ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

MODIFICACIÓN DEL FIRMWARE DEL MÓDULO ESP8266 PARA CONSTRUIR UN DISPOSITIVO ON – OFF QUE SERÁ COMANDADO POR UNA APLICACIÓN RESIDENTE EN UN DISPOSITIVO ANDROID EN UNA RED WLAN

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

MELINGTON ANTONIO CÁCERES JAGUACO

DIRECTOR: MSc. Ing. RAMIRO EDUARDO MOREJÓN TOBAR

Quito, octubre 2022

AVAL

Certifico que el presente trabajo fue desarrollado por Melington Antonio Cáceres Jaguaco, bajo mi supervisión.

_____

MSC. RAMIRO MOREJÓN
DIRECTOR DE TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Melington Antonio Cáceres Jaguaco, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

_____

MELINGTON ANTONIO CÁCERES JAGUACO

DEDICATORIA

A mi Dios, quien estoy seguro, que nos da la vida, el tiempo, el conocimiento, la sabiduría y brindarnos todas las oportunidades para poder demostrar que debemos ser mejores seres humanos.

A mis padres por siempre apoyarme con su ánimo, insistencia, ejemplo de trabajo, fortaleza e incentivarme a ser mejor cada día.

A mi esposa y mis hijos por su ayuda, paciencia y comprensión todo el tiempo que no estoy junto a ellos.

A las excelentísimas autoridades de la Facultad, en especial, a mi tutor, por esa presteza desinteresada, la gran ayuda, la paciencia y el gran interés que ha demostrado en estos momentos de duro trabajo.

A Alguien más, que estará siempre conmigo, quien llegó a ser mi inspiración y le dio rumbo a mi vida.

Melington

AGRADECIMIENTO

Agradezco de forma muy especial a todos quienes han hecho posible el desarrollo de este trabajo.

Mi Dios quien todo lo permite.

Mis padres por darme la vida.

Mi familia por brindarme su apoyo

Las Autoridades de la Facultad por la oportunidad que me han dado

A mi Tutor por su gran ayuda e interés personal.

Unas mil gracias, a todos que directa e indirectamente, me han dado una mano para terminar este trabajo de profesionalización.

Melington

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo de titulación, se centra en modificar el firmware del módulo ESP8266, por medio de la IDE de Arduino, para diseñar un dispositivo de gestión y control ON-OFF, sobre una red Wlan, manipulado por una interfaz de usuario, afincada sobre el sistema Android, como una aplicación domótica. Para ello, se deben realizar ciertos procesos que permitan efectuar el diseño y su respectiva implementación.

Entre los procesos que se deben efectuar, están primero, acoplar una fuente de energía adecuada al voltaje de alimentación, del módulo ESP8266, que le permita trabajar sin riesgo de daños. Aquella fuente se la deriva de una de 5V a 1A, por medio del uso de un circuito regulador de 3.3V.

Luego, se realiza la modificación del firmware original del módulo por otro que pueda trabajar por medio de la IDE de Arduino. Este proceso, incurre en realizar la carga de instaladores de tarjetas y librerías, que contienen de forma implícita el firmware para el ESP8266 en Arduino. Por otro lado la IDE de Arduino actuará como intermediario o FTDI, para cargar el nuevo firmware sobre el ESP8266.

La modificación del firmware del ESP8266, permitirá que el módulo pueda ser programado como Servidor Web, lo que dará cabida al diseño de una aplicación residente sobre un dispositivo Android. Así, el usuario realizará la gestión domótica del ESP8266, el que por medio de un relé controlará la actividad ON-OFF del dispositivo.

Todo se conjugará, en un módulo de gestión y control ON-OFF sobre una red Wlan.

PALABRAS CLAVE: módulo ESP8266, firmware, IDE, fuente, interfaz de usuario

ABSTRACT

This degree work focuses on modifying the firmware of the ESP8266 module, through the Arduino IDE, to design an ON-OFF management and control device, over a Wlan network, manipulated by a user interface based on the Android system, as a home automation application. Therefore, certain processes must be carried out that allow the design and its implementation.

Among the processes that must be carried out, they are first, coupling a power source, which allows the ESP8266 module to work without risk of damage. That source is derived from a 5V to 1A, through the use of a 3.3V regulator.

Then, the modification of the original software of the ESP8266 module is carried out, through the Arduino IDE. This process incurs in loading the installers and libraries, which implicitly contain the firmware for the ESP8266 on the Arduino platform. Instead the Arduino IDE will act as an intermediary or FTDI, to load the new firmware on the ESP8266.

The modification of the ESP8266 firmware will allow the module to be programmed as a Web Server, which will allow the design of a resident application on an Android device. This application will allow the user to manage the ESP8266 home automation, which by means of a relay will control the ON-OFF activity of the device.

Finally, everything will be combined, in a single module that controls ON-OFF management and control over a Wlan network.

KEYWORDS: ESP8266 module, firmware, IDE, source, user interface

1. INTRODUCCIÓN

Actualmente con el desarrollo tecnológico e informático, el ser humano es capaz de automatizar la mayor parte de actividades que crea conveniente hacerlo, ya sea con el fin de brindar confort, ahorro de energía y facilitar procesos domésticos o industriales. Eso ha permitido que se diseñen un sin número de dispositivos, artefactos y aplicaciones informáticas, capaces de manipular, controlar y gestionar los procesos mencionados de forma virtual o a distancia.

Varias de esas aplicaciones son en el campo del hardware y software. En el campo del hardware, desde hace algunas décadas [1], se han mejorado las aplicaciones domóticas e inmóticas, lo que ha facilitado y mejorado muchas actividades cotidianas del ser humano.

De igual forma, con el avance vertiginoso del software y paralelamente de plataformas de código abierto como Linux, del que después se desarrolla Android [2], han permitido que en conjunción con el hardware libre, se diseñen dispositivos que permiten gestionar y controlar varios rasgos del quehacer humano. Muchos de esos dispositivos están compuestos por microprocesadores o microcontroladores, los mismos que permiten ser controlados por aplicaciones de software libre, desarrollando así la industria domótica, inmótica y la del internet de la cosas.

Siendo ese el panorama actual, es importante mencionar que se han desarrollado plataformas como Arduino, enfocadas en el manejo de sistemas electrónicos que permiten crear aplicaciones para automatizar ciertas facetas de la actividad humana. Sin embargo, Arduino también se basa en un interfaz de código abierto que le permite ser accesible a cualquier usuario [3], esa misma razón ha hecho que se la use en una amplia gama de aplicaciones electrónicas. Esos avances han promovido el crecimiento de una comunidad de personas que trabajan de forma colaborativa para mejorar lo ya existente, como por ejemplo, el desarrollo de una gran cantidad de sensores y actuadores necesarios para la interacción entre los sistemas automatizados y el medio o magnitud sobre los que se aplican.

Entre uno de esos actuadores está, el módulo ESP8266, conocido como ESP-01[1], que es un actuador basado en Arduino, que permite comunicarse de forma inalámbrica por medio del protocolo TCP/IP [4] y las capacidades físicas de su microcontrolador. Eso es lo que permite que el módulo Wi-Fi de Arduino pueda controlar bajo una red Wlan, a dispositivos conectados al mismo pero sin esa capacidad.

Tomando en cuenta lo antes mencionado, se cree que se puede aprovechar la facilidad que brindan las plataformas de código abierto, para modificar el firmware del módulo ESP8266 a fin de diseñar un dispositivo de gestión y control ON-OFF, el mismo que pueda ser manipulado por una aplicación basada en Android, para controlar dispositivos o artefactos con ese modo de accionar y como parte de un aporte domótico.

1.1. OBJETIVOS

1.1.1. OBJETIVO GENERAL

Diseñar e implementar un dispositivo integrado por el módulo ESP8266, modificando su firmware para trabajar como un interruptor (ON – OFF), por medio de una interfaz residente en Android manteniendo la operación y gestión domótica a través de una red Wlan.

1.1.2. OBJETIVOS ESPECÍFICOS

- Revisar, comprender y aplicar los conocimientos teóricos sobre la interrelación entre el ESP8266, la IDE de Arduino y la interfaz de usuario en Android.
- Desarrollar un esquema metodológico que permita realizar el diseño del prototipo del dispositivo de gestión domótica, modificando el firmware original del módulo ESP8266, integrarlo a un interruptor magnético y desarrollar una aplicación para Android, para trabajar dentro de una red Wlan.
- Implementar el módulo integrado entre el ESP8266, para la actividad de control ON-OFF y gestionado por la interfaz desarrollada en Android.
- Comprobar y Analizar los resultados del funcionamiento del módulo integrado con su interfaz de gestión domótica.

[1] Para el caso de este proyecto se hace referencia al módulo ESP-01 como ESP8266.

1.2. ALCANCE

El propósito de este trabajo de titulación es proponer el diseño e implementación de un dispositivo integrado por el ESP8266, modificando su software y un interruptor magnético que es gestionado por un interfaz domótico para el usuario en un dispositivo Android dentro de una red Wlan. Para lograr el objetivo deseado es necesario realizar ciertos procesos de desarrollo, los que se presentan a continuación.

En primer lugar, se realiza un estudio teórico de los temas relacionados con el objetivo. Entonces se inicia efectuando un análisis básico de la plataforma Arduino, enfocándose específicamente en su IDE[2], que es la que permite realizar la modificación del Firmware [3] original del módulo ESP8266, por uno que es gestionado por Arduino. Además, se realiza un enfoque más profundo en la composición física y de software del módulo Wi-Fi. Y finalmente un análisis de la plataforma que permita desarrollar la interfaz de usuario en Android, la que ayudará en la gestión del dispositivo final por medio de la red Wlan.

Después, se realiza una perspectiva metodológica con el objeto de crear una ruta procedimental para el desarrollo del prototipo. En ese punto se cree que es muy necesario describir por medio de un esquema de bloques, los procesos que permiten alcanzar el objetivo de este proyecto.

Como tercera parte y fundamental del proyecto es centrar lo anterior en la implementación del prototipo. Aquello consiste en ir realizando paso a paso lo establecidos en la parte metodológica. Primero, desarrollar la alimentación energética del módulo ESP8266, considerando sus limitaciones en ese campo. Luego, es modificar el firmware original del módulo, por medio de usar un software compatible con la IDE de la plataforma Arduino. Después se realizará la combinación del módulo Wi-Fi con el interruptor magnético (ON-OFF), con el fin de encender o apagar un dispositivo con esa acción como una lámpara de 120VAC, pero la gestión será realizada por el usuario, al usar una interfaz residente en un dispositivo Android por medio de una red Wlan.

Finalmente con el propósito de solventar el funcionamiento, se realizará un análisis del comportamiento del dispositivo desarrollado, lo que permitirá exponer conclusiones en cuanto a su funcionamiento y recomendaciones sobre si se requiere mejoras.

[2] Entorno de Desarrollo Integrado (Integrated Development Environment). Interfaz de gestión y programación de Arduino.

[3] El firmware es el sistema operativo básico de los dispositivos con esquema SoC (System on Chip)

1.3. MARCO TEÓRICO

Para iniciar el estudio del módulo ESP8266 o ESP-01, antes, se debe conocer la historia de la plataforma Arduino. Esta plataforma es adaptativa a distintos entornos, razón por la que se han desarrollado distintos tipos sensores y actuadores, mismos que sirven de interfaz entre el hardware de control y el medio de aplicación, según sean las necesidades del usuario.

El ESP8266 se originó tal como otros módulos actuadores, con el propósito de dar mayor aplicabilidad a las tarjetas Arduino. Pero el módulo está especialmente relacionado al ámbito de comunicaciones inalámbricas (Wireless o WIFI), aprovechando el hardware del que está compuesto.

Cabe resaltar que actualmente las comunicaciones inalámbricas se han desarrollado vertiginosamente y casi se encuentran en todos los entornos que nos rodean. Uno de esos campos es el de gestión, control, monitoreo de dispositivos y equipos electrónicos, tanto industriales, como de confort en el hogar, lo que actualmente se denomina domótica.

Por esa razón se inicia el análisis con un breve estudio de la plataforma Arduino, enfocándose de forma más relevante en los componentes que afectan este proyecto.

1.4. ARDUINO

Arduino es una placa de prototipos de circuitos electrónicos basados en un microcontrolador, generalmente Atmel. Su programación y software se gestiona a través de una plataforma de código abierto, basado en un lenguaje simplificado de tipo Basic.

Los principios de Arduino, están relacionados con el proyecto del Instituto de Diseño de Interacción de Ivrea (IDII, Italia), para diseñar una herramienta electrónica estándar, que sirva como la base de proyectos de ingeniería y robótica [5], en el año 2000.

Se debe resaltar que, como todo proyecto en desarrollo y de reciente creación, las “tarjetas de prueba” o hardware en diseño, resultaron en ese momento ser muy costosas, de difícil programación y de lenta gestión de procesos. Así es que desde el año 2001, se concibe la idea de desarrollar un software propio para el hardware de la nueva plataforma. El software tenía que ser amigable y fácil de usar, así es que aparece como primera opción la plataforma de apoyo, Processing. Esta plataforma tenía limitaciones, pero no en lo

relacionad al aspecto técnico sino a la accesibilidad, pues se requería tener un amplio conocimiento de programación, lo que reducía el uso de la gran parte de usuarios.

Después unos años más tarde, se prosigue con el proyecto, pero ahora como parte de un proyecto de fin de carrera de masterado, el estudiante colombiano, Hernando Barragán, se enfoca en el diseño de la tarjeta electrónica WIRING, antecesora de la tarjeta Arduino y el desarrollo de un software propio, que tendrías su propio lenguaje de programación y plataforma de desarrollo, en el año 2004 [6]. Esta tarjeta ya mejorada en hardware, se puede decir que es la primera ARDUINO, propiamente dicha.

Prosiguiendo con las mejoras, tanto estudiantes como profesores, se enfocan en mejorar la tarjeta WIRING, así que en el año 2005 se logra obtener la primera versión de Arduino, la que se basa en un microcontrolador, el ATmega8, tal como se puede observar en la Figura 1.1. De la misma forma su software de gestión tuvo cambios, lo que le permitió dar mayor accesibilidad y fácil programación. De esa forma la tarjeta, pasa a ser gestionada y programada, por medio de un software propio totalmente basado lenguaje C/C++, además, que es de Código Abierto (Open Source).

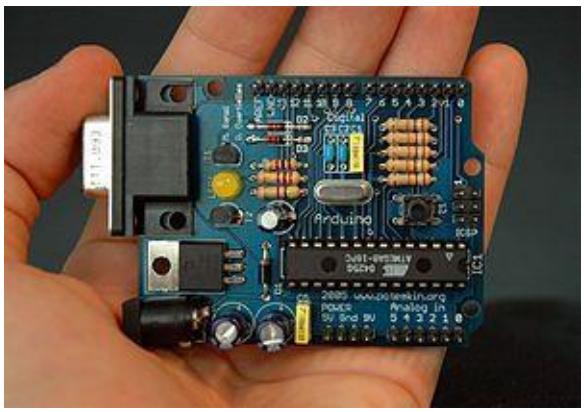

Figura 1.1. Primera tarjeta Arduino [7]

De esa forma se logró idealizar y crear una tarjeta estándar que permitiera la creación de otros proyectos electrónicos y robóticos, con la capacidad de interactuar con distintos medios con solo cambiar ciertos dispositivos denominados sensores y actuadores, la gran mayoría de ellos externos.

Finalmente luego del cierre del instituto IVREA, los gestores y desarrolladores de Arduino, optaron por determinar que el producto creado sea de Hardware y Software libres[4], a fin de evitar que los diseños sean embargados [7]. Por lo tanto, desde ese momento, las tarjetas Arduino se han desarrollado de muchos tipos, capacidad y aplicación.

Esto ha permitido también que se desarrolle una Comunidad de especialistas, desarrolladores y aficionados, que de forma colaborativa y cooperativa han estado aportando al crecimiento de las aplicaciones y usos de Arduino. A esa comunidad se la denomina MAKER, enfocándose en el lema de "hacerlo por uno mismo" [8].

Con esa reseña histórica se procede a resaltar que sea cual sea, la tarjeta Arduino que se use, todas están compuestas de dos partes fundamentales, el hardware y el software. Por esa razón, a continuación se procede a estudiarlos como parte del desarrollo de este trabajo.

1.4.1. HARDWARE DE ARDUINO

La placa Arduino está compuesta originalmente por, una tarjeta de base - soporte y el microprocesador programable denominado AVR, generalmente uno de marca, ATMEL [8] . A pesar de que las placas de Arduino tienen muchas variantes, de forma general el hardware es similar[5], por esa razón se realizará una síntesis de los principales componentes físicos de una tarjeta, según se puede observar en la Figura 1.2.

[4] Open Source, en inglés

[5] Para el caso de este proyecto se usa como referencia el Arduino UNO.

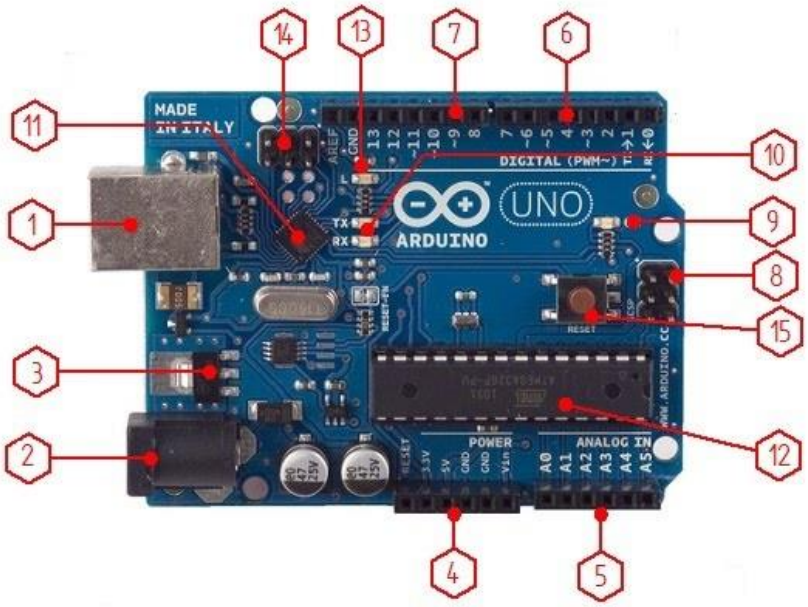

Figura 1.2. Hardware de Arduino [9]

Tomando en cuenta la Figura 1.2, los principales componentes físicos del Arduino, se encuentran numerados y se citan a continuación:

- Conector USB. (1)
- Conector de Adaptador (2)
- Regulador de 3.3VDC (3)
- Salidas de Energía (4)
- Entradas Analógicas (5)
- Pines Digitales (6 y 7)
- Pines de programación ICSP (8)
- Indicador LED de alimentación (9)
- LEDs RX - TX (10)
- Microcontrolador de interfaz (11)
- Microcontrolador de Gestión (12)
- LED (13)
- Pines ICSP Atmega16U2 (14)
- Botón de reinicio (15)

De estos componentes, se hace énfasis en cuatro, por razones de relación directa al objetivo de este trabajo, los mismos que se detallan a continuación.

1.4.2. CONECTOR USB. (1)

Este conector tiene tres propósitos fundamentales para Arduino:

- *Energización de la tarjeta Arduino.* Aprovechando las propiedades del estándar USB, que transporta energía por medio de dos líneas a dos pines terminales, que en el caso particular de este proyecto es desde el PC, hacia el dispositivo destino, la placa de Arduino.

  El voltaje que conduce es de 5 VDC, con una corriente máxima de 500 mA (0,5 A), lo que corresponde a 2,5 W como máximo, para las versiones 1.0 a 2.0. Con la USB 3.0, la corriente puede ser de hasta 1 A por puerto, lo cual da una potencia de 5 W [10].

- *Transferencia de software*. Puesto que la IDE de Arduino se encuentra instalada en una PC desde donde se desarrolla la programación y compilación de la placa. El conector y cable USB sirven para la trasmisión de los programas compilados hacia la memoria flash que contiene el microcontrolador Atmega de la placa Arduino, actuando como puerto de consola o puerto Debug.

- *Transferencia de datos.* Tomando en cuenta que la placa Arduino puede ser programada como receptor de datos de los sensores conectados a ella. Esos datos pueden ser transferidos al PC para darles el uso que se requiera. Algunos programas pueden ser bidireccionales, pueden ingresar, se procesan en la PC y luego se envían, por medio del puerto Serie (UART[6]), para la ejecución de alguna la aplicación específica.

  Con relación a la comunicación, los datos son transportados por pulsos de energía de 1_L y 0_L., esos datos lógicos se logran con la variación de voltajes. Por ejemplo, para ceros lógicos (0_L) la señal varía de 0 a 0.3 V y para unos lógicos (1_L), la señal es de 2.8 a 3.6 V, en las versiones USB 1.0 y 1.1. En la versión 2.0, la precisión es mayor, pues los voltajes son similares a las versiones anteriores pero con una variación de ±400 mV.

Es importante resaltar que las tarjetas Arduino, de igual forma que cualquier circuito electrónico tiene un consumo de energía propio del sistema. Ese valor es distinto para cada

[6] UART acrónimo de Universally Asynchronous Receiver Transmitter, puerto serial de transmisión asincrónica.

placa, pero en el caso de este trabajo se usa el Arduino UNO, que tiene un consumo de entre 46mA a 50mA en vacío, o sea, sin cargas adicionales [11].

1.4.3. CONECTOR ENERGÉTICO (2)

Es el conector de color negro, que permite a la placa Arduino recibir la alimentación autónoma de fuentes DC, como baterías o fuentes reguladas de 9V hasta 12 V. Aunque la tarjeta puede soportar hasta 20V, por poco tiempo, no se recomienda hacerlo por mucho porque se puede dañar la placa. Por esa razón es usual que la mayor parte de las placas Arduino trabajen con fuentes de entre 6 y 12 voltios [12].

1.4.4. REGULADOR DE 3.3VDC (3)

Puesto que el voltaje de trabajo de Arduino es de 5V a 12V, pero para poder interactuar con dispositivos externos, como los sensores o actuadores, algunos de bajo consumo como el módulo ESP8266, es necesario que provea un voltaje de alimentación para aquellos dispositivos, que requieren un voltaje de 3.3VDC. Eso se logra por medio del chip regulador AMS1117, el que se encarga de bajar el voltaje de 5V o mayores, recibidos por medio del USB o las fuentes DC, a 3.3Vdc. Mientras Arduino trabaje con los 5V a 12V, el chip AMS1117, puede trabajar con normalidad, pero si existen voltajes mayores a 12VDC, el chip se sobrecalienta, por lo que corre el peligro de dañarse.

1.4.5. PINES

Los pines son las entradas y salidas que están en la placa Arduino. Existen de tres tipos:

- De energía.
- Entradas y Salidas Analógicas.
- Entradas y salidas Digitales, de Modulación y de Referencia.

Para el caso particular de este proyecto se analizan las salidas de energía y en parte las Entradas y Salidas digitales.

1.4.6. SALIDAS DE ENERGÍA (4)

Son los pines que proveen de energía de 5VDC y 3,3VDC. Aquellos pines proveen de voltaje a componentes externos al Arduino como sensores y/o actuadores de bajo consumo. Se pueden identificar las salidas de acuerdo al voltaje marcado en la placa.

Cabe resaltar que esas salidas, solo pueden proveer una corriente de consumo externo dependiendo de las fuentes que alimente la placa. Por ejemplo, si se trata de la fuente

USB, la corriente máxima es de 500mA (nominal) y si es por el plug de alimentación independiente, entonces, puede ser de 500 mA a 1000 mA (1A). De ahí que cada salida puede proporcionar una corriente de 40mA, por lo que sumadas todas las salidas pueden proveer un valor estimado de 200mA [13]. A consumos mayores la placa tiene a sobrecalentarse lo que deteriora o daña la placa.

Otros pines fundamentales de las placas de Arduino son los varios pines GND[7] o tierra. En la placa Arduino UNO, tiene tres salidas, dos en el lado izquierdo y uno en el lado derecho.

1.4.7. PINES DIGITALES (6 Y 7)

Son los pines marcados con los numerales de 0 al 13, que también se denominan GPIO[8], en el caso de Arduino Uno. Pero en el caso de otras tarjetas Arduino, pueden ser de un número mayor o menor cantidad. Estos pines pueden ser utilizados tanto como entradas o salidas digitales para propósito general, según la programación efectuada por el usuario a cada pin. Trabajan usando tecnología TTL, por eso los voltajes para nivel bajo es de 0V y para nivel alto de 5V [14].

Los pines nominados 0 y 1 están marcados con las iniciales Rx y Tx[9] respectivamente, lo que se refiere a que se usan para la comunicación serie como receptor y transmisor. Se los usa, por ejemplo para conectar actuadores de comunicación como módulos bluetooth, dispositivos Wi-Fi o para conectar otro Arduino como esclavo [15]. El programador debe tomar en cuenta que esos pines no deben estar conectados al momento de ser usados por el USB durante la programación.

El resto de pines sirven para ser programados como entradas para interrupciones. En el caso de Arduino UNO el pin 2 es para la interrupción 0 y el pin 3 para la interrupción 1. Los pines digitales que están marcados con el signo (~) se usan para modulación PWM[10]. Otros pueden ser emulados como salidas analógicas por medio de los pines digitales. Arduino UNO, no posee salidas analógicas puras por lo que se deben emular. Sin embargo, otras placas sí tienen salidas analógicas puras mediante dos circuitos D/A o DAC [16], como es el caso de Arduino DUE.

[7] GND son las siglas en inglés de tierra.
[8] GPIO acrónimo de General Purpose Input Output o Entradas o Salidas de Propósito General.
[9] Rx siglas de Receptor y Tx siglas de Transmisor.
[10] PWM acrónimo de Pulse Width Modulation por sus siglas en Ingles, modulación por ancho de pulso

1.4.8. MICROCONTROLADOR DE GESTIÓN (12)

Se trata del principal circuito del Arduino, que por lo general es de la línea Atmega, de la empresa ATMEL. El microcontrolador es el dispositivo que permite recibir, almacenar, procesar y ejecutar la programación efectuada por usuario o programador. Por esa razón, se constituye en la parte central de las placas Arduino, por lo que contiene la CPU, las memorias RAM y ROM, el oscilador del reloj, los puertos de entrada y salida, temporizadores, convertidor A/D, circuito de comunicación serial (USART, SPI[11]), además de otros. Aquellos circuitos se pueden apreciar en forma de bloques en el esquema de la Figura 1.3.

En definitiva, el microcontrolador brinda a la placa Arduino la funcionalidad que la distingue. Así es que para lograr la programación del chip se usa la IDE de Arduino con su propio lenguaje de comunicación.

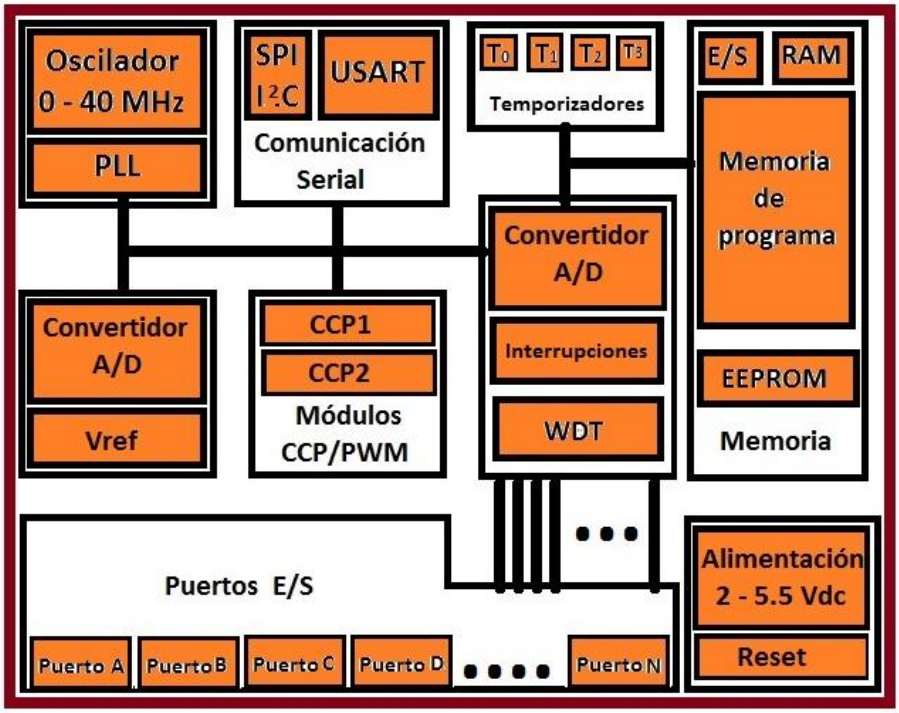

Figura 1.3. Esquema del hardware de un microcontrolador ATMEL

1.4.9. ENTORNO DE PROGRAMACIÓN

Puesto que Arduino, es una placa de gestión comandada por microcontrolador, la programación del mismo, debe ser por medio de una plataforma de gestión y control.

[11] SPI acrónimo de Serial Perimetral Interface o Interfaz Serial Periférico

Normalmente esta plataforma de gestión era un software desde el que se podía programar al microprocesador ya sea en lenguaje ASSEMBLER[12] o algún lenguaje derivado del lenguaje BASIC, como C o C++. Esa programación se la transfería al microcontrolador por medio de circuitos de escritura o comúnmente denominados “quemadores”, a la memoria del micro.

Actualmente con el desarrollo de Arduino, los microcontroladores que son integrados a las placas, se los programa por medio de una plataforma estándar que se le denomina IDE[13]. Esa resulta ser una herramienta de programación, desarrollada exclusivamente para Arduino, que tiene un lenguaje estandarizado para todas las tarjetas. Fue desarrollada en JAVA, por lo que su lenguaje está basado en la programación C o C++. Puesto que ese tipo de lenguaje es el más común para los usuarios, la IDE de Arduino resulta ser una plataforma de gestión, amigable y accesible a todo tipo de programadores con conocimientos básicos en BASIC.

Las principales características de la herramienta son el desarrollo, edición y compilación de programas en lenguaje Arduino. Después de probados, revisados y aprobados, los programas, posteriormente se los puede cargar o subir a la memora del microcontrolador de la placa, para finalmente ser ejecutados en diversos proyectos de las aplicaciones diseñadas por el usuario.

Cabe resaltar, que tanto Arduino, como la plataforma de gestión IDE son de hardware y software libre, por lo que es posible que diversos usuarios, tanto especialistas como aficionados puedan colaborar y aportar en el desarrollo de diversas aplicaciones. Por esa razón, existe la llamada comunidad MAKER, que es un grupo extenso de personas enfocadas en realizar mejoras y nuevos aportes en la plataforma de Arduino. Eso ha permitido que las aplicaciones sean mejoradas y para un sin número de aplicaciones, según las sean las necesidades existentes.

Entre las varias utilidades de la comunidad, está la constantemente renovación de la información a la página Web de Arduino. En la página se pueden hallar varios ejemplos prácticos y prediseñados los mismos que sirven como base para que otros usuarios los mejoren y apliquen, según sus necesidades particulares. Entre algunos de esos programas se encuentran los drivers de instalación de los sensores y actuadores de Arduino, los

[12] Es el lenguaje ensamblador de programación microprocesadores o microcontroladores.
[13] Es el acrónimo de Integrated Development Environment o en español Ambiente Integrado de Desarrollo.

mismos que se realizan en lenguaje C, C++ o JAVA. Otras Utilidades fundamentales son las librerías de funcionamiento de los distintos dispositivos externos, además de sus respectivos lineamientos de instalación, uso, condicionamientos y ayudas de usuario. Entre aquellos programas y funciones se encuentran el software de instalación y funcionamiento que reemplazará al firmware del módulo Wi-Fi, ESP8266.

1.4.10. ESTRUCTURA Y LENGUAJE DE PROGRAMACIÓN

Basado en el esquema del lenguaje BASIC, la IDE de Arduino es una herramienta con estructura similar y de fácil utilización para una gran cantidad de usuarios. Cabe resaltar que es necesario tener un conocimiento básico del lenguaje BASIC, para poder efectuar la programación de Arduino, ya que no es más diferente que una programación en lenguaje típico de C y/o C++.

Los programas diseñados en Arduino se los denominan *sketches*. Los *sketches* tienen una estructura simple y secuencial, compuesta en defecto por tres partes principales:

- La declaración de librerías y variables
- La función *setup()*
- La función *loop()*

La estructura general de las mismas se procede a exponer a continuación [17] :

```
//línea comentarios
/*bloque de comentarios*/
//Encabezado
Librerías;
Variables;

//funciones
void setup()
    {
        Programación;
    }

void loop()
    {
        Programación;
```

}

Los comentarios como en cualquier programación de tipo BASIC, se inician con el uso de las "/" para el caso de una línea de comentarios y "/*"para dar inicio a un bloque de comentario el que termina en una "*/".

El *Encabezado,* contiene las librerías y variables. Las variables pueden ser ejecutadas dentro del programa general o las estructuras internas del mismo. Las librerías en cambio son para la gestión dentro del programa interno como intermediarias entre los sensores o actuadores externos, muy necesarias para las aplicaciones que requiera el usuario.

La función *setup()* es la encargada de recoger la configuración general del programa, como aplicación de variables, parámetros de configuración, la correlación entre estructuras secuenciales y otras funciones dependientes [18]. Dentro de esta función se determina la configuración general del proyecto que el usuario desee desarrollar, lo que incluye la programación la comunicación entre dispositivos y los datos entrantes, como velocidades de trasmisión, tipo de transmisión y en el caso particular de este proyecto los parámetros de red de la Wlan a usar.

La función *loop()* es la que contiene el programa que se ejecutará cíclicamente, razón por la cual se denomina con el termino loop o bucle. Se encarga de la lectura de entradas, reconocimiento y procesamiento de datos, la activación de salidas, análisis de procesos y las derivaciones a otras funciones. En ese punto, el usuario puede incluir funciones y estructuras propias para lo que requiera su programa.

Además, el usuario puede hacer uso de la ayuda que existe la página de Arduino, la que integra una multitud de aportes de la comunidad MAKER. Por tratarse de una plataforma de código abierto, los aportes permiten acceder a bibliotecas, librerías y datos para el manejo de las placas y componentes externos de Arduino cuando el usuario requiera.

Muchos de esos programas prediseñados son para el control y manejo de la EEPROM, Ethernet, pines de entrada y salida, reset, tarjetas SD, servos, GPS y en particular para este proyecto las ayudas de Wi-Fi en el ESP8266 [19]. En caso de no existir el software se puede solicitar ayuda a la comunidad desde el portal [20].

En fin, la clave de la gran acogida en el uso de la plataforma Arduino y su IDE, es la programación estructurada y secuencial. La "inteligencia" de Arduino depende mucho de

la imaginación y capacidad de programación del usuario. Entonces haciendo uso de la ayuda de Arduino, las librerías y los instaladores que existen; se plantea el objetivo de este proyecto, que es modificar el firmware del módulo ESP8266, para extender los alcances de su aplicación dentro de una red Wlan y gestionado por medio de una interfaz de usuario en Android.

1.5. MÓDULO ESP8266

Como se ha explicado a breves rasgos en los apartados anteriores, con la demanda de Arduino en múltiples proyectos electrónicos y robóticos, también ha surgido la necesidad de seguir desarrollando y fabricando nuevos dispositivos sensores y actuadores que le permitan ampliar su campo de acción, aprovechado la factibilidad de que es una plataforma de hardware y software libre.

Uno de los campos, sobre los que se han desarrollado y con gran éxito, son las redes inalámbricas. Actualmente la mayor parte de tecnologías desarrolladas se enfocan en la gestión y control inalámbrico de redes y/o dispositivos, en el caso de Arduino se han desarrollado dispositivos actuadores que trabajan bajo las normas de redes IEEE 802.11 y IEEE 802.15.4 [21]. Así es que existen varios actuadores que permiten que Arduino logre mayor interactividad con dispositivos que se comunicaban de forma inalámbrica.

Entre uno de esos dispositivos actuadores está el módulo ESP8266, que aparece en agosto del 2014. La empresa China Espressif Systems, situada en Shangai, diseña y fabrica el chip ESP8266EX, el mismo que pasa a formar parte del módulo ESP-01 desarrollado por la empresa AI-Thinker. Desde ese momento el ESP-01, es fabricado como un módulo Wi-Fi para apoyo de Arduino, el que cuenta con un stack completo de comandos en el protocolo TCP/IP. Para ello cuenta con un microcontrolador Tensilica Xtensa LX106 [4] , bajo el esquema SoC[14], pero que trabaja con el uso de comandos Hayes o AT.

El módulo fue y es, un actuador para Arduino en lo relacionado a las Wlan, por cuanto se desempeña con las normativas IEEE 802.11 b, g, n. Tomando en cuenta lo antedicho y aprovechando las características constitutivas del módulo, se plantea el uso del mismo para aprovechar su funcionalidad en el presente proyecto de titulación. Entonces a

[14] SoC, acrónimo de System on Chip, sistema sobre el circuito

continuación se detalla la parte tangible e intangible del módulo, que es parte del estudio de este trabajo.

1.5.1. HARDWARE DEL MÓDULO ESP8266

El módulo ESP8266, está compuesto por algunos componentes físicos importantes para su funcionamiento, los cuales le permiten efectuar de forma precisa su trabajo en el campo de las comunicaciones inalámbricas. El módulo, al ser pequeño físicamente, está conformado de pocos elementos muy importantes, los que se pueden apreciar en la Figura 1.4, y se citan a continuación, para un análisis detallado posterior.

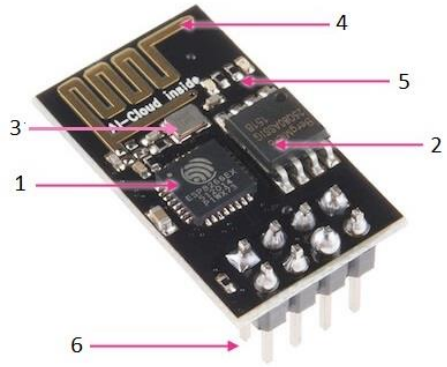

Figura 1.4. Hardware del ESP8266 [22]

- El microcontrolador ESP8266EX (1).
- Memoria externa (2)
- Cristal externo 24 MHz a 52MHz (3)
- Antena microstrip de 2,4,GHz (4)
- Leds de activación y comunicación (5)
- Pines de energía y programación (6)

Como parte inherente del trabajo de este proyecto es estudiar con detenimiento el hardware[15] y software[16] del módulo ESP8266, para poder solventar su funcionalidad. Por lo que se procede a describir cada componente del módulo según la numeración expuesta en la Figura 1.4.

1.6. MICRO ESP8266EX (1)

El microchip ESP8266EX, es un microcontrolador que contiene el firmware instalado de fábrica, basado en la tecnología SoC, lo que le permite tener una amplia funcionalidad en

[15] Hardware, conformación física del módulo
[16] Software, características o arreglos funcionales del módulo

las redes Wlan. Consta de 33 pines, los que tienen su respectiva función a fin de brindar su amplia aplicación. Por lo que al tratarse de un chip con un rango amplio de funciones, es importante analizar las más relevantes en lo respecta a lo relacionado al presente proyecto. Entonces, por eso se procede a mostrar en la Figura 1.5, los pines que conforman chip ESP8266EX, con su respectiva numeración y denominación.

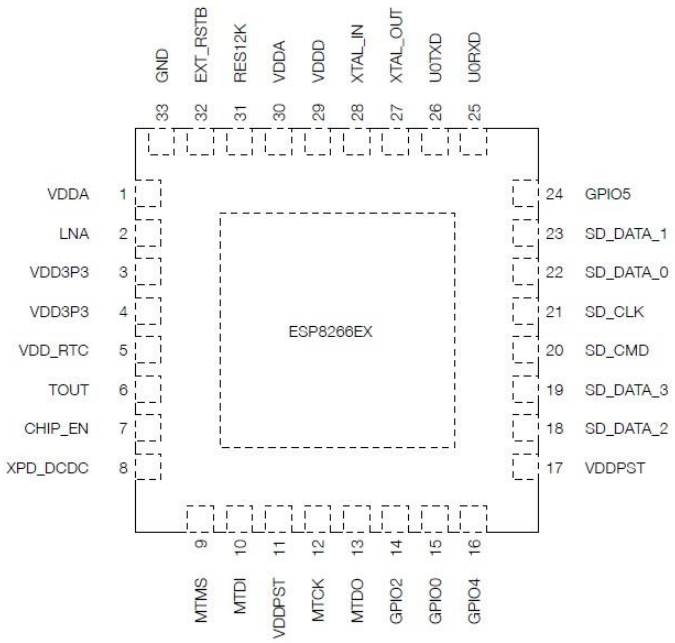

Figura 1.5. Distribución de pines del ESP8266EX [4]

Cada pin del ESP8266EX está diseñado para cumplir una o más funciones específicas, las mismas que se describen de forma general en la Tabla 1.1, que se presenta a continuación.

Tabla 1.1. Funciones de los pines del ESP8266EX [4]

Pin	Nombre	Tipo	Función
1	VDDA	P	Fuente Analógica de 2.5V – 3.6V
2	LNA	I/O	Interface de Antena de RF Chip con impedancia de salida de 39 + 6j Ω. Se sugiere que se conecte a la antena de red conservando una juntura tipo π.
3	VDD3P3	P	Amplificador de potencia (2.5V – 3.6V)
4	VDD3P3	P	Amplificador de potencia (2.5V – 3.6V)
5	VDD_RTC	P	N.C. (no conectar 1.1 V)
6	TOUT	I	Pin ADC. Este puede ser usado para verificar la fuente el voltaje del VDD3P3 (pin3 y pin4) y la entrada de

			voltaje de TOUT (pin6). Pero esas dos funciones no pueden ser usadas simultáneamente.
7	CHIP_EN	I	Chip disponible: Alto: On; chip trabaja correctamente Bajo: Off; pequeña corriente consumida
8	XPD_DCDC	I/O	Suspensión total(debe estar conectada a EXT_SRTB); GPIO16
9	MTMS	I/O	GPIO14; HSPI_CLK
10	MTDI	I/O	GPIO12; HSPI_MISO
11	VDDPST	P	I/O Digital Voltaje (1.8V – 3.6V)
12	MTCK	I/O	GPIO13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX durante la programación de la flash; GPIO2
15	GPIO0	I/O	GPIO0[17] ;SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	I/O Digital Voltaje (1.8V – 3.6V)
18	SDIO_DATA_2	I/O	Conectado a SD_D2 (R = 200 Ω);SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Conectado a SD_D3 (R = 200 Ω); SPIWP; HSPIHD; GPIO10
20	SDIO_CMD	I/O	Conectado a SD_CMD (R = 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Conectado a SD_CLK (R = 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Conectado a SD_D0 (R = 200 Ω); SPI_MOS0; GPIO7
23	SDIO_DATA_1	I/O	Conectado a SD_D1 (R = 200 Ω); SPI_MOS1; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART RX durante la programación de la flash; GPIO3
26	U0TXD	I/O	UART TX durante la programación de la flash; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Salida de conexión del oscilador de cristal, puede ser usado para proveer la entrada del reloj BT.
28	XTAL_IN	I/O	Entrada de Conexión al oscilador de cristal
29	VDDD	P	Voltaje analógico 2.5 V – 3.6 V

[17]GPIO acrónimo General Purpose Input/Output, entradas y salidas de propósito general

30	VDDA	P	Voltaje analógico 2.5 V – 3.6 V
31	RES12K	I	Conexión serial con una resistencia de 12KΩ y tierra.
32	EXT_RSTB	I	Señal de Reset Externo (Voltaje bajo)
33	GND		Tierra general

El microcontrolador posee 33 pines, como ya se ha mencionado y cada uno con varias funciones, por lo que es necesario estudiar de forma detallada algunas de ellas para entender cómo se puede realizar las debidas adaptaciones tanto en software como en hardware y lograr el propósito de este trabajo, o sea, modificar el firmware original del chip por medio de la programación en Arduino. Por esa razón y basándose en la información de sustento se presenta en la Figura 1.6, el diagrama de funcionalidad del ESP8266EX.

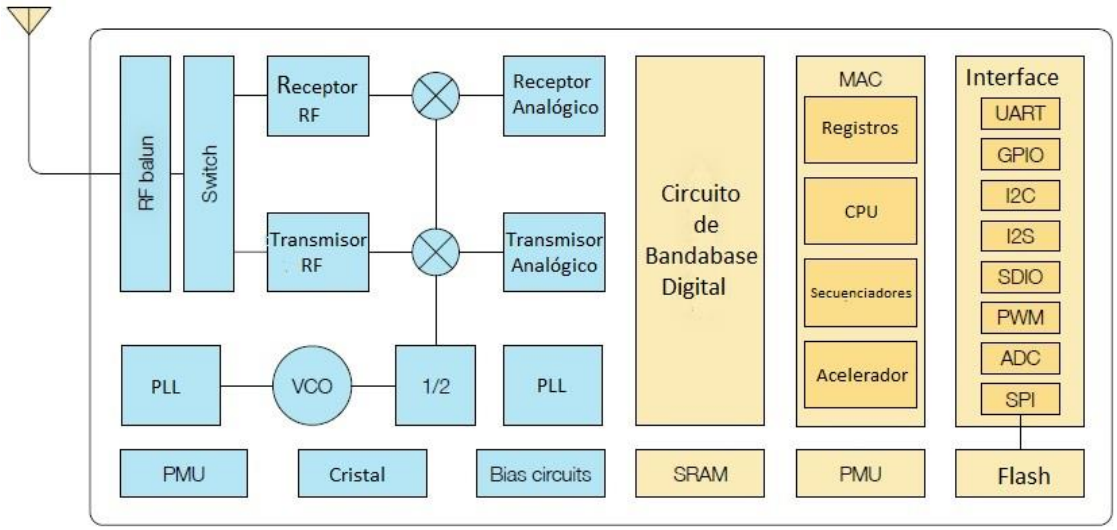

Figura 1.6. Esquema Funcional del ESP8266EX [4]

1.6.1. CPU

El ESP8266EX está compuesto por un procesador Tensilica L106, de 32 bits con tecnología RISC[18]. Se destaca por tener un extra bajo consumo de potencia y un alcance máximo de velocidad del reloj de 160 MHz. El sistema operativo es RTOS[19], el paquete de comandos de la programación y desarrollo de aplicaciones Wi-Fi consumen el 80% de la potencia.

[18] Acrónimo de Reduced Instruction Set Computer en inglés o Computador con Conjunto de Instrucciones Reducido en español

[19] RTOS es acrónimo de Real Time Operating System en inglés o Sistema Operado a Tiempo Real en español

El CPU incluye los siguientes interfaces:

- Interfaces para la Memoria RAM/ROM programable (iBus[20]) que está conectada con los controladores de memoria y puede está relacionada con la memora flash.
- El interfaz de datos RAM (dBus[21]), que puede estar conectada con los controladores de memoria.
- Interfaz AHB[22] que puede ser usado para explorar los registros.

1.6.2. MEMORIA

El ESP8266EX al ser un dispositivo de tecnología SoC, integra los controladores de memoria, los que están incluidos en las memorias SRAM y ROM. El MCU[23] puede acceder a la memoria por medio de los interfaces iBus, dBus y AHB. Todas las unidades de memoria pueden accederse por medio de requerimientos, lo que quiere decir que se puede correr una secuencia de datos específica al tiempo que se requiera y recibirá los estamentos del procesador.

El espacio asignado para los usuarios en la SRAM, es:

- Menor a 50kB de memoria RAM, que es cuando el micro trabaja en Modo Estación y se conecta a un router.
- El máximo espacio de uso es cuando trabaja en modo Acumulador de Datos[24], que le permite que usar hasta 50kB.
- Como el micro es SoC, la ROM no es programable, por lo que es necesario tener una memoria flash SPI externa, para insertar y acceder a algún programa.

1.6.3. FLASH EXTERNA

El micro ESP8266EX, permite el acceso por medio de una SPI a la Flash externa, la que sirve para almacenar los programas desarrollados por el usuario y respaldar la información en teoría hasta 16MB [4].

Los valores mínimos de uso de la Flash se muestran a continuación.

- Desactivado (OTA[25]): mínimo 512 kB.

[20] iBus interfaz de bus que se conecta a la memoria.
[21] dBus interfaz de datos que se conecta a la memora RAM.
[22] AHB es el acrónimo Advanced High-performance Bus en español Bus avanzado de alto rendimiento.
[23] MCU acrónimo de Microcontroller Unit
[24] Modo Heap + Data
[25] OTA acrónimo de Over the Air o Por el aire

- Activado (OTA): mínimo 1 MB.

Cabe destacar que la SPI puede trabajar de tres formas:

- SPI Estándar
- SPI Dual
- SPI Quad.

Es importante recalcar que cuando se desea *flashear*[26] con un nuevo firmware al ESP8266, el modo SPI especificado debe ser el modo estándar, de otra manera el firmware o programa puede que no se logre instalar correctamente.

1.6.4. RELOJ DE ALTA FRECUENCIA

El reloj de alta frecuencia del ESP8266EX, es generado por un oscilador de cristal que está dentro del chip y que trabaja en los rangos de frecuencias de 24 MHz a 52 MHz. Su función es impulsar los mezcladores de transmisión como el de recepción.

La calibración interna dentro del oscilador de cristal permite tener una amplia gama de frecuencias que se pueden usar. Sin embargo, la calidad del cristal es un factor determinante al momento de evitar desfase por ruido y tener una buena sensibilidad en la transmisión y recepción Wi-Fi. Las especificaciones que debe cumplir del reloj son las que se mencionan en la Tabla 1.2.

Tabla 1.2. Parámetros del reloj de alta frecuencia [4]

Parámetros	**Símbolo**	**Min**	**Max**	**Unidad**
Frecuencia	FXO	24	54	MHz
Capacitancia de carga	CL	-	32	pF
Capacitancia de movimiento	CM	2	5	pF
Resistencia serie	RS	0	65	Ω
Tolerancia de Frecuencia	ΔFXO	-15	15	Ppm
Frecuencia vs temperatura (-25 °C – 75°C)	ΔFXO/Temp	-15	15	Ppm

1.6.5. RADIO

El ESP8266EX está compuesto de un circuito de radio que integra las normativas de la IEEE 802.11 b, g, n, por lo que consta de los siguientes bloques:

[26] Modificar o cambiar el software de gestión o por otro similar y permitido por la empresa desarrolladora del hardware.

- Receptor de 2.4 GHz
- Transmisor de 2.4 GHz
- Reloj de alta frecuencia y cristal oscilador
- Administrador de potencia
- Reguladores y discriminador de frecuencia

1.6.6. RECEPTOR DE 2.4 GHZ

El receptor de 2,4 GHz convierte las señales de RF, en banda base en cuadratura al dominio digital por medio de dos circuitos ADC[27] de alta resolución y velocidad. Para adaptarse a las condiciones variables del canal el receptor tiene circuitos de cancelación y los filtros de banda base, filtros de RF, control automático de ganancia (AGC), sistema de compensación de DC [4].

1.6.7. TRANSMISOR DE 2.4 GHZ

El transmisor transforma las señales digitales a señales de banda base en cuadratura a 2,4 GHz y las impulsa a la antena por medio de un amplificador CMOS de alta potencia.

La función de calibración digital mejora aún más la linealidad del amplificador de potencia, lo que permite un mejor rendimiento en la entrega de potencia Tx con un promedio de +19.5 dBm para transmisión en el estándar 802.11b y a +18 dBm para transmisión en el estándar 802.11n (MSC0).

Se integran calibraciones adicionales para compensar cualquier imperfección de la transmisión de radio, como:

- Fugas del portador
- Emparejamiento de fase I / Q
- Banda base No lineales

Estas funciones reducen el tiempo de prueba de la transmisión y evitan pruebas innecesarias.

[27] ADC acróstico de Analogic Digital Converter, o Convertidor Analógico Digital

1.6.8. CANALES DE FRECUENCIA

El transceiver RF, soporta los canales de la normativa IEEE 802.11 b, g, n, o sea, los 14 canales[28]. Los canales trabajan con un ancho de banda de 20MHz y una separación entre canales de 5MHz. Cabe resaltar que el canal 14 se usa únicamente en Japón y los canales tienen una separación de 12MHz [23].

1.6.9. GENERADOR DE RELOJ

El generador de reloj crea las señales en cuadratura de 2,4 GHz para el transmisor. Usando el oscilador de alta frecuencia descrito anteriormente y otros componentes electrónicos que están integrados en el chip, como inductores, varactores, filtros de lazo, reguladores y divisores de voltaje lineal.

El generador de reloj tiene un circuito integrado de calibración y autoprueba. El ruido de fase es optimizado por el circuito con algoritmos de calibración patentados para garantizar el mejor rendimiento del receptor y transmisor.

1.6.10. WI-FI

El chip ESP8266EX, tiene implementado en su firmware el protocolo TCP/IP, junto al set completo WLAN MAC de 802.11 b, g, n.

Tiene integradas las operaciones como Estación (STA) y Punto de Acceso (AP) en el Set Básico de Servicios (BSS)[29], bajo la Función de Control Distribuido (DCF)[30].

La administración de energía se maneja por medio de la interacción mínima del host para minimizar el período de servicio activo [4].

1.6.11. RADIO WI-FI Y BANDA BASE

La banda base y la radio Wi-Fi del ESP8266EX, admiten las siguientes funciones:

- El protocolo 802.11 b y 802.11 g
- El protocolo 802.11 n MCS0-7 en ancho de banda de 20 MHz.
- Intervalo de protección 802.11 n 0.4 µs.
- Velocidad de datos hasta 72,2 Mbps.
- Recibe STBC 2 x 1.

[28] Los canales del estándar de 2,4 GHz, van desde 2,4000 – 2,4835GHz
[29] BSS acrónimo de Basic Service Set
[30] DCF acrónimo de Distributed Control Function

- Potencia de transmisión hasta 20,5 dBm.
- Potencia de transmisión ajustable.

1.6.12. MAC WI-FI

El ESP8266EX integra Wi-Fi MAC lo que le permite aplicar funciones de protocolo de bajo nivel automáticamente, para lo que se implementa de software que le permite trabajar con:

- Dos interfaces Wi-Fi virtuales.
- Modo de estación, Access Point, modo promiscuo del BSS de la infraestructura.
- Solicitud para enviar (RTS), Borrar para enviar (CTS) y ACK de bloqueo inmediato.
- Desfragmentación.
- Seguridad CCMP (CBC-MAC, modo contador), TKIP (MIC, RC4), WEP (RC4) y CRC.
- Monitoreo automático (hardware TSF[31])
- Compatibilidad y coexistencia con Bluetooth de antena doble y única con conexión simultánea, capacidad de recepción opcional (Wi-Fi/ Bluetooth).

1.6.13. CARACTERÍSTICAS GENERALES DEL RADIO WI FI

El ESP8266EX cumple con los parámetros de transmisión citados en la Tabla 1.3, datos que son de pruebas realizadas a temperatura ambiente, con un voltaje de 3,3 V.

Tabla 1.3. Características generales del Wi Fi [4]

Parámetro	**Min**	**Típico**	**Max**	**Unidad**
Frecuencia	2412		2484	MHz
Impedancia de salida		39 + 6j		Ω
Potencia de salida para 72.2Mbps	15.5	16.5	17.5	dBm
Potencia de salida para 11Mbps	19.5	20.5	21.5	dBm
Sensibilidad				
DSSS, 1 Mbps		-98		dBm
CCK, 11Mbps		-91		dBm
1/2 BPSK 6 Mbps		-93		dBm
54 Mbps		-75		dBm
HT20, MCS7 (65 Mbps, 72.2 Mbps)		-72		dBm
Rechazo del canal adyacente				

[31] TSF acrónimo de Timing synchronization function

OFDM, 6 Mbps		37		dB
OFDM, 54 Mbps		21		dB
HT20, MCS0		37		dB
HT20, MCS7		20		dB

1.6.14. GESTIÓN DE ENERGÍA

ESP8266EX tiene un sistema avanzado de administración de energía, destinado a dispositivos móviles, electrónicos portátiles y aplicaciones de Internet de las cosas.

La arquitectura de bajo consumo opera en los siguientes modos:

- Modo activo: la radio del chip permanece encendida. El chip puede recibir, transmitir o escuchar.
- Modo suspendido (Mode-sleep): la CPU permanece operativa pero el Wi-Fi y la radio están desactivados.
- Modo de suspensión ligera (Light-sleep): la CPU y todos los periféricos están en pausa. Cualquier evento de actividad (MAC, host, temporizador RTC o interrupciones externas) activarán el chip.
- Modo de suspensión profunda (Deep-sleep): solo el RTC está operativo y todas las demás partes del chip están apagadas.

De forma general todas estas actividades de administración usan los consumos energéticos expuestos en la Tabla 1.4.

Tabla 1.4. Especificaciones de Energía del ESP8266EX [4]

Modo de Potencia	**Descripción**		**Consumo promedio**
Activo	Wi Fi TX	802.11b, CCK, 11Mbps, P_{out} +17 dBm	170 mA
		802.11g, OFDM, 54Mbps, P_{out} +15 dBm	140 mA
		802.11n, MCS7, P_{out} +13 dBm	120 mA
	Wi Fi RX	802.11b, 1024 bytes por paquete, -80 dBm	50 mA
		802.11g, 1024 bytes por paquete, -70 dBm	56 mA

		802.11n, 1024 bytes por paquete, -65 dBm	56 mA
Modem – sleep	CPU está trabajando		15 mA
Light – sleep			0,9 mA
Deep – sleep	Solo RTC trabajando		20 µA
Shut down			0,5 µA

1.6.15. INTERFAZ DE ENTRADA / SALIDA DE USO GENERAL

El ESP8266EX tiene 17 pines GPIO que se puede signar a varias funciones mediante programación de los registros apropiados. Cada GPIO PAD se puede configurar con pull-up o pull-down interno (XPD_DCDC solo puede configurarse con pull-down interno, otros GPIO PAD solo se pueden configurar con pull-up) o en alta impedancia.

Cuando se los configura como entrada, los datos se almacenan en registros de software. Esa entrada también se puede configurar para disparar por flanco o dispara por nivel las interrupciones de la CPU.

En resumen, las direcciones I/O son bidireccionales, no inversoras y triestdo, lo que incluye entrada y buffer de salida con entradas de control triestado. Esos pines, cuando funcionan como GPIO, se pueden multiflexar con otras funciones como I^2C, I^2S, UART, PWM y control remoto IR, etc.

1.6.16. INTERFAZ DE ENTRADA / SALIDA DIGITAL SEGURA

El ESP8266EX tiene SDIO[32] esclavo, cuyos pines, funciones y definiciones se describen en la Tabla 1.5, que admite SDIO v1.1 de 25 MHz y SDIO v2.0 de 50 MHz, y modo SD de 1 bit / 4 bit y modo SPI.

Tabla 1.5. Funciones SDIO [4]

Pin Nombre	Pin Número	I/O	Función
SDIO_CLK	21	I/O6	SDIO_CLK
SDIO_DATA	22	I/O7	SDIO_DATA
SDIO_DATA1	23	I/O8	SDIO_DATA1
SDIO_DATA_2	18	I/O9	SDIO_DATA_2
SDIO_DATA_3	19	I/O10	SDIO_DATA_3
SDIO_CMD	20	I/O11	SDIO_CMD

[32] SDIO acróstico Secure Digital Input Output o Entrada Salida Digital Segura

1.6.17. INTERFAZ DE PERIFÉRICOS EN SERIE

ESP8266EX tiene dos modos SPI por hardware, pero uno es utilizado para conectar con la memoria flash. Por lo que queda un solo SPI que puede ser usado como:

- Un SPI esclavo / maestro general.
- Un HSPI[33] esclavo / maestro general.

Las funciones de todos estos pines se pueden implementar mediante hardware. Los pines usados son el GPIO14 (CLK), GPIO12 (MISO[34]), GPIO13 (MOSI[35]) y GPIO15 (SS[36]).

1.6.18. INTERFAZ I^2C

El ESP8266EX no tiene hardware para I^2C, por lo que se debe emular por software. Por lo tanto, se puede emplear I^2C con casi cualquier pin GPIO, tomando en cuenta la una carga de trabajo para el procesador. Por defecto, la librería usa el GPIO4 y GPIO5 (SDA[37] y SCL[38]) para el I^2C. La velocidad máxima de transmisión es de 450kHz [24].

1.6.19. TRANSMISOR RECEPTOR UNIVERSAL ASINCRÓNICO

El ESP8266EX tiene dos interfaces UART, por hardware, cuyas definiciones se pueden observar en la Tabla 1.6.

- UART0 en pines 1 y 3 (TX0 y RX0).
- UART1 en pines 2 y 8 (TX1 y RX1).

Las transferencias de datos hacia y desde las interfaces UART se pueden implementar por medio de hardware. La velocidad de transmisión de datos a través de interfaces UART alcanza 115200 x 40 (4,5 Mbps).

La UART0 se puede utilizar para la comunicación y soporta el control de flujo. La UART1 solo transmite la señal de datos (TX), generalmente se usa para imprimir el registro.

Tabla 1.6. Funciones del UART

[33] HSPI acróstico de Hardware Serial Port Interface
[34] MISO acrónimo de Master Input Slave Output, Entrada Maestro Salida Esclavo, es la señal de entrada al dispositivo, que recibe datos desde otro integrado.
[35] MOSI acrónimo de Master Output Slave Input, Salida Maestro Entrada Esclavo, transmisión de datos hacia otro integrado.
[36] SS o CS, acrónimo de Slave Select, Selector Esclavo, habilita el integrado hacia el que se envían los datos.
[37] SDA acrónimo de System Data, es la línea de pulsos de reloj que sincronizan el sistema.
[38] SCL acrónimo de System Clock, es la línea por la que se mueven los datos entre los dispositivos.

Pin	Nombre	Número	I/O	Función
UART0	U0RXD	25	I/O3	U0RXD
	U0TXD	26	I/O1	U0TXD
	MTDO	13	I/O15	U0RTS
	MTCK	12	I/O13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Sin embargo, el pin 8, se emplea para conectar con la memoria flash, por esa razón en la práctica, el puerto UART1 sólo puede actuar como el pin Tx1 (sólo puede enviar, no recibir).

Por otro lado, al UART0 también se tiene acceso por los pines 15 y 13 (RTS0 y CTS0).

1.6.20. MODULACIÓN DE ANCHO DE PULSO

El ESP8266EX no tiene salidas PWM por hardware, pero se las puede emular por software. Eso se constituye teóricamente en una ventaja, pues, se pueden simular varias salidas PWM, usando cualquiera de todos los pines GPIO, pero se debe tomar en cuenta ello conlleva una carga sobre el funcionamiento del módulo ESP8266.

Entonces, por lo expuesto, el fabricante define ciertos pines que pueden trabajar como interfaces de salida PWM y son cuatro, a fin de evitar sobrecargas en el funcionamiento. Las definiciones de pines de las interfaces PWM se muestran a continuación en la Tabla 1.7. Cabe resaltar que de ser necesario el usuario puede ampliar lo cantidad de pines adicionales.

El rango de frecuencia PWM es ajustable y va desde 1000 μs a 10000 μs, es decir, entre 100 Hz y 1 kHz, aunque puede ser cambiada según la necesidad del usuario. Cuando la frecuencia es de 1 kHz, la relación de trabajo será 1/22727. Una resolución de más de 14 bits será logrado a una frecuencia de actualización de 1 kHz [4].

Tabla 1.7. Funciones PWM [4]

Pin	Número	I/O	Función
MTDI	10	I/O12	PWM0
MTDO	13	I/O15	PWM1
MTMS	9	I/O14	PWM2

GPIO4	16	I/O4	PWM3

1.6.21. CONVERTIDOR DE ANALÓGICO - DIGITAL

El ESP8266EX tiene integrado un circuito ADC de 10 bits de resolución. TOUT (Pin6) se define como interface ADC, que es un pin independiente de GPIO [24].

El rango de entrada del ADC es de 0-1V, pero si se aumentara la tensión a un valor superior a 1V se dañará el circuito ADC. En algunas placas o módulos que usan el ESP8266EX tienen divisores de voltaje que permiten ampliar el rango para poder medir de 0 - 3.3V.

Las siguientes dos medidas de voltaje, 1V o 3.3 V, se pueden implementar usando ADC (Pin6). Sin embargo, no se las puede implementar al mismo tiempo [4].

1.7. MEMORIA W25Q40BW - W25Q80BW (2)

Se trata de una memoria adicional, que está fuera del microcontrolador. Aquella memoria permite solventar la programación que hace el usuario sobre el módulo ESP8266.

Es una memoria SPI de 4 MB [25] y 8 MB [26], según la versión de ESP8266, la que permite almacenar los datos externos del módulo ESP8266. El rango de potencia de funcionamiento es de 1.65 V a 1.95V con una corriente de 4mA, cuando está activo el dispositivo y de 1µA, cuando está en reposo.

La memoria está organizada en 2048 y 4096 bloques de 256 bytes cada uno. Se puede programar 265 bytes cada vez. Los paquetes pueden ser borrados en grupos de 16 o sectores de 4KB, en grupos de 128 o bloques de 32 KB o grupos de 256 lo que corresponde a 64KB o totalmente si se desea. Lo más recomendable es utilizar bloques de 4kB para dar más flexibilidad a las aplicaciones.

La memoria soporta los estándares SPI que puede ser ejecutada de modo DUAL o QUAD SPI (XIP) tanto de recepción como de salida, por lo que le permite trabajar con funciones como: Reloj serie, Selector de chip, Serial Data I/O0 (DI), I/O1 (DO), I/O2 (WP), I/O3 (HOLD).
El SPI de la memoria de 40MB, soporta frecuencias desde 80MHz hasta 160 MHz (80 MHz x 2) en modo dual I/O y 320 MHz (80 MHz x 4) en modo QUAD I/O.

El SPI de la memoria de 80MB, soporta frecuencias desde 104MHz hasta 208 MHz (104 MHz x 2) en modo dual I/O y 416 MHz (104 MHz x 4) en modo QUAD I/O.

La transferencia de datos puede ser asincrónica de 8 a 16 bit de forma paralela a la memoria flash. La lectura es continua con 8 pulsos por lo que puede leer direcciones con cabeceras de 24 bits.

1.8. OSCILADOR EXTERNO (3)

Este es un chip oscilador que sirve para ejecutar aplicaciones que requieren de mayor precisión. El chip tiene un reloj de cristal de 24 MHz a 52 MHz y puede ser programado desde las entradas GPIO del ESP8266. Además, debe cumplir con los parámetros establecidos en la Tabla 1.8, que se presenta a continuación.

Tabla 1.8. Parámetros del reloj externo [4]

Parámetros	**Símbolo**	**Min**	**Max**	**Unidad**
Amplitud de voltaje	VXO	0.8	1.5	Vpp
Precisión	ΔFXO ext	-15	15	ppm
Ruido de fase @ 1KHz compensado			-120	dB/Hz
Ruido de fase @ 10KHz compensado			-130	dB/Hz
Ruido de fase @ 100KHz compensado			-138	dB/Hz

1.9. ANTENA 2.4 GHZ (4)

Es uno de los dispositivos primordiales del módulo ESP-01, pues sin ella el microchip ESP8266EX quedaría inservible. Trabaja bajo las normativas 802.11 b, g, n, por lo tanto, a una frecuencia de 2,4 GHz.

La antena es de tipo PBC o microstrip, la que sirve para transmitir y recibir (TX – RX). Sus medidas son de 24.7 × 14.4 × 11 mm [27] y se encuentra implantada sobre la placa del módulo ESP-01.

Adicionalmente, están los respectivos circuitos resonantes como capacitores, inductores y resistencias para acoplamiento. El acople es por medio de un arreglo resonante tipo “π” (L-C-L), como se puede observar en la Figura 1.7. La impedancia de acoplamiento de la

antena es de 50 Ω [28]. Esa normalización le permite al módulo ser lo suficientemente adaptable para conectar antenas de mayor sensibilidad. Eso es posible al cortar el canal de acoplamiento que es de 1,6mm y luego por medio de un pigtail, ser la entrada de una antena de mayor capacidad.

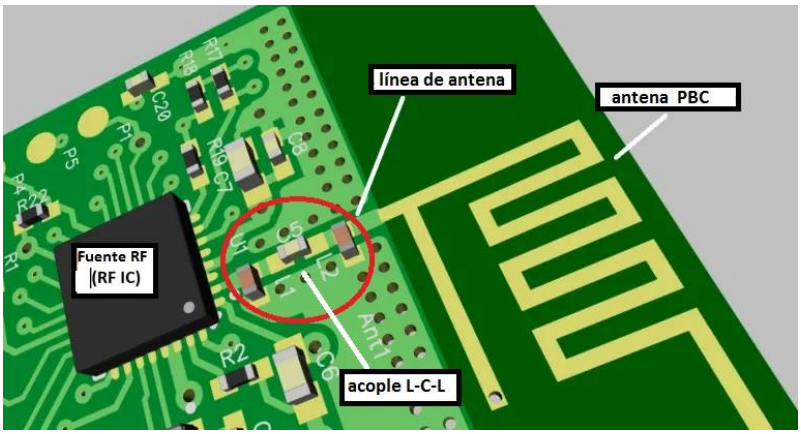

Figura 1.7. Acoplamiento de la antena del ESP8266

1.10. LEDS DE ACTIVIDAD (5)

Los leds de actividad son dos. El primero que se encuentra en el extremo derecho según se observa en la Figura 1.8, es el led de encendido. Solo se activa cuando el ESP8266 recibe un voltaje de 3.3V y permanece así hasta cuando el módulo sea desconectado de su fuente.

El segundo es el led de comunicación está al lado izquierdo, como se aprecia en la Figura 1.8, se activa cuando el módulo transmite o recibe señales de datos, ya sea por medio de Wi-Fi o el puerto serial del ESP8266.

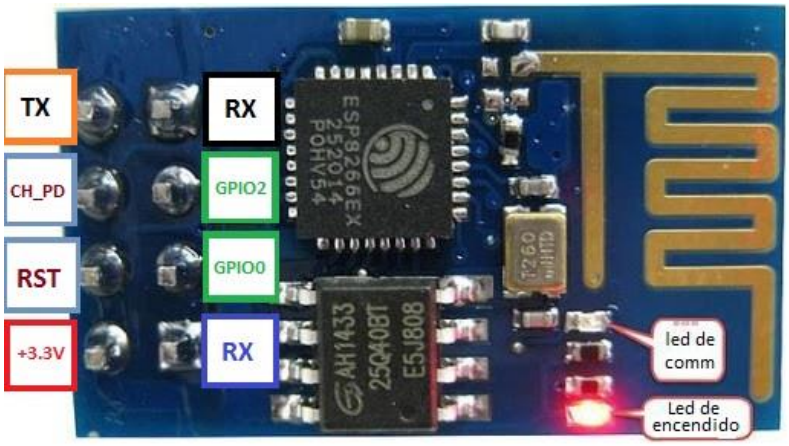

Figura 1.8. Conformación de leds y pines del ESP8266 [29]

1.11. PINES (6)

Los pines del módulo ESP8266 son ocho, los que se pueden observar en la Figura 1.8, antes expuesta, con sus respectivos nombres. Esos pines están directamente relacionados con los pines del micro ESP8266EX, el que anteriormente ya se analizó. Cabe destacar que cada uno de los pines del microcontrolador, tienen diferentes funciones, lo que permite que solo tres puedan ser manipulados y ejecutados por el usuario para las distintas operaciones del módulo ESP8266.

Los ocho pines del módulo ESP8266 son los siguientes:

- *VCC*: debe estar conectado a un voltaje de 3.3V, en caso de tener voltajes superiores puede soportar unos momentos, pero a grandes intervalos de tiempo puede sufrir una sobre carga y dañarse.
- *TX:* pin de transmisión serial usa el UART, usa el GPIO1 del microcontrolador.
- *RX:* está conectado al GPIO3 y es el pin de recepción por medio de UART.
- *CH_PD:* pin de programación activo se puede conectar de forma directa o por medio de una resistencia, al voltaje de 3.3V o 1 lógico (1_L).
- *RESET:* este pin debe estar conectado en 0V o 0 lógico (0_L).
- *GND:* está conectado a tierra.
- *GPIO 2 y GPIO 0:* estos pines son la entrada y la salida de propósito general. Normalmente están en configuración Pull-Up, pero para cargar un nuevo el firmware se los configura como Pull-Down. Estos pines definen el modo de boot con que se inicia el módulo. También decide si los pines TX/RX se utilizan para programar el módulo o para fines de entrada y/o salida en serie.

Para programar el módulo usando UART, se conecta el GPIO0 a tierra y GPIO2 a VCC o también se lo puede dejar abierto.

En el caso de usar el UART para Entradas/Salidas (I/O) en serie normal, se deja ambos pines abiertos sin conexiones a VCC ni a tierra.

Cabe resaltar que son pocos los pines del ESP8266EX que se pueden usar en la programación del módulo ESP-01, por eso a continuación, se procede a analizar las principales funciones de la relaciones de pines del módulo y del microcontrolador [24].

El microcontrolador tiene 17 pines GPIO (entradas/salidas digitales). Cuando actúan como entrada pueden reconocer tensiones entre 0V o 3.3V, la que es suministrada al GPIO. Como entrada, son tolerantes a 5V, por lo se puede utilizar una fuente de hasta 5.8V, sin ningún riesgo de dañarla.

En cambio cuando trabajan como salidas proporcionan una tensión de 0V (LOW, estado bajo) o 3.3V (HIGH, estado alto). También la corriente máxima que pueden proporcionar o absorber cada GPIO es de 12mA.

De los 17 GPIO (0 a 16) la mayor parte están destinados a varias funciones específicas del funcionamiento interno del ESP8266EX. Por lo que se citan algunas funciones a continuación.

- Seis GPIO (GPIO6 a GPIO11) se usan para conectar por SPI la memoria flash, por lo que quedan inhabilitados o sin uso para el usuario.
- Los GPIO0, GPIO2 y GIPO15 son los únicos que intervienen en el arranque del módulo ESP8266 y los que se pasa a analizar más adelante.
- Los GPIO1 y GPIO3 se los usa para la comunicación Serial (UART) como TX y RX respectivamente.

De forma general las resistencias de los pines GPIO0 a GPIO15 tienen el arreglo de Pull-Up, mientras que el GPIO16 tiene un arreglo resistencias de Pull-Down.

1.11.1. ARRANQUE (BOOT)

Para el arranque o boot del ESP8266 se tiene 3 modos:

- UART Bootloader, que se usa para subir un programa por UART a la memoria flash.
- Boot Sketch, que permite ejecutar el último programa subido a la memoria flash.
- SDIO, que no se usa cuando se programa con el IDE de Arduino.

Cada uno de los distintos modos se los debe configurar adecuadamente los pines GPIO15, GPIO0 y GPIO2, que son los únicos que puede usar el usuario, según la Tabla 1.9.

Tabla 1.9. Pines de arranque del ESP8266 EX

Modo	**GPIO15**	**GPIO0**	**GPIO2**
UART Bootloader	0V	0V	3.3V

Modo	GPIO15	GPIO0	GPIO2
Boot normal	0V	3.3V	3.3V
SDIO	3.3V	x	x

En cambio para determinar el proceso de arranque del módulo ESP8266, la gestión lo realiza de forma autónoma el software de la misma placa, pero en la mayor parte de veces el usuario debe tener en cuenta lo siguiente:

- El GPIO15, tiene por defecto un arreglo de resistencias Pull-Down, así que no se puede pasar a Pull-Up.
- El GPIO0, debe estar en HIGH (estado alto) durante el funcionamiento del módulo.
- EL GPIO2, no debe estar en LOW (estado bajo) durante el arranque el módulo.

En forma de resumen la distribución de los pines del microcontrolador ESP8266EX que se relacionan directamente con el módulo ESP8266 cuando se ejecuta el boot se destacan en la Tabla 1.10.

Tabla 1.10. Pines del ESP8266EX directamente relacionados con módulo ESP8266

Pin	GPIO	Input	Output	Comentarios
D0	GPIO16	No interrupciones	No PWM No I^2C	HIGH durante boot Resistencia Pull-Down Conectar a RST para Wake-Up
D1	GPIO5	OK	OK	SCL (I2C) (frecuentemente)
D2	GPIO4	OK	OK	SDA (I2C) (frecuentemente)
D3	GPIO0	Pull-Up	OK	Boot falla si pulled LOW Conectado a botón FLASH
D4	GPIO2	Pull-Up	OK	HIGH durante boot Boot falla si pulled LOW Built-in LED TX1
D5	GPIO14	OK	OK	SLCK (SPI)

D6	GPIO12	OK	OK	MISO (SPI)
D7	GPIO13	OK	OK	MOSI (SPI)
D8	GPIO15	Pulled to GND	OK	CS (SPI) LOW durante boot Boot falla si pulled HIGH No tiene Pull-Up
RX	GPIO3	OK	RX	HIGH durante boot o usable si se usa UART
TX	GPIO1	TX	OK	HIGH durante boot Boot falla si pulled LOW Debug output en boot No usable si se usa UART
A0	ADC0	Analog Input	NO	
-	GPIO6-11	NO	NO	Usados por la memoria FLASH

1.12. COMANDOS AT

Se debe tomar en cuenta que en sus inicios el módulo ESP8266 al tener un microcontrolador de tipo SoC, era muy obvio que debería tener su propio lenguaje de comunicación, por lo tanto, la empresa Al-thinker usa un lenguaje básico de comandos AT o conjunto de comandos Hayes. Este es un lenguaje desarrollado por la compañía Hayes Communications que se era un estándar abierto de comandos para configurar y parametrizar módems.

Se los denomina “AT”, pues se derivan de los llamados de “Atención” que preceden a dichos los comandos y es su forma de iniciar los ordenamientos de órdenes a los dispositivos. Los comandos AT básicos[39] se pueden hallar en varios links, pero no es destino de este trabajo estudiarlos, por lo que se presentan los más usados en las comunicaciones, en el Anexo A.

Como el ESP8266 tenía definido su propio firmware[40], inicialmente basado en comandos AT, su comunicación era como un simple modem, más al pasar unos pocos años se ha destacado el gran potencial de este módulo. Desde su aparición se lo usaba como un

[39] Los comandos AT se exponen en el Anexo A.

[40] El firmware es el software o sistema operativo de los sistemas con SoC.

actuador de los módulos Arduino, de la mano de su propio esquema de comunicación AT. Pero al pasar del tiempo y cabe destacar los aportes de la comunidad Maker [30] [41], el módulo ESP8266 se ha acoplado de forma extraordinaria al IDE de Arduino por medio de la creación de un firmware adaptado a esta plataforma.

Tomando en cuenta la principal característica del módulo ESP8266, que es su microcontrolador ESP8266EX, la comunidad Maker, ha diseñado varios firmwares que le permiten trabajar en el lenguaje de Arduino. Siendo recíprocos en el aspecto técnico, la empresa Espressif Systems, ha efectuado mejoras en el ESP8266 o ESP01, realizando mejoras en el módulo, dotándolo de una memoria flash y velocidades de comunicación, más amplias. Eso ha permitido que se pueda seguir realizando mejoras en el firmware que se basa en la plataforma Arduino.

Las mejoras en el módulo ESP8266 y los nuevos firmwares desarrollados, han desembocado en que el dispositivo se convierta en un módulo de gestión Wi-Fi autónomo, sin la necesidad de usar una placa Arduino. Por esa razón, ha servido de base para la creación de varios módulos mejorados como, el NodeMCU, WeMOS, WebOS, entre otros, que usan las características de Arduino, especialmente la de ser de hardware y software libre, combinados con la capacidad del microcontrolador ESP8266EX. Lo que ha permitido que se usen en varias aplicaciones especialmente en la domótica y/o en el IoT[42].

Puesto que en la actualidad hay una alta demanda del crecimiento de las plataformas de tecnológicas de domótica y su control por medio de software sobre redes de datos, las IoT, es una industria que se encuentra en auge. Por esa razón se establece como parte fundamental de este trabajo, diseñar una aplicación de hardware usando el ESP8266 y su interfaz de usuario, basado en plataformas software libre como el sistema operativo Android. Puesto que ese sistema operativo, es el más común y accesible se opta por diseñar una aplicación residente de gestión y control para el usuario en esa plataforma.

1.13. SISTEMA ANDROID

El sistema operativo Android está basado en LINUX, por lo que es de código abierto, diseñado exclusivamente para dispositivos móviles. Según los acuerdos de Google, que

[41] La comunidad de programadores y desarrolladores de Arduino.
[42] IoT es el acrónimo de Internet of Things o Internet de las cosas.

es su principal accionista y los fabricantes, la licencia de manejo de Android se basa en Apache [31].

Entonces al tratarse de un sistema operativo de código abierto, permite la accesibilidad y desarrollo de nuevas aplicaciones de los usuarios según la necesidad de los mismos. Por esa razón, existen varias plataformas de desarrollo de APPs[43] para Android, las que siempre demandan del usuario ciertos conocimientos de programación basados en lenguaje BASIC, como C o C++.

Así es que para poder realizar una aplicación de software, los diseñadores pueden optar por el uso de comandos hasta el uso de objetos prediseñados. Por eso es importante que el programador de aplicaciones, sepa qué tipo de programación desea usar. Siendo esa una actividad delimitante, se expone a continuación ciertos tipos de programación, lo que permite justificar cual es el tipo de programación utilizada para desarrollar la aplicación residente del presente trabajo.

1.13.1. PROGRAMACIÓN ESTRUCTURADA

Ese tipo de programación PE[44] se compone de un conjunto de técnicas y arreglos que realiza el desarrollador con el fin de aumentar la productividad del programa reduciendo el tiempo de depuración y mantenimiento del mismo, pues se basa solo en el uso de comandos. Puesto que se basan en el desarrollo del programa desde el nivel cero hasta su totalidad, lo que demanda destreza del programador [32].

El programador crea un número limitado de estructuras de gestión y control, con el objetivo de minimizar considerablemente los errores.

La técnica se centra en los siguientes aspectos:

- Diseño descendente (top-down): los problemas, se descomponen en estructuras jerárquicas, que van desde funciones pequeñas a grandes, de forma escalonada para formar el conjunto final.
- Recursos abstractos (simplicidad): El objetivo de este tipo de programación es reducir la complejidad de los problemas realizando funciones o estructuras que

[43] APP acrónimo de aplicaciones
[44] PE acrónimo de Programación Estructurada

realicen actividades más simples para ser resueltas con mayor facilidad. Por ello a continuación se exponen los tipos de estructuras.

- Estructuras secuénciales: en estas cada actividad sigue a otra secuencialmente, haciendo que la salida de una acción sea la entrada de otra.
- Estructuras selectivas: en estas, se evalúan las condiciones de las acciones y en función del resultado de las mismas se continúa con unas u otras actividades, lo que se podría definir como orden lógico.
- Estructuras repetitivas: son secuencias de instrucciones que se repiten un número determinado de veces, según las condiciones de las actividades a realizarse.

1.13.2. PROGRAMACIÓN MODULAR

En la programación modular como lo indican su nombre consta de varias secciones dividas en forma de procesos o módulos, que pueden ser recurrentes o en saltos.

Este tipo programación, consta de un programa principal el que coordina las llamadas a los módulos secundarios y pasa los datos necesarios en forma de parámetros, los que permiten obtener los resultados esperados según el diseño determinado.

1.13.3. PROGRAMACIÓN ORIENTADA A OBJETOS

Se la denomina orientada a objetos, porque se presentan en módulos gráficos previamente programados. Entonces, el objeto, es un conjunto complejo de funciones y estructuras específicas que se unifican para proporcionar el módulo gráfico, que puede ser de uso variado dentro del programa en desarrollo. En la programación, los objetos están diseñados, para que el usuario o programador los use enfocándose en cambiar ciertas condiciones como propiedades, funciones y acciones permitidas para un resultado general [32].

Como es sobre el objeto la programación, se han desarrollado objetos o esquemas grandes denominados, contenedores padre y otros contenedores internos denominados hijos, esa forma de enlazar actividades permite que los datos y recursos mantengan lo que denomina herencia, lo que permite compartir y optimizar los datos.

Normalmente es usada para realizar aplicaciones con presentaciones visuales más atractivas al usuario. Ayuda a reducir el tiempo de programación, pues el desarrollador no se enfoca en crear los objetos gráficos, sino solo a programar ciertos rasgos necesarios del mismo, por lo que permite obtener resultados a corto plazo. El desarrollador requiere

tener un conocimiento medio en programación y puede lograr bastante atractivo en el resultado final.

De los tipos de programación que se ha expuesto, se ha optado por usar la programación POO[45], porque permite usar objetos ya diseñados para programarlos. Lo que se desea es presentar al usuario final algo atractivo y amigable, además, de fácil de usar. Puesto que, como se ha manifestado antes, se desea crear una aplicación residente para Android, lo que demanda que se use un programa para desarrollo, basado en ese sistema operativo y sea de tipo POO.

Existen varias plataformas de programación de aplicaciones para Android, bajo los esquemas de programación antes mencionados. Algunas son Android Studio, AppMachine, Appcelerator, App Inventor, Scratch. Estos dos últimos son de POO, así que se decide utilizar App Inventor.

1.14. APP INVENTOR

Como parte del desarrollo del presente proyecto, se desea crear es una aplicación residente de fácil utilización para el usuario. Una aplicación que le permita al usuario, activar y desactivar, encender o apagar (ON-OFF) un interruptor. Debe ser lo más atractivo y familiar al usuario, por lo que es necesario presentarle algo estéticamente similar al interruptor, pero de forma virtual, por eso como se ha mencionado antes se ha optado por usar APP Inventor que permite usar la POO.

La plataforma APP Inventor, es un entorno de desarrollo de software creado por Google Labs, que permite la elaboración de aplicaciones para Android usando objetos. Algunas de sus características más importantes son:

- Es de software libre, por ello no es necesario la adquisición de licencias.
- Es multiplataforma: sólo se requiere un navegador web y la máquina virtual de Java instalada, con “java web start”.
- Es de programación POO para dispositivos móviles.

Tomando en cuenta eso se pasa a realizar un análisis de la plataforma para el desarrollo del presente trabajo.

1.14.1. INGRESO

[45] POO acrónimo de Programación Orientada a Objetos

Por ser una plataforma de software libre, cualquier usuario puede acceder a la misma, solo se requiere registrar una cuenta en la plataforma, por medio del correo electrónico, preferiblemente Gmail.

Luego de eso, en el buscador se digita, APP Inventor. Aparece el acceso, se ingresa y pide el correo establecido por el usuario. Luego de ingresado el correo, de forma inmediata se envía a la pantalla de Gestión de Proyectos, desde donde se puede iniciar la programación de un nuevo proyecto o trabajar sobre alguno que ya se ha realizado anteriormente, tal como se puede observar en la Figura 1.9.

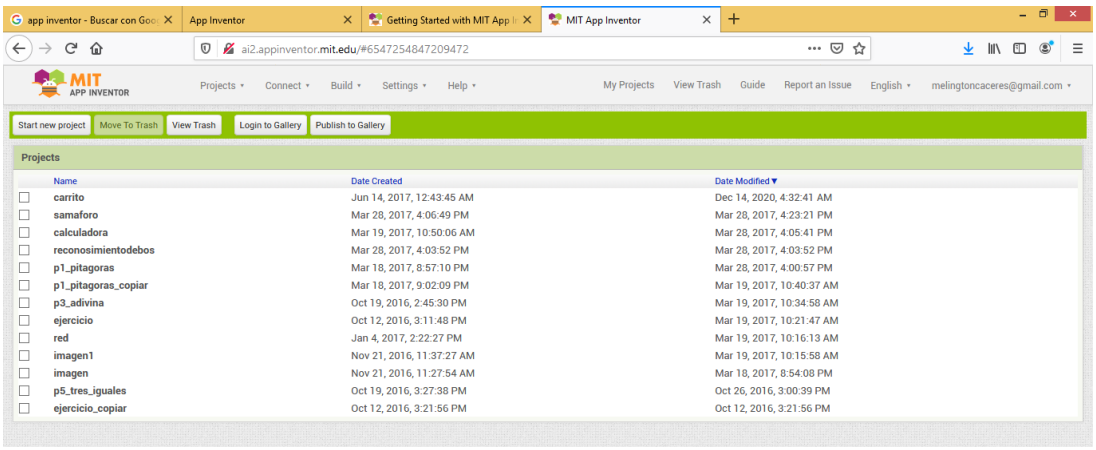

Figura 1.9. Área de Gestión de Proyectos

Después de iniciar con un Nuevo Proyecto, el usuario pasa a la ventana de trabajo.

1.14.2. ÁREA DE DISEÑO

En esta ventana se encuentran dos sub áreas la de Diseño y la de Programación. En la ventana de Diseño, se divide en cuatro áreas, las que se nombran desde la izquierda a derecha, según de aprecia en la Figura 1.10, de izquierda a derecha.

- Área de objetos.
- Área del proyecto o visor de pantalla del dispositivo.
- Área de componentes de los objetos.
- Área de propiedades de componentes.

Como se trata de una plataforma de POO, esas áreas descritas, permiten al desarrollador seleccionar características visuales preestablecidas, que desea que tenga el objeto, para ser presentado al usuario.

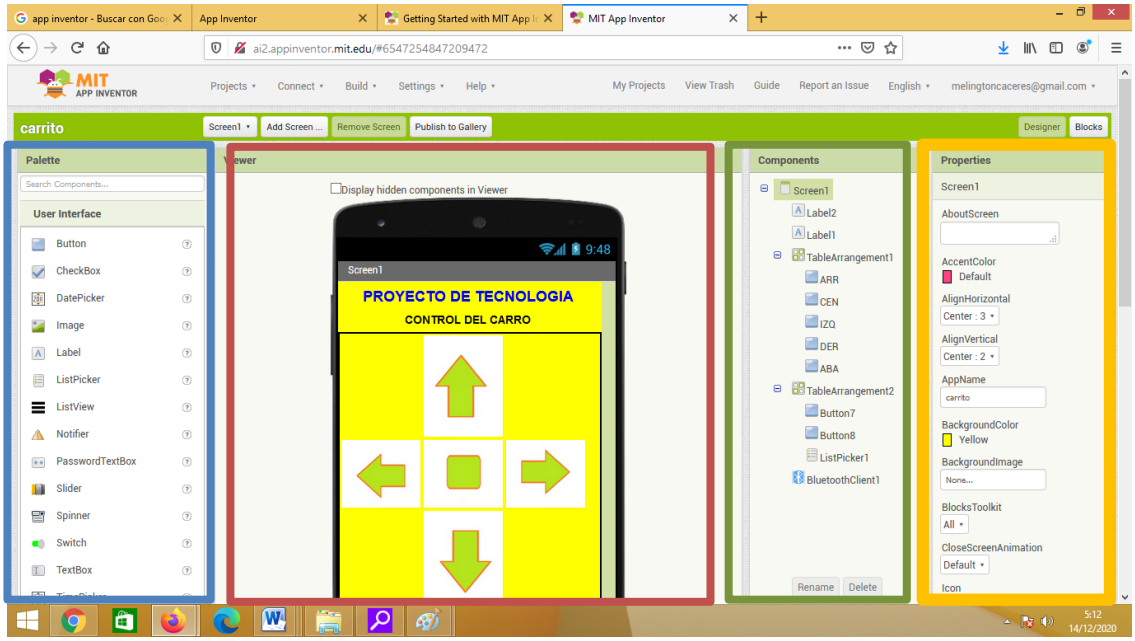

Figura 1.10. Ventana de Diseño

1.14.3. ÁREA DE PROGRAMACIÓN

Desde la ventana anterior se puede acceder al área de programación, presionando el botón *Blocks* (bloques) en la parte superior derecha. Al hacerlo se despliega la ventana que contiene los componentes seleccionados por el desarrollador para programar las acciones que ejecutaran los objetos, seleccionando las funciones que cada uno debe tener según los planes del programador.

Es importante mencionar que la ventana se divide en dos partes. En el sector de la izquierda, donde se presentan los objetos con sus respectivas funciones, las que el usuario puede seleccionar según su necesidad.

El área de la derecha, contiene en forma de bloques cada uno de los objetos y las acciones de las funciones, que se pueden programar en el mismo para los fines pertinentes según la aplicación.

Las áreas mencionadas se pueden observar en la Figura 1.11 que se presenta a continuación.

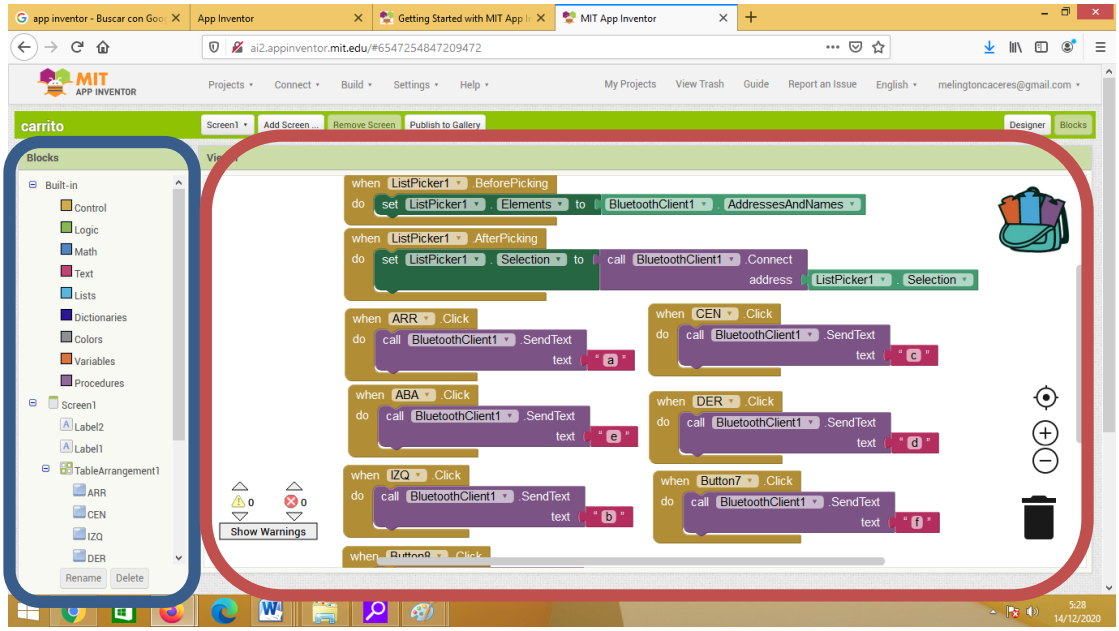

Figura 1.11. Área de Programación

Tomando en cuenta los parámetros mencionados de la programación, se puede realizar cualquier aplicación que el usuario requiera según su necesidad. Usando toda la información presentada como fundamento teórico para el presente trabajo, se pasa a diseñar el objetivo principal del mismo. Diseñar un módulo Wi-Fi usando las características combinadas del ESP8266 y el IDE de Arduino, para la gestión ON – OFF, de dispositivos o instrumentos con esa función pero de forma inalámbrica, enfocándose en el esquema denominado actualmente domótica y IoT, manejado de forma amigable por medio de una aplicación residente sobre el sistema Android.

2. METODOLOGÍA

El presente capítulo se enfoca en el diseño teórico del dispositivo de control ON-OFF gestionado y controlado por el módulo ESP8266 por medio de una red Wlan y su interfaz residente en Android.

Se debe enfatizar que actualmente la mayor parte de dispositivos desarrollados para el hogar, la industria hotelera, administración de edificios, arquitectura y otras áreas relacionadas con ellas, se enfocan en la gestión de automatización de ambientes por medio de dispositivos eléctricos o electrónicos, con el fin de brindar confort, minimizar el trabajo manual, ahorro de energía y recursos. A ese campo se le denomina domótica (hogar) e inmótica (edificios).

Además, se debe resaltar que, con el amplio desarrollo del software, de las redes de comunicaciones, el mejoramiento de velocidades de comunicación de las redes de datos, se ha desarrollado una gran industria denominada como IoT (Internet of Things o Internet de las cosas).

Por lo tanto, el presente trabajo se acopla en dichas áreas, la domótica y la IoT. Así es que se pasa a definir el proceso metodológico que se realiza para lograr el objetivo.

2.1. DESARROLLO DEL PROYECTO

Tomando en cuenta lo expuesto y resaltando el objetivo de este trabajo se procede al diseño del módulo ON – OFF, que permita el control de aparatos eléctricos o electrónicos (domótica), por medio un dispositivo de gestión a través de red Wlan, aprovechando las ventajas que brindan las plataformas de hardware y software libre como el ESP8266, el IDE Arduino y Android.

Para realizar el diseño es importante definir como se procederá para lograr el fin del trabajo, enfocándose en ciertos aspectos técnicos sobre los componentes directos del diseño del prototipo.

2.2. ESQUEMATIZACIÓN DEL PROYECTO

Para iniciar el diseño del trabajo se procede a determinar un esquema de los procesos o pasos a seguir para desarrollar el prototipo propuesto. Por eso de forma general se establece como principales los siguientes procesos, que se observan en la Figura 2.1.

Figura 2.1. Esquema de procesos para el proyecto

Cada uno de estos procesos se realiza según un orden establecido para evitar pérdidas de tiempo y recursos. Entonces a continuación se pasa al desarrollo de los cuatro primeros y cabe mencionar que el último es la parte principal del siguiente capítulo.

2.3. DISPOSITIVO DE CONTROL Y GESTIÓN

El primer paso para efectuar el diseño del presente trabajo, es centrarse en dos partes importantes, determinar el tipo de dispositivo ON – OFF que se desea usar y en segundo lugar, el dispositivo de gestión y control. Así que se pasa a estudiar las razones por las que se determinan los dos dispositivos.

2.3.1. DISPOSITIVO ON – OFF

Se define como dispositivo ON – OFF, a los dispositivos manuales o automáticos que cumplen con solo dos funciones específicas, Encendido (ON) y Apagado (OFF). Muchos de estos dispositivos se usan en varios campos del quehacer diario de las personas, como:

- Iluminación: Luminarias tanto de uso general como de confort y /o seguridad.

- Sonido: activación y desactivación de alarmas, sonidificación de ambientes.
- Gestión Energética: sistemas de calefacción o enfriamiento, sistemas de riego.
- Control: activación o desactivación de sistemas domóticos, inmóticos o industriales en general.

Es necesario mencionar que en muchas ocasiones los dispositivos ON – OFF, pueden constituirse en el mando final de varios sistemas de control, sean estos manuales o domóticos, por cuanto son muy útiles en muchos aspectos de la vida cotidiana y profesional del ser humano. Como, por ejemplo, la activación y desactivación general de los sistemas eléctricos y electrónicos, los que pueden ser de tipo manual o virtual.

Existen varios dispositivos ON – OFF, manuales como interruptores y automáticos como semiconductores (electrónicos) y relés (magnéticos). Cada uno de esos dispositivos se los usa según la necesidad el usuario. Y puesto que se desea realizar la gestión y control de forma inalámbrica por medio de una red Wlan, se cree que es más conveniente usar un actuador de tipo magnético o relé. Por supuesto, es importante destacar que se opta por un relé Arduino ya que son más estables y requieren menos circuitería pues ya está integrada en un solo módulo.

Con ese actuador se realiza el desarrollo del prototipo propuesto, el mismo que servirá de base para posteriores implementaciones de cualquier dispositivo o aparato eléctrico o electrónico que requiera la gestión y control de tipo ON-OFF.

2.3.2. DISPOSITIVO DE GESTIÓN

Después de realizada la determinación del dispositivo ON-OFF, se debe definir los dispositivos electrónicos o domóticos que permita la programación, monitoreo, gestión y control de otros dispositivos, sistemas o bloques de sistemas por medio de una red Wlan. Como ya se ha mencionado antes, se ha optado por usar el módulo ESP8266.

Normalmente existen varios dispositivos ya diseñados e implementados por varias empresas dedicadas a la domótica, que se encargan de ese tipo de gestión y control, pero en su gran mayoría no son de hardware y software libre lo que crea una dependencia de las empresas productoras.

Por esa razón, se ha optado por usar dispositivos y plataformas de hardware y software libre. Así, este trabajo se centra en utilizar el módulo ESP8266, del que ya se ha estudiado

lo suficiente para desarrollar una nueva aplicación del hardware y software libre, por medio de la red Wlan.

Entonces, para resumir lo referente al primer proceso del diseño del dispositivo de control ON-OFF, por medio del módulo ESP8266 con interfaz residente sobre una red Wlan. Se ha identificado que el dispositivo de control es un relé Arduino y el gestión es por medio del ESP8266.

2.3.3. ACTUADOR ARDUINO

El actuador de Arduino, como se ha explicado a breves rasgos antes, se trata de un relé controlado por 5V. Como se puede observar en la Figura 2.2, el relé puede trabajar a 250 VAC y 128VAC, a corrientes de máximo 10A. También, se puede observar por el lado izquierdo, tres conectores para voltaje alterno. Y por el derecho tres conexiones de voltaje directo.

Los conectores de voltaje alterno son:

- Dos para la línea de energía
- Uno NC[46].

Los conectores de voltaje directo son:

- VCC y GND.
- IN1, puerto de control.

Figura 2.2. Módulo relé Arduino 5V

[46] NC No Conectar, por razones establecidas por el fabricante.

2.4. GESTIÓN DE ENERGÍA

El proceso se enfoca en el desarrollo de la fuente de energía para la alimentación energética del dispositivo que se desea desarrollar. Es muy importante realizar este paso pues los dispositivos que intervienen, tanto el ESP8266 como el relé de Arduino, trabajan con voltajes y corrientes diferentes, por lo que se pasa a estudiar la forma de desarrollar la fuente de energización del circuito final.

2.4.1. FUENTE DE VOLTAJE DC

Un aspecto importante para el diseño del prototipo, que es el objetivo de este trabajo, es determinar la alimentación energética. Por esa razón, se debe considerar que el módulo ESP8266, está diseñado para trabajar con un voltaje de 3.3V, por cuestiones de estructura física. Por otro lado, el dispositivo ON-OFF o actuador de Arduino (un relé) trabaja con una fuente de 5V.

Entonces tomando en consideración esos aspectos del suministro energético, se debe determinar una forma de acoplamiento o derivación de energía que permita el funcionamiento de los dos dispositivos usando una misma fuente. Pues si se decide por usar fuentes independientes para cada caso, sería invertir recursos innecesarios, además de ocupación de espacio, ya que se desea que el dispositivo sea compacto y casi imperceptible.

Para lograr conseguir el tipo de fuente que cumpla con los voltajes antes mencionados, antes se debe investigar otros aspectos importantes del diseño, como las intensidades de trabajo de cada circuito y las impedancias. Así se puede tomar una decisión basada en enfoques técnicos al momento del diseño, por si se requiere fuentes acopladas o una fuente con derivaciones. Por ello, es imprescindible estudiar los comportamientos energéticos del ESP8266 y el actuador de Arduino.

Tomando en cuenta lo explicado en el capítulo I sobre las principales características del módulo ESP8266, entre ellas las energéticas, se pasa a destacar las más importantes a continuación:

- Voltaje de alimentación: 3.3VDC
- Voltaje de programación(GPIO): 3.3VDC – 5VDC
- Corriente de consumo: 80mA, sin realizar ningún trabajo.

- Corriente máxima (TX y/o RX): 56mA a 250mA, es recomendable tener una fuente de 300mA[47].
- Potencia de consumo máximo: 185mW – 1 W.

Así, se determina que la fuente para el ESP8266 debe ser de 3.3V y 300mA, por el caso de consumos adicionales propios de la fuente.

Ahora se procede a determinar las características energéticas del dispositivo de control o relé Arduino.

- Voltaje de alimentación: 5VDC
- Corriente de consumo: 90mA, por la bobina
- Voltaje de acción: 125VAC a 10A y 250VAC a 10A.

Entonces de estas dos observaciones, se deriva que:

- Por el voltaje del Relé Arduino (Vr)[48], se debe tener una fuente de 5V, pero su corriente (Ir)[49], consumo de la bobina, como mínimo debe ser de 90mA.
- El consumo de corriente (Ie)[50] del ESP8266 es aproximadamente 300mA y la fuente de voltaje (Ve)[51] debe ser de 3.3V.

Por lo tanto, la corriente de consumo de los dos circuitos, es la suma de la corriente del relé y la corriente del módulo de ESP8266, tal como se puede verificar en la ecuación 2.1:

$$It = Ir + Ie \quad (2.1)$$

$$It = 90mA \text{ (bobina relé)} + 300mA \text{ (ESP8266)} \quad (2.2)$$

$$It = 390mA$$

Por lo tanto, según el resultado de la ecuación 2.2, la corriente de consumo del circuito es de 390mA, aproximadamente 400mA. Siendo ese el resultado, se puede optar por usar una fuente de 5V y 500mA.

Pero cabe destacar que la fuente no es independiente sino una fuente derivada, que alimente con 5VDC al relé y de 3.3V para el ESP8266. Entonces se hace hincapié en los

[47] Según la hoja de datos del ESP8266
[48] Vr abreviación de voltaje de relé
[49] Ir abreviación de corriente de relé
[50] Ie abreviación de corriente de ESP8266
[51] Ve abreviación de voltaje de ESP8266

conocimientos de electrónica básica, donde se sabe que, una fuente de voltaje de dos ramas en paralelo, permite mantener el mismo voltaje en cada rama.

Mientras que en el caso de la corriente, se divide proporcionalmente en cada rama según su carga. Pues en el caso de la intensidad, se conoce, que en un circuito en paralelo se crea un divisor de corriente. Por lo tanto, el circuito que engloba este tipo de fuente es un paralelo como se observa en la gráfica de la Figura 2.3.

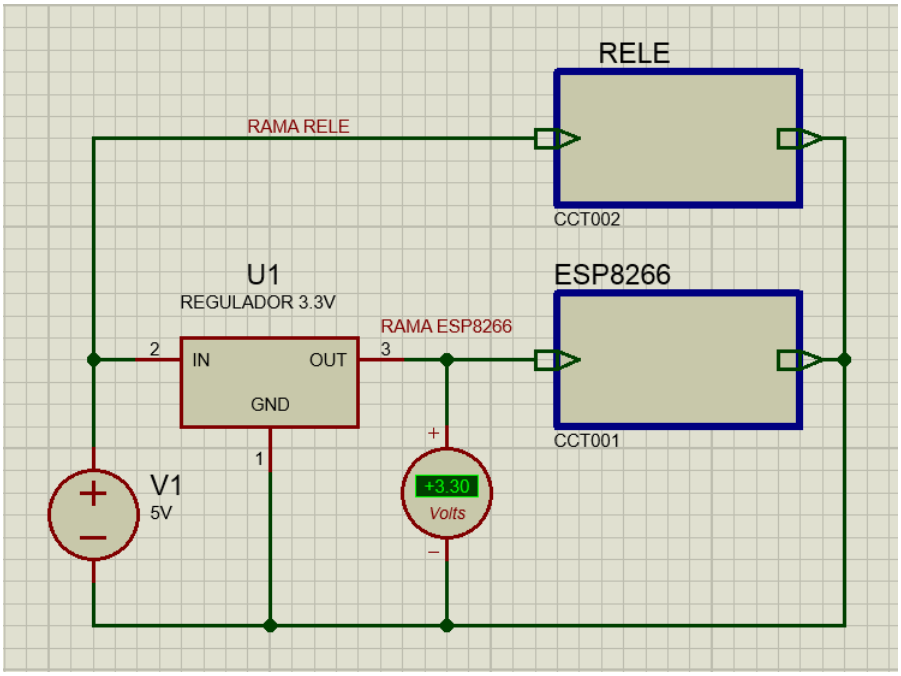

Figura 2.3. Circuito de fuente de alimentación

Donde la fuente es de 5V con una corriente de 500mA. A esa fuente se realiza una división eléctrica en paralelo en forma de dos ramales. Por un lado, por la rama donde se encuentra el relé, se mantiene el voltaje de 5V, pero con una corriente de 90mA.
Por la otra rama, donde se conecta el ESP8266, puede consumir la intensidad de 300mA, a un el voltaje de 3.3V, lo que requiere que se haga un arreglo electrónico para poder obtener ese valor de voltaje. Ese arreglo electrónico es usar un regulador de voltaje de 5V a 3.3V, el que es el circuito integrado, el LM1117.

El circuito integrado LM1117-3.3 [33] que es un regulador de voltaje lineal, de 3.3V. El voltaje de entrada de este circuito es de 5VDC, su voltaje de operación y consumo es 1.7VDC, por lo que su salida es de 3.3V. También permite el paso de una corriente de 800mA, hacia su carga.

Algo imprescindible en el arreglo circuital, es el acoplamiento del circuito LM1117-3.3, ya que pueden ocurrir picos de energía y saltos magnéticos por acumulación de corrientes estáticas, por lo que el fabricante prescribe que para el uso y estabilidad de carga del regulador, se requiere que se realice un acoplamiento capacitivo como se muestra en la Figura 2.4.

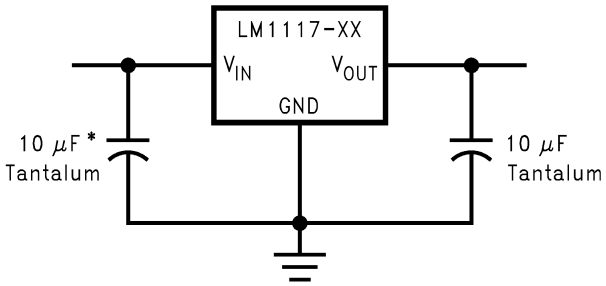

Figura 2.4. Acoplamiento capacitivo del regulador [33]

2.4.2. ADAPTACIÓN DE AC – DC

Otro aspecto fundamental del diseño del proyecto, es la fuente general de energización del dispositivo de control ON-OFF. No puede ser una fuente independiente, sino que debe aprovechar la misma fuente de energía de la red eléctrica AC, a la que el dispositivo por controlar este conectada. Se debe considerar que los dispositivos de encendido y apagado (ON – OFF), generalmente están conectados a fuentes de 110 VAC o 220 VAC, pero como ya se analizó anteriormente se requiere una fuente de 5 VDC y 3.3VDC.

La única forma de cumplir con ese requerimiento es diseñar la fuente de 120VAC y regularla a 5VDC. Pero como ese no es parte fundamental del desarrollo de este trabajo, se determina que no es necesario diseñar la fuente regulada. Entonces se concluye que es posible usar una fuente regulada prediseñada, ya que es más factible que diseñar una. Entonces se decide utilizar una fuente regulada de 5V a 0,5 A o 1 A, que lo mismo que usar un cargador de teléfono a 5V a 500mA o 1A, que se pueden conseguir fácilmente en el mercado.

2.5. MODIFICACIÓN DEL SOFTWARE DEL ESP8266

El tercer proceso se enfoca en determinar el uso combinado de las plataformas de gestión, para cumplir con el objetivo de este trabajo. Las plataformas que se usan en primera instancia son el firmware de ESP8266 y la IDE de Arduino, para modificar el firmware original del módulo, que servirá de base para poder combinar con plataforma Android.

2.5.1. PLATAFORMAS DE DESARROLLO

Como se ha explicado en el Capítulo I, las plataformas que intervienen en el desarrollo este trabajo, son tres.

La primera plataforma, es el módulo actuador Wi-Fi, ESP8266, que por defecto tiene su propio firmware, basado en comandos Hayes o comandos AT.

La segunda la plataforma, Arduino, con su IDE interviene de dos formas:

- La modificación del Firmware del módulo ESP8266.
- Programación de funciones del módulo ESP8266.

Finalmente la tercera plataforma, Android, que aporta con el software de gestión o la aplicación residente de interfaz de usuario para el control, del dispositivo ON-OFF.

Descritas las tres plataformas que intervienen se procede a describir la manera de interrelacionarse en el siguiente proceso.

2.5.2. MODIFICACIÓN DE FIRMWARE

El software del ESP8266, se basa en la comunicación de comandos Hayes o AT, el mismo que se puede actualizar, cada vez que lo requiera el usuario, como se puede observar en la página de la empresa Espressif System [34]. Pero este tipo de comunicación es muy poco amigable, pues es bastante limitada, además, que requiere de un monitor externo para su monitoreo y gestión. Por lo tanto, Como ya se ha descrito antes, el módulo ESP8266 fue en sus inicios desarrollado para que trabaje como actuador de Arduino. Su función específica era como una especie de antena inteligente que trasmitía y recibía datos generados por la placa Arduino con la que trabajaba.

Por esa razón, para que el módulo ESP8266 trabaje como un dispositivo inteligente pero independiente, es necesario mejorar su accesibilidad y formato de comunicación. Así que, observando las capacidades de hardware que posee el dispositivo, se ve necesario usar las ayudas que presenta Arduino en esos casos, para realizar una modificación del su firmware original a un sistema más amigable y escalable.

Ese tipo de software, fue desarrollado para que el módulo ESP8266, sea independiente y se aproveche el hardware que posee. Los desarrolladores de ese software lo

implementaron como parte del staff de librerías de Arduino. Es un firmware diseñado en C++, que permite que el microcontrolador del ESP8266EX, obtenga las mismas capacidades de las tarjetas Arduino, pero que mantenga sus características y potencial de sus componentes Wi-Fi.

Algunas de las características del nuevo software para el ESP8266, son las que se describen a continuación:

- Se puede implementar por medio del mismo IDE de Arduino.
- El firmware se comporta como un Cross Compiler (compilador cruzado), de modo que prácticamente se puede usar los mismos comandos de Arduino, por tanto, es suficiente con poder programar en ese lenguaje.
- Dependiendo del módulo ESP8266 que el usuario use, se dispone de más o menos pines con PWM, I^2C y SPI. Tomando en cuenta que varios módulos usan el chip ESP8266EX, tal como se expuso en el apartado 1.12. Pero para el caso particular de este trabajo se usa el módulo ESP8266-01 o ESP01, que solo tiene dos pines GPIO0 y GPIO2 para la programación.
- En lo referente a WIFI, ya no se programa por comandos AT, pues se incluye varios comandos se incluyen en las librerías, que imitan la librería WIFI de Arduino.

Con esos antecedentes y a fin de cumplir con el objetivo de este trabajo, se procede a trabajar en la modificación de firmware. Para eso se requiere, realizar ciertas actividades que permiten ejecutar el proceso.

El módulo ESP8266 solo posee dos pines GPIO y el hardware que tiene no puede ser programado directamente como sucede con Arduino. Por esa razón, se requiere de algún dispositivo que sirva de interfaz FTDI[52] o convertidor de señales de datos tanto RS232 o TTL a señales USB, que permita la transferencia de datos desde el IDE de Arduino y el módulo ESP8266.

Existen dos formas de hacerlo:

- La primera, es usar directamente un dispositivo FTDI comercial, como el que se observa en la Figura 2.4.

[52] FTDI Future Technology Devices Inhternational, empresa especializada en comunicación USB

Figura 2.4. Dispositivo FTDI [35]

- La segunda, es usar una placa de Arduino, que haga las funciones del FTDI antes mencionado. Pero como el Arduino tiene su microcontrolador, al programar se puede cometer el error de implantar el programa del ESP8266 en el Arduino. Entonces para evitar que el microcontrolador de Arduino interfiera, se procede a quitarlo de la tarjeta, entonces solo queda activo el microcontrolador del ESP8266, donde se puede almacenar el programa.

Para el caso particular de este trabajo, se opta por la segunda opción, pues eso evita correr con gastos innecesarios, ya que el FTDI solo serviría para programar el ESP8266, mientras que la placa Arduino puede servir para otro proyecto.

Tomando en cuenta esa razón, se decide por obtener una placa Arduino a la misma que se le quita su microcontrolador Atmega8. La forma de conectar el ESP8266 y el Arduino, es como se puede observar en la Figura 2.5, de ese modo la placa Arduino sirve como un transceiver o como el FTDI antes mencionado.

Según el gráfico el Arduino proporciona la energía al ESP8266, o sea, los 3.3V. Pero como se analizó antes la conexión por USB de la placa Arduino tiene una corriente cercana a 500mA[53], valor que tendría que distribuirse entre el consumo del ESP8266 [54] y el consumo de la placa Arduino[55].

[53] Se analizó en el apartado 1.3.2.1.
[54] Aproximadamente de 56mA según el apartado 1.3.4.16
[55] El consumo de la placa es de aproximadamente 50mA según se analizó en el apartado 1.3.2.1

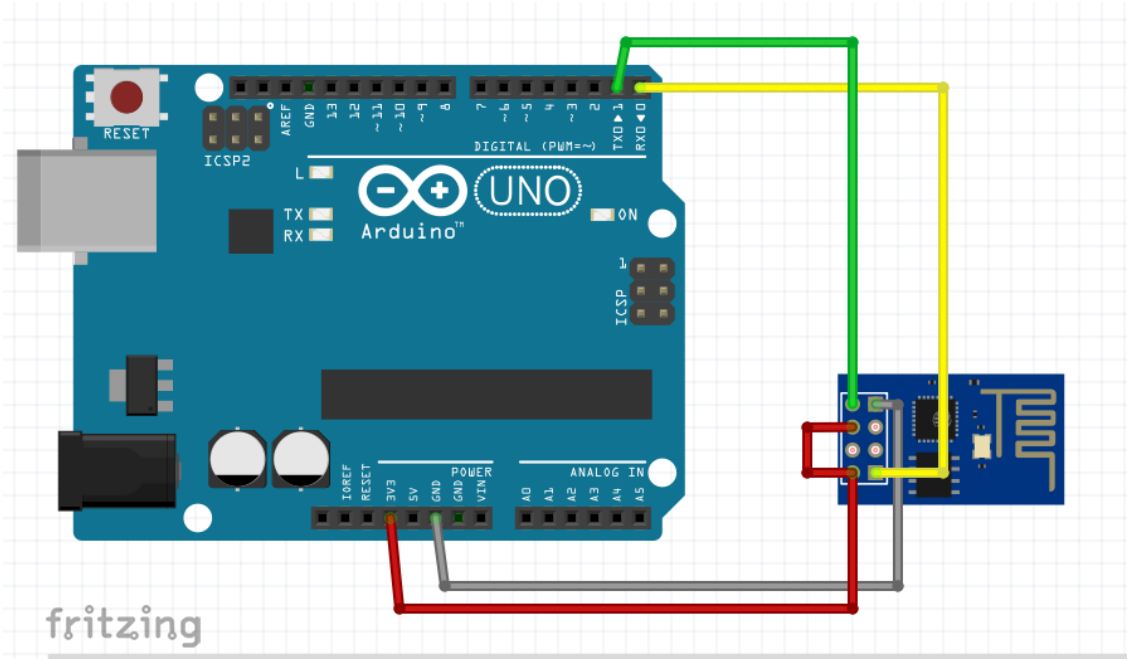

Figura 2.5. Conexiones el Arduino y ESP8266

Pero algo que se debe considerar es que cada salida de energía del Arduino UNO solo puede entregar una corriente de 40mA[56], por esa razón no es aconsejable usar Arduino por mucho tiempo como fuente de energía al Arduino UNO. Pues si la ESP8266, empieza a transmitir su consumo aumenta drásticamente a 300mA y es importante recordar que la placa Arduino servirá solo para transferir los datos del IDE hacia el ESP8266. Entonces solo se puede conectar de la forma como se exponen en la Figura 2.5, solo hasta modificar el firmware.

Después de modificado el firmware el módulo ESP8266, se energizará por medio del acople de la fuente de 5V, con el regulador LM1117 de 3.3V, tal como se vio en la Figura 2.2. en definitiva como se muestra en el esquema de conformación electrónica se puede observar en la Figura 2.6.

[56] Este aspecto técnico se relató en el apartado 1.3.2.5

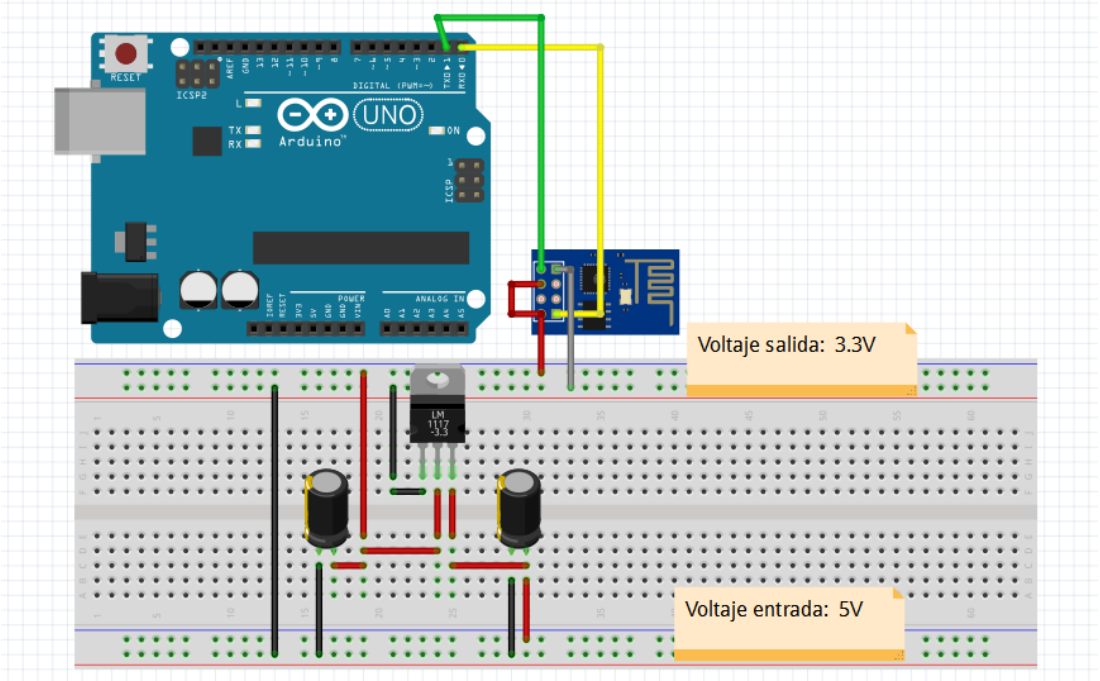

Figura 2.6. Esquema de conformación electrónica del módulo

Luego de efectuado ese proceso de acoplamiento eléctrico y de datos del ESP8266 se pasa a realizar la propuesta de software que reemplazara al firmware que tiene por defecto el módulo Wi-Fi.

2.5.3. PROGRAMACIÓN DEL MÓDULO CON ARDUINO

En este paso se procede a cargar las librerías que contienen el nuevo firmware del módulo ESP8266 en la plataforma Arduino. Para realizar este proceso se realiza ciertas actividades primordiales que se citan a continuación para guiar la forma de realizar la modificación de firmware del ESP8266.

- Primero se debe instalar el IDE de Arduino en el PC [36][57].
- Abrir el programa, de forma automática aparece un *sketch* nuevo como se muestra en la Figura 2.7.

[57] Ese software se puede bajar de forma gratuita o haciendo un donativo desde la página oficial de Arduino.

Figura 2.7. Sketch de Arduino

- En esta ventana en la parte superior aparece el menú *Herramientas*. Se abre el Menú y se escoge la opción *Administrar Bibliotecas*. Tal como se puede divisar en la Figura 2.8.

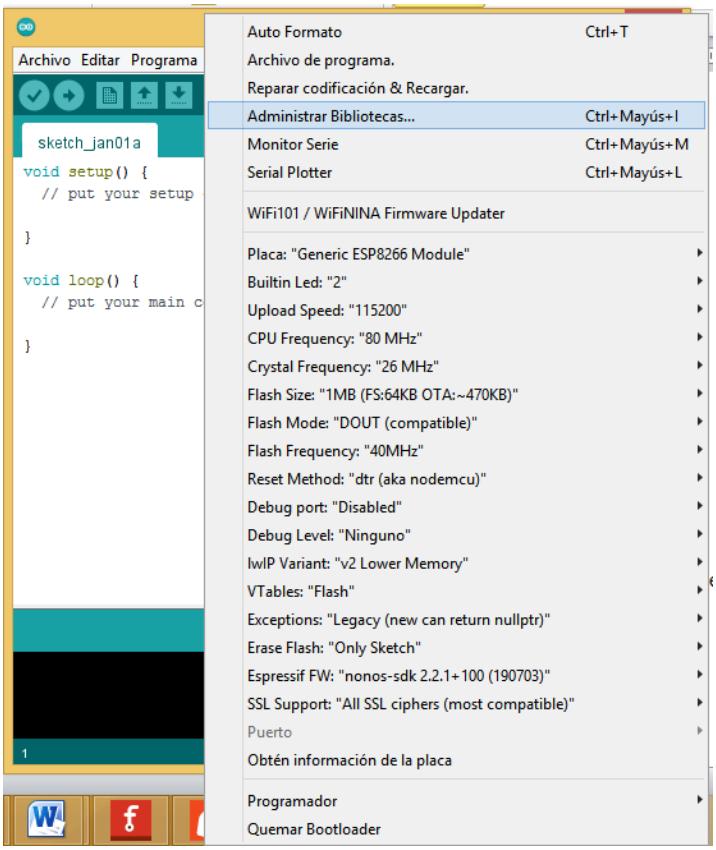

Figura 2.8. Menú Herramientas y opción Administrar Bibliotecas

- Al seleccionar esa opción aparece una ventana de búsqueda y selección, en la que se tapea la librería que se desea añadir al stack de Arduino. Para nuestro caso ESP8266, tal como se mira en la Figura 2.9.

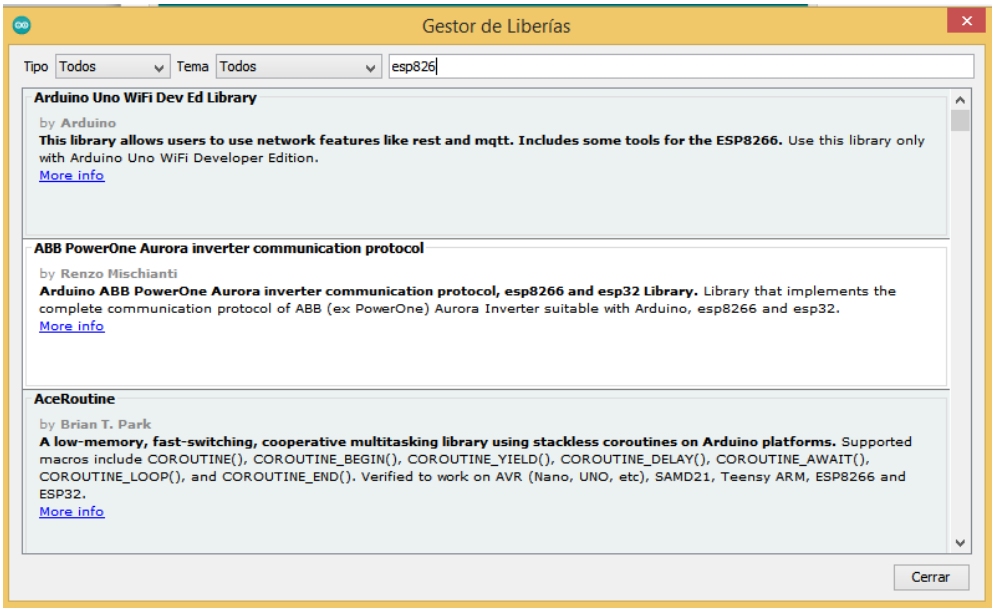

Figura 2.9. Ventana de búsqueda de librerías

- Después de tipear ESP8266 en el espacio de búsqueda, inmediatamente aparecen en la ventana todas las librerías relacionadas con el ESP8266. Muchas de esas librerías son complementos de otras que permiten a Arduino trabajar de forma directa o indirecta con el módulo ESP2266. De esas, las librerías más importantes son las que se distinguen en la Figura 2.10. En cada librería aparece un botón de Instalar o Actualizar, se presiona sobre ellos y se instala o actualiza la librería.

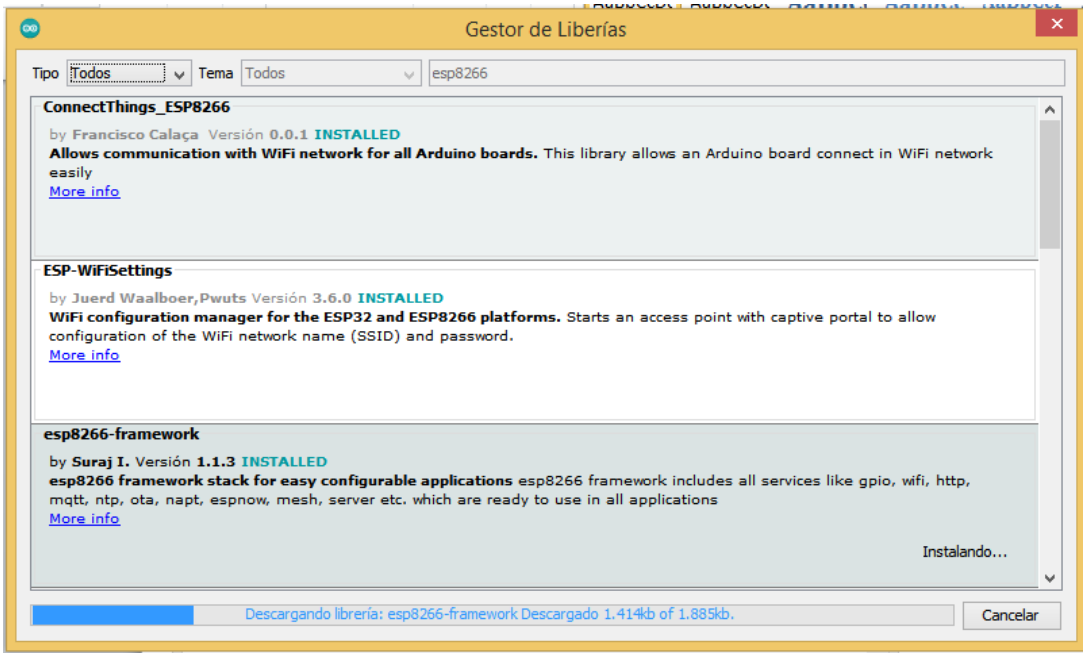

Figura 2.10. Librerías del ESP8266

- Para verificar si ya está instalado el software del módulo ESP8266, se procede a revisar en el Menú *Herramientas,* la opción *Placa.* Luego a la opción *Gestor de Tarjetas*, se presiona e inmediatamente aparece la ventana de tarjetas con sus librerías incluidas. En caso de no estar completa se presiona instalar o actualizar y el módulo ESP8266, queda completamente instalado. Eso se puede distinguir en la Figura 2.11.

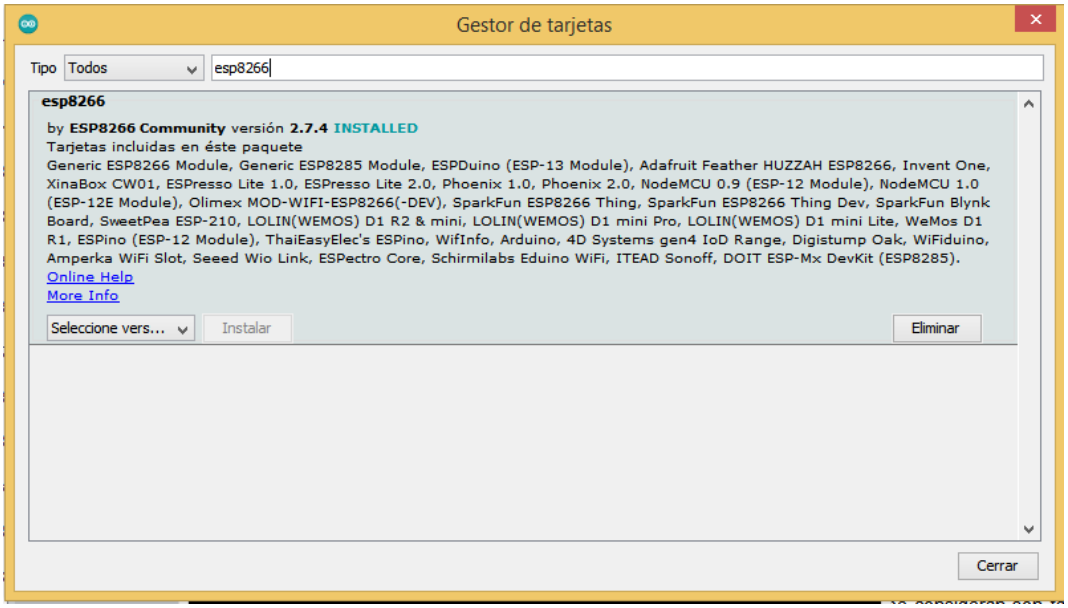

Figura 2.11. Gestor de Tarjetas

- Finalmente, módulo ESP8266 ya está incluido en el IDE de Arduino como si fuera una tarjeta similar. Como se puede distinguir en la Figura 2.11. escogiendo *Herramientas*, *Placa*, *ESP8266 Boards* y ubicando cualquiera de las placas que funcionan con el chip ESP8266EX. Para el caso específico de este trabajo se escoge la placa denominada *Generic ESP8266 Module.*

Después de haber incluido el software del ESP8266 en las Librerías de Arduino. El IDE permite usar el módulo ESP8266, tal cual fuera una placa de Arduino. Pero todavía no ha cambiado el firmware del ESP8266, para lograr eso se requiere empezar a trabajar sobre la placa ESP8266, y sobre el *sketch* en la programación incluir las librerías del ESP8266.

2.5.4. CONSIDERACIONES DE LA PROGRAMACIÓN

Antes de iniciar el proceso de programación del módulo ESP8266. Al tratarse de un dispositivo Wi-Fi, lo primero que se debe considerar, son ciertos factores técnicos específicos de su trabajo, que debe estar presentes en la programación, como:

2.5.5. MODO DE EJECUCIÓN

Puede ser AP (Access Point) o Estación. En el caso de AP, el módulo ESP8266 puede ser el dispositivo central de la red. Mientras que en modo Estación, debe conectarse por medio de un AP central.

Para el caso específico de este trabajo, se define que funcione como Estación, dentro de una red Wlan, por lo que depende de un AP central.

2.5.6. RED

Identificar o programar la red sobre la que se ejecutará su trabajo. Tomando en cuenta el modo de configuración, si es como AP entonces se debe determinar algunas características importantes como, el pool de la red, el SSID, Seguridad, entre otras.

Por otro lado, si se trata del modo Estación, debe ser capaz de identificar la red, parámetros de seguridad, establecer y mantener la comunicación interna entre él y el AP.

Como se ha determinado que trabaje como Estación, entonces se debe programar para identificar la red sobre la que trabajará, por medio del SSID y su contraseña de ingreso.

2.5.7. VELOCIDAD DE TRANSMISIÓN

Este factor es fundamental ya que el ESP8266 tiene la capacidad de trabajar a varias velocidades, pero para la transmisión de redes Wlan se sugiere que sea a la velocidad de 115200 baudios[58], aunque es posible trabajar con velocidades menores sin tener problemas de comunicación. Puesto que el firmware usado en Arduino de la herramienta *esptool.py* está diseñada para trabajar a esa velocidad [37].

2.5.8. CONFIGURACIÓN DE ENTRADA/SALIDA

Como se había mencionado en el capítulo anterior, se procese a programar las entradas/salidas de datos. Por tener solo dos GPIO[59] programables mientras el GPIO está por defecto en 0_L, se debe configurar los mismos en modo de recepción y transmisión de datos, por medio de establecer los pines en voltaje bajo (LOW o 0V) o alto (HIGH o 3.3V)[60].

[58] Esa es la velocidad máxima de la UART, se vio en el apartado 1.3.4.22
[59] Se vio en el apartado 1.3.9
[60] Se estudió en el apartado 1.3.9.1

En la programación general se declaran a GPIO0 y GPIO2 en LOW, para definir un punto de inicio, ya que no reciben ni transmiten. Entonces si los GPIO0 y GPIO2 pasan a estado HIGH se activa el modo de comunicación, ya que se activa el modo Boot. Pero si se pasa a estado LOW se desactiva la comunicación. Lo que permite desde ya realizar la gestión de los dispositivos ON – OFF, definidos anteriormente.

Por lo tanto las conexiones físicas para ejecución y programación del ESP8266 son[61]:
Modo ejecución

- GPIO0: 3.3V por defecto (nivel alto)
- GPIO2: 3.3V por defecto (nivel alto)

Modo programación

- GPIO0: 0 V por defecto (nivel bajo)
- GPIO2: 3.3V por defecto (nivel alto)

2.5.9. SERVIDOR WEB

Puesto que el módulo ESP8266, tiene la capacidad de almacenar datos en sus memorias flash. El nuevo firmware diseñado para la IDE de Arduino, le permite al módulo, la implementación de un mini Servidor WEB en su memoria.
Este servidor WEB es básico, pero le permite al ESP8266 brindar mucha aplicabilidad, ya que se puede desempeñar dentro de una red interna (intranet), gestionada por un router central o como una externa (Internet). Todo depende de la programación y necesidad del usuario.

El servidor permite gestionar datos desde la red para ser ejecutados, según la programación del desarrollador. Pero el alcance de la aplicación depende de la red que se use, ya que puede ser una red interna o externa, por lo que se pasa a analizar el tipo de red.

2.5.10. RED INTERNA

Si es dentro de una red interna, el servidor, debe ser capaz de interpretar un DHCP para recibir su IP, con la que puede transmitir o recibir información. O puede el programador definir una IP específica, dentro del rango de IPs del dispositivo central, para así poder

[61] Según lo analizado en la tabla 9.

obtener una mayor velocidad de conexión. Esta última configuración debe ser bien planteada para evitar la duplicación de IPs o la interferencia con DHCP del enrutador de la red.

También se debe programar el SSDI de la red, además de su contraseña, para poder obtener el acceso a la red. Con eso el servidor sobre el ESP8266, puede gestionar y controlar cualquier dispositivo dentro de la red que esté al alcance del router. Tomando en cuenta eso se puede decir que ese sería un inconveniente, por la cobertura limitada por el router.

2.5.11. RED EXTERNA

Si se programa el servidor para que trabaje, en una red externa, ya sea por internet, la programación debe incluir ciertos factores adicionales, como la cantidad de datos, la administración y almacenamiento de los mismos.

Con eso en mente, todo servidor WEB puede almacenar, manipular y administrar una gran cantidad de datos. Tomando en cuenta el caso de los datos, se tiene que programar los espacios de trabajo y funciones de gestión de datos. Se debe determinar si se desea trabajar muchos datos, es necesario crear una base de datos. Actualmente, eso no es muy difícil tratar con muchos datos, ya que existen varias plataformas de gestión de datos de código abierto.

Otro aspecto de un servidor WEB, que trabaje en una red externa, es el hospedaje o hosting. Él hospedaje del servidor puede ser propio o arrendado. Si el servidor tiene hospedaje propio, debe tener una IP exterior o de salida, misma que debe ser contratada a un servidor IP, para que tenga acceso directo a la nube. Eso demanda un costo adicional.

Por otro lado, si se toma en cuenta usar hosting arrendado, el usuario recibe mayor estabilidad en la transmisión - recepción de datos y posiblemente a un precio módico. Pero, antes de contratar un hosting, se debe tomar en cuenta el volumen de datos que se desea manipular, si son muchos, valdría la pena contratar uno. Pero si son pocos datos, como es el caso del dispositivo de control ON-OFF en la red Wlan, donde se tiene datos binarios de Activar y desactivar, sería un desperdicio de recursos pagar por un arrendamiento. Esas órdenes apenas demandan aproximadamente 64KB en una solo trama de tipo TCP o UDP.

Otra opción es obtener una cuenta de hosting gratuita que justamente sirven para aplicaciones de bajo contenido de datos. Existen varias plataformas de hospedaje web, que proporcionan la facilidad de obtener una cuenta gratuita.

De forma particular en el caso de este proyecto, se usa una red interna para la gestión y control de dispositivo diseñado.

2.6. DISEÑO DE APLICACIÓN RESIDENTE PARA ANDROID

Este proceso se enfoca en estudiar la forma de realizar la aplicación residente para el sistema Android, por medio de su programa de diseño de aplicaciones APP Inventor. Para ello se efectúa la descripción de los procedimientos que se realizarán a fin de desarrollar la aplicación residente que gestiona al módulo ON–OFF, por medio del ESP8266 a través de una red Wlan. A continuación se describen el procedimiento.

2.6.1. INICIALIZACION DE APP

Para iniciar el proceso de desarrollo de la aplicación residente de interfaz de usuario, en la plataforma APP Inventor para el sistema Android, se debe tomar en cuenta los siguientes pasos iniciales:

- Abrir una cuenta de correo electrónico en Gmail.
- En el buscador se ingresa a la plataforma APP Inventor, donde aparece la página de presentación como se observa en la Figura 2.12.

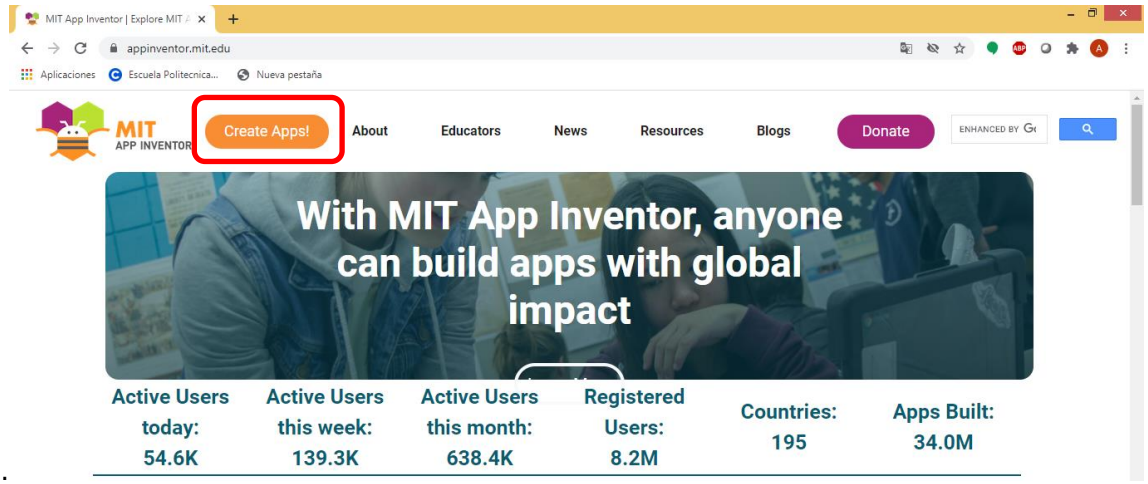

Figura 2.12. Página de Ingreso de APP Inventor

- Después se activa el botón *Create Apps!*. Al hacerlo se abre la ventana de ingreso pero por medio de la cuenta de correo electrónico.
- Luego aparece la ventana de *Gestión de Proyectos*. En esa ventana se debe presionar el botón, *Start new Project.* Además, se puede cambiar el idioma para facilidad del diseñador, para el caso particular de este trabajo de titulación, el idioma en *Español*. Tal como se observa eh la Figura 2.13.

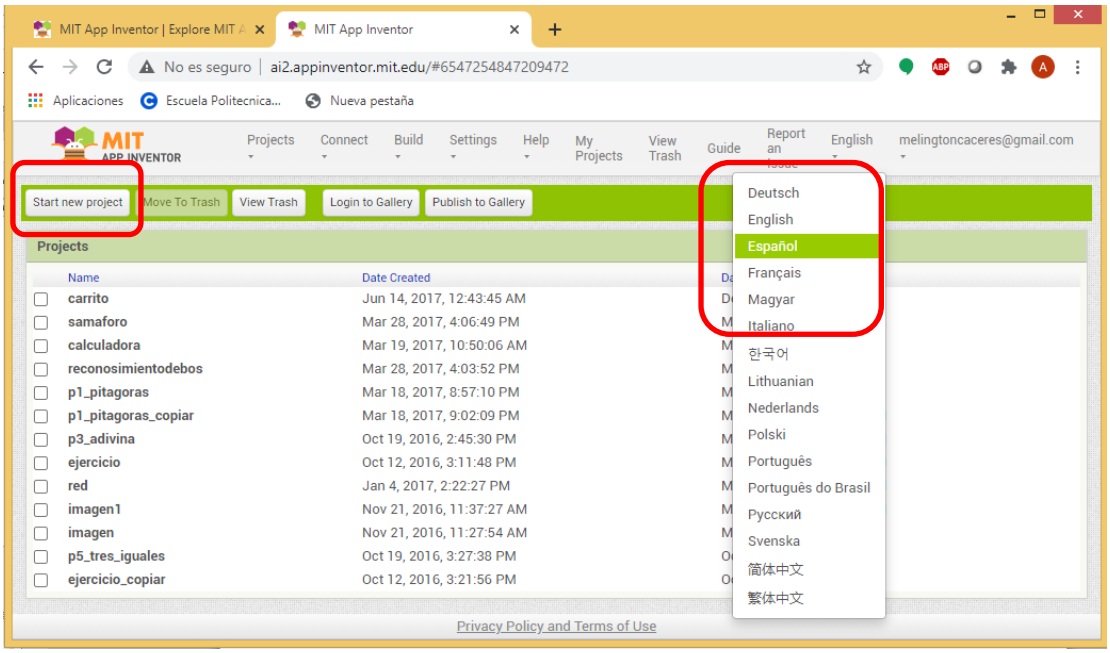

Figura 2.13. Ventana de Proyectos e Idioma

2.6.2. DISEÑO GRÁFICO DE LA INTERFAZ

Desde este punto en adelante, se inicia el diseño de la aplicación residente que controlará el módulo ON-OFF por medio del ESP8266. Por lo tanto, se prosigue con los procesos que se detallan a continuación.

- Después de dar click en el botón *Start new Project*, Entonces se abre una ventanilla, que solicita poner el nombre de la nueva aplicación, para poder proseguir. Para el caso de este proyecto, el nombre que se designa es *Control_ON_OFF.* Entonces aparece la ventana de diseño[62], como se puede apreciar en la Figura 2.14.

[62] Esta ventana se describió en el apartado 1.14.2.

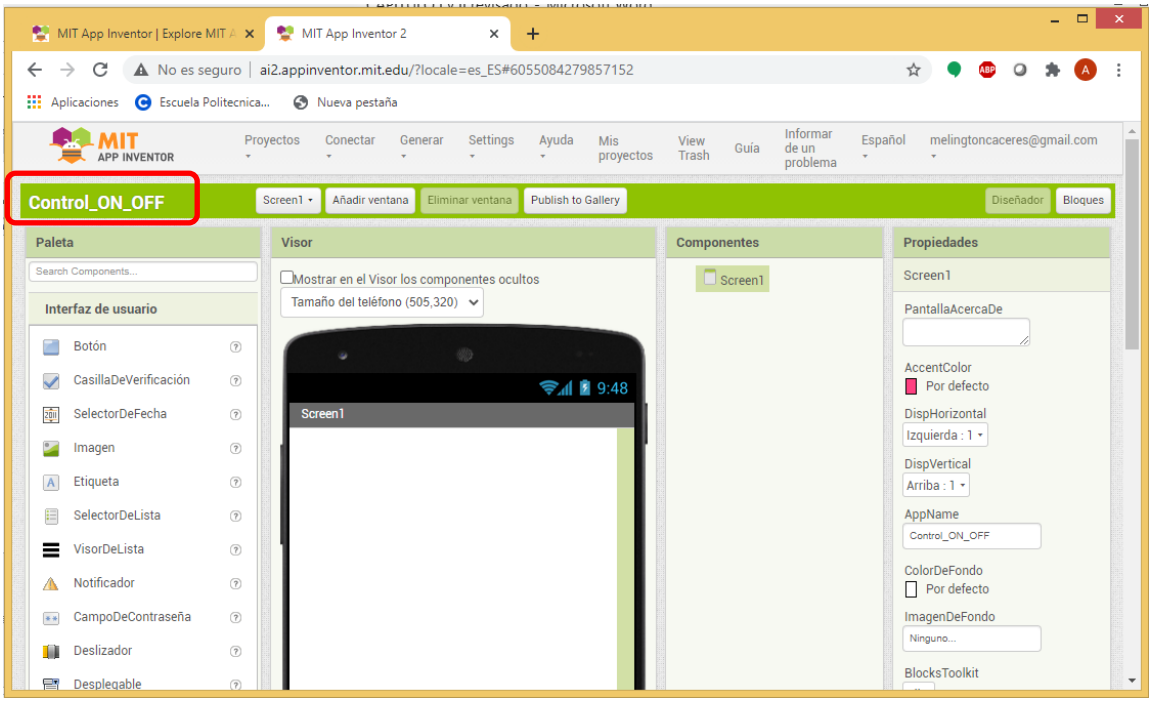

Figura 2.14. Ventana de diseño de Aplicación

- En la ventana de diseño, se procede a desarrollar el interfaz gráfico de usuario, para lo que se arrastra desde el área de Objetos o *Paleta* que se encuentra en la parte izquierda de la ventana, los objetos que serán parte del interfaz gráfico de usuario, tal como se puede mirar en la Figura 2.15.

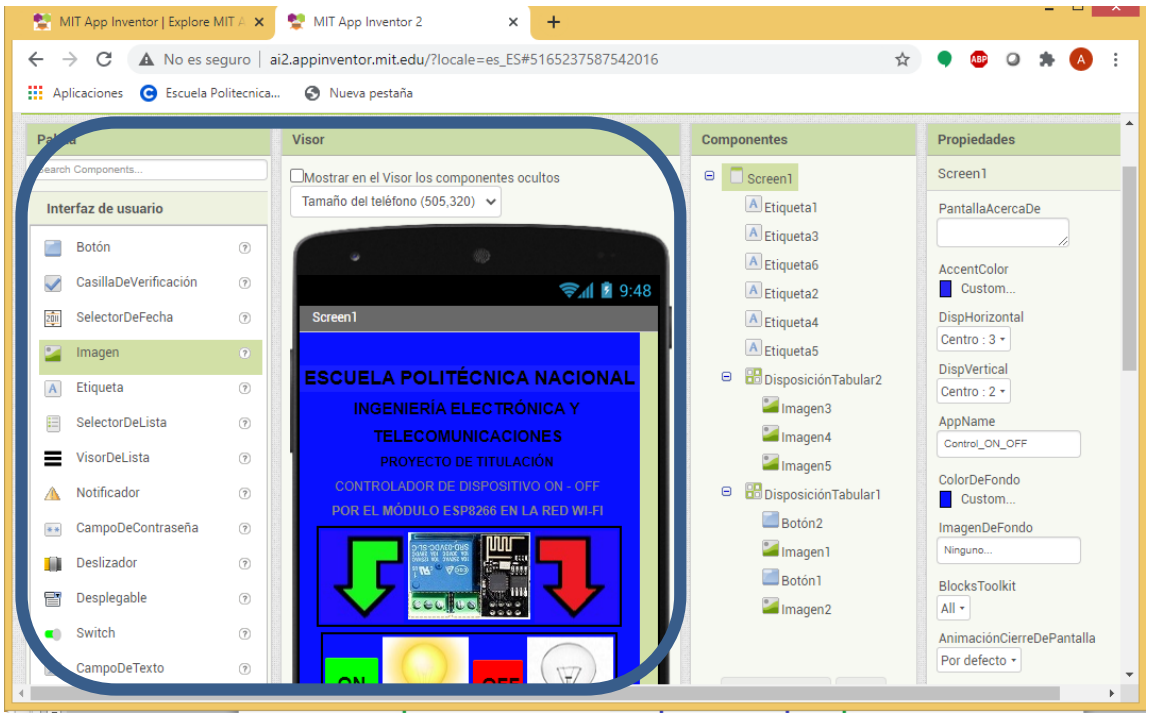

Figura 2.15. Ventana de Diseño de Interfaz de Usuario

- Los objetos que forman parte de la interfaz de usuario se pueden visualizar en pantalla simulada de la parte central media, según la Figura 2.15.
- En la parte central izquierda, se puede apreciar el área de componentes que se muestran en la Figura 2.15. En el área se detallan los componentes que contendrá la interfaz de usuario a diseñarse.
- En la misma ventana de la Figura 2.15, también se encuentra el área de Propiedades. Esa área permite al diseñador definir la mejor presentación para tener un atractivo visual del usuario final de la aplicación. Puede tener varios aspectos visuales y de presentación para que el usuario se sienta llamado la atención por el diseño gráfico.

2.6.3. PROGRAMACIÓN DE LA APLICACIÓN

Finalmente, en este proceso se halla la parte más importante del diseño de la aplicación residente, la programación de acciones de la misma. Para ir a ese punto se procede efectuar ciertos procesos que se describen a continuación:

- En la ventana de diseño, en la parte superior derecha se encuentra un botón que está nombrado como *Bloques* o *Blocks.* Eso se puede mirar en la Figura 2.16.

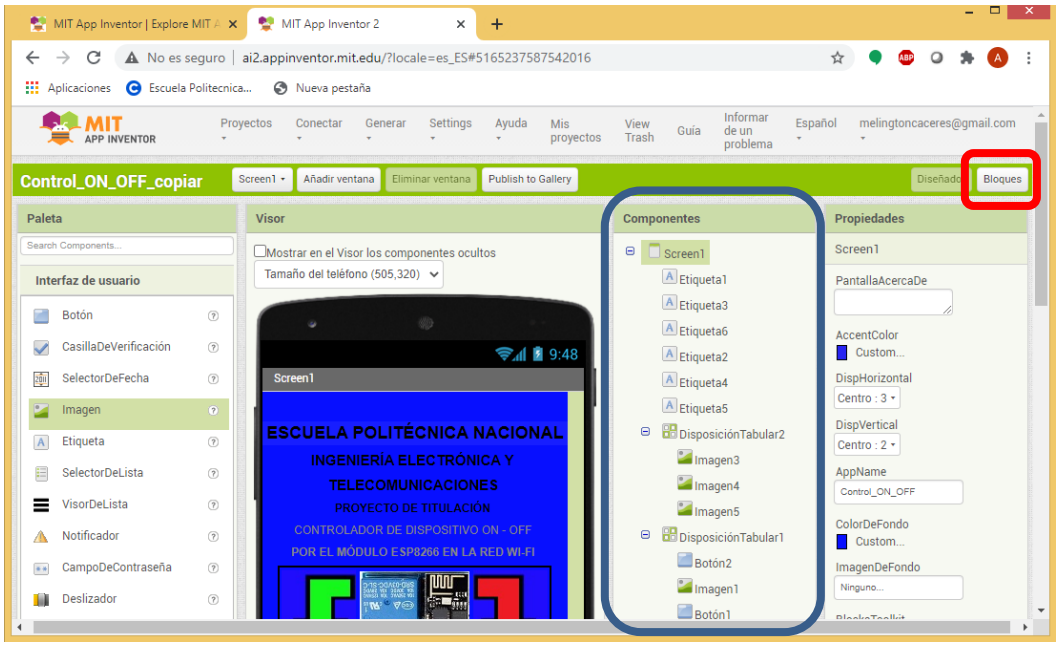

Figura 2.16. Botón Bloques

- Al momento de dar un click sobre el botón *Bloques*, aparece adjunta la pantalla de programación de los componentes que se han usado en la ventana de diseño, tal como se puede observar en la Figura 2.18.

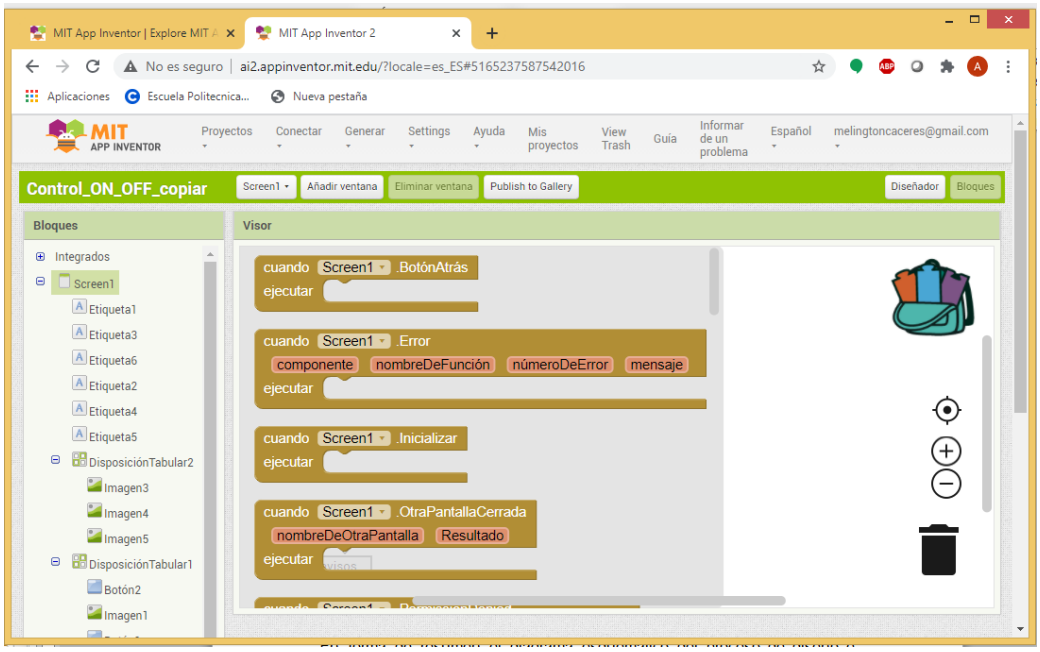

Figura 2.18. Área de programación

- En esa ventana se pueden observar en la división del lado izquierdo, los componentes que se usaron en la ventana de diseño. Pero al escoger cada uno de ellos aparecen en el área de la derecha, los que cuentan con sus respectivos bloques de programación. Esos bloques permiten realizar la programación POO[63], donde se va escogiendo dentro de cada bloque, las acciones necesarias para el funcionamiento objeto, eso depende del desarrollador. Pero esa parte del proceso finalmente se describe en el siguiente Proceso que es parte de este escrito del trabajo de titulación.
- Algo que se debe tomar en cuenta es que si se requiere realizar cambios o mejoras en la aplicación, se puede regresar a la ventana de diseño por medio de dar click en el botón *Diseñador* o *Designer*, el mismo que se encuentra en la parte superior derecha y se puede observar en la Figura 2.18.

[63] Esta programación se explicó en el apartado 1.13.3.

Como se ha explicado, de esa manera se realiza el cuarto proceso del esquema de la Figura 2.1, que es el de diseñar la aplicación residente para la gestión del dispositivo de control ON-OFF por medio del módulo ESP8266 a través de una red Wlan.

En forma de resumen se presenta un diagrama esquemático que recoge los procesos que se requieren para el diseño y la implementación del presente trabajo, tal como se muestra en la Figura 2.19.

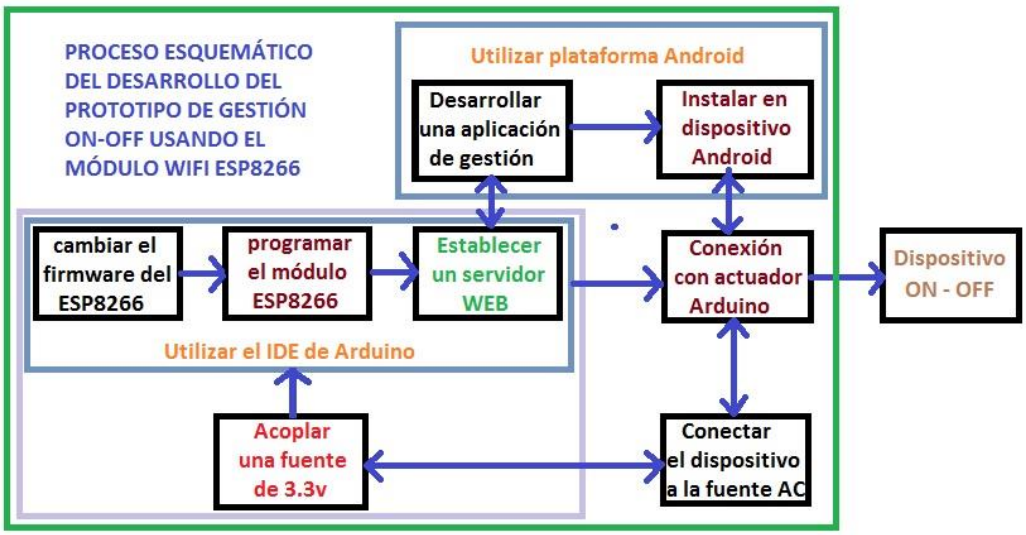

Figura 2.19. Diagrama de bloques para el diseño

De ese esquema solo faltaría por realizar la implementación del hardware y software, además del análisis de resultados del producto final. Procedimientos que se pasan a realizar a continuación en el siguiente capítulo de este trabajo.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se realiza la implementación de lo revisado en el capítulo anterior, por lo que es necesario ir detallando paso a paso los pormenores de forma práctica. Así se procede a cumplir con el objetivo de este trabajo que es implementar un dispositivo de gestión y control ON-OFF, por medio de la modificación del firmware del módulo ESP8266 con interfaz residente en Android a través de una red Wlan.

3.1. SISTEMA DE GESTIÓN DE ENERGÍA

Como se mencionó antes, la energización del sistema se realiza por medio de una fuente de energía alterna, que es la que alimenta al dispositivo ON-OFF, que se va a controlar, en este caso un foco. Proceso que se explica a continuación.

3.1.1. ENERGIZACIÓN GENERAL

La alimentación general del sistema de gestión y control ON-OFF, es la misma fuente de voltaje alterno que energiza al foco, que es el dispositivo que se controla.

Se debe resaltar que se controlará al foco, por medio de un sistema electrónico conformado por dispositivos que trabajan con voltaje directo. Esos dispositivos se basan en la plataforma Arduino, como el relé que trabaja a 5V y el ESP8266 que usa 3.3V. Por esa razón, como se expuso en el capítulo anterior, se requiere una fuente regulada de 5V, o lo que es similar, un cargador de teléfono móvil, las que regulan el voltaje alterno hasta convertirlo a 5VDC y en este caso tiene hasta 1A, tal como se puede visualizar en la Figura 3.1.

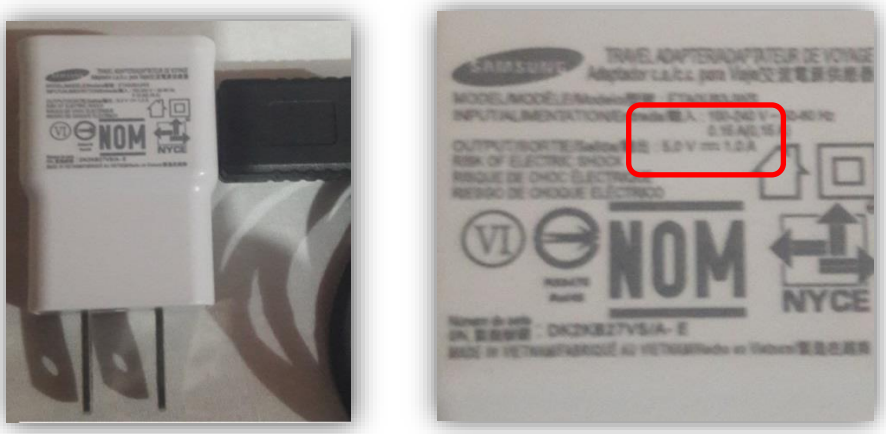

Figura 3.1. Fuente regulada de 5VDC - 1A

Para usar la fuente de 5VDC, se le conecta un cable USB y se hace las derivaciones de los cables de energía, que por lo general son los de color rojo y negro. Así, se logra usar los pines de energía de cable, tal como se observa en la Figura 3.2. Donde el cable rojo es el pin positivo (+) y el cable azul es el pin negativo (-). Después, se verifica si se obtiene los 5V con un multímetro y el resultado es correcto.

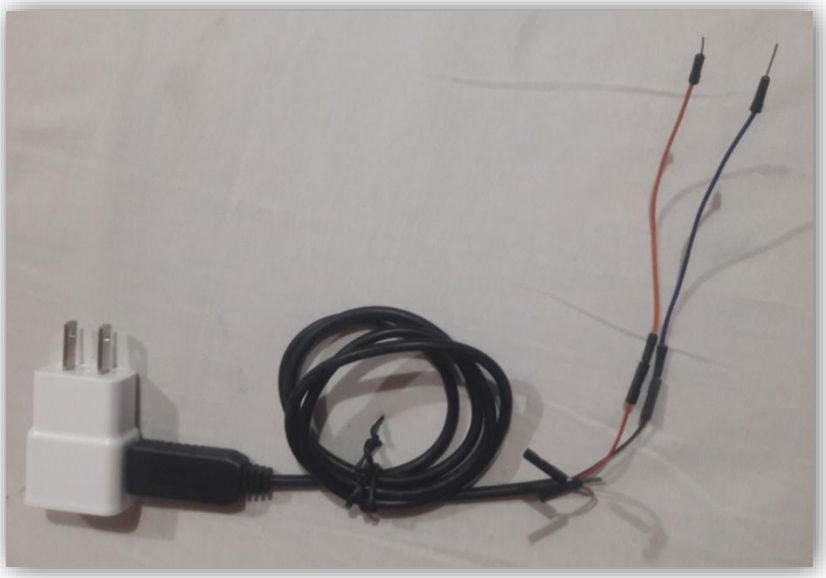

Figura 3.2. Cable de energización

Para continuar con el diseño del prototipo procede a conectar, en un protoboard la fuente de 5V, la misma que sirve de alimentación al relé y una rama al ESP8266 por medio del regulador LM1117 a 3.3V. Eso se puede revisar en la Figura 3.3.

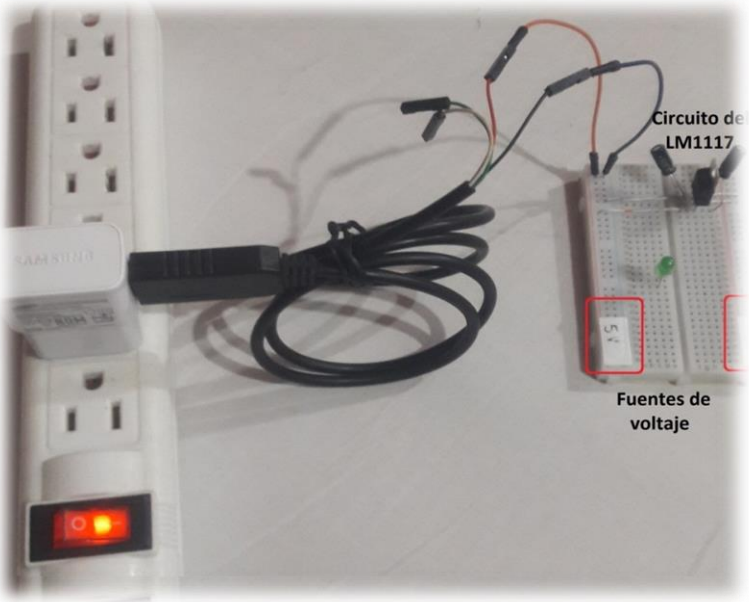

Figura 3.3. Protoboard con sus dos fuentes

3.1.2. FUENTE PARA ESP8266

Puesto que se requiere una fuente de 3.3VDC, para dar energía al módulo ESP8266, se usa la misma fuente regulada antes mencionada para ese propósito. Se usa el circuito

esquemático de la Figura 2.4[64], de tal forma que se ubica con esa conexión los capacitores y el regulador LM1117.

Es importante verificar si el circuito descrito cumple con lo esperado, o sea, proporcionar 3.3V, caso contrario no se puede suministrar la energía al ESP8266. Tomando en cuenta los requerimientos de los circuitos y su forma de conectarlos, el circuito de la distribución energética queda conformado como se muestra en la Figura 3.4, por un lado, la fuente de 5V y por el otro lado una fuente de 3.3V.

Pero es necesario verificar su funcionamiento, por lo que se hace uso de un multímetro, tal como se exponen en la Figura 3.4. Como se puede apreciar el voltaje es correcto, se obtiene 3.2VDC.

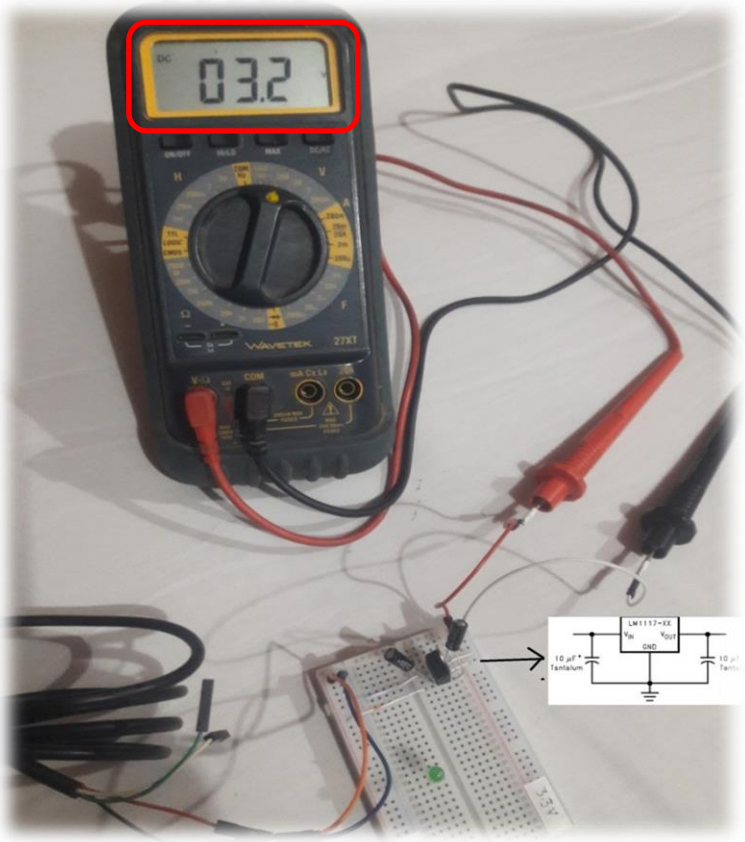

Figura 3.4. Fuente de 3.3VDC

Con relación a la corriente, se había estudiado que se requería al menos unos 300mA, para el máximo consumo del ESP8266 y unos 90mA para el relé, por lo que se estimaba unos 400mA de consumo. Por esa razón se asumía conseguir una fuente de 500mA. Pero

[64] Se analizó en el apartado 2.4.1.

el caso es que con la fuente que se logró hallar puede proveer hasta 1A, por tanto, se tiene suficiente intensidad para el trabajo del dispositivo esperado.

3.2. PROGRAMACIÓN DEL MÓDULO ESP8266

En este apartado se procederá a trabajar exclusivamente con el módulo ESP8266, pues este es el principal actor del dispositivo que se desea implementar. Entonces se debe efectuar varias tareas para lograr el objetivo deseado, por lo que se procede a detallar cómo se desarrolla ese fin.

3.2.1. MODIFICACIÓN DEL FIRMWARE DEL ESP8266

Este paso es fundamental para poder programar el ESP8266, pues como se explicó en la primera parte de este trabajo, el módulo tiene por defecto un firmware basado en comandos Hayes o AT[65]. Eso hace que el alcance aplicativo del ESP8266, sea limitado por la dificultad de programar en un lenguaje basado en códigos AT, además, que no puede trabajar de forma independiente.

Entonces como primer paso se debe verificar que en el IDE de Arduino se reconozca como una tarjeta del tipo Arduino al módulo ESP8266 y además se hayan cargado a la IDE, las librerías de la placa ESP8266[66]. Desde este momento se la denomina placa al módulo ESP8266, pues por tener su propio microcontrolador la IDE de Arduino lo reconoce como una placa más. Ese proceso se puede verificar revisando que en *Herramientas* y en Gestión de Tarjetas, se encuentre la placa ESP8266, como se puede observar en la Figura 3.5a y las librerías, como se puede observar en la Figura 3.5b.

[65] Se analizó en los apartados 1.5 y 1.12
[66] Este paso se explicó en el apartado 2.5.3

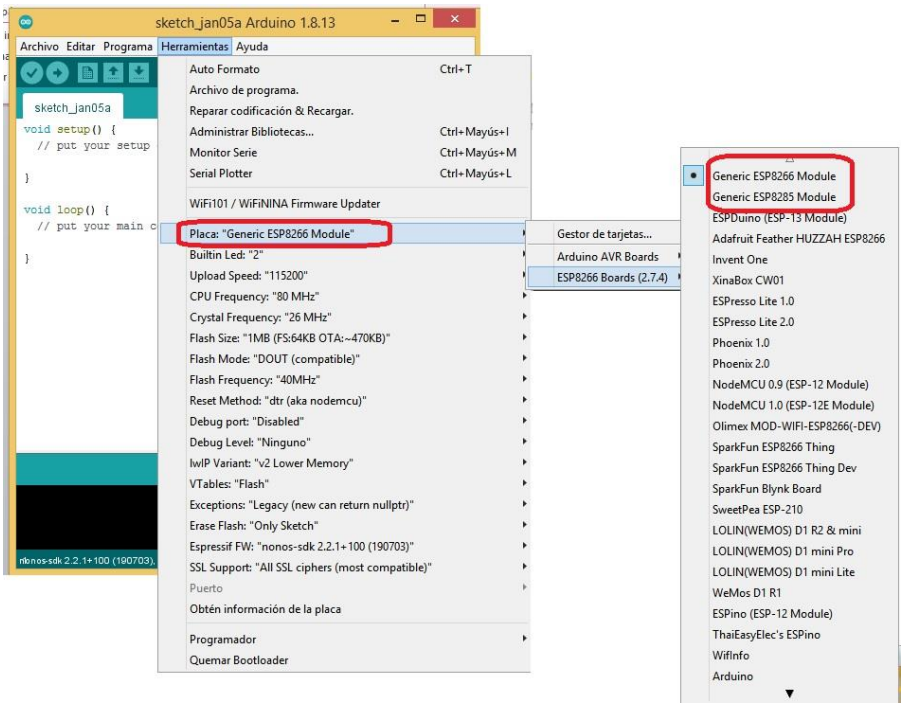

Figura 3.5a. Placa ESP8266 y librerías en el IDE de Arduino

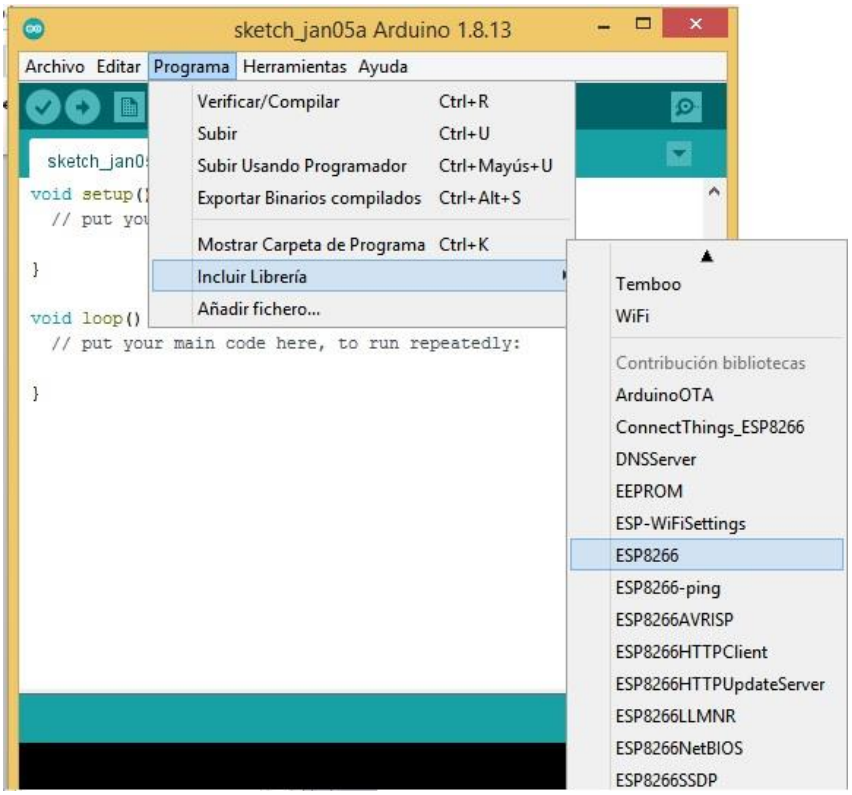

Figura 3.5b. Librerías del módulo ESP8266 en el IDE de Arduino

Por las gráficas de las Figuras 3.5 a y b, se puede deducir que la IDE de Arduino, ya puede reconocer y gestionar cualquier placa que contenga el módulo ESP8266.

Sin embargo, todavía no se puede realizar nada, a menos que se conecte la placa ESP8266 al IDE. Ese proceso se realiza de forma indirecta, por medio de una placa Arduino, la que la que actúa como FTDI.

Para que la placa de Arduino sirva como intermediario, se tiene que realizar algo imprescindible, antes. Retirar su microcontrolador, pues si no se efectúa este paso, cualquier programa que se quiera subir o cargar al ESP8266 se podría guardar en el microcontrolador de Arduino, por lo tanto, se remueve el Atmega32 del Arduino UNO, como se aprecia en la Figura 3.6.

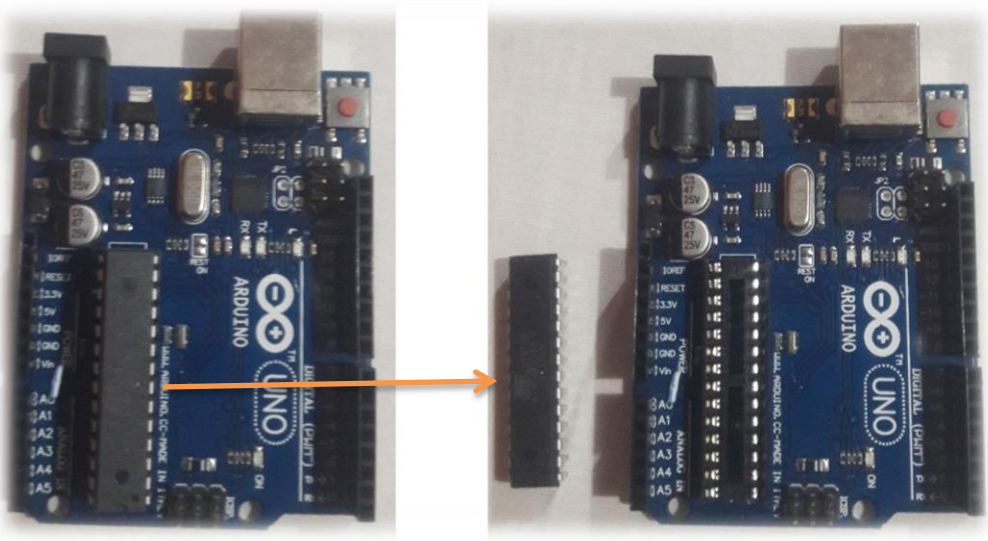

Figura 3.6. Arduino con y sin microcontrolador

Luego de este paso, se conecta el módulo Wi-Fi, de la forma que se visualiza en la Figura 3.7. El cable azul de la gráfica, hace que el GPIO0 se ponga en 0_L, eso permite que se active el modo Bootloader[67] del ESP8266.

[67] Este proceso se estudió en el apartado 1.11.1

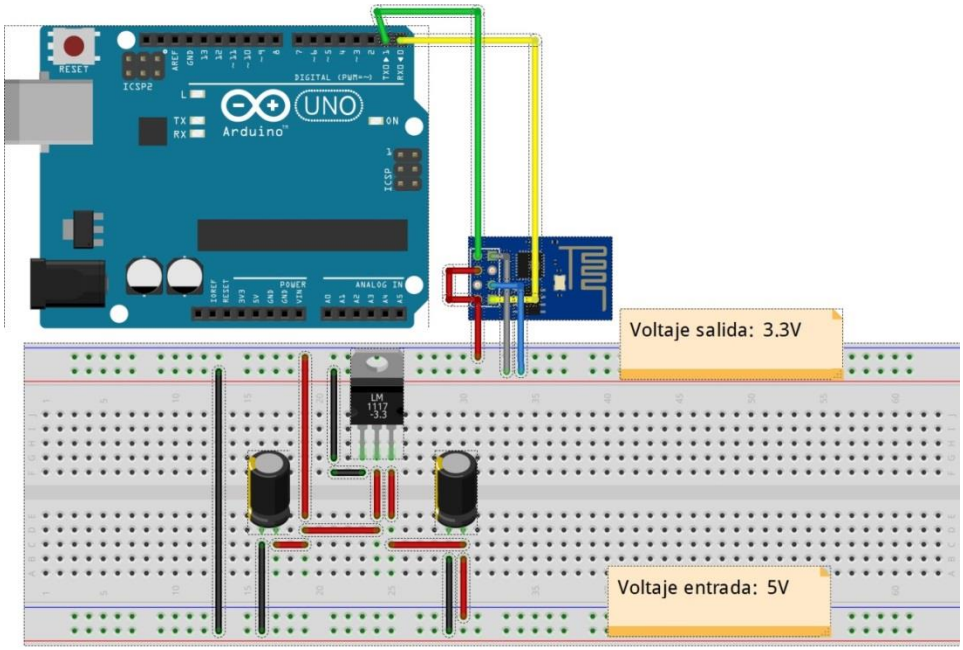

Figura 3.7. Conexión para cambiar el Firmware del ESP8266

Entonces, ya conectado de esa forma el circuito, la fuente de 3.3V, el Arduino y el ESP8266. Se abre un *sketch* para iniciar la programación del módulo Wi-Fi.

Se inicia la modificación de Firmware en el módulo ESP8266, con solo ubicar la librería *ESP8266WiFi.h*, como cabecera del programa, Figura 3.8. Esa librería permite que al cargar el *sketch* sobre el modulo el Firmware anterior se borre y el actual permita el acceso a la memoria flash de módulo Wi-Fi.

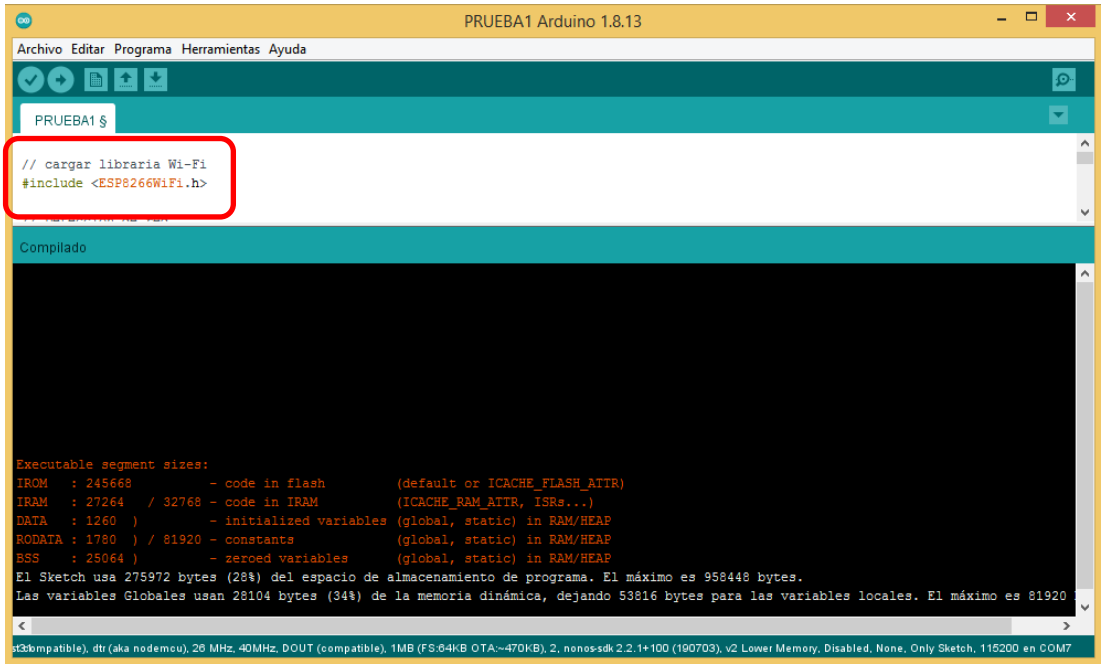

Figura 3.8. Librería que contiene el Firmware del IDE de Arduino

Al dar click sobre el botón  se compila el programa y se reconoce a la placa ESP8266, por lo que presenta la información sobre la misma tal como se observa en a gráfica de la Figura 3.8 en parte inferior.

Luego cuando se presiona el icono  , para cargar el programa sobre el módulo Wi-Fi. Así, se procede a borrar del área de programación en la memoria del ESP8266, eliminando el Firmware anterior y colocando el acceso para el IDE de Arduino. En ese momento existe transferencia directa de comunicación entre la placa de Arduino y el ESP8266, se lo nota por los leds de transmisión en las dos placas.

Además, visualmente se puede verificar que en el sketch se presenta la carga del programa sobre el módulo ESP8266, eso se puede verificar en la Figura 3.9, donde se muestra que se ha activado el *esptool.py* o la librería de herramientas del ESP8266 para Arduino, para conectarse con la memoria del mismo.

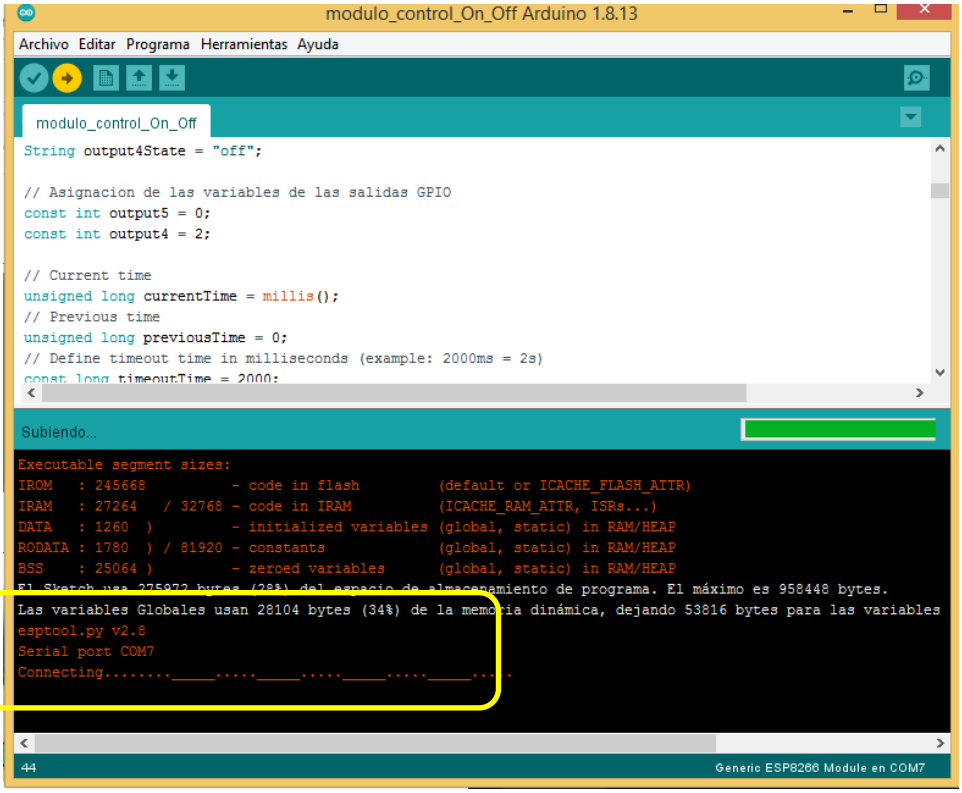

Figura 3.9. Activación del módulo ESP8266 por Arduino

3.2.2. MODO DE TRABAJO

Es importante destacar que dentro del mismo programa se realiza el desarrollo de la funcionalidad que tendrá el módulo ESP8266. Como se describió anteriormente[68], se debe determinar si trabajará como estación o como AP. Para el caso particular de este trabajo se define que sea como Estación dentro de una red Wlan.

Entonces se programa los requisitos de red, como el SSID, la contraseña, la IP, entre otras. Para ello se expone dentro del programa general la declaración de variables y los valores de red con las instrucciones.

```
// Deteccion de red
const char* ssid     = "EL_NOMBRE_DE_LA_RED ";
const char* password ="CONTRASENA_DE_LA_RED ";
```

3.2.3. DESIGNACIÓN DE IP

Después del enlace a la red Wlan, el módulo que trabaja como Estación, necesita una IP. Normalmente los enrutadores están programados para trabajar como servidores DHCP, por cuanto, el módulo ESP8266 podría también obtener una IP dinámica. Pero el problema de tener una IP por DHCP es que para poder programar nuevamente el módulo y su aplicación de interfaz domótico de usuario, se debería primero, determinar cuál es su IP y acceder a ella, que muchas veces es un proceso complejo.

Por lo tanto, se ve muy conveniente establecer una IP fija, dentro del rango de IPs de la red, para efectuar la gestión de manejo del módulo y su interfaz domótico. Entonces se programa una IP estática por comandos, lo que se citan a continuación.

```
// Configuración de la IP estática.
   IPAddress local_IP(192, 168, 100, 12);
   IPAddress gateway(192, 168, 100, 1);
   IPAddress subnet(255, 255, 255, 0);
```

3.2.4. MANTENIMIENTO DE ENLACE

El módulo de control ON-OFF, debe permanecer siempre activo, eso requiere que permanezca activo en la red, ya que tiene una IP estática. Generalmente los enrutadores

[68] Se explicó en el apartado 2.5.5

tienen un proceso de actualización de las IPs, siguiendo el proceso que se presenta a continuación:

- El cliente envía un mensaje por medio de broadcast “DHCP Discover”. El fin es, encontrar un Servidor DHCP que este escuchando.
- Si hay un servidor activo, este responde con un mensaje de “DHCP Offer”. En ese mensaje se envía la IP, mascara de red y puerta de enlace, necesarios para enlazar al usuario.
- El cliente recibe el mensaje y reenvía un mensaje “DHCP Request”, por medio de broadcast.
- Luego el servidor envía un “DHCP ACK”, en este se le asigna la dirección propuesta para que la use. Pero esas IPs se las permite de forma temporal. Por lo tanto, se requiere que el cliente y el enrutador envíen mensajes de forma continua para avisar su continuidad en el enlace.

Así, el enrutador elimina y ofrece las IPs inactivas manteniendo su tabla de IPs actualizada.

Por esa razón, a pesar que la IP es estática, en el caso del módulo de control ON-OFF, debe enviar su estado de actividad cada cierto tiempo, como en este caso es 2 segundos, si no se encuentra activa, puede que se considere como IP libre. Entonces se debe programar una forma de mantener la actividad del enlace, desde el módulo al router, lo que se expone a continuación.

```
// tiempo de conexion
unsigned long currentTime = millis();
// tiempo de inicio
unsigned long previousTime = 0;
// tiempo de salida 2 segundos
const long timeoutTime = 2000;
```

Esa programación garantiza que cada 2 segundos se envía una señal de actividad al enrutador.

3.2.5. SERVIDOR WEB

Con el fin de permitir la interacción del módulo de control ON-OFF y el interfaz domótico de usuario, se requiere que se haga un tratamiento de datos por medio de la red. Entonces, se cree que es muy conveniente diseñar un servidor WEB.

Tomando en cuenta que el módulo ESP8266, tiene espacio para albergar un servidor básico, se aprovecha esa opción. Cabe resaltar que la IDE de Arduino ya tiene integrado en las librerías del ESP8266, la capacidad de usar funciones para desarrollar el servidor en el programa que comanda el módulo Wi-Fi. Los comandos permiten la gestión del Servidor WEB son los que se citan a continuación.

```
// Activar el puerto del Servidor web
WiFiServer server(80);
// Variable para respuesta HTTP
String header;
```

En primer lugar, se activa el puerto que usará el servidor, que para el servicio WEB es el puerto 80.

Después se asigna una variable para los datos del ESP8266 en el servicio HTTP. Esa variable permite llevar la información recibida de forma virtual por el módulo Wi-Fi. Esa información es necesaria para poder activar o desactivar el GPIO, que son los encargados de físicamente realizar la actividad de ON-OFF por el módulo.

```
// activacion de GPIOs on - off
            if (header.indexOf("GET /5/on") >= 0) {
              Serial.println("GPIO 5 on");
              output5State = "on";
              digitalWrite(output5, HIGH);
            } else if (header.indexOf("GET /5/off") >= 0) {
              Serial.println("GPIO 5 off");
              output5State = "off";
              digitalWrite(output5, LOW);
            }
```

Luego, se desarrolla el código de la presentación de la página virtual del servidor. Para el servidor WEB, se procede a programar usando código HTML. La presentación de la página puede realizarse de forma externa, quizá usando programas de desarrollo más especializados. Pero como se trata de una página que maneja pocos datos, se puede programar en el mismo espacio, siendo una presentación sencilla que permita activar y desactivar los pines GPIO, para encender y apagar el relé que a su vez, trabaja con el foco.

Además, se debe resaltar que la página del servidor WEB con esos comandos es sumamente básica, pero la presentación al usuario debe ser más atractiva para el interfaz de usuario en Android desarrollada en APP Inventor.

```
// Código para la página HTML
   client.println("<!DOCTYPE html><html>");
   client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
    client.println("<link rel=\"icon\" href=\"data:,\">")
 // Botones estilo CSS  on/off
 // cambio dde color y fondo en la activacion
    client.println("<style>html { font-family: Helvetica; display: inline-
block; margin: 0px auto; text-align: center;}");
    client.println(".button { background-color: #195B6A; border:
none; color: white; padding: 16px 40px;");
     client.println("text-decoration: none; font-size: 30px; margin:
2px; cursor: pointer;}");
```

Puesto que se trabaja con los datos virtuales de la página WEB, la que contiene un botón virtual ON-OFF. Los comandos de gestión de esas acciones son las que se describen en la programación que se citan en el recuadro a continuación

```
// Estado de los botones ON/OFF para GPIO4
      client.println("<p>GPIO 4 - State " + output4State + "</p>");

 // Si el output4State es off, se activa el botón ON
 if (output4State=="off") {
     client.println("<p><a href=\"/4/on\"><button
class=\"button\">ON</button></a></p>");
         } else {

     client.println("<p><a href=\"/4/off\"><button class=\"button
button2\">OFF</button></a></p>");

         }
```

En esta parte del desarrollo se presenta un botón simple para encender y apagar. Al activarse o desactivarse el botón, el dato se trasfiere del servidor WEB a los pines GPIO, lo que permite transferir de forma física la actividad al relé.

Hasta este punto, solo se ha trabajado dentro del entorno del ESP8266. Ahora para poder activar y desactivar, el botón diseñado en la página del servidor WEB, se requiere que se acceda al servidor, por medio de HTTP, lo que resulta tedioso, ya que el usuario debe conocer cómo hacerlo por medio de la IP para lo que requiere usar los servicios de la red directamente, abriendo la página del servidor y realizar el trabajo en ella.

Pero es bien conocido que la mayor parte de usuarios no tienen los conocimientos del manejo de una página WEB, por lo que es completamente complejo el uso de este prototipo. Por esa razón, como ya se explicó anteriormente, se ha diseñado una aplicación residente de gestión domótica, más accesible y amigable al usuario, la misma que es desarrollada en APP Inventor, la que trabaja sobre el sistema operativo Android.

3.3. DISEÑO DE APLICACIÓN ANDROID

Para realizar el manejo de forma virtual del dispositivo de control ON-OFF, gestionado por el módulo ESP8266, dentro de la red Wlan. Se diseña una aplicación residente que se realiza por un programa de desarrollo de APPs para sistemas Android, esa es APP Inventor, como se describió anteriormente. Entonces se describe como se empieza el desarrollo.

3.3.1. DISEÑO GRÁFICO

Esta parte del desarrollo permite que el programador realice una App, que sea residente en algún dispositivo Android y sea atractiva de forma visual, además que sea funcional desde el punto de vista del usuario. Para crear ese tipo de presentación se usa el programa de desarrollo de aplicaciones APP Inventor, realizando los procesos antes descritos en el capítulo anterior. Por lo tanto, se procede a usar los siguientes objetos que permiten realizar lo esperado.

Label o Etiquetas. Contienen las presentaciones de texto. Se arrastra hacia el área de trabajo, que es la simulación de una pantalla de celular.

Arreglos en Tabla o Contenedores de objetos. Se usan estos contenedores para ubicar los otros objetos en espacios específicos del área de presentación.

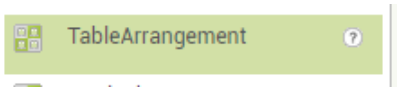

Imagen. Este objeto permite presentar imágenes de cualquier tipo dentro de áreas específicas dentro de la presentación.

Botón. Este objeto ofrece una gráfica con la funcionalidad de permitir el dar click, lo que hace más accesible al usuario.

Entonces la presentación de la aplicación para el usuario es la que se presenta en la Figura 3.10.

Figura 3.10. Presentación del Control Virtual del Módulo de control del dispositivo ON-OFF

Las propiedades como color, tipo y tamaño de letra del texto, color del screen, alineación de los objetos y otros más, para cada uno se los puede escoger en el recuadro del extremo derecho.

3.3.2. PROGRAMACIÓN DE LA APLICACIÓN

Como se explicó en el apartado 2.6.3, la programación de los objetos se realiza en el área destinada para ello, después de presionar el botón *Bloques*. Aparece el área de programación, desde donde se puede programar las acciones que deben tener los botones de ON y OFF.

La presentación que se ha diseñado para el usuario, consta de dos botones, uno con la acción ON y otro con la función OFF, como se exponen en Figura 3.11a.

Además, consta de imágenes que se activan o desactivan según la acción de los botones.

La programación en esta plataforma de desarrollo, se realiza por medio del uso de objetos que en el área de programación que se identifican como bloques de diversos colores, tal como se aprecia en la Figura 3.11b.

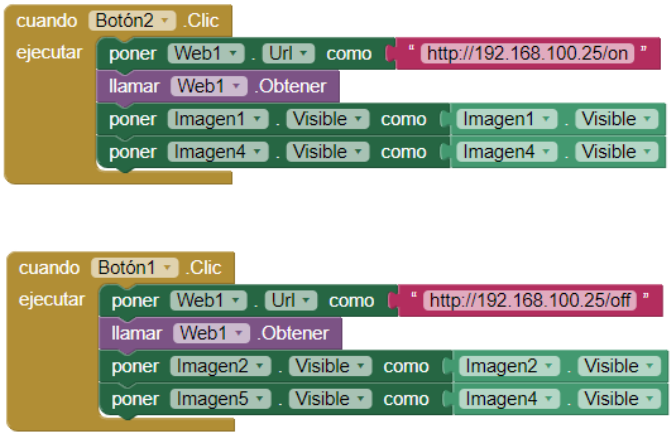

Figura 3.11a. Botones de la presentación **Figura 3.11b.**Botones programados

Los bloques de acciones se ubican dentro del bloque general que es un objeto seleccionado como el de primera actividad. Para el caso particular de esta aplicación,el bloque general es el objeto *Boton* , se comporta como un contenedor. Dentro de él, se van insertando otros bloques de objetos con acciones, que se concatenan una tras otra al dar click sobre el general.

El proceso de programación es en el orden que se expone. Por ejemplo, al dar **Click** sobre el **Boton2** (bloque de color café) se **Ejecuta** la acción se dirige al objeto pagina **Web1** en

su dirección **Url** (bloque de color verde), entonces se activa el servidor WEB en su **IP** junto a la acción de encender **ON**. Este proceso se visualiza enh la Figura 3.13.

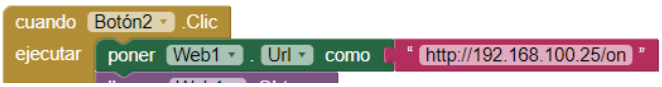

Figura 3.13. Programacion del objeto Boton

Puesto que se desea que la presentación diseñada, sea atrayente en sentido visual para el usuario. Se añaden dos imagenes que permiten hacer referencia a la acción de encender o apagar el dispositivo ON – OFF.

Seguido a los bloques antes descritos en la Figura 3.13. se añaden los bloques que se presentan en la Figura 3.14b. que hacen visible las imágenes de encendido o apagado según la Figura 3.14a.

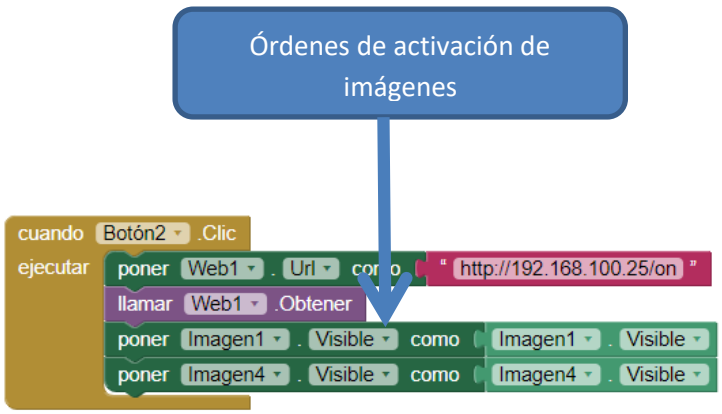

Figura 3.14a. Botón ON activado

Figura 3.14b. Programación del Botón activa las imágenes

De la misma forma se procede con el Botón1 que es el boton de OFF, pero se desactivan las acciones e imagenes anteriores, eso se puede observar en las Figuras 3.15 a y b.

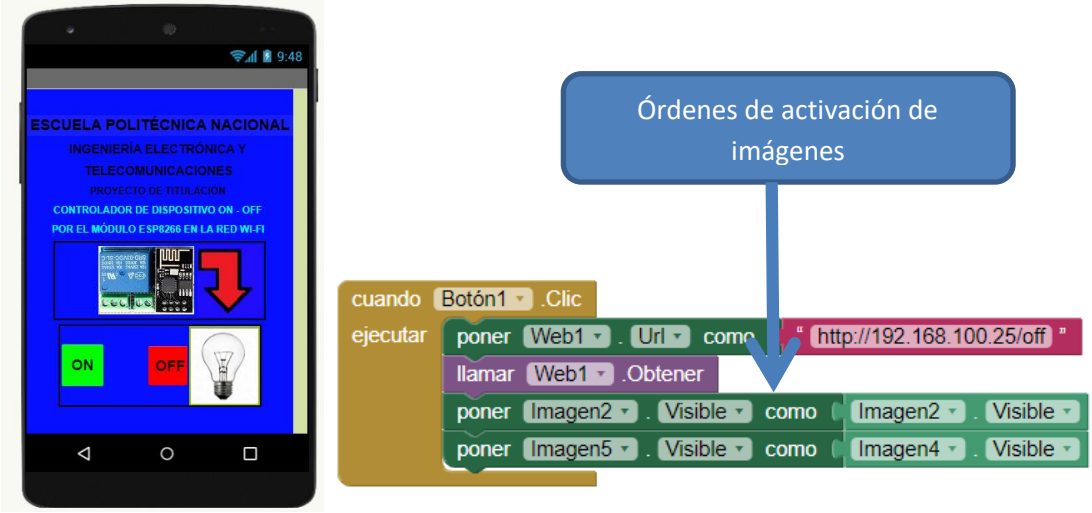

Figura 3.15a. Botón OFF activado **Figura 3.15b.** Programación del Botón activa las imágenes

3.3.3. ENLACE CON EL SERVIDOR WEB

Es importante resaltar que la presentación visual de la aplicación debe servir para activar y desactivar el módulo ON-OFF, por medio del software almacenado en el servidor del ESP8266 en la red Wlan. Por esa razón, se enfatiza que la programación de los botones descritos en el apartado anterior, permite relacionar por medio de Wi-Fi, la plataforma virtual (aplicación domótica de usuario en Android), con la plataforma física (módulo integrado ON-OFF).

En la Figura 3.16, se puede observar que además del bloque de ingreso a la IP del servidor WEB, existe dos órdenes importantes, que están en el bloque de color violeta. Las órdenes son ***llamar*** y ***Obtener***, eso permite que la orden ON, a lado de la IP del bloque rosa, se transmita al servidor WEB, que esté instalado en el software de control del ESP8266.

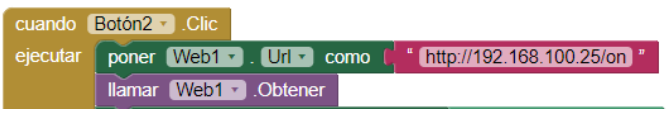

Figura 3.16. Activación del servidor WEB

Esto se puede corroborar por el lazo condicional que se expuso en el apartado 3.2.5, donde se verifica que si el servidor recibe la orden ***on*** el GPIO0 recibe un 1_L y si recibe un ***off*** el GPIO0 recibe un 0_L. Como se puede observar en el bloque de programación a continuación.

```
// Si el output4State es off, se activa el botón ON
 if (output4State=="off") {
      client.println("<p><a href=\"/4/on\"><button
class=\"button\">ON</button></a></p>");
          } else {

      client.println("<p><a href=\"/4/off\"><button class=\"button
```

Luego de haber terminado la programación de la presentación de gestión domótica, se procede a convertir en ejecutable la aplicación, por medio de un archivo .apk, tal como se observa en la Figura 3.17. Ese archivo se copia cualquier dispositivo Android y se ejecuta, así se activa la presentación y se puede usar para comandar el dispositivo de control ON-OFF, por medio del módulo ESP8266 y dentro de la misma red Wlan.

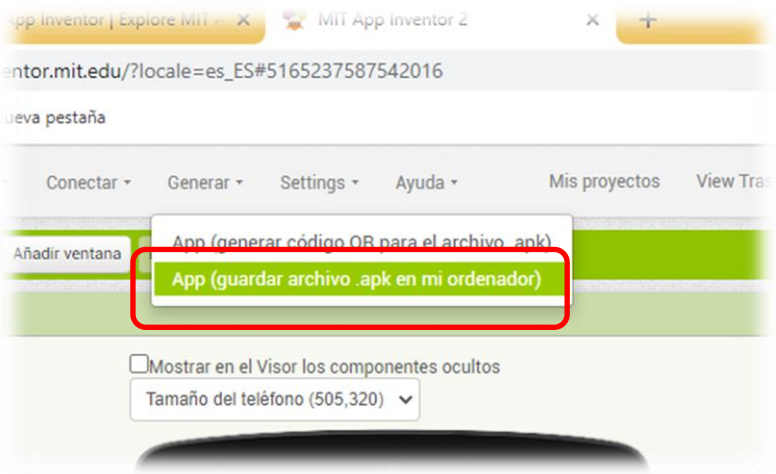

Figura 3.17. Proceso de guardado de aplicación en formato apk.

3.4. PRUEBAS DEL PROTOTIPO

Primero se efectuó el armado de la fuente de 5V a 3.3V, se usó un cargador de teléfono, ya se presentó antes. Los leds verdes permiten al usuario identificar la actividad

Luego se procede al cambió el firmware del ESP8266, usando la fuente de 3.3V y el Arduino (sin microcontrolador) que actúa como FTDI, como se evidencia en la Figura 3.18.

Figura 3.18. Cambio del Firmware del ESP8266 por medio de Arduino

Después se conecta el relé y el foco a controlar.

Figura 3.19. Relé conectado al foco

Entonces se conecta todo el sistema, que consta del ESP8266, con su nuevo Firmware, funcionando como Servidor WEB, con el relé y el foco. Se puede notar en la Figura 2.20.

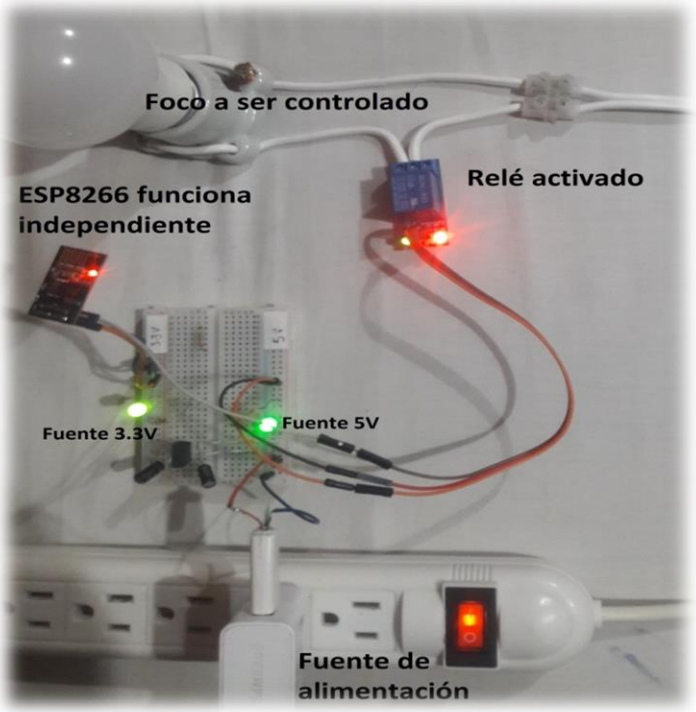

Figura 3.20. Sistema armado para pruebas

Se realiza pruebas de control y no existe ningún problema, en la gestión de encendido y apagado, tal como se puede mirar en la Figura 3.21.

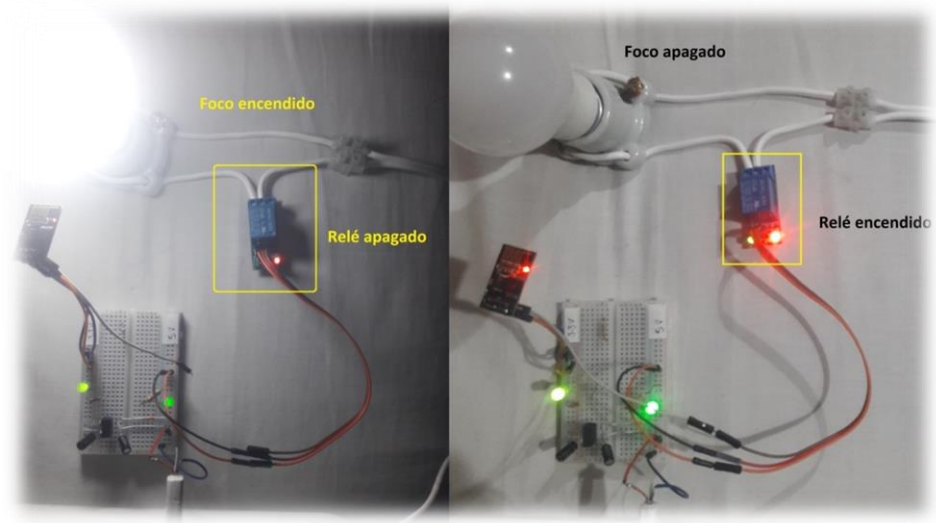

Figura 3.21. Pruebas de control ON-OFF.

Después de haber algunas pruebas de control, se comprime todo el dispositivo terminado en una sola pieza como se muestra en la Figura 3.22.

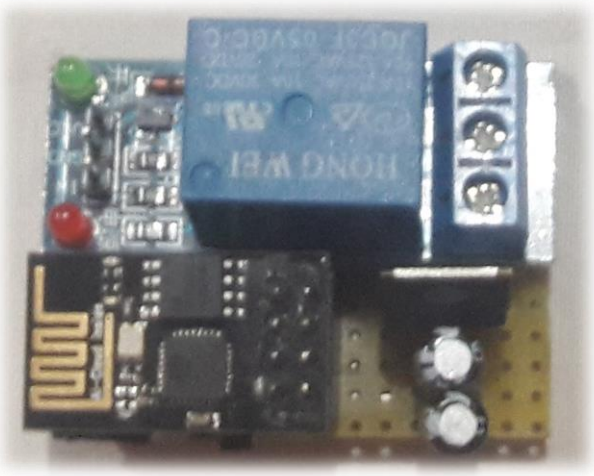

Figura 3.22. Dispositivo armado en una sola pieza

Es un sistema compacto, sus medidas son 5 cm de largo x 2,5 de ancho x 5 cm de alto. Pero en vista que el módulo resultante, debe ser programable, para darle mayor escalabilidad, se añade un arreglo mecánico, o un dip switch, para poder realizar las polarizaciones de los GPIO en la programación del ESP8266[69]. Tal como se observa en la Figura 3.23, dentro del cuadro de color amarillo.

Uno de los switch controla el GPIO2, que no debe estar conectado, mientras se realiza la programación, luego en el circuito activo, está conectado al relé. El segundo switch controla el GPIO0, que en la programación se conecta a cero y en mientras se activa el circuito se desconecta. El tercer interruptor se encarga de resetear el modulo completo, aunque no afecta la programación del ESP8266.

[69] La programación del ESP8266 se vio en el apartado 1.11.1 y 2.5.8.

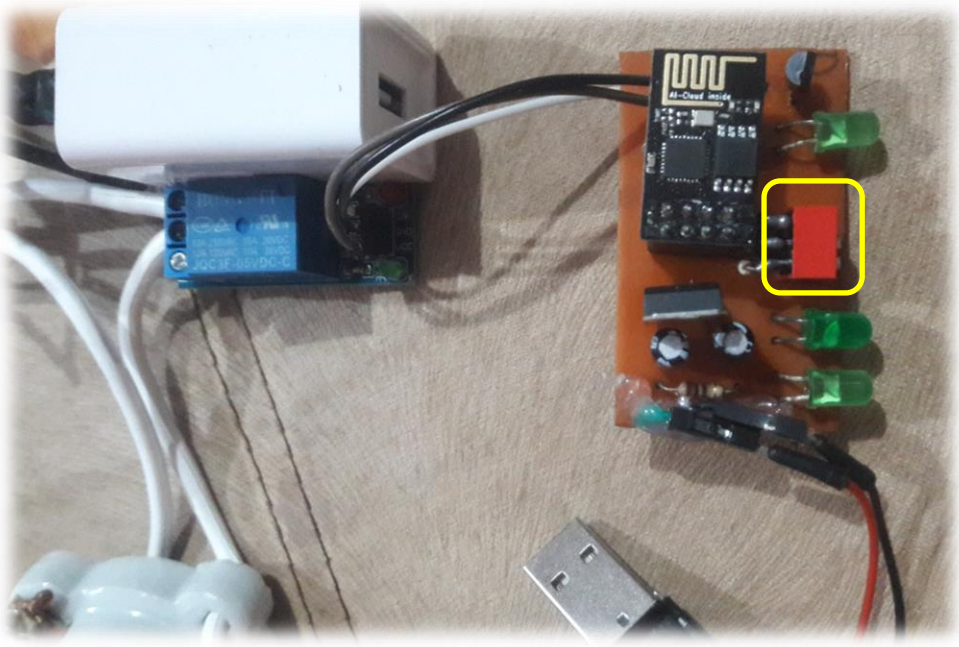

Figura 3.23. Dispositivo añadido el controlador de programación

Después de esa añadidura se procede a realizar las pruebas, pero con el dispositivo armado, tal como se puede observar en la Figura 3.23. Finalmente se prueba el dispositivo completo junto a la aplicación residente de Android, como se observa en la Figura 3.24.

Figura 3.24. Pruebas con el dispositivo armado

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

El trabajo de titulación presenta como resultado el prototipo de un dispositivo de control ON-OFF, modificando el firmware del módulo ESP8266 gestionado por su interfaz de usuario residente en el sistema Android por medio de una red Wlan. El dispositivo puede gestionar cualquier tipo de dispositivos que tengan esas dos acciones encender – apagar.

Se realizó la implementación de un sistema de alimentación energética usando una fuente regulada de 5V a 1A, para suplir de voltaje al módulo Wi-Fi ESP8266, mismo que trabaja a 3.3V y en su máximo trabajo de puede usar hasta aproximadamente 300mA.

También se efectuó la modificación de firmware del módulo ESP8266, ya que su software original solo le permite usar comandos AT, por lo que pierde funcionalidad. Para realizar el cambio de software se utilizó el IDE de Arduino, ya que, por tener un microcontrolador en su estructura física, la plataforma le atribuye propiedades de una tarjeta similar a las Arduino. Primero se instala la tarjeta ESP8266, luego se procede a la instalación y actualización de librerías que permiten el trabajo del módulo Wi-Fi.

Puesto que, para cumplir el objetivo de este trabajo, que era el de diseñar un dispositivo de control ON-OFF, que sea gestionado por el módulo Wi-Fi, fue necesario que el ESP8266 trabaje como un servidor WEB. Entonces se implementó un servidor WEB, en la memoria del ESP8266, el mismo que contiene la información de activar o desactivar los pines GPIO del módulo. Se aprovechó esa programación para usar esa señal con el fin de activar o desactivar un relé, que controla un foco, encendiéndolo o apagándolo.

Ya que la idea principal del trabajo era comandar el módulo Wi-Fi, por medio de un interfaz virtual residente en Android. Entonces se define usar una plataforma de desarrollo de aplicaciones para el sistema Android. Esa plataforma es APP Inventor, la que es de código abierto y de programación orientada a objetos (POO). Así se logró diseñar una presentación como interfaz de usuario que permita la gestión virtual del módulo.

El resultado de la integración de las actividades descritas dio como resultado un módulo, compacto capaz de controlar dispositivos de gestión ON-OFF por medio del ESP8266 y una interfaz de usuario atractiva y de fácil manejo.

4.2. RECOMENDACIONES

El producto final desarrollado en este proyecto, puede ser usado en varias aplicaciones de control ON-OFF, por medio de una red Wlan. Algunas de esas aplicaciones dentro de una red local son en el campo de la domótica, inmótica, automatización, entre otras. La función general del módulo desarrollado es activar o desactivar, por lo que se usaría en el control de puertas automáticas, luminarias, persianas, garajes, interruptores, dimmers, tomacorrientes, control maestro de sistemas domóticos e inmóticos.

Las aplicaciones del producto final, pueden ser variadas, por lo que si se desea extender el alcance del mismo, se podría usar la red de internet. Para eso se sugiere contratar una IP pública o contratar un hospedaje virtual para mantener el servidor activo a través de internet. Eso permitiría que la actividad del módulo no sea limitada solo a una red local.

Como parte de las aplicaciones del módulo resultante, se puede recomendar la adquisición y control de datos de forma remota. Al adicionarle algún tipo sensor como de temperatura, vibración, humo, gas, humedad, movimiento, entre otras magnitudes, se convertiría en un actuador automático de acción remota, para activar o desactivar algún tipo de sistema de regulación o control de ese tipo de magnitudes, a fin de evitar fallas o mal funcionamiento de los sistemas aplicados.

Puesto que existen aplicaciones electrónicas que requieren varios pines de gestión y control, se podría sugerir el uso de otros módulos basados en Arduino y el chip ESP8266EX. Algunos de esos módulos son el NodeMCU, WEBos, WeMOS, que usan los ESP-02 al ESP-12, los que tienen varios puertos de programación para el uso del desarrollador. El módulo usado en este proyecto es el ESP-01, que tiene la limitación de solo permitir dos GPIO disponibles, por lo que no sería factible su uso en aplicaciones con varios sensores.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] M. I. Y. Collado, La Domótica, un bien para todos, *Innovación y experiencias educativas ,* p. 16, 2010.

[2] D. R. F. Clodoaldo Robledo Sacristán, Programación en Android, Madrid, 2012.

[3] Ó. T. Artero, EL GENUINO-ARDUINO, Madrid: Prodigitalk, 2016.

[4] Espressif System, ESP8266EX Datasheet versión 6.6; 2020. Enlace: https:///0a-esp8266ex_datasheet_en.pdf, 2020.

[5] L. G. GOILAV N., ARDUINO Aprender a desarrollar para crear objetos inteligentes, ENI, 2016.

[6] Z. Nikola, "Arduino and Open Source Computer Hardware and Software", 11 2015. [En línea]. Available: https://www.researchgate.net/publication/297734853_Arduino_and_Open_Source_Computer_Hardware_and_Software . [Último acceso: 30 12 2020].

[7] Fundación Aquae, Sabes qué es Arduino y para qué sirve?, 2020. [En línea]. Available: https://www.fundacionaquae.org/sabes-arduino-sirve/ .

[8] M. Céspedes, CARACTERÍSTICAS DE LAS PLACAS ARDUINO, vol. 12, 2017.

[9] Google, Arduino, 2017. [En línea].

Available: https://sites.google.com/site/temasdedisenoymanufactura/arduino.

[10] *Midiendo el consumo de Arduino UNO.* [Película]. España: PROMETEC, 2016.

[11] *Midiendo el consumo de un Arduino UNO.* [Película]. España: Prometec, 2016.

[12] B. Leung, USB Type-C™'s Configuration Channel, 19 11 2018. [En línea]. Available: https://medium.com/@leung.benson/usb-type-c-s-configuration-channel-31e08047677d. [Último acceso: 23 07 2020].

[13] G. Staples, Arduino Power, Current, and Voltage Limitations, junio 2015. [En línea]. Available: https://www.electricrcaircraftguy.com/2014/02/arduino-power-current-and-voltage.html . [Último acceso: 30 07 2020].

[14] www.arduino.cc, 2020. [En línea].

Available: https://www.arduino.cc/reference/en/language/functions/zero-due-mkr-family/analogreadresolution/ . [Último acceso: 24 07 2020].

[15] J. A. R. Morales, Pines y conectores de Arduino UNO, 3 06 2019. [En línea].

Available: https://pasionelectronica.com/pines-y-conectores-de-arduino-uno/. [Último acceso: 2 08 2020].

[16] E. Crespo, Aprendiendo Arduino, 03 2020. [En línea].

Available: https://aprendiendoarduino.wordpress.com/category/pwm/. [Último acceso: 02 08 2020].

[17] J. M. R. Gutiérrez, Manual de Programación Arduino, 2007.

[18] Arduino, www.arduino.cc, 2020. [En línea].

Available: https://playground.arduino.cc/code/function/. [Último acceso: 10 08 2020].

[19] Arduino, www.arduino.cc, 2020. [En línea].

Available: https://playground.arduino.cc/en/Reference/Main/GeneralCodeLibrary/. [Último acceso: 10 08 2020].

[20] Arduino, playground.arduino.cc, 2020. [En línea].

Available: https://playground.arduino.cc/en/Hacking/Library/. [Último acceso: 11 08 2020].

[21] J. Crespo, ZigBee/XBee, 16 11 2016. [En línea].

Available: https://aprendiendoarduino.wordpress.com/2016/11/16/zigbeexbee/. [Último acceso: 11 08 2020].

[22] L. D. V. Hernández, ESP8266 todo lo que necesitas saber del módulo WiFi para Arduino, 2016. [En línea]. Available: https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/. [Último acceso: 14 09 2020].

[23] WatchGuard;, Modos y Canales Inalámbricos, 2018. [En línea].

Available: https://www.watchguard.com/help/docs/fireware/12/es-419/Content/es-419/wireless/ap_about_wireless_modes_c.html. [Último acceso: 16 09 2020].

[24] L. Llamas, Detalles de hardware y pins del ESP8266, 05 2018. [En línea].

Available: https://www.luisllamas.es/detalles-del-esp8266-diferencias-con-arduino/. [Último acceso: 18 09 2020].

[25] Winbonhd, w25q40bw revf 101113, octubre 2013.

[26] Winbond, w25q80bw revb 101113, marzo 2015.

[27] R. Santos, The Internet of things with ESP8266, 2018. [En línea].

Available: http://esp8266.net/. [Último acceso: 25 09 2020].

[28] P. Panda, Designing a WiFi PCB trace antenna for ESP8266 or ESP32, 07 2017. [En línea]. Available: http://iot-bits.com/wifi-pcb-trace-antenna-esp32-esp8266/. [Último acceso: 24 09 2020].

[29] A. Salcin, ESP8266 HARDWARE, Sarajevo, Bosnia and Herzegovina, 2019.

[30] Arduino, github, 2020. [En línea]. Available: https://github.com/esp8266/at-command-set/commit/3308899a70f5c7139803ca7806778c5b4a0c771a. [Último acceso: 02 10 2020].

[31] Diario La Nación, Los acuerdos entre Google y los fabricantes de smartphones, bajo la lupa europea, 10 2014. [En línea]. Available: https://www.lanacion.com.ar/tecnologia/la-union-europea-quiere-saber-mas-sobre-los-acuerdos-de-google-y-los-fabricantes-de-smartphone. [Último acceso: 2 10 2020].

[32] S. Alvarez, Manual de iniciación a la programación, 05 2006. [En línea].

Available: https://desarrolloweb.com/articulos/2477.php. [Último acceso: 8 10 2020].

[33] CYStech Electronics Corp, LM1117L3, 2005.

[34] Espress System, 2020. [En línea].

Available: https://www.espressif.com/en/support/download/at. [Último acceso: 24 11 2020].

[35] Sigma, Sigma Shop, [En línea].

Available: https://sigma-shop.com/userfiles/productlargeimages/product_148.jpg. [Último acceso: 23 11 2020].

[36] Arduino, www.arduino.cc, 2020. [En línea]. [Último acceso: 25 12 2020].

[37] V. Venhtura, Polaridad.es, 6 10 2016. [En línea]. Available: https://polaridad.es/modulo-wifi-esp8266-arduino-operacion-basica-conexion/.

[38] Quectel, AT Commands Set, 2002.

ANEXO A

COMANDOS AT

Comandos AT usados en comunicaciones y sus respuestas [38]

Comando para **probar**	AT+<x>=?	Este comando retorna la lista de parámetros y rangos de valores establecidos con el comando correspondiente para escribir o por procesos internos.
Comando para **leer**	AT+<x>?	Este comando retorna el valor establecido actual del parámetro o parámetros.
Comando para **escribir**	AT+<x>=<…>	Este comando establece los parámetros que se pueden definir por el usuario.
Comando para **ejecutar**	AT+<x>	Este comando lee parámetros que no se pueden modificar, afectados únicamente por el dispositivo.

Comando	**Descripción**
AT&F	Reestablecer los parámetros de fábrica
AT&V	Mostrar la configuración actual
AT&W	Guardar los parámetros establecidos en el perfil del usuario
AT+GMI	Solicitar la información del fabricante
AT+GSN	Solicitar el número identificador IMEI del dispositivo (International Mobile Equipment Identity)
AT+GMM	Obtener el modelo del dispositivo
AT+GMR	Obtener la versión del firmware del dispositivo
A/	Repite el último comando
ATA	Responde la llamada entrante
ATD><N>	Llama al número guardado en memoria
ATDL	Llama el último teléfono marcado
ATH	Se desconecta de la conexión actual
ATL	Establecer el volumen de la bocina monitor
ATT	Establecer la llamada de pulsos

Comandos AT más comunes para módulos inalámbricos

Comando	**Descripción**
AT	Comando de prueba, así puedes verificar si la comunicación es bidireccional, el módulo responderá con un «OK»
AT+STATE?	El módulo responderá con el estado actual
AT+RESET	Comando para reiniciar el módulo
AT+VERSION?	Comando para obtener la versión del firmware del módulo

AT+ORGL	Reestablece el módulo a valores de fábrica
AT+ADDR?	Dirección física alfanumérica del módulo
AT+NAME=<N>	Establece el nombre bluetooth del módulo, usa AT+NAME? para conocer el nombre del módulo actual
AT+ROLE=<N>	Establece el rol de operación, usa 0 para definir modo esclavo, 1 para modo maestro y 2 para modo esclavo-repetido; en este modo el módulo reenvía todo lo que recibe de la conexión bluetooth
A+PSWD=<N>	Establece el PIN numérico de 4 dígitos para el emparejamiento bluetooth, el predeterminado es 1234, para conocer el pin actual usa AT+PSWD?
AT+UART=<N1>,<N2>,<N3>	Establece los parámetros de la comunicación serial, N1 es el número de baudios a utilizar en la conexión, N2 es el bit de paro a utilizar en la comunicación y N3 es la paridad de la conexión, para conocer los valores definidos usa AT+UART?
AT+IRQ	Escanea y muestra la dirección física los dispositivos cercanos
AT+BIND=<N>	Establece la dirección física bluetooth a la cual el módulo se conectará al encenderse
AT+CMODE=<N>	Usa 0 para conectar a la dirección física definida con AT+BIND previamente, 1 para conectarse a cualquier dirección dentro de la cobertura del módulo o 2 para modo esclavo-repetido

La información más detallada y total se encuentra en la guía de la bibliografía, de donde se obtiene esta.

ANEXO B

PROGRAMA DEL SERVIDOR ESP8266

```
// cargar librería Wi-Fi
#include <ESP8266WiFi.h>

// Deteccion de red
const char* ssid     = "SSDI de red WIFI";
const char* password = "passworbd";

// Configuración de la IP estática.
   IPAddress local_IP(192, 168, 100, 12);
   IPAddress gateway(192, 168, 100, 1);
   IPAddress subnet(255, 255, 255, 0);

// Activar el puerto del Servidor web
WiFiServer server(80);

// Variable para respuesta HTTP
String cabecera;

// Variables Auxiliares  para el estado de la salida
String output0State = "off";
String output2State = "off";

// Asignacion de las variables de las salidas GPIO
const int output0 = 0;
const int output2 = 2;

// tiempo de ACK
unsigned long currentTime = millis();
unsigned long previousTime = 0;

// Define de desconexion 2000ms = 2s)
const long timeoutTime = 2000;
```

```
void setup() {
  // velocidad de transmision
  Serial.begin(115200);

 // Inicializacion de las variables de las salidas
  pinMode(output5, OUTPUT);
  pinMode(output2, OUTPUT);

// Poner las salidas en bajo
  digitalWrite(output0, LOW);
  digitalWrite(output2, LOW);

  // Conectar a la red con los parámetros de identificación
  Serial.print("Conectando a");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
          {
          delay(500);
          Serial.print(".");
          }
  // Se imprime la direccionh IP local y activa el Servidor WEB
  Serial.println("");
  Serial.println("WiFi conectado.");
  Serial.println("Dirección IP: ");
  Serial.println(local_IP());
  server.begin();
}

void loop(){
  WiFiClient client = server.available();

// Escucha llamados de clientes
    if (client) {                             // Si hay un nuevo cliente conecte,
    Serial.println("Nuevo Cliente.");   // mensaje al puerto serial
    String currentLine = "";                // realizar una cadena de datos del cliente
```

```
currentTime = millis();
previousTime = currentTime;
while (client.connected() && currentTime - previousTime <= timeoutTime) {
  // lazo while del cliente conectado
  currentTime = millis();
  if (client.available()) {             // si hay bytes para leer del cliente,
    char c = client.read();             // leer un byte entonces
    Serial.write(c);                    // imprimir el resultado al monitor serial
    cabecera += c;
    if (c == '\n') {                    // si el byte es una nueva linea
      // si es una línea en blanco se añaden dos caracteres a la fila
      // al fin el fin de la respuesta del cliente y respuesta:
      if (currentLine.length() == 0) {

// HTTP cabeceras siempre empieza con el código (e.g. HTTP/1.1 200 OK)
// y content-type para el cliente se conoce que llama una línea en blanco:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Conexion: Cerrado");
            client.println();

            // activación de GPIOs on - off
            if (cabecera.indexOf("GET /0/on") >= 0) {
              Serial.println("GPIO 0 on");
              output5State = "on";
              digitalWrite(output0, HIGH);
            } else if (cabecera.indexOf("GET /5/off") >= 0) {
              Serial.println("GPIO 0 off");
              output5State = "off";
              digitalWrite(output0, LOW);
            } else if (cabecera.indexOf("GET /2/on") >= 0) {
              Serial.println("GPIO 2 on");
              output4State = "on";
              digitalWrite(output2, HIGH);
            } else if (cabecera.indexOf("GET /4/off") >= 0) {
              Serial.println("GPIO 2 off");
```

```
            output4State = "off";
            digitalWrite(output2, LOW);
          }
           // Codigo para la pagina HTML
          client.println("<!DOCTYPE html><html>");
          client.println("<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1\">");
          client.println("<link rel=\"icon\" href=\"data:,\">");
          // Botones estilo CSS  on/off
          // cambio dde color y fondo en la activacion
          client.println("<style>html { font-family: Helvetica; display: inline-block; margin: 0px
auto; text-align: center;}");
          client.println(".button { background-color: #195B6A; border: none; color: white;
padding: 16px 40px;");
          client.println("text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}");
          client.println(".button2 {background-color: #77878A;}</style></head>");

          // cabeceras
          client.println("<body><h1>ESP8266 Web Servidor</h1>");

          // estado y activacion  ON/OFF de los botones GPIO
          client.println("<p>GPIO 0 - Estado" + output0State + "</p>");

           // si el output0State es off, se activa el boton ON
          if (output5State=="off") {
            client.println("<p><ahref=\"/0/on\"><button
class=\"button\">ON</button></a></p>");
          } else {
            client.println("<p><a href=\"/0/off\"><button class=\"button
button2\">OFF</button></a></p>");
          }

          client.println("</body></html>");
           // The HTTP responde al final con una linea en blanco
          client.println();
          // Break de salida del lazo  while
```

```
          break;
        } else { // si se recibe una nueva línea, entonces limpia la anterior
          currentLine = "";
        }
      } else if (c != '\r') {  // si no se recibe nada entonces retorna un carácter al carrier,
        currentLine += c;      // se añade al fin un carácter a la línea
      }
    }
  }
  // limpiar la cabecera
  cabecera = "";
  // Cerrar la conexion
  client.stop();
  Serial.println("Cliente desconectado.");
  Serial.println("");
 }
}
```

ORDEN DE EMPASTADO