



INSTITUTO TECNOLÓGICO DE TUXTLA GUTIERREZ
FACULTAD DE INGENIERÍA ELECTRÓNICA

T E S I S

“MONITOREO Y REGULACIÓN DE LUMINARIAS EN EDIFICIOS”

PARA OBTENER EL TÍTULO DE

INGENIERO EN ELECTRÓNICA.

ASESOR

DR. RUBEN HERRERA GALICIA.

PRESENTAN

CÉSAR HEBERTO LÓPEZ HERRERA.

MANUEL ALEJANDRO AGUILAR SOLÍS.

TUXTLA GUTIERREZ, CHIAPAS

SEP

SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Tuxtla Gutiérrez

DIRECCIÓN
SUBDIRECCIÓN ACADÉMICA
DIVISIÓN DE ESTUDIOS PROFESIONALES
Tuxtla Gutiérrez, Chiapas 18 de abril del 2016

OFICIO NUM. DEP-CT-60-2016

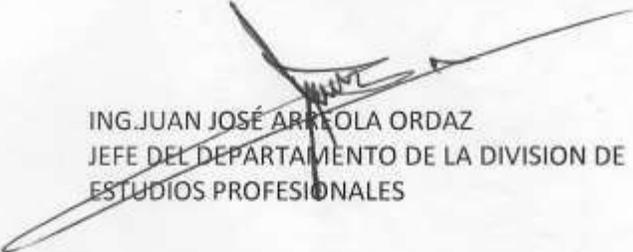
C. CESAR HEBERTO LÓPEZ HERRERA, MANUEL ALEJANDRO AGUILAR SOLIS
PASANTE DE LA CARRERA DE INGENIERÍA ELECTRÓNICA
EGRESADO DEL INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ.
P R E S E N T E.

Habiendo recibido la comunicación de su trabajo profesional por parte de los CC. DR. RUBEN HERRERA GALICIA, M.C. ARNULFO CABRERA GÓMEZ Y M.C. ANGEL SEIN PÉREZ RODRÍGUEZ en el sentido que se encuentra satisfactorio el contenido del mismo como prueba escrita, **AUTORIZO** a Usted a que se proceda a la impresión del mencionado Trabajo denominado:

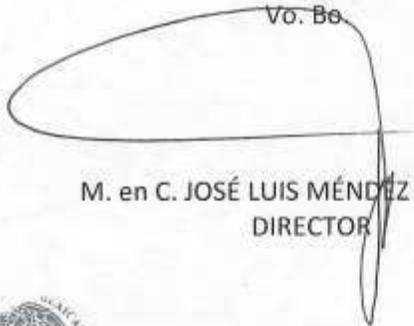
" MONITOREO Y REGULACIÓN DE LUMINARIAS EN EDIFICIOS"

Registrado mediante la opción:
I (TESÍS PROFESIONAL)

ATENTAMENTE
"CIENCIA Y TECNOLOGÍA CON SENTIDO HUMANO"


ING. JUAN JOSÉ ARRIOLA ORDAZ
JEFE DEL DEPARTAMENTO DE LA DIVISION DE
ESTUDIOS PROFESIONALES

C.c.p.- Departamento de Servicios Escolares
C.c.p.- Expediente
I'JLMN/I'JAO/I'eeam

Vo. Bo.

M. en C. JOSÉ LUIS MÉNDEZ NAVARRO
DIRECTOR



Secretaría de Educ. Pública
Instituto Tecnológico
de Tuxtla Gutiérrez,
Div. de Est. Profesionales



Carretera Panamericana Km. 1080, C.P. 29050, Apartado Postal 599
Tuxtla Gutiérrez, Chiapas, Tels. (961) 61 54285, 61 50461
www.ittg.edu.mx



Fecha de Impresión: 2016-04-18
Fecha de Terminación: 2016-04-18

Agradecimientos

Manuel Alejandro.

Le agradezco principalmente a mi Dios, por acompañarme siempre a lo largo de mi vida y de mi carrera profesional, por brindarme fortaleza en mis momentos de debilidad y por todas las experiencias, los aprendizajes y las alegrías en mi vida.

Les agradezco a mis padres Jorge Raúl y Lidia Eloina por su apoyo incondicional, por su incansable labor siempre en beneficio de sus hijos, por los valores inculcados en mi persona, por haberme permitido gozar de una excelente educación y sobre todo por el amor que en todo momento me han brindado. Lo que he logrado y lo que soy en mi vida es indudablemente gracias a ustedes.

A mis hermanos Carolina, Jorge y Liliana por ser parte fundamental en mi vida y representar la unidad familiar, por el apoyo que siempre me ayudó a conseguir mis objetivos y por el amor que siempre me llenó de fortaleza.

Alejandra Sujey, a ti por haberme apoyado en las buenas y en las malas, por tu paciencia, apoyo y amor incondicional, por ser una de mis motivaciones y por ser una persona fundamental en este logro de mi vida.

Gracias Dr. Rubén Herrera Galicia, por creer en César y en mí, por habernos brindado la oportunidad de desarrollar nuestra tesis profesional y por las facilidades a lo largo de este proceso.

A César Heberto por confiar en mí, por ser un gran compañero y amigo, y por la oportunidad de desarrollar esta tesis profesional juntos.

Sin ustedes nada de esto hubiera sido posible

Gracias.

ÍNDICE

1. Introducción	3
1.1 Antecedentes	3
1.2 Estado del arte.....	4
1.3 Justificación.....	4
1.4 Objetivo	5
1.5 Metodología, diagrama a bloques.....	5
2. Fundamento teórico.....	6
2.1 Sistema Operativo Android	6
2.2 App Inventor	7
2.3 Módem Bluetooth RN- 41N.	13
2.4 Arduino Ethernet shield.....	14
2.5 Sensor de movimiento PIR.	15
2.6 Placa Arduino Uno.	15
2.7 Lámpara Led.	17
3. Desarrollo	18
3.1 Aplicación SO Android	18
3.2 Modo de enlace WEB	23
3.2 Modo de enlace WEB	23
3.2 Etapa de potencia.....	23
3.2 Diagrama del sistema.....	23
4. Resultados y conclusiones.....	30
4.1 Resultados	30
4.2 Observaciones y sugerencias.....	30
4.3 Conclusiones	30
5. Anexos	31
6. Bibliografía.....	36

1. Introducción

El trabajo aquí descrito está enfocado a implementar un sistema capaz de regular la potencia entregada a lámparas de iluminación en interiores, con fines de uso eficiente y ahorro de energía. La calibración de la potencia entregada a las lámparas se hace desde una aplicación soportada en el sistema operativo Android con comunicación vía Bluetooth y también, desde una página web diseñada específicamente para el control del sistema.

El sistema reduce el consumo de energía vía interacción con la información que un sensor de presencia envía al sistema, la detección de presencia del sensor activa el control gradual que fue programado por el usuario, basándose en la iluminación necesaria para que el área tenga una claridad agradable. El principio básico de operación se rige por el valor de las variables de estado y de control. Cuando no se detecta presencia de usuario, el sistema disminuye el nivel de luz hasta un mínimo. Cuando se detecta presencia de usuario, el sistema aumenta el nivel de luz hasta un máximo. Los cambios son graduales para evitar incomodidad del usuario.

Protocolos para automatización de casas y edificios; BACnet, KONEXbus, LonWorks, DALI, onOCEAN, y M-Bus, son algunos de los sistemas de automatización en casas hogar y edificios públicos y privados. Estos sistemas hacen la vida más confortable y pueden cambiar drásticamente la calidad de vida. También son capaces de automatizar la administración de la energía, la seguridad, y la comunicación entre dispositivos haciendo posible el uso eficiente de la energía y contribuyendo al desarrollo sustentable de la sociedad.

La automatización de edificios se basa en el desarrollo de protocolos de comunicación entre dispositivos. El modelo OSI sigue siendo la referencia para establecer el lugar de los protocolos nacescentes. El medio de comunicación es variado. Algunos se caracterizan por usar comunicación inalámbrica, EnOcean, M-Bus. Sin embargo un importante desarrollo están viviendo los protocolos basados en comunicación Ethernet vía página Web. El esquema que se usa en él es el de envío de paquetes.

1.1 Antecedentes

Es común que las luminarias de un edificio público o privado permanezcan encendidas durante todo el lapso de tiempo de servicio, sin importar si es necesario iluminar ciertas áreas y pasillos en determinadas horas del día y peor aún, sin importar, si realmente hay personas en el área. Durante un día cualquiera, todo el día, el nivel de iluminación se mantiene constante. También para dos días diferentes de la semana el nivel de iluminación es el mismo. Significa que se carece de una diferenciación de las necesidades reales de iluminación durante las diferentes horas del día y entre un día y otro. Esta falta de diferenciación provoca un consumo excesivo innecesario de energía eléctrica.

En México más del 70% de la energía eléctrica se obtiene a partir de la quema de gas y petróleo, solo un 14% proviene de las centrales hidroeléctricas. Desde esta perspectiva desperdiciar energía eléctrica equivale a contribuir con el proceso del calentamiento global de

la atmosfera, dado que para haber producido esa energía eléctrica se debió haber quemado algún combustible, con su respectiva emisión de gases calientes a la atmosfera.

También desperdiciar energía eléctrica es una debilidad en la administración de una empresa por el gasto económico que representa.

1.2 Estado del arte

Jay Taneja de la División de Ciencias Computacionales de la Universidad de California en Berkeley California, muestra que la arquitectura basada en BOSS/BAS permite mejorar el proceso de desarrollo de aplicaciones para edificios abstrayendo del hardware, los sensores, y del código de programación.

Un aspecto importante de la Pila de Aplicaciones para Edificios BAS es el uso de secuencias lingüísticas, con una función de relación, que permiten expresar las intenciones, a alto nivel, tal que sean inherentemente portables: “apagar, las luces, cerca de las ventanas, en los cubículos del piso superior”.

Además permite explorar los componentes específicos de un edificio permitiendo a las aplicaciones de manera explícita manejar las diferencias entre edificios.

En BAS los desarrolladores de aplicaciones alternan entre vistas del edificio a nivel macro y a nivel micro: “luces en el piso superior” vs “relé de luces 1023”. Así BAS permite expresar intenciones generales o acciones específicas. Esto libera a los desarrolladores de considerar las particularidades del diseño de cada edificio permitiendo enfocar el desarrollo hacia la interacción entre espacios físicos y sus ocupantes en lugar de enfocarlo hacia los detalles del equipo.

Building Operating System Services BOSS es una arquitectura para controlar dispositivos físicos y sistemas de un edificio que origina desarrollo de una aplicación portable. Múltiples aplicaciones para edificios han sido desarrolladas antes del surgimiento de la arquitectura basada en BOSS. En donde se presenta una tabla que agrupa diversas aplicaciones encontradas en la literatura.

1.3 Justificación

El sistema que aquí se presenta tiene como finalidad permitir el ahorro de energía eléctrica, basando su operación en un control de iluminación, utilizando una aplicación móvil soportada en Android y una página web, que permite configurar un nivel máximo y mínimo de iluminación. Tiene como ventaja la facilidad de operación gracias a la interfaz gráfica. Esto permite obtener los parámetros deseados de iluminación y con esto lograr el efecto visual deseado.

El proyecto también, es importante desde el punto de vista económico porque con él se logra ahorro energético. También es importante desde el punto de vista ambiental, porque ahorro energético significa menor calentamiento global. Además que controlar la potencia que se entrega a lámparas de iluminación de un edificio con una red inteligente es un tema de actualidad que tiene un alto potencial de desarrollo tecnológico.

1.4 Objetivo

Diseñar e implementar un sistema que permita configurar el nivel de iluminación para unidades de iluminación, que permita incorporar dispositivos y sistemas eléctricos para que de manera amigable se puedan efectuar tareas de detección de presencia, monitoreo y control distribuido de energía, aplicado a iluminación de edificios.

1.5 Metodología, diagrama a bloques

La idea es un desarrollo con bloques básicos que sea expansible. El prototipo elemental está compuesto por los elementos observados en la Fig.1. En el diagrama se presentan únicamente dos sectores como representación del sistema, no excluyendo que puedan existir más sectores. El hardware está compuesto por 6 bloques donde cada uno cumple con una función importante.

La comunicación del sistema se realiza mediante Ethernet. Los dispositivos modem Bluetooth sirven para comunicarse directamente con las unidades de control. En la etapa actual el sistema cuenta con una unidad concentradora que centraliza el cableado del sistema.

El sistema cuenta con una unidad de control por cada sector de lámparas, en el diagrama mostrado son dos. La unidad de control consta de un dispositivo con una plataforma de hardware libre, basada en una placa con un micro controlador. A través del puerto RxTx del micro controlador se conecta un modem Bluetooth para la comunicación entre esta unidad y la unidad de monitoreo con dispositivo móvil.

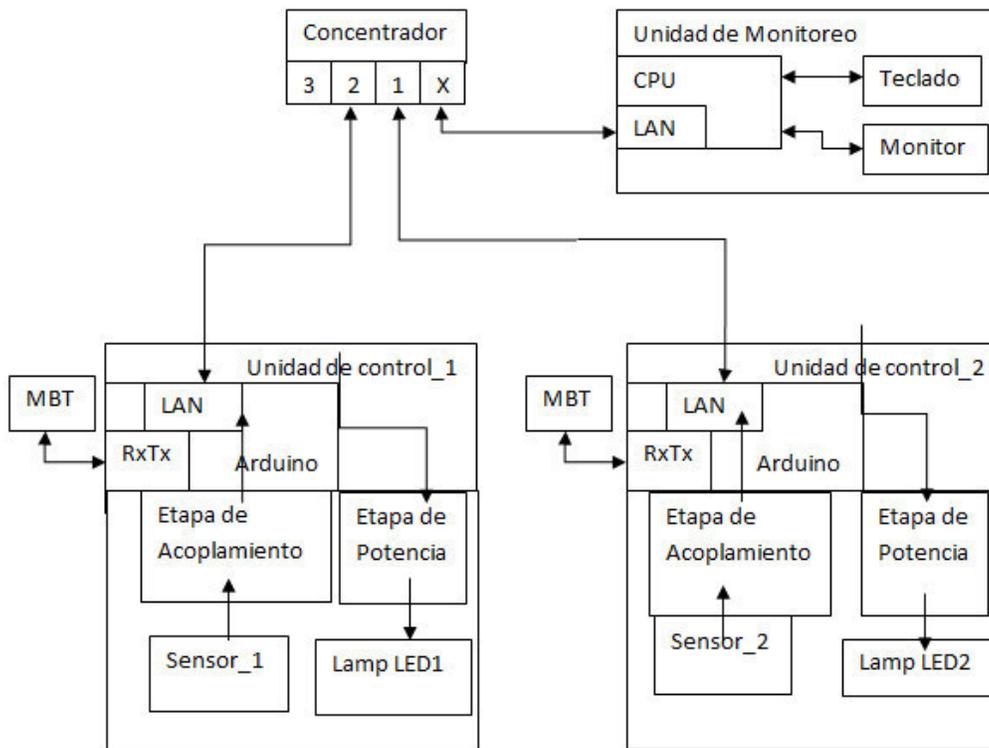


Fig. 1. Diagrama a bloques del hardware; MBT modem Bluetooth

2 Fundamento teórico

2.1 Sistema Operativo Android

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y BlackBerry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma. En la figura 2.1 se observa el logotipo oficial de Android

El sistema permite programar aplicaciones en una variación de Java llamada Dalvik. El sistema operativo proporciona todas las interfaces necesarias para desarrollar aplicaciones que accedan a las funciones del teléfono (como el GPS, las llamadas, la agenda) de una forma sencilla desde Java.



Fig. 2.1 Logotipo de Android.

El sistema operativo Android se usa en teléfonos inteligentes, ordenadores portátiles, notebooks, tabletas, Google TV, relojes de pulsera, auriculares y otros dispositivos. La plataforma de hardware principal de Android es la arquitectura ARM. Hay soporte para x86 en el proyecto Android-x86, y Google TV utiliza una versión especial de Android x86.

Android es considerado como uno de los modelos de negocio más exitosos, pues su desarrollo estratégico contempla los factores que más se tienen en cuenta dentro de las herramientas y metodologías desarrollados por expertos en negocios. Este sistema operativo se ha convertido en un modelo a seguir por desarrolladores de tendencias y negocios de alto impacto.

Android, al contrario que otros sistemas operativos para dispositivos móviles como iOS o Windows iPhone, se desarrolla de forma abierta y se puede acceder tanto al código fuente como a la lista de incidencias donde se pueden ver problemas aún no resueltos y reportar problemas nuevos.

En sus comienzos, Android era eminentemente un sistema operativo pensado para usar con teclado, y gracias a un cursor poder navegar entre las aplicaciones. Desde su comienzo, Android ha sido altamente personalizable.

Poco después, antes del lanzamiento del primer teléfono Android, esta filosofía cambió para convertirse en eminentemente táctil, y poder competir contra el iPhone, presentado 1 año y 9 meses antes.

Las aplicaciones se desarrollan habitualmente en el lenguaje Java con Android Software Development Kit (Android SDK), pero están disponibles otras herramientas de desarrollo, incluyendo un Kit de Desarrollo Nativo para aplicaciones o extensiones en C o C++, Google App Inventor, un entorno visual para programadores.

El desarrollo de aplicaciones para Android no requiere aprender lenguajes complejos de programación. Todo lo que se necesita es un conocimiento aceptable de Java y estar en posesión del kit de desarrollo de software o «SDK» provisto por Google el cual se puede descargar gratuitamente. Todas las aplicaciones están comprimidas en formato APK, que se pueden instalar sin dificultad desde cualquier explorador de archivos en la mayoría de dispositivos.

Principales características que posee Android: Conectividad: Soporta tecnologías de conectividad como Wi-Fi, Bluetooth, GSM/EDGE, UMTS, WiMAX y otras más. Mensajería: Las formas más comunes como SMS y MMS están disponibles además del servicio Push Messaging de Android. Video llamada: Por medio de la versión HoneyComb, Android soporta video llamadas a través de Google Talk.

Soporte multimedia: Puede soportar los formatos más conocidos como JPEG, MP3, MPEG-4, WAV, además de otros como WebM, H.263 y H.264. Multi- táctil: Android cuenta con soporte base para equipos móviles con pantallas multi –táctiles. Almacenamiento: Posee una base SQLite, la cual es utilizada para almacenamiento de datos. Programación. El desarrollo de aplicaciones para Android es sumamente sencillo y lo único que se necesita es un conocimiento básico de Java y poseer el kit de desarrollo de software provisto por Google.

Dispositivos. Android es el sistema operativo con mayor presencia en dispositivos móviles como notebooks, tablets, i-pods, reproductores de mp3 y más. Android es uno de los sistemas operativos que ha logrado establecerse firmemente en el mercado en poco tiempo y además es el sistema con el mayor potencial de desarrollo en el mundo de la telefonía móvil.

Se calcula que en el actualidad hay más 400 000 aplicaciones para Android y que diariamente se activan alrededor de 500 000 equipos móviles. Los precios de los equipos que cuentan con Android oscilan entre los 100 y 600 euros dependiendo de sus diferencias técnicas y modernidad.

2.1 App Inventor

Google App Inventor es una aplicación de Google Labs para crear aplicaciones de software para el sistema operativo Android. De forma visual y a partir de un conjunto de herramientas básicas, el usuario puede ir enlazando una serie de bloques para crear la aplicación. El sistema es gratuito y se puede descargar fácilmente de la web. Las aplicaciones fruto de App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil. En la figura 2.2 se muestra el logotipo de App Inventor.

La aplicación se puso a disposición del público el 12 de julio de 2010 y está dirigida a personas que no están familiarizadas con la programación informática. En la creación de App Inventor, Google se basó en investigaciones previas significativas en informática educativa.

Fue creada a mediados del 2009 por el profesor Harold Abelson del MIT. El editor de bloques de la aplicación utiliza la librería Open Blocks de Java para crear un lenguaje visual a partir de bloques. Estas librerías están distribuidas por Massachusetts Institute of Technology (MIT) bajo su licencia libre (MIT License). El compilador que traduce el lenguaje visual de los bloques para la aplicación en Android utiliza Kawa como lenguaje de programación, distribuido como parte del sistema operativo GNU de la Free Software Foundation.



Fig. 2.2 Logotipo de App inventor.

Para la creación de aplicaciones, App Inventor dispone de una interfaz de programación bastante amigable con el usuario, ya que tiene una lista desplegable de herramientas, donde se encuentran desde los componentes básicos como botones, hasta sensores entre otros. Cada componente o herramienta puede programarse o editarse mediante el uso de sus propiedades que aparecen en la parte derecha de la interfaz al dar clic sobre el elemento a utilizar.

Para crear una aplicación basta con presionar el comando “New” en la paleta de opciones, que abrirá una pequeña ventana que pedirá se ingrese el nombre del proyecto a desarrollar.



Fig.2.3 Comando New.

Una vez realizado este paso, se abrirá la ventana principal de programación, donde se diseña la aplicación del modo deseado. Es necesario mencionar que la ventana principal, despliega sobre el lado derecho una paleta de herramientas, en medio se muestra la pantalla de la aplicación sobre la que se está programando y del lado derecho de la ventana aparecen las propiedades de cada componente seleccionado en la paleta de herramientas, donde se pueden editar y programar según se requiera. Ver figura 2.4

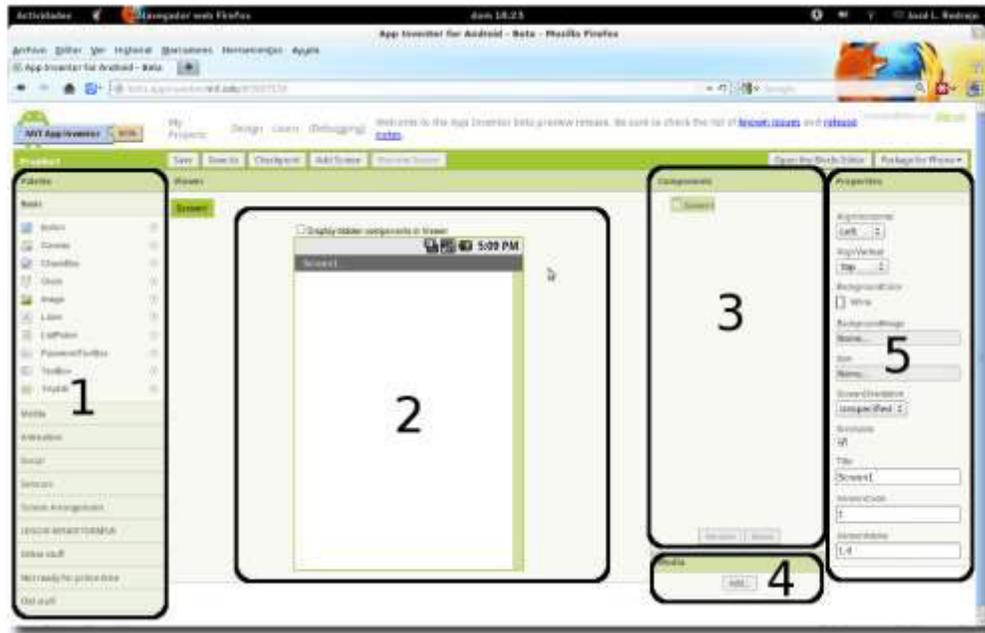


Fig. 2.4 Elementos de la ventana de principal de App Inventor.

La paleta 1.- Contiene todos los elementos que podemos insertar en nuestra aplicación. Hay elementos gráficos como cuadros de texto, botones, lienzo de dibujo (Canvas) y elementos que no se ven en la pantalla del móvil, como base de datos (TinyDB), acelerómetro, cámara de vídeo, los servidores para la conexión de Bluetooth.

Viewer 2.- El visor de la pantalla, simula la apariencia visual que tendrá la aplicación en el móvil. Para añadir un elemento a la pantalla hay que arrastrarlo desde la paleta y soltarlo en el visor.

Componentes 3.- Muestra la lista de los componentes que se han ido colocando en el transcurso del proyecto. Cualquier componente que haya sido arrastrado y soltado desde la paleta al visor aparecerá ahí.

Media 4.- Muestra las distintas imágenes y los sonidos que estarán disponibles para el proyecto. Este cuadro nos permite ver los archivos que se han cargado previamente que serán usados posteriormente. Cualquier archivo de imagen o audio que se quiera usar en la aplicación hay que insertarlo usando este apartado para que esté disponible.

Properties 5.- Cada vez que en el Viewer se selecciona un componente, en Properties aparecerán todos los detalles que se puedan cambiar de ese componente. Por ejemplo, al hacer clic sobre un componente TextBox se podrá cambiar en Properties su color, texto, fuente, entre otros.

A continuación, se describe la manera de cómo comenzar a realizar una aplicación, señalando los pasos a seguir para la creación de un nuevo proyecto, en la figura 2.5 se muestra la pantalla principal de App Inventor para la creación de un nuevo proyecto.

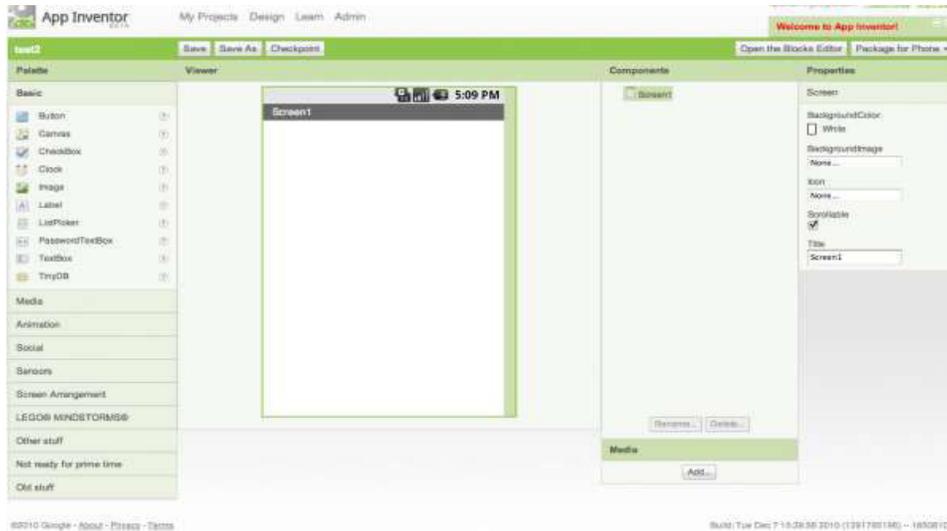


Fig. 2.5 Ventana principal de diseño de App Invento.

La forma de introducir más fácil los componentes desde la paleta de herramientas, es arrastrándolos hacia las Screen o pantalla sobre la que se está trabajando, y ubicarlos en la posición o lugar que se desee.

Una vez introducido el componente se puede disponer de sus propiedades para editarlos o configurar el modo en el que operarán al momento que se inicie la aplicación. Cabe mencionar que la programación de estos componentes se realiza en el editor de bloques de Java, donde se determinan las funciones y/o tareas que deben cumplir los componentes. Ver figura 2.6

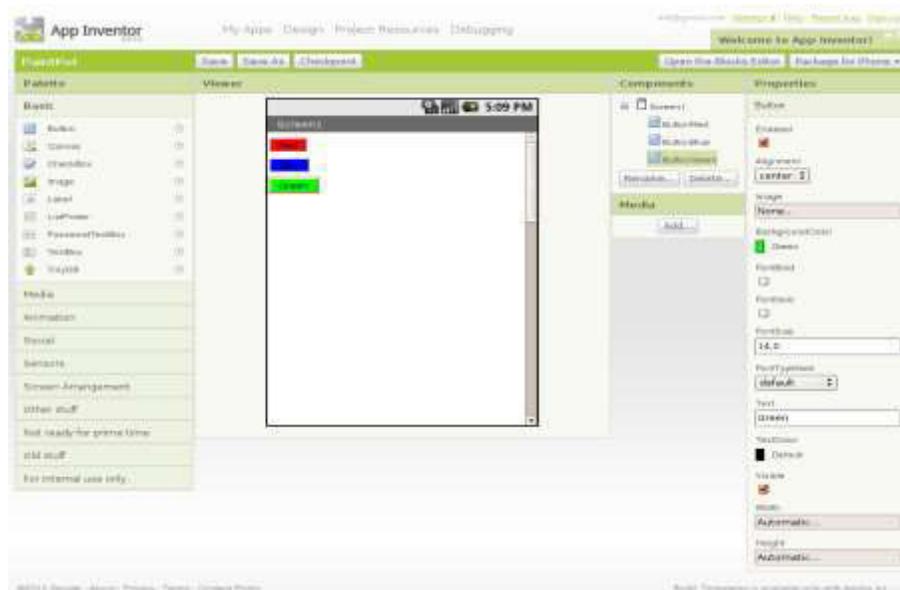


Fig. 2.6 Botones dentro de la Screen y uso de sus propiedades.

Es de esta forma como se puede diseñar la pantalla de la aplicación, donde según sea el proyecto a realizar será, la cantidad y el tipo de componentes que se utilizarán. Se puede disponer de las pantallas necesarias para la aplicación, realizando las transiciones de una a otra con una debida programación. Es necesario que el usuario tenga un mínimo conocimiento en programación para comprender las instrucciones que se le pueden otorgar a cada componente.

Una vez realizado el proceso de diseño de la aplicación es necesario abrir el editor de bloques donde se llevará a cabo la programación de la aplicación, esto se logra presionando el comando **“Open The Blocks Editor”**, ver figura 2.7, que se encuentra en la parte superior derecha de la ventana principal de App Inventor.



Fig. 2.7 Commando Open the Blocks Editor

Entonces, se abrirá la ventana del editor de bloques de Java, donde se llevará a cabo la programación de los componentes utilizados en la pantalla de la aplicación.



Fig. 2.8 Ventana principal del editor de bloques de App Inventor.

El editor de bloques cuenta con una paleta de opciones como son las incorporadas, mis bloques y avanzadas. En las incorporadas se encuentran las funciones básicas de programación como son instrucciones matemáticas, de control, lógicas, textuales, condicionamiento y de colores.

En mis bloques se aparecerán todos los elementos utilizados en la pantalla de la aplicación que se diseñó como primer paso. Y en las opciones avanzadas se disponen de precisamente opciones avanzadas para los componentes utilizados en el diseño de la aplicación.

En la parte superior derecha de la ventana se tienen las opciones de conexión de la aplicación. El comando **“New emulator”** permite correr un emulador en tiempo real de la aplicación realizada, donde aparecerá un emulador con sistema operativo Android simulando las funciones básicas de la aplicación desarrollada. Y el comando **“Connect to device”** permite al usuario empezar a correr la aplicación en el emulador antes mencionado o, cargar dicha aplicación en un dispositivo Android conectado a la Pc o conectado a la red que se está usando en el momento del desarrollo de la aplicación.

Para empezar a programar, basta con seleccionar en la paleta de opciones el comando **“My Blocks”**, ahí se desplegara la lista de todos los componentes utilizados en la pantalla de la aplicación. Seleccionamos el componente a programar y damos clic derecho sobre éste, se abrirán las opciones disponibles para ese componente. Por ejemplo, en la figura 2.9 se muestra algunas opciones que se disponen con un botón.



Fig. 2.9 Componente Boton1 y opciones.

De esta forma se selecciona la instrucción deseada y se arrastra hasta el campo de trabajo del editor de bloques. Una vez esto se selecciona en la paleta de opciones las incorporadas que permitirán asignarle funciones a dicho botón. Por ejemplo, dentro de las primeras aplicaciones para familiarizarse con App Inventor, está la del gatito que al presionar sobre su imagen en la pantalla se escucha el maullido de este.

Para esta función al botón en el diseño de la pantalla se le asigna una imagen, que en este caso es la de una gatito, y en el editor de bloques se programa el botón con la siguiente instrucción como se ve en la figura 2.10.

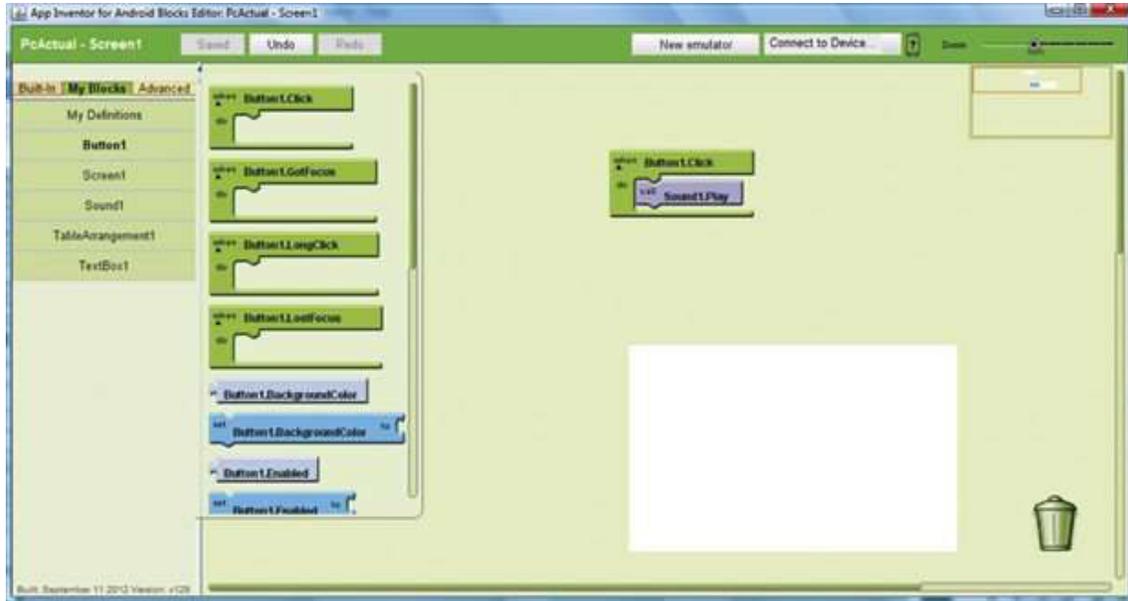


Fig. 2.10 Asignación de la función al botón.

Como se observa en la figura 2.10 se selecciona la instrucción Button1.Click dentro de las opciones del botón 1 y, se abre el componente Sound1 que previamente se asignó en la pantalla de diseño de la aplicación, y buscamos la instrucción “**Sound1.Play**” que indica que activará el sonido cada vez que se presione una vez el botón 1. Es de esta forma como se le otorgan tareas o instrucciones a los componentes utilizados en la aplicación. Una vez realizada la programación se puede descargar la aplicación a la computadora donde se obtendrá un archivo .apk que será compatible únicamente con dispositivos Android. Y con esto se puede estar realizando todo lo que imagine, utilizando App Inventor.

2.2 Módem Bluetooth RN- 41N

Módulo Bluetooth de bajo consumo clase 1, flexible y económico que cumple con el estándar 802.15.1. Soporta diferentes protocolos de comunicación. El sistema está completamente certificado, lo que lo convierte en una solución embebida Bluetooth completa. Con su antena de alto rendimiento tipo chip y su soporte de Bluetooth enhanced data rate (EDR), el RN-41 proporciona una rates de hasta 3Mbps para una distancia de hasta 100m.

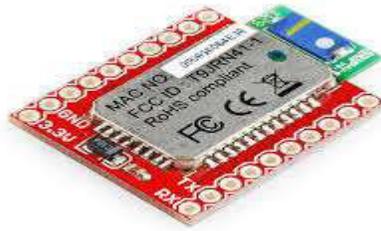


Fig. 2.11 módem Bluetooth RN-41.

Es una excelente solución para agregar comunicación inalámbrica Bluetooth a sistemas existentes.

Características.- Soporta Bluetooth v2.0+EDR, Interfaces de conexión de datos UART (SPP o HCI) y USB (sólo HCI), Soporta rates de datos SPP - 240Kbps (slave), 300Kbps (master), Soporta rates de datos HCI -, 1.5Mbps, 3.0Mbps, Dispone de software para modo HCI o SPP/DUN, Antena tipo chip, Alcance: hasta 100m con línea de vista, Frecuencia: 2.402 ~ 2.48 GHz, Modulación: FHSS/GFSK (79 canales a intervalos de 1MHz), Comunicación segura, encriptación de 128 bits, Corrección de errores, Potencia de salida: 12dBm, Sensibilidad: -80dBm, Rata de transmisión no estándar programable: 200bps hasta 921Kbps, Consumo de corriente en transmisión: 65mA, Consumo de corriente en recepción: 35mA, Voltaje de alimentación: 3.3V, Tamaño: 13.2mm x 25.8mm.

2.3 Arduino Ethernet shield

La Arduino Ethernet Shield permite a una placa Arduino para conectarse a internet. Se basa en la Wiznet W5100 chip de ethernet (hoja de datos). El Wiznet W5100 ofrece una red (IP) apilar capaz de TCP y UDP. Soporta hasta cuatro conexiones de socket simultáneas.

La Shield Ethernet tiene una conexión RJ-45 estándar, con un transformador de línea integrado y Power over Ethernet habilitado. Hay una ranura para tarjetas micro-SD a bordo, que se puede utilizar para almacenar archivos para servir a través de la red. Es compatible con el Arduino Uno y Mega

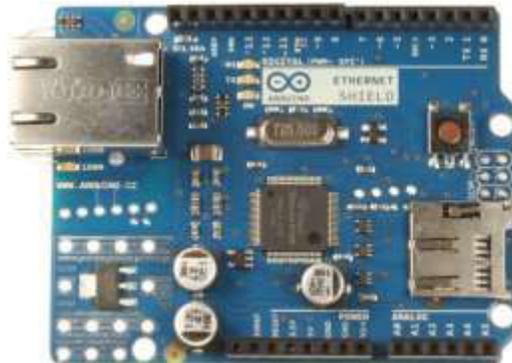


Fig. 2.12 Ethernet Shield

Características.- IEEE802 .3af compatible, Ondulación baja producción y el ruido (100mVpp), Entrada rango de voltaje de 36V a 57V, Sobrecarga y corto circuito de protección, 9V de salida, Alta eficiencia convertidor DC / DC: typ 75% @ 50% de carga, Aislamiento 1500V (entrada a la salida), Ethernet Controller: W5100 con buffer interno 16K, La velocidad de conexión: 10 / 100Mb, Conexión con Arduino en el puerto SPI.

La Shield Ethernet contiene una serie de diodos LED informativos. **PWR:** indica que la placa y la shield están encendidos, **LINK:** indica la presencia de un enlace de red y parpadea cuando la shield transmite o recibe datos, **FULLD:** indica que la conexión de red es full dúplex, **100M:** indica la presencia de un Mb / s 100 conexión de red (en contraposición a 10 Mb/s), **RX:** Parpadea cuando la shield recibe datos, **TX:** parpadea cuando la shield envía datos, **COLL:** parpadea cuando se detectan colisiones de red, **NB.**

2.4 Sensor de movimiento PIR

El sensor de movimiento **PIR “Passive Infra Red”** (figura 2.11) es un dispositivo piroeléctrico que mide los cambios en los niveles de radiación infrarroja emitida por los objetos a su alrededor a una distancia máxima de 6 metros. Como respuesta al movimiento, el sensor cambia su nivel lógico de un “pin”, por lo cual, su uso es extremadamente simple.



Fig. 2.11 Sensor de movimiento PIR.

Características técnicas.- Voltaje de alimentación = 5 vdc, Rango de medición = hasta 6 metros, Salida = estado de un pin TTL, Polaridad de activación de salida seleccionable, mínimo tiempo de calibración.

Calibración.- Al energizarse el sensor PIR requiere de un tiempo de preparación para comenzar a operar de forma adecuada. Esto se debe a que tiene que ocurrir la adaptación a las condiciones propias de operación del ambiente donde fue instalado. Durante este período el sensor “**aprende**” a reconocer el estado de reposo o no movimiento del ambiente. La duración de esta calibración puede estar entre 10 y 60 segundos y es altamente recomendable la ausencia de personas en la vecindad del sensor mientras se calibra.

2.5 Placa Arduino Uno

Arduino Uno es una placa electrónica basada en el microcontrolador **ATmega328**. Cuenta con 14 entradas/salidas digitales, de las cuales 6 se pueden utilizar como salidas **PWM** (Modulación por ancho de pulsos) y otras 6 son entradas analógicas. Además, incluye un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, una cabecera **ICSP** y un botón de reseteo.

La placa incluye todo lo necesario para que el microcontrolador haga su trabajo, basta conectarla a un ordenador con un cable USB o a la corriente eléctrica a través de un transformador.

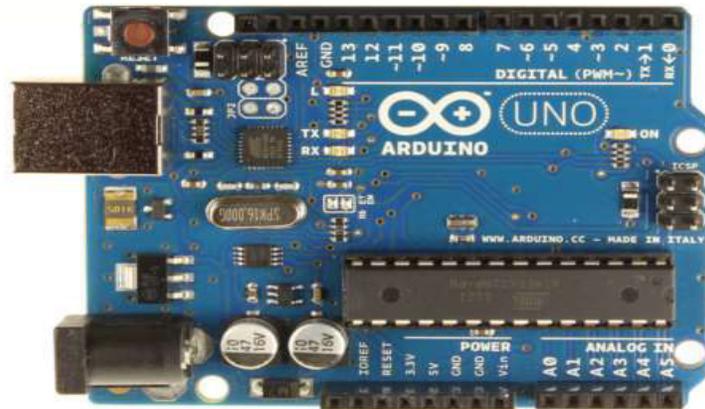


Fig. 2.12 Placa Arduino Uno

Características técnicas de Arduino Uno.- Micro controlador: ATmega328, **Voltaje:** 5V, Voltaje entrada (recomendado): 7-12V, Voltaje entrada, (limites): 6-20V, Digital I/O Pines: 14 (de los cuales 6 son salida PWM), **Entradas Analógicas:** 6, Corriente DC para I/O Pin: 40 mA, Corriente DC para 3.3V Pin: 50 mA, **Memoria flash:** 32 KB (ATmega328) de los cuales 0.5 KB son utilizados para el arranque, SRAM: 2 KB (ATmega328), **EEPROM:** 1 KB (ATmega328), **Velocidad de reloj:** 16 MHz

Terminales Digitales

Las terminales digitales de una placa Arduino pueden ser utilizadas para entradas o salidas de propósito general a través de los comandos de programación **pinMode ()**, **digitalRead ()**, y **digitalWrite ()**. Cada terminal tiene una resistencia **pull-up** que puede activarse o desactivarse utilizando **DigitalWrite ()** (con un valor de HIGH o LOW, respectivamente) cuando el pin está configurado como entrada. La corriente máxima por salida es 40 mA.

Serial.- 0 (**RX**) y 1 (**TX**). Utilizado para recibir (**RX**) y transmitir (**TX**) datos serie **TTL**. En el Arduino Diacemila, estas terminales están conectadas a las correspondientes patas del circuito integrado conversor **FTDI USB** a **TTL** serie. En el Arduino BT, están conectados a las terminales correspondientes del módulo Bluetooth WT11. En el Arduino Mini y el Arduino LilyPad, están destinados para el uso de un módulo serie TTL externo (por ejemplo el adaptador Mini-USB).

Interruptores externos.- 2 y 3. Estas terminales pueden ser configuradas para disparar una interrupción con un valor bajo, un pulso de subida o bajada, o un cambio de valor. Mira la función **attachInterrupt ()** para más detalles.

PWM.- 3, 5, 6, 9, 10, y 11. Proporcionan salidas PWM de 8 bit con la función **analogWrite ()**. En placas con **ATmega8**, las salidas PWM solo están disponibles en los pines 9, 10, y 11. Reset BT: 7. (solo en Arduino BT) Conectado a la línea de reset del módulo bluetooth. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).

Estas terminales soportan comunicación SPI. Aunque esta funcionalidad esta proporcionada por el hardware, no está incluida actualmente el lenguaje Arduino. LED: 13. En el Diacemila y el LilyPad hay un led en placa conectado al pin digital 13. cuando el pin tiene valor HIGH, el LED está encendido, cuando el pin está en LOW, está apagado

Pines Analógicos

Los pines de entrada analógicos soportan conversiones analógico-digital (ADC) de 10 bit utilizando la función `analogRead ()`. Las entradas analógicas pueden ser también usadas como pines digitales: entrada analógica 0 como pin digital 14 hasta la entrada analógica 5 como pin digital 19. Las entradas analógicas 6 y 7 (presentes en el Mini y el BT) no pueden ser utilizadas como pines digitales.

I2C: 4 (SDA) y 5 (SCL). Soportan comunicaciones I2C (TWI) utilizando la librería `Wire` (documentación en la página web de Wiring). Pines de alimentación: VIN (a veces marcada como "9V"). Es el voltaje de entrada a la placa Arduino cuando se está utilizando una fuente de alimentación externa (En comparación con los 5 voltios de la conexión USB o de otra fuente de alimentación regulada). 5V. La alimentación regulada utilizada para alimentar el microcontrolador y otros componentes de la placa. Esta puede venir de VIN a través de un regulador en placa o ser proporcionada por USB u otra fuente regulada de 5V. 3V3. (solo en el Diacemila) Una fuente de 3.3 voltios generada por el chip FTDI de la placa. GND. Pines de tierra.

Otros Pines

AREF. Referencia de voltaje para las entradas analógicas. Utilizada con la función `analogReference ()`. Reset. (Solo en el Diacemila) definida como LOW para resetear el microcontrolador. Utilizada típicamente para añadir un botón de reset a shields que bloquean el de la placa principal.

2.6 Lámpara tipo Led



Fig. 2.13 Lámpara tipo panel.

Características.- Flujo luminoso de 950 lúmenes, Potencia 18 watts, Alimentación 120 v – 60 hz, 50,000 horas de vida útil.

3. Desarrollo

El presente proyecto permite adaptarlo a cualquier tipo de edificación, ya que la idea principal es controlar el conjunto de luminarias de las que conste dicho edificio, por lo que, basta con hacer ciertas modificaciones al sistema para acoplarlo a lo que se necesite.

El sistema que se presenta a continuación está enfocado a una casa hogar, donde, para demostrar la funcionalidad del sistema únicamente se consideran tres áreas a controlar: iluminación del exterior, del pasillo central en el interior y de un cuarto principal en el hogar.

3.1 Aplicación SO Android

A continuación se muestra en la figura 3.1 la pantalla principal de la aplicación creada en App Inventor.

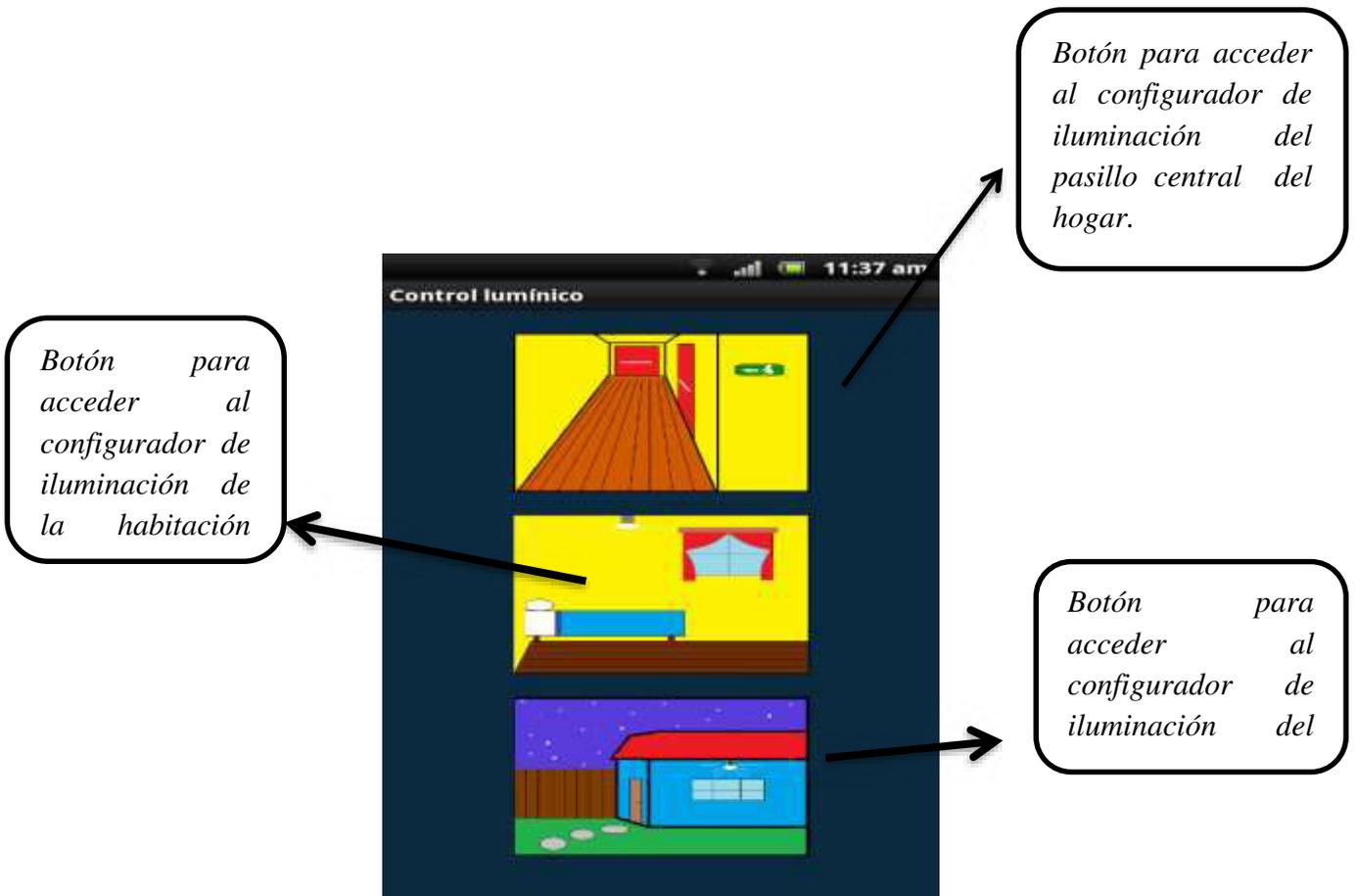


Fig. 3.1 Pantalla principal de la aplicación.

Para poder realizar la interfaz, se analizó la idea principal de un sistema que fuera sencillo, ingenioso, novedoso y eficaz, de modo que el software permitiera al usuario determinar de forma sencilla dos valores límites [Imax, Imin], de manera que se pudiera establecer primero el valor Imax, y luego guardarlo en el micro maestro; una vez realizado se finalizara realizando el mismo proceso para el valor Imin.

Tomando en cuenta que la aplicación está diseñada para un casa hogar, se creó la interfaz principal, donde se disponen de 3 botones, cada uno indica el área a la cual se accede al presionar dicho botón. Cada botón está programado para que al ser presionado, conecte al dispositivo móvil con el módulo bluetooth y al mismo tiempo, abra la interfaz de configuración de dicha área seleccionada.

Cabe mencionar, que al abrirse la interfaz de configuración, comprueba que en efecto la conexión entre ambas partes del sistema se realizó con éxito, si esto no sucede, significa que hubo algún problema en la conexión bluetooth

Una vez, que se realizó con éxito la conexión, y que se abrió la interfaz de configuración se mostrara como en la figura 3.2.



Fig. 3.2 Pantalla de configuración.

En esta etapa, la aplicación permite configurar el valor Vmax y Vmin mediante un componente Slider1 que se desliza y ajusta el valor deseado de entre un rango de 0 a 100, el valor del Slider1 se muestra en una etiqueta ubicada arriba de éste componente.

Este valor indica el porcentaje al cual la lámpara iluminará, realizando un algoritmo de programación en el micro maestro (Arduino) y utilizando un control PWM (modulación por ancho de pulso).

Una vez que el valor se ha fijado, se cuenta con 2 botones [Vmax, Vmin] que al presionar cada uno de ellos envían y guardan el valor en el micro maestro. Es necesario destacar que este es un proceso que debe llevar un orden, es por eso que la aplicación está programada para que primero se configure el valor Vmax y se envíe como primer dato, después se configure el valor Vmin y se envíe como segundo dato, de lo contrario al no tener ese orden, la aplicación no permite enviar ningún dato.

Por último, se añadió un botón que permite una vez finalizado la configuración de los valores para el área deseada, regresar a la pantalla principal de la aplicación y desconectar al módulo bluetooth de esa área con el dispositivo móvil.

3.1.1 Programación en el editor de bloques de App Inventor

En la figura 3.3 y 3.4 se muestra la programación realizada en el editor de bloques.



Fig. 3.3 Captura de pantalla del editor de bloques de App Inventor.



Fig. 3.4 Captura de pantalla del editor de bloques de App Inventor.

```

initialize global MAC1 to "00:06:66:09:85:B1"
initialize global MAC to "00:06:66:09:85:F3"
  
```



Declaración de dos variables globales llamadas MAC, MAC1, que corresponden a las direcciones MAC de los módulos Bluetooth.

```

when Button1 Click
do
  if call BluetoothClient1 Connect address "00:06:66:09:85:F3"
  then
    set TableArrangement1 Visible to false
    set TableArrangement2 Visible to true
  
```



Instrucción para el componente Button1 cuando se presione una vez (1 Clic):

Si el BluetoothClient1 con la dirección MAC que ahí se define, está conectado, entonces, el componente TableArrangement1 pasará del estado Visible "TRUE" a "FALSE", y el componente TableArrangement2 pasará del estado Visible "FALSE" a "TRUE".

```

when Button2 .Click
do
  if call BluetoothClient1 .Connect
      address "00:06:66:09:85:B1"
  then
    set TableArrangement1 .Visible to false
    set TableArrangement2 .Visible to true
  
```

Instrucción para el componente Button2 cuando se presione una vez (1 Clic):
 Si el BluetoothClient1 con la dirección MAC que ahí se define, está conectado, entonces, el componente TableArrangement1 pasará del estado Visible "TRUE" a "FALSE", y el componente TableArrangement2 pasará del estado Visible "FALSE" a "TRUE".

```

when Slider1 .PositionChanged
  thumbPosition
do
  set Label2 .Text to format as decimal number Slider1 .ThumbPosition
  places 0
  
```

Instrucción para el componente Slider1 cuando cambia de posición:
 El valor de la posición del componente Slider1 convertirlo a formato decimal y establecerlo en la etiqueta Label2.

```

when Button4 .Click
do
  call BluetoothClient1 .SendText
  text Label2 .Text
  set Button4 .Enabled to false
  set Button5 .Enabled to true
  set Slider1 .MaxValue to Label2 .Text
  set Slider1 .ThumbPosition to 0
  
```

Instrucción para el componente Button4 cuando se presione una vez (1 Clic):
 Enviar el texto de la etiqueta Label2 por medio del BluetoothClient1. Establecer el estado Enable del Button4 de "TRUE" a "FALSE", establecer el estado Enable del Button5 de "FALSE" a "TRUE". Establecer el texto de la etiqueta Label2 como valor máximo del componente Slider1. Establecer la posición del Slider1 a 0.

```

when Button5 . Click
do
  call BluetoothClient1 . SendText
  text Label2 . Text
  set Button4 . Enabled to true
  set Button5 . Enabled to false
  set Slider1 . MaxValue to 100
  set Slider1 . ThumbPosition to 0

```

Instrucción para el componente Button6 cuando se presione una vez (1 Clic):

Enviar el texto de la etiqueta Label2 por medio del BluetoothClient1. Establecer el estado Enable del Button4 de "FALSE" a "TRUE", establecer el estado Enable del Button5 de "TRUE" a "FALSE". Establecer como valor máximo del componente Slider1 el número 100. Establecer la posición del Slider1 a 0.

```

when Button6 . Click
do
  set TableArrangement1 . Visible to true
  set TableArrangement2 . Visible to false
  call BluetoothClient1 . Disconnect
  set Slider1 . ThumbPosition to 0
  set Button4 . Enabled to true

```

Instrucción para el componente Button6 cuando se presione una vez (1 Clic):

Establecer el estado Visible del componente TableArrangement1 de "FALSE" a "TRUE", establecer el estado Visible del componente TableArrangement2 de "TRUE" a "FALSE", desconectar el BluetoothClient1, establecer a "0" la posición del componente Slider1, establecer el estado Enabled del componente Button4 de "FALSE" a "TRUE".

3.2 Modo de enlace WEB

Para crear la página web y el enlace con arduino y Ethernet shield se parte desde saber que es un servidor web. Un servidor web es una computadora que guarda y transmite datos vía Internet. La información que almacenan es en forma de página Web. Cuando un cliente (navegador Web) accede a un servidor Web, envía una petición HTTP que recibe el servidor Web. Luego este envía la información a través de protocolo HTTP al cliente en formato HTML.

El Arduino está conectado a la red con una IP que se le asignó. El código HTML de la web que enviará se debe escribir en el sketch. Cuando se escribe esta dirección IP en un buscador Web de un ordenador conectado a la misma red (o desde fuera a través de Internet si se abre un puerto que direcciona a la IP del Arduino), se está enviando una petición HTTP. Cuando el Arduino lea esta petición, enviará el código HTML para que se pueda visualizar la página Web en el navegador.

Detectar una petición HTTP

Para detectar una petición HTTP con el código de Arduino, primero tuvo que crear un cliente Web. Luego cuando el cliente envía una petición HTTP al Arduino, este la lee carácter por carácter a través de una variable char. Sabiendo que una petición HTTP termina con una línea en blanco, se le asignó una condición que detecte la línea en blanco para que, posteriormente, envíe la respuesta HTML figura 3.5.

```
1 EthernetClient client = server.available(); //Creamos un cliente Web
2 //Cuando detecte un cliente a través de una petición HTTP
3 if (client) {
4   boolean currentLineIsBlank = true; //Una petición HTTP acaba con una línea en blanco
5   while (client.connected()) {
6     if (client.available()) {
7       char c = client.read(); //Leemos la petición HTTP carácter por carácter
8       Serial.write(c); //Visualizamos la petición HTTP por el Monitor Serial
9
10      //Cuando reciba una línea en blanco, quiere decir que la petición HTTP ha acabado y
11      //el servidor Web está listo para enviar una respuesta
12      if (c == '\n' && currentLineIsBlank)
13      {
14        //Código HTML
15      }
```

Fig. 3.5 Captura de pantalla código detección de petición

Al abrir el Monitor Serial se observa la petición HTTP del cliente Web figura 3.6

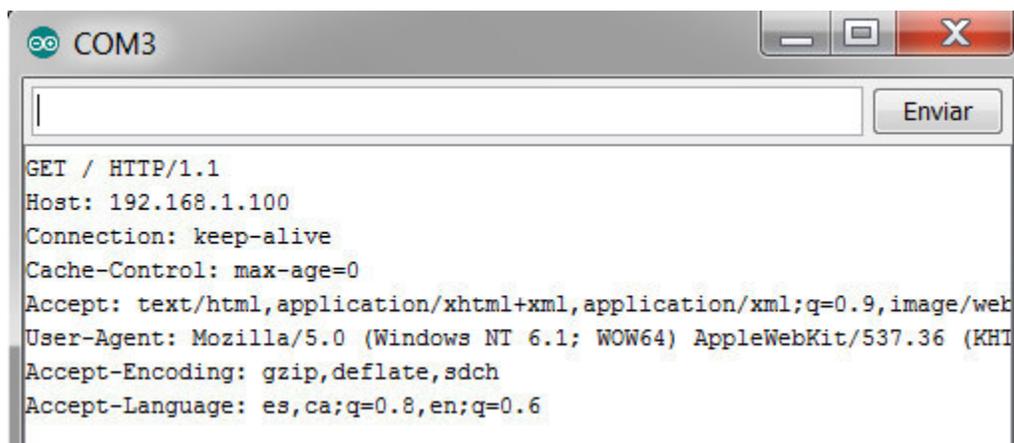


Fig. 3.6 Petición HTTP

Enviando parámetros a través de HTML

En la página Web habrá botones. Al pulsarlos, cada uno envía unos parámetros diferentes para poder identificarlos en el código Arduino. Para enviar esos parámetros a través de HTML se utiliza el método URL encoded. Quiere decir que los datos vienen por la URL si se inician con el símbolo “?”. Para esto se usa por ejemplo un botón. Figura 3.7

```
1 | client.println("<button onClick=location.href='./?LED=ON\' style='margin:auto;background-
```

Fig. 3.7 Código de un botón en HTML

Al observar el símbolo “?”, se nota que los parámetros que se envían son “LED=ON”. Si se vuelve a abrir el Monitor Serial figura 3.8, se verá que el Arduino recibe una nueva petición HTTP pero con estos parámetros escritos después de GET /.

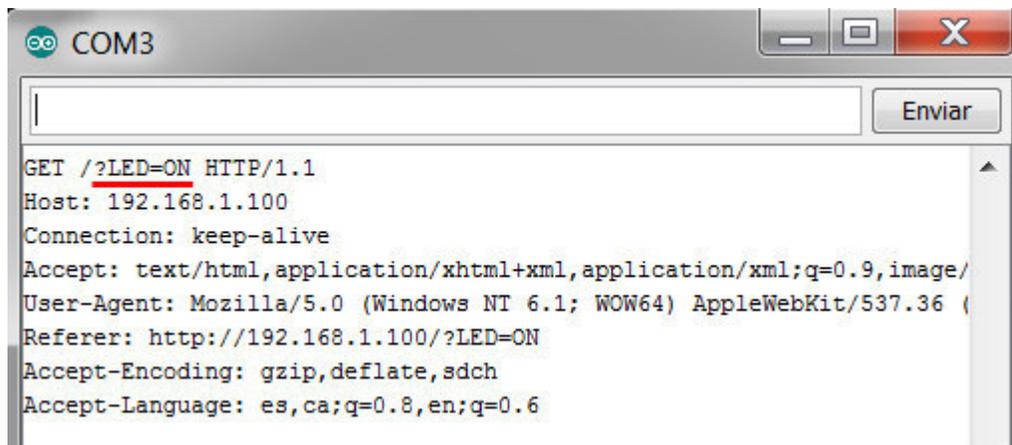


Fig. 3.8 Nueva petición HTTP

Lo siguiente es detectar esta parte del texto. Como la petición HTTP se ha almacenado en una variable tipo char, quiere decir que almacena los caracteres de manera individual. Por lo tanto se tiene que transformar estos caracteres a una cadena de caracteres (String). Para esto se crea un String vacío y se une éste String con los caracteres con la función concat(). De esta manera tenemos la petición HTTP en un String figura 3.9

```
1 | EthernetClient client = server.available(); //Creamos un cliente Web
2 | //Cuando detecte un cliente a través de una petición HTTP
3 | if (client) {
4 |   boolean currentLineIsBlank = true; //Una petición HTTP acaba con una línea en blanco
5 |   String cadena=""; //Creamos una cadena de caracteres vacía
6 |   while (client.connected()) {
7 |     if (client.available()) {
8 |       char c = client.read();//Leemos la petición HTTP carácter por carácter
9 |       Serial.write(c);//Visualizamos la petición HTTP por el Monitor Serial
10 |      cadena.concat(c);//Unimos el String 'cadena' con la petición HTTP (c). De esta manera convertimos la petición HTTP a un String
```

Fig. 3.9 Detección de petición y guardado en variable string.

3.3 Etapa de potencia

La etapa de potencia está compuesta por 2 circuitos de control: uno de aislamiento que se observa en la figura 3.10, y el otro es el detector de cruce por cero (**Zero Crossing**) figura 3.12.

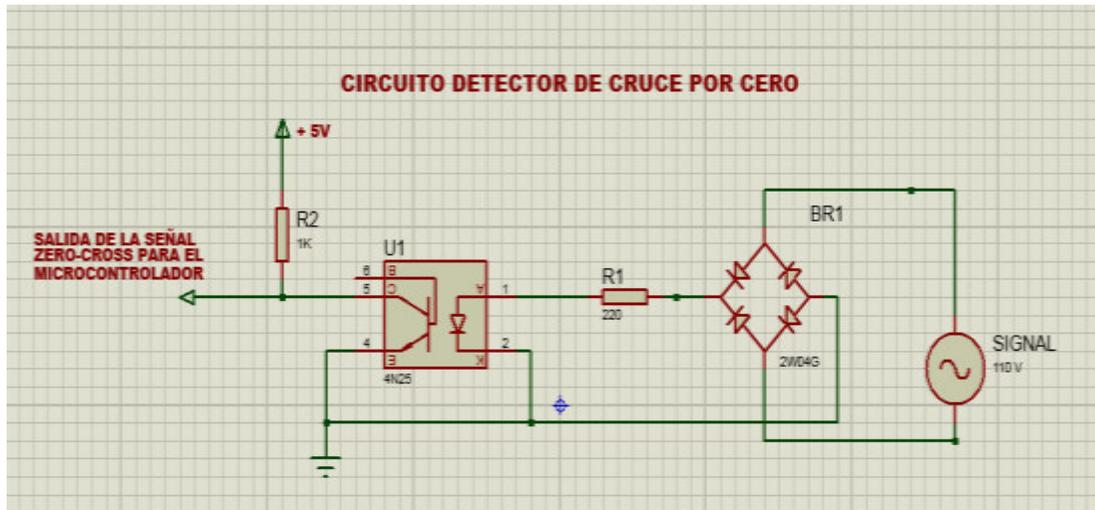


Figura 3.10 Circuito detector de cruce por cero.

Este es el diseño del detector de cruce por cero opto aislado. Su misión es conectarse al microcontrolador para controlar el cruce por cero de la señal de corriente alterna de la red eléctrica y poder hacer el disparo del Triac y controlar la fase de forma sincronizada. Solamente da un impulso con el semiciclo positivo de la señal de la red.

Su funcionamiento consiste en la rectificación de la señal de corriente alterna obteniendo solamente los semiciclos positivos usando el puente de diodos; el opto acoplador aísla y recibe la señal positiva dejando pasar únicamente una señal pulsante en cada cruce por cero de los semiciclos positivos como se observa en la figura 3.11.

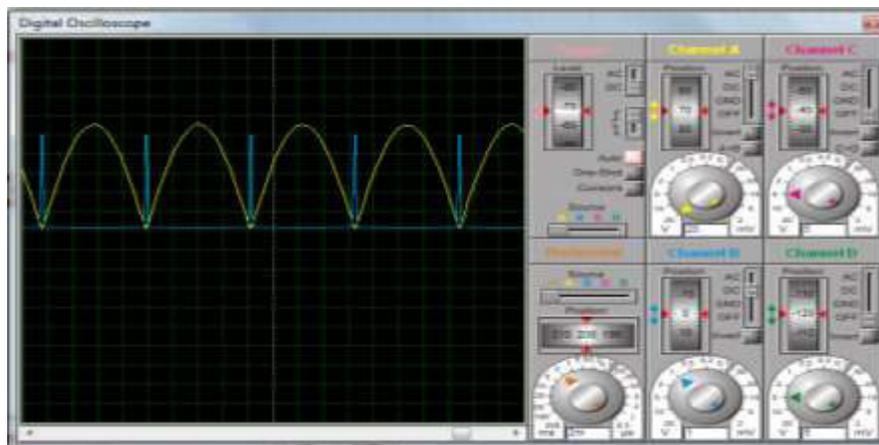


Figura 3.11 Señal rectificada y señal de cruces por cero en el osciloscopio.

Esta señal pulsante obtenida se envía al microcontrolador que la recibirá y codificará en dos estados lógicos (0 y 1), esto le permitirá al microcontrolador saber el momento en que activará el envío de la señal **PWM** para lograr el control de potencia mediante el ángulo de trabajo que se indique por el usuario.

La señal **PMW** será enviada al circuito de potencia en el que el principal componente es el **TRIAC**. Figura 3.12

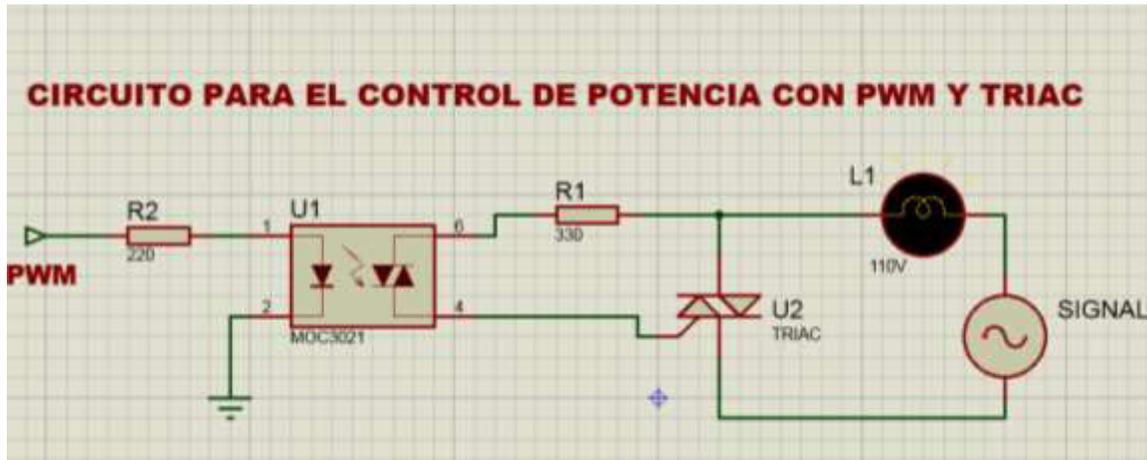


Figura 3.12 Circuito de potencia con **PWM** y **TRIAC**.

El **TRIAC** es un dispositivo semiconductor que pertenece a la familia de los dispositivos de control: los tiristores. Este es en esencia la conexión de dos tiristores en paralelo pero conectados en sentido opuesto y compartiendo la misma compuerta. **El TRIAC** sólo se utiliza en corriente alterna y al igual que el tiristor, se dispara por la compuerta. Como el **TRIAC** funciona en corriente alterna, habrá una parte de la onda que será positiva y otra negativa.

A continuación, se muestra en la figura 3.13 el circuito que se utilizará para el funcionamiento del proyecto, es en si la sinergia de ambos circuitos que permiten el control de la potencia entregada a la lámpara tipo LED.

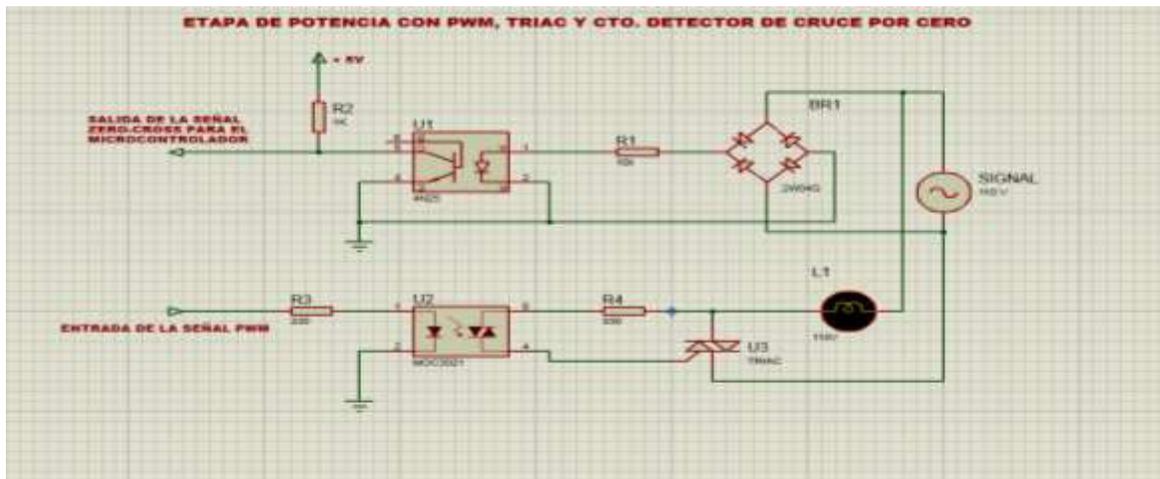


Figura 3.13 Etapa de potencia.

Es importante mencionar que el micro controlador es el encargado de procesar la información recibida del circuito detector de cruce por cero y del envío de la señal **PWM** para el activamiento del **TRIAC** para el control de la potencia entregada a la lámpara.

4. Diagrama del sistema

En la figura 4.1 se muestra el diagrama del sistema implementado

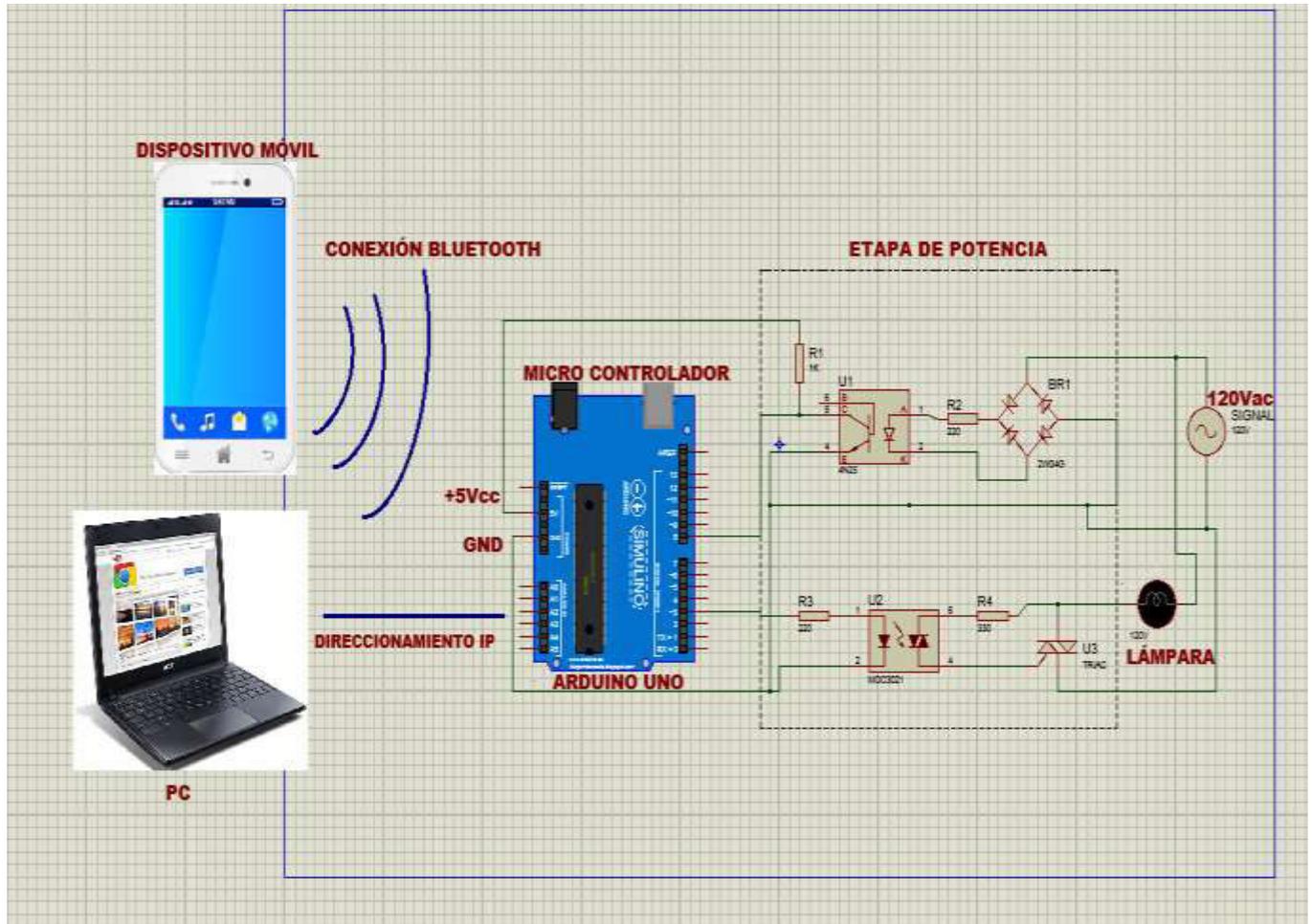


Figura 4.1 Diagrama del sistema.

Como se observa en la figura 4.1 el dispositivo móvil hace un enlace con la tarjeta **Arduino uno** mediante el módulo **Bluetooth** para el envío de datos, mientras que la **PC** hace su enlace con una **dirección IP** que está disponible en la **red de Internet**. Una vez obtenidos los datos, el micro controlador (**Arduino Uno**) hace el procesamiento de los valores obtenidos y espera a que el circuito detector de cruce por cero le indique el momento exacto para poder enviar al **TRIAC** la señal **PWM** para el control de la potencia indicada por el usuario.

4. Resultados y conclusiones

4.1 Resultados

Transmisión y recepción de datos.- Envío de datos de la Aplicación Android y recepción en el micro-maestro (Arduino). El diseño que se escogió para la aplicación funcionó perfectamente y el modo de operar también. La transmisión de los valores se realiza sin problemas y el módulo Bluetooth se comportó establemente. Se tenía la duda si los componentes utilizados en la aplicación funcionarían como se esperaba pero no se tuvo inconvenientes y los datos son exactos y tal y como se envían desde la aplicación.

Transmisión y recepción de datos.- Envío de datos de la página Web y recepción en el micro-maestro (Arduino).

La programación HTML desarrollada para el envío de datos desde el navegador resultó bastante apropiada, los datos se recibieron en el micro maestro de forma eficaz y se logró que el envío de datos fuera aún más eficiente y práctica.

Resultados generales

El desempeño del sistema en conjunto es muy satisfactorio, las luminarias operan de acuerdo a la intensidad que se les indica considerando siempre el estado del sensor de presencia, con esto se logra que exista un ahorro de energía en la iluminación del inmueble.

Por otra parte y de igual forma el sistema de navegación web para poder controlar el micro-maestro funciona con total exactitud y mucha estabilidad en el envío y recepción de datos

Podemos constatar que el método de envío de datos fueron estables y muy buen funcionamiento el micro maestro funciona a la perfección en ambos casos y el sistema como tal funciona perfectamente bajo los datos enviados por cualquiera de los dos métodos y el ahorro de energía es muy considerable dado a que solo usaremos la luz cuando se necesite y no en todo momento.

4.2 Observaciones y sugerencias

El presente proyecto se concluyó con la única observación de implementar más áreas a controlar desde la aplicación. Y como sugerencias para futuras investigaciones, sería implementar un medidor de luxes en exteriores para tener un sistema más completo y autónomo, ya que de esta forma el sistema podría monitorear la cantidad de luz existente en el área y complementar la diferencia con las lámparas tipo Led.

4.3 Conclusiones

Con la implementación del software que se presenta en este estudio se logra obtener los valores deseados con una eficacia aceptable. El comportamiento del módulo Bluetooth, así como la ethernet shield es estable y la velocidad de transmisión de 9600 baudios hace efectiva la transmisión en el envío y recepción de datos.

Se realiza una interfaz cómoda de forma que el usuario pueda interactuar sin ninguna dificultad pero con muy buenos resultados.

Dentro de las ventajas que presenta podemos destacar que gracias a los módulos bluetooth y routers se puede realizar transmisión de datos inalámbricamente a distancias considerables sin ningún fallo de transmisión.

ANEXO A

PROGRAMACION DEL MICROCONTRADOR ARDUINO.

```
String dato;          DECLARAMOS LA CADENA DATO
int valor=0;         VARIABLE "VALOR" TIPO ENTERO IGUAL A CERO.
int A=0;             VARIABLE "A" TIPO ENTERO IGUAL A CERO.
int Vmax=0;         VARIABLE "Vmax" TIPO ENTERO IGUAL A CERO.
int Vmin=0;        VARIABLE "Vmin" TIPO ENTERO IGUAL A CERO.
int b=0;            VARIABLE "b" TIPO ENTERO IGUAL A CERO.
int LAMPARA =3;    LAMPARA CONECTADA AL PIN DIGITAL 3.
int SENSOR =7;     SENSOR CONECTADO AL PIN DIGITAL 7.
```

```
void setup ()
{
Serial.begin (9600);    INICIAR PUERTO SERIAL A 9600 BAUDIOS
pinMode (3, OUTPUT);  PIN DIGITAL 3 DECLARADO COMO SALIDA
pinMode (8, OUTPUT);  PIN DIGITAL 8 DECLARADO COMO SALIDA
pinMode (7, INPUT);   PIN DIGITAL 7 DECLARADO COMO ENTRADA
}
```

```
void loop ( )
{
while (Serial.available ( ) ) MIENTRAS EL PUERTO SERIAL ESTÉ DISPONIBLE
```

```

{
delay (10);           ESPERAR 10 MILISEGUNDOS
if (Serial.available ( ) > 0)   SI EL PUERTO SERIAL DISPONIBLE CONTIENE DATOS
MAYORES A CERO
{
char a = Serial.read ();  LA VARIABLE "a" TIPO CHAR SERA IGUAL A LA LECTURA DEL
PUERTO SERIAL
dato += a;               LA CADENA "DATO" SERA IGUAL A EL VALOR DE "a".
}
}
if (dato.length ( ) >0)   SI LA LONGITUD DE LA CADENA "DATO" ES MAYOR A CERO
{
int x= dato.toInt ( );    CONVERTIR LA CADENA "DATO" A ENTERO Y AGREGAR EL
VALOR A LA VARIABLE      "X" TIPO ENTERO
if (A==0)                SI LA VARIABLE "A" ES IGUAL A CERO.
{
b= x * 2.5;              LA VARIABLE "b" SERA IGUAL AL VALOR DE "x"
MULTIPLICADO POR 2.5.
if ( x > 0 )             SI LA VARIABLE "x" ES MAYOR QUE CERO.
{
Vmax = b;               LA VARIABLE "Vmax" SERA IGUAL AL VALOR DE LA VARIABLE
"b"
Serial.println ( Vmax );  IMPRIMIR EN PUERTO SERIE EL VALOR DE LA VARIABLE
"Vmax"
A=1;                   LA VARIABLE "A" ES IGUAL A UNO
b=0;                   LA VARIABLE "b" ES IGUAL A CERO
x=0;                   LA VARIABLE "x" ES IGUAL A CERO
}
}
}
dato= " ";             LIMPIAR EL VALOR DE "DATO"

```

```

if (A==1)          SI "A" ES IGUAL A UNO
{
  b=x * 2.5;      LA VARIABLE "b" SERA IGUAL AL VALOR DE "x"
MULTIPLICADO POR 2.5
  if ( x > 0 )    SI "x" ES MAYOR CERO.
  {
    Vmin=b;      LA VARIABLE "Vmin" ES IGUAL A LA VARIABLE "b".
    Serial.println (Vmin);  IMPRIMIR EN PUERTO SERIE EL VALOR Vmin.
    ESCANEO ( );  LLAMAR A LA FUNCION ESCANEO.
    A=0;        LA VARIABLE "A" ES IGUAL A CERO.
    x=0;        LA VARIABLE "x" ES IGUAL A CERO.
    b=0;        LA VARIABLE "b" ES IGUAL A CERO.
  }
}
dato = " ";     LIMPIAR EL VALOR DE "DATO".
}
}
void ESCANEO ( )  FUNCION ESCANEO.
{
  if (digitalRead ( 7 ) ==HIGH )  SI LA LECTURA DEL PIN DIGITAL 7 ES IGUAL A
ALTO.
  {
    digitalWrite ( 8 , HIGH );    ESCRIBIR EN EL PIN DIGITAL 8 EL VALOR ALTO
    ENCENDER ( );                LLAMAR A LA FUNCION ENCENDER.
  }
  if (digitalRead ( 7 ) ==LOW )   SI LA LECTURA DEL PIN DIGITAL 7 ES IGUAL A
BAJO.

```

```

{
digitalWrite ( 8 , LOW );           ESCRIBIR EN EL PIN DIGITAL 8 EL VALOR
BAJO.
ESCANEAO ();                       LLAMAR A LA FUNCION ESCANEAO.
}
}

void ENCENDER ( )                   FUNCION ENCENDER
{
  for (valor = 0; valor <=Vmax; valor+=5) ESTABLECER PARA LA VARIABLE VALOR
VALORES DESDE CERO A 255.
  {
    analogWrite (3, valor);          ESCRIBIR EN EL PIN ANALOGICO 3 EL VALOR DE LA
VARIABLE VALOR.
    Delay (70);                     ESPERAR 70 MILISEGUNDOS
  }
  Delay (5000);                     ESPERAR 5,000 MILISEGUNDOS.

  while (digitalRead (7)==HIGH)      MIENTRAS LA LECTURA DEL PIN DIGITAL 7
SEA ALTO.
  {
    Delay (3000);                   ESPERAR 3,000 MILISEGUNDOS.
  }
  if (digitalRead (7)==LOW)          SI LA LECTURA DEL PIN DIGITAL 7 ES BAJO.
  {
    APAGAR ();                      LLAMAR A LA FUNCION APAGAR.
  }
  ESCANEAO ();                      LLAMAR A LA FUNCION ESCANEAO.
}

```

```

void APAGAR ( )                                FUNCION APAGAR
{
  for (valor=Vmax; valor>=Vmin; valor-=5)      ESTABLECER PARA LA VARIABLE "VALOR"
LOS VALORES DESDE EL VALOR DE "Vmax" HASTA EL VALOR DE "Vmin"
  {
    analogWrite (3, valor);                    ESCRIBIR EL VALOR ANALOGICO EN EL PIN 3 EL VALOR
DE LA VARIABLE VALOR.
    Delay (70);                                ESPERAR 70 MILISEGUNDOS
  }
  ESCANEEO ( );                                LLAMAR A LA FUNCION ESCANEEO.
}

```

5. Bibliografía

- 1 Tesis, Diseño de una aplicación móvil para la consulta académica de la fiis-utp. AGUIRRE CHACÓN, Luis Teodoro Perú.
- 2 García Padilla Rubén, Aplicación Android para Supermercados, Tesis de Titulación en Ingeniería Informática Técnica de Gestión, Facultad de Informática de Barcelona, Barcelona-España, 2011.
- 3 <http://electronicapractica2012.blogspot.mx/2012/06/scr-y-triac.html>
- 4 “Mobile Phone Programming and its Application to Wireless Networking”, Frank H.P. Fitzek y Frank Reichert. Editorial Springer, 2007.
- 5 Dirección Web de App Inventor. <http://beta.appinventor.mit.edu/>. Enck, William, Damien Oceau, Patrick McDaniel y Swarat Chaudhuri; a Study of Android Application Security. En Proceedings of the 20th USENIX Security Symposium, August 2011. <http://www.enck.org/pubs/enck-sec11.pdf>.
- 6 Google: Android SDK Developing Tools. <<http://developer.android.com/guide/developing/tools/>>, [Septiembre 2012].
- 7 Share, Net Market: Mobile/Tablet Operating System Market Share. <http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1>.
- 8 Android Studio 0.2.5 Lanzamiento. http://translate.googleusercontent.com/translate_c?depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://tools.android.com/recent/androidstudio025released&usg=ALkJrh0KEfgoz8rsoJxNOTXRuM-Nxe51g
- 9 Introducción a Android Studio. http://translate.googleusercontent.com/translate_c?depth=1&hl=es&ie=UTF8&prev=_t&rurl=translate.google.com&sl=en&tl=es&u=http://developer.android.com/sdk/installing/studio.html&usg=ALkJrhjr41zMKkqB6rAsNBPEINy0NeB6Kw.