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

IMPLEMENTACIÓN DE UN SISTEMA DOMÓTICO, MONITOREADO A TRAVÉS DE UNA APLICACIÓN PARA PLATAFORMAS ANDROID, UTILIZANDO LA PLATAFORMA ARDUINO

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

LUIS ALBERTO SANDOVAL VILLARREAL
luissvmh@hotmail.com

JORGE ARTURO TOBAR SALDAÑA
jats_666@hotmail.com

DIRECTOR: MSc. RICARDO XAVIER LLUGSI CAÑAR
ricardo.llugsi@epn.edu.ec

CODIRECTOR: PhD. LUIS ANIBAL CORRALES PAUCAR
luisco5049@yahoo.com

Quito, Noviembre 2015

DECLARACIÓN

Nosotros, Luis Alberto Sandoval Villarreal y Jorge Arturo Tobar Saldaña, 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.

Luis Alberto Sandoval Villarreal

Jorge Arturo Tobar Saldaña

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por Luis Alberto Sandoval Villarreal y Jorge Arturo Tobar Saldaña, bajo nuestra supervisión.

MSc. Ricardo Llugsi
DIRECTOR DEL PROYECTO

PhD. Luis Corrales
CODIRECTOR DEL PROYECTO

AGRADECIMIENTO

El presente trabajo investigativo fruto del esfuerzo y la perseverancia plasmado en el mismo, queremos dedicar con gran estima y respeto a:

A la Universidad Politécnica Nacional, Facultad de Ingeniería Electrónica y Telecomunicaciones, cuna de nuestro saber.

Al Magister Ricardo Llugsi y Dr. Luis Corrales, Director y Codirector de Tesis; quienes con sus valiosos conocimientos, experiencia y profesionalismo nos guiaron a lo largo de la elaboración del presente trabajo.

A todos y cada uno de los docentes que forjaron nuestro camino académico y de vida a lo largo de nuestra carrera.

A la Ingeniera Jessica Andrade, quien con su ayuda y perseverancia nos impulsó a concluir este proyecto de titulación.

Luis Alberto Sandoval Villarreal.

AGRADECIMIENTO

Primero doy gracias a Dios por haberme permitido llegar al final de la carrera.

A mis padres y a mis tíos que me han entregado todo su amor, su apoyo y su paciencia, a mi hermana me que brinda su afecto y su compañía, a mi sobrino que me ha enseñado el valor de las pequeñas cosas.

A mi esposa Jessica que tiene la dura tarea de aguantarme, sin su apoyo, constancia y sacrificio esta meta no hubiera podido ser alcanzada.

A nuestro tutor el Ing. Ricardo Llugsi MSc., quién nos ha orientado en todo momento en la realización de este proyecto que enmarca el último escalón de nuestra carrera.

A nuestro co-director y amigo Dr. Luis Corrales, quien con su sabiduría y experiencia supo brindarnos su ayuda y consejos para que este proyecto saliera adelante a pesar de las dificultades.

A mi compañero de tesis Luchito, que a pesar de los problemas en el camino siempre estuvo para respaldarme y buscar una salida.

A mis pocos pero buenos amigos muchas gracias de todo corazón.

Jorge Arturo Tobar Saldaña

DEDICATORIA

Este trabajo de investigación está dedicado con especial cariño a mi mujer he hijo, quienes llenaron mi vida de esperanza y amor, se convirtieron en la razón de mi ser, la motivación para superarme cada día más y así poder alcanzar con éxito las metas que me he propuesto para mi vida futura.

También se lo dedico con todo cariño, a mis padres, hermanos por darme la fuerza necesaria e inyectarme de perseverancia para llegar a este feliz término.

También a mí apreciado amigo y compañero de luchas Jorge Arturo Tobar Saldaña, quién me brindo su guía, estímulo y la fuerza necesaria para la culminación de esta etapa.

Luis Alberto Sandoval Villarreal.

DEDICATORIA

Dedico este trabajo:

Al Divino Niño, por darme la fuerza y perseverancia necesaria para no dejarme vencer, por fortalecer mi corazón e iluminar mi mente en los días más sombríos.

A mis dos madres Lucita y Charito, por ser la razón de mi lucha diaria y por darme la vida, creer en mí, porque siempre me apoyaron y por su amor.

A mis dos padres Eduardo y Jorge, por los ejemplos de perseverancia y constancia que les caracterizan, que me han inculcado siempre y por el valor mostrado para salir adelante.

Al amor de mi vida Jessica, por apoyarme en todo momento, brindándome la fuerza necesaria para continuar, por estar a mi lado en todo momento, ser esa persona incondicional en mi vida y ser la artífice de esto.

A mi hermana Maggi, por su inexplicable amor para mi superación personal, por sus consejos y su apoyo permanente.

A mi panita Jean Carlos, por llenar mi vida de alegría, de inocencia, de cariño, por no dejarme abrumar por los problemas y por ser esa personita que con una locura te saca una carcajada.

Jorge Arturo Tobar Saldaña.

CONTENIDO

DECLARACIÓN I

CERTIFICACIÓN II

AGRADECIMIENTO III

DEDICATORIA V

CONTENIDO VII

LISTA DE FIGURAS XIII

LISTA DE TABLAS XVII

RESUMEN XIX

PRESENTACIÓN XX

CAPÍTULO 1 1

MARCO TEÓRICO 1

1.1 DEFINICIÓN Y FUNCIONES DE LA DOMÓTICA 1

1.1.1 Protocolo X-10 2

1.1.2 Protocolo KNX - EIB 4

1.1.3 Protocolo LonWorks 6

1.1.3.1 Conceptos Básicos sobre *LonWorks*[2] 7

1.1.4 TIPO DE ARQUITECTURA 8

1.1.4.1 Arquitectura Centralizada 8

1.1.4.2 Arquitectura Descentralizada 8

1.1.4.3 Arquitectura Distribuida 9

1.1.4.4 Arquitectura Híbrida / Mixta 10

1.1.5 MEDIO DE TRANSMISIÓN 11

1.1.6 VELOCIDAD DE TRANSMISIÓN 11

1.1.7 PROTOCOLO DE COMUNICACIÓN 12

1.1.8 SISTEMA DOMÓTICO [4] 13

1.1.8.1 Dispositivos de un Sistema Domótico 13

1.2 PLATAFORMA ARDUINO 15

1.2.1 CARACTERISTICAS BASICAS DE LA PLATAFORMA ARDUINO 15

1.2.1.1 Hardware de la Tarjeta Arduino 15
1.2.1.2 Software del Arduino 16
1.2.2 VENTAJAS DE LA PLATAFORMA ARDUINO 18
1.2.3 MODELOS DE ARDUINO 18
1.2.3.1 Tarjeta Arduino Uno 18
1.2.3.2 Tarjeta Arduino Mega/2560 19
1.2.3.3 Tarjeta Arduino Bluetooth 21
1.2.3.4 Tarjeta Arduino Pro 21
1.2.3.5 Tarjeta Arduino Ethernet 23
1.2.3.6 Tarjeta Arduino *Lilypad* 24
1.3 SISTEMA OPERATIVO ANDROID 25
1.3.1 HISTORIA DEL SISTEMA OPERATIVO ANDROID 25
1.3.2 EVOLUCIÓN DE LAS DIFERENTES VERSIONES DE ANDROID 26
1.3.2.1 Android 1.1 *BANANA BREAD o PETIT FOUR* 26
1.3.2.2 Android 1.5 *CUPCAKE* 26
1.3.2.3 Android 1.6 *DONUT* 27
1.3.2.4 Android 2.0/2.1 *ECLAIR* 27
1.3.2.5 Android 2.2 *FROYO* 27
1.3.2.6 Android 2.3 *GINGERBREAD* 28
1.3.2.7 Android 3.0 *HONEYCOMB* 28
1.3.2.8 Android 4.0 *ICE CREAM SANDWICH* 28
1.3.2.9 Android 4.1.2 *JELLY BEAN* 28
1.3.2.10 Android 4.4 KitKat 29
1.3.2.11 Android 5.0 Lollipop 30
1.3.3 ARQUITECTURA DEL SISTEMA OPERATIVO ANDROID 31
1.3.3.1 Núcleo Linux 31
1.3.3.2 Runtime de Android 32
1.3.3.3 Librerías nativas 33
1.3.3.4 Entorno de aplicación 33
1.3.3.5 Aplicaciones 34
1.4 ESTÁNDAR IEEE 802.15.4 Y ZIGBEE 34
1.4.1 ESTÁNDAR IEEE 802.15.4 34
1.4.1.1 Arquitectura [7] 36

1.4.1.1.1 Capa física 36
1.4.1.1.2 Capa de enlace de datos.....37
1.4.1.1.3 Control de Acceso al Medio (MAC, Media Control Access) 37
1.4.1.1.4 Control de Enlace Lógico (LLC, Logical Link Control).....38
1.4.2 ZIGBEE 38
1.4.2.1 Características de Zigbee.....39
1.4.2.2 Tipos de Dispositivos en una Red *ZigBee* [11] 39
1.4.2.2.1 Full Function Device (FFD).....39
1.4.2.2.2 Reduced Function Device (RFD).....40
1.4.2.3 Red ZigBee 40
1.4.2.3.1 Topologías de Red [12] 40
1.4.2.4 Modos de Operación en *ZIGBEE* 43
1.4.2.4.1Modo Beacon-habilitado.....43
1.4.2.4.2 Modo Beacon-no habilitado.....44
1.4.2.5 Conexión de dispositivos en una red ZigBee[11].....45
1.4.2.6 Comunicación en una red ZigBee [11] 45
1.4.2.7 Seguridad en ZigBee [23].....47
1.4.2.8 Autenticación.....48
1.5 LA NUBE 49
1.5.1 PRINCIPALES APLICACIONES DE LA NUBE.....49
1.5.1.1 Dropbox 49
1.5.1.2 Google Docs y Calendar. 50
1.5.1.3 Grooveshark.....50
1.5.1.4 Picasa – Flickr.....50
1.5.1.5 Zoho.....50

CAPÍTULO 2.....52

DISEÑO E IMPLEMENTACIÓN DE LA PLATAFORMA ARDUINO EN EL SISTEMA DOMÓTICO.....52

2.1 MÓDULO INALÁMBRICO.....53
2.1.1 DISTRIBUCIÓN DE PINES DEL MÓDULO XBP24-AWI-001 55
2.1.2 MODOS DE OPERACIÓN. 56
2.1.2.1 Modo Transparente.....56

2.1.2.2 Modo API (Interfaz de programación de aplicaciones, *Application Programming Interface*) 56
2.2 SELECCIÓN DE LA TARJETA ARDUINO.....59
2.3 SELECCIÓN DEL MICROCONTROLADOR 63
2.4 CONFIGURACIÓN DE LOS MÓDULOS XBEE PRO [12].....65
2.5 DISEÑO DEL CIRCUITO CENTRAL 71
2.5.1 DESARROLLO DEL PROGRAMA PARA EL NODO CENTRAL.....74
2.6 DISEÑO DEL CIRCUITO PARA EL CONTROL DE TEMPERATURA AMBIENTAL.....77
2.6.1 DESARROLLO DEL PROGRAMA PARA CONTROL DE TEMPERATURA AMBIENTAL.....80
2.7 DISEÑO DEL CIRCUITO DEL CONTROL DE LA CORTINA MOTORIZADA..82
2.7.1 DESARROLLO DEL PROGRAMA PARA EL CONTROL DE LA CORTINA MOTORIZADA. 84
2.8 DISEÑO DEL CIRCUITO PARA CONTROL DE ACCESO. 85
2.8.1 DESARROLLO DEL PROGRAMA PARA EL CONTROL DE ACCESO. 87
2.9 DISEÑO DEL CIRCUITO DIMMER.....89
2.9.1 DESARROLLO DEL PROGRAMA PARA EL CONTROL DEL DIMMER 93
2.10 DISEÑO DEL CIRCUITO DE LA CÁMARA 94

CAPÍTULO 3.....99

DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN ANDROID 99

3.1 ALMACENAMIENTO EN LA NUBE.....99
3.1.1 SERVIDOR WEB.....100
3.1.1.1 XAMPP (Cross-Plataform Apache MySQL PHP Perl) 101
3.1.1.2 PHP (*Personal Home Page, Hypertext Preprocessor*).....103
3.2 APP INVENTOR 109
3.2.1 INTRODUCCIÓN.....109
3.2.2 DEFINICIÓN.....110
3.2.3 USO DEL DISPOSITIVO ANDROID CON APP INVENTOR.....113
3.3 APLICACIÓN DESARROLLADA 118
3.3.1 PANTALLA “CONTRASEÑA” 118
3.3.2 PANTALLA “PRESENTACION” 119

3.3.3 PANTALLA “MENÚ” 120
3.3.4 PANTALLA “CÁMARA” 121
3.3.5 PANTALLA “SENSORES” 122
3.3.6 PANTALLA “ACTUADORES” 123
CAPÍTULO 4 125
PRUEBAS FUNCIONALES DEL PROTOTIPO Y ESTIMACIÓN DE COSTOS 125
4.1 PRUEBAS EN LOS NODOS 125
4.1.1 NODO DE CONTROL DE TEMPERATURA AMBIENTAL 125
4.1.2 NODO CORTINA MOTORIZADA 128
4.1.3 NODO DE CONTROL DE ACCESO 130
4.1.4 NODO DIMMER. 131
4.1.5 NODO CÁMARA 133
4.1.6 NODO CENTRAL 141
4.2 PRUEBAS DE INTERACTIVIDAD ENTRE USUARIOS Y LA APLICACIÓN DOMÓTICA 141
4.3 PRESUPUESTO REFERENCIAL 143
4.3.1 REFERENCIA DE COSTOS PARA EL NODO CENTRAL 143
4.3.2 REFERENCIA DE COSTOS PARA EL NODO CONTROL DE TEMPERATURA AMBIENTAL 145
4.3.3 REFERNCIA DE COSTOS PARA EL NODO CORTINA MOTORIZADA 146
4.3.4 REFERENCIA DE COSTOS PARA EL NODO CONTROL DE ACCESO 147
4.3.5 REFERENCIA DE COSTOS PARA EL NODO DIMMER. 148
4.3.6 REFERENCIA DE COSTOS PARA EL NODO CÁMARA. 149
CAPÍTULO 5 152
CONCLUSIONES Y RECOMENDACIONES 152
5.1 CONCLUSIONES: 152
5.2 RECOMENDACIONES: 154
REFERENCIAS BIBLIOGRÁFICAS 156
ANEXOS 159
ANEXO A. 160

MANUAL PARA MÓDULOS XBEE PRO SERIE 1. 160
ANEXO B. 174
CARACTERÍSTICAS DEL MICROCONTROLADOR 164P. 174
ANEXO C. 180
CÁLCULO DE RESISTENCIA LIMITADORA PARA LED´S 180
ANEXO D 183
CIRCUITO INTEGRADO 7805 183
ANEXO E 185
CIRCUITO INTEGRADO LM317 185
ANEXO F. 188
MÓDULO BK008 DE 2 CANALES. 188
ANEXO G. 190
PROFORMAS. 190
ANEXO H. 196
CÓDIGO FUENTE. 196

LISTA DE FIGURAS

CAPÍTULO 1

Figura 1.1 Señal X –10 y su forma real cuando ha sido insertada en la señal. [2] .4
Figura 1.2 Esquema de arquitectura de un sistema domótico centralizado. [4].....8
Figura 1.3 Esquema de arquitectura de un sistema domótico descentralizado. [4] 9
Figura 1.4 Esquema de arquitectura de un sistema domótico distribuido.[4] 9
Figura 1.5 Esquema de arquitectura de un sistema domótico híbrido/mixto.[5] ...10
Figura 1.6 Ejemplos de dispositivos de un sistema domótico. [4] 14
Figura 1.7 Partes principales de la Tarjeta Arduino. [6] 16
Figura 1.8 Software de la Plataforma Arduino. [8] 17
Figura 1.9 Arduino Uno. [7] 19
Figura 1.10 Arduino Mega/2560. [7] 20
Figura 1.11 Tarjetas Arduino Bluetooth HC-05 y HC-06. [8] 21
Figura 1.12 Arduino Pro Mini. [7].....22
Figura 1.13 Arduino Pro Estándar. [7] 22
Figura 1.14 Arduino Ethernet. [7].....23
Figura 1.15 Arduino Lilypad. [7].....24
Figura 1.16 Ejemplos de las versiones Android. [9].....26
Figura 1.17 Arquitectura Android. [10].....31
Figura 1.18 Relación entre IEEE 802 y el modelo de referencia ISO/OSI [14].....36
Figura 1.19 Red tipo Estrella. [16] 41
Figura 1.20 Red tipo Malla. [16].....42
Figura 1.21 Red tipo Cluster Tree. [16].....42
Figura 1.22 Partes del periodo activo de una supertrama. [16].....43
Figura 1.23 Transferencia de datos en modo Beacon-Habilitado. [16] 44
Figura 1.24 Transferencia de datos en modo Beacon- no Habilitado. [16].....44
Figura 1.25 Capas del Estándar Zigbee y 802.15.4. [13].....45

CAPÍTULO 2

Figura 2.1 Esquema global del sistema domótico. 52
Figura 2.2 Módulo XBP24-AWI-001. [1].....53
Figura 2.3 Estructura de la trama API. [2].....57
Figura 2.4 Principales pines de la tarjeta Arduino Mega. 63
Figura 2.5 Distribución de pines del Microcontrolador Atmega 164p. [6] 64
Figura 2.6 Pantalla del software de configuración X-CTU, opción Range Test. ...66
Figura 2.7 Opción para ensamblar paquetes. 67
Figura 2.8 Configuración del PC settings del módulo XBee pro serie uno. 68
Figura 2.9 Modo de configuración del módulo XBee. 69
Figura 2.10 Diagrama de bloques del nodo central. 71
Figura 2.11 Diagrama de conexión entre el módulo XBee Pro y el Arduino Mega2560. 72
Figura 2.12 Diagrama de conexión del nodo central.....73
Figura 2.13 Impreso de la placa del nodo central. 74
Figura 2.14 Diagrama de flujo del nodo central. 75
Figura 2.15 Diagrama de bloques del control de temperatura ambiental. 77
Figura 2.16 Diagrama de conexión del sensor LM35 y el amplificador LM358. ...78
Figura 2.17 Diagrama de conexión del control de temperatura ambiental. 79
Figura 2.18 Impreso del control de temperatura ambiental. 80
Figura 2.19 Diagrama de flujo del control de temperatura ambiental. 81
Figura 2.20 Diagrama de bloques del nodo cortina. 82
Figura 2.21 Diagrama de conexión del nodo cortina.....83
Figura 2.22 Impreso del circuito del nodo cortina. 83
Figura 2.23 Diagrama de flujo del control de la cortina motorizada. 84
Figura 2.24 Diagrama de bloques del nodo control de acceso. 85
Figura 2.25 Diagrama de conexión del nodo control de acceso. 86
Figura 2.26 Impreso del circuito del nodo control de acceso. 87
Figura 2.27 Diagrama de flujo del control de acceso. 88
Figura 2.28 Diagrama de bloques del nodo dimmer. 89
Figura 2.29 Circuito detector de cruce por cero. 90
Figura 2.30 Circuito de potencia. 91
Figura 2.31 Impreso del circuito del nodo dimmer. 92
Figura 2.32 Diagrama de flujo del control del dimmer.....93

Figura 2.33 Diagrama de bloques del nodo cámara. 94
Figura 2.34 Cámara Jpeg Serial Ttl Ls-Y201 [20].....94
Figura 2.35 Impreso del circuito del nodo cámara. 98

CAPÍTULO 3

Figura 3.1 Esquema de conexión de un servidor Web [22].....101
Figura 3.2 Panel de control del Servidor Xampp. 102
Figura 3.3 Arranque de los servicios Apache Y MySQL. 103
Figura 3.4 Fichero del menú del sistema domótico.....105
Figura 3.5 Fichero que permite el control de los diferentes nodos del sistema domótico. 106
Figura 3.6 Fichero que permite censar los diferentes nodos del sistema domótico. 106
Figura 3.7 Fichero que permite obtener la imagen desde el nodo de la cámara. 107
Figura 3.8 Opciones del Servidor Xampp. 107
Figura 3.9 Tabla de Actuadores. 108
Figura 3.10 Tabla de Sensores. 109
Figura 3.11 Gestor de Proyectos. 111
Figura 3.12 Diseño de la aplicación a construir. 111
Figura 3.13 Editor de Bloques. 112
Figura 3.14 Aplicativo Mit Ai2 Companion disponible en el App Store. 113
Figura 3.15 Selección de Al Companion. 114
Figura 3.16 Escaneo del Código QR. 115
Figura 3.17 Ordenador y Android en la misma red Wifi. 115
Figura 3.18 Selección App (Provide QR Code for .Apk). 116
Figura 3.19 Activación de fuentes desconocidas. 117
Figura 3.20 Icono de acceso a la aplicación domótica creada. 117
Figura 3.21 Pantalla contraseña vista en Designer. 118
Figura 3.22 Pantalla contraseña vista en Blocks. 119
Figura 3.23 Pantalla presentación vista en Designer. 119
Figura 3.24 Pantalla presentación vista en Blocks. 120

Figura 3.25 Pantalla menú vista en Designer. 120
Figura 3.26 Pantalla menú vista en Blocks. 120
Figura 3.27 Pantalla cámara vista en Designer. 121
Figura 3.28 Pantalla cámara vista en Blocks. 121
Figura 3.29 Pantalla sensores vista en Designer. 122
Figura 3.30 Pantalla sensores vista en Blocks. 122
Figura 3.31 Pantalla actuadores vista en Designer. 123
Figura 3.32 Pantalla sensores vista en Blocks. 123
Figura 3.33 Diagrama del sistema domótico implementado. 124

CAPÍTULO 4

Figura 4.1 Obtención de la temperatura ambiental desde la aplicación Android. 127
Figura 4.2 Placa del nodo controlador de temperatura ambiental. 127
Figura 4.3 Nodo controlador de temperatura ambiental. 128
Figura 4.4 Placa del nodo cortina motorizada. 129
Figura 4.5 Fines de carrera y motor dc del nodo cortina motorizada. 129
Figura 4.6 Placa del nodo del control de acceso. 130
Figura 4.7 Nodo control de acceso. 131
Figura 4.8 Placa final del nodo dimmer. 132
Figura 4.9 Implementación final nodo dimmer. 132
Figura 4.10 Perdida de paquetes por la imagen transmitida. 134
Figura 4.11 Captura de bytes de la imagen transmitida. 134
Figura 4.12 Paquete de datos obtenido a través del protocolo udp. 135
Figura 4.13 Placa final del nodo cámara. 136
Figura 4.14 Nodo cámara. 136
Figura 4.15 Imagen capturada de la cámara. 138
Figura 4.16 Cámara Ip. 140
Figura 4.17 Imagen obtenida con la cámara Ip. 140
Figura 4.18 Placa del nodo central. 141
Figura 4.19 Menú del sistema domótico. 143

LISTA DE TABLAS

CAPÍTULO 1

TABLA 1.1 Funciones del protocolo tecnología X-10.....3
TABLA 1.2 Propiedades de IEEE.802.4.15 35

CAPÍTULO 2

TABLA 2.1 Descripción de los pines del módulo XBP24-AWI-001 55
TABLA 2.2 Identificador API.....57
TABLA 2.3 Distribución de pines del arduino mega.....60
TABLA 2.4 Parámetros de configuración módulos XBee red punto - multipunto. 70
TABLA 2.5 Parámetros de configuración módulos XBee red punto - punto.....70
TABLA 2.6 Pines de la cámara JPEG SERIAL TTL LS-Y201.....95

CAPÍTULO 4

Tabla 4.1 Pruebas del nodo control de temperatura ambiental. 126
Tabla 4.2 Primera prueba de retardo del sistema domótico. 137
Tabla 4.3 Segunda prueba de retardo del sistema domótico. 137
Tabla 4.4 Tercera prueba de retardo del sistema domótico. 137
Tabla 4.5 Primera prueba de retardo del sistema domótico sin el nodo cámara 139
Tabla 4.6 Segunda prueba de retardo del sistema domótico sin el nodo cámara. 139
Tabla 4.7 Tercera prueba de retardo del sistema domótico sin el nodo cámara.139
Tabla 4.8 Promedio de información obtenida del primer grupo. 142
Tabla 4.9 Promedio de información obtenida del segundo grupo.....142
TABLA 4.10 Promedio de información obtenida del tercer grupo. 142
TABLA 4.11 Costos referentes al nodo central.....144
TABLA 4.12 Costos referentes al nodo termómetro ambiental. 145
TABLA 4.13 Costos referentes al nodo cortina motorizada. 146
TABLA 4.14 Costos referentes al nodo control de acceso.....147

TABLA 4.15 Costos referentes al nodo dimmer. 148
TABLA 4.16 Costos referentes al nodo cámara 150
TABLA 4.17 Costos referentes al sistema domótico. 150

RESUMEN

El presente proyecto tiene como objetivo el diseño y construcción de un prototipo domótico monitoreado de forma remota mediante una aplicación desarrollada para Android a través de Internet. Con este sistema se busca gestionar los medios necesarios para manejar de manera eficiente los recursos energéticos y económicos de los dispositivos eléctricos y electrónicos, utilizando el protocolo Zigbee basado en el estándar IEEE 802.15.4.

Para cumplir con este objetivo se decidió implementar la plataforma Arduino en un sistema formado por un nodo central y cinco nodos secundarios comunicados inalámbricamente. El nodo central utiliza un módulo Arduino Mega 2560, el cual coordina la red, iniciando y manteniendo activa la misma, generando un acceso remoto vía Internet, mientras que los nodos secundarios se encargarán de recibir y procesar la información enviada por el nodo central, ejecutando así las peticiones hechas por el usuario. Los nodos secundarios controlarán una cortina motorizada, un Dimmer, un control de temperatura ambiental, una cerradura eléctrica y una cámara IP. Los dispositivos que permiten establecer la comunicación inalámbrica en este sistema son los módulos XBee Pro Serie Uno, mientras que la interacción con el Internet se la llevará a cabo a través de una aplicación de Android.

Para el manejo de la información recolectada por los nodos secundarios fue necesario la implementación de un servidor Web y una base de datos manejando los conceptos de almacenamiento en la nube, que faciliten la interacción entre la aplicación Android y el sistema domótico.

Las pruebas efectuadas al prototipo permiten afirmar que el sistema completo se puede implementar en cualquier vivienda con el único requerimiento de que los usuarios del sistema dispongan de un dispositivo Android donde se instalará la aplicación. El tiempo de ejecución de una solicitud bordea los 6 segundos, logrando de esta manera cumplir con los objetivos propuestos

PRESENTACIÓN

El prototipo inalámbrico empleado para el monitoreo de la seguridad residencial con acceso y manipulación remota, puede ser empleado en diversas aplicaciones tales como alarmas, control de iluminación, seguridad, entre otros.

El sistema aquí presentado trabaja mediante la plataforma Arduino con una comunicación inalámbrica, utilizando los dispositivos XBee Pro Serie Uno, los cuales hacen posible la transferencia de datos, además se suma el dispositivo electrónico funcionando con el sistema operativo Android con la finalidad de obtener un sistema útil, compacto, versátil y con tecnología novedosa para la domótica

En el primer capítulo de este proyecto se presenta una breve descripción del concepto de domótica, enfocados en el diseño de la misma y sus beneficios, se analiza el sistema domótico que se usó como base para la implementación del presente proyecto, se describe conceptos básicos de funcionamiento, ventajas y desventajas de la plataforma Arduino, además se presenta un estudio general del funcionamiento y principales características del sistema operativo Android para la domótica.

En el segundo capítulo se describe el desarrollo del diseño y construcción del sistema domótico anteriormente diseñado, el cual se mejorará con la implementación de la plataforma Arduino y con la adición de un nodo esclavo, el mismo que permite el control de una cámara IP.

En el tercer capítulo se presenta el desarrollo del servidor Web, una base de datos básica y los conceptos de almacenamiento en la nube (toda esta información como un extra para la optimización de los recursos en el sistema domótico), se describe también los conceptos básicos del software App Inventor y el desarrollo de la aplicación Android que controlará remotamente el sistema.

En el cuarto capítulo se realiza un presupuesto de los elementos que se requirieron para la implementación del sistema y se exponen las pruebas realizadas al prototipo domótico, los resultados de funcionamiento de cada nodo, los tiempos de respuesta en diferentes situaciones y una solución óptima para el problema de la transmisión de video por ZigBee.

En el quinto capítulo se expresan las conclusiones y recomendaciones obtenidas a la culminación de este proyecto. Adicionalmente se incluyen anexos importantes del proyecto.

CAPÍTULO 1

MARCO TEÓRICO

En la actualidad la sociedad ecuatoriana busca contar con sistemas de seguridad efectivos que optimicen el gasto energético y a la vez generen confort dentro del hogar; y que además sean útiles para personas con discapacidad.

Buscando desarrollar, con tecnología propia, equipos con las propiedades comentadas anteriormente se ha optado por elaborar un prototipo domótico de uso residencial que se ajuste a las necesidades mencionadas; el sistema estará basado en la plataforma Arduino, y empleará el estándar IEEE 802.15.4 (tecnología ZigBee) para lograr el monitoreo de un sistema domótico prototipo a través de una aplicación desarrollada para el sistema operativo Android.

1.1 DEFINICIÓN Y FUNCIONES DE LA DOMÓTICA

La domótica es el conjunto de tecnologías aplicadas al control y la automatización inteligente de una vivienda, permitiendo una comunicación entre el usuario y el sistema a través de redes interiores y exteriores, cuyo control se puede realizar tanto dentro como fuera del hogar.

Huidobro J.M. y Millán R. (autores del Libro Manual de Domótica), concluyen que el origen de la Domótica ocurrió en los años setenta, cuando en Estados Unidos aparecieron los primeros dispositivos de automatización de edificios basados en la tecnología X-10 (Protocolo de comunicaciones para el control de dispositivos eléctricos).

La disponibilidad y proliferación de la electrónica de bajo coste favoreció la expansión de este tipo de sistemas. A partir del año 2000 se ve un incremento de

pequeñas empresas especializadas en el estándar KNX (siglas derivadas de Konnex, estándar mundial cuyo ámbito de actuación es tanto en viviendas como en edificios de grandes dimensiones) y el estándar LONWorks despertando así el interés de la comunidad internacional por la búsqueda de la casa ideal. En el futuro, lo que parece más probable, es que el protocolo KNX prevalezca orientando sus aplicaciones a diferentes mercados de la domótica según la EIBA (*European Installation Bus Association) y* EHSA (*European Home System Association*).

La domótica persigue implementar dentro una vivienda, sistemas de controladores y actuadores con el fin de proporcionar una mejor calidad de vida aumentando el bienestar y seguridad de sus habitantes; además de aportar confort y economizar recursos energéticos.

La domótica busca formar agrupaciones de equipos asociados por ejemplo por sensores, actuadores, que dispongan de la capacidad de comunicarse entre ellos a órdenes de un nodo central, generando facilidad de uso. Los criterios básicos que usa la domótica para obtener todos sus beneficios son:

- El tipo de arquitectura utilizada para su implementación,
- El medio de transmisión,
- La velocidad de transmisión,
- El protocolo de comunicación; y
- La simplicidad que se busque en la implementación.

1.1.1 PROTOCOLO X-10

El protocolo X-10, fue desarrollado para controlar dispositivos eléctricos de forma remota, mediante la transmisión de señales de control en formato digital. El protocolo usa bits de direcciones y bits de órdenes, como se puede ver en la

Tabla 1.1, existen múltiples instrucciones utilizadas por el protocolo entre las cuales se destacan: *ON, OFF, All Lights ON, All off, DIM, Bright,* entre otros.

Tabla 1.1 Funciones del protocolo Tecnología X-10. [2]

Código	Función	Descripción
0 0 0 0	All units off	Apaga todos los dispositivos con el código de casa indicado en el mensaje
0 0 0 1	All lights on	Enciende todas las luces (con la posibilidad de controlar el brillo)
0 1 1 0	All lights off	Apaga todas las luces
0 0 1 0	On	Enciende un aparato
0 0 1 1	Off	Apaga un aparato
0 1 0 0	Dim	Atenúa la intensidad de la luz
0 1 0 1	Bright	Incrementa la intensidad de la luz
0 1 1 1	Extended code	Código de extensión
1 0 0 0	Hail request (solicitud de saludo)	Solicita una respuesta del dispositivo(s) con el código de casa indicado en el mensaje
1 0 0 1	Hail acknowledge	Respuesta al comando anterior
1 0 1 0	Pre-set dim	Permite la selección de dos niveles predefinidos de intensidad de luz
1 1 0 1	Status is on	Respuesta a la Solicitud de Estado indicando que el dispositivo está encendido
1 1 1 0	Status is off	Respuesta indicando que el dispositivo está apagado
1 1 1 1	Status request	Solicitud pidiendo el estado de un dispositivo

Las señales de control X-10, se basan en la transmisión de ráfagas de pulsos., las mismas que se sincronizan con el cruce por cero de la señal eléctrica, representando un 1 lógico con la presencia de un pulso en un semiciclo y la ausencia en el semiciclo siguiente y a la inversa se presenta un 0 lógico, es decir, la tecnología X-10, codifica la información en binario y las transmite sincronizadas con el paso por cero de la corriente alterna. En la Figura 1.1, se observa una señal X – 10 que ha sido embebida en la señal eléctrica.

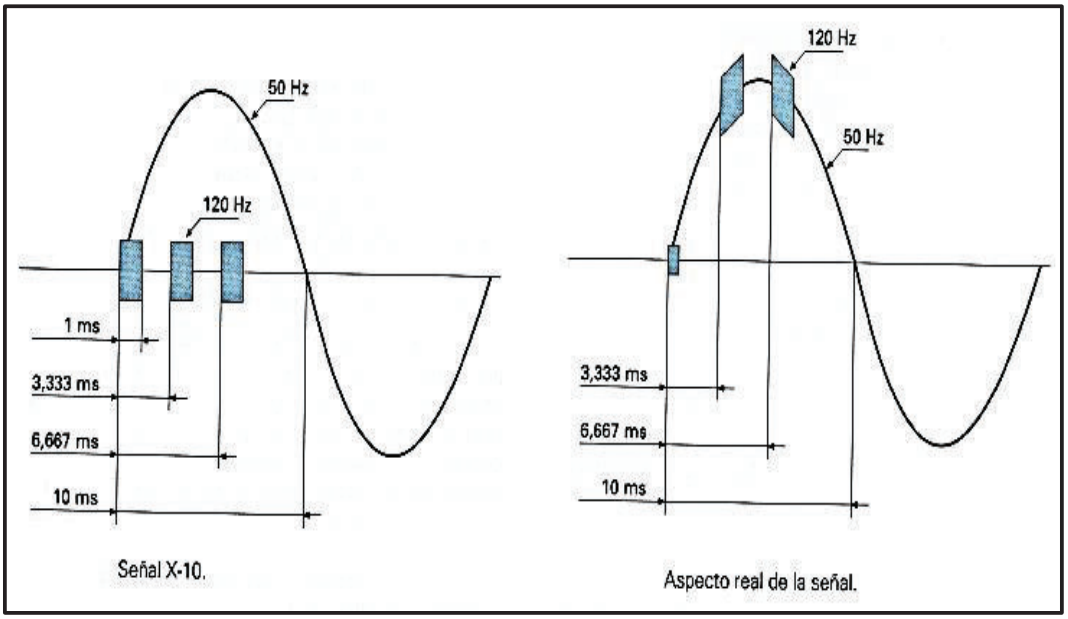


Figura 1.1 Señal X – 10 y su forma real cuando ha sido insertada en la señal. [2]

Actualmente existen empresas que mantienen el protocolo X -10, permitiendo así obtener control de voz, integración de multimedia y un sin número de aplicaciones que permiten que la domótica siga creciendo.

1.1.2 PROTOCOLO KNX - EIB

Konnex (KNX), es el estándar para la automatización de viviendas y oficinas que nació de la idea de tres asociaciones europeas unidas y estas son:

- EIBA, (*European Installation Bus Association*).
- BatiBUS *Club International*.
- EHSA (*European Home System Association*).

La iniciativa KNX nació con los siguientes objetivos:

- Crear un único estándar europeo para la domótica e inmótica que cubra todos los requisitos de las instalaciones profesionales y residenciales.

- Mejorar las prestaciones de los diversos medios físicos de comunicación sobre todo en la tecnología de radiofrecuencia.
- Introducir nuevos modos de funcionamiento que permitan aplicar una filosofía *Plug&Play* a muchos de los dispositivos típicos de una vivienda.
- Contactar con empresas proveedoras de servicios como las empresas de telecomunicaciones, con el fin de potenciar las instalaciones de telegestión técnica de las viviendas.

La versión 1.0 del protocolo KNX contempla tres modos de funcionamiento:

1. *S.mode* (*System mode*): En esta configuración, los diversos dispositivos de la red son instalados y configurados por profesionales con ayuda del software de aplicación diseñado para este propósito. Está especialmente pensado para usarlo en instalaciones de tipo comercial como: oficinas, industrias, hoteles entre otros, donde sólo técnicos especializados tendrán acceso a las herramientas de desarrollo; y los dispositivos *S.mode* sólo podrán ser comprados a través de distribuidores autorizados.

2. *E.mode* (*Easy mode*): En esta configuración "sencilla", los dispositivos son programados en fábrica para realizar una función específica. Sin embargo, algunos detalles deben ser configurados en la instalación seleccionando, en unos microinterruptores, las opciones ofrecidas en una guía de usuario muy simple. Esta configuración la puede llevar a cabo cualquier persona y su compra se la puede realizar en cualquier almacén de productos eléctricos.

3. *A.mode (Automatic mode):* Esta configuración "automática", se basa en la filosofía de *Plug&Play* (conecte y use), en donde, ni el instalador, ni el usuario tienen que configurar el dispositivo. Tan pronto como se conecte el dispositivo a la red, este se registrará en las bases de datos de todos los dispositivos activos. Este modo está

especialmente indicado para ser usado en electrodomésticos, equipos de entretenimiento y proveedores de servicios.

En conclusión, el KNX es un estándar europeo de domótica e inmótica[1] que usa un bus de datos y un cableado propio que necesita ser instalado adecuadamente en el hogar, capaz de competir en calidad, prestaciones y precios con otros sistemas norteamericanos como el *LONWorks* o el X-10.

1.1.3 PROTOCOLO LONWORKS

En el año 1992 la compañía Echelon presentó la tecnología *LonWorks*, como un estándar para la implementación de redes de control distribuidas y de automatización. Es un protocolo creado para cubrir los requisitos de la mayoría de las aplicaciones de control de edificios de oficinas, hoteles, transporte, industrias, monitorización de contadores de energía, iluminación externa, entre otros.

El protocolo *LONworks* se encuentra homologado por las normas Europeas (EN-14908), de Estados Unidos (EIA-709-1) y Chinas (GB/Z20177-2006) así como por el estándar europeo de electrodomésticos (CEDEC AIS). También se ha impuesto dentro de la asociación de petroleros como estándar para el control y comunicación de la red de gasolineras (*International Forecourt Standards Forum,* IFSF) y es ampliamente utilizado en el control de viviendas, edificios, industrial, de transporte ferroviario, naval, aeroespacial y la monitorización remota de contadores.

LONworks ofrece una arquitectura descentralizada, extremo a extremo (*peer to peer*), lo que tiene como ventaja distribuir el procesamiento entre los sensores y

---

[1] Inmótica: Es el conjunto de tecnologías aplicadas al control y la automatización inteligente de edificios no destinados a vivienda, como hoteles, centros comerciales, escuelas, universidades, hospitales y todos los edificios terciarios, permitiendo una gestión eficiente del uso de la energía, además de aportar seguridad, confort, y comunicación entre el usuario y el sistema.

los actuadores instalados, obteniendo así fiabilidad y robustez en los sistemas de control.

1.1.3.1 Conceptos Básicos sobre *LonWorks*[2]

Cualquier dispositivo *LonWorks*, está basado en un microcontrolador llamado *Neuron Chip* fabricado por Toshiba y *Cypress*, del cual se puede destacar:

- Posee un identificador único, conocido como *Neuron ID*, que permite direccionar cualquier nodo de forma exacta dentro de la red *LonWorks.* Este ID, consta de 48 bits y se graba en la memoria *EEPROM* (*Electrically Erasable Programmable Read-Only*) durante la fabricación del circuito.

- Cuenta con un modelo de comunicación totalmente independiente del medio físico sobre el que se implementa, esto es, los datos se pueden transmitir sobre medios guiados y no guiados.

- Estos circuitos se comunican entre sí enviando mensajes que contienen la dirección de destino, información para el *routing*, un *cheksum* para detección de errores, datos de control y de aplicación del usuario.

En la actualidad las decisiones sobre normalización y certificación están bajo el control de *LonMark* Internacional dejando de lado a Echelon que fue el promotor inicial.

LonMark es la asociación conformada por los fabricantes que usan la tecnología *LonWorks*, esta es la encarga de definir los perfiles necesarios para que los equipos sean interoperables entre fabricantes, por ejemplo, se define que la temperatura se expresará en grados centígrados y con dos decimales.

1.1.4 TIPO DE ARQUITECTURA

La Arquitectura de los sistemas de domótica depende de la forma en como se organice la red. Las principales arquitecturas son:

1.1.4.1 Arquitectura Centralizada

En este caso un único controlador centralizado envía la información a los actuadores e interfaces teniendo en cuenta el programa, la configuración y la información que recibe de los sensores, sistemas interconectados y usuarios. En la Figura 1.2, se aprecia el esquema de arquitectura de un sistema domótico centralizado.

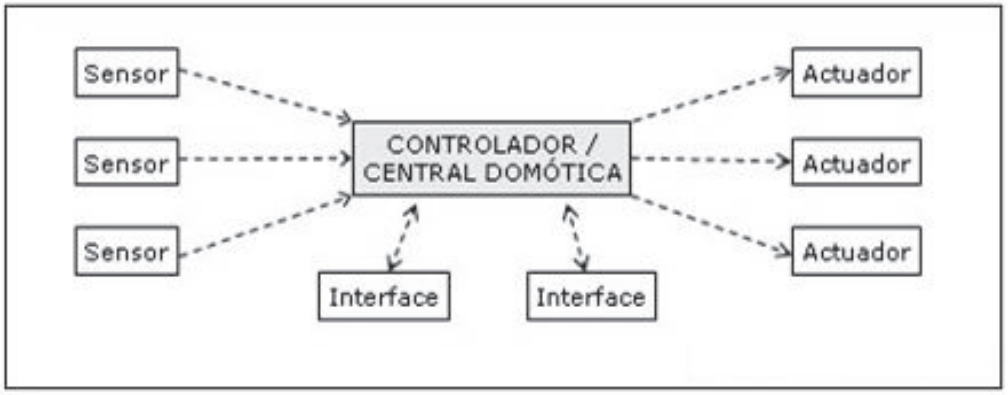


Figura 1.2 Esquema de arquitectura de un sistema domótico centralizado [4]

1.1.4.2 Arquitectura Descentralizada

En este caso, existe más de un controlador interconectados entre sí mediante un sistema de bus que envía información entre ellos y cada uno actúa como un sistema centralizado independiente en el cual cada controlador envía información a los actuadores e interfaces de acuerdo a lo registrado por los sensores o usuarios, ver Figura 1.3.

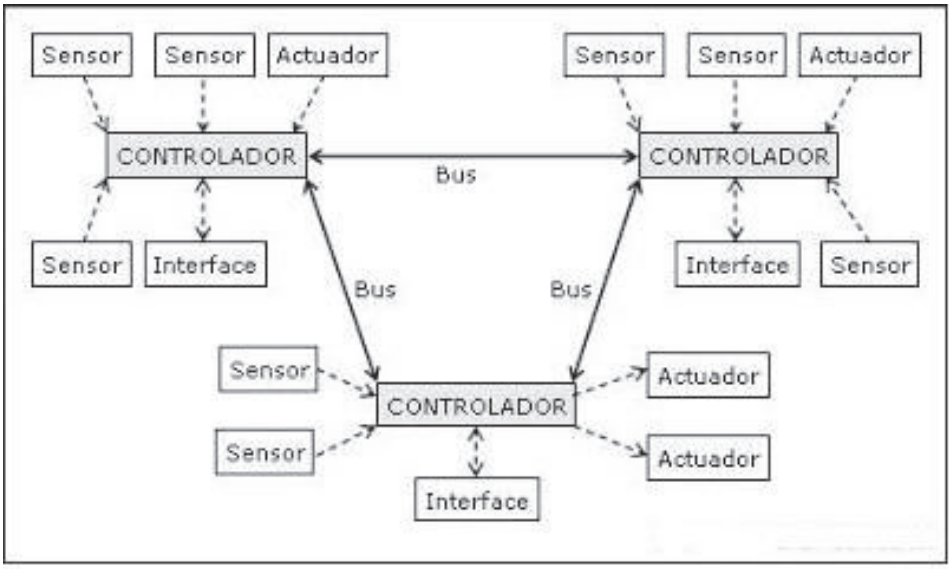


Figura 1.3 Esquema de arquitectura de un sistema domótico descentralizado. [4]

1.1.4.3 Arquitectura Distribuida

En un sistema domótico de arquitectura distribuida, no existe un controlador como tal, sino que cada sensor y actuador forman una unidad capaz de actuar y enviar información según el programa lo requiera, la información que capta por sí mismo y la que recibe de los otros dispositivos. En la Figura 1.4, se muestra el esquema de arquitectura de un sistema domótico distribuido.

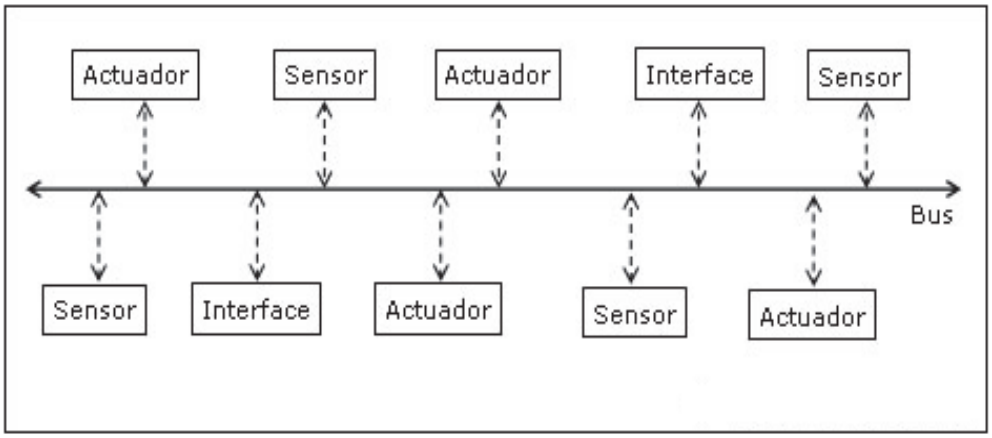


Figura 1.4 Esquema de arquitectura de un sistema domótico distribuido. [4]

1.1.4.4 Arquitectura Híbrida / Mixta

En un sistema de domótica de arquitectura híbrida (también denominado arquitectura mixta) se combinan las arquitecturas de los sistemas centralizadas, descentralizadas y distribuidas. A la vez que puede disponer de un controlador central o varios controladores descentralizados, los dispositivos de interfaces, sensores y actuadores pueden también ser controladores (como en un sistema "distribuido") y procesar la información según el programa, la información que capta por sí mismo y tanto actuarla como enviarla a otros dispositivos de la red, sin que necesariamente pase por otro controlador. En la Figura 1.5, se observa el esquema de arquitectura de un sistema domótico hibrido/mixto.

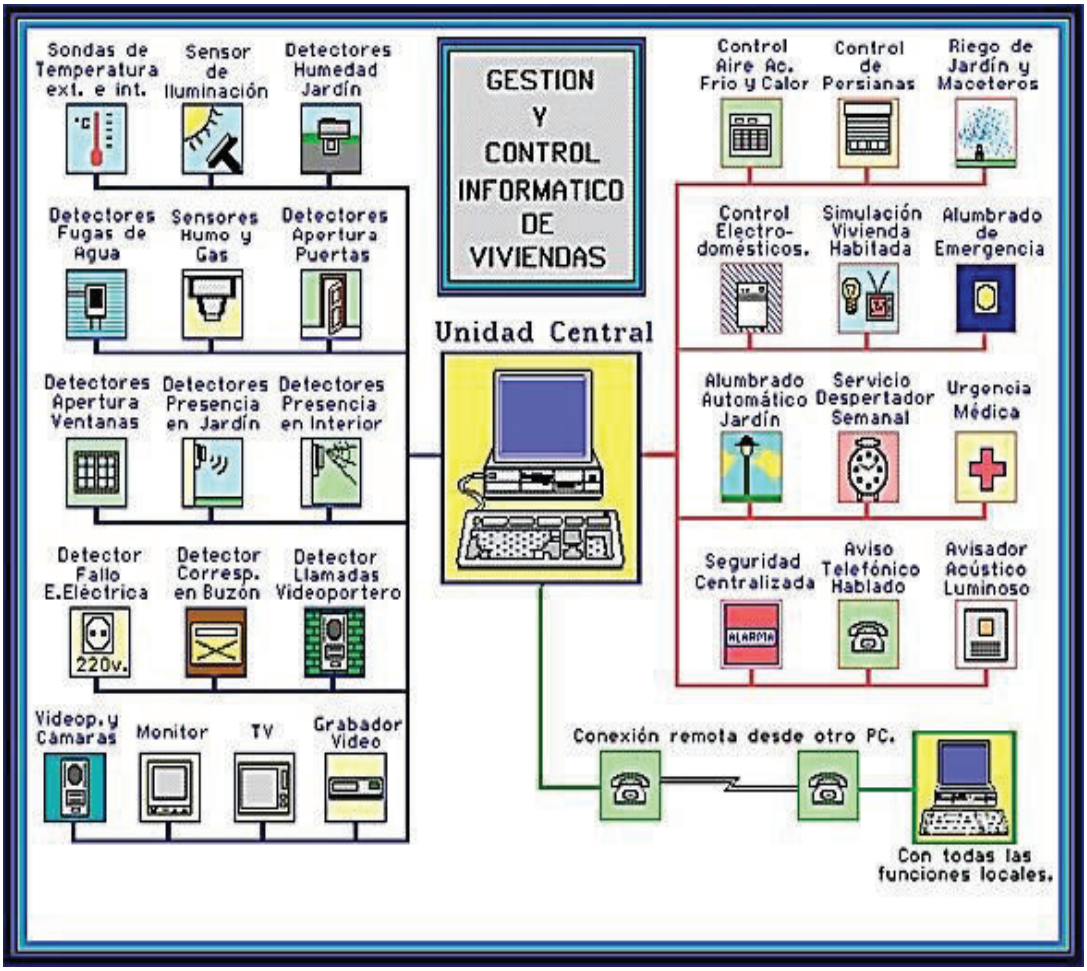


Figura 1.5 Esquema de arquitectura de un sistema domótico híbrido/mixto. [5]

1.1.5 MEDIO DE TRANSMISIÓN

Constituye el medio físico a través del cual el transmisor y receptor se comunican en un sistema de transmisión de datos. Estos medios se clasifican en dos grandes grupos: Medios guiados y no guiados.

- Medios Guiados: Conducen o guían ondas electromagnéticas a través de un camino físico, por ejemplo: cable coaxial, la fibra óptica y el par trenzado

- Medios No Guiados: Transmiten ondas electromagnéticas pero no las dirigen, por ejemplo: el aire y el vacío.

En la Domótica el medio de transmisión usado con mayor frecuencia son los no guiados

1.1.6 VELOCIDAD DE TRANSMISIÓN

La velocidad de transmisión se define como la relación entre la cantidad de información transmitida a través de una red comunicaciones y el tiempo empleado para ello. La unidad que se usa para medir la velocidad es el bit por segundo (bps), aunque es más común acompañarlo con múltiplos por ejemplo: Mbps, megabit por segundo equivalente a un millón de bits/s. A continuación se menciona algunos ejemplos de velocidad de transmisión para medios guiados y no guiados:

- Pares trenzados: Para distancias de pocos kilómetros se pueden obtener transmisiones de varios megabits, por ejemplo el cable UTP cat. 6 trabaja hasta 1Gbps.

- Cable coaxial: para cables de categoría 10Base5 de 1km, es factible obtener velocidades de datos de hasta 10Mbps, y en cables de longitudes menores, es posible obtener velocidades superiores.

- Fibra óptica: para fibra monomodo en la ventana 1550nm se puede alcanzar velocidades de hasta 10 Gbps con un alcance máximo de transmisión de 80 Km con el estándar 100GBase-Er/Ew; para fibras multimodo en la ventana 1330nm se alcanzan velocidades de 100 Mbps con distancia máxima aproximada de 2 km con el estándar 100Base-FX.

- Radio enlaces de VHF y UHF: Su alcance máximo es de 100 km, y las velocidades que permite del orden de los 9600 bps con modulación PSK (*Phase-Shift Keying*, Desplazamiento de Fase).

- Microondas: Los enlaces de microondas permiten grandes velocidades de transmisión, del orden de 10 Mbps, con excelente línea de vista. A través del estándar IEEE802.11b.

1.1.7 PROTOCOLO DE COMUNICACIÓN

La domótica usa tanto protocolos estándar como protocolos propietarios.

- **Protocolos propietarios o cerrados**: Son protocolos específicos de un fabricante determinado y que sólo son usados por dicha marca. Pueden o no ser variantes de Protocolos Estándar. Una de sus desventajas es que sólo el fabricante puede realizar mejoras y usarlo en sus dispositivos para que “hablen el mismo idioma”, lo que limita su evolución. Por ejemplo Control 4 (Basado en ZigBee).

- **Protocolos estándar o Abiertos**: Son protocolos establecidos por organismos internacionales o entre varias empresas con el objetivo de unificar criterios, son protocolos abiertos (*Open Source*), es decir, no existe registro de propiedad de manera que cualquier fabricante puede emplearlo en sus aplicaciones y productos. Los protocolos de mayor acogida en la actualidad son: X-10, KNX y LON.

1.1.8 SISTEMA DOMÓTICO [4]

Es un sistema inteligente que consta de una red de comunicación, configurada de tal manera que admita la interconexión de una serie de equipos que permitan obtener información acerca del entorno de la vivienda, con el fin de ir compilando y procesando dicha información para brindar bienestar, seguridad, gestión energética y climatización dentro de un hogar.

1.1.8.1 Dispositivos de un Sistema Domótico

Los distintos dispositivos que permiten el control de los sistemas de domótica se pueden clasificar en los siguientes grupos:

Sensor: El sensor es el dispositivo encargado de enviar al centro de control su estado según su programación para que el centro de control actúe en consecuencia; es decir, monitoriza el entorno captando información que transmite al sistema (sensores de temperatura, iluminación, etc.). Si el sensor provee una señal eléctrica como respuesta a una variable física entonces toma el nombre de transductor.

Controlador: Los controladores son los encargados de gestionar la información que llega a ellos a través de los sensores. Puede haber un sólo controlador, o varios distribuidos por el sistema.

Actuador: El actuador es un dispositivo que ejecuta una orden del controlador y realiza una acción sobre un aparato o sistema (encendido/apagado, subida/bajada, apertura/cierre, etc.).

Bus: Medio de transmisión que transporta la información entre los distintos dispositivos por un cableado propio, por la redes de otros sistemas (red eléctrica, red telefónica, red de datos) o de forma inalámbrica.

Interfaz: La interfaz hace referencia a los dispositivos (pantallas, móvil) y los formatos (binario, audio o video) en que se muestra la información del sistema para los usuarios (u otros sistemas) y donde los mismos pueden interactuar con el sistema. Forman típicamente una Interfaz Humano Maquina (HMI)

En la Figura 1.6, se observa la clasificación de los diferentes dispositivos de un sistema domótico.

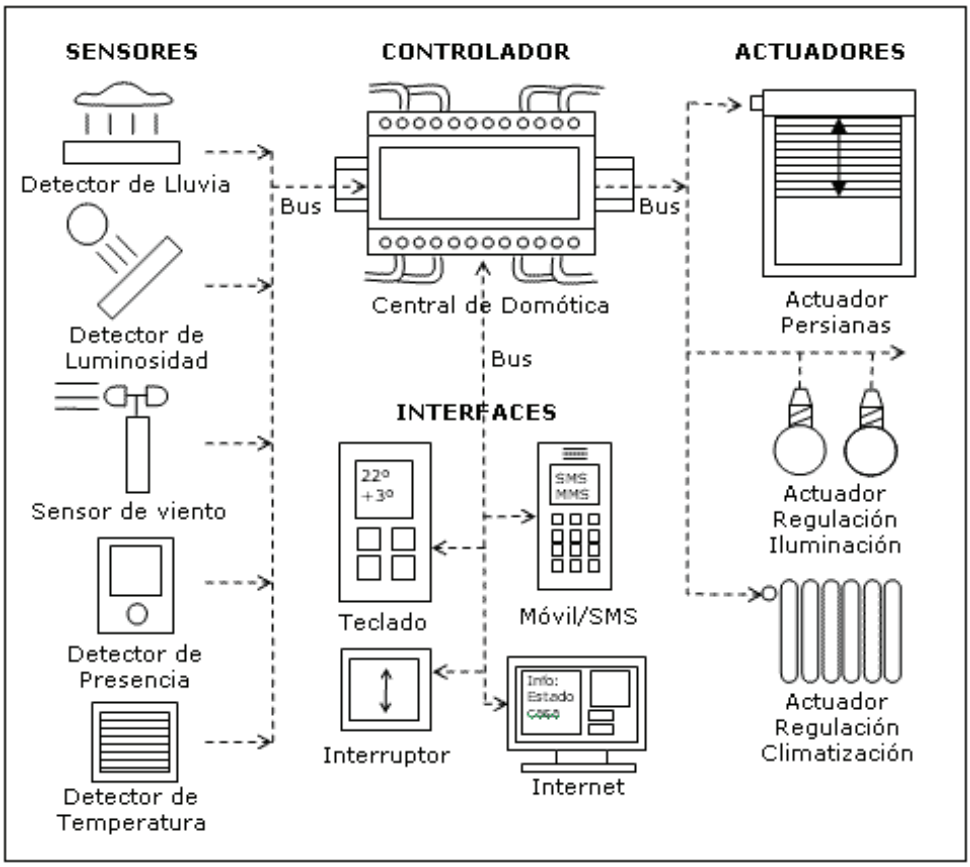


Figura 1.6 Ejemplos de dispositivos de un sistema domótico. [4]

1.2 PLATAFORMA ARDUINO

Es una plataforma que permite conectar el mundo físico con el mundo virtual, o el mundo analógico con el digital.

Sus creadores son David Cuartielles, docente de la Universidad de Mälmo, Suecia y Massimo Banzi, italiano, diseñador y desarrollador Web. El proyecto fue concebido en Italia en el año 2005.

1.2.1 CARACTERISTICAS BASICAS DE LA PLATAFORMA ARDUINO

La tarjeta Arduino es, a la vez, una plataforma hardware (*open - hardware*), una plataforma software (*open - source*) y está soportada por una comunidad de desarrollo.

Es flexible y fácil de utilizar permitiendo controlar el mundo físico a través de un dispositivo electrónico como un ordenador, un Smartphone o una Tablet, entre otros, ya que utiliza una programación simple y directa.

1.2.1.1 Hardware de la Tarjeta Arduino

Es una placa electrónica que se puede adquirir ensamblada o puede ser implementada, ya que los planos electrónicos y la licencia del producto se encuentran en el Internet.

La placa está conformada por un microcontrolador Atmel AVR y puertos de entrada/salida tanto analógica, digital y PWM (*Pulse Width Modulation*, Modulación por ancho de pulsos). Los microcontroladores más usados para implementar esta placa son el Atmega168, Atmega328, Atmega1280, ATmega8. Esto debido a su sencillez y bajo costo lo que permite el desarrollo de múltiples diseños. Tiene una fuente de poder de 3.3 y 5 V (DC).

Las ventajas de las placas ensambladas comercialmente, es que se conectan mediante USB, no necesitan alimentación adicional y se tiene la posibilidad de programar la placa a través del puerto USB, usando *In System Programing*, Programación en el sistema, sin necesidad de utilizar otra placa o dispositivo. En la Figura 1.7, se observa la tarjeta Arduino.

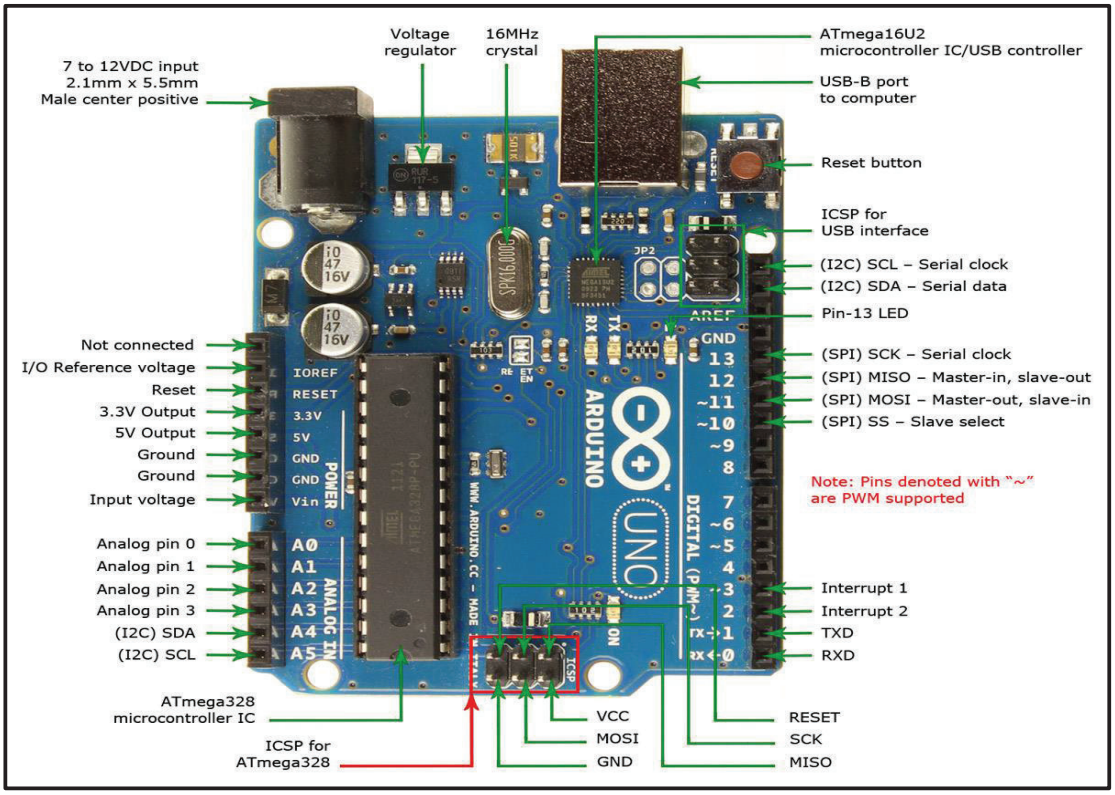


Figura 1.7 Partes principales de la Tarjeta Arduino. [6]

1.2.1.2 Software del Arduino

La plataforma software del Arduino dispone de un gran número de funciones y librerías prefabricadas, que facilitan la programación. El lenguaje de programación que utiliza es *Processing*, que es muy similar a C++, conjuntamente con la biblioteca *Wiring*. Las librerías incluidas en la plataforma (Servomotores, EEPROM, Ethernet, LCD, etc.) permiten controlar, de forma inmediata diversos dispositivos externos o componentes del propio microcontrolador. Sin embargo es

factible usar otros lenguajes de programación y aplicaciones como Java, Matlab, Perl, Visual Basic, etc.

El entorno de desarrollo Arduino está formado por una serie de menús, un editor de texto, una barra de herramientas con botones para funciones comunes y una consola de texto. En la Figura 1.8, se puede apreciar el software de la Plataforma Arduino.

Figura 1.8 Software de la Plataforma Arduino. [8]

1.2.2 VENTAJAS DE LA PLATAFORMA ARDUINO

- La tarjeta Arduino simplifica el trabajo con microcontroladores.
- Las placas Arduino son más accesibles comparadas con otras plataformas.
- Existen prototipos y módulos ya diseñados que facilitan la interconexión con los periféricos.
- La comunicación de la tarjeta Arduino con el computador se puede realizar a través del Puerto Serial (RS-232), Puerto USB o por el sistema ICSP (*In Circuit Serial Program,* Programación Serial En Circuito)[2].

1.2.3 MODELOS DE ARDUINO

Existen varios modelos de tarjetas Arduino, entre los principales se encuentran los siguientes:

1.2.3.1 Tarjeta Arduino Uno

La tarjeta Arduino uno contiene un microcontrolador ATmega328 que tiene un espacio de 32 KB de memoria flash para almacenar el código, de los cuales 0,5 KB son utilizados por el gestor de arranque[3]. También dispone de 2 KB de SRAM

---

[2] ICSP *In-Circuit Serial Programming*, Programación Serial En Circuito: Es una tecnología incluida en todos los microcontroladores de Microchip más recientes y posibilita la reprogramación de los mismos sin que sea necesaria la remoción de éstos de su circuito de aplicación.

[3] Gestor de arranque: Es un pequeño programa que se ejecuta cuando se inicia el microcontrolador, este programa no tiene la totalidad de las funciones de un sistema operativo, pero está diseñado exclusivamente para preparar todo lo que necesita el sistema operativo para funcionar.

(*Static Random Access Memory,* Memoria Estática de Acceso Aleatorio) y 1 KB de EEPROM (*Electrically Erasable Programmable Read Only Memory*, Memoria sólo de Lectura Programable y Borrable Eléctricamente).

Cuenta con 14 entradas y salidas / digitales de los cuales 6 son utilizadas como salidas PWM, además cuenta con 6 entradas analógicas, un cristal de 16 MHZ como oscilador, una conexión USB, un conector de alimentación, una cabecera ICSP (*In-Circuit Serial Programming*, Programación Serial En Circuito) y un botón de reinicio. En la Figura 1.9, se puede apreciar al Arduino Uno.

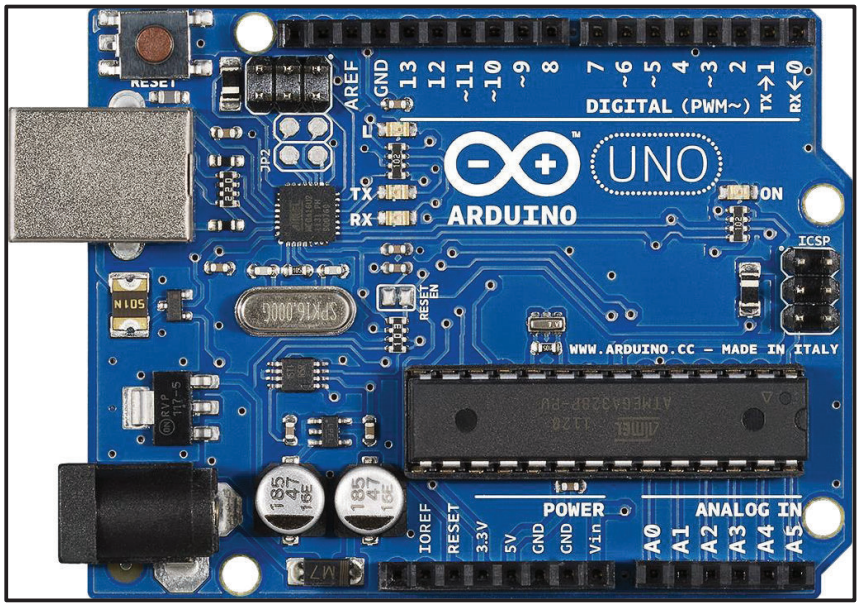


Figura 1.9 Arduino Uno. [7]

1.2.3.2 Tarjeta Arduino Mega/2560

Es con mucha diferencia el más potente y el que más entradas y salidas tiene respecto a los diferentes modelos comerciales de Arduino. Cuenta con un microcontrolador Atmega 2560, con más espacio de memoria para el programa, más RAM y 54 puertos digitales, tiene 256 KB de memoria flash donde guarda el código de los cuales 8 KB se utiliza para el gestor de arranque. Dispone también de:

- 8 KB de SRAM y 4 KB de EEPROM.
- 54 puertos digitales de entrada / salida de los cuales 15 se pueden utilizar como salidas PWM.
- 16 entradas analógicas.
- 4 puertos seriales.
- Un oscilador de 16MHz.
- Una conexión USB.
- Un conector de alimentación.
- Una cabecera ICSP.
- Un botón de reinicio.

En la Figura 1.10, se puede apreciar al Arduino Mega/2560.

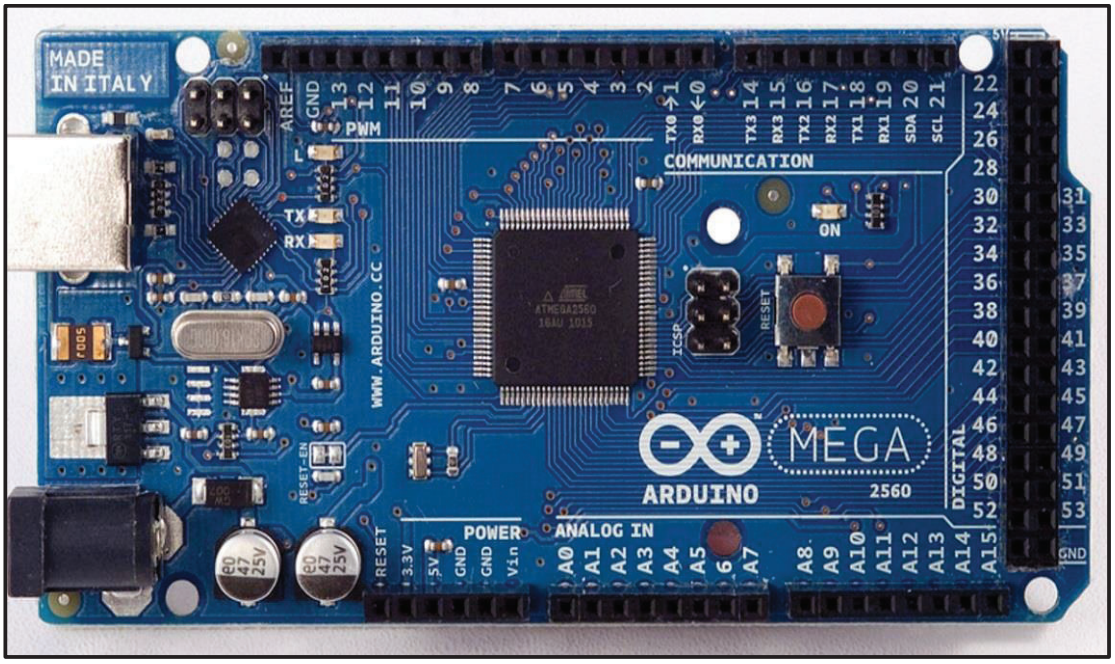


Figura 1.10 Arduino Mega/2560. [7]

Una de las ventajas de este microcontrolador es que puede trabajar conectado a un computador con un cable USB, con su adaptador AC-DC o con una batería.

1.2.3.3 Tarjeta Arduino Bluetooth

Esta versión de Arduino incorpora un módulo para la transmisión de datos de hasta 100 metros, es decir con esta placa se puede programar sin cables así como también realizar comunicaciones serie con cualquier dispositivo *Bluetooth*.

En sus inicios esta tarjeta se implementó con el microcontrolador Atmega168, pero en la actualidad se construye con el Atmega328. Cabe mencionar que no es compatible con dispositivos de audio como por ejemplo audífonos *Bluetooth*.
Dispone de 14 puertos de entradas y salidas digitales, 6 de estos se puede utilizar como salidas PWM, 6 para entradas analógicas, un oscilador de cristal de 16 MHz, terminales para alimentación y un botón de reinicio. En la Figura 1.11, se puede apreciar al Arduino Bluetooth.

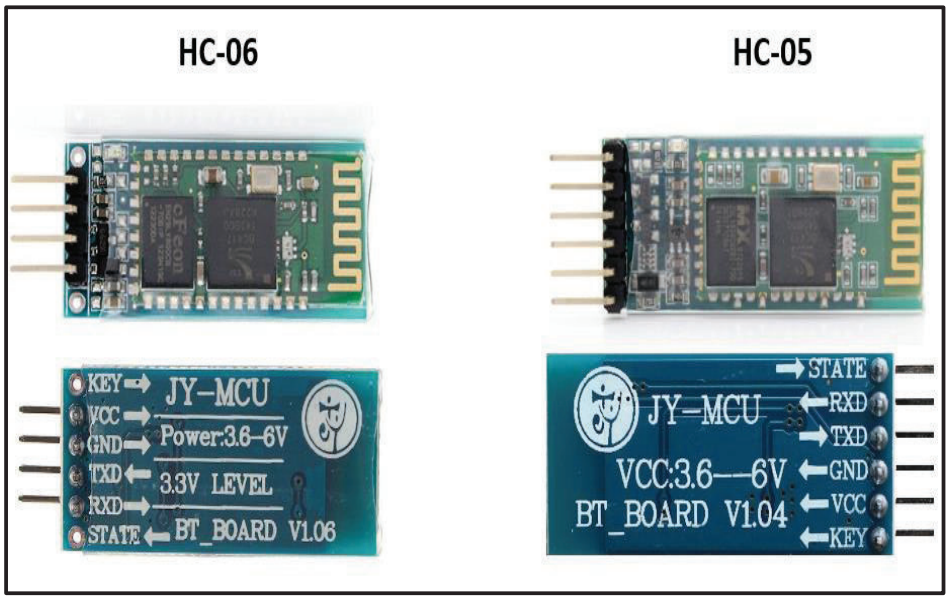


Figura 1.11 Tarjetas Arduino Bluetooth HC-05 y HC-06. [8]

1.2.3.4 Tarjeta Arduino Pro

En este caso se incorpora funcionalidades tales como un conector especial para batería LiPo, ideal para aplicaciones portátiles. Posee un microcontrolador

ATmega168 o Atmega328, con versiones de 3.3 V y 8 Mhz o 5 V para 16 Mhz. Contiene 14 puertos de E/S digitales (6 de ellos son PWM) y 6 pines analógicos. Esta tarjeta cuenta con una versión mini y una versión estándar como se puede observar en la Figura 1.12 y la Figura 1.13 respectivamente.

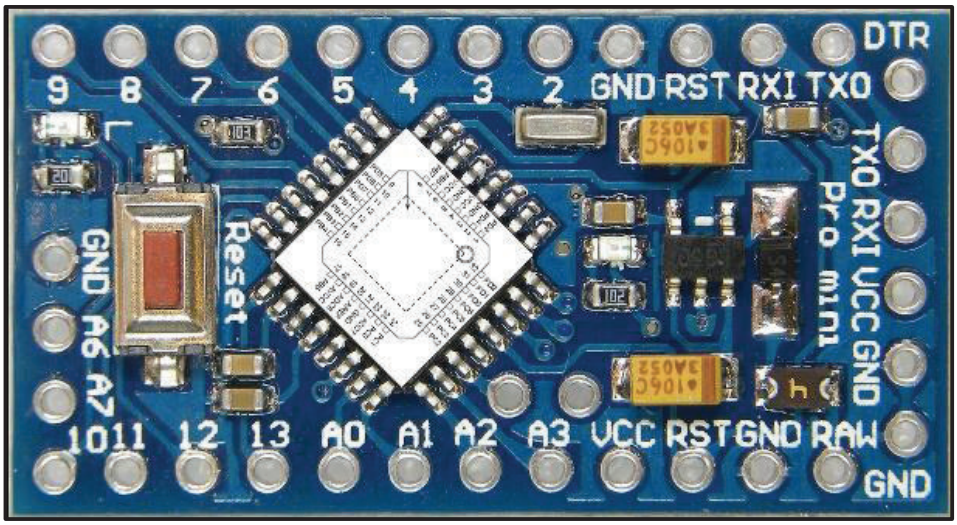


Figura 1.12 Arduino Pro Mini. [7]

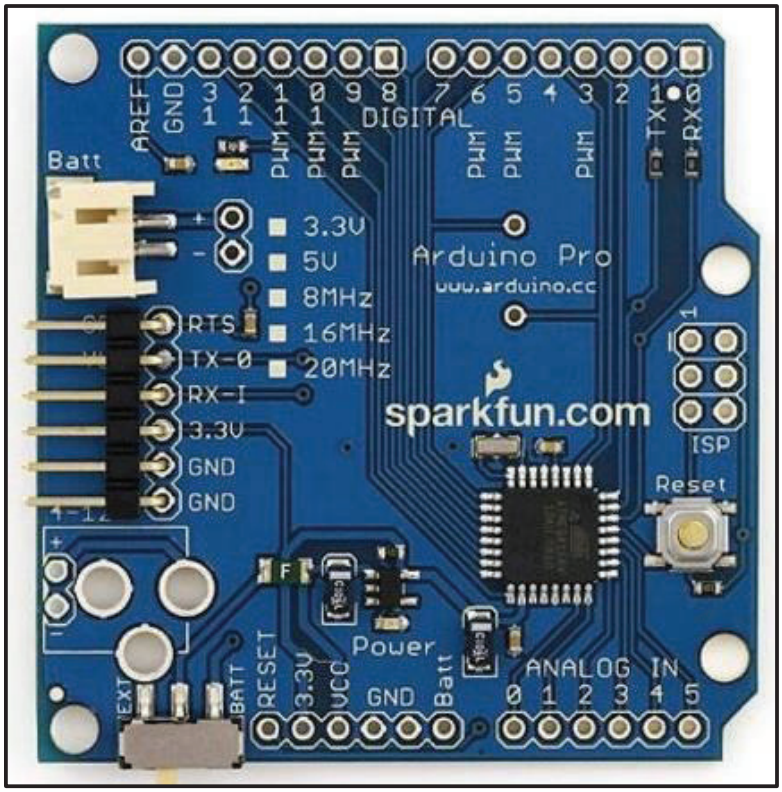


Figura 1.13 Arduino Pro Estándar. [7]

1.2.3.5 Tarjeta Arduino Ethernet

El Arduino Ethernet es bastante similar al Arduino Uno. Su microcontrolador es un ATmega328 que trabaja a 16Mhz (5V) con 2KB de SRAM, 1KB de EEPROM y 32KB de flash, además añade un controlador W5100 TCP/IP embebido que brinda la capacidad para conexión Ethernet y ranuras para tarjetas de memoria microSD. Tiene 14 puertos digitales (4 PWM) y 6 analógicos, hay que tener en cuenta que se reservan los pines 10 al 13 para ser usados para SPI (*Serial Peripheral Interface,* Interfaz Serial Periférica)[4], el 4 para la tarjeta SD y el 2 para interruptor W5100[5]. En la Figura 1.14 se puede apreciar el Arduino Ethernet.

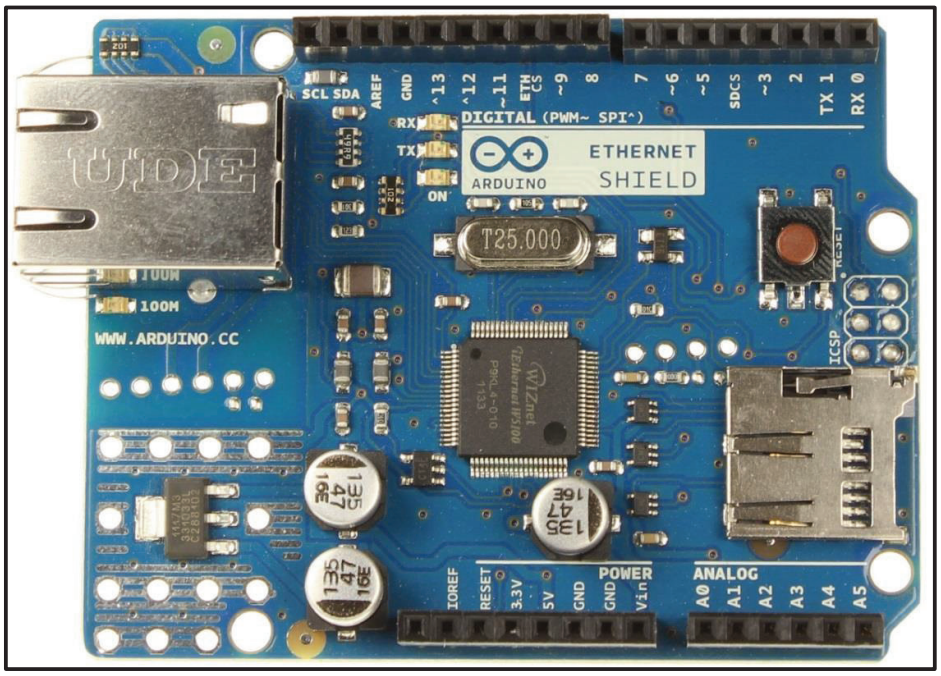


Figura 1.14 Arduino Ethernet. [7]

---

[4] SPI (*Serial Peripheral Interface,* Interfaz Serial Periférica): es un estándar de enlace de datos seriales sincronizados por un reloj que operan en modo full duplex.

[5] El Wiznet W5100 provee de una pila IP de red capaz de comunicarse por protocolos TCP y UDP. Soporta hasta cuatro conexiones por socket simultáneas. Se puede utilizar la librería ethernet para escribir programas que permiten conectarse a Internet.

1.2.3.6 Tarjeta Arduino *Lilypad*

El Arduino *Liypad* fue pensado para ser integrado en prendas y textiles, es decir diseñada para coser con un hilo conductor prendas y accesorios dinámicos e interactivos. Fue desarrollado por *Leah Buechley* y *Spark Fun Electronics* para ser empleado con los mismo fines que otros Arduinos sólo que con ciertas limitaciones a cambio de sus capacidades de integración y su base flexible.

Se basa en dos versiones de microcontrolador diferentes, ambas de bajo consumo denominadas Atmega168V y ATmega328V, esta segunda más potente. Ambos trabajan a 8Mhz, pero la primera trabaja a sólo 2,7V y a segunda a 5,5v. Dispone de 14 pines digitales (6 PWM) y 6 analógicos a lo largo de su perímetro. Además integra 16KB de memoria flash para implementar el código del programa, 1KB de SRAM y 512 bytes de EEPROM. En la Figura 1.15, se puede apreciar el Arduino Lilypad.

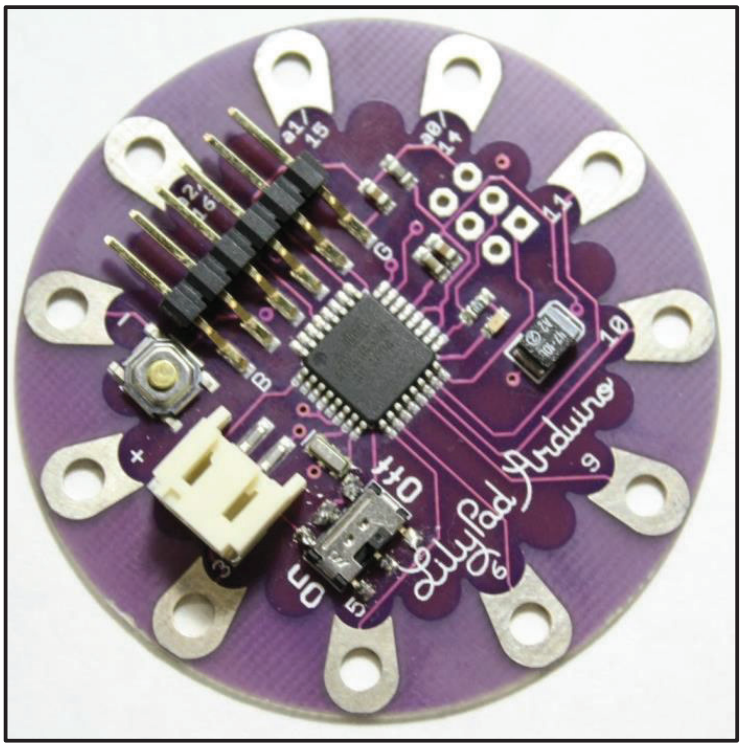


Figura 1.15 Arduino Lilypad. [7]

1.3 SISTEMA OPERATIVO ANDROID

Android es una plataforma de software basado en Linux que fue desarrollado inicialmente para dispositivos móviles y posteriormente se orientó hacia otros equipos como *tablets*, *netbooks*, reproductores de música entre otros

Android fue desarrollado por un grupo de empresas conocido como Open Handset Alliance. Android es un sistema operativo de código libre lo que implica que cualquier persona que conozca de programación en un entorno de trabajo de Java pueda modificar o incluso crear nuevas aplicaciones.

1.3.1 HISTORIA DEL SISTEMA OPERATIVO ANDROID

En el año 2003 se fundó la compañía Android Inc con el objetivo principal de desarrollar software para teléfonos móviles. En el 2005 esta empresa pasó a ser propiedad de Google naciendo el sistema operativo Android y en el 2008 se populariza el sistema operativo Android gracias a la unión de empresas conocido como Alianza *Open Handset*.

El 12 de noviembre del 2007 se presentó el primer producto conocido como *Android Software Developmet Kit* (SDK), una plataforma creada en la versión 2.6 de Kernel[6] de Linux para dispositivos móviles. El 22 de Octubre del 2008 sale al mercado el primer teléfono que contiene el sistema operativo Android de marca T-Mobile modelo G-1, el mismo que tenía un procesador de 528 Mhz, 256MB de memoria ROM y 192MB de memoria RAM.

---

[6] El kernel Linux es el componente central de un sistema operativo GNU/Linux. Este se encarga de manejar los recursos hardware como la CPU, la memoria y los discos duros, proporcionando información que recibe de las aplicaciones.

1.3.2 EVOLUCIÓN DE LAS DIFERENTES VERSIONES DE ANDROID

El sistema operativo Android, desde su creación ha evolucionado de manera exponencial. Por lo cual se hace indispensable detallar las diferentes versiones y cambios que se han presentado con el pasar del tiempo. En la Figura 1.16, se puede apreciar las diferentes versiones de Android.

Figura 1.16 Ejemplos de las versiones Android. [9]

1.3.2.1 Android 1.1 *BANANA BREAD o PETIT FOUR*

La versión 1.1 fue enfocada básicamente a implementar las actualizaciones "*over the air*" y reparar errores de la primera versión, fue lanzada sólo para el modelo de celular "*HTC Dream*". Android 1.1 fue conocido internamente como "Petit Four", aunque este nombre no se utilizó oficialmente.

1.3.2.2 Android 1.5 *CUPCAKE*

Esta versión está basada en el Kernel 2.6.27 de Linux. Lo más relevante de esta versión es la introducción de un teclado virtual de tipo QWERTY, presenta transiciones animadas entre ventanas y una mejora en la velocidad de la cámara, tiene soporte de Bluetooth estéreo y la factibilidad de grabar y reproducir videos. El teléfono celular que utilizó esta versión de Android fue el "*HTC Magic*".

1.3.2.3 Android 1.6 *DONUT*

Esta versión se basó en el Kernel 2.6.29 de Linux, la misma que presentó la posibilidad de conectarse a redes VPN, 802.1x, brinda soporte para redes CDMA y múltiples resoluciones de pantalla, lo que permite hoy en día tener varias resoluciones como HVGA, QHD, WVGA y 720p. En este caso se implementó un nuevo diseño de Android Market con introducción de listas de aplicaciones gratuitas o de pago y una nueva interfaz de la cámara.

1.3.2.4 Android 2.0/2.1 *ECLAIR*

Fueron basadas en el Kernel 2.6.39 de Linux. Presenta mejoras en el teclado virtual y soporte para HTML5 (HyperText Markup Language versión 5, Lenguaje de Marcas de Hipertexto), nuevos tamaños y resoluciones de pantalla, soporte para añadir varias cuentas de correo electrónico, sincronizaciones automáticas, incorpora un sistema de navegación de automóviles en el móvil con vista en 3D (Google Maps), soporte de Microsoft Exchange, presenta la posibilidad de personalizar el celular con fondos animados y nueva pantalla de bloqueo.

1.3.2.5 Android 2.2 *FROYO*

Esta versión se basa en el Kernel 2.6.32 de Linux, en la misma se introduce una galería con imágenes en 3D y la posibilidad de generar una contraseña o PIN (*Personal Identification Number*, Número de Identificación Personal) para el bloqueo del teléfono celular, se mejoró la calidad de video llamadas y se aumentó la capacidad de memoria, se añadió soporte para correo electrónico Microsoft Exchange y para Adobe Flash 10.1.

1.3.2.6 Android 2.3 *GINGERBREAD*

Está basada en el Kernel 2.6.35.7 de Linux. Incorpora el soporte de multitouch, se mejora la gestión de la batería, posibilita la instalación de juegos en formato 3D y permite trabajar con la tecnología NFC (*Near Field Communication*, Comunicación de campo cercano). Se realiza una actualización del diseño de la interfaz de usuario.

1.3.2.7 Android 3.0 *HONEYCOMB*

En este caso se eliminan por completo los botones físicos del dispositivo móvil integrando una serie de botones virtuales. Se presenta un navegador mejorado y una opción de multitarea mejorada que permite tener varias aplicaciones ejecutándose en segundo plano. Es un sistema operativo optimizado para tablets. Tiene un mejor soporte para redes Wi-Fi (*Wireless Fidelity*, Fidelidad Inalámbrica), videochat mediante Google Talk, Flash y DivX (*Digital Video Express*).

1.3.2.8 Android 4.0 *ICE CREAM SANDWICH*

Permite modificar el tipo de letra y a su vez corregir el texto con un diccionario incorporado, mejora el soporte 3D y permite la creación de carpetas con únicamente con arrastrar un icono a otro. Añade un gestor para el uso de datos, permite el desbloqueo del celular mediante reconocimiento facial y reconocimiento de voz.

1.3.2.9 Android 4.1.2 *JELLY BEAN*

Esta versión no presenta un gran cambio en la interfaz del usuario con respecto a Android 4.0 Ice Cream Sandwich, sino que es más bien una corrección de errores de esta última. La mejora que da esta versión es la rotación de la pantalla y

permite desplegar cualquier aplicación con el uso de un solo dedo, con lo cual aumentará automáticamente la velocidad de trabajo del procesador logrando un sistema mucho más fluido.
Se mejora los vacíos existentes en las versiones anteriores, con un mejor rendimiento, velocidad, y un mayor aprovechamiento de hardware a la par con el software.

Los primeros dispositivos que usan esta versión de Android son el Galaxy Nexus, el Nexus S y la tablet Xoom de Motorola

1.3.2.10 Android 4.4 KitKat

Esta versión tiene la capacidad de ajustarse de manera perfecta a teléfonos celulares con memoria RAM de 512 MB, esto es un claro indicativo que esta versión busca llegar a más teléfonos celulares de gama baja. Por lo cual se ha simplificado los procesos para reducir el uso de memoria generando una plataforma operativa más eficiente y rápida, que implementa la interfaz *Google Now*, la cual mediante la voz realizará la accion correspondiente sin la necesidad de un programa que lo ejecute.

Esta versión usa *HangOuts,* que es la aplicación por defecto para mensajería. Es decir, los mensajes de texto SMS (*Short Message Service*) y mensajes multimedia MMS (*Multimedia Messaging System*) se manejarán desde esta aplicación. También se añade una mejora en la pantalla con la tecnología *inmersive mode,* cuando se activa por ejemplo para leer un libro, ver una película o jugar, la interfaz de usuario se oculta automáticamente, y simplemente se debe tocar la orilla de la pantalla para que los iconos vuelvan a aparecer. De esta manera, se consigue una mayor visualización.

Android KitKat ofrece la opción de almacenar y acceder a aplicaciones o archivos directamente en la nube de Internet (servidores *online*), con la cual se logra acceder como si se tratara de un computador.

Los primeros smartphones en usar esta actualización de Android son:

- Sony Xperia serie Z,
- Samsung Serie Galaxy S4, y versiones posteriores,
- HTC serie One,
- LG Optimus L7, L5, L3
- LG Nexus 5 (con el que se presentó la versión KitKat).

1.3.2.11 Android 5.0 Lollipop

Los celulares Nexus 6 y 9 son los primeros dispositivos que cuentan con esta versión del sistema operativo Android, esta versión presenta notificaciones más inteligentes, mejoras en la multitarea, una mejor comunicación entre dispositivos Android entre otras funciones.

Sin duda la principal novedad de esta versión, es su aspecto de pantalla que se adapta a cada dispositivo a modo de una apariencia de capas, haciendo sombras unas sobre otras, con un movimiento visual muy atractivo, dinámico y rápido.

Otra gran novedad son las notificaciones del dispositivo Android y la manera que se puede interactuar con ellas. Entre las notificaciones más relevantes se tiene el sistema de ahorro de energía que permite mostrar los tiempos restantes para cargar o descargar el dispositivo Android de forma clara permitiendo así usar el dispositivo hasta 90 minutos más en promedio.

Aparece un nuevo modo para compartir, llamado 'invitado' el cual permite seleccionar qué información se requiere sincronizar con otro dispositivo Lollipop. Esta es una opción ideal para compartir el teléfono, pero no su privacidad.

También se tiene mejoras en la configuración y activación de WiFi, Bluetooth o GPS. Introduce mejoras en la accesibilidad, en la implementación de hasta 68 idiomas, en el servicio Ok Google o Android TV.

1.3.3 ARQUITECTURA DEL SISTEMA OPERATIVO ANDROID

La Arquitectura de Android está formada por cuatro capas relacionadas entre sí, las mismas que están basadas en software libre. En la Figura 1.17, se observa la arquitectura del sistema operativo Android.

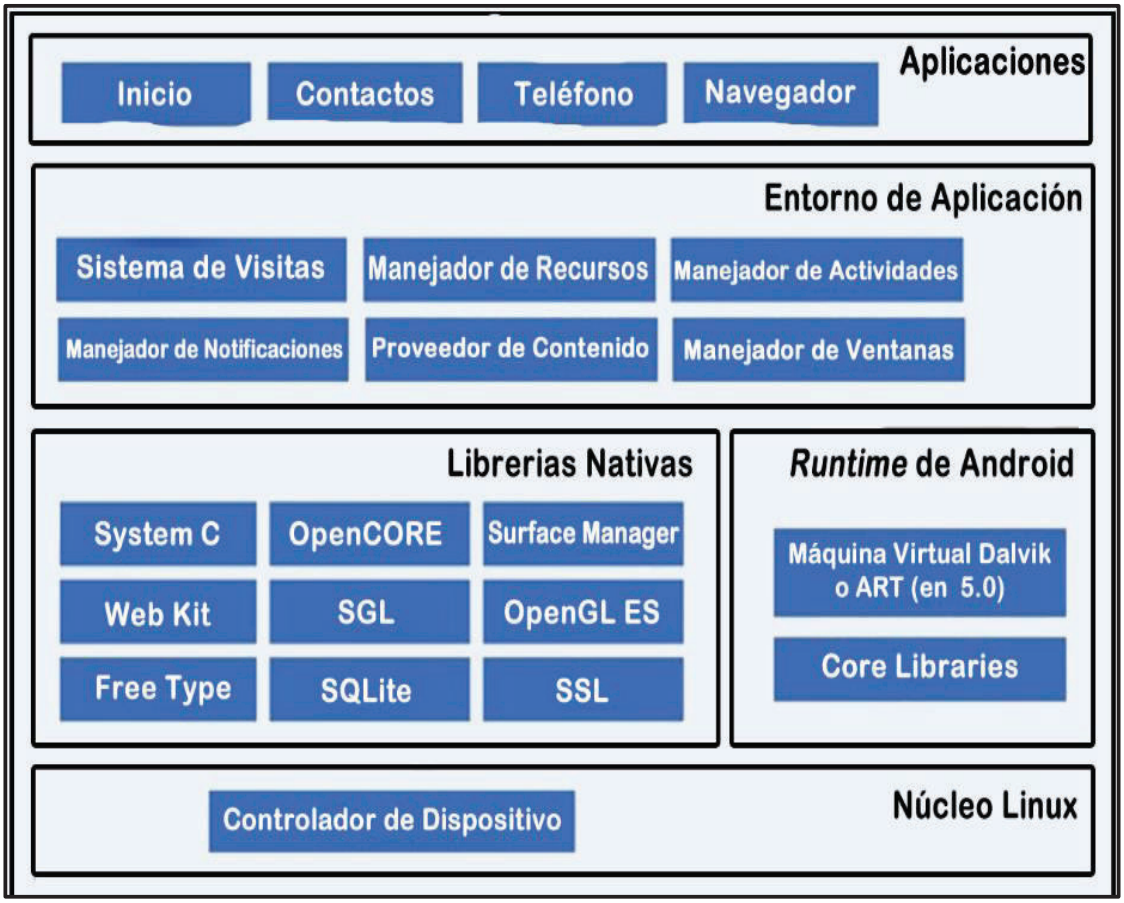


Figura 1.17 Arquitectura Android. [10]

1.3.3.1 Núcleo Linux

Android está formado sobre el *Kernel* de *Linux* versión 2.6, pero modificado drásticamente para adaptarse a dispositivos móviles, con esto se busca aprovechar las características de portabilidad, flexibilidad y seguridad que ofrece *Linux*.

Esta capa contiene los *drivers* necesarios para que cualquier componente físico pueda ser utilizado. Siempre que se desea emplear un nuevo elemento el fabricante debe crear librerías de control o *drivers* dentro del *Kernel* de *Linux* embebido en el propio Android para que pueda ser utilizado. Esta capa del modelo actúa como capa de abstracción entre el *hardware* y el resto de la pila, lo cual la hace única ya que es dependiente.

1.3.3.2 Runtime de Android

Se basa en el concepto de máquina virtual utilizado en Java, sin embargo dada la poca memoria y procesador limitado de los dispositivos donde se hará correr Android, no es posible usar esta máquina virtual. Es por esta razón, que nace la idea de máquina virtual Dalvik[7] de Google que se acopla a las limitaciones mencionadas anteriormente.

Entre las características más importantes de la máquina virtual *Dalvik*, se tiene:

- Ejecuta ficheros .dex (*dalvik executables*, dalvik ejecutables), el cual es un fichero mejorado para optimizar memoria. Estos archivos .dex son comprimidos en Paquestes Android APK (*Android Package*,)
- Se basa en registros en vez de en pilas aprovechando así el mejor rendimiento de los móviles con estos.
- Cada aplicación emplea su propio proceso Linux con su correspondiente instancia de la máquina virtual.
- Encarga al Kernel de Linux algunas funciones threading y el manejo de memoria de bajo nivel.

---

[7] *Dalvik*: Toma su nombre de un pequeño pueblo de Islandia donde vivieron antepasados de su principal creador, Dan Bornstein.

Con la versión de Android 5.0 lollipop, se incluye entre las principales características un nuevo tipo de máquina virtual conocida como ART (Android Runtime), la que trabaja en modo AOD (*Ahead Of Time,* Antes de tiempo), lo que significa que hace una pre-compilación en el momento de instalar una aplicación, por lo que al instante de ser ejecutada la misma por el usuario, se ahorra tiempo y trabajo en tener que interpretar y ejecutar. El uso de ART trae como consecuencia, que la instalación de las aplicaciones tome un poco más de tiempo, pero una vez realizada, se ejecutarán mucho más rápido. Esto se traduce en una mejora importante de la multitarea y en un significativo ahorro de batería.

1.3.3.3 Librerías nativas

Maneja librerías en lenguaje C/C++ que se encuentran compiladas en código nativo (forma de código de la programación que se configura para funcionar con el uso de un procesador especifico) del procesador, algunas de estas librerías son: *System C, Media Framework, Surface Manager, Librerías 3D, SSL y SQLite*, entre otras.

1.3.3.4 Entorno de aplicación

Ayuda a simplificar la reutilización de componentes siguiendo las reglas de seguridad del framework[8]. Los servicios más importantes que incluye son:

- Vistas (*Views*)
- Administración de Actividades y Ubicaciones
- Proveedor de Contenidos

---

[8] *Framework*: es un conjunto de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

- Administrador de Telefonía y Recursos

1.3.3.5 Aplicaciones

Las aplicaciones base incluyen navegador, cliente de correo electrónico, programa de mensajería, calendario, mapas, contactos y juegos, las mismas que están escritas en lenguaje Java o C/C++. Para desarrollar estas aplicaciones se puede usar el Android SDK (*Software Development Kit*, Kit de Desarrollo de Software) o el Android NDK (*Native Development Kit,* Kit de Desarrollo Nativo).

1.4 ESTÁNDAR IEEE 802.15.4 y ZIGBEE

1.4.1 ESTÁNDAR IEEE 802.15.4

Las siglas IEEE corresponden a *The Institute of Electrical and Electronics Engineers*, Instituto de Ingenieros Eléctricos y Electrónicos, una asociación mundial técnico - profesional dedicada a la estandarización y el desarrollo de áreas técnicas.

Según el mismo IEEE, *"su trabajo es promover la creatividad, el desarrollo y la integración, compartir y aplicar los avances en las tecnologías de la información, electrónica y ciencias en general para beneficio de la humanidad y de los mismos profesionales"* [12].

El estándar IEEE 802.15.4 fue diseñado para redes de bajas tasas de transmisión de datos LR-WPAN (*low-rate wireless personal area network*,). Este define las especificaciones de las capas a nivel físico y la de control de acceso al medio en redes inalámbricas, las características más importantes del estándar son:

- Rango de operación de corto alcance.
- Velocidad de transmisión hasta 250 kbps.
- Hardware de bajo costo y bajo consumo de energía.
- Carecen de infraestructura de red, son red del tipo Ad Hoc con capacidad de auto organización y auto gestión.

El estándar 802.15.4 indica que se puede trabajar en la capa física a una frecuencia de 2.4GHz en los canales del 11 al 26, adicional utiliza dispositivos del tipo FFD (*Full Function Device*) que permiten una comunicación entre pares donde todos los dispositivos son iguales no hay jerarquía de roles.

La motivación para el uso de la tecnología LR-WPAN es la reducción en los gastos de instalación como el cableado estructurado e intercambio de información con un mínimo de esfuerzo de instalación. Los valores más característicos del estándar se presentan en la Tabla 1.2:

Tabla 1.2 Propiedades de IEEE.802.4.15. [13]

Bandas de frecuencia - Rango de transmisión de datos	868 MHz - 20 Kb/s 915 MHz - 40 Kb/s 2,4 GHz - 250 Kb/s
Alcance	10 - 20 m
Latencia	< 15 ms
Canales	868/915 MHZ: 11 Canales 2.4 GHz: 16 Canales
Modos de direccionamiento	64 bits IEEE
Canal de acceso	CSMA-CA (Acceso múltiple con detección de portadora y evasión de colisiones)
Seguridad	128 AES (Estándar de encriptación avanzado)
Red	Hasta 2^{64} dispositivos
Rango de temperatura	-40°a + 85° C

1.4.1.1 Arquitectura [7]

Como en todos los estándares IEEE 802, este toma como referencia el modelo ISO/OSI (*International Organization for Standardization /Open System Interconnection*), tal y como se indica de manera genérica en la Figura 1.18.

A continuación se explicarán con más detalle las capas Física y de Enlace de datos del Modelo de referencia IEEE 802.

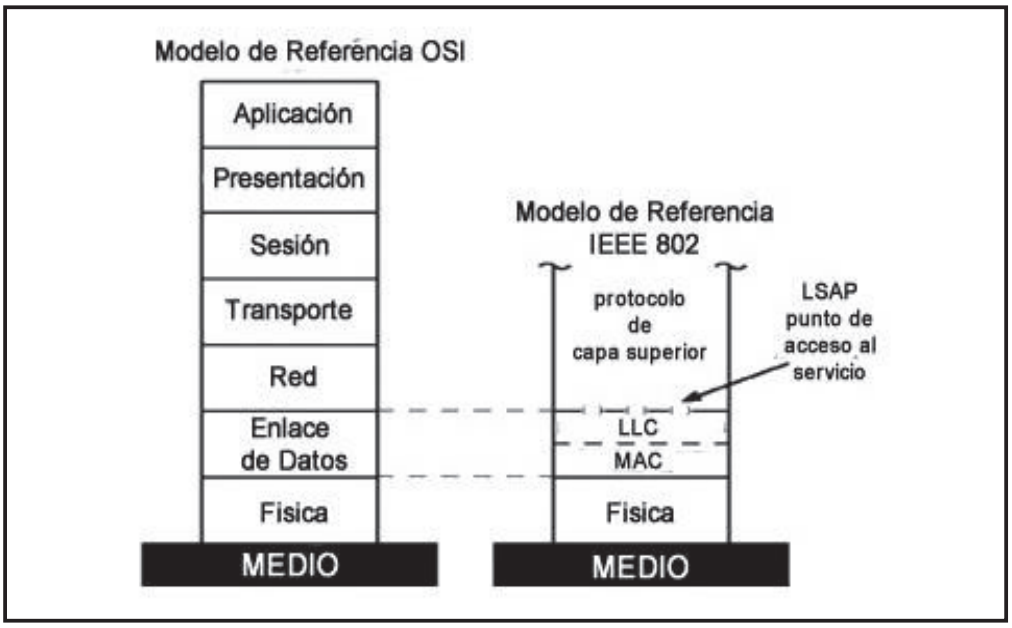


Figura 1.18 Relación entre IEEE 802 y el modelo de referencia ISO/OSI [14]

1.4.1.1.1 Capa física

El estándar IEEE 802.15.4 ofrece dos opciones de transmisión según la banda de frecuencias de operación (868/915 MHz y 2450 MHz) ambas basadas en la modulación DSSS (*Direct Sequence Spread Spectrum*), que usa un seudocódigo para modular una portadora, logrando aumentar el ancho de banda de la transmisión y reducir la densidad de potencia espectral.

La capa física al trabajar con una frecuencia de 2.4 GHz especifica su operación en la banda ISM (*Industrial, Scientific and Medical*) a nivel mundial y permite una velocidad de transmisión de 250 kb/s. Mientras que la capa física al trabajar a 868 MHz en Europa ofrece velocidades de 20 kb/s y al operar a 915 MHz ofrece velocidades de 40 kb/s, en la banda ISM en Estados Unidos.

Tanto la Capa física que trabaja a 2.4 GHz y a 868 MHz se combinan con el Control de Acceso al Medio (MAC) para permitir un amplio rango de aplicaciones en red.

1.4.1.1.2 Capa de enlace de datos

IEEE 802 divide la capa de enlace de datos (DLL, *Data Link Level*) en dos subcapas:

- La subcapa de enlace de acceso al medio (MAC) y,
- La de control de enlaces lógicos (*Logical Link Control*, LLC).

La subcapa LLC es común a todos los estándares 802, mientras que la subcapa MAC depende del hardware y varía respecto a la implementación física del sistema.

1.4.1.1.3 Control de Acceso al Medio (MAC, Media Control Access)

La capa MAC proporciona dos servicios: *MAC Data Service* y *MAC Management Service,* ambos servicios interactúan y permiten la transmisión y recepción de tramas MAC usando para ello el canal físico. Ofrece una interfaz de control y regula el acceso al canal físico y a la señalización de la red.

También controla la validación de las tramas y las asociaciones entre nodos, y garantiza ranuras de tiempo[9]. Por último, ofrece puntos de conexión para servicios seguros.

El formato general de las tramas MAC está diseñado para ser muy flexible y que se ajuste a las necesidades de las diferentes aplicaciones con diversas topologías de red, al mismo tiempo que se mantiene un protocolo simple.

1.4.1.1.4 Control de Enlace Lógico (LLC, Logical Link Control)

Es la más alta de las dos subcapas de enlace de datos definidas por el IEEE y la responsable del control de enlace lógico. La subcapa LLC maneja el control de errores, control del flujo, entramado y direccionamiento de la subcapa MAC.

1.4.2 ZIGBEE

Las redes *ZigBee* nacen alrededor del año 1998, con la idea de satisfacer los segmentos del mercado no atendidos por los estándares *WiFi* y *Bluetooth* para aplicaciones de bajo costo, bajo ciclo de servicio de conectividad y baja tasa transmisión de datos.

El enfoque inicial de esta tecnología fue desarrollar dispositivos que provean al usuario soluciones inalámbricas rentables, fáciles de usar, con altos índices de confiabilidad y seguridad.

---

[9] Ranuras de tiempo: Son intervalos de tiempo específicos para la transmisión de datos sobre un único canal de frecuencia, lo que posibilita que varios usuarios utilicen simultáneamente el canal sin interferir entre sí.

Su nombre tiene origen en la forma de comunicación de las abejas, las cuales “bailan” en zig-zag para informar una nueva fuente de alimento a la colonia de ahí el nombre que recibió este estándar donde Zig por el patrón de movimiento para comunicación entre sí y Bee por abeja.

El término ZigBee es un protocolo inalámbrico estandarizado para la conexión de una Red de Área Personal Inalámbrica o WPAN. La idea de promover este estándar, surge de permitir la interoperabilidad entre dispositivos de diferentes marcas.

1.4.2.1 Características de Zigbee

- Bajo consumo de energía.
- Los dispositivos de ZigBee son más ecológicos que sus predecesores.
- Bajo costo en la instalación y mantenimiento en los dispositivos.
- Los dispositivos ZigBee son capaces de extender la vida de las baterías sin necesidad de recargar las mismas por varios años.
- Redes de alta densidad de nodos.
- Presenta un stack de protocolos simple. Siendo esta simplicidad esencial para el costo, interoperatibilidad, y mantenimiento.
- Transferencia de datos de 20 kbps y de 250 kbps.

1.4.2.2 Tipos de Dispositivos en una Red *ZigBee* [11]

1.4.2.2.1 Full Function Device (FFD)

Es un dispositivo que posee memoria adicional, una gran capacidad de procesamiento y la habilidad de recibir mensajes del estándar 802.15.4, por lo que lo hacen ideal para cumplir con las funciones de Router o para ser usado como

dispositivos de red que interactúen con los usuarios. Este dispositivo puede soportar los siguientes modos de operación:

- Coordinador PAN (*Personal Area Network*), Identifica su propia red a la cual, otros dispositivos pueden asociarse con una sincronización global.
- Dispositivo Simple: puede actuar como ruteador o dispositivo final.

1.4.2.2.2 Reduced Function Device (RFD)

Es un dispositivo de capacidad y funcionalidad limitada es decir opera con la mínima implementación del protocolo IEEE 802.15.4. Pueden asociarse a un FFD a la vez y son usados para aplicaciones extremadamente simples.

1.4.2.3 Red ZigBee

Una red ZigBee puede tener un máximo de 65.000 (2^{16}) dispositivos, los cuales pueden ser programados para funcionar en diferentes topologías.

1.4.2.3.1 Topologías de Red [12]

El estándar ZigBee permite tres diferentes topologías de red que pueden ser implementadas dependiendo de la aplicación. A continuación se describe cada una de ellas.

1.4.2.3.1.1 Topología tipo estrella

En una topología en estrella se tiene un dispositivo tipo FFD como coordinador de red y los RFD como dispositivos finales. La comunicación en esta topología es centralizada es decir cada dispositivo se une a la red y si desea comunicarse con otros dispositivos debe enviar su información al coordinador, el cual enviará la información al dispositivo correspondiente. En la Figura 1.19, se aprecian un ejemplo de la topología tipo estrella.

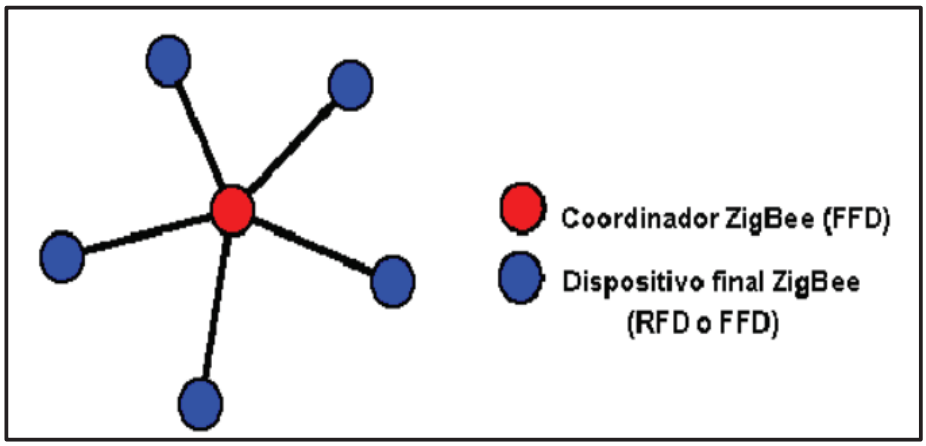


Figura 1.19 Red tipo Estrella. [16]

1.4.2.3.1.2 Topología en Malla

En la topología tipo malla todos los FFDs pueden trabajar como coordinador, router o dispositivos finales, mientras que los RFD únicamente se pueden utilizar como dispositivos finales, El coordinador ZigBee es el responsable de inicializar y elegir los parámetros de la red, y a la vez puede ser ampliada a través del uso de routers ZigBee ver Figura 1.20.

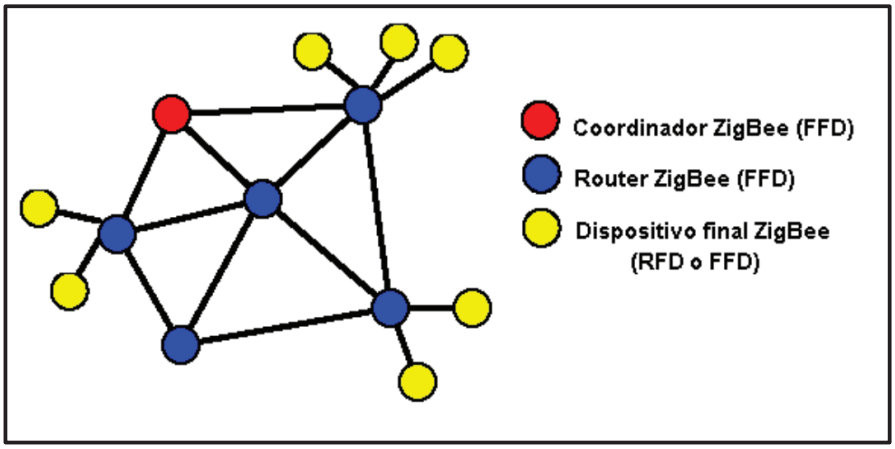


Figura 1.20 Red tipo Malla. [16]

1.4.2.3.1.3 Topología Cluster Tree

En la topología *Cluster Tree* varios dispositivos son FFDs y los RFD pueden conectarse como un nodo único al final de la red. Cualquiera de los FFDs restantes pueden actuar como coordinadores y proveer servicios de sincronización hacia otros dispositivos o coordinadores en diferentes redes, ver Figura 1.21

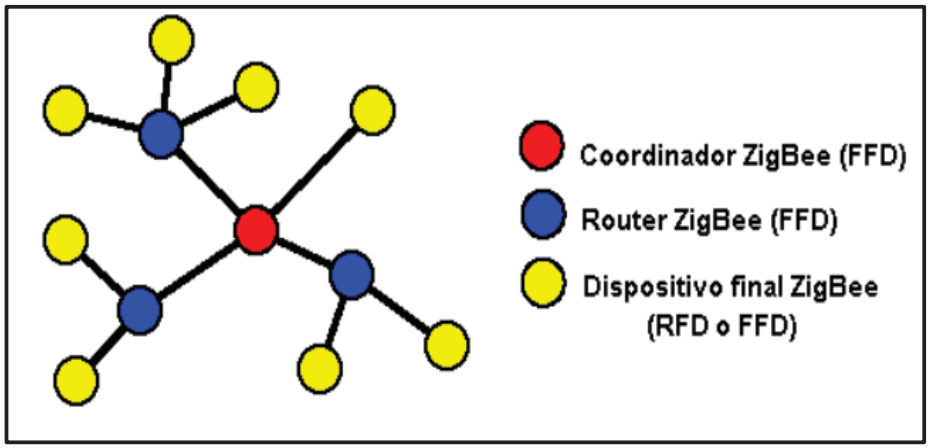


Figura 1.21 Red tipo *Cluster Tree*. [16]

1.4.2.4 Modos de Operación en *ZIGBEE*

1.4.2.4.1 Modo Beacon-habilitado

Usa una estructura de supertramas definida por el coordinador PAN para manejar la comunicación entre dispositivos enviando periódicamente en una trama beacon. El coordinador puede entrar en modo de bajo consumo durante el periodo de inactividad.

El coordinador se encarga de transmitir beacons cada cierto tiempo usando el medio CSMA-CA ranurado en donde las ranuras de backoff están alineadas con el comienzo de un beacon. En la Figura 1.22, se observa las partes activas de una supertrama.

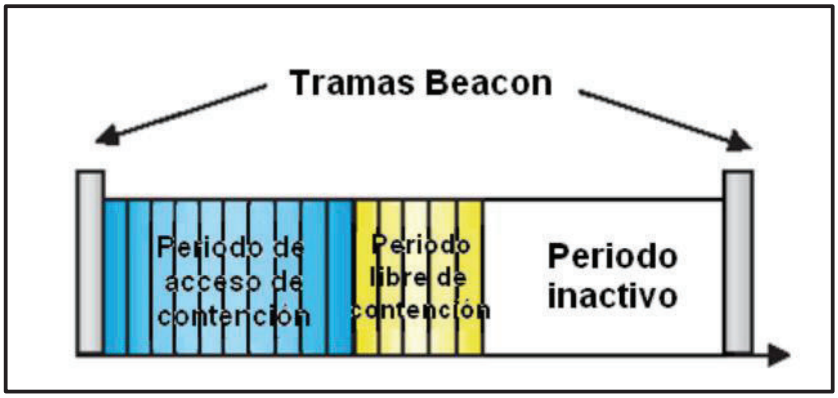


Figura 1.22 Partes del periodo activo de una supertrama. [16]

Donde una supertrama está delimitada por la transmisión de una trama beacon y puede tener un periodo activo y uno inactivo:

- Beacons
- Periodo de acceso por contención (*Contention Access Period*, CAP)
- Periodo libre de contención (*Contention Free Period*, CFP)

En la Figura 1.23 se observa la trasmisión usada en el modo Beacon- habilitado.

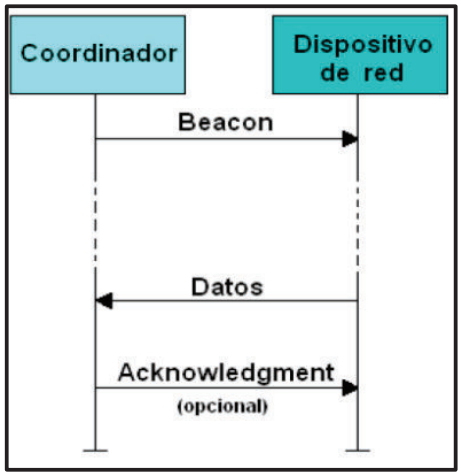


Figura 1.23 Transferencia de datos en modo Beacon-habilitado. [16]

1.4.2.4.2 Modo Beacon-no habilitado

Permite enviar datos mediante el mecanismo CSMA/CA no ranurado sin usar supertramas, lo que implica que los dispositivos transmiten en el momento que es necesario sin esperar ningún beacon de ningún dispositivo coordinador. Cada vez que un dispositivo quiere transmitir, espera un tiempo aleatorio, si encuentra un canal libre espera un tiempo de backoff e intenta transmitir, pero si el canal sigue ocupado después de concluir el tiempo de backoff vuelve a esperar otro tiempo aleatorio y otro de backoff, ver Figura 1.24.

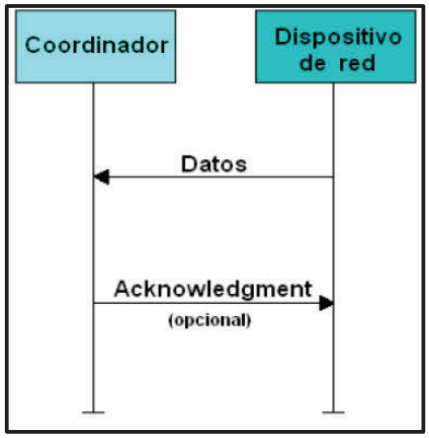


Figura 1.24 Transferencia de datos en modo Beacon- no habilitado. [16]

1.4.2.5 Conexión de dispositivos en una red ZigBee[11]

El bajo consumo de potencia es lo que hace que la tecnología ZigBee tenga un largo periodo de vida sin tener que recargar la batería de los dispositivos. Durante mucho tiempo, un dispositivo esclavo está en modo *sleep* (dormido) y pasa al modo *awake* (despierto) por una fracción de segundo para confirmar que está en línea en la red.

1.4.2.6 Comunicación en una red ZigBee [11]

Los datos se empiezan a enviar en las capas superiores de la pila, y cada capa añade su información propia, formando los PDU (*Protocol Data Unit*), es decir que cuando se envían un conjunto de datos, este contiene información de control de todas las capas de la pila y cuando llega a su destino, cada capa extrae los datos que le concierne y, si es posible o necesario, pasa a la capa superior. Así se produce una comunicación virtual entre capas de diferentes dispositivos, ver Figura 1.25.

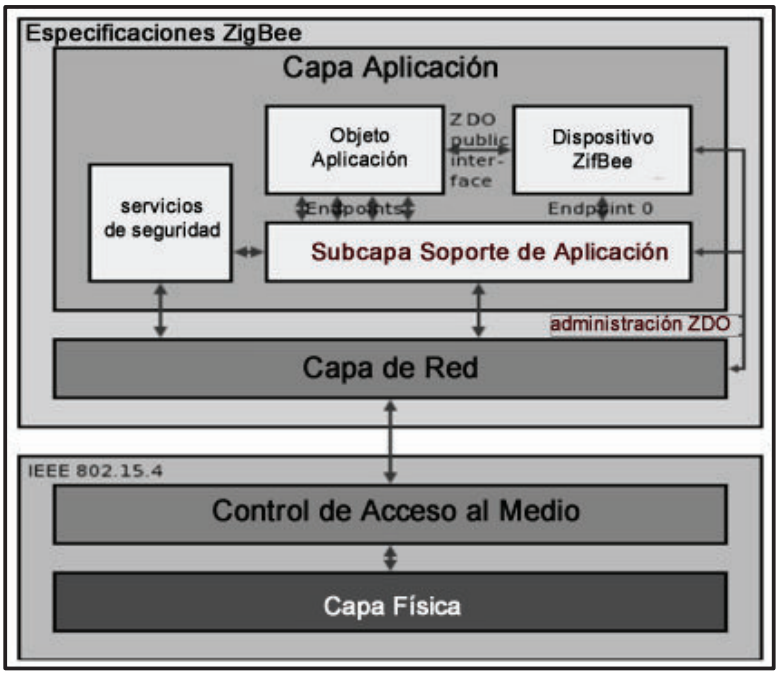


Figura 1.25 Capas del estándar ZigBee y 802.15.4. [13]

Para analizar la comunicación de datos se empezará con el nivel físico. Los dispositivos inalámbricos envían los datos usando ondas electromagnéticas; en este caso se utiliza modulación por frecuencia FSK[7], en el espectro de los 2.4 GHz. Para la transmisión se tiene disponible 16 canales separados entre sí 5 MHZ.

El acceso al canal se hace utilizando CSMA-CA[8] que es un mecanismo empleado para evitar que dos dispositivos produzcan una colisión al tratar de usar el mismo canal, cuando un dispositivo transmite el resto espera. Todos los dispositivos que estén en el alcance del transmisor podrán escuchar el mensaje, pero la mayoría de las veces solamente es necesario comunicarse con un solo dispositivo.

La capa superior o capa MAC provee una interface entre la capa física y la capa de red permitiendo identificar a los dispositivos dentro de la red. Es decir, cada dispositivo posee una dirección MAC de 64 bits, la cual es única.

Por su parte la capa de red del coordinador asigna direcciones de 16 bits a cada miembro de la PAN incluyendo un parámetro llamado radio que indica la cantidad de saltos máximos que esta puede llegar a realizar, cuando este llega a cero la trama no se retransmite a otro dispositivo.

Para cumplir su cometido, la capa de red proporciona dos servicios, uno de datos (NLDE, *Network Layer Data Entity*) y otro de gestión (NLME, *Network Layer Management Entity*). El NLDE encapsula los datos de la capa superior añadiendo las cabeceras necesarias, y los pasa a la capa MAC, para ser enviados a su destino mientras que el NLME utiliza el NLDE para lograr tareas de gestión y mantener una base de datos de objetos gestionados.

---

[7] FSK: (*Frequency-Shift Keying)* ,es un tipo de modulación de frecuencia cuya señal modulante es un flujo de pulsos binarios que varía entre valores predeterminados

[8] CSMA-CA: (*Carrier Sense Multiple Access, Collision Avoidance*), es un protocolo de control de redes de bajo nivel que permite que múltiples estaciones utilicen un mismo medio de transmisión.

Las capas *PHY-MAC-NWK* hacen posible crear una red completa, permitiendo a los nodos comunicarse con otros nodos de una misma red o de alguna distinta. Para permitir una mejor funcionalidad se suma una capa a la pila, esta interactuará entre los objetos de la aplicación que se desee usar y la capa de red, la mencionada es la subcapa de aplicación (APS, *Aplication Support sub-layer*).

La capa APS es la encargada de enviar los PDU´s de una aplicación a dos o más dispositivos, permite descubrir y enlazar dispositivos. Mantiene una base de datos de los objetivos controlados, esta es conocida como AIB (APS *Information Base*).

1.4.2.7 Seguridad en ZigBee [23]

Dadas las características de las redes inalámbricas, un mensaje puede ser recibido por cualquier dispositivo cercano a la red. Esto puede ser un grave problema para sistemas de seguridad, puesto que un intruso podría violar la privacidad o inhabilitar algún sistema. Es por esta razón, que ZigBee soporta el uso de protocolos estándar de autenticación y encriptación. Sin embargo a medida que se mejora el nivel de seguridad también se aumenta el costo del sistema, ya que se necesita de dispositivos con más memoria, capacidad de procesamiento y consumo energético.

La tecnología ZigBee, utiliza el modelo de seguridad de la subcapa MAC IEEE 802.15.4, la cual especifica 4 servicios de seguridad:

Control de accesos: El dispositivo mantiene una lista de los dispositivos "comprobados" en la red.

Datos Encriptados: Se utiliza AES (*Advance Encryption Standard*) del NIST (*National Institute of Standards and Technology*) como técnica de encriptación, el cual utiliza un código de 128 bits y tamaños de llave de 128, 192 o 256 bits.

Integración de tramas: Protege los datos para que no sean modificados por otros.

Secuencias de refresco: Sirve para comprobar que las tramas no han sido remplazadas por otras. El controlador de red comprueba estas tramas de refresco y su valor, para ver si son esperadas.

La decisión de dotar de más o menos seguridad al sistema final, dependerá exclusivamente del análisis costo - beneficio de las necesidades a satisfacer.

1.4.2.8 Autenticación

El propósito de autenticar dispositivos y datos, es asegurar que estos sean válidos y originales, es decir, que no hayan sufrido ninguna transformación durante la transmisión. Para lograr este objetivo el transmisor agrega a los datos un código especial que para la tecnología ZigBee toma el nombre de Código de Integración de Mensaje (MIC, *Message Integrity Code*). Este código de autenticación es conocido tanto por el transmisor como por el receptor, y un dispositivo ajeno a la red no debería poder crear el MIC correcto.

El código de integración de mensaje MIC en ZigBee, se genera usando el protocolo CCMP[10] (*Counter With Cipher Block Chaining Message Authentication Code*). El CCMP se usa en conjunción con AES de 128 bit y comparten la misma clave de seguridad.

En conclusión, cuando el receptor recibe el mensaje, inmediatamente calcula el MIC y lo compara con el MIC recibido junto con los datos originales, si existe

[10] CCMP (*Counter With Cipher Block Chaining Message Authentication Code*): es un protocolo de cifrado que forma parte de la 802.11i estándar para redes inalámbricas de área local. El algoritmo CCMP se basa en *Advanced Encryption Standard* del gobierno federal de Estados Unidos (AES).

coincidencia se considera al mensaje como autentico, caso contrario se lo descarta. El nivel de seguridad se incrementa con el número de bits del MIC. ZigBee y 802.15.4 soportan MIC de 32, 64 y 128 bits. Si adicionalmente se desea tener confidencialidad se debe encriptar el mensaje.

1.5 LA NUBE

Formalmente la “nube”, que en inglés se la llama *Cloud computing,* es el nombre que se le da al procesamiento y almacenamiento masivo de datos en servidores que alojen la información del usuario, es decir, que existen servidores gratuitos y/o pagados que ofrecen guardar y procesar archivos e información desde Internet.

Este concepto nace con la idea de acceso instantáneo y en todo momento, es decir, esté donde esté a través de dispositivos móviles, computadores o notebooks con la simple condición que estas cuenten con acceso a Internet.
La principal característica, aunque no es la única, es que la “Nube” no fue creada para personas expertas en tecnología, sino para usuarios comunes que desean solucionar las cosas de manera rápida y simple.

1.5.1 PRINCIPALES APLICACIONES DE LA NUBE

1.5.1.1 Dropbox

Es básicamente un disco duro en Internet. Este servicio proporciona espacio de almacenamiento en sus servidores en los cuales se puede guardar cualquier tipo de información (archivos de texto, videos, canciones o cualquier archivo de un computador). El servicio es completamente gratuito en su versión básica aunque también se ofrece una versión *premium* por un pago mensual.

1.5.1.2 Google Docs y Calendar.

Este servicio es una alternativa ideal para Microsoft Office, el cual permite crear o editar archivos de texto, hojas de cálculo, presentaciones, archivos PDF, entre otros. Una de sus características principales y funcionales con las que cuenta este servicio, es que guarda los archivos constantemente, evitando la pérdida de información por cualquier inconveniente de tipo.

Google Calendar es un servicio de calendario y agenda online. Permite organizar fechas y sincronizar la información con los contactos de Gmail y así compartir o invitarlos a diferentes eventos. Tanto con Google Docs como con Google Calendar, el usuario debe tener una dirección de correo en Gmail.

1.5.1.3 Grooveshark

Grooveshark permite escuchar música a través de Internet de manera completamente gratuita. Este sitio es ideal para los usuarios que no tienen mucha música alojada en su PC o aquellos que gustan de escuchar música nueva constantemente.

1.5.1.4 Picasa – Flickr

Son servicios que permiten un almacenamiento de imágenes, siendo los más importantes de la red: Flickr es de Yahoo! y Picasa de Google.

1.5.1.5 Zoho

Es una plataforma similar a Google Docs, la misma ofrece aplicaciones online para oficina pero de forma mucho más profesional y con una mayor cantidad de opciones, entre las más importantes tenemos:

- **Zoho Writer:** Permite crear documentos de texto.
- **Zoho Sheet:** Facilita la creación de hojas de cálculo.
- **Zoho Show:** Sirve para crear diapositivas.
- **Zoho Wiki:** Se usa para editar archivos wiki (como la Wikipedia).
- **Zoho Notebook:** Permite crear documentos de texto con videos, audio e imágenes.
- **Zoho Meeting:** Soporta la realización de videoconferencias entre una o más personas.
- **Zoho Projects:** Administra proyectos de cualquier tipo como asignación de tareas, tiempos para realizarlas, personas vinculadas, entre otros.
- **Zoho CRM:** Sirve para administrar y gestionar una agenda de contactos, tareas, gestión de grupos y más.

En este capítulo se realizó una breve descripción de los sistemas domóticos, tipos de arquitecturas y dispositivos que lo conforman, además se explicó las principales características de la plataforma Arduino, así como los principales modelos comerciales. Adicionalmente se realizó una breve reseña de la evolución del sistema operativo Android.

En el siguiente capítulo se detallará el diseño y los algoritmos que han servido para desarrollar el prototipo domótico.

CAPÍTULO 2

DISEÑO E IMPLEMENTACIÓN DE LA PLATAFORMA ARDUINO EN EL SISTEMA DOMÓTICO

El prototipo domótico a ser diseñado estará conformado por un nodo central y cinco nodos secundarios; es decir con una arquitectura punto – multipunto entre el nodo central y los nodos cortina, puerta, temperatura y dimmer y una arquitectura punto – punto entre el nodo central y el nodo de la cámara. El nodo central actúa como coordinador de la red, iniciando y verificando el estado de la misma mientras que los nodos secundarios reciben y procesan la información recibida por el central, ejecutando así la petición del usuario que se origina en la aplicación Android. En la Figura 2.1, se muestra el esquema global del sistema a diseñarse.

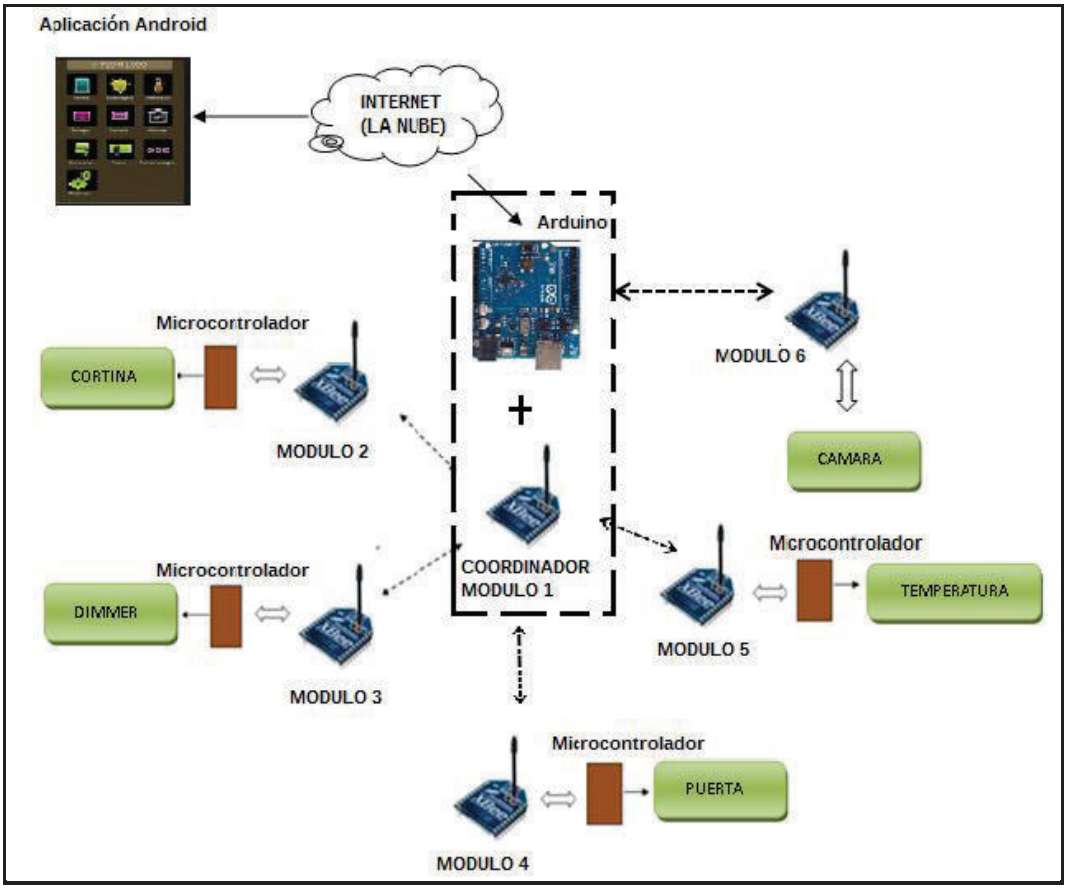


Figura 2.1 Esquema global del sistema domótico.

2.1 MÓDULO INALÁMBRICO

El módulo XBee Pro Serie Uno de la empresa *Maxtream* con referencia XBP24-AWI-001, fue seleccionado para la implementación del prototipo debido a las características que éste presenta:

- Eficiente comunicación serial.
- Bajo consumo de potencia.
- Alcance de cobertura significativo que a su vez permita obtener un enlace inalámbrico estable.

En la Figura 2.2 se muestra el módulo seleccionado.

Figura 2.2 Módulo XBP24-AWI-001. [1]

Rendimiento:

- Alcance (interiores): 100 mts
- Alcance (exteriores): 1,6 Km
- Potencia de transmisión: 60 mW (+18 dBm)
- Sensibilidad de recepción(1%): -100 dBm

- Velocidad de transmisión: 250 Kbps

Requerimientos de Potencia:

- Suministro de voltaje: 2.8 - 3.4 VDC (Absoluto) / 3.0 - 3.4 (recomendado)
- Corriente de transmisión: 215 mA
- Corriente de recepción: 55 mA

Información general:

- Banda de frecuencia: ISM 2.4 GHz
- Dimensiones: 24.38 mm x 27.61 mm, alto de antena: 25 mm
- Tipo de antena: Cable
- Completamente certificado por la FCC (Comisión Federal de Comunicaciones, *Federal Communications Commission*)
- Transmisión: Bidireccional
- Interfaz en serie: UART (Transmisor-Receptor Asíncrono Universal, *Universal Asynchronous Receiver Transmitter*).
- Modos de configuración: AT y API
- Modulación: DSSS (Espectro ensanchado por secuencia directa, *Direct Sequence Spread Spectrum*).
- Entradas ADC: 7 de 10-bit son para entradas ADC (Convertidor Analógico a Digital, *Analog to Digital Converter*)
- Pines digitales I/O : 8 pines
- 12 canales de secuencia directa
- Cifrado: 128 bits

2.1.1 DISTRIBUCIÓN DE PINES DEL MÓDULO XBP24-AWI-001

En la Tabla 2.1 se puede observar la distribución de pines del Módulo XBee.

Tabla 2.1 Descripción de los pines del Módulo XBP24-AWI-001. [1]

PIN	DENOMINACIÓN	DESCRIPCIÓN
1	VCC	Fuente de alimentación
2	DOUT	UART salida de datos
3	$\mathrm{DIN}/\overline{\mathrm{CONFIG}}$	UART entrada de datos
4	DO8*	Salida digital 8
5	RESET	Pin para reset, este pulso debe ser menor a 200ns
6	PWMO/RSSI	Salida PWM0 o Indicador de nivel de Señal Rx
7	[reserved]	Sin conexión
8	[reserved]	Sin conexión
9	$\overline{\mathrm{DTR}}/\mathrm{SLEEP\_RQ}/\mathrm{DI8}$	Control de modo dormido o entrada digital 8
10	GND	Tierra
11	AD4/DIO4	Entrada analógica 4 o I/O Digital 4
12	$\overline{\mathrm{CTS}}/\mathrm{DIO7}$	Control de flujo Clear to Send o I/O Digital 7
13	ON/SLEEP	Indicador de estado de módulo
14	VREF	Voltaje de referencia para entradas AD
15	Associate/AD5/DIO5	Pin asociado, entrada analógica 5 o I/O 5 Digital
16	RTS/AD6/DIO6	Control de flujo Request to Send, entrada analógica 6 o 6 I/O
17	AD3/DIO3	Entrada analógica 3 o I/O Digital 3
18	AD2/DIO2	Entrada analógica 2 o I/O Digital 2
19	AD1/DIO1	Entrada analógica 1 o I/O Digital 1
20	AD0/DIO0	Entrada analógica 0 o I/O Digital 0

2.1.2 MODOS DE OPERACIÓN.

Los módulos XBee pueden trabajar con comunicación serial asincrónica, con niveles lógicos TTL (Lógica transistor a transistor, *Transistor–transistor logic)*, por medio de una interfaz UART (Transmisor - Receptor Asíncrono Universal, *Universal Asynchronous Receiver Transmitter*), ya sea en arquitecturas punto - punto, punto - multi-punto o en una red malla. La velocidad de transmisión se ajusta a las necesidades de la red como tipo de conexión física, velocidad negociada entre los dispositivos, limitación controlada de la velocidad, interferencias o ruidos en la conexión física entre otras, la misma se puede modificar por software. Estos módulos proveen dos formas amigables de comunicación:

2.1.2.1 Modo Transparente

Este modo permite ingresar comandos AT al módulo XBee, para configurar, ajustar o modificar parámetros. En este modo los módulos actúan de forma que todos los datos se reciben a través de la UART, siendo la configuración más sencilla y principalmente usada en redes punto a punto aunque también se la usa en redes punto - multipunto. Esta información se detalla en el Anexo A.

2.1.2.2 Modo API (Interfaz de programación de aplicaciones, *Application Programming Interface*)

En este modo la comunicación se realiza mediante tramas. Estas permiten simplificar la comunicación con varios módulos remotos, ya que es posible identificar el origen y seleccionar el destino de la información dentro de la misma trama sin tener que modificarlo vía comandos, asegurando así la entrega de la información.

Existen dos posibilidades de configuración en modo API. La primera es conocida como modo normal $(\mathrm{AP} = 1)$ en el que la trama inicia con el carácter 0x7E después de una pausa de transmisión de datos, mientras que la segunda forma es conocida como modo con escape $(\mathrm{AP} = 2)$ donde se garantiza de manera obligatoria que el carácter 0x7E sólo aparece en la secuencia de datos para indicar el inicio de una trama, se debe detectar las secuencias de escape y reemplazarlas por los caracteres originales en la recepción cuando se envía bytes que interfieren con la estructura de la trama.

La opción de operación API facilita muchas operaciones como:

- Transmisión de datos múltiples destinos, sin entrar en modo de comando.
- Recibir el éxito o fracaso de cada paquete transmitido
- Identificar la dirección de origen de cada paquete recibido.

En la Figura 2.3, se puede visualizar el esquema de una trama en modo API.

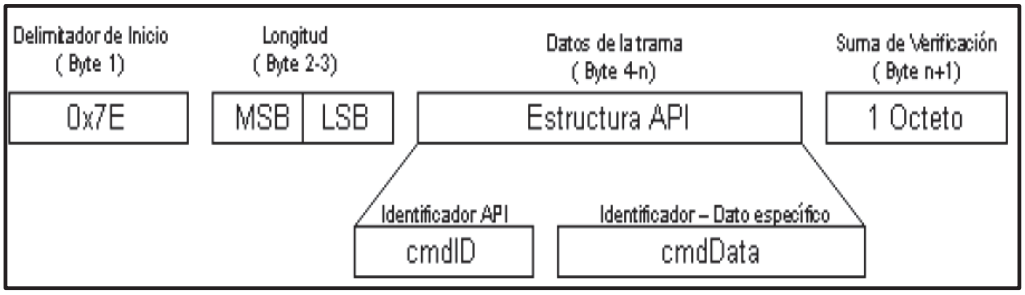


Figura 2.3 Estructura de la trama API. [2]

En el campo de datos de la trama se escribe un determinado tipo de mensaje especificado por el identificador API. En la Tabla 2.2, se indica el tipo de mensaje y el identificador API en hexadecimal.

Tabla 2.2 Identificador API. [2]

Identificador API	Tipo de mensaje
0x8A	Estado del módulo

0x08	Comando
0x09	Aplicar un Comando
0x88	Respuesta a un comando
0x01	Requerimiento de transmisión(dirección 16 bits)
0x89	Estado de la transmisión
0x81	Paquete recibido (dirección 16 bits)

A continuación se presenta un ejemplo de la implementación de una trama en modo API entre dos módulos XBee:

7E 00 0A 01 01 50 01 00 48 65 6C 6C 6F B8

Esta trama fue enviada desde el módulo transmisor observándose como mensaje la palabra Hola en el módulo receptor, a continuación se detallan las partes que conforman dicha trama.

7E	Delimitador de inicio, *Startd delimiter*
00 0A	Longitud en bytes, *Length bytes*
01	Identificador API, *API identifier*
01	Identificador de la trama API, *API frame ID*
50 01	Dirección de destino bajo*, Destination Address Low*
00	Opción de byte, *Option byte*
48 65 6C 6C 6F	Paquetes de datos, *Data packet*
B8	Chequeo de pariedad, *Checksum*

Después de analizar ambos modos se llegó a la conclusión de que el Modo transparente es el más adecuado para la elaboración de este sistema domótico, pues este facilita la ejecución de las órdenes recibidas y permite configurar

parámetros específicos que establecen una comunicación en una red punto – punto o punto – multipunto ya que en el modo API se debe generar la trama para transmitirla y a su vez se debe leer completamente para así obtener el dato necesario lo cual genera retardos en el sistema a implementarse.

2.2 SELECCIÓN DE LA TARJETA ARDUINO

Se seleccionó la tarjeta Arduino Mega para el Nodo Central ya que esta cumple con la comunicación serial, manejo de control de potencia, entre otros beneficios como 54 pines digitales que funcionan como entrada/salida; 16 entradas análogas, un cristal oscilador de 16 MHz, una conexión USB, un botón de reset y una entrada para la alimentación de la placa. Adicionalmente el Arduino Mega presenta las siguientes características:

- Un microcontrolador ATmega2560.
- Voltaje operativo de 5V.
- Voltaje de entrada de 7-12 V (límites) de 6 - 20 V.
- Pines digitales de I/O: 54(15 de ellos proveen salida PWM)
- Pines analógicos de entrada: 16
- Corriente DC por cada Pin I/O: 40 mA.
- Corriente DC entregada en el Pin: 3.3 V, 50 mA.
- Memoria Flash de 256KB (8KB usados por el *bootloader*)
- Memoria SRAM de 8KB.
- Memoria EEPROM de 4KB.
- Velocidad de reloj 16 MHz

A continuación en la Tabla 2.3, se muestra la asignación de pines para el Arduino2560.

Tabla 2.3 Distribución de pines del Arduino MEGA. [19]

PIN	DENOMINCACIÓN	DESCRIPCIÓN
1	PG5 (OC0B)	Digital pin 4 (PWM)
2	PE0 (RXD0/PCINT8)	Digital pin 0 (RX0)
3	PE1 (TXD0)	Digital pin 1 (TX0)
4	PE2 (XCK0/AIN0)	
5	PE3 (OC3A/AIN1)	Digital pin 5 (PWM)
6	PE4 (OC3B/INT4)	Digital pin 2 (PWM)
7	PE5 (OC3C/INT5)	Digital pin 3 (PWM)
8	PE6 (T3/INT6)	
9	PE7 (CLKO/ICP3/INT7)	
10	VCC	VCC
11	GND	GND
12	PH0 (RXD2)	Digital pin 17 (RX2)
13	PH1 (TXD2)	Digital pin 16 (TX2)
14	PH2 (XCK2)	
15	PH3 (OC4A)	Digital pin 6 (PWM)
16	PH4 (OC4B)	Digital pin 7 (PWM)
17	PH5 (OC4C)	Digital pin 8 (PWM)
18	PH6 (OC2B)	Digital pin 9 (PWM)
19	PB0 (SS/PCINT0)	Digital pin 53 (SS)
20	PB1 (SCK/PCINT1)	Digital pin 52 (SCK)
21	PB2 (MOSI/PCINT2)	Digital pin 51 (MOSI)
22	PB3 (MISO/PCINT3)	Digital pin 50 (MISO)
23	PB4 (OC2A/PCINT4)	Digital pin 10 (PWM)
24	PB5 (OC1A/PCINT5)	Digital pin 11 (PWM)
25	PB6 (OC1B/PCINT6)	Digital pin 12 (PWM)
26	PB7 (OC0A/OC1C/PCINT7)	Digital pin 13 (PWM)
27	PH7 (T4)	
28	PG3 (TOSC2)	
29	PG4 (TOSC1)	
30	RESET	RESET

31	VCC	VCC
32	GND	GND
33	XTAL2	XTAL2
34	XTAL1	XTAL1
35	PL0 (ICP4)	Digital pin 49
36	PL1 (ICP5)	Digital pin 48
37	PL2 (T5)	Digital pin 47
38	PL3 (OC5A)	Digital pin 46 (PWM)
39	PL4 (OC5B)	Digital pin 45 (PWM)
40	PL5 (OC5C)	Digital pin 44 (PWM)
41	PL6	Digital pin 43
42	PL7	Digital pin 42
43	PD0 (SCL/INT0)	Digital pin 21 (SCL)
44	PD1 (SDA/INT1)	Digital pin 20 (SDA)
45	PD2 (RXDI/INT2)	Digital pin 19 (RX1)
46	PD3 (TXD1/INT3)	Digital pin 18 (TX1)
47	PD4 (ICP1)	
48	PD5 (XCK1)	
49	PD6 (T1)	
50	PD7 (T0)	Digital pin 38
51	PG0 (WR)	Digital pin 41
52	PG1 (RD)	Digital pin 40
53	PC0 (A8)	Digital pin 37
54	PC1 (A9)	Digital pin 36
55	PC2 (A10)	Digital pin 35
56	PC3 (A11)	Digital pin 34
57	PC4 (A12)	Digital pin 33
58	PC5 (A13)	Digital pin 32
59	PC6 (A14)	Digital pin 31
60	PC7 (A15)	Digital pin 30
61	VCC	VCC
62	GND	GND
63	PJ0 (RXD3/PCINT9)	Digital pin 15 (RX3)
64	PJ1 (TXD3/PCINT10)	Digital pin 14 (TX3)
65	PJ2 (XCK3/PCINT11)	

66	PJ3 (PCINT12)	
67	PJ4 (PCINT13)	
68	PJ5 (PCINT14)	
69	PJ6 (PCINT 15)	
70	PG2 (ALE)	Digital pin 39
71	PA7 (AD7)	Digital pin 29
72	PA6 (AD6)	Digital pin 28
73	PA5 (AD5)	Digital pin 27
74	PA4 (AD4)	Digital pin 26
75	PA3 (AD3)	Digital pin 25
76	PA2 (AD2)	Digital pin 24
77	PA1 (AD1)	Digital pin 23
78	PA0 (AD0)	Digital pin 22
79	PJ7	
80	VCC	VCC
81	GND	GND
82	PK7 (ADC15/PCINT23)	Analog pin 15
83	PK6 (ADC14/PCINT22)	Analog pin 14
84	PK5 (ADC13/PCINT21)	Analog pin 13
85	PK4 (ADC12/PCINT20)	Analog pin 12
86	PK3 (ADC11/PCINT19)	Analog pin 11
87	PK2 (ADC10/PCINT18)	Analog pin 10
88	PK1 (ADC9/PCINT17)	Analog pin 9
89	PK0 (ADC8/PCINT16)	Analog pin 8
90	PF7 (ADC7)	Analog pin 7
91	PF6 (ADC6)	Analog pin 6
92	PF5 (ADC5/TMS)	Analog pin 5
93	PF4 (ADC4/TMK)	Analog pin 4
94	PF3 (ADC3)	Analog pin 3
95	PF2 (ADC2)	Analog pin 2
96	PF1 (ADC1)	Analog pin 1
97	PF0 (ADC0)	Analog pin 0
98	AREF	Analog reference
99	GND	GND
100	AVCC	VCC

En la Figura 2.4, se muestra los principales pines de la tarjeta Arduino Mega.

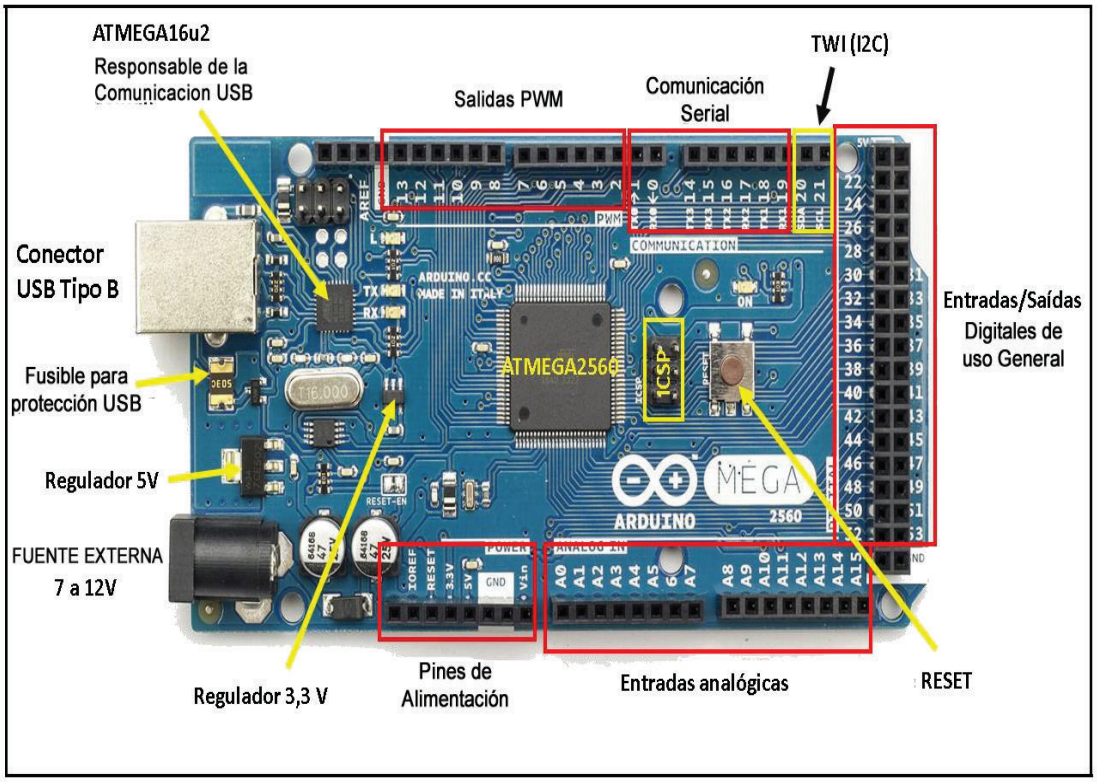


Figura 2.4 Principales pines de la tarjeta Arduino Mega.

2.3 SELECCIÓN DEL MICROCONTROLADOR

El microcontrolador *Atmega* 164P se seleccionó para los nodos secundarios del sistema, ya que cumple con la comunicación serial de estado, entre otros beneficios. Esta información se detalla en el Anexo B. En la Figura 2.5, se puede observar la distribución de pines del microcontrolador ATMEGA 164p.

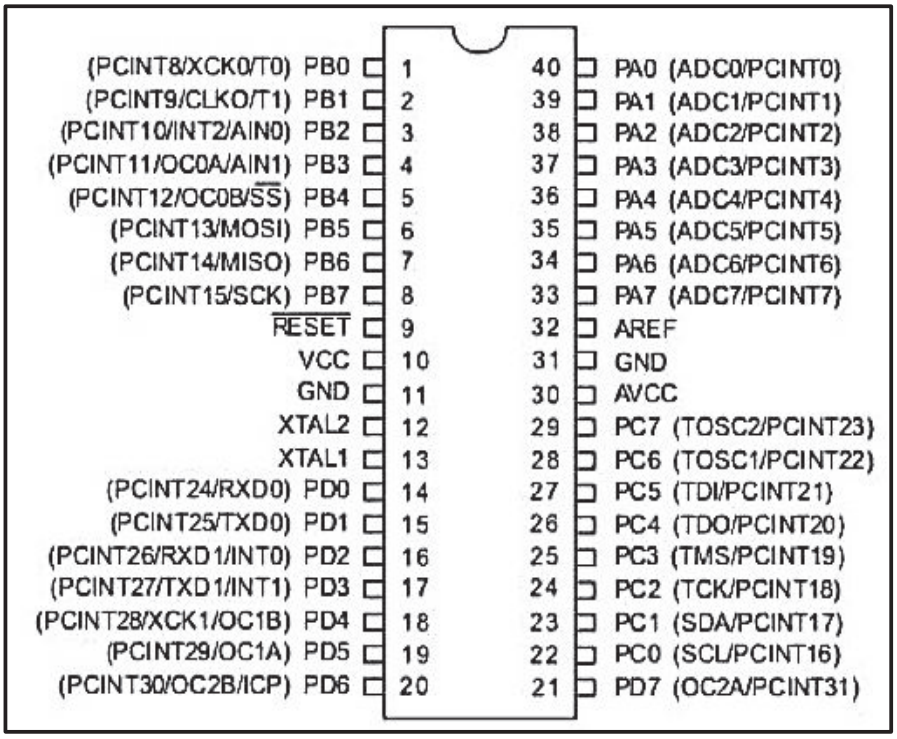


Figura 2.5 Distribución de pines del microcontrolador ATMEGA164p. [6]

Este tipo de microcontrolador AVR de 8 bits de alto rendimiento y bajo consumo presenta las siguientes características:

- 16 Kbytes de Flash autoprogramable en sistema.
- 131 instrucciones. La mayoría de un ciclo de reloj de ejecución.
- Funcionamiento estático total.
- 512 Bytes de EEPROM.
- 1 Kbyte de SRAM interna.
- Ciclos de escritura/borrado: 10.000 en Flash / 100.000 en EEPROM.
- 32 registros de trabajo de 8 bits para propósito general.
- ADC de 10 bits y 8 canales.
- Capacidad de procesamiento de unos 20 MIPS a 20MHz.

- Multiplicador por hardware de 2 ciclos.
- 6 Canales para PWM.
- Fuentes de interrupción externas e internas.
- ADC de 10 bits y 8 canales.
- Interface serie SPI maestro/esclavo.
- Dos puertos seriales USART programable.
- Dos Timer/Contadores de 8 bits con prescalamiento separado y modo de comparación.
- Un Timer/Contador de 16 bits con prescalamiento separado, modo comparación y modo de captura.
- Comparador analógico.
- Seis modos de descanso: Idle, Reducción de Ruido ADC, Power-save, Power-down, Standby y Standby extendido.
- Velocidad de funcionamiento 0 - 4MHz a 0.8 - 5.5V / 10MHz a 2.7 - 5.5V.
- Full dúplex.
- Tres líneas para comunicaciones asincrónicas.
- Siete velocidades programables.

2.4 CONFIGURACIÓN DE LOS MÓDULOS XBEE PRO [12]

Para configurar los módulos XBee Pro la empresa DIGI creó un *software* conocido como X-CTU (*Configuration & Test Utility Software)* el mismo que fue diseñado para interactuar con el *firmware* de cada dispositivo mediante una interfaz gráfica

para la configuración e iteración con los módulos de radiofrecuencia. El *software* aplicación X-CTU presenta cuatro pestañas, cada una tiene una función diferente:

Range Test: Permite al usuario realizar pruebas de rangos de alcance de la señal entre dos módulos, mediante el envío de paquetes de datos esperando el mismo paquete de respuesta dentro de un tiempo específico ver Figura 2.6.

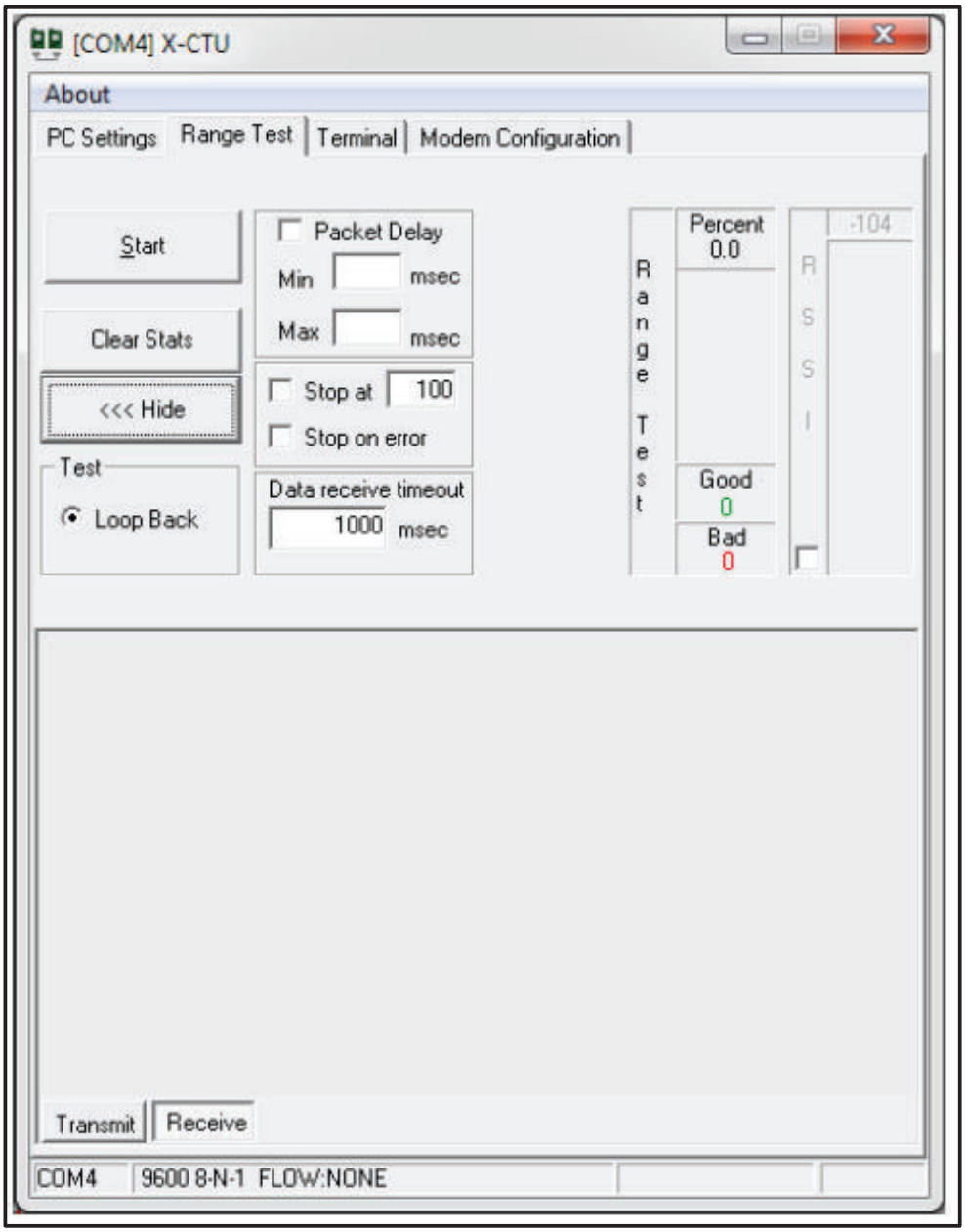


Figura 2.6 Pantalla del software de configuración X-CTU, Opción *Range Test*.

Por defecto el paquete de datos enviado es de 32 bytes en el mismo se puede ajustar el tamaño o el texto enviado.

Terminal: Presenta una pantalla similar al del *Hyperterminal* en la cual se ingresa directamente al módulo como si estuviera en el modo de comandos. Las funciones básicas de esta pestaña son:

- Emulador de Terminal.
- Capacidad para envío y recepción de paquetes predefinidos compactos, con el botón *Assemble Packet*.
- Capacidad de enviar y recibir datos en formato hexadecimal y ASCII (Código Estándar Estadounidense para el Intercambio de Información, *American Standard Code for Information Interchange*).

La opción *Assemble Packet* disponible en terminal, ha sido diseñada para permitir al usuario ensamblar un paquete, en forma ASCII o Hexadecimal ver Figura 2.7.

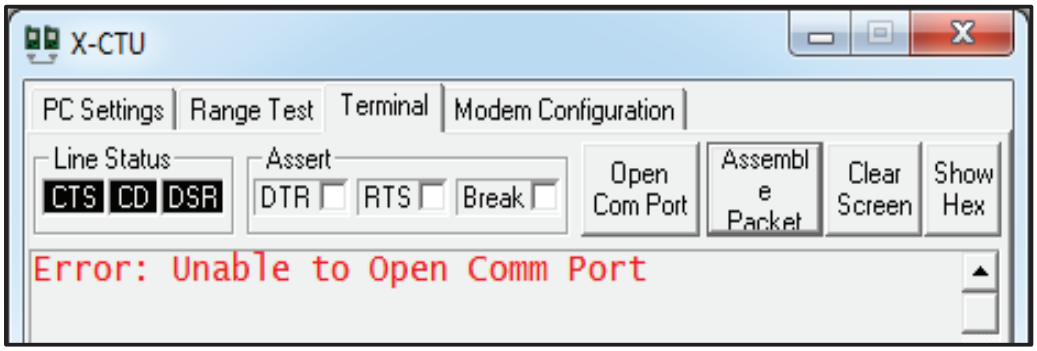


Figura 2.7 Opción para ensamblar paquetes.

PC Settings: Esta pestaña se divide en tres áreas básicas:

COM Port Setup: En este apartado se puede modificar parámetros como velocidad de transmisión estándar o no estándar, control de flujo, bits de datos, paridad, bits de parada.

Host Setup: En esta sección se permite al usuario colocar el modo de programación con el que se quiere configurar su dispositivo.

User COM Ports: En esta sección se permite al usuario colocar o borrar los puertos para la configuración de manera temporal.

Test/Query: Este botón permite probar la configuración seleccionada, de ser correcta muestra un mensaje en el que se indica el establecimiento de la comunicación con el modem ha sido exitosa ver Figura 2.8.

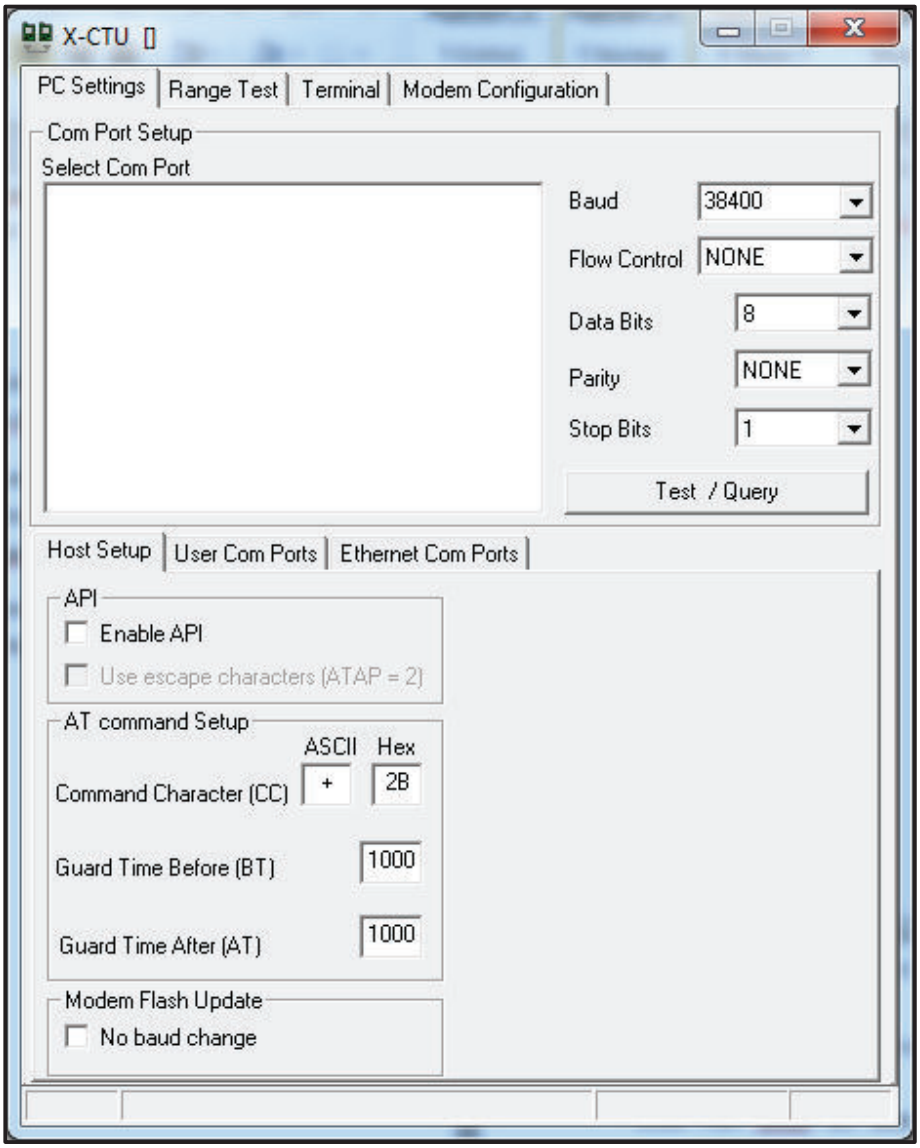


Figura 2.8 Configuración del PC Settings del módulo XBee Pro Serie Uno.

Modem Configuration: En este menú se permite especificar los parámetros del módulo a través de una interfaz gráfica, así como guardar y leer el perfil de un modem y además cuenta con la opción de actualizar el firmware descargando archivos ya sea desde el Internet o almacenado en la PC.

Cada parámetro tiene un color específico, el negro asegura que el parámetro es no configurable o sólo de lectura, el color verde indica que es un valor por defecto y el color azul indica que el parámetro fue modificado por el usuario. Un ejemplo se puede visualizar en la Figura 2.9.

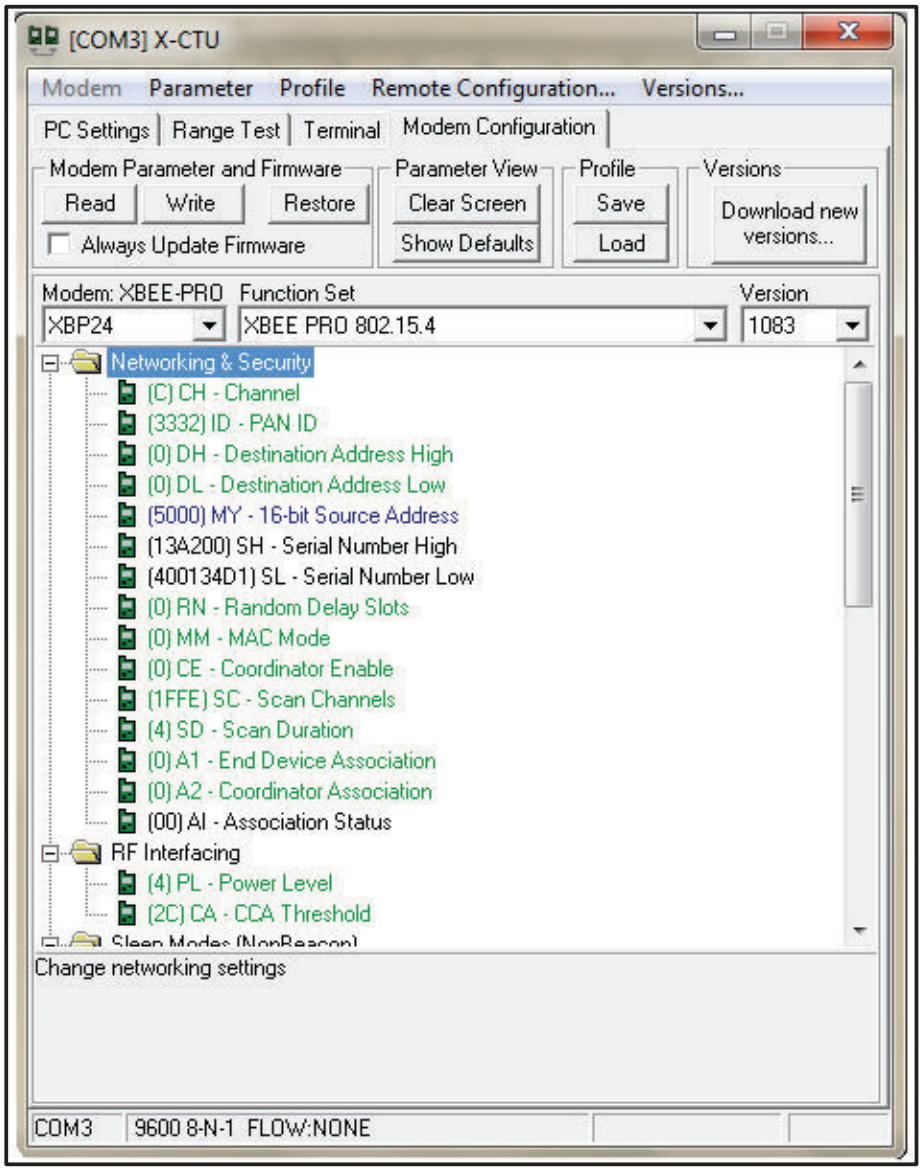


Figura 2.9 Modo de configuración del módulo XBee.

Los parámetros a modificar en cada módulo XBee dependen de la topología a ser usada en la red. Para la topología punto – multipunto y punto - punto los parámetros a configurar son:

Dirección MY: Dirección física de origen.
Canal: Canal de comunicación.
PAN ID: Identificación de la red.
DH: Dirección física de destino (Bytes más significantes)
DL: Dirección física de destino (Bytes menos significantes), este parámetro es indistinto cuando se envían los datos en modo API.

En la Tabla 2.4 se puede observar la configuración de los módulos XBee para la red Punto – Multipunto y en la Tabla 2.5 se puede observar la configuración de los módulos XBee para la red Punto – Punto.

. **Tabla 2.4** Parámetros de configuración módulos XBee red Punto - Multipunto.

NODOS DE LA RED	DIRECCION MY	CANAL	PAN ID	DH	DL	Vtx bps
Nodo Central (Coordinador)	AAAA	C	3332	0	AAAA	38400
Nodo Cortina	AAAA	C	3332	0	AAAA	38400
Nodo Dimmer	AAAA	C	3332	0	AAAA	38400
Nodo Acceso	AAAA	C	3332	0	AAAA	38400
Nodo Temperatura	AAAA	C	3332	0	AAAA	38400

Tabla 2.5 Parámetros de configuración módulos XBee red Punto - Punto.

NODOS DE LA RED	DIRECCION MY	CANAL	PAN ID	DH	DL	Vtx bps
Nodo Central (Coordinador)	EACD	C	ABCF	0	DC8A	38400
Nodo Cámara	DC8A	C	ABCF	0	EACD	38400

2.5 DISEÑO DEL CIRCUITO CENTRAL

El nodo central está conformado por un Modem/Switch, el Arduino Mega y un módulo *ZigBee*. Aquí se busca gestionar la información proveniente de la aplicación Android y dirigida a los nodos secundarios.

A continuación se presenta un diagrama que ilustra la estructura del nodo central ver Figura 2.10.

Figura 2.10 Diagrama de bloques del nodo central.

Para el control del sistema domótico se ejecuta la aplicación embebida en un dispositivo Android que permite comandar el Arduino Mega desde cualquier parte con la única condición que el dispositivo tenga una conexión a Internet.

En el Arduino Mega se mantendrá el control centralizado sobre los distintos nodos secundarios, se procesarán las peticiones provenientes de la aplicación Android y las confirmaciones a través del módulo *ZigBee* coordinador, el mismo que cambiará su configuración inicial dependiendo de la topología que se requiera.

La comunicación entre el Arduino Mega y el módulo *ZigBee* se realizará a través del periférico UART que ambos dispositivos poseen. La información que llega al módulo proveniente del Arduino Mega será transmitida inalámbricamente a los demás nodos esclavos del sistema domótico. En la Figura 2.11, se observa la conexión que se implementó para una correcta comunicación entre el Arduino Mega2560 y el módulo *ZigBee* coordinador utilizando como herramienta el simulador *Proteus*.

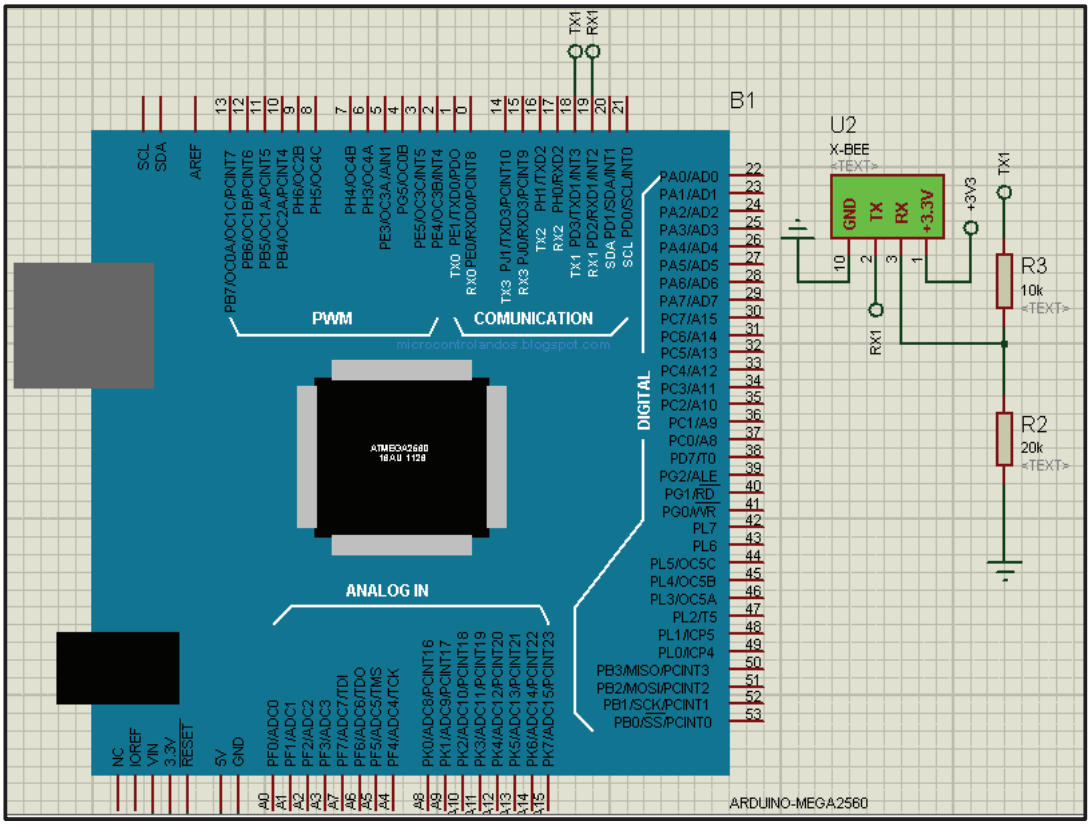


Figura 2.11 Diagrama de conexión entre el módulo XBee PRO y el Arduino Mega2560.

Es pertinente mencionar que el voltaje en la salida del Arduino Mega2560 es de 5 voltios y puesto que el nivel de tensión de entrada del módulo *ZigBee* no debe exceder de los 3.3 voltios es imprescindible la implementación de un divisor de voltaje. El arreglo de resistencias que permite obtener el divisor de voltaje requerido es con una R2 = 10 KΩ y R3 = 20 KΩ.

La conexión efectuada entre el Arduino Mega2560 y el módulo *ZigBee* será la misma que se utilizará para los demás nodos del sistema. La principal diferencia entre estos será la configuración propia de cada módulo *ZigBee* y el microcontrolador.

Adicionalmente en el nodo central se tiene un botón de reinicio del sistema, una interfaz de conexión RJ45 al modem/switch y 8 led's que permiten confirmar los siguientes estados del sistema:

- El primer led indica el inicio del sistema
- El segundo y tercer led indican la compilación del programa en el Arduino Mega
- El cuarto led indica el mal funcionamiento del módulo ZigBee
- El quinto led indica el mal funcionamiento del Arduino mega
- El sexto led indica que la red punto – punto está trabajando
- El séptimo led indica que la red punto – multipunto está trabajando
- El octavo led indica un error en la reconstrucción de la imagen.

Cada led tiene una resistencia de 470 Ω. El valor de la resistencia se obtuvo aplicando la ley de Ohm utilizando los valores de resistencia y voltaje del led ver Anexo C. Se conoce que la corriente máxima que soporta el LED es de 20 mA y su voltaje es de 4 voltios. En la Figura 2.12, se puede observar el diagrama de conexión de estos elementos.

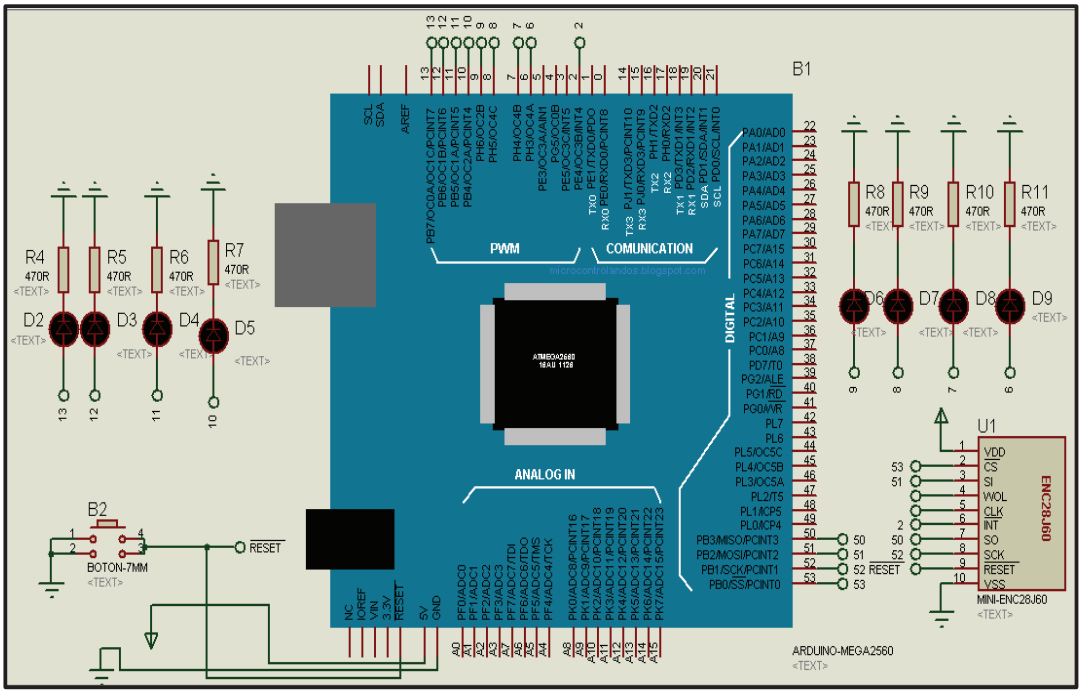


Figura 2.12 Diagrama de conexión del nodo central.

En la Figura 2.13 se observa el impreso de la placa para el nodo central.

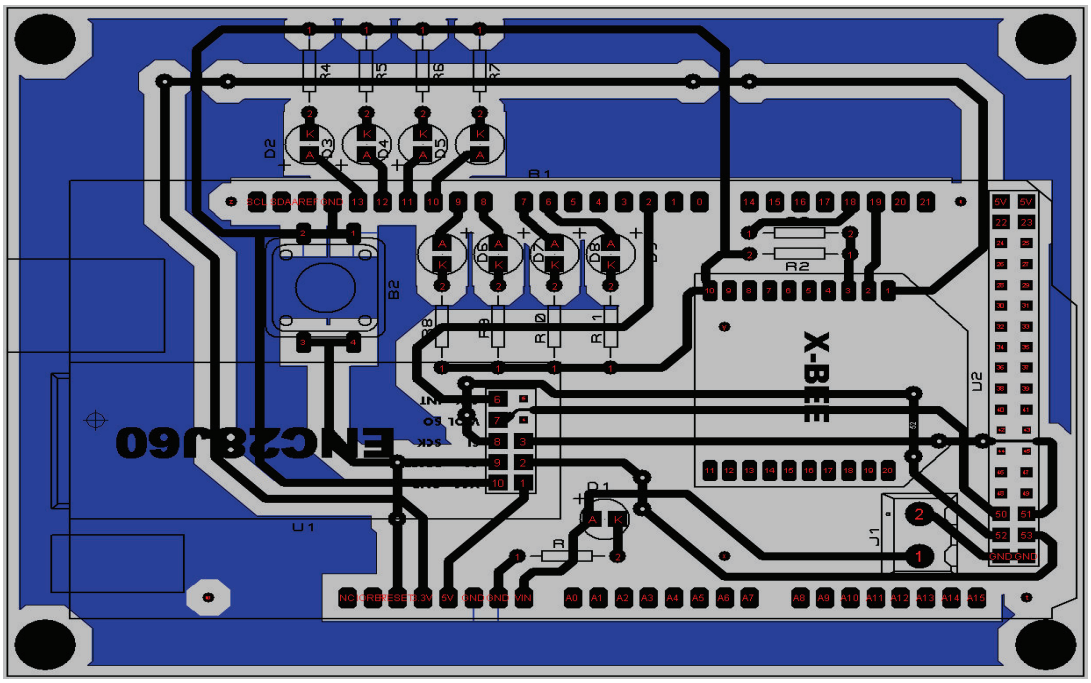


Figura 2.13 Impreso de la placa del nodo central.

2.5.1 DESARROLLO DEL PROGRAMA PARA EL NODO CENTRAL

El Arduino Mega está encargado de receptar la información proveniente de la aplicación Android, procesar la misma y transmitirla a través del módulo XBee coordinador a los demás nodos que conforman el sistema. En la Figura 2.14, se observa el diagrama del proceso que debe realizar el Arduino Mega para alcanzar la comunicación.

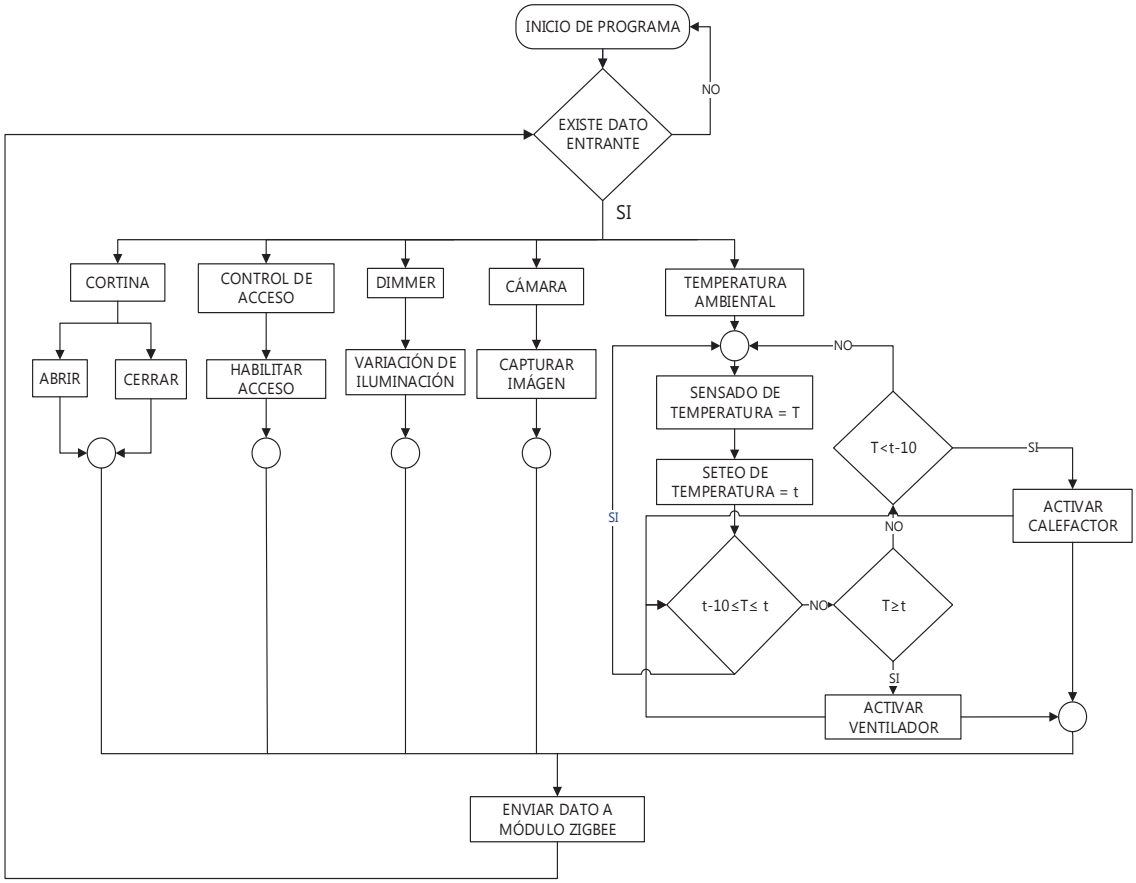


Figura 2.14 Diagrama de flujo del nodo central.

Mediante lenguaje estructurado se detalla a continuación el algoritmo que sirvió como base para programar el nodo central. Se considera para la temperatura ambiental una región de trabajo que está definida entre el valor seteado por el usuario (t) menos 10°C que genera el programa.

Ingreso de datos.
Determinar el estado actual del sistema
Recepta datos de la aplicación Android
Fin tarea

Controlar movimiento de cortina.
Desplegar submenú de control.
Enviar comando para activar la cortina
Ir a control del motor
Fin tarea

Controlar acceso.

Desplegar submenú de control.
Enviar comando para activar cerradura eléctrica.
Ir a control de acceso

Fin tarea

Controlar dimmer.

Desplegar submenú de control.
Enviar comando para activar el dimmer
Ir a control de iluminación

Fin tarea

Controlar cámara.

Desplegar submenú de control.
Enviar comando para activar la cámara
Ir a control de cámara

Fin tarea

Controlar temperatura ambiental.

Sensado de temperatura.
Enviar comando para setear temperatura
Ir a control de temperatura ambiental

Fin tarea

Controlar Arduino Mega.

Receptar datos de la aplicación Android
Procesar datos
Transmitir datos al Módulo XBee coordinador
Receptar datos

Fin tarea

Módulo XBee.

Esperar datos del Arduino Mega
Transmitir información a los módulos XBee esclavos
Receptar información a los módulos XBee esclavos

Fin de ciclo

2.6 DISEÑO DEL CIRCUITO PARA EL CONTROL DE TEMPERATURA AMBIENTAL

Para el diseño de esta parte del sistema se seleccionó un sensor de temperatura, un acondicionador, un microcontrolador, y el módulo *ZigBee*. El siguiente diagrama muestra la estructura básica del nodo, ver Figura 2.15.

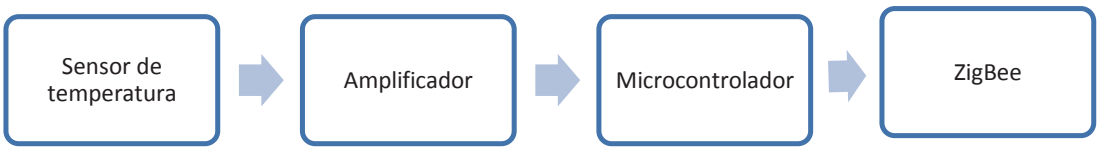


Figura 2.15 Diagrama de bloques del control de temperatura ambiental.

El sensor LM35 To-220, es un sensor de salida lineal[9] de temperatura con una precisión calibrada de 1°C, y una sensibilidad de 10.0mV/°C, funciona en un rango de alimentación comprendido entre 4 y 30 voltios, sensando cambios de temperatura entre -55°C a +150°C.

Debido a que el voltaje de salida entregado por el sensor de temperatura es muy pequeño (1°C = 10mV), se debe diseñar un amplificador no inversor que provea un adecuado voltaje al microcontrolador, de manera que este pueda interpretar y procesar correctamente la información.

El amplificador operacional utilizado para este propósito fue el LM358, mismo que presenta las siguientes características:

- Alta ganancia en DC: 100 dB.
- Alto rango de alimentación.

[9] Lineal: Un sistema es lineal si satisface las propiedades de proporcionalidad y aditividad. Es invariante en el tiempo cuando ante una entrada determinada, el sistema responde de la misma forma independientemente del instante de tiempo en el que se aplique esa entrada.

- Alimentación simple: entre 3V y 32V.
- Bajo offset de voltaje de entrada (2mV).
- Bajo consumo de energía.

En la Figura 2.16, se presenta el esquema realizado para amplificar el voltaje entregado por el sensor LM35, necesario para la interpretación de la temperatura.

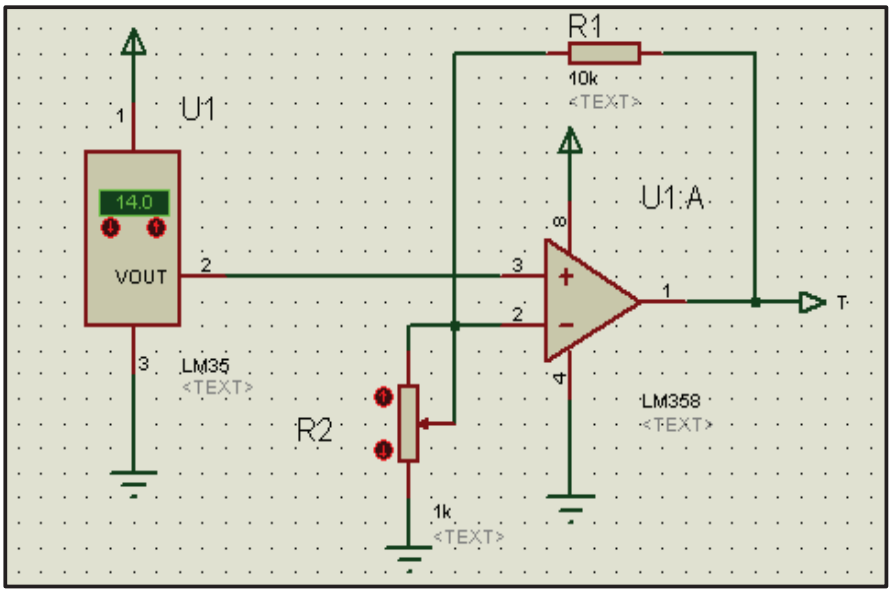


Figura 2.16 Diagrama de conexión del sensor LM35 y el amplificador LM358.

Para graduar la ganancia se utilizó un potenciómetro lineal de 1KΩ, el cual permite variar el valor de la resistencia hasta ajustarlo al valor requerido de 6,66 de ganancia con un R1 = 10 KΩ.

Los datos analógicos obtenidos por el sensor de temperatura y amplificados por el LM358 son convertidos en digitales por el microcontrolador utilizando su módulo ADC y una adecuada programación en lenguaje Arduino. El microcontrolador activará dos de sus salidas que controlan dos relés, los mismos que se encargaran de encender un ventilador y un calefactor dependiendo del seteo de témperatura que el usuario desee.

Adicionalmente en el nodo control de temperatura ambiental existe un botón de reinicio del sistema, un botón para el switcheo a control manual del sistema, un

botón para la activación de ventilador y otro para la activación del calefactor. También se cuenta con 3 led's que permiten ver los siguientes estados del sistema:

- El primer led indica el funcionamiento manual o automático del sistema
- El segundo led indica el funcionamiento del ventilador
- El tercer led indica el funcionamiento del calefactor

Cada led tiene una resistencia de 330 KΩ y la conexión del módulo XBee. En la Figura 2.17, se puede observar el diagrama de conexión de estos elementos.

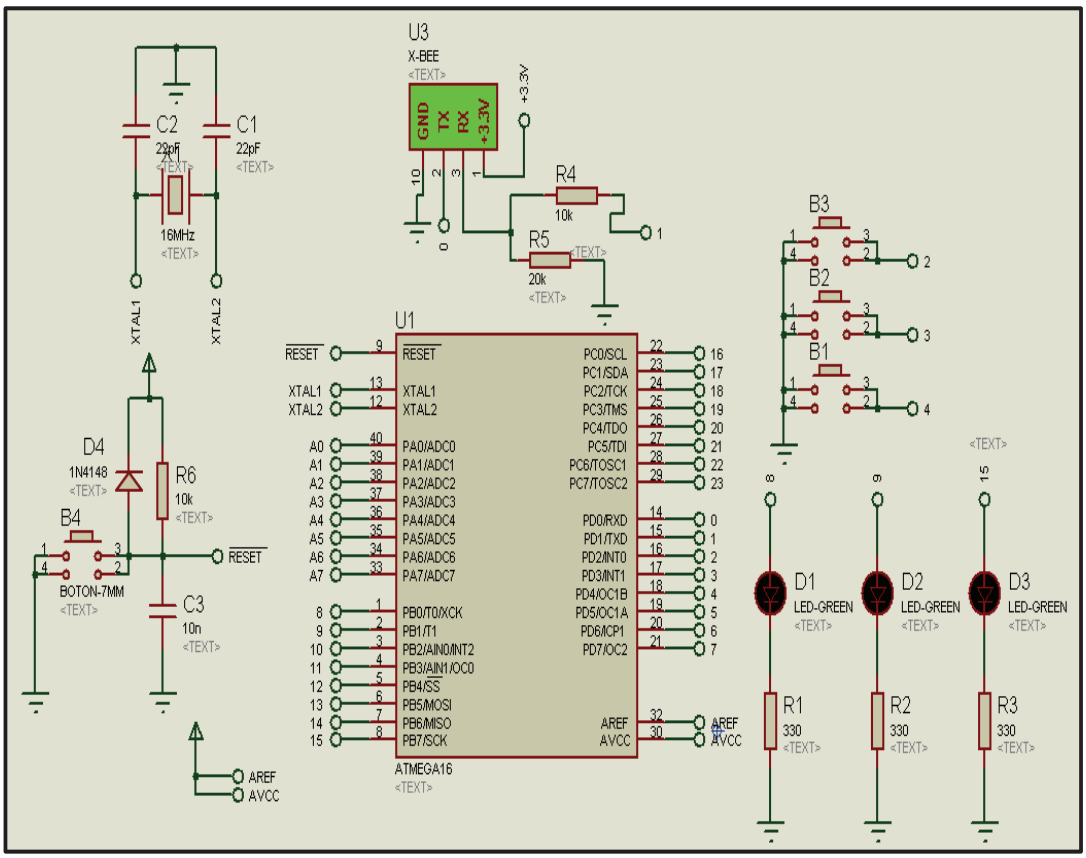


Figura 2.17 Diagrama de conexión del control de temperatura ambiental.

En la Figura 2.18, se observa el impreso del control de temperatura ambiental.

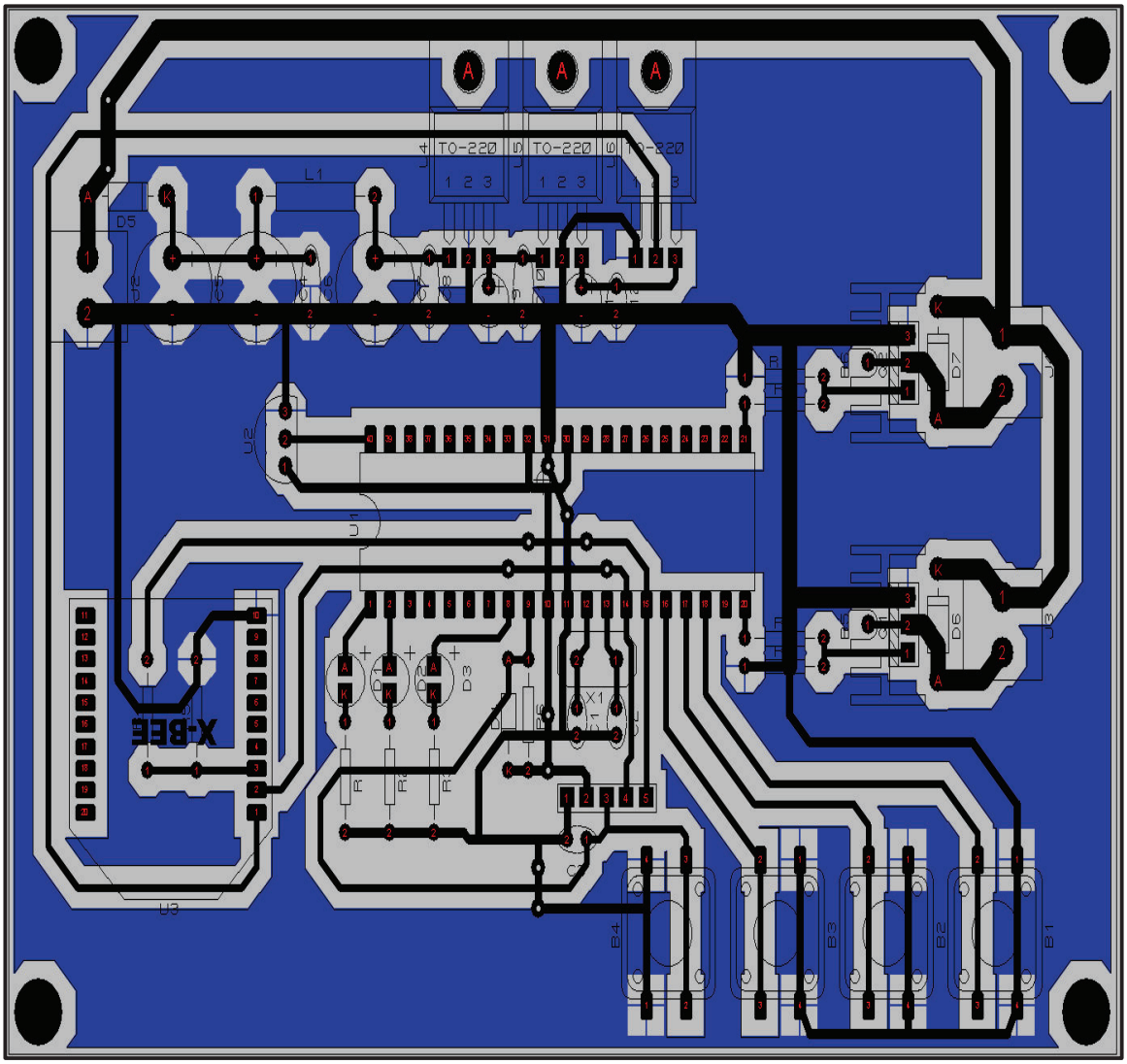


Figura 2.18 Impreso del control de temperatura ambiental.

2.6.1 DESARROLLO DEL PROGRAMA PARA CONTROL DE TEMPERATURA AMBIENTAL

El control de la temperatura permite activar o desactivar un ventilador o calefactor según el usuario lo requiera. A continuación se muestra el diagrama de flujo utilizado para desarrollar el programa del nodo de temperatura ambiental, ver Figura 2.19.

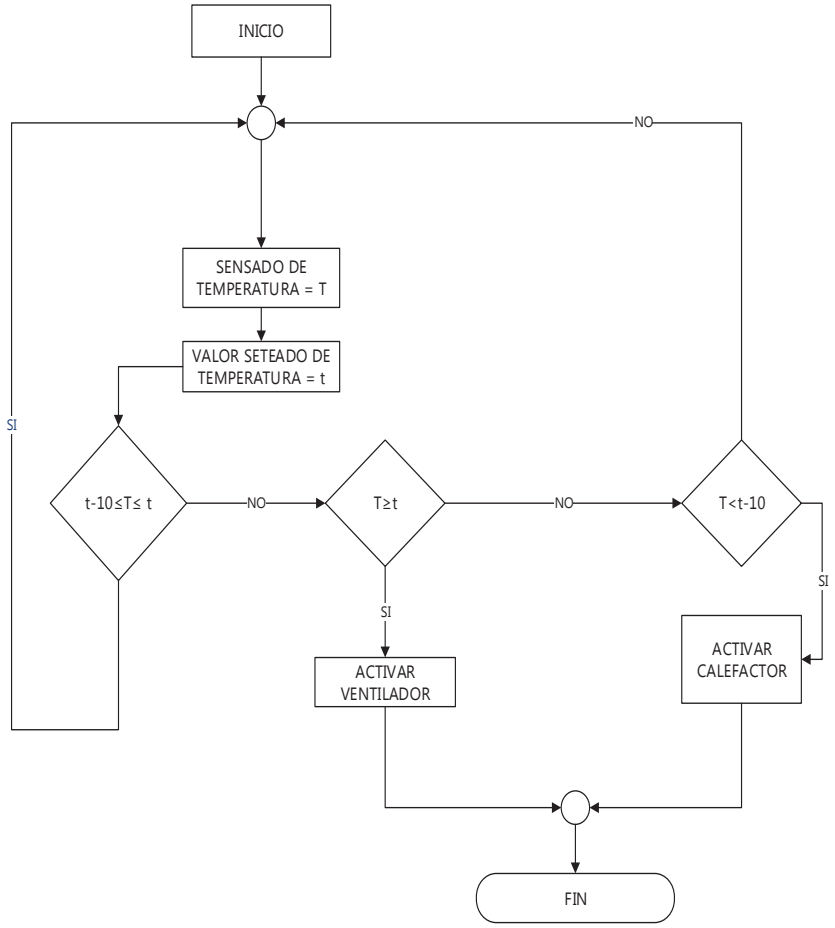


Figura 2.19 Diagrama de flujo del control de temperatura ambiental.

Inicialización.

Configurar el puerto UART
Configurar el conversor A/D
Definir y dimensionar variables
Configurar puertos

Fin tarea

Sensado de Temperatura.

Esperar peticiones del nodo central
Obtener valores a través del canal analógico del microcontrolador.
Enviar información al nodo central

Fin tarea

Seteo de Temperatura.

Identificar valor seteado de temperatura por el usuario
Temperatura mayor o igual al valor seteado activación del ventilador.
Temperatura seteada menos 10°C activación del calefactor.

Fin tarea

2.7 DISEÑO DEL CIRCUITO DEL CONTROL DE LA CORTINA MOTORIZADA

Este nodo está conformado por un módulo *ZigBee*, un microcontrolador, un driver y un motor. El siguiente diagrama muestra la estructura básica del nodo, ver Figura 2.20.

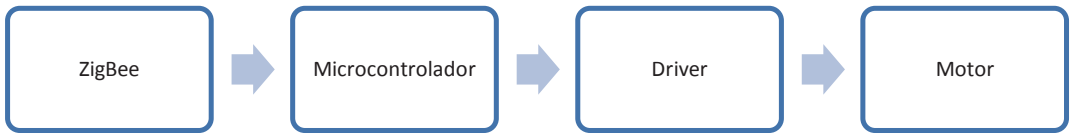


Figura 2.20 Diagrama de bloques del nodo cortina.

En este caso el microcontrolador procesará la información proveniente del módulo XBee en forma serial o, si es el caso, permitirá el movimiento del motor a través de los puertos que permiten el control manual.

El driver por su parte permite la manipulación del motor DC denominado módulo 2 – *Channel Relay* diseñado para integrarse con 2 relés para controlar motores DC, presenta señales de control a nivel TTL con una carga nominal de 7A / 240VAC, 10A / 125VAC y 10A / 28VDC.

El motor que se seleccionó para el control de la cortina es un motor de corriente continua, el mismo que está compuesto de un estator y un rotor, el estator crea el campo magnético fijo, el rotor, también conocido como armadura, lleva las bobinas cuyo campo crea, junto al del estator, el par de fuerzas que permiten el giro. También cuenta con 2 led´s que indican si la cortina se cierra o se abre. En la Figura 2.21**,** se observa el diagrama de conexión del nodo cortina motorizada y en la Figura 2.22 se observa el impreso del nodo.

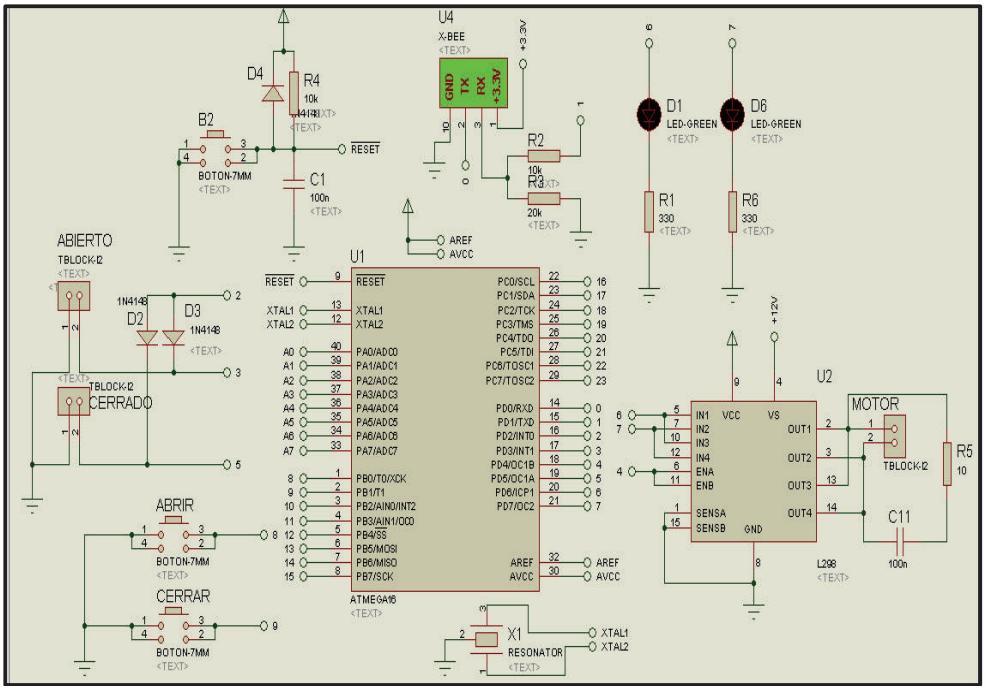


Figura 2.21 Diagrama de conexión del nodo cortina.

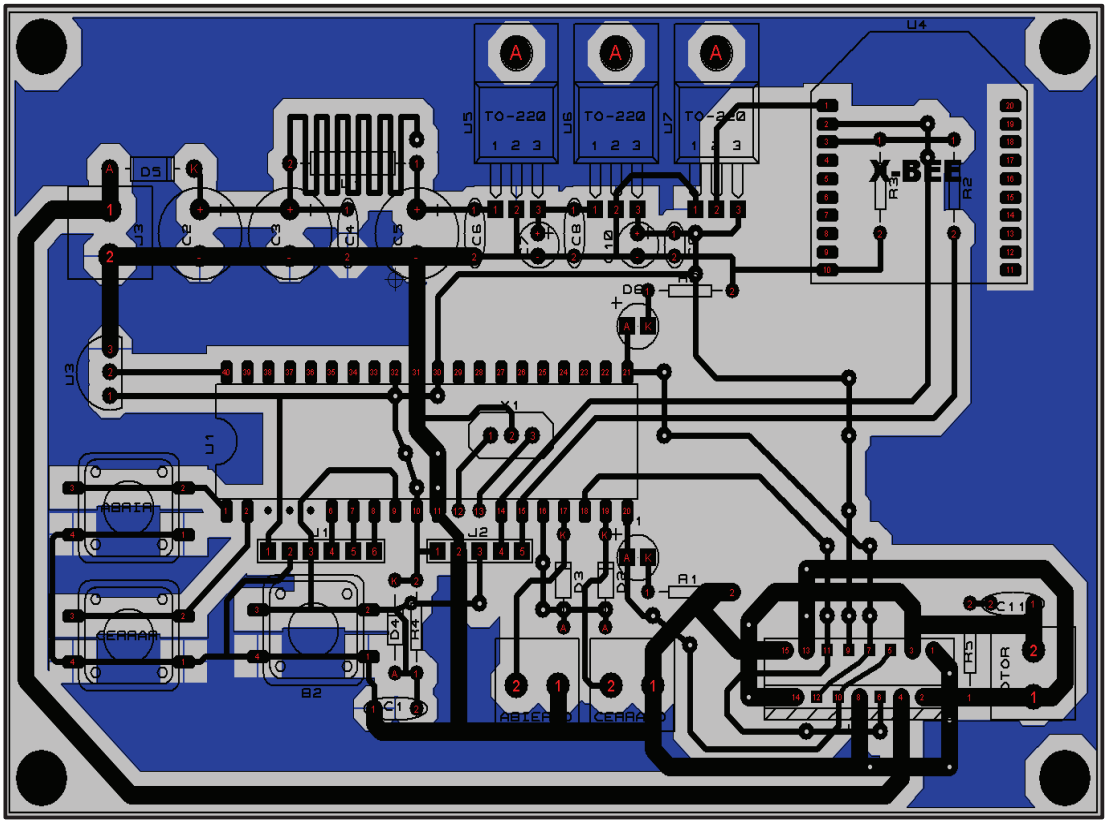


Figura 2.22 Impreso del circuito del nodo cortina.

2.7.1 DESARROLLO DEL PROGRAMA PARA EL CONTROL DE LA CORTINA MOTORIZADA.

A continuación se muestra el diagrama de flujo utilizado como base para programar el microcontrolador de la cortina motorizada, ver Figura 2.23.

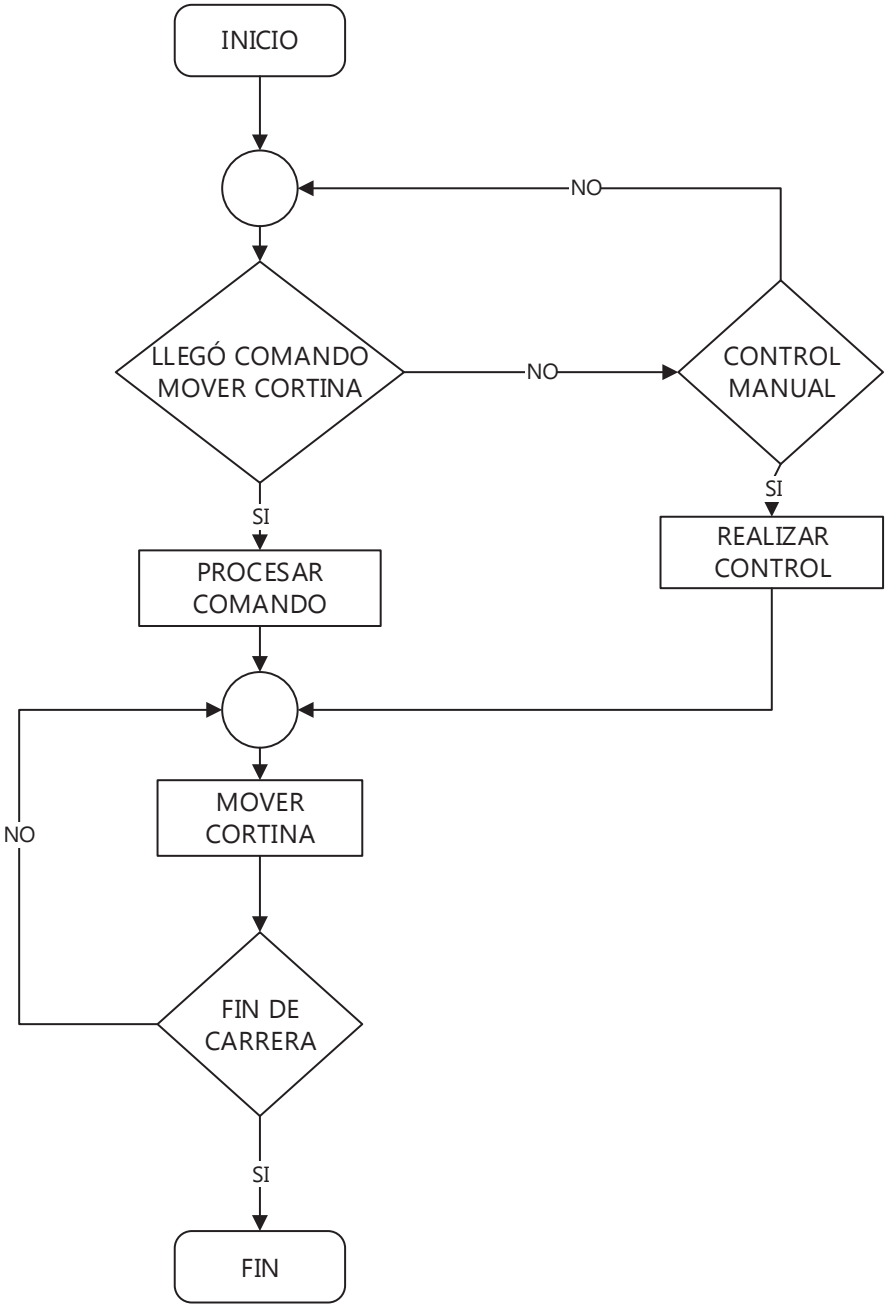


Figura 2.23 Diagrama de flujo del control de la cortina motorizada.

Inicialización.
Configurar el puerto UART
Configurar los temporizadores
Definir y dimensionar variables
Configurar puertos
Fin tarea

Procesar información.
Adquirir dato a través del puerto UART
Procesar dato
Identificar tipo de movimiento.
Si se requiere realizar control manual
Caso contrario procesar comando
Fin tarea

Control de movimiento de cortina
Mover cortina según la petición
Fin tarea

2.8 DISEÑO DEL CIRCUITO PARA CONTROL DE ACCESO.

Para el diseño de este nodo se requiere de una cerradura eléctrica, un microcontrolador, un teclado, alarma y un módulo *ZigBee*. En el siguiente diagrama se muestra la estructura adoptada, ver Figura 2.24.

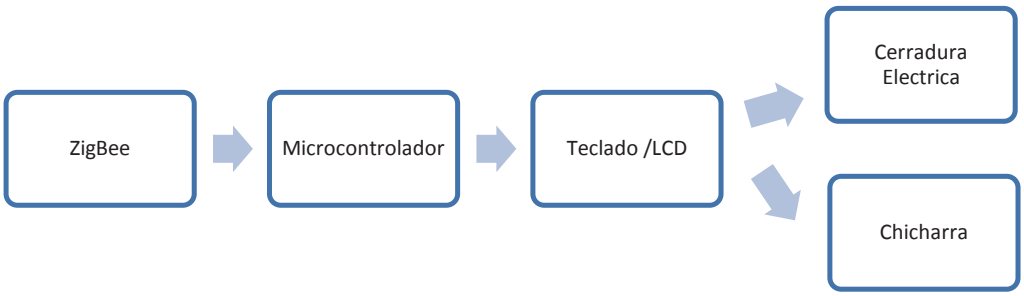


Figura 2.24 Diagrama de bloques del nodo control de acceso.

La etapa del microcontrolador es la encargada de recibir la señal del módulo XBee mediante comunicación serial, y generar una señal de voltaje durante algunos milisegundos a través de la salida seleccionada, esta señal activa un relé

de 5 voltios, que a su vez permite la conexión de la bobina de la cerradura a un transformador de 12 voltios con lo que se activa el mecanismo que abre el cerrojo de la puerta.

Adicionalmente este nodo tiene incorporado un teclado y una pantalla LCD que permiten al usuario digitar una clave de acceso de forma manual, si la clave es incorrecta se activa de manera intermitente una chicharra. También tiene 3 led´s que indican el correcto funcionamiento del pulso de apertura de la cerradura, la activación de la chicharra y el correcto ingreso de la clave de acceso. En la Figura 2.25, se observa el diagrama de conexión usado para el nodo de acceso y en la Figura 2.26, se observa el impreso del circuito.

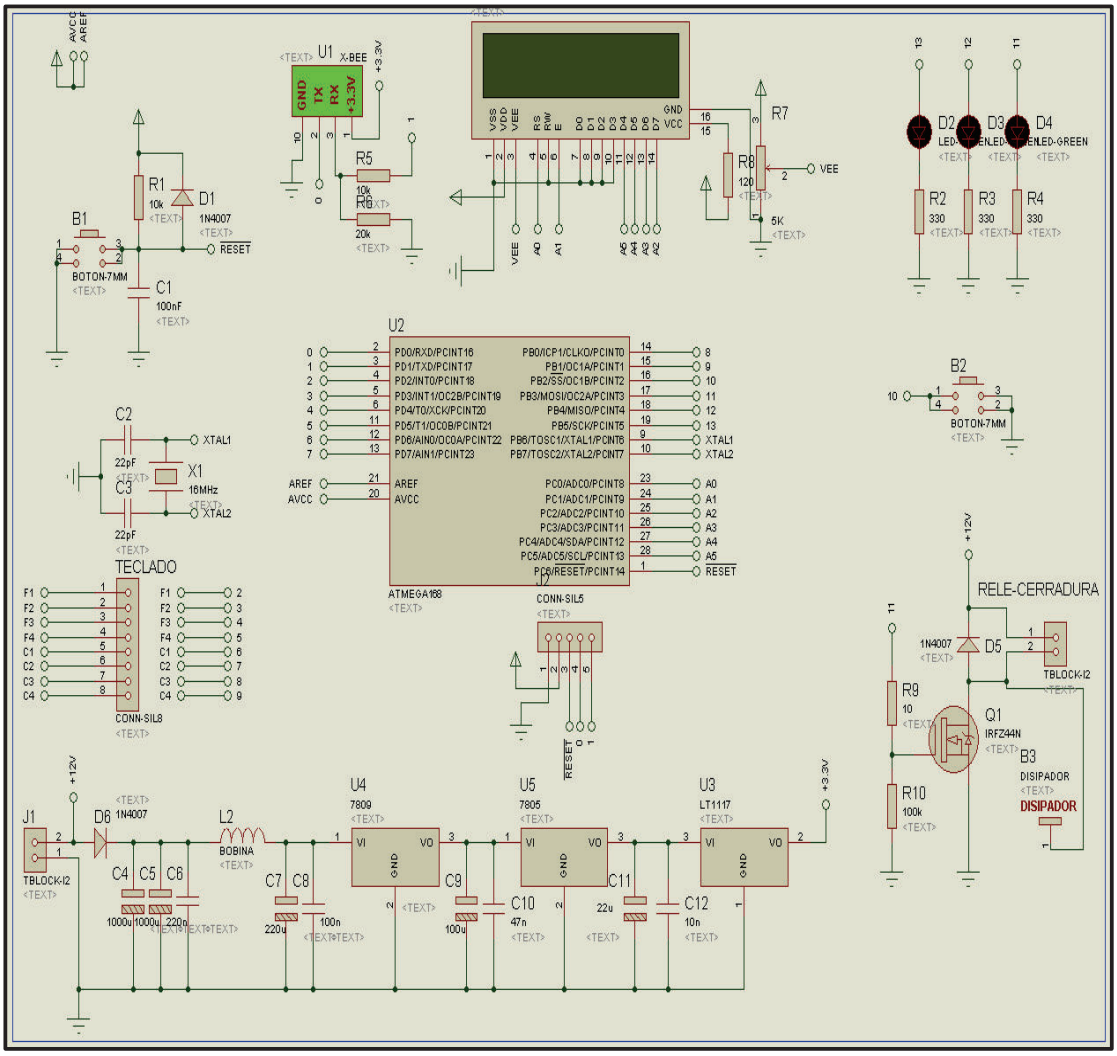


Figura 2.25 Diagrama de conexión del nodo control de acceso.

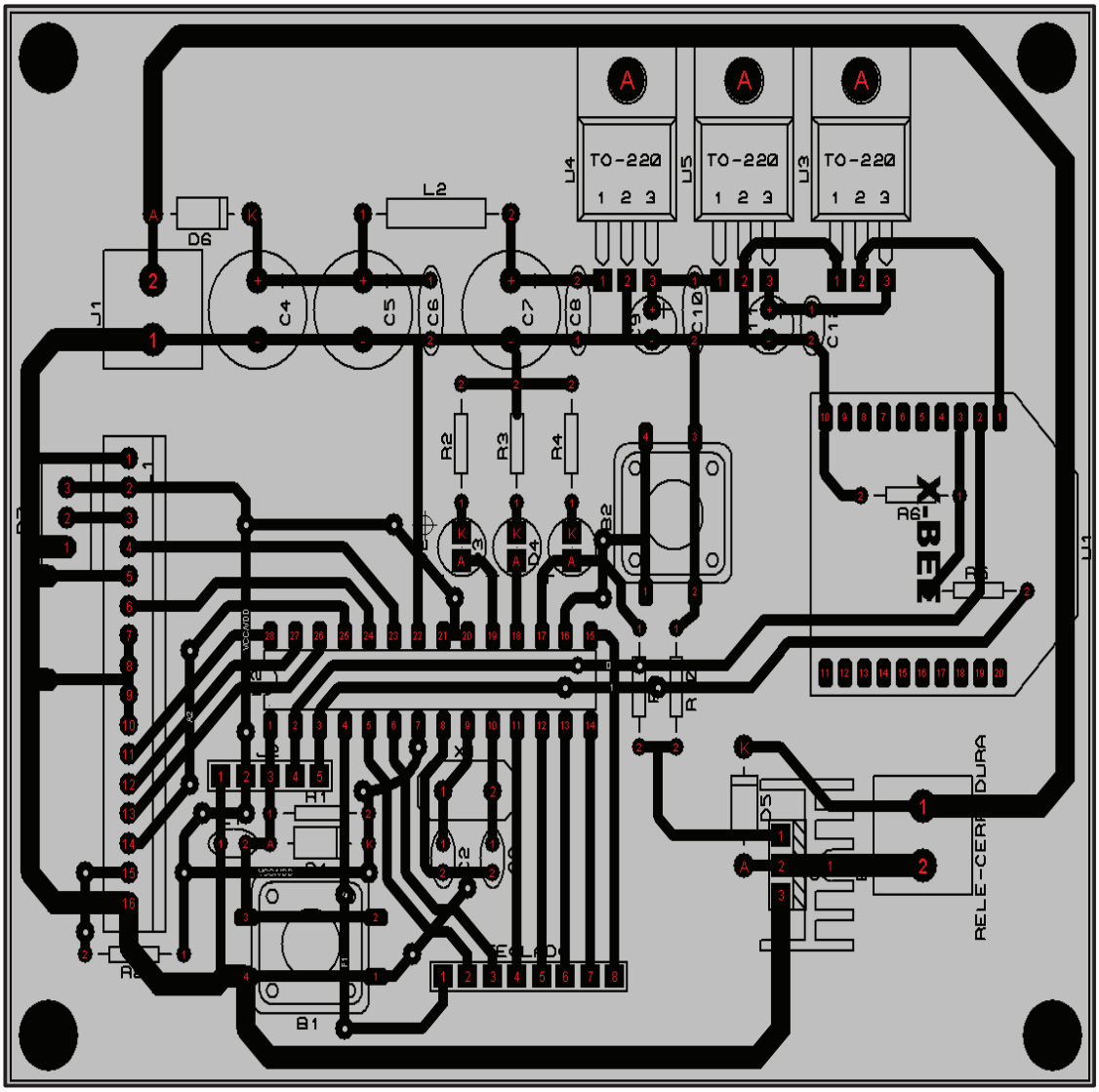


Figura 2.26 Impreso del circuito del nodo control de acceso.

2.8.1 DESARROLLO DEL PROGRAMA PARA EL CONTROL DE ACCESO.

A continuación se muestra el diagrama de flujo del control de acceso, ver figura 2.27.

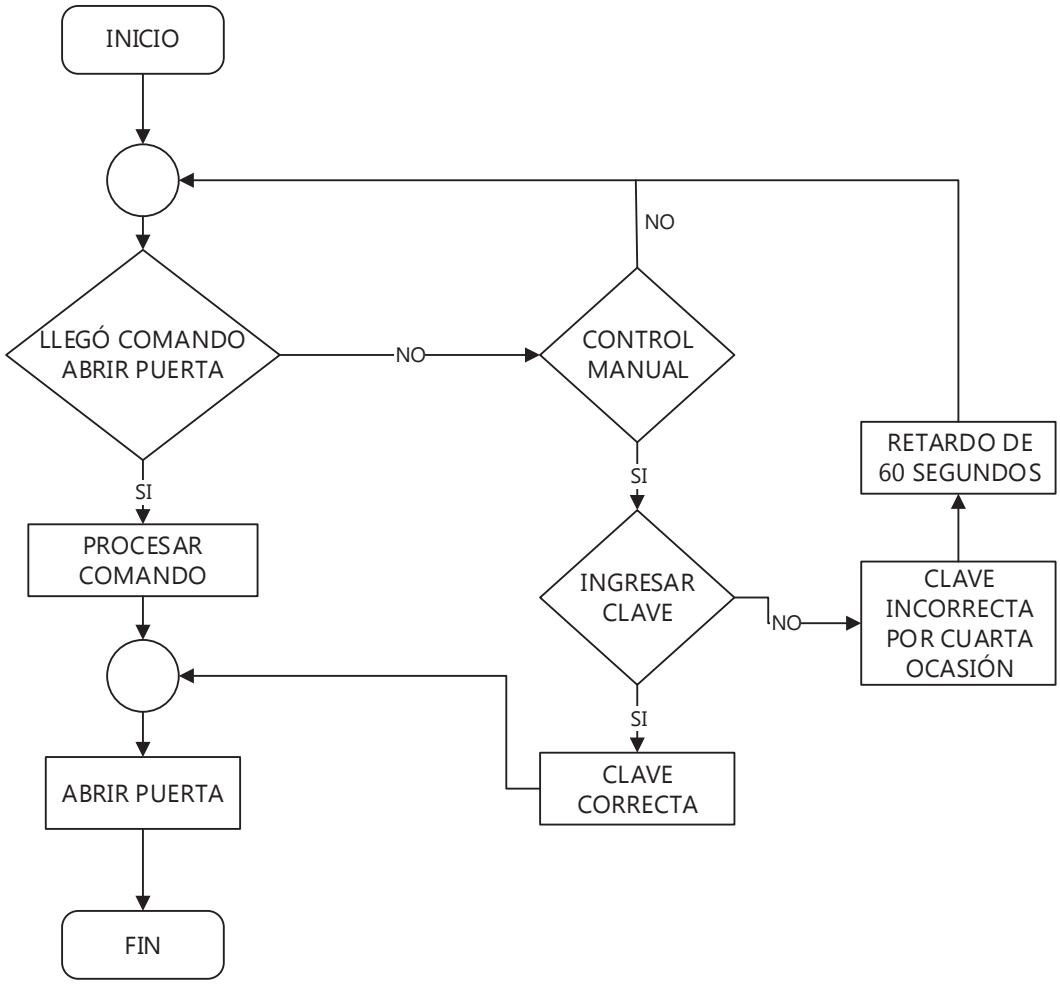


Figura 2.27 Diagrama de flujo del control de acceso.

Inicialización.

Configurar el puerto UART
Configurar los temporizadores
Definir y dimensionar variables
Configurar puertos de entrada y salida

Fin tarea

Procesar información.

Adquirir dato a través del puerto UART
Procesar dato
Ingresar clave
Caso contrario procesar comando

Fin tarea

Abrir puerta

Activar cerradura eléctrica

Fin tarea

2.9 DISEÑO DEL CIRCUITO DIMMER

Para el diseño de este nodo se requiere de un circuito detector de cruce por cero, un microcontrolador, un circuito de potencia, una lámpara incandescente y un módulo *ZigBee*. En el siguiente diagrama se muestra la estructura que sigue este nodo ver figura 2.28.

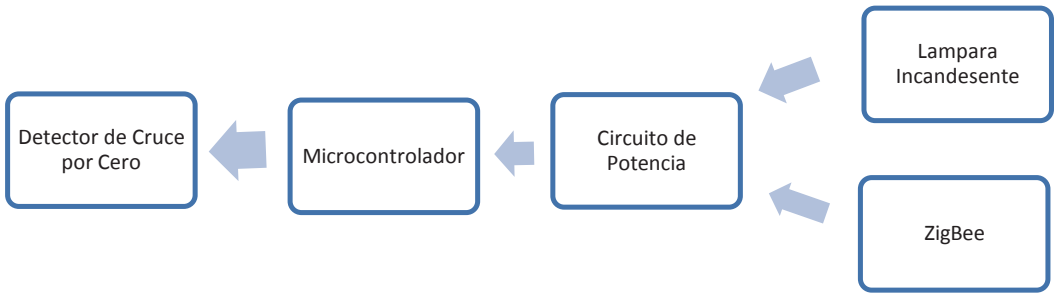


Figura 2.28 Diagrama de bloques del nodo *dimmer.*

Se generan los datos de entrada para el d*immer*, los mismos que ingresan al microcontrolador en forma de pulsos. El circuito detector de cruce por cero genera los pulsos cuando la señal AC de entrada cambia de polaridad.

El elemento que permite generar esto es el optoacoplador, este dispositivo se puede utilizar:

- A modo de interfaz entre dos circuitos, permitiendo un control de potencia.
- Como relé.
- En un sistema de recepción de telefonía.
- Aislar circuitos como la corriente alterna suministrada por la red de la tensión continua que ingresará al microcontrolador.

En la Figura 2.29, se observa el circuito detector de cruce por cero utilizado para el diseño del nodo *Dimmer.*

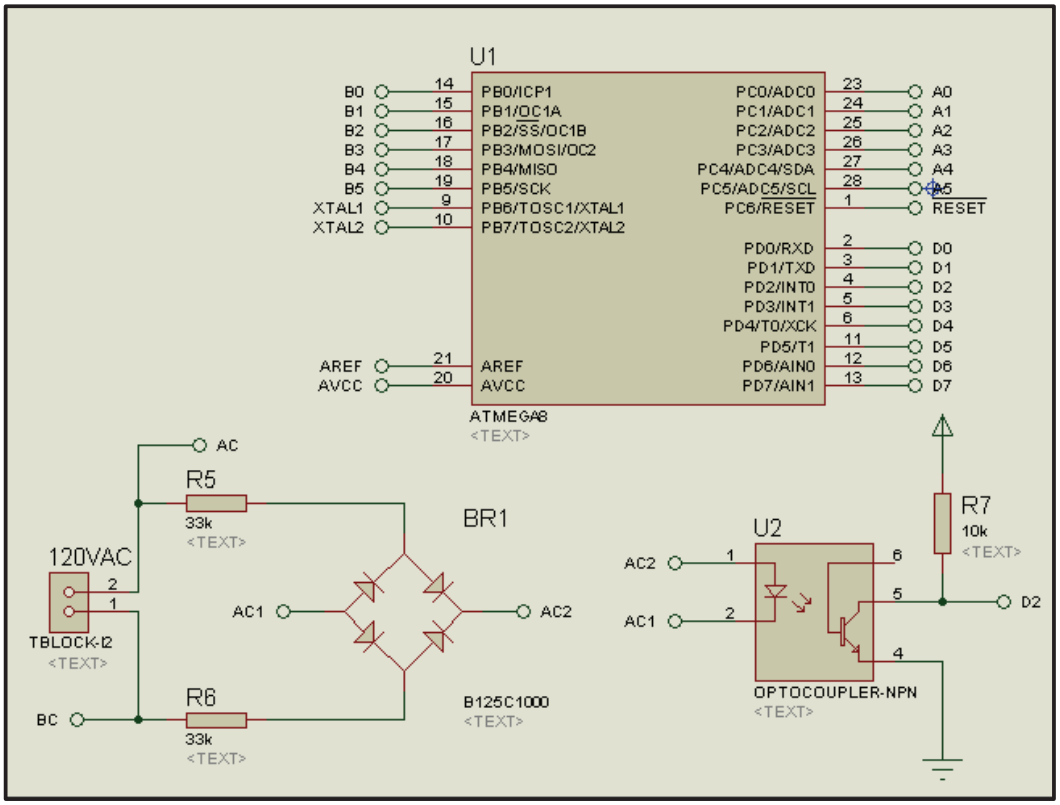


Figura 2.29 Circuito detector de cruce por cero.

El elemento fundamental en esta etapa es el optoacoplador 4N35, el cual está formado por un LED y un fototransistor. Se coloca dos resistencias de 33KΩ para disminuir la potencia que ingresa al optoacoplador, pues este es un dispositivo que trabaja en baja potencia. La tensión rectificada y las resistencias R5 y R6 establecen una corriente en el LED emisor al momento que el voltaje ánodo cátodo del led es superado. Si dicha corriente proporciona un nivel de luz adecuado, al incidir sobre el fototransistor lo saturará, generando una corriente en R7. De este modo la tensión de salida será igual a cero cuando el Voltaje de entrada es menor que el voltaje del led e iguala a 5V (fuente DC que alimenta al circuito) el momento que sea mayor.

Dado que la corriente de entrada que soporta el microprocesador es de 25mA y y este es alimentado con un voltaje de $5V_{DC}$, se calculó una resistencia (R7) de 10KΩ a la salida del optoacoplador.

Para el diseño del circuito de potencia se recurrió a la utilización del TRIAC BT136 debido a que puede soportar una carga de hasta 5A; el TRIAC es un dispositivo semiconductor que se utiliza en corriente alterna, con la particularidad de que conduce en ambos sentidos y puede ser bloqueado por inversión de la tensión o al disminuir la corriente por debajo del valor de mantenimiento, lo cual es ideal para el ensamblaje de esta etapa.

La etapa de aislamiento está formada por un optoacoplador MOC3011, el cual cuenta con un circuito detector de cruce por cero y permite que el TRIAC BT136 entre en funcionamiento (se dispare correctamente), ver Figura 2.30.

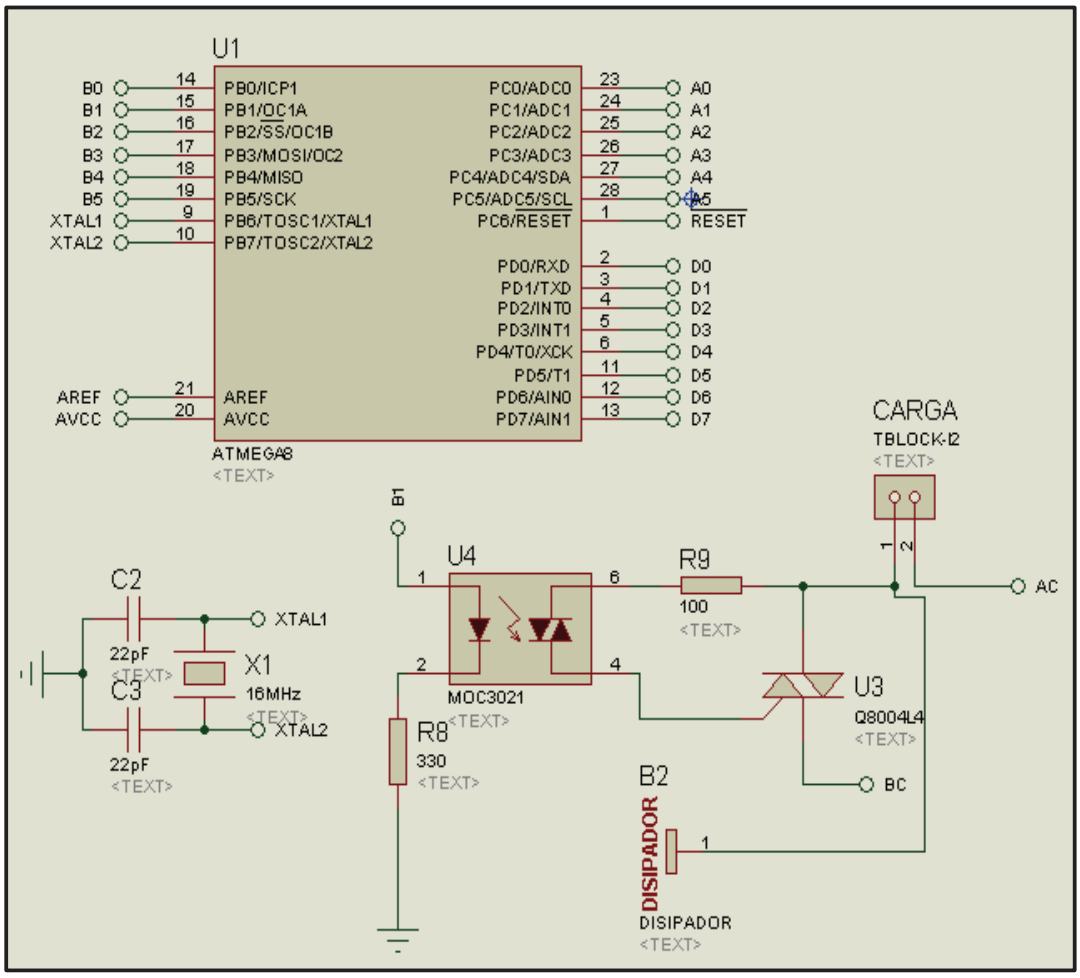


Figura 2.30 Circuito de Potencia.

El microcontrolador recibirá la señal de cruce por cero enviada por el opto acoplador a través del pin de interrupción externa, la procesará a través de la programación y generará una señal de salida variable que permitirá activar al TRIAC acorde con las órdenes recibidas. Este nodo también tiene 2 led´s que indica el correcto funcionamiento del mismo y su alimentación. El usuario podrá aumentar o disminuir el voltaje de salida manualmente o a través de la aplicación Android; el resultado de este proceso se podrá apreciar diferenciando el aumento o disminución de la intensidad luminosa en la lámpara incandescente. En la Figura 2.31, se observa el circuito impreso para el control del *dimmer*.

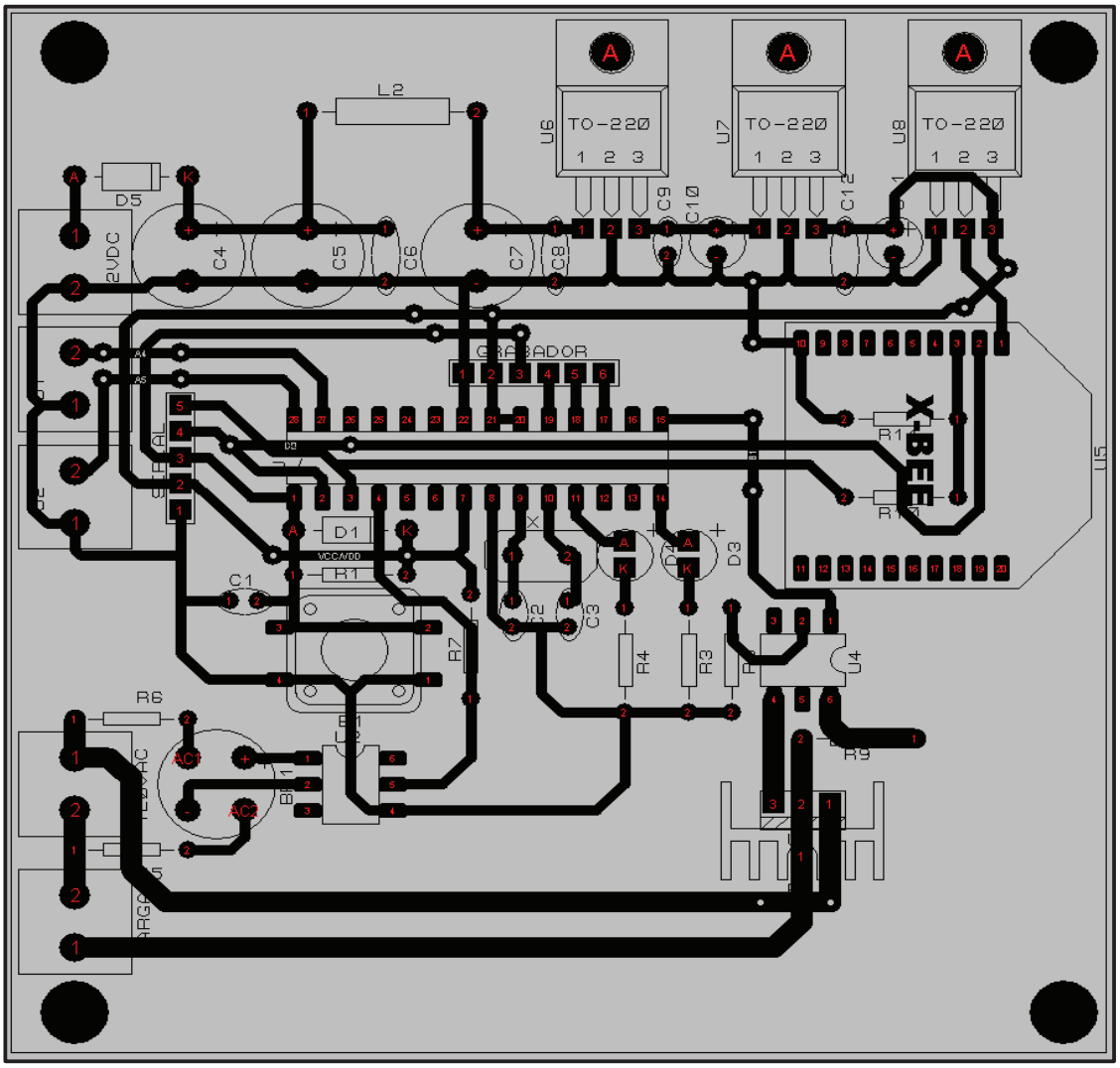


Figura 2.31 Impreso del circuito del nodo *dimmer.*

2.9.1 DESARROLLO DEL PROGRAMA PARA EL CONTROL DEL DIMMER

A continuación se muestra el diagrama de flujo del nodo dimmer, ver Figura 2.32.

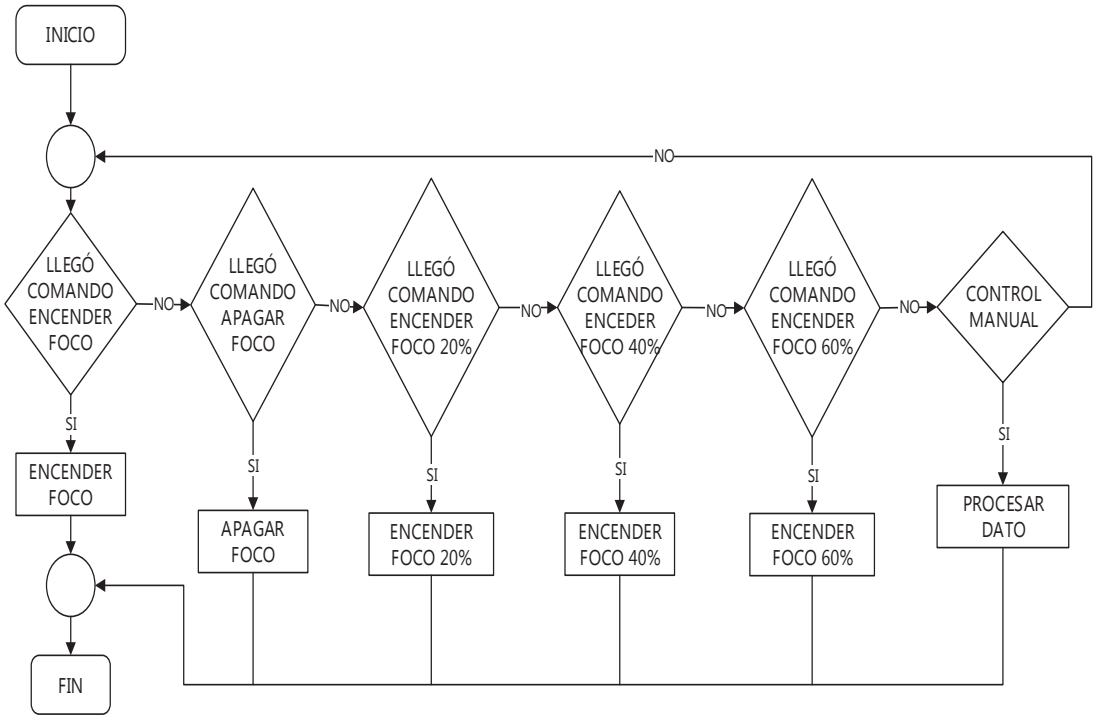


Figura 2.32 Diagrama de flujo del control del *dimmer.*

Inicialización.

Configurar la señal PWM
Configurar los temporizadores
Definir y dimensionar variables
Configurar puertos de entrada y salida

Fin tarea

Procesar Información.

Adquirir dato a través del puerto USART
Procesar dato
Identificar petición
Si se realiza control manual realizar control
Caso contrario procesar comando

Fin tarea

Control de iluminación.

Variar iluminación según petición

Fin tarea

2.10 DISEÑO DEL CIRCUITO DE LA CÁMARA

Para el diseño de este nodo se requiere de una cámara y un módulo *ZigBee*. En el siguiente diagrama se muestra la estructura que sigue este nodo, ver Figura 2.33.

Figura 2.33 Diagrama de bloques del nodo cámara.

La cámara que se usó para este nodo es la LinkSprite JPEG SERIAL TTL LS-Y201 que tiene la cualidad de obtener una imagen digital a través de un puerto serial. El sensor de la cámara posee una gran sensibilidad a la luz lo que le permite captar una imagen clara y con muy poca luz, lo que es óptimo para zonas de muy baja iluminación. En la Figura 2.34, se aprecia la imagen de la cámara mencionada.

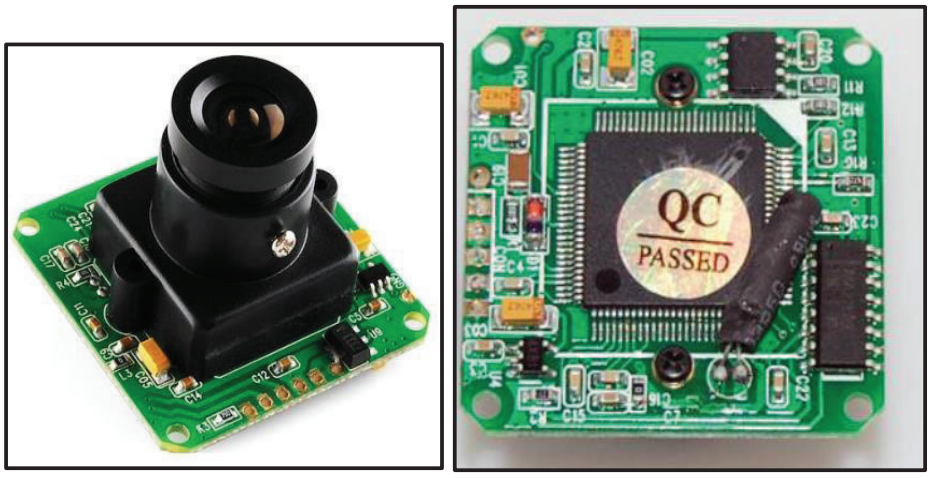


Figura 2.34 Cámara JPEG SERIAL TTL LS-Y201 [20]

Las características que presenta esta cámara son las siguientes:

- Resolución de VGA/QVGA/160x120.
- Alimentación de 3.3 V o 5 V.

- Consumo de corriente de 80 – 100mA.
- Velocidad de transmisión por defecto del puerto serie es de 38400 bps.
- Formato JPEG por puerto serial para la captura de imágenes.

Las aplicaciones que presenta esta cámara son las siguientes:

- Diferentes sistemas de captura de imagen.
- Monitoreo ambiental.
- Monitoreo de la industria.
- Equipamiento médico.
- Videollamada.
- Seguridad.

La cámara posee 5 pines los mismos que se describen en la **Tabla 2.6.**

Tabla 2.6 Pines de la cámara JPEG SERIAL TTL LS-Y201.

PIN	DESCRIPCIÓN
+5V	Alimentación positiva
GND	Tierra
TXD (OUT)	Transmisión de datos de nivel TTL
RXD (IN)	Recepción de datos de nivel TTL
TV	Salida de señal de video compuesta

Para poder llevar a cabo la comunicación serial se requiere de un protocolo de comunicación que indique a la cámara que hacer, este protocolo se caracteriza por el envío y recepción de direcciones en hexadecimal.

Para poder controlar el módulo de la cámara se requiere enviar los siguientes arreglos hexadecimales:

- Reiniciar el módulo de la cámara:

Comando (HEX)	Retorno (HEX)
56 00 26 00	76 00 26 00

- Captura de una imagen:

Comando (HEX)	Retorno (HEX)
56 00 36 01 00	76 00 36 00 00

- Tamaño de la imagen:

Comando (HEX)	Retorno (HEX)
56 00 34 01 00	76 00 34 00 04 00 00 XH XL

Donde XH y XL es la longitud del archivo JPEG.

- Leer el contenido de la imagen:

Comando (HEX)

56 00 32 00 0A 0C 00 00 MH ML 00 00 KH KL XX XX

Retorno (HEX)

76 00 32 00 00 (Intervalo de tiempo) FF D8... .... (Intervalo de tiempo) 76 00 32 00 00

Donde un archivo JPEG comienza con FF D8 y D9 y termina con FF por lo cual para leer el archivo JPEG, siempre se comienza con la dirección 00 00.

(Intervalo de tiempo) = XX XX * 0.01ms se recomienda estar 00 0A

00 00 MH ML: Dirección inicial

00 00 KH KL: Duración de archivo JPEG

- Detener captura de imagen:

Comando (HEX)	Retorno (HEX)
56 00 36 01 03	76 00 36 00 00

- Establecer el tamaño de la imagen de salida:

Comando (HEX)	Retorno (HEX)
56 00 31 05 04 01 00 19 XX	76 00 31 00 00

  Donde:
  XX: Para una imagen de 640x480 su valor es 0x00
  XX: Para una imagen de 320x240 su valor es 0x11
  XX: Para una imagen de 180x120 su valor es 0x22

- Estado de la cámara (encendida/apagada):

Comando (HEX)	Retorno (HEX)
56 00 3E 03 00 01 XX	76 00 3E 00 00

  Donde:
  XX: Para encender la cámara su valor es 0x00
  XX: Para apagar la cámara su valor es 0x11

- Velocidad de transmisión :

Comando (HEX)	Retorno (HEX)
56 00 24 03 01 XX XX	76 00 24 00 00

  Donde:
  XX XX: Para una velocidad de 9600 su valor es 0xAE 0xC8
  XX XX: Para una velocidad de 19200 su valor es 0x56 0xE4
  XX XX: Para una velocidad de 38400 su valor es 0x2A 0xF2
  XX XX: Para una velocidad de 57600 su valor es 0x1C 0x4C

Desde el nodo central del sistema domótico se administra la lógica de la cámara con la lectura y detección de la misma mediante los bytes enviados con un led que indica su correcto funcionamiento. En la Figura 2.35, se observa el circuito impreso del nodo de la cámara.

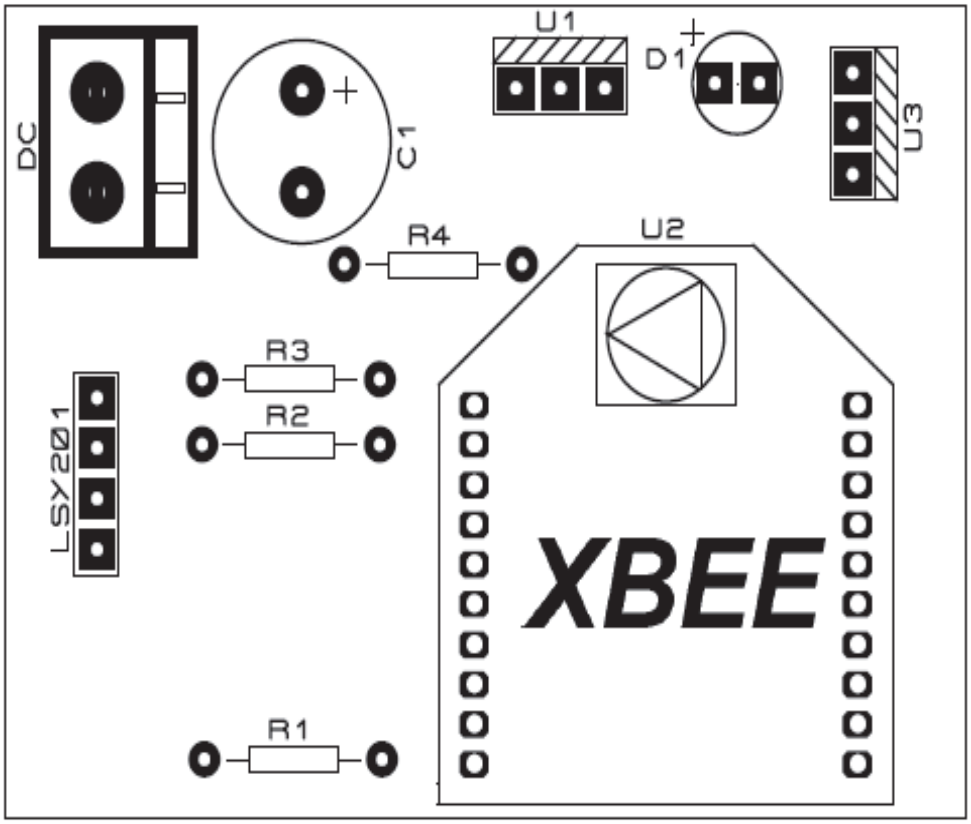


Figura 2.35 Impreso del circuito del nodo cámara.

En este capítulo se ha realizado un breve análisis de los algoritmos que han servido para desarrollar el programa y el hardware de cada nodo del sistema, permitiendo así obtener una base teórica para futuras mejoras y modificaciones en el sistema.

En el siguiente capítulo se detallarán los diferentes programas y plataformas utilizados para la implementación de una aplicación en la plataforma Arduino que permite la manipulación del sistema domótico.

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN ANDROID

En el presente capítulo se detalla el procedimiento y los programas utilizados en la creación e implementación de la aplicación Android necesaria para el control del sistema domótico.

3.1 ALMACENAMIENTO EN LA NUBE

El almacenamiento en la nube permite almacenar información y poder acceder a ella desde cualquier lugar utilizando diferentes dispositivos. Además, ofrece una seguridad adicional ya que se mantiene a salvo de problemas en los soportes físicos donde se suelen alojar los archivos, como por ejemplo discos duros.

El almacenamiento en la nube fue pensado y basado en el pago por uso, en este escenario se elimina el modelo de gastos de capital y es reemplazado por uno de gastos operativos, los mismos que comúnmente representan un 25% o 30% del presupuesto de un proyecto además permite un ahorro de personal, ya que la gestión corre por cuenta del proveedor del servicio. A continuación se anotan algunas empresas que brindan este servicio:

- ***FIREDRIVE*:** Este servicio brinda 50 GB gratis al registrarse. Tiene algunas opciones interesantes, como colocar contraseñas individuales a cada carpeta, permitir comentarios y *feedback* en ciertos archivos que sean públicos, adicionalmente se puede decidir quiénes los pueden ver y compartir.
- ***ADRIVE:*** Este servicio permite acceso FTP (*File Transport Protocol,* Protocolo de transferencia de archivos), seguridad aumentada, edición de documentos en línea (por ejemplo Word, Excel, Power

Point), expiración de archivos públicos, almacenamiento y visualización de archivos multimedia (imagen audio y video). Brinda un espacio gratuito de 50 GB al registrarse y tiene aplicaciones disponibles en Android, IOS y una multiplataforma hecha en Adobe Air la cual permite trabajar en cualquier ambiente indistintamente del sistema operativo que se esté utilizando.

- **ALMACENAMIENTO PROPIO:** En la actualidad hay alternativas que permiten crear una propia nube mediante aplicaciones de escritorio o móviles, exactamente igual como si se tuviera una nube con una empresa privada.

Es pertinente mencionar también que para fines del proyecto se optó por implementar un servidor web que optimiza la manipulación del sistema domótico, simplificando el desarrollo de la aplicación Android así como el procesamiento en el nodo Central, a continuación se describe los conceptos relacionados a este tópico.

3.1.1 SERVIDOR WEB

Un servidor web es un programa que proporciona contenido estático a un navegador, carga un archivo y lo presenta a través de la red al navegador de un usuario realizando conexiones bidireccionales. Implementa el protocolo HTTP (*Hypertext Transfer Protocol*) para transmitir datos tales como hipertexto, páginas web o páginas HTML (*HyperText Markup Language*, Lenguaje de Marcas de Hipertexto).

Se pueden utilizar varias tecnologías en el servidor para aumentar su potencia más allá de su capacidad de entregar páginas HTML; éstas incluyen scripts CGI (*Common Gateway Interface*), seguridad SSL (*Security Socket Layer*) y páginas activas del servidor ASP (*Active Server Pages*). En la Figura 3.1, se observa el esquema de conexión de un servidor web.

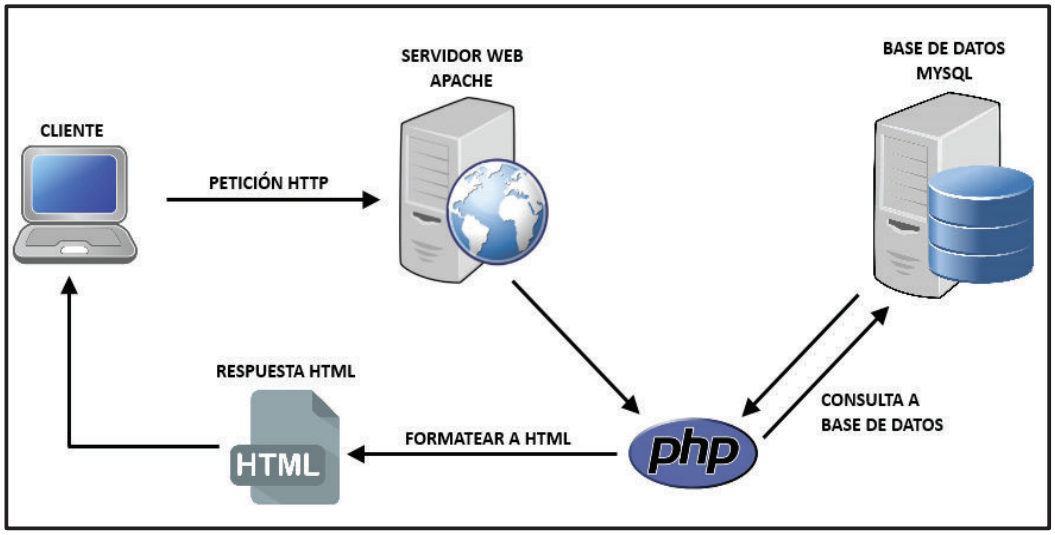


Figura 3.1 Esquema de conexión de un servidor Web [22].

3.1.1.1 XAMPP (Cross-Plataform Apache MySQL PHP Perl)

XAMPP es un servidor de código abierto que provee una configuración totalmente funcional desde el momento de su instalación. Este servidor incluye en su instalación lo siguiente:

Paquetes Básicos:

- Apache, servidor web.
- MySQL, base de datos de código abierto.
- PHP (*Hypertext Preprocessor*) y PERL (*Practical Extraction and Report Language* - Lenguaje Práctico para la Extracción e Informe): lenguajes de programación.
- ProFTPD: servidor FTP.

Paquetes Gráficos:

- GD (Graphics Draw), librería de dibujos gráficos.
- Libpng: librería oficial de referencia de PNG (*Portable Network Graphics*, Gráficos de Red Portátiles).

- Libpeg: librería oficial de referencia de JPEG (*Joint Photographic Experts Group*).

Paquete de bases de datos:

- SQLife: motor de base de datos SQL (*Structured Query Language*).

Paquetes PHP:

- PEAR: librería de PHP.
- La clase pdf que genera documentos PDF dinámicos con PHP.
- TURCK MMCache: potenciador del rendimiento de PHP.

Adicionalmente XAMPP tiene gestores como phpMyAdmin y phpSQLiteAdmin. En la Figura 3.2, se observa el panel de control del servidor XAMPP.

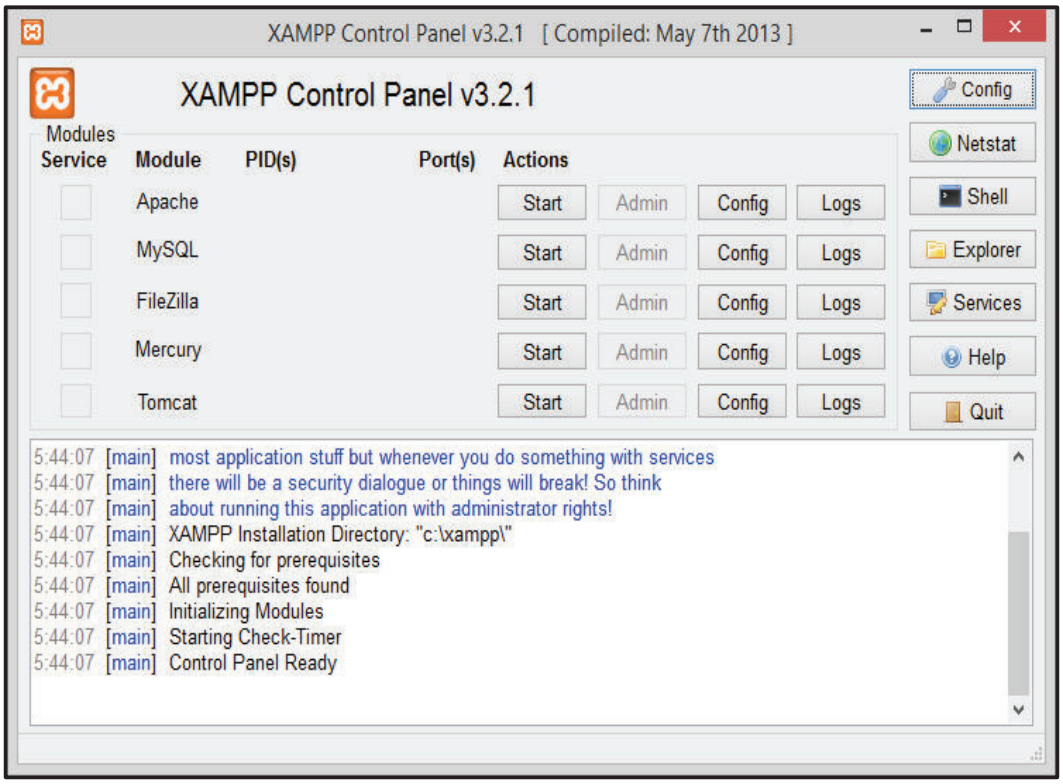


Figura 3.2 Panel de control del servidor XAMPP.

Se debe considerar que para levantar la página web creada en PHP es necesario levantar dentro de XAMPP los servicios Apache y MySQL y a su vez en la puerta de enlace se asignan los puertos 8045 para el servicio PHP y el 5000 para el servidor UDP. En la Figura 3.3, se observa la habilitación de los servicios Apache y MySQL en el panel de control del servidor XAMPP.

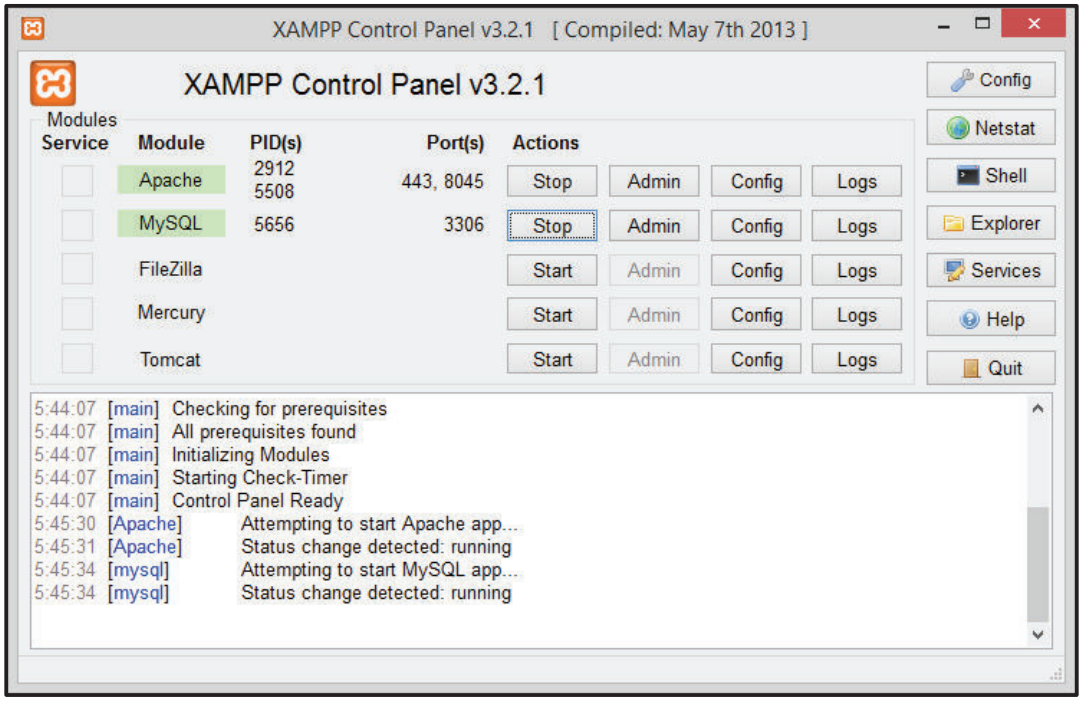


Figura 3.3 Arranque de los servicios Apache y MySQL.

3.1.1.2 PHP (*Personal Home Page, Hypertext Preprocessor*)

Es un lenguaje de código abierto y sirve para poder desarrollar páginas dinámicas que pueden ser incrustadas en HTML. El código es interpretado por un servidor web con un módulo de procesador PHP que genera la página resultante.

PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Para la elaboración de la página web en PHP se puede usar simplemente el blog de notas, herramientas o IDE (*Integrated Development Environment* - Entorno de desarrollo integrado) más avanzados como *Notepad++ o Sublime Text* conocidos como ficheros.

Un fichero PHP está formado por flujos de texto que contienen la información de la página que se desea crear; la extensión de este tipo de ficheros se define con .php.

Este lenguaje de programación permite manipular archivos a través de la gran variedad de funciones que posee y obtener así una página con varios ficheros, considerando que el fichero principal al que accede el usuario se debe llamar index.php. Las pautas principales para empezar un fichero PHP son:

```
<?php

// Código php

?>
```

Dónde:

- <?php indica el inicio del programa a ser escrito,
- // indican un comentario dentro del programa. También se puede cambiar el valor de las variables en cualquier momento, es decir que una variable puede empezar siendo un entero y terminar la ejecución de la variable como texto.
- ?> indica el final del programa.

Adicionalmente el lenguaje PHP cuenta con funciones ya definidas que permiten realizar todo tipo de operaciones simples como mostrar un texto por pantalla o descargar el contenido de una web. Las sentencias condicionales permiten elegir entre dos caminos de ejecución diferentes en función de la evaluación de una

expresión lógica, es decir que si resulta verdadera se ejecuta las sentencias y si es falsa se ignora.

Para el presente proyecto se elaboraron 4 ficheros, los mismos que permiten controlar y sensar el sistema domótico. En la Figura 3.4, se puede observar el fichero creado para el menú de la aplicación del sistema domótico.

Sin título: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?php
require_once("rpcl/rpcl.inc.php"); //verificar si el archivo ya ha sido incluido

//definicion de unidades basicas de aplicaciones móviles
use_unit("forms.inc.php");
use_unit("extctrls.inc.php");
use_unit("stdctrls.inc.php");


//Definición de clase
class MENU extends Page
{
    public $Button3 = null;//mostrar botón
    public $Button2 = null;
    public $Button1 = null;
    function QButton1Click($sender, $params)
    {
     header("location:ACTUADORES.php"); // permite enviar una cabecera HTTP pura para los actuadores
       exit;
    }
    function QButton2Click($sender, $params)
    {
     header("location:SENSORES.php"); // permite enviar una cabecera HTTP pura para los sensores
       exit;
    }
    function QButton3Click($sender, $params)
    {
    header("location:IMAGEN.php"); // permite enviar una cabecera HTTP pura para la imágen
       exit;
    }

}

global $application;

global $MENU;

//Crear la forma
$MENU=new MENU($application);

//leer del archivo de origen
$MENU->loadResource(__FILE__);

//mostrar el menú
$MENU->show();

?>
```

Figura 3.4 Fichero del menú del sistema domótico.

En la Figura 3.5, se puede observar el fichero creado para controlar los diferentes nodos del sistema domótico.

Sin título: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?php
require_once("rpcl/rpcl.inc.php");//verifica si el archivo ya ha sido incluido

//definicion de unidades basicas de aplicaciones móviles
use_unit("forms.inc.php");
use_unit("extctrls.inc.php");
use_unit("stdctrls.inc.php");
use_unit("qooxdoo/stdctrls.inc.php");

//Definicion de clase

class ACTUADORES extends Page
{
    public $Image4 = null; //mostrar imagen
    public $Image11 = null;
    public $Label1 = null; //mostrar texto
    public $Label2 = null;
    public $Label4 = null;
    public $Label5 = null;
    public $Label8 = null;
    public $Label9 = null;
    public $Image1 = null;
    public $Image3 = null;
    public $Image5 = null;
    public $Image6 = null;
    public $Image7 = null;
    public $Image8 = null;
    public $Label3 = null;
    public $Image9 = null;
    public $Image10 = null;
    public $Image13 = null;
    public $Image14 = null;
    public $Image15 = null;
    public $Image16 = null;
    public $Image17 = null;
    public $Image18 = null;
    public $Label6 = null;
    public $Image2 = null;
    public $Label7 = null;
    public $Edit1 = null;
    public $Button2 = null; // mostrar boton
    function Image1Click($sender, $params)
    {
                        try
                            {
                             $conexion = mysql_connect("localhost", "imagenes", "imagenes");
                              mysql_select_db("imagenes", $conexion);

                              $queEmp = "UPDATE  actuadores SET a1=1";
                              $resEmp = mysql_query($queEmp, $conexion) or die(mysql_error());
                               if($resEmp==1)
```

Figura 3.5 Fichero que permite el control de los diferentes nodos del sistema domótico.

En la Figura 3.6, se puede observar el fichero creado para monitorizar los diferentes nodos del sistema domótico.

Sin título: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?php
//Obtencion de las variables a traves del metodo GET
$s1=$_GET['s1'];
$s2=$_GET['s2'];
$s3=$_GET['s3'];
$s4=$_GET['s4'];
$s5=$_GET['s5'];
$s6=$_GET['s6'];
$s7=$_GET['s7'];
$s8=$_GET['s8'];
$s9=$_GET['s9'];
$s10=$_GET['s10'];
$s11=$_GET['s11'];
$s12=$_GET['s12'];

//Establecimiento de la conexion con la base de datos
//http://localhost/casa/sensor.php?s1=1&s2=1&s3=1&s4=1&s5=1&s6=1&s7=1&s8=1&s9=1&s10=1&s11=1&s12=1
$conexion = mysql_connect("localhost", "imagenes", "imagenes");
mysql_select_db("imagenes", $conexion);

//$queEmp = "UPDATE sensores SET s1='$s1', s2='$s2', s3='$s3', s4='$s4', s5='$s5', s6='$s6', s7='$s7', s8='$s8', s9='$s9', s10='$s10', s11='$s11', s12='$s12'";

//Query de Actualizacion de los sensores
$queEmp = "UPDATE sensores SET s1='$s1', s2='$s2', s3='$s3', s6='$s6', s7='$s7', s8='$s8', s9='$s9', s10='$s10', s11='$s11', s12='$s12'";
//$queEmp = "UPDATE sensores SET s1='$sen1', s2='$sen2', s3='$sen3', s4='$sen4'";

//Ejecucion del query que actualiza los parametros s1,s2,s3....,s12
$resEmp = mysql_query($queEmp, $conexion) or die(mysql_error());

echo "OK"; //Mensaje de ok para que se lo visualize en la pantalla
?>
```

Figura 3.6 Fichero que permite monitorizar los diferentes nodos del sistema domótico.

En la Figura 3.7, se puede observar el fichero creado para el nodo de la cámara.

Sin título: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?php
//Declaracion de los modulos de php
require_once("rpcl/rpcl.inc.php");
//Includes
use_unit("forms.inc.php");
use_unit("extctrls.inc.php");
use_unit("stdctrls.inc.php");

//Declaracion de la Clase IMAGEN
class IMAGEN extends Page
{
    //http://192.168.1.9:8045/imagen.php
    //Declaracion e Inicializacion de variables
        public $Button4 = null;
    public $Image1 = null;
    public $Button1 = null;
    public $Button2 = null;
    public $Button3 = null;

    function Button1Click($sender, $params) //Funcion que se ejecuta al pulsar el boton 1
    {
     $this->Image1->Width=$this->Image1->Width+10; //Se imcrementa el ancho de la imagen en 10 pixeles
     $this->Image1->Height=$this->Image1->Height+10; //Se imcrementa el alto de la imagen en 10 pixeles
    }

    function Button2Click($sender, $params) //Funcion que se ejecuta al pulsar el boton 2
    {
     $this->Image1->Width=$this->Image1->Width-10; //Se decrementa el ancho de la imagen en 10 pixeles
      $this->Image1->Height=$this->Image1->Height-10; //Se decrementa el ancho de la imagen en 10 pixeles
    }

    function Button3Click($sender, $params) //Funcion que se ejecuta al pulsar el boton 3
    {
      $self = $_SERVER['IMAGEN.php']; //Obtenemos la página en la que nos encontramos
      header("refresh:0.01; url=$self");
    }
```

Figura 3.7 Fichero que permite obtener la imagen desde el nodo de la cámara.

La base de datos que se tiene para la creación de la página antes descrita se almacena en el servidor XAMPP, en el gestor phpMyAdmin. En la Figura 3.8, se puede observar el acceso al servidor XAMPP y la dirección http://localhost:8045/xampp/, que permite el acceso al gestor phpMyAdmin.

Figura 3.8 Opciones del servidor XAMPP.

El gestor phpMyAdmin puede administrar desde algo muy sencillo hasta un servidor MySQL entero, todo depende de las necesidades del usuario. Para poder administrar el gestor phpMyAdmin es necesario tener un usuario MySQL debidamente configurado, el mismo que permita leer o escribir dicha base de datos sin afectar otras que puedan estar presentes en el servidor.

En la actualidad, phpMyAdmin puede borrar, crear, copiar, renombrar y alterar bases de datos, tablas, campos e índices, así como mostrar múltiples resultados a través de procedimientos almacenados o consultas, además permite realizar labores de mantenimiento al servidor, bases de datos y tablas.

Para la manipulación del sistema domótico se tiene diferentes tablas que se encuentran ligadas a los ficheros anteriormente creados. En la Figura 3.9, se observa la pantalla del gestor phpMyAdmin en el cual se almacena diferentes tablas que permiten la configuración del sistema domótico como es el caso de los actuadores.

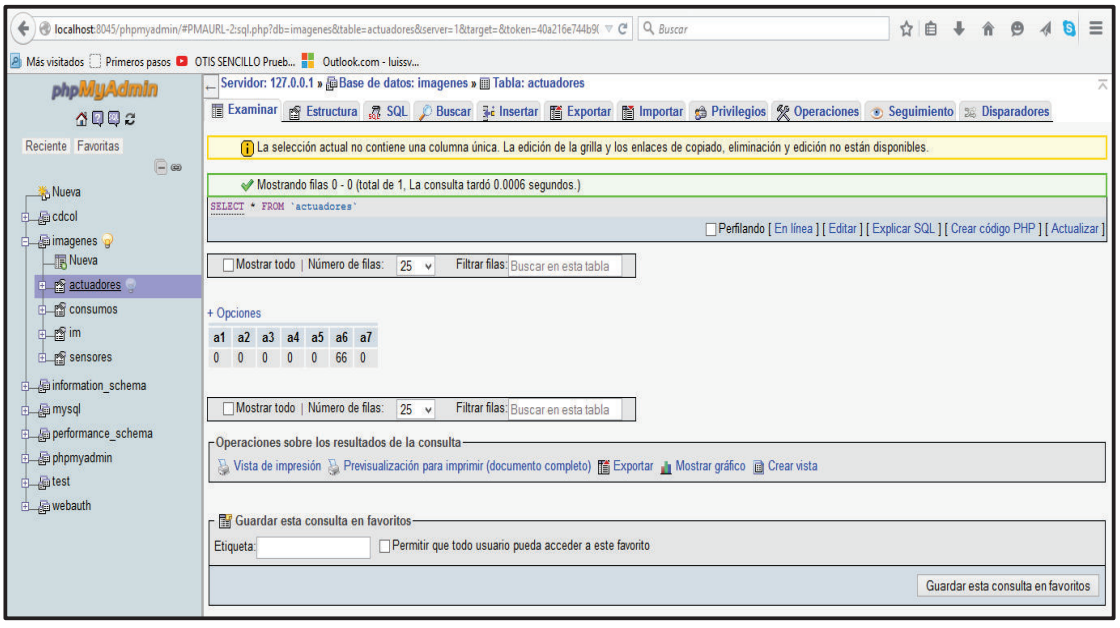


Figura 3.9 Tabla de actuadores.

En la Figura 3.10, se observar la tabla empleada para los sensores del sistema domótico.

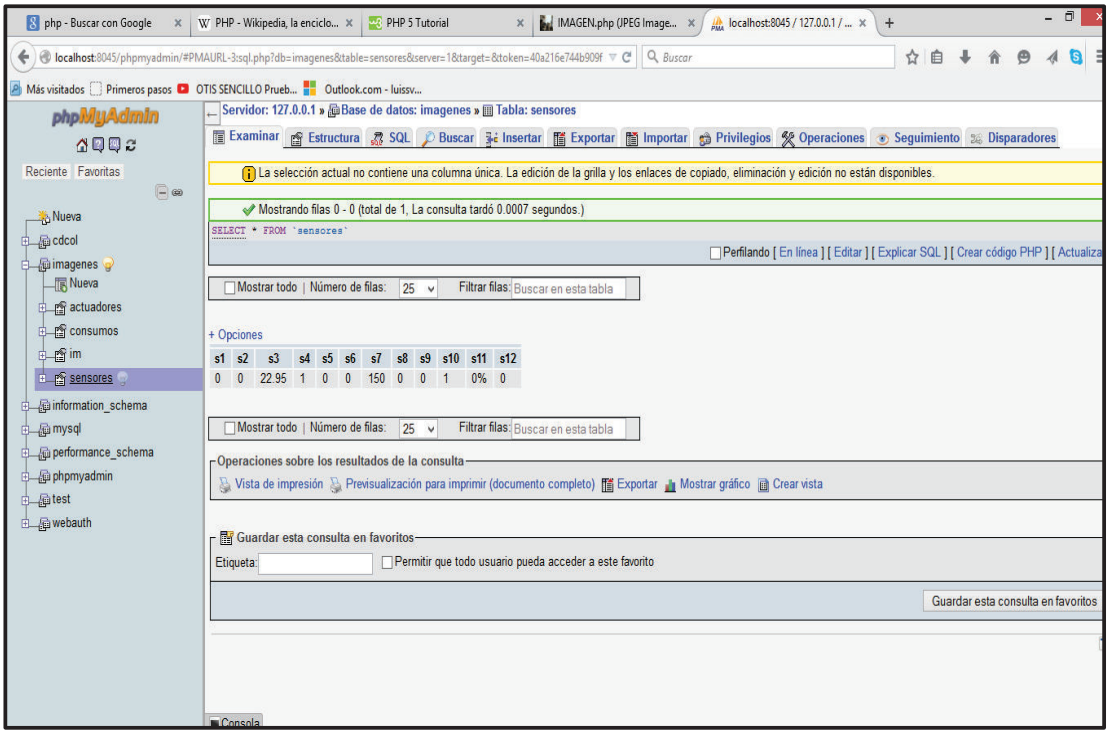


Figura 3.10 Tabla de sensores.

Adicionalmente se levantó un Server UDP, el mismo que recibe los paquetes enviados principalmente por la cámara con Python ejecutando un pequeño script en el servidor el mismo que crea un socket en el número de puerto especificado en este caso el puerto 5000, este pequeño programa empieza a escuchar a la espera de que el cliente envíe información a la base de datos.

3.2 APP INVENTOR

3.2.1 INTRODUCCIÓN

En los inicios de la programación en Android Google en el 2010 ideó una forma sencilla de desarrollar aplicaciones utilizando la programación gráfica y consiguiendo atraer de esta manera a los desarrolladores expertos e inexpertos, sin embargo un año después de obtener excelentes resultados abandonó este proyecto, dejando abierto el código para todo aquel que deseara continuar con su

desarrollo lo hiciera. En el 2011 el MIT (Instituto Tecnológico de Massachusetts) para desarrollar el aprendizaje móvil retoma el proyecto y se ha mantenido así hasta la actualidad.

3.2.2 DEFINICIÓN

App Inventor es un *framework* creado inicialmente por Google Labs y luego acogido por el MIT y que ha sido ideado para que cualquier persona con interés y poco conocimiento de programación pueda crear su propia aplicación móvil, ya sea a nivel empresarial, para el hogar o para otros intereses. Todo el trabajo se desarrolla en un entorno visual y la programación se realiza por medio de bloques enlazables, a modo de rompecabezas. Finalmente se puede hacer uso de las capacidades propias de un *Smartphone* como son la cámara fotográfica, la cámara de video, el acelerómetro, el GPS, el Bluethoot.

App Inventor es un servicio basado en la nube por lo que todas las tareas se realizan en un navegador, para su funcionamiento es necesario tener una cuenta en google (gmail). En esta plataforma el usuario puede ir ensamblando de forma visual, una serie de bloques para crear la aplicación a partir de un conjunto de herramientas básicas. Las aplicaciones de *App Inventor* son limitadas debido a su simplicidad, pero permiten cubrir un gran número de necesidades básicas en un dispositivo móvil.
Al construir las aplicaciones para Android se trabajará con tres herramientas: Gestor de proyectos, *App Inventor Designer* y *App Inventor Blocks Editor*.

Gestor de Proyectos: Permite dar un seguimiento de todos los proyectos asociados con la cuenta de *gmail*. En la Figura 3.11, se observan los proyectos asociados a una cuenta *gmail* especifica.

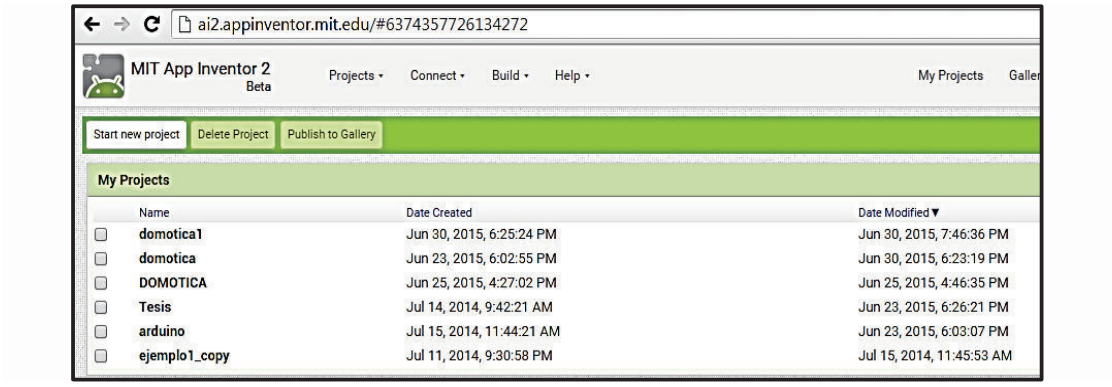


Figura 3.11 Gestor de proyectos.

***App Inventor Designer* (Diseño de la aplicación):** En esta ventana se puede implementar la interfaz de usuario y los componentes que se usarán en la aplicación. En la Figura 3.12, se visualiza los componentes que permiten la construcción de la interfaz de usuario.

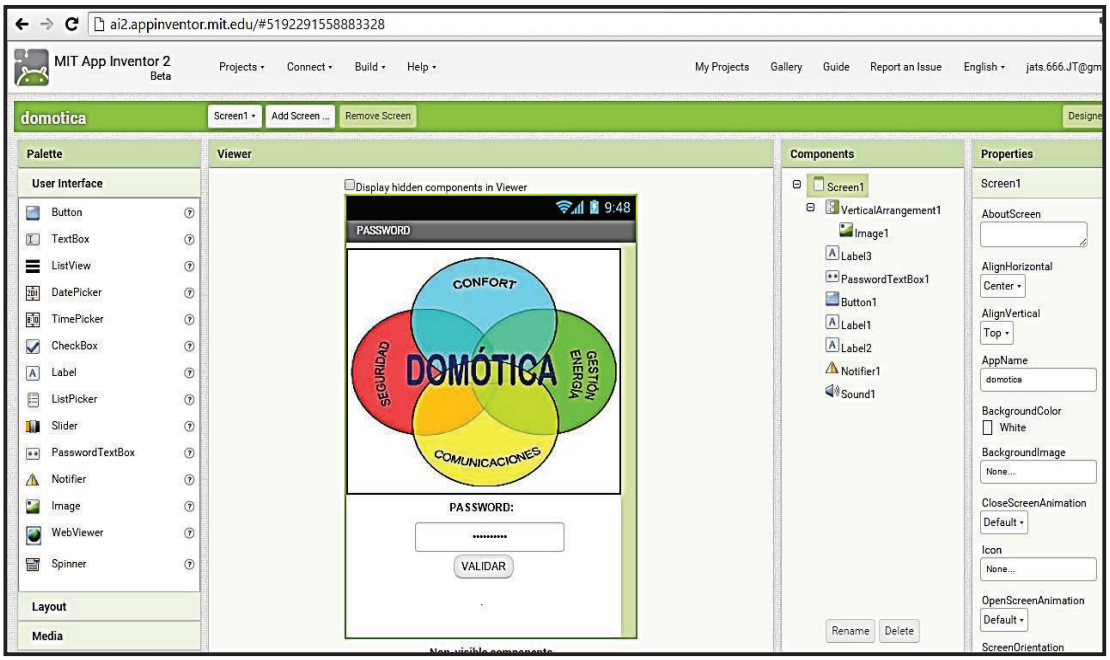


Figura 3.12 Diseño de la aplicación a construir.

App Inventor Blocks Editor (editor de bloques): En esta ventana se define el comportamiento de los componentes de la aplicación, utiliza la librería *Open Blocks* de Java para crear un lenguaje visual y el compilador que lo traduce para

la aplicación en Android. En esta sección se utiliza Kawa[11] como código de programación. En la Figura 3.13, se visualiza el editor de bloques que permiten la creación de la aplicación.

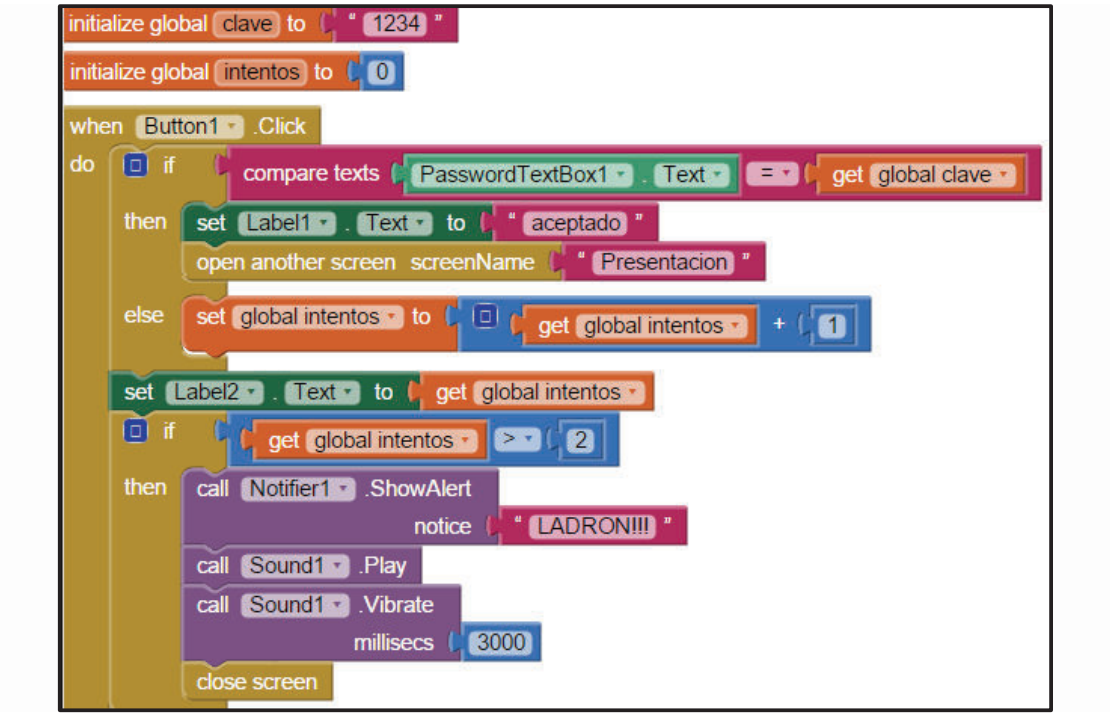


Figura 3.13 Editor de bloques.

App Inventor permite probar las aplicaciones que se van construyendo en tiempo real en el mismo dispositivo Android mediante el escaneo del código QR (*Quick Response Code*, Código de Respuesta Rápida) que se genera al seleccionar la pestaña *"Connect"* >> "*AI Companion*".

El entorno de desarrollo de *App Inventor* es compatible con Mac OS X, GNU / Linux y sistemas operativos de Windows y Android. El sistema es gratuito y se puede descargar fácilmente de la web. Con Google App Inventor, se espera un

---

[11] Kawa es un lenguaje de programación libre y es pare del proyecto GNU para la plataforma Java, también puede ser utilizado como un "lenguaje de scripting", pero incluye un compilador y todos los beneficios de un lenguaje de programación "real". Ha sido utilizado en aplicaciones como el App Inventor para Android.

incremento importante en el número de aplicaciones para Android debido a dos grandes factores:

- La simplicidad de uso, que facilita la creación de nuevas aplicaciones.
- Android Market, el centro de distribución de aplicaciones para Android, donde cualquier usuario puede distribuir sus creaciones libremente.

3.2.3 USO DEL DISPOSITIVO ANDROID CON APP INVENTOR

Para probar la aplicación en el dispositivo Android es necesario instalar el aplicativo *MIT AI2 Companion*, disponible en el *APP STORE* de Google de forma gratuita. En la Figura 3.14, se observa el aplicativo disponible en el APP STORE.

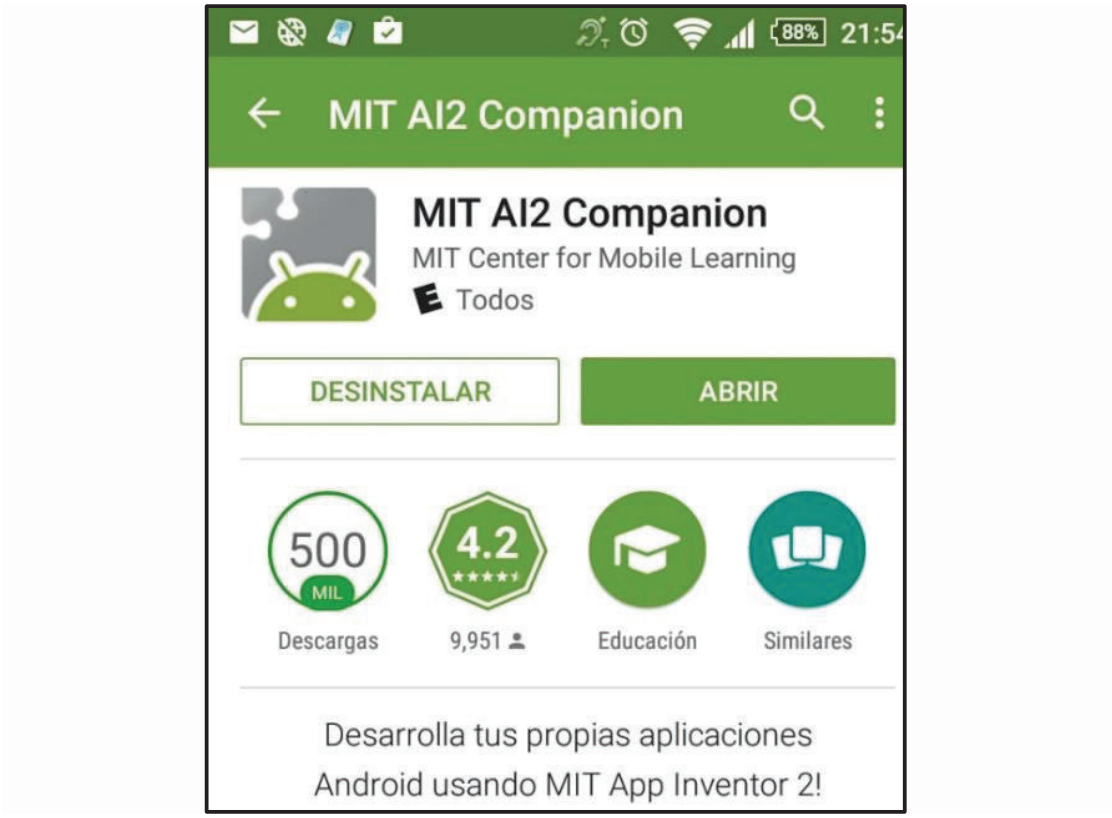


Figura 3.14 Aplicativo MIT AI2 Companion disponible en el APP STORE.

A continuación se debe abrir el proyecto en el App Inventor, luego se selecciona la pestaña "*Connect*" y se selecciona "*AI Companion*" la cual genera un código QR *(Quick Response Code)* y un código de texto como se observa en la Figura 3.15.

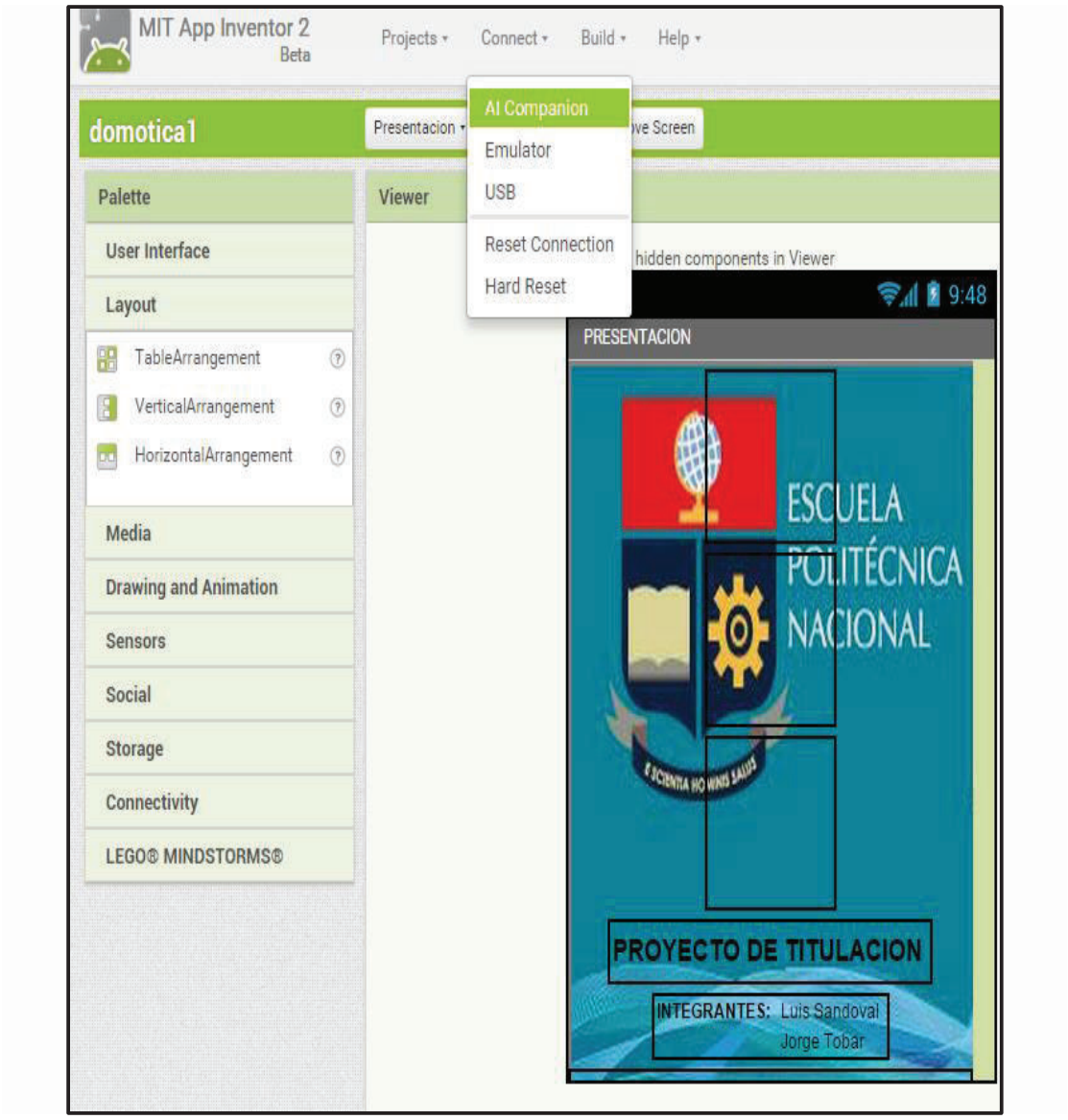


Figura 3.15 Selección de AI Companion.

Estos códigos deben ser ingresados al dispositivo mediante el escaneo o la digitación de caracteres pulsando "*Connect with code*". En la Figura 3.16, se observa el escaneo del código QR.

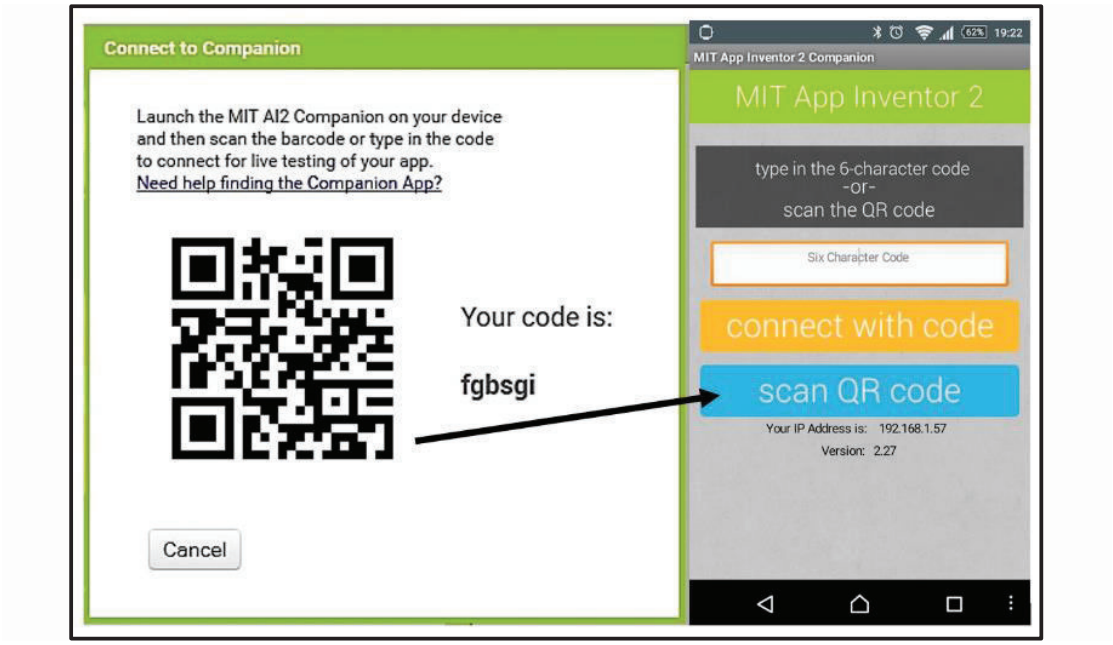


Figura 3.16 Escaneo del código QR.

Para que *App Inventor* muestre automáticamente la aplicación que se está construyendo, el ordenador y el dispositivo Android deben estar conectados a la misma red como se puede observar en la Figura 3.17.

Figura 3.17 Ordenador y android en la misma red WiFi.

La aplicación se actualiza a medida que se van realizando los cambios tanto en el diseño como en el comportamiento, a esta característica se le denomina "*live testing*" (prueba en directo). Una vez que se ha finalizado la creación de la aplicación y se ha decidido instalar la misma en el dispositivo es necesario crear el archivo con extensión .apk, mediante la pestaña "*Build*" >> "*App (provide QR code for .apk)*" como se muestra en la Figura 3.18.

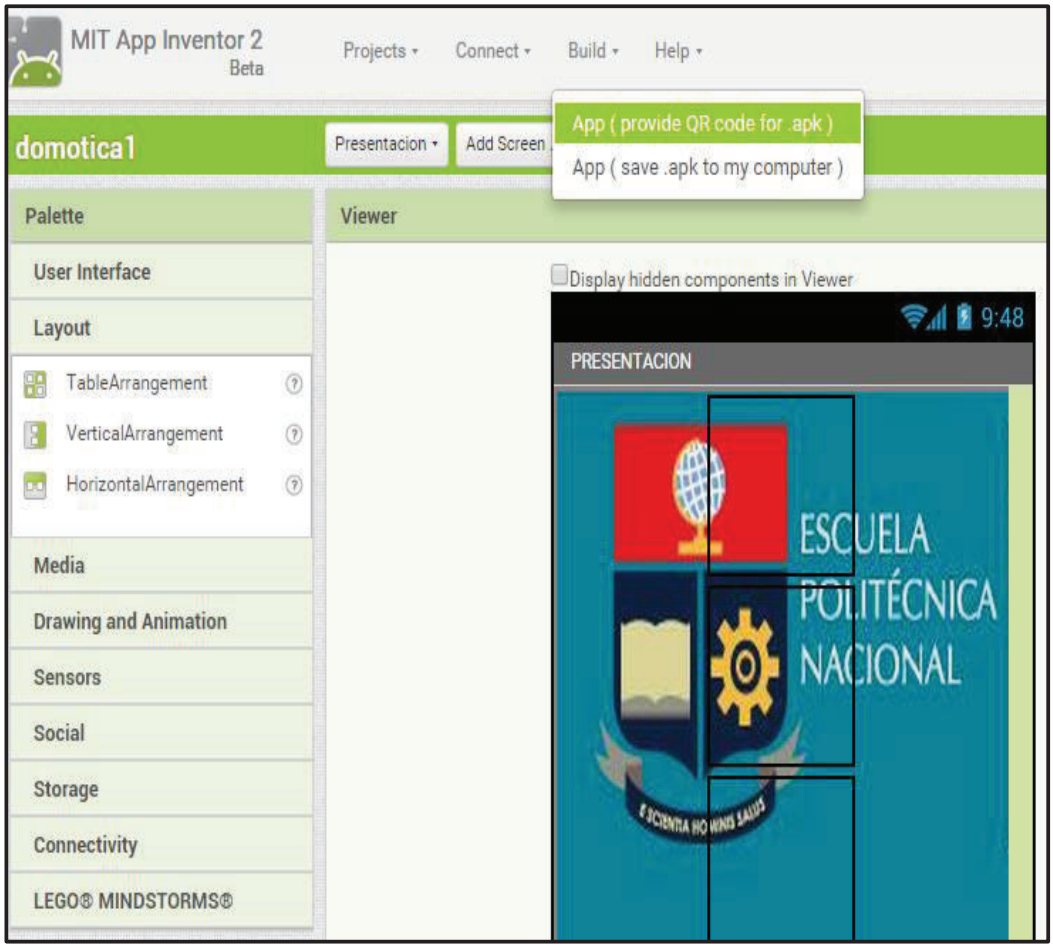


Figura 3.18 Selección App (*provide QR code for .apk*).

Inmediatamente aparece en la pantalla una barra de estado en donde se indica el progreso en la construcción de la aplicación, al completarse se despliega el código QR, el mismo que al ser escaneado y aceptado por el dispositivo Android la app creada se descarga e instala quedando lista para su utilización.

Para que el dispositivo permita la instalación de la aplicación final de Inventor se debe asegurar que la opción de fuentes desconocidas esté activada, con esto se permite la instalación de aplicaciones provenientes de otro tipo de fuentes como se visualiza en la Figura 3.19.

Figura 3.19 Activación de fuentes desconocidas.

Una vez aceptada la instalación de la aplicación en el dispositivo se crea un icono con el nombre del proyecto generado en el App Inventor como se observa en la Figura 3.20.

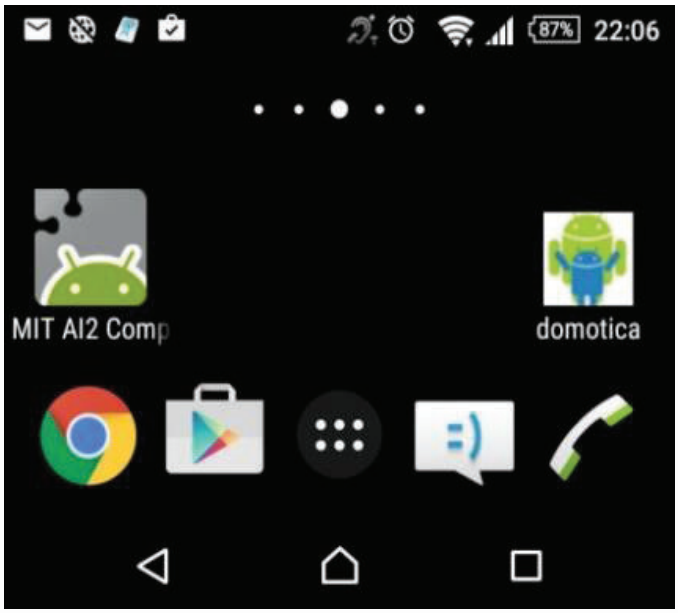


Figura 3.20 Icono de acceso a la aplicación domótica creada.

3.3 APLICACIÓN DESARROLLADA

Es esta sección se mostrará las imágenes de la aplicación desarrollada para el sistema domótico tanto en la vista de *Designer* como en la de *Blocks Editor*.

3.3.1 PANTALLA "CONTRASEÑA"

Esta es la pantalla de inicio de la aplicación donde se ingresa la contraseña configurada previamente en los bloques.

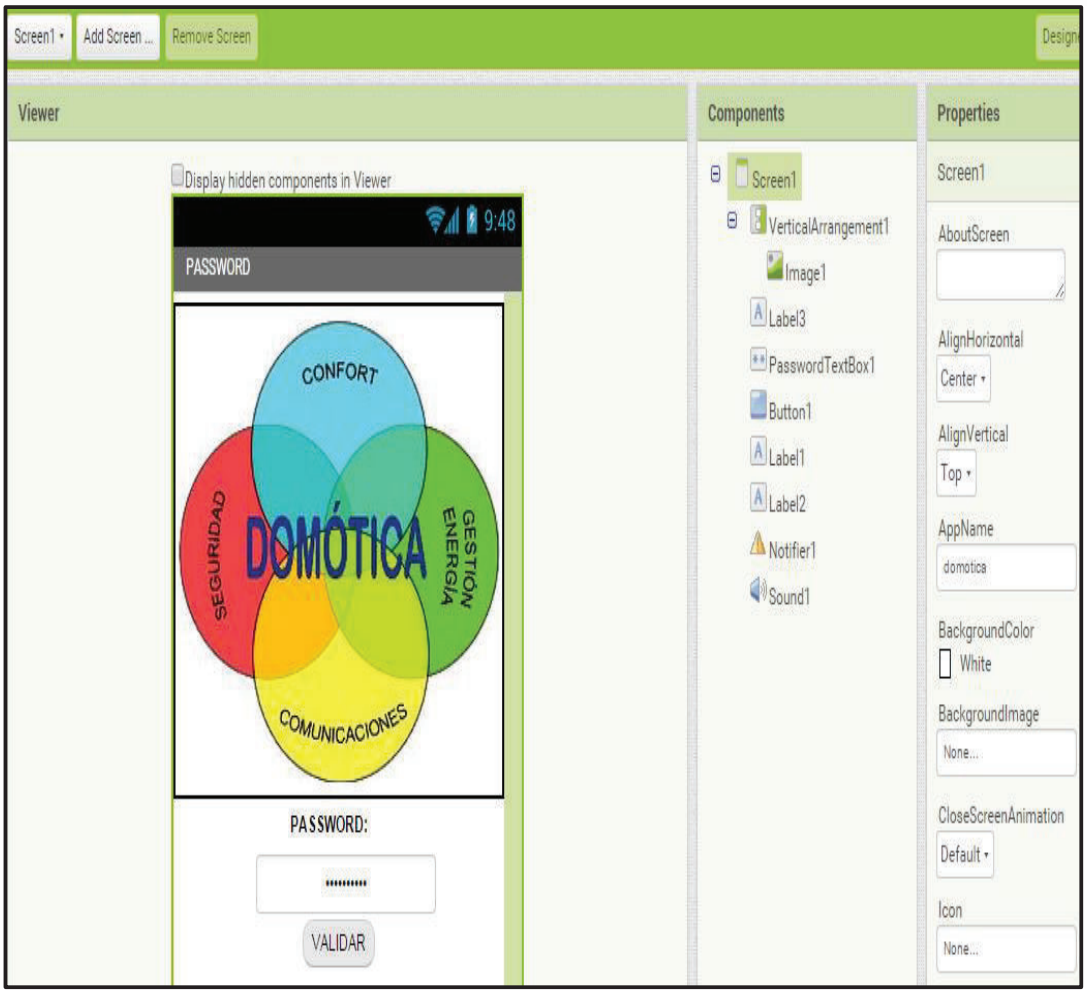


Figura 3.21 Pantalla contraseña vista en *Designer.*

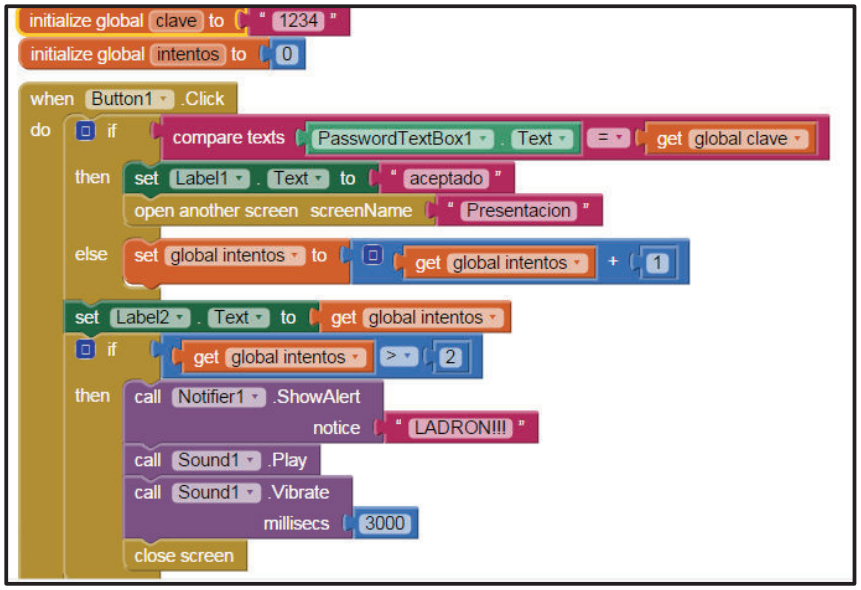


Figura3.22 Pantalla contraseña vista en *Blocks.*

3.3.2 PANTALLA "PRESENTACION"

En esta pantalla se muestra la presentación de la aplicación y los nombres de los autores.

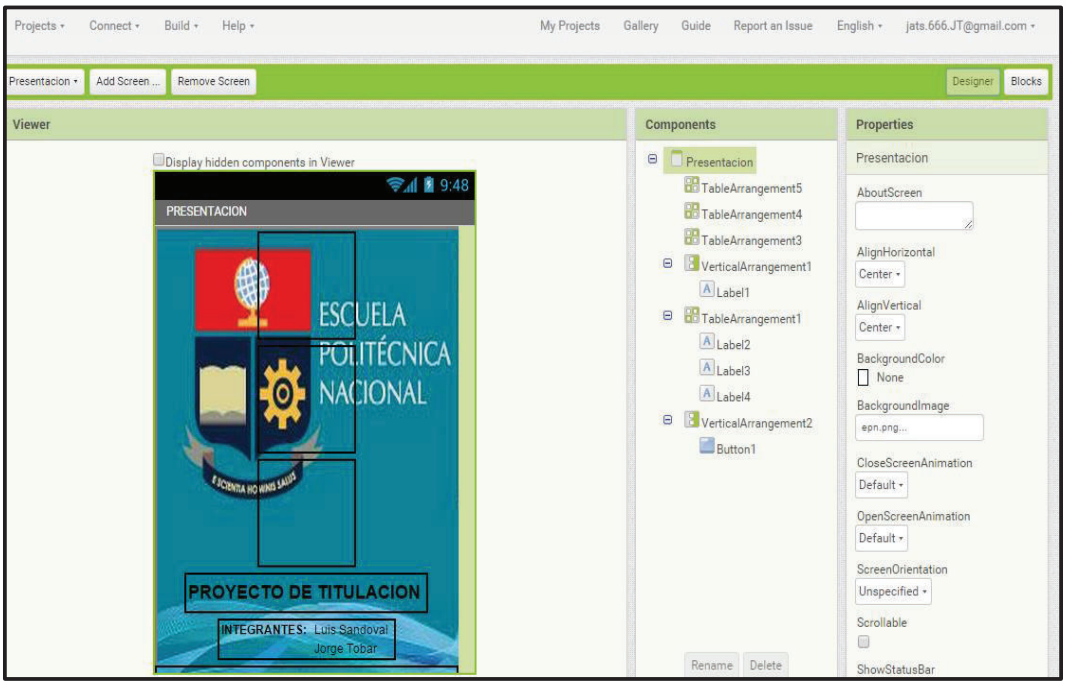


Figura 3.23 Pantalla presentación vista en *Designer.*

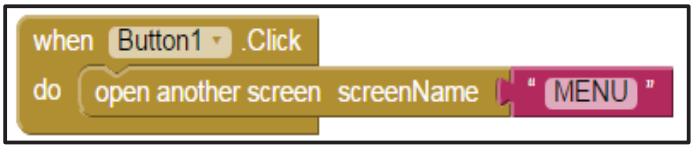


Figura 3.24 Pantalla presentación vista en *Blocks.*

3.3.3 PANTALLA "MENÚ"

En esta pantalla se muestra el menú de la aplicación y los botones que dirigen a diferentes pantallas.

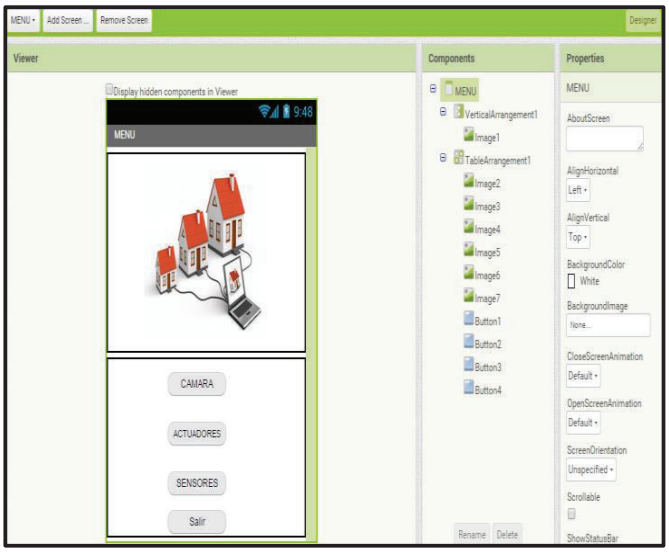


Figura 3.25 Pantalla menú vista en *Designer.*

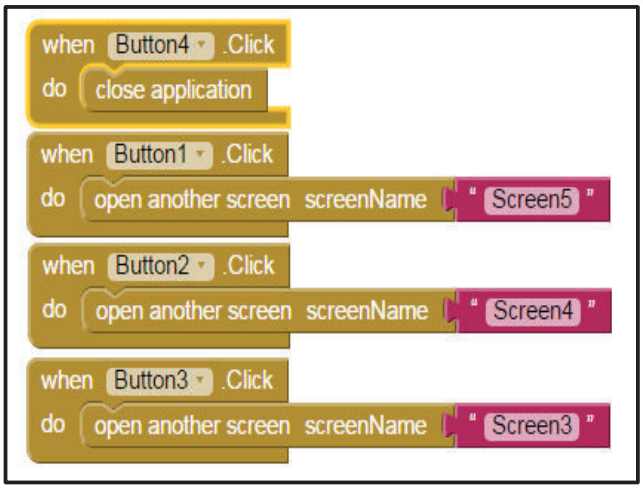


Figura 3.26 Pantalla menú vista en *Blocks.*

3.3.4 PANTALLA "CÁMARA"

En esta pantalla se observa la página Web almacenada en el servidor que muestra las imágenes captadas por la cámara y los diferentes valores de zoom.

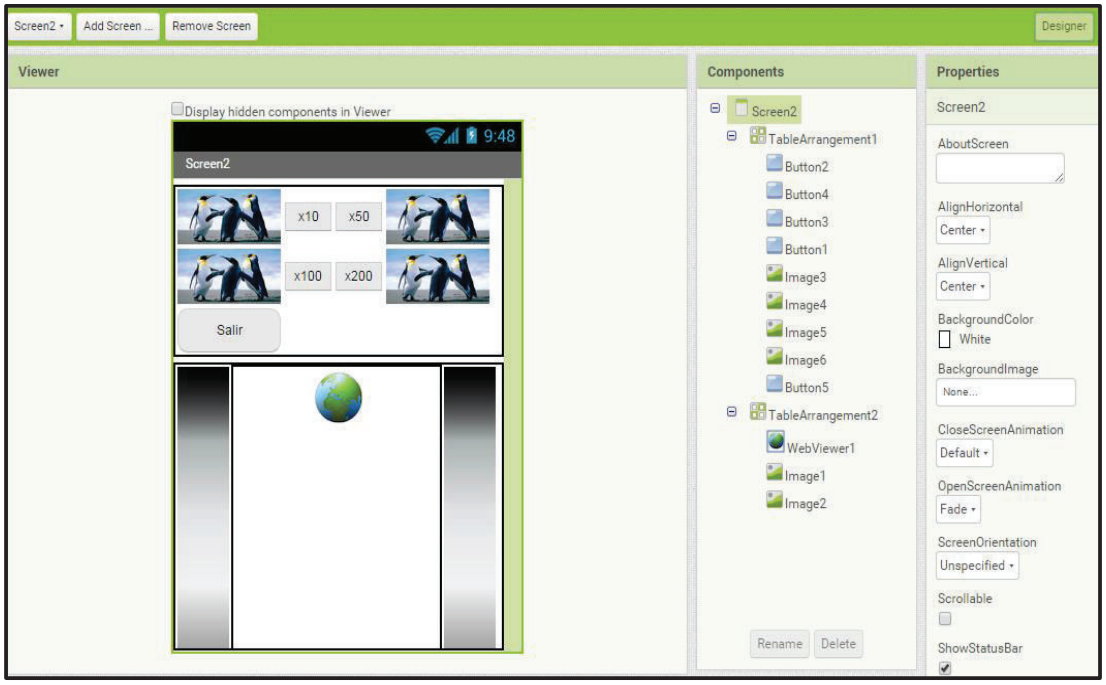


Figura 3.27 Pantalla cámara vista en *Designer.*

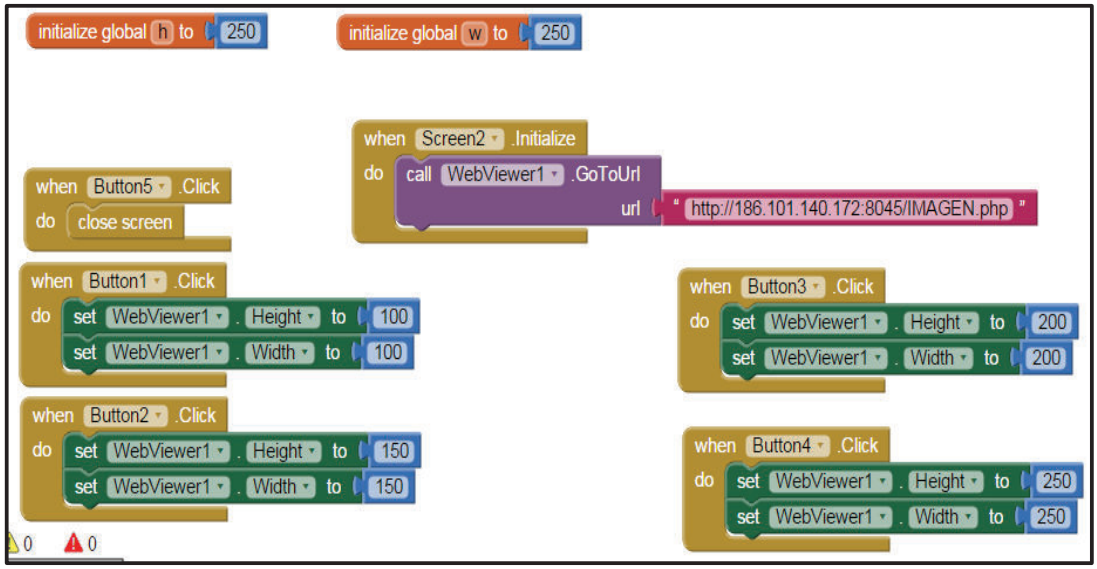


Figura 3.28 Pantalla cámara vista en *Blocks.*

3.3.5 PANTALLA "SENSORES"

En esta pantalla se observa la página Web almacenada en el servidor que muestra el estado de los sensores asociados a cada nodo dentro del sistema domótico.

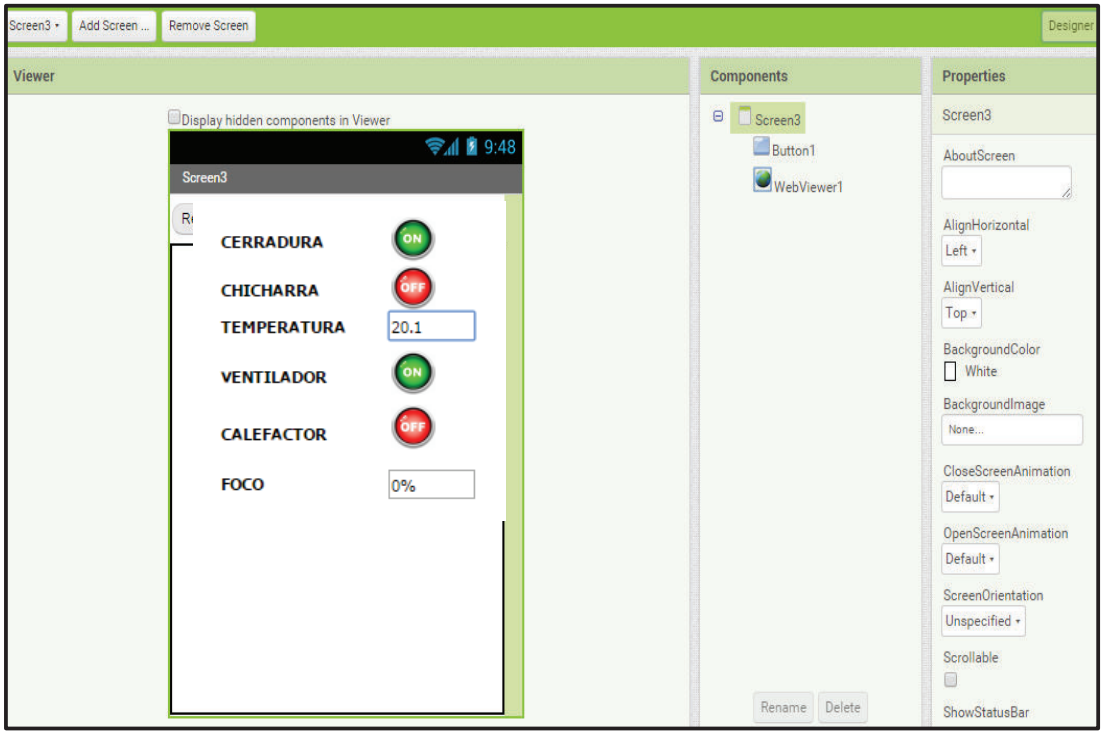


Figura 3.29 Pantalla sensores vista en *Designer.*

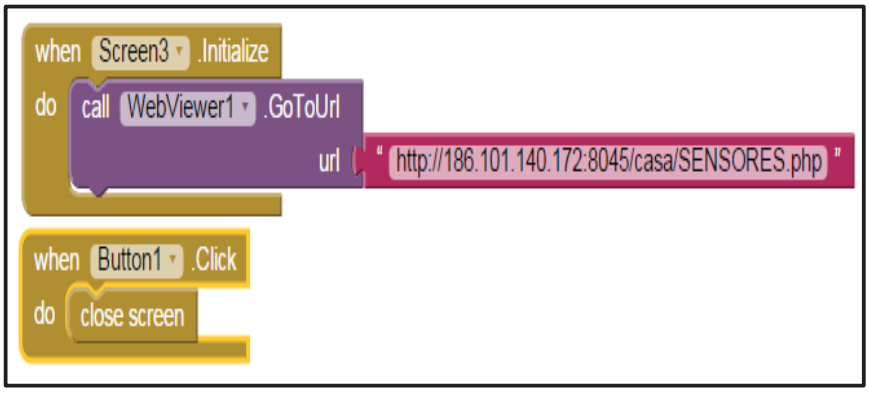


Figura 3.30 Pantalla sensores vista en *Blocks.*

3.3.6 PANTALLA "ACTUADORES"

En esta pantalla se observa la página Web almacenada en el servidor que muestra el estado de los sensores asociados a cada nodo dentro del sistema domótico.

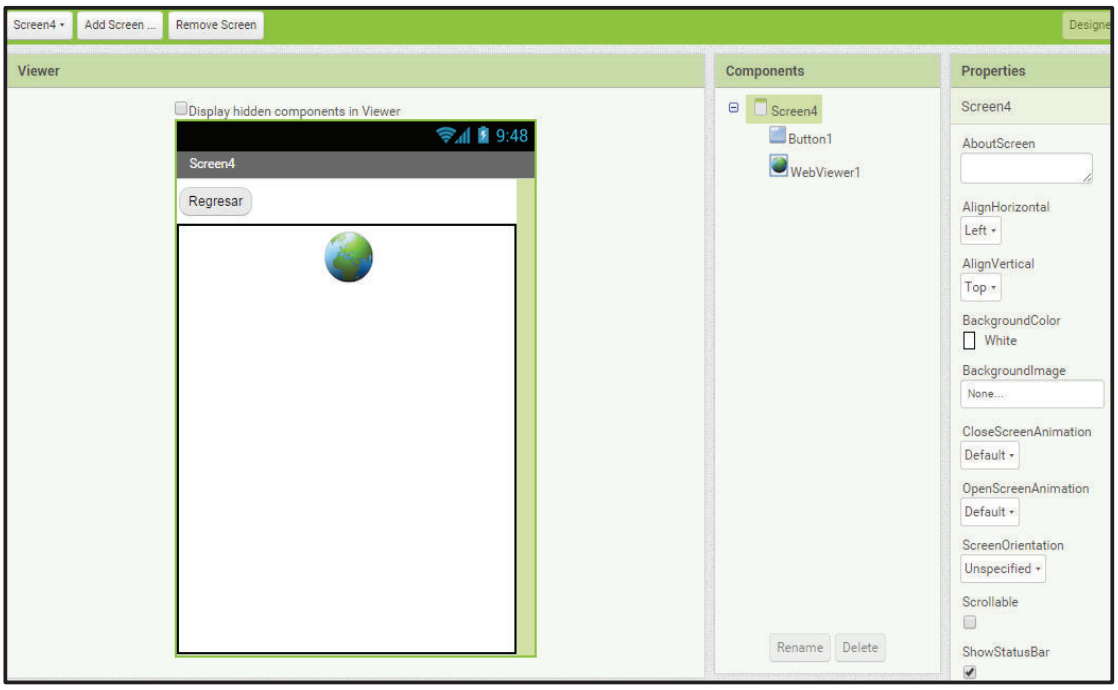


Figura 3.31 Pantalla actuadores vista en *Designer.*

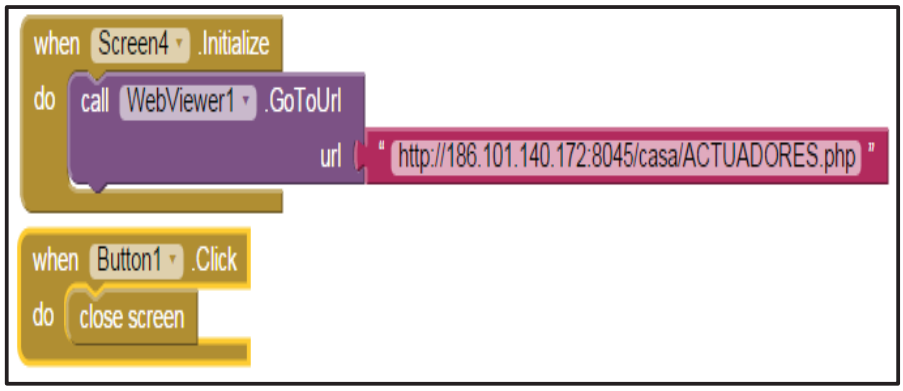


Figura 3.32 Pantalla sensores vista en *Blocks.*

En la Figura 3.33 se observa el diagrama final del sistema domótico implementado.

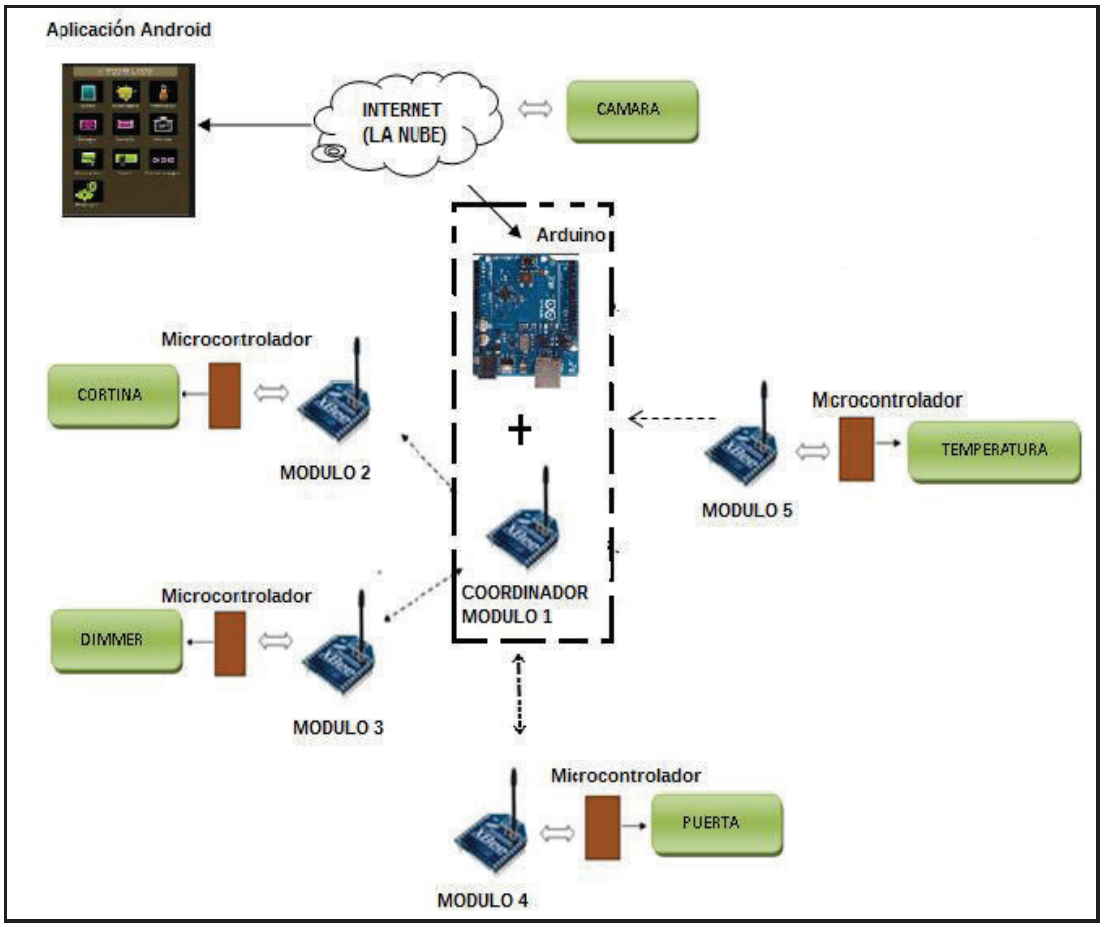


Figura 3.33 Diagrama del sistema domótico implementado.

En este capítulo se realizó una explicacion de las principales características del servidor Web y del App Inventor usados en la elaboración de la aplicación Android que permite el control de todo el sistema domótico.

En el siguiente capítulo se detallarán las pruebas realizadas en los nodos del sistema domótico y se hará una cotización de los materiales utilizados para la implementación del prototipo.

CAPÍTULO 4

PRUEBAS FUNCIONALES DEL PROTOTIPO Y ESTIMACIÓN DE COSTOS

En el presente capítulo se hace una descripción detallada sobre los resultados obtenidos después de realizar las diferentes pruebas de funcionamiento y el costo invertido en la elaboración del sistema domótico.

Entre los parámetros más relevantes que se consideraron para las pruebas funcionales del sistema domótico están:

- Fiabilidad.
- Robustez.
- Detección automática de estado.
- Retardo en la conexión del sistema.

4.1 PRUEBAS EN LOS NODOS

4.1.1 NODO DE CONTROL DE TEMPERATURA AMBIENTAL

La aplicación del control del calefactor y el ventilador que se implementó para este sistema funciona correctamente y sin ningún tipo de complicación. El usuario puede establecer el valor de la temperatura según sus necesidades a través de la aplicación Android.

En la Tabla 4.1, se observa una comparación de temperatura entre el valor obtenido desde la aplicación Android y el de un termómetro, obteniendo una variación de 1°C en cada toma con lo cual se concluye que el nodo de control de temperatura ambiental funciona correctamente. Además se realizó pruebas de

establecimiento de temperatura desde la aplicación Android ingresando valores indistintos para verificar así el funcionamiento adecuado del calefactor y ventilador respectivamente intentado alcanzar la temperatura establecida como idealmente era la acción esperada.

Tabla 4.1 Pruebas del nodo control de temperatura ambiental.

Valor Establecido	Valor de la Aplicación	Valor de Termómetro	Calefactor	Ventilador	Observación	Prueba
50°C	28°C	27°C	ON	OFF	Calefactor intentará subir la temperatura a 50°C	Exitosa
10°C	28°C	29°C	OFF	ON	Ventilador intentará bajar la temperatura a 10°C	Exitosa
60°C	27°C	28°C	ON	OFF	Calefactor intentará subir la temperatura a 60°C	Exitosa
20°C	27°C	28°C	OFF	ON	Ventilador intentará bajar la temperatura a 20°C	Exitosa
40°C	28°C	28°C	ON	OFF	Calefactor intentará subir la temperatura a 40°C	Exitosa

En la Figura 4.1, se aprecia el valor de la temperatura ambiental dentro del hogar desde la Aplicación Android.

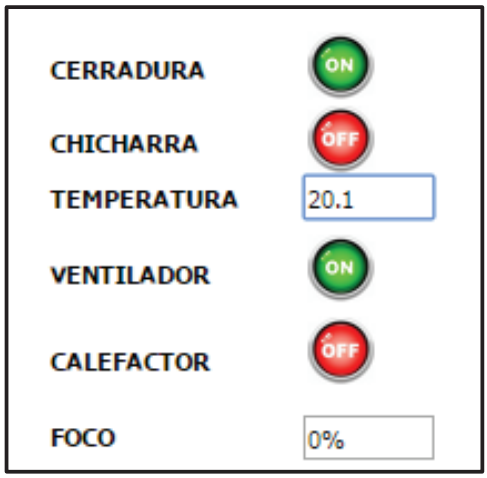


Figura 4.1 Obtención de la temperatura ambiental desde la aplicación android.

En la Figura 4.2 y Figura 4.3 se observa la placa y la implementación final del nodo controlador de temperatura ambiental.

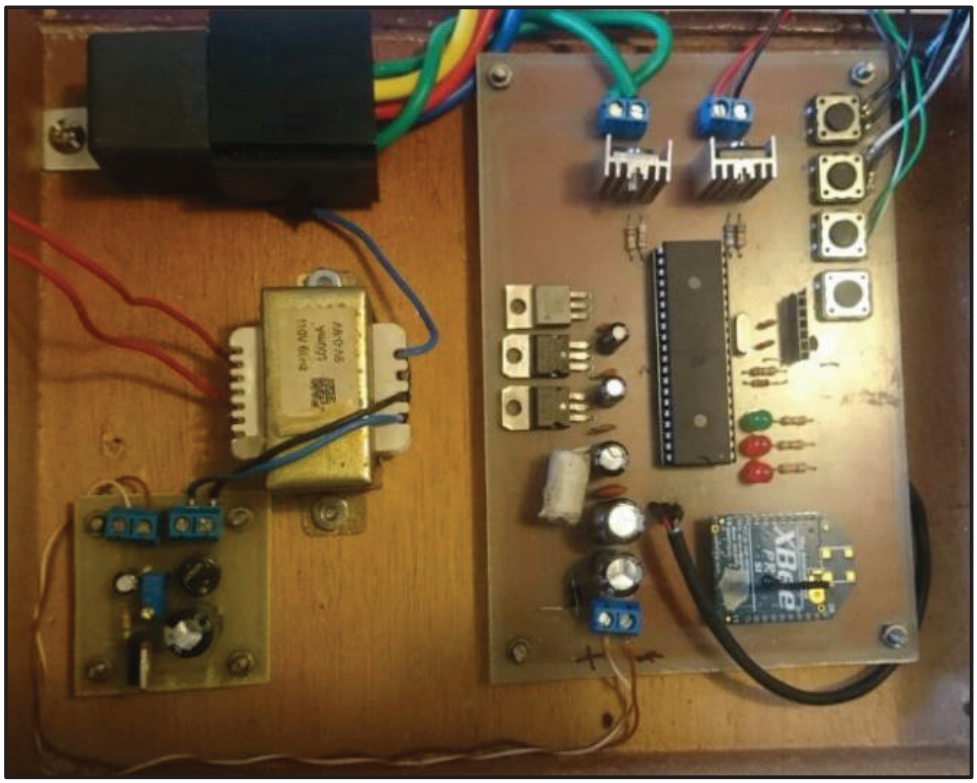


Figura 4.2 Placa del nodo controlador de temperatura ambiental.

Figura 4.3 Nodo controlador de temperatura ambiental.

4.1.2 NODO CORTINA MOTORIZADA

Con respecto a la cortina motorizada no se tuvo ningún inconveniente en la comunicación entre el nodo central y el nodo de dicha cortina. Se puede controlar el motor a través de la aplicación Android o del control manual logrando abrir, cerrar y detener la cortina de forma adecuada. En la Figura 4.4**,** se muestra la placa del nodo cortina motorizada con los distintos elementos que lo conforman.

Figura 4.4 Placa del nodo cortina motorizada.

En la Figura 4.5, se visualiza el motor y los fines de carrera que se implementaron para el nodo cortina motorizada.

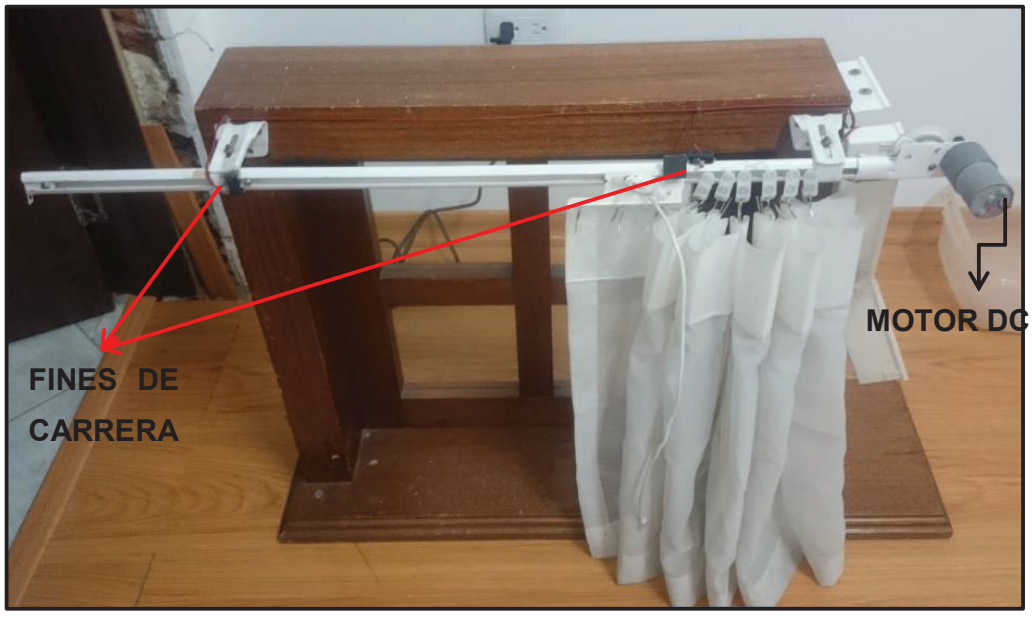


Figura 4.5 Fines de carrera y motor DC del nodo cortina motorizada.

4.1.3 NODO DE CONTROL DE ACCESO

La comunicación inalámbrica entre el nodo central y el nodo de control de acceso se desarrolló sin ningún problema ejecutándose la acción de abrir la puerta y activar el sonido de la chicharra cada vez que el nodo central emitía esta orden.

El control manual de este nodo se realiza mediante la digitación de una clave de acceso previamente almacenada por el usuario, la misma que se puede cambiar cuando requiera mediante el ingreso de una clave de seguridad establecida en la programación del nodo. Al ingresar incorrectamente la clave de acceso por 4 ocasiones, se bloquea el sistema por un minuto. En la Figura 4.6 y Figura 4.7 se observa la placa y la implementación final del nodo control de acceso respectivamente.

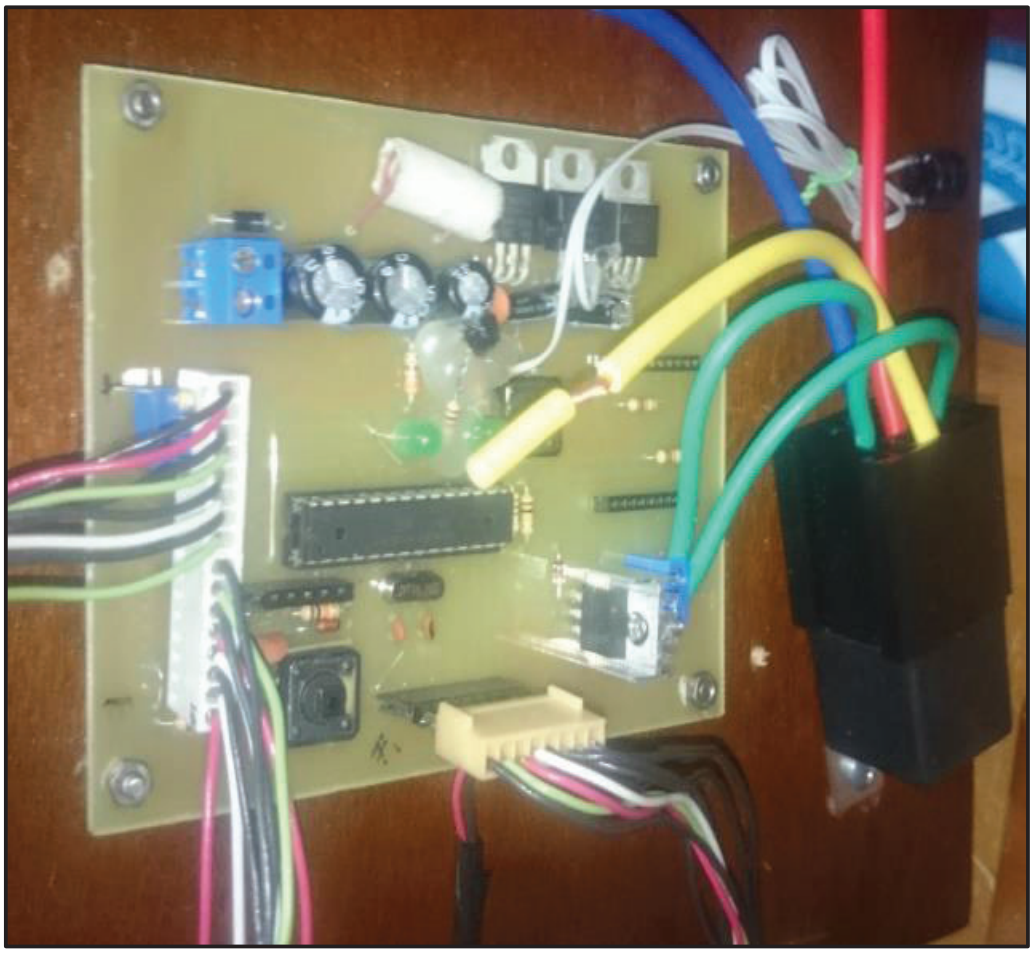


Figura 4.6 Placa del nodo del control de acceso.

Figura 4.7 Nodo control de acceso.

4.1.4 NODO DIMMER.

Para verificar la correcta comunicación entre el nodo central y el nodo del *dimmer* se llevaron a cabo pruebas de encendido, apagado y variación de la intensidad luminosa desde la aplicación Android, obteniéndose una respuesta correcta en cada cambio solicitado.

El control manual del *dimmer* al igual que en la aplicación Android funciona con un tiempo de respuesta de alrededor de 8 segundos en promedio, con lo que se concluye que las necesidades de control y monitoreo para la iluminación de un hogar con una infraestructura de dos pisos es adecuada, siendo una herramienta útil para manipular sectores de iluminación de manera fácil y eficiente. En la

Figura 4.8 y Figura 4.9 se observa la placa y la implementación final del nodo *dimmer* respectivamente.

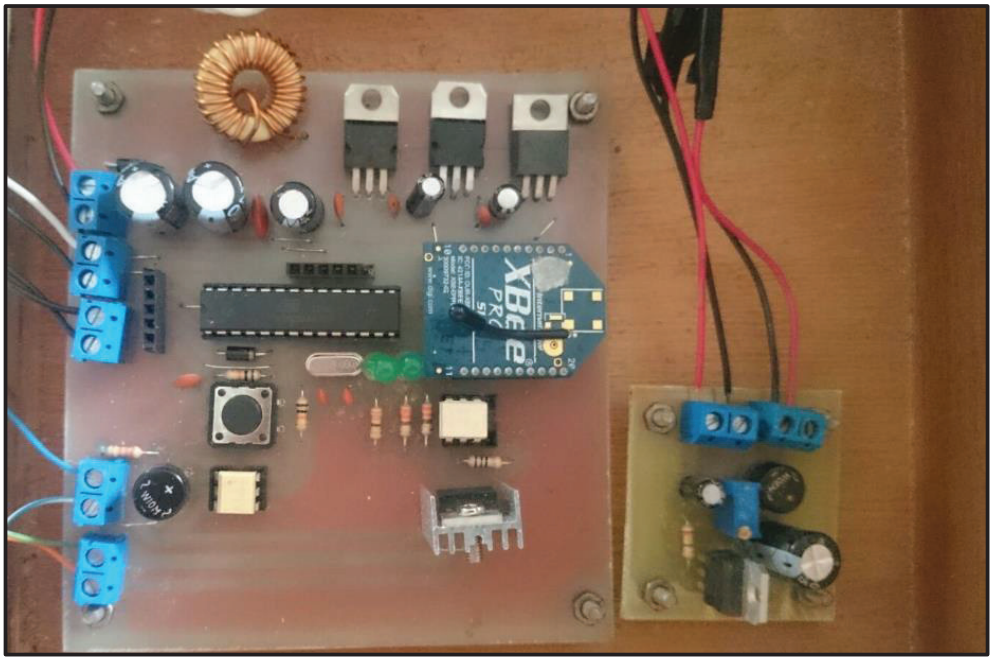


Figura 4.8 Placa final del nodo dimmer.

Figura 4.9 Implementación final nodo dimmer.

4.1.5 NODO CÁMARA

En la comunicación entre el nodo central y el nodo de la cámara se tuvieron inconvenientes con la transmisión del video, debido a que la mínima transmisión requiere una resolución de 426x240 ppp (puntos por pulgada) y una velocidad mínima de 300 kbps.

"A las redes WPAN se las divide a su vez en redes de alta, media y baja velocidad. IEEE802.15.3 es un ejemplo de red de alta velocidad que puede ser usada por ejemplo para transmitir video desde una cámara a un TV cercano. Bluetooth es un ejemplo de estándar de media velocidad. Puede ser usado para transmisión de música de alta calidad desde un equipo de audio a auriculares inalámbricos. También se emplea para conectar teclados, ratones y otros periféricos a computadoras. ZigBee, con una velocidad de datos máxima de 250 kbps, es considerada una red personal inalámbrica de baja velocidad". [23]

En las pruebas que se realizaron se verifico la clasificación mencionada en el párrafo anterior respecto a Zigbee, pues sólo se pudo llegar a obtener una imagen de 320 x240 ppp (puntos por pulgada) a una velocidad de transmisión de 3800bps, la misma que se refresca cada minuto, tiempo que le lleva al sistema en obtener la foto lo cual hace imposible tener un *streaming* de video en tiempo real.

Al realizar las pruebas se tuvo pérdidas considerables de paquetes dentro del sistema domótico y se generan retardos en el sistema debido a la longitud de la imagen que envía el nodo. En la Figura 4.10, se observa un ping extendido a la IP del nodo central generando perdida de paquetes por la imagen transmitida y en la Figura 4.11 se visualiza el terminal donde el nodo central realiza la captura de la imagen.

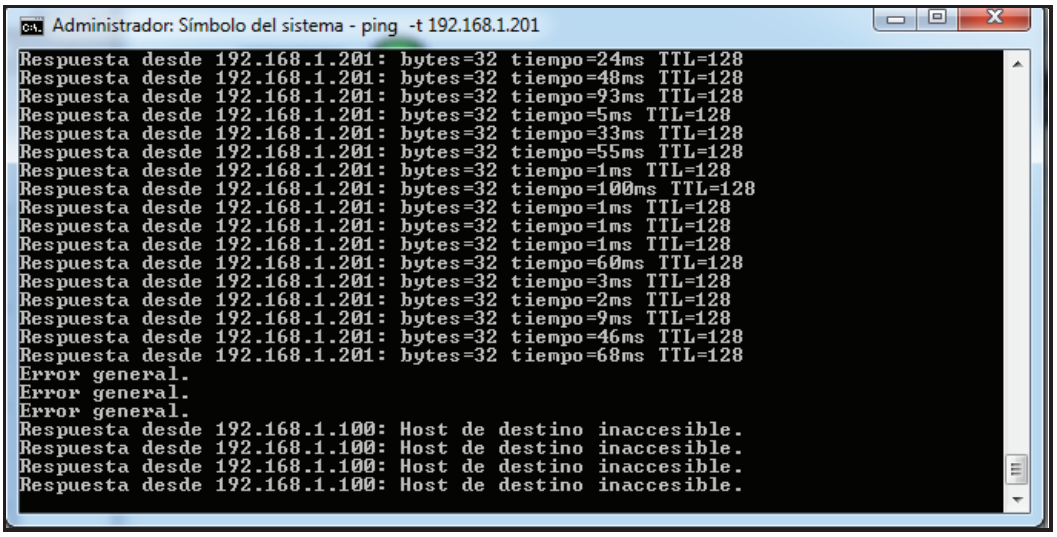


Figura 4.10 Perdida de paquetes por la imagen transmitida.

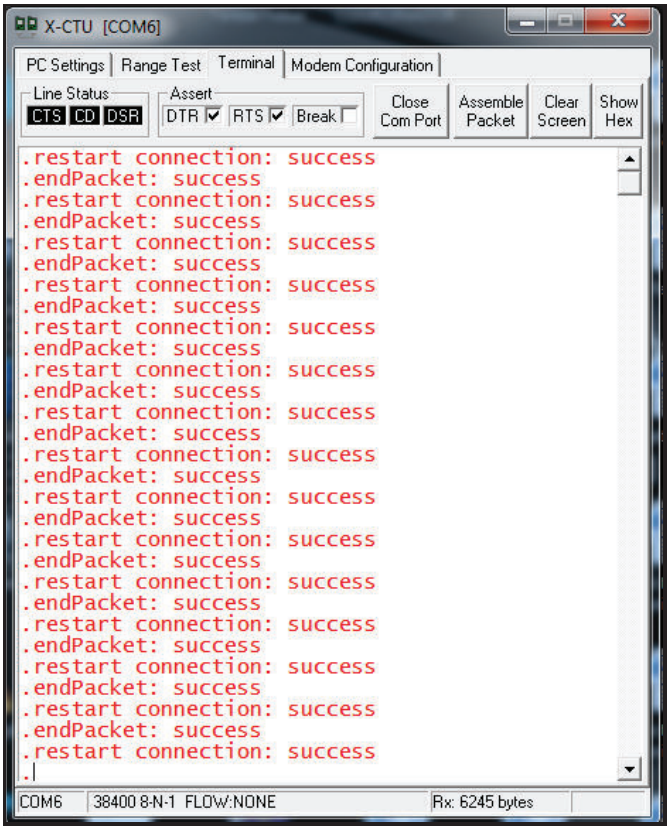


Figura 4.11 Captura de bytes de la imagen transmitida.

Para visualizar la imagen desde la aplicación Android se reenvían los paquetes de datos desde el nodo central hacia un servidor alojado en la nube a través del

protocolo UDP permitiendo así la reconstrucción de la misma. Se optó realizar este procedimiento ya que el procesamiento del nodo central no es suficientemente robusto para reconstruir la imagen. En la Figura 4.11, se observa él envió de paquetes de la imagen a través del protocolo UDP al servidor alojado en la nube.

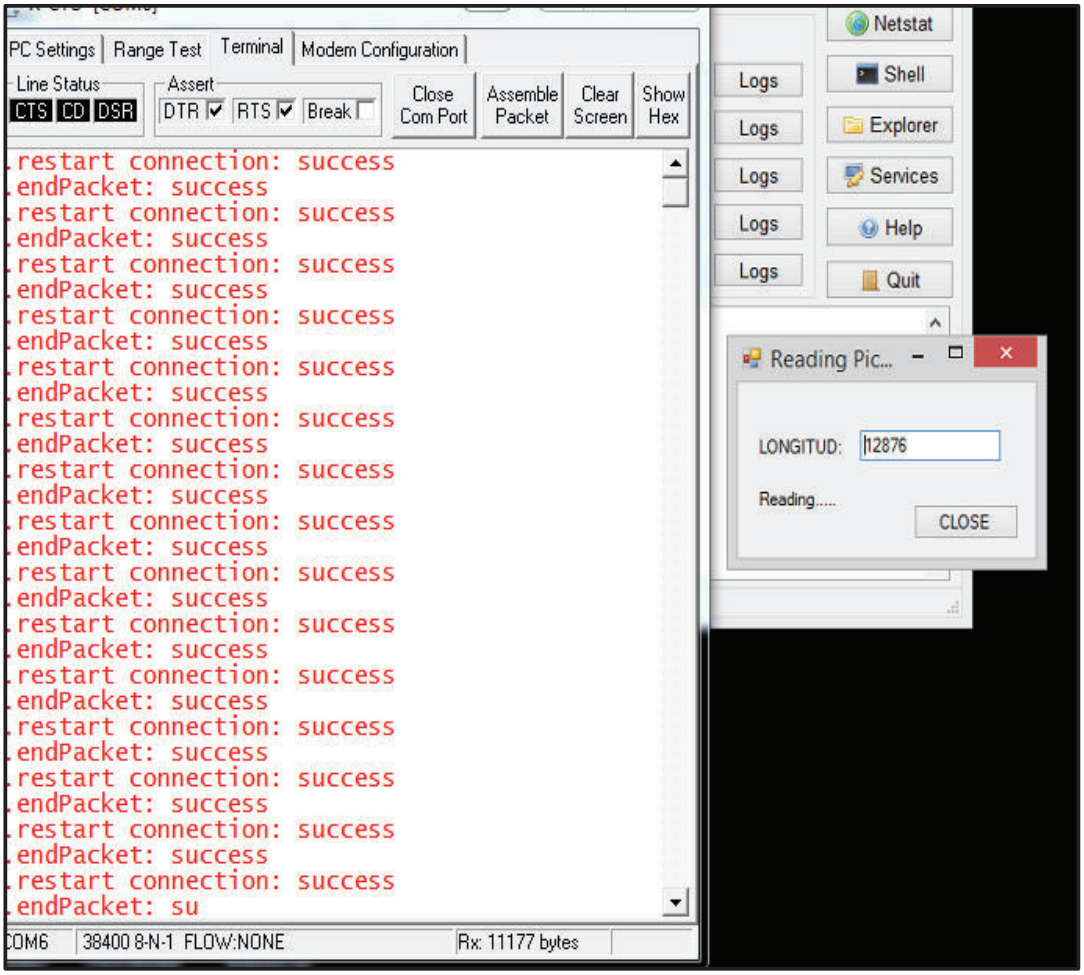


Figura 4.12 Paquete de datos obtenido a través del protocolo UDP.

En la aplicación Android se tiene un zoom que permite ampliar o reducir la imagen en proporción a la pantalla del dispositivo. En la Figura 4.13 y Figura 4.14, se observa la placa y la implementación final del nodo cámara respectivamente.

Figura 4.13 Placa final del nodo cámara.

Figura 4.14 Nodo cámara.

Para evitar pérdidas de información y poder reconstruir la imagen captada por la cámara, se optó por elaborar una red punto - punto entre el nodo central, el nodo de la cámara y una red punto - multipunto con los demás nodos secundarios del sistema.

Se realizaron pruebas de tiempo de respuesta de todo el sistema domótico obteniendo los siguientes resultados que se pueden apreciar en la Tabla 4.2, Tabla 4.3 y la Tabla 4.4.

Tabla 4.2 Primera prueba de retardo del sistema domótico.

Prueba #1					
Dispositivo	**Estado**	**Tiempo Inicio**	**Tiempo Final**	**Duración**	**Observación**
Dimmer	ON	15:05:00	15:06:39	0:01:39	Éxito
Cortina	ON	15:05:00	15:06:39	0:01:39	Éxito
Control de Acceso	ON	15:05:00	15:06:39	0:01:39	Éxito
Chicharra	ON	15:05:00	15:06:39	0:01:39	Éxito
Cámara	CAPTURA	15:05:00	15:06:39	0:01:39	Éxito
Calefactor	ON	15:05:00	15:06:39	0:01:39	Éxito
Ventilador	OFF	15:05:00	15:06:39	0:01:39	Éxito

Tabla 4.3 Segunda prueba de retardo del sistema domótico.

Prueba #2					
Nodo	**Estado**	**Tiempo Inicio**	**Tiempo Final**	**Duración**	**Observación**
Dimmer	20%	15:07:00	15:08:03	0:01:03	Éxito
Cortina	OFF	15:07:00	15:08:03	0:01:03	Éxito
Control de Acceso	ON	15:07:00	15:08:03	0:01:03	Éxito
Chicharra	ON	15:07:00	15:08:03	0:01:03	Fallido
Cámara	CAPTURA	15:07:00	15:08:03	0:01:03	Fallido
Calefactor	OFF	15:05:00	15:06:39	0:01:39	Éxito
Ventilador	ON	15:05:00	15:06:39	0:01:39	Fallido

Tabla 4.4 Tercera prueba de retardo del sistema domótico.

Prueba #3					
Nodo	**Estado**	**Tiempo Inicio**	**Tiempo Final**	**Duración**	**Observación**
Dimmer	60%	15:12:00	15:12:53	0:00:53	Éxito
Cortina	ON	15:12:00	15:12:53	0:00:53	Éxito
Control de Acceso	ON	15:12:00	15:12:53	0:00:53	Éxito
Chicharra	OFF	15:12:00	15:12:53	0:00:53	Éxito
Cámara	CAPTURA	15:12:00	15:12:53	0:00:53	Éxito
Calefactor	OFF	15:05:00	15:06:39	0:01:39	Éxito
Ventilador	OFF	15:05:00	15:06:39	0:01:39	Fallido

De lo anterior se concluye, que se produce un retardo de más de un minuto en ejecutar las peticiones que el usuario requiera debido al continuo refrescamiento de la imagen y a la monitorización del resto de nodos secundarios.

En la Figura 4.15, se observa la imagen que se obtiene de la cámara.

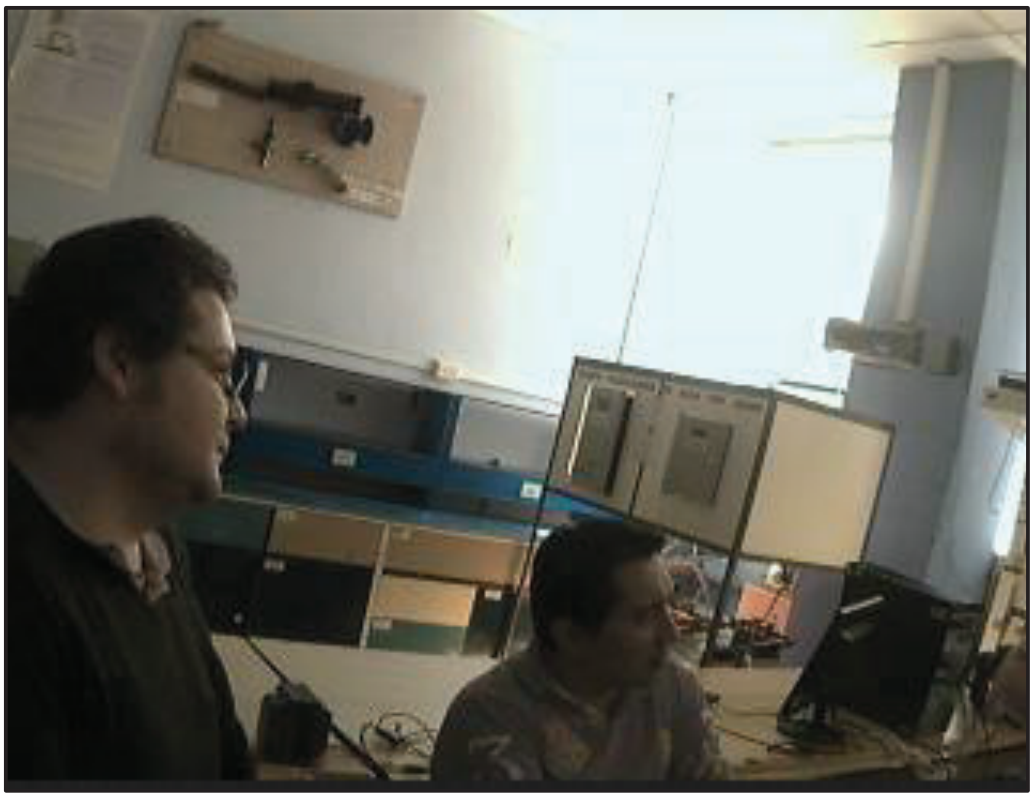


Figura 4.15 Imagen capturada de la cámara.

Debido a todas las limitantes ya expuestas anteriormente se buscó una solución que sea transparente para los usuarios finales pero que satisfaga y optimice los recursos del sistema domótico. La mejor opción para este caso es la selección de una cámara IP que genera una imagen de 640 x480 ppp (puntos por pulgada), ver Figura 4.16, ya que en este dispositivo se mantiene como prioridad la comunicación inalámbrica y el video en tiempo real y los retardos del sistema mejoran considerablemente como se muestra en la Tabla 4.5, Tabla 4.6 y Tabla 4.7.

Tabla 4.5 Primera Prueba de retardo del sistema domótico sin el nodo cámara.

Prueba #1					
Dispositivo	**Estado**	**Tiempo Inicio**	**Tiempo Final**	**Duración**	**Observación**
Dimmer	ON	15:20:00	15:20:08	0:00:08	Éxito
Cortina	ON	15:20:00	15:20:08	0:00:08	Éxito
Control de Acceso	ON	15:20:00	15:20:08	0:00:08	Éxito
Chicharra	ON	15:20:00	15:20:08	0:00:08	Éxito
Calefactor	ON	15:20:00	15:20:08	0:00:08	Éxito
Ventilador	OFF	15:20:00	15:20:08	0:00:08	Éxito

Tabla 4.6 Segunda Prueba de retardo del sistema domótico sin el nodo cámara.

Prueba #2					
Nodo	**Estado**	**Tiempo Inicio**	**Tiempo Final**	**Duración**	**Observación**
Dimmer	20%	15:21:00	15:21:06	0:00:06	Éxito
Cortina	OFF	15:21:00	15:21:06	0:00:06	Éxito
Control de Acceso	ON	15:21:00	15:21:06	0:00:06	Éxito
Chicharra	ON	15:21:00	15:21:06	0:00:06	Éxito
Calefactor	OFF	15:21:00	15:21:06	0:00:06	Éxito
Ventilador	ON	15:21:00	15:21:06	0:00:06	Éxito

Tabla 4.7. Tercera Prueba de retardo del sistema domótico sin el nodo cámara.

Prueba #3					
Nodo	**Estado**	**Tiempo Inicio**	**Tiempo Final**	**Duración**	**Observación**
Dimmer	80%	15:21:30	15:21:37	0:00:07	Éxito
Cortina	ON	15:21:30	15:21:37	0:00:07	Éxito
Control de Acceso	ON	15:21:30	15:21:37	0:00:07	Éxito
Chicharra	ON	15:21:30	15:21:37	0:00:07	Éxito
Calefactor	OFF	15:05:00	15:06:39	0:01:39	Éxito
Ventilador	OFF	15:05:00	15:06:39	0:01:39	Fallido

Figura 4.16 Cámara IP.

En la Figura 4.17, se observa la imagen obtenida a través de la cámara IP.

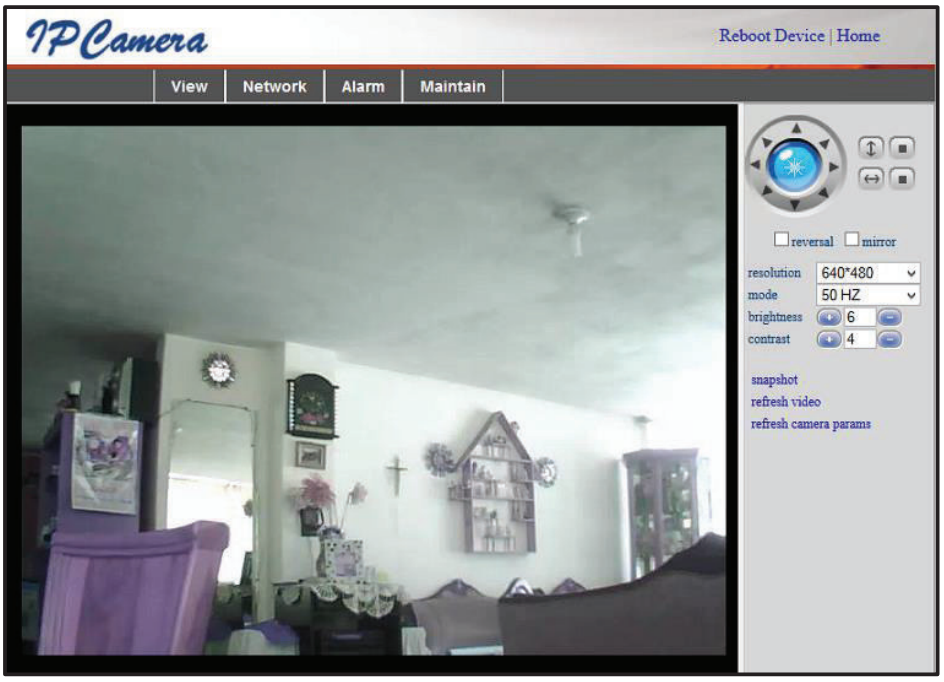


Figura 4.17 Imagen obtenida con la cámara IP.

Como conclusión de las pruebas efectuadas se puede decir que se tiene una mejora en el tiempo de respuesta de 8 segundos del sistema con la incorporación de la imagen obtenida por la cámara IP.

4.1.6 NODO CENTRAL

El control de los módulos secundarios depende del nodo central, el cual realiza sin complicaciones todas las funciones encomendadas. La interacción entre la interfaz del usuario y los dispositivos finales se desarrolló con eficiencia, el tiempo de ejecución de las órdenes es el adecuado para el sistema domótico y la transferencia de información en su conjunto fue exitosa. En la Figura 4.18, se observa la placa y los distintos elementos que conforman el nodo central.

Figura 4.18 Placa del nodo central.

4.2 PRUEBAS DE INTERACTIVIDAD ENTRE USUARIOS Y LA APLICACIÓN DOMÓTICA

Las pruebas realizadas para acceder y controlar el sistema domótico a través de la aplicación Android, consistieron en seleccionar grupos de personas de diferentes edades, con el objetivo de analizar el nivel de afinidad entre usuarios y el uso de la aplicación desarrollada.

En las pruebas se analizaron los parámetros de seguridad, sensores, actuadores y video de la aplicación considerando diferentes aspectos en las perspectivas del usuario, en la Tabla 4.8, Tabla 4.9 y la Tabla 4.10 se muestra los resultados obtenidos considerando una escala donde 1 representa el mínimo y 10 el máximo grado de satisfacción.

El primer grupo se conformó por 10 personas entre edades de 5 a 15 años en la ciudad de Quito.

Tabla 4.8 Promedio de información obtenida del primer grupo.

Parámetro/ Aplicación	Interfaz	Confort	Intuitivo	Útil
Seguridad	4.10	5.5	3.6	5.2
Sensores	6.1	4.3	6.2	2.4
Actuadores	6.3	5.1	7.3	6.1
Video	7.1	7.2	5.1	6.4

El segundo grupo se conformó por 15 personas entre edades de 15 a 45 años en la ciudad de Quito.

Tabla 4.9 Promedio de información obtenida del segundo grupo.

Parámetro/ Aplicación	Interfaz	Confort	Intuitivo	Útil
Seguridad	7.10	8.5	7.4	9.8
Sensores	9.1	8.4	6.2	8.4
Actuadores	6.8	7.4	6.9	8.3
Video	8.1	7.5	8.9	9.3

El tercer grupo se conformó por 10 personas entre edades de 45 a 75 años.

Tabla 4.10 Promedio de información obtenida del tercer grupo.

Parámetro/ Aplicación	Interfaz	Confort	Intuitivo	Útil
Seguridad	5.10	6.5	6.20	6.50
Sensores	1.3	3.3	5.1	4.5
Actuadores	3.7	2.6	3.5	2.1
Video	4.1	4.6	4.9	5.9

En resumen se puede afirmar que los resultados fueron positivos ya que cubre y satisface las necesidades del sistema. Se recomienda que el uso de esta aplicación sea para grupos de personas mayores de 15 años ya que el sistema domótico maneja claves de seguridad para el acceso a la misma. En la Figura 4.19, se puede visualizar la aplicación Android utilizada para los actuadores del sistema domótico.

Figura 4.19 Menú del sistema domótico.

4.3 PRESUPUESTO REFERENCIAL

4.3.1 REFERENCIA DE COSTOS PARA EL NODO CENTRAL

Los elementos del nodo central se pudieron adquirir sin dificultad en las distintas tiendas electrónicas de la ciudad como se puede observar, ver Anexo G. En la Tabla 4.11 se detalla la información del costo de los distintos elementos que se usaron para la construcción del dispositivo.

Tabla 4.11 Costos referentes al nodo central.

DETALLE	CANTIDAD	PRECIO UNITARIO (USD)	TOTAL (USD)
Elementos Electrónicos:			
Resistencia de 10KΩ ¼ W	3	0.02	0.06
Puente rectificador 2W10G	1	0.35	0.35
Capacitor Electrolítico 1000uF	1	0.15	0.15
Circuitos Integrados:			
ARDUINO-MEGA2560	1	33	33
Módulo XBee Pro 60MW WIR	1	62.90	62.90
LM317	1	0.65	0.65
Conectores:			
Cable Arduino	4	0.24	0.96
Bornera TBLOCK-I2	2	0.25	0.50
Espadín hembra simple	1	0.50	0.50
Espadín macho simple	4	0.40	1.60
Zócalo Módulo XBee	1	1.50	2.00
Varios:			
Tuerca	8	0.08	0.64
Perno	8	0.08	0.64
Botón-7MM	1	0.12	0.12
Placa 11x8.5 cm	1	13.10	13.10
Armado	1	8	8
		SUB TOTAL	123.99
		IVA 12%	14.88
		TOTAL (USD)	138.87

4.3.2 REFERENCIA DE COSTOS PARA EL NODO CONTROL DE TEMPERATURA AMBIENTAL

Los elementos electrónicos que tomaron parte en la elaboración del nodo control de temperatura ambiental fueron adquiridos a nivel local, la proforma de estos dispositivos se puede ver en el Anexo G. En la Tabla 4.12 se puede apreciar en detalle la información de costos de implementación del nodo.

Tabla 4.12 Costos referentes al nodo termómetro ambiental.

DETALLE	CANTIDAD	PRECIO UNITARIO ($)	TOTAL ($)
Elementos Electrónicos:			
Disipador	1	0.65	0.65
Puente rectificador de 1.5 AMP	1	0.35	0.35
Resistencia de 1KΩ ¼ W	2	0.02	0.04
Capacitor Electrolítico 470uF	1	0.09	0.09
Diodo 1N4007	2	0.06	0.12
Switch Pulsador GDE Abierto Corto	2	0.80	1.60
Circuitos Integrados:			
Microcontrolador ATMEGA 164p	1	5.80	5.80
Módulo XBee Pro 60MW WIR	1	62.90	62.90
LM35	1	2.81	2.81
LM317	1	0.65	0.65
Conectores:			
Zócalo 6 pines	2	0.08	0.16
Zócalo 28 pines	1	0.12	0.12
Espadín hembra simple	2	0.50	0.50
Bornera TBLOCK-I2	5	0.25	1.25
Zócalo 40 pines	1	0.18	0.18
Zócalo Módulo XBee	1	1.50	1.50
Varios:			
Tuerca	8	0.08	0.64
Perno	8	0.08	0.64
Botón-7MM	1	0.12	0.12
Foco	2	0.90	1.80

Transformador 12V-1A	1	5.36	5.36
Placa 11x10 cm	1	15.04	15.04
Armado	1	8	8.00
		SUB TOTAL	110.32
		IVA 12%	13.24
		TOTAL (USD)	123.56

4.3.3 REFERNCIA DE COSTOS PARA EL NODO CORTINA MOTORIZADA

En la Tabla 4.13 se puede apreciar la información acerca del gasto incurrido para la elaboración del nodo cortina motorizada, ver Anexo G.

Tabla 4.13 Costos referentes al nodo cortina motorizada.

DETALLE	CANTIDAD	PRECIO UNITARIO ($)	TOTAL ($)
Elementos Electrónicos:			
Resistencia de 4.7KΩ ¼ w	4	0.02	0.08
Capacitor Cerámico 0.2uF	2	0.12	0.24
Capacitor Electrolítico 470uF	1	0.09	0.09
Potenciómetro de precisión 5KΩ	1	0.45	0.45
Transistor TIP122	4	0.15	0.60
Circuitos Integrados:			
Microcontrolador ATMEGA 164p	1	5.80	5.80
LM317	1	0.65	0.65
Módulo XBee Pro 60MW WIR	1	62.90	62.90
Conectores:			
Zócalo 40 pines	1	0.18	0.18
Molex 4p	4	0.40	0.40
Zócalo Módulo XBee	1	2.00	2.00
Varios:			
Tuerca de Sujeción	8	0.08	0.64
Perno de Sujeción	8	0.08	0.64
Transformador 12V-1A	1	5.36	5.36

Motor DC	1	33.00	33.00
Botón-7MM	2	0.12	0.24
Placa 15x10 cm	1	16.80	16.80
Espaciadores	8	0.05	0.40
Armado	1	8	8.00
		SUB TOTAL	138.47
		IVA 12%	16.62
		TOTAL (USD)	155.09

4.3.4 REFERENCIA DE COSTOS PARA EL NODO CONTROL DE ACCESO

El costo de los elementos electrónicos utilizados en la construcción del nodo control de acceso ver proformas en el Anexo G, se puede apreciar detalladamente en la Tabla 4.14.

Tabla 4.14 Costos referentes al nodo control de acceso.

DETALLE	CANTIDAD	PRECIO UNITARIO ($)	TOTAL ($)
Elementos Electrónicos:			
Resistencia de 2KΩ ¼ W	1	0.02	0.02
Potenciómetro de precisión 5KΩ	1	0.45	0.45
Buzzer 5v	1	0.65	0.65
Teclado electrónico numérico	1	1.50	1.50
Capacitor Electrolítico 10uF	1	0.09	0.09
LCD_16X2_DIL	1	5.17	5.17
Diodo 1N4007	1	0.06	0.06
Transistor 2N3904	1	0.08	0.08
Puente de diodos 2A/1000V	1	0.27	0.27
Circuitos Integrados:			
Microcontrolador ATMEGA 164p	1	5.80	5.80
LM317	1	0.65	0.65
Módulo XBee Pro 60MW WIR	1	62.90	62.90

Conectores:			
Zócalo 28 pines	1	0.12	0.12
Espadín tipo CONN-SIL5	1	0.55	0.55
Zócalo Módulo XBee	1	1.50	1.50
Varios:			
Tuerca de Sujeción	8	0.08	0.64
Relé-Cerradura	2	0.25	0.50
Perno de Sujeción	8	0.08	0.64
Cerradura Elect. CENTURY	1	22.77	22.77
Transformador 12V-1A	1	5.36	5.36
Botón-7MM	2	0.12	0.24
Placa 12x10 cm	1	16.80	16.80
Espaciadores	8	0.05	0.40
Armado	1	8	8.00
		SUB TOTAL	135.16
		IVA 12%	16.22
		TOTAL (USD)	151,38

4.3.5 REFERENCIA DE COSTOS PARA EL NODO DIMMER.

La información acerca de los precios de los elementos utilizados en la construcción del nodo dimmer ver Anexo G, se detalla en la Tabla 4.15.

Tabla 4.15 Costos referentes al nodo *dimmer.*

DETALLE	CANTIDAD	PRECIO UNITARIO ($)	TOTAL ($)
Elementos Electrónicos:			
Resistencia de 180Ω ¼ W	1	0.02	0.02
Resistencia de 330Ω ¼ W	1	0.02	0.02
Resistencia de 1KΩ ¼ W	1	0.02	0.02
Resistencia de 1.2KΩ ¼ W	1	0.02	0.02
Capacitor Electrolítico 10uF	1	0.09	0.09
Capacitor Electrolítico 470uF	1	0.09	0.09

Transistor 2N3904	1	0.10	0.10
Triac BT137	1	1.00	1.00
Diodo 1N4007	1	0.06	0.06
Puente de diodos 2A/1000V	1	0.27	0.27
Puente de diodos 3A	1	0.27	0.27
Switch Pulsador GDE Abierto Corto	2	0.80	1.60
Circuitos Integrados:			
Optoacoplador 4N35	1	0.50	0.50
Microcontrolador ATMEGA 164p	1	5.80	5.80
Optoacoplador MOC3011	1	1.00	1.00
LM317	1	0.65	0.65
LM7805	1	0.40	0.40
Módulo XBee Pro 60MW WIR	1	62.90	62.90
Conectores:			
Bornera 2 pines	3	0.22	0.66
Bornera 4 pines	1	0.40	0.40
Zócalo 40 pines	1	0.18	0.18
Zócalo Módulo XBee	1	2.00	2.00
Varios:			
Tuerca	8	0.08	0.64
Transformador 12V-1A	1	5.36	5.36
Foco	1	0.90	0.90
Perno	8	0.08	0.64
Armado	1	8.00	8.00
Placa 15x10 cm	1	16.80	16.80
		SUB TOTAL	81.73
		IVA 12%	9.81
		TOTAL (USD)	91.53

4.3.6 REFERENCIA DE COSTOS PARA EL NODO CÁMARA.

La información acerca de los precios de los elementos utilizados en la construcción del nodo Cámara se los detalla en la Tabla 4.16 y en el Anexo G.

Tabla 4.16 Costos referentes al nodo cámara

DETALLE	CANTIDAD	PRECIO UNITARIO ($)	TOTAL ($)
Elementos Electrónicos:			
Cámara IP	1	66.97	66.97
		SUB TOTAL	87.55
		IVA 12%	7.03
		TOTAL (USD)	75.00

4.3.7 COSTOS TOTALES.

La Tabla 4.17 muestra el costo total del prototipo que se diseñó.

Tabla 4.17 Costos referentes al sistema domótico.

NODO	TOTAL (USD)
CENTRAL	138.87
TERMÓMETRO AMBIENTAL	123.56
CORTINA MOTORIZADA	155.09
CONTROL DE ACCESO	151.38
DIMMER	91.53
CÁMARA	75.00
TOTAL	$735.43

Tanto en el mercado nacional como en el internacional existe una gran variedad de sistemas domóticos, cuyos precios varían desde USD 1000 hasta los USD 8000 esta información se detalla en el Anexo G, dependiendo de la cantidad de elementos utilizados, la capacidad y alcance de los módulos inalámbricos además de la complejidad de cada sistema.

Analizando la inversión que se ha hecho y el tiempo asequible en la elaboración del proyecto se puede decir que el prototipo puede competir en precio con muchos de los sistemas domóticos existentes en el mercado sin ninguna dificultad.

En este capítulo se realizaron diversas pruebas al prototipo trabajando en conjunto con todas las partes que lo conforman y también por separado, con la finalidad de justificar el correcto funcionamiento del sistema, económicamente es apto para implementarse en cualquier hogar ya que su inversión no es costosa.

En el siguiente capítulo se recopilará las conclusiones y recomendaciones obtenidas de este proyecto de titulación.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones a las que se ha llegado después de analizar los resultados de las pruebas realizadas en el proyecto.

5.1 CONCLUSIONES:

- Este proyecto nació con la idea de mejorar la calidad de vida de muchas personas que por distintos motivos presentan algún tipo de dificultad de movilidad, es por esta razón que se escogieron los aspectos que podrían causar más dificultad para dichas personas, sin embargo al finalizar el desarrollo del proyecto, se concluyó que existen muchas más actividades como programación del sistema de riego, detección de humo, control de cargas eléctricas, sensor de presencia que se podrían aumentar en el sistema domótico.

- El hecho de que la plataforma Arduino cumpla las características de *Open Source* y *Open Hardware*, la vuelve una alternativa muy interesante para el desarrollo de nuevas aplicaciones y soluciones tecnológicas.

- Por su versatilidad y facilidad en la manipulación, la plataforma Arduino facilita el control de una red domótica con operaciones de control de entrada/salida y automatización de forma rápida e intuitiva.

- El principal limitante que se encontró al desarrollar el sistema domótico con transmisión de video usando el protocolo de comunicación inalámbrica ZigBee, radica en la definición del protocolo el cual fue concebido para uso en aplicaciones de muy baja transmisión de datos y mínimo consumo energético.

- Para mejorar la transmisión de imágenes bajo el protocolo ZigBee fue necesario fragmentar la red, es decir tener una topología punto a punto para la red entre el nodo concentrador y la cámara; y una topología punto - multipunto para la red del nodo concentrador con el resto de actuadores, este switcheo entre topologías se logró variando en cada ciclo de programa las configuraciones de los módulos Xbee, obteniendo como resultado una mejora significativa en calidad de imagen pero a costo de aumentar los tiempos de respuesta y procesamiento del sistema domótico.

- Debido a que no se cumplía con el objetivo planteado inicialmente de construir un sistema domótico que permitiera controlar el acceso en tiempo real, se buscó una alternativa que fuera transparente para el usuario final pero que tuviera la versatilidad de acoplarse al sistema ya implementado, llegando a la conclusión que la mejor alternativa era la de reemplazar la cámara inalámbrica mediante ZigBee por una cámara IP Wifi con movimiento incluído.

- Al realizar pruebas de latencia del sistema domótico, se pudo visualizar que al emplear el protocolo ZigBee para la transmisión de imágenes se produce un retardo de aproximadamente 60 segundos en la ejecución de la petición del usuario, mientras que para el mismo sistema al usar una cámara IP WiFi, el retardo es de aproximadamente de 5 a 8 segundos.

- El diseño y elaboración de las fuentes de alimentación es un factor determinante para alcanzar un óptimo desempeño del sistema domótico, de lo contrario se corre el riesgo de dañar los dispositivos que dependen de ellas.

- La captura de la imagen se realiza a través del protocolo UDP, este protocolo fue empleado para evitar que la tarjeta Arduino procese esta información, ya que el mismo por su baja capacidad de procesamiento y memoria no puede realizar esta operación, es por esta razón que el Arduino es empleado únicamente para el envío de la información.

- Debido a la naturaleza de los sistemas domóticos, no es necesario que las peticiones de los usuario se ejecuten de forma instantánea, razón por la cual el retardo ingresado por la red celular, de aproximadamente 4 segundos; no se considera como una limitante en el rendimiento.

- *App Inventor* es un *software* que utiliza una interfaz gráfica muy sencilla que permite a los programadores sean expertos o novatos, ir "armando" sus aplicaciones a manera de rompecabezas y probarlas en tiempo real directamente en el dispositivo Android.

- Para manejar de mejor manera la información generada y recopilada por los nodos secundarios fue necesaria la implementación de un servidor en la nube el cual, además de recolectar los estados de los nodos secundarios, recolecta las solicitudes ingresadas por el usuario, las procesa y se encarga de almacenarla, para que cuando sea necesario las ejecute.

5.2 RECOMENDACIONES:

De la experiencia adquirida durante la ejecución de este proyecto se puede extraer las siguientes recomendaciones.

- Cuando se vaya a trabajar con transmisión de video en tiempo real, se recomienda no hacerlo bajo el protocolo de comunicación inalámbrica ZigBee, ya que por su bajo consumo de energía, bajo ancho de banda y baja tasa de transferencia presenta dificultades de implementación.

- Se debe procurar ubicar al nodo central cerca de un punto de red, de manera que se facilite el acceso a la conexión de Internet.

- Es muy importante mantener el voltaje recomendado evitando las fluctuaciones tanto para la tarjeta Arduino como para los módulos XBee,

ya que estas en algún momento podrían causar daños irreversibles en los dispositivos.

- El prototipo domótico implementado puede mejorarse, no sólo variando su tecnología sino también implementando nuevos nodos secundarios que a futuro permitan tener el control total de una vivienda.

- Se recomienda elaborar diagramas de flujo antes de iniciar con la programación, ya que estos facilitan la comprensión, edición y depuración de los programas.

- Cuando se trabaja por primera vez con la plataforma Arduino es necesario realizar pequeños y muy variados programas que faciliten el entendimiento de la estructura de programación aunque en gran medida presenta similitud con el lenguaje C++.

- Cuando se trabaja con un solo nodo coordinador y varios nodos secundarios a la vez, es recomendable iniciar probando cada uno de los secundarios con el coordinador, una vez que se tiene la certeza que se encuentran operando sin ningún tipo de error, la integración en el mismo sistema se torna mucho más fácil.

- Se debe considerar que el trabajar con motores, relés, transformadores y cualquier otro tipo de elementos electro-mecánicos conlleva una posibilidad de introducir ruido al circuito, lo que puede causar que el sistema no trabaje de la mejor manera.

REFERENCIAS BIBLIOGRÁFICAS

[1] Hugo Martín Domínguez y Fernando Sáez Vacas, (2006). Domótica un Enfoque Sociotécnico, Centro De Domótica Integral, Universidad Politécnica de Madrid, [Online], Disponible: http://www.gsi.dit.upm.es/~fsaez/intl/libro_domotica.pdf

[2] Vitoria/Gasteiz, Universidad del País Vasco (UPV/EHU), DOMOTICA, Expresión gráfica y proyectos de ingeniería, [Online], Disponible: http://www.ehu.eus/alfredomartinezargote/tema_4_archivos/domotica/Transparenciasdomotica.pdf

[3] LA NACIÓN, Domótica para el nuevo milenio, [Online], Disponible: http://www.lanacion.com.ar/207456-domotica-para-el-nuevo-milenio

[4] Todo sobre edificios inteligentes, CASADOMO, [Online], Disponible: http://www.casadomo.com/noticiasDetalle.aspx?c=14

[5] DOMOTICA, 04 de noviembre de 2009, [Online], Disponible: http://es.slideshare.net/guestb42134/domotica-2423605?next_slideshow=1,

[6] Análisis Comparativo de las placas Arduino (oficiales y compatibles), [Online], Disponible: http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/

[7] Emilio Lledó Sánchez, Diseño de un Sistema de Control Domótico basado en la plataforma Arduino. Ingeniería Técnica en Informática de Sistemas, Universidad Politécnica de Valencia,]Disponible: https://riunet.upv.es/bitstream/handle/10251/18228/Memoria.pdf?sequence=1,

[8] Arduino, Disponible: http://arduino.cc

[9] Android y sus nombres de postres, [Online], Disponible: http://www.taringa.net/post/offtopic/18543376/Android-y-sus-nombres-de-postres.html,

[10] Universidad Politécnica de Valencia , Arquitectura de Android, [Online], Disponible:http://www.androidcurso.com/index.php/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android,

[11] Esteban Reinoso P. Diseño e implementación de un prototipo para una red de domótica y seguridad para un hogar utilizando el estándar IEEE 802.15.4

“ZIGBEE". Facultad de Ingeniería Eléctrica y Electrónica, Escuela politécnica Nacional.

[12] Universidad Técnica Particular de Loja, 16 de abril del 2011, ¿QUE ES EL IEEE?, [Online], Disponible: http://ieee.utpl.edu.ec/que-es-el-ieee

[13] ZIGBEE Y EL ESTÁNDAR IEEE 802.15.4, [Online], Disponible: http://www.dea.icai.upco.es/sadot/Comunicaciones/avanzadas/Alberto_Gasc%C3%B3n_Zigbee%20y%20el%20Est%C3%A1ndar%20IEEE%20802.15.4.pdf,

[14] Jordi Mayné, SILICA, Rev. 0, IEEE 802.15.4 y ZigBee, [Online], Disponible: http://www.bairesrobotics.com.ar/data/ieee_zigbee_silica.pdf

[15] Gustavo Alexander Álvarez Álvarez, Standard ZigBee, Escuela Académico Profesional de Informática, UNIVERSIDAD NACIONAL DE TRUJILLO, [Online], Disponible: http://es.scribd.com/doc/19049146/El-Standard-ZigBee

[16] A fondo ZIGBEE, DOMODESK todo en Domótica inmódica y control, El Zumbido de las Abejas ZIGBEE, [Online], Disponible: http://www.domodesk.com/content.aspx?co=97&t=21&c=47

[17] Leandro Pazmiño O. Ricardo Sánchez G. (2010) Diseño y construcción de un prototipo inalámbrico RF para el monitoreo de la seguridad residencial de forma remota empleando la Internet. Facultad de Ingeniería Eléctrica y Electrónica, Escuela politécnica Nacional.

[18] BRANDO, ¿Qué es la nube, para qué sirve y cuáles son los servicios que tenéis que conocer?, [Online], Disponible: http://www.conexionbrando.com/1389864-que-es-la-nube-para-que-sirve-y-cuales-son-los-servicios-que-tenes-que-conocer

[19] ATmega2560-Arduino Pin Mapping, [Online], Disponible: http://www.arduino.cc/en/Hacking/PinMapping2560

[20] Micro Robotics, CAMERA 2MP TTL SERIAL INTERFACE, [Online], Disponible: https://www.robotics.org.za/index.php?route=product/product&product_id=93

[21] ElecFreaks, Freaduino Mega2560, [Online], Disponible: http://www.elecfreaks.com/wiki/index.php?title=Freaduino_Mega2560

[22] DiyMakers, RASPBERRY PI COMO SERVIDOR WEB, (07 de agosto de 2014), [Online], Disponible: http://diymakers.es/raspberry-pi-como-servidor-web/

[23] Jorge Pablo Dignani, (2011), ANÁLISIS DEL PROTOCOLO ZIGBEE, Facultad de Informática, Universidad Nacional de La Plata, [Online], Disponible: http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Dignanni_Jorge_Pablo.pdf

ANEXOS

ANEXO A.

MANUAL PARA MÓDULOS XBEE PRO SERIE 1.

A.1 Circuito básico para el XBee.

La Figura1 muestra las conexiones mínimas que necesita el módulo XBee para poder ser utilizado. Luego de esto, se debe configurar según el modo de operación que se desea para aplicación requerida por el usuario.

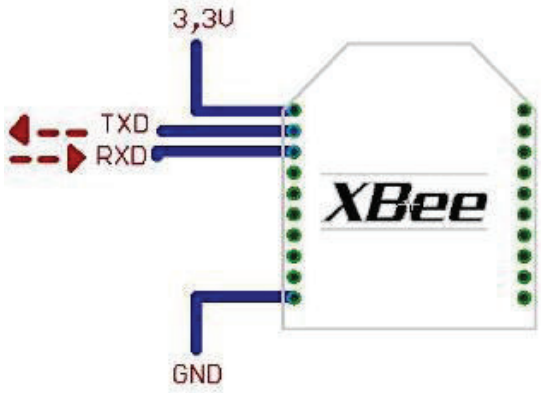


Figura. 1 Conexiones mínimas requeridas para el XBEE.

El módulo requiere una alimentación desde 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del UART (TXD y RXD) para comunicarse con un microcontrolador, o directamente a un puerto serial utilizando algún conversor adecuado para los niveles de voltaje.

Esta configuración, no permite el uso de Control de Flujo (RTS & CTS), por lo que esta opción debe estar desactivada en el terminal y en el módulo XBEE. En caso de que se envíe una gran cantidad de información, el buffer del módulo se puede sobrepasar.

Para evitar existen dos alternativas:

- Bajar la tasa de transmisión
- Activar el control de flujo.

A.2 DISTRIBUCIÓN DE PINES DEL MÓDULO XBEP24-AWI-00.

PIN	DENOMINACIÓN	DESCRIPCIÓN
1	VCC	Fuente de alimentación
2	DOUT	UART salida de datos
3	$\mathrm{DIN}/\overline{\mathrm{CONFIG}}$	UART entrada de datos
4	DO8*	Salida digital 8
5	RESET	Pin para reset, este pulso debe ser menor a 200ns
6	PWMO/RSSI	Salida PWM0 o Indicador de nivel de Señal Rx
7	[reserved]	Sin conexión
8	[reserved]	Sin conexión
9	$\overline{\mathrm{DTR}}/\mathrm{SLEEP\_RQ}/\mathrm{DI8}$	Control de modo dormido o entrada digital 8
10	GND	Tierra
11	AD4/DIO4	Entrada Analógica 4 o I/O Digital 4
12	$\overline{\mathrm{CTS}}/\mathrm{DIO7}$	Control de Flujo Clear to Send o I/O Digital 7
13	ON/SLEEP	Indicador de estado de módulo
14	VREF	Voltaje de Referencia para entradas AD
15	Associate/AD5/DIO5	Pin asociado, Entrada Analógica 5 o I/O 5 Digital
16	RTS/AD6/DIO6	Control de Flujo Request to Send, Entrada Analógica 6 o I/O Digital 6
17	AD3/DIO3	Entrada Analógica 3 o I/O Digital 3
18	AD2/DIO2	Entrada Analógica 2 o I/O Digital 2
19	AD1/DIO1	Entrada Analógica 1 o I/O Digital 1
20	AD0/DIO0	Entrada Analógica 0 o I/O Digital 0

A.3 Modos de Operación.

Los módulos XBee, pueden operar en los siguientes 5 modos:

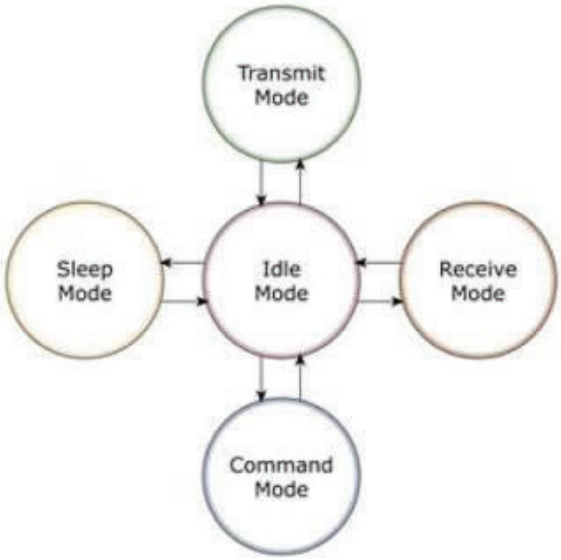


Figura 2 Modos de operación del módulo XBee.

A.3.1 MODO RECIBIR/TRANSMITIR.

Se encuentra en estos modos cuando al módulo le llega algún paquete RF a través de la antena(modo Receive) o cuando se manda información serial al buffer del pin 3 (UART Data in) que luego será transmitida (modo Transmit).
La información transmitida puede ser Directa o Indirecta. En el modo directo la información se envía inmediatamente a la dirección de destino. En el modo Indirecto la información es retenida durante un período de tiempo y es enviada sólo cuando la dirección de destino la solicita.

Además es posible enviar información por dos modos. Unicast y Broadcast. Por el primero, la comunicación es desde un punto a otro, y es el único modo que permite respuesta de quien recibe el paquete RF, es decir, quien recibe debe enviar un ACK (paquete llamado así, y que indica que recibió el paquete, el usuario no puede verlo, es interno de los módulos) a la dirección de origen. Quien envió el paquete, espera recibir un ACK, en caso de que no le llegue, reenviará el paquete hasta 3 veces o hasta que reciba el ACK. En el modo Broadcast lacomunicación es entre un nodo y a todos los nodos de la red. En este modo, no hay confirmaciónpor ACK.

A.3.2 Modo de Bajo Consumo (Sleep Mode).

El modo de sueño hace posible que el módulo RF entre en un modo de bajo consumo de energía cuando no se encuentra en uso.
Para poder entrar en modo de sueño, se debe cumplir una de las siguientes condiciones:

- Sleep_RQ(pin 9) está en alto y el módulo está en pin sleep mode (SM= 1,2 o 5)
- El módulo está en reposo (no hay transmisión ni recepción de datos) por la cantidad de tiempo definido por ST (Time before Sleep). [ST sólo está activado cuando SM=4,5]

La configuración de los ciclos de sueño se realiza principalmente con el comando SM. Por defecto, los modos de sueños están deshabilitados (SM=0), permaneciendo el módulo en estado de reposo/recepción. En este estado el módulo está siempre preparado para responder a un comando, ya sea, por el puerto serial o la interfaz RF.

Modos de sueño controlado por Pin.

Pin de Hibernación.

Este modo minimiza el consumo de energía cuando el módulo se encuentra en reposo. Este modo se habilita cuando Sleep_RQ (pin 9) está en alto, el módulo terminará cualquier transmisión, recepción o procedimientos de asociación y entrará en modo de reposo y luego en modo de sueño. En este estado el módulo no responderá a comandos entrantes, ya sea, desde la interfaz serial como RF.Cuando se baja el estado lógico de Sleep_RQ (pin 9) el módulo saldrá del modo de sueño y estará listo para recibir o enviar datos.

Pin Doze.

Este modo funciona de la misma forma que el modo Pin de Hibernación, sin embargo, Pin Doze presenta un tiempo de activación menor y mayor consumo de energía.

Para despertar un módulo operando en modo Pin Doze, se debe bajar Sleep_RQ (pin9) y éste comenzará a transmitir o recibir información cuando la línea CTS este en nivel lógico bajo.

Modos de sueño cíclicos.

Sueño cíclico remoto (SM = 4).

El modo de sueño cíclico remoto permite que el módulo revise la data por la interfaz RF periódicamente. Cuando el parámetro SM es seteado a 4, el módulo se configura para efectuar ciclos de sueño, luego, despierta una vez por ciclo para revisar si existen datos en el coordinador de sueño de la red (SM = 0, CE = 1). El módulo remoto, envía esta solicitud al coordinador a intervalos de tiempo determinados por el parámetro ST (Periodo de dormido). El coordinador transmitirá los datos que se puedan encontrar en su buffer de salida al módulo remoto una vez recibida la solicitud de datos.

En el caso que el coordinador no tenga datos para ser enviados al módulo que realiza la solicitud, el coordinador no transmitirá y el módulo remoto retornará a su estado de sueño. Si existen datos para ser enviados al módulo solicitante, el coordinador se quedará despierto y transmitiendo hasta el Timer ST (Tiempo antes de dormir) se complete.

Sueño cíclico remoto y Pin para despertar (SM = 5).

Este modo se utiliza para despertar un módulo remoto, ya sea por la interfaz RF o por poner en estado bajo el pin Sleep_RQ utilizado para comunicación orientada a eventos. El sueño cíclico funciona de la misma forma que el modo de sueño cíclico remoto con la funcionalidad extra de poder despertar el módulo utilizando un pin.

Cualquier actividad limpiará la cuenta de ST(Tiempo antes de dormir), de esta manera el módulo sólo volverá a dormir luego que no exista alguna actividad durante el periodo de ST. Si el módulo despierta por cambios en el pin Sleep_RQ, los posteriores cambios serán ignorados durante su funcionamiento.

Coordinador de sueño.

Este modo configura al módulo para funcionar como coordinador de sueño. El coordinador acepta mensajes a un módulo específico con direcciones de 16 o 64 bit y los mantiene en su buffer interno hasta que los módulos remotos despiertan y solicitan datos al coordinador. El parámetro SP del coordinador debe ser establecido con el mismo valor que los módulos remotos, para que se pueda producir la comunicación entre los ciclos de sueño.

Resumen.

A continuación se presenta una tabla resumen, considerando los modos de operación, consumos, voltajes y condiciones para entrar en modo de bajo consumo.

Modo	Consumo alimentación			Modo Sleep	Modo Wake -up
	2.8 – 3 V	3.2 V	3.4V		
SM=1	<3 uA	32 uA	255 uA	Sleep_RQ	Sleep_RQ
SM=2	<35 uA	48 uA	170 uA	Sleep_RQ	Sleep_RQ
SM=3		(reservado)		(reservado)	(reservado)
SM=4	<34 uA	49 uA	240 uA	Comando ST	Comando SP
SM=5	<34 uA	49 uA	240 uA	Comando ST	Sleep RQ

Tabla 1 .Tabla Modo Sleep y consumos de corriente.

A.3.3 Modo de Comando.

Este modo permite ingresar comandos AT al módulo Xbee, para configurar, ajustar o modificar parámetros. Permite ajustar parámetros como la dirección propia o la de destino, así como su modo de operación entre otras cosas. Para poder ingresar los comandos AT es necesario utilizar el Hyperterminal de

Windows, el programa X-CTU o algún microcontrolador que maneje UART y tenga los comandos guardados en memoria o los adquiera de alguna otra forma.
Para ingresar a este modo se debe esperar un tiempo dado por el comando GT (Guard Time, por defecto ATGT=0x3E8 que equivalen a 1000ms) luego ingresar +++ y luego esperar otro tiempo GT. Como respuesta el módulo entregará un OK.
El módulo XBee viene por defecto con una velocidad de 9600bps. En caso de no poder ingresar al modo de comandos, es posible que sea debido a la diferencia de velocidades entre el módulo y la interfaz que se comunica vía serial.

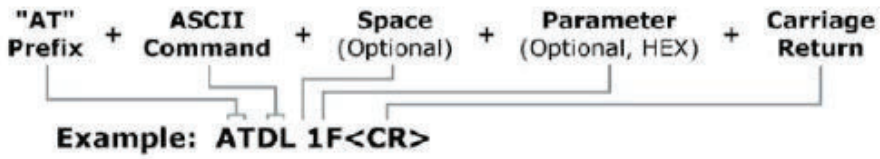


Figura 3 Ejemplo Comando AT.

En la Figura 3 se observa la sintaxis de un comando AT. Luego de ingresar a este modo, se debe ingresar el comando deseado para ajustar los parámetros del módulo XBEE. La lista de comandos de encuentra en las siguientes secciones.
Por ejemplo si se desea modificar la dirección de origen del módulo de 16-bit (con el comando MY), se debe ingresar:

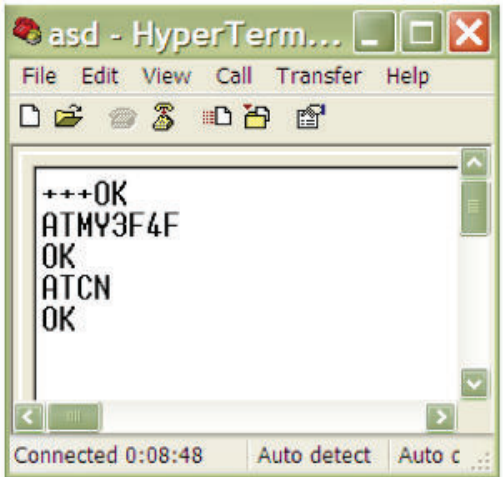


Figura 4. Ejemplo escritura de parámetros.

En el ejemplo anterior, la dirección asignada equivale a 0x3F4F. Se observa que primero se ingresa al modo de comandos AT, recibiendo un OK de respuesta. Luego se ingresa el comando ATMY3F4F y se presiona ENTER o carácter <CR><LF> (Carrier Return y Line Feed) si se maneja desde un microcontrolador. Con ello se recibe un<CR><LF>OK<CR><LF>como respuesta.

Para salir del modo de Comandos se ingresa ATCN y se presiona ENTER. En caso de que no se ingrese ningún comando AT válido durante el tiempo determinado por CT (Command Mode Timeout), el módulo se saldrá automáticamente. Para que los cambios realizados tengan efecto se debe ingresar ATCN (sale del modo de comandos) o ATAC (aplica los cambios inmediatamente). Con el comando ATWR, se guardan los cambios en la memoria no volátil del módulo, pero sólo tendrán efecto una vez ingresado el comando AC o CN.

Para consultar por algún comando se ingresa el comando directamente sin ningún parámetro, en el ejemplo anterior se ingresaría ATMY. La siguiente figura muestra lo anterior:

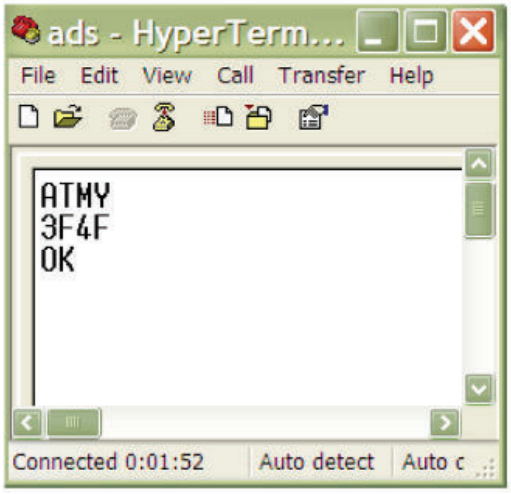


Figura 5. Ejemplo lectura de parámetros.

Se observa que se ingresa ATMY, donde el que módulo responde con un 3F4F, que es la dirección que tiene configurada, y luego con un OK. Otra forma de

configurar el módulo por comando AT, es ingresando varios comandos separados por coma (´,´). Esto se muestra a continuación de las dos formas:

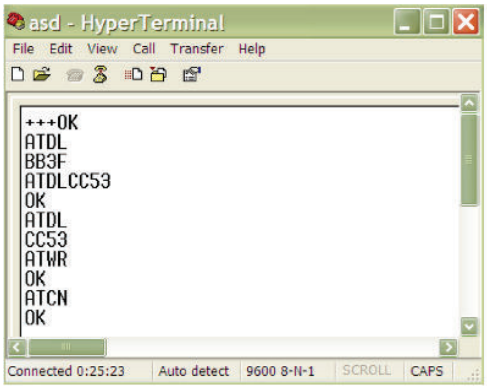


Figura 6. Forma normal de escribir.

Se observa que luego de ingresar al modo de Comandos (+++), se pregunta por la dirección de destino (ATDL), para lo cual se tiene como respuesta la dirección 0xBB3F. Con ello se modifica esta dirección por 0xCC53 ingresando ATDLCC53, obteniendo un OK como respuesta.

Se confirma el correcto ajuste preguntando nuevamente y luego se guarda la configuración en la memoria no volátil del módulo usando ATWR, para lo cual se vuelve a obtener un OK. Posteriormente se saldrá del modo de comando utilizando ATCN obteniendo otro OK. Otra forma de hacer lo mismo se muestra en la siguiente figura:

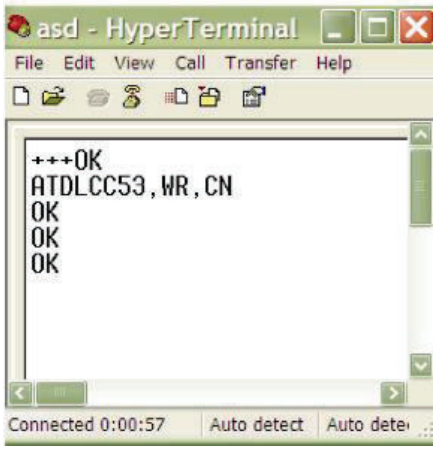


Figura 7. Forma abreviada.

Se observa que luego de ingresar al modo de comandos, se ingresa ATDLCC35, WR, CN que indica que se ingresarán tres comandos en uno, y éstos serán ATDLCC35, ATWR y ATCN.
Luego se obtendrá la respuesta para cada comando en el mismo orden en que fueron ingresados.
Además de los modos anteriores, existen otros modos referentes a lo que son la forma de transmitir la información. Estos modos son: Modo Transparente y Modo API.

A.3.4 Modo Transparente.

En este modo todo lo que ingresa por el pin 3 (Data in), es guardado en el buffer de entrada y luego transmitido y todo lo que ingresa como paquete RF, es guardado en el buffer de salida y luego enviado por el pin 2 (Data out). El modo Transparente viene por defecto en los módulos XBee.

Este modo está destinado principalmente a la comunicación punto a punto, donde no es necesario ningún tipo de control. También se usa para reemplazar alguna conexión serial por cable, ya que es la configuración más sencilla posible y no requiere una mayor configuración.

En este modo, la información es recibida por el pin 3 del módulo XBee, y guardada en el buffer de entrada. Dependiendo de cómo se configure el comando RO, se puede transmitir la información apenas llegue un carácter (RO=0) o después de un tiempo dado sin recibir ningún carácter serial por el pin 3. En ese momento, se toma lo que se tenga en el buffer de entrada, se empaqueta, es decir, se integra a un paquete RF, y se transmite. Otra condición que puede cumplirse para la transmisión es cuando el buffer de entrada se llena, esto es, más de 100 bytes de información.

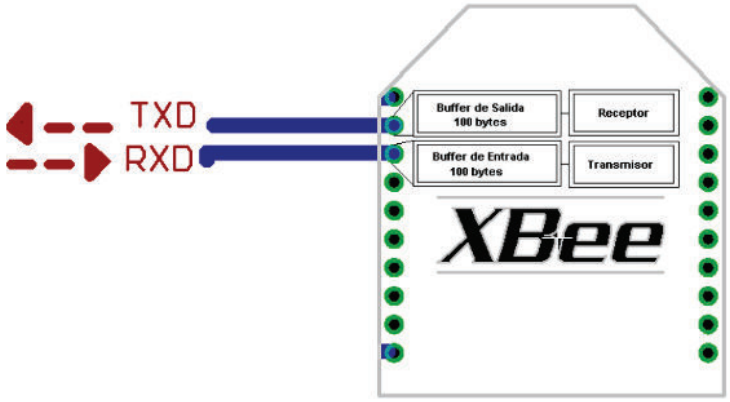


Figura 8. Correspondencia de Buffer.

A.3.5 Modo de operación API.

Este modo es más complejo, pero permite el uso de frames con cabeceras que aseguran la entrega de los datos, al estilo TCP. Extiende el nivel en el cual la aplicación del cliente, puede interactuar con las capacidades de red del módulo. Cuando el módulo XBEE se encuentra en este modo, toda la información que entra y sale, es empaquetada en frames, que definen operaciones y eventos dentro del módulo.

Así, un Frame de Transmisión de Información (información recibida por el pin 3 o DIN) incluye:

- Frame de información RF transmitida.
- Frame de comandos (equivalente a comandos AT).
- Mientras que un Frame de Recepción de Información incluye:
- Frame de información RF recibida.
- Comando de respuesta.
- Notificaciones de eventos como Reset, Associate, Disassociate, etc.

Esta API, provee alternativas para la configuración del módulo y ruteo de la información en la capa de aplicación del cliente. Un cliente puede enviar

información al módulo XBee. Estos datos serán contenidos en un frame cuya cabecera tendrá información útil referente el módulo.

Esta información además se podrá configurar, esto es, en vez de estar usando el modo de comandos para modificar las direcciones, la API lo realiza automáticamente. El módulo así enviará paquetes de datos contenidos en frames a otros módulos de destino, con información a sus respectivas aplicaciones, conteniendo paquetes de estado, así como el origen, RSSI (potencia de la señal de recepción) e información de la carga útil de los paquetes recibidos.
Entre las opciones que permite la API, se tienen:

- Transmitir información a múltiples destinatarios, sin entrar al modo de Comandos.
- Recibir estado de éxito/falla de cada paquete RF transmitido.
- Identificar la dirección de origen de cada paquete recibido.

A.3.6 Idle.

Cuando el módulo no se está en ninguno de los otros modos, se encuentra en éste. Es decir, si no está ni transmitiendo ni recibiendo, ni ahorrando energía ni en el modo de comandos, entonces se dice que se encuentra en un estado al que se le llama IDLE.

A.4 Direccionamiento de los módulos.

Los módulos permiten 2 tipos de direccionamiento. La de 16 bit y la de 64 bits. La principal diferencia es que en la de 64 bit, es posible obtener una mayor cantidad de direcciones y por lo tanto, una mayor cantidad de nodos o equipos funcionando en la misma red. Son a través de estas direcciones que los módulos se comunican entre sí.

A.4.1 Direccionamiento de 16 bit.

El comando MY, define un número de 16 bit como dirección del módulo dentro de la red. El rango se encuentra entre 0x0 y 0xFFFE (la dirección 0xFFFF y 0xFFFE son para habilitar la dirección de 64-bit, por lo que si se desea utilizar direccionamiento de 16 bits, estos valores no deben ser usados). Para definirla se ingresa ATMY y el número en formato hexadecimal, pero sin el „0x". Por ejemplo si a un módulo se le quiere asignar la dirección 0x3BF1 (15345 en decimal), entonces se debe ingresar el comando ATMY3BF1.

El comando DL, permite definir un número de 16 bit como dirección del módulo de destino dentro de la red al cual se va a realizar la comunicación. El rango debe estar entre 0x0 y 0xFFFE (las direcciones 0xFFFE y 0xFFFF se utilizan para direccionamiento de 64 bits).

Así para habilitar el direccionamiento de 16 bit, se debe utilizar una dirección menor a 0xFFFE con el comando MY, de igual modo para DL y se debe dejar en cero el comando DH=0 (ATDH0). No se permite usar la dirección 0xFFFE ni 0xFFFE para el direccionamiento de 16.

A.4.2 Direccionamiento de 64 bits.

El número 0xFFFF y 0xFFFE del comando MY, se usa cuando se desea desactivar el direccionamiento de 16 bit, y se habilita el uso de la dirección de 64 bit. Con este direccionamiento ya no es posible definir la dirección de origen del módulo, ya que ésta se asigna automáticamente. En este caso, la dirección del módulo corresponde a su número serial, que viene de fábrica y el cual es imposible de cambiar. Este número se encuentra guardado en dos variables de 32 bit cada una (SL y SH) y es único. SL lee los 32 bit menos significativos del número serial y SH los 32 más significativos.

Cuando se utiliza direccionamiento de 64 bit, para asignar una dirección de destino, se utilizan los comandos DL y DH. Éstos son de 32 bit cada uno (para el direccionamiento de 16 bit, DL se maneja como uno de 16, mientras que DH se mantiene en cero) y juntos (DL+DH) forman el número de 64 bit que debe corresponder con el número serial de otro módulo formado por SL+SH. Así para algún dato, DL debe ser igual a SL y DH debe ser igual a SH, donde SL+SH corresponden al número serial de un módulo destino configurado para direccionamiento de 64 bits.

Para el direccionamiento de 64 bit, se debe dejar MY como 0xFFFF (ATMYFFFF) o 0xFFFE (ATMYFFFE) y elegir una dirección de destino usando DL+DH, que debe corresponder a una dirección de 64 bit de otro módulo, indicando su número serial dado por SL+SH. Para consultar este número se debe ingresar ATSL (32 bit menos significativos) y luego ATSH (32 bit más significativos), entregando como respuesta los números seriales en formato hexadecimal.

ANEXO B.

CARACTERÍSTICAS DEL MICROCONTROLADOR 164P.

B.1 CARACTERÍSTICAS.

Arquitectura Avanzada RISC.

- 131 instrucciones. La mayoría de un ciclo de reloj de ejecución.
- 32 registros de trabajo de 8 bits para propósito general.
- Funcionamiento estático total.
- Capacidad de procesamiento de unos 20 MIPS a 20 MHz.
- Multiplicador por hardware de 2 ciclos.

Memorias de programa y de datos no volátiles de alta duración.

- 16/32/44 K bytes de FLASH auto programable en sistema.
- 512B/1K/2K bytes de EEPROM.
- 1/2/4K bytes de SRAM Interna.
- Ciclos de escritura/borrado: 10.000 en Flash / 100.000 en EEPROM.
- Retención de Datos: 20 años a 85ºC / 100 años a 25ºC.
- Sección opcional de código Boot con bits de bloqueo independientes.
- Programación en sistema del programa Boot que se encuentra dentro del mismo chip. Operación de lectura durante la escritura.
- Bloqueo programable para la seguridad del software.

Interfase JTAG.

- Capacidades de Boundary Scan de acuerdo con el estándar JTAG.
- Soporte Extendido Debug dentro del chip.

- Programación de FLASH, EEPROM, fusibles y bits de bloqueo a través de la interfase JTAG.

Características de los periféricos.

- Dos Timer/Contadores de 8 bits con prescalamiento separado y modo comparación.
- Un Timer/Contador de 16 bits con prescalamiento separado, modo comparación y modo de captura.
- Contador en Tiempo Real con Oscilador separado.
- 6 Canales para PWM.
- ADC de 10 bits y 8 canales.
- Modo Diferencial con ganancia seleccionable a x1, x10 o x200.
- Interface serie de dos hilos con byte orientado.
- Dos puertos Seriales USART Programables.
- Interfaz Serial SPI maestro-esclavo.
- Watchdog Timer programable con oscilador independiente, dentro del mismo chip.
- Comparador Analógico dentro del mismo Chip.

Características especiales del microcontrolador.

- Power-on Reset (en el encendido) y detección de Brown-out (pérdida de polarización) programable.
- Oscilador RC interno calibrado.
- Fuentes de interrupción externas e internas.
- 6 modos de descanso: Idle, Reducción de Ruido ADC, Power-save, Power-down, Standby y Standby extendido.
- Encapsulados para Entradas/Salidas (E/S)
- 32 líneas de E/S programables.
- PDIP de 40 pines, TQFP y QFN/MLF de 44 pines.

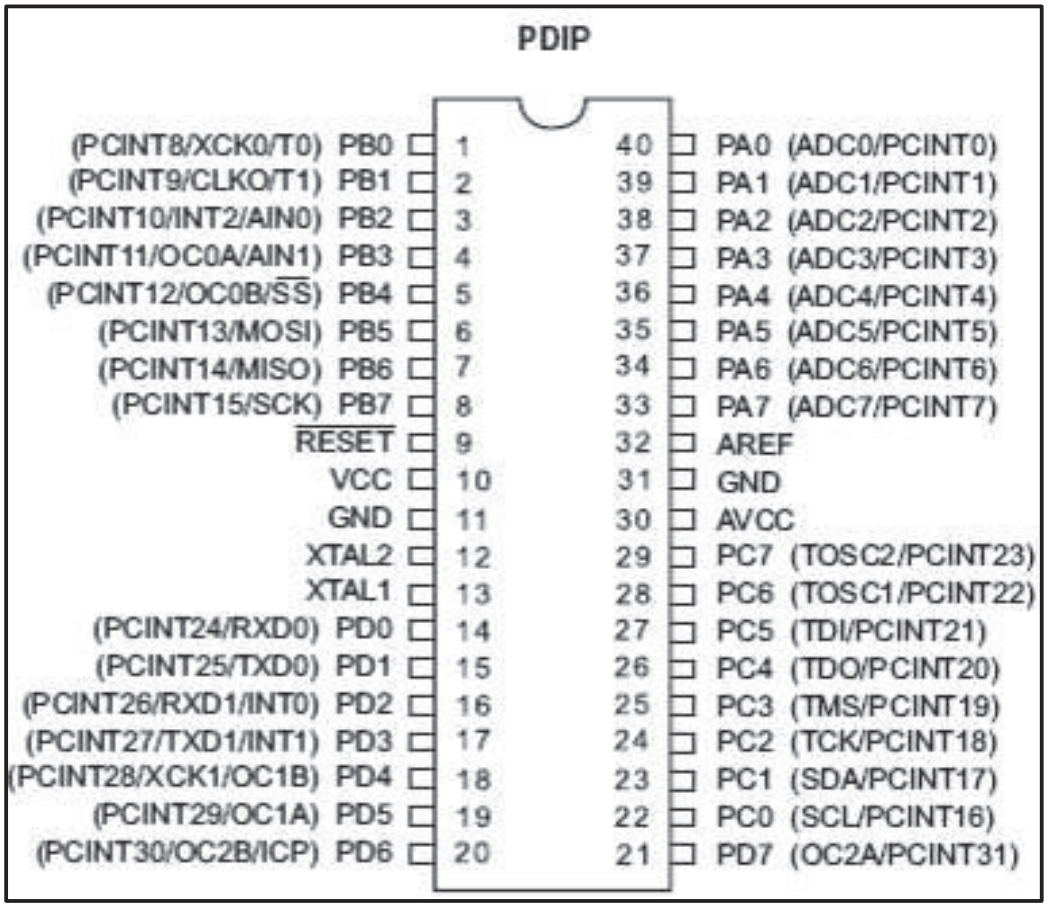
Voltajes de Operación.

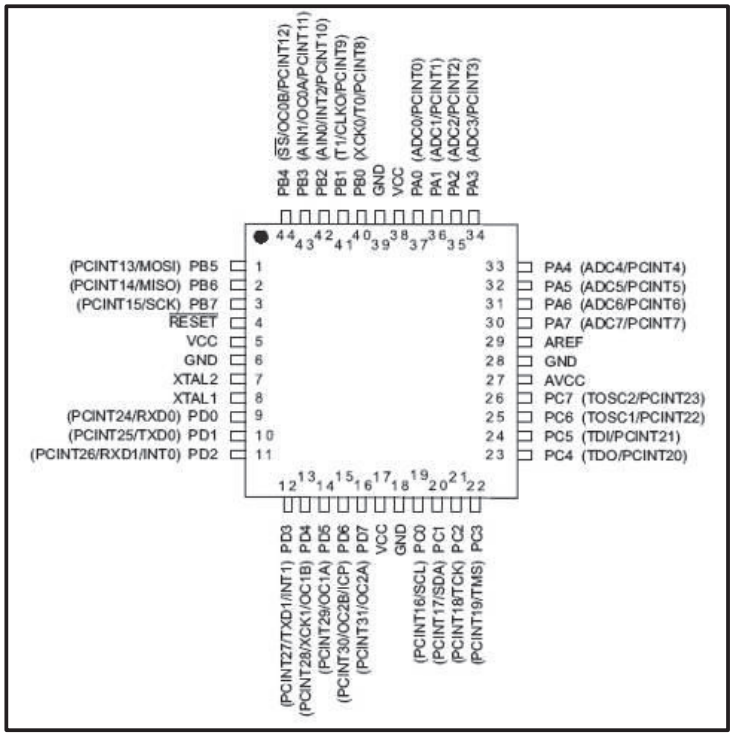
- 1.8 – 5.5V para el ATMEGA 164P/324P/644PV.
- 2.7 – 5.5V para el ATMEGA 164P/324P/644P.

Velocidad de Funcionamiento.

- ATMEGA 164P/324P/644PV: 0 – 4MHz @1.8–5.5V-10MHz @ 2.7 – 5.5V.
- ATMEGA 164P/324P/644PV: 0 – 10MHz @ 2.7 – 5.5V - 20MHz @ 4.5 5.5V.

B.2 CONFIGURACIÓN DE PINES.





B.3 Descripción de Pines.

VCC.

Alimentación de Voltaje Digital

GND.

Tierra

Puerto A (PA7:PA0).

El puerto A sirve como entradas analógicas para el conversor Análogo Digital.

El puerto A también sirve como un puerto bidireccional de 8 bits con resistencias internas de pull up (seleccionables para cada bit). Los buffers de salida del puerto A tienen características simétricas controladas con fuentes de alta capacidad.

Los pines del puerto A están en tri-estado cuando las condiciones de reset están activadas o cuando el reloj no este corriendo. El puerto A también sirve para varias funciones especiales del ATmega164P como la Conversión Análoga Digital.

Port B (PB7:PB0).

El puerto B es un puerto bidireccional de 8 bits de E/S con resistencias internas de pull up. Las salidas de los buffers del puerto B tienen características simétricas controladas con fuentes de alta capacidad.

Los pines del puesto B están en tri-estado cuando las condiciones de reset están activadas o cuando el reloj no esté corriendo. El puerto B también sirve para varias funciones especiales del ATmega164P/324P/644P como se menciona en las páginas iniciales.

Port C (PC7:PC0).

El puerto C es un puerto bidireccional de 8 bits de E/S con resistencias internas de pull up (seleccionadas por cada bit). Las salidas de los buffers del puerto C tienen características simétricas controladas con fuentes de alta capacidad. Los pines del puesto C están en tri-estado cuando las condiciones de reset están activadas siempre y cuando el reloj no este corriendo. El puerto C también sirve para las funciones de Interfaz del JTAG, con funciones especiales del ATmega164P/324P/644P como se menciona en las páginas iniciales.

Port D (PD7:PD0).

El Puerto D es un puerto bidireccional de entradas y salidas con resistencias internas de pull up (seleccionadas por cada bit). Las salidas de los buffers del puerto D tienen características simétricas controladas con sumideros de fuentes de alta capacidad.Los pines del Puerto D están en tri-estado cuando llega una condición de reset activa, siempre y cuando el reloj no esté corriendo.

El puerto D tambiénsirve para varias funciones especiales del ATmega164P como se menciona en las páginas iniciales.

RESET.

Entrada del Reset. Un pulso de nivel bajo en este pin por períodos de pulso mínimo genera un reset, siempre y cuando el reloj no esté corriendo. Pulsos cortos no son garantizados para generar un reset.

XTAL1.

Entrada para el amplificador del oscilador invertido y entrada para el circuito de operación del reloj interno.

XTAL2.

Salida del Oscilador amplificador de salida.

AVCC.

AVCC es la alimentación de voltaje para el pin del Puerto F y el Conversor Análogo a Digital. Este debe ser conectado externamente a VCC, siempre y cuando el ADC no sea usado. Si el ADC es usado, este deberá ser conectado a VCC a través de un filtro paso bajo.

AREF.

Esta es la referencia para el pin de la conversión Análoga a Digital.

ANEXO C.

CÁLCULO DE RESISTENCIA LIMITADORA PARA LED´S

Los diodos LED son seguramente la forma más popular de señalización de estados en los equipos electrónicos. Veremos la manera de sacar todo el provecho posible a estos coloridos componentes.

Para que un LED funcione, necesitamos que una corriente lo atraviese. La intensidad de esta corriente debe ser cuidadosamente calculada, dado que si excedemos los límites especificados en la hoja de datos del componente, este se destruirá. La lista siguiente nos da una idea de que tensión aproximada necesita la juntura de los LEDs de colores comunes para funcionar.

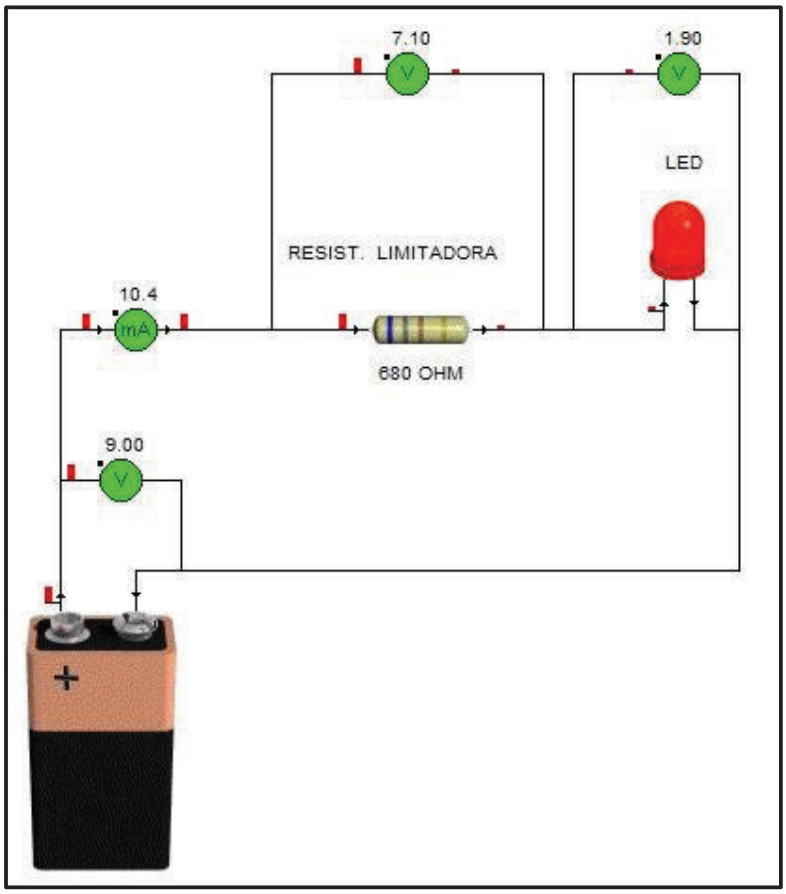
COLOR DEL LED	TENSIÓN UMBRAL
Rojo	1,6V
Rojo alta luminosidad	1,9V
Amarillo	1,7V a 2V
Verde	2,4V
Naranja	2,4V
Blanco brillante	3,4V
Azul	3,4V
Azul 430nm	4,6V

La corriente que debe atravesar el LED depende del tipo de componente pero se puede generalizar un valor entre 10mA y 30mA.
Correspondiendo el primer valor a la corriente mínima para que encienda, con vida útil muy larga y el segundo valor a la corriente máxima que soporta, con vida útil bastante más corta. Con una corriente mayor a ésta el componente se inutiliza

Por lo general se toma un valor medio de 20mA (0,02A) con el cuál se logra un buen brillo con una vida útil larga.

Para limitar la corriente que circula por el LED, lo más usual, es colocar un

resistor en serie con él.



Una resistencia en serie con el LED limita la corriente que lo atraviesa.

El cálculo del valor de esta resistencia es muy sencillo, ya que implica el uso de la ley de ohm. Debemos restar la tensión del LED a la tensión de la fuente, y dividir el resultado por la corriente que deseamos atraviese el componente. Si usamos las unidades correctas (tensiones en Volts y corrientes en Amperes), el resultado estará expresado en Ohms.

A menudo es necesario colocar dos o más LEDs en serie, entre sí. En ese caso, debemos asegurarnos que todos funcionan con la misma corriente, para evitar que alguno resulte dañado. Luego, simplemente reemplazamos en la formula que hemos visto el valor de V(LED) por la suma de las tensiones de cada uno de los LEDs implicados. Por supuesto, el valor de esta suma no debe ser mayor a la tensión suministrada por la fuente.

Las resistencias sólo se pueden conseguir en los siguientes valores comerciales.

x 1	x 10	x 100	x 1.000 (K)	x 10.000 (10K)	x 100.000 (100K)	x 1.000.000 (M)
1 Ω	10 Ω	100 Ω	1 KΩ	10 KΩ	100 KΩ	1 M Ω
1,2 Ω	12 Ω	120 Ω	1K2 Ω	12 KΩ	120 KΩ	1M2 Ω
1,5 Ω	15 Ω	150 Ω	1K5 Ω	15 KΩ	150 KΩ	1M5 Ω
1,8 Ω	18 Ω	180 Ω	1K8 Ω	18 KΩ	180 KΩ	1M8 Ω
2,2 Ω	22 Ω	220 Ω	2K2 Ω	22 KΩ	220 KΩ	2M2 Ω
2,7 Ω	27 Ω	270 Ω	2K7 Ω	27 KΩ	270 KΩ	2M7 Ω
3,3 Ω	33 Ω	330 Ω	3K3 Ω	33 KΩ	330 KΩ	3M3 Ω
3,9 Ω	39 Ω	390 Ω	3K9 Ω	39 KΩ	390 KΩ	3M9 Ω
4,7 Ω	47 Ω	470 Ω	4K7 Ω	47 KΩ	470 KΩ	4M7 Ω
5,1 Ω	51 Ω	510 Ω	5K1 Ω	51 KΩ	510 KΩ	5M1 Ω
5,6 Ω	56 Ω	560 Ω	5K6 Ω	56 KΩ	560 KΩ	5M6 Ω
6,8 Ω	68 Ω	680 Ω	6K8 Ω	68 KΩ	680 KΩ	6M8 Ω
8,2 Ω	82 Ω	820 Ω	8K2 Ω	82 KΩ	820 KΩ	8M2 Ω
						10M Ω

Recuerda: Si al calcular el valor de una resistencia te sale un valor que no es comercialmente accesible, debes elegir la más próxima, por arriba o abajo del valor deseado dependiendo de lo que busques en tu circuito. Por ejemplo, si se trata de proteger un Led, podemos seleccionar el valor más cercano por encima del calculado para protegerlo, en cambio si se trata de polarizar un transistor debemos seleccionar el valor más cercano por debajo del calculado para asegurarnos que el transistor entre en conducción.

ANEXO D

CIRCUITO INTEGRADO 7805

D.1. CARACTERÍSTICAS DEL CIRCUITO INTEGRADO 7805.

La serie de KA78XX/KA78XXApositivode tres terminal es regulador están disponibles en el paqueteyTO-220/D-PAK con varias tensiones de salida fija, que los hace útiles en una amplia gama de aplicaciones. Cada tipo emplea interno limitación de corriente, térmico operativo cerrado y seguro zona protección, por lo que es esencialmente indestructible. Si es adecuada disipación de calor se proporciona, se pueden entregar través dela salida1 Amperio actual. Aunque diseñado principalmente como voltaje fijo reguladores, estos dispositivos pueden ser utilizados con externo componentes para obtener voltajes y corrientes ajustables.

Entre las características principales se tiene:

- Salida de corriente hasta 1 Amperio.
- Tensiones de salida de5, 6, 8, 9, 10, 12, 15, 18, 24 Voltios.
- Protección de sobrecarga térmica.
- Protección contra cortocircuitos.
- Salida de transistor Safe Operating Área de Protección.

D.1.1 Nominales absolutos máximos.

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 (KA78XX/A/R)	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

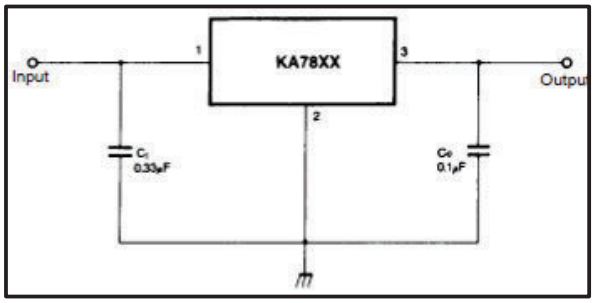
D.1.2 Características Eléctricas.

Referencia para prueba del circuito, 0 ° C <TJ <125°C,IO =500mA,VI =10V,IC =0.33µF, CO = 0.1µF, a menos que se especifique lo contrario)

Parameter	Symbol	Conditions		KA7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 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	VDrop	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	rO	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

D.2 Aplicaciones Típicas.

Parámetro DC.



ANEXO E

CIRCUITO INTEGRADO LM317

E.1. CARACTERÍSTICAS DEL CIRCUITO INTEGRADO LM317.

El LM317 es un regulador de tensión ajustable de 3 terminales, capaz de suministrar 1,5 amperios en condiciones normales, en un rango de salida que va desde de 1,2 voltios hasta 37 voltios. Son muy fáciles de utilizar y requieren sólo dos resistencias externas para conseguir el valor de tensión de salida deseado. Además, la línea de carga y regulación son mejores que el estándar de los reguladores fijos.

Además de las mejores características respecto a los reguladores fijos, dispone de protección por limitación de corriente y exceso de temperatura, siendo funcional la protección por sobrecarga, incluso si el terminal de regulación está desconectado. Normalmente no necesita condensadores mientras esté a menos de 15 centímetros de los filtros de alimentación. Dado que es un regulador flotante y únicamente ve la entrada a la salida del voltaje diferencial, se puede utilizar para regular altas tensiones mientras no se supere el diferencial de entrada/salida de 40 voltios.

Para aplicaciones que requieren una mayor corriente de salida, se puede usar la serie LM150 (3 amperios) y la serie LM138 (5 amperios).

En resumen las características principales de este circuito integrado son:

- Corriente de salida de hasta 1.5 amperios.
- Tensión de salida desde 1.2 voltios hasta 37 voltios.
- Tolerancia del voltaje de salida del 1%.
- La limitación de corriente no depende de la temperatura.
- Protección contra el ruido a la entrada (RR= 80dB).

- Puede trabajar a temperaturas altas (125ºC).
- Tensión nominal (Vref) de 1.25 voltios

E.1.2 CARACTERISTICAS ELÉCTRICAS

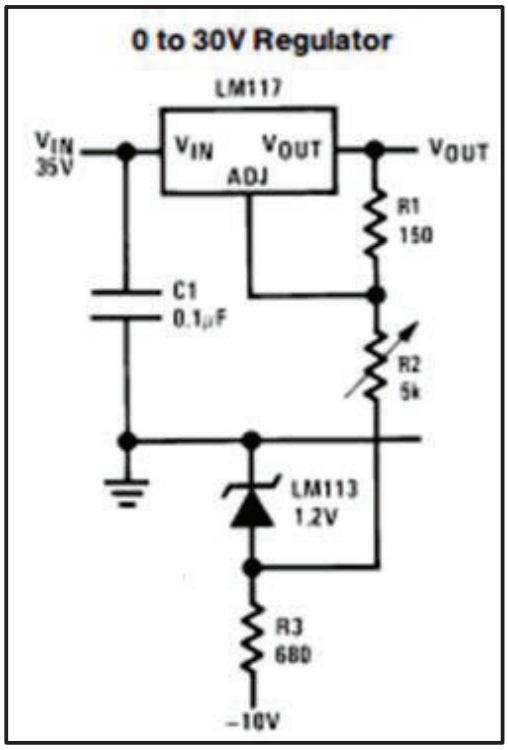
PARAMETER	TEST CONDITIONS[1]			MIN	TYP	MAX	UNIT
Line regulation[2]	$V_I - V_O$ = 3 V to 40 V		T_J = 25°C		0.01	0.04	%/V
			T_J = 0°C to 125°C		0.02	0.07	
Load regulation	I_O = 10 mA to 1500 mA	C_{ADJ}[3] = 10 µF, T_J = 25°C	$V_O \le 5$ V			25	mV
			$V_O \ge 5$ V		0.1	0.5	$\%V_O$
		T_J = 0°C to 125°C	$V_O \le 5$ V		20	70	mV
			$V_O \ge 5$ V		0.3	1.5	$\%V_O$
Thermal regulation	20-ms pulse,	T_J = 25°C			0.03	0.07	$\%V_O$/W
ADJUST terminal current					50	100	µA
Change in ADJUST terminal current	$V_I - V_O$ = 2.5 V to 40 V, $P_D \le 20$ W, I_O = 10 mA to 1500 mA				0.2	5	µA
Reference voltage	$V_I - V_O$ = 3 V to 40 V, $P_D \le 20$ W, I_O = 10 mA to 1500 mA			1.2	1.25	1.3	V
Output-voltage temperature stability	T_J = 0°C to 125°C				0.7		$\%V_O$
Minimum load current to maintain regulation	$V_I - V_O$ = 40 V				3.5	10	mA
Maximum output current	$V_I - V_O \le 15$ V,	$P_D < P_{MAX}$[4]		1.5	2.2		A
	$V_I - V_O \le 40$ V,	$P_D < P_{MAX}$[4],	T_J = 25°C	0.15	0.4		
RMS output noise voltage (% of V_O)	f = 10 Hz to 10 kHz,	T_J = 25°C			0.003		$\%V_O$
Ripple rejection	V_O = 10 V,	f = 120 Hz	C_{ADJ} = 0 µF[3]		57		dB
			C_{ADJ} = 10 µF[3]	62	64		
Long-term stability	T_J = 25°C				0.3	1	%/1k hr

E1.3 CONDICIONES RECOMENDADAS DE OPERACIÓN.

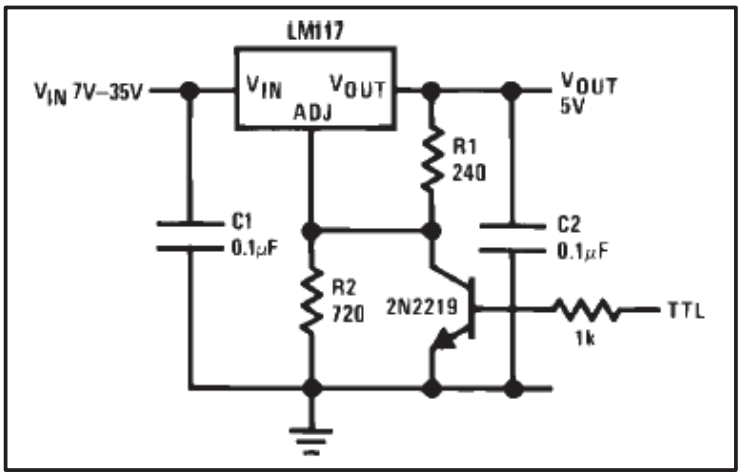
		MIN	MAX	UNIT
V_O	Output voltage	1.25	7	V
$V_I - V_O$	Input-to-output differential voltage	3	40	V
I_O	Output current		1.5	A
T_J	Operating virtual junction temperature	0	125	°C

E 1.4 APLICACIONES.

Regulador de 0 a 30 voltios.

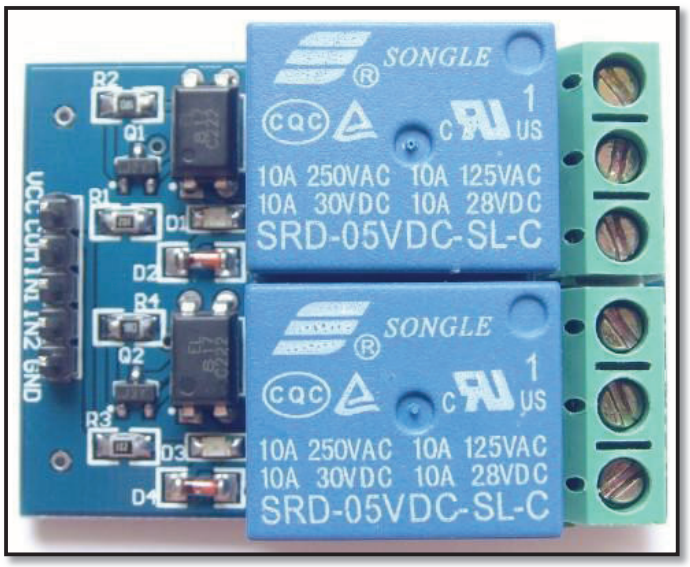


Regulador con electrónica de cierre de 5 voltios.



ANEXO F.
MÓDULO BK008 DE 2 CANALES.

F.1. CARACTERÍSTICAS DEL MÓDULO BK008 DE DOS CANALES.



El módulo BK008 de dos canales es un interruptor o conmutador que permite activar o desactivar un circuito utilizando voltaje y/o corriente superior a la que un microcontrolador podría manejar, de forma que aisla la parte del circuito que comanda las señales de actuación con la parte del circuito que es conmutada, permitiendo de ese modo que señales de baja potencia, accionen mecanismos de potencia elevada.

Las principales características de este módulo son:

- Materiales: plástico y cobre.
- Para trabajar en corriente alterna los valores máximos son: 10 amperios y 250 voltios.
- Para trabajar en corriente continua los valores máximos son: 10 amperios y 30 voltios.

- Tiene un voltaje de bobina de 5 voltios.
- Tamaño del módulo: 45,8mm x 32,4mm.
- El módulo presenta protección de corriente de diodo (sirve para proteger al transistor del breve alto voltaje producido cuando el relé es desconectado).
- Vida útil mecánica: mínimo 10'000.000 de conmutaciones.
- Vida útil eléctrica: mínimo 100.000 conmutaciones.
- Excelente capacidad de aislamiento (mínima interferencia).
- Posee un optoacoplador con objeto de aislar físicamente el circuito que alimenta la base del transistor del propio Arduino.

F.2 VALORES INICIALES DE RENDIMIENTO

Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-40°C to +85°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

ANEXO G.

PROFORMAS.

ΩOMEGA
ELECTRÓNICA

PROFORMA

Fecha: Quito, 18 de julio del 2015
Cliente: Sandoval Villarreal Luis Alberto **RUC:** 1713141438
Dirección: Turubamba Alto Borbón Oe3-116 y Alberto Spencer **TELF:** 022681781

ITEM	DETALLE	CANTIDAD	PRECIO UNITARIO ($)	TOTAL ($)
1	2W10G	1	$ 0,35	$ 0,35
2	4N35	1	$ 0,50	$ 0,50
3	ARDUINO-MEGA2560	1	$ 33,00	$ 33,00
4	ARDUINO-MEGA2560	1	$ 33,00	$ 33,00
5	Bornera 2 pines	3	$ 0,22	$ 0,66
6	Bornera 4 pines	1	$ 0,40	$ 0,40
7	BOTON-7MM	6	$ 0,12	$ 0,72
8	Buzzer 5v	1	$ 0,65	$ 0,65
9	Cables Arduinio	4	$ 0,24	$ 0,96
10	Cámara Ip	1	$ 66,97	$ 66,97
11	Capacitor Cerámico 0.2uF	2	$ 0,12	$ 0,24
12	Capacitor Electrolítico 10uF	1	$ 0,09	$ 0,09
13	Capacitor Electrolítico 470uF	3	$ 0,09	$ 0,27
14	Capacitor Electrolítico 1000uF	1	$ 0,15	$ 0,15
15	Capacitor Electrolítico 10uF	1	$ 0,09	$ 0,09
16	Cerradura Electronica. CENTURY	1	$ 22,77	$ 22,77
17	CONN-SIL5	1	$ 0,55	$ 0,55
18	Diodo 1N4007	4	$ 0,06	$ 0,24
19	Disipador	1	$ 0,65	$ 0,65
20	Espaciadores	16	$ 0,05	$ 0,80
21	Espadin hembra simple	3	$ 0,50	$ 1,50
22	Espadines macho simple	4	$ 0,40	$ 1,60
23	Foco	3	$ 0,90	$ 2,70
24	LCD_16X2_DIL	1	$ 5,17	$ 5,17
25	LM317	6	$ 0,65	$ 3,90
26	LM35	1	$ 2,81	$ 2,81
27	LM7805	1	$ 0,40	$ 0,40
28	Microcontrolador ATMEGA 164p	4	$ 5,80	$ 23,20
29	MOC3011		$ 1,00	$ 0,00
30	Modulo XBee Pro 60MW WIR	1	$ 62,90	$ 62,90
31	Modulo XBee Pro 60MW WIR	5	$ 62,90	$ 314,50
32	Molex 4p	4	$ 0,40	$ 1,60
33	Motor DC	1	$ 33,00	$ 33,00
34	Pernos	40	$ 0,08	$ 3,20

Av. 6 de Diciembre N2381 entre Baquerano y Wilson
Teléfonos: +593-2-520-365/+593-2-522-908/+593-992-529-665
E-mail: ventas@electronicaomega.com

ΩOMEGA
ELECTRÓNICA

35	Placa 11x10 cm	1	$ 15,04	$ 15,04
36	Placa 11x8.5 cm	1	$ 13,10	$ 13,10
37	Placa 12x10 cm	1	$ 16,80	$ 16,80
38	Placa 15x10 cm	2	$ 16,80	$ 33,60
39	Potenciómetro de precisión 5KΩ	2	$ 0,45	$ 0,90
40	Puente de diodos 2A/1000V	2	$ 0,27	$ 0,54
41	Puente de diodos 3A	1	$ 0,27	$ 0,27
42	Puente rectificador de 1.5 AMP	1	$ 0,35	$ 0,35
43	RELE-CERRADURA	2	$ 0,25	$ 0,50
44	Resistencia de 1.2KΩ 1/4W	1	$ 0,02	$ 0,02
45	Resistencia de 10KΩ 1/4W	3	$ 0,02	$ 0,06
46	Resistencia de 180Ω 1/4W	1	$ 0,02	$ 0,02
47	Resistencia de 1KΩ 1/4W	3	$ 0,02	$ 0,06
48	Resistencia de 2KΩ 1/4W	1	$ 0,02	$ 0,02
49	Resistencia de 330Ω 1/4W	1	$ 0,02	$ 0,02
50	Resistencia de 4.7KΩ 1/4W	4	$ 0,02	$ 0,08
51	Switch Pulsador GDE Abierto Corto	3	$ 0,80	$ 2,40
52	TBLOCK-I2	7	$ 0,25	$ 1,75
53	TECLADO	1	$ 1,50	$ 1,50
54	Transformador 12V-1A	4	$ 5,36	$ 21,44
55	Transistor 2n3904	2	$ 0,10	$ 0,20
56	Transistor TIP122	4	$ 0,15	$ 0,60
57	TriacBT137	1	$ 1,00	$ 1,00
58	Tuercas	40	$ 0,08	$ 3,20
59	Zócalo 28 pines	2	$ 0,12	$ 0,24
60	Zócalo 40 pines	4	$ 0,18	$ 0,72
61	Zócalo 6 pines	2	$ 0,08	$ 0,16
62	Zócalo Módulo XBee	5	$ 1,50	$ 7,50
			TOTAL	$ 741,63

Av. 6 de Diciembre N2381 entre Baquerano y Wilson
Teléfonos: +593-2-520-365/+593-2-522-008/+593-092-520-665
E-mail: ventas@electronicaomega.com

Mic@ela.ec

Turubamba Alto Borbón Oe3-116 y Malvas

PROFORMA

Fecha: Quito, 16 de julio del 2015
Cliente: Sandoval Villarreal Luis Alberto
Dirección: Turubamba Alto Borbon Oe3-116 y Alberto Spencer

RUC: 1713141438
TELF: 022681781

ITEM	DETALLE	CANTIDAD	PRECIO UNITARIO $	TOTAL ($)
1	**Elementos Electrónicos:**			
2	2W10G	1	0,35	0,35
3	4N35	1	0,5	0,5
4	ARDUINO-MEGA2560	1	33	33
5	Buzzer 5v	1	0,65	0,65
6	Cámara Ip	1	00,97	00,97
7	Capacitor Cerámico 0.2uF	2	0,12	0,24
8	Capacitor Electrolítico 10uF	1	0,09	0,09
9	Capacitor Electrolítico 4.7UUF	3	0,09	0,27
10	Capacitor Electrolítico 1000uF	1	0,15	0,15
11	Capacitor Electrolítico 10uF	1	0,09	0,09
12	Cerradura Elect. CENTURY	1	22,77	22,77
13	CONN-SIL5	1	0,55	0,55
14	Diodo 1N4007	4	0,06	0,24
15	Disipador	1	0,65	0,65
16	Foco	3	0,9	2,7
17	LCD_16X2_DIL	1	5,17	5,17
18	Modulo XBee Pro 60MW WIR	1	62,9	62,9
19	Molex 4p	4	0,4	1,6
20	Motor DC	1	33	33
21	Potenciómetro de precisión 5KΩ	2	0,45	0,9
22	Puente de diodos 2A/1000V	2	0,27	0,54
23	Puente de diodos 3A	1	0,27	0,27
24	Puente rectificador de 1.5 AMP	1	0,35	0,35
25	RELE-CERRADURA	2	0,25	0,5
26	Resistencia de 180Ω 1/4W	1	0,02	0,02
27	Resistencia de 330Ω 1/4W	1	0,02	0,02
28	Resistencia de 1KΩ 1/4W	3	0,02	0,06
29	Resistencia de 1.2KΩ 1/4W	1	0,02	0,02
30	Resistencia de 2KΩ 1/4W	1	0,02	0,02
31	Resistencia de 4.7KΩ 1/4W	4	0,02	0,08
32	Resistencia de 10KΩ 1/4W	3	0,02	0,06
33	Switch Pulsador GDE Abierto Corto	3	0,8	2,4
34	TBLOCK-I2	7	0,25	1,75
35	TECLADO	1	1,5	1,5
36	Transformador 12V-1A	4	5,36	21,44
37	Transistor 2n3904	2	0,1	0,2
38	Transistor TIP122	4	0,15	0,6
39	Triac BT137	1	1	1
			TOTAL	263,62

Mic@ela.ec

Turubamba Alto Borbón Oe3-116 y Malvas

PROFORMA

Fecha: Quito, 16 de julio del 2015

Cliente: Sandoval Villarreal Luis Alberto RUC: 1713141438

Dirección: Turubamba Alto Borbon Oe3-116 y Alberto Spencer TELF: 022681781

ITEM	DETALLE	CANTIDAD	PRECIO UNITARIO $	TOTAL ($)
	Circuitos Integrados:			
1	ARDUINO MEGA2560	1	33	33
2	Modulo XBee Pro 60MW WIR	5	62,9	314,5
3	LM317	6	0,65	3,9
4	LM7805	1	0,1	0,1
5	LM35	1	2,81	2,81
6	MOC3011		1	0
7	Microcontrolador ATMEGA 164p	4	5,9	23,7
	Circuitos Integrados:			
8	Cables Arduinio	4	0,24	0,96
9	Espadin hembra simple	3	0,5	1,5
10	Espadines macho simple	4	0,4	1,6
11	Zócalo Módulo XBee	5	1,5	7,5
12	Zócalo 6 pines	2	0,08	0,16
13	Zócalo 28 pines	2	0,12	0,24
14	Zócalo 40 pines	4	0,18	0,72
15	Bornera 2 pines	3	0,22	0,66
16	Bornera 4 pines	1	0,4	0,4
	Circuitos Integrados:			
17	Tuercas	40	0,08	3,2
18	Pernos	40	0,08	3,2
19	BOTON-7MM	6	0.12	0.72
20	Placa 11x8.5 cm	1	13,1	13,1
21	Placa 11x10 cm	1	15,04	15,04
22	Placa 15x10 cm	2	16,8	33,6
23	Placa 12x10 cm	1	16,8	16,8
24	Espaciadores	16	0,05	0,8
			TOTAL	478,01

Componentes Loxone.

Todos los componentes que mencionamos están disponibles en nuestra Tienda Online. Permita que el Miniserver tome el control de los dispositivos de su casa y que sus gastos energéticos se vean reducidos.
Cuantificación costes en €

Componentes	Entrada digital	Salida digital	Entrada analógica	Salida analógica	Precio final (IVA incluido)
1x Miniserver	8	8	4	4	502,15 €
1x Extension	12	8	4	4	401,32 €
1x Relay Extension		14			477,95 €
1x DMX Extension	Control de temperatura a través de señal DMX. 128 canales DMX (128 tiras LED monocromo o 42 liras LED RGB).				271,25 €
1x 1-Wire Extension 9x Sensores temperatura	Posibilidad de conectar hasta 32 sensores de temperatura.				266,97 €
1x Lector iButton 4x llaves electrónicas	Control de accesos mediante llaves electrónicas o iButtons.				27,44 €
2x Sensores presencia					253,86 €
1x Fuente de alimentación 4,2A					67,53 €
Total (IVA incluido)					**2268,47 €**

Domintell
La Domótica eficaz

¿Cuánto cuesta la instalación domótica de una vivienda?

En estas tablas encontrará información sobre el coste medio de los componentes Domintell necesarios para realizar su instalación domótica organizados en tres niveles de equipamiento. Los precios son orientativos y obtenidos en base a la media de proyectos Domintell realizados en los últimos 10 años en viviendas en Europa. Las tablas no incluyen los costes de instalación ni el IVA, diferentes en

cada país. Las instalaciones domóticas para oficinas, hoteles, tiendas y locales comerciales requieren siempre de un estudio personalizado.

Contáctenos, estaremos encantados de realizar un presupuesto para su vivienda o equipamiento de acuerdo con sus necesidad.

	En Euros		
Tipo de vivienda	**Equipamiento estándar**	**Equipamiento lujoso**	**Equipamiento premium**
Apartamento 2 habitaciones	2.200	5.900	8.800
Unifamiliar 3 habitaciones	4.200	8.900	12.700
Chalet 4 habitaciones	9.000	12.600	18.700

Con Domintell, disfrute de todas las ventajas de la domótica con un presupuesto ligeramente superior al de una instalación eléctrica convencional.

Como puede apreciarse en los cuadros, una instalación domótica es asequible, incluso con un equipamiento premium que le permitirá disfrutar del máximo confort. Según diferentes estudios, el estándar de una instalación eléctrica convencional representa aproximadamente el 3% del total del presupuesto necesario para la construcción de un apartamento o de una casa nueva. Una instalación Domintell cuesta entre un 1,5% y 6,8% de este presupuesto. Contacte con nosotros mediante nuestro formulario de contacto o llámenos al 931 749 925 y le realizaremos un estudio personalizado.

ANEXO H.

CÓDIGO FUENTE.

H.1 NODO CENTRAL.

```
$ #include <UIPEthernet.h>
EthernetUDP udp;

int decValue=0;
char len1=0;
char len2=0;

String data="";
char message[10];
char dato[40];
/*char dato1[4000];
char dato2[4000];
char dato3[4000];*/

int success;
EthernetClient client;
//char server[] = { "192.168.1.41"}; //
Google
//char server[] = {"192.168.1.9"}; //
Google
char server[] = {"181.112.68.91"}; //
Google
String cadena_request="";
//**********************************
//*Variables para enviar al sistema
Web
//*para mostrar en la Pagina del
estado
//*de la casa
//*
//**********************************
char magnetico=48; //Sensor
magnetico placa de cerradura
char chicharra=48; // El estado de la
chicharra
String lm35="23.884"; // Temperatura
3 digitos
char ventilador=48; //On/off ventilador
char calefactor=48; //On/off calefactor
char automatic=48; //Si el sistema en
automatico o manual
String velocidad="79"; //La velocidad
en 3 digitos
char direccion=48; //Izquierda o
derecha sólo 1 letra
char final_c1=48; //Final de carrera 1
si es on / 0 off
char final_c2=48; //Final de carrera 1
si es on / 0 off
String foco="80"; //Porcentaje de
encendido 0 - 100
char ac=48; //Si esta a 110 VAC

//**********************************
//*Variables que se reciben de la web
//*para interactuar con los actuadores
//*del sistema
//*
//**********************************
char cerradura=0; //Abre o cierra la
puerta
char chicharra_o=0; //Prende o apaga
la chicharra
char rele_ventilador=0; //activa el
ventilador
char rele_calefactor=0; //activa el
calefactor
char activa_foco=0; // apaga o prende
el foco
char porcentaje_foco[3]; //
dimmerizado del foco
char alarma_activa=0; //activa o
desactiva la alarma
String cadena;

void setup() {
  Serial.begin(38400);
  Serial1.begin(38400);
```

```
  uint8_t mac[6] =
{0x00,0x01,0x02,0x03,0x04,0x05};

Ethernet.begin(mac,IPAddress(192,1
68,1,11));
  success = udp.begin(5000);
  Serial.print("initialize: ");
  Serial.println(success ? "success" :
"failed");
  change_xbee(); //Communicate
xbee - camera
//  change_xbee_normal();
//Communicate xbee - modem

  Serial.println("Xbee Ok
configurated");
}

//****************************************
************
//*Xbee Configuration change DL &
MY to comunicate xbee
//*with camera
//****************************************
************
void change_xbee()
{
   int salir=0;
   char incoming_char=0;
   int intentos=0;
   Serial.print("Star to send");
   Serial1.print("+++");
   do
   {
     if(Serial1.available() >0)
     {
        incoming_char=Serial1.read();
        Serial.print(incoming_char);
        if (incoming_char==79)
          salir=salir+1;
        if (incoming_char==75)
          salir=salir+1;
     }
     delay(2);
     intentos=intentos+1;
     if (intentos>1000)
     {
       Serial1.print("+++");
       intentos=0;
       salir=0;
     }
   }while (salir!=2);
   Serial.print("Xbee en modo
comando");
   salir=0;
   intentos=0;
   Serial1.print("ATDLDC8A");
   Serial1.write(13);
   do
   {
     if(Serial1.available() >0)
     {
        incoming_char=Serial1.read();
        Serial.print(incoming_char);
        if (incoming_char==79)
          salir=salir+1;
        if (incoming_char==75)
          salir=salir+1;
     }
     delay(2);
     intentos=intentos+1;
     if (intentos>1000)
     {
       Serial1.print("ATDLDC8A");
       Serial1.write(13);
       intentos=0;
       salir=0;
     }
   }while (salir!=2);
   Serial.print("Xbee cambio DL");

   salir=0;
   intentos=0;
   Serial1.print("ATMYEACD");
   Serial1.write(13);
   do
   {
     if(Serial1.available() >0)
     {
        incoming_char=Serial1.read();
        Serial.print(incoming_char);
        if (incoming_char==79)
          salir=salir+1;
        if (incoming_char==75)
          salir=salir+1;
     }
     delay(2);
     intentos=intentos+1;
```

```
  if (intentos>1000)
  {
    Serial1.print("ATMYEACD");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
}while (salir!=2);
Serial.print("Xbee cambio MY");

salir=0;
intentos=0;
Serial1.print("ATIDABCF");
Serial1.write(13);
do
{
  if(Serial1.available() >0)
  {
     incoming_char=Serial1.read();
     Serial.print(incoming_char);
     if (incoming_char==79)
       salir=salir+1;
     if (incoming_char==75)
       salir=salir+1;
  }
  delay(2);
  intentos=intentos+1;
  if (intentos>1000)
  {
    Serial1.print("ATIDABCF");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
}while (salir!=2);
Serial.print("Xbee cambio ATID");

salir=0;
intentos=0;
Serial1.print("ATWR");
Serial1.write(13);
do
{
  if(Serial1.available() >0)
  {
     incoming_char=Serial1.read();
     Serial.print(incoming_char);
     if (incoming_char==79)
       salir=salir+1;
     if (incoming_char==75)
       salir=salir+1;
  }
  delay(2);
  intentos=intentos+1;
  if (intentos>1000)
  {
    Serial1.print("ATWR");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
}while (salir!=2);
Serial.print("Xbee write OK");

salir=0;
intentos=0;
Serial1.print("ATCN");
Serial1.write(13);
do
{
  if(Serial1.available() >0)
  {
     incoming_char=Serial1.read();
     Serial.print(incoming_char);
     if (incoming_char==79)
       salir=salir+1;
     if (incoming_char==75)
       salir=salir+1;
  }
  delay(2);
  intentos=intentos+1;
  if (intentos>1000)
  {
    Serial1.print("ATCN");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
}while (salir!=2);
Serial.print("Xbee en mode
normal");
  delay(2000);
}

//**************************************
***********
//*Xbee Configuration change DL &
MY to comunicate xbee
```

```
//*with al slaves modules
//**************************************
************
void  change_xbee_normal()
{
  int salir=0;
  char incoming_char=0;
  int intentos=0;
  Serial.print("Star to send");
  Serial1.print("+++");
  do
  {
    if(Serial1.available() >0)
    {
       incoming_char=Serial1.read();
       Serial.print(incoming_char);
       if (incoming_char==79)
         salir=salir+1;
       if (incoming_char==75)
         salir=salir+1;
    }
    delay(2);
    intentos=intentos+1;
    if (intentos>1000)
    {
      Serial1.print("+++");
      intentos=0;
      salir=0;
    }
  }while (salir!=2);
  Serial.print("Xbee en modo
comando");
  salir=0;
  intentos=0;
  Serial1.print("ATDLAAAA");
  Serial1.write(13);
  do
  {
    if(Serial1.available() >0)
    {
       incoming_char=Serial1.read();
       Serial.print(incoming_char);
       if (incoming_char==79)
         salir=salir+1;
       if (incoming_char==75)
         salir=salir+1;
    }
    delay(2);
    intentos=intentos+1;
    if (intentos>1000)
    {
      Serial1.print("ATDLAAAA");
      Serial1.write(13);
      intentos=0;
      salir=0;
    }
  }while (salir!=2);
  Serial.print("Xbee cambio DL");

  salir=0;
  intentos=0;
  Serial1.print("ATMYAAAA");
  Serial1.write(13);
  do
  {
    if(Serial1.available() >0)
    {
       incoming_char=Serial1.read();
       Serial.print(incoming_char);
       if (incoming_char==79)
         salir=salir+1;
       if (incoming_char==75)
         salir=salir+1;
    }
    delay(2);
    intentos=intentos+1;
    if (intentos>1000)
    {
      Serial1.print("ATMYAAAA");
      Serial1.write(13);
      intentos=0;
      salir=0;
    }
  }while (salir!=2);
  Serial.print("Xbee cambio MY");

  salir=0;
  intentos=0;
  Serial1.print("ATID3332");
  Serial1.write(13);
  do
  {
    if(Serial1.available() >0)
    {
       incoming_char=Serial1.read();
       Serial.print(incoming_char);
       if (incoming_char==79)
         salir=salir+1;
```

```
      if (incoming_char==75)
        salir=salir+1;
  }
  delay(2);
  intentos=intentos+1;
  if (intentos>1000)
  {
    Serial1.print("ATID3332");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
}while (salir!=2);
Serial.print("Xbee cambio ATID");

salir=0;
intentos=0;
Serial1.print("ATWR");
Serial1.write(13);
do
{
  if(Serial1.available() >0)
  {
     incoming_char=Serial1.read();
     Serial.print(incoming_char);
     if (incoming_char==79)
       salir=salir+1;
     if (incoming_char==75)
       salir=salir+1;
  }
  delay(2);
  intentos=intentos+1;
  if (intentos>1000)
  {
    Serial1.print("ATWR");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
}while (salir!=2);
Serial.print("Xbee write OK");

salir=0;
intentos=0;
Serial1.print("ATCN");
Serial1.write(13);
do
{
  if(Serial1.available() >0)
  {
     incoming_char=Serial1.read();
     Serial.print(incoming_char);
     if (incoming_char==79)
       salir=salir+1;
     if (incoming_char==75)
       salir=salir+1;
  }
  delay(2);
  intentos=intentos+1;
  if (intentos>1000)
  {
    Serial1.print("ATCN");
    Serial1.write(13);
    intentos=0;
    salir=0;
  }
 }while (salir!=2);
 Serial.print("Xbee en mode
normal");
}

bool take_picture()
{
   Serial1.write(86);
   Serial1.write(0);
   Serial1.write(54);
   Serial1.write(1);
   Serial1.write(0);
   Serial.print("SEND OK");
   bool out=false;
   int conteo=0;
   char dato[5];
   char v=0;
   do
   {
    if(Serial1.available())
    {
      v=Serial1.read();
      if (v==118)
        out=true;
      if (out==true)
      {
        dato[conteo]=v;
        conteo=conteo+1;
      }
    }
   }while(conteo<5);
   Serial.print("*");
```

```
    for (int i=0;i<5;i++)
    {
      Serial.write(dato[i]);
    }
    Serial.print("*");
    if
((dato[0]==118)&&(dato[1]==0)&&(dat
o[2]==54)&&(dato[3]==0)&&(dato[4]=
=0))
    {
       Serial.print("Take Picture OK");
       return true;
    }
    else
    {
       Serial.print("Error en Take
Picture");
       return false;
    }
}
bool stop_picture()
{
    Serial1.write(86);
    Serial1.write(0);
    Serial1.write(54);
    Serial1.write(1);
    Serial1.write(3);
    bool out=false;
    int conteo=0;
    char dato[5];
    char g=0;
    do
    {
      if(Serial1.available())
      {
        g=Serial1.read();
        if (g==118)
          out=true;
        if (out=true)
        {
          dato[conteo]=g;
          conteo=conteo+1;
          Serial.print(g);
        }
      }
    }while(conteo<5);
    if
((dato[0]==118)&&(dato[1]==0)&&(dat
o[2]==54)&&(dato[3]==0)&&(dato[4]=
=0))
    {
       Serial.print("Stop Take Picture
OK");
       return true;
    }
    else
    {
       Serial.print("Error en Stop Take
Picture");
       return false;
    }
}
unsigned int hexToDec(String
hexString) {

  unsigned int decValue = 0;
  int nextInt;

  for (int i = 0; i < hexString.length();
i++) {

    nextInt = int(hexString.charAt(i));
    if (nextInt >= 48 && nextInt <= 57)
nextInt = map(nextInt, 48, 57, 0, 9);
    if (nextInt >= 65 && nextInt <= 70)
nextInt = map(nextInt, 65, 70, 10, 15);
    if (nextInt >= 97 && nextInt <= 102)
nextInt = map(nextInt, 97, 102, 10,
15);
    nextInt = constrain(nextInt, 0, 15);
    decValue = (decValue * 16) +
nextInt;
  }
  return decValue;
}
bool take_length()
{
    Serial1.write(86);
    Serial1.write(0);
    Serial1.write(52);
    Serial1.write(1);
    Serial1.write(0);
    bool out=false;
    int conteo=0;
    char dato[9];
    byte ca=0;
    do
    {
```

```
      if(Serial1.available())
      {
        ca=Serial1.read();
        if (ca==118)
          out=true;

        if (out==true)
        {
          dato[conteo]=ca;
          conteo=conteo+1;
        }
      }
    }while(conteo<9);
    if
((dato[0]==118)&&(dato[1]==0)&&(dat
o[2]==52))
    {
        Serial.print("Obtain Length OK");
        len1=dato[7];
        len2=ca;

decValue=hexToDec(String(dato[7],H
EX)+String(ca,HEX));
        Serial.print(decValue);
        begin_package();
        send_package(64);

send_package_string(String(decValu
e));
        send_package(42);
        end_package();
        return true;
    }
    else
    {
       Serial.print("Error en trama de
longitud");
       return false;
    }


}
bool read_picture()
{
    Serial1.write(86);
    Serial1.write(0);
    Serial1.write(50);
    Serial1.write(12);
    Serial1.write(0);
    Serial1.write(10);
    Serial1.write(0);
    Serial1.write(0);
    Serial1.write(0);
    Serial1.write(0);
    Serial1.write(0);
    Serial1.write(0);
    Serial1.write(len1);
    Serial1.write(len2);
    Serial1.write(0);
    Serial1.write(10);
    bool out=false;
    int conteo=0;
    int flag=0;
    char d=0;
    decValue=decValue+10;
    int conteo2=0;
    do
    {
      if(Serial1.available())
      {
        d=Serial1.read();
        if (d==118)
           out=true;

        if (out==true)
        {
          conteo=conteo+1;
          if (conteo2==0)
            begin_package();

          if (conteo2<100)
          {
            send_package(d);
            conteo2=conteo2+1;
          }
          if (conteo2==100)
          {
            conteo2=0;
            end_package();
          }
        }
      }
    }while(conteo<decValue);
    end_package();
    Serial.print("Obtencion  de imagen
OK");
    Serial.print(conteo);
    Serial.print("@");
```

```
    for (int i=0;i<50;i++)
    {
      Serial.write(dato[i]);
    }
    Serial.print("@");
    return true;
}
void begin_package()
{
//  success =
udp.beginPacket("181.112.90.221",5
000);
//   success =
udp.beginPacket("192.168.1.9",5000)
;
//   success =
udp.beginPacket("192.168.1.41",500
0);
//   success =
udp.beginPacket("192.168.1.9",5000)
;
success =
udp.beginPacket("181.112.68.91",50
00);

}
void send_package(char a)
{
  udp.print(a);
}
void send_package_string(String a)
{
  udp.print(a);
}
void end_package()
{
    success = udp.endPacket();
    Serial.print("endPacket: ");
    Serial.println(success ? "success" :
"failed");
    udp.stop();
    Serial.print("restart connection: ");
    Serial.println (udp.begin(5000) ?
"success" : "failed");
}
bool capture()
{
    bool retorno=false;
    delay(4000);
    if (take_picture())
    {
      delay(8000);
      if (take_length())
      {
        delay(2000);
        if (read_picture())
        {
            Serial.print("END TAKE
PICTURE");
            delay(9000);
            if (stop_picture())
            {
                Serial.print("Salio todo
Ok");
                retorno=true;
            }
        }
      }
    }
    delay(10000);
    return retorno;
}
//Method to insert values of sensor's
void save_data()
{
//  client.connect(server, 80);
  client.connect(server, 8045);
  delay(1000);

//http://localhost/casa/sensor.php?s1=
1&s2=1&s3=1&s4=1&s5=1&s6=1&s7
=1&s8=1&s9=1&s10=1&s11=1&s12=
1

//http://localhost/casa/sensor.php?s1=
1&s2=1&s3=1&s4=1&s5=1&s6=1&s7
=1&s8=1&s9=1&s10=1&s11=1&s12=
1
  Serial.println("connecting...");
  cadena_request="GET
/casa/sensor.php?s1=1&s2=1&s3=1&
s4=1&s5=1&s6=1&s7=1&s8=1&s9=1
&s10=1&s11=1&s12=1 HTTP/1.0";
  if (client.connected()) {
    Serial.println("connected");
//    client.println(cadena_request);
//    client.println("GET
/sensor.php?sen1=0&sen2=0&sen3=
```

```
0&sen4=0 HTTP/1.0");
    client.print("GET
/casa/sensor.php?s1=");
    client.print((char)magnetico);
    client.print("&s2=");
    client.print((char)chicharra);
    client.print("&s3=");
    client.print(lm35);
    client.print("&s4=");
    client.print((char)ventilador);
    client.print("&s5=");
    client.print((char)calefactor);
    client.print("&s6=");
    client.print((char)automatic);
    client.print("&s7=");
    client.print(velocidad);
    client.print("&s8=");
    client.print((char)direccion);
    client.print("&s9=");
    client.print((char)final_c1);
    client.print("&s10=");
    client.print((char)final_c2);
    client.print("&s11=");
    client.print(foco);
    client.print("&s12=");
    client.print((char)ac);
    client.println(" HTTP/1.0");
    client.println();
    client.stop();
  } else {
    Serial.println("connection failed");
  }
}
void ini()
{
//  client.connect(server, 80);
  client.connect(server, 8045);
  delay(1000);

  Serial.println("connecting...");
  if (client.connected()) {
    Serial.println("connected");
    client.println("GET
/casa/infocasa.php HTTP/1.0");
    client.println();
  } else {
    Serial.println("connection failed");
  }
}
void do_request()
{
  //{@*1*1*1*1*1*111*1*@}
  //*1*1*1*1*1*111*1*@
  char take=0;
  cadena="";
  ini();
  do
  {
  if (client.available()>0) {
    char c = client.read();
//    Serial.print(c);
    if ((char)c==125)
      take=2;
    if (take==1)
    {
//        cadena[cadena1]=c;
//        cadena1=cadena1+1;
        cadena=cadena+(char)c;
    }
    if ((char)c==64)
      take=1;
  }
  if (0 == (millis()%1000))
    Serial.println(millis()); //debug
output

  if (!client.connected()) { //never true
    Serial.println();
    Serial.println("disconnecting.");
    Serial.println(cadena);
    analisis();
    client.stop();
  }
 }while(client.connected());
}
void analisis()
{
  //*1*1*1*1*1*11*1*@

    if
(cadena.substring(1,2).equals("1"))
      cerradura=49;
    else
      cerradura=48;

    if
(cadena.substring(3,4).equals("1"))
      chicharra_o=49;
```

```
    else
      chicharra_o=48;

    if
(cadena.substring(5,6).equals("1"))
      rele_ventilador=49;
    if
(cadena.substring(5,6).equals("0"))
      rele_ventilador=48;
    if
(cadena.substring(5,6).equals("2"))
      rele_ventilador=50;
    if
(cadena.substring(5,6).equals("3"))
      rele_ventilador=51;
    if
(cadena.substring(5,6).equals("4"))
      rele_ventilador=52;
    if
(cadena.substring(5,6).equals("5"))
      rele_ventilador=53;
    if
(cadena.substring(5,6).equals("6"))
      rele_ventilador=54;
    if
(cadena.substring(5,6).equals("7"))
      rele_ventilador=55;

    if
(cadena.substring(7,8).equals("1"))
      rele_calefactor=49;
    if
(cadena.substring(7,8).equals("0"))
      rele_calefactor=48;
    if
(cadena.substring(7,8).equals("2"))
      rele_calefactor=50;

    if
(cadena.substring(9,10).equals("0"))
      activa_foco=48;
    if
(cadena.substring(9,10).equals("1"))
      activa_foco=49;
    if
(cadena.substring(9,10).equals("2"))
      activa_foco=50;
    if
(cadena.substring(9,10).equals("3"))
      activa_foco=51;
    if
(cadena.substring(9,10).equals("4"))
      activa_foco=52;
    if
(cadena.substring(9,10).equals("5"))
      activa_foco=53;
    if
(cadena.substring(9,10).equals("6"))
      activa_foco=54;


cadena.substring(11,13).toCharArray(
porcentaje_foco,3);

    if
(cadena.substring(14,15).equals("1"))
      alarma_activa=49;
    else
      alarma_activa=48;

  Serial.println(cerradura);
  Serial.println(chicharra_o);
  Serial.println(rele_ventilador);
  Serial.println(rele_calefactor);
  Serial.println(activa_foco);
  Serial.println(porcentaje_foco);
  Serial.println(cerradura);
}
void loop() {
  delay(1000);
  if (capture()==true)
    Serial.print("IMAGEN TOMADA
OK");
  else
    asm volatile ("  jmp 0");

  delay(1000);

  change_xbee_normal();

  delay(1000);
  do_request();

  delay(1000);

  lecturaAcceso();
  Serial.print("magnetico:
");Serial.println((char)magnetico);
```

```
  Serial.print("pito:
");Serial.println((char)chicharra);
  delay(1000);
  save_data();
  delay(1000);
  change_xbee();
}

void lecturaAcceso()
{
  int c1=0;

  if((char)cerradura == '1')
  {

    Serial1.println("abrir");

    boolean fpuerta = true;
    String auxp="";
    while(fpuerta == true)
    {
      while(Serial1.available()){
        char inBytep =
(char)Serial1.read();
        auxp += inBytep;
        if(inBytep=='\n' || inBytep=='\r'){
          auxp.trim();
          //Serial.println(auxp);
          if(auxp == "1"){
            chicharra = '1';
            fpuerta = false;
          }
        }
      }
      delay(2);
      c1=c1+1;
      if (c1>=2000)
      {
          Serial1.println("abrir");
          c1=0;
          auxp="";
      }

    }
    }
  if((char)chicharra_o == '1')
  {
    Serial1.println("pito");

    boolean fpito = true;
    String auxp="";
    while(fpito == true)
    {
      while(Serial1.available()){
        char inBytep =
(char)Serial1.read();
        auxp += inBytep;
        if(inBytep=='\n' || inBytep=='\r'){
          auxp.trim();
          //Serial.println(auxp);
          if(auxp == "0"){
            chicharra='0';
            fpito = false;
          }
          if(auxp == "1"){
            chicharra = '1';
            fpito = false;
          }
        }
      }
      delay(2);
      c1=c1+1;
      if (c1>=2000)
      {
          Serial1.println("pito");
          c1=0;
          auxp="";
      }

    }
    }

    Serial1.println("estado");
    boolean festado = true;
    String auxe="";
    c1=0;
    while(festado == true){
      while(Serial1.available())
      {
        char inBytee =
(char)Serial1.read();
        auxe += inBytee;
        if(inBytee=='\n' || inBytee=='\r'){
          auxe.trim();
          Serial.println("DATOS DE
ESTADO");
          Serial.println(auxe);
```

```
        if(auxe.indexOf("0")!=-1){
          magnetico='0';
          festado = false;
          Serial.println("MAGNETICO:
");
          Serial.println(magnetico);
        }
        if(auxe.indexOf("1")!=-1){
          magnetico='1';
          festado = false;
          Serial.println("MAGNETICO:
");
          Serial.println(magnetico);
        }

      }
     }
     delay(2);
     c1=c1+1;
     if (c1>=2000)
     {
        Serial1.println("estado");
        c1=0;
        auxe="";
     }
   }

/****************************************
Inicio de Pruebas ****************/
   //1 - > Encender cualquier aparato
   //2 - > Apagar cualquier aparato
   //Ventilador
   //ON
   if((char)rele_ventilador == '1')
   {
      ventilador=espera("AA661177");
   }
   //OFF
   if((char)rele_ventilador == '2')
   {
      ventilador=espera("AA661278");
   }
   //Calefactor
   //ON
   if((char)rele_calefactor == '1')
   {
      calefactor=espera("AA661076");
   }
  //OFF
   if((char)rele_calefactor == '2')
   {
      calefactor=espera("AA661278");
   }
   //FOCO
   //Bajar intensidad
   if((char)activa_foco == '1')
   {
      ac=espera("1234581470");
   }
   //Subir intensidad
   if((char)activa_foco == '2')
   {
      ac=espera("1234581471");
   }
   //Setear 20%
   if((char)activa_foco == '3')
   {
      ac=espera("1234581472");
   }
   //Setear 40%
   if((char)activa_foco == '4')
   {
      ac=espera("1234581473");
   }
   //Setear 60%
   if((char)activa_foco == '5')
   {
      ac=espera("1234581474");
   }
   //Setear 80%
   if((char)activa_foco == '6')
   {
      ac=espera("1234581475");
   }
   //ACTIVACION DE CORTINA
   //Abrir cortina
   if((char)rele_ventilador == '3')
   {
      ac=espera("AA551166");
   }
   //Cerrar cortina
   if((char)rele_ventilador == '4')
   {
      for (int i=0;i<10;i++)
      {
        Serial1.print("AA551065");
        delay(100);
      }
```

```
      ac=espera("AA551065");
   }
   //Parar cortina
   if((char)rele_ventilador == '7')
   {
      ac=espera("AA551671");
   }
   //AUMENTO Y REDUCCION DE
VELOCIDAD
   //Aumenta la velocidad
   if((char)rele_ventilador == '5')
   {
      ac=espera("AA551469");
   }
   //Reduce la velocidad
   if((char)rele_ventilador == '6')
   {
      ac=espera("AA551570");
   }
   //LECTURA DE SEÑALES
ACTUALES EN LA TARJETA
   final_c1=espera("AA551267");
   final_c2=espera("AA551368");

velocidad=datos_tres("AA551469");
   foco=datos_tres("1234581476");
   delay(500);
   lm35=datos_tres("AA661379");
}

char espera(String cadena)
{
   char dato=0;
   int c1=0;
   Serial1.println(cadena);
   boolean wait = true;
   String auxp="";
   while(wait == true)
   {
     while(Serial1.available()){
       char inBytep =
(char)Serial1.read();
       auxp += inBytep;
       if(inBytep=='\n' || inBytep=='\r'){
         auxp.trim();
         if(auxp == "0"){
           dato='0';
           wait = false;
         }
         if(auxp == "1"){
           dato = '1';
           wait = false;
         }
       }
     }
     delay(2);
     c1=c1+1;
     if (c1>=2000)
     {
        Serial1.println(cadena);
        c1=0;
        auxp="";
     }
   }
   return dato;
}

String datos_tres(String cadena)
{
   String dato="";
   int c1=0;
   Serial1.println(cadena);
   boolean wait = true;
   String auxp="";
   while(wait == true)
   {
     while(Serial1.available()){
       char inBytep =
(char)Serial1.read();
       auxp += inBytep;
       if(inBytep=='\n' || inBytep=='\r'){
         auxp.trim();
         dato=auxp;
         wait = false;
       }
     }
     delay(2);
     c1=c1+1;
     if (c1>=2000)
     {
        Serial1.println(cadena);
        c1=0;
        auxp="";
     }
   }
   return dato;
}
```

H.2 NODO CONTROL DE TEMPERATURA.

```
String inputString = "";         // a string
to hold incoming data
boolean stringComplete = true;  //
whether the string is complete

boolean EstadoFuncionamiento = 0;
float LM35 = 0.00;
String cabecera = "AA";
String direccion = "66";
String comando1 = "10";
String comando2 = "11";
String comando3 = "12";
String comando4 = "13";
String comando5 = "14";
String comando6 = "15";
byte tempmin = 20;
byte tempmax = 35;

void setup() {
  Serial.begin(38400);

  pinMode(0,OUTPUT);
  pinMode(1,OUTPUT);
  digitalWrite(0,0);
  digitalWrite(1,0);

  pinMode(7,OUTPUT);
  digitalWrite(7,0);

  pinMode(10,INPUT);
  pinMode(11,INPUT);
  pinMode(12,INPUT);
  digitalWrite(10,1);
  digitalWrite(11,1);
  digitalWrite(12,1);

  pinMode(14,OUTPUT);
  pinMode(15,OUTPUT);
  digitalWrite(14,0);
  digitalWrite(15,0);

  inputString.reserve(200);
  Serial.println("Arduino en Atmega16
esta online....");
}

void loop() {

  lecturaTemperatura();

  if (stringComplete) {
    //Serial.println(inputString);
    inputString.trim();
    ValidarComando();

    inputString = "";
    stringComplete = false;
  }
  if(!digitalRead(12)){
    while(!digitalRead(12)){delay(1);}
```

```
    EstadoFuncionamiento =
EstadoFuncionamiento^1;

digitalWrite(7,EstadoFuncionamiento)
;
    delay(50);
  }
  if(EstadoFuncionamiento == 1){
    if(LM35 >= tempmin && LM35 <=
tempmax){
      digitalWrite(0,0);
      digitalWrite(1,0);
      digitalWrite(14,0);
      digitalWrite(15,0);
      //Serial.println("Ventilador
Apagado");
      //Serial.println("Calefactor
Apagado");
    }else
    if(LM35 < tempmin){
      digitalWrite(0,0);
      digitalWrite(1,1);
      digitalWrite(14,0);
      digitalWrite(15,1);
      //Serial.println("Ventilador
Apagado");
      //Serial.println("Calefactor
Encendido");
    }else
    if(LM35 > tempmax){
      digitalWrite(0,1);
      digitalWrite(1,0);
      digitalWrite(14,1);
      digitalWrite(15,0);
      //Serial.println("Ventilador
Encendido");
      //Serial.println("Calefactor
Apagado");
    }
  }

  if(EstadoFuncionamiento == 0){

    if(!digitalRead(10)){
      while(!digitalRead(10)){delay(1);}
      digitalWrite(0,digitalRead(0)^1);
      digitalWrite(14,digitalRead(14)^1);
      /*Serial.print("LM35:
");*//*Serial.println(LM35);*/

//if(!digitalRead(14)){/*Serial.println("V
entilador Apagado");*/}

//if(digitalRead(14)){/*Serial.println("V
entilador Encendido");*/}
    }
    if(!digitalRead(11)){
      while(!digitalRead(11)){delay(1);}
      digitalWrite(1,digitalRead(1)^1);
      digitalWrite(15,digitalRead(15)^1);
      /*Serial.print("LM35:
");*///Serial.println(LM35);

//if(!digitalRead(15)){Serial.println("Ca
lefactor Apagado");}
```

```
//if(digitalRead(15)){Serial.println("Cal
efactor Encendido");}
    }
  }



}

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n' || inChar == '\r') {
      stringComplete = true;
    }
  }
}

void ValidarComando(){
   if(inputString.substring(0,2) ==
cabecera){
     if(inputString.substring(2,4) ==
direccion){
       if(inputString.substring(4,6) ==
comando1){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
           digitalWrite(0,1);
           digitalWrite(1,0);
           digitalWrite(14,1);
           digitalWrite(15,0);
           Serial.println("1");
           //Serial.println("Ventilador
Apagado");
           //Serial.println("Calefactor
Encendido");
         }
       }
     }
   }

   if(inputString.substring(0,2) ==
cabecera){
     if(inputString.substring(2,4) ==
direccion){
       if(inputString.substring(4,6) ==
comando2){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
```

```
        if(num3 == sum){
          digitalWrite(0,0);
          digitalWrite(1,1);
          digitalWrite(14,0);
          digitalWrite(15,1);
          Serial.println("1");
          //Serial.println("Ventilador
Encendido");
          //Serial.println("Calefactor
Apagado");
        }
       }
     }
   }

   if(inputString.substring(0,2) ==
cabecera){
     if(inputString.substring(2,4) ==
direccion){
       if(inputString.substring(4,6) ==
comando3){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
           digitalWrite(0,0);
           digitalWrite(1,0);
           digitalWrite(14,0);
           digitalWrite(15,0);
           Serial.println("1");
           //Serial.println("Ventilador
Apagado");
           //Serial.println("Calefactor
Apagado");
         }
        }
      }
    }

    if(inputString.substring(0,2) ==
cabecera){
      if(inputString.substring(2,4) ==
direccion){
        if(inputString.substring(4,6) ==
comando4){
          String cks1 =
inputString.substring(2,4);
          String cks2 =
inputString.substring(4,6);
          String cks =
inputString.substring(6,8);
          byte num1 = cks1.toInt();
          byte num2 = cks2.toInt();
          byte num3 = cks.toInt();
          byte sum = num1 + num2;
          if(num3 == sum){
            //Serial.print("LM35:");
            Serial.println(LM35);
          }
```

```
      }
     }
    }

   if(inputString.substring(0,2) ==
cabecera){
     if(inputString.substring(2,4) ==
direccion){
       if(inputString.substring(4,6) ==
comando5){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
           //Serial.print("Vent:");
           Serial.println(digitalRead(14));
         }
       }
     }
   }

   if(inputString.substring(0,2) ==
cabecera){
     if(inputString.substring(2,4) ==
direccion){
       if(inputString.substring(4,6) ==
comando6){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
           //Serial.print("foco:");
           Serial.println(digitalRead(15));
         }
       }
     }
   }

}

void lecturaTemperatura(){
  LM35 = analogRead(0);
  LM35 = ((LM35*5.00)/1024.00)*100;
}
```

H.3 NODO CORTINA MOTORIZADA

```
String inputString = "";         // a string
to hold incoming data
boolean stringComplete = true;  //
whether the string is complete

int pin = 0;
volatile int state = LOW;
int velocidad = 100;
void setup() {
  // initialize serial:
  Serial.begin(38400);
  pinMode(10,INPUT);
  digitalWrite(10,1);
  attachInterrupt(0, int_ext_isr,
FALLING);
  pinMode(pin,OUTPUT);

  pinMode(12,OUTPUT);
  pinMode(14,OUTPUT);
  pinMode(15,OUTPUT);
  digitalWrite(12,0);
  digitalWrite(14,1);
  digitalWrite(15,1);

  pinMode(11,INPUT);
  pinMode(13,INPUT);
  digitalWrite(11,1);
  digitalWrite(13,1);

  pinMode(0,INPUT);
  pinMode(1,INPUT);
  digitalWrite(0,1);
  digitalWrite(1,1);

  analogWrite(12, velocidad);

  // reserve 200 bytes for the
inputString:
  inputString.reserve(200);
  Serial.println("Arduino en Atmega16
esta online....");
}

void loop() {
/*if(!digitalRead(11)){
    digitalWrite(14,0);
    digitalWrite(15,0);
    //Serial.println("paro pin 3
ABIERTO");
  delay(100);
}

if(!digitalRead(13)){
    digitalWrite(14,0);
    digitalWrite(15,0);
    //Serial.println("paro pin 5
CERRADO");
  delay(100);
}*/
  if (stringComplete) {
    //Serial.println(inputString);
```

```
//Serial.println(inputString.substring(0,
1));
   inputString.trim();

   if(inputString.substring(0,2) ==
"AA"){
     if(inputString.substring(2,4) ==
"55"){
       if(inputString.substring(4,6) ==
"10"){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         /*Serial.println(cks1);
         Serial.println(cks2);
         Serial.println(cks);
         Serial.println(num1);
         Serial.println(num2);
         Serial.println(num3);
         Serial.println(sum);*/
         if(num3 == sum){
           //if(!digitalRead(13)){
             digitalWrite(14,1);
             digitalWrite(15,0);
             Serial.println("1");
             //Serial.println("Abriendo...");
             //}
           }
         }
       }
     }

   if(inputString.substring(0,2) ==
"AA"){
     if(inputString.substring(2,4) ==
"55"){
       if(inputString.substring(4,6) ==
"11"){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
           //if(!digitalRead(11)){
             digitalWrite(14,0);
             digitalWrite(15,1);
             Serial.println("1");
             //Serial.println("Cerrando...");
             //}
           }

         }
       }
```

```
    }

    if(inputString.substring(0,2) ==
"AA"){
      if(inputString.substring(2,4) ==
"55"){
        if(inputString.substring(4,6) ==
"12"){
          String cks1 =
inputString.substring(2,4);
          String cks2 =
inputString.substring(4,6);
          String cks =
inputString.substring(6,8);
          byte num1 = cks1.toInt();
          byte num2 = cks2.toInt();
          byte num3 = cks.toInt();
          byte sum = num1 + num2;
          if(num3 == sum){

Serial.println(digitalRead(11));
          }
        }
      }
    }

    if(inputString.substring(0,2) ==
"AA"){
      if(inputString.substring(2,4) ==
"55"){
        if(inputString.substring(4,6) ==
"13"){
          String cks1 =
inputString.substring(2,4);
          String cks2 =
inputString.substring(4,6);
          String cks =
inputString.substring(6,8);
          byte num1 = cks1.toInt();
          byte num2 = cks2.toInt();
          byte num3 = cks.toInt();
          byte sum = num1 + num2;
          if(num3 == sum){
            Serial.println(digitalRead(13));
          }
        }
      }
    }

    if(inputString.substring(0,2) ==
"AA"){
      if(inputString.substring(2,4) ==
"55"){
        if(inputString.substring(4,6) ==
"14"){
          String cks1 =
inputString.substring(2,4);
          String cks2 =
inputString.substring(4,6);
          String cks =
inputString.substring(6,8);
          byte num1 = cks1.toInt();
          byte num2 = cks2.toInt();
          byte num3 = cks.toInt();
          byte sum = num1 + num2;
          if(num3 == sum){
```

```
          velocidad= velocidad+5;
          if(velocidad > 250){ velocidad
= 255;}
          analogWrite(12,velocidad);
          Serial.println(velocidad);
        }
       }
     }
   }

   if(inputString.substring(0,2) ==
"AA"){
     if(inputString.substring(2,4) ==
"55"){
       if(inputString.substring(4,6) ==
"15"){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
          velocidad= velocidad-5;
          if(velocidad < 5){ velocidad =
5;}
          analogWrite(12,velocidad);
          Serial.println(velocidad);
         }
        }
      }
    }

   if(inputString.substring(0,2) ==
"AA"){
     if(inputString.substring(2,4) ==
"55"){
       if(inputString.substring(4,6) ==
"16"){
         String cks1 =
inputString.substring(2,4);
         String cks2 =
inputString.substring(4,6);
         String cks =
inputString.substring(6,8);
         byte num1 = cks1.toInt();
         byte num2 = cks2.toInt();
         byte num3 = cks.toInt();
         byte sum = num1 + num2;
         if(num3 == sum){
          digitalWrite(14,1);
          digitalWrite(15,1);
          Serial.println("1");
         }
        }
      }
    }

   inputString = "";
   stringComplete = false;
  }
```

```
  /*if(!digitalRead(0)){
    while(!digitalRead(0)){
      delay(1);
    }*/
    if(!digitalRead(0)){
      digitalWrite(14,1);
      digitalWrite(15,0);
      delay(200);
      //Serial.println("Abriendo...");
    }
  //}

  /*if(!digitalRead(1)){
    while(!digitalRead(1)){
      delay(1);
    }*/
    if(!digitalRead(1)){
      digitalWrite(14,0);
      digitalWrite(15,1);
      delay(200);
      //Serial.println("Cerrando...");
    }

    if(digitalRead(1) == 0 &&
digitalRead(0) == 0){
      digitalWrite(14,1);
      digitalWrite(15,1);
      delay(5000);
      //Serial.println("Cerrando...");
    }
  //}
}
```

```
void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n' || inChar == '\r') {
      stringComplete = true;
    }
  }
}

void int_ext_isr(){
  //state = !state;
  //if(!digitalRead(11)){
    digitalWrite(14,0);
    digitalWrite(15,0);
    //Serial.println("paro pin 3
ABIERTO");
  //}

  //if(!digitalRead(13)){
    //digitalWrite(14,0);
    //digitalWrite(15,0);
    //Serial.println("paro pin 5
CERRADO");
  //}

Lo ":
Angulo = 3
Case "k":
Angulo = 6
End Select
Return
```

H.4 NODO CONTROL ACCESO

```
#include <LiquidCrystal.h>
#include <Keypad.h>
#define senp 10
LiquidCrystal lcd(A0, A1, A5, A4, A3,
A2);
#include <EEPROM.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //three columns
char keys[ROWS][COLS] = {
   {'A','1','2','3'},
   {'B','4','5','6'},
   {'C','7','8','9'},
   {'D','*','0','#'}
};

byte rowPins[ROWS] = {2, 3, 4, 5};
//connect to the row pinouts of the
keypad
byte colPins[COLS] = {6, 7, 8, 9};
//connect to the column pinouts of the
keypad

Keypad keypad = Keypad(
makeKeymap(keys), rowPins,
colPins, ROWS, COLS );
byte ledPin = 13;

byte rele = 11;
byte pito = 12;

boolean blink = false;
boolean ledPin_state;

String clave = "";
boolean flagteclado = 0;

byte contint=0;

String inputString = "";
boolean stringComplete = false;

byte unidad;
byte decena;
byte centena;
byte mil;
String claveEE = "";
String claveSeguridad = "DCBA";

void setup(){
   Serial.begin(38400);
   inputString.reserve(200);
   lcd.begin(16, 2);
   pinMode(ledPin, OUTPUT);
// Sets the digital pin as output.
   pinMode(rele, OUTPUT);
   pinMode(pito,OUTPUT);
   pinMode(senp, INPUT);
   digitalWrite(senp,HIGH);
   digitalWrite(rele,0);
   digitalWrite(ledPin, HIGH);          //
Turn the LED on.
```

```
    ledPin_state = digitalRead(ledPin);
// Store initial LED state. HIGH when
LED is on.

    unidad  = EEPROM.read(1);
    decena  = EEPROM.read(2);
    centena = EEPROM.read(3);
    mil     = EEPROM.read(4);
    claveEE += (char)unidad;
    claveEE += (char)decena;
    claveEE += (char)centena;
    claveEE += (char)mil;
}

void loop(){
    char key = keypad.getKey();
    lcd.setCursor(0,0);
    lcd.print("*Ingrese  Clave*");
    if (key) {
      clave+=key;
      //Serial.println(key);
      //lcd.setCursor(0,1);
      //lcd.print(clave.length());

      if(clave.length()==1){
        lcd.setCursor(6,1);
        lcd.print("*");
      }else
      if(clave.length()==2){
        lcd.setCursor(7,1);
        lcd.print("*");
      }
      if(clave.length()==3){
        lcd.setCursor(8,1);
        lcd.print("*");
      }
      if(clave.length()==4){
        lcd.setCursor(9,1);
        lcd.print("*");
        delay(500);
        flagteclado = 1;
      }
    }

    if(flagteclado == 1){
      contint++;
      flagteclado = 0;
      if(clave == claveEE){
        contint=0;
        Serial.println("Clave Correcta!!");
        lcd.setCursor(0,0);
        lcd.print("Acceso Permitido");
        digitalWrite(rele,HIGH);
        delay(500);
        digitalWrite(rele,LOW);
        delay(500);
      }else
      if(clave == claveSeguridad){
        contint=0;
        lcd.setCursor(0,0);
        lcd.print("Cambio De  Clave");
        delay(3000);
        digitalWrite(pito,HIGH);
        delay(100);
        digitalWrite(pito,LOW);
        delay(100);
```

```
      digitalWrite(pito,HIGH);
      delay(100);
      digitalWrite(pito,LOW);
      delay(100);
      cambioClave();
    }
    else{
      Serial.println("Clave Incorrecta");

Serial.print("Intento:");Serial.println(co
ntint);
      if(contint>3){
        contint=0;
        lcd.setCursor(0,0);
        lcd.print("Acceso  Denegado");
        Serial.println("Sistema
Bloqueado");
        for(int tiempo = 0;tiempo<60;
tiempo++){
          delay(1000);

digitalWrite(pito,digitalRead(pito)^1);
          Serial.print("tiempo bloqueado
");Serial.println(tiempo);
        }
      digitalWrite(pito,0);
      }
    }
    lcd.clear();
    clave="";
  }

  if (stringComplete) {
    //Serial.println(inputString);
    inputString.trim();
    if(inputString.indexOf("abrir")!= -
1){
      Serial.println("1");
      lcd.setCursor(0,0);
      lcd.print("Acceso Permitido");
      digitalWrite(rele,HIGH);
      delay(500);
      digitalWrite(rele,LOW);
      delay(500);
      lcd.setCursor(0,1);

lcd.print(inputString.substring(inputStr
ing.indexOf("abrir")));
    }
    if(inputString == "estado"){
      Serial.println(digitalRead(senp));
      lcd.setCursor(0,1);
      lcd.print(inputString);
    }
    if(inputString.indexOf("pito")!= -1){

digitalWrite(pito,digitalRead(pito)^1);
      Serial.println(digitalRead(pito));
      lcd.setCursor(0,1);

lcd.print(inputString.substring(inputStr
ing.indexOf("pito")));
    }
    inputString = "";
    stringComplete = false;
  }
```

```
digitalWrite(ledPin,!digitalRead(ledPin
));
    delay(100);
}

void serialEvent() {
  while (Serial.available()) {
    char inChar = (char)Serial.read();
    inputString += inChar;
    if (inChar == '\n' || inChar == '\r') {
      stringComplete = true;
    }
  }
}

void cambioClave(){
  lcd.clear();
  boolean ncu = true;
  while(ncu==true){
    char key = keypad.getKey();
    if (key){
      EEPROM.write(1, (int)key);
        lcd.setCursor(0,0);
        lcd.print("Digito Uno: ");
        lcd.setCursor(6,1);
        lcd.print((int)key);
        ncu = false;
        delay(2000);
      }
    }
  //lcd.clear();
  boolean ncd = true;
  while(ncd==true){
    char key = keypad.getKey();
    if (key){
      EEPROM.write(2, (int)key);
      lcd.setCursor(0,0);
      lcd.print("Digito Dos: ");
      lcd.setCursor(7,1);
      lcd.print((int)key);
      ncd = false;
      delay(2000);
    }
  }
  lcd.clear();
  boolean ncc = true;
  while(ncc==true){
    char key = keypad.getKey();
    if (key){
      EEPROM.write(3, (int)key);
      lcd.setCursor(0,0);
      lcd.print("Digito tres: ");
      lcd.setCursor(8,1);
      lcd.print((int)key);
      ncc = false;
      delay(2000);
    }
  }
  lcd.clear();
  boolean ncm = true;
  while(ncm==true){
    char key = keypad.getKey();
    if (key){
      EEPROM.write(4, (int)key);
```

```
    lcd.setCursor(0,0);
    lcd.print("Digito cuatro: ");
    lcd.setCursor(9,1);
    lcd.print((int)key);
    ncm = false;
    delay(2000);
  }
 }
```

```
 lcd.clear();
  unidad  = EEPROM.read(1);
  decena  = EEPROM.read(2);
  centena = EEPROM.read(3);
  mil     = EEPROM.read(4);
  claveEE += (char)unidad;
  claveEE += (char)decena;
  claveEE += (char)centena;
```

H.5 NODO DIMMER

```
$regfile="m164pdef.dat"
$baud= 9600
$crystal= 8000000
'Control  Triac
Config Portc.1 =Output
Portc.1 = 0
Triac AliasPortc.1
Ddrb. 0 =0 :Portb.0 = 1 'input Pull up
Ddrb.1 = 0 :Portb.1 = 1
Ddrb.2 = 0 :Portb.2 = 1
Ddrb.3 = 0 :Portb.3 = 1
Ddrb.4 = 0 :Portb.4 = 1
Ddrb.5 = 0 :Portb.5 = 1
OnTimer1 Int_time1
Enable Interrupts
Enable Urxc
Dim Bright As Integer
Dim Luz As Integer
Dim A As Integer
Dim B As Integer
Dim Dato As String* 1
```

```
Principal:
If Pinb.0 = 0 Then
EnableTimer1
Luz = 100   'Foco On 100%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Pinb.1 = 0 Then
EnableTimer1
Luz = 0 'Foco 20%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Pinb.2 = 0 Then
EnableTimer1
Luz = 10 'Foco 40%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
```

```
If Pinb.3 = 0 Then
EnableTimer1
Luz = 20 'Foco 60%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Pinb.4 = 0 Then
EnableTimer1
Luz = 70 'Foco 80%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Pinb.5 = 0 Then
Enable Timer1
Getchar:
Dato =Inkey()
If Dato ="b"Then
Luz = 100 'Foco On 100%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Dato ="c" Then
EnableTimer1
Luz = 0 'Foco 20%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Dato ="d "Then
EnableTimer1
Luz = 10 'Foco 40%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Dato ="e" Then
EnableTimer1
Luz = 20 'Foco 60%
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
If Dato ="f" Then
EnableTimer1
Luz = 70 'Foco 80%
A = 100 – Luz
B = A * 128
Bright = 65535 - B
End If
If Dato ="g" Then
EnableTimer1
Luz = 40 'Foco Off
A = 100 - Luz
B = A * 128
Bright = 65535 - B
End If
Return
```