DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: "DISEÑO Y CONSTRUCCIÓN DE UN MÓDULO DIDÁCTICO DE BAJO COSTO, BASADO EN INSTRUMENTACIÓN VIRTUAL PARA LA REALIZACIÓN DE PRÁCTICAS EN ELECTRÓNICA GENERAL EN EL COLEGIO TÉCNICO DE BACHILLERATO DR. TRAJANO NARANJO I.".

AUTORA: ROCÍO IBETH PASTUÑA DOICELA

DIRECTOR: ING. JOSÉ BUCHELI

LATACUNGA

2018

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"DISEÑO Y CONSTRUCCIÓN DE UN MÓDULO DIDÁCTICO DE BAJO COSTO, BASADO EN INSTRUMENTACIÓN VIRTUAL PARA LA REALIZACIÓN DE PRÁCTICAS EN ELECTRÓNICA GENERAL EN EL COLEGIO TÉCNICO DE BACHILLERATO DR. TRAJANO NARANJO I."**, realizado por la señorita **Rocío Ibeth Pastuña Doicela**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a la señorita **Rocío Ibeth Pastuña Doicela** para que lo sustente públicamente.

Latacunga, 01 de Marzo de 2018

Ing. José Bucheli
DIRECTOR DEL PROYECTO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Yo, **Rocío Ibeth Pastuña Doicela,** con cédula de ciudadania No. **0503134413** declaro que este trabajo de titulación, **"DISEÑO Y CONSTRUCCIÓN DE UN MÓDULO DIDÁCTICO DE BAJO COSTO, BASADO EN INSTRUMENTACIÓN VIRTUAL PARA LA REALIZACIÓN DE PRÁCTICAS EN ELECTRÓNICA GENERAL EN EL COLEGIO TÉCNICO DE BACHILLERATO DR. TRAJANO NARANJO I."**, ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 01 de Marzo de 2018

Rocío Ibeth Pastuña Doicela
C.C.: 0503134413

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Yo, **Rocío Ibeth Pastuña Doicela,** autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca virtual de la Institución el presente trabajo de titulación **"DISEÑO Y CONSTRUCCIÓN DE UN MÓDULO DIDÁCTICO DE BAJO COSTO, BASADO EN INSTRUMENTACIÓN VIRTUAL PARA LA REALIZACIÓN DE PRÁCTICAS EN ELECTRÓNICA GENERAL EN EL COLEGIO TÉCNICO DE BACHILLERATO DR. TRAJANO NARANJO I."**, cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 01 de Marzo de 2018

Rocío Ibeth Pastuña Doicela
C.C.: 0503134413

DEDICATORIA

Quiero dedicar la presente investigación a mis amados padres, por su apoyo constante y desmedido, por impulsar mis sueños de ser una profesional útil a la sociedad, por poner todo su esfuerzo y sacrificio para darme educación y hacer de mi un ser humano con principios y valores, su infinito amor y ternura me impulsaron día a día a seguir adelante y a conseguir mis metas.

AGRADECIMIENTO

Quiero agradecer a todas aquellas personas que de una u otra manera han hecho posible concluir esta tesis. De manera especial:

A mis padres, que sin saberlo han sido el mayor impulso y el pilar que me ha sostenido incluso en los momentos más difíciles a lo largo de este proyecto. Por compartir conmigo largas horas de trabajo, con una actitud crítica, objetiva y constructiva; y sobre todo por motivarme a seguir siempre adelante.

Al Ing. José Bucheli, que más que un director de tesis se ha convertido en un amigo. Sus conocimientos, visión, inmenso empuje en el trabajo revisando línea a línea el borrador de esta tesis y su desempeño académico han sido altamente enriquecedores para el desarrollo de la misma.

A mis hermanos, por darme siempre muestras de apoyo y confianza al colaborar con mis anhelos a través de su cariño.

Finalmente, un agradecimiento a los docentes de la Carrera de Ingeniería en Electrónica e Instrumentación, por transmitirme sus enseñanzas y sobre todas las cosas demostrar su don de gente y su enorme sencillez y calidad humana.

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En la actualidad existe en el mercado muchos módulos didácticos para ensayos de Electrónica Básica, siendo el principal inconveniente los costos elevados, que salen del presupuesto de las Instituciones educativas públicas, razón por la cual muchas veces se hace imposible su adquisición, lo que genera una práctica deficiente de los estudiantes que posteriormente se verá afectada en el ámbito laboral. Mediante el diseño, construcción e implementación del módulo didáctico para el Laboratorio de Eléctrica y Electrónica del Colegio Técnico Dr. Trajano Naranjo, con la utilización de una tarjeta Arduino se ha conseguido agrupar funciones tales como: Multímetro, Osciloscopio, Entradas Digitales, Entradas analógicas, Fuentes AC/DC y Generador de Funciones; las mismas que están ligadas a una interfaz gráfica visualizadas en una PC con software libre. La construcción de este módulo didáctico de bajo costo, permitió agrupar instrumentos de medida, periféricos de entradas y salidas, fuentes de alimentación en Vcc y Vac, así como también un generador de funciones, para la implementación de circuitos básicos de Electrónica, garantizando el aprendizaje y fortaleciendo las competencias de los alumnos. La interacción de la placa Arduino con una interfaz gráfica, permite al estudiante interactuar de forma amigable y segura entre los equipos, elementos a ensayar y los instrumentos que conforman el módulo didáctico. La importancia fundamental de la construcción e implementación del módulo didáctico de Eléctrica y Electrónica en el Colegio Técnico Dr. Trajano Naranjo ayudó a cubrir el déficit de instrumentos de medición necesarios para realizar prácticas de Electrónica Básica, con lo que se garantiza el aprendizaje práctico de los estudiantes para que vayan con bases sólidas a la vida universitaria.

PALABRAS CLAVE:

- **MÓDULO DIDÁCTICO DE ELECTRÓNICA BÁSICA**
- **INTERFAZ GRÁFICA**
- **TARJETAS ARDUINO**

ABSTRACT

At present, many didactic modules exist in the market for basic electronics tests. The main inconvenience is the high costs, which come from the budget of public educational institutions, which is why it is often impossible to acquire them, which generates a deficient practice of students who will subsequently be affected in the workplace. With the design, construction and implementation of the didactic module for the Laboratory of Electrical and Electronics of the Technical College Dr. Trajano Naranjo, with the use of an Arduino card has been grouped functions such as: Multimeter, Oscilloscope, Digital Inputs, Analog Inputs, AC / DC Sources and Function Generator; The same ones that are linked to a graphical interface visualized in a PC with free software. The construction of this low-cost didactic module allowed the grouping of measurement instruments, input and output peripherals, Vcc and Vac power supplies, as well as a function generator for the implementation of basic electronic circuits, guaranteeing the learning and strengthening the competencies of students. The interaction of the Arduino board with a graphical interface allows the student to interact in a friendly and secure way between the equipment, elements to be tested and the instruments that make up the didactic module. The fundamental importance of the construction and implementation of the didactic module of Electrical and Electronics in the Technical College Dr. Trajano Naranjo helped to cover the deficit of measurement instruments necessary to perform basic electronics practices, thus guaranteeing the practical learning of the students to go with solid foundations to university life.

KEYWORDS:

- **BASIC ELECTRONICS DIDACTIC MODULE**
- **GRAPHIC INTERFACE**
- **ARDUINO CARDS**

CAPÍTULO I

FUNDAMENTO TEÓRICO

1.1. Módulos Didácticos de Electrónica Básica

En la actualidad el aprendizaje y la enseñanza se realizan mediante la estrecha interrelación de la parte teórica con la parte práctica, es por esto que hoy en día se busca métodos y dispositivos que integren los mismos, el uso de módulos didácticos garantiza que el estudiante pueda poner en práctica todo el conocimiento teórico adquirido a lo largo de su preparación estudiantil.

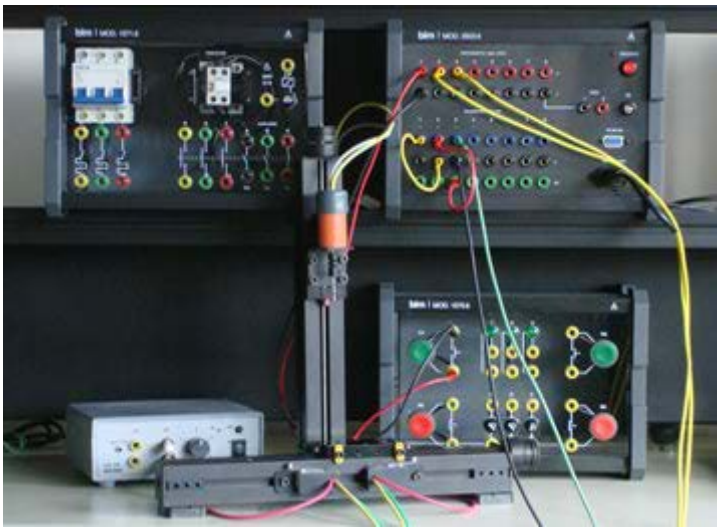

Figura 1. Módulo Didáctico
Fuente: (Técnica Didáctica, 2015)

Un módulo didáctico como su nombre lo indica debe ser o estar constituido de tal forma que facilite el aprendizaje e ilustre claramente las partes más importantes del tema en estudio. (Miñarro, 2006)

Entre las características principales de un módulo didáctico debe tenerse en cuenta las siguientes:

- **Facilitar el aprendizaje.-** Es decir el módulo debe poseer un lenguaje fácil y simplificado para que el estudiante que lo utilice se asocie fácilmente a él.

- **Interrelacionar el conocimiento teórico con el práctico.-** Un módulo didáctico debe estar basado en la parte teórica del tema de estudio e implementarlo hacia su parte práctica, para que el alumno comprenda y ponga en ejecución el conocimiento adquirido de forma teórica.
- **Visualizar.-** Como se trata de un módulo para el aprendizaje didáctico debe poseer una parte visual-gráfica para poder comprender de una manera real las características del tema que está siendo objeto de estudio.

Las ventajas de la implementación de un módulo didáctico sobre los métodos convencionales, son entre otros la facilidad de aprendizaje para el estudiante al ser un instrumento simplificado y práctico de la información teórica, la visualización gráfica plasmada en un computador donde evidencia el resultado de manipular diferentes elementos electrónicos y la facilidad que ofrece al alumno al permitir manipular un elemento electrónico y observar por sí mismo lo que esto provoca, conseguirá que el estudiante asimile el conocimiento en su memoria y sea mucho más difícil olvidarlo que si lo realizara únicamente de forma teórica.

1.2. Fuentes de Voltaje de Corriente Alterna y Corriente Continua

Cuando se conocía poco acerca del sistema eléctrico, el ser humano tenía que limitarse tan solo al uso de esa electricidad sin entenderla, pero como ahora es posible el enlace entre la conceptualización y la aplicación de los elementos que conforman el sistema eléctrico, se favorece a la comprensión de la estructura y funcionalidad de dicho recurso.

1.2.1. Corriente continua (CC)

Es el flujo continuo de electrones a través de un conductor entre dos puntos de distinto potencial, en ella las cargas eléctricas circulan siempre en la misma dirección, es decir, los terminales de mayor y de menor potencial son siempre los mismos. Aunque comúnmente se identifica la corriente continua con la corriente

constante (por ejemplo la suministrada por una batería), es continua toda corriente que mantenga siempre la misma polaridad (Nilson, 2005). La Figura 2 muestra la corriente continua.

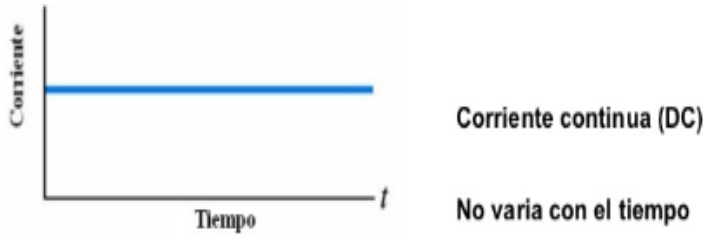

Figura 2. Corriente Continua
Fuente: (Nilson, 2005)

a. Fuentes de corriente continua

Un circuito eléctrico está formado por la asociación de una serie de elementos conductores que hacen posible el mantenimiento por su interior de una corriente eléctrica. Si los generadores producen una diferencia de potencial constante entre sus bornes o polos, la corriente producida será continua. Tal es el caso de las pilas y de las baterías (Hubert, 2006).

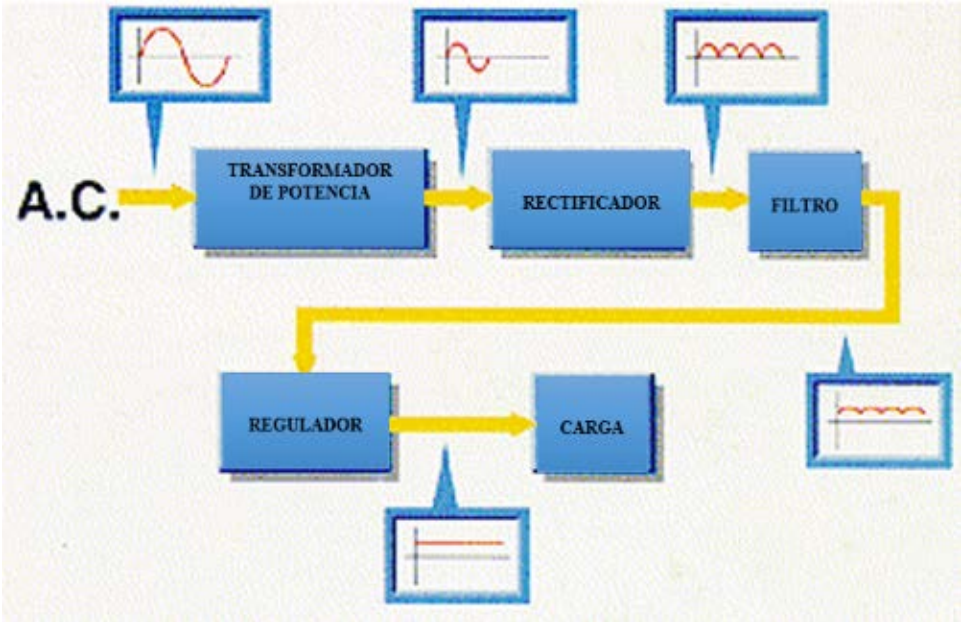

Figura 3. Corriente Continua
Fuente: (Nilson, 2005)

Como se puede ver a la izquierda de la Figura 4, está una batería de las comúnmente utilizadas en los coches y todo tipo de vehículo motorizado. A la derecha, pilas de amplio uso, lo mismo en linternas que en aparatos y dispositivos eléctricos y electrónicos.

Figura 4. Fuentes de Corriente continua CC
Fuente: (Hubert, 2006)

En los circuitos de corriente continua pueden distinguirse básicamente dos tipos de elementos, los generadores y los receptores. Ver Figura 5. Los primeros aportan al circuito la energía eléctrica necesaria para mantener la corriente eléctrica, los segundos consumen energía eléctrica, o bien la disipan en forma de calor, como es el caso de las resistencias, o bien la convierten en otra forma de energía, como sucede en los motores. Una fuente de corriente continua en un circuito eléctrico se representa mediante el símbolo ⊣⊢ que refleja la polaridad del generador.

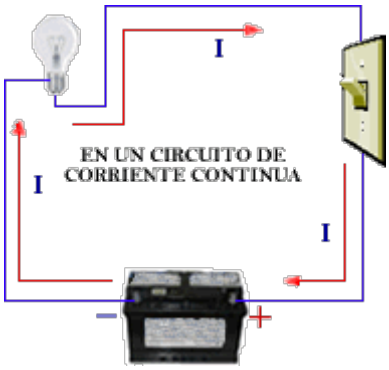

Figura 5. Circuito de Corriente Continua CC
Fuente: (Hubert, 2006)

El estudio cuantitativo de los circuitos eléctricos de corriente continua se efectúa como una aplicación de dos principios básicos:

El enunciado del primer principio, llamada regla de las mallas, dice que la suma algebraica de la variación de potencial a lo largo de cualquier malla del circuito debe ser igual a cero $\sum Vi=0$, y se deduce a partir del simple hecho de que en el estado estacionario la diferencia de potencial entre dos puntos cualesquiera es constante (Hubert, 2006).

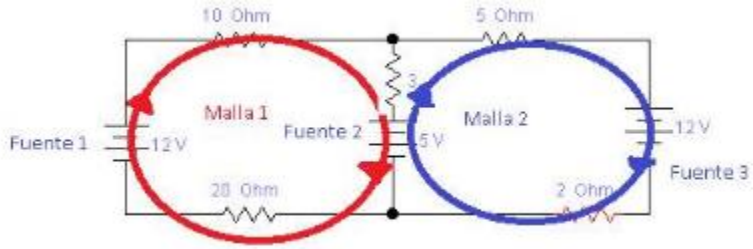

Figura 6. Regla de Mallas
Fuente: (Hubert, 2006)

El segundo enunciado, llamada de los nudos, dice que en un punto o nudo de ramificación de un circuito en donde puede dividirse la corriente, la suma de las corrientes que entran en el nudo debe ser igual a la suma de las corrientes que salen y se deduce de la conservación de cargas (Figura 7). Esta regla es necesaria para circuitos de múltiples mallas que contienen puntos en los que la corriente puede dividirse.

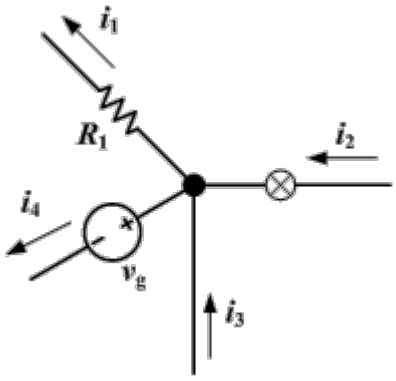

Figura 7. Regla de Nodos
Fuente: (Hubert, 2006)

Es importante conocer que las baterías, los generadores, o cualquier otro dispositivo similar crean cargas eléctricas, de hecho, todos los elementos conocidos

en la naturaleza contienen cargas eléctricas, pero para establecer el flujo en forma de corriente eléctrica es necesario ponerlas en movimiento.

1.2.2. La corriente alterna (C.A.):

Es la corriente eléctrica en la que la magnitud y dirección varían cíclicamente, la forma de onda de la corriente alterna más utilizada es la de una onda senoidal, puesto que se consigue una transmisión más eficiente de la energía. Sin embargo, en ciertas aplicaciones se utilizan otras formas de onda periódicas, tales como la triangular o la cuadrada (Nilson, 2005). La Figura 8 muestra una señal alterna con los parámetros típicos.

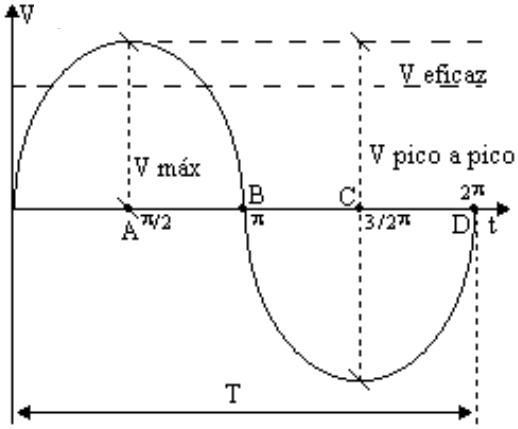

Figura 8. Corriente alterna CA
Fuente: (Nilson, 2005**)**

La razón del amplio uso de la corriente alterna viene determinada por su facilidad de transformación, cualidad de la que carece la corriente continua.

En el caso de la corriente continua la elevación de la tensión se logra conectando dínamos en serie, lo cual no es muy práctico, al contrario en corriente alterna se cuenta con un dispositivo: el transformador, que permite elevar la tensión de una forma eficiente.

USOS

- Para el uso de artefactos electrónicos (tostadora, licuadora, etc.).

- Para crear imanes electromagnéticos.
- Para motores (se transmiten datos mediante la corriente alterna).

a. Fuentes de corriente alterna

La corriente en todo circuito fluye del terminal negativo hacia el positivo, por lo mismo, para que haya flujo de corriente alterna la polaridad debe cambiar su dirección. Las fuentes con estas características se llaman fuentes de corriente alterna, y los circuitos que trabajan con este tipo de corriente se llaman circuitos de C.A., a la potencia que consumen potencia de C.A. (Nilson, 2005).

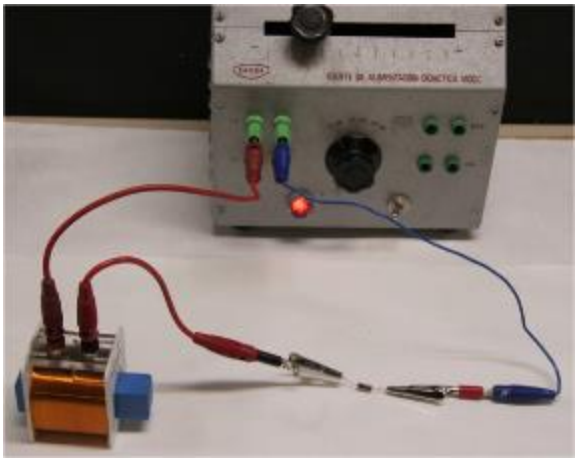

Figura 9. Fuentes de Corriente alterna
Fuente: (Nilson, 2005)

El efecto es el mismo, no importando la dirección de la corriente, ejemplo: cuando por un resistor fluye una corriente, produce calor, ya sea esta directa o alterna, entonces el calor es el efecto que se producirá en el resistor, en el ciclo positivo o negativo de la corriente alterna.

La primera corriente descubierta y por lo mismo usada, fue la corriente directa (C.D.), pero en cuanto se descubrió la corriente alterna, esta fue sustituyendo a la anterior. Hoy, el uso de la corriente alterna se puede decir que es la que mayormente se usa en el mundo, aunque en algunos lugares, se sigue usando

corriente directa o continua, tal como ocurre en las baterías, las dinamos o en cualquier otra fuente generadora de este tipo de corriente eléctrica.

1.3. Instrumentos de Medida

Mediante el uso de ellos se miden e indican magnitudes eléctricas, como corriente, carga, potencial y energía, o las características eléctricas de los circuitos, como la resistencia, la capacitancia y la inductancia. Además permiten localizar las causas de una operación defectuosa en aparatos eléctricos en los cuales no es posible apreciar su funcionamiento en una forma visual, como en el caso de un aparato mecánico (Instrumentos eléctricos de medición, 2008)

1.3.1. Multímetro

El multímetro es un instrumento de medición muy conocido también con los nombres: VOM (Voltios, Ohmios, Miliamperímetro), Tester, Polímetro. En la actualidad hay multímetros con capacidad de medir muchas otras magnitudes. (Capacitancia, frecuencia, temperatura, etc.). Existen otros instrumentos como el osciloscopio que tiene un precio más alto y se utiliza para realizar mediciones más informativas (Rashid, 2004).

Figura 10. Multímetro
Fuente: (Rashid, 2004)

a. Los multímetros analógicos.

Los multímetros analógicos son fáciles de identificar por una aguja que al moverse sobre una escala indica del valor de la magnitud medida. Un multímetro analógico se muestra en la Figura 11.

Figura 11. Multímetro analógico
Fuente: (Máquinas eléctricas, 2015)

b. Los multímetros digitales

Los multímetros digitales se identifican principalmente por un panel numérico para leer los valores medidos, tienen un selector de función y un selector de escala (algunos no tienen selector de escala pues el VOM la determina automáticamente). Algunos tienen un solo selector central como se muestra en la Figura 12.

Figura 12. Multímetro digital
Fuente: (Máquinas eléctricas, 2015)

1.3.2. Osciloscopio

El osciloscopio es un dispositivo de visualización gráfica que también mide los parámetros de la señal incluyendo al voltaje y las características de tiempo de la CA y la CC, así como los tiempos de transición, la frecuencia, el período, etc. Éste aparato demuestra ser particularmente útil para las mediciones de cualquier punto instantáneo en una forma de onda. Ver Figura 13.

El uso principal del osciloscopio implica mostrar la variación de una señal a través del tiempo. Los osciloscopios modernos son digitales y muestran una colección de datos instantáneos de señales repetitivas. Estos aparatos tienen múltiples canales para la visualización simultánea de diferentes señales, además de que poseen pantallas a color para distinguir fácilmente señales diferentes. (Dorf & J.A., 2006)

Figura 13. Osciloscopio
Fuente: (Dorf & J.A., 2006)

El osciloscopio es un instrumento muy útil para visualizar las formas de ondas que se presentan en un circuito.

El osciloscopio digital tiene la ventaja de que permite que la información, recopilada por el instrumento, sea transferida a una PC o una pantalla LCD. Para que la señal se presente estática en la pantalla, ésta es trazada varia veces por segundo, así parece continua en el tiempo a diferencia de los osciloscopios antiguos la tensión a medir se aplica a las placas de desviación vertical oscilante de un tubo de rayos catódicos.

1.4. Generador de Funciones

El generador de funciones es un equipo capaz de generar señales variables en el dominio del tiempo para ser aplicadas posteriormente sobre el circuito bajo prueba (Enciclopedia Electrónica, 2015). Ver figura 14.

Figura 14. Generador de Funciones
Fuente: (Enciclopedia Electrónica, 2015)

Las formas de onda típicas son las triangulares, cuadradas y sinodales. También son muy utilizadas las señales TTL que pueden ser utilizadas como señal de prueba o referencia en circuitos digitales. Un generador de funciones tiene una frecuencia máxima a la cual puede llegar el instrumento, al igual que una amplitud máxima en volts. Otras aplicaciones del generador de funciones pueden ser las de calibración de equipos, rampas de alimentación de osciloscopios, etc.

Un generador de funciones además se utiliza para generar o simular señales específicas con determinadas características. De esta forma, podemos aplicar esta señal generada a un circuito para ver su respuesta. Entonces, para resumir lo anterior, es un simulador de señales de diferentes características (Acha E., 2002). Aunque existen multitud de generadores de funciones de mayor o menor complejidad todos incorporan ciertas funciones y controles básicos que se describen a continuación. Ver Figura 15.

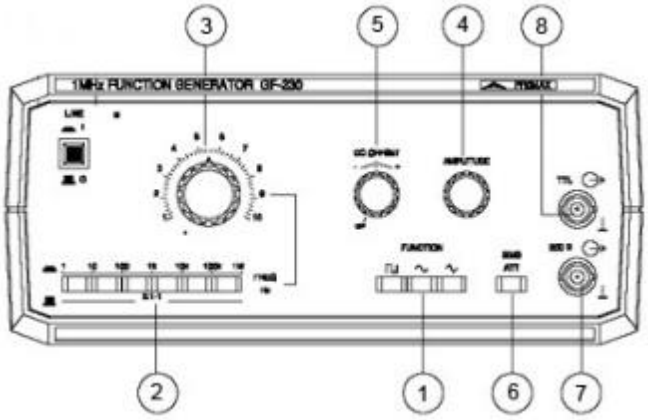

Figura 15. Partes del Generador de Funciones
Fuente: (Hubert, 2006)

- *1.- Selector de funciones.* Controla la forma de onda de la señal de salida. Puede ser triangular, cuadrada o senoidal.
- *2.- Selector de rango.* Selecciona el rango o margen de frecuencias de trabajo de la señal de salida. Su valor va determinado en décadas, es decir, de 1 a 10 Hz, de 10 a 100, etc.
- *3.- Control de frecuencia.* Regula la frecuencia de salida dentro del margen seleccionado mediante el selector de rango.
- *4.- Control de amplitud.* Mando que regula la amplitud de la señal de salida.
- *5.- DC offset.* Regula la tensión continua de salida que se superpone a la señal variable en el tiempo de salida.
- *6.- Atenuador de 20dB.* Ofrece la posibilidad de atenuar la señal de salida 20 dB (100 veces) sobre la amplitud seleccionada con el control número 4.
- *7.- Salida.* Conector de salida que entrega la señal elegida con una impedancia determinada.
- *8.- Salida TTL.* Entrega una consecución de pulsos TTL (0 – 5 V) con la misma frecuencia que la señal de salida.

1.5. Sensores y actuadores

En general, cualquier sistema que requiera adquirir, procesar, medir, controlar y monitorear información de su entorno necesitará de componentes periféricos de

entrada y salida llamados transductores, los cuales se dividen en transductores de entrada y transductores de salida. En forma genérica, los primeros son conocidos como sensores y los segundos como actuadores.

1.5.1. Sensores

Un sensor es un dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas (Nilson, 2005).

Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, nivel, torsión, humedad, pH, etc. Ver Figura 16.

a. Características

- Rango de medida: dominio en la magnitud medida en el que puede aplicarse el sensor.
- Precisión: es el error de medida máximo esperado.
- Sensibilidad de un sensor: es el mínimo valor de entrada que genera un cambio a la salida.
- Resolución: mínima variación de la magnitud de entrada que puede apreciarse a la salida.
- Rapidez de respuesta: puede ser un tiempo fijo o depender de cuánto varíe la magnitud a medir. Depende de la capacidad del sistema para seguir las variaciones de la magnitud de entrada.
- Repetibilidad: error esperado al repetir varias veces la misma medida.

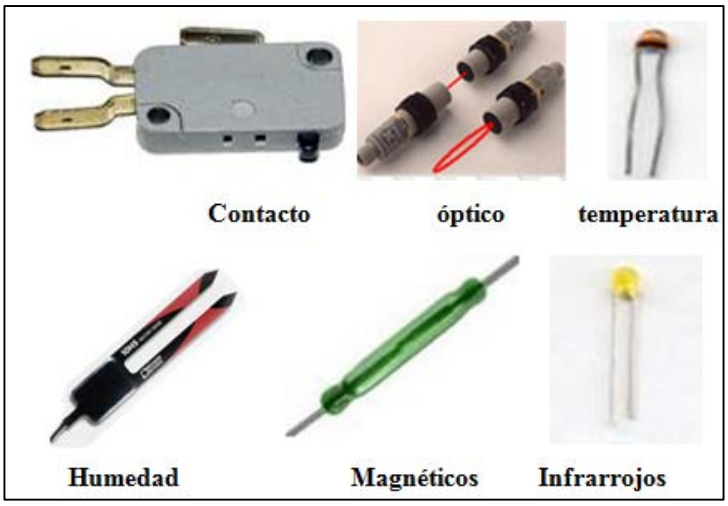

Figura 16. Tipos de sensores
Fuente: (Recursostic.educacion.sensores, 2015)

Los sensores se pueden clasificar de varias formas, pero una de las más comunes es por su área de aplicación, como por ejemplo:

- De contacto: para detectar el final del recorrido o la posición límite del componente.
- Ópticos: detectan la presencia de una persona u objeto cuando interrumpe el haz de luz que llega al sensor.
- De Temperatura: Regulación de la temperatura alta o baja – termistores, RTD.
- De humedad: Detectan el nivel de líquido en un depósito.
- Magnéticos: Detecta los campos magnéticos que provocan los imanes o las corrientes eléctricas.
- Infrarrojos: Detectan las emisiones de los diodos

1.5.2. Actuadores

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado (Nilson, 2005). Ver Figura 17.

Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control como, por ejemplo, una válvula. Se puede clasificar a los actuadores de la siguiente manera:

- Electrónicos, aquellos que son accionados por medio de corrientes eléctricas, también son muy utilizados en los aparatos mecatrónicos, como por ejemplo, en los robots.

Figura 17. Actuador eléctrico o electrónico
Fuente: (Recursostic.educacion.sensores, 2015)

- Hidráulicos, aquellos que se emplean cuando lo que se requiere es potencia, son los de mayor antigüedad, pueden ser clasificados de acuerdo con la forma de operación, funcionan en base a fluidos a presión.

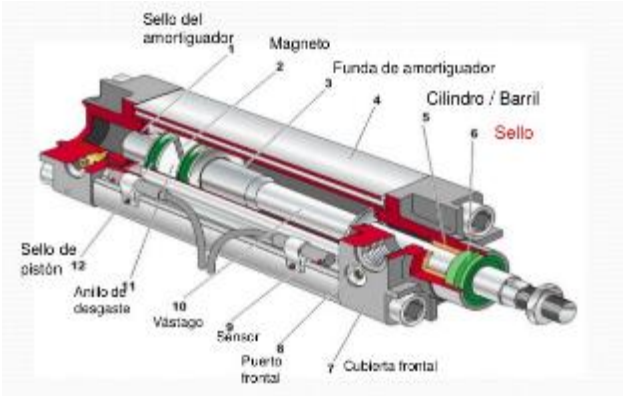

Figura 18. Actuador Hidráulico
Fuente: (Recursostic.educacion.sensores, 2015)

- A los mecanismos que convierten la energía del aire comprimido en trabajo mecánico se les denomina actuadores neumáticos. Aunque en esencia son idénticos a los actuadores hidráulicos, el rango de compresión es mayor en este caso.

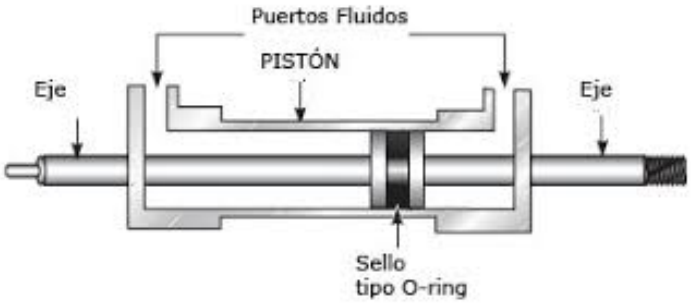

Figura 19. Actuador Neumático
Fuente: (Recursostic.educacion.sensores, 2015)

La estructura de un actuador eléctrico es simple en comparación con la de los actuadores hidráulicos y neumáticos, ya que sólo requieren de energía eléctrica como fuente de poder; se utilizan cables eléctricos para transmitir electricidad y las señales, es altamente versátil, prácticamente no hay restricciones respecto a la distancia entre la fuente de poder y el actuador.

Figura 20. Válvula de bola con actuador eléctrico con BSR
Fuente: (Festo, 2015)

1.6. Instrumentación Virtual

El concepto de instrumentación virtual nace a partir del uso del computador personal como “instrumento” de medición de señales tales como temperatura, presión, caudal, entre otras. Es decir, el PC comienza a ser utilizado para realizar mediciones de fenómenos físicos representados en señales de corriente (como 4...20mA) y/o voltaje (por ejemplo, 0-5 Vdc) normalmente con una gran precisión y número de muestras por segundo (Instrumentacion virtual, 2007). Ver figura 21.

Sin embargo, el concepto de "instrumentación virtual" va más allá de la simple medición de corriente o voltaje. También involucra el procesamiento, análisis, almacenamiento, distribución y despliegue de los datos e información relacionados con la medición de una o varias señales específicas, mediante software que permitan la implementación de algoritmos de control, es factible integrar y controlar complicados procesos. Es decir, el instrumento virtual no se conforma con la adquisición de la señal, sino que también involucra la interfaz gráfica, las funciones de análisis y procesamiento de señales.

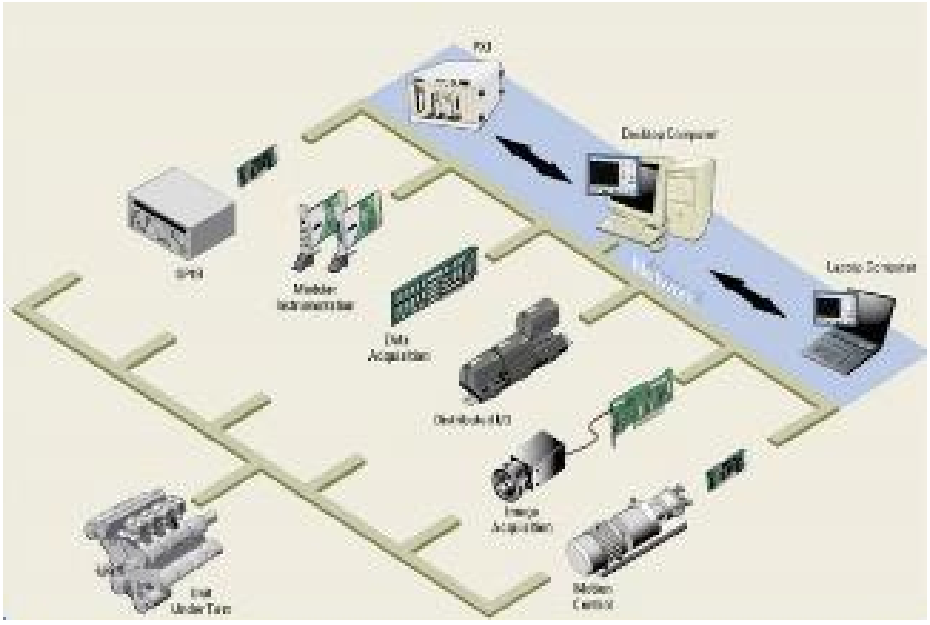

Figura 21. Diagrama de Instrumentos tradicionales- instrumentos virtuales
Fuente: (Instrumentacion virtual, 2015)

Los instrumentos tradicionales y los instrumentos virtuales basados en software comparten en su gran mayoría los mismos componentes, aunque son diseñados con una filosofía diferente. Un instrumento virtual puede realizar las tres funciones básicas de un instrumento convencional: adquisición, análisis y presentación de datos.

Instrumentación virtual se define como “la combinación de software poderoso y personalizable con hardware de medición modular los cuales, en conjunto, desempeñan una tarea específica definida por el usuario” (Biel Solé, Olivé Duran, Prat Tasias, & Sánchez Robert, 2009).

Un instrumento virtual combina tecnologías comerciales, como una PC o estación de trabajo, con software flexible y una gran variedad de módulos, de hardware, de medición y control para crear sistemas que cumplan con sus necesidades. Esto representa un cambio fundamental: de sistemas tradicionales de instrumentación centrados en hardware a sistemas ahora centrados en software que aprovechan el poder de cómputo, la productividad y las capacidades de despliegue y conectividad de una PC. De esta forma, es posible construir sistemas que se adaptan a las necesidades de la aplicación, sin estar limitado por la funcionalidad definida de los instrumentos tradicionales.

1.7. Interfaces Gráficas

La interfaz gráfica de usuario, conocida también como GUI (de inglés graphycal user interface) es un programa informático que actúa de interfaz de usuario con un grupo de objetos gráficos para representar la información y acciones dispoibres en la interfaz. Como se puede apreciar en la figura 22, su principal uso, consiste en proporcionar un entorno visual sencillo para permitir la comunicación con el sistema operativo de una máquina o computador (Sheneidereman, 2008)

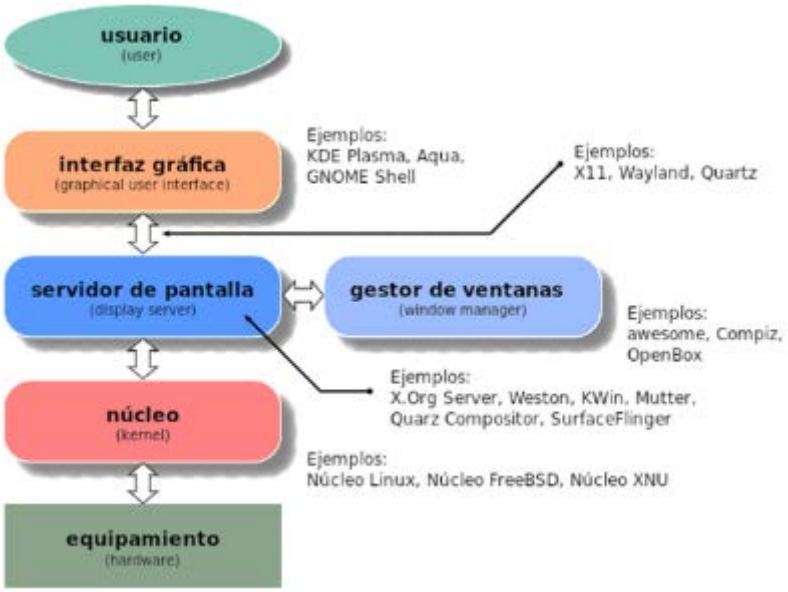

Figura 22. Interfaz gráfica de usuario
Fuente: (memarcos.com, 2004**)**

Una interfaz permite comunicar dos sistemas que no hablan el mismo lenguaje. Restringido a aspectos técnicos, se emplea el término interfaz para definir el juego de conexiones y dispositivos que hacen posible la comunicación entre dos sistemas. Sin embargo, cuando se habla de interfaz se refiera a la cara visible de los programas tal y como se presenta a los alumnos para que interactúen con la máquina. La interfaz gráfica implica una pantalla constituida por una serie de menús e iconos que representan las opciones que el usuario puede tomar dentro del sistema. Ver figura 23.

Las características básicas de una buena interfaz podrían sintetizarse en:

- Facilidad de comprensión, aprendizaje y uso
- Representación fija y permanente de un determinado contexto de acción (fondo)
- El objeto de interés ha de ser de fácil identificación
- Diseño ergonómico mediante el establecimiento de menús, barras de acciones e iconos de fácil acceso
- Las interacciones se basarán en acciones físicas sobre elementos de código visual o auditivo (iconos, botones, imágenes, mensajes de texto o sonoros, barras de

desplazamiento y navegación) y en selecciones de tipo menú con sintaxis y órdenes.

- Las operaciones serán rápidas, incrementales y reversibles, con efectos inmediatos.
- Existencia de herramientas de ayuda y consulta.
- Tratamiento del error bien cuidado y adecuado al nivel de usuario.

Figura 23. Interfaz gráfica de usuario

La interfaz es el elemento que permite al usuario interactuar con los contenidos, no sólo se aprecia una interfaz atractiva, sino funcional.

1.8. Tarjeta Arduino

Arduino es una plataforma de hardware de código abierto, basada en una sencilla placa con entradas y salidas, analógicas y digitales, en un entorno de desarrollo que está basado en el lenguaje de programación Processing. Es un dispositivo que conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital (figura 24).

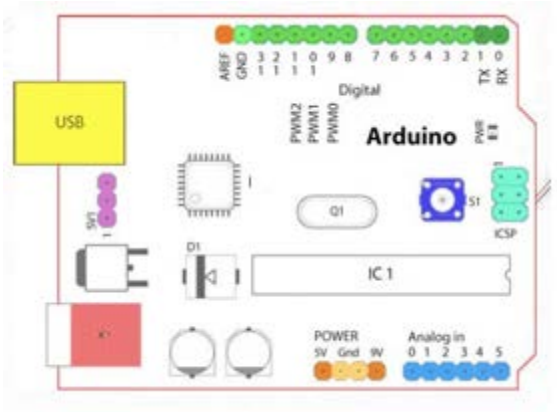

Figura 24. Tarjeta Arduino
Fuente: Arduino,2017

En los últimos años Arduino ha surgido como una plataforma de creación de prototipos electrónicos revolucionaria, con gran facilidad, variedad y exigibilidad para implementar proyectos tanto por su software como por su hardware pues no se necesita amplia experiencia o conocimiento en la programación de microcontroladores para poder utilizarlo (Arduino, 2017). En la tabla 1 se puede apreciar las especificaciones técnicas de la tarjeta Arduino Uno.

Tabla 1.
Especificaciones técnicas de Arduino Uno.

Microcontrolador	**ATmega328**
Tensión de operación	5 V
Tensión de alimentación (recomendada)	7-12 V
Voltaje de entrada (limite)	6-20 V
Pines para entrada-salida digital	14 (6 pueden usarse como salida de PWM)
Pines de entradas analógicas	6
Corriente continua por pin I/O	40 mA
Corriente continua en el pin 3.3 V	50 mA
Memoria Flash	32 kB (0,5 ocupados por el bootloader)
SRAM	2 kB
EEPROM	1 kB
Frecuencia de reloj	16 MH

Fuente:(Arduino, 2017).

1.8.1. Características de Arduino

Su tamaño es de 74x53 mm. Se programa mediante una conexión USB que también se usa para alimentarla (5V). Existe la posibilidad de usar la alimentación externa, que ha de ser de 9V. Posee 14 pines de E/S digital (6 de las cuales pueden ser usadas como PWM) y 6 pines analógicos. Lo cual da la oportunidad de alimentar el circuito con dos voltajes distintos, 5V o bien 3,3V (Noble J. , 2009). Ver Figura 25.

Figura 25. Distribución de pines de tarjeta Arduino Uno
Fuente: (Arduino, 2017)

Contiene un microprocesador ATmega328, que posee una memoria flash de 32 KB (512 bytes son usados por el bootloader), RAM de 2KB y 1KB de memoria EEPROM. El voltaje de operación es de 5V y la frecuencia de trabajo del reloj es de 16 MHz. Se destaca también la preinstalación del bootloader.

Dispone de un reset (botón rojo) que suministra un valor LOW que reinicia el microcontrolador, un conector ICSP (In Circuit Serial Programming), que es el sistema utilizado en los dispositivos PIC para programarlos sin ser necesario retirar el chip del circuito del que formase parte.

Los pines 3, 5, 6, 9, 10 y 11 son pines provistos de 8 bits de salida PWM. Estos pines permiten obtener información del exterior y que la placa actúe en función de dicha información (sensores, motores, servos,..). Los pines 0 (Rx) y 1 (Tx) son los encargados de enviar y recibir datos serie TTL.

La función de los pines 2 y 3 es la de manejar interrupciones (Arduino UNO sólo es capaz de manejar dos interrupciones. Los pines 10, 11, 12 y 13 sirven de apoyo a la comunicación SPI con la biblioteca SPI. El bus SPI (Serial Peripheral Interface) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Es un estándar para el control de cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj. Existen 3 pines de tierra marcados como GND (0V).

La alimentación al circuito puede ser de 5V o 3,3V. Se puede aplicar un voltaje de entrada a la placa mediante el pin Vin cuando ésta sea alimentada por una fuente externa y conocer el valor exacto del voltaje aplicado a la placa. Respecto a entradas analógicas, por ejemplo Arduino UNO dispone de 6 distribuidas en los pines A0, A1, A2, A3, A4 y A5. Cada una de ellas proporciona una resolución de 10 bits (1024 valores).

El puerto USB permite una comunicación serie con el ordenador mediante el estándar de los controladores USB COM, sin necesidad de controlador externo. La placa envía un aviso con un parpadeo de los leds Rx y Tx que la comunicación se está llevando a cabo.

El conector plug hembra de 2.1 mm se lo puede usar para alimentar a la placa externamente, evitando así el uso del USB (si el sketch ya está cargado en la placa, no necesita el ordenador para que la placa funcione, basta alimentarla).

1.9. Tipo de red que conforman las tarjetas Arduino

Si se tiene dos o más tarjetas Arduino, es posible comunicarlas entre sí por el puerto serie, sin que se comporten como "una única placa". Cada placa tiene su procesador y su programa. Si desde una de las placas se quiere manejar dispositivos de la otra se deberá establecer algún protocolo de mensajes para comunicarlas entre sí. Ver Figura 26.

Las tarjetas Arduino se comunican a través de los puertos TWI. El Pin Análogo 4 (SDA) y el Pin Análogo 5 (SCL), se conecta la tierra entre sí, ya que los dos tienen electricidad exterior independiente, uno de ellos es el master y el otro es el esclavo, es necesario utilizar la Libreria Wire de Arduino (Banzi, 2009).

Más que querer controlar todos los pines desde un solo micro, lo interesante de unir dos o más placas es poder simplificar el trabajo y dividir las tareas en dos.

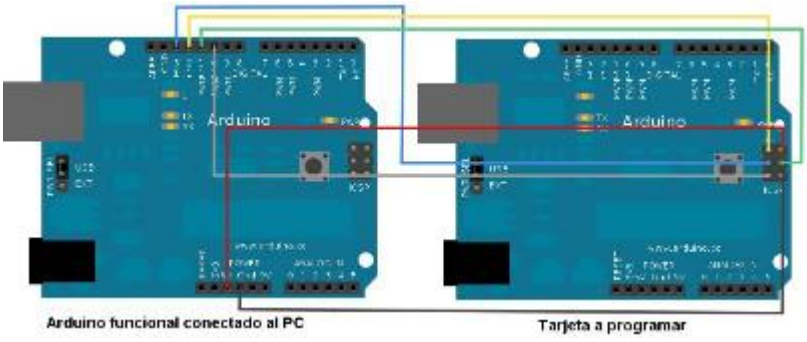

Figura 26. Red de Tarjetas Arduino
Fuente: (Arduino, 2017)

1.10. Software a utilizar

Los lenguajes de programación son una herramienta vital a la hora de crear nuevos proyectos de hardware porque todo el proyecto tiene interacción con el software. Debido a que adquirir una licencia hoy en día hace encarecer cualquier proyecto que se pretende desarrollar, es necesario, adaptarse a software de código abierto o software libre, los mismos que son fáciles de descargar e instalar.

Lo que permite desarrollar una aplicación basada en Software libre que trabaje de manera adecuada con una tarjeta Arduino (Oxer & Blemings, 2009).

Sin embargo, es importante recalcar que Arduino puede tomar información del entorno a través de sus entradas analógicas y digitales, puede controlar luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computador.

La tarjeta Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing que es similar a C++, ya que es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización.

Es posible comunicar una aplicación que corra sobre Arduino con otros dispositivos que corran otros lenguajes de programación y aplicaciones populares, debido a que Arduino usa la transmisión serial de datos, la cual es soportada por un sin número de lenguajes. Y para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida.

1.11. Metodología de enseñanza aprendizaje

Mediante la implementación de un módulo didáctico que cubra las necesidades de los laboratorios de electrónica básica, es posible solucionar parte de la problemática existente en la educación a nivel de bachillerato, ya que la práctica de enseñanza – aprendizaje de la electrónica actualmente se viene desarrollando mediante métodos expositivos tradicionales y no se enfatiza las prácticas experimentales y menos aún se hace uso adecuado de las nuevas tecnologías de la información y la comunicación para el refuerzo del aprendizaje de los alumnos

Luego de revisado el Plan analítico del Área de Eléctrica y Electrónica que maneja el Colegio Técnico de Bachillerato Dr. Trajano Naranjo I. Se determinó que de manera teórica el alumno recibe los conocimientos necesarios sobre los dispositivos electrónicos básicos, los cuales serán utilizados en el diseño de circuitos prácticos tanto digitales como analógicos.

El tiempo que conlleva la realización de una sola práctica por día con dos horas académicas de laboratorio, en donde solo se podrá implementar una por tema, impide que el estudiante pueda reforzar la teoría con la práctica.

Con la implementación de este módulo se garantiza el aumento de las competencias de los futuros bachilleres, ya que concreta la solidez en los conocimientos para posteriores estudios en centros universitarios. Investigaciones han llegado a la conclusión de que cuando se aplica el método experimental didáctico en la enseñanza de electrónica se realiza el refuerzo del aprendizaje significativamente.

1.12. Condiciones de calibración del módulo didáctico.

El calibrado o calibración es el procedimiento de comparación entre lo que indica un instrumento y lo que "debiera indicar" de acuerdo a un patrón de referencia con valor conocido. De esta definición se deduce que para calibrar un instrumento es necesario disponer de uno de mayor precisión que proporcione el valor convencionalmente verdadero que es el que se empleará para compararlo con la indicación del instrumento sometido a calibrado. Esto se realiza mediante una cadena ininterrumpida y documentada de comparaciones hasta llegar al patrón primario, y que constituye lo que se llama trazabilidad. El objetivo del calibrado es mantener y verificar el buen funcionamiento de los equipos, responder a los requisitos establecidos en las normas de calidad y garantizar la fiabilidad y trazabilidad de las medidas (Hubert, 2006).

Durante el calibrado, se contrasta el valor de salida del instrumento a calibrar frente a un patrón en diferentes puntos de calibración. Si el error de calibración (error puesto de manifiesto durante la calibración) es inferior al límite de rechazo, la calibración será aceptada. En caso contrario se requerirá ajuste del instrumento y una contrastación posterior, tantas veces como sea necesario hasta que se obtenga un error inferior al límite establecido.

En la calibración, los resultados deben documentarse con un certificado de calibración, en el cual se hacen constar los errores encontrados así como las correcciones empleadas, errores máximos permitidos, además pueden incluir tablas, gráficos, etc.

Parámetros a considerar en toda calibración.

- Error de medición: Resultado de una medición menos el valor verdadero del medido.
- Desviación: Valor medido menos su valor de referencia.
- Error relativo: Es la relación entre el error de medida y un valor verdadero del mensurando. Valor del mensurando recogido en el patrón. El error relativo se suele expresar también en forma porcentual: 100 %.
- Error sistemático: Serían debidos a causas que podrían ser controladas o eliminadas, por ejemplo medidas realizadas con un aparato averiado o mal calibrado.
- Corrección: Valor sumado algebraicamente al resultado sin corregir de una medición para compensar un error sistemático. De lo que se deduce que la corrección, o bien sea reflejada en la hoja de calibración o bien minimizada mediante el ajuste, solo aplica a las derivas de los instrumentos.
- Ajuste: Al proceso de corrección se le denomina ajuste, y es la operación destinada a llevar a un instrumento de medida a un estado de funcionamiento conveniente para su utilización. El ajuste puede ser automático, semiautomático o manual.
- Patrón primario: Patrón que es designado o ampliamente reconocido como poseedor de las más altas cualidades metrológicas y cuyo valor se acepta sin referirse a otros patrones de la misma magnitud.

- Patrón secundario: Patrón cuyo valor se establece por comparación con un patrón primario de la misma magnitud.
- Patrón de referencia: Patrón, en general de la más alta calidad metrológica, disponible en un lugar dado o en una organización determinada, del cual se derivan las mediciones realizadas en dicho lugar.
- Patrón de trabajo: Patrón que se utiliza corrientemente para calibrar o controlar medidas materializadas, instrumentos de medida o materiales de referencia.
- Patrón de medida: Valor de medición materializado, aparato o sistema de medida con el que se intenta definir, realizar, conservar, o reproducir una unidad física o bien uno o varios valores conocidos de una magnitud con el fin de que sirvan de comparación a otros elementos de medida (Berenguer, 2012).

De lo indicado anteriormente se deberá tener un instrumento de mayor precisión que proporcione el valor convencionalmente verdadero, para ello se utilizará un multímetro marca fluke con el cual se podrá comparar los valores obtenidos de nuestro módulo didáctico, de existir variaciones o un error significativo se realizará la calibración mediante algoritmos de programación.

CAPÍTULO II

SELECCIÓN DE LOS DISPOSITIVOS

En este capítulo se detalla el diseño de las fuentes de voltaje, la selección de: tarjetas de interface para instrumentos de medida, interfaces gráficas, y la tarjeta de adquisición de datos Arduino. En la Figura 27 se muestra un Sistema de medida de variables eléctricas a medir.

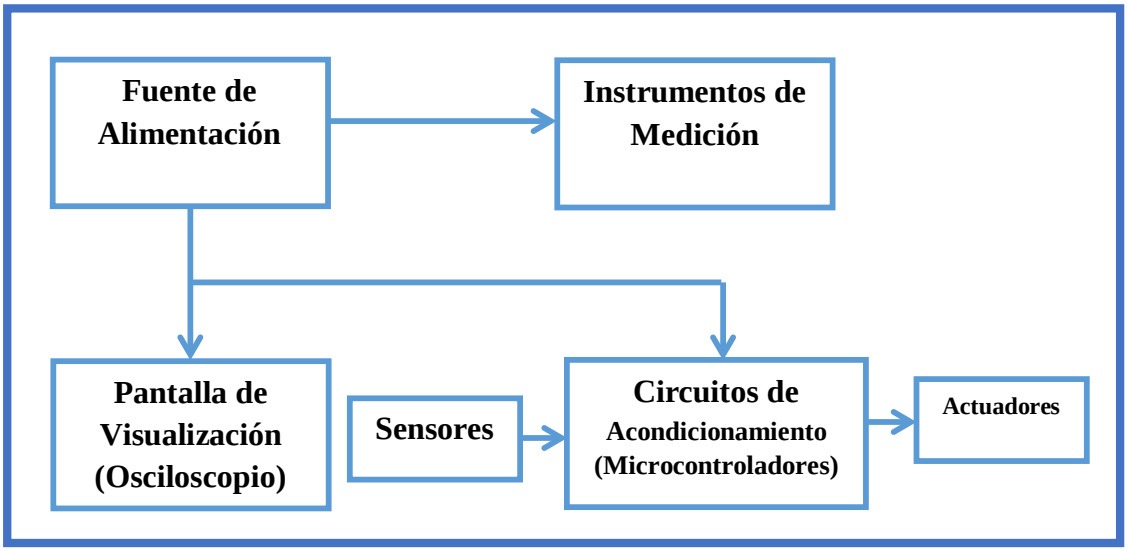

Figura 27. Diagrama de bloques de un Sistema de medida de variables eléctricas.

2.1. Diseño de las Fuentes de Voltaje

Para las diferentes aplicaciones se usaron tres fuentes de corriente continua: una fuente fija dual de +/- 5 V, una fuente fija dual de +/- 12V y fuentes variable de 0 a 24V y de 0 a -24V.

Las corrientes que las fuentes podrán entregar serán de 1.5 amperios. Todas las fuentes están centralizadas en un Microcontrolador Arduino UNO R3, un LCD de 16x2 sirve como interfaz para seleccionar los rangos de niveles que se vayan a utilizar de acuerdo a cada necesidad.

El módulo propuesto requiere de seis fuentes de voltaje: cuatro fijas y dos variables para lo cual se ha seleccionado los reguladores LM337, LM317 para una salida de voltaje variable de (1,3v a 24v) y (-1,3v a -24v), los reguladores 7805, 7812 para tensiones fijas positivas de +5v y +12v y finalmente los reguladores 7905, 7912 para tensiones fijas negativas de -5v y -12v.

2.1.1. Regulador Ajustable Negativo LM337

Se seleccionó el circuito integrado LM337, el cual es un regulador de tres terminales de voltaje negativo capaz de entregar hasta 1,5A y un rango de salida de -1.2 a -37 volts. Requiere solo dos resistores para fijar la tensión de salida, además las regulaciones de línea y de carga son mejores que la de los reguladores fijos. Ver Figura 28.

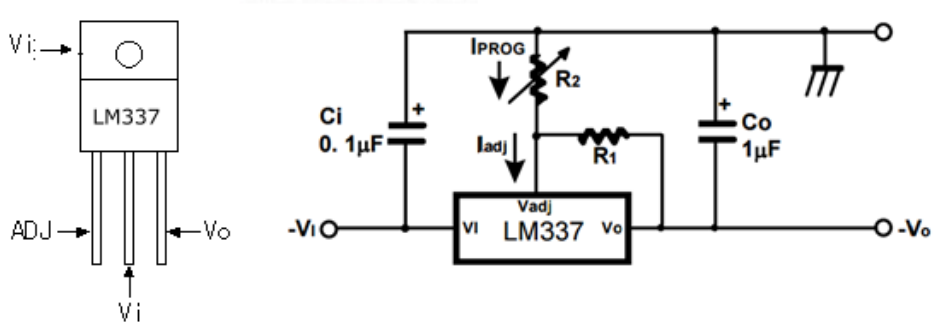

Figura 28. Diagrama de distribución de pines del circuito integrado LM337
Fuente: (Circuito integrado LM337, 2017)

El LM337 ofrece una protección completa contra sobrecarga. Incluyendo en el chip un limitador de corriente y protección contra sobrecarga térmica. Todas las protecciones de sobrecarga permanecen funcionales aun cuando la terminal de ajuste esté desconectada (Circuito integrado LM337, 2017).

En la Tabla 2 se puede apreciar las caractrísticas que posee el circuito integrado LM337.

Tabla 2.
Especificaciones del Circuito integrado LM337

ESPECIFICACIONES	DESCRIPCIÓN
Voltaje de salida ajustable:	-1.2 V a -37V
Corriente de salida máxima:	1.5 A
Regulación de línea típicamente:	0.01% / volts,
Regulación de la carga típicamente:	0.1%
Encapsulado:	TO-220AB
Rechazo de rizado:	80 db
Protección contra cortocircuito	
Etapa de salida a transistor con compensación de área segura	
Operación flotante para aplicaciones de altos voltajes	

Fuente: Electrónica aplicada, Antonio Hermosa

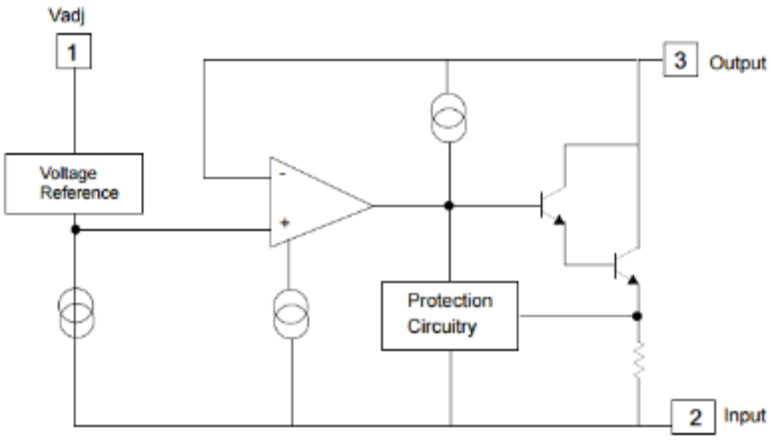

Figura 29. Diagrama de conexión interna del regulador LM337
Fuente: (Imagen Circuito integrado LM337, 2017)

2.1.2. Regulador Ajustable positivo LM317

Para el proyecto se requiere una fuente ajustable positiva para lo cual se seleccionó el circuito integrado LM317, ya que es un regulador de tensión ajustable de tres terminales, capaz de suministrar en condiciones normales 1.5 A, en un rango que va desde 1,2 hasta 37 Voltios. Para su empleo solo requiere

dos resistores exteriores para conseguir el valor de salida, se seleccionó este circuito integrado principalmente por que dispone de protección por limitación de corriente y exceso de temperatura, siendo funcional la protección por sobrecarga, incluso si el terminal de regulación está desconectado. La Figura 30 describe el diagrama esquemático representativo del circuito integrado LM317

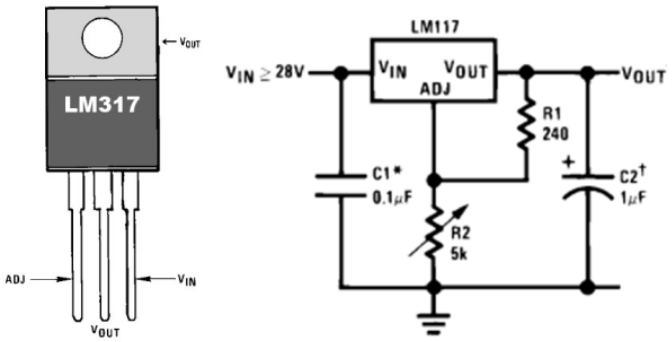

Figura 30. Diagrama de distribución de pines del circuito integrado LM317
Fuente: (Circuito integrado LM317, 2017)

Este regulador de tensión proporciona una tensión de salida variable sin más que añadir una resistencia y un potenciómetro. La Tabla 3 describe las principales características eléctricas que posee el circuito integrado LM317.

Tabla 3.
Especificaciones del circuito integrado LM317

ESPECIFICACIONES	DESCRIPCIÓN
Regulador ajustable de tensión	+1,2 a 37 V
Voltaje primario de entrada	40V
Rango ajustable del voltaje de salida	1.2V a 37V
Corriente de salida	1.5 A
Rango de operación de temperatura	0 °C a +125 °C
Voltaje máximo de salida	37V
Voltaje mínimo de salida	1,2V
Número de base	317
Marcador	LM317T
Número genérico CI	317
Tipo de terminación	Agujero pasante
Tipo de regulador de tensión	Ajustable positivo

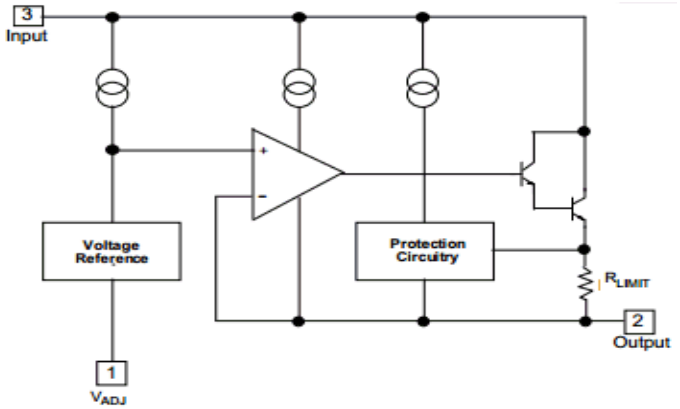

Figura 31. Diagrama de circuito interno de regulador integrado LM317
Fuente: (Circuito integrado LM317, 2017)

2.1.3. Fuente de tensión positiva fija

Para la fuente fija se ha seleccionado el regulador de tensión positiva de la serie 78xx. Este componente tiene tres terminales (voltaje de entrada, masa y voltaje de salida) y especificaciones similares que sólo difieren en la tensión de salida suministrada o en la intensidad. La intensidad máxima depende del código intercalado tras los dos primeros dígitos.

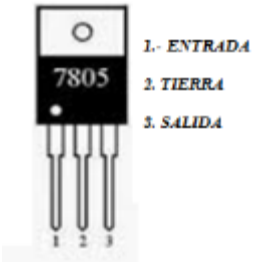

Figura 32. Regulador de voltaje de salida 7805
Fuente: (Imagen Regulador de voltaje 7805, 2017)

En este caso específicamente se ha seleccionado el regulador de voltaje de salida 7805, ya que entrega 5V de corriente continua, lo que lo hace sumamente útil para alimentar dispositivos TTL. El encapsulado en el que usualmente se lo utiliza es el TO220, aunque también se lo encuentra en encapsulados pequeños de montaje superficial y en encapsulados grandes y metálicos como el TO3 (Regulador de voltaje de salida 7805, 2017)

La tensión de alimentación debe ser un poco más de 2 voltios superior a la tensión que entrega el regulador y menor a 35V. Usualmente, el modelo estándar (TO220) soporta corrientes de hasta 1 A aunque hay diversos modelos en el mercado con corrientes que van desde los 0,1A. El dispositivo posee como protección un limitador de corriente por cortocircuito, y además, otro limitador por temperatura que puede reducir el nivel de corriente.

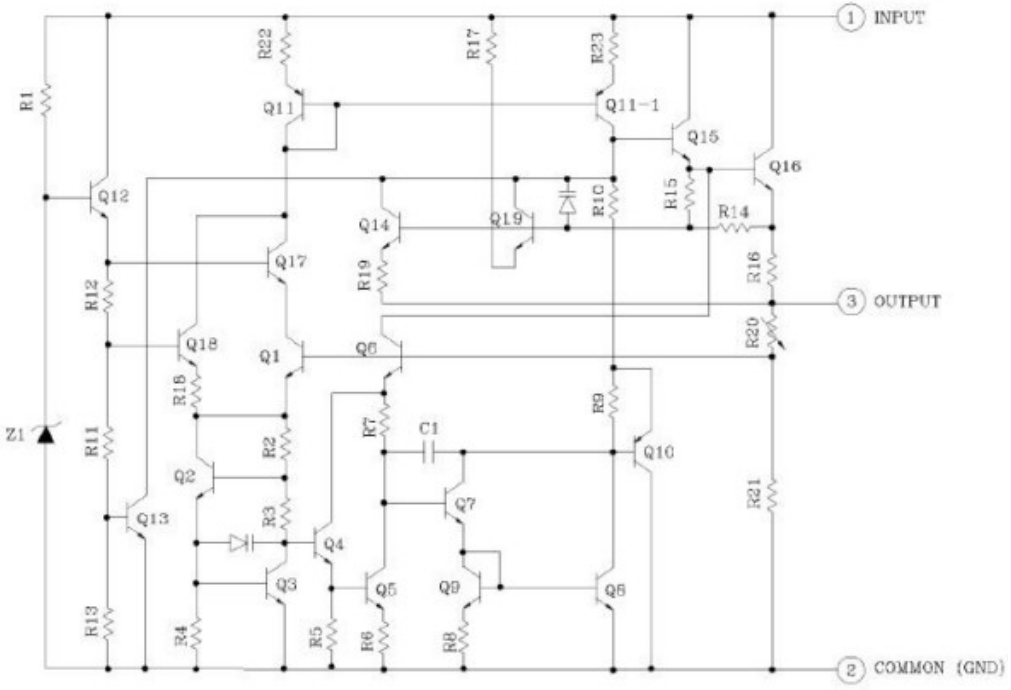

Figura 33. Circuito interno del integrado 7805
Fuente: (Imagen Regulador de voltaje 7805, 2017)

2.1.4. Fuente de tensión negativa fija

El aspecto es como el anterior, sin embargo, este se suele usar en combinación con el 78XX para suministrar tensiones simétricas. La tensión entre Vout y GND es de - XX voltios, por eso se dice que este es un regulador de tensión negativa. La forma de llamarlos es la misma: el 7905 es de 5V, el 7912 es de -12V.

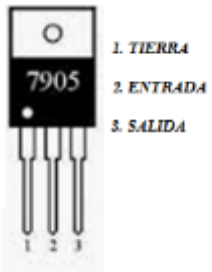

Figura 34. Regulador de voltaje negativo 7905
Fuente: (Imagen regulador de voltaje negativo 7905, 2017)

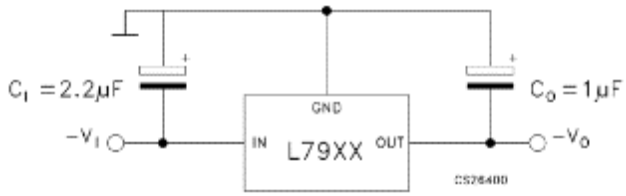

Figura 35. Diagrama de conexión del integrado 7905
Fuente: (electronica2017embajadores.com)

Tabla 4.
Especificaciones del regulador de voltaje negativo 7905

ESPECIFICACIONES	DESCRIPCIÓN
Voltaje de salida	-5V
Corriente de salida máx	1 A
Voltaje de entrada máx	-35 V
Voltaje dropout típico	2 V
Protección contra sobrecarga térmica	
Protección contra cortocircuito	
Encapsulado TO-220	

Fuente: Electrónica aplicada, 2016

2.1.5. Fuente de voltaje ±24 Vdc

Para propósitos del presente proyecto, se ha seleccionado el cual es una fuente simétrica regulada variable, que puede proporcionar hasta 24Vdc por sección

(-24 y +24), para un consumo de hasta 1A, la cual puede modificarse para proporcionar hasta 3A o incluso más. Los transistores Q1 y Q2 deben se montados en disipadores térmicos, al igual que los integrados LM317 y LM350T. (Diseño de la fuente de voltaje, 2017)

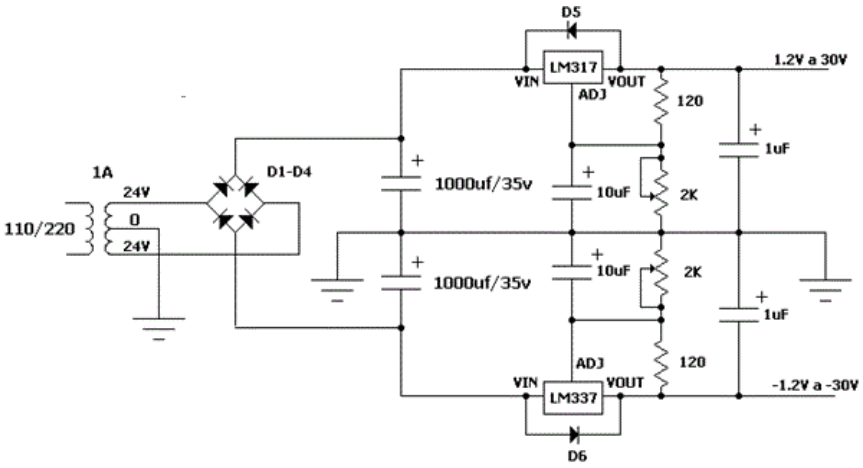

Figura 36. Circuito de la fuente de alimentación simétrica 24V.
Fuente: (Circuito de la fuente de alimentacion, 2017)

Los componentes que se utilizaron de acuerdo a los requerimientos fueron los siguientes:

- T1 - Transformador con primario adecuado para la red eléctrica (110 o 220V) y secundario de 24+24 para 2A.
- IC1 - Circuito Integrado LM317
- IC2 - Circuito Integrado LM337
- Q1 - Transistor TIP3055
- Q2 - Transistor TIP2955
- Q3 - Transistor BC548 o similar
- Q4 - Transistor BC558 o similar
- D1 al D4 - Diodos 1N5804 o similares.
- D5 y D6 - LEDs
- C1 y C2 - Condensadores electrolíticos 4700uF 35V
- C3 al C6 - Condensadores de 0.1uF (100nF) 50V
- R1 y R2 - Resistencias de 1000 ohms 1/2W
- R3 y R4 - Resistencias de 220 ohms 1/2W

- R5 y R6 - Resistencias de 0.5 ohm 5W
- R7 y R8 - Resistencias de 470 Kohms 1/2W
- P1 y P2 - Potenciómetros de 5000 ohms

Todos estos materiales son considerados al momento de realizar el diseño de la Fuente, los mismos que pueden ser modificados en el transcurso de las pruebas, con la finalidad de mejorar las características de la fuente.

2.1.6. Pantalla LCD

La pantalla LCD seleccionada para este proyecto es una pantalla de 2x16 (2 líneas de 16 caracteres). Cada caracter dispone de 5x7 pixeles. Este dispositivo está gobernado internamente por un microcontrolador y regula todos los parámetros de presentación.

Figura 37. Pantalla LCD de 2x16
Fuente: (Display LCD de 2x16, 2017)

Los pines de conexión de estos módulos incluyen un bus de datos de 8 bits, un pin de habilitación (E), un pin de selección que indica si lo que se está enviando por el bus es un dato o una instrucción (RS) y un pin que indica si se va a leer o escribir en el módulo (R/W). La Tabla 5 describe en detalle los pines mencionados.

Tabla 5.
Función de los Pines del LCD 16X2

PIN	SIMBOLO	Nombre y Función
1	VSS	GND (Tierra 0v)
2	VDD	Alimentación +5V

3	VO	Ajuste del contraste
4	RS#	Selección DATO/CONTROL
5	RW#	Lectura o escritura en LCD
6	E	Habilitación
7	D0	D0 bit menos significativo
8	D1	D1
9	D2	D2
10	D3	D3
11	D4	D4
12	D5	D5
13	D6	D6
14	D7	D7 bit más significativo
15	LED+	Ánodo de LED backlight
16	LED-	Cátodo de LED backlight

Nota: El signo # significa negado

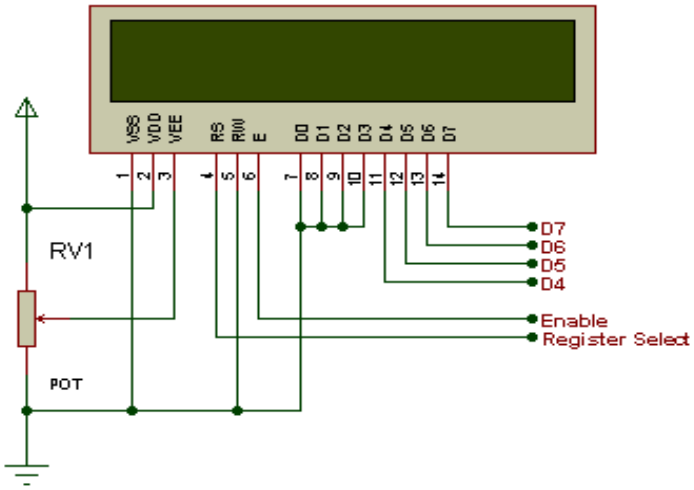

Figura 38. Distribución de los pines de la pantalla LCD 16X2
Fuente: (Distribucion de los pines pantalla LCD 2x16, 2017)

Según la operación que se desee realizar en el módulo LCD, los pines de control E, RS#, RW# deben tener un estado determinado. Además, debe tener en el bus de datos un código que indique un carácter para mostrar en la pantalla o una instrucción de control para el display.

Los módulos LCD responden a un conjunto especial de instrucciones, estas deben ser enviadas por el microcontrolador o sistema de control al display, según la operación que se requiera. Se muestran a continuación el conjunto de instrucciones del módulo LCD.

Tabla 6.
Instrucciones del módulo LCD

CONTROL Y DATO	SEÑAL DE CONTROL		DATO / DIRECCIÓN								DESCRIPCIÓN
INSTRUCCIÓN	**RS**	**RW**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**	
Borrar pantalla	0	0	0	0	0	0	0	0	0	1	Limpia todo el display y retoma el cursor a la posición de inicio, dirección 0
Cursos a casa	0	0	0	0	0	0	0	0	1	-	Retoma el cursos a la posición inicio (dirección 0) También retoma el display, desplazando a la posición original. Los contenidos de la DDRAM permanecen sin cambios
Seleccionar modo	0	0	0	0	0	0	0	1	I/D	S	Configura la dirección de movimiento y si se desplaza o no el display. Esta operación es realizada durante operaciones de lectura escritura
Encender / apagar pantalla	0	0	0	0	0	0	1	D	C	B	Configura el estado ON/OFF de todo el display (D), el cursor (C) y el parpadeo del caracter en la posición del cursor.
Desplazar cursor / pantalla	0	0	0	0	0	1	S/C	R/L	-	-	Mueve el cursor y desplaza el display sin cambiar los contenidos de la DDRAM
Activar función	0	0	0	0	1	D/L	N	F	-	-	Configura el tamaño de la interfaz (DL), el número de líneas del display (N) y la fuente del carácter (F). N=0 es 1 línea N=1 es 2 líneas
CG RAM	0	0	0	1	Dirección generador de RAM						Ajusta la dirección del generador de caracteres. El dato CG RAM es enviado y recibido después de este ajuste
DD RAM	0	0	1	Dirección de datos RAM							Ajusta la dirección de la DDRAM. La dirección es enviado y recibido después de este ajuste
Bandera de ocupado	0	0	BF	AC							Lectura de la bandera Busy Flag. Indicando que operaciones internas son realizadas y lectura de los contenidos del contador de direcciones

Escritura CG RAM/DD RAM	1	0	Escritura de dato	Escribe datos en la DDRAM o en la CGRAM
Lectura CGRAM / DDRAM	1	1	Lectura de dato	Lectura de datos desde la DDRAM o la CGRAM

Fuente: Principios de electricidad electrónica, 2008

NOTA: El pin RS# debe tomar el valor 0(cero) cuando lo que se va a enviar es una instrucción de control y debe tomar el valor 1(uno) cuando lo que se va a enviar es un dato.

Tabla 7.
Significado de las abreviaturas

ABREVIATURA	SIGNIFICADO
I/D	= 1 incremental = 0 decrementa
S	= 1 desplaza el mensaje en la pantalla = 0 mensaje fijo en la pantalla
D	= 1 encender (activar) la pantalla = 0 apagar la pantalla (descativar)
C	= 1 activar cursor = 0 desactivar cursor
B	= 1 parpadea caracter señalado por el cursor = 0 no parpadea el caracter
S/C	= 1 desplaza pantalla = 0 mueve el cursor
RL	= 1 desplazamiento a la derecha = 0 desplazamiento a la izquierda
DL	= 1 datos de 8 bits = 0 datos de 4 bits
BF	= 1 durante operación interna del módulo = 0 finaliza la operación interna

Fuente: Componentes electrónicos, 2005

2.2. Selección de las Tarjetas de Interface para instrumentos de medida

Una vez definida y seleccionada la fuente de voltaje, es necesario definir tanto las interfaces de entrada y de salida que se utilizarán en el módulo didáctico, para ello se describe a continuación cada una de ellas.

2.2.1. Potenciómetros

Se seleccionó un potenciómetro ya que es un resistor con valor de resistencia variable, permite controlar la intensidad de corriente que fluye por un circuito si se conecta en paralelo, o la diferencia de potencial al conectarlo en serie.

Figura 39. Potenciómetro
Fuente: (Imagen de potenciometro, 2017)

Potenciómetros rotatorios. Se controlan girando su eje. Son los más habituales, son de larga duración y ocupan poco espacio.

Potenciómetros deslizantes. La pista resistiva es recta, de modo que el recorrido del cursor también lo es. Son más frágiles que los rotatorios y ocupan más espacio. Además, suelen ser más sensibles al polvo.

2.2.2. Pulsadores

Un pulsador es un operador eléctrico que, cuando se oprime, permite el paso de la corriente eléctrica y, cuando se deja de oprimir, lo interrumpe. Hay dos tipos de pulsadores (NA) normalmente abiertos y (NC) normalmente cerrados.

Figura 40. Pulsadores
Fuente: (Pulsadores, 2017)

Se utilizó pulsadores que normalmente tienen los contactos cerrados; es decir, la corriente estará circulando hasta que lo usemos. Al pulsar, el circuito se abre y deja de funcionar. Este tipo de pulsadores se utilizan normalmente para la parada de emergencia de máquinas o mecanismos.

2.2.3. Interruptores

La razón para utilizar un interruptor es porque permite abrir o cerrar un circuito de forma permanente. Al accionarlo, varía su posición, abriendo un circuito que estaba cerrado o cerrando uno que estaba abierto, y permanece así hasta que se lo vuelva a accionar. El módulo didáctico lleva varios interruptores para permitir el paso de la corriente y regular su funcionamiento.

Figura 41. Interruptores
Fuente: (Interruptores, 2017)

2.3. Selección de la tarjeta Arduino

La selección adecuada la tarjeta Arduino que se la hace de acuerdo a las necesidades del proyecto como el número de pines analógicos y digitales que se necesitan para un programa muy largo, con muchas constantes y variables demandará una cantidad mayor de memoria flash para su almacenamiento, por lo que se debe elegir una placa adecuada para no quedarse cortos. La RAM será la encargada de cargar los datos para su inmediato procesamiento, va ligada al microcontrolador, puesto que ambos afectan a la agilidad de procesamiento de Arduino. En cuanto al voltaje, una placa Arduino podría trabajar incluso con tensiones de 220v en alterna con el uso por ejemplo de relés.

2.3.1. Arduino Uno R3

Una vez analizado todos estos aspectos se determinó que la tarjeta a ser utilizada para la construcción del módulo didáctico y que permita interactuar con una Red de Arduinos es la Tarjeta Arduino Uno R3, ya que es una tarjeta electrónica basada en el microcontrolador Atmega328. Dispone de 14 entradas/salidas digitales, 6 de las cuales se pueden emplear como salidas PWM (modulación de anchura de pulsos) (Noble J. , 2009)

Dispone también de 6 entradas analógicas, un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP y un pulsador para el reset. La placa sólo se conecta al ordenador a través de un cable USB, se puede alimentarla con un adaptador de corriente AC/DC.

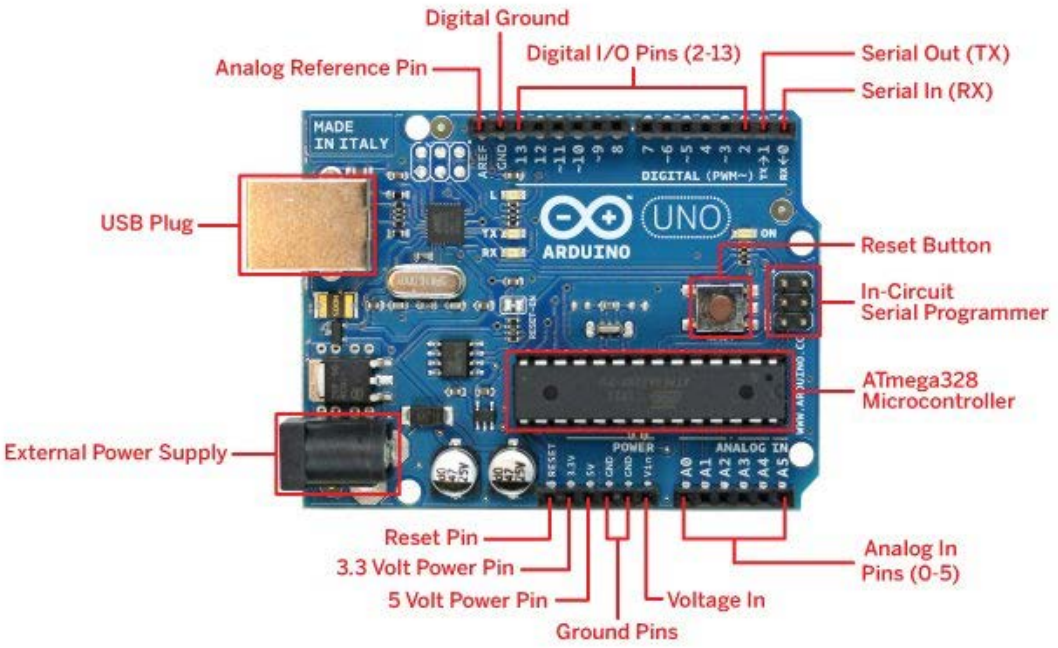

Figura 42. Arduino Uno R3 – Vista Frontal
Fuente: (Arduino Uno R3, 2017)

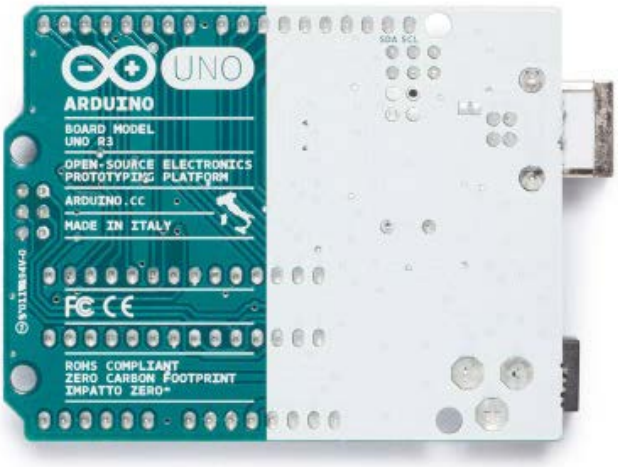

Figura 43. Arduino Uno R3 – Vista Posterior
Fuente: (Arduino Uno R3, 2017)

Arduino Uno R3 no utiliza el convertidor USB-serie, por el contrario, integra un microcontrolador Atmega16U2 (Atmega8U2 version R2) programado como un convertidor USB a serie. El Arduino Uno R3 posee las siguientes características:

Tabla 8.
Especificaciones del Arduino Uno R3

ESPECIFICACIÓN	DESCRIPCIÓN
Microcontrolador	ATmega328
Voltage de operación	5V
Voltaje de entrada (recomendado)	7 – 12 V
Voltaje de entrada (límite)	6 – 20 V
Pines Digitales de Entrada/Salida	14 (de los cuales 6 proveen salidas PWM)
Pines de Entrada Analógica	6
Corriente DC por E / S Pin	40 mA
Corriente DC de 3.3V Pin	50 mA
Memoria Flash	32 KB (ATmega328) de los cuales 0,5 KB utilizado por el gestor de arranque
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidad de reloj	16 MHz

Fuente: Análisis básico de circuitos eléctricos y electrónicos, 2015

Arduino puede verse afectado por señales electromagnéticas y de radio frecuencia, viéndose afectados los sistemas de adquisición de datos. El microcontrolador en el tablero está programado utilizando el lenguaje de Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing), pero es compatible con varios lenguajes de programación entre ellos Phyton que es el utilizado para la programación del módulo didáctico.

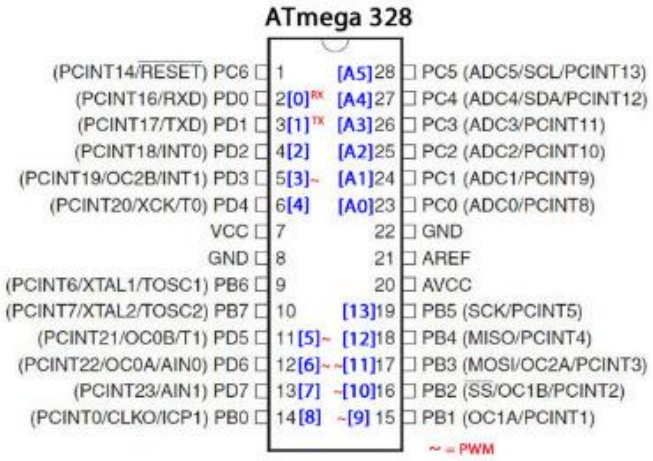

Figura 44. ATmega328
Fuente: (ATmega328, 2017)

Cuenta con 14 pines digitales de entrada / salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, un resonador cerámico 16 MHz, una conexión USB, un conector de alimentación, un header ICSP, y un botón de reinicio.

2.3.2. Microcontrolador SainSmart Mega 2560R3

Son placas SainSmart totalmente compatibles con Arduino Mega, utilizan un chip ATmega8u2 programado para controlar el USB y convertirlo a serie.

Figura 45. SainSmart Mega 2560R3
Fuente: (ATmega328, 2017)

Tabla 9.
Especificaciones Microcontrolador SainSmart Mega 2560R3

ESPECIFICACIÓN	DESCRIPCIÓN
Microcontrolador	ATmega2560
Voltage de operación	5V
Voltaje de entrada (recomendado)	7 – 12 V
Voltaje de entrada (límite)	6 – 20 V
Pines Digitales de Entrada/Salida	54 (de los cuales 15 proveen salidas PWM)
Pines de Entrada Analógica	16
Corriente DC por E / S Pin	40 mA
Corriente DC de 3.3V Pin	50 mA
Memoria Flash	256 KB de los cuales 8 KB son usados por el gestor de arranque
SRAM	8 KB
EEPROM	4 KB
Velocidad de reloj	16 MHz

Fuente: (ATmega328, 2017)

La tarjeta Arduino Mega 2560 R3 emplea un ATmega8U2 en lugar de un chip FTDI. Esto permite tasas de transferencia mayores, elimina la necesidad de instalar drivers para Linux, Mac o Windows; y brinda la capacidad de mostrar a Arduino Uno como un teclado, mouse, joystick etc., al conectarlo a una computadora.

Incluye un procesador de AVR ATMEGA2560 con un amplio espacio de memoria para programar a 16Mhz. Posee una alta cantidad de pines de entrada y salida y sus 4 puertos UART por hardware de esta manera se da el soporte necesario para el microcontrolador.

2.4. Sensor de corriente.

El sensor de corriente ACS712 (Figura 46) trabaja con un sensor de efecto Hall que detecta el campo magnético que se produce por inducción de la corriente que circula por la línea que se está midiendo.

Figura 46. Sensor de corriente ACS712
Fuente: (ATmega328, 2017)

Este sensor entrega una salida de voltaje proporcional a la corriente, su conexión es sencilla debido a que dispone de una bornera para conectar la línea que queremos medir y 3 pines (GND, salida analógica y Vcc) como se ilustra en la siguiente Figura.

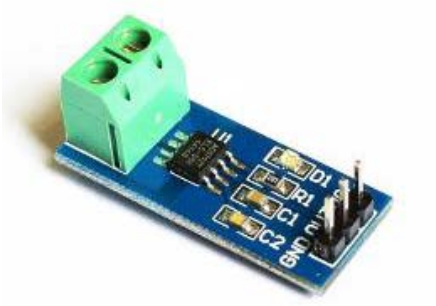

Figura 47. SainSmart Mega 2560R3

Fuente: (ATmega328, 2017)

Hay tres modelos de sensores ACS712 que se detallan en la tabla 10. El rango de medicion y sensibilidad varían dependiendo del modelo del integrado.

Tabla 10.
Rango y sensibilidad del sensor de corriente ACS 712

Modelo	Rango	Sensibilidad
ACS712ELCTR-05B-T	-5 a 5 A	185 mV/A
ACS712ELCTR-20A-T	-20 a 20 A	100 mV/A
ACS712ELCTR-30A-T	-30 a 30 A	66 mV/A

El sensor entrega un valor de 2.5 voltios para una corriente de 0A y a partir de allí incrementa proporcionalmente de acuerdo a la sensibilidad, teniendo una relación lineal entre la salida de voltaje del sensor y la corriente. En la siguiente fórmula se describe el comportamiento de estas dos variables para el aacondicionamiento.

$$V = m\,I + 2.5$$

Donde, m se define como la pendiente y equivale a la Sensibilidad. Despejando tendremos la ecuación para hallar la corriente a partir de la lectura del sensor, esta ecuación es indispensable para la programación en Arduino:

$$I = \frac{V - 2.5}{Sensibilidad}$$

2.5. Sensor de Voltaje AC

Es un módulo utilizado para medir el voltaje de fase de corriente alterna. Posee un transformador, por lo que solo se puede utilizar para leer el voltaje alterno.

Figura 48. Sensor de Voltaje alterno
Fuente: (ATmega328, 2017)

Este es un módulo transformador de voltaje, tiene una salida monofásica activa. Además, contiene un circuito con amplificador operacional para compensar el offset de la salida análoga. Puede medir voltaje de baja tensión y la salida analógica es ajustable con el potenciómetro que posee en la placa.

Sus especficiaciones técnicas se apuntan a continuación:

- Voltaje de alimentación: 5Vdc
- Señal de salida: analógica
- Dimensiones: 5 cm x 2 cm x 2.4 cm
- Propiedades del transformador:
- Corriente nominal de entrada y salida: 2mA
- Ratio en entrada salida 1000:1000
- Diferencia de fase <30º (a 50ohm)
- Rango linear 0-3mA (a 50ohm)
- Linearidad 1%
- Presición 0.2
- Aislamiento eléctrico: hasta 3000V

CAPÍTULO III

DISEÑO E IMPLEMENTACIÓN DE SOFTWARE Y HARDWARE

En este capítulo se desarrollará el diseño e implementación del hardware y software necesario para el diseño y construcción de un módulo didáctico de bajo costo para prácticas de electrónica general de la Unidad Educativa “Dr. Trajano Naranjo” donde el estudiante pordra interactuar con el módulo a través de una intefaz gráfica de fácil manipulación.

3.1. Diagrama de bloques del módulo didáctico

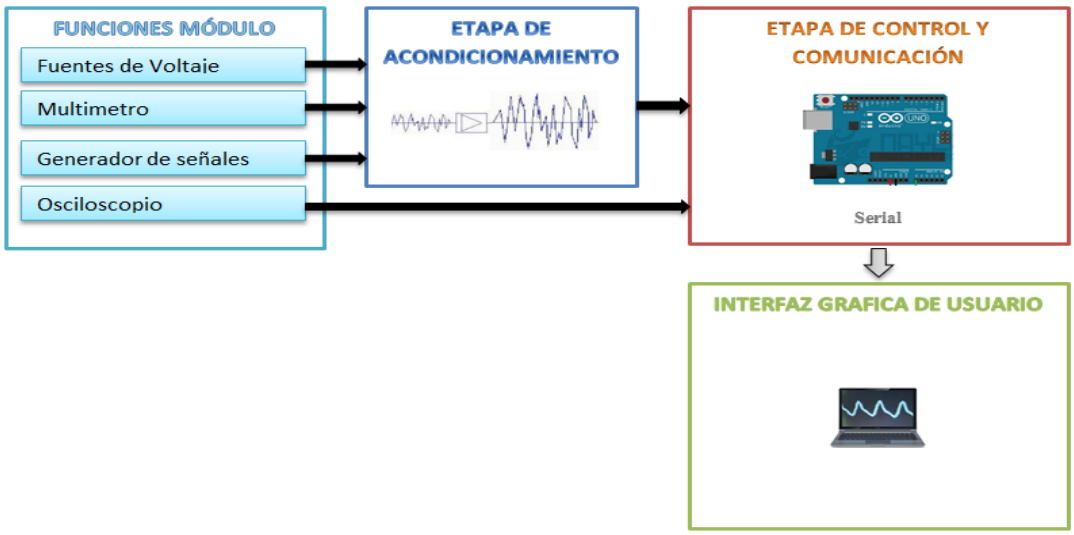

Figura 49. Diagrama de bloques del módulo didáctico

3.2. Implementación de Hardware del módulo didáctico

Se procede a la construcción del módulo didáctico, una vez seleccionados los materiales requeridos y la ubicación de los diferentes elementos.

3.2.1. Construcción de la estructura del módulo didáctico

La construcción del módulo didáctico se la realizó con una carcasa metálica para brindar una mayor resistencia y durabilidad en la parte exterior o frontal a la que tendrá acceso el usuario, donde están ubicados los indicadores, potenciómetros, puntas de pruebas y puntos de conexión para las diferentes aplicaciones que brindará el módulo didáctico. Esta estructura se lo realizo en acero inoxidable de 40cm x 30cm x 20cm.

En la parte frontal se realizaron perforaciones para colocar dispositivos y elementos necesarios para interactuar con el módulo. Figura 50.

Figura 50. Perforación de la parte frontal del módulo

Se coloca papel adhesivo para mejorar la apariencia del módulo, dando un aspecto mucho más estetico.

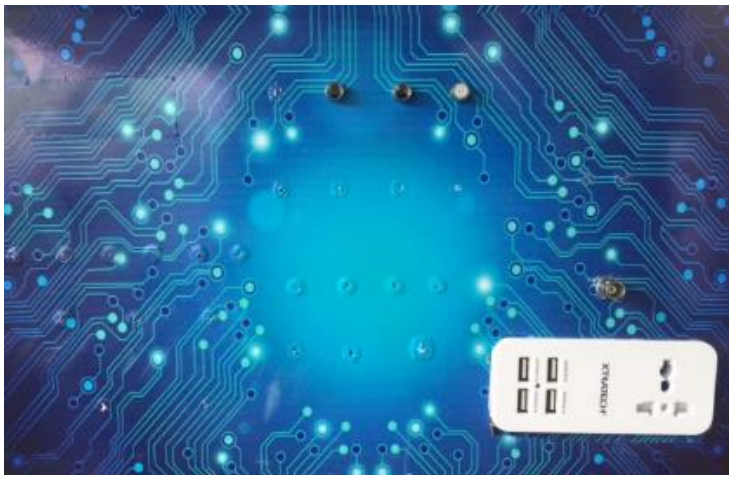

Figura 51. Panel frontal con papel adhesivo

3.2.2. Diseño y construcción de las fuentes de voltaje

Las distintas fuentes de alimentación del módulo convirten la entrada de voltaje alterno de la red en una tensión continua, estas constan de las siguientes etapas ilustradas en la figura 52: transformación, rectificación, filtrado y regulación.

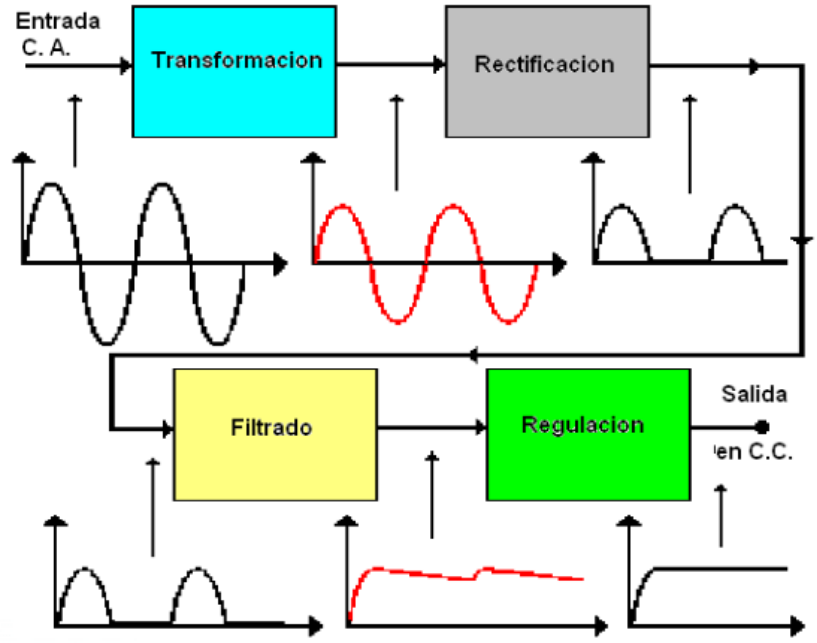

Figura 52. Diagrama de bloques de una fuente de voltaje

a. Fuentes de voltaje fijas

El transformador de la tensión de entrada es de 110V que convierte en una tensión fácil de trabajar, en este caso de 12v y alimenta el circuito con una intensidad de 1 A. Esta tensión sigue manteniéndose en alterna, es decir entre +12v y -12v para el requerimiento de la fuente es necesario tener una tensión continua que no oscile con valores negativos. Para ello se requiere un puente rectificador el cual su función es rectificar esta corriente para tener a la salida una señal pulsatoria que oscila entre los 0v y 15,5V. Por último, se filtra la corriente para que se suministre un voltaje casi constante con dos condensadores en paralelo a la entrada del estabilizador de tensión.

Para un voltaje eficaz Vefc=12V, se obtiene un valor pico de esta tensión de $Vp=12\sqrt{2}=16,9V$, este valor pasa por los diodos rectificadores y su salida es Vrectificada=Vmáx=(16,9-1,4)V dando una tensión máxima a la salida del rectificador Vmáx=15,5V, al pasarla por la etapa del filtrado se obtiene un Vppr=10%(15,5) =1,5V, luego el Voltaje mínimo de la tensión filtrada será Vmín=Vmáx-Vppr=15,5V-1,5V de donde Vmín=14V que es la tensión de entrada Vent del regulador.

Entonces para el cálculo del capacitor utilizamos la siguiente formula:

$$C = \frac{10*I}{f*(Vp-1,4)}$$

Al reemplazar valores obtenemos una capacitancia de 2500µF para lo cual es recomendable que sea mayor a este valor para que el regulador trabaje en forma adecuada, se utiliza uno de 3300uF a 35V porque se recomienda que sea al menos el doble de la tensión máxima rectificada por eso de los picos o sobretensiones que pueden aparecer, en este caso será 2*(15,5)=31V por lo tanto el un condensador implementado es de 35V.

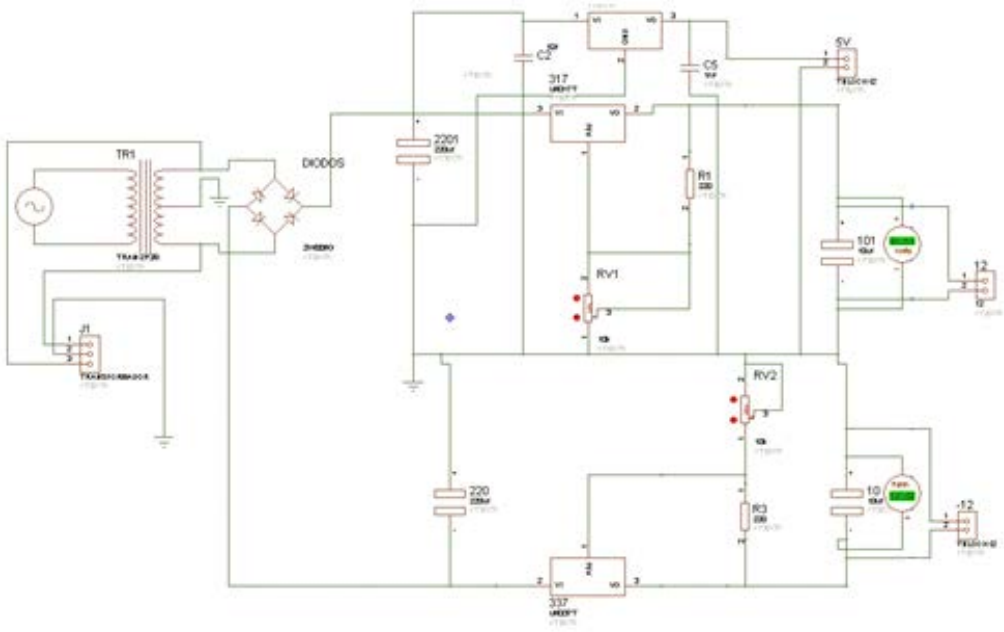

Figura 53. Circuito de las fuentes fijas +5V -5V +12V -12V

b. Fuentes de voltaje variable

Para construir esta fuente se utiliza un transformador con derivación central (15V-0V-15V) es decir que al medir entre los extremos mida 30V y entre el tab central y uno de los extremos mida 15V además de entregar una corriente mínima de 1A. Figura 54.

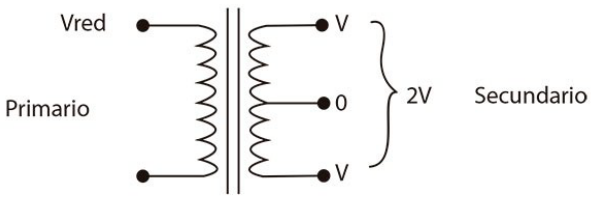

Figura 54. Transformador con derivación central
Fuente: mrelbernitutoriales.com

Para obtener tensión variable positiva continua se utiliza el regulador integrado LM317T y para la tensión variable negativa continua se usa el regulador integrado LM337T.

Los cálculos que se realizan para el capacitor se toman semejantes a la del literal anterior dando como resultado un capacitor de C= 3300µF de modo que se utilizan dos condensadores cuya capacitancia sea C=3300uF a 35V. Se coloca también dos capacitores, el uno en la entrada del regulador de 100nF para filtrar el ruido que se produce en el cable o pista que los une y el otro de 1µF para mejorar la respuesta transitoria.

Según la recomendación de la hoja de datos del regulador R1=240Ω y R2=2,1KΩ que esta última es la resistencia de ajuste para la parte positiva del circuito de alimentación, siendo el voltaje de salida la suma de los voltajes en las dos resistencias (Vsal=VR1+VR2). Figura 55.

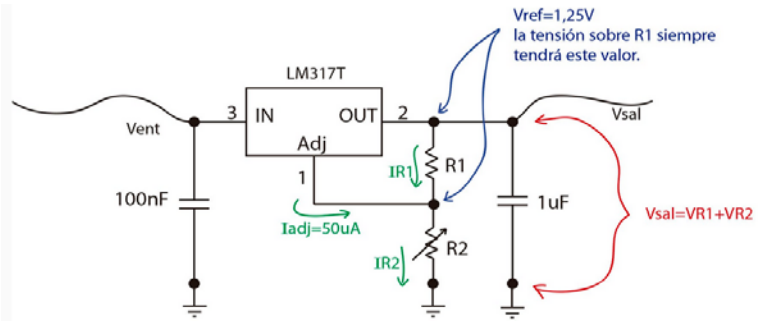

Figura 55. Voltaje de salida positiva de la fuente
Fuente: Principios de electricidad electrónica, 2015

Para el calculo del capacitor de la fuente negativa se utiliza la misma fórmula y el cálculo será similar a la de la parte positiva, dando un resultado de C=3300µF, R1=240Ω y R2=2,1KΩ con la siguiente configuración.

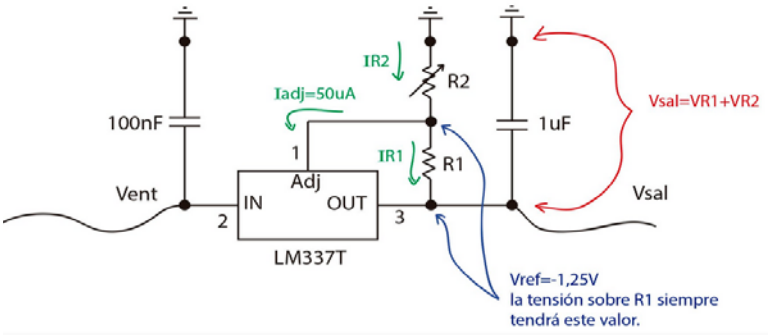

Figura 56. Voltaje de salida positiva de la fuente
Fuente: Principios de electricidad electrónica, 2015

Entonces el circuito completo diseñado para la fuente simétrica se muestra en la siguiente figura.

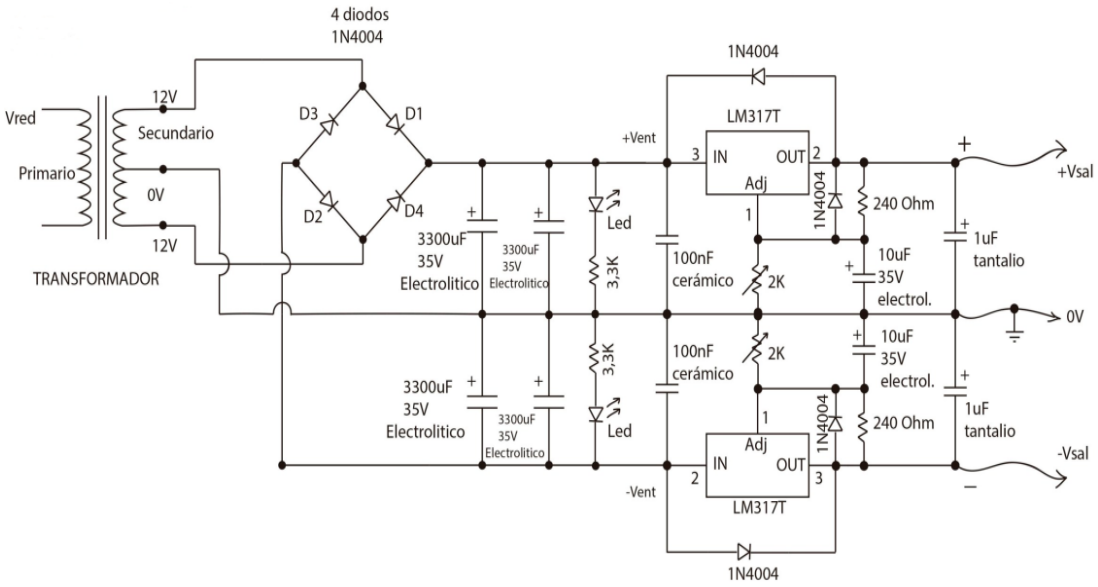

Figura 57. Voltaje de salida positiva de la fuente
Fuente: Principios de electricidad electrónica, 2015

3.2.3. Diseño y construcción del multímetro

a. Lectura de voltaje de las fuentes variables

La tarjeta Arduino pueden medir hasta 5V por lo tanto es recomendable utilizar resistencias para ayudar a protegerlo de sobrecargas de tensión inesperadas. Es común utilizar un divisor de tensión que consta de dos resistencias (R1 y R2) en serie, es decir, dividir el voltaje de entrada para adaptarlo al rango de voltaje que pueden leer las entradas analógicas del Arduino (5V).

Debido a que las fuentes de voltaje del módulo están diseñadas para una salida máxima de 24V, por seguridad se realiza este divisor para que su lectura llegue hasta +30V (Figura 58a) y -30V (Figura 58b) en las entradas analógica del Arduino.

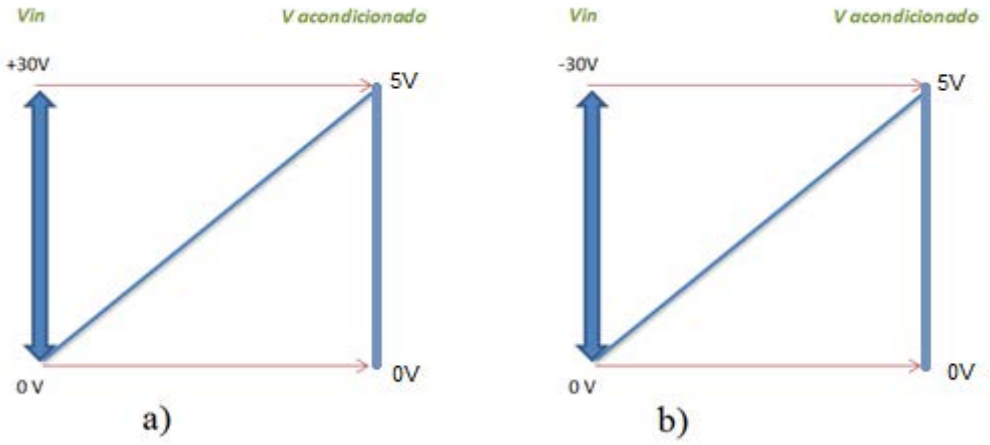

Figura 58. Acondicionamiento de la señal de voltaje a) Lectura positiva b) Lectura negativa

La función del divisor es entregar una tensión al pin analógico de Arduino, ésta tensión será convertida en formato digital que procese el microcontrolador. En este caso, el voltaje de entrada después de tomar del divisor de tensión entre R1 y R2 ingresa al pin A4. El circuito muestra los valores para R1 de 62KΩ (calculada por la fórmula del divisor) en serie con R2 de 10KΩ.

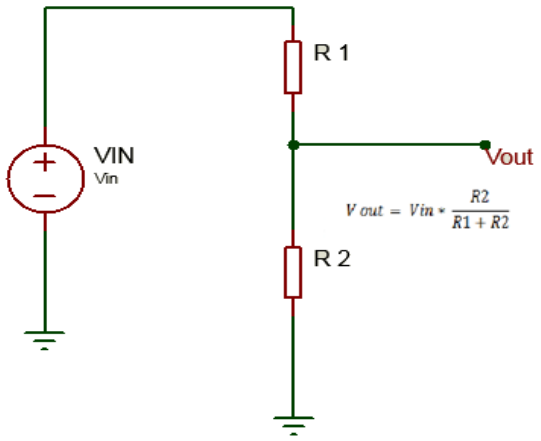

Figura 59. Divisor de tensión para Acondicionamiento de la señal de voltaje

Para la lectura del voltaje negativo se usa la misma configuración del divisor de tensión donde a su salida se tienen valores de Vout negativos. Está claro que la tarjeta no puede recibir directamente valores de voltaje negativos, para lo cual se utiliza un inversor de voltaje.

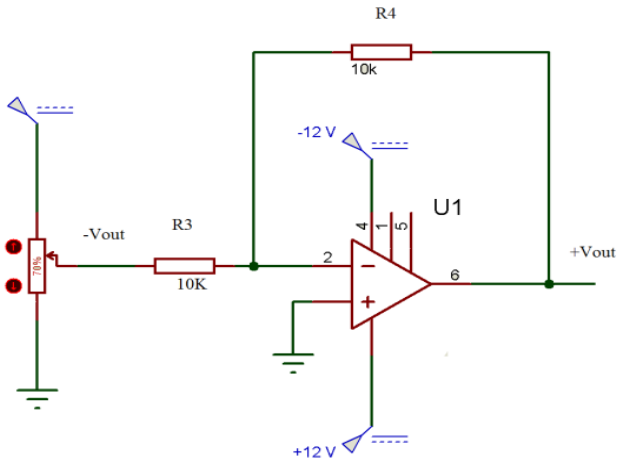

Figura 60. Circuito inversor de voltaje

El diagrama de la lectura de los voltajes positivos y negativos de las fuentes variables del módulo se ilustra en la Figura 61.

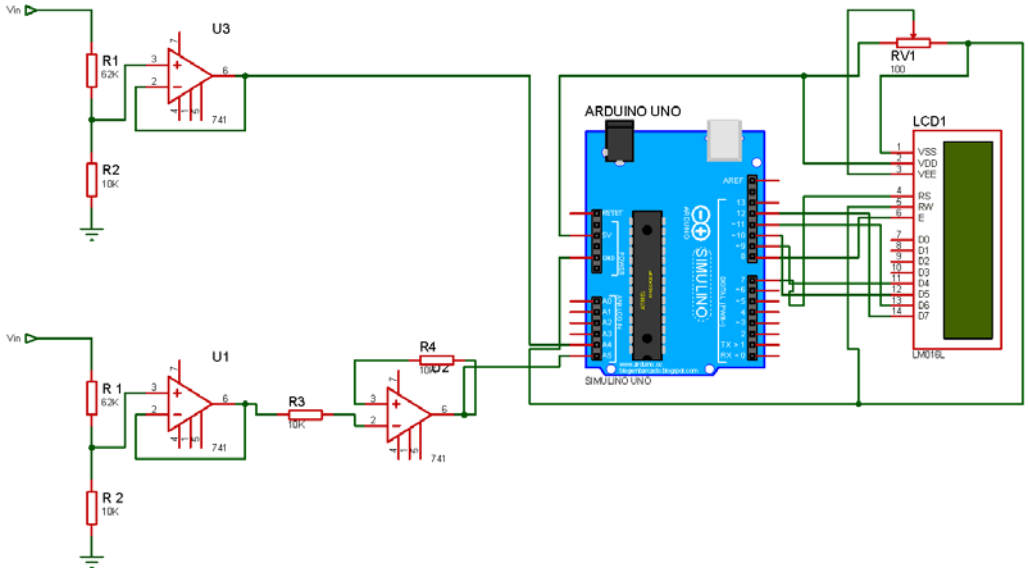

Figura 61. Circuito de lectura de voltajes variables positivo y negativo del módulo

b. Medición de resistencia

El principio para la medición del valor de una resistencia desconocida es el mismo que para la medición de voltaje, se utiliza un divisor de tensión (Figura 62), donde se asume el valor de una de ellas en este caso R2 (se asumió un valor de 1,5KΩ) y R1 sera la resistencia a medir indirectamente a través de la caída de tensión en esta resistencia. La fórmula de salida del divisor de tensión se tomará en cuenta para la programación en Arduino para luego visualizar la medida en la interfaz gráfica del PC.

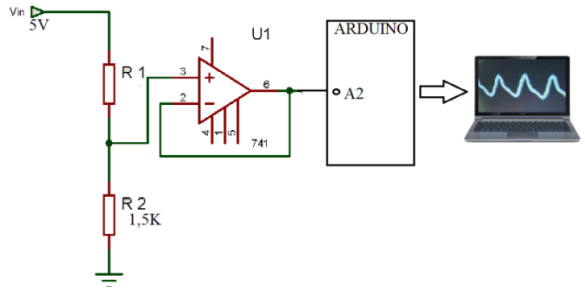

Figura 62. Circuito de acondicionamiento para medición de resistencia

c. Medición de Voltaje continuo positivo y negativo.

Se diseña el acondicionamiento similar al de la lectura de la fuente explicado en el numeral 3.2.3.1. Los valores positivos y negativos están en un solo rango para ser leídos en la entrada analógica A3 del arduino. La relación entre la entrada y la salida se muestra en la Figura 63.

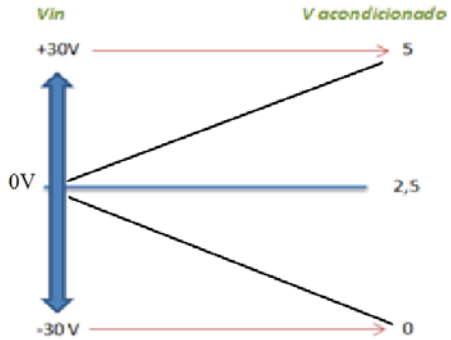

Figura 63. Acondicionamiento para la medición de voltajes positivos y negativos.

Cuando el voltaje de entrada Vin sea +30V el valor acondicionado para la entrada analógica del arduino será aproximadamente 5V, si la entrada es -30V su valor acondicionado será 0V.

Cuando el voltaje a medir varíe entre ±30V en el divisor de tensión la variación será de -2,5V a 2,5V y como se ha explicado anteriormente los valores negativos no podrán ser ingresados directamente al microcontrolador. Para convertir a un rango solo positivo es necesario utilizar un sumador no inversor de dos entradas, en la una se aplica el valor del divisor de voltaje (-2,5 a 2,5V) y en la segunda un voltaje de referencia de 2,5V. Esto hace que el voltímetro pueda medir voltajes positivos hasta 30V en este caso el rango de voltaje acondicionado es de 2,5V a 5V (voltaje de salida de sumador no inversor), y para la medición de voltajes negativos hasta -30V, la salida del simador varía de 0 a 2,5V

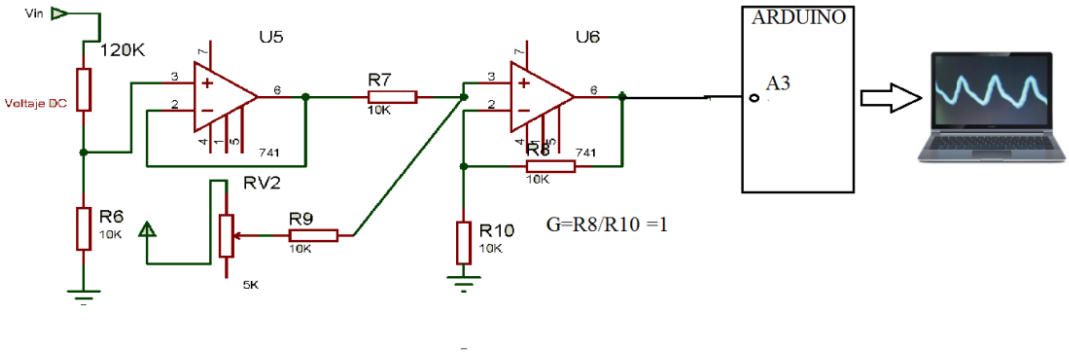

Figura 64. Circuito de acondicionamiento para medición voltajes positivos y negativos

d. Medición de Voltaje Alterno.

De acuerdo a las especificaciones del sensor de voltaje de corriente alterna debe circular una corriente de 200mA, por lo tanto si la entrada de Voltaje alterna es de 110VAC y su corriente es 200mA entonces su resistencia calculada (R=V/I) será aproximadamente de 55KΩ para la cual se utilizó una resistencia de 62KΩ para asegurar que la corriente que circule sea menor al límite especificado. La salida del

sensor proporciona un rango de voltaje continuo de 0 a 5V, formato que puede ser procesada por el microcontrolador.

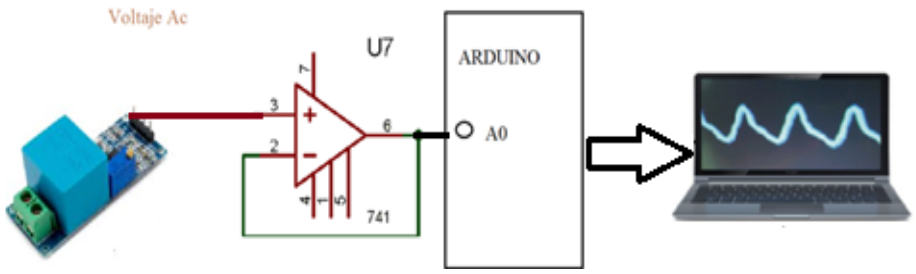

Figura 65. Circuito de acondicionamiento para medición de voltaje alterno

e. Medición de corriente continúa y alterna.

No se requiere un acondicionamiento especial, se conecta la placa sensora por medio de un seguidor de tensión a la tarjeta arduino a la entrada analógica A1.

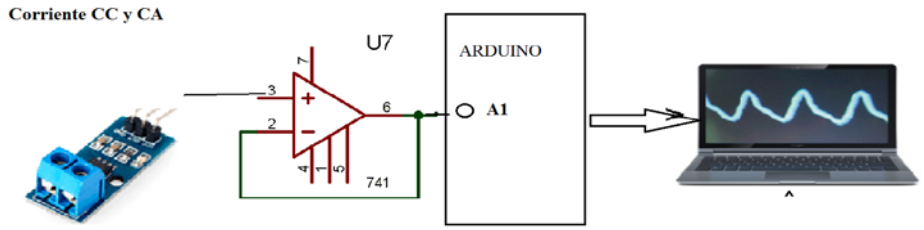

Figura 66. Circuito de acondicionamiento para medición de corriente

A continuación, se muestra el diagrama de conexión del multímetro del módulo didáctico que incluye mediciones de corriente y voltajes de AC y DC así como resistencia.

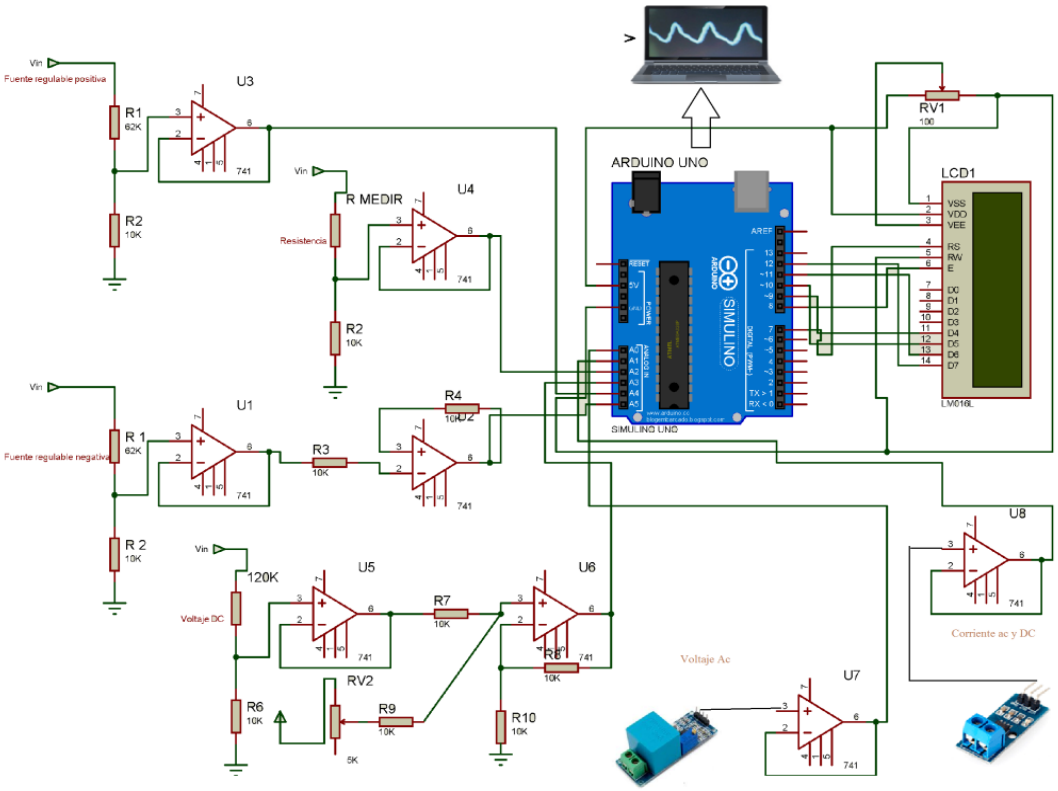

Figura 67. Diagrama de conexión del multímetro

3.2.4. Diseño y construcción del Generador de Funciones

Un generador de funciones es muy útil para las prácticas que se realizarán con el módulo didáctico. El el diagrama de bloques de la siguente figura se muestra el proceso para la generación de ls formas de onda senoidal, cuadrada, rampa, riangular y PWM.

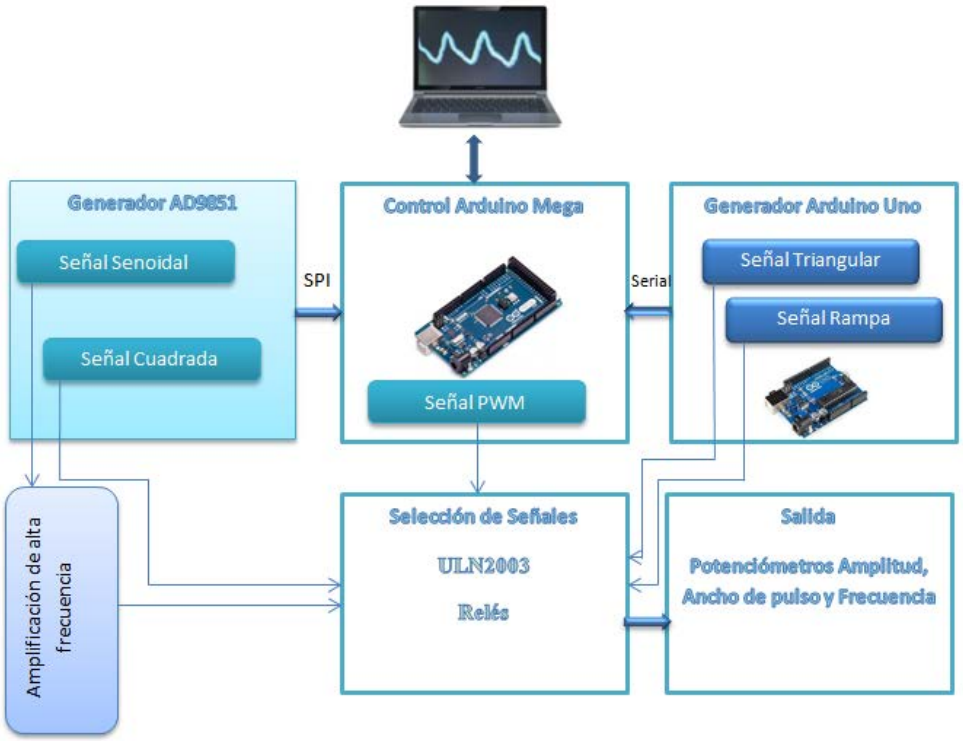

Figura 68. Diagrama de bloques del generador de funciones.

La señal senoidal generada desde el circuito integrado AD9851 tiene una amplitud de 1Vpp. Esta señal es amplificada con un OPA4131 (Amplificador de alta frecuencia) en modo amplificador no inversor como se aprecia en la siguiente Figura.

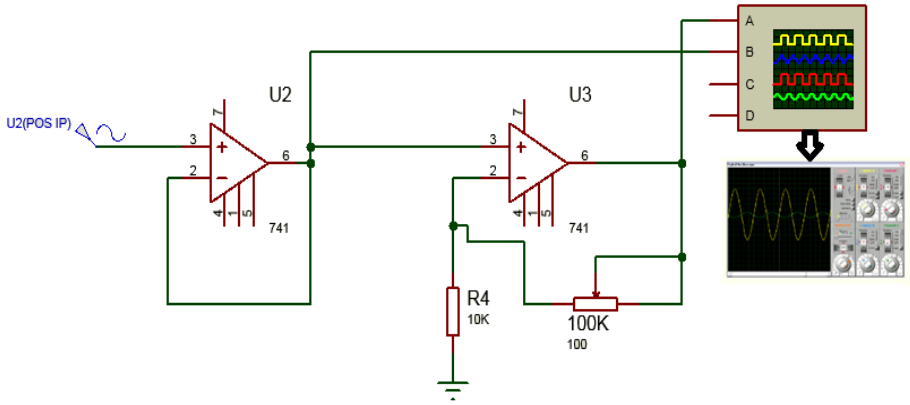

Figura 69. Acondicionamiento para amplificación de la señal senoidal.

Para la señal cuadrada y PWM no es necesario diseñar ningún acondicionamiento debido a que AD9851 genera dichas señales amplificadas (5Vpp). La señal triangular y rampa se generan desde la tarjeta arduino, el control de

sus parámetros como amplitud, frecuencia y ancho de pulso se lo realiza a través de software.

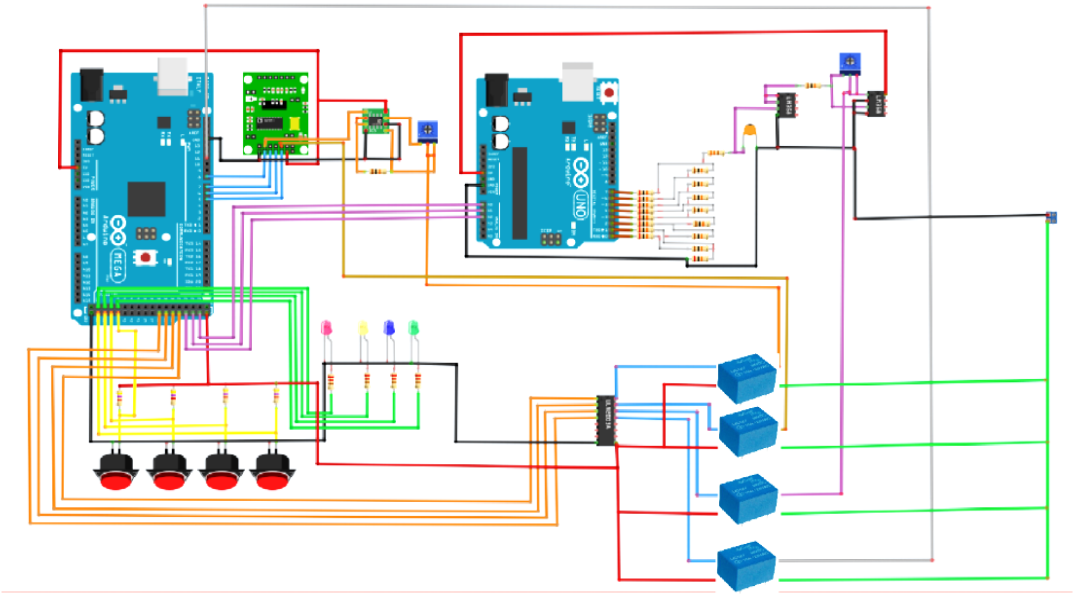

Figura 70. Diagrama de conexión del generador de funciones

3.2.5. Diseño y construcción del Osciloscopio

El Osciloscopio es una herramienta importante dentro del estudio de la electrónica. El diseño de esta función del módulo se basa en la constante adquisición de datos a la entrada de la tarjeta arduino.

Figura 71. Diagrama de bloques del osciloscopio

La interface gráfica fue diseñada para ser lo más intuitiva y familiar posible al usuario, conservando el formato de un osciloscopio tradicional físico. Su operación se basa en la programación en la tarjeta arduino.

3.2.6. Diseño de las placas del módulo didáctico utilizando Proteus (ISIS)

Se procede a la construcción del módulo didáctico, una vez seleccionados los materiales requeridos y la ubicación de los diferentes elementos. Para esto, se usan placas de baquelitas de distintos tamaños, en las que se imprime el diseño del circuito establecido por el Programa Proteus (ISIS), (Figura 72).

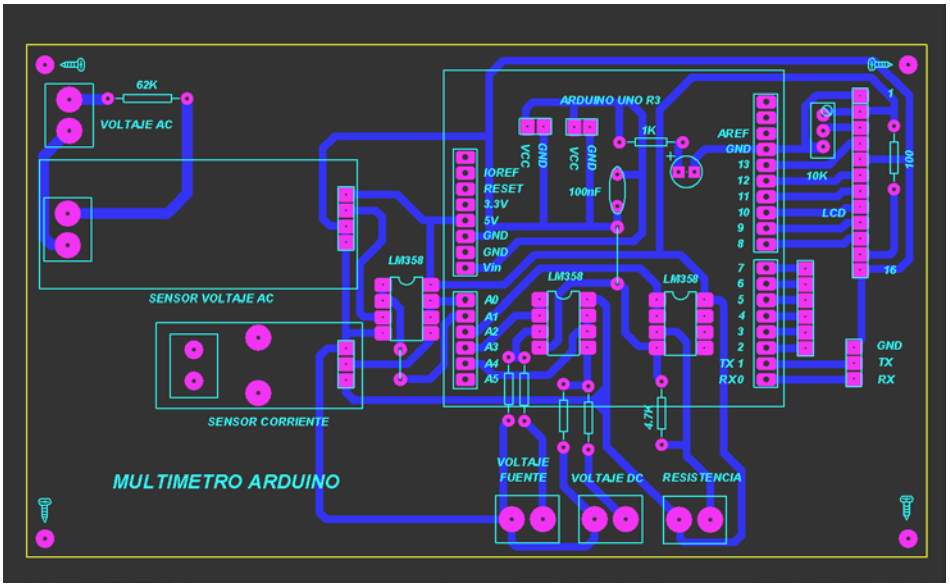

Figura 72. Diseño del circuito en el programa Proteus (ISIS)

3.2.7 Construcción de las placas del módulo didáctico

Luego del diseño se procede con un proceso de impresión del circuito PCB a travez de software, una vez listas las diferentes placas se procede a la colocación de lo elementos y dispositivos electrónicos correspondientes.

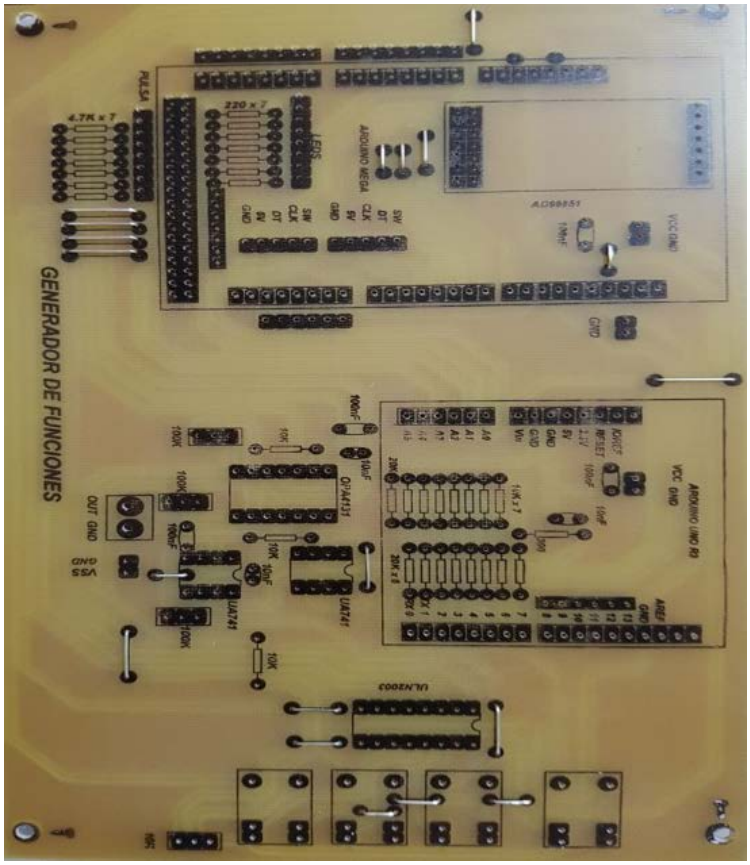

Figura 73. Placa impresa y perforada

3.2.8. Montaje de las placas dentro de la estructura del módulo didáctico

Lista tanto la estructura metálica como las placas electrónicas del módulo con todos los elementos soldados, se procede a la colocación de los mismos sobre una plancha de acrílico cuya función es aislar de la estructura metalica y posibles contactos (Figura 74).

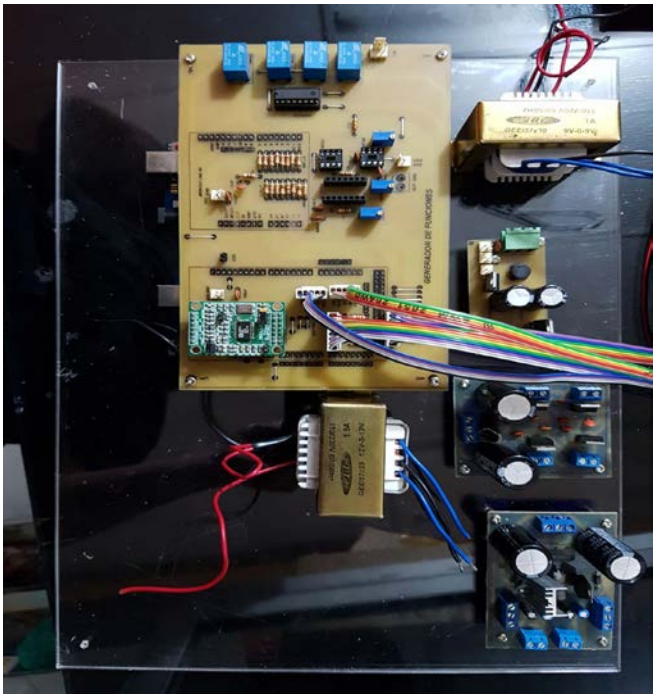

Figura 74. Montaje de placas sobre la plancha de acrílico

Se coloca sobre la plancha de acrílico el resto de placas y las tarjetas para posteriormente ubicarlo dentro de la estructura del módulo didáctico.

Figura 75. Placa de acrílico con elementos colocados

La distribución y detalle de cada una de las partes que conforman el módulo didáctico se indica a continuación: (Figura 76).

1.- Circuitos acondicionadores de señal
2.- Multímetro
3.- Fuentes fijas – 5v , 5v; -12v , 12v
4.- Fuente variable -24v a 24v
5.- Osciloscopio
6.- Generador de funciones

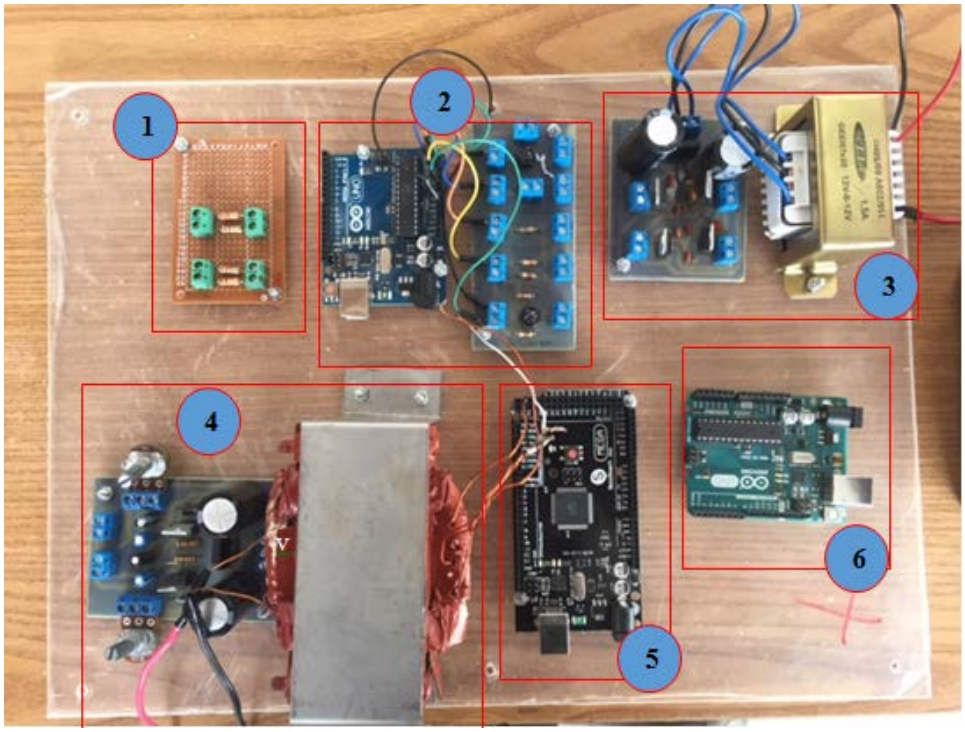

Figura 76. Distribución de las partes del módulo didáctico

Por último, se realiza el cableado entre placas, panel frontal y demás elementos (potenciómetros, borneras y perillas) (Figura 77)

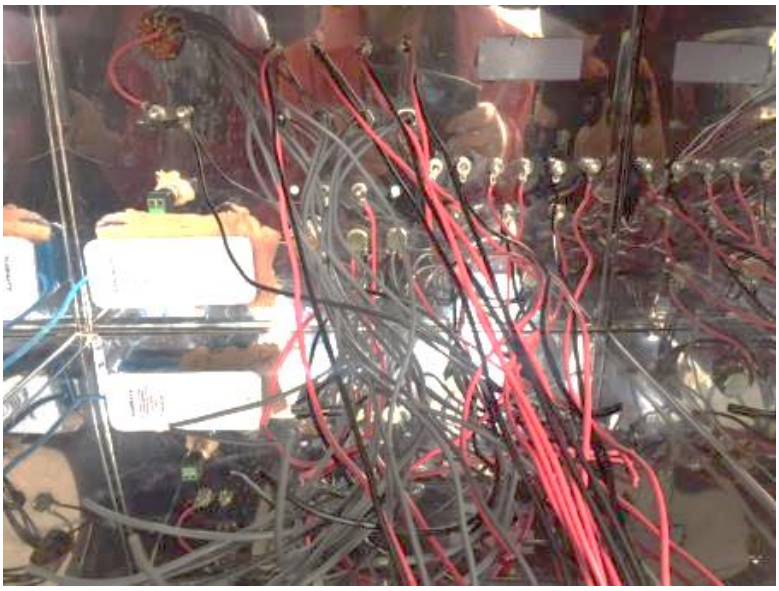

Figura 77. Instalación de cables hacia los elementos de control

3.3. Instalación y Configuración del Software a aplicar

Con el software Arduino para el sistema operativo Windows 7, es necesario desempaquetar el instalador en el PC por primera y única vez, hasta que se cambie de ordenador o vuelva a cargar el Sistema Operativo de nuevo, se crea en el ordenador una carpeta con el nombre Arduino para guardar en ella todo lo que se refiere a la Tarjeta. Dentro de esta carpeta se crean varias subcarpetas donde se irá guardando ordenadamente todos los ficheros que se generen, con o por, el uso de esta tarjeta.

3.3.1. Instalación de Arduino Uno en el Ordenador

Se instala el software (Software IDE de desarrollo), además los controladores de Arduino, (device drivers), software que informa al PC, el tipo de aparato que se le conecta por USB y cómo tiene que manejarlo, la tarjeta Arduino y la PC identifican que debe enviar y que recibe por ese puerto USB, teclados, pantallas, cámaras fotográficas, etc.

El software de instalación para Windows de Arduino se lo obtuvo de la dirección https://www.arduino.cc/en/Main/Donate (Software de Instalación de Arduino, 2017). Una vez que se descarga el programa, se abrirá el siguiente cuadro de dialogo. Dar un click en I Agree para aceptar los términos de la licencia.

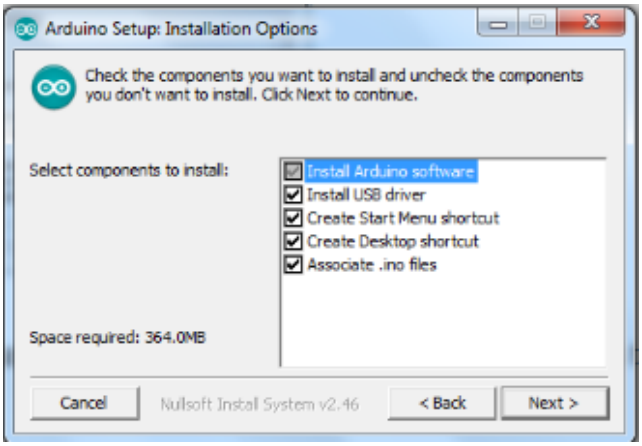

Figura 78. Componentes a Instalar
Fuente: (Software de Instalación de Arduino, 2017)

Click en Next y elegir la ubicación del disco en donde se va a instalar el programa, Clic en Aceptar y la instalación se iniciará. Una vez terminado aparecerá una serie de ventanas para instalar los complementos necesarios para un correcto funcionamiento del software y los dispositivos, por lo tanto, es necesario dar un click en instalar cada vez que aparezca una ventana de diálogo.

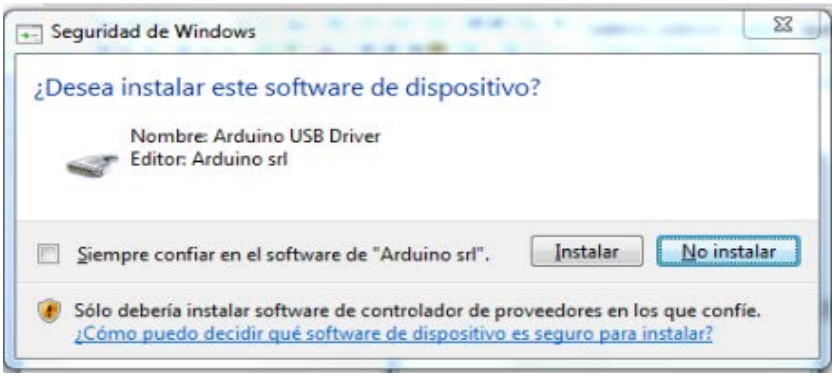

Figura 79. Ventana emergente de instalación
Fuente: (Software de Instalación de Arduino, 2017)

Una vez que termine de instalar el software de todos los dispositivos, aparecerá una pantalla que indica que la instalación está completa, dar un click en Close para continuar. El acceso directo aparecerá en el escritorio al concluir la instalación, dar doble click en el ícono para que el programa se ejecute.

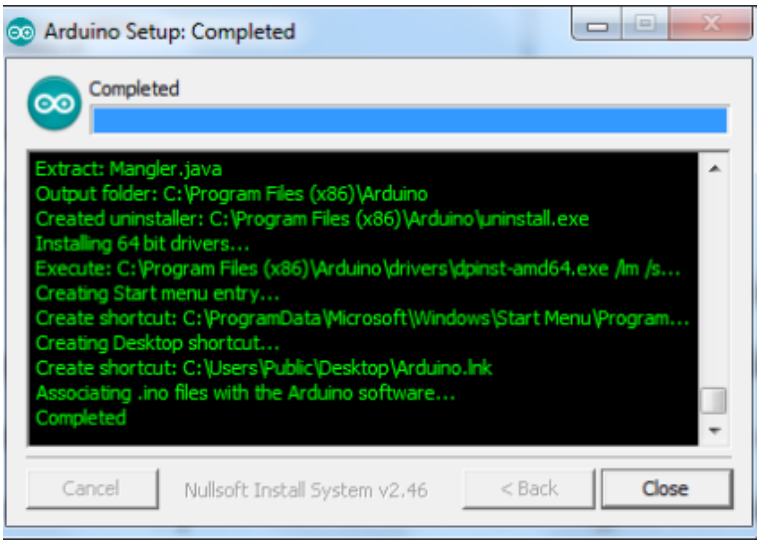

Figura 80. Instalación terminada

Figura 81. Pantalla de inicialización de paquetes
Fuente: (Software de Instalación de Arduino, 2017)

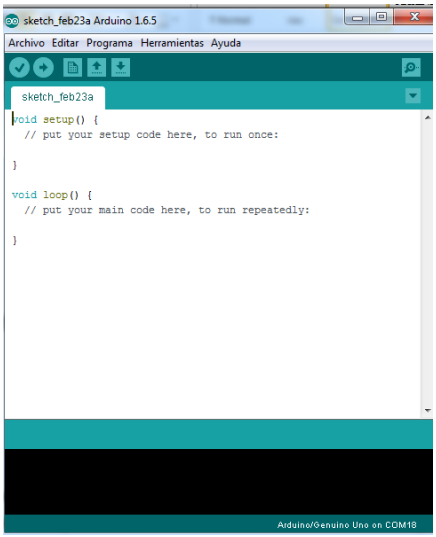

Figura 82. Pantalla Inicial de Arduino Uno
Fuente: (Software de Instalación de Arduino, 2017)

Es necesario tener en cuenta que para iniciar la programación se debe seleccionar el tipo de placa Arduino que se va a utilizar, y el tipo de programador que se utilizará, para este caso es Arduino ISP, como lo muestra las figuras 83 y 84.

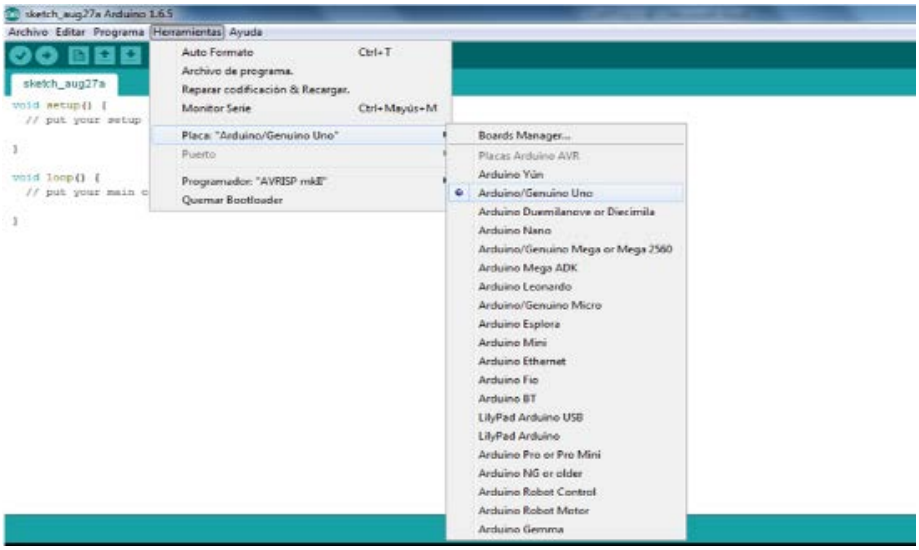

Figura 83. Selección de Placa Arduino Uno
Fuente: (Software de Instalación de Arduino, 2017)

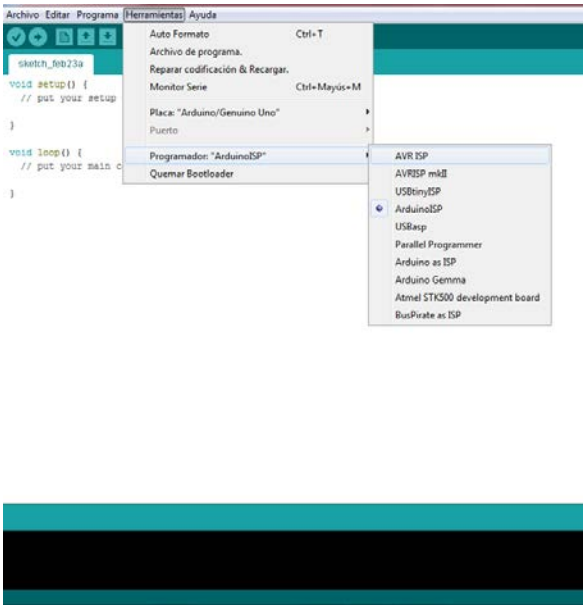

Figura 84. Selección del programador Arduino as ISP
Fuente: (Software de Instalación de Arduino, 2017)

A continuación, conectar la placa Arduino Uno en el puerto USB del ordenador y Windows lo reconocerá automáticamente e instalará los drivers respectivos.

Figura 85. Conexión de Arduino en el Puerto USB
Fuente: (Software de Instalación de Arduino, 2017)

Para comprobar que los controladores se instalaron correctamente y para identificar el puerto en el cual fue asignado, es necesario abrir el administrador de dispositivos/puertos (COM y LPT), para este caso Arduino Uno está asignado en COM3.

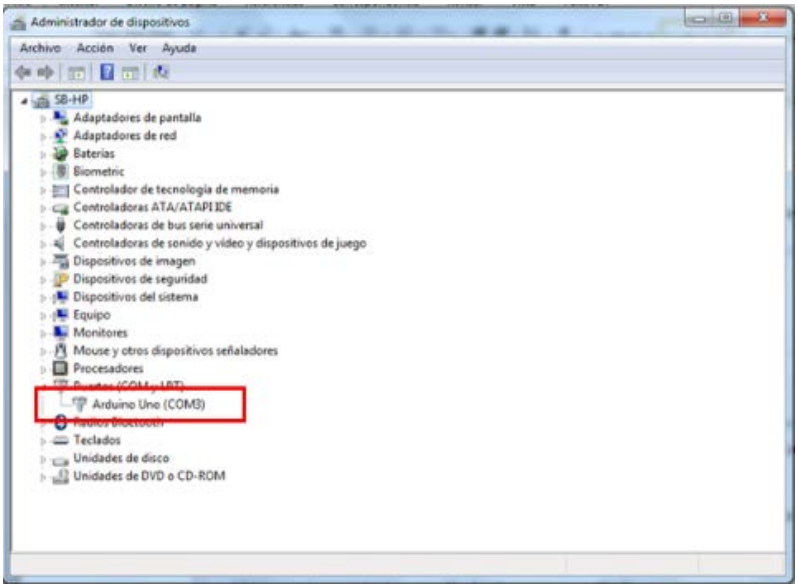

Figura 86. Puerto COM3

Una vez que el controlador se instala correctamente este se podrá visualizar en el programa de Arduino en donde es necesario asegurarse de que el puerto seleccionado sea el mismo que el asignado. (Figura 87).

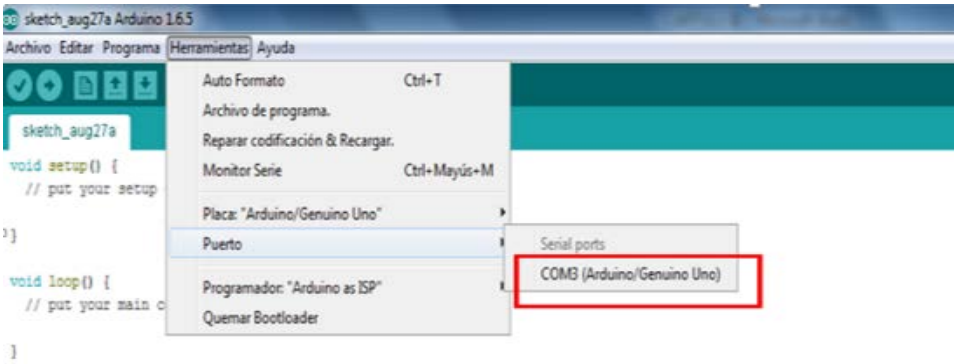

Figura 87. Verificación del Puerto COM3

3.3.2. Diseño del Código de Programación en Arduino

A continuación, se adjunta el código de programación realizado para la comunicación entre Arduino y Python, se utiliza el programa ejecutado para la función del multímetro.

3.3.2.1. Programación en el Arduino para uso del Multímetro.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

int medida;
char serInString[10];
int serInIndx = 0;
float voltTolerancia=1.0;
float volt1anterior=0;
float volt2anterior=0;

void readSerialString () {
  char sb;
  serInIndx = 0;
  if (Serial.available()) {
    delay(200);
    while (Serial.available()) {
      sb = Serial.read();
      serInString[serInIndx] = sb;
      serInIndx++;
    }
  }

  if (strstr(serInString, "VA#")) {
    medida = 1;
  } else if (strstr(serInString, "VD#")) {
    medida = 2;
  } else if (strstr(serInString, "CA#")) {
    medida = 3;
  } else if (strstr(serInString, "R#")) {
    medida = 4;
  } else if (strstr(serInString, "CD#")) {
    medida = 5;
  }  else{
    medida=9;
  }
}
float voltac() // FUNCIÓN QUE DEVUELVE EL VOLTAJE AC
{
   int pico = 0;
  int volt = 0;
  int lectura = 0;
  float salida;
  for (int i = 0; i < 400; i++) {
    lectura = analogRead(1);
    if (lectura < 512) {
```

```
      volt = 512 - lectura;

    } else {
      volt = lectura - 512;
    }
    if (volt > pico) {
      pico = volt;
    }
    if (pico < 18) {
      pico = 0;
    }
    delay(1);
  }
  salida = (227.0 / 509.00) * (float)pico;
  return salida;
}


float volt_dc() {// FUNCIÓN QUE DEVUELVE EL VOLTAJE DC
  unsigned int lectura = 0;
  int medio = 534;
  int volt = 0;
  for (int i = 0; i < 51; i++) {
    lectura = analogRead(3) + lectura;
  }
  lectura = lectura / 51;
  if (lectura < medio) {
      volt = (medio - lectura)*-1;

    } else {
      volt = lectura - medio;
    }

  float salida = (33.20 / (float)medio) * volt;
  return salida;
}


float corriente() {// FUNCIÓN QUE DEVUELVE CORRIENTE AC Y DC
  int pico = 0;
  int volt = 0;
  int lectura = 0;
  float salida;
  int medio = 510;
  for (int i = 0; i < 500; i++) {

      lectura = analogRead(0);
```

```
    if (lectura < medio) {
      volt = medio - lectura;

    } else {
      volt = lectura - medio;
    }
    if (volt > pico) {
      pico = volt;
    }
    delay(1);
  }
  salida = (24.20 / (float)medio) * (float)pico;
  if(salida<0.2){salida=0;}
  return salida;
}


float resistencia() {// FUNCIÓN QUE DEVUELVE EL VALOR DE RESISTENCIA
  unsigned int lectura = 0;
  for (int i = 0; i < 51; i++) {
    lectura = analogRead(2) + lectura;
  }
  lectura = lectura / 51;


  float vo = (5.00 / 1023.000) * (float)lectura * 100;
  float salida =  (1480.0 * (500.0 - (float)vo)) / (float)vo;
  if(vo<1){salida=0;}
  return salida;
}



void volt_fuentes() {// FUNCIÓN QUE DEVUELVE EL VOLTAJE DC
  unsigned int lectura = 0;
  for (int i = 0; i < 51; i++) {
    lectura = analogRead(4) + lectura;
  }
  lectura = lectura / 51;
  float volt1 = (37.18 / 1023.00) * (float)lectura*1.0;

  lectura = 0;
  for (int i = 0; i < 51; i++) {
    lectura = analogRead(5) + lectura;
  }
```

```
  lectura = lectura / 51;
  float volt2 = (35.98 / 1023.00) * (float)lectura*-1.00;

  if (abs(volt1 - volt1anterior) > voltTolerancia){
    volt1anterior=volt1;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("VOLT DC+");
    lcd.setCursor(10, 0);
    lcd.print(volt1,1);
    lcd.setCursor(15, 0);
    lcd.print("V");
    lcd.setCursor(0, 1);
    lcd.print("VOLT DC-");
    lcd.setCursor(9, 1);
    lcd.print(volt2,1);
    lcd.setCursor(15, 1);
    lcd.print("V");
    delay(200);
  }

  if (abs(volt2 - volt2anterior) > voltTolerancia){
    volt2anterior=volt2;
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("VOLT DC-");
    lcd.setCursor(9, 1);
    lcd.print(volt2,1);lcd.setCursor(0, 0);
    lcd.print("VOLT DC+");
    lcd.setCursor(10, 0);
    lcd.print(volt1,1);
    lcd.setCursor(15, 0);
    lcd.print("V");
    lcd.setCursor(15, 1);
    lcd.print("V");
    delay(200);
  }

}

void setup() {

  Serial.begin(115200);// activacion del puerto serial y la velocidad

  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("VOLTAJE DC+ 0 V");
  lcd.setCursor(0, 1);
```

```
  lcd.print("VOLTAJE DC- 0 V");
}

void loop() {

  if (Serial.available() != 0) { //verifica si le llego algun comando por serial

    readSerialString ();
  }

  volt_fuentes();

  if (medida == 1)// VERIFICA SI LA SELECCIÓN ES VOLTAJE AC
  {
    Serial.print("VA#");
    Serial.print(voltac());
    Serial.println(" V");
    delay(500);
  }

  if (medida == 2) // VERIFICA SI LA SELECCIÓN ES VOLTAJE DC
  {
    Serial.print("VD#");
    Serial.print(volt_dc());
    Serial.println(" V");
    delay(500);
  }

  if (medida == 3)// VERIFICA SI LA SELECCIÓN ES CORRIENTE AC
  {
    Serial.print("CA#");
    Serial.print(corriente());
    Serial.println(" A");
    //delay(500);
  }

  if (medida == 4) // VERIFICA SI LA SELECCION ES RESISTENCIA
  {
    Serial.print("R#");
    Serial.print(resistencia());
    Serial.println(" Ohm");
    delay(500);
  }

  if (medida == 5)// VERIFICA SI LA SELECCION ES CORRIENTE AC
  {
    Serial.print("CD#");
    Serial.print(corriente());
```

```
    Serial.println(" A");
    //delay(500);
  }

}
```

Una vez terminado el programa es necesario compilarlo y cargarlo al Arduino.

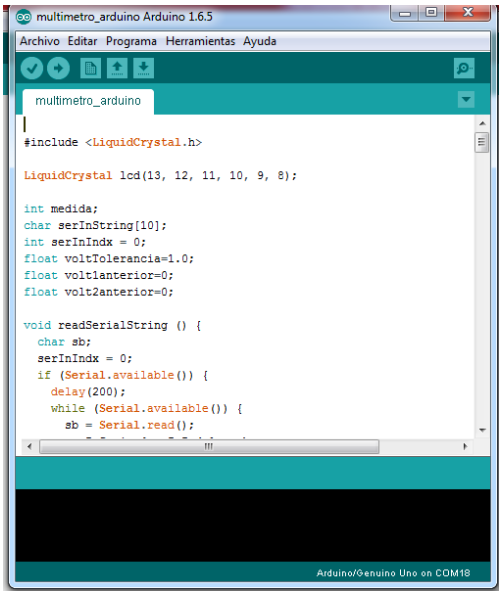

Figura 88. Código de Programación del Multímetro en Arduino.

3.3.3. Código de programación para el generador de funciones en Arduino

a. Programación en Arduino Mega para señal senoidal cuadrada y PWM

```
#include <PWM.h>
int32_t Pwmfrec = 20;
int pwmduty=127;

char serInString[15];
int serInIndx = 0;
int funsion=0;

unsigned long newfrec=0;
int pulsos=10;
int newpulsos=10;
```

```
int relecua=35;
int relesen=33;
int reletri=31;
int relepwm=29;
int cnt=0;
int aux=0;
int ledcua=46;
int ledsen=52;
int ledtri=48;
int ledram=50;
int ledpwm=44;

int pulsacua=45;
int pulsasen=49;
int pulsatri=51;
int pulsaram=47;
int pulsapwm=43;

const unsigned long max_frecuencia_step = 100000;
const unsigned long max_frecuencia = 1500000;
const int min_frecuencia=10;
unsigned long last_frecuencia = 10;
unsigned long frecuencia_step = 1;

// Rotary encoder
const int EncoderPinCLK = 2;
const int EncoderPinDT = 3;
const int EncoderPinSW = 4;
const int EncoderPinCLK2 = 20;
const int EncoderPinDT2 = 21;
const int EncoderPinSW2 = 9;
byte dds_RESET = 5;
byte dds_DATA  = 6;
byte dds_LOAD  = 7;
byte dds_CLOCK = 8;
int inByte=0;
unsigned volatile long frecuencia = 10;
void isr ()  {
  static unsigned long lastInterruptTime = 0;
  unsigned long interruptTime = millis();
  if (interruptTime - lastInterruptTime > 5) {
    if (digitalRead(EncoderPinDT) == LOW)
    {

      frecuencia=frecuencia+frecuencia_step ;
          }
    else {
      frecuencia=frecuencia-frecuencia_step ;
```

```
      newpulsos--;
    }
    frecuencia = min(max_frecuencia, max(min_frecuencia, frecuencia));
    if(newpulsos>1000){newpulsos=1000;}
    if(newpulsos<1){newpulsos=1;}
    lastInterruptTime = interruptTime;
  }
}
void isr2 ()  {
  static unsigned long lastInterruptTime2 = 0;
  unsigned long interruptTime2 = millis();
  if (interruptTime2 - lastInterruptTime2 > 5) {
    if (digitalRead(EncoderPinDT2) == LOW)
    {
     pwmduty--;
         }
    else {
      pwmduty++;
    }
   if(pwmduty>255){pwmduty=255;}
   if(pwmduty<1){pwmduty=1;}

    lastInterruptTime2 = interruptTime2;
  }
}
void show_frecuencia()
{
  int ceros=0;
    Serial.print("FRE#");
    if(frecuencia<10){
       ceros=6;
    }else if(frecuencia<100){
       ceros=5;
    }else if(frecuencia<1000){
       ceros=4;
    }else if(frecuencia<10000){
       ceros=3;
    }else if(frecuencia<100000){
       ceros=2;
    }else if(frecuencia<1000000){
       ceros=1;
    }
    for(int i=0;i<ceros;i++){
     Serial.print("0");
    }
    Serial.print(frecuencia);
    Serial.println(" Hz");
}
```

```
void setup_dds()
{
  // DDS pins for data, clock and load
  pinMode (dds_DATA, OUTPUT);
  pinMode (dds_CLOCK, OUTPUT);
  pinMode (dds_LOAD, OUTPUT);
  pinMode (dds_RESET, OUTPUT);
  digitalWrite(dds_DATA, LOW);
  digitalWrite(dds_CLOCK, LOW);
  digitalWrite(dds_LOAD, LOW);
  digitalWrite(dds_RESET, LOW);

  delay (2000);
  init_dds();
  reset_dds();
}
void init_dds()
{
  digitalWrite(dds_RESET, LOW);
  digitalWrite(dds_CLOCK, LOW);
  digitalWrite(dds_LOAD, LOW);
  digitalWrite(dds_DATA, LOW);
}
void reset_dds()
{
    digitalWrite(dds_CLOCK, LOW);
    digitalWrite(dds_LOAD, LOW);

     digitalWrite(dds_RESET, LOW);
     delay(5);
     digitalWrite(dds_RESET, HIGH);
     delay(5);
     digitalWrite(dds_RESET, LOW);
     delay(5);

     digitalWrite(dds_CLOCK, LOW);
     delay(5);
     digitalWrite(dds_CLOCK, HIGH);
     delay(5);
     digitalWrite(dds_CLOCK, LOW);
     delay(5);
     digitalWrite(dds_DATA, LOW);

     digitalWrite(dds_LOAD, LOW);
     delay(5);
     digitalWrite(dds_LOAD, HIGH);
     delay(5);
     digitalWrite(dds_LOAD, LOW);
```

```
}
void dds(unsigned long freq)
{
 int last8;
 unsigned long DDSLong;
 unsigned long Bitmask32 = 1;
 byte Bitmask8 = 1;
 byte FirstBit = 1;

 float clock_frecuencia = 180000000;
 float twoE32 = pow (2,32);

 DDSLong = ((twoE32 * (freq))/ clock_frecuencia);
 for (Bitmask32 = 1; Bitmask32 > 0; Bitmask32 <<= 1)
 {
  if (DDSLong & Bitmask32)
    digitalWrite(dds_DATA,HIGH);
  else
  digitalWrite(dds_DATA,LOW);
  digitalWrite(dds_CLOCK,HIGH);
  delayMicroseconds(1);
  digitalWrite(dds_CLOCK,LOW);
 }

 for (Bitmask8 = 1; Bitmask8 > 0; Bitmask8 <<= 1)
 {
  if (Bitmask8 & FirstBit
    digitalWrite(dds_DATA,HIGH);
  else
    digitalWrite(dds_DATA,LOW);

  digitalWrite(dds_CLOCK,HIGH);
  delayMicroseconds(1);
  digitalWrite(dds_CLOCK,LOW);
 }

 digitalWrite (dds_LOAD, HIGH);
 delayMicroseconds(1);
 digitalWrite (dds_LOAD, LOW);
  return;
}
void decodenum(){
   newfrec=(serInString[4]-48)*1000000;
 newfrec=newfrec+(serInString[5]-48)*100000;
 newfrec=newfrec+(serInString[6]-48)*10000;
 newfrec=newfrec+(serInString[7]-48)*1000;
 newfrec=newfrec+(serInString[8]-48)*100;
 newfrec=newfrec+(serInString[9]-48)*10;
```

```
  newfrec=newfrec+(serInString[10]-48);
    for (int i=0;i<15;i++){
    serInString[i]=0;
  }
}
void readSerialString () {
  char sb;
  serInIndx = 0;
  if (Serial.available()) {
    delay(200);
    while (Serial.available()) {
      sb = Serial.read();
      serInString[serInIndx] = sb;
      serInIndx++;
    }
  }
  if (strstr(serInString, "SEN")) {
   funsion=0;
   decodenum();
   frecuencia=newfrec;
  } else if (strstr(serInString, "CUA")) {
    decodenum();
   funsion=1;
   frecuencia=newfrec;
  } else if (strstr(serInString, "TRI")) {
   decodenum();
   funsion=2;
   if (newfrec<9000){
     newpulsos=newfrec/7;
   }
   Serial.println(newpulsos);
  } else if (strstr(serInString, "RAM")) {
   funsion=3;
   decodenum();
   if (newfrec<9000){
     newpulsos=newfrec/7;
   }
   Serial.println(newpulsos);
  } else if (strstr(serInString, "PWM")) {
   funsion=4;
   decodenum();
   Pwmfrec=newfrec;
   setuppwm();
  }
}
void setuppwm(){

    InitTimersSafe();
```

```
  bool success = SetPinFrequencySafe(11, Pwmfrec);
  if(success) {
    pinMode(11, OUTPUT);
    digitalWrite(11, HIGH);
  }
  }
void setup() {
  Serial.begin(115200);

  pinMode(EncoderPinCLK, INPUT);
  pinMode(EncoderPinDT, INPUT);

  pinMode(relecua, OUTPUT);
  pinMode(relesen, OUTPUT);
  pinMode(reletri, OUTPUT);
  pinMode(relepwm, OUTPUT);
  pinMode(ledcua, OUTPUT);
  pinMode(ledsen, OUTPUT);
  pinMode(ledtri, OUTPUT);
  pinMode(ledram, OUTPUT);
  pinMode(ledpwm, OUTPUT);
  pinMode(pulsacua, INPUT);
  pinMode(pulsasen, INPUT);
  pinMode(pulsatri, INPUT);
  pinMode(pulsaram, INPUT);
  pinMode(pulsapwm, INPUT);
  pinMode(EncoderPinSW, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(EncoderPinCLK), isr, LOW);

  pinMode(EncoderPinCLK2, INPUT);
  pinMode(EncoderPinDT2, INPUT);

    attachInterrupt(digitalPinToInterrupt(EncoderPinCLK2), isr2, LOW);
    setup_dds();
  show_frecuencia();
  dds(frecuencia);
  pinMode(22, OUTPUT);
  pinMode(23, OUTPUT);
  pinMode(24, OUTPUT);
  pinMode(25, OUTPUT);
  pinMode(12,OUTPUT);
  digitalWrite (22, HIGH);
  digitalWrite (23, HIGH);
  digitalWrite (24, HIGH);
  digitalWrite (25, HIGH);
          setuppwm();
}
void loop() {
```

```
  if (Serial.available() > 0) {
   readSerialString ();
 }
pwmWrite(11, pwmduty);
if(digitalRead(pulsasen)==LOW){
 funsion=0;
}
 if((digitalRead(pulsacua)==HIGH)&&(aux==0)){aux=1;}
 if((digitalRead(pulsacua)==LOW)&&(aux==1)){
   aux=2;
   cnt++;
   if(cnt==1){funsion=1;}
   if(cnt>1){
    cnt=0;
    funsion=4;
   }
   delay(100);
}
if((digitalRead(pulsacua)==LOW)&&(aux==2)){aux=0;}
  if(digitalRead(pulsatri)==LOW){
 funsion=2;
}
 if(digitalRead(pulsaram)==LOW){
 funsion=3;
}
if (funsion==0){
  digitalWrite (relesen, HIGH);
  digitalWrite (relecua, LOW);
  digitalWrite (reletri, LOW);
  digitalWrite (relepwm, LOW);
  digitalWrite (ledsen, HIGH);
  digitalWrite (ledcua, LOW);
  digitalWrite (ledtri, LOW);
  digitalWrite (ledram, LOW);
  digitalWrite (ledpwm, LOW);
  funsion=9;
}else if (funsion==1){
  digitalWrite (relesen, LOW);
  digitalWrite (relecua, HIGH);
  digitalWrite (reletri, LOW);
  digitalWrite (relepwm, LOW);
  digitalWrite (ledsen, LOW);
  digitalWrite (ledcua, HIGH);
  digitalWrite (ledtri, LOW);
  digitalWrite (ledram, LOW);
  digitalWrite (ledpwm, LOW);
  funsion=9;
}else if (funsion==2) {
```

```
  digitalWrite (22, LOW);
  delay(10);
  digitalWrite (22, HIGH);
  digitalWrite (relesen, LOW);
  digitalWrite (relecua, LOW);
  digitalWrite (reletri, HIGH);
  digitalWrite (relepwm, LOW);
  digitalWrite (ledsen, LOW);
  digitalWrite (ledcua, LOW);
  digitalWrite (ledtri, HIGH);
  digitalWrite (ledram, LOW);
  digitalWrite (ledpwm, LOW);
  funsion=7;
 }else if (funsion==3) {
  digitalWrite (23, LOW);
  delay(10);
  digitalWrite (23, HIGH);
  digitalWrite (22, HIGH);
  digitalWrite (relesen, LOW);
  digitalWrite (relecua, LOW);
  digitalWrite (reletri, HIGH);
  digitalWrite (relepwm, LOW);
  digitalWrite (ledsen, LOW);
  digitalWrite (ledcua, LOW);
  digitalWrite (ledtri, LOW);
  digitalWrite (ledram, HIGH);
  digitalWrite (ledpwm, LOW);
  funsion=7;
 }else if (funsion==4){
  digitalWrite (relesen, LOW);
  digitalWrite (relecua, LOW);
  digitalWrite (reletri, LOW);
  digitalWrite (relepwm, HIGH);
  digitalWrite (ledsen, LOW);
  digitalWrite (ledcua, LOW);
  digitalWrite (ledtri, LOW);
  digitalWrite (ledram, LOW);
  digitalWrite (ledpwm, HIGH);
  funsion=9;
 }
 if (funsion==7){
       if(pulsos<newpulsos){
     for(int i=pulsos;i<newpulsos;i++){
       digitalWrite (24, LOW);
       delay(10);
       digitalWrite (24, HIGH);
       delay(10);
     }
```

```
   }else if(pulsos>newpulsos){
     for(int i=newpulsos;i<pulsos;i++){
        digitalWrite (25, LOW);
        delay(10);
        digitalWrite (25, HIGH);
        delay(10);
     }
   }
   pulsos=newpulsos;
    }
  if ((!digitalRead(EncoderPinSW))) {
   while (!digitalRead(EncoderPinSW))
    delay(10);
   Serial.println("Reset");
   if (frecuencia_step==max_frecuencia_step)
   {
    frecuencia_step=1;
   }
   else
   {
    frecuencia_step=frecuencia_step*10;
   }

    }
  if (frecuencia != last_frecuencia) {
      show_frecuencia();
   dds(frecuencia);
   last_frecuencia = frecuencia ;
  }
}
```

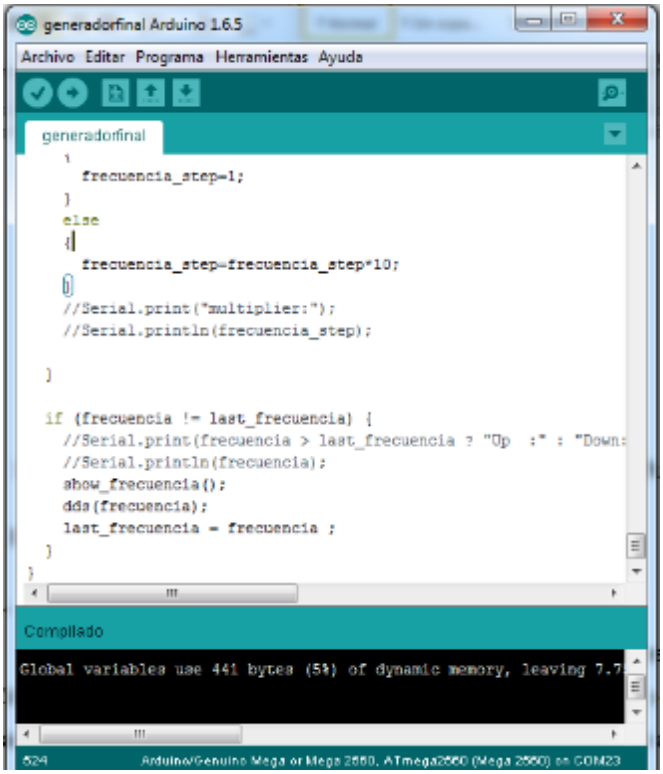

Figura 89. Código de Programación del generador de funciones Arduino Mega

3.3.3.2. Programación en Arduino Uno Rev3 para la señal triangular y rampa

```
byte shape = 0;
int frequency;
int freqCurrent;
byte freqTolerance = 2;
float freqscaled;
byte wave;
long samplerate;
float period;

float sawByte = 0;
float sawInc;
float triByte = 0;
float triInc;

const float freqTable[] = {
 16.351 , 17.324 , 18.354 , 19.445 , 20.601 , 21.827 , 23.124 , 24.499 ,
25.956 , 27.5 , 29.135 , 30.868 ,
 32.703 , 34.648 , 36.708 , 38.891 , 41.203 , 43.654 , 46.249 , 48.999 ,
51.913 , 55 , 58.27 , 61.735 ,
 65.406 , 69.296 , 73.416 , 77.782 , 82.407 , 87.307 , 92.499 , 97.999 ,
103.826 , 110 , 116.541 , 123.471 ,
```

```
 130.813 , 138.591 , 146.832 , 155.563 , 164.814 , 174.614 , 184.997 ,
195.998 , 207.652 , 220    , 233.082 , 246.942 ,
 261.626 , 277.183 , 293.665 , 311.127 , 329.628 , 349.228 , 369.994 ,
391.995 , 415.305 , 440    , 466.164 , 493.883 ,
 523.251 , 554.365 , 587.33  , 622.254 , 659.255 , 698.456 , 739.989 ,
783.991 , 830.609 , 880    , 932.328 , 987.767 ,
 1046.502 , 1108.731 , 1174.659 , 1244.508 , 1318.51  , 1396.913 , 1479.978 ,
1567.982 , 1661.219 , 1760    , 1864.655 , 1975.533 ,
 2093.005 , 2217.461 , 2349.318 , 2489.016 , 2637.021 , 2793.826 , 2959.955 ,
3135.964 , 3322.438 , 3520    , 3729.31  , 3951.066 ,
 4186.009 , 4434.922 , 4698.636 , 4978.032 , 5274.042 , 5587.652 , 5919.91  ,
6271.928 , 6644.876 , 7040    , 7458.62  , 7902.132 ,
 8372.018 , 8869.844 , 9397.272 , 9956.064 , 10548.084, 11175.304, 11839.82 ,
12543.856, 13289.752, 14080   , 14917.24 , 15804.264
};

void pciSetup(byte pin)
{
  *digitalPinToPCMSK(pin) |= bit (digitalPinToPCMSKbit(pin));
  PCIFR  |= bit (digitalPinToPCICRbit(pin));
  PCICR  |= bit (digitalPinToPCICRbit(pin));
}
ISR (PCINT1_vect) // handle pin change interrupt for A0 to A5 here
{
  if (digitalRead(A0) == LOW)
  {
       shape = 0;
  }
  if (digitalRead(A1) == LOW)
  {
    shape = 1;
  }
  if (digitalRead(A2) == LOW)
  {
    freqCurrent=freqCurrent+2;
    if (freqCurrent>=1020)freqCurrent=1000;
  }
  if (digitalRead(A3) == LOW)
  {
    freqCurrent=freqCurrent-2;
    if (freqCurrent<=1)freqCurrent=1;
  }
}
void checkFreq() {
  if (abs(freqCurrent - frequency) > freqTolerance)
  {
    frequency = freqCurrent;
      freqscaled = frequency / 8.5;
```

```
    if (freqscaled <= 0) freqscaled = 0;
    if (freqscaled > 119) freqscaled = 119;
    freqscaled = freqTable[(int)freqscaled];
    period = samplerate / freqscaled;
    triInc = 511 / period;
    if (triInc == 0)
    {
      triInc = 1;
    }
    sawInc = 255 / period;
    if (sawInc == 0)
    {
      sawInc = 1;
    }
  }
}
ISR(TIMER1_COMPA_vect)
{
  static long t;
  t += 1;
  if (t >= period)
  {
    t = 0;
  }
  switch (shape) {
    case 0://triangle
     if ((period - t) > t)
      {
        if (t == 0)
        {
          triByte = 0;
        }
        else
        {
          triByte += triInc;
        }
      }
      else  //high values of t
      {
        triByte -= triInc;
      }
      if (triByte > 255)
      {
        triByte = 255;
      }
      else if (triByte < 0)
      {
        triByte = 0;
```

```
    }
    wave = triByte * 0.75 + 31;
        break;

case 1:
    if (t == 0) {
      sawByte = 0;
    }
    else {
      sawByte += sawInc;
    }
       wave = sawByte * 0.75 + 31;
         break;
 }
 PORTD = wave;
}void setup()
{
 Serial.begin(115200);
 DDRD = 0xFF;
 DDRC = 0x00;
 DDRB = 0xFF;

 cli();//disabilita interrupciones
 //timer 1:
 TCCR1A = 0;
 TCCR1B = 0;

 samplerate = 100000;
 OCR1A = 16000000 / samplerate - 1;
 TCCR1B |= (1 << WGM12);
 TCCR1B |= (1 << CS10);
 TIMSK1 |= (1 << OCIE1A);
 PORTB = 0;
 PORTB = 1 << shape;
 pciSetup(A0);
 pciSetup(A1);
 pciSetup(A2);
 pciSetup(A3);
   freqCurrent=1;
 frequency = 10;
 freqscaled = frequency / 8.5;
 if (freqscaled <= 0) freqscaled = 0;
 if (freqscaled > 119) freqscaled = 119;
 freqscaled = freqTable[(int)freqscaled];
 period = samplerate / freqscaled;

 triInc = 512 / period;
 sawInc = 256 / period;
```

```
  sei();//enable interrupts
}
void loop() {

  checkFreq();
  PORTB = 1 << shape;}
```

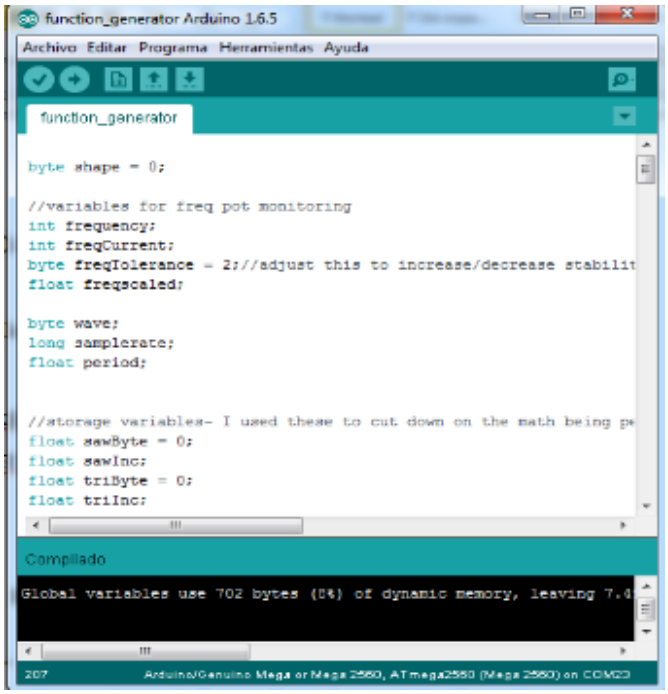

Figura 90. Ejecución del programa función Generador

3.3.4. Código de Programación en el Arduino para uso del Osciloscopio

```
int inPin = 0; //Definicion de entrada 0

void setup(){

        Serial.begin(9600); //Define parametros de comunicacion serial}

void loop() // Lazo sin fin para lectura continua{

        float in = getVoltage(inPin); //Asigna como variable de punto flotante a la
        entrada y lee el pin 0

                in = (in - 2.25); // Escala

        Serial.println(in,DEC); //Ubica en el puerto la lectura acondicioinada
```

```
delay(250); // Tiempo de espera (con este valor cambiar el tiempo de
adquisicion de datos)

}

float getVoltage(int pin){

return (analogRead(pin) * .004882814); //lecturas de 0 a 1024 por resolucion

}
```

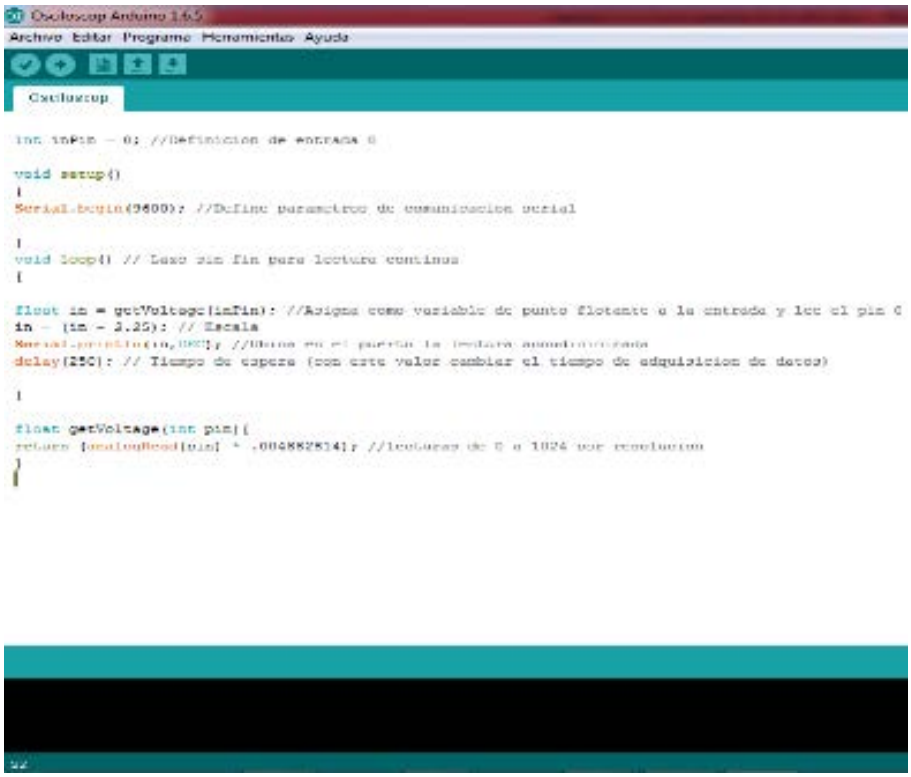

Figura 91. Ejecución del programa función Osciloscopio

3.3.5. Código para leer el puerto serie de Arduino con Python y Pyserial

Para iniciar es necesario colocar en el Arduino un código muy simple, que genera un número aleatorio, de -100 a 100, y lo escribe en el puerto serie.

```
int x = 0;    // variable
void setup() {
Serial.begin(9600);      // abre el puerto serie a 9600 bps:
```

```
}
void loop() {
Serial.println(random(-100, 100)); // Escribe en el puerto un numero aleatorio de -
100 a 100
delay(2000);
}
```

3.4. Instalación del programa PyCcharm para la interfaz Gráfica

3.4.1. Implementación de la Interfaz Gráfica

Dercargar e instalar el programa con el que de creará la interfaz gráfica, en el siguiente link se puede hacer: https://www.jetbrains.com/pycharm-edu/download/#section=windows. Donde Una vez ejecutado e instalado aparece una ventana similar a la siguiente:

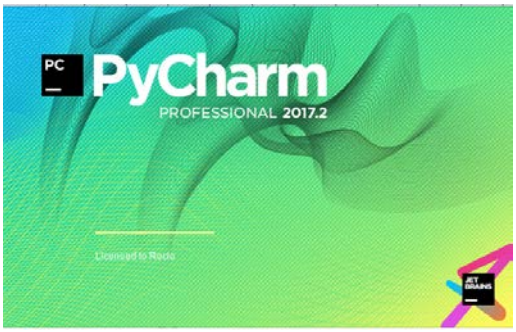

Figura 92. Ejecución del programa PyCharm

3.4.2. Código de programación en PyCharm

```
# coding=utf-8
import glob
import sys, os
import serial
import threading
from PyQt4 import QtGui, QtCore
```

```
class MainWindow(QtGui.QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.setGeometry(0, 30, 300, 350)
        self.setWindowTitle("Serial Metrics")
        self.setWindowIcon(QtGui.QIcon('icons/icon.png'))
        self.main_window()

    def main_window(self):
        bigFont = QtGui.QFont("Arial", 8, QtGui.QFont.Bold)
        labelUniversidad = QtGui.QLabel("UNIVERSIDAD DE LAS FUERZAS
ARMADAS ESPE")
        labelUniversidad.setFont(bigFont)
        labelUniversidad.resize(250,20)
        labelUniversidad.move(20,20)
        self.layout().addWidget(labelUniversidad)

        logoEspe = QtGui.QLabel(self)
        logoEspe.resize(141,36)
        logoEspe.move(70,40)
        logoEspe.setPixmap(QtGui.QPixmap('icons/logo1.png'))
        logoEspe.show()

        labelNombre = QtGui.QLabel("MÓDULO DIDÁCTICO PARA PRÁCTICAS
DE ELECTRÓNICA GENERAL")
        labelNombre.setFont(bigFont)
        labelNombre.setAlignment(QtCore.Qt.AlignCenter)
        labelNombre.resize(250, 40)
        labelNombre.move(20, 100)
        self.layout().addWidget(labelNombre)

        btnFunciones = QtGui.QPushButton("Funciones", self)
        btnFunciones.clicked.connect(self.openFunciones)
        btnFunciones.resize(100,50)
        btnFunciones.move(100,150)

        btnMultimetro = QtGui.QPushButton("Multimetro", self)
        btnMultimetro.clicked.connect(self.openMultimetro)
        btnMultimetro.resize(100, 50)
        btnMultimetro.move(100, 200)

        btnOsciloscopio = QtGui.QPushButton("Osciloscopio", self)
```

```
        btnOsciloscopio.clicked.connect(self.openOsciloscopio)
        btnOsciloscopio.resize(100, 50)
        btnOsciloscopio.move(100, 250)

        self.show()


    def openFunciones(self):
        Funciones(self)

    def openMultimetro(self):
        Multimetro(self)

    def openOsciloscopio(self):
        os.system("C:\scope\scope.exe") // Llama al Graficador
        print "Osciloscopio"

class Funciones(QtGui.QMainWindow):
    serialPort = serial.Serial()
    funcionSeleccionada = ""
    threadFunciones = threading.Thread()

    def __init__(self, parent=None):
        super(Funciones, self).__init__(parent)
        self.setGeometry(320, 30, 500, 400)
        self.setWindowTitle("Serial Metrics - Funciones")
        self.setWindowIcon(QtGui.QIcon('icons/icon.png'))
        self.funciones()

    def funciones(self):
        bigFont = QtGui.QFont("Arial", 16, QtGui.QFont.Bold)
        buttonsFont = QtGui.QFont("Arial", 14, QtGui.QFont.Bold)
        serialLabelFont = QtGui.QFont("Arial", 12, QtGui.QFont.Normal)
        smallFont = QtGui.QFont("Arial", 8, QtGui.QFont.Normal)

        comLabel = QtGui.QLabel("Puerto:")
        comLabel.setFont(serialLabelFont)
        comLabel.resize(50, 20)
        comLabel.move(60, 10)

        comCombo = QtGui.QComboBox()
        comCombo.addItem("")
```

```
        comCombo.addItems(self.serial_ports())
        comCombo.resize(60, 20)
        comCombo.move(120, 10)

        velocidadLabel = QtGui.QLabel("Velocidad:")
        velocidadLabel.setFont(serialLabelFont)
        velocidadLabel.resize(80, 20)
        velocidadLabel.move(190, 10)

        velocidadCombo = QtGui.QComboBox()
        velocidadCombo.addItem("")
        velocidadCombo.addItems(["9600", "115200"])
        velocidadCombo.resize(60, 20)
        velocidadCombo.move(280, 10)

        btnConectar = QtGui.QPushButton("Conectar", self)
        btnConectar.clicked.connect(
            lambda: self.conectarSerial(str(comCombo.currentText()),
str(velocidadCombo.currentText())))
        btnConectar.resize(60, 20)
        btnConectar.move(360, 10)

        rbSeno = QtGui.QRadioButton("Seno")
        rbSeno.setFont(bigFont)
        rbSeno.resize(80,40)
        rbSeno.move(50,50)
        rbSeno.toggled.connect(lambda: self.seleccionarSeno())

        rbCuadrada = QtGui.QRadioButton("Cuadrada")
        rbCuadrada.setFont(bigFont)
        rbCuadrada.resize(120, 40)
        rbCuadrada.move(50, 100)
        rbCuadrada.toggled.connect(lambda: self.seleccionarCuadrada())

        rbPWM = QtGui.QRadioButton("PWM")
        rbPWM.setFont(bigFont)
        rbPWM.resize(120, 40)
        rbPWM.move(50, 150)
        rbPWM.toggled.connect(lambda: self.seleccionarPWM())

        rbTriangular = QtGui.QRadioButton("Triangular")
        rbTriangular.setFont(bigFont)
```

```
rbTriangular.resize(120, 40)
rbTriangular.move(300, 50)
rbTriangular.toggled.connect(lambda: self.seleccionarTriangular())


rbRampa = QtGui.QRadioButton("Rampa")
rbRampa.setFont(bigFont)
rbRampa.resize(120, 40)
rbRampa.move(300, 100)
rbRampa.toggled.connect(lambda: self.seleccionarRampa())


frecuencia1Label = QtGui.QLabel("Frecuencia")
frecuencia1Label.setFont(bigFont)
frecuencia1Label.resize(120, 40)
frecuencia1Label.move(60, 200)

frecuencia1RangoLabel = QtGui.QLabel("(10Hz - 10^6 Hz)")
frecuencia1RangoLabel.setFont(smallFont)
frecuencia1RangoLabel.resize(100, 20)
frecuencia1RangoLabel.move(70, 230)

frecuencia2Label = QtGui.QLabel("Frecuencia")
frecuencia2Label.setFont(bigFont)
frecuencia2Label.resize(120, 40)
frecuencia2Label.move(310, 200)

frecuencia2RangoLabel = QtGui.QLabel("(50Hz - 7000 Hz)")
frecuencia2RangoLabel.setFont(smallFont)
frecuencia2RangoLabel.resize(100, 20)
frecuencia2RangoLabel.move(320, 230)

frecuencia1Value = QtGui.QSpinBox()
frecuencia1Value.setFont(smallFont)
frecuencia1Value.setRange(10, 1000000)
frecuencia1Value.setValue(10)
frecuencia1Value.resize(130,30)
frecuencia1Value.move(50, 250)

frecuencia2Value = QtGui.QSpinBox()
frecuencia2Value.setFont(smallFont)
frecuencia2Value.setRange(50, 7000)
```

```
        frecuencia2Value.setValue(50)
        frecuencia2Value.resize(130, 30)
        frecuencia2Value.move(300, 250)

        btnEnviarFrecuencia1 = QtGui.QPushButton("Enviar", self)
        btnEnviarFrecuencia1.clicked.connect(lambda:
self.enviarFuncionFrecuencia(str(frecuencia1Value.value())))
        btnEnviarFrecuencia1.resize(100, 30)
        btnEnviarFrecuencia1.move(60, 300)
        btnEnviarFrecuencia1.setFont(buttonsFont)

        btnEnviarFrecuencia2 = QtGui.QPushButton("Enviar", self)
        btnEnviarFrecuencia2.clicked.connect(lambda:
self.enviarFuncionFrecuencia(str(frecuencia2Value.value())))
        btnEnviarFrecuencia2.resize(100, 30)
        btnEnviarFrecuencia2.move(310, 300)
        btnEnviarFrecuencia2.setFont(buttonsFont)

        frecuenciaArduino = QtGui.QLabel("")
        frecuenciaArduino.setFont(serialLabelFont)
        frecuenciaArduino.resize(150, 40)
        frecuenciaArduino.setStyleSheet(
            "background-color: rgb(255,255,255); margin:5px; border:2px solid rgb(0, 0,
0); ")
        frecuenciaArduino.setAlignment(QtCore.Qt.AlignHCenter |
QtCore.Qt.AlignVCenter)
        frecuenciaArduino.move(170, 350)

        layout = self.layout()
        layout.addWidget(comLabel)
        layout.addWidget(comCombo)
        layout.addWidget(velocidadLabel)
        layout.addWidget(velocidadCombo)
        layout.addWidget(rbSeno)
        layout.addWidget(rbCuadrada)
        layout.addWidget(rbPWM)
        layout.addWidget(rbTriangular)
        layout.addWidget(rbRampa)
        layout.addWidget(frecuenciaArduino)
        layout.addWidget(frecuencia1Label)
        layout.addWidget(frecuencia1RangoLabel)
        layout.addWidget(frecuencia2Label)
```

```
        layout.addWidget(frecuencia2RangoLabel)
        layout.addWidget(frecuencia1Value)
        layout.addWidget(frecuencia2Value)

        self.threadFunciones = threading.Thread(target=self.leerSerialValue,
args=(frecuenciaArduino,))

        self.show()

    def serial_ports(self):
        """ Lista de puertos
        if sys.platform.startswith('win'):
            ports = ['COM%s' % (i + 1) for i in range(256)]
        elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
            # this excludes your current terminal "/dev/tty"
            ports = glob.glob('/dev/tty[A-Za-z]*')
        elif sys.platform.startswith('darwin'):
            ports = glob.glob('/dev/tty.*')
        else:
            raise EnvironmentError('Unsupported platform')

        result = []
        for port in ports:
            try:
                s = serial.Serial(port)
                s.close()
                result.append(port)
            except (OSError, serial.SerialException):
                pass
        return result

    def conectarSerial(self, com, velocidad):
        if not com or not velocidad:
            print "No selecciono ningun dispositivo a conectar o velocidad a la que
conectarse"
        else:
            print "Conectando al puerto serial con el puerto: " + com + " y velocidad: " +
velocidad
            self.serialPort.close()
            self.serialPort = serial.Serial(com, int(velocidad), timeout=3)
            self.threadFunciones.start()
```

```
    def leerSerialValue(self, frecuenciaArduino):
        while True:
            try:
                if (self.serialPort.inWaiting() > 0):
                    string = self.serialPort.readline()
                    # print(string)
                    arduinoValue = string.split("#")
                    print arduinoValue[1]
                    frecuenciaArduino.setText(arduinoValue[1])
            except:
                frecuenciaArduino.setText("0")

    def closeEvent(self, *args, **kwargs):
        self.serialPort.close()

    def enviarFuncionFrecuencia(self, frecuencia):
        frecuencia = frecuencia.zfill(7)
        if not self.funcionSeleccionada:
            QtGui.QMessageBox.information(self, 'Error Funciones',
                                          'Debe seleccionar al menos una funcion para enviar',
                                          QtGui.QMessageBox.Ok)
        else:
            # print self.funcionSeleccionada + "#" + frecuencia

            try:
                self.serialPort.write((self.funcionSeleccionada + "#" +
frecuencia).encode())
                print "Enviada funcion: " + self.funcionSeleccionada + " a frecuencia: " +
frecuencia

            except:
                QtGui.QMessageBox.information(self, 'Error Funciones',
                                              'Ocurrio un error de comunicacion, asegurese que su
dispositivo esta conectado',
                                              QtGui.QMessageBox.Ok)


    def seleccionarSeno(self):
        self.funcionSeleccionada = "SEN"
    def seleccionarCuadrada(self):
        self.funcionSeleccionada = "CUA"
    def seleccionarPWM(self):
```

```
        self.funcionSeleccionada = "PWM"
    def seleccionarTriangular(self):
        self.funcionSeleccionada = "TRI"
    def seleccionarRampa(self):
        self.funcionSeleccionada = "RAM"
    def leerSerialValue1(self, frecuencia1):
        while True:
            try:
                if (self.serialPort.inWaiting() > 0):
                    string = self.serialPort.readline()
                    # print(string)
                    arduinoValue = string.split("#")
                    print arduinoValue[1]
                    frecuencia1.setText(arduinoValue[1])
            except:
                pass

    def leerSerialValue2(self, frecuencia2):
        while True:
            try:
                if (self.serialPort.inWaiting() > 0):
                    string = self.serialPort.readline()
                    # print(string)
                    arduinoValue = string.split("#")
                    print arduinoValue[1]
                    frecuencia2.setText(arduinoValue[1])
            except:
                pass

class Multimetro(QtGui.QMainWindow):
    serialPort = serial.Serial()
    threadMultimetro = threading.Thread()

    def __init__(self, parent=None):
        super(Multimetro, self).__init__(parent)
        self.setGeometry(320, 370, 500, 300)
        self.setWindowTitle("Serial Metrics - Multimetro")
        self.setWindowIcon(QtGui.QIcon('icons/icon.png'))
        self.multimetro()

    def serial_ports(self):
        """ Lista de puertos serials disponibles
```

```
        """
        if sys.platform.startswith('win'):
            ports = ['COM%s' % (i + 1) for i in range(256)]
        elif sys.platform.startswith('linux') or sys.platform.startswith('cygwin'):
            # this excludes your current terminal "/dev/tty"
            ports = glob.glob('/dev/tty[A-Za-z]*')
        elif sys.platform.startswith('darwin'):
            ports = glob.glob('/dev/tty.*')
        else:
            raise EnvironmentError('Unsupported platform')

        result = []
        for port in ports:
            try:
                s = serial.Serial(port)
                s.close()
                result.append(port)
            except (OSError, serial.SerialException):
                pass
        return result

    def multimetro(self):
        bigFont = QtGui.QFont("Arial", 36, QtGui.QFont.Bold)
        buttonsFont = QtGui.QFont("Arial", 20, QtGui.QFont.Bold)
        serialLabelFont = QtGui.QFont("Arial", 12, QtGui.QFont.Normal)

        comLabel = QtGui.QLabel("Puerto:")
        comLabel.setFont(serialLabelFont)
        comLabel.resize(50, 20)
        comLabel.move(60, 10)

        comCombo = QtGui.QComboBox()
        comCombo.addItem("")
        comCombo.addItems(self.serial_ports())
        comCombo.resize(60, 20)
        comCombo.move(120, 10)

        velocidadLabel = QtGui.QLabel("Velocidad:")
        velocidadLabel.setFont(serialLabelFont)
        velocidadLabel.resize(80, 20)
        velocidadLabel.move(190, 10)
```

```
        velocidadCombo = QtGui.QComboBox()
        velocidadCombo.addItem("")
        velocidadCombo.addItems(["9600", "115200"])
        velocidadCombo.resize(60, 20)
        velocidadCombo.move(280, 10)

        btnConectar = QtGui.QPushButton("Conectar", self)
        btnConectar.clicked.connect(lambda:
self.conectarSerial(str(comCombo.currentText()),
str(velocidadCombo.currentText())))
        btnConectar.resize(60,20)
        btnConectar.move(360, 10)

        multimetroLabel = QtGui.QLabel("")
        multimetroLabel.setFont(bigFont)
        multimetroLabel.resize(400, 70)
        multimetroLabel.setStyleSheet("background-color: rgb(255,255,255);
margin:5px; border:2px solid rgb(0, 0, 0); ")
        multimetroLabel.setAlignment(QtCore.Qt.AlignHCenter |
QtCore.Qt.AlignVCenter)
        multimetroLabel.move(50, 60)

        btnCA = QtGui.QPushButton("CA", self)
        btnCA.clicked.connect(lambda: self.medirCorrienteAlterna(multimetroLabel))
        btnCA.resize(80, 40)
        btnCA.move(160, 140)
        btnCA.setFont(buttonsFont)

        btnCD = QtGui.QPushButton("CD", self)
        btnCD.clicked.connect(lambda: self.medirCorrienteDirecta(multimetroLabel))
        btnCD.resize(80, 40)
        btnCD.move(260, 140)
        btnCD.setFont(buttonsFont)

        btnVA = QtGui.QPushButton("VA", self)
        btnVA.clicked.connect(lambda: self.medirVoltajeAlterno(multimetroLabel))
        btnVA.resize(80, 40)
        btnVA.move(160, 190)
        btnVA.setFont(buttonsFont)

        btnVD = QtGui.QPushButton("VD", self)
        btnVD.clicked.connect(lambda: self.medirVoltajeDirecto(multimetroLabel))
```

```
        btnVD.resize(80, 40)
        btnVD.move(260, 190)
        btnVD.setFont(buttonsFont)

        btnR = QtGui.QPushButton("R", self)
        btnR.clicked.connect(lambda: self.medirResistencia(multimetroLabel))
        btnR.resize(80, 40)
        btnR.move(210, 240)
        btnR.setFont(buttonsFont)

        layout = self.layout()
        layout.addWidget(comLabel)
        layout.addWidget(comCombo)
        layout.addWidget(velocidadLabel)
        layout.addWidget(velocidadCombo)
        layout.addWidget(multimetroLabel)

        self.threadMultimetro = threading.Thread(target=self.leerSerialValue,
args=(multimetroLabel,))


        self.show()

    def closeEvent(self, *args, **kwargs):
        self.serialPort.close()

    def medirCorrienteAlterna(self, multimetroLabel):
        opcion = QtGui.QMessageBox.question(self, 'Corriente Alterna',
                                            "Seguro que quiere medir Corriente Alterna?",
                                            QtGui.QMessageBox.Yes | QtGui.QMessageBox.No
                                            )
        if opcion == QtGui.QMessageBox.Yes:
            try:
                self.serialPort.write("CA#".encode())
                print "Midiendo Corriente Alterna"

                multimetroLabel.setText("CA")
            except:
                QtGui.QMessageBox.information(self, 'Error Corriente Alterna',
                                              'Ocurrio un error de comunicacion, asegurese que su
dispositivo esta conectado',
                                              QtGui.QMessageBox.Ok)
```

```
        else:
            print "Configure su equipo y regrese"
            return

    def medirCorrienteDirecta(self, multimetroLabel):
        opcion = QtGui.QMessageBox.question(self, 'Corriente Directa',
                                            "Seguro que quiere medir Corriente Directa?",
                                            QtGui.QMessageBox.Yes | QtGui.QMessageBox.No
                                            )
        if opcion == QtGui.QMessageBox.Yes:
            try:
                self.serialPort.write("CD#".encode())
                print "Midiendo Corriente Directa"

                multimetroLabel.setText("CD")

            except:
                QtGui.QMessageBox.information(self, 'Error Corriente Directa',
                                              'Ocurrio un error de comunicacion, asegurese que su
dispositivo esta conectado',
                                              QtGui.QMessageBox.Ok)

        else:
            print "Configure su equipo y regrese"
            return

    def medirVoltajeAlterno(self, multimetroLabel):
        opcion = QtGui.QMessageBox.question(self, 'Voltaje Alterno',
                                            "Seguro que quiere medir Voltaje Alterno?",
                                            QtGui.QMessageBox.Yes | QtGui.QMessageBox.No
                                            )
        if opcion == QtGui.QMessageBox.Yes:
            try:
                self.serialPort.write("VA#".encode())
                print "Midiendo Voltaje Alterno"
                multimetroLabel.setText("VA")
            except:
                QtGui.QMessageBox.information(self, 'Error Voltaje Alterno',
                                              'Ocurrio un error de comunicacion, asegurese que su
dispositivo esta conectado',
                                              QtGui.QMessageBox.Ok)
```

```
        else:
            print "Configure su equipo y regrese"
            return

    def medirVoltajeDirecto(self, multimetroLabel):
        opcion = QtGui.QMessageBox.question(self, 'Voltaje Directo',
                                            "Seguro que quiere medir Voltaje Directo?",
                                            QtGui.QMessageBox.Yes | QtGui.QMessageBox.No
                                            )
        if opcion == QtGui.QMessageBox.Yes:
            try:
                self.serialPort.write("VD#".encode())
                print "Midiendo Voltaje Directo"
                multimetroLabel.setText("VD")


            except:
                QtGui.QMessageBox.information(self, 'Error Voltaje Directo',
                                              'Ocurrio un error de comunicacion, asegurese que su
dispositivo esta conectado',
                                              QtGui.QMessageBox.Ok)

        else:
            print "Configure su equipo y regrese"
            return

    def medirResistencia(self, multimetroLabel):
        opcion = QtGui.QMessageBox.question(self, 'Resistencia',
                                            "Seguro que quiere medir Resistencia?",
                                            QtGui.QMessageBox.Yes | QtGui.QMessageBox.No
                                            )
        if opcion == QtGui.QMessageBox.Yes:
            try:
                self.serialPort.write("R#".encode())

                print "Midiendo Resistencia"

            except:
                QtGui.QMessageBox.information(self, 'Error Resistencia',
                                              'Ocurrio un error de comunicacion, asegurese que su
dispositivo esta conectado',
```

```
QtGui.QMessageBox.Ok)

        else:
            print "Configure su equipo y regrese"
            return

    def leerSerialValue(self, multimetroLabel):
        while True:
            try:
                if (self.serialPort.inWaiting() > 0):
                    string = self.serialPort.readline()
                    # print(string)
                    arduinoValue = string.split("#")
                    print arduinoValue[1]
                    multimetroLabel.setText(arduinoValue[1])
            except:
                multimetroLabel.setText("00.00")

    def conectarSerial(self, com, velocidad):
        if not com or not velocidad:
            print "No selecciono ningun dispositivo a conectar o velocidad a la que
conectarse"
        else:
            print "Conectando al puerto serial con el puerto: " + com + " y velocidad: " +
velocidad
            self.serialPort.close()
            self.serialPort = serial.Serial(com, int(velocidad), timeout=3)
            self.threadMultimetro.start()
def run():
    app = QtGui.QApplication(sys.argv)
    GUI = MainWindow()
    sys.exit(app.exec_())

run()
```

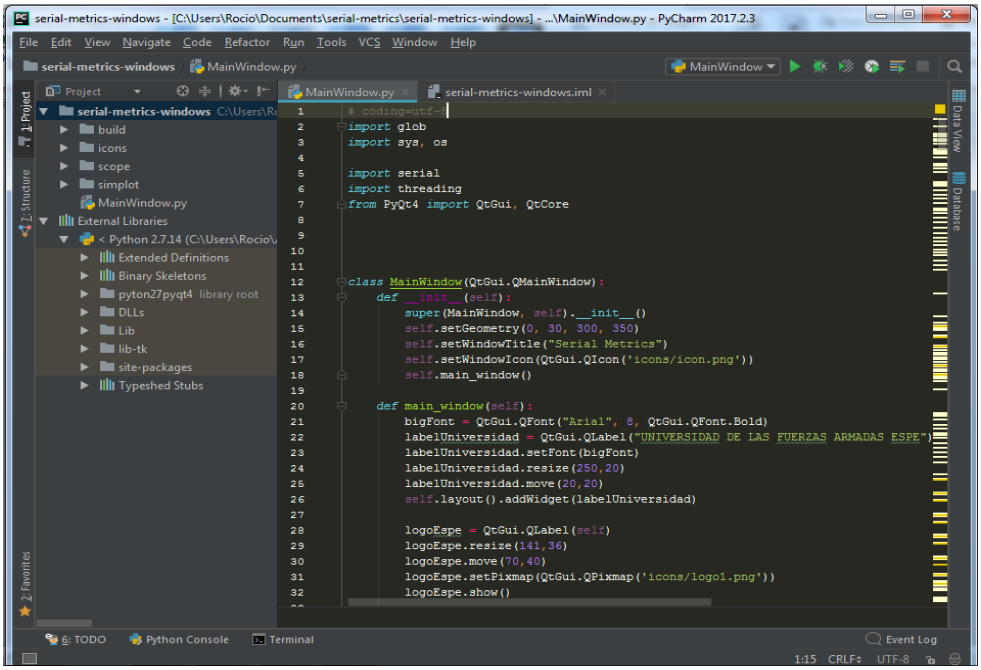

Figura 93. Ejecución del programa en JetBrainsPyCharm

Al dar un click en Run se desplegará la siguiente pantalla como se muestra en la figura 94.

Figura 94. Pantalla de ejecución del programa en PyCharm

Seleccionamos cualquiera de las funciones para iniciar. Por ejemplo al escoger Multímetro se ejecuta la función realizada en el circuito de acondicionamiento de Arduino.

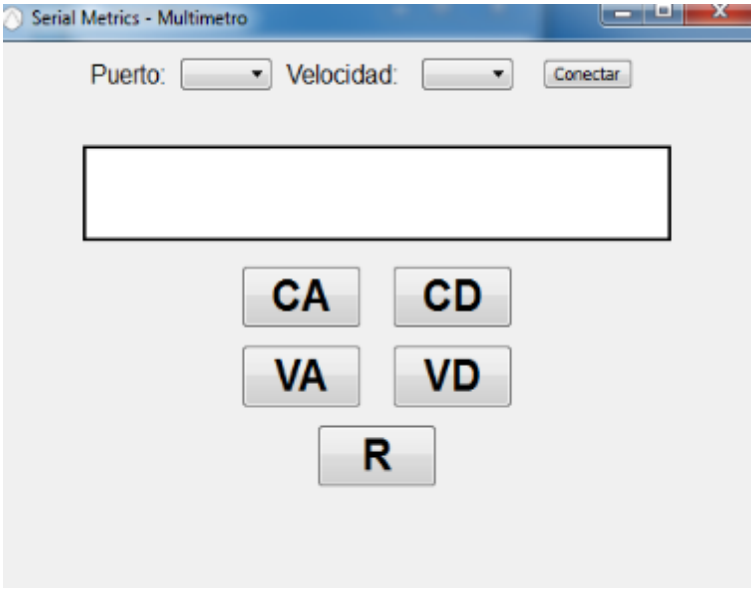

Figura 95. Selección y opciones de la función Multímetro

CAPÍTULO IV

PRUEBAS Y RESULTADOS

4.1. Pruebas y resultados del funcionamiento del módulo didáctico

Una vez finalizada la construcción del módulo didáctico, instalado el software requerido para su funcionamiento, se procedió a verificar el mismo en las instalaciones del Colegio Técnico de Bachillerato Dr. Trajano Naranjo I, se instalan los programas y librerías en cada uno de los computadores del laboratorio de Electrónica de la institución educativa, se conecta a través de cable USB el módulo al PC y se prueba el equipo: osciloscopio, multímetro, generador de funciones, fuentes de voltaje fijas y variables; como se detalla a continuación.

Para demostrar que los resultados obtenidos son confiables, se presentan algunas tablas comparativas de resultados obtenidos mediante la utilización tanto de un equipo de referencia (patrón) como de las funciones del módulo didáctico.

4.2. Pruebas y resultados de las fuentes de voltajes fijas y variables

Para este tipo de pruebas se realizó las medidas con un Multímetro marca Agilent U1232A (figura 96), se realizan mediciones de las fuentes, ±5V, ±12V fijos y 0 a 24V, 0 a -24V variables.

Figura 96. Multímetro Agilent U1232A

4.2.1. Pruebas y resultados de las fuentes de voltaje fijas

Tabla 11.
Resultados de valores de voltajes tomados con equipo de referencia (patrón) de las fuentes del proyecto.

FUENTES FIJAS	VALOR MEDIDO CON PATRÓN	ERROR
+5V	+ 4,98	0.02
+-5V	-4,88V	0.12
+12V	+11,96V	0.04
-12V	-11,77V	0,23

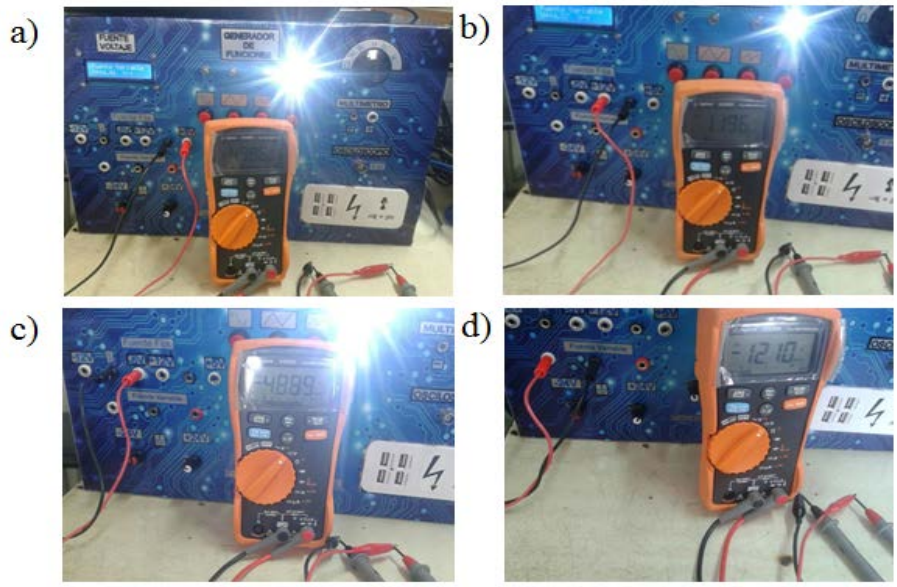

Figura 97. Resultados de las fuentes fijas
a)+5V b)+12V c) -5V d) -12 V

Los valores generados de las fuentes con respecto al valor esperado se encuentran dentro de un margen aceptable de error por ende las fuentes de voltaje fijas del módulo cumplen con los requerimientos señalados.

4.2.2. Pruebas y resultados de la fuente variable de voltaje.

Tabla 12.
Resultados de las mediciones de las fuentes variables de voltaje.

FUENTES VARIABLES	VALOR MEDIDO CON EQUIPO PATRÓN	
+1.2V a +24V	Medida Mínima	+1,44V
	Medida Máxima	+28,2V
-1.2V a -24V	Medida Mínima	-1,22V
	Medida Máxima	-31,22V

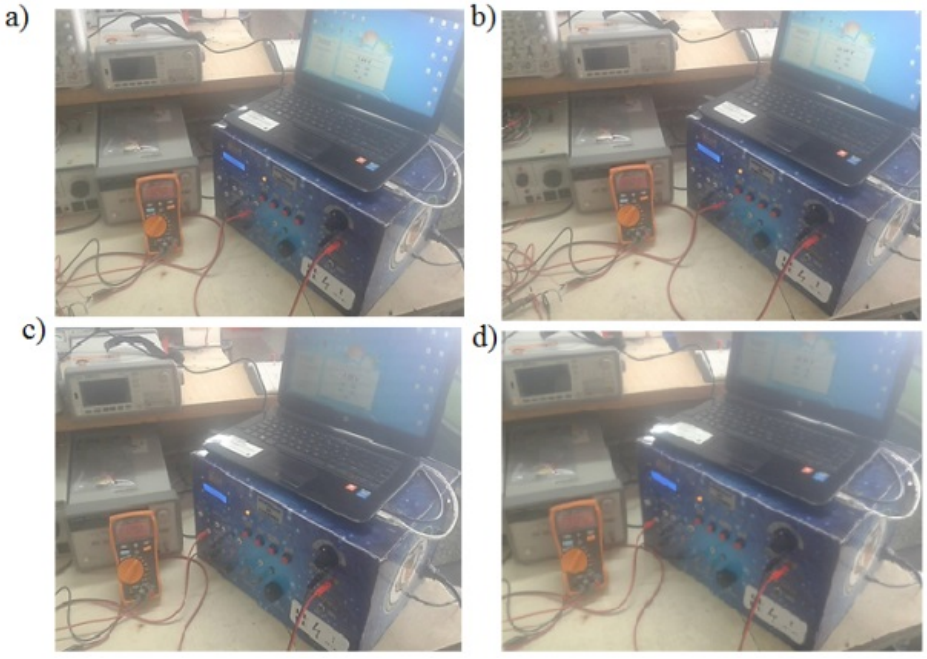

Figura 98. Resultados de las fuentes variables positiva y negativa
a) Valor mínimo medido de la fuente positiva b) Valor máximo medido de la fuente positiva c) Valor mínimo medido de la fuente negativa d) Valor máximo medido de la negativa

Los valores medidos de las fuentes se encuentran dentro de los requerimientos para las aplicaciones prácticas, el valor máximo de las fuentes tanto positiva como negativa sobrepasan el valor especificado en el diseño debido a la tensión de salida superior del transformador.

4.3. Pruebas y resultados del Multímetro de módulo didáctico.

Para la prueba de la función multímetro se utilizó como elemento patrón un multímetro marca Agilent U1232A del Laboratorio de Electrónica Digital de la Universidad de las Fuerzas Armadas ESPE (Figura 96).

4.3.1. Pruebas y resultados del Multímetro - Ohmetro (R)

En siguiente tabla se muestran los datos medidos tanto con el dispositivo patrón así como con el módulo (Figura 99) para esto se ha tomado como referencia cinco resistencias (5% de tolerancia) de valores: 0.51K Ω, 1K Ω, 5,1K Ω, 10K Ω y 20K Ω.

Tabla 13.
Valores medidos de diferentes resistencias.

Valor Resistencia	Valor Medido Patrón	Valor Medido Módulo	Error
0,51K Ω	0,50K Ω	0,49 KΩ	0,01
1KΩ	0,98 KΩ	0,97 KΩ	0,01
5,1 KΩ	5,10KΩ	5,10KΩ	0
10 KΩ	9,93 KΩ	9,99 KΩ	-0,06
20 KΩ	19,58 KΩ	20,14 KΩ	-0,56

Figura 99. Resultados de las mediciones de resistencia a) Medición con el módulo b) Medición con dispositivo patrón

Como se puede observar en la Tabla 13 existe una mínima diferencia entre los valores obtenidos con el equipo construido y el equipo de referencia utilizado.

4.3.2. Pruebas y resultados del Multímetro – Voltímetro en DC

El voltaje para la medición ha sido fijado desde una fuente variable de voltaje continuo Agilent 3631A del laboratorio de la Universidad de las Fuerzas Armadas ESPE. Figura 100.

Figura 100. Fuente regulable de voltaje continuo Agilent 3631A.

Tabla 14.
Valores de voltajes continuos medidos con el módulo (Voltímetro DC).

Valor Voltaje Vdc	Valor medido patrón	Medida Medido Módulo	Error
5 V	5,00V	4,99 V	0,01
10 V	10,00 V	10,07 V	0,07
15 V	15,00 V	17,98 V	0,02
20 V	20,00 V	19,96 V	0,04
25 V	25,00 V	24,93 V	0,07

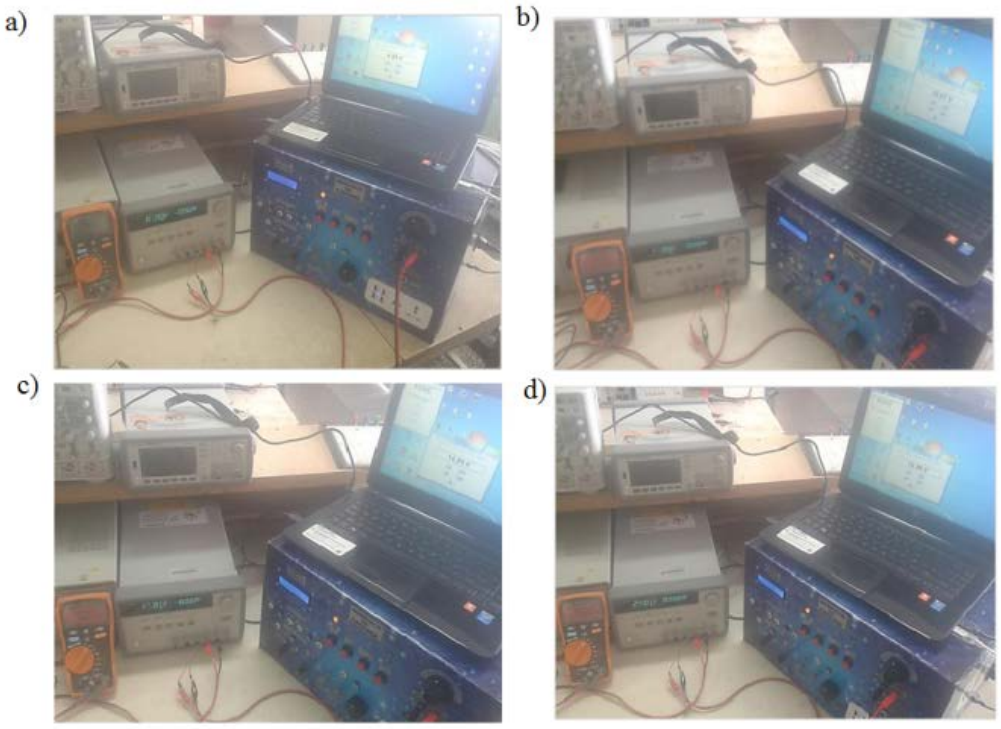

Figura 101. Resultados de las mediciones con el Multimetro - Voltaje DC
a) Medición de 5V b) Medición de 10V c) Medición de 15V d) Medición de 20V

El mayor error calculado es de 0,07 y se puede observar que existe una mínima diferencia entre el resto de valores medidos para los dos instrumentos.

4.3.3. Pruebas y resultados del Multímetro (Voltímetro AC)

Para la prueba del Voltímetro de AC se fijaron valores de una fuente de voltaje alterno variable analógico BK-PRECISION del laboratorio de la Universidad

que tiene un rango de 0 a 150Vac como se observa en la figura 102. Según la tabla 12 los valores medidos con los dos instrumentos son muy similares.

Figura 102. Fuente variable de voltaje alterno del laboratorio de Electrónica Digital de la Universidad de las Fuerzas Armadas ESPE

Tabla 15.
Medición de voltajes alternos con el módulo.

Valor Voltaje Vac	Valor Medido Módulo	Error
50 V	50,11V	0,11
60 V	60,68 V	0,68
70 V	70,34V	0,34
80 V	80,04V	0,04
90 V	90,11V	0,11
100V	100,22V	0,22

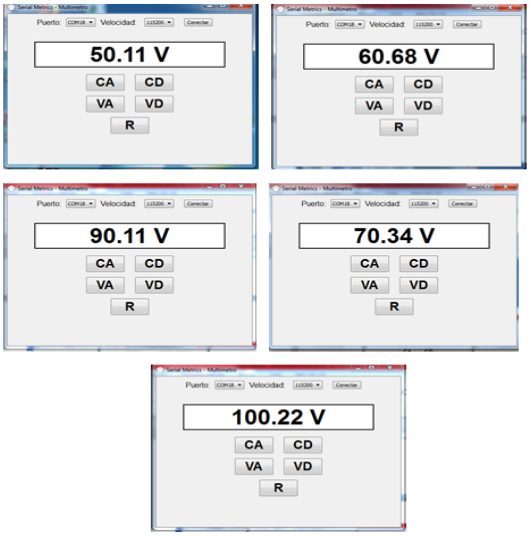

Figura 103. Resultados de las mediciones de Voltaje AC

4.3.4. Pruebas y resultados del Multímetro (Amperímetro)

La figura 104 muestra el circuito que ha sido necesario implementar para la medición de corriente con el amperímetro de laboratorio y el amperímetro del módulo construido.

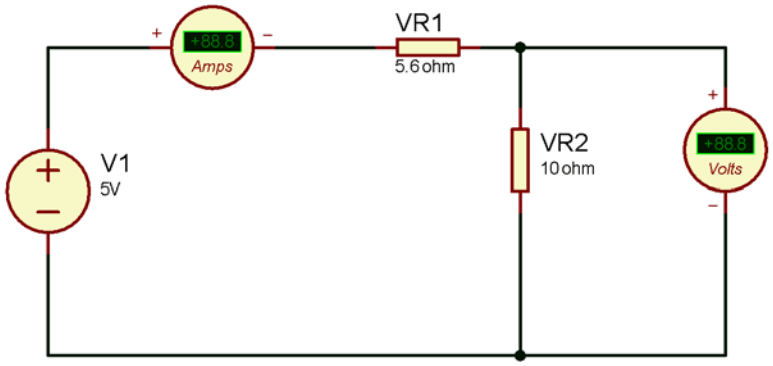

Figura 104. Circuito serie para la medición de la corriente

En el circuito serie la resistencia total es 15,6 Ω, con este valor se calcula la corriente que circula por el circuito. Aplicando la Ley de Ohm (I=V/R) y se obtiene la corriente en R1 que es la misma en R2.

Vin= 5V
R_{TOTAL}=10Ω+5,6Ω
R_{TOTAL}=15,6Ω

I=5V/15,6Ω
I_{R1}=0,32
I_{R2}=0,32

Tabla 16.
Valores de corriente medidos.

	Valor calculado	Valor medido patrón	Valor Medido Módulo	Error
Corriente en R1	0,32 A	0,31 A	0,32 A	0,01
Corriente en R2	0,32 A	0,31 A	0,32 A	0,01

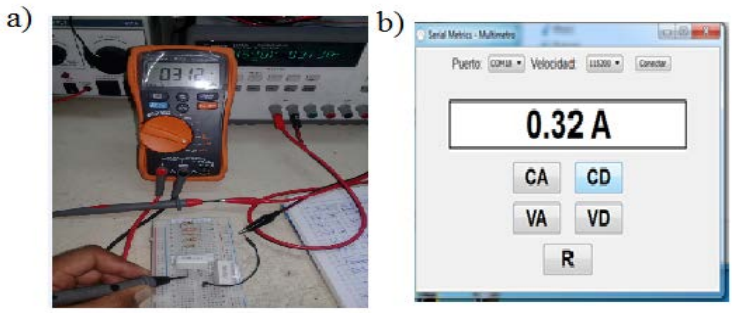

Figura 105. Medición de la corriente
a) Medición con equipo de referencia b) Medición con amperímetro del módulo.

La corriente es uno de los parámetros que se puede medir con éste módulo y se visualiza en la tabla anterior, su error es mínimo entre los dos equipos de medición.

4.4. Pruebas del generador de funciones.

Para este tipo de prueba se utilizó como elemento patrón un generador de funciones marca Agilent Technologies del laboratorio de Electrónica Digital de la Universidad de las Fuerzas Armadas ESPE, debido a que este equipo cuenta con las certificaciones respectivas. Comparando la forma de onda y los parámetros de la señal, ésta coincide con los obtenidos por el generador construido en el módulo. (Figura 106b)

Para la visualización de las señales se utilizó un osciloscopio de marca Agilent Technologies modelo MSO-X-2014A del mismo laboratorio (Figura 106a).

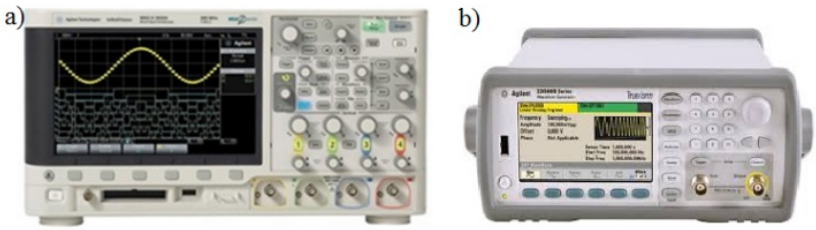

Figura 106. a) Osciloscopio b) Generador de Funciones de Laboratorio de Electrónica Digital de la Universidad de las Fuerzas Armadas ESPE.

En la pantalla del usuario se muestra y se puede seleccionar las señales (senoidal, cuadrada, PWM, triangular y rampa), se fijan los parámetros de frecuencia y amplitud los cuales fueron comprobados al medir con el osciloscopio físico. Figura 107.

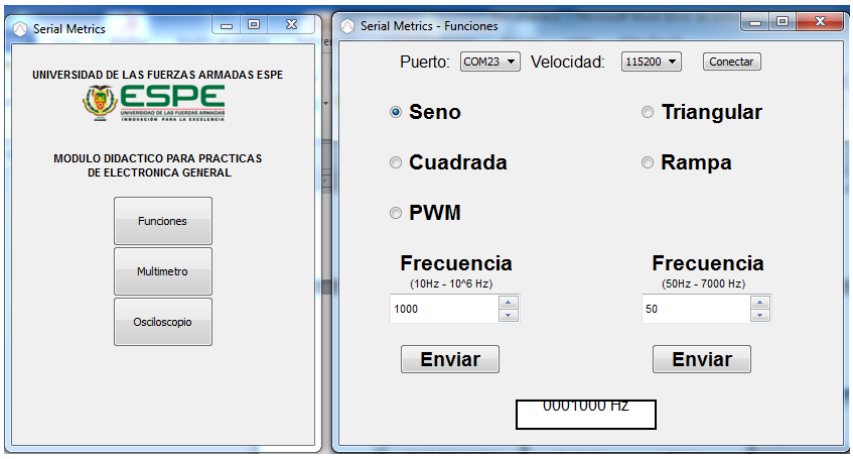

Figura 107. Interfaz general del módulo para generador de funciones.

A continuación se muestran las formas de onda generadas desde el generador del módulo didáctico, para esto se ha fijado una frecuencia de 1Khz y una amplitud de 4Vp-p las mismas que son mostradas por medio del osciloscopio existente en el laboratorio de Electrónica Digital de la Universidad de las Fuerzas Armadas ESPE.

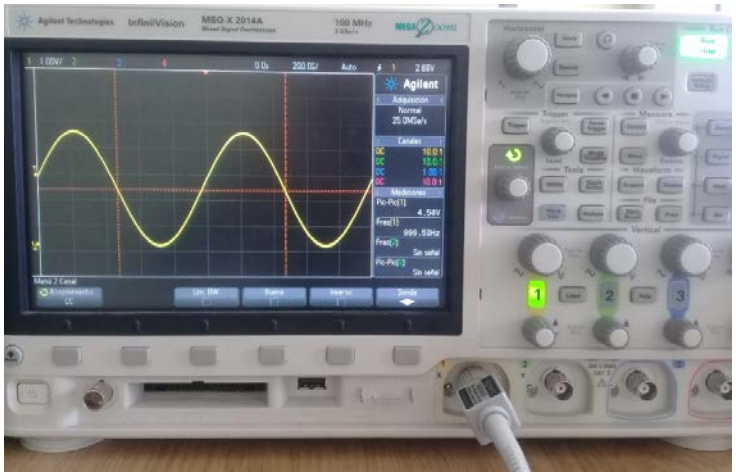

Figura 108. Generador de ondas - Señal Senoidal

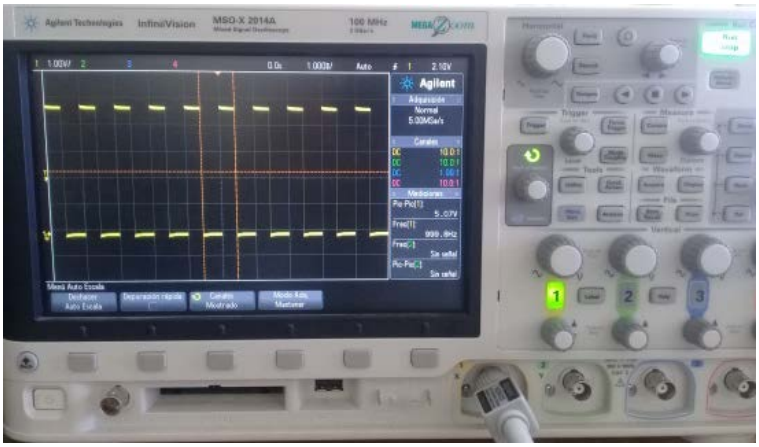

Figura 109. Generador de ondas - Señal Cuadrada

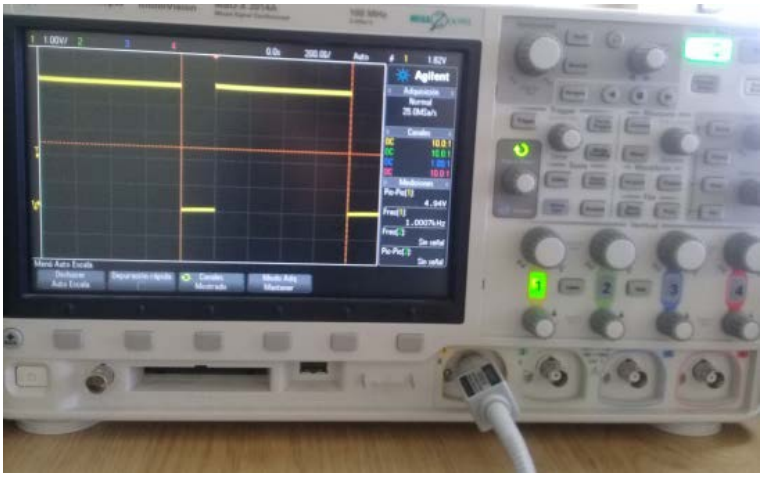

Figura 110. Generador de ondas - Señal PWM

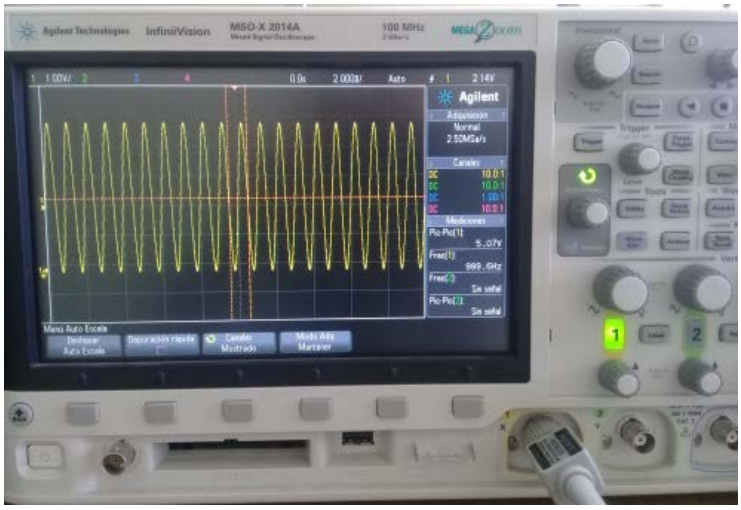

Figura 111. Generador de ondas - Señal Triangular

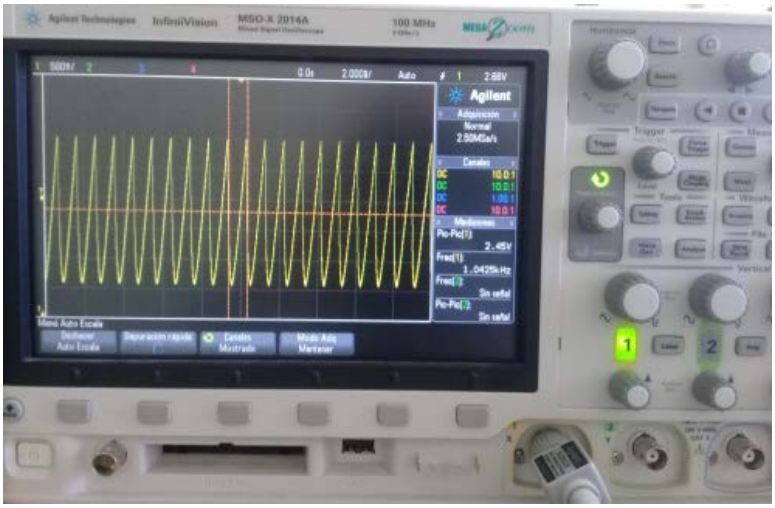

Figura 112. Generador de ondas - Señal Rampa

4.5. Pruebas del Osciloscopio Virtal (Módulo Didáctico).

Para las pruebas del osciloscopio se utilizan las señales obtenidas en el generador del numeral anterior. Se ha comprobado entonces que la forma de onda mostrada tanto por un osciloscopio físico y las del osciloscopio virtual son similares en cuanto a su forma y parámetros eléctricos como frecuencia, período y amplitud

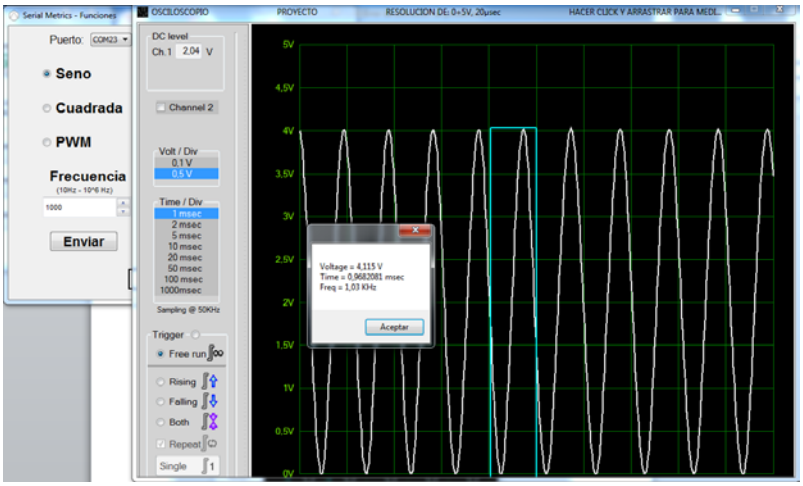

Figura 113. Osciloscopio Virtual – Señal Senoidal

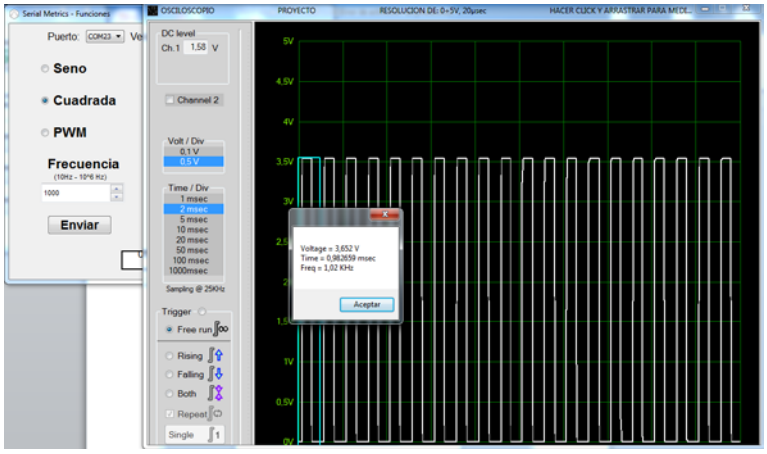

Figura 114. Osciloscopio Virtual – Señal Cuadrada

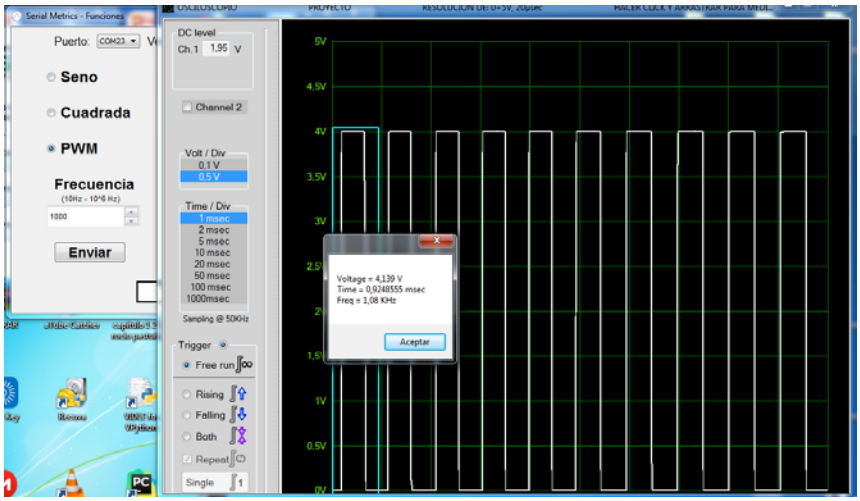

Figura 115. Osciloscopio Virtual – Señal PWM

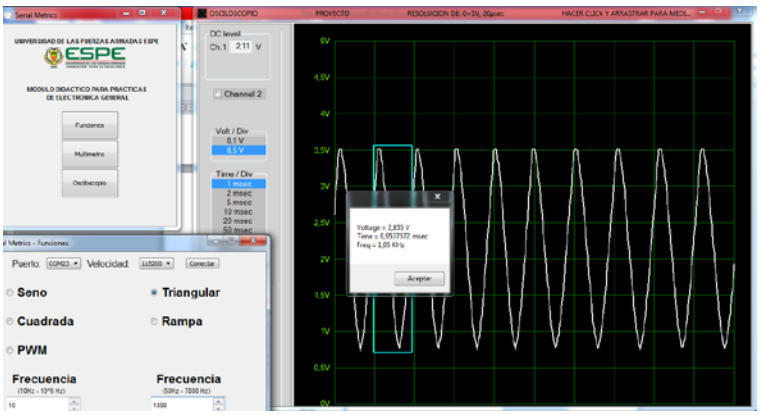

Figura 116. Osciloscopio Virtual – Señal Triangular

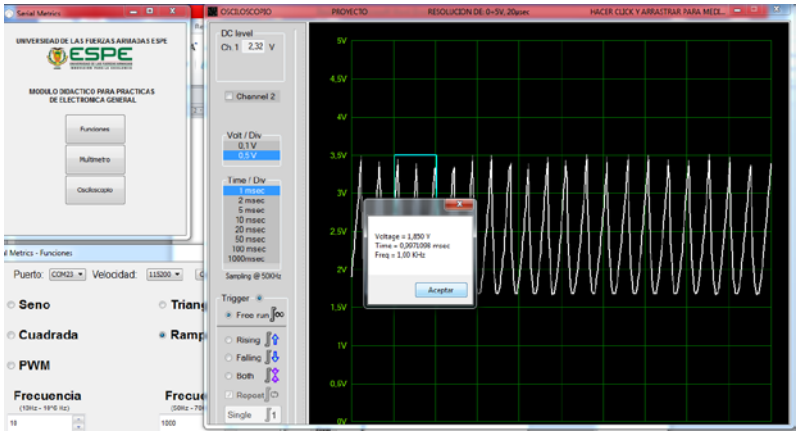

Figura 117. Osciloscopio Virtual – Señal Rampa

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Se diseñó e implementó un módulo didáctico de bajo costo, basado en instrumentación virtual para la realización de prácticas de electrónica general en el Colegio Técnico de Bachillerato “Dr. Trajano Naranjo Iturralde”.
- Se diseñó e implementó una fuente variable de corriente continuo de -24v a 0v, 0v a 24V a 1 amperio, que satisfacen la demanda requerida, con las respectivas protecciones tanto para sobrecargas y corrientes de cortocircuito.
- Se diseñó e implementó un generador de funciones para onda sinusoidal, cuadrada y pwm con una frecuencia de trabajo desde 10 Hz a 1MHz asi como la triangular y diente de sierra con rango de frecuencia de 70Hz a 5KHz y amplitud máxima de 5 voltios pico pico.
- Se realizó la adquisición de datos mediante la programación desarrollada en el programa PyCharm, con la ayuda de la librería Pyserial, la que nos permite la comunicación serial RS232 entre la tarjeta Arduino y la PC.
- Mediante la utilización de PyCharm se desarrolló la interfaz gráfica para el módulo didáctico, que permite simular instrumentos virtuales tales como: voltímetros de cd y ca cuyas escalas se encuentran seteadas de 0 a 120Vca y de 0 a ±30Vcd respectivamente, óhmetro, amperímetro y osciloscopio.
- Se generó una pantalla inicial en la cual al seleccionar una de las variables eléctricas a medir, ésta presenta automáticamente su valor medido.
- Python no posee graficador propio por lo tanto se usan opciones como importar programas que se enlazan a través de subrutinas para la visualización de las ondas.
- La estructura consta de un banco de trabajo de 40x30x20 cm, fabricado en acero inoxidable, con una sub base de acrílico para garantizar el correcto aislamiento de los elementos y placas electrónicas, en la parte frontal se encuentran todos los

elementos de control y visualización, en la parte interna se encuentra el cableado entre los elementos de control y las tarjetas electrónicas, lo que permite una fácil manipulación en futuros mantenimientos.

- En las pruebas que se realizaron con los diferentes dispositivos de medida del módulo didáctico, se contrastaron con equipos de medida del Laboratorio de Electrónica Digital de la Universidad de las Fuerzas Armadas ESPE Extensión Latacunga, existiendo pequeños errores que están dentro del margen aceptable.
- Las guias desarrolladas para la realización de prácticas en Electrónica General Básicas abarcan temas desde transistores y amplificadores operacionales, etc.

5.2. Recomendaciones

Dentro del proceso de diseño e implementación de un Módulo Didáctico de bajo costo, basado en instrumentación virtual para la realización de prácticas de electrónica general en el Colegio Técnico de Bachillerato Dr. Trajano Naranjo I. se recomienda:

- El módulo didáctico requiere que sea conectado a la alimentación de la red (110V, 60Hz), además se requiere de un cable USB para la conexión entre el mismo y la PC.
- A la hora de utilizar el módulo realizar bien las conexiones al panel frontal, así como alimentar adecuadamente el módulo didáctico (Alimentación de red 110Vac, cable USB), para asegurar un correcto funcionamiento de los circuitos implementados.
- Para la correcta ejecución de las prácticas se debe seguir los paso indicados en las guías de laboratorio.
- Tomar en cuenta las recomendaciones dadas en el manual de usuario, antes de manipular el módulo didáctico, ya que esto permitirá un correcto funcionamiento, alargando así la vida útil del equipo construido.
- El uso de las tarjetas Arduino no genéricas son mas robustas ya que aumenta la fiabilidad en el proceso, pero conlleva un gasto mayor, por esta razón se

recomienda aislar la parte de control con la etapa de potencia como medio de seguridad y protección.

- Como trabajo futuro se recomienda implementar un sistema de almacenamiento que permita generar un archivo que almacene la información obtenida durante la práctica de laboratorio, a fin de realizar un análisis de resultados y una toma de decisión por parte del instructor.
- Para implementaciones futuras, se sugiere implementar sistemas de laboratorio en la parte electrónica en realidad virtual.

REFERENCIAS BIBLIOGRÁFICAS

Acha E., A. V.-L. (2002). *"Power Electronic Control in Electrical Systems"*. Newnes Power Engineering Series.

Arduino Mega 2560. (20 de 12 de 2013). Recuperado el 20 de agosto del 2017. Obtenido de : http://arduino.cc/en/uploads/Main/ArduinoMega2560_R3:Front.jgp

Arduino Uno R3. (10 de mayo de 2017). Recuperado el 20 de agosto del 2017.Obtenido de http://www.forefront.io/attachments/uno.jpg

Arduino, u. (11 de 02 de 2017). *Caracteristicas de arduino uno*. Recuperado el 28 de agosto del 2017. Obtenido de http://www3.gobiernodecanarias.org/medusa/ecoblog/ralvgon/files/2013/05/Caracter%C3%ADsticas-Arduino.pdf

ATmega328. (10 de mayo de 2017). Recuperado el 28 de agosto del 2017. Obtenido de >A< A< http://cdn.instructables.com/FR9/7029/H4B2N7DQ/FR97029H4B2N7DQ.LARGE.jpg

Banzi, M. (2009). *Getting Started with Arduino (1ª edición)*. . Make Books. p. 128. ISBN 0596155514.

Berenguer, X. (2012). *Arte en la Electrónica, Perspectivas de una nueva estética.* Langelot.

Biel Solé, D., Olivé Duran, J., Prat Tasias, J., & Sánchez Robert, F. J. (2009). *"Instrumentación Virtual. Adquisición, procesado y análisis de señales"*. Barcelona: UPC.

Circuito de la fuente de alimentacion. (10 de mayo de 2017). Recuperado el 28 de agosto del 2017. Obtenido de http://www.comunidadelectronicos.com/proyectos/fuente4.gif

Circuito integrado LM317. (08 de mayo de 2017). Recuperado el 29 de agosto del 2017. Obtenido de http://www.electronicaembajadores.com/Productos/Detalle/31/SMCILM317T/circuito-integrado-lm317t---ts317cz-c0

Circuito integrado LM337. (8 de marzo de 2017). Recuperado el 06 de Abril del 2017. Obtenido de https://www.electronicaembajadores.com/datos/pdf1/sm/smci/lm337.pdf

Comunicacion serial. (16 de febrero de 2017). Recuperado el 25 de Junio del 2017. Obtenido de https://pypi.python.org/pypi/pyserial

Diagrama de conexión Regulador 78xx. (08 de mayo de 2017). Recuperado el 28 de Mayo del 2017. Obtenido de https://images.search.yahoo.com/images/view;_ylt=AwrB8pxuoEZZ00QAS042nIlQ;_ylu=X3oDMTIzMnNibzIwBHNlYwNzcgRzbGsDaW1nBG9pZANlYWIxNjlkMDY1NTc5MTM3ZjU2MDEyODA3NTEwOTQyNgRncG9zAzIyBGl0A2Jpbmc-?.origin=&back=https%3A%2F%2Fimages.search.yahoo.com%2Fyhs%2Fsearch%

Diseño de la fuente de voltaje. (10 de mayo de 2017). Recuperado el 28 de Junio del 2017. Obtenido de http://www.comunidadelectronicos.com/proyectos/fuente4.htm

Display 2x16. (08 de julio de 2015). Recuperado el 28 de Enero del 2017. Obtenido de http://ep.yimg.com/ca/I/yhst-27389313707334_2252_105677010

Display LCD de 2x16. (10 de mayo de 2017). Recuperado el 28 de agosto del 2017. Obtenido de https://es.scribd.com/doc/44252680/LCD-16X2

Distribucion de los pines pantalla LCD 2x16. (10 de mayo de 2017). Recuperado el 28 de agosto del 2017 Obtenido de http://www.8051projects.net/lcd-interfacing/lcd.png

Dorf, R., & J.A., S. (2006). *"Introduction to Electric Circuits (7th edition)"*. Wiley. Pág. 74.

Enciclopedia Electrónica, s. (10 de 02 de 2015). Recuperado el 10 de Marzo del 2017. *Enciclopedia Electrónica*. Obtenido de http://www.wikipedia.org/ (Enciclopedia electrónica)

Festo. (2015). Obtenido de http://www.festo.com/cat/es-mx_mx/products_010000

Hubert, C. I. (2006). *"Circuitos eléctricos CA/CC . Enfoque integrado"*. McGraw-Hill., Biblioteca de la E.U.P. Pág 53.

Imagen circuito integrado LM317. (08 de mayo de 2017). Recuperado el 15 de agosto del 2017 Obtenido de https://images.search.yahoo.com/yhs/search?p=Circuito+integrado+LM337&fr=yhs-blp-default&hspart=blp&hsimp=yhs-default&imgurl=http%3A%2F%2Fblog.novaeletronica.com.br%2Fimg%2FCircuito-Testador-LM317.png#id=1&iurl=http%3A%2F%2Fblog.novaeletronica.com.br%2Fim

Imagen Circuito integrado LM337. (08 de mayo de 2017). Recuperado el 30 de agosto del 2017. Obtenido de https://www.electronicaembajadores.com/datos/pdf1/sm/smci/lm337.pdf

Imagen Circuito integrado LM337. (08 de mayo de 2017). Recuperado el 10 de agosto del 2017. Obtenido de http://www.electronicoscaldas.com/datasheet/KA337-LM337_Fairchild.pdf

Imagen de potenciometro. (10 de mayo de 2017). Recuperado el 28 de Septiembre del 2017. Obtenido de http://img.directindustry.es/images_di/photo-g/potenciometro-nivelador-cuadro-11910-4262261.jpg

Imagen Regulador de voltaje 7805. (08 de mayo de 2017). Recuperado el 03 de agosto del 2017. Obtenido de http://www.ecured.cu/index.php/Archivo:7805.jpg

Imagen regulador de voltaje negativo 7905. (08 de mayo de 2017). Recuperado el 13 de Mayo del 2017. Recuperado el 04 de Noviembre del 2017 Obtenido de http://www.electronicoscaldas.com/756-thickbox_default/regulador-5v-ka7905.jpg

Instrumentacion virtual. (29 de Enero de 2007). Recuperado el 17 de Abril del 2017. Obtenido de http://www.electronicosonline.com/2007/01/29/Mejorando-aplicaciones-con-la-Instrumentacion-Virtual/

Instrumentacion virtual. (2015). Obtenido de www.ni.com/virtualinstrumentation/esa

Instrumentos eléctricos de medición. (05 de Junio de 2008). Recuperado el 28 de Noviembre del 2017. Obtenido de http://www.mitecnologico.com/Main/InstrumentosElectricosMedicion

Interruptores. (10 de mayo de 2017). Recuperado el 10 de agosto del 2017. Obtenido de https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcTrSUwhxTkP_DamaEi_R375dCzorO9gzkelJUpzwGSfrdczBGjyEw

KUBOTA, S. (2007). *"Ergonomic comparison of liquid crystal display and cathode ray tube display, Display and imaging"*. 181-190.

LCD SaintSmart TFT 7". (10 de mayo de 2017). Recuperado el 11 de septiembre del 2017. Obtenido de https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcSKyTHpu6qqUn0WKzDyKs0_F0QFDQNu68gRENtA_ZncQepF7IDqvw

Máquinas eléctricas, (. (08 de 03 de 2015). *Máquinas eléctricas*. Recuperado el 08 de Julio del 2017. Obtenido de http://alek.pucp.edu.pe/cursos/pregrado/iee215/pag_principal/maquinasel.htm

Martelli, A. (2007). *Python Guía de referencia.* Gorjón Salvador: Primera Edición, Anaya Multimedia. ISBN 978-84-415-2317-3.

memarcos.com. (2004). Recuperado el 19 de Noviembre del 2017. Obtenido de http://www.mcmarcos.com/pdf/2004_visualizacion-modd.pdf

memarcos.com. (2004). Recuperado el 05 de agosto del 2017. Obtenido de http://www.mcmarcos.com/pdf/2004_visualizacion-modd.pdf

MENOZZI, M. L. (2001). *"CRT versus LCD. Effects of refresh rate, display technology and background luminance in visual performance Displays"*. 79-85.

Miñarro, J. R. (2006). *"Módulos Didácticos"*. España: ITE Gobierno de España.

Módulo LCD. (10 de julio de 2015). Recuperado el 13 de agosto del 2017. Obtenido de http://es.scribd.com/doc/44252680/LCD-16X2

Nilson, J. W. (2005). *"Circuitos eléctricos 7ma. Edición"*. Prentice Hall, Pág. 78.

Noble, J. (2009). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and open Framework.* (1ª edición). O'Reilly Media. p. 768. ISBN 0596154143.

Noble, J. (2009). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and open Framework (1ª edición).* . O'Reilly Media. p. 768. ISBN 0596154143.

Oxer, J., & Blemings, H. (2009). *Practical Arduino: Cool Projects for Open Source Hardware (1ª edición).* . Apress. p. 500. ISBN 1430224770.

Pulsadores. (10 de mayo de 2017). Recuperado el 12 de agosto del 2017. Obtenido de https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcQ3elwbj3LgdLIviWjsxrr1yoRBJAQfraZR0SgocfG3r7RLHegqUQ

Rashid, M. H. (2004). *Electrónica de potencia: Circuitos, dispositivos y aplicaciones.* Pearson Educación.

Recursostic.educacion.sensores. (2015). Obtenido de http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena11/4quincena11_contenidos_3a.htm

Regulador de voltage negativo 7905. (08 de mayo de 2017). Recuperado el 05 de Septiembre del 2017. Obtenido de http://www.electronicoscaldas.com/reguladores-referencias-conversores-de-voltaje/323-regulador-5v-ka7905.html

Regulador de voltaje de salida 7805. (08 de mayo de 2017). Recuperado el 28 de agosto del 2017. Obtenido de http://www.ecured.cu/index.php/Circuito_Integrado_lm7805

SainSmart Mega 2560R3. (28 de mayo de 2017). Recuperado el 04 de agosto del 2017. Obtenido de http://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

SainSmart TFT LCD Adjustable SHield for Arduino Mega 2560 R3 1280 A082. (08 de julio de 2015). Recuperado el 13 de Julio del 2017. Obtenido de http://www.saintsmart.com/arduino/arduino-shields/sainsmart-tft-lcd-adjustable-shield-for-arduinomega-2560-r3-1280-a082-plug.html

Sheneidereman, B. (2008). *"Designing the user interface: strategies for effective human-computer interaction. Reading"*. Addison-Wesley.

Software de Instalación de Arduino. (12 de febrero de 2017). Recuperado el 28 de agosto del 2017. Obtenido de https://www.arduino.cc/en/Main/Donate

Software Python. (20 de febrero de 2017). Recuperado el 28 de agosto del 2017. Obtenido de https://www.python.org/downloads/.

Técnica Didáctica. (2015). Obtenido de http://www.tecnica-didactica.com.ar/educacion_tecnica/

VPython. (27 de febrero de 2017). Recuperado el 28 de agosto del 2017. Obtenido de http://sourceforge.net/projects/vpythonwx/files/6.11-release/VPython-Win-32-Py2.7-6.11.exe/download

ANEXOS

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por la señorita **Rocío Ibeth Pastuña Doicela,**

En la ciudad de Latacunga, a los 23 días del mes de febrero del 2018

Ing. José Bucheli
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO