REPÚBLICA DEL ECUADOR
Escuela Politécnica Nacional
"E SCIENTIA HOMINIS SALUS"

La versión digital de esta tesis está protegida por la Ley de Derechos de Autor del Ecuador.

Los derechos de autor han sido entregados a la "ESCUELA POLITÉCNICA NACIONAL" bajo el libre consentimiento del (los) autor(es).

Al consultar esta tesis deberá acatar con las disposiciones de la Ley y las siguientes condiciones de uso:

- Cualquier uso que haga de estos documentos o imágenes deben ser sólo para efectos de investigación o estudio académico, y usted no puede ponerlos a disposición de otra persona.

- Usted deberá reconocer el derecho del autor a ser identificado y citado como el autor de esta tesis.

- No se podrá obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.

El Libre Acceso a la información, promueve el reconocimiento de la originalidad de las ideas de los demás, respetando las normas de presentación y de citación de autores con el fin de no incurrir en actos ilegítimos de copiar y hacer pasar como propias las creaciones de terceras personas.

Respeto hacia sí mismo y hacia los demás.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DOMÓTICO PARA PERMITIR EL CONTROL CENTRALIZADO DE DISPOSITIVOS DENTRO DEL HOGAR CONTROLADO POR FPGA UTILIZANDO EL PROTOCOLO X10

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

JIMA MASACHE MARÍA ELIZABETH
mariajima88@gmail.com

ZAMBRANO ESPINOSA JONATHAN HENRY
jonathancitozambrano@hotmail.com

DIRECTOR: ING. CARLOS ALFONSO HERRERA MUÑOZ
carlos.herrera@epn.edu.ec

Quito, Febrero 2015

DECLARACIÓN

Nosotros, María Elizabeth Jima Masache y Jonathan Henry Zambrano Espinosa, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____ _____

María Jima Jonathan Zambrano

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por María Elizabeth Jima Masache y Jonathan Henry Zambrano Espinosa, bajo mi supervisión.

Ing. Carlos Herrera

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

En primer lugar quiero agradecer a Dios y a Mariuxi por concederme el don de la vida y la oportunidad de cumplir esta meta, a mi familia que han estado apoyándome física y espiritualmente, a Jonathin por toda la dedicación y esfuerzo que nos ha permitido culminar el presente proyecto y al Ingeniero Carlos Herrera por su guía, sus consejos y enseñanzas en el transcurso de este proyecto.

Deseo además expresar un sincero agradecimiento a Tosqui y a mis amigos de la universidad, con quienes he vivido maravillosos momentos y quienes siempre han estado dispuestos a apoyarme con sus conocimientos, con su alegría y sobre todo por siempre estar a pesar todas las dificultades.

Finalmente deseo agradecer a la Escuela Politécnica Nacional, a mis profesores, y amig@s de la Facultad, de quienes pude aprender y crecer no sólo profesionalmente, sino además personalmente.

María Jima

Quiero agradecer en primer lugar a Dios, por llenar con bendiciones mi vida. A mi Madre y Hermana por ser mi apoyo en cada momento. A Mariita por brindarme su amistad en los buenos y malos momentos. A todos mis amig@s que me han acompañado en este largo camino, y con quienes he podido compartir y aprender, sobre todo, a Fausto, Juan Pablo, Alejandra, Jorge, José, Franklin, Christian, María Fernanda, María Gabriela, Pamela y Oswaldo quienes aportaron con su grano de arena y por quienes también es posible este trabajo.

Al Ing. Carlos Herrera, por su paciencia y ayuda constante en el desarrollo del presente proyecto. A la Escuela Politécnica Nacional y a todos mis maestros, quienes sabiamente han sabido guiarme en mi carrera.

Jonathan Zambrano

DEDICATORIAS

A mis padres y a Dianita por enseñarme el valor de la vida y brindarme su amor y su fuerza siempre.

María Jima

Este trabajo está dedicado a Dios, a mis padres, hermanos y a mi familia, por quienes he llegado a ser la persona soy. Y a todas las personas que de una manera u otra aportaron para la culminación de este proyecto.

Jonathan Zambrano

ÍNDICE DE CONTENIDOS

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS 1

1.1 DOMÓTICA 1

1.1.1 HISTORIA 1

1.1.2 DEFINICIÓN 1

1.1.3 APLICACIONES 2

1.1.4 ELEMENTOS DE UN SISTEMA DOMÓTICO 3

1.1.5 TIPOS DE ARQUITECTURA DOMÓTICAS 5

1.1.6 PROTOCOLOS DE COMUNICACIÓN 8

1.2 PROTOCOLO X10 10

1.2.1 HISTORIA 10

1.2.2 TIPOS DE DISPOSITIVOS 10

1.2.3 PRINCIPIOS DEL PROTOCOLO X10 11

1.2.4 TRANSMISIÓN EN LA RED ELÉCTRICA 12

1.2.5 MENSAJE X10 13

1.2.6 FORMATO DE LA TRANSMISIÓN 15

1.2.7 FUNCIONES X10 18

1.3 DISPOSITIVOS LÓGICO PROGRAMABLES 20

1.3.1 FPGA 23

1.3.1.1 Arquitectura General de un FPGA 24

1.3.1.1.1 Los Bloques Lógicos Configurables (CLBs) 25

1.3.1.1.2 Los Bloques Configurables de Entrada/Salida (IOBs) 26

1.3.1.1.3 Matriz de Interconexiones Programables 26

1.4 LA PLATAFORMA SPARTAN 3 DE XILINX 27

1.4.1 ARQUITECTURA DE LA FPGA SPARTAN 3 28
1.4.2 TARJETA XILINX SPARTAN-3E STARTER KIT 30
1.4.2.1 Puerto Serial RS-232 32
1.5 PROGRAMACIÓN DE UN FPGA 33
1.5.1 VHSIC Hardware Description Language (VHDL) 34
1.5.1.1 Flujo de Diseño en VHDL 35
CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE SISTEMA DOMÓTICO 39
2.1 DESCRIPCIÓN DEL SISTEMA 39
2.2 DISEÑO DEL HARDWARE 42
2.2.1 DISEÑO DEL CIRCUITO DETECTOR DE CRUCES POR CERO 42
2.2.2 DISEÑO DEL CIRCUITO TRANSMISOR X10 44
2.2.3 DISEÑO DEL CIRCUITO RECEPTOR X10 49
2.2.4 DISEÑO DEL CIRCUITO DE CONTROL DE ILUMINACIÓN 56
2.2.5 DISEÑO DEL CIRCUITO DE CONTROL DE PERSIANAS Y PUERTAS .. 60
2.2.6 DISEÑO DEL CIRCUITO PARA LA INTERFAZ DE COMUNICACIÓN SERIAL 63
2.2.7 DISEÑO DE LOS CIRCUITOS DE LAS FUENTES DE ALIMENTACIÓN . 65
2.2.7.1 Diseño de la fuente de alimentación de los circuitos transmisor y receptores 65
2.2.7.2 Diseño de la fuente de alimentación de los circuitos actuadores. 66
2.3 DISEÑO DE LOS PROGRAMAS DEL PROTOTIPO 67
2.3.1 DISEÑO DEL PROGRAMA PARA EL CONTROL DOMÓTICO EN EL FPGA 67
2.3.1.1 Diseño del programa para el control de iluminación, accesos y persianas en tiempo real. 68

2.3.1.1.1 Descripción del diseño del programa para el control de iluminación, accesos y persianas en tiempo real. 68

2.3.1.2 Diseño del programa para generar el reloj para el usuario 81

2.3.1.2.1 Descripción del programa para generar el reloj para el usuario 82

2.3.1.3 Diseño del programa para el control programado de iluminación y persianas 87

2.3.1.3.1 Descripción del programa para el control programado de iluminación y persianas. 88

2.3.2 DISEÑO DE LOS PROGRAMAS DE TRANSMISIÓN Y RECEPCIÓN DE LOS CÓDIGOS X10 105

2.3.2.1 Asignación de espacios de memoria 106

2.3.2.1.1 Palabra de Configuración 107

2.3.2.1.2 Asignación de las direcciones de memoria de los registros a utilizar 110

2.3.2.2 Subrutinas de inicio 113

2.3.2.2.1 Subrutinas de inicio comunes en la transmisión y recepción de códigos X10 113

2.3.2.2.2 Subrutinas de inicio para el Transmisor 122

2.3.2.3 Rutina de Interrupciones 127

2.3.3 DISEÑO DEL PROGRAMA PARA EL ACTUADOR DE ILUMINACIÓN .. 144

2.3.4 DISEÑO DEL PROGRAMA PARA EL ACTUADOR DE ACCESOS Y PERSIANAS 152

2.4 DISEÑO DE LA INTERFAZ DE USUARIO 155

2.4.1 DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO 155

2.4.1.1 LABVIEW (Laboratory Virtual Instrument Engineering Workbench) .. 155

2.4.1.2 Desarrollo de la interfaz gráfica de usuario en Labview 157

2.4.1.2.1 Tiempo Real 158
2.4.1.2.2 Programadas 160
2.4.1.2.3 Reloj 162
2.5 CONSTRUCCIÓN DEL PROTOTIPO 163
CAPÍTULO 3: PRUEBAS DE FUNCIONAMIENTO Y RESULTADOS 169
3.1 PRUEBAS REALIZADAS A LA SEÑAL X10 169
3.1.1 GENERACIÓN DE LA SEÑAL DE CONTROL EN EL FPGA 169
3.1.2 TRANSMISIÓN DE LA SEÑAL X10 HACIA LA RED ELÉCTRICA 170
3.1.3 RECEPCIÓN DE LA SEÑAL X10 DESDE LA RED ELÉCTRICA 171
3.1.4 TRANSMISIÓN DEL CÓDIGO DE FUNCIÓN X10 HACIA EL ACTUADOR 172
3.2 PRUEBA INTEGRAL DEL SISTEMA DOMÓTICO EN TIEMPO REAL 173
3.2.1 PRUEBAS DEL CONTROL DE ILUMINACIÓN 174
3.2.1.1 Encendido de Luminarias 174
3.2.1.1.1 Procedimiento para comprobar el encendido de luminarias 174
3.2.1.1.2 Resultados de las pruebas para el encendido de luces 175
3.2.1.2 Apagado de Luminarias 176
3.2.1.2.1 Procedimiento para comprobar el apagado de luces 177
3.2.1.2.2 Resultados de las pruebas para el apagado de luces 177
3.2.1.3 Dimerización de Luminarias 179
3.2.1.3.1 Procedimiento para comprobar el aumento de la intensidad de las luces 179
3.2.1.3.2 Procedimiento para comprobar la disminución de la intensidad de las luces 181

3.2.1.3.3 Resultados de la pruebas para el aumento y disminución de la intensidad de luz de las luminarias..... 183
3.2.2 PRUEBAS DEL CONTROL DE PERSIANAS..... 185
3.2.2.1 Abrir totalmente las persianas..... 186
3.2.2.1.1 Procedimiento para comprobar que se abren totalmente las persianas..... 186
3.2.2.1.2 Resultados de las pruebas para abrir totalmente las persianas.. 187
3.2.2.2 Cerrar totalmente las persianas..... 188
3.2.2.2.1 Procedimiento para comprobar que se cierran totalmente las persianas..... 189
3.2.2.2.2 Resultados de las pruebas para cerrar totalmente las persianas 190
3.2.2.3 Abrir parcialmente las persianas..... 191
3.2.2.3.1 Procedimiento para comprobar que se abren parcialmente las persianas..... 191
3.2.2.3.2 Resultados de la pruebas para abrir parcialmente las persianas 193
3.2.2.4 Cerrar parcialmente las persianas..... 194
3.2.2.4.1 Procedimiento para comprobar que se cierran parcialmente las persianas..... 194
3.2.2.4.2 Resultados de la pruebas para cerrar parcialmente las persianas 196
3.2.3 PRUEBAS DEL CONTROL DE ACCESOS..... 197
3.2.3.1 Abrir las Puertas..... 198
3.2.3.1.1 Procedimiento para comprobar que se abren las puertas..... 198
3.2.3.1.2 Resultados de las pruebas para abrir las puertas..... 199
3.2.3.2 Cerrar las Puertas..... 200
3.2.3.2.1 Procedimiento para comprobar que se cierran las puertas..... 200

3.2.3.2.2 Resultados de las pruebas para cerrar las puertas 201

3.3 PRUEBA INTEGRAL DEL SISTEMA DOMÓTICO EN TIEMPO PROGRAMADO 204

3.3.1 PROCEDIMIENTO DE LA PRUEBA EN TIEMPO PROGRAMADO 204

3.3.2 RESULTADOS DE LAS PRUEBAS DEL SISTEMA DOMÓTICO EN TIEMPO PROGRAMADO 208

CAPÍTULO 4: COSTOS REFERENCIALES DEL PROTOTIPO 213

4.1 COMPARACIÓN CON EL SISTEMA DOMÓTICO DE LA EMPRESA LUTRON 219

4.2 COMPARACIÓN CON EL SISTEMA DOMÓTICO COMERCIAL X10 219

CAPÍTULO 5: CONCLUSIONES Y RECOMENDACIONES 221

5.1 CONCLUSIONES 221

5.2 RECOMENDACIONES ..... **¡Error! Marcador no definido.**

REFERENCIAS BIBLIOGRÁFICAS 225

ANEXOS 231

ÍNDICE DE FIGURAS

Figura 1.1: Elementos de un sistema domótico 5
Figura 1.2: Arquitectura Centralizada 6
Figura 1.3: Arquitectura Descentralizada 6
Figura 1.4: Arquitectura Distribuida 7
Figura 1.5: Arquitectura Mixta 8
Figura 1.6: Cruce por cero 11
Figura 1.7: Transmisión en un sistema trifásico 12
Figura 1.8: Código de Inicio 16
Figura 1.9: Bloque de mensajes X10 16
Figura 1.10: Ciclos de Pausa 16
Figura 1.11: Diagrama de bloques de una PAL (Programmable Logic Array) 21
Figura 1.12: Estructura Básica de una GAL 22
Figura 1.13: Estructura Básica de un CPLD 22
Figura 1.14: Arquitectura Básica de un FPGA 24
Figura 1.15: Estructura de un Bloque Lógico Configurable 25
Figura 1.16: Estructura de una Look-Up Table de n entradas 26
Figura 1.17: Arquitectura de la Spartan 3 29
Figura 1.18: Vista superior de la tarjeta Spartan-3E Starter Kit y sus componentes 31
Figura 1.19: Puertos DCE y DTE de la Spartan-3E Starter Kit 32
Figura 1.20: Puertos Seriales RS-232 33
Figura 1.21: Flujo de Diseño en VHDL 38
Figura 2.1: Configuración del Sistema Domótico 41
Figura 2.2: Detector de cruces por cero 44

Figura 2.3: Circuito de transmisión de los mensajes X10 48

Figura 2.4: Circuito integrado CD4069 50

Figura 2.5: Voltaje de salida en función del voltaje de entrada 50

Figura 2.6: Comportamiento del circuito detector de envolvente 54

Figura 2.7: Circuito de recepción de los mensajes X10 55

Figura 2.8: Control de la señal eléctrica mediante PWM 56

Figura 2.9: Triac BT136 57

Figura 2.10: Circuito actuador de iluminación 59

Figura 2.11: Motor a pasos 28BYJ-48 60

Figura 2.12: Secuencia de paso doble para un motor a pasos 62

Figura 2.13: Circuito actuador de puertas y persianas 63

Figura 2.14: Circuito de la interfaz de comunicación serial usando un MAX232 64

Figura 2.15: Fuente DC 66

Figura 2.16: Fuente con transformador 67

Figura 2.17: Diagrama de flujo del programa de control de iluminación, accesos y persianas en tiempo real 69

Figura 2.18: Diseño del reloj de recepción 72

Figura 2.19: Diseño del reloj de transmisión 74

Figura 2.20: Formato de recepción asíncrono 75

Figura 2.21: Comportamiento de las señales de recepción del programa 76

Figura 2.22: Formato de transmisión asíncrono 80

Figura 2.23: Diagrama de flujo del diseño del reloj 82

Figura 2.24: Diseño del reloj del usuario 84

Figura 2.25: Diagrama de flujo del control programado de iluminación y persianas 88

Figura 2.26: Programación de las luces en la interfaz de usuario 94

Figura 2.27: Trama *"LUCES"* 95

Figura 2.28: Interfaz de usuario y su correspondiente trama "*LUCES"* 95

Figura 2.29: Programación de las persianas en la interfaz de usuario 96

Figura 2.30: Trama "*CORTINAS"* 97

Figura 2.31: Interfaz de usuario y su correspondiente trama "*Cortinas"* 97

Figura 2.32: Estructura del programa principal de transmisión y recepción X10 106

Figura 2.33: Asignación de espacios de memoria 107

Figura 2.34: Palabra de configuración del transmisor X10 109

Figura 2.35: Palabra de configuración del receptor X10 109

Figura 2.36: Banderas del registro X10Flags 111

Figura 2.37: Subrutinas de inicio 113

Figura 2.38: Subrutinas InicZeroXDetect 114

Figura 2.39: Bit de selección de flanco del OPTION_REG 114

Figura 2.40: Bit de habilitación de la interrupción externa del INTCON 115

Figura 2.41: Subrutinas InicX10Variables 115

Figura 2.42: Subrutina InicSerial 117

Figura 2.43: Configuración del registro RCSTA para recepción serial 119

Figura 2.44: Configuración del registro PIE1 para habilitar la interrupción por recepción 119

Figura 2.45: Configuración del registro TXSTA para transmisión serial 120

Figura 2.46: Subrutina EnableInterrupts 121

Figura 2.47: Configuración del registro INTCON para habilitar las interrupciones globales y periféricas 121

Figura 2.48: Subrutina InicTimer0 122

Figura 2.49: Configuración del registro OPTION_REG para inicializar el timer 0 ... 123

Figura 2.50: Configuración del registro INTCON para habilitar el timer 0 124

Figura 2.51: Subrutina InicPWM..... 124

Figura 2.52: Configuración del registro CCP1CON para configurar el timer 2 como PWM 127

Figura 2.53: Configuración del registro T2CON para habilitar el timer 2 y su preescalador..... 127

Figura 2.54: Rutina de interrupciones de un transmisor X10..... 128

Figura 2.55: Rutina de interrupciones de un receptor X10 129

Figura 2.56: Subrutina para interrupción serial..... 130

Figura 2.57: Subrutina PTx..... 131

Figura 2.58: Subrutina InitX10Tx..... 132

Figura 2.59: Registros temporales antes de rotar 132

Figura 2.60: Registros temporales después de rotar..... 133

Figura 2.61: Subrutina RotateTxData..... 133

Figura 2.62: Subrutina ZeroCrossing para la transmisión 134

Figura 2.63: Subrutina TransmitX10Bit 135

Figura 2.64: Subrutina TransmitX10Tx..... 137

Figura 2.65: Subrutina Inic2ndTx 138

Figura 2.66: Subrutina Inic2ndBloque 139

Figura 2.67: Subrutina ZeroCrossing para la recepción 140

Figura 2.68: Subrutina ReceiveX10Bit 140

Figura 2.69: Subrutina X10Rx 141

Figura 2.70: Subrutina para guardar códigos X10..... 142

Figura 2.71: Subrutina ToggleInterruptEdge 142

Figura 2.72: Subrutina EndX10Envelope 143

Figura 2.73: Diagrama de flujo del programa para el actuador de iluminación 144

Figura 2.74: Diagrama de flujo de la interrupción externa 0 149

Figura 2.75: Diagrama de flujo de la interrupción externa 1 149

Figura 2.76: Diagrama de flujo de la interrupción por recepción serial 151

Figura 2.77: Diagrama de flujo del programa para el control de puertas y persianas 153

Figura 2.78: Panel Frontal de LabVIEW 156

Figura 2.79: Diagrama de Bloques de LabVIEW 156

Figura 2.80: Paletas de LabVIEW 157

Figura 2.81: Pantalla para seleccionar el tipo de función que se desea ejecutar ... 158

Figura 2.82: Ventana de control en Tiempo Real 160

Figura 2.83: Ventana de control programado 161

Figura 2.84: Ventana del control del Reloj 162

Figura 2.85: Prototipo del sistema domótico 164

Figura 2.86: Instalaciones eléctricas internas del sistema domótico 165

Figura 2.87: Etiquetado del prototipo 166

Figura 2.88: Conexión de los actuadores a la instalación eléctrica 167

Figura 2.89: Conexión completa del sistema domótico 168

Figura 3.1: Señal de control generada en el FPGA para encender la luminaria de la sala 170

Figura 3.2: Señal X10 colocada en la red eléctrica por parte del transmisor X10 .. 170

Figura 3.3: Señal X10 recibida desde la red eléctrica por parte del receptor X10 .. 171

Figura 3.4: Señal X10 regenerada en el circuito receptor X10 172

Figura 3.5: Código de función X10 transmitido hacia el actuador de luz de la sala 172

Figura 3.6: Uso de la interfaz gráfica de usuario para encender la luminaria 175

Figura 3.7: Luminarias de la vivienda encendidas..... 176

Figura 3.8: Uso de la interfaz gráfica de usuario para apagar una luminaria 177

Figura 3.9 Luminarias de la vivienda apagadas 178

Figura 3.10: Uso de la interfaz gráfica de usuario para aumentar la intensidad de luz de una luminaria..... 180

Figura 3.11: Control manual del aumento de la intensidad de luz de una luminaria 181

Figura 3.12: Uso de la interfaz gráfica de usuario para disminuir la intensidad de luz de un foco..... 182

Figura 3.13: Control manual para la disminución de la intensidad de luz de una luminaria..... 183

Figura 3.14: Luminarias de la vivienda con diferentes niveles de intensidad de luz185

Figura 3.15: Uso de la interfaz gráfica de usuario para abrir una persiana 187

Figura 3.16: Persianas de la vivienda totalmente abiertas 188

Figura 3.17: Uso de la interfaz gráfica de usuario para cerrar totalmente una persiana 189

Figura 3.18: Persianas de la vivienda totalmente cerradas 191

Figura 3.19: Uso de la interfaz gráfica para abrir parcialmente una persiana 192

Figura 3.20: Persianas de la vivienda parcialmente abiertas 194

Figura 3.21: Uso de la interfaz gráfica para cerrar parcialmente una persiana 195

Figura 3.22: Persianas de la vivienda parcialmente cerradas 197

Figura 3.23: Uso de la interfaz gráfica para abrir una puerta 199

Figura 3.24: Puertas de la vivienda abiertas 200

Figura 3.25: Uso de la interfaz gráfica para cerrar una puerta 201

Figura 3.26: Puertas de la vivienda cerradas 202

Figura 3.27 Resultados de la prueba de funcionamiento del sistema domótico integrado en tiempo real 203

Figura 3.28 Interfaz gráfica de usuario para seleccionar las tareas que deben ser ejecutadas 205

Figura 3.29 Interfaz gráfica de usuario para configurar el tiempo de ejecución de las tareas 206

Figura 3.30 Interfaz gráfica de usuario para configurar las tareas y horario programados 207

Figura 3.31 Programación del control total del sistema 208

Figura 3.32: Ejecución del control de luminarias y persianas a la hora programada de encendido 209

Figura 3.33: Ejecución del control de luminarias y persianas a la hora programada de apagado 209

Figura 3.34 Reloj del sistema domótico 210

ÍNDICE DE TABLAS

Tabla 1.1: Códigos de Casa 13

Tabla 1.2: Códigos de Unidad 14

Tabla 1.3: Códigos de Función 14

Tabla 1.4: Generación de tarjetas Spartan 3 28

Tabla 2.1: Elementos y funciones del sistema domótico 40

Tabla 2.2: Instrucciones para el sistema de control domótico 77

Tabla 2.3: Códigos X10 del prototipo de sistema domótico 79

Tabla 3.1: Resultados de las pruebas para el encendido de luminarias 175

Tabla 3.2: Resultados de las pruebas para apagar las luminarias 178

Tabla 3.3: Resultados de las pruebas para cambiar la intesidad de luz en las luminarias 184

Tabla 3.4: Resultados de las pruebas para abrir totalmente las persianas 187

Tabla 3.5: Resultados de las pruebas para cerrar totalmente las persianas 190

Tabla 3.6: Resultados de las pruebas para abrir parcialmente las persianas..... 193

Tabla 3.7: Resultados de las pruebas para cerrar parcialmente las persianas..... 196

Tabla 3.8: Resultados de las pruebas para abrir las puertas..... 199

Tabla 3.9: Resultados de las pruebas para cerrar las puertas..... 201

Tabla 3.10: Datos medidos de corriente y potencia del sistema domótico 211

RESUMEN

El objetivo del presente proyecto fue ofrecer al mercado un sistema domótico que utilice un dispositivo lógico programable (FPGA) como controlador del sistema, de tal forma que se aprovechen sus propiedades de gran capacidad de procesamiento y adaptación en beneficio de los usuarios, aportando de esta manera al uso de nuevas tecnologías para desarrollo de la domótica en el país.

Para cumplir este propósito, se procedió a la construcción de un prototipo que simula un hogar, donde se instaló los circuitos de transmisión, recepción y control del mismo. El prototipo es controlado por la tarjeta Spartan 3E Starter Board de Xilinx, y se utilizó el protocolo X10, para enviar la información de control a través de la línea eléctrica, de tal manera que no sea necesaria la instalación de nuevo cableado para el control del sistema domótico. Este protocolo fue programado tanto en el FPGA como en cada uno de los actuadores para controlar la iluminación, accesos y persianas.

Para que el usuario pueda controlar el sistema se procedió a la creación de una HMI (Interfaz Humano-Máquina), de tal forma que se pueda controlar de manera “amigable” el prototipo, mediante el envío de las instrucciones mediante la tarjeta de control basada en el FPGA. Los datos son transmitidos vía comandos del protocolo X10.

Las pruebas diseñadas para corroborar la validez del diseño implementado, y una vez que se corrigieron problemas de programación de los PWM que controlan los actuadores, dieron como resultado un 100% de confiabilidad toda vez que se enviaron los comandos para controlar tanto la iluminación, como las persianas y el control de accesos. El prototipo resultó con un costo competitivo (US 1200) con respecto a productos comerciales que se ofertan en la actualidad en el país.

PRESENTACIÓN

Actualmente el mercado inmobiliario presenta un gran desarrollo a nivel local, incorporando sistemas inteligentes que aumentan el confort, seguridad y ahorro energético de las viviendas. El presente proyecto de titulación tiene el propósito de proponer una nueva alternativa de control mediante un sistema basado en un FPGA, y X10 como el protocolo de comunicaciones que nos permite enviar la información de control a través de la línea eléctrica, para evitar el costo y demoras de la instalación de nuevo cableado para el sistema domótico. El trabajo que se realizó se lo reporta aquí, en cinco capítulos en los que se describen los fundamentos teóricos importantes para construir el sistema domótico, el diseño y la implementación del sistema, las pruebas y resultados obtenidos luego de la implementación, el costo del prototipo construido y finalmente las conclusiones y recomendaciones a las que se llegaron una vez que se culminó con todo el proyecto.

El primer capítulo presenta una descripción general sobre los sistemas domóticos, sus características, arquitectura y diferentes elementos que los conforman, enfatizando principalmente en el protocolo X10. Además se realiza una breve descripción de los dispositivos lógico programables, detallando de manera especial las características y programación de la tarjeta Spartan 3E Starter Board de Xilinx, que es la tarjeta FPGA que se utiliza para la implementación del prototipo correspondiente al proyecto.

En el segundo capítulo se detalla el diseño e implementación del sistema domótico, para lo cual se realiza en primer lugar una descripción sobre el diseño de todo el hardware empleado en el sistema, a continuación se describe el diseño de toda la programación necesaria para poder controlar el sistema, y finalmente se detalla la construcción final del sistema domótico completo.

En el tercer capítulo se detallan cada una de las pruebas realizadas y los resultados obtenidos en cada de una de ellas, luego de finalizar la implementación del sistema domótico.

En el cuarto capítulo se muestran los costos referenciales del prototipo implementado y su comparación con los costos de otros prototipos domóticos existentes actualmente en el mercado.

Finalmente en el quinto capítulo se presentan las conclusiones a las que se llegó una vez finalizada la construcción del prototipo, y las recomendaciones sugeridas para éste y para futuros proyectos de domótica.

CAPÍTULO 1

FUNDAMENTOS TEÓRICOS

1.1 DOMÓTICA

1.1.1 HISTORIA [24]

La domótica da sus primeros pasos en la década de los años 70, siendo la tecnología X-10 la pionera en desarrollar dispositivos para la automatización del hogar. Más adelante la comunidad internacional se fue interesando en el desarrollo de la casa ideal, empezando a ensayar con diversos electrodomésticos y dispositivos automáticos para el hogar. En Estados Unidos fue donde se instalaron los primeros sistemas domóticos, los cuales permitían la regulación de la temperatura en oficinas.

A finales de la década de los 80 y principios de los 90 aparecieron los denominados “edificios inteligentes”, los cuales se caracterizan por incorporar sistemas de cableado estructurado, facilitando de esta forma el transporte de datos, voz y la conexión de dispositivos de control y seguridad.

Posteriormente toda esta automatización disponible en edificios inteligentes, se ha ido implementado también en viviendas, dando lugar a lo que actualmente se conoce como domótica, cuya finalidad es satisfacer las necesidades tecnológicas de los usuarios de los hogares modernos.

1.1.2 DEFINICIÓN [18]

El sistema domótico es un conjunto de sistemas que permiten el control y automatización de una vivienda y está conformado por una unidad central que controla y gestiona las actividades de todos los elementos de dicho sistema.

El usuario puede realizar el control de su vivienda de forma preprogramada o en tiempo real, ya sea desde un ambiente interno o externo a la misma, permitiéndole de esta manera un mejor aprovechamiento del tiempo y aumento de la comodidad.

Es por ello que el objetivo principal de un sistema domótico es brindar al usuario confort, seguridad, ahorro de energía y facilidades de comunicación dentro del hogar.

1.1.3 APLICACIONES [13], [18], [25]

En la actualidad el crecimiento y la influencia de la tecnología, han hecho que los sistemas domóticos incorporen informática y nuevas tecnologías de la información, permitiendo así el desarrollo de numerosas aplicaciones que se describen a continuación.

- ***Seguridad:*** Su principal objetivo es proteger a las personas y sus bienes mediante la implementación de alarmas de prevención de intrusos y robo, simulación de presencia o alarmas de detección de gases, humo, inundaciones, etc.

- ***Gestión de la energía:*** Su principal objetivo es racionalizar el consumo energético en función de las necesidades del usuario. Las aplicaciones más destacadas son programación de la climatización, programación de equipos domésticos, ahorro energético mediante el manejo optimizado de las cargas eléctricas del hogar.

- ***Sistemas de confortabilidad:*** Su principal objetivo es la simplificación de algunas tareas incrementando de esta manera el confort del usuario dentro de su hogar. Las aplicaciones más destacadas son automatización y regulación de luminosidad, control remoto de luminarias y dispositivos, gestión de persianas y toldos, sistemas de riego automatizado, etc.

- ***Comunicaciones:*** Su principal objetivo es permitir el intercambio de información con otras personas y con los dispositivos del hogar. Algunas aplicaciones de comunicaciones son los intercomunicadores, la incorporación de internet y de la red telefónica para el control eléctrico.

Todas estas aplicaciones pueden ser ejecutadas de manera local o remota dependiendo del escenario en el cual se encuentre el usuario. Para poder llevar a cabo una tarea se puede recurrir a distintas herramientas que facilitan la activación de la misma, estas herramientas pueden ser:

- Pulsación de un interruptor o botón de un control.
- Un mensaje de voz.
- Un mensaje de texto.
- Pulsación de una ventana de internet.
- El ingreso o salida de una persona en una habitación.

El transporte de la información que el usuario envía al dispositivo se efectúa a través de medios de transmisión guiados y no guiados. A continuación se describen los principales medio utilizados:

- ***Par trenzado, cable coaxial ó fibra óptica:*** Estos medios conforman el denominado cableado propio, puesto que pueden ser utilizados para el intercambio de información exclusiva de los sistemas domóticos.

- ***Red eléctrica, red telefónica o redes de datos:*** Estos medios conforman el denominado cableado compartido, puesto que se envía la información de los sistemas domóticos a través de redes ya existentes y que desempeñan más de una función dentro de la vivienda.

- ***Radiofrecuencia o infrarrojo:*** Estos medios permiten el envío de la información usando comunicación inalámbrica, aumentando de esta manera la flexibilidad dentro del hogar.

1.1.4 ELEMENTOS DE UN SISTEMA DOMÓTICO [13], [17], [18]

En la actualidad existen una gran variedad de sistemas domóticos, los cuales se diseñan de acuerdo a la topología de la casa y del estilo de vida de las personas que allí habitan. Todos los sistemas domóticos están conformados por uno o varios

dispositivos que permiten automatizar el hogar. Estos elementos se describen a continuación:

- ***Controlador:*** Es la parte central de todo el sistema domótico, se encarga de recibir la información proveniente de los sensores, procesarla y emitir una respuesta acorde a la programación realizada para satisfacer las necesidades del usuario.
- ***Actuador:*** Es un dispositivo que recibe una orden proveniente del controlador y la ejecuta sobre un aparato o sistema del hogar. Por ejemplo se encarga del encendido y apagado de luces, subida y bajada de cortinas, apertura y cierre de puertas.
- ***Sensor:*** Es un dispositivo que se encuentra monitoreando y recogiendo información del entorno. Dicha información la transmite al controlador para que lleve a cabo el trabajo correspondiente. Con la finalidad de satisfacer las diversas necesidades de los usuarios, en el mercado se puede encontrar sensores de agua, humo, temperatura, humedad, iluminación, etc.
- ***Bus:*** Es el medio de transmisión a través del cual se transporta la información entre los diferentes dispositivos que conforman el sistema domótico.
- ***Interfaz:*** Es un dispositivo que le permite al usuario interactuar con el sistema domótico, por ejemplo el teléfono celular, una pantalla, un control remoto, computadora portátil, internet, etc.

A continuación en la figura 1.1, se muestran los elementos que conforman un sistema domótico:

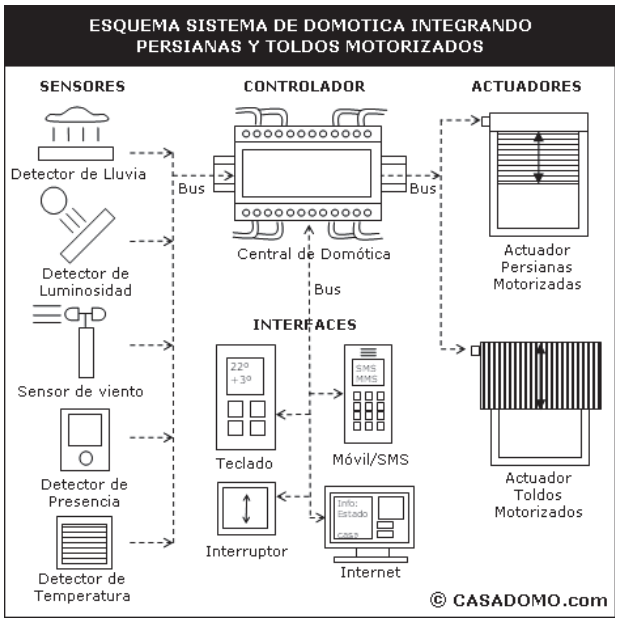


Figura 1.1: Elementos de un sistema domótico [40]

En este ejemplo se puede visualizar la forma como trabajan los dispositivos de un sistema domótico. Se visualiza que la información puede ser generada por los sensores instalados en la vivienda ó por el usuario. El controlador recibe dicha información y genera una respuesta acorde a lo que el usuario desea y la transmite a los actuadores, los cuales ejecutan la tarea ordenada por el controlador.

1.1.5 TIPOS DE ARQUITECTURA DOMÓTICAS [13], [17], [18]

En domótica se pueden presentar varios tipos de arquitectura dependiendo de los elementos que se utilicen y la forma como se distribuyen y trabajan en el sistema domótico del hogar. Las principales arquitecturas son las siguientes:

- ***Arquitectura Centralizada:*** Se caracteriza por tener un único controlador que recibe toda la información proveniente de las entradas, procesa dicha información y envía las órdenes a los actuadores correspondientes. En esta arquitectura si el controlador se daña, todo el sistema deja de funcionar. Sin embargo presenta varias ventajas, por ejemplo es una arquitectura de fácil instalación y utilización, los dispositivos empleados son del tipo universal y

tiene un costo moderado. A continuación, en la figura 1.2, se observa un diagrama de flujo que describe el comportamiento de los sistemas domóticos con arquitectura centralizada.

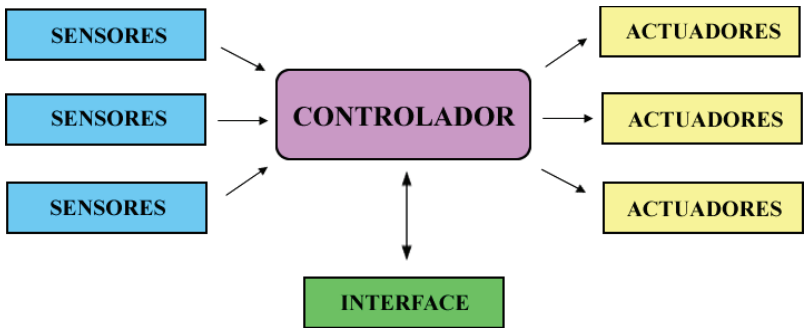


Figura 1.2: Arquitectura Centralizada

- ***Arquitectura Descentralizada:*** Se caracteriza por tener varios controladores interconectados, permitiéndoles así comunicarse entre ellos y con sus respectivos actuadores e interfaces. A continuación, en la figura 1.3, se observa un diagrama de flujo que describe el comportamiento de los sistemas domóticos con arquitectura descentralizada.

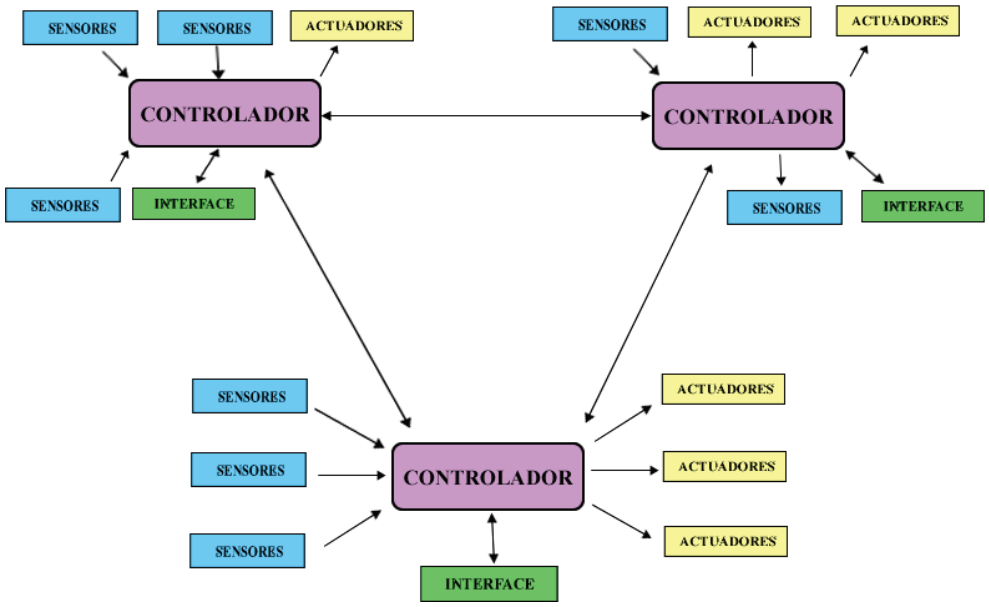


Figura 1.3: Arquitectura Descentralizada

- ***Arquitectura Distribuida:*** Se caracteriza porque cada uno de los elementos del sistema comparten la misma línea de comunicación y además actúan de forma independiente debido a que pueden realizar las funciones de un actuador/sensor y de un controlador dependiendo de la circunstancia en la que se encuentre determinado elemento. A continuación, en la figura 1.4, se observa un diagrama de flujo que describe el comportamiento de los sistemas domóticos con arquitectura distribuida.

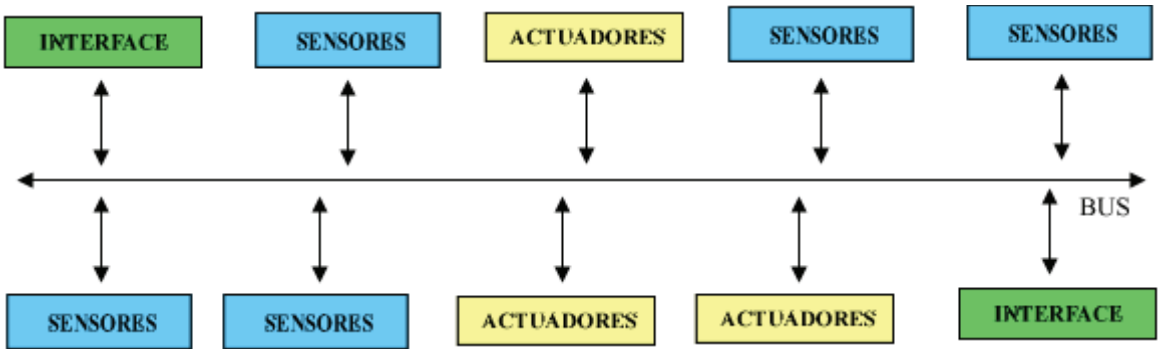


Figura 1.4: Arquitectura Distribuida

- ***Arquitectura Mixta:*** Se caracteriza por combinar las arquitecturas centralizada, descentralizada y distribuida en el mismo sistema. Aquí pueden existir uno o varios controladores que no necesariamente están interconectados. Los sensores, actuadores e interfaces pueden actuar como controladores ó enviarse información entre ellos sin la necesidad de recurrir a un controlador. A continuación, en la figura 1.5 se observa un diagrama de flujo, que describe el comportamiento de los sistemas domóticos con arquitectura mixta.

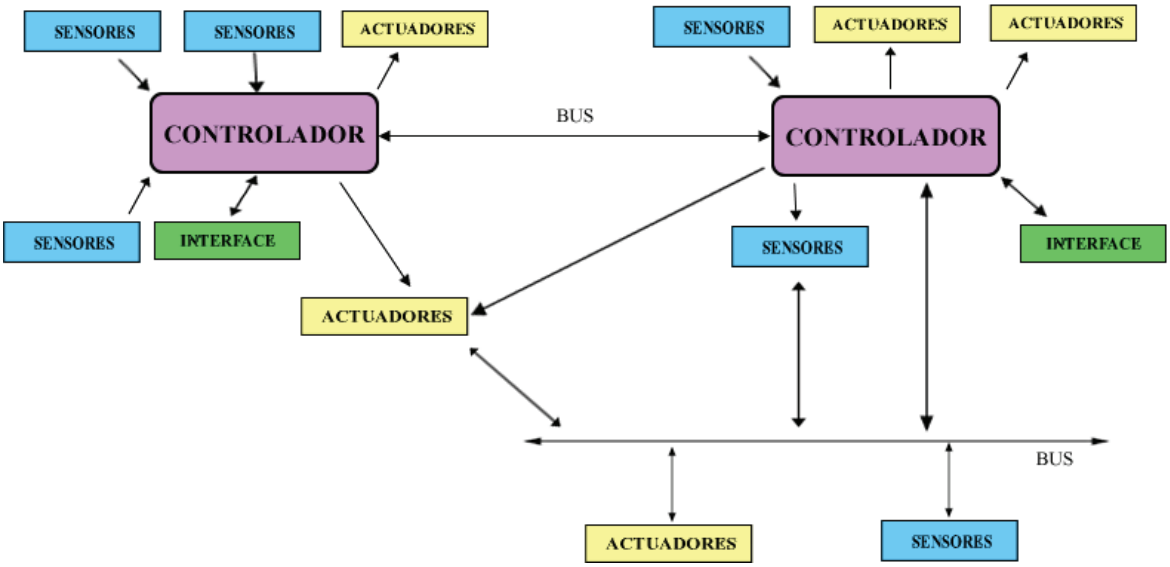


Figura 1.5: Arquitectura Mixta

Con la finalidad de satisfacer de manera óptima los requerimientos del usuario, se puede implementar cualquiera de estas arquitecturas tomando en cuenta las características físicas de la vivienda y los recursos disponibles para el sistema domótico.

1.1.6 PROTOCOLOS DE COMUNICACIÓN [22], [31]

Un protocolo de comunicación es el lenguaje que utilizan el controlador y los dispositivos del sistema domótico para comunicarse entre sí, de tal manera que todos puedan entenderse e intercambiar información. Los protocolos de comunicación se clasifican en propietarios y abiertos.

- ***Propietarios ó de fabricante:*** Son protocolos que le pertenecen específicamente a un propietario o marca, y sólo pueden ser usados por ella. Esto implica que no se pueden utilizar dispositivos de otros fabricantes en un mismo sistema, de tal forma que se protejan los derechos del fabricante, por ejemplo la marca Crestron.

- ***Estándar o abiertos:*** Son protocolos que pueden ser implementados por diferentes fabricantes ó usuarios, es decir que se pueden desarrollar diversos

productos y aplicaciones que se comuniquen libremente entre sí a pesar de haber sido creados por diferentes fabricantes. En la actualidad los protocolos abiertos más usados en domótica son los siguientes:

KNX (KONNEX)

Es un protocolo europeo de domótica que se caracteriza por ofrecer un gran número de dispositivos provenientes de diversos fabricantes en toda Europa, los cuales desempeñan exitosamente sus funciones, soportan la gran mayoría de medios de transmisión y además puede ser fácilmente acoplado a otros sistemas. La única desventaja que presentan, es el costo elevado de sus dispositivos en comparación a los otros protocolos.

Lonworks

Es un protocolo norteamericano que trabaja principalmente con aplicaciones de control en edificios, hoteles, transporte, industrias, viviendas, etc. Está homologado por normas europeas, chinas y norteamericanas. Su principal característica es la fiabilidad y robustez que ofrece, además de la interoperabilidad y que soporta la gran mayoría de medios de transmisión.

X10

Es uno de los primeros protocolos de domótica, desarrollado en Escocia y que en la actualidad es uno de los más utilizados a nivel mundial debido a su fácil instalación y cómodos costos. Se caracteriza porque utiliza la red eléctrica de 120V ya existente para transmitir la información. El presente Proyecto, se realiza empleando este protocolo, por lo cual se lo detalla en la sección 1.2 que se describe a continuación.

1.2 PROTOCOLO X10

1.2.1 HISTORIA [20], [45]

El protocolo X10 fue creado por la empresa Pico Electronics Ltd., la cual se creó en 1970 por medio de un grupo de ingenieros en Glenrothes, Escocia. Luego de varios proyectos (9 en total) se inicia en 1975 el proyecto X10 llamado así por ser el décimo proyecto de Pico Electronics Ltd. (8 Calculadoras con CI y un cambiador de registros llamados Accutrac). Luego, después de varios años diseñando la tecnología X10 en 1978, se empiezan a comercializar los productos X10 en tiendas como RadioShack (siendo en la actualidad uno de los principales distribuidores de productos X10) y tiendas Sears. Pico y BSR (British Sound Reproduction) se unieron para formar X10 Ltd., modificando el protocolo X10 para llamarlo protocolo X10-BSR, que consistía en un sistema de 16 canales, un módulo para lámparas y un módulo actuador para electrodomésticos. En 1984 la empresa BSR quiebra, por lo que Pico se separa y forma X10 (USA) Inc. De ahí en adelante todo tipo de sistemas que van desde seguridad hasta comodidad se han creado para funcionar con el protocolo X10.

El protocolo X10 es uno de los más antiguos usados en sistemas domóticos y que aún sigue en vigencia. Es un protocolo abierto que no ha sido estandarizado, por lo cual se pueden realizar modificaciones según el fabricante crea conveniente.

1.2.2 TIPOS DE DISPOSITIVOS [10]

- ***Transmisores*:** Son los dispositivos que envían los mensajes X10 por la red eléctrica.

- ***Receptores:*** Son los dispositivos que reciben los mensajes X10 desde la red eléctrica.

- ***Bidireccionales:*** Son los dispositivos que transmiten y reciben los mensajes X10 por y desde la red eléctrica.

- ***Inalámbricos:*** Son dispositivos que utilizan estándares inalámbricos para comunicarse con receptores o transmisores X10 con el objetivo de proveer los mensajes X10. Un dispositivo inalámbrico debe ir junto a un receptor ó transmisor por lo cual, no puede controlar directamente una carga o transmitir los mensajes X10 en la red eléctrica.

1.2.3 PRINCIPIOS DEL PROTOCOLO X10 [48]

El protocolo X10 es un protocolo de comunicaciones que usa la tecnología PLC (Power Line Communication), la cual, utiliza la línea eléctrica como medio de transmisión de sus señales para el control remoto de dispositivos. Para realizar la transmisión de las señales X10, se aprovecha los cruces por cero de la señal de 60 Hz como modo de sincronización de los dispositivos X10 (Figura 1.6). De esta manera, los dispositivos saben cuándo transmitir o recibir las señales X10. La representación de 1L es mediante la presencia de la señal portadora de 120 KHz por un milisegundo después del cruce por cero de la señal eléctrica, y un 0L se representa por la ausencia de esta portadora.

Los dispositivos X10 utilizan los puntos de cruce por cero de la señal eléctrica para sincronizarse, y buscan transmitir y recibir dentro del primer milisegundo después del cruce por cero.

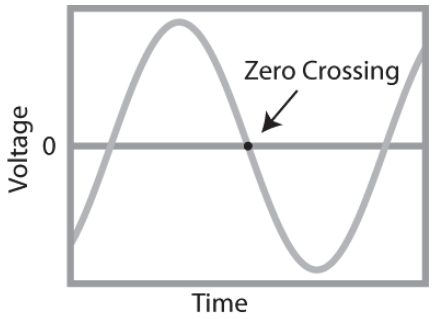


Figura 1.6: Cruce por cero [21]

1.2.4 TRANSMISIÓN EN LA RED ELÉCTRICA [13]

El formato de transmisión de los bits por la red eléctrica se da en cada ciclo de la señal eléctrica (2 cruces por cero). Los bits en X10 se transmiten junto con su complemento, es decir, que la transmisión se hace con el bit en un cruce por cero y su complemento en el otro cruce por cero. Por lo cual, un 0L se representa por la transmisión de un 0 en el primer cruce por cero y de un 1 por el segundo cruce por cero, y un 1L se representa por la transmisión de un 1 en el primer cruce por cero y de un cero en el segundo cruce por cero. Siendo un 1 la presencia de la señal portadora y en 0 la ausencia de la señal portadora. Por lo tanto, cada bit de información digital a ser transmitido necesita un ciclo completo de la señal eléctrica.

Para asegurar que los mensajes X10 se transmitan por todo el sistema eléctrico, el protocolo X10 provee dos transmisiones adicionales después del cruce por cero de la señal eléctrica, de tal forma que las señales puedan ser receptadas por dispositivos conectados en cualquier fase de un sistema trifásico (Figura 1.7).

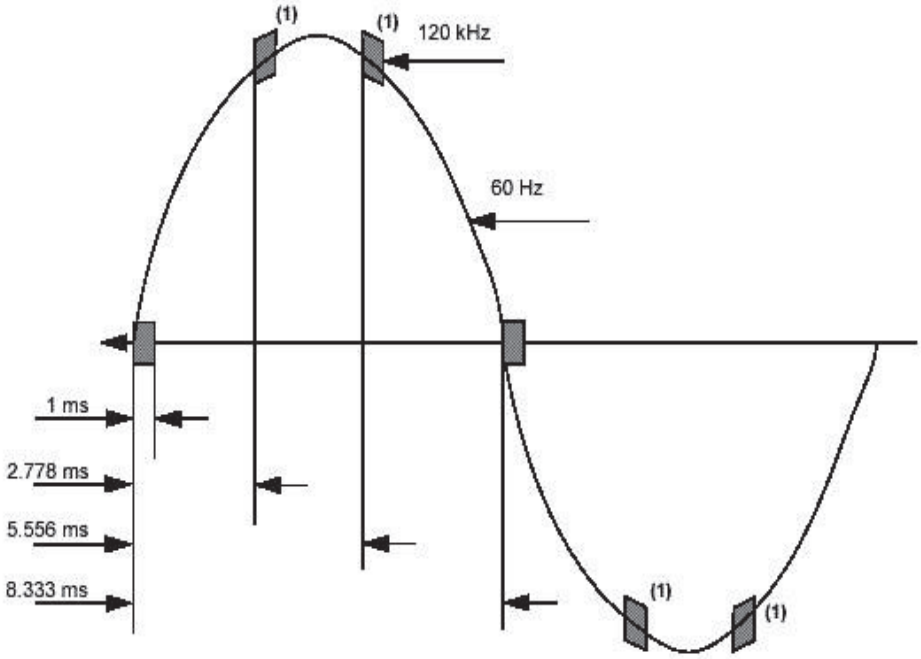


Figura 1.7: Transmisión en un sistema trifásico [38]

1.2.5 MENSAJE X10 [9]

Para la comunicación entre dispositivos X10, se utiliza un formato de mensaje X10 preestablecido. Este está formado por un código de inicio, un código de casa, un código de clave y un sufijo. El código de clave puede ser un código de unidad o un código de función.

- ***Código de Inicio:*** Es un código preestablecido e igual a 1110. Es utilizado por los receptores para identificar el inicio de un mensaje X10.

- ***Código de Casa:*** Son 16 códigos, los cuales tienen la finalidad de ser uno de los identificadores de los dispositivos X10. Los códigos de casa van desde la A a la P (Tabla 1.1).

Tabla 1.1: Códigos de Casa [9]

Código de Casa	C3	C2	C1	C0
A	0	1	1	0
B	1	1	1	0
C	0	0	1	0
D	1	0	1	0
E	0	0	0	1
F	1	0	0	1
G	0	1	0	1
H	1	1	0	1
I	0	1	1	1
J	1	1	1	1
K	0	0	1	1
L	1	0	1	1
M	0	0	0	0
N	1	0	0	0
O	0	1	0	0
P	1	1	0	0

- ***Código de Unidad:*** Estos código son el segundo identificador de los dispositivos X10 siendo 16, los cuales, van del 1 al 16 (Tabla 1.2).

Tabla 1.2: Códigos de Unidad [9]

Código de Unidad	U3	U2	U1	U0
1	0	1	1	0
2	1	1	1	0
3	0	0	1	0
4	1	0	1	0
5	0	0	0	1
6	1	0	0	1
7	0	1	0	1
8	1	1	0	1
9	0	1	1	1
10	1	1	1	1
11	0	0	1	1
12	1	0	1	1
13	0	0	0	0
14	1	0	0	0
15	0	1	0	0
16	1	1	0	0

- ***Código de Función:*** El código de función provee 15 códigos preestablecidos para el funcionamiento de los dispositivos X10 (Tabla 1.3).

Tabla 1.3: Códigos de Función [9]

Código de Función	F3	F2	F1	F0
Todas las Unidades Off	0	0	0	0
Todas las Unidades On	0	0	0	1
On	0	0	1	0
Off	0	0	1	1
Oscurecer	0	1	0	0
Aclarar	0	1	0	1
Todas las luces Off	0	1	1	0
Código Extendido	0	1	1	1
Solicitud de Saludo (HR)	1	0	0	0
Respuesta a Saludo (HA)	1	0	0	1
Ajustar Intensidad	1	0	1	X
Código Extendido (Analógico)	1	1	0	0
Estado = On	1	1	0	1
Estado = Off	1	1	1	0
Solicitud de Estado	1	1	1	1

- ***Sufijo:*** El sufijo es un bit identificador de Código de Unidad (bit igual a 0) o Código de Función (bit igual a 1).

Un dispositivo X10 viene representado por el código de casa y el código unidad. Esta identificación de Casa (A - P) y Unidad (1 - 16) proporciona un total de 256 direcciones de dispositivos X10 como máximo, junto con 15 funciones posibles. Por lo tanto, para que un dispositivo X10 pueda ser controlado remotamente, el receptor X10 debe recibir un código de casa y un código de unidad como identificación y un código de función como comando.

1.2.6 FORMATO DE LA TRANSMISIÓN [9], [10]

Para la transmisión por la red eléctrica, se estructura un formato de mensaje X10, este mensaje está formado por un código de inicio (1110), un código de casa, un código de clave (Código de Unidad o Código de Función) y un código de prefijo.

Para una transmisión completa se envían dos bloques de mensajes X10 separados por 3 ciclos de la señal eléctrica. Cada bloque está formado por dos mensajes X10 sin separación entre ellos. Por lo cual, se envía el código de casa y unidad en el primer bloque, mientras que el código de casa y función se envía en el segundo bloque.

Cada mensaje X10 se envía dos veces con el objetivo de dar fiabilidad a las transmisiones X10, y se deja 3 ciclos de la señal eléctrica entre bloques de mensajes X10, para darle tiempo suficiente al microcontrolador para guardar en sus registros la información recibida (figura 1.8, 1.9 y 1.10).

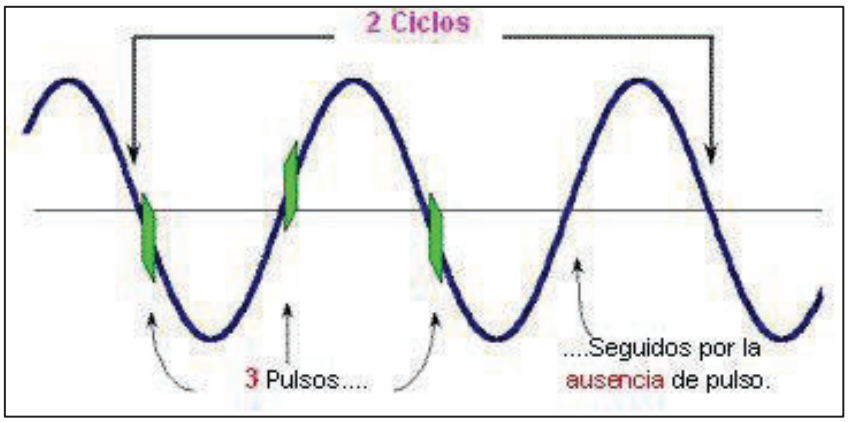


Figura 1.8: Código de Inicio [10]

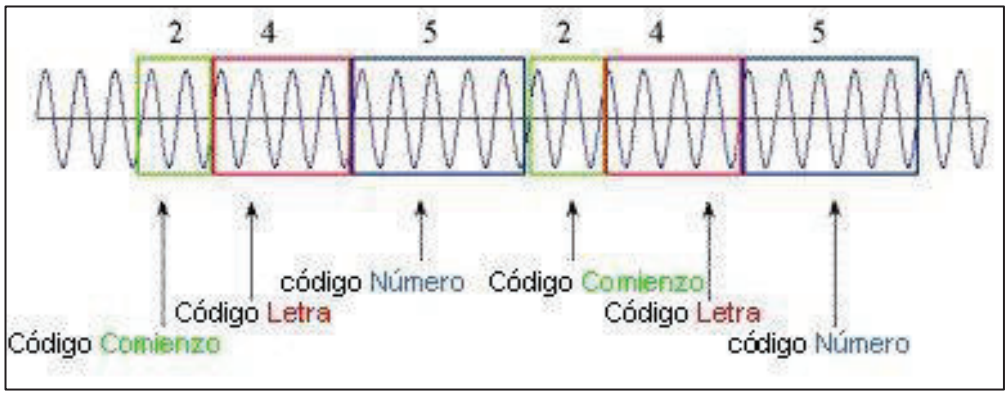


Figura 1.9: Bloque de mensajes X10 [10]

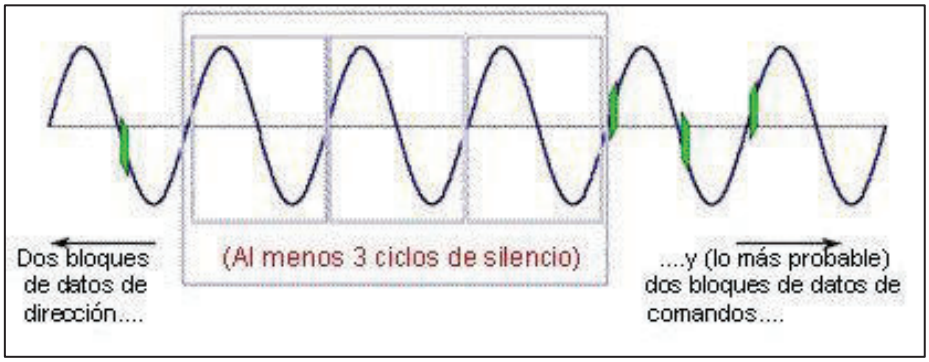


Figura 1.10: Ciclos de Pausa [10]

Para ubicar la información digital en la red eléctrica, se utiliza un ciclo de la señal eléctrica por cada bit a transmitir, puesto que, cada bit es enviado junto con su

complemento. Solo para la transmisión del código de inicio se lo hace sin su complemento, por lo cual, el código de inicio utiliza 2 ciclos de la señal eléctrica, mientras que el código de casa, unidad y función utilizan 4 ciclos y el prefijo utiliza un ciclo.

En total para la transmisión completa de código de casa, unidad y función se necesitan un total de 50 ciclos de la señal eléctrica, distribuidos de la siguiente manera:

- *22 Ciclos para el primer bloque*
    - 11 Ciclos para el primer mensaje X10 (Código de Casa y Unidad)
        - 2 Ciclos del Código de Inicio
        - 4 Ciclos del Código de Casa
        - 4 Ciclos del Código de Unidad
        - 1 Ciclo del Prefijo
    - 11 Ciclos de retransmisión del primer mensaje X10

- *3 Ciclos de Espera*

- *22 Ciclos para el segundo bloque*
    - 11 Ciclos para el segundo mensaje X10 (Código de Casa y Función)
        - 2 Ciclos del Código de Inicio
        - 4 Ciclos del Código de Casa
        - 4 Ciclos del Código de Función
        - 1 ciclo del Prefijo
    - 11 Ciclos de retransmisión del primer mensaje X10

- *3 Ciclos de Espera*

1.2.7 FUNCIONES X10 [9], [32]

A continuación se provee una descripción del funcionamiento y formato de transmisión de cada una de las funciones X10 preestablecidas.

- ***Todas las Unidades Off:*** Este comando permite apagar todas las unidades de una casa específica, por lo cual, para emitir este comando solo se necesita la transmisión de un bloque de mensajes X10 enviando el código de casa y la función.

- ***Todas las Luces On:*** Este comando permite encender todas las luces de una casa específica, al igual que en la función anterior solo es necesario la transmisión de un bloque de mensajes X10 (código de casa y función).

- ***On:*** Enciende el dispositivo asociado a una unidad específica.

- ***Off:*** Apaga el dispositivo asociado a una unidad específica.

- ***Oscurecer (Dim):*** Disminuye la intensidad del dispositivo asociado a una unidad específica. En la transmisión de esta función se presenta una excepción, en la cual, los dos primeros bloques de mensajes X10 (código de casa, unidad y función), se transmite normalmente, pero los siguientes bloques de mensajes X10 se transmiten solamente con la información de código de casa y función sin dejar los 3 ciclos de la señal eléctrica libres.

- ***Aclarar (Bright):*** Aumenta la intensidad del dispositivo asociado a una unidad específica. La transmisión de esta función se la realiza en el mismo formato de la función oscurecer (Dim).

- ***Todas las Luces Off:*** Este comando permite apagar todas las luces de una casa específica, al igual que en la función “Todas las luces on”, solo es

necesario la transmisión de un bloque de mensajes X10 (código de casa y función).

- ***Código Extendido:*** Este comando nos permite añadir más funciones a las 15 preestablecidas en el protocolo X10. Para la transmisión de esta función se envía el código de casa, unidad y función normalmente, y a continuación se envía sin pausas la nueva función en un formato de 4 bits (4 ciclos de la señal eléctrica).

- ***Solicitud de Saludo (Hail Request):*** Esta función sirve para encontrar otros dispositivos transmisores que se encuentren en la misma área del Código de Casa. El propósito de este comando es conocer otros transmisores con la misma dirección de casa, con el fin de escoger otra dirección del transmisor en el caso de que el código seleccionado estuviera ocupado.

- ***Respuesta a Saludo (Hail Acknowledge):*** Esta función sirve para responder a una Solicitud de Saludo (Hail Request).

- ***Ajustar Intensidad (Pre-set Dim):*** Esta función permite ajustar un nivel de intensidad preestablecido y presenta un pequeño cambio en su estructura. El primer bloque de mensajes X10 se envía en el formato ya conocido (Código de Inicio, Casa y Unidad), pero en el segundo bloque se utilizan los bits F0, H3, H2, H1 y H0 (F0 es el bit más significativo y H0 el bit menos significativo), para determinar el nivel de intensidad, por lo que, para el ajuste de la intensidad se tiene una precisión de 5 bits.

- ***Datos Extendidos (Analógico*)**: Esta función es utilizada para enviar información analógica adicional usando el protocolo X10 luego de haber pasado por un conversor A/D. El protocolo no especifica exactamente la manera como debe enviarse esta información analógica, por lo cual, se recomienda enviar los dos primeros bloques de información sin ningún cambio, en seguida sin pausas se envía un código de 8 bits (8 ciclos de la

señal eléctrica), indicando el número de bits de información analógica que se va a enviar, y a continuación sin pausas la información analógica.

- ***Solicitud de Estado:*** Esta función la utilizan los dispositivos transmisores para monitorear el estado de los dispositivos X10.

- ***Estado = On:*** Respuesta On a la solicitud de estado.

- ***Estado = Off:*** Respuesta Off a la solicitud de estado.

La ventaja principal que representa el uso de dispositivos con el protocolo X10 es su facilidad de instalación y su no necesidad de instalar cableado adicional para su correcto funcionamiento. Esto está contrastado con su principal desventaja, la cual es la dependencia del funcionamiento de la red eléctrica.

Al depender de la red eléctrica para su funcionamiento, significa que está propenso a atenuaciones por parte de señales que estén próximas a la frecuencia de la portadora, así como también a las atenuaciones inherentes a la red eléctrica (como motores). De igual manera, el uso de la red eléctrica limita la velocidad de transmisión de las señales, puesto que la frecuencia de la señal eléctrica no puede ser alterada.

Para asegurar un funcionamiento correcto del sistema X10, es común hacer uso de filtros X10 (pasa-bajos) para limitar el recorrido de la señal X10, así como para evitar que estas se propaguen hacia otras instalaciones eléctricas.

1.3 DISPOSITIVOS LÓGICO PROGRAMABLES [12], [26], [27], [30], [39], [46]

Un Dispositivo Lógico Programable (PLD) es un circuito integrado digital, cuyas funciones requeridas son programadas por el usuario, quien diseñará el PLD de tal forma que le permita cumplir los objetivos del sistema.

La estructura interna de un PLD está compuesta principalmente por matrices de compuertas AND, OR, conexiones y además se incluyen registros. Gracias al empleo de todos estos recursos que posee un PLD, se pueden implementar las funciones lógicas deseadas, empleando el software y programador más adecuado.

A continuación se realizará una breve descripción de los principales PLDs que anteceden al FPGA.

- ***PROM (Programmable Read-Only Memory):*** Es el primer dispositivo considerado como PLD, posee una tabla de decodificación completa de las entradas a la memoria y permite la implementación de cualquier función lógica combinacional.

- ***PAL (Programmable Array Logic):*** Estos dispositivos básicamente están conformados por un bloque que permite negar las entradas, un nivel de compuertas AND programables y un nivel de compuertas OR no programables. Las salidas de las compuertas AND se conectan a las entradas de las compuertas OR, permitiendo de esta manera expresar cualquier función lógica como una suma de productos. La figura 1.11 muestra el diagrama de bloques que describe el comportamiento de una PAL.

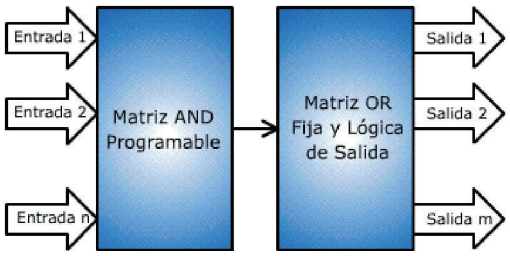


Figura 1.11: Diagrama de bloques de una PAL (Programmable Logic Array) [30]

- ***GAL (Generic Array Logic):*** Estos dispositivos son un poco más complejos que las PAL. Están conformados por un nivel de lógica AND reprogramable a la entrada y a continuación por un bloque lógico conocido como macrocelda a la salida. Las GAL permiten realizar cualquier función lógica en forma de suma

de productos con un número limitado de variables. Además las GAL se caracterizan porque se pueden programar una y otra vez gracias a que usan tecnología EECMOS (Electrically Erasable CMOS). En la figura 1.12 se muestra la estructura básica de una GAL, en donde se puede evidenciar la presencia de la compuerta AND y de la macrocelda a la salida.

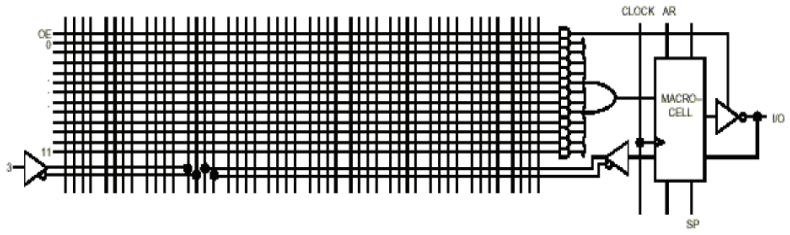


Figura 1.12: Estructura Básica de una GAL [26]

- ***CPLD (Complex Programmable Logic Device):*** Estos dispositivos combinan varios bloques lógicos similares a las PAL o GAL con una matriz de interconexión programable. Los CPLD se fabrican con distintas capacidades para adaptarlos a las necesidades del usuario. Su estructura está formada no sólo por macroceldas, sino además por una PIM (Programmable Interconect Matrix), una red de distribución de reloj dedicada y bloques de entrada salida que permiten interfaces compatibles con diferentes estándares eléctricos. Los bloques AND y OR están ambos incluidos en la macrocelda del dispositivo. A continuación en la figura 1.13, se muestra la estructura de un CPLD.

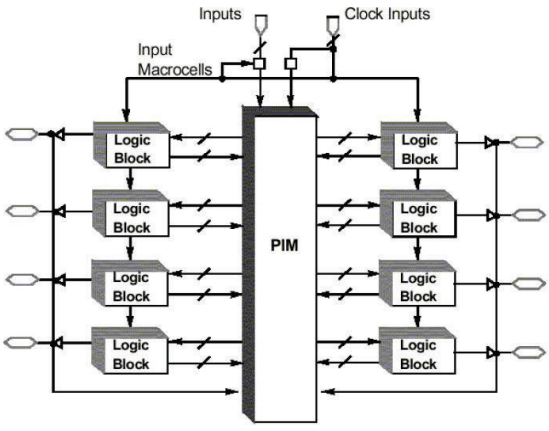


Figura 1.13: Estructura Básica de un CPLD [26]

1.3.1 FPGA [15], [19], [36]

Un FPGA (Field Programmable Gate Array) es un dispositivo lógico programable compuesta por un arreglo de bloques lógicos que se comunican entre sí por medio de canales de interconexión, permitiendo de esta manera la implementación de diversos circuitos digitales en él. Es decir que el usuario puede programar al FPGA para que cumpla determinadas funciones personalizadas sin la necesidad de utilizar grandes cantidades de recursos.
El FPGA ofrece varios beneficios entre los cuales se puede destacar:

- ***Rendimiento:*** Un FPGA puede llevar a cabo diferentes procesos de manera paralela. Gracias a esta característica, dichos procesos no necesitan competir por los mismos recursos, ya que a cada uno de ellos se les asigna una determinada sección de bloques lógicos del FPGA, consiguiendo de esta manera ejecutar cada proceso de forma independiente, segura y optimizada en un mismo ciclo de reloj.

- ***Adaptabilidad:*** Debido a la flexibilidad y gran capacidad que ofrece un FPGA, se puede enfrentar los futuros retos del mercado empleando este dispositivo. Para ello se puede comprobar una determinada idea en el hardware del FPGA y posteriormente realizar la respectiva programación e implementación de dicha idea sin tener que recurrir a ningún proceso de fabricación adicional.

- ***Costo:*** Conforme avanza el tiempo, los requerimientos de los sistemas cambian, y la naturaleza reprogramable de un FPGA permite modificar los diseños a un precio casi insignificante en comparación al precio de rediseño de otras tecnologías, por lo cual el empleo de un FPGA representa un gran ahorro económico para el usuario tanto en el presente como para el futuro.

- ***Fiabilidad:*** Un FPGA posee hardware dedicado a cada tarea gracias a lo cual se asegura la ejecución sin errores de un determinado programa.

Los FPGAs poseen un campo de aplicación sumamente amplio debido a su versatilidad y flexibilidad. Entre las principales aplicaciones se encuentran el procesamiento digital de señales, que a su vez incluye comunicaciones, procesamiento de datos, además de radio definido por software, sistemas aeroespaciales y de defensa, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, entre otras.

1.3.1.1 Arquitectura General de un FPGA [15], [16], [19], [36], [46]

Un FPGA está conformado básicamente por un Conjunto de Bloques Lógico Configurables (Configurable Logic Blocks: CLBs), los cuales se encuentran rodeados por un perímetro de Bloques Configurables de Entrada/Salida (Programmable Input/Output Blocks: IOBs). Finalmente todo este conjunto de bloques funcionales, se encuentran conectados por una Matriz de Interconexiones Programables.

En la figura 1.14, se muestra la arquitectura básica de un FPGA, con los elementos anteriormente mencionados.

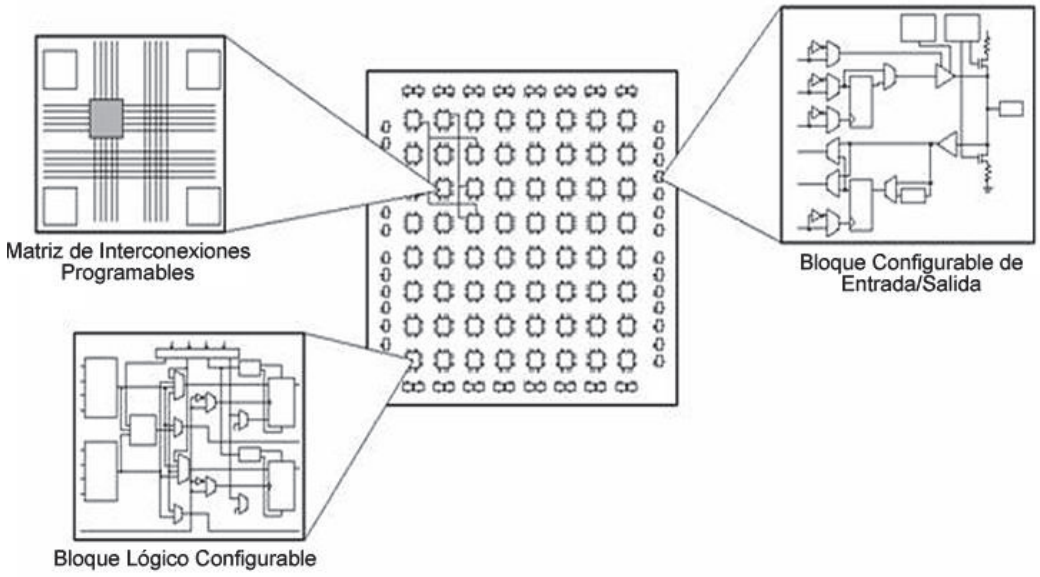


Figura 1.14: Arquitectura Básica de un FPGA [36]

1.3.1.1.1 Los Bloques Lógicos Configurables (CLBs)

Los Bloques Lógico Configurables constituyen la parte más importante del FPGA, están conformados por Look-Up Tables (LUTs) y por flip-flops, y además son capaces de implementar funciones lógicas, tanto combinacionales como secuenciales. En la figura 1.15, se observa la estructura de un bloque lógico configurable, en donde un multiplexor puede ser configurado para aceptar la salida de la LUT o aceptar una entrada del bloque lógico configurable directamente.

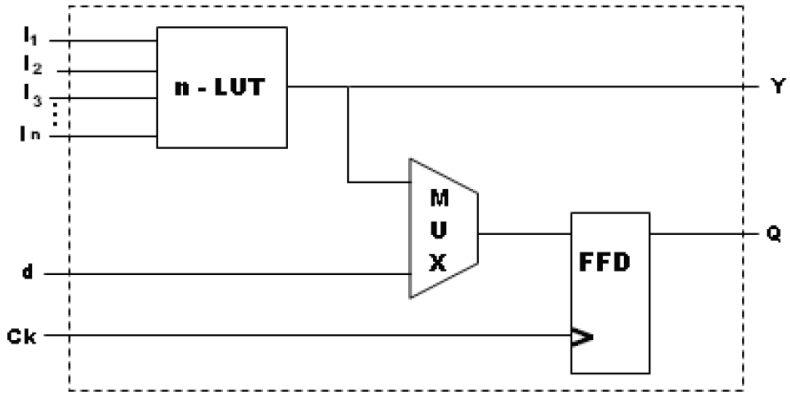


Figura 1.15: Estructura de un Bloque Lógico Configurable [16]

- ***Tablas de Búsqueda – Look-Up Tables (LUTs):*** Son parte importante del FPGA ya que cuentan memoria RAM interna y permiten la implementación de la lógica combinacional. Las LUTs almacenan una tabla de verdad de una función lógica determinada, donde las direcciones de la memoria son las entradas de la función lógica que se desea implementar, y en cada celda de dicha memoria se almacena el resultado de la combinación de dichas entradas. Por lo tanto en una n-LUT se puede implementar cualquier función lógica de n entradas. En la figura 1.16 se muestra la estructura de una LUT de n entradas.

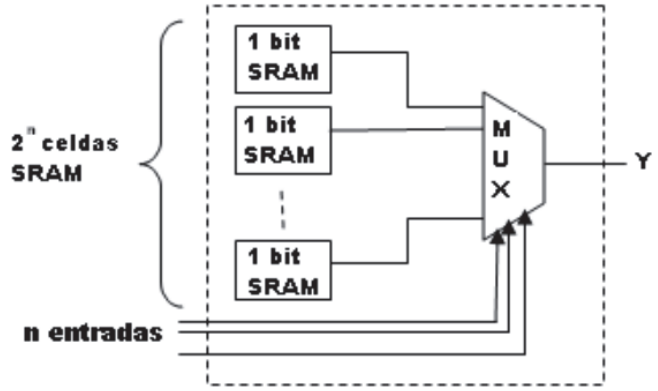


Figura 1.16: Estructura de una Look-Up Table de n entradas [16]

- ***Elementos de almacenamiento. Flip-flops:*** Permiten la sincronización de la salida con una señal de reloj externa, gracias a lo cual es posible la implementación de la lógica secuencial y de registros.

1.3.1.1.2 Los Bloques Configurables de Entrada/Salida (IOBs)

Los bloques de entrada/salida permiten el flujo de las señales hacia afuera y hacia adentro del dispositivo y además constan de los siguientes elementos:

- Salidas configurables como Tri-State u Open-Collector.
- Entradas con posibilidad de pull-up o pull-down programables.
- Registros de entrada.
- Registros de salida

1.3.1.1.3 Matriz de Interconexiones Programables

Las interconexiones programables son canales de líneas de conexión que se encuentran distribuidos de forma tanto horizontal como vertical y que permiten la comunicación entre cada uno de los bloques funcionales. Existen cuatro formas de realizar estas conexiones entre bloques:

- ***Líneas directas:*** Comunican un bloque lógico con sus bloques vecinos.
- ***Conexiones de propósito general:*** Comunican dos bloques no adyacentes.

- ***Líneas largas:*** Se utilizan cuando las señales deben atravesar grandes distancias, evitando de esta forma los retrasos producidos por las interconexiones intermedias.

- ***Líneas rápidas:*** Son las encargadas de realizar las conexiones entre las LUTs cuando los bloques lógicos poseen más de una.

1.4 LA PLATAFORMA SPARTAN 3 DE XILINX [15], [23], [42]

Xilinx es la mayor empresa dedicada a la investigación y desarrollo de dispositivos de lógica programable como CPLDs y FPGAs, que son utilizados en numerosas aplicaciones. Las familias actuales de dispositivos de Xilinx son:

- Spartan, caracterizada por su bajo costo.
- Artix y Kintex, caracterizadas por ofrecer mayor rendimiento que la familia Spartan.
- Virtex, caracterizada por ser la tarjeta de más alto rendimiento y costo.

Para un mayor conocimiento sobre cada una de estas familias se recomienda visitar la página oficial de Xilinx, ver [23], en la cual se encuentra disponible de manera detallada toda la información acerca de cada dispositivo de Xilinx.

Todas las familias anteriormente descritas permiten la implementación del protocolo de domótica X10, puesto que los recursos físicos y lógicos que poseen son de gran capacidad. Sin embargo en el presente Proyecto, nos vamos a enfocar en la familia Spartan 3 de Xilinx, ya que ofrece un bajo costo y permiten el desarrollo de manera óptima de un gran número de aplicaciones que no requieren demasiada memoria como acceso de banda ancha, redes para el hogar, equipo de televisión digital, etc.

En la tabla 1.4 se describe la generación de tarjetas Spartan 3, las cuales incluyen las siguientes familias y subfamilias:

Tabla 1.4: Generación de tarjetas Spartan 3

FAMILIA	SUBFAMILIA
Spartan-3	Ninguna
Spartan-3A	Spartan-3A
	Spartan-3AN
	Spartan-3A DSP
Spartan-3E	Ninguna

El presente Proyecto se desarrolla con la tarjeta Spartan 3E Starter Kit, por lo cual a continuación se detallan las principales características de esta tecnología.

1.4.1 ARQUITECTURA DE LA FPGA SPARTAN 3 [15], [19]

La familia Spartan 3 comparte una arquitectura común, existen unas pequeñas diferencias entre familias, pero en general su arquitectura consiste de 5 elementos fundamentales:

- ***BLOQUES LÓGICO CONFIGURABLES (Configurable Logic Blocks - CLBs):*** Contienen Look-Up Tables (LUTs) basadas en tecnología RAM. Estos bloques pueden ser programados para implementar una gran variedad de elementos lógicos y de almacenamiento que pueden ser utilizados como flip-flops o latches.

- ***BLOQUES DE ENTRADA/SALIDA (Input/Output Blocks - IOBs):*** Controlan el flujo de datos entre los pines de entrada/salida y la lógica interna del dispositivo. Cada uno de los bloques de entrada/salida soporta flujo de datos bidireccional, operación tri-estado, varios estándares de voltaje, entre ellos voltajes diferenciales de alto rendimiento, registros DDR (Double Data Rate) e impedancia controlada digitalmente (Digitally Controlled Impedance - DCI).

- ***BLOQUES DE MEMORIA RAM:*** Permiten el almacenamiento de datos en bloques de 18Kbits.

- ***BLOQUES DE MULTIPLICACIÓN:*** Aceptan como entrada dos números binarios de 18 bits y calculan el producto de dichos números.

- ***ADMINISTRADOR DIGITAL DE RELOJ (Digital Clock Manager – DCM):*** Provee de soluciones digitales autocalibradas y completas para la distribución, retraso, multiplicación, división y cambio de fase de las señales de reloj de todo el circuito.

En la figura 1.17 se puede observar la arquitectura de una tarjeta Spartan 3. Un anillo de IOBs rodea a un arreglo regular de CLBs, dicho arreglo es atravesado por una columna de bloques de memoria RAM, esta columna está formada por varios bloques de 18 Kbits, los cuales están asociados a un multiplicador dedicado. Finalmente en el extremo de las columnas de bloques RAM se encuentra los DCMs.

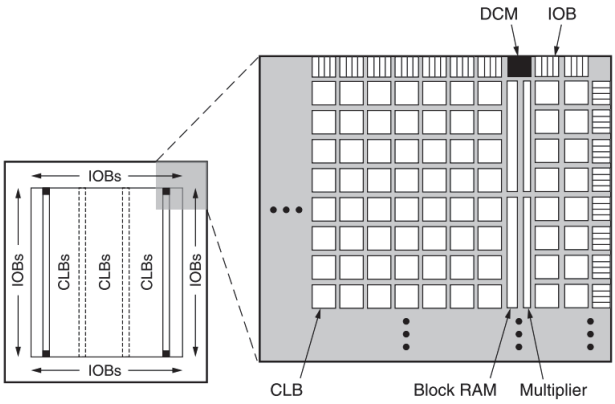


Figura 1.17: Arquitectura de la Spartan 3 [19]

En el Anexo 1 se tiene una descripción más detallada de cada uno de los elementos funcionales de la Spartan 3.

1.4.2 TARJETA XILINX SPARTAN-3E STARTER KIT [5]

La tarjeta Spartan-3E Starter Kit nos proporciona las herramientas necesarias para el desarrollo del presente Proyecto. A continuación se mencionan las características más importantes de la tarjeta:

- *Dispositivos Xilinx integrados:*
  - FPGA de la familia Spartan-3E modelo XC3S500E-4FG320C (con 232 pines de E/S para el usuario, 1164 CLBs y 1000 módulos lógicos.
  - Controlador CPLD CoolRunner™-II CPLD (XC2C64A-5VQ44C)
  - 4 Mbits de memoria PROM Platform Flash (XCF04S-VO20C)
- *Reloj*:
  - Oscilador de 50MHz
  - Socket de 8 pines para un oscilador auxiliar
  - Conector tipo SMA (SubMiniature versión A) que puede ser utilizado como entrada o salida de señales de reloj.
- *Memorias incluidas:*
  - 128 mbit de memoria Parallel Flash
  - 16 mbit de memoria SPI (Serial Peripheral Interface) Flash
  - 64 mbytes de memoria DDR SDRAM
- *Interfaces y conectores:*
  - 4 salidas DAC (Digital-to-Analog Converter)
  - 2 entradas ADC (Analog-to-Digital Converter)
  - Conexión Ethernet 10/100
  - Puerto USB embebido que facilita la configuración y programación del FPGA a través de un puerto USB de un PC sin recurrir a algún cable especial.
  - 2 puertos serie RS-232 (DTE y DCE)
  - Puerto VGA
  - Puerto PS/2 para ratón o teclado

- 4 switches de deslizamiento
- 4 switches tipo pulsador
- 1 encoder de rotación (Rotary Shaft Encoder)
- 8 diodos tipo led individuales
- 3 conectores de expansión de seis pines.
- 1 conector de expansión Hirose FX2 de alta velocidad con 43 señales.

❖ *Display:* Pantalla LCD de 2 líneas de 16 caracteres cada una.

En la figura 1.18, se muestra los elementos de la tarjeta Spartan-3E Starter Kit descritas anteriormente y su localización en el FPGA:

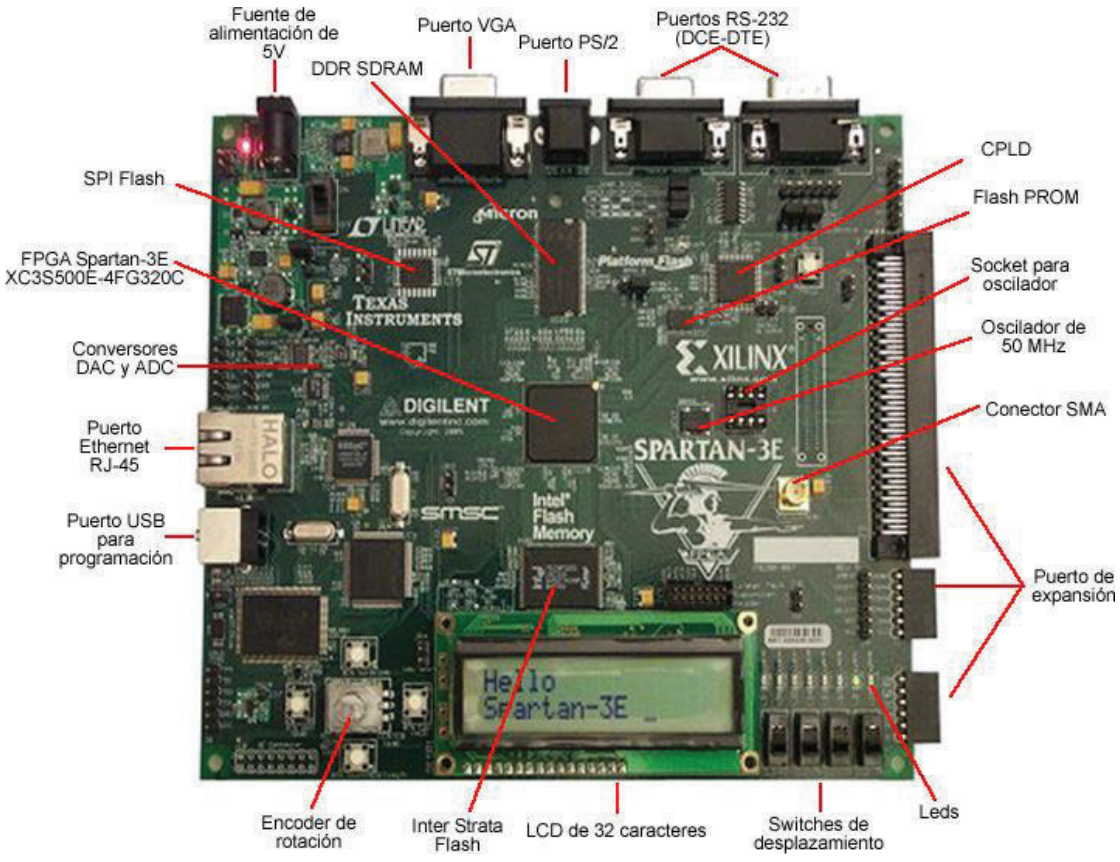


Figura 1.18: Vista superior de la tarjeta Spartan-3E Starter Kit y sus componentes [5]

A continuación, en la sección 1.4.2.1 se describe el principal componente del FPGA utilizado en este Proyecto.

1.4.2.1 Puerto Serial RS-232 [5]

La tarjeta Spartan-3E Starter Kit tiene 2 puertos seriales: un conector DB9 hembra y uno macho. El conector hembra puede trabajar como DCE (Data Communications Equipment) y el conector macho puede trabajar como DTE (Data Terminal Equipment).

El puerto DCE se conecta directamente al puerto serial que se encuentra disponible en computadoras mediante un cable serial directo, no es necesario utilizar cable null modem, gender changers ó cable cruzado. El puerto DTE se utiliza para controlar otros periféricos RS-232 como módems, impresoras ó realizar pruebas simples con el conector DCE.

En la figura 1.19, se pueden observar un ejemplo de los tipos de dispositivos que se pueden conectar en cada puerto respectivamente y su ubicación física en la tarjeta.

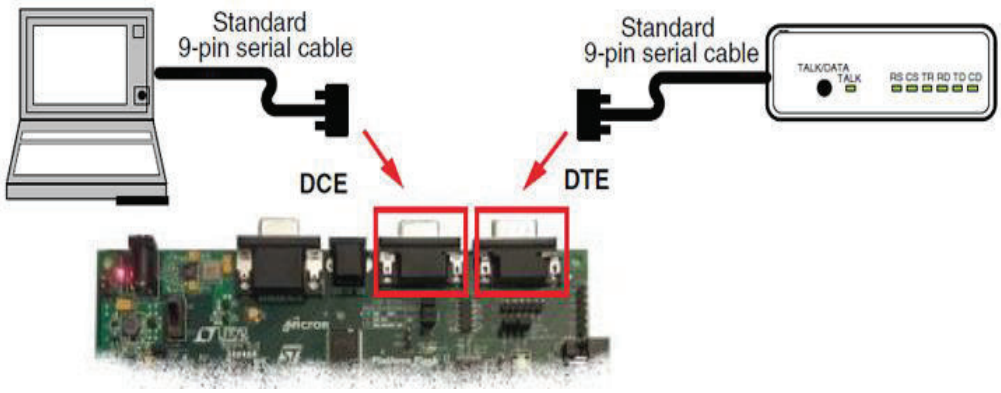


Figura 1.19: Puertos DCE y DTE de la Spartan-3E Starter Kit [5]

La figura 1.20 muestra cómo es la conexión entre el FPGA y los dos conectores DB9. El FPGA entrega a la salida datos seriales en niveles LVTTL ó LVCMOS al

dispositivo Maxim, el cual a su vez convierte dicho valor lógico al nivel de voltaje RS-232 apropiado. De igual forma el dispositivo Maxim, convierte la entrada de datos serial RS-232 a niveles de voltaje LVTTL para el FPGA. Existe una resistencia en serie entre el pin de salida del Maxim y el pin de recepción del FPGA que impide que existan conflictos accidentales de nivel de voltaje.

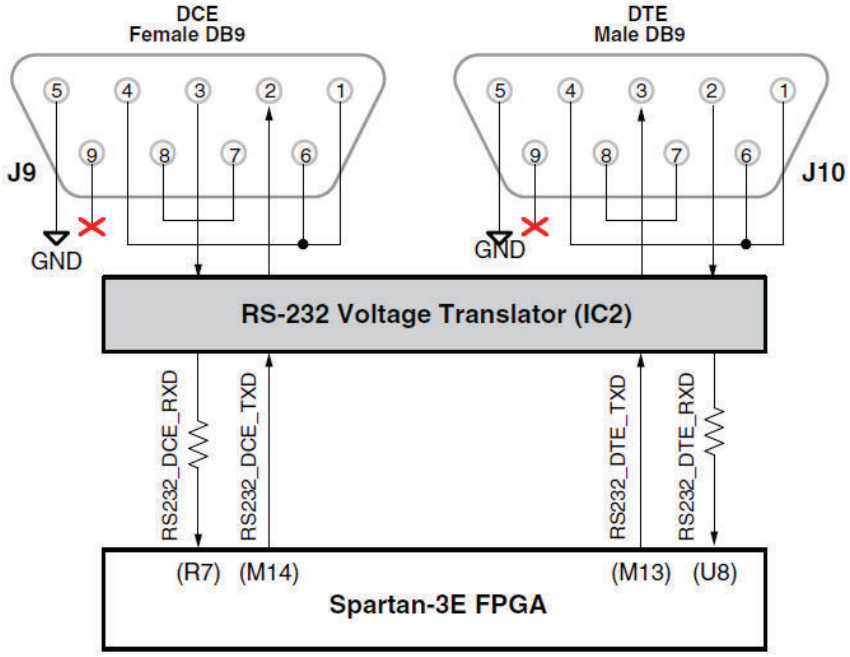


Figura 1.20: Puertos Seriales RS-232 [5]

En este Proyecto, se usan los dos puertos RS-232, uno de ellos va a recibir las señales desde una interfaz gráfica, las cuales van a ser codificadas de acuerdo al protocolo X10, y a través del otro puerto las señales codificadas serán enviadas para realizar el control del respectivo dispositivo dentro del hogar.

1.5 PROGRAMACIÓN DE UN FPGA [19], [42]

El FPGA tiene celdas que se configuran con una función específica ya sea como memoria (FLIP-FLOP tipo D), como multiplexor ó con una función lógica tipo AND, OR, XOR. El trabajo de la persona que va a programar el FPGA, es definir la función

lógica que realizará cada uno de los CLB, seleccionar el modo de trabajo de cada IOB e interconectarlos.

En la actualidad se cuenta con una gran variedad de entornos de desarrollo especializados en el diseño de sistemas a implementarse en un FPGA. Un diseño puede ser capturado ya sea como esquemático, o haciendo uso de un lenguaje de programación especial. Estos lenguajes de programación especiales son conocidos como HDL ó Hardware Description Language (lenguajes de descripción de hardware). Los HDLs más utilizados son:

- VHDL
- Verilog
- ABEL

El lenguaje de programación que se utiliza en este proyecto es VHDL, razón por la cual se lo describe a continuación.

1.5.1 VHSIC Hardware Description Language (VHDL) [12], [14], [37], [41], [43]

VHDL significa VHSIC (Very High Speed Integrated Circuit) Hardware Description Language. VHDL es un lenguaje estandarizado por la IEEE para la descripción del hardware que se va a utilizar en la modelación, documentación, simulación, verificación y síntesis de circuitos digitales y además para la programación de PLD (Programmable Logic Device), FPGA (Field Programmable Gate Array), ASIC y similiares.

VHDL presenta un gran número de beneficios, entre los cuales se destacan:

- Permite el diseño jerárquico y modular, esto implica que un diseño grande se puede descomponer en bloques más pequeños e independientes.

- Ejecución del diseño completo desde un nivel alto de abstracción hasta un nivel bajo. Un nivel alto de abstracción se caracteriza porque el diseñador solo describe el comportamiento del sistema, no se preocupa por las señales o componentes internos, mientras que en un nivel bajo de abstracción se describe la estructura interna del circuito
- Es un lenguaje independiente de los dispositivos y del fabricante, es decir se puede adaptar a diferentes simuladores y metodologías de diseño.

- Puede ser reutilizado, es decir que permite implementar los diseños existentes en nuevas tecnologías.

- Posee una sintaxis amplia y flexible, gracias a lo cual permite la descripción de sistemas desde los más simples hasta los más complejos.

Un proyecto en VHDL está conformado por uno o varios ficheros, cada uno de los cuales guarda el código VHDL utilizado. La estructura básica de un fichero es la siguiente:

- ***Llamadas a librerías (libraries)*:** Almacenan elementos que permitirán la descripción de las entidades.

- ***Entidades (entities)*:** Aquí se definen las entradas y salidas que tendrá el circuito que se va a diseñar.

- ***Arquitecturas (architectures)*:** Describe el funcionamiento del circuito que se va a diseñar. Pueden existir diversas arquitecturas para una misma entidad.

1.5.1.1 Flujo de Diseño en VHDL [14], [33], [41]

El flujo de diseño en VHDL consta de los siguientes pasos:

1. ***Creación del modelo:*** Consiste en describir mediante el lenguaje VHDL el diseño del circuito digital que se pretende realizar.

2. ***Simulación funcional:*** Comprueba que el código VHDL que se escribió anteriormente se ejecuta de acuerdo a lo que se desea, caso contrario se debe modificar el código hasta que funcione correctamente.

3. ***Síntesis del modelo diseñado:*** Una vez que el modelo ha sido validado en la simulación, se procede a sintetizarlo. La síntesis consiste en adaptar el modelo diseñado a un hardware específico, en nuestro caso a un FPGA. Este proceso lo realiza el software utilizado de forma automática, dando como resultado un archivo que contiene la lista de elementos de hardware y sus interconexiones. Este archivo no es otra cosa más que una representación a nivel de compuertas del modelo diseñado anteriormente en lenguaje VHDL.

4. ***Simulación funcional post-síntesis:*** Por medio de una simulación a nivel de compuertas, se comprueba que los resultados de la síntesis del modelo se hayan realizado correctamente.

5. ***Implementación:*** El proceso de implementación del diseño consta de varias fases que se detallan a continuación:

    5.1. *Traducción (translate):* Como su nombre lo indica, esta es una fase de traducción en donde el archivo generado en el proceso de síntesis cambia de su formato original a un formato estándar. Para el caso de los FPGA de Xilinx usando la herramienta NGDBUILD el archivo se traduce de formato NGC (Native Generic Circuit) a formato NGD (Native Generic Design).

    5.2. *Mapeo (mapping):* Esta fase consiste en que los elementos lógicos se mapean en los componentes físicos reales del FPGA.

5.3. *Ubicación y Ruteo (placement and routing):* El proceso de ubicación consiste en ubicar los elementos mapeados previamente en los lugares correspondientes en el FPGA, mientras que el proceso de ruteo se encarga de interconectar estos elementos de tal forma que los retardos de propagación sean mínimos. Finalizada esta fase se genera un archivo que contiene los elementos del FPGA localizados e interconectados, en el caso de los FPGA de Xilinx este archivo es de formato NCD (Native Circuit Description).

6. ***Análisis de soporte:*** En este proceso se extrae y analiza los retardos internos del FPGA que se originan al finalizar la fase de Ubicación y Ruteo, determinando de esta forma si dichos retardos son aceptables o no.

7. ***Simulación Temporal post-implementación:*** Permite realizar una simulación del diseño final, comprobando de esta manera si el diseño funciona correctamente, de no ser así, se debe volver a alguno de los pasos anteriores y corregir el error.

8. ***Programación en el dispositivo:*** Una vez finalizado todo el proceso de diseño y sus respectivas verificaciones, se genera un archivo de configuración que contiene la trama de bits producida por la configuración del FPGA, en el caso de una tarjeta de Xilinx este archivo se denomina *bitstream* y es generado por la herramienta BITGEN. Con este archivo se implementa el diseño en la tarjeta Spartan-3E Starter Kit y se comprueba su funcionamiento, completando de esta forma el proceso de diseño en VHDL.

Para el diseño e implementación del presente Proyecto se utiliza la tarjeta Spartan-3E Starter Kit y el paquete de software ISE Design Suite 14.5 de Xilinx, adicionalmente en la figura 1.21 se muestra el flujo de un diseño que se debe seguir en VHDL de tal forma que se pueden conseguir exitosamente los objetivos propuestos.

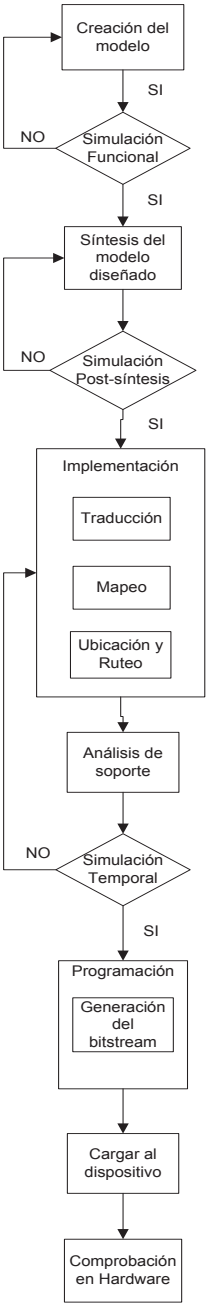


Figura 1.21: Flujo de Diseño en VHDL

CAPÍTULO 2

DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO DE SISTEMA DOMÓTICO

2.1 DESCRIPCIÓN DEL SISTEMA

El prototipo de sistema domótico consiste de una vivienda a escala de tres pisos. El área de esta vivienda es 1.10 m^2 y se encuentra distribuida de la siguiente manera:

- En el primer piso se encuentran localizados la entrada, la sala- comedor, la cocina y el garaje.
- En el segundo piso se encuentran localizados la habitación 1 con su respectivo baño y la habitación 2.
- En el tercer piso se encuentra el ático.

En el Anexo 2 se encuentran los planos del prototipo de vivienda, en la cual se implementó el sistema domótico.

En base a la descripción del prototipo de vivienda, se establecieron los siguientes requerimientos de control para el sistema domótico:

- Control de iluminación sencillo (ON / OFF) en tiempo real y en horarios programados.
- Control dimmerizado de iluminación en tiempo real.
- Control de las persianas de la vivienda en tiempo real y en horarios programados.
- Control de acceso a la vivienda en tiempo real.

En la tabla 2.1 se describe la función y ubicación de los elementos que se van a controlar dentro del prototipo de vivienda.

Tabla 2.1: Elementos y funciones del sistema domótico

LUGAR	ELEMENTO	FUNCIÓN	TIEMPO REAL	HORARIO PROGRAMADO
Sala-Comedor	Luminaria	ON / OFF	X	X
	Luminaria	Bright/Dim	X	
	Persiana	ON / OFF	X	X
	Persiana	Bright/Dim	X	
Cocina	Luminaria	ON / OFF	X	X
Habitación 1	Luminaria	ON / OFF	X	X
	Luminaria	Bright/Dim	X	
	Persiana	ON / OFF	X	X
	Persiana	Bright/Dim	X	
Habitación 2	Luminaria	ON / OFF	X	X
	Luminaria	Bright/Dim	X	
	Persiana	ON / OFF	X	X
	Persiana	Bright/Dim	X	
Baño	Luminaria	ON / OFF	X	X
Ático	Luminaria	ON / OFF	X	X
Entrada	Luminaria	ON / OFF	X	
	Puerta	ON / OFF	X	
Garaje	Luminaria / Puerta	ON / OFF	X	

De manera general y tomando como referencia la tabla 2.1, se puede evidenciar que el prototipo del sistema domótico, permite el control de los siguientes subsistemas dentro del hogar:

- Iluminación
- Persianas
- Accesos

Por esta razón para crear un ambiente amigable para el usuario se optó por diseñar el sistema en base a una arquitectura centralizada, en la cual el usuario a través de una interfaz gráfica, envía las instrucciones domóticas para su vivienda, las cuales son receptadas por el controlador, que en este prototipo, es el FPGA, el cual a la vez procesa dichas instrucciones y las convierte a formato X10, para finalmente enviar la información a los actuadores, los cuales ejecutan la función requerida por el usuario.

En la figura 2.1 se puede observar la configuración e integración total del prototipo del sistema desde la interfaz de usuario hasta la aplicación que desea controlar.

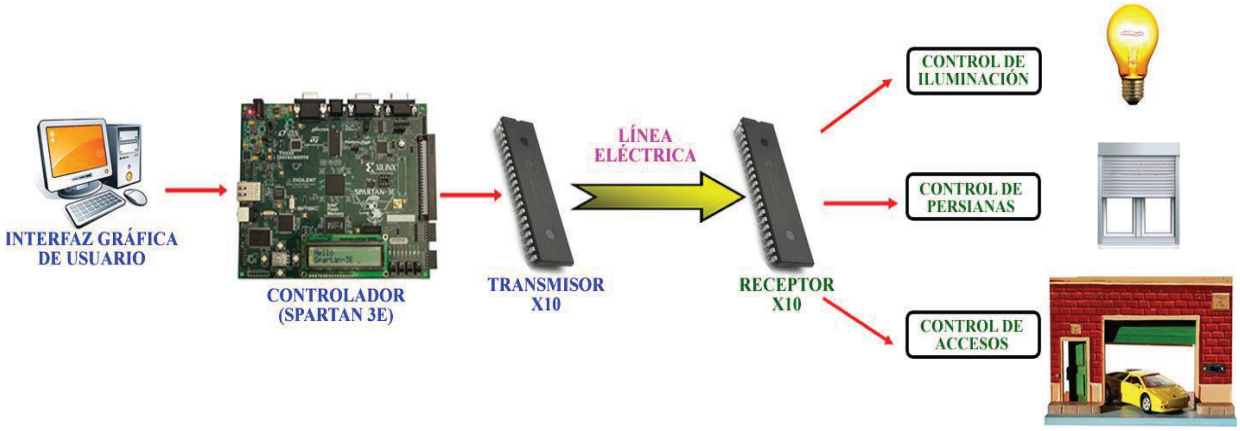


Figura 2.1: Configuración del Sistema Domótico

Para explicar de una mejor manera el diseño y la implementación de este sistema domótico, se divide a este capítulo en cuatro partes, en cada una de las cuales se explica el diseño del hardware, el diseño de los programas, el diseño de la interfaz de usuario y finalmente la construcción total del prototipo respectivamente.

2.2 DISEÑO DEL HARDWARE

En esta sección se describe el diseño del hardware necesario para la implementación del prototipo de sistema domótico. Empezando por el circuito transmisor X10, el cual recibe la información proveniente del controlador (FPGA) y la envía al circuito receptor X10, el cual dependiendo de la codificación la envía a su respectivo actuador.

En algunos casos se requieren de circuitos de sincronismo, como el detector de cruces por cero, ó de circuitos que permitan la intercomunicación entre diferentes niveles de voltaje, como la interfaz de comunicación serial.

Además se describe el diseño y construcción de los circuitos para las fuentes de alimentación, las cuales les permiten desempeñar satisfactoriamente las respectivas funciones de los circuitos anteriormente descritos.

Todos estos circuitos se presentan a continuación en los siguientes literales.

2.2.1 DISEÑO DEL CIRCUITO DETECTOR DE CRUCES POR CERO

El detector de cruces por cero es útil en el diseño del sistema domótico, puesto que sirve para sincronizar los cruces por cero de la señal eléctrica con el actuador de control de iluminación.

El circuito empleado en el prototipo fue tomado de los apuntes de clases, y está formado principalmente por un optoacoplador 4N25 (Ver Anexo 6), el cual al recibir el semi-ciclo de la señal eléctrica rectificada, activa la conducción de su transistor, facilitando de esta manera el paso de corriente hacia el emisor, donde se encuentra una resistencia, que permite la salida del voltaje de la fuente. Caso contrario, al recibir una señal de voltaje 0 V correspondiente al semi-ciclo negativo de la señal eléctrica después de su paso por el rectificador de media onda, se desactiva la conducción en el transistor, frenando el paso de corriente hacia la resistencia ubicada en el emisor, lo cual, impide la salida de voltaje de la fuente.

Para el proceso de rectificación de media onda, se utiliza un diodo rectificador 1N4007, cuyas características son 1000V y 1A, las cuales, al igual que el diodo 1N4005, se presentan en el Anexo 3, y permiten el correcto funcionamiento del detector de cruces por cero.

Dado que el optoacoplador posee un diodo led interno, se debe utilizar una resistencia limitadora de corriente a la entrada del optoacoplador. Por lo tanto, sabiendo que el optoacoplador soporta una corriente de entrada máxima de 60 mA, en nuestro diseño utilizamos una resistencia de 10 KΩ, con lo cual, se obtiene el siguiente valor de corriente.

$$I = \frac{V}{R} = \frac{120VAC}{10\ K\Omega}$$

$$\boldsymbol{I = 12\ mA}$$

Este valor permite determinar que la resistencia de 10 KΩ permite el paso de corriente de entrada al optoacoplador que no supera a la máxima soportada por el mismo.

Además de la corriente de entrada, se debe calcular la potencia que es disipada por la resistencia, como se muestra a continuación:

$$P = I^2R = (12\ mA)^2 * (10\ K\Omega)$$

$$\boldsymbol{P = 1,44\ W}$$

En base al resultado del cálculo de la potencia disipada, se decidió utilizar una resistencia de 5 W, con lo cual se asegura que dicha resistencia va a soportar la potencia que está disipando.

En la figura 2.2 se observa el detector de cruces por cero, el cual se utiliza en diferentes circuitos a lo largo del sistema domótico.

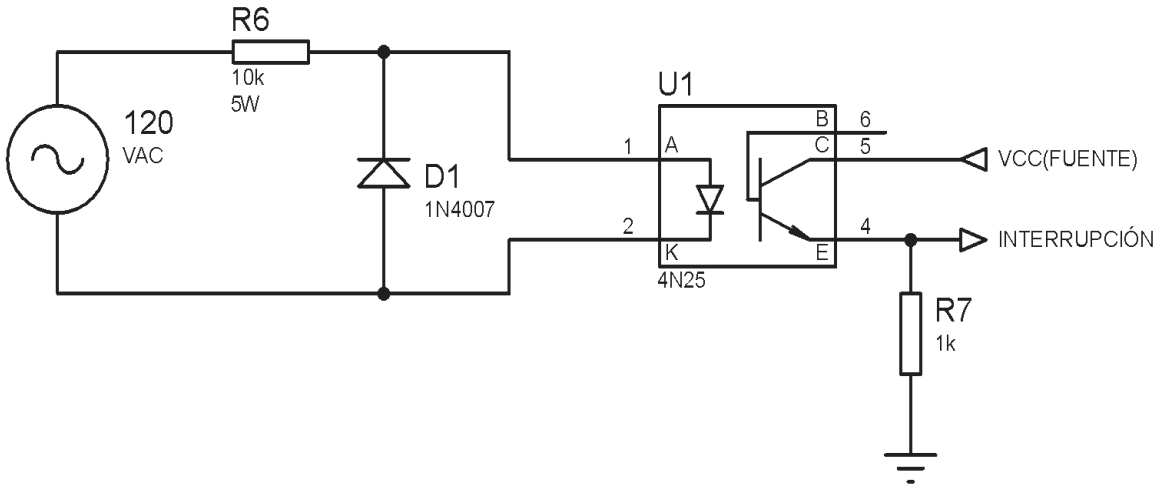


Figura 2.2: Detector de cruces por cero

2.2.2 DISEÑO DEL CIRCUITO TRANSMISOR X10 [9]

El circuito transmisor de los códigos X10 que se construyó para el presente prototipo, es el recomendado por el fabricante Microchip en su publicación AN236, y se encuentra conformado principalmente por el microcontrolador PIC16F877A (Ver Anexo 7). Cabe destacar que el circuito transmisor se encuentra alimentado por la fuente que se describe más adelante en la sección 2.2.7.1.

El circuito de transmisión usa un microcontrolador para procesar y transmitir las señales X10, el cual, utiliza cada cruce por cero de la señal eléctrica para sincronización. El microcontrolador posee diodos de protección de entrada en sus pines de I/O, los cuales, fijan cualquier voltaje superior a 5 V en 5 V o menor a 0 V en 0 V. Por esta razón, se utiliza esta propiedad del microcontrolador como detector de cruces por cero para la transmisión de las señales X10. Para limitar la corriente de entrada en el pin I/O del microcontrolador, se debe utilizar una resistencia limitadora de corriente. Para calcular el valor de esta resistencia, se utiliza el cálculo realizado en el artículo de Microchip, de la siguiente manera:

$$I_{PICO} = \frac{VAC_{PICO}}{R_{LIMITADORA}}$$

Considerando que la corriente pico máxima soportada por los pines I/O del microcontrolador es 250 mA, se tiene que la resistencia limitadora mínima debe ser:

$$R_{LIMITADORA-MIN} = \frac{VAC_{PICO}}{I_{PICO}}$$

$$R_{LIMITADORA-MIN} = \frac{165\ VAC}{250\ mA}$$

$$R_{LIMITADORA-MIN} = 660\ \Omega$$

Este valor corresponde a la mínima resistencia limitadora de corriente en la entrada del pin I/O del microcontrolador. Tomando como referencia el artículo de Microchip, se utiliza una resistencia limitadora de 5 MΩ, con el cual, se tendrá la siguiente corriente a la entrada del pin I/O del microcontrolador.

$$I_{PICO} = \frac{VAC_{PICO}}{R_{LIMITADORA}}$$

$$I_{PICO} = \frac{165\ VAC}{5\ M\Omega}$$

$$I_{PICO} = 33\ \mu A$$

Este valor de corriente es lo suficientemente bajo de tal forma, que no dañará al pin de I/O del microcontrolador.

Además, se utiliza el puerto serial para recibir los códigos X10 desde el controlador del sistema domótico (FPGA), una vez recibido el código completo, se procede la transmisión de los pulsos X10, para ello se utiliza el pin I/O (Entrada/Salida) 2 del puerto C, el cual a su vez está conectado a un transistor amplificador de corriente, con el objetivo de no forzar la salida de corriente del pin I/O del PIC, y acoplarlo para su salida en la red eléctrica.

Adicionalmente, el microcontrolador utiliza un cristal externo que le provee la señal de reloj, en nuestro caso utilizamos un cristal de cuarzo de 7,3247 MHz en la

configuración que el fabricante recomienda, puesto que este cristal lo podemos encontrar fácilmente en el mercado local.

A la entrada del pin VCC del microcontrolador se utiliza un arreglo de dos capacitores en paralelo con el objetivo de evitar que el microcontrolador sufra una reducción abrupta del voltaje de alimentación. Además para el control del Reset del microcontrolador se usa un pulsador en conjunto con un capacitor anti-rebote.

Posteriormente, como parte del circuito de transmisión, es necesario el uso de un limitador de voltaje para recortar el voltaje de la señal eléctrica que ingresa al circuito de salida del transmisor. Para este efecto se usan dos diodos zener 1N4736A (Ver Anexo 4), los cuales van a recortar la señal a 6.8 V protegiendo de esta manera al circuito transmisor.

Una de las partes más importantes del protocolo X10 es el envío de la información domótica a través de la línea eléctrica, para ello es fundamental la adaptación de la señal que contiene los mensajes X10 a la red eléctrica. Por esta razón es necesaria la utilización de un filtro pasa altos / acoplador, el cual presenta una alta impedancia a la señal eléctrica (60 Hz), mientras que presenta una baja impedancia a la señal de los mensajes X10 (120 KHz), esto significa que no permite el paso de la señal eléctrica hacia el interior del circuito mientras que si permite la salida y adaptación de los pulsos que contienen la información X10 para que viajen sobre la red eléctrica.

Para este efecto, Microchip utiliza en el filtro/acoplador un capacitor de 0.1 uF, con el cual las impedancias que presenta a cada señal se calculan de la siguiente manera:

La impedancia que presenta el filtro pasa altos a la señal eléctrica de 60 Hz es:

$$Z = \frac{1}{\omega C} = \frac{1}{2\pi f C} = \frac{1}{2\pi * (60\ Hz) * (0{,}1\ uF)}$$

$$\boldsymbol{Z = 26{,}53\ K\Omega}$$

La impedancia que presenta el filtro pasa altos a la información de 120 KHz es:

$$Z = \frac{1}{\omega C} = \frac{1}{2\pi f C} = \frac{1}{2\pi * (120\ KHz) * (0{,}1\ uF)}$$

$$\boldsymbol{Z = 13{,}26\ \Omega}$$

Por lo tanto se observa que el filtro pasa altos, como su nombre lo indica, bloquea el paso de la señal de baja frecuencia (60Hz) permitiendo una óptima transmisión de los mensajes X10 hacia su destino.

Una característica importante del circuito correspondiente al filtro / acoplador, es que, el capacitor que se utiliza es del tipo X2, el cual ayuda al circuito a suprimir el ruido, EMI y RFI, consiguiendo de esta manera que la señal de 120KHz se acople exitosamente a la señal eléctrica. Adicionalmente junto con el capacitor X2 se coloca en paralelo una resistencia de 1 MΩ, para que permita la descarga de este capacitor en cada semi-ciclo de la señal eléctrica así como también cuando el circuito se desconecte completamente de cualquier tipo de alimentación.

A continuación en la figura 2.3 se observa el circuito de transmisión de la información X10, en el cual constan todas las partes anteriormente mencionadas.

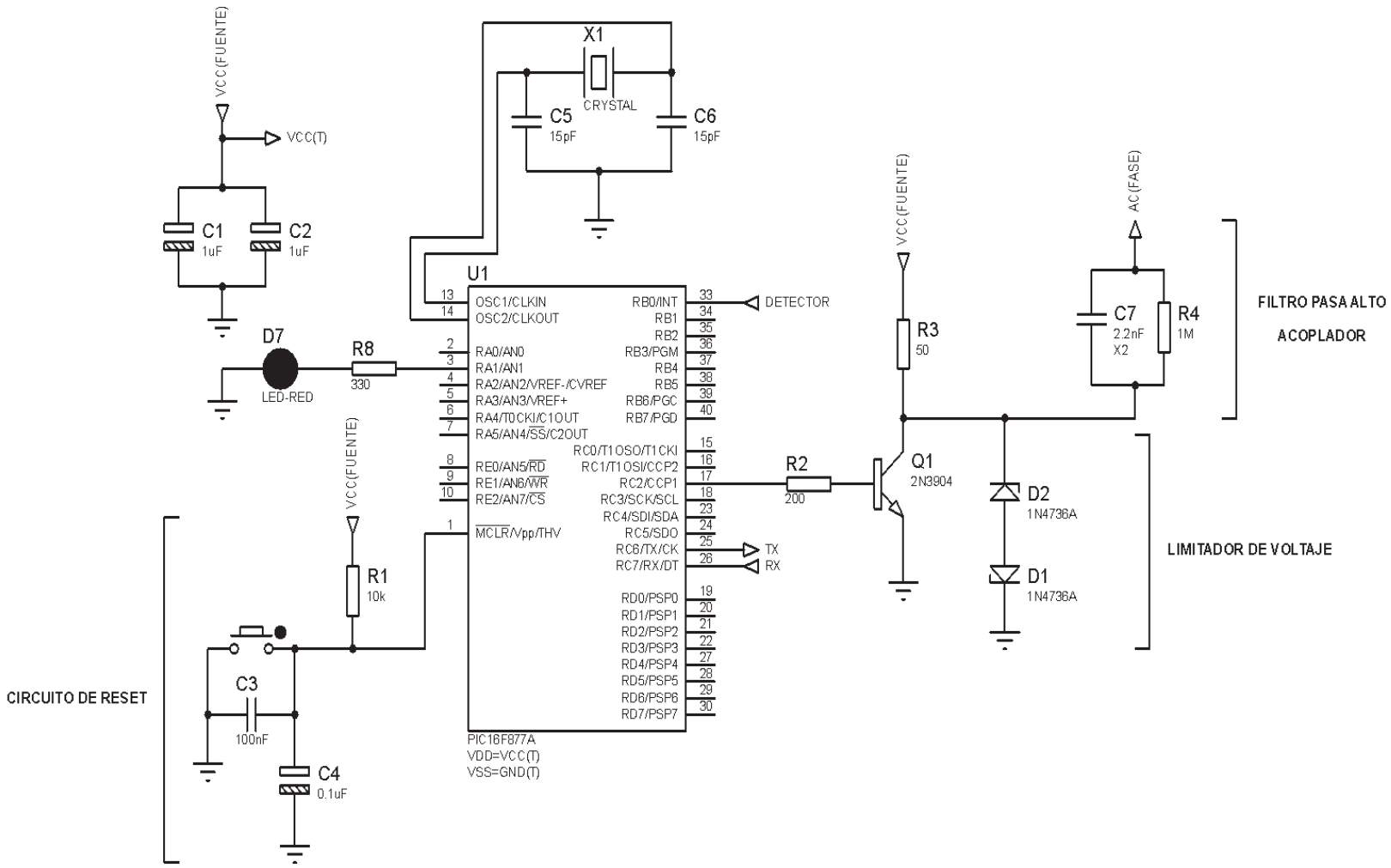


Figura 2.3: Circuito de transmisión de los mensajes X10

2.2.3 DISEÑO DEL CIRCUITO RECEPTOR X10 [9]

El circuito receptor de los códigos X10 que se construyó para el presente prototipo, es el utilizado por el fabricante Microchip en su publicación AN236, y emplea un microcontrolador PIC16F877A, tal y como lo recomienda Microchip tanto para el circuito transmisor como receptor de señales X10. De manera similar, la alimentación del circuito receptor proviene de la fuente que se construyó en la sección 2.2.7.1, y se utiliza el detector de cruces por cero del microcontrolador (PIN I/O - RB0) para sincronizar la recepción de los mensajes X10 con la red eléctrica.

Al igual que en el circuito transmisor, se emplea un arreglo de capacitores para evitar que el microcontrolador se vea afectado ante una variación brusca de voltaje de alimentación y adicionalmente se incluye un circuito de reset en caso de que se necesite resolver algún problema con el microcontrolador.

El proceso de recepción está constituido por varias etapas, todo empieza cuando el circuito receptor toma los mensajes X10 que se transportan por la red eléctrica, a continuación la información pasa por una resistencia limitadora de corriente y de esta manera pasa a las siguientes etapas que se describen a continuación:

Filtro Pasa Altos

Está constituido por un circuito RC, cuya frecuencia de corte está calculada de la siguiente manera:

$$f_C = \frac{1}{2\pi RC} = \frac{1}{2\pi * (33\ K\Omega) * (150\ pF)}$$

$$\boldsymbol{f_C = 32\ KHz}$$

Es decir, que este filtro permite el paso de señales de frecuencia mayores a 32 KHz. Por lo tanto, las señales X10 podrán pasar fácilmente a través del filtro, mientras que la señal eléctrica de 60 Hz no podrá pasar hacia el interior del circuito.

Amplificador Sintonizador [2], [8]

Este circuito consiste en un amplificador que usa como dispositivo principal un inversor CMOS (CD4069). Este dispositivo está formado internamente por 6 inversores CMOS (Ver Anexo 8), de los cuales se utilizan 2 para este amplificador sintonizador. A continuación en la figura 2.4 se observa el circuito integrado CD4069 y su constitución interna.

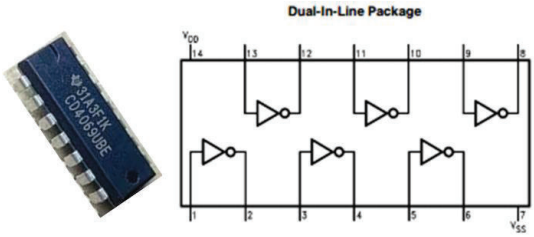


Figura 2.4: Circuito integrado CD4069 [2]

El principio de funcionamiento de amplificación de este dispositivo se basa en su comportamiento lineal cuando su voltaje de entrada y salida se igualan, para lo cual, con un voltaje de alimentación de 5 VDC se tiene la siguiente gráfica del voltaje de salida en función del voltaje de entrada que se muestra en la figura 2.5.

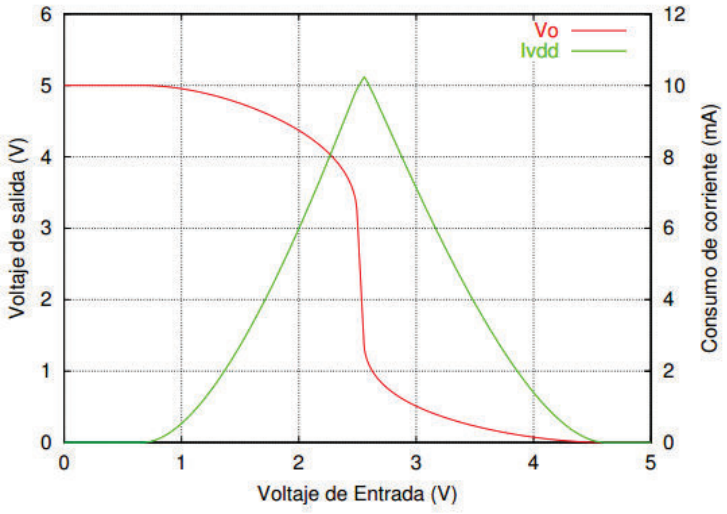


Figura 2.5: Voltaje de salida en función del voltaje de entrada [8]

En esta figura se puede observar que a 2.5 VDC de entrada la curva de salida presenta un comportamiento lineal, por lo tanto para asegurar este comportamiento se debe usar una realimentación de la salida a la entrada del inversor. Esta igualación de voltajes se produce cuando el voltaje de entrada y salida son igual al voltaje de alimentación sobre dos.

Cuando el inversor se encuentra en la zona lineal su comportamiento es bastante sensible a las variaciones de voltaje, permitiendo de esta manera la amplificación de pequeños valores de voltaje. La ganancia del amplificador varía típicamente entre -20 y -100, este valor depende del voltaje de entrada al inversor y de la pendiente de la zona lineal.

Para la sintonización de la frecuencia de los mensajes X10 (120 KHz), se utiliza un tanque LC, el cual está formado por elementos cuyos valores permiten obtener la siguiente frecuencia de sintonización:

$$f_S = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{(220\ uH) * (8nF)}}$$

$$\boldsymbol{f_S = 119,97\ KHz}$$

Puesto que comercialmente no se dispone de capacitores de 8nF, Microchip utiliza dos capacitores comerciales en paralelo (3,3 nF y 4,7 nF) para alcanzar el valor de capacitancia deseado. Además, dado que este circuito tanque solo permite el paso de señales con frecuencias próximas a 120 KHz, la realimentación necesaria para la amplificación se producirá solamente con las señales cuya frecuencia sea cercana a los 120 KHz anteriormente mencionados.

Para polarizar al inversor, el voltaje de alimentación de la fuente ingresa a través de una resistencia a la entrada del inversor, esto nos permite asegurar una retroalimentación directa desde la entrada a la salida del inversor sin alterar el funcionamiento del circuito tanque.

Microchip utiliza dos circuitos amplificadores sintonizadores, de tal manera que se asegura el paso de la señal X10 hacia la siguiente etapa de amplificación que se describe a continuación.

Amplificador

Esta etapa del circuito receptor se basa en el mismo principio del inversor CMOS utilizado por el amplificador sintonizador. En este caso se utilizan los siguientes dos inversores CMOS del CD4069 y una combinación RC que se encarga de permitir o impedir la realimentación para la amplificación de acuerdo a la señal de entrada del circuito. Por ejemplo, al tener una señal DC constante, el circuito RC impide el paso de voltaje a la entrada del inversor haciendo que este no trabaje en su zona lineal, por el contrario, cuando la señal X10 pasa por el circuito RC, obliga a la carga y descarga del capacitor, permitiendo el paso de la realimentación, haciendo que este trabaje en su zona lineal, y por lo tanto, amplifique dicha señal.

Este circuito de amplificación es importante puesto que la señal X10, al viajar por la red eléctrica, llega sumamente atenuada al receptor, por lo cual es importante que sea doblemente amplificada de tal manera que permita llevar a cabo la siguiente etapa del circuito receptor, que es el detector de envolvente que se describe a continuación.

Detector de envolvente

Este circuito se encarga de detectar la presencia o ausencia de la señal portadora (señal X10 de 120 KHz), emitiendo un 0L cuando no existe portadora y un 1L si existe la portadora.

Su funcionamiento se basa en un circuito RC, el cual toma como referencia la señal portadora de tal manera que mientras no haya señal X10, la entrada al circuito va a ser un voltaje constaste de 5 VDC, lo cual, provoca que el capacitor permanezca descargado, haciendo que el voltaje de la fuente entre al inversor, obteniendo de esta manera una señal de 0 VDC a la salida del detector, como se puede observar en la figura 2.6 a).

De lo contrario, al tener una señal portadora a la entrada del detector, este producirá una carga y descarga en el capacitor. Cuando se tiene un nivel bajo de la señal portadora provocará que el capacitor comience a cargarse, haciendo que todo el voltaje de la fuente se consuma en la resistencia de descarga durante su periodo, por lo tanto a la entrada del inversor se tiene un voltaje de 0 VDC, obteniendo a la salida un voltaje de 5 VDC como se observa en la figura 2.6 b).

Durante el nivel alto de la señal portadora se produce la descarga del capacitor, manteniendo de esta manera la señal a la entrada del inversor en 0 VDC puesto que hace que el voltaje de la fuente se consuma en la resistencia de descarga como se observa en la figura 2.6 c). Antes de que el capacitor se haya descargado totalmente y permitido el paso de la señal de 5 VDC de la fuente a la entrada del inversor, se produce el siguiente cambio de la señal portadora, provocando que el capacitor se vuelva a cargar permitiendo que el voltaje de la fuente siga siendo consumida por la resistencia de descarga tal y como se observa en la figura 2.6 d).

Luego de que culmina el paso de la señal portadora a través del circuito receptor, el voltaje a la entrada del detector de envolvente se restablece en 5 VDC, permitiendo la descarga final del capacitor y provocando que el voltaje de la fuente pase al inversor tal y como se muestra en la figura 2.6 e).

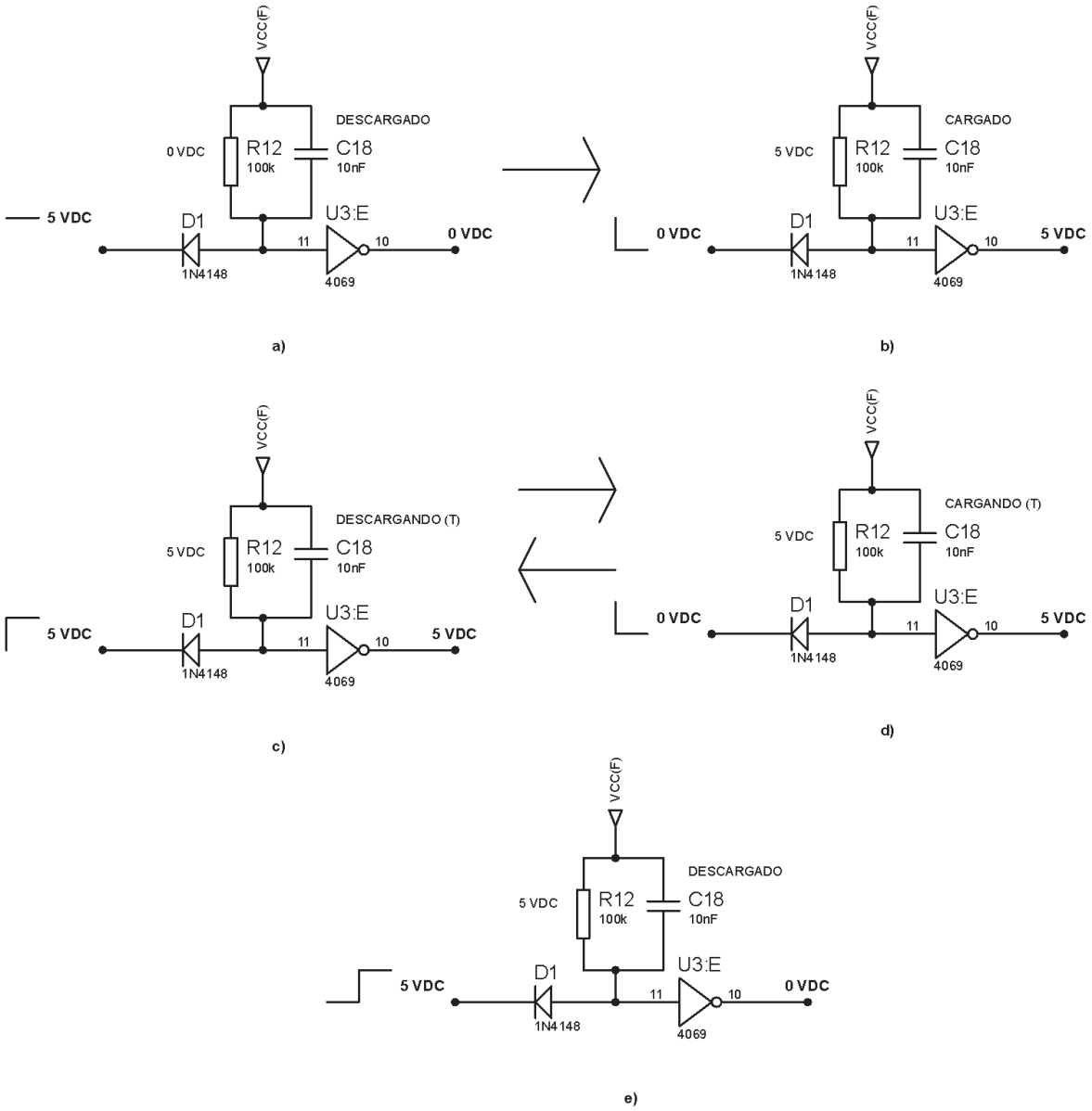


Figura 2.6: Comportamiento del circuito detector de envolvente

Para complementar el correcto funcionamiento del detector de envolvente y asegurar que la señal portadora solo sirva de referencia, se debe usar un diodo que impida el paso de la corriente desde las etapas amplificadoras, y que asegure el paso de la señal portadora. Para ello Microchip utiliza un diodo de alta velocidad, 1N4148 (Ver Anexo 9), en polarización inversa consiguiendo los objetivos descritos.

Finalmente, para culminar con el proceso de recepción, la señal de salida del detector de envolvente ingresa a un inversor del CD4069 cuya señal de salida finalmente ingresa al PIN 18 del PIC16F877A para ser procesada y enviada al respectivo actuador mediante comunicación serial.

A continuación en la figura 2.7, se muestra el circuito receptor X10 completo, incluyendo todas las etapas anteriormente descritas.

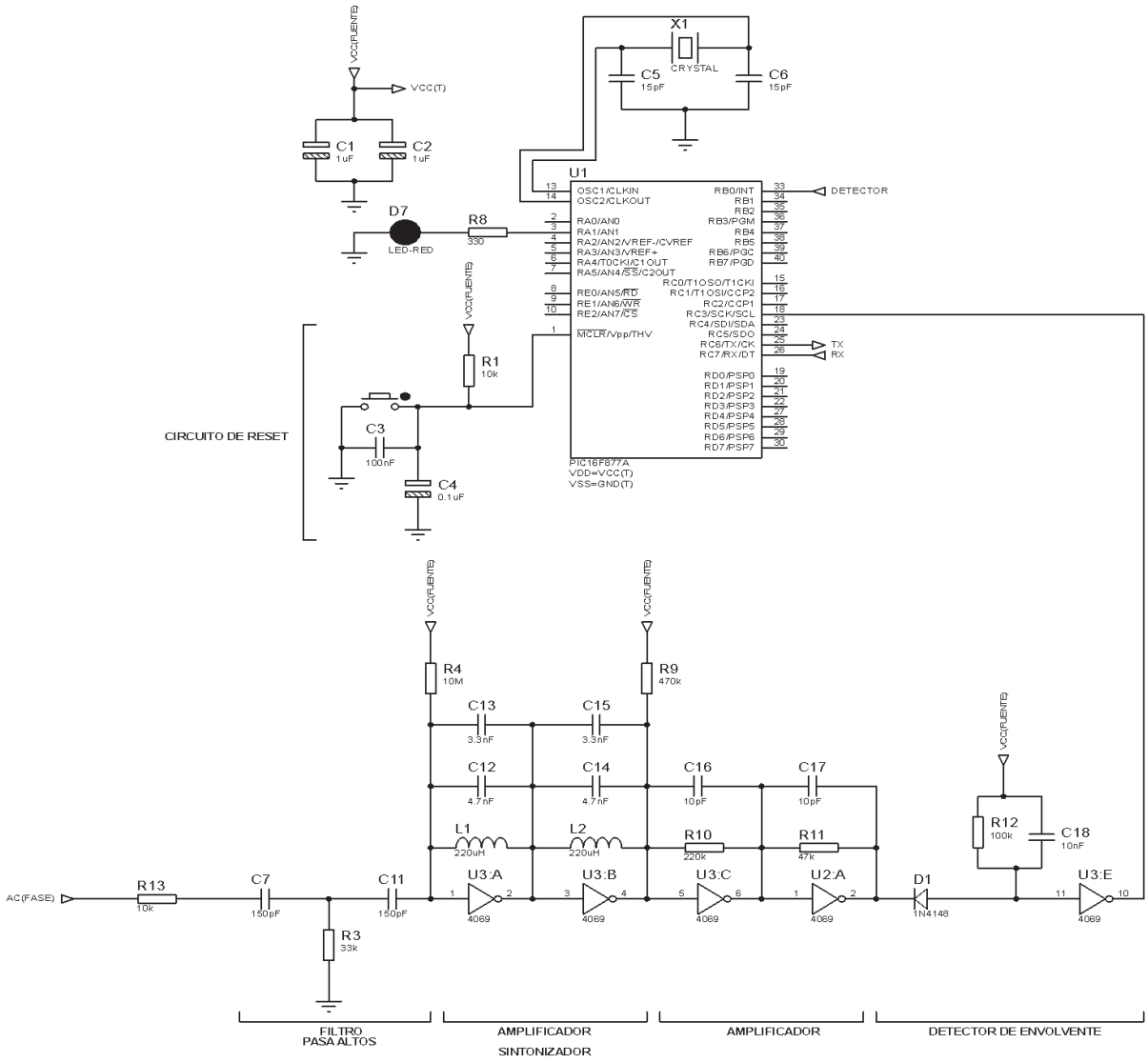


Figura 2.7: Circuito de recepción de los mensajes X10

2.2.4 DISEÑO DEL CIRCUITO DE CONTROL DE ILUMINACIÓN [6], [28]

El circuito para el control de iluminación está constituido por varias secciones, las cuales fueron diseñadas en base a las recomendaciones dadas por los fabricantes en las hojas de datos respectivas de los dispositivos que se emplearon en la construcción del circuito, además de tomar como referencia los apuntes de clases y los circuitos empleados para la construcción del transmisor y receptor de las señales X10.

El circuito de control de iluminación utiliza principalmente un microcontrolador ATMEGA8535, el cual, presenta unas características óptimas para llevar a cabo este control (Ver Anexo 10). El proceso de control de iluminación empieza cuando el microcontrolador recibe la instrucción a ser ejecutada y la procesa.

Para ejercer el control de luz se utiliza un circuito de control de fase directo y un PWM (Modulador por Ancho de Pulso), con lo cual se consigue recortar la onda sinusoidal de voltaje eléctrico dependiendo de la cantidad de luz que se desee tal y como se muestra en la figura 2.8.

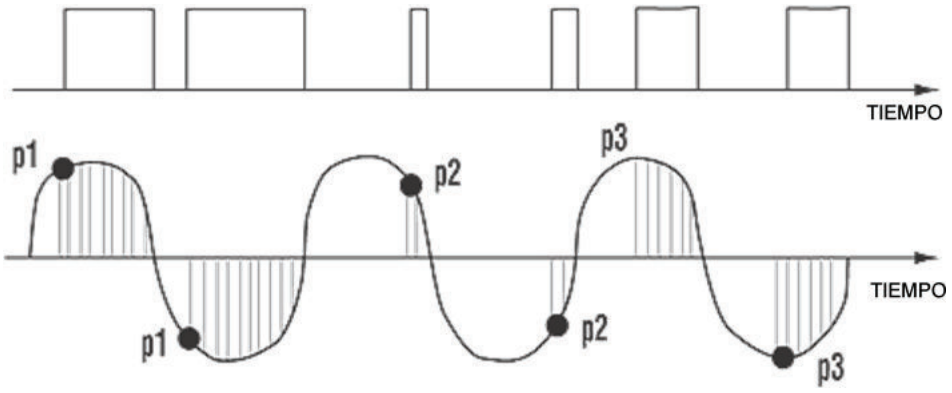


Figura 2.8: Control de la señal eléctrica mediante PWM [28]

El microcontrolador ATMEGA8535 se encarga de generar la señal del PWM de un periodo de la mitad de la señal de la red eléctrica, por lo tanto el periodo del PWM será de 8.33 ms, calculado de la siguiente manera:

$$T_{AC} = \frac{1}{f_{AC}} = \frac{1}{60\ Hz}$$

$$T_{AC} = 16{,}67\ mseg$$

$$T_{PWM} = \frac{T_{AC}}{2} = \frac{16{,}6\hat{6}\ mseg}{2}$$

$$\therefore \boldsymbol{T_{PWM} = 8{,}3\hat{3}\ mseg}$$

La razón por la cual se requiere que el periodo de la señal PWM, sea la mitad de la señal eléctrica, es para ejercer un control de la cantidad de señal eléctrica que se va a dejar pasar en cada semi-ciclo. Para que el control del PWM funcione correctamente, se debe sincronizar el inicio de esta señal con el cruce por cero de la señal eléctrica, para ello se utiliza el circuito detector de cruces por cero descrito en el literal 2.2.1.

El control del paso del voltaje de la red eléctrica hacia el foco se lo realiza mediante un triac BT136 (Ver Anexo 11), que es disparado por el control proveniente del PWM. Un triac es un elemento semiconductor bidireccional, utilizado como switch para corriente alterna. Este dispositivo consta de tres pines, como se muestra a continuación en la figura 2.9.

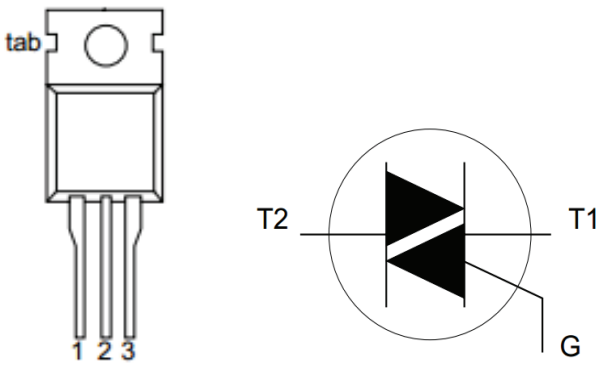


Figura 2.9: Triac BT136 [6]

Donde,

- T1 = Main Terminal 1

- T2 = Main Terminal 2
- G = Gate

Para permitir el paso de voltaje del T1 al T2 y viceversa, se debe aplicar una señal de control al G, la cual es un voltaje DC.

Cabe recalcar que en el sistema diseñado es importante aislar la parte de control y la parte de potencia, para ello se utiliza un optoaislador triac MOC3021 (Ver Anexo 12), éste recibe el control PWM por parte del ATMEGA8535, el cual pasa primero por un transistor de paso, permitiendo que no haya un alto consumo de corriente por parte del puerto del microcontrolador. Luego, dependiendo de la señal de control que entra al G, el triac BT136 permite o no el paso del voltaje de la red eléctrica hacia la carga conectada a su T1.

Además del control automático que ejecuta el sistema domótico, también se puede controlar la iluminación de forma manual. Este control lo realiza el usuario por medio del ATMEGA8535, pero en este caso el microcontrolador recibe el control por medio de su conversor A/D, al cual se encuentra conectado un potenciómetro que varía su voltaje de salida entre 0 VDC y 5 VDC. Para poder establecer la referencia del conversor A/D, se utilizan los pines de AREF y AVCC del microcontrolador, dando la referencia a tierra y a 5 VDC como lo recomienda la hoja de datos del microcontrolador en el Anexo 10.

Puesto que el uso del control manual por parte del usuario es aleatorio, es necesaria la utilización de un circuito que informe al microcontrolador que el tipo de control que el usuario va a ejecutar es manual. Para este efecto se usa un interruptor, el cual, activa la interrupción externa 1 del microcontrolador. La función que cumple el interruptor es generar un flanco de bajada o subida de la señal DC con lo cual, se dispara la interrupción externa en el microcontrolador. Esta función está habilitada hasta que el usuario vuelva a controlar la luz de forma remota, mediante el uso de la interfaz gráfica de usuario.

A continuación en la figura 2.10 se visualiza el circuito completo para el actuador de iluminación, el cual permite ejecutar un control tanto manual como remoto de todas las luces del sistema domótico.

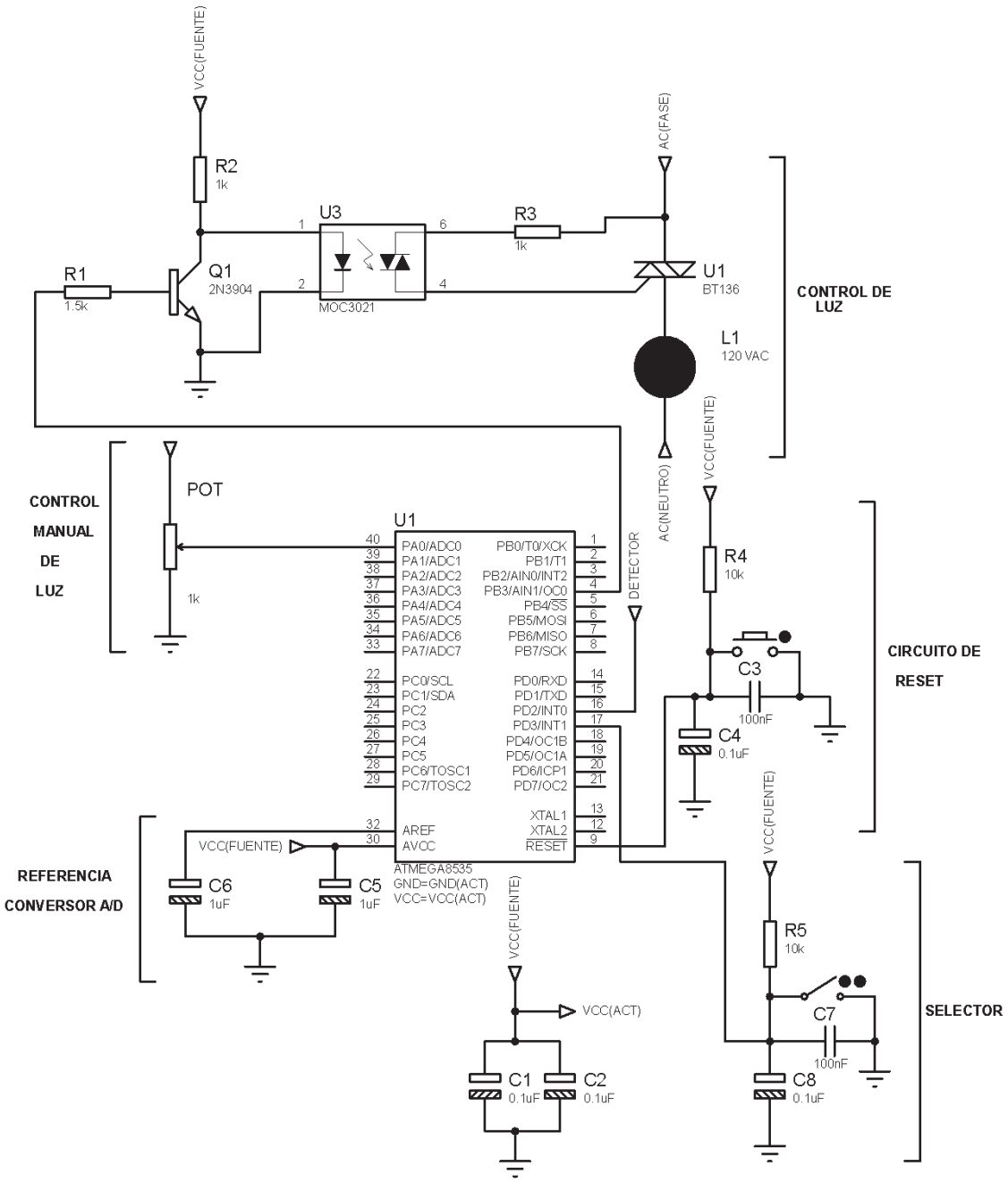


Figura 2.10: Circuito actuador de iluminación

2.2.5 DISEÑO DEL CIRCUITO DE CONTROL DE PERSIANAS Y PUERTAS [1], [44]

Para el funcionamiento de las persianas y puertas, se hace uso de motores a pasos (28BYJ-48), los cuales son controlados por un microcontrolador ATMEGA8535 que recibe los comandos vía comunicación serial desde su respectivo dispositivo receptor X10. A continuación en la figura 2.11 se observa el motor a pasos que se utiliza en el sistema domótico, cuyas especificaciones técnicas se encuentran en la hoja de datos del motor, adjunto en el Anexo 13.

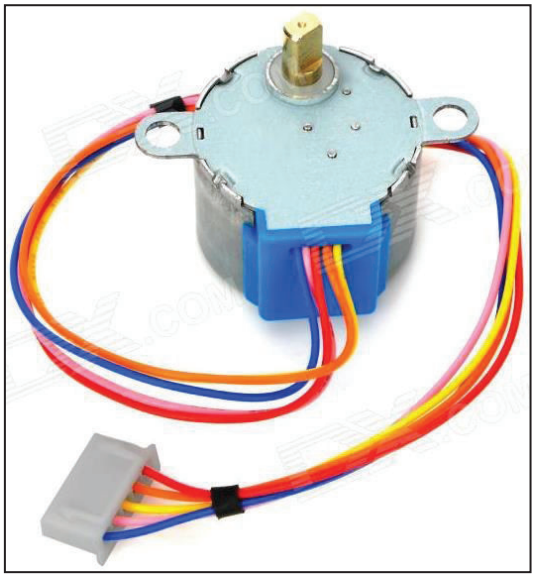


Figura 2.11: Motor a pasos 28BYJ-48 [1]

La corriente que necesita cada bobina del motor, se calcula de la siguiente forma:

$$R_L = 26\ \Omega$$

$$I_L = \frac{V_{Activación}}{R_L} = \frac{5\ VDC}{26\ \Omega}$$

$$\boldsymbol{I_L = 192\ mA}$$

Puesto que los puertos del microcontrolador ATMEGA8535 no pueden abastecer de suficiente corriente para el funcionamiento del motor a pasos, se hace uso de un

driver amplificador de corriente L293D (Ver Anexo 14), con el fin de alimentar al motor con suficiente corriente para que éste se pueda mover. Cabe resaltar que la forma de conexión del motor al driver es la recomendada por el fabricante en la hoja de datos del Anexo 14.

El motor a pasos que se utiliza en el sistema domótico es un motor unipolar, por lo cual tiene 5 pines, 4 pines para los pasos y un pin común, éste último se debe conectar a tierra. Los pasos del motor se activan cuando el puerto del microcontrolador, al cual se encuentran conectados los pines del motor, presentan salidas de 1L en la secuencia correcta, permitiendo de esta manera el movimiento del motor.

Para dar mayor torque a los pasos del motor se utiliza secuencias de dos fases, las cuales consisten en activar dos bobinas a la vez, como se observa en la figura 2.12, donde se observa que inicialmente se activa A y B, a continuación se desactiva A y se activa B y C, luego se desactiva B y se activa C y D y finalmente se desactiva C y se activa D y A y así sucesivamente.

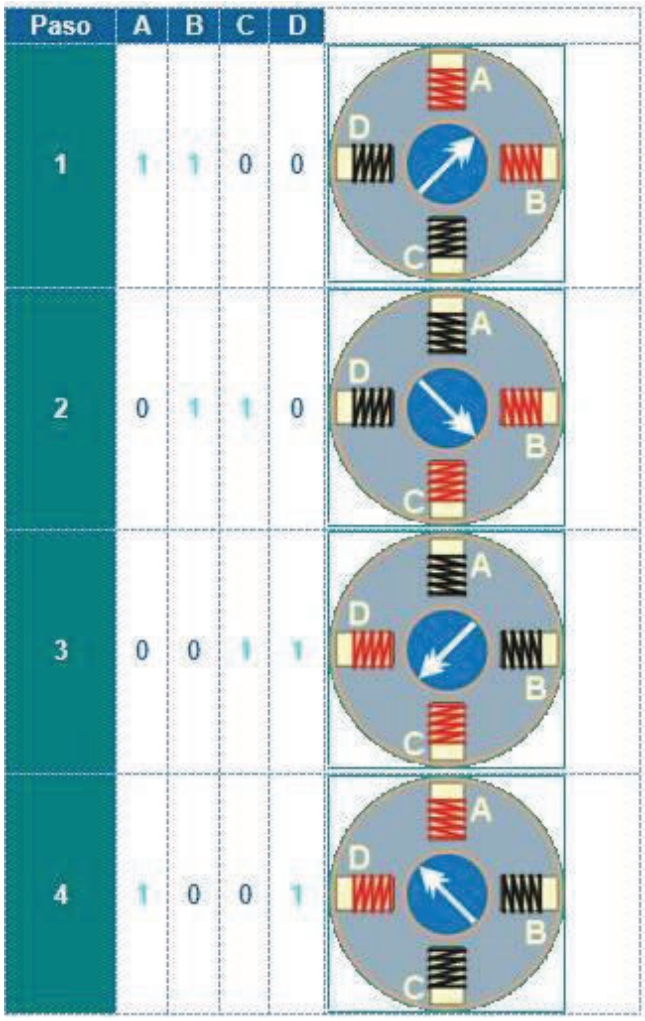


Figura 2.12: Secuencia de paso doble para un motor a pasos [44]

Finalmente en la figura 2.13 se muestra el circuito completo para el control de puertas y persianas, incluido el driver y el motor a pasos.

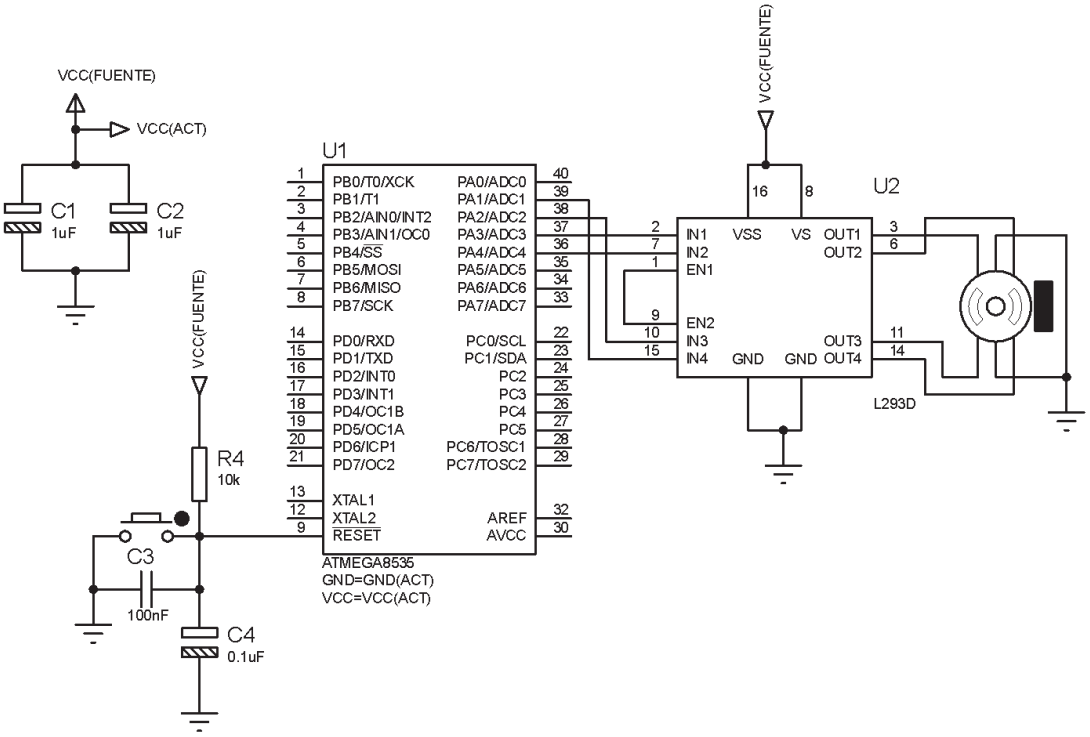


Figura 2.13: Circuito actuador de puertas y persianas

Para que un receptor pueda controlar varios actuadores, el sistema está diseñado para que cada actuador utilice su puerto USART para transmitir vía comunicación serial, el comando recibido previamente. En el prototipo de sistema domótico se implementó este procedimiento para el control de iluminación y accesos en el garaje, de tal forma que un solo receptor puede enviar el mismo comando a varios actuadores.

2.2.6 DISEÑO DEL CIRCUITO PARA LA INTERFAZ DE COMUNICACIÓN SERIAL

Puesto que en el sistema domótico se requiere comunicación serial entre dispositivos que trabajan con diferentes niveles de voltaje, sea RS-232 o TTL, es necesario el uso de una interfaz convertidora de niveles de voltaje. Para este propósito, se usa el circuito integrado MAX232, el cual, convierte las señales con niveles de voltaje RS-232 a TTL y viceversa.

En el sistema domótico existe comunicación serial de diferentes niveles de voltaje entre los siguientes dispositivos:

- Comunicación serial entre el FPGA y el Transmisor X10.

Tomando en cuenta, que los puertos seriales del FPGA trabajan a niveles de voltaje RS-232, y el microcontrolador del transmisor X10 trabajan con niveles de voltaje TTL, a continuación en la figura 2.14 se muestra la conexión que se realizó entre los diferentes dispositivos y el MAX232 con el fin de permitir una comunicación serial exitosa:

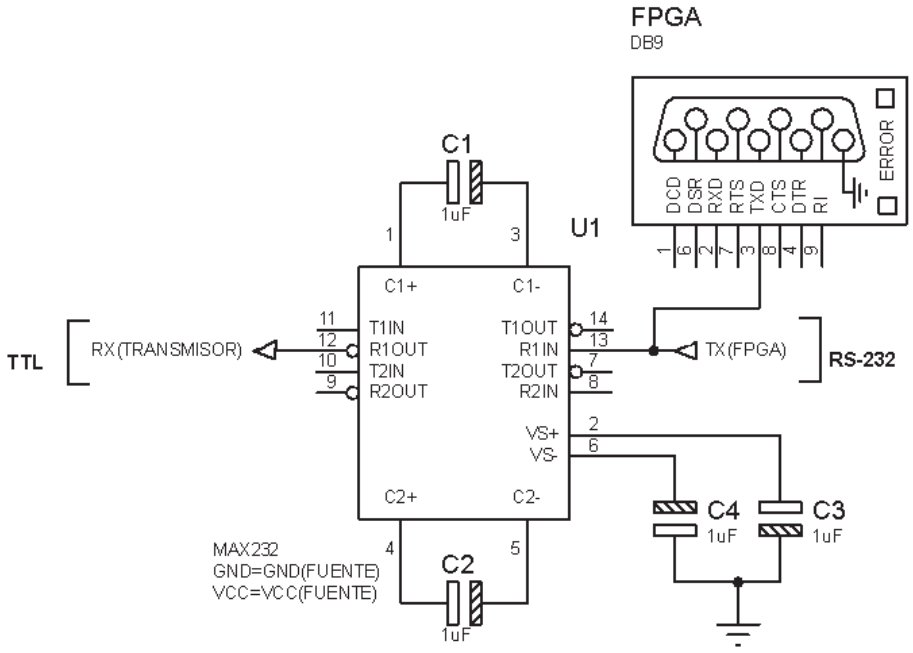


Figura 2.14: Circuito de la interfaz de comunicación serial usando un MAX232

Cabe recalcar que las conexiones para el funcionamiento del circuito integrado MAX232 son las recomendadas por el fabricante en su hoja de datos (Ver Anexo 15).

En el lado del FPGA los pines de transmisión corresponden a los pines del conector DB9 macho, es decir:

- Pin 3: Transmisión
- Pin 5: Tierra

De esta forma se consiguió exitosamente que el sistema domótico se pueda comunicar desde la parte del usuario hasta el circuito actuador, sin presentar problemas por diferencia de niveles de voltaje.

2.2.7 DISEÑO DE LOS CIRCUITOS DE LAS FUENTES DE ALIMENTACIÓN

Los circuitos descritos anterioremente requieren de fuentes de alimentación de 5 VDC, por ello se optó por la implementación de dos tipos de fuentes, una de las cuales utiliza un transformador mientras que la otra no lo hace. La razón por la cual se optó por dos fuentes de alimentación es por requerimientos de corriente y tamaño de cada circuito.

2.2.7.1 Diseño de la fuente de alimentación de los circuitos transmisor y receptores.

Para la alimentación de los circuitos transmisores y receptores se usa una fuente con batería y regulador de voltaje. Las ventajas de esta fuente son su bajo costo, tamaño y facilidad de implementación. Por otro lado, esta fuente no provee altos valores de corriente estando en el orden de pocos cientos de miliamperios, con lo cual es suficiente para el funcionamiento correcto de los circuitos de transmisión y recepción de las señales X10.

A continuación en la figura 2.15 se muestra la fuente utilizada, la cual, consta de tres partes principales:

- Voltaje de Entrada.
- Regulación del Voltaje.
- Voltaje de Salida.

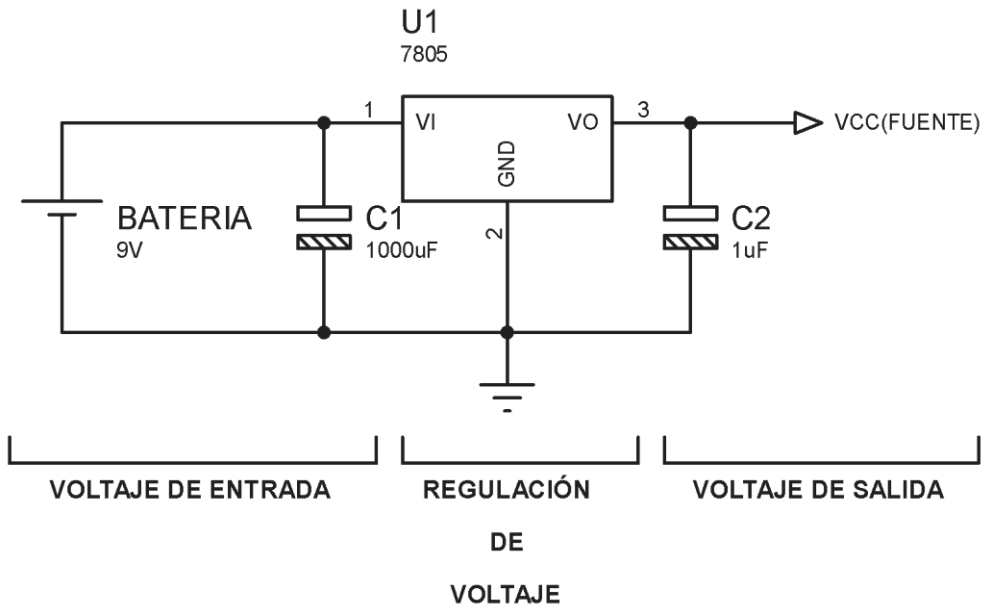


Figura 2.15: Fuente DC

Basados en la configuración recomendada por el fabricante del circuito integrado LM7805 (Ver Anexo 5), se usan capacitores electrolíticos a la entrada y salida del regulador de voltaje con el objetivo de almacenar energía que ayude a regular las variaciones instantáneas y la estabilidad del voltaje, y reducción del ruido.

2.2.7.2 Diseño de la fuente de alimentación de los circuitos actuadores.

Dado que los circuitos actuadores van a manejar cargas más altas que la de los transmisores y receptores, se optó por utilizar el diseño de fuente que provee mayor corriente a los circuitos. Este diseño incluye el uso de un transformador, el cual reduce el voltaje de la red eléctrica (120 VAC) a 12 VAC.

Este voltaje pasa por el proceso de rectificación de onda completa, para lo cual se usa un puente de diodos rectificadores. A continuación, se filtra la señal para disminuir el rizado y finalmente se usa el circuito integrado LM7805 para regular la tensión de salida.

En la figura 2.16 se puede observar la fuente de alimentación con transformador, la cual va a proporcionar 5 VDC a los circuitos actuadores, los cuales corresponden a

cargas resistivas e inductivas con un requerimiento mayor de corriente que cualquier otro circuito del sistema.

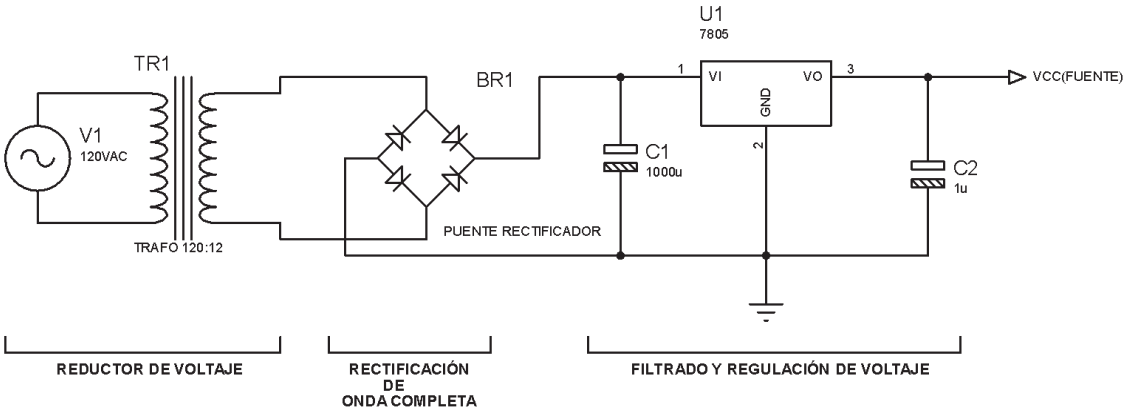


Figura 2.16: Fuente con transformador

2.3 DISEÑO DE LOS PROGRAMAS DEL PROTOTIPO

En esta sección se describe el diseño de todos los programas que, junto a los circuitos anteriormente descritos, permiten el óptimo desempeño de nuestro sistema domótico.

El programa principal de codificación X10, se genera en el controlador, que en nuestro caso es la tarjeta Spartan 3E Starter Board de Xilinx, a continuación la información X10 pasa al transmisor X10, el cual la coloca en la red eléctrica para que el receptor X10 la reciba y la envíe a sus respectivos actuadores. Todo esto es posible mediante la correcta programación del FPGA y de los microcontroladores que componen a todo nuestro sistema domótico.

A continuación se procede a describir cada uno de los programas desarrollados y su respectiva función dentro del sistema.

2.3.1 DISEÑO DEL PROGRAMA PARA EL CONTROL DOMÓTICO EN EL FPGA

Para la programación del FPGA se utilizó el paquete ISE Design Suite 14.5 de Xilinx. Antes de leer la descripción de los programas para el control domótico, se

recomienda revisar el Anexo 16, en donde se detalla el proceso de configuración de un programa utilizando ISE Design Suite 14.5 de Xilinx.

El control domótico que usamos en el prototipo está diseñado por completo en lenguaje VHDL. Comprende funciones básicas de domótica para una vivienda, tales como el control de iluminación, persianas y puertas, ya sea en tiempo real o funciones programadas según lo desee el usuario.

El FPGA recibe las instrucciones provenientes de una interfaz de usuario desarrollada en Labview, codifica dichas instrucciones con el protocolo de domótica X10 y las envía al transmisor X10.

A continuación se detalla cada función que se ejecuta en el FPGA.

2.3.1.1 Diseño del programa para el control de iluminación, accesos y persianas en tiempo real.

Con la finalidad de facilitar al usuario su desplazamiento y comodidad dentro del hogar, mediante el uso de la interfaz gráfica, se pueden realizar las siguientes funciones:

- Encender ó apagar las luces de todos los lugares de su vivienda en el instante deseado.
- Abrir ó cerrar las persianas de la sala y de las habitaciones de su vivienda en el instante deseado.
- Abrir ó cerrar la puerta del garaje y de la entrada en el instante deseado.

2.3.1.1.1 Descripción del diseño del programa para el control de iluminación, accesos y persianas en tiempo real.

Para cumplir los requerimientos anteriormente propuestos para la iluminación, persianas y puertas, a continuación en la figura 2.17, se observa la lógica con la que trabaja el FPGA para poder realizar este control dentro del sistema domótico.

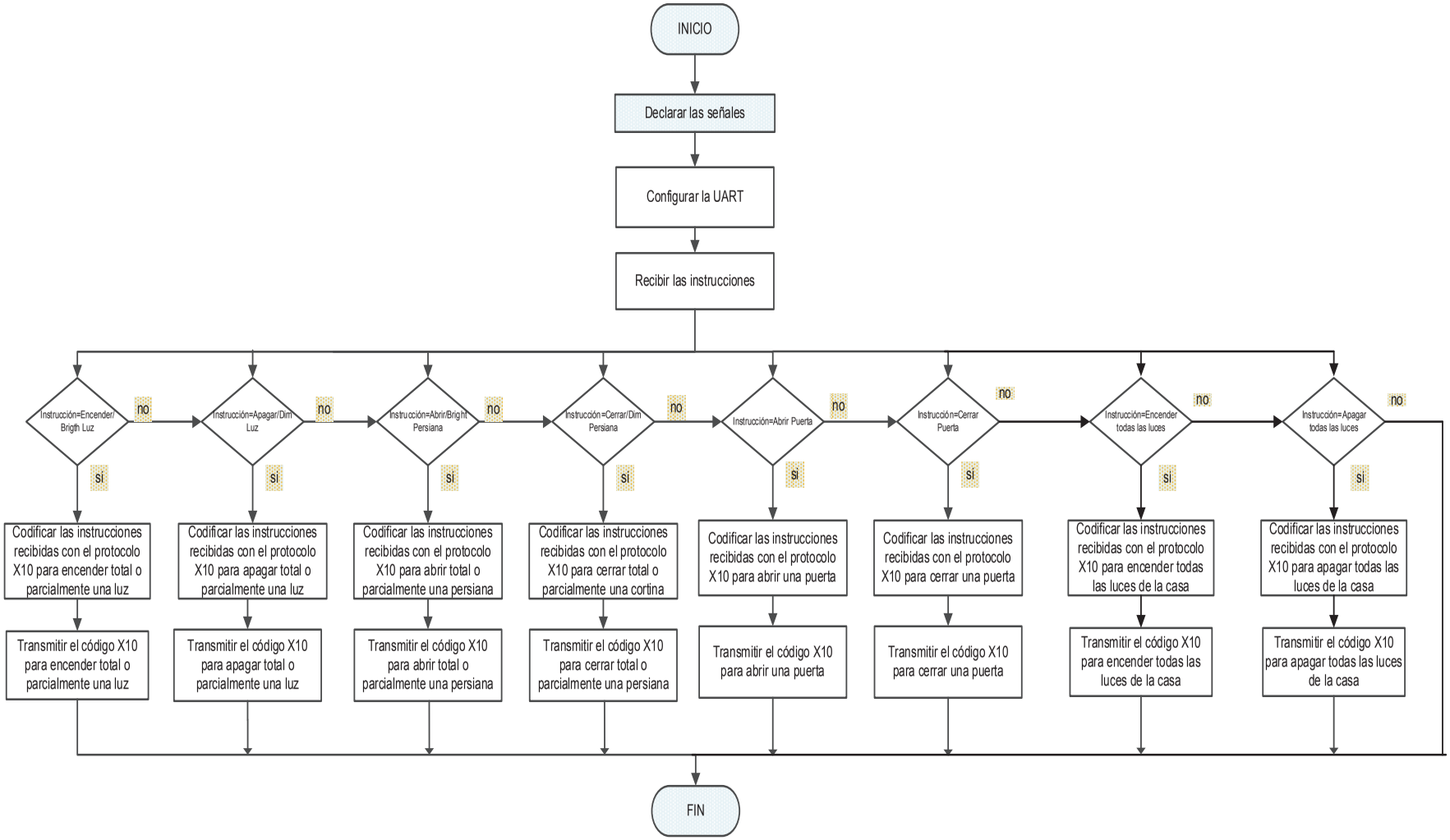


Figura 2.17: Diagrama de flujo del programa de control de iluminación, accesos y persianas en tiempo real

Con la finalidad de comprender el diagrama de flujo mostrado, a continuación se detallan cada uno de sus componentes.

1. Declaración las señales

Consiste en la asignación de los nombres y tipos de dato que tendrán las diferentes señales que utilizamos a lo largo del programa.

2. Configuración de la UART [11]

UART significa Universal Asynchronous Receiver and Transmitter, y permite la transmisión y recepción de datos de manera asíncrona y un caracter a la vez.

Trabajar de manera asíncrona significa que no se envían señales de reloj ni señales de sincronización en la información, sino mas bien, tanto el receptor como el transmisor se sincronizan de manera interna cuando se recibe ó se transmite el bit de inicio de cada caracter, el cual está formado por un número pequeño de bits dependiendo del código que se vaya a utilizar, en el presente proyecto se utilizan caracteres de 8 bits.

Es esencial que tanto el transmisor como el receptor se encuentren trabajando a la misma velocidad, en nuestro caso dicha velocidad es de 9600 baudios. Una vez configurada la UART, permitirá ejecutar todos los procesos de transmisión y recepción en el FPGA.

2.1 Diseño del Receptor

Para diseñar el receptor se debe tomar en cuenta que todo el Proyecto va a trabajar a una velocidad de 9600 baudios, es por ello que en primer lugar vamos a generar un reloj que permita al receptor trabajar a esta velocidad.

Este proceso se caracteriza porque recibe uno a uno los bits y una vez culminada la recepción, se ensambla el dato completo. Por esta razón, es sumamente

importante tomar en cuenta que se debe analizar minuciosamente cada bit recibido. Para ello, el reloj del receptor debe estar diseñado, de tal forma que permita efectuar un muestreo de 8, de tal manera que se pueda recibir cada bit 8 veces, obteniendo de esta manera el bit correcto que fue enviado al receptor hasta finalizar todo el proceso, donde están incluidos el bit de inicio, bits de datos y bit de parada.

El reloj interno de la tarjeta Spartan-3E Starter Board es de 50 MHz, entonces nos basamos en este valor para poder generar el reloj de recepción de datos. Para ello definimos un contador que permita determinar cuántos ciclos del reloj del FPGA (50 MHz) se deben tomar para poder generar un ciclo del reloj de recepción, el cual debe trabajar a 9600 baudios y además ejecutar 8 veces cada muestreo. A continuación se muestran los cálculos realizados para determinar este contador:

$$contador = \frac{\frac{frecuencia\ del\ reloj\ FPGA}{velocidad\ de\ recepción}}{Número\ de\ veces\ del\ muestreo}$$

$$contador = \frac{\frac{50MHz}{9600\ Baudios}}{8}$$

$$contador = \frac{50MHz}{9600\ Baudios * 8}$$

$$contador = \frac{50MHz}{76800}$$

$$contador = 651$$

$$contador\ recepción = \frac{contador}{2}$$

$$contador\ recepción = \frac{651}{2}$$

$$\boldsymbol{contador\ recepción = 325}$$

Este valor se representa en la figura 2.18 en donde se puede observar que los 651 ciclos que se toman del reloj del FPGA permiten recibir 8 veces un bit. Además cabe recalcar que estos 651 ciclos se deben dividir para 2, de tal manera que se pueda generar un ciclo completo del reloj de recepción, es decir se toman 325 ciclos del reloj del FPGA para la parte positiva del ciclo del reloj de recepción y 325 para la parte negativa del mismo:

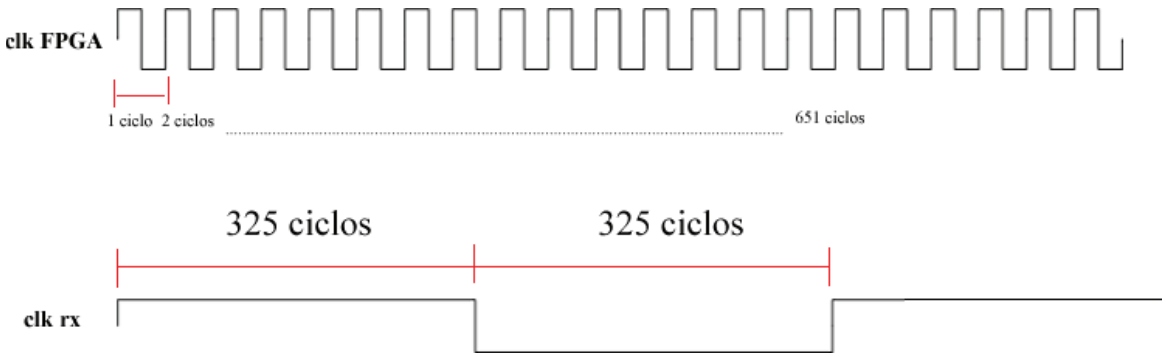


Figura 2.18: Diseño del reloj de recepción

El programa que se desarrolla en el FPGA para generar el reloj de recepción consiste en asignarle al contador de recepción el valor de 325 de tal manera que cada vez que se cuente un ciclo del reloj del FPGA se disminuye en 1 el contador de recepción hasta llegar a 0, una vez que esto sucede se asigna '1' o '0' a la señal *"clk rx",* que es la señal que guarda los ciclos del reloj de recepción.

Este proceso se repite indefinidamente, consiguiendo de esta manera el reloj que permite la recepción a 9600 baudios de cualquier dato proveniente del usuario.

2.2 Diseño del Transmisor

En primer lugar para poder diseñar el transmisor se debe tomar en cuenta que todo el Proyecto va a operar a una velocidad de 9600 baudios, es por ello que en

primer lugar vamos a generar un reloj que le permita al transmisor enviar datos a esta velocidad.

El reloj interno de la tarjeta Spartan-3E Starter Board es de 50 MHz, entonces nos basamos en este valor para poder generar elreloj de transmisión de datos. Para ello definimos un contador que permita determinar cuántos ciclos del reloj del FPGA (50 MHz) se deben tomar para generar un ciclo del reloj de transmisión, el cual debe operar a 9600 baudios. A continuación se muestran los cálculos realizados para determinar este contador:

$$contador = \frac{frecuencia\ del\ reloj\ FPGA}{velocidad\ de\ transmisión}$$

$$contador = \frac{50MHz}{9600\ Baudios}$$

$$contador = 5208$$

$$contador\ transmisión = \frac{contador}{2}$$

$$contador\ transmisión = \frac{5208}{2}$$

$$\boldsymbol{contador\ transmisión = 2604}$$

Este valor se representa en la figura 2.19 en donde se puede observar que los 5208 ciclos que se toman del reloj del FPGA permiten el envío de un bit. Además cabe recalcar que estos 5208 ciclos se deben dividir para 2, de tal manera que se pueda generar un ciclo completo del reloj de transmisión, es decir se toman 2604 ciclos del reloj del FPGA para la parte positiva del ciclo del reloj de transmisión y 2604 para la parte negativa del mismo.

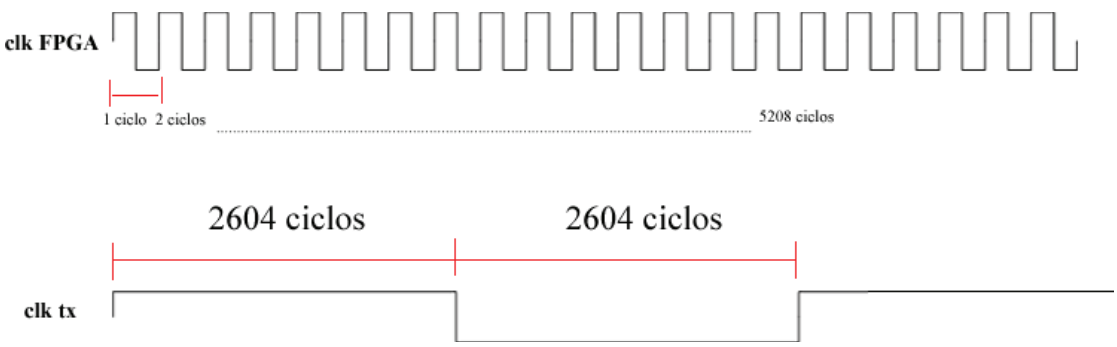


Figura 2.19: Diseño del reloj de transmisión

El programa que se desarrolla en el FPGA para generar el reloj de transmisión consiste en asignarle al contador de transmisión el valor de 2604 de tal manera que cada vez que se cuente un ciclo del reloj del FPGA se disminuye en 1 el contador de transmisión hasta llegar a 0, una vez que esto sucede se asigna '1' ó '0' a la señal *"clk tx"*, que es la señal que guarda los ciclos del reloj de transmisión.

Este proceso se repite indefinidamente, consiguiendo de esta manera el reloj que permite la transmisión a 9600 baudios de cualquier código X10.

3. Recepción de las instrucciones

Una vez configurado el reloj para la recepción de datos, se procede a usar este reloj para recibir las instrucciones del usuario.

En la figura 2.20 se observa el formato de recepción de un dato de forma asíncrona, el cual consiste en recibir 8 veces cada bit. Para ello en primer lugar se recibe el bit de inicio, el cual es un '0', seguido por los 8 bits de datos y finalmente se recibe un bit de parada, el cual es un '1'. Una vez recibido el bit de parada, el receptor entra en reposo y se encuentra a la espera de recibir los próximos datos provenientes del usuario.

Gracias al muestreo que se realiza en este proceso, se garantiza la recepción confiable de la información.

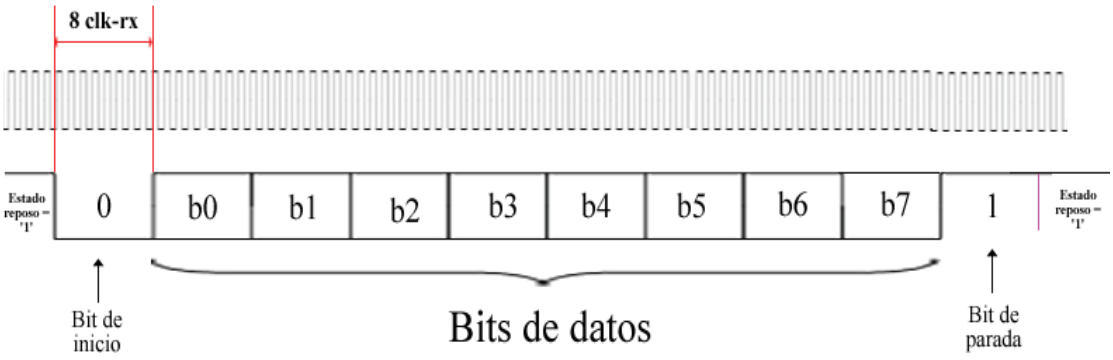


Figura 2.20: Formato de recepción asíncrono

Basándonos en este formato, el programa que se desarrolla en el FPGA para recibir las instrucciones comienza detectando el bit de inicio '0', una vez detectado el bit de inicio se activa en '1' la señal *"act"*, lo cual significa que el receptor se encuentra ocupado recibiendo un dato. Mientras esta señal se encuentra en '1' el receptor va recibiendo bit a bit hasta que ha recibido los 10 bits correspondientes a:

- 1 bit de inicio.
- 8 bits de datos.
- 1 bit de parada.

Una vez que recibió el bit de de parada, la señal *"act"* se pone en '0', lo cual significa que ha culminado la recepción. Entonces se activa la señal *"listo"* en '1', indicando que el dato está listo para usarse internamente en los procesos requeridos.

En este caso el FPGA recibe el dato correspondiente a *"instrucción",* la cual representa a la función que el usuario desea que se ejecute. Cada vez que se recibe un dato, éste se guarda en los respectivos registros para poder usarlos posteriormente y se desactiva la señal *"listo",* permitiéndole al receptor reposar y disponerse a recibir en cualquier momento un nuevo dato.

En la figura 2.21 se muestra cómo es el proceso de recepción de un dato, el cual fue anteriormente descrito. Una vez completa la señal "*dato_aux",* se procede a guardar el dato de 8 bits en el registro *"instrucción".*

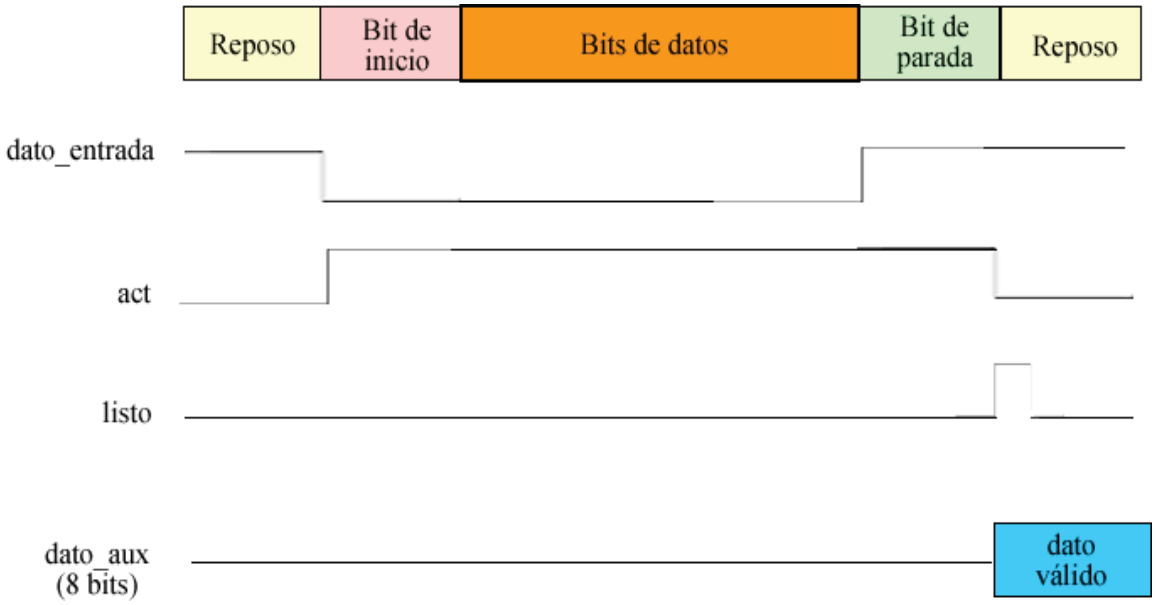


Figura 2.21: Comportamiento de las señales de recepción del programa

4. Identificación de la instrucción recibida

Una vez que se ha recibido serialmente la instrucción, se procede a identificar la función a la cual corresponde la señal recibida, las cuales pueden ser:

- Encender, apagar, bright o dim una determinada luminaria.
- Abrir, cerrar, bright o dim una determinada persiana.
- Abrir ó cerrar las puertas.
- Encender o apagar todas las luces de la casa.

Las instrucciones que se pueden recibir y sus correspondientes funciones se muestran a continuación en la tabla 2.2:

Tabla 2.2: Instrucciones para el sistema de control domótico

FUNCIÓN	DISPOSITIVO	LUGAR	INSTRUCCIÓN
Subir	Persiana	Sala	10100001
Bajar	Persiana	Sala	01010001
Subir	Luminaria	Sala	10010001
Bajar	Luminaria	Sala	01100001
Abrir	Persiana	Sala	11000001
Cerrar	Persiana	Sala	00110001
Encender	Luminaria	Sala	00000001
Apagar	Luminaria	Sala	11110001
Subir	Persiana	Cuarto1	10100010
Bajar	Persiana	Cuarto1	01010010
Subir	Luminaria	Cuarto1	10010010
Bajar	Luminaria	Cuarto1	01100010
Abrir	Persiana	Cuarto1	11000010
Cerrar	Persiana	Cuarto1	00110010
Encender	Luminaria	Cuarto1	00000010
Apagar	Luminaria	Cuarto1	11110010
Subir	Persiana	Cuarto2	10100011
Bajar	Persiana	Cuarto2	01010011
Subir	Luminaria	Cuarto2	10010011
Bajar	Luminaria	Cuarto2	01100011
Abrir	Persiana	Cuarto2	11000011
Cerrar	Persiana	Cuarto2	00110011
Encender	Luminaria	Cuarto2	00000011
Apagar	Luminaria	Cuarto2	11110011
Subir	Luminaria	Cocina	10010100
Bajar	Luminaria	Cocina	01100100
Encender	Luminaria	Cocina	00000100
Apagar	Luminaria	Cocina	11110100
Subir	Luminaria	Baño	10010101
Bajar	Luminaria	Baño	01100101
Encender	Luminaria	Baño	00000101
Apagar	Luminaria	Baño	11110101
Subir	Luminaria	Atico	10010110

Bajar	Luminaria	Atico	01100110
Encender	Luminaria	Atico	00000110
Apagar	Luminaria	Atico	11110110
Encender	Luminaria	Entrada	00000111
Apagar	Luminaria	Entrada	11110111
Abrir	Puerta	Entrada	11100111
Cerrar	Puerta	Entrada	00010111
Abrir/Encender	Puerta/Luminaria	Garaje	11101000
Cerrar/Apagar	Puerta/Luminaria	Garaje	00011000
Encender	Lámpara	Casa	00001001
Apagar	Lámpara	Casa	11111001

Por ejemplo, si el usuario desea encender la luminaria de la sala, el FPGA recibirá serialmente la instrucción *"00000001"* que está asignada específicamente a dicha función.

5. Codificación de las instrucciones recibidas con el protocolo X10

Cuando el FPGA ha identificado la función a la cual corresponde la instrucción recibida, procede a codificar dicha instrucción con el código X10.

Esta codificación consiste en asignar tres códigos, los cuales corresponden al *Código de Casa, Código de Unidad y Código de Función* respectivamente.

En la tabla 1.1, mostrada en la sección 1.2.5, se observan los códigos X10 que se deben asignar como códigos de casa.

Igualmente, en las tablas 1.2 y 1.3, mostradas en la sección 1.2.5, se observan los códigos X10 que se deben asignar como códigos de unidad y de función respectivamente.

En base a los códigos X10 que se muestran en las tablas 1.1, 1.2 y 1.3, a continuación en la tabla 2.3 se presenta la codificación X10 que se realizó en el

prototipo de sistema domótico, y que busca satisfacer la totalidad de las principales necesidades de un usuario.

Tabla 2.3: Códigos X10 del prototipo de sistema domótico

HOUSE CODES	KEY CODES	FUNCTION CODES	LUGAR	CODIFICACIÓN X10		
				CODIGO 1	CODIGO 2	CODIGO 3
A	1	ON	LUZ SALA	01101001	01101001	01011001
A	1	OFF	LUZ SALA	01101001	01101001	01011010
A	1	BRIGHT	LUZ SALA	01101001	01101001	01100110
A	1	DIM	LUZ SALA	01101001	01101001	01100101
A	2	BRIGHT	PERSIANA SALA	01101001	10101001	01100110
A	2	DIM	PERSIANA SALA	01101001	10101001	01100101
B	3	ON	LUZ CUARTO 1	10101001	01011001	01011001
B	3	OFF	LUZ CUARTO 1	10101001	01011001	01011010
B	3	BRIGHT	LUZ CUARTO 1	10101001	01011001	01100110
B	3	DIM	LUZ CUARTO 1	10101001	01011001	01100101
B	4	ON	PERSIANA CUARTO 1	10101001	10011001	01011001
B	4	OFF	PERSIANA CUARTO 1	10101001	10011001	01011010
B	4	BRIGHT	PERSIANA CUARTO 1	10101001	10011001	01100110
B	4	DIM	PERSIANA CUARTO 1	10101001	10011001	01100101
C	5	ON	LUZ CUARTO 2	01011001	01010110	01011001
C	5	OFF	LUZ CUARTO 2	01011001	01010110	01011010
C	5	BRIGHT	LUZ CUARTO 2	01011001	01010110	01100110
C	5	DIM	LUZ CUARTO 2	01011001	01010110	01100101
C	6	ON	PERSIANA CUARTO 2	01011001	10010110	01011001
C	6	OFF	PERSIANA CUARTO 2	01011001	10010110	01011010
C	6	BRIGHT	PERSIANA CUARTO 2	01011001	10010110	01100110
C	6	DIM	PERSIANA CUARTO 2	01011001	10010110	01100101
D	7	ON	LUZ COCINA	10011001	01100110	01011001
D	7	OFF	LUZ COCINA	10011001	01100110	01011010
D	7	BRIGHT	LUZ COCINA	10011001	01100110	01100110
D	7	DIM	LUZ COCINA	10011001	01100110	01100101
E	8	ON	LUZ BAÑO	01010110	10100110	01011001
E	8	OFF	LUZ BAÑO	01010110	10100110	01011010
E	8	BRIGHT	LUZ BAÑO	01010110	10100110	01100110
E	8	DIM	LUZ BAÑO	01010110	10100110	01100101
F	9	ON	LUZ ATICO	10010110	01101010	01011001
F	9	OFF	LUZ ATICO	10010110	01101010	01011010
F	9	BRIGHT	LUZ ATICO	10010110	01101010	01100110
F	9	DIM	LUZ ATICO	10010110	01101010	01100101
G	10	ON	LUZ ENTRADA	01100110	10101010	01011001
G	10	OFF	LUZ ENTRADA	01100110	10101010	01011010
G	11	ON	PUERTA ENTRADA	01100110	01011010	01011001
G	11	OFF	PUERTA ENTRADA	01100110	01011010	01011010
H	14	ON	LUZ/PUERTA GARAJE	10100110	10010101	01011001
H	14	OFF	LUZ/PUERTA GARAJE	10100110	10010101	01011010
I	13	ALL UNITS ON	LUCES DE LA CASA	01101010	01010101	01010110
I	13	ALL UNITS OFF	LUCES DE LA CASA	01101010	01010101	01010101

Retomando el ejemplo para encender la luminaria de la sala, una vez recibida la instrucción para encender la luminaria de la sala, el FPGA codifica dicha instrucción con los siguientes códigos X10:

01101001	01101001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

De esta manera el FPGA realiza las respectivas codificaciones para todas las tareas que el usuario desea automatizar en su vivienda.

6. Transmisión del código X10

Una vez que se ha realizado la codificación con el protocolo X10 de la instrucción que se desea ejecutar, procedemos a trasmitir el código X10. Para ello usamos el reloj configurado para la transmisión de datos a 9600 baudios.

El formato de transmisión de datos de forma asíncrona es muy parecido al formato que presenta el proceso de recepción, excepto porque no se realiza el muestreo ya que la transmisión desde el FPGA es confiable, por lo tanto se transmite un bit con cada ciclo del reloj de transmisión. En la figura 2.22, se puede observar que en primer lugar se envía el bit de inicio, el cual es un '0', seguido por los 8 bits de datos y finalmente en nuestro caso transmitimos 1 bit de parada, el cual es un '1', dejando al transmisor en reposo y a la espera de transmitir los próximos datos.

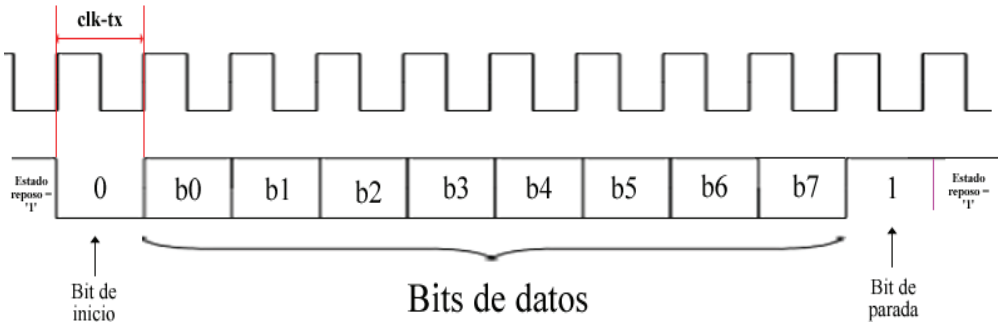


Figura 2.22: Formato de transmisión asíncrono

En base a este formato, el programa que se desarrolla en el FPGA para transmitir los códigos X10 comienza cambiando el estado de reposo del transmisor, para ello se envía el bit de inicio ‘0’, una vez enviado este bit, el transmisor a continuación envía los 8 bits de datos, uno a uno hasta completar el envío del byte. Finalmente se envía el bit de parada ‘1’ que vuelve al transmisor al estado de reposo inicial, hasta que se envíe un bit de inicio para realizar el proceso de transmisión nuevamente.

Cabe recalcar que en este caso se envían tres códigos, por lo tanto se realizan tres procesos de transmisión de manera consecutiva, es decir se envía la trama correspondiente a CASA, luego la trama correspondiente a UNIDAD, y al final se envía la trama correspondiente a FUNCIÓN.

De esta manera se envían los códigos X10 que serán reconocidos en el PIC, el cual se encarga de ejecutar las correspondientes tareas que desea el usuario.

Para completar el ejemplo de transmisión del código X10 para encender la luminaria de la sala, enviamos en primer lugar de manera serial el código de casa (01101001), a continuación del código de unidad (01101001) y finalmente el código de función (01011001), logrando de esta manera encender la luminaria de la sala de forma inmediata.

Una vez finalizada la transmisión, se culmina este proceso dejando al transmisor y receptor en reposo y dispuestos a cumplir sus funciones cuando el usuario así lo requiera.

2.3.1.2 Diseño del programa para generar el reloj para el usuario

Un reloj es un instrumento que le permite al usuario mantenerse informado sobre el tiempo actual de su localidad. Con este propósito se ha implementado un reloj en el FPGA, cuyas principales funciones son:

- Informar al usuario sobre su hora y minuto actual.
- Igualar al reloj con respecto a la hora y minutos actuales en caso de que presente un retraso o adelanto del tiempo actual.
- Programar la hora y minutos en los cuales desea que el sistema domótico realice alguna actividad sin necesidad que el usuario se encuentre en su vivienda.

2.3.1.2.1 Descripción del programa para generar el reloj para el usuario

Para cumplir los requerimientos del reloj, a continuación en la figura 2.23, se observa la lógica con la que trabaja el FPGA para poder programar e igualar la hora y minutos del reloj.

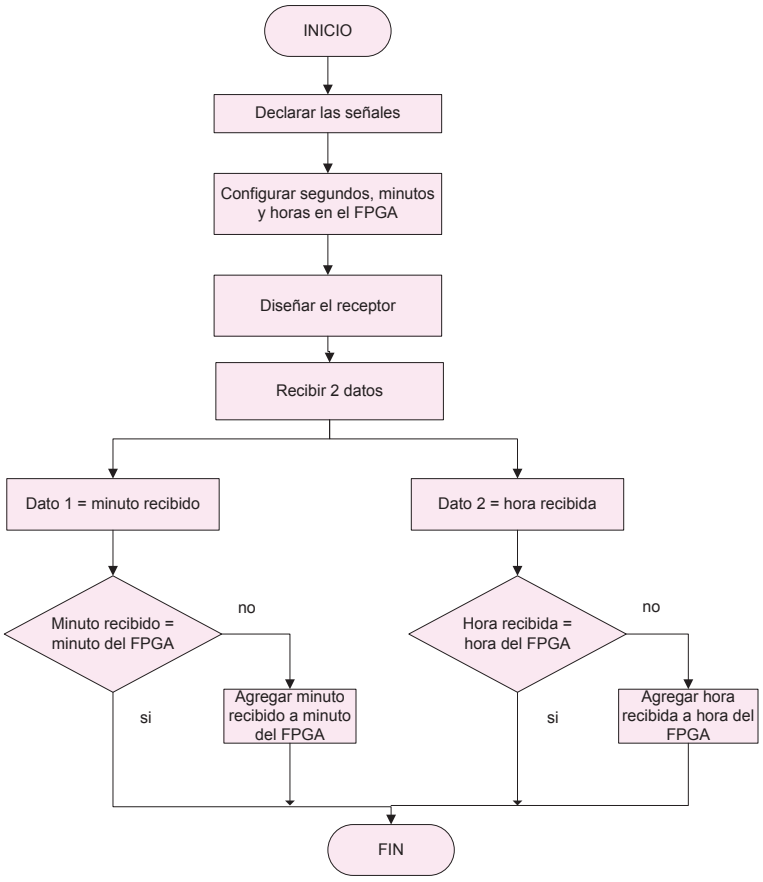


Figura 2.23: Diagrama de flujo del diseño del reloj

Con la finalidad de comprender el diagrama de flujo mostrado, a continuación se detallan cada uno de sus componentes.

1. Declaración de las señales

Consiste en la asignación de los nombres y tipos de dato que tendrán las diferentes señales que utilizamos a lo largo del programa para configurar, programar e igualar el reloj.

2. Configuración de los segundos, minutos y horas en el FPGA

El reloj que el FPGA va a mostrar al usuario, permite visualizar la hora y minuto actual de su localidad. Para conseguir este objetivo se debe configurar el reloj de 50MHz del FPGA, de tal manera que permita obtener segundos, que es la base de nuestro reloj. Los 50MHz nos indican que hay 50 millones de ciclos del reloj del FPGA en un segundo, por lo tanto para obtener un segundo debo tomar los 50 millones de ciclos del reloj interno del FPGA.

Se crea un contador que facilita la definición del número de ciclos del reloj del FPGA que se van a tomar para generar un segundo:

$$contador = 50MHz$$

$$contador = \frac{50000000\ ciclos}{segundo}$$

$$contador\ reloj = \frac{contador}{2}$$

$$contador\ reloj = \frac{50000000}{2}$$

$$\boldsymbol{contador\ reloj = 25000000}$$

Este valor se representa en la figura 2.24, en donde se puede observar que en un segundo están contenidos los 50000000 ciclos que se toman del reloj del FPGA. Cabe recalcar que estos 50000000 de ciclos se deben dividir para dos, puesto que se toman 25000000 de ciclos del reloj del FPGA para generar la parte positiva del ciclo de la señal *"clk_real"* y 25000000 para la parte negativa del mismo. Se debe tomar en cuenta que un ciclo completo de la nueva señal corresponde a un segundo.

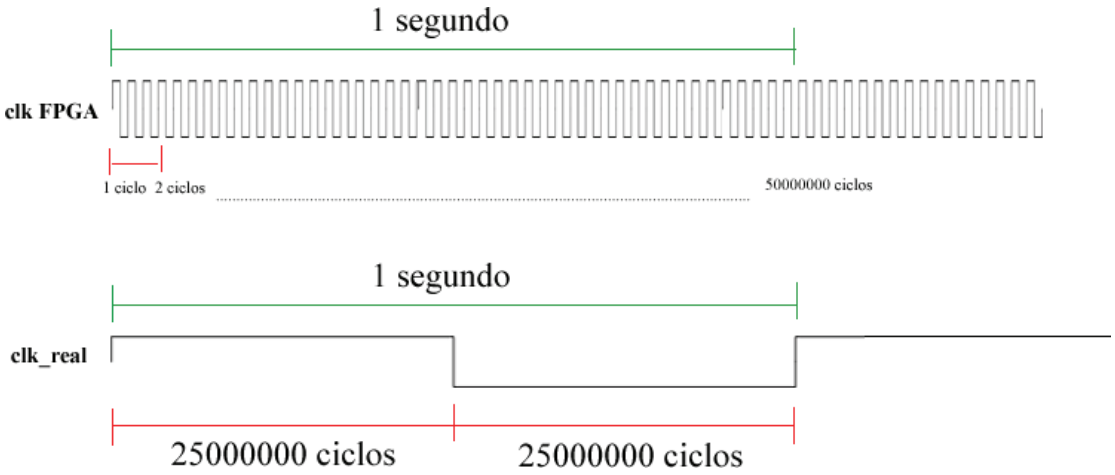


Figura 2.24: Diseño del reloj del usuario

Una vez creada la señal *"segundo",* la cual guarda un segundo del reloj del usuario, el FPGA procede a contar 60 veces esta señal de tal manera que se genera un minuto. A continuación procede a contar 60 veces la señal *"minuto"* y se genera una hora. Finalmente cuenta 24 veces la señal *"hora"*, una vez que contó 24 veces dicha señal, automáticamente se inicializan las señales y se repite todo el proceso desde el inicio y de forma indefinida tal y como lo realiza cualquier reloj.

De esta manera se genera los segundos, minutos y horas que el usuario podrá visualizar en tiempo real y en cualquier momento.

3. Diseño del receptor

De manera similar que en el proceso del control de la iluminación, persianas y puertas, en este proceso se utiliza exactamente el mismo receptor diseñado

anteriormente. Si se desea obtener más información acerca de este diseño, refiérase al numeral 2.3.1.1.1 para mayor detalle.

4. Recepción de dos datos

Una vez diseñado el receptor obtenemos como resultado el reloj que se utiliza en la recepción de datos. Este reloj nos permite recibir los datos de manera asíncrona en el formato ya descrito en el numeral 2.3.1.1.1.

Los datos que se reciben para este proceso corresponden a la hora y minuto actuales. Es decir que si el reloj del usuario se adelanta o retrasa con respecto al tiempo real, el usuario tiene la opción de igualar su reloj ingresando en la interfaz de usuario el minuto y hora deseados, de tal manera que sus procesos posteriores se puedan ejecutar correctamente.

Basándonos en esta explicación, el programa que se desarrolla en el FPGA para recibir el minuto y la hora actual, comienza detectando el bit de inicio '0', una vez detectado el bit de inicio se activa en '1' la señal *"act"* que indica que el receptor se encuentra ocupado recibiendo un dato. Mientras esta señal se encuentra en '1' el receptor va recibiendo bit a bit hasta que ha recibido los 10 bits correspondientes a un bit de inicio, 8 bits de datos y un bit de parada, una vez que ha culminado la recepción, *"act"* se pone en '0' indicando el fin de la recepción, y entonces se activa la señal *"listo"* en '1', indicando que el dato está listo para usarse internamente en el proceso requerido.

En este caso el FPGA recibe en primera instancia el dato denominado *"minuto"*, el cual corresponde al minuto actual del reloj. En segundo lugar recibe el dato denominado *"hora"*, el cual corresponde a la hora actual del reloj. Cuando se recibe uno de estos datos, se guardan en los respectivos registros para poder usarlos en el proceso de sincronización del reloj del usuario y a continuación se desactiva la señal

“listo” permitiéndole al receptor reposar y disponerse a recibir en cualquier momento el siguiente dato.

5.1 Dato 1 igual a minuto recibido

Este paso consiste en guardar el primer dato recibido por el FPGA en el registro *“minuto_recibido”,* más adelante este registro nos permitirá realizar una comparación vital para el proceso de sincronización del reloj.

5.2 Dato 2 igual a hora recibida

Este paso consiste en guardar el segundo dato recibido por el FPGA en el registro *“hora_recibida”,* este registro junto a *“minuto_recibido”,* nos permitirá en el siguiente paso realizar una comparación vital para el proceso de sincronización del reloj.

6.1 Minuto recibido igual a minuto del FPGA

En este paso el FPGA compara el dato correspondiente a *“minuto_recibido”* con la información que el FPGA guarda como “minuto”. Es decir se realiza una comparación entre el minuto que el FPGA ha generado internamente con el minuto que el FPGA ha recibido por parte del usuario.

Si el minuto generado por el FPGA es diferente al minuto que el usuario ha ingresado, se toma el valor del minuto que ingresó el usuario como valor real y se desecha el valor que el FPGA había generado. Caso contrario, si el valor del minuto ingresado por el usuario y el valor del minuto generado por el FPGA son los mismos, se toma el valor del minuto generado por el FPGA como valor real y se finaliza el programa.

6.2 Hora recibida igual a hora del FPGA

En este paso el FPGA compara el dato correspondiente a *“hora_recibida”* con la información que el FPGA guarda como “hora”. Es decir se realiza una comparación

entre la hora que el FPGA ha generado internamente con la hora que el FPGA ha recibido por parte del usuario.

Si la hora generada por el FPGA es diferente a la hora que el usuario ha ingresado, se toma el valor de la hora que ingresó el usuario como valor real y se desecha el valor que el FPGA había generado. Caso contrario, si el valor de la hora ingresada por el usuario y el valor de la hora generada por el FPGA son los mismos, se toma el valor de la hora generada por el FPGA como valor real y se finaliza el programa.

De esta manera el usuario puede sincronizar su reloj, lo cual le será de gran utilidad cuando desee programar funciones sin necesidad de que se encuentre en su vivienda.

2.3.1.3 Diseño del programa para el control programado de iluminación y persianas

Con la finalidad de proporcionar seguridad y ahorro de energía en la vivienda, el usuario mediante el uso de la interfaz gráfica puede realizar las siguientes funciones:

- En caso de que el usuario se disponga a llegar muy tarde a su casa, éste puede programar previamente el horario de encendido y apagado de las luces de cualquier lugar de su vivienda. De esta manera se busca fomentar cierto nivel de seguridad en su hogar, ya que este control le permite simular que alguien se encuentra en su casa encendiendo o apagando determinadas luces. Cabe recalcar que se fomenta el ahorro energético al controlar tanto el encendido como apagado de las luces de forma automática, de tal manera que solo se utiliza la energía necesaria por un tiempo determinado.

- Al igual que en el caso de las luces, si el usuario se dispone a llegar muy tarde a su casa, este puede programar previamente el horario para abrir ó cerrar las persianas de su vivienda. De esta manera se pretende evitar que cualquier persona descubra que nadie se encuentra en su casa.

2.3.1.3.1 Descripción del programa para el control programado de iluminación y persianas.

Para cumplir los requermientos para el control programado de la iluminación y persianas, a continuación en la figura 2.25 se observa la lógica con la que trabaja el FPGA para poder realizar este control dentro del sistema domótico.

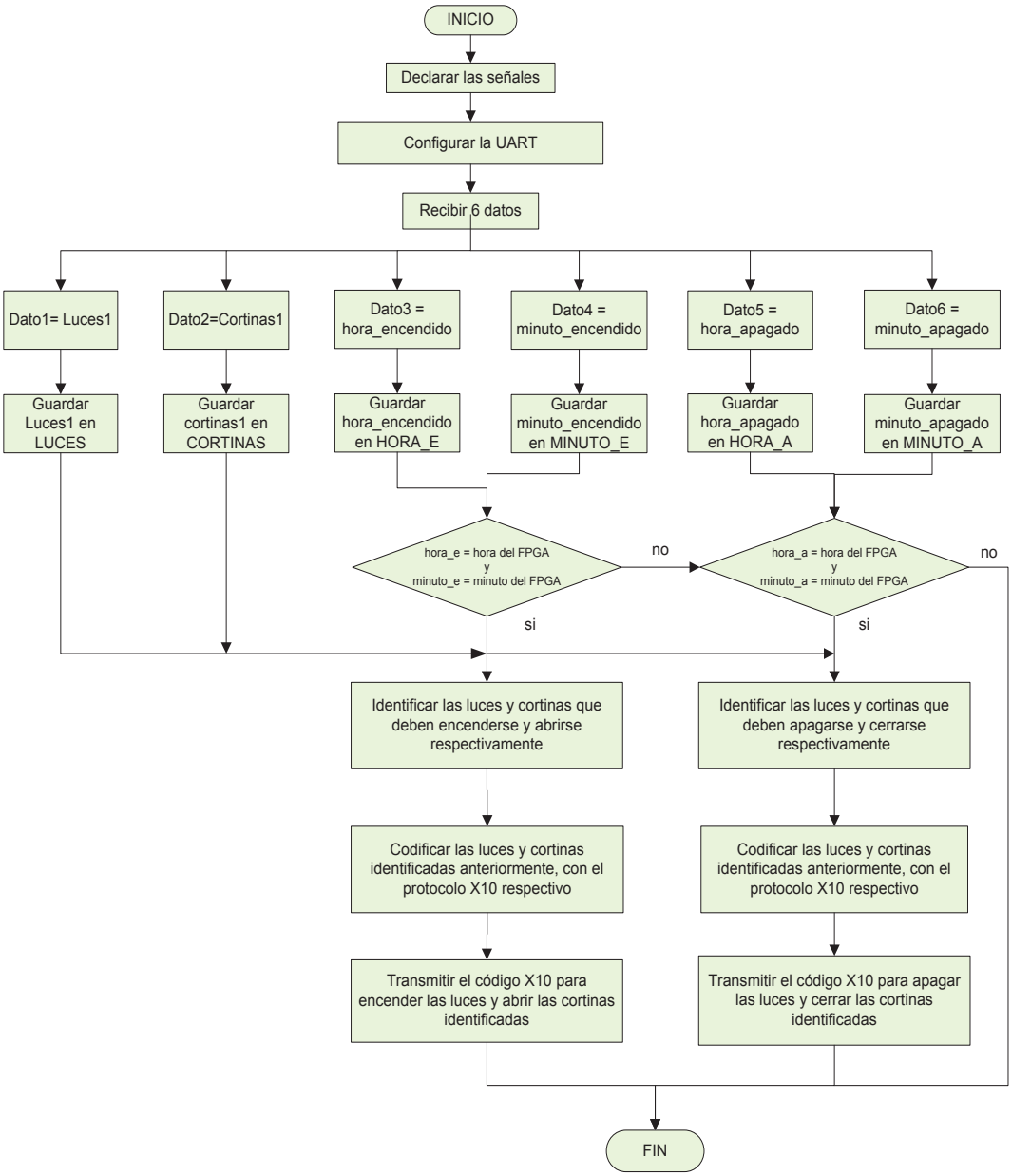


Figura 2.25: Diagrama de flujo del control programado de iluminación y persianas

Con la finalidad de comprender el diagrama de flujo mostrado, a continuación se detallan cada uno de sus componentes.

1. Declaración de las señales

Consiste en la asignación de los nombres y tipos de dato que tendrán las diferentes señales que utilizamos para poder programar las tareas que deben ser ejecutadas.

2. Configuración de la UART

Este proceso del diagrama de flujo consiste en diseñar el transmisor y el receptor que van a trabajar de forma asíncrona y con una velocidad de 9600 baudios. Para mayor información acerca de la UART, refiérase al literal 2.3.1.1.1 en donde se detalla su funcionamiento.

2.1 Diseño del Receptor

Al igual que en los procesos de control en tiempo real y configuración del reloj, en este proceso se utiliza exactamente el mismo receptor diseñado anteriormente. Si se desea obtener más información acerca de este diseño, refiérase al numeral 2.3.1.1.1 para mayor detalle.

2.2 Diseño del Transmisor

Refiérase al numeral 2.3.1.1.1 para mayor información acerca de las características y parámetros del diseño del transmisor.

3. Recepción de 6 datos

En este proceso recibimos 6 datos de forma serial, para ello el programa que se desarrolla en el FPGA empieza detectando el bit de inicio ‘0’, una vez detectado el bit de inicio se activa en ‘1’ la señal *“act”* que indica que el receptor se encuentra ocupado recibiendo un dato. Mientras esta señal se encuentra en ‘1’ el receptor va recibiendo bit a bit el dato hasta que ha recibido los 10 bits correspondientes a un bit de inicio, 8 bits de datos y un bit de parada. Una vez que ha culminado la recepción,

“act” se pone en ‘0’ indicando el fin de la recepción, y entonces se activa la señal *“listo”* en ‘1’, indicando que el dato está listo para usarse internamente.

El orden de los datos que se reciben serialmente es el siguiente:

1. **Luces:** La información que contiene esta trama permite determinar la luminaria de la vivienda que el usuario desea que se encienda o se apague.

2. **Cortinas:** La información que contiene esta trama permite determinar la persiana de la vivienda que el usuario desea que se abra o se cierre.

3. **Minuto de encendido:** Esta trama guarda el minuto en que el usuario desea que se encienda una luminaria ó se abra una persiana de su vivienda.

4. **Minuto de apagado:** Esta trama guarda el minuto en que el usuario desea que se apague una luminaria ó se cierre una persiana de su vivienda.

5. **Hora de encendido:** Esta trama guarda la hora en que el usuario desea que se encienda una luminaria ó se abra una persiana de su vivienda.

6. **Hora de apagado:** Esta trama guarda la hora en que el usuario desea que se apague una luminaria ó se cierre una persiana de su vivienda.

Cuando se recibe uno de estos datos, se guardan en sus respectivos registros para poder usarlos en el proceso de control programado de las funciones de la vivienda del usuario, y a continuación se desactiva la señal *“listo”* permitiéndole al receptor reposar y disponerse a recibir en cualquier momento el siguiente dato.

4.1 Dato 1 igual a LUCES1 → Guardar en LUCES

Como se mencionó anteriormente, el primer dato que recibe el FPGA en este proceso corresponde a la trama de luces, una vez que se recibe este dato se guarda en su respectivo registro "*LUCES"* con la finalidad de más adelante poder determinar qué luces fueron programadas para que se enciendan o apaguen automáticamente.

4.2 Dato2 igual a CORTINAS1 → Guardar en CORTINAS

El segundo dato que recibe el FPGA en este proceso corresponde a la trama de cortinas, una vez que se recibe este dato se guarda en su respectivo registro "*CORTINAS"* con la finalidad de más adelante poder determinar las persianas que fueron programadas para que se abran o se cierren automáticamente.

4.3 Dato3 igual a hora_encendido → Guardar en HORA_E

El tercer dato que recibe el FPGA en este proceso corresponde a la trama de hora de encendido, una vez que se recibe este dato se guarda en su respectivo registro "*HORA_E".* La información que contiene este registro le permite al FPGA ejecutar la instrucción programada, exactamente a la hora que el usuario desea.

4.4 Dato4 igual a minuto_encendido → Guardar en MINUTO_E

El cuarto dato que recibe el FPGA en este proceso corresponde a la trama de minuto de encendido, una vez que se recibe este dato se guarda en su respectivo registro "*MINUTO_E".* La información que contiene este registro le permite al FPGA ejecutar la instrucción programada, exactamente al minuto que el usuario desea.

4.5 Dato5 igual a hora_apagado → Guardar en HORA_A

El quinto dato que recibe el FPGA en este proceso corresponde a la trama de hora de apagado, una vez que se recibe este dato se guarda en su respectivo registro "*HORA_A".* La información que contiene este registro le permite al FPGA ejecutar la instrucción programada, exactamente a la hora que el usuario desea.

4.6 Dato4 igual a minuto_apagado → Guardar en MINUTO_A

Finalmente, el sexto dato que recibe el FPGA en este proceso corresponde a la trama de minuto de apagado, una vez que se recibe este dato se guarda en su respectivo registro "*MINUTO_A".* La información que contiene este registro le permite al FPGA ejecutar la instrucción programada, exactamente al minuto que el usuario desea.

5.1 HORA_E igual a hora del FPGA y MINUTO_E igual a minuto del FPGA

En este paso el FPGA compara los datos correspondientes a *"hora_e"* y *"minuto_e"* con la hora y minuto programados en el diseno del reloj en el FPGA.

Si la informacion de la hora y minuto de encendido coinciden con la hora y minuto actual que el FPGA genera con el reloj, entonces el FPGA envia la orden previamente solicitada por el usuario. Cuando se trata de hora de encendido y minuto de encendido, se pueden llevar a cabo dos funciones, encender una luminaria o abrir una persiana. Mas adelante se detalla como el FPGA determina la luminaria que debe encenderse o la persiana que debe abrirse en la vivienda.

Cuando la informacion de hora y minuto de encendido no son iguales a la hora y minuto actual, no se ejecuta ninguna funcion y mas bien el FPGA continua con sus actividades dentro del control domotico.

5.2 HORA_A = hora del FPGA y MINUTO_A = minuto del FPGA

En este paso el FPGA compara los datos correspondientes a *"hora_a"* y *"minuto_a"* con la hora y minuto programados en el diseno del reloj en el FPGA.

Si la informacion de la hora y minuto de apagado coinciden con la hora y minuto actual que el FPGA genera con el reloj, entonces el FPGA envia la orden previamente solicitada por el usuario. Cuando se trata de hora de apagado y minuto

de apagado, se pueden llevar a cabo dos funciones, apagar una luminaria o cerrar una persiana. Más adelante se detalla como el FPGA determina la luminaria que debe apagarse o la persiana que debe cerrarse en la vivienda.

Cuando la información de hora y minuto de apagado no son iguales a la hora y minuto actual, no se ejecuta ninguna función y mas bien el FPGA continúa con sus actividades dentro del control domótico.

6. Identificación de las luces que deben encenderse o las persianas que deben abrirse

Como se menciono anteriormente, el primer dato que recibe el FPGA corresponde a las luces y el segundo dato que recibe el FPGA corresponde a las persianas. A continuacion se detalla como se identifica los lugares en cada caso.

6.1 Luces

Para determinar las luces que el usuario desea que sean encendidas, la interfaz de usuario nos permite escoger la luminaria que deseamos encender. El prototipo de vivienda permite el control programado de siete lugares en la casa, cuyas respectivas luces son:

1. Luminaria de la sala
2. Luminaria de la habitación 1
3. Luminaria de la habitación 2
4. Luminaria de la cocina
5. Luminaria del baño
6. Luminaria del ático
7. Luminaria de la entrada

En la figura 2.26 se puede observar el orden como se muestran las luces en la interfaz de usuario, y en donde a través de un visto se puede seleccionar las luces deseadas.

Figura 2.26: Programación de las luces en la interfaz de usuario

La trama "*LUCES"* está formada por 8 bits, la posición de cada bit representa la luminaria de un lugar determinado. A continuación en la figura 2.27 se puede visualizar que el bit 0 de la trama corresponde a la luminaria de la sala, el bit 1 corresponde a la luminaria de la habitación 1, el bit 2 corresponde a la luminaria de la habitación 2, el bit 3 corresponde a la luminaria de la cocina, el bit 4 corresponde a la luminaria del baño, el bit 5 corresponde a la luminaria del ático y el bit 6 corresponde a la luminaria de la entrada.

Se debe tomar en cuenta que el bit 7 de la trama "*LUCES"* siempre será '0' puesto que no tenemos ningún lugar designado a ese bit.

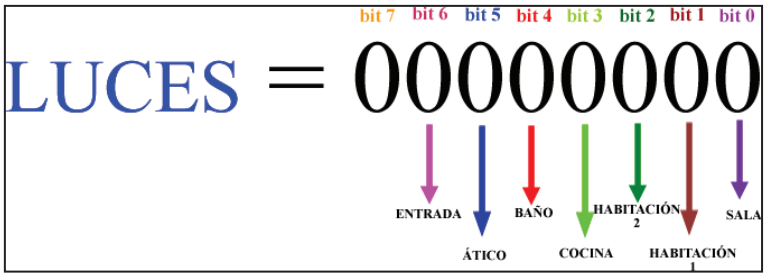


Figura 2.27: Trama *"LUCES"*

Cuando una o varias luces de la casa son seleccionadas por el usuario para ser encendidas, el bit correspondiente a esa luminaria cambia a '1', como se puede observar en la figura 2.28, en donde vamos a encender la lumianria de la sala y del ático, por lo tanto la trama *"LUCES"* es "00100001".

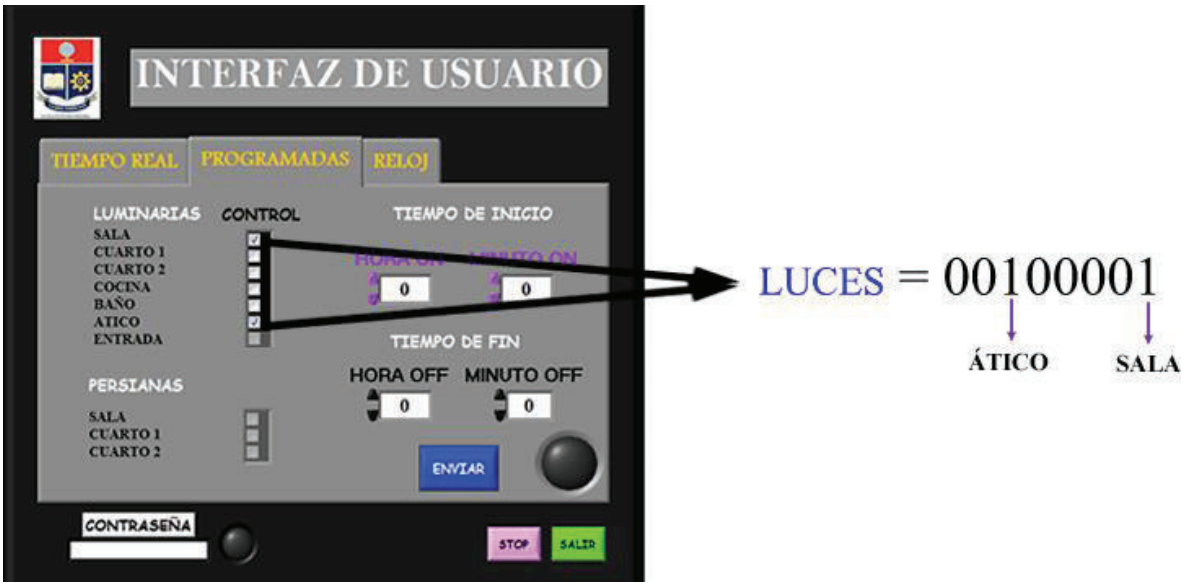


Figura 2.28: Interfaz de usuario y su correspondiente trama "*LUCES*"

De esta manera se controla el encendido programado de las luces dentro del prototipo de sistema domótico, logrando iluminar varios lugares de la vivienda al mismo tiempo y sin necesidad de que el usuario se encuentre presente controlando la iluminación.

6.2 Persianas

Para determinar las persianas que el usuario desea que sean abiertas, la interfaz de usuario nos permite escoger la persiana que deseamos abrir. En el prototipo se pueden controlar las persianas de 3 lugares:

1. Persiana de la sala
2. Persiana de la habitación 1
3. Persiana de la habitación 2

En la figura 2.29 se puede observar el orden como se muestran las persianas en la interfaz de usuario, y en donde a través de un visto se puede seleccionar las persianas deseadas.

Figura 2.29: Programación de las persianas en la interfaz de usuario

La trama "*CORTINAS*" está formada por 8 bits, la posición de cada bit representa la persiana de un lugar determinado. A continuación en la figura 2.30 se puede visualizar que el bit 0 de la trama corresponde a la persiana de la sala, el bit 1 corresponde a la persiana de la habitación 1 y finalmente el bit 2 corresponde a la persiana de la habitación 2.

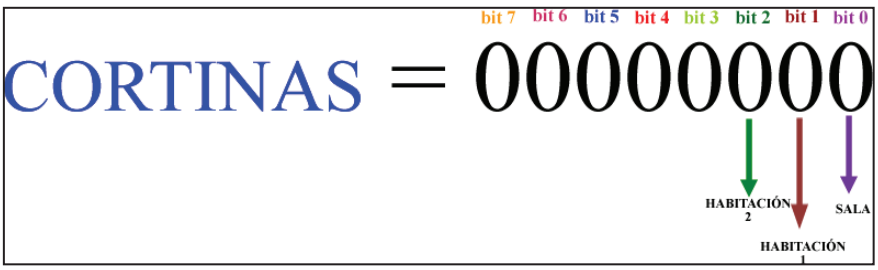


Figura 2.30: Trama "*CORTINAS"*

Se debe tomar en cuenta que los bits 3, 4, 5, 6, 7 de la trama "*CORTINAS"* siempre sera '0', puesto que no tenemos ningún lugar designado a esos bits.

Cuando una o varias persianas de la casa son seleccionadas por el usuario para ser abiertas, el bit correspondiente a esa persiana cambia a '1', como se puede observar en la figura 2.31, en donde vamos a abrir la persiana de la sala y de la habitación 2, por lo tanto la trama *"CORTINAS"* es "00000101".

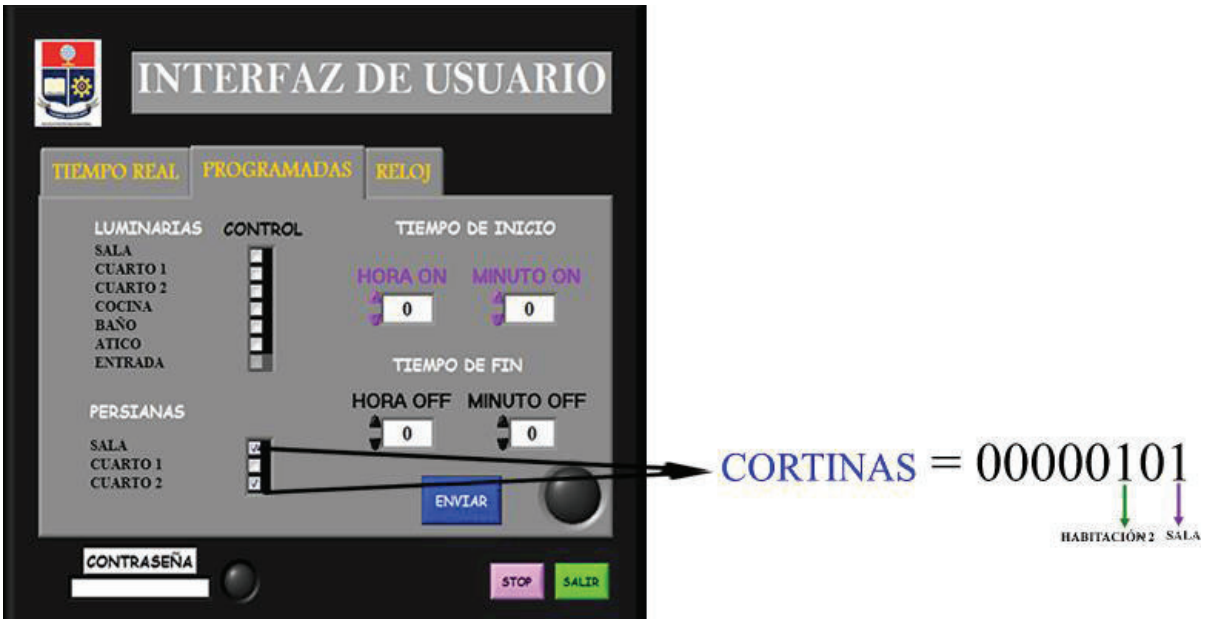


Figura 2.31: Interfaz de usuario y su correspondiente trama "*Cortinas"*

De esta manera se controla la apertura programada de las persianas en nuestro sistema domotico, logrando este objetivo en varios lugares de la vivienda al mismo tiempo y sin necesidad de que el usuario se encuentre presente controlando esta función.

7. Identificación de las luces que deben apagarse o las persianas que deben cerrarse

Una vez identificados los lugares cuyas luces y persianas se van a controlar, se procede a realizar el proceso inverso, es decir a apagar las luces y cerrar las persianas que el usuario seleccionó. A continuación se detalla el proceso que se realiza en cada caso.

7.1 Luces

En el proceso de encendido de luces se identificó el lugar que el usuario deseaba controlar, en este proceso usamos la misma trama *"LUCES"* para reconocer las luces que deseamos apagar, es decir que se van a apagar las luces que se encendieron anteriormente. Por esta razón el usuario debe ingresar el tiempo de encendido y el tiempo de apagado, para que automáticamente el FPGA reconozca la hora a la cual debe ejecutar cada función.

En el ejemplo de encendido, la trama *"LUCES"* es "00100001", es decir que el usuario programó su sistema domótico para que a una hora determinada se enciendan las luces de la sala y del ático. Nuestra interfaz le permite al usuario apagar esas mismas luces configurando una hora de apagado, es decir que a una hora determinada la luminaria de la sala y del ático serán apagadas y la trama *"LUCES"* nuevamente sera "00000000", es decir que todas las luces de la casa están apagadas.

Una vez terminado el proceso de control programado de luces, el FPGA ha culminado exitosamente las tareas que el usuario le designó para seguridad y ahorro energético de su vivienda.

7.2 Persianas

En el proceso para abrir las persianas, se identificó el lugar que el usuario deseaba controlar, en este proceso usamos la misma trama *"CORTINAS"* para reconocer las persianas que deseamos cerrar, es decir que se van a cerrar las persianas que se abrieron anteriormente. Por esta razón el usuario debe ingresar el

tiempo de apertura y el tiempo de cerrado, para que automáticamente el FPGA reconozca la hora a la cual debe ejecutar cada función.

En el ejemplo de apertura, la trama *"CORTINAS"* es "00000101", es decir que el usuario programó su sistema domótico para que a una hora determinada se abran las persianas de la sala y de la habitacion 2. Nuestra interfaz le permite al usuario cerrar esas mismas persianas configurando una hora de cerrado, es decir que a una hora determinada la persiana de la sala y de la habitación 2 serán cerradas automáticamente y la trama *"CORTINAS"* nuevamente sera "00000000", es decir que todas las persianas de la casa están cerradas.

Una vez terminado el proceso de control programado de persianas, el FPGA ha culminado exitosamente las tareas que el usuario le designó para seguridad de su vivienda.

8. Codificación de las luces y persianas identificadas con el protocolo X10 respectivo.

El control domótico en tiempo real consiste en recibir una instrucción enviada por el usuario, codificarla con el protocolo X10 y enviarla para ejecutar el control. En este proceso de control programado se repite exactamente la misma lógica, la única diferencia es que no envía los códigos inmediatamente, sino a la hora que el usuario ha programado que se ejecuten las funciones en su vivienda.

Como se mencionó anteriormente, la codificación de las instrucciones con el protocolo X10, consiste en asignar los tres códigos que corresponden al *Código de Casa, Código de Unidad y Código de Función* respectivamente.

Para conocer los códigos de casa, unidad y función del protocolo X10 se recomienda referirse a las tablas 1.1, 1.2 y 1.3. Adicionalmente para recordar la codificación X10 que usamos en este prototipo de sistema domótico, se recomienda referirse a la tabla 2.3, todas anteriormente expuestas

Retomando el ejemplo de encendido y apagado de luces programado, una vez recibida la trama *"LUCES = 00100001"* para encender y apagar la luminaria de la sala y del ático, el FPGA codifica dicha instrucción con los siguientes códigos X10:

- Encender la luminaria de la sala:

01101001	01101001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Encender la luminaria del ático:

01011001	01011001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Apagar la luminaria de la sala:

01101001	01101001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

- Apagar la luminaria del ático:

01011001	01011001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

De esta manera el FPGA realiza las respectivas codificaciones para todas las luces que el usuario desea encender y apagar en su vivienda.

Así mismo para el ejemplo de apertura y cerrado de persianas programado, una vez recibida la trama *"CORTINAS = 00000101"* para abrir y cerrar la persiana de la sala y de la habitación 2, el FPGA codifica dicha instrucción con los siguientes códigos X10:

- Abrir la persiana de la sala:

01101001	10101001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Abrir la persiana de la habitación 2:

10101001	10011001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Cerrar la persiana de la sala:

01101001	10101001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

- Cerrar la persiana de la habitación 2:

10101001	10011001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

Al igual que con las luces, el FPGA realiza las respectivas codificaciones para todas las persianas que el usuario desea abrir y cerrar en su vivienda.

Todos estos códigos X10 para encender y apagar las luces, y abrir y cerrar las persianas, son enviados serialmente a la hora y minuto correspondientes a cada función que el usuario programó, logrando controlar eficientemente su vivienda en base al protocolo X10.

9. Transmisión del código X10 para el control programado de funciones

Una vez que se ha realizado la codificación con el protocolo X10 de las funciones programadas que se desean ejecutar, se procede a trasmitir el código X10 respectivo. Para ello usamos el reloj configurado para la transmisión de datos a 9600 baudios.

Basándonos en la explicación sobre transmisión serial asíncrona descrita en el numeral 2.3.1.1.1, el programa que se desarrolla en el FPGA para transmitir los códigos X10 consiste en que, una vez identificados los lugares que se van a controlar, se los codifica y finalmente a la hora y minuto exactos elegidos por el usuario, se envían los códigos X10 respectivos a las funciones que se deben ejecutar. Esta transmisión se la realiza siguiendo los parámetros de transmisión serial asíncrona, es decir enviamos el bit de inicio '0', una vez enviado este bit, el transmisor a continuación envía los 8 bits de datos, uno a uno hasta completar el envío del byte. Finalmente se envía el bit de parada '1' que vuelve al transmisor al estado de reposo inicial, hasta que se envíe un bit de inicio para realizar el proceso de transmisión nuevamente.

Cabe recalcar que por cada función que el usuario desea que se ejecute, se deben enviar tres códigos, por lo tanto realizamos tres procesos de transmisión de manera consecutiva, es decir enviamos la trama correspondiente a CASA, luego enviamos la trama correspondiente a UNIDAD, y al final enviamos la trama correspondiente a FUNCIÓN.

En el caso de funciones programadas por el usuario, el número de tramas que transmite el FPGA es proporcional al número de funciones que el usuario desea que sean ejecutadas. Es decir que si el usuario desea que se enciendan las luces de la sala y del ático, el FPGA transmitirá un total de seis tramas, es decir tres tramas correspondientes al código X10 para encender la luminaria de la sala y tres tramas correspondientes al código X10 para encender la luminaria del ático.

Todos estos códigos X10 son enviados a la hora y minuto exactos, de tal manera que serán reconocidos en el PIC del circuito transmisor X10, el cual se encarga de ejecutar las correspondientes tareas que el usuario programó.

Para finalizar el proceso de control programado, se realiza la transmisión serial del código X10 de los ejemplos anteriormente expuestos, es decir a la hora de encendido se transmiten un total de 12 tramas, que son las siguientes:

- Encender la luminaria de la sala:

01101001	01101001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Encender la luminaria del ático:

01011001	01011001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Abrir la persiana de la sala:

01101001	10101001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

- Abrir la persiana de la habitación 2:

10101001	10011001	01011001
CASA	***UNIDAD***	***FUNCIÓN***

A la hora de apagado se transmite igualmente un total de 12 tramas, cuya finalidad es apagar todo lo que fue encendido. Dichas tramas son las siguientes:

- Apagar la luminaria de la sala:

01101001	01101001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

- Apagar la luminaria del ático:

01011001	01011001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

- Cerrar la persiana de la sala:

01101001	10101001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

- Cerrar la persiana de la habitación 2:

10101001	10011001	01011010
CASA	***UNIDAD***	***FUNCIÓN***

Una vez finalizada la transmisión, se culmina este proceso dejando al transmisor y receptor en reposo y dispuestos a cumplir sus funciones cuando el usuario así lo requiera.

Se debe tomar en cuenta que el FPGA puede desempeñar todos los procesos de manera paralela, es decir mientras nos muestra el reloj, nos permite programar alguna funcion para que se ejecute en la noche, y mientras llega la noche podemos enviar diferentes instrucciones para que se ejecuten de forma instantánea. Es decir

nos permite llevar a cabo todas las actividades programadas para el control total de nuestro prototipo de manera eficiente.

Finalmente, es importante recalcar que toda la programación desarrollada en el FPGA (VHDL), se encuentra detallada en el Anexo 17, por lo cual se recomienda revisar este anexo con la finalidad de analizar y comprender el código fuente del controlador del prototipo de sistema domótico, que en este caso es la tarjeta Spartan 3E Starter Board.

2.3.2 DISEÑO DE LOS PROGRAMAS DE TRANSMISIÓN Y RECEPCIÓN DE LOS CÓDIGOS X10 [9]

Para la implementación del protocolo X10 se han utilizado microcontroladores PIC, específicamente, los microcontroladores PIC16F877A (Ver Anexo 7), siendo el propuesto por la empresa Microchip en su publicación AN236, “X-10 Home Automation Using the PIC16F877A”.

El programa para el microcontrolador PIC, se desarrolló en lenguaje assembler, teniendo a la mano sus 30 instrucciones junto con las directivas que posee el microcontrolador. Además, para los programas se utilizó una librería llamada X10lib.inc creada por Microchip con las definiciones y constantes de los programas X10.

Los programas de transmisión y recepción se diferencian principalmente en las subrutinas de interrupción y en la configuración del microcontrolador. En la descripción del desarrollo del diseño de este programa, se diferenciarán los diagramas de flujo tanto para la transmisión como para la recepción de los códigos X10.

A continuación en la figura 2.32 se muestra la estructura del programa principal, el cual está formado por tres bloques principales.

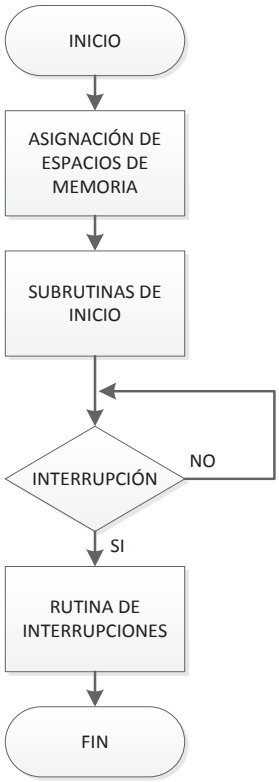


Figura 2.32: Estructura del programa principal de transmisión y recepción X10

2.3.2.1 Asignación de espacios de memoria

En el desarrollo de los programas de transmisión y recepción se utilizan diferentes variables, las cuales se almacenan en la memoria RAM del microcontrolador. Para un mejor desarrollo del programa se utilizan etiquetas, las cuales, se refieren a una ubicación de memoria RAM. En este bloque se asignan las direcciones de la memoria RAM a las etiquetas a utilizar, además, se añade la palabra de configuración (14 Bits) que sirve para ingresar en el microcontrolador PIC la configuración con la que va a trabajar.

A continuación, en la figura 2.33 se muestra el proceso de asignación de espacios de memoria en el programa.

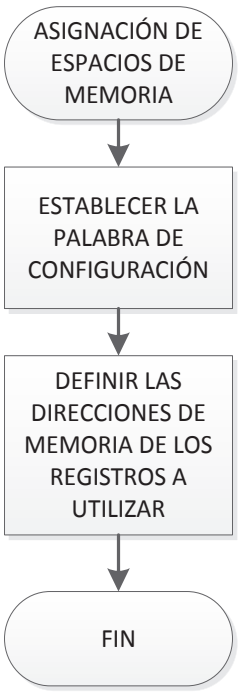


Figura 2.33: Asignación de espacios de memoria

En este bloque además de asignar etiquetas a las variables de la memoria RAM, se establece la palabra de configuración del PIC, la cual, varía entre transmisor y receptor puesto que el receptor utiliza un cristal oscilador de menor frecuencia.

2.3.2.1.1 Palabra de Configuración

Es un conjunto de registros compuesto por 14 bits, los cuales, especifican entre otras configuraciones la fuente de la señal de reloj y las protecciones de la memoria flash y EEPROM. Los bits se detallan a continuación:

- **Protección por Código (Bit 13):** Permite proteger el programa dentro del PIC impidiendo su lectura. En los programas de transmisión y recepción no se protegerá el programa.

- **Bit 12:** No usado, se rellena con 0.

- **Debug (Bit 11):** Permite al programador usar los pines RB6 y RB7 para el programador/depurador In Circuit de Microchip. En los programas de transmisión y recepción se usarán los pines RB6 y RB7 como I/O de propósito general.

- **Protección de Escritura de la Memoria Flash (Bit 10 – Bit 9):** Permite proteger la memoria Flash contra escritura. No se utiliza protección de escritura de la memoria Flash.

- **Bit de Protección de la Memoria EEPROM (Bit 8):** Permite proteger el código de la memoria EEPROM. No se utiliza protección de la Memoria EEPROM.

- **Bit de Programación Serial In-Circuit de Bajo Voltaje (Bit 7):** Al habilitar este pin se configura al microcontrolador PIC para que sea programado con un voltaje dentro del rango de funcionamiento del mismo. De esta manera, ya no es necesario enviar un alto voltaje al pin MCRL; en cambio, se aplica un voltaje bajo al pin RB3, dejando al mismo inhabilitado como pin de entrada/salida. Se habilita la Programación de Bajo Voltaje.

- **Bit de Reset por Brown-Out (Bit 6)**: El reseteo por Brown-Out se da cuando el voltaje de alimentación disminuye debajo de 4V por al menos 100 uS manteniéndose reseteado por un tiempo de 72 mS. Se habilita el Reset por Brown-Out.

- **Bit 5:4:** No usado, se rellena con 0.

- **Habilitación del Timer Power-Up (Bit 3):** El Timer Power-Up le da un tiempo de 72 mS al PIC después de ser alimentado para esperar que el voltaje de alimentación llegue a un nivel aceptable. No se habilita el Timer Power-Up.

- **Timer Watch Dog (Bit 2):** El Timer Watch Dog resetea al PIC cuando este está en modo SLEEP. No se habilita el Timer Watch Dog.

- **Bits de Selección de Oscilador (Bits 1:0):** Estos bits permiten seleccionar el tipo de oscilador a usar, entre cuatro opciones:
  - Oscilador RC
  - Oscilador HS (High Speed)
  - Oscilador XT (Cristal Cuarzo)
  - Oscilador LP (Low Power)

Por lo tanto, como en el circuito de transmisión se usara un cristal de 7.3728 MHz, la configuración a utilizar es como Oscilador HS.

Por otro lado, en el circuito de recepción se utilizara un cristal de 3.579545 MHz puesto que una frecuencia de reloj alta impide que el PIC recepte de manera correcta cada bit en la línea eléctrica, por lo que la configuración sería la de Oscilador XT.

Con el conjunto de bits explicados anteriormente se obtienen las palabras de configuración tanto para el transmisor (0x2FC2), como para el receptor (0x2FC1), las cual se muestran a continuación en las figuras 2.34 y 2.35 respectivamente.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	0	0	0	1	0
2		F				C				2			

Figura 2.34: Palabra de configuración del transmisor X10

13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	0	0	0	0	1
2		F				C				1			

Figura 2.35: Palabra de configuración del receptor X10

2.3.2.1.2 Asignación de las direcciones de memoria de los registros a utilizar

Este paso consiste en asignar direcciones de la memoria RAM a las etiquetas de las variables a utilizar. Estas etiquetas se describen a continuación:

- **X10Flags:** Este registro es el banco de banderas utilizados como indicadores de transmisión y recepción. En este registro se tienen 8 banderas:
    - **RxBit:** Bandera que guarda el último bit recibido.
    - **TxBit:** Bandera que guarda el último bit a transmitir.
    - **RxCommandFlag:** Bandera que indica si el mensaje recibido es un comando.
    - **TxCommandFlag:** Bandera que indica si el mensaje a transmitir es un comando.
    - **X10RxFlag:** Bandera que indica la recepción de un mensaje X10.
    - **X10TxFlag:** Bandera que indica que se tiene un mensaje X10 a transmitir. (Se mantiene levantada mientras se realiza la transmisión de todos los bits).
    - **SecondTxFlag:** Bandera que indica la segunda transmisión del mensaje X10.
    - **SecondTxBloq:** Bandera que indica la segunda transmisión del bloque X10.

La posición de cada una de estas banderas en el registro X10Flags, se muestra a continuación en la figura 2.36.

X10Flags							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Rx Bit	Tx Bit	RxCommand Flag	TxCommand Flag	X10Rx Flag	X10Tx Flag	SecondTx Flag	SecondTx Bloq

Figura 2.36: Banderas del registro X10Flags

- **WREG:** Registro temporal que guarda el registro W durante una interrupción.

- **_STATUS:** Registro temporal que guarda el registro STATUS durante una interrupción.

- **_PCLATCH:** Registro temporal que guarda el registro PCLATCH durante una interrupción.
- **_FSR:** Registro temporal que guarda el registro FSR durante una interrupción.

- **Registros del Transmisor:**
    - **NumTxBits:** Este registro indica el número de bits transmitidos.
    - **TxFunction:** Este registro guarda el código de función.
    - **TxHouse:** Este registro guarda el código de casa.
    - **TxUnit:** Este registro guarda el código de unidad.
    - **TxData0, TxData1, TxData2, TxData3:** Registros temporales de los bits a transmitir.
    - **Conteo:** Registro que guarda el número de veces que se van a rotar los registros temporales TxData.

- **Contador:** Registro que guarda el número de bytes recibidos vía comunicación serial (0-3).

- **Dato:** Registro que guarda temporalmente el byte recibido vía comunicación serial.

- **Registros del Receptor:**
  - **RxFunction:** Este registro guarda el código de la función recibida.

  - **RxHouse:** Este registro guarda el código de casa recibida.

  - **RxUnit:** Este registro guarda el código de unidad recibida.
  - **RxHouse1, RxHouse2:** Estos registros se encargan de guardar las direcciones de Casa recibidas en cada transmisión X10.

  - **RxKey1, RxKey2:** Estos registros se encargan de guardar las direcciones de Unidad o Función recibidas en cada transmisión X10.

  - **RxData0, RxData1, RxData2, RxData3:** Registros temporales que guardan los bits recibidos.

  - **Bandera:** Registro de ayuda para la verificación de coincidencia entre las dirección de Casa recibido y la dirección del Receptor X10.

  - **RxFunction:** Registro que almacena la función recibida.

2.3.2.2 Subrutinas de inicio

Las subrutinas de inicio se encargan de preparar al microcontrolador PIC con las configuraciones iniciales de Timers, Interrupciones y registros para la correcta transmisión y recepción del protocolo X10.

A continuación, en la figura 2.37 se muestra la lógica de funcionamiento de las subrutinas de inicio, tanto para la transmisión como para la recepción de los códigos X10.

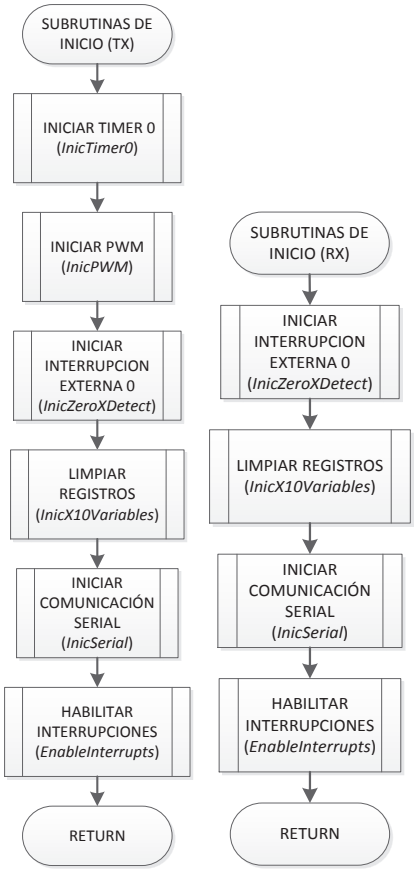


Figura 2.37: Subrutinas de inicio

2.3.2.2.1 Subrutinas de inicio comunes en la transmisión y recepción de códigos X10

Estas subrutinas, como su nombre lo indica, se realizan tanto en el proceso de transmisión como de recepción. Por esta razón las describimos a continuación:

- **Subrutina para Iniciar la Interrupción Externa (InicZeroXDetect):** Es una subrutina utilizada para habilitar la interrupción externa del puerto RB0, de tal manera que se pueda efectuar la detección de cruces por cero, y su lógica de trabajo se muestra a continuación en la figura 2.38.

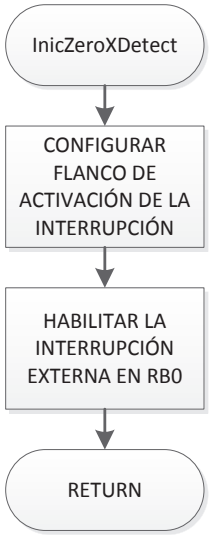


Figura 2.38: Subrutinas InicZeroXDetect

Para iniciar la detección de cruces por cero, habilitamos la interrupción externa en RB0 por el flanco de subida, para lo cual efectuamos los siguientes procedimientos:

- *Habilitar el bit de selección de flanco (Bit 6) del registro OPTION_REG:* Se setea este bit en 1L (figura 2.39) para que el microcontrolador PIC detecte el flanco de subida para la interrupción.

OPTION_REG							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	1	X	X	X	X	X	X

Figura 2.39: Bit de selección de flanco del OPTION_REG

- *Habilitar el bit de habilitación de la interrupción externa RB0/INT (Bit 4) del registro INTCON:* Se setea este bit en 1L (figura 2.40) para habilitar la interrupción externa.

INTCON							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	X	1	X	X	X	X

Figura 2.40: Bit de habilitación de la interrupción externa del INTCON

- **Subrutina para inicializar los registros (InicX10Variables):** Subrutina utilizada para inicializar variables, es decir, borrar el contenido de los registros temporales que se van a utilizar en los procesos X10, y su lógica de trabajo se muestra a continuación en la figura 2.41.

Figura 2.41: Subrutinas InicX10Variables

En el transmisor se inicializan los siguientes registros:

- TxData0
- TxData1
- TxData2
- TxData3
- TxHouse
- TxUnit

- TxFunction
- X10Flags
- Conteo
- Contador
- Dato

En el receptor se inicializan los siguientes registros:

- X10Flags
- RxFunction
- RxHouse
- RxUnit
- RxHouse1
- RxHouse2
- RxKey1
- RxKey2
- RxData0
- RxData1
- RxData2
- RxData3
- Bandera
- RxFunction

- **Subrutina para iniciar la comunicación serial (InicSerial):** En esta subrutina se configura el puerto USART del PIC para que reciba los bytes enviados por el FPGA, y la forma de configurarlo se muestra a continuación en la figura 2.42.

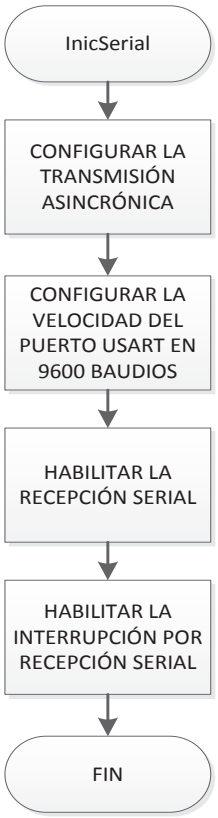


Figura 2.42: Subrutina InicSerial

Primero, procedemos a configurar la velocidad de transmisión en 9600 baudios, para lo cual, modificamos los registros TXSTA y SPBRG. En el registro TXSTA se encuentra el bit de configuración de la velocidad de transmisión (BRGH) en el cual, se determina si la velocidad de transmisión es alta o baja. Para lo cual, se selecciona una velocidad de transmisión alta.

- *Bit de Selección de Modo de Transmisión (Bit 4):* El microcontrolador PIC posee los modos de transmisión sincrónico y asincrónico. En este caso se utiliza el modo de transmisión asincrónico.

- *Bit de Selección de Velocidad de Transmisión Alta (Bit 2):* Habilitamos la velocidad de transmisión alta, lo que modificara el cálculo del registro SPBRG.

Ahora, para configurar la velocidad de recepción se debe calcular el valor de Sampling Baud Rate Generator (SBRG), para lo cual, se utiliza la siguiente fórmula:

$$Baud\ Rate = \frac{F_{osc}}{16(X+1)}$$

Dónde:

X = Valor del registro SPBRG (entre 0 y 255)

Para una velocidad de transmisión de 9600 Baudios, se tiene:

$$9600 = \frac{7{,}3728\ MHz}{16(X+1)}$$

$$X = \frac{7{,}3728\ MHz}{9600 * 16} - 1 = 47$$

$$\therefore \boldsymbol{X = 47}$$

En el transmisor se usa el puerto serial, para recibir los comandos a enviar. Por lo tanto, para configurar la recepción serial, se modifican los siguientes bits del registro RCSTA (figura 2.43):

- *Bit de Habilitación del Puerto Serial (Bit 7):* Con este bit se habilita el puerto serial para TX y RX. Para lo cual, habilitamos seteando en 1L.

- *Bit de Habilitación de la RX de 9 bits (Bit 6):* Con este bit de habilita la recepción de 9 bits por el puerto serial. Para lo cual, deshabilitamos ubicando un 0L.

- *Bit de Habilitación de Recepción Continua (Bit 4):* Con este bit de habilita la recepción continua por el puerto serial. Para lo cual, habilitamos seteando en 1L.

- *Bit de Paridad (Bit 0):* Sin bit de paridad.

RCSTA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	0	X	1	0	0	0	0

Figura 2.43: Configuración del registro RCSTA para recepción serial

A continuación se habilita la interrupción por recepción, para lo cual, es necesario ubicarse en el registro PIE1 (Peripherical Interrupts Enable) y habilitar la interrupción seteando este bit en 1L (figura 2.44):

- *Bit de Habilitación de Interrupción por Recepción (Bit 5):* Habilita la interrupción por recepción del puerto USART.

PIE1							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	1	X	X	X	X	X

Figura 2.44: Configuración del registro PIE1 para habilitar la interrupción por recepción

En el lado del receptor, también se utiliza la comunicación serial pero para transmitir el comando recibido hacia el actuador, por lo cual, en este caso se configurará la comunicación serial para transmisión modificando el registro TXSTA (figura 2.45), de la siguiente manera:

- *Bit de Habilitación de la transmisión de 9 bits (Bit 6):* Con este bit se habilita la transmisión de 9 bits por el puerto serial. Por lo tanto, puesto que la transmisión es de 8 bits, se setea este bit en 0L.

- *Bit de Habilitación de Transmisión Serial (Bit 5):* Con este bit de habilita la transmisión por el puerto serial. Para lo cual, habilitamos seteando el bit en 1L.

TXSTA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	0	1	0	0	0	X	X

Figura 2.45: Configuración del registro TXSTA para transmisión serial

Ahora para configurar la velocidad de recepción se debe calcular el valor de Sampling Baud Rate Generator (SBRG), para lo cual, se utiliza la siguiente fórmula:

$$Baud\ Rate = \frac{F_{osc}}{64(X+1)}$$

Dónde:

X = Valor del registro SPBRG (entre 0 y 255)

Para una velocidad de transmisión de 1200 Baudios, se tiene:

$$1200 = \frac{3{,}579545\ MHz}{64(X+1)}$$

$$X = \frac{3{,}579545\ MHz}{1200 * 64} - 1 = 45{,}6$$

$$\therefore \boldsymbol{X = 46}$$

Finalmente se habilita la comunicación serial, seteando en 1L en el bit 7 del registro RCSTA.

- **Subrutina para Habilitar las Interrupciones (EnableInterrupts):** Esta subrutina es utilizada para habilitar las interrupciones periféricas y globales. Su lógica de trabajo se muestra a continuación en la figura 2.46.

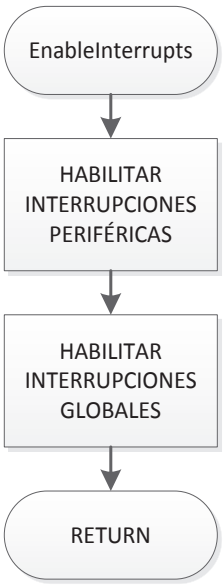


Figura 2.46: Subrutina EnableInterrupts

Esta subrutina se encarga de modificar los registros PIR1, PIR2 e INTCON. Primero se setea en 0L todas las banderas de interrupciones limpiando los registros PIR1 y PIR2. A continuación, en el registro INTCON se habilitan las interrupciones periféricas y globales (figura 2.47) de la siguiente manera:

- *Bit de Habilitación de Interrupciones Globales (Bit 7):* Habilita todas las interrupciones.

- *Bit de Habilitación de Interrupciones Periféricas (Bit 6):* Habilita las interrupciones periféricas.

INTCON							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	1	X	X	X	X	X	X

Figura 2.47: Configuración del registro INTCON para habilitar las interrupciones globales y periféricas

2.3.2.2.2 Subrutinas de inicio para el Transmisor

Estas subrutinas se ejecutan solamente en el proceso de transmisión y son las siguientes:

- **Subrutina para Iniciar el Timer 0 (InicTimer0):** Subrutina utilizada para inicializar el Timer 0 (Contador de 1ms). Su lógica de programación se muestra a continuación en la figura 2.48.

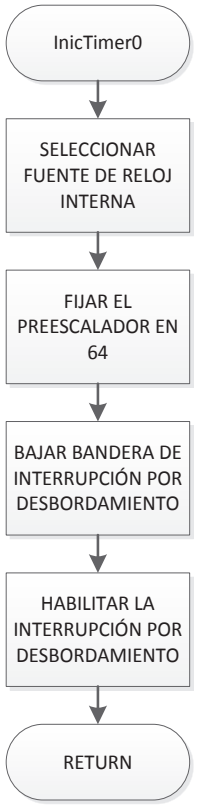


Figura 2.48: Subrutina InicTimer0

El Timer 0 es un timer/contador de 8 bits de resolución, y se utiliza como contador de 1 ms, que es la duración de la señal portadora. Existen 3 registros asociados al Timer 0:

- *TMR0:* Registro del módulo del Timer 0.
- *INTCON:* Registro de interrupciones.
- *OPTION_REG:* Registro de opciones del Timer 0.

Para la inicialización de este Timer se hace uso de dos de los registros asociados al Timer 0, OPTION_REG e INTCON, mientras que el tercer registro será utilizado posteriormente.

En el registro OPTION_REG (figura 2.49) se utilizan los siguientes bits:

- *Bit de Selección de fuente de reloj (Bit 5):* Este bit permite seleccionar entre la fuente de reloj interna del PIC o el pin TOCKI (pin RA4). Se utiliza como fuente, el reloj interno del microcontrolador PIC.

- *Bit de Asignación del pre-escalador (Bit 3):* Dado que el pre-escalador del Timer 0 puede ser usado para sí mismo o para el Watch Dog Timer, este bit es utilizado para determinar si es el Timer 0 o el WDT que va a utilizar el pre-escalador. El Timer 0 será quien utiliza el pre-escalador.

- *Bits del pre-escalador (Bits 2:0) – 1 0 1:* Estos tres bits permiten escoger el valor del pre-escalador desde un valor de 1 hasta 256. El uso del pre-escalador es de utilidad para cálculos posteriores. Se utiliza un pre-escalador de 64.

OPTION_REG							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1	X	0	X	0	1	0	1

Figura 2.49: Configuración del registro OPTION_REG para inicializar el timer 0

En el registro INTCON se utilizan los bits correspondientes a las banderas de interrupción y habilitación del Timer 0, tal y como se presenta en la figura 2.50.

- *Bit de habilitación de las interrupciones del Timer 0 (Bit 5):* Bit encargado de habilitar la interrupción por desbordamiento del Timer 0. Para habilitarlo debemos setear en 1L este bit.

- *Bandera de Interrupción del Timer 0 (Bit 2):* Bandera de la interrupción por desbordamiento del Timer 0, la cual, se dispara cada vez que el contador del Timer 0 llega a 0x00 o 0xFF. En este caso se setea en 0L esta bandera para que esté lista al utilizar el Timer.

INTCON							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
X	X	1	X	X	0	X	X

Figura 2.50: Configuración del registro INTCON para habilitar el timer 0

- **Subrutina para iniciar el PWM (InicPWM):** Esta subrutina es utilizada para inicializar el módulo PWM del Timer 2, es decir se va a generar la señal portadora de 120 KHz. El funcionamiento de esta subrutina se muestra a continuación en la figura 2.51.

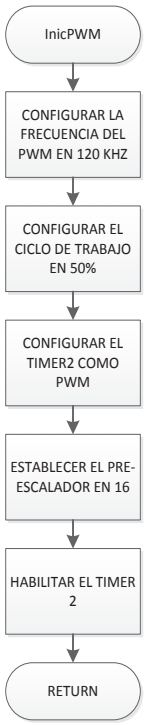


Figura 2.51: Subrutina InicPWM

En esta subrutina se utiliza el Timer 2 puesto que es el único que puede ser utilizado como PWM (Pulse Wide Modulation).

Los registros utilizados en el Timer 2 son: PR2, T2CON, CCPR1L, PIE1, CCP1CON y T2CON.

Para iniciar el funcionamiento PWM se deshabilita la salida (Pin RB2) antes de iniciar la configuración del mismo.

Inicialmente, configuramos el registro PR2, este registro es utilizado para determinar el valor del periodo PWM, el cual se obtiene con la siguiente fórmula:

$$Periodo\ PWM = [(PR2) + 1] * 4 * T_{OSC} * (Preescalado\ del\ TMR2)$$

Dado que el periodo PWM sería el inverso de la señal 120KHz:

$$Periodo\ PWM = \frac{1}{F_{PWM}} = \frac{1}{120KHz}$$

$$\boldsymbol{Periodo\ PWM = 8,3\widehat{3}\ uS}$$

Por lo cual tomando un cristal de oscilación de 7,3728 MHz, el valor de PR2 sería:

$$PR2 = \frac{T_{PWM}}{4 * T_{OSC} * (Preescalado\ del\ TMR2)} - 1$$

$$PR2 = \frac{\frac{1}{F_{PWM}}}{4 * \frac{1}{F_{OSC}} * 64} - 1$$

$$PR2 = \frac{\frac{1}{120\ KHz}}{4 * \frac{1}{7{,}3728\ MHz} * 16} - 1 = -0{,}04$$

$$\therefore \boldsymbol{PR2 = 0}$$

Este valor de 0 debemos escribirlo en el registro PR2 para configurar la frecuencia del PWM en 120KHz.

A continuación se debe configurar el ciclo de trabajo del PWM, para lo cual se recurre a dos registros, los cuales están asociados a esta función:

- CCPR1L (Bit 7:0).
- CCP1CON (Bit 5:4).

Estos dos registros permiten que el ciclo de trabajo tenga una resolución de 10 bits.

El registro CCPR1L corresponden los bits más significativos del ciclo de trabajo, mientras que los bits 5 y 4 del registro CCP1CON corresponden a los dos bits menos significativos del ciclo de trabajo.

El ciclo de trabajo responde a la siguiente fórmula:

$$CT = \left(CCPR1L{:}\,CCP1CON(5{:}4)\right) * T_{OSC} * (Preescalado\ del\ TMR2)$$

Ahora calculamos el valor a introducir en los registros CCPR1L y CCP1CON, para lo cual, despejamos de la fórmula:

$$\left(CCPR1L{:}\,CCP1CON(5{:}4)\right) = \frac{CT}{T_{OSC} * (Preescalado\ del\ TMR2)}$$

Dado que el ciclo de trabajo que vamos a usar es del 50%, el ciclo positivo del PWM durará la mitad del periodo PWM, y además sabiendo que el reloj de oscilación es de 7,3728 MHz, procedemos a reemplazar en la fórmula:

$$\left(CCPR1L{:}\,CCP1CON(5{:}4)\right) = \frac{\frac{1}{2} * \frac{1}{120\ KHz}}{\frac{1}{7{,}3728\ MHz} * (16)} = 1{,}92$$

$$\therefore \left(\boldsymbol{CCPR1L{:}\,CCP1CON(5{:}4)}\right) = \mathbf{2}$$

El siguiente paso es configurar el Timer 2 como PWM, para lo cual, modificaremos el registro CCP1CON (figura 2.52), en el cual, utilizaremos los siguientes bits:

- *Bits de Selección de Modo CCP (Bit 5:2):* Estos cuatro bits configuran el funcionamiento del Timer 2, para lo cual, lo configuraremos como PWM.

CCP1CON							
7	6	5	4	3	2	1	0
-	-	1	0	1	1	X	X

Figura 2.52: Configuración del registro CCP1CON para configurar el timer 2 como PWM

Finalmente configuramos un pre-escalado de 16 y habilitamos el timer 2, para lo cual usamos el registro T2CON (figura 2.53) de la siguiente manera:

- *Bit de Habilitación del TIMER2 (Bit 2):* Este bit habilita el funcionamiento del TIMER2.

- *Bits de Selección del Pre-escalador del Timer 2 (Bit 1:0):* Bits de selección del Pre-escalador:

  00: Pre-escalador a 1

  01: Pre-escalador a 4

  1X: Pre-escalador a 16

T2CON							
7	6	5	4	3	2	1	0
-	X	X	X	X	1	1	X

Figura 2.53: Configuración del registro T2CON para habilitar el timer 2 y su preescalador

2.3.2.3 Rutina de Interrupciones

Después de las subrutinas de Inicio, el microcontrador PIC, no realiza ninguna acción además de la espera de una interrupción, pudiendo ser fuente de interrupción:

- El puerto USART al recibir una transmisión serial por parte del FPGA.
- Una interrupción externa en el pin RB0 indicando un cruce por cero de la señal de 60Hz.

- Una interrupción del Timer0 indicando el fin del conteo de 1 ms después de habilitada la señal PWM.

Las interrupciones que va a recibir el PIC pueden ser de transmisión ó de recepción, dependiendo de la tarea que realiza un determinado PIC.

A continuación en la figura 2.54 se muestra la lógica del funcionamiento de la rutina de interrupciones para un PIC que va a transmitir códigos X10.

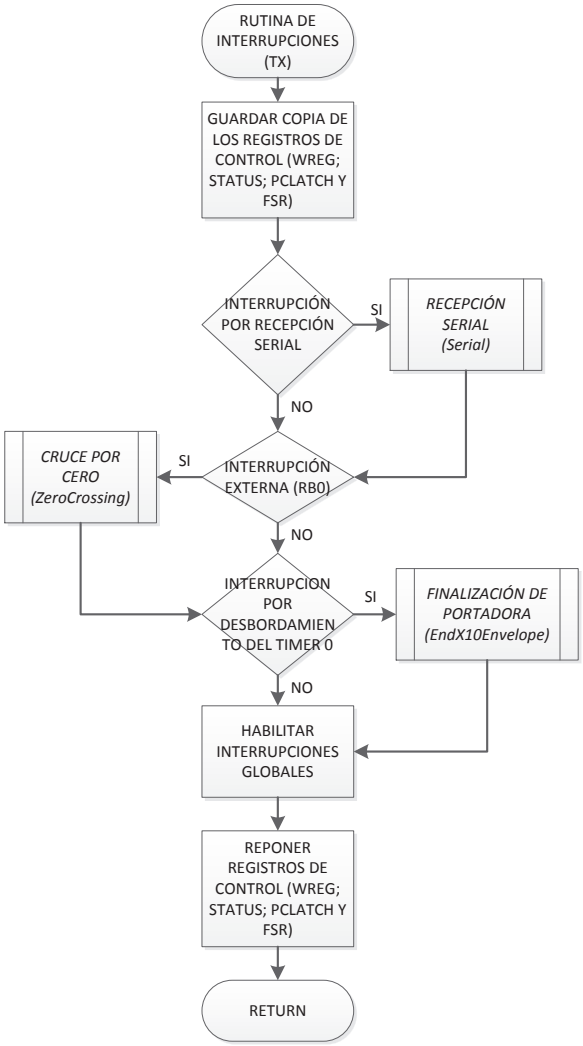


Figura 2.54: Rutina de interrupciones de un transmisor X10

Por otro lado, en la figura 2.55 se muestra la lógica del funcionamiento de la rutina de interrupciones para un PIC que va a recibir códigos X10.

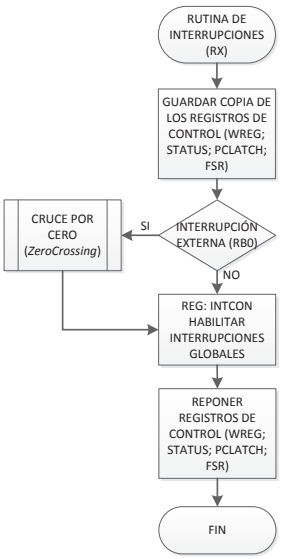


Figura 2.55: Rutina de interrupciones de un receptor X10

Todas las subrutinas que se observan en las figuras 2.54 y 2.55 se encargan de procesar las diferentes interrupciones provenientes del timer 0 (conteo de 1 ms - Tx), de la interrupción externa (cruce por cero – Tx y Rx) y de la recepción del puerto USART (Tx). Para realizar este procesamiento se siguen tres pasos:

1. **Guardar los registros de control antes de la interrupción:** Consiste en guardar los registros de control en registros temporales antes de comenzar las subrutinas de interrupciones, ya que almacenan la configuración temporal del funcionamiento del microcontrolador. Estos registros son los siguientes:

    - **Registro Status:** Contiene el estado aritmético de la ALU, el estado del Reset y los bits de selección de banco.
    - **Registro Pclath:** Es un registro utilizado para el manejo de tablas, en este se almacena el número de página donde se encuentra almacenado el programa.

    - **Registro W:** Es un registro ampliamente usado que ayuda a almacenar datos temporales.

- **Registro FSR:** Este registro es usado para el direccionamiento indirecto, en este registro se guarda la dirección de memoria que se va a direccionar indirectamente.

2. **Detectar y ejecutar la subrutina de interrupción:** Dependiendo de la razón de la interrupción se puede tener tres subrutinas:

- ***Subrutina para Recepción Serial (Serial)***

  En la figura 2.56 se observa la lógica de trabajo de esta subrutina, la cual se muestra a continuación:

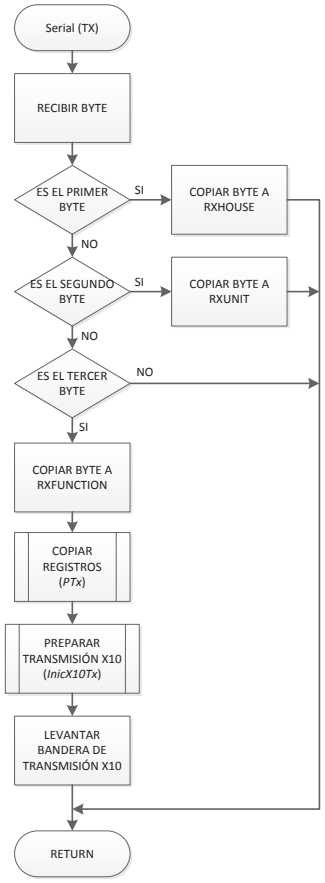


Figura 2.56: Subrutina para interrupción serial

Como se puede observar en la figura 2.56, esta subrutina se encarga de procesar la recepción serial. La comunicación serial está programada de tal manera, que el

FPGA envía tres bytes en formato X10, incluyendo la dirección de Casa, Unidad y Función. Por lo cual, la interrupción por Recepción Serial se encarga de recibir estos tres bytes y asignarlos a los registros correspondientes (TxHouse, TxUnit y TxFunction respectivamente). Después de haber recibido el tercer byte se sigue una secuencia de dos subrutinas encargadas de preparar los registros para la transmisión. Estas subrutinas son las siguientes:

Subrutina para Copiar Registros (PTx):

Internamente para la transmisión de los mensajes X10, el PIC maneja 4 registros temporales (TxData0, TxData1, TxData2 y TxData3) encargados del envío bit a bit del mensaje. Esta subrutina se encarga de copiar la información a enviar en el primer mensaje a los registros temporales como se observa en el diagrama de flujo de la figura 2.57.

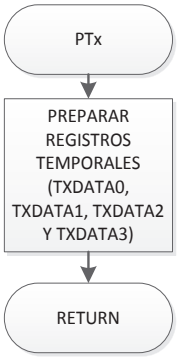


Figura 2.57: Subrutina PTx

Subrutina para preparar la Transmisión X10 (InitX10Tx):

Esta subrutina se encarga de preparar los registros temporales para su transmisión bit a bit. Se utiliza la subrutina para rotar los registros temporales 4 veces (RotateTxData), el objetivo de rotar estos registros es ubicar el primer bit a transmitir como el bit más significativo del registro temporal TxData3, ya que en cada cruce por cero se transmite el bit más significativo del último registro temporal y se rota una vez los registros temporales, así los bits son transmitidos uno por uno en cada cruce por cero de la señal de AC. En el diagrama de flujo de la figura 2.58 se observa el comportamiento de esta subrutina:

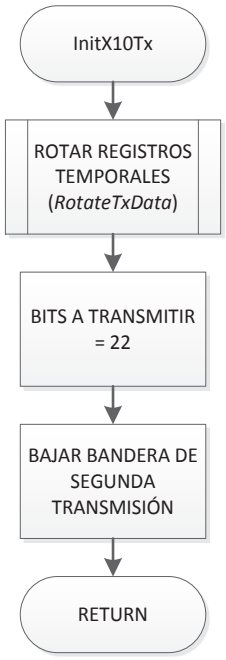


Figura 2.58: Subrutina InitX10Tx

Además, para poder administrar las transmisiones, se utilizan banderas que indican al microcontrolador PIC si se va a transmitir un mensaje o un bloque. Por ejemplo en la figura 2.59 se muestran los registros temporales antes de rotar, mientras que en la figura 2.60 se muestran los registros temporales después de rotar.

Registro Temporal TxData3:

0	**0**	**0**	**0**	**CI3**	**CI2**	**CI1**	**CI0**

Registro Temporal TxData2:

H7	**H6**	**H5**	**H4**	**H3**	**H2**	**H1**	**H0**

Registro Temporal TxData1:

C7	**C6**	**C5**	**C4**	**C3**	**C2**	**C1**	**C0**

Registro Temporal TxData0:

P1	**P0**	**0**	**0**	**0**	**0**	**0**	**0**

Figura 2.59: Registros temporales antes de rotar

Registro Temporal TxData3:

CI3	CI2	CI1	CI0	H7	H6	H5	H4

Registro Temporal TxData2:

H3	H2	H1	H0	C7	C6	C5	C4

Registro Temporal TxData1:

C3	C2	C1	C0	P1	P0	0	0

Registro Temporal TxData0:

0	0	0	0	0	0	0	0

Figura 2.60: Registros temporales después de rotar

Finalmente, se procede a establecer el número de bits a transmitir en 22 (11 bits del mensaje X10 transmitido dos veces) y a levantar la bandera de transmisión.

Subrutina para rotar los registros temporales de transmisión (RotateTxData):

Esta subrutina se encarga de rotar los registros temporales de transmisión el número de veces en las que se configure el contador. Esta subrutina es útil para preparar los registros temporales para una nueva transmisión, ya sea de mensaje o bloque y a continuación, en la figura 2.61 se muestra la lógica de funcionamiento de esta subrutina.

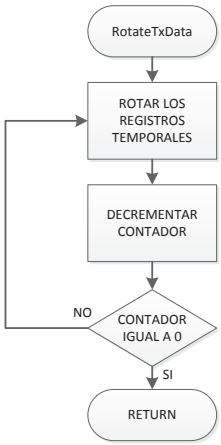


Figura 2.61: Subrutina RotateTxData

Subrutina para el Cruce por Cero de la Señal Eléctrica (ZeroCrossing)

Esta subrutina se da tanto en el transmisor como en el receptor. El objetivo de la subrutina de interrupción externa es el de ejecutar la transmisión y recepción de cada bit X10 en cada cruce por cero. En el transmisor, se tienen las subrutinas para transmitir el bit X10 y para preparar el siguiente bit a transmitir. En el lado del receptor se tiene de igual manera, una subrutina para recibir el bit X10 y otra para preparar el bit recibido. Después de estar subrutinas, ambos programas terminan con el cambio de flanco de interrupción externa y limpieza de la bandera de interrupción externa.

Subrutina del cruce por cero para el proceso de transmisión:

A continuación en la figura 2.62 se muestra el diagrama de flujo de esta subrutina para el proceso de transmisión.

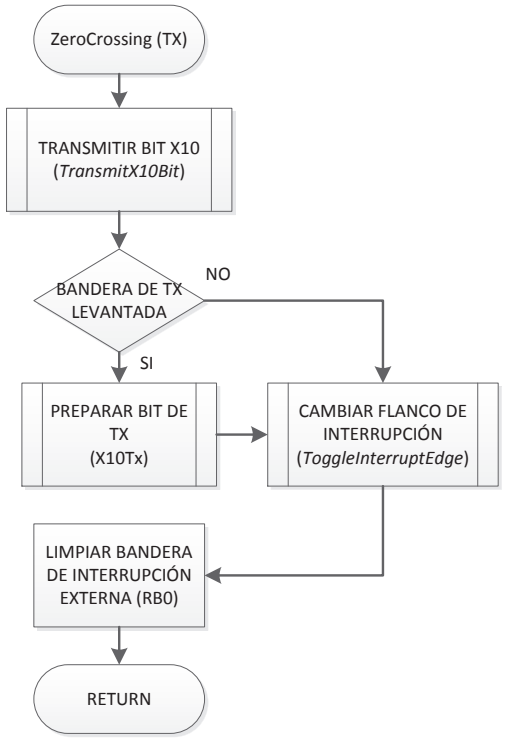


Figura 2.62: Subrutina ZeroCrossing para la transmisión

Subrutina para Transmitir Bit X10 (TransmitX10Bit):

El diagrama de flujo de esta subrutina se muestra a continuación en la figura 2.63.

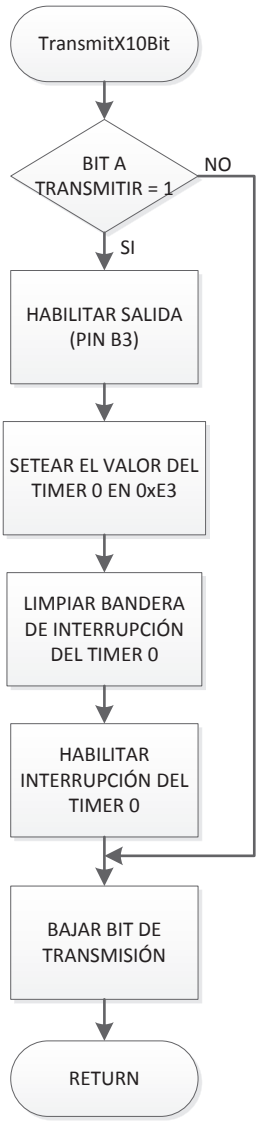


Figura 2.63: Subrutina TransmitX10Bit

Esta subrutina se encarga de transmitir el bit X10 correspondiente (determinador por el Bit TxBit del registro X10Flags). Cuando el microcontrolador tiene que enviar un 1L por la red eléctrica, este habilita la salida de la señal PWM e iniciar el contador de 1 ms, por el contrario, si el bit a transmitir es un 0L, no se realiza ninguna acción. Para el caso de enviar un 1L, el microcontrolador habilita el pin de salida de la señal PWM, cambiando la configuración de este pin (RB3) como salida. Para el tiempo de

transmisión de la señal PWM de 1 ms, se debe configurar el conteo en el microcontrolador, para este efecto, se debe calcular el valor del registro TMR0 para que realice un conteo por un 1ms, por lo cual, tenemos:

$$\boldsymbol{Ciclos(Sin\ preescalador)} = \frac{1ms}{4 * T_{OSC}} = \frac{F_{OSC} * 1ms}{4} = \frac{7{,}3728\ MHz * 1ms}{4}$$

$$\boldsymbol{Ciclos(Sin\ preescalador)} = 1.843{,}2$$

$$\boldsymbol{Ciclos(Preescaladorx64)} = \frac{1.843{,}2}{64} = 28{,}8$$

$$\therefore \boldsymbol{Ciclos(Preescaladorx64)} = 29$$

Esto nos dice que el microcontrolador tiene que contar 20 ciclos de reloj con un pre-escalador de 64. Para lo cual, para saber el valor a ser configurado en el registro TMR0, se debe restar este valor del conteo total del microcontrolador, que será de 256. Ya que al finalizar el conteo del TIMER0, se dispara la interrupción por desbordamiento, indicando que se debe deshabilitar la salida de la señal PWM.

$$\boldsymbol{Valor\ registro\ TMR0} = 256 - 29 = 227 = 0xE3 \text{ (Hexadecimal)}$$

Subrutina para preparar la transmisión X10 (X10Tx):

Recordando lo explicado en el capítulo 1, tenemos que una transmisión completa X10 (Casa, Unidad y Función), consta de 2 bloques X10, cada bloque está compuesto de 2 mensajes X10, y cada mensaje X10 está compuesto por un código de casa y un código de unidad o función (1er Bloque o 2do Bloque respectivamente). Cada mensaje consta de 22 bits y entre cada bloque se debe dejar tres ciclos de la señal eléctrica sin transmitir (6 ceros). Esta subrutina se encarga de preparar los bits para su transmisión en el siguiente cruce por cero de la red eléctrica. Por lo cual, su función es preparar el bit a transmitir (TxBit) y rotar los bits restantes para su posterior transmisión. Además, controla el conteo del número de bits, mensajes y bloques transmitidos. Para este fin, se utiliza un contador de bits transmitidos, el cual,

al terminar de enviarse el primer mensaje X10 (22 bits), rota los registros temporales 10 veces para volver a tener el mismo orden inicial (Init2ndTx). Posteriormente, inicia la transmisión del 2do mensaje X10, en este caso, los bits a transmitir son 28, incluyendo los 3 ciclos de espera de la señal eléctrica (6 cruces por cero). A continuación, al terminarse la segunda transmisión del primer bloque inicia la primera transmisión del segundo bloque. Pero primero prepara los registros temporales (Init2ndBloque). Luego de terminada la segunda transmisión del segundo bloque, la subrutina se encarga de bajar todas las banderas utilizadas y limpiar todos los registros temporales utilizados.

A continuación en la figura 2.64 se observa el diagrama de flujo que describe el comportamiento de esta subrutina.

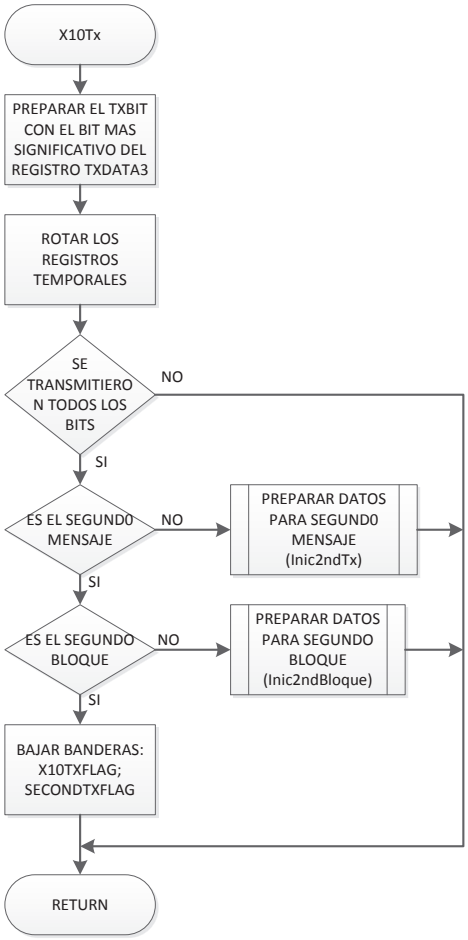


Figura 2.64: Subrutina TransmitX10Tx

Subrutina para preparar la transmisión del segundo mensaje X10 (Inic2ndTx):

Puesto que los registros temporales de transmisión después de la transmisión del primer mensaje X10 quedan desfasados diez veces de la posición original de transmisión, se usa esta subrutina para preparar estos registros de transmisión, restablecer los bits a transmitir en 28 (22 bits del mensaje X10 añadido los seis ciclos de espera entre bloques X10) y levantar la bandera de transmisión del segundo mensaje X10, a continuación en la figura 2.65 se muestra la lógica de esta subrutina.

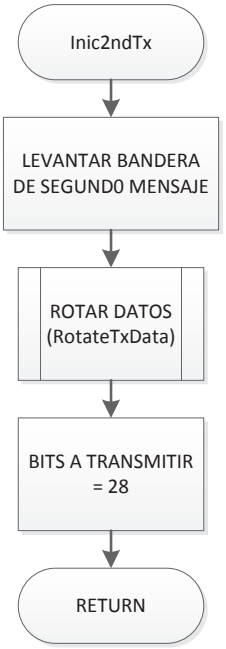


Figura 2.65: Subrutina Inic2ndTx

Subrutina para preparar la transmisión del segundo bloque X10 (Inic2ndBloque):

Para preparar la transmisión del segundo bloque se vuelven a cargar los registros temporales con los valores de la dirección de casa y función, además del prefijo que identifica al mensaje X10 como un mensaje de función. De igual manera como en el primer bloque, se preparan los registros temporales para iniciar la transmisión bit a bit en cada cruce por cero de la red eléctrica, con la subrutina para preparar la transmisión X10 (InicX10Tx). Al igual que en el resto de subrutinas, se utilizan banderas que indican la transmisión del segundo bloque, como se puede observar a continuación en la figura 2.66.

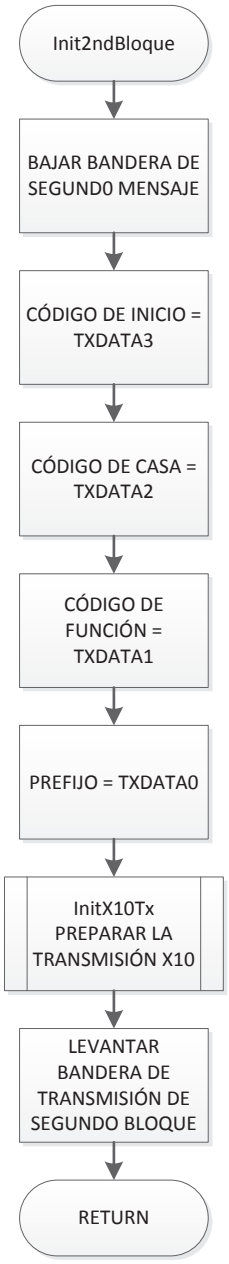


Figura 2.66: Subrutina Inic2ndBloque

Subrutina del cruce por cero para el proceso de recepción:

A continuación en la figura 2.67 se muestra el diagrama de flujo de esta subrutina para el proceso de recepción.

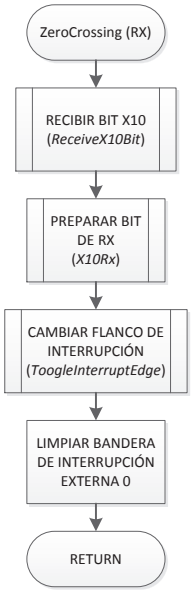


Figura 2.67: Subrutina ZeroCrossing para la recepción

Subrutina para recibir un bit X10 (ReceiveX10Bit):

Esta subrutina se encarga de detectar la presencia de la señal portadora de 120 KHz. Para lo cual, lee el puerto de recepción (RB3) y setea el bit RxBit si el puerto esta en 1L, caso contrario baja el bit RxBit, tal y como se muestra en el diagrama de flujo de la figura 2.68.

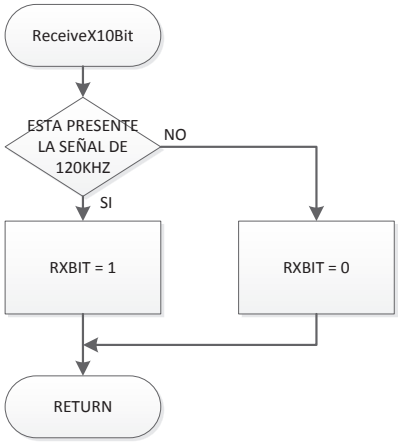


Figura 2.68: Subrutina ReceiveX10Bit

Subrutina para preparar el Bit de Recepción (X10Rx):

Esta subrutina copia el último bit recibido a los registros temporales y determina si se completó la recepción. El indicador que le mostrará la recepción completa será la presencia del código de inicio. Si el código de inicio se encuentra presente, el programa ejecutara una subrutina para guardar los códigos X10 recibidos, como se muestra en la figura 2.69 a continuación:

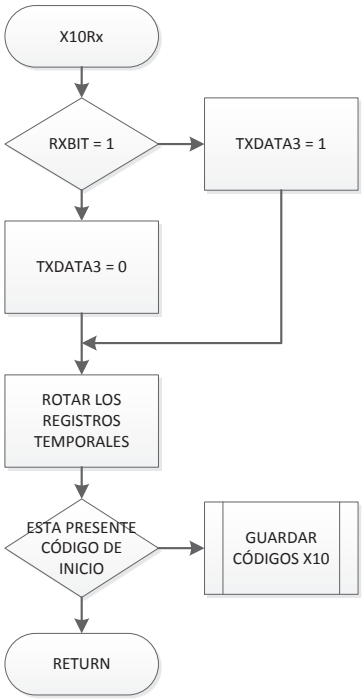


Figura 2.69: Subrutina X10Rx

Subrutina para guardar los Códigos X10:

Esta subrutina se encarga de almacenar los códigos de Unidad y Función recibidos. Primero, se comprueba que los códigos de Casa recibidos en ambas transmisiones concuerden. Si concuerdan, entonces, se procede a determinar si es un código de casa o función y los almacena según corresponda (Sea esta la dirección del receptor o la dirección de todas las casas). A continuación en la figura 2.70 se muestra la lógica de funcionamiento de esta subrutina.

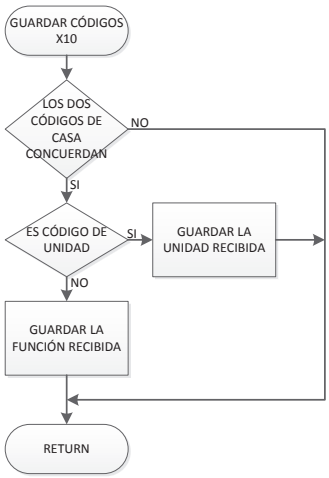


Figura 2.70: Subrutina para guardar códigos X10

Subrutina para Cambiar el Flanco de Interrupción (ToggleInterruptEdge):

Puesto que al microcontrolador PIC se le debe especificar con qué flanco de la interrupción externa va a trabajar, después de haber terminado la subrutina de interrupción de detección por cruces por cero, el programa preparara al PIC para recibir la siguiente interrupción externa por el flanco correspondiente, sea este de subida o de bajada, tal y como se muestra a continuación en la figura 2.71.

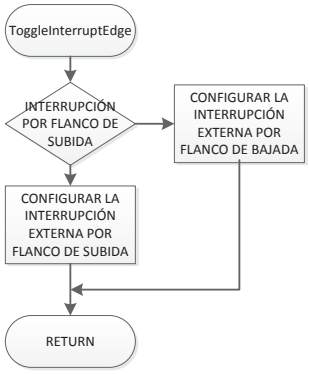


Figura 2.71: Subrutina ToggleInterruptEdge

Finalmente se baja la bandera de interrupción externa y se finaliza la subrutina de detección de cruces por cero.

Subrutina para Finalizar Portadora (EndX10Envelope)

También conocida como subrutina de interrupción por desbordamiento del timer 0. El Timer 0 es utilizado como contador para determinar el final del tiempo de transmisión de la señal portadora PWM (1 mS). Por lo cual, finalizado el conteo del Timer 0 se disparara la interrupción por desbordamiento del Timer 0. Al recibir la señal de la interrupción por desbordamiento del Timer 0, el microcontrolador PIC deshabilitara la salida de la señal PWM. Para finalizar la interrupción se baja la bandera de interrupción por desbordamiento del timer 0 y se deshabilita la interrupción por desbordamiento del Timer 0. Esta lógica de programación se describe a continuación en la figura 2.72.

Figura 2.72: Subrutina EndX10Envelope

Finalmente, el último paso del procesamiento de las interrupciones es:

3. **Restablecer los registros de control a sus valores antes de la interrupción:** Este paso consiste en cerrar las subrutinas de interrupciones, habilitando nuevamente las interrupciones globales y reponiendo los registros de control.

De esta manera se consigue que los circuitos transmisor y receptor X10 puedan comunicarse, llevando la información del protocolo X10 desde el lado del controlador hacia los respectivos actuadores.

Adicionalmente, es importante recalcar que toda la programación desarrollada en los PICS transmisor y receptor, se encuentra detallada en el Anexo 18, por lo cual se recomienda revisar este anexo con la finalidad de analizar y comprender el código fuente del transporte de la información X10 dentro del sistema domótico.

2.3.3 DISEÑO DEL PROGRAMA PARA EL ACTUADOR DE ILUMINACIÓN

El programa para el control de iluminación trabaja en conjunto al circuito actuador correspondiente, con la finalidad de permitirle al usuario controlar la iluminación total de su vivienda. Para ello se utiliza el microcontrolador ATMEGA8535, sobre el cual desarrollamos el programa en lenguaje assembler. A continuación, en la figura 2.73 se muestra la lógica del programa para el control de iluminación de nuestro sistema domótico:

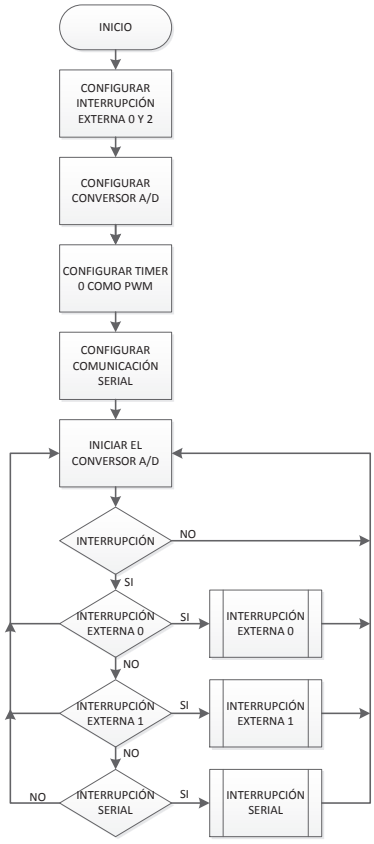


Figura 2.73: Diagrama de flujo del programa para el actuador de iluminación

El programa consta de varias etapas, las cuales se describen de manera detallada a continuación:

1. Configuración de las interrupciones externas 0 y 1

La interrupción externa 0 le permite al microcontrolador sincronizar el PWM con los cruces por cero, mientras que la interrupción externa 1 sirve para que el microcontrolador reconozca el tipo de control que el usuario desea ejecutar, sea este manual, mediante el uso de un potenciómetro, o remoto, mediante el uso de la interfaz gráfica de usuario.

Cabe recalcar que los cambios en las señales de activación de las interrupciones ocurren en ambos flancos, por esta razón se configuran las interrupciones para que se disparen en cualquier cambio de la señal de control.

2. Configuración del Conversor A/D

El conversor A/D es la fuente del control manual de iluminación puesto que toma el valor de voltaje del potenciómetro externo (0 VDC a 5 VDC) y lo convierte a un valor digital entre 0 y 255. Este nuevo valor digital se utiliza para modificar el periodo PWM.

Puesto que la fuente de conversión es el voltaje que ingresa por el pin AD0 (40), es importante que en la configuración del conversor se seleccione como voltaje de referencia al proveniente del pin AVCC (30) del microncontrolador.

3. Configuración del Timer 0 como PWM

El PWM es la señal de control de la intensidad de luz del sistema. Esta señal tiene una lógica negativa, es decir, cuando el PWM presente un ciclo positivo mayor que el negativo, la intensidad de luz será baja y viceversa.

El PWM se lo configura de tal forma que se usa el modo PWM rápido, siendo éste el más simple, ya que utiliza un contador cuya salida se setea en 0L al iniciar, y cambia su estado cuando el conteo iguala al valor del registro de control. El registro

de control PWM es un registro que va de 0 a 255 y que determina el ancho de pulso que el PWM va a tener, así entre menor sea este valor menor será el ancho del pulso PWM y viceversa.

El periodo de la señal PWM debe ser la mitad del periodo de la señal eléctrica, de tal manera, que haya un control de fase directo en cada semi-ciclo de la señal eléctrica. Para ello se realiza el cálculo de la frecuencia de reloj del microcontrolador utilizando la siguiente fórmula:

$$\boldsymbol{f_{PWM}} = \frac{f_{CLK}}{N * 256}$$

Dónde:

f_{PWM}: Frecuencia del PWM

f_{CLK}: Frecuencia de reloj del microcontrolador

N: Pre-escalador del microcontrolador

Para usar la frecuencia de reloj del microcontrolador de 8 MHz, se debe calcular el valor del pre-escalador de la siguiente manera:

$$\boldsymbol{N} = \frac{f_{CLK}}{f_{PWM} * 256} = \frac{8\ MHz}{120\ Hz * 256}$$

$$\boldsymbol{N = 260,42}$$

Dado que los valores de pre-escaladores del microcontrolador están previamente definidos, se debe escoger el valor que más se acerque al N requerido. Para lo cual se utiliza un pre-escalador de 256, con lo cual se tiene el siguiente valor de frecuencia de la señal PWM:

$$\boldsymbol{f_{PWM}} = \frac{f_{CLK}}{N * 256} = \frac{8\ MHz}{256 * 256}$$

$$\boldsymbol{\therefore f_{PWM} = 122,07\ Hz}$$

Este valor obtenido de 122,07 Hz se aproxima mucho al valor de frecuencia deseado, por lo cual utilizamos el valor 256 como pre-escalador en este programa.

4. Configuración de la Comunicación Serial

La comunicación serial se utiliza para que el microcontrolador reciba el comando a ejecutar por parte del dispositivo receptor X10.

Dentro del programa se configura una comunicación serial asíncrona de 8bits de datos, 1 bit de parada, sin bit de paridad y velocidad de recepción de 9600 Baudios.

Además, para configurar la velocidad de recepción se debe calcular el valor del registro UBRR, de la siguiente manera:

$$\boldsymbol{UBRR} = \frac{f_{CLK}}{16(Baud\ rate)} - 1 = \frac{8\ MHz}{16\ (9600\ Baudios)} - 1$$

$$\boldsymbol{UBRR} = 51{,}08$$

$$\therefore \boldsymbol{UBRR} = 51$$

Dónde:

UBRR: Registro indicador de la velocidad de la comunicación serial.

$\boldsymbol{f_{CLK}}$**:** Frecuencia del Reloj

De esta manera se permite la recepción y posterior ejecución del comando que recibió el microcontrolador por comunicación serial.

Como se mencionó anteriormente, el control de iluminación puede ser manual o remoto:

- ***Control manual:*** En este caso la iluminación es controlada por el potenciómetro externo, el cual varía el voltaje de entrada al conversor A/D, variando directamente el ancho del PWM.

- ***Control remoto:*** En este caso la iluminación es controlada por la recepción de comandos vía comunicación serial, esta variación es fija teniendo un total de 8 pasos para llegar al cien por ciento de iluminación. Esta variación fija se obtiene al sumar o restar el valor de 32 al registro de control del PWM. Este valor de 32 se obtiene al dividir el valor máximo del registro PWM sobre el número de pasos deseados, de la siguiente manera:

$$Constante\ de\ Variación\ del\ PWM = \frac{256}{Número\ de\ Niveles} = \frac{256}{8}$$

$$\boldsymbol{Constante\ de\ Variación\ del\ PWM = 32}$$

Cabe recalcar que el programa del control de iluminación permanece haciendo una conversión A/D, a la espera de alguna interrupción, las cuales se describen a continuación:

- ***Interrupción Externa 0 (Detección de Cruces por cero)***

Como se observa en el diagrama de flujo de la figura 2.74, esta interrupción se dispara cuando la señal eléctrica cruza la línea de 0 VAC, en la cual se reinicia el PWM y se setea el nuevo valor del registro de control PWM.

Esta interrupción sirve para sincronizar el inicio de la señal PWM con el cruce por cero. Además, dependiendo del tipo de control de iluminación, el nuevo valor que se va a cargar en el registro de control del PWM es el proveniente de la conversión A/D (Control Manual) o el proveniente de la comunicación serial (Control Remoto).

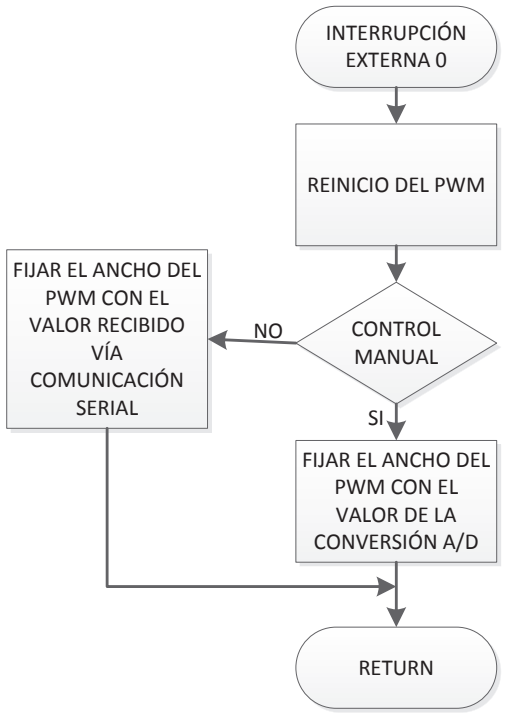


Figura 2.74: Diagrama de flujo de la interrupción externa 0

- ***Interrupción Externa 1 (Cambio a Control Manual de Iluminación)***

Esta interrupción se dispara cada vez que el usuario elige utilizar el control manual de iluminación. Además permite que en la detección de cruces por cero, al fijar el nuevo valor de control PWM, se utilice el valor de la última conversión A/D. A continuación en la figura 2.75 se muestra el diagrama de flujo de esta interrupción.

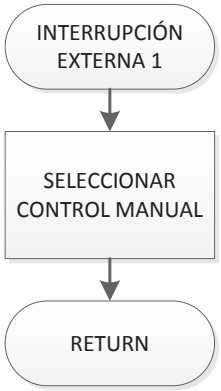


Figura 2.75: Diagrama de flujo de la interrupción externa 1

- ***Interrupción por Recepción Serial (Recepción del comando enviado por el Receptor X10)***

Como se observa en el diagrama de flujo de la figura 2.76, esta interrupción se dispara cuando el actuador recibe un comando vía comunicación serial desde el receptor X10, los comandos que puede recibir el actuador son:

- **BRIGHT:** Este comando aumenta en un nivel la intensidad de luz. Cuando la intensidad de iluminación llega a su nivel máximo (Nivel 7), este comando no tiene efecto.

- **DIMMER:** Este comando disminuye en un nivel la intensidad de luz. Cuando la intensidad de iluminación llega a su nivel mínimo (Nivel 0), este comando no tiene efecto.

- **ENCENDER:** Este comando aumenta en un solo paso la intensidad de luz a su máximo nivel posible (Nivel 7), esto se logra seteando el puerto de salida PWM en 1L.

- **APAGAR:** Este comando disminuye en un solo paso la intensidad de luz a su mínimo nivel posible (Nivel 0), esto se logra seteando el puerto de salida PWM en 0L.

Luego de procesar el comando recibido, de ser el caso, el actuador envía vía comunicación serial este comando, para que sea procesado por el siguiente actuador conectado a él.

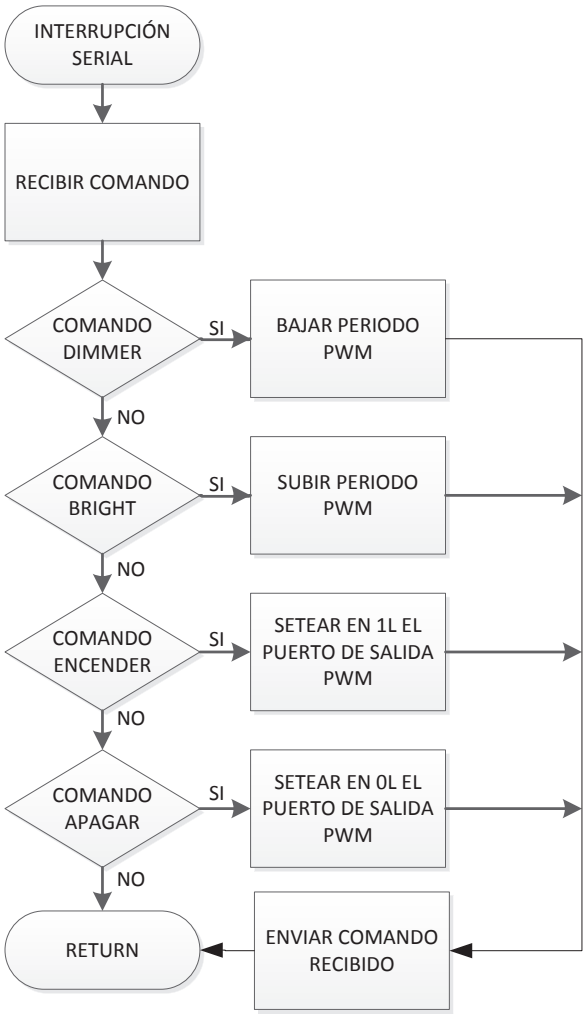


Figura 2.76: Diagrama de flujo de la interrupción por recepción serial

Por lo expuesto anteriormente, se puede evidenciar que el microcontrolador se encuentra en continuo funcionamiento, ya sea realizando la conversión A/D o atendiendo alguna de las interrupciones, de tal manera que está dispuesto a ejecutar en cualquier momento el control de la iluminación requerida por el usuario.

Finalmente, cabe recalcar que toda la programación desarrollada para este actuador, se encuentra detallada en el Anexo 19, por lo cual se recomienda revisar este anexo con la finalidad de analizar y comprender el código fuente del control de la iluminación en el prototipo del sistema domótico.

2.3.4 DISEÑO DEL PROGRAMA PARA EL ACTUADOR DE ACCESOS Y PERSIANAS

El programa para el control de accesos (puertas) y persianas trabaja en conjunto a los circuitos actuadores correspondientes, con la finalidad de permitirle al usuario controlar los accesos y las persianas principales en su vivienda. Para ello se utiliza el microcontrolador ATMEGA8535, sobre el cual desarrollamos el programa en lenguaje assembler.

Para el control de puertas y persianas se utiliza un motor a pasos. Dado que el control del motor es de doble paso, se procede a configurar el programa de tal manera que permite la activación secuencial y simultánea de dos pines del puerto de salida que se encuentra conectado al motor.

Puesto que la velocidad del reloj del microcontrolador es alta, se utiliza una subrutina de retardo que le da tiempo suficiente al eje del motor para moverse hasta que se active el siguiente paso. Para elegir el sentido de movimiento del motor se utiliza una secuencia ascendente o descendente, la cual se basa en la secuencia propia del motor.

El control del giro del motor se lo hace por tiempo, es decir, en el programa se determina el tiempo que el eje del motor va a permanecer en movimiento. Este tiempo se lo programa en base al cálculo de un retardo con una duración igual al tiempo que el motor va a permanecer girando.

El motor se mueve en pasos de un ciclo, llegando a un máximo de cuatro ciclos, es decir, que la persiana/puerta se encuentran cerrados en el ciclo 0 y se encuentran totalmente abiertos en el ciclo 4.

Cada ciclo de giro de las persianas corresponden a un 25% de la elongación total de cada persiana, siendo cuatro ciclos los necesarios para abrir/cerrar totalmente una persiana.

A continuación, en la figura 2.77 se muestra la lógica del programa para el control de puertas y persianas de nuestro sistema domótico:

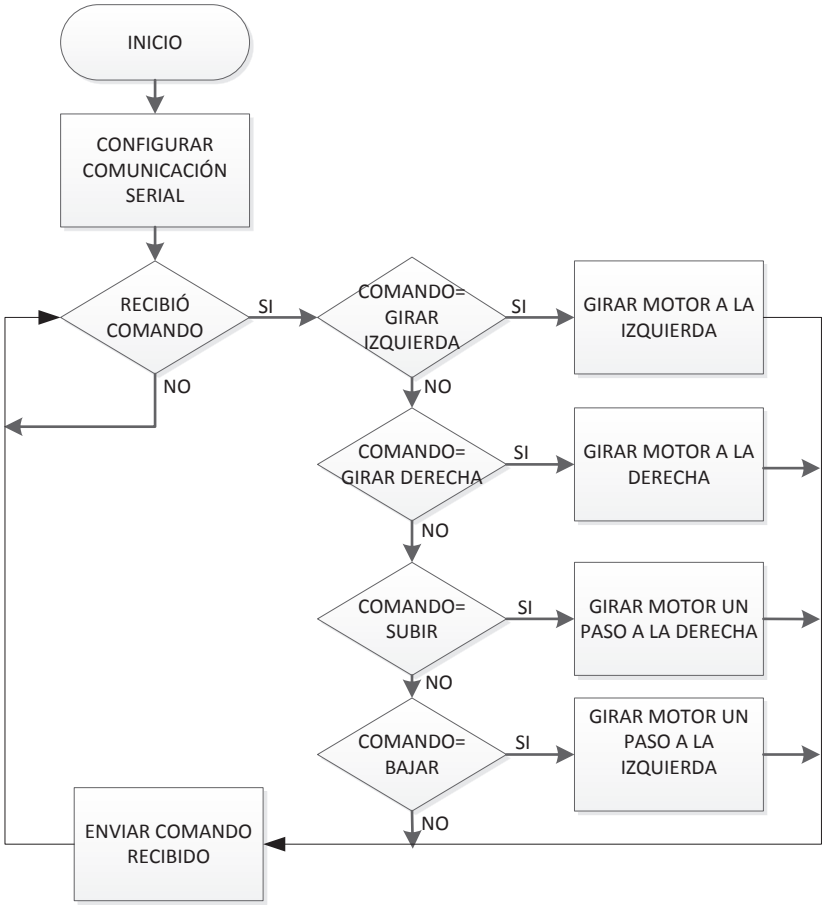


Figura 2.77: Diagrama de flujo del programa para el control de puertas y persianas

El programa inicia con la configuración del puerto USART para comunicación serial, como se describe a continuación.

1. Configuración del puerto USART

De igual manera que en el actuador de iluminación, este actuador recibe comandos por parte del Receptor X10 vía comunicación serial. Por lo tanto, la configuración de la comunicación serial es la misma, es decir 8 bits de datos, 1 bit de parada, sin bit de paridad y con una velocidad de recepción de 9600 Baudios.

En el funcionamiento del programa, el microcontrolador está a la espera de recibir un comando por parte del Receptor X10. Los comandos que el actuador puede recibir son:

- **SUBIR:** Este comando le permite al microcontrolador girar totalmente en sentido horario (cuatro ciclos), haciendo que la persiana/puerta se abra totalmente.

- **BAJAR:** Este comando le permite al microcontrolador girar totalmente en sentido anti-horario (cuatro ciclos), haciendo que la persiana/puerta se cierre totalmente.

- **GIRAR IZQUIERDA:** Este comando le permite al microcontrolador girar un ciclo en sentido horario, haciendo que la persiana se abra parcialmente.

- **GIRAR DERECHA:** Este comando le permite al microcontrolador girar un ciclo en sentido anti-horario, haciendo que la persiana se cierre parcialmente.

De igual manera que en el actuador de luces, el actuador para el control de puertas y persianas, envía vía comunicación serial el comando recibido, para que el mismo pueda ser procesado, de ser el caso, por el siguiente actuador conectado a él.

Por lo expuesto anteriormente, se puede evidenciar que el microcontrolador está dispuesto a ejecutar en cualquier momento el control de las puertas o de las persianas requeridas por el usuario.

Adicionalmente, se reitera que la programación desarrollada para este actuador, se encuentra detallada en el Anexo 20, por lo cual se recomienda revisar este anexo con la finalidad comprender el código fuente para poder controlar el movimiento del motor tanto en el caso de las puertas como de las persianas dentro del sistema domótico.

2.4 DISEÑO DE LA INTERFAZ DE USUARIO

La interfaz de usuario es el medio que permite la comunicación entre personas y máquinas, es decir que por medio de la interfaz un usuario puede controlar a su máquina y ejecutar las funciones que el desea.

En el prototipo de sistema domótico se ha diseñado una interfaz gráfica de usuario con la finalidad de establecer la comunicación usuario-máquina.

2.4.1 DISEÑO DE LA INTERFAZ GRÁFICA DE USUARIO [32]

Una interfaz gráfica de usuario (GUI) permite la comunicación entre el usuario y la máquina por medio de un entorno gráfico. Es decir que por medio de una aplicación visual una persona puede ordenar a su máquina que ejecute una acción eligiendo el ícono que representa a determinada tarea. En el prototipo de sistema domótico la interfaz gráfica que le permite al usuario controlar su hogar fue desarollada en LabVIEW.

2.4.1.1 LABVIEW (Laboratory Virtual Instrument Engineering Workbench) [7]

LabVIEW de National Instruments es un entorno de desarrollo de aplicaciones que se caracteriza por utilizar un lenguaje gráfico de programación. Este programa permite además la adquisión, análisis y presentación de datos.

Las aplicaciones que se desarrollan en Labview se denominan VIs (Instrumentos Virtuales), puesto que simulan a los instrumentos que podemos usar en la vida real.

Un VI está conformado por 3 partes:

1. ***Panel Frontal:*** Es la pantalla principal que muestra la interfaz gráfica de usuario. Este panel se puede modificar o decorar para que sea más agradable a la persona que lo va a utilizar. A continuación, en la figura 2.78 se observa el panel frontal de nuestra aplicación.

Figura 2.78: Panel Frontal de LabVIEW

2. ***Diagrama de bloques:*** Es la pantalla donde se realiza la programación gráfica del VI. Es decir que en esta pantalla se configura lo que el usuario va a observar en la interfaz gráfica. A continuación, en la figura 2.79 se observa el diagrama de bloques de nuestra aplicación.

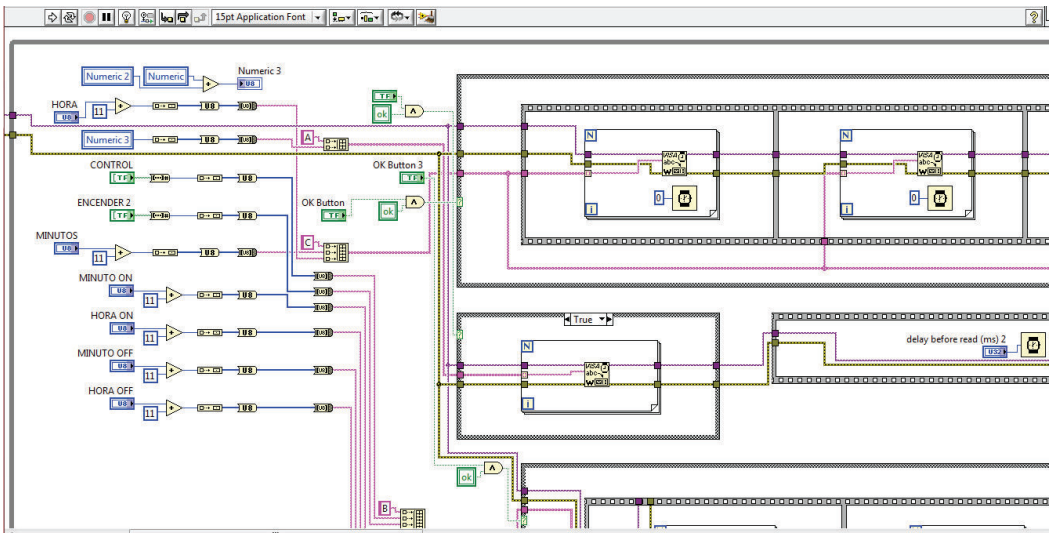


Figura 2.79: Diagrama de Bloques de LabVIEW

3. ***Paletas:*** Son las herramientas que se utiliza para realizar la programación gráfica en el diagrama de bloques y también para modificar la apariencia de la interfaz gráfica que se muestra en el panel frontal. A continuación en la figura 2.80 se observa un ejemplo de paletas de Labview.

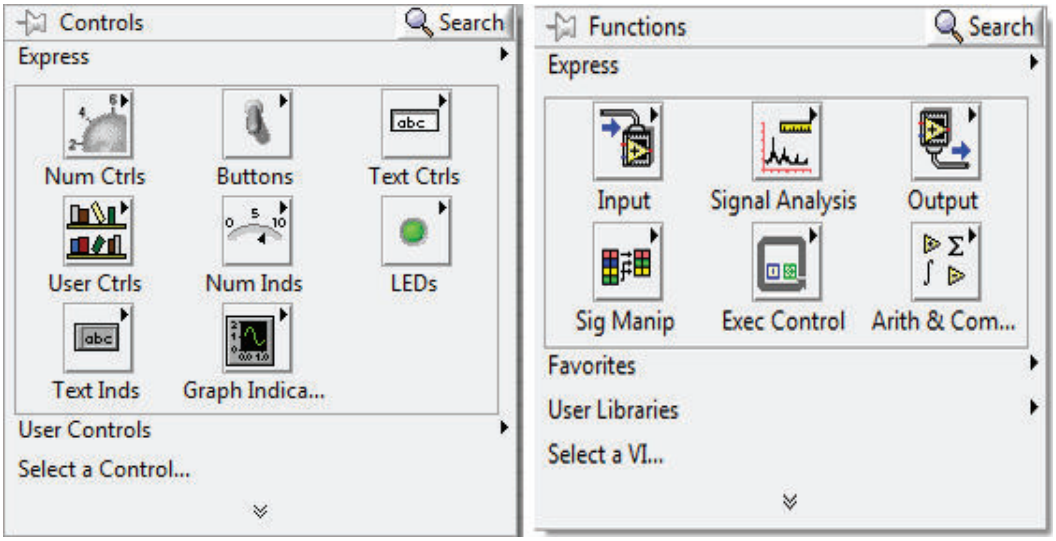


Figura 2.80: Paletas de LabVIEW

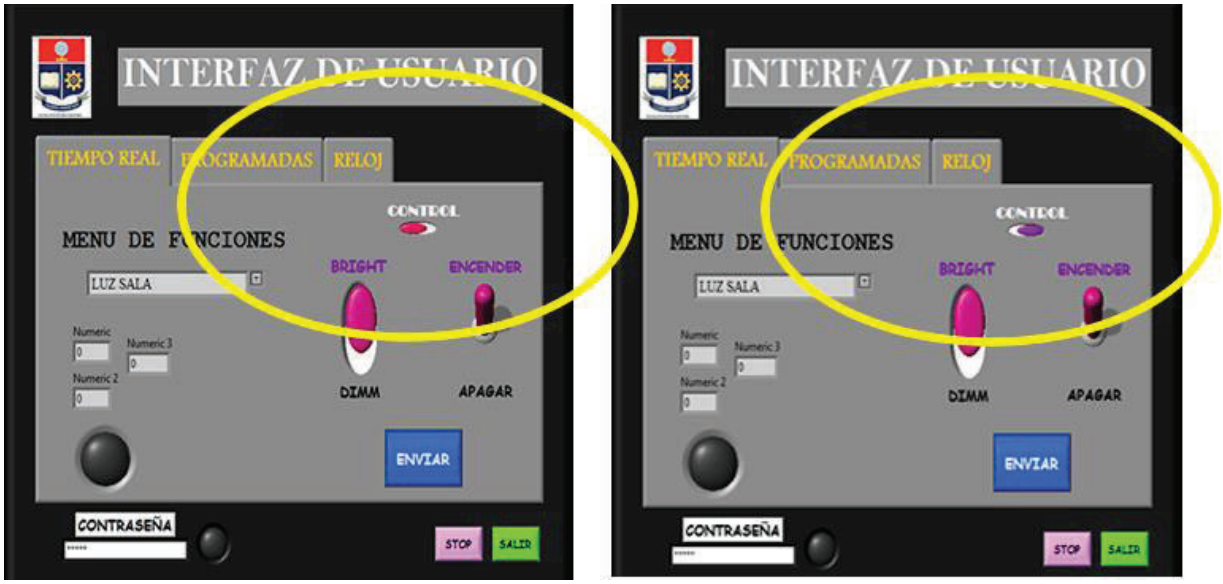
2.4.1.2 Desarrollo de la interfaz gráfica de usuario en Labview

Se eligió LabVIEW para el desarrollo de la interfaz gráfica de usuario gracias a que presenta una gran facilidad de programación y manejo de diferentes interfaces de comunicaciones, especialmente la del puerto serial, que es el principal instrumento para la transmisión y recepción de la información de toda la casa.

La interfaz gráfica de usuario empieza mostrando las ventanas de las tres secciones que conforman el control del sistema domótico. Este diseño presenta un ambiente muy amigable para el usuario, puesto que le permite ingresar fácilmente a la sección que el requiera y ejecutar la función que desee. Cada sección está dividida de acuerdo las aplicaciones de nuestro sistema domótico, las cuales se describen a continuación.

2.4.1.2.1 Tiempo Real

En la sección de tiempo real, el usuario puede elegir la función que desea ejecutar en este instante. Para ello en primer lugar selecciona de un menú de funciones la luminaria, persiana o puerta que desea controlar. A continuación selecciona el tipo de función que desea ejecutar, es decir si desea controlar mediante las funciones bright/dimm a las luminarias o persinas coloca el botón en el lado izquierdo del selector (Ver figura 2.81 a)), caso contrario si desea controlar mediante funciones ON/OFF a las luminarias, persianas o puertas, coloca el botón en el lado derecho del selector (Ver figura 2.81 b)).



a) b)

Figura 2.81: Pantalla para seleccionar el tipo de función que se desea ejecutar

Si el usuario seleccionó las funciones bright/dimm, a continuación debe colocar el botón del selector en la función deseada (BRIGHT o DIMM), caso contrario si el usuario seleccionó las funciones ON/OFF, debe colocar el toggle switch en encender ó apagar dependiendo de lo que desee realizar el usuario. Finalmente para enviar la instrucción debe presionar el botón enviar.

El menú de funciones de la sección tiempo real nos permite controlar las siguientes unidades:

- Luminaria de la sala
- Luminaria de la habitación 1
- Luminaria de la habitación 2
- Luminaria de la cocina
- Luminaria del baño
- Luminaria del ático
- Luminaria de la entrada
- Persiana de la sala
- Persiana de la habitación 1
- Persiana de la habitación 2
- Puerta de la entrada
- Luminaria/Puerta del garaje

Se debe tomar en cuenta que existe una sección para ingresar la contraseña de usuario, de tal manera que solo el dueño de la casa pueda controlar su sistema domótico. La contraseña actual es "*maria",* si se la ingresa correctamente el led de contraseña se enciende, caso contrario permanece apagado.

A continuación en la figura 2.82, se observa la interfaz gráfica de tiempo real, que presenta el menú de funciones, el selector para el tipo de control, el selector para las funciones Bright/Dimm, el toggle switch para las funciones Encender/Apagar, el botón de enviar y la sección de contraseña.

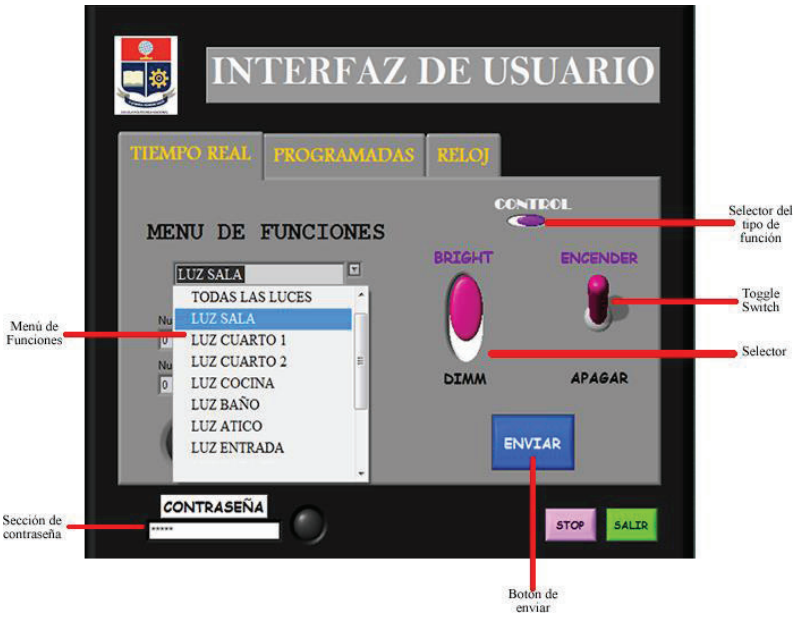


Figura 2.82: Ventana de control en Tiempo Real

2.4.1.2.2 Programadas

En la sección programadas el usuario puede elegir la función que debe ser ejecutada a la hora y minuto programados por él. Para ello selecciona mediante un visto las luces y persianas que desea controlar, a continuación selecciona la hora y minuto a los cuales desea encender las unidades, luego seleccina la hora y minuto a los cuales desea que se apaguen dichas unidades y finalmente presiona enviar.

Las unidades que el usuario puede controlar de manera programada son las siguientes:

- Luminaria de la sala
- Luminaria de la habitación 1
- Luminaria de la habitación 2
- Luminaria de la cocina
- Luminaria del baño
- Luminaria del ático
- Luminaria de la entrada

- Persiana de la sala
- Persiana de la habitación 1
- Persiana de la habitación 2

El formato que se utiliza para la visualización del tiempo es de 24 horas, es decir que el usuario puede programar una función desde las 0H00 hasta las 23H59.

Las funciones programadas se transmiten al FPGA apenas el usuario presiona el botón enviar. Una vez que llegan al FPGA, se guardan en los registros correspondientes hasta que sea la hora que el usuario ha programado para que se ejecuten las funciones que el desea en su hogar.

A continuación en la figura 2.83 se observa la interfaz gráfica de programadas, la cual presenta las luces y persianas que se desean controlar, la hora y minutos para la programación del control de encendido, la hora y minutos para la programación del control de apagado y el botón de enviar.

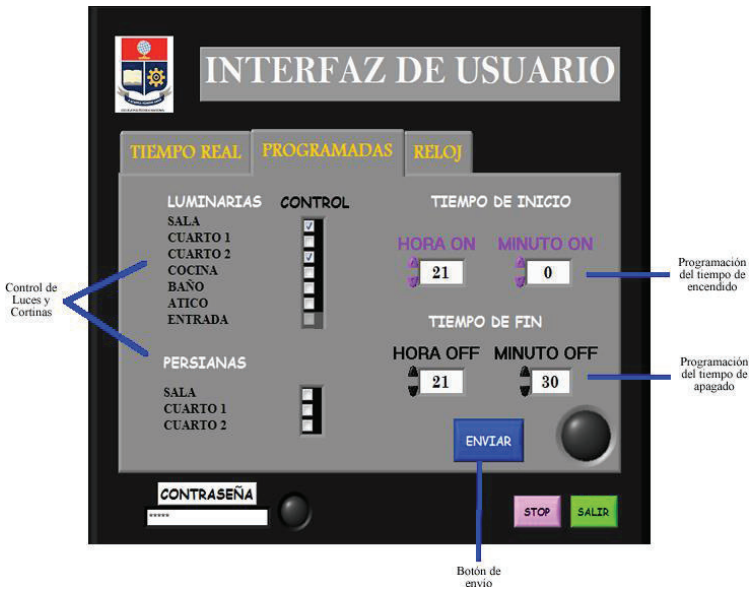


Figura 2.83: Ventana de control programado

2.4.1.2.3 Reloj

En la sección reloj el usuario puede observar el tiempo actual del sistema domótico en su ciudad. Además en caso de que la hora o minuto no sean los correctos, existe la opción de igualar su reloj, para ello selecciona la hora y minuto actual y presiona el botón actualizar, automáticamente observará a continuación el reloj actualizado.

Cuando el usuario realiza el control programado de su sistema domótico es importante que el reloj del sistema sea el correspondiente a su ciudad, por lo tanto el reloj no solo le permite al usuario observar la hora, sino también mantener el correcto funcionamiento de todo su sistema para ejecutar las funciones en tiempo real y programadas.

A continuación en la figura 2.84 se observa la interfaz gráfica del reloj, la cual presenta la hora y minuto actuales del sistema, la sección para configurar la hora acutal, la sección para configurar el minuto actual y el botón de actualizar.

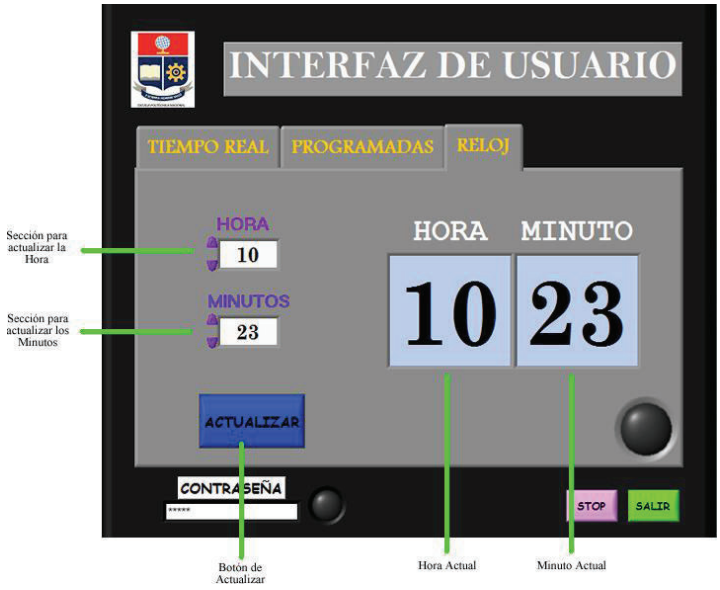


Figura 2.84: Ventana del control del Reloj

De esta manera el usuario de la vivienda utiliza la interfaz gráfica para controlar su sistema domótico. Cabe mencionar que en la actualidad existen diversas aplicaciones que nos permiten comunicarnos remotamente con nuestro computador. Aprovechando estas facilidades, el usuario puede usar una de estas aplicaciones para comunicarse con la interfaz gráfica desde una ubicación externa a su hogar, permitiéndole controlar su vivienda a cualquier hora y desde cualquier lugar.

Es importante recalcar que la programación desarrollada para la interfaz gráfica de usuario se encuentra detallada en el Anexo 21, por lo cual se recomienda revisar este anexo con la finalidad comprender el código fuente de la interfaz gráfica de usuario.

A continuación, en la última sección del capítulo se presenta la forma cómo se van a ubicar todos los circuitos anteriormente diseñados, para realizar la implementación el sistema domótico.

2.5 CONSTRUCCIÓN DEL PROTOTIPO

Para construir el prototipo se debe tomar en cuenta que cada una de las partes anteriormente descritas deben trabajar de manera eficiente en conjunto, evitando de esta forma que existan fallas en el sistema final que el usuario va a adquirir.

En la figura 2.85 que se presenta a continuación, se observa el prototipo construido, el cual consiste en una casa de madera a escala dentro de la cual se encuentran instalados todos los elementos que se van a controlar dentro del prototipo de vivienda.

Figura 2.85: Prototipo del sistema domótico

Para alimentar a todos los circuitos de la vivienda a escala, se utiliza una toma la cual permite realizar la instalación eléctrica dentro del prototipo.

Una vez que se ha descrito el diseño tanto del hardware como de los programas, para poder realizar la implementación del prototipo se procedió a imprimir todos los circuitos diseñados en fibra de vidrio, para posteriormente soldar los elementos requeridos en cada circuito. Cabe resaltar que los circuitos fueron diseñados y simulados en Proteus y adicionalmente en el Anexo 22 se encuentran todos los PCB (Printed Circuit Board) de los circuitos que permitieron el desarrollo de este proyecto.

Para realizar la instalación eléctrica dentro del prototipo se utilizó cable de cobre 12 AWG, el cual al tratarse de un prototipo pequeño de bajo consumo de corriente y potencia, permite el flujo de corriente eléctrica por todo el sistema sin inconvenientes. A continuación en la figura 2.86 se muestra las instalaciones eléctricas dentro de todo el prototipo, el cual recorre el techo de cada piso hasta llegar a la parte posterior de la vivienda, en donde culmina toda la instalación eléctrica de la vivienda.

Figura 2.86: Instalaciones eléctricas internas del sistema domótico

Para poder reconocer a qué elemento corresponde cada cable de la instalación eléctrica se procedió a etiquetar cada salida en la parte posterior de la casa (figura 2.87), de tal forma que sea fácil reconocer la conexión que debe ser implementada para la automatización de una vivienda. Esta es una buena práctica tanto para el prototipo como para una vivienda real, puesto que al etiquetar de forma correcta, se mantiene el orden y se ahorra el tiempo de verificación e instalación de todas las conexiones dentro de una vivienda.

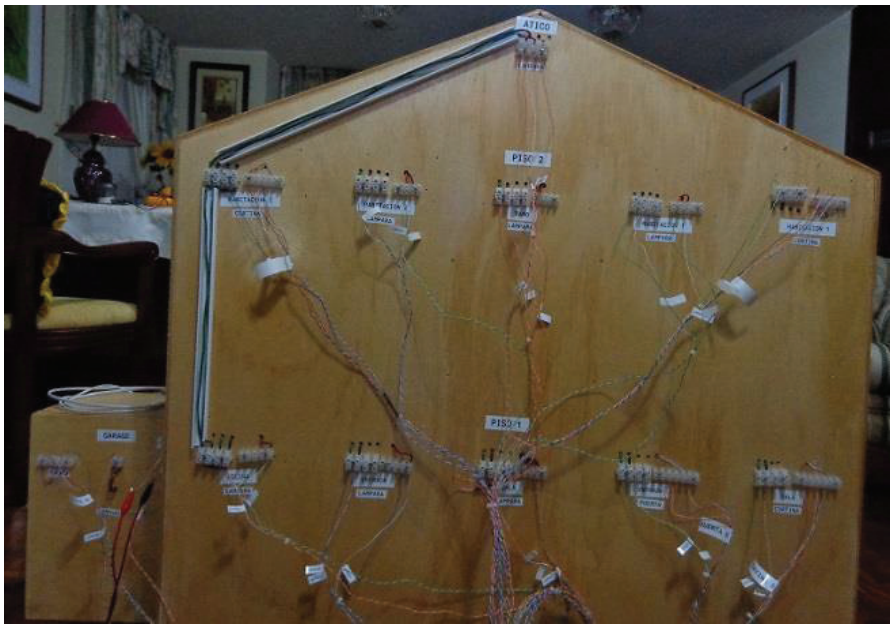


Figura 2.87: Etiquetado del prototipo

Posteriormente, una vez que se encuentran realizadas las instalaciones dentro de la vivienda, se procede a automatizar cada dispositivo instalado. Para ello se conecta la salida de cada elemento domótico de la vivienda con su respectivo actuador (figura 2.88). Cabe resaltar que cada actuador está conectado a su respectivo circuito receptor, los cuales son controlados por el FPGA, quién mediante protocolo X10 permite la ejecución de las órdenes requeridas por el usuario.

Figura 2.88: Conexión de los actuadores a la instalación eléctrica

Como se observa en la figura 2.88 los circuitos transmisor, receptores y actuadores también se encuentran etiquetados con el nombre del respectivo elemento domótico que controlan dentro del prototipo implementado, de tal manera que se pueda conservar el orden en el sistema.

Una vez realizada la conexión entre la instalación interna de la vivienda y sus respectivos circuitos de control, se procede a conectar el enchufe que permite energizar a todo el sistema (figura 2.89), de tal forma que se puedan efectuar las pruebas de funcionamiento que se describen en el siguiente capítulo.

Figura 2.89: Conexión completa del sistema domótico

CAPÍTULO 3

PRUEBAS DE FUNCIONAMIENTO Y RESULTADOS

En este capítulo se muestran y describen todas las pruebas realizadas en el prototipo de sistema domótico implementado, de tal manera que se pueda comprobar el correcto funcionamiento parcial y total del sistema.

Este capítulo es importante ya que nos permite verificar que todos los requerimientos del usuario del sistema domótico son cubiertos de manera satisfactoria, y de esta forma el óptimo funcionamiento del protocolo X10 para el control de una vivienda.

3.1 PRUEBAS REALIZADAS A LA SEÑAL X10

Antes de proceder a realizar las pruebas del sistema domótico, es importante conocer el comportamiento que tienen las señales desde que se generan en la interfaz de usuario, hasta que llegan a su destino, ya sea el actuador de iluminación, accesos o control de persianas.

Es por ello que a continuación se muestran los resultados observados en el osciloscopio a la salida de cada elemento por el que se transmiten las señales X10.

3.1.1 GENERACIÓN DE LA SEÑAL DE CONTROL EN EL FPGA

En la figura 3.1 se observa la señal de control que se genera en el FPGA y que sale del puerto serial para dirigirse hacia el transmisor X10 para que pueda llegar a su destino. En este caso la señal de control que se genera corresponde al requerimiento para encender la luminaria de la sala, es decir que se envían mediante comunicación serial tres tramas de ocho bits cada una, cuyo código en formato X10 es 01101001 01101001 01011001, el cual corresponde a los códigos de lugar, unidad y función respectivamente.

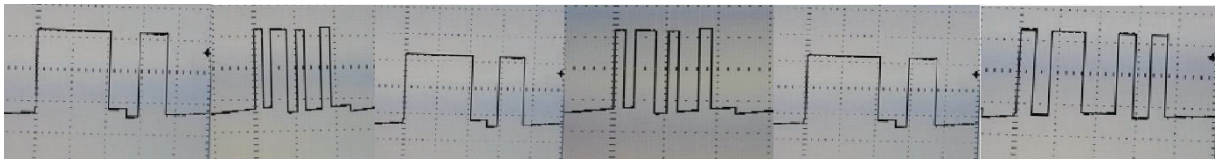


Figura 3.1: Señal de control generada en el FPGA para encender la luminaria de la sala

3.1.2 TRANSMISIÓN DE LA SEÑAL X10 HACIA LA RED ELÉCTRICA

En la figura 3.2 se observa la señal X10 que sale del transmisor X10, la cual se coloca en la red eléctrica para que sea recibida por el dispositivo que el usuario desea controlar. El código X10 que se transmite permite al usuario encender la luz de la sala, y es 01101001 01101001 01011001, el cual representa a los códigos de lugar, unidad y función respectivamente.

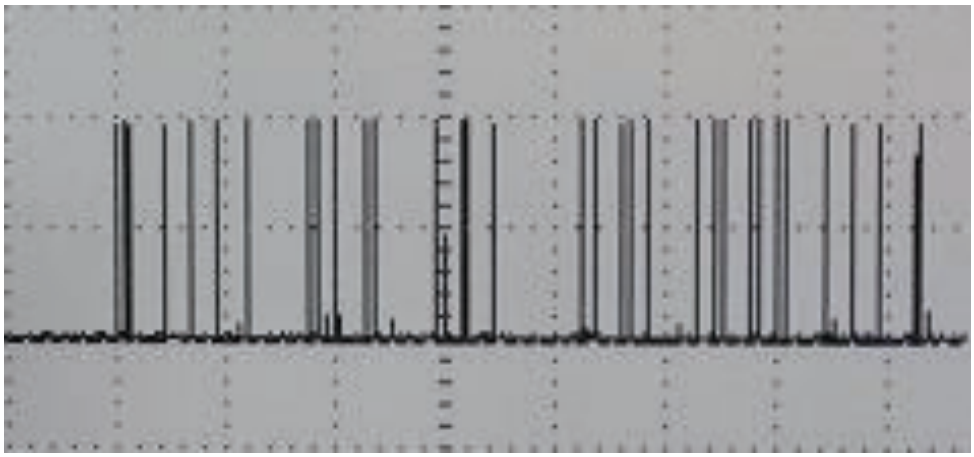


Figura 3.2: Señal X10 colocada en la red eléctrica por parte del transmisor X10

Como se puede evidenciar en la figura 3.2, la señal X10 que se observa tiene el formato de la trama X10, puesto que el transmisor X10 toma los códigos enviados serialmente por el FPGA y los transmite a la red eléctrica con el formato de una trama X10, de tal manera que pueda viajar por toda la instalación eléctrica de la vivienda y pueda ser reconocida por el elemento que el usuario desea controlar.

3.1.3 RECEPCIÓN DE LA SEÑAL X10 DESDE LA RED ELÉCTRICA

En la figura 3.3 se observa la señal X10 que llega al receptor después de haber viajado por la red eléctrica. Se debe tomar en cuenta que cuando la señal X10 viaja por la red eléctrica, va a llegar a todos los receptores X10, sin embargo sólo aquel que reconozca el código que está recibiendo va a enviar el comando de función hacia el actuador correspondiente.

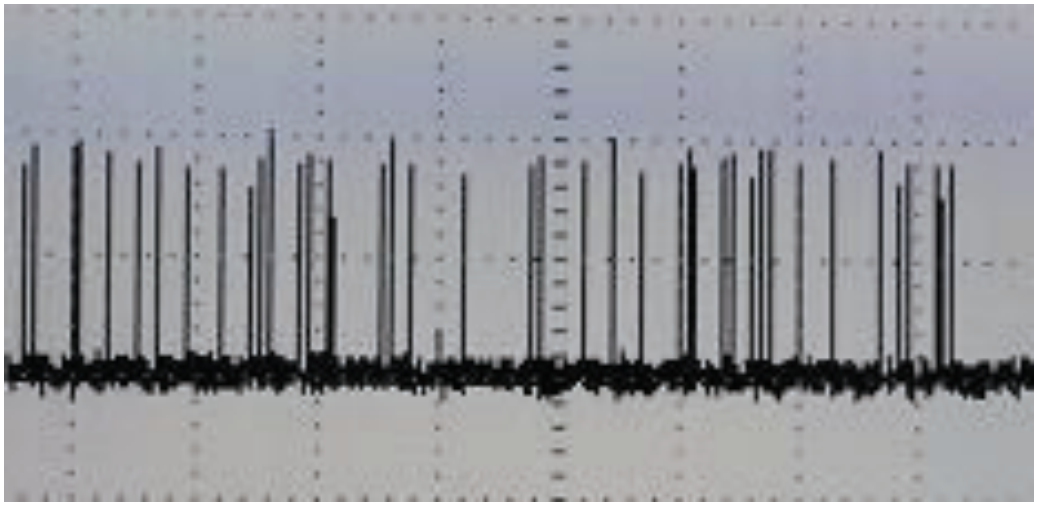


Figura 3.3: Señal X10 recibida desde la red eléctrica por parte del receptor X10

En base a la figura 3.3, se puede evidenciar que la señal que llega al receptor X10 se encuentra sumamente atenuada y no permite la identificación del código que representa, por lo cual debe pasar por los filtros pasa altos, sintonizadores y amplificadores correspondientes al circuito de recepción, para que la señal nuevamente recupere sus características iniciales e ingrese al microcontrolador de recepción X10, tal y como se muestra en la siguiente figura 3.4, en donde se puede observar claramente que el código X10 que se recibió es 01101001 01101001 01011001, el cual corresponde al código para encender la luz de la sala.

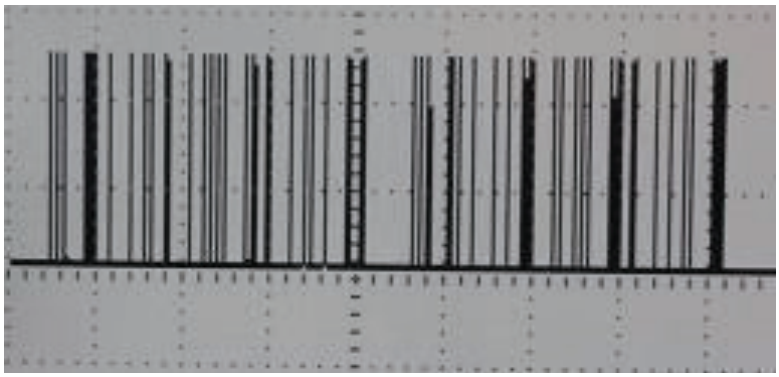


Figura 3.4: Señal X10 regenerada en el circuito receptor X10

3.1.4 TRANSMISIÓN DEL CÓDIGO DE FUNCIÓN X10 HACIA EL ACTUADOR

Tal y como se pudo observar en la figura 3.4, el código X10 que llegó al receptor es 01101001 01101001 01011001, éste código X10 es comparado con los códigos de casa y unidad que se encuentran almacenados en el receptor y si son iguales, el receptor X10 procede a enviar hacia el actuador el código de función recibido.

A continuación en la figura 3.5 se observa la imagen del código de función que el receptor X10 trasmite hacia el actuador de luz, que en este caso es el siguiente 01011001, de tal manera que al llegar al actuador, éste encenderá la luminaria de la sala.

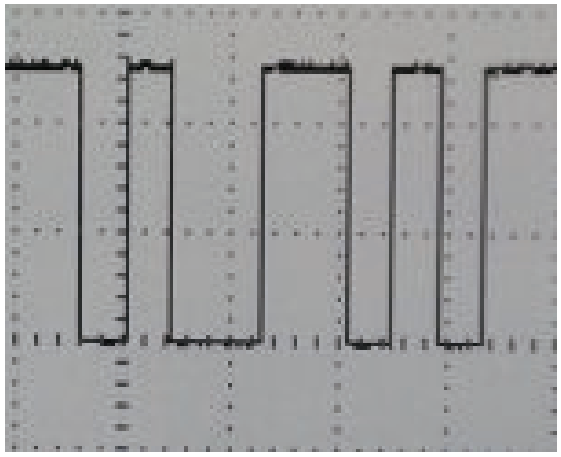


Figura 3.5: Código de función X10 transmitido hacia el actuador de luz de la sala

El diseño e implementación de este prototipo nos permitió llevar a cabo el control total del sistema domótico, de tal manera que en el FPGA se generan y transmiten las señales de control hasta que llegan al respectivo actuador, el cual ejecuta el control requerido por el usuario, tal y como se pudo observar en las figuras anteriormente mostradas, que nos indican todas las etapas que deben atravesar los códigos X10 hasta llegar finalmente al actuador en donde se ejecuta la función deseada.

A continuación se describen las pruebas y resultados obtenidos al indicar al sistema domótico que ejecute el control de todas las funciones que éste puede realizar, tanto de manera individual como de forma integrada.

3.2 PRUEBA INTEGRAL DEL SISTEMA DOMÓTICO EN TIEMPO REAL

Para poder verificar que el usuario no tenga ningún problema con alguno de los dispositivos automatizados de su hogar, se realizaron las pruebas individuales de cada subsistema de la vivienda, y las pruebas totales de todo el sistema implementado.

El procedimiento que se sigue para comprobar el correcto funcionamiento del sistema en tiempo real consiste en usar la interfaz gráfica para someter al sistema a las siguientes pruebas:

1. Encender las luminarias de la vivienda.
2. Apagar las luminarias de la vivienda.
3. Incrementar y disminuir el nivel de iluminación de la vivienda
4. Abrir cada una de las persianas de la vivienda.
5. Cerrar cada una de las persianas de la vivienda.
6. Apertura y cerrado a diferentes niveles de las persianas de la vivienda
7. Abrir las puertas de la vivienda.
8. Cerrar las puertas de la vivienda.
9. Realizar el control manual de iluminación mediante el uso de la perilla.

Todas estas pruebas se realizaron en tiempo real, de tal manera que podamos evidenciar el adecuado funcionamiento del sistema.

3.2.1 PRUEBAS DEL CONTROL DE ILUMINACIÓN

Estas pruebas están orientadas a comprobar el encendido, apagado, y dimerización de todas las luminarias del sistema domótico. A continuación se procede a describir el procedimiento que se sigue para realizar la verificación del funcionamiento del control de iluminación, y posteriormente los resultados después de cada procedimiento.

3.2.1.1 Encendido de Luminarias

A continuación se procede a explicar el procedimiento y los resultados obtenidos cuando el usuario desea encender una determinada luminaria.

3.2.1.1.1 Procedimiento para comprobar el encendido de luminarias

Este procedimiento consiste en que el usuario cuando desea encender cualquier luminaria de su vivienda, por medio del uso de la interfaz de usuario, indica al sistema domótico encender determinada luminaria.

En primer lugar se debe abrir la aplicación y automáticamente se despliega el *“Menú de Funciones”,* el cual contiene la lista de dispositivos que se puede controlar. Si se desea encender la luminaria de la sala, se selecciona *Luz Sala*, luego se coloca el botón del selector en el lado derecho, a continuación coloca el toggle switch en la posición ON y finalmente presiona ENVIAR. De esta manera el usuario puede encender la luminaria de la sala tal y como se muestra en la figura 3.6.

Figura 3.6: Uso de la interfaz gráfica de usuario para encender la luminaria

3.2.1.1.2 Resultados de las pruebas para el encendido de luces

A continuación en la tabla 3.1 se muestran los resultados obtenidos al indicar al sistema el encendido de la luminarias de cada uno de los lugares de la vivienda usando la interfaz gráfica de usuario, con el fin de comprobar que esta interfaz permite el encendido de cualquier luminaria de la vivienda.

Tabla 3.1: Resultados de las pruebas para el encendido de luminarias

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA
		INTERFAZ GRÁFICA DE USUARIO
Sala	ON	Exitoso
Cocina	ON	Exitoso
Habitación 1	ON	Exitoso
Habitación 2	ON	Exitoso
Baño	ON	Exitoso
Ático	ON	Exitoso
Entrada	ON	Exitoso
Garaje	ON	Exitoso

Como se puede evidenciar en la tabla 3.1, el encendido de cada una de las luminarias del sistema domótico fue exitoso, lo que nos permite asegurar que el usuario no tendrá ningún problema encendiendo cualquier luminaria de su vivienda.

A continuación en la figura 3.7, se muestra una fotografía de las luminarias de la vivienda encendidas por requerimiento del usuario.

Figura 3.7: Luminarias de la vivienda encendidas

3.2.1.2 Apagado de Luminarias

A continuación se procede a explicar el procedimiento y los resultados obtenidos cuando el usuario desea apagar una determinada luminaria.

3.2.1.2.1 Procedimiento para comprobar el apagado de luces

Este procedimiento consiste en que el usuario cuando desea apagar una luminaria de cualquier lugar de su vivienda, por medio del uso de la interfaz de usuario, indica al sistema domótico apagar determinada luminaria.

En primer lugar se debe abrir la aplicación y automáticamente se despliega el *"Menú de Funciones",* el cual contiene la lista de dispositivos que se puede controlar. En este caso si se desea apagar la luminaria de la sala, se selecciona *Luz Sala*, luego se coloca el botón del selector en el lado derecho, a continuación coloca el toggle switch en la posición OFF y finalmente presiona ENVIAR. De esta manera el usuario puede apagar la luminaria de la sala tal y como se muestra en la figura 3.8.

Figura 3.8: Uso de la interfaz gráfica de usuario para apagar una luminaria

3.2.1.2.2 Resultados de las pruebas para el apagado de luces

A continuación en la tabla 3.2, se muestran los resultados obtenidos al indicar al sistema el apagado de la luminarias que fueron encendidas en cada uno de los lugares de la vivienda, usando para ello la interfaz gráfica de usuario, con el fin de comprobar que dicha interfaz permite apagar cualquier luminaria de la vivienda.

Tabla 3.2: Resultados de las pruebas para apagar las luminarias

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA
		INTERFAZ GRÁFICA DE USUARIO
Sala	OFF	Exitoso
Cocina	OFF	Exitoso
Habitación 1	OFF	Exitoso
Habitación 2	OFF	Exitoso
Baño	OFF	Exitoso
Ático	OFF	Exitoso
Entrada	OFF	Exitoso
Garaje	OFF	Exitoso

Como se puede evidenciar en la tabla 3.2, el apagado de cada una de las luminarias del sistema domótico fue exitoso, lo que nos permite asegurar que el usuario no tendrá ningún problema apagando cualquier luminaria de su vivienda.

A continuación en la figura 3.9 se muestra una fotografía de las luminarias de la vivienda apagadas por requerimiento del usuario.

Figura 3.9 Luminarias de la vivienda apagadas

3.2.1.3 Dimerización de Luminarias

A continuación se procede a explicar el procedimiento y los resultados obtenidos cuando el usuario desea dimerizar una determinada luminaria. Se debe tomar en cuenta que el proceso de dimerización consiste en aumentar o disminuir la intensidad de la luz de una determinada luminaria, para lo cual se realizaron pruebas para los dos procesos.

3.2.1.3.1 Procedimiento para comprobar el aumento de la intensidad de las luces

Este procedimiento consiste en que el usuario al ingresar a cualquier lugar de su vivienda, por medio del uso de la interfaz de usuario o de forma manual, indica al sistema domótico aumentar la intensidad de luz de una determinada luminaria.

Si el usuario desea utilizar la interfaz gráfica de usuario, debe seleccionar en la aplicación la luminaria que desea controlar, en este caso se selecciona “Luz Sala”, luego se coloca el botón del selector de control en el lado izquierdo, a continuación se coloca el respectivo selector en la posición BRIGHT y finalmente se presiona la tecla ENVIAR. De esta manera el usuario puede incrementar la intensidad de luz del dispositivo, tal y como se muestra en la figura 3.10, donde se observa el procedimiento que sigue el usuario para aumentar la intensidad de la luminaria de la sala usando la interfaz gráfica de usuario.

Es importante resaltar que el actuador de iluminación fue programado de tal forma que al estar completamente apagada una luminaria, el máximo nivel de luz se consigue al presionar ocho veces la tecla ENVIAR, es decir que el máximo nivel de intensidad de luz de una luminaria está programado en el nivel 8.

Figura 3.10: Uso de la interfaz gráfica de usuario para aumentar la intensidad de luz de una luminaria

Si el usuario desea aumentar la intensidad de luz de manera manual, debe presionar el switch de control manual y a continuación utilizar la perilla, tanto el switch como la perilla se encuentran acoplados en el interruptor de pared, de tal forma que girando la perilla en sentido horario se obtiene un aumento de la intensidad de luz hasta el nivel que el usuario desee, tal y como se muestra en la figura 3.11, en donde se observa cómo al girar la perilla la intensidad de luz va aumentado hasta que la perilla llega a 360°, permitiéndole al foco encenderse completamente.

Figura 3.11: Control manual del aumento de la intensidad de luz de una luminaria

3.2.1.3.2 Procedimiento para comprobar la disminución de la intensidad de las luces

Este procedimiento consiste en disminuir la intensidad de luz en cualquier lugar de la vivienda, cuyas luminarias se encuentren encendidas, por medio del uso de la interfaz gráfica de usuario o de manera manual.

Si el usuario desea utilizar la interfaz gráfica de usuario, debe seleccionar en la aplicación la luminaria que desea controlar, en este caso se selecciona “Luz Sala”, luego se coloca el botón del selector de control en el lado izquierdo, a continuación se coloca el respectivo selector en la posición DIMM y finalmente se presiona la tecla ENVIAR.. De esta manera el usuario puede disminuir la intensidad de luz del dispositivo, tal y como se muestra en la figura 3.12, donde se observa el procedimiento que sigue el usuario para disminuir la intensidad de la luminaria de la sala usando la interfaz gráfica de usuario.

Es importante resaltar que el actuador de iluminación fue programado de tal forma que al estar completamente encendida una luminaria, el mínimo nivel de luz se consigue al presionar ocho veces la tecla ENVIAR, es decir que el mínimo nivel de intensidad de luz de una luminaria está programado en el nivel 1.

Figura 3.12: Uso de la interfaz gráfica de usuario para disminuir la intensidad de luz de un foco

Si el usuario desea disminuir la intensidad de luz de manera manual, debe presionar el switch de control manual y a continuación utilizar la perilla de control manual, ambos se encuentran acoplados en el interruptor de pared, de tal forma que girando la perilla en sentido anti-horario, va a disminuir la intensidad hasta el nivel que el usuario desee, tal y como se muestra en la figura 3.13, en donde se observa cómo se gira la perilla y la intensidad de luz va disminuyendo hasta que la perilla gira 360° permitiéndole a la luminaria apagarse completamente.

Figura 3.13: Control manual para la disminución de la intensidad de luz de una luminaria

3.2.1.3.3 Resultados de la pruebas para el aumento y disminución de la intensidad de luz de las luminarias.

A continuación en la tabla 3.3, se muestran los resultados obtenidos al indicar al sistema el aumento y disminución de la intensidad de luz de la luminarias internas de la vivienda, usando para ello tanto la interfaz gráfica como el control manual de iluminación, con el fin de comprobar que las dos interfaces permitan ejecutar la dimerización de las luminarias internas de la vivienda.

Tabla 3.3: Resultados de las pruebas para cambiar la intesidad de luz en las luminarias

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA	
		INTERFAZ GRÁFICA DE USUARIO	CONTROL MANUAL
Sala	BRIGHT	Exitoso	Exitoso
	DIMM	Exitoso	Exitoso
Cocina	BRIGHT	Exitoso	Exitoso
	DIMM	Exitoso	Exitoso
Habitación 1	BRIGHT	Exitoso	Exitoso
	DIMM	Exitoso	Exitoso
Habitación 2	BRIGHT	Exitoso	Exitoso
	DIMM	Exitoso	Exitoso
Baño	BRIGHT	Exitoso	Exitoso
	DIMM	Exitoso	Exitoso
Ático	BRIGHT	Exitoso	Exitoso
	DIMM	Exitoso	Exitoso

Como se puede evidenciar en la tabla 3.3, el aumento y disminución de la intensidad de luz en cada una de las luminarias del sistema domótico fue exitoso, lo que nos permite asegurar que el usuario no tendrá ningún problema ejecutando estas funciones en cualquier luminaria interna de su vivienda.

A continuación en la figura 3.14 se muestran las luminarias de la vivienda, con diferentes niveles de intensidad de luz der acuerdo al requerimiento del usuario.

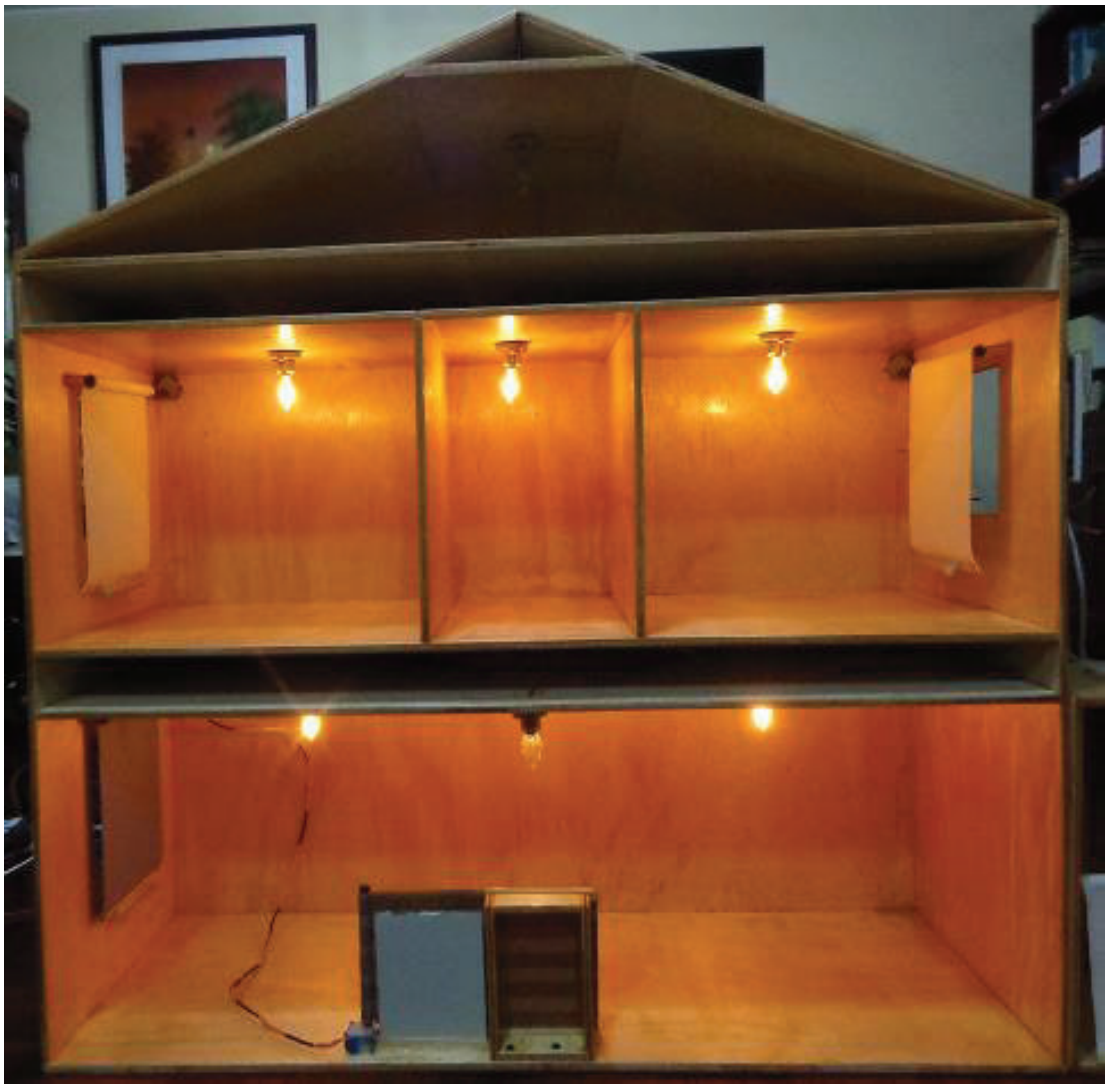


Figura 3.14: Luminarias de la vivienda con diferentes niveles de intensidad de luz

Finalmente, cabe recalcar que el resultado en todos los casos fue exitoso, usando tanto la interfaz gráfica así como el control manual. Por lo cual se garantiza el uso de cualquier interfaz que el usuario desee para controlar la iluminación de su vivienda.

3.2.2 PRUEBAS DEL CONTROL DE PERSIANAS

Estas pruebas están orientadas a comprobar que todas las persianas del sistema domótico se abren y cierran total y parcialmente, acorde a los requerimientos del usuario. Cabe resaltar que en el sistema domótico se implementó este control en tres

lugares considerados como principales para ejecutarlos, es decir en la sala, la habitación 1 y la habitación 2.

A continuación se procede a describir el procedimiento que se sigue para realizar la verificación del funcionamiento del control de persianas, y posteriormente los resultados después de cada procedimiento.

3.2.2.1 Abrir totalmente las persianas

A continuación se procede a explicar el procedimiento y los resultados obtenidos cuando el usuario desea abrir totalmente una determinada persiana.

3.2.2.1.1 Procedimiento para comprobar que se abren totalmente las persianas

Este procedimiento consiste en que el usuario al ingresar a la sala o a alguna de las habitaciones, por medio del uso de la interfaz gráfica de usuario, indica al sistema domótico abrir totalmente una determinada persiana.

En primer lugar se debe abrir la aplicación y automáticamente se despliega el *"Menú de Funciones",* el cual contiene la lista de dispositivos que se puede controlar. Si se desea abrir la persiana de la sala, se selecciona *Persiana Sala*, luego se coloca el botón del selector en el lado derecho, a continuación coloca el toggle switch en la posición ON y finalmente presiona ENVIAR. De esta manera el usuario puede abrir la persiana de la sala tal y como se muestra en la figura 3.15.

Figura 3.15: Uso de la interfaz gráfica de usuario para abrir una persiana

3.2.2.1.2 Resultados de las pruebas para abrir totalmente las persianas

A continuación en la tabla 3.4, se muestran los resultados obtenidos al indicar al sistema domótico que se abran totalmente las persianas de la vivienda, usando la interfaz gráfica de usuario, con el fin de comprobar que dicha interfaz permitan abrir las persianas del sistema.

Tabla 3.4: Resultados de las pruebas para abrir totalmente las persianas

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA
		INTERFAZ GRÁFICA DE USUARIO
Sala	ON	Exitoso
Habitación 1	ON	Exitoso
Habitación 2	ON	Exitoso

Como se puede evidenciar en la tabla 3.4, los resultados de las pruebas para abrir totalmente las persianas del sistema domótico fueron exitosos, lo que permite asegurar que el usuario no tendrá ningún problema abriendo cualquier persiana de su vivienda.

A continuación en la figura 3.16 se muestran las persianas de la vivienda abiertas totalmente por requerimiento del usuario.

Figura 3.16: Persianas de la vivienda totalmente abiertas

3.2.2.2 Cerrar totalmente las persianas

A continuación se indica el procedimiento y los resultados obtenidos cuando el usuario desea cerrar totalmente una determinada persiana.

3.2.2.2.1 Procedimiento para comprobar que se cierran totalmente las persianas

Este procedimiento consiste en que el usuario al ingresar a la sala o a alguna de las habitaciones, cuyas persianas se encuentran abiertas, por medio del uso de la interfaz gráfica de usuario, indica al sistema domótico cerrar totalmente determinada persiana.

En primer lugar se debe abrir la aplicación y automáticamente se despliega el *"Menú de Funciones",* el cual contiene la lista de dispositivos que se puede controlar. Si se desea cerrar la persiana de la sala, se selecciona *Persiana Sala*, luego se coloca el botón del selector en el lado derecho, a continuación coloca el toggle switch en la posición OFF y finalmente presiona ENVIAR. De esta manera el usuario puede cerrar persiana de la sala tal y como se muestra en la figura 3.17.

Figura 3.17: Uso de la interfaz gráfica de usuario para cerrar totalmente una persiana

3.2.2.2.2 Resultados de las pruebas para cerrar totalmente las persianas

A continuación en la tabla 3.5, se muestran los resultados obtenidos al indicar al sistema domótico que se cierren totalmente las persianas de la vivienda, usando la interfaz gráfica de usuario, con el fin de comprobar que dicha interfaz permite cerrar las persianas del sistema.

Tabla 3.5: Resultados de las pruebas para cerrar totalmente las persianas

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA
		INTERFAZ GRÁFICA DE USUARIO
Sala	OFF	Exitoso
Habitación 1	OFF	Exitoso
Habitación 2	OFF	Exitoso

Como se puede evidenciar en la tabla 3.5, los resultados de las pruebas para cerrar totalmente las persianas del sistema domótico fueron exitosos, lo que permite asegurar que el usuario no tendrá ningún problema cerrando cualquier persiana de su vivienda.

A continuación en la figura 3.18, se muestra una fotografía de las persianas de la vivienda cerradas totalmente por requerimiento del usuario.

Figura 3.18: Persianas de la vivienda totalmente cerradas

3.2.2.3 Abrir parcialmente las persianas

A continuación se indica el procedimiento y los resultados obtenidos cuando el usuario desea abrir parcialmente una determinada persiana.

3.2.2.3.1 Procedimiento para comprobar que se abren parcialmente las persianas

Este procedimiento consiste en que el usuario al ingresar a cualquier lugar de su vivienda, cuyas persianas estén automatizadas, por medio del uso de la interfaz gráfica indica al sistema domótico abrir parcialmente dicha persiana.

Para este efecto se debe seleccionar en la aplicación la persiana que se desea controlar, en este caso se selecciona “Persiana Sala”, luego se coloca el botón del selector de control en el lado izquierdo, a continuación se coloca el respectivo selector en la posición BRIGHT y finalmente se presiona la tecla ENVIAR. De esta manera el usuario puede abrir parcialmente una persiana tal y como se muestra en la figura 3.19, donde se observa el procedimiento que sigue el usuario para abrir parcialmente la persiana de la sala usando la interfaz gráfica de usuario.

Es importante resaltar que el actuador para el control de persianas fue programado de tal forma que al estar completamente cerrada la persiana, el máximo nivel de apertura se consigue al presionar 4 veces la tecla ENVIAR, nivel donde se ha determinado mediante programación, que la persiana ya no podrá abrirse más.

Figura 3.19: Uso de la interfaz gráfica para abrir parcialmente una persiana

3.2.2.3.2 Resultados de la pruebas para abrir parcialmente las persianas

A continuación en la tabla 3.6, se muestran los resultados obtenidos al indicar al sistema domótico abrir parcialmente las persianas de la vivienda, usando la interfaz gráfica, con el fin de comprobar que el usuario no tendrá ningún problema abriendo parcialmente las persianas de su vivienda.

Tabla 3.6: Resultados de las pruebas para abrir parcialmente las persianas

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA USANDO LA INTERFAZ GRÁFICA DE USUARIO
Sala	BRIGHT	Exitoso
Habitación 1	BRIGHT	Exitoso
Habitación 2	BRIGHT	Exitoso

Como se puede evidenciar en la tabla 3.6, los resultados de las pruebas realizadas para abrir parcialmente las persianas del sistema domótico fueron exitosos, lo que permite asegurar que el usuario no tendrá ningún problema ejecutando esta función en cualquier persiana de su vivienda.

A continuación en la figura 3.20 se muestran las persianas de la vivienda abiertas a diferente nivel de acuerdo al requerimiento del usuario

Figura 3.20: Persianas de la vivienda parcialmente abiertas

3.2.2.4 Cerrar parcialmente las persianas

A continuación se indica el procedimiento y los resultados obtenidos cuando el usuario desea cerrar parcialmente una determinada persiana.

3.2.2.4.1 Procedimiento para comprobar que se cierran parcialmente las persianas

Este procedimiento consiste en que el usuario al ingresar a cualquier lugar de su vivienda, cuyas persianas estén automatizadas y abiertas, por medio del uso de la interfaz gráfica, indica al sistema domótico cerrar parcialmente dicha persiana.

Para este efecto se debe seleccionar en la aplicación la persiana que se desea controlar, en este caso se selecciona “Persiana Sala”, luego se coloca el botón del selector de control en el lado izquierdo, a continuación se coloca el respectivo selector en la posición DIMM y finalmente se presiona la tecla ENVIAR. De esta manera el usuario puede cerrar parcialmente una persiana tal y como se muestra en la figura 3.21, donde se observa el procedimiento que sigue el usuario para cerrar parcialmente la persiana de la sala usando la interfaz gráfica de usuario.

Es importante resaltar que el actuador para el control de persianas fue programado de tal forma que al estar completamente abierta la persiana, el máximo nivel de cerrado se consigue al presionar 4 veces la tecla ENVIAR, nivel donde se ha determinado ,mediante programación, que la persiana ya no podrá cerrarse más.

Figura 3.21: Uso de la interfaz gráfica para cerrar parcialmente una persiana

3.2.2.4.2 Resultados de la pruebas para cerrar parcialmente las persianas

A continuación en la tabla 3.7, se muestran los resultados obtenidos al indicar al sistema domótico cerrar parcialmente las persianas de la vivienda, usando la interfaz gráfica, con el fin de comprobar que el usuario no tendrá ningún problema cerrando parcialmente las persianas de su vivienda.

Tabla 3.7: Resultados de las pruebas para cerrar parcialmente las persianas

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA USANDO LA INTERFAZ GRÁFICA
Sala	DIM	Exitoso
Habitación 1	DIM	Exitoso
Habitación 2	DIM	Exitoso

Como se puede evidenciar en la tabla 3.7, los resultados de las pruebas realizadas para cerrar parcialmente las persianas del sistema domótico fueron exitosos, lo que permite asegurar que el usuario no tendrá ningún problema ejecutando esta función en cualquier persiana de su vivienda.

A continuación en la figura 3.22, se muestran las persianas de la vivienda cerradas a diferente nivel de acuerdo al requerimiento del usuario.

Figura 3.22: Persianas de la vivienda parcialmente cerradas

Finalmente, cabe recalcar que el resultado en todos los casos fue positivo. Por lo cual garantizamos el uso de la interfaz gráfica para controlar las persianas de su vivienda.

3.2.3 PRUEBAS DEL CONTROL DE ACCESOS

Estas pruebas están orientadas a comprobar que las puertas del prototipo de sistema domótico se abren y se cierran según los requerimientos del usuario. Cabe resaltar que en el prototipo de sistema domótico se implementó este control en dos

lugares considerados como principales para ejecutarlos, es decir en la entrada a la vivienda y en el garaje.

A continuación se indica el procedimiento que se sigue para realizar la verificación del funcionamiento del control de accesos, y posteriormente los resultados después de cada procedimiento.

3.2.3.1 Abrir las Puertas

El procedimiento y los resultados obtenidos cuando el usuario desea ingresar o salir de su vivienda, se indican a continuación.

3.2.3.1.1 Procedimiento para comprobar que se abren las puertas

Este procedimiento consiste en que el usuario para poder ingresar o salir de su vivienda debe abrir la puerta, para lo cual por medio del uso de la interfaz gráfica, indica al sistema domótico cumplir este requerimiento.

En primer lugar se debe abrir la aplicación y automáticamente se despliega el *"Menú de Funciones",* el cual contiene la lista de dispositivos que se puede controlar. Si se desea abrir la puerta de la entrada, selecciona *Puerta Entrada*, luego se coloca el botón del selector en el lado derecho, a continuación coloca el toggle switch en la posición ON y finalmente presiona ENVIAR. De esta manera el usuario puede abrir la puerta de la entrada tal y como se muestra en la figura 3.23.

Figura 3.23: Uso de la interfaz gráfica para abrir una puerta

3.2.3.1.2 Resultados de las pruebas para abrir las puertas

A continuación en la tabla 3.8, se muestran los resultados obtenidos al indicar al sistema domótico que se abran las puertas de la vivienda, usando para ello la interfaz gráfica, con el fin de comprobar que el usuario no tendrá ningún problema abriendo las puertas de su vivienda.

Tabla 3.8: Resultados de las pruebas para abrir las puertas

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA USANDO LA INTERFAZ GRÁFICA
Entrada	ON	Exitoso
Garaje	ON	Exitoso

Como se puedo evidenciar en la tabla 3.8, los resultados de las pruebas para abrir las puertas del sistema domótico fueron exitosos, lo que nos permite asegurar que el usuario no tendrá ningún problema abriendo las puertas de su vivienda.

A continuación en la figura 3.24 se muestran las puertas de la vivienda abiertas por requerimiento del usuario.

Figura 3.24: Puertas de la vivienda abiertas

3.2.3.2 Cerrar las Puertas

El procedimiento y los resultados obtenidos cuando el usuario ha ingresado o ha salido de su vivienda y desea cerrar las puertas, se indica a continuación.

3.2.3.2.1 Procedimiento para comprobar que se cierran las puertas

Este procedimiento consiste en que el usuario, una vez que ha ingresado o salido de su vivienda, debe cerrar la puerta, para lo cual por medio del uso de la interfaz gráfica, indica al sistema domótico cumplir este requerimiento.

En primer lugar se debe abrir la aplicación y automáticamente se despliega el *"Menú de Funciones",* el cual contiene la lista de dispositivos que se puede controlar. Si se desea cerrar la puerta de la entrada, selecciona *Puerta Entrada*, luego se coloca el botón del selector en el lado derecho, a continuación coloca el toggle switch

en la posición OFF y finalmente presiona ENVIAR. De esta manera el usuario puede abrir la puerta de la entrada tal y como se muestra en la figura 3.25.

Figura 3.25: Uso de la interfaz gráfica para cerrar una puerta

3.2.3.2.2 Resultados de las pruebas para cerrar las puertas

A continuación en la tabla 3.9, se muestran los resultados obtenidos al indicar al sistema domótico que se cierren las puertas de la vivienda, usando la interfaz gráfica, con el fin de comprobar que el usuario no tendrá ningún problema cerrando las puertas de su vivienda.

Tabla 3.9: Resultados de las pruebas para cerrar las puertas

LUGAR	FUNCIÓN	RESULTADO DE LA PRUEBA USANDO LA INTERFAZ GRÁFICA
Entrada	OFF	Exitoso
Garaje	OFF	Exitoso

Como se puede evidenciar en la tabla 3.9, los resultados de las pruebas para cerrar las puertas del sistema domótico fueron exitosos, lo que nos permite asegurar que el usuario no tendrá ningún problema cerrando las puertas de su vivienda.

A continuación en la figura 3.26, se muestran las puertas de la vivienda cerradas por requerimiento del usuario.

Figura 3.26: Puertas de la vivienda cerradas

Finalmente, cabe recalcar que el resultado en los dos casos fue positivo. Por lo cual se asegura el uso de la interfaz gráfica para mantener en orden la seguridad de la vivienda, permitiéndoles exclusivamente al usuario y a sus conocidos ingresar o salir de la misma.

Es importante destacar que el control de la puerta y luminaria del garaje se ejecutan simultáneamente, puesto que al ser un lugar oscuro requiere de iluminación cada vez que el usuario desee ingresar o salir con su vehículo. Adicionalmente permite demostrar que un circuito de recepción de las señales X10 puede controlar exitosamente dos actuadores diferentes dentro del prototipo implementado.

Una vez descritas individualmente las pruebas y resultados de cada uno de los elementos que conforman el sistema domótico, a continuación en la figura 3.27 se muestra el resultado de la prueba integral del prototipo de sistema domótico.

a) b)

Figura 3.27 Resultados de la prueba de funcionamiento del sistema domótico integrado en tiempo real

En la figura 3.27 a) se observa la vivienda con las luminarias encendidas y las persianas y puertas abiertas por requerimiento del usuario, mientras que en la figura 3.27 b) se observa la vivienda con las luminarias apagadas y las persianas y puertas cerradas por requerimiento del usuario. Todos estos resultados se ejecutan de manera inmediata al requerimiento del usuario.

Una vez realizada la prueba de funcionamiento de todo el prototipo de sistema domótico integrado en tiempo real, se puede evidenciar en la figura 3.27, que el sistema respondió positivamente a todas las peticiones del usuario, tanto las realizadas usando la interfaz gráfica, como las realizadas de manera manual. Por lo

tanto se puede asegurar que todos los elementos automatizados de la vivienda van a responder correctamente en tiempo real y de forma integrada.

3.3 PRUEBA INTEGRAL DEL SISTEMA DOMÓTICO EN TIEMPO PROGRAMADO

Puesto que el prototipo de sistema domótico implementado permite satisfacer las necesidades del usuario, tanto de manera instantánea como de manera programada, es necesario en este caso realizar las pruebas correspondientes para asegurarnos que no existen inconvenientes en cuanto a la ejecución de tareas programadas por parte del sistema automatizado.

3.3.1 PROCEDIMIENTO DE LA PRUEBA EN TIEMPO PROGRAMADO

Para poder realizar el control de la vivienda de manera programada, cabe recalcar que se debe utilizar la interfaz gráfica de usuario, la cual nos permite llevar a cabo estas funciones en la vivienda.

Para comprobar el correcto funcionamiento del sistema, se procede a acceder a la interfaz gráfica y marcar la tarea que se desea programar, es importante resaltar que el usuario puede programar una o varias actividades para que sean ejecutadas por el sistema, tal y como se muestra a continuación en la figura 3.28, en donde se puede observar la interfaz gráfica de usuario siendo programada para encender la luminaria de la sala, la luminaria de la habitación 1, la luminaria del ático y además selecciona abrir la persiana de la habitación 2.

Figura 3.28 Interfaz gráfica de usuario para seleccionar las tareas que deben ser ejecutadas

Una vez seleccionados los elementos que se desean controlar, se prosigue a seleccionar la hora y minuto a las cuales el usuario desea que se ejecute el control.

Cabe mencionar que el usuario elige tanto el tiempo de encendido como de apagado del dispositivo, de tal manera que el usuario no necesita estar en su vivienda para apagar las luminarias encendidas o para cerrar las persianas abiertas. A continuación en la figura 3.29 se muestra la manera de configurar el tiempo en la interfaz gráfica para ejecutar el control de la vivienda anteriormente mencionado, que en este caso será a las 20H00 para encender y abrir y a las 22H00 para apagar y cerrar los respectivos elementos.

Figura 3.29 Interfaz gráfica de usuario para configurar el tiempo de ejecución de las tareas

Finalmente se presiona el botón ENVIAR (figura 3.30) para que el controlador del sistema domótico (FPGA), guarde dicha información para ejecutarla a la hora programada por el usuario.

Figura 3.30 Interfaz gráfica de usuario para configurar las tareas y horario programados

Como se pudo observar en la figura 3.28, constan todos los lugares y elementos controlables de forma programada del sistema domótico. Para comprobar el correcto funcionamiento de este sistema, se seleccionaron todos los elementos disponibles en la interfaz gráfica y se programó el tiempo de encendido a las 19H30 y el tiempo de apagado a las 22H30 (figura 3.31), de tal manera de que podamos comprobar que el sistema responde a todos los requerimientos del usuario de forma programada.

Figura 3.31 Programación del control total del sistema

3.3.2 RESULTADOS DE LAS PRUEBAS DEL SISTEMA DOMÓTICO EN TIEMPO PROGRAMADO

Una vez que el sistema fue programado por el usuario para encender todos los dispositivos a las 19H30, a la hora mencionada observamos el comportamiento del prototipo y se evidenció que efectivamente todas las luminarias de la vivienda se encendieron y todas las persianas se abrieron, tal y como se muestra en la figura 3.32.

Figura 3.32: Ejecución del control de luminarias y persianas a la hora programada de encendido

Todos los elementos programados se mantuvieron encendidos hasta que fueron las 22H30, en donde todos los elementos que estaban encendidos y abiertos, se apagaron y cerraron automáticamente (figura 3.33), de tal manera que se comprobó que el sistema total integrado no tuvo conflictos al ejecutar las órdenes del usuario a largo plazo.

Figura 3.33: Ejecución del control de luminarias y persianas a la hora programada de apagado

Para comprobar que el tiempo en el cual se ejecuta el control del sistema domótico es el programado por el usuario, se debe observar el reloj que muestra la interfaz gráfica (figura 3.34), en donde el usuario puede observar la hora y minuto actual que corre en su sistema, de tal manera que pueda elegir correctamente los tiempos de ejecución de sus requerimientos. En caso de que el reloj esté adelantado o retrasado en comparación al tiempo actual, el usuario debe igualarlo en la misma interfaz con la finalidad de evitar errores en el desempeño del sistema.

Figura 3.34 Reloj del sistema domótico

Una vez comprobado el correcto funcionamiento individual e integrado de todo el prototipo de sistema domótico, tanto en tiempo real como de manera programada, se procedió a la determinación del consumo total de corriente y potencia que presenta el sistema domótico, tal y como se muestra en la tabla 3.10.

Tabla 3.10: Datos medidos de corriente y potencia del sistema domótico

LUGAR	DISPOSITIVO DE CONTROL	TIPO DE CARGA	CORRIENTE TOTAL CONSUMIDA EN REPOSO (mA)	CORRIENTE TOTAL CONSUMIDA EN FUNCIONAMIENTO (mA)
SALA	LUMINARIA	RESISTIVA	20	60
HABITACIÓN 1	LUMINARIA			
HABITACIÓN 2	LUMINARIA			
ENTRADA	LUMINARIA			
GARAJE	LUMINARIA			
COCINA	LUMINARIA			
BAÑO	LUMINARIA			
ÁTICO	LUMINARIA			
SALA	MOTOR	INDUCTIVA	40	250
HABITACIÓN 1	MOTOR	INDUCTIVA	40	250
HABITACIÓN 2	MOTOR	INDUCTIVA	40	250
ENTRADA	MOTOR	INDUCTIVA	40	250
GARAJE	MOTOR	INDUCTIVA	40	250
		TOTAL	**220**	**1310**

En base a la tabla 3.10 se puede observar que los dispositivos utilizados para la iluminación corresponden a luminarias de 20 W de potencia, mientras que los dispositivos utilizados para el control de las puertas y las persianas corresponden a motores a paso, cuyo consumo máximo de corriente en estado de trabajo es 250 mA, de esta manera se puede evidenciar que los elementos que permiten el control automatizado no presentan un alto de consumo de corriente ni de potencia dentro del sistema.

Cabe mencionar que una vez finalizadas las pruebas se determinaron ciertas limitaciones, las cuales en un sistema real se pueden superar con actuadores propios del protocolo X10. A continuación se mencionan las principales limitaciones del prototipo:

- Al ser un sistema unidireccional, el usuario debe reconocer el estado actual de las persianas de forma visual, de tal manera que se puedan satisfacer sus requerimientos con respecto al control de las persianas.

- El prototipo no es interactivo con el usuario, es decir que si el usuario desea controlar un nuevo dispositivo en su vivienda necesita configurar en el FPGA los códigos de CASA, UNIDAD y FUNCIÓN que se van a asignar a dicho elemento, por esta razón se necesita que las personas que diseñaron el prototipo, utilicen el manual de usuario para programar todos los elementos que el usuario desee agregar a su sistema domótico.

- Al ser una vivienda a escala, se usan luminarias de 20 W que se acoplan mejor al tamaño y estética del prototipo, las cuales en una vivienda real no brindan suficiente iluminación. Sin embargo el prototipo está diseñado de tal forma que se pueden usar luminarias de cualquier valor de potencia para cubrir de manera eficiente cada una de las áreas de una vivienda real.

Para finalizar la descripción de todas las pruebas realizadas tanto individualmente como de manera integrada, se asegura al usuario que el prototipo de sistema domótico responderá correctamente a todos sus requerimientos, tanto en tiempo real como de manera programada. Cabe resaltar que en este último caso no se encuentran activas las funciones de dimmer y de accesos puesto que hemos considerado que no son necesarias tanto por ahorro de energía, como por seguridad de la vivienda.

CAPÍTULO 4

COSTOS REFERENCIALES DEL PROTOTIPO

Hoy en día la tendencia de los hogares modernos apuntan hacia la automatización de la vivienda, buscando específicamente aumentar el confort, seguridad y ahorro energético, por lo cual el propósito de este Proyecto es brindar una solución económica y confiable, que permita satisfacer las principales requerimientos tecnológicos actuales y futuros de una vivienda.

A continuación se muestran cada una de las tablas que contienen el listado detallado de los materiales empleados en la construcción de los elementos que conforman el prototipo de sistema domótico.

La tabla 4.1 muestra los costos referenciales de los materiales empleados en la construcción del circuito de transmisión de las señales X10.

Tabla 4.1: Costo del circuito de transmisión de las señales X10

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	Capacitor X2 - 0.1uF - 275VAC	$ 0,55	$ 0,55
2	Capacitor cerámico - 15pF	$ 0,07	$ 0,14
1	Capacitor cerámico - 100nF	$ 0,07	$ 0,07
3	Capacitor electrolítico - 0.1uF	$ 0,07	$ 0,21
1	Resistencia 1MΩ	$ 0,02	$ 0,02
1	Resistencia 5MΩ	$ 0,02	$ 0,02
1	Resistencia 50Ω	$ 0,02	$ 0,02
1	Resistencia 200Ω	$ 0,02	$ 0,02
1	Resistencia 330Ω	$ 0,02	$ 0,02
1	Resistencia 10KΩ	$ 0,02	$ 0,02
1	Transistor - 2N3904	$ 0,07	$ 0,07
2	Diodo zener - 1N4736A - 6.8V	$ 0,98	$ 1,96
1	Cristal - 7MHz	$ 0,49	$ 0,49
1	Led rojo	$ 0,07	$ 0,07
1	Pulsador	$ 0,11	$ 0,11
1	Zócalo - 40 pines	$ 0,18	$ 0,18
1	PIC 16F877A	$ 5,10	$ 5,10
1	Placa de fibra de vidrio	$ 9,75	$ 9,75
1	Pines - 20	$ 0,40	$ 0,40
1	Fusible - 1A	$ 0,11	$ 0,11
1	**Circuito Transmisor X10**	**$ 19,34**	**$ 19,34**

La tabla 4.2 muestra los costos referenciales de los materiales empleados en la construcción del circuito de recepción de las señales X10.

Tabla 4.2: Costo del circuito de transmisión de las señales X10

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
2	Resistencia - 10KΩ	$ 0,02	$ 0,04
1	Resistencia - 33KΩ	$ 0,02	$ 0,02
1	Resistencia - 10MΩ	$ 0,02	$ 0,02
1	Resistencia - 470KΩ	$ 0,02	$ 0,02
1	Resistencia - 220KΩ	$ 0,02	$ 0,02
1	Resistencia - 47KΩ	$ 0,02	$ 0,02
1	Resistencia - 100KΩ	$ 0,02	$ 0,02
1	Resistencia - 330Ω	$ 0,02	$ 0,02
2	Capacitor cerámico - 150pF	$ 0,07	$ 0,14
2	Capacitor cerámico - 3.3nF	$ 0,07	$ 0,14
2	Capacitor cerámico - 4.7nF	$ 0,07	$ 0,14
1	Capacitor cerámico - 100pF	$ 0,07	$ 0,07
2	Capacitor cerámico - 10pF	$ 0,07	$ 0,14
1	Capacitor cerámico - 10nF	$ 0,07	$ 0,07
1	Capacitor cerámico - 100nF	$ 0,07	$ 0,07
2	Capacitor cerámico - 15pF	$ 0,07	$ 0,14
3	Capacitor electrolítico - 0.1uF	$ 0,07	$ 0,21
2	Bobina - 220uH	$ 0,36	$ 0,72
1	Diodo 1N4148	$ 0,07	$ 0,07
1	Inversor CD4069	$ 0,60	$ 0,60
1	Cristal - 3.579MHz	$ 0,49	$ 0,49
1	Led rojo	$ 0,07	$ 0,07
1	Pulsador	$ 0,11	$ 0,11
1	Zócalo - 40 pines	$ 0,18	$ 0,18
1	Zocalo - 14 pines	$ 0,09	$ 0,09
1	PIC 16F877A	$ 5,10	$ 5,10
1	Placa de fibra de vidrio	$ 7,35	$ 7,35
1	Pines - 20	$ 0,40	$ 0,40
1	Fusible - 1A	$ 0,11	$ 0,11
13	**Circuito Receptor X10**	**$ 16,62**	**$ 216,01**

La tabla 4.3 muestra los costos referenciales de los materiales empleados en la construcción del circuito para la comunicación serial.

Tabla 4.3: Costo del circuito de comunicación serial

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
4	Capacitor electrolítico - 1uF	$ 0,07	$ 0,29
1	Max232	$ 1,92	$ 1,92
1	Zócalo - 16 pines	$ 0,09	$ 0,09
1	Pines - 20	$ 0,40	$ 0,40
1	Placa baquelita	$ 3,00	$ 3,00
1	**Circuito del MAX232**	**$ 5,70**	**$ 5,70**

La tabla 4.4 muestra los costos referenciales de los materiales empleados en la construcción del circuito para la fuente de alimentación de los circuitos de transmisión y recepción de las señales X10.

Tabla 4.4: Costo del circuito de la fuente de alimentación de los circuitos de transmisión y recepción de las señales X10.

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	Capacitor electrolítico - 1uF	$ 0,07	$ 0,07
1	Capacitor electrolítico - 1000uF - 50V	$ 0,42	$ 0,42
1	Regulador de voltaje LM7805	$ 0,40	$ 0,40
1	Placa de fibra de vidrio	$ 2,10	$ 2,10
1	Pines - 20	$ 0,40	$ 0,40
1	Batería - 9V	$ 3,50	$ 3,50
14	**Circuito Fuente 5V**	**$ 6,89**	**$ 96,47**

La tabla 4.5 muestra los costos referenciales de los materiales empleados en la construcción del circuito para la fuente de alimentación de los circuitos actuadores de iluminación, persianas y puertas.

Tabla 4.5: Costo del circuito de la fuente de alimentación de los circuitos actuadores de iluminación, persianas y puertas.

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	Puente de diodos	$ 0,25	$ 0,25
1	Capacitor electrolítico - 1uF	$ 0,07	$ 0,07
1	Capacitor electrolítico - 1000uF - 50V	$ 0,42	$ 0,42
1	Regulador de voltaje LM7805	$ 0,40	$ 0,40
1	Fusible - 1A	$ 0,11	$ 0,11
1	Placa de fibra de vidrio	$ 5,00	$ 5,00
1	Pines - 20	$ 0,40	$ 0,40
1	Transformador 120 / 12 V	$ 11,00	$ 11,00
1	**Circuito Fuente 5V con Transformador**	**$ 17,65**	**$ 17,65**

La tabla 4.6 muestra los costos referenciales de los materiales empleados en la construcción del circuito actuador para el control de iluminación.

Tabla 4.6: Costo del circuito actuador para el control de iluminación

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
2	Resistencia - 1KΩ	$ 0,02	$ 0,04
1	Resistencia - 1.5KΩ	$ 0,02	$ 0,02
3	Resistencia - 10KΩ	$ 0,02	$ 0,06
1	Resistencia - 10KΩ - 5W	$ 0,13	$ 0,13
1	Potenciometro - 1KΩ	$ 0,25	$ 0,25
2	Capacitor cerámico - 1nF	$ 0,07	$ 0,14
4	Capacitor electrolítico - 1uF	$ 0,07	$ 0,29
2	Capacitor electrolítico - 0.1uF	$ 0,07	$ 0,14
1	Transistor - 2N3904	$ 0,07	$ 0,07
1	Pulsador	$ 0,11	$ 0,11
1	Zocalo - 40 pines	$ 0,18	$ 0,18
2	Zocalo - 6 pines	$ 0,05	$ 0,10
1	Microcontrolador ATMEGA 8535	$ 3,00	$ 3,00
1	Placa de fibra de vidrio	$ 6,15	$ 6,15
1	Pines - 20	$ 0,40	$ 0,40
1	Triac BT136	$ 0,58	$ 0,58
1	Moc 3021	$ 0,67	$ 0,67
1	Diodo 1N4007	$ 0,07	$ 0,07
1	Optoacoplador 4N25	$ 0,40	$ 0,40
8	**Circuito de control de iluminación**	**$ 12,80**	**$ 102,42**

La tabla 4.7 muestra los costos referenciales de los materiales empleados en la construcción del circuito actuador para el control de accesos y persianas.

Tabla 4.7: Costo del circuito actuador para el control de accesos y persianas

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	Resistencia - 10KΩ	$ 0,02	$ 0,02
1	Capacitor cerámico - 1nF	$ 0,07	$ 0,07
1	Capacitor electrolítico - 1uF	$ 0,07	$ 0,07
2	Capacitor electrolítico - 0.1uF	$ 0,07	$ 0,14
1	Pulsador	$ 0,11	$ 0,11
1	Zócalo - 40 pines	$ 0,18	$ 0,18
1	Zócalo - 16 pines	$ 0,09	$ 0,09
1	Microcontrolador ATMEGA 8535	$ 3,00	$ 3,00
1	Placa de fibra de vidrio	$ 6,15	$ 6,15
1	Pines - 20	$ 0,40	$ 0,40
1	Driver L293D	$ 2,46	$ 2,46
1	Motor a pasos	$ 5,00	$ 5,00
5	**Circuito de control de accesos y persianas**	**$ 17,69**	**$ 88,46**

La tabla 4.8 muestra los costos referenciales de los materiales empleados en la construcción del prototipo de vivienda.

Tabla 4.8: Costo del prototipo de vivienda

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	Vivienda a escala de madera	$ 60,00	$ 60,00
2	Plancha de triplex	$ 27,00	$ 54,00
2	Barniz	$ 5,00	$ 10,00
2	Brochas	$ 1,50	$ 3,00
4	Lijas	$ 0,35	$ 1,40
9	Focos	$ 0,75	$ 6,75
9	Boquillas	$ 0,70	$ 6,30
8	Cable eléctrico 12"	$ 0,45	$ 3,60
10	Cable de parlante	$ 0,35	$ 3,50
70	Cable UTP	$ 0,50	$ 35,00
5	Borneras grandes	$ 1,50	$ 7,50
5	Borneras pequeñas	$ 0,62	$ 3,10
100	Tornillos	$ 0,08	$ 8,00
1	Canaleta grande	$ 2,65	$ 2,65
2	Canaleta pequeña	$ 1,80	$ 3,60
1	Triplex pequeño	$ 8,50	$ 8,50
3	Manguito termocontraible	$ 0,80	$ 2,40
1	Tubo corrugado	$ 0,30	$ 0,30
4	Ganchos	$ 0,09	$ 0,36
1	Interruptor de luz	$ 4,10	$ 4,10
1	**Prototipo de vivienda**	**$ 224,06**	**$ 224,06**

La tabla 4.9 muestra el costo de la tarjeta Spartan 3E Starter Board de Xilinx (FPGA), la cual desempeña el papel de controlador en el sistema domótico implementado.

Tabla 4.9: Costo de la tarjeta Spartan 3E Starter Board

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
1	**Tarjeta Spartan 3E Starter Board (FPGA)**	**$200**	**$200**

La tabla 4.10 muestra los costos de los materiales extra que se emplearon para la implementación del prototipo.

Tabla 4.10: Costo de los materiales extra

CANTIDAD	DESCRIPCIÓN	PRECIO UNITARIO	PRECIO TOTAL
100	Cables hembra-macho	$ 0,30	$ 30,00
1	Cable serial macho	$ 25,00	$ 25,00
60	Postes placas (metálicos)	$ 0,45	$ 27,00
1	Material didáctico (goma, cartulina, tiraderas, taipe)	$ 15,00	$ 15,00
2	Amarras	$ 0,85	$ 1,70
1	Transformador aislamiento 120 / 120	$ 95,00	$ 95,00
1	**Material extra para la implementación del prototipo**	**$ 193,70**	**$ 193,70**

Finalmente en la tabla 4.11 se muestra el costo total del prototipo de sistema domótico implementado en el presente Proyecto.

Tabla 4.11: Costo Total del Proyecto

RUBRO	COSTO
Circuito Transmisor X10	$19.34
Circuito Receptor X10	$216.01
Circuito Fuente 5V	$96.41
Circuito Fuente 5V con Transformador	$17.65
Circuito de control de iluminación	$102.42
Circuito de control de accesos y persianas	$88.46
Circuito de comunicación serial	$5.70
Prototipo de vivienda	$224.06
Tarjeta Spartan 3E Starter Board (FPGA)	$200
Material extra	$193.7
TOTAL	**$1,163.81**

Como se puede observar, el costo total de la implementación de este prototipo es 1163 USD. Para poder comparar este sistema domótico con otros sistemas presentes en el mercado, no se debe incluir el costo de prototipo de vivienda, el cual corresponde al rubro de los materiales empleados en la construcción de la vivienda a escala (224 USD).

Descontando este rubro, el costo real del sistema domótico sería de 939 USD, el cual es un valor relativamente bajo tomando en cuenta el ahorro que implica a futuro

para el usuario, puesto que no requiere de la instalación de nuevo cableado para su implementación y los costos de mantenimiento son casi nulos. Además el consumo de energía de los módulos que conforman el sistema es bajo, por lo cual no implica un gran impacto económico a largo plazo para el usuario.

Es importante mencionar que el presente prototipo representa una alternativa de sistema domótico competente frente a otros sistemas actualmente existentes. Para ello a continuación se muestra una breve descripción y comparación entre el presente sistema domótico y otros sistemas de características similares que se ofrecen actualmente en el mercado.

4.1 COMPARACIÓN CON EL SISTEMA DOMÓTICO DE LA EMPRESA LUTRON [34]

Lutron es una compañía domótica dedicada a la creación y comercialización de sistemas alámbricos e inalámbricos para el control de iluminación y persianas alrededor del mundo.

El costo de los módulos que permiten el control de iluminación y persianas en ocho lugares dentro de una vivienda de manera alámbrica tienen un costo de $ 1930, este precio no incluye el control de los accesos a la vivienda, lo cual implica que el sistema domótico de Lutron es más costoso que el sistema propuesto en este proyecto (939 USD). En el Anexo 23, se muestra de manera detallada el precio de cada módulo de Lutron necesario para la implementación de un sistema domótico de características similares al sistema implementado en este Proyecto.

4.2 COMPARACIÓN CON EL SISTEMA DOMÓTICO COMERCIAL X10 [29]

Los módulos X10 comerciales que se deben emplear para la implementación de un sistema domótico que permita controlar la iluminación y persianas en ocho lugares de una vivienda, tiene un costo aproximado de $585, tal y como se detallan a continuación en la tabla 4.12:

Tabla 4.12: Costo de los módulos comerciales X10

Cantidad	Elemento	Precio Unitario	Precio Final
4	Minimando RF-4 Slimfire	$21	$84
1	Receptor RF/MA	$32.33	$32.33
8	Módulo de iluminación con cable	$33.66	$269.28
3	Mando de Persiana Empotrable + Receptor RF	$66.56	$199.68
		TOTAL	**$585.29**

Se debe tomar en cuenta que este valor no incluye el costo de los módulos para el control de puertas, el uso de una tarjeta Spartan 3E como controladora, ni el precio de los elementos que se van a controlar dentro de la vivienda. Por lo cual el costo de un sistema domótico que utiliza módulos comerciales X10, tiene un valor similar a un sistema que utiliza los módulos construidos en este proyecto.

Una vez finalizadas las comparaciones se puede deducir que el sistema domótico implementado en este Proyecto, puede competir en el mercado local de manera exitosa tanto en el ámbito tecnológico como económico, ofreciendo una nueva alternativa y oportunidad a todos los usuarios interesados en automatizar su vivienda de manera confiable.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Una vez efectuadas las pruebas de transmisión de instrucciones hacia el sistema domótico, se puede concluir que la utilización del protocolo X10 fue acertada, puesto que las señales de control se transmitieron eventualmente sin problemas a través de la red eléctrica existente, permitiendo un ahorro de tiempo y de dinero, lo que permite concluir que la selección del protocolo mencionado fue correcta.

- Se seleccionó la tarjeta Spartan 3E-Starter Board (FPGA) por ser una de las más económicas del mercado. De los resultados obtenidos en la utilización de la misma, se pudo verificar que sus características físicas y de procesamiento son las adecuadas para la implementación de un sistema domótico, por lo mismo se puede concluir que su selección fue acertada.

- Gracias al diseño y construcción separada de cada circuito transmisor, receptor y actuador se pudo verificar la adaptación de de cada uno de los circuitos construidos en diferentes lugares de la vivienda, con lo cual se pudo efectuar un control eficiente de todos los elementos del sistema domótico, por esta razón se concluyó que la idea de diseñar de manera independiente cada uno de los circuitos del sistema domótico fue exitosa.

- El prototipo implementado está diseñado de tal forma que ofrece escalabilidad y reconfiguración dependiendo de las necesidades del usuario. Los resultados de las pruebas permitieron concluir que la idea de agregar al software del FPGA los códigos X10 de los dispositivos de control o actuación fue correcta por la facilidad con que esto se realizó.

- Para permitir la comunicación entre el usuario y el sistema domótico se diseñó e implementó una interfaz gráfica desarrollada en Labview. De los resultados obtenidos al enviar las instrucciones desde la interfaz gráfica hacia el sistema domótico, se puede concluir que la interfaz desarrollada es amigable y de fácil manejo para el usuario, cumpliendo de manera satisfactoria la funcionalidad deseada.

- Durante las pruebas realizadas en el laboratorio para la transmisión de las señales X10 a través de la red eléctrica, se pudo observar que existe muchas señales de ruido que hacen difícil el reconocimiento de las señales domóticas en el circuito receptor, por lo tanto se puede concluir que uno de los principales problemas que se presentan durante la transmisión de las señales X10 en la vivienda, es la interferencia provocada por ruidos dentro de la red eléctrica. Las fuentes más comunes de ruido eléctrico son: motores, luces fluorescentes, halógenas, y de vapor de sodio, computadoras, entre otros.

- Las pruebas de funcionamiento efectuadas sobre el sistema domótico, permitieron concluir que la intensidad de las señales en la red eléctrica va disminuyendo cuanto mayor es su recorrido. Para contrarrestar este efecto, se hizo uso de un transformador de aislamiento, con el fin de limitar el recorrido de las señales X10 dentro de la red eléctrica. Evitando de esta manera que las señales X10 se atenúen, y permitiendo además que el dispositivo receptor pueda detectar y amplificar las señales X10 que llegan hasta él.

- Tomando en cuenta que el sistema domótico fue implementado usando un único transmisor y múltiples receptores, es importante mencionar que se realizaron modificaciones al estándar para incluir nuevas aplicaciones al protocolo, tales como la reutilización de actuadores en diferentes lugares de la vivienda, con lo cual se puede concluir que el prototipo desarrollado no es totalmente compatible con los sistemas domóticos X10 comerciales.

- El objetivo del proyecto fue diseñar e implementar un prototipo de sistema domótico controlado por un FPGA, de tal manera que se pueda programar una aplicación domótica en el mismo. También fue emplear el protocolo de domótica X10, para la comunicación entre los sensores y actuadores del sistema. Los resultados fueron del 100 % de efectividad las pruebas a las que se sometió todo el sistema integrado, por lo cual permiten concluir que se cumplió con este gran objetivo.

5.2 RECOMENDACIONES

- El prototipo de sistema domótico puede ser mejorado y competir con las aplicaciones que presentan los sistemas modernos, para lo cual, se recomienda explotar las características de la FPGA implementando un servidor web de tal manera que se pueda controlar el sistema remotamente por Internet y localmente dentro de una red LAN, lo cual, abre la posibilidad de desarrollo de nuevas aplicaciones y permitir el control desde nuevos dispositivos (celulares y tablets).

- Igualmente, se puede desarrollar el prototipo implementado haciendo uso de la interfaz gráfica en Labview para controlar el sistema remotamente por Internet o localmente dentro de una red LAN mediante el uso de un programa de escritorio remoto.

- El uso de múltiples transmisores en el sistema domótico presenta el problema de ocurrencia de colisiones, para solventar este problema se recomienda la implementación de una comunicación maestro – esclavo entre transmisores o un protocolo de detección/evasión de colisiones.

- Si se desea usar múltiples transmisores, se recomienda también el uso de botoneras con ambientes pre-establecidos, los cuales, permiten colocar a los actuadores en estados que el usuario desee.

- Para implementar el prototipo diseñado en una vivienda real, se recomienda la instalación de filtros X10 comerciales en la red eléctrica que se vaya a utilizar como medio de transmisión, con el objetivo de asegurar que las señales X10 se transmitan sin problemas dentro de la vivienda. De igual manera, en el mercado existen dispositivos que permiten atenuar la interferencia por ruido eléctrico generado por electrodomésticos y motores, además de evitar la interferencia de señales X10 provenientes de otras viviendas.

REFERENCIAS BIBLIOGRÁFICAS

LIBROS, MANUALES Y PUBLICACIONES

[1] ANÓNIMO. “28BYJ-48 – 5V Stepper Motor”. Kiatronics. Nueva Zelanda. Mayo, 2014.

[2] ANÓNIMO. “CD4069UBC Inverter Circuits”. Fairchild Semiconductor TM. Enero, 1999.

[3] ANÓNIMO. “EMI/RFI Suppression Capacitors”. Illinois Capacitor, Inc. Mayo, 2014.

[4] ANÓNIMO. “Spark Quenchers Introduction RC Networks”. OKAYA. Mayo, 2014.

[5] ANÓNIMO. “SPARTAN-3E FPGA Starter Kit Board User Guide. UG230 (v 1,1)”. Xilinx. 20 junio de 2008.

[6] ANÓNIMO. “Triacs”. Philips Semiconductors. Agosto, 1997.

[7] ANÓNIMO. “Tutorial de Labview”. España. Mayo, 2014.

[8] ARIAS, J. “Circuitos Analógicos con inversores CMOS”. 9 de julio del 2008.

[9] BURROUGHS, Jon. “X-10 Home Automation Using the PIC16F877A”. Microchip Technology, Inc. 9 de Abril del 2010.

[10] DÍAZ INFANTES, Juan A. “Descripción de X10, Biblioteca de Conexión de Arduinos con el protocolo X10”. Departamento Lenguajes y Ciencias de la Computación, Universidad de Málaga. España. 25 de Enero del 2009.

[11] JIMÉNEZ JIMÉNEZ, María Soledad. Folleto “Comunicación Digital”. Escuela Politécnica Nacional. Quito, Ecuador. Enero, 2011.

PROYECTOS DE TITULACION

[12] ALVERCA MAZA, Yohana Elizabeth. “Síntesis de Circuitos Digitales utilizando VHDL (VHSIC Hardware Description Language) y FPGAs (Field Programmable Gate Arrays)”. EPN. Febrero, 2008.

[13] ANÓNIMO. “Sistemas Domóticos utilizando OpenSoftware y OpenHardware”.

[14] BAENA ALONSO, Alejandro. “Diseño sobre FPGA de una Unidad Aritmética Decimal”. Universitat Rovira I Virgili. Febrero, 2010.

[15] JIMÉNEZ ESPINOSA, Pablo Xavier. “Diseño e implementación de un sistema de adquisición, compresión y almacenamiento de imágenes empleando VHDL (VHSIC Hardware Description Language) y FPGAs (Field Programmable Gate Arrays)”. EPN. Enero, 2011.

[16] JIMÉNEZ NUÑEZ, Sixto Alejandro. PANCHI CAMPOS, Diego Enrique. “Diseño e implementación de un modulador y demodulador N-QAM empleando Xilinx ISE, System Generator y Simulink sobre una tarjeta de entrenamiento basada en un FPGA de Xilinx”. EPN. Mayo 2011.

[17] JIMÉNEZ VILLANUEVA, Mayra Alejandra. ORTIZ CASALLAS, Diana Carolina. “Diseño e implementación de un sistema domótico por control vía celular”. Universidad Distrital Francisco José de Caldas. Marzo, 2011.

[18] LEMA SAMANIEGO, Magaly Alexandra. QUINTUÑA PADILLA, Sandro Xavier. VILLA ZARI, Milton Patricio. “Diseño e implementación de un sistema domótico mediante la utilización de tecnologías celular e internet”. UPS. Abril, 2008.

[19] LÓPEZ ENCALADA, Rodrigo Alexander. “Control del Módulo Pendubot utilizando una FPGA”. EPN. Agosto, 2011.

DIRECCIONES ELECTRÓNICAS

[20] Acuña Rodrigo, Ahumada Jonhson, Avendaño Mariela. “Protocolo X10”. [En línea].Mayo del 2011. [citado 4 de Octubre del 2013]. Disponible en la web: http://www2.udec.cl/~racuna/domotica/x10.htm

[21] ANÓNIMO. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://librairie.immateriel.fr/baw/9780596154141/httpatomoreillycomsourceoreillyimages308073.png

[22] ANÓNIMO, “¿Cuáles son los Estándares Internacionales de la Domótica en la Actualidad?”. IECOR. Córdoba-Argentina. [En línea]. 2007-2013. [citado 19 de abril del 2014]. Disponible en la web: http://www.iecor.com/domotica-cordoba/informacion/domotica-estandares-protocolos.html

[23] ANÓNIMO. “ALL PROGRAMABLE FPGAs”. Xilinx Inc. Copyright 2014. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://www.xilinx.com/products/silicon-devices/fpga/index.htm

[24] ANÓNIMO. “Breve Historia e Introducción a la Domótica”. Casa Control. [En línea]. 28 de febrero de 2011. [citado 19 de abril del 2014]. Disponible en la web: http://casa-control.com/blog/?p=25

[25] ANÓNIMO, “Concepto de Domótica”. BLOGDOMOTICA.COM. [En línea]. 2014. [citado 19 de abril del 2014]. Disponible en la web: http://www.blogdomotica.com/concepto-de-domotica/

[26] ANÓNIMO. “Dispositivos Lógicos Programables”. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://www.jimenez-ruiz.es/ernesto/II/VHDL/vhdl.html

[27] ANÓNIMO, “DISPOSITIVOS LÓGICOS PROGRAMABLES”. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://yaqui.mxl.uabc.mx/~aglay/PLDS.pdf

[28] ANÓNIMO. “Electronic Design”. [En línea]. [citado 18 de mayo del 2014]. Disponible en la web: http://electronicdesign.com/site-files/electronicdesign.com/files/archive/electronicdesign.com/files/29/5687/figure_02.gif

[29] ANÓNIMO. “DOMOTICA VIVA.COM - DOMÓTICA X10”. ©Domótica Viva S.L. [En línea]. [citado 3 de septiembre del 2014]. Disponible en la web: http://www.domoticaviva.com/Tienda/tienda_domotica_x10.php

[30] ANÓNIMO, “ELECTRÓNICA DIGITAL I, Capítulo 4: Dispositivos Lógicos Programables”. Bogotá-Colombia. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://www.virtual.unal.edu.co/cursos/ingenieria/2000477/lecciones/040401.htm

[31] ANÓNIMO, “El protocolo de comunicaciones, el lenguaje de la domótica”. DomoPrac. [En línea].2007-2014. [citado 19 de abril del 2014]. Disponible en la web: http://www.domoprac.com/protocolos-de-comunicacion-y-sistemas-domoticos/el-protocolo-de-comunicaciones-el-lenguaje-de-la-domotica.html

[32] ANÓNIMO. “Interfaz Gráfica de Usuario (GUI)”. Fundación del Español Urgente. [En línea]. [citado 18 de mayo del 2014]. Disponible en la web: http://www.fundeu.es/escribireninternet/interfaz-grafica-de-usuario-gui/

[33] ANÓNIMO. “LAB. MICROELECTRÓNICA-ITT SE FLUJO DE DISEÑO SOBRE FPGAs Versión 1.0”. Departamento Electrónica. [En línea]. Marzo, 2009. [citado 19 de abril del 2014]. Disponible en la web: http://delep.uah.es/antigua/PLANES%20ANTIGUOS/ITTSSEE/3/Lab.%20Microelectr%F3nica/Apuntes/P2_1.pdf

[34] ANÒNIMO. “LUTRON - NUESTRA HISTORIA”. Lutron Electronics Co., Inc. [En línea]. [citado 3 de septiembre del 2014]. Disponible en la web: http://www.lutron.com/es-LA/Company-Info/Paginas/AboutUS/OurStory.aspx

[35] ANÓNIMO. “MPRCP & MPRCS, EMI/RFI Suppression Capacitors-Parallel RC Unit and EMI/RFI Suppression Capacitors-Series RC Unit (X2 Capacitor)”. Livingston Industrial (Group) Ltd. [En línea]. [citado 18 de mayo del 2014]. Disponible en la web: http://www.livingston.com.tw/es/2_1993_66985/product/Supresi%C3%B3n_de_la_EMI_de_condensadores_Unidad_RC_paralelo_id369285.html

[36] ANÓNIMO. “NI FPGA, ¿Qué es un FPGA?”. NATIONAL INSTRUMENTS CORPORATION. [En línea]. 2012. [citado 19 de abril del 2014]. Disponible en la web: http://www.ni.com/fpga/esa/

[37] ANÓNIMO. “Programación en VHDL. Introducción”. Wikilibros. [En línea]. 22 de noviembre del 2013. [citado 19 de abril del 2014]. Disponible en la web: http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_VHDL/Introducci%C3%B3n

[38] ANÓNIMO. “Tecnología 2.0. Hogares inteligentes ó domótica doméstica: Introducción al X10”. [En línea]. [citado 15 de Octubre del 2013]. Disponible en la web: http://tecnologiadospuntocero.files.wordpress.com/2011/03/senalx10.jpg

[39] ANÓNIMO, “TIPOS DE PLD´s”. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://server-die.alc.upv.es/asignaturas/lsed/2002-03/PLDs/Web%5Ctipos.htm

[40] ANÓNIMO, “Todo sobre Edificios Inteligentes”. CASADOMO.COM. [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://www.casadomo.com/images/news/canal_domotica_intro_ejemplos_dispositivos.gif

[41] ANÓNIMO. “VHDL”. Wikipedia. [En línea]. 3 de enero del 2014. [citado 19 de abril del 2014]. Disponible en la web: http://es.wikipedia.org/wiki/VHDL

[42] ANÓNIMO. “Xilinx”. Wikipedia. [En línea]. 20 de octubre del 2013. [citado 19 de abril del 2014]. Disponible en la web: http://es.wikipedia.org/wiki/Xilinx

[43] BERBIN, Yigsina. COA, Guillermo. "Electrónica Integrada. VHDL". [En línea]. [citado 19 de abril del 2014]. Disponible en la web: http://ygelectronicaintegrada.blogspot.com/2008/01/vhdl.html

[44] Domingo. "Ascensor con motor paso a paso". Monografías. [En línea]. [citado 18 de mayo del 2014]. Disponible en la web: http://www.monografias.com/trabajos90/ascensor-motor-paso-paso/ascensor-motor-paso-paso.shtml#secuenciab

[45] Driscoll, Edward B. Jr. "The History of X10". [En línea]. 10 de Octubre del 2013. [citado 19 de abril del 2014]. Disponible en la web: http://home.planet.nl/~lhendrix/x10_history_english.htm

[46] GUICHAL, Guillermo. "Diseño Digital Utilizando Lógicas Programables" Versión 0.6. [En línea]. Junio 29 del 2005. [citado 19 de abril del 2014]. Disponible en la web: http://es.scribd.com/doc/122471802/50015304-NotasCompletas-Sobre-Fpga-2

[47] Prag, Micke. Standard and Extended X10 Code Protocol. [En línea]. 11 de Julio del 2011. [citado 24 de Octubre del 2013]. Disponible en la web: http://developer.telldus.se/attachment/ticket/124/X10%20xtdcode.pdf

[48] Smart Home USA. "How X10 Works". [En línea]. 2013. [citado 10 de Octubre del 2013]. Disponible en la web: http://www.smarthomeusa.com/info/x10theory#theory

ANEXOS

ANEXO 1: ARQUITECTURA DE LA FPGA Spartan III DE XILINX

Las FPGA Spartan III de Xilinx están conformadas por un conjunto de Bloques Lógicos Configurables (Configurable Logic Blocks: CLBs) rodeados por un perímetro de Bloques Programables de entrada/salida (Programmable Input/Output Blocks: IOBs). Estos elementos funcionales están interconectados por una jerarquía de canales de conexión (Routing Channels), la que incluye una red de baja capacitancia para la distribución de señales de reloj de alta frecuencia. Adicionalmente el dispositivo cuenta con 24 bloques de memoria RAM de 2Kbytes de doble puerto, cuyos anchos de buses son configurables, y con 12 bloques de multiplicadores dedicados de 18 X 18 bits.

Los cinco elementos funcionales programables que la componen son los siguientes:

- Bloques de entrada/salida (Input/Output Blocks – IOBs): Controlan el flujo de datos entre los pines de entrada/salida y la lógica interna del dispositivo. Soportan flujo bidireccional más operación tri-estado y un conjunto de estándares de voltaje e impedancia controlados de manera digital.
- Bloques Lógicos configurables (Configurable Logic Blocks – CLBs): Contienen Look-Up Tables basadas en tecnología RAM (LUTs) para implementar funciones lógicas y elementos de almacenamiento que pueden ser usados como *flip-flop*s o como *latches*.
- Bloques de memoria RAM (Block RAM): Proveen almacenamiento de datos en bloques de 18 Kbits con dos puertos independientes cada uno.
- Bloques de multiplicación que aceptan dos números binarios de 18 bit como entrada y entregan uno de 36 bits.
- Administradores digitales de reloj (Digital Clock Managers – DCMs): Estos elementos proveen funciones digitales auto calibradas, las que se encargan de distribuir, retrazar arbitrariamente en pocos grados, desfasar en 90, 180, y 270 grados, dividir y multiplicar las señales de reloj de todo el circuito.

Los elementos descritos están organizados como se muestra en la Figura 11. Un anillo de IOBs rodea un arreglo regular de CLBs. Atraviesa este arreglo una columna de Bloques de memoria RAM, compuesta por varios bloques de 18 Kbit, cada uno de los cuales está asociado con un multiplicador dedicado. Los DCMs están colocados en los extremos de dichas columnas.

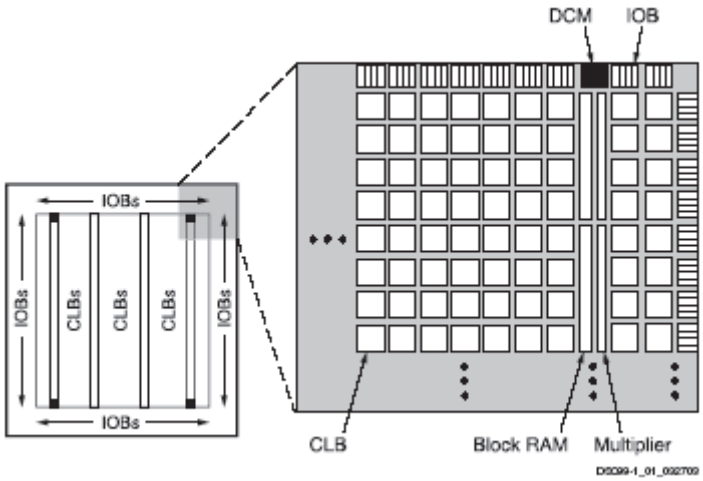


Figura 1: Arquitectura de la Spartan III

A continuación se hace una descripción más detallada de cada uno de los elementos funcionales de la FPGA, y luego se describe el proceso de configuración de la misma.

Bloques de entrada/salida IOB

Los bloques de entrada/salida (IOB) suministran una interfaz bidireccional programable entre un pin de entrada/salida y la lógica interna de la FPGA. Un diagrama simplificado de la estructura interna de un IOB aparece en la Figura . Hay tres rutas para señales en un IOB: la ruta de salida, la ruta de entrada y la ruta tri-estado. Cada ruta tiene su propio par de elementos de almacenamiento que pueden actuar tanto como registros o como latches. Las tres rutas principales son como sigue:

- La ruta de entrada, que lleva datos desde el *pad*, que está unido al pin del package, a través de un elemento de retardo opcional programable, directamente a la línea I. Después del elemento de retardo hay rutas alternativas a través de un par de elementos de almacenamiento hacia las líneas IQ1 e IQ2. Las tres salidas del IOB todas conducen a la lógica interna de la FPGA.
- La ruta de salida, que parte con las líneas O1 y O2, lleva datos desde la lógica interna de la FPGA, a través de un multiplexor y del *driver* tri-estado hacia el *pad* del IOB. En suma a esta ruta directa, el multiplexor da la opción de insertar un par de elementos de almacenamiento.
- La ruta tri-estado determina cuando el driver de salida está en alta impedancia. Las líneas T1 y T2 llevan datos desde la lógica interna a través de un multiplexor hacia el driver de salida. En suma a esta ruta directa, el multiplexor da la opción de entregar un par de elementos de almacenamiento.

Todas las rutas de señales que entran al IOB, incluidas aquellas asociadas con los elementos de almacenamiento tienen una opción de inversión. Cualquier inversor colocado (en la programación) en estas rutas es automáticamente absorbido dentro del IOB.

Hay tres pares de elementos de almacenamiento en cada IOB, un par para cada uno de las tres rutas. Es posible configurar cada uno de esos elementos como un *flip-flop* D gatillado por flanco (FD) o como un *latch* sensible a nivel (LD). Estos elementos son controlados con la misma red de distribución de relojes que se utiliza para todo el sistema.

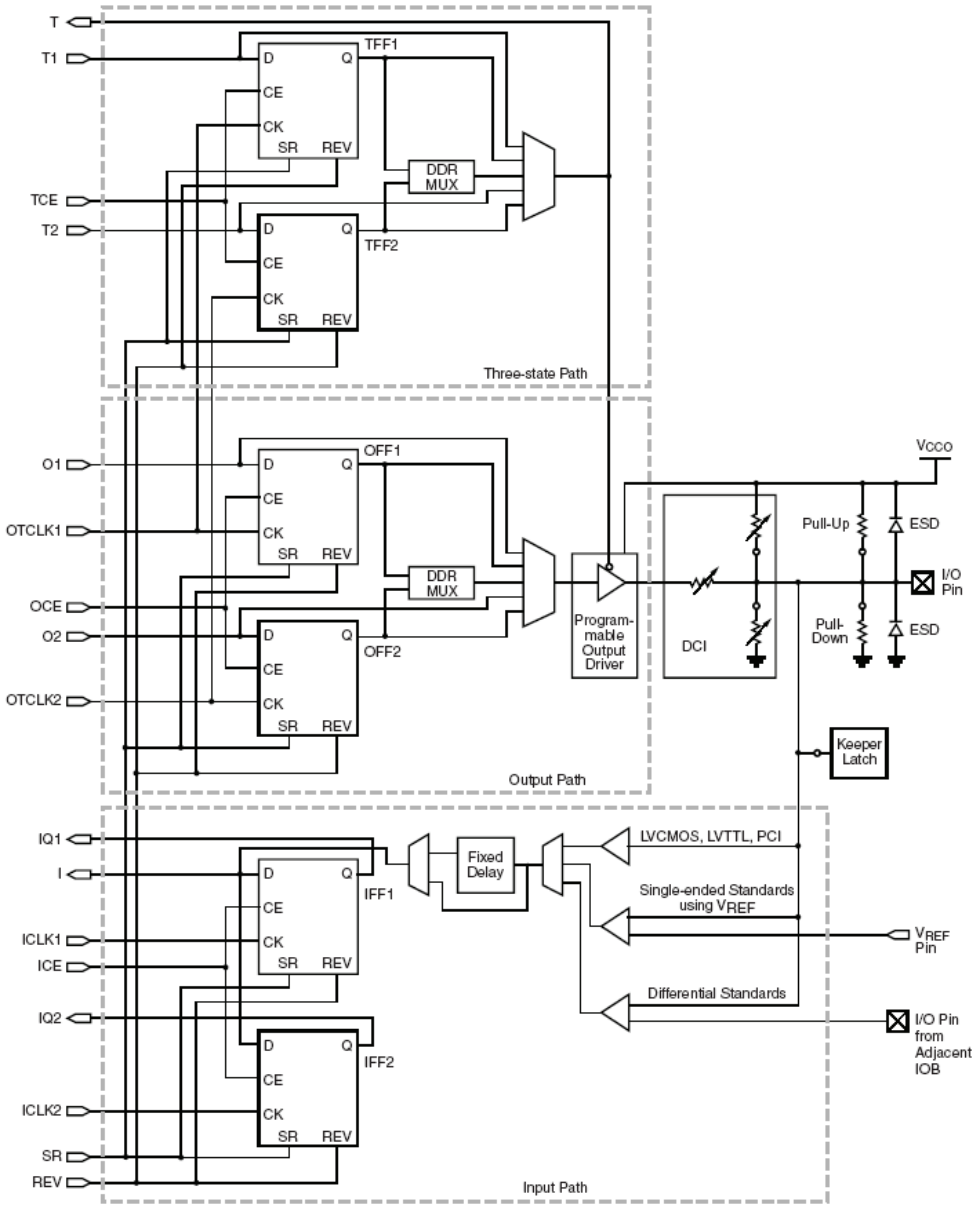


Figura 2: Diagrama simplificado de un IOB de la Spartan III

El par de elementos de almacenamiento tanto de la ruta de salida o de la del *driver* tri-estado pueden ser usados en conjunto, con un multiplexor especial para producir transmisión de doble tasa de datos (DDR). Esto se logra tomando datos sincronizados con el flanco de subida del reloj y convirtiéndolos en bits sincronizados

tanto con el flanco de subida como con el de bajada. A esta combinación de dos registros y un multiplexor se le llama *flip flop* tipo D de doble tasa de datos (FDDR).
Cada IOB cuenta además con otros elementos, entre los cuales cuentan las resistencias de *Pull-Up* y de *Pull-Down*, que tienen el objetivo de establecer niveles altos o bajos respectivamente en las salidas de los IOBs que no están en uso; un circuito de retención (Keeper) del último nivel lógico que se mantiene, después de que todos los drivers han sido apagados, lo que es útil para cuidar que las líneas de un bus no floten, cuando los drivers conectados están en alta impedancia; un circuito de protección para descargas electro estáticas (protección ESD), que utiliza diodos de protección.

Finalmente cada IOB cuenta con un control para el *slew rate* y para la corriente de salida máxima. El primero otorga la posibilidad de elegir una tasa alta de cambio de nivel (con bajo *slew rate*) o una tasa máxima menor, pero con un control de transciente, para la utilización de los puertos en la integración a buses, donde al pasar de alta impedancia a un nivel de voltaje suele producirse transiciones inesperadas. El segundo entrega siete niveles deferentes de corrientes máximas tanto para el estándar CMOS como para el TTL, lo que permite adaptarse a dispositivos que necesitan mayores corrientes para su activación; en el caso del estándar LVCMOS a 2.5V el rango de corrientes es de 2 a 24 mA.(2, 4, 6, 8, 12, 16, 24 mA).

Los IOB soportan 17 estándares de señales de salida de terminación única y seis de señal diferencial; también cuentan con un sistema integrado, para coincidir con la impedancia de las líneas de transmisión que llegan a la FPGA, llamado Control Digital de Impedancia (DCI), el que permite elegir hasta 5 tipos diferentes de terminaciones, utilizando una red de resistencias internas que se ajustan en serie o en paralelo, dependiendo de las necesidades del estándar elegido.

Bloques de Lógica Configurable (CLB)

El bloque básico de la red que compone la FPGA es la *slice*. Existen dos tipo de slice, éstas se diferencian en algunos elementos, pero son muy parecidas, (ver más adelante). Luego estas *slices* se organizan en los bloques lógicos elementales, que son los que se describen a continuación.

Los Bloques de Lógica Configurable (CLBs) constituyen el recurso lógico principal para implementar circuitos síncronos o combinacionales. Cada CLB está compuesta de cuatro slices interconectadas entre si, tal como se muestra en la Figura3.

Las cuatro slices que componen un CLB tienen los siguientes elementos en común: dos generadores de funciones lógicas, dos elementos de almacenamiento, multiplexores de función amplia, lógica de *carry* y compuertas aritméticas, tal como se muestra en la Figura . Los dos pares de *slices* usan estos elementos para entregar funciones lógicas y aritméticas de ROM. Además de lo anterior, el par de la izquierda soporta dos funciones adicionales: almacenamiento de datos usando RAM distribuida y corrimiento de datos con registros de 16 bits.

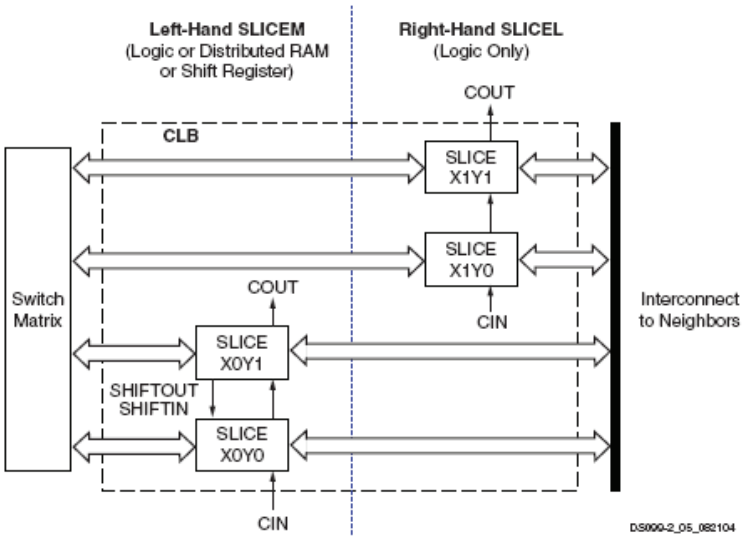


Figura 3: Arreglo de slices en un CLB

La Figura 4 es un diagrama de una slice del par del lado izquierdo, por lo tanto representa un súper conjunto de los elementos y conexiones que se encuentran el las *slices*.

El generador de funciones basado en RAM –también conocido como *Look-Up Table* (LUT)- es el recurso principal para implementar funciones lógicas dentro de la FPGA. Más aún, las LUTs en cada par de *slices* del lado izquierdo pueden ser configuradas como RAM distribuida o como un registro de corrimiento de 16 bits. Los generadores de funciones ubicados en las porciones superiores e inferiores de la slice son referidos como “G-LUT” y “F-LUT” respectivamente en la Figura 4.

El elemento de almacenamiento, el cual es programable tanto como un *flip flop* tipo D o como un *latch* sensible a nivel, provee un medio para sincronizar datos a una señal de reloj, entre otros usos. Estos elementos de almacenamiento, que se encuentran en las porciones superiores e inferiores de la slice son llamados “FFY” y “FFX”, respectivamente.

Los multiplexores de función amplia combinan las LUTs para permitir operaciones lógicas más complejas, cada slice tiene dos de éstos, en la Figura 4 corresponden a F5MUX y F1MUX.

La cadena de *carry*, en combinación con varias compuertas lógicas dedicadas, soportan implementaciones rápidas de operaciones matemáticas. La cadena de *carry* entra a la *slice* como CIN y sale como COUT. Cinco multiplexores controlan la cadena: CYINIT, CY0F y CYMUXF en la porción inferior, así como CY0G y CYMUXG en la porción superior. La lógica aritmética dedicada incluye compuertas XOR y AND en cada porción de la slice.

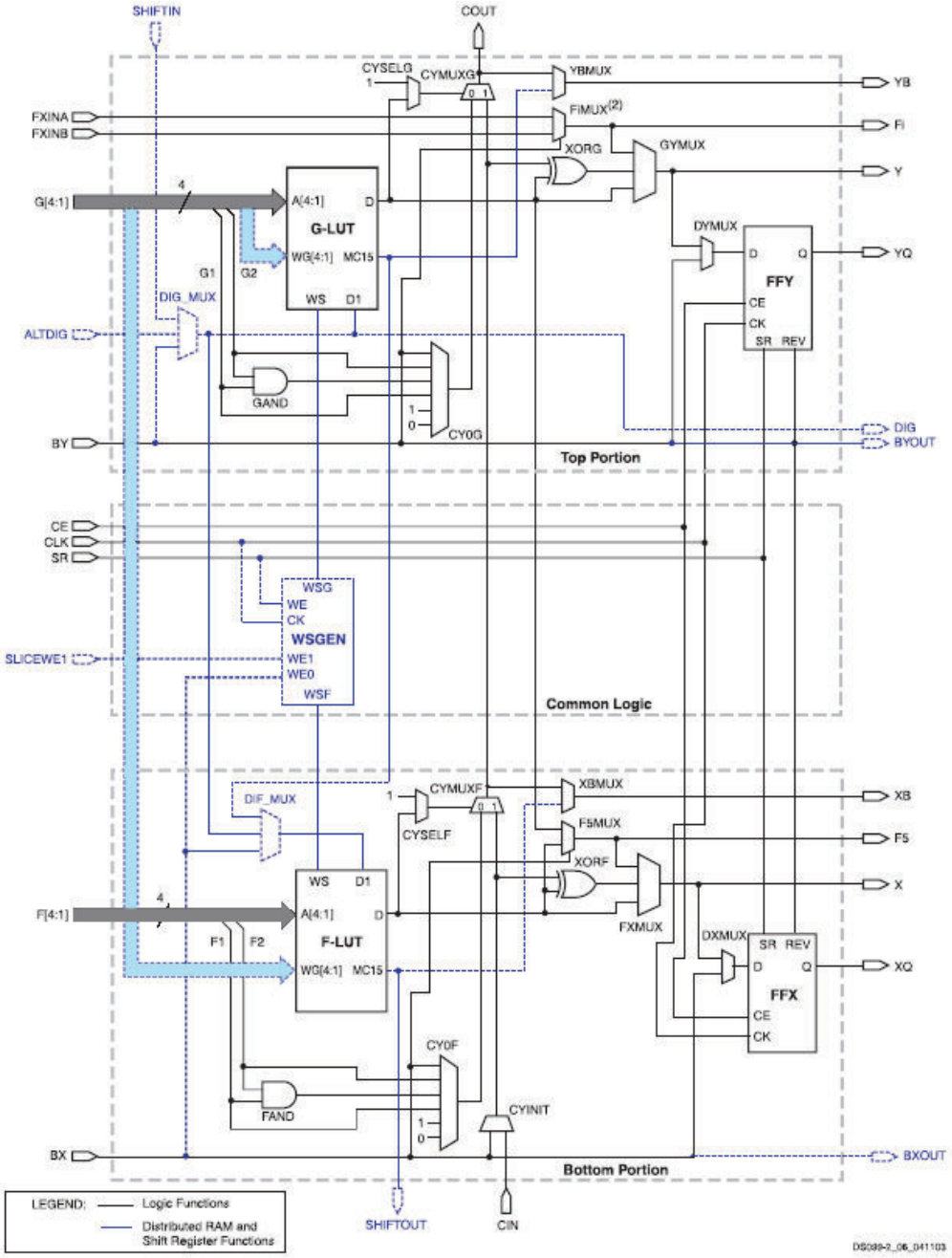


Figura 4: Diagrama simplificado de una slice del lado izquierdo de un CLB

Con un rol central en la operación de cada *slice* se encuentran dos rutas de datos casi idénticas. Para la descripción que prosigue se usan los nombres de la parte inferior de la Figura . La ruta básica tiene su origen en la matriz de *switches* de interconexión colocada fuera del CLB. Cuatro líneas, F1 a F4 entran en la slice y se conectan directamente a la LUT. Una vez dentro de la slice, la ruta de los 4 bits inferiores pasa a través de un generador de funciones F que realiza operaciones lógicas. La ruta de salida del generador de funciones, D, ofrece cinco posibles rutas posibles:

- Salir de la *slice* por la línea X y volver a interconectarse.
- Dentro de la *slice*, X sirve como entrada al DXMUX que alimenta la entrada de datos D, correspondiente al elemento de almacenamiento FFY. La salida Q de este elemento maneja la ruta XQ que sale de la *slice*.
- Controlar el multiplexor CYMUXF de la cadena de *carry*.
- Con la cadena de *carry*, servir como una entrada a la compuerta XORF, que realiza operaciones aritméticas y produce el resultado en X.
- Manejar el multiplexor F5MUX para implementar funciones lógicas más anchas que 4 bits. Las salidas D de los F-LUT y G-LUT sirven de entradas de datos para este multiplexor.

En suma a los caminos lógicos principales descritos recién, existen dos rutas de *bypass* que entran a la slice como BX y BY. Una vez dentro de la FPGA, BX en la parte de debajo de la slice (o BY en la parte superior) puede tomar cualquiera de varias ramas diferentes:

- Hacer *bypass* de la LUT y del elemento de almacenamiento, luego salir de la slice como BXOUT y volver a interconectarse.
- Hacer *bypass* a la LUT, y luego pasar a través del elemento de almacenamiento, para luego salir como XQ.
- Controlar el multiplexor F5MUX.

- Servir como una entrada a la cadena de *carry* vía los multiplexores.
- Manejar la entrada DI de la LUT.
- BY puede controlar la entrada REV de FFY y de FFX.
- Finalmente, el multiplexor DIG_MUX puede derivar la ruta BY hacia la línea DIG que sale de la *slice*.

Cada una de las dos LUTs (F y G) de una *slice* tiene cuatro entradas lógicas (A1–A4) y una única salida D. Esto permite programar cualquier operación lógica booleana de cuatro variables en este dispositivo. Además, los multiplexores de función amplia pueden usarse para combinar LUTs dentro del mismo CLB o incluso a través de diferentes CLBs, haciendo posible funciones con mayor número de variables.

Las LUT de ambos pares de *slices* dentro de un CLB no sólo soportan las funciones descritas, si no que también pueden funcionar como ROM (Read Only Memory) con datos inicializados al momento de configurar la FPGA. Las LUTs del lado izquierdo de cada CLB soportan además dos funciones adicionales: primero, es posible programarlas como RAM distribuida, lo que permite contar con espacios de memoria de 16 bits en cualquier parte de la topología de la FPGA. Segundo, es posible programar una de estas LUTs como un registro de desplazamiento de 16 bits, con lo que se pueden producir retardos de hasta 16 bits o combinaciones de varias LUTs pueden producirlos de cualquier largo de bits.

Bloques dedicados de memoria RAM

La Spartan III tiene 24 bloques de 18 Kbits de memoria RAM. El ancho del bus de datos versus el de direcciones (relación de aspecto) de cada bloque es configurable y se puede combinar varios de éstos para formar memorias más anchas o de mayor profundidad.

Tal como se muestra en la Figura 5, los bloques de RAM tienen una estructura de doble puerto. Dos puertos idénticos llamados A y B permiten acceso independiente al mismo rango de memoria, que tiene una capacidad máxima de 18.432 bits – o 16.384 cuando no se usan las líneas de paridad. Cada puerto tiene su propio set de líneas de control, de datos y de reloj para las operaciones síncronas de lectura y escritura. Estas operaciones tienen lugar de manera totalmente independiente en cada uno de los puertos.

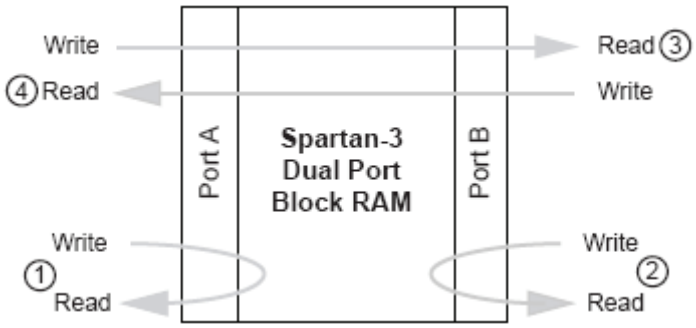


Figura 5: Diagrama de un bloque de RAM dedicado de la Spartan III

Multiplicadores dedicados

La Spartan III provee multiplicadores embebidos que aceptan palabras de 18 bits como entrada y entregan productos de 36 bits. Los buses de entrada de estos multiplicadores aceptan datos en complemento dos (tanto 18 bits con signo, como 17 bits sin signo). Para cada bloque de RAM hay un multiplicador inmediatamente colocado y conectado; dicha proximidad permite manejo eficiente de los datos.

Digital Clock Manager (DCM) y red de distribución de relojes

La Spartan III tiene 4 bloques para el control de todos los aspectos relacionados con la frecuencia, la fase y el *skew* de la red de relojes de la FPGA. Cada DCM tiene cuatro componentes funcionales: El Delay-Locked Loop (DLL), El Sintetizador Digital

de Frecuencia (DFS) y el Desplazador de fase (PS). Además incluye cierta lógica para *status*.

La Figura 6 muestra un diagrama de bloques de este elemento funcional de la FPGA.

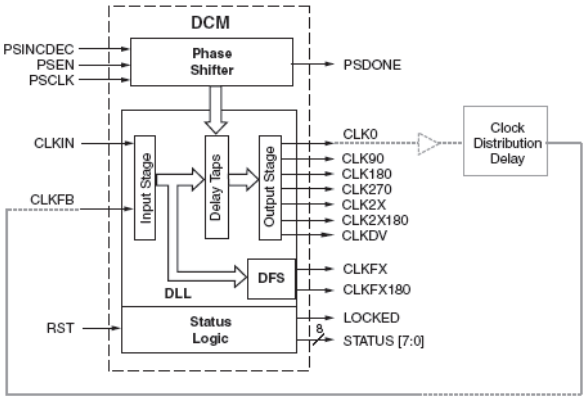


Figura 6: Diagrama de bloques de uno de los cuatro DCMs de la Spartan III

El DCM realiza tres funciones principales:

- Eliminación de *skew* de reloj: El concepto de *skew* describe el grado al cual las señales de reloj pueden, bajo circunstancias normales, desviarse del alineamiento de la fase cero. Ello ocurre cuando pequeñas diferencias en los retardos de las rutas causan que la señal de reloj llegue a diferentes puntos del circuito en tiempos diferentes. Este *skew* de reloj puede incrementar los requerimientos de *set-up time* y de *hold time*, lo que puede perjudicar el desempeño de aplicaciones de alta frecuencia. El DCM elimina el *skew* de reloj alineando la salida de la señal de reloj que genera con otra versión de la misma señal que es retroalimentada. Como resultado se establece una relación de cero desfase entre ambas señales.

- Síntesis de frecuencia: Provisto de una señal de reloj de entrada, el DCM puede generar diferentes relojes de salida. Ello se logra multiplicando y/o dividiendo la frecuencia del reloj de entrada.

- Corrimiento de fase: El DCM puede producir desfases controlados de la señal de reloj de entrada y producir con ello relojes de salida con diferentes fases.

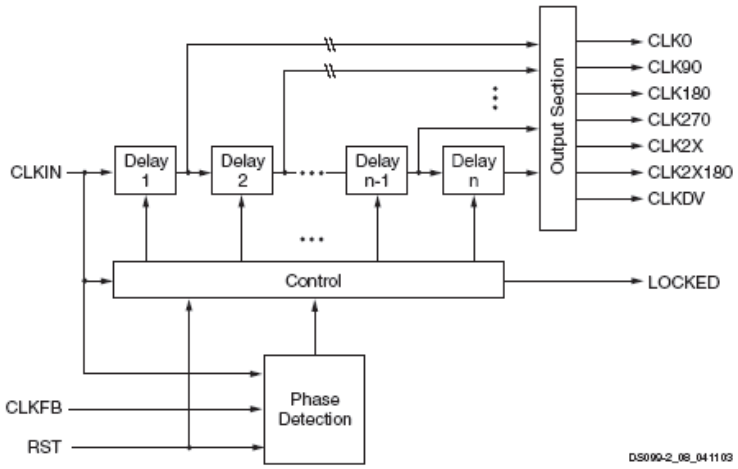


Figura 7: Diagrama funcional del Delay-Locked Loop (DLL)

El DLL tiene como principal función eliminar el *skew* de reloj. La ruta principal del DLL consiste en una etapa de entrada, seguida por una serie de elementos de retardo discreto o *taps*, los cuales conducen a una etapa de salida. Esta ruta, junto con lógica para detección de fase y control conforman un sistema completo con retroalimentación, tal como se muestra en la Figura 7.

La señal CLK0 es entregada a la red de distribución de señales de reloj de la FPGA, que sincroniza todo los registros del circuito que ha sido configurado. Estos registros pueden ser tanto internos como externos a la FPGA. Luego de pasar por dicha red, la señal de reloj retorna al DLL a través de la entrada CLKFB. El bloque de control del DLL mide el error de fase entre ambas señales, que es una medida del *skew* de reloj que toda la red introduce. El bloque de control activa el número apropiado de elementos de retardo para cancelar el *skew* de reloj. Una vez que se ha eliminado el

desfase, se eleva la señal LOCKED, que indica la puesta en fase del reloj con respecto a la retro alimentación.

Las señales de reloj tienen una red dedicada especial para su distribución. Esta red tiene ocho entradas globales, por medio de *buffers*. La red tiene baja capacitancia y produce muy bajo *skew* de reloj, lo que la hace adecuada para conducir señales de alta frecuencia. Tal como se muestra en la Figura8, las entradas GCLK0 a GCLK3 están puestas en la parte inferior de la oblea de la FPGA, mientras que las entradas GCLK4 a GCLK7 están colocadas en la parte superior. Se puede conducir cualquiera de dichas entradas hacia cada uno de los CLBs, por medio de las líneas principales, que se observan en negro grueso en la Figura8. Las líneas más delgadas representan líneas que conducen hacia los elementos síncronos de cada una de las *slices* de los CLBs.

Las entradas a la red se distribuyen a través de 4 multiplexores 2-1 a cada lado de la red, los que también conducen las señales provenientes de los DCMs. Con el propósito de minimizar la disipación de potencia dinámica en la red de distribución de relojes, el *software* de síntesis automáticamente deshabilita aquellas líneas que no son utilizadas en el diseño.

Esta red de distribución de señales de reloj es completamente independiente de la malla de interconexiones entre CLBs que se describe en el siguiente punto.

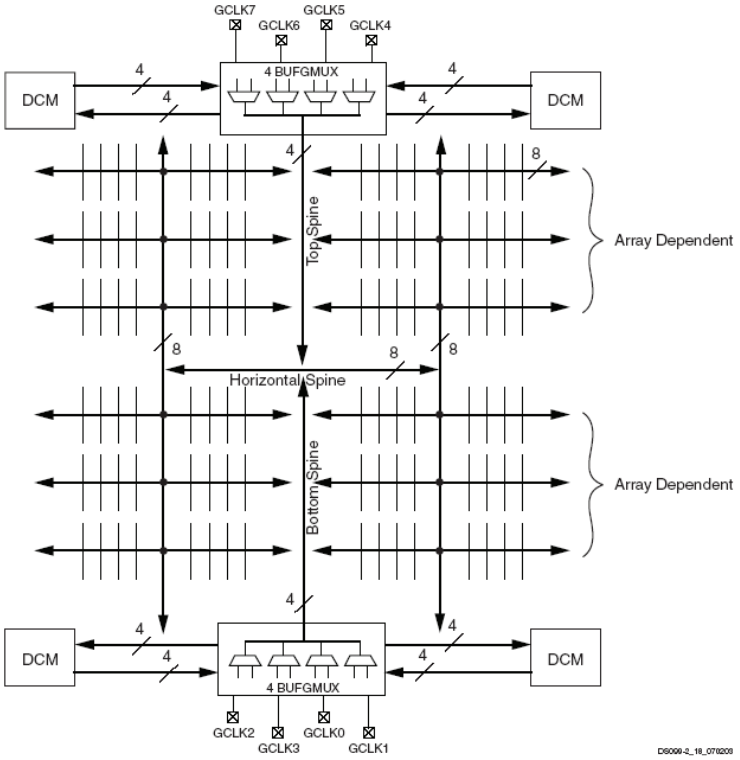


Figura 8: Red de distribución de señales de reloj de la Spartan III

Red de interconexiones de la FPGA

La red de interconexión conduce las señales entre varios elementos funcionales de la Spartan III. Hay cuatro tipos de interconexiones: *Long lines, Hex lines, Double lines y Direct lines.*

Long lines son aquellas que conectan una salida de cada seis CLBs (Figura9a). Debido a su baja capacitancia, estas líneas son adecuadas para conducir señales de alta frecuencia. Si las ocho entradas para las redes de reloj están ocupadas, estas líneas son adecuadas como alternativa. *Hex lines* son las que conectan una salida de cada tres CLBs (Figura9b). Son líneas que ofrecen mayor conectividad que las anteriores, pero un poco menos de capacidad en alta frecuencia. Las *Double lines* conectan todos los otros CLBs (Figura9c), lo que las hace conexiones más flexibles. Las *Direct lines* entregan conexiones directas de cada CLB hacia cada uno de sus

ocho vecinos (figura 9d). Estas líneas son usadas más a menudo para conducir una señal proveniente de un CLB de origen hacia una *Double line*, *Hex line* o *Long line* y desde esa ruta larga hacia otra *Direct line* que llevará la señal hacia el CLB de destino.

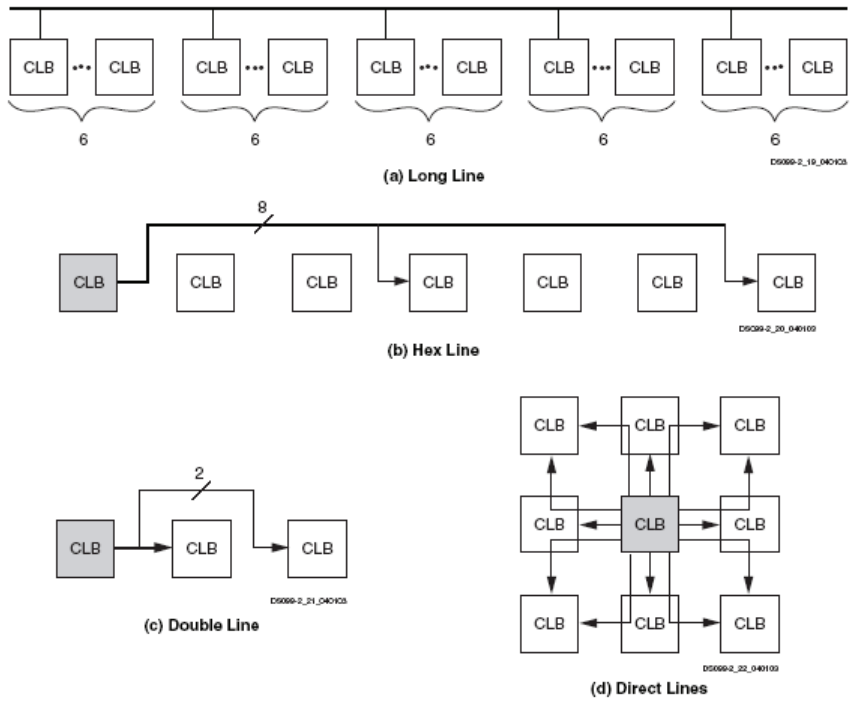
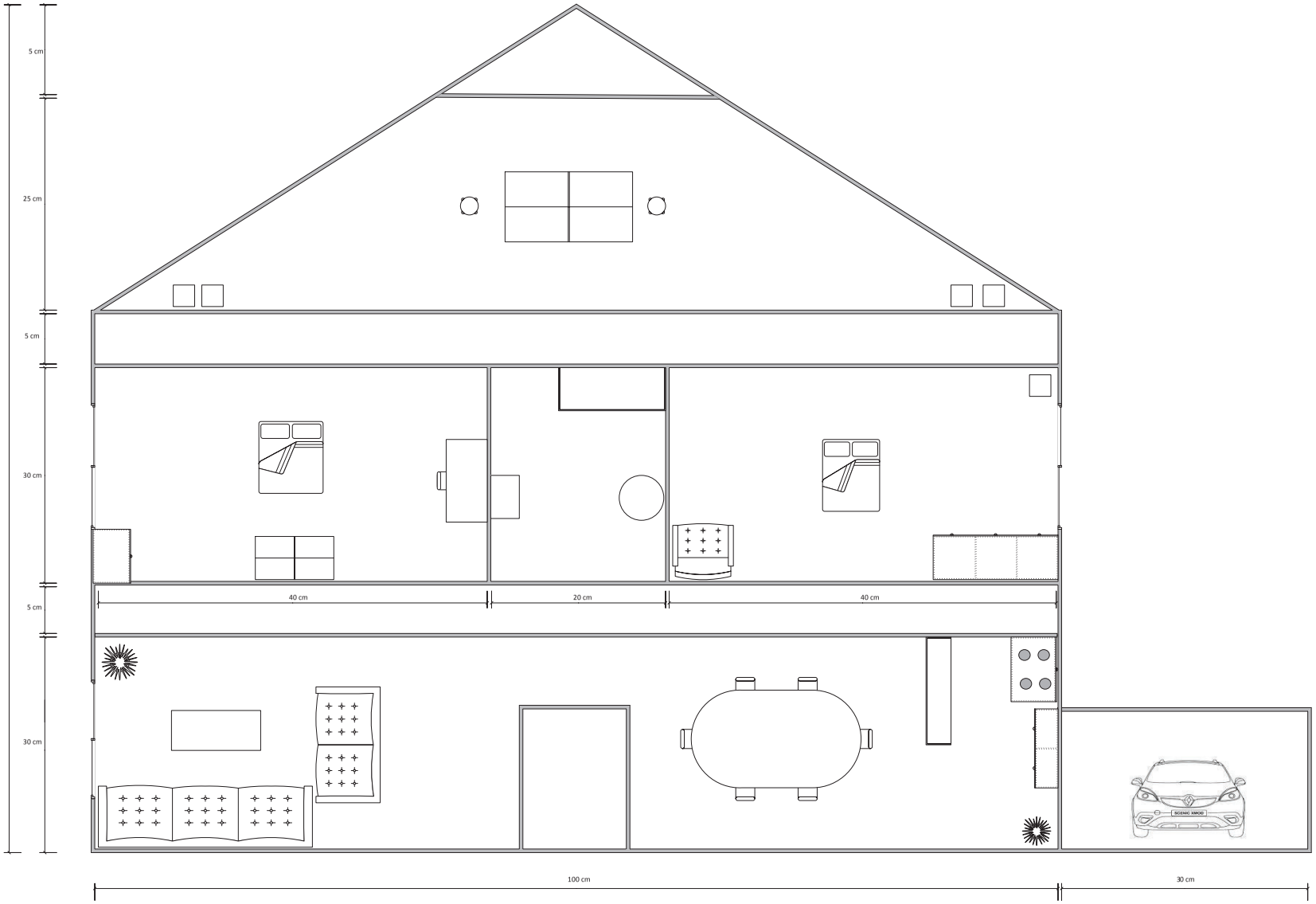


Figura 9: Tipos de interconexiones entre CLBs en la Spartan III

ANEXO 2: PLANO DEL PROTOTIPO DE VIVIENDA



ANEXO 3: HOJA DE DATOS DE DIODO 1N4005 Y 1N4007

DIODES INCORPORATED

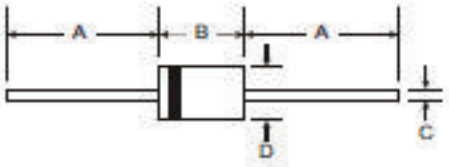
1N4001/L - 1N4007/L

1.0A RECTIFIER

SPICE MODELS: 1N4001 1N4002 1N4003 1N4004 1N4005 1N4006 1N4007

Features

- Diffused Junction
- High Current Capability and Low Forward Voltage Drop
- Surge Overload Rating to 30A Peak
- Low Reverse Leakage Current
- Plastic Material: UL Flammability Classification Rating 94V-0



Mechanical Data

- Case: Molded Plastic
- Terminals: Plated Leads Solderable per MIL-STD-202, Method 208
- Polarity: Cathode Band
- Weight: DO-41 0.30 grams (approx)
  A-405 0.20 grams (approx)
- Mounting Position: Any
- Marking: Type Number

	DO-41 Plastic		A-405	
Dim	Min	Max	Min	Max
A	25.40	—	25.40	—
B	4.06	5.21	4.10	5.20
C	0.71	0.864	0.53	0.64
D	2.00	2.72	2.00	2.70
All Dimensions in mm				

"L" Suffix Designates A-405 Package
No Suffix Designates DO-41 Package

Maximum Ratings and Electrical Characteristics @ T_A = 25°C unless otherwise specified

Single phase, half wave, 60Hz, resistive or inductive load.
For capacitive load, derate current by 20%.

Characteristic	Symbol	1N 4001/L	1N 4002/L	1N 4003/L	1N 4004/L	1N 4005/L	1N 4006/L	1N 4007/L	Unit
Peak Repetitive Reverse Voltage Working Peak Reverse Voltage DC Blocking Voltage	V_{RRM} V_{RWM} V_R	50	100	200	400	600	800	1000	V
RMS Reverse Voltage	$V_{R(RMS)}$	35	70	140	280	420	560	700	V
Average Rectified Output Current (Note 1) @ T_A = 75°C	I_O	1.0							A
Non-Repetitive Peak Forward Surge Current 8.3ms single half sine-wave superimposed on rated load (JEDEC Method)	I_{FSM}	30							A
Forward Voltage @ I_F = 1.0A	V_{FM}	1.0							V
Peak Reverse Current @ T_A = 25°C at Rated DC Blocking Voltage @ T_A = 100°C	I_{RM}	5.0 50							μA
Typical Junction Capacitance (Note 2)	C_j	15				8			pF
Typical Thermal Resistance Junction to Ambient	$R_{\theta JA}$	100							K/W
Maximum DC Blocking Voltage Temperature	T_A	+150							°C
Operating and Storage Temperature Range (Note 3)	T_j T_{STG}	-65 to +175							°C

Notes: 1. Leads maintained at ambient temperature at a distance of 9.5mm from the case.
2. Measured at 1. MHz and applied reverse voltage of 4.0V DC.
3. JEDEC Value

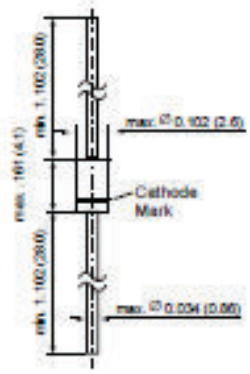
© Diodes Incorporated

ANEXO 4: HOJA DE DATOS DE DIODO ZENER 1N4733 Y 1N4736A

1N4728 THRU 1N4764

ZENER DIODES

DO-41 Glass



Dimensions in inches and (millimeters)

FEATURES

- Silicon Planar Power Zener Diodes
- For use in stabilizing and clipping circuits with high power rating.

- Standard Zener voltage tolerance is ±10%. Add suffix "A" for ±5% tolerance. Other Zener voltages and tolerances are available upon request.
- These diodes are also available in the MELF case with type designation ZM4728 thru ZM4764

MECHANICAL DATA

Case: DO-41 Glass Case
Weight: approx. 0.35 g

MAXIMUM RATINGS

Ratings at 25°C ambient temperature unless otherwise specified.

	SYMBOL	VALUE	UNIT
Zener Current (see Table "Characteristics")			
Power Dissipation at T_{amb} = 25°C	P_{tot}	1.0[1]	Watts
Junction Temperature	T_j	175	°C
Storage Temperature Range	T_S	– 65 to +175	°C

Characteristics at Tamb = 25 °C

	SYMBOL	MIN.	TYP.	MAX.	UNIT
Thermal Resistance Junction to Ambient Air	R_{thJA}	–	–	170[1]	°C/W
Forward Voltage at I_F = 200 mA	V_F	–	–	1.2	Volts

NOTES:
(1) Valid provided that electrodes at a distance of 10mm from case are kept at ambient temperature

GENERAL SEMICONDUCTOR®

12/3/98

1N4728 THRU 1N4764

ELECTRICAL CHARACTERISTICS

Ratings at 25°C ambient temperature unless otherwise specified.

Type	Nominal Zener voltage[2] at I_{ZT} V_Z V	Test current I_{ZT} mA	Maximum Zener impedance[1] Z_{ZT} at I_{ZT} Ω	Z_{ZK} Ω	at I_{ZK} mA	Maximum reverse leakage current I_R μA	at V_R V	Surge current at T_A = 25°C I_R mA	Maximum regulator current[3] I_{ZM} mA
1N4728	3.3	76	10	400	1.0	100	1	1380	276
1N4729	3.6	69	10	400	1.0	100	1	1260	252
1N4730	3.9	64	9	400	1.0	50	1	1190	234
1N4731	4.3	58	9	400	1.0	10	1	1070	217
1N4733	5.1	49	7	550	1.0	10	1	890	178
1N4735	6.2	41	2	700	1.0	10	3	730	146
1N4737	7.5	34	4.0	700	0.5	10	5	605	121
1N4738	8.2	31	4.5	700	0.5	10	6	550	110
1N4739	9.1	28	5.0	700	0.5	10	7	500	100
1N4740	10	25	7	700	0.25	10	7.6	454	91
1N4741	11	23	8	700	0.25	5	8.4	414	83
1N4742	12	21	9	700	0.25	5	9.1	380	76
1N4743	13	19	10	700	0.25	5	9.9	344	69
1N4744	15	17	14	700	0.25	5	11.4	304	61
1N4745	16	15.5	16	700	0.25	5	12.2	285	57
1N4746	18	14	20	750	0.25	5	13.7	250	50
1N4747	20	12.5	22	750	0.25	5	15.2	225	45
1N4748	22	11.5	23	750	0.25	5	16.7	205	41
1N4749	24	10.5	25	750	0.25	5	18.2	190	38
1N4750	27	9.5	35	750	0.25	5	20.6	170	34
1N4751	30	8.5	40	1000	0.25	5	22.8	150	30
1N4752	33	7.5	45	1000	0.25	5	25.1	135	27
1N4753	36	7.0	50	1000	0.25	5	27.4	125	25
1N4754	39	6.5	60	1000	0.25	5	29.7	115	23
1N4755	43	6.0	70	1500	0.25	5	32.7	110	22
1N4756	47	5.5	80	1500	0.25	5	35.8	95	19
1N4757	51	5.0	95	1500	0.25	5	38.8	90	18
1N4758	56	4.5	110	2000	0.25	5	42.6	80	16
1N4759	62	4.0	125	2000	0.25	5	47.1	70	14
1N4760	68	3.7	150	2000	0.25	5	51.7	65	13
1N4761	75	3.3	175	2000	0.25	5	56.0	60	12
1N4762	82	3.0	200	3000	0.25	5	62.2	55	11
1N4763	91	2.8	250	3000	0.25	5	69.2	50	10
1N4764	100	2.5	350	3000	0.25	5	76.0	45	9

NOTES:

(1) The Zener impedance is derived from the 1KHz AC voltage which results when an AC current having an RMS value equal to 10% of the Zener current (I_{ZT} or I_{ZK}) is superimposed on I_{ZT} or I_{ZK}. Zener impedance is measured at two points to insure a sharp knee on the breakdown curve and to eliminate unstable units.

(2) Valid provided that electrodes at a distance of 10mm from case are kept at ambient temperature.

(3) Measured under thermal equilibrium and DC test conditions.

GENERAL SEMICONDUCTOR

ANEXO 5: HOJA DE DATOS DEL REGULADOR DE VOLTAJE LM7805

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

MC78XX/LM78XX/MC78XXA

3-Terminal 1A Positive Voltage Regulator

Features

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

TO-220

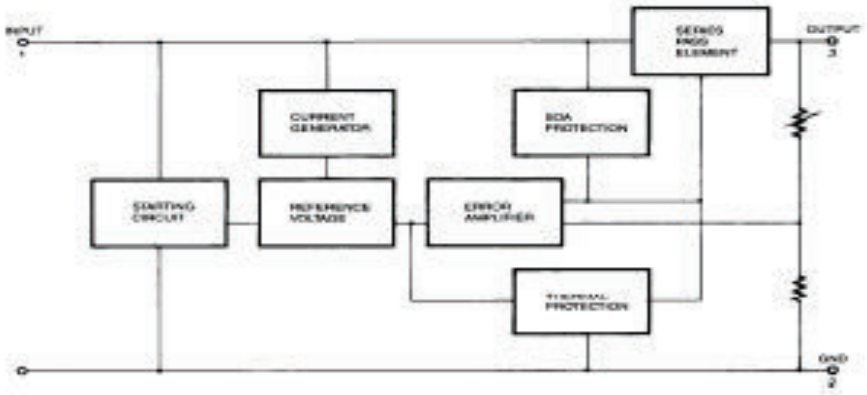
1

D-PAK

1

1. Input 2. GND 3. Output

Internal Block Digram



Rev. 1.0.1

©2001 Fairchild Semiconductor Corporation

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	°C/W
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	°C/W
Operating Temperature Range	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics (MC7805/LM7805)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I = 10V, C_I= 0.33μF, C_O= 0.1μF, unless otherwise specified)

Parameter	Symbol	Conditions		MC7805/LM7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ I_O ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	μV/Vo
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 °C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_D due to heating effects must be taken into account separately. Pulse testing with low duty is used.

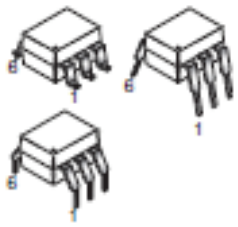
ANEXO 6: HOJA DE DATOS DEL OPTOACOPLADOR 4N25

FAIRCHILD
SEMICONDUCTOR®

GENERAL PURPOSE 6-PIN PHOTOTRANSISTOR OPTOCOUPLERS

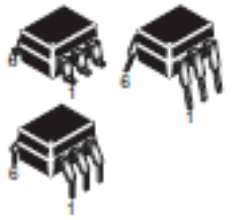
4N25 4N37	4N26 H11A1	4N27 H11A2	4N28 H11A3	4N35 H11A4	4N36 H11A5

WHITE PACKAGE (-M SUFFIX)



SCHEMATIC

BLACK PACKAGE (NO -M SUFFIX)



DESCRIPTION

The general purpose optocouplers consist of a gallium arsenide infrared emitting diode driving a silicon phototransistor in a 6-pin dual in-line package.

FEATURES

- Also available in white package by specifying -M suffix, eg. 4N25-M
- UL recognized (File # E90700)
- VDE recognized (File # 94766)
  - Add option V for white package (e.g., 4N25V-M)
  - Add option 300 for black package (e.g., 4N25.300)

APPLICATIONS

- Power supply regulators
- Digital logic inputs
- Microprocessor inputs

© 2002 Fairchild Semiconductor Corporation

ANEXO 7: HOJA DE DATOS DEL MICROCONTROLADOR PIC16F877A

MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

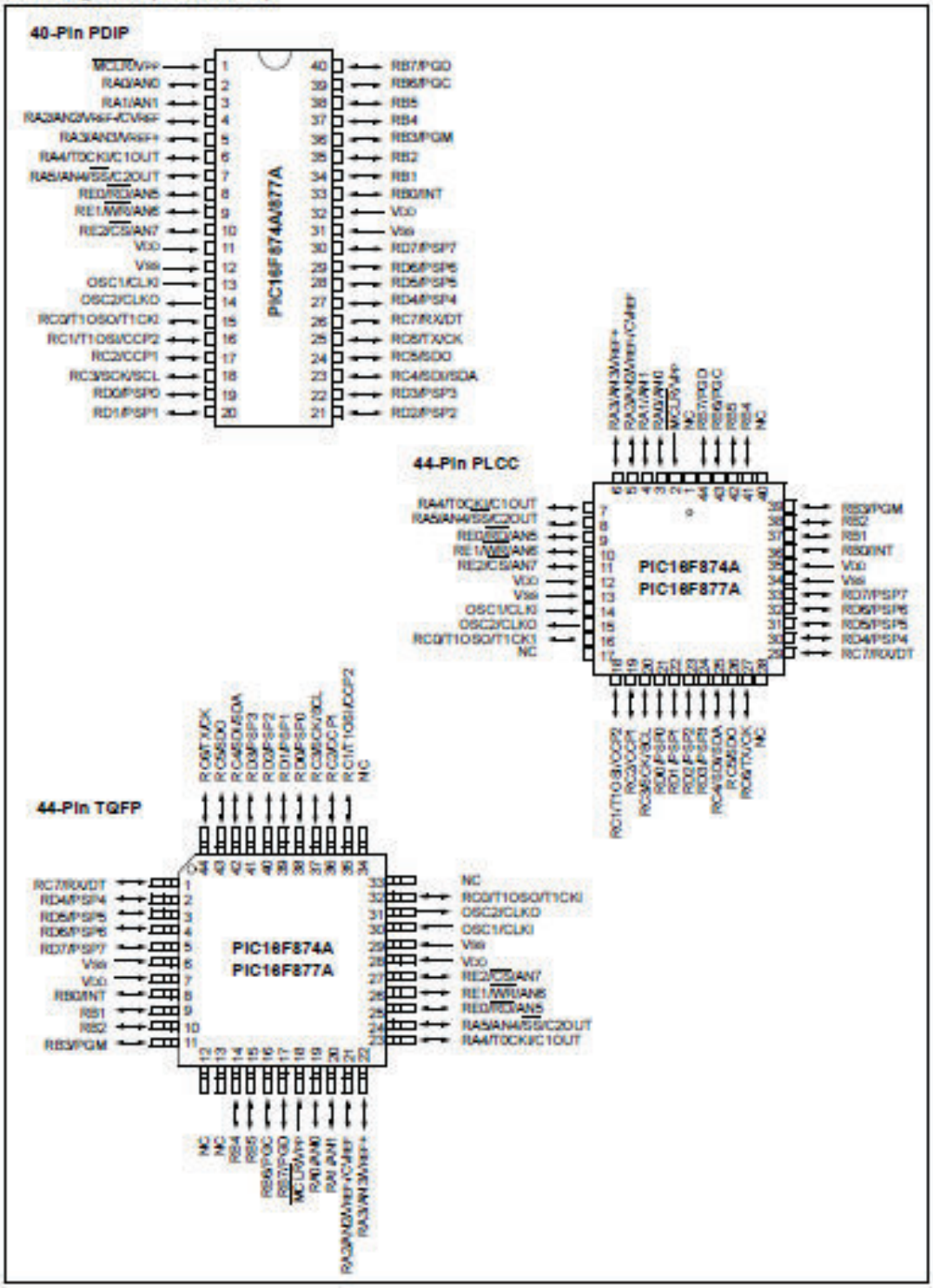
CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

© 2001-2013 Microchip Technology Inc.

Pin Diagrams (Continued)



ANEXO 8: HOJA DE DATOS DEL INVERSOR CD4069

National Semiconductor

February 1988

CD4069UBM/CD4069UBC Inverter Circuits

General Description

The CD4069UB consists of six inverter circuits and is manufactured using complementary MOS (CMOS) to achieve wide power supply operating range, low power consumption, high noise immunity, and symmetric controlled rise and fall times.

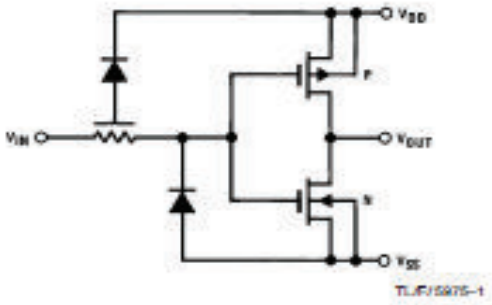
This device is intended for all general purpose inverter applications where the special characteristics of the MM74C901, MM74C903, MM74C907, and CD4049A Hex Inverter/Buffers are not required. In those applications requiring larger noise immunity the MM74C14 or MM74C914 Hex Schmitt Trigger is suggested.

All inputs are protected from damage due to static discharge by diode clamps to V_{DD} and V_{SS}.

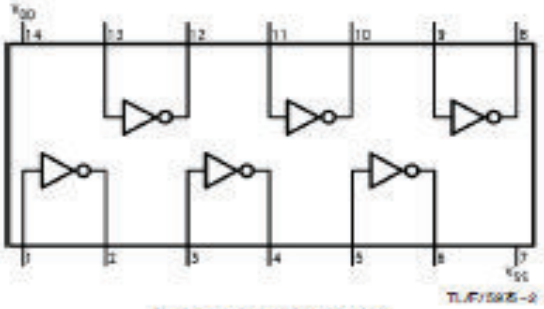
Features

■ Wide supply voltage range	3.0V to 15V
■ High noise immunity	0.45 V_{DD} typ.
■ Low power TTL compatibility	Fan out of 2 driving 74L or 1 driving 74LS
■ Equivalent to MM54C04/MM74C04	

Schematic and Connection Diagram

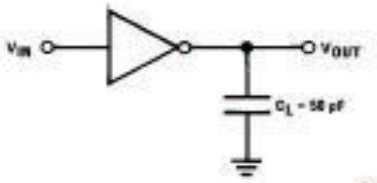


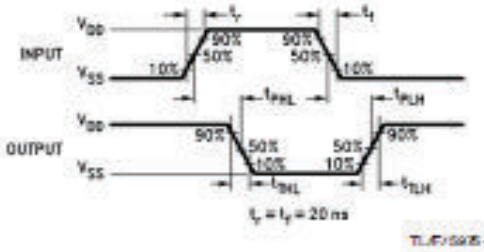
Dual-In-Line Package



Order Number CD4069UB

AC Test Circuits and Switching Time Waveforms





© 1988 National Semiconductor Corporation TL/F/5975

RRD-B30M105/Printed in U.S.A.

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

DC Supply Voltage (V_{DD})	−0.5V to +18 V_{DC}
Input Voltage (V_{IN})	−0.5V to V_{DD} +0.5 V_{DC}
Storage Temperature Range (T_S)	−65°C to +150°C
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Lead Temperature (T_L)	
(Soldering, 10 seconds)	260°C

Recommended Operating Conditions (Note 2)

DC Supply Voltage (V_{DD})	3V to 15V_{DC}
Input Voltage (V_{IN})	0V to V_{DD} V_{DC}
Operating Temperature Range (T_A)	
CD4069UBM	−55°C to +125°C
CD4069UBC	−40°C to +85°C

DC Electrical Characteristics CD4069UBM (Note 2)

Symbol	Parameter	Conditions	−55°C		+25°C			+125°C		Units
			Min	Max	Min	Typ	Max	Min	Max	
I_{DD}	Quiescent Device Current	V_{DD} = 5V, V_{IN} = V_{DD} or V_{SS}		0.25			0.25		7.5	µA
		V_{DD} = 10V, V_{IN} = V_{DD} or V_{SS}		0.5			0.5		15	µA
		V_{DD} = 15V, V_{IN} = V_{DD} or V_{SS}		1.0			1.0		30	µA
V_{OL}	Low Level Output Voltage	$\|I_O\|$ < 1 µA								
		V_{DD} = 5V		0.05		0	0.05		0.05	V
		V_{DD} = 10V		0.05		0	0.05		0.05	V
		V_{DD} = 15V		0.05		0	0.05		0.05	V
V_{OH}	High Level Output Voltage	$\|I_O\|$ < 1 µA								
		V_{DD} = 5V	4.95		4.95	5		4.95		V
		V_{DD} = 10V	9.95		9.95	10		9.95		V
		V_{DD} = 15V	14.95		14.95	15		14.95		V
V_{IL}	Low Level Input Voltage	$\|I_O\|$ < 1 µA								
		V_{DD} = 5V, V_O = 4.5V		1.0			1.0		1.0	V
		V_{DD} = 10V, V_O = 9V		2.0			2.0		2.0	V
		V_{DD} = 15V, V_O = 13.5V		3.0			3.0		3.0	V
V_{IH}	High Level Input Voltage	$\|I_O\|$ < 1 µA								
		V_{DD} = 5V, V_O = 0.5V	4.0		4.0			4.0		V
		V_{DD} = 10V, V_O = 1V	8.0		8.0			8.0		V
		V_{DD} = 15V, V_O = 1.5V	12.0		12.0			12.0		V
I_{OL}	Low Level Output Current (Note 3)	V_{DD} = 5V, V_O = 0.4V	0.64		0.51	0.88		0.36		mA
		V_{DD} = 10V, V_O = 0.5V	1.6		1.3	2.25		0.9		mA
		V_{DD} = 15V, V_O = 1.5V	4.2		3.4	8.8		2.4		mA
I_{OH}	High Level Output Current (Note 3)	V_{DD} = 5V, V_O = 4.6V	−0.64		−0.51	−0.88		−0.36		mA
		V_{DD} = 10V, V_O = 9.5V	−1.6		−1.3	−2.25		−0.9		mA
		V_{DD} = 15V, V_O = 13.5V	−4.2		−3.4	−8.8		−2.4		mA
I_{IN}	Input Current	V_{DD} = 15V, V_{IN} = 0V		−0.10		-10^{-5}	−0.10		−1.0	µA
		V_{DD} = 15V, V_{IN} = 15V		0.10		10^{-5}	0.10		1.0	µA

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the devices should be operated at these limits. The table of "Recommended Operating Conditions" and "Electrical Characteristics" provides conditions for actual device operation.

Note 2: V_{SS} = 0V unless otherwise specified.

Note 3: I_{OH} and I_{OL} are tested one output at a time.

DC Electrical Characteristics CD4069UBC (Note 2)

Symbol	Parameter	Conditions	−40°C		+25°C			+85°C		Units
			Min	Max	Min	Typ	Max	Min	Max	
I_{DD}	Quiescent Device Current	V_{DD} = 5V, V_{IN} = V_{DD} or V_{SS}		1.0			1.0		7.5	μA
		V_{DD} = 10V, V_{IN} = V_{DD} or V_{SS}		2.0			2.0		15	μA
		V_{DD} = 15V, V_{IN} = V_{DD} or V_{SS}		4.0			4.0		30	μA
V_{OL}	Low Level Output Voltage	$\|I_O\|$ < 1 μA								
		V_{DD} = 5V		0.05		0	0.05		0.05	V
		V_{DD} = 10V		0.05		0	0.05		0.05	V
		V_{DD} = 15V		0.05		0	0.05		0.05	V
V_{OH}	High Level Output Voltage	$\|I_O\|$ < 1 μA								
		V_{DD} = 5V	4.95		4.95			4.95		V
		V_{DD} = 10V	9.95		9.95			9.95		V
		V_{DD} = 15V	14.95		14.95			14.95		V
V_{IL}	Low Level Input Voltage	$\|I_O\|$ < 1 μA								
		V_{DD} = 5V, V_O = 4.5V		1.0			1.0		1.0	V
		V_{DD} = 10V, V_O = 9V		2.0			2.0		2.0	V
		V_{DD} = 15V, V_O = 13.5V		3.0			3.0		3.0	V
V_{IH}	High Level Input Voltage	$\|I_O\|$ < 1 μA								
		V_{DD} = 5V, V_O = 0.5V	4.0		4.0			4.0		V
		V_{DD} = 10V, V_O = 1V	8.0		8.0			8.0		V
		V_{DD} = 15V, V_O = 1.5V	12.0		12.0			12.0		V
I_{OL}	Low Level Output Current (Note 3)	V_{DD} = 5V, V_O = 0.4V	0.52		0.44	0.88		0.36		mA
		V_{DD} = 10V, V_O = 0.5V	1.3		1.1	2.25		0.9		mA
		V_{DD} = 15V, V_O = 1.5V	3.6		3.0	8.8		2.4		mA
I_{OH}	High Level Output Current (Note 3)	V_{DD} = 5V, V_O = 4.6V	−0.52		−0.44	−0.88		−0.36		mA
		V_{DD} = 10V, V_O = 9.5V	−1.3		−1.1	−2.25		−0.9		mA
		V_{DD} = 15V, V_O = 13.5V	−3.6		−3.0	−8.8		−2.4		mA
I_{IN}	Input Current	V_{DD} = 15V, V_{IN} = 0V		−0.30		-10^{-5}	−0.30		−1.0	μA
		V_{DD} = 15V, V_{IN} = 15V		0.30		10^{-5}	0.30		1.0	μA

AC Electrical Characteristics*

T_A = 25°C, C_L = 50 pF, R_L = 200 kΩ, t_r and t_f ≤ 20 ns, unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{PHL} or t_{PLH}	Propagation Delay Time from Input to Output	V_{DD} = 5V		50	90	ns
		V_{DD} = 10V		30	60	ns
		V_{DD} = 15V		25	50	ns
t_{THL} or t_{TLH}	Transition Time	V_{DD} = 5V		80	150	ns
		V_{DD} = 10V		50	100	ns
		V_{DD} = 15V		40	80	ns
C_{IN}	Average Input Capacitance	Any Gate		6	15	pF
C_{PD}	Power Dissipation Capacitance	Any Gate (Note 4)		12		pF

*AC Parameters are guaranteed by DC correlated testing.

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the devices should be operated at these limits. The table of "Recommended Operating Conditions" and "Electrical Characteristics" provides conditions for actual device operation.

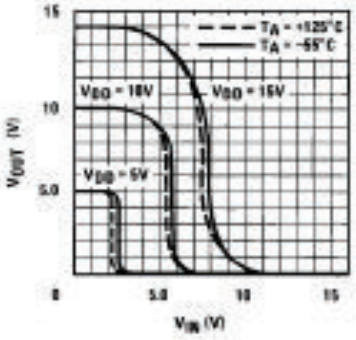
Note 2: V_{SS} = 0V unless otherwise specified.

Note 3: I_{OH} and I_{OL} are tested one output at a time.

Note 4: C_{PD} determines the no load AC power consumption of any CMOS device. For complete explanation, see 54C/74C Family Characteristics application note—AN-90.

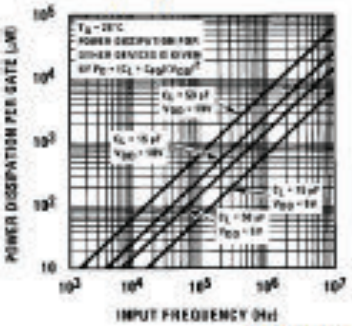
Typical Performance Characteristics

Gate Transfer Characteristics



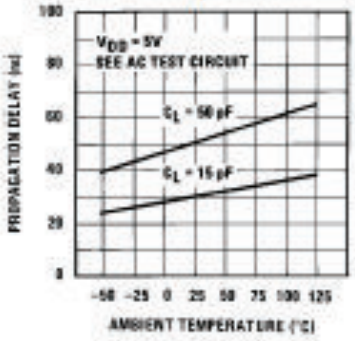
TL/F/5975–5

Power Dissipation vs Frequency



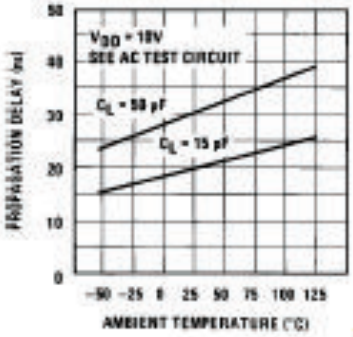
TL/F/5975–6

Propagation Delay vs Ambient Temperature



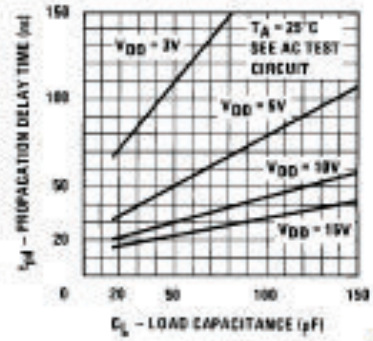
TL/F/5975–7

Propagation Delay vs Ambient Temperature



TL/F/5975–8

Propagation Delay Time vs Load Capacitance



TL/F/5975–9

ANEXO 9: HOJA DE DATOS DEL DIODO 1N4148

High-speed diodes 1N4148; 1N4448

FEATURES

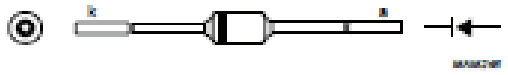
- Hermetically sealed leaded glass SOD27 (DO-35) package
- High switching speed: max. 4 ns
- General application
- Continuous reverse voltage: max. 75 V
- Repetitive peak reverse voltage: max. 100 V
- Repetitive peak forward current: max. 450 mA.

APPLICATIONS

- High-speed switching.

DESCRIPTION

The 1N4148 and 1N4448 are high-speed switching diodes fabricated in planar technology, and encapsulated in hermetically sealed leaded glass SOD27 (DO-35) packages.



The diodes are type branded.

Fig.1 Simplified outline (SOD27; DO-35) and symbol.

LIMITING VALUES

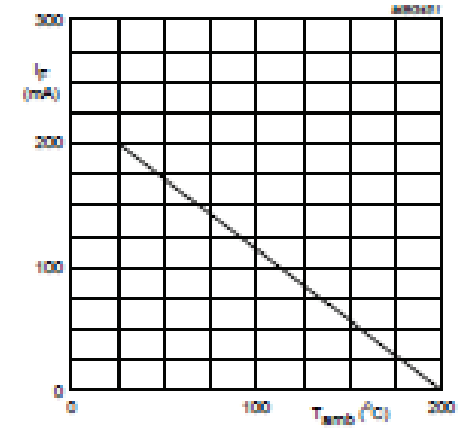
In accordance with the Absolute Maximum Rating System (IEC 60134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{RRM}	repetitive peak reverse voltage		–	100	V
V_R	continuous reverse voltage		–	75	V
I_F	continuous forward current	see Fig.2; note 1	–	200	mA
I_{FRM}	repetitive peak forward current		–	450	mA
I_{FSM}	non-repetitive peak forward current	square wave; T_j = 25 °C prior to surge; see Fig.4			
		t = 1 μs	–	4	A
		t = 1 ms	–	1	A
		t = 1 s	–	0.5	A
P_{tot}	total power dissipation	T_{amb} = 25 °C; note 1	–	500	mW
T_{stg}	storage temperature		−65	+200	°C
T_j	junction temperature		–	200	°C

Note

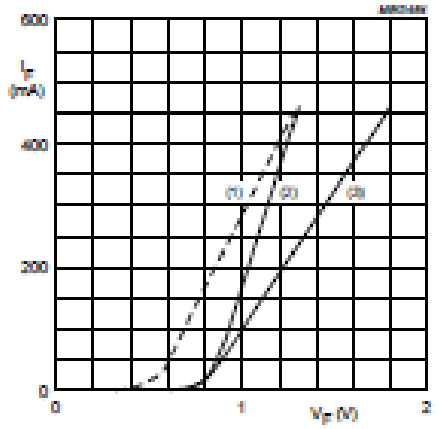
1. Device mounted on an FR4 printed circuit-board; lead length 10 mm.

GRAPHICAL DATA



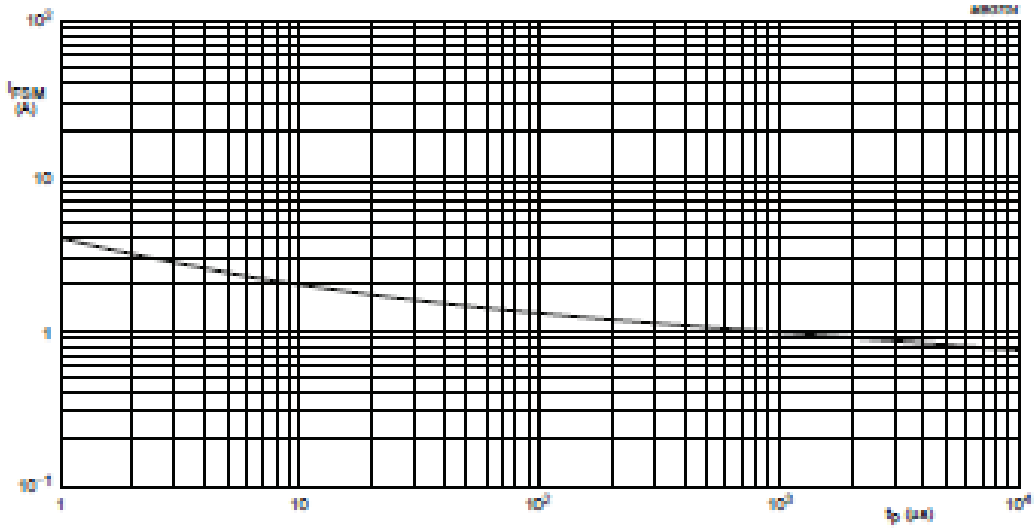
Device mounted on an FR4 printed-circuit board; lead length 10 mm.

Fig.2 Maximum permissible continuous forward current as a function of ambient temperature.



(1) T_j = 175 °C; typical values.
(2) T_j = 25 °C; typical values.
(3) T_j = 25 °C; maximum values.

Fig.3 Forward current as a function of forward voltage.



Based on square wave currents.
T_j = 25 °C prior to surge.

Fig.4 Maximum permissible non-repetitive peak forward current as a function of pulse duration.

ANEXO 10: HOJA DE DATOS DEL MICROCONTROLADOR ATMEGA 8535

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 130 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 8K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 512 Bytes Internal SRAM
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels for TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x for TQFP Package Only
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, 44-lead PLCC, and 44-pad MLF
- Operating Voltages
  - 2.7 - 6.6V for ATmega8535L
  - 4.5 - 6.6V for ATmega8535
- Speed Grades
  - 0 - 8 MHz for ATmega8535L
  - 0 - 16 MHz for ATmega8535

ATMEL

8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

ATmega8535
ATmega8535L

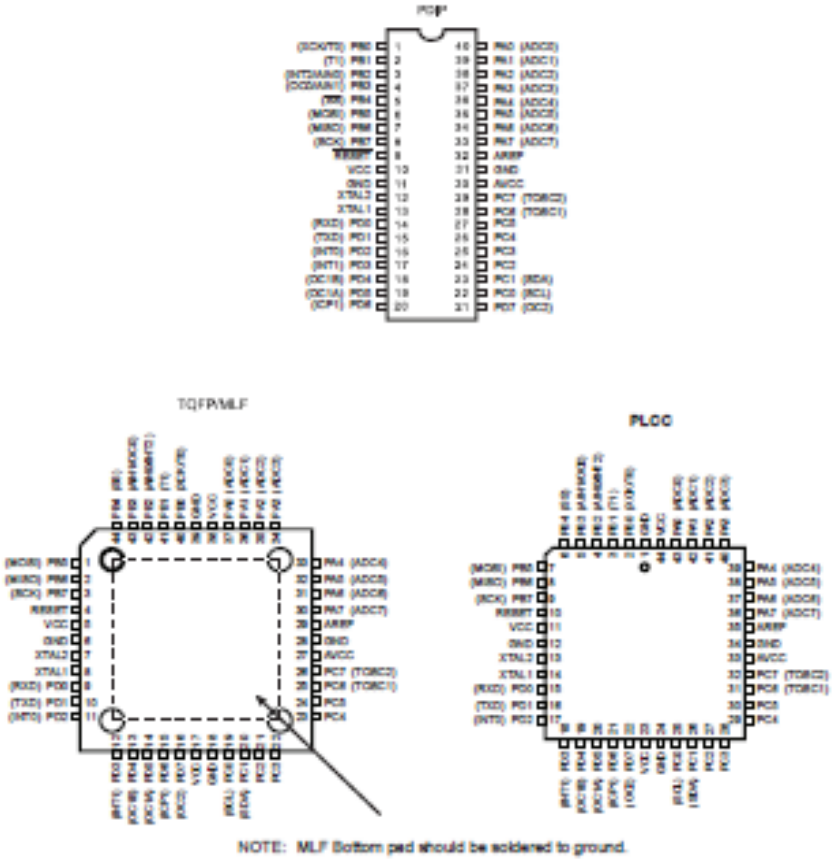
Preliminary Summary

Rev. 2502ES–AVR–12/03

Note: This is a summary document. A complete document is available on our Web site at www.atmel.com.

Pin Configurations

Figure 1. Pinout ATmega8535



NOTE: MLF Bottom pad should be soldered to ground.

Disclaimer

Typical values contained in this data sheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

ANEXO 11: HOJA DE DATOS DEL TRIAC BT136

GENERAL DESCRIPTION

Glass passivated triacs in a plastic envelope, intended for use in applications requiring high bidirectional transient and blocking voltage capability and high thermal cycling performance. Typical applications include motor control, industrial and domestic lighting, heating and static switching.

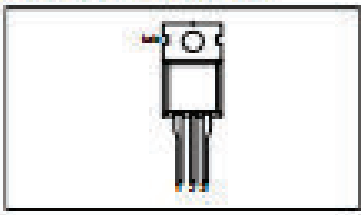
QUICK REFERENCE DATA

SYMBOL	PARAMETER	MAX.	MAX.	MAX.	UNIT
	BT136-	500	600	800	
	BT136-	500F	600F	800F	
	BT136-	500G	600G	800G	
V_{DRM}	Repetitive peak off-state voltages	500	600	800	V
$I_{T(RMS)}$	RMS on-state current	4	4	4	A
I_{TSM}	Non-repetitive peak on-state current	25	25	25	A

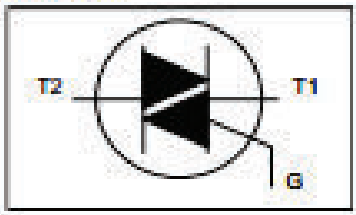
PINNING - TO220AB

PIN	DESCRIPTION
1	main terminal 1
2	main terminal 2
3	gate
tab	main terminal 2

PIN CONFIGURATION



SYMBOL



LIMITING VALUES

Limiting values in accordance with the Absolute Maximum System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS		MIN.	MAX.			UNIT
V_{DRM}	Repetitive peak off-state voltages			-	-500 500[1]	-600 600[1]	-800 800	V
$I_{T(RMS)}$	RMS on-state current	full sine wave; $T_{mb} \leq 107$ °C		-	4			A
I_{TSM}	Non-repetitive peak on-state current	full sine wave; $T_j = 25$ °C prior to surge						
		t = 20 ms		-	25			A
		t = 16.7 ms		-	27			A
I^2t	I^2t for fusing	t = 10 ms		-	3.1			A^2s
dI_T/dt	Repetitive rate of rise of on-state current after triggering	I_{TM} = 6 A; I_G = 0.2 A; dI_G/dt = 0.2 A/µs						
			T2+ G+	-	50			A/µs
			T2+ G-	-	50			A/µs
			T2- G-	-	50			A/µs
			T2- G+	-	10			A/µs
I_{GM}	Peak gate current			-	2			A
V_{GM}	Peak gate voltage			-	5			V
P_{GM}	Peak gate power			-	5			W
$P_{G(AV)}$	Average gate power	over any 20 ms period		-	0.5			W
T_{stg}	Storage temperature			-40	150			°C
T_j	Operating junction temperature			-	125			°C

1 Although not recommended, off-state voltages up to 800V may be applied without damage, but the triac may switch to the on-state. The rate of rise of current should not exceed 3 A/µs.

ANEXO 12: HOJA DE DATOS DEL OPTOAISLADOR TRIAC MOC3021

MOTOROLA
SEMICONDUCTOR TECHNICAL DATA

Order this document by MOC3020/D

MOC3021
[IFT = 15 mA Max]
MOC3022
[IFT = 10 mA Max]
MOC3023*
[IFT = 5 mA Max]

*Motorola Preferred Device

6-Pin DIP Random-Phase Optoisolators Triac Driver Output (400 Volts Peak)

The MOC3020 Series consists of gallium arsenide infrared emitting diodes, optically coupled to a silicon bilateral switch.

- *To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.*

They are designed for applications requiring isolated triac triggering.

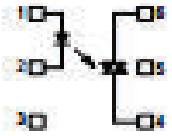
Recommended for 115/240 Vac(rms) Applications:

- Solenoid/Valve Controls
- Lamp Ballasts
- Interfacing Microprocessors to 115 Vac Peripherals
- Motor Controls
- Static ac Power Switch
- Solid State Relays
- Incandescent Lamp Dimmers

STYLE 6 PLASTIC

STANDARD THRU HOLE
CASE 730A-04

SCHEMATIC



1. ANODE
2. CATHODE
3. NC
4. MAIN TERMINAL
5. SUBSTRATE
   DO NOT CONNECT
6. MAIN TERMINAL

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INFRARED EMITTING DIODE			
Reverse Voltage	V_R	3	Volts
Forward Current — Continuous	I_F	60	mA
Total Power Dissipation @ T_A = 25°C Negligible Power in Triac Driver Derate above 25°C	P_D	100 1.33	mW mW/°C
OUTPUT DRIVER			
Off-State Output Terminal Voltage	V_{DRM}	400	Volts
Peak Repetitive Surge Current (PW = 1 ms, 120 pps)	I_{TSM}	1	A
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	300 4	mW mW/°C
TOTAL DEVICE			
Isolation Surge Voltage[1] (Peak ac Voltage, 60 Hz, 1 Second Duration)	V_{ISO}	7500	Vac(pk)
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	330 4.4	mW mW/°C
Junction Temperature Range	T_J	−40 to +100	°C
Ambient Operating Temperature Range[2]	T_A	−40 to +85	°C
Storage Temperature Range[2]	T_{stg}	−40 to +150	°C
Soldering Temperature (10 s)	T_L	260	°C

1. Isolation surge voltage, V_{ISO}, is an internal device dielectric breakdown rating. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.
2. Refer to Quality and Reliability Section in Opto Data Book for information on test conditions.

Preferred devices are Motorola recommended choices for future use and best overall value.
GlobalOptoisolator is a trademark of Motorola, Inc.

REV 1

© Motorola, Inc. 1996

MOTOROLA

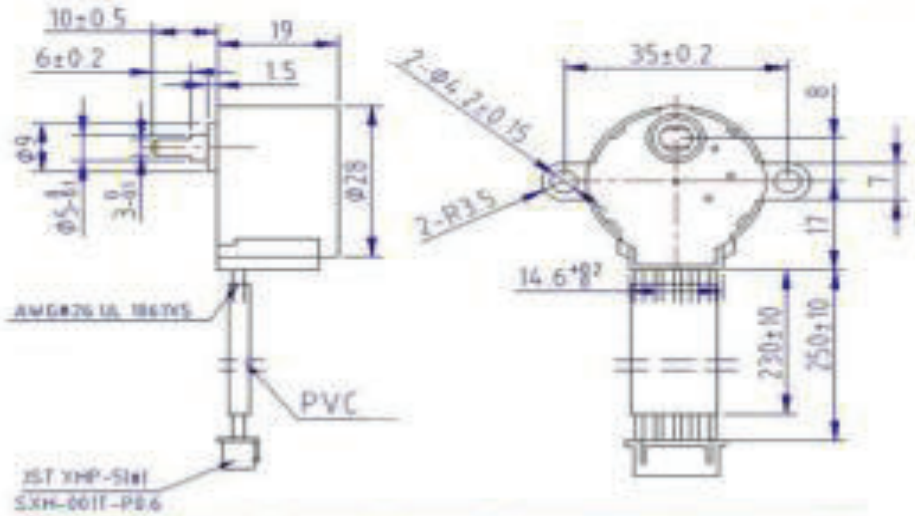
ANEXO 13: HOJA DE DATOS DEL MOTOR A PASOS 28BYJ

Kiatronics

28BYJ-48 – 5V Stepper Motor

The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25°C)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V



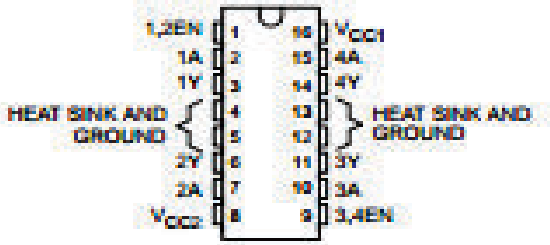
ANEXO 14: HOJA DE DATOS DEL DRIVER DE CORRIENTE L239D

L293, L293D
QUADRUPLE HALF-H DRIVERS

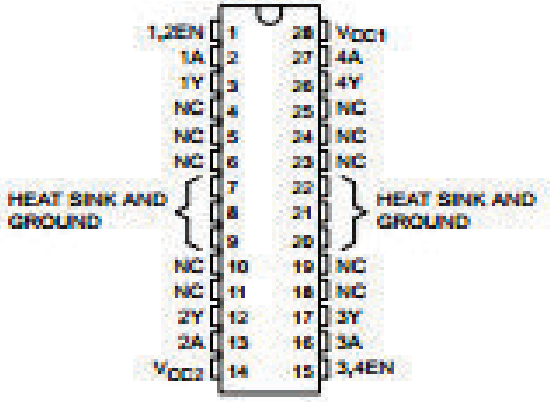
SLRS009B – SEPTEMBER 1986 – REVISED JUNE 2002

- Featuring Unitrode L293 and L293D Products Now From Texas Instruments
- Wide Supply-Voltage Range: 4.5 V to 36 V
- Separate Input-Logic Supply
- Internal ESD Protection
- Thermal Shutdown
- High-Noise-Immunity Inputs
- Functional Replacements for SGS L293 and SGS L293D
- Output Current 1 A Per Channel (600 mA for L293D)
- Peak Output Current 2 A Per Channel (1.2 A for L293D)
- Output Clamp Diodes for Inductive Transient Suppression (L293D)

N, NE PACKAGE
(TOP VIEW)



DWP PACKAGE
(TOP VIEW)



description

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293and L293D are characterized for operation from 0°C to 70°C.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

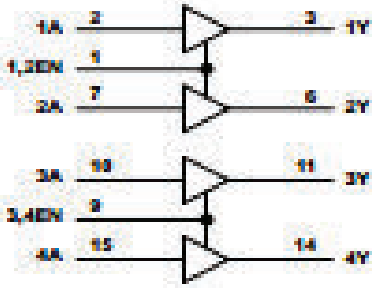
Copyright © 2002, Texas Instruments Incorporated

FUNCTION TABLE
(each driver)

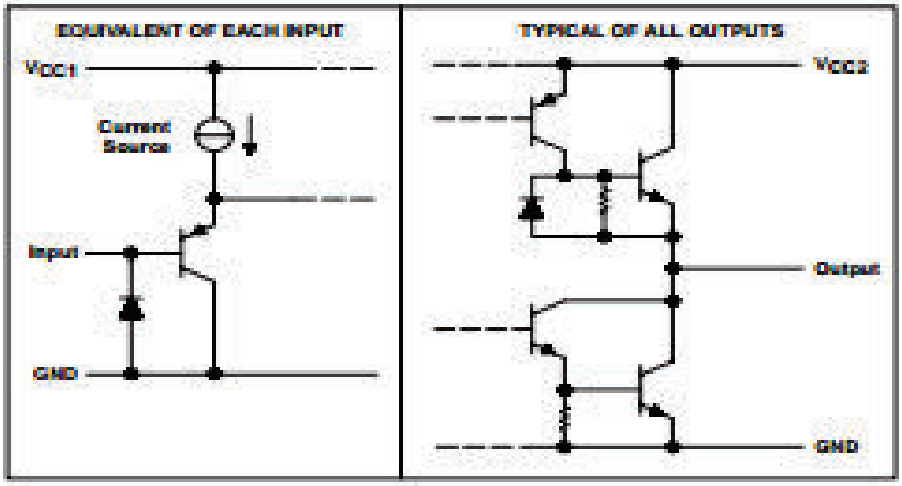
INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)
† In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

logic diagram



schematics of inputs and outputs (L293)



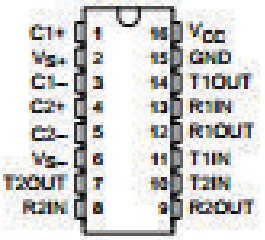
ANEXO 15: HOJA DE DATOS DEL CIRCUITO INTEGRADO MAX232

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

- Meet or Exceed TIA/EIA-232-F and ITU Recommendation V.28
- Operate With Single 5-V Power Supply
- Operate Up to 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Designed to be Interchangeable With Maxim MAX232
- ESD Protection Exceeds JESD 22
  – 2000-V Human-Body Model (A114-A)
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



lescription/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube	MAX232N	MAX232N
	SOIC (D)	Tube	MAX232D	MAX232
		Tape and reel	MAX232DR	
	SOIC (DW)	Tube	MAX232DW	MAX232
		Tape and reel	MAX232DWR	
	SOP (NS)	Tape and reel	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube	MAX232IN	MAX232IN
	SOIC (D)	Tube	MAX232ID	MAX232I
		Tape and reel	MAX232IDR	
	SOIC (DW)	Tube	MAX232IDW	MAX232I
		Tape and reel	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

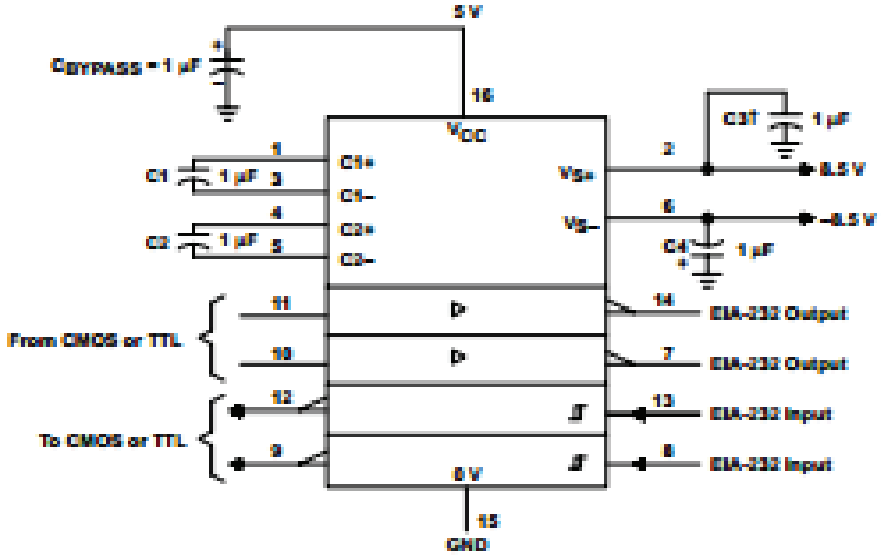
LinASIC is a trademark of Texas Instruments.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

Copyright © 2002, Texas Instruments Incorporated

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

APPLICATION INFORMATION



† C3 can be connected to V_{CC} or GND.

Figure 4. Typical Operating Circuit

ANEXO 16: MANUAL PARA EL DESARROLLO DE UN PROGRAMA EN ISE DESIGN DE XILINX

ISE WebPACK Design Software es un programa gratuito de Xilinx que permite el diseño, síntesis, simulación, implementación, montaje y la programación JTAG, es decir ofrece una funcionalidad completa.

Para su descarga acudimos a la página oficial de Xilinx, opción Downloads, damos un clic en la pestaña ISE Design Tools, nos dirigimos a ISE Design Suite y descargamos Full DVD Single File Download Image.

A continuación en la figura 10 se muestra una captura de pantalla del software que se descarga para el desarrollo del programa en el FPGA.

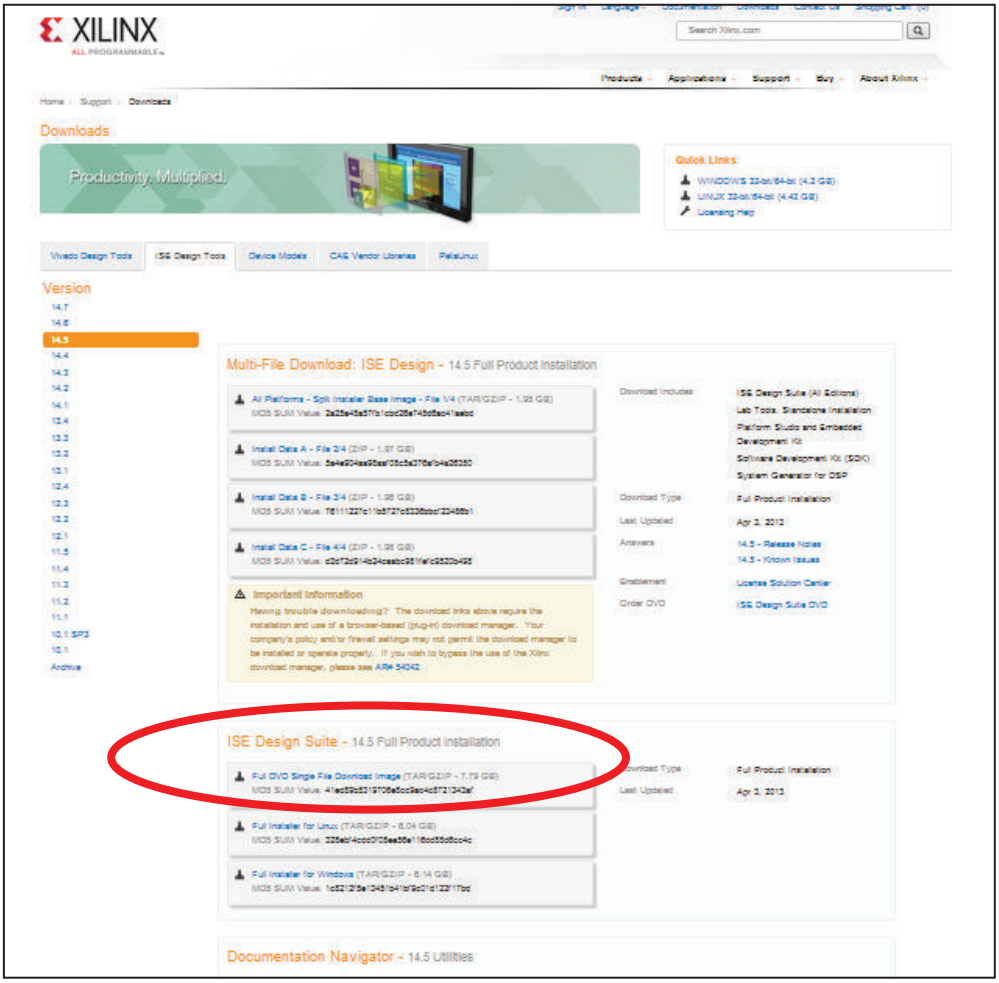


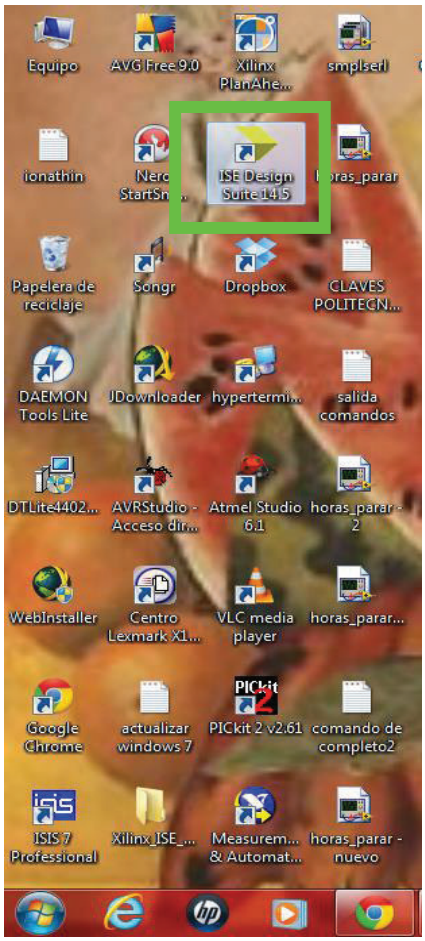
Figura 10: Descarga de ISE Design Suite 14.5

Una vez finalizada la descarga se procede a la instalación y utilización del software.

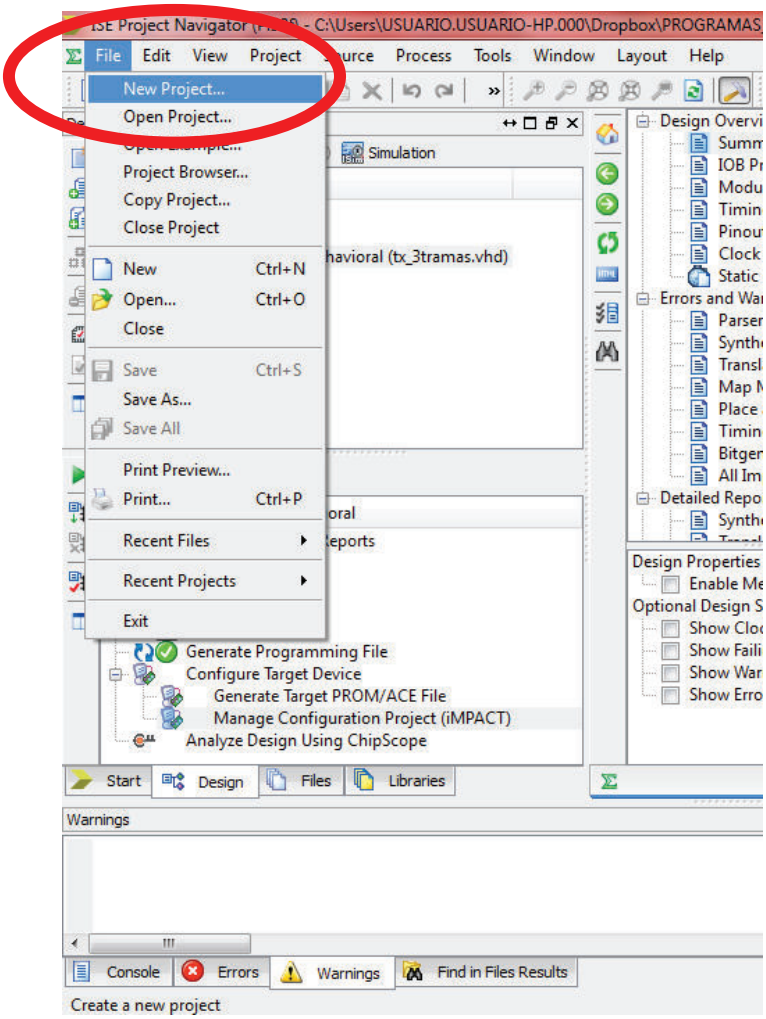
DISEÑO DE UN PROGRAMA PARA ENCENDER O APAGAR UN LED EN ISE Design Suite 14.5

Una vez instalado el software de Xilinx podemos utilizarlo para desarrollar un sinnúmero de programas para distintas aplicaciones, en este ejemplo se va a diseñar un programa que permita encender o apagar un led controlado por un switch. Para ello se describe a continuación el proceso que seguimos para el desarrollo de este programa:

1. Abrir la aplicación *ISE Design Suite 14.5*



2. Una vez abierto el programa, damos un clic en *File – New Project*



3. Se despliega la ventana *New Project Wizard*

3.1 En la opción *Name* ingresamos el nombre del nuevo proyecto, en este caso será: *LED*

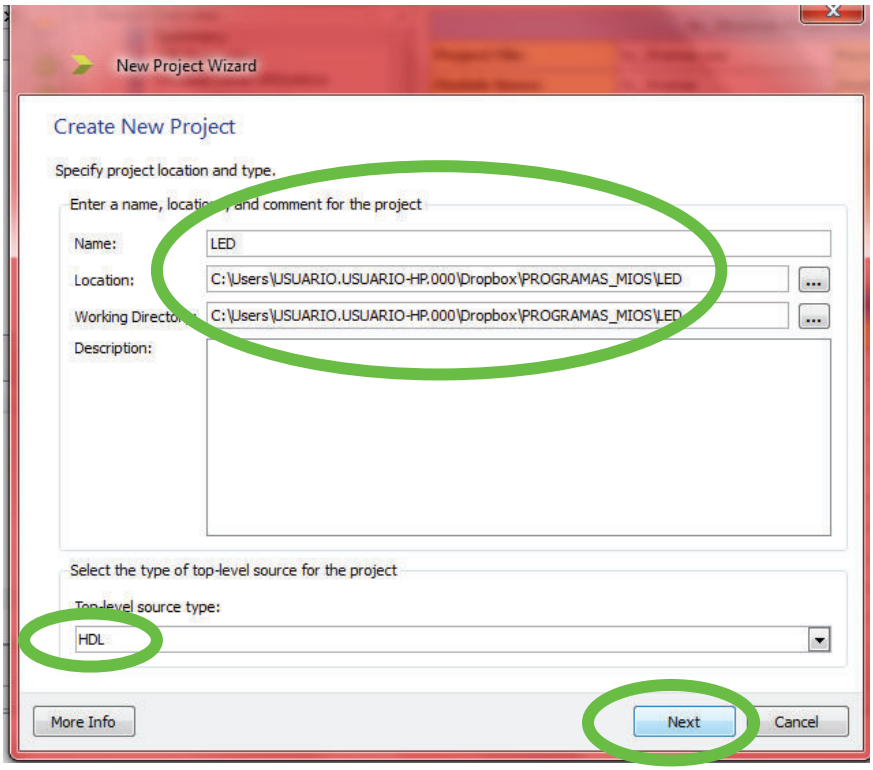
3.2 En la opción *Location* ingresamos la ubicación donde se guardará el nuevo proyecto, en este caso será:

C:\Users\USUARIO.USUARIO-HP.000\Dropbox\PROGRAMAS_MIOS\LED

3.3 El *Working Directory* se crea automáticamente al crear la locación del programa.

3.3 En la opción *Top-level source type* seleccionamos *HDL*

3.4 Damos un clic en Next



4. La siguiente ventana de *New Project Wizard* nos permite especificar las propiedades de nuestro proyecto y tarjeta que vamos a utilizar.

4.1 En *Evaluation Development Board* seleccionamos *Spartan-3E Starter Board* que es la tarjeta con la que desarrollamos todas nuestras aplicaciones.

4.2 Las opciones Product *Category, Family, Device, Package, Speed, Top-Level Source Type* ya vienen dadas al seleccionar la tarjetade trabajo.

4.3 En la opción *Synthesis Tool* viene seleccionada automáticamente *XST(VHDL/Verilog).*

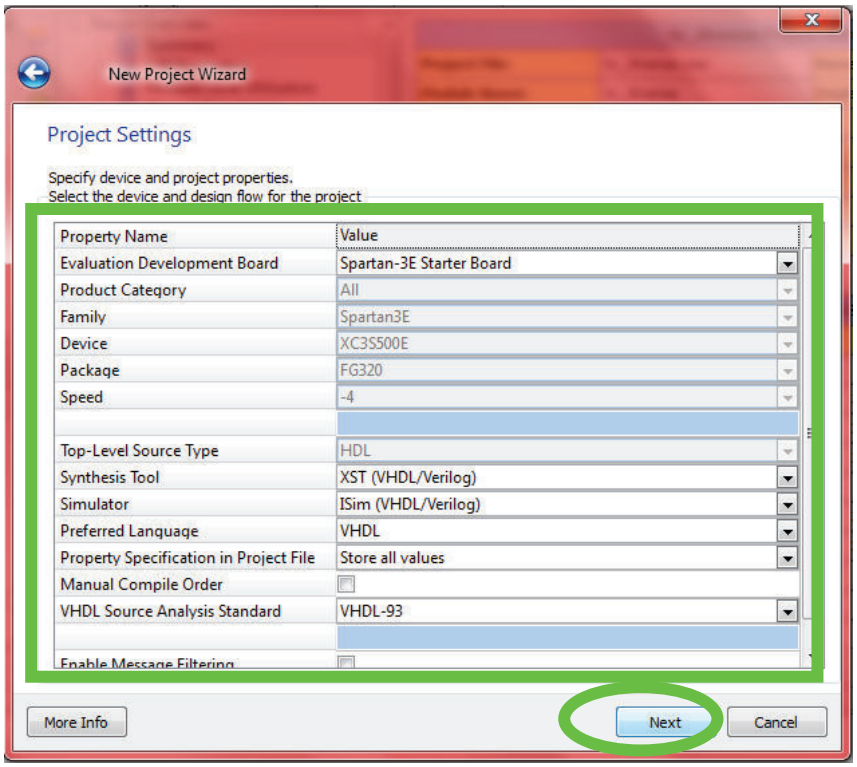
4.4 En la opción *Simulator* seleccionamos *Isim(VHDL/Verilog).*

4.5 En la opción *Preferred Language* seleccionamos *VHDL*

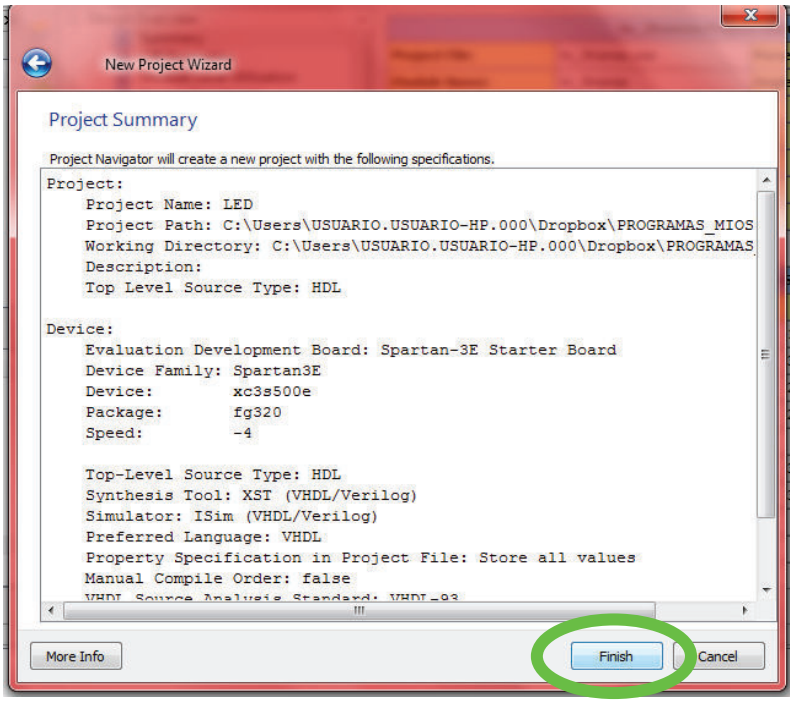
4.6 En la opción *Property Specification in Project File* seleccionamos la opción *Store all values*

4.7 En la opción VHDL *Source Analysis Standard* seleccionamos la opción *VHDL-93*

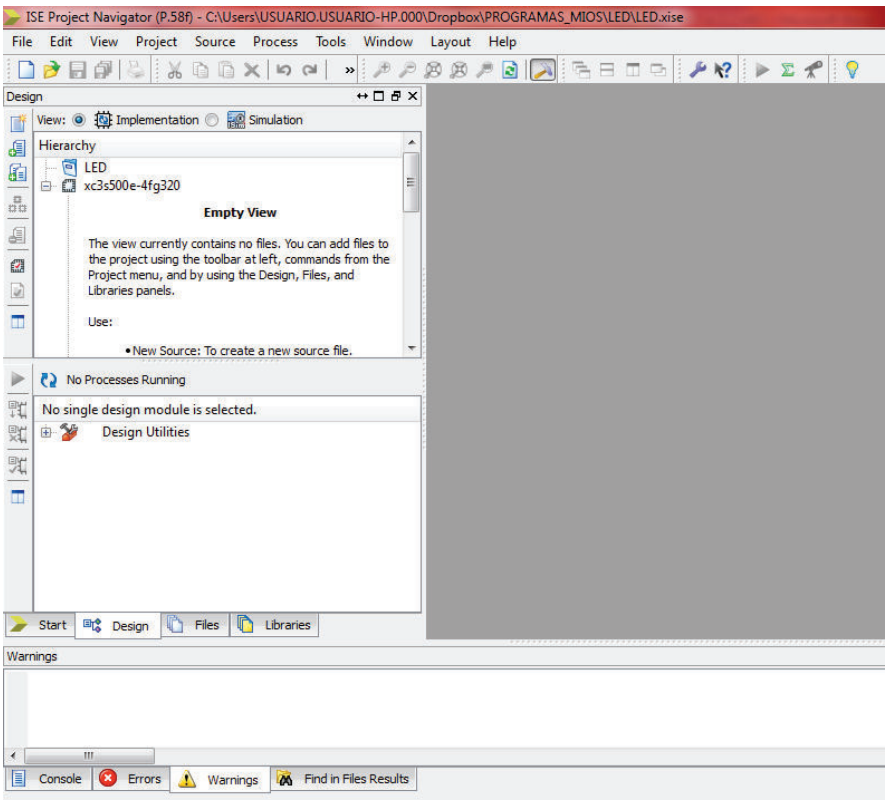
4.8 Damos un clic en *Next*



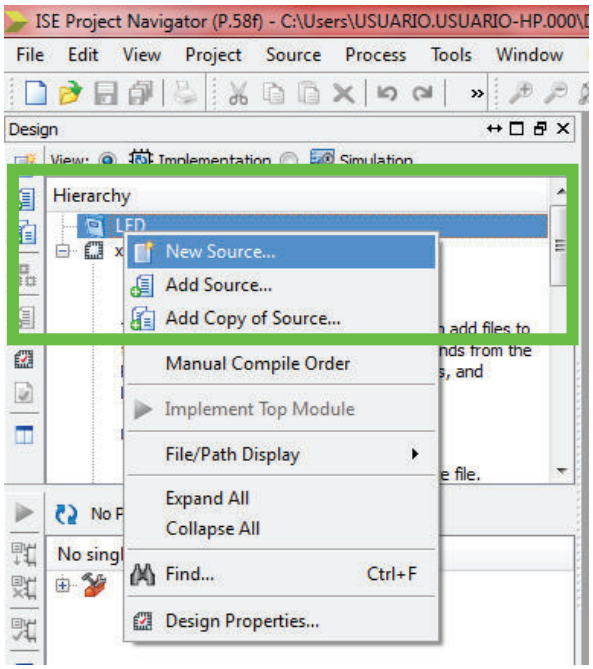
5. Se despliega una nueva ventana *Project Summary*, en la cual se observa un resumen de todas las especificaciones anteriormente seleccionadas. Comprobamos que la información sea correcta y damos un clic en *Finish.*



6. Una vez que se ha creado el proyecto, se abre en una ventana en donde podemos observar que se encuentra temporalmente vacío.



7. Damos clic derecho en *LED – New Source*



8. Se despliega una ventana denominada *Select Source Type*, donde procedemos a seleccionar:

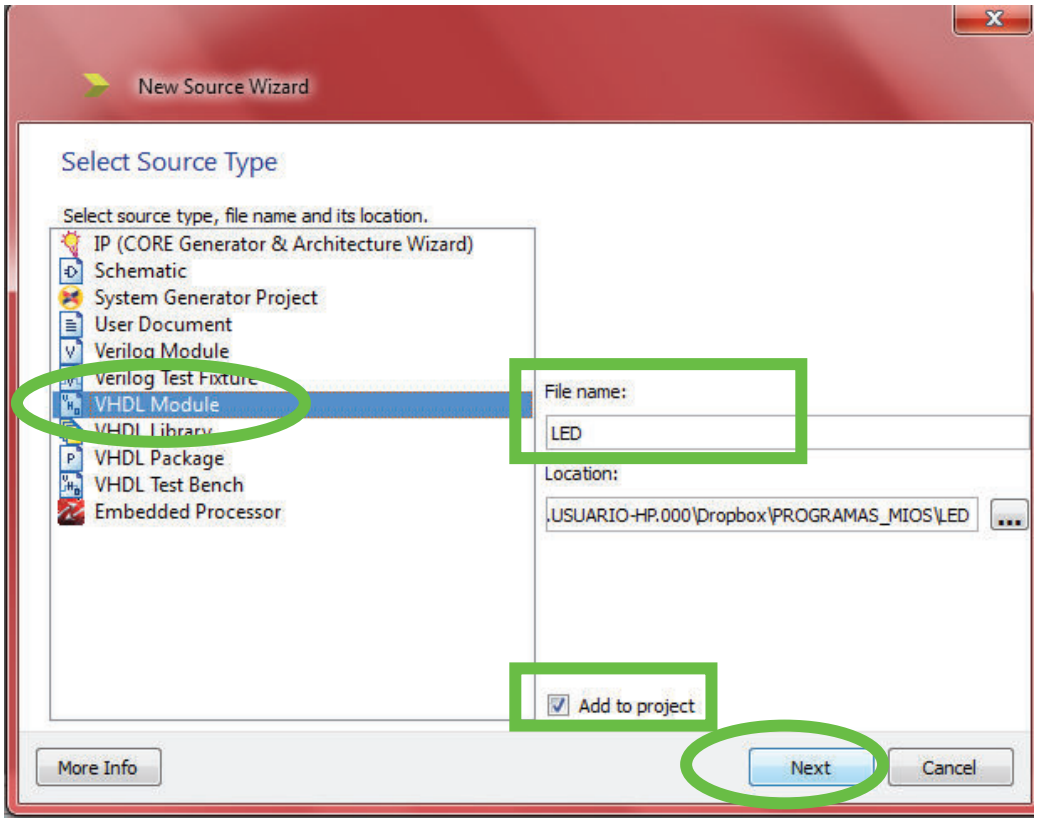
8.1 En la opción *Source Type* seleccionamos *VHDL Module*

8.2 En la opción *File Name* designamos un nombre a nuestro archivo que en este caso será *LED*

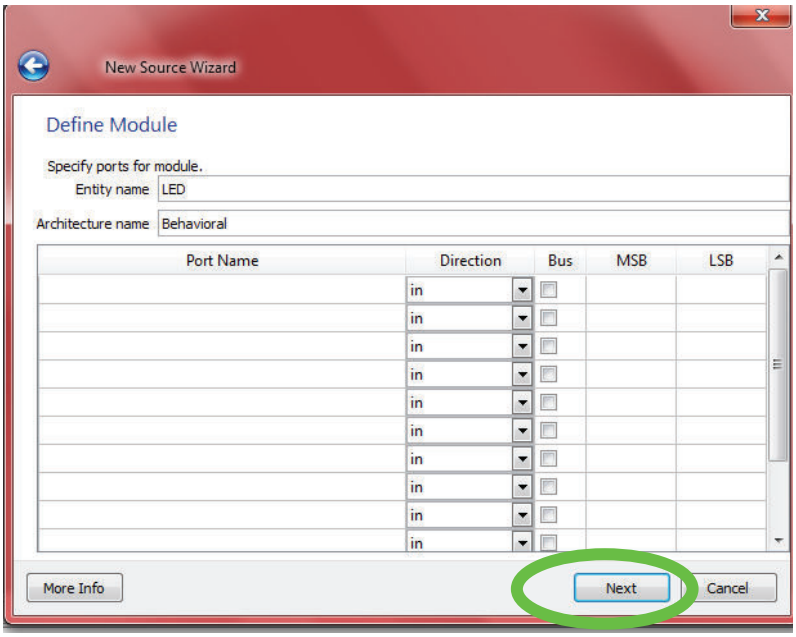
8.3 En la opción *Location* dejamos la ubicación que viene por defecto al haber creado el proyecto.

8.4 Debemos cerciorarnos que esté seleccionada la opción *Add to Project*

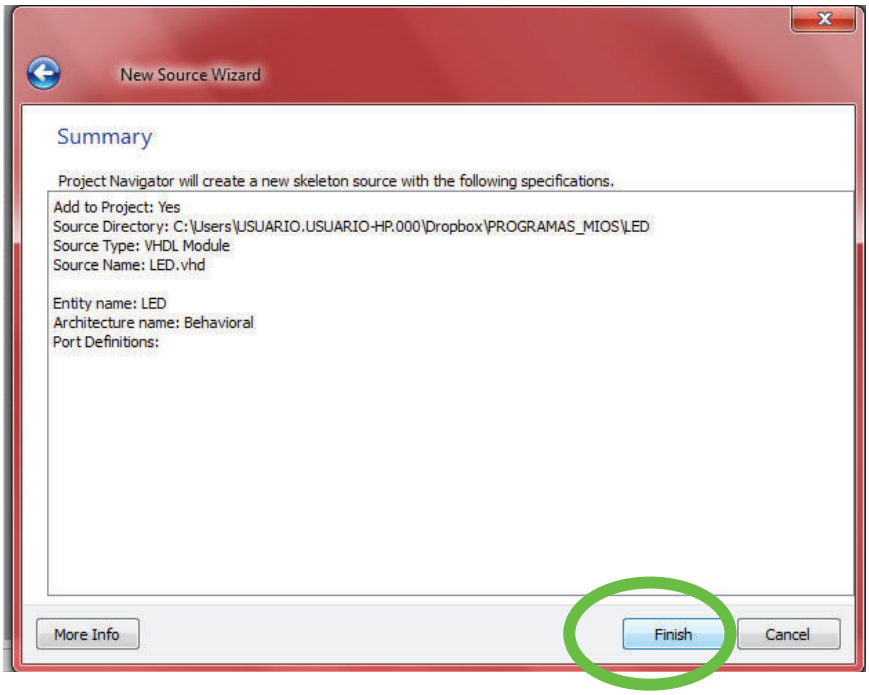
8.5 Damos un clic en *Next*



9. A continuación se abre una ventana *Define Module* en donde se pueden especificar los puertos de entrada y salida que se van a utilizar en el proyecto que estamos creando, sin embargo en nuestro caso definiremos dichos puertos en *Entidad* dentro del programa. Por esta razón no modificamos nada y damos clic en *Next.*



10. A continuación se abre una nueva ventana *Summary* en donde podemos verificar la información que hemos ingresado. Si todo está correcto damos clic en *Finish*.

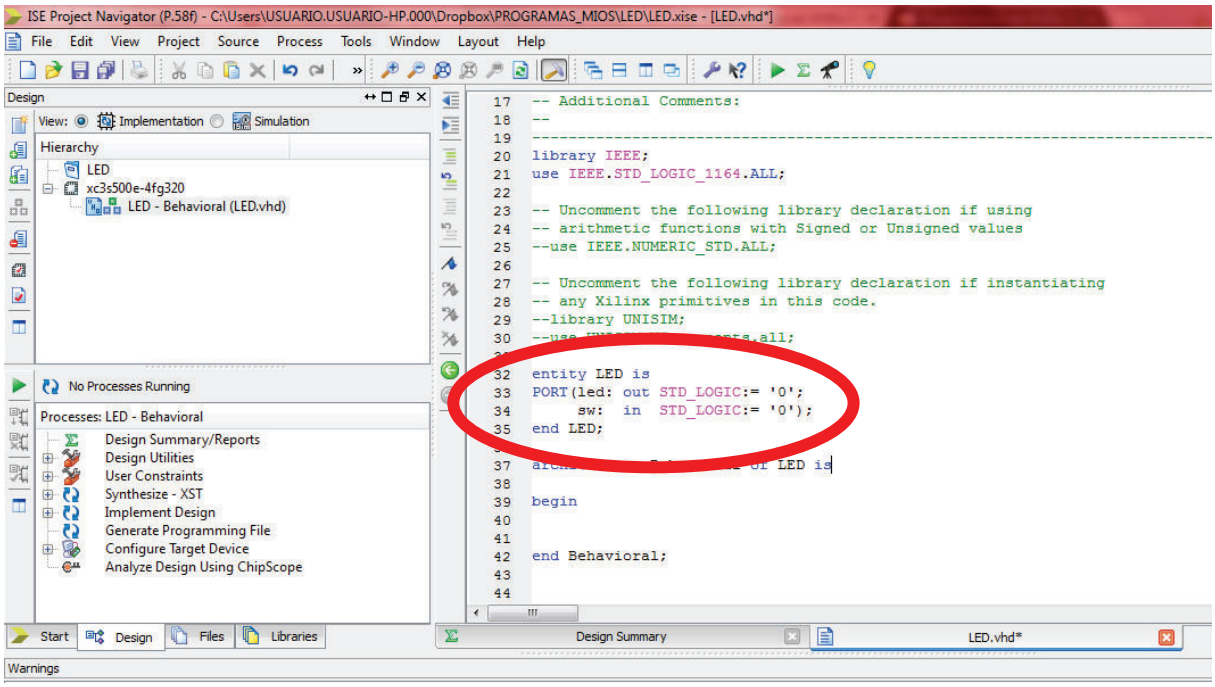


11. A continuación una vez que se ha creado el proyecto procedemos a configurar las entradas y salidas que usaremos en nuestro FPGA, se debe tomar en cuenta que

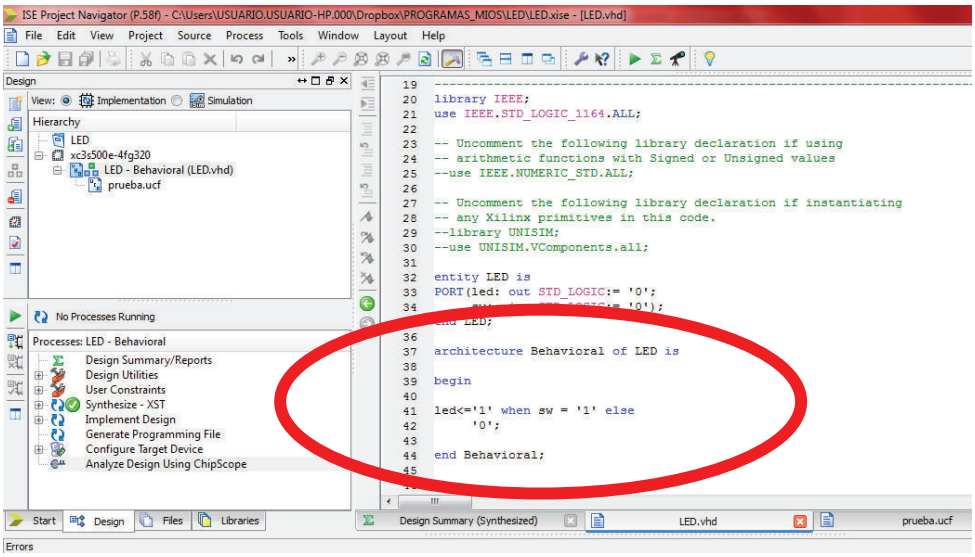
las entradas se definen en la sección ENTITY que se crea automáticamente en el proyecto.

Para definir las entradas se utiliza la palabra *PORT* y a continuación se escribe el nombre que usaremos para cada entrada o salida respectivamente, finalmente el tipo de dato que le asignaremos es *STD_LOGIC* que dependiendo del programa puede asignará al puerto un '1' ó '0'.

En nuestro caso usamos *led* como una salida inicializada en '0', es decir que está apagado y *sw* como una entrada inicializada en '0' y que controlará el encendido o apagado del led.

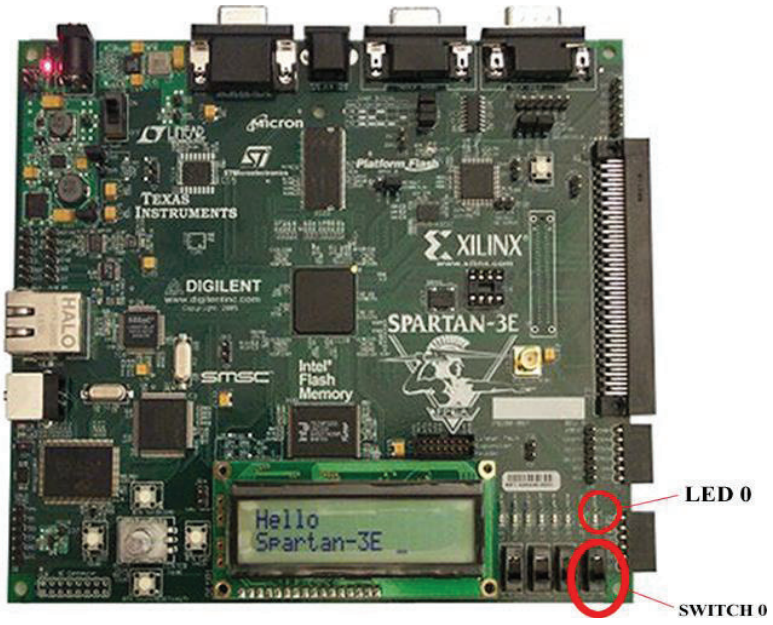


12. Describimos el comportamiento del circuito en la parte denominada *ARCHITECTURE*, en donde *BEGIN* es la palabra que da inicio al programa. En nuestro caso, hacemos que *led* se encienda cuando *sw* está en '1' caso contrario *led* se mantiene apagado. El programa finaliza con *end Behavioral.*

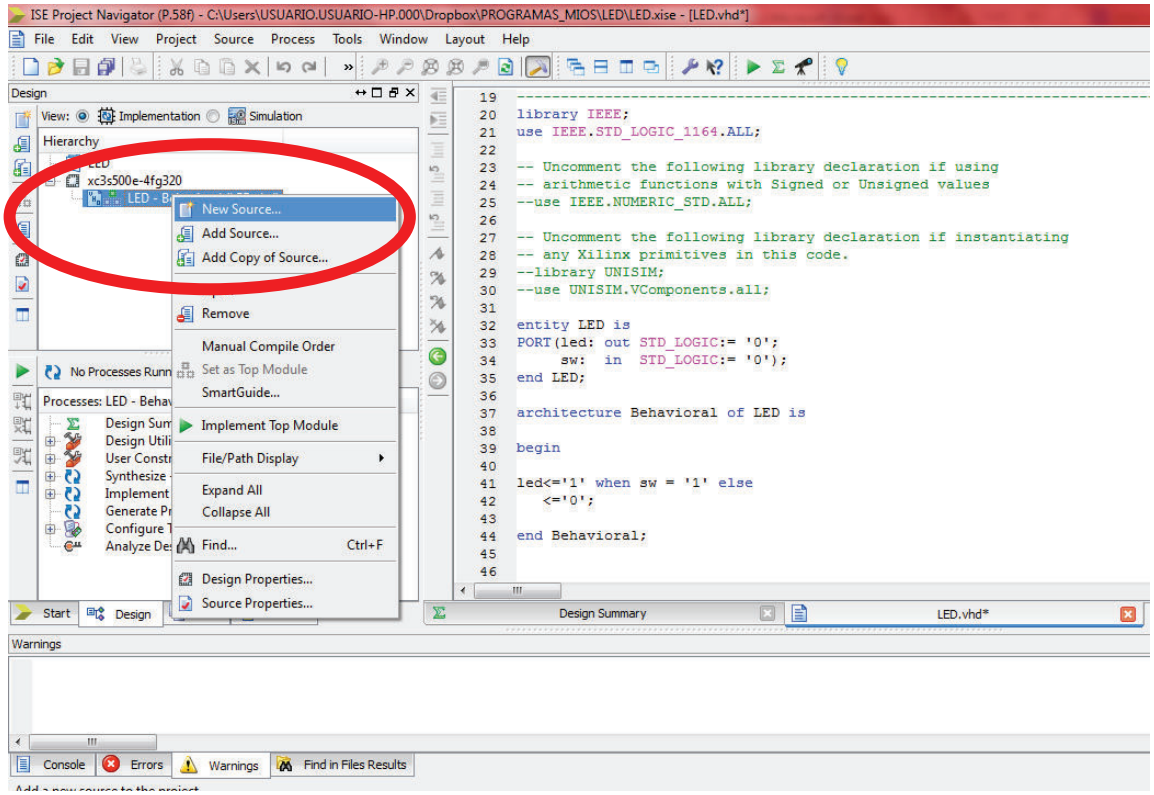


13. Una vez finalizado el programa, procedemos a la asignación de los pines físicos del FPGA que vamos a utilizar a las entradas y salidas configuradas en la entidad del programa. Para ello revisamos la guía del usuario de la Spartan-3E Starter Board *(UG230*), en la cual nos indica el valor correspondiente de los pines que estamos utilizando.

En nuestro caso vamos a utilizar el *led 0* y el *switch 0* del FPGA, así que buscamos en la guía dichos pines.



14. Creamos el *User Constraint File (UCF),* para ello doy un clic derecho en *LED-Behavioral – New Source*



15. Se despliega una ventana denominada *Select Source Type*, donde procedemos a seleccionar:

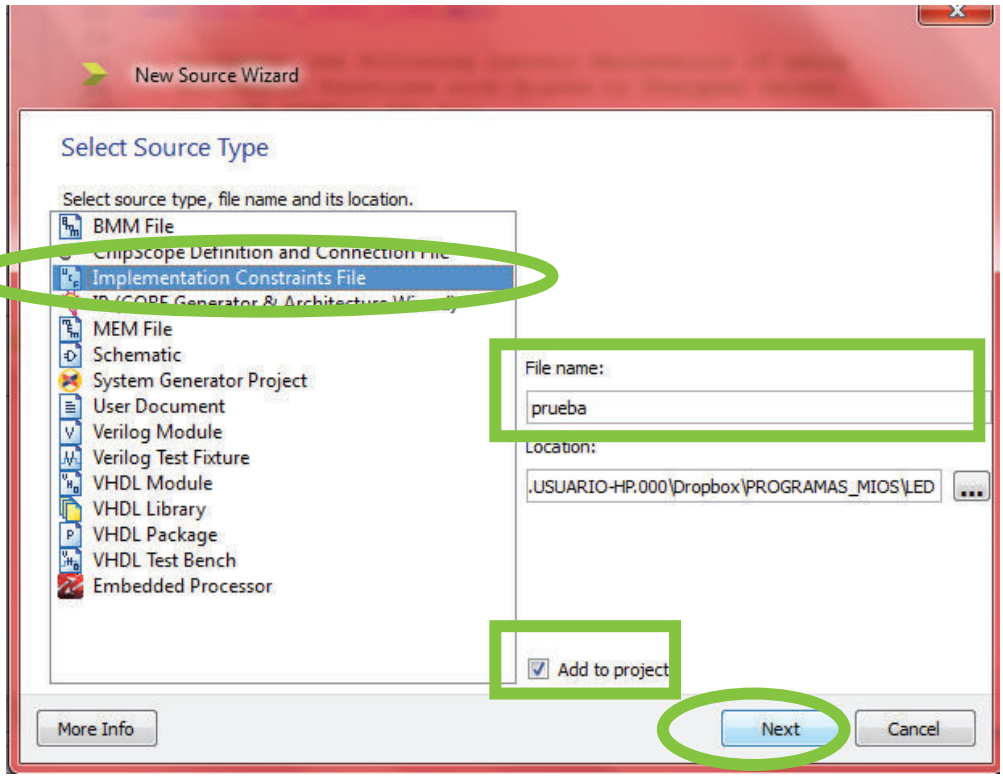
15.1 En la opción *Source Type* seleccionamos *Implementation Constraints File*

15.2 En la opción *File Name* designamos un nombre a nuestro archivo que en este caso será *prueba*

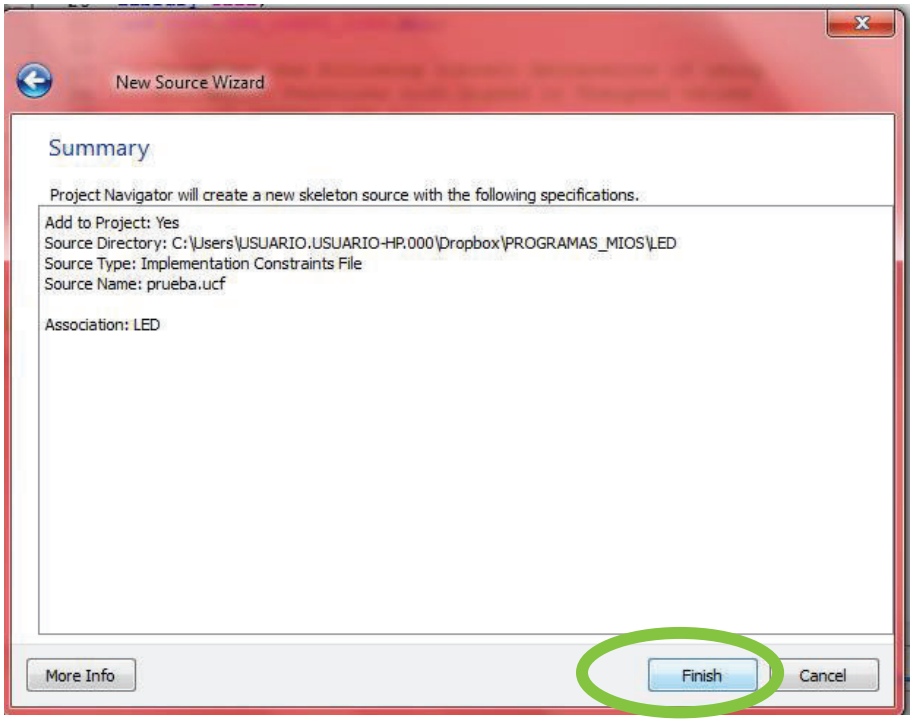
15.3 En la opción *Location* dejamos la ubicación que viene por defecto al haber creado el proyecto.

15.4 Debemos cerciorarnos que esté seleccionada la opción *Add to Project*

15.5 Damos un clic en *Next*

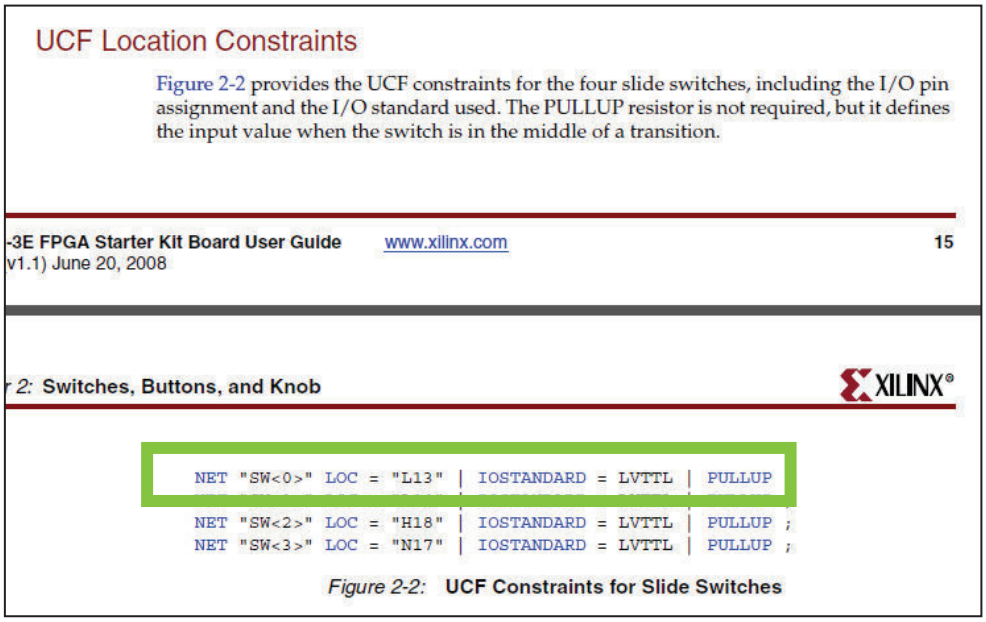


16. A continuación se abre una nueva ventana *Summary* en donde podemos verificar la información que hemos ingresado. Si todo está correcto damos clic en *Finish*.



17. A continuación buscamos los valores de *switches* y *leds* en la guía y copiamos los valores que vamos a utilizar, en nuestro caso utilizamos el valor de *SW0* y de *LED0*.

Valores de los Switches:



UCF Location Constraints

Figure 2-2 provides the UCF constraints for the four slide switches, including the I/O pin assignment and the I/O standard used. The PULLUP resistor is not required, but it defines the input value when the switch is in the middle of a transition.

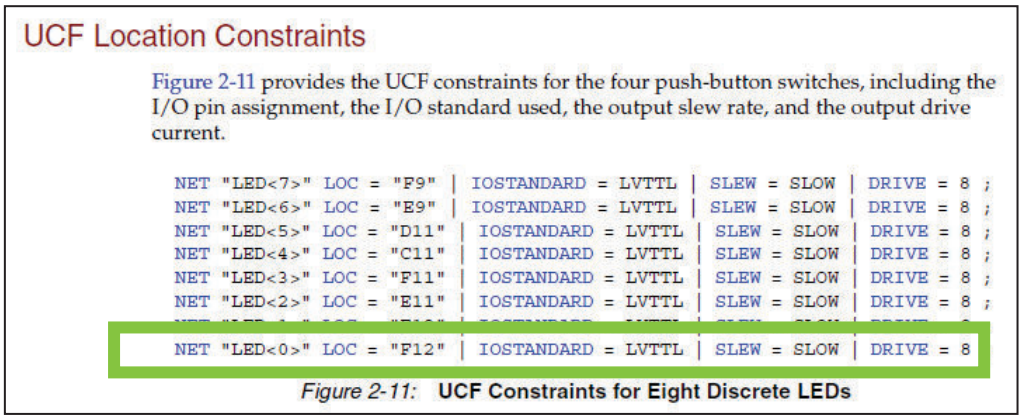
-3E FPGA Starter Kit Board User Guide www.xilinx.com 15
v1.1) June 20, 2008

r 2: Switches, Buttons, and Knob XILINX®

```
NET "SW<0>" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;
```

Figure 2-2: **UCF Constraints for Slide Switches**

Valores de los Leds:



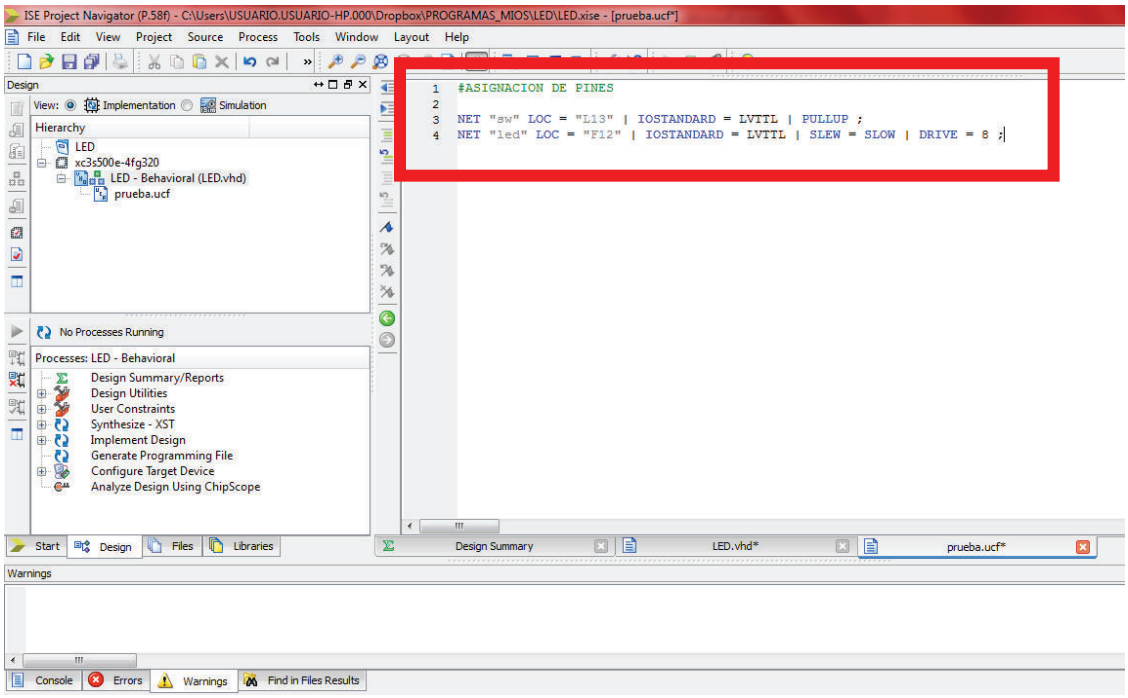
UCF Location Constraints

Figure 2-11 provides the UCF constraints for the four push-button switches, including the I/O pin assignment, the I/O standard used, the output slew rate, and the output drive current.

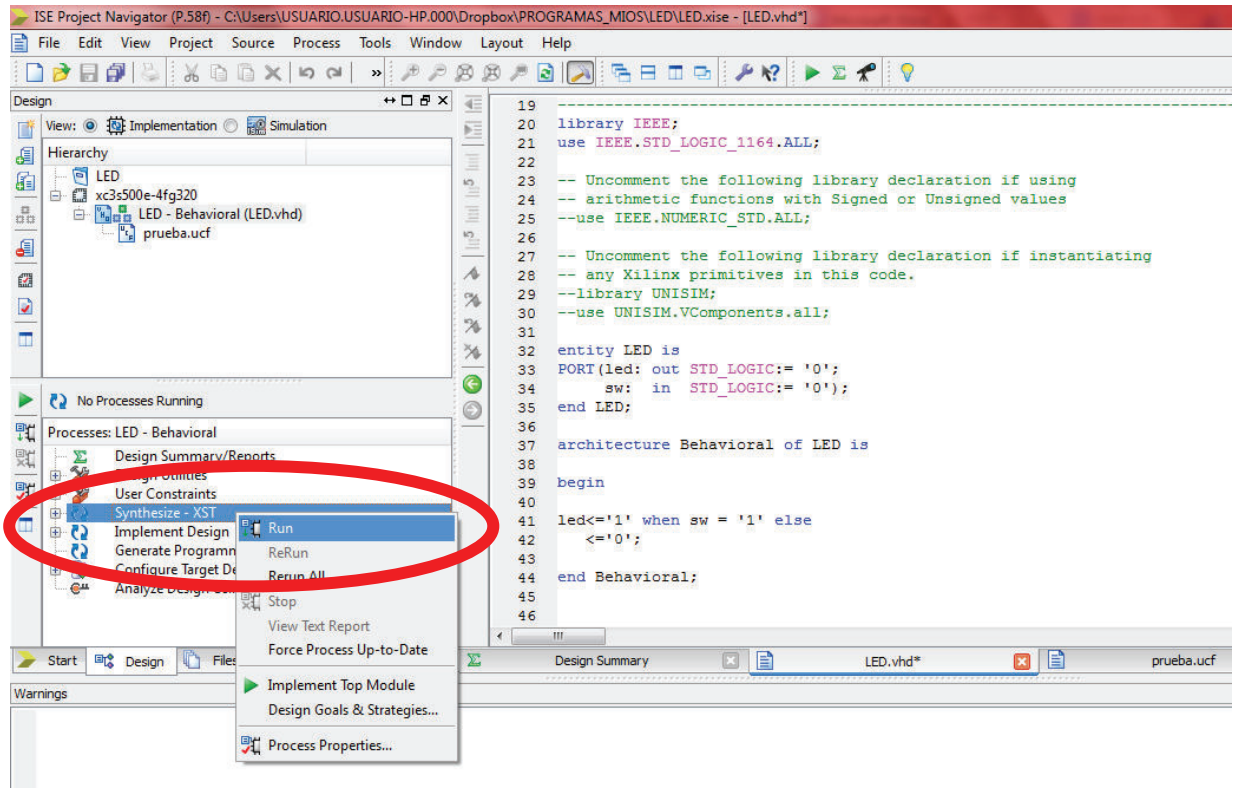
```
NET "LED<7>" LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<6>" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8
```

Figure 2-11: **UCF Constraints for Eight Discrete LEDs**

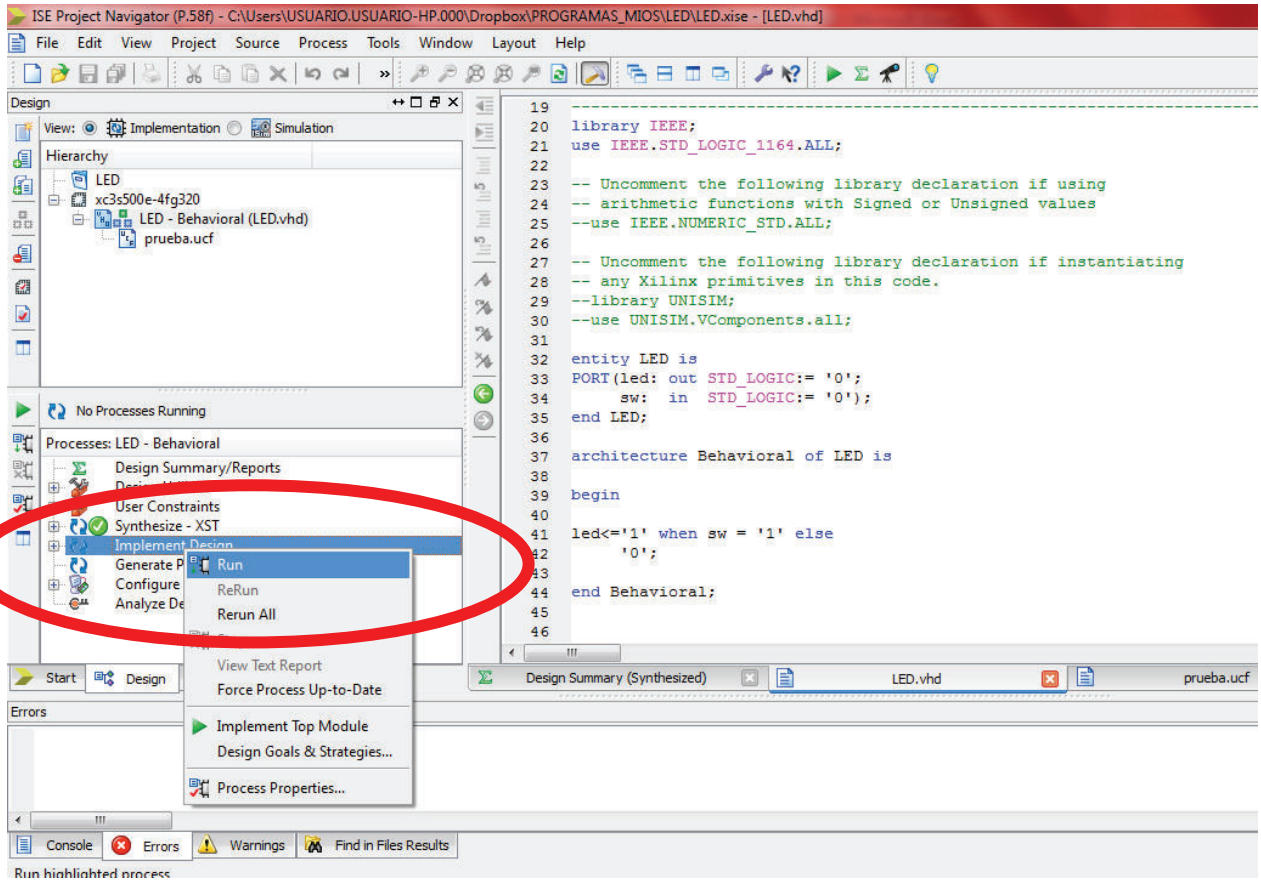
18. A continuación se abre el archivo prueba.ucf en donde pegamos los valores copiados en el literal anterior, tomando en cuenta que debemos cambiar los nombres de los pines de tal manera que correspondan a los nombres que asignamos en las entradas y salidas de nuestro programa.



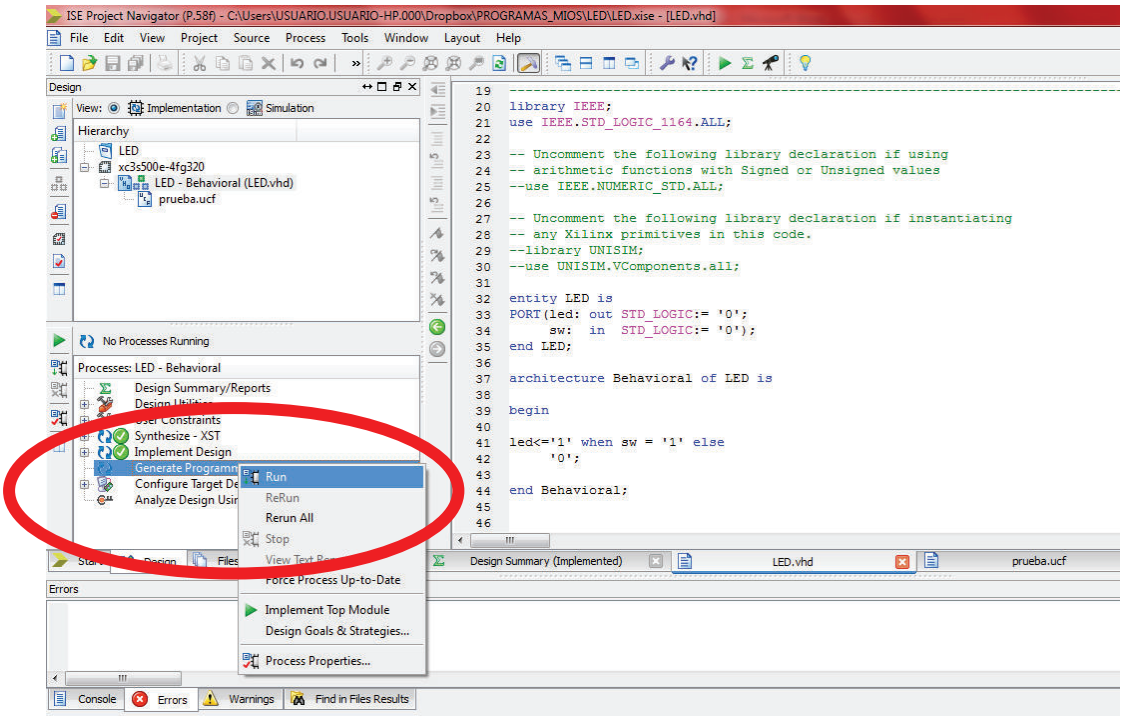
19. Una vez asignados los pines, procedemos a guardar los cambios y a sintetizar el programa. Para ello damos clic derecho en *Synthesize – Run.*



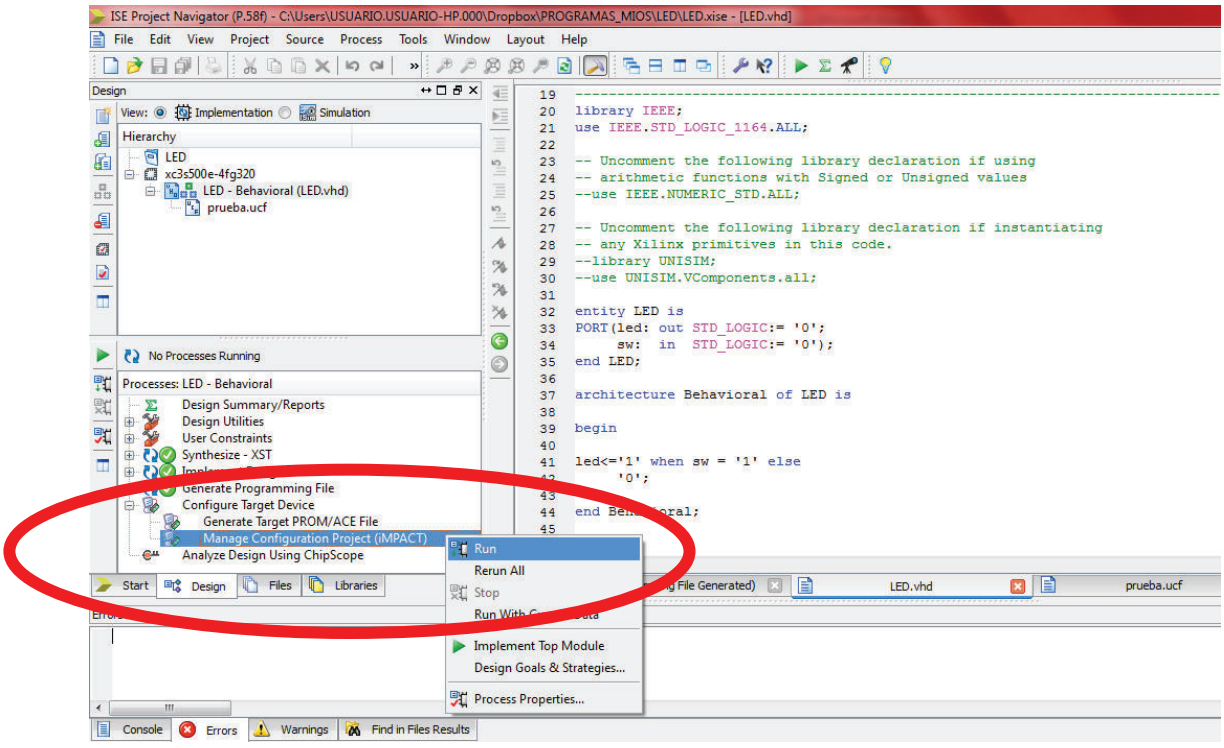
20. A continuación una vez sintetizado, ejecutamos *Implement Desing* de la misma manera como sintetizamos el programa.



21. A continuación debemos generar el archivo *.bit* que es el que se carga en la tarjeta para realizar la implementación. Para ello ejecutamos *Generate Programming File dando* clic derecho y *Run.*

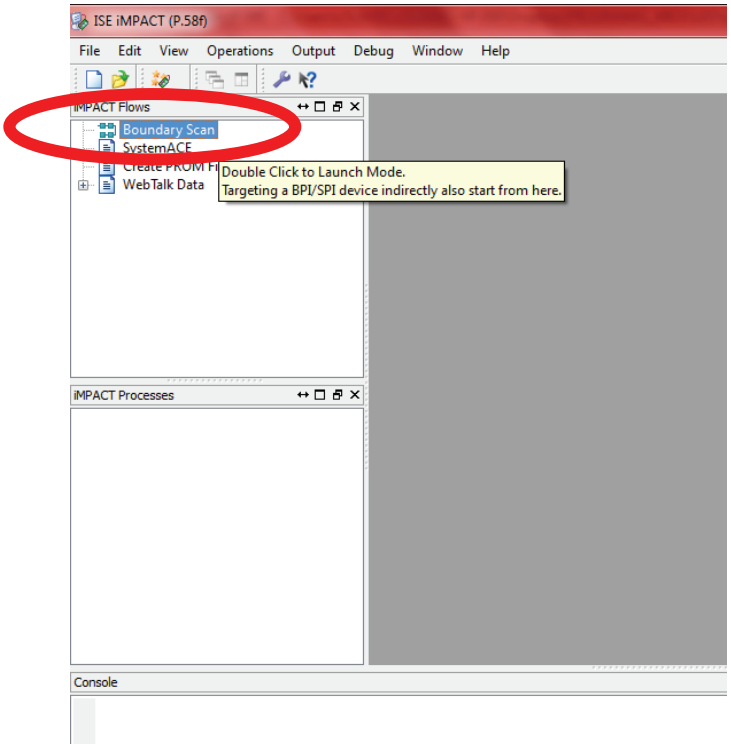


22. Para cargar el archivo .bit en la tarjeta procedemos a ejecutar la opción *Configure Target Device – Manage Configuration Project (iMPACT) – Run.*

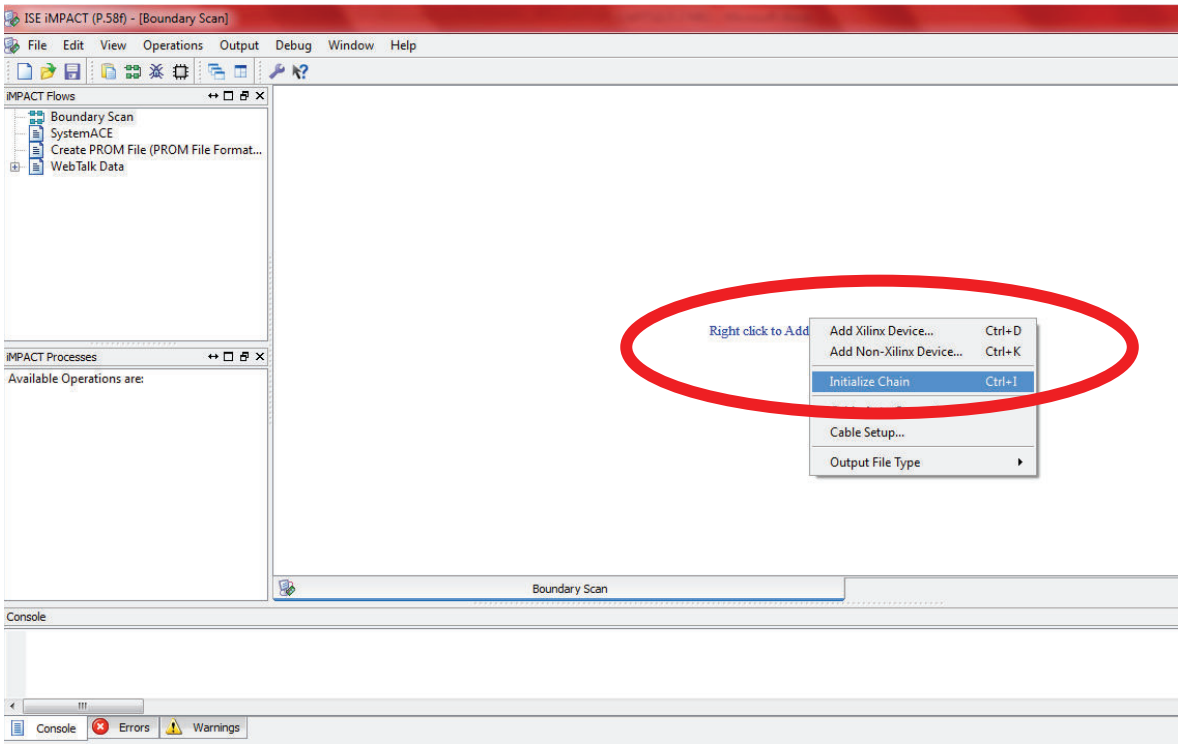


23. Conectamos la tarjeta a la computadora para proceder a cargar el programa.

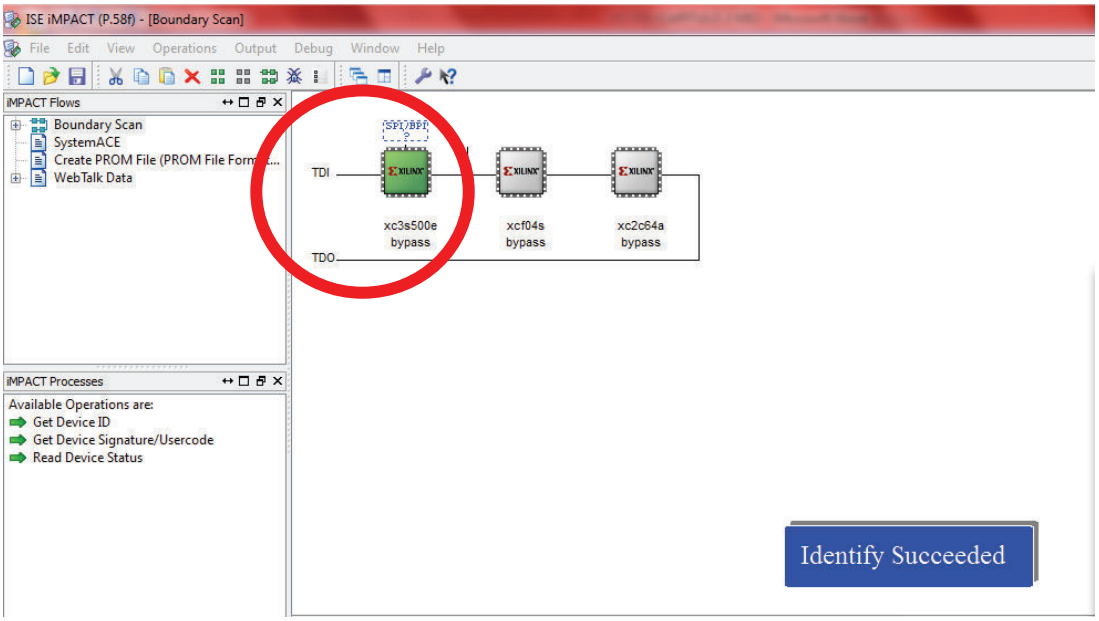
24. Automáticamente se abre la ventana de iMPACT y damos doble clic en la opción *Boundary Scan.*



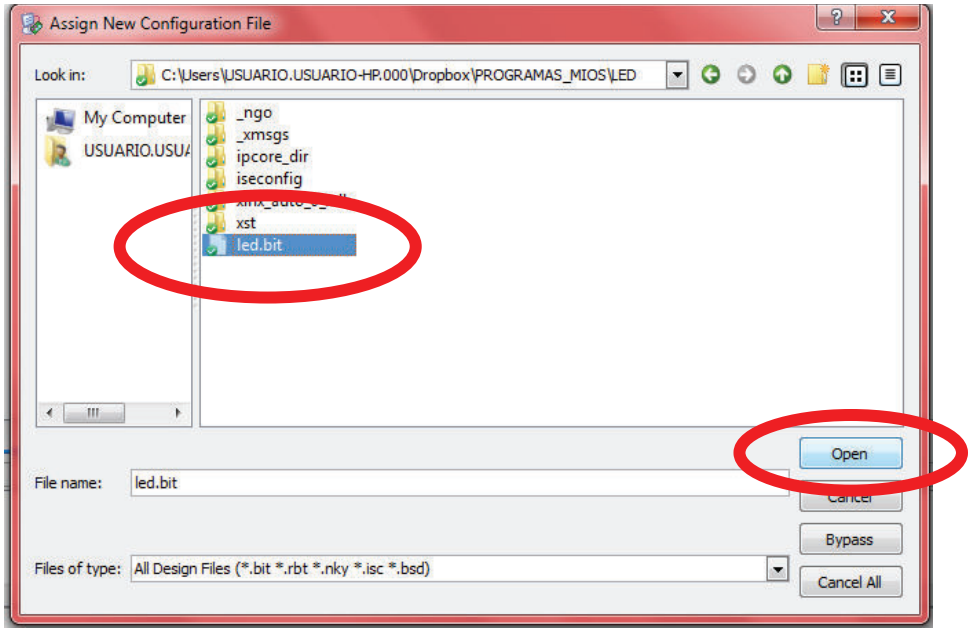
25. A continuación damos clic derecho en la pantalla principal *Initialize Chain*.



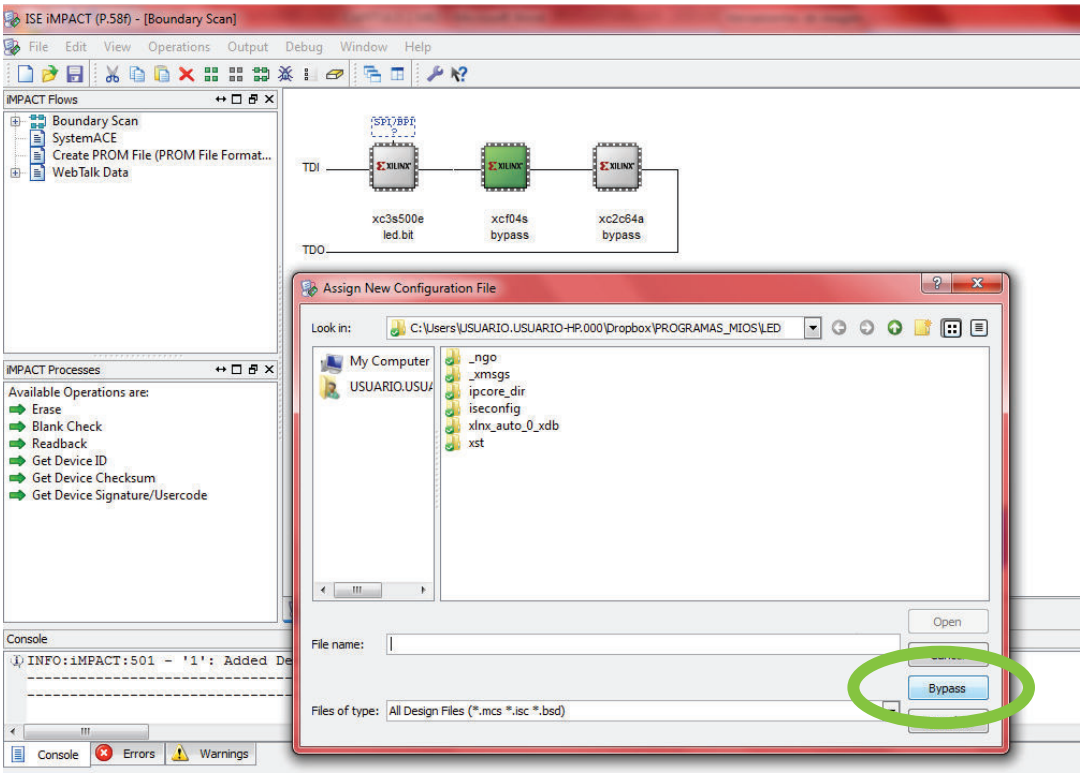
26. Para cargar el programa se observan 3 chips correspondientes a la tarjeta, en el que vamos a guardar el programa es en el primer chip que se encuentra de color verde.



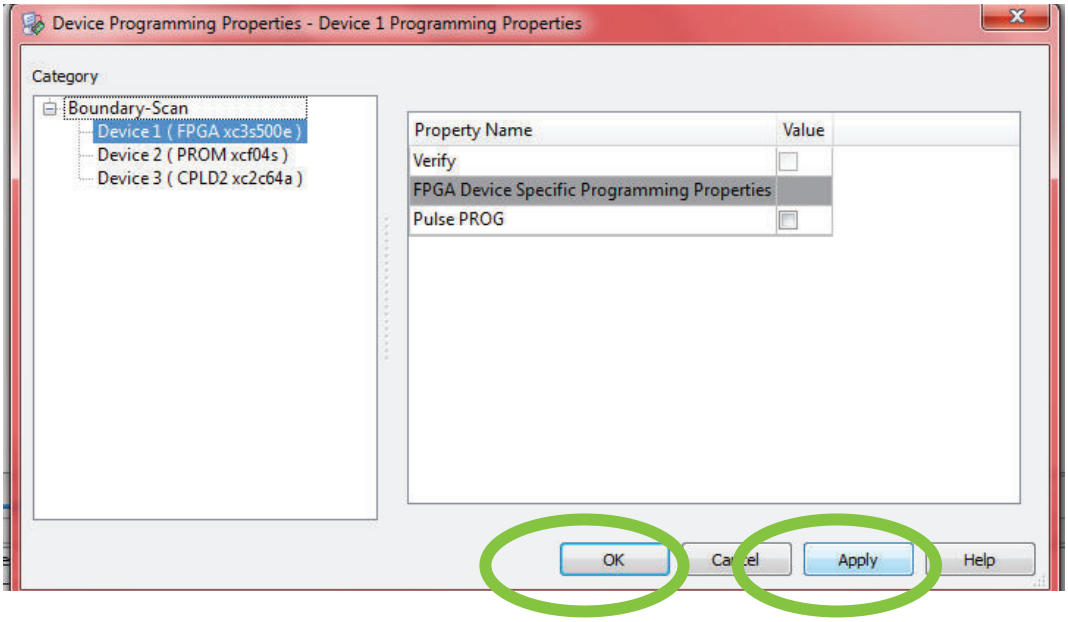
27. En la ventana *Assign New Configuration File* seleccionamos *LED – led-bit – OPEN.*



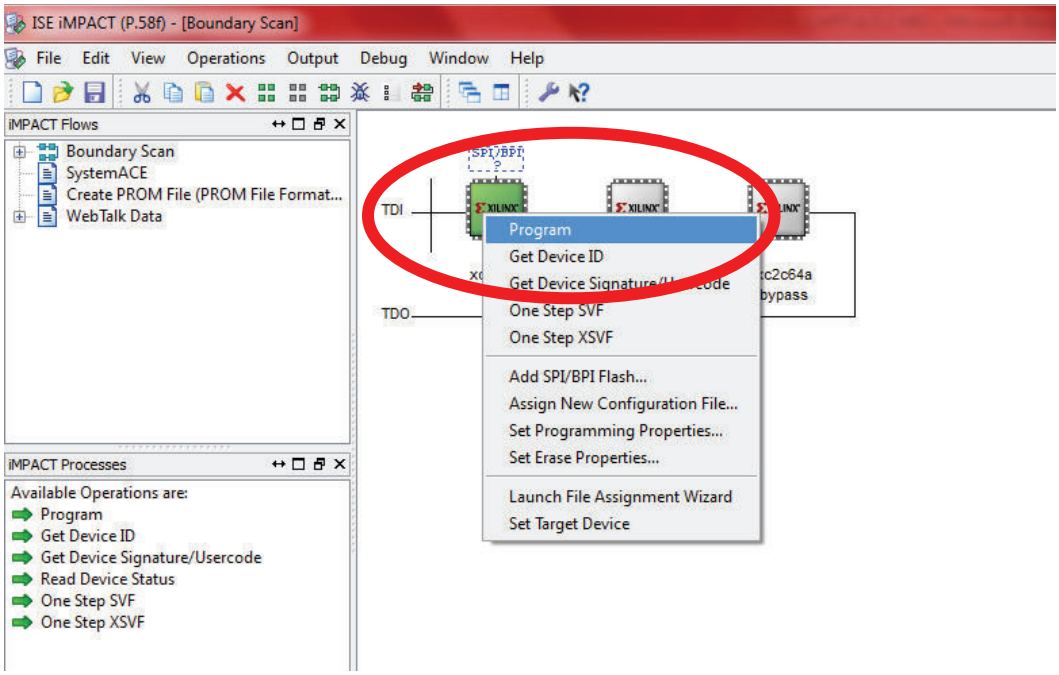
28. Se abren más ventanas que indican si se desea cargar algún archivo en los dos siguientes chips, como no se desea hacerlo damos clic en *Bypass.*

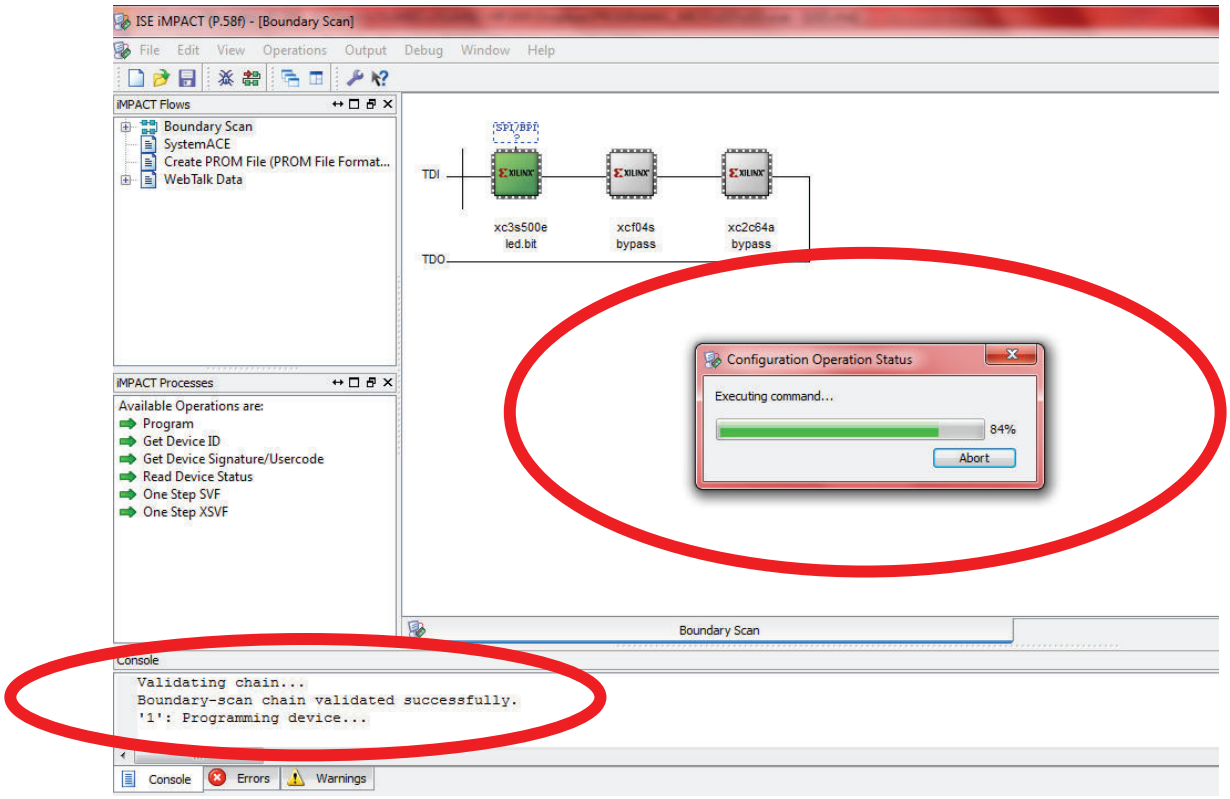


29. En la siguiente ventana damos clic en *Apply* y luego *OK.*

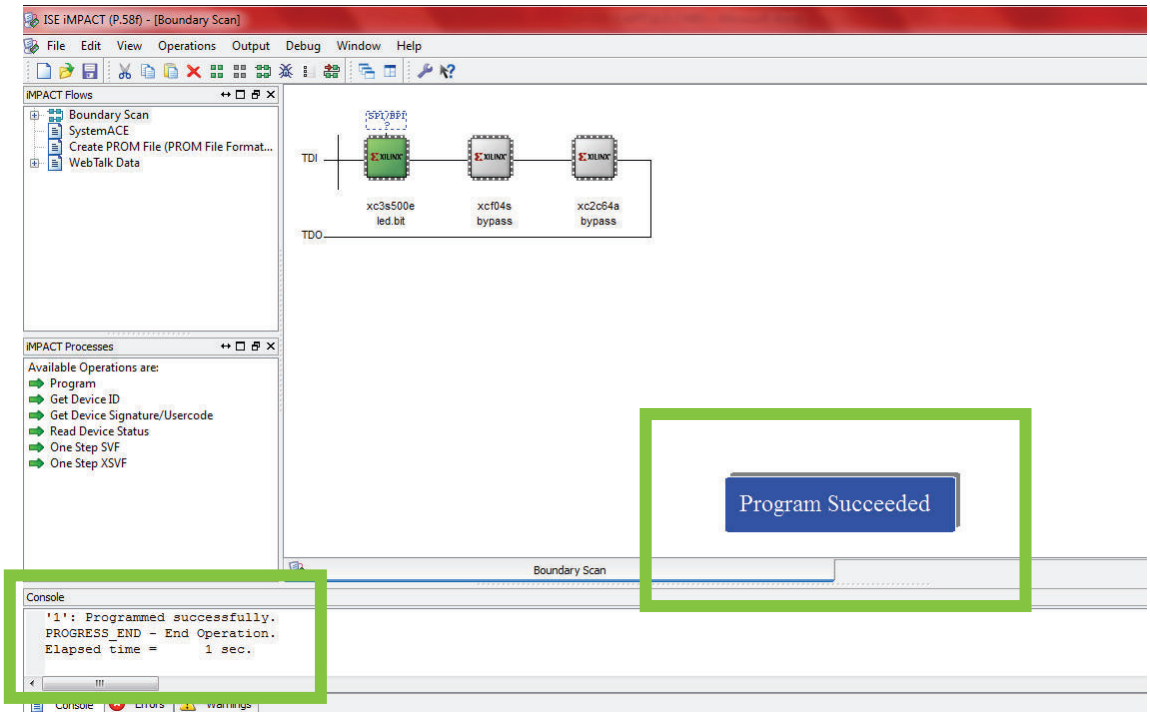


30. En el primer chip damos clic derecho *Program* para poder cargar el archivo *led.bit* en la tarjeta.



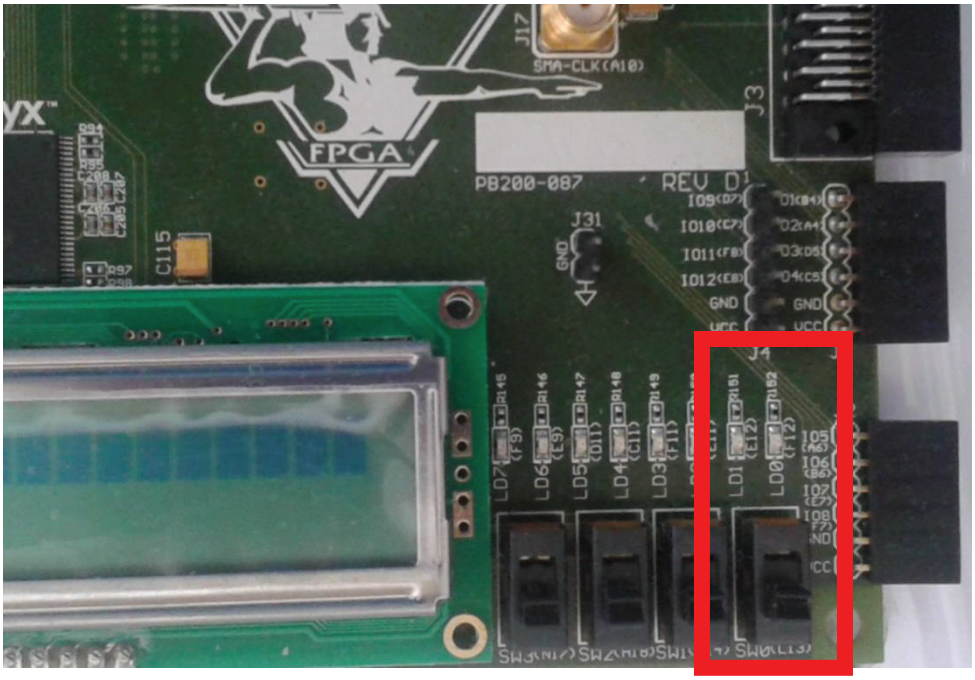


31. Si el archivo se cargó correctamente entonces dará un mensaje de *Program Succeeded.*

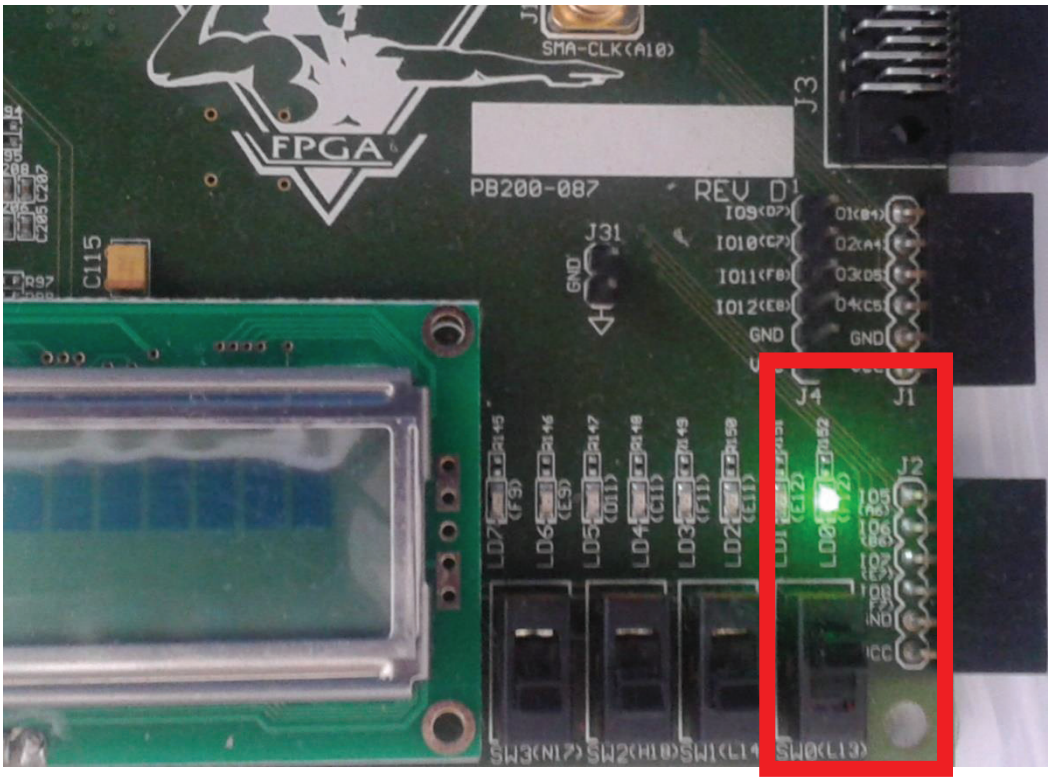


32. Finalmente comprobamos el funcionamiento del programa en la tarjeta.

32.1 Si el *switch 0* está en *'0'* entonces el *led 0* se encuentra apagado.



32.2 Si el *switch 0* está en *'1'* entonces el *led 0* se enciende.



ANEXO 17: PROGRAMA PARA EL CONTROL DOMÓTICO DESARROLLADO EN EL FPGA

```
----------------------------------------------------------------------------------
-- Company: Escuela Politécnica Nacional
-- Engineer: María Jima y Jonathan Zambrano
--
-- Create Date:    21:35:57 04/15/2014
-- Design Name:
-- Module Name:   PROTOCOLO X10 - Behavioral
-- Project Name:  DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE SISTEMA DOMÓTICO PARA PERMITIR EL CONTROL CENTRALIZADO DE DISPOSITIVOS DENTRO DEL HOGAR CONTROLADO POR FPGA UTILIZANDO EL PROTOCOLO X10.
-- Target Devices: Spartan 3E-Starter Board
-- Tool versions:
-- Description: Desarrollo del programa para la generación de los códigos del protocolo X10 para el control centralizado de la vivienda.
--
-- Dependencies: Facultad de Ingeniería Eléctrica y Electrónica
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;
use IEEE.std_logic_unsigned.all;



-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity PROTOCOLO_X10 is

Port ( clk : in  STD_LOGIC; --reloj de 50MHz del FPGA
       dato_entrada: in STD_LOGIC; --es el dato que recibira el FPGA por el puerto serial
        orden: out  STD_LOGIC:='1'; --es la orden que envia el FPGA para que que sea ejecutada en tiempo real o a largo plazo
      reloj: out  STD_LOGIC :='1'); --corresponde a la hora y los minutos actuales
end PROTOCOLO_X10;

architecture Behavioral of PROTOCOLO_X10 is

--Señales de Recepcion:

signal clk_rx : STD_LOGIC; --Reloj para recibir a 9600 baudios y con un muestreo de 8
signal act : STD_LOGIC := '0'; --Señal que se pone en 1 mientras esta recibiendo algun dato por el puerto serial
signal listo : STD_LOGIC :='0'; --Señal que se pone en 1 cuando ha terminado de recibir un dato de 8 bits por el puerto serial
signal cont_dato: integer:=0; --Es un contador que sirve para llevar un conteo de que procesos se deben realizar cada vez que se incrementa en uno dependiendo del numero de datos que reciba el FPGA por el puerto serial
signal comando: STD_LOGIC_VECTOR (7 downto 0):="00000000"; --Es una señal que guarda el primer dato que se recibe por el puerto serial y que indica el proceso que se desea realizar. Los procesos son A, B, C
signal labview : STD_LOGIC := '0'; --Es una señal que se activa cuando va a recibir un dato desde la interfaz de labview
```

```
signal dato_recepcion_min: integer; --Es una señal que guarda el minuto actual que se desea para el reloj del sistema domotico en formato de entero
signal dato_recepcion_min1: integer; --Es una señal que sirve para convertir a entero y guardar el minuto actual que se desea para el reloj del sistema domotico
signal dato_recepcion_min2: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que sirve para guardar el minuto actual del sistema domotico en formato de vector
signal dato_recepcion_hora: integer; --Es una señal que guarda la hora actual que se desea para el reloj del sistema domotico en formato de entero
signal dato_recepcion_hora1: integer; --Es una señal que sirve para convertir a entero y guardar la hora actual que se desea para el reloj del sistema domotico
signal dato_recepcion_hora2: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que sirve para guardar la hora actual del sistema domotico en formato de vector
signal hora_encendido1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda la hora a la cual se desea que se encienda algun dispositivo en formato de vector
signal minuto_encendido1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el minuto al cual se desea que se encienda algun dispositivo en formato de vector
signal hora_apagado1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda la hora a la cual se desea que se apague el dispositivo que fue encendido en formato de vector
signal minuto_apagado1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el minuto al cual se desea que se apague el dispositivo que fue encendido en formato de vector
signal hora_encendido2: integer; --Es una señal que convierte a entero y guarda la hora a la cual se desea que se encienda algun dispositivo
signal minuto_encendido2: integer; --Es una señal que convierte a entero y guarda el minuto al cual se desea que se encienda algun dispositivo
signal hora_apagado2: integer; --Es una señal que convierte a entero y guarda la hora a la cual se desea que se apague el dispositivo que fue encendido
signal minuto_apagado2: integer; --Es una señal que convierte a entero y guarda el minuto al cual se desea que se apague el dispositivo que fue encendido
signal hora_e: integer; --Es una señal que guarda la hora_encendido2 para poder comparar en los futuros procesos
signal minuto_e: integer; --Es una señal que guarda el minuto_encendido2 para poder comparar en los futuros procesos
signal hora_a: integer; --Es una señal que guarda la hora_apagado2 para poder comparar en los futuros procesos
signal minuto_a: integer; --Es una señal que guarda el minuto_apagado2 para poder comparar en los futuros procesos


--Señales de Transmision:

signal clk_tx : STD_LOGIC; --Reloj para transmitir a 9600 baudios
signal contador_tx : integer := 0; --Contador para la tx de los minutos y las horas
signal contador_control : integer:=0; --Contador para la tx de las instrucciones de la interfaz grafica
signal contador_programado : integer := 0; --Contador para la tx de las instrucciones de la interfaz grafica a largo plazo
signal aux : STD_LOGIC := '0'; --Señal que actua como bandera cuando se puede transmitir las instrucciones
signal borrar: STD_LOGIC := '0'; --Señal que se activa en 1 cuando se va a transmitir el codigo x10 correspondiente a una instruccion una sola vez y en tiempo real
signal aux2: STD_LOGIC := '0'; --Señal que actua como bandera cuando se deben transmitir las instrucciones a largo plazo
signal erase: STD_LOGIC := '0'; --Señal que se activa en 1 cuando se va a transmitir el codigo x10 una sola vez en la hora y minuto programado
signal bandera: STD_LOGIC := '0'; --Es una señal que se activa cuando es la hora y minuto preciso para realizar la tx de las ordenes. Permite el inicio del contador_programado
signal contador: integer:=0; --Es un contador que permite enviar una sola vez en el momento exacto (hora, minuto y segundo) las ordenes programadas por el usuario


--Señales del reloj

signal clk_real : STD_LOGIC; --Reloj para segundos, minutos y horas
signal segundo : integer:=0; --Es un contador para los segundos y se cuenta de 0 a 59 segundos
signal minuto1:integer:=11; --Es un contador para los minutos, cada 60 segundos se cuenta un minuto. Se empieza en 11 hasta 70 porque en labview no se puede visualizar el 10
signal minuto2: STD_LOGIC_VECTOR (7 downto 0); --Es el vector de minuto1 para poder tx serialmente
signal minuto: STD_LOGIC_VECTOR (7 downto 0); --Es un vector que guarda a minuto2 para tx por el puerto serial
```

```
signal hora1:integer:=11; --Es un contador para las horas, cada 60 minutos se cuenta una hora. Se empieza en 11 hasta 34 porque en labview no se puede visualizar el 10
signal hora2: STD_LOGIC_VECTOR (7 downto 0); --Es el vector de hora1 para poder tx serialmente
signal hora: STD_LOGIC_VECTOR (7 downto 0); --Es un vector que guarda a hora2 para tx por el puerto


--Señales que permiten la designacion posterior de códigos X10

signal instruccion1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el dato que contiene la instruccion que se desea que se realice. Sirve además para condicionar cuando se desea tx la instruccion inicializando el contador_control.
signal instruccion: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda la instruccion1 cuando debe transmitirla o se encera cuando no tiene nada que tx

signal luces1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda la informacion acerca de que luces se desea que se enciendan y apaguen dependiendo de los bits del vector luces1 que se encuentren en 1
signal luces: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que en la hora y minuto programado permite la desginacion los valores correspondientes a las luces de la casa para que posteriormente se le asigne el codigo X10 para que se enciendan o se apaguen en el momento programado
signal cortinas1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda la informacion acerca de que cortinas se desea que se enciendan y apaguen dependiendo de los bits del vector cortinas1 que se encuentren en 1
signal cortinas: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que en la hora y minuto programado permite la desginacion los valores correspondientes a las cortinas de la casa para que posteriormente se le asigne el codigo X10 para que se abran o se cierren en el momento programado

signal luz_sala_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se encienda la luz de la sala en la hora y minuto programado de encendido cuando el bit 0 de luces es 1
signal luz_sala_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz de la sala en la hora y minuto programado de encendido cuando el bit 0 de luces es 1
signal luz_cuarto1_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se encienda la luz del cuarto 1 en la hora y minuto programado de apagado cuando el bit 1 de luces es 1
signal luz_cuarto1_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz del cuarto 1 en la hora y minuto programado de apagado cuando el bit 1 de luces es 1
signal luz_cuarto2_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que seencienda la luz del cuarto 2 en la hora y minuto programado de apagado cuando el bit 2 de luces es 1
signal luz_cuarto2_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz del cuarto 2 en la hora y minuto programado de apagado cuando el bit 2 de luces es 1
signal luz_cocina_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se encienda la luz de la cocina en la hora y minuto programado de encendido cuando el bit 3 de luces es 1
signal luz_cocina_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz de la cocina en la hora y minuto programado de encendido cuando el bit 3 de luces es 1
signal luz_baño_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se encienda la luz del baño en la hora y minuto programado de encendido cuando el bit 4 de luces es 1
signal luz_baño_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz del baño en la hora y minuto programado de encendido cuando el bit 4 de luces es 1
signal luz_atico_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se encienda la luz del atico en la hora y minuto programado de encendido cuando el bit 5 de luces es 1
signal luz_atico_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz del atico en la hora y minuto programado de encendido cuando el bit 5 de luces es 1
signal luz_entrada_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se encienda la luz de la calle en la hora y minuto programado de encendido cuando el bit 6 de luces es 1
```

```
signal luz_entrada_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se apague la luz de la calle en la hora y minuto programado de encendido cuando el bit 6 de luces es 1
signal cortina_sala_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se abra la cortina de la sala en la hora y minuto programado de encendido cuando el bit 0 de cortinas es 1
signal cortina_sala_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se cierre la cortina de la sala en la hora y minuto programado de encendido cuando el bit 0 de cortinas es 1
signal cortina_cuarto1_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se abra la cortina de la sala en la hora y minuto programado de encendido cuando el bit 1 de cortinas es 1
signal cortina_cuarto1_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se cierre la cortina de la sala en la hora y minuto programado de encendido cuando el bit 1 de cortinas es 1
signal cortina_cuarto2_ON: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se abra la cortina de la sala en la hora y minuto programado de encendido cuando el bit 2 de cortinas es 1
signal cortina_cuarto2_OFF: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que guarda el código para que se cierre la cortina de la sala en la hora y minuto programado de encendido cuando el bit 2 de cortinas es 1


--Señales de asignacion de codigos X10

signal trama1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal a la cual se le asigna el código X10 CASA, es decir la primera señal X10 que sera transmitida en base a la instruccion dada
signal trama2: STD_LOGIC_VECTOR (7 downto 0); --Es una señal a la cual se le asigna el código X10 UNIDAD, es decir la segunda señal X10 que sera transmitida en base a la instruccion dada
signal trama3: STD_LOGIC_VECTOR (7 downto 0); --Es una señal a la cual se le asigna el código X10 FUNCION, es decir la tercera señal X10 que sera transmitida en base a la instruccion dada
signal orden_1: STD_LOGIC_VECTOR (7 downto 0); --Es una señal a la cual se le asigna el código X10 CASA, es decir la primera señal X10 que sera transmitida a la hora y minuto programado
signal orden_2: STD_LOGIC_VECTOR (7 downto 0); --Es una señal a la cual se le asigna el código X10 UNIDAD, es decir la segunda señal X10 que sera transmitida a la hora y minuto programado
signal orden_3: STD_LOGIC_VECTOR (7 downto 0); --Es una señal a la cual se le asigna el código X10 FUNCION, es decir la tercera señal X10 que sera transmitida a la hora y minuto programado
signal pausa: STD_LOGIC_VECTOR (7 downto 0); --Es una señal que sirve para resetear la memoria del puerto de tx y hacer que tx lo que se desea que tx


begin



--Configuracion del reloj para la recepcion de la informacion a 9600 baudios y muestreado a 8

reloj_rx: process(clk)
variable clk_rx_aux : STD_LOGIC := '0'; --Variable auxiliar que guarda el reloj para recibir serialmente a 9600 baudios y con muestreo de 8
variable cont_rx: natural range 0 to 325 := 0; --Es un contador resultante de dividir 50MHz/9600/8/2 para poder recibir a 9600 baudios y con muestreo de 8
          begin
          if rising_edge(clk) then
          if cont_rx = 0 then
             cont_rx:= 325;
             clk_rx_aux := not clk_rx_aux;
          else
             cont_rx := cont_rx-1;
          end if;
          end if;
             clk_rx <= clk_rx_aux;
          end process;


--Procesos de recepcion:

recepcion: process(clk_rx,dato_entrada)
variable cont_rx_aux: integer := 0; --Es un contador que al recibir un dato se va incrementando hasta empezar en 10 un conteo que permite la recepcion muestreada de 8 en 8
```

```
variable cont_muestreo : integer := 0; --Es un contador que cuenta de 1 hasta 8 permitiendo el
muestreo para recibir la informacion. Al terminar el muestreo de un dato se termina la
recepcion
variable dato_aux : STD_LOGIC_VECTOR (7 downto 0); --Es un vector que va guardando el dato que
se recibe por el puerto serial hasta q termina de recibir y lo guarda en un vector d 8 bits
para usarlo en los procesos posteriores

           begin

           if rising_edge(clk_rx) then


           if (dato_entrada = '0' and cont_rx_aux = 0) then
           act <= '1';
           labview<='1';
           end if;
           if (act = '1') then
           cont_rx_aux := cont_rx_aux + 1;
           end if;

           if labview='1' then
            if (cont_rx_aux = 10 + 8*cont_muestreo) then
                if (cont_muestreo < 8) then
                dato_aux(cont_muestreo) := dato_entrada;
                end if;
            cont_muestreo := cont_muestreo + 1;
            end if;
           end if;


           if (cont_muestreo = 9) then
           act <= '0';
           listo <= '1';
           labview<='0';
           cont_rx_aux := 0;
           cont_muestreo := 0;
              end if;

           if listo = '1' then
                   cont_dato<=cont_dato+1;
            if cont_dato=1 then
            comando<= dato_aux;
            listo <= '0';
            end if;
            if cont_dato=2 then
               if comando = "01000001" then
               instruccion1<=dato_aux;
               cont_dato<=0;
               end if;
               if comando = "01000010" then
               luces1<=dato_aux;
               end if;
               if comando = "01000011" then
               dato_recepcion_min2<= dato_aux;
               dato_recepcion_min1<=conv_integer(dato_recepcion_min2);
               end if;
               listo <= '0';
            end if;
                if cont_dato=3 then
               if comando = "01000010" then
               cortinas1<=dato_aux;
               end if;
               if comando = "01000011" then
               dato_recepcion_hora2<= dato_aux;
               dato_recepcion_hora1<=conv_integer(dato_recepcion_hora2);
               cont_dato<=0;
               end if;
                   listo <= '0';
                end if;
            if cont_dato=4 then
```

```
               if comando = "01000010" then
               minuto_encendido1<=dato_aux;
               minuto_encendido2<=conv_integer(minuto_encendido1);
               end if;
               listo <='0';
            end if;
            if cont_dato=5 then
               if comando = "01000010" then
               hora_encendido1<=dato_aux;
               hora_encendido2<=conv_integer(hora_encendido1);
               end if;
                  listo <='0';
            end if;
            if cont_dato=6 then
               if comando = "01000010" then
               minuto_apagado1<=dato_aux;
               minuto_apagado2<=conv_integer(minuto_apagado1);
               end if;
               listo <='0';
            end if;
            if cont_dato=7 then
               if comando = "01000010" then
                  hora_apagado1<=dato_aux;
               hora_apagado2<=conv_integer(hora_apagado1);
               cont_dato<=0;
               end if;
                  listo <='0';
            end if;
         end if;


end if;
end process;

--Señales que guardan los datos que se recibieron y que serán utilizados en otros procesos

dato_recepcion_min<=dato_recepcion_min1;
dato_recepcion_hora<=dato_recepcion_hora1;
hora_e<=hora_encendido2;
minuto_e<=minuto_encendido2;
hora_a<=hora_apagado2;
minuto_a<=minuto_apagado2;



--Configuracion del reloj para transmitir ordenes en tiempo actual una sola vez

tx_instruccion:process(clk_tx)
            begin

            if rising_edge(clk_tx) then

               if cont_dato=2 and comando="01000001" then
               aux<='1';
                end if;

            if instruccion1/="00000000" then
                  if contador_control < 60 then
               if aux='1' then
               contador_control <= contador_control+1;
               borrar<='1';
               end if;
               else
               aux<='0';
               contador_control<=0;
            borrar<='0';

            end if;
            else
            contador_control<=0;
```

```
            borrar<='0';
            end if;
            end if;
end process;



--Asignacion del dato que corresponde a la instruccion recibida en tiempo real
instruccion <= instruccion1 when borrar='1' else
      "00000000";

--Asignacion del codigo X10 a las tramas dependiendo de la instruccion que se recibe en tiempo
real


trama1<="01101001" when instruccion = "00000001" else --luz sala ON
        "01101001" when instruccion = "11110001" else --luz sala OFF
        "01101001" when instruccion = "10010001" else --luz sala BRIGHT
        "01101001" when instruccion = "01100001" else --luz sala DIM
        "01101001" when instruccion = "11000001" else --cortina sala ON
        "01101001" when instruccion = "00110001" else --cortina sala OFF
        "01101001" when instruccion = "10100001" else --cortina sala BRIGHT
        "01101001" when instruccion = "01010001" else --cortina sala DIM
        "10101001" when instruccion = "00000010" else --luz cuarto1 ON
        "10101001" when instruccion = "11110010" else --luz cuarto1 OFF
        "10101001" when instruccion = "10010010" else --luz cuarto1 BRIGHT
        "10101001" when instruccion = "01100010" else --luz cuarto1 DIM
        "10101001" when instruccion = "11000010" else --cortina cuarto1 ON
        "10101001" when instruccion = "00110010" else --cortina cuarto1 OFF
        "10101001" when instruccion = "10100010" else --cortina cuarto1 BRIGHT
        "10101001" when instruccion = "01010010" else --cortina cuarto1 DIM
        "01011001" when instruccion = "00000011" else --luz cuarto2 ON
        "01011001" when instruccion = "11110011" else --luz cuarto2 OFF
        "01011001" when instruccion = "10010011" else --luz cuarto2 BRIGHT
        "01011001" when instruccion = "01100011" else --luz cuarto2 DIM
        "01011001" when instruccion = "11000011" else --cortina cuarto2 ON
        "01011001" when instruccion = "00110011" else --cortina cuarto2 OFF
        "01011001" when instruccion = "10100011" else --cortina cuarto2 BRIGHT
        "01011001" when instruccion = "01010011" else --cortina cuarto2 DIM
        "10011001" when instruccion = "00000100" else --luz cocina ON
        "10011001" when instruccion = "11110100" else --luz cocina OFF
        "10011001" when instruccion = "10010100" else --luz cocina BRIGHT
        "10011001" when instruccion = "01100100" else --luz cocina DIM
        "01010110" when instruccion = "00000101" else --luz baño ON
        "01010110" when instruccion = "11110101" else --luz baño OFF
        "01010110" when instruccion = "10010101" else --luz baño BRIGHT
        "01010110" when instruccion = "01100101" else --luz baño DIM
        "10010110" when instruccion = "00000110" else --luz atico ON
        "10010110" when instruccion = "11110110" else --luz atico OFF
        "10010110" when instruccion = "10010110" else --luz atico BRIGHT
        "10010110" when instruccion = "01100110" else --luz atico DIM
        "01100110" when instruccion = "00000111" else --luz entrada ON
        "01100110" when instruccion = "11110111" else --luz entrada OFF
        "01100110" when instruccion = "11100111" else --puerta entrada ON
        "01100110" when instruccion = "00010111" else --puerta entrada OFF
        "10100110" when instruccion = "11101000" else --luz-puerta garage ON
        "10100110" when instruccion = "00011000" else --luz-puerta garage OFF
        "01101010" when instruccion = "00001001" else --luces casa ON
        "01101010" when instruccion = "11111001" else --luces casa OFF
        "00000000";

trama2<="01101001" when instruccion = "00000001" else --luz sala ON
        "01101001" when instruccion = "11110001" else --luz sala OFF
        "01101001" when instruccion = "10010001" else --luz sala BRIGHT
        "01101001" when instruccion = "01100001" else --luz sala DIM
        "10101001" when instruccion = "11000001" else --cortina sala ON
        "10101001" when instruccion = "00110001" else --cortina sala OFF
        "10101001" when instruccion = "10100001" else --cortina sala BRIGHT
        "10101001" when instruccion = "01010001" else --cortina sala DIM
        "01011001" when instruccion = "00000010" else --luz cuarto1 ON
```

```
        "01011001" when instruccion = "11110010" else --luz cuarto1 OFF
        "01011001" when instruccion = "10010010" else --luz cuarto1 BRIGHT
        "01011001" when instruccion = "01100010" else --luz cuarto1 DIM
        "10011001" when instruccion = "11000010" else --cortina cuarto1 ON
        "10011001" when instruccion = "00110010" else --cortina cuarto1 OFF
        "10011001" when instruccion = "10100010" else --cortina cuarto1 BRIGHT
        "10011001" when instruccion = "01010010" else --cortina cuarto1 DIM
        "01010110" when instruccion = "00000011" else --luz cuarto2 ON
        "01010110" when instruccion = "11110011" else --luz cuarto2 OFF
        "01010110" when instruccion = "10010011" else --luz cuarto2 BRIGHT
        "01010110" when instruccion = "01100011" else --luz cuarto2 DIM
        "10010110" when instruccion = "11000011" else --cortina cuarto2 ON
        "10010110" when instruccion = "00110011" else --cortina cuarto2 OFF
        "10010110" when instruccion = "10100011" else --cortina cuarto2 BRIGHT
        "10010110" when instruccion = "01010011" else --cortina cuarto2 DIM
        "01100110" when instruccion = "00000100" else --luz cocina ON
        "01100110" when instruccion = "11110100" else --luz cocina OFF
        "01100110" when instruccion = "10010100" else --luz cocina BRIGHT
        "01100110" when instruccion = "01100100" else --luz cocina DIM
        "10100110" when instruccion = "00000101" else --luz baño ON
        "10100110" when instruccion = "11110101" else --luz baño OFF
        "10100110" when instruccion = "10010101" else --luz baño BRIGHT
        "10100110" when instruccion = "01100101" else --luz baño DIM
        "01101010" when instruccion = "00000110" else --luz atico ON
        "01101010" when instruccion = "11110110" else --luz atico OFF
        "01101010" when instruccion = "10010110" else --luz atico BRIGHT
        "01101010" when instruccion = "01100110" else --luz atico DIM
        "10101010" when instruccion = "00000111" else --luz entrada ON
        "10101010" when instruccion = "11110111" else --luz entrada OFF
        "01011010" when instruccion = "11100111" else --puerta entrada ON
        "01011010" when instruccion = "00010111" else --puerta entrada OFF
        "10010101" when instruccion = "11101000" else --luz-puerta garage ON
        "10010101" when instruccion = "00011000" else --luz-puerta garage OFF
        "01010101" when instruccion = "00001001" else --luces casa ON
        "01010101" when instruccion = "11111001" else --luces casa OFF
        "00000000";

trama3<="01011001" when instruccion = "00000001" else --luz sala ON
        "01011010" when instruccion = "11110001" else --luz sala OFF
        "01100110" when instruccion = "10010001" else --luz sala BRIGHT
        "01100101" when instruccion = "01100001" else --luz sala DIM
        "01011001" when instruccion = "11000001" else --cortina sala ON
        "01011010" when instruccion = "00110001" else --cortina sala OFF
        "01100110" when instruccion = "10100001" else --cortina sala BRIGHT
        "01100101" when instruccion = "01010001" else --cortina sala DIM
        "01011001" when instruccion = "00000010" else --luz cuarto1 ON
        "01011010" when instruccion = "11110010" else --luz cuarto1 OFF
        "01100110" when instruccion = "10010010" else --luz cuarto1 BRIGHT
        "01100101" when instruccion = "01100010" else --luz cuarto1 DIM
        "01011001" when instruccion = "11000010" else --cortina cuarto1 ON
        "01011010" when instruccion = "00110010" else --cortina cuarto1 OFF
        "01100110" when instruccion = "10100010" else --cortina cuarto1 BRIGHT
        "01100101" when instruccion = "01010010" else --cortina cuarto1 DIM
        "01011001" when instruccion = "00000011" else --luz cuarto2 ON
        "01011010" when instruccion = "11110011" else --luz cuarto2 OFF
        "01100110" when instruccion = "10010011" else --luz cuarto2 BRIGHT
        "01100101" when instruccion = "01100011" else --luz cuarto2 DIM
        "01011001" when instruccion = "11000011" else --cortina cuarto2 ON
        "01011010" when instruccion = "00110011" else --cortina cuarto2 OFF
        "01100110" when instruccion = "10100011" else --cortina cuarto2 BRIGHT
        "01100101" when instruccion = "01010011" else --cortina cuarto2 DIM
        "01011001" when instruccion = "00000100" else --luz cocina ON
        "01011010" when instruccion = "11110100" else --luz cocina OFF
        "01100110" when instruccion = "10010100" else --luz cocina BRIGHT
        "01100101" when instruccion = "01100100" else --luz cocina DIM
        "01011001" when instruccion = "00000101" else --luz baño ON
        "01011010" when instruccion = "11110101" else --luz baño OFF
        "01100110" when instruccion = "10010101" else --luz baño BRIGHT
        "01100101" when instruccion = "01100101" else --luz baño DIM
        "01011001" when instruccion = "00000110" else --luz atico ON
```

```
        "01011010" when instruccion = "11110110" else --luz atico OFF
        "01100110" when instruccion = "10010110" else --luz atico BRIGHT
        "01100101" when instruccion = "01100110" else --luz atico DIM
        "01011001" when instruccion = "00000111" else --luz entrada ON
        "01011010" when instruccion = "11110111" else --luz entrada OFF
        "01011010" when instruccion = "11100111" else --puerta entrada ON
        "01011001" when instruccion = "00010111" else --puerta entrada OFF
        "01011010" when instruccion = "11101000" else --luz-puerta garage ON
        "01011001" when instruccion = "00011000" else --luz-puerta garage OFF
        "01010110" when instruccion = "00001001" else --luces casa ON
        "01010101" when instruccion = "11111001" else --luces casa OFF
        "00000000";

-- Configuracion para hacer el reloj

dia: process(clk_real,dato_recepcion_min,dato_recepcion_hora, listo, comando, minuto1, hora1)

   begin

         if rising_edge(clk_real) then
            if segundo < 59 then
            segundo <= segundo + 1;
            else
            segundo <= 0;
               if minuto1 < 70 then
                   minuto1 <= minuto1+1;
                   else
                   minuto1 <= 11;
                  if hora1 < 34 then
                  hora1<=hora1+1;
                  else
                  hora1<=11;
                  end if;
                      end if;
                  end if;
                        end if;

                      if listo='1' and (comando="01000011" and minuto1/= dato_recepcion_min)
                      then
         minuto1<=dato_recepcion_min;
         end if;


                      if listo='1' and (comando="01000011" and hora1/= dato_recepcion_hora)
                      then
         hora1<=dato_recepcion_hora;
         end if;


minuto2<=conv_std_logic_vector(minuto1,8);
hora2<=conv_std_logic_vector(hora1,8);

end process;


--Configuración del reloj para transmitir a 9600 baudios


reloj_tx: process(clk)
variable cont_tx : natural range 0 to 2604 := 0; --Es un contador resultante de dividir
50MHz/9600//2 para poder transmitir a 9600 baudios
variable clk_tx_aux : STD_LOGIC := '0'; --Variable auxiliar que guarda el reloj para
transmitir serialmente a 9600 baudios
   begin
      if rising_edge(clk) then
         if cont_tx = 0 then
         cont_tx:= 2604;
         clk_tx_aux := not clk_tx_aux;
         else
         cont_tx := cont_tx-1;
```

```
        end if;
       end if;
   clk_tx <= clk_tx_aux;
   end process;



--Configuracion de un contador para transmitir el reloj actual e igualado

tx_salida1: process(clk_tx)
   begin
      if rising_edge(clk_tx) then
         if contador_tx < 36 then
         contador_tx <= contador_tx+1;
         else
                  contador_tx <= 0;
         end if;
      end if;
   end process;



--Configuración de un contador para sincronizar la salida de un solo dato a largo plazo

tx_hora_minuto: process(clk_tx)
   begin
      if rising_edge(clk_tx) then
         if contador < 10416 then --multiplico 2604 que es el contador de tx a 9600 baudios x 4
para poder tx una sola vez en un segundo
         contador <= contador + 1;
         else
                     contador<= 0;
         end if;
      end if;
   end process;

--Asignacion de la bandera dependiendo de la hora y minutos programados

bandera<='1' when ((hora_e = hora1 and minuto_e=minuto1 and segundo=0 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=1 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=2 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=3 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=4 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=5 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=6 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=7 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=8 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=9 and contador=0) or
          (hora_e = hora1 and minuto_e=minuto1 and segundo=10 and contador=0) or

          (hora_a = hora1 and minuto_a=minuto1 and segundo=0 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=1 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=2 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=3 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=4 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=5 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=6 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=7 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=8 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=9 and contador=0) or
          (hora_a = hora1 and minuto_a=minuto1 and segundo=10 and contador=0)) else
          '0';



--Configuracion del contador para transmisitir ordenes a  largo plazo

 tx_orden: process(clk_tx)

begin
```

```
    if rising_edge(clk_tx) then

            if bandera='1' then
            aux2<='1';
            end if;

            if luces1/="00000000" or cortinas1/="00000000" then
                  if contador_programado < 60 then
                  if aux2='1' then
                  contador_programado <= contador_programado+1;
                  erase<='1';
                  end if;
               else
               aux2<='0';
               contador_programado<=0;
               erase<='0';
               end if;
            else
            contador_programado<=0;
            erase<='0';
            end if;
    end if;
end process;

--Asignacion de las luces y cortinas dependiendo de la hora y minuto para que exista
repeticion en la tx

luces<= luces1 when erase='1' and ((hora_e = hora1 and minuto_e=minuto1 and segundo=0) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=1) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=2) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=3) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=4) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=5) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=6) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=7) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=8) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=9) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=10) or

                  (hora_a = hora1 and minuto_a=minuto1 and segundo=0) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=1) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=2) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=3) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=4) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=5) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=6) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=7) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=8) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=9) or
                  (hora_a = hora1 and minuto_a=minuto1 and segundo=10)) else
                           "00000000";

cortinas<=cortinas1 when erase='1' and ((hora_e = hora1 and minuto_e=minuto1 and segundo=0) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=1) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=2) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=3) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=4) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=5) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=6) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=7) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=8) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=9) or
                  (hora_e = hora1 and minuto_e=minuto1 and segundo=10) or

               (hora_a = hora1 and minuto_a=minuto1 and segundo=0) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=1) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=2) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=3) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=4) or
```

```
               (hora_a = hora1 and minuto_a=minuto1 and segundo=5) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=6) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=7) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=8) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=9) or
               (hora_a = hora1 and minuto_a=minuto1 and segundo=10)) else
              "00000000";

--Asignacion de los valores de lugares de la casa dependiendo de las luces y cortinas
activadas
luz_sala_ON <= "00000001" when luces(0)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=0) else "00000000";
luz_sala_OFF <= "11110001" when luces(0)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=0) else  "00000000";
luz_cuarto1_ON  <= "00000010" when luces(1)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=1) else     "00000000";
luz_cuarto1_OFF <= "11110010" when luces(1)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=1) else    "00000000";
luz_cuarto2_ON  <= "00000011" when luces(2)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=2) else "00000000";
luz_cuarto2_OFF <= "11110011" when luces(2)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=2) else     "00000000";
luz_cocina_ON    <= "00000100" when luces(3)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=3) else     "00000000";
luz_cocina_OFF  <= "11110100" when luces(3)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=3) else     "00000000";
luz_baño_ON     <= "00000101" when luces(4)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=4) else    "00000000";
luz_baño_OFF     <= "11110101" when luces(4)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=4) else     "00000000";
luz_atico_ON     <= "00000110" when luces(5)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=5) else     "00000000";
luz_atico_OFF    <= "11110110" when luces(5)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=5) else     "00000000";
luz_entrada_ON   <= "00000111" when luces(6)='1' and (hora_e = hora1 and minuto_e=minuto1 and
segundo=6) else     "00000000";
luz_entrada_OFF <= "11110111" when luces(6)='1' and (hora_a = hora1 and minuto_a=minuto1 and
segundo=6) else     "00000000";


cortina_sala_ON           <= "11000001" when cortinas(0)='1' and (hora_e = hora1 and
minuto_e=minuto1 and segundo=8) else   "00000000";
cortina_sala_OFF         <= "00110001" when cortinas(0)='1' and (hora_a = hora1 and
minuto_a=minuto1 and segundo=8) else   "00000000";
cortina_cuarto1_ON     <= "11000010" when cortinas(1)='1' and (hora_e = hora1 and
minuto_e=minuto1 and segundo=9) else  "00000000";
cortina_cuarto1_OFF   <= "00110010" when cortinas(1)='1' and (hora_a = hora1 and
minuto_a=minuto1 and segundo=9) else   "00000000";
cortina_cuarto2_ON     <= "11000011" when cortinas(2)='1' and (hora_e = hora1 and
minuto_e=minuto1 and segundo=10) else "00000000";
cortina_cuarto2_OFF   <= "00110011" when cortinas(2)='1' and (hora_a = hora1 and
minuto_a=minuto1 and segundo=10) else "00000000";

--Asignacion de códigos X10 dependiendo de los lugares que se desea que se enciendan o apaguen
a largo plazo

pausa<= "00000000";

orden_1<="01101001" when luz_sala_ON="00000001" and segundo=0 else
         "01101001" when luz_sala_OFF="11110001" and segundo=0 else
         "10101001" when luz_cuarto1_ON="00000010" and segundo=1 else
         "10101001" when luz_cuarto1_OFF="11110010" and segundo=1 else
         "01011001" when luz_cuarto2_ON ="00000011" and segundo=2 else
         "01011001" when luz_cuarto2_OFF="11110011" and segundo=2 else
        "10011001" when luz_cocina_ON ="00000100" and segundo=3 else
         "10011001" when luz_cocina_OFF="11110100" and segundo=3 else
         "01010110" when luz_baño_ON ="00000101" and segundo=4 else
         "01010110" when luz_baño_OFF="11110101" and segundo=4 else
         "10010110" when luz_atico_ON ="00000110" and segundo=5 else
         "10010110" when luz_atico_OFF="11110110" and segundo=5  else
```

```
          "01100110" when luz_entrada_ON ="00000111" and segundo=6 else
          "01100110" when luz_entrada_OFF="11110111" and segundo=6 else
          "01101001" when cortina_sala_ON ="11000001" and segundo=7 else
          "01101001" when cortina_sala_OFF="00110001" and segundo=7 else
          "10101001" when cortina_cuarto1_ON ="11000010" and segundo=8 else
          "10101001" when cortina_cuarto1_OFF="00110010" and segundo=8 else
          "10101001" when cortina_cuarto2_ON ="11000011" and segundo=9 else
          "10101001" when cortina_cuarto2_OFF="00110011" and segundo=9 else
          "00000000";



orden_2<="01101001" when luz_sala_ON="00000001" and segundo=0 else
          "01101001" when luz_sala_OFF="11110001" and segundo=0 else
          "01011001" when luz_cuarto1_ON="00000010" and segundo=1 else
          "01011001" when luz_cuarto1_OFF="11110010" and segundo=1 else
          "01010110" when luz_cuarto2_ON ="00000011" and segundo=2 else
          "01010110" when luz_cuarto2_OFF="11110011" and segundo=2 else
          "01100110" when luz_cocina_ON ="00000100" and segundo=3 else
          "01100110" when luz_cocina_OFF="11110100" and segundo=3 else
          "10100110" when luz_baño_ON ="00000101" and segundo=4 else
          "10100110" when luz_baño_OFF="11110101" and segundo=4 else
          "01101010" when luz_atico_ON ="00000110" and segundo=5 else
          "01101010" when luz_atico_OFF="11110110" and segundo=5 else
          "10101010" when luz_entrada_ON ="00000111" and segundo=6 else
          "10101010" when luz_entrada_OFF="11110111" and segundo=6 else
          "10101001" when cortina_sala_ON ="11000001" and segundo=7 else
          "10101001" when cortina_sala_OFF="00110001" and segundo=7 else
          "10101001" when cortina_cuarto1_ON ="11000010" and segundo=8 else
          "10101001" when cortina_cuarto1_OFF="00110010" and segundo=8 else
          "10011001" when cortina_cuarto2_ON ="11000011" and segundo=9 else
          "10011001" when cortina_cuarto2_OFF="00110011" and segundo=9 else
          "00000000";



orden_3<="01011001" when luz_sala_ON="00000001" and segundo=0 else
          "01011010" when luz_sala_OFF="11110001" and segundo=0 else
          "01011001" when luz_cuarto1_ON ="00000010" and segundo=1 else
          "01011010" when luz_cuarto1_OFF="11110010" and segundo=1 else
          "01011001" when luz_cuarto2_ON ="00000011" and segundo=2 else
          "01011010" when luz_cuarto2_OFF="11110011" and segundo=2 else
          "01011001" when luz_cocina_ON ="00000100" and segundo=3 else
          "01011010" when luz_cocina_OFF="11110100" and segundo=3 else
          "01011001" when luz_baño_ON ="00000101" and segundo=4 else
          "01011010" when luz_baño_OFF="11110101" and segundo=4 else
          "01011001" when luz_atico_ON ="00000110" and segundo=5 else
          "01011010" when luz_atico_OFF="11110110" and segundo=5 else
          "01011001" when luz_entrada_ON ="00000111" and segundo=6 else
          "01011010" when luz_entrada_OFF="11110111" and segundo=6 else
          "01011001" when cortina_sala_ON ="11000001" and segundo=7 else
          "01011010" when cortina_sala_OFF="00110001" and segundo=7 else
          "01011001" when cortina_cuarto1_ON ="11000010" and segundo=8 else
          "01011010" when cortina_cuarto1_OFF="00110010" and segundo=8 else
          "01011001" when cortina_cuarto2_ON ="11000011" and segundo=9 else
          "01011010" when cortina_cuarto2_OFF="00110011" and segundo=9 else
          "00000000";

-- Transmision de las instrucciones en tiempo real y programadas

salidas:process (instruccion, contador_control, trama1, trama2, trama3, contador_programado,
orden_1, orden_2, orden_3)

begin

if instruccion/="00000000" then

if contador_control=1 then
orden<='1';
end if;
```

```
if contador_control=2 then
orden<='0';
end if;
if contador_control=3 then
orden<=pausa(0);
end if;
if contador_control=4 then
orden<=pausa(1);
end if;
if contador_control=5 then
orden<=pausa(2);
end if;
if contador_control=6 then
orden<=pausa(3);
end if;
if contador_control=7 then
orden<=pausa(4);
end if;
if contador_control=8 then
orden<=pausa(5);
end if;
if contador_control=9 then
orden<=pausa(6);
end if;
if contador_control=10 then
orden<=pausa(7);
end if;
if contador_control=11 then
orden<='1';
end if;
if contador_control=12 then
orden<='0';
end if;
if contador_control=13 then
orden<=trama1(0);
end if;
if contador_control=14 then
orden<=trama1(1);
end if;
if contador_control=15 then
orden<=trama1(2);
end if;
if contador_control=16 then
orden<=trama1(3);
end if;
if contador_control=17 then
orden<=trama1(4);
end if;
if contador_control=18 then
orden<=trama1(5);
end if;
if contador_control=19 then
orden<=trama1(6);
end if;
if contador_control=20 then
orden<=trama1(7);
end if;
if contador_control=21 then
orden<='1';
end if;
if contador_control=22 then
orden<='0';
end if;
if contador_control=23 then
orden<=pausa(0);
end if;
if contador_control=24 then
orden<=pausa(1);
end if;
if contador_control=25 then
```

```
orden<=pausa(2);
end if;
if contador_control=26 then
orden<=pausa(3);
end if;
if contador_control=27 then
orden<=pausa(4);
end if;
if contador_control=28 then
orden<=pausa(5);
end if;
if contador_control=29 then
orden<=pausa(6);
end if;
if contador_control=30 then
orden<=pausa(7);
end if;
if contador_control=31 then
orden<='1';
end if;
if contador_control=32 then
orden<='0';
end if;
if contador_control=33 then
orden<=trama2(0);
end if;
if contador_control=34 then
orden<=trama2(1);
end if;
if contador_control=35 then
orden<=trama2(2);
end if;
if contador_control=36 then
orden<=trama2(3);
end if;
if contador_control=37 then
orden<=trama2(4);
end if;
if contador_control=38 then
orden<=trama2(5);
end if;
if contador_control=39 then
orden<=trama2(6);
end if;
if contador_control=40 then
orden<=trama2(7);
end if;
if contador_control=41 then
orden<='1';
end if;
if contador_control=42 then
orden<='0';
end if;
if contador_control=43 then
orden<=pausa(0);
end if;
if contador_control=44 then
orden<=pausa(1);
end if;
if contador_control=45 then
orden<=pausa(2);
end if;
if contador_control=46 then
orden<=pausa(3);
end if;
if contador_control=47 then
orden<=pausa(4);
end if;
if contador_control=48 then
orden<=pausa(5);
```

```
end if;
if contador_control=49 then
orden<=pausa(6);
end if;
if contador_control=50 then
orden<=pausa(7);
end if;
if contador_control=51 then
orden<='1';
end if;
if contador_control=52 then
orden<='0';
end if;
if contador_control=53 then
orden<=trama3(0);
end if;
if contador_control=54 then
orden<=trama3(1);
end if;
if contador_control=55 then
orden<=trama3(2);
end if;
if contador_control=56 then
orden<=trama3(3);
end if;
if contador_control=57 then
orden<=trama3(4);
end if;
if contador_control=58 then
orden<=trama3(5);
end if;
if contador_control=59 then
orden<=trama3(6);
end if;
if contador_control=60 then
orden<=trama3(7);
end if;

else

if orden_1/="00000000" then

if contador_programado=1 then
orden<='1';
end if;
if contador_programado=2 then
orden<='0';
end if;
if contador_programado=3 then
orden<=pausa(0);
end if;
if contador_programado=4 then
orden<=pausa(1);
end if;
if contador_programado=5 then
orden<=pausa(2);
end if;
if contador_programado=6 then
orden<=pausa(3);
end if;
if contador_programado=7 then
orden<=pausa(4);
end if;
if contador_programado=8 then
orden<=pausa(5);
end if;
if contador_programado=9 then
orden<=pausa(6);
end if;
if contador_programado=10 then
```

```
orden<=pausa(7);
end if;
if contador_programado=11 then
orden<='1';
end if;
if contador_programado=12 then
orden<='0';
end if;
if contador_programado=13 then
orden<=orden_1(0);
end if;
if contador_programado=14 then
orden<=orden_1(1);
end if;
if contador_programado=15 then
orden<=orden_1(2);
end if;
if contador_programado=16 then
orden<=orden_1(3);
end if;
if contador_programado=17 then
orden<=orden_1(4);
end if;
if contador_programado=18 then
orden<=orden_1(5);
end if;
if contador_programado=19 then
orden<=orden_1(6);
end if;
if contador_programado=20 then
orden<=orden_1(7);
end if;
end if;


if orden_2/="00000000" then
if contador_programado=21 then
orden<='1';
end if;
if contador_programado=22 then
orden<='0';
end if;
if contador_programado=23 then
orden<=pausa(0);
end if;
if contador_programado=24 then
orden<=pausa(1);
end if;
if contador_programado=25 then
orden<=pausa(2);
end if;
if contador_programado=26 then
orden<=pausa(3);
end if;
if contador_programado=27 then
orden<=pausa(4);
end if;
if contador_programado=28 then
orden<=pausa(5);
end if;
if contador_programado=29 then
orden<=pausa(6);
end if;
if contador_programado=30 then
orden<=pausa(7);
end if;
if contador_programado=31 then
orden<='1';
end if;
if contador_programado=32 then
```

```
orden<='0';
end if;

if contador_programado=33 then
orden<=orden_2(0);
end if;
if contador_programado=34 then
orden<=orden_2(1);
end if;
if contador_programado=35 then
orden<=orden_2(2);
end if;
if contador_programado=36 then
orden<=orden_2(3);
end if;
if contador_programado=37 then
orden<=orden_2(4);
end if;
if contador_programado=38 then
orden<=orden_2(5);
end if;
if contador_programado=39 then
orden<=orden_2(6);
end if;
if contador_programado=40 then
orden<=orden_2(7);
end if;
end if;



if orden_3/="00000000" then
if contador_programado=41 then
orden<='1';
end if;
if contador_programado=42 then
orden<='0';
end if;
if contador_programado=43 then
orden<=pausa(0);
end if;
if contador_programado=44 then
orden<=pausa(1);
end if;
if contador_programado=45 then
orden<=pausa(2);
end if;
if contador_programado=46 then
orden<=pausa(3);
end if;
if contador_programado=47 then
orden<=pausa(4);
end if;
if contador_programado=48 then
orden<=pausa(5);
end if;
if contador_programado=49 then
orden<=pausa(6);
end if;
if contador_programado=50 then
orden<=pausa(7);
end if;
if contador_programado=51 then
orden<='1';
end if;
if contador_programado=52 then
orden<='0';
end if;
if contador_programado=53 then
orden<=orden_3(0);
```

```
end if;
if contador_programado=54 then
orden<=orden_3(1);
end if;
if contador_programado=55 then
orden<=orden_3(2);
end if;
if contador_programado=56 then
orden<=orden_3(3);
end if;
if contador_programado=57 then
orden<=orden_3(4);
end if;
if contador_programado=58 then
orden<=orden_3(5);
end if;
if contador_programado=59 then
orden<=orden_3(6);
end if;
if contador_programado=60 then
orden<=orden_3(7);
end if;
end if;

end if;
end process;

--Configuracion del reloj del FPGA para conseguir un reloj en segundos, minutos y horas

reloj_real: process(clk)
variable cont_reloj : natural range 0 to 25000000 := 0; --Es un contador que permite el diseño
de un reloj real en el FPGA ya que permite la creacion de un reloj con segundos (50MHz/2 =
1seg)
variable clk_reloj : std_logic := '0'; --Es una señal auxiliar que guarda el reloj de un
segundo de duracion y que sirve para el diseño de un reloj real


   begin
      if rising_edge(clk) then
         if cont_reloj = 0 then
            cont_reloj:=25000000;
            clk_reloj := not clk_reloj;
         else
                 cont_reloj := cont_reloj-1;
         end if;
      end if;
      clk_real<=clk_reloj;
   end process;

--Transmision del reloj actual e igualado

minuto<=minuto2;
hora<=hora2;

reloj <='1' when contador_tx < 3 else
        '0' when contador_tx = 3 else
        '1' when contador_tx = 4 else
        '1' when contador_tx = 5 else
        '1' when contador_tx = 6 else
        '1' when contador_tx = 7 else
        '1' when contador_tx = 8 else
        '1' when contador_tx = 9 else
        '1' when contador_tx = 10 else
        '0' when contador_tx = 11 else
        '1' when contador_tx >11 and contador_tx < 14 else
        '0' when contador_tx =14 else
  minuto(0) when contador_tx = 15 else
  minuto(1) when contador_tx = 16 else
  minuto(2) when contador_tx = 17 else
  minuto(3) when contador_tx = 18 else
```

```
  minuto(4) when contador_tx = 19 else
  minuto(5) when contador_tx = 20 else
  minuto(6) when contador_tx = 21 else
  minuto(7) when contador_tx = 22 else
        '1' when contador_tx > 23 and contador_tx <26 else
        '0' when contador_tx = 26 else
  hora(0) when contador_tx = 27 else
  hora(1) when contador_tx = 28 else
  hora(2) when contador_tx = 29 else
  hora(3) when contador_tx = 30 else
  hora(4) when contador_tx = 31 else
  hora(5) when contador_tx = 32 else
  hora(6) when contador_tx = 33 else
  hora(7) when contador_tx = 34 else
    '1';


end Behavioral;
```

ANEXO 18: PROGRAMA PARA LA TRANSMISIÓN Y RECEPCIÓN DE LAS SEÑALES X10 DESARROLLADO EN LOS PICS 16F877A

- **PROGRAMA PARA LA TRANSMISIÓN DE LAS SEÑALES X10**

```
;*******************************************************************************
;                                PROGRAMA DE TRANSMISIÓN X10              *
;                                     *
;                                         TX.ASM                    *
;                                     *
;                                         TX 1.0                    *
;                                     *
;*******************************************************************************
    list    p=16F877a         ; Directiva que identifica al procesador
#include    <p16F877a.inc> ; Definiciones del Procesador
#include    "x10lib.inc"   ; Definicioes y Constantes X10
    errorlevel -302           ; Quita el mensaje "register not in bank0, check page ;     ;
                              bits".

;*******************************************************************************
;   ASIGNACIÓN DE ESPACIOS DE MEMORIA                                           *
;*******************************************************************************
 __config 0x2F42              ; Archivo de Configuración

;*******************************************************************************
;   Variables (Asignación de Espacios de Memoria)                    *
;*******************************************************************************
grp0           udata   0x20
X10Flags res   1        ; Banderas X10
NumTxBits   res   1        ; Contador de Bits Transmitidos
TxFunction  res   1        ; Código de Función
TxHouse     res   1        ; Código de Casa (0-15 == A-P)
TxUnit      res   1        ; Código de Unidad (0-15 == 1-16)
TxData0     res   1        ; Registros Temporales de Transmisión
TxData1     res   1
TxData2     res   1
TxData3     res   1
Conteo      res   1        ; Contador de Rotaciones de los Registros Temporales
Contador       res   1        ; Contador de Bytes recibidos via Com. Serial
Dato           res   1        ; Byte recibido via Com. Serial
_WREG    EQU   0x71     ; Registro Temporal WREG
_STATUS        EQU   0x72     ; Registro Temporal STATUS
_PCLATH     EQU   0x73  ; Registro Temporal PCLATH
_FSR     EQU   0x74  ; Registro Temporal FSR
;-------------------------------------------------------------------------------
; Código de Inicio                                -
;-------------------------------------------------------------------------------
STARTUP        code  0x00
      nop      ; 0x0000
      goto  Inicio   ; 0x0001
      nop            ; 0x0002
         nop         ; 0x0003
         org      0x04
      goto  RI ; 0x0004  (Dirección de las Rutinas de Interrupción)

;-------------------------------------------------------------------------------
RIcode      code  0x0005                          -
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; RI                                    -
;  Rutinas de Interrupción.                          -
;-------------------------------------------------------------------------------
;  Fuentes de Interrupción:                          -
```

```
;  TMR0:    Controla la duración de 1 ms de la señal portadora de 120 KHz.   -
;  RBO/INT: Detector de cruces por cero. Sincronización de la transmisión.   -
;-------------------------------------------------------------------------------------
RI
   banksel     PCLATH
      movwf    _WREG    ; Guardar el registro WREG
   movf     STATUS,W       ; Guardar el registro STATUS
   clrf     STATUS
   movwf    _STATUS
   movf     PCLATH, W      ; Guardar el registro PCLATCH
   movwf    _PCLATH
   movf     FSR, W         ; Guardar el registro FSR
   movwf    _FSR
   pagesel  Serial
      btfsc    PIR1,RCIF      ; ¿Interrupción por recepción Serial?
      call     Serial         ; Si, Llamar a subrutina de Com. Serial
      pagesel     ZeroCrossing
   btfsc    INTCON, INTF   ; ¿Interrupción Externa?
   call     ZeroCrossing   ; Si, Llamar a la Subrutina de Cruces por Cero
   pagesel     EndX10Envelope
   btfsc    INTCON, T0IF   ; ¿Interrupción del Timer 0?
   call     EndX10Envelope ; Si, Terminar transmisión de Portadora
      bsf      INTCON, GIE    ; Habilitar Interrupciones Globales
      banksel     PCLATCH
   movf     _FSR, W        ; Restaurar el registro FSR
   movwf    FSR
   movf     _PCLATH, W  ; Restaurar el registro PCLATCH
   movwf    PCLATH
   movf     _STATUS, W     ; Restaurar el registro STATUS
   movwf    STATUS
   swapf    _WREG, F       ; Restaurar el registro W
   swapf       _WREG, W
   retfie

;-------------------------------------------------------------------------------------
PROG1    code                                -
;-------------------------------------------------------------------------------------
;*************************************************************************************
;  SUBRUTINAS DE INICIO                                                              *
;*************************************************************************************
Inicio
      pagesel     InicTimer0     ; Subrutina para iniciar el Timer 0
   call     InicTimer0
   pagesel     InicPWM        ; Subrutina para iniciar el Timer 2
   call     InicPWM
   pagesel     InicZeroXDetect   ; Subrutina para iniciar el Detector de
                                     ; Cruces
   call     InicZeroXDetect
   pagesel     InicX10Variables     ; Subrutina para borrar variables a utilizar
   call     InicX10Variables
   pagesel     EnableInterrupts     ; Subrutina para habilitar Interrupciones
   call     EnableInterrupts
      pagesel  InicSerial     ; Subrutina para iniciar la com. serial
      call     InicSerial
      bcf      TRISA,0
      bcf      TRISA,1
      bcf      PORTA,0
      bsf      PORTA,1
      clrf     W
      movwf    TRISD
;-------------------------------------------------------------------------------------
;  PROGRAMA PRINCIPAL                              -
;-------------------------------------------------------------------------------------
Main
      movfw    TxHouse
      movwf    PORTD
      goto     Main

;-------------------------------------------------------------------------------------
; InicTimer0                                                                        -
```

```
;-------------------------------------------------------------------------------------
;  - El Timer0 es usado para determinar la duración del pulso de 1 ms de la señal  -
;  de 120 KHz.                                                    -
;  - Fosc = 7.3728 MHz. Con un pre-escalador de 64 se tiene:              -
;  - Conteo de 1ms con un valor de 0xE2 (.226) en el registro TMR0                -
;-------------------------------------------------------------------------------------
InicTimer0
      Banksel     OPTION_REG
      movlw    b'10010101'    ; No RBPU, Reloj Interno, Pre-escalador de
                                        ; 1:64
      movwf    OPTION_REG
      bcf      INTCON, T0IF   ; Limpiar Bandera de Interrupción por
                                        ; Desbordamiento
      bsf      INTCON, T0IE   ; Habilitar Interrupción del Timer0 por
                                        ; Desbordamiento
      return

;-------------------------------------------------------------------------------------
; InicPWM                                    -
;  - La Salida en RC2 esta inicialmente deshabilitada.                -
;  - Inicializar el PWM para producir la portadora de 120 KHz.         -
;  - Oscilador Principal es de 7.3728 MHz, Tosc = 1/7.3728 MHz             -
;  - Periodo PWM = 1/120kHz =  8.333us                        -
;  - Periodo PWM = [(PR2) + 1] * 4 * Tosc * (Pre-esclado TMR2)                    -
;           = [(0) + 1] * 4 * 1/7.3728MHz * (16) = 8.333us          -
;  - Por lo tanto, PR2 = 0, Pre-escalado TMR2 = 16              -
;  - Ciclo de Trabajo PWM al 50% = 8.333us/2 = DCxB9:DCxB0 * Tosc * (Pre-escalado
;        TMR2)                                    -
;                     DCxB9:DCxB0 = 8.333us/2/7.3728MHz/(16) = 2          -
;-------------------------------------------------------------------------------------
InicPWM
      banksel  TRISC
      bsf      XOUT     ;Salida deshabilitada
      banksel  TRISA
         bcf      TRISA,0
         bcf      TRISA,2
         bcf      PORTA,0
         bsf      PORTA,2
         banksel     CCP1CON
      clrf     CCP1CON     ; Modulo CCP deshabilitado
      banksel     TMR2
      clrf     TMR2     ; Limpiar registro TMR2
      banksel     PR2
      clrf     PR2      ; Setear Periodo PWM
      banksel     CCPR1L
      clrf     CCPR1L      ; Setear Ciclo de Trabajo PWM (1)
      banksel     PIE1
      clrf     PIE1     ;
      movlw    b'00101100'
      banksel     CCP1CON
      movwf    CCP1CON     ; Configurar el módulo CCP1 para PWM
      movlw    b'00000110'
      banksel     T2CON
      movwf    T2CON    ; Setear el pre-escalado y Habilitar el TMR2
      return

;-------------------------------------------------------------------------------------
; InicZeroXDetect                              -
;  - Detección de Cruces por cero utilizada para Sincronizar la Tx           -
;  - Utiliza la interrupción externa 0                      -
;-------------------------------------------------------------------------------------
InicZeroXDetect
      Banksel     OPTION_REG
      bsf      OPTION_REG, INTEDG   ; Setear la Interrupción por Ciclo
                  ; Positivo
      bsf      INTCON, INTE      ; Habilitar la Interrupción Externa
      return

;-------------------------------------------------------------------------------------
; InicX10Variables                                -
```

```
;  - Prepara registros a utilizar (Borra su contenido)                -
;-------------------------------------------------------------------------------------
InicX10Variables
      banksel     TxData0
         clrf     TxData0     ; Registros Temporales
      clrf     TxData1
      clrf     TxData2
      clrf     TxData3
      clrf     TxUnit      ; Dirección de Unidad
      clrf     TxHouse     ; Dirección de Casa
      clrf     TxFunction  ; Dirección de Función
      clrf     X10Flags ; Banderas X10
         clrf        Conteo      ; Contador de Rotaciones de los Registros
                                  ; Temporales
               clrf     Contador ; Contador de Bytes recibidos via Com.
                                  ; Serial
         clrf     Dato     ; Dato recibido por Com. Serial
      clrf     PORTA
         return

;-------------------------------------------------------------------------------------
; InicSerial                                    -
;  - Se utiliza la Com. Serial para recibir direcciones y comandos del FPGA        -
;  - Parámetros: 9600 Baudios, 8 Bits de Datos, sin paridad, 1 Bit de Parada.      -
;-------------------------------------------------------------------------------------
InicSerial
         banksel  TXSTA
         bcf      TXSTA,SYNC  ; Modo Asincrónico
         bsf      TXSTA,BRGH  ; Modo Asincrónico de Alta Velocidad
         banksel  SPBRG
         movlw    .46      ; Setear 9600 Baudios
         movwf    SPBRG
         banksel  RCSTA
         movlw    b'10010000'
         movwf    RCSTA    ; Habilitar Puerto Serial
         banksel  PIE1
         bsf      PIE1,RCIE   ; Habilitar Interrupción por Recepción
         return

;-------------------------------------------------------------------------------------
; EnableInterrupts                                -
;  - Habilita las interrupciones periféricas y globales             -
;-------------------------------------------------------------------------------------
EnableInterrupts
      banksel     PIR1
      clrf     PIR1     ; Borrar banderas de Interrupciones
                                  ; Periféricas
      clrf     PIR2
         bsf      INTCON, PEIE   ; Habilitar Interrupciones Periféricas
         bsf      INTCON, GIE ; Habilitar Interrupciones Globales
      return

;-------------------------------------------------------------------------------------
; Serial                                -
;  - Recibir Datos del FPGA                          -
;-------------------------------------------------------------------------------------
Serial
      bcf      PIR1,RCIF   ; Reponer Bandera de Interrupción Serial
      incf     Contador ; Aumentar contador de Tramas Recibidas
      movf     RCREG,W        ; Lectura del dato recibido
      movwf    Dato           ; Visualización del dato
      movlw    b'00000010'
      pagesel  CASA
      xorwf    Contador,W
      btfsc    STATUS,Z ; Primera Trama?
      goto     CASA     ; Si, Vaya a Casa
      movlw    b'00000100'
      bcf      STATUS,Z
      xorwf    Contador,W
      pagesel  UNIDAD
```

```
      btfsc    STATUS,Z ; Segunda Trama?
      goto     UNIDAD      ; Si, Vaya a Unidad
      movlw    b'00000110'
      bcf      STATUS,Z
      xorwf    Contador,W
      pagesel  ORDEN
      btfsc    STATUS,Z ; Tercera Trama?
      goto     ORDEN    ; Si, Vaya a Orden
      pagesel  FIN
      goto     FIN
CASA
      movfw    Dato
      movwf    TxHouse     ; Copiar la Trama a TxHouse
      bcf      STATUS,Z
      pagesel  FIN
      goto     FIN
UNIDAD
      movfw    Dato
      movwf    TxUnit   ; Copiar la Trama a TxUnit
      pagesel  FIN
      goto     FIN
ORDEN
      movfw    Dato
      movwf    TxFunction  ; Copiar la Trama a TxFunction
      clrf     Contador ; Borrar Contador
      call     PTx      ; Subrutina para Preparar la 1era Tx
      call     InitX10Tx   ; Preparar registros para la Tx
      bsf      X10TxFlag   ; Levantar Bandera de Tx
      bsf      PORTA,0
FIN
      return

;-------------------------------------------------------------------------------------
; Zero Crossing                                  -
;  - Transmitir Bit X10                                 -
;-------------------------------------------------------------------------------------
ZeroCrossing
   call     TransmitX10Bit ; Subrutina para Transmitir Bit X10
   pagesel  ZeroXReceive
   btfss    X10TxFlag   ; Hay un Bit X10 a Transmitir
   goto     ZeroXTransmit  ; Si, Vaya a X10Tx
      call     X10Tx
ZeroXTransmit
   call     ToggleInterruptEdge  ; Subrutina para Cambiar Ciclo de
                                                ; Interrupción
   bcf      INTCON, INTF      ; Limpiar bandera de Interrupción Externa
   return

;-------------------------------------------------------------------------------------
; EndX10Envelope                                 -
;  - Corta Transmisión de Portadora 120 KHz                  -
;-------------------------------------------------------------------------------------
EndX10Envelope
   banksel     TRISC
   bsf      XOUT     ; Deshabilitar la salida de 120 KHz
   bcf      INTCON, T0IE   ; Deshabilitar la Interrupción del Timer 0
   bcf      INTCON, T0IF   ; Limpiar la bandera de interrupción del Timer 0
      return

;-------------------------------------------------------------------------------------
; TransmitX10Bit                                 -
;  - Transmite el Bit X10 correspondiente                    -
;  - Si TxBit=1, habilita la salida de 120 KHz                  -
;  - Si TxBit=0, no hace nada                              -
;-------------------------------------------------------------------------------------
TransmitX10Bit
   pagesel     Tx0
   btfss    TxBit    ; El Bit a Transmitir es 1L?
   goto     Tx0      ; No, No haga nada.
Tx1
```

```
   banksel     TRISC
   bcf         XOUT     ; Si, Habilitar la salida de 120 KHz
   banksel     TMR0
   movlw    0xE2     ; Setear el registro TRM0 para conteo de 1 mS
   movwf    TMR0
   bcf      INTCON, T0IF   ; Limpiar bandera de Interrupción del TMR0
   bsf      INTCON, T0IE   ; Enable Timer0 interrupt.
Tx0
   bcf      TxBit    ; Limpiar el TxBit
   return

;-------------------------------------------------------------------------------------------
; InitX10Tx                                    -
;  - Rota los registros temporales 4 veces para preparar la transmisión      -
;-------------------------------------------------------------------------------------------
InitX10Tx
   movlw    .4
      call     RotateTxData   ; Subrutina para Rotar Registros
   movlw    .22            ; Setea el número de bits a transmitir
   movwf    NumTxBits
   bcf      SecondTxFlag   ; Limpia la bandera de 2da Tx
   return

;-------------------------------------------------------------------------------------------
; X10Tx                                  -
;  - Prepara el bit a transmitir                        -
;-------------------------------------------------------------------------------------------
X10Tx
      bcf      TxBit    ; Limpia el TxBit
   btfsc    TxData3, 7  ; El bit mas significativo de TxData3 esta en 1L?
   bsf      TxBit    ; Si, Setear TxBit
   bcf      STATUS, C   ; Limpiar el Carry
   btfsc    TxBit    ; Esta el TxBit en 1L?
   bsf      STATUS, C   ; Si, Setear el Carry
   rlf      TxData0, F  ; Rotar los registros temporales una vez a la
                                    ; izquierda
   rlf      TxData1, F
   rlf      TxData2, F
   rlf      TxData3, F
   pagesel     EndX10Tx
   decfsz      NumTxBits, F   ; Se transmitieron todos los bits?
   goto     EndX10Tx       ; No, Finalizar la rutina
   pagesel     Init2ndTx
   btfss    SecondTxFlag   ; Si, fue la segunda transmisión?
   goto     Inic2ndTx      ; No, Vaya a Inic2ndTx
   pagesel  Inic2ndBloque
      btfss    SecondTxBloq   ; Si, fue el segundo bloque?
      goto     Inic2ndBloque  ; No, Vaya a Inic2ndBloque
      bcf      X10TxFlag        ; Si, la Transmisión Terminó
      bcf      SecondTxFlag
      pagesel     InicX10Variables
   call     InicX10Variables ; Reiniciar Variables
   pagesel     EndX10Tx
   goto     EndX10Tx
Inic2ndBloque
      banksel  TxData0
      bcf      SecondTxFlag
      movlw    b'00001110'
      movwf    TxData3
      movfw    TxHouse
      movwf    TxData2
      movfw    TxFunction
      movwf    TxData1
      movlw    b'10000000'
      movwf    TxData0
      call     InitX10Tx
      bsf      SecondTxBloq
      pagesel  EndX10Tx
      goto     EndX10Tx
Inic2ndTx
```

```
   bsf      SecondTxFlag   ; Setear la bandera de segunda Tx
   movlw    .10
      call     RotateTxData   ; Reponer bits al final del buffer
   movlw    .28      ; Enviar 22 bits otra vez
   movwf    NumTxBits
EndX10Tx
   return
;-------------------------------------------------------------------------------------
; ToggleInterruptEdge                                  -
;  - Cambiar el Ciclo de Interrupción para cada cruce por cero         -
;-------------------------------------------------------------------------------------
ToggleInterruptEdge
   banksel     OPTION_REG
   pagesel     IntFallingEdge
   btfsc    OPTION_REG, INTEDG   ; Esta la interrupción en el ciclo positivo?
   goto     IntFallingEdge    ; Si, Vaya a IntFallingEdge
   bsf      OPTION_REG, INTEDG   ; No, Setear la interrupción por el ciclo
                                          ; positivo
   pagesel     EndToggleInterruptEdge
   goto     EndToggleInterruptEdge
IntFallingEdge
   bcf      OPTION_REG, INTEDG   ; Setear la interrupcion por el ciclo
                                          ; negativo
EndToggleInterruptEdge
   return

;-------------------------------------------------------------------------------------
; RotateTxData                                  -
;  - Rotar a la Izquierda los registros temporales las veces que indique el reg. W -
;-------------------------------------------------------------------------------------
RotateTxData
   movwf    Conteo
   bcf         STATUS, C   ; Limpiar el Carry
   btfsc    TxData3, 7  ; Esta en 1L el MSb de TxData3?
   bsf      STATUS, C   ; Si, setear el Carry
   rlf      TxData0, f  ; Rotar los registros temporales
   rlf      TxData1, f
   rlf      TxData2, f
   rlf      TxData3, f
   decfsz      Conteo, f   ; Termino el Conteo?
   goto     $-5      ; No, Volver a rotar
   return

;-------------------------------------------------------------------------------------
; PTx                                  -
;  - Preparar la primera transmisión                        -
;-------------------------------------------------------------------------------------
PTx
      movlw    b'00001110' ; Código de Inicio
      movwf    TxData3
      movfw    TxHouse     ; Código de Casa
      movwf    TxData2
      movfw    TxUnit      ; Código de Unidad
      movwf    TxData1
      movlw    b'01000000' ; Prefijo de Unidad
      movwf    TxData0
      return

;*************************************************************************************
; FIN DE PROGRAMA                                                                    *
;*************************************************************************************
   end      ;directiva 'Fin del Programa'
```

- **PROGRAMA PARA LA RECEPCIÓN DE LAS SEÑALES X10**

```
;*******************************************************************************
;                                PROGRAMA DE RECEPCIÓN X10             *
;                                     *
;                                         RX.ASM                    *
;                                     *
;                                         RX 1.0                    *
;                                     *
;*******************************************************************************
    list    p=16F877a         ; Directiva que identifica al procesador
#include    <p16F877a.inc> ; Definiciones del Procesador
#include    "x10lib.inc"   ; Definicioes y Constantes X10
    errorlevel -302           ; Quita el mensaje "register not in bank0, check page ;     ;
                              bits".
;*******************************************************************************
;   ASIGNACIÓN DE ESPACIOS DE MEMORIA                                           *
;*******************************************************************************
; CONFIG
 __config   0x2F41

;*******************************************************************************
;   Variables (Asignación de Espacios de Memoria)                   *
;*******************************************************************************
grp0           udata    0x20
X10Flags res   1  ; Banderas X10
RxFuncion   res   1  ; Código de Función
RxHouse     res   1  ; Código de Casa (0-15 == A-P)
RxUnit      res   1  ; Código de Unidad (0-15 == 1-16)
RxHouse1 res   1  ; Registros Temporales de Recepción
RxHouse2 res   1
RxKey1      res   1
RxKey2      res   1
RxData0     res   1
RxData1     res   1
RxData2     res   1
RxData3     res   1
Bandera        res   1  ; Bandera que indica primera/segunda transmisión
R1             res   1  ; Registros de retraso
R2             res   1
_WREG    EQU   0x71     ; Registro Temporal WREG
_STATUS        EQU   0x72     ; Registro Temporal STATUS
_PCLATH     EQU   0x73  ; Registro Temporal PCLATH
_FSR     EQU   0x74  ; Registro Temporal FSR
#define        Elegido        PORTA, 0 ; Indicador
;-------------------------------------------------------------------------------
; Código de Inicio                                  -
;-------------------------------------------------------------------------------
STARTUP        code  0x00
      nop      ; 0x0000
      goto  Inicio   ; 0x0001
      nop            ; 0x0002
         nop         ; 0x0003
         org      0x04
      goto  RI ; 0x0004  (Dirección de las Rutinas de Interrupción)
;-------------------------------------------------------------------------------
RIcode      code  0x0005                          -
;-------------------------------------------------------------------------------

;-------------------------------------------------------------------------------
; RI                                    -
;  Rutinas de Interrupción.                         -
;-------------------------------------------------------------------------------
;  Fuentes de Interrupción:                         -
;  RBO/INT: Detector de cruces por cero. Sincronización de la recepción.     -
;-------------------------------------------------------------------------------
RI
```

```
   banksel     PCLATH
      movwf    _WREG    ; Guardar el registro WREG
   movf     STATUS,W       ; Guardar el registro STATUS
   clrf     STATUS
   movwf    _STATUS
   movf     PCLATH, W      ; Guardar el registro PCLATCH
   movwf    _PCLATH
   movf     FSR, W         ; Guardar el registro FSR
   movwf    _FSR
   pagesel     ZeroCrossing
   btfsc    INTCON, INTF   ; ¿Interrupción Externa?
   call     ZeroCrossing   ; Si, Llamar a la Subrutina de Cruces por Cero
      bsf      INTCON, GIE    ; Habilitar Interrupciones Globales
      banksel     PCLATCH
   movf     _FSR, W        ; Restaurar el registro FSR
   movwf    FSR
   movf     _PCLATH, W  ; Restaurar el registro PCLATCH
   movwf    PCLATH
   movf     _STATUS, W     ; Restaurar el registro STATUS
   movwf    STATUS
   swapf    _WREG, F       ; Restaurar el registro W
   swapf       _WREG, W
   retfie

;-------------------------------------------------------------------------------------------
PROG1    code                                -
;-------------------------------------------------------------------------------------------
;*******************************************************************************************
;  SUBRUTINAS DE INICIO                                                                    *
;*******************************************************************************************
Inicio
      bcf     Elegido
Reinicio
   pagesel     InicZeroXDetect   ; Subrutina para iniciar el Detector de
                                       ; Cruces
   call     InicZeroXDetect
   pagesel     InicX10Variables     ; Subrutina para borrar variables a utilizar
   call     InicX10Variables
      pagesel  InicSerial     ; Subrutina para iniciar la com. serial
      call     InicSerial
   pagesel     EnableInterrupts     ; Subrutina para habilitar Interrupciones
   call     EnableInterrupts
      banksel  TRISA
      bcf      TRISA,0
      banksel  TRISC
      bsf      TRISC,3
      bcf      TRISC,6
      bsf      TRISC,7
      banksel  TRISB
      bsf      TRISB,0
      banksel  TRISD
      clrf     TRISD
      banksel  PORTD
      clrf     PORTD
      movlw    b'11111111'
      movwf    R1       ; Setear retardo
      movlw    b'11010111'
      movwf    R2

;-------------------------------------------------------------------------------------------
;  PROGRAMA PRINCIPAL                              -
;-------------------------------------------------------------------------------------------
Main
      banksel  X10Flags
      pagesel  Prueba
      btfsc    X10RxFlag      ; Bandera de recepción habilitada?
      call     Prueba         ; Si, Vaya a Prueba
      goto     Main
Prueba
      movfw    RxHouse
```

```
      xorlw    b'01101001'       ; Dirección de Casa del Dispositivo
      banksel  STATUS
      btfsc    STATUS,Z    ; Coincide la Dirección de Casa?
      goto     $+7         ; No, Termine la Subrutina
        movfw     RxHouse
      xorlw    b'01101010'       ; Dirección de Todas las Casas
      banksel  STATUS
      pagesel  ENDI
      btfss    STATUS,Z    ; Coincide la Dirección de Todas las Casas?
      goto     ENDI        ; No, Termine la Subrutina
      pagesel  Unidad
      banksel  Bandera
      btfss    Bandera,1      ; La Bandera está en 1L
      goto     Unidad         ; Si, Vaya a Unidad
      pagesel  Funcion
      goto     Funcion        ; No, Vaya a Función
Unidad
      movfw    RxUnit
      xorlw    b'01101001'       ; Dirección de Unidad del Dispositivo
      banksel  STATUS
         btfsc    STATUS,Z    ; Coincide la Dirección de Unidad?
      goto     $+7         ; No, Finalice la Subrutina
      movfw    RxUnit
      xorlw    b'01010101'       ; Dirección de Todas las Unidades
      banksel  STATUS
      pagesel  ENDI
      btfss    STATUS,Z    ; Coincide la Dirección de Todas las
                                          ; Unidades?
      goto     ENDI        ; No, Finalice la Subrutina
      banksel  Bandera
      bsf      Bandera, 1     ; Setear Bandera en 1L
      banksel  PORTA
      bsf      Elegido        ; Encienda Elegido
      pagesel  FINC
      goto     FINC
Funcion
      banksel  Bandera
      bcf      Bandera,1      ; Setear Bandera en 0L
      banksel  PORTB
      movfw    RxFunction
      movwf    TXREG       ; Enviar Comando vía Com. Serial
      banksel  PORTD
      movwf    PORTD
      pagesel  Reinicio
      goto     Reinicio    ; Reiniciar Programa
ENDI
      banksel  PORTA
      bcf      Elegido        ; Apagar Elegido
      banksel  PORTD
      clrf     PORTD       ; Limpiar Registros
      clrf     RxHouse
      clrf     RxUnit
FINC
      bcf      X10RxFlag      ; Limpiar Bandera de Recepción
      return

;-------------------------------------------------------------------------------------
; InicX10Variables                                    -
;  - Prepara registros a utilizar (Borra su contenido)                -
;-------------------------------------------------------------------------------------
InicX10Variables
   banksel     RxData0
   clrf     RxData0        ; Registros Temporales
   clrf     RxData1
   clrf     RxData2
   clrf     RxData3
      clrf     RxHouse1
   clrf     RxHouse2
   clrf     RxKey1
   clrf     RxKey2
```

```
   clrf        RxHouse        ; Dirección de Casa
      clrf     RxUnit         ; Dirección de Unidad
   clrf     X10Flags    ; Banderas X10
      clrf     RxFunction     ; Dirección de Función
      clrf     Bandera        ; Bandera de Recepción
      clrf     R1       ; Registros de Retraso
      clrf     R2
   return

;-------------------------------------------------------------------------------------------
; InicZeroXDetect                                -
;  - Detección de Cruces por cero utilizada para Sincronizar la Tx           -
;  - Utiliza la interrupción externa 0                       -
;-------------------------------------------------------------------------------------------
InicZeroXDetect
   banksel     OPTION_REG
   bsf      OPTION_REG, INTEDG   ; Setear la Interrupción por Ciclo
                 ; Positivo
   bsf      INTCON, INTE      ; Habilitar la Interrupción Externa
   return

;-------------------------------------------------------------------------------------------
; InicSerial                                    -
;  - Se utiliza la Com. Serial para enviar el comando recibido al actuador         -
;  - Parámetros: 9600 Baudios, 8 Bits de Datos, sin paridad, 1 Bit de Parada       -
;-------------------------------------------------------------------------------------------
InicSerial
      banksel  TXSTA
      movlw    b'00100100'    ; Habilitar la Transmisión Serial
      movwf    TXSTA
      banksel  SPBRG
      movlw    .22         ; Setear 9600 Baudios
      movwf    SPBRG
      banksel  RCSTA
      bsf      RCSTA,SPEN     ; Habilitar la Com. Serial
      return

;-------------------------------------------------------------------------------------------
; EnableInterrupts                                  -
;  - Habilita las interrupciones periféricas y globales             -
;-------------------------------------------------------------------------------------------
EnableInterrupts
   banksel     PIR1
   clrf     PIR1        ; Borrar banderas de Interrupciones
                                       ; Periféricas
   clrf     PIR2
      bsf      INTCON, PEIE      ; Habilitar Interrupciones Periféricas
      bsf      INTCON, GIE    ; Habilitar Interrupciones Globales
   return

;-------------------------------------------------------------------------------------------
; Zero Crossing                                 -
;  - Recibir Bit X10                               -
;-------------------------------------------------------------------------------------------
ZeroCrossing
   call     ReceiveX10Bit     ; Recibir el RxBit
      call     X10Rx       ; Procesar Bit Recibido
      call     ToggleInterruptEdge  ; Cambiar Ciclo de Interrupción
   bcf      INTCON, INTF      ; Limpiar Bandera de Interrupción
   return

;-------------------------------------------------------------------------------------------
; ReceiveX10Bit                                     -
;  - Si la portadora de 120 KHz está presente, se setea RxBit en 1L          -
;  - Si no se detecta la portadora de 120 KHz, se setea RxBit en 0L              -
;-------------------------------------------------------------------------------------------
ReceiveX10Bit
       bcf       RxBit       ; Limpar el RxBit
      pagesel  Retardo
      call     Retardo        ; Subrutina de Retardo
```

```
      btfsc    XIN              ; Se detecta la portadora de 120 KHz?
   bsf      RxBit       ; Si, Setear el RxBit en 1L
      Return

;-------------------------------------------------------------------------------------
; X10Rx                                        -
;  - Almacena en registros temporales los Bits X10 recibidos.                    -
;  - Busca el Código de Inicio para Guardar los registros recibidos          -
;-------------------------------------------------------------------------------------
X10Rx
   banksel  STATUS
      bcf      STATUS, C   ; Limpia el Carry
   btfsc    RxBit    ; Esta el RxBit en 1L
   bsf      STATUS, C   ; Si, setear el Carry en 1L
   banksel  RxData0
      rlf      RxData0, F     ; Rotar los registros temporales
   rlf      RxData1, F
   rlf      RxData2, F
   rlf      RxData3, F
   movf     RxData3, W
   andlw    b'00001111'
   xorlw    b'00001110'
   pagesel     EndX10Rx
   btfss    STATUS, Z   ; Esta presente el código de inicio? '1110'
   goto     EndX10Rx ; No, Finalice la Subrutina
; Si se detectó el Código de Inicio, se procede con la siguiente rutina:
   movf     RxHouse1, W
   movwf    RxHouse2
   movf     RxKey1, W
   movwf    RxKey2
   movf     RxData2, W
   movwf    RxHouse1
   movf     RxData1, W
   movwf    RxKey1
   movf     RxHouse1, W
   xorwf    RxHouse2, W
   pagesel     EndX10Rx
   btfss    STATUS, Z   ; Coinciden los últimos códigos de Casa Recibidos?
   goto     EndX10Rx ; No, Finalice la Subrutina
   movf        RxHouse1,W
      movwf    RxHouse
      movf     RxKey1, W
   xorwf    RxKey2, W
   pagesel     EndX10Rx
   btfss    STATUS, Z   ; Coinciden los últimos códigos de Unidad/Función
                                    ; Recibidos?
       goto   EndX10Rx ; No, Finalice la Subrutina
   movf     RxData0, W
   andlw    b'11000000'
   xorlw    b'01000000'
   pagesel     AddressRcvd
   btfsc    STATUS, Z   ; Esta presente el sufijo de Unidad?
   goto     AddressRcvd ; Si, Se recibió una Dirección de Unidad
   movf     RxData0, W
   andlw    b'11000000'
   xorlw    b'10000000'
   pagesel     CommandRcvd
   btfsc    STATUS, Z   ; Esta presente el sufijo de Función?
   goto     CommandRcvd ; Si, Se recibió un Comando
   pagesel     EndX10Rx
   goto     EndX10Rx
AddressRcvd
      movf     RxKey2, W
      movwf    RxUnit
   banksel     X10Flags
      bcf      RxCommandFlag  ; Limpiar Bandera de Comando Recibido
   bsf      X10RxFlag
   pagesel     EndX10Rx
   goto     EndX10Rx
CommandRcvd
```

```
      movf     RxKey2, W
   movwf    RxFuncion
   banksel  X10Flags
      bsf      RxCommandFlag  ; Setear Bandera de Comando Recibido en 1L
   bsf      X10RxFlag   ; Setear Bandera de Recepción en 1L
      pagesel     EndX10Rx
      goto     EndX10Rx
EndX10Rx
      return

;-------------------------------------------------------------------------------------------
; Retardo                                       -
;  - Permite que el microcontrolador lea el Puerto en el momento correcto.         -
;-------------------------------------------------------------------------------------------
Retardo
      decfsz   R1    ; Decrementar R1, R2 Veces
      goto     $-1
      movlw    b'11111111'
      movwf    R1
      movlw    b'11010111'
         movwf    R2
      return

;-------------------------------------------------------------------------------------------
; ToggleInterruptEdge                                   -
;  - Cambiar el Ciclo de Interrupción para cada cruce por cero         -
;-------------------------------------------------------------------------------------------
ToggleInterruptEdge
   banksel     OPTION_REG
   pagesel     IntFallingEdge
   btfsc    OPTION_REG, INTEDG   ; Esta la interrupción en el ciclo positivo?
   goto     IntFallingEdge    ; Si, Vaya a IntFallingEdge
   bsf      OPTION_REG, INTEDG   ; No, Setear la interrupción por el ciclo
                                          ; positivo
   pagesel     EndToggleInterruptEdge
   goto     EndToggleInterruptEdge
IntFallingEdge
   bcf      OPTION_REG, INTEDG   ; Setear la interrupción por el ciclo
                                          ; negativo
EndToggleInterruptEdge
   return

;*******************************************************************************************
; FIN DE PROGRAMA                                                                          *
;*******************************************************************************************
   end      ;directiva 'Fin del Programa'
```

ANEXO 19: PROGRAMA PARA EL CONTROL DE ILUMINACIÓN DESARROLLADO EN EL MICROCONTROLADOR ATMEGA 8535

```
;*****************************************************************************************
;                              PROGRAMA DE CONTROL DE ILUMINACIÓN              *
;                                        *
;                                          LUZ.ASM                      *
;                                        *
;                                          LUZ 1.0                      *
;                                        *
;*****************************************************************************************
.include "m8535def.inc"
       .org 0x00
       rjmp inicio
       .org 0x01
       rjmp exter
       .org 0x02
       rjmp cambio
       .org 0x0B
       rjmp serial

;-----------------------------------------------------------------------------------------
; Código de Inicio                                  -
;-----------------------------------------------------------------------------------------
inicio:
      ldi      r16,high(ramend)
      out      SPH,r16
      ldi      r16,low(ramend)
      out      SPL,r16
;*****************************************************************************************
; Configuraciones Iniciales                                        *
;*****************************************************************************************
      clr      r16         ; Configuración de Puertos
      out      ddra,r16
      ser      r16
      out      ddrb,r16
      out      ddrc,r16
      cbi      ddrd,2
      sbi      ddrb,3
      cbi      ddrd,0
      cbi      ddrd,6
      cbi      ddrd,3

;-----------------------------------------------------------------------------------------
; Interrupciones Externas                                -
;  - Se configura la Interrupción Externa 0, para la detección de cruces por cero  -
;  - Se configura la Interrupción Externa 1, para el control manual de luz         -
;-----------------------------------------------------------------------------------------
      ldi      r16,0b11000000
      out      gicr, r16      ; Activar Interrupción Externa 0 y 1
      ldi      r16, 0b00000111   ; Configurar Int0 por Ciclo Positivo
      out      mcucr, r16     ; Configurar Int1 para cualquier
                                                ; Ciclo

;-----------------------------------------------------------------------------------------
; Conversor A/D                                     -
;  - Utilizado para el control manual de Luz                    -
;-----------------------------------------------------------------------------------------
      ldi      r16,0b01100000    ; Voltaje de Referencia en AVCC
      out      admux, r16     ; Utilizar Puerto ADC0
      ldi      r16,0b10000000
      out      adcsra,r16     ; Habilitar la conversión A/D

;-----------------------------------------------------------------------------------------
; PWM                                                    -
;  - Utilizado para el control de Luz                           -
;-----------------------------------------------------------------------------------------
```

```
      ldi      r16,0b01101100    ; Pre-escalado = 256
      out      tccr0,r16      ; Funcionamiento como PWM
      ldi      r16,128
      out      ocr0,r16    ; Setear nivel de luz al 50%

;-----------------------------------------------------------------------------------
; Comunicación Serial                                        -
;  - Utilizado para recibir y enviar los comandos de función              -
;  - Parámetros: 9600 Baudios, 1 Bit de parada, sin paridad            -
;-----------------------------------------------------------------------------------
      clr      r16
      out      UBRRH,r16
      ldi      r16,0b00011001
      out      UBRRL,r16      ; Setear 9600 Baudios
      clr      r16
      out      ucsra,r16      ; Limpiar Banderas Com. Serial
      ldi      r16,0b10011000    ; Habilitar Tx y Rx
      out      ucsrb,r16      ; Habilitar Interrupción por Rx
      ldi      r16,0b10000110
      out      ucsrc,r16      ; Com. Asincronica, 8 Bits

;-----------------------------------------------------------------------------------
; Constantes                                          -
;  - Valores de los comandos a recibir                      -
;-----------------------------------------------------------------------------------
      ldi      r20,85         ; Comando Todas Luz OFF
      ldi      r21,86         ; Comando Todas Luz ON
      ldi      r22,89         ; Comando Luz ON
      ldi      r23,90         ; Comando Luz OFF
      ldi      r24,63         ; Incremento/Decremento
      ldi      r25,102        ; Comando Bright
      ldi      r26,101        ; Comando Dim
      sei               ; Activar Interrupciones Globales

;-----------------------------------------------------------------------------------
;  PROGRAMA PRINCIPAL                                -
;-----------------------------------------------------------------------------------
Main:
      sbi      adsc,adcsra    ; Iniciar conversión A/D
      rjmp     Main

;-----------------------------------------------------------------------------------
; Subrutina Interrupción Externa 0                          -
;-----------------------------------------------------------------------------------
exter:
      clr      r16
      out      tcnt0,r16      ; Iniciar PWM
      ldi      r16,0b01101100
      out      tccr0,r16      ;
      sbrs     r27,0       ; Control Manual?
      rjmp     endi
      in    r16,adcl    ; Si, setear valor del conversor A/D
      in    r16,adch
      com      r16
      out      ocr0,r16
endi:
      reti

;-----------------------------------------------------------------------------------
; Subrutina Interrupción Externa 1                           -
;-----------------------------------------------------------------------------------
cambio:
      sbr      r27,1       ; Levantar Bandera de Control Manual
      reti

;-----------------------------------------------------------------------------------
; Subrutina Comunicación Serial                            -
;-----------------------------------------------------------------------------------
serial:
      in    r16,udr        ; Tomar trama de Puerto USART
```

```
      mov      r17,r16
      cp    r16,r26        ; Comando Dim recibido?
      breq     Subir       ; Si, Vaya a Subir
      mov      r16,r17
      cp    r16,r25        ; Comando Bright recibido?
      breq     Bajar       ; Si, Vaya a Bajar
      mov      r16,r17
      cp    r16,r23        ; Comando Luz OFF recibido?
      breq     Prender        ; Si, Vaya a Prender
      mov      r16,r17
      cp    r16,r20        ; Comando Todos Luz OFF recibido?
      breq     Prender        ; Si, Vaya a Prender
      mov      r16,r17
      cp    r16,r22        ; Comando Luz ON recibida?
      breq     Apagar         ; Si, Vaya a Apagar
      mov      r16,r17
      cp    r16,r21        ; Comando Luz Todos ON recibida?
      breq     Apagar         ; Si, Vaya a Apagar
      rjmp     fin
Subir:
      in    r16,ocr0
      add      r16,r24        ; Aumentar ancho de PWM
      brcs     uno         ; Llego al máximo?
      rjmp     set1
uno:
               ldi      r16,255        ; Si, setear salida en 1L
      out      portb,r16
      rjmp     set1
set1:
               out      ocr0,r16
      rjmp     fin

Bajar:
      in    r16,ocr0
      sub      r16,r24        ; Disminuir ancho PWM
      brcs     dos         ; Llego al mínimo?
      rjmp     set2
dos:
               ldi      r16,0
      out      portb,r16      ; Si, setear la salida en 0L
      rjmp     set2
set2:
               out      ocr0,r16
      rjmp     fin

Prender:
      ser      r16
      out      ocr0,r16    ; Setear la salida en 1L
      rjmp     fin
Apagar:
      clr      r16
      out      ocr0,r16    ; Setear la salida en 0L
fin:
               clr      r27
      mov      r16,r17
      out      udr,r16        ; Enviar Comando en Cascada
      reti
```

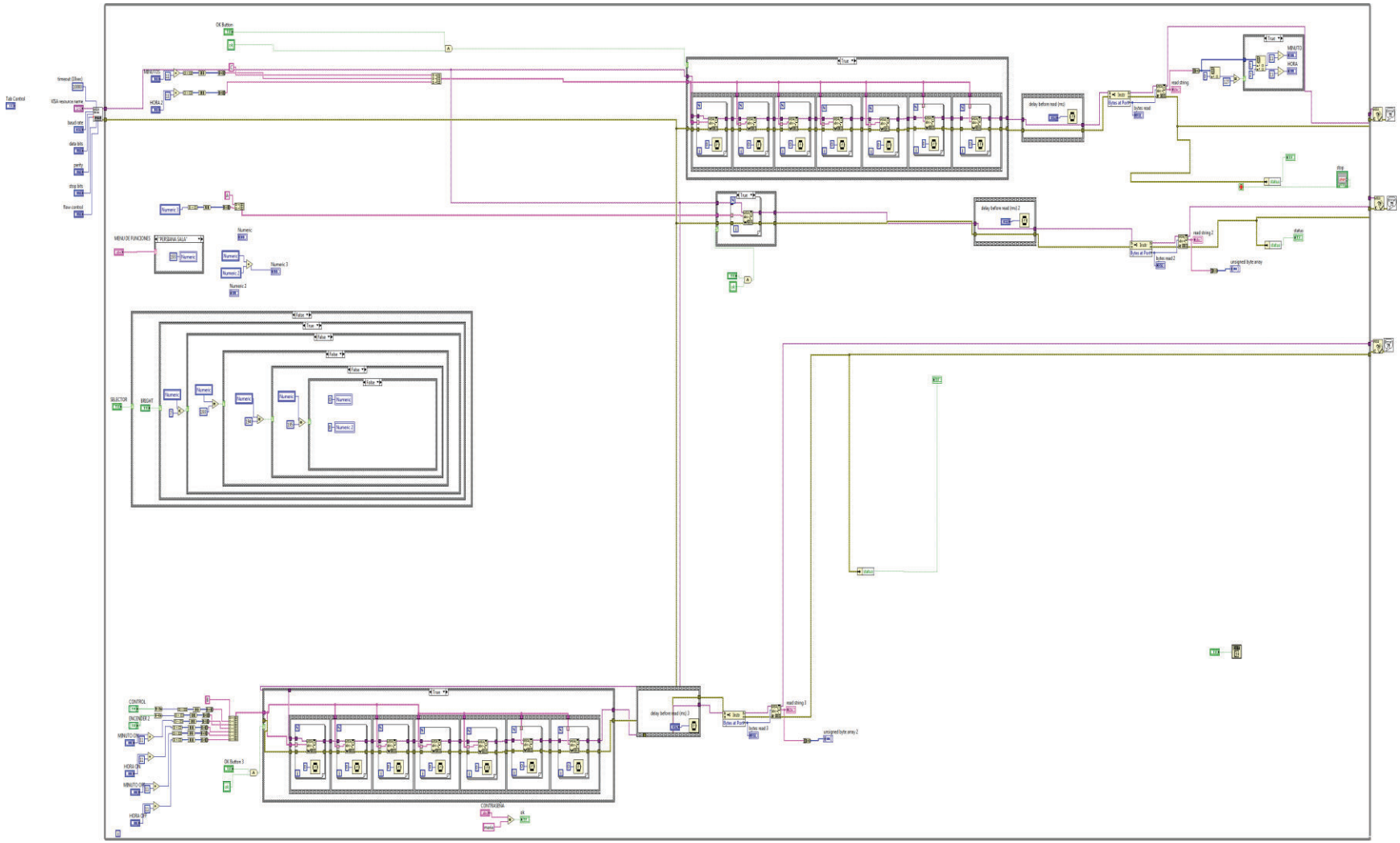
ANEXO 20: PROGRAMA PARA EL CONTROL DE ACCESOS Y PERSIANAS DESARROLLADO EN EL MICROCONTROLADOR ATMEGA 8535

```
;*****************************************************************************************
;                          PROGRAMA PARA EL CONTROL DE ACCESOS Y PERSIANAS              *
;                                      *
;                                          MOTOR.ASM                    *
;                                      *
;                                          MOTOR 1.0                    *
;                                      *
;*****************************************************************************************
.include "m8535def.inc"
       .org    0x00
       rjmp    inicio
       .org    0x0B
       rjmp    serial

;-----------------------------------------------------------------------------------------
; Código de Inicio                                -
;-----------------------------------------------------------------------------------------
inicio:
      ldi   r16,high(ramend)
      out   sph,r16
      ldi   r16,low(ramend)
      out   spl,r16

;*****************************************************************************************
; Configuraciones Iniciales                                *
;*****************************************************************************************
      clr   r16         ; Configuración de Puertos
      out   ddra,r16
      sbi   ddrb,3
      cbi   ddrd,0
      cbi   ddrd,2

;-----------------------------------------------------------------------------------------
; Comunicación Serial                                    -
;  - Utilizado para recibir y enviar los comandos de función              -
;  - Parámetros: 9600 Baudios, 1 Bit de parada, sin paridad            -
;-----------------------------------------------------------------------------------------
      clr   r16
      out   UBRRH,r16
      ldi   r16,51
      out   UBRRL,r16      ; Setear 9600 Baudios
      clr   r16
      out   UCSRA,r16      ; Limpiar Banderas de Com. Serial
      ldi   r16,0b10011000    ; Habilitar Tx y Rx
      out   UCSRB,r16      ; Habilitar Interrupción por Recepción
      ldi   r16,0b10000110
      out   UCSRC,r16      ; Com. Asincronica, 8 Bits
;-----------------------------------------------------------------------------------------
; Constantes                                        -
;  - Valores de los comandos a recibir                        -
;-----------------------------------------------------------------------------------------
      ldi   r21,89         ; Comando Motor ABRIR
      ldi   r22,90         ; Comando Motor CERRAR
      ldi   r23,102        ; Comando Motor Subir
      ldi   r24,101        ; Comando Motor Bajar
      clr   r26
      clr   r16
      sei            ; Activar Interrupciones Globales

;-----------------------------------------------------------------------------------------
;  PROGRAMA PRINCIPAL                              -
;-----------------------------------------------------------------------------------------
```

```
Main:    sbrs  r26,7       ; Hay comando de Motor?
      rjmp  Main
      sbrs  r26,6       ; Si, En qué sentido debe girar?
      rjmp  DERECHA
      rjmp  IZQUIERDA
      rjmp  Main
DERECHA:
      dec   r19         ; Girar el motor a la derecha
      breq  NX1
      rjmp  motorS
NX1:
            ldi   r19,255
      dec   r20         ; Determina el número de vueltas del motor
      breq  FN1
      rjmp  motorS
FN1:
            clr   r26
      clr   r16
      out   porta,r16
      rjmp  Main
IZQUIERDA:
      dec   r19         ; Gira el motor a la izquierda
      breq  NX2
      rjmp  motorB
NX2:
            ldi   r19,255
      dec   r20         ; Determina el número de vueltas del motor
      breq  FN2
      rjmp  motorB
FN2:
            clr   r26
      clr   r16
      out   porta,r16
      rjmp  Main
motorS:
            ldi   r16,0b00000110    ; Girar Derecha
      out   porta,r16
      rcall retardo
      ldi   r16,0b00001100
      out   porta,r16
      rcall retardo
      ldi   r16,0b00011000
      out   porta,r16
      rcall retardo
      ldi   r16,0b00010010
      out   porta,r16
      rcall retardo
      rjmp  Main
motorB:
            ldi   r16,0b00010010    ; Girar Izquierda
      out   porta,r16
      rcall retardo
      ldi   r16,0b00011000
      out   porta,r16
      rcall retardo
      ldi   r16,0b00001100
      out   porta,r16
      rcall retardo
      ldi   r16,0b00000110
      out   porta,r16
      rcall retardo
      rjmp  Main
retardo:
      ldi   r17,0b11111111    ; Retardo para alimentar bobinas del motor
      ldi   r18,0b00100000
decremen1:
      dec   r17
      breq  segundo
      rjmp  decremen1
segundo:
```

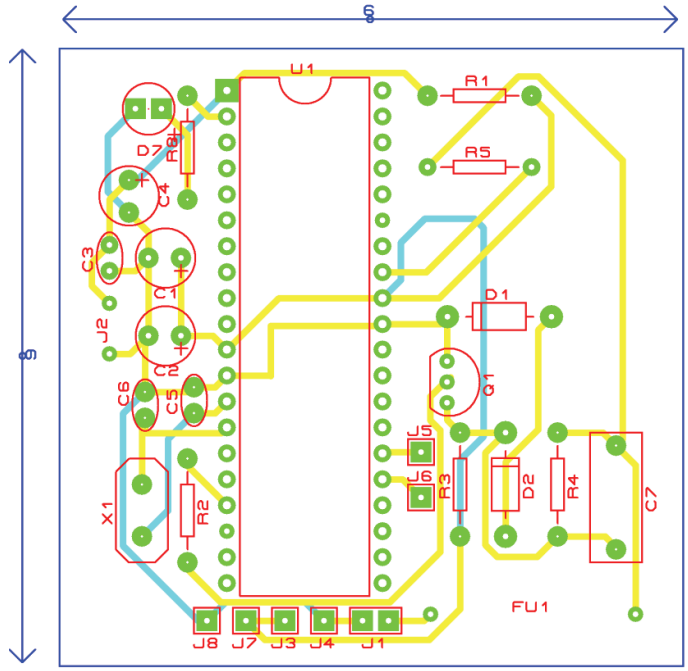
```
      dec   r18
      breq  salir
      ldi   r17,0b11111111
      rjmp  decremen1
salir:
              ret

;-----------------------------------------------------------------------------------------
; Subrutina Comunicación Serial                          -
;-----------------------------------------------------------------------------------------
serial:
              in r27,UDR
      cp r27,r21        ; Comando Motor ABRIR recibida?
      breq  ENCENDER    ; Si, Vaya a Encender
      cp r27,r22        ; Comando Motor CERRAR recibida?
      breq  APAGAR         ; Si, Vaya a Apagar
      cp r27,r23        ; Comando Motor SUBIR recibida?
      breq  SUBIR       ; Si, Vaya a Subir
      cp r27,r24        ; Comando Motor BAJAR recibida?
      breq  BAJAR       ; Si, Vaya a Bajar
      rjmp  end
ENCENDER:
      ldi   r19,255
      ldi   r20,7       ; Setear 4 ciclos de movimiento
      ldi   r16,0b11000000
      mov   r26,r16        ; Bandera Motor a la Derecha
      rjmp  end
APAGAR:
      ldi   r19,255
      ldi   r20,7       ; Setear 4 ciclos de movimiento
      ldi   r16,0b10000000
      mov   r26,r16        ; Bandera Motor a la Izquierda
      rjmp  end
SUBIR:
              ldi   r19,223
      ldi   r20,2       ; Setear 1 Ciclo de movimiento
      ldi   r16,0b11000000
      mov   r26,r16        ; Bandera Motor a la derecha
      rjmp  end
BAJAR:
              ldi   r19,223
      ldi   r20,2       ; Setear 1 Ciclo de movimiento
      ldi   r16,0b10000000
      mov   r26,r16        ; Bandera Motor a la izquierda
end:
              ser   r16
      out   ddra,r16
      mov   r16,r27
      out   udr,r16        ; Enviar commando en cascada
      reti
```

ANEXO 21: PROGRAMA DE LA INTERFAZ GRÁFICA DE USUARIO DESARROLLADO EN EL LABView

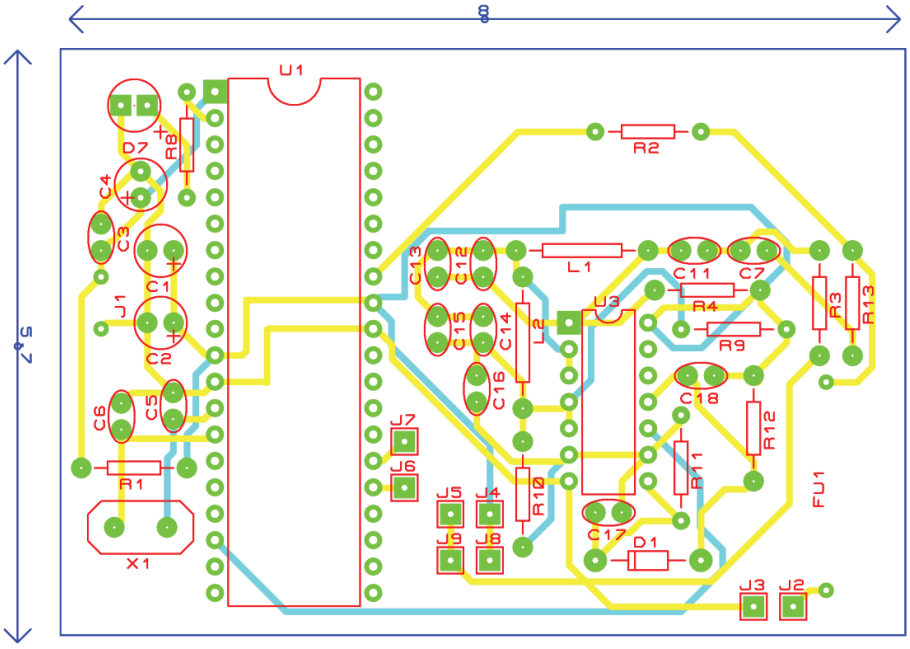


ANEXO 22: DIAGRAMAS DE LOS CIRCUITOS IMPRESOS (PCB)

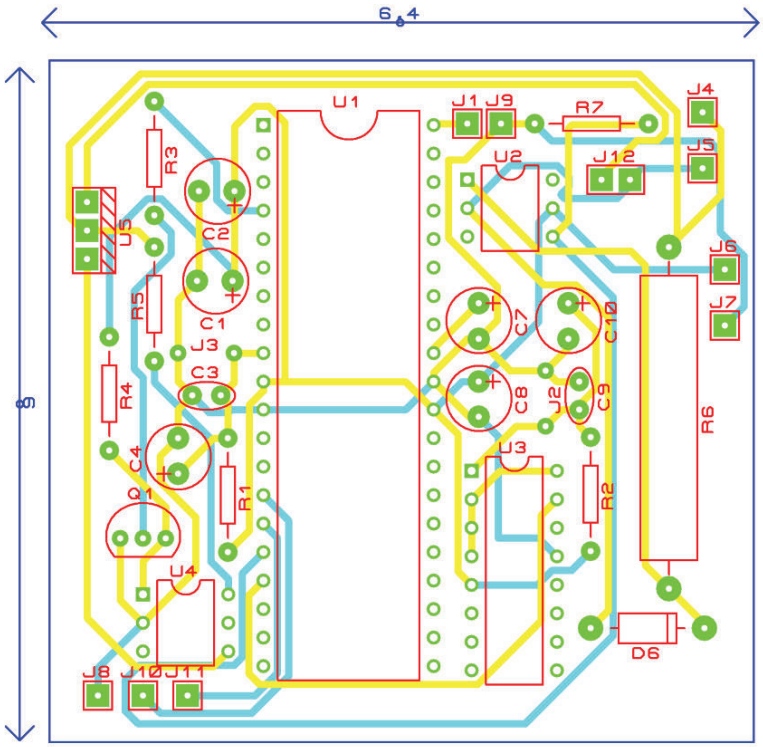
- **DIAGRAMA DEL CIRCUITO PARA LA TRANSMISIÓN DE SEÑALES X10**



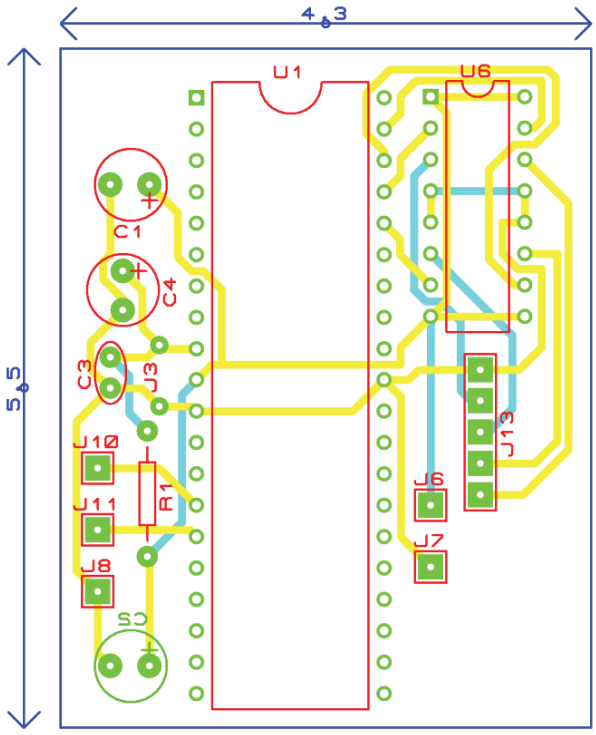
- **DIAGRAMA DEL CIRCUITO PARA LA RECEPCIÓN DE SEÑALES X10**



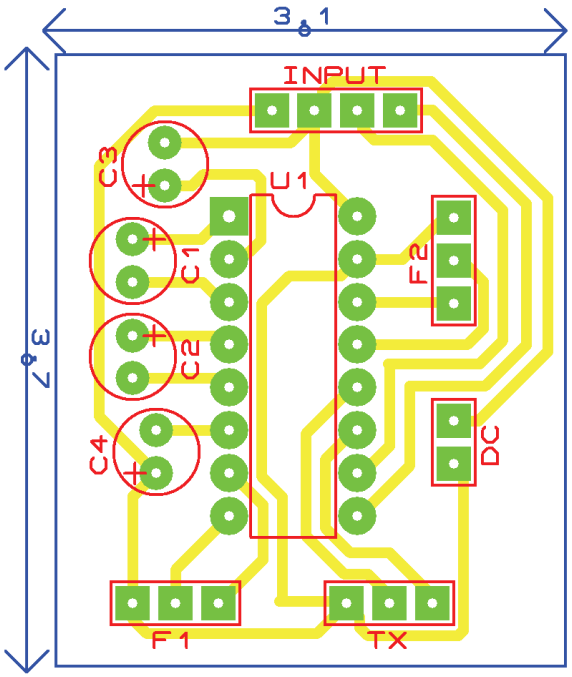
- **DIAGRAMA DEL CIRCUITO PARA EL CONTROL DE ILUMINACIÓN**



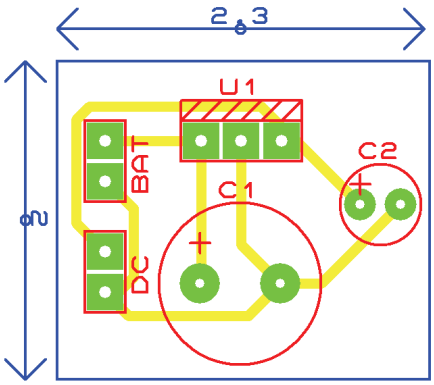
- **DIAGRAMA DEL CIRCUITO PARA EL CONTROL DE PERSIANAS Y ACCESOS**



- **DIAGRAMA DEL CIRCUITO PARA LA INTERFAZ DE COMUNICACIÓN SERIAL**



- **DIAGRAMA DEL CIRCUITO PARA LAS FUENTES**



ANEXO 23: PRESUPUESTO DE LOS MÓDULOS DOMÓTICOS DE LA EMPRESA LUTRON

Bill of Materials - Tesis

Bill of Materials

	Model	Description	Unit Price	Qty	Total Cost
	RRD-6D-WH	Maestro 600W 2-Wire Dimmer	$149,00	8	$1.192,00
	CW-1-WH	Claro Wallplate 1 Gang	$4,90	8	$39,20
	QSYRx-Sxxx	Roller Shade. Please use Shade Configuration Tool (SCT) to order all shades/blinds.	Contact Lutron	3	Contact Lutron
	RR-MAIN-REP-WH	Main Repeater	$499,00	1	$499,00
			Total Price (USD)		**$1.730,20***

***Note: Total pricing is incomplete.**