INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

MAESTRÍA EN INGENIERÍA ELECTRÓNICA

TESIS

TEMA:

MONITOREO INALÁMBRICO DE CONSUMO ELÉCTRICO CON SENSOR NO INVASIVO Y MICROCONTROLADOR ARDUINO

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN INGENIERÍA ELECTRÓNICA

PRESENTA:

ING. CÉSAR NICOLÁS BRAVO DÍAZ

N.C. M16290012

ASESOR:

M.C. GUSTAVO OCHOA MATA

CIUDAD GUZMÁN, JALISCO, MÉXICO. AGOSTO DE 2018

Cd. Guzmán, Jal. a 20/Agosto/2018

OFICIO No. S/N

ASUNTO : AUTORIZACIÓN DE IMPRESIÓN

ING. CESAR NICOLAS BRAVO DIAZ
M16290012

En cumplimiento con el documento normativo de las disposiciones para la operación de estudios de posgrado del Tecnológico Nacional de México y con base en la aprobación del Comité Tutorial comisionado para la revisión, la División de Estudios de Posgrado e Investigación le otorga la autorización de impresión de su trabajo de tesis intitulado:

"**MONITOREO INALÁMBRICO DE CONSUMO ELECTRICO CON SENSOR NO INVASIVO Y MICROCONTROLADOR ARDUINO**", dirigido por el M. C, GUSTAVO OCHOA MATA, desarrollado como requisito parcial para la obtención del grado de Maestro en Ingeniería Electrónica, de acuerdo al plan de estudios MPIEO-2011-13.

Sin otro asunto en particular, quedo de usted.

S.E.P. TecNM
INSTITUTO TECNOLOGICO
DE CD. GUZMAN
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION

A T E N T A M E N T E

DR. HUMBERTO BRACAMONTES DEL TORO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.p. Archivo

Dedicatoria

A mi Padre, Nicolás Bravo Sánchez por ser el mayor ejemplo durante toda mi vida y dar los mejores consejos para salir adelante.

A mi Madre, Alicia Díaz Torrez por su apoyo incondicional y su constante preocupación por mi bienestar para cumplir mis metas.

Agradecimientos

Al Consejo Nacional de Ciencia y Tecnología por el beneficio de la beca durante el posgrado.

A mi familia y amigos que siempre me apoyaron y motivaron en los estudios.

A mi asesor el M.C. Gustavo Ochoa Mata por su atención, apoyo y paciencia durante la maestría.

Al Dr. Jesús Alberto Verduzco Ramírez por aconsejarme y motivarme a hacer un posgrado.

A los profesores de la División de Estudios de Posgrado de Electrónica del Instituto Tecnológico de Ciudad Guzmán.

A Dios por permite llegar y terminar esta etapa de mi vida.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

“MONITOREO INALÁMBRICO DE CONSUMO ELÉCTRICO CON SENSOR NO INVASIVO Y MICROCONTROLADOR”

RESUMEN

Con la constante innovación y aplicación de tecnologías como el internet de las cosas en el hogar, surgen áreas de estudio como la domótica, que nos permite tener un control sobre los diferentes sistemas de nuestro hogar, control de luz, calefacción, sistemas de riego, control automático de puertas, siendo capaces de monitorear todas estas variables desde nuestros dispositivos móviles.

En el presente proyecto de tesis se presenta la elaboración e implementación de un prototipo de medición de consumo eléctrico usando un sensor no invasivo capaz de medir la corriente por medio de inducción electromagnética evitando así la modificación de la instalación original. Para la interpretación del sensor se cuenta con un microcontrolador Arduino UNO y para el envió de estos datos a una red local se utiliza el módulo WIFI ESP8266.

Ofreciendo de esta manera otra alternativa a los diferentes sistemas de medición de consumo eléctrico y ampliando el desarrollo en el área de la domótica a un menor costo.

Palabras Clave:

Domótica, Consumo Eléctrico, ESP8266, SCT-013-030.

“WIRELESS MONITORING OF ELECTRICAL CONSUMPTION WITH NON-INVASIVE SENSOR AND MICROCONTROLLER”

ABSTRACT

With the constant innovation and application of technologies such as the internet of things in the home, areas of study such as home automation arise, which allows us to have control over the different systems of our home, light control, heating, irrigation systems, automatic control of doors, being able to monitor all these variables from our mobile devices.

The present thesis project presents the elaboration and implementation of a prototype of measurement of electrical consumption using a non-invasive sensor capable of measuring the current by means of electromagnetic induction, thus avoiding the modification of the original installation. For the interpretation of the sensor there is an Arduino UNO microcontroller and for sending this data to a local network the WIFI module ESP8266 is used.

Offering in this way another alternative to the different systems of measurement of electrical consumption and expanding the development in the area of home automation at a lower cost.

Keywords

Home Automation, Electrical Consumption, ESP8266, SCT-013-030.

CAPÍTULO 1
INTRODUCCIÓN

1.1 Presentación

Actualmente las tendencias en ahorro de energía han mostrado ser la forma más efectiva de constituir un ahorro significativo tanto en el hogar como en la industria, con el fin de hacer más eficiente un proceso industrial o simplemente monitorear el desempeño de un enser doméstico, que quizás posteriormente pueda ser sustituido y/o nos ayude a saber cuánto nos cuesta mantener cierto electrodoméstico.

La energía eléctrica es un recurso indispensable en el hogar común por lo que ayudado por lo que es el internet de las cosas surge la idea de automatizar, monitorizar y controlar estos recursos de manera remota, con dispositivos que usemos a diario y de manera inteligente.

El proyecto surge de la necesidad de tener un conocimiento sobre nuestro consumo eléctrico en tiempo real dado que los proporcionados por la Comisión Federal de Electricidad (CFE) en México proporcionan un conocimiento básico y sin manera de hacerlo remotamente.

Proponiendo una alternativa a esto y a la variedad de medidores de consumo eléctrico que hay en el mercado se realizó un medidor de consumo eléctrico usando un sensor no invasivo lo cual nos permite la medición del consumo sin necesidad de modificación o alteración del cableado eléctrico, enviando los datos inalámbricamente a través de nuestra red local para su consulta en cualquiera de nuestros dispositivos móviles con conexión a internet.

1.2 Objetivo general

Diseñar e implementar un sistema de monitoreo inalámbrico de consumo eléctrico que nos permita conocer los datos en red local.

1.3 Objetivos específicos

- ✓ Comunicar el microcontrolador por medio de WIFI con el módulo ESP8266.
- ✓ Implementar y Calibrar el sensor SCT 013 para la red eléctrica local.
- ✓ Enviar la medición del consumo eléctrico por WIFI.
- ✓ Reducir el costo del desarrollo del prototipo.
- ✓ Visualizar las mediciones en cualquier dispositivo móvil con conexión a la red local.
- ✓ Análisis y comparación de resultados.

1.4 Metas

Los resultados que se esperan obtener para el proyecto son los siguientes:

- ✓ Obtener el menor costo del desarrollo del proyecto.
- ✓ Lograr una medición verídica del consumo eléctrico.
- ✓ Enviar los datos del consumo eléctrico por WIFI.
- ✓ Visualizar el consumo eléctrico con cualquier dispositivo móvil con acceso a la red local.
- ✓ Lograr un prototipo funcional.

1.5 Impacto o beneficio en la solución a un problema relacionado con el sector productivo o la generación del conocimiento científico o tecnológico.

Se espera que el impacto que tenga el proyecto beneficie a los hogares domóticos, proporcionando una manera remota y fácil de saber el consumo que se tiene sobre el hogar ayudando así a que el propietario aplique las medidas necesarias para hacer un ahorra significativo en su bolsillo y el medio ambiente.

El prototipo realizado servirá de ejemplo para aquellas personas que deseen seguir desarrollándolo y lograr un producto superior mejorando aquellos aspectos que al final de esta tesis se marcaran como trabajos futuros

CAPÍTULO 2
MARCO TEÓRICO

2.1 Sensor SCT-013

La familia SCT-013 son sensores de corrientes no invasivos que permiten medir la intensidad que atraviesa un conductor sin necesidad de cortar o modificar el conductor. Se puede emplear estos sensores con un procesador como Arduino para medir la intensidad o potencia consumida por una carga.

Los sensores SCT-013 son transformadores de corriente, dispositivos de instrumentación que proporcionan una medición proporcional a la intensidad que atraviesa un circuito. La medición se realiza por inducción electromagnética. Disponen de un núcleo ferromagnético partido (como une pinza) que permite abrirlo para arrollar un conductor de una instalación eléctrica sin necesidad de cortarlo.

Dentro de la familia SCT-013 existen modelos que proporcionan la medición como una salida de intensidad o de tensión.

La precisión del sensor puede ser de 1-2%, pero para ello es muy importante que el núcleo ferromagnético se cierre adecuadamente. Hasta un pequeño hueco de aire puede introducir desviaciones del 10%.

Como desventaja, al ser una carga inductiva, el SCT-013 introduce una variación del ángulo de fase cuyo valor es función de la carga que lo atraviesa, pudiendo llegar a ser de hasta 3º.

Los transformadores de intensidad son componentes frecuentes en el mundo industrial y en distribución eléctrica, permitiendo monitorizar el consumo de puntos de consumo donde sería imposible otro tipo de medición. También forman parte de múltiples instrumentos de medición, incluso en equipos portátiles como en pinzas perimétricas o analizadores de red.

Podemos comprar diferentes tipos de sensores de corriente alterna SCT-013 que se pueden organizar en dos grupos. Los que proporcionan una corriente o los que proporcionan un voltaje. La gran diferencia entre ellos es que en los primeros no viene incluida una resistencia de carga y en los segundos sí.

El SCT-013-000 es el único que nos proporciona una corriente y no tiene resistencia de carga. Puede medir una corriente de entre 50 mA y 100 A**.**

Figura 2.1: Sensor SCT-013-030.

El resto de la familia de sensores SCT-013 sí que tienen incluida la resistencia de carga. Se pueden encontrar varios modelos, pero todos tienen un voltaje de salida entre 0V y 1V.

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

Figura 2.2: Tipos de Sensor SCT-013.

2.1.1 Funcionamiento del SCT 013

La familia de sensores SCT-013 son pequeños transformadores de corriente que son instrumentos ampliamente empleados como elementos de medición.

Un transformador de corriente es similar a un transformador de tensión y está basado en los mismos principios de funcionamiento. Sin embargo, persiguen objetivos diferentes y, en consecuencia, están diseñados y construidos de forma distinta.

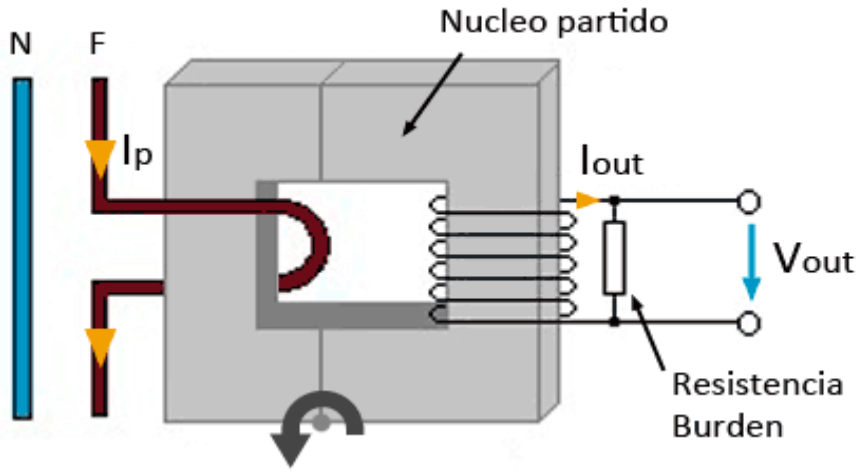

Figura 2.3: Funcionamiento del sensor SCT-013.

Un transformador de corriente busca generar una intensidad en el secundario que sea proporcional a la intensidad que atraviesa el primario. Para ello se desea que el primario esté formado un número de espiras reducido.

Podemos emplear esto para construir un sensor de corriente no invasivo. En un sensor se corriente el núcleo ferromagnético puede estar dividido de forma que pueda abrirse y arrollar un conductor.

De esta forma, se tenemos un transformador en el que:

- El cable por el que circula la corriente es el devanado primario
- La “pinza” es el núcleo magnético
- El devanado secundario está integrado como parte del transformador.

Cuando la corriente alterna circula por el conductor se genera un flujo magnético en el núcleo ferromagnético, que a su vez genera una corriente eléctrica en el devanado secundario.

El primario generalmente está formado por una única espira formada por el conductor a medir. Aunque es posible enrollar el conductor haciendo que pase más de una vez por el interior de la “pinza”. El número de espiras del secundario, integrado en el transformador, varía 1000-2000 según modelos del SCT-013.

A diferencia de los transformadores de tensión, en un transformador de intensidad el circuito secundario nunca debería estar abierto, porque las corrientes inducidas podrían llegar a dañar el componente. Por ese motivo, los sensores de SCT-130 disponen de protecciones (resistencia burden en los sensores de salida por tensión, o diodos de protección en los sensores de salida por corriente).

2.1.2 Efecto Hall

El efecto Hall se produce cuando se ejerce un campo magnético transversal sobre un cable por el que circulan cargas. Como la fuerza magnética ejercida sobre ellas es perpendicular al campo magnético y a su velocidad (ley de la fuerza de Lorentz), las cargas son impulsadas hacia un lado del conductor y se genera en él un voltaje transversal o voltaje Hall (V_H). Edwin Hall (1835 - 1938) descubrió en 1879 el efecto, que, entre otras muchas aplicaciones, contribuyó a establecer, diez

años antes del descubrimiento del electrón, el hecho de que las partículas circulan por un conductor metálico tiene carga negativa.

En la siguiente imagen se muestra un dispositivo experimental destinado a medir el voltaje Hall. Sobre una corriente eléctrica actúa un imán que produce un campo magnético (B). La fuerza magnética (F_m) desvía a las cargas móviles hacia uno de los lados del cable, lo que implica que dicho lado queda con carga de ese signo y el opuesto queda con carga del signo contrario. En consecuencia, entre ambos se establece un campo eléctrico y su correspondiente diferencia de potencial o voltaje Hall.

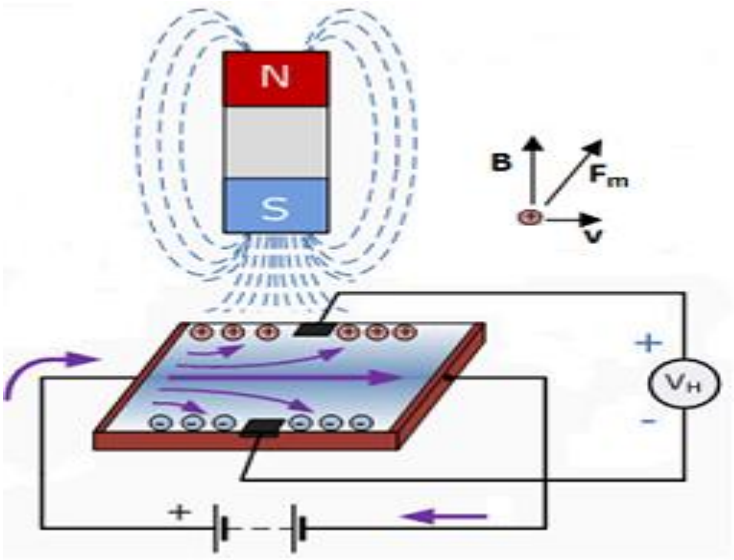

Figura 2.4: Dispositivo experimental para medir voltaje Hall.

La obtención experimental del voltaje Hall, permite deducir la velocidad de los portadores de carga y su concentración, puesto que, desde que se alcanza la situación estacionaria, la fuerza eléctrica ejercida sobre cada carga ($F_e = q \cdot E$) se equilibra con la fuerza magnética [$F_m = q \cdot (v \times B)$]. De ello se deduce que el voltaje Hall es directamente proporcional a la corriente eléctrica y al campo magnético y es inversamente proporcional al número de portadores por unidad de volumen. Por lo

tanto, con un sensor de efecto Hall, se puede determinar la fuerza que ejerce un campo magnético si se conoce la corriente a la que se aplica dicho campo, y viceversa.

2.2 Arduino

Arduino es en realidad tres cosas:

Una placa hardware libre que incorpora un microcontrolador reprogramable y una serie de pines-hembra (los cuales están unidos internamente a las patillas de E/S del microcontrolador) que permiten conectar allí de forma muy sencilla y cómoda diferentes sensores y actuadores.

Cuando se habla de “placa hardware” nos estamos refiriendo en concreto a una PCB (del inglés “printed circuit board”, o sea, placa de circuito impreso). Las PCBs son superficies fabricadas de un material no conductor (normalmente resinas de fibra de vidrio reforzada, cerámica o plástico) sobre las cuales aparecen laminadas (“pegadas”) pistas de material conductor (normalmente cobre). Las PCBs se utilizan para conectar eléctricamente, a través de los caminos conductores, diferentes componentes electrónicos soldados a ella. Una PCB es la forma más compacta y estable de construir un circuito electrónico (en contraposición a una breadboard, perfboard o similar) pero, al contrario que estas, una vez fabricada, su diseño es bastante difícil de modificar. Así pues, la placa Arduino no es más que una PCB que implementa un determinado diseño de circuitería interna.

No obstante, cuando se habla de “placa Arduino”, debe especificarse el modelo concreto, ya que existen varias placas Arduino oficiales, cada una con diferentes características (como el tamaño físico, el número de pines-hembra ofrecidos, el modelo de microcontrolador incorporado –y como consecuencia, entre otras cosas, la cantidad de memoria utilizable–, etc.). Conviene conocer estas características para identificar qué placa Arduino es la que convendrá más en cada proyecto.

De todas formas, aunque puedan ser modelos específicos diferentes los microcontroladores incorporados en las diferentes placas Arduino pertenecen todos a la misma “familia tecnológica”, por lo que su funcionamiento en realidad es bastante parecido entre sí. En concreto, todos los microcontroladores son de tipo AVR, una arquitectura de microcontroladores desarrollada y fabricada por la marca Atmel.

El diseño hardware de la placa Arduino está inspirado originalmente en el de otra placa de hardware libre preexistente, la placa Wiring. Esta placa surgió en 2003 como proyecto personal de Hernando Barragán, estudiante por aquel entonces del Instituto de Diseño de Ivrea (lugar donde surgió en 2005 precisamente la placa Arduino).

Un software (más en concreto, un “entorno de desarrollo”) **gratis, libre y multiplataforma** (ya que funciona en Linux, MacOS y Windows) que se debe instalar en el ordenador y que permitirá escribir, verificar y guardar (“cargar”) en la memoria del microcontrolador de la placa Arduino el conjunto de instrucciones que se desea que este empiece a ejecutar. Es decir: permite su programación. La manera estándar de conectar el computador con la placa Arduino para poder enviarle y grabarle dichas instrucciones es mediante un simple cable USB, gracias a que la mayoría de las placas Arduino incorporan un conector de este tipo.

Los proyectos Arduino pueden ser autónomos o no. En el primer caso, una vez programado su microcontrolador, la placa no necesita estar conectada a ningún computador y puede funcionar autónomamente si dispone de alguna fuente de alimentación. En el segundo caso, la placa debe estar conectada de alguna forma permanente (por cable USB, por cable de red Ethernet, etc.) a un computador ejecutando algún software específico que permita la comunicación entre este y la placa y el intercambio de datos entre ambos dispositivos. Este software específico se deberá programar generalmente por el usuario mediante algún lenguaje de programación estándar como Python, C, Java, Php, etc., y será independiente completamente del entorno de desarrollo Arduino, el cual no se necesitará más, una vez que la placa ya haya sido programada y esté en funcionamiento.

Un lenguaje de programación libre. Por “lenguaje de programación” se entiende cualquier idioma artificial diseñado para expresar instrucciones (siguiendo unas determinadas reglas sintácticas) que pueden ser llevadas a cabo por máquinas. Concretamente dentro del lenguaje Arduino, se encuentran elementos parecidos a muchos otros lenguajes de programación existentes (como los bloques condicionales, los bloques repetitivos, las variables, etc.), así como también diferentes comandos –asimismo llamados “órdenes” o “funciones” – que permiten especificar de una forma coherente y sin errores las instrucciones exactas que se quieren programar en el microcontrolador de la placa. Estos comandos se escriben mediante el entorno de desarrollo Arduino.

Tanto el entorno de desarrollo como el lenguaje de programación Arduino están inspirado en otro entorno y lenguaje libre preexistente: Processing, desarrollado inicialmente por Ben Fry y Casey Reas. Que el software Arduino se parezca tanto a Processing no es casualidad, ya que este está especializado en facilitar la generación de imágenes en tiempo real, de animaciones y de interacciones visuales, por lo que muchos profesores del Instituto de Diseño de Ivrea lo utilizaban en sus clases. Como fue en ese centro donde precisamente se inventó Arduino es natural que ambos entornos y lenguajes guarden bastante similitud. No obstante, hay que aclarar que el lenguaje Processing está construido internamente con código escrito en lenguaje Java, mientras que el lenguaje Arduino se basa internamente en código C/C++.

Con Arduino se pueden realizar multitud de proyectos de rango muy variado: desde robótica hasta domótica, pasando por monitorización de sensores ambientales, sistemas de navegación, telemática, etc. Realmente, las posibilidades de esta plataforma para el desarrollo de productos electrónicos son prácticamente infinitas y tan solo están limitadas por la imaginación.

2.2.1 ¿Qué es un microcontrolador?

Un microcontrolador es un circuito integrado o “chip” (es decir, un dispositivo electrónico que integra en un solo encapsulado un gran número de componentes) que tiene la característica de ser programable. Es decir, que es capaz de ejecutar de forma autónoma una serie de instrucciones previamente definidas por el usuario.

Por definición, un microcontrolador (también llamado comúnmente “micro”) ha de incluir en su interior tres elementos básicos:

- **CPU (Unidad Central de Proceso):** es la parte encargada de ejecutar cada instrucción y de controlar que dicha ejecución se realice correctamente. Normalmente, estas instrucciones hacen uso de datos disponibles previamente (los “datos de entrada”), y generan como resultado otros datos diferentes (los “datos de salida”), que podrán ser utilizados (o no) por la siguiente instrucción.

- **Diferentes tipos de memorias:** son en general las encargadas de alojar tanto las instrucciones como los diferentes datos que estas necesitan. De esta manera posibilitan que toda esta información (instrucciones y datos) esté siempre disponible para que la CPU pueda acceder y trabajar con ella en cualquier momento. Generalmente encontraremos dos tipos de memorias: las que su contenido se almacena de forma permanente incluso tras cortes de alimentación eléctrica (llamadas “persistentes”), y las que su contenido se pierde al dejar de recibir alimentación (llamadas “volátiles”). Según las características de la información a guardar, esta se grabará en un tipo u otro de memoria de forma automática, habitualmente.

- **Diferentes pines de E/S (entrada/salida):** son las encargadas de comunicar el microcontrolador con el exterior. En los pines de entrada del microcontrolador podremos conectar sensores para que este pueda recibir

datos provenientes de su entorno, y en sus pines de salida se puede conectar actuadores para que el microcontrolador pueda enviarles órdenes y así interactuar con el medio físico. De todas formas, muchos de los pines de la mayoría de los microcontroladores no son exclusivamente de entrada o de salida, sino que pueden ser utilizados indistintamente para ambos propósitos (de ahí el nombre de E/S).

Es decir, un microcontrolador es un computador completo (aunque con prestaciones limitadas) en un solo chip, el cual está especializado en ejecutar constantemente un conjunto de instrucciones predefinidas. Estas instrucciones irán teniendo en cuenta en cada momento la información obtenida y enviada por las patillas de E/S y reaccionarán en consecuencia. Lógicamente, las instrucciones serán diferentes según el uso que se le quiera dar al microcontrolador, y deberemos de decidir nosotros cuáles son.

Cada vez existen más productos domésticos que incorporan algún tipo de microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y coste, mejorar su fiabilidad y disminuir el consumo. Así, es posible encontrar microcontroladores dentro de multitud de dispositivos electrónicos que se usan en la vida diaria, como pueden ser desde un simple timbre hasta un completo robot pasando por juguetes, frigoríficos, televisores, lavadoras, microondas, impresoras, el sistema de arranque de nuestro coche, etc.

2.2.2 ¿Qué quiere decir que Arduino sea software libre?

Según la Free Software Foundation, organización encargada de fomentar el uso y desarrollo del software libre a nivel mundial, un software para ser considerado libre ha de ofrecer a cualquier persona u organización cuatro libertades básicas e imprescindibles:

- **Libertad 0**: La libertad de usar el programa con cualquier propósito y en cualquier sistema informático.
- **Libertad 1**: La libertad de estudiar cómo funciona internamente el programa, y adaptarlo a las necesidades particulares. El acceso al código fuente es un requisito previo para esto.
- **Libertad 2**: La libertad de distribuir copias.
- **Libertad 3**: La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. El acceso al código fuente es un requisito previo para esto.

Un programa es software libre si los usuarios tienen todas estas libertades. Así pues, el software libre es aquel software que da a los usuarios la libertad de poder ejecutarlo, copiarlo y distribuirlo (a cualquiera y a cualquier lugar), estudiarlo, cambiarlo y mejorarlo, sin tener que pedir ni pagar permisos al desarrollador original ni a ninguna otra entidad específica. La distribución de las copias puede ser con o sin modificaciones propias, y atención, puede ser gratis ¡o no!: el "software libre" es un asunto de libertad, no de precio.

Para que un programa sea considerado libre a efectos legales ha de someterse a algún tipo de licencia de distribución, entre las cuales se encuentran la licencia GPL (General Public License), o la LGPL, entre otras.

El software Arduino es software libre porque se publica con una combinación de la licencia GPL (para el entorno visual de programación propiamente dicho) y la licencia LGPL (para los códigos fuente de gestión y control del microcontrolador a nivel más interno). La consecuencia de esto es, en pocas palabras, que cualquier persona que quiera (y sepa), puede formar parte del desarrollo del software Arduino y contribuir así a mejorar dicho software, aportando nuevas características, sugiriendo ideas de nuevas funcionalidades, compartiendo soluciones a posibles errores existentes, etc. Esta manera de funcionar provoca la creación espontánea de una comunidad de personas que colaboran mutuamente a través de Internet, y

consigue que el software Arduino evolucione según lo que la propia comunidad decida. Esto va mucho más allá de la simple cuestión de si el software Arduino es gratis o no, porque el usuario deja de ser un sujeto pasivo para pasar a ser (si quiere) un sujeto activo y partícipe del proyecto.

2.2.3 ¿Qué quiere decir que Arduino sea hardware libre?

El hardware libre (también llamado “open-source” o “de fuente abierta”) comparte muchos de los principios y metodologías del software libre. En particular, el hardware libre permite que la gente pueda estudiarlo para entender su funcionamiento, modificarlo, reutilizarlo, mejorarlo y compartir dichos cambios. Para conseguir esto, la comunidad ha de poder tener acceso a los ficheros esquemáticos del diseño del hardware en cuestión (que son ficheros de tipo CAD). Estos ficheros detallan toda la información necesaria para que cualquier persona con los materiales, herramientas y conocimientos adecuados pueda reconstruir dicho hardware por su cuenta sin problemas, ya que consultando estos ficheros se puede conocer qué componentes individuales integran el hardware y qué interconexiones existen entre cada uno de ellos.

La placa Arduino es hardware libre porque sus ficheros esquemáticos están disponibles para descargar de la página web del proyecto con la licencia Creative Commons Attribution Share-Alike, la cual es una licencia libre que permite realizar trabajos derivados tanto personales como comerciales (siempre que estos den crédito a Arduino y publiquen sus diseños bajo la misma licencia). Así pues, uno mismo se puede construir su propia placa Arduino “a mano”. No obstante, lo más normal es comprarlas de un distribuidor ya preensambladas y listas para usar; en ese caso, lógicamente, la placa Arduino, aunque sea libre, no puede ser gratuita, ya que es un objeto físico y su fabricación cuesta dinero.

A diferencia del mundo del software libre, donde el ecosistema de licencias libres es muy rico y variado, en el ámbito del hardware todavía no existen prácticamente licencias específicamente de hardware libre, ya que el concepto de

“hardware libre” es relativamente nuevo. De hecho, hasta hace poco no existía un consenso generalizado en su definición. Para empezar a remediar esta situación, en el año 2010 surgió el proyecto OSHD, el cual pretende establecer una colección de principios que ayuden a identificar como “hardware libre” un producto físico. OSHD no es una licencia (es decir, un contrato legal), sino una declaración de intenciones (es decir, una lista general de normas y de características) aplicable a cualquier artefacto físico para que pueda ser considerado libre. El objetivo de la OSHD (en cuya redacción ha participado gente relacionada con el proyecto Arduino, entre otros) es ofrecer un marco de referencia donde se respete por un lado la libertad de los creadores para controlar su propia tecnología y al mismo tiempo se establezcan los mecanismos adecuados para compartir el conocimiento y fomentar el comercio a través del intercambio abierto de diseños. En otras palabras: mostrar que puede existir una alternativa a las patentes de hardware que tampoco sea necesariamente el dominio público. El proyecto OSHD abre, pues, un camino que crea precedentes legales para facilitar el siguiente paso lógico del proceso: la creación de licencias libres de hardware.

El objetivo del hardware libre es, por lo tanto, facilitar y acercar la electrónica, la robótica y en definitiva la tecnología actual a la gente, no de una manera pasiva, meramente consumista, sino de manera activa, involucrando al usuario final para que entienda y obtenga más valor de la tecnología actual e incluso ofreciéndole la posibilidad de participar en la creación de futuras tecnologías. Básicamente, el hardware abierto significa tener la posibilidad de mirar qué es lo que hay dentro de las cosas, y que eso sea éticamente correcto. Permite, en definitiva, mejorar la educación de las personas. Por eso el concepto de software y hardware libre es tan importante, no solo para el mundo de la informática y de la electrónica, sino para la vida en general.

2.2.4 ¿Por qué elegir Arduino?

Existen muchas otras placas de diferentes fabricantes que, aunque incorporan diferentes modelos de microcontroladores, son comparables y ofrecen una funcionalidad más o menos similar a la de las placas Arduino. Todas ellas también vienen acompañadas de un entorno de desarrollo agradable y cómodo y de un lenguaje de programación sencillo y completo. No obstante, la plataforma Arduino (hardware + software) ofrece una serie de ventajas:

- **Arduino es libre y extensible:** esto quiere decir que cualquiera que desee ampliar y mejorar tanto el diseño hardware de las placas como el entorno de desarrollo software y el propio lenguaje de programación, puede hacerlo sin problemas. Esto permite que exista un rico “ecosistema” de extensiones, tanto de variantes de placas no oficiales como de librerías software de terceros, que pueden adaptarse mejor a nuestras necesidades concretas.

- **Arduino tiene una gran comunidad:** muchas personas lo utilizan, enriquecen la documentación y comparten continuamente sus ideas.

- **Su entorno de programación es multiplataforma:** se puede instalar y ejecutar en sistemas Windows, Mac OS X y Linux. Esto no ocurre con el software de muchas otras placas.

- **Su entorno y el lenguaje de programación son simples y claros:** son muy fáciles de aprender y de utilizar, a la vez que flexibles y completos para que los usuarios avanzados puedan aprovechar y exprimir todas las posibilidades del hardware. Además, están bien documentados, con ejemplos detallados y gran cantidad de proyectos publicados en diferentes formatos.

- **Las placas Arduino son baratas:** la placa Arduino estándar (llamada Arduino UNO) ya preensamblada y lista para funcionar cuesta alrededor de

300 pesos. Incluso, uno mismo se la podría construir (Arduino es hardware libre, recordemos) adquiriendo los componentes por separado, con lo que el precio total de la placa resultante sería incluso menor.

- **Las placas Arduino son reutilizables y versátiles:** reutilizables porque se puede aprovechar la misma placa para varios proyectos (ya que es muy fácil de desconectarla, reconectarla y reprogramarla), y versátiles porque las placas Arduino proveen varios tipos diferentes de entradas y salidas de datos, los cuales permiten capturar información de sensores y enviar señales a actuadores de múltiples formas.

2.2.5 ¿Qué son las librerías?

Una librería (mala traducción del inglés “library”, que significa “biblioteca”) es un conjunto de instrucciones de un lenguaje de programación agrupadas de una forma coherente. Se puede entender una librería como un “cajón” etiquetado que guarda unas determinadas instrucciones relacionadas entre sí. Así, se tiene el “cajón” de las instrucciones para controlar motores, el “cajón” de las instrucciones para almacenar datos en una tarjeta de memoria, etc.

Las librerías sirven para proveer funcionalidad extra (manipulando datos, interactuando con hardware, etc.) pero también para facilitar el desarrollo de proyectos, porque están diseñadas para que a la hora de escribir nuestro programa no se tenga que hacer el trabajo de conocer todos los detalles técnicos sobre el manejo de un determinado hardware, o ser un experto en la configuración de determinado componente, ya que las librerías ocultan esa complejidad.

Además, la organización del lenguaje en librerías permite que los programas resulten más pequeños y sean escritos de una forma más flexible y modular, ya que

solo están disponibles en cada momento las instrucciones ofrecidas por las librerías utilizadas en ese instante.

2.2.6 Arduino UNO

El Arduino Uno utiliza el microcontrolador ATmega328. En adición a todas las características de las tarjetas anteriores, el Arduino Uno utiliza el ATmega16U2 para el manejo de USB en lugar del 8U2 (o del FTDI encontrado en generaciones previas). Esto permite velocidades de transferencia altas y más memoria. No se necesitan drivers para Linux o Mac (el archivo inf para Windows es necesario y está incluido en el IDE de Arduino).

La tarjeta Arduino Uno incluso añade pins SDA y SCL cercanos al AREF. Es más, hay dos nuevos pines cerca del pin RESET. Uno es el IOREF, que permite a los shields adaptarse al voltaje brindado por la tarjeta. El otro pin no se encuentra conectado y está reservado para propósitos futuros. La tarjeta trabaja con todos los shields existentes y podrá adaptarse con los nuevos shields utilizando esos pines adicionales.

El Arduino es una plataforma computacional física open-source basada en una simple tarjeta de I/O y un entorno de desarrollo que implementa el lenguaje Processing/Wiring. El Arduino Uno puede ser utilizado para desarrollar objetos interactivos o puede ser conectado a software de tu computadora (por ejemplo, Flash, Processing, MaxMSP). El IDE open-source puede ser descargado gratuitamente (actualmente para Mac OS X, Windows y Linux).

Características:

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad.

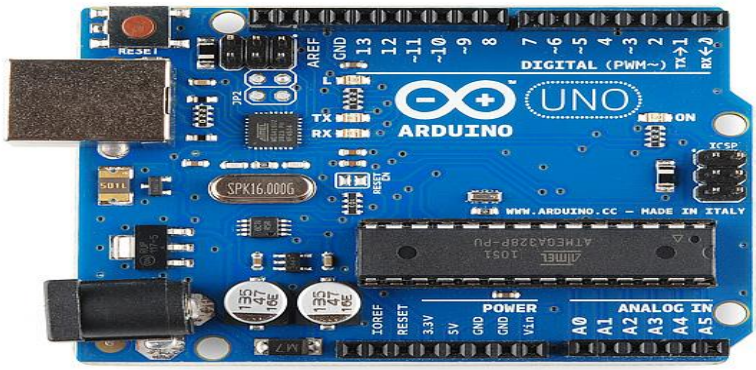

Figura 2.5: Arduino UNO.

2.2.7 Características del micro de la placa Arduino uno

Existe un modelo “estándar” de placa, que es el más utilizado con diferencia y que es el que utilizaremos también nosotros en este libro en todos los proyectos: la placa Arduino UNO. Desde que apareció en 2010 ha sufrido tres revisiones, por lo que el modelo actual se suele llamar UNO Rev3 o simplemente UNO R3.

2.2.7.1 El modelo del Microcontrolador

El microcontrolador que lleva la placa Arduino UNO es el modelo ATmega328P de la marca Atmel. La “P” del final significa que este chip incorpora la tecnología “Picopower” (propietaria de Atmel), la cual permite un consumo eléctrico sensiblemente menor comparándolo con el modelo equivalente sin “Picopower”, el Atmega328 (sin la “P”). De todas formas, aunque el ATmega328P pueda trabajar a un voltaje menor y consumir menos corriente que el Atmega328 (especialmente en los modos de hibernación), ambos modelos son funcionalmente idénticos.

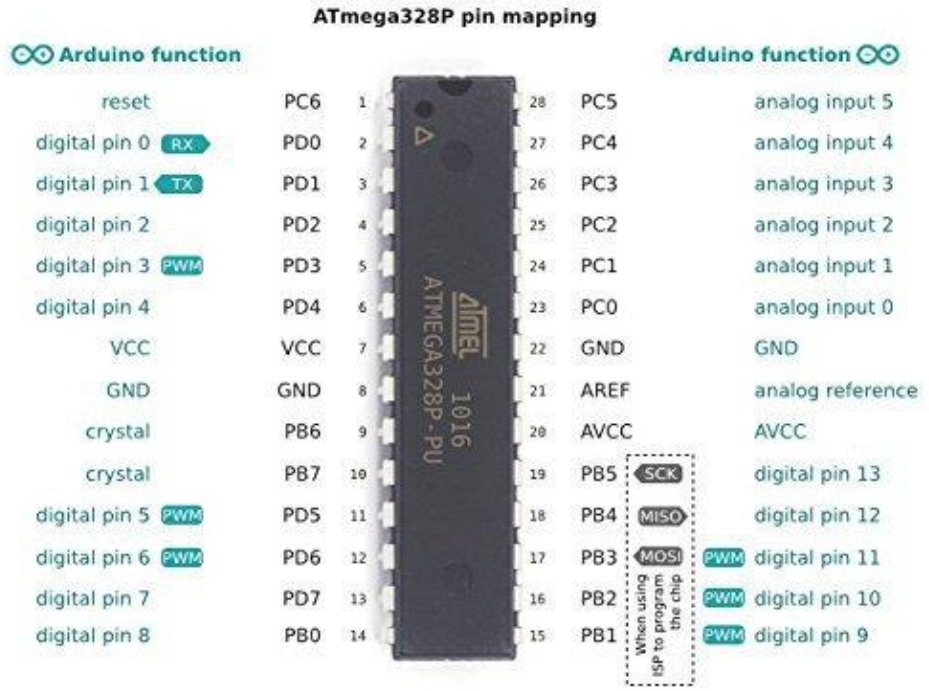

Figura 2.6: ATmega328P.

Al igual que ocurre con el resto de los microcontroladores usados en otras placas Arduino, el ATmega328P tiene una arquitectura de tipo AVR, arquitectura desarrollada por Atmel y en cierta medida “competencia” de otras arquitecturas como por ejemplo la PIC del fabricante Microchip. Más concretamente, el ATmega328P pertenece a la subfamilia de microcontroladores “megaAVR”. Otras

subfamilias de la arquitectura AVR son la “tinyAVR” (cuyos microcontroladores son más limitados y se identifican con el nombre de ATtiny) y la “XMEGA” (cuyos microcontroladores son más capaces y se identifican con el nombre de ATxmega).

2.2.7.2 Las memorias del microcontrolador

Otra cosa que hay que saber de los microcontroladores son los tipos y cantidades de memoria que alojan en su interior. En el caso del ATmega328P tenemos:

Memoria Flash: memoria persistente donde se almacena permanentemente el programa que ejecuta el microcontrolador (hasta una nueva reescritura si se da el caso). En el caso del ATmega328P tiene una capacidad de 32KB.

En los microcontroladores que vienen incluidos en la placa Arduino no se podra usar toda la capacidad de la memoria Flash porque existen 512 bytes (el llamado “bootloader block”) ocupados ya por un código preprogramado de fábrica (el llamado “bootloader” o “gestor de arranque”), el cual permite usar la placa Arduino de una forma sencilla y cómoda sin tener que conocer las interioridades electrónicas más avanzadas del microcontrolador. Los ATmega328P que se adquieran individualmente normalmente no incluyen de fábrica este pequeño programa, por lo que sí que ofrecen los 32 KB íntegros, pero a cambio no se podrá esperar conectarlos a una placa Arduino y que funcionen sin más ya que les faltará tener grabada esa “preconfiguración”.

Memoria SRAM: memoria volátil donde se alojan los datos que en ese instante el programa (grabado separadamente en la memoria Flash, recordemos) necesita crear o manipular para su correcto funcionamiento. Estos datos suelen tener un contenido variable a lo largo del tiempo de ejecución del programa y cada uno es de un tipo concreto (es decir, un dato puede contener un valor numérico entero, otro un número decimal, otro un valor de tipo carácter… también pueden ser

cadenas de texto fijas u otros tipos de datos más especiales). Independientemente del tipo de dato, su valor siempre será eliminado cuando se deje de alimentar eléctricamente al microcontrolador. En el caso del ATmega328P esta memoria tiene una capacidad de 2KB.

Si se necesitara ampliar la cantidad de memoria SRAM disponible, siempre se puede adquirir memorias SRAM independientes y conectarlas al microcontrolador utilizando algún protocolo de comunicación conocido por este (como SPI o I2 C.

Memoria EEPROM: memoria persistente donde se almacenan datos que se desea que permanezcan grabados una vez apagado el microcontrolador para poderlos usar posteriormente en siguientes reinicios. En el caso del ATmega328P esta memoria tiene una capacidad de 1 KB, por lo que se puede entender como una tabla de 1024 posiciones de un byte cada una.

Si se necesitara ampliar la cantidad de memoria EEPROM disponible, siempre se puede adquirir memorias EEPROM independientes y conectarlas al microcontrolador utilizando algún protocolo de comunicación conocido por este (como SPI o I2 C). O bien, alternativamente, adquirir tarjetas de memoria de tipo SD (“Secure Digital”) y comunicarlas mediante un circuito específico al microcontrolador. Las memorias SD son en realidad simples memorias Flash, encapsuladas de una forma concreta; son ampliamente utilizadas en cámaras digitales de foto/vídeo y en teléfonos móviles de última generación, ya que ofrecen muchísima capacidad (varios gigabytes) a un precio barato. La razón por la cual este tipo de tarjetas son capaces de ser reconocidas por el ATmega328P es porque pueden funcionar utilizando el protocolo de comunicación SPI.

Toda esta información se puede deducir en que la arquitectura a la que pertenece el chip ATmega328P (y en general, toda la familia de microcontroladores AVR) es de tipo Harvard. En este tipo de arquitectura, la memoria que aloja los datos (en nuestro caso, la SRAM o la EEPROM) está separada de la memoria que aloja

las instrucciones (en nuestro caso, la Flash), por lo que ambas memorias se comunican con la CPU de forma totalmente independiente y en paralelo, consiguiendo así una mayor velocidad y optimización. Otro tipo de arquitectura (que es la que vemos en los PCs) es la arquitectura Von Neumann, en la cual la CPU está conectada a una memoria RAM única que contiene tanto las instrucciones del programa como los datos, por lo que la velocidad de operación está limitada (entre otras cosas) por el efecto cuello de botella que significa un único canal de comunicación para datos e instrucciones.

2.2.7.3 Los registros del microcontrolador

Los registros son espacios de memoria existentes dentro de la propia CPU de un microcontrolador. Son muy importantes porque tienen varias funciones imprescindibles: sirven para albergar los datos (cargados previamente desde la memoria SRAM o EEPROM) necesarios para la ejecución de las instrucciones previstas próximamente (y así tenerlos perfectamente disponibles en el momento adecuado); sirven también para almacenar temporalmente los resultados de las instrucciones recientemente ejecutadas (por si se necesitan en algún instante posterior) y sirven además para alojar las propias instrucciones que en ese mismo momento estén ejecutándose.

Su tamaño es muy reducido: tan solo tienen capacidad para almacenar unos pocos bits cada uno. Pero este factor es una de las características más importantes de cualquier microcontrolador, ya que cuanto mayor sea el número de bits que “quepan” en sus registros, mayores serán sus prestaciones, en cuanto a poder de cómputo y velocidad de ejecución. En efecto, es fácil ver (simplificando mucho) que un microcontrolador con registros el doble de grandes que otro podrá procesar el doble de cantidad de datos y por tanto, trabajar el “doble de rápido” aun funcionando los dos al mismo ritmo. De hecho, es tan importante esta característica que cuando escuchamos que un microcontrolador es de “8 bits” o de “32 bits”, nos estamos refiriendo precisamente a este dato: al tamaño de sus registros.

Dependiendo de la utilidad que se le vaya a dar al microcontrolador, será necesario utilizar uno con un tamaño de registros suficiente. Por ejemplo, el control de un electrodoméstico sencillo como una batidora no requiere más que un microcontrolador de 4 u 8 bits. En cambio, el sistema de control electrónico del motor de un coche o el de un sistema de frenos ABS se basan normalmente en un microcontrolador de 16 o 32 bits.

El chip ATmega328P es de 8 bits. De hecho, todos los microcontroladores que incorporan las diferentes placas Arduino son de 8 bits excepto el incorporado en la placa Arduino Due, que es de 32 bits.

2.2.7.4 El chip ATmega16U2

La conexión USB de la placa Arduino, además de servir como alimentación eléctrica, sobre todo es un medio para poder transmitir datos entre el computador y la placa, y viceversa. Este tráfico de información que se realiza entre ambos aparatos se logra gracias al uso del protocolo USB, un protocolo de tipo serie que tanto el computador como la placa Arduino son capaces de entender y manejar. No obstante, el protocolo USB internamente es demasiado complejo para que el microcontrolador ATmega328P pueda comprenderlo por sí mismo sin ayuda, ya que él tan solo puede comunicarse con el exterior mediante protocolos mucho más sencillos técnicamente como son el I2 C o el SPI y pocos más. Por tanto, es necesario que la placa disponga de un elemento “traductor” que facilite al ATmega328P (concretamente, al receptor/transmisor serie de tipo TTL-UART que lleva incorporado) la manipulación de la información transferida por USB sin que este tenga que conocer los entresijos de dicho protocolo.

La placa Arduino UNO R3 dispone de un chip que realiza esta función de “traductor” del protocolo USB a un protocolo serie más sencillo (y viceversa). Ese chip es el ATmega16U2. El ATmega16U2 es todo un microcontrolador en sí mismo (con su propia CPU, con su propia memoria –tiene por ejemplo 16 Kilobytes de

memoria Flash para su uso interno, de ahí su nombre–, etc.) y por tanto podría realizar muchas más tareas que no solo la “traducción” del protocolo USB. De hecho, técnicamente es posible desprogramarlo para que haga otras cosas y convertir así la placa Arduino en virtualmente cualquier tipo de dispositivo USB conectado al computador (un teclado, un ratón, un dispositivo MIDI…). No obstante, por defecto el ATmega16U2 que viene incluido en la placa Arduino viene ya con el firmware preprogramado para realizar exclusivamente la función de “intérprete” al ATmega328P y ya está.

Este firmware es software libre, por lo que se puede acceder a su código fuente y también están disponible su correspondiente fichero “.hex”, dentro del conjunto de ficheros descargados, junto con el entorno de desarrollo oficial de Arduino (concretamente, dentro de la carpeta “firmwares”, dentro de “hardware/arduino”).

En modelos de la placa Arduino anteriores al UNO (como el modelo NG, el Diecimila o el Duemilanove) el chip ATmega16U2 no venía: en su lugar aparecía un circuito conversor de USB a serie del fabricante FTDI, concretamente el FT232RL. Una ventaja de haber sustituido el FT232RL por el ATmega16U2 es el precio. Otra ventaja es el tener la posibilidad de reprogramar el ATmega16U2 para que en vez de funcionar como un simple conversor USB-Serie pueda simular ser cualquier otro tipo de dispositivo USB, que, por ejemplo, con el FT232RL no se podría porque está pensado para hacer tan solo la función para la que fue construido.

El ATmega16U2 viene acompañado en la placa Arduino por un reloj oscilador de cristal, de uso exclusivo para él, que sirve para mantener la sincronización con la comunicación USB.

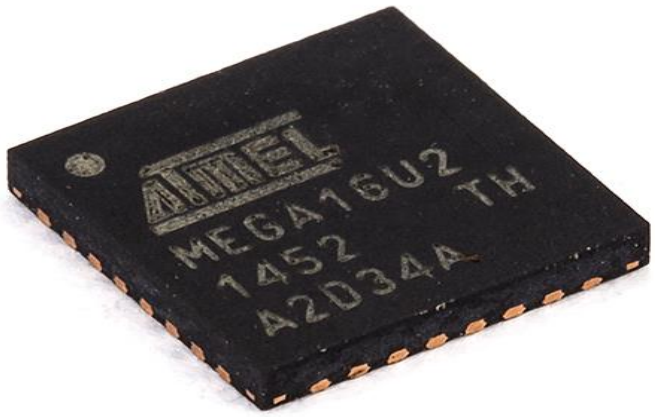

Figura 2.7: Chip ATmega16U2.

2.2.7.5 Las entradas y salidas digitales

La placa Arduino dispone de 14 pines-hembra de entradas o salidas (según lo que convenga) digitales, numeradas desde la 0 hasta la 13. Es aquí donde conectaremos nuestros sensores para que la placa pueda recibir datos del entorno, y también donde conectaremos los actuadores para que la placa pueda enviarles las órdenes pertinentes, además de poder conectar cualquier otro componente que necesite comunicarse con la placa de alguna manera. A veces a estos pines-hembra digitales de “propósito general” se les llama pines GPIO (de “General Purpose Input/Output”).

Todos estos pines-hembra digitales funcionan a 5 V, pueden proveer o recibir un máximo de 40 mA y disponen de una resistencia “pull-up” interna de entre 20 KΩ y 50 KΩ que inicialmente está desconectada (salvo que nosotros indiquemos lo contrario mediante programación software).

Hay que tener en cuenta, no obstante, aunque cada pin individual pueda proporcionar hasta 40 mA como máximo, en realidad, internamente la placa agrupa los pines digitales de tal forma que tan solo pueden aportar 100 mA a la vez el conjunto de los pines nº 0,1,2,3 y 4, y 100 mA más el resto de los pines (del 5 al 13). Esto quiere decir que como mucho podríamos tener 10 pines ofreciendo 20 mA a la vez.

2.2.7.6 Las entradas analógicas

La placa Arduino dispone de 6 entradas analógicas (en forma de pines hembra etiquetados como “A0”, “A1” ... hasta “A5”) que pueden recibir voltajes dentro de un rango de valores continuos de entre 0 y 5 V. No obstante, la electrónica de la placa tan solo puede trabajar con valores digitales, por lo que es necesaria una conversión previa del valor analógico recibido a un valor digital lo más aproximado posible. Esta se realiza mediante un circuito conversor analógico/digital incorporado en la propia placa.

El circuito conversor es de 6 canales (uno por cada entrada) y cada canal dispone de 10 bits (los llamados “bits de resolución”) para guardar el valor del voltaje convertido digitalmente.

En general, la cantidad de bits de resolución que tiene un determinado conversor analógico/digital es lo que marca en gran medida el grado de precisión conseguida en la conversión de señal analógica a digital, ya que cuantos más bits de resolución tenga, más fiel será la transformación. Por ejemplo, en el caso concreto del conversor incorporado en la placa Arduino, si contamos el número de combinaciones de 0s y 1s que se pueden obtener con 10 posiciones, vemos que hay un máximo de 210 (1024) valores diferentes posibles. Por tanto, la placa Arduino puede distinguir para el voltaje digital desde el valor 0 hasta el valor 1023. Si el conversor tuviera por ejemplo 20 bits de resolución, la variedad de valores digitales que podría distinguir sería muchísimo más grande (220 = 1048576) y podría afinar la precisión mucho más.

Esto es fácil verlo si se divide el rango analógico de entrada (5 V - 0V = 5 V) entre el número máximo posible de valores digitales (1024). Se obtendra que cada valor digital corresponde a una “ventana” analógica de aproximadamente 5 V/1024≈5 mV. En otras palabras: todos los valores analógicos dentro de cada rango de 5 mV (desde 0 a 5 V) se “colapsan” sin distinción en un único valor digital (desde 0 a 1023). Así pues, no se podrá distinguir valores analógicos distanciados por menos de 5 mV.

En muchos de los proyectos ya es suficiente este grado de precisión, pero en otros puede que no. Si el conversor analógico/digital tuviera más bits de resolución, el resultado de la división rango_analógico_entrada/número_ valores_digitales sería menor, y por tanto la conversión sería más rigurosa. Pero como no se pueden aumentar los bits de resolución del conversor de la placa, si se quiere más exactitud se ha de optar por otra solución: en vez de aumentar el denominador de la división anterior, se puede reducir su numerador (es decir, el rango analógico de entrada, o más específicamente, su límite superior –por defecto igual a 5 V–, ya que el inferior es 0). Este límite superior en la documentación oficial se suele nombrar como “voltaje de referencia”.

Por último, hay que decir que estos pines-hembra de entrada analógica tienen también toda la funcionalidad de los pines de entrada-salida digitales. Es decir, que si en algún momento se necesitan más pines-hembra digitales más allá de los 14 que la placa Arduino ofrece (del 0 al 13), los 6 pines-hembra analógicos pueden ser usados como unos pines-hembra digitales más (numerándose entonces del 14 al 19) sin ninguna distinción.

2.2.7.7 Las salidas analógicas

En los proyectos a menudo se necesitará enviar al entorno señales analógicas, por ejemplo, para variar progresivamente la velocidad de un motor, la frecuencia de un sonido emitido por un zumbador o la intensidad con la que luce un LED. No basta con simples señales digitales: tenemos que generar señales que

varíen continuamente. La placa Arduino no dispone de pines-hembra de salida analógica propiamente dichos (porque su sistema electrónico interno no es capaz de manejar este tipo de señales), sino que utiliza algunos pines-hembra de salida digitales concretos para “simular” un comportamiento analógico. Los pines-hembra digitales que son capaces trabajar en este modo no son todos: solo son los marcados con la etiqueta “PWM”. En concreto para el modelo Arduino UNO son los pines número: 3, 5, 6, 9, 10 y 11.

Las siglas PWM vienen de “Pulse Width Modulation” (Modulación de Ancho de Pulso). Lo que hace este tipo de señal es emitir, en lugar de una señal continua, una señal cuadrada formada por pulsos de frecuencia constante (aproximadamente de 490 Hz). La gracia está en que, al variar la duración de estos pulsos, se estara variando proporcionalmente la tensión promedio resultante. Es decir: cuanto más cortos sean los pulsos (y, por tanto, más distantes entre sí en el tiempo, ya que su frecuencia es constante), menor será la tensión promedio de salida, y cuanto más largos sean los pulsos (y por tantos, más juntos en el tiempo estén), mayor será dicha tensión. El caso extremo se tendría cuando la duración del pulso coincidiera con el período de la señal, momento en el cual de hecho no habría distancia entre pulso y pulso (sería una señal de un valor constante) y la tensión promedio de salida sería la máxima posible, que son 5 V. La duración del pulso se puede cambiar en cualquier momento mientras la señal se está emitiendo, por lo que como consecuencia la tensión promedio puede ir variando a lo largo del tiempo de forma continua. Las siguientes figuran ilustran lo acabado de explicar:

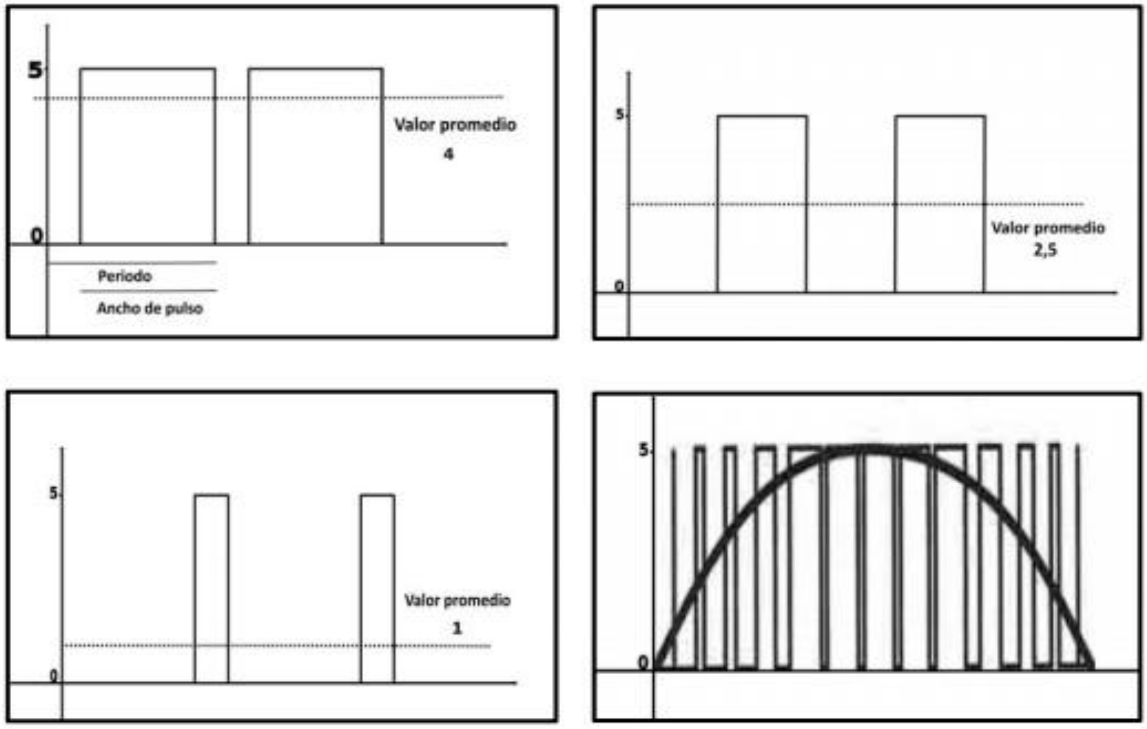

Figura 2.8: PWM.

Cada pin-hembra PWM tiene una resolución de 8 bits. Esto quiere decir que, si se cuentan el número de combinaciones de 0s y 1s que se pueden obtener con 8 posiciones, se obtendrán un máximo de 28 (256) valores diferentes posibles. Por tanto, se pueden tener 256 valores diferentes para indicar la duración deseada de los pulsos de la señal cuadrada (o dicho de otra forma: 256 valores promedio diferentes). Estableciendo (mediante programación software) el valor mínimo (0), se estarán emitiendo unos pulsos extremadamente estrechos y generando una señal analógica equivalente a 0 V; estableciendo el valor máximo (255), se estarán emitiendo pulsos de máxima duración y generando una señal analógica equivalente a 5 V. Cualquier valor entremedio emitirá pulsos de duración intermedia y, por tanto, generará una señal analógica de un valor entre 0 V y 5 V.

La diferencia de voltaje analógico existente entre dos valores promedio contiguos (es decir, entre por ejemplo el valor número 123 y el número 124) se puede calcular mediante la división: rango_voltaje_salida

/número_valores_promedio. En este caso, sería (5 V – 0 V)/256 ≈19,5 mV. Es decir, cada valor promedio está distanciado del anterior y del siguiente por un “saltito” de 19,5 mV.

Es posible cambiar la frecuencia por defecto de la señal cuadrada utilizada en la generación de la señal “analógica”, pero no es un procedimiento trivial, y en la mayoría de las ocasiones no será necesario. En este sentido, tan solo sé comentara que los pines PWM vienen controlados por tres temporizadores diferentes que mantienen la frecuencia constante de los pulsos emitidos; concretamente, los pines 3 y 11 son controlados por el “Timer1”, los pines 5 y 6 por el “Timer2” y los pines 9 y 10 por el “Timer3”.

2.2.7.8 Los protocolos de comunicación

Cuando se desea transmitir un conjunto de datos desde un componente electrónico a otro, se puede hacer de múltiples formas. Una de ellas es estableciendo una comunicación “serie”; en este tipo de comunicación la información es transmitida bit a bit (uno tras otro) por un único canal, enviando por tanto un solo bit en cada momento. Otra manera de transferir datos es mediante la llamada comunicación “paralela”, en la cual se envían varios bits simultáneamente, cada uno por un canal separado y sincronizado con el resto.

El microcontrolador, a través de algunos de sus pines de E/S, utiliza el sistema de comunicación serie para transmitir y recibir órdenes y datos hacia/desde otros componentes electrónicos. Esto es debido sobre todo a que en una comunicación serie solo se necesita en teoría un único canal (un único “cable”), mientras que en una comunicación en paralelo se necesitan varios cables, con el correspondiente incremento de complejidad, tamaño y coste del circuito resultante.

No obstante, no se puede hablar de un solo tipo de comunicación serie. Existen muchos protocolos y estándares diferentes basados todos ellos en la

transferencia de información en serie, pero implementando de una forma diferente cada uno los detalles específicos (como el modo de sincronización entre emisor y receptor, la velocidad de transmisión, el tamaño de los paquetes de datos, los mensajes de conexión y desconexión y de dar paso al otro en el intercambio de información, los voltajes utilizados, etc.). De entre el gran número de protocolos de comunicación serie reconocidos por la inmensa variedad de dispositivos electrónicos del mercado, los que se conocerán son los que el ATmega328P es capaz de comprender y, por tanto, los que podrá utilizar para contactar con esa variedad de periféricos. En este sentido, los estándares más importantes son:

I2C (Inter-Integrated Circuit, también conocido con el nombre de TWI –de "TWo-wIre", literalmente "dos cables" en inglés–): es un sistema muy utilizado en la industria principalmente para comunicar circuitos integrados entre sí. Su principal característica es que utiliza dos líneas para transmitir la información: una (llamada línea "SDA") sirve para transferir los datos (los 0s y los 1s) y otra (llamada línea "SCL") sirve para enviar la señal de reloj. En realidad, también se necesitarían dos líneas más: la de alimentación y la de tierra común, pero estas ya se presuponen existentes en el circuito.

Por "señal de reloj" se entiende una señal binaria de una frecuencia periódica muy precisa que sirve para coordinar y sincronizar los elementos integrantes de una comunicación (es decir, los emisores y receptores) de forma que todos sepan cuándo empieza, cuánto dura y cuándo acaba la transferencia de información. En hojas técnicas y diagramas a la señal de reloj en general se le suele describir como CLK (del inglés "clock").

Cada dispositivo conectado al bus I^2C tiene una dirección única que lo identifica respecto el resto de los dispositivos, y puede estar configurado como "maestro" o como "esclavo". Un dispositivo maestro es el que inicia la transmisión de datos y además genera la señal de reloj, pero no es necesario que el maestro

sea siempre el mismo dispositivo: esta característica se la pueden ir intercambiando ordenadamente los dispositivos que tengan esa capacidad.

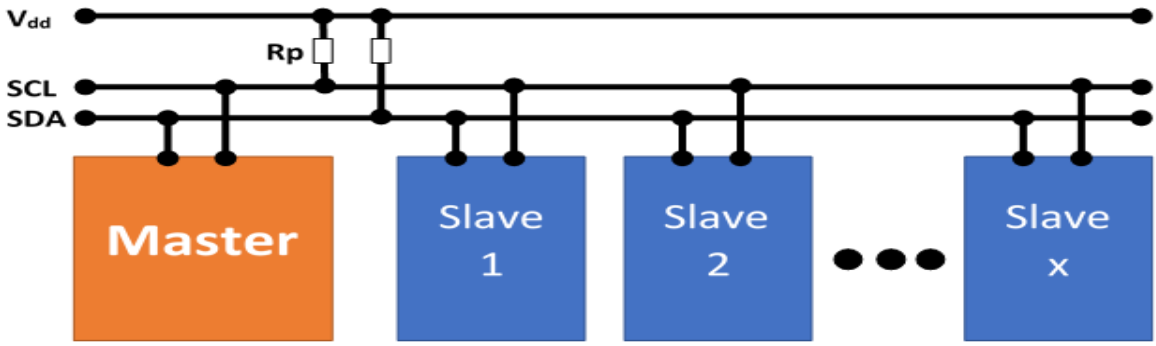

Figura 2.9: Protocolo I2C.

Tal como se muestra en el diagrama anterior, para funcionar correctamente tanto la línea “SDA” como la “SCL” necesitan estar conectadas mediante una resistencia “pull-up” a la fuente de alimentación común, la cual puede proveer un voltaje generalmente de 5 V o 3,3 V (aunque sistemas con otros voltajes pueden ser posibles).

La velocidad de transferencia de datos es de 100 Kbits por segundo en el modo estándar (aunque también se permiten velocidades de hasta 3,4 Mbit/s). No obstante, al haber una única línea de datos, la transmisión de información es “half duplex” (es decir, la comunicación solo se puede establecer en un sentido al mismo tiempo) por lo que en el momento que un dispositivo empiece a recibir un mensaje, tendrá que esperar a que el emisor deje de transmitir para poder responderle.

SPI (Serial Peripheral Interface): al igual que el sistema I²C, el sistema de comunicación SPI es un estándar que permite controlar (a cortas distancias) casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie sincronizado (es decir, regulado por un reloj). Igualmente, un dispositivo conectado al bus SPI puede ser “maestro” –en inglés, “master” – o “esclavo” –en inglés, “slave“–, donde

el primero es el que inicia la transmisión de datos y además genera la señal de reloj (aunque, como con I²C , con SPI tampoco es necesario que el maestro sea siempre el mismo dispositivo) y el segundo se limita a responder.

La mayor diferencia entre el protocolo SPI y el I²C es que el primero requiere de cuatro líneas (“cables”) en vez de dos. Una línea (llamada normalmente “SCK”) envía a todos los dispositivos la señal de reloj generada por el maestro actual; otra (llamada normalmente “SS”) es la utilizada por ese maestro para elegir en cada momento con qué dispositivo esclavo se quiere comunicar de entre los varios que puedan estar conectados (ya que solo puede transferir datos con un solo esclavo a la vez); otra (llamada normalmente “MOSI”) es la línea utilizada para enviar los datos –0s y 1s– desde el maestro hacia el esclavo elegido; y la otra (llamada normalmente “MISO”) es la utilizada para enviar los datos en sentido contrario: la respuesta de ese esclavo al maestro. Es fácil ver que, al haber dos líneas para los datos la transmisión de información es “full duplex” (es decir, que la información puede ser transportada en ambos sentidos a la vez).

En las siguientes figuras se muestra el esquema de líneas de comunicación existentes entre un maestro y un esclavo y entre un maestro y tres esclavos respectivamente. Se puede observar que, para el caso de la existencia de varios esclavos es necesario utilizar una línea “SS” diferente por cada uno de ellos, ya que esta línea es la que sirve para activar el esclavo concreto que en cada momento el maestro desee utilizar (esto no pasa con las líneas de reloj, “MOSI” y “MISO”, que son compartidas por todos los dispositivos).Técnicamente hablando, el esclavo que reciba por su línea SS un valor de voltaje BAJO será el que esté seleccionado en ese momento por el maestro, y los que reciban el valor ALTO no lo estarán (de ahí el subrayado superior que aparece en la figura).

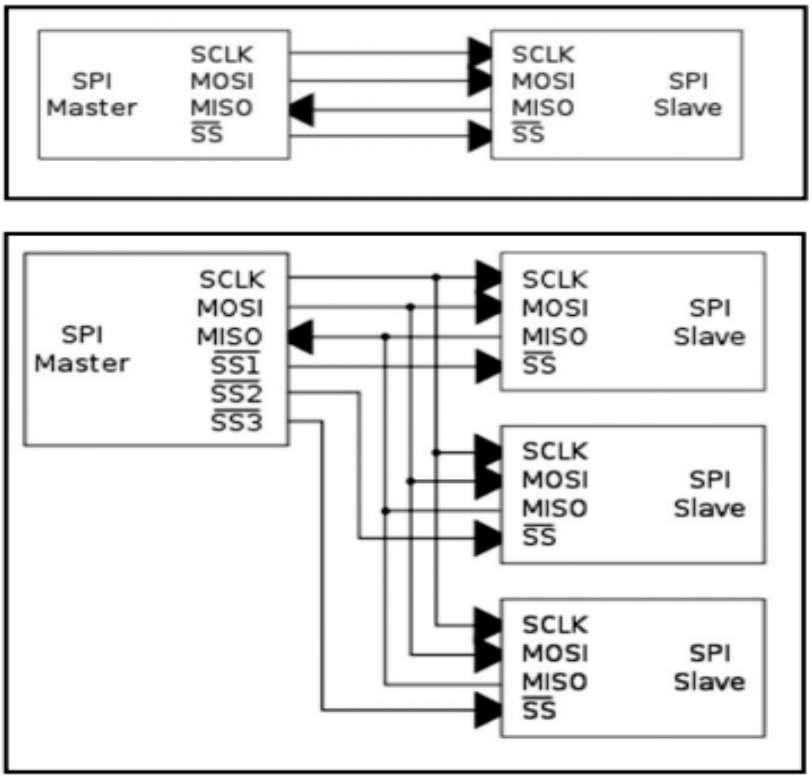

Figura 2.10: Protocolo SPI.

Como se puede ver, el protocolo SPI respecto el I²C tiene la desventaja de exigir al microcontrolador dedicar muchos más pines de E/S a la comunicación externa. En cambio, como ventaja podemos destacar que es más rápido y consume menos energía que I²C.

Tal como se puede observar en la figura que muestra la disposición de pines del microcontrolador ATmega328P, los pines correspondientes a las líneas I2C, SDA y SCL son los números 27 y 28, respectivamente, y los pines correspondientes a las líneas SPI, SS, MOSI, MISO y SCK son los números 16, 17, 18 y 19, respectivamente. Si se necesitaran más líneas SS (porque haya más de un dispositivo esclavo conectado en nuestro circuito), se podría utilizar cualquier otro pin de E/S siempre que respete el convenio de poner el valor de su voltaje de salida

a BAJO cuando se desee trabajar con el dispositivo esclavo asociado y poner a ALTO el resto de los pines SS.

2.2.7.9 La alimentación del Arduino

El voltaje de funcionamiento de la placa Arduino (incluyendo el microcontrolador y el resto de los componentes) es de 5 V. Se puede obtener esta alimentación eléctrica de varias maneras:

Conectando la placa Arduino a una fuente externa, tal como un adaptador AC/DC o una pila. Para el primer caso, la placa dispone de un zócalo donde se puede enchufar una clavija de 2,1 milímetros de tipo “jack”. Para el segundo, los cables salientes de los bornes de la pila se pueden conectar a los pines-hembra marcados como “Vin” y “Gnd” (positivo y negativo respectivamente) en la zona de la placa marcada con la etiqueta “POWER”. En ambos casos, la placa está preparada en teoría para recibir una alimentación de 6 a 20 voltios, aunque, realmente, el rango recomendado de voltaje de entrada (teniendo en cuenta el deseo de obtener una cierta estabilidad y seguridad eléctricas en nuestros circuitos) es menor: de 7 a 12 voltios. En cualquier caso, este voltaje de entrada ofrecido por la fuente externa siempre es rebajado a los 5 V de trabajo mediante un circuito regulador de tensión que ya viene incorporado dentro de la placa.

Conectando la placa Arduino a nuestro computador mediante un cable USB. Para ello, la placa dispone de un conector USB hembra de tipo B. La alimentación recibida de esta manera está regulada permanentemente a los 5 V de trabajo y ofrece un máximo de hasta 500 mA de corriente (por lo tanto, la potencia consumida por la placa es en ese caso de unos 2,5 W). Si en algún momento por el conector USB pasa más intensidad de la deseable, la placa Arduino está protegida mediante un polifusible reseteable que automáticamente rompe la conexión hasta que las condiciones eléctricas vuelven a la normalidad. Una consecuencia de esta protección contra posibles picos de corriente es que la intensidad de corriente

recibida a través de USB puede no ser suficiente para proyectos que contengan componentes tales como motores, solenoides o matrices de LEDs, los cuales consumen mucha potencia.

Sea cual sea la manera elegida para alimentar la placa, esta es lo suficientemente “inteligente” para seleccionar automáticamente en cada momento la fuente eléctrica disponible y utilizar una u otra sin que se tenga que hacer nada especial al respecto.

Si se utiliza una pila como alimentación externa, una ideal sería la de 9 V (está dentro del rango recomendado de 7 a 12 voltios), y si se utiliza un adaptador AC/DC, se recomienda el uso de uno con las siguientes características:

El voltaje de salida ofrecido ha de ser de 9 a 12 V DC. En realidad, el circuito regulador que lleva incorporado la placa Arduino es capaz de manejar voltajes de salida (de entrada para la placa) de hasta 20 V, así que en teoría se podrían utilizar adaptadores AC/DC que generen una salida de 20 V DC. No obstante, esta no es una buena idea porque se pierde la mayoría del voltaje en forma de calor (lo cual es terriblemente ineficiente) y además puede provocar el sobrecalentamiento del regulador, y como consecuencia dañar la placa.

La intensidad de corriente ofrecida ha de ser de 250 mA (o más). Si conectamos a nuestra placa Arduino muchos componentes o unos pocos, pero consumidores de mucha energía (como por ejemplo una matriz de LEDs, una tarjeta SD o un motor) el adaptador debería suministrar al menos 500 mA o incluso 1 A. De esta manera nos aseguraremos de que tenemos suficiente corriente para que cada componente pueda funcionar de forma fiable.

El adaptador ha de ser de polaridad “con el positivo en el centro”. Esto quiere decir que la parte externa del cilindro metálico que forma la clavija de 5,5/2,1 mm del adaptador ha de ser el borne negativo y el hueco interior del cilindro ha de ser el borne positivo. Lo más sencillo para asegurarse de que el adaptador es el adecuado en este sentido es observar si tiene imprimido en algún sitio el siguiente símbolo:

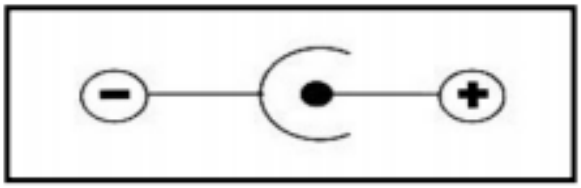

Figura 2.11: Simbología de polaridad positiva en el centro.

Por otro lado, dentro de la zona etiquetada como “POWER” en la placa Arduino existe una serie de pines-hembra relacionados con la alimentación eléctrica, como son:

” **GND”**: pines-hembra conctados a tierra. Es muy importante que todos los componentes de los circuitos compartan una tierra común como referencia. Estos pines-hembra se ofrecen para realizar esta función.

”**Vin**”: este pin-hembra se puede utilizar para dos cosas diferentes: si la placa está conectada mediante la clavija de 2,1mm a alguna fuente externa que aporte un voltaje dentro de los márgenes de seguridad, se puede conectar a este pin-hembra cualquier componente electrónico para alimentarlo directamente con el nivel de voltaje que esté aportando la fuente en ese momento (¡sin regular por la placa!) . Si la placa está alimentada mediante USB, entonces ese pin-hembra aportará 5 V regulados. En cualquier caso, la intensidad de corriente máxima aportada es de 40 mA (esto hay que tenerse en cuenta cuando se conecten dispositivos que consuman mucha corriente, como por ejemplo motores). También se puede usar el pin-hembra “Vin” para otra cosa: para alimentar la propia placa directamente desde alguna fuente de alimentación externa sin utilizar ni la clavija ni el cable USB. Esto se hace conectándole el borne positivo de la fuente (por ejemplo, una pila de 9 V) y conectando el borne negativo al pin de tierra. Si se usa este montaje, el regulador de tensión que incorpora la placa reducirá el voltaje recibido de la pila al voltaje de trabajo de la placa (los 5 V).

”**5 V**”: este pin-hembra se puede utilizar para dos cosas diferentes: tanto si la placa está alimentada mediante el cable USB como si está alimentada por una fuente externa que aporte un voltaje dentro de los márgenes de seguridad, se puede conectar a este pin-hembra cualquier componente para que pueda recibir 5 V regulados. En cualquier caso, la intensidad de corriente máxima generada será de 40 mA. Pero también se puede usar este pin hembra para otra cosa: para alimentar la propia placa desde una fuente de alimentación externa previamente regulada a 5 V sin utilizar el cable USB ni la clavija de 2,1mm.

”**3,3 V**”: este pin-hembra ofrece un voltaje de 3,3 voltios. Este voltaje se obtiene a partir del recibido indistintamente a través del cable USB o de la clavija de 2,1 mm, y está regulado (con un margen de error del 1%) por un circuito específico incorporado en la placa: el LP2985. En este caso particular, la corriente máxima generada es de 50 mA. Al igual que con los pines anteriores, se puede usar este pin para alimentar componentes de los circuitos que requieran dicho voltaje (los más delicados), pero en cambio, no se podrá conectar ninguna fuente externa aquí porque el voltaje es demasiado limitado para poder alimentar a la placa.

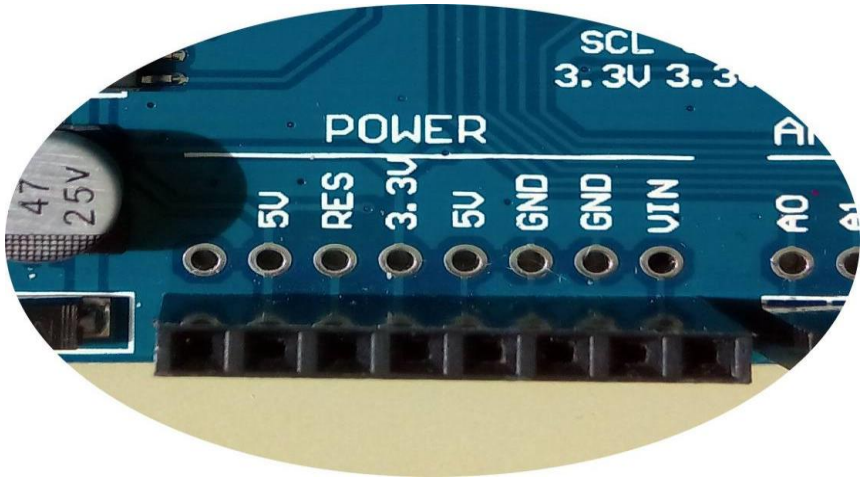

Figura 2.12: Conectores de alimentación del Arduino.

2.2.7.10 Otros usos de los pines hembra en Arduino

Pin 0 (RX) y pin 1 (TX): permiten que el microcontrolador ATmega328P pueda recibir directamente datos en serie (por el pin RX) o transmitirlos (por el pin TX) sin pasar por la conversión USB-Serie que realiza el chip ATmega16U2. Es decir, estos pines posibilitan la comunicación sin intermediarios de dispositivos externos con el receptor/transmisor serie (de tipo TTL-UART) que incorpora el propio ATmega328P. De todas maneras, estos pines están internamente conectados (mediante resistencias de 1 KΩ) al chip ATmega16U2, por lo que los datos disponibles en el USB también lo estarán en estos pines.

Pines 2 y 3: se pueden usar, con la ayuda de programación software, para gestionar interrupciones.

Pines 10 (SS), 11 (MOSI) , 12 (MISO) y 13 (SCK): se pueden usar para conectar algún dispositivo con el que se quiera llevar a cabo comunicaciones mediante el protocolo SPI.

Pin 13: este pin está conectado directamente a un LED incrustado en la placa (identificado con la etiqueta “L”) de forma que si el valor del voltaje recibido por este pin es ALTO (HIGH), el LED se encenderá, y si dicho valor es BAJO (LOW), el LED se apagará. Es una manera sencilla, y rápida de detectar señales de entradas externas sin necesidad de disponer de ningún componente extra.

También existen un par de pines-hembra de entrada analógica que tienen una función extra además de la habitual:

Pines A4 (SDA) y A5 (SCL): se pueden usar para conectar algún dispositivo con el que se quiera llevar a cabo comunicaciones mediante el protocolo I2C/TWI. La placa Arduino ofrece (por una simple cuestión de comodidad y ergonomía) una

duplicación de estos dos pines-hembra en los dos últimos pines-hembra tras el pin “AREF”, los cuales están sin etiquetar porque no hay más espacio físico.

Finalmente, a lo largo de la placa existen diferentes pines-hembra no comentados todavía que no funcionan ni como salidas ni como entradas porque tienen un uso muy específico y concreto:

Pin AREF: ofrece un voltaje de referencia externo para poder aumentar la precisión de las entradas analógicas.

Pin RESET: si el voltaje de este pin se establece a valor BAJO (LOW), el microcontrolador se reiniciará y se pondrá en marcha el bootloader. Para realizar esta misma función, la placa Arduino ya dispone de un botón, pero este pin ofrece la posibilidad de añadir otro botón de reinicio a placas supletorias (es decir, placas que se conectan encima de la placa Arduino para ampliarla y complementarla), las cuales por su colocación puedan ocultar o bloquear el botón de la placa Arduino.

Pin IOREF: en realidad este pin es una duplicación regulada del pin “Vin”. Su función es indicar a las placas supletorias conectadas a la placa Arduino el voltaje al que trabajan los pines de entrada/salida de esta, para que las placas supletorias se adapten automáticamente a ese voltaje de trabajo (que en el caso del modelo UNO ya es sabido que es 5 V).

Pin sin utilizar: justo el pin a continuación del IOREF, el cual está sin etiquetar, actualmente no se utiliza para nada, pero se reserva para un posible uso futuro.

Una vez conocidos todos los pines-hembra de la placa Arduino UNO, es muy interesante observar qué correspondencia existe entre cada uno de ellos y las patillas del microcontrolador ATmega328P. Porque en realidad, la mayoría de estos

pines hembra lo que hacen es simplemente ofrecer de una forma fácil y cómoda una conexión directa a esas patillas, y poco más.

2.2.8 Que es un IDE

Un programa es un conjunto concreto de instrucciones, ordenadas y agrupadas de forma adecuada y sin ambigüedades que pretende obtener un resultado determinado. Cuando se dice que un microcontrolador es “programable”, se está diciendo que permite grabar en su memoria de forma permanente (hasta que se regrabe de nuevo si es necesario) el programa deseado para que dicho microcontrolador ejecute. Si no se introduce ningún programa en la memoria del microcontrolador, este no sabrá qué hacer.

Las siglas IDE vienen de Integrated Development Environment, lo que traducido al español significa Entorno de Desarrollo Integrado. Esto es simplemente una forma de llamar al conjunto de herramientas software que permite a los programadores poder desarrollar (es decir, básicamente escribir y probar) sus propios programas con comodidad. En el caso de Arduino, se necesita un IDE que permita escribir y editar el programa (también llamado “sketch” en el mundo de Arduino), permitiendo comprobar que no se haya cometido ningún error y que además permita, cuando se esté seguro de que el sketch es correcto, grabarlo en la memoria del microcontrolador de la placa Arduino para que este se convierta a partir de entonces en el ejecutor autónomo de dicho programa.

Para poder empezar a desarrollar los sketches (o probar algun) se debera instalar en el computador el IDE que proporciona el proyecto Arduino en su página oficial.

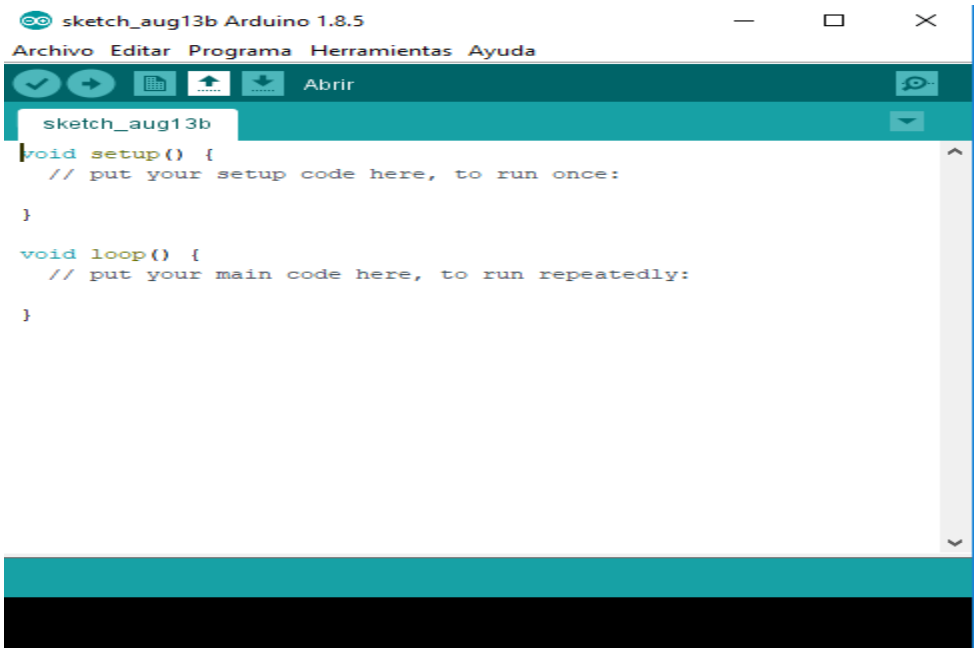

Figura 2.13: IDE Arduino.

2.3 Modulo WIFI ESP8266 01

ESP8266 es un puente de puerto serie a WiFi, incluye un microcontrolador para manejar el protocolo TCP/IP y el software necesario para la conexión 802.11, la mayoría de los modelos dispone de entradas/salidas (I/O) digitales y algunos modelos una entrada analógica al igual que otros microcontroladores, su punto fuerte es disponer de acceso WIFI y por su bajo precio el chip ESP8266 parece destinado a dar un gran empujón a lo que se ha llamado Internet de las cosas.

ESP8266 se puede programar usando el lenguaje interpretado Lua en entornos como ESPlorer, y el IDE y lenguaje de Arduino Processing/Wiring. Es diseñado por una compañía china llamada Espressif Systems en su sede en Shangai. Pero su producción en masa inicio hasta principios del año 2014, donde se anunció que este chip sería una excelente solución autómata de redes wifi que se ofrece como puente entre los microcontroladores que hasta ahora existen o que tiene la capacidad de ejecutar aplicaciones independientes.

Un ESP8266 salido de fábrica no sería de mucha utilidad ya que su producción está basada en la compactación de un chip SMT (Tecnología de Montaje Superficial por sus siglas en inglés - Surface Mount Technology) el cual viene en un pequeño paquete de tan solo cinco milímetros cuadrados. La buena noticia es que gracias a que diversos fabricantes que construyen placas de circuito impreso prefabricadas adecuándolos y dejándolos listos para ser usados. Esto permite trabajar con este dispositivo único acoplado a un microcontrolador, para desarrollar proyectos o como sistema autónomo para ciertas aplicaciones.

Características generales:

- CPU RISC de 32-bit: Tensilica Xtensa LX106 a un reloj de 80 MHzª
- RAM de instrucción de 64 KB, RAM de datos de 96 KB
- Capacidad de memoria externa flash QSPI - 512 KB a 4 MB* (puede soportar hasta 16 MB)
- IEEE 802.11 b/g/n Wi-Fi
  - Tiene integrados: TR switch, balun, LNA, amplificador de potencia de RF y una red de adaptación de impedancias
  - Soporte de autenticación WEP y WPA/WPA2
- 16 pines GPIO (Entradas/Salidas de propósito general)
- SPI, I²C,
- Interfaz I²S con DMA (comparte pines con GPIO)
- Pines dedicados a UART, más una UART únicamente para transmisión que puede habilitarse a través del pin GPIO2
- 1 conversor ADC de 10-bit
- Voltaje: 3.3 V.
- Consumo de corriente: 10 µA – 170Ma

2.3.1 Procesador

El system on a chip (SoC) ESP9266EX usa un microcontrolador Tensilica Xtensa L106, que es un procesador de 32 bit con instrucciones de 16 bit.

Figura 2.14: Procesador ESP8266EX.

El SoC describe la tendencia cada vez más frecuente de usar tecnologías de fabricación que integran todos o gran parte de los módulos que componen un computador o cualquier otro sistema informático o electrónico en un único circuito integrado o chip.

El procesador funciona por defecto a 80 MHz, pero puede ir hasta 160 MHz, tiene ~ 80kB de DRAM (Data RAM), y ~ 35kB IRAM (Instruction RAM). La IRAM se carga en el arranque con lo que el usuario quiere mantener en el procesador, aunque el procesador puede ejecutar el código directamente fuera del flash externo a una velocidad más baja.

Tiene una arquitectura de Harvard, con lo cual la CPU puede tanto leer una instrucción como realizar un acceso a la memoria de datos al mismo tiempo, incluso sin una memoria caché.

En consecuencia, una arquitectura de computadores Harvard puede ser más rápida para un circuito complejo, debido a que la instrucción obtiene acceso a datos y no compite por una única vía de memoria.

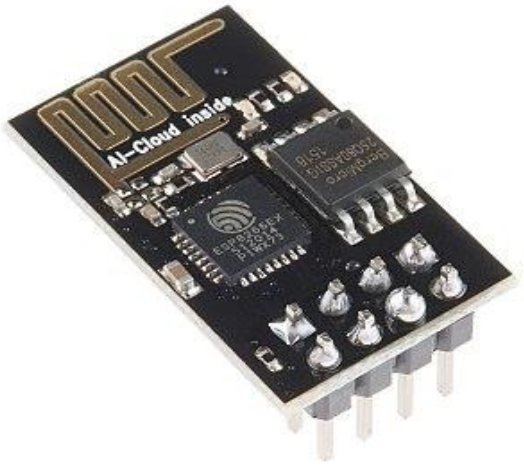

Figura 2.15: ESP8266 01.

2.3.2 Conjunto de comandos Hayes

El conjunto de comandos Hayes es un lenguaje desarrollado por la compañía Hayes Communications que prácticamente se convirtió en estándar abierto de comandos para configurar y parametrizar módems. Los caracteres «AT», que preceden a todos los comandos, significan «Atención», e hicieron que se conociera también a este conjunto de comandos como comandos AT.

Un aparato que implemente el conjunto de comandos Hayes se considera compatible Hayes. Parte del conjunto de comandos Hayes fue incluido por la ITU-T en el protocolo V.25ter, actual V.250. La adopción de este estándar hizo el desarrollo de controladores específicos para distintos módems superfluo.

A partir de la versión 3.x de Windows el sistema operativo contaba con una implementación de controlador para módems compatibles con Hayes. Sin embargo,

a partir de Windows 95 se desarrollaron controladores específicos para cada modem, así que la compatibilidad con Hayes dejó de ser importante y por esta razón cada vez menos módems la implementaron. Esto dificultó su uso en otros sistemas operativos, pues no resulta frecuente que haya controladores disponibles.

2.4 Domótica

Un sistema domótico es capaz de recoger información proveniente de unos sensores o entradas, procesarla y emitir órdenes a unos actuadores o salidas. El sistema puede acceder a redes exteriores de comunicación o información.

La domótica permite dar respuesta a los requerimientos que plantean estos cambios sociales y las nuevas tendencias de nuestra forma de vida, facilitando el diseño de casas y hogares más humanos, más personales, polifuncionales y flexibles.

El sector de la domótica ha evolucionado considerablemente en los últimos años, y en la actualidad ofrece una oferta más consolidada. Hoy en día, la domótica aporta soluciones dirigidas a todo tipo de viviendas, incluidas las construcciones de vivienda oficial protegida. Además, se ofrecen más funcionalidades por menos dinero, más variedad de producto, que, gracias a la evolución tecnológica, son más fáciles de usar y de instalar. En definitiva, la oferta es mejor y de mayor calidad, y su utilización es ahora más intuitiva y perfectamente manejable por cualquier usuario. Paralelamente, los instaladores de domótica han incrementado su nivel de formación y los modelos de implantación se han perfeccionado. Asimismo, los servicios posventa garantizan el perfecto mantenimiento de todos los sistemas. En definitiva, la domótica de hoy contribuye a aumentar la calidad de vida, hace más versátil la distribución de la casa, cambia las condiciones ambientales creando diferentes escenas predefinidas, y consigue que la vivienda sea más funcional al permitir desarrollar facetas domésticas, profesionales, y de ocio bajo un mismo techo.

La red de control del sistema domótico se integra con la red de energía eléctrica y se coordina con el resto de las redes con las que tenga relación: telefonía, televisión, y tecnologías de la información, cumpliendo con las reglas de instalación aplicables a cada una de ellas. Las distintas redes coexisten en la instalación de una vivienda o edificio. La instalación interior eléctrica y la red de control del sistema domótico están reguladas por el Reglamento Electrotécnico para Baja Tensión (REBT). En particular, la red de control del sistema domótico está regulada por la instrucción ITC-BT-51 Instalaciones de sistemas de automatización, gestión técnica de la energía y seguridad para viviendas y edificios.

2.4.1 ¿Qué aporta la Domótica?

La domótica contribuye a mejorar la calidad de vida del usuario:

- Facilitando el ahorro energético: gestiona inteligentemente la iluminación, climatización, agua caliente sanitaria, el riego, los electrodomésticos, etc., aprovechando mejor los recursos naturales, utilizando las tarifas horarias de menor coste, y reduciendo así, la factura energética. Además, mediante la monitorización de consumos, se obtiene la información necesaria para modificar los hábitos y aumentar el ahorro y la eficiencia.

- Fomentando la accesibilidad: facilita el manejo de los elementos del hogar a las personas con discapacidades de la forma que más se ajuste a sus necesidades, además de ofrecer servicios de teleasistencia para aquellos que lo necesiten.

- Aportando seguridad mediante la vigilancia automática de personas, animales y bienes, así como de incidencias y averías. Mediante controles de intrusión, cierre automático de todas las aberturas, simulación dinámica de presencia, fachadas dinámicas, cámaras de vigilancia, alarmas personales, y a través de alarmas técnicas que permiten detectar incendios, fugas de gas, inundaciones de agua, fallos del suministro eléctrico, etc.

- Convirtiendo la vivienda en un hogar más confortable a través de la gestión de dispositivos y actividades domésticas. La domótica permite abrir, cerrar, apagar, encender, regular... los electrodomésticos, la climatización, ventilación, iluminación natural y artificial, persianas, toldos, puertas, cortinas, riego, suministro de agua, gas, electricidad...).

- Garantizando las comunicaciones mediante el control y supervisión remoto de la vivienda a través de su teléfono, PC..., que permite la recepción de avisos de anomalías e información del funcionamiento de equipos e instalaciones. La instalación domótica permite la transmisión de voz y datos, incluyendo textos, imágenes, sonidos (multimedia) con redes locales (LAN) y compartiendo acceso a Internet; recursos e intercambio entre todos los dispositivos, acceso a nuevos servicios de telefonía IP, televisión digital, por cable, diagnóstico remoto, videoconferencias, teleasistencia...

Figura 2.16: Domótica.

CAPÍTULO 3
ESTADO DEL ARTE

3.1 Estado del Arte

Un monitor de energía para el hogar es un tipo de vatihorímetro o contador eléctrico que proporciona información en tiempo real del consumo eléctrico de manera comprensible para cualquier usuario sin necesidad de conocimientos técnicos. Estos dispositivos también pueden mostrar al usuario el coste de la energía consumida y estimaciones de las emisiones de gases de efecto invernadero. Algunos estudios muestran una reducción del consumo de energía en el hogar de 4-15% gracias al uso de monitores de energía.

3.1.1 Smappee

Figura 3.1: Smappee.

El monitor Smappee es un pequeño dispositivo diseñado para identificar y medir el consumo de energía eléctrica de los electrodomésticos del hogar, su aplicación permite conocer en tiempo real la cantidad de energía que gasta cada uno de ellos, contribuyendo así a un consumo más eficiente de la energía en el hogar y ayudando a mejorar el medio ambiente. Es simple y seguro de instalar y no requiere ningún cambio en el sistema eléctrico.

El monitor Smappee no entra en contacto directo con la corriente eléctrica simplemente se acopla a los cables de la caja de fusibles o cuadro interruptor del hogar mediante la abrazadera del sensor, aunque es fácil de instalar y aparentemente no supone ningún riesgo se recomienda que su instalación la realice un electricista cualificado.

Una vez instalado el monitor empezará a identificar cada uno de los electrodomésticos por sus señales o huellas energéticas, después de unas horas enviará los primeros datos a través de la red Wi-Fi a la aplicación gratuita Smappee.

El sistema de reconocimiento de dispositivos de Smappee se basa en una innovadora tecnología llamada NILM (Non-Intrusive Load Monitoring, que significa “monitorización no intrusiva de la carga”). ¿Cómo funciona? Cada aparato consume corriente eléctrica de una manera característica. Podría decirse que cada uno tiene su propia firma eléctrica. Cada vez que se enciende o apaga un dispositivo, Smappee detecta esta firma eléctrica y reconoce el encendido o apagado en forma de suceso registrable. Con el uso frecuente de los dispositivos, el monitor Smappee aprende a identificar la firma eléctrica de cada uno, si bien hace falta un poco de tiempo para aprender a asociar con precisión los sucesos con el aparato del que se trate. La propia palabra ya lo dice: reconocimiento.

3.1.2 Sense

Figura 3.2: Sense.

Sense Energy Monitor proporciona visibilidad del uso de electricidad en tiempo real, incluso a nivel de dispositivo (aunque la precisión en el nivel del dispositivo no es perfecta).

El dispositivo viene con una aplicación de monitoreo de teléfonos inteligentes que también proporciona notificaciones. La compañía utiliza un algoritmo de detección de aprendizaje automático avanzado para distinguir los dispositivos individuales. El algoritmo analiza las tendencias actuales (en comparación con la potencia pura) y las compara con los patrones actuales de dispositivos comunes (por ejemplo, una tostadora).

La característica principal del sistema de monitoreo es la pantalla "Ahora" que muestra su consumo de electricidad actual y las burbujas para cada dispositivo que usa electricidad. Las burbujas se adaptan al uso de energía y ajustarán las horas extras ya que Sense recoge más dispositivos. Puede explorar en cualquiera de las burbujas para ver cuánta potencia ha usado ese dispositivo en horas extra.

Sense Energy Monitor fue creado por el equipo que trajo la tecnología avanzada de reconocimiento de voz a teléfonos móviles, incluida la búsqueda por voz, mensajes de texto y dictado, por lo que se espera ver mejoras de productos más emocionantes por parte de la compañía.

3.1.3 Eyedro

Figura 3.3: Eyedro.

El Eyedro monitorea el uso de electricidad en tiempo real y es una alternativa de menor costo si no necesita todas las características de otros modelos. Eyedro también ofrece un paquete para empresas.

La unidad viene con dos sensores de 200 A que se conectan directamente al panel eléctrico. El dispositivo requiere una conexión directa de Ethernet que conecta dos dispositivos separados. Los consumidores informan que la unidad es muy fácil de instalar.

Aunque la compañía actualmente no ofrece una aplicación de teléfono inteligente, la monitorización en línea es amigable para dispositivos móviles.

3.1.4 Neurio

Figura 3.4: Neurio.

El Neurio Home Energy Monitor es un sistema completo de monitoreo de electricidad que le indica dónde y cuándo se consume energía en su hogar.

Después de instalar el sensor único, puede controlar el pulso de energía de su hogar con las aplicaciones iOS, Android y web fáciles de usar para rastrear su consumo de energía en tiempo real. Neurio estima que la aplicación puede ayudar

a los propietarios de viviendas a ahorrar hasta un 20% de energía cada año y proporciona consejos inteligentes de ahorro de energía, comparaciones de viviendas y uso histórico. El dispositivo se conecta al panel eléctrico de la casa y la instalación no requiere ningún corte de cable y toma alrededor de 15 minutos.

El Neurio está instalado en más de 100,000 hogares y es el modelo más popular. También se puede comprar un kit de expansión para monitorear la producción solar y la medición neta.

CAPÍTULO 4
MATERIALES Y MÉTODOS

4.1 Corriente Alterna

Se le denomina corriente alterna (CA o AC en inglés) a la corriente eléctrica que cambia repetidamente de polaridad. esto es, su voltaje instantáneo va cambiando en el tiempo desde 0 a un máximo positivo, vuelve a cero y continúa hasta otro máximo negativo y así sucesivamente. La corriente alterna más comúnmente utilizada, cambia sus valores instantáneos de acuerdo con la función trigonométrica seno, de ahí su denominación de corriente alterna sinusoidal.

Básicamente lo que sucede es que la intensidad varía entre un máximo y un mínimo de forma periódica. El máximo es un número positivo y el mínimo es un número negativo. La frecuencia con la que cambia es rápida, entre 50 o 60 veces por segundo que se mide en Hercios**.**

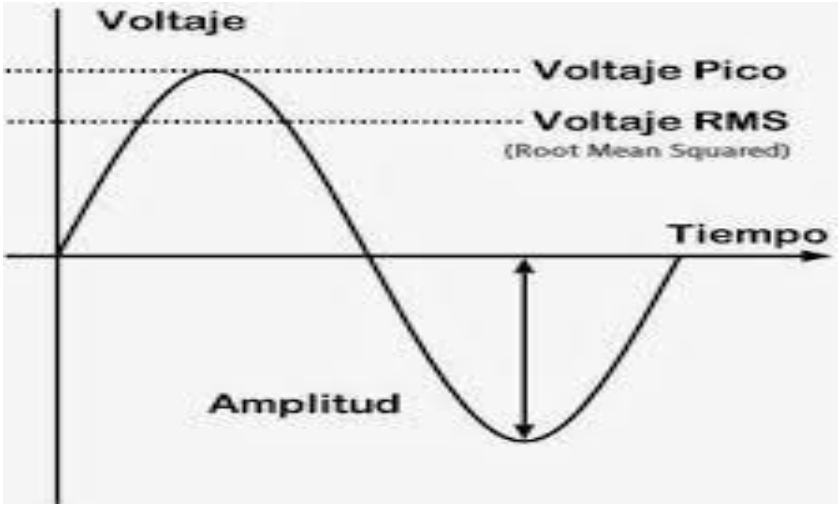

Figura 4.1: Onda sinusoidal de voltaje.

El suministro comercial de energía eléctrica utilizado de manera generalizada en nuestros días se efectúa en corriente alterna.

4.1.1 Consumo eléctrico de electrodomésticos

La unidad de energía en la que se factura la energía por las compañías proveedoras de esta es el KWH, que equivale a la potencia en Watts que un electrodoméstico consume si estuviese funcionando una hora.

4.2 Sensor SCT-013-030

Para la realización de este proyecto se seleccionó el SCT-013-030 ya que este modelo permite realizar medidas en un rango de 30 A que se considera es necesario para los consumos que puedan existir en un hogar convencional. Además, cuenta con una resistencia de carga interna, entregando una salida de voltaje. La relación es de 30 A/1V.

La salida de este sensor es una señal alterna, cuyos valores no están dentro del rango de las entradas analógicas (0 a +5V) del Arduino, si bien el rango del sensor puede ser inferior, la parte negativa de la señal podría afectar la placa Arduino.

Una manera de trabajar con esta señal es rectificándola. Para rectificar no se pueden usar diodos, puesto que la caída de voltaje en el diodo es muy grande en comparación al voltaje de la señal. Para esto se usará un operacional, configurado en un seguidor de voltaje, se usará el operacional LM358, que trabaja con polaridad positiva, de esta forma se eliminará la parte negativa de la señal, si bien no es un rectificador de onda completa, pero con una rectificación de media onda se puede trabajar.

El LM358 se alimenta con 5V, se satura con 3.5V aproximadamente, motivo por el cual no se puede amplificar hasta 5V, pero si trabaja con Arduino no se necesita alcanzar los 5V, se puede trabajar con la referencia interna de 1.1V y de esta forma aprovechar en el rango completo de la lectura a analógica.

4.2.1 Conexión

Para la conexión del sensor al circuito, este tiene como pines un conector mini Jack estéreo de audio. Con el objetivo de manipular sin cortar el cable se utilizó un conector hembra del mini Jack, tomando de ahí las salidas correspondientes del sensor. El pin central esta desconectado y los otros 2 son la salida del sensor como se puede apreciar en la siguiente figura.

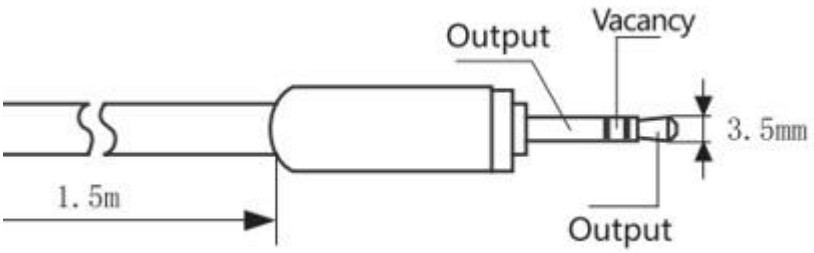

Figura 4.2: Conexión de la salida mini Jack estéreo del SCT-013.

Y las conexiones del circuito con el acondicionador de señal son las mostradas en la siguiente figura. Se puentean los pines 1 y 2 para retroalimentación, el pin 3 se conecta la salida del sensor, el pin cuatro se manda a tierra, así como una de las salidas del sensor y el pin 8 se alimenta con 5V positivos.

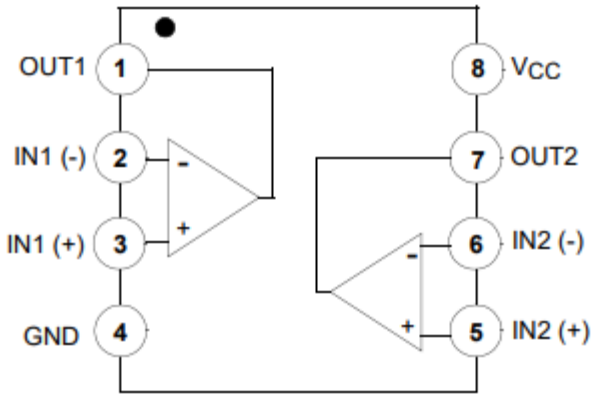

Figura 4.3: Pines del operacional LM358.

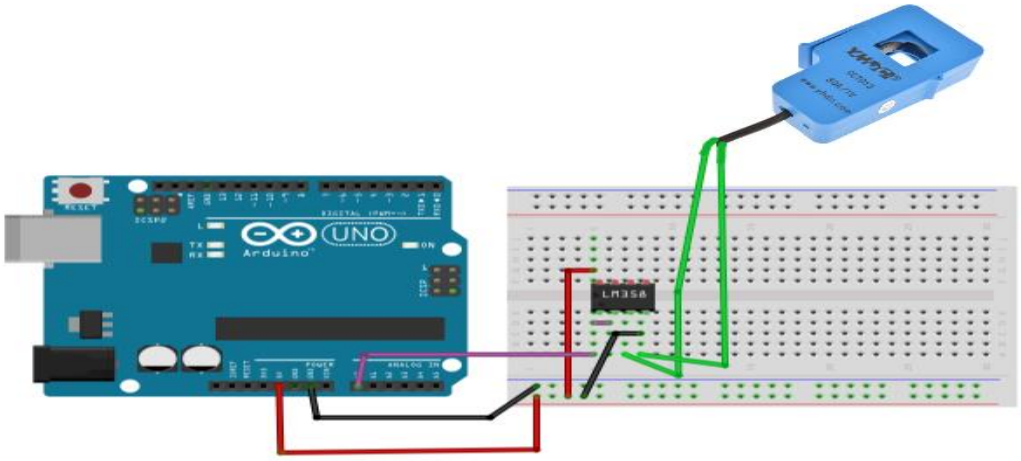

Figura 4.4: Conexión del SCT-013 al Arduino.

4.2.2 Rectificación de media onda

Para empezar a hacer el procesamiento adecuado de la señal del sensor se comprobará primero que la rectificación de media onda al usar el operacional sea correcta. Haciendo lectura del puerto analógico, así como su voltaje, se multiplica esa lectura por la relación corriente/voltaje del sensor para obtener la corriente, obteniendo la siguiente lectura usándose una licuadora que consume 400W.

```
void setup() {
  Serial.begin(9600);
  analogReference(INTERNAL);
}

void loop() {
  float voltajeSensor = analogRead(A0) * (1.1 / 1023.0); //voltaje del sensor
  float I=voltajeSensor*30.0; //I=VoltajeSensor*(30A/1V)
  Serial.println(I);         //envio por el puerto serie
  delay(300);

}
```

Figura 4.5: Código para leer voltaje del sensor.

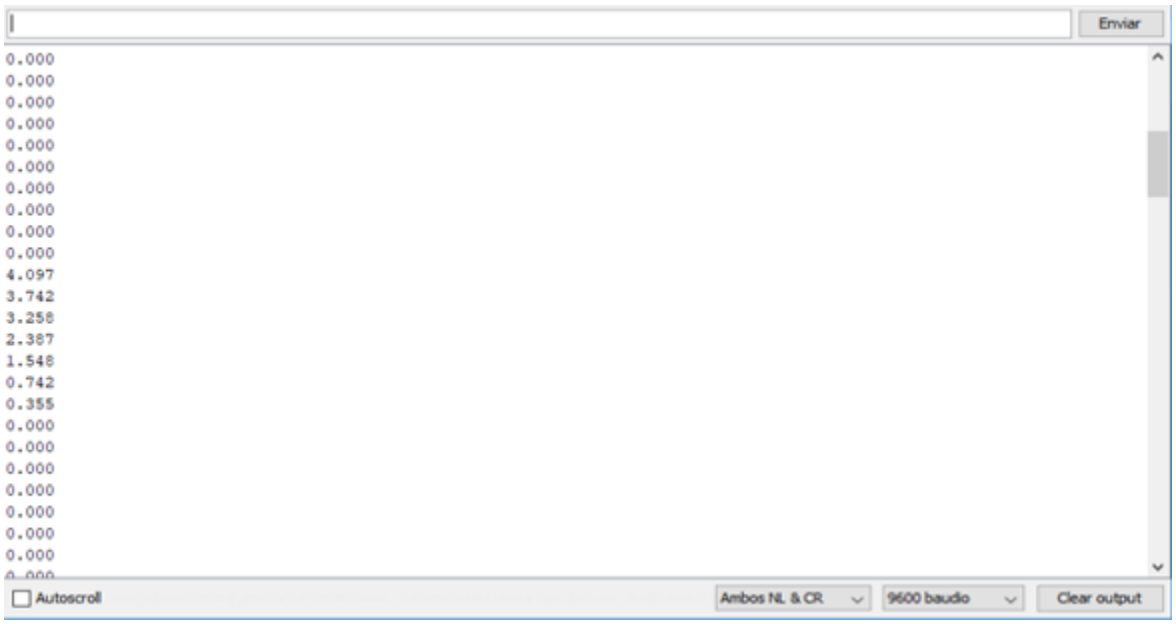

Figura 4.6: Medición por puerto serial de la señal del SCT-013 rectificada.

Usando el Serial Plotter del IDE de Arduino, permite observar el comportamiento de la onda generando semiciclos sinusoidales.

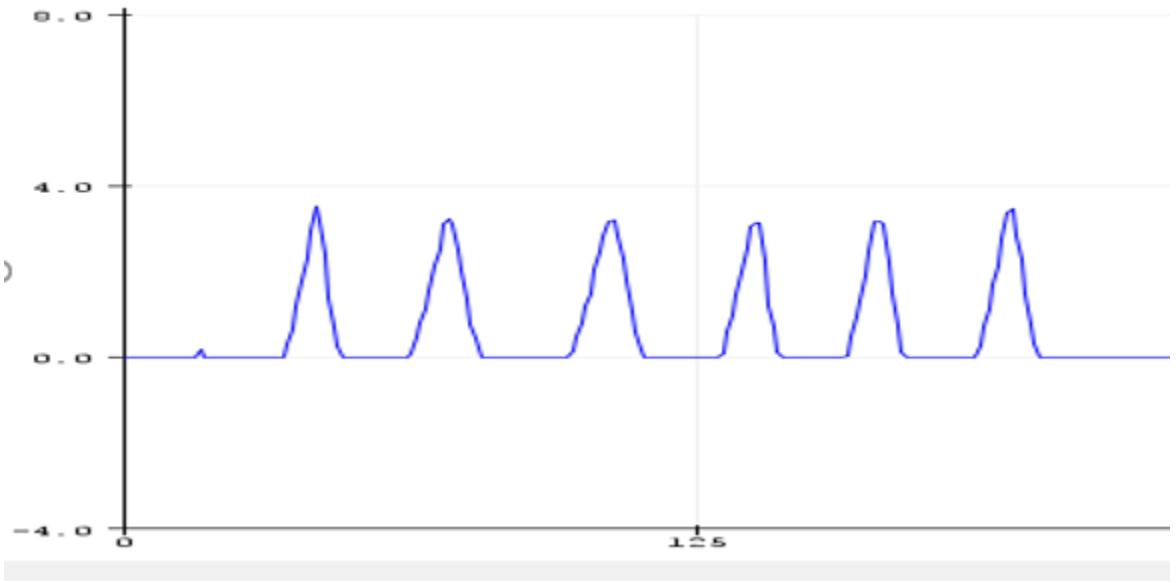

Figura 4.7: Medición por Serial Plotter de la señal del SCT-013 rectificada.

4.2.3 Realizando mediciones de corriente eficaz y potencia

Se llama valor eficaz de una corriente alterna, al valor que tendría una corriente continua que produjera la misma potencia que dicha corriente alterna, al aplicarla sobre una misma resistencia.

Para calcular la corriente eficaz se utiliza la siguiente formula

(4.1)

$$Irms = \sqrt{\frac{1}{t}\int_0^T i^2 \, dt}$$

Como el microcontrolador toma muestras en ciertos tiempos de la señal se tiene que transformar la formula a tiempo discreto, la cual quedaría de la siguiente forma:

(4.2)

$$Irms = \sqrt{\frac{1}{N}\sum_{n=0}^{N} i_{n^2}}$$

Siendo “N” el número de muestras en un periodo. En la programación se duplica el valor de la sumatoria para compensar para compensar el semiciclo negativo que se anuló en la rectificación de onda.

El tiempo que dura la toma de muestras para el cálculo de la corriente eficaz es de 500ms que representan 30 ciclos de una señal de 60 HZ.

Obteniendo la corriente eficaz y sabiendo que el voltaje usado en el país de México es de 120 V, se multiplica estas dos variables para obtener la potencia consumida.

(4.3)

$$P = I \, x \, V$$

```
float get_corriente()
{
  float voltajeSensor;
  float corriente=0;
  float Sumatoria=0;
  long tiempo=millis();
  int N=0;
  while(millis()-tiempo<500)   //Tiempo de muestreo de 0.5 segundos
    voltajeSensor = analogRead(A0) * (1.1 / 1023.0);    //voltaje del sensor
    corriente=voltajeSensor*20;
    Sumatoria=Sumatoria+sq(corriente);  //Sumatoria de Cuadrados
    N=N+1;
    delay(1);
  }
  Sumatoria=Sumatoria*4;    //Compensacion semiciclos negativos.
  corriente=sqrt((Sumatoria)/N);   //ecuación de corriente eficaz
  return(corriente);
}
```

Figura 4.8: Función para el cálculo de la corriente y potencia eficaz.

4.3 Uso del Módulo ESP8266 01

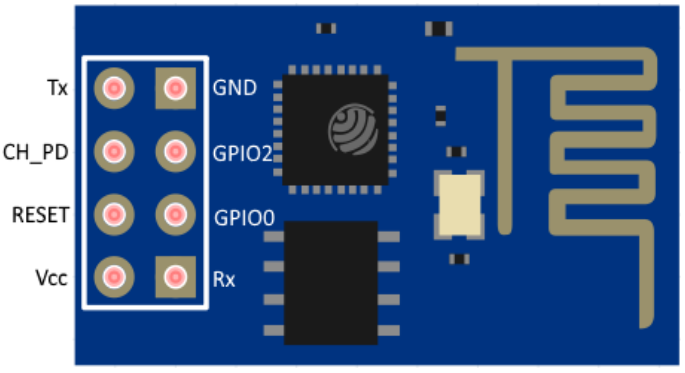

Figura 4.9: Pines del módulo ESP8266 01.

La conexión de este módulo es realmente sencilla, conectando a 3.3 V del Arduino, los pines TX y RX a sus respectivos puertos en el 3 y 2 de la placa del microcontrolador, como se puede observar en la siguiente tabla y figura:

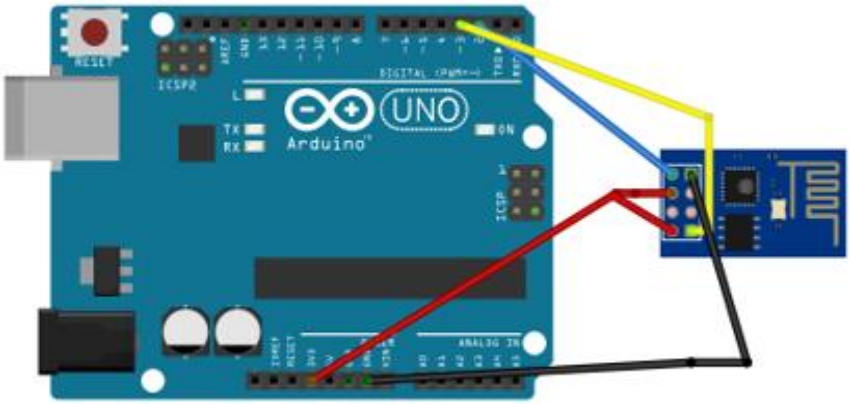

Figura 4.10: Conexión al Arduino del módulo ESP8266 01.

La comunicación con el ESP01 con el firmware por defecto se realiza a través de comandos AT, que no son más que comandos de texto enviados por Serial.

Se pueden enviar estos comandos por un conversor USB-TTL (FT232, CH340G o CP2102) o, en este caso, se usará Arduino y Software serial como adaptador utilizando el siguiente código:

```
#include <SoftwareSerial.h>
SoftwareSerial ESP(3, 2); // RX | TX

void setup()
  {  Serial.begin(9600);
     ESP.begin(9600);
  }

void loop()
  {
     if (ESP.available())
         { char c = ESP.read() ;
           Serial.print(c);
         }

     if (Serial.available())
         {  char c = Serial.read();
            ESP.print(c);
         }
   }
```

Figura 4.11: Código para lectura y escritura con el módulo ESP8266.

4.3.1 Comprobando conexión y velocidad de comunicación serial

Al conectar el módulo y cargar el programa para lectura y escritura se usarán los comandos AT para su configuración, como se puede ver las siguientes figuras:

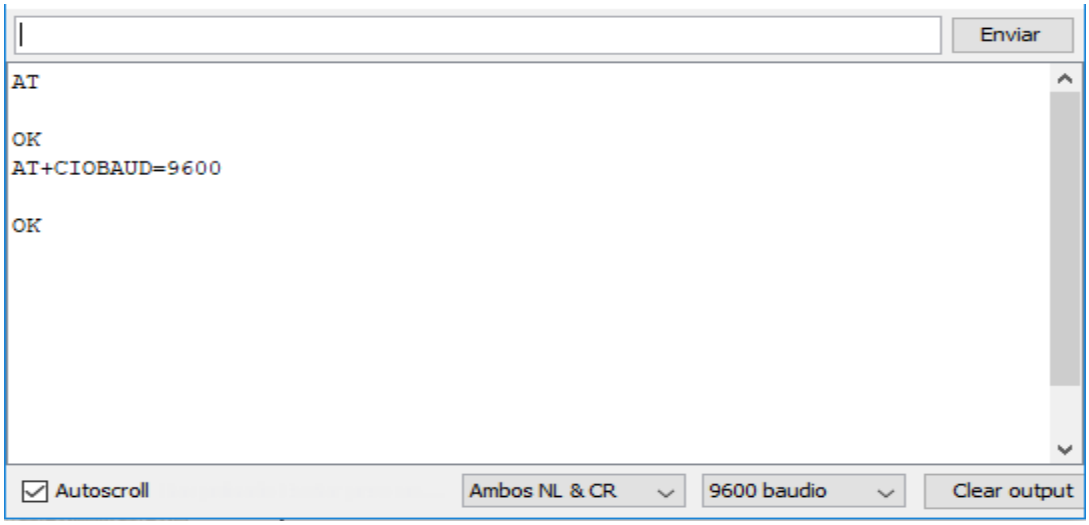

Figura 4.12: Configuración de la velocidad de comunicación Serial.

4.3.2 Habilitando el modo de trabajo dual y visualizando las redes

Habiendo comprobado las conexiones se escribe el comando “AT” para verificar la comunicación, si se recibe como respuesta “OK”, la comunicación es correcta, en caso contrario o si responde con dígitos sin sentido habrá que escribir “AT+CIOBAUD=9600” para cambiar la velocidad de la comunicación serial ya que suele venir por default la velocidad de “115200”.

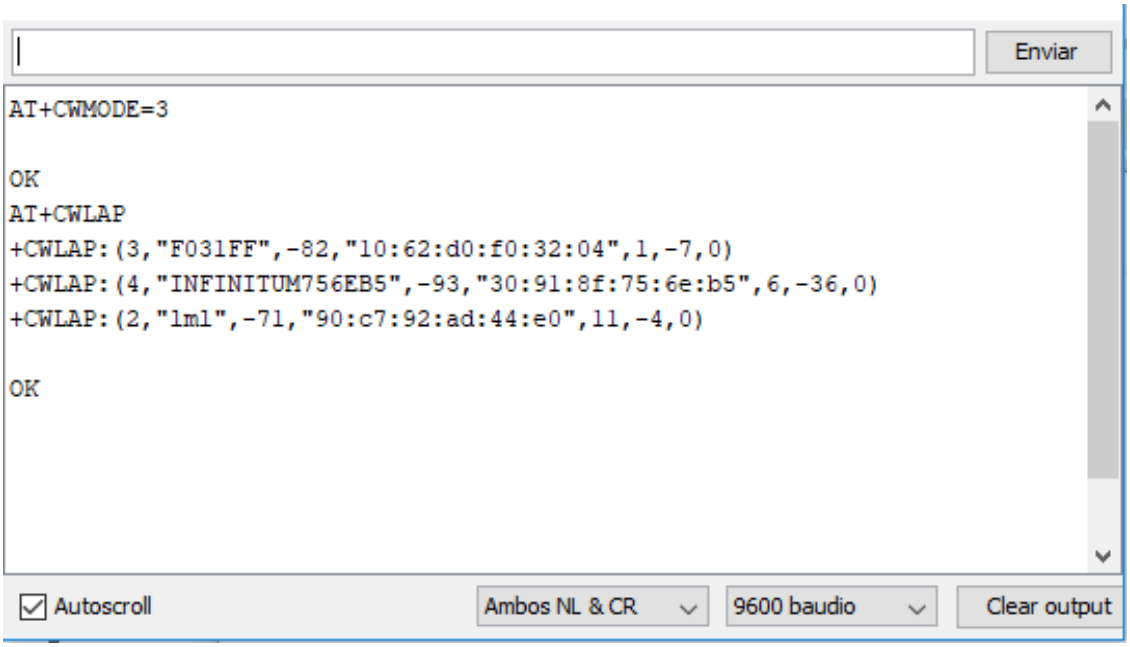

Figura 4.13: Modo de trabajo y redes inalámbricas.

4.3.3 Conectando a red y habilitando conexiones múltiples

Con el comando “AT+CWMODE=3” se configura el modo dual de trabajo del módulo, siendo cliente y huésped estas opciones. El comando “AT+CWLAP” permite observar las redes inalámbricas que detecta, con valores que informan sobre su intensidad y tipo de seguridad.

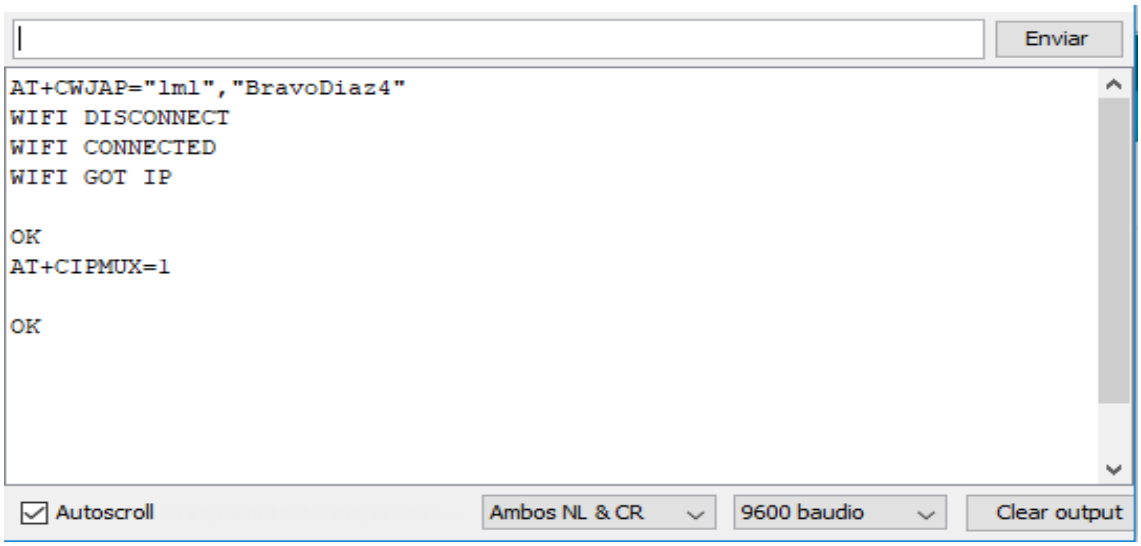

Figura 4.14: Conectando a red inalámbrica y habilitando conexiones múltiples.

4.3.4 Habilitando servidor y obteniendo IP

Al comando “AT+CWLAP” usado anteriormente se le agrega entre comillas el nombre de la red inalámbrica a la que se desee conectar y su contraseña separados por coma para iniciar la conexión. Dará como respuesta que se conecto y que ha obtenido la IP. Con el comando “AT+CIPMUX=1” para habilitar las conexiones múltiples.

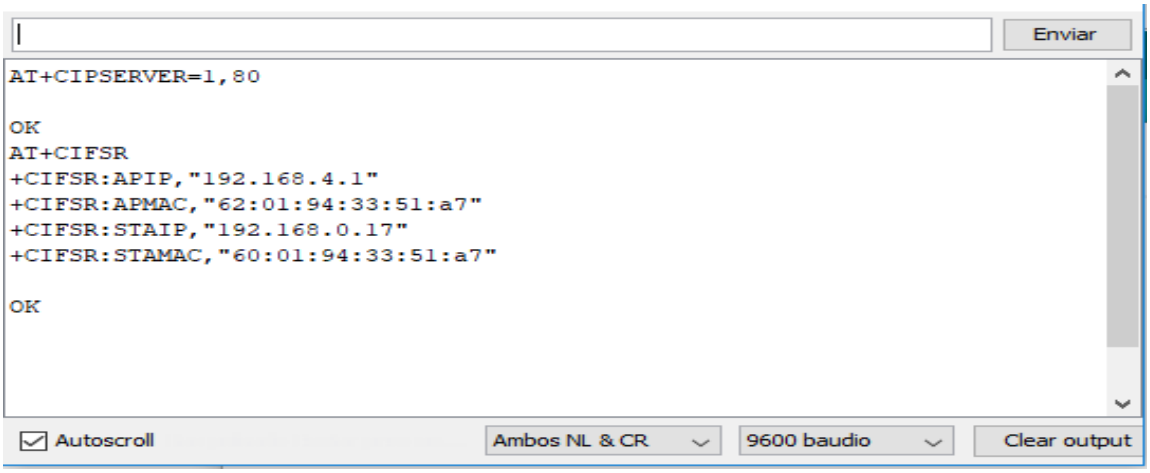

Figura 4.15: Inicializando él servidor y obteniendo IP.

Para inicializar el servidor se usa el comando “AT+CIPSERVER=1,80” siendo el número de servicio 1 en el puerto 80. Para conseguir la IP que asigno el módulo se usa el comando “AT+CIFSR” que es este caso resulta ser 192.168.4.1.

Por último, para comprobar que se tiene comunicación con el servidor local que se ha creado se conecta el dispositivo móvil a la red del módulo ESP8266 01 que por defecto se llama “AI-THINKER” y en el navegador se introducirá la IP que genero el módulo, generando una respuesta de comunicación en el puerto serie.

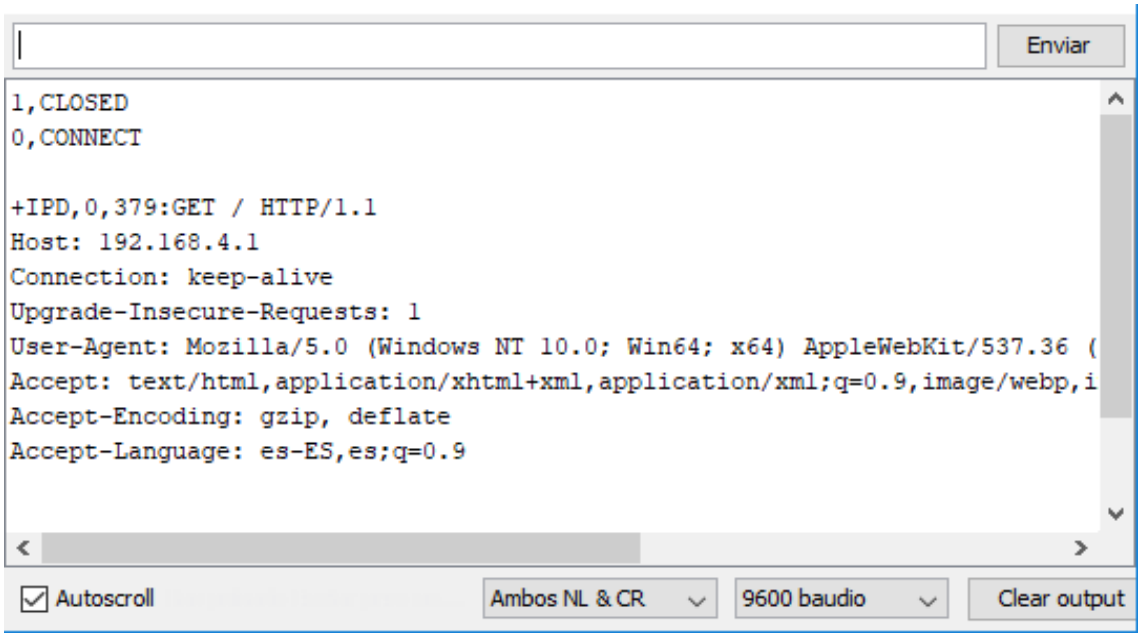

Figura 4.16: Respuesta al ingresar a la IP de la red local.

CAPÍTULO 5
EXPERIMENTOS Y RESULTADOS

5.1 Pruebas y calibración del sensor SCT 013 030

En el hogar se encuentran diversos tipos de electrodomésticos con rangos muy diferentes de consumo energético siendo desde 60W con los focos de luz incandescente hasta 1500 W con un microondas o hasta 1800W con aspiradoras.

El sensor de SCT 013 030 no tiene resistencia de carga propia por lo que utiliza la resistencia de referencia interna que contiene el microcontrolador para hacer las mediciones correspondientes del sensor. Como esta resistencia y la relación corriente/voltaje del sensor son fijas hay lecturas distantes de ser correctas en algunos rangos de potencia.

Para resolver esta diferencia se emplea un factor de calibración en el código que habiendo sido probado y comprobado con un multímetro los rangos que pueden producirse en el hogar se condiciono para los cambios de potencia y así se pueda ofrecer lecturas confiables para todo tipo de consumo eléctrico.

Enseguida se muestran las lecturas de diferentes tipos de aparatos eléctricos en el hogar.

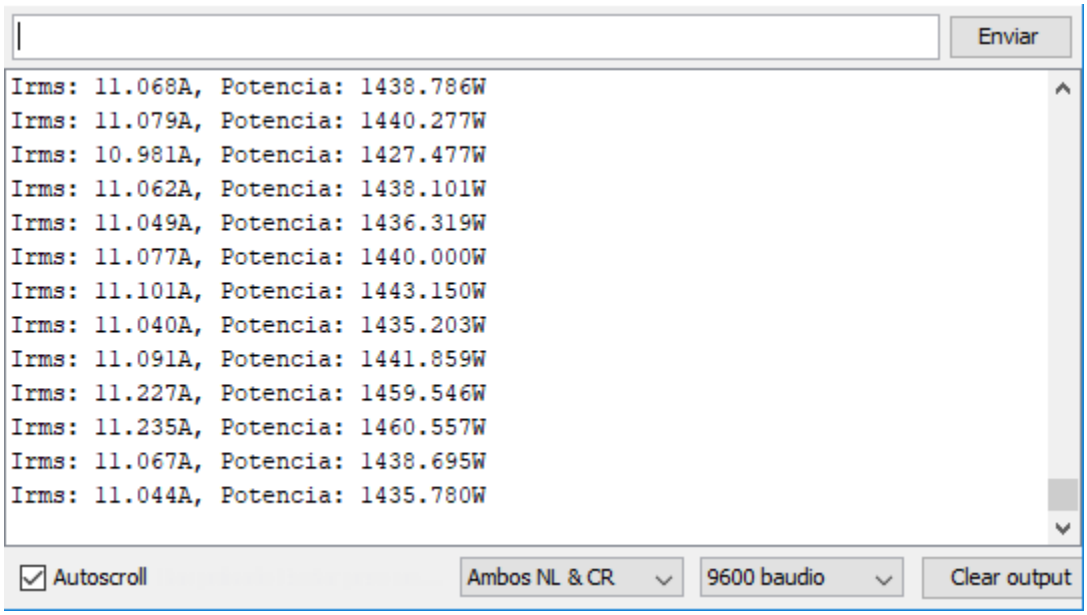

Figura 5.1: Consumo de un microondas de 1500W aproximado.

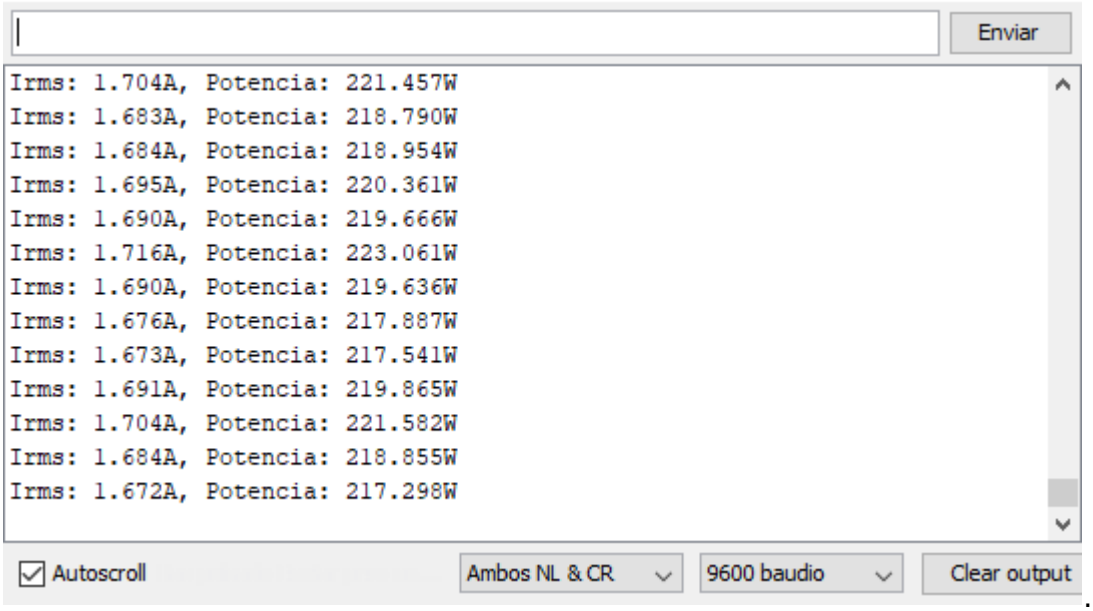

Figura 5.2: Consumo de una licuadora de 300W aproximado.

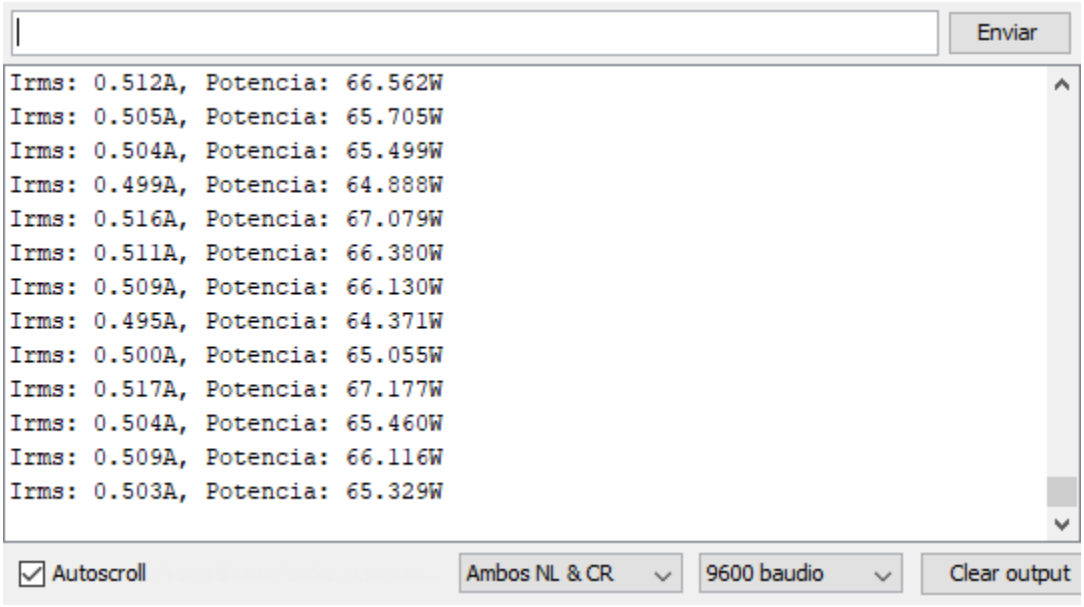

Figura 5.3: Consumo de un foco incandescente de 60W aproximado.

5.2 Estructura y diseño de página web en ESP8266 01

Una página web necesita de código HTML sigla en inglés de *HyperText Markup Language* (lenguaje de marcas de hipertexto) para funcionar y dar forma a su contenido como texto, imágenes, videos, juegos y otros.

```
void servidor(void)
    {

      escribir("<!DOCTYPE HTML>");                //Definicion de tipo de dcumento
      escribir("<html>");                                          //Inicializacion de la sintaxis HTMI
      escribir("<head><title>Monitor de Energia</title>");    //nombre de la pestaña que llevara la pagina
      escribir("<meta http-equiv=\"refresh\" content=\"5\"></head>");    //tiempo para refrescar la pagina web
      escribir("<body><h1> <FONT SIZE=\"5\" COLOR=\"red\"> Monitor SCT-013 </h1>");
      escribir(" <BODY BGCOLOR=\"black\"> ");                //Color de fondo


          escribir("<FONT FACE=\"Arial\" SIZE=\"5\" COLOR=\"blue\"> El consumo es </FONT>"); //escribe y camb
          escribir(String(P));       //imprimime la variable
          escribir(" Watts <br /><br />");    //Tpipo de Variable
```

Figura 5.4: Envió de código HTML para la página web.

En el módulo se escribe código HTML para diseño y visualización de la potencia, como estructura se tiene la siguiente:

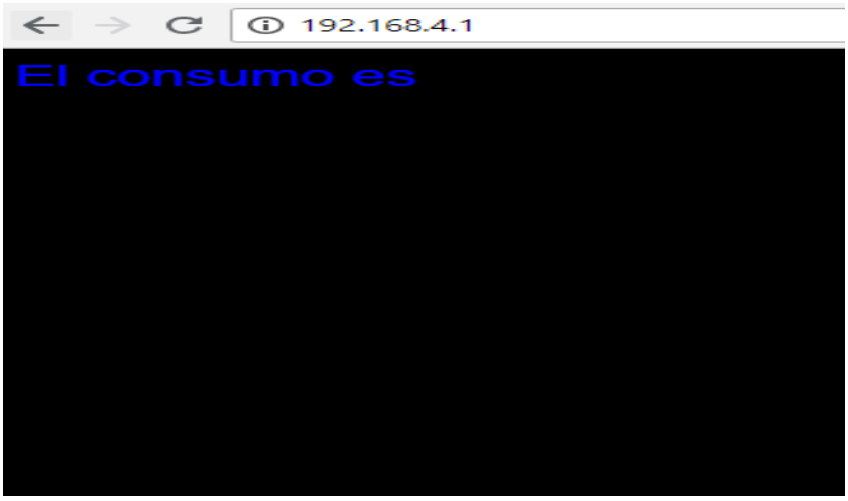

Figura 5.5: Estructura de la página web para el consumo eléctrico.

5.3 Integración del SCT 013 y el módulo ESP8266

Habiendo probado y calibrado el sensor SCT 013 para la medición de consumo eléctrico, el siguiente paso es integrarlo a la programación del ESP8266 para su envió de datos por red local y de esta manera lograr visualizar sus mediciones el cualquier dispositivo móvil.

Integrando los 2 programas en uno solo se puede comprobar su funcionamiento empezando a medir diferentes aparatos eléctricos y el consumo total del hogar, como pruebas de esto, se tienen las siguientes figuras.

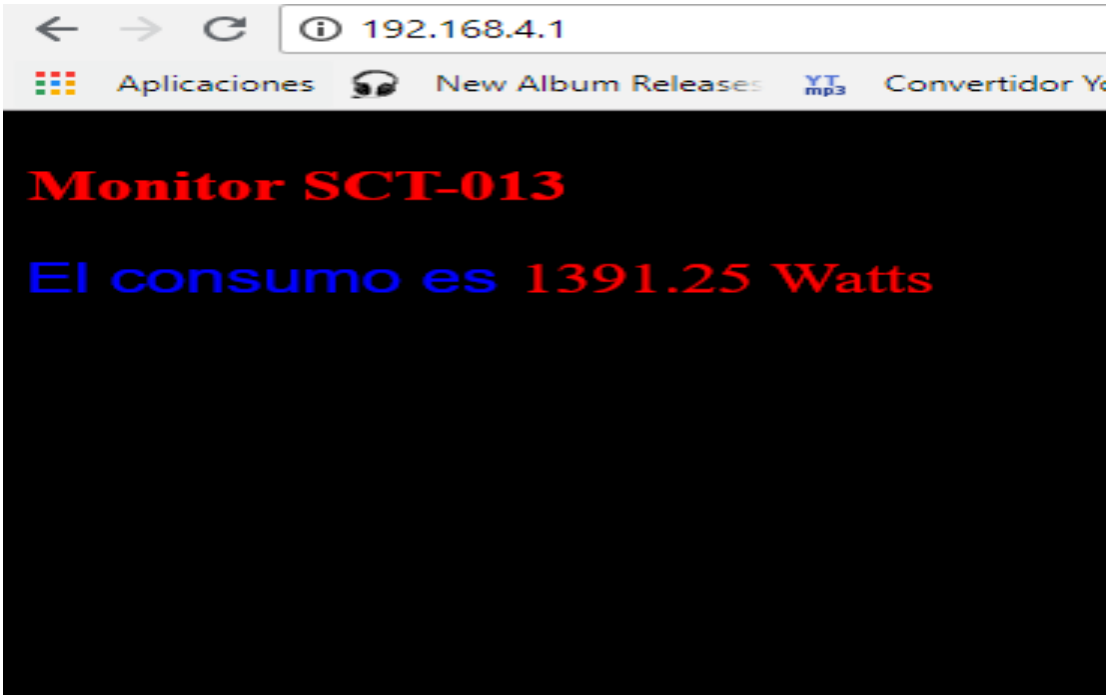

Figura 5.6: Lectura del consumo eléctrico de un microondas en PC portátil.

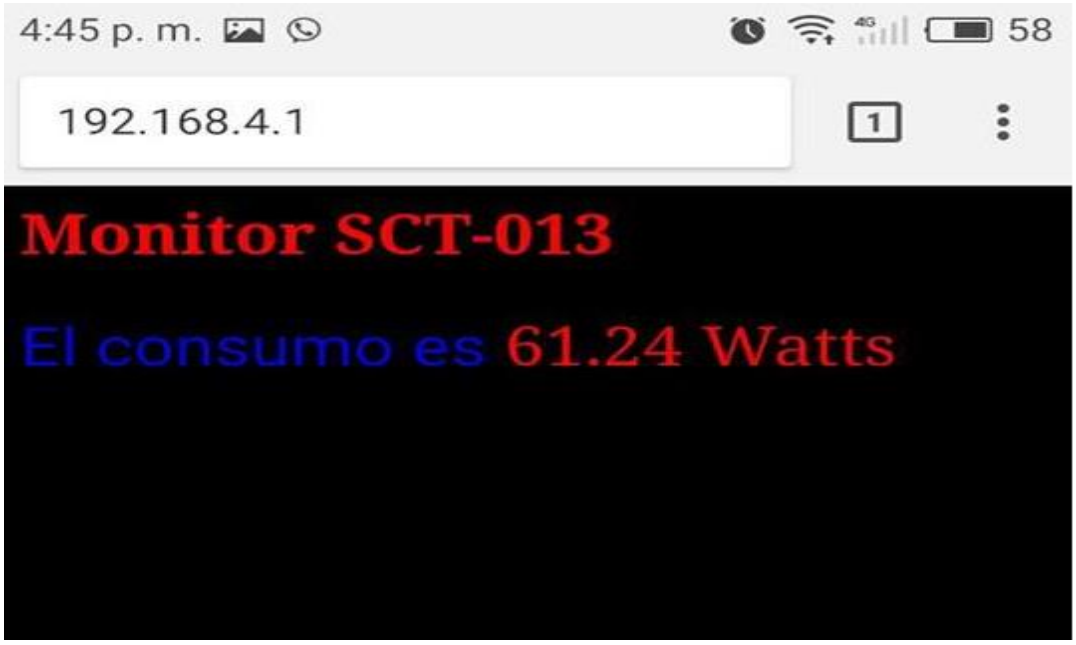

Figura 5.7: Lectura del consumo de un foco desde un smartphone.

La página web nos muestra el consumo en tiempo real en Watts, refrescando la lectura cada 5 segundos, esto permitiendo monitorear en cualquier momento con nuestro dispositivo móvil, sea tableta, smartphone o computadora portátil mientras se tenga acceso a la red del módulo ESP8266.

CAPÍTULO 6
ANÁLISIS DE RESULTADOS Y CONCLUSIONES

6.1 Análisis de resultados

Los resultados de este trabajo de investigación son la codificación y calibración del sensor SCT 013 con el cual se logra hacer medidas del consumo eléctrico para determinado hogar o aparato.

La configuración del módulo wifi ESP8266 01 por medio de comandos AT y comunicación serial a través del microcontrolador, teniendo como objetivo él envió de datos recopilados por el sensor SCT 013, diseñando y habilitando una página web de la que se puede acceder desde cualquier dispositivo móvil dentro de la red local.

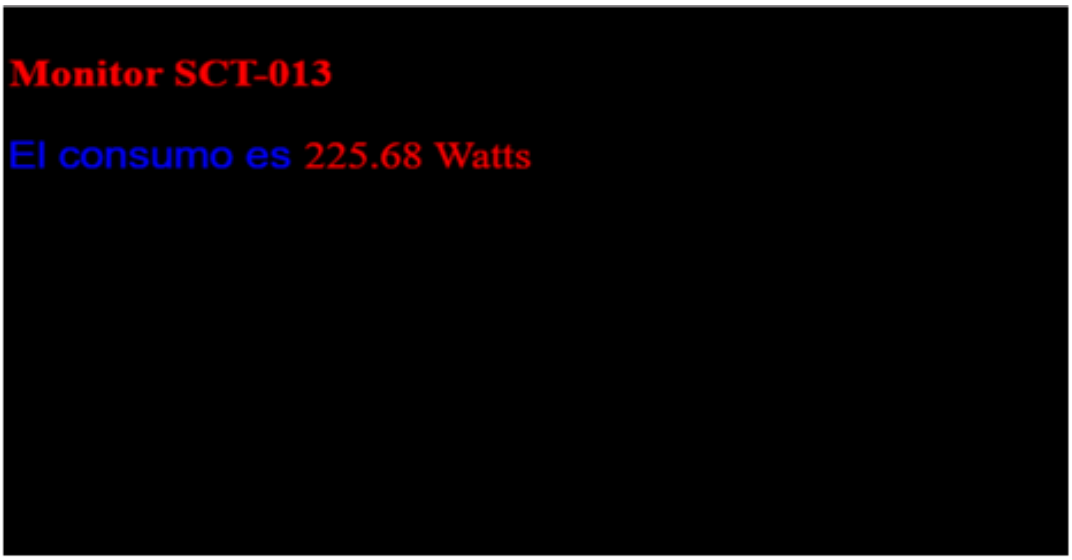

Figura 6.1: Lectura en tiempo real del hogar.

Figura 6.2: Uso del sensor SCT 013 en panel eléctrico.

En las figuras 6.1 y 6.2 se puede observar la lectura en tiempo real del consumo del hogar y la instalación simple del sensor en el panel eléctrico sin necesidad de intervenir físicamente algún cable para su lectura, permitiendo un manejo fácil para cualquier usuario de este sensor, logrando pasar del panel eléctrico a cualquier aparato eléctrico en particular.

6.2 Trabajos a futuro

Este proyecto de investigación esta en su primera fase de desarrollo por lo que hay diferentes puntos que se pueden mejorar, añadiendo el avance tecnológico que se produce y producirá en los años venideros. Parte de esos puntos son los siguientes:

- Envió y consulta de datos a un servidor publico

Para que la lectura de esta información pueda ser consultada desde cualquier parte del mundo que se tenga acceso a internet, uno de los pasos a trabajar en el futuro es cambiar de servidor local a un servidor público, sea gratuito o de paga, según el acceso con el que se quiera contar, omitiendo de esta manera el estar cerca de la red local para conocer el consumo eléctrico del hogar.

- Mejora de la página web y creación de aplicación móvil

Como modo de mejorar la interacción del usuario con esta información se puede mejorar el diseño de la pagina web, proporcionando una interfaz mas amigable, asi como la creación de app móvil para los diferentes sistemas operativos que existen en el mercado.

- Uso de interruptores remotos

El diseño y construcción de interruptores remotos que puedan accionarse desde la aplicación móvil, proporcionando así al usuario un mejor control sobre sus dispositivos eléctricos y regular el consumo de su hogar sin la necesidad de estar presente.

- Historial de consumo eléctrico

La creación de un historial de consumo eléctrico sería de gran uso para el usuario permitiéndole conocer que aparatos y en que temporadas del año sus gastos aumentan o disminuyes para hacer alguna acción que le ayude a llevar una vida sustentable ahorrando dinero en el camino.

- Creación de alertas

Un punto que mejorar en el futuro es la creación de alertas configurables para el usuario, el cual puede plantearse metas de consumo conociendo ya su historial, notificando que está cerca de sobrepasar limites, etc.

6.3 Conclusiones

El uso de las herramientas de software libre en la domótica es una actividad que deberá seguir en aumento, logrando bajar costos y estando en mejora continua.

Al término de este proyecto de investigación se logra la integración del sensor SCT 013 y el modulo wifi ESP8266 01, permitiendo la medición del consumo eléctrico en el hogar sin la necesidad de intervenir físicamente el cableado eléctrico. Con una programación sencilla y envió de comandos por puerto serie, se habilito un servidor en red local para que permite leer los datos de forma remota desde cualquier dispositivo móvil con conexión a esta red.

Las tendencias en la actualidad debido a los problemas ambientales que se viven son precisamente el disminuir nuestra huella ecológica. Este proyecto, buscando apoyar esta tendencia, logra su cometido de informar al usuario sobre su consumo eléctrico, concientizando y viendo los aparatos vulnerables a gastar mas de lo debido, añadiendo a esto un ahorra significativo en su pago de tarifa eléctrica.

Presentando así una alternativa de bajo costo a los diferentes monitores que hay en el mercado además de la generación de conocimiento en esta área para trabajos venideros de estudiantes o investigadores que quieran trabajar en el área de la domótica.

REFERENCIAS BIBLIOGRÁFICAS

[1] Amezcua, J. C. (2013). *Sistema de Monitoreo Inalambrico de Consumo Electrico con Microcontrolador.* Cd. Guzman, Jalisco.

[2] Arduino. (2005). *Arduino*. Recuperado el 12 de Noviembre de 2017, de http://arduino.cl/que-es-arduino/

[3] Artero, Ó. T. (2013). *ARDUINO - Curso práctico de formación.* Mexico, DF.: Alfaomega.

[4] Calaza, G. T. (2015). *Taller de Arduino - Un enfoque práctico para principiantes.* Alfaomega, Marcombo.

[5] Ceja, J., Rentería, R., Ruelas, R., & Ochoa, G. (2017). Módulo ESP8266 y sus aplicaciones en el internet de las cosas. *Revista de Ingeniería eléctrica*.

[6] Schwartz, M. (2016). *Internet of the Things with ESP8266.* Birmingham, UK.: Packt Publishing.

[7] Serway, R. A., & John W. Jewett, J. (2015). *Electricidad y Magnetismo.* CENGAGE LEARNING.

[8] *Soraya Paniagua*. Internet de las cosas. Dsiponible en: http://www.sorayapaniagua.com/2012/04/15/un-poco-de-historia-sobre-internet-de-las-cosas/ Recuperado el 25 de Agosto de 2017.

[9] Naylampmechatronics. Tutorial SCT-013. Disponible: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html (Consulta: 2017, septiembre)

[10] Naylampmechatronics. Tutorial ESP8266. Disponible: https://naylampmechatronics.com/blog/21_Tutorial-ESP8266-Parte-I.html (Consulta: 2017, octubre)

[11] Enciclopedia. Corriente Alterna. Disponible: http://enciclopedia.us.es/index.php/Corriente_alterna (Consulta: 2018, mayo)

[12] Fairchild Semiconductor. Datasheet LM358. Disponible: http://pdf.datasheetcatalog.com/datasheet/fairchild/LM358.pdf (Consulta: 2018, abril)

[13] IES Leonardo Da Vinci, Alicante. Efecto Hall. Disponible: http://intercentres.edu.gva.es/iesleonardodavinci/Fisica/Electromagnetismo/Electromagnetismo07b.htm (Consulta: 2018, enero)

[14] YHDC. Datasheet SCT-013. Disponible: https://www.mcielectronics.cl/website_MCI/static/documents/Datasheet_SCT013.pdf (Consulta: 2017, noviembre)

ANEXOS

ANEXO A: Programa para rectificación de media onda

```
void setup() {
  Serial.begin(9600);
  analogReference(INTERNAL);
}

void loop() {
  float voltajeSensor = analogRead(A0) * (1.1 / 1023.0); //voltaje del sensor
  float I=voltajeSensor*30.0; //I=VoltajeSensor*(30A/1V)
  Serial.println(I);         //envio por el puerto serie
  delay(300);

}
```

ANEXO B: Programa para calculo de corriente y potencia

```
void setup() {
  Serial.begin(9600);
  analogReference(INTERNAL);
  }

void loop() {

  float Irms=get_corriente(); //Corriente eficaz (A)
  float P=Irms*130; // P=I*V (Watts)

  Serial.print("Irms: ");
  Serial.print(Irms,3);
  Serial.print("A, Potencia: ");
  Serial.print(P,3);
  Serial.println("W");
```

```
  delay(300);
}

float get_corriente()
 {
  float voltajeSensor;
  float corriente=0;
  float Sumatoria=0;
  long tiempo=millis();
  int N=0;
  while(millis()-tiempo<500)       //Tiempo de muestreo de 0.5 segundos
  {
    voltajeSensor = analogRead(A0) * (1.1 / 1023.0);    //voltaje del sensor
    corriente=voltajeSensor*20;
    Sumatoria=Sumatoria+sq(corriente);   //Sumatoria de Cuadrados
    N=N+1;
    delay(1);
  }
  Sumatoria=Sumatoria*4;        //Compensacion de los semiciclos negativos.
  corriente=sqrt((Sumatoria)/N);  //ecuación de la corriente eficaz
  return(corriente);
 }
```

ANEXO C: Programa para lectura y escritura en el Módulo ESP8266.

```
#include <SoftwareSerial.h>
SoftwareSerial ESP(3, 2); // RX | TX

void setup()
  {  Serial.begin(9600);
     ESP.begin(9600);
  }
```

```
void loop()
 {
    if (ESP.available())
       { char c = ESP.read() ;
         Serial.print(c);
       }

    if (Serial.available())
       {  char c = Serial.read();
          ESP.print(c);
       }
  }
```

ANEXO D: Programa de integración y envió de datos a red local

```
#include <SoftwareSerial.h>
SoftwareSerial ESP(3, 2); // RX | TX

void setup()
 {
   Serial.begin(9600);
   analogReference(INTERNAL);
   ESP.begin(9600);
  }

void loop()
{
   while (ESP.available() >0 )
 {
   char c = ESP.read();
```

```
    if (c == 71)
  {
    Serial.println("peticion web enviada");
    delay(500);
    servidor();
   }
 }
}
    float get_corriente()
{
    float voltajeSensor;
    float corriente=0;
    float Sumatoria=0;
    long tiempo=millis();
    int N=0;
    while(millis()-tiempo<500)//Duración 0.5 segundos(Aprox. 30 ciclos de 60Hz)
  {
    voltajeSensor = analogRead(A0) * (1.1 / 1023.0);//voltaje del sensor
    corriente=voltajeSensor*15;
    Sumatoria=Sumatoria+sq(corriente);//Sumatoria de Cuadrados
    N=N+1;
    delay(1);
  }
    Sumatoria=Sumatoria*4;//Para compensar los cuadrados de los semiciclos
negativos.
    corriente=sqrt((Sumatoria)/N); //ecuación del RMS
    return(corriente);
}

void escribir(String text)
 {
```

```
    ESP.print("AT+CIPSEND=0,");
    ESP.println(text.length());
    if (ESP.find(">"))            // Si se recibe el mensaje
 {
    Serial.println(text);
    ESP.println(text);            // SE manda el mensaje por el wifi
    delay(10);
    while ( ESP.available() > 0 )
  {
    if (  ESP.find("SEND OK") )  //se busca respuesta y sale
    break;
  }
 }
}

void servidor(void)
   {

     escribir("<!DOCTYPE HTML>");              //Definicion de tipo de dcumento
     escribir("<html>");                                    //Inicializacion de la sintaxis
HTML
     escribir("<head><title>Monitor de Energia</title>");   //nombre de la pestaña que
llevara la pagina
     escribir("<meta http-equiv=\"refresh\" content=\"5\"></head>");     //tiempo para
refrescar la pagina web
     escribir("<body><h1>  <FONT  SIZE=\"5\"  COLOR=\"red\">  Monitor  SCT-013
</h1>");   //titulo del inicio de la pagina
     escribir(" <BODY BGCOLOR=\"black\"> ");              //Color de fondo

     float Irms=get_corriente(); //Corriente eficaz (A)
     float P=Irms*130; // P=I*V (Watts)
```

```
    escribir("<FONT FACE=\"Arial\" SIZE=\"5\" COLOR=\"blue\"> El consumo es </FONT>"); //escribe y cambia el tamaño, letra y color
    escribir(String(P));       //imprimime la variable
    escribir(" Watts <br /><br />");    //Tipo de Variable

     delay(1);
     ESP.println("AT+CIPCLOSE=0");   //Cierra la comunicacion antes de refrescar
   }
```

ANEXO E: Comandos AT

Comando	Respuesta	Función
AT - Probar iniciación correcta		
AT	OK	Prueba si el módulo responde correctamente
AT+RST - Reinicia el módulo		
AT+RST	OK	Resetéa el módulo
AT+CWMODE - Modo Wifi		
AT+CWMODE=?	+CWMODE:(1-3) OK	Lista los modos validos
AT+CWMODE?	+CWMODE:modo OK	Pregunta en que modo AP esta actualmente el módulo
AT+CWMODE=modo	OK	Establece el módulo en el modo dado 1 = Modo estación (cliente) 2 = Modo AP (huésped) 3 = Modo AP + Estación (modo dual)

AT+CWLAP - Lista APs disponibles		
AT+CWLAP	AT+CWLAP:ecn,ssid,rssi,mac OK	Lista los Acess Points disponibles para conectarse. ecn: codificación, puede ser: 0 = Abierto 1 = WEP 2 = WPA PSK 3 = WPA2 PSK 4 = WPA WPA2 PSK ssid: String que contiene el SSID del AP rssi: Fuerza de la señal mac: String que contiene la dirección MAC
AT+CWLAP=ssid,mac,ch	+CWLAP:ecn,ssid,rssi,mac OK	Busca Acess Points disponibles para conectarse con las condiciones especificadas
AT+CWJAP - Unirse a un Access Point		
AT+CWJAP?	+ CWJAP:ssid OK	Imprime el SSID al que el módulo esta conectado
AT+CWJAP=ssid,pwd	OK	El módulo se conecta al la red con el nombre ssid indicado y la contraseña pwd suministrada
AT+CWQAP - Desconectarse de una Access Point		
AT+CWQAP	OK	Se desconecta de la red que esta actualmente conectado
AT+CWSAP - Configurar el softAP del módulo		
AT+CWSAP?	+CWSAP:ssid,pwd,ch,ecn OK	Pregunta la configuración actual del softAP
AT+CWSAP=ssid,pwd,ch,ecn	OK	Configura el softAP con ssid: String con el nombre de la red pwd: Contraseña, no mayor a 64 caracteres ch: Canal inalámbrico ecn: Tipo de codificación 1 = Abierto 2 = WPA_PSK 3 = WPA2_PSK 4 = WPA_WPA2_PSK
AT+CIPSTATUS - Información acerca de la coneción		
AT+CIPSTATUS	STATUS:status +CIPSTATUS:id,type,addr,port,tetype OK	status: 2 = Se obtuvo IP 3 = Conectado 4 = Desconectado id: ID de la conexión en caso de multiconexión (1-4) type: Tipo de conexión, "TCP" o "UDP" addr: Dirección IP de la conexión port: Numero del puerto tetype: 0 = El módulo corre como cliente 1 = El módulo corre como servidor
AT+CIPMUX - Habilitar o deshabilitar multiples conexiones		
AT+CIPMUX=mode	OK	mode: 0 = Conexión unica 1 = Múltiples conexiones, hasta 4
AT+CIPMUX?	+CIPMUX:mode OK	Imprime el mode, el modo de conexión actual

AT+CIPSTART - Establece una conexión TCP o registra un puerto UDP e inicia la conexión		
AT+CIPSTART=type,addr,port	OK	Empieza una conexión como cliente (en modo conexión única) type: puede ser "TCP" o "UDP" addr: String que contenga la dirección IP remota port: String que contenga el puerto remoto
AT+CIPSTART=id,type,addr,port	OK	Empieza una conexión como cliente (En modo conexión múltiple) id: ID de la conexión (1-4)
AT+CIPSTART=?	[+CIPSTART:(id)("type"),("ip address"),(port)] OK	Lista los posibles comandos
AT + CIPCLOSE - Cierra la conexión TCP o UDP		
AT+CIPCLOSE=?	OK	
AT+CIPCLOSE=id	OK	Cierra la conexión TCP o UDP con el ID "id" en modo conexión múltiple
AT+CIPCLOSE	OK	Cierra la conexión TCP o UDP para modo de conexión única
AT+CIPSEND - Enviar datos		
AT+CIPSEND=?	OK	
AT+CIPSEND=length	SEND OK	Establece la longitud de datos a enviarse (máximo 2048). Para un envío normal (modo conexión única)
AT+CIPSEND=id,length	SEND OK	Establece la longitud de datos a enviarse en la conexión número "id". Para un envío normal (modo conexión múltiple)
AT+CIPSEND		Envía datos sin adornos cada 20ms. El módulo retorna ">" después ejecutar el comando, si se recibe el comando "+++" se regresa al modo comando.

AT+CIFSR - Obtener la dirección IP local		
AT+CIFSR=?	OK	
AT+CIFSR	+CIFSR:ip OK	Retorna la dirección IP local del módulo como cliente.
AT+CIFSERVER - Configurar como servidor		
AT+CIPSERVER=mode[,port]	OK	Configura el módulo como servidor donde el modo: 0 = Borrar servidor 1 = Crear servido puerto: numero del puerto, por defecto es el 333
AT+CIOBAUD Cambiar la velocidad de transmisión serial		
AT+CIOBAUD=?	+CIOBAUD:(9600-921600) OK	Nos informa que las velocidades de transmisión permitidas están en este rango
AT+CiOBAUD?	+CIOBAUD:baudrate OK	Nos indica que el módulo está actualmente configurado a 'baudrate'
AT+CIOBAUD=baudrate	OK	Configura la velocidad de transmisión a 'baudrate'

ANEXO F: Tabla de Costos

Gasto	Costo $
Arduino UNO	**170**
ESP8266 01	**100**
SCT 013 030	**200**
Rectificador de media onda	**20**
Conector hembra de mini Jack	**15**
Total	**$ 505**