Escuela Politécnica Superior de Jaén

Grado en Ingeniería Electrónica Industrial

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior

Trabajo Fin de Grado

MONITORIZACIÓN DE ANALIZADOR DE LA CALIDAD ELÉCTRICA PARA SMART GRIDS

Alumno: Juan Francisco Vílchez García

Tutor: Antonio Cano Ortega
Francisco José Sánchez Sutil

Dpto: Ingeniería Eléctrica

SANCHEZ SUTIL FRANCISCO JOSE - 26004797P

Firmado digitalmente por SANCHEZ SUTIL FRANCISCO JOSE - 26004797P
Fecha: 2020.09.10 08:58:22 +02'00'

Junio, 2020

Índice

Resumen

El objetivo de este proyecto es la monitorización de los datos medidos con un analizador comercial en una instalación eléctrica mediante el bus RS485, capaz de medir parámetros como la corriente, tensión, potencias … pudiendo gestionar también la energía generada y consumida.

El dispositivo está basado en un controlador de *hardware libre*, como es Arduino, donde el sistema de control se programará empleando el mismo. Utilizaremos una shield RS485 para arduino mega, el cual dará comunicación entre los periféricos.

Además de monitorizar y controlar los parámetros de la red, el prototipo se configurará para que se pueda visualizar en una app móvil con sistema operativo Android. De esta manera, desde cualquier punto con conexión a Internet podremos ver en todo momento los datos de la red.

Palabras clave

- Arduino
- Hardware y software libre
- Analizador de redes de 4 cuadrantes
- Comunicación bus RS485
- Firebase
- Aplicación móvil

Agradecimientos

- En primer lugar, a mi familia por el apoyo incondicional en todos los momentos duros a lo largo de esta trabajosa y bonita etapa. A todos mis amigos y amigas que han mostrado apoyo y ayuda en determinados momentos a lo largo de la carrera.

- A los tutores del proyecto por el compromiso y atención en cualquier momento en los tiempos excepcionales de la realización del mismo.

- A todos los profesores y profesoras de los diferentes departamentos de la Escuela Politécnica Superior de Jaén por la ayuda y atención que nos han puesto a lo largo de la carrera.

1. Introducción

1.1 Motivación del proyecto

El objeto del presente proyecto es el control y la monitorización de datos medidos por un analizador comercial en una instalación eléctrica.

La decisión por la cual elegí este Trabajo de Fin de Grado fue el afán de saber a fondo como monitorizar los datos medidos por un analizador comercial, que en nuestro caso será el CVM-B150, poder ver los datos en cualquier parte sin estar presente en dicha fábrica, la facilidad de controlar los distintos parámetros mediante un celular sin tener muchos conocimientos de informática, conocer los aparatos de medida en una industria…

El control se llevará a cabo a través de arduino trabajando con una interfaz llamada RS485 y un protocolo de comunicación MODBUS diferente al que hemos trabajado a lo largo de la carrera universitaria cursada.

La elección de los componentes del proyecto se hará teniendo en cuenta los diferentes dispositivos prestados por la universidad.

1.2 Alcance

Este proyecto se basa en el desarrollo de la comunicación mediante bus RS485 necesario para poder saber en todo momento el valor de los parámetros significativos de la instalación eléctrica.

Además, será controlado por un celular en el cual se instalará una aplicación con sistema operativo Android, esto lo explicaremos minuciosamente en apartados posteriores.

1.3 Antecedentes

Se trata de la obtención de datos a través del analizador CVM-B150 en modo trifásico, pero debido a la situación provocada por el coronavirus, no todos los datos lo pudimos hacer en su momento.

Cuando empezó a mejorar la situación, los datos lo obtuvimos a través de un motor de jaula de ardilla ya que otros instrumentos no proporcionaban bien los resultados.

1.4 Objetivos

Los objetivos principales de este proyecto son:

- Desarrollo de la comunicación mediante bus RS485.
- Obtención de datos medidos por el analizador.
- Subida de datos a la nube.
- Diseño de app y plataforma web para visualizador de datos.

2. Estado del arte

En este apartado hablaremos sobre las ofertas del mercado sobre dispositivos capaces de realizar comprobaciones en la red eléctrica. Es un punto importante ya que, podemos elegir con determinación el instrumento más adecuado para nuestro proyecto. Hablaremos de 2 modelos de marcas diferentes, uno que utilizamos en la facultad (Fluke 433) y otro que será el que utilizaremos en dicho proyecto (CVM-B150):

- ***Fluke 433/434***

Este instrumento portátil (*Figura 2.1*) ideal para analizar la calidad eléctrica, utilizado en varias prácticas de una asignatura de electricidad en la escuela, ofrece una completa serie de potentes funciones para la comprobación de sistemas de distribución eléctrica. Algunas de estas funciones le permiten obtener una visión general del funcionamiento del sistema eléctrico, mientras que otras le sirven para examinar detalles específicos.

El modelo Fluke 434 incorpora funciones adicionales como interarmónicos, transitorios, utilización de la energía, memoria adicional para almacenar pantallas y datos, el software FlukeView y un cable óptico USB aislado. La instalación de estas funciones es opcional.

Figura 2.1: Fluke 434

Tanto si se encuentra en una planta industrial, en una fábrica o en una instalación de servicios públicos, estos tipos de redes eléctricas y gestión energética se han diseñado para mantener el máximo rendimiento y fiabilidad.

Hay diferentes modos de medida para examinar en detalle como son las tensiones y corrientes de fase, factor de cresta, armónicos, flicker (Parpadeo), fluctuaciones, frecuencia y desequilibrio.

Este analizador presenta los resultados de las medidas de forma totalmente eficaz en cinco pantallas diferentes que describiremos (Figura 2.2):

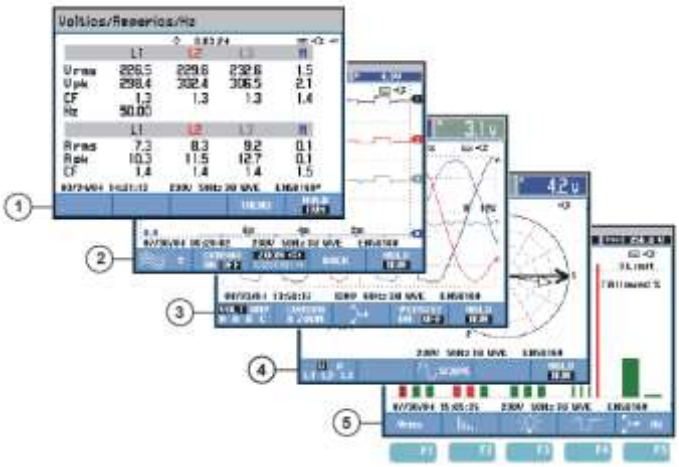


Figura 2.2: Presentación de los tipos de pantalla

1. *Pantalla de tabla*: presenta los datos inmediatos de un gran número de valores numéricos de medidas importantes. Utilizada para: Volt. /Amp. /Hz, fluctuaciones, armónicos, potencia y energía, flicker, desequilibrio y supervisión de la calidad eléctrica.
2. *Pantalla de tendencia*: esta pantalla se relaciona con la de tabla permitiendo mostrar las variaciones a lo largo del tiempo de los valores enumerados en la tabla. Tras seleccionar el modo de medida, el analizador comienza a registrar las lecturas en la tabla.
3. *Pantalla de formas de onda*: muestra las formas de onda de tensión y corriente como si se hiciera con un osciloscopio.
4. *Pantalla de diagrama fasorial*: muestra la relación de fases entre tensiones y corrientes en un diagrama vectorial.
5. *Pantalla de gráfico de barras*: muestra la densidad de cada parámetro como un porcentaje mediante un gráfico de barras. Utilizada para armónicos y supervisión de la calidad eléctrica.

- ***CVM-B150-ITF-HF***

Figura 2.3: CVM-B150

Es un equipo de altas prestaciones que mide, calcula y visualiza los principales parámetros eléctricos en redes monofásicas sobre un panel de 144x144mm, de dos fases con y sin neutro, trifásicas equilibradas, con medida

en ARON o desequilibradas. La medida se realiza en verdadero valor eficaz, mediante cuatro entradas en tensión CA y cuatro entradas de corriente.

Estos analizadores pueden ser instalados en (*Figura 2.4*):

- **Cabecera**: donde se utilizan equipos dedicados a la monitorización, registro y control de la variables eléctricas y energéticas de cualquier instalación. Medir en redes de media tensión o en cabecera de instalaciones de baja tensión. Diseñados, además, para medir otras fuentes de energía a través de pulsos como el agua, gas, …
- **Subcuadros principales**: Equipos dedicados a la lectura de parámetros eléctricos y consumos energéticos en cuadros principales. Equipos dedicados a aportar información relevante sobre el estado general de cada línea, además de actuar sobre cualquier alarma que pueda aparecer.
- **Subcuadros secundarios**: Equipos dedicados a la lectura de cargas situadas a la final de línea o pequeños cuadros de distribución en baja tensión, dando información completa sobre el estado y consumo de cada carga o línea.

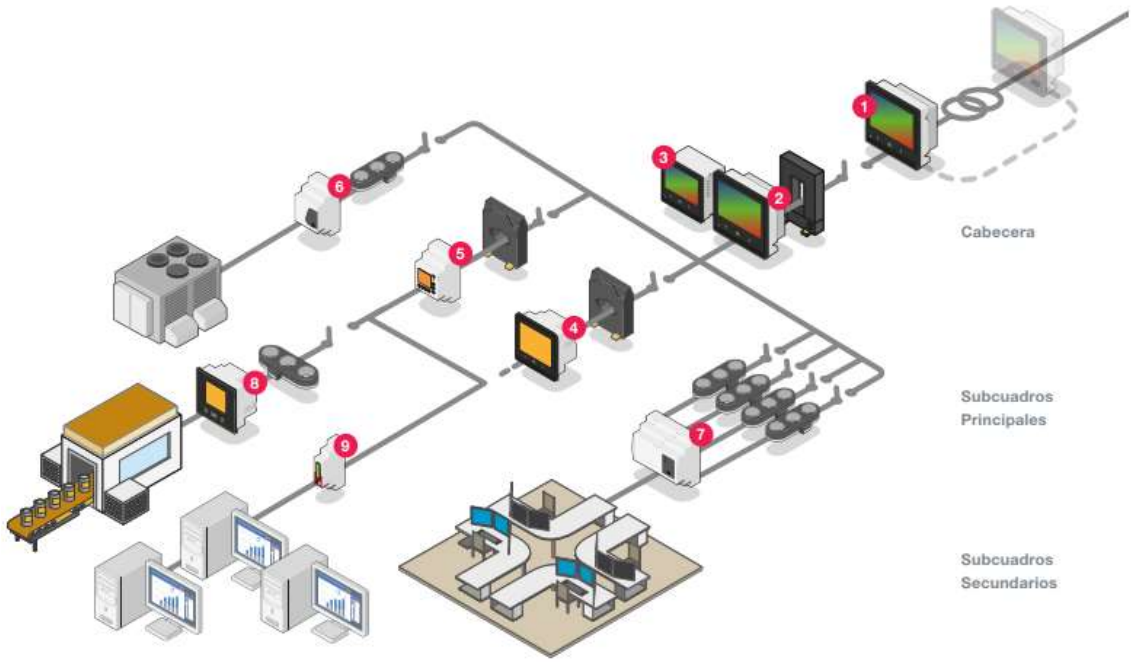


Figura 2.4: Lugares de instalación del CVM-B150

Las características de dicho equipo son:

- Pantalla VGA con gran resolución y color.
- Protección frontal IP65.
- 5 entradas de tensión (3 fases + neutro + Tierra).
- 4 entradas de corriente ITF.
- Precisión en tensión, corriente clase 0.2.
- Precisión en energías clase 0.5S.
- Equipo expandible de hasta 4 módulos, combinando entradas, salidas digitales, analógicas, Mbus/TCP, XML.
- Modular (posibilidad de insertar módulos de expansión).
- Botones de desplazamientos táctiles.
- Fuente de alimentación universal.
- Punto de comunicaciones **RS485** (protocolo **MODUBS/RTU** Y BACnet).
- Personalización de los parámetros a mostrar.

Otras características:

- Innovador interfaz SCV (Slide, Choose & view) de presentación de datos versátil que permite la personalización de los parámetros a mostrar por pantalla.
- Parámetros eléctricos instantáneos, máximo, mínimos, demanda
- Parámetros eléctricos incrementales (energías), horas, costes, emisiones
- 3 Tarifas (seleccionables por entrada digital o por comunicaciones RS485)
- Capaz de mostrar costes y emisiones de kgCO2, por pantalla según la energía consumida o generada.
- 2 salidas de relé para alarmas con retardo, tiempos, ON y OFF, etc.
- 2 salidas a transistor para alarmas o generación de impulsos con todos los posibles parámetros de configuración.
- 2 entradas digitales con posibilidad de control sobre la selección de tarifas del equipo o configurables para monitorización, mediante comunicaciones **RS485 Modbus**, de estados lógicos de otros equipos electro mecánicos. (Interruptores diferenciales, magnetos térmicos …)

Sus aplicaciones son:

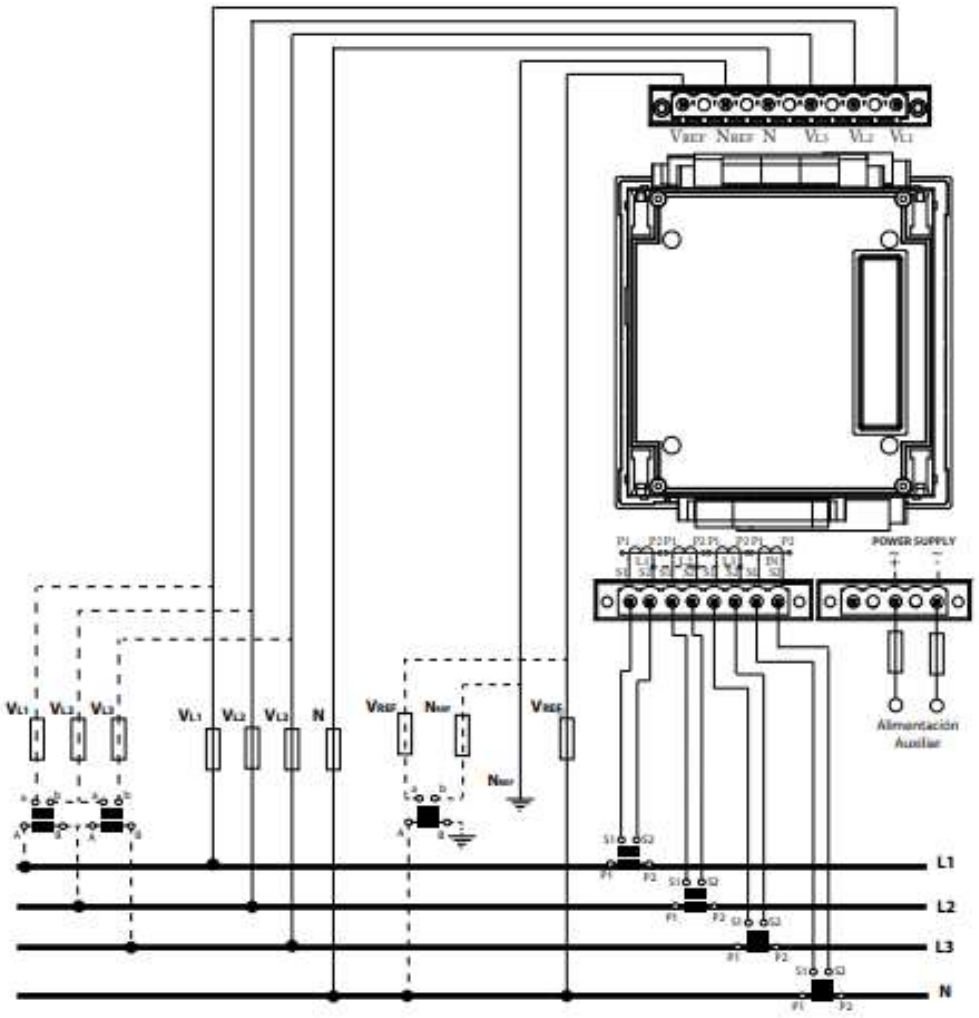
- Control y monitorización de todos los parámetros eléctricos medidos en cuadros eléctricos de distribución y acometidas de baja y alta tensión.
- 4 alarmas (2 por transistor y 2 por relé) totalmente programables de forma independiente según un valor bajo, alto, histéresis, retardos a conexión desconexión, estado de reposo normalmente abierto o cerrado y enclavamiento.
- Generación de impulsos mediante salidas a transistor, totalmente configurables de forma independiente sobre cualquier parámetro incremental (energías, costes, kgCO2, horas tanto por contador total o como por tarifa)
- Convertidor a señales analógicas de cualquier parámetro instantáneo que el equipo mide o calcula, incorporando módulos de expansión con salidas analógicas.
- Visualizador de señales de proceso incorporando módulo de expansión de entradas analógicas, con posibilidad de reportarlas a sistemas SCADA mediante comunicaciones
- Control de maniobras de cargas eléctricas o señales de alarma por programación de las salidas de transistor o relé integradas o añadidas mediante módulos de expansión.
- Datalogger autónomo con servidor WEB mediante conexión a un equipo EDS. Permite la monitorización directa de datos históricos almacenados en la unidad mediante un navegador WEB convencional.

Características técnicas

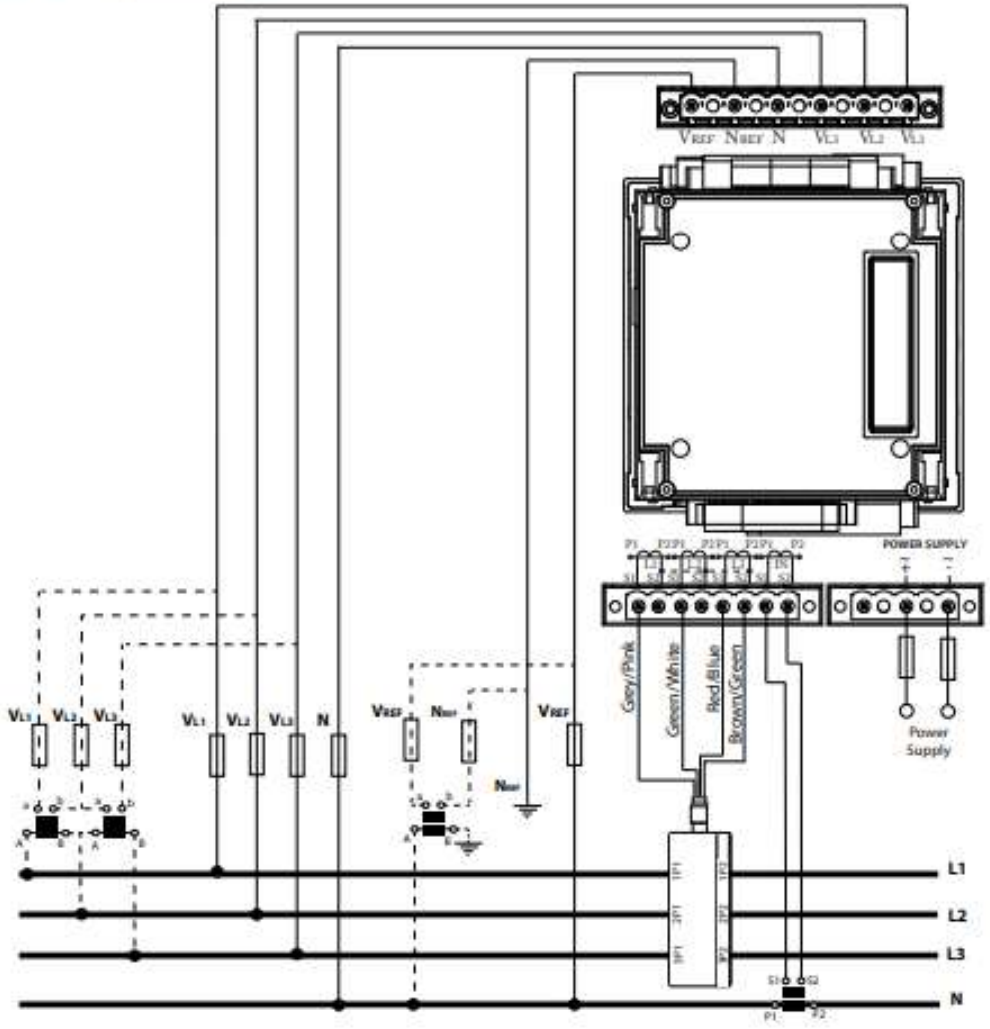
Conexiones		
Entradas digitales	3 (selección de tarifas, estados o alarmas externas)	
	Tipo	Contacto libre de potencial optoaislada
	Corriente de activación	4 mA (12V tensión máxima en contacto abierto)
	Aislamiento	4 kW
Salidas digitales	Generación impulsos o Alarma	
	Tipo	2 transistor NPN
	Salidas digitales a relé	2
	Tensión máxima de maniobra	+/- 400 V a.c.
	Intensidad máxima de maniobra	+/- 130 mA
	Frecuencia máxima	1000 impulsos / segundo
	Duración pulso (T on / T off)	0.3 / 0.7 ms (1 ms de impulso completo)
	Alarmas	
	Tipo	2 relé
	Potencia máxima de maniobra	1500 VA / 180 W
	Tensión máxima de maniobra	400 V
	Intensidad máxima commutación	6A
	Vida eléctrica (400V / 6A)	3 x 104 (85 ºC)
	Vida mecánica	1 x 107
Comunicaciones integradas	RS-485 Modbus o Bacnet RTU A(+) y B(-)	
	Velocidad	9600...115200
	bits, paridad,stop	8,n,1
Condiciones ambientales	Temperatura de uso	-10...+50ºC
	Humedad relativa	5...95%
	Altitud	2000 m
Características constructivas	Formato	Montaje en panel 96x96mm ó 144x144
	Cota profundidad	110mm sin módulos de expansión (ambos modelos)
	Protección para IP frontal	IP65
	Protección IP trasera	IP20
Alimentación universal	Circuito de alimentación: 100...230 V c.a. +/- 15% / 100...260 V c.c. +/-15% Frecuencia de alimentación: 45...65 Hz	
Seguridad	Diseñado para instalaciones CAT III 300/520 V c.a. según EN 61010 Protección frente al choque eléctrico por doble aislamiento clase II	
Normas	IEC 62053-22, ANSI (clase 0.5S), IEC 62053-23 ANSI C12.1 (clase 2), IEC 61010, IEC 61000, UNE-EN 55022Medida según MID, diseño según UL IEC 61000-4-2, IEC 61000-4-3, IEC 61000-4-11, IEC 61000-4-4, IEC 61000-4-5	

En cuanto a las conexiones podemos hacerlo de 4 maneras diferentes según nuestro objetivo:

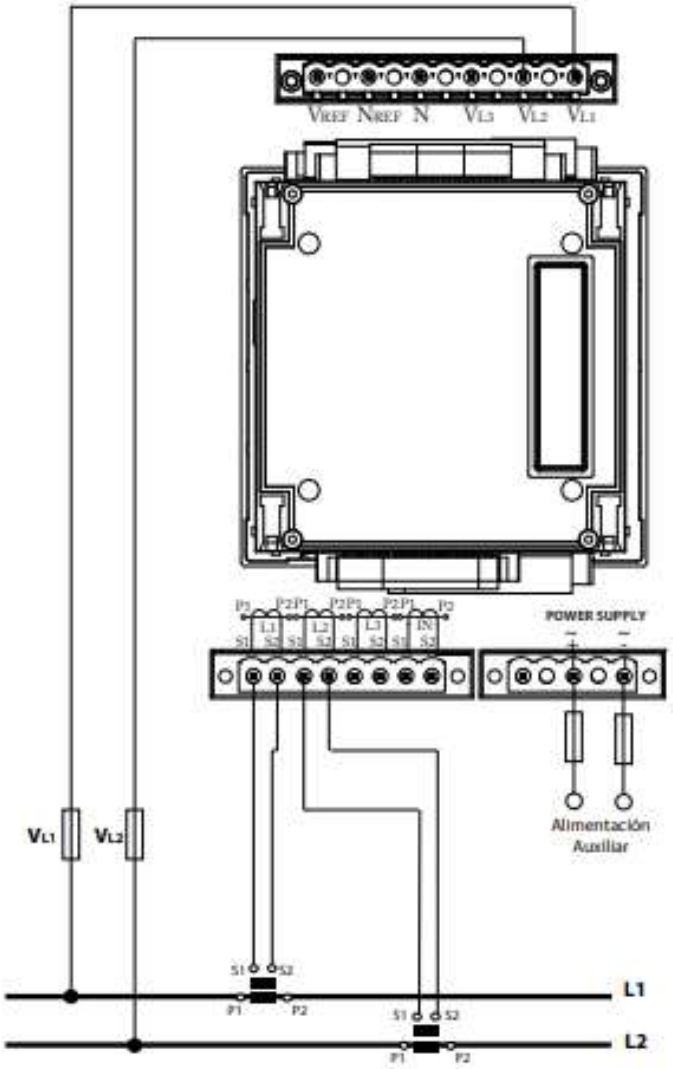
Medida trifásica con o sin transformador de tensión y transformadores de corriente



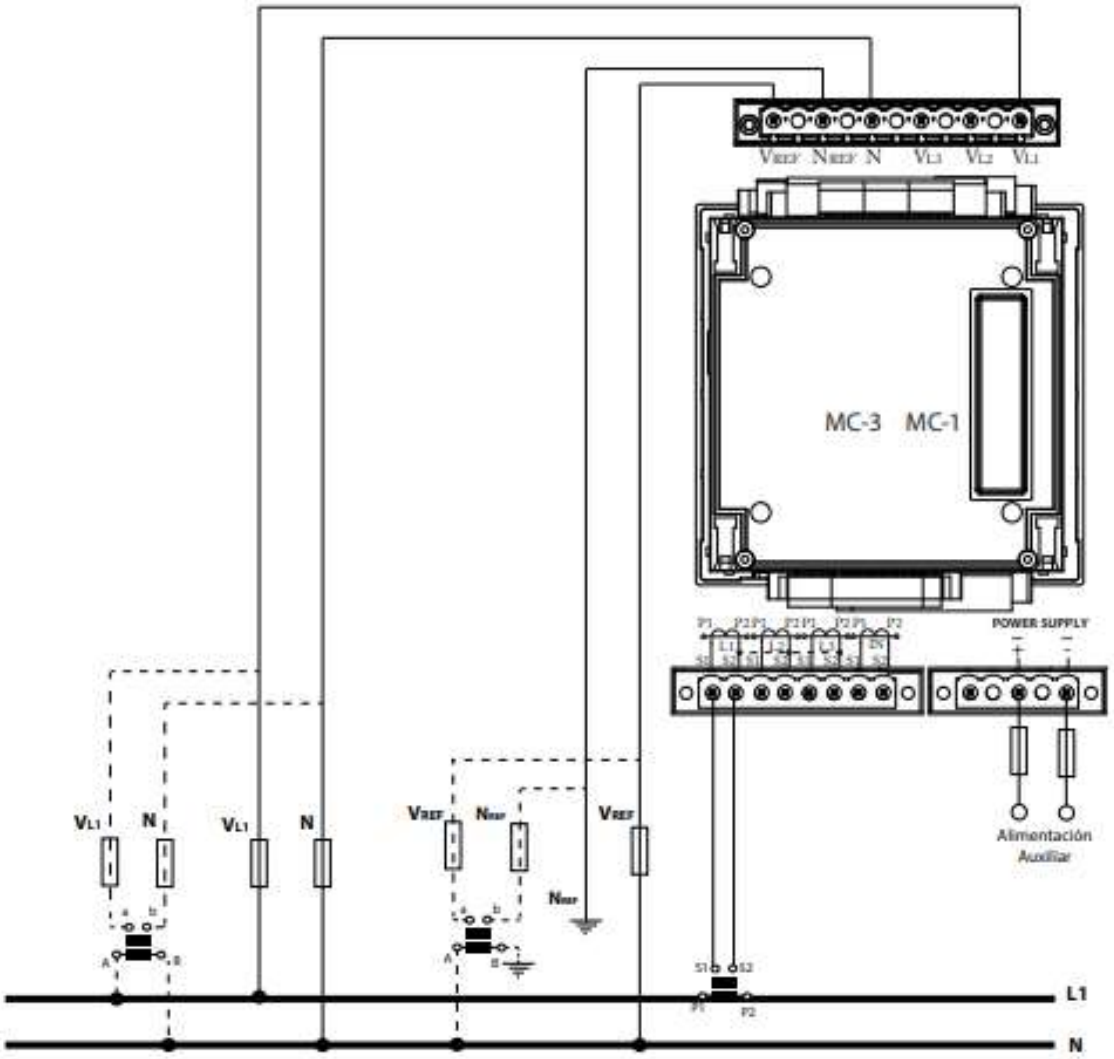
Medida trifásica con o sin transformador de tensión y transformadores tipo MC-3 (1250 mA) + MC1 para corriente de neutro



Medida directa fase-fase con transformadores de corriente



Medida en sistema monofásico con o sin transformador de tensión



El equipo también dispone de:

- 3 teclas, que permiten moverse por las diferentes pantallas y realizar la programación del equipo.
- 3 LED de indicación: CPU, ALARMA y en las teclas de navegación.
- Display LCD de 5.6'', para visualizar todos los parámetros.
- 2 entradas digitales, para la selección de la tarifa, para detectar el estado lógico de señales exteriores o como entrada de impulsos.

El **CVM-B150** se puede ampliar con los siguientes módulos de expansión:

- ✓ **M-CVM-AB-8I-8OTR**, módulo de expansión con 8 entradas digitales de 8 salidas digitales de transistor.
- ✓ **M-MVM-AB-8I-8OR**, módulo de expansión con 8 entradas digitales y 8 salidas digitales de relés.
- ✓ **M-CVM-AB-4AI-8OA**, módulo de expansión de 4 entradas y 8 salidas analógicas.
- ✓ **M-CVM-AB-Modbus TCP (Bridge**), módulo de expansión que permite conectar el equipo a una red Modbus/TCP y realizar las funciones de pasarela Ethernet a RS-485.
- ✓ **M-CVM-AB-LON**, módulo de expansión que permite conectar al equipo a una red LonWorks.
- ✓ **M-CVM-AB-Profibus**, módulo de expansión que permite conectar el equipo a una red Profibus.
- ✓ **M-CVM-AB-MBus**, módulo de expansión que permite conectar el equipo a una red M-Bus.
- ✓ **M-CVM-AB-Datalogger**, módulo de expansión que permite el almacenamiento de datos bajo la plataforma PowerStudio embebido integrada en el módulo.
- ✓ **M-CVM-AB-Modbus TCP (Switch),** módulo de expansión que permite conectar el equipo a una red Modbus/TCP y realizar funciones de pasarela Ethernet a Ethernet.

Este equipo tiene la posibilidad de mostrar los parámetros eléctrico que se muestran en la siguiente *Tabla 3.1*.

Tabla 3.1. Parámetros de medida del CVM-B150

Parámetro	Unidades	Fases L1-L2-L3	N	TOTAL III
Tensión fase-neutro	Vph-N	✓	✓	✓
Tensión fase-fase	Vph-ph	✓		✓
Corriente	A	✓	✓	✓
Frecuencia	Hz	✓		✓
Potencia Activa	kW	✓		✓
Potencia Aparente	kVA	✓		✓
Potencia Reactiva Total	kvar	✓		✓
Potencia Reactiva Inductiva	kvarL	✓		✓
Potencia Reactiva Capacitiva	kvarC	✓		✓
Factor de potencia	PF	✓		✓
Cos φ	φ	✓		✓
Energía Activa Total	kWh	✓		✓
Energía Reactiva Inductiva Total	kvarLh	✓		✓
Energía Reactiva Capacitiva Total	kvarCh	✓		✓
Energía Reactiva Total	kvarh	✓		✓
Energía Aparente Total	kVAh	✓		✓
Energía Activa Tarifa 1	kWh	✓		✓
Energía Reactiva Inductiva Tarifa 1	kvarLh	✓		✓
Energía Reactiva Capacitiva Tarifa 1	kvarCh	✓		✓
Energía Reactiva Total Tarifa 1	kvarh	✓		✓
Energía Aparente Tarifa 1	kVAh	✓		✓
Energía Activa Tarifa 2	kWh	✓		✓
Energía Reactiva Inductiva Tarifa 2	kvarLh	✓		✓
Energía Reactiva Capacitiva Tarifa 2	kvarCh	✓		✓
Energía Reactiva Total Tarifa 2	kvarh	✓		✓
Energía Aparente Tarifa 2	kVAh	✓		✓
Energía Activa Tarifa 3	kWh	✓		✓
Energía Reactiva Inductiva Tarifa 3	kvarLh	✓		✓
Energía Reactiva Capacitiva Tarifa 3	kvarCh	✓		✓
Energía Reactiva Total Tarifa 3	kvarh	✓		✓

Energía Aparente Tarifa 3	kVAh	✓		✓
Máxima Demanda de la Corriente, Tarifa 1	A	✓		✓
Máxima Demanda de la Potencia Activa, Tarifa 1	kW	✓		✓
Máxima Demanda de la Potencia Aparente, Tarifa 1	kVA	✓		✓
Máxima Demanda de la Corriente, Tarifa 2	A	✓		✓
Máxima Demanda de la Potencia Activa, Tarifa 2	kW	✓		✓
Máxima Demanda de la Potencia Aparente, Tarifa 2	kVA	✓		✓
Máxima Demanda de la Corriente, Tarifa 3	A	✓		✓
Máxima Demanda de la Potencia Activa, Tarifa 3	kW	✓		✓
Máxima Demanda de la Potencia Aparente, Tarifa 3	kVA	✓		✓

Parámetro	Unidades	Tarifa T1-T2-T3	Tarifa Total
Nº horas de la tarifa activa	hours	✓	✓
Coste	COST	✓	✓
Emisiones CO_2	kgCO_2	✓	✓

Las ventajas de la monitorización realizada es que tenemos un mejor análisis, teniendo los datos en unos periodos de tiempo muy cortos, subiéndolos a la nube donde lo podremos ver en cualquier día y momento, pudiendo crear gráficas y ver la evolución a lo largo del tiempo.

Además, con la app móvil podremos ver siempre el valor de los todos los parámetros a tiempo real, esto conlleva la prevención de posibles problemas gracias a una detección prematura.

3. Arduino

3.1 ¿Qué es arduino?

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso, en nuestro proyecto utilizaremos la placa popular de desarrollo llamada $NODEMCU^{19}$.

Arduino es una placa basada en un microcontrolador ATMEL. Los microcontroladores son circuitos integrados en los que se pueden grabar instrucciones, las cuales las escribes con el lenguaje de programación que puedes utilizar con el entorno Arduino IDE, que puede ser utilizado, configurado y programado a nuestra necesidad.

Dado su barato coste y difusión, abundan en el mercado sensores y actuadores que se pueden conectar directamente a la placa utilizándolos de una manera muy cómoda.

3.2 Software y Hardware libre

Definimos como software libre aquel que puedes ser usado, copiado, modificado y redistribuido libremente incluyendo el código fuente, de esta manera facilitando el control, implementación y mejores en el diseño por la comunidad de desarrolladores.

Debemos tener en cuenta que por el simple hecho de que sea libre no conlleva a que sea gratis. Pero el coste es lo que más lo diferencia con respecto al software con licencia.

Con respecto al hardware libre toma las mismas ideas del software libre para aplicarlas en su campo, con objetivo de crear diseños de aparatos electrónicos de forma que las personas puedan acceder, como mínimo, a los planos de construcción de los dispositivos.

3.3 Microcontrolador de Arduino

El microcontrolador de Arduino posee lo que se llama una interfaz de entrada, que es una conexión en la que podemos conectar en la placa diferentes tipos de periféricos. La información de estos periféricos que conectes se trasladará al microcontrolador, el cual se encargará de procesar los datos que le lleguen a través de ellos.

También cuenta con una interfaz de salida, que es la que se encarga de lleva la información que se ha procesado en el Arduino a otros periféricos. Estos periféricos pueden ser pantallas o altavoces en los que se puede reproducir los datos procesados, pero también pueden ser otras placas o controladores.

Su entorno de programación es medianamente simple y claro cómo se verá a continuación.

3.4 IDE Arduino

El entorno de desarrollo integrado también llamado IDE (sigla en inglés de Integrated Development Environmnet), es un programa informático compuesto por un conjunto de herramientas de programación.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los sketches constan básicamente de una cabecera, la cual se definen librerías (#include), se declaran variables globales y se crear las instancias necesarias de cada librería. En el void setup (), se lleva a cabo la configuración inicial de las variables las cuales se ejecutarán solo una vez al iniciar el programa, y por último el void loop (), se incluye las tareas que se quieren realizar y se repiten de manera infinita.

Los programas de Arduino están compuestos por un solo fichero con extensión “ino”, aunque es posible organizarlo en varios ficheros. El fichero principal siempre debe estar en una carpeta con el mismo nombre que el fichero.

En la imagen de la *Figura 3.3* mostramos el entorno de programación.

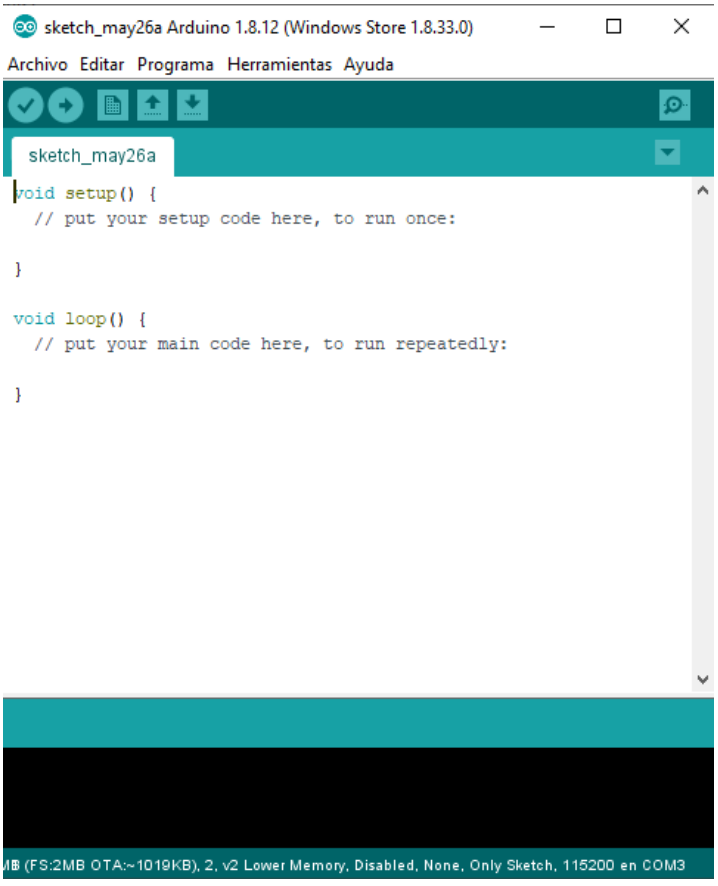


Figura 3.3: IDE Arduino

3.4.1 Librerías Arduino

Las librerías son trozos de código hechas por terceros que usamos en nuestro sketch. Disponemos de infinidad de ellas y para instalar una librería basta con descargarla en formato .zip y en el menú Programa/incluir librería, hacer clic en la opción añadir biblioteca .ZIP …

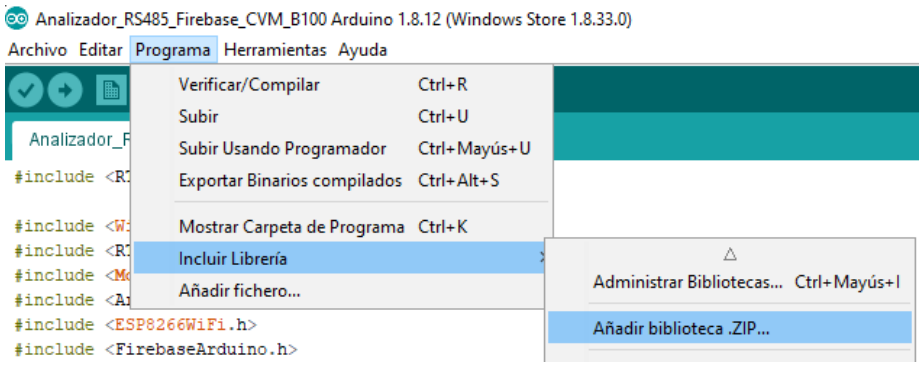


Figura 3.4: Incluir Librería Arduino

3.4.2 Cargar un programa en la placa Arduino

Para cargar y compilar un programa antes debemos seleccionar el modelo de placa Arduino, que en nuestro caso deberemos escoger el nodeMCU 1.0 como veremos a continuación:

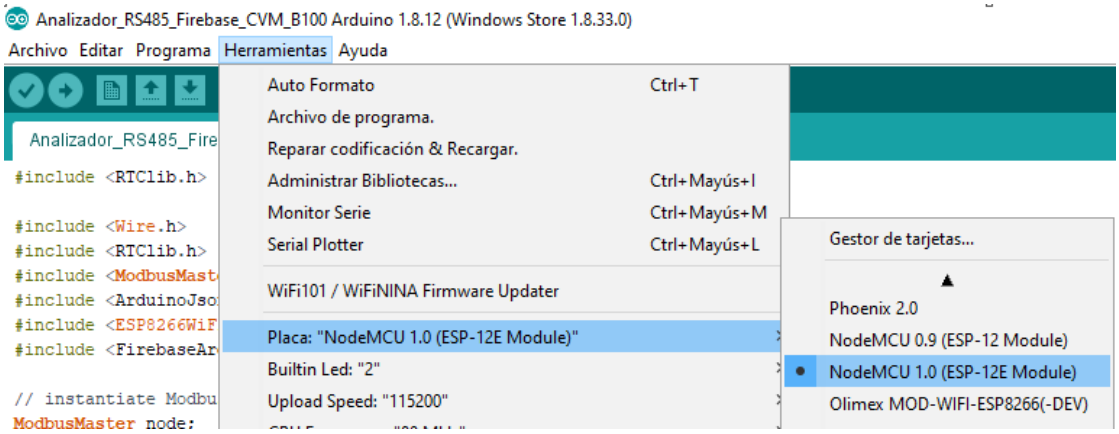


Figura 3.5: Seleccionar placa Arduino

Una vez elegida la placa, debemos seleccionar el puerto serie utilizado en la parte inferior del submenú Herramientas:

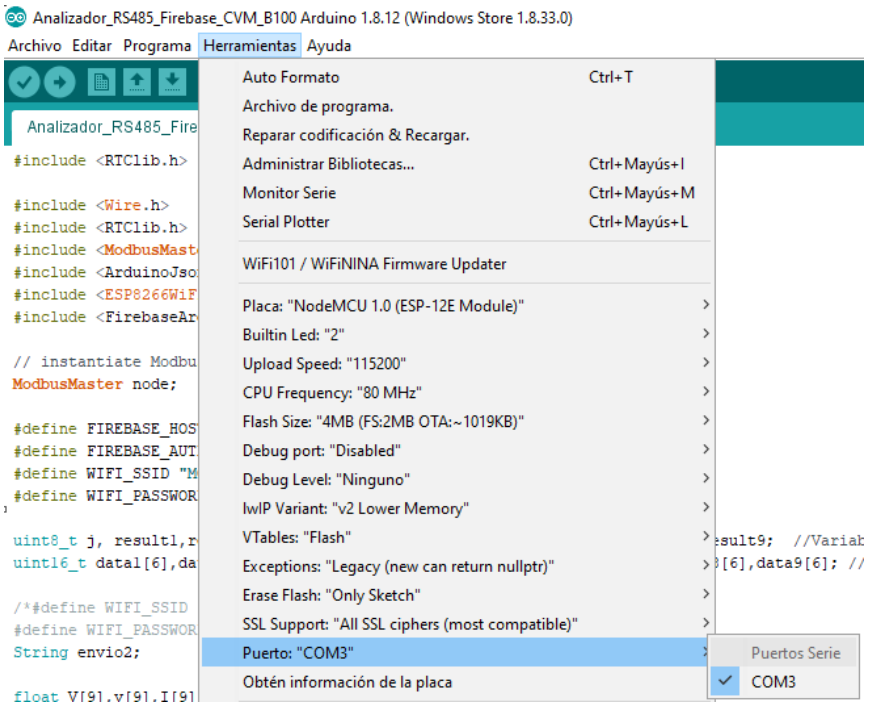


Figura 3.6: Selección del puerto serie

Finalmente, una vez escrito nuestro programa verificamos que está escrito correctamente, lo cargamos:

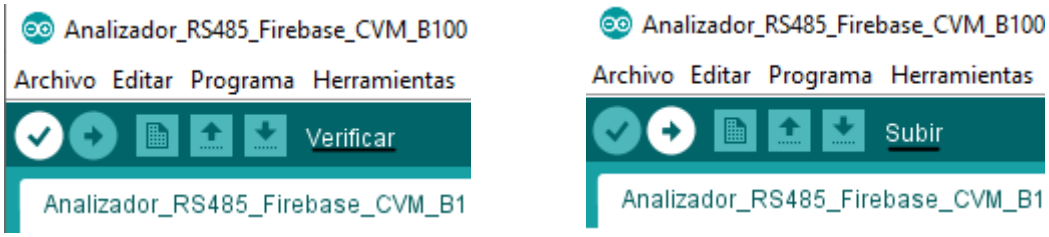


Figura 3.7: Verificar y subir a la placa

4 Placas y comunicación utilizada

4.1 NodeMCU

Esta placa fue una de las primeras placas de desarrollo con el microcontrolador ESP8266. NodeMCU se popularizó rápidamente porque permitía programar este microcontrolador de una manera mucho más sencilla. Para utilizarla no hace falta nada más que un cable USB y un ordenador.

No hay que confundir microcontrolador con placa de desarrollo. NodeMCU no es un microcontrolador, son placas o kits de desarrollo que llevan incorporados un chip que se suele llamar SoC (System on a Chip) que dentro tiene un microcontrolador o MCU. Éste serie el esquema general de este tipo de placas:

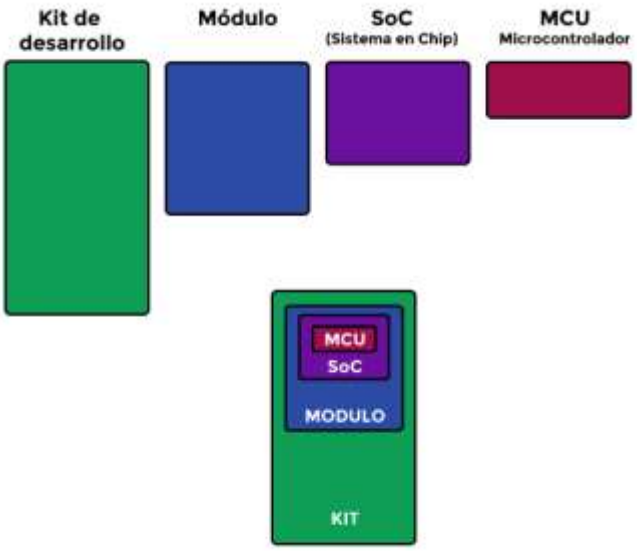


Figura 4.1: Esquema general de un kit de desarrollo

El nombre de NodeMCU representa por tanto la unión de la placa de desarrollo junto con el firmware. Este permite escribir código con el lenguaje de programación LUA, aunque con la aparición de la integración con el IDE de Arduino, este firmware ha caído poco a poco en desuso.

En cuanto a hardware, nodeMCU integra el ESP8266 con un nombre técnico de ESP8266EX. Se trata de un SoC, que básicamente consiste en un chip que tiene todo integrado para que pueda funcionar de forma autónoma como si fuera un ordenador. En el caso del ESP8266 lo único que no tiene es una memoria para almacenar los programas y esto supone un inconveniente.

Como todas las placas que utilizan ESP8266, se le puede cargar cualquier firmware y puede usarse desde el propio de NodeMCU con lenguaje LUA, a MicroPython. También se puede utilizar como una placa de Arduino, donde hacemos el firmware desde cero.

La alimentación el ESP12 se alimenta con 3.3V, pero NodeMCU incluye un regulador de tensión, lo que nos permite alimentarla por USB con 5V, por ejemplo, desde el ordenador que estamos utilizando para programarla. Este regulador tiene un consumo residual de 8mA, lo que hace que esta placa no sea adecuada como módulo definitivo en un proyecto que se requiera un bajo consumo.

La placa consta de 11 pines de entrada salida y el puerto de entrada analógica. Los pines GPIO del 6 al 11 están también conectados, pero no es muy recomendable usarlos. Los pines GPIO 1 y 3 corresponden con el Rx y el Tx del puerto serie. Tiene 2 botones, uno conectado al pin de RESET y otro al GIPIO0, que permite activar el modo de carga de firmware, aunque no es necesario ya que usamos el IDE de Arduino. NodeMCU también facilita el uso del puerto analógico.

En la siguiente imagen puedes ver una visión general de todos los pines del NodeMCU:

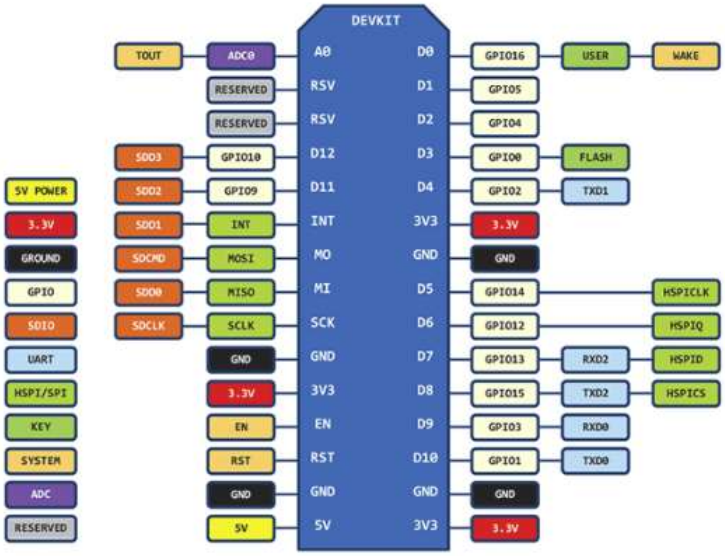


Figura 4.2: Pines del NodeMCU

Como norma, siempre que hagamos referencia a este diagrama de pines de NodeMCU, equivaldrá a si ponemos la placa con el puerto USB hacia abajo como se muestra en la *figura 4.3:*

Figura 4.3: Pines del NodeMCU referenciados

En cuanto a su tamaño y precio, es un poco mayor que el módulo ESP12, siendo sus medidas 49 mm de alto por 24,5 mm de ancho con un precio económico de alrededor de 8€.

Figura 4.4: NodeMCU

4.2 Módulo de reloj en tiempo real (RTC)

El módulo de reloj RTC (Real Time Clock) está diseñado para controlar fecha y hora con alta precisión, por medio del micro controlador DS3231. Conectado al Arduino, el software dispone de acceso informativo a la fecha y ahora actual (*figura 4.5*)

El software de Arduino es capaz de realizar instrucciones temporizadas, o contar el tiempo de un proceso; pero a diferencia del RTC, tiene una gran desviación del valor real del tiempo debido a que su hardware no fue diseñado con esa prioridad. Lo mismo sucede en ordenadores, servidores, etc. Aumenta el rendimiento si se dedica el procesador principal a operaciones más importantes y añadir por separado un módulo RTC.

Además, tiene la ventaja de que cuenta con una pila de botón CR2032 en la parte trasera, que alimenta el módulo en caso de desconexión de la alimentación, manteniendo actualizados los datos de fecha y hora (*figura 4.6*).

El módulo es compatible con Arduino y se alimenta a la misma tensión que él. La comunicación funciona por el puerto 12C, es decir mediante los pines SCL (CLOCK) y SDA (DATA). Los pines sobrantes sirven para añadir un sensor de temperatura que en este caso no es necesario. Tratándose de un módulo de hardware interno, se diseña un acoplamiento a la PCB mediante conectores.

Especificaciones del módulo RTC:

- Tensión de alimentación: 5 Vdc.
- Tensión de batería: 3,3 Vdc.
- Consumo máximo de corriente: 0,3 mA.
- Comunicación: 12 C.
- Dirección: 0x68.
- Dimensiones: 36x22 mm.
- Tipo de montaje: PCB mediante conectores de agujero pasante.

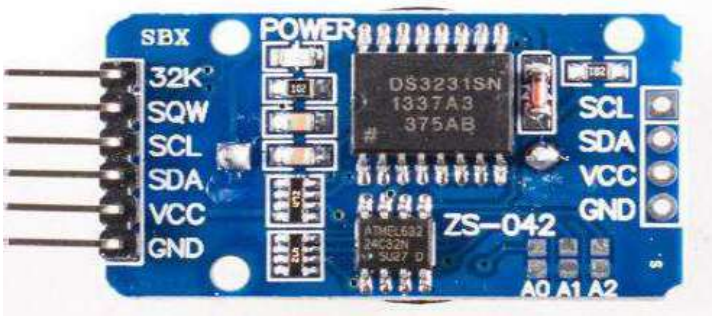


Figura 4.5: RTC vista de la planta superior

Figura 4.6: RTC vista de la planta inferior.

4.3 Arduino Mega

Este Arduino tiene probablemente el microcontrolador más capaz de la familia arduino. Posee 54 pines digitales que funcionan como entrada/salida; 16 entradas análogas, un cristal oscilador de 16 MHz, una conexión USB, un botón de reset y una entrada para la alimentación de la placa (*Figura 4.7*), más adelante explicaremos cada uno de ellos:

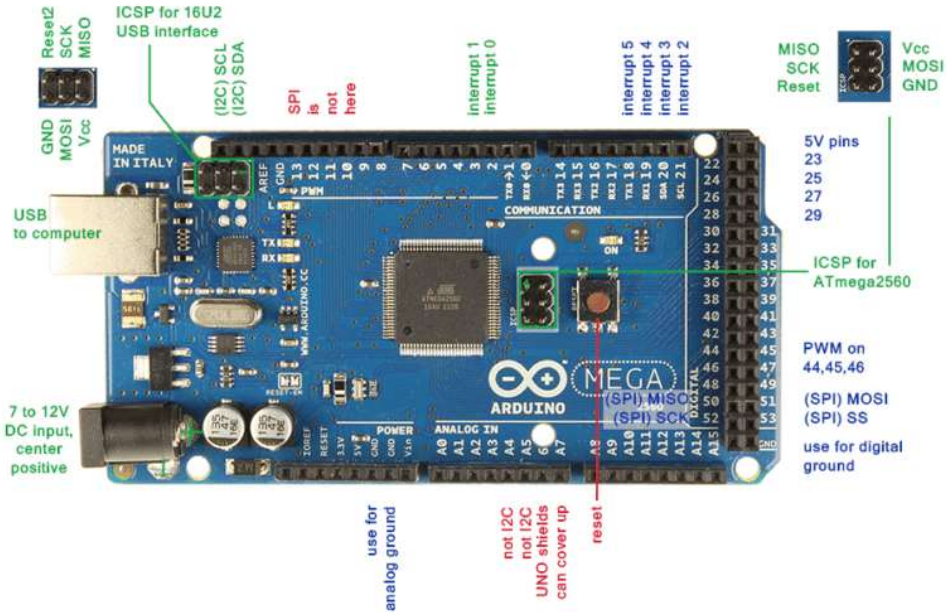


Figura 4.7: Pines Arduino Mega

La comunicación entre la computadora y Arduino se produce a través del Puerto Serie. Posee un convertidor USB-serie, por lo que sólo se necesita conectar el dispositivo a la computadora utilizando un cable USB como el que utilizan las impresoras.

Necesitamos esta placa Arduino para poder captar la información recibida por el analizador y escribirla por otro puerto serie, ya que iniciamos el node en el puerto serie con el nodeMCU y se sobre escriben los datos.

```
node.begin(1, Serial);
```

Figura 4.6: Inicialización de node en Serial

Las características técnicas del Arduino Mega:

- **Microcontrolador**: ATmega2560.
- **Voltaje operativo**: 5 V
- **Voltaje de entrada**: 7-12 V
- **Voltaje de entrada (límite)**: 20 V
- **Pines digitales de entrada/salida**: 54 (de los cuales 15 proveen salida PWM)
- **Pines análogos de entrada**: 16
- **Corriente DC por cada pin entrada/salida**: 40 mA
- **Corriente DC entregada en el pin 3.3 V**: 50 mA
- **Memoria Flash**: 256KB (8KB usados por el bootloader)
- **SRAM**: 8 KB

A continuación, explicaremos cada parte del Arduino Mega:

- **Botón de reset**: Pulsador que su misión es resetear la placa, cuando se pulsa, la placa nos ofrece el último programa que se cargó.
- **Puerto USB**: Conector el cual se utiliza para la conexión entre Arduino y PC o viceversa.
- **Fuente de alimentación externa**: Se trata de un conector Jack de 2.1mm instalado en una fuente de tensión de entre 7 y 12 voltios.
- **Pines de alimentación**:
    - IORF: Pin de referencia de tensión a la cual debe trabajar el microcontrolador.
    - RESET: Resetea la placa, igual que el pulsador descrito anteriormente.
    - 3,3 V: Este pin ofrece 3,3 voltios, será necesario para los sensores o actuadores que tengan que

trabajar a una tensión menor a la que puede ofrecer Arduino en el resto de pines (5 V).
    - 5 V: Dicho pin ofrece 5 V. La gran parte de sensores o actuadores que se conectan a Arduino tendrán que ser alimentación a dicha tensión.
    - GND: Este pin es la toma de tierra (0V).
    - Vin: Con este pin se podrá alimentar a Arduino.

- **Pines analógicos**: Dichos pines podrán configurarse como entrada o salida y se utilizarán cuando se empleen sensores que proporcionen información analógica.
- **Microcontrolador ATmega**: Elemento que se encargar de ejecutar las instrucciones de los programas creados en la IDE de Arduino.
- **Tx**-**Rx**: Informarán al usuario, parpadeando, de la comunicación entre Arduino y el Pc por vía puerto serie.
- **Conectores ICSP**: Dichos conectores se utilizan cuando se programa Arduino desde un entorno diferente del IDE y de la conexión por USB.
- **Indicador de encendido**: Se trata del led verde que nos indica que Arduino está alimentado adecuadamente.
- **Indicador de carga**: Nos informa que un programa se está cargando en Arduino mediante un led que parpadeara.
- **Pines digitales**: Tienes la misma función que los analógicos pero la información se transmite de Arduino al exterior y vicerversa.

Cuando se trabaja con una fuente externa de poder se debe utilizar un convertidor AC/DC y regular dicho voltaje en el rango operativo de la placa. De igual manera se puede alimentar el micro mediante el uso de baterías.

4.4 Shield RS485

El interfaz RS485 (también llamado TIA485) es un estándar de la capa física de la comunicación del modelo de interconexión de sistemas abiertos o también llamado modelo OSI (Open System interconnection). La capa física es el canal de comunicación y el método de transmisión de la señal. RS485 fue creado con el fin de ampliar las capacidades físicas de la interfaz RS232.

La tecnología RS485 sigue siendo la base de muchas redes de comunicación, donde sus principales ventajas son:

- Intercambio de datos bidireccional a través de un par de hilos trenzados.
- Capacidad de interconexión, soportando varios transceptores conectados a la misma línea.
- Gran longitud de la línea de comunicación.
- Alta velocidad de transmisión.
- Rechazo al ruido de la tierra y en modo común.
- Bajo costo.

La red de comunicaciones construida en dicha interfaz consta de transceptores (dispositivo que cuenta con un transmisor y un receptor dentro de la misma caja) conectados por un cable de par trenzado. El principio básico de la interfaz RS485 es la transmisión de datos diferencial (equilibrada), esto significa que la señal es transportada por dos cables. Con esto, un cable del par transmite la señal original y el otro transporta su copia inversa.

En cuanto a la comunicación de dicha interfaz, no pueden transmitir y recibir datos al mismo tiempo por su naturaleza, lo que lleva a un conflicto de transmisores. El protocolo de comunicación RS485, los comandos son enviados por el nodo definido como maestro, todos los demás nodos conectados al maestro reciben los datos a través de

puertos RS485. Dependiendo de la información enviada, cero o más nodos en la línea responden al maestro.

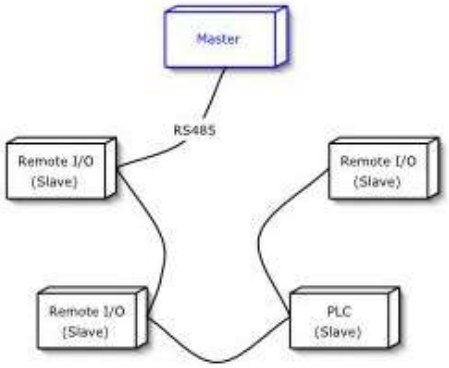


Figura 4.5: Comunicación RS485

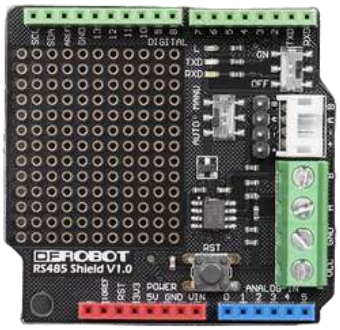


Figura 4.6: Shield RS485

Además, permite enviar datos a largas distancias y a velocidades relativamente altas, 100kbits/s a unos 1200 metros. Por ejemplo, un cable de 20 metros permite una velocidad de transmisión de datos máxima de 5Mbits/s.

La shield RS485 lleva integrado un puerto RS485 estándar, un puerto mini RS485 y encabezados RS485. Además, proporciona un switch para cambiar entre el modo automático y el modo manual, lo que amplía el ámbito de aplicación. También incluye un interruptor para deshabilitar esta shield cuando arduino necesita ser programado.

Una de las características principales que diferencia la comunicación RS485 de cualquier otra comunicación serie es el formato de los datos intercambiados, la mayoría de los dispositivos RS485 utilizan MODBUS.

4.4.1 Comunicación MODBUS

MODBUS es un protocolo estándar que puede gestionar una comunicación tipo cliente-servidor entre distintos equipos conectados físicamente con un bus serie. Este protocolo fue ideado para los PLCs Modicon en 1979, y con el tiempo se ha convertido en un protocolo muy empleado en las comunicaciones industriales. Las principales razones de ello son la sencillez del protocolo, versatilidad, y que sus especificaciones, gestionadas por la MODBUS Organization, son de acceso libre y gratuito.

MODBUS es un protocolo de tipo Petición/Respuesta, por lo que en una transacción de datos se puede identificar al dispositivo que realiza una petición como el cliente o maestro, y al que devuelve la respuesta como el servidor o esclavo de la comunicación. En una red MODBUS se dispone de un equipo maestro que puede acceder a varios equipos esclavos. Cada esclavo de la red se identifica con una dirección única de dispositivo.

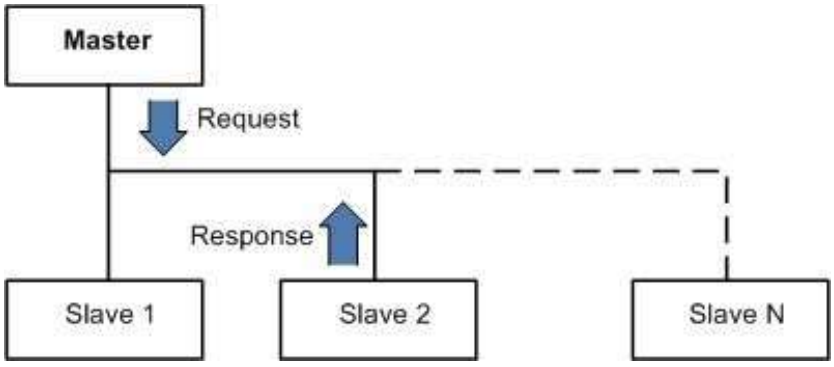


Figura 4.7: Maestro/Esclavo MODBUS

Un maestro puede hacer dos tipos de peticiones a un esclavo: enviar datos a un esclavo y esperar su respuesta de confirmación, o para pedir datos a un esclavo y espera su respuesta con los datos. Las peticiones de lectura y escritura que envía a un maestro llevan asociado

un código de función que el esclavo debe ejecutar. Según ese código, el esclavo interpretará los datos recibidos del maestro y decidirá qué datos debe devolver. Los códigos de función dependen de los dispositivos y de las tareas que estos pueden realizar.

Para intercambiar las peticiones y respuestas, los dispositivos de una red MODBUS organizan los datos en tramas. Dado que MODBUS es un protocolo de nivel de aplicación, se requiere utilizarlo sobre una pila de protocolos que resuelva los temas específicos del tipo de red empleada. En función de la arquitectura de protocolos usada, se distinguen 2 tipos de MODBUS: RTU y ASCII (*Figura 4.8*).

Estos tipos de protocolos están pensados para ser utilizadas directamente sobre un medio físico serie asíncrono, como por ejemplo RS232, RS485 o RS422.

Hay otros tipos llamados *MODBUSTCP*, que no es utilizado en dicho proyecto ya que está desarrollado para funcionar sobre redes que utilizan la arquitectura TCP/IP por lo que permite usar MODBUS sobre redes como Ethernet o Wifi; o *BACnet*, que es uno de los protocolos de comunicación más completo y potente en la automatización de edificios utilizado en el CVM-B150 para comunicar entre sí a los diferentes aparatos electrónicos.

En la *Figura 4.8* muestra el formato de trama de las dos opciones de MODBUS, que se describen con más detalle a continuación.

Los campos **Función** y **Datos** representan la trama de nivel de aplicación de MODBUS, y dependen de las distintas opciones de peticiones y respuestas que describirán en el apartado 0. El tamaño del campo de datos siempre depende de la función utilizado. Para la colocación de los bytes en los distintos campos hay que tener en cuenta que MODBUS siempre utiliza codificación BIG-ENDIAN, según la cual el byte más significativo se envía primero (a la izquierda).

La **dirección** es un valor que debe identificar unívocamente a un dispositivo esclavo de la red. Este valor de identificación debe corresponderse con un número entre 1 y 247 en configuraciones multipunto, como los buses RS422 y RS485 que tienen como mínimo un maestro y un esclavo. El valor especial de dirección 248 se utiliza sólo cuando MODBUS se emplea sobre una conexión punto a punto, por ejemplo, con un maestro y solo un esclavo en una conexión RS232. Por último, el valor 0 es la dirección de difusión o *broadcast*, y una petición enviada a esta dirección es atendida por todos los esclavos. Este tipo de peticiones no producen una respuesta de los esclavos ya que no se podría controlar el acceso al medio de esto y habría colisiones. Por eso mismo, tampoco se recibirá ninguna respuesta si se ejecuta una petición de lectura con dirección de *broadcast*.

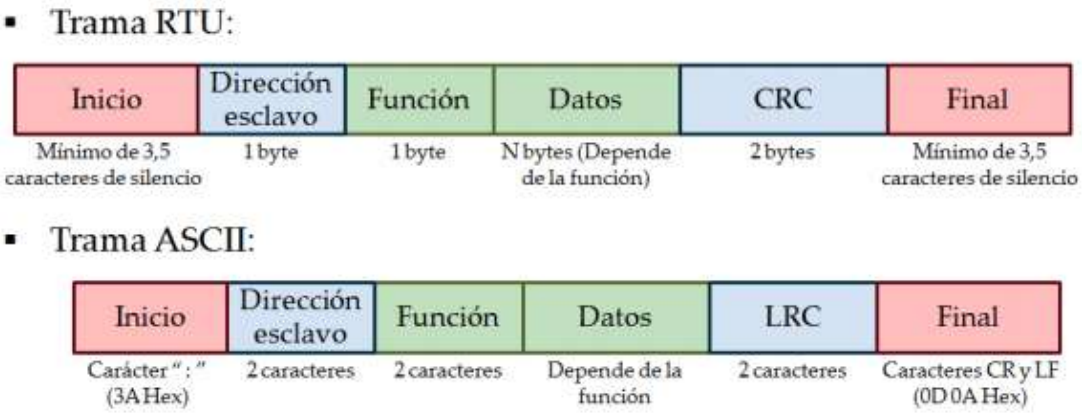


Figura 4.8: Formato de las tramas de MODBUS serie

5. Firebase

5.1 ¿Qué es Firebase?

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por James Tamplin y Andrew Lee en 2012 y adquirida por google en 2014.

Figura 5.1: Icono Firebase

Tras lanzar el servicio chat, Tamplin y Lee observaron que dicho chat estaba siendo utilizado en gran escala por los desarrolladores para pasar paquetes de información de sus aplicaciones, como el estado de las partidas en el caso de los juegos. Fue entonces cuando decidieron separa ambas funcionalidades, el sistema de chat y el sistema de arquitectura en tiempo real que lo propulsaba, dando como resultado la fundación de Firebase en abril de 2012.

La base de datos a tiempo real es una base alojada en la nueve con formato JSON. Los datos se sincronizan a tiempo real con la aplicación. La comunicación es bidireccional, es decir que cuando modifico datos en la bbdd estos cambios se van reflejados directamente en la web y viceversa.

Con respecto a los servicios que nos ofrece vamos a ver los de más peso:

- **Rapidez.** Es una de sus características principales, aparte de tener funcionalidades que ayudan a optimizar el tiempo dedicado al desarrollo y optimización.

- **Infraestructura**. Implementar Firebase es rápido es fácil. Gracias a API intuitivas contenidas en un solo SDK, puedes concéntrate en resolver los problemas de los clientes y evitar perder tiempo en crear una infraestructura compleja.

- **Analítica**. Firebase Analytics es la solución de análisis gratuita e ilimitada directamente integrada a Firebase. Funciona con otras características de Firebase para que puedas controlarlo todo, desde la tasa de clics hasta los fallos de la app.

- **Multiplataforma**. Ofrece apps multiplataforma con API integradas a SDK individuales para Android, IOS y JavaScript.

- **Notificaciones.** Podrás gestionar la programación y envío de notificaciones push a los usuarios de tus apps, algo realmente útil para mantenerlos al día de las últimas novedades.

- **Soporte gratuito.** Nos ofrece un soporte gratuito por correo electrónico a todos los desarrolladores.

5.2 Condiciones para utilizar Firebase

Para trabajar con la herramienta Firebase debes acceder a través de un navegador web si se dispone de una cuenta de usuario de Google. A continuación, explicaremos como crear un proyecto y cuáles son los datos importantes que tendremos que tener en cuenta:

1) Tendremos que abrir el navegador que tengamos en nuestro Pc e ir a la página oficial de Firebase: https://firebase.google.com/?authuser=1.

2) Introducimos nuestra identificación de Google: *usuario y contraseña.*

3) Pinchamos en *Ir a la consola* en la parte derecha del menú de barras. También podemos cambiar de idioma en la lista desplegable de su izquierda.

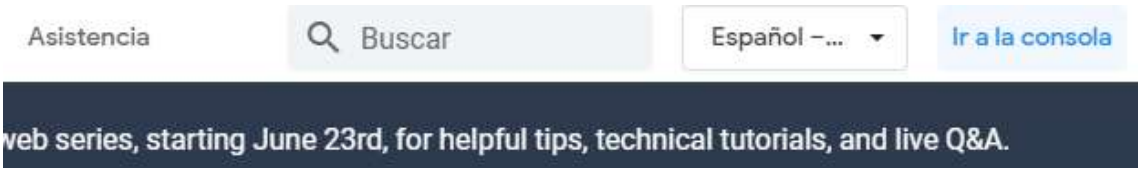


Figura 5.2: Acceder a nuestro perfil

4) Una vez dentro, pinchando en *añadir proyecto* crearemos un nuevo proyecto poniendo el título que sea adecuado.

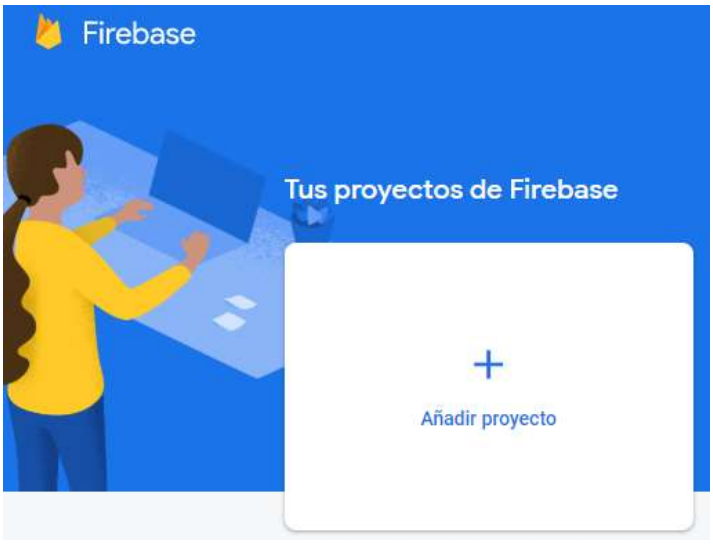


Figura 5.3: Añadir proyecto

5) Creado el proyecto, tendremos una barra de opciones en la parte izquierda, donde en *database* podremos ver los datos subidos, en nuestro caso por arduino.

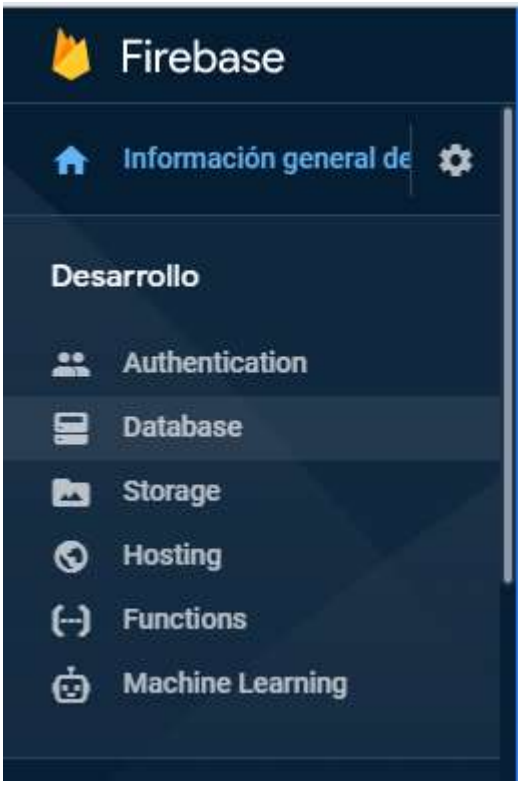


Figura 5.4: Añadir proyecto

Hay dos datos muy importantes los cuales los tendrás que tener en cuenta a la hora de la comunicación de los dispositivos y subida de datos.

En la zona del menú anterior nos saldrá un enlace el cual lo necesitaremos para ponerlo en arduino (*Figura 6.4*) junto con el secreto de la base de datos, metiéndonos en la rueda de *ajustes* > *Configuración del proyecto* > *Cuentas y servicio* > *Secretos de la base de datos* (*Figura 6.5*).

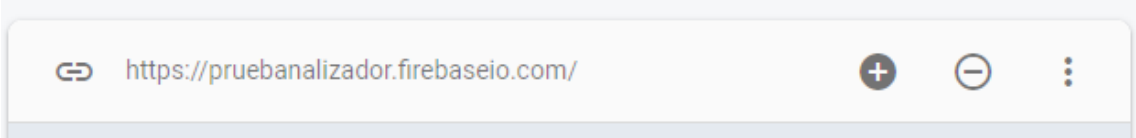


Figura 5.5: Dirección del proyecto

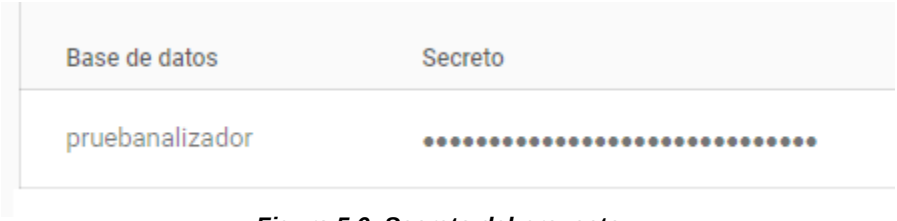


Figura 5.6: Secreto del proyecto

6 App inventor

6.1 ¿Qué es App inventor?

App Inventor es un entorno de desarrollo de software creado por Google para la elaboración de aplicaciones destinadas al sistema operativo Android. Las aplicaciones creadas con App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil permitiéndote almacenar tu trabajo y ayudarte a realizar un seguimiento de tus proyectos.

Figura 6.1: Logo Mit app inventor

Para desarrollar aplicaciones sólo necesitas un navegador web y un teléfono o Tablet Android, si no tienes ninguna de estas opciones podrías probar tus aplicaciones en un emulador.

Para crear una aplicación solo debemos de hacer 3 sencillos pasos:

1. *Diseñar interfaz de usuario*: Muestra la pantalla de un celular el cual podremos insertar y modificar distintos componentes como imágenes, botones, audios …
2. *Editor de bloques*: En esta ventana se dispondrá de la programación necesaria para el funcionamiento adecuado de nuestra aplicación.
3. *Generador de la aplicación*: Terminada la aplicación, se generará un archivo APK (instalador de la aplicación) mediante un código QR para poder visualizarlo en nuestro celular.

6.2 Ventajas e inconvenientes de App Inventor

Con respecto a las ventajas que tenemos de programar con App Inventor encontramos las siguientes:

- Se puede crear aplicaciones por medio de bloques de manera intuitiva y gráfica, sin necesidad de saber código de programación.
- Se puede acceder en cualquier momento y cualquier lugar siempre que estemos conectados a internet.
- Nos ofrece varias formas de conectividad, directa, wifi o por medio de un emulador.
- Nos permite descargar la aplicación mediante el APK de nuestro pc.

Sin embargo, no son varios los inconvenientes que encuentra un usuario de nivel medio o avanzado:

- No genera código Java para desarrollos más profundos.
- Solo se puede desarrollar para el sistema operativo Android.

6.3 Condiciones para utilizar AI2

Para trabajar con la herramienta de AI2 debes acceder a través de un navegador web si se dispone de una cuenta de usuario de Google, esto hace que no sea necesario tener instalado en el ordenador ningún programa específico. El sistema idóneo para crear la aplicación es un ordenador PC (Windows, Mac o Linux) y el navegador recomendado es la última versión de Google Chrome o Mozilla Firefox.

Para iniciar AI2 seguiremos los siguientes pasos:

1. Abrir el navegador que tengamos en nuestro ordenador, preferentemente Google Chrome.
2. Accedemos a la dirección oficial de AI2: https://appinventor.mit.edu/
3. Pinchamos en la opción *Create Apps!*

Figura 6.2: Iniciar aplicación en AI2

4. Introducimos nuestra identificación de Google: *usuario y contraseña.*

5. Para cambiar el idioma de la interfaz, en la barra superior del menú, pulsamos en el idioma que esté por defecto y se desplegará una lista y podremos elegir el conveniente.

Figura 6.3: Cambiar idioma en AI2

6. Para crear o gestionar nuestros proyectos, en la primera pantalla seleccionamos *Mis proyectos*.

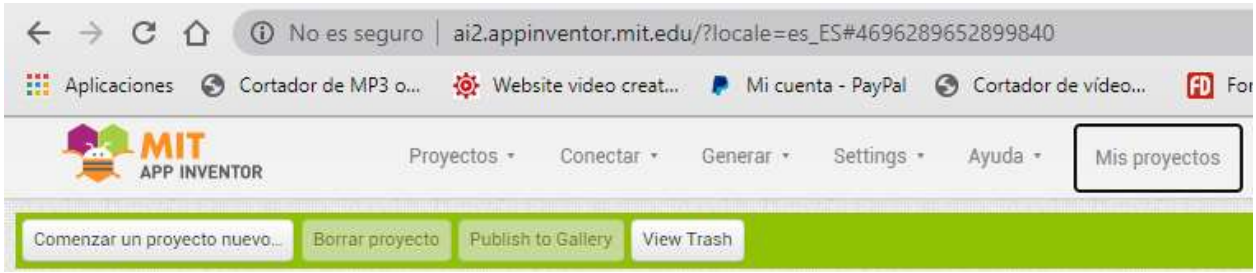


Figura 6.4: Comenzar o gestionar nuestros proyectos

7. Una vez creado un proyecto nuevo, nos saldrá la siguiente pantalla donde podemos ver diferentes bloques:

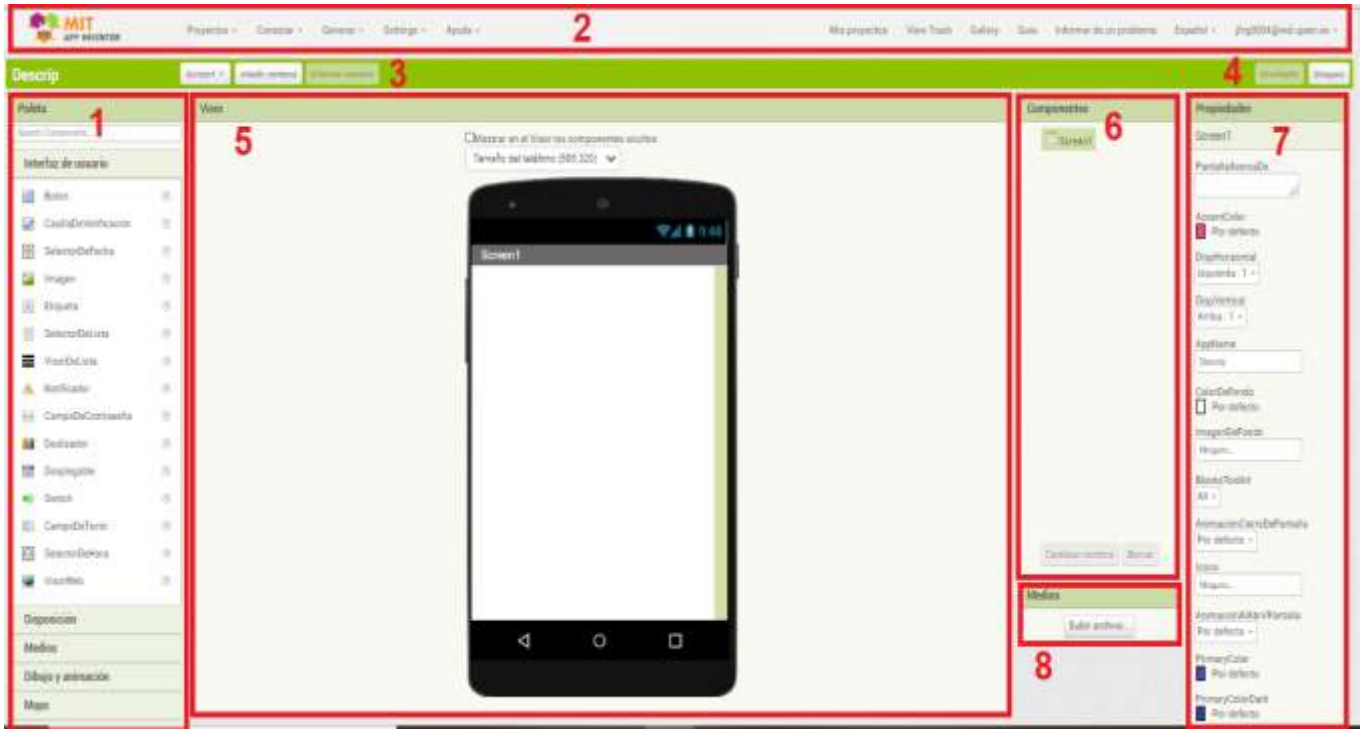


Figura 6.5: Menú al iniciar nuestro proyecto

Ahora explicaremos cada zona una vez creado nuestro proyecto:

1. *Bloques integrados*: Aquí se encuentran en lenguaje de programación organizado por familias.
2. *Barra de menú*: Se encuentran los diferentes botones para trabajar con AI2.
3. *Gestor de pantallas*: Controlamos las diferentes pantallas de nuestro proyecto, pudiendo añadir o eliminar cualquiera.
4. *Diseñador/Bloques*. Con estos dos botones podremos pasar de la ventana de la interfaz de usuario (Diseñador) a la ventana donde se encuentra toda nuestra programación (Bloques).
5. *Visor*: Se trata de la pantalla donde veremos en todo momento como queda nuestra interfaz de usuario y donde colocaremos los componentes necesarios.

6. *Componentes*: En este apartado se puede ver una estructura de tipo árbol que parte del componente *Screen* (Pantalla). Con esta estructura tendremos que ir colocando los componentes de una manera adecuada para que se quede a nuestro gusto.
7. *Propiedades*: En esta zona tendremos las características de nuestros componentes (color de fondo, tamaño de letra, alto, ancho, forma …) que podremos ir cambiando a nuestro antojo.
8. *Medios*: Con este botón podremos *Subir* o *Eliminar* los archivos de imágenes, audio … utilizados en la aplicación.

Sistema para probar en directo las aplicaciones

Existen 3 posibilidades para poder ver la aplicación creada en un dispositivo mientras se está construyendo la misma. Esta prueba es muy recomendable porque no siempre la interfaz se ve lo mismo en el móvil de la aplicación que en el nuestro o cualquier otro.

A continuación, se detallarán las 3 posibles formas.

Pc + WIFI + Dispositivo Android

Ésta es la opción más recomendada. Consiste en instalar la aplicación *MIT AI2 Companion* en el dispositivo Android que tengamos y establecer conexión con la web https://appinventor.mit.edu/ mediante el mismo wifi y probar el trabajo realizado. El requisito imprescindible es que el dispositivo tenga el sistema operativo Android, no siendo válido ningún otro.

Figura 6.6: Conexión Pc+WIFI+Dispositivo Android

Para realizar dicho proceso deberemos seguir los correspondientes pasos indicados a continuación:

1. Descargar e instalar la aplicación *MIT AI2 Companion* en el dispositivo Android elegido (*Figura 7.7*).

Figura 6.7: App para dispositivo Android

2. Una vez dentro de nuestra aplicación en nuestro Pc tenemos que generar un código QR dándole a la pestaña de *Conectar* y *AI Companion* (*Figura 7.8*), aunque también se podría hacer la conexión a través de un código alfanumérico.

Figura 6.8: Generar código QR para la conexión

Si queremos generar el código QR para el archivo APK y tener la aplicación descargada en nuestro móvil independientemente que esté conectado al mismo WIFI, le daremos al botón de la barra de menús *Generar* y pulsaremos la opción indicada (*Figura 7.9*).

Figura 6.9: Generar código QR para el archivo APK

Sin WIFI + PC + Conexión USB con dispositivo Android

En este caso, para probar nuestra aplicación deberemos de utilizar el cable USB desde el PC a la entrada de carga del dispositivo Android. Esta forma se emplea cuando se carece de conexión a internet en los dos dispositivos.

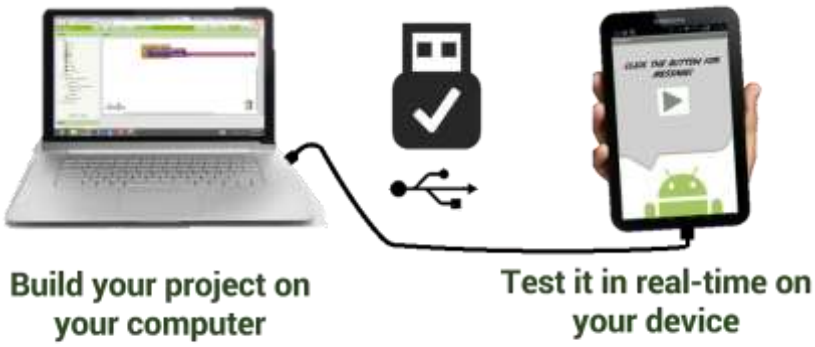


Figura 6.10: Conexión por cable USB

A continuación, se detallan los pasos a seguir para realizar la prueba en directo:

1. Descargar el instalador de MIT App Inventor tools: http://appinv.us/aisetup_windows. Ejecutamos dicho instalador y seguimos sus pasos por defecto.

2. Descargamos e instalamos la app *MIT AI2 Companion* en el dispositivo Android.

3. Para utilizar el cable USB es necesario utilizar un programa llamado aiStarter que permitirá al navegador comunicarse con el cable USB. Una vez ejecutado el programa se dejará abierta la ventana en negro mientras funciona el programa.

4. Configuramos el dispositivo Android en modo depuración accediendo a Ajustes > Opciones de desarrollo > Depuración USB.

5. Conectamos ambos dispositivos usando un cable USB. En el dispositivo Android lo conectamos como dispositivo de almacenamiento masivo y aceptamos en el cuadro *Permitir depuración USB.*

6. Una vez hecho los pasos anteriores, nos vamos a la web de AI2 y seleccionamos en el menú *Conectar* > *USB* (*Figura 7.11*).

Figura 6.11: Paso final para la conexión por USB

Solo uso de PC sin dispositivo Android

Ésta solución la prueba directa se realizará mediante el mismo ordenador donde se crea la aplicación mediante un emulador.

Figura 6.12: Prueba directa con emulador

A continuación, se detallan los pasos para hacer la prueba directa con la aplicación desde el ordenador:

1. Descargar el instalador de MIT App Inventor tools: http://appinv.us/aisetup_windows. Ejecutamos dicho instalador y seguimos sus pasos por defecto.
2. Descargamos e instalamos la app *MIT AI2 Companion* en el dispositivo Android.
3. Conectaremos el simulador, desplazándonos a la barra de menús seleccionaremos *Conectar>Emulador* (*Figura 7.12*). Puede tardar unos segundos, pero al final mostrará la aplicación que está abierta con App Inventor.

Figura 6.13: Paso para conectar simulador

7 Ensayo con jaula de ardilla

En su forma instalada, un rotor o motor de jaula de ardilla es un cilindro montado en un eje. Internamente contiene barras conductoras longitudinales de aluminio o cobre con surcos y conectados juntos en ambos extremos poniendo en cortocircuito los anillos que forman la jaula.

El nombre se deriva de la semejanza entre esta jaula de anillos y barras y la rueda de un hámster. La base del rotor se construye de un apilado hierro de laminación.

Respecto a su funcionamiento, los devanados inductores en el estator de un motor de inducción instan al campo magnético a rotar alrededor del rotor. El movimiento relativo entre este campo y la rotación del rotor induce corriente eléctrica, un flujo en las barras conductoras. Alternadamente estas corrientes que fluyen longitudinalmente en los conductores reaccionan con el campo magnético del motor produciendo una fuerza que actúa tangente al rotor, dando por resultado un esfuerzo de torsión para dar vuelta al eje.

A continuación, veremos la placa característica de dicho motor trifásico:

Figura 7.1: Placa característica del motor trifásico de jaula de ardilla

Utilizando este motor de jaula de ardilla en los ensayos, hemos tomado datos de las siguientes variables con diferentes velocidades y pares respectivamente, los cuales representaremos gráficamente en los siguientes apartados:

- Tensión de línea
- Tensión de fase
- Intensidad de fase
- Potencia activa

- 50 Hz → 0.1; 0.2; 0.3; 0.4; 0.5 y 0.6 Nm
- 45 Hz
- 55 Hz

  → 0.1; 0.3 y 0.6 Nm
- 60 Hz
- 65 Hz

  → 0.1 y 0.3 Nm

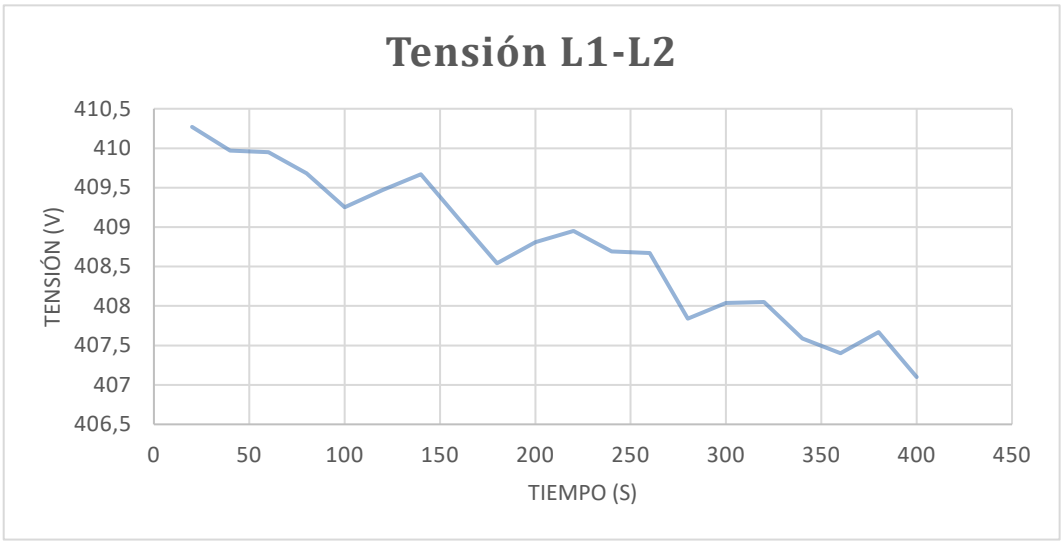
Las condiciones de par las hemos asignado de dicha manera porque no debemos pasar la intensidad máxima del motor, ya que rebasando esa intensidad podemos romperlo.

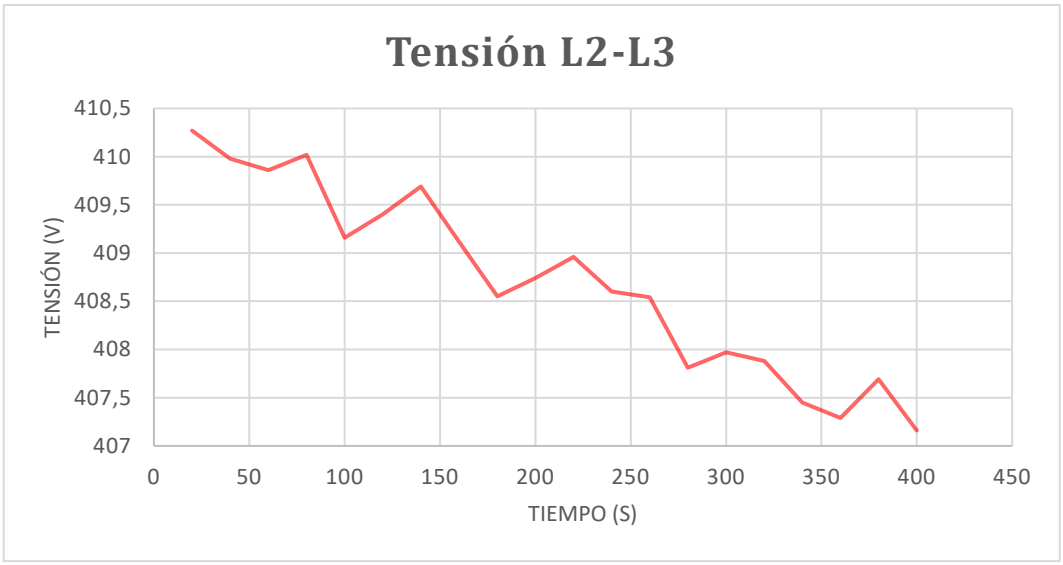
7.1 45 Hz

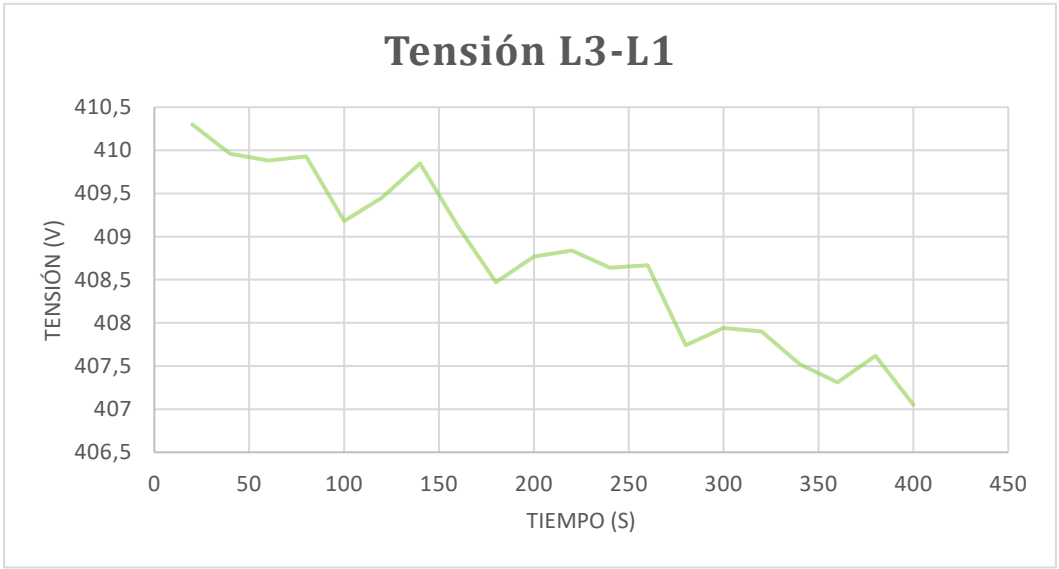
7.1.1 Tensión de línea

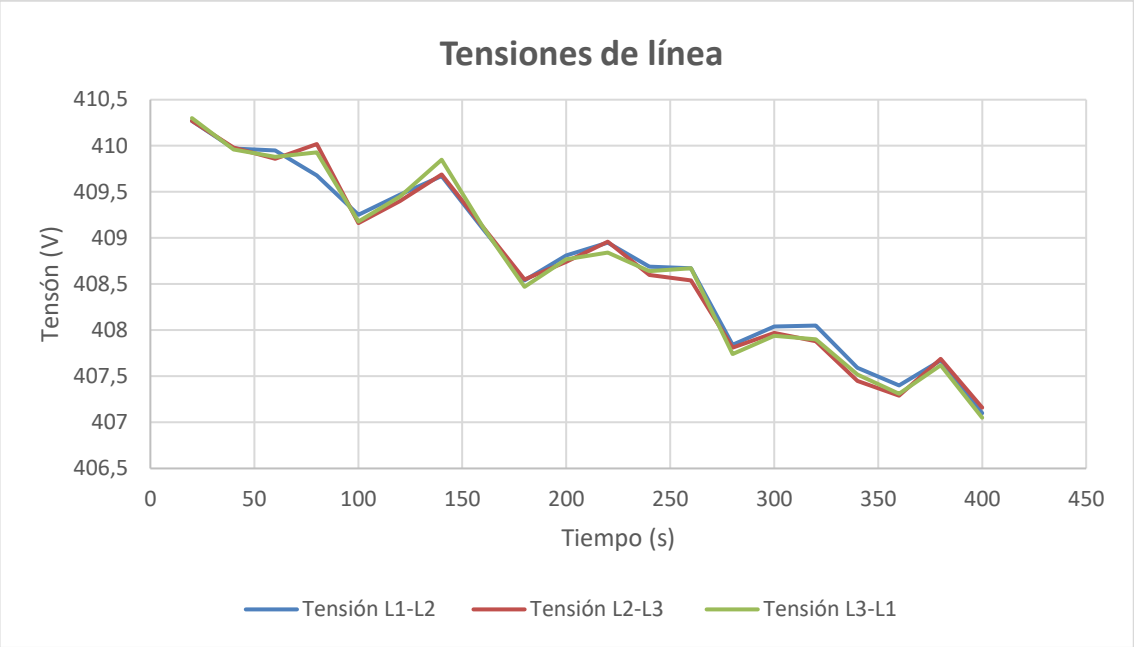
V1(V)	V2(V)	V3(V)
410,27	410,27	410,3
409,97	409,98	409,96
409,95	409,86	409,88
409,68	410,02	409,93
409,25	409,16	409,18
409,47	409,4	409,45
409,67	409,69	409,85
409,1	409,12	409,12
408,54	408,55	408,47
408,81	408,74	408,77
408,95	408,96	408,84
408,69	408,6	408,64
408,67	408,54	408,67
407,84	407,81	407,74
408,04	407,97	407,94
408,05	407,88	407,9
407,59	407,45	407,52
407,4	407,29	407,31
407,67	407,69	407,62
407,1	407,16	407,05

T(s)	Par
20	0,1Nm
40	
60	
80	
100	
120	
140	0,3 Nm
160	
180	
200	
220	
240	
260	0,5Nm
280	
300	
320	
340	
360	
380	
400	


Tensión L1-L2
410,5
410
409,5
409
408,5
408
407,5
407
406,5
TENSIÓN (V)
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)


Tensión L2-L3
410,5
410
409,5
409
408,5
408
407,5
407
TENSIÓN (V)
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)


Tensión L3-L1
410,5
410
409,5
409
408,5
408
407,5
407
406,5
TENSIÓN (V)
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)

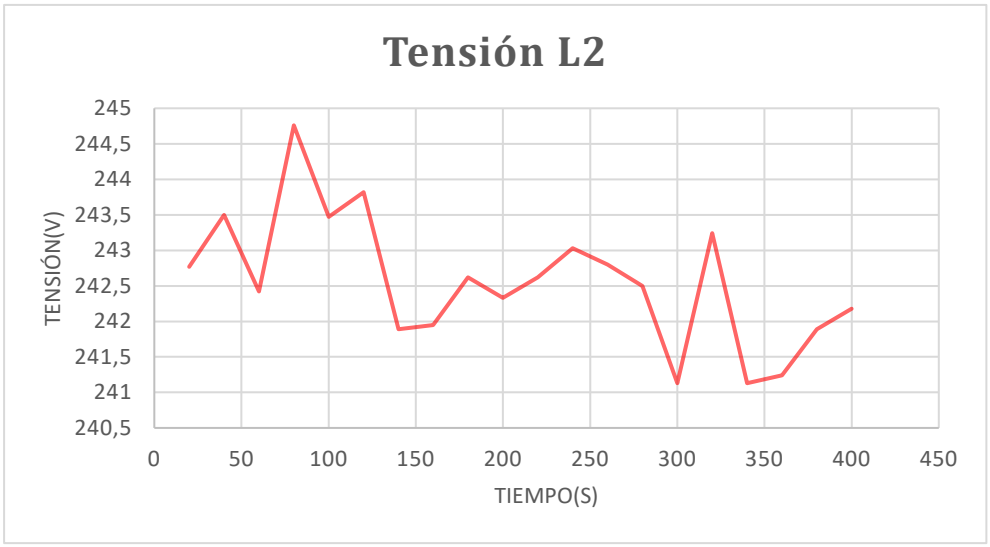


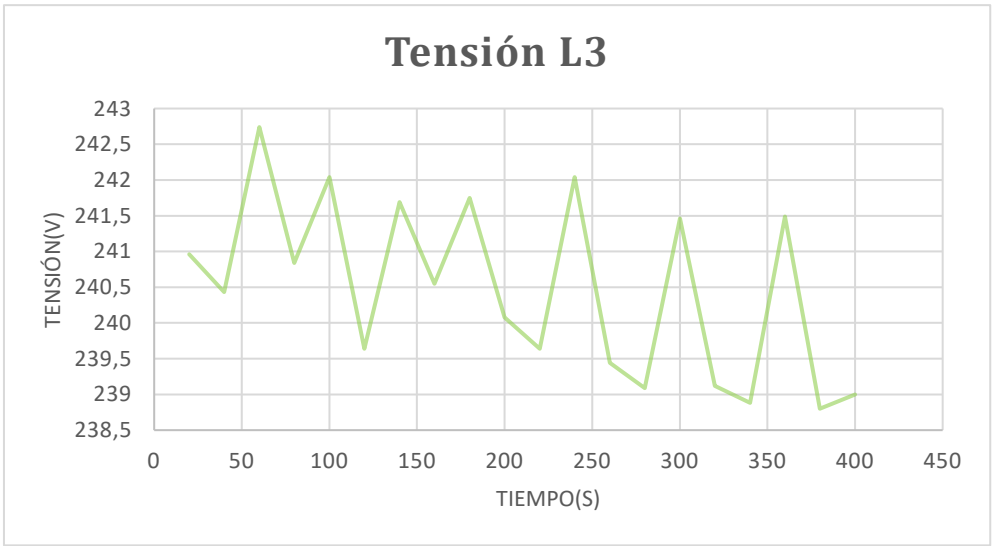
7.1.2 Tensión de fase

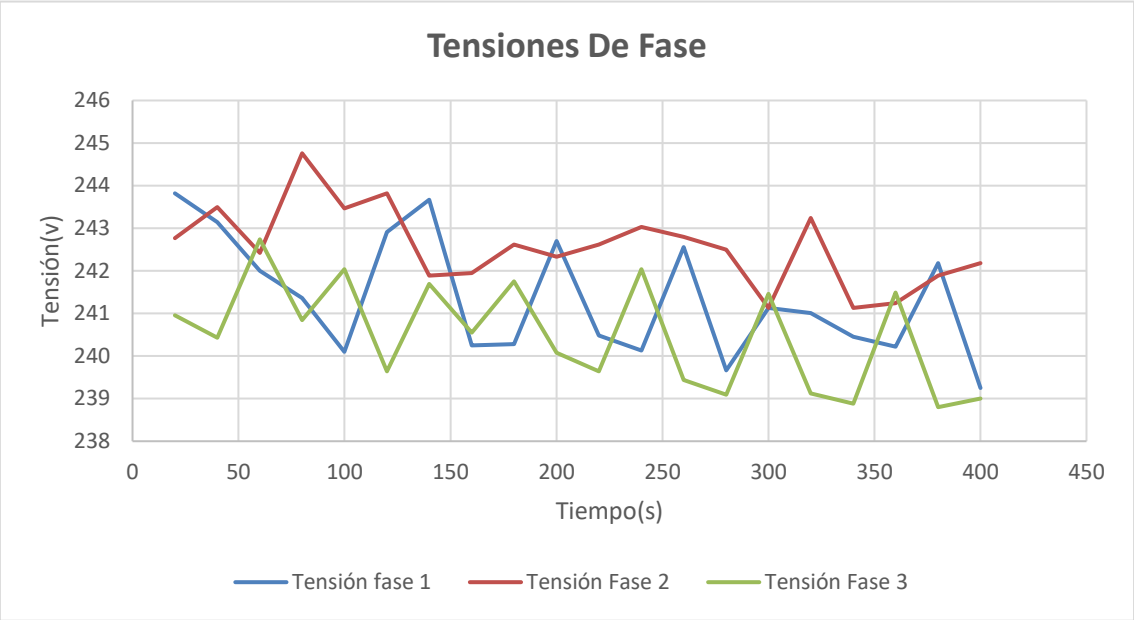
v1(V)	v2(V)	v3(V)	T(S)
243,82	242,77	240,96	20
243,14	243,5	240,43	40
242	242,42	242,74	60
241,36	244,76	240,84	80
240,1	243,47	242,04	100
242,91	243,82	239,64	120
243,67	241,89	241,69	140
240,25	241,95	240,55	160
240,28	242,62	241,75	180
242,7	242,33	240,08	200
240,48	242,62	239,64	220
240,13	243,03	242,04	240
242,56	242,8	239,44	260
239,66	242,5	239,09	280
241,13	241,13	241,46	300
241,01	243,24	239,12	320
240,45	241,13	238,88	340
240,22	241,24	241,49	360
242,18	241,89	238,8	380
239,25	242,18	239	400

T(s)	Par
20	0,1Nm
40	
60	
80	
100	
120	
140	0,3 Nm
160	
180	
200	
220	
240	
260	0,5Nm
280	
300	
320	
340	
360	
380	
400	


Tensión L1
TENSIÓN(V)
245
244
243
242
241
240
239
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)


Tensión L2
TENSIÓN(V)
245
244,5
244
243,5
243
242,5
242
241,5
241
240,5
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)

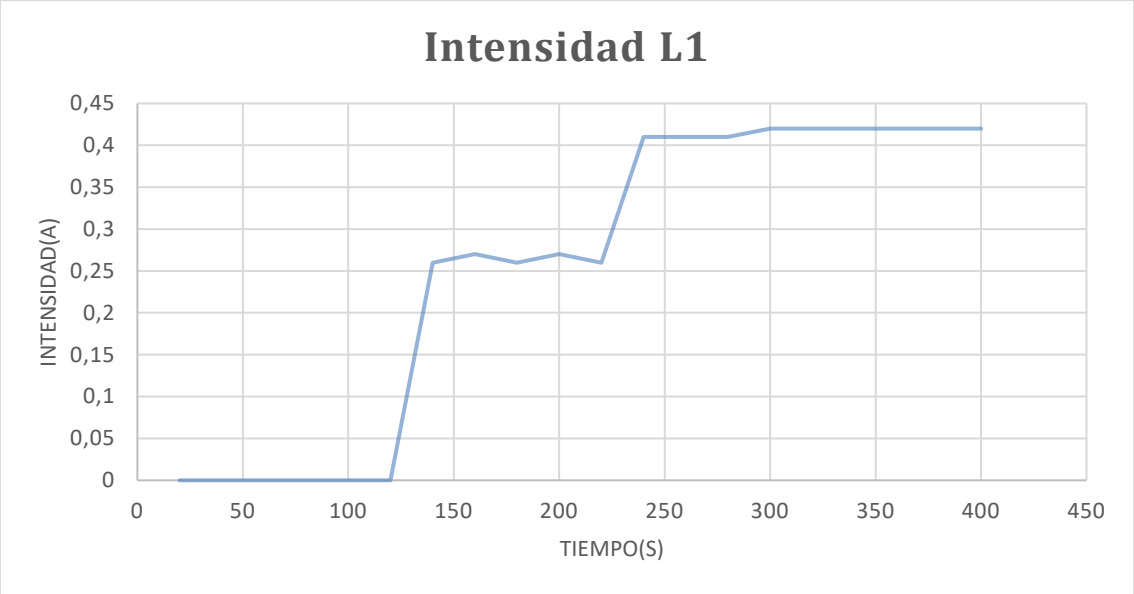

Tensión L3
TENSIÓN(V)
243
242,5
242
241,5
241
240,5
240
239,5
239
238,5
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)

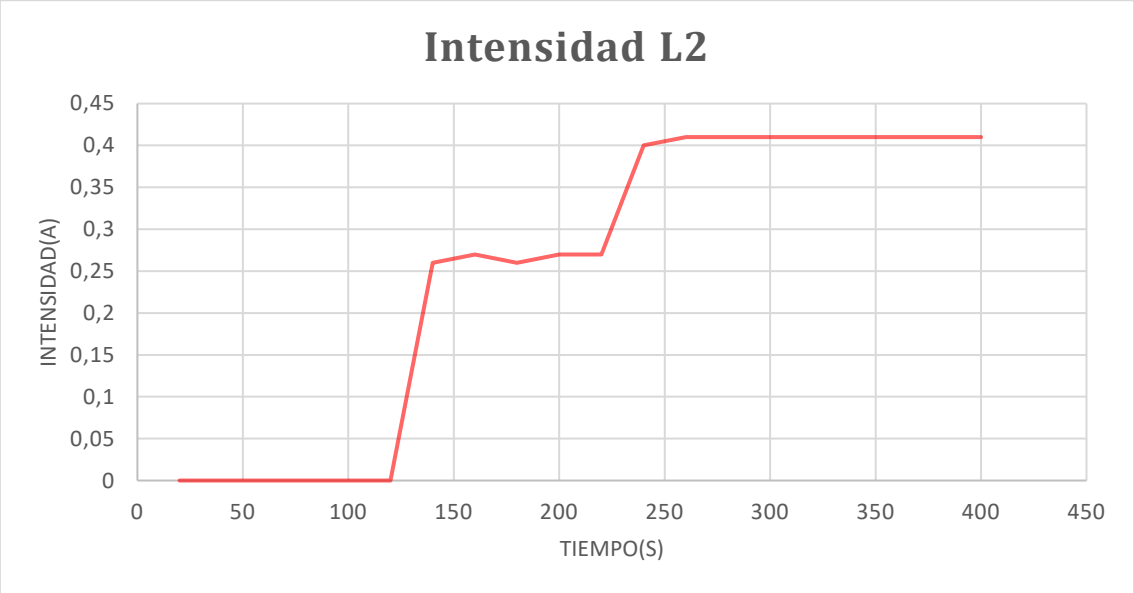


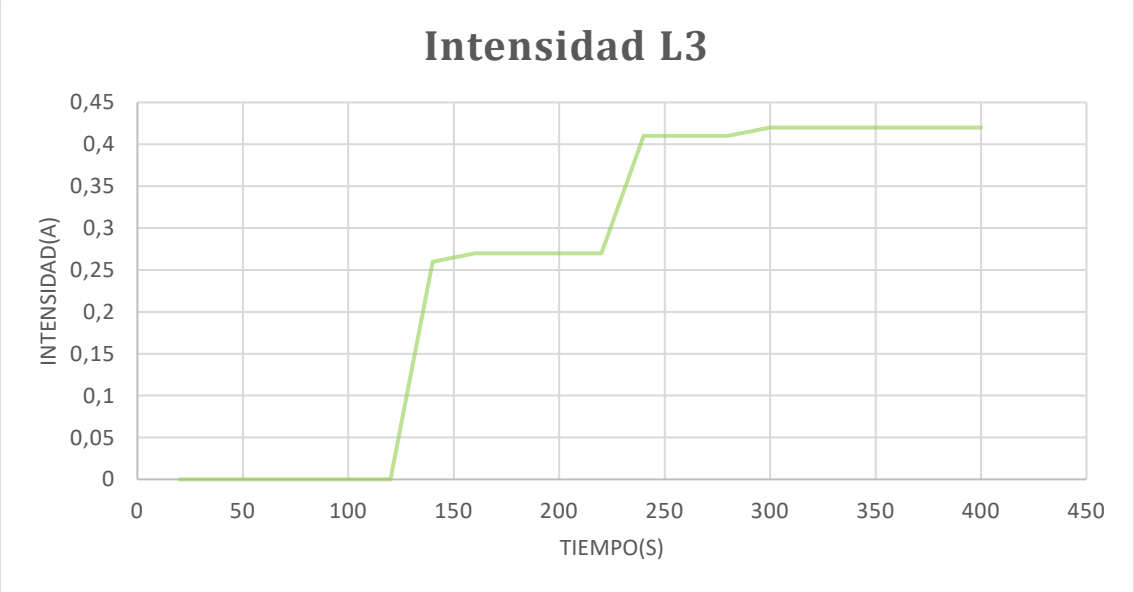
7.1.3 Intensidad de fase

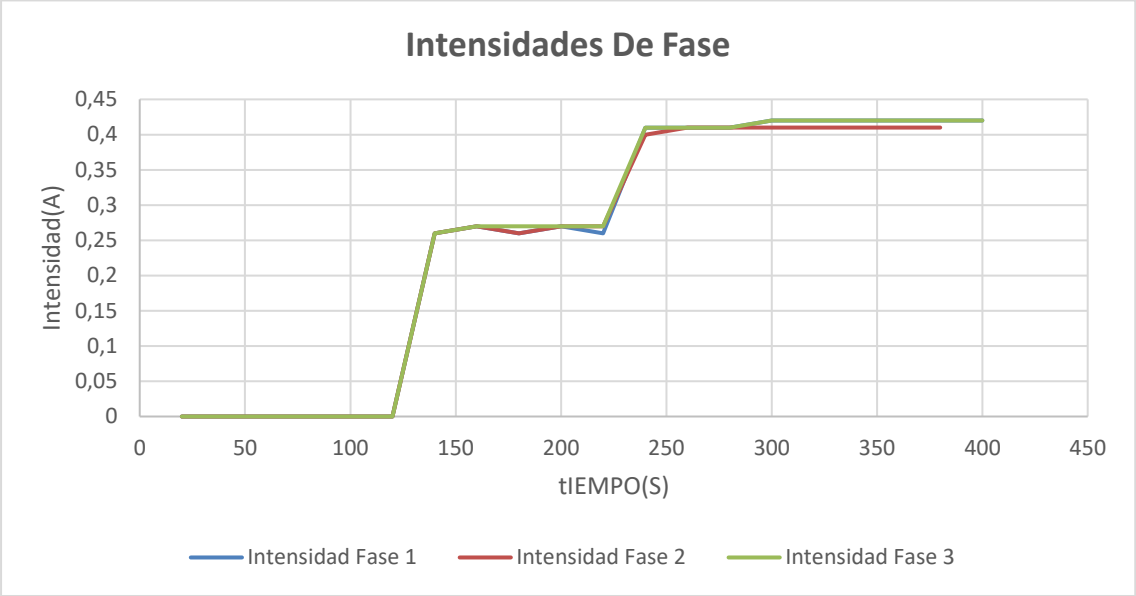
I1(A)	I2(A)	I3(A)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0,26	0,26	0,26
0,27	0,27	0,27
0,26	0,26	0,27
0,27	0,27	0,27
0,26	0,27	0,27
0,41	0,4	0,41
0,41	0,41	0,41
0,41	0,41	0,41
0,42	0,41	0,42
0,42	0,41	0,42
0,42	0,41	0,42
0,42	0,41	0,42
0,42	0,41	0,42
0,42	0,41	0,42

T(s)	Par
20	0,1Nm
40	
60	
80	
100	
120	
140	0,3 Nm
160	
180	
200	
220	
240	
260	0,5Nm
280	
300	
320	
340	
360	
380	
400	


Intensidad L1
INTENSIDAD(A)
0,45
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)


Intensidad L2
INTENSIDAD(A)
0,45
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)


Intensidad L3
INTENSIDAD(A)
0,45
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)

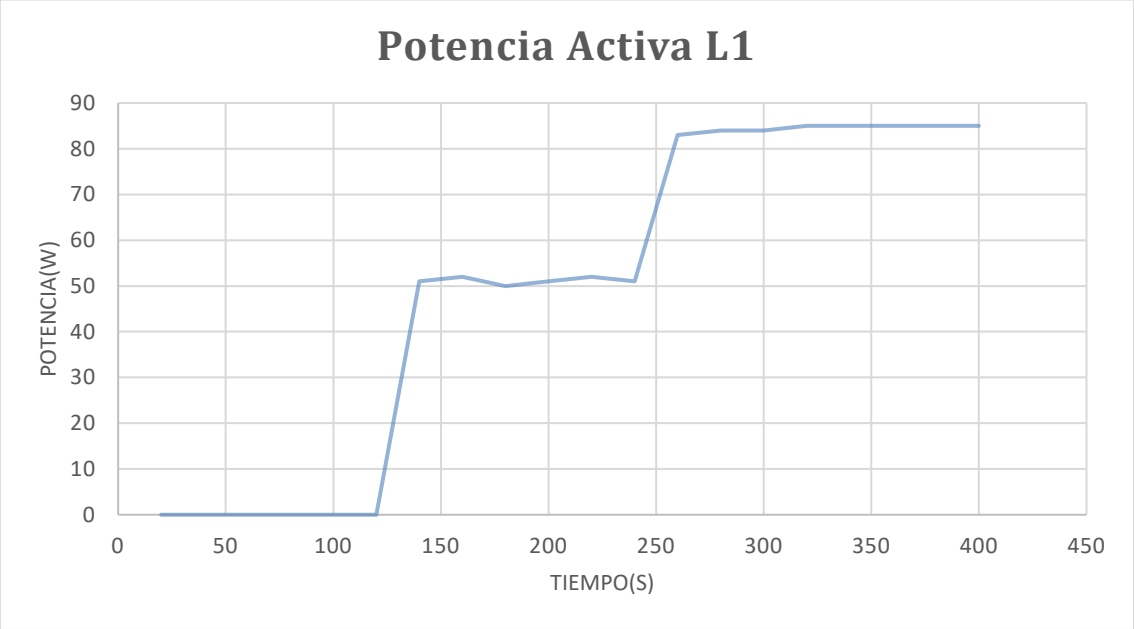


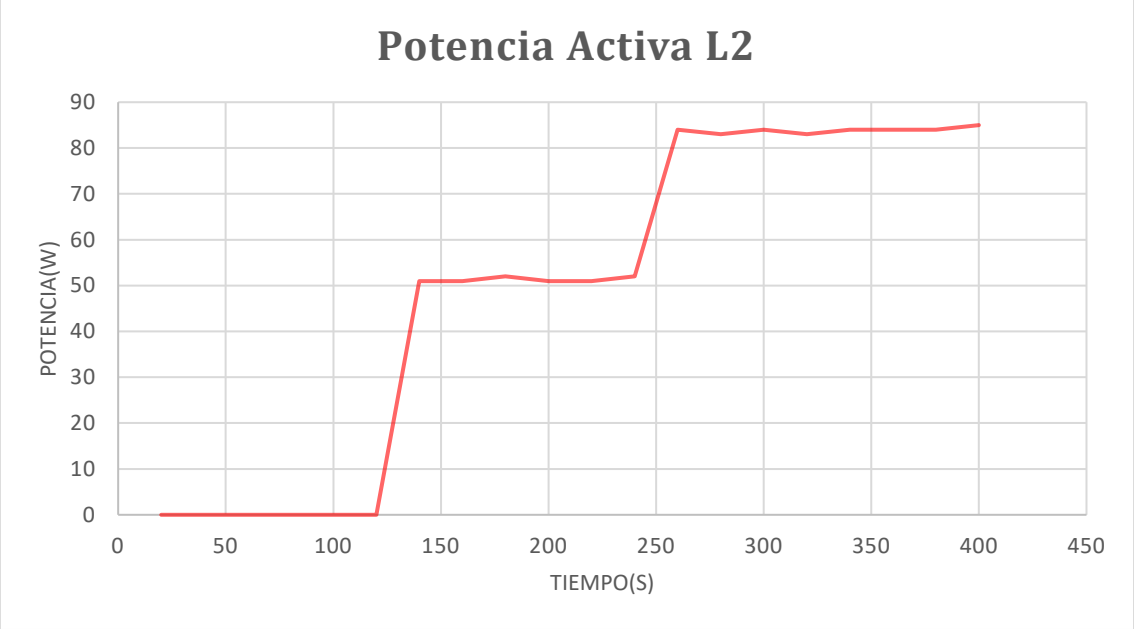
7.1.4 Potencias activas

P1(W)	P2(W)	P3(W)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
51	51	53
52	51	53
50	52	52
51	51	53
52	51	52
51	52	52
83	84	84
84	83	84
84	84	85
85	83	86
85	84	85
85	84	85
85	84	85
85	85	87

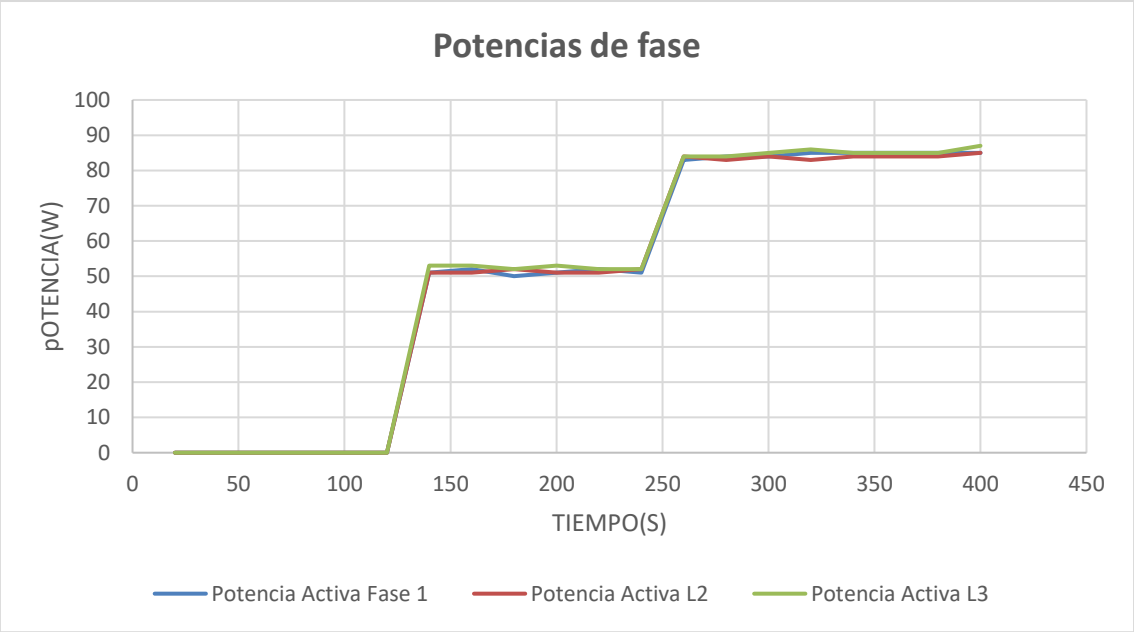
T(s)
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400

Par
0,1Nm
0,3 Nm
0,5Nm


Potencia Activa L1
POTENCIA(W)
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)


Potencia Activa L2
POTENCIA(W)
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)

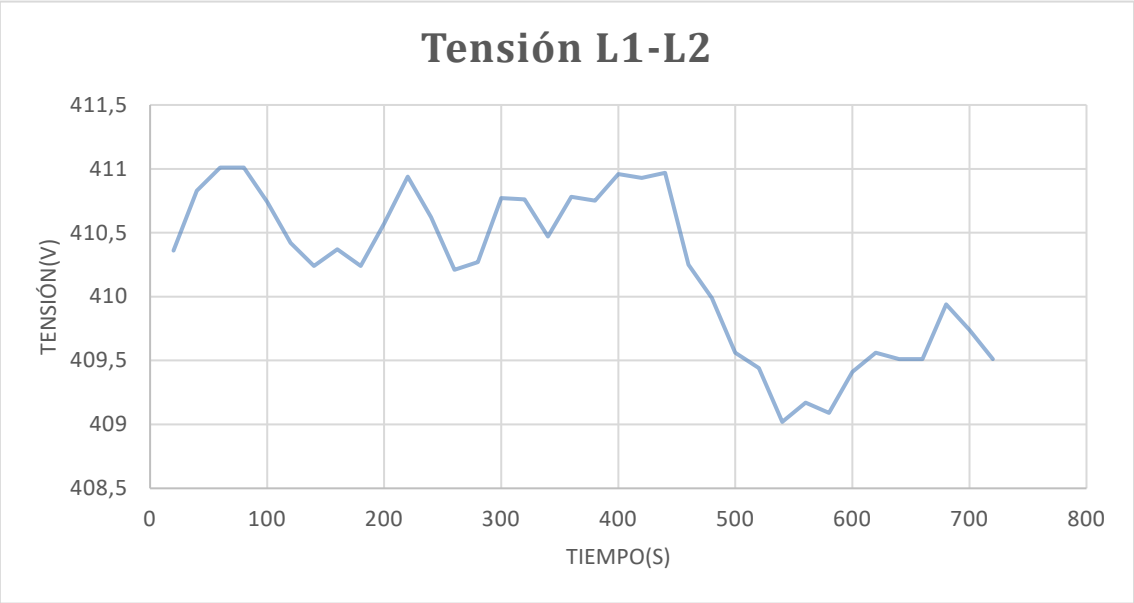

Potencia Activa L3
POTENCIA(W)
100
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)

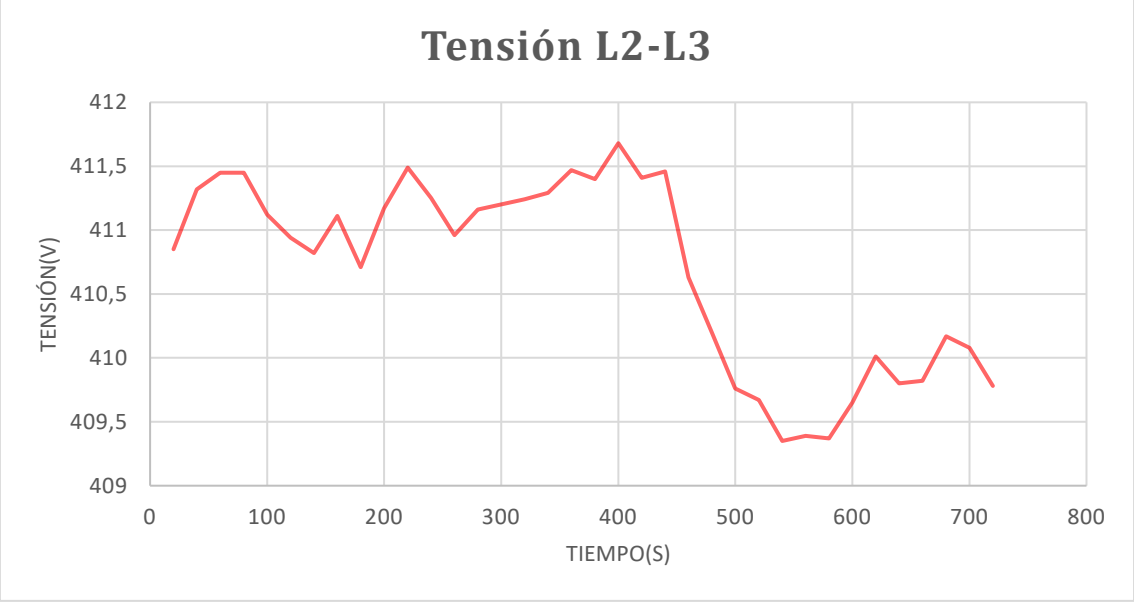

Potencias de fase
pOTENCIA(W)
100
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
350
400
450
TIEMPO(S)
Potencia Activa Fase 1
Potencia Activa L2
Potencia Activa L3

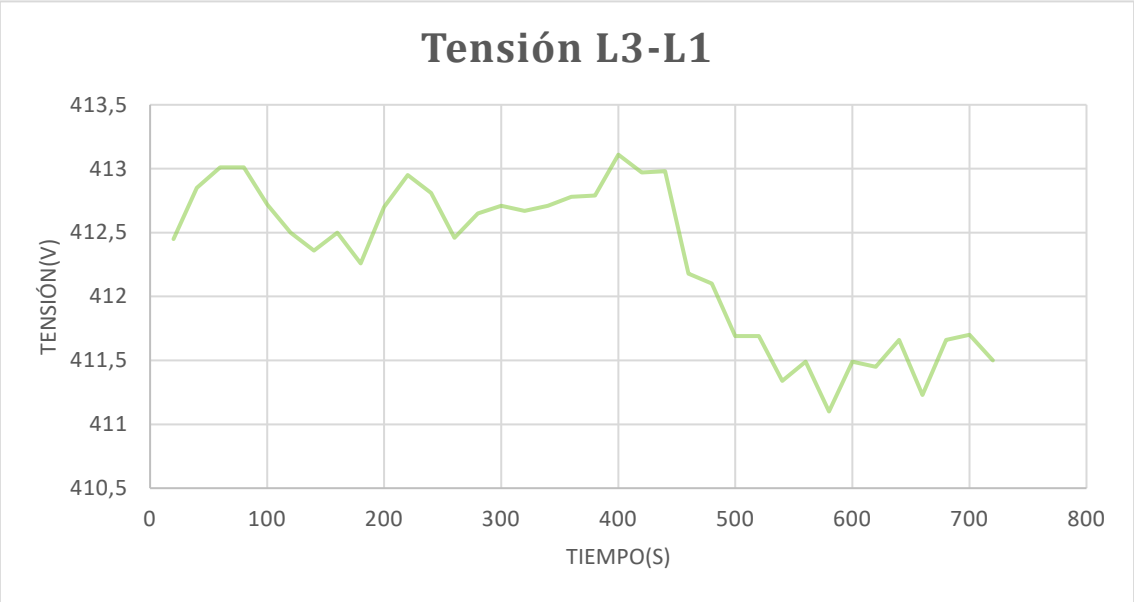
7.2 50 Hz

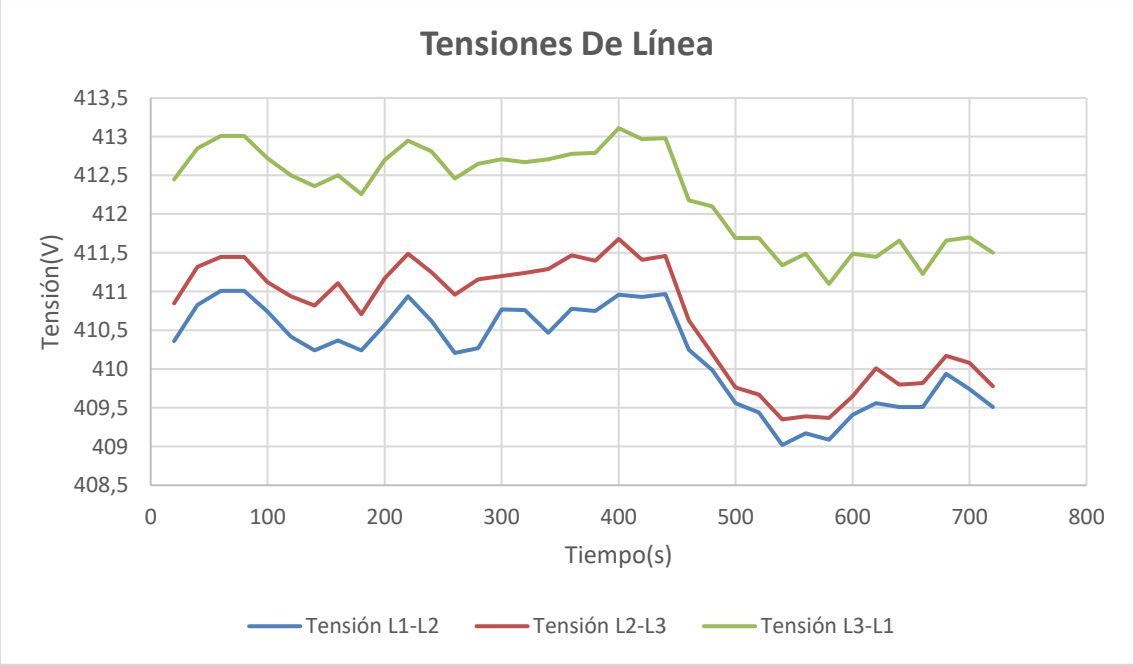
7.2.1 Tensiones de línea

V1(V)	V2(V)	V3(V)	T(s)		Par
410,36	410,85	412,45	20	2 minutos	0,1 Nm
410,83	411,32	412,85	40		
411,01	411,45	413,01	60		
411,01	411,45	413,01	80		
410,74	411,12	412,72	100		
410,42	410,94	412,5	120		
410,24	410,82	412,36	140	4 minutos	0,2 Nm
410,37	411,11	412,5	160		
410,24	410,71	412,26	180		
410,57	411,17	412,7	200		
410,94	411,49	412,95	220		
410,62	411,25	412,81	240		
410,21	410,96	412,46	260	6 minutos	0,3 Nm
410,27	411,16	412,65	280		
410,77	411,2	412,71	300		
410,76	411,24	412,67	320		
410,47	411,29	412,71	340		
410,78	411,47	412,78	360		
410,75	411,4	412,79	380	8 minutos	0,4 Nm
410,96	411,68	413,11	400		
410,93	411,41	412,97	420		
410,97	411,46	412,98	440		
410,25	410,63	412,18	460		
409,99	410,2	412,1	480		
409,56	409,76	411,69	500	10 minutos	0,5 Nm
409,44	409,67	411,69	520		
409,02	409,35	411,34	540		
409,17	409,39	411,49	560		
409,09	409,37	411,1	580		
409,41	409,65	411,49	600		
409,56	410,01	411,45	620	12 minutos	0,6 Nm
409,51	409,8	411,66	640		
409,51	409,82	411,23	660		
409,94	410,17	411,66	680		
409,74	410,08	411,7	700		
409,51	409,78	411,5	720		


Tensión L1-L2
411,5
411
410,5
410
409,5
409
408,5
TENSIÓN(V)
0
100
200
300
400
500
600
700
800
TIEMPO(S)


Tensión L2-L3
412
411,5
411
410,5
410
409,5
409
TENSIÓN(V)
0
100
200
300
400
500
600
700
800
TIEMPO(S)


Tensión L3-L1
413,5
413
412,5
412
411,5
411
410,5
TENSIÓN(V)
0
100
200
300
400
500
600
700
800
TIEMPO(S)


Tensiones De Línea
413,5
413
412,5
412
411,5
411
410,5
410
409,5
409
408,5
Tensión(V)
0
100
200
300
400
500
600
700
800
Tiempo(s)
Tensión L1-L2
Tensión L2-L3
Tensión L3-L1

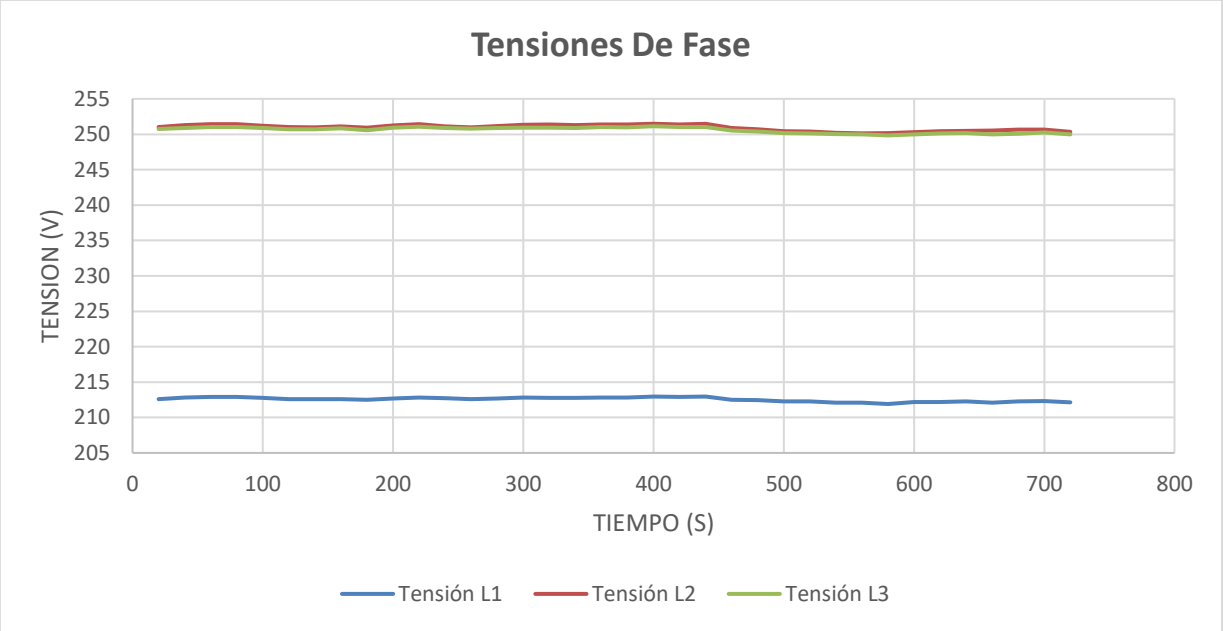
7.2.2 Tensiones de fase

v1(V)	v2(V)	v3(V)	T(s)		Par
212,61	251,04	250,75	20	2 minutos	0,1 Nm
212,81	251,27	250,9	40		
212,9	251,42	251,04	60		
212,9	251,42	251,04	80		
212,78	251,22	250,87	100		
212,61	251,04	250,69	120		
212,58	250,98	250,69	140	4 minutos	0,2 Nm
212,58	251,13	250,84	160		
212,49	250,92	250,58	180		
212,67	251,24	250,93	200		
212,84	251,42	251,07	220		
212,72	251,13	250,9	240		
212,58	250,98	250,78	260	6 minutos	0,3 Nm
212,67	251,16	250,87	280		
212,84	251,33	250,93	300		
212,78	251,36	250,93	320		
212,78	251,3	250,87	340		
212,84	251,39	251,01	360		
212,81	251,36	250,98	380	8 minutos	0,4 Nm
212,96	251,48	251,13	400		
212,93	251,36	251,01	420		
212,96	251,45	251,01	440		
212,52	250,89	250,52	460		
212,46	250,69	250,4	480		
212,29	250,45	250,14	500	10 minutos	0,5 Nm
212,26	250,39	250,11	520		
212,11	250,19	250,02	540		
212,11	250,13	249,96	560		
211,91	250,16	249,84	580		
212,17	250,31	249,99	600		
212,2	250,45	250,11	620	12 minutos	0,6 Nm
212,29	250,48	250,14	640		
212,11	250,51	249,96	660		
212,26	250,66	250,08	680		
212,31	250,66	250,25	700		
212,14	250,36	249,99	720		


Tensión L1
213,2
213
212,8
212,6
212,4
212,2
212
211,8
TENSION (V)
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Tensión L2
251,6
251,4
251,2
251
250,8
250,6
250,4
250,2
250
TENSION (V)
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Tensión L3
251,2
251
250,8
250,6
250,4
250,2
250
249,8
249,6
TENSION (V)
0
100
200
300
400
500
600
700
800
TIEMPO (S)

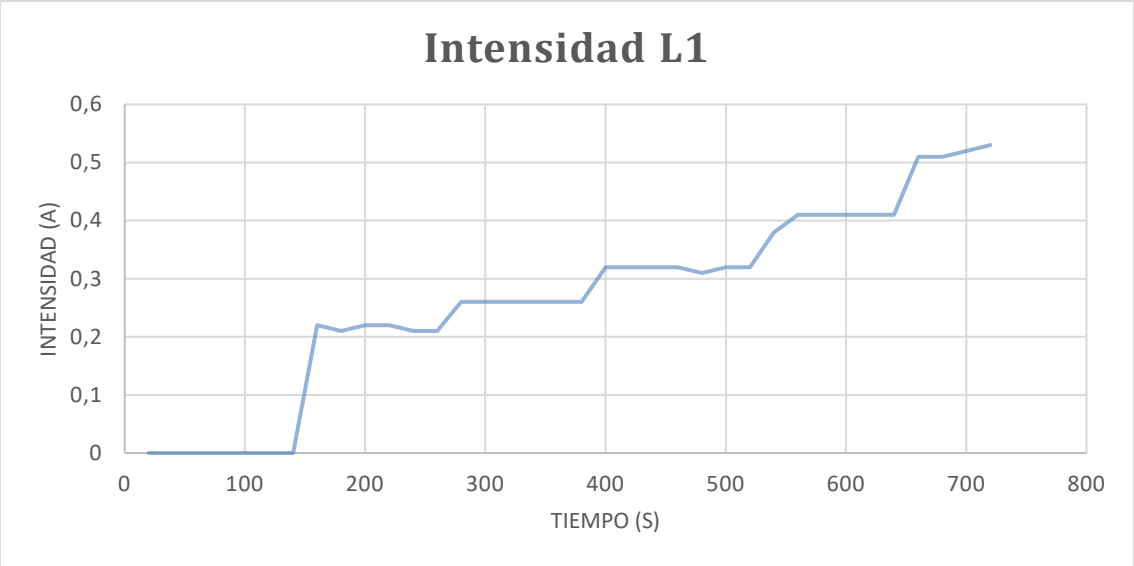

Tensiones De Fase
255
250
245
240
235
230
225
220
215
210
205
TENSION (V)
0
100
200
300
400
500
600
700
800
TIEMPO (S)
Tensión L1
Tensión L2
Tensión L3

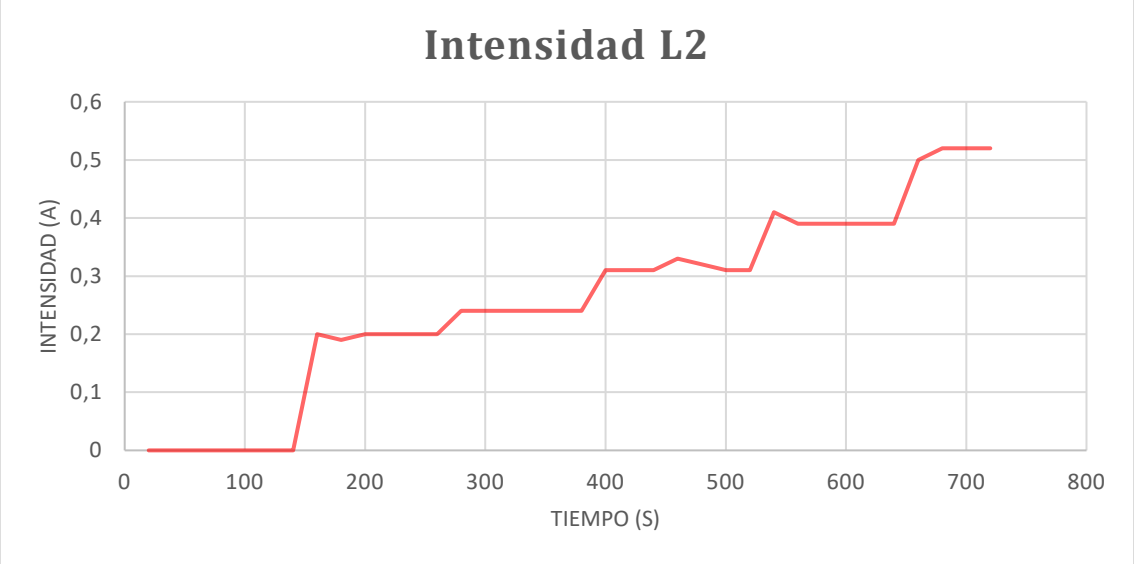
7.2.3 Intensidades de fase

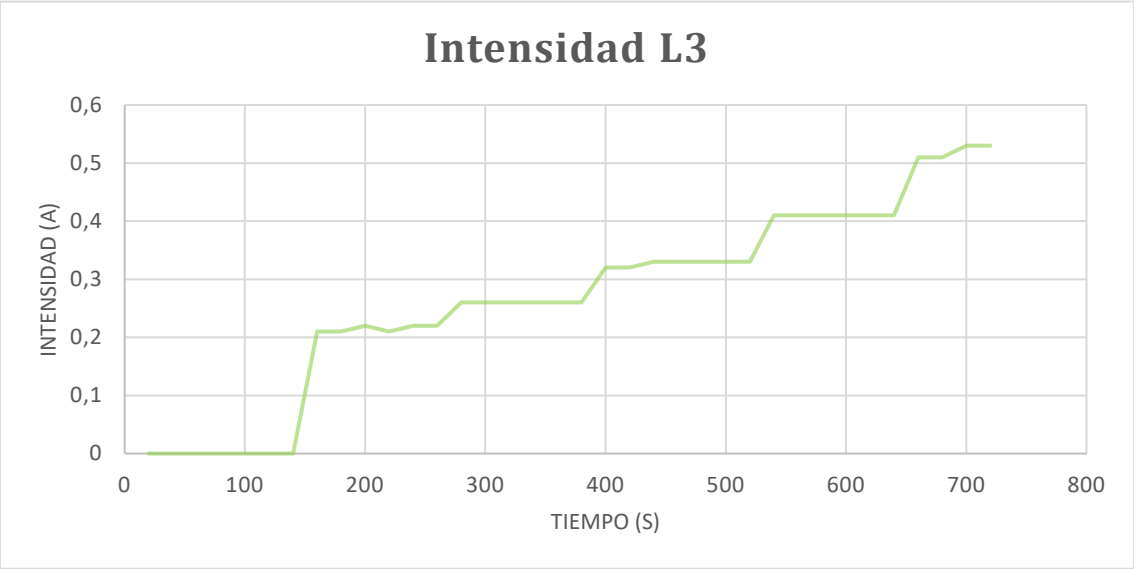
I1(A)	I2(A)	I3(A)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0,22	0,2	0,21
0,21	0,19	0,21
0,22	0,2	0,22
0,22	0,2	0,21
0,21	0,2	0,22
0,21	0,2	0,22
0,26	0,24	0,26
0,26	0,24	0,26
0,26	0,24	0,26
0,26	0,24	0,26
0,26	0,24	0,26
0,26	0,24	0,26
0,32	0,31	0,32
0,32	0,31	0,32
0,32	0,31	0,33
0,32	0,33	0,33
0,31	0,32	0,33
0,32	0,31	0,33
0,32	0,31	0,33
0,38	0,41	0,41
0,41	0,39	0,41
0,41	0,39	0,41
0,41	0,39	0,41
0,41	0,39	0,41
0,41	0,39	0,41
0,51	0,5	0,51
0,51	0,52	0,51
0,52	0,52	0,53
0,53	0,52	0,53

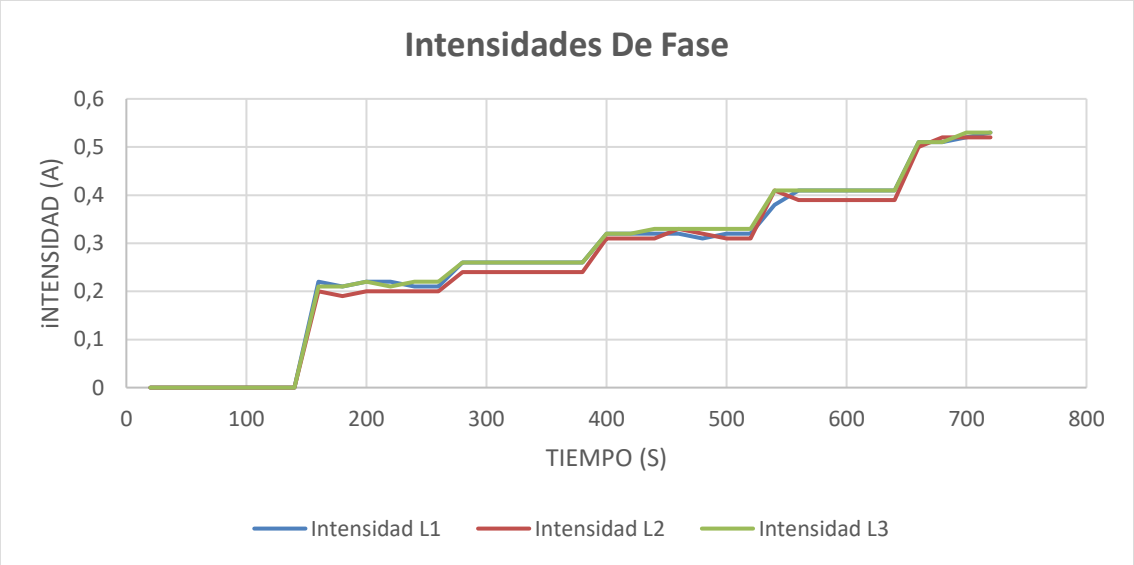
T(s)	
20	2 minutos
40	
60	
80	
100	
120	
140	4 minutos
160	
180	
200	
220	
240	
260	6 minutos
280	
300	
320	
340	
360	
380	8 minutos
400	
420	
440	
460	
480	
500	10 minutos
520	
540	
560	
580	
600	
620	12 minutos
640	
660	
680	
700	
720	

Par
0,1 Nm
0,2 Nm
0,3 Nm
0,4 Nm
0,5 Nm
0,6 Nm


Intensidad L1
INTENSIDAD (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Intensidad L2
INTENSIDAD (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Intensidad L3
INTENSIDAD (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Intensidades De Fase
iNTENSIDAD (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)
Intensidad L1
Intensidad L2
Intensidad L3

7.2.4 Potencias activas

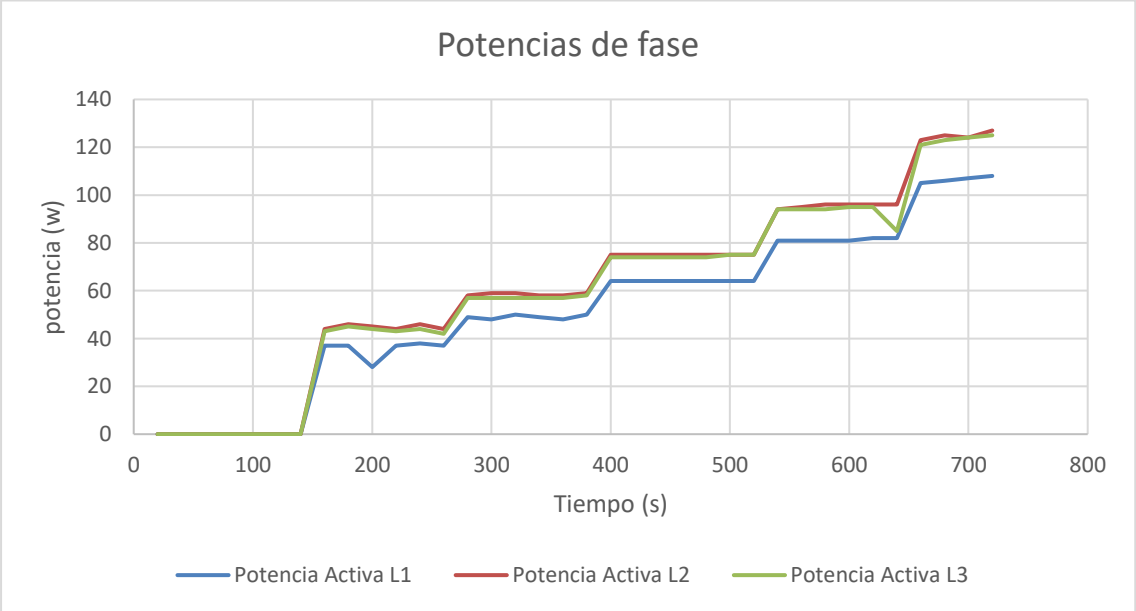
P1(W)	P2(W)	P3(W)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
37	44	43
37	46	45
28	45	44
37	44	43
38	46	44
37	44	42
49	58	57
48	59	57
50	59	57
49	58	57
48	58	57
50	59	58
64	75	74
64	75	74
64	75	74
64	75	74
64	75	74
64	75	75
64	75	75
81	94	94
81	95	94
81	96	94
81	96	95
82	96	95
82	96	85
105	123	121
106	125	123
107	124	124
108	127	125

T(s)		Par
20	2 minutos	0,1 Nm
40		
60		
80		
100		
120		
140	4 minutos	0,2 Nm
160		
180		
200		
220		
240		
260	6 minutos	0,3 Nm
280		
300		
320		
340		
360		
380	8 minutos	0,4 Nm
400		
420		
440		
460		
480		
500	10 minutos	0,5 Nm
520		
540		
560		
580		
600		
620	12 minutos	0,6 Nm
640		
660		
680		
700		
720		


Potencia Activa L1
POTENCIA (W)
120
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Potencia Activa L2
POTENCIA (W)
140
120
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)


Potencia Activa L3
POTENCIA (W)
140
120
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800
TIEMPO (S)

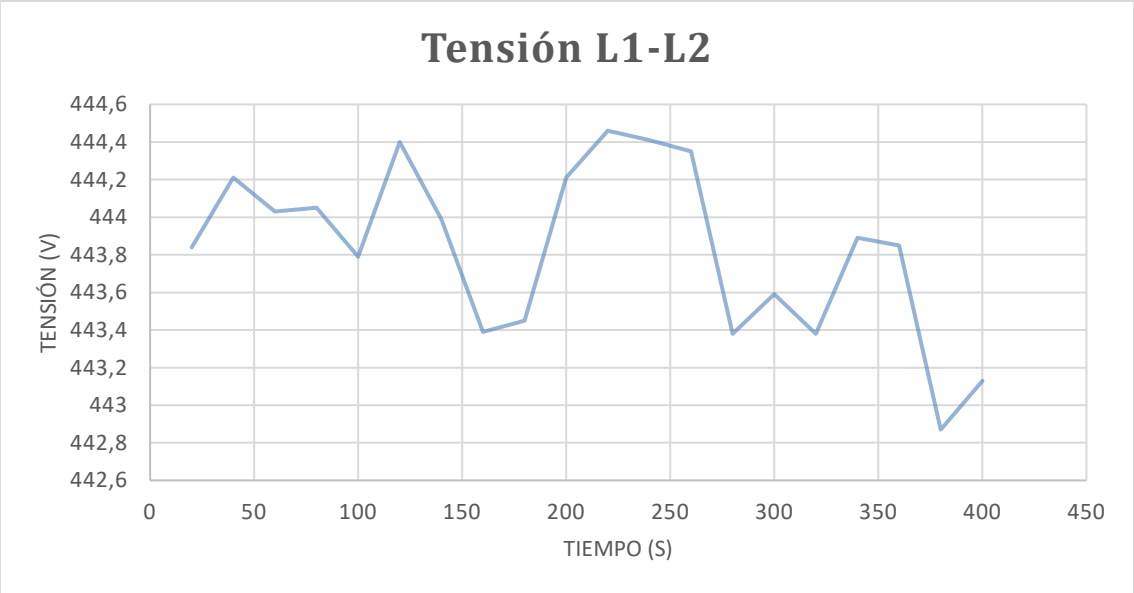

Potencias de fase
potencia (w)
140
120
100
80
60
40
20
0
0
100
200
300
400
500
600
700
800
Tiempo (s)
Potencia Activa L1
Potencia Activa L2
Potencia Activa L3

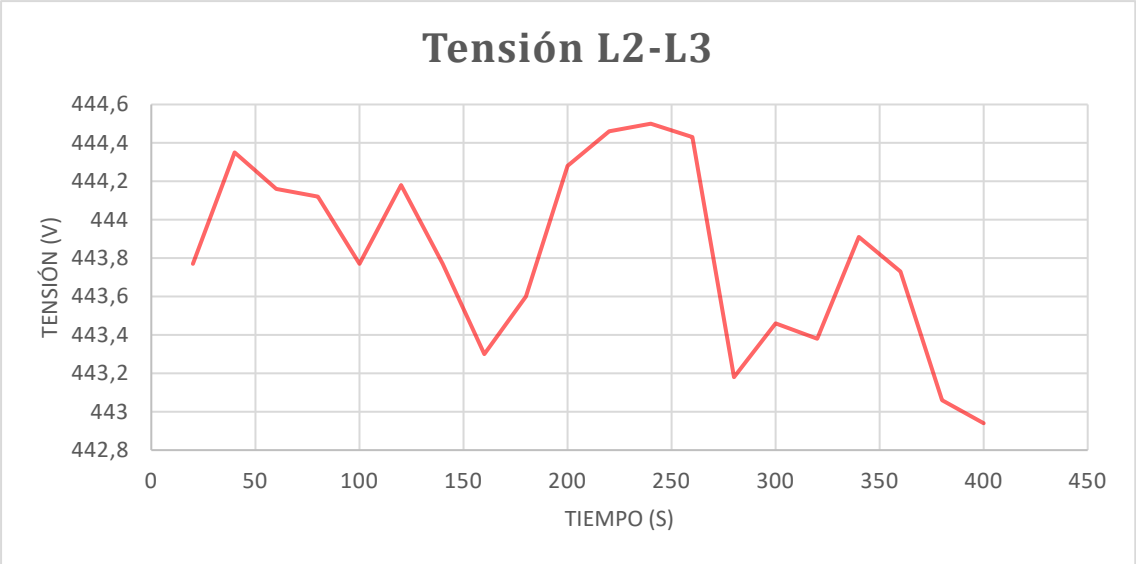
7.3 55 Hz

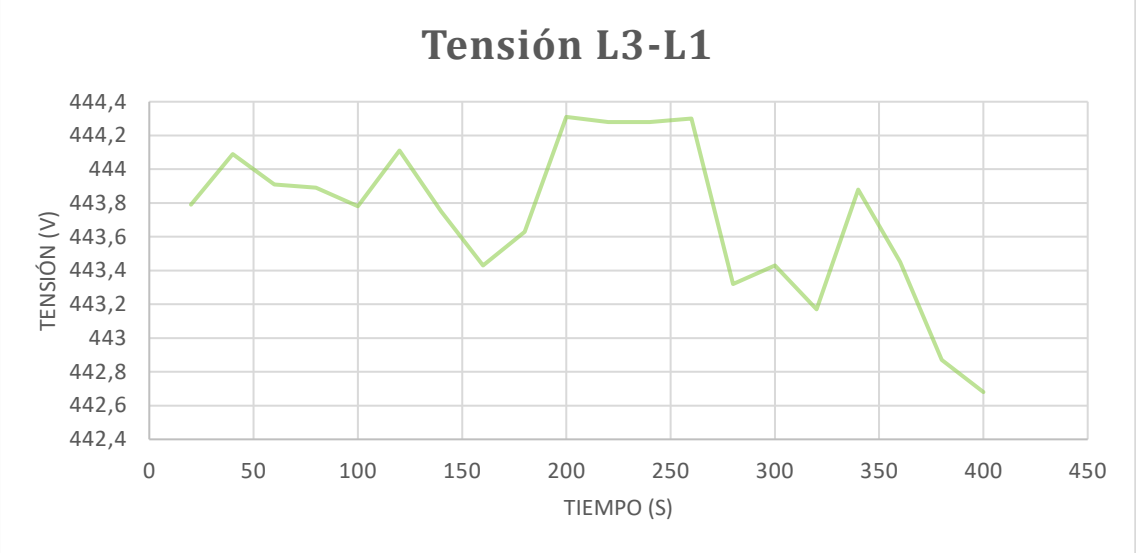
7.3.1 Tensiones de línea

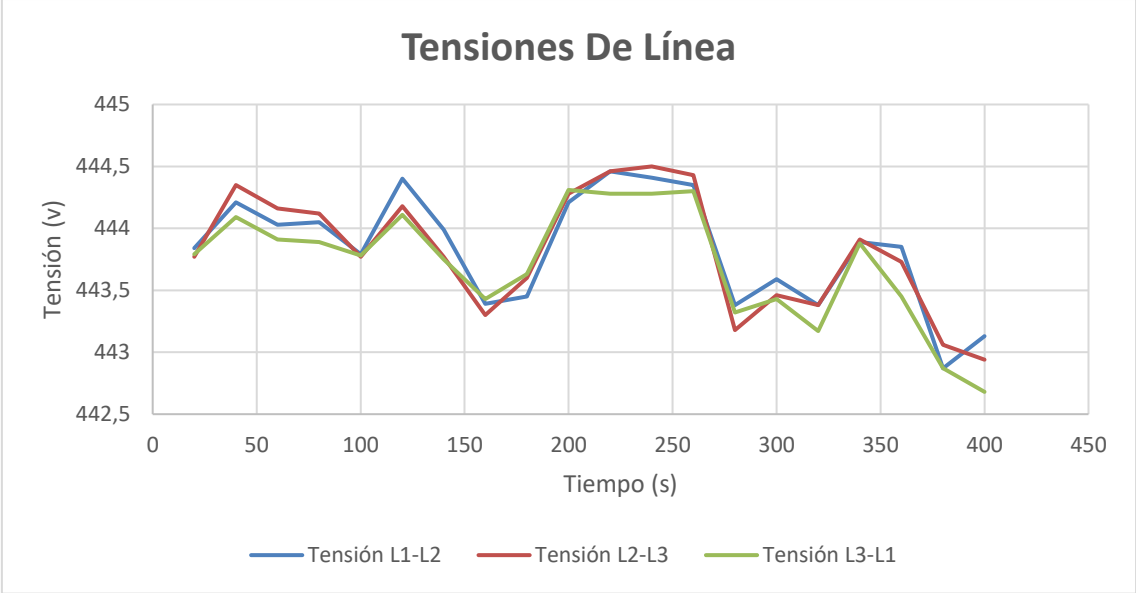
V1(V)	V2(V)	V3(V)
443,84	443,77	443,79
444,21	444,35	444,09
444,03	444,16	443,91
444,05	444,12	443,89
443,79	443,77	443,78
444,4	444,18	444,11
443,99	443,77	443,75
443,39	443,3	443,43
443,45	443,6	443,63
444,21	444,28	444,31
444,46	444,46	444,28
444,41	444,5	444,28
444,35	444,43	444,3
443,38	443,18	443,32
443,59	443,46	443,43
443,38	443,38	443,17
443,89	443,91	443,88
443,85	443,73	443,45
442,87	443,06	442,87
443,13	442,94	442,68

T(s)	Par(Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	0,5
280	
300	
320	
340	
360	
380	
400	




Tensión L2-L3
444,6
444,4
444,2
444
443,8
443,6
443,4
443,2
443
442,8
TENSIÓN (V)
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)

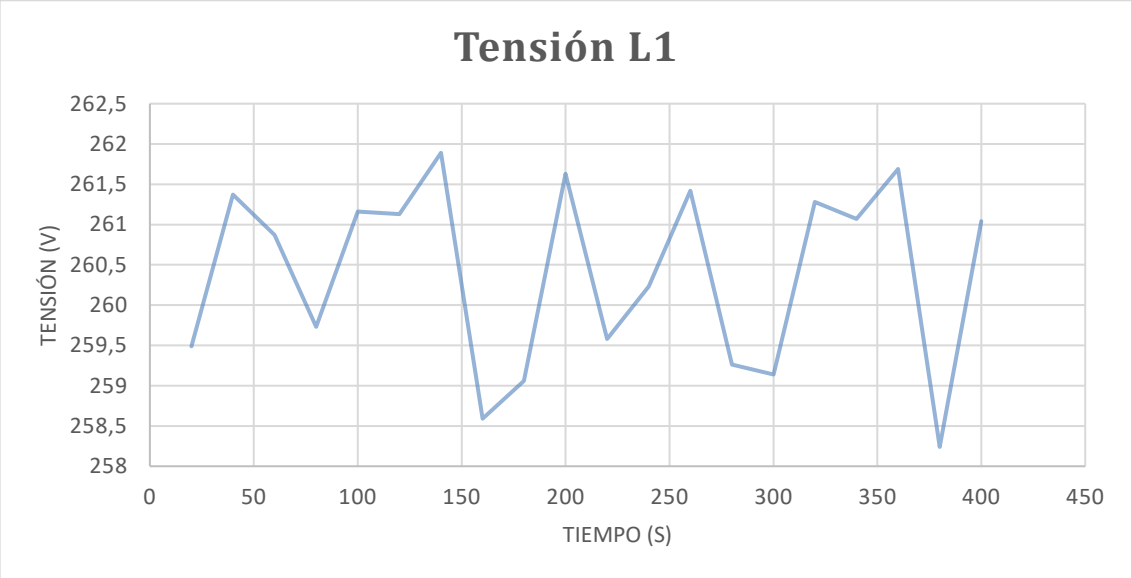

Tensión L3-L1
444,4
444,2
444
443,8
443,6
443,4
443,2
443
442,8
442,6
442,4
TENSIÓN (V)
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)


Tensiones De Línea
445
444,5
444
443,5
443
442,5
Tensión (v)
0
50
100
150
200
250
300
350
400
450
Tiempo (s)
Tensión L1-L2
Tensión L2-L3
Tensión L3-L1

7.3.2 Tensiones de fase

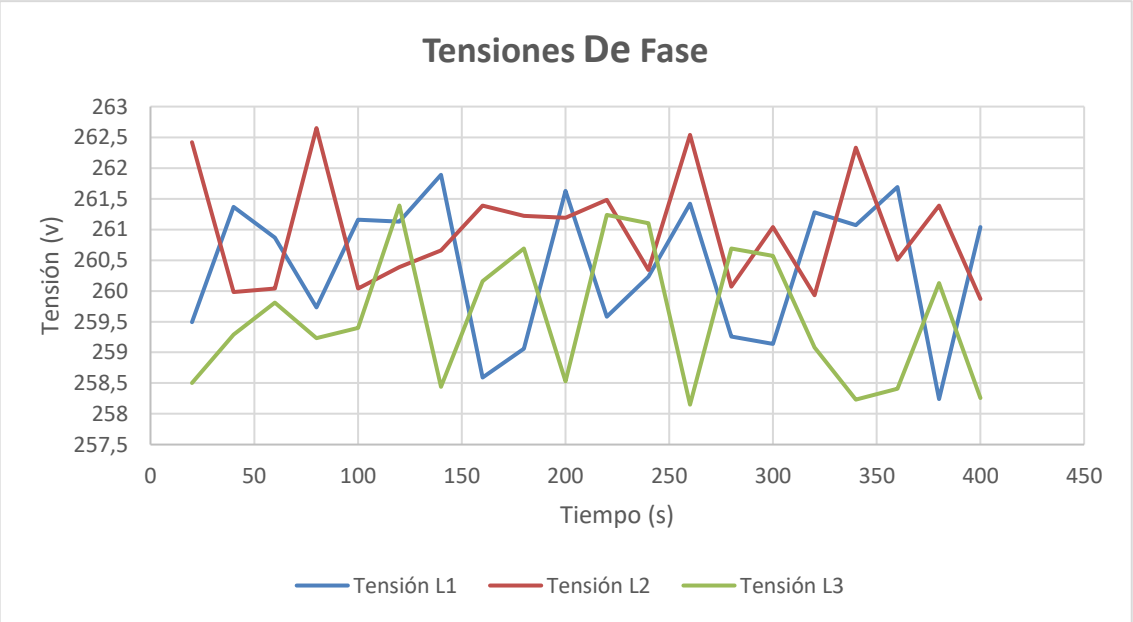
v1(V)	v2(V)	v3(V)
259,49	262,42	258,5
261,37	259,98	259,29
260,87	260,04	259,81
259,73	262,65	259,23
261,16	260,04	259,4
261,13	260,39	261,39
261,89	260,66	258,44
258,59	261,39	260,16
259,06	261,22	260,69
261,63	261,19	258,53
259,58	261,48	261,24
260,23	260,34	261,1
261,42	262,54	258,15
259,26	260,07	260,69
259,14	261,04	260,57
261,28	259,93	259,08
261,07	262,33	258,23
261,69	260,51	258,41
258,24	261,39	260,13
261,04	259,87	258,26

T(s)	Par(Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	0,5
280	
300	
320	
340	
360	
380	
400	




Tensión L2
TENSIÓN (V)
263
262,5
262
261,5
261
260,5
260
259,5
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)

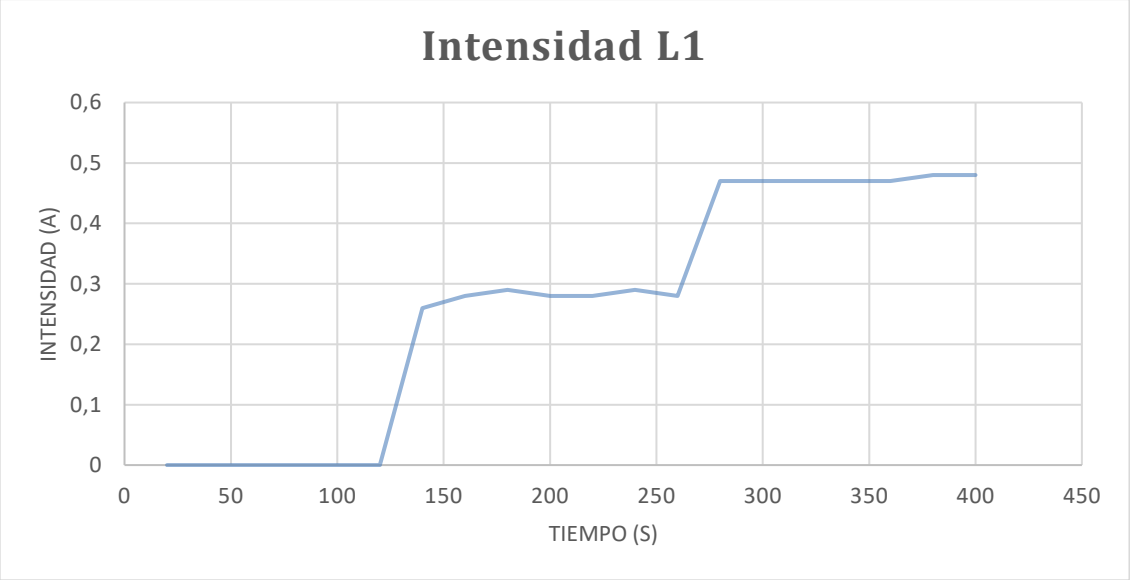

Tensión L3
TENSIÓN (V)
262
261,5
261
260,5
260
259,5
259
258,5
258
257,5
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)

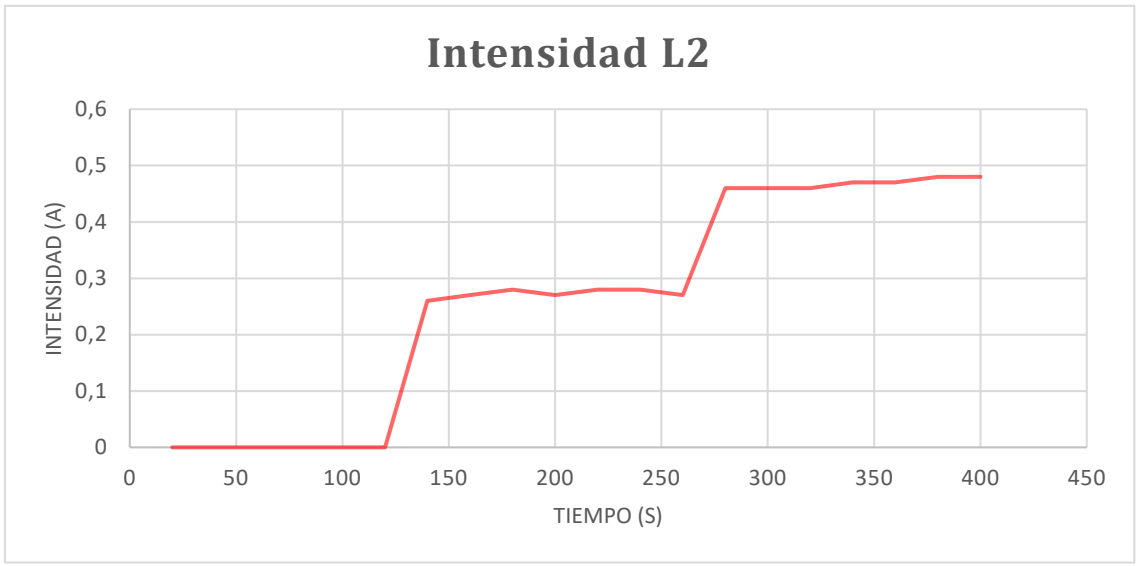

Tensiones De Fase
Tensión (v)
263
262,5
262
261,5
261
260,5
260
259,5
259
258,5
258
257,5
0
50
100
150
200
250
300
350
400
450
Tiempo (s)
Tensión L1
Tensión L2
Tensión L3

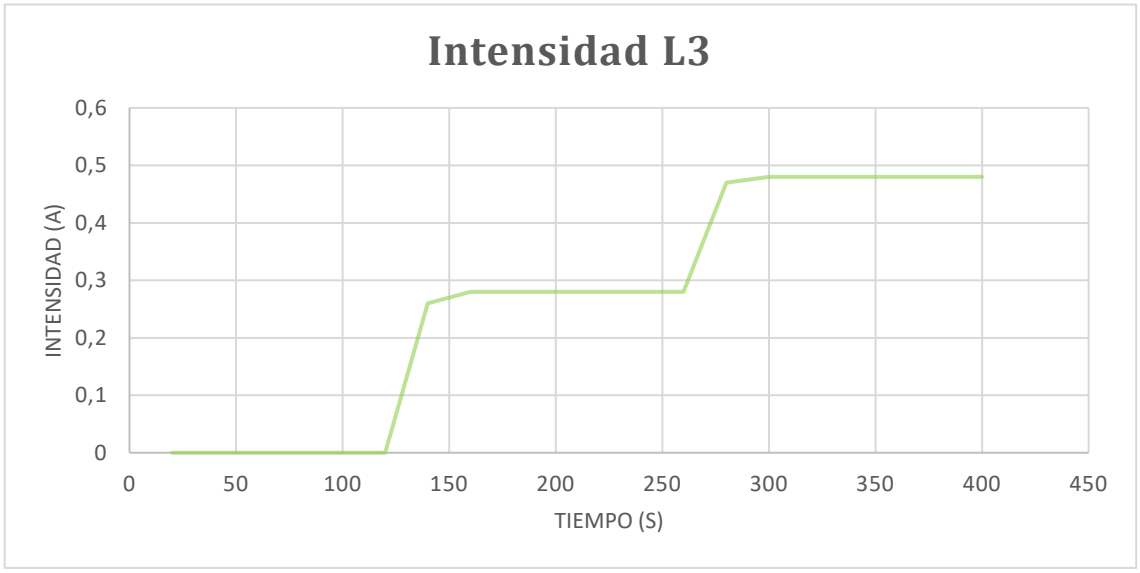
7.3.3 Intensidades de fase

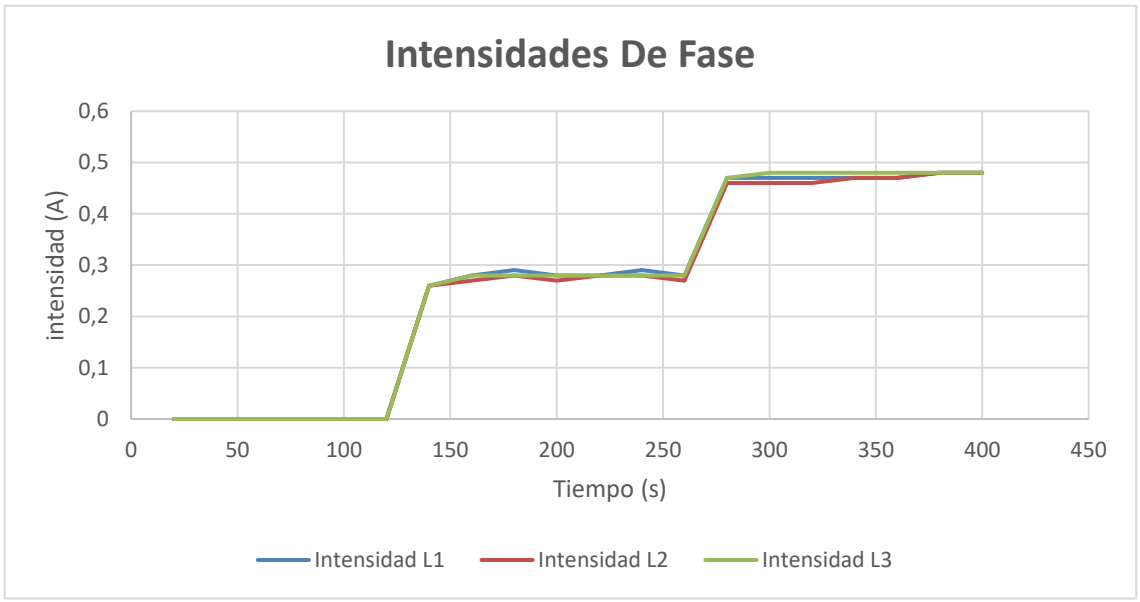
I1(A)	I2(A)	I3(A)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0,26	0,26	0,26
0,28	0,27	0,28
0,29	0,28	0,28
0,28	0,27	0,28
0,28	0,28	0,28
0,29	0,28	0,28
0,28	0,27	0,28
0,47	0,46	0,47
0,47	0,46	0,48
0,47	0,46	0,48
0,47	0,47	0,48
0,47	0,47	0,48
0,48	0,48	0,48
0,48	0,48	0,48

T(s)	Par(Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	0,5
280	
300	
320	
340	
360	
380	
400	




Intensidad L2
INTENSIDAD (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)

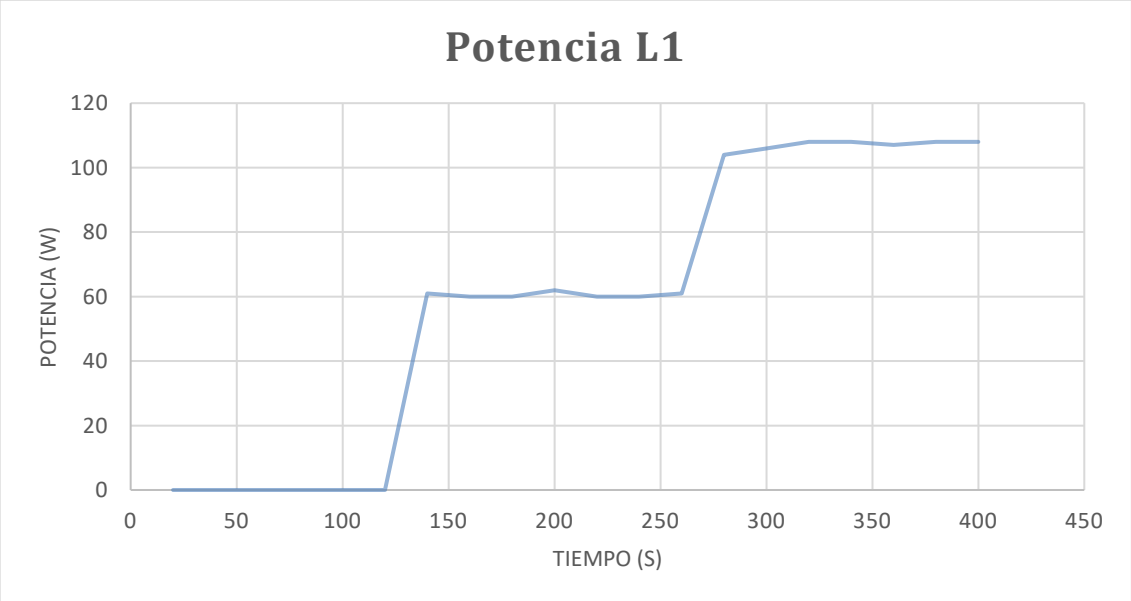

Intensidad L3
INTENSIDAD (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)


Intensidades De Fase
intensidad (A)
0,6
0,5
0,4
0,3
0,2
0,1
0
0
50
100
150
200
250
300
350
400
450
Tiempo (s)
Intensidad L1
Intensidad L2
Intensidad L3

7.3.4 Potencias activas

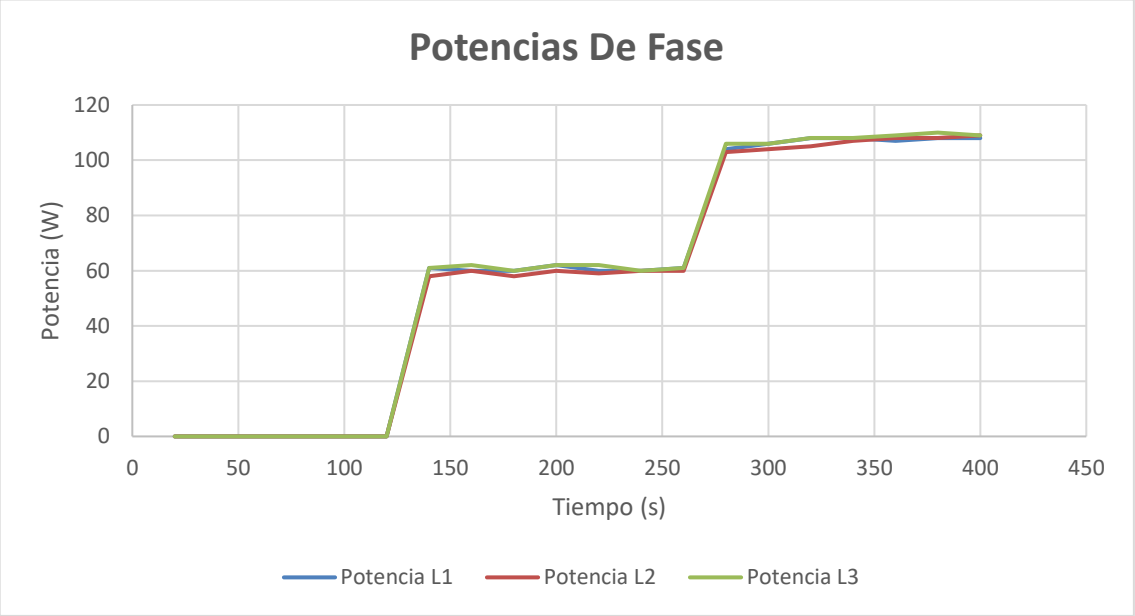
P1(W)	P2(W)	P3(W)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
61	58	61
60	60	62
60	58	60
62	60	62
60	59	62
60	60	60
61	60	61
104	103	106
106	104	106
108	105	108
108	107	108
107	108	109
108	108	110
108	109	109

T(s)	Par(Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	0,5
280	
300	
320	
340	
360	
380	
400	




Potencia L2
POTENCIA (W)
120
100
80
60
40
20
0
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)


Potencia L3
POTENCIA (W)
120
100
80
60
40
20
0
0
50
100
150
200
250
300
350
400
450
TIEMPO (S)

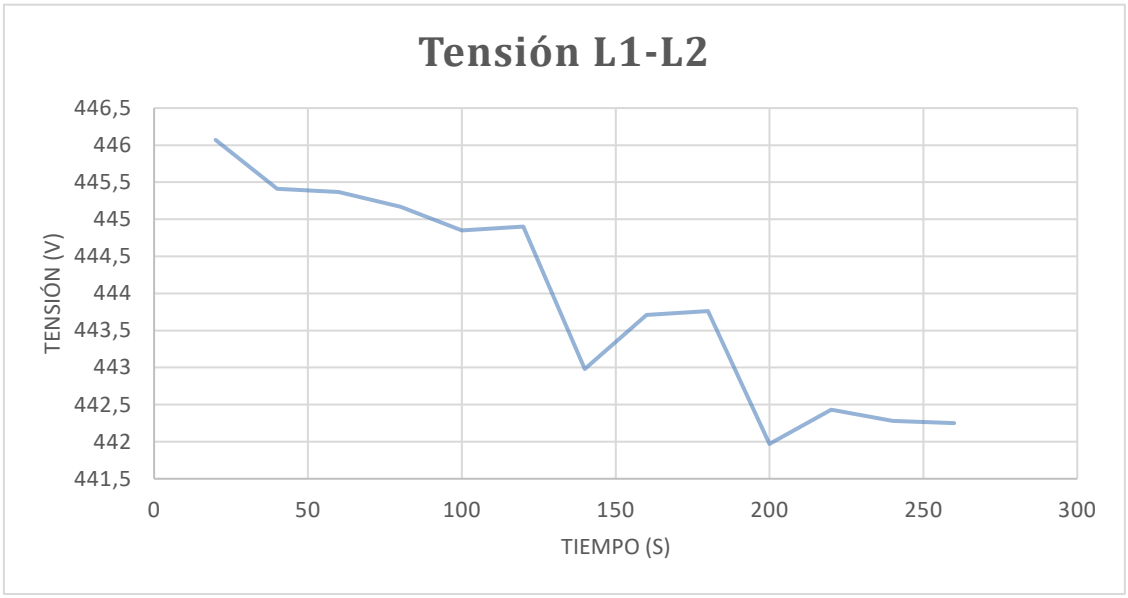

Potencias De Fase
Potencia (W)
120
100
80
60
40
20
0
0
50
100
150
200
250
300
350
400
450
Tiempo (s)
Potencia L1
Potencia L2
Potencia L3

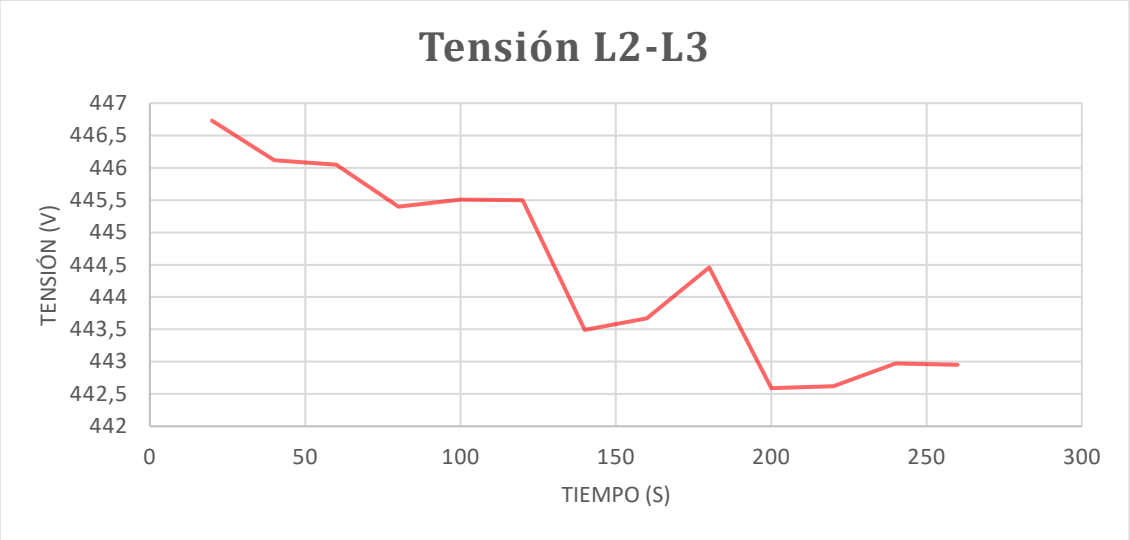
7.4 60 Hz

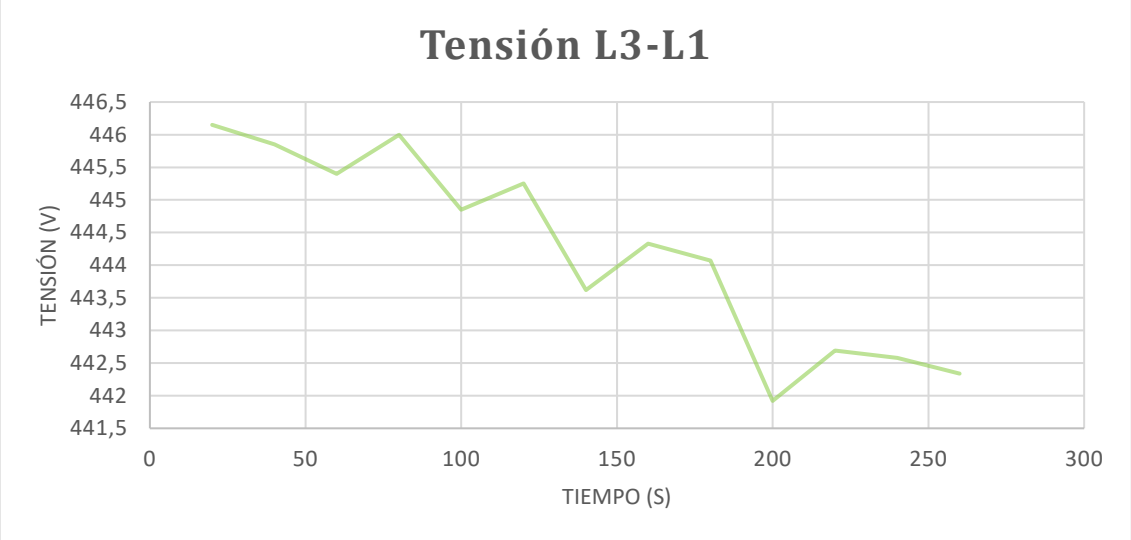
7.4.1 Tensiones de línea

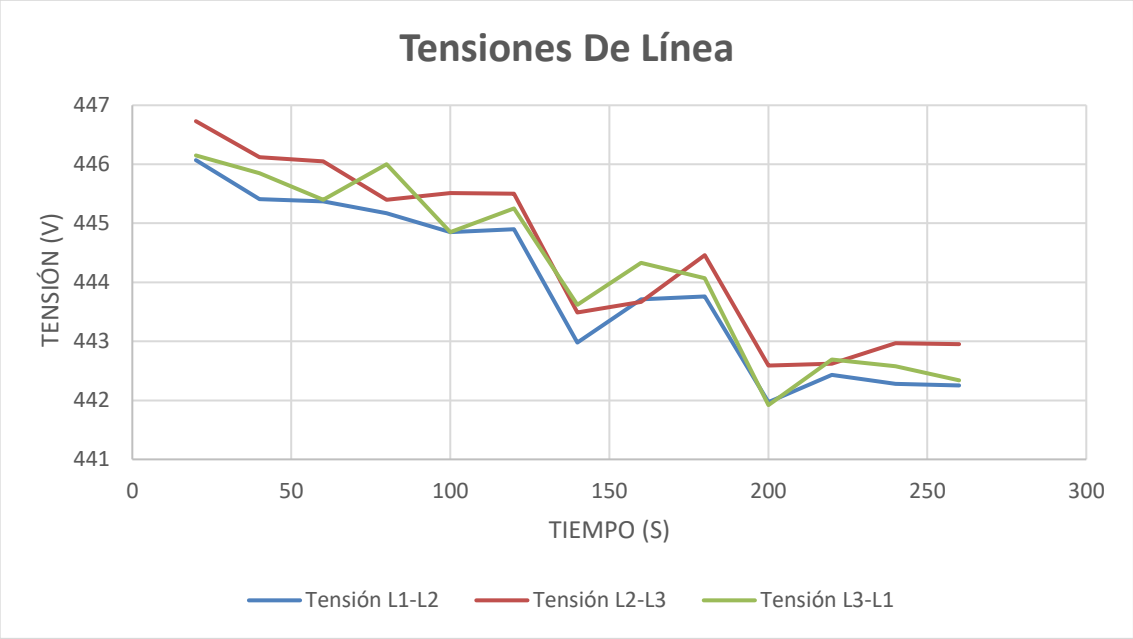
V1 (V)	V2 (V)	V3 (V)
446,07	446,73	446,15
445,41	446,12	445,85
445,37	446,05	445,4
445,17	445,4	446
444,85	445,51	444,85
444,9	445,5	445,25
442,98	443,49	443,62
443,71	443,67	444,33
443,76	444,46	444,07
441,97	442,59	441,92
442,43	442,62	442,69
442,28	442,97	442,58
442,25	442,95	442,34

T (s)	Par
20	0,1 Nm
40	
60	
80	
100	
120	
140	0,3 Nm
160	
180	
200	
220	
240	
260	



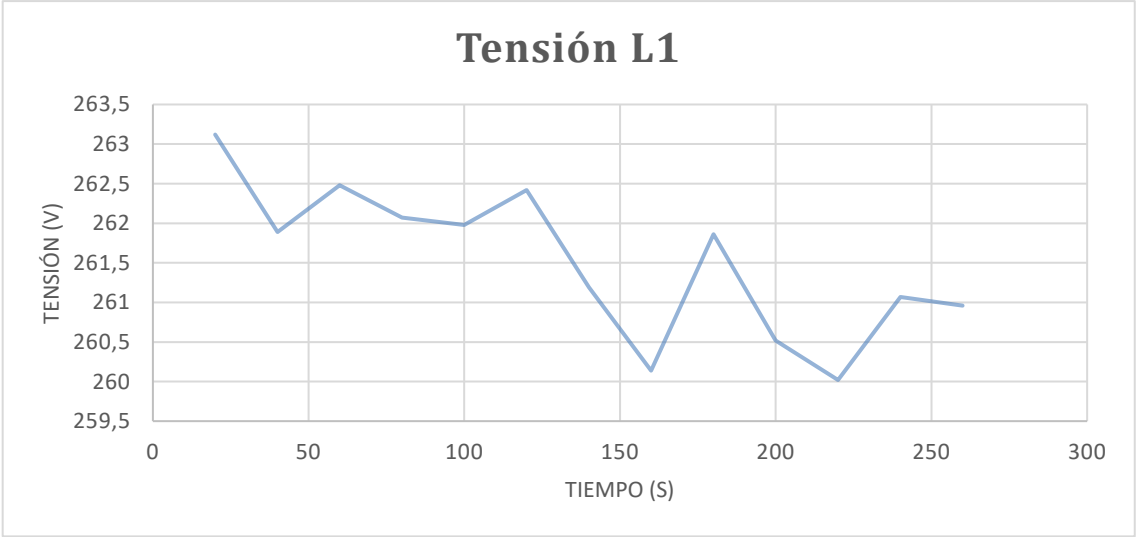

Tensión L2-L3
TENSIÓN (V)
447
446,5
446
445,5
445
444,5
444
443,5
443
442,5
442
0
50
100
150
200
250
300
TIEMPO (S)


Tensión L3-L1
TENSIÓN (V)
446,5
446
445,5
445
444,5
444
443,5
443
442,5
442
441,5
0
50
100
150
200
250
300
TIEMPO (S)

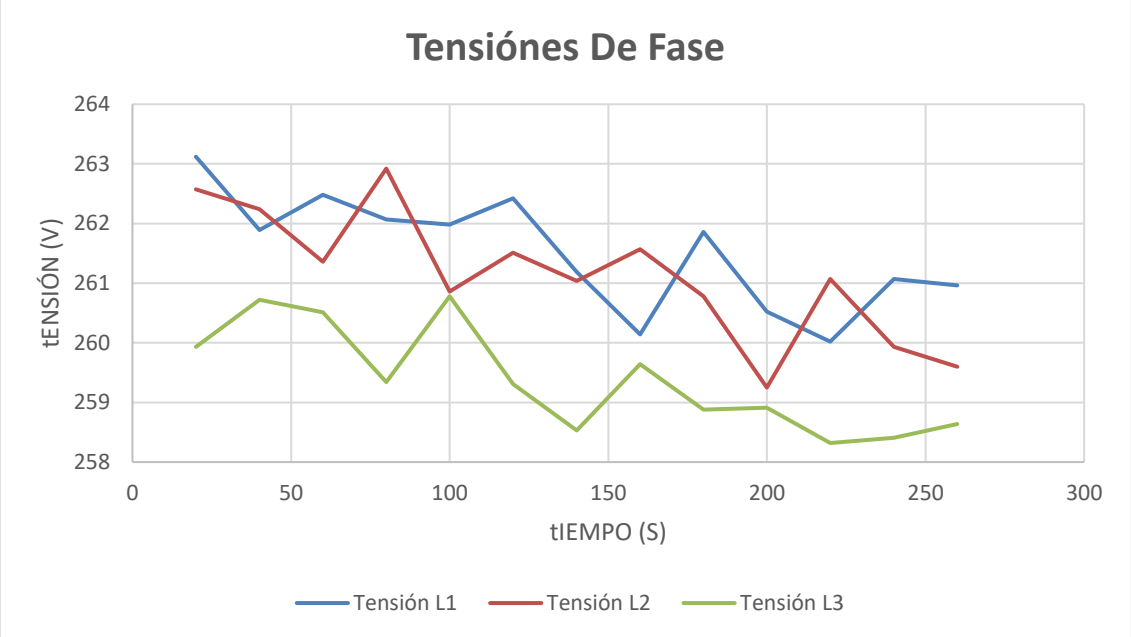

Tensiones De Línea
TENSIÓN (V)
447
446
445
444
443
442
441
0
50
100
150
200
250
300
TIEMPO (S)
Tensión L1-L2
Tensión L2-L3
Tensión L3-L1

7.4.2 Tensiones de fase

v1 (V)	v2 (V)	v3 (V)	T (s)	Par
263,12	262,57	259,93	20	0,1 Nm
261,89	262,24	260,72	40	
262,48	261,36	260,51	60	
262,07	262,92	259,34	80	
261,98	260,86	260,78	100	
262,42	261,51	259,31	120	
261,19	261,04	258,53	140	0,3 Nm
260,14	261,57	259,64	160	
261,86	260,78	258,88	180	
260,52	259,25	258,91	200	
260,02	261,07	258,32	220	
261,07	259,93	258,41	240	
260,96	259,6	258,64	260	



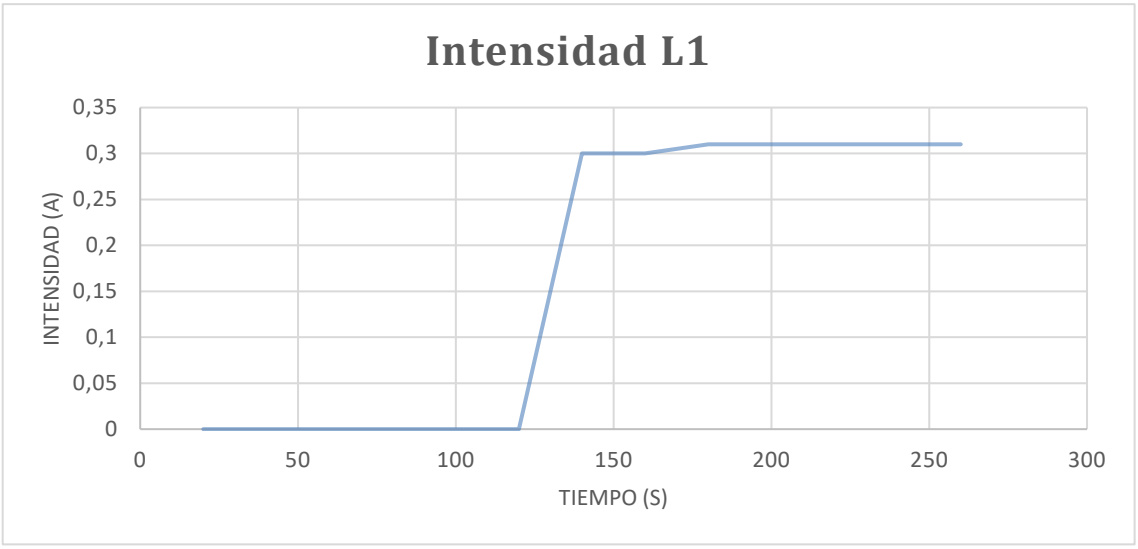

Tensión L3
261
260,5
260
259,5
259
258,5
258
TENSIÓN (V)
0
50
100
150
200
250
300
TIEMPO (S)

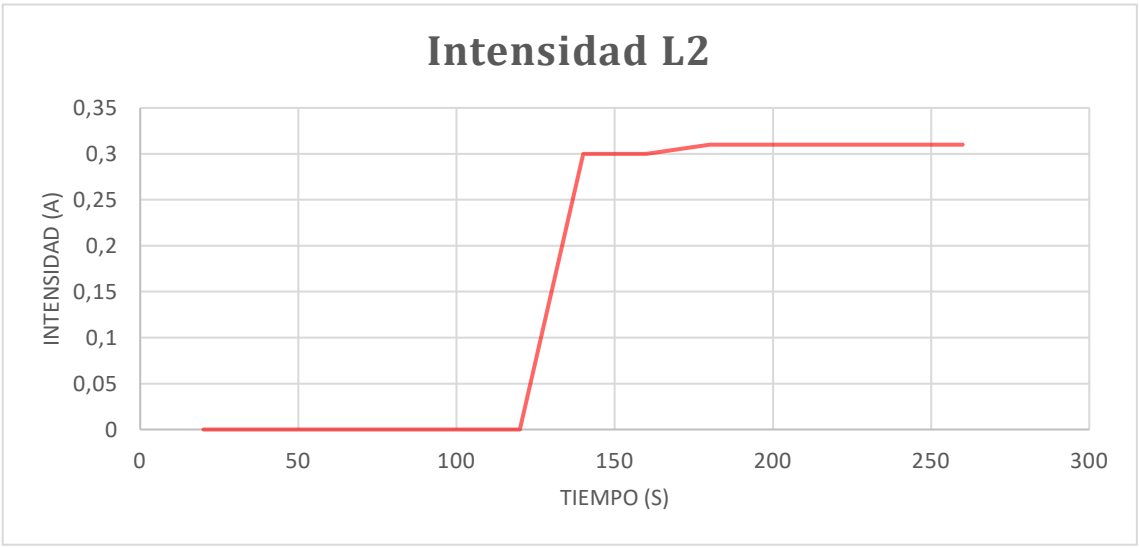

Tensiónes De Fase
264
263
262
261
260
259
258
tENSIÓN (V)
0
50
100
150
200
250
300
tIEMPO (S)
Tensión L1
Tensión L2
Tensión L3

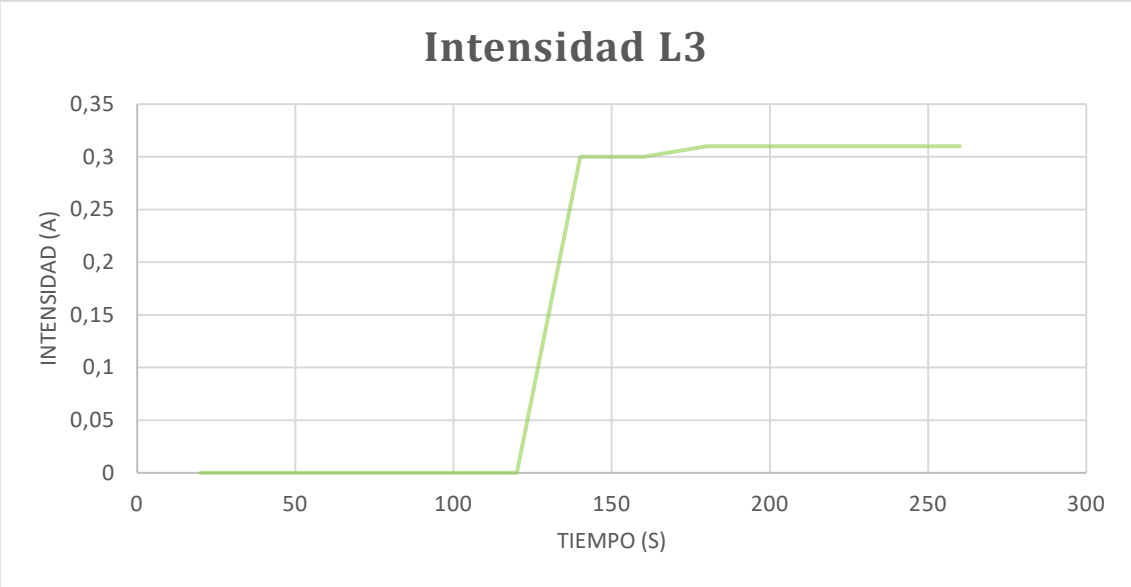
7.4.3 Intensidades de fase

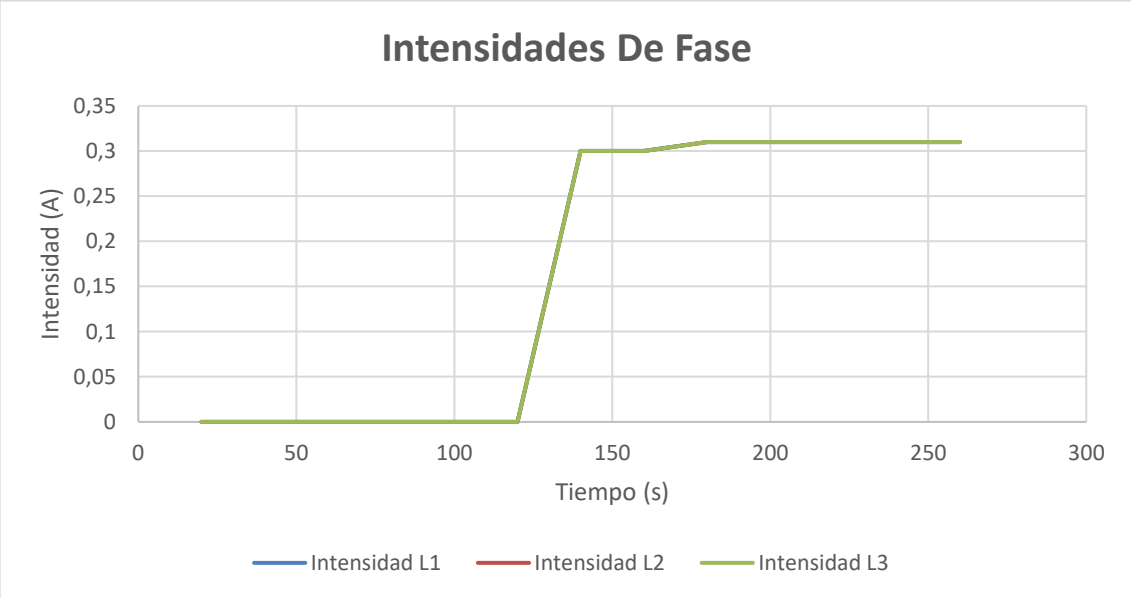
I1 (A)	I2 (A)	I3 (A)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0,3	0,3	0,3
0,3	0,3	0,3
0,31	0,31	0,31
0,31	0,31	0,31
0,31	0,31	0,31
0,31	0,31	0,31
0,31	0,31	0,31

T (s)	Par
20	0,1 Nm
40	
60	
80	
100	
120	
140	0,3 Nm
160	
180	
200	
220	
240	
260	





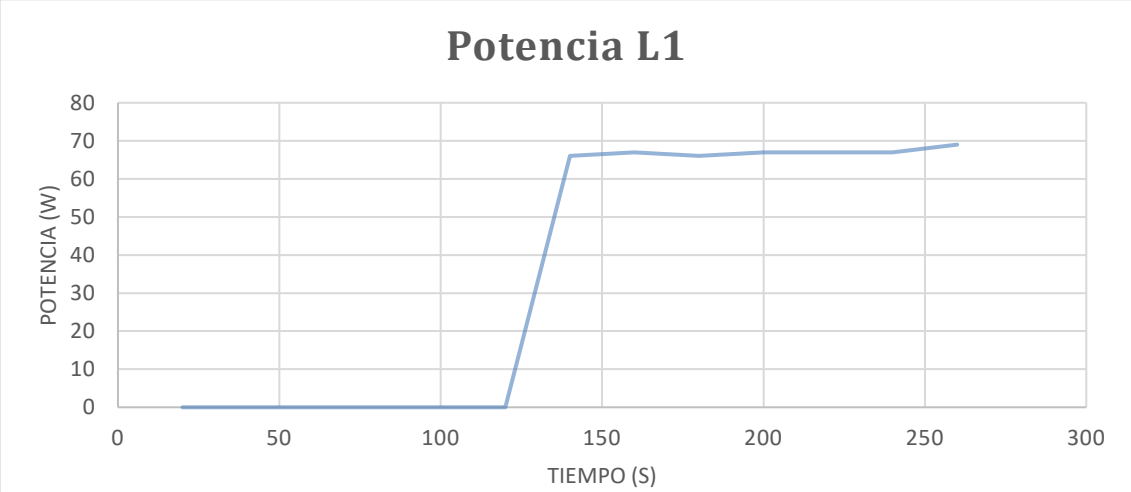

Intensidad L3
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
INTENSIDAD (A)
0
50
100
150
200
250
300
TIEMPO (S)

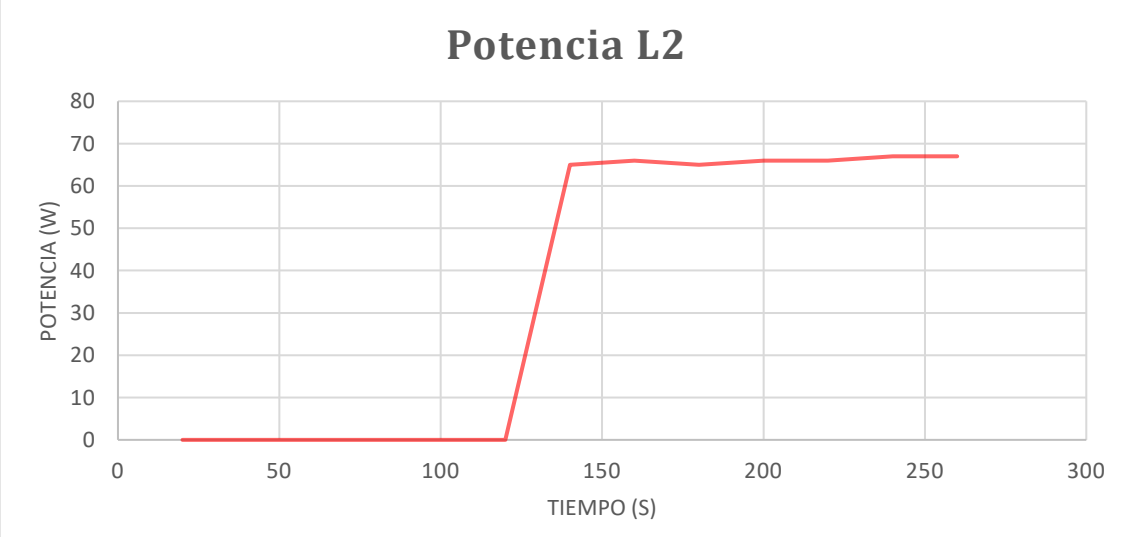

Intensidades De Fase
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
Intensidad (A)
0
50
100
150
200
250
300
Tiempo (s)
Intensidad L1
Intensidad L2
Intensidad L3

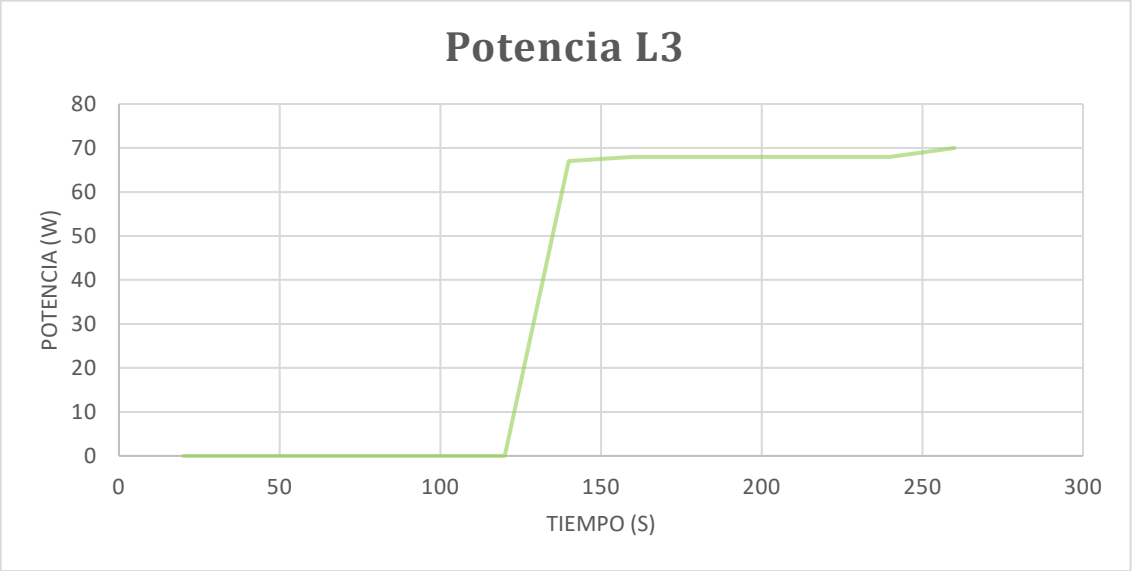
7.4.4 Potencias de fase

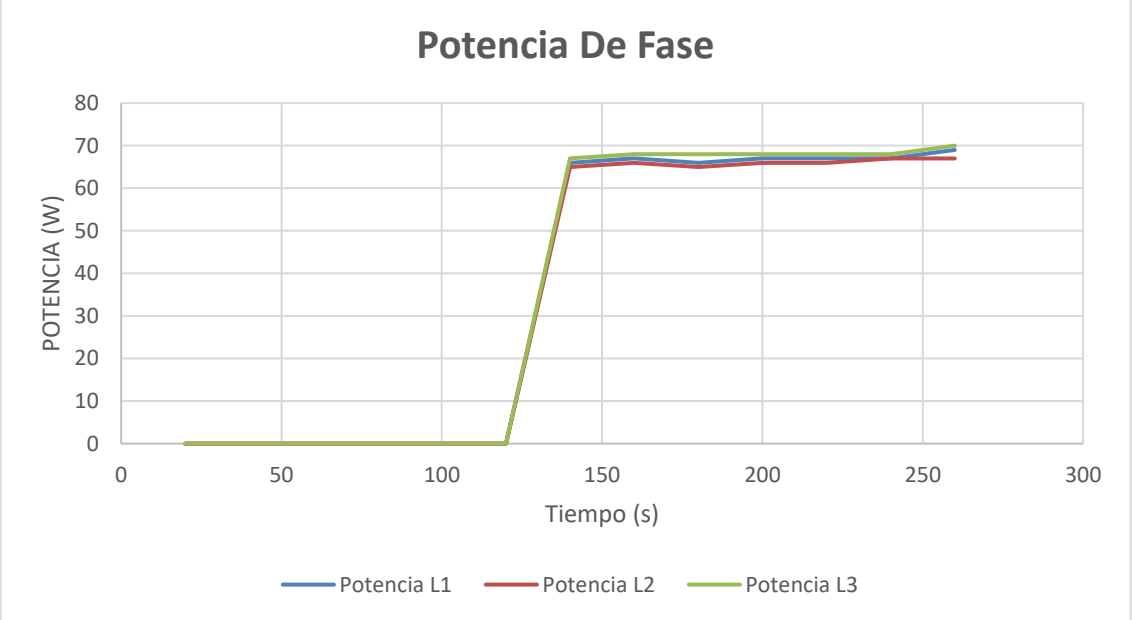
P1 (W)	P2 (W)	P3 (W)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
66	65	67
67	66	68
66	65	68
67	66	68
67	66	68
67	67	68
69	67	70

T (s)	Par
20	0,1 Nm
40	
60	
80	
100	
120	
140	0,3 Nm
160	
180	
200	
220	
240	
260	






Potencia L3
POTENCIA (W)
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
TIEMPO (S)

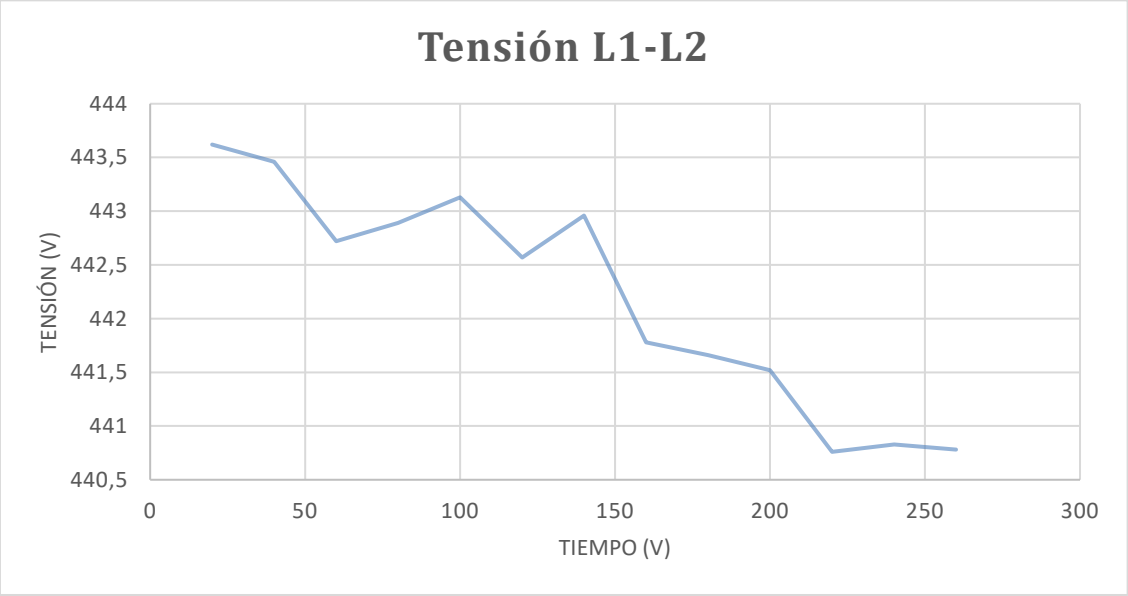

Potencia De Fase
POTENCIA (W)
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
Tiempo (s)
Potencia L1
Potencia L2
Potencia L3

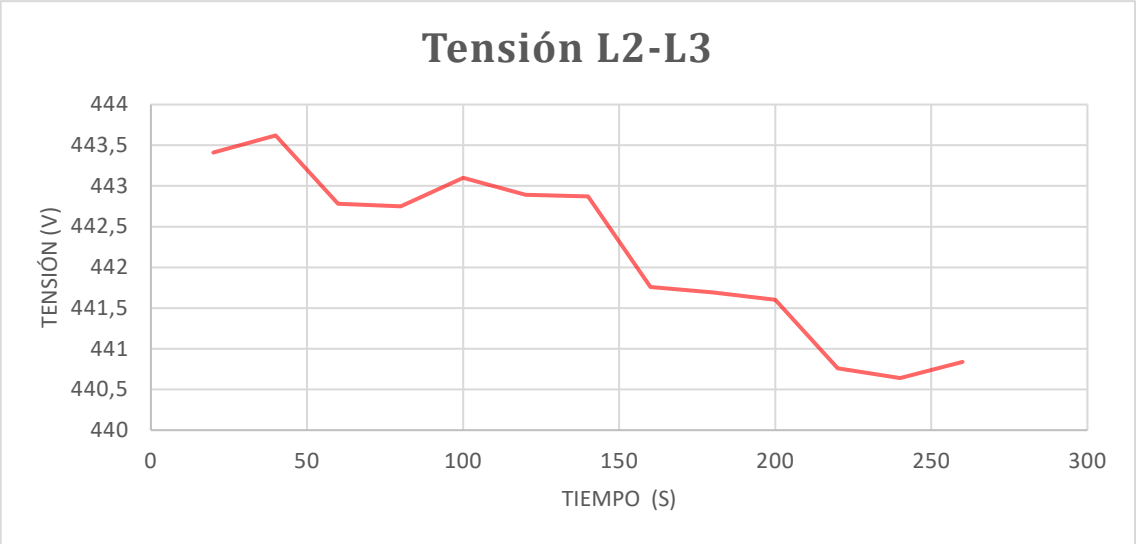
7.5 65 Hz

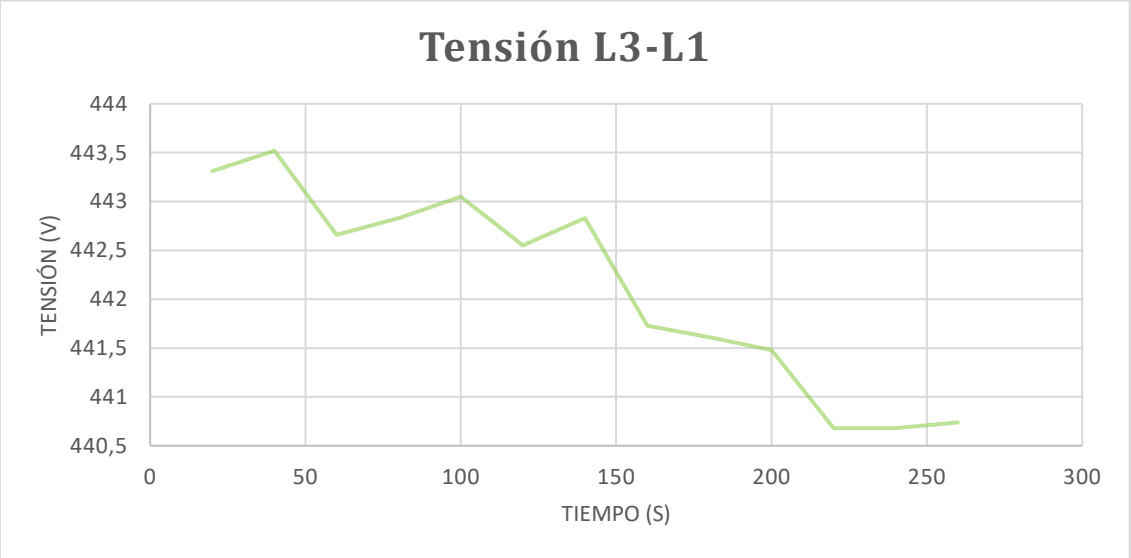
7.5.1 Tensiones de línea

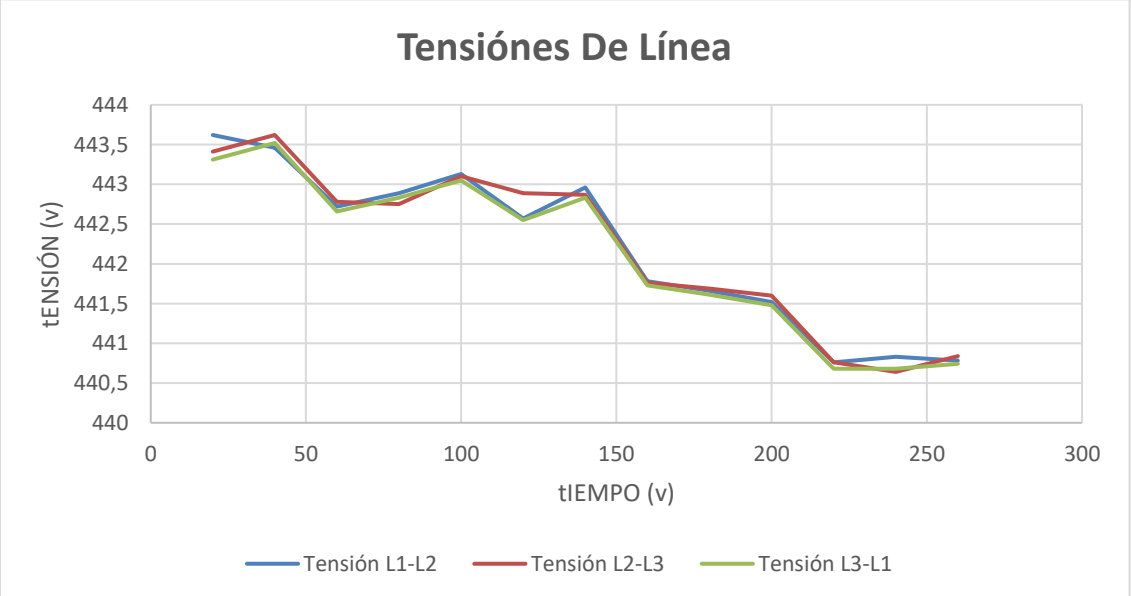
V1(V)	V2(V)	V3(V)
443,62	443,41	443,31
443,46	443,62	443,52
442,72	442,78	442,66
442,89	442,75	442,83
443,13	443,1	443,05
442,57	442,89	442,55
442,96	442,87	442,83
441,78	441,76	441,73
441,66	441,69	441,61
441,52	441,6	441,48
440,76	440,76	440,68
440,83	440,64	440,68
440,78	440,84	440,74

T(s)	Par (Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	




Tensión L2-L3
444
443,5
443
442,5
442
441,5
441
440,5
440
TENSIÓN (V)
0
50
100
150
200
250
300
TIEMPO (S)

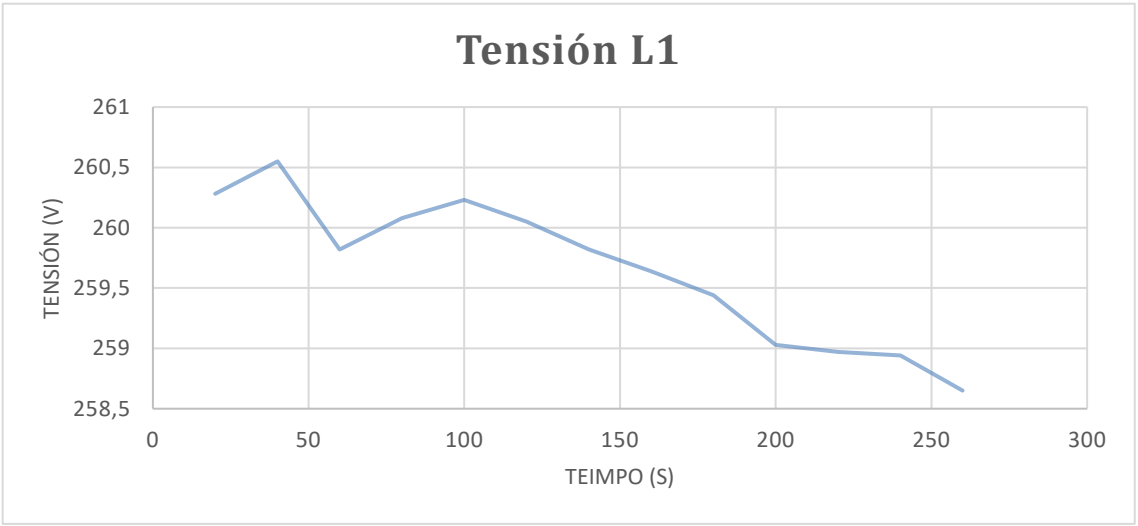

Tensión L3-L1
444
443,5
443
442,5
442
441,5
441
440,5
TENSIÓN (V)
0
50
100
150
200
250
300
TIEMPO (S)

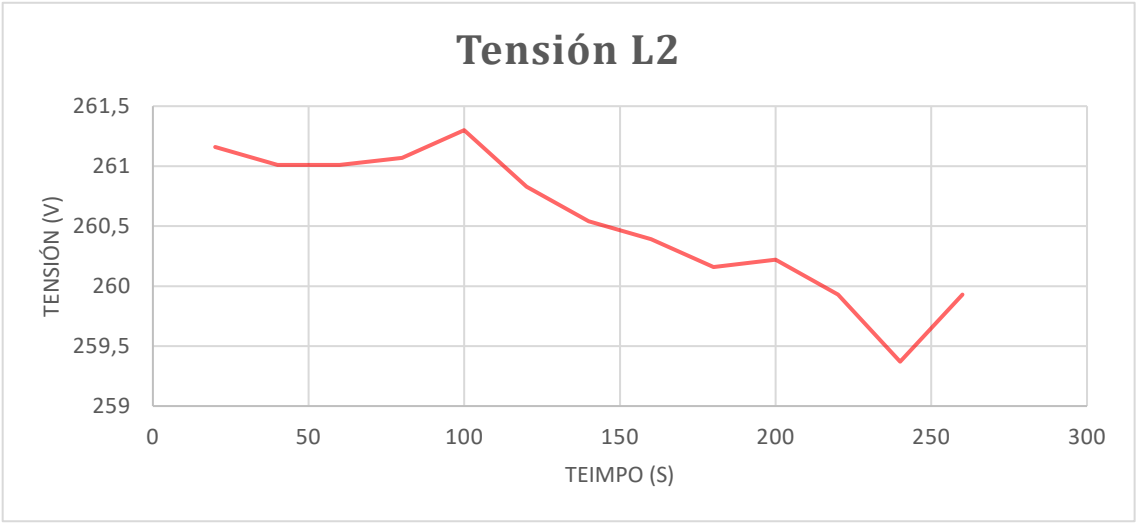

Tensiónes De Línea
444
443,5
443
442,5
442
441,5
441
440,5
440
tENSIÓN (v)
0
50
100
150
200
250
300
tIEMPO (v)
Tensión L1-L2
Tensión L2-L3
Tensión L3-L1

7.5.2 Tensiones de fase

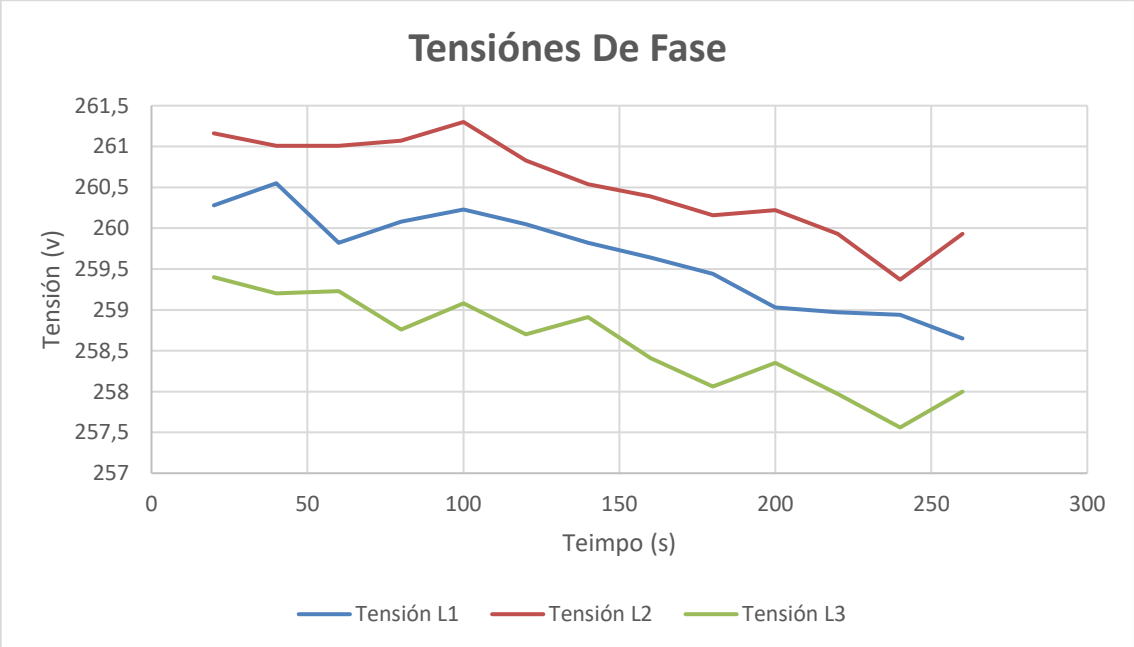
v1(V)	v2(V)	v3(V)
260,28	261,16	259,4
260,55	261,01	259,2
259,82	261,01	259,23
260,08	261,07	258,76
260,23	261,3	259,08
260,05	260,83	258,7
259,82	260,54	258,91
259,64	260,39	258,41
259,44	260,16	258,06
259,03	260,22	258,35
258,97	259,93	257,97
258,94	259,37	257,56
258,65	259,93	258

T(s)	Par (Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	





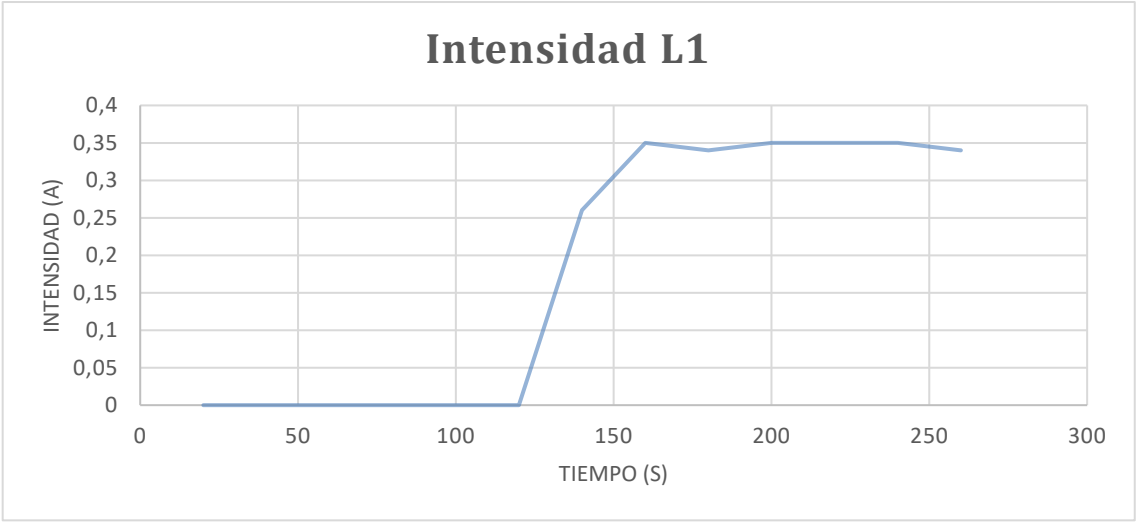

Tensión L3
259,5
259
258,5
258
257,5
257
TENSIÓN (V)
0
50
100
150
200
250
300
TEIMPO (S)

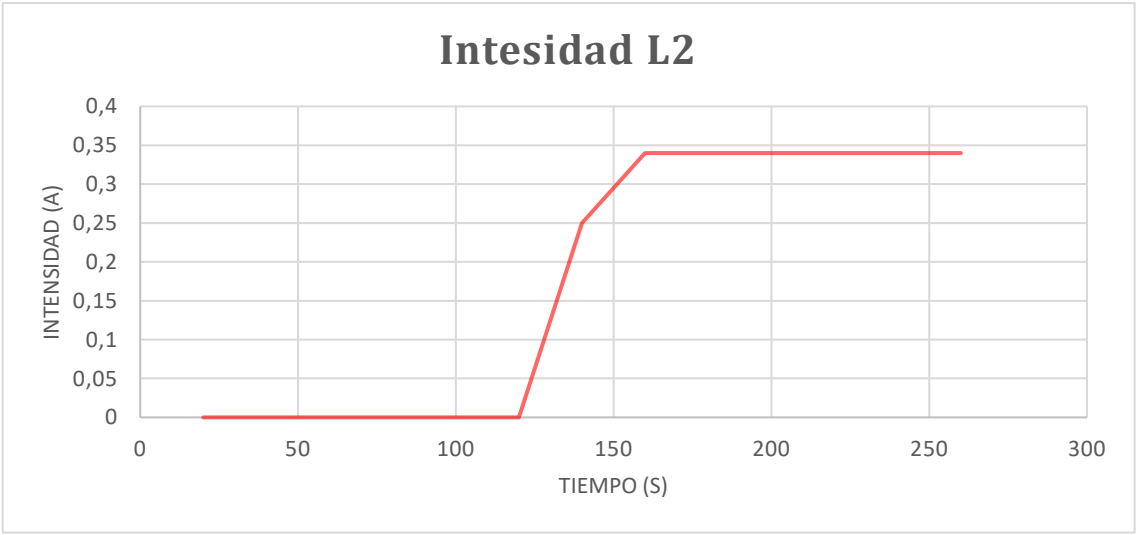

Tensiónes De Fase
261,5
261
260,5
260
259,5
259
258,5
258
257,5
257
Tensión (v)
0
50
100
150
200
250
300
Teimpo (s)
Tensión L1
Tensión L2
Tensión L3

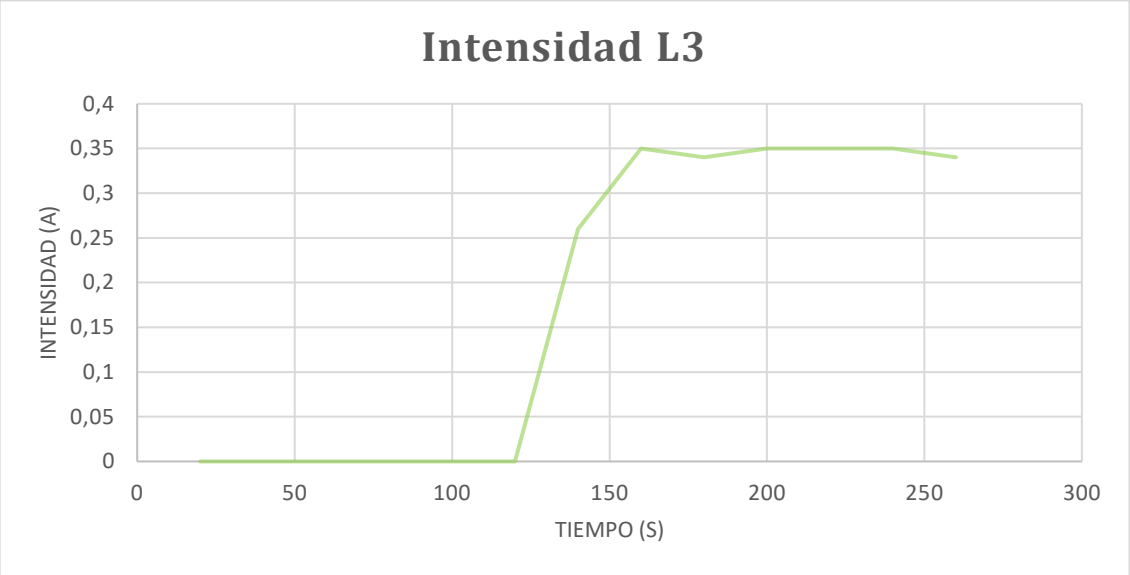
7.5.3 Intensidades de fase

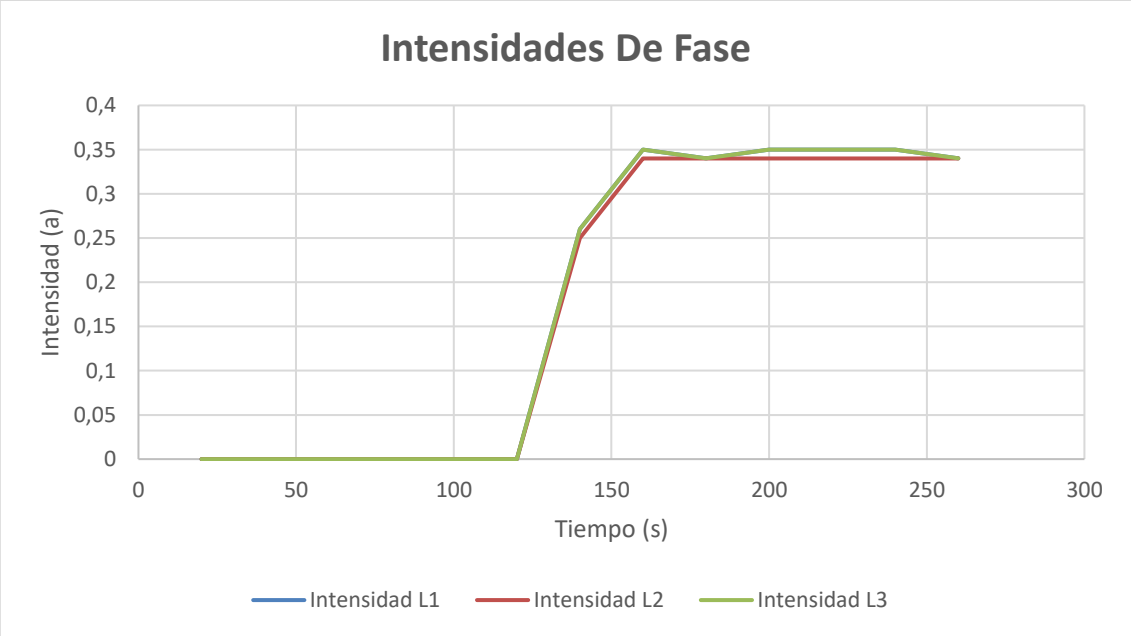
I1(A)	I2(A)	I3(A)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0,26	0,25	0,26
0,35	0,34	0,35
0,34	0,34	0,34
0,35	0,34	0,35
0,35	0,34	0,35
0,35	0,34	0,35
0,34	0,34	0,34

T(s)	Par (Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	





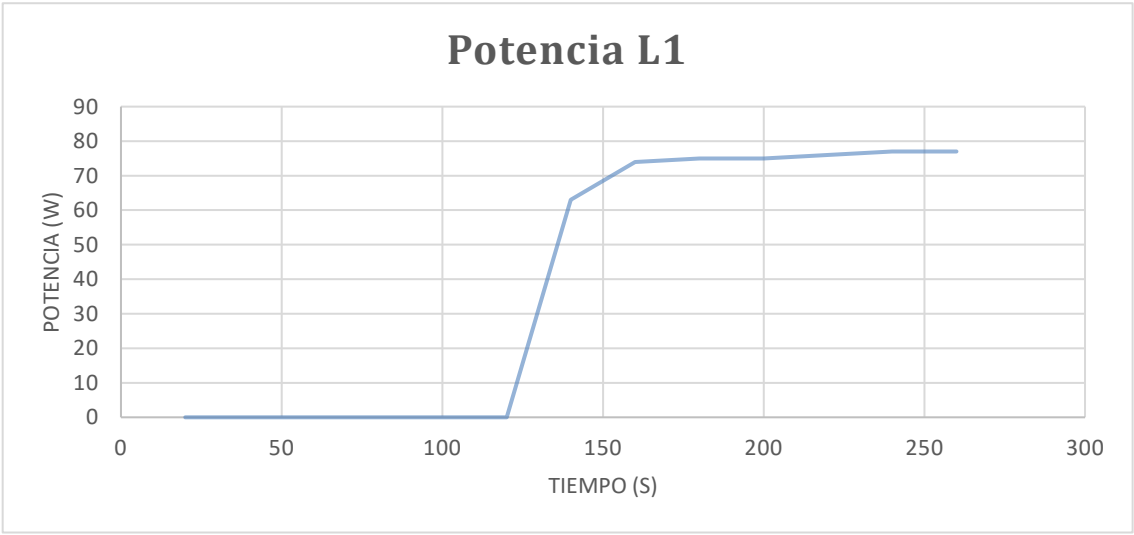

Intensidad L3
INTENSIDAD (A)
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
50
100
150
200
250
300
TIEMPO (S)

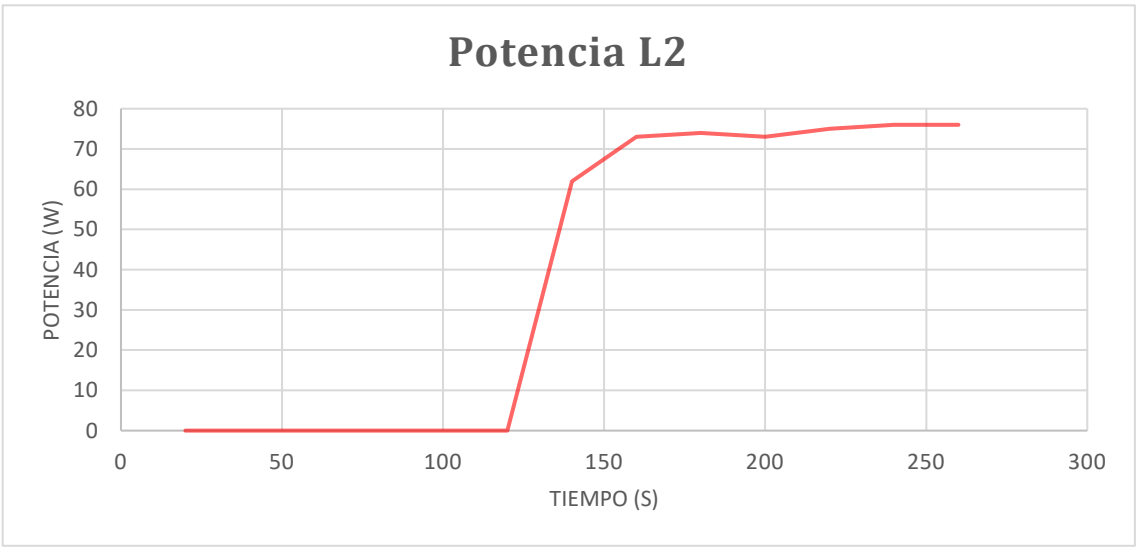

Intensidades De Fase
Intensidad (a)
0,4
0,35
0,3
0,25
0,2
0,15
0,1
0,05
0
0
50
100
150
200
250
300
Tiempo (s)
Intensidad L1
Intensidad L2
Intensidad L3

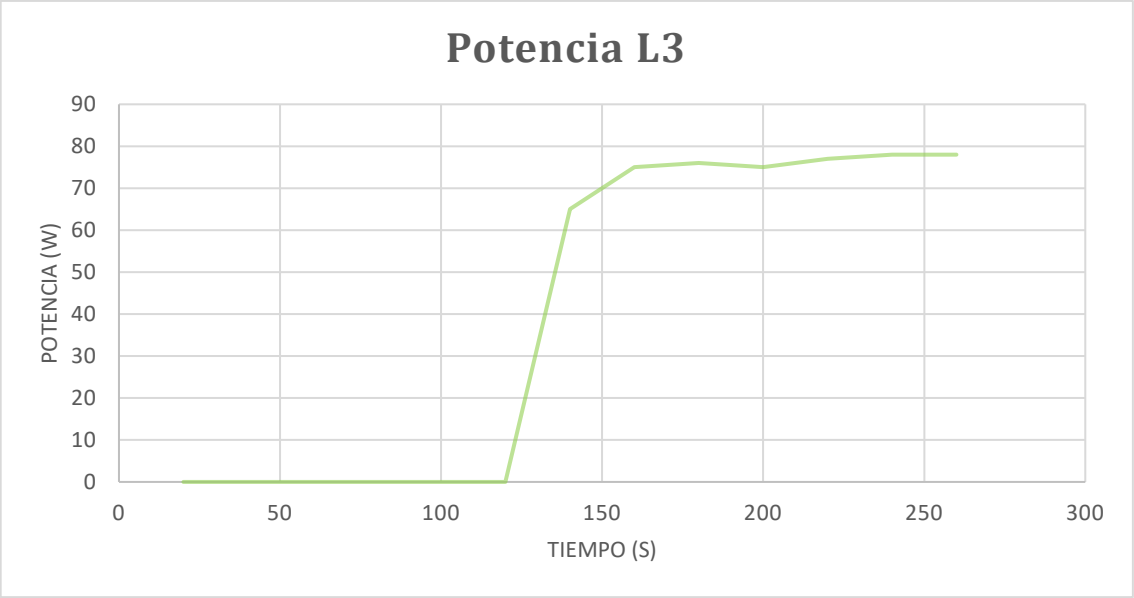
7.5.4 Potencias de fase

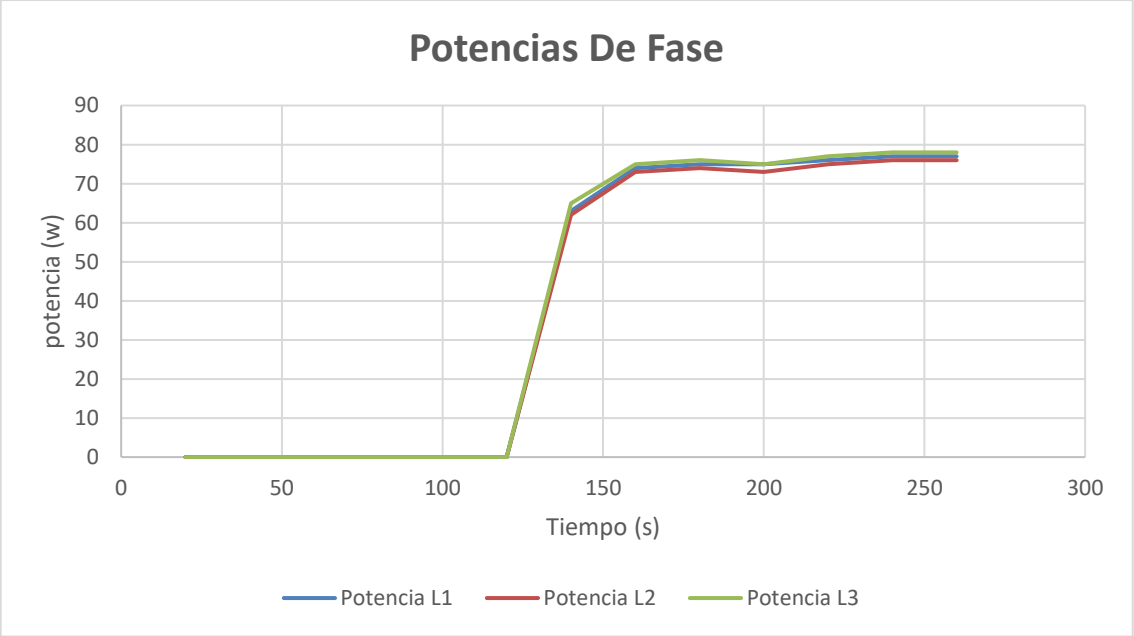
P1(W)	P2(W)	P3(W)
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
63	62	65
74	73	75
75	74	76
75	73	75
76	75	77
77	76	78
77	76	78

T(s)	Par (Nm)
20	0,1
40	
60	
80	
100	
120	
140	0,3
160	
180	
200	
220	
240	
260	






Potencia L3
POTENCIA (W)
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
TIEMPO (S)


Potencias De Fase
potencia (w)
90
80
70
60
50
40
30
20
10
0
0
50
100
150
200
250
300
Tiempo (s)
Potencia L1
Potencia L2
Potencia L3

8 Bibliografía

(s.f.).

Aguiar, E. U. (s.f.). *Universidad Politécnica de Sinaloa.* Obtenido de http://repositorio.upsin.edu.mx/formatos/TesinaEdgarUlisesAgueroAguiar6855.pdf

Alfonso, E. (s.f.). *Academia*. Obtenido de https://www.academia.edu/7581220/Como_funciona_un_motor_electrico_trifasico_de_jaula_de_ardilla

Alicante, U. d. (s.f.). *Comunicación con RS-485 y MODBUS.* Obtenido de https://rua.ua.es/dspace/bitstream/10045/18990/1/AA-p3.pdf

AprendiendoArduino. (s.f.). Obtenido de https://aprendiendoarduino.wordpress.com/2016/11/16/librerias-arduino-2/

Barastegui, J. J. (s.f.). *Escuela Técnica Superior de Ingeniería Universidad de Sevilla.* Obtenido de http://bibing.us.es/proyectos/abreproy/91400/fichero/Memoria+TFG+JJRB.pdf

ECURED. (s.f.). Obtenido de https://www.ecured.cu/Rotor_de_jaula_de_ardilla

González, A. G. (s.f.). *PANAMAHITEK*. Obtenido de http://panamahitek.com/arduino-mega-caracteristicas-capacidades-y-donde-conseguirlo-en-panama/

Programo ERGO SUM. (s.f.). Obtenido de https://www.programoergosum.com/cursos-online/appinventor/27-curso-de-programacion-con-app-inventor/primeros-pasos

Programo ERGO SUM. (s.f.). Obtenido de https://www.programoergosum.com/cursos-online/appinventor/27-curso-de-programacion-con-app-inventor/primeros-pasos

Riego, A. R. (s.f.). *App Inventor en Español*. Obtenido de https://sites.google.com/site/appinventormegusta/instalacion/instalar-y-ejecutar-el-emulador-en-ai2

Valle, L. d. (s.f.). *programarfacil.com*. Obtenido de https://programarfacil.com/podcast/nodemcu-tutorial-paso-a-paso/

ANEXOS

Anexo 1: Código Arduino

➢ Comunicación Modbus

```
#include <Wire.h>
#include <RTClib.h>
#include <ModbusMaster.h>

// instantiate ModbusMaster object
ModbusMaster node;


uint8_t j,
result1,result2,result3,result4,result5,result6,result7,result8,
result9;  //Variables de 8 bits
uint16_t
data1[6],data2[6],data3[6],data4[6],data5[6],data6[6],data7[6],d
ata8[6],data9[6]; //Vectores de 6 cajas, cada una de 16 bits .

String envio;

char envio20[100];


String envio2;
String envio3;
String envio4;


float V[9],v[9],Aux[9],I[9],P[9],Q[9],R[9],S[9],F[9],C[9],N[9];
float S3[3],Fp[3],Cos[3],Cu[3];
float Pot3[9],VyA3[9],QL[9],Qcon[9],Qgen[9],Q3[9];
float CN[1];

//DynamicJsonBuffer jsonBuffer;
RTC_DS1307 rtc;


void setup()
{
  // use Serial (port 0); initialize Modbus communication baud
rate
  Serial.begin(19200);
  Serial1.begin(9600);
  //Serial1.begin(19200);

   // Si se ha perdido la corriente, fijar fecha y hora
  if (! rtc.begin()) {
    //Serial.println("Couldn't find RTC");
    while (1);
  }
   if (!rtc.isrunning()) {
      // Fijar a fecha y hora de compilacion
      rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
```

```
      //Serial.print("Reloj no funciona");
      // Fijar a fecha y hora específica. En el ejemplo, 21 de
Enero de 2016 a las 03:00:00
      //rtc.adjust(DateTime(2016, 1, 21, 3, 0, 0));
    }
  else {Serial.println("Reloj funcionando");}
  //rtc.adjust(DateTime(2020, 07, 17, 9, 7, 0)); Una vez subida
se quita para que le reloj siga desde ese momento.
  // communicate with Modbus slave ID 2 over Serial (port 0)
  node.begin(1,Serial);

  delay(5000);

}

void loop()
{


  DateTime now = rtc.now();
  envio3="t";
  envio3 +=String(now.day());
  envio3 += ";";
  envio3 += String(now.month());
  envio3 += ";";
  envio3 += String(now.year());
  envio3 += ";";
  envio3 += String(now.hour());
  envio3 += ";";
  envio3 += String(now.minute());
  envio3 += ";";
  envio3 += String(now.second());
  envio3 += ";";


 lee("V",0x36,0x38,0x3A,0x016C,0x036C,0x0170,0x0370,0x0174,0x037
4,100); //Tensiones linea
  for(int i=0; i<9; i++) {V[i]=Aux[i];}
    envio="V";
    for (int i=0; i<9; i++){
      envio += ";";
      envio += String(V[i]);
    }
      for(int i=1; i<=100; i++) {envio20[i]=";";}
      envio.toCharArray(envio20,91);
      Serial1.write(envio20);
      Serial1.println();
    delay(200);


 lee("v",0x00,0x010,0x020,0x0100,0x0300,0x0120,0x0320,0x0140,0x0
340,100); //Tensiones fase en L1
  for(int i=0; i<9; i++) {v[i]=Aux[i];}
  envio="v";
  for (int i=0; i<9; i++){
      envio += ";";
```

```
      envio += String(v[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  // Enviamos la fecha para escribirla en cada firebase.
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"V";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
```

```
 lee("I",0x02,0x12,0x22,0x0104,0x0304,0x0124,0x0324,0x0144,0x034
4,1000); //Corrientes
  for(int i=0; i<9; i++) {I[i]=Aux[i];}
  envio="I";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(I[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"I";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
```

```
 lee("P",0x04,0x14,0x24,0x0108,0x0308,0x0128,0x0328,0x0148,0x034
8,1); //Activa
  for(int i=0; i<9; i++) {P[i]=Aux[i];}
  envio="P";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(P[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"P";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
```

```
  Serial1.write(envio20);
  Serial1.println();
```

```
 lee("Q",0x06,0x16,0x26,0x010C,0x030C,0x012C,0x032C,0x014C,0x034
C,1); //Potencia Reactiva inductiva
  for(int i=0; i<9; i++) {Q[i]=Aux[i];}
  envio="Q";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(Q[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"q";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
```

```
 lee("R",0x08,0x18,0x28,0x0110,0x0310,0x0130,0x0330,0x0150,0x035
0,1000); //Potencia Reactiva capacitiva
  for(int i=0; i<9; i++) {R[i]=Aux[i];}
  envio="R";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(R[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"R";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
```

```
 lee("S",0x0A,0x1A,0x2A,0x0114,0x0314,0x0134,0x0334,0x0154,0x035
4,1000); //Potencia Aparente
  for(int i=0; i<9; i++) {S[i]=Aux[i];}
  envio="S";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(S[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
```

```
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"S";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("F",0x0C,0x1C,0x2C,0x0118,0x0318,0x0138,0x0338,0x0158,0x035
8,100); //Factor de potencia
  for(int i=0; i<9; i++) {F[i]=Aux[i];}
  envio="F";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(F[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"F";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("C",0x0E,0x1E,0x2E,0x011C,0x031C,0x013C,0x033C,0x015C,0x035
C,100); //Coseno fi
  for(int i=0; i<9; i++) {C[i]=Aux[i];}
  envio="C";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(C[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"C";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("N",0x30,0x32,0x34,0x0160,0x0360,0x0164,0x0364,0x0168,0x036
8,100); //Tension y corriente del neutro y frecuencia
```

```
  for(int i=0; i<9; i++) {N[i]=Aux[i];}
  envio="N";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(N[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"N";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("VyA3",0x03E,0x0178,0x0378,0x03E,0x040,0x017C,0x037C,0x0180
,0x0380,100); //Tensión línea III ,Tensión y corriente
trifásica.
  for(int i=0; i<9; i++) {VyA3[i]=Aux[i];}
  envio="VyA3";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(VyA3[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"VyA3";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("Pot3",0x042,0x044,0x046,0x0184,0x0384,0x0188,0x0388,0x018C
,0x038C,1); //Potencia activa,inductiva y capacitiva trifásica
  for(int i=0; i<9; i++) {Pot3[i]=Aux[i];}
  envio="Pot3";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(Pot3[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Pot3";
```

```
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();

  leer("S3",0x048,0x0190,0x0390,1); //Potencia aparente
trifásica
  for(int i=0; i<3; i++) {Q3[i]=Aux[i];}
  envio="S3";
  for (int i=0; i<3; i++){
      envio += ";";
      envio += String(S3[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"S3";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();

  leer("Fp",0x04A,0x0194,0x0394,100); //Factor de potencia
trifásica
  for(int i=0; i<3; i++) {Fp[i]=Aux[i];}
  envio="Fp";
  for (int i=0; i<3; i++){
      envio += ";";
      envio += String(Fp[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Fp";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();

  leer("Cos",0x04C,0x198,0x0398,100); //Cos de fi trifásico
  for(int i=0; i<3; i++) {Cos[i]=Aux[i];}
  envio="Cos";
  for (int i=0; i<3; i++){
      envio += ";";
      envio += String(Cos[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
```

```
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Cos";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("QL",0x05E,0x060,0x062,0x01BC,0x03BC,0x01C0,0x03C0,0x01C4,0
x03C4,1); //Potencia reactiva en L1, L2 y L3.
  for(int i=0; i<9; i++) {QL[i]=Aux[i];}
  envio="QL";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(QL[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"QL";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("Qcon",0x066,0x068,0x06A,0x01CC,0x03CC,0x01D0,0x03D0,0x1D4,
0x03D4,1); //Potencia reactiva consumida por L1, L2 y L3
  for(int i=0; i<9; i++) {Qcon[i]=Aux[i];}
  envio="Qcon";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(Qcon[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Qcon";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("Qgen",0x06E,0x06E,0x070,0x01DC,0x03DC,0x01E0,0x03E0,0x01E4
,0x03E4,1); //Potencia reactiva generada en L1, L2 y L3
  for(int i=0; i<9; i++) {Qgen[i]=Aux[i];}
  envio="Qgen";
  for (int i=0; i<9; i++){
```

```
      envio += ";";
      envio += String(Qgen[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Qgen";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();


 lee("Q3",0x064,0x06C,0x074,0x01C8,0x03C8,0x01D8,0x03D8,0x01E8,0
x03D8,1); //Potencia reactiva trifasica, consumida y generada
trifásica.
  for(int i=0; i<9; i++) {Q3[i]=Aux[i];}
  envio="Q3";
  for (int i=0; i<9; i++){
      envio += ";";
      envio += String(Q3[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Q3";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Q3";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();

  leer("Cu",0x076,0x077,0x078,1); //Cuadrante Fase I,II y III
  for (int i=0; i<3; i++) {Cu[i]=Aux[i];}
  envio="Cu";
  for (int i=0; i<3; i++){
      envio += ";";
      envio += String(Cu[i]);
  }
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"Cu";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
```

```
  leeuno("CN",0x079,1); //Cuadrante Neutro
  CN[0]=Aux[0];
  envio="CN";
  envio += ";";
  envio += String(CN[0]);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();
  delay(200);
  for(int i=1; i<=100; i++) {envio20[i]=";";}
  envio4=envio3+"CN";
  envio4 += ";";
  envio4.toCharArray(envio20,91);
  Serial1.write(envio20);
  Serial1.println();



  delay(2000);
}

void lee(String variable, int d1, int d2, int d3, int d4, int
d5, int d6, int d7, int d8, int d9, float multiplicador){
  delay(50);
  result1=node.readInputRegisters(d1,2);
  if (result1==node.ku8MBSuccess){ //Coincide de que se ha leido
correcto (0) y la transaccion es correcta (0).

    result1=0;
    for (j =0; j < 6; j++) {data1[j]=node.getResponseBuffer(j);}
  }
  delay(50);

  result2=node.readInputRegisters(d2,2);
  if (result2 == node.ku8MBSuccess)
  {

    result2=0;
    for (j = 0; j < 6; j++) {data2[j] =
node.getResponseBuffer(j);}
  }
  delay(50);

  result3=node.readInputRegisters(d3,2);
  if (result3 == node.ku8MBSuccess)
  {

    result3=0;
    for (j = 0; j < 6; j++) {data3[j] =
node.getResponseBuffer(j);}
  }
  delay(50);
```

```
  result4=node.readInputRegisters(d4,2);
  if (result4 == node.ku8MBSuccess)
  {
    result4=0;
    for (j = 0; j < 6; j++) {data4[j] =
node.getResponseBuffer(j);}
  }
  delay(50);

  result5=node.readInputRegisters(d5,2);
  if (result5 == node.ku8MBSuccess)
  {
    result5=0;
    for (j = 0; j < 6; j++) {data5[j] =
node.getResponseBuffer(j);}
  }
  delay(50);

  result6=node.readInputRegisters(d6,2);
  if (result6 == node.ku8MBSuccess)
  {
    result6=0;
    for (j = 0; j < 6; j++) {data6[j] =
node.getResponseBuffer(j);}
  }
  delay(50);

  result7=node.readInputRegisters(d7,2);
  if (result7 == node.ku8MBSuccess)
  {
    result7=0;
    for (j = 0; j < 6; j++) {data7[j] =
node.getResponseBuffer(j);}
  }
  delay(50);

  result8=node.readInputRegisters(d8,2);
  if (result8 == node.ku8MBSuccess)
  {
    result8=0;
    for (j = 0; j < 6; j++) {data8[j] =
node.getResponseBuffer(j);}
  }
  delay(50);

  result9=node.readInputRegisters(d9,2);
  if (result9 == node.ku8MBSuccess)
  {
    result9=0;
    for (j = 0; j < 6; j++) {data9[j] =
node.getResponseBuffer(j);}
  }

  envio2=String(data1[1]);
  Aux[0]=envio2.toFloat()/multiplicador;
  envio2=String(data2[1]);
```

```
  Aux[1]=envio2.toFloat()/multiplicador;
  envio2=String(data3[1]);
  Aux[2]=envio2.toFloat()/multiplicador;
  Aux[3]=envio2.toFloat()/multiplicador;
  envio2=String(data5[1]);
  Aux[4]=envio2.toFloat()/multiplicador;
  envio2=String(data6[1]);
  Aux[5]=envio2.toFloat()/multiplicador;
  envio2=String(data7[1]);
  Aux[6]=envio2.toFloat()/multiplicador;
  envio2=String(data8[1]);
  Aux[7]=envio2.toFloat()/multiplicador;
  envio2=String(data9[1]);
  Aux[8]=envio2.toFloat()/multiplicador;
}

void leer(String variable, int d1, int d2, int d3,float
multiplicador){
  delay(50);
  result1=node.readInputRegisters(d1,2);
  if (result1==node.ku8MBSuccess){
    result1=0;
    for(j=0; j<6 ;j++){
      data1[j]=node.getResponseBuffer(j);
    }
  }
  delay(50);

  result2=node.readInputRegisters(d2,2);
  if (result2==node.ku8MBSuccess){
    result2=0;
    for(j=0; j<6; j++){
      data2[j]=node.getResponseBuffer(j);
    }
  }
  delay(50);

  result3=node.readInputRegisters(d3,2);
  if (result3==node.ku8MBSuccess){
    result3=0;
    for(j=0; j<6; j++){
      data3[j]=node.getResponseBuffer(j);
    }
  }

  envio2=String(data1[1]);
  Aux[0]=envio2.toFloat()/multiplicador;
  envio2=String(data2[1]);
  Aux[1]=envio2.toFloat()/multiplicador;
  envio2=String(data3[1]);
  Aux[2]=envio2.toFloat()/multiplicador;
}

void leeuno(String variable,int d1,float multiplicador){
  delay(50);
  result1=node.readInputRegisters(d1,2);
```

```
  if (result1==node.ku8MBSuccess){
    result1=0;
    for(j=0; j<6 ;j++){
      data1[j]=node.getResponseBuffer(j);
    }
  }
  envio2=String(data1[1]);
  Aux[0]=envio2.toFloat()/multiplicador;
}
```

- **Subida de datos a la nube**

```
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <Wire.h>

#define FIREBASE_HOST "pruebanalizador.firebaseio.com"
#define FIREBASE_AUTH "LWkMRmrL26YaIl6oTbBkS8sYQwBFcb7X10RV2CZz"

#define WIFI_SSID "IngElectrica"
#define WIFI_PASSWORD "C@IngEle#U120"

DynamicJsonBuffer jsonBuffer;
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"V1\":0.00,\"V2\":0.00
,\"V3\":0.00,\"V4\":0.00,\"V5\":0.00,\"V6\":0.00,\"V7\":0.00,\"V
8\":0.00,\"V9\":0.00,\"v1\":0.00,\"v2\":0.00,\"v3\":0.00,\"v4\":
0.00,\"v5\":0.00,\"v6\":0.00,\"v7\":0.00,\"v8\":0.00,\"v9\":0.00
}");
JsonObject& json1 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"I1\":0.00,\"I2\":0.00
,\"I3\":0.00,\"I4\":0.00,\"I5\":0.00,\"I6\":0.00,\"I7\":0.00,\"I
8\":0.00,\"I9\":0.00}");
JsonObject& json2 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"P1\":0.00,\"P2\":0.00
,\"P3\":0.00,\"P4\":0.00,\"P5\":0.00,\"P6\":0.00,\"P7\":0.00,\"P
8\":0.00,\"P9\":0.00}");
JsonObject& json3 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Q1\":0.00,\"Q2\":0.00
,\"Q3\":0.00,\"Q4\":0.00,\"Q5\":0.00,\"Q6\":0.00,\"Q7\":0.00,\"Q
8\":0.00,\"Q9\":0.00}");
JsonObject& json4 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"R1\":0.00,\"R2\":0.00
,\"R3\":0.00,\"R4\":0.00,\"R5\":0.00,\"R6\":0.00,\"R7\":0.00,\"R
8\":0.00,\"R9\":0.00}");
JsonObject& json5 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"S1\":0.00,\"S2\":0.00
,\"S3\":0.00,\"S4\":0.00,\"S5\":0.00,\"S6\":0.00,\"S7\":0.00,\"S
8\":0.00,\"S9\":0.00}");
JsonObject& json6 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"F1\":0.00,\"F2\":0.00
,\"F3\":0.00,\"F4\":0.00,\"F5\":0.00,\"F6\":0.00,\"F7\":0.00,\"F
8\":0.00,\"F9\":0.00}");
JsonObject& json7 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"C1\":0.00,\"C2\":0.00
```

```
,\"C3\":0.00,\"C4\":0.00,\"C5\":0.00,\"C6\":0.00,\"C7\":0.00,\"C
8\":0.00,\"C9\":0.00}");
JsonObject& json8 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"N1\":0.00,\"N2\":0.00
,\"N3\":0.00,\"N4\":0.00,\"N5\":0.00,\"N6\":0.00,\"N7\":0.00,\"N
8\":0.00,\"N9\":0.00}");
JsonObject& json9 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"VyA31\":0.00,\"VyA32\
":0.00,\"VyA33\":0.00,\"VyA34\":0.00,\"VyA35\":0.00,\"VyA36\":0.
00,\"VyA37\":0.00,\"VyA38\":0.00,\"VyA39\":0.00}");
JsonObject& json10 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Pot31\":0.00,\"Pot32\
":0.00,\"Pot33\":0.00,\"Pot34\":0.00,\"Pot35\":0.00,\"Pot36\":0.
00,\"Pot37\":0.00,\"Pot38\":0.00,\"Pot39\":0.00}");
JsonObject& json11 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"S31\":0.00,\"S32\":0.
00,\"S33\":0.00}");
JsonObject& json12 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Fp1\":0.00,\"Fp2\":0.
00,\"Fp3\":0.00}");
JsonObject& json13 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Cos1\":0.00,\"Cos2\":
0.00,\"Cos3\":0.00}");
JsonObject& json14 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"QL1\":0.00,\"QL2\":0.
00,\"QL3\":0.00,\"QL4\":0.00,\"QL5\":0.00,\"QL6\":0.00,\"QL7\":0
.00,\"QL8\":0.00,\"QL9\":0.00}");
JsonObject& json15 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Qcon1\":0.00,\"Qcon2\
":0.00,\"Qcon3\":0.00,\"Qcon4\":0.00,\"Qcon5\":0.00,\"Qcon6\":0.
00,\"Qcon7\":0.00,\"Qcon8\":0.00,\"Qcon9\":0.00}");
JsonObject& json16 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Qgen1\":0.00,\"Qgen2\
":0.00,\"Qgen3\":0.00,\"Qgen4\":0.00,\"Qgen5\":0.00,\"Qgen6\":0.
00,\"Qgen7\":0.00,\"Qgen8\":0.00,\"Qgen9\":0.00}");
JsonObject& json17 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Q31\":0.00,\"Q32\":0.
00,\"Q33\":0.00,\"Q34\":0.00,\"Q35\":0.00,\"Q36\":0.00,\"Q37\":0
.00,\"Q38\":0.00,\"Q39\":0.00}");
JsonObject& json18 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"Cu1\":0.00,\"Cu2\":0.
00,\"Cu3\":0.00}");
JsonObject& json19 =
jsonBuffer.parseObject("{\"dia\":0.0,\"mes\":0.0,\"ano\":0.0,\"h
ora\":0.0,\"minuto\":0.0,\"segundo\":0.0,\"CN\":0.00}");
```

```
char str[100];
int c;
String s2;

void setup() {
  Serial.begin(9600);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Serial.println("Firebase");
  delay(5000);
}

void loop() {
  unsigned long t;
  int i=0;
  int j;
  String s;

  int k;
  String dia;
  String mes;
  String ano;
  int dian,mesn,anon;
  t=millis();

  if (Serial.available()) {
    delay(100); //allows all serial sent to be received together
    while(Serial.available() && i<=90) {
      str[i++] = Serial.read();
    }
    str[i++]='\0';
  }
  if(i>0) {
    String s1;
    i=0;
    s="";
    j=1;
    k=0;
   s1=str[0];
   if (s1=="V"){
      //s2="V";
   while (j<90) {
      s1 = str[j];
      if (s1 != ";") {
        s += s1;
```

```
    }
    else {
      switch(k) {
        case 1: json["V1"]=s.toFloat();
        break;
        case 2: json["V2"]=s.toFloat();
        break;
        case 3: json["V3"]=s.toFloat();
        break;
        case 4: json["V4"]=s.toFloat();
        break;
        case 5: json["V5"]=s.toFloat();
        break;
        case 6: json["V6"]=s.toFloat();
        break;
        case 7: json["V7"]=s.toFloat();
        break;
        case 8: json["V8"]=s.toFloat();
        break;
        case 9: json["V9"]=s.toFloat();
        break;
      }
      s="";
      k += 1;
    }
    j += 1;
   }
}

 s="";
 j=1;
 k=0;
s1=str[0];
//s2=s1;
if (s1=="v"){
 while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["v1"]=s.toFloat();
       break;
       case 2: json["v2"]=s.toFloat();
       break;
       case 3: json["v3"]=s.toFloat();
       break;
       case 4: json["v4"]=s.toFloat();
       break;
       case 5: json["v5"]=s.toFloat();
       break;
       case 6: json["v6"]=s.toFloat();
       break;
       case 7: json["v7"]=s.toFloat();
       break;
```

```
          case 8: json["v8"]=s.toFloat();
          break;
          case 9: json["v9"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
     }
    }

    s="";
    j=1;
    k=0;
   s1=str[0];
   if (s1=="I"){
     //s2="I";
    while (j<90) {
      s1 = str[j];
      if (s1 != ";") {
        s += s1;
      }
      else {
        switch(k) {
          case 1: json["I1"]=s.toFloat();
          break;
          case 2: json["I2"]=s.toFloat();
          break;
          case 3: json["I3"]=s.toFloat();
          break;
          case 4: json["I4"]=s.toFloat();
          break;
          case 5: json["I5"]=s.toFloat();
          break;
          case 6: json["I6"]=s.toFloat();
          break;
          case 7: json["I7"]=s.toFloat();
          break;
          case 8: json["I8"]=s.toFloat();
          break;
          case 9: json["I9"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
     }
    }

    s="";
    j=1;
    k=0;
   s1=str[0];
   if (s1=="P"){
```

```
  // s2="P";
while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["P1"]=s.toFloat();
       break;
       case 2: json["P2"]=s.toFloat();
       break;
       case 3: json["P3"]=s.toFloat();
       break;
       case 4: json["P4"]=s.toFloat();
       break;
       case 5: json["P5"]=s.toFloat();
       break;
       case 6: json["P6"]=s.toFloat();
       break;
       case 7: json["P7"]=s.toFloat();
       break;
       case 8: json["P8"]=s.toFloat();
       break;
       case 9: json["P9"]=s.toFloat();
       break;
     }
     s="";
     k += 1;
   }
   j += 1;
  }
 }

 s="";
 j=1;
 k=0;
s1=str[0];
if (s1=="Q"){
  // s2="Q";
while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["Q1"]=s.toFloat();
       break;
       case 2: json["Q2"]=s.toFloat();
       break;
       case 3: json["Q3"]=s.toFloat();
       break;
       case 4: json["Q4"]=s.toFloat();
       break;
       case 5: json["Q5"]=s.toFloat();
```

```
          break;
          case 6: json["Q6"]=s.toFloat();
          break;
          case 7: json["Q7"]=s.toFloat();
          break;
          case 8: json["Q8"]=s.toFloat();
          break;
          case 9: json["Q9"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
    }
  }

  s="";
  j=1;
  k=0;
 s1=str[0];
 if (s1=="R"){
  //s2="R";
 while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["R1"]=s.toFloat();
        break;
        case 2: json["R2"]=s.toFloat();
        break;
        case 3: json["R3"]=s.toFloat();
        break;
        case 4: json["R4"]=s.toFloat();
        break;
        case 5: json["R5"]=s.toFloat();
        break;
        case 6: json["R6"]=s.toFloat();
        break;
        case 7: json["R7"]=s.toFloat();
        break;
        case 8: json["R8"]=s.toFloat();
        break;
        case 9: json["R9"]=s.toFloat();
        break;
      }
      s="";
      k += 1;
    }
    j += 1;
  }
 }
```

```
 s="";
 j=1;
 k=0;
s1=str[0];
if (s1=="S"){
  //s2="S";
 while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["S1"]=s.toFloat();
       break;
       case 2: json["S2"]=s.toFloat();
       break;
       case 3: json["S3"]=s.toFloat();
       break;
       case 4: json["S4"]=s.toFloat();
       break;
       case 5: json["S5"]=s.toFloat();
       break;
       case 6: json["S6"]=s.toFloat();
       break;
       case 7: json["S7"]=s.toFloat();
       break;
       case 8: json["S8"]=s.toFloat();
       break;
       case 9: json["S9"]=s.toFloat();
       break;
     }
     s="";
     k += 1;
   }
   j += 1;
  }
 }

 s="";
 j=1;
 k=0;
s1=str[0];
if (s1=="F"){
  // s2="F";
while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["F1"]=s.toFloat();
       break;
       case 2: json["F2"]=s.toFloat();
       break;
```

```
          case 3: json["F3"]=s.toFloat();
          break;
          case 4: json["F4"]=s.toFloat();
          break;
          case 5: json["F5"]=s.toFloat();
          break;
          case 6: json["F6"]=s.toFloat();
          break;
          case 7: json["F7"]=s.toFloat();
          break;
          case 8: json["F8"]=s.toFloat();
          break;
          case 9: json["F9"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
    }
  }

  s="";
  j=1;
  k=0;
 s1=str[0];
 if (s1=="C"){
   //s2="C";
  while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["C1"]=s.toFloat();
        break;
        case 2: json["C2"]=s.toFloat();
        break;
        case 3: json["C3"]=s.toFloat();
        break;
        case 4: json["C4"]=s.toFloat();
        break;
        case 5: json["C5"]=s.toFloat();
        break;
        case 6: json["C6"]=s.toFloat();
        break;
        case 7: json["C7"]=s.toFloat();
        break;
        case 8: json["C8"]=s.toFloat();
        break;
        case 9: json["C9"]=s.toFloat();
        break;
      }
      s="";
      k += 1;
```

```
    }
    j += 1;
   }
  }

  s="";
  j=1;
  k=0;
 s1=str[0];
 if (s1=="N"){
    // s2="N";
  while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["N1"]=s.toFloat();
        break;
        case 2: json["N2"]=s.toFloat();
        break;
        case 3: json["N3"]=s.toFloat();
        break;
        case 4: json["N4"]=s.toFloat();
        break;
        case 5: json["N5"]=s.toFloat();
        break;
        case 6: json["N6"]=s.toFloat();
        break;
        case 7: json["N7"]=s.toFloat();
        break;
        case 8: json["N8"]=s.toFloat();
        break;
        case 9: json["N9"]=s.toFloat();
        break;
      }
      s="";
      k += 1;
    }
    j += 1;
   }
  }

  s="";
  j=1;
  k=0;
 s1=str[0]+str[1]+str[2]+str[3];
 if (s1=="VyA3"){
    //s2="VyA3";
   while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
```

```
        switch(k) {
          case 1: json["VyA31"]=s.toFloat();
          break;
          case 2: json["VyA32"]=s.toFloat();
          break;
          case 3: json["VyA33"]=s.toFloat();
          break;
          case 4: json["VyA34"]=s.toFloat();
          break;
          case 5: json["VyA35"]=s.toFloat();
          break;
          case 6: json["VyA36"]=s.toFloat();
          break;
          case 7: json["VyA37"]=s.toFloat();
          break;
          case 8: json["VyA38"]=s.toFloat();
          break;
          case 9: json["VyA39"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
    }
   }

   s="";
   j=1;
   k=0;
  s1=str[0]+str[1]+str[2]+str[3];
  if (s1=="Pot3"){
     // s2="Pot3";
   while (j<90) {
     s1 = str[j];
     if (s1 != ";") {
       s += s1;
     }
     else {
       switch(k) {
         case 1: json["Pot31"]=s.toFloat();
         break;
         case 2: json["Pot32"]=s.toFloat();
         break;
         case 3: json["Pot33"]=s.toFloat();
         break;
         case 4: json["Pot34"]=s.toFloat();
         break;
         case 5: json["Pot35"]=s.toFloat();
         break;
         case 6: json["Pot36"]=s.toFloat();
         break;
         case 7: json["Pot37"]=s.toFloat();
         break;
         case 8: json["Pot38"]=s.toFloat();
         break;
```

```
          case 9: json["Pot39"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
    }
  }

  s="";
  j=1;
  k=0;
 s1=str[0];
 if (s1=="S"){
     //s2="S";
  while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["S31"]=s.toFloat();
        break;
        case 2: json["S32"]=s.toFloat();
        break;
        case 3: json["S33"]=s.toFloat();
        break;
      }
      s="";
      k += 1;
    }
    j += 1;
  }
 }

  s="";
  j=1;
  k=0;
 s1=str[0]+str[1]+str[2];
 if (s1=="Fp1"){
   //  s2="Fp1";
  while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["Fp1"]=s.toFloat();
        break;
        case 2: json["Fp2"]=s.toFloat();
        break;
        case 3: json["Fp3"]=s.toFloat();
        break;
```

```
        }
        s="";
        k += 1;
      }
      j += 1;
     }
    }

    s="";
    j=1;
    k=0;
   s1=str[0]+str[1]+str[2];
   if (s1=="Cos"){
     //  s2="Cos";
    while (j<90) {
      s1 = str[j];
      if (s1 != ";") {
        s += s1;
      }
      else {
        switch(k) {
          case 1: json["Cos1"]=s.toFloat();
          break;
          case 2: json["Cos2"]=s.toFloat();
          break;
          case 3: json["Cos3"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
     }
    }

    s="";
    j=1;
    k=0;
   s1=str[0]+str[1];
   if (s1=="QL"){
      // s2="QL";
    while (j<90) {
      s1 = str[j];
      if (s1 != ";") {
        s += s1;
      }
      else {
        switch(k) {
          case 1: json["QL1"]=s.toFloat();
          break;
          case 2: json["QL2"]=s.toFloat();
          break;
          case 3: json["QL3"]=s.toFloat();
          break;
          case 4: json["QL4"]=s.toFloat();
          break;
```

```
          case 5: json["QL5"]=s.toFloat();
          break;
          case 6: json["QL6"]=s.toFloat();
          break;
          case 7: json["QL7"]=s.toFloat();
          break;
          case 8: json["QL8"]=s.toFloat();
          break;
          case 9: json["QL9"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
    }
  }

  s="";
  j=1;
  k=0;
 s1=str[0]+str[1]+str[2]+str[3];
 if (s1=="Qcon"){
   //  s2="Qcon";
  while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["Qcon1"]=s.toFloat();
        break;
        case 2: json["Qcon2"]=s.toFloat();
        break;
        case 3: json["Qcon3"]=s.toFloat();
        break;
        case 4: json["Qcon4"]=s.toFloat();
        break;
        case 5: json["Qcon5"]=s.toFloat();
        break;
        case 6: json["Qcon6"]=s.toFloat();
        break;
        case 7: json["Qcon7"]=s.toFloat();
        break;
        case 8: json["Qcon8"]=s.toFloat();
        break;
        case 9: json["Qcon9"]=s.toFloat();
        break;
      }
      s="";
      k += 1;
    }
    j += 1;
  }
 }
```

```
 s="";
 j=1;
 k=0;
s1=str[0]+str[1]+str[2]+str[3];
if (s1="Qgen"){
  //  s2="Qgen";
 while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["Qgen1"]=s.toFloat();
       break;
       case 2: json["Qgen2"]=s.toFloat();
       break;
       case 3: json["Qgen3"]=s.toFloat();
       break;
       case 4: json["Qgen4"]=s.toFloat();
       break;
       case 5: json["Qgen5"]=s.toFloat();
       break;
       case 6: json["Qgen6"]=s.toFloat();
       break;
       case 7: json["Qgen7"]=s.toFloat();
       break;
       case 8: json["Qgen8"]=s.toFloat();
       break;
       case 9: json["Qgen9"]=s.toFloat();
       break;
     }
     s="";
     k += 1;
   }
   j += 1;
  }
 }

 s="";
 j=1;
 k=0;
s1=str[0]+str[1];
if (s1="Q3"){
    //s2="Q3";
 while (j<90) {
   s1 = str[j];
   if (s1 != ";") {
     s += s1;
   }
   else {
     switch(k) {
       case 1: json["Q31"]=s.toFloat();
       break;
       case 2: json["Q32"]=s.toFloat();
```

```
          break;
          case 3: json["Q33"]=s.toFloat();
          break;
          case 4: json["Q34"]=s.toFloat();
          break;
          case 5: json["Q35"]=s.toFloat();
          break;
          case 6: json["Q36"]=s.toFloat();
          break;
          case 7: json["Q37"]=s.toFloat();
          break;
          case 8: json["Q38"]=s.toFloat();
          break;
          case 9: json["Q39"]=s.toFloat();
          break;
        }
        s="";
        k += 1;
      }
      j += 1;
    }
  }

  s="";
  j=1;
  k=0;
 s1=str[0]+str[1];
 if (s1=="Cu"){
   //  s2="Cu";
  while (j<90) {
    s1 = str[j];
    if (s1 != ";") {
      s += s1;
    }
    else {
      switch(k) {
        case 1: json["Cu1"]=s.toFloat();
        break;
        case 2: json["Cu2"]=s.toFloat();
        break;
        case 3: json["Cu3"]=s.toFloat();
        break;
        case 4: json["Cu4"]=s.toFloat();
        break;
        case 5: json["Cu5"]=s.toFloat();
        break;
        case 6: json["Cu6"]=s.toFloat();
        break;
        case 7: json["Cu7"]=s.toFloat();
        break;
        case 8: json["Cu8"]=s.toFloat();
        break;
        case 9: json["C9"]=s.toFloat();
        break;
      }
      s="";
```

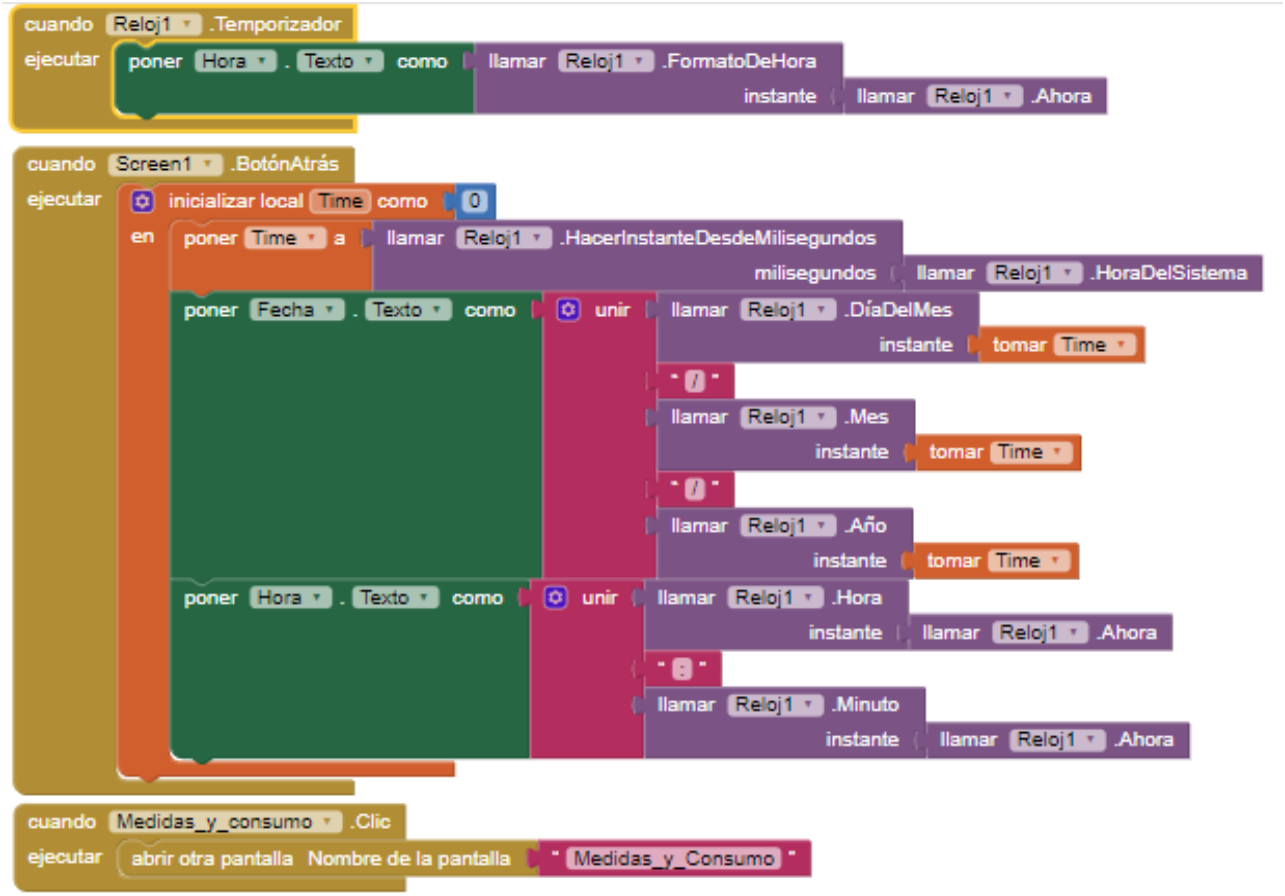
```
        k += 1;
      }
      j += 1;
     }
    }

    s="";
    j=1;
    k=0;
   s1=str[0]+str[1];
   if (s1=="CN"){
      // s2="CN";
    while (j<90) {
      s1 = str[j];
      if (s1 != ";") {
        s += s1;
      }
      else {
         json["CN"]=s.toFloat();
      }
        s="";
        k += 1;
      }
      j += 1;
     }

   s="";
    j=1;
    k=1;
  s1=str[0];


   if (s1=="t"){
    while (j<90) {
      s1 = str[j];
      if (s1 != ";") {
        s += s1;
      }
      else {
        switch(k) {
          case 1:
               json["dia"]=s.toFloat();
               dia = s;
               dian = s.toInt();
          break;
          case 2:
               json["mes"]=s.toFloat();
               mes = s;
               mesn = s.toInt();
          break;
          case 3:
               json["ano"]=s.toFloat();
               ano = s;
               anon = s.toInt();
          break;
          case 4: json["hora"]=s.toFloat();
```

```
            break;
            case 5: json["minuto"]=s.toFloat();
            break;
            case 6: json["segundo"]=s.toFloat();
            break;
            case 7:
               s2=s;
            break;
          }
          s="";
          k += 1;
        }
        j += 1;
      }


      String fecha = dia;
      fecha += "-";
      fecha += mes;
      fecha += "-";
      fecha += ano;

      Serial.print("fecha:");
      Serial.println(fecha);

      s2=s2+"/"+fecha;

      Serial.println(s2);


      Firebase.push(s2,json);
      s2="";
      // handle error
      if (Firebase.failed()) {
         Serial.print("setting /message failed:");
         Serial.println(Firebase.error());
         //return;
       }
      }
      for (int l=0; l>=99; l++) {str[l]=47;}
      unsigned long t1;
      t1 = millis();
      Serial.println(t1-t);
      if (t1-t<500) delay(500-(t1-t));
      else Serial.flush();
  }


   }

}
```

Anexo 2: Código de bloques AppInventor

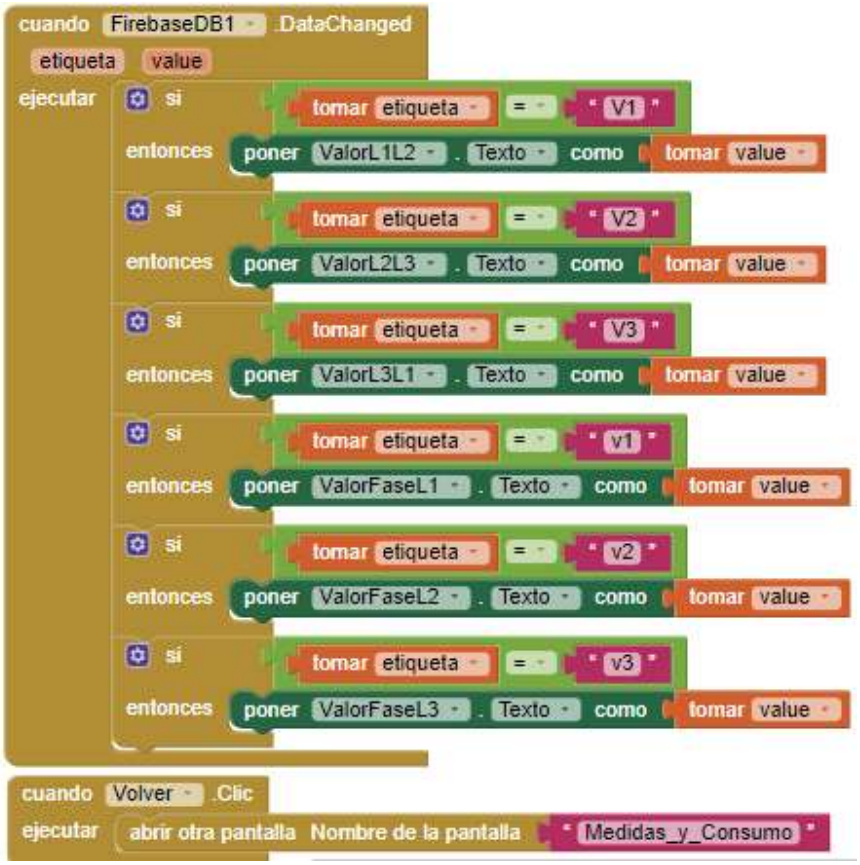
- **Screen Inicial**

  Como este bloque (reloj) será obligatorio en las demás pantallas para que nos muestre la fecha y hora actual no lo incluiremos en los siguientes apartados.

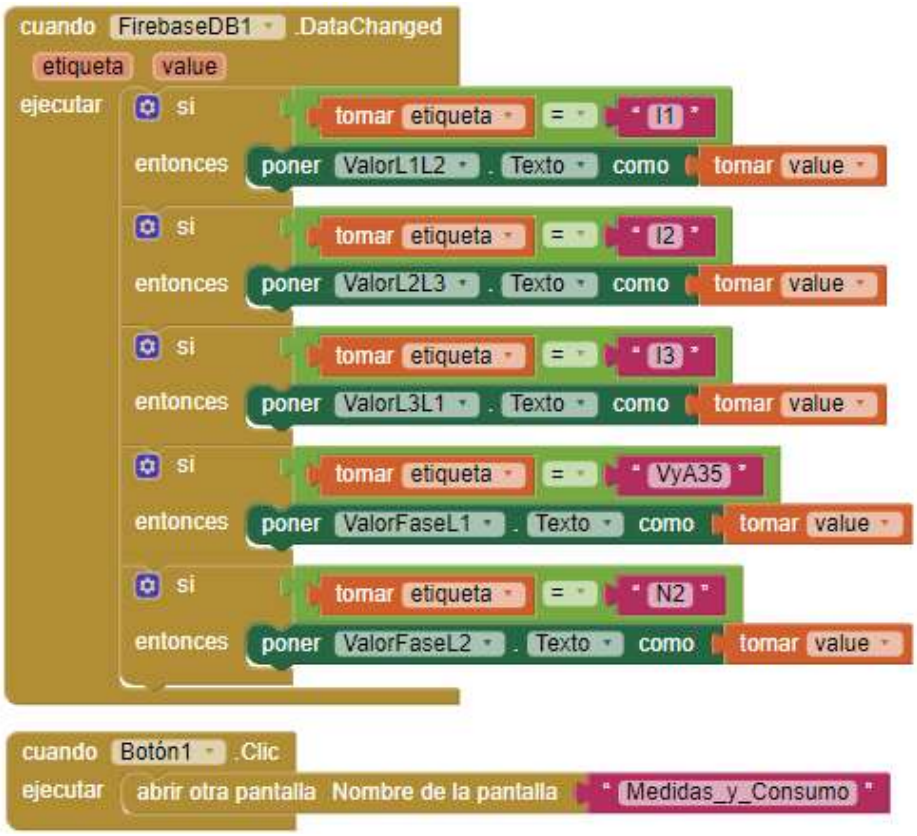


- **Screen Medidas**

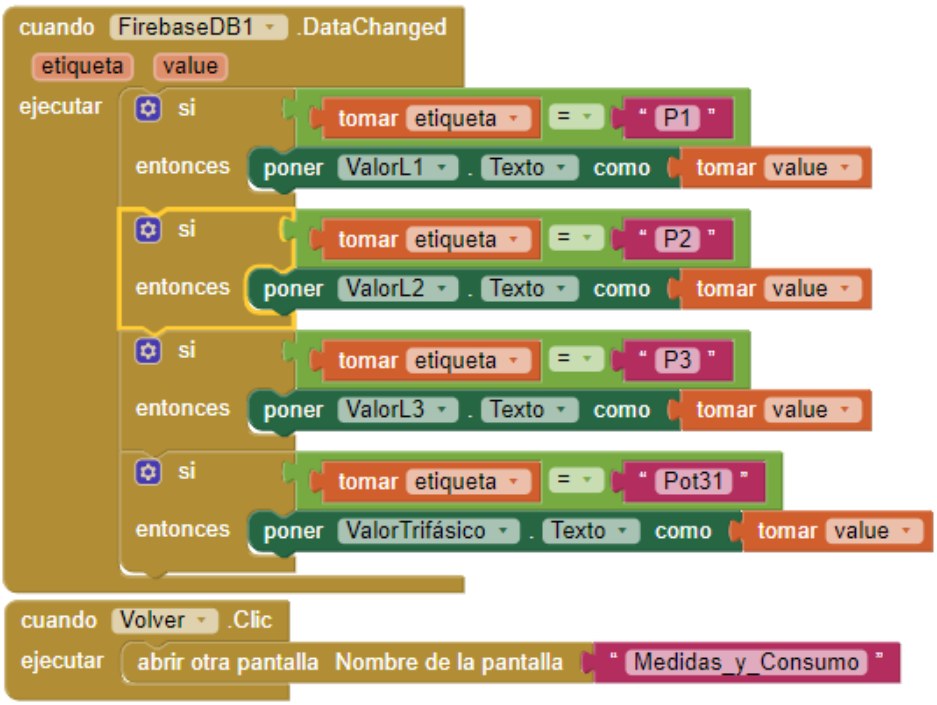
- **Screen Tensiones**



- **Screen Corrientes**



- **Screen Potencias Activas**



- **Screen Potencias Reactivas**

INTERFAZ GRÁFICA DE LA APP MÓVIL

- **Screen Inicial**

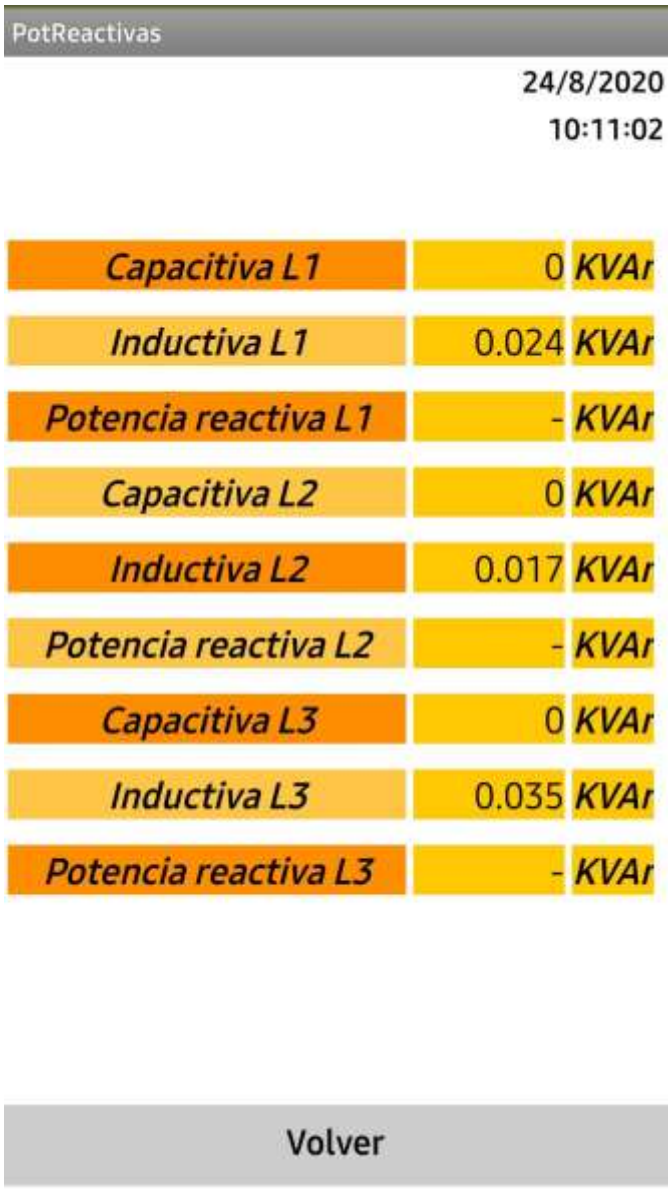
- **Screen Medidas**

- **Screen Tensiones**

- **Screen Corrientes**

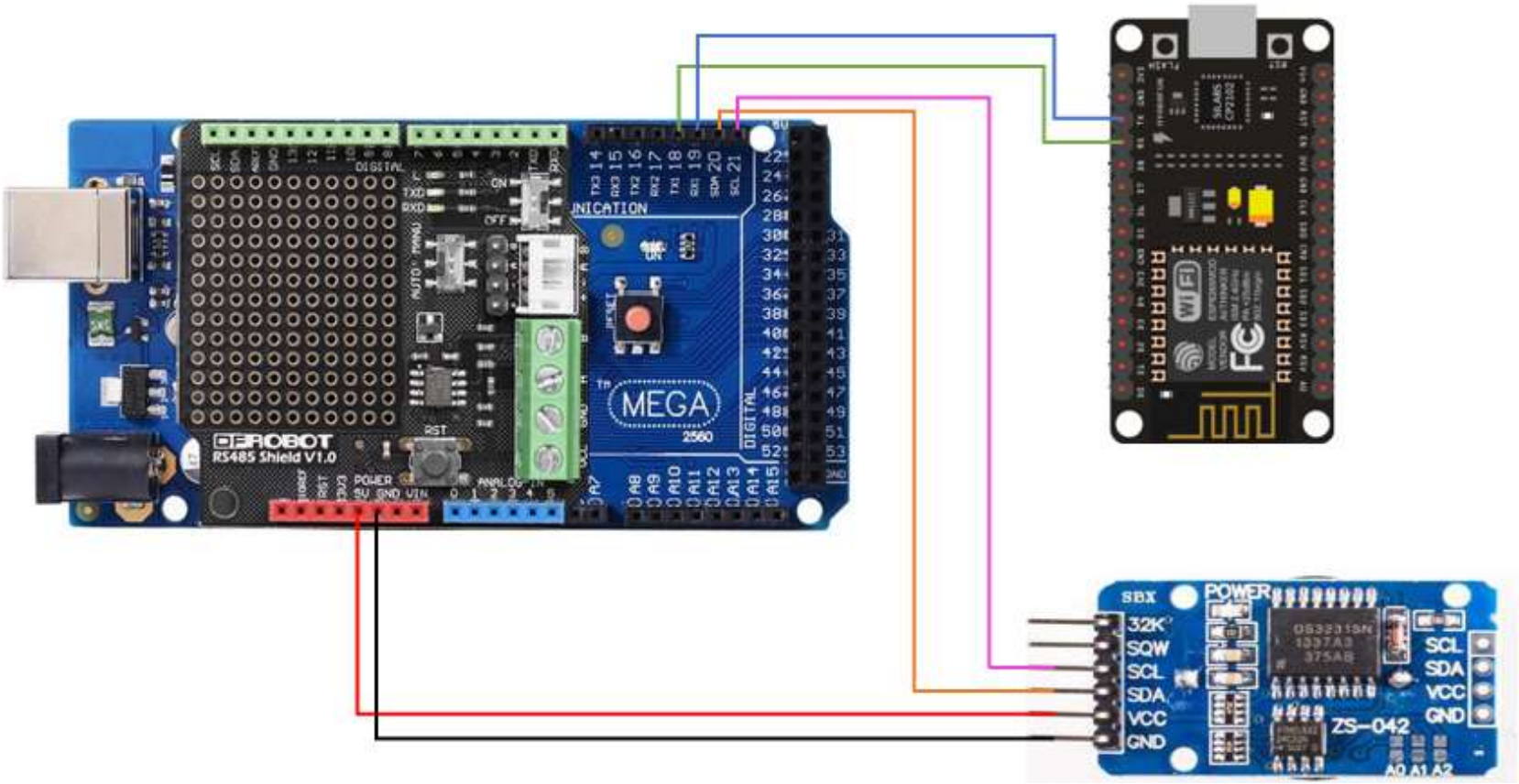
- **Screen Potencias Activas**

- **Screen Potencias Reactivas**

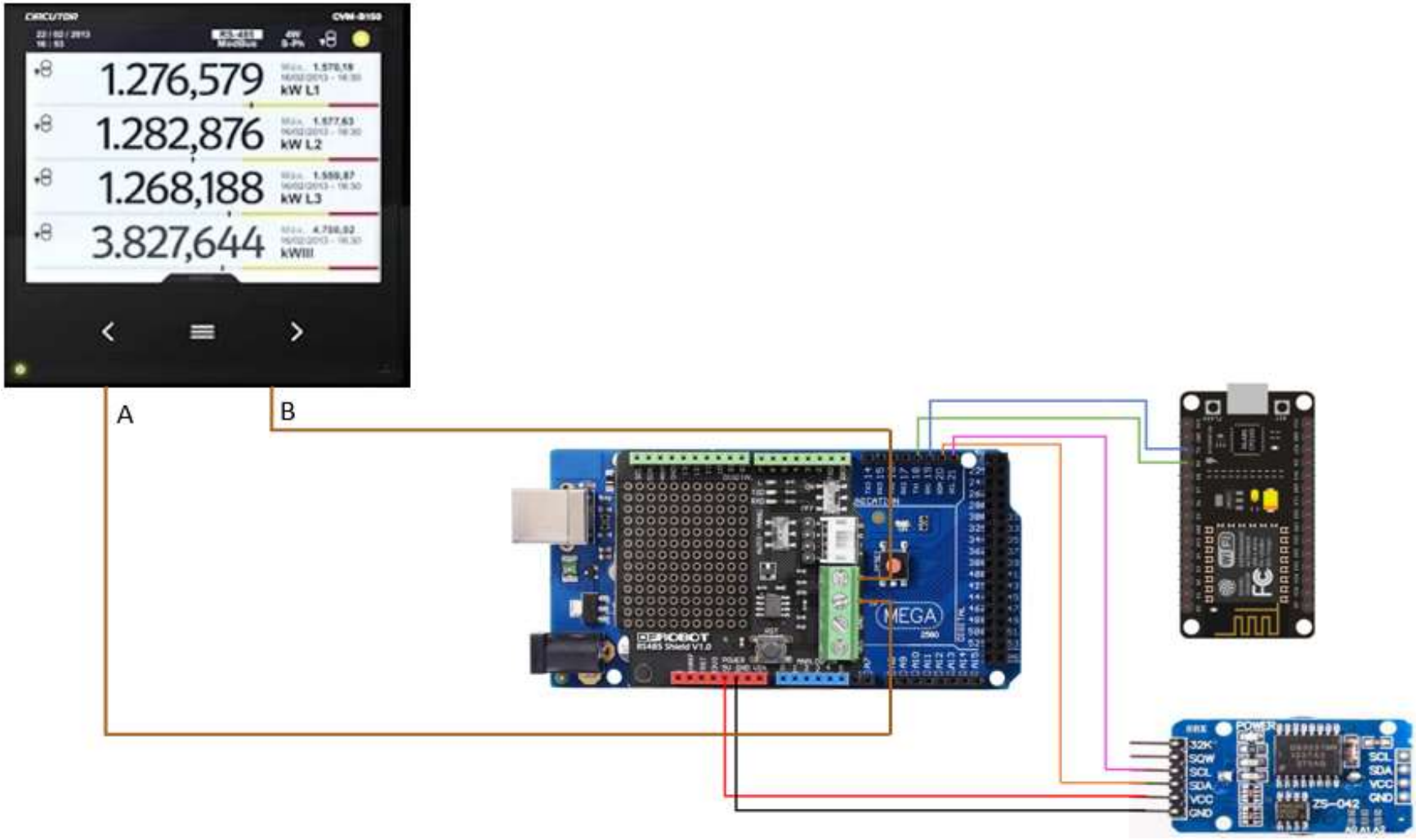


PLANOS

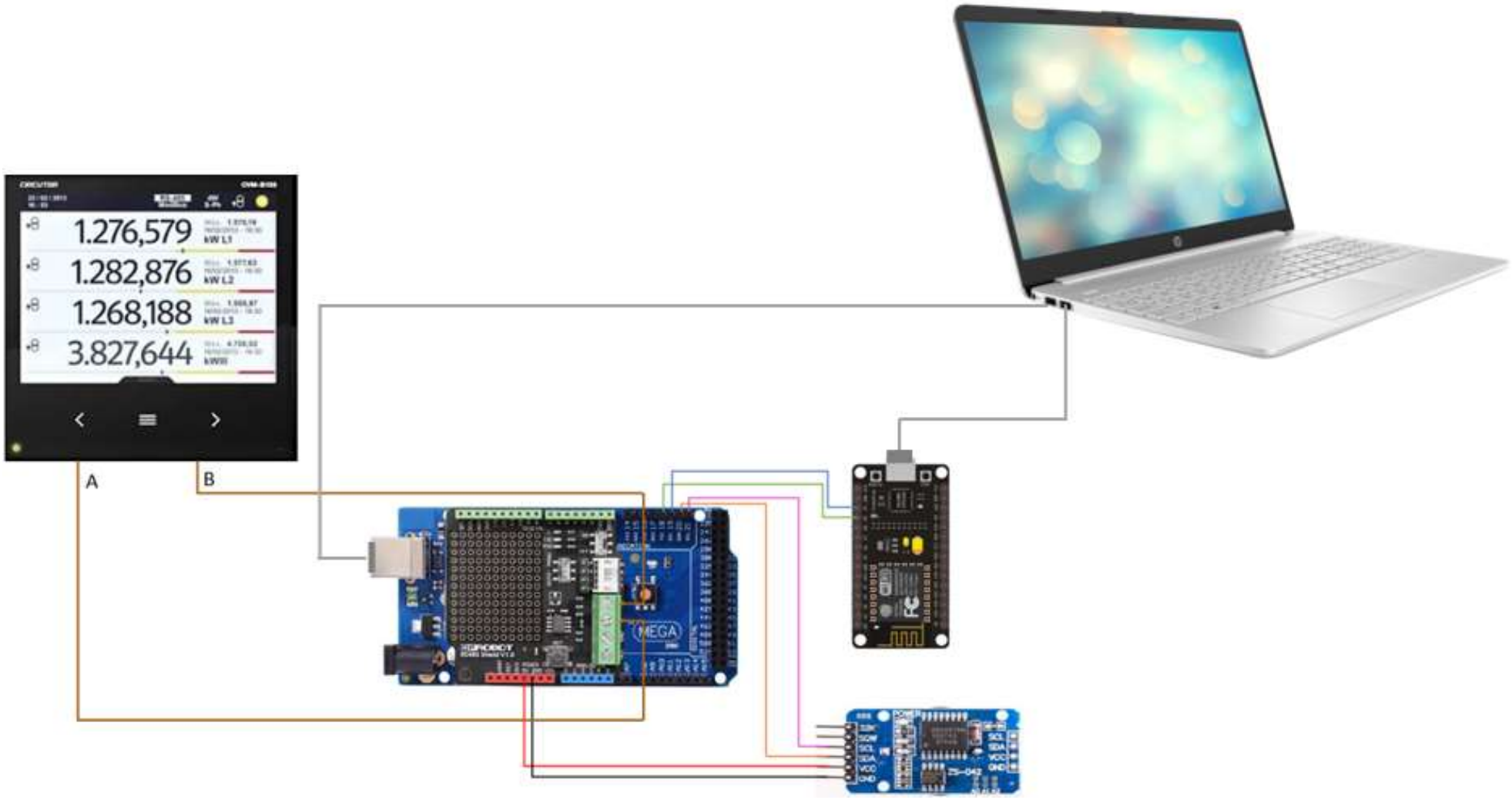
- **Conexión unidad principal**
  - *Arduino Mega / RS485 shield / NodeMCU / RTC*



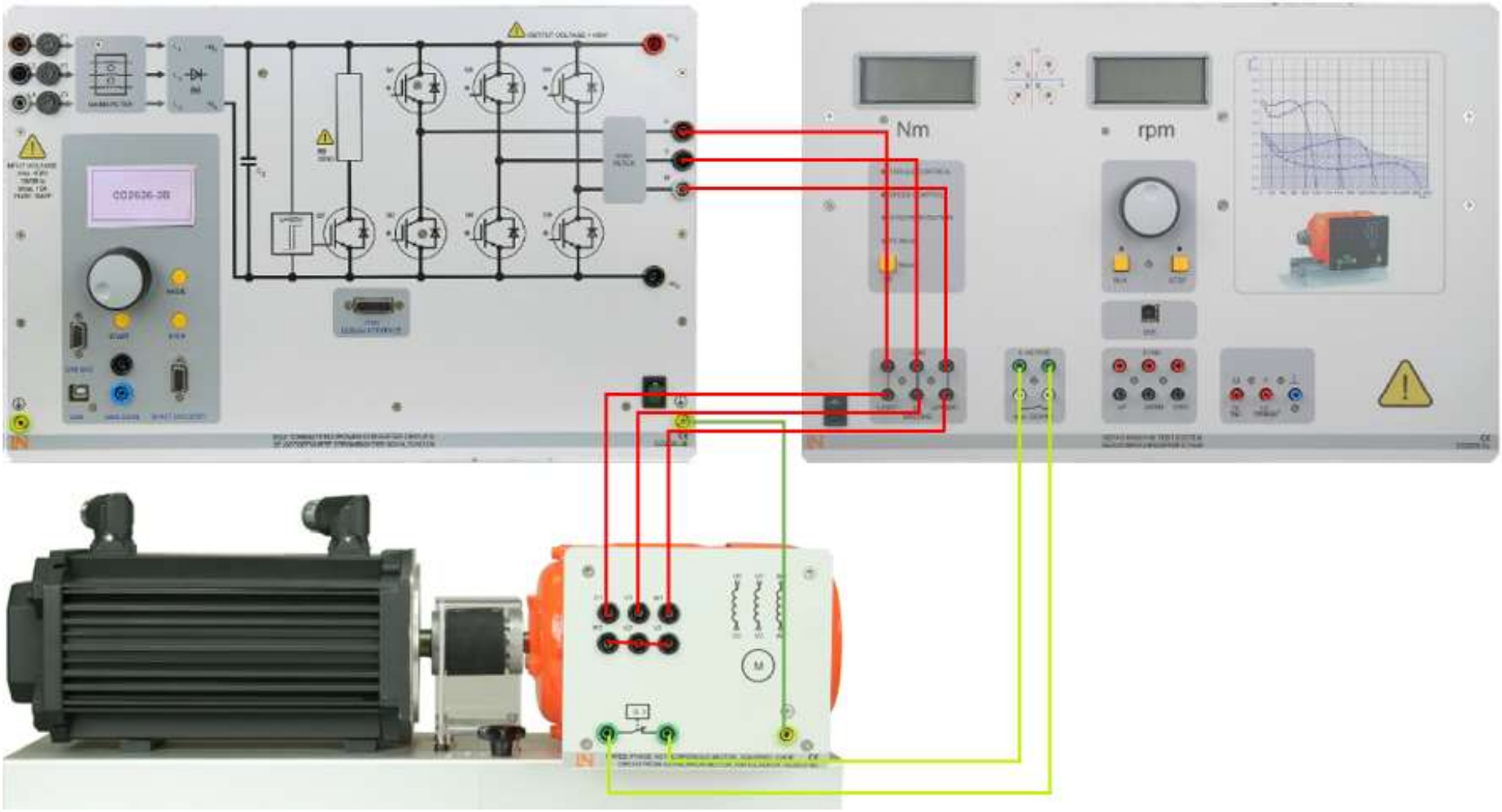
- *Arduino Mega* / *RS485 shield* / *NodeMCU* / *RTC* / CVM-B150-ITF-HF



- *Arduino Mega* / *RS485 shield* / *NodeMCU* / *RTC* / CVM-B150-ITF-HF / Pc

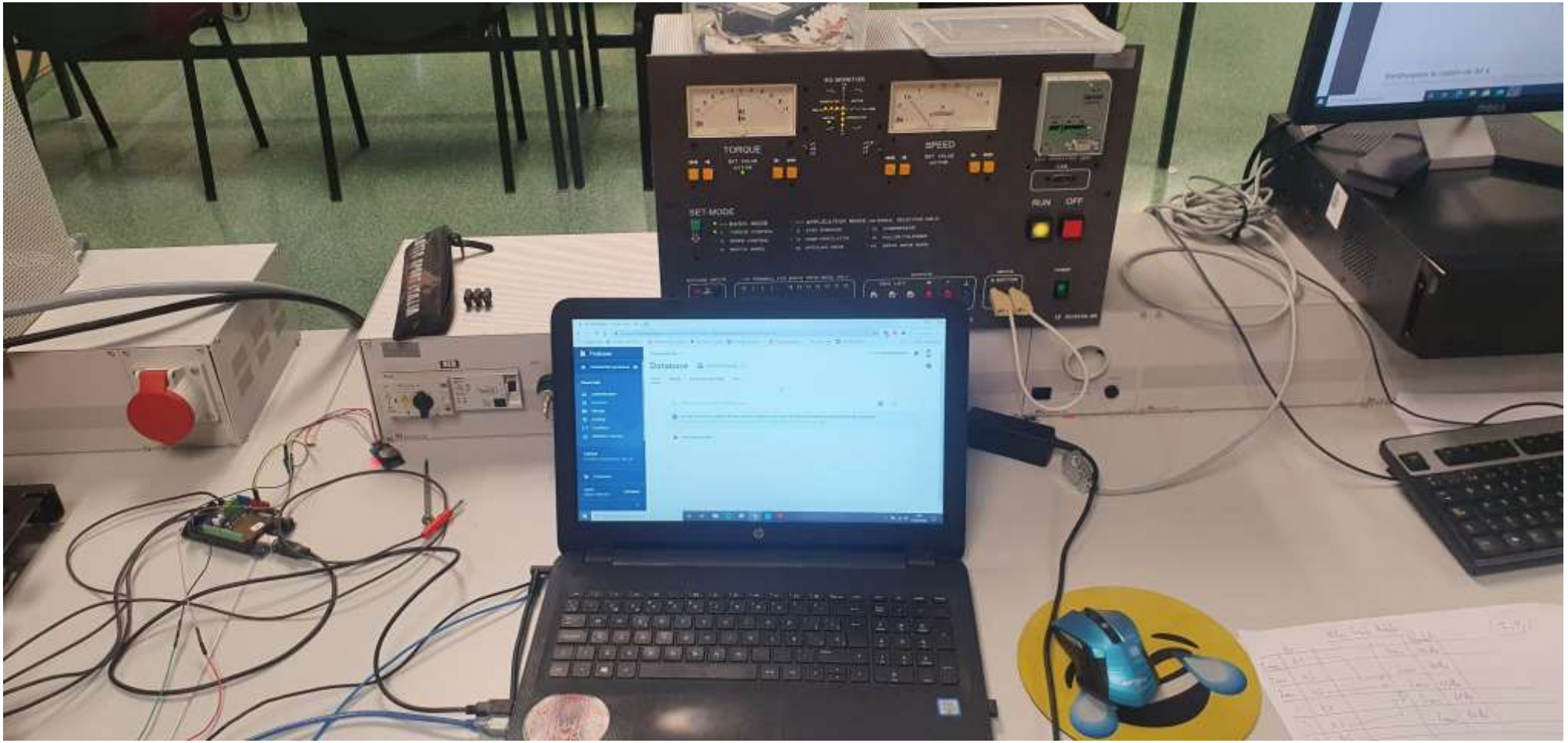


- **Conexión de equipos**
  - *Motor jaula de ardilla / Circuitos convertidores autoconmutados con IGBT / Variador de momento y velocidad*



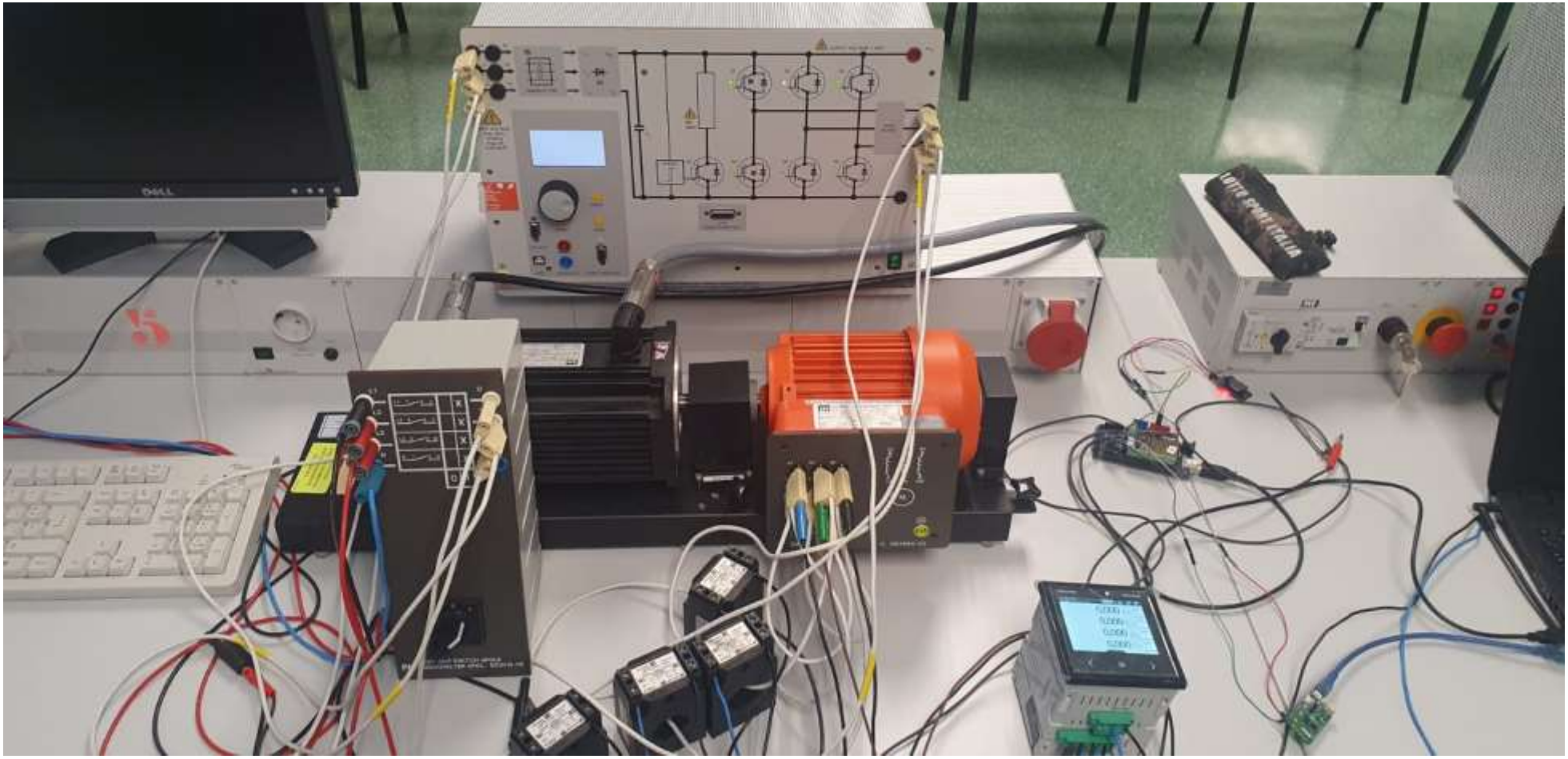
IMÁGENES DEL ENSAYO PRÁCTICO

- *Conexionado arduino / Entrenador de máquinas eléctricas / Pc*



- *Interruptor de corte y protección / Jaula de ardilla / Transformadores de intensidad 100/5 / CVM-B150-ITF-HF / Conexionado arduino*

- *Interruptor de corte y protección / Jaula de ardilla / Transformadores de intensidad 100/5 / CVM-B150-ITF-HF / Conexionado arduino / Circuitos convertidores autoconmutados con IGBT*



- *Interruptor de corte y protección / Jaula de ardilla / Transformadores de intensidad 100/5 / CVM-B150-ITF-HF / Conexionado arduino / Conexionado de equipos / Pc*