ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

SIMULACIÓN DE UN SISTEMA DE CONTROL DE LUCES Y PERSIANAS EN LA DIRECCIÓN, SUBDIRECCIÓN Y SALA DE REUNIONES DE LA ESFOT

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

KEVIN FERNANDO ROMERO FLORES

kevin.romero@epn.edu.ec

STALIN FERNANDO HERNÁNDEZ CAIZA

stalin.hernandez@epn.edu.ec

DIRECTOR: ING. LEANDRO PAZMIÑO, MSC.

leandro.pazmino@epn.edu.ec

CODIRECTOR: ING. MÓNICA VINUEZA RHOR, MSC.

monica.vinueza@epn.edu.ec

Quito, diciembre 2021

CERTIFICACIÓN

Certificamos que el presente trabajo fue desarrollado por KEVIN FERNANDO ROMERO FLORES y STALIN FERNANDO HERNÁNDEZ CAIZA como requerimiento parcial a la obtención del título de TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES, bajo nuestra supervisión.

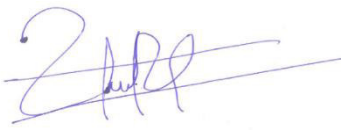
Ing. Leandro Pazmiño, MSc.	Ing. Mónica Vinueza, MSc.
DIRECTOR DEL PROYECTO	CODIRECTORA DEL PROYECTO

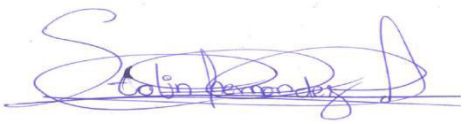
DECLARACIÓN

Nosotros, Kevin Fernando Romero Flores, Stalin Fernando Hernández Caiza declaramos bajo juramento que el trabajo aquí descrito es de mi/nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he/hemos consultado las referencias bibliográficas que se incluyen en este documento.

Sin prejuicio de los derechos reconocidos en el primer párrafo del artículo 144 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación – COESC-, soy/somos titular/titulares de la obra en mención y otorgo/otorgamos una licencia gratuita, intransferible y no exclusiva de uso con fines académicos a la Escuela Politécnica Nacional.

Entregamos toda la información técnica pertinente, en caso de que hubiese una explotación comercial de la obra por parte de la EPN, se negociará los porcentajes de los beneficios conforme lo establece la normativa nacional vigente.





Kevin Fernando Romero Flores
CI: 172519879-8
Teléfono: 0968151813
Correo: kevin.romero@epn.edu.ec

Stalin Fernando Hernández Caiza
CI: 100386415-2
Teléfono: 0984397277
Correo: stalin.hernandez@epn.edu.ec

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El documento que se presenta se basa en la simulación de un sistema de control de luces y persianas en la Dirección, Subdirección y sala de reuniones de la ESFOT, permitiendo que a futuro el sistema sea implementado, y de esa manera mejorar las instalaciones de la institución. De esta manera se facilita a los profesores el uso de las luminarias, así como de las persianas, además, de optimizar el uso de la energía eléctrica manteniendo encendidas las iluminarias solo cuando sea necesario o cuando el usuario así lo requiera.

Para llevar a cabo el desarrollo de la simulación es necesario contar con la ayuda de un *software* adecuado que permita diseñar y simular circuitos electrónicos de una forma práctica, esta herramienta ayuda a crear el modelo de simulación con el propósito de que el sistema de control pueda ejecutar las acciones de encendido y apagado de las luces de cada oficina, así como la apertura y cierre de las persianas de una manera automática, personalizada y manual. Esta última opción está disponible solo para el sistema de iluminación.

Otras de las opciones que tiene el sistema, es permitir una comunicación inalámbrica entre el sistema de control y el usuario mediante una aplicación móvil, con la finalidad de facilitar el uso del sistema. No obstante, al presentar esta iniciativa lo que se pretende además de automatizar el sistema de luces y persianas, es modernizar las áreas ya mencionadas y también ayudar a solventar las necesidades visuales creando estaciones de trabajo bien iluminadas y confortables, y contribuir en el ahorro energético.

Palabras clave: domótica, control iluminación, control persianas, *Arduino*

ABSTRACT

The document presented here is based on the simulation of a control system for lights and blinds in the ESFOT's Director's Office, Deputy Director's Office and meeting room, allowing the system to be implemented in the future, and thus improve the institution's facilities. This will make it easier for teachers to use the lights and blinds, as well as optimize the use of electrical energy by keeping the lights on only when necessary or when required by the user.

To carry out the development of the simulation it is necessary to have the help of a suitable software that allows to design and simulate electronic circuits in a practical way, this tool helps to create the simulation model with the purpose that the control system can execute the actions of turning on and off the lights of each office, as well as the opening and closing of the blinds in an automatic, customized, and manual way. This last option is available only for the lighting system.

Another of the system's options is to allow wireless communication between the control system and the user through a mobile application, to facilitate the use of the system. However, the aim of this initiative, in addition to automating the lighting and blinds system, is to modernize the aforementioned areas and to help solve visual needs by creating well-lit and comfortable workstations, and to contribute to energy savings.

Keywords: *home automation, lighting control, blinds control, Arduino*

1 INTRODUCCIÓN

Las condiciones adecuadas de trabajo dentro de una institución ayudan a la optimización de los procesos que se ejecutan dentro de la misma, en este contexto el factor de mayor importancia dentro de un área de trabajo es la iluminación, puesto que juega un rol importante en el bienestar del trabajador, y que con ello se puede evitar varios problemas de salud visual. El ser humano tiene la capacidad de adaptarse a las diferentes calidades lumínicas, sin embargo, la deficiencia de luz produce un aumento de la fatiga visual (las variaciones de luz pueden ser peligrosas, pues ciegan temporalmente), generar una reducción en el rendimiento del desarrollo de las actividades, incremento de errores y en ocasiones provocar accidentes [1].

Actualmente la Dirección, Subdirección y Sala de reuniones de la ESFOT cuentan con un sistema manual de cierre y apertura de persianas que presentan inconvenientes tales como: al momento de tirar de la cinta, ésta tiende a trabarse en los ejes del panel, teniendo dificultades tanto para abrir o cerrar las persianas, obstruyendo el paso de la luz natural a las diferentes áreas de trabajo, es importante tener en cuenta este sistema debido a que brinda un soporte a la iluminación artificial que posee cada oficina.

Otro inconveniente que se presenta es la pérdida de tiempo y concentración del usuario al momento de ejecutar de manera manual el encendido o apagado de las luces, así como cierre o apertura de las persianas.

Contar con un sistema de luces y persianas automatizado y eficiente que permita satisfacer necesidades visuales, optimización del tiempo del usuario, modernizar las áreas de trabajo, además de contribuir con el energético [2]. Considerando que la iluminación representa casi un 20% del gasto de electricidad total de una institución, contar con un sistema de luz artificial eficiente, permite utilizar la luz natural como soporte luminario, contribuyendo con el ahorro energético de hasta un 80% [3].

Con lo mencionado anteriormente, se realiza la simulación del sistema de control de luces y persianas, el cual se pudiera implementar a futuro con el fin de cumplir con los propósitos de: automatizar el sistema de luces y persianas, modernizar las áreas de trabajo ya mencionadas, ayudar a mejorar las necesidades visuales creando lugares visuales saludables y cómodos, contribuir con el ahorro energético, además optimizar el tiempo de trabajo del usuario ya que evitará que el usuario controle de manera manual la iluminación en su entorno ya sea encendiendo las luces o moviendo las persianas, evitando que el usuario se desconcentre en el desarrollo de las actividades.

1.1 Objetivo general

Simular un sistema de control de luces y persianas en la Dirección, Subdirección y sala de reuniones de la ESFOT.

1.2 Objetivos específicos

- Diseñar un sistema de control de luces y persianas.
- Determinar el *software* de programación y el programa de simulación del sistema de control.
- Crear una aplicación móvil que controle el sistema de control.
- Simular la aplicación móvil y el sistema de control de luces y persianas.

2 METODOLOGÍA

2.1 Objetivo 1: Diseñar un sistema de control de luces y persianas

Para realizar el diseño se tomó en cuenta que el circuito principal debía dividirse en dos subcircuitos; para el primer subcircuito se asignó el nombre de “circuito de persianas” capaz de controlar el movimiento de los motores de las persianas, y el segundo subcircuito denominado “circuito de iluminación” que basa su funcionamiento en el encendido, apagado y regulación de la intensidad de las luces, este último circuito contará con tres etapas. A continuación, se describe el funcionamiento de cada subcircuito.

- **Subcircuito 1**

El circuito en mención tiene como elemento central una placa electrónica *Arduino* UNO, el cual lee y procesa la información, y realiza determinadas acciones, generando como resultado un proceso donde las persianas se abran o cierren. Para controlar la apertura y cierre de las persianas se utilizó el motor a pasos *Nema-23* el cual se controla mediante el *Driver* L293D este controlador entrega al motor una corriente de salida de 1.2 (A) permitiendo que el giro se desarrolle con el torque necesario de tal manera que el eje de las persianas rote en cualquiera de las direcciones sin mayor complicación. En el circuito de persianas se utiliza una fotoresistencia la cual cumple la función de monitorear la cantidad de luz que hay en el área de trabajo, y con base a esta información controlar el motor *Nema-*23, de una manera automática.

Este circuito dispone la opción de abrir y cerrar las persianas cuando la persona así lo desee, para esto se creará una aplicación móvil la cual permite controlar este circuito a través de una conexión *Bluetooth*, para ello se conecta el módulo *Bluetooth* HC-05 con *Arduino* UNO.

Subcircuito 2

Primera etapa: circuito de potencia

En esta etapa lo que se busca es realizar un circuito que permita determinar el punto de cruce por cero de la señal senoidal de la red eléctrica el cual sirve como referencia para determinar la cantidad de energía que es entregada a la carga en este caso focos *LED*

dimerizables. Para realizar este circuito de potencia se utiliza un puente de diodos que permite obtener una forma de onda pulsante la cual es utilizada para que a través de un optoacoplador PC817 y una resistencia como protección se detecte el cruce por cero.

Segunda etapa: circuito de control

Para el sistema de control se usó *Arduino* UNO el cual permitió acoplar todos los elementos tanto como el circuito de cruce por cero como el circuito de disparo, así como también los demás elementos que permiten que el circuito de iluminación cumpla con los modos de funcionamiento.

Se usó una fotoresistencia *LDR* que permite captar la cantidad de luz que existe en el entorno el cual *Arduino* UNO procesará esta información enviando órdenes al circuito de disparo para encender o apagar las luces. Se acopló al circuito un potenciómetro el cual, al variar su posición, el usuario pueda cambiar la intensidad de luz de los focos, este circuito también dispone de un módulo *bluetooth* está en constante comunicación con la aplicación móvil permitiendo al usuario controlar el estado de los focos.

Tercera etapa: circuito de disparo

Para esta etapa se usó un optoacoplador MOC3021 el cual está compuesto por un *LED* infrarrojo y un optotriac, donde este elemento sirve para protección entre el TRIAC BT136 y la placa *Arduino* UNO.

La ventaja de usar el MOC3021 es el optotriac, permitiendo que la señal eléctrica AC fluya en ambos sentidos aprovechando así toda la capacidad de la energía eléctrica que pasa a través del TRIAC BT136, este último elemento es usado para poder controlar la cantidad de energía que se entrega a las luces, teniendo como resultado la variación de la intensidad luminosa de estas.

Para poder controlar el tiempo de disparo del TRAIC, se polariza el *LED* que el optoacoplador MOC3021 contiene con la finalidad de activar el optotriac para que la señal alterna fluya a través del TRAIC BT136 y controlar la cantidad de energía que reciben las luces.

2.2 Objetivo 2: Determinar el *software* de programación y el programa de simulación del sistema

Con base en los requerimientos identificados y detalles descritos en la sección anterior, se selecciona el entorno de programación y el programa de simulación del sistema de control de luces y persianas. En primera instancia, se utilizó el entorno de desarrollo *Arduino* IDE, donde se desarrolló el código fuente con lenguaje de programación C++.

Para realizar la simulación de los circuitos se usó *Proteus Design Suite* en la versión *Proteus 8 Professional* dado que permite diseñar y simular circuitos electrónicos. La aplicación móvil se creó mediante el *software* de desarrollo *App Inventor*, misma que proporciona facilidades de programación y simulación de la aplicación. El sistema de control interactúa con la aplicación móvil mediante *Bluetooth* empleando el módulo HC-05.

2.3 Objetivo 3: Crear una aplicación móvil que controle el sistema de control

La aplicación móvil se creó a través del entorno de desarrollo *App Inventor*, este *software* permite crear aplicaciones para dispositivos con sistema operativo *Android*. *App Inventor* es una herramienta de desarrollo visual que permite programar por medio de un código de bloques que se enlazan entre sí para crear una determinada acción dentro del programa. La aplicación móvil tendrá tres etapas que permiten cumplir con los objetivos planteados en el proyecto.

Primera etapa: conectividad Bluetooth

En esta etapa se colocó un menú en la interfaz de la pantalla, el cual sirve para visualizar los dispositivos *Bluetooth* disponibles a los que se puede conectar, una vez enlazado se puede transmitir información y controlar el sistema.

Segunda etapa: control de luces y persianas

En esta etapa la aplicación tiene la función de encendido, apagado y control de la intensidad de las luces, al igual que la apertura y cierre de las persianas, para lo cual se dispone de cuatro botones con el fin de cumplir las funciones anteriores mencionadas. Para el control de la intensidad de las luces se incorporará una *slider* en la aplicación,

al momento de variar la posición se envía al módulo HC-05 información de control que *Arduino* UNO procesa y envía la orden para que las luces varíen la intensidad de acuerdo con la posición del deslizador o *slider*.

La finalidad de la creación de la aplicación es tener el control del circuito de luces y persianas a través de una conexión *Bluetooth*, con lo que se puede prender, apagar o variar la intensidad de las luces, y de igual manera cambiar la posición de las persianas.

2.4 Objetivo 4: Simular el sistema de control de luces y persianas

Como último procedimiento se realizaron las pruebas de funcionamiento, para ello se tiene el código fuente completo el cual determina el funcionamiento de cada subcircuito, además de contar con la aplicación móvil. Con lo anterior mencionado, se procedió a compilar, ejecutar y simular el funcionamiento del sistema. Se verificó y analizó el desarrollo del sistema, para posterior a ello realizar respectivas correcciones.

3 RESULTADOS Y DISCUSIÓN

El proyecto tiene como iniciativa la simulación de un sistema de control de luces y persianas, permitiendo tener una visión clara del funcionamiento del sistema de control y verificar el comportamiento de cada elemento electrónico que forma parte del sistema.

3.1 Diseño del sistema de control de luces y persianas

Análisis de las condiciones actuales de las oficinas

La Dirección, Subdirección y sala de reuniones de la ESFOT cuentan con un sistema manual de cierre y apertura de persianas, mismo que al momento de tirar de la cinta, ésta tiende a trabarse en los ejes del panel, teniendo dificultades tanto para abrir como para cerrar las persianas. Otro inconveniente que se presenta es no aprovechar el recurso tiempo y concentración del usuario al momento de realizar dicha actividad.

Al ser un sistema manual, el usuario debe abrir las persianas al iniciar su jornada laboral, y cerrar las persianas al finalizar sus actividades o cuando el usuario desee privacidad. Muchas de las veces este proceso no es rutinario, ya sea por olvido o falta de interés en realizar dicha actividad. Si por olvido no abre las persianas en la mañana, el área de trabajo no cuenta con una iluminación natural adecuada, por lo que para tener una buena visibilidad el usuario utiliza la luz artificial, desaprovechando así la oportunidad de generar un ahorro en el consumo de energía eléctrica.

No obstante, el sistema de luces actualmente también se maneja de forma manual, creando algunos inconvenientes de los cuales la desconcentración del usuario al momento de prender o apagar este sistema y el uso poco eficiente de la energía eléctrica son los principales puntos por cubrir en el desarrollo del proyecto.

En la Figura 3-1. se puede apreciar el sistema de iluminación tradicional instalado con que cuentan las oficinas previamente mencionadas.

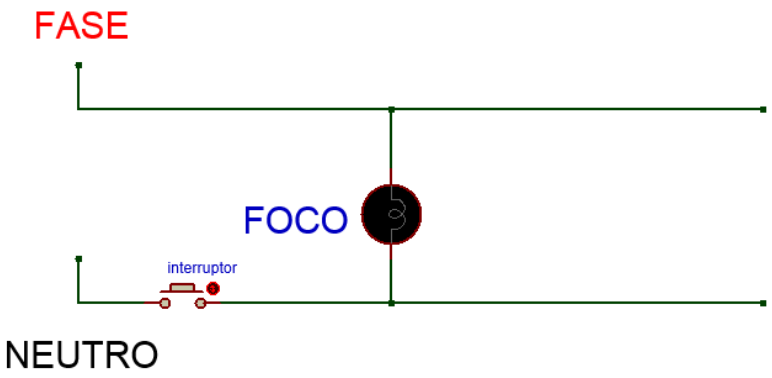


Figura 3-1 Sistema de iluminación actual de las oficinas

Requerimientos para el diseño del sistema de control

El presente proyecto se basa en diseñar y simular un sistema de control de luces y persianas para lo cual se debe cumplir los objetivos propuestos en un inicio. Con base al análisis realizado sobre la situación actual de las oficinas se determinó que el sistema de control debe cumplir con las siguientes funciones:

- Permitir apertura y cierre de las persianas, de igual manera el encendido y apagado de las luminarias sin necesidad de que el usuario realice de manera manual tales acciones.
- Para realizar las acciones anteriores mencionadas, el sistema consta de tres modos de operación que se detallan a continuación:
    - Modo automático: funciona cuando el sistema usa la información de la fotoresistencia y en función de la lectura del sensor realiza el encendido, apagado de las luminarias, y de igual manera el cierre o apertura de las persianas.
    - Modo personalizado: permite controlar el sistema de manera inalámbrica, conectándose la aplicación móvil al microcontrolador, el cual controla los dispositivos electrónicos, mediante un módulo *Bluetooth*.
    - Modo manual: disponible únicamente para el sistema de iluminación, este modo de funcionamiento permite que el usuario prenda, apague o regule la intensidad de las luces de forma manual.

El sistema se divide en dos subcircuitos, donde el primer circuito controla el movimiento de las persianas y el segundo se encarga del encendido y apagado, y regulación de la intensidad de las luces.

Cada subcircuito cuenta con una placa *Arduino* diferente con el fin de evitar interferencias en la ejecución del código fuente, ya que *Arduino* UNO no puede ejecutar

dos acciones a la vez, es decir al momento de abrir las persianas y apagar las luminarias de manera sincronizada no se podrá realizar tal acción, primero se abrirá las persianas y después de finalizar esa acción se puede prender o apagar las luces, este inconveniente se presenta también al momento de realizar la comunicación inalámbrica, entre la aplicación móvil y los dos subcircuitos.

En la Figura 3-2 se representa de manera gráfica el circuito de luces y persianas en donde se muestra en breves rasgos los componentes utilizados para el diseño del sistema de control.

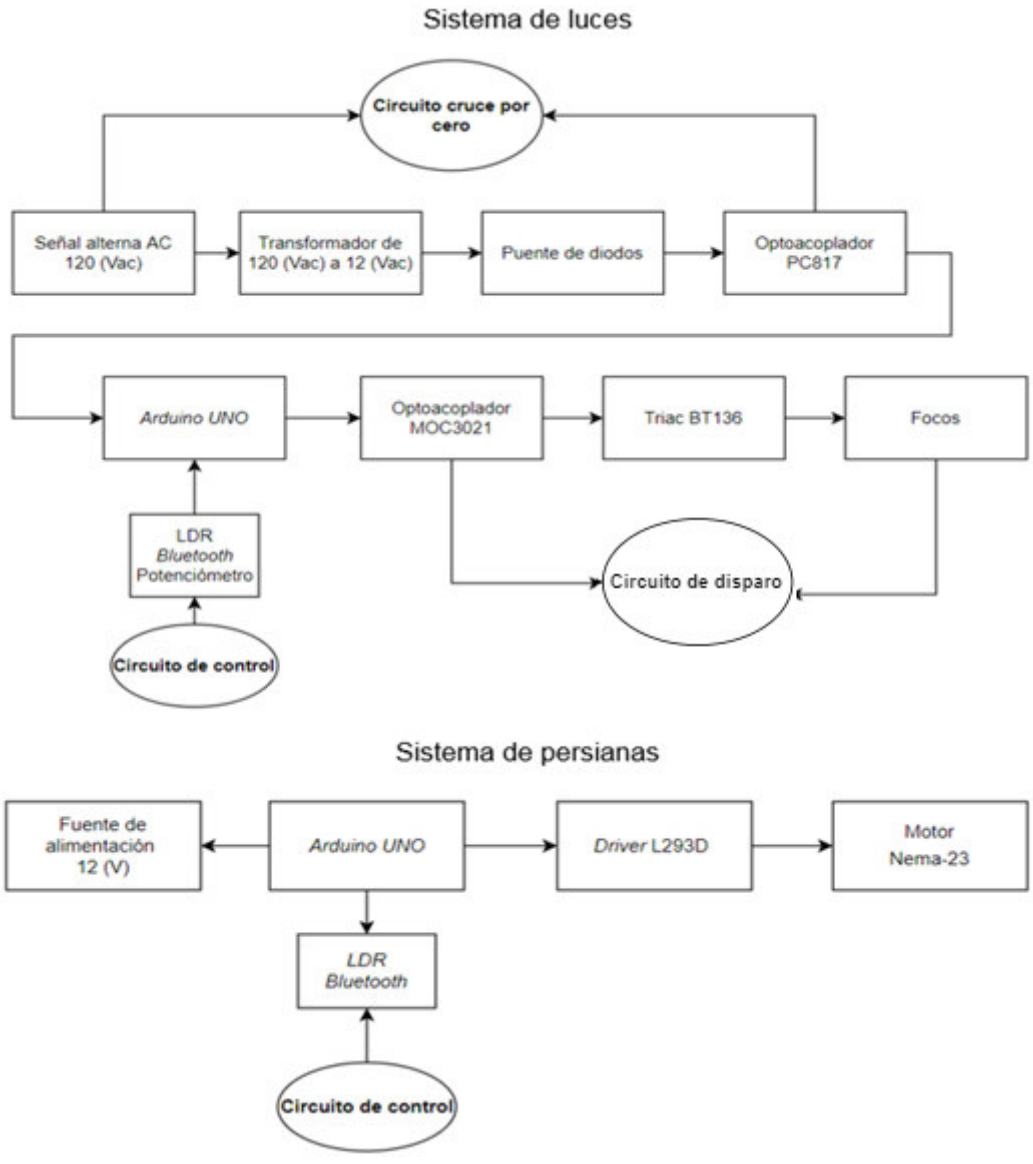


Figura 3-2 Diagrama esquemático del sistema de luces y persianas

Al estar el sistema de control en funcionamiento, se consideró un posible corto circuito, fallo o deterioro en cualquier dispositivo electrónico a largo plazo, de ser el caso afectaría un solo subcircuito y no los dos, por consiguiente, la inspección y el arreglo del sistema sería más fácil y en menor tiempo.

Para el sistema de control de luces se usa una placa *Arduino* UNO, el cual recibe y procesa la información y en base a ello dispone realizar determinadas acciones tales como: encender, apagar o regular la intensidad de las luces, de la misma manera se puede ejecutar la apertura o cierre de las persianas, estas acciones mencionadas se realizan de manera automática, personalizada o manual dependiendo el caso.

Para la comunicación inalámbrica entre los dos subcircuitos y la aplicación móvil se utilizó un módulo *Bluetooth* el cual se conecta con el microcontrolador, este tiene la función de recibir los datos generados desde la aplicación móvil.

Adicional a los elementos ya mencionados, el sistema de control requiere de un dispositivo que permita la lectura de la cantidad de luz existente en el área de trabajo. Para lo cual se utiliza una resistencia, una fotoresistencia y una fuente de voltaje de 5 (V), al conectar estos elementos se forma un circuito divisor de tensión, éste tiene la función de emitir un voltaje de salida en función de la variación de la intensidad de luz existente. Al realizar este proceso mencionado se procede a realizar acciones como apertura o cierre de las persianas, así mismo se puede controlar de manera automática el de encendido, apagado o regulación de la intensidad de las luces.

Para el circuito de persianas además de los elementos ya citados, se tomó en cuenta otros posibles dispositivos electrónicos que completaran el diseño del subcircuito controlador de persianas, con base a un análisis de diversos elementos electrónicos se determinó usar un motor a pasos el cual permite girar el eje de las persianas, para que este funcione en óptimas condiciones se usa un controlador de motores a pasos, mismo que es alimentado con una fuente externa de 12 (V).

Para complementar el circuito de luces, se verificó que en primera instancia se debe controlar la señal de la corriente alterna la cual alimentará las luminarias; para ello se analizó los posibles elementos electrónicos que se debe disponer para llevar a cabo este cumplimiento. Los dispositivos electrónicos apropiados son: resistencias, optoacopladores, un transistor, un diodo rectificador, un puente de diodos, un *TRIAC*, un puente de diodos, un transformador, mismos que permiten controlar la señal eléctrica dando como resultado el cumplimiento de las funciones del circuito de luces, anterior mencionadas.

En la Figura 3-3, se puede apreciar el diagrama general de los elementos que se involucran en el sistema de control de luces y persianas, y las funciones que cumplen cada dispositivo o circuito.

Figura 3-3 Diagrama general del sistema de control de luces y persianas

Una vez establecido la forma de funcionamiento del sistema de control, se procederá a diseñar los circuitos que serán capaces de cumplir con cada función que demanda el proyecto.

Diseño del circuito de control de persianas

Para el diseño del circuito de persianas se procedió a conectar en primer lugar el motor a pasos con el controlador, en la Figura 3-4 se puede verificar el diagrama de conexiones entre los dos dispositivos.

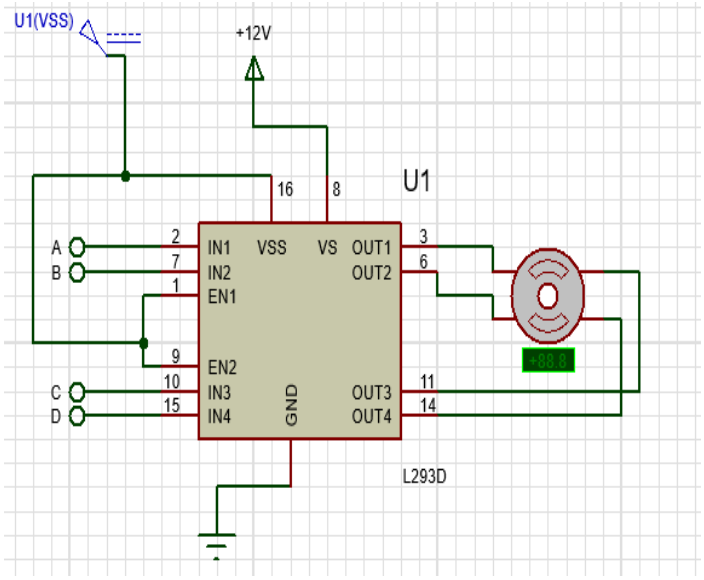


Figura 3-4 Conexión del controlador L293D con el motor *Nema*-23

Posterior a ello, se procedió a conectar los pines (A, B, C, D) de entrada del controlador, con los pines de la placa *Arduino* UNO, como se muestra en la Figura 3-5, esta conexión permite el control de la velocidad y del sentido de giro del motor.

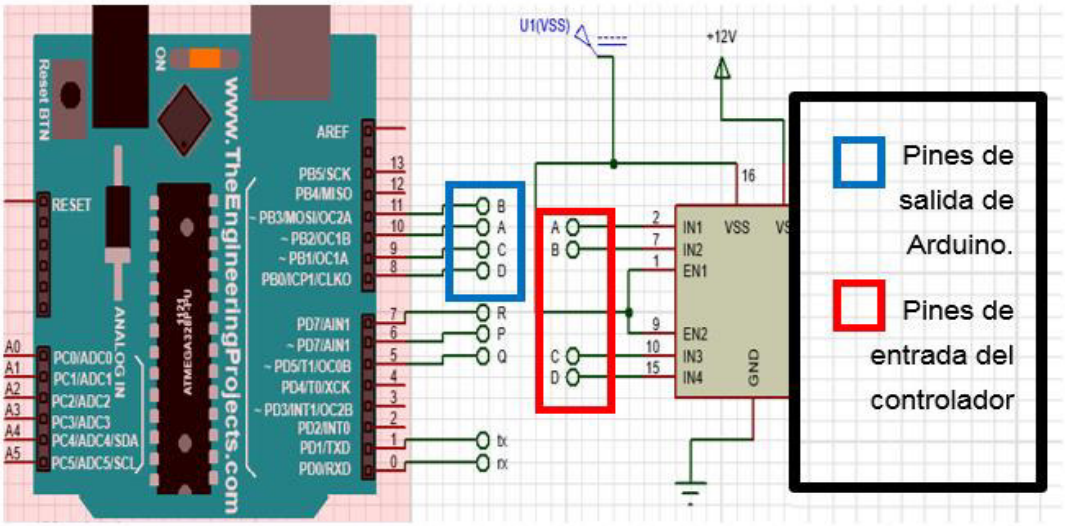


Figura 3-5 Conexión del motor *Nema*-23 con *Arduino* UNO

Al circuito se añadió un botón, el cual permite habilitar y deshabilitar el modo de funcionamiento personalizado, como se puede ver en la Figura 3-6; mientras el botón se encuentre deshabilitado, funciona el modo automático, caso contario funciona el modo personalizado.

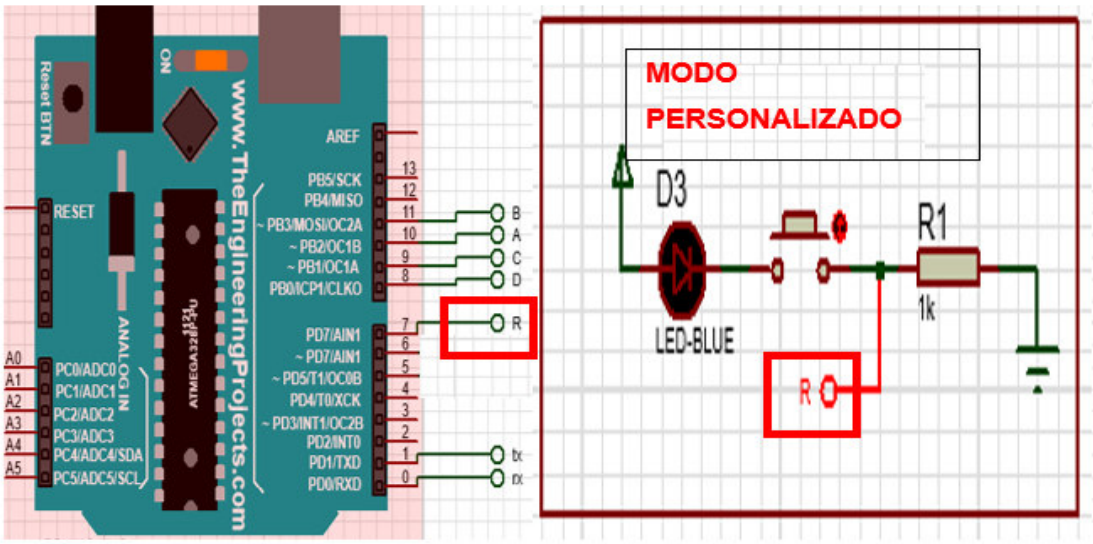


Figura 3-6 Botón habilitador del modo personalizado en el circuito de persianas

Al presionar el botón mencionado, se activa la comunicación inalámbrica entre el circuito de persianas y la aplicación móvil, para llevar a cabo esta acción, se realiza la conexión entre el módulo *Bluetooth* y *Arduino* UNO, como se puede apreciar en la siguiente Figura 3-7.

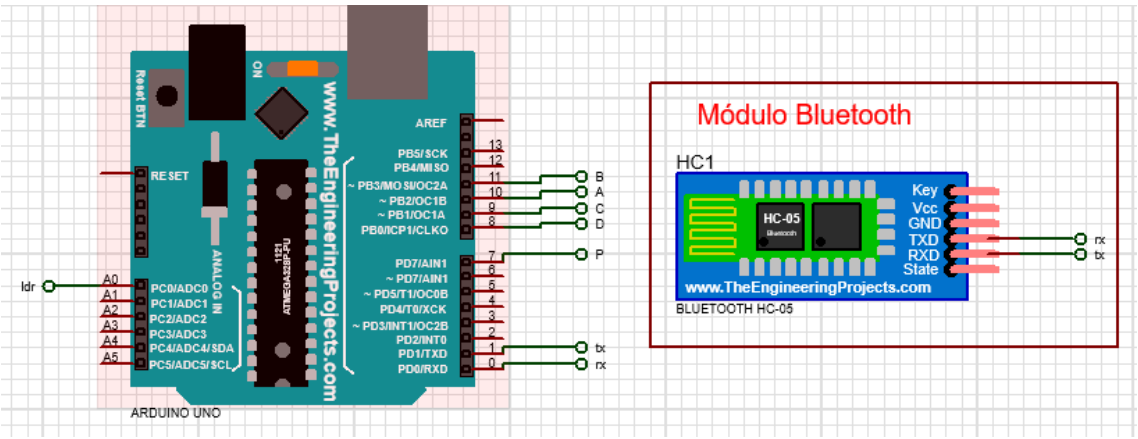


Figura 3-7 Conexión entre el módulo *Bluetooth* y el circuito de persianas

A este circuito se añadió una fotoresistencia que actúa como una resistencia variable, en la Figura 3-8 se puede observar cómo el sensor *LDR* cumple la función de captar la cantidad de luz en el entorno de trabajo. Para leer los valores captados se usó un circuito divisor de tensión el cual, en función de la variación de la fotoresistencia, varía el voltaje de salida (Vout) que se encuentra entre la resistencia (R4) y la fotoresistencia (*LDR*).

Con ello, el modo de operación automático cumple con el rol de cerrar o abrir las persianas haciendo uso de la información obtenida por el *LDR*, este modo de trabajo será suspendido si el usuario hace uso de la aplicación móvil.

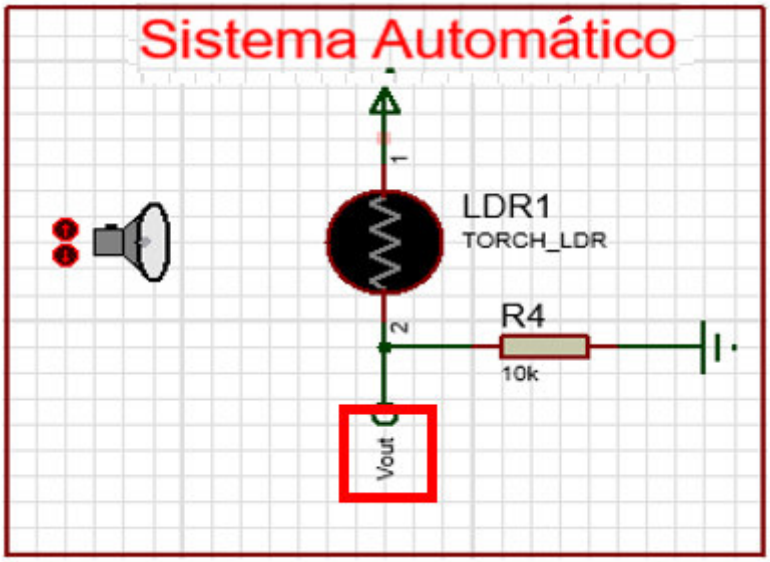


Figura 3-8 Circuito para lectura de los valores del *LDR*

A continuación, se presenta la Ecuación 3.1 que permite obtener el valor de voltaje de salida del circuito divisor de tensión, este valor lee el microcontrolador y en función de ello ejerce una acción previamente determinada en el código fuente.

$$\mathrm{Vout} = \frac{R_{LDR}}{R1 + R_{LDR}} * V_{LDR}$$

Ecuación 3.1 Valor de tensión en función de la lectura del *LDR* [1]

Donde:

$\mathrm{R_{LDR}}$: Valor de la resistencia del *LDR*

$\mathrm{R1}$: 10 (KΩ)

Vout : Voltaje de salida que leer el *Arduino*

$\mathrm{V_{LDR}}$: 5 (V)

El circuito divisor de tensión que se muestra en la Ecuación 3.1 tiene como objetivo permitir el desarrollo del sistema en el modo de funcionamiento automático, donde *Arduino UNO* al estar en operación automática lee el valor del voltaje de salida que entrega el circuito en mención.

Para que el motor gire en sentido horario o antihorario, el valor del voltaje de salida debe ser mayor o igual a 2.5 (V), obteniendo como resultado que las persianas empiecen a abrir, y si el valor es menor a 2.5 (V) las persianas se cerrarán.

Diseño del circuito de control de luces

Para controlar el apagado, encendido y la variación de la intensidad del sistema luminario, se procedió a diseñar un sistema de control capaz de realizar dichas acciones. Este diseño cuenta con tres etapas las cuales permiten tener el control total de la iluminaria de las oficinas previamente mencionadas:

- Primera etapa: circuito cruce por cero

Este circuito es muy importante debido que se genera un pulso de sincronización relacionado con el ángulo de fase de corriente alterna para poder controlar la intensidad luminosa de las luces.

Para lograr controlar la intensidad luminosa de las luces se debe conocer el voltaje y la frecuencia de la corriente alterna, en este caso al estar en Ecuador se trabaja con un voltaje de 120 (VAC) con una frecuencia de 60 (HZ)

Para el diseño se utilizó un transformador de 120 (VAC) a 12 (VAC) para proteger los demás elementos que conforman el circuito; una vez conectado el transformador a la red eléctrica, se podrá visualizar las señales a través del osciloscopio en *Proteus,* como se puede apreciar en la Figura 3-9 las señales tienen diferente voltaje, pero la misa frecuencia.

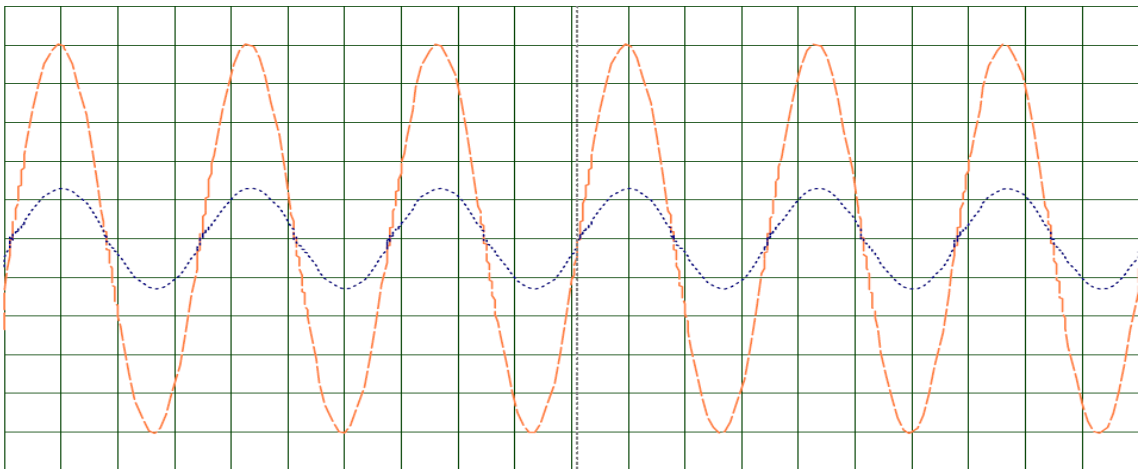


Figura 3-9 Voltaje a la salida del transformador

En este circuito el puente diodos es uno de los elementos más importante debido a que va a permitir tener una señal rectificada de onda completa, en la Figura 3-10 se muestra la señal de salida con color lila que pertenece a la señal resultante del puente de diodos y la señal de entrada de corriente alterna de color rojo.

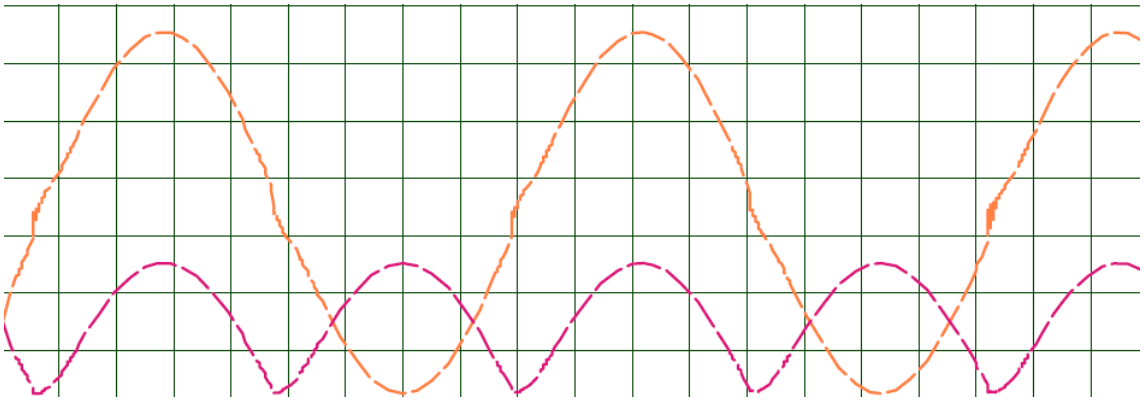


Figura 3-10 Voltaje a la salida del puente de diodos

Una vez que se obtenga la señal de salida del puente de diodos, se hará uso de un optoacoplador PC817 el cual permitirá obtener un pulso cada vez que la señal de corriente alterna cambie de polaridad enviando un pulso a *Arduino* UNO con el cual se podrá activar el *TRIAC* y controlar las luces.

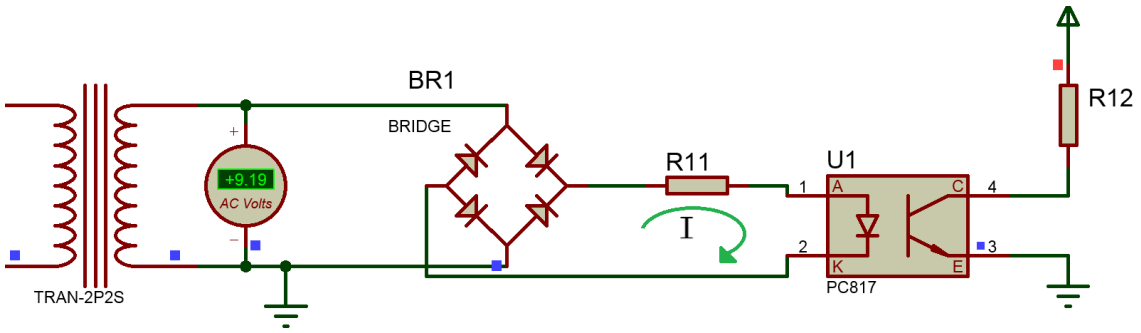


Figura 3-11 Circuito de disparo

En la Figura 3-11, se utilizó la ley de voltaje de Kirchhoff, obteniendo la siguiente Ecuación 3.2

$$Vf = V_{R11} + V_{led}$$

Ecuación 3.2 Ley de voltaje de Kirchhoff [14]

Donde

Vf : 13 (VAC) voltaje de entrada

V_{R11} : voltaje R11 (V)

V_{led} : 3 (V) voltaje *LED*

Despejando V_{R11} de la Ecuación 3.2 se tuvo como resultado

$V_{R11} = Vf - V_{led}$

Remplazando valores de la Ecuación 3.2

$$V_{R11} = 13(V) - 3(V)$$

$$V_{R11} = 10(V)$$

Para encontrar el valor de la resistencia del optoacoplador se utiliza la Ecuación 3.3

$$V = I * R$$

Ecuación 3.3 Ley de Ohm

Donde:

V : Voltaje (V)

I : Intensidad (A)

R : Resistencia (Ω)

Despejando la resistencia de la Ecuación 3.3

$$R_{11} = \frac{V_{R11}}{I}$$

$$R_{11} = \frac{10(V)}{2.8(mA)}$$

$$R_{11} = \frac{10(V)}{2.8(mA)}$$

$$R_{11} = 3.5(\mathrm{K\Omega})$$

A continuación, se procede a calcular la potencia que la resistencia R11 disipa.

$$\mathrm{P} = \mathrm{V} * \mathrm{I}$$

Ecuación 3.4 Calculo de la potencia en función del voltaje y la corriente [15]

Donde

P : potencia de R11 (W)

V : voltaje R11 (V)

I : corriente de activación (A)

Remplazando valores de la Ecuación 3.4 se tiene como resultado

$$P = 10(V) * 2.8(mA)$$

$$P = 0.028\ (W)$$

En conclusión, la resistencia R11 tendrá un valor de 3.5 (KΩ) y de 0.25 (W).

Al momento que la señal rectificada ingrese al optoacoplador PC817, se activará el *LED* infrarrojo que posee, saturando al fototransistor obteniendo a la salida un pulso de 0 (V) a 5 (V) como se puede ver en la Figura 3-12.

La señal de color azul resultante del circuito de disparo será leída por *Arduino UNO,* la cual indica al circuito de control cuando la señal eléctrica cambie de polaridad con la finalidad de obtener un mejor control al momento de variar la cantidad de energía que reciben las luces.

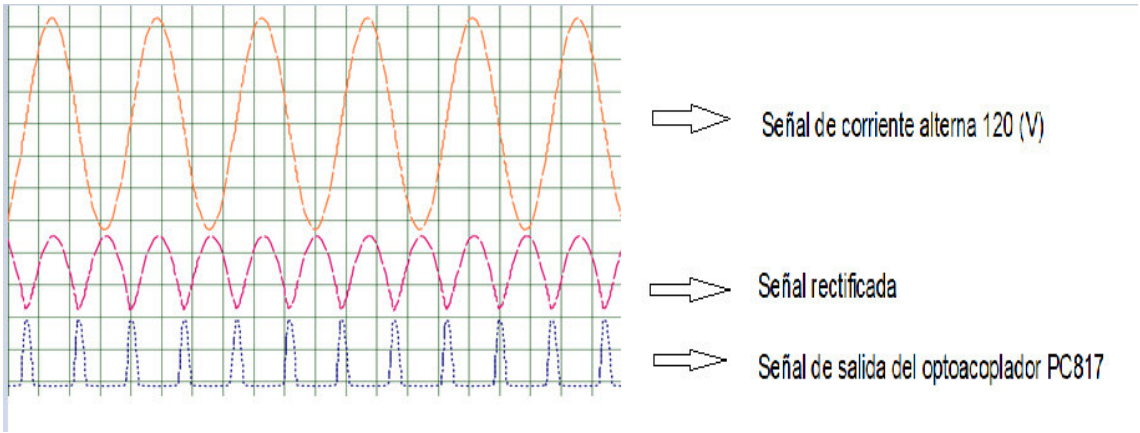


Figura 3-12 Pulso de control

- Segunda etapa: circuito de disparo

En la Figura 3-13 se muestra la forma en que se encuentra conectado el circuito de disparo. Para el diseño de este circuito, el cual es el responsable de entregar energía a las luces, se usó un optoacoplador MOC3021 el cual tiene un diodo *LED* infrarrojo que

al momento de encenderse activa un *FOTOTRIAC* que a su vez activa el *TRIAC* obteniendo como resultado el control de las luces.

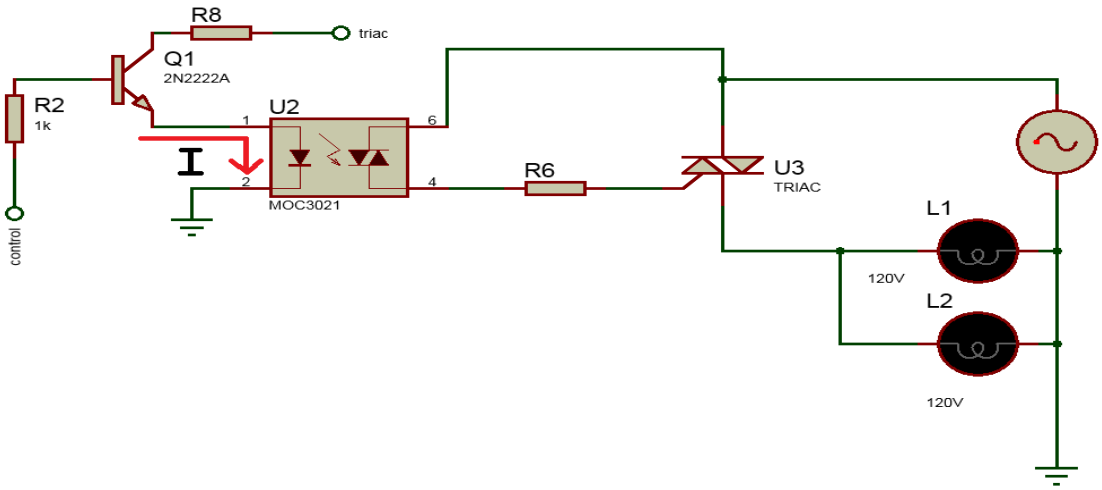


Figura 3-13 Circuito de disparo

Para que todos los elementos funcionen de una manera adecuada se, procedió a hacer el cálculo de las resistencias de protección tanto para el optoacoplador MOC3021 como para el *TRIAC* BT136.

Cálculo de la resistencia de protección del optoacoplador MOC3021

Donde:

Vf : 5 (V) voltaje de entrada

V_{R8} : voltaje R8 (V)

I_{led} : 1.5 (V) voltaje *LED*

Usando Ley de voltaje de Kirchhoff en el circuito de la Figura 3-13 y usando la Ecuación 3.2

$$V_{R8} = Vf - V_{led}$$

$$V_{R8} = 5(V) - 1.5(V)$$

$$V_{R8} = 3.5(V)$$

Para encontrar el valor de la resistencia de protección se usó la Ecuación 3.3

$$R_8 = \frac{V_{R8}}{I}$$

$$R_8 = \frac{3.5(V)}{35(mA)}$$

$$R_8 = 100(\Omega)$$

Cálculo de resistencia de protección para el *TRIAC* BT136

Vf : 120 (VAC) voltaje de entrada

V_{R6} : voltaje R6 (V)

I_{led} : 12 (V) voltaje *LED*

Usando la Ley de voltaje de Kirchhoff en el circuito anterior, Figura 3-13 se obtiene que:

$$V_{R6} = Vf - V_{led}$$

$$\mathrm{V_{R6}} = 120(\mathrm{Vac}) - 12(\mathrm{V})$$

$$\mathrm{V_{R6}} = 108(\mathrm{V})$$

Para encontrar el valor de la resistencia de protección se usó la Ecuación 3.3

$$\mathrm{R_6} = \frac{\mathrm{V_{R6}}}{\mathrm{I}}$$

$$\mathrm{R_6} = \frac{108(\mathrm{V})}{0.1(\mathrm{A})}$$

$$\mathrm{R_6} = 1080(\Omega)$$

$$\mathrm{R_6} = 1(\mathrm{K}\Omega)$$

Una vez finalizado el cálculo de las resistencias de protección, se procederá a realizar el circuito control responsable de que todos los elementos funcione en forma homogénea.

- Tercera etapa: Circuito de control

Para el diseño del circuito de control se estableció tres modos de funcionamiento

- Modo automático
- Modo manual
- Modo personalizado

Modo automático

Como se puede ver en la Figura 3-14 el modo automático cuenta con un pulsador que al presionarlo activa un diodo *LED* indicando al usuario que este modo esta activado.

Para controlar el encendido o apagado de las luces de manera automática, se usó una fotoresistencia responsable de captar la intensidad de luz que hay en el ambiente.

Para lograr esto, *Arduino* estará leyendo constantemente la fotoresistencia con la finalidad de encender o apagar las luces en función de la cantidad de luz que incide en el entorno.

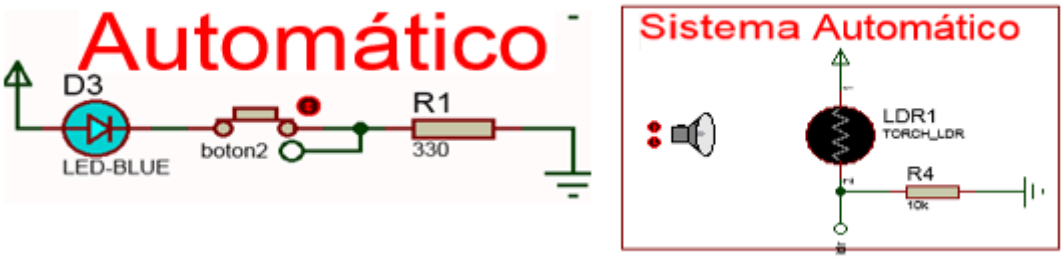


Figura 3-14 Botón automático

Modo manual

Para activar esta modalidad, se colocó un pulsador llamado “botón manual”, como se muestra en la Figura 3-15. Al momento de ser presionado este pulsador, un *LED* se enciende mostrando al usuario que esta modalidad está en funcionamiento, esta función cuenta con un potenciómetro que permite variara la intensidad luminosa de las luces en función de la posición de dicho potenciómetro.

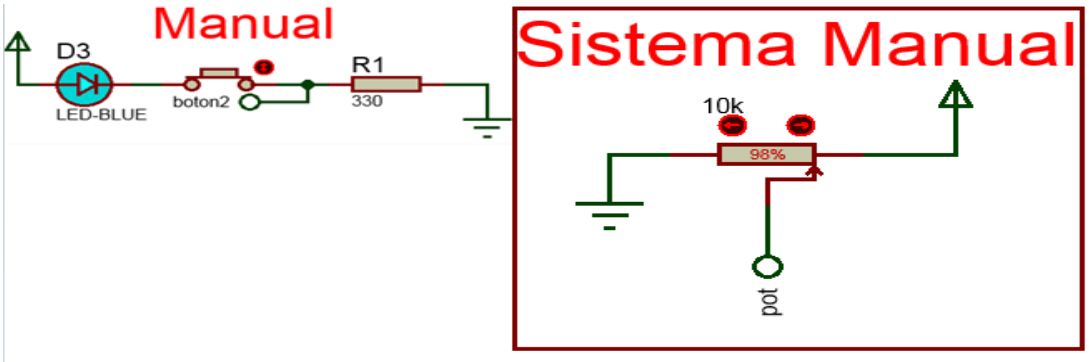


Figura 3-15 Botón manual

Para lograr que *Arduino* envíe la señal de control hacia el circuito de disparo en el momento exacto, constantemente se estará leyendo el pulso que envíe el circuito cruce

por cero cada vez que la señal de corriente alterna pase por su punto mínimo con la finalidad de que *Arduino* esté sincronizado con la frecuencia de la corriente alterna.

Para calcular el periodo que la señal demora en completar un ciclo, se debe conocer su frecuencia en Ecuador la frecuencia de trabajo es de 60 (Hz), por lo tanto.

$$T = \frac{1}{f}$$

Ecuación 3.5 Cálculo del tiempo [16]

Donde

T : periodo (s)

f : frecuencia (Hz)

usando la Ecuación 3.5 se obtendrá el tiempo que la señal eléctrica toma en completar un ciclo.

$$T = \frac{1}{60Hz}$$

$$T = 16.666\ (ms)$$

Como se observa en la Figura 3-12 en la salida del puente de diodos, a la señal le toma la mitad del tiempo en cumplir un ciclo de trabajo a comparación de la señal original, por lo que esta señal se demora 8.333 (ms) en cumplir un ciclo de trabajo. Por lo tanto, cada vez que la señal del puente de diodos alcancé su valor mínimo la seña eléctrica cambia su polaridad.

Al tener esta información por parte del circuito cruce por cero, *Arduino* procesara esta señal con la finalidad de sincronizar la frecuencia de la corriente alterna.

Como se puede visualizar en la Figura 3-16 al momento de variar la posición del potenciómetro. *Arduino* enviará un pulso de activación al circuito de disparo con la finalidad de variar la cantidad de energía entregada a las luces en función de la posición del potenciómetro.

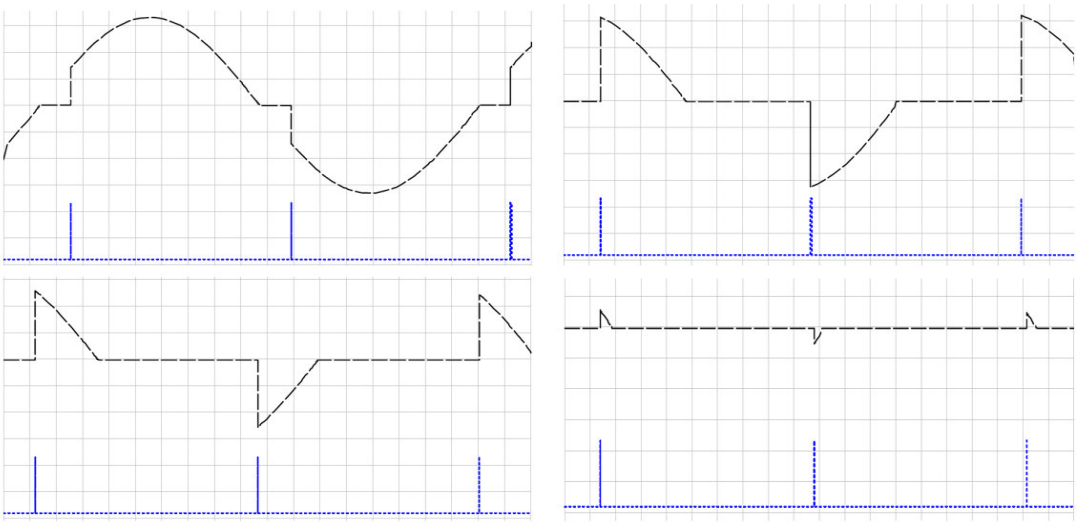


Figura 3-16 Variación de la señal eléctrica

Donde la señal de color azul es el pulso que *Arduino* entrega al circuito de disparo para

Modo personalizado

Para activar este modo personalizado tanto el botón manual como el botón automático, deben estar en estado bajo como se muestra en la Figura 3-17. Al estar encendido el *LED* del modo personalizado, indicará al usuario que esta modalidad está activada y se podrá controlar las luces a través de una conexión *Bluetooth.*

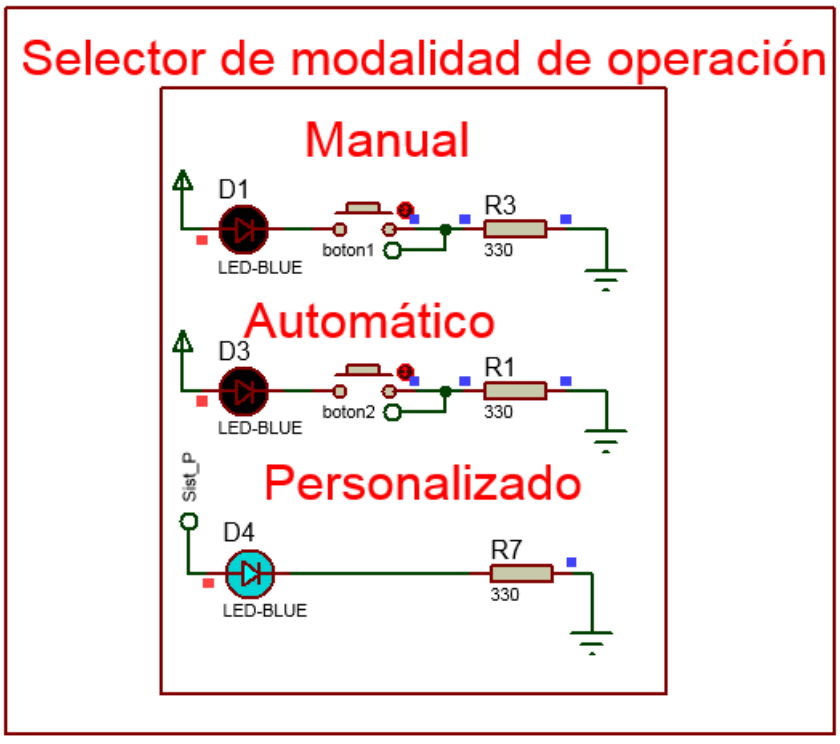


Figura 3-17 Modo personalizado

El usuario podrá controlar las luces de forma inalámbrica debido a que este sistema cuenta con un módulo *Bluetooth* que permite el intercambio de datos entre la aplicación móvil y el circuito de luces.

Diseño del diagrama de flujo para el circuito de control de persianas

Para realizar el sistema de control y llevar a cabo la creación del código fuente, es necesario crear un diagrama de flujo el cual permite dar una visión de lo que se pretende hacer mediante programación; a continuación, en la Figura 3-18 se muestra el diagrama de flujo para el sistema de control de persianas. Donde se puede verificar que el sistema tiene dos formas en las que puede operar ya sea modo automático o modo personalizado.

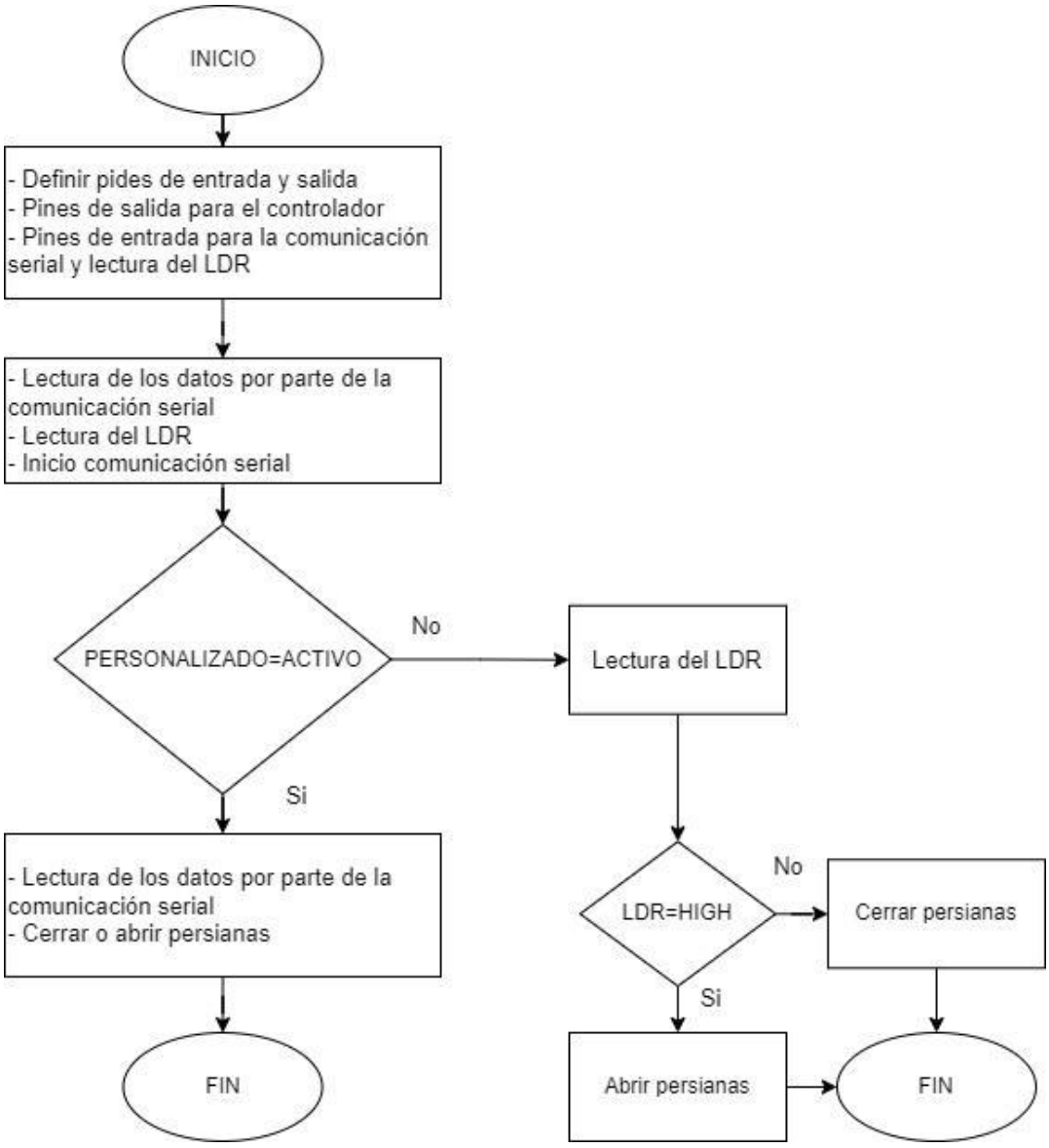


Figura 3-18 Diagrama de flujo del circuito de control persianas

Diseño del diagrama del flujo para el circuito de control de luces

Para realizar el código que se va a programar en *Arduino*, es necesario hacer uso de un diagrama de flujo donde se muestra el funcionamiento de todos los elementos que constituyen el sistema de luces, como se puede aprecia en la Figura 3-19 se puede notar cada una de las tres modalidades que presenta este circuito de luces.

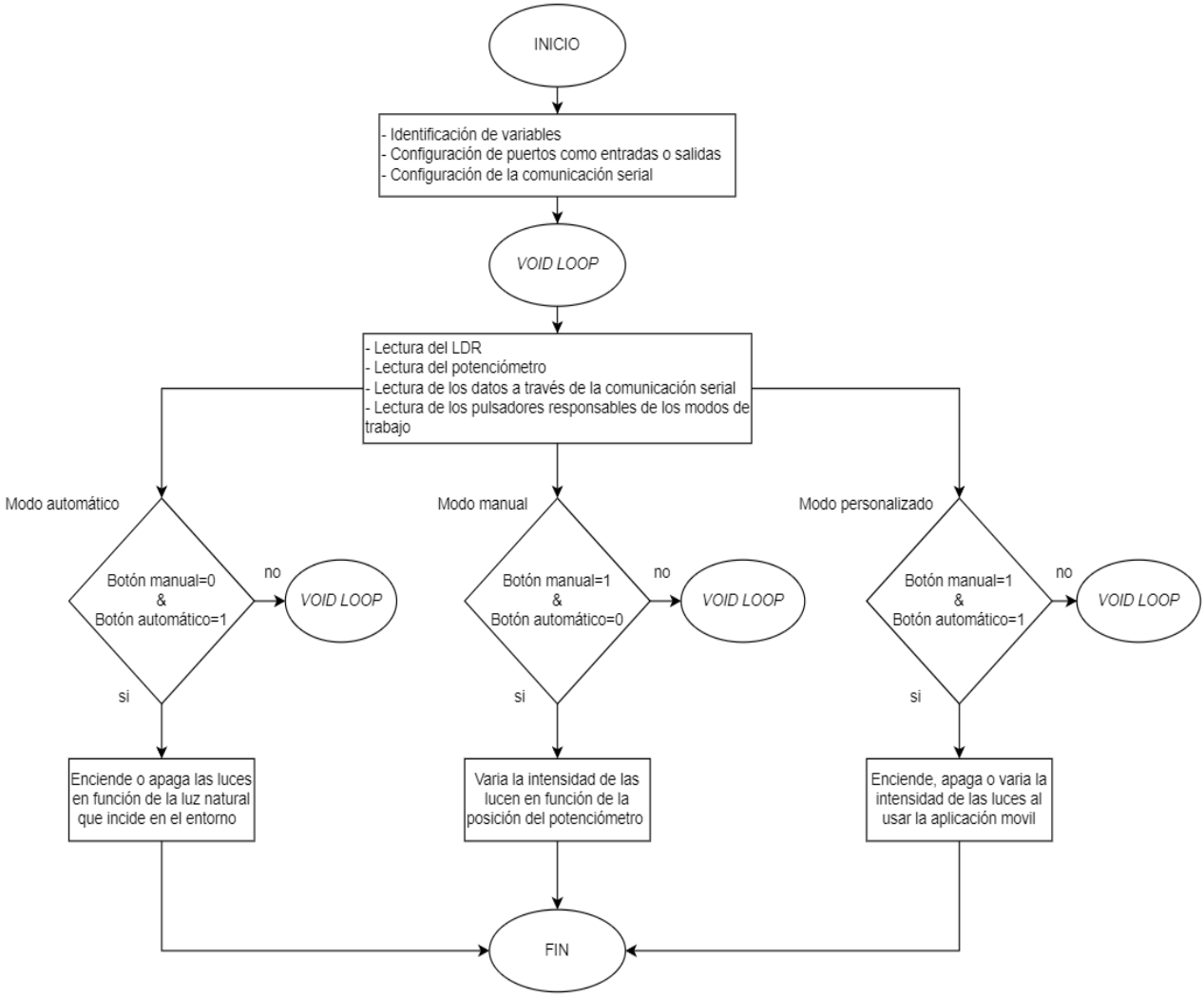


Figura 3-19 Diagrama de flujo del circuito de control de luces

Descripción del código fuente para el sistema de control de persianas

Para la programación de código fuente que controlará el sistema de persianas, se inicia declarando variables, y añadiendo la librería que permitirá controlar el motor a pasos, como se puede ver en la Figura 3-20.

```
#include <Stepper.h> // librería para controlar motores paso a paso
#define STEPS 200 // número de pasos que necesita para dar una vuelta. 200 en nuestro caso
Stepper stepper(STEPS, 8, 9, 10, 11); //Stepper nombre motor (número de pasos por vuelta, pins de control)

const int PERSONALIZADO1=7;  //Declaracion del pin que activara o desactivara el modo personalizado
const int FINCARRERA1=6;   //Declaracion del pin de fin de carrera
const int ONANDOFF1=5;   //Declaracion del pin que abrira a cerrara las persianas dependiendo de la descision del usuario

char cambio2;

int onandoff;      // constante que leera el valor pin 5
int fin;      // constante que leera el valor pin 6
int personalizado;  // constante que leera el valor pin 7

const int LDRPin = A0;   //Pin del LDR
int LDRLECTURA;
```

Figura 3-20 Declaración de variables y adición de la librería stepper.h

A continuación, como se muestra en la Figura 3-21 se procede a configurar la comunicación serial, determinar la velocidad del motor, y configurar los pines.

```
void setup()
{
    pinMode(7, INPUT);
    Serial.begin(9600);
    stepper.setSpeed(20);
}
```

Figura 3-21 Configuración de los pines

En la siguiente Figura 3-22, se describe el proceso de funcionamiento cuando el sistema de control de persianas basa su trabajo en el modo personalizado es decir cuando, mantiene la comunicación con el celular móvil.

```
 personalizado=digitalRead(PERSONALIZADO1);

if (personalizado==LOW)
 delay (1000);
{
INICIO:
     if (Serial.available()>0)
     cambio2 = Serial.read();


  if(cambio2=='A')
  {
            stepper.step(380);
            goto APAGARMOTOR;
  }

  if(cambio2=='C')
  {
            stepper.step(-380);
            goto APAGARMOTOR;
  }

}
```

Figura 3-22 Líneas de código para el Modo de funcionamiento personalizado

En cuanto para el modo automático, se procede a testear en primera instancia el pin personalizado, dado que, si su valor es alto, entra en funcionamiento este modo, donde el *LDR* censa los valores de luz y en función de eso varía su resistencia, y al estar dentro de un circuito divisor de tensión variará un voltaje, el cual *Arduino* leerá y entorno a los resultados girará el motor ya sea para cerrar o abrir las personas, y se mantendrán en ese estado hasta que el nuevo valor sea diferente. En la Figura 3-23 se puede verificar el código fuente de la parte mencionada.

```
if((personalizado==HIGH)||(cambio2=='P'))
 {
        delay(10000);

        LDRLECTURA = analogRead(LDRPin);
        if (LDRLECTURA <600)  // CERRANDO LAS PERSIANAS
             {
             CERRAR:
             stepper.step(380);
             delay(500);
             goto BOTOM1;
             }
         else                    //ABRIMOS LAS PERSIANAS
             {
             ABRIR:
             stepper.step(-380);
             delay(500);
             goto BOTOM;
             }
                          BOTOM:
                          {
                               if (Serial.available()>0)
                               cambio2 = Serial.read();
                               if((cambio2=='A')||(cambio2=='C') )
                               {goto APAGARMOTOR;}
                                PORTB= B00000000;
                                delay(100);
                                LDRLECTURA = analogRead(LDRPin);
                                if (LDRLECTURA >=600)
                                goto BOTOM;
                                else goto CERRAR;
                          }

                          BOTOM1:
                          {
                               if (Serial.available()>0)
                               cambio2 = Serial.read();
                               if((cambio2=='A')||(cambio2=='C') )
                               {goto APAGARMOTOR;}
                               PORTB= B00000000;
                               delay(100);

                               LDRLECTURA = analogRead(LDRPin);
                               if (LDRLECTURA <600)
                               goto BOTOM1;
                               else goto ABRIR;
                          }
         APAGARMOTOR:
         {
         cambio2 = 0;
         PORTB= B00000000;

         }

 }
```

Figura 3-23 Código de programación en modo automático subcircuito de persianas

Descripción del código de programación del sistema de control de luces

El código inicia con la asignación de pines donde se encuentran conectados los pulsadores, sensores, el circuito cruce por cero y el circuito de disparo; también se asigna las variables para guardar los datos de los sensores y de la aplicación móvil, en la Figura 3-24, se puede apreciar la parte mencionada.

```
int ldr=A0; /// pin para leer el sensor ldr
int pot=A1; /// pin para leer el potenciometro
int cruce_cero = 2; // este es el pin para detectar el cruce por cero de la señal AC
int triac = 4; // pin para controlar el circuito de disparo
int valorpot; /// variable para guardar el valor del potenciometro
int valorldr; /// variable para guardar el valor del ldr
int tiempo_pot = 0; //variable para medir los tiempos de disparo para activar el triac en funcion del potenciometro
int manual=7; /// pulsador para activar el modo manual
int automatico=8; /// pulsador para activar el modo automatico
int valor1; // variable para el estado del pin 7
int valor2; // variable para el estado del pin 8
int dato_bt; // variable para guardar los valores enviados por la aplicacion
int tiempo_bt; // variable para guardar los tiempos de disparo para activar el triac en funcion del bluethoot
int control = 3; // pin para activar el timepo de disparo del triac
```

Figura 3-24 Asignación de pines y variables del diseño del programa

En el *void setup* se escriben las líneas de código correspondiente al inicio de la comunicación para el módulo *Bluetooth*, se define a los pines de *Arduino* como entradas o salidas como se puede verificar en la Figura 3-25.

```
void setup() {
  //inicio de la comunicacion con el modulo bluetooth
 Serial.begin(9200);

 // pines asignados a los circuitos e potencia
 pinMode(cruce_cero, INPUT);
 pinMode(triac, OUTPUT);
 pinMode(control, OUTPUT);
 digitalWrite(control, LOW);
 digitalWrite(triac, HIGH);

 // pines asignados a los pulsadores
 pinMode(manual,INPUT);
 pinMode(automatico,INPUT);

 //activacion de la interrupcion
 attachInterrupt(digitalPinToInterrupt(cruce_cero), Cambio,FALLING);
}
```

Figura 3-25 Configuración de pines como entradas o salidas

En la estructura del *void loop* se inicia con las lecturas del sensor *LDR* al igual que la lectura de los pines de entrada de *Arduino* UNO, en la Figura 3-26 se puede apreciar cómo las lecturas se almacenan en su respectiva variable.

```
//asignacion de variables para guardar los datos de los sensores
valor1=digitalRead(manual);
valor2=digitalRead(automatico);
valorldr = analogRead(ldr);
valorpot = analogRead(pot);
```

Figura 3-26 Lectura de los sensores

Para leer los datos que provengan de la aplicación móvil, se lee el puerto serial de *Arduino* con la función (Serial.read()) almacenando estos datos en una variable, como se puede ver en la Figura 3-27.

```
//se asigna una varable donde se guardara las datos que provengan de la aplicacion movil
if((Serial.available()>0))
{dato_bt = Serial.read();}
```

Figura 3-27 Lectura del módulo *Bluetooth*

En la en la Figura 3-28 se puede apreciar que los datos obtenidos por parte de los sensores y del módulo *Bluetooth* son responsables de controlar el sistema de luces; a continuación, se configura el estado de los pulsadores, mismos que al ser oprimidos permiten la activación de cualquiera de los tres modos de funcionamiento del sistema de control de luces.

```
/*configuracion de los pulsadores para activar el modo automatico la fotorresistencia
establecera el estado de encendido o apagado de las luces en funcion de la luz natural*/
if((valor2==1) && (valor1==0))
{
if(valorldr<=512){digitalWrite(control,HIGH); digitalWrite(triac,HIGH); }
if(valorldr>=513){digitalWrite(control,LOW); digitalWrite(triac,LOW);} }

//configuracion de los pulsadores para activar el modo personalizado
if((valor2==1) && (valor1==1)){
if(dato_bt=='a')
{
tiempo_pot=0;
digitalWrite(control,HIGH);
digitalWrite(triac,HIGH);
}

if(dato_bt=='b')
{
tiempo_bt=0;
digitalWrite(control,LOW);
digitalWrite(triac,LOW);
dato_bt=0;
} }
// valores asignados correpondientes a la posocion del deslizador de la aplicacion movil
if(dato_bt=='z') tiempo_bt = 10;
if(dato_bt=='y') tiempo_bt = 787;
if(dato_bt=='x') tiempo_bt = 1564;
if(dato_bt=='w') tiempo_bt = 2341;
if(dato_bt=='v') tiempo_bt = 3118;
if(dato_bt=='u') tiempo_bt = 3895;
if(dato_bt=='t') tiempo_bt = 4672;
if(dato_bt=='s') tiempo_bt = 5449;
if(dato_bt=='r') tiempo_bt = 6226;
if(dato_bt=='q') tiempo_bt = 7000;
if((valor2==1) && (valor1==1))
{digitalWrite(control,LOW); digitalWrite(triac,LOW);}
}
```

Figura 3-28 Configuración de los modos de funcionamiento

A continuación, se procede a describir la configuración de la interrupción externa de *Arduino* la cual ayuda a saber cada vez que la señal eléctrica cambie de polaridad con el fin sincronizar a *Arduino* con la frecuencia de la señal alterna.

Una vez monitoreada la frecuencia de dicha señal se enviará una señal de control hacia el circuito de diapiro con la finalidad de poder controlar la intensidad de la luz. Esta variación de intensidad luminosa está ligada por la posición del potenciómetro cuando el sistema está en modo manual, en caso de que se encuentre en modo personalizado las luces variarán su luminosidad en función del deslizador que tiene la aplicación.

En la Figura 3-29 se detalla la función que cada línea de código cumple en el sistema de control.

```
//interrupcion
void Cambio(){
//sistema manual para la variacion de la intensidad luminosa controlado por el potenciometro
if((valor2==0) && (valor1==1))
{
digitalWrite(control,HIGH);
tiempo_pot= map(valorpot, 0,1024, 0,7000); // esta funcion map permite relacionar la posicion del potenciometro con los valores dados
digitalWrite(triac, LOW); //  estado bajo al momneto que el pin2 detecte un flanco de bajada
delayMicroseconds(tiempo_pot); //retardo de un tiempo t en funcion del valor tiempo_pot
digitalWrite(triac, HIGH); // estado alto para activar el circuito de disparo
delayMicroseconds(10); // retador para mantener activado el circuito de disparo
digitalWrite(triac, LOW); //estado bajo para apagar el circuito de disparo
}

//sistema perzonalizado para la variacion de la intensidad luminosa controlado por la aplicacion movil
if((valor2==1) && (valor1==1))
{
digitalWrite(triac, LOW); //  estado bajo al momneto que el pin2 detecte un flanco de bajada
delayMicroseconds(tiempo_bt); //retardo de un tiempo t en funcion del valor tiempo_bt
digitalWrite(triac, HIGH); // estado alto para activar el circuito de disparo
delayMicroseconds(10); // retador para mantener activado el circuito de disparo
digitalWrite(triac, LOW); //estado bajo para apagar el circuito de disparo

}}
```

Figura 3-29 Control de luces en función del potenciómetro y de la aplicación móvil

Diseño del diagrama del flujo para la ampliación móvil

A continuación, en la Figura 3-30 se presenta, el diagrama de flujo el cual permitió el desarrollo y creación de la aplicación móvil

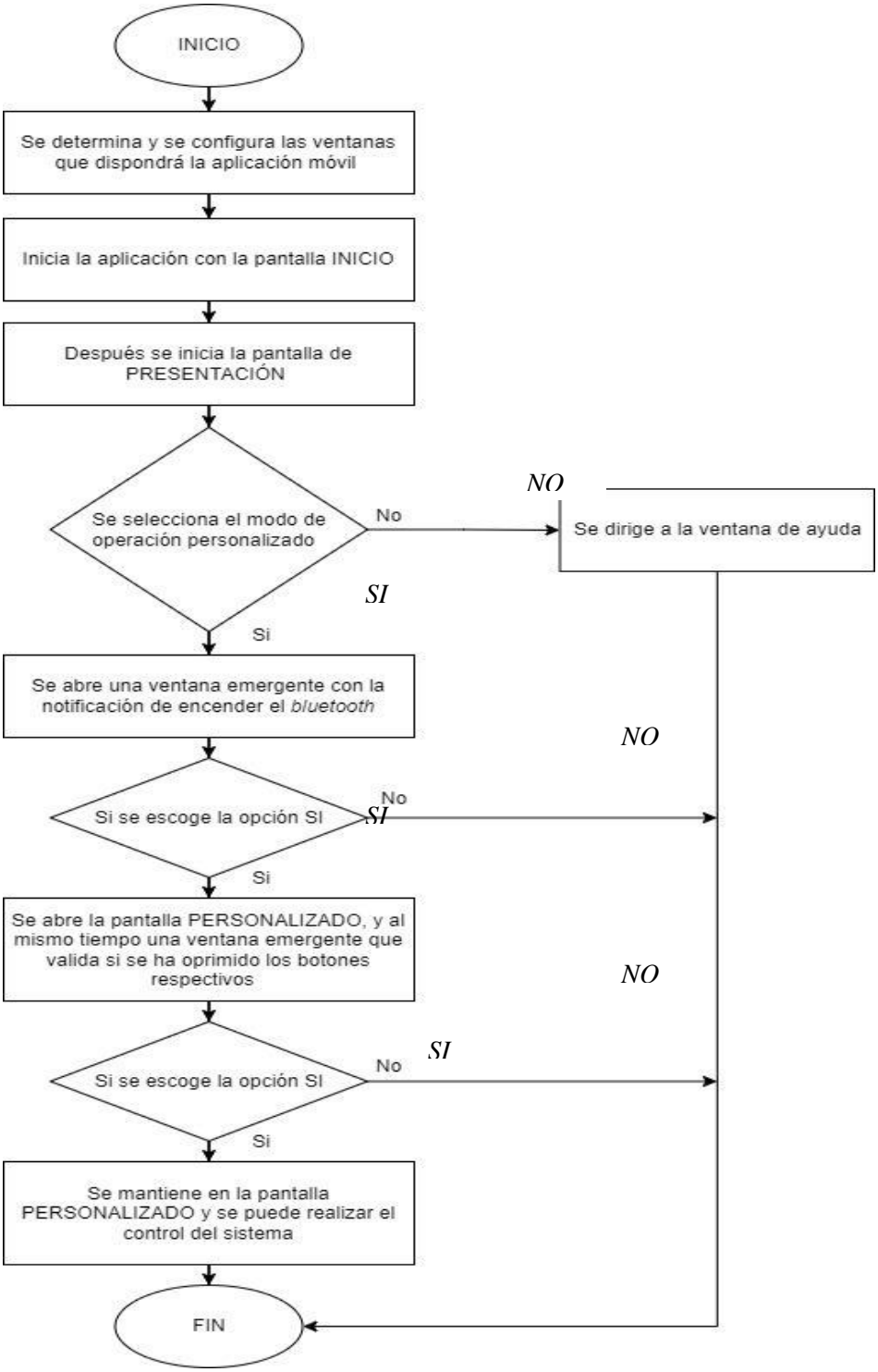


Figura 3-30 Diagrama de flujo de la aplicación móvil

Posterior a terminar con los diseños de los dos subcircuitos de control tanto el de persianas como el de luces, al igual que los diagramas de flujos que servirán para definir el funcionamiento de cada circuito de control, se procede a diseñar el cableado eléctrico.

Diseño del cableado eléctrico del sistema de control

Diseño del cableado sistema de control de persianas

Una vez analizado los elementos electrónicos que se utiliza, en el sistema de control de persianas, se procede a distribuir los elementos electrónicos en el área de trabajo de la misma manera el cableado electrónico que se utiliza para el sistema de control, ver imagen Figura 3-32.

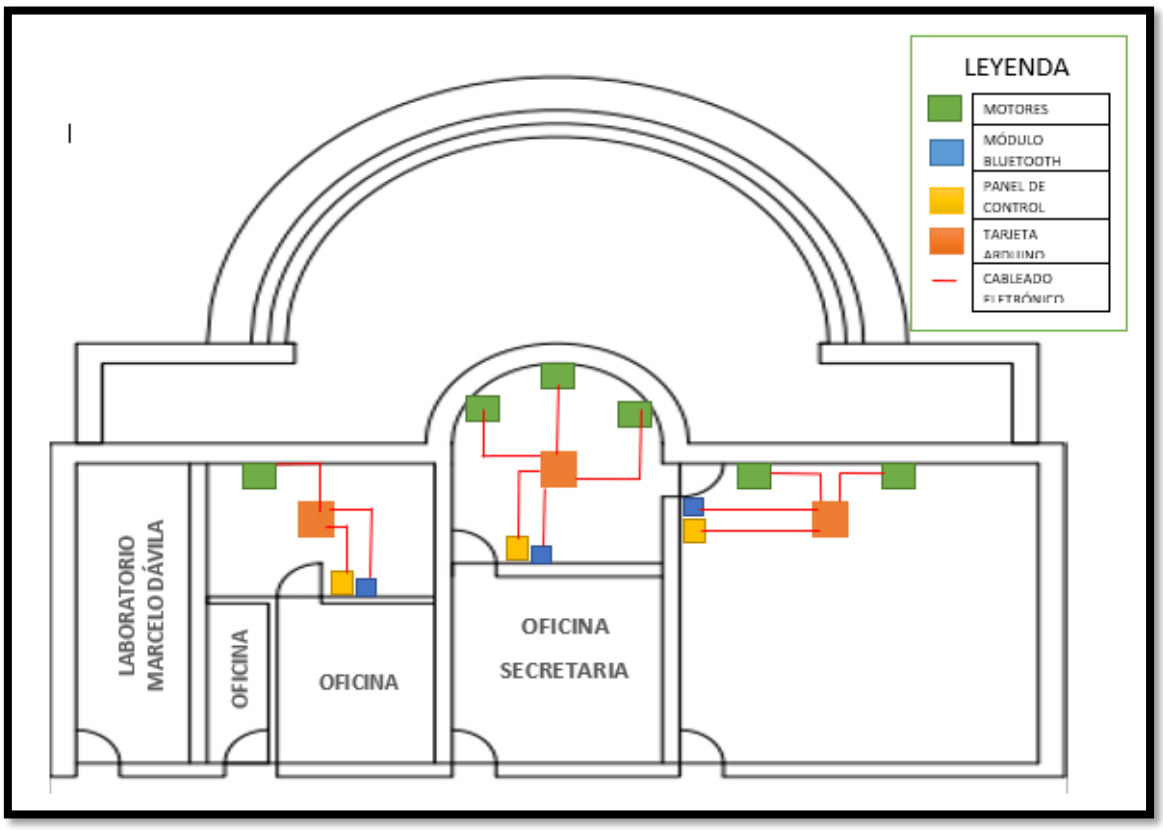


Figura 3-31 Distribución de elementos electrónicos en el área de trabajo

Diseño del cableado sistema de control de persianas

Al igual que el sistema control de persianas, se procede a distribuir los elementos electrónicos en el área de trabajo de la misma manera el cableado electrónico y eléctrico del sistema de control de luces, ver imagen **Figura 3-32**

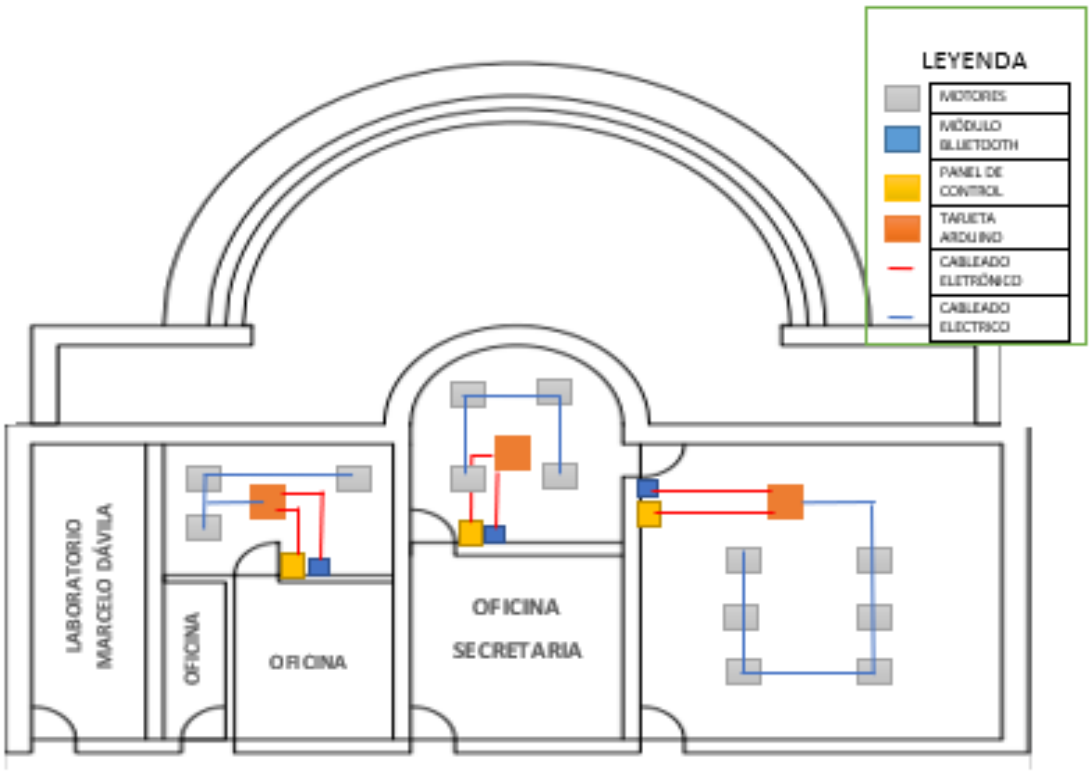


Figura 3-32 Cableado y distribución de elementos electrónicos en las oficinas

3.2 Determinar el *hardware* y *software* necesario para la simulación del sistema de control

Para determinar el tipo de placa de programación que el sistema de control debe usar, se realizó un análisis entre los dos tipos de tarjetas programables; una de ellas es *Arduino* y la otra es *Raspberry*, cada una de ellas se analizó en la versión clásica. En la Tabla 3.1 se pueden verificar las características generales de *Arduino* UNO y *Raspberry* Pi.

Tabla 3.1 Diferencias entre *Arduino* UNO y *Raspberry* PI [2] [3]

Características	Arduino	Raspberry
Procesador	ATMega328	ARM Cortex-A7
Velocidad	16 (MHz)	900 (MHz)
RAM	2 (KB)	1 (GB)
Entradas / Salidas	14 digitales y 6 analógicas	27 pines GPIO

Software de programación	Arduino IDE	GNU/Linux, Fedora, Raspbian.
Precio	7 USD	70 USD

Conforme a lo detallado en el apartado anterior, se decidió utilizar la placa *Arduino* UNO, ya que las características que se detallan son suficientes para el desarrollo del proyecto, no obstante, la tarjeta de Rasberry PI cuenta con características de alto rendimiento para la creación del sistema por lo que se desaprovecharía el uso eficiente de la placa mencionada. Una vez determinado el tipo de placa que se usa para el desarrollo del proyecto, se procede a analizar la versión que se debe utilizar, para ello se realizó un análisis de las tarjetas de *Arduino* UNO, Leonardo y Mega, en la Tabla 3.2 se puede apreciar las características generales de cada versión.

Tabla 3.2 Comparación entre *Arduino* UNO, Leonardo y Mega [2]

Características	*Arduino*		
Versión	UNO	Leonardo	Mega
Procesador	ATmega328P	ATmega32U4	ATmega2560
Velocidad / CPU	16 (MHz)	16 (MHz)	16 (MHz)
Entradas / Salidas analógicas	6	12	16
Características	*Arduino*		
Versión	UNO	Leonardo	Mega
Entradas / Salidas digitales	14/6	20/7	54/15
RAM	2 (KB)	2.5 (KB)	8 (KB)
FLASH	32 (KB)	32 (KB)	256 (KB)
Precio	7 USD	30 USD	60 USD

A fin de seleccionar la versión de *Arduino*, en primera instancia se tomó en cuenta la cantidad de memoria que dispone cada versión y considerando la complejidad del proyecto el cual no es muy alto y evaluando que la cantidad de código de programación no es muy extenso, se decidió elegir una versión con capacidad de memoria limitada, dejando de lado la versión Mega de *Arduino*.

Al considerar que el proyecto se va a dividir en dos subcircuitos, y cada subcircuito necesita de 9 pines de entrada o salida, dando como resultado la elección de *Arduino* UNO con la placa con las mejores características para este proyecto.

Para la comunicación inalámbrica se analizaron dos módulos *Bluetooth*, el uno es el HC-05 y el otro es el HC-06 en la Tabla 3.3 se puede verificar las diferencias entre los dos módulos.

Tabla 3.3 Comparación entre el módulo *Bluetooth* HC-05 y HC-06 [6]

Características	Módulo Bluetooth	
Versión	HC-05	HC-06
Alimentación	3.3 – 5 (V)	3.3 – 5 (V)
Configuración	Maestro – esclavo	Esclavo
Precio	5 USD	5 USD

Se puede apreciar que los dos módulos tienen las mismas características a excepción del tipo de configuración que maneja cada uno.

Para el desarrollo del proyecto se puede utilizar cualquier módulo; sin embargo, se seleccionó el módulo HC-06 debido a que el proyecto solo se va hacer uso de la configuración en modo esclavo.

Para captar la intensidad luminosa que existe en el entorno, se determinó el uso de una fotoresistencia *LDR* debido a que es sensible a la luz que incide sobre este elemento brindando una resistencia mayor o menor en función de la luz que recibe.

A continuación, se describe las características técnicas de la fotoresistencia *LDR* expresadas en la Tabla 3.4.

Tabla 3.4 Características técnicas *LDR* [4]

Características	Valor
Resistencia (con luz)	10 (kΩ)
Resistencia (oscuridad)	1 (MΩ)
Disipación a 25 (°C)	100 (mW)
Voltaje máximo	150 (V)

Determinación del *hardware* del circuito de control de persianas

Para controlar la apertura y cierre de las persianas, se acopló al eje un motor a pasos. Para determinar el tipo de motor que se usa para el desarrollo de este proyecto se realizó un análisis entorno a los diversos tipos de motores a pasos que se dispone hoy en día, dando como resultado que actualmente los motores a pasos bipolares y unipolares son los que disponen de mayores prestaciones; a continuación, en la Tabla 3.5 se puede apreciar las diferencias entre los dos tipos de motores a pasos.

Tabla 3.5 Diferencia entre el motor a pasos bipolar y unipolar [5]

Motor a pasos bipolar	Motor a pasos unipolar
Presenta un mayor torque	Presenta un menor torque
Es de menor tamaño	Es de mayor tamaño
Es más económico	Es menos económico
Su control es más complicado, requiere una tarjeta con etapas de control en potencia y de giro	Su control es más simple al requerir únicamente un circuito de alimentación
Mayor anclaje debido a los embobinados	Menor anclaje
Presenta un mayor torque	Presenta un menor torque
Es de menor tamaño	Es de mayor tamaño

Con lo mencionado con anterioridad se decidió por utilizar el motor a pasos *Nema*-23, en la Tabla 3.6 se puede apreciar las características generales, donde la principal particularidad por la que se utilizó este motor es el torque que genera al estar en funcionamiento, mismo que facilita el movimiento de las persianas sin mayor dificultad. No obstante, otro beneficio que se tiene al trabajar con este tipo de motor es controlar la posición y el sentido de giro del eje del motor, esto se aprovecha al momento de cerrar o abrir las persianas ya que estas acciones se realizan de manera exacta.

Tabla 3.6 Características generales del motor *Nema*-23 [6]

Características	Unidad de operación
Tipo de motor	Bipolar
Ángulo de rotación del motor	1.8°
Torque	1.26 Nm (178.5oz.in) (12.85 Kg/cm)
Corriente máxima de fase	2 (A)

Voltaje	4.5 (V) mínimo por bobina
Inductancia	2.5 mH±20%(1KHz)
Resistencia	2.25 (Ω) por bobina

Para que el motor a pasos funcione, se debe utilizar un controlador el cual tiene como función principal proveer de corriente para que el motor gire con el torque necesario, de tal manera que se pueda mover las persianas, se conoce que la corriente máxima de fase con la que el motor debe operar es de 2 (A), esta información se puede apreciar en la Tabla 3.6.

En ese contexto, se analizaron varios controladores y se determinó utilizar el Driver L293D, ya que la corriente de salida que entrega cada canal es de 1.2 (A), suficiente para mover el motor *Nema*-23 con el torque deseado. En la Tabla 3.7 se puede constatar el valor de corriente de salida y las características eléctricas del controlador.

Tabla 3.7 Características eléctricas del Driver L293D [6]

Voltajes	Valores máximos
Rango de voltaje de alimentación	4.5 (V) – 36 (V)
Máximo voltaje soportado por los pines	7 (V)
Salida de corriente por canal pico	1.2 (A)
Salida de corriente por canal de forma continua	600 (mA)

Determinación del *hardware* de circuito de control de luces

Elementos para el circuito cruce por cero

Para reducir el voltaje de entrada de la red eléctrica, se determinó que un transformador de 120 (VAC) a 12(VAC) es un elemento de protección importante debido a que ayuda a proteger los elementos de baja tensión.

Para poder determinar los cambios de polaridad de la corriente alterna, se determinó que el puente de diodos 2W005 es un elemento que entrega a la salida una señal rectificada de onda completa.

El puente de diodos 2W005 presenta las características adecuadas debido a que se usa un transformador para disminuir la tensión de entrada de la red eléctrica a continuación, se detallan las características en la Tabla 3.8.

Tabla 3.8 Características técnicas del puente de diodos 2W005 [7]

Características	Valor
Voltaje máximo	50 (V)
Corriente máxima	2 (A)
Rango de temperatura de funcionamiento	-65 a +125 (°C)

Es necesario hacer uso de un componente electrónico que permita monitorear cada vez que la rectificada de onda completa tenga un valor de 0 (V) con la finalidad de tener una referencia cada vez que la corriente alterna cambie de polaridad. Para eso se selecciona un optoacoplador de la serie PC817, el cual está constituido por un *LED* infrarrojo y un fototransistor que cada vez la corriente alterna tenga un valor de 0 (V) envíe un pulso al *Arduino* UNO. Las características que se tomaron en cuenta para seleccionar este dispositivo electrónico se detallan en la Tabla 3.9.

Tabla 3.9 Características técnicas del optoacoplador PC817 [8]

Características	Valor
Voltaje de entrada máximo	6 (V)
Corriente de entrada máximo	50 (mA)
Voltaje de salida colector-emisor	35 (V)
Voltaje de salida emisor-colector	6 (V)

Elementos para el circuito de disparo

En esta etapa del circuito de control de luces se determinará los elementos necesarios para encender, apagar o variar la intensidad luminosa de las luces; para ello, se debe determinar un dispositivo que permita conmutar la corriente alterna en ambos sentidos para poder controlar la cantidad de energía que se va a entregar a las luces para ello un *TRIAC* es un dispositivo semiconductor que permite lograr esta función.

Las características técnicas del *TRIAC* BT136 se detallan en la Tabla 3.10.

Tabla 3.10 Características TRAIC BT136 [9]

Características	Valor
Voltaje de entrada máximo	600 (V)
Corriente Máxima de Disparo del *gate*	10 (mA)
Voltaje Máxima de Disparo del *gate*	1.5 (V)

Para separar *Arduino* UNO del *TRIAC* responsable de controlar las luces, se debe usar un dispositivo electrónico capaz de unir la parte de corriente alterna con la parte de bajo

voltaje, para lo cual se utiliza un optoacoplador MOC3021 que al estar compuesto por un *LED* infrarrojo y un *FOTOTRIAC* que permite pasar la corriente alterna en ambos sentidos, permitiendo que trabajen de forma simultánea el optoacoplador y *TRIAC*.

A continuación, se presenta las características del MOC3021 descritas en la Tabla 3.11

Tabla 3.11 Características técnicas MOC3021 [10]

Características	Valor
Voltaje de entrada máximo	3 (V)
Corriente de entrada máximo	50 (mA)
Voltaje máximo de apagado	400 (V)

Determinación del *software* de programación del sistema de control

Se determinó usar el entorno de programación IDE *Arduino* dado que se seleccionó las tarjetas programables de la misma marca. Cabe mencionar que para elegir las placas de *Arduino* se tomó en cuenta el *software* donde se llevará a cabo el desarrollo del código fuente. En ese contexto, se analizó dos posibles entornos de programación que permiten programar microcontroladores, el ya mencionado IDE *Arduino* y *MPLAB*.

MPLAB es un entorno de desarrollo integrado que permite programar en lenguaje ensamblador los microcontroladores *PIC*, al finalizar la programación de código fuente se debe disponer de un dispositivo extra que permita grabar el programa en el microcontrolador, este elemento permite quemar el código con la ayuda de una computadora [11]. Una de las ventajas que se dispone al programar en lenguaje de bajo nivel es el control que se tiene al momento de programar un microcontrolador.

IDE ARDUINO es un entorno de programación que permite crear líneas de código en lenguaje C++, al finalizar el programa se puede grabar directamente desde la computadora hacia la placa simplemente conectando un cable entre los dos dispositivos [12].

Se sabe que existen otros entornos de programación como es el caso de *MicroPython* creado en 2013, el cual es una implementación del lenguaje de programación de *Python*, que en esta versión puede crear líneas de código para microcontroladores; sin embargo, son herramientas recientes que tienden a estar en constante modificaciones y por ende el desarrollo de proyectos electrónicos se complica ya que no cuentan con librerías, las velocidades de procesamiento no son muy rápidas por lo que no se puede desarrollar proyectos grande, las interfaces graficas no son muy intuitivas con el usuario, etc [13].

En base a las descripciones de los dos posibles entornos de desarrollo que se utilizará se optó por *IDE ARDUINO*, ya que el lenguaje de programación que maneja este *software* es de comprensión rápida, además de que no se debe instalar un programa o adquirir ningún otro elemento para grabar el programa en la tarjeta.

Determinación del *software* de simulación del sistema de control

Para determinar el programa de simulación, se determinó usar *Proteus Design Suite,* el cual es un programa que además de realizar diseños y la simulación de circuitos electrónicos, se puede incorporar a sus librerías de elementos electrónicos que no poseen por defecto en su *software.*

Proteus tiene la opción de diseñar y crear las placas de los circuitos impresos, y permite una visualización en 3D del circuito creado. Al ser un *software* muy usado por los usuarios se puede encontrar una gran variedad de información al momento de incorporar nuevas librerías de elementos ajenos a este, por lo que el manejo y la comprensión de estos elementos pueden ser más fáciles de usar.

Determinación del software para la creación de la aplicación móvil

Para determinar el *software* adecuado para la creación de la aplicación móvil, se tomó en cuenta, que las aplicaciones desarrolladas sean compatibles con distintos sistemas operativos, además de ello se evaluó otras características tales como: el acceso al entorno de desarrollo, y tipo de lenguaje de programación. Razón por la cual se determinó utilizar el entorno de desarrollo *App Inventor*; entre sus principales ventajas se destacan el uso libre de la plataforma, basta con disponer de una cuenta en Google, el lenguaje de programación en bloques es intuitivo y amigable con el usuario por lo que no se necesita mayor conocimiento para desarrollar una aplicación móvil.

3.3 Creación de la aplicación móvil

Para la creación de la aplicación móvil capaz de controlar el sistema de control a través de una conexión *bluetooth*, lo primero que se estableció es la forma de funcionamiento de dicha aplicación la cual se puede apreciar de forma general en el diagrama de flujo de la Figura 3-30.

Diseño de la aplicación móvil

El uso de esta aplicación se hará por medio de un dispositivo con sistema operativo *Android*, con la cual se podrá utilizar el modo personalizado para controlar el sistema de

luces y persianas por medio del celular. Como se mencionó con anterioridad el *software* que permitirá la creación de esta herramienta es *APP Inventor.*

Para lograr que el sistema de control se comunique con la aplicación móvil se usó un módulo *bluetooth* HC-06, el cual recibe información enviada por la aplicación a la placa *Arduino* UNO, para controlar el estado de las luces y el movimiento de las persianas.

Diseño de la interfaz de la aplicación móvil

En primera instancia se determinó la cantidad de ventanas que tendrá la aplicación, siendo la primera ventana, una pantalla de inicio como se puede ver a continuación en la Figura 3-33, esta tiene la función de iniciar el aplicativo.

Figura 3-33 Pantalla de inicio de la aplicación móvil

Como segunda ventana se tiene la presentación de las opciones que el usuario dispone; una de ellas es el modo de trabajo personalizado y la otra es la opción de ayuda. En la Figura 3-34, se puede ver la interfaz gráfica de la pantalla.

Figura 3-34 Pantalla de presentación de la aplicación móvil

En cuanto se cambia de la ventana de inicio a la de presentación, se despliega una ventana de notificación, la cual tiene como función recordar al usuario que para que el celular tenga conexión con la aplicación móvil se debe activar el *Bluetooth*, como se puede ver en la Figura 3-35. La pregunta que realiza el notificador es precisa; de ser la respuesta positiva, se podrá seguir a la siguiente pantalla, caso contrario se cierra la aplicación hasta que el usuario prenda el *Bluetooth*.

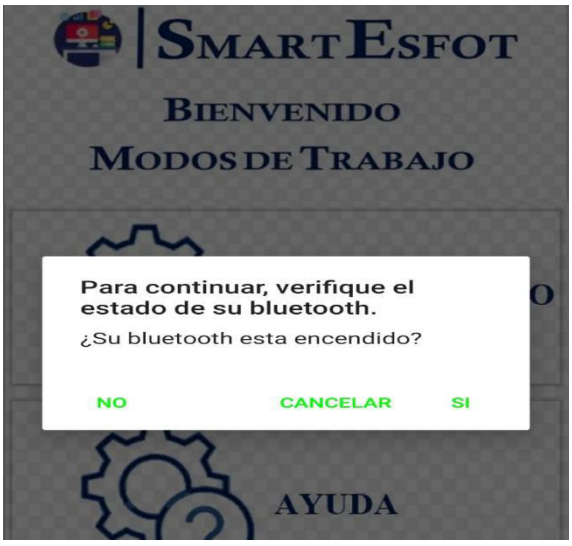


Figura 3-35 Pantalla del notificador *Bluetooth* de la aplicación móvil

Al seleccionar la opción de personalizado, y escoger la opción “SI” ante la pregunta que realiza el notificador mediante una pantalla emergente, se despliega una ventana la cual permitirá el manejo del sistema, dentro de esta opción se dispondrá el manejo tanto del sistema de luces ver en la Figura 3-36 como el de persianas.

Figura 3-36 Pantalla personalizada

Antes de seguir usando la aplicación, se desplegará una ventana, como se muestra en la Figura 3-37, la cual notifica y pregunta al usuario si ha habilitado el modo personalizado manualmente, si la respuesta es afirmativa se podrá seguir utilizando la aplicación, caso contrario se cierra.

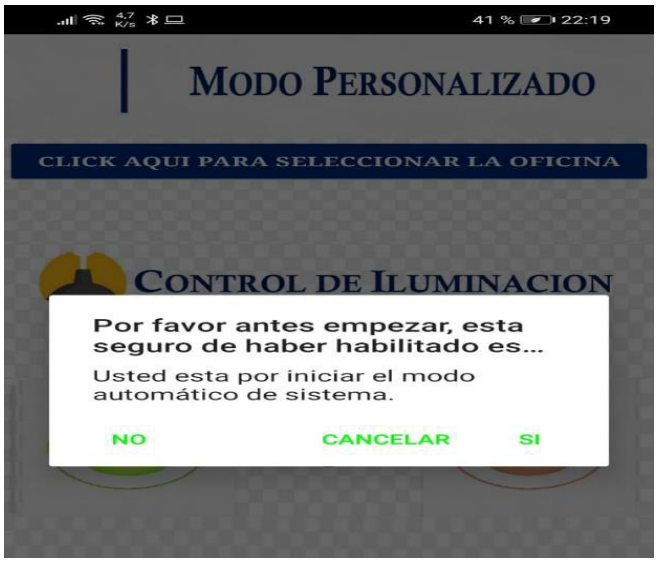


Figura 3-37 Pantalla del Notificador si está habilitado el modo personalizado

Adicional a la opción de personalizado, también tiene la alternativa de ayuda, donde al oprimir el botón se presenta una ventana como se puede visualizar en la Figura 3-38 la cual contiene información sobre el funcionamiento del sistema.

Figura 3-38 Pantalla de ayuda

3.4 Simulación de la aplicación móvil y el sistema de control de luces y persianas

Una vez diseñado el sistema de control de luces y persianas, se procederá a cargar el código fuente creado en el entorno de desarrollo IDE *Arduino*, en el sistema de control simulado en *Proteus*, con la finalidad de comprobar el funcionamiento de cada dispositivo electrónico.

El sistema que controlará la rotación del eje de las persianas tiene dos modos de operación, mientras que el sistema de control de luces cuenta con tres modos de funcionamiento, a continuación, se detalla el funcionamiento simulado del sistema de control.

Modo automático del sistema de control de luces y persianas

Para entrar en funcionamiento a este modo de trabajo, se debe presionar un pulsador denominado "botón automático", en ese momento *Arduino* UNO procesa la información de la fotoresistencia y en función de ello, se realizará las acciones determinadas.

En la

Figura 3-39, se puede observar que al estar separada la lámpara de la fotoresistencia la luminaria está encendida, mientras que al acercar la lámpara la luminaria se apaga.

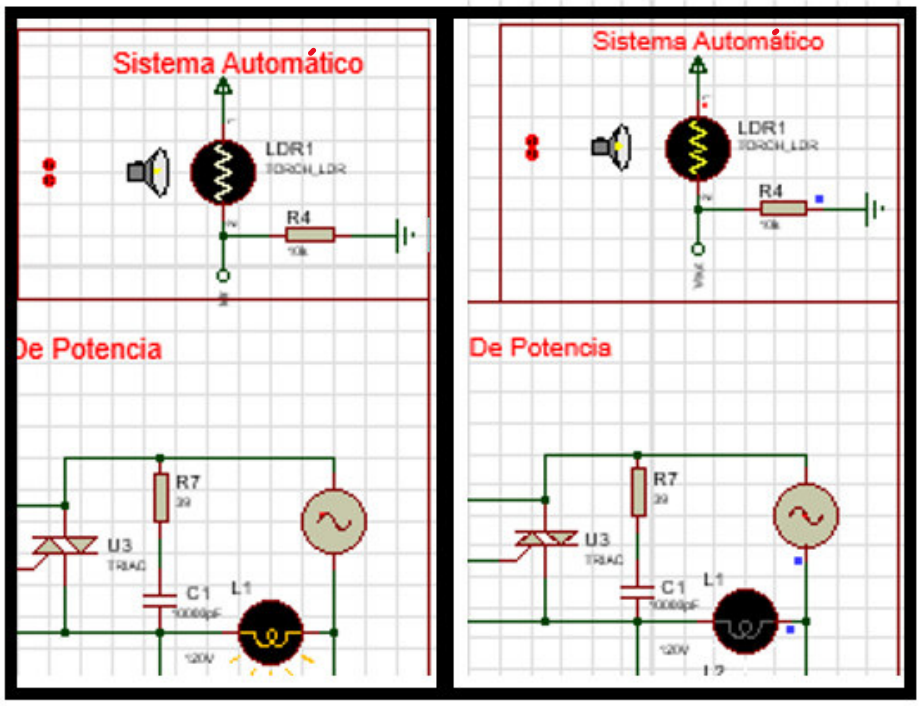


Figura 3-39 Modo de funcionamiento automático del circuito de luces

En el circuito de control de persianas, al estar separada la lámpara de la fotoresistencia, da a entender que existe deficiencia de luz por lo que el motor empieza a girar en sentido horario simulando cerrar las persianas, como se puede ver en la Figura 3-40.

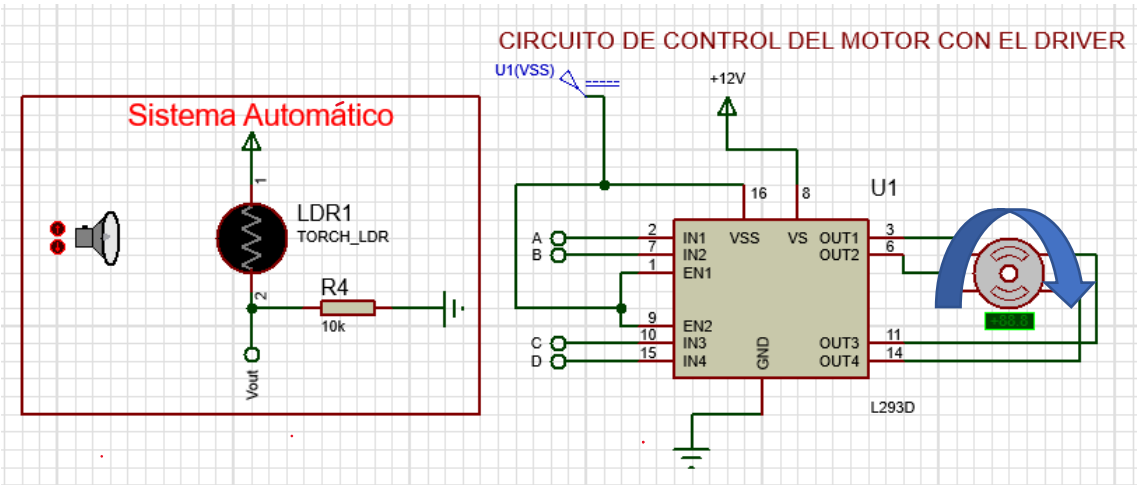


Figura 3-40 Simulación de giro del motor en sentido horario

Mientras que, al acercarse la lámpara a la fotoresistencia, el motor girará en sentido antihorario simulando abrir las persianas, en el Figura 3-41 se puede observar la acción descrita.

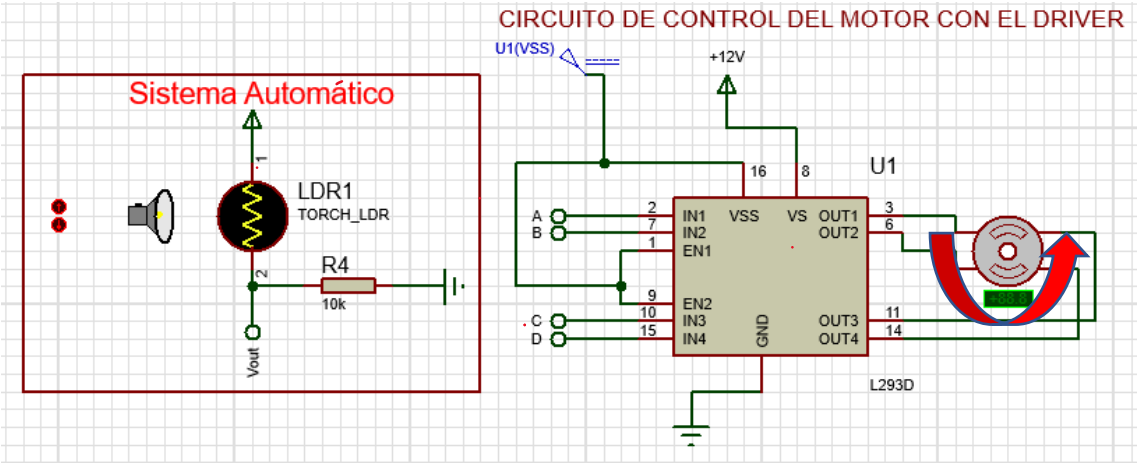


Figura 3-41 Simulación de giro del motor en sentido antihorario

Modo personalizado del sistema de control de luces y persianas

Como paso previo para el manejo en modo personalizado del sistema tanto de las persianas como el de las luminarias, se debe establecer la conexión *Bluetooth* mediante un puerto virtual entre la computadora, el celular, y *Proteus*. Este proceso se detalla a continuación.

Para controlar de manera inalámbrica el sistema de control a través de la aplicación, se debe crear un puerto virtual para que el *Bluetooth* de la computadora simule ser el módulo *Bluetooth* HC-06; como primer paso se debe agregar un nuevo dispositivo *Bluetooth* en la computadora.

En la Figura 3-42 se muestra cómo emparejar el dispositivo móvil que tenga la aplicación instalada y el equipo.

Figura 3-42 Vinculación de un dispositivo *Bluetooth*

En la Figura 3-43 se observa las configuraciones del *Bluetooth*, se debe seleccionar más opciones de *Bluetooth*.

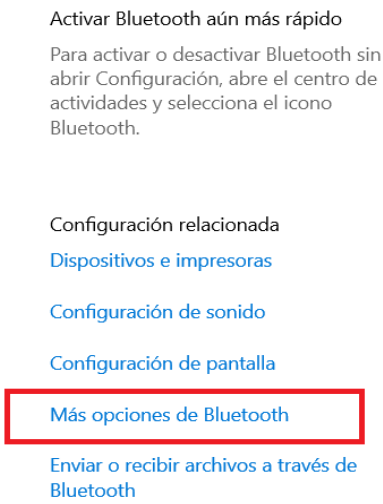


Figura 3-43 Opciones de *Bluetooth*

Se despliega una pantalla, como se puede ver en la Figura 3-44, se hace clic en agregar, y se agrega un nuevo puerto virtual.

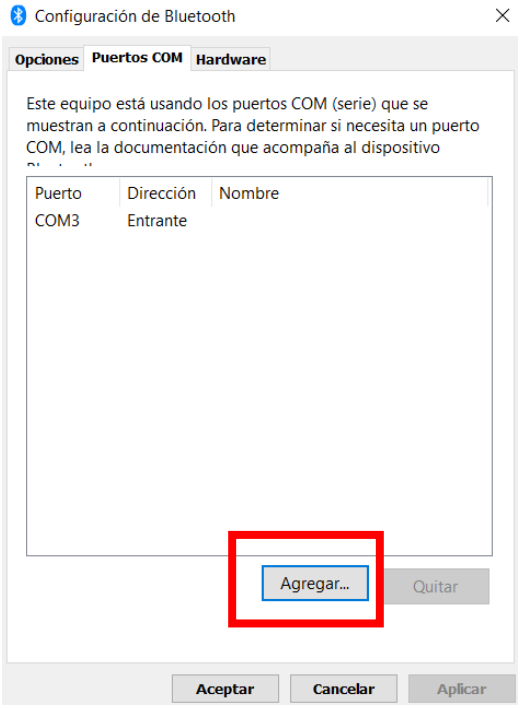


Figura 3-44 Crear un puerto virtual

Una vez creado el puerto virtual se debe ir a *Proteus*, se abre las configuraciones del módulo *Bluetooth* HC-05 y se asigna el puerto creado, así como se muestra en la Figura 3-45.

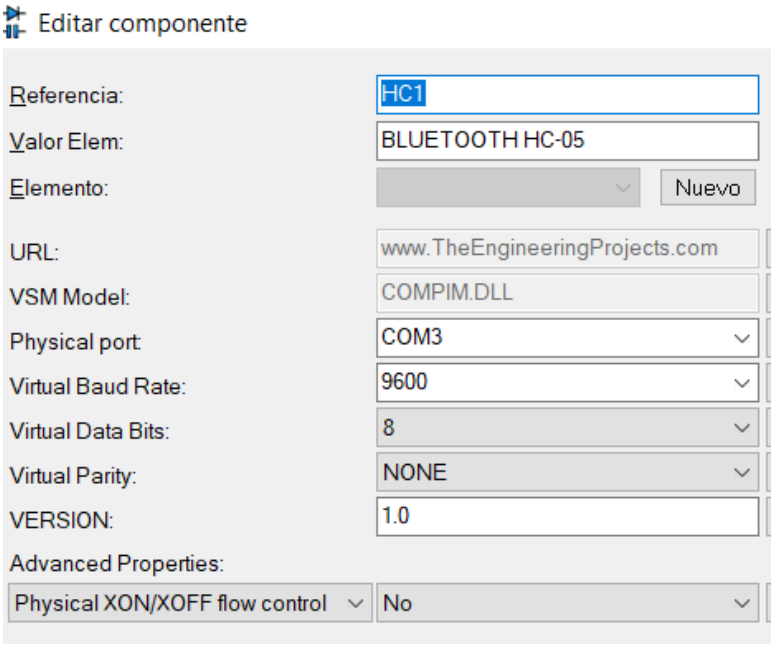


Figura 3-45 Configuración del módulo *Bluetooth*

Al finalizar la creación del puerto virtual y emparejar los dispositivos *Bluetooth*, se puede hacer uso de la aplicación móvil. En primera instancia, se debe conectar el celular con el módulo *Bluetooth;* se abre la aplicación y seleccionar la oficina en donde el usuario se encuentre, para seleccionar el *Bluetooth* de la oficina se debe hacer clic en "CLICK AQUÍ PARA SELECCIONAR LA OFICINA", como se muestra en la Figura 3-46.

Figura 3-46 Conexión al módulo *Bluetooth*

Al hacer clic en seleccionar la oficina, se desplegará una lista con los dispositivos *Bluetooth* disponibles, como se muestra en la Figura 3-47.

Figura 3-47 Lista de dispositivos *Bluetooth* disponibles

Una vez conectada la aplicación móvil con el sistema de control, se verifica si la conexión fue o no exitosa, mediante un mensaje que se despliega en la pantalla.

En la Figura 3-48 se puede verificar el estado de conexión.

Figura 3-48 Estado de conexión, entre la ampliación móvil y el módulo *Bluetooth*

Para que el sistema de iluminación trabaje en modo personalizado, se configuro el estado de los pulsadores en bajo en el código de programación cómo se puede observar en la Figura 3-49.

```
//configuración de los pulsadores para el modo personalizado
//donde valor1 hace referencia al botón manual y valor2 al botón automático
if((valor2==0) && (valor1==0))
```

Figura 3-49 Programación del estado de los pulsadores en modo personalizado

Una vez configurado el estado de los pulsadores en el código de programación, se configura en la simulación los pulsadores en estado bajo para que esta modalidad, se active como se muestra en la Figura 3-17.

La aplicación móvil dispone de dos botones al presionar el botón de encendido se prende las luminarias, como se puede apreciar en la Figura 3-50.

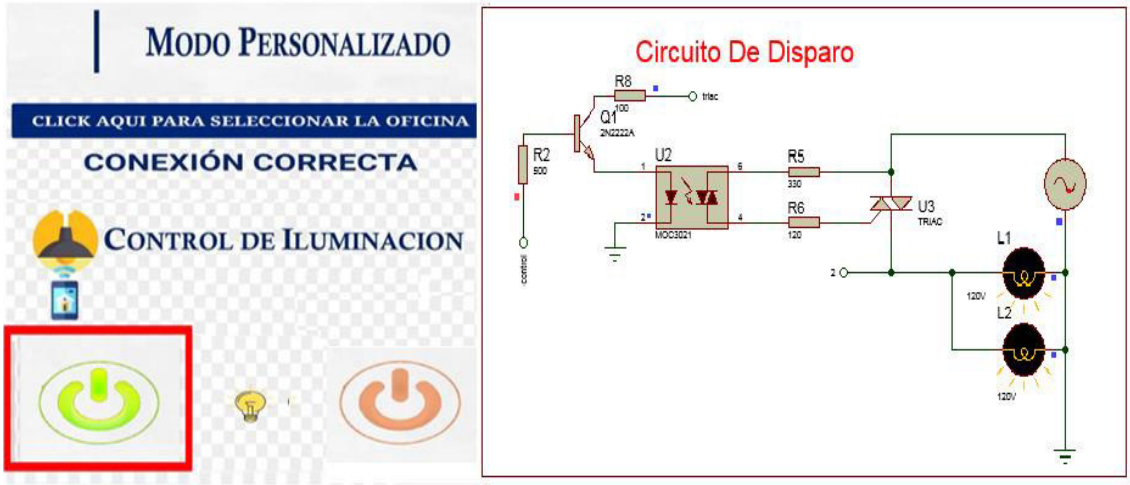


Figura 3-50 Modo personalizado luminarias encendidas

Al momento de oprimir el botón de apagado, las luces se apagan, como se puede ver en la Figura 3-51 para regular la intensidad de las luces se dispone de una *slider*, se puede ver los botones anteriores mencionados.

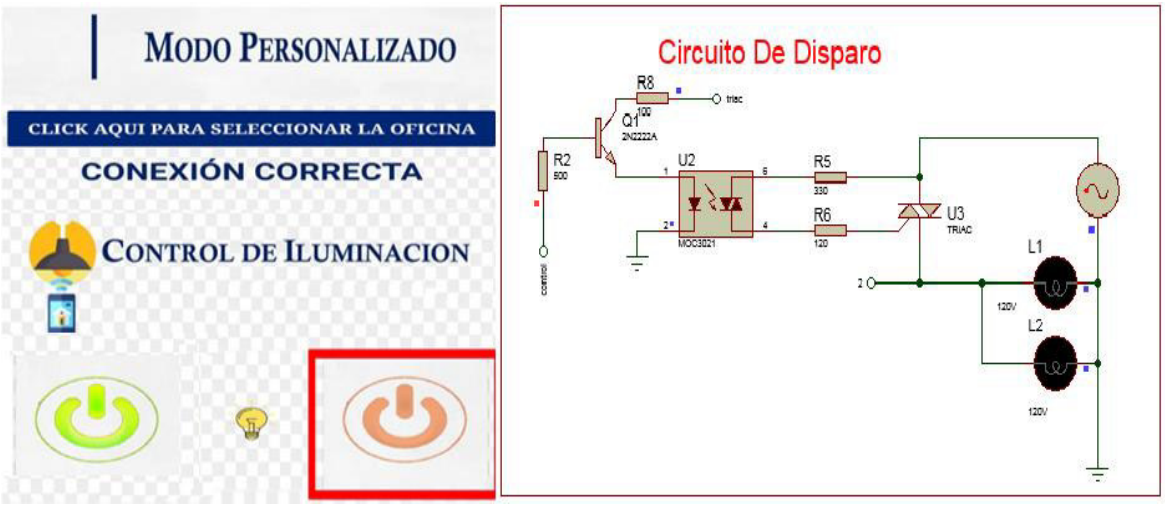


Figura 3-51 Modo personalizado luminarias apagadas

Para la activación del modo personalizado del circuito de persianas, el pulsador denominado “botón personalizado” debe estar oprimido, en ese momento *Arduino* Uno empieza a leer los datos que se envía mediante la aplicación móvil, como se puede ver en la Figura 3-52, la interfaz gráfica dispone de dos botones tanto para la apertura o cierre de las persianas, si se oprime el botón de la derecha en la aplicación, la persianas se empezará a cerrar, o si se escoge el botón de la izquierda la persiana se abrirá.

Figura 3-52 Modo personalizado sistema de control de persianas

Modo manual del circuito de control de luces

Para el modo Manual, el sistema cuenta con un pulsador llamado “botón manual” que al momento de ser presionado activará la función de variar la intensidad del sistema lumínico en función del cambio de posición del potenciómetro. Este cambio de posición

detectará *Arduino* UNO y enviará pulsos de control en intervalos de tiempo con la finalidad que el circuito de disparo pueda entregar una cierta cantidad de energía a las luces.

En la Figura 3-53 y Figura 3-54 se puede apreciar que al variar el potenciómetro la señal de la onda cada vez se va recortando o aumentando, entregando más o menos energía a las luces dependiendo de la posición del potenciómetro.

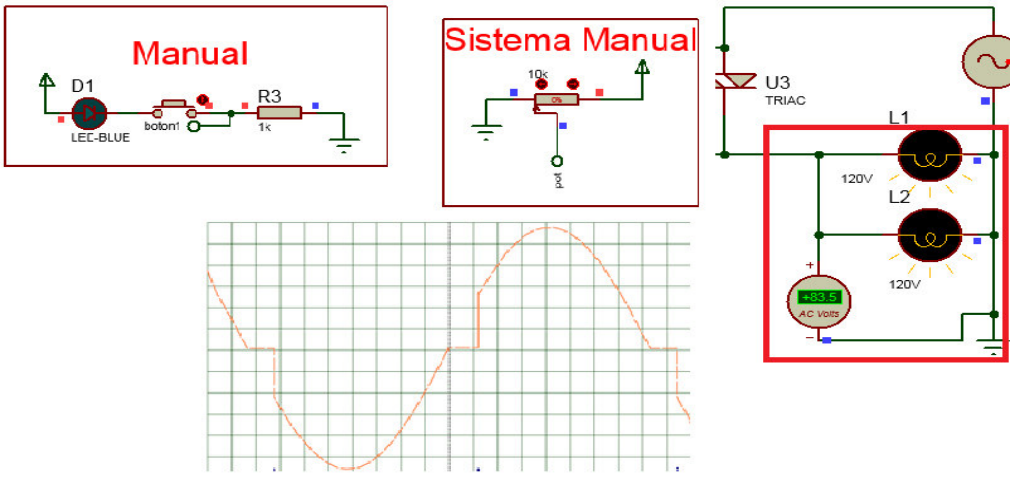


Figura 3-53 Posición 1 del potenciómetro

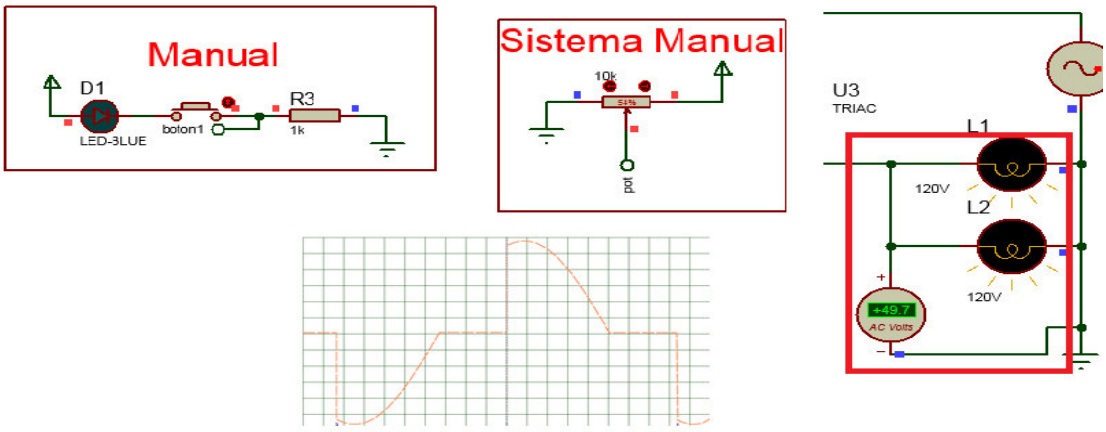


Figura 3-54 Posición 2 del potenciómetro

3.5 Manual de Uso y Mantenimiento

En la Figura 3-55 se muestra el código QR que direcciona al enlace donde se puede descargar la aplicación móvil, adicional a ello a continuación se coloca el enlace.

Enlace para descargar la aplicación

https://epnecuador-my.sharepoint.com/personal/leandro_pazmino_epn_edu_ec/_layouts/15/onedrive.aspx?ct=1643259498553&or=OWA%2DNT&cid=efa67806%2Dbc7e%2D926a%2Dfbc3%2D58933a96ca74&id=%2Fpersonal%2Fleandro%5Fpazmino%5Fepn%5Fedu%5Fec%2FDocuments%2FThesis%2FRomero%20%2D%20Hernandez%2FSMART%2DESFOT%2Eapk&parent=%2Fpersonal%2Fleandro%5Fpazmino%5Fepn%5Fedu%5Fec%2FDocuments%2FThesis%2FRomero%20%2D%20Hernandez

Figura 3-55 Código QR, Aplicación móvil

En la Figura 3-56 se muestra el código QR que direcciona al video del manual de uso del sistema de control de luces y persianas.

Figura 3-56 Código QR, video de uso y mantenimiento sistema de control

Enlace del video del manual de uso

https://epnecuador-my.sharepoint.com/personal/leandro_pazmino_epn_edu_ec/_layouts/15/onedrive.aspx?ct=1643651237917&or=OWA%2DNT&cid=36a53faa%2Dd512%2D60d5%2Db25a%2D1c592cf73066&id=%2Fpersonal%2Fleandro%5Fpazmino%5Fepn%5Fedu%5Fec%2FDocuments%2FThesis%2FRomero%20%2D%20Hernandez%2FMANUAL%20DE%20USO%20Y%20FUNCIONAMIENTO%20SISTEMA%20DE%20CONTROL%20DE%20LUCES%20Y%20PERSIANAS%2Emp4&parent=%2Fpersonal%2Fleandro%5Fpazmino%5Fepn%5Fedu%5Fec%2FDocuments%2FThesis%2FRomero%20%2D%20Hernandez

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se diseñó un sistema de control de luces y persianas con la finalidad de modernizar las áreas que abarca el proyecto, contribuir con el ahorro energético, optimizar el tiempo de trabajo de los usuarios ya que este sistema trabaja de manera automática ya sea encendiendo las luces o moviendo las persianas, evitando que el usuario se desconcentre en el desarrollo de las actividades.
- El sistema de control de persianas basa su funcionamiento en no más de cinco dispositivos electrónicos entre los cuales se tiene un motor a pasos *Nema*-23,

un *Driver* L293D, un módulo *Arduino* Uno, un módulo *Bluetooth HC-05,* y por último un interruptor, siendo de tal manera un sistema económico.

- En un determinado evento si el usuario desea abrir las persianas y al mismo tiempo apagar las luminarias, estas dos acciones no se pueden ejecutar ya que *Arduino* UNO permite la ejecución de una acción, razón por la cual se decidió dividir el sistema en dos subcircuitos, cada uno con un diferente microcontrolador con el objetivo de tener autonomía y que el usuario controle el sistema sin limitaciones.
- Se dividió el sistema de control en dos subcircuitos tomando en cuenta un posible corto circuito, fallo o deterior de los dispositivos electrónicos, de ser el caso se vería afectado un solo subcircuito, por consiguiente, la inspección y el arreglo sería en un periodo de tiempo corto.
- Para que el usuario pueda elegir el modo de trabajo del sistema de control, se utilizó pulsadores los cuales permiten escoger entre los tres modos de funcionamientos.
- En el desarrollo del código fuente se pudo analizar que para calibrar la duración de giro del motor y que cumpla su función, se debe programar la cantidad de pasos que debe dar el motor para cerrar o abrir las persianas, ya que, si se programa con base al tiempo de duración, el código tiende a caer en un bucle infinito. En otras palabras, si para abrir o cerrar las persianas el motor necesita de 1000 pasos o dos minutos, lo que se programa son los pasos que debe dar el motor y no el tiempo que se demora.
- Los dos subcircuitos se controlan de manera independiente; sin embargo, en el modo de funcionamiento personalizado estarán conectados mediante la misma fotoresistencia, la cual capta las variaciones de luz y envía la información a los dos microcontroladores y cada uno genera una acción distinta. En el modo personalizado pasa algo similar; sin embargo, en esta parte el dispositivo que une los microcontroladores es el módulo *Bluetooth*.
- El circuito cruce por cero es el elemento que permite regular la intensidad luminosa de las luces debido a la función que cumple en el sistema, es capaz de detectar el cambio de polaridad de la corriente alterna y ejecutar en el momento exacto cuando el circuito de disparo se active teniendo una sincronización entre la señal alterna y *Arduino*.
- Los dispositivos optoacopladores fueron elementos de gran importancia para cumplir con los requerimientos del sistema, además de aislar ópticamente los

circuitos que trabajan con corriente alterna del circuito de control que maneja bajos voltajes.

- Tener un sistema automatizado de luces y persianas permite al usuario ahorrar en cierta medida el consumo de energía eléctrica debido a que aprovecha al máximo la luz natural que incide en el entorno manteniendo apagadas las luces en función de la luz que incide en el entorno.
- Desarrollar la aplicación móvil *Android* en App Inventor resulta una tarea sencilla debido a que este *software* utiliza un lenguaje de programación en bloques que resulta intuitivo para programadores con poca experiencia obteniendo aplicaciones funcionales.
- Al momento de realizar la primera prueba de funcionamiento, se había realizado un solo código para el sistema de control; sin embargo, no se tomó en cuenta que el módulo *Arduino* UNO, solo realiza una actividad a la vez, de tal manera que, si se procedía a abrir las persianas, y a su vez tratar de encender o apagar las luminarias, en el *software* de simulación el sistema no procesaba las dos instrucciones y tendía a realizar un solo proceso a la vez, por tal razón se optó por dividir el sistema en subcircuitos.
- Al realizar pruebas de funcionamiento con el potenciómetro para regular la intensidad de las luces, se colocó los valores teóricos calculados en el código de *Arduino* para detectar cada vez que la señal cambie de polaridad. Al realizar estas pruebas se presentaron inconvenientes al momento de variar el potenciómetro dando como resultado la intermitencia de las luces debido al desfase de tiempo que hubo entre el periodo de la señal alterna y el tiempo de lectura de *Arduino* UNO por lo que se probó con varios valores de tiempo hasta que el problema de la intermitencia de las luces se eliminara.

4.2 Recomendaciones

- Para la simulación se recomienda utilizar un ordenador que cuente con módulo *Bluetooth* integrado o sino con un módulo externo para que al momento de simular el circuito de control permita establecer la conexión entre el programa de simulación y la aplicación móvil.
- Cuando se implemente el sistema de control de persianas, se debe tomar en cuenta el mantenimiento de este o un daño, razón por la cual por ningún motivo se deberá cortar las poleas propias de las persianas, ya que, en cualquiera de los dos casos mencionados anteriormente, este sistema manual de apertura y cierre seguirá funcionando mientras se soluciona los inconvenientes.

- Para la simulación del sistema de control es de gran importancia revisar el manual de usuario donde se explica detalladamente el funcionamiento de este sistema con la finalidad de que el usuario no presente problemas al momento de ejecutarlo.
- Para trabajos futuros se recomienda utilizar nuevos dispositivos de control que permitan obtener mejores resultados de respuesta cada vez que un módulo de funcionamiento esté en ejecución.
- Para el sistema de luces se recomienda usar luces *LED* dimerizables, ya que si se usa focos *LED* normales presenta inconvenientes como por ejemplo que el foco empieza a parpadear al momento de regular la intensidad luminosa.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] I. N. d. S. e. H. e. e. T. (INSHT), «ILUMINACION EN EL PUESTO DE TRABAJO,» MADRID, 2015.

[2] E. Rodriguez, «XATACA,» 22 Septiembre 2021. [En línea]. Available: https://www.xataka.com/makers/empezar-arduino-que-placa-kits-iniciacion-comprar. [Último acceso: 2021 Octubre 2021].

[3] e-elektronic, «E-ELEKTRONIC,» 28 Agosto 2013. [En línea]. Available: https://e-elektronic.com/tarjeta-arduino-vs-raspberri-pi-vs-beaglebone/. [Último acceso: 24 Octubre 2021].

[4] «photoconductive cell,» [En línea]. Available: https://www.sparkfun.com/datasheets/Sensors/Imaging/SEN-09088-datasheet.pdf. [Último acceso: 2021 octubre 2021].

[5] MecatrónicaLATAM, «MecatrónicaLATAM,» 24 Abril 2021. [En línea]. Available: https://www.mecatronicalatam.com/es/tutoriales/motor/motores-electricos/motor-de-corriente-continua/motor-paso-a-paso/. [Último acceso: 2021 Octubre 2021].

[6] J. R. P. Plazas, «Cómo programar en lenguaje C los microcontroladores PIC16F88, 16F628A y 16F877A,» de *L293D / L293B: driver para motores CC (DC)*, QUITO, ECUADOR, 2010, p. 39.

[7] FMS, «BRIDGE RECTIFIERS,» [En línea]. Available: http://pdf.datasheetcatalog.net/datasheet/formosa/2W08.pdf. [Último acceso: 24 octubre 2021].

[8] Sharp, «PC817 series,» [En línea]. Available: https://pdf1.alldatasheet.es/datasheet-pdf/view/43371/SHARP/PC817.html. [Último acceso: 24 octubre 2021].

[9] «TRIAC BT136,» [En línea]. Available: http://www.datasheet.es/PDF/121569/BT136-pdf.html. [Último acceso: 24 octubre 2021].

[10] uelectronics, «MOC3021,» [En línea]. Available: https://uelectronics.com/producto/moc3021-optoacoplador/. [Último acceso: 24 octubre 2021].

[11] M. Corona, 23 Agosto 2010. [En línea]. Available: https://es.slideshare.net/wapohot/introduccion-mplab. [Último acceso: 19 Octubre 2021].

[12] aprendiendoarduino, «aprendiendoarduino,» 11 Diciembre 2016. [En línea]. Available: https://aprendiendoarduino.wordpress.com/2016/12/11/ide-arduino/. [Último acceso: 19 Octubre 2021].

[13] MicroPython, «MicroPython,» 24 Enero 2019. [En línea]. Available: https://micropython.org/. [Último acceso: 19 Octubre 2021].

[14] «khan academy,» 26 Octubre 2018. [En línea]. Available: https://es.khanacademy.org/science/physics/circuits-topic/circuits-resistance/a/ee-kirchhoffs-laws#:~:text=Ley%20de%20voltaje%20de%20Kirchhoff,malla%20es%20igual%

20a%20cero.&text=Si%20caminas%20alrededor%20de%20la,de%20Kirchhoff%20conserva%20su%20validez..

[15] ELECTRONICALAB, «https://electronilab.co/,» 12 09 2017. [En línea]. Available: https://electronilab.co/tienda/motor-paso-a-paso-nema-23-125-oz-in-200-pasos-vuelta/. [Último acceso: 1 Septiembre 2021].

[16] *Oscilaciones,* Cordova, 2008.

[17] F. ILUMINACION, «Iluminación artificial e iluminación natural en los puestos de trabajo,» Santiago, Chile , 2019.

[18] Asociaciòn Española de Domòtica - CEDOM , «Dómotica,» Barcelona, 2013.

ANEXOS

ANEXO 1: CERTIFICADO DE FUNCIONAMIENTO

ESCUELA POLITECNICA NACIONAL
Campus Politécnico "J. Rubén Orellana R

Quito, 13 de diciembre de 2021

CERTIFICADO DE FUNCIONAMIENTO DE PROYECTO DE TITULACIÓN

Yo, *Leandro Pazmiño del director,* docente a tiempo completo de la Escuela Politécnica Nacional y como director de este trabajo de titulación, certifico que he constatado el correcto funcionamiento de la simulación de un sistema de control de luces y persianas para Dirección, Subdirección y Sala de Reuniones de la ESFOT, los cuales fueron implementados por los estudiantes Kevin Romero y Stalin Hernández

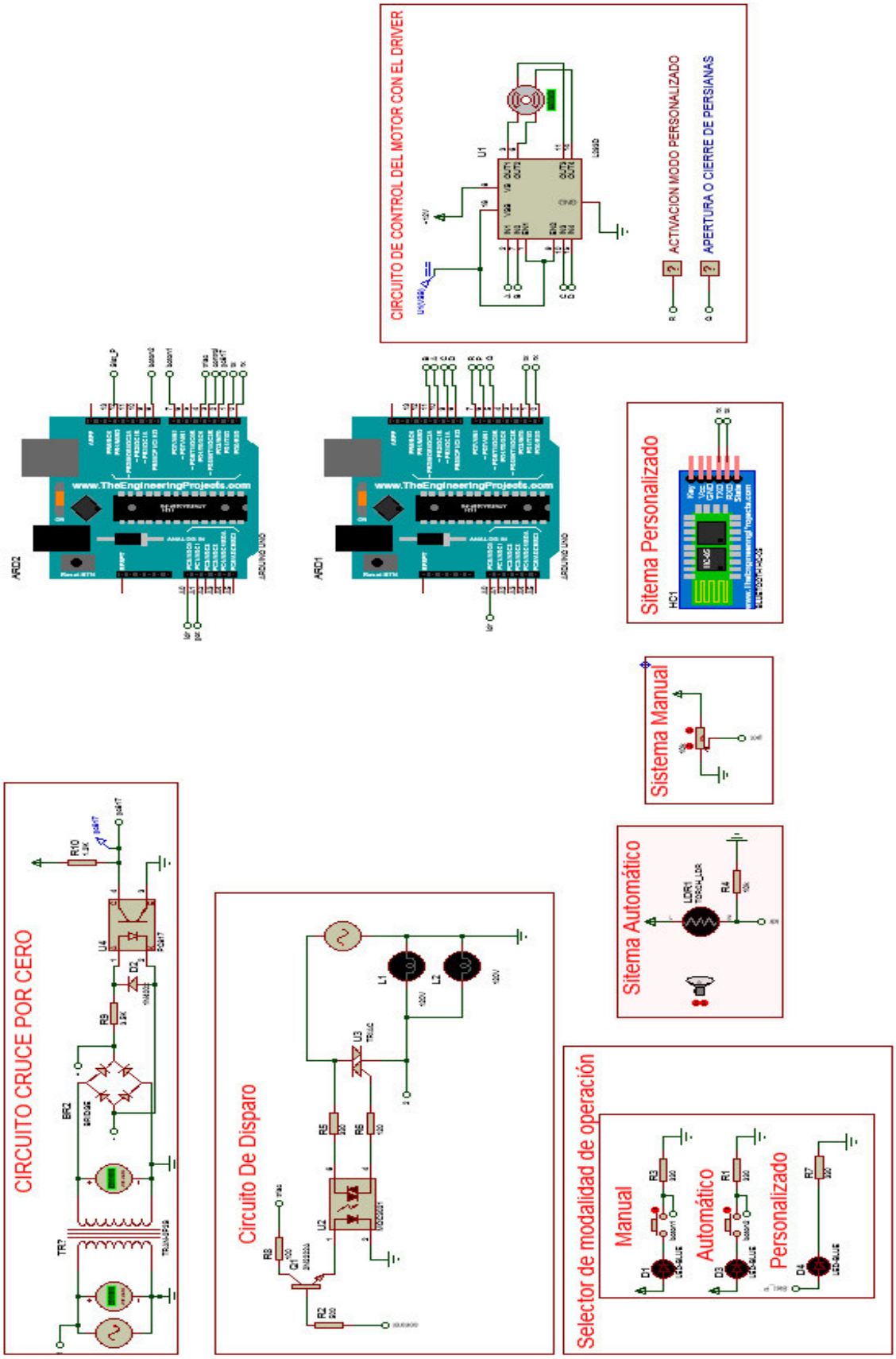
El proyecto cumple con los requerimientos de diseño y parámetros.

DIRECTOR
Ing. Leandro Pazmiño., MSc.

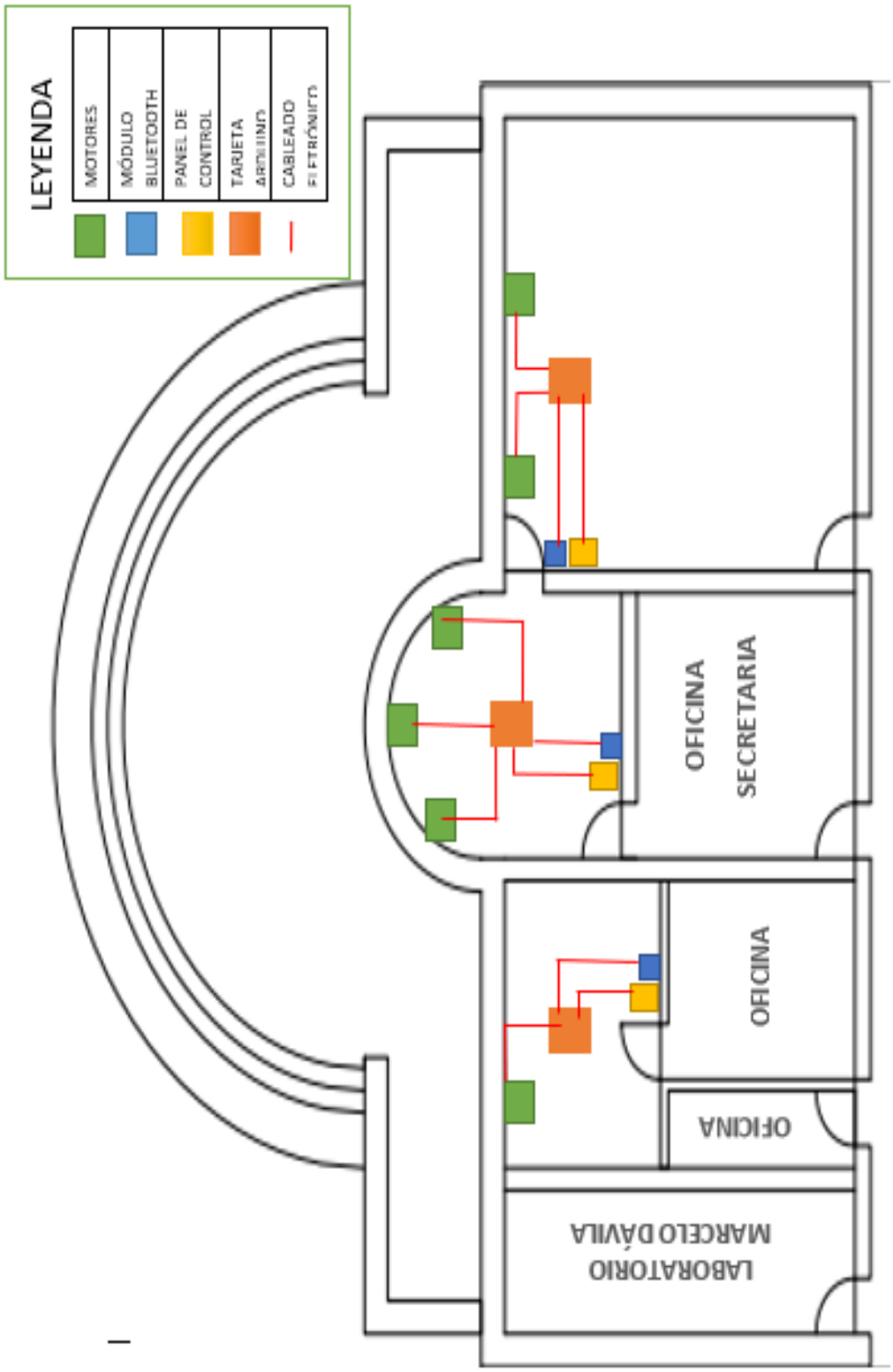
Ladrón de Guevara E11-253, Escuela de Formación de Tecnólogos, Telf: (593) 02 2976300 EXT: 2743
email: Leandro.pazmino@epn.edu.ec Quito-Ecuador

ANEXO 2: PLANOS Y ESQUEMAS

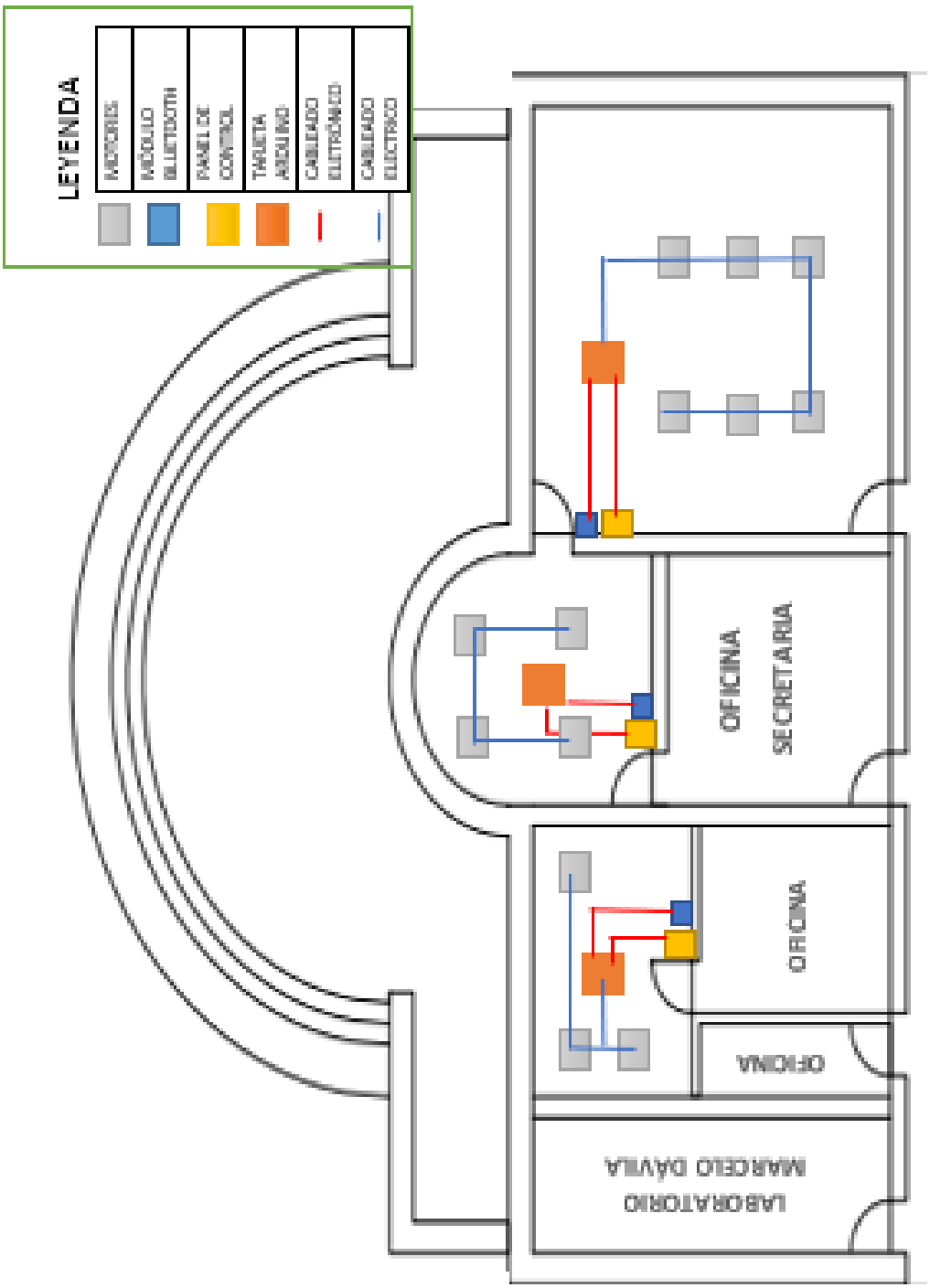
Esquema sistema control de luces y persianas



Plano del cableado electrónico sistema de control de persianas



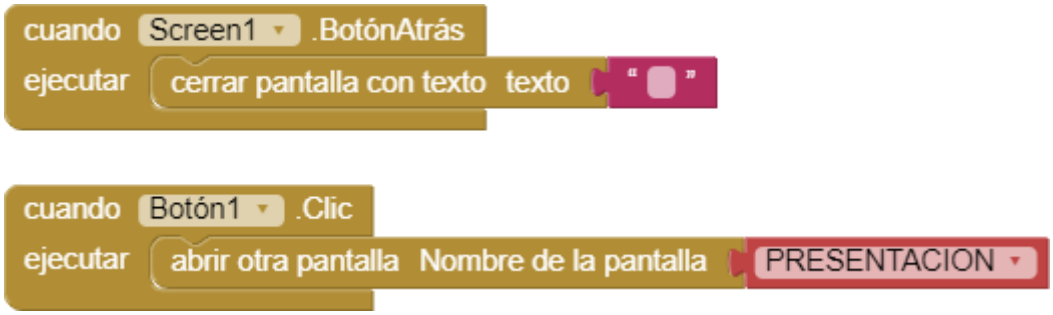
Plano del cableado electrónico sistema de control de luces



Aplicación móvil

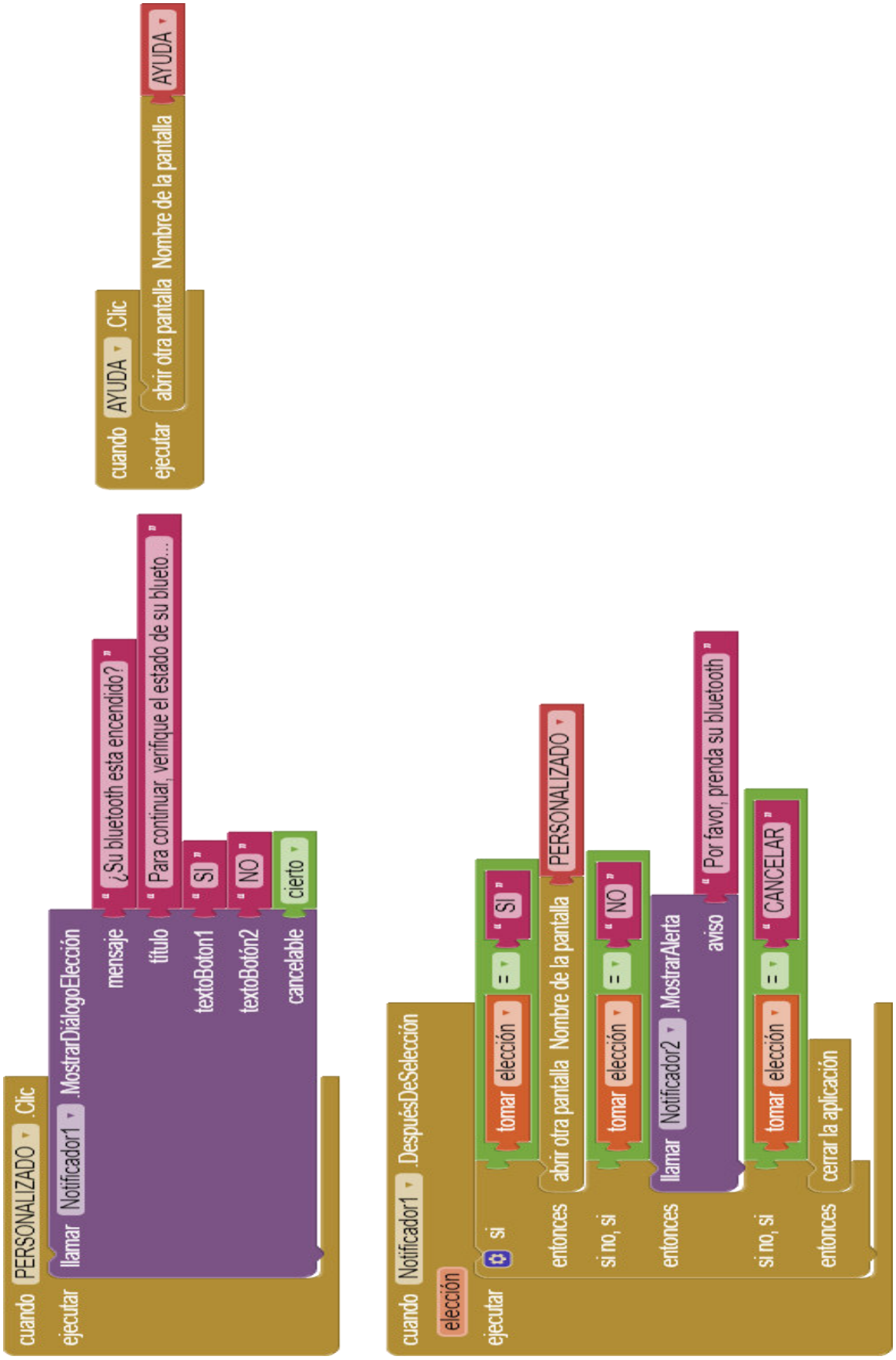
Interfaz gráfica

Código fuente

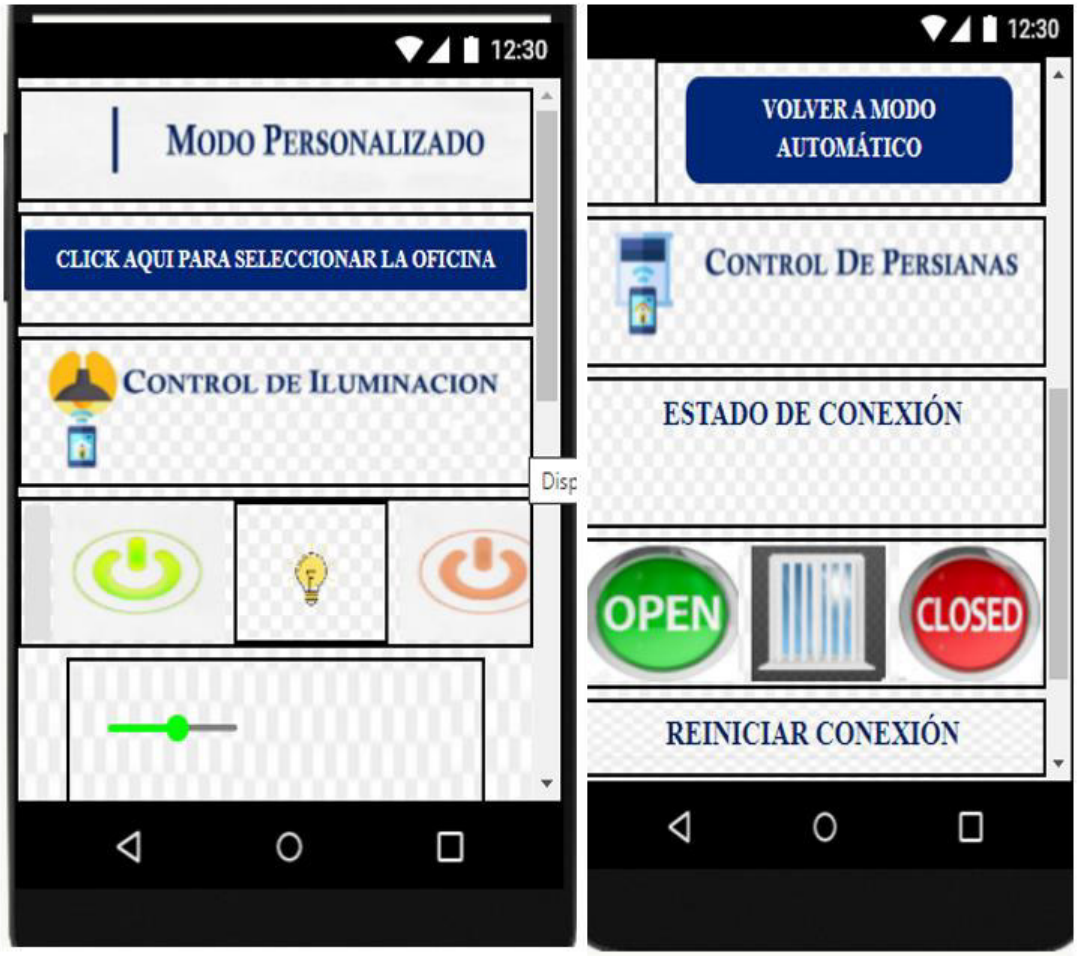


Interfaz gráfica

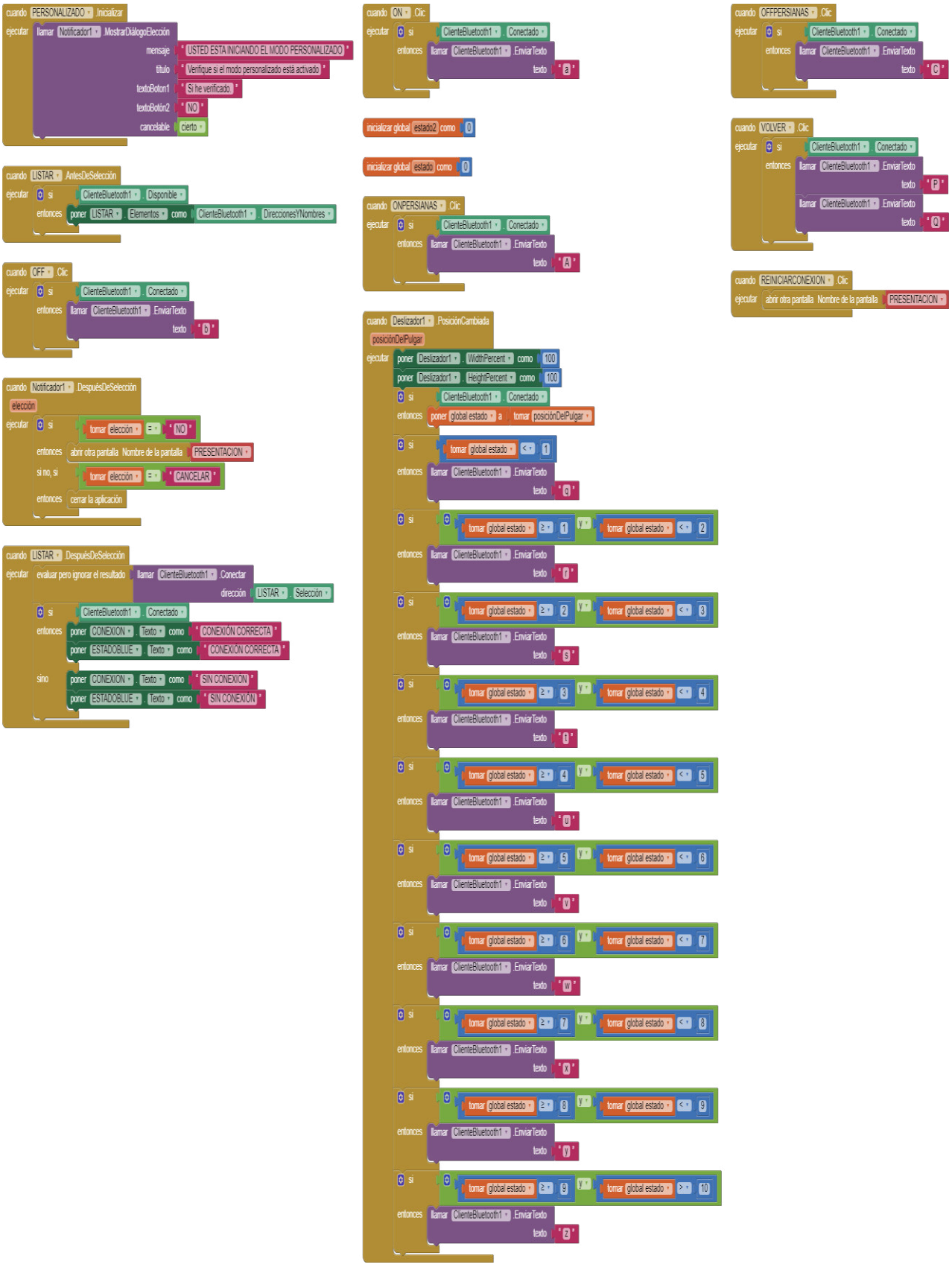
Código fuente



Interfaz gráfica



Código fuente



Interfaz gráfica

Código fuente

Código fuente sistema de control de persianas

```
#include <Stepper.h> //Importamos la librería para controlar motores paso a paso

#define STEPS 200 //Ponemos el número de pasos que necesita para dar una vuelta. 200 en nuestro caso

                                // Ponemos nombre al motor, el número de pasos y los pins de control

Stepper stepper(STEPS, 8, 9, 10, 11); //Stepper nombre motor (número de pasos por vuelta, pins de control)


const int PERSONALIZADO1=7;  //Declaracion del pin que activara o desactivara el modo personalizado

//const int CERRADA1=5;   //Declaracion del pin de fin de carrera

const int ONANDOFF1=6;    //Declaracion del pin que abrira a cerrara las persianas dependiendo de la descision del usuario


char cambio2;


int onandoff;      // constante que leera el valor pin 5

//int cerrada;      // constante que leera el valor pin 6

int personalizado;  // constante que leera el valor pin 7


const int LDRPin = A0;   //Pin del LDR

int LDRLECTURA;


void setup()

{

  pinMode(7, INPUT);
```

```
   pinMode(6, INPUT);
   pinMode(5, INPUT);
   digitalWrite(7, LOW);
   digitalWrite(5, LOW);
   Serial.begin(9600);
   stepper.setSpeed(20);
}
void loop()
{

 personalizado=digitalRead(PERSONALIZADO1);
 onandoff=digitalRead(ONANDOFF1);
// cerrada=digitalRead(CERRADA1);
INICIO:
     if (Serial.available()>0)
     cambio2 = Serial.read();
   if((cambio2=='A')&&(onandoff==HIGH)&&(personalizado==HIGH))                    //ON
PERSIANAS
  {
          stepper.step(200);
          goto APAGARMOTOR;
  }

  if((cambio2=='C')&&(onandoff==HIGH)&&(personalizado==HIGH))                    //OFF
PERSIANAS
  {
```

```
        stepper.step(-200);
        goto APAGARMOTOR;
 }

  if((personalizado==LOW)&&(onandoff==HIGH))          //CAMBIO A MODO
PERSONALIZADO
  {
       delay(1000);
       LDRLECTURA = analogRead(LDRPin);
       if (LDRLECTURA <600)  // CERRANDO LAS PERSIANAS
          {
          CERRAR:
          stepper.step(200);
          delay(500);
          goto BOTOM1;
          }
        else if (LDRLECTURA >=600)          //ABRIMOS LAS PERSIANAS
          {
          ABRIR:
          stepper.step(-200);
          delay(500);
          goto BOTOM;
          }
                    BOTOM:
                    {
                        if (Serial.available()>0)
```

```
        cambio2 = Serial.read();
        if((cambio2=='A')||(cambio2=='C'))
         {goto APAGARMOTOR;}
         PORTB= B00000000;
         delay(100);
         LDRLECTURA = analogRead(LDRPin);
        if (LDRLECTURA >=600)
         {goto BOTOM;}
         else
         {goto CERRAR;}
    }
    BOTOM1:
    {
        if (Serial.available()>0)
        cambio2 = Serial.read();
        if((cambio2=='A')||(cambio2=='C') )
        {goto APAGARMOTOR;}
        PORTB= B00000000;
        delay(100);
        LDRLECTURA = analogRead(LDRPin);
        if (LDRLECTURA <600)
        goto BOTOM1;
        else goto ABRIR;
    }
APAGARMOTOR:
```

```
        {
         cambio2 = 0;
        PORTB= B00000000;
        }
   }
 if    ((onandoff==LOW)&&(personalizado==LOW)||(personalizado==HIGH))        //OFF
PERSIANAS
  {
          stepper.step(0);
          PORTB= B00000000;
   }
}
```

Código fuente sistema de control luces

```
int ldr=A0; /// pin para leer el sensor ldr
int pot=A1; /// pin para leer el potenciometro
int cruce_cero = 2; // este es el pin para detectar el cruce por cero de la señal AC
int triac = 4; // pin para controlar el circuito de disparo
int valorpot; /// variable para guardar el valor del potenciometro
int valorldr; /// variable para guardar el valor del ldr
```

```
int tiempo_pot = 0; //variable para medir los tiempos de disparo para activar el triac en funcion del potenciometro

int manual=7; /// pulsador para activar el modo manual

int automatico=8; /// pulsador para activar el modo automatico

int valor1; // variable para el estado del pin 7

int valor2; // variable para el estado del pin 8

int dato_bt; // variable para guardar los valores enviados por la aplicacion

int tiempo_bt; // variable para guardar los tiempos de disparo para activar el triac en funcion del bluethoot

int control = 3; // pin para activar el timepo de disparo del triac

int personalizado = 12;



void setup() {

  //inicio de la comunicacion con el modulo bluetooth

 Serial.begin(9200);


 // pines asignados a los circuitos e potencia

 pinMode(cruce_cero, INPUT);

 pinMode(triac, OUTPUT);

 pinMode(control, OUTPUT);

 digitalWrite(control, LOW);

 digitalWrite(triac, HIGH);

 pinMode(personalizado, OUTPUT);

 digitalWrite(personalizado, LOW);
```

```
 // pines asignados a los pulsadores
 pinMode(manual,INPUT);
 pinMode(automatico,INPUT);

 //activacion de la interrupcion
 attachInterrupt(digitalPinToInterrupt(cruce_cero), Cambio,FALLING);
}

void loop() {

//asignacion de variables para guardar los datos de los sensores
valor1=digitalRead(manual);
valor2=digitalRead(automatico);
valorldr = analogRead(ldr);
valorpot = analogRead(pot);

//se asigna una varable donde se guardara las datos que provengan de la aplicacion movil
if((Serial.available()>0))
{dato_bt = Serial.read();}

/*configuracion de los pulsadores para activar el modo automatico la fotorresistencia
establecera el estado de encendido o apagado de las luces en funcion de la luz natural*/
if((valor2==1) && (valor1==0))
{
if(valorldr<=512){digitalWrite(control,HIGH); digitalWrite(triac,HIGH); }
```

```
if(valorldr>=513){digitalWrite(control,LOW); digitalWrite(triac,LOW);} }

//configuracion de los pulsadores para activar el modo personalizado
if((valor2==0) && (valor1==0)){
  digitalWrite(personalizado, HIGH);
if(dato_bt=='a')
{
tiempo_pot=0;
digitalWrite(control,HIGH);
digitalWrite(triac,HIGH);

}

if(dato_bt=='b')
{
tiempo_bt=0;
digitalWrite(control,LOW);
digitalWrite(triac,LOW);

dato_bt=0;
} }
else{digitalWrite(personalizado, LOW);}
// valores asignados correpondientes a la posocion del deslizador de la aplicacion movil
if(dato_bt=='z') tiempo_bt = 10;
if(dato_bt=='y') tiempo_bt = 787;
```

```
if(dato_bt=='x') tiempo_bt = 1564;

if(dato_bt=='w') tiempo_bt = 2341;

if(dato_bt=='v') tiempo_bt = 3118;

if(dato_bt=='u') tiempo_bt = 3895;

if(dato_bt=='t') tiempo_bt = 4672;

if(dato_bt=='s') tiempo_bt = 5449;

if(dato_bt=='r') tiempo_bt = 6226;

if(dato_bt=='q') tiempo_bt = 7000;

if((valor2==1) && (valor1==1))

{digitalWrite(control,LOW); digitalWrite(triac,LOW);}

}


//interrupcion

void Cambio(){

//sistema manual para la variacion de la intensidad luminosa controlado por el potenciometro

if((valor2==0) && (valor1==1))

{

digitalWrite(control,HIGH);

tiempo_pot= map(valorpot, 0,1024, 0,7000); // esta funcion map permite relacionar la posicion del potenciometro con los valores dados

digitalWrite(triac, LOW); //  estado bajo al momneto que el pin2 detecte un flanco de bajada

delayMicroseconds(tiempo_pot); //retardo de un tiempo t en funcion del valor tiempo_pot

digitalWrite(triac, HIGH); // estado alto para activar el circuito de disparo

delayMicroseconds(10); // retador para mantener activado el circuito de disparo
```

```
digitalWrite(triac, LOW); //estado bajo para apagar el circuito de disparo
}

//sistema perzonalizado para la variacion de la intensidad luminosa controlado por la aplicacion movil
if((valor2==0) && (valor1==0))
{
digitalWrite(triac, LOW); //  estado bajo al momneto que el pin2 detecte un flanco de bajada
delayMicroseconds(tiempo_bt); //retardo de un tiempo t en funcion del valor tiempo_bt
digitalWrite(triac, HIGH); // estado alto para activar el circuito de disparo
delayMicroseconds(10); // retador para mantener activado el circuito de disparo
digitalWrite(triac, LOW); //estado bajo para apagar el circuito de disparo

}}
```