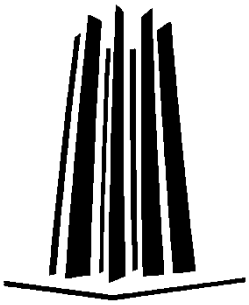
Universidad Nacional Autónoma de México

Facultad de Estudios Superiores

Aragón



Sistema de medición y monitoreo de consumo eléctrico con interfaz en teléfono inteligente para gestión racional de aprovechamiento energético

PRESENTA

Alejandro Andrés Serapio Carmona

ASESOR DE TESIS

Dr. Ismael Díaz Rangel

Cd. Nezahualcoyotl, Edo. de México

2016

Agradecimientos

Dedico este trabajo a mis padres: Ninfa Carmona Campos y Alejandro Adalberto Serapio Farías, por el labor incansable e inaudito que me han demostrado desde el primer momento en que respire por primera vez; sus acciones, proezas, su cariño y su infinito amor me han demostrado que ¡SIEMPRE SE PUEDE! Contra cualquier obstáculo que nos presente la vida.

Agradezco también a mis abuelitos: Teresa, Octaviano, Andrés y Leonor. Y a mis hermanos: Ninfa, Octaviano y Mauricio, siempre han sido una luz que me ha iluminado y acompañado en mi camino.

A mi familia, que, aunque me encuentre en otro país, siempre estarán conmigo.

También a todos mis ¡Nakama! Que en las buenas y en las malas me han acompañado y nunca se han rajado.

También quiero reconocer el tiempo, la ayuda y la paciencia de mis revisores de este trabajo, Al Dr. Vega Ramírez, Mtro. Gastaldi Pérez, M. en I. Gutiérrez Flores y al Ing. García Zárraga, sin su tiempo y dedicación, esto no hubiera sido posible.

Merece una mención y un agradecimiento especial mi asesor de tesis: Dr. Ismael Díaz Rangel, por todo el apoyo, la atención, y la ayuda brindada en mi desarrollo académico, profesional y personal, sus observaciones, consejos y su inmensurable apoyo han hecho posible este momento, me siento muy afortunado por haberlo encontrado en mi camino.

A la máxima casa de estudios, mi Universidad Nacional Autónoma de México, por haber formado en mí, a un profesionista y universitario, por haberme proporcionado las herramientas y los conocimientos necesarios para mi preparación y la de mis compañeros.

Gracias

Contenido

Índice de ilustraciones

Índice de tablas

Capítulo 1. Introducción

En la actualidad a nivel mundial existe un alto porcentaje de personas que cuentan con energía eléctrica, en el año 2012 la relación global del uso de electricidad fue del 84.6% (energia, 2016), lo que indica que más de tres cuartas partes del planeta usan o tienen acceso a la electricidad.

Tan solo en México, en el mismo año, se llegó a cubrir un 99,1% del total de su población en términos de abastecimiento eléctrico; es una buena cifra, siempre y cuando existan los recursos naturales necesarios para poder continuar con el abastecimiento.

Dado que día a día más personas tienen acceso a la energía eléctrica, es de suponer que debe ser mayor la necesidad de cuidar los recursos energéticos; para poder cubrir la demanda energética, las empresas deberán incursionar en mejores métodos para la producción de potencial eléctrico, como son las energías renovables; además, las personas deberán gestionar eficientemente el uso de la energía eléctrica.

En México la Comisión Federal de Electricidad (CFE) es la encargada de brindar el servicio de abastecimiento eléctrico en el país, posee una gran infraestructura, ofrece sus servicios a los sectores domésticos, industriales, privados, transporte, público y comercial; además, ha incursionado en la generación de energía a través del uso de las energías renovables.

Lo que se debe tomar en cuenta no es solamente la generación de la energía, sino cómo y cuánto se está cobrando por el servicio.

CFE instala un medidor de consumo, que es el encargado de registrar cuánta energía estamos consumiendo, para después podernos cobrar. La desventaja de este sistema, es que el usuario no conoce explícitamente cuánto ha consumido en el periodo en curso, y por ende cuanto va a pagar, hasta que llega el recibo, claro. Además, CFE maneja varias tarifas que se ajustan a cada zona específica del país; dentro del rubro de servicio doméstico, CFE utiliza 3 rangos tarifarios: consumo básico, medio y alto. Cada tarifa varía en función del rango de consumo y zona dónde se lleva a cabo. El truco del que todos los usuarios somos víctimas, es que cuando superamos un rango de consumo, el excedente fuera del rango, se

cobrará conforme al precio de la siguiente tarifa, y sí el siguiente rango también es rebasado, sucederá lo mismo, hasta llegar al rango de la última tarifa; no suena tan alarmante, hasta que se supera el ultimo rango de consumo, que, para este caso, se cobrará por el total del consumo, a un solo precio, lo que genera un incremento sustancial en el recibo de pago.

Es por ello que la razón de este trabajo es brindar a las personas una herramienta que los auxilie en la gestión y racionalización de la energía, mediante un dispositivo que sea capaz de enviar la información hacia una aplicación desarrollada para un teléfono inteligente, que indique la cantidad de energía eléctrica consumida en lo que va del periodo en curso, así como un estimado del monto a pagar por dicha energía; adicionalmente, mostrará los días faltantes para la fecha de corte, que es cuando se toman las lecturas y nos cobran por el uso de la energía.

1.1 Objetivo general

Diseñar, construir e implementar un medidor de consumo eléctrico que sea preciso, fácil de instalar y que transmita su información hacia una aplicación desarrollada para el sistema operativo Android; la aplicación mostrará el consumo a partir de la fecha de corte, así como un estimado del monto a pagar por dicho consumo, y también indicará los días restantes del periodo que se está evaluando.

1.2 Objetivos específicos

- Detallar la situación en México de la producción y consumo eléctrico.
- Identificar las condiciones de cobro de consumo eléctrico.
- Presentar los diferentes métodos de medición de potencia eléctrica.
- Desarrollar un sistema de medición de consumo eléctrico.
- Definir e implementar un sistema de control basado en microcontrolador.
- Especificar e incorporar un reloj de tiempo real.
- Diseñar una aplicación para visualización y configuración del sistema.
- Describir el funcionamiento de la comunicación Bluetooth.
- Registrar la información del consumo en una memoria externa.

- Seleccionar y aplicar un método estadístico para determinar la validez de las mediciones del instrumento propuesto.

1.3 Motivación

En las viviendas se cuenta con un medidor de consumo eléctrico que es proporcionado por la empresa que brinda el suministro eléctrico. Pero el instrumento conocido como "medidor de luz" lo que muestra es el acumulado del consumo desde que fue instalado; esto implica que es más fácil enterarse de la energía consumida en un periodo hasta, que llega el recibo de pago. Debido a ello, se decidió crear un sistema que proporcione la información acerca del consumo correspondiente al periodo en curso, con la finalidad de que el consumidor tenga una herramienta que le permita; ejercer una gestión racional sobre el consumo de la energía eléctrica, y así disminuir el total del recibo de la luz.

1.4 Justificación

Los beneficiados del uso de este sistema seremos todos, se podrá contar con una herramienta que proporcionará de manera anticipada a la recepción del recibo, así como un aproximado de la cantidad que se deberá pagar por dicho consumo; además indicará los días faltantes para el término del periodo. Además el usuario podrá configurar: las tarifas, el periodo de medición, y una alerta de consumo (que se podrá determinar) si quiere conocer el consumo diario, semana, mensual, por periodo o por año, o cuándo llegue a un monto específico. Además, toda la información que el sistema proporcione, podrá ser usada para que el usuario, conozca la situación de la energía que ha consumido y conozca de manera anticipada a la llegada del recibo, el pago que deberá realizar para cubrir el servicio otorgado.

1.5 Descripción del capitulado

Capítulo 2. Se describe el marco teórico de esta propuesta, el cual hablará de: CFE en México, Tarifas de CFE, funcionamiento del cobro del servicio, sistemas de control, desarrollo de aplicaciones en Android, información de dispositivos (comunicación bluetooth, sensores, contadores y demás elementos que conforman al sistema).

Capítulo 3. Se hablará del desarrollo experimental del dispositivo, los problemas hallados, sus soluciones y de cómo se condujo la elaboración del mismo.

Capítulo 4. Se referirán las pruebas hechas al dispositivo, así como los resultados obtenidos.

Por último, se darán las conclusiones y se hablará del trabajo por realizar.

Capítulo 2. Marco Teórico

2.1 Comisión Federal de Electricidad

La Comisión Federal de Electricidad (CFE) ha brindado el servicio de suministro eléctrico desde el 14 de agosto de 1937, (Comision Federal de Electricidad, 2014) fecha en la cual se dio de alta como entidad organizadora, a nivel nacional, de la energía eléctrica. Se ha encargado desde entonces de coordinar, mantener y administrar las etapas de generación, elevación, transportación, distribución y transformación para hacer llegar la energía a nuestros hogares y/o empresas.

Para poder cobrar de manera el consumo eléctrico, la manera de cuantificar la cantidad de energía que hemos consumido es mediante un dispositivo llamado: Watthorímetro, con la lectura que proporciona, se calcula el precio a pagar dependiendo de la lectura, pues existen rangos diversos de clasificación de pagos llamados tarifas.

2.1.1 Tarifas

La empresa CFE que suministra la energía eléctrica al país, tiene un sistema de tarifas establecidas, en donde se compara tu consumo en un periodo y dependiendo del rango que has cubierto en el que coincida tal consumo, será la tarifa que te aplican, más el excedente que hayas tenido, se te cobrara con el precio de la siguiente tarifa. Ahora bien, estas tarifas se cobran a distintos precios y la diferencia entre una y otra es un *KWh* (kilowatt-hora), por ejemplo: si tu consumo fue de *173.0 KWh*, rebasaste la tarifa básica considerada en 150.0 *KWh* por 23 KWh, donde se cobra $0.793 por cada uno los primeros 150 KWh, y 0.953 porcada uno de los 23 KWh restantes, siguiendo el ejemplo: deberás pagar $118.95 de la tarifa básica y $21.98 por 23 KWh excedidos de la tarifa base pagando un total de $140.93.[1]

Ahora, lo más preocupante viene después, cuando has excedido la tarifa de alto consumo que es mayor a 500 KWh, en el periodo, se te clasificará en tarifa DAC (Domestica de Alto Consumo), la cual tiene un precio unitario para todo lo que consumiste, (ya no hay rangos de tarifas, se te cobra el total de tu consumo), el gran detalle de esta tarifa, es que el precio no es fijo, es variable, (en un periodo puede estar en ochenta y tres centavos, y para el

[1] Tarifas con inicio de vigencia a partir de enero 2016.

próximo periodo, se puede encontrar en lo doble); lo que representa un gasto mucho mayor para el usuario. (*Tabla 2.1*).

Tabla 2.1 Precio tarifas

CONSUMO	TARIFAS (KWh)	TARIFA DAC ($2.80 x KWh)
173KWh	$ 140.93	$484.4

Las tarifas oficiales que maneja CFE están impresas en nuestros recibos de luz y se encuentran publicadas en su portal de internet (Comision Federal Electricidad, 2016). (*Tabla 2.2*).

Tabla 2.2a Tarifas CFE

Rango de consumo	Dic./2015	Ene.	Feb.	Mar.	Abr.	May.	Jun.	Jul.	Ago.	Sep.	Oct.	Nov.	Dic.
Básico 1-150	0.809	0.793	0.793	0.793	0.793	0.793	0.793	0.793	0.793	0.793	0.793	0.793	0.793
Intermedio 151-280	0.976	0.956	0.956	0.956	0.956	0.956	0.956	0.956	0.956	0.956	0.956	0.956	0.956
Excedente 281-500	2.859	2.802	2.802	2.802	2.802	2.802	2.802	2.802	2.802	2.802	2.802	2.802	2.802

Tabla 2.2b Tarifas DAC

CARGOS POR	Dic./2015	Ene.	Feb.	Mar.	Abr.	May.	Jun.	Jul.	Ago.	Sep.	Oct.	Nov.	Dic.
Cargo fijo ($/mes)	86.23	86.16	86.62	88.04	89.24	88.76	88.81	90.09	91.77	92.45	92.86		
Cuotas por energía consumida en fuera de verano													
($/kWh)	3.273	3.319	3.436	3.385	3.536	3.356	3.417	3.612	3.705	3.862	3.866		

2.1.2 Medidor (Watthorímetro)

Los medidores de consumo de energía eléctrica, se usan para poder cuantificar el consumo eléctrico de un usuario, con el fin de poder tener una forma de cobrarle al cliente la cantidad correspondiente al consumo. Y para ello todos los medidores basan sus lecturas sobre una unidad establecida el Kilo-Watt-hora (KWh), la cual representa 1000Watts consumidos a lo largo de una hora.

2.1.2.1 Medidores Analógicos

ILUSTRACIÓN 2.1 CARATULA MEDIDOR ANALÓGICO

Los medidores analógicos (*ilustración 2.1*), también conocidos como medidores electromecánicos, o medidores de inducción (el principio físico por el cual registran el consumo eléctrico), fueron los primeros medidores que se instalaron en los hogares, negocios e industrias. Con el único y principal fin, de proporcionarle a la empresa responsable del servicio, la lectura de la cantidad de energía consumida en un periodo, para poder cobrar la cantidad respectiva por dicho consumo.

El funcionamiento de este medidor se encuentra sus componentes (*ilustración 2.2*). La energía entra por un borne de conexión, llega hasta una bobina, después sale hacia la toma principal de la carga (casa, oficina, empresa, industria, etc.), es decir que la bobina está conectada en serie con la línea principal de la carga, por lo que la corriente será la misma. La corriente circula atreves de una bobina, la cual produce un campo electromagnético (por el principio de inducción), de fuerza proporcional a la cantidad de energía que se encuentra circulando por ella; es el campo electromagnético, el encargado de mover un disco

magnético, el cual, al entrar en contacto con el campo electromagnético de la bobina, gira en proporción a la intensidad del campo; es decir, que a más energía que demande la carga, más rápidos serán los giros del disco. El disco se encuentra conectado a un juego de engranes, los cuales componen un sistema numérico conocido como registro, donde los valores de miles se encuentran a la izquierda, y las unidades son el de la derecha, donde cada vez que el contador de las unidades da una vuelta (del 0 al 9) provoca que el próximo a la izquierda se incremente en una unidad y así sucesivamente hasta llegar el disco de los miles.

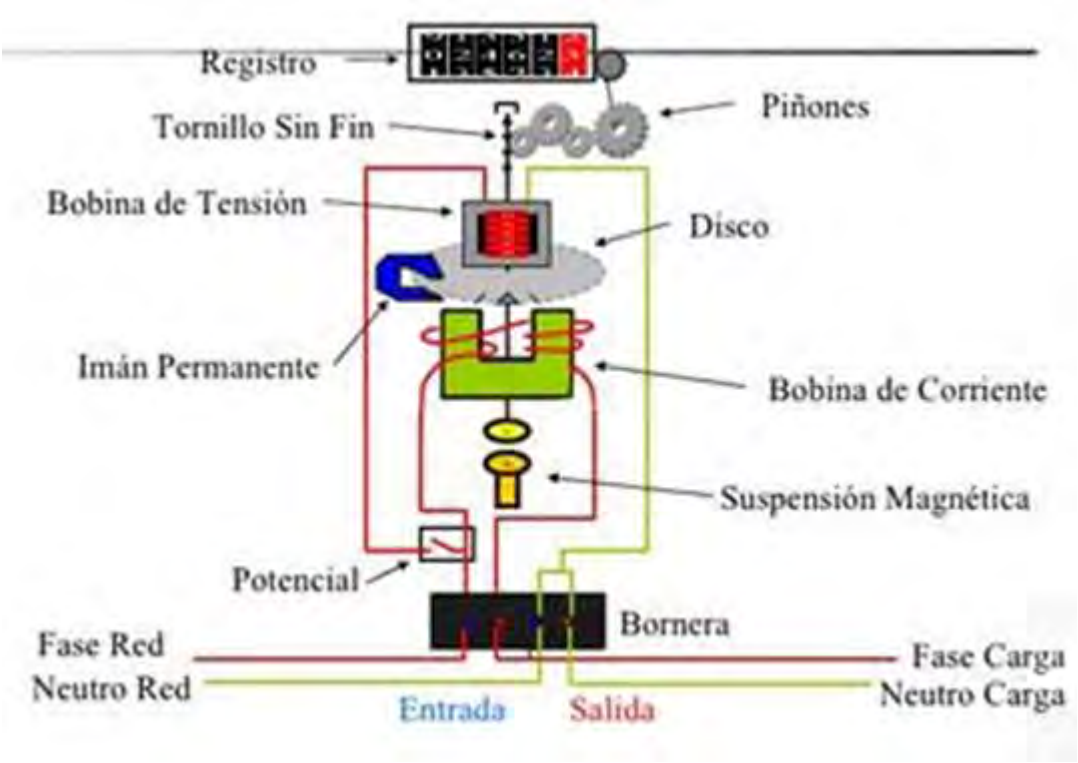


ILUSTRACIÓN 2.2 COMPONENTES MEDIDOR ANALÓGICO

2.1.2.2. Medidores Digitales

Se diferencian principalmente por no tener registro de engranes ni disco magnético. Y la manera de manejar la información es digital. La aplicación es la misma, medir el consumo de energía, solo que, en estos medidores, la forma de adquirir las lecturas, es mediante el uso de elementos electrónicos, como sensores de corriente, de campo magnético, etc. Los cuales adquieren información sobre el consumo de energía, y muestran una lectura, que es enviada a un sistema de control (generalmente a un procesador), que se encarga de

interpretar, convertir y enviar la información ya procesada a una pantalla, registro de memoria, o dirección de internet, todo depende de las aplicaciones con las que haya sido fabricado.

Desde el 2012 CFE ha proporcionado el cambio de medidores analógicos por nuevos medidores digitales, los cuales, poseen una pantalla donde anuncian el consumo y un led que indica el estado del medidor.

2.2 Sistemas de control

En todo sistema siempre se busca la estabilidad del mismo, la linealidad o el comportamiento predecible, es un factor que todos los sistemas de control o monitoreo siempre buscan, la estabilidad brinda orden y control.

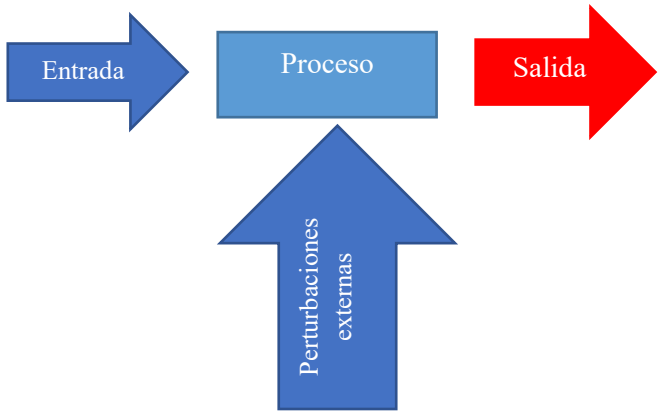


ILUSTRACIÓN 2.3 DIAGRAMA SISTEMA

Se define sistema, como la combinación de varios elementos que actúan en conjunto, con el fin de realizar una o más acciones. Por ejemplo, una bicicleta es un sistema, pues está compuesta por ruedas, llantas, una cadena, manubrio, etc. Cada elemento individualmente sería un elemento solamente, pero cuando se combinan crean un sistema.

Se caracterizan principalmente por tener una entrada, después un proceso y al último una salida. Retomando el ejemplo de la bicicleta:

- La entrada del sistema, es la fuerza motriz que otorga el movimiento de las piernas.
- El proceso es cuando los pedales, la cadena y las estrellas, producen el movimiento de la rueda trasera, debido al movimiento de las piernas.
- La salida, es desplazamiento de la bicicleta.

- Las perturbaciones externas se consideran como los factores ajenos al sistema, como lo puede ser un hoyo en el piso, un tope, etc.

Un sistema de control es un conjunto de dispositivos o elementos que se encuentran conectados entre sí y son capaces de corregir, manipular o dirigir a otro sistema. Tan solo, un ejemplo de ello es nuestro cerebro (sistema de control a través de la sinapsis entre las neuronas), es el encargado de incrementar la frecuencia con la que el corazón bombea sangre cuando uno se mantiene en un estado de excitación como él estar corriendo, y al mismo tiempo comienza a secretar por los poros de la piel sustancias (sudor) que funcionan como refrigerante, para mantener la temperatura del cuerpo en un rango estable.

Los sistemas de control se caracterizan por mantener el control sobre el funcionamiento o proceso de algún sistema, para garantizar la adecuada operación del mismo. Para ello se dedican al monitoreo de una o varias variables, de las cuales toma lecturas, las compara con parámetros establecidos y dependiendo del resultado será la línea de acción.

Existen dos tipos de sistemas de control, lazo cerrado y lazo abierto.

2.2.1 Sistemas de control de lazo abierto

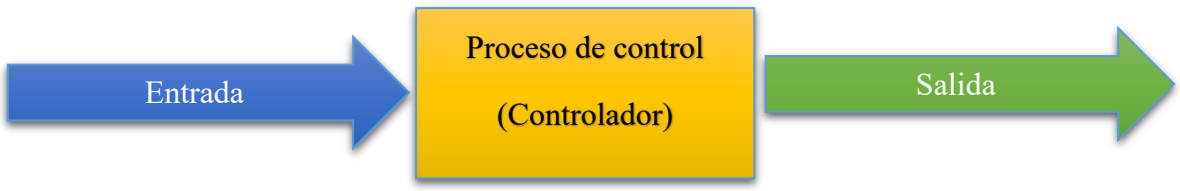


ILUSTRACIÓN 2.4 DIAGRAMA SISTEMA DE CONTROL DE LAZO ABIERTO.

Los sistemas de control de lazo abierto, son aquellos que siguen una secuencia especifica (rutina) una y otra vez; por ejemplo, un semáforo, es un aviso luminoso que indica a los automovilistas y peatones cuando se debe de cruzar, el funcionamiento del semáforo se encuentra programado, así que solamente realiza una rutina para ceder o detener el paso de los vehículos, sin importar si su funcionamiento entorpece el tráfico. Solamente sigue una rutina, no toma información del tránsito o el flujo vehicular. Una característica de este tipo de sistemas es que solamente ejecutan una rutina, sin tomar información del medio, lo que impide que exista una mayor eficiencia y estabilidad del sistema.

Un ejemplo de ello es una bomba de agua, la que se encarga de subir el líquido a través de un sistema de tubos, para llegar a un contenedor en la parte más alta de las casas. Si solamente encendemos la bomba de agua, está comenzara a adicionarle al agua energía cinética, la cual al estar contenida en tubos provocara un desplazamiento del líquido de manera proporcional a la potencia de la bomba, hasta que el agua alcance la energía potencial a la que se encuentra el contenedor, haciendo que el agua se almacene en el contenedor. Si se mantiene energizada la bomba más tiempo del debido (más tiempo del que tarda la bomba en llenar el contenedor), el resultado será la superación de la capacidad del contenedor resultando en un derrame de líquido.

2.2.2 Sistemas de control de lazo cerrado

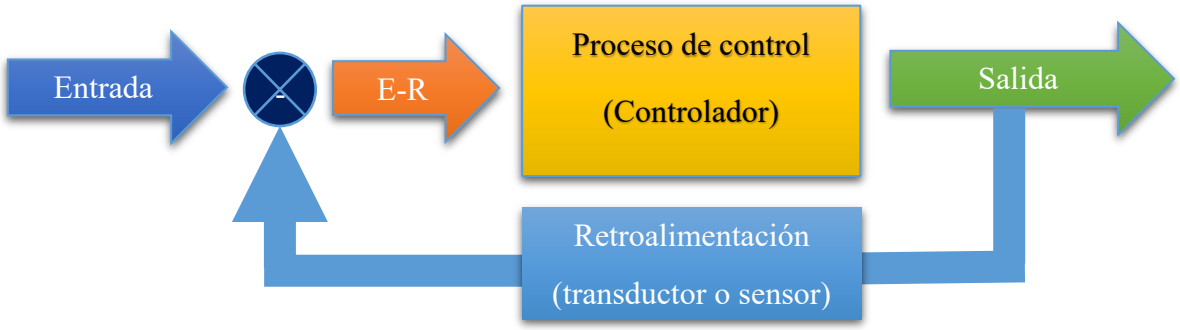


ILUSTRACIÓN 2.5 DIAGRAMA DE SISTEMA DE CONTROL DE LAZO CERRADO.

Este tipo de sistemas de control se caracterizan principalmente por tomar información de la salida del sistema y la comparan con la entrada, se llama retroalimentación, y lo que genera es una modificación o corrección del proceso para adecuar la salida a los requerimientos planteados del sistema. Por ejemplo: el sistema de medición de emisiones que los vehículos modernos (del año 2000 hacia delante) poseen; dicho sistema es capaz de modificar la cantidad de aire/combustible que existe en el proceso de combustión, para producir menos emisiones y mejorar el consumo de combustible; realiza mediciones de gases en la salida del múltiple de expulsión para conocer el estado de los gases, y si el sistema determina que las emisiones son altas en gasolina, disminuye la cantidad de combustible. Dependiendo del valor obtenido a la salida, ajusta la entrada.

Dónde:

- Entrada. – Se define como una señal de estímulo que se aplica al sistema de control para producir una respuesta.
- E-R. – Es la señal de entrada al sistema de control se obtiene restando a la entrada el valor de la retroalimentación, que es la salida del sistema.
- Proceso de control. – Se refiere a las instrucciones o acciones que el sistema debe ejecutar para obtener la salida. De manera que esta sea lo más estable posible.
- Retroalimentación. – Es la señal de reingreso al sistema que se define como la señal que el sistema debe ajustar, esta puede ser transformado por algún transductor, o proporcionada por algún sensor.
- Salida. – Es la respuesta obtenida del sistema, también se conoce como variable controlada (pues el sistema de control la puede modificar).

2.2.3 Microcontroladores

Un controlador, es un dispositivo que contiene los elementos necesarios para ejecutar órdenes y control.

ILUSTRACIÓN 2.6 ENCAPSULADO DIP ATMEGA 328

En la actualidad han evolucionado tanto los controladores, que se ha llegado a reducirlos a tamaños de hasta algunos milímetros. Es tanta la reducción de tamaño de estos dispositivos que se les ha denominado como microcontroladores, siguen siendo dispositivos capaces de producir control sobre otros dispositivos solo que ahora son de talla pequeña. Se pueden encontrar por si solos como circuitos integrados en forma DIP (Dual In-line Package) (ver *ilustración 2.6)*, encapsulados de plástico negro con un buen de patitas (terminales de conexión), que se pueden atravesar por los orificios de las tarjetas fenólicas, o tarjetas de prueba (protoboards). O SOIC (Small Outline Integrated Circuit) (ver *ilustración 2.7)*, que

son aún más reducidos que los DIP, solamente que para su montaje, se requieren de herramientas especializadas, por la exactitud con la que deben de contar, pues dado al tamaño aún más reducido, las terminales de conexión (patitas) son muy reducidas para la utilización de herramientas convencionales, además, este tipo de encapsulados se fijan a las tarjetas fenólicas de manera superficial, es decir, no existen orificios por los cuales atravesar las terminales, sino estás se fijan en pistas sobre la superficie.

ILUSTRACIÓN 2.7 ENCAPSULADO SOIC ATMEGA 328

Estos microcontroladores son realmente sistemas de alta escala de integración, pues aunque apenas midan unos centímetros el potencial de estos dispositivos es muy grande, es tanto el potencial y fama que han tenido, que en torno a los microcontroladores, las compañías que los fabrican han creado tarjetas de desarrollo, las cuales siguen conservando las mismas características del microcontrolador, solamente que la versión con la tarjeta contiene los elementos necesarios para realizar experimentos, pruebas y aplicaciones.

Un microcontrolador está compuesto (ver *ilustración 2.8),* a grandes rasgos, por un microprocesador (CPU), una memoria volátil (RAM) y no volátil (ROM), temporizadores, una señal de sincronía o de reloj, periféricos, entradas y salidas.

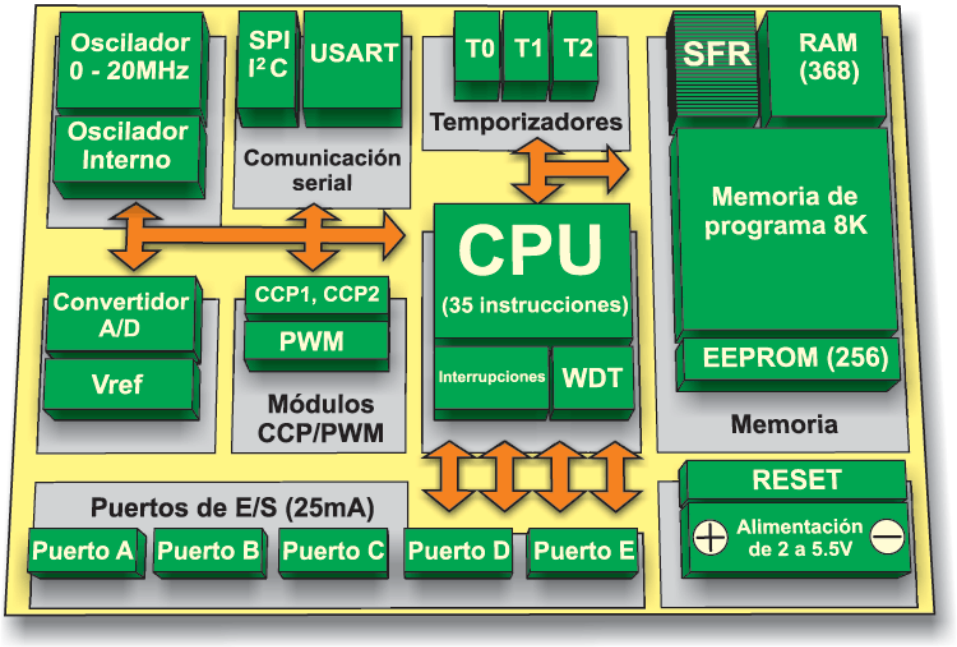


ILUSTRACIÓN 2.8 COMPONENTES MICROCONTROLADOR

Dependiendo del modelo, serán los periféricos con los que doten al microcontrolador. Existen versiones que contienen convertidores analógicos digitales, módulos de comunicación serial, módulos de PWM (Pulse Wave Modulation), etc.

2.2.3.1 Tarjetas con Microcontroladores

Una tarjeta de desarrollo, (*ilustración 2.9*) es un dispositivo electrónico que cuenta con un microcontrolador, entradas y salidas, periféricos, una serie de componentes electrónicos para su funcionamiento y un entorno de programación, que es la forma de comunicarle a la tarjeta que quieres que haga.

ILUSTRACIÓN 2.9 TARJETAS DE DESARROLLO

Los microcontroladores son circuitos encapsulados de alta escala de integración. Los microcontroladores son las piezas principales de toda tarjeta de desarrollo, pues son donde se concentran todas las operaciones lógicas, procesos programados para la ejecución de algún programa o serie de instrucciones a realizar.

Existen muchos fabricantes de tarjetas de desarrollo, como:

- Arduino Genuino
- ChipKit
- Tessel
- LaunchPad
- Raspberry Pi
- Beaglebon

Solo por mencionar algunos.

Dentro de sus principales características, todos cuentan con un microprocesador, memorias, entradas y salidas programables, periféricos, comunicación serial. Algunos cuentan con convertidores analógicos digitales (CAD), interfaces precisas para cámaras, micrófonos, pantallas, actuadores, sensores.

2.2.3.1.1 Arduino

Para la creación del dispositivo se revisó una serie de tarjetas de desarrollo, para decidir cuál sería la más idónea para el desarrollo del prototipo, se revisaron:

- Raspberry Pi 2. - Una tarjeta de desarrollo muy poderosa, pues se define como una mini computadora, pues debido a sus características de procesamiento y memoria, puede procesar programas operativos como Linux, Ubuntu y hasta una versión específica de Windows 10. Su precio se encuentra alrededor de los $1000.00 pesos.
- Arduino DUE. – Una versión más grande y completa del Arduino UNO, sus características más prominentes son: que posee una mayor cantidad de memoria, más entradas y salidas programables, y tiene más puertos de conexión serial. Su precio en el mercado está alrededor de $500.
- Arduino UNO. – (ver *ilustración 2.10)* La versión estándar de Arduino, posee lo general de todas las versiones, cuenta con un convertidor analógico digital, entradas

y salidas configurables, un puerto de comunicación serial, una memoria no volátil de 1kByte, también cuenta con salidas PWM, su precio ronda los $300.

Se escogió la versión UNO, debido a las características que posee la tarjeta (Atmel, 2015):

- Microcontrolador: ATmega328P.
- Alimentación del microcontrolador: 5Vdc.
- Voltaje de alimentación (conector): 7Vdc – 12Vdc.
- Corriente por pin de E/S: 20mA.
- Corriente en pin 3.3Vdc: 50mA.
- Memoria Flash: 32KB, 0.5KB reservados para el programa de arranque.
- SRAM (RAM estática): 2KB.
- EEPROM (**ROM eléctricamente borrable programable**): 1KB.
- Velocidad de reloj: 16MHZ.
- ADC (Convertidor Analógico Digital): 10 bits (resolución).

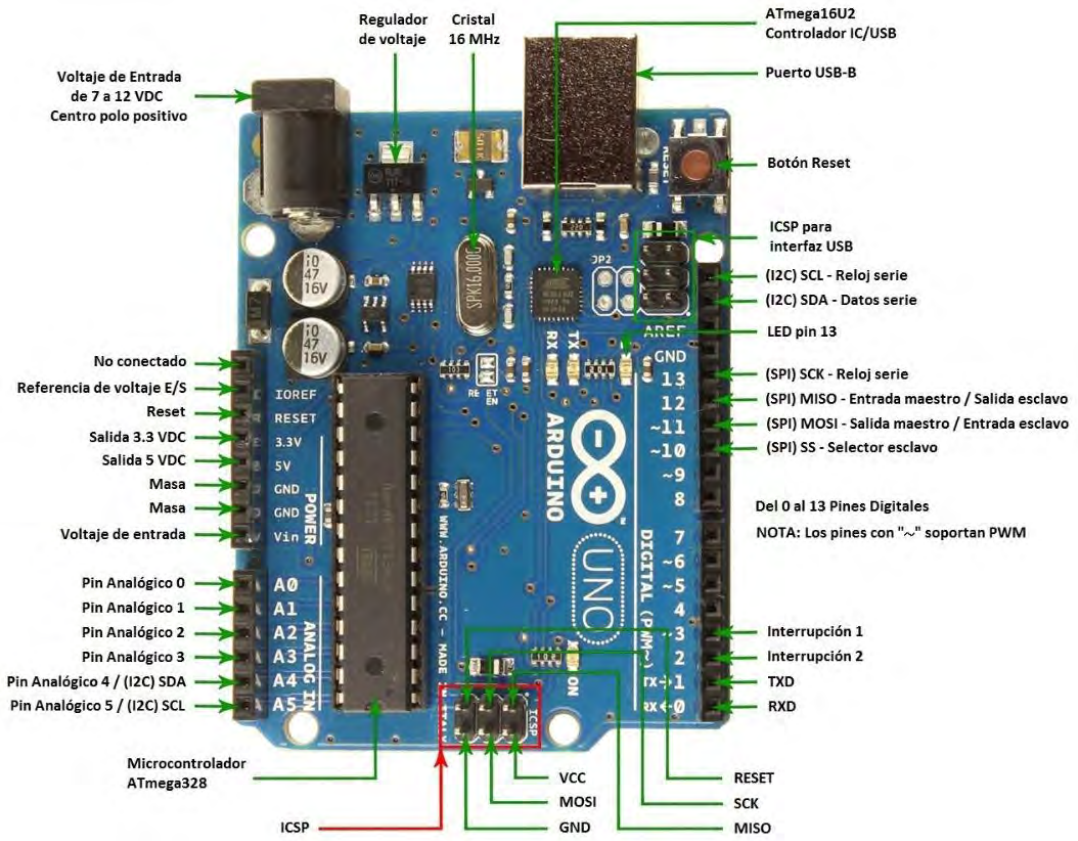


ILUSTRACIÓN 2.10 TARJETA DE DESARROLLO ARDUINO UNO

2.3 Entorno de programación Arduino

Para poder trabajar con el entorno de Arduino (*ilustración 2.11)* hay que conocer un poco de él. Hay que ingresar a la página oficial de Arduino, para descargar el entorno de forma gratuita (las donaciones son bien recibidas). El entorno tiene una zona de trabajo en donde se escribe el código, una barra de herramientas con diversos menús y una sección de mensajes del compilador.

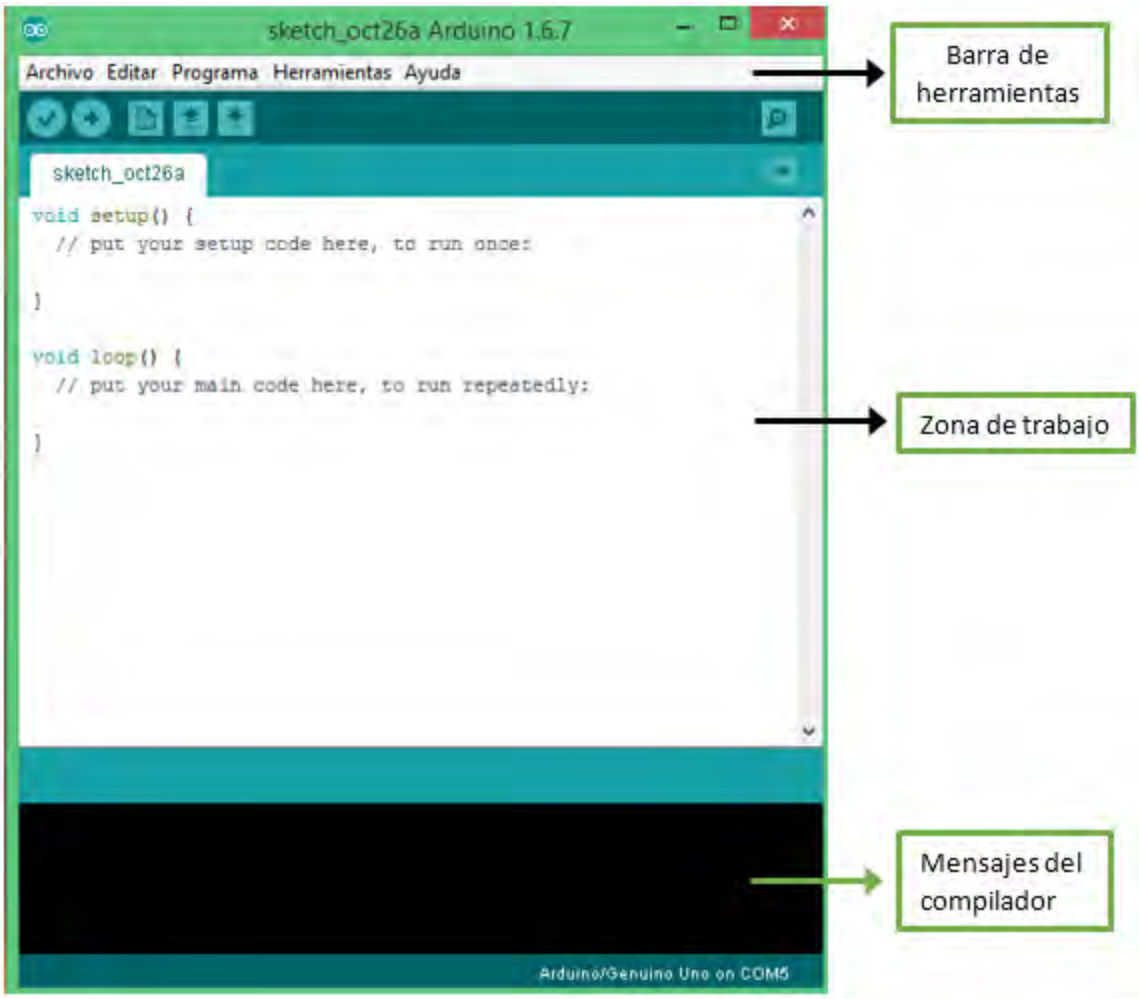


ILUSTRACIÓN 2.11 ENTORNO ARDUINO IDE

Zona de trabajo. – Se encuentra a la mitad de la ventana, posee dos zonas principales, llamadas SETUP y LOOP respectivamente, donde ambas **forzosamente** deben estar incluidas en el código del programa, pues sin estas dos zonas el programa no funcionaria. En la zona de SETUP, colocaremos las variables principales del programa, bibliotecas, y configuraciones que solo queramos que se configuren una sola vez. En la zona LOOP se

encontrará nuestro programa principal que se ejecutará una y otra vez, aquí es donde se colocan todas las instrucciones que queremos que se estén ejecutando.

Barra de herramientas. – Se encuentra en la parte superior de la ventana del entorno, existen 5 opciones de menú:

- Archivo. – Encontramos las opciones para abrir, guardar, empezar un nuevo proyecto, cargar algunos ejemplos con los que cuenta el entorno.
- Editar. – Es un apartado que contiene las herramientas para la edición del texto, como el tamaño, seleccionar texto, búsqueda de palabras.
- Programa. – Son las opciones para compilar el código, cargarlo o subirlo a la tarjeta, añadir bibliotecas, ficheros, abrir la ubicación del archivo.
- Herramientas. – Se encuentran las opciones para configurar el puerto de lectura (que es donde se encuentra conectado la tarjeta), iniciar el monitor serial, seleccionar el modelo de la tarjeta con la estamos trabajando.
- Ayuda. – En esta sección, encontraremos categorías que nos podrán ayudar a aclarar alguna duda respecto al funcionamiento del entorno, así como de sus funciones, también para buscar o comentar alguna duda en el foro oficial de Arduino, para mostrar el problema y conocer si alguien que estaba en la misma situación logro darle solución y si fue así, que fue lo que hizo.

Los botones que se encuentran debajo de la barra de herramientas, son botones de acceso directo a funciones específicas, el primero de izquierda a derecha, que se parece mucho a una palomita, (de las que te ponen en los exámenes, no de las que vuelan…) sirve para revisar el código que has escrito, para conocer si existe algún error que impida el correcto funcionamiento del código, pueden ser del tipo: ortográfico (como un ; al final de una instrucción), o si tienes un error de declaraciones (como si a un valor entero le propusieras un valor racional). El segundo es una flecha con sentido hacia la derecha, funciona para cargar el programa a la tarjeta. El tercero es una hoja de papel, que sirve para crear un nuevo espacio de trabajo. El cuarto es una flecha que apunta hacia arriba, y sirve para guardar tu código y no perderlo. El quinto, una flecha que apunta hacia abajo, sirve para cuando quieres abrir algún otro código previamente guardado. Y, por último, un botón que se encuentra hasta el borde derecho, es como una lupa, y sirve para iniciar el monitor serial

para poder enviar y recibir información por el puerto serial de la computadora hacia la tarjeta y viceversa.

Mensajes del compilador. – Es una sección de la ventana del entorno que nos muestra todos los mensajes que el entorno nos envía, como cuando hay un error en el código, es por este espacio por donde nos lo hace saber, lo mismo para cuando el código es correcto y se ha cargado con éxito a la placa, en este mismo espacio es por donde nos lo hace saber.

2.3.1 Estructura del código

En la zona de trabajo (ver *ilustración 2.11)*, existe una estructura definida, sin la cual, el entorno no podría cargar los códigos a la tarjeta Arduino, estas estructuras son ***Setup*** y ***Loop***. Además de ellas, se pueden crear funciones fuera de las estructuras, estas funciones tienen la característica de disminuir la cantidad de código si es que se debe repetir una serie de las mismas acciones una y otra vez en distintas partes del código, pues solamente se toman esas series de acciones se ponen fuera del Loop y se nombran, así cada vez que se necesite realizar la serie de acciones, solo se escribe el nombre. Es una forma de ordenar el código, algunos programadores prefieren establecer un programa mediante funciones, es una manera más ordenada de generar un programa, pues si alguna parte del programa falla, se puede ir a la función que está fallando y repararla.

- Setup. – Es necesario y obligatorio colocarlo al inicio de todo programa al igual que Loop, (de hecho, cada que abres un nuevo archivo, aparecen), en esta función se colocan los inicios de variables, los inicios de algunas bibliotecas y se configuran los pines como entrada y salida. Esto se debe a que la función Setup solo es invocada una vez, después de que se enciende o reinicia la tarjeta. Así que, todo lo que coloquemos dentro de ella solo se ejecutara una vez.
- Loop. – Es una función en donde se coloca (normalmente), el código del programa que queremos se cargue a la tarjeta, dado a que esta función se ejecuta de manera infinita (siempre y cuando este energizada la tarjeta), por lo que sea que queramos que se siga ejecutando una y otra vez se coloca dentro de esta función. Aunque en ocasiones si deseamos que el programa solo se ejecute una sola vez, lo podemos escribir fuera del Loop, se compilara el código, se cargara a la tarjeta y se ejecutara una sola vez.

2.3.2 Variables

El entorno de programación, cuenta con varios tipos de variable, cada una posee características diferentes a las otras, y dependiendo del uso que se desee dar al programa, serán los tipos de variables que se utilizaran. Existen dos tipos de variables, las que son constantes y las que son de datos.

A continuación, se describen las características de algunas de las variables con las que opera el entorno:

Datos. – Son variables que pueden tomar un amplio rango de valores:

- Void. – Se usa solamente en declaraciones de funciones, indica que la función no retorna algún tipo de valor.
- Boolean. – Solo puede almacenar un valor verdadero o falso, ocupa 8 bits.
- Char. – Representa a los caracteres, puede almacenar los caracteres que componen al código ASCII.
- Byte. – Puede contener una variable de tipo entera o un número que solo ocupe 8 bits, es decir, de 0 a 255 en decimal.
- Int. – Se usa para valores enteros, puede contener valores de hasta 16 bits, negativos o positivos.
- Float. – Es para valores que se tomen con punto decimal, este tipo de variables posee más precisión que los de tipo entero, pues otorga un máximo de 7 dígitos de precisión, tiene una talla de 4 bytes, 32 bits de información.
- Double. – Es el doble de preciso que las variables float, pues ofrece una exactitud de 15 dígitos, ocupa 4 bytes de espacio.

Las variables de tipo constante, son aquellas cuyo valor jamás se modifica:

- HIGH. – Es un estado lógico alto, corresponde a un 1 logico.
- LOW. – Es un estado lógico bajo, corresponde a un 0 logico.
- True. – Es una sentencia verdadera en muchos casos se interpreta como un 1 lógico.
- False. – Corresponde a una sentencia falsa, se puede interpretar como un 0 lógico.
- INPUT/OUTPUT. – Son entradas y salidas respectivamente, corresponden a la activación o desactivación de un pin de la tarjeta.

2.3.3 Instrucciones, funciones y estructuras de control del entorno Arduino

El entorno de programación Arduino, cuenta con un amplio catálogo de funciones, y éste se amplía aún más, cuando se cargan bibliotecas al entorno.

Algunas de las estructuras de programación más usuales que se usan en el entorno de programación son los ciclos condicionales conocidos en los procesos de creación de diagramas de flujo, y de programación:

- For. – ciclo condicional, su sintaxis es ***for(inicio, condición, incrementos),*** es usado para realizar procesos que requieran una determinada cantidad de repeticiones.
- If. – El ciclo if representa una toma de decisión evalúa una situación y si es verdadera, el flujo del programa ira en la opción de verdad, si es falsa, el flujo del programa seguirá por el camino de falso. Su sintaxis es la siguiente ***if(condición).***
- While. – Es un ciclo que se repetirá siempre que la condición que es evaluada sea verdadera. Su sintaxis es ***While(condición)***.
- Switch case. – Es una función que evalúa el valor de una condición y si coincide con uno de los valores que se encuentran contenidos en ella, ejecuta el proceso descrito en el valor de la condición. Solo se repite una vez, a menos que se encuentre en una función Loop o While. Su sintaxis es ***While(variable) case:'1'…***
- Break. – Es un paro de programa, cuando se coloca la función Break, el flujo de programa es detenido hasta donde la función break se encuentre.
- Map. – Es una función que otorga el entorno de programación, fue diseñada para simplificar el cálculo de las reglas de tres, es decir, escala valores entre dos escalas definidas. Su sintaxis es ***map(el valor mínimo, el valor máximo, el valor mínimo a convertir, el valor máximo a convertir, la variable a convertir).***
- Pin Mode. – Es una sentencia que establece la configuración de un pin de la tarjeta, se puede configurar como entrada o como salida. Su sintaxis es ***pinMode(pin,configuración)***.
- Analog read. – Es una función utilizado para cuando se desea iniciar una lectura con el convertidor analógico digital, existen 6 pines específicos para esta función, son los pines que se llaman desde A0 hasta A5. Solo se puede realizar una lectura a la vez. Su sintaxis es ***analogRead(pin donde se realizara la lectura).***

- Delay. – Es una función del entorno que permite al programa una pausa de una cantidad definida de tiempo, se dice de "pause", pues en el intervalo en el que la función delay este configurada, la mayoría de las funciones de la tarjeta se detienen (como el convertidor analógico digital, los módulos PWM, etc.). Su sintaxis es ***delay(cantidad de tiempo en milésimas de segundo).***
- highByte. – Toma los ocho bits más significativos de una variable, y los copia en otra variable. Su sintaxis es ***highByte(varible)=variable auxiliar.***
- lowByte. – Toma los ocho bits menos significativos de una variable y los copia en otra variable. ***lowByte(varible)=variable auxiliar.***

2.3.3.1 Serial

La función Serial en el entorno de programación es una herramienta importante, pues permite enviar y recibir información de manera serial mediante el puerto serial. La comunicación serial se establece cuando se envía la envía la información mediante un solo cable, es decir, que se ordena la información digital a manera de que la información quede "formada en fila" pues los paquetes de información (generalmente expresadas en binario) son enviados uno tras otro.

La función distingue entre valores y caracteres, es decir, que la información que podemos enviar a través del puerto serial: pueden ser caracteres o cadenas de caracteres (palabras o letras sin algún valor cuantificable); o variables del programa que contienen un valor numérico cuantificable que puede ser procesado por algún segundo dispositivo.

Algunas de sus funciones se describen a continuación:

- Serial.begin. – Establece la velocidad de transferencia con la que se van a enviar y recibir los datos por el puerto serial, si no se define esta función, la comunicación serial no se llevará a cabo. Su sintaxis es ***Serial.begin(velocidad)*** (9600, es el valor más general).
- Serial.print. – Es una función del entorno de programación, permite enviar uno o más datos por el puerto serial. Su sintaxis es ***Serial.print(valor a enviar)***, con las siguientes reglas:

- Si son caracteres los valores contenidos entre los paréntesis, éstos van entre comillas (“”) si son más de 1 carácter; pues si es un solo carácter se coloca entre apostrofes (‘’).
- Si es un valor contenido en una variable, no lleva comillas, pero puede ir acompañado de una función, que describa el formato en el que se desea visualizar la información, en binario, decimal, hexadecimal. (BIN, DEC, HEX, respectivamente).

- Serial.read. – Es una función que sirve para indicar que se desea leer el puerto serial. Su sintaxis es ***Serial.read() = variable en donde se almacenará la lectura.***

La forma de leer la información no es tan sencilla como enviarla, pues la configuración del entorno establece, que toda la información que llega por el puerto serial es tratada como si fuera un envió de caracteres, es por ello que se crearon funciones para poder convertir la información para que sea más útil.

Uno de ellos es:

- Serial.parse.Int. – Es una función que analiza la información recibida y en caso de que aparezca un carácter que no pertenezca a letras o números, detiene la lectura y brinca a la siguiente instrucción. Su sintaxis es ***Serial.parseInt() = variable en donde se almacenará la lectura.***

2.3.4 Bibliotecas

Las bibliotecas son agrupaciones de código o funciones que se ejecutan para llevar a cabo un proceso, generalmente se crean para facilitar el uso de algunos dispositivos que requieren una serie de pasos específicos para poder funcionar o procesar los datos que obtienen. Las bibliotecas se pueden descargar de la Internet, pues muchas veces quien genera las bibliotecas son personas que se concentrar en hacer funcionar algún dispositivo, y cuando lo logran comparten su trabajo con aquellos que quieran conocerlo. Una vez que se tiene la biblioteca que se desea usar, hay que seguir una serie de pasos para poder cargarla al entorno de programación:

1. Se busca la biblioteca en Internet, comúnmente se encuentra en un archivo comprimido en .zip, aunque también lo hay en .rar solo que para este tipo de archivo hay otra manera de cargarlo al entorno.
2. Para **.zip** se descarga el archivo, se guarda en la computadora, se abre el entorno de programación Arduino, y en la sección de barras de herramientas (ver *ilustración 2.11*), escogemos la opción de programa, se desplegará un menú, y escogemos la penúltima opción que dice incluir biblioteca, va a abrir otro menú, y escogeremos donde dice añadir biblioteca .zip abre una ventana de explorador de archivos, escogemos la ubicación en donde se encuentra la biblioteca y listo.
3. Si el archivo se encuentra en formato .rar, se debe de realizar lo siguiente: descomprimimos el archivo, la carpeta que obtenemos después de descomprimir la copiaremos al escritorio, abrimos el explorador de archivos y entramos a los archivos de programa que se encuentran en C:\\, buscamos la carpeta que dice Arduino, escogemos, abrimos y en la carpeta que dice bibliotecas la abrimos y colocamos la carpeta que dejamos en el escritorio, y listo.

2.3.4.1 Bibliotecas usadas

A continuación, se describen las bibliotecas utilizadas para la elaboración de este proyecto, explicando y describiendo algunos de las instrucciones y funciones descritas por cada biblioteca.

2.3.4.1.1 RTC.lib

Es una biblioteca que proporciona el soporte para el uso de un módulo de tiempo real. La biblioteca se descargó de la página electrónica: *https://github.com/adafruit/RTClib* se encuentra en formato .zip, (lo que facilita su instalación en el entorno). Algunas de sus funciones son:

- RTC_DS1307 RTC. – Esta función inicia una definición del tipo de reloj que estamos usando, es este caso es el reloj DS1307, se escribe fuera de Setup y Loop, en la zona donde se declaran las variables globales para el programa, es decir en la parte de arriba de la zona de trabajo. Su sintaxis es: ***RTC_DS1307 RTC.***

- RTC_MILLIS. – Establece un nombre al registro que lleva el conteo del contador interno del Reloj, se coloca en la zona de las variables globales. Su sintaxis es: ***RTC_MILLIS rtc.*** (Así se nombró (rtc) a la función)
- DateTime. – Sirve para indicarle al reloj, que se desea conocer la hora, la función almacena los valores del tiempo en el momento en el que se solicitó. Su sintaxis es ***DateTime now=rtc.now()***[2]
- Now.hour()/now.minute()/now.second(). – Son funciones que establecen, cada una de las unidades del tiempo, pueden ser desde segundos hasta el año, se almacena su información en variables del programa. Su sintaxis queda: ***s=now.second().*** NOTA: estas funciones, solo se pueden llevar a cabo, una vez que se ha colocado la función (***DateTime.now)*** anteriormente a ellos.

2.3.4.1.2 LiquidCrystal.h

La biblioteca contiene una serie de funciones que nos facilitan la configuración de la pantalla y su manejo, como la forma de colocar la escritura en la pantalla mediante instrucciones sencillas:

- LiquidCrystal.lcd. – La función establece los pines en los que se realizara la comunicación con la tarjeta, por lo que se aconseja, configurar los pines que aún no poseen una aplicación (es decir, estén libres). Se coloca fuera de Setup y Loop, en la zona de variables globales, en la parte superior de la zona de trabajo. Su sintaxis establece el orden de los pines a los que se conectara la tarjeta y corresponden a lo pines de la pantalla: ***LiquidCrystal.lcd(RS, Enable, D4, D5, D6, D7, R/W).***
- Lcd.begin. – Describe las dimensiones de la pantalla. Se coloca en la zona de Setup. Su sintaxis es: ***lcd.begin(Columnas,Filas)***
- Lcd.print. – Muestra la información en la pantalla, toma por default la columna cero y la fila cero (si es que no se ha configurado un punto específico de la pantalla), la instrucción, es capaz de imprimir los espacios, con los que son acompañados la información a mostrar, su sintaxis es ***lcd.print(valor a mostrar)***, para poder mostrar la información existen unas sencillas reglas (al igual que en la transmisión serial):

[2] Nota: rtc, es el nombre asignado para cuando se desea conocer los valores de la fecha, por ello se invoca el nombre de la función como rtc.

- Si son caracteres, los valores contenidos entre los paréntesis, si son más de 1 carácter, éstos van entre comillas (“”); si es un solo carácter, se coloca entre apostrofes (‘’).
- Si es un valor contenido en una variable, no lleva comillas, pero puede ir acompañado de una función, que describa el formato en el que se desea visualizar la información, en binario, decimal, hexadecimal. (BIN, DEC, HEX, respectivamente).

- Lcd.clear. – Borra el texto que se encuentre impreso en la pantalla. Su sintaxis es ***lcd.clear()***.
- Lcd.setCursor. – Configura el lugar en la pantalla en donde se comenzará a imprimir la información, es usado para acomodar la información cuando se desea mostrar en una sola pantalla diversas variables simultáneamente. Su sintaxis es ***lcd.setCursor(columna,fila)***.

2.3.4.1.3 EEPROM.h

Esta biblioteca permite el uso de la memoria EEPROM del microcontrolador, para poder, almacenar los valores de algunas variables, que sean importantes y que no se desee perder la información contenida en ellas cuando exista una interrupción en la alimentación.

Una memoria, es un dispositivo electrónico de alta escala de integración que sirve para guardar información en forma de unos y ceros; a la cantidad de bits que almacena cada una de sus celdas o localidades se le denomina ancho de palabra. Para acceder a la información de cada celda (siempre de manera individual) se utiliza el bus de datos, y para activar una celda se utiliza el bus de direcciones.

La memoria EEPROM con la que cuenta un ATmega328P es de 1Kbyte. Un Kilo, es un factor de multiplicación por 1000, un byte, representa una agrupación de 8 bits, por lo que un KiloByte representa 8000 bits, por lo que esta memoria posee un bus de direcciones para 1000 celdas, el bus de direcciones va de 0 a 999, y su ancho de palabra es de 8 bits. La vida útil de la memoria EEPROM es aproximadamente de 100000 ciclos de lectura o escritura.

La biblioteca permite escribir y leer información en bits en las celdas de la memoria, se debe tener en cuenta el ancho de palabra de la memoria a la hora de almacenar información.

Las instrucciones son:

- EEPROM.write. – Permite escribir información en bits en una localidad de la memoria. Su sintaxis es: ***EEPROM.write(dirección de memoria, variable o valor a almacenar).***
- EEPROM.read. – Establece una lectura del contenido de una celda de la memoria, en una variable. Su sintaxis es: ***EEPROM.read(dirección de la memoria)=variable donde se desea almacenar la información.***
- EEPROM.update. – Actualiza el valor de una variable cuando este cambia, almacena el nuevo valor en la misma localidad de memoria (hay que tener en cuenta la vida útil de la memoria). Su sintaxis es ***EEPROM.update(dirección, valor o variable).***
- EEPROM.get. – Obtiene una variable o valor de la memoria EEPROM (hay que tener en cuenta la vida útil de la memoria). Su sintaxis es ***EEPROM.get(dirección, valor o variable).*** La dirección proporcionada será el punto de inicio de la búsqueda.
- EEPROM.put. – Almacena cualquier tipo de variable o valor de manera automática, cuando se le proporciona la dirección de la cual puede iniciar. Su sintaxis es: ***EEPROM.put(dirección inicial, variable o valor)***. Esta instrucción utiliza la instrucción EEPROM.update, por lo que se recomienda el uso racional de la misma, recordando la vida útil de la memoria EEPROM.

2.3.4.1.4 Wire.h

Es una biblioteca que se encarga de la configuración del protocolo para la comunicación I^2C. La cual es una comunicación serial que se caracteriza por ser del tipo síncrono, en donde existe un dispositivo maestro (tarjeta) y un esclavo (un dispositivo), el dispositivo maestro es el encargado de configurar las condiciones de la comunicación (pone las reglas del juego, él manda), mientras que el dispositivo esclavo solo obedece. Se comunican mediante dos terminales conocidas como SDA y SCL.

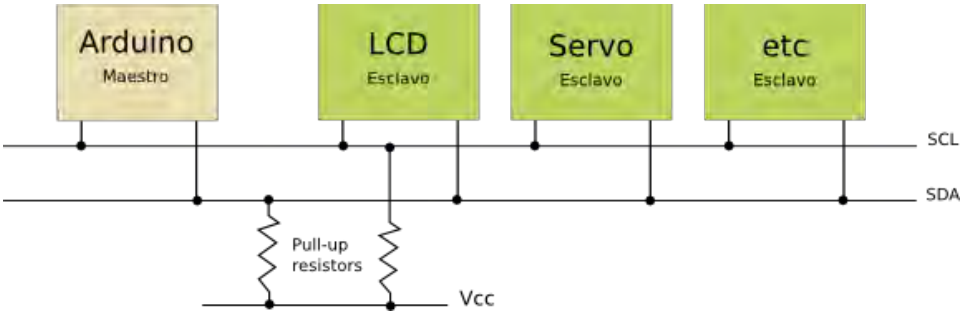


ILUSTRACIÓN 2.12 COMUNICACIÓN I^2C

Los pines para realizar esta comunicación, son los pines llamados A4 y A5, A4 es en donde se conecta SDA, y en A5 se conecta SCL, ver *ilustración 2.11*

SDA. – Terminal por donde se enviarán y recibirán los datos, esta comunicación tiene la característica de que la terminal de transferencia de datos es bidireccional, se puede transmitir y recibir por el mismo puerto, pero no de manera simultánea.

SCL. – Terminal de la señal de sincronía, o de pulso de reloj, esta señal es la que se encarga de coordinar los tiempos entre el dispositivo maestro y el dispositivo esclavo. Normalmente es suministrada por el dispositivo maestro.

Esta biblioteca se agregó al trabajo, es necesaria para el funcionamiento dela biblioteca RTC.lib.

2.3.4.1.5 SD.h

La biblioteca SD brinda el soporte para poder utilizar los recursos de una memoria flash MicroSD, otorga funciones que facilitan tareas como la creación, escritura y lectura, de un archivo de texto en la memoria. La biblioteca contiene su propio protocolo para poder escribir o leer información en la tarjeta, lo primero que se debe realizar es, la creación de un archivo en la memoria, para después ir al archivo, abrirlo para poder escribir o leer, al terminar hay cerrar el archivo. El archivo que se genera en la memoria es del tipo .txt, es un archivo de texto sin formato, ocupa muy poco espacio (algunos Kbyte de información).

Las funciones usualmente empleadas de la biblioteca son:

- File. – Permite la creación de un archivo, solamente se nombra y listo. Su sintaxis es File “nombre del archivo”.

- SD.open. – Es una instrucción que siempre se coloca en el inicio de cualquier lectura o escritura, le indica al sistema que está disponible el archivo para lectura o escritura. Su sintaxis es ***SD.open(nombre del archivo)***.
- File_write. – Esta instrucción permite la lectura y escritura en el archivo, es el habilitador del archivo. Su sintaxis es ***File_write.***
- Print (). – Es una función que permite la escritura de texto o el valor de una variable en un archivo de la tarjeta SD. Su sintaxis es ***archivo.print(valoriable ó texto).***
- Read(). – Indica la lectura de la información almacenada en la memoria, una vez que se almacena el valor de una variable en la memoria SD, la información es convertida a un formato de caracteres, o string, así que se debe tomar esta consideración al leer información de un archivo en la memoria SD. Su sintaxis es ***archivo.lectura();***
- Close (). – cierra el archivo, impide la edición del archivo, solo hasta que se vuelva a abrir el archivo. Su sintaxis es ***archivo.close()***.

2.3.4.1.6 SPI.h

La biblioteca SPI permite la comunicación con dispositivos que utilicen el protocolo de comunicación SPI por sus siglas en inglés, Serial Perpheral Interface, el cual es un medio de transferencia de datos de tipo síncrono, donde existe un dispositivo maestro y un dispositivo esclavo. La comunicación se establece mediante 4 líneas de comunicación:

MISO (Master In Slave Out). - Es la línea donde el dispositivo esclavo envía la información al dispositivo maestro.

MOSI (Master Out Slave In). – Es la línea donde el dispositivo maestro envía la información al dispositivo esclavo.

SCK (Serial Clock). – Es la señal de sincronía que establece el dispositivo maestro.

SS (Slave Select). – Es para escoger al dispositivo esclavo.

Se añadió el uso de esta biblioteca al programa, pues es necesaria para poder llevar a cabo la comunicación con la memoria MicroSD.

2.4 Desarrollo de aplicaciones en Android

El sistema operativo Android, es de los más usados por los teléfonos inteligentes, cuenta con una gran cantidad de usuarios, y por ello posee una gran cantidad de aplicaciones, así como entornos de programación que ayudan a la elaboración de aplicaciones.

Para poder diseñar una aplicación, primero deberemos abordar algunas herramientas que nos facilitaran esta tarea.

Como una aplicación es una ejecución de una serie de instrucciones, para poder llevar a cabo un procedimiento u obtener alguna información o resultado. Para desarrollar aplicaciones en Android existen muchos lenguajes de programación como java, C, C++, BASIC, entre otros. Y para los que no conocen ni dominan los lenguajes de programación, existen programas que ayudan en la elaboración de aplicaciones, mediante el armado por bloques de instrucciones. Existen ventajas y desventajas en ambas formas de desarrollo, en una (la primera), es necesario conocer del lenguaje de programación sobre el que esté basado el programa, y en la otra se carece de elementos específicos, es decir, que solo se cuenta con los recursos con los que está conformado el programa, por ejemplo: Si un programa no cuenta con la interfaz para un dispositivo Bluetooth, será muy difícil realizar una aplicación que requiera de una comunicación por Bluetooth, pues este programa no posee el soporte necesario para brindar esa característica. Es por eso que a la hora de escoger un programa hay muchos elementos que se deben de tomar en cuenta al momento de realizar la selección.

Para el desarrollo de este proyecto se hizo uso del entorno en línea “App Inventor 2”, un programa de uso libre, creado por el MIT (Massachusetts Institute of Tecnology) y la compañía Google. Es un programa que opera bajo la programación mediante bloques, en donde escogemos los segmentos de código y los unimos con conectores, para crear algoritmos o programas completos.

App inventor 2, consta de 2 ventas, una es la ventana de diseño (*ilustración 2.13*) y otra es la ventana de configuración de bloques (*ilustración 2.14*).

Ventana de Diseño. – En esta ventana trabajaremos con los elementos visuales de nuestra aplicación, tales como botones, imágenes, casillas de texto, avisos, fondos, orientaciones,

etc. Esta ventana va de la mano con la ventana de configuración, a cada elemento contenido en la ventana de diseño, le corresponde una configuración especifica que aparecerá en la ventana de configuración; la interfaz que maneja el programa es sencilla de usar, solo hay que escoger un elemento de la paleta de elementos (en la parte izquierda de la pantalla) y arrastrarlo hasta la zona de trabajo.

- Paleta de elementos: en esta parte de la pantalla, encontraremos los elementos que podremos agregar a la aplicación como botones, cuadros para capturar texto, deslizadores, avisos, alarmas, notificaciones.
- Zona de trabajo de la aplicación: es una pantalla como la de un celular con tres botones abajo, el fondo negro, la barra superior de estado dice “screen”. Todos los elementos que estén visibles, se verán en la pantalla, y como quede esta zona de trabajo, será como se verá en el celular.
- Barra de configuraciones: en este apartado encontraremos celdas de opciones que cambiaran conforme al elemento seleccionado, pues en esta parte modificaremos las características de los elementos que se hayan colocado en la zona de trabajo, como el espacio que ocuparan en pantalla, el color, el texto en ellos, el tipo de letra, el tamaño de la letra, la alineación, etc.

Ventana de programación por bloques. – En esta pantalla escribiremos el código que gobernará el funcionamiento de la aplicación. Se ensamblarán bloques de funciones dentro de otros bloques para formar el código de la aplicación. En esta pantalla existen dos partes:

- Bloques: se encuentra a la izquierda de la pantalla, contiene varios submenús, más los nombres de los componentes que hemos agregado en la ventana de diseño, al escoger cualquiera de los elementos agregados, veremos las funciones en bloques que pueden realizar, como cambiar de color, tamaño, o la función específica que realiza, por ejemplo, un bluetooth, envía y recibe información, se encontraran bloques que digan recibir… o enviar…
- Vista o zona de bloques: es la sección en blanco, tiene dos elementos en la orilla de la pantalla, uno es una mochila, que sirve para almacenar temporalmente bloques de código para usarlos en otros programas (y así ya no tener que volverlos a conectar) y un bote de basura en donde los bloques que no funcionen o ya no se necesiten

serán colocados en él, para su eliminación. Y el espacio en blanco es la zona en donde podemos colocar los bloques para después conectarlos y crear programas.

Una vez que nuestra aplicación ya está lista, tanto la parte visual como la de programación, ahora solo queda crear el archivo que será cargado a un teléfono con sistema operativo Android. Para realizar esto, existen dos maneras:

- Creación de la aplicación y descarga con el navegador de la computadora, solo hay que generar el archivo .APK, para ello colocamos el puntero en la sección Build de la barra de herramientas principal (Se encuentra en cualquiera de las pantallas), aparecerán dos opciones, se escoge la segunda, aparecerá una barra de estado y cuando se complete al 100% aparecerá la ventana de descarga.
- Creación de la aplicación y descarga directa al teléfono mediante un código QR, el cual contiene la dirección para iniciar la descarga del archivo desde internet, la ventaja es que se guarda directamente en el teléfono inteligente. Un requisito, el teléfono debe de contar con una aplicación que lea los códigos QR.

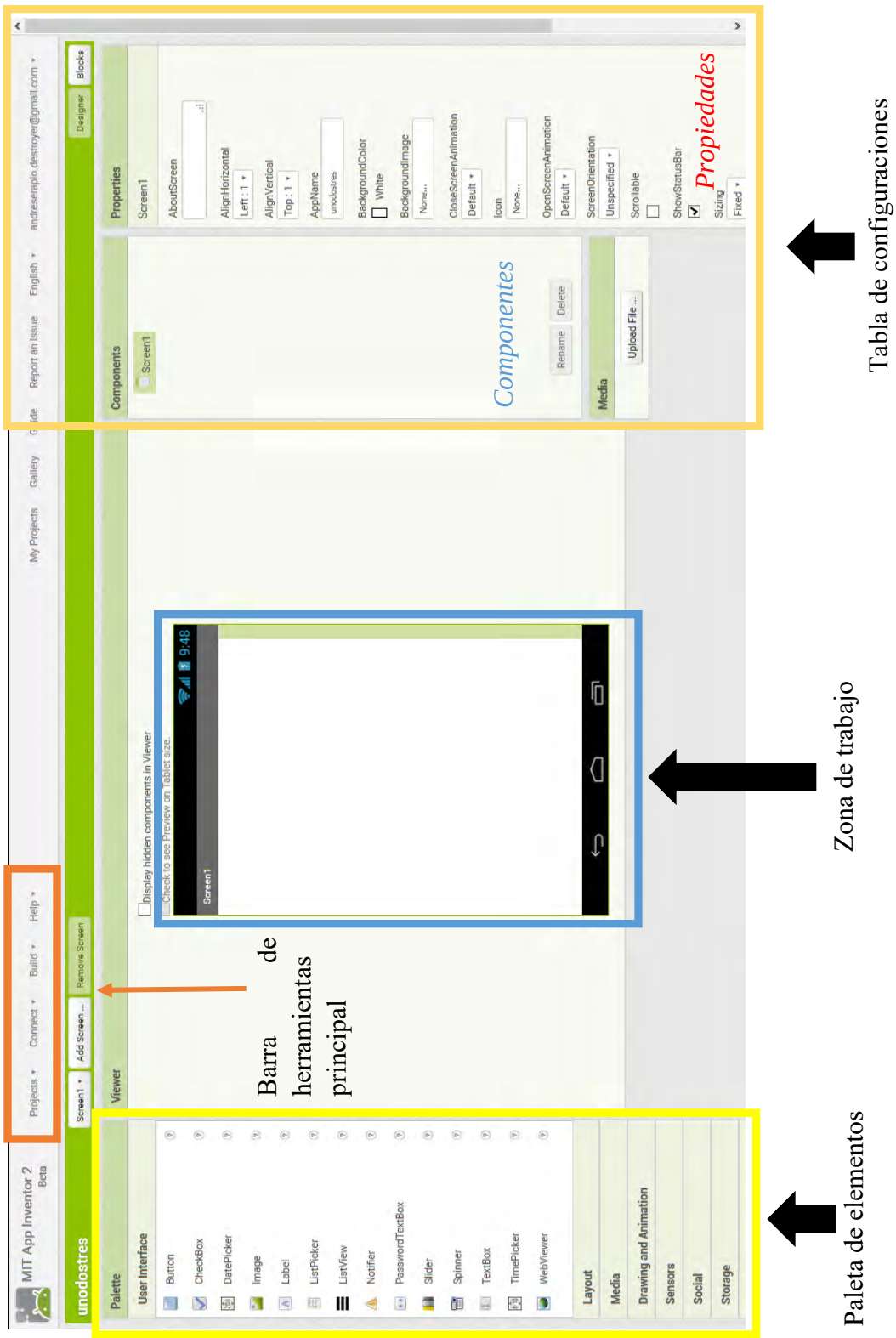


ILUSTRACIÓN 2.13 VENTANA DE DISEÑO

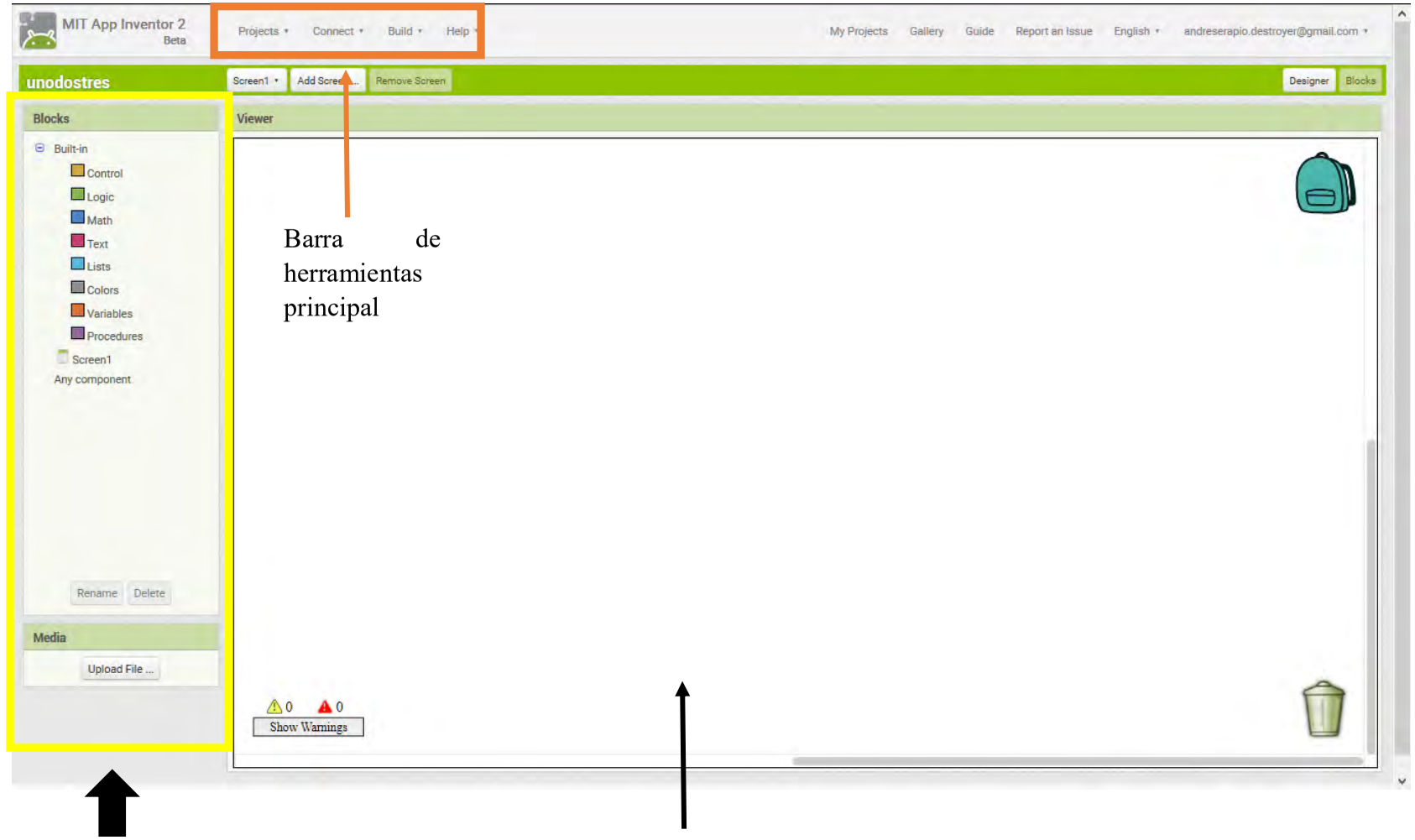


ILUSTRACIÓN 2.14 VENTANA DE BLOQUES

2.5 Dispositivos y/o sensores electrónicos

Dentro del mundo de la electrónica, existen una infinidad de dispositivos que se encargan de realizar una amplia gama de tareas, las hay sencillas, hasta específicas, desde medir la intensidad de la luminosidad solar, hasta contar las partículas disueltas en el aire.

Dada la modernización de la producción en masa, los dispositivos se han vuelto más accesibles, lo que conlleva a una mayor facilidad para obtener los dispositivos a un bajo precio; la participación y consumo por parte de las personas se ha incrementado, pues al existir una mayor competencia entre cantidades y precios, las personas pueden escoger los dispositivos según su capacidad financiera, lo que resulta en un alto interés en el desarrollo de algún dispositivo, o, la realización de un experimento meramente por el gusto de hacerlo.

Es por ello, que la información alrededor de estos dispositivos es bastante amplia, pues, ahora existen muchas personas que se encuentran usando el mismo dispositivo o alguno parecido, lo que nos lleva a una colaboración colectiva, pues el alto desarrollo tecnológico ha permitido que sean menos las personas que no cuentan con una conexión a internet, lo que ha incrementado la interacción virtual, que para este caso en específico, es de gran ayuda, pues cuando te encuentras con algún problema, durante o al usar un dispositivo, puedes tener la ventaja de comentarlo en un blog, o sitio oficial (como con el que cuenta el entorno de Arduino) y preguntar a los demás participantes si han tenido algún problema parecido, y entre tantos participantes muchas veces se llega a una solución viable.

En los dispositivos a usar, las características más generales de todos ellos son:

- Origen chino (la gran mayoría), pues es el principal país productor/maquilador de tecnología, y gracias a su gigantesca fuerza laboral y a los precios de la producción en masa, que se abaratan cada día más. Lo que significa que cada día que pasa la tecnología está más al alcance de las personas (hablando financieramente).
- Su alimentación eléctrica se comprende en el rango de 3.3V a 5V, en la mayoría de los dispositivos electrónicos.
- Se necesita de algún sistema de control, como una tarjeta de desarrollo, o un microcontrolador, para poder interpretar la información que produce el dispositivo.

- La comunicación que establecen puede ser mediante la comunicación serial asíncrona[3], serial síncrona y protocolos especiales de comunicación, como I^2C.
- Su funcionamiento dependerá de la aplicación sobre la cual este construido.

De entre los sensores más usados se encuentran las pantallas LCD, los relojes de tiempo real, los acelerómetros (que poseen una gran cantidad de aplicaciones), los sensores de luz, de temperatura, de humedad, de ritmo y pulso cardiaco, entre muchos otros, estos son algunos de los dispositivos más usados, y de los que el mercado está inundado, pues sus aplicaciones pueden calificar para algún evento diario que se desee monitorear, medir, estandarizar, o simplemente por el gusto de ver que es lo que sucede cuando.

2.5.1 Sensores de corriente

ILUSTRACIÓN 2.15 SENSOR SCT-013-30A

Los sensores de corriente entregan una cantidad de voltaje directamente proporcional a la corriente que están midiendo, se realizó una búsqueda de dispositivos y se encontraron dos de precios accesibles y con buenas características y bastante información acerca de su funcionamiento y utilización.

[3] La comunicación Serial asíncrona, es un proceso de transferencia de información, donde los dispositivos se comunican de manera configurada, es decir, se establecen las características de la comunicación, como el tiempo, la manera de recibir y enviar los datos.
En la comunicación síncrona, se establecen dispositivos esclavos y un maestro, donde el maestro es quien proporciona la señal de reloj (de sincronía) de la comunicación, y el esclavo solo se comunica con el maestro cuando el pulso de reloj es identificado por el esclavo. Un dispositivo esclavo no puede enviar una señal de reloj.

Sensor SCT- 013 – 30 A (*ilustración 2.15*). – Es de tipo no invasivo, pues posee un gancho que se ajusta fácilmente a los calibres normales de los cables de energía de una casa-hogar. Su funcionamiento se basa en el principio de funcionamiento de las bobinas de inducción, pues mediante la inducción eléctrica, genera un voltaje, que es directamente proporcional a la cantidad de corriente que está circulando a través del cable conductor que se está midiendo (que para este caso es la toma principal del hogar). Para su lectura, se necesitará de un convertidor analógico digital, que mida el voltaje y lo convierta en una señal digital, para que los datos puedan ser procesados e interpretados por un programa. Su rango de medición es de 0A hasta 30A. Sus características (Chen, 2011) principales son:

- Soporta hasta 30 Amperes AC.
- Fácil instalación, pues solo se abre al gancho y se pone el conductor.
- Rango de temperatura de operación -25°C a 70°C.
- Voltaje de salida 0V – 1V.
- NO es lineal.
- Dimensiones máximas del cable, diámetro de 13mm.

ILUSTRACIÓN 2.16 SENSOR ACS 712

Sensor ACS 712 (*ilustración 2.16*). – Es un sensor de corriente de tipo invasivo, pues es necesario abrir una sección del cable conductor, para insertar ambas terminales en las entradas del sensor, para que pueda obtener una lectura de la corriente que se encuentra circulando por el cable conductor; opera bajo el principio del efecto Hall, el cual establece que cuando se tiene una corriente bajo la influencia de un campo magnético se obtendrá un

voltaje Hall, el cual es directamente proporcional a la corriente eléctrica, dado este efecto, es como el sensor lo aprovecha para brindar una lectura en intensidad de voltaje la cual se puede conectar a un convertidor analógico digital para poder procesar los datos otorgados por el sensor y cuantificar la cantidad de corriente. Sus características (Allegro Microsystems, 2012) son:

- 5V de alimentación.
- 66 a 185 mV/A resolución (depende del modelo).
- Funciona para AC y DC.
- Temperatura de operación -40°C a 85°C.
- Soporta hasta 20 A (en su modelo para 20 A, existe también para 30 A).

2.5.1.1 Funcionamiento SCT-013

El sensor tiene tres cables que salen de él, uno es un pin vacío y los otros dos son la salida que entrega una cantidad de voltaje directamente proporcional a la cantidad de corriente que se encuentra atravesando el cable conductor, para poder medir su lectura se debe de enviar su señal de salida a la entrada de un convertidor analógico digital, es lo que se recomienda. Una vez conectado el sensor entregara una cantidad de voltaje de entre 0V y 1V (es el funcionamiento del sensor, ya que la cantidad máxima que puede entregar a la salida es 1V), que correspondería a una escala de 0A a 30A, es una regla de tres, lo que se recomienda realizar es lo siguiente:

$$i = (V_L)\left(\frac{30}{1}\right) \tag{2.1}$$

Dónde:

- i. – Corriente eléctrica (Amperes)
- V_L. – Lectura de voltaje del ADC (Volts)
- (30/1). – Constante de conversión (para este sensor en específico).

Se sugiere este procedimiento, pues es el funcionamiento del sensor según datos del fabricante.

2.5.1.2 Funcionamiento ACS 712 20A

El sensor ACS 712, entrega una salida de voltaje que es directamente proporcional a la cantidad de corriente. La característica de este sensor es que es lineal, y tiene una margen de error del 1.5%. El ACS 712 cuenta con tres terminales, VCC, GND y OUT. VCC y GND son las terminales de alimentación del dispositivo, OUT es la salida del sensor, su conexión con cualquier tarjeta es fácil, lo complicado es conectar la entrada del sensor, pues recordemos, que es un sensor de tipo invasivo, lo que quiere decir que es necesario abrir un segmento del conductor para conectarlo en serie con la entrada del sensor para que este tome las lecturas de la corriente. La salida del sensor se podría conectar a una entrada de un convertidor analógico digital, para poder convertir la información analógica en digital y así facilitar el procesamiento de los datos obtenidos pues la salida que se obtiene del sensor, es una cantidad de voltaje que se encuentra en el rango de alimentación del dispositivo, que va de 0V a 5V, cuyo valor corresponderá a un valor equivalente a la escala de 0 a 20A (pues es el modelo que estamos ocupando). Para su utilización se propone una función específica en el entorno de programación, el algoritmo debe de registrar los mínimos y los máximos como si se tratara de un voltaje pico a pico, para tener un valor de corriente en voltaje Pico a pico aproximado.

El resultado obtenido de la función serán los volts equivalentes a los amperes, pero para convertir el voltaje obtenido a los amperes que se necesitan para calcular la cantidad de corriente, debemos convertir ese voltaje a un valor RMS[4] (Root Mean Square, raíz media cuadrada), el cual es un valor que se toma cuando se tienen señales variantes en el tiempo, para ello debemos de dividir el valor entre 2 (pues solo vamos a trabajar con la parte positiva de la señal) para después multiplicarlo por 0.7071. El valor obtenido será el voltaje RMS del cual solo debemos de dividirlo entre la constante del sensor para obtener la corriente RMS.

$$Voltaje_{RMS} = \frac{Voltaje_{pico\ a\ pico}}{2} * \frac{\sqrt{2}}{2} \tag{2.2}$$

[4] Valor RMS. – Es el valor efectivo, es decir, la magnitud de voltaje o corriente, que realiza un trabajo que es equivalente al aplicado con una fuente de directa sobre una carga resistiva.

El sensor que estamos empleando es de 20 Amperes, por lo que la constante es: 100mV/A (por cada 100mV que el sensor detecte, habrá 1 Ampere). Solo resta realizar la operación para poder conocer el valor de la corriente RMS:

$$I_{RMS} = \frac{Voltaje_{Rms}}{100mV/A} \tag{2.3}$$

2.5.2 Módulo Bluetooth

ILUSTRACIÓN 2.17 HC-05

La comunicación por bluetooth se realiza mediante el envío de información por radio frecuencias de alta intensidad del rango de Giga Hertz (GHz), lo que brinda una comunicación de manera inalámbrica entre dispositivos. El módulo más vendido y más usado en el mercado es el módulo HC-05 (*ilustración 2.17*), el cual funciona como receptor y transmisor, es sencillo de utilizar y es compatible con una amplia variedad de teléfonos inteligentes (excepto Apple). Sus terminales para su conexión son (*ilustración 2.18*), donde los pines más importantes son los de alimentación VCC y GND, donde VCC se puede conectar a 5V y GND es tierra (punto de conexión común en todos los dispositivos) y las de comunicación TX y RX transmisión y recepción respectivamente, sirven para recibir y transmitir los datos.

ILUSTRACIÓN 2.18 PINES DE CONEXIÓN HC-05

Y sus características principales (ITeadStudio, 2010) son:

- 3Mbps velocidad de envió
- Frecuencia de transmisión 2.4GHz
- 5V de alimentación
- Sensibilidad -80dBm
- Antena incluida

2.5.3 Relojes o contadores

ILUSTRACIÓN 2.19 TINY RTC

Se necesita de un dispositivo que pueda brindarle a la tarjeta la información relacionada con la fecha y la hora, pues como se pretende que sea un monitoreo constante del consumo eléctrico, es importante conocer la hora para poder brindar horarios para conocer en qué momento guardar los datos obtenidos y así brindar un respaldo en caso de que se pierda el suministro de energía eléctrica.

Dentro de estos dispositivos se cuenta con los módulos Tiny-RTC (*ilustración 2.19)* que son los comúnmente más usados. Entre sus principales características, cuentan con un respaldo de energía, una batería de 3V, que posee una duración aproximada de 20 años, por lo que una vez configurada la hora, se mantendrá hasta que la pila de respaldo se agote. Se comunica de manera serial síncrona, pues necesita de una señal que le indique en que momento debe mandar la información. Sus características (Elecrow, 2014) más importantes son:

- Diseño basado en el chip DS1307
- Trabaja de manera automática, diferencia ente año normal y bisiesto
- Se ajusta automáticamente cuando hay meses de 30 o 31 Días
- Puede operar en formato de 24 horas o 12 con A.M. y P.M
- Comunicación I^2C
- Consumo menor a 500nA con Oscilador funcionando

Sus pines de conexión:

- SCL. – Señal de reloj, para comunicarse con el dispositivo maestro.
- SDA. – Terminal de transferencia de datos.
- VCC y GND. – Alimentación.

2.5.4 Pantallas

Una pantalla ofrece una interfaz visual que ayuda al usuario para que se pueda enterar de los resultados que entrega el prototipo, además ayuda a la hora de tomar datos de manera visual, en la categoría de las pantallas existen muchos tipos de ellas, pues las hay de pantalla táctil, de alta resolución, LCD con iluminación para ver la información en la noche.

El modelo más “básico” que hay sobre las pantallas es una pantalla LCD de 16 columnas

ILUSTRACIÓN 2.20 LCD 16X2

por dos filas (*ilustración 2.20*), se comunica mediante un protocolo de comunicación en paralelo. Logra imprimir en la pantalla, una amplia gama de caracteres, entre los que destacan: letras mayúsculas, minúsculas, números del 0 al 9 y algunos caracteres especiales, signos de interrogación, admiración, asterisco, paréntesis, etc. (QIU, 2008). Las terminales para su conexión se encuentran enunciadas en la parte superior de la *ilustración 2.20,* de las cuales en la mayor parte de los casos solo se utilizan:

- VSS. – Voltaje a surtidor, es decir, esta terminal va a tierra.
- VDD. – Voltaje de drenado, esta terminal va a 5V.
- V0. – Se usa para controlar el brillo de la pantalla (generalmente se coloca un potenciómetro de 10Komhs para variar el brillo).
- Las terminales D4, D5, D6 y D7 son las que se usan para la comunicación entre el controlador de la pantalla y la tarjeta de desarrollo.
- A y K. – Son terminales de alimentación de la iluminación de fondo (+, -, respectivamente).

2.5.5 Módulo memoria MicroSD

ILUSTRACIÓN 2.21 MÓDULO MICROSD

Es un adaptador para una memoria flash MicroSD, brinda la utilidad de poder almacenar valores numéricos o cadenas de texto en la memoria mediante la creación y edición de uno o más archivos de texto ".txt", el cual carece de formato.

Una memoria externa facilita la tarea de generar registros históricos sobre alguna variable que se encuentre monitoreando constantemente en un proceso de control. Pues la mayoría de los dispositivos de control como los microcontroladores, carecen de una gran capacidad de almacenamiento, pues en la concepción de su diseño, solo se pensaron para lo más necesario, por lo que no pueden almacenar en sus registros de memoria, información estática, que solo se almacena y se almacena, no se le realiza algún tipo de proceso. Es por ello que la incorporación de una memoria externa para el almacenamiento de información es excelente para una aplicación como la generación de un histórico.

ILUSTRACIÓN 2.22 TERMINALES DE CONEXIÓN MÓDULO MICROSD

El módulo se comunica a través del protocolo de comunicación síncrona SPI. El adaptador posee 6 terminales para su conexión, donde VCC y GND son terminales de alimentación.

SCK. – Es donde se coloca la entrada del pulso de reloj.

MOSI. – Terminal por la que se envía la información.

MISO. – Terminal en la que el dispositivo recibe la información proveniente del sistema de control.

2.6 Método Bland-Altman

Es un método gráfico para el análisis estadístico, del grado de concordancia o similitud que existe entre los resultados de dos mediciones, que son obtenidas por dos instrumentos distintos que realizan un monitoreo constante sobre un mismo evento. Donde, el evento que se encuentran midiendo, es variante en el tiempo, quiere decir que los resultados no son constantes. Por ejemplo, el grado de cansancio que presenta una persona al final del día.

Este método a diferencia de los coeficientes de correlación: de la (r) de Pearson, que solo describe la razón de cambio entre dos series de resultados. Y del índice de correlación intraclase, que describe el grado de concordancia entre una serie de resultados que se obtienen de la medición de un evento constante. Esta creado para obtener la relación entre las magnitudes de los resultados obtenidos de monitorear un evento que no es constante en el tiempo.

Para poder utilizar el método es necesario aplicar un procedimiento a los resultados obtenidos de las mediciones, primero, se obtiene la diferencia entre la primera lectura con respecto a la segunda, y segundo, se realiza una media entre las mediciones, estos dos pasos se aplican a todas las parejas de datos obtenidos de las mediciones. El proceso descrito se basa en las ecuaciones del método:

$$S_{(x,y)} = \frac{S_1 + S_2}{2}, S_1 - S_2 \tag{2.4}$$

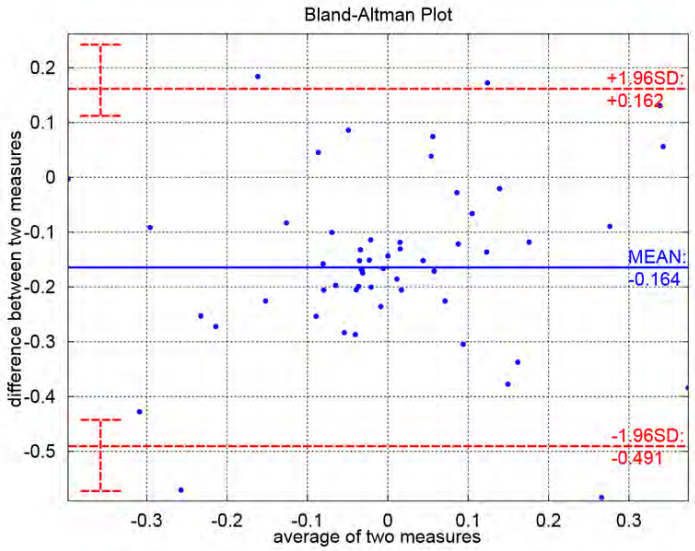


ILUSTRACIÓN 2.23 EJEMPLO GRAFICA BLAND-ALTMAN

El método Bland-Altman toma los valores y genera un gráfico como el mostrado arriba, donde, los puntos son los resultados de las comparaciones realizadas en el método, donde se comparan las diferencias entre las parejas de las mediciones, con respecto al promedio de las mismas. Las líneas horizontales, son los límites de confianza que genera el método, para poder indicar si existe un buen grado de confianza en las mediciones. En el eje Y (el eje vertical), mostrara las diferencias entre los resultados de las mediciones comparadas. Y en el eje X (eje horizontal), se muestra el promedio de los resultados. Dependiendo de la ubicación en donde se concentren los puntos, será el grado de parecido que existe entre los dispositivos. Si la mayoría de puntos se encuentra dentro de los límites horizontales, quiere decir que los datos obtenidos por los dos sistemas son muy parecidos, por lo que el grado de concordancia entre los sistemas es muy alto.

Capítulo 3. Desarrollo experimental de la propuesta

El proyecto aquí propuesto, consiste en la creación de un dispositivo, que sea capaz de tomar lecturas del consumo eléctrico, y además realizar la sumatoria del consumo eléctrico en un hogar o inmueble, dentro del periodo en curso, además brindará un estimado del precio a pagar por dicho consumo, y como punto final, mostrará los días faltantes para el término del periodo consumo. Con el fin de poder brindar al usuario una herramienta, con una interfaz en un teléfono inteligente con sistema operativo Android, que lo ayude en la visualización del consumo y el gasto energético en términos monetarios, para que el usuario logre establecer una gestión racional de los recursos energéticos.

Lo que lleva a pensar ¿Cómo hacerlo?, pues para ello, primero se realizó un diagrama de bloques que describiera las características del dispositivo, para después generar un diagrama de flujo más concentrado, donde se contuvieran los diagramas de flujo de cada una de las funciones que debe realizar el dispositivo. El segundo paso fue realizar una investigación en la Internet, para identificar y seleccionar los diferentes componentes necesarios para el desarrollo experimental del sistema propuesto.

Los componentes se encuentran descritos en el capítulo 2 de este trabajo, y son los dispositivos que se emplearon en la construcción de los prototipos de esta propuesta.

3.1 Algoritmo

Un algoritmo es un conjunto de instrucciones que realizadas en orden permiten la solución de un problema.

Para mostrar a grandes rasgos lo que el algoritmo para esta propuesta debe cumplir, se ilustrara mediante un diagrama a bloques, para después mostrar el diagrama de flujo del programa.

Los diagramas a bloques son una representación gráfica de las relaciones que tienen las variables en un sistema, se usa para mostrar el flujo de las instrucciones y las funciones realizadas por los componentes del sistema.

3.1.1 Diagrama a bloques

El diagrama a bloques de la propuesta es el siguiente:

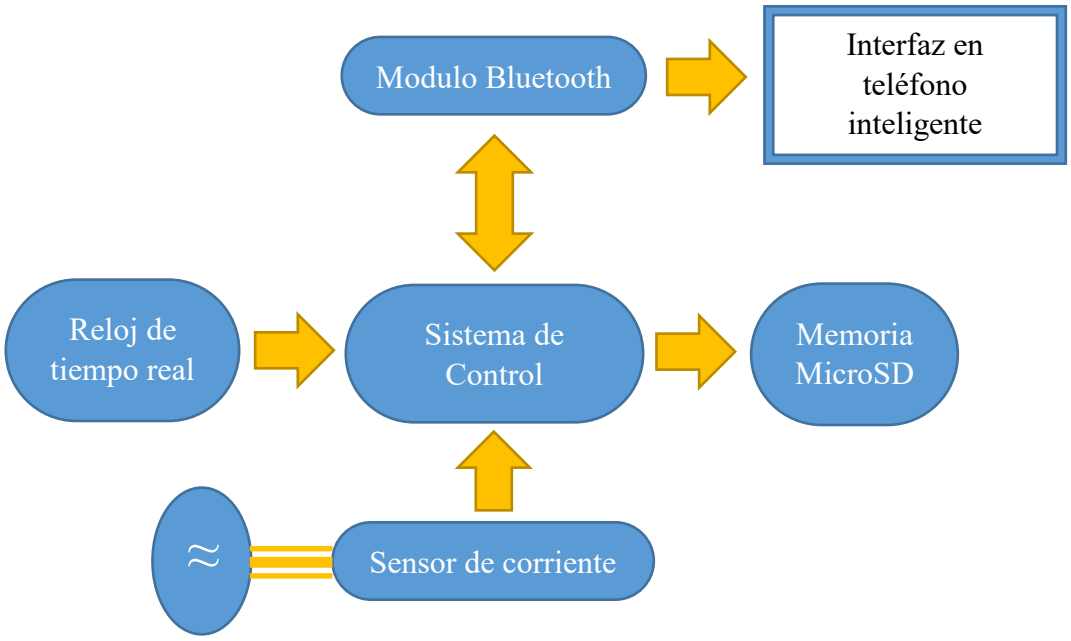


ILUSTRACIÓN 3.1 DIAGRAMA A BLOQUES

Dónde:

- Sistema de Control: es la tarjeta Arduino UNO.
- Módulo Bluetooth: es el módulo HC-05, se conecta a las terminales de la tarjeta Arduino en los pines 0 (Rx) y 1 (Tx), donde el pin Rx del módulo HC-05, se conecta al pin 1 de la tarjeta (RX-Tx). Y el pin Tx del módulo se conecta al pin 0 de la tarjeta, (Tx-Rx).
- Sensor de corriente: son los sensores utilizados en la propuesta, el SCT-013 y el ACS 317, donde, la salida de información de cada dispositivo se conectará a una terminal del convertidor analógico digital, que será la terminal A0 de la tarjeta.
- Reloj de tiempo real: es el módulo Tiny RTC, el cual requiere de una previa configuración para que comience a trabajar. Se conecta a la tarjeta mediante dos líneas de datos, los pines del módulo SDA y SCL, se conectarán a los pines A4 y A5 de la tarjeta, respectivamente.
- Memoria MicroSD: es el adaptador para una tarjeta de memoria MicroSD, se comunica mediante 4 terminales, MISO, MOSI, SCK, CS, los cuales se conectan a

la tarjeta Arduino, los pines son, para MISO-pin12, MOSI-pin 11, SCK-pin 13 y SC-pin 4.

- Interfaz en teléfono inteligente: es la aplicación desarrollada con App Inventor 2 para el sistema operativo Android.

3.1.1 Diagrama de Flujo

Un diagrama de flujo es una representación del proceso que seguirá el algoritmo para resolver el problema. Un diagrama de flujo es más descriptivo en los procesos de control, selección, envío, recepción y guardado de la información, en comparación con el diagrama a bloques. Los procesos de cada una de las funciones del algoritmo se describirán a continuación:

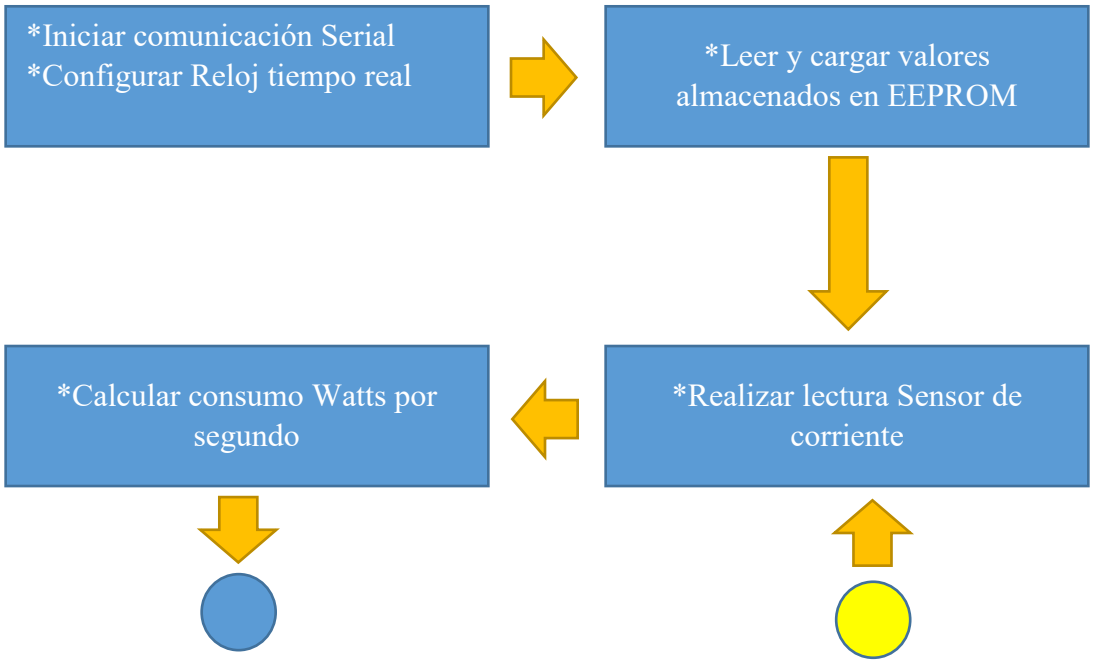


ILUSTRACIÓN 3.2A DIAGRAMA DE FLUJO DE ALGORITMO

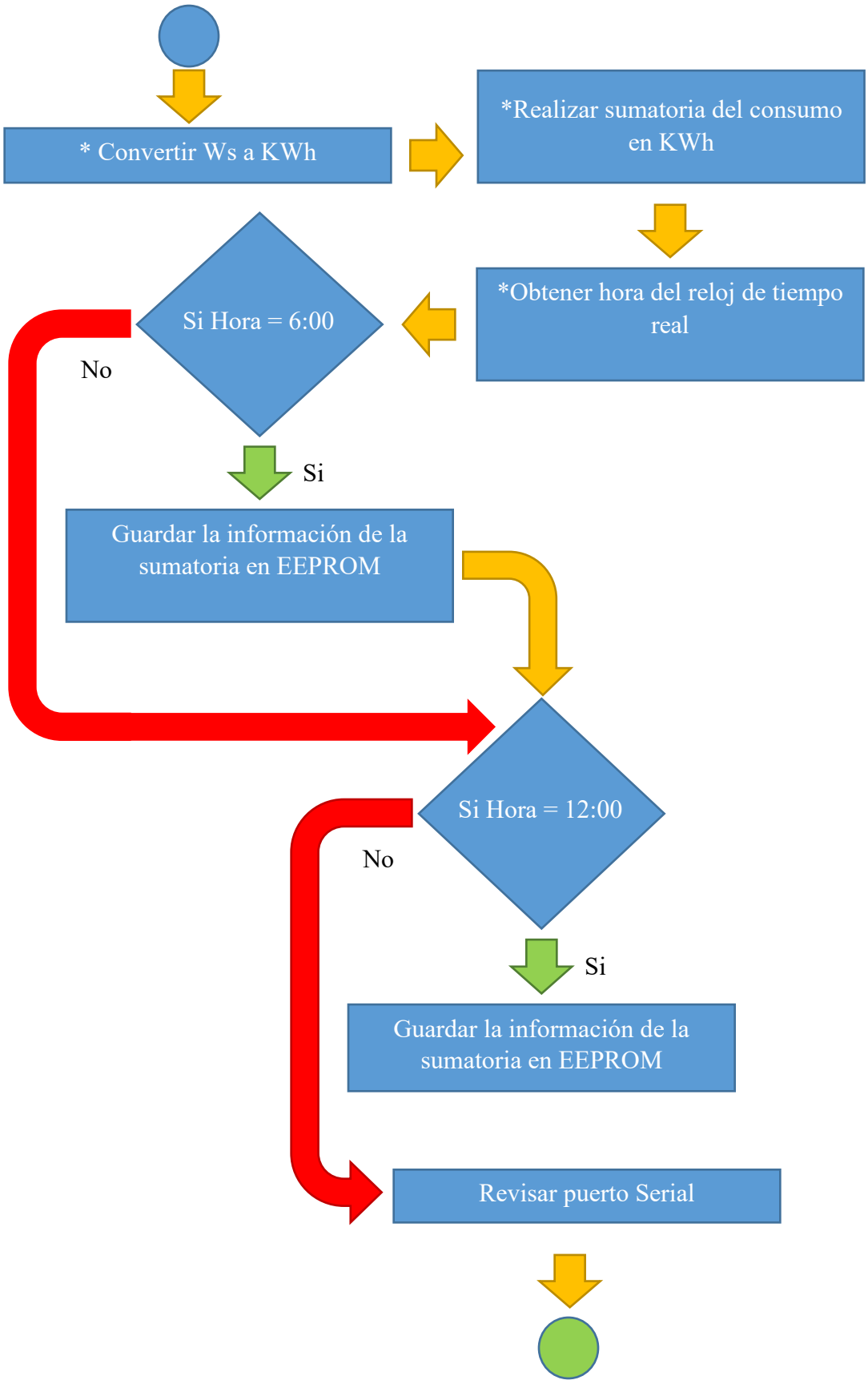


ILUSTRACIÓN 3.2B DIAGRAMA DE FLUJO DE ALGORITMO

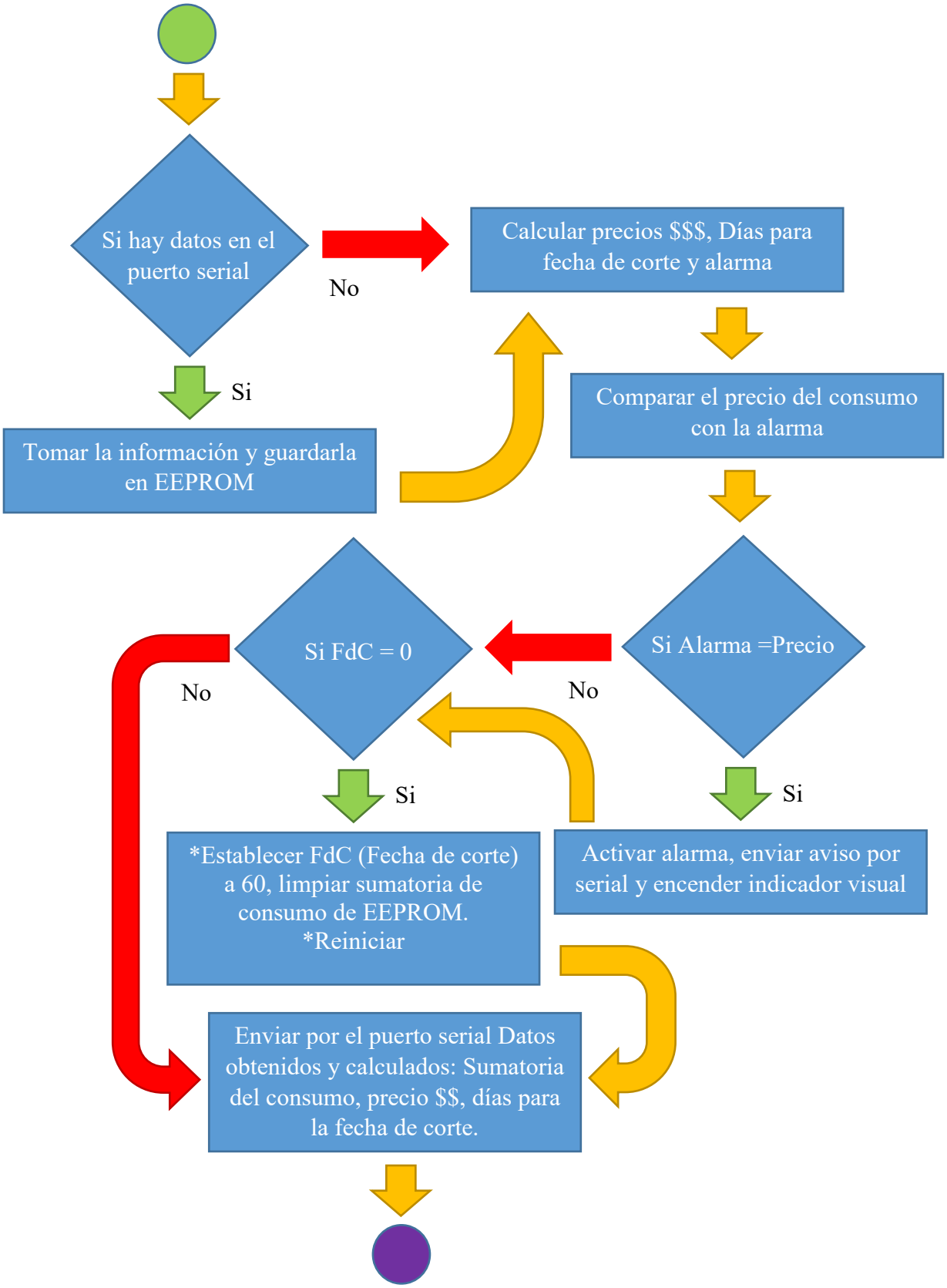


ILUSTRACIÓN 3.2C DIAGRAMA DE FLUJO DE ALGORITMO

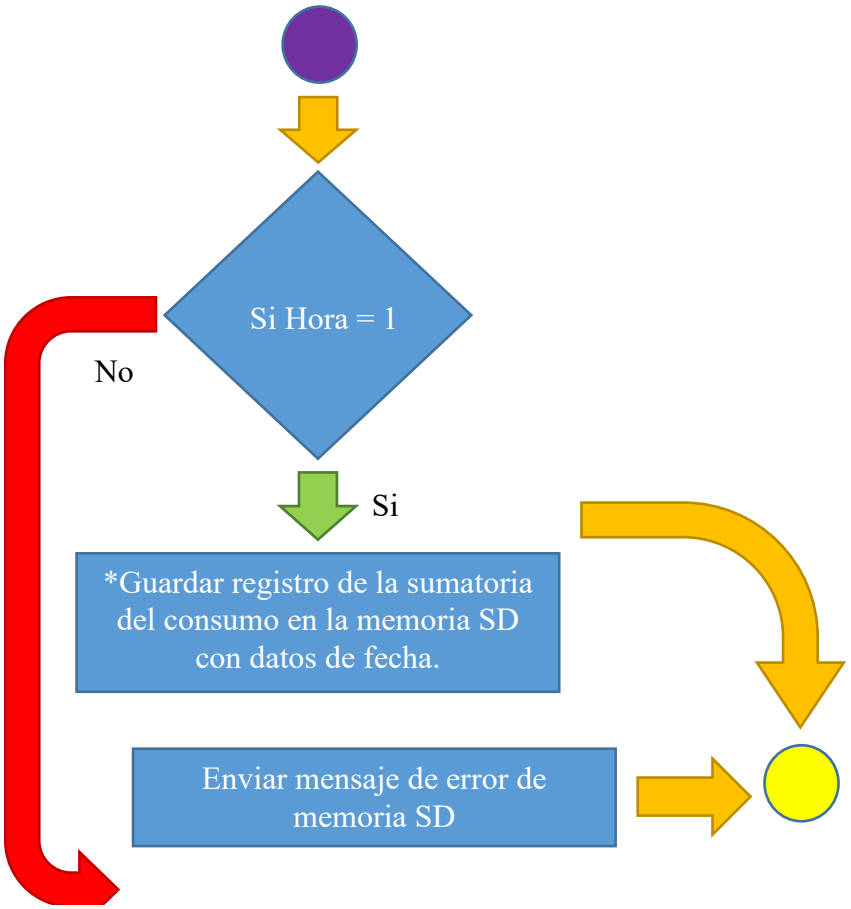


ILUSTRACIÓN 3.2D DIAGRAMA DE FLUJO DE ALGORITMO

3.1.2 Código en el entorno de Arduino.

3.1.2.1 Primer prototipo

El primer prototipo elaborado utiliza el sensor de corriente de tipo no invasivo, el SCT-013-30 A. Este primer prototipo no cuenta con todas las funciones optimizadas como lo describe el diagrama de flujo del algoritmo, pues este prototipo se abandonó después de la primera prueba dado a sus resultados, que serán mencionados y explicados en el capítulo 4 de este trabajo. Este prototipo sirvió como parte aguas para el segundo prototipo.

El primer segmento de código es el siguiente:

```
#include <Wire.h>
#include <RTClib.h>
#include <SD.h>
#include <LiquidCrystal.h>
RTC_DS1307 RTC;

double sample2=0.0;
double val=0.0;
double actualval=0.0;
double amps=0.0;
double ampers=0.0;
byte tiempo=0;
byte bytes=0;
int hora=0;


LiquidCrystal lcd(7, 6, 5, 8, 3, 2);
```

SEGMENTO DE CÓDIGO "CONFIGURACIÓN"

Corresponde al inicio de las configuraciones, se cargan las bibliotecas que se van a ocupar en el programa, se definen las configuraciones de los dispositivos usados, como: la pantalla LCD, se definen las terminales de conexión (pines) en los que se debe conectar la pantalla para su uso, también se describe la configuración inicial del reloj de tiempo real; y se nombran algunas variables globales que serán usadas en los procesos de cálculo y obtención de datos del programa más adelante:

- Sample2, val, actualval, amps, ampers. – Se utilizan como variables del programa en donde se almacena información sobre los procesos que va realizando el programa.

- Tiempo, bytes. – Se utiliza como segundos registros, donde se almacena la información proveniente del puerto serial.
- hora. – Es una variable del programa en donde se almacenará el valor de la hora (tiempo).
- LiquidCrystal. – Es la configuración de la pantalla LCD.

A continuación, se muestra la segunda parte del código:

```
void setup()
{

  Serial.begin(9600);
  Wire.begin();
  RTC.begin();
inicio:  lcd.begin(16, 2);

  if (!RTC.isrunning())
  {
    lcd.print("RTC no FUNCA!");
    delay(2000);
    goto inicio;
  }

  lcd.print("  RTC FUNCA  ");
  RTC.adjust(DateTime(__DATE__, __TIME__));
  delay(2000);
  lcd.clear();
  lcd.print("Ingrese duracion");
  delay(2000);
}
```

SEGMENTO DE CÓDIGO "INICIO"

En la que se puede apreciar el inicio de la comunicación mediante el puerto serial, se realizan los inicios de la pantalla LCD, del reloj de tiempo real.

La velocidad de comunicación por el puerto serial se establece en 9600 baudios, se genera una función en la que se evalúa la condición en la que se encuentra el reloj de tiempo real, si éste se encuentra funcionando muestra un mensaje en la pantalla LCD, donde dice al usuario que está funcionando correctamente, mientras si no se encuentra funcionando, muestra un mensaje en pantalla y reinicia, lo que genera que el programa no avance y se

quede en un ciclo, del cual no va a salir a menos que se atienda el problema del reloj de tiempo real.

Si el reloj de tiempo real se encuentra funcionando correctamente, el programa configura una nueva fecha y hora, que obtiene del entorno de programación. Espera un par de segundos, limpia la pantalla y muestra un mensaje al usuario, donde le indica que "ingrese la duración", se refiere al tiempo que tomará la medición de la energía eléctrica.

```
void loop()
{
  lcd.clear();
  lcd.println("    Cargando    ");
  delay(2000);

    tiempo=Serial.read();
    Serial.println(tiempo);

    lcd.clear();
    lcd.print("Ok");

    lcd.setCursor(0,1);
    lcd.println("Ingrese Hora");
    lcd.println("    Cargando    ");

    hora=Serial.read();
    lcd.clear();
    lcd.print("Ok");

    DateTime now=RTC.now();

    int horas= now.hour();

    if ( horas==hora)
    {
        int ts=tiempo*24*60*60;
      for (int i=0;i<=ts;i++)
        {
          double amp=sample();
          double totalamps=totalamps+amp;
          ampers=totalamps;
        }
    }
  lcd.print(" Aun no es hora ");
}
```

SEGMENTO DE CÓDIGO "FUNCIONES PRINCIPALES"

Se colocó en la zona de Setup este aviso, pues la configuración de la duración de la medición, solo se debía configurar una vez cada que se iniciaba el dispositivo, así, cada vez que se reiniciara, se podría volver a configurar la duración.

En el tercer segmento del programa (Funciones principales) se muestra el ciclo de trabajo del dispositivo, donde se incluyen las instrucciones que el dispositivo realizara una y otra vez, mientras la duración de la medición lo permita.

Se muestra un mensaje, donde se indica al usuario que ingrese el tiempo, el procedimiento es escribir la duración en la aplicación en el teléfono inteligente, para que después, la aplicación envía la información a la tarjeta, esta recibe la información y la almacena en una de las variables descritas al inicio del programa.

La pantalla es limpiada para volver a mostrar un mensaje, en el cual indica al usuario que ingrese la hora, la cual sirve como referencia después, para una función creada para el monitoreo y el cálculo de la información sobre el consumo eléctrico.

```
double sample()
{
    for(int i=0;i<200;i++)
 {
    sample2+=analogRead(A0);
    delay(2);
 }

  sample2=sample2/200;
  val =(5.0*sample2)/1024.0;
  actualval =val-2.5; // la ganacia de voltaje es 2.5v
  amps =actualval*10;

  return amps;
}
```

SEGMENTO DE CÓDIGO "FUNCIÓN CÁLCULO DE CORRIENTE"

La función creada, se pensó como una manera de monitorear el consumo durante el tiempo propuesto por el usuario, para que el programa únicamente se dedicara a tomar lecturas. Dentro del ciclo propuesto, se incluyó una función, que se generó fuera del Loop para reducir la cantidad de código colocada en el ciclo, además, si existía un error en las mediciones se revisaría la función creada, pues es en donde se encuentra la obtención de la

información por parte del sensor de corriente. La función toma la información obtenida del convertidor analógico digital, la almacena en un registro donde toda la información es acumulada, generando una sumatoria, con la intención de poder generar un promedio de la lectura, con el objetivo de estabilizar el valor obtenido por el convertidor, es por eso que el resultado del registro en el que se realizaba la sumatoria, es dividido por el mismo número de muestras tomadas, a el resultado obtenido del promedio, se le aplica una regla de tres, en donde se incluye el rango máximo que puede tener (en voltaje) y la resolución máxima del convertidor, con ello se logra obtener una cantidad en voltaje que es directamente proporcional a la cantidad de corriente medida por el sensor.

Después se resta el voltaje de ganancia, brindado a la salida del sensor, y el valor resultante es multiplicado para visualizar mejor la cantidad.

Los problemas hallados en el código, y las pruebas hechas al funcionamiento del programa y del sensor (se verán más a detalle en el capítulo 4 de este trabajo), sirvieron de base para el desarrollo del prototipo definitivo, el segundo prototipo.

El código mostrado arriba no se modificó, fue el primer intento generado para el primer prototipo, hubo demasiados errores, sin los cuales, no se hubieran evidenciado muchas de las deficiencias que el código original tenía, pero gracias a este experimento fue que se logró implementar una versión mucho mejor, que fue aplicada al desarrollo del segundo prototipo.

Así que disculpe las molestias ocasionadas por la lectura tan tediosa y confusa, pues el primer código generado así tenía que ser, para poder proyectar y mejorar exponencialmente para que el segundo prototipo fuera mucho mejor.

3.1.2.2 Segundo prototipo

En el segundo prototipo se utilizó el sensor de corriente de tipo invasivo, ACS 317.

El código generado para este prototipo está basado en el diagrama de flujo descrito anteriormente.

La primera parte del programa describe las tres primeras partes del diagrama de flujo:

```
void setup()
{
 Serial.begin(9600);
 Wire.begin();
 RTC.begin();
RTC.adjust(DateTime(__DATE__, __TIME__));

    if (!RTC.isrunning())
    {
      Serial.println("RTC NO FUNCIONA!");
      delay(2000);
      Serial.println("");
     RESTART;
    }

    else if(RTC.isrunning())
    {
      Serial.println("  RTC FUNCIONA  ");
      RTC.adjust(DateTime(__DATE__, __TIME__));
      delay(1000);
      Serial.println("");
     }
```

SEGMENTO DE CÓDIGO "CONFIGURACIÓN INICIAL"

Configuraciones iniciales. – El inicio del diagrama de flujo, se encuentra descrito en la función Setup(), es donde se definen las configuraciones iniciales del reloj de tiempo real, y la comunicación Serial. En la configuración de la comunicación serial, se inicia la comunicación y se establece una velocidad de transferencia de 9600 baudios.

Para el reloj se configura la fecha y la hora, y se realiza una evaluación mediante un ciclo de decisión, "si el reloj se encuentra funcionando", si no funciona, muestra un mensaje que es enviado por el puerto serial, y realiza un reinicio del programa, para que el dispositivo no opere hasta que todos sus elementos se encuentren funcionando correctamente; si se encuentra funcionando, envía un mensaje por el puerto serial y comienza a ejecutar la

siguiente parte del código. Corresponde a la siguiente sección del diagrama de flujo, la lectura y carga de los valores almacenados en la memoria EEPROM (Valores EEPROM) se almacenan seis variables en las localidades de la memoria.

Para poder almacenar información en la memoria EEPROM, lo primero que se tuvo que realizar, fue definir las localidades de memoria que cada variable ocuparía, la primera variable almacenada, es la única que ocupa un espacio de 4 localidades de la memoria, por ello se estableció que sus localidades serían de la dirección 0 hasta la 3. Las demás variables almacenadas, ocupan cada una dos localidades de memoria, así que para los cinco restantes se asignaron las direcciones 4,5; 6,7; 8,9; 10, 11; 12, 13; respectivamente.

```
      //Lectura de valores almacenados en EEPROM
for (int i = 0; i <= 3 ; i++)
{

   //Se almacena el valor en la EEPROM en los bits menos significativos.
   D = EEPROM.read(i);
   long(PotenciaFaux);
   //Se desplazan los bits tantos bytes como sea necesario, teniendo en
   //cuenta que el byte final no debe ser desplazado.
   if (i <= 3)
   {
     PotenciaFaux = PotenciaFaux << 8;
   }
  //Se almacena en la variable final el resultado de la operación OR de
  //ambos datos. Al realizar la operación OR con una variable que solo
  //contiene ceros (variable Dato), el resultado es el valor de la otra
  //variable. OR implica => 0+0=0, 0+1=1, 1+0=1, 1+1=1.
 PotenciaFaux = D || PotenciaFaux;
 //Se limpia la variable auxiliar para que en las siguientes iteraciones
 //no se cambie el resultado de los otros bytes.
 D = 0;
}
```

SEGMENTO DE CÓDIGO "VALORES EEPROM"

La primera variable, ***PotenciaFaux***, es en donde se registra la sumatoria del consumo en KWh que va generando el sistema. Para poder recuperar su información, se diseñó un ciclo (ilustración 3.1.2.2-2a) en el que la variable ***PotenciaFaux***, es compuesta por bytes. El proceso es el siguiente, se toma el byte más significativo de la variable, (almacenado en la

EEPROM, en la localidad más baja), se almacena temporalmente en una variable auxiliar

```
//lectura de valor tarifa 1
  ht1=EEPROM.read(4);
  lt1=EEPROM.read(5);
  tarifas1= ht1;
  tarifas1= tarifas1 <<8;
  tarifas1= tarifas1|lt1;

//lectura de valor tarifa 2
  ht2=EEPROM.read(6);
  lt2=EEPROM.read(7);
  tarifas2= ht2;
  tarifas2= tarifas2 <<8;
  tarifas2= tarifas2|lt2;

//lectura de valor tarifa 3
  ht3=EEPROM.read(8);
  lt3=EEPROM.read(9);
  tarifas3= ht3;
  tarifas3= tarifas3 <<8;
  tarifas3= tarifas3|lt1;

//lectura de valor alarmas
  halar=EEPROM.read(10);
  lalar=EEPROM.read(11);
  alarmas= halar;
  alarmas= alarmas <<8;
  alarmas= alarmas|lalar;

//lectura de valor fecha de corte
  hrei=EEPROM.read(12);
  lrei=EEPROM.read(13);
  reinicios= hrei;
  reinicios= reinicios <<8;
  reinicios= reinicios|lrei;
```

SEGMENTO DE CÓDIGO "VALORES EEPROM"

de tipo *byte*, la cual tomara la información de la EEPROM, para poder almacenar la información en la variable. Una vez que se almaceno la información, la variable ***PotenciaFaux***, es sometida a un corrimiento de 8 bits (un byte), en el cual se desplazan sus 8 bits más significativos a la derecha (recorriéndolos), permitiendo el paso a los próximos ocho bits; el ciclo se vuelve a repetir, hasta que se leen las cuatro localidades de memoria y se recupera la información de la variable, almacenada en la memoria.

Para las siguientes variables se usó una alternativa distinta para cargar los datos almacenados en la memoria, pues las variables utilizadas son de tipo *int*, el cual está compuesto por 16 bits, 2 bytes, por lo que se realizó el almacenamiento de la variable principal en dos variables auxiliares, en una variable auxiliar se almacena temporalmente el byte más significativo, y en la otra el byte menos significativo, ambos obtenidos de las direcciones de la memoria, correspondientes a cada variable; el siguiente paso es igualar la variable que contiene los bits más significativos con la variable principal, con el fin de almacenar los bits más significativos de la variable en los lugares menos significativos, para poder aplicar un corrimiento de 8 lugares hacia la izquierda, para que los bits que

estaban en el las posiciones menos significativas, pasen a estar a las posiciones más significativas. Ahora solo falta cargar la variable auxiliar que contiene los bits menos significativos de la variable principal, para poder insertar los bits en la variable principal, se efectúa una operación *or* entre la variable principal y la variable auxiliar, permitiendo que los bits menos significativos ocupen los lugares menos significativos de la variable principal. Este procedimiento se realiza para las 5 variables restantes, de las cuales, tres almacenan información con respecto a las tarifas establecidas por CFE, son las tres primeras variables ***tarifas1, tarifas2 y tarifas3***. La variable ***alarmas***, contiene la información sobre el monto, en el que el sistema tiene que avisar que se ha alcanzado la cantidad, para que el usuario este consiente de que alcanzo la cantidad establecida. Por último, la variable ***reinicios***, contiene los días que restan para que el periodo de medición se termine.

La siguiente sección de código obtiene la información del sensor, obtener los Watts, convertirlos a KWh y realizar la sumatoria del consumo.

```
void loop()
{

 Voltage = getVPP();
 VRMS = (Voltage/2.0) *0.707;
 AmpsRMS = (VRMS * 1000)/mVperAmp;

 Potencia= volts*AmpsRMS;

    // Se realiza la sumatoria del consumo

    PotenciaZ=PotenciaZ+Potencia;
    PotenciaZ=PotenciaZ-13.06;

    PotenciaF=PotenciaZ*0.0003; //Wh
    PotenciaF=PotenciaF/1000.0; //KWh
    PotenciaFaux=PotenciaFaux+PotenciaF; //registro sumatoria
```

SEGMENTO DE CÓDIGO "OBTENCIÓN KWH"

Se obtiene la información del sensor de corriente mediante una función (***getVPP()***) que regresa un valor en voltaje, el cual, es usado para convertirlo a un valor de corriente mediante la constante del sensor, con el valor de la corriente podemos calcular el consumo eléctrico que se mide en Watts, para ello solo se multiplico la intensidad de corriente (la

que el sensor se encuentra midiendo) con el voltaje que es suministrado, el cual es constante y se encuentra almacenado en una variable llamada *volts*. Una vez que se obtuvo el consumo, se define una variable como un registro (***PotenciaZ)*** donde se almacenan los resultados de la primera operación, para después poder convertir los valores almacenados a KWh, para ello se realizan dos operaciones, en una se convierten los segundos a horas, y en la otra se convierten los Wh a KWh. Con las unidades correctas, se crea un último registro (***PotenciaFaux***) donde se almacena la sumatoria final con las unidades KWh, es el registro que se almacena en la EEPROM.

```
float getVPP()
{
  float result;

  int readValue;             //valor obtenido del sensor
  int maxValue = 0;
  int minValue = 1024;

   uint32_t start_time = millis();
   while((millis()-start_time) < 50)
   {
       readValue = analogRead(sensorIn);
       // see if you have a new maxValue
       if (readValue > maxValue)
       {

           maxValue = readValue;
       }
       if (readValue < minValue)
       {

           minValue = readValue;
       }
   }

  result = ((maxValue - minValue) * 5.0)/1024.0;

  return result;
}
```

SEGMENTO DE CÓDIGO "OBTENCIÓN DE INFORMACIÓN DEL SENSOR"

Para obtener la información del sensor de corriente, se diseñó una función que se encarga de obtener la información del sensor, en la cual, se inicia una lectura del convertidor analógico digital, durante un intervalo de tiempo establecido, realiza una serie de muestras

durante 50 milisegundos y estabiliza la muestra de manera que la lectura del sensor se aproxime a un valor promedio, el valor obtenido es retornado al programa. Donde se procesa el valor, se procesa y se almacena.

La siguiente captura de código:

```
if(h==6)
{
   if(m==30)
   {
     if(s<10)
     {
       for (int i = 0; i <= 3; i++)
      {
        long(PotenciaFaux);
        //Se guarda el byte más significativo
        B = highByte(PotenciaFaux);
        EEPROM.write(i, B);
        //Se desplazan 8 bits hacia la izquierda de modo que los 8 primeros
        //bits se pierden y los 8 siguientes pasan a ser los primeros
        PotenciaFaux = PotenciaFaux << 8;
      }
      //fecha de corte
        hrei = highByte(reinicios);
        lrei = lowByte(reinicios);
        EEPROM.write(12, hrei);
        EEPROM.write(13, lrei);
     }
   }
}
```

SEGMENTO DE CÓDIGO "GUARDADO DE INFORMACIÓN"

Corresponde a la medición de la hora del día, para poder almacenar la información guardada en ***PotenciaFaux*** y en ***reinicios*** en la memoria EEPROM. En ***PotenciaFaux*** se encuentra almacenada la sumatoria del consumo, para realizar el guardado en la EEPROM se realizó una secuencia para almacenar su información. Es la siguiente: para poder almacenar su información se diseñó un ciclo en el que la variable ***PotenciaFaux***, es descompuesta por bytes. El proceso es el siguiente, se toma el byte más significativo de la variable, se almacena temporalmente en una variable auxiliar de tipo *byte*, la cual brindará la información a la EEPROM, para poder almacenar información en la primera dirección definida para esta variable, la dirección cero. Una vez que se almaceno la información, la

variable ***PotenciaFaux***, es sometida a un corrimiento de 8 bits (un byte), en el cual se desplazan sus 8 bits más significativos a la derecha (perdiéndolos), permitiendo el paso a los próximos ocho bits, el ciclo se vuelve a repetir, hasta que son ocupadas las 4 localidades de memoria. Después de que se almaceno la información de la variable ***PotenciaFaux***, se procede a guardar la variable ***reinicios***, la cual contiene la información acerca de los días restantes para el término del periodo, para esta variable se creó un ciclo para el almacenamiento en la EEPROM, se realizó lo siguiente, como la variable en la que se almacena la información es de tipo *int*, ocupa un espacio de 16 bits, (2 bytes), por lo que la separación de los bits se facilita, pues se separa la variable en 8 bits más significativos, y 8 bits menos significativos, se almacena cada grupo de información en una variable auxiliar de tipo *byte*, la cual sirve de intermediaria en el proceso de almacenamiento de la EEPROM. El proceso de guardado de información en la memoria se realiza cada 6 horas a lo largo del día.

El próximo procedimiento a realizar según el diagrama de flujo, es revisar si existen datos en el Puerto serial y dependiendo del resultado será la línea de acción del programa, si no existen datos en el puerto, el programa ejecuta su próxima instrucción; si existen datos, se realiza el siguiente protocolo (ilustración 3.1.2.2-6): Se revisa el puerto serial en busca de información, si hay información, recibe la información en una variable hasta que encuentra un carácter que no es alfanumérico, (no pertenece ni a números ni a letras), en el momento en que lo encuentra, detiene el guardado y ejecuta la siguiente instrucción, que es la misma que la instrucción anterior; se repite el proceso 5 veces, pues se espera la entrada de 5 variables, donde tres de ellas son las tarifas, una es el valor de la alarma y la ultima los días que durara el periodo de medición.

Una vez que ha recibido y guardado la información, se procede a guardar los datos recibidos en las celdas de la memoria EEPROM, donde cada variable tiene asignada dos direcciones en la memoria: para la variable tarifas1 se asignaron las direcciones 4 y 5; para la variable tarifas2 se le asignaron las direcciones 6 y 7; para la variable tarifas3 se le asignaron las direcciones 8 y 9; para la variable alarmas se le asignaron las direcciones 10 y

11; para la variable reinicios se le asignaron las direcciones 12 y 13.

```
while(Serial.available())
{
  tarifa1=Serial.parseInt();// recepcion de datos x parte de la app
  tarifa2=Serial.parseInt();
  tarifa3=Serial.parseInt();
  alarma=Serial.parseInt();
  reinicio=Serial.parseInt();

     if(Serial.read()=='\n')//finaliza recepcion de datos
  {
    tarifas1=tarifa1;  //carga de datos recibidos a variables fija
    tarifas2=tarifa1;
    tarifas3=tarifa1;
    alarmas=alarma;
    reinicios=reinicio;
  }

 //Guardar informacion en EEPROM
//tarifa1                                          //tarifa2

        //Se guarda el byte más significativo              ht2 = highByte(tarifas2);
        ht1 = highByte(tarifas1);                          lt2 = lowByte(tarifas2);
        lt1 = lowByte(tarifas1);                           EEPROM.write(6, ht2);
        EEPROM.write(4, ht1);                              EEPROM.write(7, lt2);
        EEPROM.write(5, lt1);

//tarifa3                                          //alarmas

        ht3 = highByte(tarifas3);                          halar = highByte(alarmas);
        lt3 = lowByte(tarifas3);                           lalar = lowByte(alarmas);
        EEPROM.write(8, ht3);                              EEPROM.write(10, halar);
        EEPROM.write(9, lt3);                              EEPROM.write(11, lalar);

                        //fecha de corte
                                hrei = highByte(reinicios);
                                lrei = lowByte(reinicios);
                                EEPROM.write(12, hrei);
                                EEPROM.write(13, lrei);
```

SEGMENTO DE CÓDIGO "PUERTO SERIAL"

Como las variables que se usaron para almacenar la información ocupan 16 bits y las localidades de la memoria EEPROM son de 8 bits, es necesario dividir la información en dos grupos de 8 bits, uno es el grupo de los bits más significativos, y el otro es el grupo de los bits menos significativos, esta operación se realiza mediante dos instrucciones con las que cuenta el entorno, los bits más significativos se almacenaron en una variable auxiliar de tipo *byte*, donde será la variable que se almacenara en la EEPROM.

La próxima captura de código describe el análisis de la fecha de término del periodo de medición, donde se evalúa la variable ***reinicios***. Se generó una secuencia donde se evalúa la hora, y donde cada 24 horas se reduce el valor de la variable en una unidad, cuando la variable llega a cero, se limpian las direcciones de la EEPROM que contienen la sumatoria del consumo. Y después reinicia el sistema.

```
if(h==12)
   {
      if(m==30)
      {
        if(s<10)
        {
          inicio=inicio+1;
        }
      }
   }

if(inicio==2)
   {
     reinicios=reinicios-1;
     inicio=0;
   }

if(reinicios==0)
   {
     for (int i = 0; i <= 3; i++)
        {
          EEPROM.write(i, 0);
        }
     RESTART;
   }
```

SEGMENTO DE CÓDIGO "FECHA DE CORTE"

La próxima captura de código muestra las operaciones realizadas para la aproximación del costo del consumo, donde se utilizan los valores proporcionados por el usurario (las tarifas), que después son evaluados en ciclos de decisión para conocer que tarifa se debe

utilizar en función de la cantidad del consumo.

```
//Para la alarma, de tipo monetaria
  tarifas1=tarifas1*0.001;
  tarifas2=tarifas2*0.001;
  tarifas3=tarifas3*0.01;

if(PotenciaF<=150.0)
  {
     dinero= PotenciaF*tarifas1;
  }
if(150.0<PotenciaF<=280.0)
  {
     dinero= PotenciaF*tarifas2+(tarifa1*150);
  }
if(PotenciaF>280.0)
  {
     dinero= PotenciaF*tarifas3+(tarifa1*150)+(tarifa2*130);
  }
if(dinero>(alarma-50))
  {
    digitalWrite(5,HIGH);
    Serial.print(100);
    Serial.println("   ");
  }
```

SEGMENTO DE CÓDIGO "TARIFAS $$$"

Las tarifas son multiplicadas respectivamente por un operando específico, pues las variables se encuentran sin punto decimal, por lo que para su utilización se deben de adecuar, para poder realizar una correcta estimación del costo de la energía consumida.

También se muestra el ciclo diseñado para el funcionamiento de la alarma, donde se otorgan 50 unidades de ventaja, en las que la alarma es activada antes de que llegue a la cantidad establecida por el usuario. Cuando la condición se cumple, se activa una salida, de la tarjeta, donde se encuentra conectado un indicador visual, el cual cuando se activa la salida, el indicador se enciende y emite una luz, para que el usuario se entere mediante el dispositivo y la aplicación que la alarma ha sido alcanzada.

El siguiente bloque de instrucciones del código, pertenece al proceso, del diagrama de flujo, donde se establece el envió de la información obtenida y calculada a través del puerto serial de la tarjeta, al cual, se encuentra conectado el módulo de comunicación inalámbrica por bluetooth, con el fin de que el dispositivo envíe la información al teléfono inteligente, para

que la aplicación logre recibir la información, para después mostrarla en la pantalla del teléfono inteligente.

```
//Envio de informacion por puerto serial
    Serial.print(PotenciaF+1000);
    Serial.println("   ");
    Serial.print(dinero+2000);
    Serial.println("   ");
    Serial.println(reinicios+3000);
    delay(1000);
```

SEGMENTO DE CÓDIGO "ENVÍO DE INFORMACIÓN"

Se establecieron cantidades de referencia para que, cuando la información sea enviada, facilite la discriminación de variables por parte de la aplicación, así que, a cada variable se le asignó una cantidad de referencia específica que no interviene con la cantidad obtenida y calculada por el programa.

La siguiente y última etapa del código corresponde al almacenamiento del consumo eléctrico en una memoria flash, que se encuentra conectada de manera externa a la tarjeta; el consumo ira acompañado de la fecha y hora en la que fue guardado, con el fin de proveer al usuario un registro histórico de los consumos realizados en el tiempo que el dispositivo se encuentre conectado. Los guardados de información se establecieron cada hora, por lo que, por día, existirán 24 históricos almacenados.

```
for(int a=1; a<=12; a++)
{
  if(a==h)
  {
    if(m==30)
    {
      if(s<10)
      {
        medicion = SD.open("Consumo.txt", FILE_WRITE);
       if (medicion)
        {
         medicion.print("Fecha:");
         medicion.print(' ');
         medicion.print(y, DEC);
         medicion.print('/');
         medicion.print(mes, DEC);
         medicion.print('/');
         medicion.print(dia, DEC);
         medicion.print(' ');
         medicion.print(h, DEC);
         medicion.print(':');
         medicion.print(m, DEC);
         medicion.print(':');
         medicion.print(s,DEC);
         medicion.println();
         medicion.print("Consumo:");
         medicion.print(' ');
         medicion.print(PotenciaFaux);
         medicion.println();
         medicion.close();
        }
        else
        {
        Serial.println("Error al guardar SD");
        }
      }
    '
```

SEGMENTO DE CÓDIGO "ALMACENAMIENTO DE CONSUMO EN SD"

3.2 Diagramas esquemáticos

Los diagramas esquemáticos son representaciones gráficas de los circuitos electrónicos y para este caso, representaran las conexiones establecidas con la tarjeta y sus periféricos.

3.2.1 Diagrama conexiones primer prototipo

A continuación, se muestra el esquema de conexiones que se realizaron para el desarrollo del prototipo.

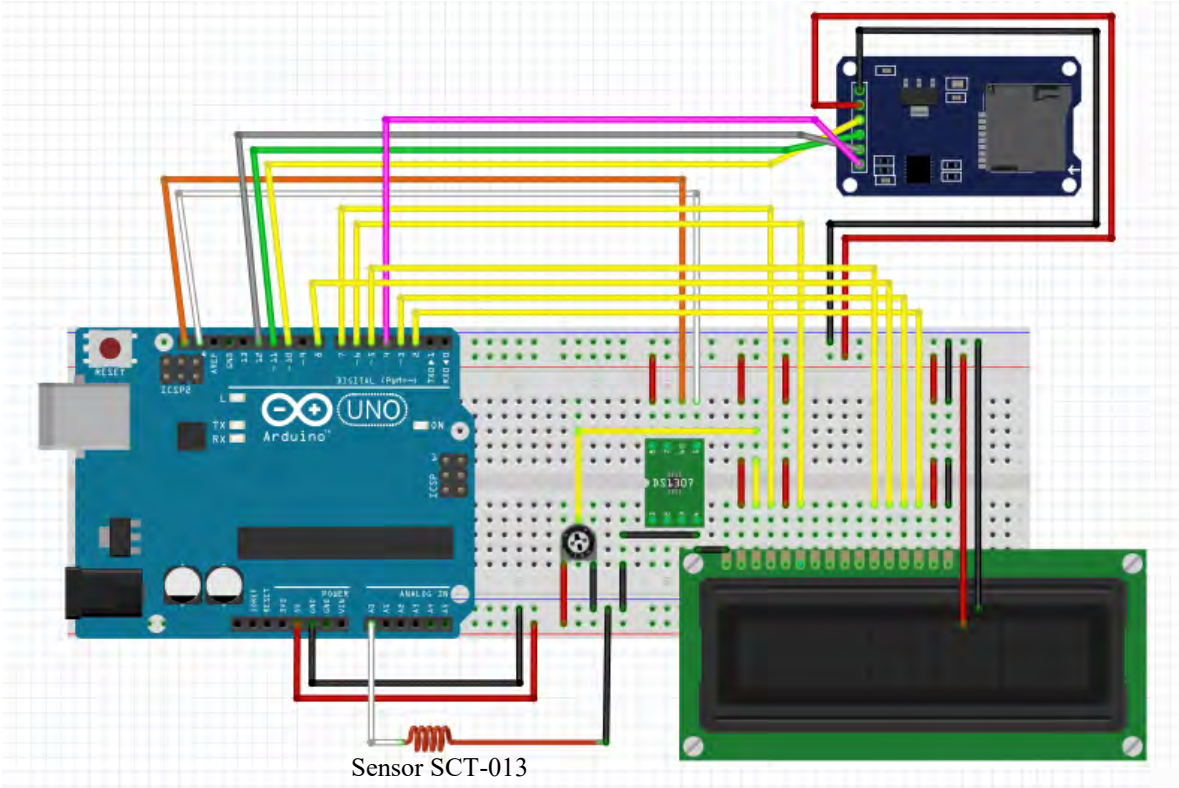


ILUSTRACIÓN 3.3 DIAGRAMA DE CONEXIONES PROTOTIPO I

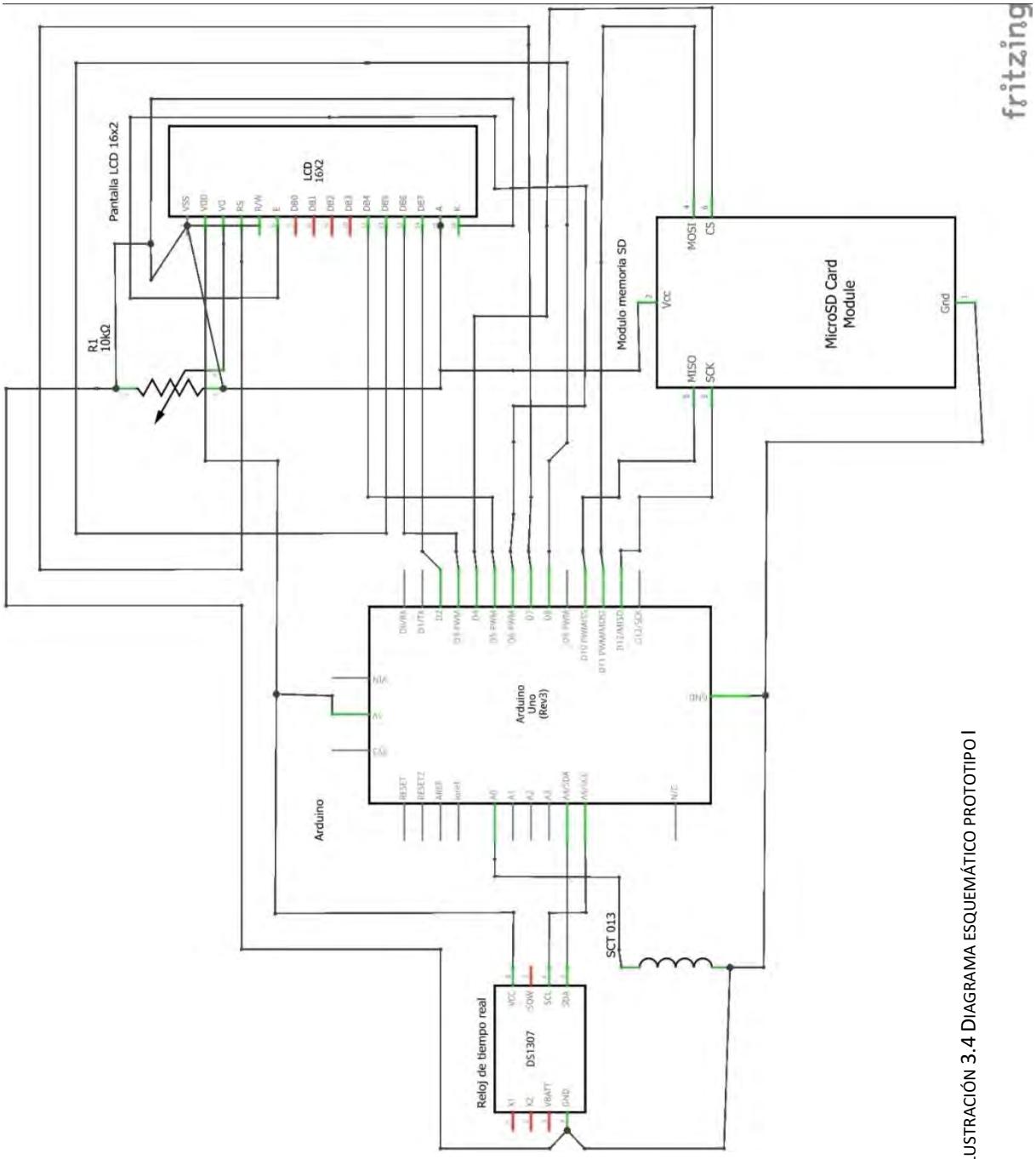


ILUSTRACIÓN 3.4 DIAGRAMA ESQUEMÁTICO PROTOTIPO I

3.2.2 Diagrama de conexiones segundo prototipo

A continuación, se muestra el diagrama de conexiones realizadas para las pruebas al dispositivo.

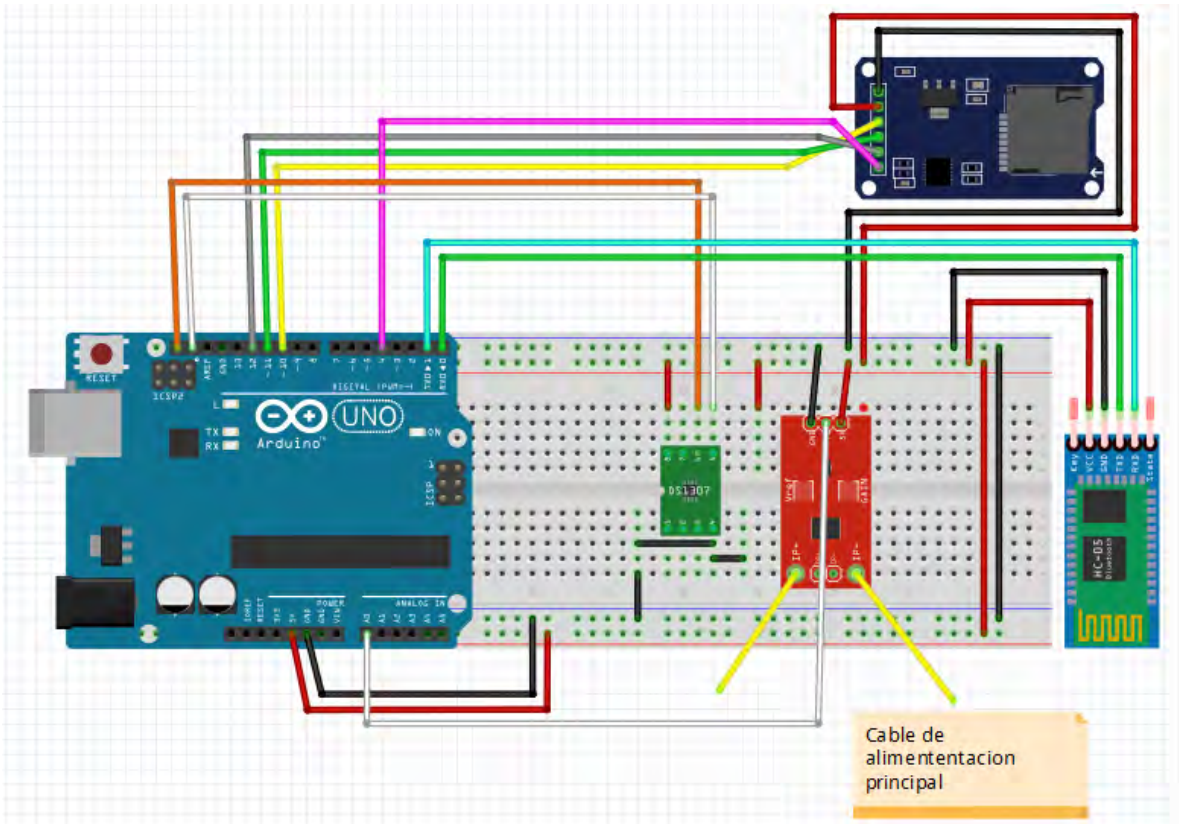


ILUSTRACIÓN 3.5 DIAGRAMA DE CONEXIONES PROTOTIPO II

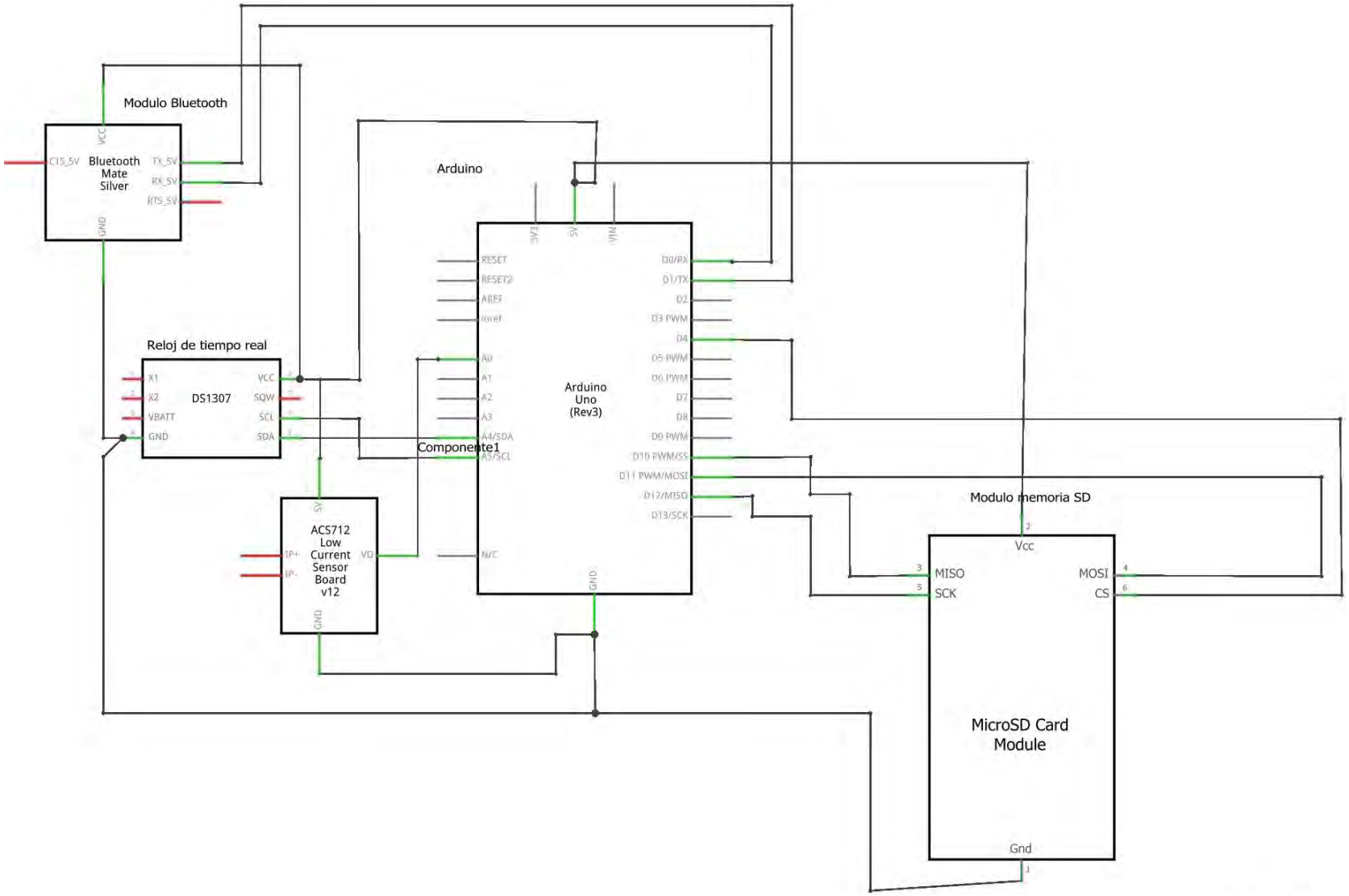


ILUSTRACIÓN 3.6 DIAGRAMA ESQUEMÁTICO PROTOTIPO II

3.3 Diseño y construcción de la interfaz en Android

El entorno que se escogió para la creación de la interfaz para el teléfono inteligente, es App Inventor 2.

ILUSTRACIÓN 3.7
"ICONO DE LA INTERFAZ"

Para empezar a trabajar en App inventor, es requisito indispensable abrir o tener una cuenta de correo en GMail; para poder ingresar al sitio de creación (plataforma en internet), solo hay que ingresar a la siguiente dirección electrónica: http://ai2.appinventor.mit.edu, proporcionar un correo electrónico de google, una nueva contraseña y listo.

El diseño de la aplicación se basa en muros de trabajo, es decir, los elementos visuales están contenidos en dos arreglos llamados ***Table arrangement*** (arreglo de tabla), los cuales se utilizan como pantallas; en un arreglo de tabla se introdujo el menú principal, es la pantalla que el usuario observará más frecuentemente, pues cuenta con las casillas en donde se mostrará la información que el sistema de control brinde. Y en el otro arreglo se introdujeron todos los elementos para que usuario pueda capturar los datos referentes a la configuración, como el precio de las tarifas, la alarma monetaria, y los días para el término del periodo de medición. A continuación, se describirá el desarrollo de cada pantalla.

3.3.1 Pantalla Principal

ILUSTRACIÓN 3.8 INTERFAZ "PANTALLA PRINCIPAL"

La pantalla principal (ilustración 3.8), para poder colocar un arreglo de tabla, se va a la paleta de accesorios y se escoge la categoría de ***layout*** *(arreglo)*, de las 5 opciones que aparecen se escoge la tercera, la que dice ***Table arrangement***, se selecciona, se arrastra y se suelta hasta la zona de trabajo, donde al colocarla se marca un margen en la pantalla; en la sección de configuración (del entorno), en la columna de propiedades, aparecerán tres casillas, configuraremos las propiedades de la tabla para tener una columna, y tres filas, la casilla de visible debe estar activada.

Se añadirán a la pantalla: un elemento de comunicación bluetooth, en la sección de la paleta de elementos en la sección de conectividad, se escogerá un elemento llamado ***bluetooth client***, para la realización de la comunicación por bluetooth; un ***timer***, que se encuentra en la sección de sensores, es para poder establecer rangos de lectura de la comunicación bluetooth, se establece en al área de propiedades del reloj, un intervalo de 10 microsegundos, y se activa la opción de siempre disparando; y por ultimo un ***notificador***, el cual se encuentra en la sección de usuario.

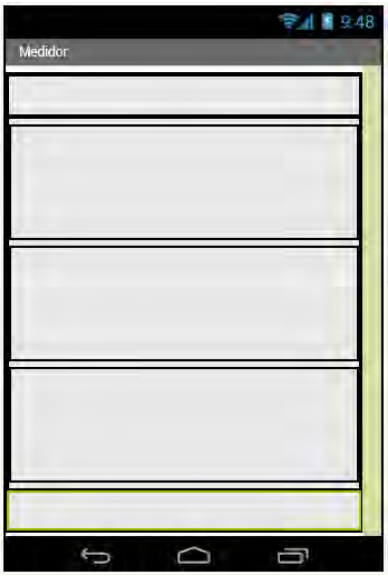


ILUSTRACIÓN 3.9 INTERFAZ
"SOPORTE PARA ELEMENTOS"

Una vez que tenemos la tabla configurada, ahora solo queda acomodar los espacios donde se visualizará la información, para ello, colocaremos en la primera fila de la tabla, un ***arreglo horizontal***, se encuentra en la categoría de arreglos en la paleta, lo usaremos para colocar dos botones y una imagen, de manera que los tres elementos queden distribuidos horizontalmente en la fila.

En la fila de en medio, colocaremos un ***arreglo vertical***, y dentro del arreglo, introduciremos tres arreglos horizontales, es donde se visualizará la información, cada arreglo horizontal tendrá un alto de 100 pixeles y un ancho de 310 pixeles, esto se modifica en la tabla de propiedades en la sección de configuración, para poder editar un elemento, es necesario seleccionarlo antes de intentar editarlo.

Y en la última fila, introduciremos un arreglo horizontal pues se colocarán un botón y un cuadro de contraseña. Ya que tenemos el soporte (ilustración 3.9) de los elementos ahora se colocarán los elementos en donde corresponden:

En la primera fila, se tomará un botón de la paleta de elementos se arrastrará y se soltará hasta que se encuentre dentro del arreglo horizontal, después realizaremos el mismo procedimiento con un ***List picker***, y una ***imagen***, el ***List picker*** debe quedar al centro, en sus propiedades, modificaremos el color del fondo, (donde dice *background color*, en la

parte de propiedades), se escogió amarillo, un tamaño de letra de 20, y en donde dice Texto, colocaremos “Conectar”. Para la ***imagen***, se utilizó una imagen de un símbolo de aviso, o advertencia, las dimensiones de la imagen son de 30 pixeles de alto por 30 pixeles de ancho, se activa la opción ajustar imagen. Para el botón, se escribió en el campo de texto la palabra “ok”, las dimensiones se dejaron automáticas, el fondo azul y el color de la letra en verde, el campo de visualización se desactivo. La configuración de las propiedades para esta fila será automática en las medidas y la orientación será al centro.

En la Fila del centro, donde se colocaron los tres arreglos horizontales, introduciremos en cada arreglo horizontal: Un ***arreglo vertical***, en el arreglo vertical se introducirán dos Label, a cada una se le asignará un valor de 30 pixeles de alto por 165 pixeles de ancho, sobre el mismo arreglo horizontal, fuera del arreglo vertical colocaremos un ***Label***, el cual tendrá 50 pixeles de alto por 100 pixeles de ancho, esta configuración se aplicará a cada arreglo horizontal contenido en la segunda fila.

En los ***Label*** superiores (los que quedaron encima) que se encuentran en los arreglos verticales, dentro de cada arreglo horizontal, escribiremos en ellos, pues serán las caratulas de la información que recibe la aplicación. En el primer ***Label*** del primer arreglo horizontal, escribiremos en el campo de texto de la barra de propiedades, “Has consumido”, con un tamaño de letra 18 y un fondo amarillo; para el ***Label*** que se encuentra fuera del arreglo vertical, escribiremos en el campo de texto “KWh”, con un tamaño de letra 28 (es para mostrar las unidades de las cantidades que se visualizaran. Para el segundo ***Label*** superior, del segundo arreglo horizontal, escribiremos en el campo de texto “Vas a pagar” con un tamaño de letra 18, y un fondo de color verde; para el ***Label*** que se encuentra fuera del arreglo vertical, se escribió en el campo de texto, tres signos de pesos ($$$, para representar que la cantidad mostrada es en pesos), con un tamaño de letra 28. Para el último Label en el último arreglo horizontal de la fila, se escribió en el campo de texto “Aún quedan” con un fondo de color cian, tamaño de letra 18; para el label externo al arreglo vertical, se escribió en su campo de texto “Días” con un tamaño de letra 28.

En la última fila, se colocó un ***botón*** y un ***cuadro de contraseña***. Para el botón se establecieron dimensiones automáticas, se escribió en su campo de texto la palabra “configuración”, con un tamaño de letra 20 con un color de letra blanco y un fondo en color

rojo; para el cuadro de contraseña, se dejaron sus propiedades en automático, a excepción del campo de visible, pues se desactivo. La pantalla principal quedo configurada, con cada elemento previamente establecido en su lugar (ilustración 3.8), por lo que ahora se describirá la pantalla de configuración.

3.3.2 Pantalla de configuraciones

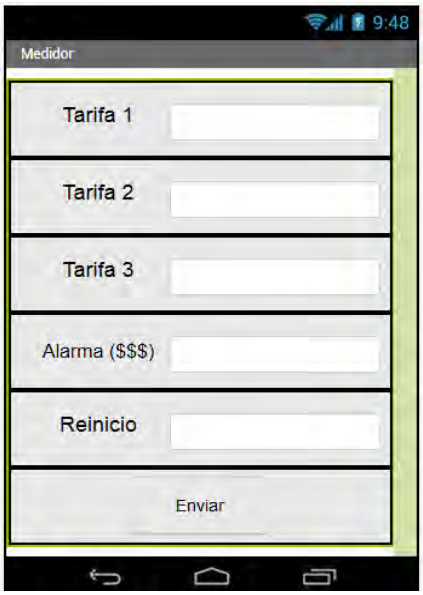


ILUSTRACIÓN 3.10 INTERFAZ "CONFIGURACIÓN"

En esta pantalla se diseñó, para poder ingresar la información sobre los parámetros que se deben de enviar al sistema de control para que pueda realizar los cálculos de la manera correcta y mostrarlos en la aplicación. En este arreglo de tabla se definieron 6 filas, una columna; en cada una de las primeras cinco filas (contadas de arriba hacia abajo), se agregó un arreglo horizontal, con dimensiones de 15% de altura por un 98% de ancho; y dentro de cada arreglo se colocó un ***Label***, con una altura de 30 pixeles por 100 pixeles de ancho; y un cuadro de texto, que permite la escritura de texto, con una altura automática y un ancho de 160 pixeles y la casilla de únicamente números activada. En el último arreglo horizontal, de la última fila, se colocó un botón con una altura de 50 pixeles con un ancho de 110 pixeles, en la casilla de texto se rescribió "Enviar" con un tamaño de letra 16, centrada.

En el primer ***label*** de la primera fila se escribió en la casilla de texto "Tarifa 1" con un tamaño de letra 21, centrada. Se realizó lo mismo para las dos siguientes ***label*** en las

siguientes filas, con el número del final del texto con un 2 y 3 respectivamente. Para la cuarta ***label*** se escribió en la casilla de texto "Alarma $$$" con un tamaño de letra 18; y para el último ***label*** de la última fila, se escribió en su casilla de texto "reinicio".

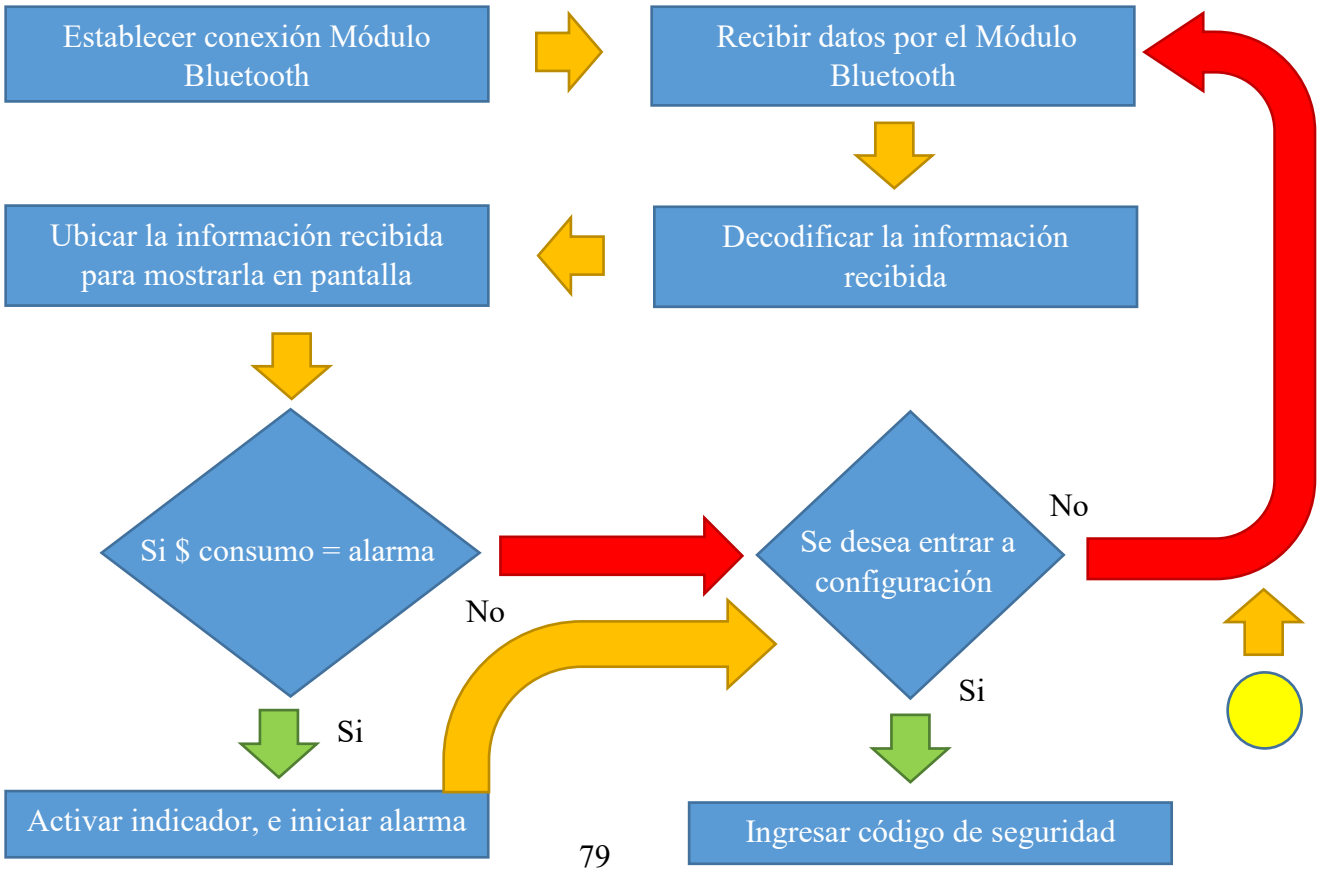
3.3.3 Programación de la Interfaz

La programación en App Inventor se realiza mediante la conexión de bloques, que realizan funciones específicas, por lo que no es necesaria la escritura del código mediante descripciones de código utilizando instrucciones, la programación por bloques, se escoge la función que se desea realizar, se conecta con otra función para realizar las acciones deseadas.

Para conocer el funcionamiento de la aplicación se describirá mediante un diagrama de flujo las funciones que debe realizar.

3.3.3.1 Diagrama de Flujo

La aplicación contiene varias características para su funcionamiento, debe establecer una conexión con un módulo Bluetooth, recibir y enviar información, decodificar la información, mostrarla en una pantalla para que el usuario la visualice.



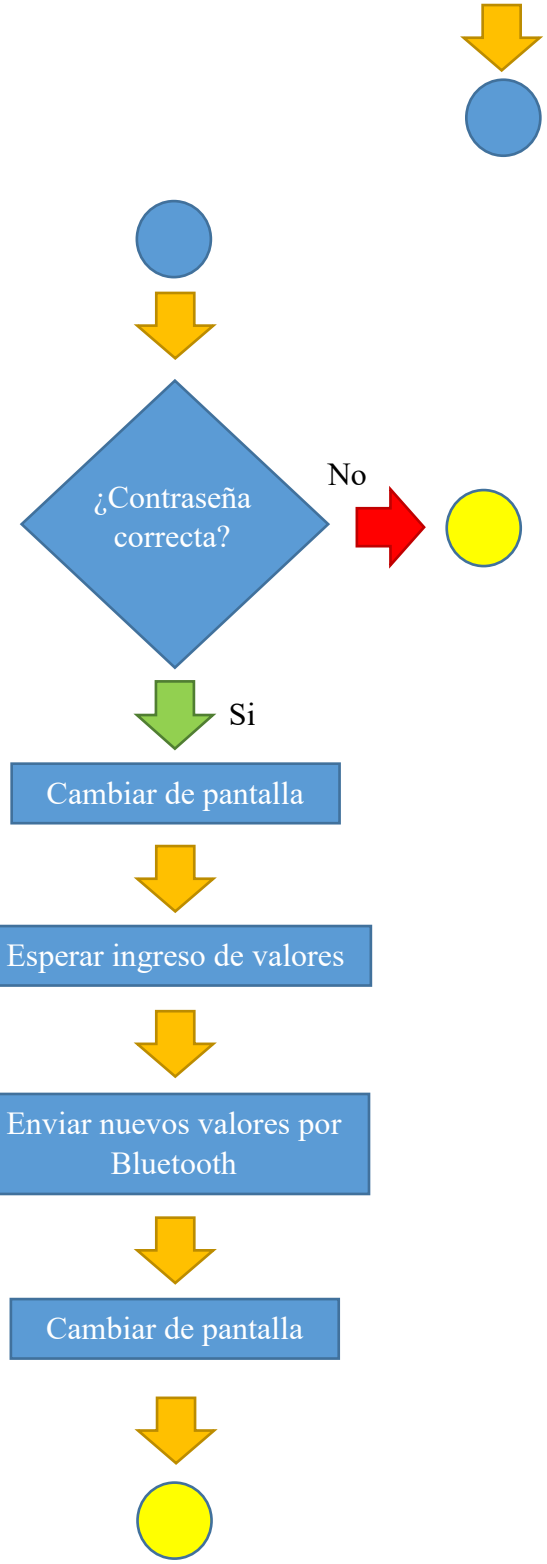


ILUSTRACIÓN 3.11 DIAGRAMA DE FLUJO DE INTERFAZ

3.3.3.2 Construcción del código

ILUSTRACIÓN 3.12A "OBTENCIÓN DE INFORMACIÓN"

El primer conjunto de bloques que se construyó, fue para poder llevar a cabo las primeras instrucciones del diagrama de flujo, establecer una conexión con el módulo Bluetooth (ilustración 3.12 A). También se describe la petición de información acerca de los dispositivos enlazados al teléfono inteligente, para que una vez que se haya oprimido el *list picker* "***conectar***" de la pantalla principal, se puedan visualizar los dispositivos previamente emparejados con el teléfono inteligente, así se escoge el módulo bluetooth con el que se establecerá una conexión.

El próximo grupo de bloques (ilustración 3.12b), muestra un efecto que se produce en el list picker ***conectar***, cuando la conexión con el dispositivo bluetooth se ha hecho con éxito, el

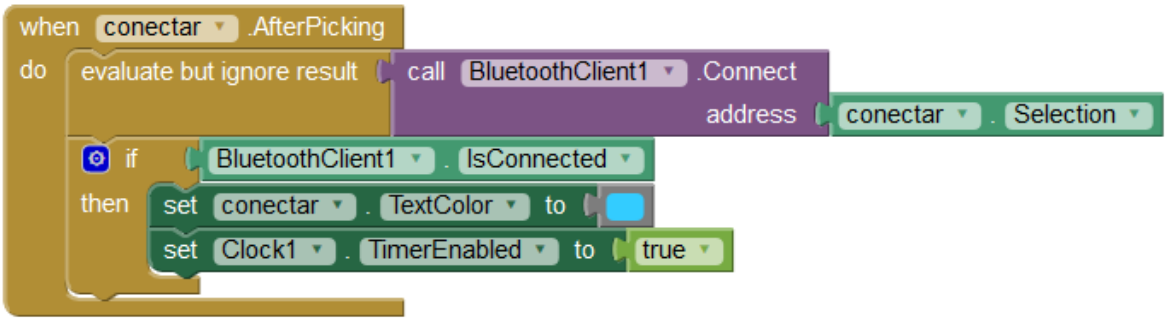


ILUSTRACIÓN 3.12B "AVISO DE CONEXIÓN"

efecto consiste en que las letras del *list picker* cambian de color, pasan de negro a azul. También se activa un ***timer***, para comenzar a analizar la información que recibe la aplicación por el bluetooth. La siguiente captura de bloques muestra la captura de los datos enviados por el bluetooth.

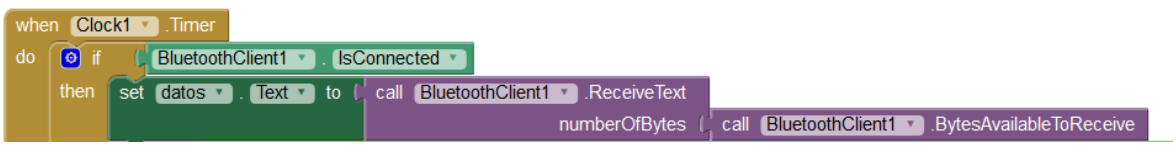


ILUSTRACIÓN 3.13 "RECEPCIÓN DE DATOS"

Los datos obtenidos mediante el bluetooth son almacenados en un registro llamado Datos, los valores son guardados tal cual están, para poder decodificar los datos y conocer el origen de cada valor, es necesario realizar un proceso de discriminación de información, que es mostrado en la siguiente ilustración 3.14.

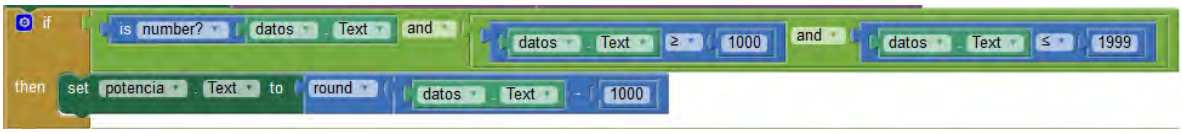


ILUSTRACIÓN 3.14 "DECODIFICACIÓN Y DISCRIMINACIÓN DE DATOS"

El código se refiere a un proceso en donde los valores contenidos en el registro datos, son evaluados, si pertenecen a un intervalo de cantidades quiere decir, que el valor encontrado corresponde a determinada variable, en la ilustración de arriba, se puede ver que el valor que se encuentra en el rango de 1000 a 1999, corresponde a la variable potencia, por lo que se mueve la información hacia el registro de potencia (que es un label en la pantalla principal), y se remueve la codificación establecida por el sistema de control. El mismo proceso se realiza para las otras dos variables restantes.

Una vez que la información ha sido reubicada, es mostrada en la pantalla principal en el formato correcto.

La próxima ilustración, muestra el ciclo realizado para la evaluación de la información de la alarma, donde la señal de activación es enviada por el sistema de control y cuando la aplicación recibe la información correspondiente a la señal de alarma, muestra una señal de aviso (la imagen colocada en la pantalla principal), y enciende una alarma de tipo visual, modifica el color de fondo del arreglo vertical en el que está contenida la información del precio del consumo ($$$).

ILUSTRACIÓN 3.15 "SEÑAL DE ALARMA"

El próximo segmento de código, muestra el proceso en el que se evalúa la petición de cambiar de pantalla, cuando el botón de configuración es presionado. Ya sea por error o con motivo, para cambiar de pantalla se pide un código, para evitar posibles problemas.

ILUSTRACIÓN 3.16A "ENTRANDO A CONFIGURACIÓN"

El código mostrado arriba, describe el funcionamiento del protocolo generado para entrar a la pantalla de configuración. Cuando se presiona el botón de configuración, se activa el cuadro de contraseñas, y el botón de la primera fila de la pantalla principal, para poder ingresar a la pantalla de configuración se necesita escribir un código en el cuadro de contraseñas y presionar el botón de "Ok" (es el aviso que muestra el notificador).

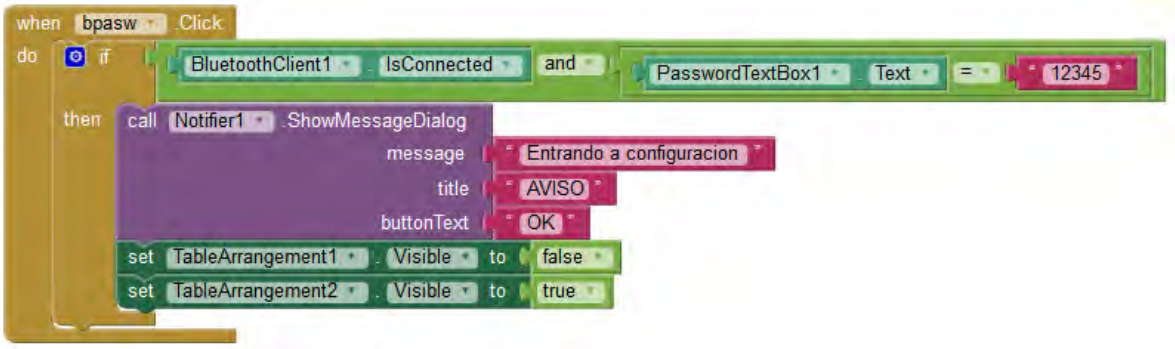


ILUSTRACIÓN 3.16B "ENTRANDO A CONFIGURACIÓN"

Cuando el botón de Ok es presionado, el código mostrado arriba se ejecuta, se evalúa la información escrita en el cuadro de contraseña, y si es equivalente a la contraseña configurada a la información del cuadro de contraseñas y si existe una conexión con el módulo bluetooth se permite el cambio de pantalla. El arreglo de tabla I, en el que se encuentran contenidos todos los elementos que componen a la pantalla principal desaparece

(su propiedad de visible se desactiva), y la pantalla de configuración aparece (lo contrario a la pantalla principal).

Una vez que la pantalla de configuración es mostrada, se deben de completar todos los campos de datos en la pantalla, pues la información será enviada al sistema de control y si existe un vacío en los datos, la información es recorrida, lo que genera problemas con los cálculos en el sistema de control. Por ello es muy importante llenar todos los campos de la pantalla. Los valores que se deben escribir en las casillas solo deben contener números, de

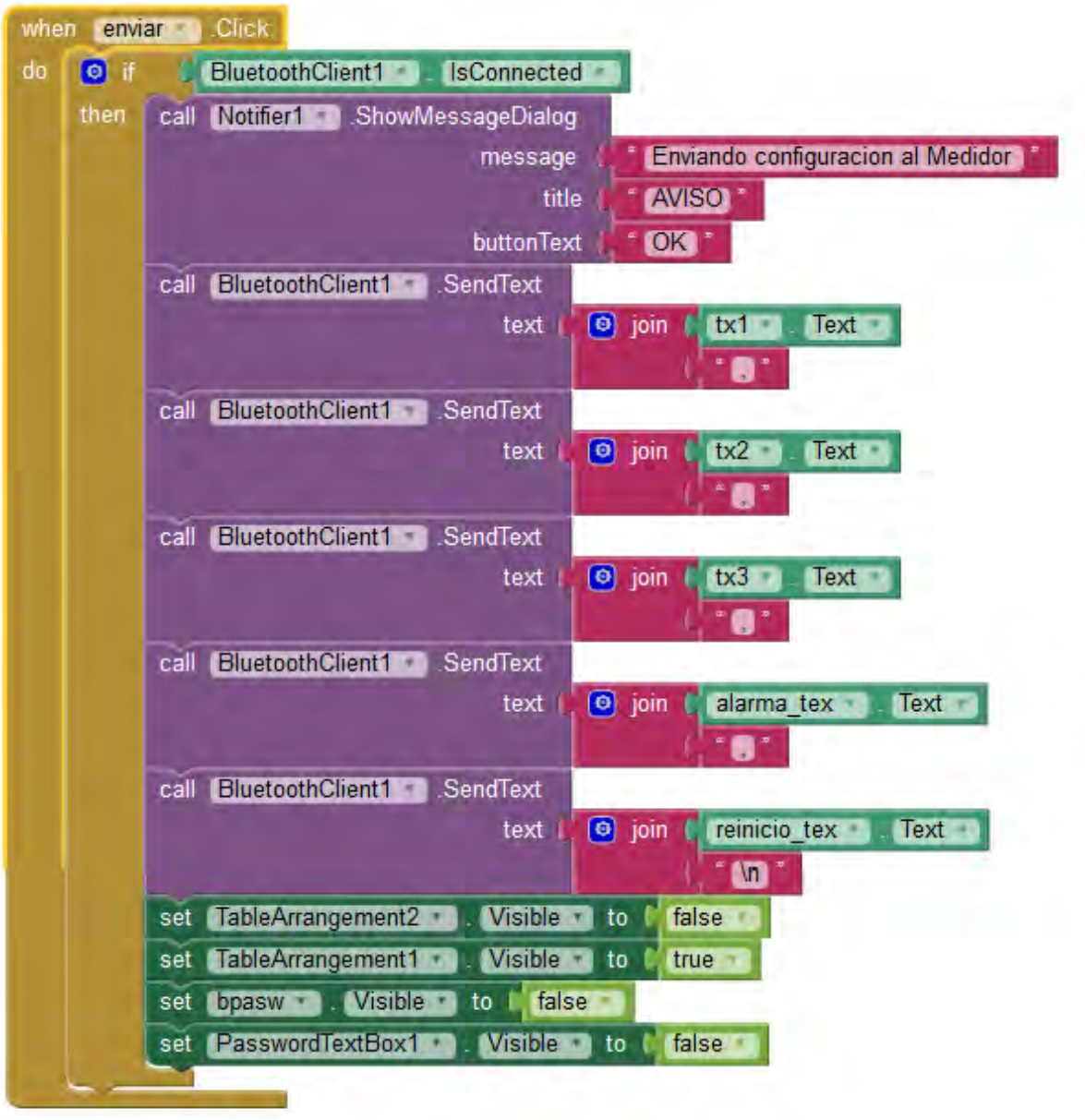


ILUSTRACIÓN 3.17 "ENVÍO DE DATOS Y CAMBIO DE PANTALLA"

preferencia tres dígitos.

Cuando se han llenado los datos en la pantalla se presiona el botón enviar, el cual, comienza a realizar las instrucciones mostradas en la ilustración 3.17; lo que realiza es el envío de información a través de la comunicación bluetooth, al final de todos los datos, envía un salto de línea, que el sistema de control interpreta como un final de envío de datos, las últimas instrucciones, cambian las pantallas, la pantalla de configuración se vuelve invisible y aparece la pantalla principal. Una vez en la pantalla principal, se realiza la operación de recibir la información sobre el consumo eléctrico, la cantidad a pagar por dicho consumo, y los días faltantes para que el periodo termine.

Capítulo 4. Pruebas y Resultados

Como pruebas para demostrar el funcionamiento, resultados y eficiencia con la que el dispositivo trabaja, se realizaron una serie de mediciones y se almacenaron en una memoria; tanto del sistema creado, como las de un medidor certificado por la CFE; además, se probó el correcto funcionamiento, según lo esperado, de cada función que se pretendió dotar al sistema.

4.1 Pruebas y resultados del prototipo 1

La primera prueba que se realizó al prototipo 1 consistió en la comparación de los valores entregados por el sistema, contra las mediciones obtenidas por un amperímetro, ambos dispositivos se encuentran midiendo el mismo evento, bajo las mismas condiciones.

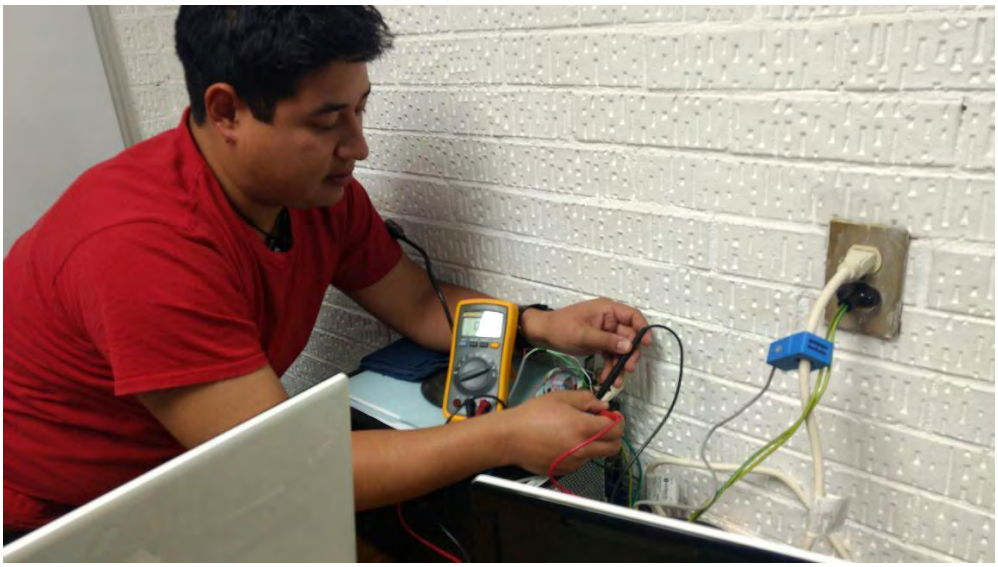


ILUSTRACIÓN 4.1 "PRUEBA DE CORRIENTE"

La prueba consistió en medir la cantidad de corriente que consume una lámpara de 75 watts, alimentada con 125 Volts de corriente alterna.

Para tener un estimado de la cantidad de corriente que puede demandar el foco, se realizaron algunos cálculos matemáticos, se utilizó la expresión:

$$P = V \times I \tag{4.1}$$

donde, se despejo a la corriente, dado a que es la única incógnita, porque se conoce el valor de la potencia y del voltaje, quedando la siguiente expresión:

$$I = \frac{P}{V} \tag{4.2}$$

Sustituyendo los valores:

$$I = \frac{75\ W}{125V} = 0.6\ Amperes$$

Ahora se conoce una cantidad que se puede tomar como referencia para las mediciones.

Se conectó el Prototipo 1 a la lámpara incandescente, recordemos que el prototipo 1 utiliza un sensor de corriente de tipo no invasivo, por lo que solo hay que colocarlo en el cable de alimentación de la lámpara (ilustración 4.1), se encendió la lámpara y se anotó la lectura del prototipo 1, se apagó la lámpara. Se conectó el amperímetro en serie al cable de alimentación de la lámpara, se encendió la lámpara y se anotó la lectura, se apagó la lámpara.

Este proceso se realizó un total de 10 veces, siguiendo la misma serie de pasos.

Los resultados obtenidos (ver tabla 4.1) se muestran a continuación:

Tabla 4.1 “Resultados primer experimento”

# de medición	Amperímetro	Prototipo1
1	0.56	0.82
2	0.55	0.76
3	0.55	0.80
4	0.56	0.96
5	0.55	1.02
6	0.56	0.92
7	0.56	0.72
8	0.55	1.01
9	0.55	0.85
10	0.54	0.96

Revisando los resultados, se nota por simple inspección, que las lecturas proporcionadas por el prototipo 1, no son parecidas, ni siquiera su razón de cambio es proporcional o lineal, lo que llevo a pensar que el prototipo 1 se encuentra midiendo de manera errónea la información. Para aclarar esta situación, se realizó otra prueba, en la que ambos sistemas (amperímetro y prototipo 1) se conectaron al mismo tiempo y medirían la misma lámpara, solamente que ahora, la lámpara se encendería y apagaría en intervalos aleatorios.

Los resultados obtenidos (tabla 4.2), mostraron que el prototipo 1 no era capaz de proporcionar datos fiables, pues cuando se apagaba la lámpara, el prototipo 1 seguía entregando valores. Lo que significa que el prototipo no se encuentra trabajando adecuadamente.

Tabla 4.2 “Resultados segundo experimento”

Estado Lámpara	**Amperímetro**	**Prototipo1**
Encendido	0.56	0.32
Apagado	0.0	0.55
Encendido	0.55	0.48
Apagado	0.0	0.0
Encendido	0.55	1.03
Apagado	0.0	0.62

Lo que resultó en una nueva búsqueda de información, y de elementos para la medición de la corriente. Pues el sensor que se utilizó en el prototipo 1 era el SCT-013, el cual, según datos del fabricante (Chen, 2011), no es lineal, además la cantidad máxima en voltaje que puede entregar cuando existe una cantidad de 30A es 1V, lo que reduce el rango de medición para los valores, pues se necesitaría de una mayor resolución para lograr obtener toda la información del sensor.

Los resultados obtenidos por parte del prototipo 1 no fueron los esperados, las mediciones no coincidían, no había relación entre los valores obtenidos; y el prototipo obtenía valores

de “la nada”; pues, aunque el dispositivo que se estaba midiendo, estaba apagado, el prototipo 1 seguía entregando resultados. Lo que conduce a que el programa, y el sensor no se encontraban funcionando correctamente. Por ello se detuvo el avance del prototipo y se comenzó con el desarrollo del segundo prototipo, tomando en cuenta los resultados obtenidos del primer prototipo se procuró tener más cautela a la hora de realizar el segundo prototipo.

4.2 Pruebas realizadas al prototipo 2

Durante el desarrollo del prototipo 2 se realizaron varias pruebas, a cada función que se agregaba al prototipo, con el fin de comprobar el correcto funcionamiento de cada nueva característica implementada.

4.2.1 Pruebas de medición de corriente

La misma prueba efectuada en el prototipo 1 (la lámpara de 75W), se realizó con el prototipo 2, con la modificación de que el sensor utilizado es del tipo invasivo, por lo que hubo que conectar el sensor en serie con la entrada de la lámpara, se obtuvieron los siguientes resultados:

Tabla 4.3 “Resultados primera prueba”

# de medición	Amperímetro	Prototipo 2
1	0.56	0.54
2	0.55	0.53
3	0.55	0.54
4	0.56	0.55
5	0.55	0.53
6	0.56	0.54
7	0.54	0.52
8	0.55	0.53
9	0.56	0.54
10	0.55	0.52

Para la segunda prueba en donde se apaga y se enciende la lámpara, mientras que el prototipo 2 y el amperímetro se encuentran conectados, se obtuvieron los siguientes resultados:

Tabla 4.4 “Resultados segunda prueba”

Estado Lámpara	**Amperímetro**	**Prototipo1**
Encendido	0.56	0.54
Apagado	0.0	0.1
Encendido	0.55	0.52
Apagado	0.0	0.0
Encendido	0.55	0.53
Apagado	0.0	0.1

4.2.1.1 Resultados en pruebas de corriente

Los resultados obtenidos de las pruebas de medición de corriente fueron prometedores, pues en ambas pruebas, se aproximaron mucho las mediciones realizadas con el amperímetro y el prototipo 2, y ambas lecturas son muy cercanas a los cálculos realizados.

Lo que indica que el sensor y el programa implementado en el sistema de control, funcionan correctamente, pues el prototipo mide de manera correcta la corriente, por lo que se puede continuar en el desarrollo del prototipo 2.

4.2.2 Método Bland-Altman

Este método estadístico, permite establecer un grado de confianza entre los resultados de las mediciones obtenidas por dos dispositivos distintos que se encuentran monitoreando un mismo evento que es variante en el tiempo.

Esta prueba consiste en comparar el consumo de la energía en una casa, comparando los resultados obtenidos por dos dispositivos que se encuentran realizando la medición del consumo energético. Se compararon, las mediciones obtenidas por el prototipo 2, contra las mediciones del medidor profesional y certificado, instalado por CFE.

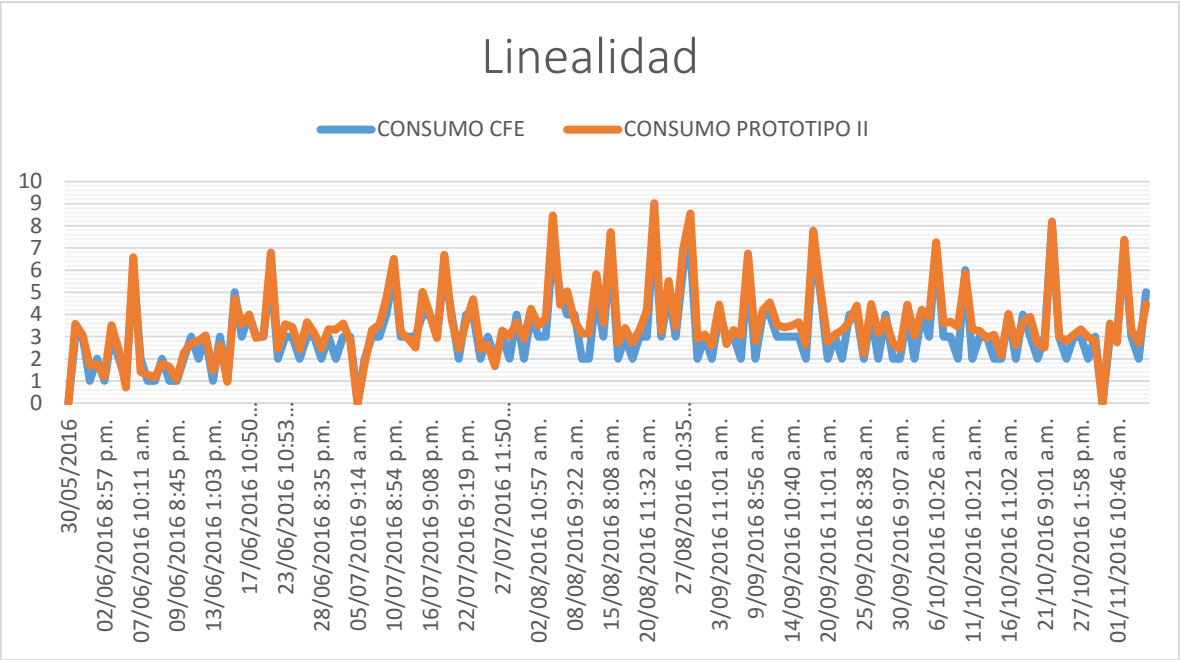


ILUSTRACIÓN 4.2"DATOS MÉTODO BLAND-ALTMAN"

Los valores para esta prueba se han tomado desde el 30 de mayo del 2016, hasta el 6 de noviembre del 2016, donde se ha tomado una captura a diario, de las mediciones de ambos medidores, se han introducido en una hoja de Excel donde se calculan las cantidades de las mediciones de cada día. Arriba, se muestra una gráfica donde se representan las mediciones de ambos sistemas.

Para utilizar el método de Bland-Altman, se utilizó un programa que realiza los cálculos para poder llevar a cabo el método, se introdujeron los datos, y se obtuvo la siguiente gráfica:

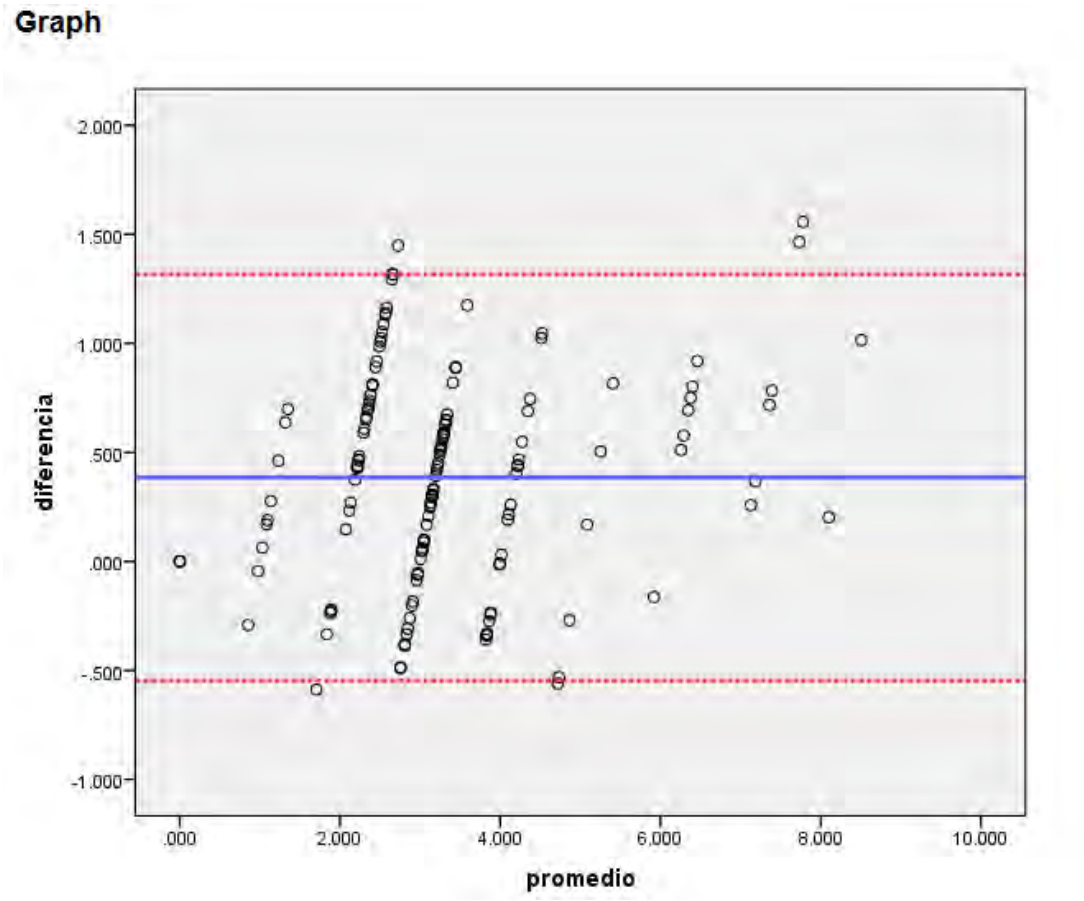


ILUSTRACIÓN 4.3 GRAFICA MÉTODO BLAND-ALTMAN

En la gráfica se puede apreciar, que la mayoría de los puntos, se encuentran dentro de los márgenes rojos, los cuales indican un 95% de confianza, lo que quiere decir, que mientras más puntos de la gráfica, se encuentren entre los rangos, los resultados tendrán una mayor concordancia. Para este experimento, de las 150 muestras tomadas de los medidores, solamente un 3% se encuentran fuera del rango, por lo que existe un 97% de concordancia entre las mediciones. Lo que indica que los sistemas estudiados son iguales.

4.2.3 Pruebas realizadas al funcionamiento de los dispositivos

Las pruebas realizadas a los sensores se llevaron a cabo para corroborar el correcto funcionamiento de los elementos que componen al prototipo.

4.2.3.1 Experimento funcionamiento módulo Bluetooth.

ILUSTRACIÓN 4.4 "ADQUISICIÓN DE DATOS EN LA INTERFAZ"

El módulo bluetooth proporciona una comunicación serial asíncrona inalámbrica entre dos dispositivos, puede enviar y recibir información. La prueba realizada a este módulo, consistió en supervisar los datos enviados y recibidos a través del puerto serial, para ello se utilizó el monitor serial con el que cuenta el entorno, revisando que la información recibida coincida con la información que se envió desde la aplicación.

Los resultados obtenidos, muestran que la información enviada por el sistema de control es recibida y procesada por la interfaz en el teléfono inteligente, por lo que el funcionamiento del módulo bluetooth es el correcto. Se puede ver en las imágenes que el envío se está realizando y la interfaz está recibiendo los datos.

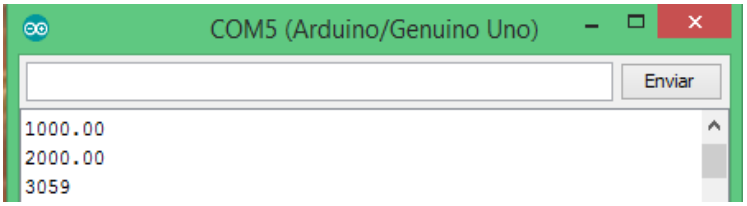


ILUSTRACIÓN 4.5 "ENVÍO DE DATOS POR PUERTO SERIAL"

4.2.3.2 Experimento de funcionamiento del Reloj de tiempo Real y Módulo memoria MicroSD

Se realizó una prueba con una función del prototitpo2 que incluye el funcionamiento del módulo de memoria MicroSD y el reloj de tiempo real. En esta etapa se realiza un proceso de lectura de la fecha y hora, donde cada 10 minutos se almacena en el archivo creado en la memoria, el registro de la sumatoria del consumo, acompañado de la fecha y hora cuando se guardó el dato.

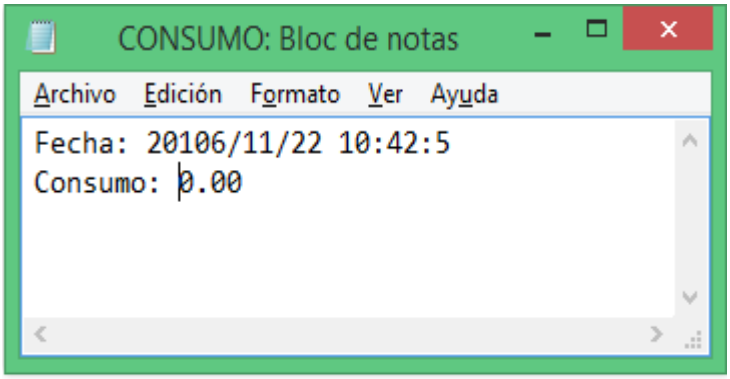


ILUSTRACIÓN 4.6 "ALMACENAMIENTO DE CONSUMO CON FECHA Y HORA"

Los resultados se obtuvieron analizando el contenido de la memoria, y al analizar los valores se constató que se obtuvieron los resultados esperados; por lo que el reloj de tiempo real y el módulo de memoria se encuentran trabajando correctamente.

4.2.4 Pruebas a la interfaz en Android

La prueba realizada a la interfaz, consistió en el manejo de la aplicación por una persona adulta mayor, a la que se le explico brevemente, cómo funciona la aplicación y que es lo que realiza; esto para que la persona indicará que elementos no están claros, cuáles, si lo son, y que aspectos se pueden mejorar.

Los resultados fueron los siguientes:

- El adulto mayor se sintió confundido a la hora de ingresar la contraseña en el teléfono, pues no encontraba los números en el teclado del teléfono inteligente.
    - Se mejoró la utilidad del teclado a la hora de introducir la información numérica, estableciendo que el teclado en la aplicación, muestre primero los números, en vez de las letras.
- Observó que la visualización de la información no era la adecuada, pues los números y letras no lograba distinguirlas.
    - Se modificó el tamaño de la fuente, con el fin de que la información se logrará apreciar mejor.
- Mostró confusión a la hora de ingresar los datos sobre las tarifas, la alarma y la duración del periodo.
    - Se propuso realizar un menú de ayuda para que explicará, que datos introducir en cada campo.

Se tomaron en cuenta las recomendaciones y se realizaron los cambios propuestos (ya están incluidos en el desarrollo de la interfaz).

Conclusiones

Se realizó una búsqueda de información acerca del estado de la generación de la energía eléctrica en el país, lo que resultó en el hallazgo de la empresa que suministra el servicio, desde la producción, hasta la transmisión de la energía eléctrica, CFE.

Se reconocieron las circunstancias del cobro del suministro eléctrico, que se realiza mediante la clasificación de usuarios por rangos de tarifas, lo que llevó a poder incluir en el funcionamiento del sistema de esta propuesta, una estimación de la cantidad a cobrar por el suministro de la energía eléctrica.

Se realizaron pruebas con distintos métodos de medición de energía eléctrica, se utilizó un sensor de corriente de tipo no invasivo, el cual, dadas sus características de funcionamiento, no fue capaz de proporcionar la información requerida para el funcionamiento del sistema de esta propuesta. También se utilizó un sensor de corriente de tipo invasivo, en el que se tuvo que abrir una sección del cable conductor para poder insertar el sensor, el cual, funciono de la manera deseada para el sistema de este trabajo.

Se definió el funcionamiento del dispositivo en base a un microcontrolador contenido en una tarjeta de desarrollo conocida como Arduino, donde se lograron crear, varias funciones alrededor de la obtención del consumo de la energía eléctrica. Una de las funciones desarrolladas, tiene fundamento base, en el funcionamiento del cobro del servicio, para poder aproximar el precio del consumo realizado en un periodo.

Se implementó un reloj de tiempo real, para el monitoreo del tiempo, para poder establecer horarios en los que la información es almacenada en la memoria interna del dispositivo, y en el módulo de la memoria MicroSD.

Se diseñó una aplicación para el sistema operativo Android, la cual funciona como interface visual y de configuración, para que el usuario logre ver la información proporcionada por el sistema y pueda configurar cambios en el sistema.

La comunicación de la aplicación-sistema (y viceversa), se realiza mediante la transferencia de datos mediante la tecnología conocida como comunicación Bluetooth, la cual, envía y

transmite paquetes de datos de manera inalámbrica, enviando los datos a manera de ondas electromagnéticas del rango de Giga Hertz.

Se utilizó el método estadístico Bland-Altman, para analizar los valores obtenidos por el sistema de esta propuesta, con respecto a los valores obtenidos por un medidor profesional y certificado por la empresa CFE. Los resultados obtenidos muestran un alto grado de concordancia (del 97%), entre los valores obtenidos por los sistemas, por lo que la medición del consumo eléctrico, es igual para ambos sistemas.

El dispositivo que se diseñó, construyo, implemento y se describió su funcionamiento a lo largo de esta propuesta, cumple con los objetivos planteados de este trabajo. Pues el dispositivo es capaz de llevar a cabo las tareas definidas en el primer capítulo.

Quedaron algunas características del funcionamiento del dispositivo en desarrollo, es decir, que se proponen desarrollar como trabajo futuro, un menú de ayuda que muestre al usuario que información debe introducir en los campos de captura. Y un menú, en el que el usuario, pueda configurar el mismo la fecha y la hora en el dispositivo.

Por lo que, de acuerdo con lo establecido en los objetivos de esta investigación puedo concluir que el dispositivo desarrollado cumple con lo establecido en el objetivo general, permitiéndome confirmar, que el sistema creado, es una herramienta de gran auxilio, para conocer el estado del consumo eléctrico durante un periodo, así como un estimado del precio a cubrir por el consumo realizado. Para que el usuario logre definir una línea de acción para corregir el estado de su consumo y así poder generar una conciencia del cuidado de la energía eléctrica.

De manera personal, me siento satisfecho con la realización de este trabajo de investigación e integración tecnológica, no solo por el requisito como tramite de titulación, sino como una culminación de una etapa muy importante en mi vida.

Muchas gracias.

Bibliografía

Allegro Microsystems. (16 de Noviembre de 2012). *allegromicro.* Obtenido de www.allegromicro.com/~/media/.../datasheets/acs712-datasheet

Arribas, J. (03 de Octubre de 2014). *pisos.com.* Obtenido de http://www.pisos.com/hogar/bricolaje/tus-reformas/domotica/medidores-de-consumo-y-control-de-la-energia-electrica/

Artero, O. T. (2013). *Arduino, Curso practico de formacion.* Distrito Federal: Alfaomega.

Atmel. (01 de 11 de 2015). *atmel.* Obtenido de http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Banzai, M. (2009). *Getting Started with Arduino.* Sebastopol, California: O'Reilly.

Cavada, G. (01 de Marzo de 2013). *revistasoched.* Obtenido de http://revistasoched.cl/3-2013/8-Cavada.pdf

Chen, E. (27 de Julio de 2011). *webmeteobox.* Obtenido de http://webmeteobox.ru/docs/SCT013-030V.pdf

Comision Federal de Electricidad. (10 de noviembre de 2014). *Comision Federal de Electricidad.* Obtenido de http://www.cfe.gob.mx/ConoceCFE/1_AcercadeCFE/CFE_y_la_electricidad_en_Mexico/Paginas/CFEylaelectricidadMexico.aspx

Comision Federal Electricidad. (1 de Enero de 2016). *Comision Federal Electricidad.* Obtenido de http://app.cfe.gob.mx/Aplicaciones/CCFE/Tarifas/Tarifas/tarifas_casa.asp

Comision reguladora de energia. (1 de enero de 2014). *comision reguladora de energia.* Obtenido de http://www.cre.gob.mx/documento/3045.pdf

Elecrow. (7 de Octubre de 2014). *openhacks.* Obtenido de https://www.openhacks.com/uploadsproductos/tiny_rtc_-_elecrow.pdf

Electrica la guia del electrisista . (1 de Agosto de 2015). *Electrica la guia del electrisista* . Obtenido de http://electrica.mx/la-evolucion-de-los-medidores-de-energia-electrica-watthorimetros/

energia, A. i. (17 de 10 de 2016). *BANCO MUNDIAL*. Obtenido de BANCO MUNDIAL: http://datos.bancomundial.org/indicador/EG.ELC.ACCS.ZS

Giavarina, D. (2 de Junio de 2015). *Biochem Med (Zagreb)*. Obtenido de https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4470095/

ITeadStudio. (18 de Junio de 2010). *electronica estudio.* Obtenido de www.electronicaestudio.com/docs/istd016A.pdf

Kuo, B. C. (1996). *Sistemas de control automatico* . Estado de México: Prentice Hall.

Maargolis, M. (2011). *Arduino Cookbook.* Sebastopol CA: O'Reilly.

Ogata, K. (2010). *Ingenieria de control moderna.* Madrid: Pearson.

QIU. (29 de 10 de 2008). *sparkfun.* Obtenido de https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

Sistema interconectado nacional. (03 de Noviembre de 2016). *gob.mx.* Obtenido de http://www.cenace.gob.mx/GraficaDemanda.aspx

Anexos

A) Código Arduino

```
#include <SD.h>
#include <SPI.h>
#include <EEPROM.h>
#include "RTClib.h"
#include <Wire.h>

RTC_DS1307 RTC;
RTC_Millis rtc;
#define RESTART asm("jmp 0x0000");

File medicion;

const int volts=120.0;
double Potencia=0.0;
double PotenciaZ=0.0;
float PotenciaF=0.0;
double PotenciaFaux=0.0;

byte D;
byte B;

byte ht1;
byte lt1;
byte ht2;
byte lt2;
```

```
byte ht3;
byte lt3;
byte halar;
byte lalar;
byte hrei;
byte lrei;

int x=0;
int y=0;
int mes=0;
int dia=0;
int h=0;
int m=0;
int s=0;

const int sensorIn = A0;
int mVperAmp = 111; // use 100 for 20A Module and 66 for 30A Module

double Voltage = 0;
double VRMS = 0;
double AmpsRMS = 0;


int tarifa1;
int tarifa2;
int tarifa3;
```

```
int alarma;
int reinicios=60;

int tarifas1=793;
int tarifas2=956;
int tarifas3=280;
int alarmas=200;
int reinicio;

int inicio = 0;

double dinero;



void setup()
{
 pinMode(5,OUTPUT);
 Serial.begin(9600);
 Wire.begin();
 RTC.begin();
rtc.begin(DateTime(F(__DATE__), F(__TIME__)));

   if (!RTC.isrunning())
   {
     Serial.println("RTC NO FUNCIONA!");
     delay(2000);
     Serial.println("");
    RESTART;
```

```
}

else if(RTC.isrunning())
{
  Serial.println("  RTC FUNCIONA  ");
  rtc.begin(DateTime(F(__DATE__), F(__TIME__)));
  delay(1000);
  Serial.println("");
 }
if (!SD.begin(4))
{
 Serial.println("Error memoria SD");

}       //Lectura de valores almacenados en EEPROM
 for (int i = 0; i <= 3 ; i++)
 {
   //Se almacena el valor en la EEPROM en los bits menos significativos.
   D = EEPROM.read(i);
   long(PotenciaFaux);
   //Se desplazan los bits tantos bytes como sea necesario, teniendo en
   //cuenta que el byte final no debe ser desplazado.
   if (i <= 3)
   {
     PotenciaFaux = PotenciaFaux << 8;
   }
  //Se almacena en la variable final el resultado de la operación OR de
  //ambos datos. Al realizar la operación OR con una variable que solo
  //contiene ceros (variable Dato), el resultado es el valor de la otra
```

```
  //variable. OR implica => 0+0=0, 0+1=1, 1+0=1, 1+1=1.
 PotenciaFaux = D || PotenciaFaux;
 //Se limpia la variable auxiliar para que en las siguientes iteraciones
 //no se cambie el resultado de los otros bytes.
 D = 0;
}
//lectura de valor tarifa 1
  ht1=EEPROM.read(4);
  lt1=EEPROM.read(5);
  tarifas1= ht1;
  tarifas1= tarifas1 <<8;
  tarifas1= tarifas1|lt1;

//lectura de valor tarifa 2
  ht2=EEPROM.read(6);
  lt2=EEPROM.read(7);
  tarifas2= ht2;
  tarifas2= tarifas2 <<8;
  tarifas2= tarifas2|lt2;

//lectura de valor tarifa 3
  ht3=EEPROM.read(8);
  lt3=EEPROM.read(9);
  tarifas3= ht3;
  tarifas3= tarifas3 <<8;
  tarifas3= tarifas3|lt1;

//lectura de valor alarmas
```

```
    halar=EEPROM.read(10);
    lalar=EEPROM.read(11);
    alarmas= halar;
    alarmas= alarmas <<8;
    alarmas= alarmas|lalar;

  //lectura de valor fecha de corte
    hrei=EEPROM.read(12);
    lrei=EEPROM.read(13);
    reinicios= hrei;
    reinicios= reinicios <<8;
    reinicios= reinicios|lrei;
}

void loop()
{

 Voltage = getVPP();
 VRMS = (Voltage/2.0) *0.707;
 AmpsRMS = (VRMS * 1000)/mVperAmp;

 Potencia= volts*AmpsRMS;

  // Se realiza la sumatoria del consumo
  PotenciaZ=PotenciaZ+Potencia;
  PotenciaZ=PotenciaZ-13.06;

  PotenciaF=PotenciaZ*0.0003; //Wh
```

```
 PotenciaF=PotenciaF/1000.0; //KWh
 PotenciaFaux=PotenciaFaux+PotenciaF; //registro sumatoria

 DateTime now = rtc.now();
  y=now.year();
  mes=now.month();
  dia=now.day();
  h=now.hour();
  m=now.minute();
  s=now.second();

if(h==6)
{
  if(m==30)
  {
   if(s==10)
   {
    for (int i = 0; i <= 3; i++)
    {
     long(PotenciaFaux);
     //Se guarda el byte más significativo
     B = highByte(PotenciaFaux);
     EEPROM.write(i, B);
     //Se desplazan 8 bits hacia la izquierda de modo que los 8 primeros
     //bits se pierden y los 8 siguientes pasan a ser los primeros
     PotenciaFaux = PotenciaFaux << 8;
    }
    //fecha de corte
```

```
        hrei = highByte(reinicios);
        lrei = lowByte(reinicios);
        EEPROM.write(12, hrei);
        EEPROM.write(13, lrei);
      }
    }
}
if(h==12)
{
   if(m==30)
   {
     if(s==10)
     {
       for (int i = 0; i <=3; i++)
       {
       long(PotenciaFaux);
       //Se guarda el byte más significativo
        B = highByte(PotenciaFaux);
        EEPROM.write(i, B);
        //Se desplazan 8 bits hacia la izquierda de modo que los 8 primeros
        //bits se pierden y los 8 siguientes pasan a ser los primeros
        PotenciaFaux = PotenciaFaux << 8;
       }
       //fecha de corte
        hrei = highByte(reinicios);
        lrei = lowByte(reinicios);
        EEPROM.write(12, hrei);
        EEPROM.write(13, lrei);
```

```
    }
   }
 }

delay(100);
while(Serial.available())
{
  tarifa1=Serial.parseInt();// recepcion de datos x parte de la app
  tarifa2=Serial.parseInt();
  tarifa3=Serial.parseInt();
  alarma=Serial.parseInt();
  reinicio=Serial.parseInt();

    if(Serial.read()=='\n')//finaliza recepcion de datos
  {
   tarifas1=tarifa1;  //carga de datos recibidos a variables fija
   tarifas2=tarifa1;
   tarifas3=tarifa1;
   alarmas=alarma;
   reinicios=reinicio;
  }
 //Guardar informacion en EEPROM
//tarifa1
        //Se guarda el byte más significativo
        ht1 = highByte(tarifas1);
        lt1 = lowByte(tarifas1);
        EEPROM.write(4, ht1);
        EEPROM.write(5, lt1);
```

```
//tarifa2

        ht2 = highByte(tarifas2);
        lt2 = lowByte(tarifas2);
        EEPROM.write(6, ht2);
        EEPROM.write(7, lt2);


//tarifa3

        ht3 = highByte(tarifas3);
        lt3 = lowByte(tarifas3);
        EEPROM.write(8, ht3);
        EEPROM.write(9, lt3);


//alarmas

        halar = highByte(alarmas);
        lalar = lowByte(alarmas);
        EEPROM.write(10, halar);
        EEPROM.write(11, lalar);

//fecha de corte
        hrei = highByte(reinicios);
        lrei = lowByte(reinicios);
```

```
        EEPROM.write(12, hrei);
        EEPROM.write(13, lrei);
  }
//Reinicio a los 60 dias

if(h==12)
   {
     if(m==30)
     {
      if(s==15)
      {
        inicio=inicio+1;
      }
     }
   }

if(inicio==2)
   {
    reinicios=reinicios-1;
    inicio=0;
   }

if(reinicios==0)
   {
    for (int i = 0; i <= 3; i++)
       {
         EEPROM.write(i, 0);
       }
```

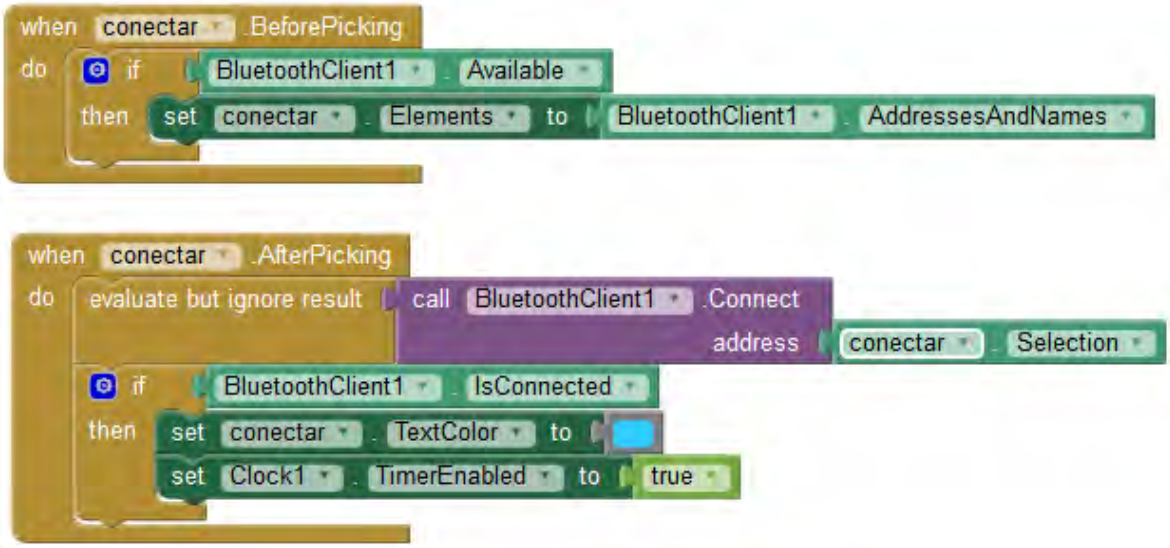
```
   RESTART;
  }
 //Para la alarma, de tipo monetaria
  tarifas1=tarifas1*0.001;
  tarifas2=tarifas2*0.001;
  tarifas3=tarifas3*0.01;

 if(PotenciaF<=150.0)
  {
    dinero= PotenciaF*tarifas1;
  }
 if(150.0<PotenciaF<=280.0)
  {
    dinero= PotenciaF*tarifas2+(tarifa1*150);
  }
 if(PotenciaF>280.0)
  {
    dinero= PotenciaF*tarifas3+(tarifa1*150)+(tarifa2*130);
  }
if(dinero>(alarma-50))
  {
   digitalWrite(5,HIGH);
   Serial.print(100);
   Serial.println("    ");
  }

//Envio de informacion por puerto serial
  Serial.print(PotenciaF+1000);
```

```
Serial.println("    ");
Serial.print(dinero+2000);
Serial.println("    ");
Serial.println(reinicios+3000);
delay(1000);

for(int a=1; a<=12; a++)
{
 if(a==h)
 {
  if(m==30)
  {
   if(s==5)
   {
    medicion = SD.open("Consumo.txt", FILE_WRITE);
    if (medicion)
     {
      medicion.print("Fecha:");
      medicion.print(' ');
      medicion.print(y, DEC);
      medicion.print('/');
      medicion.print(mes, DEC);
      medicion.print('/');
      medicion.print(dia, DEC);
      medicion.print(' ');
      medicion.print(h, DEC);
      medicion.print(':');
      medicion.print(m, DEC);
```

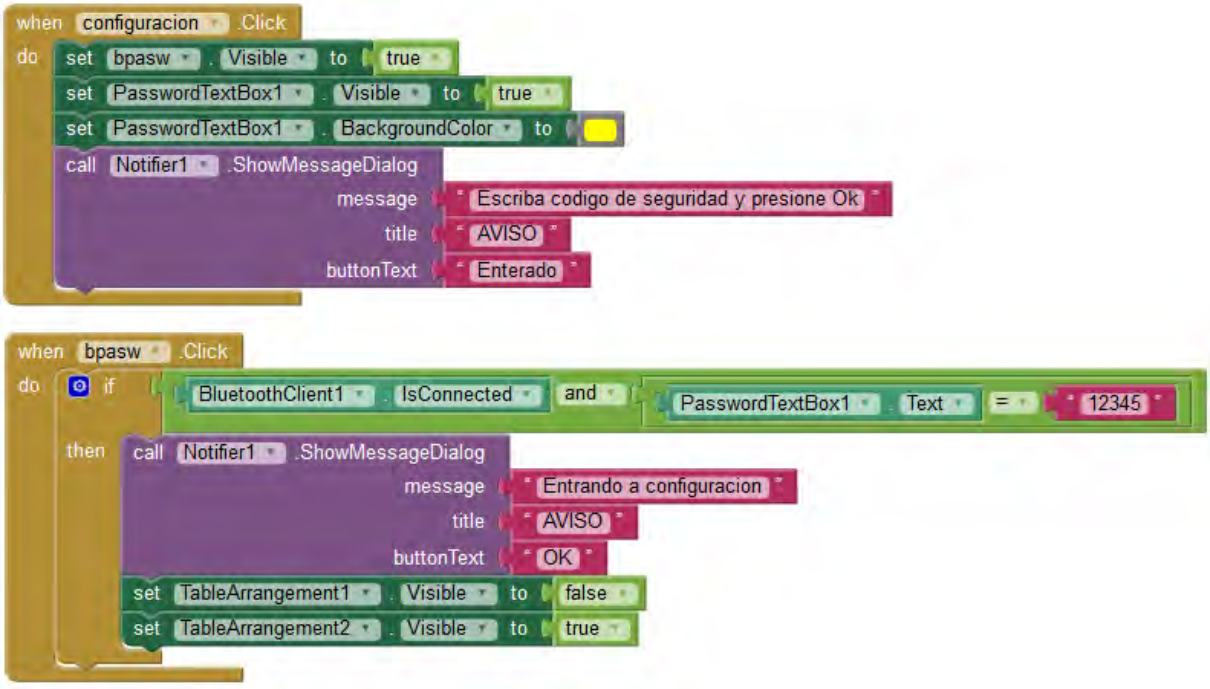
```
            medicion.print(':');
            medicion.print(s,DEC);
            medicion.println();
            medicion.print("Consumo:");
            medicion.print(' ');
            medicion.print(PotenciaFaux);
            medicion.println();
            medicion.close();
           }
           else
           {
           Serial.println("Error al guardar SD");
           }
         }
       }
     }
   }
}

float getVPP()
{
  float result;

  int readValue;             //valor obtenido del sensor
  int maxValue = 0;
  int minValue = 1024;

   uint32_t start_time = millis();
```

```
  while((millis()-start_time) < 50)
  {
     readValue = analogRead(sensorIn);
     // see if you have a new maxValue
     if (readValue > maxValue)
     {

        maxValue = readValue;
     }
     if (readValue < minValue)
     {



        minValue = readValue;
     }
  }
  result = ((maxValue - minValue) * 5.0)/1024.0;

  return result;
}
```

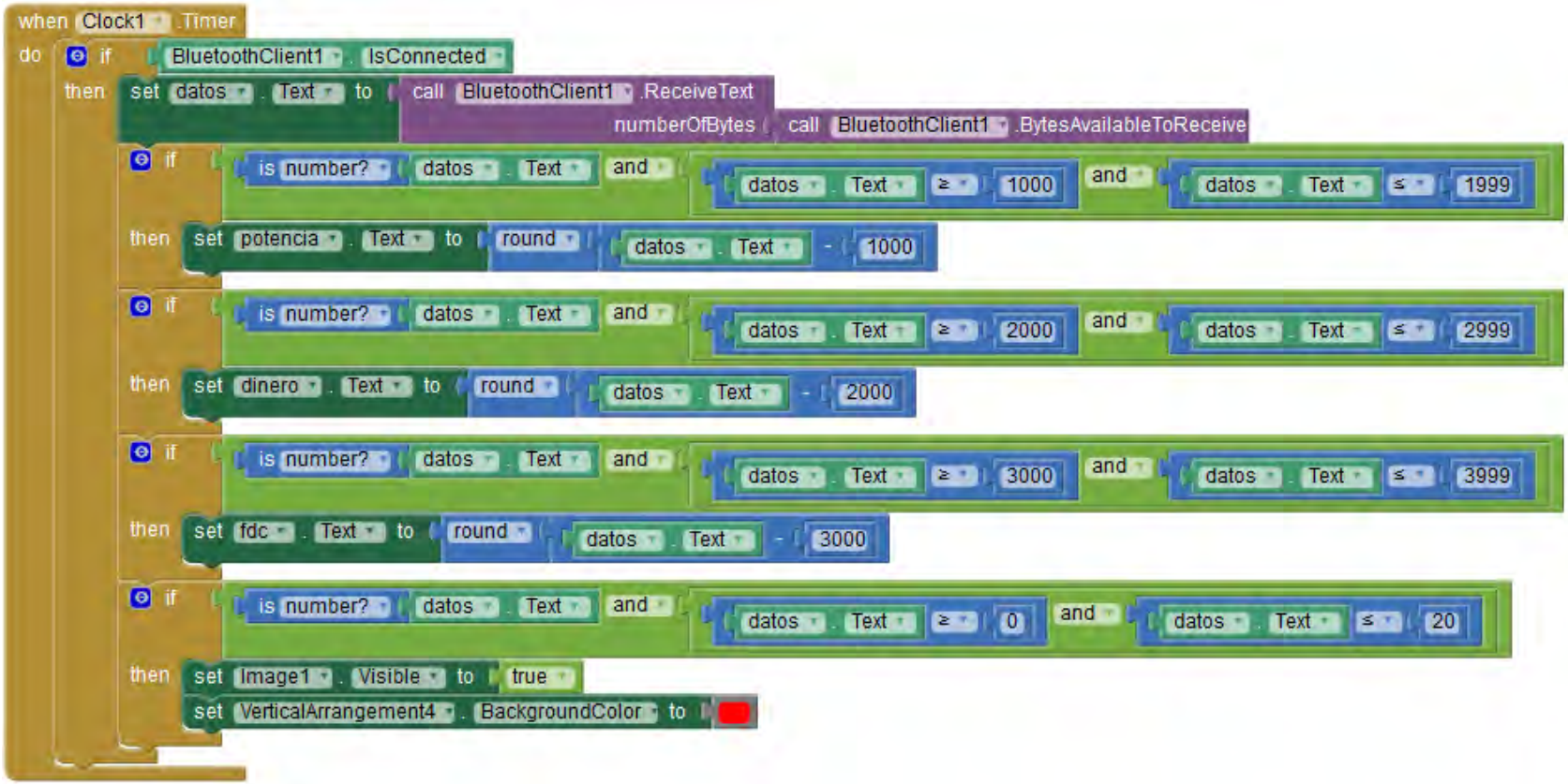
B) Código App inventor 2



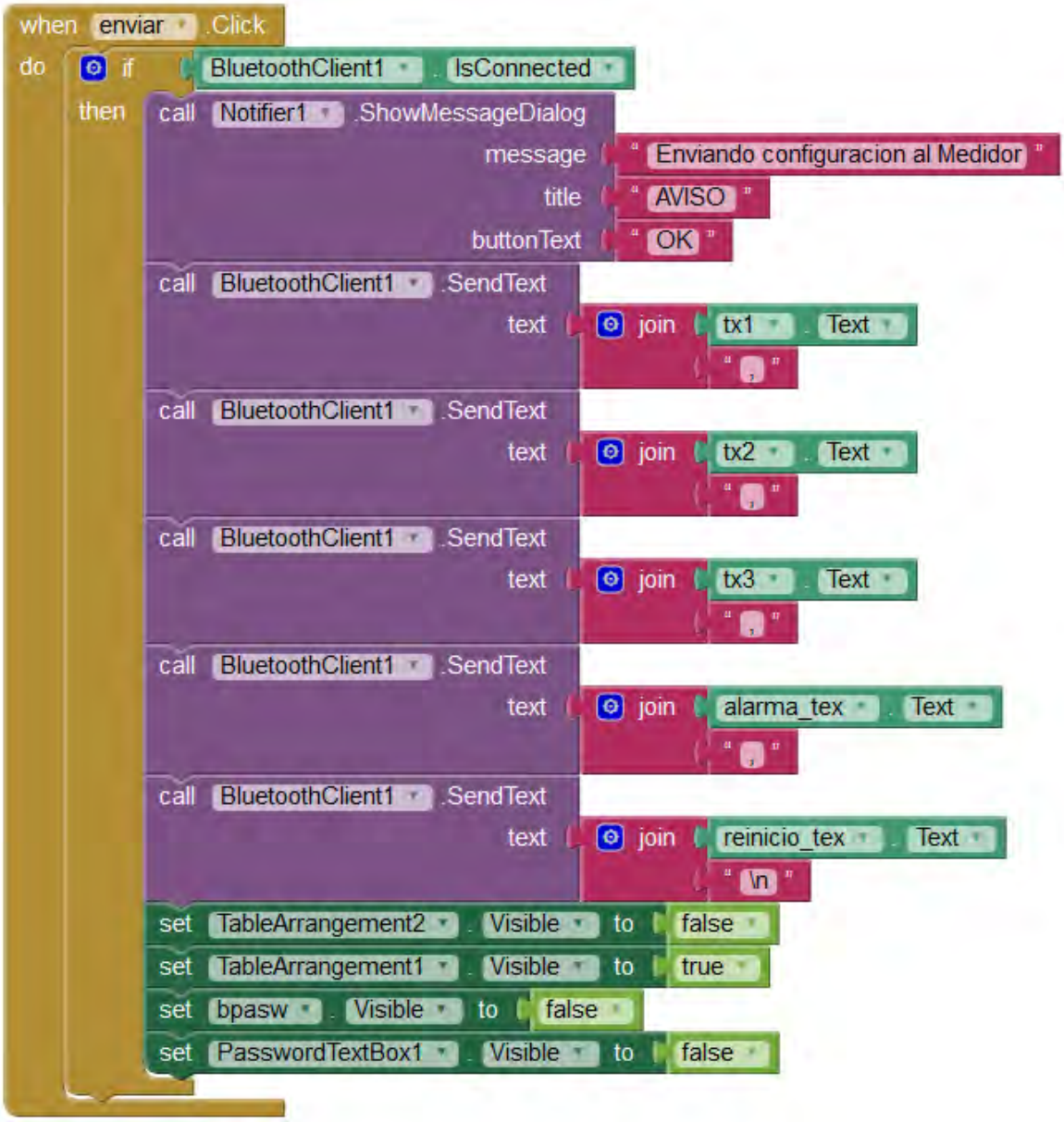
CÓDIGO DE CONFIGURACIÓN CONEXIÓN BLUETOOTH



CÓDIGO INGRESO A CONFIGURACIÓN



CÓDIGO RECEPCIÓN DE DATOS BLUETOOTH



CÓDIGO ENVIÓ DE INFORMACIÓN BLUETOOTH