UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad al servicio del Pueblo

UNIDAD ACADÉMICA DE INGENIERÍAS, INDUSTRIAS Y CONSTRUCCIÓN

CARRERA DE INGENIERÍA ELÉCTRICA

Implementación de un sistema SCADA en Matlab para monitorizar parámetros solares

TRABAJO DE INVESTIGACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ELÉCTRICO

AUTOR: PAUL SANTIAGO LOJANO ILLESCAS

DIRECTOR: ING. JAVIER BERNARDO CABRERA MEJÍA MSc.

MATRIZ CUENCA

2017

I. DECLARACIÓN

Yo, Paul Santiago Lojano Illescas declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento; y eximo expresamente a la Universidad Católica de Cuenca y a sus representantes legales de posibles reclamos o acciones legales.

La Universidad Católica de Cuenca puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y la normatividad institucional vigente.

Paul Santiago Lojano Illescas

II. CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Paul Santiago Lojano Illescas bajo mi supervisión.

_____

Ing. Javier Bernardo Cabrera Mejía MSc.

DIRECTOR

III. AGRADECIMIENTOS

En primer lugar, agradecer a Dios por la vida y la salud que me supo dar, para culminar esta meta propuesta.

A mis padres que supieron enseñarme los valores y las ganas de seguir adelante siempre a pesar de los obstáculos y retos que nos da el día a día.

A toda mi familia, que con sabias palabras y apoyo incondicional supieron alentarme en mis momentos de flaqueza y colocarme de nuevo en el camino correcto. Xavier, Fabián, Adriana, Fernando, Lucia, Claudia, Carlos y Jenny

A mi Profesor guía Ing. Javier Cabrera, quien con sus sabios conocimientos supo guiarme y fortalecer mis conocimientos, además de brindarme las herramientas necesarias para la realización de este trabajo.

A todos mis amigos que en su debido momento supieron aportar con todo cuanto estuvo en sus manos, en especial a Miriam M. y Andrés C.

IV. DEDICATORIA

Para las personas que dieron sentido a mi vida como son mis padres Clara, José y de forma muy especial a mi amada esposa Fanny y a mi querida hija Aracely quienes supieron comprenderme y sacrificar todo el tiempo que dedique a mis estudios y perderme esos momentos únicos. Gracias por estar a mi lado en las buenas y en las malas, de este camino recorrido.

V. ÍNDICE DE CONTENIDOS

VI. LISTA DE FIGURAS

VII. LISTA DE TABLAS

VIII. LISTA DE ANEXOS

IX. RESUMEN

Debido al gran crecimiento de las energías alternativas, como medio para mitigar las altas emisiones contaminantes en el ambiente. Se ha visto conveniente monitorizar los parámetros solares mediante un sistema SCADA desarrollado en Matlab y una tarjeta de adquisición de datos como es Arduino, para visualizar en tiempo real las variaciones que tiene el panel solar las 24 horas de día entre sus niveles de voltaje, corriente y potencia entregados, generando de esta manera un aporte que reduce el costo de los equipos empleados para estas nuevas tecnologías, haciendo que sea mucho más accesible para usuarios que anteriormente no podían optar por estas tecnologías debido a su estado económico.

Palabras clave: SCADA, ARDUINO, PANEL SOLAR, ENERGÍA FOTOVOLTAICA, ADQUISICIÓN DE DATOS.

X. ABSTRACT

Due to the great growth of alternative energies, as a means to mitigate the high polluting emissions in the environment. It has been convenient to monitor the solar parameters through a SCADA system developed in Matlab and a data acquisition card such as Arduino, to visualize in real time the variations that the solar panel has 24 hours a day between its voltage levels, current and delivered power, generating in this way a contribution that reduces the cost of the equipment used for these new technologies, making it much more accessible for users who previously could not opt for these technologies due to their economic status.

Keywords: SCADA, Arduino, solar panel, photovoltaic energy, data acquisition

CAPITULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

La energía eléctrica se encuentra presente en todos los aspectos que el ser humano ha generado o inventado para mejorar su forma de vida, con ello ha llegado a un alto nivel de industrialización y comunicación global, haciendo que todos se encuentre enlazados de forma permanente, pero todo este avance nos ha costado un precio muy alto, de tal manera que el medio ambiente se vea amenazado por las altas demandas de recursos que esta energía requiere para su producción, llevándonos a medios de producción que actualmente son insostenibles si hablamos de forma ambiental.

No obstante, el crecimiento y demanda de la energía eléctrica es imparable por lo que se ha buscado medios alternativos de generación, que no extiendan la degeneración y destrucción de nuestro propio planeta, razón por la cual la Generación Distribuida es una salida viable para nuevas formas de generación renovables. Teniendo como objetivo acortar las distancias de generación y que el usuario pueda ser consumidor y generador al mismo tiempo, además es esto el respeto al medio ambiente. Estas tecnologías pueden ser, como la fotovoltaica, eólica, solar térmica, geotérmica entre otras. Sirviendo esto como base para futuros sistemas eléctricos.

Estas fuentes de generación renovables se han ido implantando en escalas grandes, pero no se ha tenido grandes resultados a nivel de usuarios o pequeñas escalas, en lo cual una de las grandes razones, son los costos de implantación y recuperación del capital invertido en ello, haciendo que se casi inaccesible estas tecnologías a los usuarios.

Aunque en la actualidad con los continuos avances tecnológicos se han ido mejorando el rendimiento de dichos sistemas de generación renovables, pero a pesar de esto los equipos de protección, medición y control de estos sistemas aún tienen precios elevados

Razón por la cual se plantea un sistema SCADA desarrollado en Matlab y una tarjeta de adquisición de datos Arduino. Para visualizar en tiempo real los parámetros de los paneles solares, ya sea el voltaje, corriente y potencia que el panel en un momento preciso se encuentra entregando al sistema. Donde la plataforma de Arduino realiza la lectura de los transductores y el Software realizado en Matlab presenta los datos en un entorno grafico amigable con el usuario y al mismo tiempo se puede realizar el almacenamiento de los datos mostrados para que posteriormente puedan ser procesados y analizados según convenga o lo requiera el usuario de este Sistema SCADA.

1.2 Generación Distribuida

Los sistemas de Generación Distribuida es la producción de energía eléctrica con medios mucho más pequeños que las plantas convencionales y que se encuentra lo más próxima a los centros de carga, además de esto permite interactuar de tal manera que se pueda comprar o vender (Comisión Nacional para el Uso Eficiente de la Energía, 2014). En los últimos tiempos los recursos renovables como la Eólica y Solar, se han incrementado de manera progresiva en los lugares de consumo. Naciendo de esta manera el concepto de Generación Distribuida (Álvarez Pelegry & Castro Legarza, 2014).

1.2.1 Aspectos generales.

1.2.1.1 Económicos.

Debido a que uno de los inconvenientes más grandes de la producción centralizada es el transporte de Energía hacia los centros de consumo, lo que representa costos de inversión, operación y mantenimiento muy elevados, razón por la cual la generación descentralizada con pequeños generadores, es una gran opción para proporcionar energía dentro del área local. (Cando, 2011)

1.2.1.2 Técnicos.

Dentro de los aspectos técnicos se podría encontrar sistemas de Generación Distribuida conectados a la red y los que están aislados a la red. (Cando, 2011)

1.2.1.3 Tecnológicos.

Los avances tecnológicos nos van ofreciendo cada vez mejores y más eficientes medios de generación, de manera miniaturizada y con técnicas de modularidad pudiendo ser instaladas en pocas horas según la fuente energética a utilizar. (Cando, 2011)

1.2.1.4 Ambientales.

Los acuerdos mundiales relacionados con la mitigación del efecto invernadero han generado una pauta para que la Generación Distribuida vaya creciendo y las energías limpias una salida para bajar las emociones de carbono a la atmosfera. Desarrollando tecnologías amigables con el medio ambiente. (Cando, 2011)

1.2.2 Tecnología fotovoltaica en la Generación Distribuida

La energía solar como fuente de generación es ya muy conocida en todo el mundo, con una estructura modular ya sea en serie o paralelo según sean los requerimientos, aunque los costos de los equipos necesarios para poner en marcha son altos. Pero para su generación la misma no produce contaminación ambiental, requiriendo un manteamiento mínimo para sus unidades. (Cando, 2011)

1.3 Sistemas SCADA

Un sistema SCADA (por sus siglas en ingles de Supervisory Control And Data Adquisition), se fundamenta en la automatización de procesos, permitiendo al ser humano interactuar con el proceso ya sea en la adquisición de datos, control o toma de decisiones. Todo esto se debe a la interfaz gráfica que estos sistemas ofrecen al usuario.

Los avances tecnológicos relacionados con la electrónica y software hacen posible que los sistemas SCADA se puedan irse integrando en todos los ámbitos relacionados a las actividades humanas.

Dentro de las grandes ventajas que ofrece, los más representativos serian detectar fallas, realizar diagnósticos preventivos y correctivos. Creando un conjunto de regalas que permiten al usuario comprender, asimilar y tomar decisiones en base a los datos mostrados.

1.4 MATLAB

El nombre de MATLAB proviene de Matrix LABoratory, debido a que sus gestiones en los procesos son con matrices (Holly, 2007). Es un software muy potente diseñado para el cálculo numérico, modelado y simulación, análisis y procesamiento de datos, entre otros. Muy ampliamente utilizado en el campo científico e investigativo por su serie de herramientas que tiene para resolver problemas comunes, además de incorporar librerías especializadas para la resolución de problemas puntuales.

Su entorno grafico es muy amigable con el usuario ayudando a que se pueda utilizar y visualizar sus funciones de forma rápida. (Arellano, 2013)

1.4.1 App Designer.

App Designer es una de las aplicaciones de Matlab, diseñadas para realizar aplicaciones similares a las que se solía hacer en GUIDE, pero este nuevo aplicativo cuenta con nuevos recursos como son: indicadores, lámparas, perillas e interruptores, etc. Además, su código generado es estructurado facilitando su desarrollo. (MATHWORS, 2017)

1.5 ARDUINO

Arduino es una plataforma gratuita o código abierto (open-source), la cual se fundamenta en hardware y software que son muy fáciles de usar tanto para principiantes como para nivel profesional. Utilizado en múltiples proyectos estudiantiles y científicos de bajo costo. Convirtiéndose en una de las herramientas más versátiles, debido a que se puede acoplar a la placa electrónica módulos o Shields que permiten controlar, supervisar y censar procesos. Además de esto también podemos mencionar que las tarjetas electrónicas Arduino son relativamente de bajo costo, su entorno de programación se pude ejecutar en las tres

plataformas más conocidas como son: Windows, Macintosh OSX y Linux. (Massimo Banzi, 2017)

1.5.1 IDE de Arduino.

El IDE (por sus siglas en inglés Integrated Development Environment) de Arduino es un software que permite escribir y cargar un código en la tarjeta electrónica de manera muy sencilla llamando a cada trabajo un Sketch. En la figura 2 se muestra el IDE de Arduino. (Massimo Banzi, 2017)

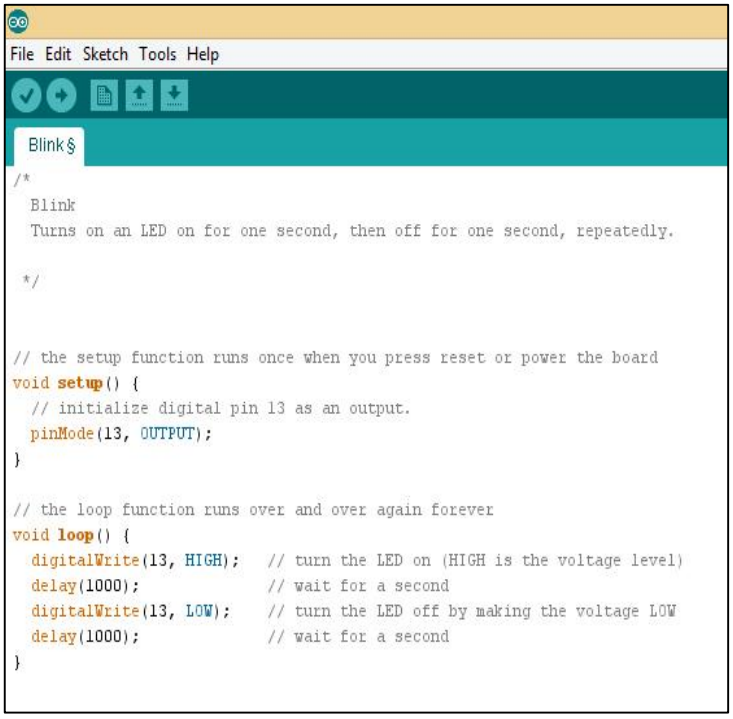


Figura 1 IDE de Arduino

Fuente: http://www.arduino.org/learning/getting-started/first-steps-with-arduino-ide

El sketch mostrado en la fotografía es el código por defecto que toda placa electrónica de Arduino viene cargada de fábrica y se trata de apagar y prender el Led del pin 13, el mismo que se encuentra integrado en la placa.

1.5.1.1 Sketch

El Sketch del Arduino está estructurado por dos funciones principales las cuales son el **Voidsetup()** que se encarga de inicializar el programa cargando las variables, estado de los pines y las librerías, esta función se ejecuta una sola vez cuando se energiza la placa electrónica y el **Voidloop()** que hace un ciclo infinito ejecutando el código realizado dentro del mismo. (Massimo Banzi, 2017)

1.5.1.2 Comunicación serial

Se puede realizar una comunicación serial entre la placa electrónica Arduino y un computador mediante un cable USB y los pines digitales Rx y Tx de Arduino, en donde es muy necesario establecer la velocidad de comunicación (baudios), las cuales pueden ser 300, 1200, 2400, 4800, 9600, entre otras. Pudiendo de esta manera enviar y recibir datos entre Arduino y otro dispositivo. (Massimo Banzi, 2017)

CAPITULO 2. COMPONENTES DEL SISTEMA SCADA

El módulo de adquisición de datos está formado por una serie de dispositivos, los cuales se irán describiendo uno a uno en este capítulo.

2.1 Arduino Mega 2560.

Es una placa electrónica con un micro procesador ATmega2560, que cuenta con 54 pines digitales de salida y entrada, 16 pines Analógicos, una conexión USB, un conector de alimentación. En la figura 2 se muestra la tarjeta electrónica. (Massimo Banzi, 2017)

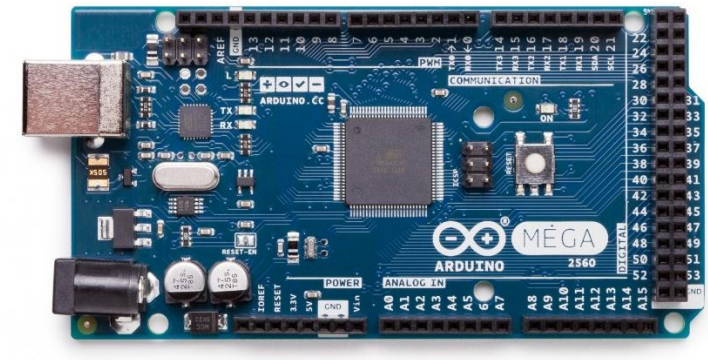


Figura 2 Arduino Mega 2560

Fuente: https://store.arduino.cc/usa/arduino-mega-2560-rev3

2.2 Sensor SCT-013

Es un sensor de corriente no invasivo, el cual por medio de un transformador nos suministra medidas proporcionales a las reales, esto lo hace mediante inducción magnética. Dispone de un núcleo ferromagnético partido, lo que permite abrirlo para introducir un cable conductor de electricidad. En la siguiente imagen se puede apreciar el sensor. (Llamas, 2007)

Figura 3 Sensor SCT-013

Fuente: https://www.luisllamas.es/arduino-sensor-corriente-sct-013/

La precisión del sensor puede ser del 1-2% por ello es conveniente que la pinza se encuentre correctamente cerrada.

2.2.1 Funcionamiento

Un transformador de corriente genera una corriente en el secundario proporcional a la corriente que pasa por el primario. Por tal razón se considera el cable que lleva la energía como primario, la pinza como núcleo magnético y el secundario es todo el sensor SCT-013.

Al hacer circular corriente por el cable (primario) se genera un flujo magnético en núcleo ferromagnético, el mismo a su vez genera una corriente en el secundario.

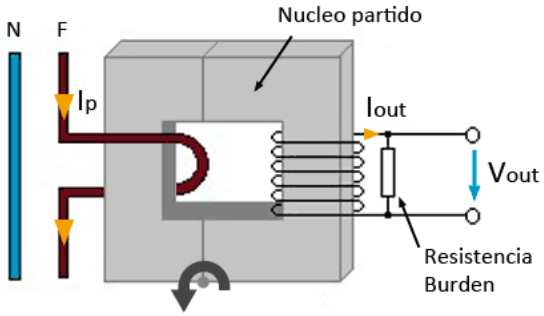


Figura 4 Funcionamiento del Sensor SCT-013

Fuente: https://www.luisllamas.es/arduino-sensor-corriente-sct-013/

2.2.2 ADS1115

La señal de salida del sensor SCT-030 no está dentro del rango de los valores aceptados por las entradas analógicas del Arduino Mega 2560. Por esta razón se utilizará como intermediario un ADS1115. El cual es un conversor externo de analógico a digital, permitiéndonos tener una resolución de 16 bits, conectándose en los pines SCL y SDA del Arduino (Llamas, 2007), en la imagen se puede apreciar la forma correcta de conectar el conversor. (Llamas, 2007)

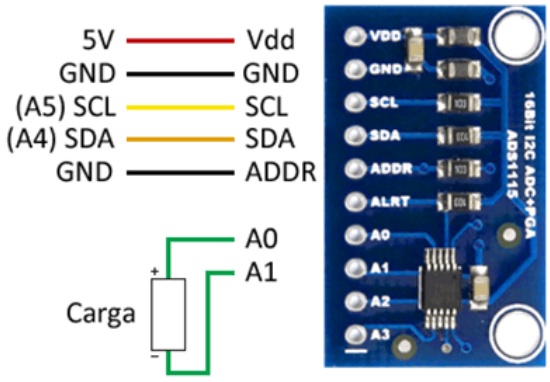


Figura 5 Conexión ADS115

Fuente: https://www.luisllamas.es/entrada-analogica-adc-de-16-bits-con-arduino-y-ads1115/

2.3 Divisor de Voltaje

Para poder censar el voltaje que está entregando el panel solar es necesario dividir el voltaje en dos partes, en la cual la menor parte tendrá un valor aproximado a 5v, esto se logra con un divisor de voltaje y con la siguiente formula.

$$Vsalida = \frac{R1}{R1+R2} * Ventrada \quad (1)$$

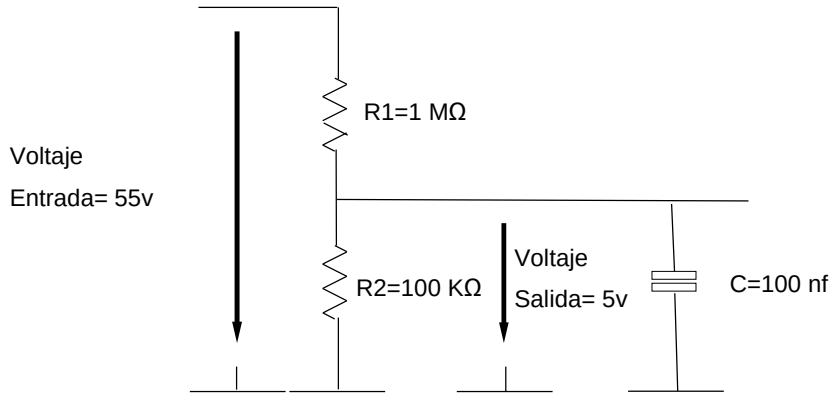


Figura 6 Divisor de Voltaje

Donde:

R1 es la primera resistencia

R2 es La Segunda resistencia

C es un capacitor

Vsalida es el voltaje de salida

Ventrada es el voltaje de la fuente

De esta manera podemos leer el voltaje, debido a que los pines de la placa Arduino como máximo pueden recibir 5 VC, caso contrario de superar este límite podríamos dañar permanentemente la placa electrónica.

CAPITULO 3. FUNCIONAMIENTO DEL SISTEMA

La estructura SCADA consta desde el voltaje, corriente y potencia que el panel fotovoltaico está entregando a una determinada carga, y los mismos que se puedan visualizar en la pantalla del computador en tiempo real los cambios que este tenga en su funcionamiento a lo largo del día. En la siguiente imagen se muestra la arquitectura de sistema.

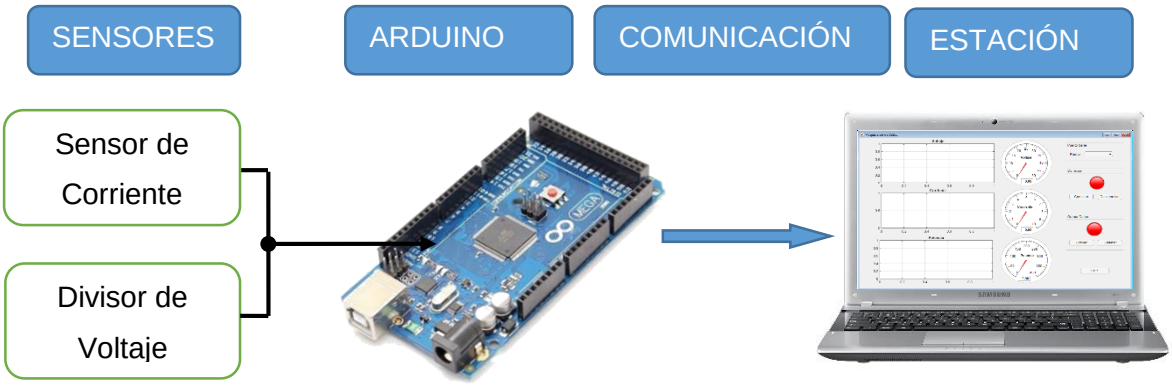


Figura 7 Arquitectura del sistema SCADA

3.1 Conexión

Para realizar este sistema SCADA se diseñó un hardware que nos ayude con la recolección de datos, que como se mencionó anteriormente está compuesta básicamente de un sensor de corriente y un divisor de voltaje, los mismo que se encuentran conectados en el Arduino mega 2560. En cuanto a la comunicación entre el computador y la placa Arduino, se lo realiza mediante el cable USB que viene incluido cuando se adquiere dicha tarjeta. En el anexo 1 se muestra un diagrama de la conexión de todos sus componentes.

3.2 Sketch en el ID de Arduino

El código creado en el ID de Arduino es bastante fácil, debido a que solo está destinado a recolectar valores de voltaje que ingresen a los pines analógicos A1 y A7, y al mismo tiempo convertirlos a los valores reales censados. Una vez realizado esto imprime los valores de voltaje y corriente en el puerto serial.

En el Anexo 2, se puede observar el código realizado en el sketch de Arduino.

3.3 App Designer

El algoritmo realizado en App Designer está basado en la adquisición de datos desde Arduino mediante la comunicación serial (MATHWORKS, 2017). Para luego procesarlos. Todo esto se visualizará en una ventana que cuenta con un entorno grafico muy amigable con el usuario. En el anexo 3 se muestra el código desarrollado.

Este código desarrollado en App Designer no tiene valides si no se le acopla a la interfaz gráfica donde se encuentra los botones, medidores y gráficos necesarios para su correcto funcionamiento, esto se debe a que App Designer crear el código de forma estructurada y solo deja a forma de edición los parámetros que considera que sean necesarios y el resto del código no se puede editar.

3.3.1 Interfaz grafica

Para hacer la interfaz de forma amigable se dividió en tres partes principales como son: Las gráficas, Medidores y Configuración como se muestra en la siguiente gráfica, de esta forma la interpretación de los datos obtenidos y mostrados se vuelve muy fácil.

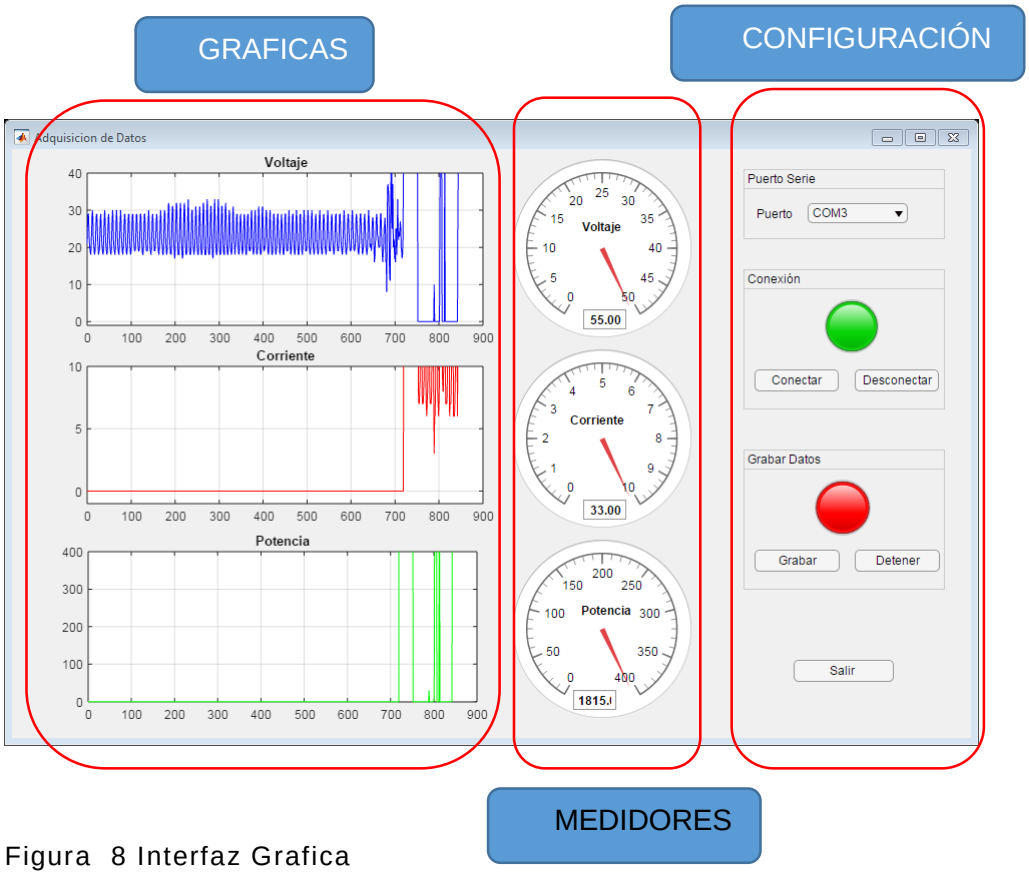


Figura 8 Interfaz Grafica

3.3.1.1 Graficas.

Las tres graficas están realizadas de tal manera que se puedan visualizar en tiempo real los datos adquiridos y visualizar en pantalla, con un margen de 1000 muestras. Las líneas de las gráficas están asignadas con colores diferentes, el color azul para el voltaje, el rojo para la corriente y el verde para la potencia.

3.3.1.2 Medidores.

Los medidores colocan la aguja en valor correspondiente según el valor leído en concordancia con los valores de las gráficas, además de esto se puede ver que se tiene unos visualizadores numéricos para cada parámetro leído.

3.3.1.3 Configuración.

En la configuración se tiene que seleccionar el puerto serial que está conectado el Arduino, aunque mediante código interno debe salir de forma automática el puerto al que corresponde el Arduino, posteriormente se debe dar clic en conectar para comenzar con la comunicación entre el computar y el Arduino, también se ha colocado un botón para desconectar la conexión serial entre los dos dispositivos.

Dentro de la configuración esta también el botón de guardar los datos leídos, donde nos pedirá la dirección en la que deseemos que se vayan guardando los datos. Creando de esta manera dos carpetas donde la una se guarda un archivo temporal de los datos y la segunda guarda los archivos diarios, esto nos ayuda a grabar dentro de una hoja de Excel los datos, creando un archivo diario para poder procesar según convenga por el usuario de esta aplicación. Si ya no desea continuar con la grabación de datos se puede detener dando clic en el botón parar, de esta manera solo quedara la parte de las gráficas y los medidores en funcionamiento.

Como última parte se ha colocado dos botones los cuales nos ayudan a cerrar completamente la aplicación y el otro botón nos da una venta donde consta el nombre del creador de la aplicación y su correo para preguntas posteriores que puede requerir el usuario final.

Se debe tomar en cuente las luces de aviso que tiene la aplicación, debido a que el color rojo indica que el proceso está detenido y el color verde indica que está en ejecución.

CAPITULO 4. PRUEBAS DEL SISTEMA SCADA

4.1 Medición de Voltaje

Para realizar estas pruebas se armó el divisor de voltaje en un Protoboard y se utilizó un panel solar (velleman serie: SOL5N), se utilizó este panel porque es el que se tenía disponible en ese momento. En las figuras siguientes se muestra el desarrollo de estas pruebas. Y también se ha divido en cuatro partes principales para poder dar una explicación más detallada. Además, se pondrá a consideración la lectura del multímetro y de la interfaz gráfica cuando se estaba realizando estas pruebas.

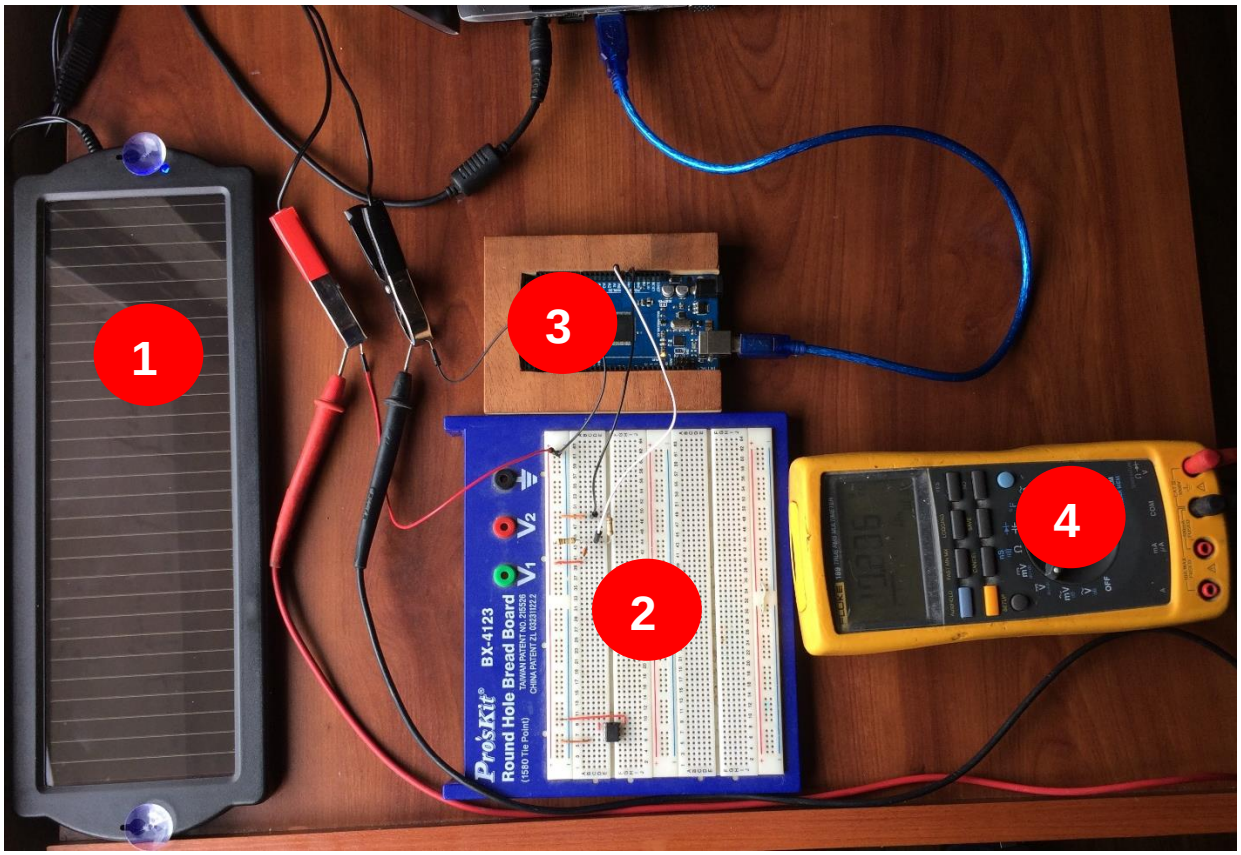


Figura 9 Fotografía general (Midiendo Voltaje)

4.1.1 Panel solar (1)

El panel solar de marca velleman con un voltaje de salida de 12V con una potencia máxima de 1.5W.

En las pruebas realizadas se ha comprobado que el panel puede entregar hasta 22V cuando está en el punto más óptimo de recepción de luz solar. Se debe tomar en cuenta que este ya se lo tenía y me sirvió de mucho para realizar las pruebas del sistema. En la imagen siguiente se muestra una fotografía del panel solar.

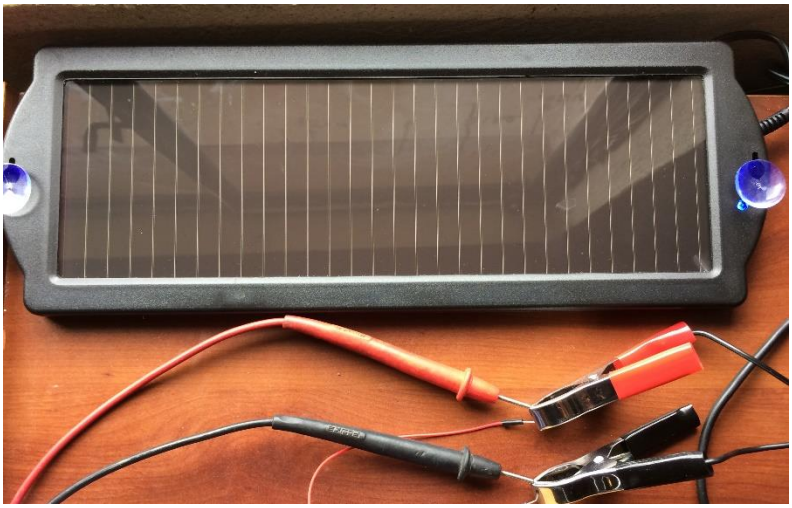


Figura 10 Panel Solar velleman

4.1.2 Divisor de voltaje (2)

El divisor de voltaje se lo realizo tal y como se lo propuso en el capítulo 2. Donde el cable de color blanco es el encargado de llevar la señal al pin analógico A0 para leer los valores de voltaje censados, los cables de color negro son la referencia a neutro y el cable rojo es el voltaje aportado por el panel solar.

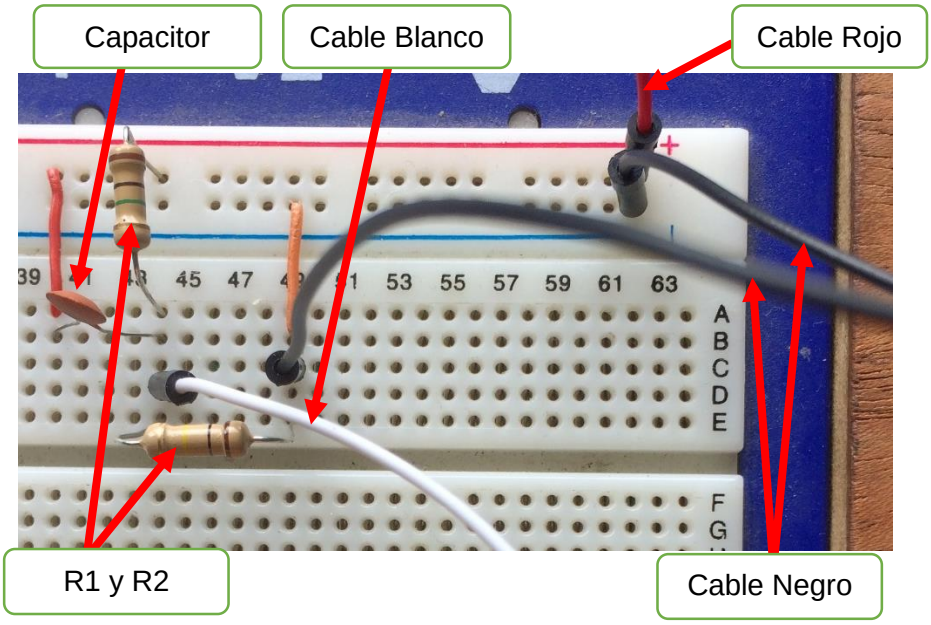


Figura 11 Circuito de prueba del divisor de voltaje

4.1.3 Arduino Mega 2560 (3)

En la placa electrónica Arduino se realizó la conexión según la imagen del anexo 1, quedando de la siguiente manera según muestra la imagen.

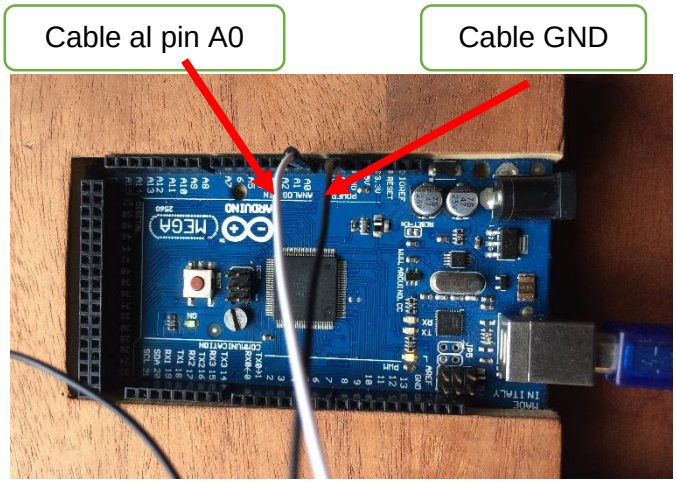


Figura 12 Conexión cables Arduino Mega 2560

4.1.4 Lectura Multímetro

La siguiente imagen muestra la lectura censada por el multímetro

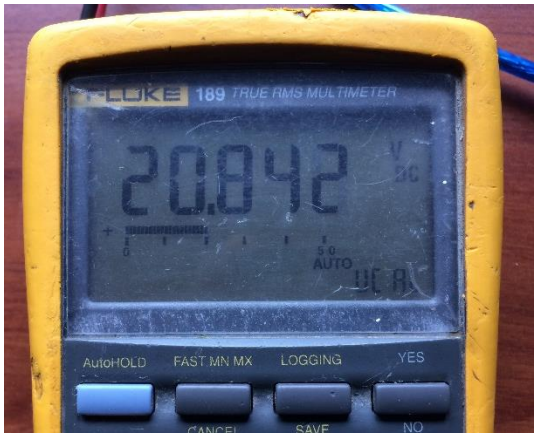


Figura 13 Lectura realizada por el multímetro

4.1.5 Interfaz Grafica

En la figura se muestra el valor recolectado por la aplicación, y con la gráfica de 1000 muestras.

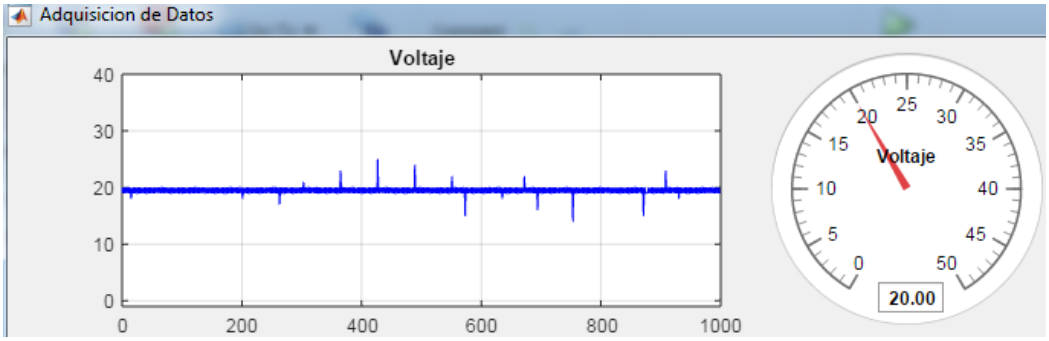


Figura 14 Valores de voltaje en la aplicación

4.2 Medición de Corriente

Para las pruebas de medición de corriente se realizó las conexiones en protoboard y como carga se utilizó un motor monofásico de ½ HP. En las siguientes figuras se muestra el desarrollo de esta prueba. Las cuales se han divido en 4 partes principales.

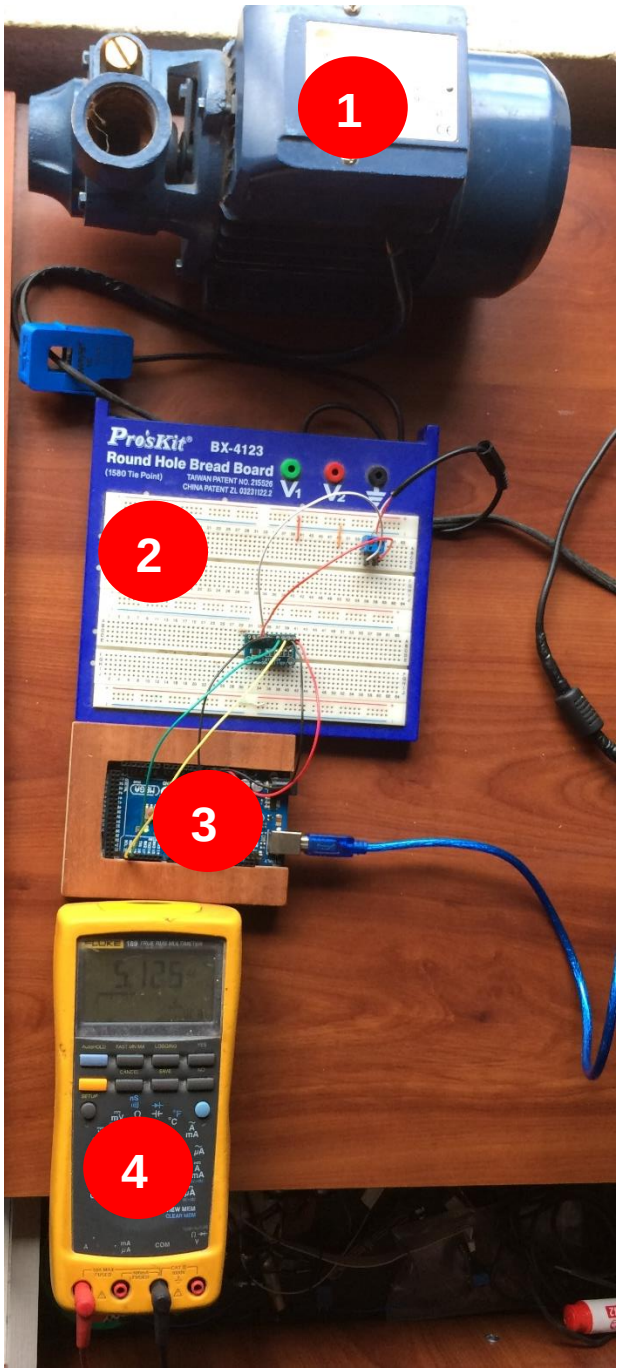


Figura 15 Medición de Corriente

4.2.1 Motor monofásico

Se utilizó este motor para las pruebas debido a que se lo tenía disponible en ese momento, el cual tiene una potencia de ½ HP con una corriente de 5.3 A, según datos de placa. En la siguiente imagen se muestra el motor y el sensor conectado en unos de los cables de alimentación.

Figura 16 Motor y SCT-030 conectados

4.2.2 Conexión ADS1115

Las conexiones de los cables se las realizo en el protoboard debido a que estaba en periodo de prueba y es más fácil corregir algún error y garantizar el correcto funcionamiento. En la imagen siguiente se muestra la conexión del ADS115. Las conexiones se realizaron según el diagrama presentado en la figura 5.

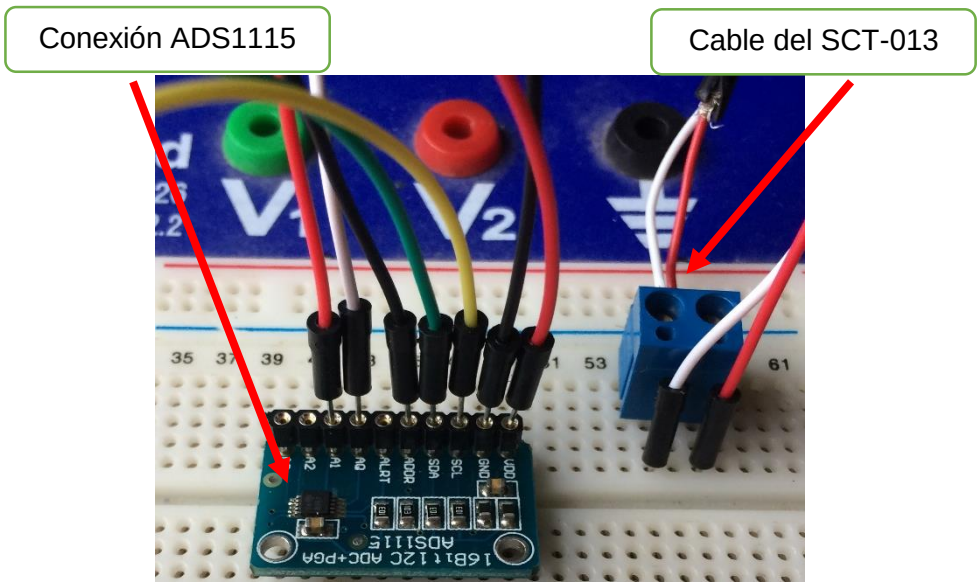


Figura 17 Conexión ADS1115

4.2.3 Conexión Arduino

Las conexiones en el Arduino se lo realizo según el diagrama propuesto en el anexo 1.

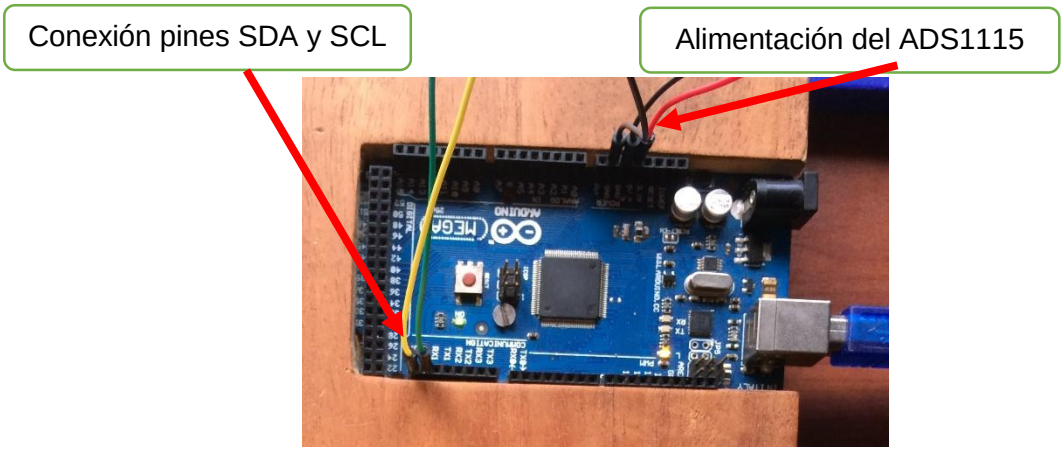


Figura 18 Conexión en Arduino ADS1115

4.2.4 Lectura Multímetro

En la siguiente imagen se muestra el valor de corriente censada.

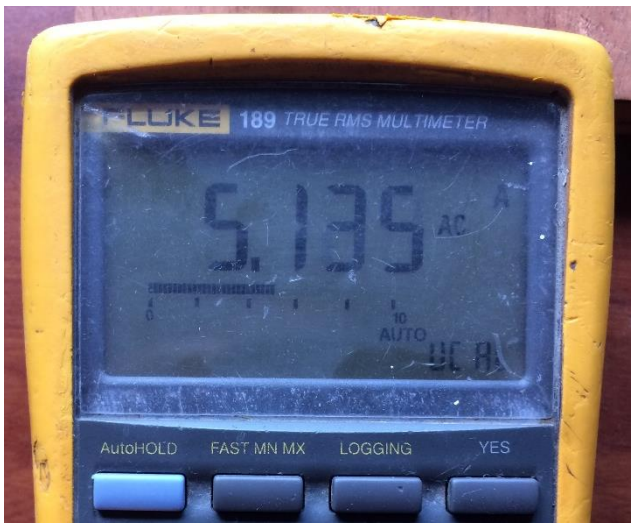


Figura 19 Lectura de Corriente

4.2.5 Interfaz Grafica

Los datos recolectados por la aplicación se muestran en la siguiente figura.

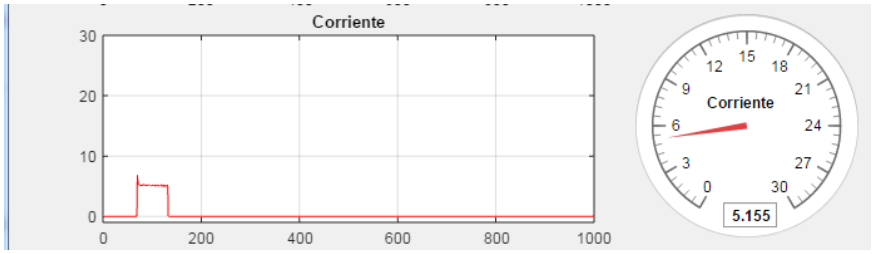


Figura 20 Valores de Corriente en la aplicación

CAPITULO 5. IMPLEMENTACIÓN

5.1 Diseño placa electrónica.

Para evitar cruces de cables y errores en las mediciones por cables sueltos u otro problema, se vio la necesidad de crear una placa electrónica que contenga todos los elementos propuestos de forma compacta y manejable. En las siguientes figuras se muestra el diseño de la misma en vista virtual 3D. realizada en Proteus (Labcenter, 2017)y la realizada en forma física.

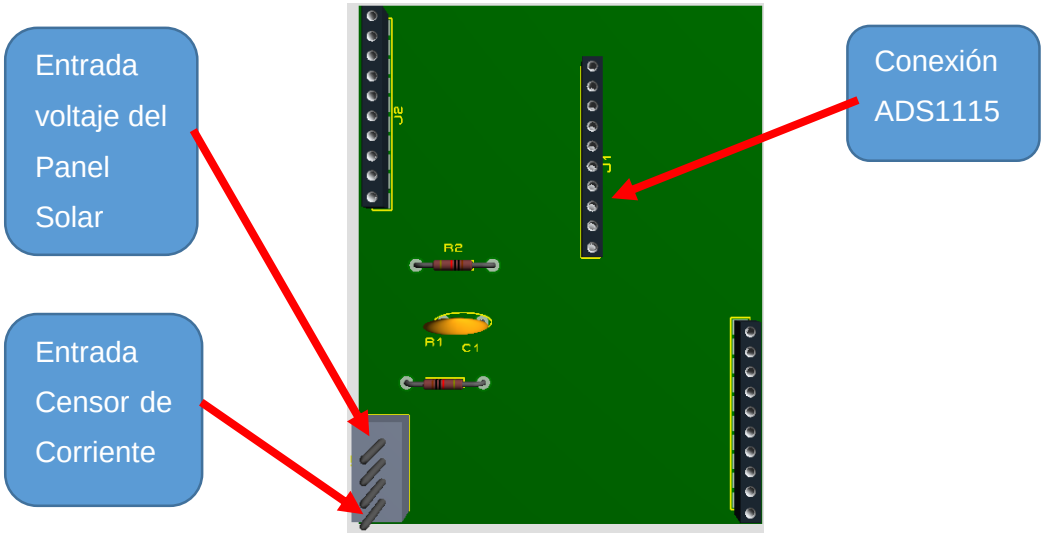


Figura 21 Placa Electrónica en Proteus

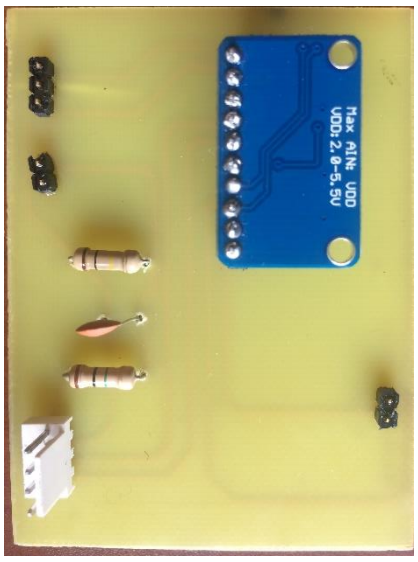


Figura 22 Placa Realizada

5.2 Diseño final interfaz grafica

Una vez revisado el funcionamiento del sistema y su código se procedió a mejorarlo en su presentación. En la figura se muestra el diseño final, cabe recalcar que solo cambio la parte grafica en cuanto a su código no ha cambiado en nada.

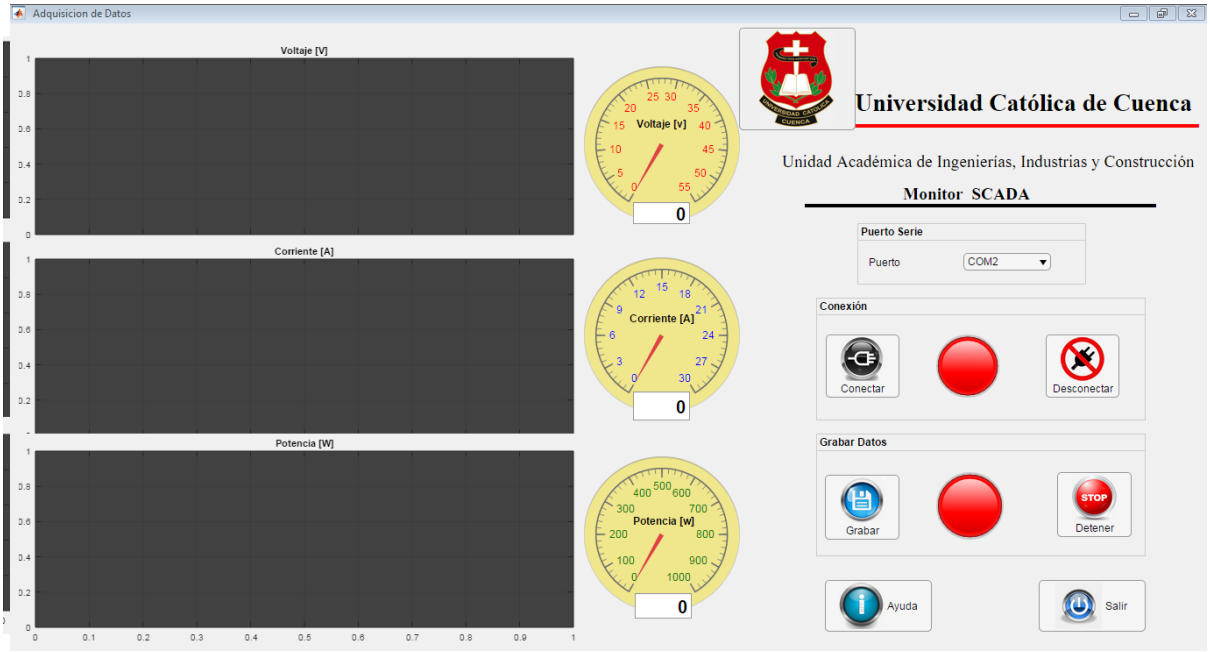


Figura 23 Diseño final interfaz grafica

5.3 Lectura de datos.

Para la lectura de datos se unió todos los componentes que son: Interfaz gráfica, placa electrónica, Arduino, Sensor de Corriente y los cables para censar el voltaje. Quedando de la siguiente manera como la siguiente imagen.

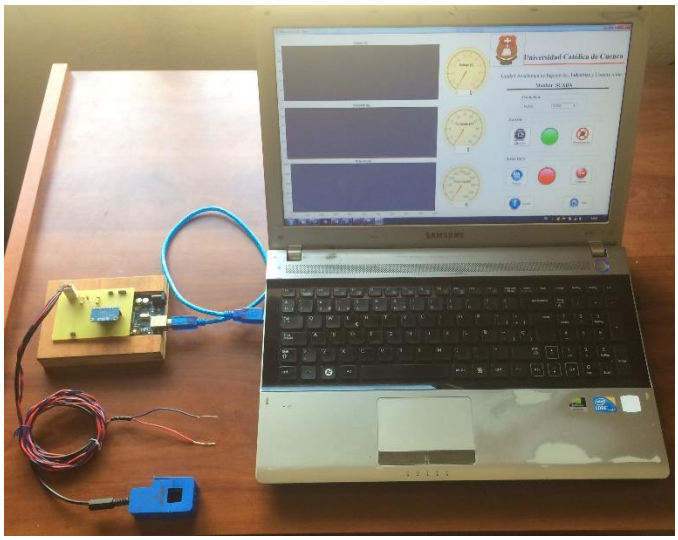


Figura 24 Sistema SCADA

Una vez unido todos los componentes del sistema SCADA se procedió a realizar las medidas de los parámetros solares de un panel solar y un banco de baterías que alimentan a tres focos, en la siguiente imagen se muestra este proceso.

Figura 25 Generador fotovoltaico puesto en marcha

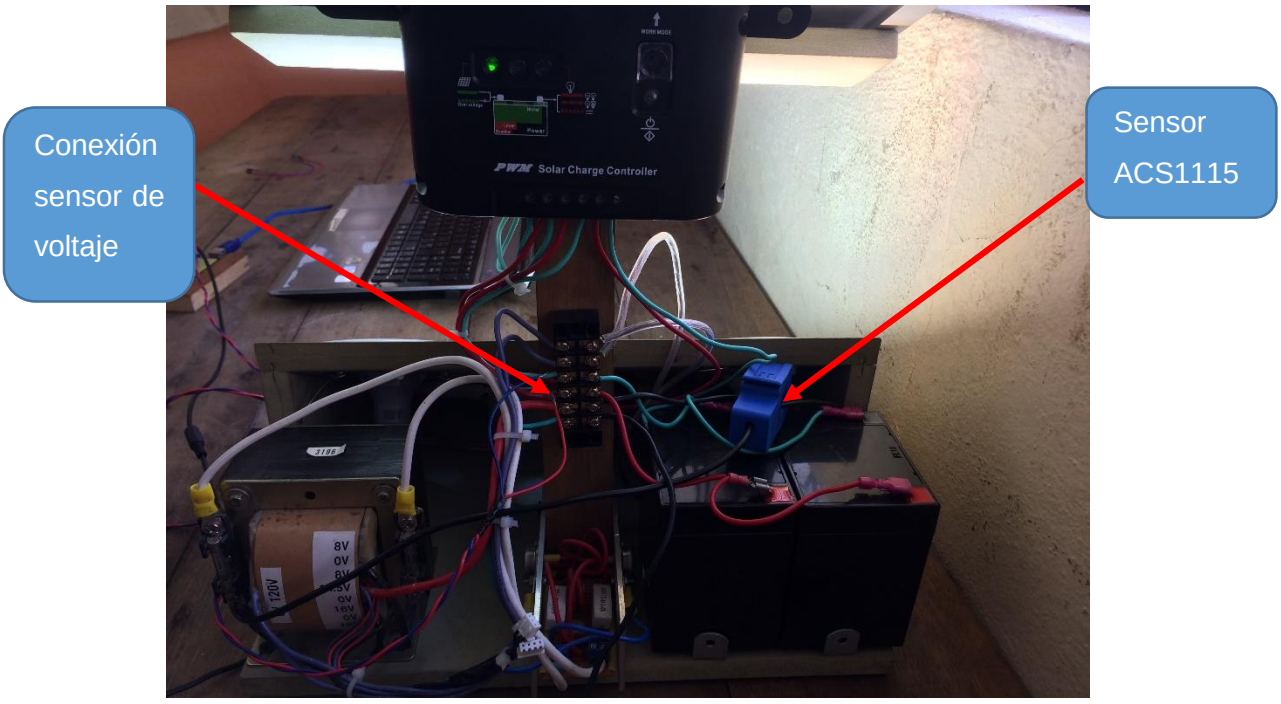


Figura 26 Conexión de sensores

5.4 Datos guardados

Después de realizar las conexiones pertinentes y poner en marcha el sistema SCADA se procedió a obtener datos, los cuales se fueron guardando en el computador, en la siguiente tabla se muestra una parte de los datos, donde la primera columna representa al voltaje, la segunda columna representa a la corriente y la tercera y última columna es la potencia.

Tabla 1 Datos Guardados

Voltaje	Corriente	Potencia
10,8482015	0,07245688	0,78602688
11,2887376	0,07416198	0,83719519
10,6830005	0,07745967	0,82750166
11,4539387	0,07582875	0,8685379
11,2887376	0,07416198	0,83719519
10,7931345	0,07416198	0,80044028
11,4539387	0,07582875	0,8685379
10,9032685	0,07416198	0,80860804
10,8482015	0,07416198	0,80452416
10,4076654	0,07582875	0,78920031
9,36139218	0,07416198	0,69425943
9,74686127	0,07416198	0,72284658
9,58166024	0,07416198	0,71059494
8,92085608	0,07416198	0,66158839
9,47152621	0,07416198	0,70242718
9,14112413	0,07416198	0,67792391
8,64552102	0,07416198	0,641169
8,92085608	0,07245688	0,64637743
8,42525297	0,07245688	0,61046757
8,53538699	0,07416198	0,63300124
8,42525297	0,07416198	0,62483348
8,26005193	0,07416198	0,61258185
8,37018595	0,07416198	0,6207496
8,31511894	0,07582875	0,63052511
7,98471686	0,07582875	0,60547113
8,09485089	0,07582875	0,61382246
8,03978388	0,07582875	0,6096468
7,92964985	0,07582875	0,60129547
8,70058803	0,07245688	0,6304175
10,5728665	0,07245688	0,76607696
10,9583356	0,07245688	0,79400685
11,2336706	0,07416198	0,83311131
10,6830005	0,07416198	0,79227252
11,4539387	0,07582875	0,8685379
11,0684696	0,07582875	0,83930826

5.5 Empaquetado

Luego de las pruebas y puesta en marcha del sistema se procedió a realizar el empaquetado del programa con todos los archivos necesarios, para su instalación y correcto funcionamiento.

Siendo incluidos los archivos con el código para cargar en el Arduino, así como también las librerías. De la mima manera el código desarrollado en App designer, cada uno en diferentes carpetas para su fácil localización. En la siguiente imagen se muestra el archivo instalador.

Figura 27 Instalador

Los archivos necesarios para cagar en el Arduino se copian en la carpeta de instalación, si la mima lo realizo por defecto los podrá encontrar en la siguiente dirección C:/Archivos de programa/Universidad Católica de Cuenca/SACADA/application

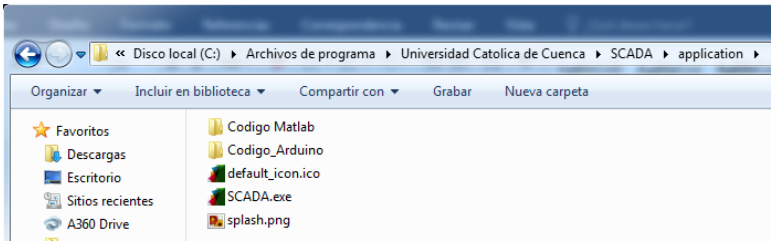


Figura 28 Ubicación de los Archivos

5.6 Instalación

A continua se describen los pasos necesarios para instalar el ejecutable creado en el empaquetado.

5.6.1 Ejecutar el instalador

Hacer doble clic o clic derecho en el archivo instalador para empezar con la ejecución.

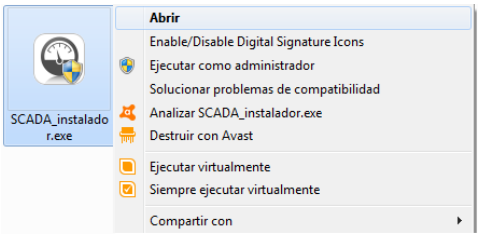


Figura 29 Ejecutar el Instalador

Nos va a preguntar si deseamos que el instalador realice cambios en el equipo y le decimos que sí.

5.6.2 Instalador

Se nos abre la primera venta con los datos más relevantes, leemos todo y clic en Next

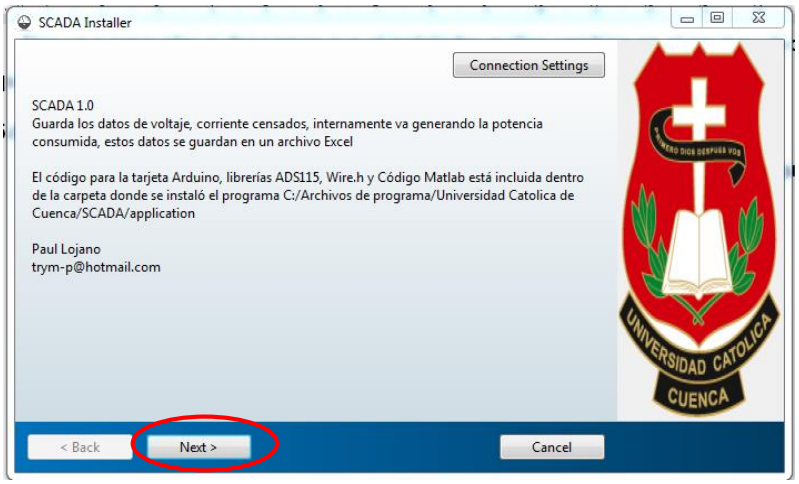


Figura 30 Ventana instalador

5.6.3 Opciones de Instalación

En esta ventana vamos a dejar la carpeta de instalación por defecto y marcamos la opción Add a shortcut to the desktop, lo que nos permite crear el acceso directo de nuestro programa en el escritorio. Posteriormente de damos clic en Next.

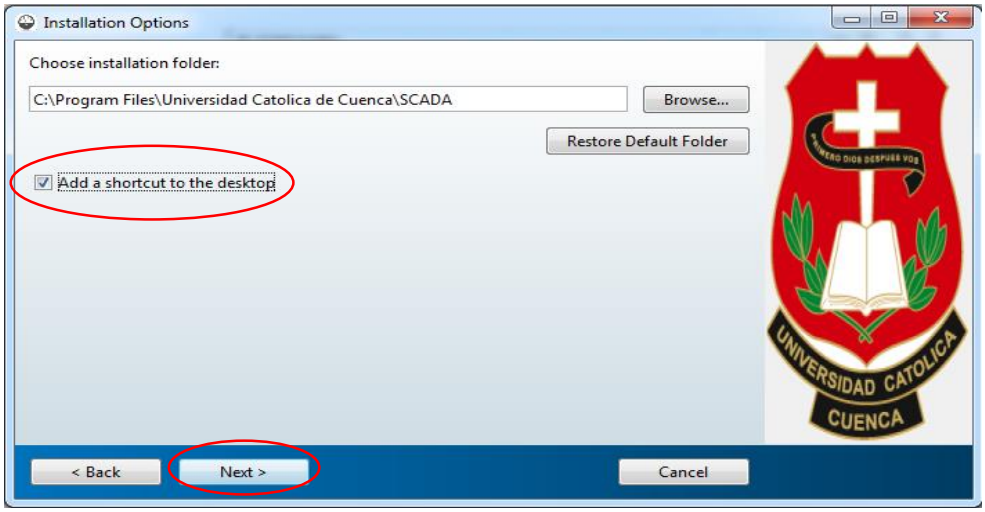


Figura 31 Opciones de instalación

5.6.4 Requerimientos del Software

La ventana nos muestra los requerimientos que necesita ser instalados para su correcto funcionamiento, así como también los derechos que tiene el programa, luego de leer el texto de damos clic en Next.

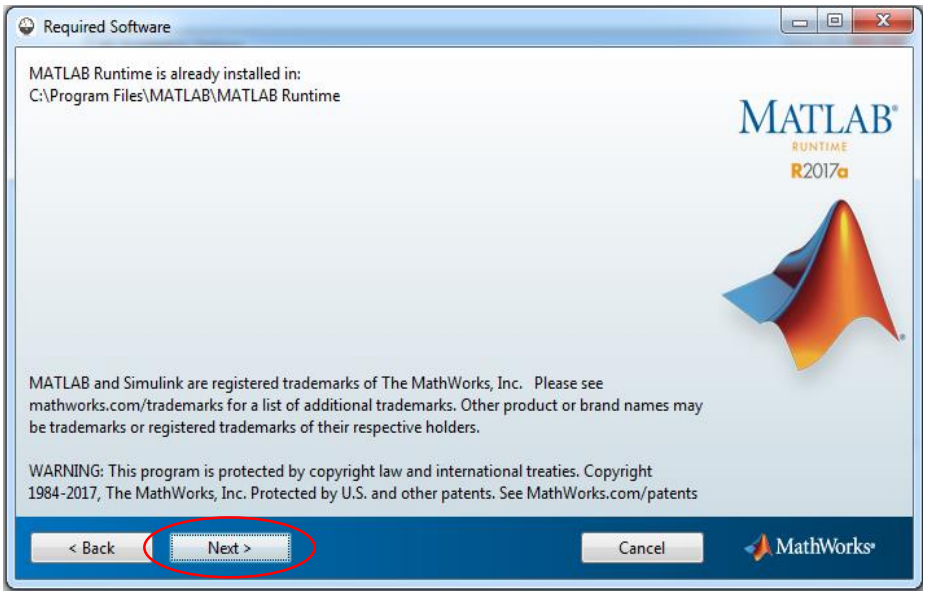


Figura 32 Requerimientos del software

5.6.5 Confirmación de la instalación

La venta muestra la confirmación de los parámetros de la instalación, revisamos todo y le damos clic en Install.

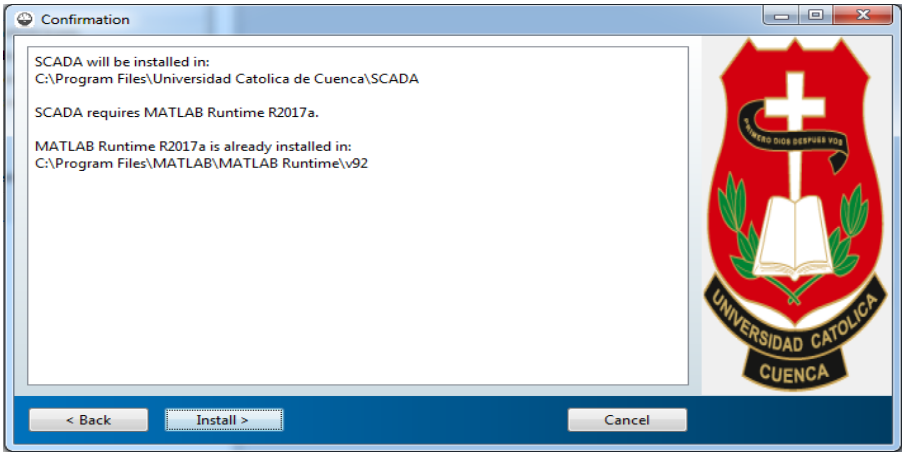


Figura 33 Confirmación de la instalación

5.6.6 Instalación

Luego de pasar por todo el proceso, nos muestra la pantalla de instalación del programa, en donde se puede observar el progreso en porcentaje de esta instalación. Una vez llegado al 100% el programa se encuentra instalado en el computador. Y el acceso directo lo entrara en el escritorio con el nombre de SCADA.

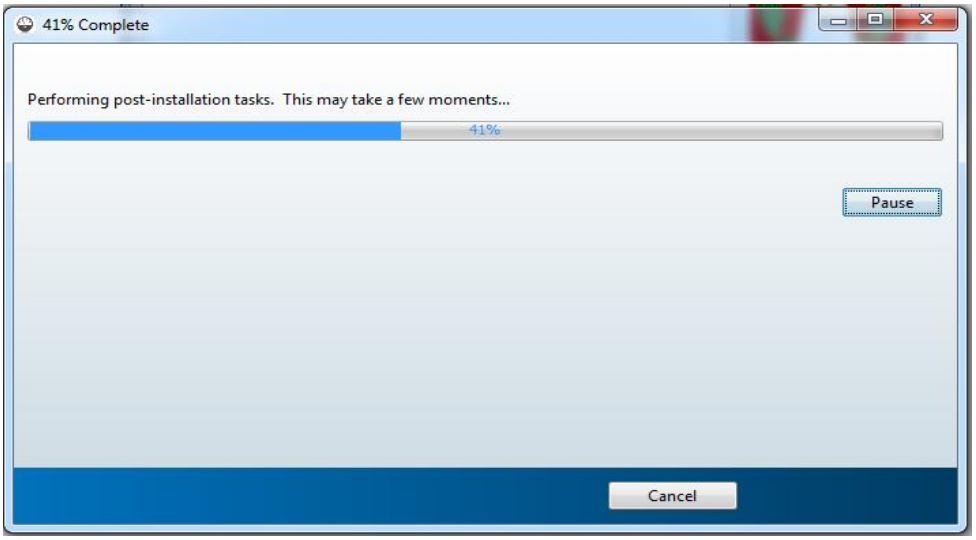


Figura 34 Progreso de la instalación

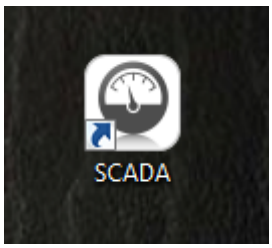


Figura 35 Acceso directo al programa

CONCLUSIONES

Una vez terminado el desarrollo de este trabajo de titulación se pudo aprender y recordar muchos aspectos aprendidos en la vida universitaria y personal. Donde el papel más importante se puede decir que es el manejo y tratamiento de la información mediante código o lenguaje de programación. Se tuvo que pasar por diversas etapas, cada una con sus problemas o inconvenientes, que había que ir superando conforme se avanzaba con este proyecto.

Como recopilación se puede decir que el primer problema que se tuvo en frente, es la adquisición de datos, donde los datos leídos se debían representar en una interfaz gráfica, inicialmente se pensó en realizar en la aplicación GUIDE de Matlab, pero por su bajo contenido gráfico se pasó a realizarlo en App Designer del mismo Matlab ya que el mismo ya contaba con muchas herramientas graficas que no tiene GUIDE, superado el primer inconveniente, se presente el segundo, que es graficar los datos obtenidos, en donde se pudo observar que la forma de escribir el código en App Designer es bastante diferente a su antecesor GUIDE, lo que tomo mucho tiempo en acoplarse a la nueva forma de escribir el código, a pesar de todo esto se pudo realizar el código para graficar los datos y como tercer y último problema a resolver, fue la perdida de datos obtenidos mediante el sensor de corriente específicamente, en donde el rango de resolución de los datos leídos era de 1 Amperio, con esto quiero decir que los datos de los miliamperios casi eran obviados, por los que se tomó la decisión de adquirir un convertidor de analógico a digital antes de que los datos sean leídos por el Arduino. Esto no ayudo a mejorar la resolución a 16 bits, con lo que se pudo resolver el problema. Cabe recalcar que todos los problemas presentados en el transcurso de este trabajo se los resolvió mediante prueba y error. Realizando una mejora continua a lo realizado.

Finalmente se obtuvo como producto final el sistema SCADA de adquisición de datos para medir los parámetros eléctricos de un Panel Solar.

RECOMENDACIONES

Conocer y manejar los lenguajes de programación es muy útil cuando se requieran realizar estos tipos de sistemas que unen el software y hardware al mismo tiempo, siendo este lenguaje los que los unen y los pone en entendimiento.

El crecimiento de la demanda energética hace que los sistemas de generación distribuida sean una solución viable para este problema, así como también, es necesario monitorear los parámetros de estas tecnologías, por esta razón los sistemas SCADA son muy útiles en estos casos.

La utilización de sistemas de Medición y adquisición de datos son una alternativa viable para dar seguimiento al consumo diario de energía eléctrica en nuestros hogares, contribuyendo de esta manera a la eficiencia energética.

Tomar en cuenta que este sistema de Adquisición de datos tiene sus limitantes en cuanto a la corriente que es de treinta amperios y el voltaje a ser censado es de 55 voltios en continuo, no se debe sobrepasar estos límites debido a que esto puede acarear daños permanentes en los sensores y la Placa Arduino quedando el sistema obsoleto.

Antes de conectar los cables del sensor de voltaje es necesario conocer, que cable es el positivo y negativo del panel.

Los cables del sensor de voltaje se distinguen como: El rojo es positivo y el Azul es negativo. Se debe conectar según los especificado, caso contrario pondríamos en peligro la placa Arduino, debido a que es poner en cortocircuito el panel solar con la placa Arduino.

Si tiene problemas para localizar el puerto, donde se encuentre conectado la placa Arduino diríjase a la siguiente página donde le guiaran paso a paso (https://technet.microsoft.com/es-es/library/cc754081(v=ws.11).aspx) la forma de ingresar al administrador de dispositivos y seleccione puertos CON LPT donde se mostrara el COM que está asignado a la placa.

Si por erro desconecta la placa Arduino del computador, cierre el programa y desconecte la placa Arduino. Vuelva a conectar la placa Arduino y ejecute el programa. Por defecto el programa reconoce el puerto COM automáticamente, si desconecto por erro la placa, el programa no puede reconocerlo automáticamente, así que tendrá que elegir del listado desplegable el puerto COM que corresponda. Podrá verificar el puerto asignado en el Administrador de dispositivos.

REFERENCIAS BIBLIOGRÁFICAS

[1]. Arellano, M. A. (2013). *MATLAB & SIMULINK PARA INGENIERIA NIVEL 1.* Lima.

[2]. Cando, J. A. (2011). *Estudio Técnico - Económico sobre la implementación de generacion distribuida en el sistema eléctrico Ecuatoriano.* Quito.

[3]. Holly, M. (2007). *MATLAB para ingenieros.* Mexico: PEARSON.

[4]. Labcenter. (24 de noviembre de 2017). *Labcenter* . Obtenido de https://www.labcenter.com/

[5]. Llamas, I. L. (12 de 10 de 2007). *Luisllamas*. Obtenido de https://www.luisllamas.es/entrada-analogica-adc-de-16-bits-con-arduino-y-ads1115/

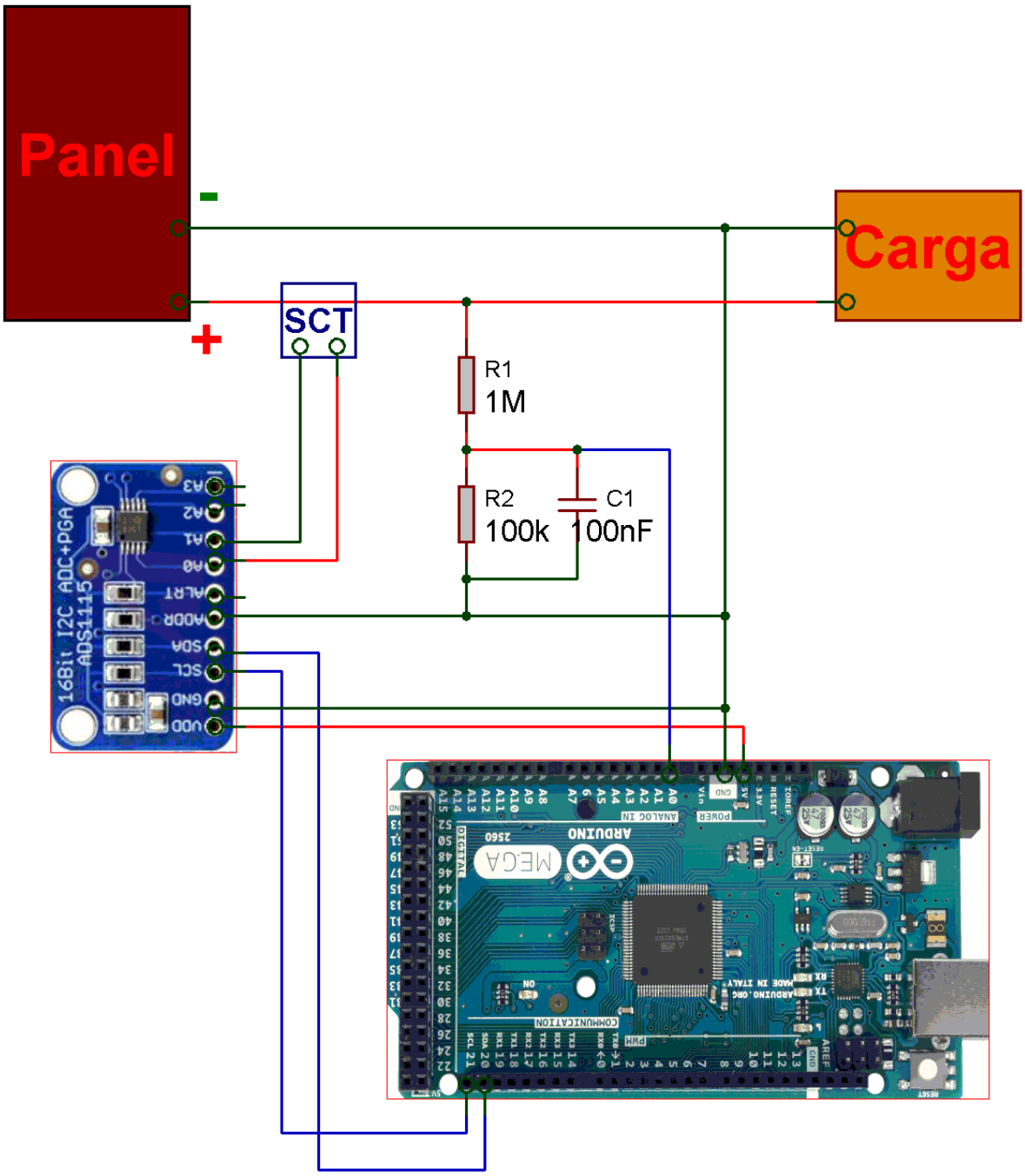
[6]. Massimo Banzi, D. C. (2017). *ARDUINO*. Obtenido de https://www.arduino.cc/

[7]. MATHWORKS. (11 de julio de 2017). *MATLAB Answers*. Obtenido de https://es.mathworks.com/matlabcentral/answers/276824-how-to-use-bytesavailablefcn-for-serial-communication-in-app-designer

[8]. MATHWORS. (10 de Agosto de 2017). *MATHWORS*. Obtenido de https://es.mathworks.com/products/matlab/app-designer.html

ANEXOS

Anexo 1 Diagrama del Hardware diseñado



Anexo 2 Sketch Arduino

```
//////////////////////////////////////VOLTAJE//////////////////////////////////////////

const int Voltaje = A0;   // Pin de entrada
float ventrada = 0.0; //Valor leído del pin A0

//////////////////////////////////////CORRIENTE//////////////////////////////////////

#include <Wire.h>// Librería de cableado
#include <Adafruit_ADS1015.h> // librería del ADS
Adafruit_ADS1115 ads; // configuración para el ADS1115
const float MX30 = 30; // Máximo valor de corriente que puede ser leído
const float Ganancia = 0.0925F;// Ganancia

//////////////////////////////////////////////////////////////////////////////////
void setup()

{
  Serial.begin(9600); //Velocidad de comunicación
  ads.setGain(GAIN_TWO);// Referencia de voltaje censado
  ads.begin();// Inicia el ADS
  pinMode(Voltaje, INPUT); // Modo del Pin
}

void loop()

{
//////////////////////////////////CODIGO VOLTAJE//////////////////////////////////////

  float ventrada = analogRead(Voltaje); // valor leído pin analógico A0

//////////////////////////////////////////////////////////////////////////////////

//////////////////////////////////CODIGO CORRIENTE//////////////////////////////////

  float voltajeADS;
  float csalida;
  float sumADS = 0;
  for (int x = 0; x < 250; x++) {
    Toma una muestra de 250 datos
    voltajeADS = ads.readADC_Differential_0_1() * Ganancia; // Lee el valor del ADS
    csalida = voltajeADS * MX30;
    csalida /= 250.0;
    sumADS += sq(csalida);

  }
  //////////////////////////////////////////////////////////////////////////////////

  Serial.println(ventrada); // imprime el valor del voltaje en el puerto serial
  Serial.println(sumADS);//Imprime el valor de la corriente
}
```

Anexo 3 Código App Designer Matlab

```
classdef GraficasVAP < matlab.apps.AppBase

    % Properties that correspond to app components

    properties (Access = public)

        Grafico           matlab.ui.Figure                  % Adquisi...

        axesVoltaje       matlab.ui.control.UIAxes          % Voltaje

        axesCorriente     matlab.ui.control.UIAxes          % Corriente

        axesPotencia      matlab.ui.control.UIAxes          % Potencia

        Panel             matlab.ui.container.Panel         % Puerto ...

        LabelDropDown     matlab.ui.control.Label           % Puerto

        Selport           matlab.ui.control.DropDown        % , COM0,...

        Panel2            matlab.ui.container.Panel         % Conexión

        Empesar           matlab.ui.control.Button          % Conectar

        Desconectar       matlab.ui.control.Button          % Descone..

        LedConectar       matlab.ui.control.Lamp

        Panel3            matlab.ui.container.Panel         % Grabar ...

        Grabar            matlab.ui.control.Button          % Grabar

        Detener           matlab.ui.control.Button          % Detener

        LedGuardar        matlab.ui.control.Lamp

        ValueVoltaje      matlab.ui.control.Gauge           % [0 50]

        ValueCorriente     matlab.ui.control.Gauge           % [0 10]

        ValuePotencia      matlab.ui.control.Gauge           % [0 400]

        LabelNumericEditField  matlab.ui.control.Label           % Voltaje

        Valv              matlab.ui.control.NumericEditField % [-Inf Inf]

        LabelNumericEditField2 matlab.ui.control.Label        % Corriente

        Vala            matlab.ui.control.NumericEditField % [-Inf Inf]
```

```
        LabelNumericEditField3 matlab.ui.control.Label      % Potencia
        Valp              matlab.ui.control.NumericEditField % [-Inf Inf]
        Salir             matlab.ui.control.Button           % Salir
        Ayuda             matlab.ui.control.Button           % Ayuda
    end
    methods (Access = private)
        % Code that executes after component creation
        function startupFcn(app)
            clc
            global puerto
            sp = instrhwinfo('serial'); %busca los puertos disponibles
            puerto = [];
            if~(isempty(sp.AvailableSerialPorts))
app.Selport.Items = sp.AvailableSerialPorts;% carga el puerto disponible
                puerto =sp.AvailableSerialPorts;
            end
        end
        % Selport value changed function
        function listSelPuerto(app, event)
            value = app.Selport.Value;
            global puerto
            puerto = value;  %LEE EL VALOR DEL SELECTOR DE PUERTO
        end
        % Empesar button pushed function
        function Conectar(app)
            clc
```

```
            global puerto sensor conex Stop guardar DATOS N ruta1 ruta2
creloj minutos gd

            if ~(isempty(puerto))

                clear sensor

                delete(instrfind({'Port'},{app.Selport.Value}));

       sensor=serial(app.Selport.Value);%Conecta con el puerto seleccionado

       sensor.BaudRate=9600; %velocidad de comunicación

        warning('off','MATLAB:serial:fscanf:unsuccessfulRead');

                fopen(sensor);% abre el puerto

                conex = 1;

       app.LedConectar.Color = [0.0314 0.8275 0.0314];% Color del indicador

                clc

            else

         helpdlg('Seleccione el puerto de conexión ','Error de conexión ');

            end


 %--------------------LECTURA---------------------------------

  fs=1;                                                 %Frecuencia de muestreo

  n= 1000;                                               %numero de datos

  v = zeros(n,1);                                       %Vector de Datos

  c = zeros(n,1);

  p = zeros(n,1);

  t = linspace(0,(n-1)/fs,n);

  N = 1;

  n1=1;

  i=0;

 Stop = 1;                            %VARIABLE DE CONTROL DE REPETICIÓN
```

```
DATOS=0;
resp2 = exist(ruta2);
voltajeValor=0;
corrienteValor=0;

if~(resp2 == 0)
    delete(ruta2);
end
app.axesVoltaje.YLim = [-1 40];
app.axesCorriente.YLim = [-1 10];
app.axesPotencia.YLim = [-1 400];
while (Stop==1)
   if n1<=n
    ventrada=fscanf(sensor,'%d')'; %LEER VOLTAJE DE ARDUINO
    csalida=fscanf(sensor,'%d')';%LEER CORRIENTE DE ARDUINO
    v(n1) = ventrada;
    DATOS(N,1) = v(n1);
    c(n1) = csalida;
    DATOS(N,2) = c(n1);
    p(n1) = v(n1)*c(n1);
    DATOS(N,3) = p(n1);
    n1=n1+1;
   else
    ventrada=fscanf(sensor,'%d')'; %LEER VOLTAJE DE ARDUINO
    csalida=fscanf(sensor,'%d')';%LEER CORRIENTE DE ARDUINO
    v(1:end-1) = v(2:end);
    v(end) = ventrada;
```

```
                DATOS(N,1) = v(end);

                c(1:end-1) = c(2:end);

                c(end) =csalida;

                DATOS(N,2) = c(end);

                p(1:end-1) = p(2:end);

                p(end) = v(end)*c(end);

                DATOS(N,3) = p(end);

               end

               app.ValueVoltaje.Value = DATOS(N,1);%grafica datos en el
medidor

               app.ValueCorriente.Value = DATOS(N,2);

               app.ValuePotencia.Value = DATOS(N,3);

               app.Valv.Value = DATOS(N,1);% Coloca los valores en los
caudros de valores

               app.Vala.Value = DATOS(N,2);

               app.Valp.Value = DATOS(N,3);

               %GRAFICA LOS DATOS

               N = N + 1;

               plot(app.axesVoltaje,t,v,'color','b')% garfica los datos

               plot(app.axesCorriente,t,c,'color','r')

               plot(app.axesPotencia,t,p,'color','g')

               drawnow

%------------GUARDAR DATOS---------------------------------------------------

                reloj = clock;

               if (guardar == 1)

                   if (creloj(5)~= reloj(5))

                      creloj(5) = reloj(5);

                      minutos = minutos +1;
```

```
        gd=1;
    end
    if (minutos ==2)
        resp2 = exist(ruta2);
         if~(resp2 == 0)
            delete(ruta2);
              xlswrite(ruta2,DATOS)
          else
             xlswrite(ruta2,DATOS)
         end

        minutos=0;
    end
    if((reloj(4) == 23)&(reloj(5)==59)&(gd==1))
        nombre=strcat('DatosVCP_',num2str(reloj(1)));
        nombre=strcat(nombre,num2str(reloj(2)));
        nombre=strcat(nombre,num2str(reloj(3)));
        nombre=strcat(nombre,'.xlsx');
        ruta=strcat(ruta1,nombre);
        datosT=xlsread(ruta2);
        xlswrite(ruta,datosT);
        clear datosT
        delete(ruta2);
        gd=0;
        N=1;
    end
end % end if
```

```
    end %end while
    %---------------------FIN LECTURA-------------------------------


end
% Desconectar button pushed function
function BtnDesconectar(app)
    global puerto sensor conex Stop


    if (conex == 1)
        delete(instrfind({'Port'},{puerto}))%eleminina el puerto
        app.LedConectar.Color = [1 0 0];


        conex = 0;
        Stop = 0;
    else
        msgbox('No hay conexión establesida')
    end
end
% Grabar button pushed function
function btnGrabar(app)
    global guardar ruta1 ruta2 creloj minutos gd DATOS N
    ruta = uigetdir('C:\');
    if(ruta ~= 0)
        ruta1 = strcat(ruta,'\ArchivosDiarios\');%crea ruta
        ruta2 = strcat(ruta,'\ArchivoDiario\DatosVCP.xlsx');
        resp1 = exist(ruta1);
```

```
            resp2 = exist(ruta2);
            if(resp1 == 0)
                mkdir(ruta,'ArchivosDiarios');
            end
            if(resp2 == 0)
                mkdir(ruta,'ArchivoDiario');
            end
            guardar = 1;
            app.LedGuardar.Color = [0.0314 0.8275 0.0314];
            app.Grabar.Enable = 'off';
            creloj = clock;
            minutos = 0;
            gd=1;
            clear DATOS
            N=1;
        else
            msgbox('Ruta no seleccionada')
        end
    end
    % Detener button pushed function
    function btnNograbar(app)
        global guardar
        guardar = 0;
        app.LedGuardar.Color = [1 0 0];
        app.Grabar.Enable = 'on';
    end
    % Salir button pushed function
```

```
        function SalirButtonPushed(app)
            closereq;
        end
        % Ayuda button pushed function
        function AyudaButtonPushed(app)
            msgbox('Ralizado por: Paul Lojano...Correo: trym-p@hotmail.com')
        end
    end
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create Grafico
            app.Grafico = uifigure;
            app.Grafico.Position = [100 100 957 588];
            app.Grafico.Name = 'Adquisicion de Datos';
            setAutoResize(app, app.Grafico, true)
            % Create axesVoltaje
            app.axesVoltaje = uiaxes(app.Grafico);
            title(app.axesVoltaje, 'Voltaje');
            app.axesVoltaje.Box = 'on';
            app.axesVoltaje.XGrid = 'on';
            app.axesVoltaje.YGrid = 'on';
            app.axesVoltaje.Position = [56 392 426 194];
            % Create axesCorriente
            app.axesCorriente = uiaxes(app.Grafico);
            title(app.axesCorriente, 'Corriente');
```

```
app.axesCorriente.Box = 'on';

app.axesCorriente.XGrid = 'on';

app.axesCorriente.YGrid = 'on';

app.axesCorriente.Position = [56 214 426 179];

% Create axesPotencia

app.axesPotencia = uiaxes(app.Grafico);

title(app.axesPotencia, 'Potencia');

app.axesPotencia.Box = 'on';

app.axesPotencia.XGrid = 'on';

app.axesPotencia.YGrid = 'on';

app.axesPotencia.Position = [50 16 426 192];

% Create Panel

app.Panel = uipanel(app.Grafico);

app.Panel.BorderType = 'line';

app.Panel.Title = 'Puerto Serie';

app.Panel.FontName = 'Helvetica';

app.Panel.FontUnits = 'pixels';

app.Panel.FontSize = 12;

app.Panel.Units = 'pixels';

app.Panel.Position = [730 498 201 70];

% Create LabelDropDown

app.LabelDropDown = uilabel(app.Panel);

app.LabelDropDown.HorizontalAlignment = 'right';

app.LabelDropDown.Position = [12 17 36 15];

app.LabelDropDown.Text = 'Puerto';

% Create Selport

app.Selport = uidropdown(app.Panel);
```

```
            app.Selport.Items = {'', 'COM0', 'COM1', 'COM2', 'COM3', 'COM4',
'COM5', 'COM6', 'COM7', 'COM8', 'COM9', 'COM10', 'COM11', 'COM12', 'COM13',
'COM14', 'COM15', 'COM16', 'COM17', 'COM18', 'COM19', 'COM20'};

            app.Selport.ValueChangedFcn = createCallbackFcn(app,
@listSelPuerto, true);

            app.Selport.Position = [63 15 100 20];

            app.Selport.Value = '';

            % Create Panel2

            app.Panel2 = uipanel(app.Grafico);

            app.Panel2.BorderType = 'line';

            app.Panel2.Title = 'Conexión';

            app.Panel2.FontName = 'Helvetica';

            app.Panel2.FontUnits = 'pixels';

            app.Panel2.FontSize = 12;

            app.Panel2.Units = 'pixels';

            app.Panel2.Position = [730 328 201 140];

            % Create Empesar

            app.Empesar = uibutton(app.Panel2, 'push');

            app.Empesar.ButtonPushedFcn = createCallbackFcn(app, @Conectar);

            app.Empesar.Position = [10 17 84 22];

            app.Empesar.Text = 'Conectar';

            % Create Desconectar

            app.Desconectar = uibutton(app.Panel2, 'push');

            app.Desconectar.ButtonPushedFcn = createCallbackFcn(app,
@BtnDesconectar);

            app.Desconectar.Position = [110 17 84 22];

            app.Desconectar.Text = 'Desconectar';

            % Create LedConectar

            app.LedConectar = uilamp(app.Panel2);
```

```
            app.LedConectar.Position = [80 55 54 54];
            app.LedConectar.Color = [1 0 0];
            % Create Panel3
            app.Panel3 = uipanel(app.Grafico);
            app.Panel3.BorderType = 'line';
            app.Panel3.Title = 'Grabar Datos ';
            app.Panel3.FontName = 'Helvetica';
            app.Panel3.FontUnits = 'pixels';
            app.Panel3.FontSize = 12;
            app.Panel3.Units = 'pixels';
            app.Panel3.Position = [730 148 201 140];
            % Create Grabar

            app.Grabar = uibutton(app.Panel3, 'push');
            app.Grabar.ButtonPushedFcn = createCallbackFcn(app, @btnGrabar);
            app.Grabar.Position = [10 17 85 22];
            app.Grabar.Text = 'Grabar';
            % Create Detener

            app.Detener = uibutton(app.Panel3, 'push');
            app.Detener.ButtonPushedFcn = createCallbackFcn(app,
@btnNograbar);
            app.Detener.Position = [110 17 85 22];
            app.Detener.Text = 'Detener';
            % Create LedGuardar
            app.LedGuardar = uilamp(app.Panel3);
            app.LedGuardar.Position = [70 53 56 56];
```

```
    app.LedGuardar.Color = [1 0 0];
    % Create ValueVoltaje
    app.ValueVoltaje = uigauge(app.Grafico, 'circular');
    app.ValueVoltaje.Limits = [0 50];
    app.ValueVoltaje.MajorTicks = [0 5 10 15 20 25 30 35 40 45 50];
    app.ValueVoltaje.Position = [500 400 178 178];
    % Create ValueCorriente
    app.ValueCorriente = uigauge(app.Grafico, 'circular');
    app.ValueCorriente.Limits = [0 10];
    app.ValueCorriente.MajorTicks = [0 1 2 3 4 5 6 7 8 9 10];
    app.ValueCorriente.Position = [500 210 178 178];
    % Create ValuePotencia
    app.ValuePotencia = uigauge(app.Grafico, 'circular');
    app.ValuePotencia.Limits = [0 400];
app.ValuePotencia.MajorTicks = [0 50 100 150 200 250 300 350 400];
    app.ValuePotencia.Position = [500 20 178 178];
    % Create LabelNumericEditField
    app.LabelNumericEditField = uilabel(app.Grafico);
    app.LabelNumericEditField.HorizontalAlignment = 'right';
    app.LabelNumericEditField.FontWeight = 'bold';
    app.LabelNumericEditField.Position = [569 503 39 15];
    app.LabelNumericEditField.Text = 'Voltaje';
    % Create Valv
    app.Valv = uieditfield(app.Grafico, 'numeric');
    app.Valv.ValueDisplayFormat = '%.2f';
    app.Valv.FontWeight = 'bold';
    app.Valv.Position = [570 408 43 20];
```

```
            % Create LabelNumericEditField2
            app.LabelNumericEditField2 = uilabel(app.Grafico);
            app.LabelNumericEditField2.HorizontalAlignment = 'right';
            app.LabelNumericEditField2.FontWeight = 'bold';
            app.LabelNumericEditField2.Position = [557 310 57 15];
            app.LabelNumericEditField2.Text = 'Corriente ';
            % Create Vala
            app.Vala = uieditfield(app.Grafico, 'numeric');
            app.Vala.ValueDisplayFormat = '%.2f';
            app.Vala.FontWeight = 'bold';
            app.Vala.Position = [570 218 43 20];
            % Create LabelNumericEditField3
            app.LabelNumericEditField3 = uilabel(app.Grafico);
            app.LabelNumericEditField3.HorizontalAlignment = 'right';
            app.LabelNumericEditField3.FontWeight = 'bold';
            app.LabelNumericEditField3.Position = [568 120 50 15];
            app.LabelNumericEditField3.Text = 'Potencia';
            % Create Valp
            app.Valp = uieditfield(app.Grafico, 'numeric');
            app.Valp.ValueDisplayFormat = '%.2f';
            app.Valp.FontWeight = 'bold';
            app.Valp.Position = [560 28 43 20];
            % Create Salir
            app.Salir = uibutton(app.Grafico, 'push');
        app.Salir.ButtonPushedFcn = createCallbackFcn(app, @SalirButtonPushed);
            app.Salir.Position = [826 16 100 22];
            app.Salir.Text = 'Salir ';
```

```
            % Create Ayuda
            app.Ayuda = uibutton(app.Grafico, 'push');
        app.Ayuda.ButtonPushedFcn = createCallbackFcn(app, @AyudaButtonPushed);
            app.Ayuda.Position = [712 16 100 22];
            app.Ayuda.Text = 'Ayuda';
        end
    end
    methods (Access = public)
        % Construct app
        function app = GraficasVAP()
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.Grafico)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.Grafico)
        end
    end
end
```