Diseño e implementación de un sistema de medidor de consumo de energía eléctrica residencial prepago.

Darío Fernando López López

Asesor: Ing. Sindy Johana Escobar

Universidad Nacional Abierta y a Distancia UNAD.

Escuela de Ciencias Básicas, Tecnología e Ingeniería

Ingeniería Electrónica

Dosquebradas

2020

Tabla de contenido

Índice de figuras

Índice de figuras anexo 2

Lista de tablas

Introducción

En el presente trabajo de grado que lleva por nombre “Diseño e implementación de un sistema de medidor de consumo de energía eléctrica residencial prepago”, se realiza en base a investigaciones experimentales, cuyo objetivo central del trabajo, es el desarrollo de un dispositivo electrónico medidor de consumo de energía eléctrica residencial, esto para que las personas de escasos recursos económicos puedan administrar su servicio de energía, ajustando de manera efectiva el consumo en estos lugares.

Lo primero que se hace en este trabajo tanto escrito como experimental es, el de abordar antecedentes del proyecto, sus inicios, pruebas y evoluciones a lo largo de la historia, también es relevante mencionar sobre las implementaciones realizadas de los medidores de energía eléctrica residencial prepago a nivel país.

El proyecto está formado por dos partes, el software y el hardware. El software se divide en tres partes, la interfaz de usuario, el aplicativo lector de código QR y el código que controla la tarjeta electrónica. Por su parte el hardware se desarrolla en torno a arduino.

Finalmente se realiza la prueba de funcionamiento del prototipo integrando tanto el hardware y el software, esto con cargas eléctricas como bombillas incandescentes.

Definición del problema

Las personas de escasos recursos tienen que pagar un consumo eléctrico en un tiempo determinado, esto conlleva a que algunas veces las facturas de este servicio suelen tener valor muy alto, y además poco administrable e inasequible para este tipo personas.

Según la ley 142 y 143 de 1994, los usuarios tienen como derecho de que las empresas tienen que satisfacer las necesidades básicas de la población y asegurar la disponibilidad del servicio, además del uso racional de la energía, la eficiencia y transparencia en la prestación del servicio, manteniendo siempre un nivel de calidad y seguridad, también de obtener de las empresas la medición de sus consumos en tiempos reales mediante instrumentos tecnológicos apropiados dentro de los plazos y términos que se fijan por la comisión de regulación de energía y gas (CREG). (Normatividad - Ministerio de minas y Energia , 2019)[1]

Para la medición del consumo eléctrico en los hogares y las industrias, aquí en Colombia, desde siempre se han implementado medidores de tipo analógicos, estos elementos solo permiten censar el consumo de energía en un determinado periodo de tiempo, mas no tiene la capacidad de tener control sobre dichas cargas, lo que en resumen se puede decir que no hay un buen uso, o aprovechamiento de la energía eléctrica en una residencia común, esto también se traduce a que el usuario no tiene la posibilidad de aprovechar el recurso de energía esto siempre ajustándose a las tarifas del cobro del servicio eléctrico, mas no por el consumo racional que el usuario hace.

[1] (Normatividad - Ministerio de minas y Energía , 2019

Justificación

La justificación por la cual se pretende desarrollar este proyecto, es para que las personas de escasos recursos económicos puedan administrar su servicio de energía, ajustando de manera efectiva y racional el consumo en los lugares más vulnerables, esto gracias a los medidores de energía eléctrica prepago.

De acuerdo con el artículo 63 de la Ley 812 de 2003, el gobierno nacional desarrollará un programa de normalización de redes eléctricas en barrios subnormales.[2]

La normalización de los Barrios Subnormales implica la instalación o adecuación de las redes de distribución de energía eléctrica, la acometida a la vivienda del usuario, incluyendo el contador o sistema de medición del consumo el cual podrá ser un sistema de medición prepago.[3]

[2] (Comisión de Regulación de Energía y Gas, Programa de normalización)
[3] (CREG, Programa de normalización.)

Marco referencial

Medidores Prepago: es una herramienta de manejo de crédito, promovida por las empresas comercializadoras de servicios públicos domiciliarios para recuperar la deuda y prevenir la acumulación futura de la misma, brindando al mismo tiempo varios beneficios al usuario como gestión de la demanda y uso racional de energía.

Consumo prepago: Consumo que un suscriptor o usuario paga en forma anticipada a la empresa, ya sea porque el suscriptor o usuario desea pagar por el servicio en esa forma, o porque el suscriptor o usuario se acoge voluntariamente a la instalación de medidores de prepago.

Sistema de tarifa en energía eléctrica: La Comisión de Regulación de Energía y Gas CREG mediante resolución CREG 119 de 2007 aprueba la fórmula tarifaria general que permite a los Comercializadores Minoristas de electricidad establecer los costos de prestación del servicio a usuarios regulados en el Sistema Interconectado Nacional.[4] Siguiendo la metodología normativa desarrollada en esta y otras resoluciones calculamos los costos máximos de prestación del servicio de energía eléctrica y las tarifas aplicables a los usuarios finales regulados como se debe mostrar mensualmente, en específico la empresa de energía del Quindío Edeq S.A. E.S.P, para el mes de marzo de 2019 tiene una tarifa de 532.52 CU (costo unitario).[5]

[4] (Resolución de la CREG – Ministerio de Minas y Energía)
[5] (Edeq – tarifas de energía.)

Vatio (W): es la unidad de potencia estándar internacional. Habitualmente el vatio se asocia a la unidad de potencia eléctrica, es decir, vinculado a la electricidad. Sin embargo, esta unidad del sistema internacional de medidas también se utiliza para referirse a la energía mecánica, en el electromagnetismo, una diferencia de potencial eléctrico en voltios (V) se produce cuando un flujo de corriente de amperaje (A) es un vatio, es decir:

$$\boldsymbol{W = V * A}$$

Voltaje: Denominado también como tensión o diferencia de potencial es una magnitud física que impulsa a los electrones a lo largo de un conductor en un circuito eléctrico cerrado, provocando el flujo de una corriente eléctrica. En el Sistema Internacional de Unidades, la diferencia de potencial se mide en Voltios (V).

Corriente eléctrica: la corriente eléctrica es el flujo neto de carga eléctrica que circula de forma ordenada por un medio o material conductor, la unidad de medida es el amperio (A).

Potencia Activa (energía): es aquella que al ingresar a una instalación por los conductores de electricidad produce luz, calor y movimiento. La energía activa representa numéricamente la dedicación que tuvo una porción de las maquinas generadoras de electricidad hacia nuestra instalación durante una determinada cantidad de tiempo, o lo que es lo mismo, pero desde el punto de vista del consumidor, el gasto resultante del uso de sus equipos eléctricos durante cierta cantidad de tiempo. De una manera más técnica podemos decir que la potencia activa es la capacidad que tiene un equipo o artefacto eléctrico para desarrollar trabajo. A mayor potencia, el equipo estará en capacidad de desarrollar más trabajo, La unidad de potencia activa es el vatio (W), pero para una mejor lectura o una mejor cuantificación se utiliza los prefijos, el kilovatio (kW), igual a 1000 W, el megavatio (MW), igual a 1.000.000 W, el Gigavatio (GW),

igual a 1.000.000.000 W. la potencia activa es la cantidad total de potencia "útil" que consume un equipo eléctrico, esta potencia es utilizada tanto en circuitos DC (Corriente directa) como en AC (Corriente Alterna). Comúnmente los equipos que presentan su potencia en kW son los equipos resistivos, como estufas eléctricas, calefacción, bombillo incandescente etc.

Se puede calcular de la siguiente forma.

$$P_{Activa} = V * I * Cos\varphi$$

Potencia reactiva (energía): es la requerida para crear el campo magnético en las bobinas de motores, transformadores, balastos magnéticos, se puede decir que esta energía es generada por las inductancias de los motores que están conectados en la toma eléctrica. No es una potencia (energía) realmente consumida en la instalación, ya que no produce trabajo útil debido a que su valor medio es nulo, su unidad de medida es el voltio – amperio reactivo (Var)

Se puede calcular de la siguiente forma.

$$P_{Reactiva} = V * I * sen\varphi$$

Potencia aparente (energía): es la potencia total consumida por la carga y es el producto de los valores eficaces de tensión e intensidad. Se obtiene como la suma vectorial de las potencias activa y reactiva y representa la ocupación total de las instalaciones debidas a la conexión del receptor, y su unidad de medida es el voltamperio (VA).

$$\mathbf{P_A = V_{(Voltios)} * I_{(Amperios)}}$$

También se puede calcular con la siguiente formula.

$$P_A = \sqrt{P_{(Activa)^2} + P_{(Reactiva)^2}}$$

Factor de potencia: Es un indicador de consumo de la energía reactiva en la instalación en comparación con el consumo efectuado de la energía activa.

$$FP = \frac{KWh}{(kWh)2 + (kVARh)2} \qquad FP = \cos\left(\arctan\frac{kVARh}{kWh}\right)$$

Empresa distribuidora de energía eléctrica: El sistema eléctrico colombiano está estructurado por los procesos de Generación, trasmisión, distribución y comercialización de energía eléctrica. Las empresas de distribución de energía eléctrica son aquellas compañías que se dedican a transportar energía a nivel departamental y municipal en los hogares, comercio e industria.

Sistema de medida: Los medidores electrónicos de energía eléctrica con los cuales hoy cuenta Colombia, son en la modalidad de prepago y pospago y son de última generación, se operan en condición opuesta a online, es decir en modo off-line y con sistemas de comunicación alámbricas e inalámbrica, que permiten controlar en tiempo real bajo el sistema de medición avanzada inteligente AMI. Son medidores monofásicos, trifilares y trifásicos bicuerpo para operación en pospago o prepago, es decir está conformado por dos módulos; el MCU (Unidad de control de medición) y el CIU (Control de interface de usuario).[6]

[6] (Co-0353-R1 Sánchez Gómez - Medición Centralizada.p.6).

Medidor eléctrico de inducción: es un aparato el cual mide la energía eléctrica integrada en un intervalo de tiempo, a potencia variable. Básicamente está formado por disco de aluminio, el cual su movimiento, es generado por un campo electromagnético que se forma en un par de embobinados (uno es el electroimán voltímetro y. el otro es el electroimán amperimetrico. El disco esta acoplado a una parte mecánica, formada por un engranaje en el cual esta los números que indican el consumo de las cargas eléctricas.

Figura 1 medidor eléctrico de inducción.

Fuente: https://spanish.alibaba.com/product-detail/hot-sell127v-230v-single-phase-front-board-installation-analog-energy-meter-kwh-meter-electric-meter-with-pc-transparent-cover-987107337.html

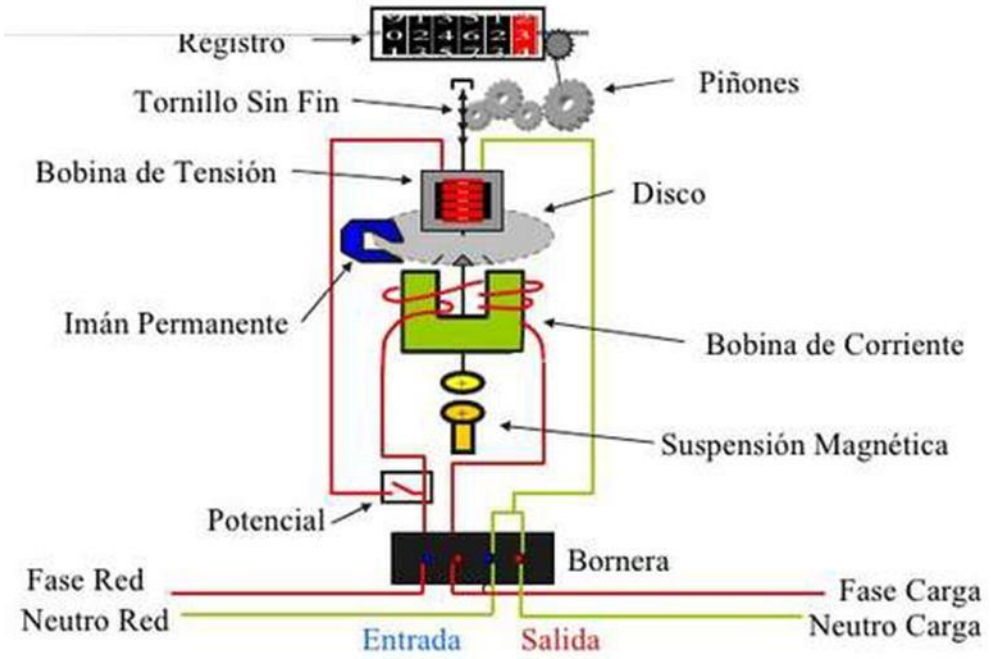


Figura 2. Partes de un medidor de energía por inducción.

Fuente: http://cursosvega.blogspot.com/p/contador-de-energia.html

Medidor eléctrico estático: Al igual que los medidores eléctricos de inducción cumple la misma función, pero la diferencia radica es en la forma en que lo hacen. Los medidores electrónicos tienen dispositivos que reciben el voltaje, y este en la salida del medidor se refleja en pulsos y cuya frecuencia es proporcional a los Vatios-hora.

Estos equipos, generalmente son de mayor precisión que los electromagnéticos, por tal motivo se utilizan en sitios donde se requiere una mayor atención a su registro.

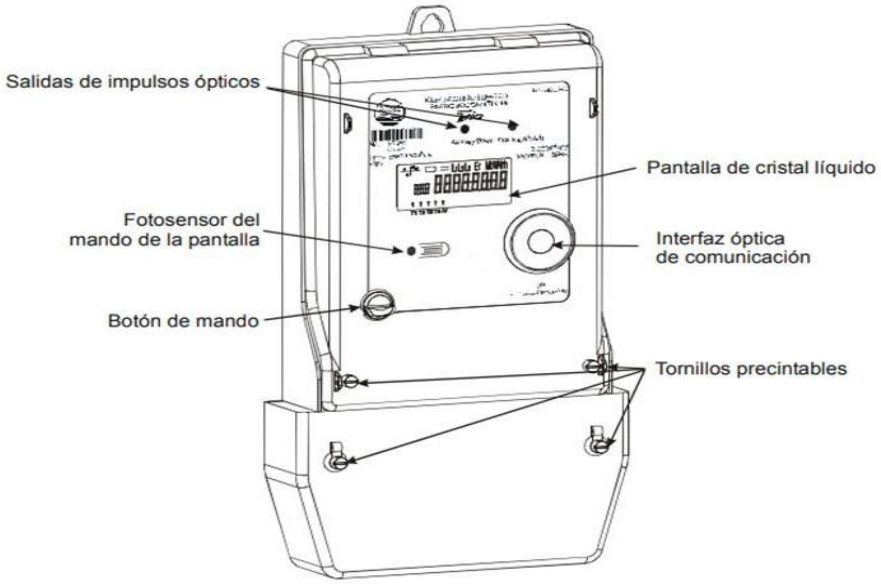


Figura 3. Partes de un medidor eléctrico estático.

Fuente: https://docplayer.es/8904718-Contador-de-energia-electrica-trifasico-estatico-para-energia-activa-y-reactiva-epqs.html

Figura 4. Medidor de energía, tipo estático.

Fuente: imágenes propias

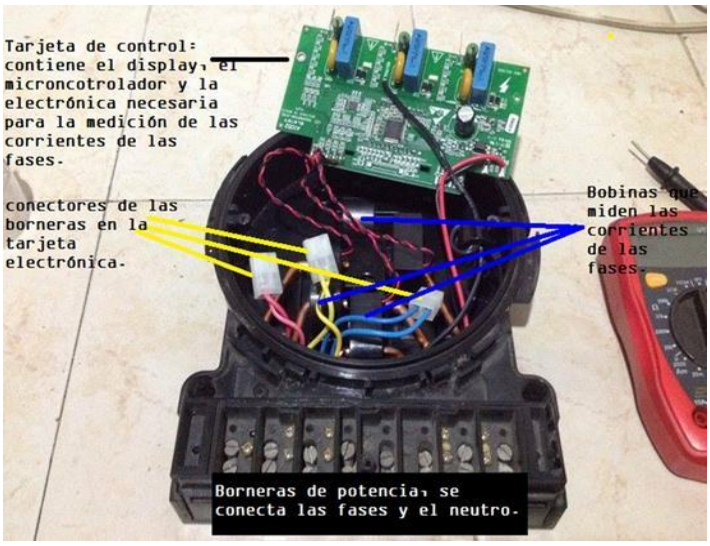


Figura 5. Partes que forman un medidor de energía estático

Fuente: imágenes propias

Medidor monofásico bifilar: Es un aparato de medida para el consumo de energía eléctrica, dicho elemento puede ser de inducción (disco) o estático (electrónico), además por ser monofásico bifilar indica que puede censar una fase.

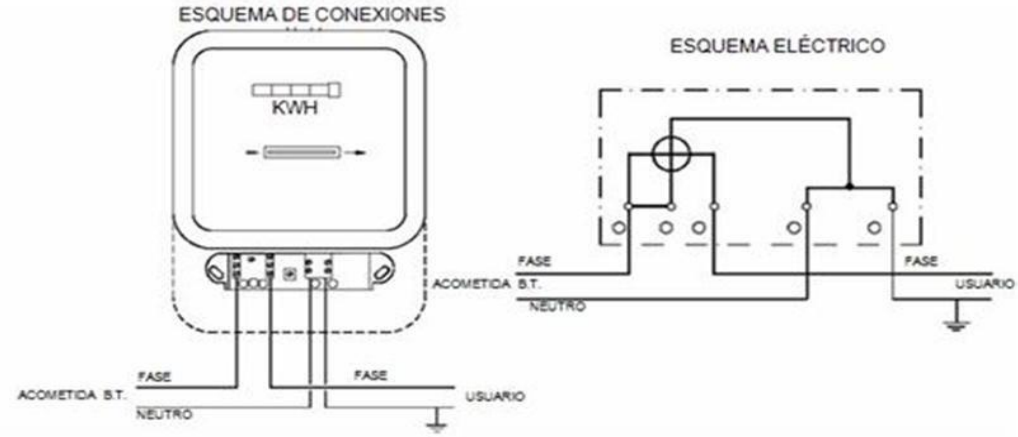


Figura 6. Conexión de un medidor de energía monofásico bifilar

Fuente: https://sites.google.com/site/labmediup/temas-de-la-unidad/pl-7-medicion-de-potencia/7-4-teoria-y-tecnica

Sensor de Corriente AC 100A No invasivo: Es un sensor utilizado para la medición de corriente alterna. Son particularmente útiles para medir el consumo o generación de electricidad de una casa, se basa de en el principio de inducción electromagnética, estos dispositivos tienen la gran ventaja de no afectar la corriente del circuito del cual están "midiendo" esto indica que son totalmente aislados. Técnicamente se puede decir que la cantidad de espiras del transformador interno del sensor, representa la relación entre corriente que circula por el cable y la que el sensor nos entrega, esta relación o proporción es la que diferencia entre los diferentes modelos de sensores SCT-013, adicionalmente pueden tener una resistencia de carga en la salida de esta forma en lugar de corriente se trabaja con una salida voltaje.

Figura 7. Sensor de corriente no invasivo SCT013

Fuente: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html

El sensor de corriente a utilizar en este proyecto se trata de un sensor de referencia SCT013-100A, donde el último número, después del guion, corresponde al amperaje soportado. Características técnicas del sensor SCT013-100A:

Sensor de corriente tipo pinza para medición no invasiva, trasformador de corriente con núcleo de ferrita.

Medición para corriente AC.

Rango de medición de 0 a 100Amp.

Relación de transformación de 2000:1 – (100Amp: 50mA).

Salida analógica de corriente, por lo tanto se requiere de un circuito de conversión de corriente a voltaje.

No linealidad: +/-3%.

Desconexión súbita cuando el trasformador se energiza.

Temperatura de operación: -25 C a + 70 C.

Resistente al fuego: UL94-V0.

Cable blindado de 1 mts de longitud, con plug 3mm.

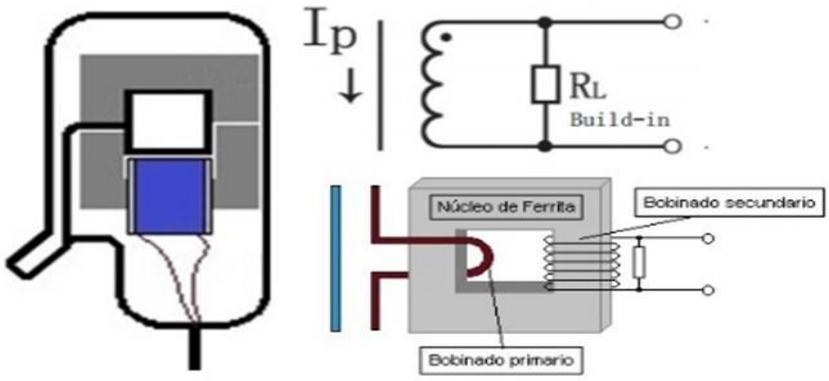


Figura 8. Sensor de corriente y sus partes internas.

Fuente: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html

Circuito Acondicionador: como el sensor de corriente genera una señal alterna (sinusoidal), en el orden de los mili-voltios, es necesario realizar un circuito de

acondicionamiento, el cual se “transforme” la señal alterna de corriente, rectificándola y amplificándola.

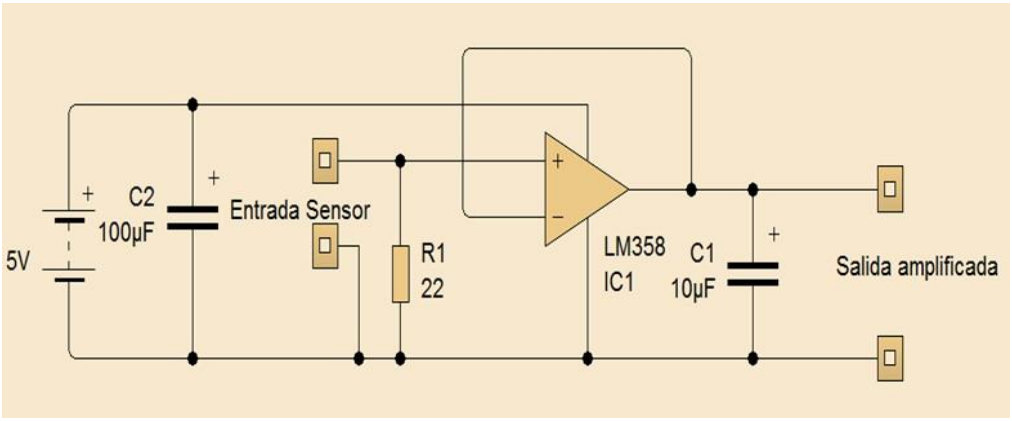


Figura 9. Circuito acondicionador para el sensor SCT013-100A.

Fuente: http://www.diverteka.com/?p=1966

Arduino Mega: es una placa de desarrollo basada en el microcontrolador ATmega2560. Tiene 54 entradas/salidas digitales (de las cuales 15 pueden ser usadas como salidas PWM), 16 entradas analógicas, 4 UARTs, un cristal de 16Mhz, conexión USB, Jack para alimentación DC, conector ICSP, y un botón de reseteo.

Figura 10. Placa de desarrollo Arduino mega 2560.

Fuente: https://www.antratek.com/arduino-mega-2560

Display LCD 2X16: es un periférico muy común, que basa su funcionamiento en el controlador HD44780 de Hitachi. El display es muy utilizado en electrónica para la visualización de caracteres como letras, números y hasta figuras en su pantalla, este tipo de elementos cuenta con varios pines los cuales sirven para la transmisión de los datos desde el microcontrolador hasta el display, además cuenta con leds los cuales dan buena iluminación a dicho elemento.

Figura 11. Display LCD 2X16.

Fuente: https://www.geekfactory.mx/tutoriales/tutoriales-arduino/lcd-16x2-por-i2c-con-arduino/

Teclado Matricial: Es un dispositivo que agrupa varios pulsadores y permite controlarlos empleando un número de conductores inferior al que necesitaríamos al usarlos de forma individual. Los teclados, agrupan los pulsadores en filas y columnas formando una matriz, disposición que da lugar a su nombre. Tal vez como desventaja de usar un teclado matricial es que pueden causar problemas cuando se pulsa más de una tecla simultáneamente.

Figura 12. Teclado matricial 4x4.

Fuente: https://naylampmechatronics.com/interfaz-de-usuario/420-teclado-matricial-4x4-de-botones-plasticos.html

Modulo bluetooth HC-06: El módulo HC-06 de bluetooth lo podemos usar para conectar y comunicar con arduino o cualquier placa de desarrollo por vía bluetooth, permitiendo el control de ciertos elementos vía inalámbrica, además tiene la enorme ventaja de estar integrado de fábrica en la mayoría de dispositivos como portátiles, tablets, y Smartphone. Además, otro punto a su favor es su uso independiente del sistema operativo.

Tiene 4 pines, los cuales dos de ellos son de comunicación, conocidos como TX (transmisión) y RX (recepción) y tiene dos pines de alimentación VCC= 5V y gnd.

La comunicación Bluetooth es similar al uso del puerto serie normal, pero la diferencia radica es que, en lugar de un conectar un cable, tendremos que emparejar el módulo con nuestro dispositivo. El proceso de emparejado depende del sistema operativo pero es, en general, un proceso sencillo.

Figura 13. Modulo Bluetooth – HC06

Fuente: https://naylampmechatronics.com/blog/15_Configuración--del-módulo-bluetooth-HC-06-usa.html

Código QR: Código gráfico de barras bidimensional, o código de respuesta rápida, este se desarrolló en Japón en el año de 1994 por Denso Wave, compañía subsidiaria de Toyota. El código puede contener caracteres alfa numéricos (letras y números) o solo números, los cuales pueden ser el precio de un artículo, el link de una página web, la composición de una receta, etc. El código sea de barras o gráfico, como se dice es un sistema de encriptación de datos que trata de un numero binario o un numero hexadecimal.

Figura 14. Código Qr típico.

Fuente: https://www.unitag.io/es/qrcode/what-is-a-qrcode

Lector de código QR: es un equipo electrónico llamado escáner lector, se trata de un dispositivo óptico, que por lo general utiliza un haz de luz láser esto para iluminar el código, luego por un método reflectivo, el sensor que es un fotodiodo recibe ese haz de luz pero ya con información del código escaneado.

Actualmente existen aplicaciones para el teléfono, esto para convertirlo como lector del código gracias a su cámara que sirve como escáner, en general casi todos los teléfonos inteligentes vienen dotados con un software para tal fin.

Figura 15. Lector código QR.

Fuente: https://www.pccomponentes.com/unotec-lector-de-codigos-de-barras-2d-con-lector-de-qr

Software para lectura de QR: se trata de una plataforma, cuya función es la de convertir y almacenar los códigos que pasan por la pistola lectora laser, este software es ampliamente utilizado en supermercados, empresas de servicio público (gas, agua, energía), en tiendas de ropa, en teléfonos inteligentes etc.

El software de lectura QR se puede dividir en dos partes, esto derivado de su funcionalidad, el primer software es el driver que permite la comunicación entre el dispositivo electrónico (la

pistola lectora) y el PC o la terminal de procesamiento. La otra parte es el software de captura, procesamiento y almacenamiento del código, es decir el aplicativo como tal.

Las plataformas más populares son XRenQRCode, Code Two QR Code Desktop Reader, Beetag Reader, QuickMark Reader, I.nigma Reader.

Figura 16. Software populares lectores QR.

Fuente: https://uqr.me/es/blog/los-11-mejores-lectores-de-codigos-qr-para-android-iphone-windows-phone-y-blackberry/

Contactor: es un aparato eléctrico de mando a distancia, que puede cerrar o abrir circuitos, ya sea en vacío o en carga. Es la pieza clave del automatismo en el motor eléctrico. Su principal aplicación es la de efectuar maniobras de apertura y cierra de circuitos eléctricos relacionados con instalaciones de motores, o de circuitos con grandes corrientes.

Un contactor está formado por una bobina y unos contactos, que pueden estar abiertos o cerrados, y que hacen de interruptores de apertura y cierre de la corriente en el circuito.

La bobina es un electroimán que acciona los contactos cuando le llega corriente, abre los contactos cerrados y cierra los contactos abiertos. De esta forma se dice que el contactor está accionado o "enclavado". Cuando le deja de llegar corriente a la bobina los contactos vuelven a su estado anterior de reposo y el contactor está sin accionar o en reposo.

En resumen el contactor se puede decir que también es un relevador pero de mucha mayor potencia, soporte cargas eléctricas con corrientes mayores.

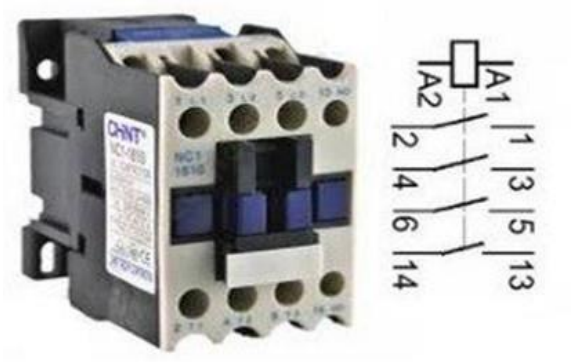


Figura 17. Contactor en su forma física y en símbolo.

Fuente: https://www.areatecnologia.com/electricidad/contactor.html

Como se puede apreciar en el símbolo del contactor se tiene las terminales A1-A2 del embobinado, este embobinado puede trabajar con tensiones de 24v, 110v y 220v, también puede observarse los contactos marcados como 1-2, 3-4, 5-6,etc.

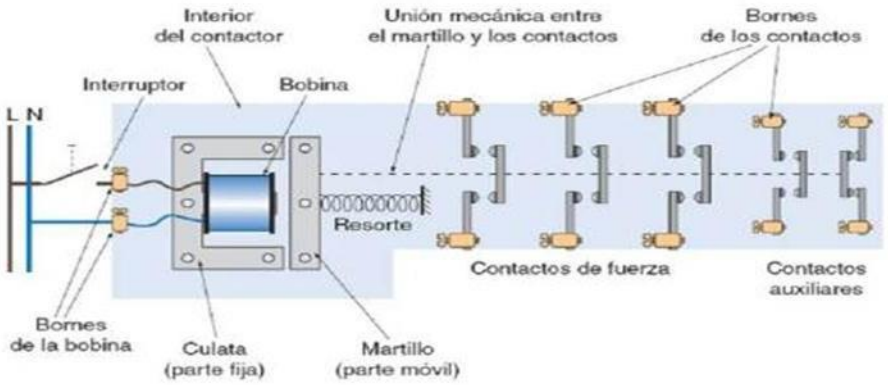


Figura 18. Partes que forman un contactor.

Fuente: https://www.areatecnologia.com/electricidad/contactor.html

Modulo relé: Al igual que el contactor, la función del relé es activar / desactivar cargas eléctricas pero lo que hace la diferencia es que el relevo funciona para cargas de poca corriente y su embobinado se alimenta con tensiones bajas. El relé en su versión más simple es un pequeño electro-imán que cuando lo excitamos mueve la posición de un contacto eléctrico de conectado a desconectado o viceversa.

Figura 19. Modulo relé de 5v

Fuente: https://articulo.mercadolibre.com.co/MCO-453055579-modulo-rele-1-canal-5v-arduino

App inventor: App Inventor es un entorno de desarrollo de software creado por Google Labs para la elaboración de aplicaciones destinadas al sistema operativo Android. El usuario puede, de forma visual y a partir de un conjunto de herramientas básicas, ir enlazando una serie de bloques para crear la aplicación. Con AppInventor, se espera un incremento importante en el número de aplicaciones para Android debido a dos grandes factores: la simplicidad de uso, que facilitará la aparición de un gran número de nuevas aplicaciones; y Google Play, el centro de distribución de aplicaciones para Android donde cualquier usuario puede distribuir sus creaciones libremente. Esta programación por bloques es muy fácil e intuitiva de hacer, además es gratuito y se puede

acceder fácilmente de la web. Las aplicaciones creadas con AppInventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil.

QT – Creator: es un IDE multi plataforma programado en C++, JavaScript y QML creado por Trolltech el cual es parte de SDK para el desarrollo de aplicaciones con Interfaces Gráficas de Usuario, este software puede trabajar divinamente en los sistemas operativos, como Linux 2.6, Mac 10.4 y Windows con la versión 4.4.3

Dentro de las características más relevantes que cuenta QT son las siguientes.

Editor de código con soporte para C++, QML y ECMAscript

Herramientas para la rápida navegación del código

Resaltado de sintaxis y auto-completado de código

Control estático de código y estilo a medida que se escribe

Soporte para refactoring de código

Figura 20. Logotipo QT.

Fuente: https://doc.arcgis.com/es/appstudio/extend-apps/whatisqtcreator.htm

Estado del arte

Los medidores eléctricos surgen con la necesidad de “medir” las cargas eléctricas conectadas a un determinado circuito eléctrico. Debemos tener presente que el medidor eléctrico es un vatímetro, por tal motivo y para conocer el antepasado del medidor eléctrico, debemos primero de conocer la historia dos “medidores” por separado.

El primero instrumento de medida se trata del voltímetro, este permite censar la tensión, voltaje o diferencia de potencial de un determinado circuito, este instrumento fue desarrollado con elementos electromecánicos como lo son bobinas y partes metálicas, este instrumento también se le conoció como galvanómetro, fue desarrollado por Han Oersted, en el año de 1820, Luego, en 1881, con Jacques D’ Arsonval, aparece el instrumento de medición que conocemos en la actualidad, cuya característica principal es originar la deflexión de una aguja cuando a través de él circula una corriente continua, proporcional a la magnitud de la variable que se está midiendo.

Cabe resaltar que el galvanómetro es un instrumento de medida netamente analógico cuenta con un “espejo” el cual tiene impreso una serie de números, los cuales indica la “cantidad” de tensión o de intensidad. También se hace valer que el galvanómetro fue el primer multímetro puesto que funcionaba para voltímetro, colocando las puntas en paralelo con el voltaje a medir, y para medir corriente colocando las puntas en serie con la carga a medir.

Al paso de los años se fue mejorando los instrumentos de medición de magnitudes físicas como lo es la corriente y el voltaje, pero si se quería saber el consumo o potencia que tenía una

carga eléctrica se debía de realizar los respectivos cálculos entre voltaje y corriente, puesto que no había aun un instrumento para medir dicho consumo o potencia.

Se puede decir que el vatímetro, formalmente fue patentado en 1878, por JB Fuller, creando un equipo que podría medir el consumo de energía en el momento de una lámpara funciona con corriente alterna.

Estos equipos fueron trasladados en cuanto su aplicación para la mediciones de consumo eléctrico en residencias e industrias, fueron mejorando su construcción y diseño, implementando medidores más robustos, análogos, los cuales poseían un disco de aluminio y una estructura metálica que actúan como núcleo donde se instala una par de bobinas, una para medir la tensión y la otra bobina medir corriente. Cabe decir que este tipo de medidores se utilizan hoy en día

Pasa el tiempo, En 1972, mientras trabajaba para Boeing en Hunstville, Alabama, Estados Unidos. Ted Paraskevakos desarrolló un sistema digital de monitoreo de incendios, seguridad y sistemas médicos de alarmas. Este también podía leer medidores de servicios públicos. La tecnología se derivó del sistema automático de identificación de líneas telefónicas, conocido hoy en día como identificador de llamadas.

El Sr. Paraskevakos obtuvo una patente estadounidense para esta tecnología en 1974. Tres años más tarde, creó Metretek, Inc., la empresa que produjo el primer medidor remoto completamente automatizado comercialmente disponible y el sistema de gestión de cargas y lecturas.

A este tipo de aparatos se les conoce como medidores inteligentes, los medidores de electricidad tradicionales sólo medían el consumo total y no proporcionaban mayor información. Los medidores inteligentes pueden incluso medir sobretensiones, lo cual permite un diagnóstico de problemas con la calidad de la energía.

Hablando a nivel nacional, a nivel país, los medidores de energía se reglamentan en la norma NTC2050 – (normatividad técnica colombiana) y el RETIE, en específico los medidores en Colombia debe cumplir, teniendo en cuenta la clase (se utilizan medidores clases: 0,2, 0,2s, 0,5, 0,5s, 1 y 2. Siendo de mayor exactitud el medidor clase 0,2s.) Y según el caso, las siguientes normas:

NTC 2288 "Equipos de medición de energía eléctrica -C.A.-. Requisitos particulares. Medidores electromecánicos de energía activa -clases 0,5, 1 y 2-.", basada en la norma IEC 62053-11.[7]

NTC 2147 "Medidores Estáticos de Energía Activa. Especificaciones Metrológicos para clase 0.2S y 0.5S", basada en la norma IEC 62053-22.[8]

NTC 4052 "Medidores Estáticos de Energía Activa para corriente alterna clase 1 y 2", basada en la norma IEC 62053-21.[9]

NTC 4569 "Equipos de medición de energía eléctrica –C.A.-. Requisitos particulares. Medidores estáticos de energía reactiva (Clases 2 y 3)", basada en la norma IEC 62053-23.[10]

[7] (Normatividad Técnica para la instalación de medidores , requisitos)
[8] (Normatividad Técnica, Equipos de medición de energía eléctrica)
[9] (Normatividad Técnica, medidores clase 1 y 2)
[10] (Normatividad Técnica, medidores clase 2 y 3)

El programa de Energía Prepago en Colombia, dio sus primeros pasos en febrero de 2005, cuando EPM (empresas públicas de Medellín), inició un proyecto piloto con 92 familias de bajos recursos económicos de Medellín. Con ellos la empresa evaluó el grado de aceptación, hábitos de consumo y comportamiento frente a esta opción, que desde su nacimiento buscó incentivar la conciencia del consumo responsable por parte de los usuarios.

Tras una amplia aceptación de las familias que tomaron parte del piloto, la Junta Directiva de EPM aprobó la masificación de la opción de Energía Prepago, dando vía libre a la adquisición de medidores residenciales que permitieron implementarla a partir de julio de 2007.

Esta empresa (EPM), fue la pionera en Colombia, en el uso de este sistema de medición de consumo de energía eléctrica, implementado en hogares, en específico en barrios humildes. Otra zona en Colombia es Bucaramanga es la segunda ciudad que tiene a su disposición un sistema de energía prepago. La Electrificadora de Santander, ESSA, empezó a implementar el proyecto en una de las zonas más humildes de la ciudad[11]

[11] (epm, energía prepago, 10 años contribuyendo a la calidad de vida de las familias antioqueñas, 28 de julio 2017)

Objetivos

Objetivo Principal

Implementar un sistema de medición electrónica de recarga prepago para el servicio de energíaeléctrica, esto para llevar el servicio a familias de escaso recursos económicos

Objetivos Específicos

Realizar un circuito electrónico que tenga comunicación con el Smartphone y quecontrole una etapa de potencia. (Se incluye la etapa de acondicionamiento de señal y de control por teclado).

Realizar una etapa de potencia, la cual tenga la capacidad de activar/desactivar todos los circuitos de una casa de mediano tamaño. (Se incluye la etapa de transferenciaeléctrica y fuente de poder).

Realizar un aplicación que lea códigos QR, dichos códigos contienen el valor de larecarga, realizada por el usuario.

Realizar una plataforma donde se genere la factura y los respectivos códigos de recarga.

Cronograma

Tabla 1. Cronograma de actividades.

CRONOGRAMA DE ACTIVIDADES												
ACTIVIDAD	**MES 1**	**MES 2**	**MES 3**	**MES 4**	**MES 5**	**MES 6**	**MES 7**	**MES 8**	**MES 9**	**MES 10**	**MES 11**	**MES 12**
Investigar y recolectar información sobre el funcionamientos de medidores eléctricos -		X	X									
Investigar y recolectar información sobre cómo funciona un código QR				X	X							
Tratar de diseñar códigos QR (hacer pruebas)						X	X					
Hacer pruebas y simulaciones de circuitos de control y de potencia								X				
Realizar el prototipo									X	X		
Presentar proyecto, funcionando											X	X

Recursos

Tabla 2. Presupuesto del prototipo.

RECURSO	DESCRIPCIÓN	PRESUPUESTO
Equipo Humano	Apoyo en asesorías de docente en la UNAD	0
Equipos y Software	Computador – multímetro – memorias USB - Desarrolladores de software y simuladores electrónicos.	1’800.000
Viajes y Salidas de Campo	Visita al campus de la UNAD Dosquebradas	200.000
Materiales y suministros	Dispositivos electrónicos (modulo Bluetooth – microcontroladores – componentes en general), dispositivos eléctricos (contactor-relés – medidores eléctricos. Etc.). Accesorios eléctricos – cajas para chasis.	800.000
Bibliografía	Resolución, normas, decretos, proveedores.	0
TOTAL 2’800.000		

Diseño de la solución

La unidad de energía en la que las empresas prestadoras del servicio eléctrico nos facturan es el kW/h. Esto equivale a la energía consumida por un electrodoméstico u otro aparato cuya potencia fuese un kilovatio y estuviese funcionando durante una hora.

Esto indica que un aparato que consume 1.000 vatios es decir 1 kilovatio. En este caso si queremos mantener encendido este aparato durante una hora, tendríamos un consumo de 1 kW/h, otro ejemplo seria, una máquina que consume 2.000 vatios (2 kW) durante media hora con un 1 kW/h, esto en otras palabras indica que el KW/h no dice el tiempo en que se consume esa energía, sino más bien la cantidad de energía.

A continuación se expone de manera analítica este concepto.

$$E = P_{Vatios} * T_{Hora}$$

Como ejemplo se tiene una bombilla de 100W (vatios), y esta es encendida durante 10 horas.

$$E = 100W * 10H \rightarrow E = 1000\,{}^{W}/_{h} \rightarrow 1kw/h$$

Como unidad fundamental de medida de la potencia eléctrica es el vatio (Watt), que se traduce como el producto entre el voltaje y la corriente eléctrica.

$$W = V_{Voltios} * I_{Amperios}$$

Para efectos prácticos del proyecto, y por tratarse de un prototipo, se tiene tres cargas eléctricas (bombilla eléctrica) de 100w, ya conociendo la potencia, o consumo se realiza, el respectivo cálculo de la corriente máxima eficaz.

$$P = V * I_{RMS}$$

$$100W = 120V * I_{RMS}$$

$$\frac{100W}{120V} = I_{RMS} \rightarrow 0.833\ Amperios$$

Ya calculando la corriente máxima eficaz, se calcula el valor pico de corriente, aplicando la siguiente ecuación.

$$I_{RMS} = \frac{I_{Pico}}{\sqrt{2}}$$

$$I_{RMS} * \sqrt{2} = I_{Pico}$$

$$0.833\ A * \sqrt{2} = I_{Pico} = 1.178\ Amp$$

Para calcular la corriente de pico del devanado secundario del sensor se aplica la siguiente ecuación.

$$\frac{N_P}{N_S} = \frac{I_S}{I_P} \rightarrow I_S = \frac{N_P * I_P}{N_S}$$

Donde $N_P =$ numero de espiras del primario (es la acometida del tablero de distribución)

$N_P =$ Numero de espiras del secundario del sensor de corriente no invasivo *SCT013-110Amp.* (Según el Datasheet tiene 2000 espiras).

$I_P =$ 1.178 Amp.

Calculando I_S

$$I_S = \frac{N_P * I_P}{N_S} = \frac{1 * 1.178\ Amp}{2000}$$

$$I_S = \frac{1 * 1.178\ Amp}{2000} = 0{,}000589\ Amp$$

Si el medidor se fuera a implementar en un hogar, se debe tener en cuenta todas las cargas eléctricas que estén conectadas en el tablero de distribución, a continuación se muestra un cuadro donde se exponen dichas cargas, su corriente, voltaje y potencia eléctrica.

Tabla 3. Cuadro de cargas eléctricas, consumo de potencia y corriente.

Elemento	**Voltaje (voltios AC)**	**Corriente (Amperios AC)**	**Potencia (Vatios AC)**
Nevera no frost	120V AC	4.16 Amp	500 W
Lavadora	120V AC	18.33 Amp	2200W
Plancha de ropa	120V AC	14.16 Amp	1700 W
Ducha eléctrica	120V AC	58.33 Amp	7000W
Televisor LED	120V AC	0.54 Amp	65W
Computador	120V AC	3.33 Amp	400 W

de escritorio			
Equipo de sonido	120V AC	1.66 Amp	200W
Iluminación	120V AC	0.75 Amp	90W
Microondas	120V AC	12.5 Amp	1500W
TOTAL	120V AC	113.76 Amp	13655W

Con los datos antes obtenidos, se debe tener en cuenta el comando AREF que tiene el arduino, esto como voltaje de referencia para censar voltajes de los sensores, que para este caso es el de corriente no invasivo *SCT013-110Amp.*

Para la creación del prototipo del medidor de energía eléctrica prepago, se tiene en cuenta que este consta de dos partes, la física (hardware), y la lógica (software).

Para la parte física (el medidor) se implementa con la tarjeta de desarrollo ARDUINO MEGA 2560.

En la figura 21, se muestra los diferentes elementos que se incluyen dentro de la caja de plástico, los cuales forman prototipo del medidor prepago.

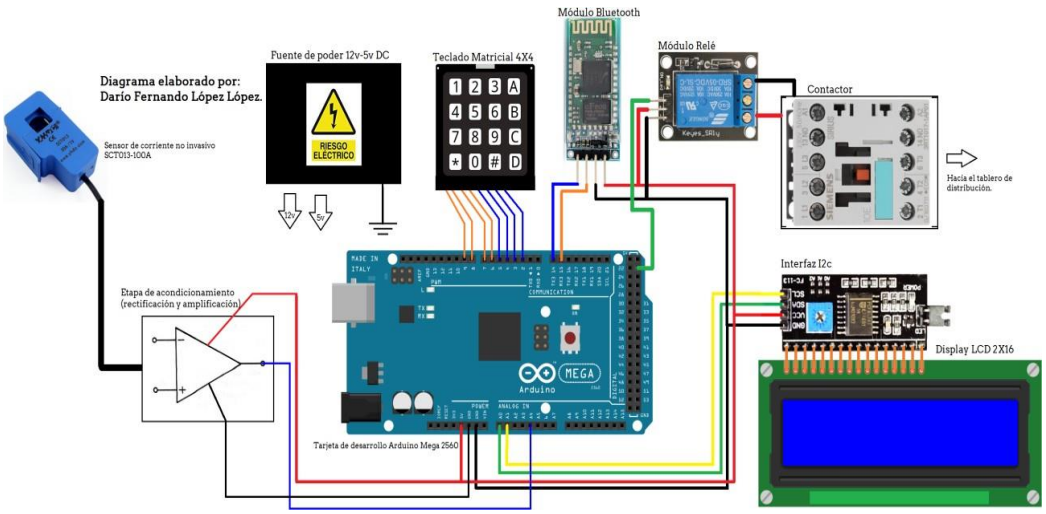


Figura 21. Componentes que forman el prototipo del medidor prepago (físico).

Fuente: propia

A continuación se muestra el diagrama a bloques del medidor prepago.

Figura 22. Diagrama a bloques del prototipo.

Fuente: propia.

La primer parte (etapa 1) que se desarrolla, es el acondicionamiento de la señal, la cual, es captada por el sensor no invasivo de corriente *SCT013-110A*, el objetivo de este sensor, en

esencia es la de “detectar” la corriente eléctrica que pasa por el cable “vivo” o fase de la acometida.

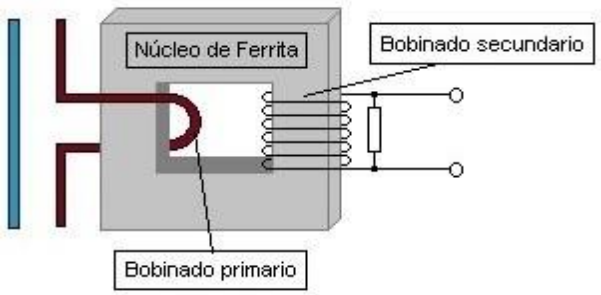


Figura 23. Elementos que forman el sensor de corriente no invasivo.

Fuente: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html

Hay que destacar que, con este sensor podemos medir una corriente hasta de 100A, teniendo como salida 50mA para una corriente de 100A, la proporción entonces es de 100A/50mA.

Se elige dicho sensor porque tiene las siguientes características, las cuales son relevantes para el buen funcionamiento del proyecto.

Rango de medición de 0 a 100Amp.

Medición para corriente AC.

Salida analógica de corriente.

Temperatura de operación: -25 C a + 70 C.

Sensor de corriente tipo pinza para medición no invasiva, trasformador de corriente con núcleo de ferrita.

Resistente al fuego: UL94-V0.

Relación de transformación de 2000:1 – (100Amp: 50mA).

El sensor en definitiva se trata de un transformador, el cual tiene núcleo de ferrita, se puede decir que el embobinado primario es el cable de la fase de la acometida, por su parte el embobinado secundario está fabricado con espiras de cobre de un menor calibre, dicho embobinado esta enrollado sobre el mencionado núcleo.

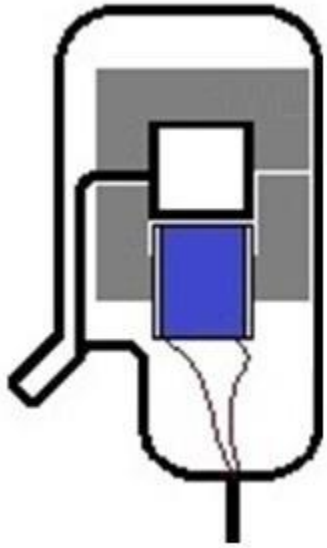


Figura 24. Sensor de corriente no invasivo.

Fuente: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html

El circuito implementado para el acondicionamiento de la señal (corriente eléctrica), se desarrolla con el amplificador operacional LM358, a continuación se muestra el diagrama esquemático.

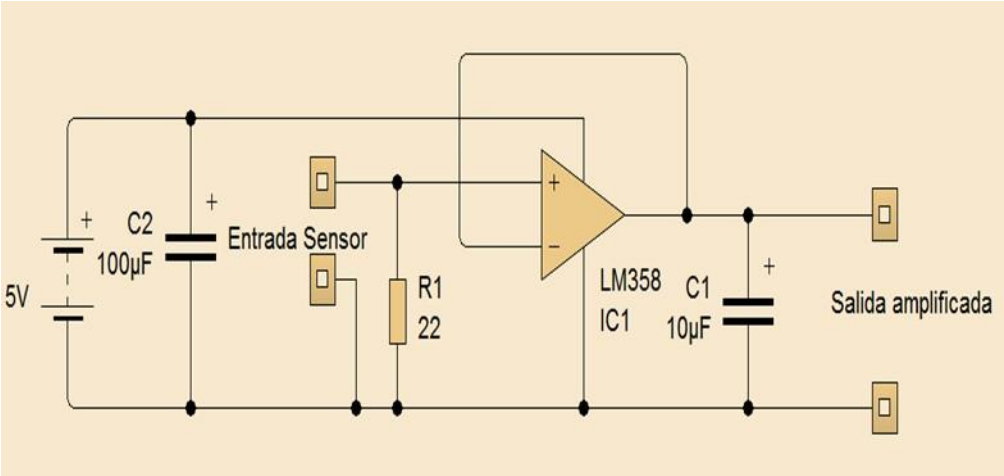


Figura 25. circuito implementado como acondicionador de señal para sensor SCT013-110A

Fuente: propia

Funcionamiento del circuito.

Cuando la señal (la fase) es captada por el sensor, este trasmite la corriente por el núcleo que es de ferrita, luego pasa por campo electromagnético llegando al embobinado secundario. La señal que ingresa se trata de una corriente eléctrica que está en el orden de miliamperios, como se trata de un vatímetro (se debe medir voltaje y corriente), se coloca en paralelo a la terminales del sensor de corriente (salida del embobinado secundario) un resistor de 22 Ohm, esto para convertir la señal de corriente en voltaje (por ley de ohm), además porque el registro de los valores deben estar alrededor de 1.1V (50mmA X 22 ohm).

El circuito acondicionador de la señal se trata de un amplificador operacional configurado como seguidor unitario. Se optó por realizar el circuito de acondicionamiento con el circuito integrado con matricula comercial LM358, esto debido a las siguientes características, la alimentación del circuito integrado es de 5 voltios de DC.

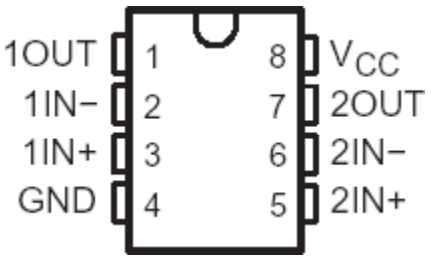


Figura 26. Distribución de pines del LM358

Fuente: http://rogerbit.com/wprb/2018/08/medidor-de-corriente-frecuencia-y-potencia-aparente-de-red-no-invasivo/

Contiene dos amplificadores operacionales

No requiere fuente dual

Voltaje de alimentación: 3 V a 32 V fuente sencilla (±1.5 V a ±16 V fuente dual)

Bajo consumo de potencia

Ancho de banda típico: 0.7 MHz

Compensado en frecuencia internamente

Alta ganancia

Compatible con todas las formas de lógica

Encapsulado DIP 8 pines

A continuación se expone la conexión del circuito de acondicionamiento con la tarjeta de desarrollo Arduino MEGA 2560 (figura 27).

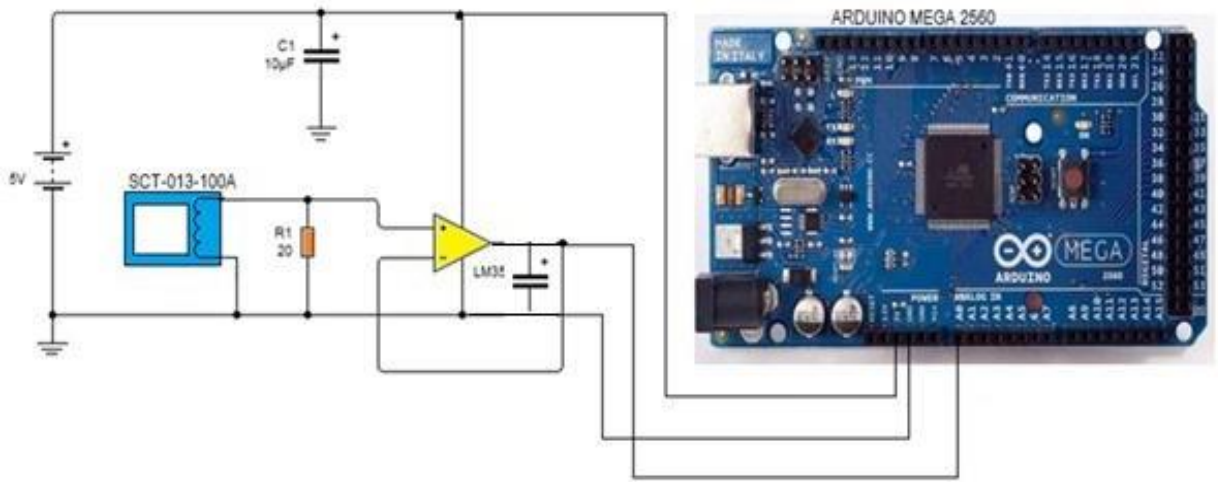


Figura 27. Conexiones del circuito de acondicionamiento con la tarjeta de desarrollo.

Fuente: propia

Como se observa en la figura 28, el circuito de acondicionamiento se implementa en una baquelita de tipo universal, esto lo hace muy práctico, económico y estético. La señal ya acondicionada es pasada a la tarjeta de desarrollo arduino MEGA2560, se ingresa por el pin análogo A0.

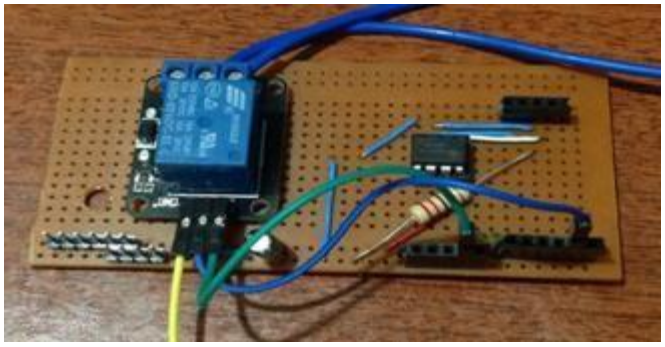


Figura 28. Circuito de acondicionamiento.

Fuente: propia.

A continuación se muestra el diagrama esquemático del arduino MEGA 2560 con los demás componentes que forman el medidor de energía eléctrica, tales elementos son el módulo relé, el teclado matricial 4X4, el módulo I2C, el display LCD 2X16 y el módulo bluetooth HC06, (figura 29).

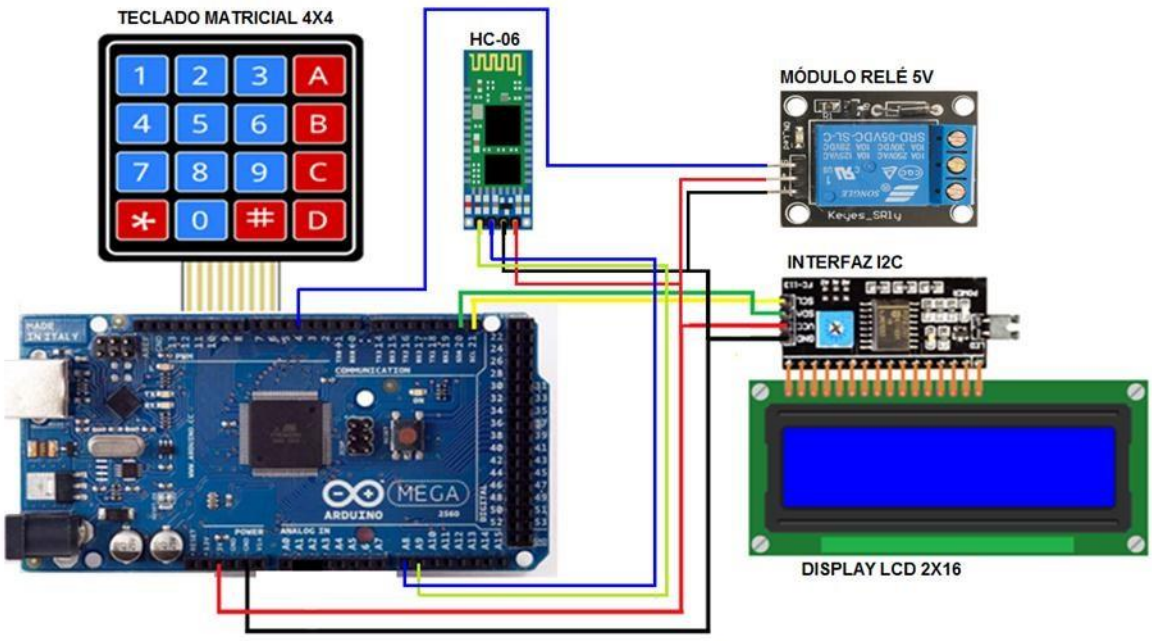


Figura 29. Diagrama esquemático, medidor de energía prepago.

Fuente: propio

Para entender mejor las funciones de los elementos de entrada y salida del medidor prepago, se presenta a continuación un diagrama a bloques.

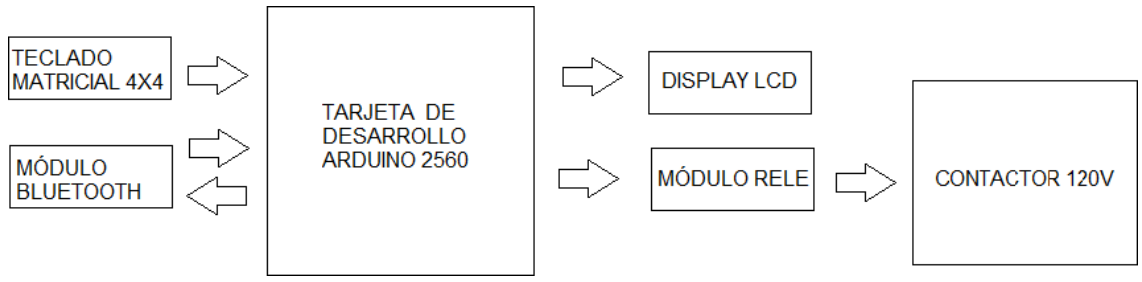


Figura 30. Diagrama a bloques elementos de entrada y salida del medidor prepago.

Fuente: propia.

El medidor cuenta con un teclado matricial 4X4 este se trata de una matriz de botones las cuales están conectados entre filas y columnas, y permiten que el usuario ingrese el valor alfanumérico del código generado por el programa que tiene la empresa de energía, el cual se describirá más adelante.

El propósito del relevador, es el de activar / desactivar el contactor, cada vez que el usuario recargue el medidor prepago, el programa interno del arduino permite esta acción y en definitivas hace que las cargas eléctricas conectadas al contactor se enciendan o se apaguen. Por su parte el modulo bluetooth permite hacer la misma función que la del teclado, solo que de manera inalámbrica, esto es para que el usuario desde el aplicativo que se le instale al teléfono Smartphone (programa el cual será explicado más adelante), pueda ingresar ya sea por el lector del código QR, o por teclado, pueda activar el medidor prepago y prender también las cargas eléctricas.

También el proyecto cuenta con un display LCD 2X16 este solo permite visualizar que el usuario este informado de la activación del equipo, sepa el valor de la recarga, el consumo de las cargas eléctricas y los valores numéricos de las recargas.

En la figura 31 se expone el diagrama esquemático de la etapa de potencia, se tiene tanto el modulo relé de 5v, formado por un transistor bipolar de tipo NPN actuando como interruptor electrónico, su funcionamiento básicamente es, cuando se ingresa un voltaje de control en la base del transistor este, voltaje está limitado por un resistor, el transistor deja pasar la corriente desde emisor a colector, en este último pin se conecta la bobina del relevo. Por su parte los contactos del relevo permiten controlar la bobina del contactor, esto debido a que dicha bobina se energiza con una tensión de 110v, cabe decir que el pin digital 4 del arduino MEGA 2560, está configurado como salida y está conectado al pin de señal del módulo de relevo.

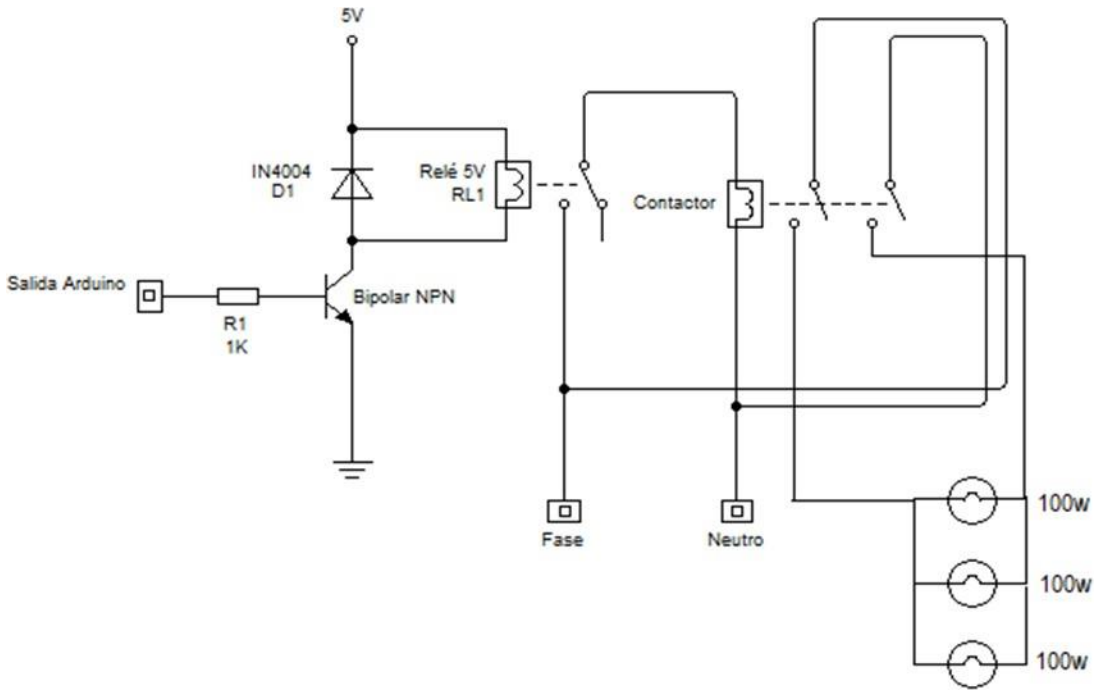


Figura 31. Circuito de potencia del medidor prepago.

Fuente: propia.

La fuente implementada en el proyecto, es una fuente de 12 voltios a 1 Amperio, conocida como fuente conmutada, tiene la función de alimentar los circuitos del sistema. El proyecto también cuenta con un circuito de transferencia, esta para que el sistema pueda seguir alimentado en caso de un fallo en la red eléctrica, es decir el proyecto cuenta con baterías que respaldo que proporcionan la misma tensión de la fuente, a continuación se muestra las respectivas figuras de los elementos mencionados. Figura 32 y figura 33.

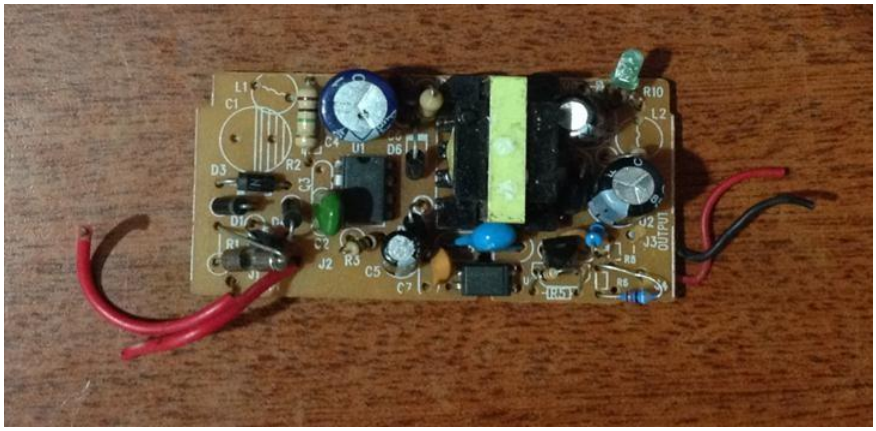


Figura 32. Fuente de alimentación.

Fuente: propia.

Figura 33. Circuito de transferencia.

Fuente: propia.

Para la activación de las cargas eléctricas, que para el caso del proyecto y para fines prácticos se implementa un contactor con embobinado de 110v con capacidad de controlar cargas con corrientes de 12 Amperios, a continuación las características del elemento (figura 34).

Marca: Chint.

Referencia: NC1-1201-110V

Voltaje: 110v

Corriente: 12Amp

Frecuencia: 60Hz.

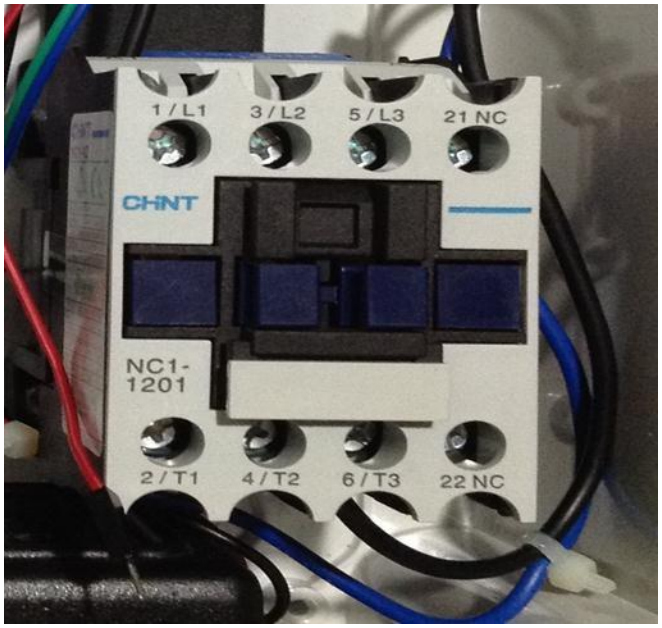


Figura 34. Contactor implementado en el proyecto.

Fuente: propia.

En esta parte del proyecto se muestra la parte de software, esta parte se divide en dos, y es porque se crea el código para el aplicativo del Smartphone (lector del código Qr) y el

aplicativo de la interfaz de usuario (para él operario de la empresa), esto para crear las recargas en el medidor, y poder activarlo. El medio físico de la activación del medidor es el cable USB que viene desde la placa de arduino hasta el PC del operario de la empresa de energía.

Para la elaboración del aplicativo que “lea” el código Qr se implementó el programa de desarrollo llamado App Inventor, se quiso utilizar este programa ya que es muy práctico, funcional, estable y con interfaz entendible, (figura 35).

A continuación se muestra los bloques del aplicativo, lector del código QR.

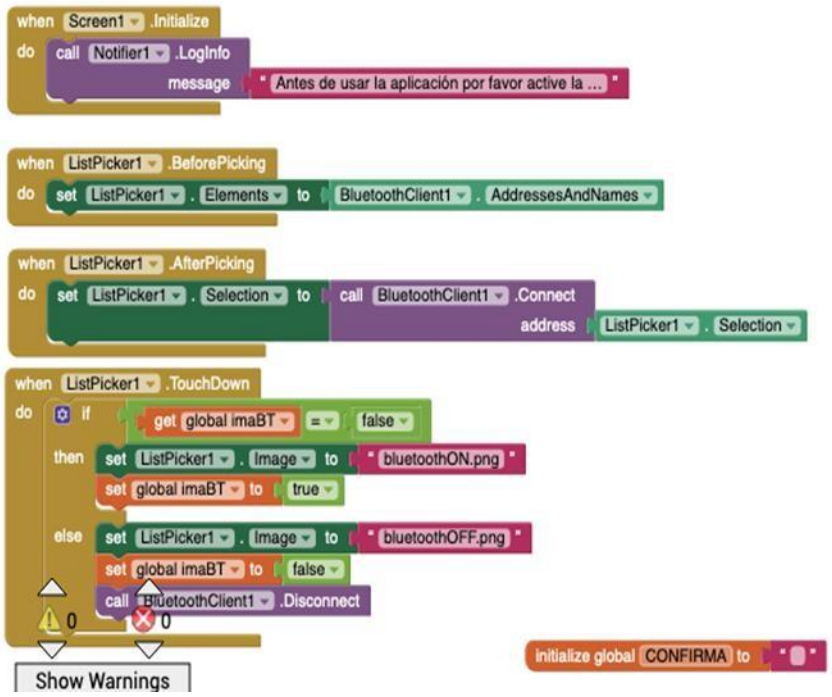


Figura 35. Bloques del aplicativo para el Smartphone.

Fuente: propia.

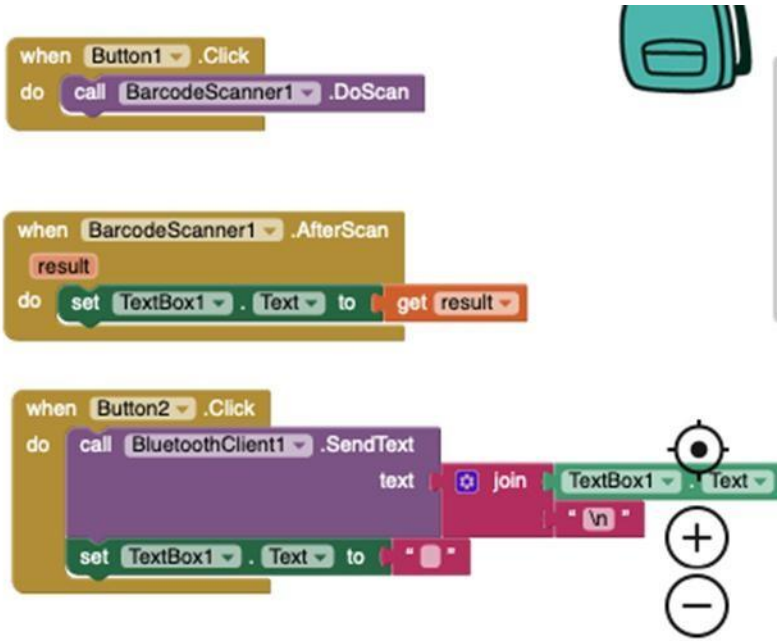


Figura 36. Bloques del aplicativo para el Smartphone.

Fuente: propia.

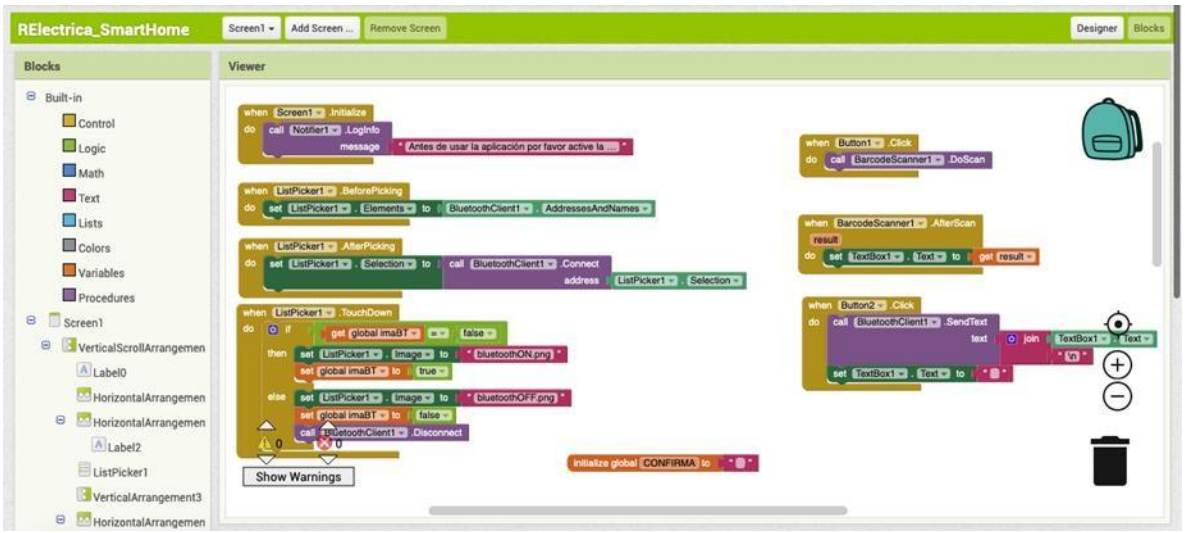


Figura 37. Pantallazo general de los bloques del aplicativo para el lector de código QR, para el Smartphone.

Fuente: propia.

En los bloques utilizados para el aplicativo del lector del código Qr se tiene bloques para la activación en primer lugar del bluetooth esto para la comunicación desde el teléfono hacia el módulo HC-06 que está incorporado en el medidor de energía, (figura 36).

Una vez habilitado el aplicativo puede ser utilizado tanto en modo de texto, puesto que se le agrego una caja de texto para que el usuario pueda ingresar el código en forma alfanumérica o también utilizando la cámara con el aplicativo BardCore Scanner, para la captura del código Qr, (figura 37). También se utilizaron elementos como botones, esto para habilitar las opciones que tiene el aplicativo, se tiene listPicker, elementos de ciclos para ejecutar las acciones hechas por el usuario, (figura 38).

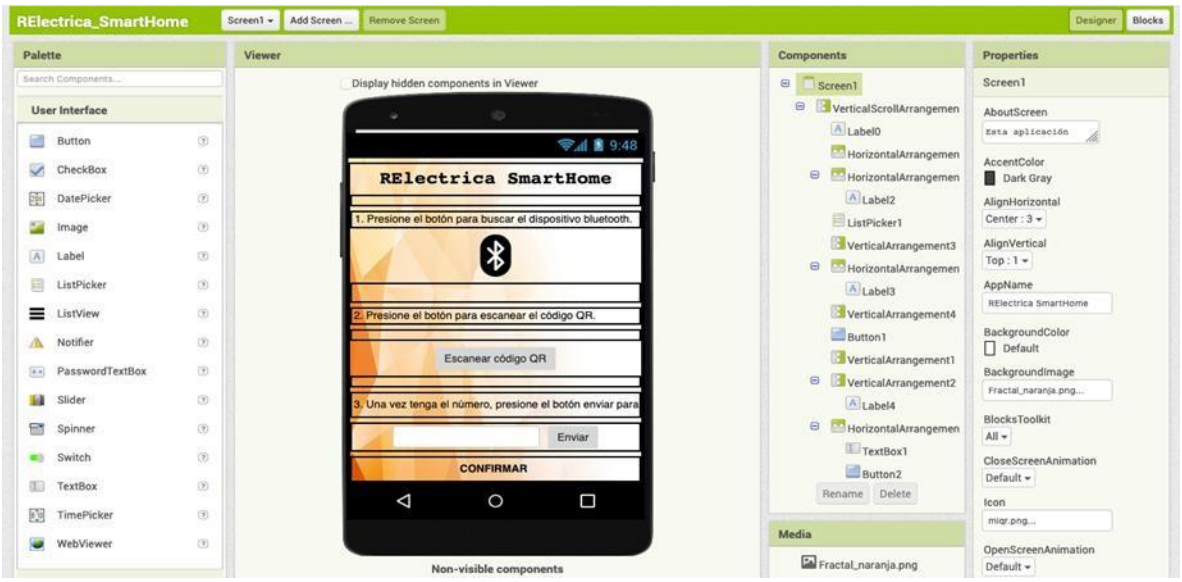


Figura 38. Pantallazo del aspecto grafico del aplicativo.

Fuente: propia

A continuación se muestra el aplicativo realizado en el QT Creator, esto para la generación de los códigos Qr y la elaboración de la factura, (figura 39).

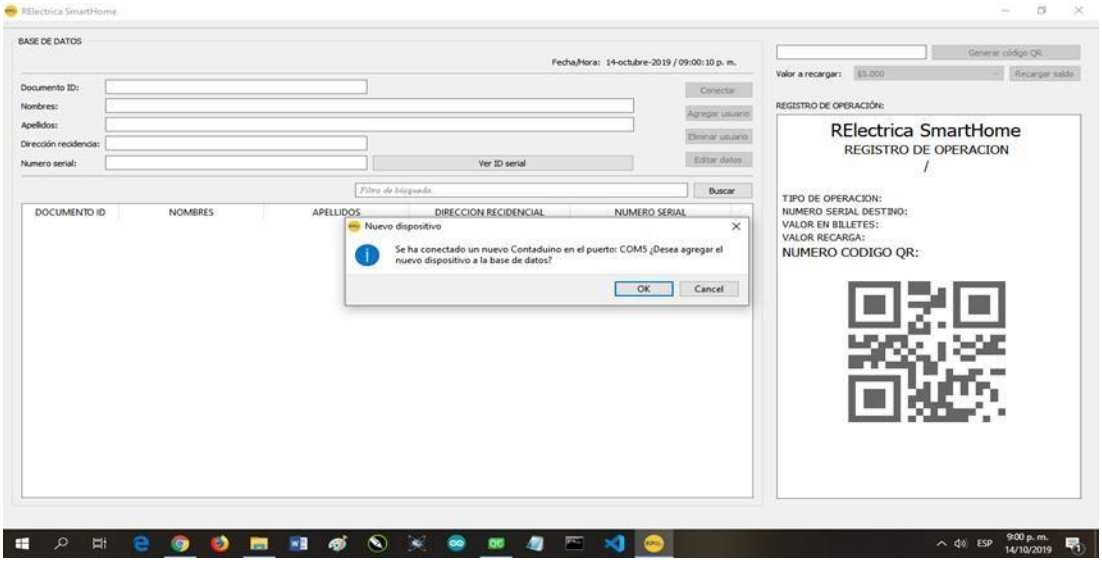


Figura 39. Entorno grafico del aplicativo para la elaboración de la empresa.

Fuente: propia.

Este programa es básicamente una mini base de datos, se crea en base a métodos, los cuales están emparejados con arreglos, estos me permiten guardar los datos más relevantes del usuario, como por ejemplo número de documento, nombre, apellido, dirección, y número de identificación del medidor.

Este último atributo es el más relevante puesto, que con este serial del medidor la empresa puede asociar los valores de recarga con el dispositivo, además este último asocia los demás datos del usuario, generando el código tanto en forma de texto como en el código Qr. También se desarrolló ventanas emergentes esto para dar aviso al usuario de las acciones que se esté ejecutando. Se incorporó además botones para que el programa genere los códigos, también para la habilitación de funciones como agregar nuevo usuario, eliminar usuario, conexión con el medidor, recargue de saldos etc. Cada vez que se genera un código se guarda en una carpeta en formato pdf.

Análisis de los resultados

En esta parte del trabajo, se muestra ya físicamente la implementación del proyecto, su circuitería, y su parte de software.

El proyecto se elaboró en una caja plástica, conocida comercialmente como caja de derivación dexson. En esta caja se integra la tarjeta de desarrollo, el circuito de acondicionamiento para el sensor, la fuente de alimentación de 12v de corriente continua, la etapa de potencia (contactor y relé de activación), circuito de transferencia. Además en la tapa de dicha caja se acondiciona un display LCD 2*16 y el teclado matricial 4*4.

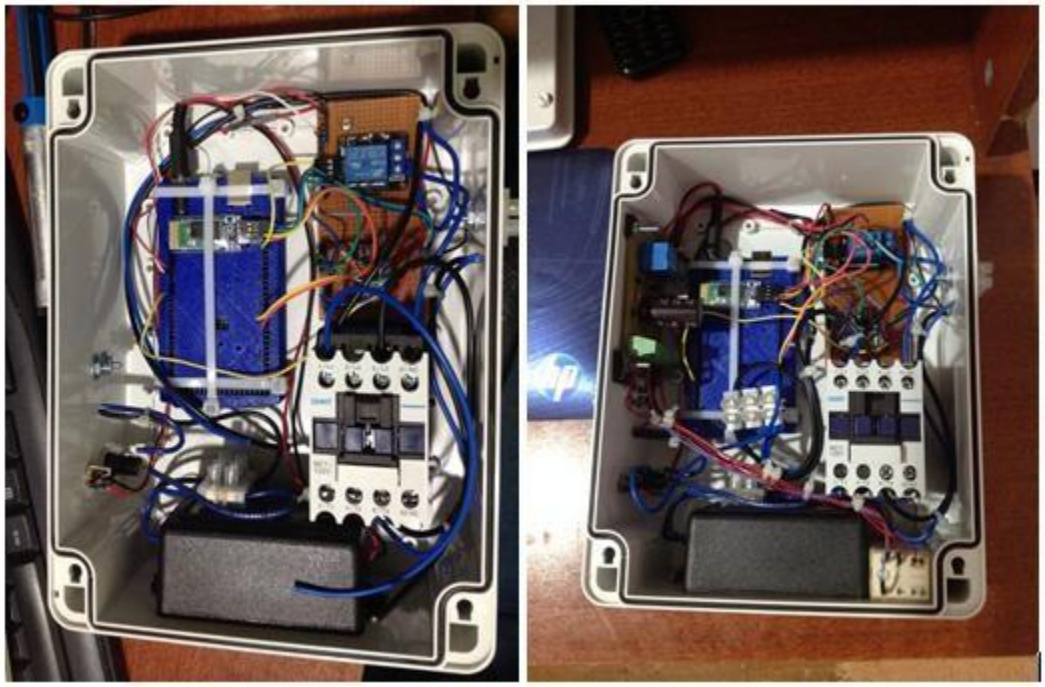


Figura 40. Ensamble del medidor de energía prepago.

Fuente: propia.

Para las conexiones de las acometidas tanto de entrada como de salida se utilizó cable AWG # 14 y terminales de conexión a tornillo, esto para ser más fácil la instalación del equipo. Además se utilizó una porta fusible, y un fusible de 3 amperios como elemento de protección, para la conexión del sensor se utilizó un Jack hembra de 3mm. Figura 41.

Figura 41. Conexiones de entrada y salida del medidor.

Fuente: propia.

Como carga eléctrica a censar y para fines prácticos se utilizaron tres lámparas incandescentes con potencia de 100 W (cada una).

El aspecto final del proyecto es el siguiente.

Figura 42. Aspecto final del medidor de energía eléctrica prepago.
Fuente: propia.

La conexión de las lámparas está en paralelo y están conectadas en las borneras de tornillo que salen del contactor. También se le agrego un vatímetro esto solo como elemento de comparación entre el display del medidor este último, figura 42.

Link del video donde se explica el funcionamiento del medidor de energía eléctrica prepago
https://drive.google.com/file/d/1LFs0W7A9OX9o5K7e_54SrrBXLaX_XDt9/view?usp=sharing

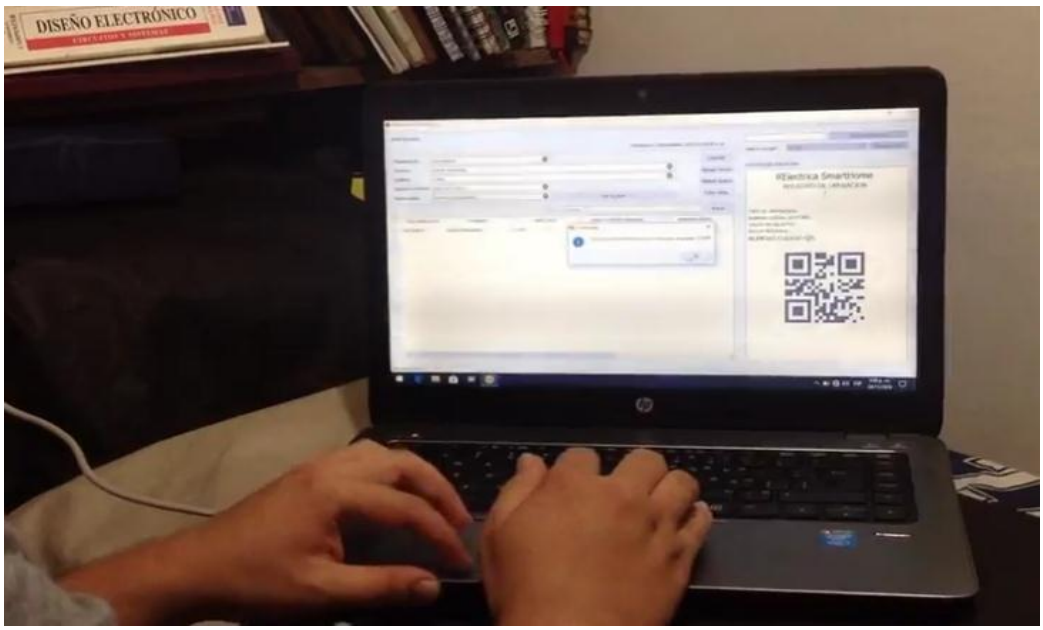


Figura 43. Ejecución de la interfaz de operario de la empresa de energía.

Fuente: Propia

Como se puede apreciar en la figura 43, se ejecuta el aplicativo, en el cual se genera el código QR y/o el código alfanumérico para que el usuario pueda cancelar el valor de la recarga y luego pueda disfrutar del servicio de energía.

Hay que decir que para ejecutar el programa generador de códigos QR, se utilizó una computadora portátil HP, con sistema operativo Windows 10.

El funcionamiento del aplicativo que tiene el trabajador de la empresa de energía es el siguiente:

El programa está definido para hacer recargar con valores que van desde los 5000 pesos el cual es el valor mínimo, hasta los 50.000 pesos, el cual es el valor máximo. Cuando el usuario llega a la empresa y solicita la recarga, el trabajador de la empresa, le pide datos como el nombre y apellidos del suscrito, cedula del suscrito, dirección donde está instalado el medidor, y serial del medidor, y por su puesto el valor de la recarga.

Cabe decir que el medidor tiene un serial, este sería como el código NIU (Número de Identificación Único), figura 44.

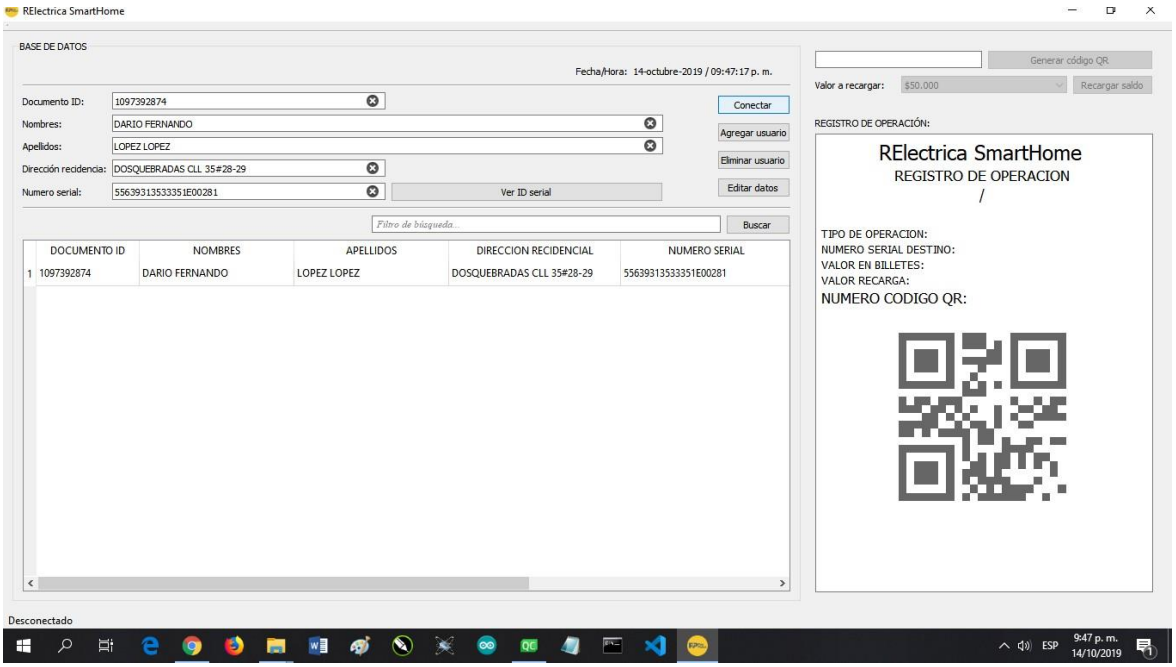


Figura 44. Interfaz de usuario donde se anexan los datos del usuario.

Fuente: propia.

Cuando el usuario ya tiene el código QR, tiene la opción de ingresar el código alfanumérico, esto por medio del teclado matricial instalado en el medidor, (figura 45). Cuando ya se ingresa el código el medidor identifica y reconoce el código ingresado, luego da un mensaje de activación y activa la parte de potencia, energizando las cargas conectas en el medidor, (figura 46).

Figura 45. Manejo del medidor de energía prepago por medio de teclado.

Fuente: Propia

Figura 46. Medidor en funcionamiento.

Fuente: Propia

Conclusiones

Dentro de las conclusiones se puede mencionar que se desarrolló por completo los objetivos planteados tanto en el anteproyecto como en el trabajo final.

Se elaboró un circuito de control formado con la tarjeta de desarrollo arduino mega 2560, esta etapa del proyecto permite activar la etapa de potencia, en base a las señales recibidas por los dispositivos de entrada, (modulo bluetooth HC-06, teclado matricial 4*4).

Se desarrolla una etapa de potencia, formada por un módulo relé de 5v y un contactor a 110 v, este último activa/ desactiva las cargas eléctricas (lámparas de 100w).

Se desarrolla el aplicativo en app inventor, el cual permite leer el código QR de la factura y además permite la comunicación desde el teléfono y el medidor. Se crea además una interfaz de usuario, esto para que el operario de la empresa de energía genere el código Qr.

Bibliografía

Abellán M. (2019). Qué es App Inventor: programoergosum. Recuperado el 31 de marzo de 2019, de: https://www.programoergosum.com/cursos-online/appinventor/27-curso-de-programacion-con-app-inventor/primeros-pasos

Carlos. (2018). Medidor de corriente, frecuencia de red y potencia no invasivo: rogerbit.com. Recuperado el 04 de junio de 2020, de http://rogerbit.com/wprb/2018/08/medidor-de-corriente-frecuencia-y-potencia-aparente-de-red-no-invasivo/

Comisión de Regulación de Energía y Gas RESOLUCIÓN96 DE 2004. (s.f). de xperta.legis.co: https://xperta.legis.co/visor/temp_legcol_07d4ccf0-5947-4546-907b-fd86cf875dc4

Llamas L. (2016). Usar un teclado matricial con arduino: Luisllamas.es. Recuperado el 02 de marzo de 2019: https://www.luisllamas.es/arduino-teclado-matricial/

Laboratorios MeI. (s.f). de sites.google.com: https://sites.google.com/site/labmediup/temas-de-la-unidad/pl-7-medicion-de-potencia/7-4-teoria-y-tecnica

Normatividad - Ministerio de Minas y Energía. (s. f.-a). Recuperado Agosto 16 de 2020, de https://www.minenergia.gov.co/web/guest/busquedas?param=40483

Normatividad - Ministerio de Minas y Energía. (s. f.-a). Recuperado Agosto 16 de 2020, de https://www.minenergia.gov.co/web/guest/busquedas?param=40483

Planas O. (2018). Vatio: Energia-nuclear.net. Recuperado el 04 de junio de 2020, de : https://energia-nuclear.net/energia/generacion-de-electricidad/vatio

Potencia en un circuito de corriente alterna. (S.f). de Proyecto987.es: http://www.proyecto987.es/corriente_alterna_11.html

¿Qué es Qt Creator? (2019), doc.arcgis.com, de: https://doc.arcgis.com/es/appstudio/extend-apps/whatisqtcreator.htm

Que es la potencia activa P (kW). (S.f). de Electro aplicada: https://www.electricaplicada.com/que-es-la-potencia-activa-p-kw/

SCT-013 mide el consumo eléctrico en tu casa con Arduino.(S.f). De ProgamaFacil.com: https://programarfacil.com/blog/arduino-blog/sct-013-consumo-electrico-arduino/

Tutorial sensor de corriente AC no invasivo SCT-013. (s.f). De naylampmechatronics: https://naylampmechatronics.com/blog/51_tutorial-sensor-de-corriente-ac-no-invasivo-s.html

Anexo 1

Código del arduino mega.

```
/*
  Autor del proyecto: Dario Fernando López L.
  Proyecto Final: Tesis de  Ingeniería Electrónica.
  Fecha: Octubre 26 del 2019.

  Genuino MEGA CONTADOR:

  Product ID: 0x0042
  Vendor ID: 0x2341
  Version: 0.01
  Serial Number: 55639313533351E00281
*/

#include <SoftwareSerial.h>       // Librería para crear un puerto serial vitual en otros pines.
#include <EEPROM.h>               // Librería para leer y escribir en la memoria interna del Arduino.
#include <Wire.h>                 // Librería para establecer comunicación I2C.
#include <LiquidCrystal_I2C.h>    // Librería que indica el uso de una pantalla LCD con modulos I2C.
#include <Keypad.h>               // Librería que indica el uso de un teclado matricial.
#include <Separador.h>            // Librería que separa las cadenas de caracteres según el delimitador.

// Configure the baud rate:
#define BAUDRATE_QT 9600
#define BAUDRATE_BT  19200

#define SERIAL_QT Serial          // Define el nombre en un macro para ser reemplazado por otro.

// Configure pinout Arduino para conexiones con los proyectos:

#define modulo_BT       13        // Define el pin digital 13.
#define Row1            12        // Define el pin digital 12.
#define Row2            11        // Define el pin digital 11.
#define Row3            10        // Define el pin digital 10.
#define Row4             9        // Define el pin digital 9.
#define Col1            8         // Define el pin digital 8.
#define Col2            7         // Define el pin digital 7.
```

```
#define Col3            6       // Define el pin digital 6.
#define Col4            5       // Define el pin digital 5.
#define Potencia        4       // Define el pin digital 4 para conectar un rele.
#define buzzer          3       // Define el pin digital 3 para conectar un buzzer.

#define Sensor_SCT_013  A0      // Define el pin analógico A0 para conectar un sensor.

// Configure pinout Arduino:
#define RX2             A9      // Define RX2 de Arduino MEGA
#define TX2             A8      // Define TX2 de Arduino MEGA

#define SDA             20      // Define el pin analógico 20 para establecer conexión I2C con otro dispositivo.
#define SCL             21      // Define el pin analógico 21 para establecer conexión I2C con otro dispositivo.

//************************************ Variables Macros ************************************************

#define eeAddress0              0       // Variable que define la ubicación donde queremos que se pongan los datos.
#define eeAddress1              10      // Variable que define la ubicación donde queremos que se pongan los datos.
#define eeAddress2              20      // Variable que define la ubicación donde queremos que se pongan los datos.
#define eeAddress3              25      // Variable que define la ubicación donde queremos que se pongan los datos.

#define eeCodeQR_Array          0       // Variable que define la ubicación donde queremos que se pongan los datos.
#define eeValueRecarga_Str      1       // Variable que define la ubicación donde queremos que se pongan los datos.
#define eeValueRecarga_Long     2       // Variable que define la ubicación donde queremos que se pongan los datos.
#define eelicenseActive_Bool    3       // Variable que define la ubicación donde queremos que se pongan los datos.

//************************************ Variables Globales ************************************************

const byte Rows  = 4;           // Cuatro filas
const byte Cols  = 4;           // Cuatro columnas

byte Pins_Rows[]  = { Row1, Row2, Row3, Row4 };     // Pines Arduino a los que contamos las filas.
byte Pins_Cols[]  = { Col1, Col2, Col3, Col4 };     // Pines Arduino a los que contamos las columnas.
```

```
char Teclas [ Rows ][ Cols ] =
{
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

char codigoQR_Recarga[7]; // Aqui se almacena el código QR y el valor de la recarga enviado desde la interfaz de QT (DINAMICO QUE SE INTRODUCE POR PUERTO SERIAL)
// Para cambiar el tamaño de la clave, solo hay que cambiar el tamaño del array
char valueRecarga_Array[7];

int posicion = 0;              // Variable necesaria para la clave
int cursor = 5;                // Variable posicion inicial de la clave en el LCD
int clave = 0;                 // Variable para el LCD
int luz = 0;                   // Variable para el LCD
int tiempo = 0;                // Variable para el LCD

String codeQR      = "";       // Variable que almacena la primera cadena separada por el caracter ','.
String valueRecarga = "";      // Variable que almacena la segunda cadena separada por el caracter ','.

double valueRecargaLong = 0;   // Variable que almacena el valor del dinero recargado enviado en String a un valor numerico.
bool licenseActive = false;    // Variable bandera que notifica si el sistema tiene una recarga valida.
bool license = false;          // Variable bandera auxiliar que notifica si el sistema tiene una recarga valida. esto fue lo ultimo en cambiar.

long consumido = 0;            // Variable que almacena el consumo en dinero retornado para comparar con el recargado.

const float constResolu = 1.396;      // Si utiliza una resistencia de 22 ohm y un capacitor electrolitico de 10uF-50V entre la salida del AMPLIFICADOR OPERACIONAL y tierra
const float constFiltro =  0.7;       // Constate para filtrar valores paracitos de lectura debido a que se utiliza un AMPLIFICADOR OPERACIONAL (SE USA CON: constResolu_1 )

const float voltajeAC = 120.0;        // Variable constante que almacena el valor de la red electrica.
const float corrienteMaxSensor = 100.0;  // Variable constante que almacena el valor de la corriente maxima que soporta el sensor de corriente.

double kilos = 0;              // Variable que almacena el cálculo de los kilovatios hora usados.
int peakPower = 0;             // Variable
```

```
unsigned long startMillis;                  // Variable que almacena el tiempo inicial o tiempo transurrido desde que se inicio el contador de tiempo.
unsigned long endMillis;                    // Variable que almacena el tiempo final o tiempo transurrido desde que se inicio el contador de tiempo.

const double valor_kWh = 530.56;            // Variable constante que se cobra por el kilovatio hora.
double dineroConsumo = 0.0;                 // Variable variable que almacena el consumo energetico en terminos de dinero.

int opBack = 0;                             // Variable bandera que devuelve atras en el menu.
bool lockNumeral = false;                   // Variable bandera que activa el numeral solo cuando se usa la opción del teclado.

//********************************** Declaración de Objetos ***********************************************
// Aquí conectamos los pins RXD,TDX del módulo Bluetooth.
SoftwareSerial SERIAL_BT (RX2, TX2);        // Crea nuevo puerto serial vitual por los pines: PIN 11 (RX) a TX y PIN 12 (TX) a RX para la conexion al bluetooth

// Aquí se configuran los pines asignados a la pantalla del PCF8574
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

// Aquí se configuran los pines asignados al teclado matricial 4x4 de membrana
Keypad Teclado = Keypad( makeKeymap(Teclas), Pins_Rows, Pins_Cols, Rows, Cols );

Separador buscaComa;                        // Crea un objeto con el cual se buscan las comas que separan los mensajes recibidos.
Separador buscaPunto;                       // Crea un objeto con el cual se buscan las comas que separan los mensajes recibidos.

//********************************** Declaración Funciones ***********************************************
uint8_t getch();                            // Función que verifica y lee si llego un dato por el puerto serial principal.
uint8_t getchBT();                          // Función que verifica y lee si llego un dato por el puerto serial secundario o (virtual) del módulo Bluetooth.

void msgLicense();                          // Función que recibe los mensajes de la interfaz Qt.
void validateLicenseBT();                   // Función que recibe los mensajes de la App por Bluetooth.
void validateLicenseKeypad();               // Función que recibe los mensajes por el teclado matricial 4x4.
long activateElectricMeter();               // Función que activa la etapa de potencia para trabajar según el valor recargado.
void convertCadenaNumero();                 // Función que convierte el valor de la recarga en de cadena a valor numerico.
```

```
void initialScreenshot();                 // Función que muestra un pantallazo inicial en la interfaz LCD.
void menuScreen();                        // Función que muestra el menú en pantalla luego de recibir el código QR.
void erase_a_Digit(char);                 // Función que borra un digito errado y lo muestra en la interfaz LCD
void reset_digiteCodeQR_Screenshot();     // Función que resetea la clave y muestra un pantallazo en la interfaz LCD.
void notification_CodeQR_Valid();         // Función donde notifica de manera visual y auditiva que el codigo QR ingresado es valido.

void corrienteMedida();                   // Función que muestra la medida de la corriente sensada.
float get_corriente();                    // Función que adquiere la medida de la corriente sensada.

void keypadEvent(KeypadEvent pulsacion); // Función que cuida algunos eventos especiales del teclado.

void read_Data_EEPROM(int);               // Función que lee los valores almacenados en la memoria EEPROM del Arduino.
void write_Data_EEPROM(int);              // Función que escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.

//********************************** Inicio Programa Principal **********************************************
void setup() {
  Wire.begin();                   // Iniciamos la comunicación I2C, en el maestro la dirección no es obligatoria
  SERIAL_QT.begin( BAUDRATE_QT );    // Iniciamos el puerto que esta conectado al PC a 19200 Baudios
  SERIAL_BT.begin( BAUDRATE_BT );    // Iniciamos el puerto nuevo Serial_BT a 19200 Baudios
  lcd.begin( 16, 2 );             // Inicializamos el LCD. columns, rows. use 16,2 for a 16x2 LCD, etc.

  Teclado.addEventListener(keypadEvent); // Add an event listener for this keypad

  pinMode( modulo_BT, OUTPUT );     // Se configura como salida
  pinMode( Potencia, OUTPUT );      // Se configura como salida
  pinMode( buzzer, OUTPUT );        // Se configura como salida
  analogReference(INTERNAL1V1);       // Referencia interna Arduino MEGA para mejorar la resolución de la adquisición de datos.

  read_Data_EEPROM(eelicenseActive_Bool);           // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.

  if ( license ) {
```

```
   menuScreen();                          // Invoca la función que muestra el menú en pantalla luego de recibir el código QR.
   read_Data_EEPROM(eeCodeQR_Array);      // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.
   read_Data_EEPROM(eeValueRecarga_Str);  // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.
   read_Data_EEPROM(eeValueRecarga_Long); // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.

   valueRecarga = String(valueRecarga_Array);    // Convierte el arreglo de caracteres en un String.

  } else {
   initialScreenshot();             // Invoca la función que muestra un pantallazo inicial en la interfaz LCD.
  }
}

void loop() {
  if (SERIAL_QT.available()) {      // Si llega caracteres por el puerto serial principal
   msgLicense();                    // Invoca la función que recibe los mensajes de la interfaz Qt
   read_Data_EEPROM(eelicenseActive_Bool);  // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.
   SERIAL_QT.end();                 // Finaliza el puerto principal Serial_QT.
  }

  if ( license ) {
   char pulsacion1 = Teclado.getKey() ;     // Lee la tecla pulsada y la almacena en la variable pulsacion1.
   if ( pulsacion1 != 0 && pulsacion1 != '*' ) {        // Si el valor es diferente de 0 y diferente de * se ha pulsado una tecla.
     tone(buzzer, 350);                     // Activa el tono de pulsación a una frecuencia de 350 Hz durante 200 milisegundos.
     delay(200);                    // Espera 200 milisegundos con el tono activo.
     noTone(buzzer);                // Desactiva el tono.

     if ( pulsacion1 == 'A' ) {     // Si la tecla pulsada es la 'A' entonces...
      SERIAL_BT.begin( BAUDRATE_BT );     // Iniciamos el puerto nuevo Serial_BT a 19200 Baudios.
       digitalWrite(modulo_BT, HIGH);     // Enciende el módulo bluetooth HC-06.
       lcd.clear();                 // Limpia la LCD antes de poner un nuevo dato.
       lcd.setCursor(0, 0);         // Situa el cursor el la posición 0 de la linea 0.
       lcd.print("ACTIVE BLUETOOTH");     // Escribe el mensaje en LCD.
       lcd.setCursor(3, 1);         // Situa el cursor el la posición 3 de la linea 1.
       lcd.print("DE SU MOVIL");    // Escribe el mensaje en LCD.

       while ( license ) {          // Si license es true repitase infinitamente.
```

```
        lockNumeral = true;              // Variable bandera que activa el numeral solo cuando se usa la opción del teclado.

        pulsacion1 = Teclado.getKey() ;    // Lee la tecla pulsada y la almacena en la variable pulsacion1.

        if ( opBack == -1 ) {            // Si se pulsa la tecla * devuelve una variable que indica que se retrocede en el menu.
          opBack = 0;                  // Inicializa la variable para futuros usos.
          break;                       // Rompe el ciclo infinito para poder volver al menu inicial.
        }
        if (SERIAL_BT.available()) {      // Si llega caracteres por el puerto serial virtual
          validateLicenseBT();             // Invoca la función que recibe los mensajes de la App por Bluetooth
        }
       }
     } else if ( pulsacion1 == 'B' ) {     // Si la tecla pulsada es la 'B' entonces...
      reset_digiteCodeQR_Screenshot();     // Invoca la función que resetea la clave y muestra un pantallazo en la interfaz LCD.

      while ( license ) {               // Si la licencia no es true muestre los mensajes.
        lockNumeral = false;             // Variable bandera que activa el numeral solo cuando se usa la opción del teclado.
        validateLicenseKeypad();          // Invoca la función que recibe los mensajes por el teclado matricial 4x4.
        if ( opBack == -1 ) {            // Si se pulsa la tecla * devuelve una variable que indica que se retrocede en el menu.
          opBack = 0;                  // Inicializa la variable para futuros usos.
          break;                       // Rompe el ciclo infinito para poder volver al menu inicial.
        }
       }
     }
   }
  }
}

//************************************** Inicio Funciones ************************************************************
uint8_t getch() {                    // Función que verifica y lee si llego un dato por el puerto serial principal.
  while (!SERIAL_QT.available());     // Repitase hasta que no alla un dato disponible
  return SERIAL_QT.read();           // Retorna el valor del dato leido por el puerto serie.
}

uint8_t getchBT() {                  // Función que verifica y lee si llego un dato por el puerto serial secundario o (virtual) del módulo Bluetooth.
  while (!SERIAL_BT.available());     // Repitase hasta que no alla un dato disponible
```

```
  return SERIAL_BT.read();          // Retorna el valor del dato leido por el puerto serie.
}

void msgLicense() {      // Función que recibe los mensajes de la interfaz Qt.
  char inChar = "";      // Variable que almacena los caracteres recibidos.
  String msgQt = "";     // Cadena que concatena todos los caracteres recibidos.

  while ( (inChar = getch()) != '\n' ) {          // Recorre el mensaje hasta encontrar el '\n'
    msgQt += inChar;                              // Almacena los caracteres en una variable String
  }

  codeQR       = buscaComa.separa(msgQt, ',', 0);     // La primera cadena separada por el caracter ',' se almacena en una variable tipo String.
  valueRecarga = buscaComa.separa(msgQt, ',', 1);     // La segunda cadena separada por el caracter ',' se almacena en una variable tipo String.

  valueRecarga.toCharArray(valueRecarga_Array, 8);    // Convierte el mensaje de una cadena a un arreglo de caracteres.
  write_Data_EEPROM(eeValueRecarga_Str);              // Invoca la función que lee y escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.

  codeQR.toCharArray(codigoQR_Recarga, 8);            // Convierte el mensaje de una cadena a un arreglo de caracteres.
  write_Data_EEPROM(eeCodeQR_Array);                  // Invoca la función que lee y escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.

  convertCadenaNumero();                              // Invoca la función que convierte el valor de la recarga en de cadena a valor numerico.

  lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(0, 0);            // Situa el cursor el la posición 1 de la linea 0.
  lcd.print("ECO CODIGO QR!!!");  // Escribe el mensaje en LCD.
  lcd.setCursor(0, 1);            // Situa el cursor el la posición 1 de la linea 1.
  lcd.print(msgQt);               // Escribe el mensaje en LCD.
  delay(5000);                    // Espera 5 segundos para visualizar el mensaje en la LCD.

  SERIAL_QT.print(msgQt + "\n");                      // Reenvia el mensaje recibido a modo de eco para informar que si se recibio el mensaje.
  menuScreen();                                       // Invoca la función que muestra el menú en pantalla luego de recibir el código QR.

  licenseActive = true;                               // Indica que el codigo QR y el valor recargado fueron almacenados correctamente.
  write_Data_EEPROM(eelicenseActive_Bool);            // Invoca la función que lee y escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.
}
```

```
void validateLicenseBT() {        // Función que recibe los mensajes de la App por Bluetooth.
  char inCharBT = "";             // Variable que almacena los caracteres recibidos.
  String msgBT = "";              // Cadena que concatena todos los caracteres recibidos.

  while ( (inCharBT = getchBT()) != '\n' ) {      // Recorre el mensaje hasta encontrar el '\n'
    msgBT += inCharBT;                            // Almacena los caracteres en una variable String
  }

  lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(1, 0);            // Situa el cursor el la posición 1 de la linea 0.
  lcd.print("CodeQR: ");          // Escribe el mensaje en LCD.
  lcd.print(codeQR);              // Escribe el mensaje en LCD.

  lcd.setCursor(1, 1);            // Situa el cursor el la posición 1 de la linea 1.
  lcd.print("CodeBT: ");          // Escribe el mensaje en LCD.
  lcd.print(msgBT);               // Escribe el mensaje en LCD.
  delay(5000);                    // Espera 5 segundos para visualizar el mensaje en la LCD.

  if ( msgBT == codeQR ) {        // Si msgBT es igual a codeQR entonces...

    notification_CodeQR_Valid();        // Invoca la función donde notifica de manera visual y auditiva que el codigo QR ingresado es valido.

    SERIAL_BT.print('1');          // Reenvia el mensaje recibido a modo de eco para informar que si se recibio el mensaje vía bluetooth.
    delay(250);                   // Espera 250 milisegundos para enviar el mensaje de confirmación a la app del celular.
    digitalWrite(modulo_BT, LOW);     // Apaga el módulo bluetooth HC-06.

    lcd.clear();                  // Limpia la LCD antes de poner un nuevo dato.
    msgBT = "";                   // Limpia la variable para nuevos codigos
    licenseActive = false;        // Indica que el numero de licencia ya se utilizo correctamente.
    write_Data_EEPROM(eelicenseActive_Bool);      // Invoca la función que lee y escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.
    read_Data_EEPROM(eelicenseActive_Bool);       // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.

    do {
      consumido = activateElectricMeter();              // Invoca la función que activa la etapa de potencia para trabajar según el valor recargado
    } while ( consumido !=  valueRecargaLong );     //

    while ( !license ) {              // Si la licencia no es true muestre los mensajes
      lcd.clear();                  // Limpia la LCD antes de poner un nuevo dato.
      lcd.setCursor(1, 0);          // Situa el cursor el la posición 1 de la linea 0.
      lcd.print("POTENCIA TOTAL");      // Escribe el mensaje en LCD.
```

```
    lcd.setCursor(4, 1);              // Situa el cursor el la posición 4 de la linea 1.
    lcd.print("CONSUMIDA");           // Escribe el mensaje en LCD.
    delay(5000);                      // Espera 5 segundos para visualizar el mensaje en la LCD.

    lcd.clear();                      // Limpia la LCD antes de poner un nuevo dato.
    lcd.setCursor(2, 0);              // Situa el cursor el la posición 2 de la linea 0.
    lcd.print("RECARGUE UN");         // Escribe el mensaje en LCD.

    lcd.setCursor(0, 1);              // Situa el cursor el la posición 0 de la linea 1.
    lcd.print("NUEVO CODIGO QR!");    // Escribe el mensaje en LCD.
    delay(5000);                      // Espera 5 segundos para visualizar el mensaje en la LCD.
   }

  } else {
   SERIAL_BT.print('0');               // Reenvia el mensaje recibido a modo de eco para informar que si se recibio el mensaje vía bluetooth.
   delay(250);                        // Espera 250 milisegundos para enviar el mensaje de confirmación a la app del celular.
   msgBT = "";                        // Limpia la variable para nuevos codigos.

   lcd.clear();                       // Limpia la LCD antes de poner un nuevo dato.
   lcd.setCursor(0, 0);               // Situa el cursor el la posición 0 de la linea 0.
   lcd.print( "CODIGO ERROR!!!" );     // Escribe el mensaje en LCD.
   delay(1500);                       // Espera 1,5 segundos para visualizar el mensaje en la LCD.

   lcd.clear();                       // Limpia la LCD antes de poner un nuevo dato.
   lcd.setCursor(3, 0);               // Situa el cursor el la posición 3 de la linea 0.
   lcd.print("REENVIE EL");           // Escribe el mensaje en LCD.

   lcd.setCursor(3, 1);               // Situa el cursor el la posición 3 de la linea 1.
   lcd.print("CODIGO QR");            // Escribe el mensaje en LCD.
   delay(1500);                       // Espera 1,5 segundos para visualizar el mensaje en la LCD.
  }
}

void validateLicenseKeypad() {          // Función que recibe los mensajes por el teclado matricial 4x4.
  char pulsacion = Teclado.getKey();    // Lee la tecla pulsada y la almacena en la variable pulsación.
  if ( pulsacion != 0 ) {               // Si el valor es 0 es que no se ha pulsado ninguna tecla
    if (pulsacion != '#' && pulsacion != '*' && clave == 0) {     // descartamos almohadilla y asterisco

      lcd.print(pulsacion);             // Imprimimos el caracter que fue pulsado.
      cursor++;                         // Incrementa el cursor.
      tone(buzzer, 350);                // Activa el tono de pulsación a una frecuencia de 350 Hz durante 200 milisegundos.
```

```
    delay(200);                 // Espera 200 milisegundos con el tono activo.
    noTone(buzzer);             // Desactiva el tono.

    //--- Condicionales para comprobar la clave introducida ----------

    if (pulsacion == codigoQR_Recarga[posicion]) {      // Compara la entrada con cada uno de los digitos, uno a uno.
      posicion ++;                                      // Aumenta en uno el contador de posicion si es correcto el digito.
    }

    if ( posicion == 7 ) {                  // Comprueba que se han introducido los 7 caracteres correctamente.

      notification_CodeQR_Valid();          // Invoca la función donde notifica de manera visual y auditiva que el codigo QR ingresado es valido.

      lcd.setCursor(cursor, 1);             // Situa el cursor el la posición 5 de la linea 1.
      clave = 1;                            // Indica que se ha introducido la clave correctamente.
      lcd.clear();                          // Limpia la LCD antes de poner un nuevo dato.
      licenseActive = false;                // Indica que el numero de licencia ya se utilizo correctamente.
      write_Data_EEPROM(eelicenseActive_Bool);      // Invoca la función que lee y escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.
      read_Data_EEPROM(eelicenseActive_Bool);       // Invoca la función que lee los valores almacenados en la memoria EEPROM del Arduino.

      do {
        consumido = activateElectricMeter();        // Invoca la función que activa la etapa de potencia para trabajar según el valor recargado
      } while ( consumido !=  valueRecargaLong );   // Mientras consumo no sea igual al valor recargado se va a repetir la funcion.

      while ( !license ) {              // Si la licencia no es true muestre los mensajes
        lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
        lcd.setCursor(1, 0);            // Situa el cursor el la posición 1 de la linea 0.
        lcd.print("POTENCIA TOTAL");     // Escribe el mensaje en LCD.

        lcd.setCursor(4, 1);            // Situa el cursor el la posición 4 de la linea 1.
        lcd.print("CONSUMIDA");         // Escribe el mensaje en LCD.
        delay(5000);                    // Espera 5 segundos para visualizar el mensaje en la LCD.

        lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
        lcd.setCursor(2, 0);            // Situa el cursor el la posición 2 de la linea 0.
        lcd.print("RECARGUE UN");       // Escribe el mensaje en LCD.

        lcd.setCursor(0, 1);            // Situa el cursor el la posición 0 de la linea 1.
```

```
        lcd.print("NUEVO CODIGO QR!");    // Escribe el mensaje en LCD.
        delay(5000);                      // Espera 5 segundos para visualizar el mensaje en la LCD.
       }

     }
     //--- En el caso de que este incompleta o no hayamos acertado ----------
     if (cursor > 11) {                   // comprobamos que no pase de la septima posición
       cursor = 5;                       // Inicializa la variable del cursor en la posición 5.
       posicion = 0;                     // Inicializa la variable de posición.
       lcd.setCursor(cursor, 1);         // Situa el cursor el la posición 5 de la linea 1.
       lcd.print("       ");             // Borra la clave de la pantalla LCD.
       lcd.setCursor(cursor, 1);         // Situa el cursor el la posición 5 de la linea 1.
       if (clave == 0) {                 // Si clave es igual a 0 entonces... comprueba que no hemos
acertado
         tone(buzzer, 70, 500);          // Activa el tono de clave incorrecta a una frecuencia de 70
Hz durante 500 milisegundos.
         delay(250);                     // Espera 250 milisegundos con el tono activo.
         noTone(buzzer);                 // Desactiva el tono.
       }
     }
    }
   }
}

void keypadEvent(KeypadEvent pulsacion) {     // Función que cuida algunos eventos especiales
del teclado.
  switch (Teclado.getState()) {
    case PRESSED: // Entra en este evento la tecla es PRESIONADA (por un periodo corto de
tiempo).
      //--- Condicionales para borrar digitos introducidos ------------

     if ( pulsacion == '#' && lockNumeral == false ) { // Comprueba si la tecla # fue pulsada para
realizar la función de borrar un digito a la vez.
       if ( cursor != 5 ) {                  // Si cursor es diferente de 5 entonces...
         erase_a_Digit(pulsacion);           // Invoca la función que borra un digito errado y lo
muestra en la interfaz LCD.
       } else {                            // Sino entonces...
         tone(buzzer, 350);                  // Activa el tono de pulsación a una frecuencia de 350
Hz durante 200 milisegundos.
         delay(200);                       // Espera 200 milisegundos con el tono activo.
         noTone(buzzer);                   // Desactiva el tono.
       }

     }

     //
     //--- Condicional para volver atras en el menu opciones ---------
```

```
    if ( pulsacion == '*' ) {              // Comprueba si la tecla * fue pulsada para realizar la función de volver atras en el menu de opciones.
      tone(buzzer, 350);                   // Activa el tono de pulsación a una frecuencia de 350 Hz durante 200 milisegundos.
      delay(200);                          // Espera 200 milisegundos con el tono activo.
      noTone(buzzer);                      // Desactiva el tono.

      if (digitalRead(modulo_BT) == HIGH) { // Si esta encendido entonces...
        digitalWrite(modulo_BT, LOW);      // Apaga el Módulo Bluetooth HC-06.
        SERIAL_BT.flush();                 // Espera a que se complete la transmisión de datos seriales salientes.
        SERIAL_BT.end();                   // Finaliza el puerto nuevo Serial_BT.
      }

      if ( !lockNumeral ) {                // Si lockNumeral no es true entonces...
        lockNumeral = true;                // Variable bandera que desactiva el numeral solo cuando se usa la opción del teclado.
      }

      opBack = -1;                         // Variable que indica que se devuelve en el menu.
      menuScreen();                        // Invoca la función que muestra el menú en pantalla luego de recibir el código QR.
    }

    //-------------------------------------------------------------------
    break;

  case HOLD: // Entra en este evento la tecla es RETENIDA eso quiere decir si se deja precionado el por mucho tiempo.
    //--- Condicionales para resetear clave introducida ----------

    if ( pulsacion == '#' && lockNumeral == false ) {     // Comprueba si la tecla # fue pulsada para realizar la función de borrar todo el codigo QR mal digitado.
      reset_digiteCodeQR_Screenshot();                    // Invoca la función que resetea la clave y muestra un pantallazo en la interfaz LCD.
    }
    //-------------------------------------------------------------------
    break;
  }
}

void convertCadenaNumero() {      // Función que convierte el valor de la recarga en de cadena a valor numerico.

  String value = valueRecarga.substring(1, valueRecarga.length());    // Guarda la ultima palabra del mensaje ("255089C,$5.000\n") en el String ejm: "$5.000"
```

```
  value = buscaPunto.separa(value, '.', 0);                    // La primera cadena separada por el caracter '.' se almacena en una variable tipo String.

  valueRecargaLong = value.toInt();            // El String numerico se convierte en un entero para realizar operaciones con el.
  //<< valueRecargaLong *= 1000;                    // Se multiplica por 1000 para trabajar con los mismos valores de precio (Descomentar para uso REAL).

  write_Data_EEPROM(eeValueRecarga_Long);        // Invoca la función que lee y escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.
}


void initialScreenshot() {            // Función que muestra un pantallazo inicial en la interfaz LCD.
  lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor( 2, 0 );              // Situa el cursor el la posición 2 de la linea 0.
  lcd.print( "BIENVENIDO AL" );       // Escribe el mensaje en LCD.
  lcd.setCursor( 3, 1 );              // Situa el cursor el la posición 3 de la linea 1.
  lcd.print( "CONTADUINO" );          // Escribe el mensaje en LCD.
  lcd.setCursor( cursor, 1 );         // Situa el cursor el la posición 5 de la linea 1.
}

void menuScreen() {                    // Función que muestra el menú en pantalla luego de recibir el código QR.

  if ( opBack == 0 ) {                // Si no se han regresado en el menu entonces...
    lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
    lcd.setCursor(0, 0);              // Situa el cursor el la posición 0 de la linea 0.
    lcd.print("LICENCIA CODIGO");     // Escribe el mensaje en LCD.

    lcd.setCursor(4, 1);              // Situa el cursor el la posición 4 de la linea 1.
    lcd.print("RECIBIDO");            // Escribe el mensaje en LCD.

  } else {
    initialScreenshot();              // Invoca la función que muestra un pantallazo inicial en la interfaz LCD.
  }

  delay(1500);                      // Espera 1,5 segundos para visualizar el mensaje en la LCD.
  lcd.clear();                    // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(0, 0);              // Situa el cursor el la posición 0 de la linea 0.
  lcd.print("ELIJA UN OPCION:");      // Escribe el mensaje en LCD.

  lcd.setCursor(0, 1);              // Situa el cursor el la posición 0 de la linea 1.
  lcd.print("A.MOVIL  B.TECLA");      // Escribe el mensaje en LCD.
}
```

```
void erase_a_Digit(char pulsacion) { // Función que borra un digito errado y lo muestra en la interfaz LCD
  pulsacion = 0;                  // borramos el valor para poder leer el siguiente condicional
  cursor--;                       // Incrementa el cursor.

  if (pulsacion == codigoQR_Recarga[pulsacion]) {       // Compara la entrada con cada uno de los digitos, uno a uno.
    pulsacion --;                                       // Disminuye en uno el contador de posición si se borra el digito.
  }

  lcd.setCursor(cursor, 1);          // Situa el cursor el la posición 5 de la linea 1.
  lcd.print(" ");                    // Borra la clave de la pantalla LCD.
  lcd.setCursor(cursor, 1);          // Situa el cursor el la posición 5 de la linea 1.
  tone(buzzer, 350);                 // Activa el tono de pulsación a una frecuencia de 350 Hz durante 200 milisegundos.
  delay(200);                        // Espera 200 milisegundos con el tono activo.
  noTone(buzzer);                    // Desactiva el tono.
}

void reset_digiteCodeQR_Screenshot() { // Función que resetea la clave y muestra un pantallazo en la interfaz LCD.
  lcd.clear();                       // Limpia la LCD antes de poner un nuevo dato.
  posicion = 0;                      // Inicializa la posicion en los digitos en la LDC.
  cursor = 5;                        // Coloca el cursor en la columna 5.
  clave = 0;                         // Inicializa la variable clave para volver a introducir los numeros.
  posicion = 0;                      // Inicializa la posicion en los digitos en la LDC.
  lcd.setCursor(0, 0);               // Situa el cursor el la posición 0 de la linea 0.
  lcd.print("DIGITE CODIGO QR");     // Escribe el mensaje en LCD.

  lcd.setCursor(cursor, 1);          // Situa el cursor el la posición 5 de la linea 1.
  lcd.print("      ");               // Borra de la pantalla los numeros
  lcd.setCursor(cursor, 1);          // Situa el cursor el la posición 5 de la linea 1.
}

void notification_CodeQR_Valid() {   // Función donde notifica de manera visual y auditiva que el codigo QR ingresado es valido.
  lcd.clear();                       // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(0, 0);               // Situa el cursor el la posición 0 de la linea 0.
  lcd.print( "CODIGO VALIDADO!" );   // Escribe el mensaje en LCD.
  delay(200);                        // Espera 200 milisegundos para visualizar el mensaje en la LCD.

  tone(buzzer, 500 );                // Activa el tono de clave correcta a una frecuencia de 500 Hz durante 100 milisegundos.
  delay(100);                        // Espera 100 milisegundos con el tono activo.
  noTone(buzzer);                    // Desactiva el tono.
```

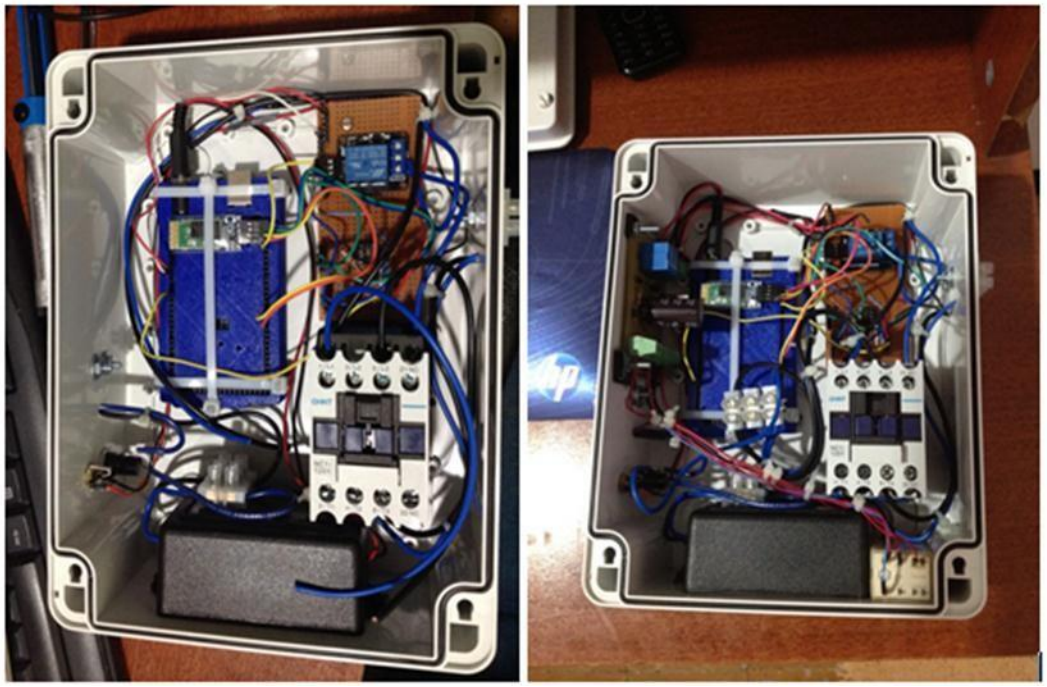
```
  tone(buzzer, 600 );                 // Activa el tono de clave correcta a una frecuencia de 600 Hz durante 100 milisegundos.
  delay(100);                       // Espera 100 milisegundos con el tono activo.
  noTone(buzzer);                   // Desactiva el tono.
  tone(buzzer, 800 );                 // Activa el tono de clave correcta a una frecuencia de 800 Hz durante 100 milisegundos.
  delay(100);                       // Espera 100 milisegundos con el tono activo.
  noTone(buzzer);                   // Desactiva el tono.

  lcd.setCursor(1, 1);              // Situa el cursor el la posición 1 de la linea 1.
  lcd.print("VALOR: ");             // Escribe el mensaje en LCD.
  lcd.print(valueRecarga);          // Escribe el mensaje en LCD.
  delay(1500);                      // Espera 1,5 segundos para visualizar el mensaje en la LCD.

  startMillis = millis();           // Almacena el valor del tiempo que empieza a contar al pasar por esta linea.
}

long activateElectricMeter() {        // Función que activa la etapa de potencia para trabajar según el valor recargado
  lcd.setCursor(4, 0);              // Situa el cursor el la posición 4 de la linea 0.
  lcd.print("FUNCTION");            // Escribe el mensaje en LCD.

  lcd.setCursor(1, 1);              // Situa el cursor el la posición 1 de la linea 1.
  lcd.print("ACTIVATE POWER");       // Escribe el mensaje en LCD.
  delay(1500);                      // Espera 1,5 segundos para visualizar el mensaje en la LCD.
  lcd.clear();                      // Limpia la LCD antes de poner un nuevo dato.

  digitalWrite(Potencia, HIGH);       // Enciende los Bombillos.

  do {
    corrienteMedida();                      // Invoca la función que muestra la medida de la corriente sensada.
    if ( dineroConsumo >= valueRecargaLong ) {     // Si el dinero consumido es mayor e igual al valor de la recarga, entonces...
      dineroConsumo = valueRecargaLong;         // Iguala los valores.
      digitalWrite(Potencia, LOW);              // Apaga los Bombillos.
      return dineroConsumo;                     // Retorna el valor del dinero consumido.
    }
  } while ( dineroConsumo != valueRecargaLong ); // Hagalo mientras el dineroConsumo sea direferente del valueRecargaLong.
}

void corrienteMedida() {              // Función que muestra la medida de la corriente sensada.

  float Irms = get_corriente();       // Corriente eficaz (A)
  float RMSPower = voltajeAC * Irms;    // Calcula los RMS Power con un voltaje de 110VAC.
```

```
  endMillis = millis();                                  // Variable que almacena el tiempo final o tiempo transurrido desde que se inicio el contador de tiempo.
  unsigned long time = endMillis - startMillis;          // Variable que almacena la diferencia de tiempo en que inicio el contador y el tiempo transcurrido.
  kilos = kilos - ((RMSPower * time) / (1000000 * 3600)) ;  // Calcular kilovatios hora usados.

  dineroConsumo = kilos * valor_kWh;     // Variable que procesa y almacena el valor en dinero del consumo kilovatio hora.
  startMillis = millis();                // Variable que almacena el tiempo inicial o tiempo transurrido desde que se inicio el contador de tiempo.

  lcd.clear();                  // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(3, 0);          // Situa el cursor el la posición 3 de la linea 0. Displays all current data
  lcd.print("IRMS: ");          // Escribe el mensaje en LCD.
  lcd.print(Irms);              // Escribe el mensaje en LCD. (VALOR DE LA CORRIENTE SENSADA EN RMS). RMSCurrent
  lcd.print("A");               // Escribe el mensaje en LCD.
  lcd.setCursor(0, 1);          // Escribe el mensaje en LCD.
  lcd.print("PWRFULL: ");       // Escribe el mensaje en LCD.
  lcd.print(RMSPower);          // Escribe el mensaje en LCD. (VALOR DE LA POTENCIA TOTAL CONSUMIDA POR LA CARGA).
  lcd.print("W");               // Escribe el mensaje en LCD.
  delay(3000);                  // Espera 3 segundos para visualizar el mensaje en la LCD.

  lcd.clear();                  // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(0, 0);          // Situa el cursor el la posición 0 de la linea 0.
  lcd.print("QR CODE: ");       // Escribe el mensaje en LCD.
  lcd.print(codeQR);            // Escribe el mensaje en LCD.
  lcd.setCursor(0, 1);          // Situa el cursor el la posición 0 de la linea 0.
  lcd.print("RECARGA: ");       // Escribe el mensaje en LCD.
  lcd.print(valueRecarga);      // Escribe el mensaje en LCD. (VALOR NUMERICO CADENA).
  delay(3000);                  // Espera 3 segundos para visualizar el mensaje en la LCD.

  lcd.clear();                  // Limpia la LCD antes de poner un nuevo dato.
  lcd.setCursor(0, 0);          // Situa el cursor el la posición 0 de la linea 0.
  lcd.print("ENERGY: ");        // Escribe el mensaje en LCD.
  lcd.print(kilos);             // Escribe el mensaje en LCD.
  lcd.print("kWh");             // Escribe el mensaje en LCD.
  lcd.setCursor(0, 1);          // Situa el cursor el la posición 0 de la linea 1.
  lcd.print("CONSUM: ");        // Escribe el mensaje en LCD.
  lcd.print("$");               // Escribe el mensaje en LCD.
  lcd.print(dineroConsumo);     // Escribe el mensaje en LCD.
  delay(3000);                  // Espera 3 segundos para visualizar el mensaje en la LCD.
}
```

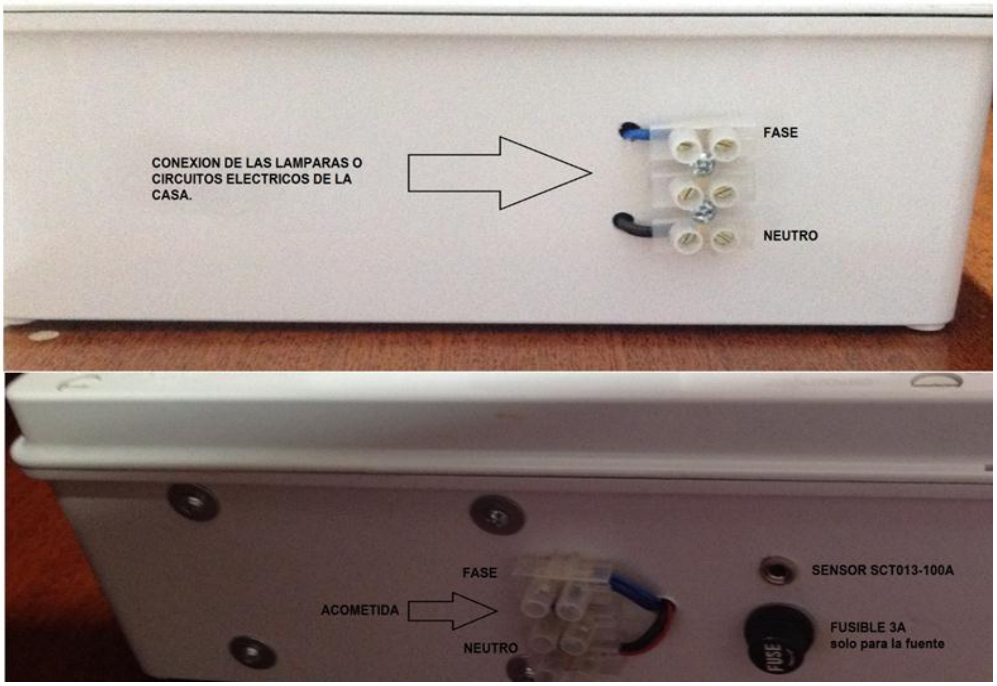
```
float get_corriente() {           // Función que adquiere la medida de la corriente sensada.
  float voltajeSensor = 0;        // Inicializa la variable del sensor de corriente.
  float corriente = 0;            // Inicializa la variable de corriente.
  float Sumatoria = 0;            // Inicializa la variable de Sumatoria.
  long tiempo = millis();         // Inicializa la variable contador de tiempo.
  int N = 0;                      // Inicializa la variable N.

  while (millis() - tiempo < 500) //Duración 0.5 segundos(Aprox. 30 ciclos de 60Hz)
  {
    voltajeSensor = analogRead(Sensor_SCT_013) * ( constResolu / 1023.0);     // Adquiere los
valores de voltaje del sensor.
    //corriente=voltajeSensor*30.0;                                          // Ajusta la resolución del sensor
corriente=VoltajeSensor*(30A/1V)
    corriente = voltajeSensor * corrienteMaxSensor;                          // Ajusta la resolución del
sensor corriente=VoltajeSensor*(100A/1V)
    Sumatoria = Sumatoria + sq(corriente);                                   // Cálcula la Sumatoria de
Cuadrados.
    N = N + 1;                                                               // Incrementa la variable N.
    delay(1);                                                                // Espera 1 milisegundo para realizar el cálculo
de la Sumatoria.
  }

  Sumatoria = Sumatoria / 2;          // Para compensar los cuadrados de los semiciclos negativos.
  corriente = sqrt((Sumatoria) / N);  // ecuación del RMS

  if ( corriente <= constFiltro ) { // Si no hay flujo de corriente
    corriente = 0;                  // Se iguala a CERO para no tener errores
  }
  return (corriente);               // Retorna el valor de la corriente
}

void read_Data_EEPROM(int ctrl) {           // Función que lee los valores almacenados en la
memoria EEPROM del Arduino.

  switch (ctrl)
  {
    case eeCodeQR_Array:
      codeQR = String(EEPROM.get( eeAddress0, codigoQR_Recarga ));                // Lee el dato
recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
    case eeValueRecarga_Str:
      valueRecarga = String(EEPROM.get( eeAddress1, valueRecarga_Array ));        // Lee el dato
recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
    case eeValueRecarga_Long:
```

```
      valueRecargaLong = EEPROM.get( eeAddress2, valueRecargaLong );              // Lee el dato recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
    case eelicenseActive_Bool:
      license = EEPROM.get( eeAddress3, licenseActive );                         // Lee el dato recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
  }
}
void write_Data_EEPROM(int ctrl) {          // Función que escribe los valores recibidos por el puerto serial principal en la memoria EEPROM del Arduino.

  switch (ctrl)
  {
    case eeCodeQR_Array:
      EEPROM.put( eeAddress0, codigoQR_Recarga );                    // Escribe el dato recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
    case eeValueRecarga_Str:
      EEPROM.put( eeAddress1, valueRecarga_Array );                  // Escribe el dato recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
    case eeValueRecarga_Long:
      EEPROM.put( eeAddress2, valueRecargaLong );                    // Escribe el dato recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
    case eelicenseActive_Bool:
      EEPROM.put( eeAddress3, licenseActive );                       // Escribe el dato recibido y lo guarda en la memoria NO volatil EEPROM del Arduino a utilizar.
      break;
  }
}
```

Anexo 2

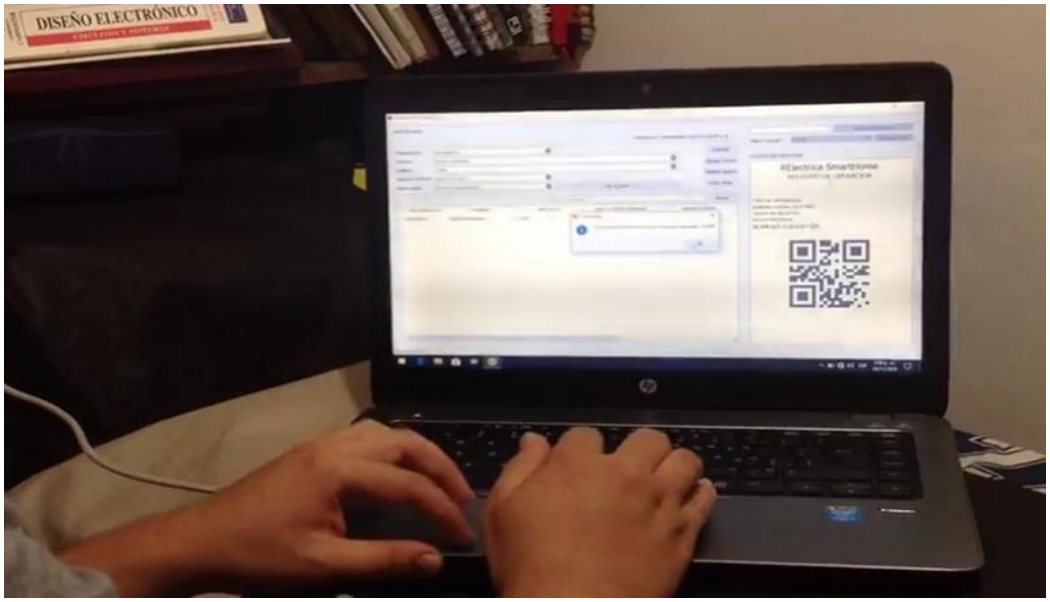


Anexo 2 figura 1 vista de los componentes utilizados para la fabricación del medidor



Anexo 2 figura 2 vista de las conexiones del medidor

Anexo 2 figura 3 Aspecto final del medidor de energía prepago.



Anexo 2 figura 4 vista del aplicativo que genera los códigos Qr.

Anexo 2 figura 5 puesta en funcionamiento del medidor.

Anexo 2 figura 6 medidor en funcionamiento.