EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Control de iluminación mediante conectividad multiprotocolo inalámbrica enfocada a recintos semiabiertos, utilizando tecnologías digitales de acceso libre, seguras y de bajo costo

presentada por

Ing. Luis Bryan Dominguez Barrera

como requisito para la obtención del grado de

Maestro en Ciencias en Ingeniería Electrónica

Director de tesis

Dr. Jaime E. Arau Roffiel

Codirector de tesis

Dr. Gabriel Calzada Lara

Cuernavaca, Morelos, México. Diciembre de 2020.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 12/abril/2021
No. de Oficio: DIE/090/2021
Asunto: Aceptación de documentos de tesis

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **C. Ing. Luis Bryan Domínguez Barrera**, con número de control **M17CE054** de la Maestría en Ciencias en Ingeniería Electrónica, le informamos que hemos revisado el trabajo de tesis profesional titulado **"Control de iluminación mediante conectividad multiprotocolo inalámbrica enfocado a recintos semiabiertos, utilizando tecnologías digitales de acceso libre, seguras y de bajo costo"** y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Jaime Eugenio Arau Roffiel
Doctor en Ciencias en Ingeniería Electrónica
Cédula profesional 9684229

CODIRECTOR DE TESIS

Dr. Gabriel Calzada Lara
Doctor en Ingeniería Eléctrica
Cédula profesional 7599980

REVISOR 1

Dr. Mario Ponce Silva
Doctor en Ciencias en Ingeniería Electrónica
Cédula profesional 3516427

REVISOR 2

Dr. Luis Gerardo Vela Valdés
Doctor en Ciencias en Ingeniería Electrónica
Cédula profesional 7980044

C.p.: M.E. Guadalupe Garrido Rivera /Jefa del Departamento de Servicios Escolares.
Dr. Hugo Estrada Esquivel/ Jefe del Departamento de Ingeniería Electrónica.
Expediente
HEE/lrr.

Cuernavaca, Mor., 28/abril/2021
No. de Oficio: SAC/61/2021
Asunto: Autorización de impresión de tesis

LUIS BRYAN DOMÍNGUEZ BARRERA
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS
EN INGENIERÍA ELECTRÓNICA
P R E S E N T E

Por este conducto tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Control de iluminación mediante conectividad multiprotocolo inalámbrica enfocado a recintos semiabiertos, utilizando tecnologías digitales de acceso libre, seguras y de bajo costo"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
"Excelencia en Educación Tecnológica"
"Educación Tecnológica al Servicio de México"

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO SUBDIRECCIÓN ACADÉMICA

C.c.p. M.E. Guadalupe Garrido Rivera. Jefa del Departamento de Servicios Escolares
Expediente
CMAZ/CHG

Dedicatoria

La presente tesis tiene dedicatoria a mis papás, a mis hermanas y a mi esposa por la paciencia y el apoyo que tuve por parte de ellos al iniciar mis estudios superiores.

A mi madre por su apoyo incondicional y comprensión de las situaciones que atravesamos, por sus consejos y por su forma de ver las cosas porque sé que siempre estará ahí.

A mi padre por sus consejos y por compartir su sabiduría, por la constante presión en que me convierta en una mejor persona, así como guiarme en el camino que debo seguir y ayudarme a no perderme entre tantas decisiones por tomar.

A mi esposa, por ser pieza clave de mi desarrollo profesional, por el apoyo y tutorías en temas que desconocía, así como de ser la persona que día con día me motiva a ser mejor, a terminar las cosas que empiezo y a darle sentido a todo por lo que uno trabaja.

A mis hermanas por servir como apoyo en mis actividades escolares y extracurriculares, por sus palabras de aliento, de ánimo y sus consejos.

A todos, gracias totales.

Agradecimientos

A Dios por haberme permitido concluir mis estudios superiores, así como facilitarme las cosas para que, a lo largo de mi estancia en la maestría, las piezas se acomodaran para poder finalizar el proyecto que se me encomendó.

A mis padres por el apoyo moral y económico que aun en esta etapa de la vida me siguieron brindando sin objeción alguna. Por la comprensión con respecto al tiempo invertido en el tema de tesis prescindiendo de actividades familiares.

A mis hermanas por apoyarme en grandes decisiones, por poder contar con ellas cuando algo estaba mal o cuando mi ánimo decaía, por ser confidentes y apoyarme en decisiones cuando, en ocasiones nadie más sabia, muchas gracias.

A mi esposa por siempre me animo a terminar, a no perder el tiempo, a continuar, perseverar y alcanzar la meta que al inicio me propuse. Por ayudarme a superar tiempos difíciles en los cuales, sin su apoyo, hubiera desistido de lograr este grado. Por ayudarme en temas que no comprendía y por asesorarme en otros que apenas dominaba. Por apoyarme tanto emocional como monetariamente, de corazón gracias.

Al doctor Jaime Arau por haberme aceptado en el tema de tesis, por buscar alternativas para continuar con el proyecto, por aportar ideas y por las enseñanzas que me brindo tanto dentro como fuera del aula. También por buscar hacer las cosas bien, por las revisiones que se tomó el tiempo de hacer, así como los comentarios resultantes de ello, gracias.

Al doctor Gabriel Calzada por tomar el puesto de codirector de tesis, por siempre apoyar el proyecto, y aportar ideas para el desarrollo del proyecto, así como por brindar las herramientas necesarias para llegar más lejos en el proyecto.

A mis revisores, el doctor Mario Ponce y al doctor Gerardo Vela por encaminarme a hacer un trabajo acorde al nivel al cual se está aspirando, a las clases que me impartieron, así como los ratos de convivencia y consejos recibidos de ellos.

A los compañeros de generación, tanto de potencia como de control con quienes se logró tener un ambiente muy agradable para trabajar, así como para pasar un buen rato. Por todas las costumbres y formas propias de cada región que cada uno compartió durante la estancia en CENIDET.

A la institución por proporcionar los medios para desarrollar el proyecto de investigación, así como al personal que hizo de esta estancia un agradable y grato recuerdo.

Al Consejo Nacional de Ciencia y Tecnología por la beca otorgada durante dos años que sirvieron para sustento durante el periodo de desarrollo de tesis.

Resumen

El desarrollo del internet de las cosas ha permitido la integración de tarjetas electrónicas de bajo costo en un sinfín de aplicaciones, entre ellas la iluminación. Continuamente los sistemas se vuelven más eficientes, buscando aprovechar de la mejor manera la energía consumida y conseguir un ahorro energético. Un área de oportunidad son los entornos semi abiertos en los que se puede aprovechar la iluminación natural incidente en el recinto, para ahorrar energía y mantener la iluminación óptima de un recinto.

El presente trabajo muestra la implementación de un sistema que controla la iluminación de una luminaria LED diseñado para edificios con entornos semiabiertos. La lámpara ajusta el nivel de iluminación en función de la cantidad de luz natural que incida sobre el recinto. En caso de que la luz natural sea suficiente para iluminar el recinto, la lámpara permanecerá apagada y se encenderá gradualmente a medida que la iluminación natural disminuya, hasta encender completamente.

La cantidad de luz es medida por un circuito que integra un sensor de luz de día conectado a un microcontrolador. La información que el microcontrolador recibe es transmitida a un módulo central mediante comunicación Bluetooth.

El módulo principal consiste en una tarjeta embebida la cual cuenta con tecnología Bluetooth y WiFi. Dicha tarjeta genera una señal PWM para controlar la lámpara la cual varía en función de la información recibida mediante Bluetooth, así como de los parámetros de ajuste provenientes de la plataforma WEB, transmitidos por WiFi.

El sistema cuenta con un sitio WEB que actúa como interfaz de usuario y se utiliza para monitorear la iluminación que existe en el recinto, así como para determinar el nivel de iluminación que se desea en el mismo. Se puede ajustar el sistema para que regule la iluminación en función de estándares de iluminación o de las necesidades personales del usuario mediante la interfaz programada también en tecnologías Web.

El desarrollo del prototipo se logró utilizando tarjetas de desarrollo y módulos de comunicación libres y de bajo costo en conjunto con programación en diferentes leguajes, como MicroPython, C, HTML, etc. El desarrollo de una plataforma de comunicación propia permite agregar seguridad al sistema al no utilizar código de terceros o rutas de comunicación desconocidas.

La propuesta de solución se validó utilizando un solo punto de control de iluminación, pero es perfectamente extrapolable a un sistema con múltiples puntos de control, el sistema de comunicación se desarrolló pensando en el trabajo simultaneo de múltiples circuitos.

Abstract

The development of the Internet of Things has allowed the integration of low-cost electronic cards in many applications, including lighting. Systems continually become more efficient, searching to make the best use of the energy consumed and achieve energy savings. One area of opportunity is semi-open environments in which the natural lighting incident in the venue can be used, to save energy and maintain the optimal lighting of a venue.

This work shows the implementation of a system that controls the lighting of an LED luminaire designed for buildings with semi-open environments. The lamp adjusts the lighting level according to the amount of natural light that exist on the room. In case the natural light is sufficient to illuminate the room, the lamp will remain off and will turn on gradually as the natural light diminishes, until it lights up completely.

The amount of light is measured by a circuit that integrates a daylight sensor connected to a microcontroller. The information that the microcontroller receives is transmitted to a central module through Bluetooth communication.

The main module consists of an embedded card which has Bluetooth and WiFi technology. This card generates a PWM signal to control the lamp, which varies depending on the information received with Bluetooth protocol, as well as the adjustment parameters from the WEB platform, transmitted through WiFi.

The system has a WEB site that acts as a user interface and is used to monitor the lighting that exists in the room as well as to determine the level of lighting that is desired in it. The system can be adjusted to regulate lighting according to lighting standards or the user's own needs through the interface programmed also in Web technologies.

The development of the prototype was achieved using low cost development cards and communication modules in conjunction with programming in different languages, such as MicroPython, C, HTML, etc. The development of a proprietary communication platform allows adding security to the system by not using third-party code or unknown communication routes.

The solution proposal was validated using a single lighting control point, but it is perfectly extrapolated to a system with multiple control points, the communication system was developed thinking about the simultaneous work of multiple circuits.

Contenido

Índice de Figuras

Índice de tablas

Capítulo I

Introducción

1.1 Antecedentes

El ahorro energético es un tema que se ha vuelto de suma importancia debido al deterioro ambiental que se presenta actualmente por el uso desmedido de combustibles no renovables. Dicha cuestión ha motivado a los fabricantes a crear dispositivos que sean cada vez más eficientes logrando dispositivos más sofisticados y compactos.

La iluminación artificial es uno de los temas que más se han desarrollado en los últimos años y gracias a ella se pueden realizar actividades sin depender solo de la luz del día. Es por ese motivo que el tema de iluminación ha avanzado a pasos agigantados, evolucionando desde una ampolleta que iluminaba como consecuencia de llevar a un filamento a un punto de incandescencia, a dispositivos de estado sólido que utilizan el efecto fotovoltaico para generar luz, un ejemplo de ello es la tecnología LED.

Estas nuevas tecnologías permiten aprovechar mejor la energía eléctrica que consumen al alcanzar eficiencias superiores al 90 por ciento lo que permite tener dispositivos más potentes con el mismo consumo que tecnologías anteriores. Si bien la tecnología de iluminación contribuye en gran manera al ahorro energético no es el único medio para lograrlo.

Existen algunos trabajos para generar dispositivos que sean capaces de utilizar de la mejor manera la energía eléctrica. En el entorno de iluminación una estrategia que se ha implementado en diferentes lugares es añadir al sistema sensores de movimiento, de presencia o de luz ambiental con el objetivo de encender las luminarias cuando realmente se necesiten. Estas acciones se traducen en un ahorro energético debido a que la iluminación no se mantiene encendida de manera permanente, por lo que se mejora el aprovechamiento de la energía apagando o encendiendo la luminaria dependiendo si se cumplen o no ciertas condiciones.

Este tipo de sistemas son instalados en entornos cerrados como oficinas, cubículos, salones, entre otros lugares. Algunos de estos sitios tienen acceso a iluminación natural y se les conoce como entornos semiabiertos, en dichos sitios se aprovecha la iluminación natural a lo largo del día En estos recintos el nivel de iluminación natural generalmente es suficiente para

realizar actividades cotidianas, de acuerdo a estándares de iluminación, la mayor parte del día [5]–[7].

Algunos fabricantes implementan dispositivos para realizar un control de la iluminación mediante circuitos especialmente diseñados para ello, como el protocolo DALI. Este protocolo es cableado y se encuentra ampliamente trabajado por lo que cuenta con mucha documentación disponible. Una de las complicaciones que tiene es durante la implementación debido a que para poder instalarlo se debe colocar cableado especial lo que complica instalarlo en inmuebles ya construidos, sin embargo, resulta atractivo en inmuebles nuevos. Para inmuebles ya construidos lo ideal es modificar lo menos posible lo que ya se encuentra instalado por lo que las alternativas inalámbricas se vuelven una alternativa más interesante.

La comunicación inalámbrica es una alternativa cada vez más viable debido a que los módulos de comunicación cada vez son más eficientes y potentes tanto en hardware como en software. Los módulos de comunicación cada vez consumen menos potencia pensando en funcionar con una batería, además de aumentar las capacidades e instrucciones para realizar mayor número de aplicaciones o tareas.

La mayoría de sistemas de potencia actuales se diseñaron para funcionar de manera local en conjunto con sistemas de control los cuales adquieren datos de la planta y la procesan en el mismo sitio. La comunicación y monitoreo de las plantas se logra transmitiendo los datos por protocolos cableados o protocolos inalámbricos de corto alcance.

1.2 Planteamiento del problema

A pesar del avance tecnológico aún son pocos los sistemas que trabajan en conjunto con sistemas de potencia y con herramientas de comunicación digital con el fin de integrarlos a entornos como redes inteligentes o micro redes en CD. Estos sistemas tienen como objetivo gestionar de una mejor manera la energía y tener un mejor control de los dispositivos interconectados.

A pesar que existen trabajos para incluir circuitos digitales en luminarias, muchos de ellos se basan solo en un control de encendido/apagado. De esta manera se tiene cierto control sobre las lámparas de un recinto ahorrando energía, sin embargo, no se tiene control sobre la cantidad de iluminación que ofrece la lámpara ni se puede asegurar una iluminación ideal del área de cobertura en el recinto. Utilizando esta tecnología existen equipos que comercialmente ofrecen control de diferentes dispositivos a través de voz o de comandos a través del celular, no obstante, lo que realmente realizan es activar o desactivar un contacto conectado en serie con la carga a controlar.

Los equipos comerciales que se encuentran actualmente en el mercado solucionan en parte la problemática de controlar dispositivos a distancia. El problema que se presenta con estas

alternativas es la información que se requiere utilizar del usuario, así como el flujo de información de la misma. Estos datos se transmiten mediante distintas etapas mismas que el usuario desconoce además de ignorar quien tiene acceso a los datos que se transmiten vulnerando la seguridad del sistema.

Existe otra limitación con respecto a las soluciones comerciales actuales puesto que si el usuario desea automatizar un inmueble se ve obligado a utilizar una sola marca. Esto resulta un problema debido a los costos de añadir circuitos a los dispositivos a controlar, resultando en un elevado costo de instalación solo para añadir la capacidad de encender o apagar dispositivos.

En entornos semiabiertos la problemática es aún mayor porque a pesar que se cuenta con iluminación natural en el recinto la iluminación no se encuentra preparada para aprovecharla. Cuando disminuye la iluminación natural, la iluminación artificial se enciende, a veces, en ocasiones donde no es necesaria del todo, de esta manera la iluminación que aún ingresa no se aprovecha ya que no se puede encender parcialmente las luminarias. Por estas razones utilizar un control estilo *on/off* no es la mejor elección ya que no se controla el nivel de iluminación de la lámpara provocando que encienda a su máximo nivel todo el tiempo o no se tenga una iluminación óptima al no encenderlas por no necesitarlas del todo.

La falta de iluminación en un recinto puede ser un problema de seguridad y de salud ya que puede afectar el desarrollo de las actividades de los trabajadores. Cada espacio debe tener un nivel óptimo de iluminación para asegurar el bienestar y el correcto desempeño de las actividades en los diferentes espacios de trabajo [1], [2].

1.3 Objetivos

A continuación, se presentan los objetivos generales y específicos considerados para elaborar el presente trabajo de investigación.

1.3.1 Objetivo General

Diseñar un sistema que permita mantener constante la iluminación a lo largo del día en recintos semiabiertos utilizando conectividad inalámbrica multiprotocolo con tarjetas de libre acceso y bajo costo.

1.3.2 Objetivos Específicos

a) Validar el diseño de una interfaz web funcionando sobre internet, que permita la comunicación entre el usuario y el módulo WiFi, que asegure el intercambio de información para un monitoreo constante desde el navegador web tanto del sistema como de las tarjetas enlazadas inalámbricamente.
b) Validar las etapas de medición y comunicación del sistema, con un convertidor CD/CD que alimente una lámpara de LEDs que actualmente se utilice en el

Departamento de Ingeniería Electrónica del CENIDET, para ajustar el nivel de iluminación de una determinada área, a partir de la cantidad de luz ambiental medida por un circuito intercomunicado a través de Bluetooth.

c) Demostrar la posibilidad de modificar el firmware de dispositivos digitales de comunicación y de tarjetas de desarrollo que se utilizarán en el prototipo para poder programar todos los elementos de la tarjeta sin incluir componentes adicionales, como microcontroladores.

1.4 Alcances y limitaciones

1.4.1 Alcances

Los alcances que tendrá la presente tesis se presentan a continuación:

a) Implementar un convertidor CD/CD tipo Buck con una potencia de 30W y un voltaje de salida de 60V, el cual será la fuente de alimentación para la lámpara LED de 30W.
b) Diseñar e implementar un circuito que mida la cantidad de luz ambiental y lo transmita inalámbricamente al convertidor CD/CD por medio de Bluetooth para lograr un ajuste de la iluminación en función de la luz presente y el valor establecido por el usuario.
c) Diseñar una interfaz gráfica WEB que sea capaz de recolectar los parámetros obtenidos de los sensores en el sistema, así como de enviar instrucciones sobre el nivel de atenuación con el que trabajará la lámpara.

1.4.2 Limitaciones

a) Se utilizarán herramientas básicas para el desarrollo de la interfaz web.
b) Se cuenta con un número limitado de tarjetas por lo que no se podrá hacer prueba a pleno rendimiento de la plataforma en línea y solo se validará el principio de funcionamiento.
c) Se cuenta con solo dos lámparas de prueba, pero no un lugar físico para realizar el banco de pruebas, por lo que se emulará el comportamiento en un entorno controlado.

1.5 Organización del documento de tesis

Capítulo 1. Introducción. - En el capítulo uno se describe el contexto en el que se desarrolla el proyecto, así como la razón para realizarlo, se describe la problemática que se está atacando y los objetivos que tiene el proyecto.

Capítulo 2. Marco Teórico, revisión del uso de la tecnología y diseño conceptual. - En este capítulo se hace un breve repaso de los elementos y dispositivos tecnológicos que se requerirán para la realización del sistema tales como: tarjetas digitales, microcontroladores, lenguajes de programación, bases de datos, internet de las cosas, comunicación Bluetooth,

comunicación wifi, entre otras cosas. Se presentan también los cálculos necesarios, consideraciones de acuerdo a las hojas de datos de los fabricantes y pruebas a realizar a los componentes que integrarán el proyecto.

Capítulo 3. Pruebas y resultados. - Se describe el banco de prueba que se implementó para obtener los resultados, así como las pruebas realizadas y las consideraciones que se tomaron. Se muestra el sistema operando bajo las condiciones de validación que se acotaron.

Capítulo 4. Conclusiones y trabajos futuros. - Durante el capítulo 4 se presentan los resultados una vez realizadas las pruebas y las conclusiones a las que se llegó al término del proyecto de investigación. De igual manera se mencionan las actividades y trabajos que pudieran dar continuidad al trabajo realizado en esta tesis.

Capítulo II

Marco Teórico, revisión del uso de la tecnología y diseño conceptual

En el presente capítulo se muestran conceptos útiles para la comprensión del trabajo de investigación además de trabajos similares que comparten tecnología o que buscan resolver un problema común desde otra perspectiva. Se presenta el desarrollo de las diferentes tarjetas de comunicación que integran al proyecto y las características que tienen. Se incluye también el diseño del convertidor, del sistema web, los diferentes circuitos que integran el sistema y tanto los diagramas esquemáticos como los dispositivos que integran cada circuito impreso.

2.1 Fuentes conmutadas

Este tipo de fuentes utilizan una manera diferente de regular la tensión de salida empleando únicamente dispositivos de conmutación en conjunto con inductores y capacitores (utilizados en el proceso de almacenamiento y trasferencia de energía). A diferencia de las fuentes lineales, las fuentes conmutadas evitan el uso de resistores para no disipar energía en forma de calor. Al utilizar solo compontes reactivos, se logra que el circuito sea más eficiente y más compacto ya que suelen operar en altas frecuencias.

Existen muchas topologías de convertidores conocidas como *Buck, Boost, Cuk, Zeta y Flyback* entre otros [5]. Algunos de ellos son equivalentes entre sí, sin embargo, cada uno de los circuitos presentan características propias, por lo que para cada aplicación se debe hacer una correcta selección del convertidor a utilizar.

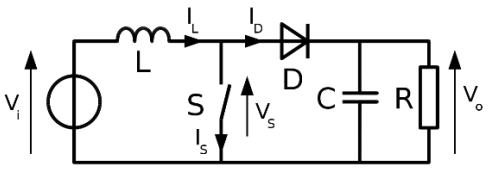


Figura 2. 1.- Fuente conmutada en topología elevadora (también conocida como Boost).

Este tipo de fuentes trabajan en dos estados de operación los cuales dependen del estado del transistor y comúnmente se conocen como estado de encendido (ON) y estado de apagado (OFF). El objetivo es mandar un ancho de pulso, el cual se calcula en función del voltaje de salida y del voltaje de entrada, para lograr un voltaje promedio en la salida que se desea entregar a la carga.

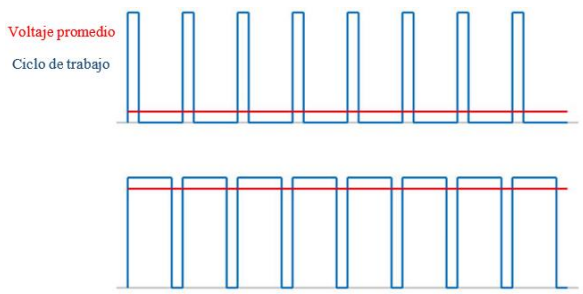


Figura 2. 2.-Voltaje de salida promedio con respecto al ciclo de trabajo.

Las fuentes conmutadas tienen mejores características que las fuentes lineales ya que pueden diseñarse para potencias más elevadas además de contar con la ventaja de trabajar en alta frecuencia lo que conlleva a que el tamaño resultante sea más compacto que su contraparte lineal de la misma capacidad. Desde luego otra ventaja importante es la eficiencia que pueden alcanzar las fuentes de alimentación conmutadas.

Dependiendo de la relación de entrada y salida, se pueden utilizar topologías específicas que se mencionan a continuación de manera breve, profundizando solamente en el caso del convertidor reductor o Buck que se implementará en el presente trabajo de investigación.

2.1.1 Convertidor reductor o Buck

El convertidor *Buck* es una topología de fuente conmutada cuya función es reducir el voltaje de salida con respecto al de su entrada. La topología se basa en utilizar un transistor en serie y un diodo en paralelo a la fuente, estos dispositivos son los responsables de conmutar y presentar los dos modos de operación.

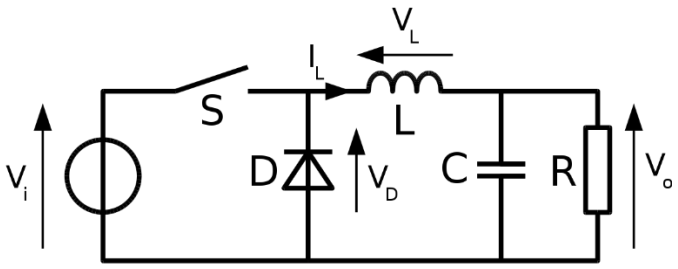


Figura 2. 3.- Fuente conmutada reductora tipo Buck.

El primer modo de operación es cuando el circuito se encuentra en estado *ON*, es decir cuando el transistor se encuentra en saturación. En este estado se conecta la fuente de alimentación directamente a la bobina y en paralelo a la carga y al capacitor. El diodo se encuentra polarizado en inversa, de manera que no conduce, actuando como circuito abierto. El circuito equivalente resultante se muestra en la Figura 2. 4.

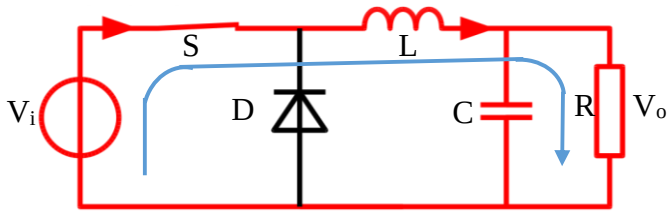


Figura 2. 4.- Convertidor Buck en estado ON.

El convertidor entra en estado off en cuanto el transistor entra en la región de corte. En dicho estado la fuente se desconecta del circuito restante, la polaridad cambia provocando que el diodo entre en conducción, cerrando la malla de salida tal cual se muestra en la Figura 2. 5.

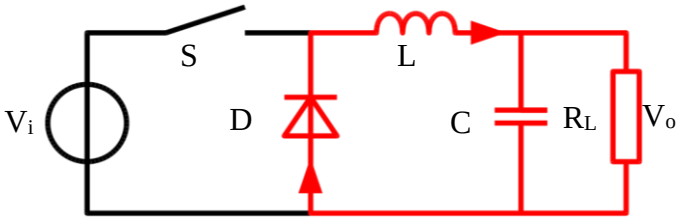


Figura 2. 5.- Convertidor Buck en estado off.

Las expresiones para el diseño del convertidor Buck se presentan con detalle mas adelante.

2.2 Módulos *WiFi*

Uno de los módulos que más avances y aplicaciones ha tenido es el ESP8266 quien trabaja con tecnología *WiFi* y es desarrollado por la compañía *Esspresif* [8]. Este chip cuenta con los protocolos necesarios para comunicarse mediante *WiFi*. El módulo *WiFi* se comunica originalmente con comandos AT y está pensado para trabajar en conjunto con un microcontrolador.

Los módulos son muy fáciles de utilizar con Arduino ya que se crearon librerías especializadas para trabajar con él y hay mucha documentación en internet con ejemplos y resolución de problemas. Su uso extendido puede atribuirse a la versatilidad que tiene ya que es capaz de trabajar en dos modos de operación simultáneamente de ser necesario. La primera configuración del chip permite que trabaje como estación, es decir, genera una señal *WiFi* con su propio nombre (*SSID*) y cuya contraseña es opcional habilitar. A este modo de operación se le llama AP (*Access Point*). La segunda configuración del módulo es como cliente, es decir, conectarlo a un modem proporcionándole la *SSID* de un *modem* y su contraseña tal cual lo haría un dispositivo inteligente cualquiera, de esta manera es capaz de responder a comunicaciones de manera local al poder entablar comunicación con dispositivos que se encuentran conectados al mismo *modem*, a este modo se le conoce como STA (*Station*).

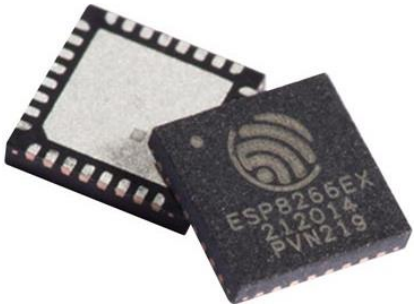


Figura 2. 6.- Chip de comunicación WiFi ESP8266.

2.2.1 Tarjeta ESP-01

La tarjeta ESP01 integra el chip de comunicación ESP8266 junto con memorias, un microcontrolador, osciladores y antenas. Esta tarjeta se popularizó ya que se podía utilizar en conjunto con Arduino para establecer comunicaciones *WiFi*. El Arduino contiene la programación para ejecutar cierta acción mientras que el módulo ESP8266 en la tarjeta ESP-01 sirve únicamente como puente para establecer la comunicación a través de *WiFi*.

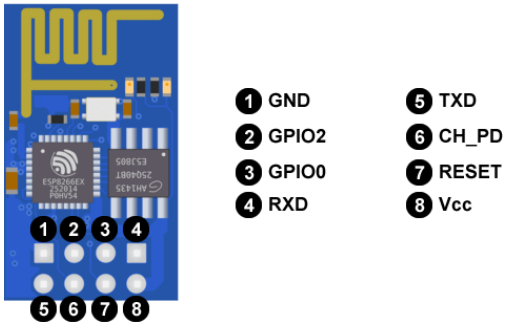


Figura 2. 7.- Tarjeta ESP-01 y su disposición de pines.

La comunicación entre el microcontrolador y la tarjeta de comunicación *WiFi* se realiza de manera serial utilizando el estándar UART incluido en ambos elementos. Esto resulta ser muy sencillo ya que el módulo de internet solo necesita recibir comandos AT específicos para realizar la acción necesaria, de esta manera el microcontrolador envía de manera serial un comando dependiendo de lo que se requiera realizar y el módulo lo ejecuta.

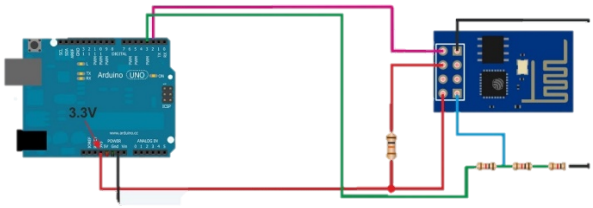


Figura 2. 8.- Conexión entre Arduino y la tarjeta ESP-01.

Las características de la tarjeta ESP-01 son:

- Microcontrolador de 32 bits
- Módulo con instrucciones TCP/IP
- Reguladores de voltaje, PLL
- Comunicación *WiFi* del estándar 2.4GHz
- Modo estación y modo cliente
- Comunicación serial: SPI, UART, I2C
- Pines GPIO, PWM
- Antena con 20dBm de ganancia
- Memoria *Flash* de 1MB

Al utilizar la comunicación serial UART para realizar la comunicación entre un microcontrolador y la tarjeta ESP-01 únicamente se necesitan dos pines para realizar la comunicación bidireccional entre los dispositivos.

2.2.2 Tarjeta ESP-12

Con el tiempo las tarjetas se fueron desarrollando hasta llegar al modelo ESP-12 el cual mejora muchos aspectos con respecto a la ESP-01. Entre los más destacables es la integración del convertidor analógico digital, un procesador más potente, mayor número de pines digitales de propósito general y una memoria *flash* con mayor capacidad para almacenar programas. El encapsulado de la tarjeta ESP-12 es de montaje superficial con pines alrededor siendo de un tamaño un poco mayor al ESP-01.

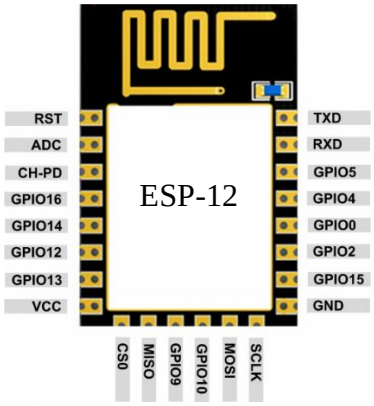


Figura 2. 9.- Encapsulado de la tarjeta ESP-12 y la disposición de sus pines.

Al igual que su antecesora, esta tarjeta tiene precargado un firmware que lo habilita para trabajar con comandos AT, sin embargo, debido a sus características, es posible modificar ese firmware para que se pueda programar mediante otros lenguajes.

Es importante mencionar que, para cualquier tarjeta basada en el chip esp8266, existen dos modos de operación, el primero se utiliza para guardar programas en la memoria *flash* mientras que el otro ejecuta el programa que tenga guardado en la memoria *flash*. Para habilitar un modo u otro se debe conectar los pines como se muestra en la Tabla 2. 1.

Tabla 2. 1.- Modos de operación del chip del esp8266.

GPIO15	GPIO0	GPIO2	Modo
0V	0V	3.3V	UART Bootloader
0V	3.3V	3.3V	Ejecución de programa
3.3V	x	x	Modo STDIO

Este módulo *WiFi* ha servido de base para la creación de tarjetas de desarrollo con comunicación inalámbrica. Diferentes empresas han colocado la tarjeta ESP-12 en conjunto con reguladores de voltaje, convertidores TTL-USB y demás periféricos para hacer aún más compatible el módulo con diferentes tecnologías.

2.2.3 Tarjeta de desarrollo Feather Huzza ESP8266

La tarjeta Feather Huzza es un elemento de desarrollo de la empresa Adafruit la cual integra a su vez la tarjeta ESP-12. Cuenta con un microcontrolador a una velocidad de reloj de 80MHz un circuito para carga de batería LiPo además del convertidor TTL-USB CP2104 de SiLabs el cual facilita la comunicación con una computadora al emular un puerto COM tal como lo hace el Arduino.

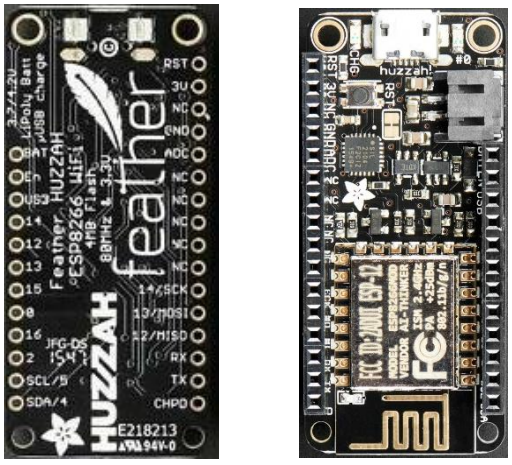


Figura 2. 10.- Tarjeta Feather Huzza de la compañía Adafruit.

Este chip está pensado para una programación más fácil ya que puede programarse con el mismo entorno de Arduino utilizando programación en el lenguaje C++. Por otro lado, al ser una tarjeta libre también existen otras alternativas ya que puede cargarse versiones ligeras del interprete Python 3 tal como CircuitPython o MicroPython.

Esta tarjeta también incluye los circuitos mínimos para entrar en modo de programación y en ejecución de programa. Al encender la tarjeta automáticamente entrará en modo de ejecución de programa por lo que si se desea entrar en modo de carga de programa se debe presionar el botón de *bootloader* junto con el de *reset* integrados en la tarjeta.

La tarjeta cuenta con una entrada micro USB trabajando a 5V, adicionalmente, cuenta con reguladores a 3.3V debido a que algunos módulos internos operan a dicha tensión, entre ellos el chip ESP8266.

2.3 Módulos *Bluetooth* HC05 y HC06

El circuito más popular por sus prestaciones y su precio es el HC05 el cual es un módulo *Bluetooth* que funciona a través de comunicación serial UART mediante comandos AT. Los comandos AT consisten en una serie de palabras clave que tienen un fin específico. Estas palabras se colocan después de las letras AT+ y de acuerdo a lo que se coloque será la acción que realizará el comando, por ejemplo, *AT +NAME,* el cual retornará el nombré del dispositivo.

El módulo maestro es el que inicia la conexión con el módulo esclavo [11]. Existe un módulo esclavo con número de parte HC06 que sirve únicamente como receptor y se empareja con el módulo maestro. La diferencia entre ambos módulos radica en el *firmware* que se carga en la memoria *flash* y es el responsable de su funcionamiento además de dos pines adicionales con los que cuenta el módulo HC05. Debido a la similitud entre ambos circuitos es posible convertir un HC06 en un HC05 y viceversa cargando el respectivo programa en su memoria *flash* y añadiendo algunos circuitos.

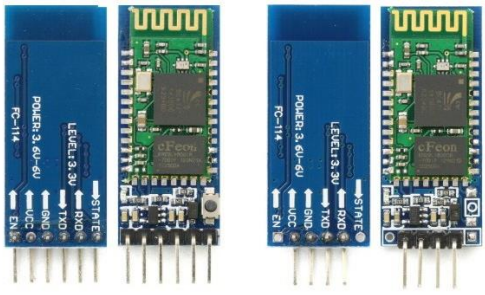


Figura 2. 11.- Módulos Bluetooth HC-05 y HC-06.

Para poder comunicarse con el módulo *Bluetooth* se deben mandar comandos AT específicos. La tarjeta contestará un *OK* en caso de que la instrucción fue escrita y procesada correctamente. Los comandos más importantes se muestran en la siguiente tabla.

Tabla 2. 2.- Comandos AT básicos de configuración de los módulos HC05-06.

Sintaxis del comando	Función
AT	Comando de prueba, debe responder con un *OK*.
AT+ROLE=1	Configuración del módulo en modo Maestro.
AT+ROLE=0	Configuración del módulo en modo Esclavo.
AT+VERSION?	Comando para obtener la versión del *firmware*.
AT+UART=115200,1,2	Comando de configuración del puerto serial.
AT+PIO=10,1	Comando para colocar el pin IO de propósito general en alto.
AT+BAUD<Numero>	Configuración de la velocidad de comunicación.
AT+NAME<Nombre>	Configuración del nombre del dispositivo *Bluetooth*.
AT+PIN<pin de 4 dígitos>	Configuración del pin de emparejamiento.
AT+ADDR?	Comando para obtener la dirección MAC del módulo BT
AT+BIND	Comando para iniciar la conexión con otro módulo _BT

Es importante mencionar que los módulos *Bluetooth* tienen diferentes modos de operación independientemente si son maestros o esclavos los cuales se describen a continuación.

Cuando se energiza el módulo y aun no se ha configurado o no se ha emparejado con otro dispositivo entrara en el primer modo que es *desconectado*. Dentro de este modo el led incluido en la tarjeta empezará a parpadear rápidamente. Es importante mencionar que el módulo HC05 en este modo es capaz de interpretar comandos AT provenientes del pin RX a diferencia del HC06.

Existen dos modos en donde los módulos interpretan comandos AT, para acceder a ellos es necesario interactuar con el botón pulsador integrado en las tarjetas. Para ingresar al modo *AT 1* se debe presionar el botón una vez energizado el circuito, inmediatamente la tarjeta comenzará a interpretar comandos AT a la velocidad en baudios que se le configuró. El modo *AT 2* permite también ejecutar comandos AT, pero a una velocidad fija de 38400 baudios, este modo es útil cuando por alguna razón desconocemos la velocidad de transmisión de la tarjeta. Para entrar al modo AT2 se debe mantener presionado el botón antes de energizar la tarjeta y 5 segundos posterior al encendido del módulo.

Cuando se establece una conexión se accederá al modo *conectado*, el módulo el led realizara un parpadeo doble y transmitirá por *Bluetooth* todo lo que reciba a través del puerto serial. En este modo ningún módulo interpreta comandos AT, por lo que la configuración se deberá realizar en otro modo de operación.

2.4 Núcleo STM32L476RG

La compañía *STM Microelectronics* es una empresa dedicada al desarrollo de componentes electrónicos, tarjetas digitales de desarrollo y cuenta con diferentes opciones de microcontroladores, así como de herramientas de software para programarlos. *STM Microelectronics* cuenta con familias de microcontroladores entre las que se encuentra la serie *STM32476xx*. Los dispositivos pertenecientes a esta familia son capaces de ejecutar instrucciones RISC de 32 bits con una frecuencia de operación de al menos 80MHz además

de ser de bajo consumo. Estas características permiten ocupar estas tarjetas de desarrollo en aplicaciones en donde se demanden muchos recursos computacionales ya que, incluso son capaces de conectarse con *software* como simulink y ejecutar programas enlazados a través de USB o descargar un programa y ejecutarlo desde la memoria.

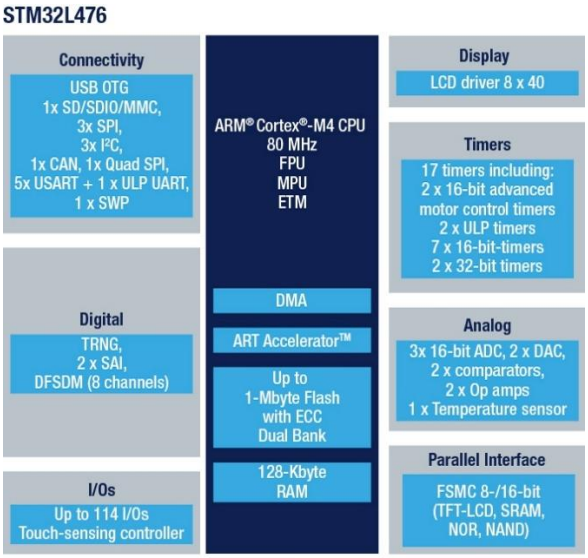


Figura 2. 12.- Características de las tarjetas de la seria STM32L472.

Otra de las características de las tarjetas STM es que son compatibles con los *Shield* de Arduino debido a la disposición interna de pines hembra con las que cuentan las tarjetas, los cuales corresponden, de acuerdo al modelo, a las versiones de Arduino Nano, Uno y Mega. En el caso de esta versión de tarjeta los pines son compatibles con la versión uno de Arduino y la disposición de pines corresponde exactamente a la de Arduino.

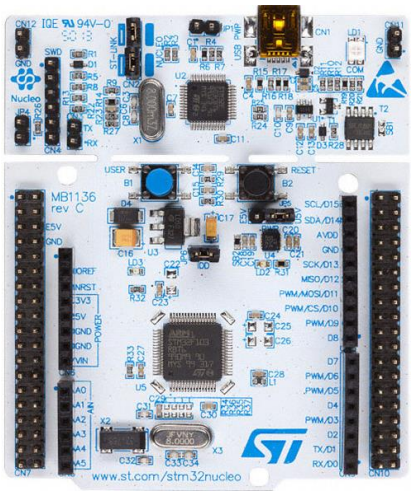


Figura 2. 13.- Disposición de pines de la tarjeta STM32L476RG.

Además de los pines correspondientes a los de Arduino, cuenta con cuatro hileras de pines las cuales tienen funciones diferentes y especializadas además de contar con el procesador ARM Cortex-M4 y con un módulo de punto flotante lo cual, en conjunto, permite ejecutar operaciones complejas.

2.5 Programación en Arduino

El lenguaje con el cual trabaja el IDE de Arduino es un lenguaje de alto nivel el cual toma como bases otros lenguajes como *procesing* y *C++*. Este lenguaje se adaptó para poder comunicarse con los microcontroladores de Atmel basados en el estándar avr-libc.

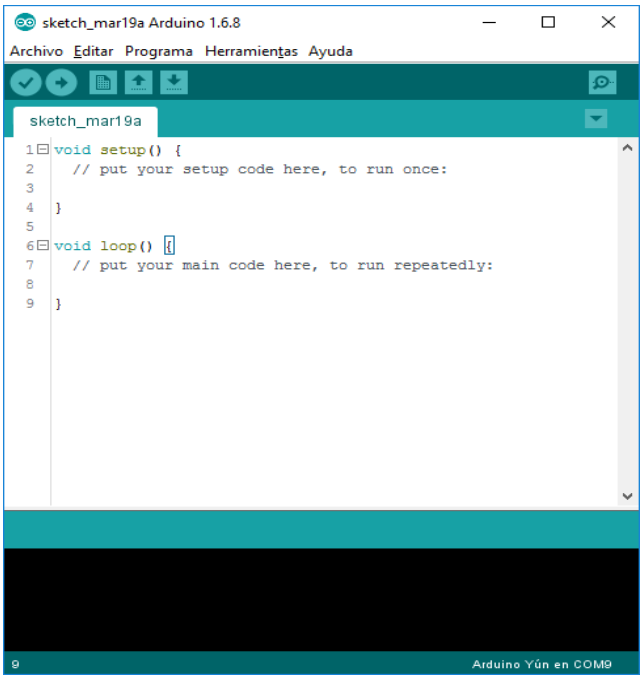


Figura 2. 14.- Entorno de programación de Arduino .

La figura anterior muestra el entorno de desarrollo de Arduino el cual también es conocido como IDE por las siglas en inglés de *Integrated Development Environment*. Desde esta interfaz de usuario es donde se escribe la codificación del programa y se interactúa con las diferentes tarjetas electrónicas que soporta.

El éxito de dicha plataforma se basa en su comunidad, ya que existe mucha información proporcionada por la compañía de Arduino a la par con la información que se encuentra en la red de internet, proporcionada por los usuarios. Gracias al esfuerzo conjunto se han creado librerías para poder reducir la cantidad de código necesario para realizar una tarea específica, es decir, no se debe tener un conocimiento muy profundo de programación para lograr comunicarse con la tarjeta.

La estructura de cualquier programa en Arduino consta básicamente de dos funciones, una de configuración y la otra de ejecución. La función *setup* se encarga de la configuración de

la tarjeta y es la primera función que se ejecuta al energizarse la tarjeta y solo se ejecuta una vez. En esta función se colocan las sentencias iniciales como configuración, definición de pines, configuración de puertos, velocidades de reloj, comunicaciones a habilitar, entre otras cosas. La función *loop* es la función principal en donde se colocará el segmento de código que contiene las rutinas que seguirá el microcontrolador quien las ejecutará secuencialmente y una vez terminada la ejecución iniciará de nuevo entrando en un ciclo infinito el cual deja de ciclarse cuando se desconecta la tarjeta.

El éxito de Arduino fue tan grande que la propia comunidad se ha encargado de hacer modificaciones en el entorno de programación para poder programar otras tarjetas u otros dispositivos desde la misma interfaz de usuario, un ejemplo de ello son las tarjetas basadas en el módulo ESP8266 los cuales se pueden programar desde la IDE de Arduino.

2.6 Programación en Python

Es común encontrar que en diferentes fuentes se le conozca a Python como un lenguaje de programación lo cual no es del todo correcto. Es más preciso llamar a Python un intérprete de alto nivel ya que ejecuta en tiempo real las instrucciones sin necesidad de un compilador. Python es una tecnología utilizada en diferentes entornos y aplicaciones como: bases de datos, servidores, páginas web, entre otras, facilitando la comunicación entre ellas.

El intérprete cuenta con un modo el cual se llama REPL por sus siglas en inglés *Read Evaluate Print Loop*. Dicho modo permite escribir código de Python y obtener resultados inmediatos debido a la característica de ejecutar comandos en tiempo real. Este modo es muy útil para realizar pruebas a las librerías, y realizar depuración de un conjunto de instrucciones línea por línea con el fin de verificar que lo que se programó se está ejecutando como se diseñó.

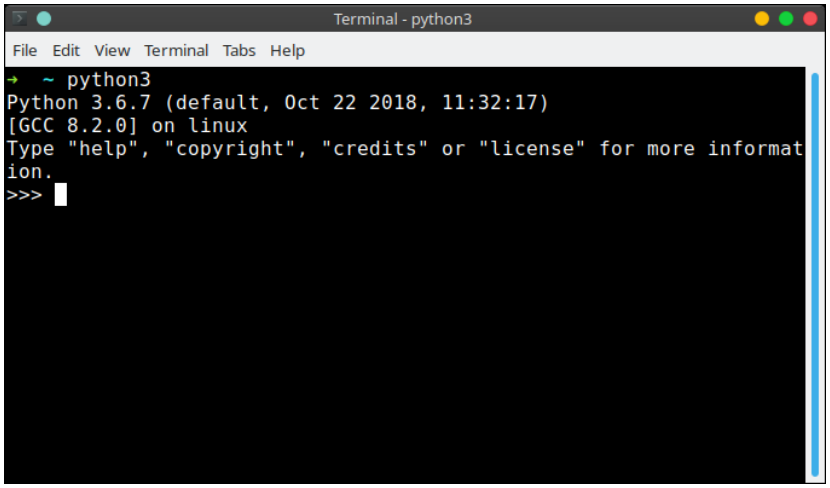


Figura 2. 15.- Terminal REPL presente en Python en todas sus versiones incluyendo MicroPython.

El modo REPL tiene de interfaz una terminal como la que se observa en la figura anterior, se caracteriza por iniciar cada línea de código con tres símbolos de mayor que (>>>) los cuales

indican que se está corriendo el intérprete de Python. En esta terminal se pueden agregar operaciones lógicas y aritméticas y al presionar la tecla *enter* la terminal responderá con el resultado correspondiente. También se pueden declarar librerías desde este modo, ejecutar programas guardados (*scripts*) o ejecutar alguna instrucción la cual se realizará tan pronto se presione la tecla *enter*.

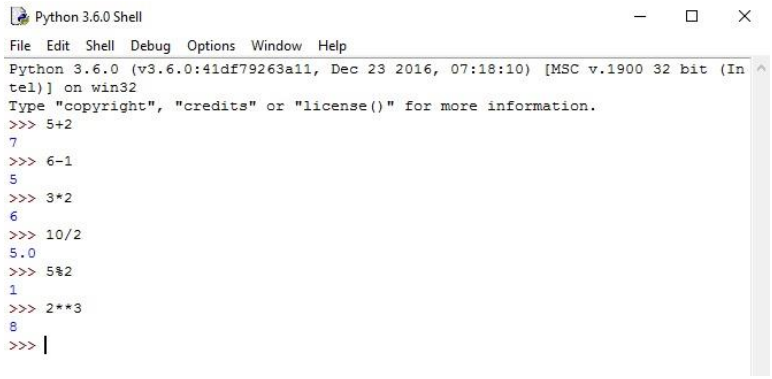


Figura 2. 16.- Operaciones básicas en el modo REPL de Python.

Al ser Python un lenguaje de muy alto nivel cuenta con una sintaxis que es muy fácil de comprender además de no utilizar caracteres especiales. En lugar de agregar llaves para separar bloques de código, el intérprete utiliza identación logrando crear scripts aún más claros y entendibles.

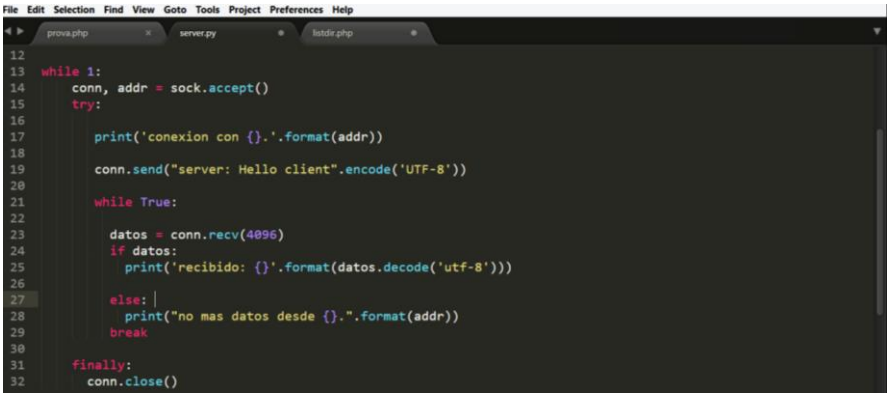


Figura 2. 17.- Ejemplo de identación en un bloque de código en Python.

Existe un *firmware* desarrollado para programar microcontroladores llamado Micro Python el cual es una versión ligera y eficiente del lenguaje de programación Python 3. Dicho firmware contiene la mayoría de las librerías estándar de Python recortadas u optimizadas para ocupar el menor tamaño posible para caber en las reducidas memorias de los microcontroladores[12]. Al igual que otros lenguajes de programación, MicroPython está optimizado para programar hardware, es por ello que además de contar con la mayoría de las librerías presentes en la versión completa de Python agrega otras especializadas en comunicarse con hardware.

Muchas librerías son las mismas en ambas versiones y algunas se modificaron para hacerlas compatibles. Para diferenciar cuando se trabaja en Python o Micro Python las librerías suelen tener la letra *u* antes de la librería, es decir, si en Python se utilizan las librerías *request, json, collections* o *array*, en Micro Python se llamarán: *urequest, ujason, ucollections* y *uarray*, agregando la *u* haciendo alusión al prefijo *micro*.

Esta versión de MicroPython también incluye el modo REPL para realizar pruebas y acciones de depuración en el código. Adicionalmente se puede acceder a este modo a través de internet cuando el *firmware* es cargado en tarjetas de comunicación WiFi activando el modo WebREPL. Para activarlo se utiliza el comando *import webrepl* en la tarjeta digital, para comunicarse se ingresa a la dirección http://micropython.org/webrepl/ y se introduce la dirección IP que se le ha asignado a la tarjeta para de esta manera controlarla inalámbricamente a través de este modo.

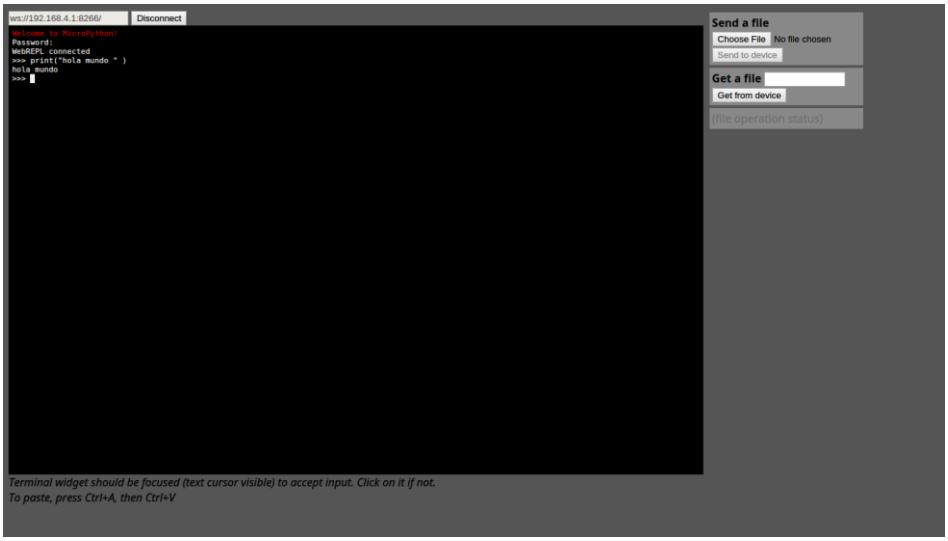


Figura 2. 18.- Modo REPL corriendo a través del navegador y vinculado a un módulo WiFi.

A pesar de que MicroPython se diseñó como una alternativa de programación de tarjetas digitales, existen librerías específicas que diferentes compañías han adaptado al *firmware* original y solo pueden ser utilizadas por ciertas tarjetas por ejemplo Pycom con sus tarjetas *WiPy*. Estas tarjetas son creadas por compañías que al tiempo de crear el *hardware* ofrecen un firmware basado en MicroPython en donde agregan funciones específicas para hardware propio de tarjetas en específico y facilitan el código con librerías propias las cuales solo funcionan con esas tarjetas.

Existen librerías específicas para las tarjetas que están basados en los *chips* ESP8266 y ESP32. Estas librerías pueden manejar funciones propias del chip *WiFi* como: mandar a reposo la tarjeta, realizar reinicios, manejar registros, añadir o eliminar archivos de programación, entre otras cosas.

2.7 Comunicación WEB

La invención del internet en los años 60´s fue un gran avance tecnológico en todo el mundo, ya que se podía acceder a información contenida en una red de computadoras interconectadas mediante nodos de internet desde cualquier punto en donde se tuviera acceso a la red. En la actualidad navegar en la red a través de internet es algo común y a pesar de ello, pocas personas saben el proceso que se lleva a cabo para lograr una comunicación en internet.

La comunicación web es básicamente acceder a archivos guardados en computadoras dedicadas al intercambio de información, llamados servidores. Para ingresar se utilizan programas específicos llamados navegadores *Web*, como lo es *Internet Explorer*, *Google Chrome* o *Mozilla Firefox*, por mencionar algunos. Estos navegadores solicitan información al sitio que se requiere acceder detrás de la interfaz de usuario. El sitio responde con una serie de códigos, usualmente lenguaje HTML, los cuales interpreta el navegador y muestra de manera ordenada.

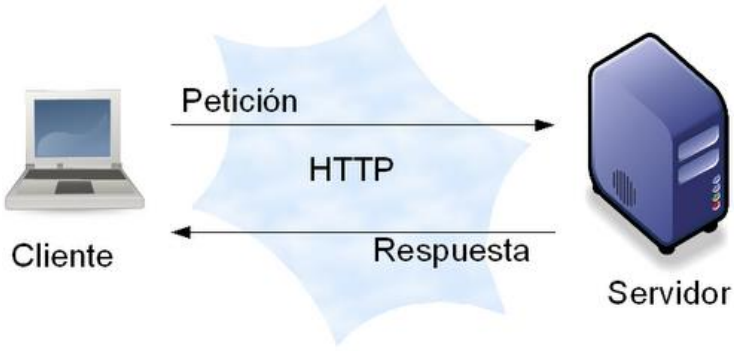


Figura 2. 19.- Comunicación web a través del modelo cliente-servidor.

En la Figura 2. 19 se muestra el modelo cliente-servidor el cual se utiliza para todas las páginas en internet. Consiste en un intercambio de información ante ciertos eventos, estos eventos pueden ser cuando se ingresa la dirección URL al navegador, cuando se da clic en un hipervínculo, o cuando se hace una búsqueda en algún navegador. En cada uno de los casos, la computadora ejecuta una serie de peticiones a las cuales el servidor responde retornando la información solicitada o algún mensaje en caso de algún error.

El internet no es más que distintos dispositivos intercambiando información entre sí, para ello se debe contar con protocolos para retornar información a quien se la solicite. Para atender el intercambio de información y asegurar que la información llega al dispositivo que la solicitó se utiliza el protocolo TCP/IP el cual asigna a los dispositivos dos posibles direcciones IP, públicas o privadas. Independientemente si es privada o pública, la dirección IP se representa por un número binario de 32 bits dividido en cuatro octetos separados por un punto entre ellos. Dependiendo de otros factores como la clase, el protocolo TCP/IP se suele acompañar de una máscara de subred presentada en el mismo formato.

```
Sufijo DNS específico para la conexión. . :
Vínculo: dirección IPv6 local. . . : fe80::c4d3:d008:62db:78e6%19
Dirección IPv4. . . . . . . . . . . . : 192.168.0.4
Máscara de subred . . . . . . . . . . : 255.255.255.0
Puerta de enlace predeterminada . . . . . : 192.168.0.1
```

Figura 2. 20.- Formato de dirección IP en una computadora.

La dirección que suele aparecer en la configuración de la computadora es una IP privada y quien lo asigna es el router. Cada dirección es única para cada dispositivo conectado a la red local y no se repite, cuando se desconecta el dispositivo es posible reasignar su dirección a otro dispositivo por lo que si se reconecta es posible que se le asigne otra dirección IP diferente. En una red local la dirección IP privada consta de tres primeros octetos fijos y uno variable. Esto quiere decir que la dirección IP de un dispositivo será diferente de otro mediante el ultimo octeto el cual varía en un rango de 0 a 255. De este rango el *modem* solo puede asignar 254 direcciones diferentes ya que una la reserva para la puerta de enlace (*Gateway*) y otra para el *broadcast* necesario para la comunicación.

192. 168. 0. x
192. 168. 0. 0
192. 168. 0. 1
. . . .
. . . .
192. 168. 0. 255

Figura 2. 21.- Ejemplo de las posibles direcciones de una red privada.

Los dispositivos conectados a un mismo nodo a través de un router o un *switch* se pueden intercomunicar entre ellos sin necesidad de que exista un proveedor de internet, a esta comunicación se le llama comunicación *LAN* por las siglas de *Local Area Network.* Para que exista comunicación a través de *internet* se debe ocupar una dirección IP pública la cual es asignada al punto de acceso, generalmente un *router.*

En la Figura 2. 22 se observa que cada dispositivo tiene una dirección IP diferente sin embargo todos comparten una IP pública asignada al router mediante la cual se envía y recibe información a través de *internet*.

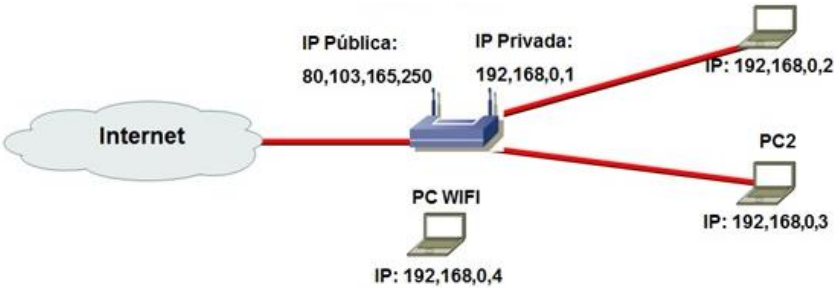


Figura 2. 22.- Diagrama de asignación de direcciones IP públicas y privadas.

Las redes clase A son asignadas a dispositivos como puntos de acceso que necesitan tener direcciones diferentes y son masivos tales como puntos de acceso. Para alcanzar un mayor número de direcciones disponibles solo utilizan el primer octeto para identificar la red mientras que los tres octetos restantes se utilizan para asignar direcciones alcanzando una capacidad de 16 777 214 direcciones diferentes. Este tipo de dirección IP es utilizada principalmente por servidores ya que utilizan esta dirección púbica para entablar comunicación con un cliente además de ser la que se utiliza para ligar el dominio del sitio web al que corresponda.

El internet consiste en un constante intercambio de información mediante el uso de direcciones IP entre sitios web diferentes enlazados a través de puntos de acceso. A pesar de ello el usuario no ingresa manualmente una dirección IP cada vez que desea visitar un sitio, en su lugar coloca una url un navegador quien lo enlaza con el sitio con ayuda de un *servidor de nombre de dominio*, DNS por sus siglas en inglés. Este servidor de dominios asocia la dirección IP pública a un nombre alusivo al sitio web a visitar, el cual es llamado *dominio*. En pocas palabras, en lugar de escribir en el navegador una dirección como 10.17.185.2, se suele escribir *www.unapaginaweb.com* la cual es más fácil de recordar y que está asociada al conjunto de números.

2.7.1 Lenguaje de Marcas de Hipertexto HTML

Se mencionó anteriormente que la comunicación a través de web se realiza por medio de intercambio de paquetes de información, sin embargo, este proceso no es visible por el usuario. Para ello utiliza programas especiales llamados navegadores web, los cuales son básicamente interfaces que interpretan un Lenguaje de Marcas de Hipertexto llamado HTML, que es el lenguaje con el que trabajan las páginas web.

```
<!doctype html>
<html>

  <head>
     <meta charset="utf-8"/>
     <title>Título de la web</title>
  </head>

  <body>
     Contenido de la web
  </body>

</html>
```

Figura 2. 23.- Contenido básico de una página web en HTML.

HTML no es propiamente un lenguaje de programación, aunque es común encontrar fuentes que expresen que si lo es. Codificar en HTML no es complicado ya que no se ocupan símbolos especiales, en lugar de eso se utilizan etiquetas. Para lograr una acción se debe encerrar argumentos en cierto tipo de etiquetas las cuales realizan una función específica sobre el contenido que tenga como argumento. Existen etiquetas de títulos, de vistas, de contenido, entre otras más.

Una página web se caracteriza por tener una estructura definida que se compone de una cabecera (*head*) y un cuerpo (*body*) como se muestra en la Figura 2. 24. El encabezado se coloca dentro de las etiquetas <*head*> además de información como el título y palabras claves que sirven como identificadores para dar información a los motores de búsqueda sobre el contenido de la página.

Figura 2. 24.- Etiquetas colocadas en la cabecera.

La Figura 2. 24 muestra un encabezado básico en el cual se colocan etiquetas que dan información acerca de la página que se está mostrando. Se puede ver que existen dos tipos de etiquetas, unas que contienen instrucciones o parámetros dentro de la misma etiqueta (cuadro verde) y otras que encierran argumentos entre una etiqueta de apertura y una de cierre (cuadro amarillo).

El cuerpo de la página se coloca entre las etiquetas *body*. En este apartado se colocan los argumentos que contendrá la página en sí y que se mostraran mediante el navegador al usuario.

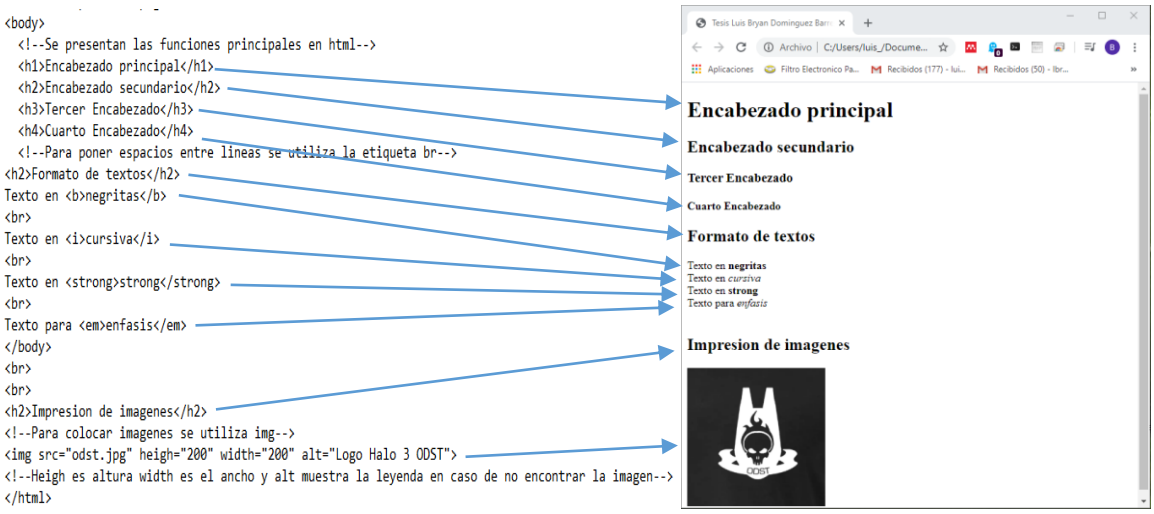


Figura 2. 25.- Etiquetas utilizadas en HTML.

2.7.2 Framework de Yii2

El uso de herramientas como HTML fueron suficientes para las primeras páginas web que se construían en donde se mostraba en su mayoría información en forma de texto con pocas imágenes o videos. En la actualidad el avance de la tecnología permite que los sitios *web* sean más robustos permitiendo ejecutar funciones complejas las cuales son difícil de implementar solo utilizando HTML. Para construir sitios *web* elaborados, es común utilizar otras tecnologías como PHP, JavaScript, CCS, etc. Estas tecnologías permiten implementar animaciones, transiciones, efectos, funcionalidades y demás, sin embargo, utilizarlas todas en un solo proyecto resultaría confuso a nivel código y complicado de administrar los archivos necesarios.

Un *Framework* es un marco de trabajo en donde se conjuntan convenciones y estándares a los que se debe sujetar el usuario para lograr una funcionalidad. Este tipo de entornos obligan al usuario a programar de manera más ordenada lo que resulta en un proyecto mejor estructurado, además de contar con código ya existente que se puede utilizar en forma de plantillas las cuales ayudan a ahorrar trabajo. Estas plantillas resultan ser funciones ya desarrolladas que sirven como base para el desarrollo de un proyecto nuevo las cuales se pueden modificar y adaptar a las necesidades del usuario.

Yii2 es un *framework* de alto desempeño especializado en el desarrollo de sitios web. Este framework trabaja con un patrón de diseño modelo-vista-controlador conocido como MVC. Los modelos representan datos, la lógica de operación, y las reglas que se seguirán; las vistas representan la interfaz que se mostrará al usuario, es decir, la salida de los modelos; y por último los controladores toman datos de entrada y los convierten en instrucciones para los modelos y vistas.

Figura 2. 26.- Diagrama del modelo MVC.

Además del patrón de diseño, el framework de Yii2 cuenta con diferentes entidades como: Scripts de entrada, aplicaciones, componentes de aplicación, filtros y widgets lo que permite integrar muchas funcionalidades a un sitio web de manera más sencilla que solo utilizando HTML.

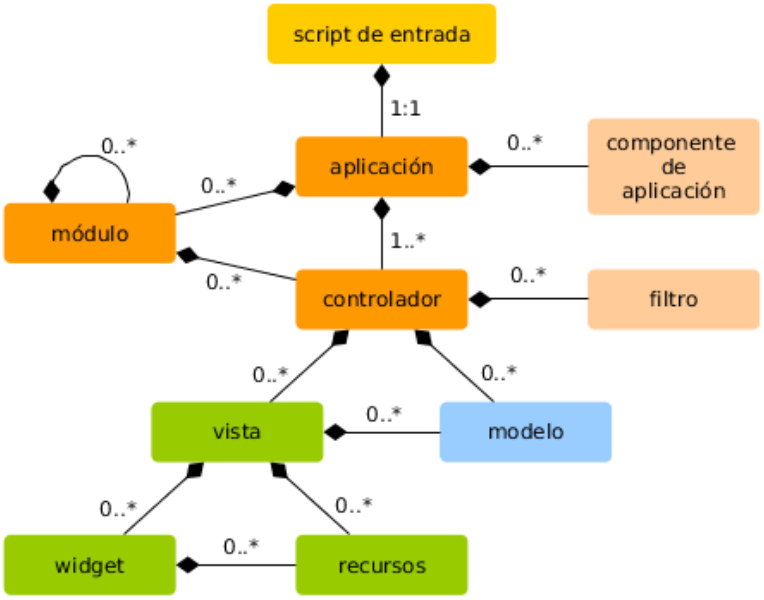


Figura 2. 27.- Relaciones entre las entidades disponibles en el framework de Yii2.

2.7.3 NetBeans

NetBeans es un entorno de desarrollo integrado (IDE) gratuito y de código abierto el cual se emplea para el desarrollo de aplicaciones en diferentes sistemas operativos. Se puede utilizar para el desarrollo de aplicaciones *web*, corporativas, de escritorio o incluso de aplicaciones móviles basadas en lenguaje *Java* y *HTML5*. El IDE adicionalmente es capaz de interpretar lenguajes como PHP y C/C++.

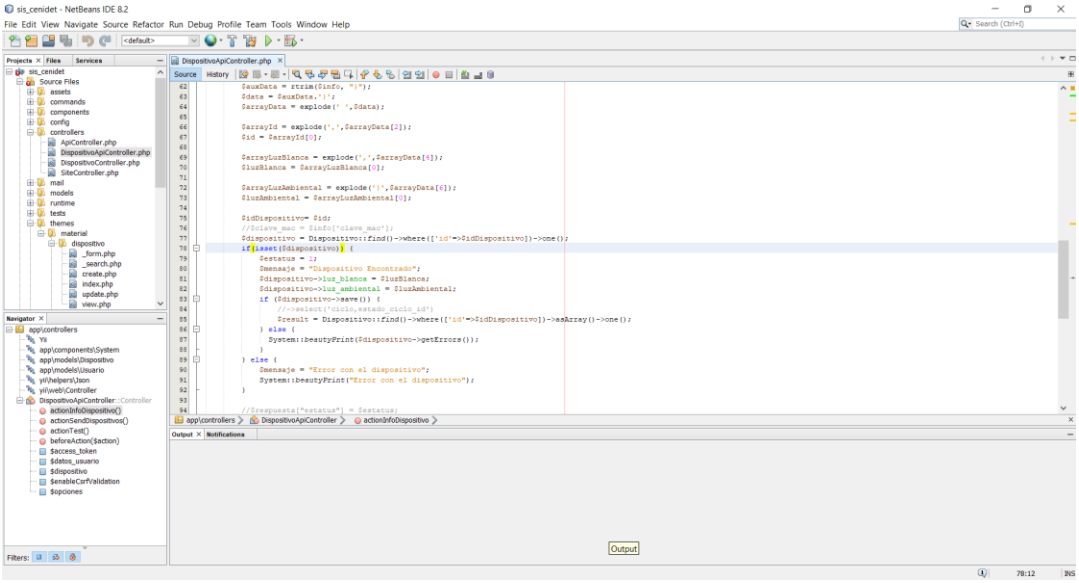


Figura 2. 28.- Interfaz de programación de NetBeans.

Esta IDE es muy útil debido a que soporta una gran variedad de lenguajes de programación además de la posibilidad de enlazar el programa al servidor en línea, esto permite trabajar de manera más eficaz programando desde un entorno de programación y no directamente en los archivos del host además de hacer visualmente más cómoda la tarea.

2.8 Estado del arte y aplicación de la tecnología.

Existen muchos trabajos reportados en la literatura que buscan trabajar la electrónica de potencia en conjunto con tarjetas digitales con distintos fines. Ya sea para centralizar el control de diferentes dispositivos domóticos, implementación de rutinas para ahorro energético, por motivos de seguridad o para implementar ciudades inteligentes con dispositivos controlados inalámbricamente [13]. Estas aplicaciones cada vez se acercan más al llamado *internet de las cosas*. El *internet* de las cosas no solo significa comunicación a través de internet, dentro de este nuevo ecosistema tecnológico se utilizan módulos con diferentes comunicaciones como *Bluetooth*, *WiFi*, *Ethernet*, *Mesh*, etc. Para crear aplicaciones y entornos como ciudades inteligentes[14].

El control de iluminación ha sido un tema de interés al contar con nuevas tecnologías que permitan utilizar de manera más inteligente la energía consumida. Uno de los principales

protocolos implementados es DALI (*Digital Addressable Lighting Interface*) basado en el estándar IEC62386. Este protocolo utiliza 5 cables, dos de alimentación eléctrica, dos cables sin polaridad y uno de tierra física. El par de alimentación energiza el sistema y consta de una línea de neutro y la otra maneja la fase. El par trenzado sin polaridad corresponde al bus DALI por el cual se envía y recibe información de todos los dispositivos.
En una instalación común se cuenta con los cables de línea, neutro y tierra física por tanto agregar el protocolo DALI implica agregar únicamente dos hilos adicionales en el conducto que lleva el cableado. A través del bus de datos DALI se manda la instrucción seguido de la dirección del dispositivo al cual se quiera comunicar, por ello muchos dispositivos pueden compartir el mismo bus sin existir interferencia entre ellos.

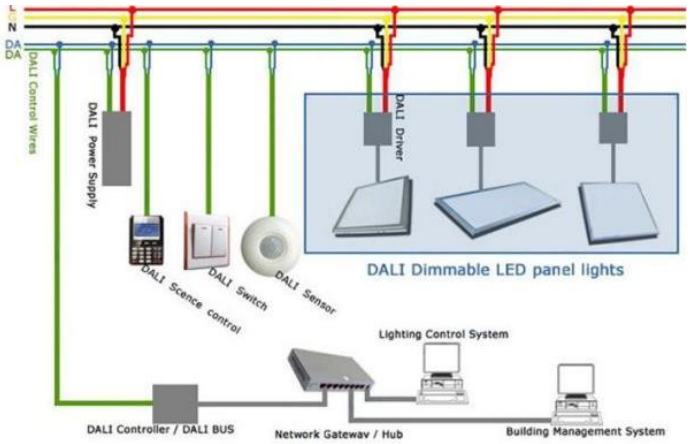


Figura 2. 29.- Ejemplo de conexión del protocolo DALI.

Se ha mencionado mucho a cerca del protocolo DALI y es que se popularizó tanto que se hizo un estándar. Esto significa que cualquier dispositivo que se desee lanzar al mercado bajo esta norma está obligado a realizar pruebas de laboratorio como: tiempos de respuesta, respuesta ante fallos, reseteos y niveles de regulación, entre otras cosas, con el fin de asegurar el correcto funcionamiento y compatibilidad entre equipos de distintos fabricantes.
Es común ver este tipo de protocolo en oficinas ya que debido a su versatilidad es sencillo realizar cambios sin afectar el cableado, ya que se pueden armar bloques para encender o apagar simultáneamente las luminarias y cambiar una lámpara de bloque no implica más que modificar sus direcciones [15].

En [16] se presenta un trabajo en donde se realiza un sistema de iluminación LED trabajando con el protocolo DALI. En él se diseña una interfaz de usuario para controlar el sistema vía USB, un sistema de atenuación con un PIC16F684 y un circuito puente para establecer comunicación entre la computadora y el circuito.

A pesar de que el protocolo DALI no es idóneo para exteriores por su limitado número de direcciones, existen trabajos que implementan el protocolo DALI tanto para alumbrado público como para iluminación doméstica. Estos trabajos automatizan completamente las luminarias del hogar o implementan algoritmos que permiten hacer más eficiente el uso de

lámparas públicas encendiendo solo cuando hay presencia de usuarios en un área determinada [17].

Sabiendo que el protocolo DALI es cableado, existen trabajos que combinan las capacidades de dicho estándar con protocolos de telemetría para acercarse al paradigma del internet de las cosas. De acuerdo a [18], el término IoT hace referencia a un área que gana importancia día a día con el avance de la tecnología y muestra un trabajo en el que se utiliza el protocolo DALI para realizar la comunicación entre un grupo de lámparas e implementa un *Raspberry* Pi para mandar los datos recabados por otro protocolo llamado MQTT, por las siglas en inglés del nombre Message Queuing Telemetry Transport.

Existen en la literatura otros trabajos en donde, a diferencia de los anteriores, no se basan en algún estándar y se desarrolla control de iluminación con alternativas inalámbricas. En [19] y [14] se implementa un sistema en el que se instrumenta la iluminación del inmueble para responder a tarjetas de identificación por radiofrecuencia, mejor conocidas como RFID por sus siglas en inglés. Estos sistemas contemplan al usuario como una variable del sistema de control de energía. En ella se hace un encendido selectivo de lámparas para lograr una iluminación óptima alrededor del usuario. El sistema determina la posición del usuario mediante su tarjeta RFID, la cual porta, misma que constantemente está comunicada con el sistema.

En [20] se muestra un sistema de alumbrado público en el contexto de una red inteligente. El objetivo del trabajo es monitorear remotamente el alumbrado público y apagar o encender el sistema automáticamente considerando temporización, condiciones climáticas, visibilidad y fallas. Este sistema utiliza comunicación por radiofrecuencia mediante dispositivos Xbee, lo que permite que cada poste de luz contenga un módulo con distintos sensores y este comunicado con los demás mediante RF. La señal no necesita ser tan potente debido a que cada nodo funciona como un repetidor por lo que la señal viaja de poste a poste creando una red tan larga como dispositivos se tenga.

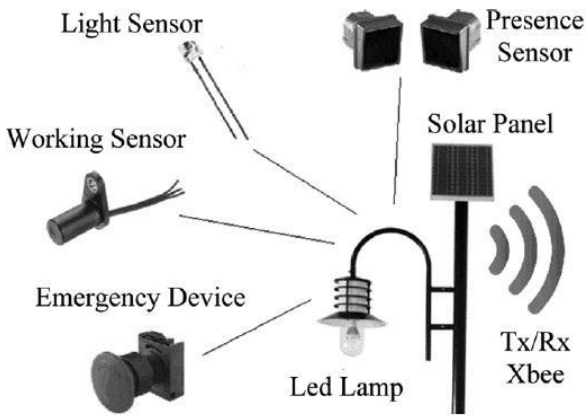


Figura 2. 30.- Nodo emisor de un sistema de alumbrado público con ZigBee.

En [21] se desarrolla un sistema para exteriores que utiliza un servidor *web* el cual recibe en tiempo real información acerca del clima al mismo tiempo de proveer de una interface para usuarios autorizados. El sistema es capaz de mostrar en tiempo real los parámetros de intensidad de la señal entre las diferentes lámparas LED. El sistema toma el dato de un servidor *web* al mismo tiempo que la comunica vía GSM a un nodo inicial del sistema, a partir de ese nodo la señal es retransmitida poste a poste mediante enlace ZigBee.

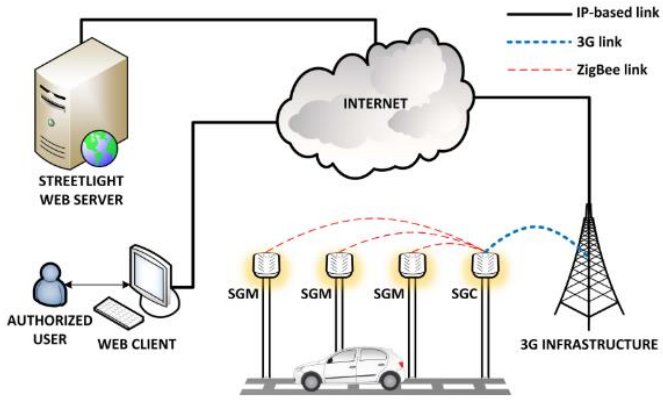


Figura 2. 31.- Comunicación multiprotocolo para iluminación en exteriores.

Existen fabricantes que aprovechan el direccionamiento por IP para funcionar, así como lograr conectar muchos dispositivos e intercomunicar con funciones adicionales. En [22] se muestra un equipo de un fabricante dedicado a la instalación de interfones, este periférico se conecta a sus tarjetas principales dando la capacidad de controlar iluminación. Este fabricante maneja diferentes tecnologías, pero las últimas que ha integrado son con comunicación basada en IP utilizando dichas direcciones para cada dispositivo y agregando la capacidad de montar video y audio sobre conexión rs232 con un puerto RJ45 siendo compatibles con *switches* y *routers* comerciales.

Otros trabajos muestran trabajos para automatizar viviendas con tecnología inalámbrica y procesando los datos a través de la computadora utilizándola como dispositivo terminal. Este trabajo muestra el uso de sensores ZigBee los cuales son muy atractivos ya que pueden crear grandes redes de sensores[23].

La tendencia que se sigue es hacer inteligentes diferentes sistemas con múltiples propósitos. Uno de los conceptos con mucha oportunidad de crecimiento son las ciudades inteligentes en donde se pretende realizar funciones como: monitoreo de ruido, contaminación ambiental, congestión de tráfico y ahorro energético. Este tipo de sistemas pretende gestionar de una mejor manera la energía utilizada monitoreando la cantidad de electricidad consumida, identificando las principales fuentes de consumo energético y estableciendo planes y prioridades para optimizar su comportamiento. Uno de los temas importantes es el alumbrado inteligente el cual se puede optimizar aplicando algoritmos o encendiendo gradualmente la luminaria de acuerdo a la hora del día, las condiciones climáticas y la presencia de personas [24], [25].

El avance de la tecnología permite que cada vez sea más sencillo y eficiente instrumentar sistemas con sensores y dispositivos de comunicación para lograr participar en paradigmas como ciudades inteligentes e internet de las cosas que buscan una interconexión de dispositivos de uso cotidiano para una mejor interacción con el entorno y un ahorro energético. Adicionalmente, la electrónica digital ha permitido el desarrollo de múltiples tarjetas de comunicación inalámbrica que permiten desarrollar sistemas de monitoreo y control que puedan implementarse en aplicaciones ya establecidas en los que otros protocolos, como DALI, requerirían una modificación del cableado importante.

Se ha mencionado que existen muchas tarjetas digitales que cuentan con diferentes protocolos integrados en mismo circuito lo cual ayuda al avance del IoT. Algunas tarjetas son desarrolladas por grandes compañías como por ejemplo Sil Labs con la tarjeta *Migthy Gecko* que cuenta con protocolos *Bluetooth*, *Mesh* y *ZigBee* a un precio aproximado de 49 dólares. Particle con la tarjeta Boron que cuenta con comunicación *WiFi, Mesh* y *Bluetooth* a un precio aproximado de 28 dólares y Pycom con la tarjeta WiPy con Bluetooth y WiFi con precio de 28 dólares. Es importante mencionar que además de las tarjetas en ocasiones es necesario comprar tarjetas de expansión para actualizar el *firmware* cargado por defecto en la tarjeta. Esto implica comprar un circuito más, elevando el costo total del proyecto.

Figura 2. 32.- Tarjeta de expansión para el circuito WiPy.

Se pueden diseñar soluciones con multiprotocolo utilizando las tarjetas antes mencionadas o armar circuitos con tarjetas basadas en módulos de uso libre como por ejemplo el ESP8266 y módulos Bluetooth Hc05-06 con precios aproximados de 12 y 5 dólares respectivamente. Existen además tarjetas que utilizan como base el chip ESP8266, como Feather Huzza o Node MCU. Estas tarjetas se han logrado programar con muchos lenguajes y a través de distintos entornos de programación lo que hace posible que pueda integrarse a una gran diversidad de proyectos. Las tarjetas cuentan con circuitos para funcionar tan pronto se cargue el programa, con entradas y salidas de propósito general (GPIO) y con convertidores ADC o DAC lo que eleva la gama de aplicaciones con las que pueden trabajar.

2.9 Propuesta de solución

La solución que se propone es crear un sistema que sea capaz de controlar la iluminación de una lámpara led para un recinto semi abierto en donde incida luz natural. El sistema debe ser capaz de medir la cantidad de iluminación que exista en todo momento y comunicarlo con el sistema para ajustar la iluminación en función de la luz natural incidente.

A diferencia de las soluciones comerciales, se trabajará con el diseño de la fuente de alimentación para lograr modificar la iluminación y no solo realizar un control ON-OFF. Esto permitirá aprovechar la iluminación natural controlando la intensidad de la lámpara, ahorrando energía con dicha acción.

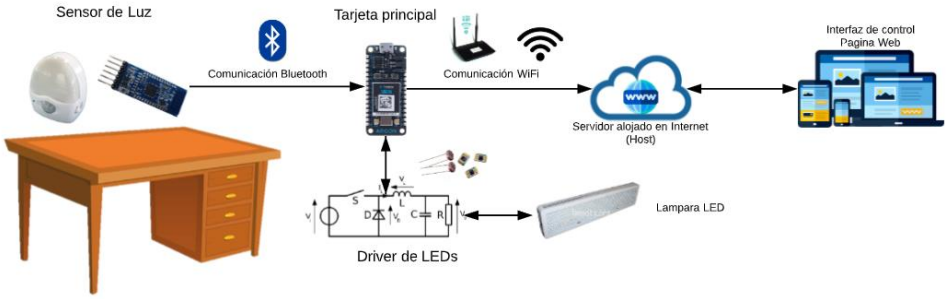


Figura 2. 33.- Diagrama a bloques de la propuesta de solución.

El sensor de iluminación no estará integrado en la lámpara por lo que la comunicación con la tarjeta principal se realizará utilizando comunicación *Bluetooth* como se observa en el diagrama de la Figura 2. 33.

Se diseñará un sitio web en línea al cual deberá poder acceder el sistema, el cual servirá como interfaz de usuario y mostrará algunos datos recolectados por el sensor de iluminación. En esta misma interfaz se podrán configurar el estado de la lámpara así como el nivel al cual la lámpara ajustará la iluminación entregada de acuerdo a estándares (o selección personal del usuario) [2]. Para este objetivo el sistema debe contar con comunicación inalámbrica utilizando *WiFi*.

La interfaz de usuario será basada en tecnologías *web* para monitorear las variables adquiridas mediante los sensores, el estado del sistema de iluminación, así como para la comunicación con él. La plataforma estará programada con ayuda de Yii2 para crear la interfaz y los comandos de comunicación con la tarjeta a través de *internet*, la cual es una tecnología libre y al programarse desde cero el proyecto se tiene la certeza del flujo de información que seguirá el prototipo.

Con el fin de conocer el flujo de información que se controla se utilizará un servidor propio y se configurará desde cero, se programará el sitio web, la interfaz de usuario, las rutinas de comunicación con las tarjetas (API´s) y las bases de datos necesarias para el proyecto.

Se utilizarán tarjetas de comunicación *WiFi* libres basados en chips ESP8266 y módulos de comunicación de bajo costo HC05 y HC06. Se programarán las tarjetas utilizando entornos de programación libres con sus respectivos lenguajes de programación integrados como Arduino con C#, así como *Upycraft* con *MicroPython*.

2.10 Diseño del convertidor

Como se mencionó en capítulos anteriores, se implementará una fuente conmutada basada en la topología reductora *Buck* debido a que se ajusta en buena manera a las necesidades del proyecto. La lámpara se utilizará en interiores, por lo tanto, la energía para alimentar la lámpara se obtendrá de la línea eléctrica de CFE, la cual tiene un valor nominal de 127 VCA. Considerando que la lámpara LED trabaja con un valor nominal de 60 VCD es necesario rectificar el voltaje de línea y reducir su amplitud con el fin de ajustar la alimentación a los parámetros de operación de la lámpara.

Como ya se conoce, el convertidor cuenta con modos de operación ON y OFF. A partir de estos dos estados se obtienen graficas de funcionamiento, que posteriormente se utilizan para obtener las expresiones que describen la operación del circuito y con las cuales se realiza el diseño y cálculo de los componentes. Una de las gráficas de interés es la corriente en el inductor en la cual se puede observar el comportamiento ante la intermitencia de estados ON y OFF como se muestra en la Figura 2. 34.

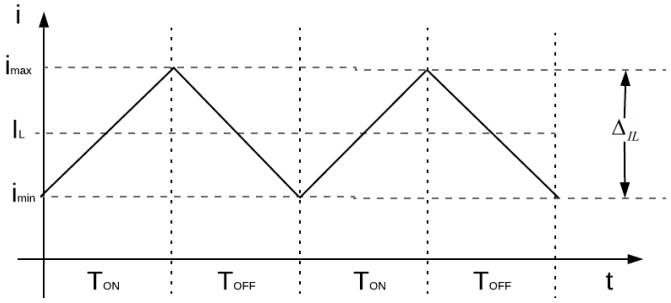


Figura 2. 34.- Gráfica de la corriente en el inductor.

Otro gráfico de utilidad es el comportamiento de la corriente en el capacitor con la cual se obtiene el valor de capacitancia en el convertidor Buck. La forma de onda de la corriente promedio en el inductor es la misma que la de la corriente promedio en el capacitor, pero con un desplazamiento ya que, en el caso del capacitor, si hay un cambio de dirección en la corriente, tal como se muestra en la siguiente figura.

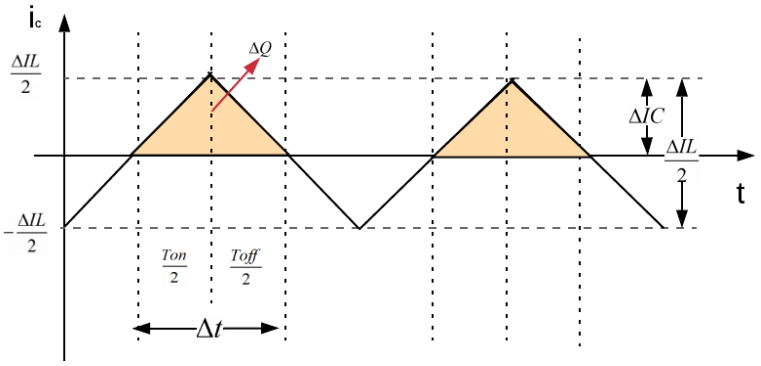


Figura 2. 35.- Gráfica de corriente en el capacitor.

Para obtener los valores de los componentes del convertidor se utilizará principalmente el estado ON. En este estado, mostrado en la Figura 2. 4, pueden aplicarse leyes de voltaje de Kirchhoff de donde se obtienen las siguientes expresiones:

$$v_i - v_L - v_o = 0 \qquad (2.\ 1)$$

$$L = \frac{(v_i - v_o)dt}{di} \qquad (2.\ 2)$$

Del análisis de la malla se obtiene la ecuación (2. 1) de donde se despeja la inductancia resultando la ecuación (2. 2). Para poder obtener un valor numérico de la inductancia se puede asumir que el diferencial de tiempo (dt) se puede sustituir por el periodo en el que es válida la ecuación. En el estado ON el tiempo de encendido es el producto del ciclo de trabajo D por el periodo de activación T_s. El diferencial de corriente (di) se puede aproximar como el rizo de corriente que se tendrá producto de la conmutación, se representa como Δ_{i_L} .

$$L = \frac{(v_i - v_o)DT_s}{\Delta_{i_L}} \qquad (2.\ 3)$$

Para calcular el valor del capacitor se hace uso de la gráfica de voltaje del capacitor. En primera instancia se asume que la corriente promedio en el inductor es igual a la corriente promedio en el capacitor.

$$I_L = I_c \qquad (2.\ 4)$$

También es conocida la ecuación de la corriente en el capacitor la cual es:

$$I_c = C\frac{dV_c}{dt} \qquad (2.\ 5)$$

Tomando en cuenta que dV_c representa el rizo de voltaje que corresponde al rizo de voltaje, representado por V_{pp} y dt el diferencial de tiempo Δt , la ecuación (2. 5) se puede manipular de la siguiente manera:

$$I_c \Delta t = CV_{pp} \tag{2. 6}$$

Aprovechando esta forma se debe recordar el concepto de carga del capacitor.

$$Q = CV(1 - e^{-\frac{t}{RC}}) \tag{2. 7}$$

Esta fórmula se simplifica cuando se trabaja en el estado estable en donde después de mucho tiempo el exponencial se aproxima a cero, resultando en:

$$Q = CV \Rightarrow \Delta Q = CV_{pp} \tag{2. 8}$$

Sustituyendo la expresión (2. 8) en la (2. 6).

$$I_c \Delta t = \Delta Q \tag{2. 9}$$

En la Figura 2. 35 se puede observar que la carga del capacitor es el que se encierra bajo la curva, el cual forma un triángulo. De esta manera se puede utilizar la fórmula del área del triángulo para obtener el diferencial de carga sabiendo que la base vale Δ_t y la altura resultando en la ecuación (2. 10).

$$\Delta_Q = \frac{\Delta_{i_L}}{8 f_s} \tag{2. 10}$$

Sustituyendo Δ_Q de la ecuación (2. 10) en la ecuación (2. 8) se obtiene:

$$CV_{pp} = \frac{\Delta_{i_L}}{8 f_s} \tag{2. 11}$$

$$C = \frac{\Delta_{i_L}}{8 f_s V_{pp}} \tag{2. 12}$$

Conociendo que el voltaje pico a pico es equivalente al rizo que se presentara en el voltaje la expresión necesaria para calcular el capacitor es la mostrada en la ecuación (2. 13).

$$C = \frac{\Delta_{i_L}}{8 f_s \Delta_{V_C}} \tag{2. 13}$$

Para obtener la resistencia que representa la carga se utiliza la ley de Ohm como se muestra a continuación:

$$R_L = \frac{V_o}{I_o} \tag{2. 14}$$

El ciclo de trabajo máximo al cual operará el convertidor se calcula con ayuda de la ganancia la cual para el convertidor Buck es igual al ciclo de trabajo y corresponde a la relación entre el voltaje de salida con respecto al de entrada como se presenta en la siguiente ecuación.

$$D = \frac{V_o}{V_i} \qquad (2.\ 15)$$

La luminaria que se utilizará para el trabajo es una lámpara que opera con una tensión de 60V y tiene un consumo de 0.5 A. Esta lámpara está conformada por un arreglo serie-paralelo de LEDs blancos cuyo conjunto consume una potencia de 30W.

Figura 2. 36.- Lámpara de prueba marca MAGG de 30W.

Se decidió utilizar estos valores debido a que muchas de las lámparas colocadas en oficinas tienen estas características incluyendo las instalaciones de CENIDET. Estas lámparas son suficientes para iluminar laboratorios y oficinas por lo que son una buena referencia.

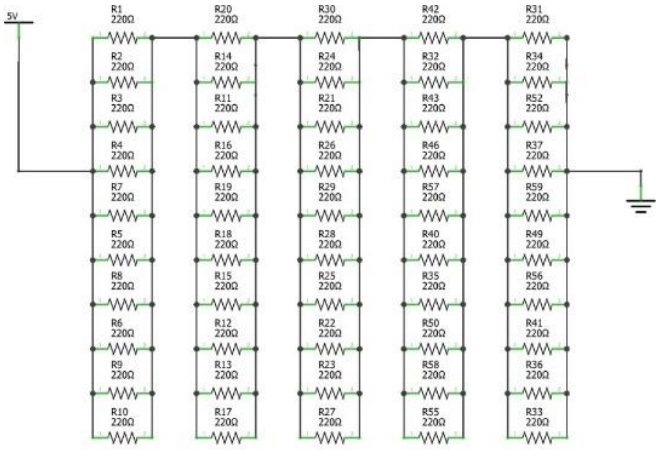


Figura 2. 37.- Diagrama eléctrico equivalente de la lámpara LED.

Para seleccionar los parámetros de diseño se consideró el tamaño resultante de los componentes que integran el convertidor. Es por esta razón que se propuso utilizar una señal PWM de alta frecuencia para disminuir el tamaño tanto del capacitor como del inductor.

Los parámetros de diseño obtenidos se concentran en laTabla 2. 3..

Tabla 2. 3.- Parámetros de diseño del convertidor

$V_{in} = 180\ V_{CD}$	$P = 30W$	$\Delta_{V_c} = 0.5V$	$fs = 100kHz$
$\Delta_{i_L} = 0.15A$	$V_{out} = 60V_{CD}$	$I = 0.5A$	

De acuerdo a la Tabla 2. 3 se conoce el voltaje de entrada y el de salida del convertidor por lo que se puede calcular el ciclo de trabajo con la ecuación (2. 15).

$$D = \frac{60V}{180V} = 0.3333 \qquad (3.\ 1)$$

La ecuación (3. 1) demuestra que para lograr los 60_{VCD} requeridos en la salida el ciclo de trabajo debe ser del 33%. Por otro lado, se conoce el voltaje de salida y la corriente que circulará por la lámpara, por lo que se puede calcular la resistencia de carga equivalente que tendrá el convertidor mediante la ley de Ohm.

$$R_L = \frac{V_o}{I_o} = \frac{60V}{0.5A} = 120\Omega \qquad (3.\ 2)$$

Una vez calculada la resistencia de carga se calcula el valor del inductor con la ecuación (2. 3).

$$L = \frac{(180-60)0.3}{(0.3*0.5)100000} = 2.66mH$$

Para el cálculo del capacitor se utiliza la ecuación (2. 13) obtenida en el capítulo anterior.

$$C = \frac{0.15}{8*100000*0.5} = 375nF \qquad (3.\ 3)$$

De acuerdo a la ecuación 3.3 lo mínimo de capacitancia requerida para lograr el rizo de 0.5V son 375 nF el cual es un valor de capacitancia aceptable e incluso se podría reducir el rizo más con un capacitor más grande.

Una vez calculado el valor de los componentes con los que se construirá el convertidor Buck, se deben calcular los esfuerzos a los que estarán sometidos los dispositivos de conmutación para realizar la selección de los componentes, dichos esfuerzos se presentan en la siguiente tabla.

Tabla 2. 4.- Esfuerzos en los componentes electrónicos del convertidor.

$V_{DS} = 180V$	$I_{DS} = 0.5A$	$V_{AK} = 180V$
$I_{AK} = 180V$	$I_L = 0.5A$	$V_C = 60V$

Con la Tabla 2. 4 se puede seleccionar tanto el transistor como el diodo que conmutarán en el convertidor. De esta manera para seleccionar el diodo se debe cuidar que resista un voltaje inverso mayor a los 180V.

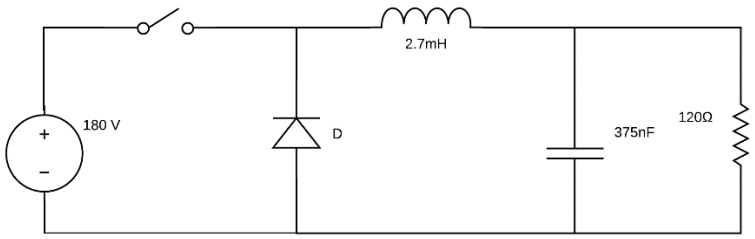


Figura 2. 38.- Convertidor Buck con valores definidos.

Para realizar el diseño se utiliza el diagrama que se muestra en la Figura 2. 3 sin embargo, es muy común realizar un cambio equivalente en el diseño con el fin de poder acoplar las tierras de la etapa de potencia y la etapa de control para cuando se introduzcan los pulsos PWM no exista ningún problema y el circuito responda como se diseñó, este cambio se presenta en la siguiente figura.

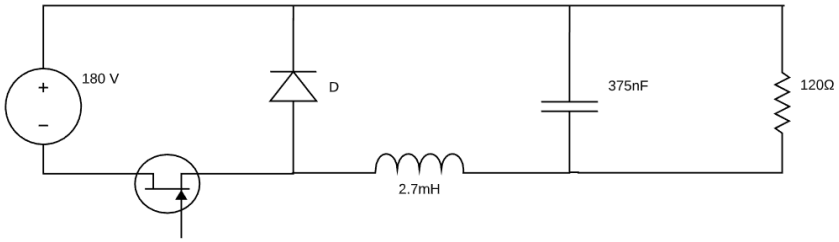


Figura 2. 39.- Convertidor Buck aterrizado a tierra.

Como se puede apreciar, tanto el interruptor como la bobina cambiaron su lugar dentro de la misma malla sin alterar el principio de funcionamiento, de esta manera el circuito está referenciado a tierra permitiendo acoplar las tierras tanto de la señal PWM como del impulsor que disparará al transistor.

2.11 Diseño del circuito de medición de luz

Una pieza clave en el diseño del prototipo es el circuito encargado de medir la cantidad de luz ambiental existente en el recinto, con el fin de que el controlador ajuste la energía que proporcionará la lámpara. Como se había mencionado en la propuesta de solución, el circuito será colocado en un punto estratégico para medir la incidencia de luz ambiental a la par de la luz aportada por la lámpara presente en el lugar, esta medida será enviada al controlador ubicado junto a la lámpara, por lo que una conexión alámbrica no es práctica.

2.11.1 Sensor de iluminación VCNL4040

Para poder registrar el nivel de iluminación se utiliza el sensor de luz ambiental VCNL4040 el cual cumple con dicha función y entrega una cantidad en formato binario de 16 bits mediante comunicación serial. Este chip incluye además un sensor de movimiento mediante un emisor infrarrojo de potencia el cual también se puede habilitar activando el registro correspondiente [26].

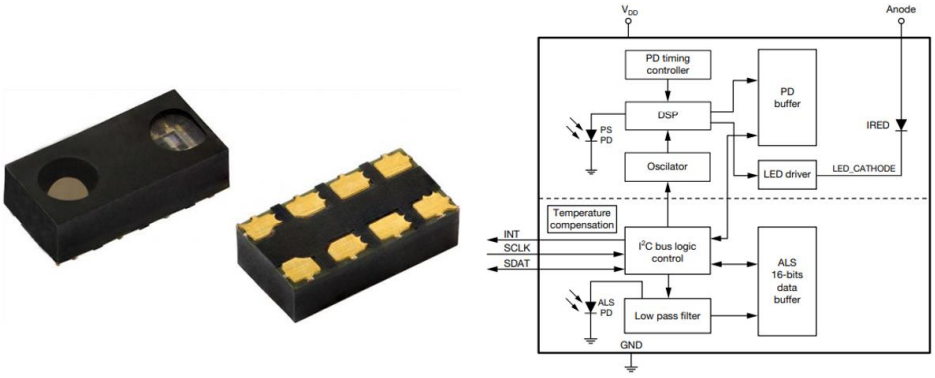


Figura 2. 40.- Sensor VCNL4040 y su diagrama esquemático interno.

Tal como se muestra en la Figura 2. 40, el sensor es un *chip* muy pequeño de montaje superficial el cual internamente cuenta con un sensor de luz ambiental y un sensor infrarrojo para la función como sensor de presencia. Este sensor está diseñado para comunicarse con tarjetas digitales por medio del protocolo I2C entregando la medida exacta sin necesidad de que el microcontrolador realice una interpretación, ya que el procesado de la información lo hace internamente el *chip*.

Debido a que el *chip* utiliza comunicación I2C se deben agregar algunos componentes para lograr la comunicación entre el sensor y el microcontrolador que leerá la información. Se deben agregar un par de resistencias de *pull-up* en los canales de comunicación marcados como SDA y SCL además de agregar conexiones externas y capacitores para filtrar ruido [27].

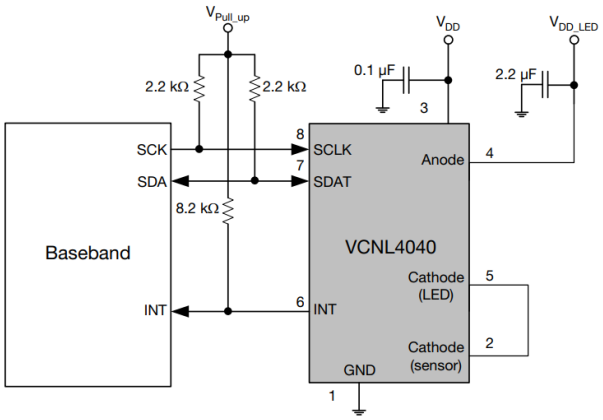


Figura 2. 41.- Circuitos mínimos recomendado por el fabricante para la comunicación I2C.

Una vez realizado el circuito se debe entender cómo comunicarse con el sensor y de los diferentes bits que utiliza que son: bit de inicio (start), bit de paro (stop) y bit de respuesta (*acknowledge*). Este protocolo funciona con tramas de 8 bits alternando entre los bits de cabecera para lograr la comunicación, por ejemplo, para enviar un dato se debe iniciar la comunicación, luego enviar la dirección del dispositivo con el que se comunicará y posteriormente el dato que se desee enviar, este proceso se ilustra en la Figura 2. 43.

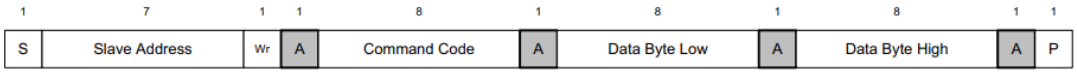


Figura 2. 42.- Proceso de escritura por el bus I2C.

De la misma manera para leer un dato se realiza un proceso similar, se debe mandar la dirección del dispositivo con el cual se desea enlazar, el comando que debe ejecutar el sensor y luego de eso el sensor responderá con una trama de 16 bits separado en dos bloques de 8 bits por un bit de respuesta.

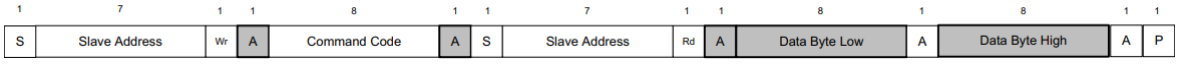


Figura 2. 43.- Proceso de lectura por protocolo I2C.

Una vez determinado cómo es la comunicación y conectadas las resistencias y periféricos necesarios para lograr la comunicación se debe programar el microcontrolador el cual será el que se comunique con el sensor y con esto se logra la primera parte del diseño el cual es determinar la cantidad de iluminación presente.

Se mencionó que el sensor es capaz de dar información en un formato hasta de 16 bits sin embargo esto tiene que ver con la resolución que se seleccione.

PART NUMBER	OPERATING RANGE (mm)	OPERATING VOLTAGE RANGE (V)	I²C BUS VOLTAGE RANGE (V)	IRED PULSE CURRENT (1) (mA)	AMBIENT LIGHT RANGE (lx)	AMBIENT LIGHT RESOLUTION (lx)	OUTPUT CODE	ADC RESOLUTION PROXIMITY / AMBIENT LIGHT
VCNL4040	0 to 200	2.5 to 3.6	1.8 to 3.6	200	0.0125 to 6553	0.0125	16 bit, I²C	16 bit / 16 bit

Figura 2. 44.-Características de salida del sensor VCNL4040.

Para el desarrollo del presente proyecto se utilizará la configuración por defecto el cual consiste en utilizar una resolución de 16 bits para tener una lectura con una sensibilidad de 0.0125luxes por cada bit. Cuando el microcontrolador se comunica con el sensor obtiene lecturas que corresponden a enteros y cuya magnitud corresponde a los luxes.

2.11.2 Comunicación Bluetooth

Una vez que obtiene el valor de iluminación por parte del sensor VCNL4040 se debe enviar la información al módulo principal que a su vez lo enviará al sitio web para almacenarlo en la base de datos y presentarlo en la interfaz. El protocolo más viable de integrar es el *Bluetooth* ya que el sistema requiere una distancia solo de un par de metros, tiene una cadencia baja de comunicación y además los módulos con dicho protocolo no son costosos.

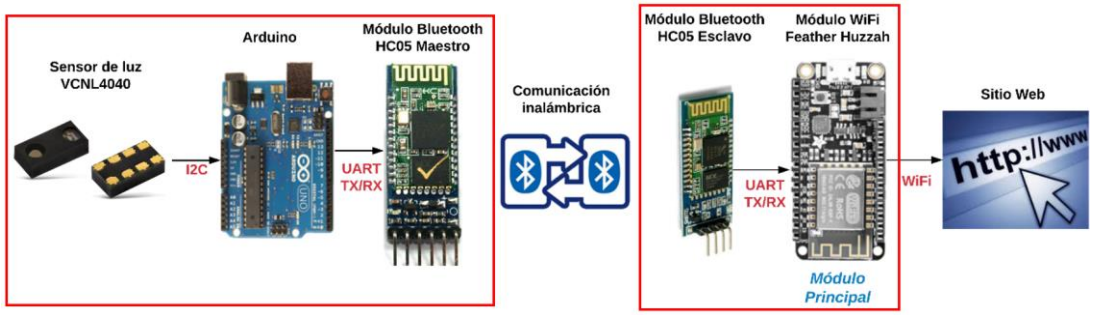


Figura 2. 45.- Esquema de la comunicación entre el circuito sensor de iluminación y el módulo principal.

La comunicación entre los módulos *Bluetooth* no es muy compleja sin embargo se deben realizar configuraciones para lograr entablar comunicación inalámbrica. Si bien la configuración se puede realizar mediante un microcontrolador enviando comandos de configuración previamente especificados, es mejor utilizar una computadora, agregando un convertidor TTL-USB y utilizando una terminal de comunicación serial. El IDE de Arduino es muy útil para enviar una cadena de caracteres a diferencia de otros, como Putty o TeraTerm, quienes envían un caracter a la vez.

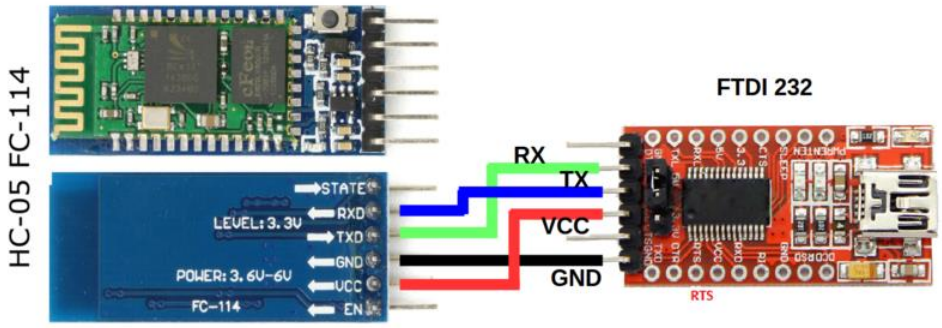


Figura 2. 46.- Conexión del módulo Bluetooth con un convertidor TTL-USB.

Una vez conectado el módulo como se aprecia en la Figura 2. 46, la computadora lo reconocerá como un puerto COM y le asignará un numero de puerto COM/LPT. Desde el IDE de Arduino se abre el monitor serial y se envían los comandos AT.

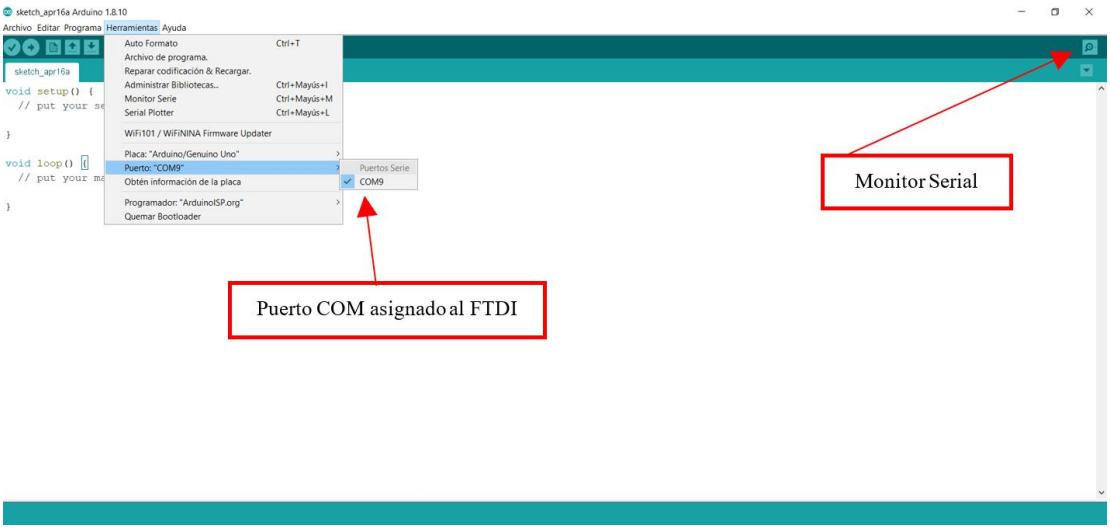


Figura 2. 47.- Configuración del módulo Bluetooth a través del IDE de Arduino.

Con el puerto COM asignado el IDE reconocerá al FTDI conectado al HC05 y se puede comunicar con el monitor serial. Desde la terminal, el primer comando que se debe mandar para saber si el módulo se encuentra conectado correctamente y los parámetros están bien ajustados es *AT*. El modulo *Bluetooth* responderá con un OK si se ajustó bien la velocidad de comunicación y si las conexiones se realizaron correctamente.

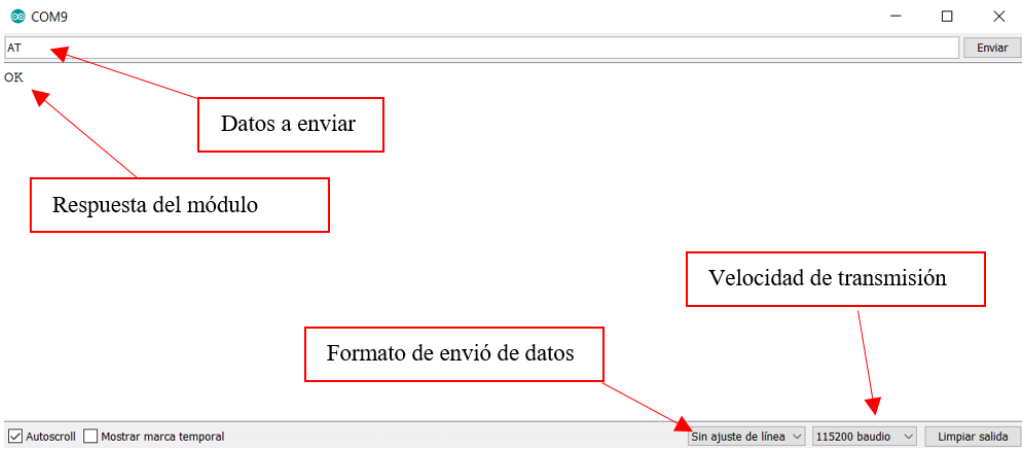


Figura 2. 48.- Envío del comando de prueba.

Si al realizar la prueba ilustrada en la Figura 2. 48 el módulo responde correctamente, se pueden realizar otras configuraciones. El módulo permite la modificación de parámetros como el nombre del dispositivo, el pin, la velocidad de transmisión, entre otras cosas. Los comandos para estas configuraciones se enlistan en la Tabla 2. 2.

Para configurar la conexión entre el módulo maestro y el esclavo es necesario conocer la dirección MAC de ambos y homologar la velocidad de transmisión. En principio se debe acceder al dispositivo maestro y ponerlo en modo conexión con el comando *AT+CMODE=0* y posteriormente enviar la dirección MAC del dispositivo al cual se conectará. La dirección MAC debe enviarse en formato de 4, 2, 6 dígitos con el comando BIND. Por ejemplo, la dirección MAC del esclavo en el presente prototipo es 98:D3:51:FD:94:79 por lo que el comando quedaría como: *AT+BIND=98D3,51,FD9479*. Si al enviar el comando *BIND* se retorna un OK la configuración se realizó exitosamente.

El módulo *Bluetooth* una vez conectado se comporta como un "puente" entre el microcontrolador con el cual se conecta y el microcontrolador conectado al otro punto de la comunicación, ya que todo lo que llega por su pin TX lo transmite de manera transparente mediante *Bluetooth*, asimismo todo lo que recibe de manera inalámbrica lo transfiere tal cual a su pin RX. Es importante saber que una vez conectado el módulo no responde a comandos AT.

2.11.3 Integración del circuito sensor de luz

Con las consideraciones antes mencionadas se puede realizar la integración de todo el módulo seleccionando por último el controlador que se encargará de obtener la información del sensor y de enviarla inalámbricamente a través del módulo *Bluetooth*. El microcontrolador que se utilizará es un Arduino UNO, ya que además de ser de uso libre es económico, de bajo consumo y es capaz de comunicarse con ambos dispositivos y el diseño puede replicarse utilizando cualquiera de sus demás versiones, desde el Arduino nano hasta el Arduino Mega.

El circuito emplea dos tipos de comunicación: I2C y UART para comunicarse con el sensor de luz y para enviarlo al módulo *Bluetooth* respectivamente. Para el protocolo I2C se ocupan los pines marcados como SCLK Y SCDA que para el caso del Arduino son los pines A4 y A5 respectivamente mientras que en el sensor VCNL4040 son el 7 y el 8.

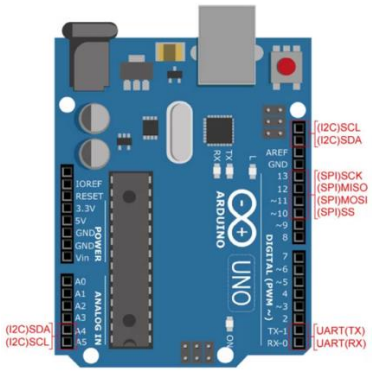


Figura 2. 49.- Interfaces de comunicación digital de la tarjeta Arduino UNO.

Para el caso del envío de información, Arduino cuenta con pines dedicados para la carga del programa a la memoria flash del dispositivo y que a su vez se pueden utilizar para cualquier otra comunicación UART. El problema de utilizar estos pines para comunicarse con el módulo HC05 se presenta durante la carga del programa ya que no se realizará hasta que se dejen de utilizar las terminales. Para evitar este problema se utiliza una librería que permite habilitar cualquier otro par de pines digitales para la comunicación UART, por lo que para este prototipo se habilitaron los pines 10 y 11 como TX y RX. De esta manera el UART0 utilizado por el propio Arduino quedan libres solo para ese propósito y la comunicación con el modulo *Bluetooth* se establece con pines diferentes.

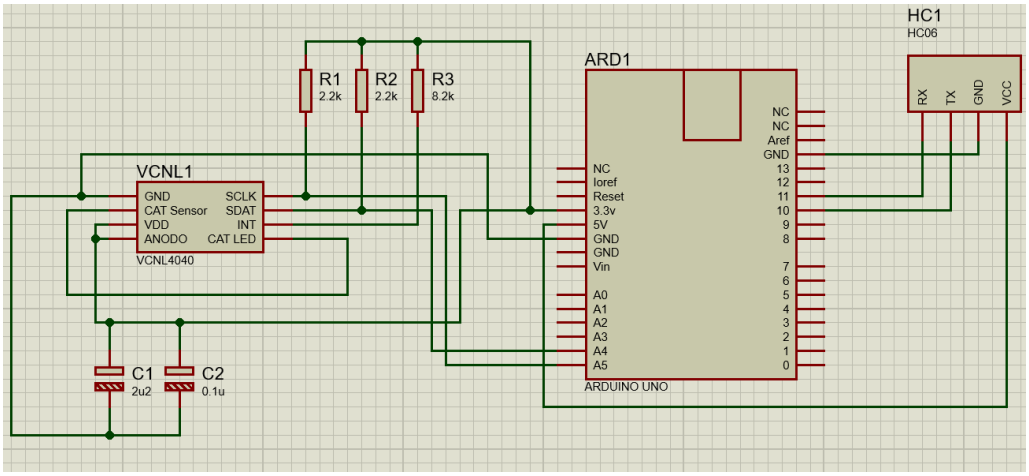


Figura 2. 50.- Diseño esquemático del circuito completo de medición de luz.

Debido que para el prototipo se utilizará un Arduino UNO el circuito que se diseña será a modo de un *SHIELD* para aprovechar el espacio y aprovechar lo mejor posible las conexiones tal como se muestra en la Figura 2. 50.

Una vez concretado el circuito se puede programar la rutina que seguirá el microcontrolador con ambos periféricos conectados y configurados.

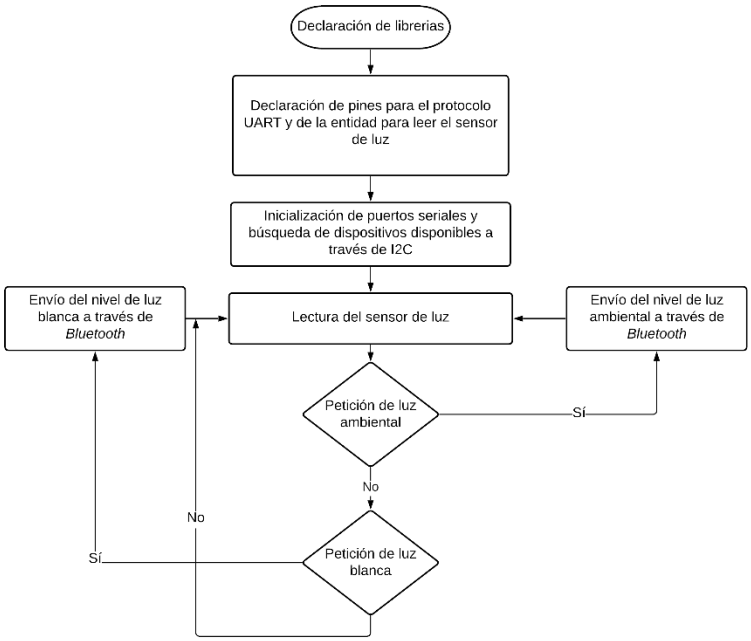


Figura 2. 51.- Funcionamiento esquemático del circuito de medición de luz.

2.12 Generación de la señal PWM para el convertidor *Buck*

De acuerdo a los cálculos obtenidos en el apartado 2.10 para obtener el valor nominal de la lámpara correspondiente a los 60V se necesita generar un ciclo de trabajo del 30% con una frecuencia de 100KHz para obtener un bajo valor de inductancia. El módulo *WiFi* utilizado es capaz de generar señales PWM variables a una frecuencia máxima de 1KHz cifra que no alcanza el requerimiento calculado, por lo que no es viable utilizar el generador interno PWM del propio módulo.

El módulo *WiFi* cuenta con un puerto de comunicación completo full dúplex que ocupa para cargar el programa a la memoria de la tarjeta y que también se utiliza para comunicarse con el módulo *bluetooth*. Se dice que es puerto completo por que puede haber comunicación bidireccional entre el módulo y el periférico conectado a través del UART0. También se cuenta con otro módulo UART, pero es *Half-Duplex*, es decir que solo puede enviar información por el pin de comunicación Tx. Este módulo corresponde al UART1 y se aprovecha para enviar información a la tarjeta que generará el ciclo de trabajo.

Una tarjeta con la que se cuenta y que tiene la capacidad de generar la señal con las especificaciones requeridas es la placa de desarrollo STM32L476RG de la compañía *ST Microelectronics*. Esta tarjeta es capaz de programarse desde el software propio de *ST* de nombre *CUBE* o también con el entorno de desarrollo de Arduino.

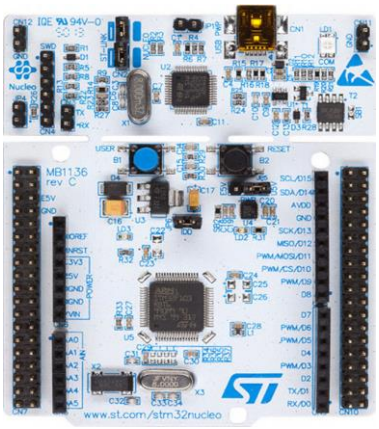


Figura 2. 52.- Tarjeta de desarrollo STM32L476RG.

La tarjeta mostrada en la Figura 2. 52 cuenta con diversos puertos seriales UART para la comunicación con el módulo *WiFi*. Adicionalmente contiene pines dedicados a generar señales PWM pudiendo alcanzar frecuencias de operación en el orden de los MHz siendo posible de esta manera lograr generar la señal a los 100KHz requeridos.

2.13 Configuración del sitio *WEB*

Como se mencionó en el capítulo 2, es necesario contar con un sitio *WEB* para poder tener comunicación a través de internet ya que no basta solo con tener conectividad *WiFi* en una tarjeta, sino que además hay que añadir direcciones válidas a las cuales comunicarse. Estas problemáticas están resueltas por diferentes compañías que venden alternativas *WiFi*, por ejemplo, Particle, quien además de vender el producto ofrece el uso de su propio servidor en la nube para una rápida programación. Esto permite comunicar la tarjeta a través de *internet*, no obstante, el usuario no sabe qué tan confidencial es la información que circula por el servidor ni la seguridad que tenga el sitio ante una saturación por el uso de los demás usuarios. Por ello la mejor alternativa es diseñar un sitio propio en el cual todo el flujo de información sea conocido.

Antes de comenzar a programar en un host montado en un servidor en internet, se considera buena práctica realizar pruebas con un servidor local para comprobar la comunicación con los dispositivos y descartar errores propios de un servidor en línea, por ejemplo, un mal acceso por errores en los usuarios, una mala configuración del servidor o disparidades en bases de datos. Si la comunicación de manera local funciona es muy seguro que al migrarlo a un servidor en línea funcione de igual manera ya que la configuración es muy parecida solo utilizando otro directorio alojado ahora en la nube.

Existen diferentes programas que permiten levantar un servidor de manera local, uno de los más conocidos e implementados es XAMPP. Este programa permite emular un servidor virtual en la máquina en donde se instale bajo el nombre de *LocalHost*. Este servidor responderá únicamente a peticiones locales las cuales sirven para realizar pruebas de comunicación entre la tarjeta y el *Host* Local y observar la lógica de comunicación.

Antes de poder activar el servidor se debe configurar y lo primero a realizar es abrir la interfaz del programa XAMPP desde donde se iniciará el funcionamiento de dos procesos. Apache es una rutina que se encarga de ejecutar un servidor virtual, mismo que administra y da respuesta a peticiones, mientras que MySQL se encarga de la comunicación con la base de datos.

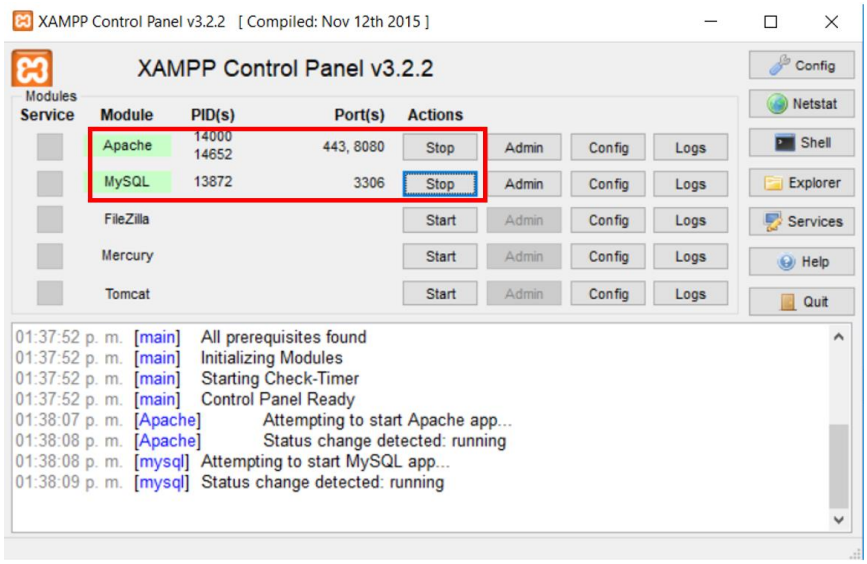


Figura 2. 53.- Panel de control de XAMPP con Apache y MySQL ejecutándose.

Una vez iniciadas ambos procesos, se puede acceder a los recursos para modificar o para agregar documentos. Para acceder a la creación de bases de datos se debe abrir el navegador e introducir la siguiente dirección: http://localhost:8080/phpmyadmin/. Es importante observar que a pesar de ingresar a través del navegador la comunicación se está llevando a cabo de manera local, prueba de ello es observar la dirección a la cual se está ingresando en donde se observa que se está accediendo al *Local Host* a través del puerto 8080.

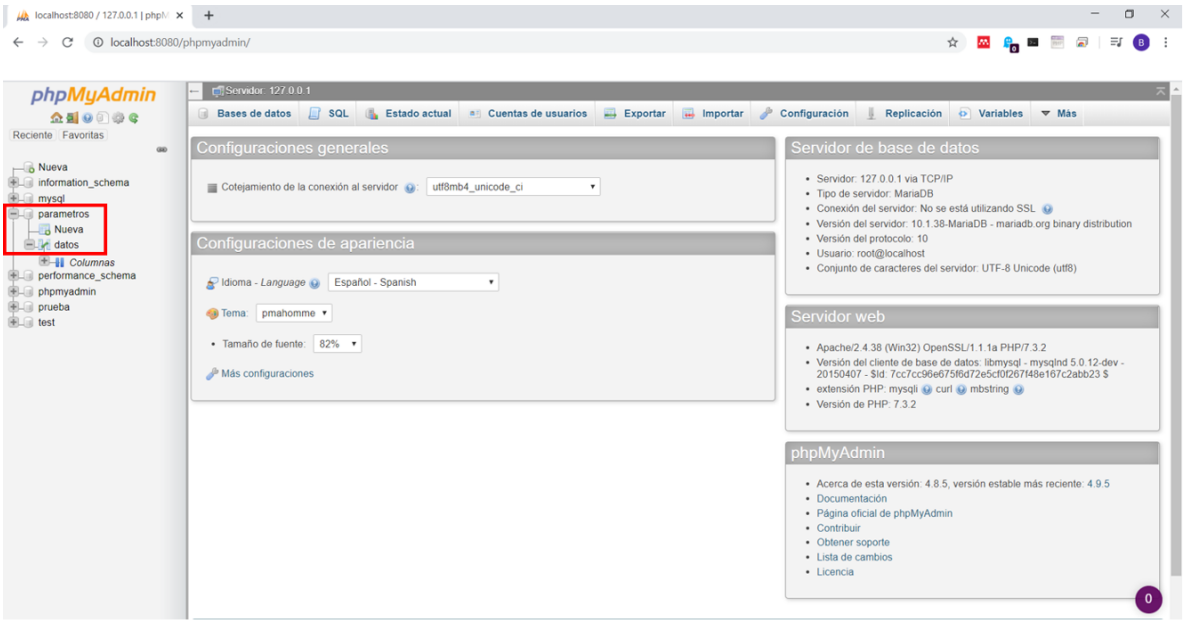


Figura 2. 54.- Gestión de archivos para bases de datos en XAMPP desde el navegador.

Desde la configuración del proyecto en el apartado *phpmyadmin* se pueden crear bases de datos para interactuar con la tarjeta *WiFi* y probar comunicación local con el intérprete de programación Python con el cual trabaja el módulo *WiFi*. Se observa que se creó una base de datos con el nombre *datos* la cual se ocupa para comunicarse con la tarjeta. Esta base de datos cuenta con dos campos que son estado y ciclo de trabajo.

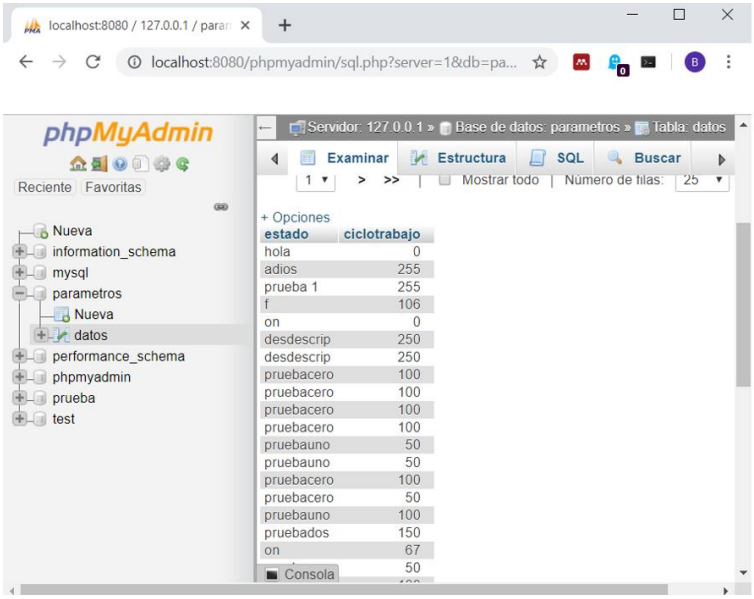


Figura 2. 55.- Base de datos creada de manera local y montada en servidor local.

A pesar que el servidor se encuentra funcionando correctamente y se puede acceder a sus directorios (como *PhpMyAdmin*), aún no es capaz de mostrar nada debido a que aún no se

cargan los archivos para visualizar la página *web*, es decir, una interfaz. Para poder cargar los archivos se debe ingresar al disco local y dirigirse a la carpeta Xampp y posteriormente entrar a la carpeta de *htdocs* en donde se colocará la carpeta con los archivos de la página web.

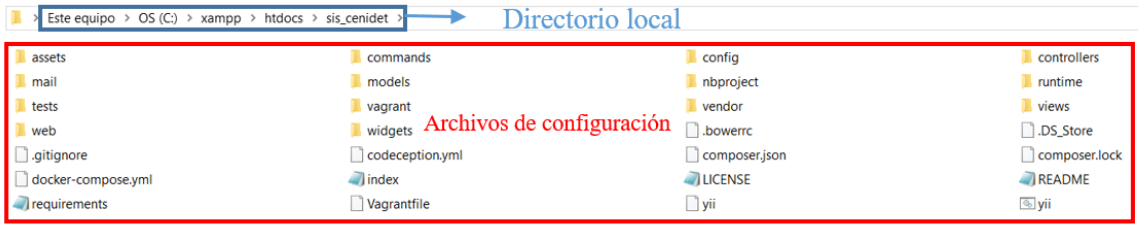


Figura 2. 56.- Archivos de configuración dentro de la carpeta sis_cenidet en htdocs.

Con los archivos cargados correctamente se puede ingresar la página principal que se programó como inicio. Para ingresar se utiliza la url *localhost:8080/sis_cenidet* la cual se puede apreciar es muy parecida a la utilizada en la Figura 2. 54 para ingresar a las bases de datos, sin embargo, finaliza con *sis_cenidet* que corresponde a la carpeta en donde se guardaron los archivos del sitio web en el directorio local.

Figura 2. 57.- Página de inicio del sitio web corriendo en servidor local.

Con los archivos colocados en la carpeta correspondiente y la base de datos creada se puede comenzar a realizar pruebas de comunicación. Al entrar al sitio se observa que la interfaz se despliega correctamente, lo cual significa que se realizó la carga de archivos exitosamente como se observa en la Figura 2. 57. Con esta prueba se puede comprobar que el *framework* que se utiliza funciona en un entorno local, por lo que es posible con el diseño de las siguientes pantallas.

Se puede probar que la base de datos funciona correctamente creando *scripts* en Python que realicen actualizaciones en las tablas de la base y que eliminen archivos.

```python
#Escritura de DB


import pymysql


#parametros de conexion al servidor
#host="mksgabat.online"
#user="datos"
#passwd="1994datos"
#database="datosonline"
#prueba a local localhost
host="localhost"
user="root"
passwd=""
database="parametros"



conexion=pymysql.connect(host,user,passwd,database)
cursor=conexion.cursor()



# Definir comandos para insertar registros
# Se define la tabla de la base de datos en este caso se llama "datos"
registro1= "INSERT INTO datos VALUES ('enlace1', '25');"
registro2= "INSERT INTO datos VALUES ('pruebahost', '100');"
registro3= "INSERT INTO datos VALUES ('lecturaexitosa', '120');"
cursor.execute(registro1)
cursor.execute(registro2)
cursor.execute(registro3)
conexion.commit()
conexion.close()
```

```python
#Consulta de DB

import pymysql
#parametros de conexion al servidor
host="mksgabat.online"
user="datos"
passwd="1994datos"
database="datosonline"


conexion=pymysql.connect(host,user,passwd,database)
cursor=conexion.cursor()

# Recuperar registros de la tabla 'parametros'
tablas = "SHOW TABLES;"
registros = "SELECT * FROM parametros;"

# Mostrar tablas
cursor.execute(tablas)
filas = cursor.fetchall()
print("Tablas de 'parametros':")
for fila in filas:
   print(fila)

# Mostrar registros
cursor.execute(registros)
filas = cursor.fetchall()
print("Registros de 'datos':")
for fila in filas:
   print(fila[0], ":", fila[1])

# Finalizar
conexion.commit()
conexion.close()
```

Figura 2. 58.- Programas para leer y escribir datos en la base de datos local con Python.

Los *scripts* de Python de la Figura 2. 58 son capaces de interactuar con la base de datos montada en el servidor. El de la izquierda tiene como función abrir la comunicación, realizar tres registros y cerrar el puerto. El de la derecha busca en la base de datos e imprime el contenido de todas las columnas. Se puede observar que los datos de acceso corresponden al servidor local, el mismo *script* tiene la capacidad de comunicarse también con sitios online modificando los datos de acceso.

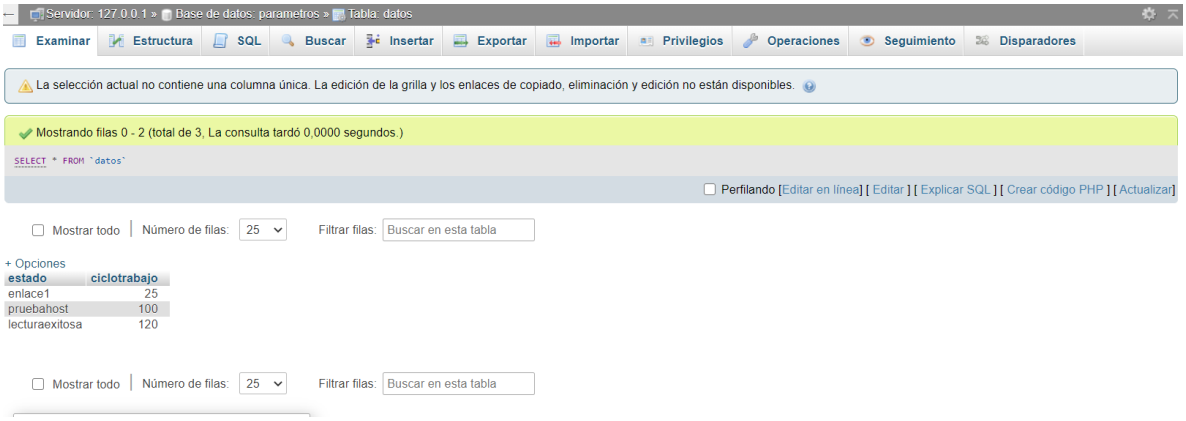


Figura 2. 59.-Registros guardados en la base de datos por el script de Python.

Se puede observar que existen tres registros y corresponden a los datos enviados con el programa hecho en Python, esto no solo quiere decir que funciona la interfaz, sino que además existe comunicación entre la base de datos y el servidor local, de tal manera que ahora el proyecto se puede migrar a un servidor funcionando en línea.

Para poder subir un sitio a *internet* se debe recordar el concepto de servidor, el cual no es más que un equipo de cómputo muy potente ejecutando un programa que permite la gestión de peticiones a través de una dirección IP pública asignada. Se puede configurar una computadora para habilitar un servidor y ponerlo en línea, sin embargo, el equipo se tendría que encontrar operando todo el día o al menos durante las pruebas. Configurar un servidor requiere conocimiento profundo acerca de Linux ya que la mayoría de servidores se basan en dicho sistema operativo, lo que dificulta elegir esta opción. Una alternativa es rentar un equipo a un proveedor que se dedique a ello, y una vez rentado se puede ocupar el equipo de acuerdo al paquete contratado. En el caso de este trabajo de investigación no se requiere de mucha potencia ya que el número de peticiones es bajo además que la cantidad de información que se intercambia también lo es por lo que la renta es la mejor opción.

Realizando una búsqueda sobre los diferentes proveedores de servidores en *internet* se encontró la empresa GoDaddy, que ofrece paquetes de desarrollo de sitios *web*, rentas de host y de dominios. La empresa ofrece un periodo de prueba lo que resulta útil para verificar si cumple el propósito al cual se requiere además que los precios de los productos son menores que la competencia.

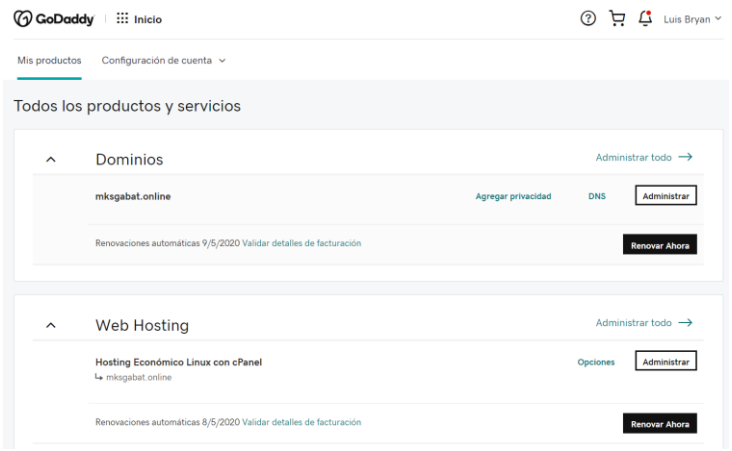


Figura 2. 60.- Host y dominio rentado en la empresa GoDaddy.

En la Figura 2. 60 se observan los productos contratados con la empresa GoDaddy los cuales son el *host* y el dominio. El host es necesario para para alojar el sitio web mientras que el dominio sirve para vincular el sitio con un nombre y facilitar la comunicación con el sitio *web*. Para el proyecto ya se contaba con un dominio por lo que no se requirió rentar otro, este tiene por nombre www.mksgabat.online.

Para poder ligar ambos contenidos se debe acceder a la cuenta de GoDaddy en donde automáticamente al contar con ambos productos ofrecerá la opción de ligarlos de tal manera que cuando termine el proceso, al ingresar el nombre del dominio en el buscador, se redirigirá al sitio *web* alojado en el servidor una vez cargados los archivos correspondientes.

Figura 2. 61.- Pantalla de inicio del sitio Web.

Se debe pensar que en la parte experimental se utilizará solo un sistema debido a la disponibilidad de lámparas y material, sin embargo, la interfaz de comunicación se diseñó para albergar más de un sistema, preparando el trabajo de investigación para trabajos futuros.

2.14 Diseño del sitio *WEB*

Para poder programar el funcionamiento del sitio *web* una vez configurado tanto el sitio como la cuenta en GoDaddy se utiliza el programa *NetBeans* para facilitar la creación los archivos de programación. Para poder utilizar el programa se debe realizar una configuración inicial para asignar los datos con los que ingresará al servidor.

La configuración inicia al abrir el programa y seguir la ruta: *Set configuration →Customize →Run configuration.* Una vez seleccionando la última opción de la ruta aparecerá una pantalla como la que se muestra a continuación.

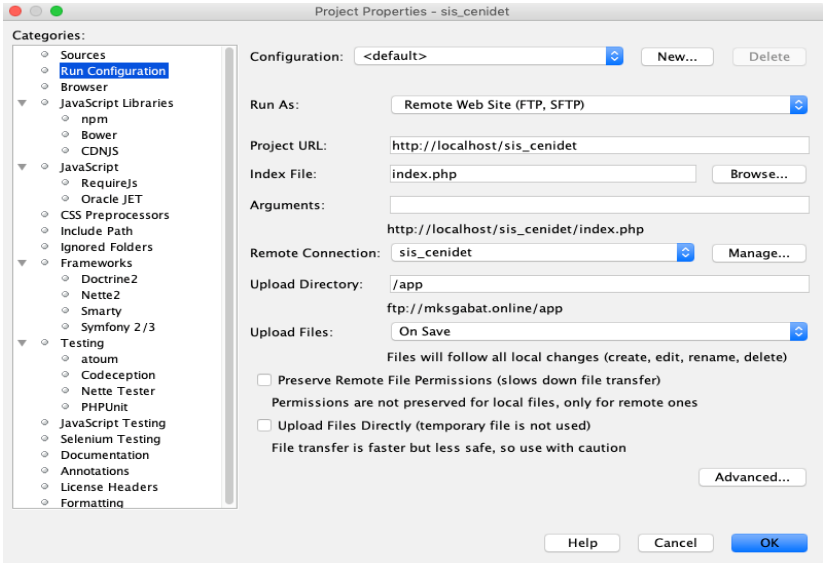


Figura 2. 62.- Pantalla de configuración de NetBeans.

En esta pantalla de carga se ingresan parámetros importantes como el tipo de conexión que se realiza FTP en este caso. Se indica en el campo *index File* que cree un archivo de nombre *index* con extensión php, en este archivo se generará la vista principal. Por último, se ingresa el nombre del archivo que contendrá la información del proyecto y los datos de acceso. Una vez llenados los campos como se muestra en la Figura 2. 62 se mostrará otra pantalla en la que se deberán ingresar los datos de acceso al sitio.

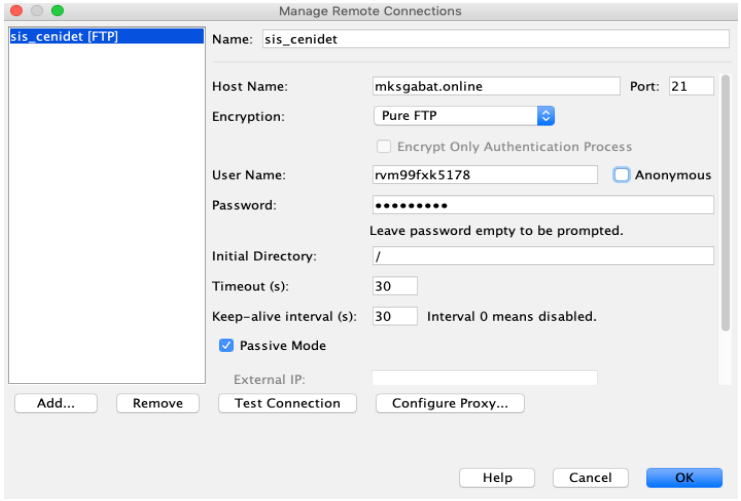


Figura 2. 63.- Pantalla de configuración remota en NetBeans.

En la siguiente pantalla se ingresan los datos del servidor en línea como el nombre, el protocolo, el usuario y contraseña del sitio, así como los parámetros de sincronización. Cuando se ingresan los datos es recomendable realizar una prueba de conexión para verificar que se realice correctamente la conexión presionando el botón *Test Connection.* Si los datos son correctos debe aparecer un mensaje indicando que la conexión fue exitosa.

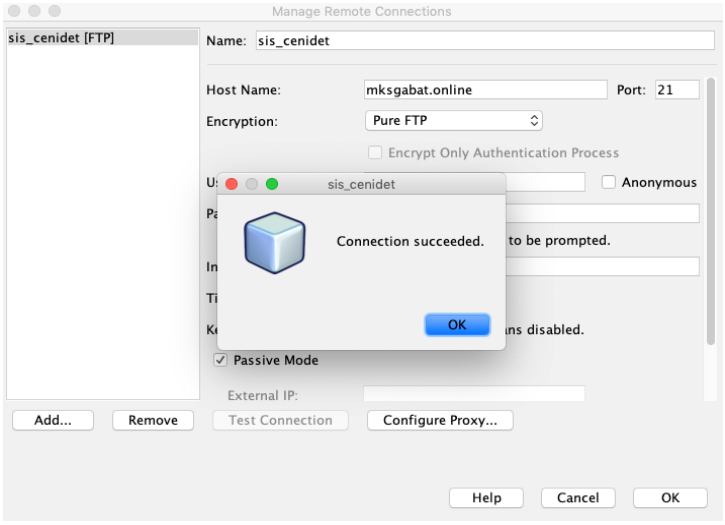


Figura 2. 64.- Mensaje de conexión exitosa.

Una vez validada la comunicación se debe configurar la base de datos, para ello se debe realizar un diagrama conocido como Entidad-Relación. En ella se aprecia la cantidad de tablas que se necesitarán, así como qué relación tiene cada tabla entre sí. A partir de este diagrama se generará el código para crear la base en donde se almacenarán las variables del sitio web.

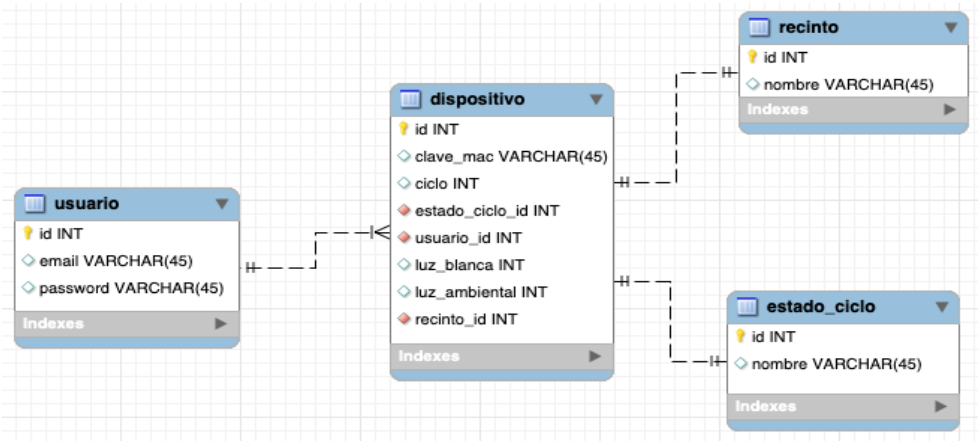


Figura 2. 65.- Diagrama Entidad-Relación de la base de datos.

Se puede observar que 4 tablas principales son las que se requieren para el funcionamiento del sistema. La tabla llamada *usuario* almacenará el correo electrónico y la contraseña de los usuarios para ingresar al sistema. Esta tabla se conecta con la llamada *dispositivo*, en ella se almacenarán los datos del dispositivo como: la clave MAC, el ciclo de trabajo, así como los valores de luz blanca y luz ambiental. Esta tabla se conecta con la tabla *estado_ciclo* donde guarda si la lámpara debe estar encendida o apagada. Por último, la tabla recinto almacenará los estándares de iluminación y los recintos a los cuales se enfoque el sistema.

Cuando se realiza la configuración de los servicios antes mencionados el proyecto muestra una estructura debido al framework que se está trabajando. Este framework genera un proyecto con diferentes carpetas en donde se almacenarán los archivos del programa, así como carpetas que necesita el propio programa para funcionar.

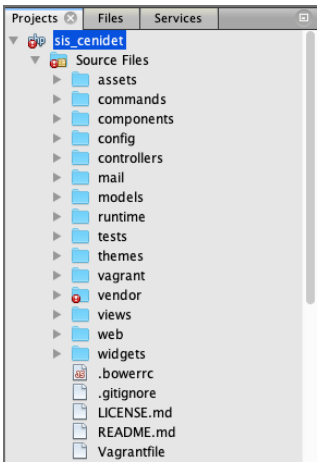


Figura 2. 66.- Estructura del proyecto almacenado en el servidor de GoDaddy.

Es en estas carpetas se deberán colocar los archivos para crear el sitio web. Existen datos que se deben incluir en el proyecto para tener acceso al sistema por ejemplo el acceso a la base de datos. Se necesita un usuario y contraseña validados en el servidor para que se permitan realizar cambios en la base de datos como medida de seguridad para evitar cambios no autorizados. En la carpeta *config* se debe crear el archivo *db.php* el cual contiene los datos de acceso a la base de datos.

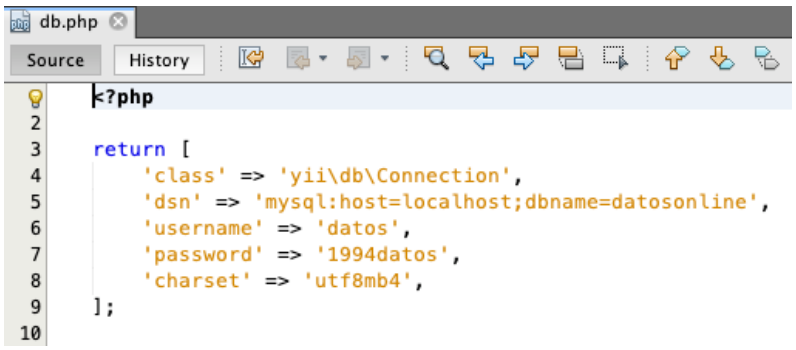


Figura 2. 67.-Datos de conexión a la base de datos en el servidor.

Se debe indicar además de los datos de acceso la dirección en donde se encuentra dicha base de datos, para esto es necesario crear un archivo llamado *params.php* el cual contiene esta información.

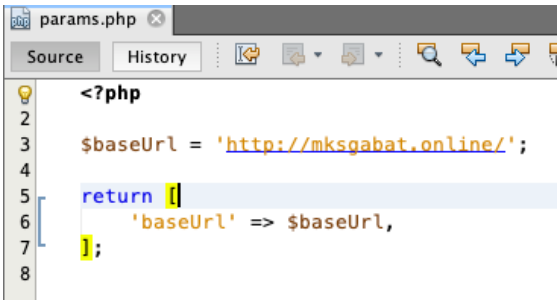


Figura 2. 68.- Dirección de la base de datos.

Un archivo más es necesario en el cual se integren los archivos que se generaron, este tiene por nombre *web.php*. En este archivo se asignan los datos recabados en una rutina la cual se encargará de realizar la comunicación a través de internet.

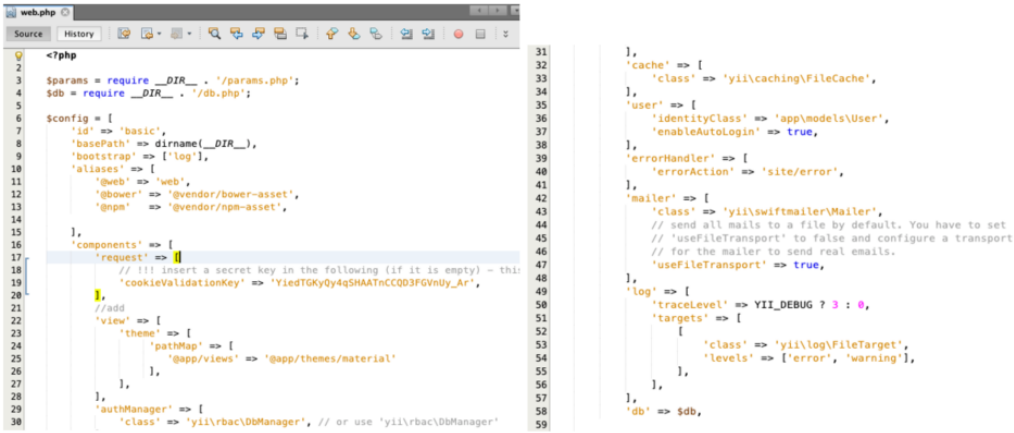


Figura 2. 69.- Código contenido en el archivo web.php.

Se debe recordar que el *framework* trabaja en un esquema MVC por lo que es necesario generar código en cada una de estas carpetas e iniciar con el diseño propiamente de la interfaz.

Los modelos son las carpetas que contienen las instrucciones para acceder a información, por ejemplo, la base de datos. Los modelos contienen la información necesaria para acceder a las tablas y realizar funciones como actualizaciones, o cambios de datos en la base. También en los modelos se puede encontrar los atributos de algunas funciones. En la carpeta *Models* encuentran los modelos: *Dispositivo*, *EstadoCiclo*, *Recinto* y *usuario*, además de modelos que se generan por default.

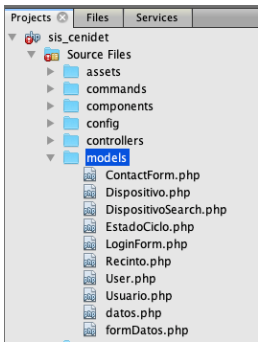


Figura 2. 70.- Modelos existentes en el proyecto.

En la carpeta *views* se pueden encontrar las vistas del sitio web. En esta carpeta se almacenan inicialmente las pantallas con las que cuenta el sitio Web sin embargo para un mejor manejo de las vistas se trasladaron a la carpeta *themes* para la posibilidad de agregar un mejor diseño en trabajos futuros.

Los controladores se encuentran en la carpeta *controllers*, estos archivos contienen la lógica en general y es donde se construyen las rutinas de código encargadas de registrar información, validar campos, gestionar peticiones, entre otras cosas.

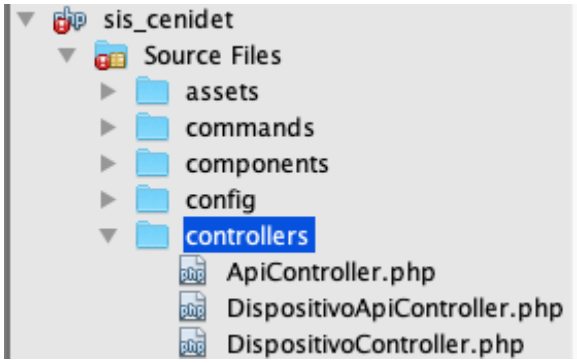


Figura 2. 71.- Controladores necesarios para el proyecto.

La Figura 2. 71 muestra tres controladores de los cuales se ocupan dos. El llamado *DispositivoApiController.php* contiene las funciones necesarias para comunicarse con la tarjeta, para recibir los datos que se reciben como parámetro en la petición *POST*, así como otra que retorna el ciclo de trabajo y el estado. Por otro lado *DispositivoController.php* se encarga de manejar el contenido de las vistas, como formularios, tablas, actualizaciones y en general que las interfaces de usuario funcionen correctamente.

Es importante mencionar que el controlador *DispositivoApiController.php* no tiene vistas y solo se ocupa como medio de comunicación entre el sitio y la tarjeta. En este controlador se encuentra la función responsable del intercambio de datos bajo el nombre de InfoDispositivo la cual se puede apreciar en la URL a la que se comunica la tarjeta.

```php
public function actionInfoDispositivo() {
    $post = Yii::$app->request->post();
    //System::beautyPrint($post);
    $respuesta = [];
    $estatus = 0;
    $mensaje = "";
    $info = $post['{"informacion'];
    $auxData = rtrim($info, "}");
    $data = $auxData.'}';
    $arrayData = explode(' ',$data);

    $arrayId = explode(',',$arrayData[2]);
    $id = $arrayId[0];

    $arrayLuzBlanca = explode(',',$arrayData[4]);
    $luzBlanca = $arrayLuzBlanca[0];

    $arrayLuzAmbiental = explode('}',$arrayData[6]);
    $luzAmbiental = $arrayLuzAmbiental[0];

    $idDispositivo= $id;
    //$clave_mac = $info['clave_mac'];
    $dispositivo = Dispositivo::find()->where(['id'=>$idDispositivo])->one();
    if(isset($dispositivo)) {
        $estatus = 1;
        $mensaje = "Dispositivo Encontrado";
        $dispositivo->luz_blanca = $luzBlanca;
        $dispositivo->luz_ambiental = $luzAmbiental;
        if ($dispositivo->save()) {
            //->select('ciclo,estado_ciclo_id')
            $result = Dispositivo::find()->where(['id'=>$idDispositivo])->asArray()->one();
        } else {
          System::beautyPrint($dispositivo->getErrors());
        }
    } else {
        $mensaje = "Error con el dispositivo";
        System::beautyPrint("Error con el dispositivo");
    }


    $respuesta["dispositivo_actualizado"] = $result;

    return JSON::encode($respuesta);
}
```

Figura 2. 72.- Función de comunicación con la tarjeta llamada InfoDispositivos.

En el proyecto existen además de las carpetas mencionadas otras que se generan automáticamente como parte del *firmware* de Yii2, estas carpetas permiten que funcione el proyecto correctamente además de permitir mejoras en trabajos futuros.

2.15 Carga del *firmware* de MicroPython

Se ha mencionado anteriormente las ventajas que tiene programar en MicroPython, sin embargo, no todas las tarjetas cuentan con el *firmware* por defecto en la tarjeta y en caso de contenerlo, es recomendable actualizarlo en su última versión para asegurar contar con la versión más estable posible, este proceso en su mayoría debe cargarse manualmente.

Existen distintas maneras de modificar el *firmware* de una tarjeta, sin embargo, lo más eficiente es mediante el uso de la terminal debido a que al no contar con una interfaz responde de mejor manera a los comandos. Existen diferentes versiones del *firmware* en el repositorio GitHub diseñados para distintas tarjetas y todos trabajan de forma similar. En el presente proyecto se cargó el *firmware* de MicroPython en módulos como ESP01, ESP12, Feather Huzza, NodeMCU los cuales toman como base el *chip* ESP8266 resultando en una programación idéntica en donde lo único que cambió es el tamaño del archivo. Se puede gestionar los archivos de los módulos *WiFi* desde la terminal de cualquier sistema operativo

solo cuidando utilizar los comandos correspondientes. En este trabajo de investigación se utilizó el sistema operativo Windows 10, el proceso debe ser parecido en otros sistemas operativos modificando ajustando los comandos correspondientes. Los pasos a seguir para cargar el *firmware* se muestran detallados en [12].

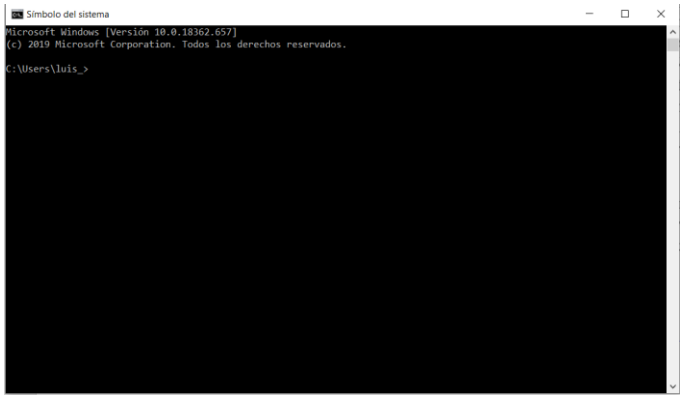


Figura 2. 73.- Consola de comandos en Windows.

Este medio permite gestionar los archivos del sistema de una manera más potente ya que, en casos en donde la interfaz gráfica no responde, la consola de comandos si puede. Los comandos más comunes en la consola se presentan en la Tabla 2. 5.

Tabla 2. 5.-Comandos principales de la consola de Windows.

Nombre del comando	Función del comando
help	Muestra los comandos disponibles e información sobre algún comando en específico.
attrib	Muestra los atributos de un archivo.
cd	Accede al directorio indicado
cd..	Retrocede al directorio superior
cls	Limpia la terminal
chkdsk	Realiza un chequeo del disco duro
del “carpeta”	Elimina la carpeta indicada
dir	Muestra el contenido de un directorio
Md “carpeta”	Crea nueva carpeta

Como primer paso, se debe descargar el *firmware* que se instalará el cual se puede encontrar en *micropython.org/download#esp8266* en donde a partir de la capacidad de memoria con la que cuenta la tarjeta se elegirá el archivo adecuado. A la par de contar con el *firmware* a cargar en la tarjeta, se utiliza una herramienta llamada *esptool* la cual sirve para gestionar la memoria de los módulos *WiFi*, esta herramienta se puede descargar desde el enlace *https://github.com/espressif/esptool* o directamente desde la consola utilizando el comando *pip install esptool* desde donde se descargará el archivo y se instalará automáticamente.

Para poder cargar el *firmware* en la tarjeta es necesario conocer en qué puerto se encuentra conectado el módulo. Para verificar los periféricos de entrada conectados a la computadora

se puede abrir el administrador de dispositivos el cual mostrara los dispositivos conectados y el puerto por el cual se están comunicando.

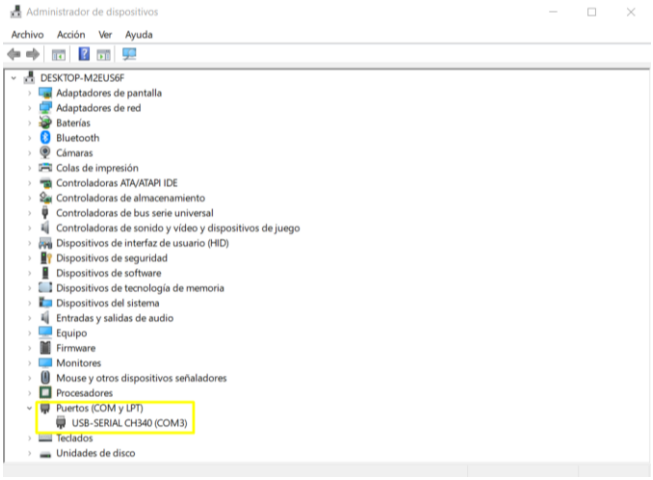


Figura 2. 74.- Localización del puerto COM desde el administrador de dispositivos.

Cuando ya se cuenta con el *firmware*, el archivo esptool y se conoce el puerto al que está conectado el módulo, se debe acceder al símbolo de sistema desde el mismo directorio en donde se encuentren ambos archivos y desde ahí ejecutar el siguiente comando: *esptool.py –port COM3 erase_flash.* Este comando borrará el contenido de la memoria que tenga precargada la tarjeta, liberando el espacio para el nuevo *firmware* de MicroPython. Si la comunicación es correcta y el comando se escribió correctamente la consola borrará la memoria y mostrará una secuencia como se observa en la Figura 2. 75.

Figura 2. 75.- Proceso de borrado de la memoria ROM del módulo WiFi.

Una vez que la memoria se libera es momento de cargar el nuevo archivo para lo cual se utiliza la siguiente instrucción.: *esptool.py --port COM3 --baud 460800 write_flash --flash_size=detect 0 esp8266-20170108-v1 8.7.bin.* Esta instrucción escribe el *firmware* en la memoria del *chip* especificando el puerto, la velocidad y el nombre del archivo.

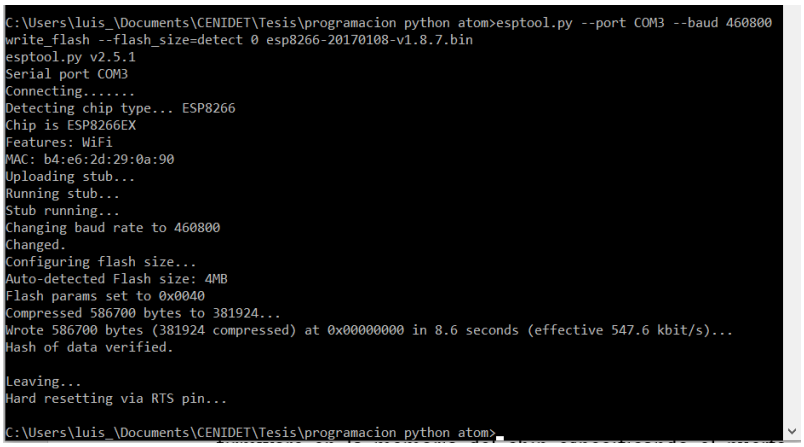


Figura 2. 76.- Proceso de carga del firmware de MicroPython en las tarjetas ESP8266.

Cuando la consola termine de arrojar resultados y vuelva a aceptar comandos significa que el proceso de carga finalizó exitosamente y es posible comunicarse de manera serial mediante un terminal serial, por ejemplo, Putty. En cualquier terminal que se utilice se deben configurar los parámetros como: puerto de comunicación, la velocidad en Baudios, la paridad y el control de flujo, como se muestra en la siguiente imagen.

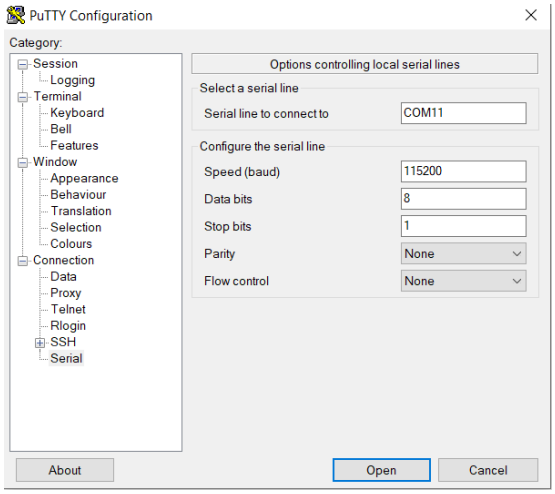


Figura 2. 77.- Configuración básica de la terminal de comunicación serial.

Cuando se ejecuta el terminal serial con los parámetros configurados se debe observar un texto de inicio mostrando la versión de MicroPython que está ejecutando, la fecha y el módulo ESP8266 seguido de tres símbolos >>> lo que indica que se está ejecutando una versión de Python sobre la tarjeta. Si no se llegaran a ver los tres símbolos significa que no se cargó correctamente el programa por lo que se debe repetir el proceso cuidando bien los pasos hasta que se muestre una pantalla como se observa en la siguiente figura.

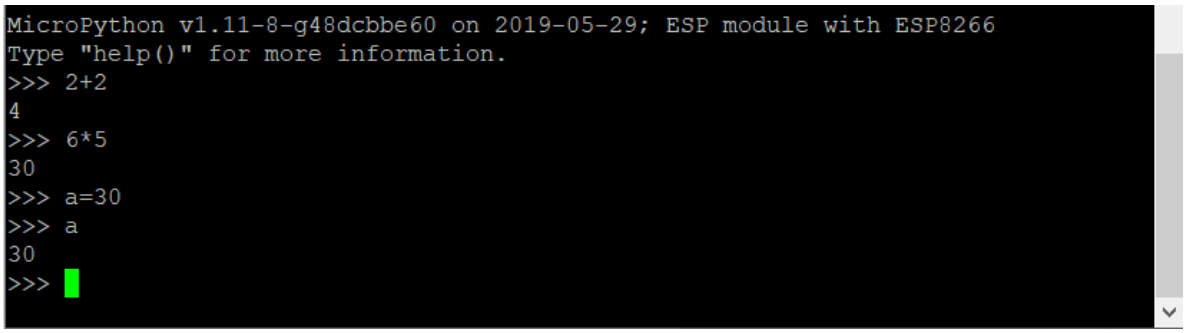


Figura 2. 78.-Comunicación serial por terminal Putty en tarjeta ESP12.

Al comunicarse con la tarjeta desde un terminal serial es posible ejecutar comandos línea por línea interpretándolos al momento de pulsar la tecla enter, similar a trabajar en una terminal Python en una computadora. Adicionalmente es posible crear scripts y ejecutarlos desde la terminal REPL o grabarlos en la memoria del microcontrolador para ejecutarlo posteriormente al reiniciar el circuito.

Los módulos están diseñados para ejecutar dos programas diferentes los cuales tienen por nombre *boot.py* y *main.py*. El primer archivo que ejecutará será el *boot.py* quien contiene la configuración inicial de la tarjeta y una vez terminado ejecuta el *main.py* en donde se contiene la secuencia de instrucciones principales. Estos archivos se pueden generar desde cualquier IDE que genere archivos con la extensión .py que corresponde a Python, y se graban también en la tarjeta mediante una herramienta en consola.

Para cargar los archivos programados a la memoria de la tarjeta se utiliza nuevamente la consola utilizando una herramienta llamada ampy la cual se puede obtener en el enlace *https://github.com/adafruit/ampy*. Esta herramienta cuenta con diferentes instrucciones que permiten leer el directorio de archivos en la memoria y recuperar, borrar o almacenar archivos desde la memoria de la tarjeta, por ejemplo, para grabar el programa en la tarjeta se utiliza la siguiente instrucción: *ampy --port COM3 put boot.py* mientras que si se requiere saber que archivos hay en la tarjeta se utiliza*: ampy --port COM3 ls.* Se puede apreciar que se debe especificar en la instrucción el puerto a través del cual se encuentra conectada la tarjeta, así como el nombre del archivo en caso de realizar una modificación al directorio de la memoria.

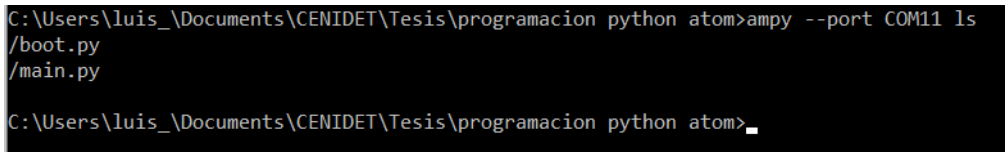


Figura 2. 79.- Ejemplo del uso de la herramienta ampy en la tarjeta ESP-12.

Con las herramientas antes mencionadas y con las configuraciones realizadas se puede iniciar con la programación de la tarjeta que se seleccione. Es un proceso largo y complicado en un inicio, sin embargo, una vez realizado se pueden programar tarjetas de bajo costo utilizando

el intérprete de alto nivel Python con el que se puede lograr funciones y prestaciones que comparables a productos más costosos de grandes marcas comerciales.

2.16 Programación de la tarjeta Feather Huzzah ESP8266

El circuito que se implementará junto al convertidor tiene que ser capaz de comunicar información para la generación del ciclo de trabajo necesario para el convertidor. Debe ser capaz de comunicarse a través de internet para recuperar parámetros de ajuste por el usuario, así como recibir por *Bluetooth* el nivel de iluminación obtenido por el circuito de medición de luz.

Para lograr las funciones principales se utiliza un módulo *WiFi* basado en el *chip* ESP8266 desarrollado por Adafruit llamado Feather Huzzah. Este *chip* se conecta a su vez mediante UART a un HC05 que se utilizara para la comunicación *Bluetooth* como se mostró en la Figura 2. 45.

La secuencia de programación inicia con la declaración de las librerías necesarias para el manejo de pines y la comunicación inalámbrica. Se debe recordar desactivar el modo REPL para poder enviar y recibir datos a través del puerto UART 0 y no se presenten interferencias o pérdidas de datos. Se inicializan instancias como pines de entrada, salida, velocidad del UART, y frecuencia de operación de la señal PWM.

Una vez realizada la configuración se utiliza el puerto UART 0 para enviar la petición de luz ambiental y luz blanca mediante caracteres homologados en ambas tarjetas, al recibir dicha clave la otra tarjeta responderá con el dato correspondiente. Estos datos llegan al host para que se almacenen en la base de datos y se presenten en la interfaz de usuario, no obstante, no es posible enviar los datos al servidor tal cual se reciben ya que es necesario enviarlo en un formato especial y adjuntarlos como parámetros de una petición *POST* al URL habilitado para almacenar los datos. Para darle el formato correcto se agregan los datos a una variable de tipo diccionario y posteriormente convertirlos a formato JSON con ayuda de una librería del mismo nombre.

El sitio *web* se programó para recibir una petición de tipo POST desde el módulo *WiFi*, en la cual se adjuntan una serie de datos incluyendo el ID del dispositivo. Si el sitio web encuentra el dispositivo registrado accederá a los valores establecidos correspondientes al dispositivo y los regresará como respuesta a la petición encapsulados en una variable de tipo JSON la cual recibirá el microcontrolador la procesará y utilizará para ajustar el nivel de ciclo de trabajo que aplicará al convertidor.

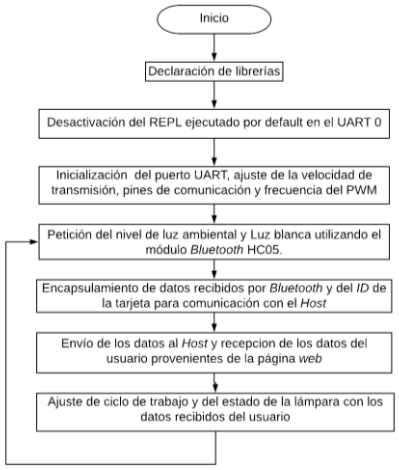


Figura 2. 80.- Rutina que sigue el controlador de la lámpara con conectividad WiFi y Bluetooth.

El flujo de instrucciones que sigue el controlador de la lámpara LED se ilustra de manera resumida en la Figura 2. 80.

Todos los módulos que integrarán el proyecto se han descrito a lo largo del presente capítulo, así como la relación y comunicación que tendrán entre ellos. A continuación, se presenta de manera gráfica la información que provee cada módulo, así como el flujo que sigue la información para lograr las funcionalidades descritas en la propuesta de solución.

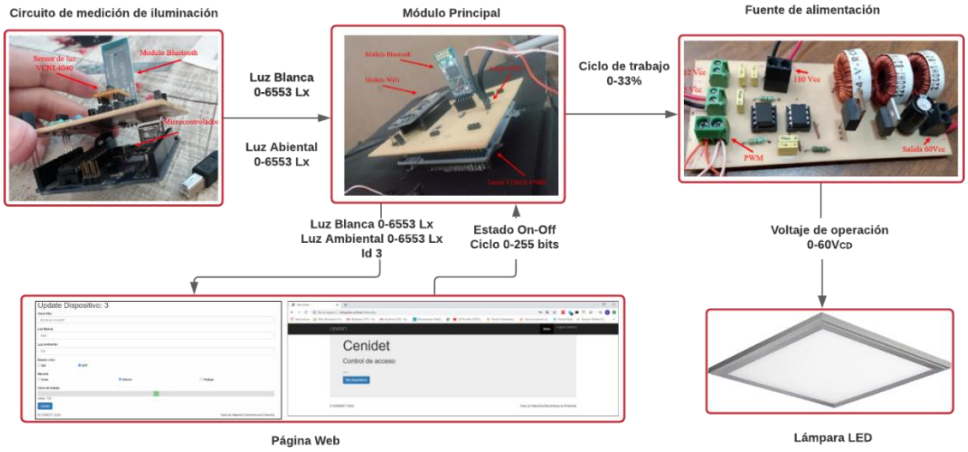


Figura 2. 81.-Integración de los diferentes módulos en el sistema.

La figura anterior muestra el flujo de datos que sigue el sistema, por ejemplo, el circuito de medición de iluminación transmite en nivel de luxes que obtiene tanto de luz blanca como de luz ambiental y a través del módulo principal manda los datos al sitio web. El sitio retorna el estado que debe tener la lámpara (encendida o apagada) y el ciclo de trabajo en un rango de 8 bits, la procesa y entrega al convertidor un ciclo de trabajo de 0 a 33% para generar a la salida del convertidor un voltaje de 0 a 60 V_{CD}. Se puede apreciar que toda la información pasa por el módulo principal, razón por la cual se asignó el nombre de “principal”.

Capítulo III

Pruebas y resultados

3.1 Descripción del sistema

Para que el sistema funcione como se propuso en los primeros capítulos son necesarias diferentes etapas las cuales se pueden separar con el fin de explicar de una mejor manera en que contribuyen individualmente para después explicar cómo funciona todo el sistema en general.

3.1.1 Fuente de alimentación

Se construyó el convertidor siguiendo el diagrama mostrado en la Figura 2. 39 con el fin de poder acoplar las tierras tanto del convertidor como de los impulsores MIC4451. La placa del convertidor también contiene opto acopladores para aislar las señales de control con la potencia.

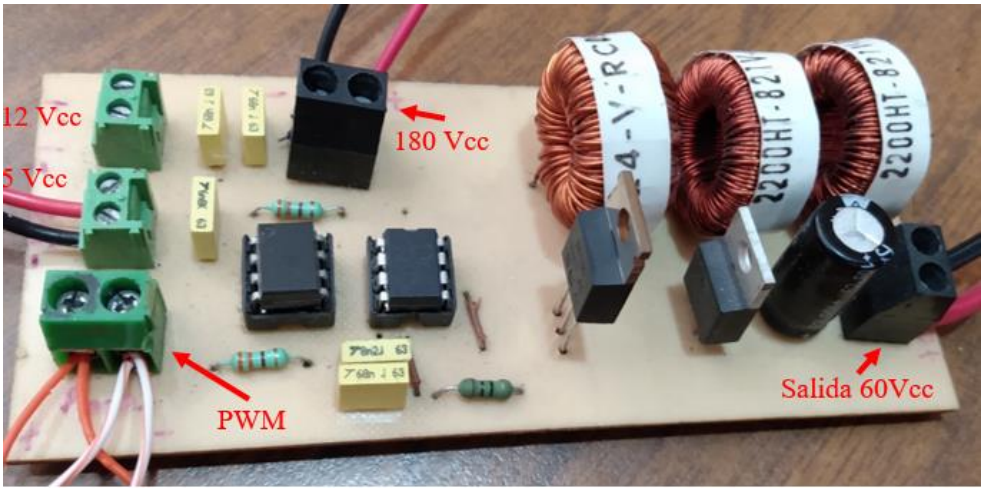


Figura 3. 1.- Placa de alimentación de la lámpara con el convertidor Buck, impulsor para el MOSFET y aislamiento óptico.

La tarjeta mostrada en la figura anterior muestra de lado izquierdo las entradas de alimentación para hacer funcionar el opto acoplador para aislar la señal PWM y los impulsores MIC4451 encargados de amplificar la señal[28]. Cuenta con la entrada para conectar el bus de CD mismo que utilizará el convertidor BUCK para generar la tensión con lo que se alimenta la lámpara.

De acuerdo al cálculo que se realizó en el capítulo anterior, cuando el convertidor recibe un ciclo de trabajo del 30% la tensión de salida debe tener un valor nominal de 60 V_{CD} teniendo

como entrada 180 V_{CD} proveniente del voltaje de línea después de la etapa de rectificación. Estos valores son coherentes con los observados durante las pruebas experimentales las cuales se muestran en la Figura 3. 2.

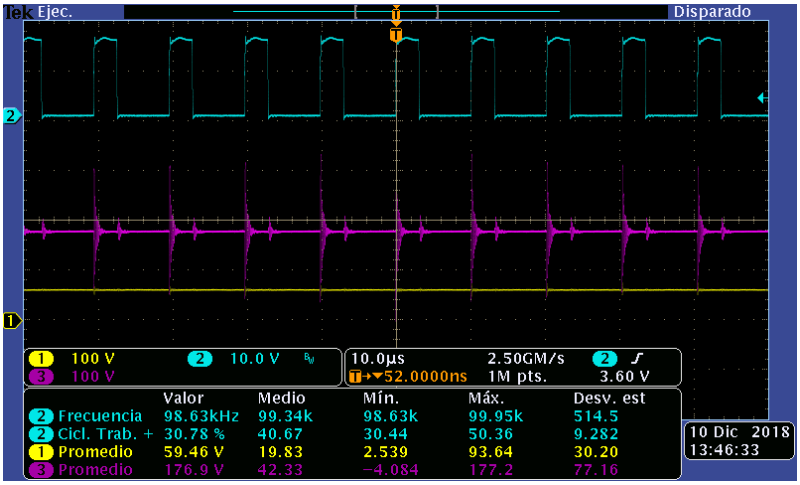


Figura 3. 2.- Voltaje se salida (señal en amarillo), ciclo de trabajo (señal en azul) y voltaje de entrada (señal en morado) del convertidor Buck.

Al aplicar esa forma de onda a la lámpara de leds se puede comprobar que enciende correctamente e incluso responde a variaciones de ciclo de trabajo atenuando la iluminación que entrega tal cual se muestra en la siguiente figura.

Figura 3. 3.- Convertidor funcionando con la carga de prueba.

Debido a que los valores observados en la práctica y los calculados coinciden, se puede validar que el convertidor opera satisfactoriamente, sin embargo, se debe recordar que el nivel de iluminación va a cambiar a lo largo del día, en función de la cantidad de iluminación que incida en el recinto. Estas pruebas se muestran en apartados posteriores.

3.1.2 Módulo principal

El módulo principal es el circuito encargado de comunicarse con el sitio web, con el sensor de iluminación y de generar el ancho de pulso correspondiente con la información recopilada entre ambas comunicaciones. Estas funciones precisan de los tres circuitos que se muestran, el módulo *WiFi*, el módulo *Bluetooth* y la tarjeta STM32L476RG.

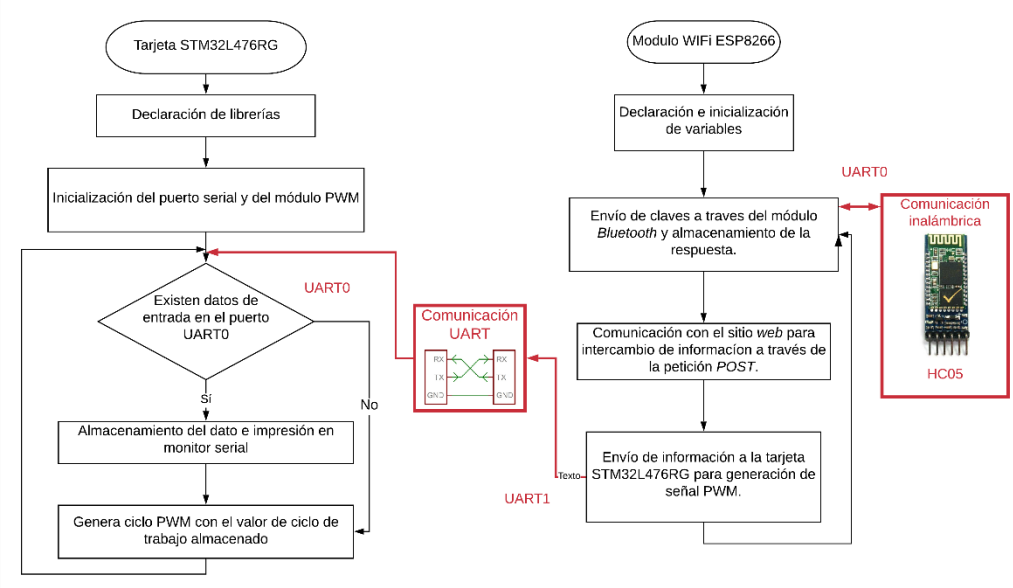


Figura 3. 4.-Flujo de información del módulo principal.

Como se había explicado anteriormente el módulo principal depende de circuitos independientes que trabajan conjuntamente para lograr la funcionalidad completa. En el esquema presentado en la Figura 3. 4 se presentan los flujos de instrucciones que siguen las tarjetas así como la intercomunicación que tienen entre ellas señalado de color rojo. El módulo *WiFi* es el que concentra la información y se apoya de la tarjeta de ST Electronics para generar el ancho de pulso misma que sigue una rutina independiente.

Los circuitos se interconectaron, así como a los periféricos necesarios en un circuito impreso mostrado en la Figura 3. 5. Esta tarjeta ofrece conexiones más cortas para evitar el ruido radiado por consecuencia de los diferentes estándares de comunicación utilizados.

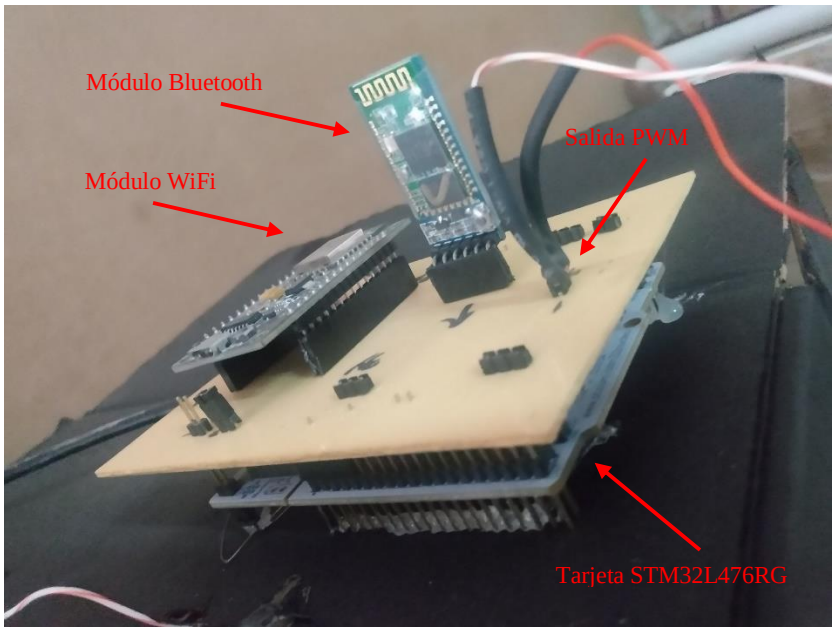


Figura 3. 5.-Módulo principal con comunicación Bluetooth y WiFi conectados a la tarjeta STM32L472RG.

Este circuito impreso integra el módulo *WiFi* basado en el chip ESP8266 para el intercambio de información con el sitio *Web* en donde se envía el valor de iluminación recibido a través de *Bluetooth* a la par de recibir los datos establecidos por el usuario en la interfaz en línea. La comunicación *Bluetooth* se logra mediante el Módulo HC05 el cual se conecta directamente al UART0 del módulo *WiFi* y de manera inalámbrica al HC06 contenido en el módulo de medición de iluminación. A través del puente inalámbrico *Bluetooth* es que el módulo principal obtiene niveles de iluminación del sensor ubicado en otra parte del recinto, los procesa y los envía tanto al sitio *web* como a la tarjeta STM conectada al UART1 del módulo *WiFi*.

La tarjeta STM32L476RG recibe los datos de iluminación y del sitio *WEB* para compararlos y generar una señal PWM de acuerdo a las condiciones en el recinto. Si hay más iluminación de la que se requiere el ancho de pulso disminuirá continuamente hasta apagar la lámpara, de lo contrario el ancho de pulso aumentará hasta un máximo del 30%.

3.1.3 Módulo de medición de iluminación

El sistema es capaz de determinar la iluminación en el entorno a través del módulo de medición de iluminación el cual basa su funcionamiento en el sensor VCNL4040 y un microcontrolador. El microcontrolador que se utiliza es Arduino para facilitar la lectura y transmisión de la información que obtiene el sensor, misma que se comparte con el módulo principal a través de *Bluetooth*.

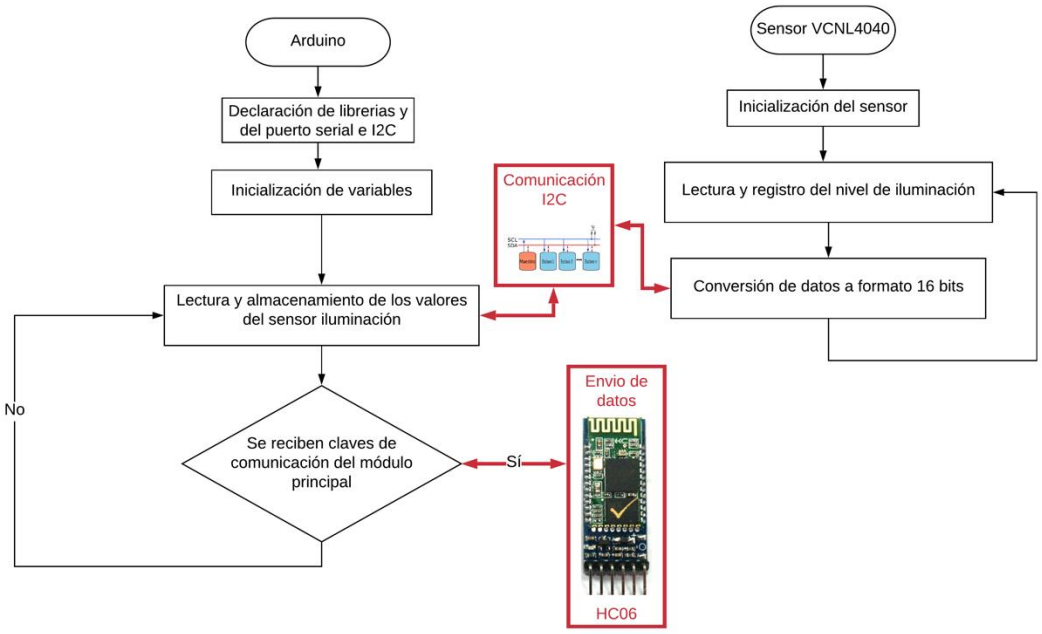


Figura 3. 6.-Flujo de información de las tarjetas que componen al módulo de medición de iluminación.

Como se aprecia en el diagrama de lado derecho correspondiente al sensor de iluminación el cual una vez se inicializa obtiene el nivel de iluminación constantemente y lo mantiene disponible para compartirlo en cuanto un dispositivo lo solicite. La comunicación entre tarjetas se muestra nuevamente en color rojo, se aprecia que la intercomunicación entre tarjeta se da a través de I2C y luego a través de UART con el módulo HC06 quien envía información inalámbrica al módulo principal mediante el módulo HC05.

En la rutina del microcontrolador mostrada de lado izquierdo de la Figura 3. 6, se observa el funcionamiento principal de la tarjeta. Después de inicializar la tarjeta entra en un bucle donde constantemente se comunica con el sensor de iluminación y espera la petición de datos por parte del módulo principal a través de comunicación *Bluetooth*.

Se diseñó un circuito impreso que contiene el sensor de luz VCNL4040, el microcontrolador Arduino y el modulo complementario *Bluetooth* HC06 para la comunicación inalámbrica *Bluetooth*, así como los periféricos necesarios para la comunicación I2C.

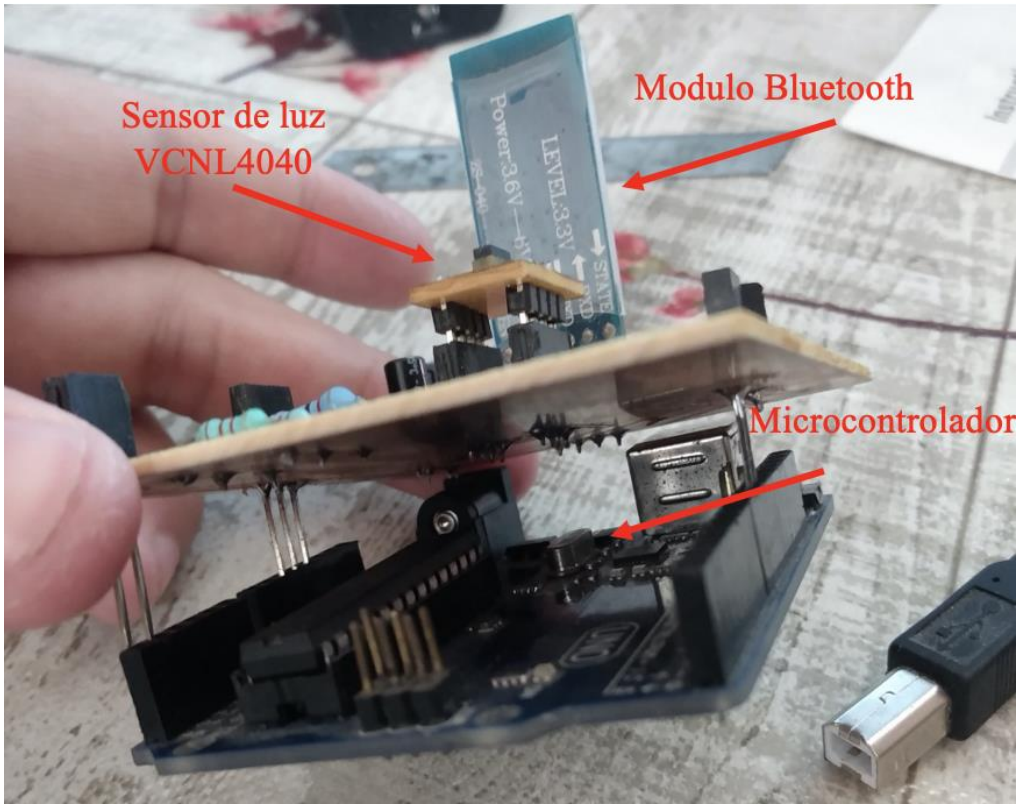


Figura 3. 7.- Circuito de medición de iluminación.

En la figura anterior se muestran los dispositivos en el circuito impreso y las conexiones entre los dispositivos tanto de alimentación como de comunicación. Los módulos incluidos se encuentran configurados para que solo sea necesario alimentar con 5V para que la tarjeta funcione.

Este circuito impreso se diseñó para colocarse en un punto estratégico en donde se desee medir la incidencia luminosa en el recinto. Una vez colocado, el valor de iluminación que llega al módulo HC06 es enviado al HC05 conectado al módulo principal y recibido finalmente por el módulo *WiFi*.

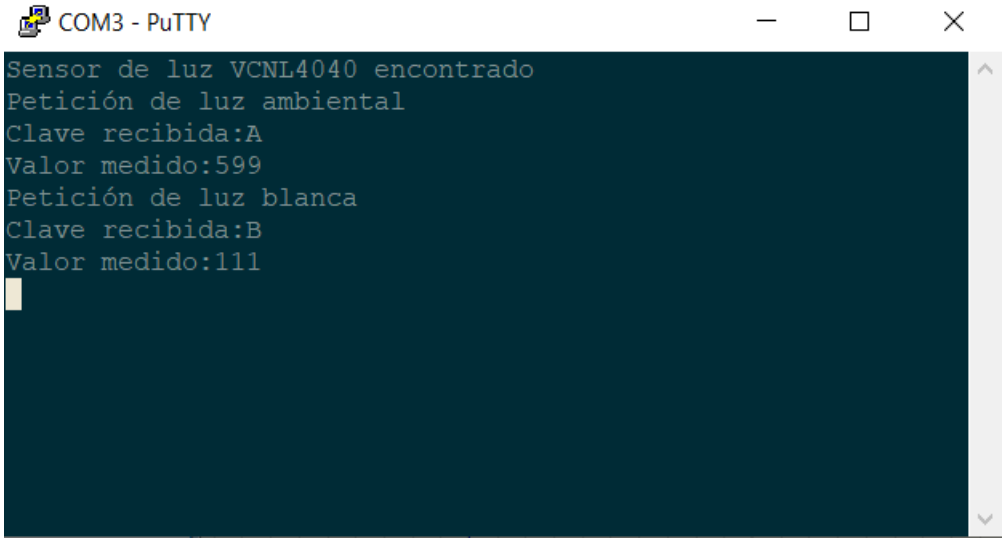


Figura 3. 8.- Comunicación del microcontrolador con el sensor VCNL4040.

En la figura anterior se observa que se obtuvo el nivel de iluminación por parte del sensor VCNL4040 a la vez que se tiene comunicación con el módulo HC06. De este último circuito se reciben la clave *A* para enviar el valor de luz ambiental, mientras que si se recibe la clave *B* se envía el nivel registrado de luz blanca.

3.1.4 Pruebas de conexión *Bluetooth*

Para que la información se pueda transmitir desde el sensor de iluminación hasta el módulo principal de manera inalámbrica basta con enlazar el módulo HC05, conectado al módulo principal, con el HC06 ubicado en el circuito de medición de iluminación. Para ello se deben seguir los pasos mencionados en el anteriormente donde se detallan los comandos utilizados.

Se pueden utilizar herramientas que permiten visualizar los datos con la finalidad de verificar la información que se envía y concretamente el formato con el que se envía, ya que algunas tecnologías, al realizar el envío, modifican el tipo de dato para facilitar su trasmisión. Es importante saber exactamente el formato de la información se envía ya que en algunos casos se debe especificar qué tipo de dato se envía, tal cual se hace con la comunicación con el sitio web.

Para verificar la comunicación del lado de la PC y realizar depuraciones se suele utilizar un FTDI para que la computadora se pueda comunicar con el módulo *Bluetooth* a través de USB detectándolo como un puerto COM. Una vez detectado es posible comunicarse con el módulo mediante un terminal serial como *TeraTerm* o *Putty*.

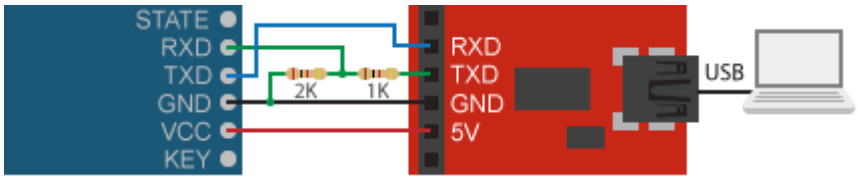


Figura 3. 9.- Conexión del módulo HC05 con un FTDI para comunicación USB.

Durante las pruebas el terminal serial que incluye el IDE de Arduino es muy útil debido a que se puede escribir una cadena de caracteres completa antes de enviarla a diferencia de las otras terminales las cuales van enviando los datos caracter por caracter tan pronto son ingresados por el teclado. Otra razón más de utilizar esta terminal es que al momento de escribir cadenas largas quedan guardadas en un espacio de memoria y se puede reescribir con tan solo presionar la tecla de dirección de arriba.

Con el FTDI y el terminal serial se puede observar la información que transmite el módulo *Bluetooth*, pero es deseable poder ver el otro lado de la comunicación. Para ello bastaría con conectar otro FTDI a la contraparte *Bluetooth* para que se observe mediante otra ventana del terminal serial la comunicación sin embargo pudiera haber dudas sobre la comunicación inalámbrica. Para ello en lugar del par *Bluetooth* se puede utilizar el que tiene integrado un *smartphone* y observar el intercambio de datos instalando una aplicación, *BlueTerm* en este caso.

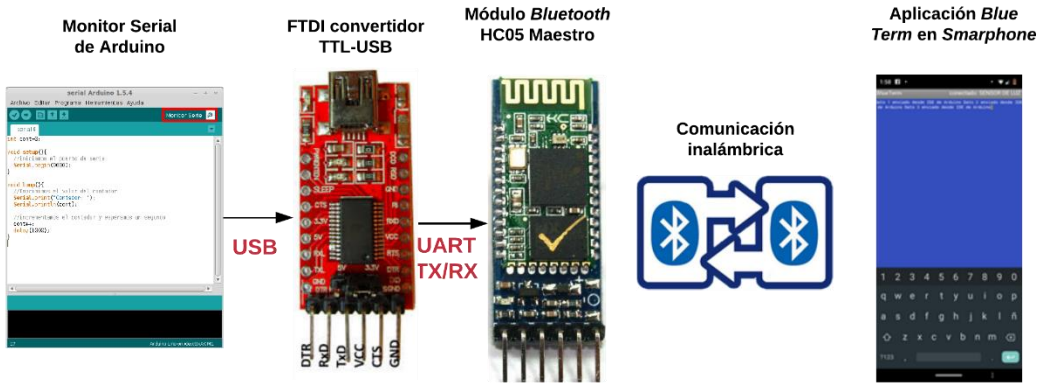


Figura 3. 10.- Flujo de información para la prueba del módulo maestro HC05.

Una vez enlazada la comnicación *Bluetooth* tanto en la terminal de Arduino como en la aplicación de BlueTerm es posible observar el intercambio de información *Bluetooth*. Desde ambas terminales la comunicación es bidireccional, esto significa que lo que se envíe a través de la terminal de Arduino se recibira en BlueTerm y viceversa.

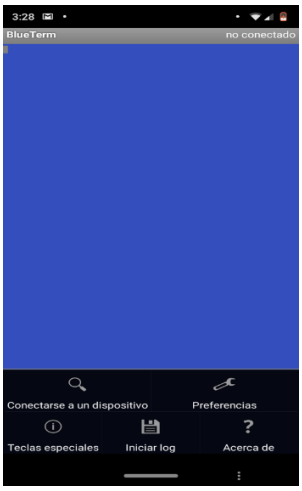


Figura 3. 11.- Interfaz de la aplicación BlueTerm en un Smartphone.

Para utilizar la aplicación *BlueTerm* se debe realizar una configuración inicial la cual consiste solo en el emparejamiento con el *Bluetooth* que se desea monitorear. La pantalla de inicio que se muestra tan pronto abre la aplicación es la que se presenta en la Figura 3. 11. En esta pantalla se encuentra la opción *"Conectarse a un dispositivo"* lo cual permite visualizar los dispositivos *Bluetooth* disponibles a los cual el teléfono puede conectarse.

Figura 3. 12.- Lista de dispositivos disponibles en la aplicación BlueTerm.

En la lista aparecerá el circuito sensor de luz con el nombre que se haya configurado tan pronto se encienda el circuito tal como se muestra en la Figura 3. 12. Para realizar la conexión entre el teléfono celular y el HC05 se debe seleccionar el nombre del bluetooth e introducir el pin que se configuró. Si el pin es correcto se realiza el enlace inalámbrico, en ese momento el led integrado en el circuito *Bluetooth* comenzará a parpadear con una menor frecuencia indicando que se encuentra en el modo de conexión inalámbrica.

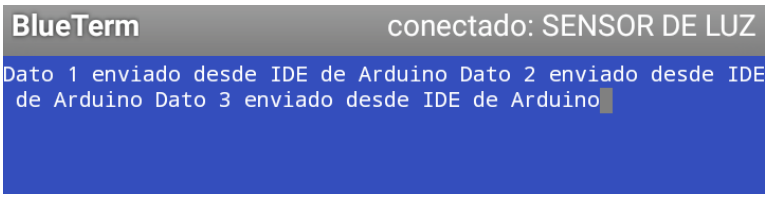


Figura 3. 13.-Información enviada desde la terminal de Arduino y vista en un Smartphone.

La aplicación BlueTerm muestra los datos en el formato original que lo recibe sin agregar caracteres especiales, por lo que la información que se observa puede parecer desordenada o confusa. El objetivo de observar la información es verificar que tipo de dato se envía y con qué tipo de dato se recibe para evitar pérdidas de información.

El terminal serial de Arduino tiene la opción de observar los datos de manera ordenada mediante saltos de línea al finalizar la recepción. Si por el contrario se requiere ver el formato original se debe seleccionar “Sin ajuste de línea” dentro del terminal para que se muestre la información sin agregar caracteres especiales.

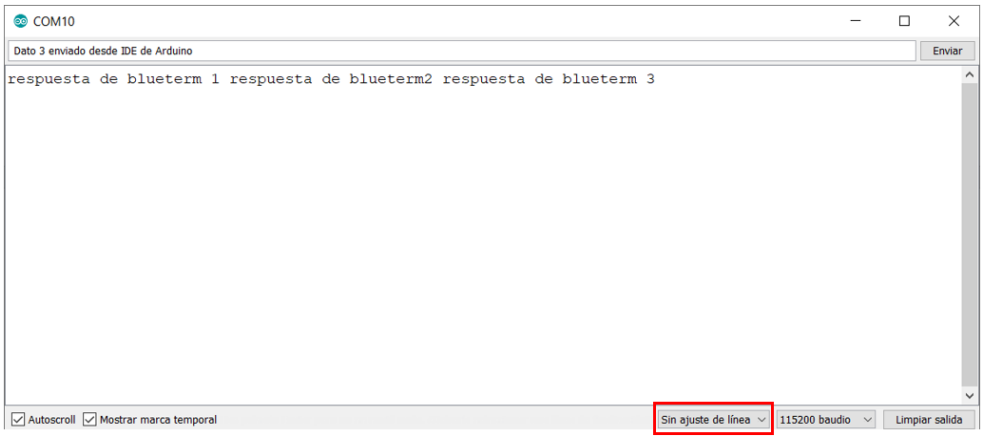


Figura 3. 14.- Información enviada desde un Smartphone y vista en la terminal de Arduino.

De esta manera se comprueba que la comunicación es exitosa y se comprobó que la manera de mandar múltiples datos es mediante un arreglo tipo JSON ya que es más fácil interpretarlo en el sitio *web* y todas las tarjetas pueden generarlo e interpretarlo.

Figura 3. 15.- Prueba de comunicación al servidor.

En la Figura 3. 15 se muestra una prueba realizada con los datos enviados entre los módulos *Bluetooth* que deben llegar al servidor. Como se puede ver el servidor acepta los datos y devuelve una respuesta que contiene los mismos datos enviados, pero en formato JSON para corroborar que durante la transmisión de datos no existen pérdidas.

3.2 Pruebas de comunicación

La comunicación local se encuentra establecida a través del cableado de los circuitos y del enlace inalámbrico de los módulos *Bluetooth*, pero aún falta la comunicación hacia el sitio *web* a través de *WiFi*. Para la comunicación se colocaron los archivos de programación en las carpetas correspondientes tanto para habilitar las rutinas de comunicación con la tarjeta como para mostrar las diferentes pantallas que corresponden a la interfaz de usuario en el sitio *web*.

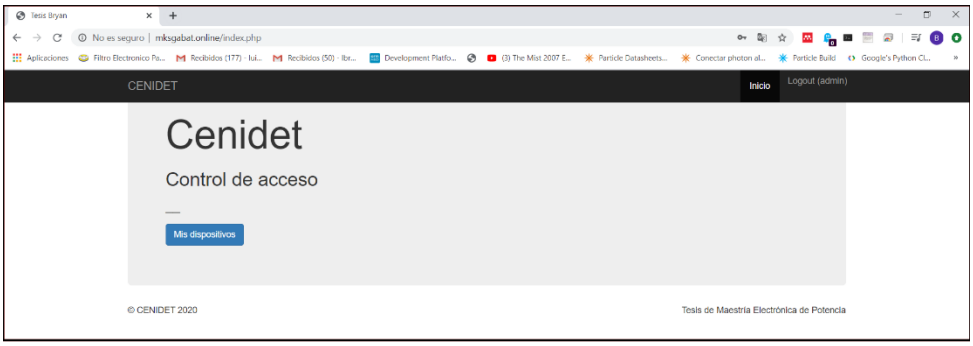


Figura 3. 16.- Página de inicio del sitio web..

Debido a que la interfaz se diseñó para ser utilizadas por múltiples usuarios se debe tener el control de inicio de sesión para identificar a cada usuario y sus dispositivos vinculados. Atendiendo ese motivo se debe registrar cada usuario y una vez registrado iniciar sesión para poder acceder a las pantallas de dispositivos.

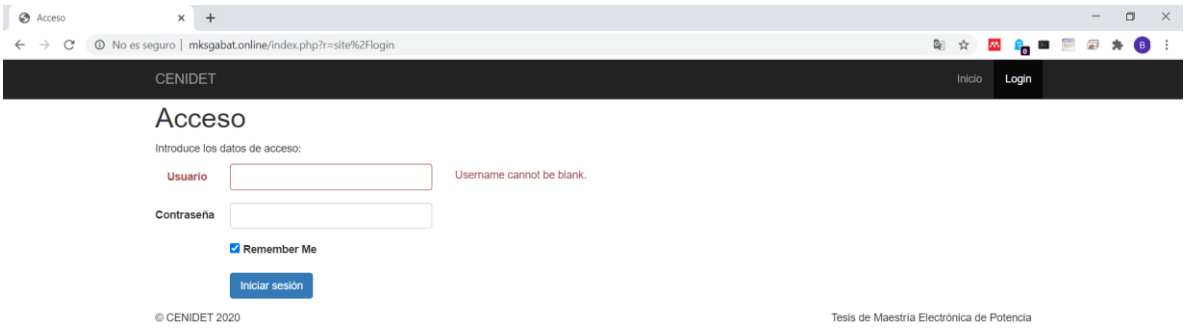


Figura 3. 17.- Pantalla de acceso al sistema.

Es la pantalla mostrada en la Figura 3. 17 se ingresa al sistema introduciendo un usuario y una contraseña. Para este proyecto se registró un usuario de nombre *admin* cuya contraseña es *password*. Estos datos suelen ser los utilizados por defecto para realizar pruebas y depurar el sitio web.

La tarjeta Feather Huzzah se comunica con el sitio web a través de una petición POST en donde envía como parámetro su ID asignado junto con otros datos los cuales almacena el sitio web en la base de datos. Estos datos se pueden visualizar en la pantalla que se muestra en la siguiente figura la cual se muestra una vez iniciando sesión en la aplicación.

#	id	Clave Mac	Ciclo	usuario_id	estado_ciclo_id	
1	3	00:1B:44:11:3A:B7	122	1	OFF	
2	2	01:8B:44:11:A3:B7	120	1	ON	
3	1	11:3F:44:1F:B2:F5	10	1	ON	

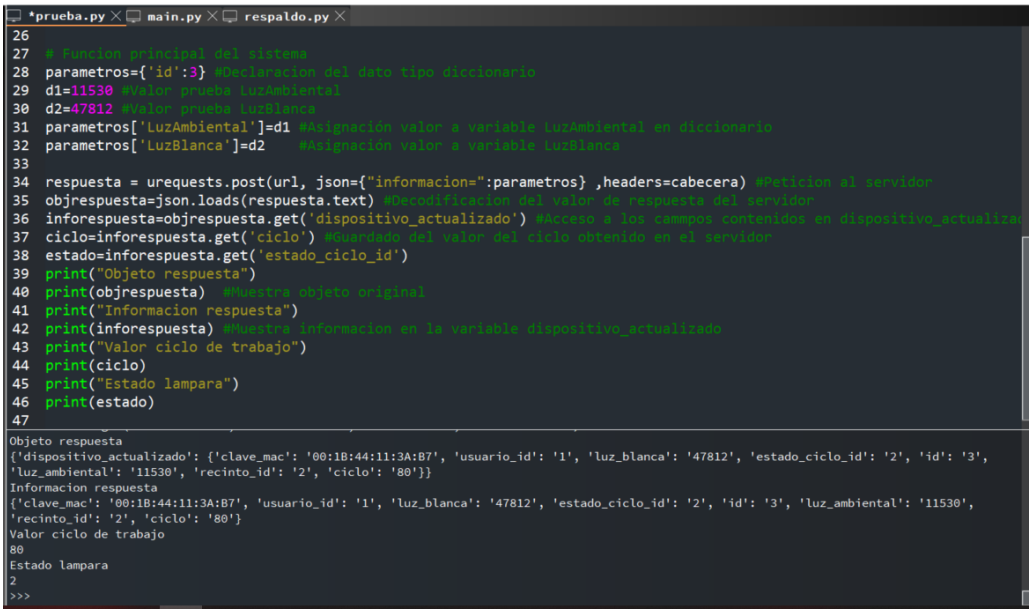
Figura 3. 18.- Tabla de dispositivos registrados en el sitio web..

En la Figura 3. 18 se muestran tres dispositivos, dos de los cuales corresponden a tarjetas cuyo programación consiste en realizar peticiones consecutivas con el fin de probar que el sistema puede responder a múltiples peticiones simultáneamente. A pesar que se tiene limitado el número de elementos para armar fuentes de alimentación para controlar iluminación, el sitio *web* está preparado para manejar una cantidad elevada de dispositivos comunicándose continuamente.

El dispositivo registrado con el ID número 3 corresponde al módulo *WiFi* conectado al convertidor y es quien tiene comunicación con el sitio. El usuario ingresa sus preferencias a través del botón *actualizar* cuyo icono tiene forma de lápiz y está ubicado de lado derecho de la tabla. Al dar clic se muestra otra pantalla en donde el usuario modifica parámetros que se almacenan en la base de datos la cual consulta el módulo cada que envía una petición POST.

Figura 3. 19.- Vista correspondiente a la actualización de los parámetros de un dispositivo.

Cada vez que el usuario realiza cambios y presiona el botón *actualizar* la información se actualiza en la base de datos. La tarjeta accede a los datos de la base mediante la petición POST al mismo tiempo de entregar el nivel de iluminación que recibe del sensor de iluminación.



```
*prueba.py   main.py   respaldo.py
26
27  # Funcion principal del sistema
28  parametros={'id':3} #Declaracion del dato tipo diccionario
29  d1=11530 #Valor prueba LuzAmbiental
30  d2=47812 #Valor prueba LuzBlanca
31  parametros['LuzAmbiental']=d1 #Asignación valor a variable LuzAmbiental en diccionario
32  parametros['LuzBlanca']=d2    #Asignación valor a variable LuzBlanca
33
34  respuesta = urequests.post(url, json={"informacion=":parametros} ,headers=cabecera) #Peticion al servidor
35  objrespuesta=json.loads(respuesta.text) #Decodificacion del valor de respuesta del servidor
36  inforespuesta=objrespuesta.get('dispositivo_actualizado') #Acceso a los cammpos contenidos en dispositivo_actualizado
37  ciclo=inforespuesta.get('ciclo') #Guardado del valor del ciclo obtenido en el servidor
38  estado=inforespuesta.get('estado_ciclo_id')
39  print("Objeto respuesta")
40  print(objrespuesta)  #Muestra objeto original
41  print("Informacion respuesta")
42  print(inforespuesta) #Muestra informacion en la variable dispositivo_actualizado
43  print("Valor ciclo de trabajo")
44  print(ciclo)
45  print("Estado lampara")
46  print(estado)
47
Objeto respuesta
{'dispositivo_actualizado': {'clave_mac': '00:1B:44:11:3A:B7', 'usuario_id': '1', 'luz_blanca': '47812', 'estado_ciclo_id': '2', 'id': '3',
'luz_ambiental': '11530', 'recinto_id': '2', 'ciclo': '80'}}
Informacion respuesta
{'clave_mac': '00:1B:44:11:3A:B7', 'usuario_id': '1', 'luz_blanca': '47812', 'estado_ciclo_id': '2', 'id': '3', 'luz_ambiental': '11530',
'recinto_id': '2', 'ciclo': '80'}
Valor ciclo de trabajo
80
Estado lampara
2
>>>
```

Figura 3. 20.- Segmento del código para la comunicación online con el sitio Web.

El segmento de código mostrado en la figura se utiliza para inicializar puertos UART, declarar variables, así como preparar y enviar la petición al servidor. Se puede observar que se declara el ID del dispositivo, la url del api de comunicación y las funciones que decodifican el archivo que se retorna.

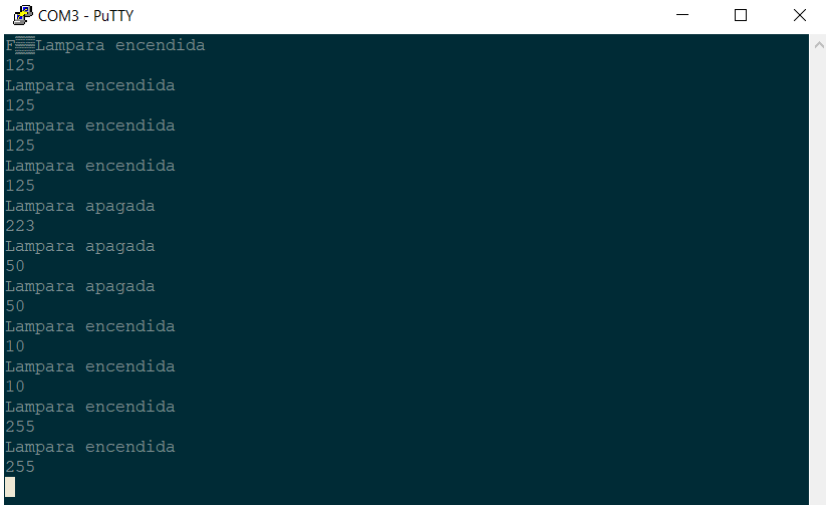


Figura 3. 21.- Respuesta a la prueba de comunicación con el host.

La Figura 3. 21 muestra el comportamiento de la tarjeta y el resultado de las peticiones hechas con el script de la Figura 3. 20. La tarjeta imprime en el puerto serial el estado de la lámpara y el valor del ciclo de trabajo, el cual, si se modifica en la página web, se ve reflejado en la terminal debido a que cada cierto tiempo la tarjeta consulta el valor de los parámetros mediante una petición POST validando la comunicación de la tarjeta con el sitio web.

Con estas comunicaciones se concluyen las pruebas de comunicación por lo que se cargan los programas definitivos y se realizan las intercomunicaciones físicas para observar el funcionamiento de todos los circuitos en conjunto.

3.3 Sistema completo

En los apartados anteriores se explican las partes del sistema y la manera en que contribuyen al funcionamiento en general del mismo. En este apartado se muestran las pruebas en conjunto y todos los circuitos funcionando simultáneamente, el intercambio de datos, el ajuste del ciclo de trabajo al mismo tiempo que la iluminación, así como los resultados obtenidos.

La Figura 3. 22 muestra los tres elementos necesarios para la alimentación de la lámpara: el rectificador, el convertidor y el módulo principal. El rectificador toma los $127V_{CA}$ de la línea eléctrica y lo convierte en 180 V_{CD}, valor con el cual el convertidor genera una salida máxima de $60V_{CD}$ para la lámpara con ciclo de trabajo calculado. El módulo principal recibe los valores de iluminación y genera un ancho de pulso que varía de acuerdo a la iluminación que exista en el recinto, de esta manera la lámpara continuamente ajusta la cantidad de luz que entrega.

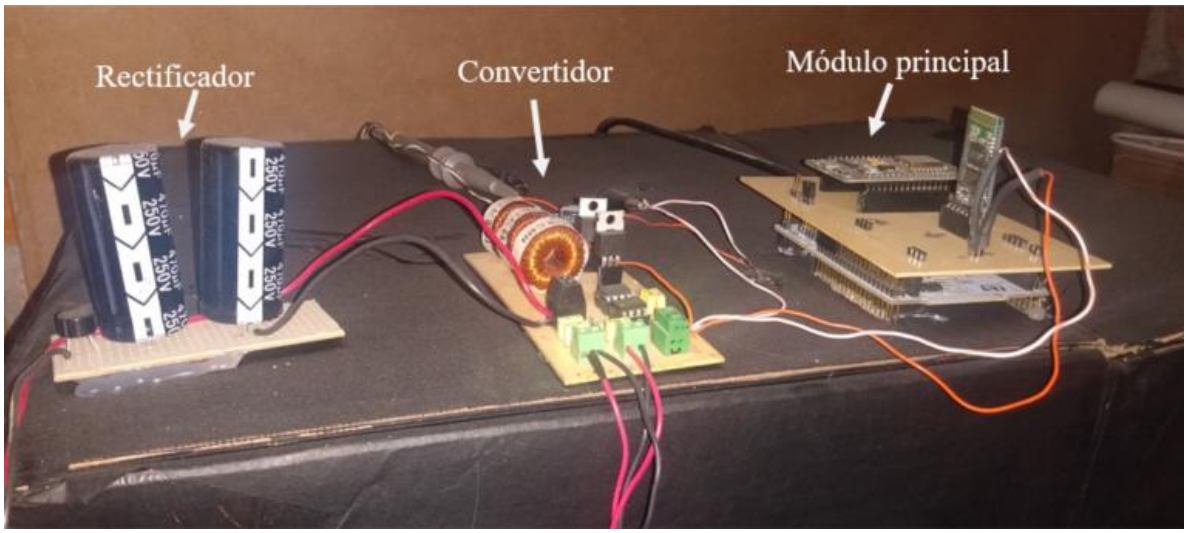


Figura 3. 22.- Rectificador, convertidor y módulo principal interconectados para el banco de pruebas.

El circuito encargado de medir la iluminación debe colocarse en el área de interés para registrar la iluminación que incide y mandar la retroalimentación al módulo principal. Una ventaja de que el circuito tenga conexión inalámbrica es la posibilidad de ubicarlo en cualquier parte que se desee dentro de la cobertura de enlace *Bluetooth* alimentando el circuito con una batería para hacer funcionar el módulo.

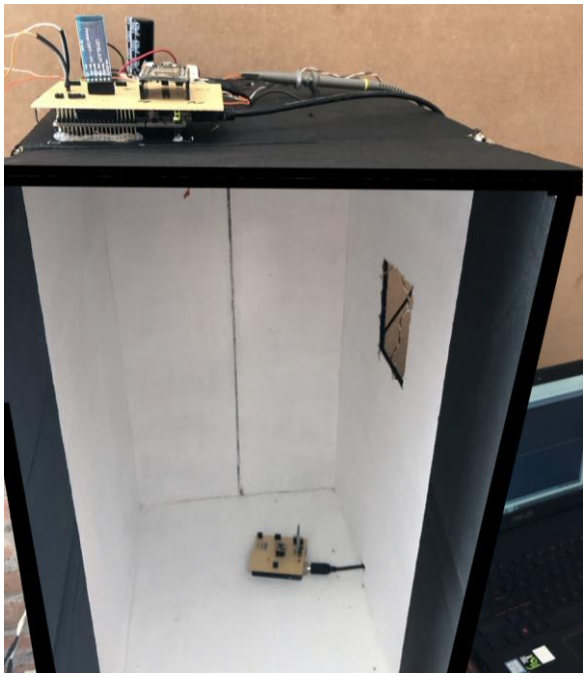


Figura 3. 23.- Ubicación del circuito de medición de iluminación.

El sistema tan pronto se energiza carga valores predeterminados y los manda al sitio web dando tiempo a que se inicialicen los módulos de comunicación, el sensor de iluminación y

los módulos *Bluetooth*. Se debe inicializar el módulo *WiFi* y conectarse al punto de acceso, lo que toma cerca de 5 segundos desde que enciende el módulo *WiFi*. Una vez encendido el módulo *WiFi* e inicializados los circuitos se comienzan a intercomunicarse iniciando por el sensor de iluminación el cual mide la iluminación y espera la petición del módulo principal.

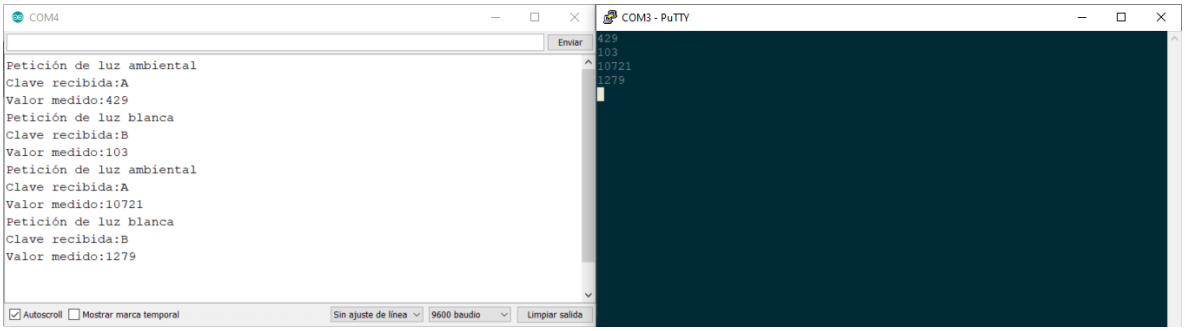


Figura 3. 24.- Lecturas del sensor de iluminación y lecturas del módulo principal respectivamente.

Se observa en la figura anterior de lado izquierdo las lecturas del terminal serial ubicado en el circuito de medición de iluminación en la que se aprecia tanto las claves que recibe como los datos que envía. El lado derecho corresponde al terminal serial del módulo principal, se aprecia que únicamente recibe los valores sin formato alguno lo que facilita su recepción y almacenamiento.

El módulo principal recibe las mediciones y las acomoda en un arreglo que se envía al sitio web para su almacenamiento. Se mencionó anteriormente la necesidad de verificar el formato de la comunicación ya que después de múltiples pruebas se determinó que la manera más eficiente de realizar intercambio de datos en las tarjetas en mediante un arreglo tipo JSON.

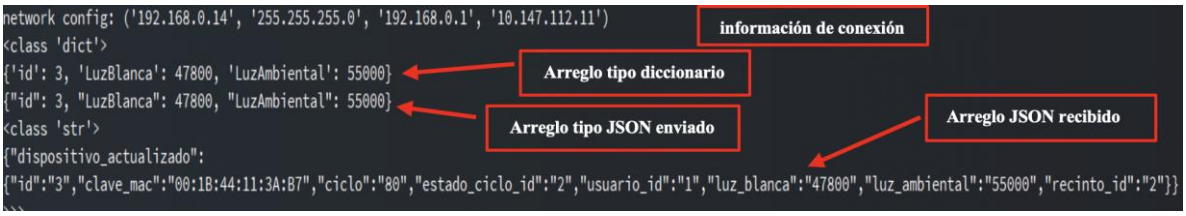


Figura 3. 25.- Comunicación con el sitio web.

En la Figura 4.25 se muestra el monitor serial del circuito ESP8266 en donde se aprecia en primer lugar el dato de tipo diccionario para luego, mediante una línea de código, convertirlo a formato JSON el cual se envía al servidor. El servidor retorna otro archivo JSON en donde se muestra la leyenda "*Dispositivo actualizado*" seguido de los campos que se actualizaron.

El sitio web recibe el arreglo de datos y los descompone en un arreglo para poder manejar los datos individualmente para posteriormente almacenarlos en la casilla correspondiente de la base de datos.

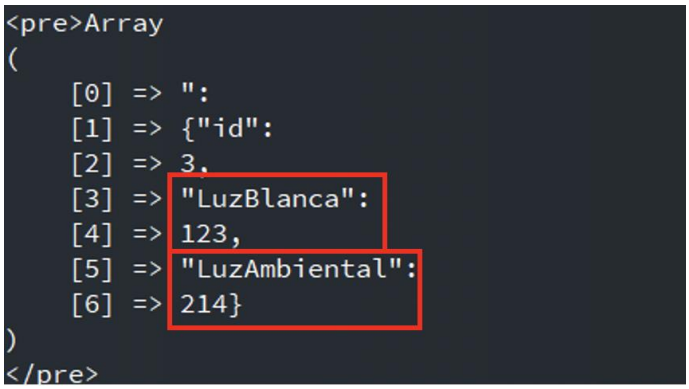


Figura 3. 26.-Formato de la información enviada al servidor.

La información del sitio se actualiza cada cinco segundos aproximadamente que es el tiempo en que el módulo realiza una petición POST. Con cada petición la base de datos se actualiza y el usuario puede ver la información tanto en la tabla de dispositivos como en los campos de actualización de cada dispositivo. La interfaz recibe y muestra en los campos de la tabla el ultimo valor registrado por lo que si no se actualiza la página no se ve reflejada la actualización por parte de la tarjeta.

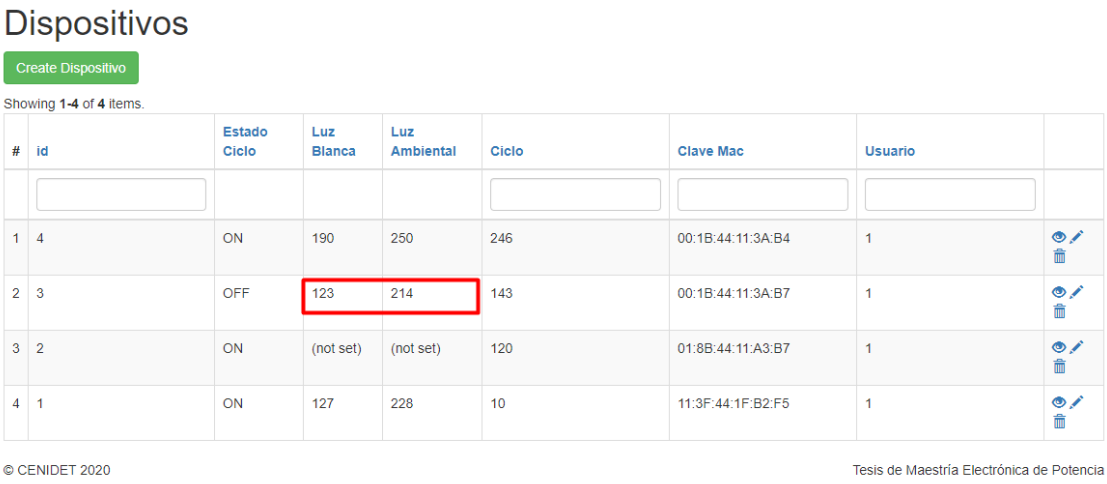


Figura 3. 27.- Datos almacenados en la tabla de dispositivos del sitio Web.

Tal como se observa en la Figura 3. 27, los datos que se almacenan en la tabla corresponden a los valores enviados en la petición de la Figura 3. 26 y cada que se envía un valor diferente la base de datos se actualiza y al recargar la página se puede visualizar en la interfaz. Al mismo tiempo de que el módulo *WiFi* entrega información al sitio web recibe el arreglo JSON que contiene el nivel de iluminación al cual el sistema se ajusta y el estado de la lámpara.

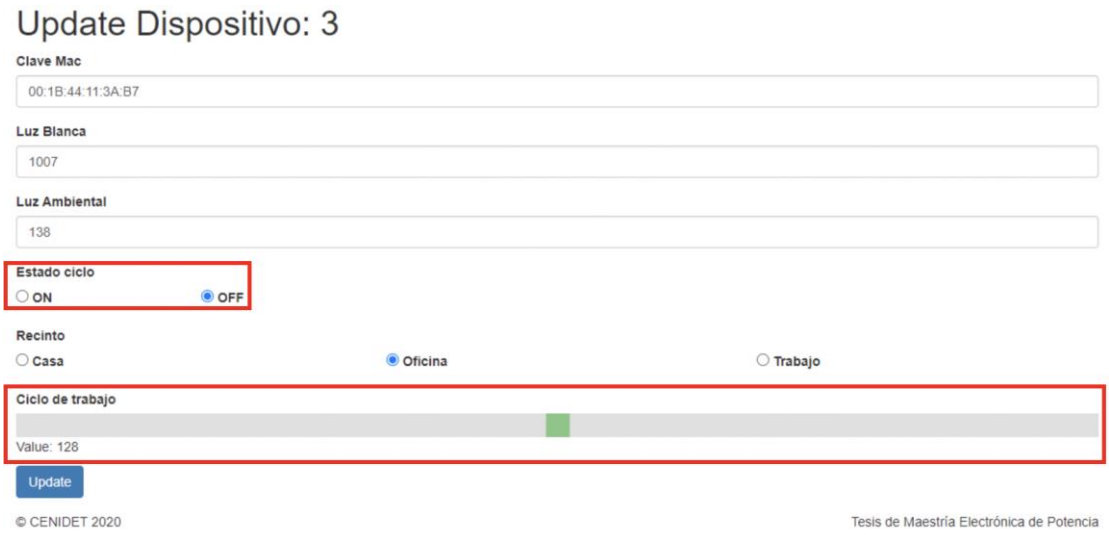


Figura 3. 28.-Información definida por el usuario en la interfaz.

Los datos señalados con rojo en la Figura 3. 28 son los campos que puede modificar el usuario en la interfaz y los que se retornan como respuesta en cada petición que hace el módulo principal. Tal como se mencionó anteriormente los datos se envían en un arreglo JSON para facilitar su transmisión y posteriormente lo decodifica el módulo *WiFi* para enviarlo a la tarjeta STM32L476RG.

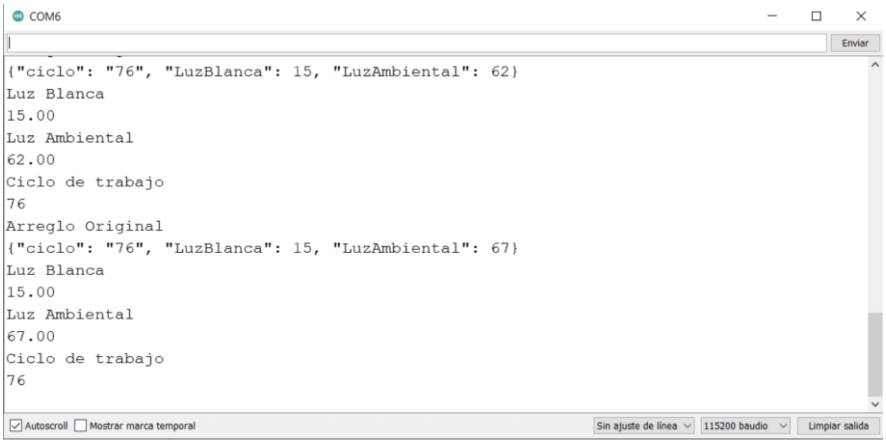


Figura 3. 29.- Respuesta del servidor a la petición del módulo principal.

La respuesta del servidor contiene la medición de luz ambiental y luz blanca, así como el valor al que debe ajustarse la iluminación. Con esta información el circuito determina el ciclo de trabajo correspondiente, así como las modificaciones que debe hacer progresivamente. Este circuito recibe la información cada 2 segundos y realiza el ajuste con este intervalo por lo que la lámpara continuamente ajusta su luminosidad (de ser necesario).

En la siguiente imagen se aprecia el valor máximo de ciclo de trabajo que genera el módulo principal correspondiente al 30% con una frecuencia de operación de 100KHz.

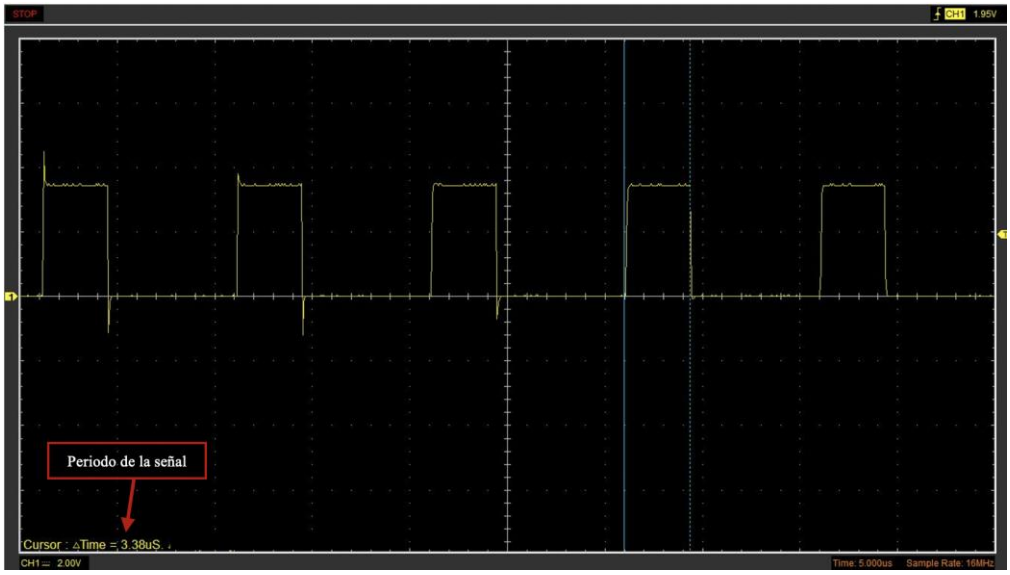


Figura 3. 30.- Señal PWM a 100KHz y un ciclo de trabajo del 33%.

El máximo ciclo de trabajo que el módulo puede generar es del 33% con el cual el convertidor obtiene 60VCD de salida el cual es el voltaje nominal de la lámpara led de prueba. Se aprecia en la Figura 3. 30 que se alcanza la frecuencia de 100KHz, conociendo que el periodo de la señal completa es de 10µs el 33% de la señal corresponde aproximadamente a 3.3µs lo cual se observa en la figura.

Capítulo IV

Conclusiones y Trabajos Futuros

4.1 Conclusiones

El objetivo del presente trabajo de investigación fue desarrollar un sistema que permitiera controlar la iluminación de una lámpara para aprovechar la iluminación en recintos semiabiertos. El desarrollo se logró tomando en cuenta el objetivo de utilizar tecnologías libres y de bajo costo, demostrando que pueden lograrse buenos desarrollos tecnológicos sin utilizar soluciones comerciales de grandes compañías desarrollando el sistema de comunicación desde cero con la certeza de una mejor seguridad al no depender de servicios de terceros.

Las tarjetas digitales presentadas durante la tesis contribuyeron en gran manera al desarrollo del proyecto y tienen un amplio campo de aplicación para resolver diversas problemáticas. El avance tecnológico permite que cada vez más se integren circuitos digitales en actividades cotidianas como ciudades inteligentes, redes de CD, control de iluminación, aplicaciones domóticas y monitoreo de diversos sistemas. Esto es cada vez más rentable debido al gran número de alternativas presentes en el mercado que permiten seleccionar alternativas libres y de bajo costo.

A la par de desarrollar sistemas eficientes que aprovechen de la mejor manera la energía consumida se debe hacer conciencia sobre el aprovechamiento de recursos disponibles como la iluminación natural en entornos semiabiertos. Se pueden desarrollar sistemas para trabajar en conjunto con alternativas ecológicas logrando el objetivo de ahorro energético.

De acuerdo a lo anteriormente mencionado y a las pruebas realizadas en capítulos anteriores se presentan las siguientes aportaciones y conclusiones basadas en los objetivos.

- Se diseñó e implementó un sistema capaz de modificar la iluminación de acuerdo al nivel de iluminación medido y de la referencia establecida en la interfaz web por el usuario.
- Se desarrolló un sitio web propio a través del cual circula información y se maneja de forma segura sin utilizar medios de transporte de información de terceras personas.
- Se diseñó y construyó un convertidor capaz de operar a los valores establecidos de voltaje y corriente necesarios para la lámpara LED seleccionada además de contar con la capacidad de comunicarse inalámbricamente vía Bluetooth y WiFi.

- Se utilizaron herramientas y tarjetas de uso libre y de bajo costo logrando buenos rendimientos y estabilidad en la comunicación inalámbrica.
- Se diseñó un circuito capaz de adquirir niveles de iluminación y transmitirlos inalámbricamente sin perdidas de información en la comunicación.
- Se sustituyó el firmware de módulos WiFi basados en el chip ESP8266 haciéndolo capaz de funcionar con el intérprete de alto nivel MicroPython con el fin de lograr comunicación bidireccional con el sitio web

4.2 Trabajos Futuros

A continuación, se presentan trabajos que contribuirían a lograr un sistema mas robusto:

- Realizar pruebas con un mayor número de convertidores comunicándose simultáneamente con el sitio web para verificar la estabilidad de la interfaz web.
- Instrumentar el convertidor para obtener valores de corriente y voltaje para presentarlos en la interfaz en internet.
- Probar el sistema con lámparas de diferentes marcas con las mismas especificaciones eléctricas.
- Añadir funciones adicionales al sitio web utilizando información de más sensores para lograr funcionalidades como estimación de vida útil, o ciclos de operación de la lámpara, entre otras cosas.
- Realizar pruebas con diferentes módulos WiFi de otras marcas además de la de Espressif y su módulo ESP8266.
- Utilizar tarjetas de desarrollo que contengan múltiples de protocolos de comunicación integrados y comparar el desempeño con el sistema actual.

Referencias bibliográficas

[1] D. Oficial and P. Sección, “NOM 025-STPS-2008.”

[2] NOAO *et al.*, “Niveles de iluminación recomendados,” 2015.

[3] Z. Ang and Y. C. Liang, “Wide range dimmable LED lighting system with fault compensation protocol,” *Proc. Int. Conf. Power Electron. Drive Syst.*, vol. 2015-Augus, no. June, pp. 184–187, 2015.

[4] A. Kar and A. Kar, “Advancements in solid state lighting technology for smart and energy efficient illumination system design,” *Proceeding IEEE Int. Conf. Green Comput. Commun. Electr. Eng. ICGCCEE 2014*, pp. 1–6, 2014.

[5] M. K. Kazimierczuk, *Pulse-Width Modulated DC-DC Power Converters*. 2016.

[6] O. A. C. May, J. S. Gio, and J. J. P. Koo, “Internet De Las Cosas Para Controlar El Encendido Y Apagado De Aires Acondicionados Y Luminarias,” *Pistas Educativas*, vol. 38, no. 122. 2016.

[7] O. L. Vele, “PROGRAMACIÓN DE PICs EN LENGUAJE C,” pp. 1–12.

[8] Espressif, “ESP8266EX Datasheet,” *Espr. Syst. Datasheet*, pp. 1–31, 2015.

[9] Particle, “Wi-Fi connected Internet of Things development kit with a small footprint, powered by a STM32 ARM Cortex M3 microcontroller and Cypress Wi-Fi radio,” 2015.

[10] Pycom, “WiPy Datasheet,” 2018.

[11] ITead Studio, “Bluetooth to serial port module,” 2010.

[12] G. Damien and P. Sokolovsky, “Getting started with MicroPython on the ESP8266 — MicroPython 1.12 documentation,” 2020. [Online]. Available: http://docs.micropython.org/en/latest/esp8266/tutorial/intro.html#deploying-the-firmware. [Accessed: 08-Mar-2020].

[13] H. Oudani, M. Elasikri, K. Karimi, and H. El Bousty, “Smart Cities : Energy Consumption in Wireless Sensor Networks for Road Traffic Modeling Using Simulator SUMO,” no. d, pp. 1–7, 2017.

[14] M. K. Patra, “An architecture model for smart city using Cognitive Internet of Things(CIoT),” *2017 Second Int. Conf. Electr. Comput. Commun. Technol.*, pp. 1–6, 2017.

[15] Digital Illumination Interface Alliance, “standards - Digital Illumination Interface Alliance.” [Online]. Available: https://www.digitalilluminationinterface.org/dali/standards.html. [Accessed: 21-Feb-2020].

[16] T. Juu Liang, J. Fong Huang, and P. Kumar Yadav, “Design and Implementation of Dimmable LED Control Circuit with DALI protocol,” 2016.

[17] J. Cardesín, D. García-Llera, J. Ribas, and D. Gacio, “Low cost intelligent LED driver for public Lighting Smart Grids,” 2013.

[18] A. Sinha, S. Sharma, P. Goswami, V. K. Verma, and M. Manas, “Design of an energy efficient Iot enabled smart system based on DALI network over MQTT protocol,” *3rd IEEE Int. Conf.*, pp. 1–5, 2017.

[19] D. Floarea and V. Sgarciu, “LED Smart Illumination with DFID Indoor Positioning,” in *Proceedings - 2017 21st International Conference on Control Systems and Computer, CSCS 2017*, 2017, pp. 515–522.

[20] H. B. Khalil, N. Abas, and S. Rauf, “Intelligent street light system in context of smart grid,” *8th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2017*, 2017.

[21] P. T. Daely, H. T. Reda, G. B. Satrya, J. W. Kim, and S. Y. Shin, “Design of smart LED streetlight system for smart city with web-based management system,” *IEEE Sens. J.*, vol. 17, no. 18, pp. 6100–6110, 2017.

[22] Fermax, “RBLED500 / REG. ILUMINACION LED 500W | Domótica | FERMAX.” [Online]. Available: https://www.fermax.com/spain/pro/productos/domotica/SF-80-reguladores/PR-11641-rbled500-reg-iluminacion-led-500w.html. [Accessed: 25-Jun-2020].

[23] C. Monterrey and O. Access, “Diseño e Implementación de un Sistema Remoto de Monitoreo y Control de la Iluminación de una Vivienda Utilizando una Red de Sensores Inalámbricos -Edición Única,” 2009.

[24] a Zanella, N. Bui, a Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for Smart Cities,” *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[25] H. Zhu, A. S. F. Chang, R. S. Kalawsky, K. F. Tsang, G. P. Hancke, and L. Lo Bello, “Review of State-of-the-Art Wireless Technologies and Applications in Smart Cities,” pp. 6187–6192, 2017.

[26] Vishay, “Fully Integrated Proximity and Ambient Light Sensor with Infrared Emitter, I 2 C Interface, and Interrupt Function,” 2015.

[27] Vishay, “Designing the VCNL4040 Into an Application,” 2019.

[28] MICREL, “12A-Peak Low-Side MOSFET Driver MIC4451/4452,” 2011.

Anexos

Programa de la tarjeta de medición de iluminación

El programa que se corre en el circuito sensor de iluminación está programado en Arduino y es un código sencillo el cual se describe a continuación.

```
#include <Adafruit_VCNL4040.h>
#include <SoftwareSerial.h>

SoftwareSerial BT(10,11);  // BT(RX,TX)
Adafruit_VCNL4040 vcnl4040 = Adafruit_VCNL4040();
char LBT;
int long lam;
int long lb;
String d1;
String d2;
void setup() {

    BT.begin(115200);
    Serial.begin(115200);
    if (!vcnl4040.begin()) {
    Serial.println("No se encuentra un dispisitivo");
    while (1);
  }
  Serial.println("Sensor de luz VCNL4040 encontrado");

  }
```

Figura Anexo 1. 1.- Configuración del programa en el circuito de medición de iluminación.

Se inicia declarando dos librerías, una para leer el sensor VCNL4040 y la otra para agregar un segundo puerto UART al circuito. El primer UART lo ocupa el Arduino para realizar la carga y comunicación con el monitor serial del IDE por lo que para ocuparlo se tendría que desconectar el periférico cada que se actualice el programa. Para evitar esto se declara un segundo puerto UART al que se conectara el módulo HC05 dejando libre el UART1 para carga de programa y visualización por monitor serial.

Con la instrucción SoftwareSerial se define el nombre del UART que se declara, así como los pines correspondientes a las interfaces Rx y Tx. Después de declarar las variables se inicializan ambos puertos UART a una velocidad de 115200 a la cual trabaja el sensor de iluminación. Una vez inicializados se inicializa el sensor de luz y cuando se encuentra se muestra un mensaje en la consola diciendo que se encontró el sensor. Con esto se termina la parte de configuración.

```
void loop() {

  lam=vcnl4040.getAmbientLight();
  lb=vcnl4040.getWhiteLight();
  d1=String(lam);
  d2=String(lb);

  if (BT.available())
    {
    LBT=BT.read();
    //Serial.print(LBT);
    if (LBT=='A'){
    Serial.println("Petición de luz ambiental");
    BT.println(d1);
    Serial.print("Clave recibida:");
    Serial.println(LBT);
    Serial.print("Valor medido:");
    Serial.println(d1);
    }
    if (LBT=='B'){

    Serial.println("Petición de luz blanca");
    BT.println(d2);
    Serial.print("Clave recibida:");
    Serial.println(LBT);
    Serial.print("Valor medido:");
    Serial.println();

    }
    }
}
```

Figura Anexo 1. 2.- Código principal del circuito.

Dentro de la función *void loop* se ejecutan las instrucciones principales del sistema. En primer lugar, se obtienen tanto la cantidad de luz ambiental como la de luz blanca y se almacenan en las variables *lam* y *lb* respectivamente y posteriormente se convierten en datos tipo strings, esto para facilitar su envío a través de comunicación serial. El programa continúa al verificar si existen datos esperando en el puerto serial y en caso de ser así, se obtiene el valor y se compara para determinar si se envió algunas de las claves que se programaron en ambas tarjetas, de lo contrario esperara hasta recibir alguna de ellas. Si el sistema recibe la letra *A* enviara a través de Bluetooth el valor de luz ambiental y mandara por puerto serial mensajes para verificar que llega y que se envía en caso que se quiera verificar el flujo de datos. De manera similar si el sistema recibe la letra *B* realizara el mismo proceso, pero esta vez con los valores de luz blanca.

Programa de la tarjeta para generar la señal PWM

La tarjeta STM32L476RG también se programó utilizando el IDE de Arduino agregando las tarjetas de manera manual y cargando la información de la tarjeta en el entorno para poder programarlo con este lenguaje.

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(PC10, PC12); // RX, TX
String data;
int digPin = PB4;
int dato;
int duty;
void setup()
{
   analogWriteFrequency(100000);
   Serial.begin(115200);
   mySerial.begin(115200);

  mySerial.println("Listo para recibir");
}

void loop()
{

   if (mySerial.available())
   {
      data = mySerial.readStringUntil('*');
      Serial.println(data);
      Serial.print(duty);
   }
   dato=data.toInt();
   if(dato>=0&&dato<=255)
   duty=dato*85/255;
   analogWrite(digPin,duty);
}
```

Figura Anexo 1. 3.- Código utilizado para el generador PWM de la tarjeta STM32L476RG.

El programa inicia de manera similar al anterior código declarando una segunda interfaz UART y declarando variables para su uso posterior. En la función de configuración *void setup* se ajusta la frecuencia a la cual se generará la señal PWM, y se ajusta la velocidad de ambos puertos seriales, una vez configurados se mostrará mediante interfaz serial un mensaje indicando que el dispositivo se encuentra preparado.

Entrando a la función principal se inicia validando si existen mensajes en el puerto serial y de ser así se utiliza la función *readStringUntil* para leer el dato entrante. Se utiliza esta función especial debido a que ahora se recibirá una cadena de caracteres en lugar de uno solo, y el tamaño del dato varía en función de la cantidad de luz que se registra. Para evitar perder dato el microcontrolador que envía la información enviará el símbolo * una vez terminada la transmisión, de esta manera se asegura que se reciba la información completa y se minimiza los errores de transmisión.

Las últimas líneas de código convierten el valor a tipo entero para poder validar que se encuentre en un rango aceptable, y de ser así lo ajustan para que el valor no exceda nunca el 33% de ciclo de trabajo, correspondiente al máximo de diseño al que puede operar el convertidor.

Una vez ajustado el valor se ejecuta la función *AnalogWrite* en donde se coloca el valor de ciclo de trabajo y el pin a través del cual se generará la función.

Programa del módulo principal WiFi.

Programa de carga boot.py

El módulo wifi ESP8266 está diseñado para ejecutar dos programas al energizarse. El primero es la configuración inicial y es un programa llamado boot.py el cual se muestra en la Figura Anexo 1. 4.

```
import uos
from machine import Pin, UART
import os
import gc
import time
gc.collect()
led1=Pin(16,Pin.IN)


def connect():
    import network
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to network...')
        sta_if.active(True)
        sta_if.connect('SSID', 'PASSWORD')
        while not sta_if.isconnected():
            pass
    print('network config:', sta_if.ifconfig())


uos.dupterm(UART(0,115200), 1) # Activar el modo REPL en el UART(0)
time.sleep(3)

valorin=led1.value()
if (valorin==0):
  os.remove('main.py')
  time.sleep(2)
```

Figura Anexo 1. 4.- Programa de configuración contenido en boot.py.

Aunque el lenguaje es diferente el flujo es parecido, se inicia declarando librerías e inicializando un pin, este pin servirá para salir de un ciclo infinito. El módulo WiFi se debe conectar a internet inicialmente y posteriormente cuando se reinicia guarda la configuración y se conecta automáticamente. Para realizar la conexión se creó una función llamada *connect* la cual determina si el módulo se encuentra conectado a internet y de no ser así inicia la conexión. En el script basta con modificar los campos *SSID y PASSWORD* con los datos de la red a la que se desea conectar para lograr la conexión. Una vez que el módulo se encuentra conectado a internet se mostrará los parámetros de conexión.

La instrucción *uos.dupterm* se utiliza para activar nuevamente el modo REPL y se coloca en el fichero boot.py por si existe la posibilidad de terminar con el ciclo infinito ya que este

fichero se ejecuta en primer lugar. Si se desea terminar con el ciclo se debe poner en bajo el pin que se declaró al inicio y reiniciar la tarjeta, de esta manera se borrará el programa *main.py* posibilitando al usuario modificar los archivos en la tarjeta. Mientras el pin se mantenga en alto la tarjeta ejecutará el programa boot.py y posteriormente ejecutará el programa principal, main.py.

Se comentó anteriormente que para poder utilizar las interfaces UART se debe deshabilitar el intérprete de MicroPython el cual se inicia por default, sin embargo, una vez que se deshabilita la tarjeta deja de responder a comandos y las formas de detenerlo es borrar el firmware y volverlo a cargar o borrar el programa *main.py* en donde se encuentra la instrucción para deshabilitar el modo REPL.

Rutina principal main.py

El segundo programa que se ejecuta tan pronto termina boot.py corresponde al programa principal *main.py*. En este programa se encuentran codificadas las instrucciones que permiten que el módulo principal cumpla las funciones que se pusieron como meta. El código se presenta a continuación de manera segmentada para explicar el funcionamiento de cada parte.

```
from machine import Pin, UART,PWM
import time
import uos
import urequests
import ujson
import network
uos.dupterm(None, 1)
time.sleep(2)
uart=UART(0,115200) #Seleccion del puerto UART y la velocidad
uart1=UART(1,115200)
uart.init(115200, bits=8, parity=None, stop=1) #Parametros de inicio del puerto UART
uart1.init(115200, bits=8, parity=None, stop=1)
```

Figura Anexo 1. 5.- Inicio del programa main.py.

El programa inicia declarando las librerías que se utilizarán a lo largo del programa e inmediatamente después se desactiva el modo REPL para poder utilizar el UART0 en donde se conecta el módulo HC05. Se nombran los dos puertos UART que se utilizarán y se inicializan. El puerto UART1 se utiliza para enviar el valor de ciclo de trabajo que generará la tarjeta encargada de dicha tarea.

```
#direccion del host
url="http://mksgabat.online/index.php?r=dispositivo-api%2Finfo-dispositivo"
#cabecera para indicar el tipo de dato a enviar
cabecera={'Content-Type':'application/x-www-form-urlencoded'}
dato1=60000
dato2=57000
connect() #Funcion para establecer la conexion
parametros={'id':3} #Declaracion del dato tipo diccionario
```

Figura Anexo 1. 6.- Inicialización de datos en main.py.

Antes de entrar al ciclo infinito se indica la dirección a la cual se realizará la petición, así como la cabecera la cual indica el tipo de datos que se enviarán. Se inicializan las variables que almacenarán el valor de iluminación que se recibirá a través de Bluetooth con un número aproximado al mayor valor que tendrá la variable para que el intérprete de MicroPython asigne un espacio de memoria suficiente para los valores de iluminación. Se llama la función para verificar la conexión y se crea una variable de tipo diccionario en la cual se asigna el número de dispositivo.

```
while(1):

  time.sleep(3)

  uart.write('A')
  if uart.any()>0:
    dato1=int(uart.readline())
    time.sleep(1)
  uart.write('B')
  if uart.any()>0:
    dato2=int(uart.readline())
    time.sleep(1)

  parametros['LuzAmbiental']=dato1 #Asignacion valor a variable LuzAmbiental en diccionario
  parametros['LuzBlanca']=dato2    #Asignacion valor a variable LuzBlanca
```

Figura Anexo 1. 7.- Entrada a ciclo infinito del programa main.py.

La tarjeta ejecuta el programa una sola vez, sin embargo, se requiere que el programa se ejecute constantemente, para ello se utiliza un ciclo infinito y dentro de él se colocan las instrucciones principales. El primer bloque de instrucciones se utiliza para comunicarse con el circuito de medición de luz, en primera instancia se envía la letra *A* y se espera a que el circuito envíe la información, posteriormente se repite la acción, pero con la letra *B*. Los valores se almacenan en *dato1 y dato2*. Estos valores posteriormente se almacenan en el diccionario de nombre *parámetros* que se declaró anteriormente, ya que se enviarán como parámetros en la petición.

```
#Peticion al servidor
respuesta = urequests.post(url, json={"informacion=":parametros} ,headers=cabecera)
#Decodificacion del valor de respuesta del servidor
objrespuesta= ujson.loads (respuesta.text)
#Acceso a los cammpos contenidos en dispositivo_actualizado
inforespuesta=objrespuesta.get('dispositivo_actualizado')
#Guardado del valor del ciclo obtenido en el servidor
ciclo=inforespuesta.get('ciclo')
estado=inforespuesta.get('estado_ciclo_id')


envio_ciclo=str(ciclo)
uart1.write(envio_ciclo)
uart1.write('*')
time.sleep(5)
```

Figura Anexo 1. 8.-Petición al servidor en el programa main.py.

Cuando se reunieron los datos que se enviarán al servidor se realiza la petición al servidor, en ella se coloca la dirección en donde se encuentra la función *info-dispositivos* la cual se declaró anteriormente, se envían los *parámetros* como un arreglo JSON y se indica la cabecera para indicar el tipo de dato. Al tiempo que el servidor recibe los datos regresa el valor de ciclo de trabajo y el estado en un formato al cual se accede con el nombre de la función agregando *.text* además de decodificarla con la función *ujson.loads.* El servidor responde con un dato de tipo diccionario al cual se puede acceder utilizando las llaves, por ejemplo, para acceder al ciclo se debe utilizar la línea *ciclo=inforespuesta.get('ciclo')* como se muestra en la figura anterior.

El valor del ciclo de trabajo se extrajo del diccionario de la respuesta y se almacenó en la variable ciclo y es de tipo entero. Este valor se enviará por puerto serial a la tarjeta STM32L476RG para generar el PWM en función de este valor por lo que es conveniente convertirlo a un tipo *string* para evitar la pérdida de información y una vez recibido en la otra tarjeta convertirlo nuevamente a entero. Para enviarlo se utiliza la línea de código uart1.write(envio_ciclo) notando que la interfaz serial que se ocupa ahora es el UART1.

Controlador principal del sitio *web*

Como se mencionó anteriormente en el apartado 2.14 Diseño del sitio *W* se utilizan muchos archivos para lograr el correcto funcionamiento del sitio web, sin embargo la función que permite la comunicación entre el sitio y la tarjeta se encuentra en la carpeta *controllers* bajo el nombre *DispositivoApiController*. Este controlador tiene diferentes funciones que se encargan de capturar datos de la interfaz, de actualizar la base de datos, de leer y mostrar información de la base de datos en la tabla de vista, entre otros sin embargo la que se describirá será la de *InfoDispositivo*. Es importante mencionar que la función está programada en lenguaje PHP.

```php
public function actionInfoDispositivo() {
    $post = Yii::$app->request->post();
    $respuesta = [];
    $estatus = 0;
    $mensaje = "";
    $info = $post['{"informacion'];
    $auxData = rtrim($info, "}");
    $data = $auxData.'}';
    $arrayData = explode(' ',$data);

    $arrayId = explode(',',$arrayData[2]);
    $id = $arrayId[0];

    $arrayLuzBlanca = explode(',',$arrayData[4]);
    $luzBlanca = $arrayLuzBlanca[0];

    $arrayLuzAmbiental = explode('}',$arrayData[6]);
    $luzAmbiental = $arrayLuzAmbiental[0];

    $idDispositivo= $id;
    $dispositivo = Dispositivo::find()->where(['id'=>$idDispositivo])->one();
    if(isset($dispositivo)) {
        $estatus = 1;
        $mensaje = "Dispositivo Encontrado";
        $dispositivo->luz_blanca = $luzBlanca;
        $dispositivo->luz_ambiental = $luzAmbiental;
        if ($dispositivo->save()) {
            $result = Dispositivo::find()->where(['id'=>$idDispositivo])->asArray()->one();
        } else {
          System::beautyPrint($dispositivo->getErrors());
        }
    } else {
        $mensaje = "Error con el dispositivo";
        System::beautyPrint("Error con el dispositivo");
    }
   $respuesta["dispositivo_actualizado"] = $result;

    return JSON::encode($respuesta);
}
```

Figura Anexo 1. 9.- Función de comunicación con la tarjeta InfoDispositivo.

El primer bloque de código recibe la petición del módulo WiFi con la primera línea de código, y la almacena en la variable *post*. Las siguientes líneas completan el formato ya que, a pesar de que la información se envía en formato JSON existe perdida de información en cuanto a llaves y comillas de tal manera que se concatenan los símbolos que se pierden para posteriormente decodificarlo.

Cuando el formato se encuentra completo se utiliza la función *explode* para convertir el diccionario en un arreglo con el fin de separar los valores de las llaves y almacenar únicamente el valor numérico. Después de una serie de manipulaciones los valores se almacenan en las variables *luzBlanca* y *luzAmbiental*. Los valores se encuentran almacenados en las variables, pero aún no se guardan en la base de datos, antes de almacenarlos se debe validar a que dispositivo pertenecen los datos y si el dispositivo se encuentra registrado, para ello se utiliza la primer sentencia *if* la cual realiza la función. Si el dispositivo se encuentra registrado entonces se almacenan los datos en la base con la función *$dispositivo→luz_blanca=$luzBlanca y $dispositivo→luz_ambiental=$luzAmbiental.* De esta manera los valores recibidos se actualizaron en la base de datos reflejándose en la interfaz de usuario. Se realiza una validación para saber si se almacenaron correctamente y de lo contrario obtiene la razón por la que no se pudo validar, en caso de error retornará la razón que provoco el error.

Finalmente se obtienen los valores almacenados en la tabla con la línea *$respuesta ["dispositivo actualizado"]=$result* y se envía a la tarjeta con formato JSON con la instrucción *return JSON:encode($respuesa)* finalizando la función.