UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA DE LA ENERGÍA

DISEÑO Y MONTAJE DE UN ANALIZADOR DE REDES TRIFÁSICO MEDIANTE UN MICROCONTROLADOR ARDUINO, PROGRAMACIÓN DE UNA HMI Y COMUNICACIÓN DEL SISTEMA MEDIANTE MODBUS - APLICACIÓN A UNA MICRORRED BASADA EN ENERGÍAS RENOVABLES

AUTOR: RAÚL MAÍCAS MUÑOZ

TUTOR: CARLOS AFRANIO VARGAS SALGADO

Curso Académico: 2016-17

Agradecimientos

A mi tutor, por todos los conocimientos que me ha sabido transmitir, y por la ayuda, resolución de dudas y apoyo que me ha transmitido en todo momento durante la realización de este trabajo.

A mi familia, por estar siempre ahí, por no permitir que me rinda nunca, y por el esfuerzo económico que han realizado para que haya podido llegar hasta aquí.

A mis amigos, por tenerlos a mi lado, tanto en los buenos como en los malos momentos.

Resumen

Este trabajo tiene como objetivo el diseño, la programación y el montaje de un analizador de redes trifásico a un precio competitivo. Éste se ha integrado en la microrred del laboratorio de energías renovables LabDER, de manera que los valores medidos por el mismo se integren al sistema de automatización y control de la microrred utilizando el protocolo Modbus. También se detalla la programación de un terminal táctil que permite visualizar, no sólo los valores medidos por este analizador, sino también todos los parámetros energéticos que están siendo recogidos en el laboratorio. Además, se programa el autómata para que éste pueda comunicarse y obtener información del analizador de redes, e intercambiar datos con el terminal táctil. El documento se divide en tres partes:

En la primera parte, se describe el desarrollo completo de un analizador trifásico, desde el montaje de sus componentes físicos hasta el desarrollo del software necesario para su funcionamiento. Este analizador consta de una placa Arduino, un módulo Ethernet para posibilitar la comunicación con el sistema de automatización de la microrred y una pantalla LCD local para mostrar los valores medidos.

La segunda parte se centra en la programación de un terminal táctil HMI, cuya función será la monitorización de una manera visual de todos los valores relevantes que están siendo medidos en tiempo real en la microrred. Esto es posible gracias a la conexión de este terminal a los dos autómatas programables que forman parte de la red local, los cuales contienen toda la información de los sensores y medidores instalados en la microrred.

La tercera parte relaciona las dos partes realizadas previamente. Este último apartado se centra en la conexión del analizador mediante el protocolo Modbus a uno de los PLCs del laboratorio, para posteriormente poder mostrar los valores medidos en el terminal táctil programado previamente.

Resum

Este treball té com a objectiu el disseny, la programació i el muntatge d'un analitzador de xarxes trifàsic a un preu competitiu. Este s'ha integrat en la microrred del laboratori d'energies renovables LabDER, de manera que els valors mesurats pel mateix s'integren al sistema d'automatització i control de la microrred utilitzant el protocol Modbus. També es detalla la programació d'un terminal tàctil que permet visualitzar no sols els valors mesurats per este analitzador, sinó també tots els paràmetres energètics que estan sent arreplegats en el laboratori. A més, es programa l'autòmat perquè este puga comunicar-se i obtindre informació de l'analitzador de xarxes, i intercanviar dades amb el terminal tàctil. El document es dividix en tres parts:

En la primera part, es descriu el desenrotllament complet d'un analitzador trifàsic, des del muntatge dels seus components físics fins al desenrotllament del programari necessari per al seu funcionament. Este analitzador consta d'una placa Arduino, un mòdul Ethernet per a possibilitar la comunicació amb el sistema d'automatització de la microrred i una pantalla LCD local per a mostrar els valors mesurats.

La segona part se centra en la programació d'un terminal tàctil HMI, la funció de la qual serà la monitorització d'una manera visual de tots els valors rellevants que estan sent mesurats en temps real en la microrred. Açò és possible gràcies a la connexió d'este terminal als dos autòmats programables que formen part de la xarxa local, els quals contenen tota la informació dels sensors i mesuradors installats en la microrred.

La tercera part relaciona les dos parts realitzades prèviament. Este últim apartat se centra en la connexió de l'analitzador per mitjà del protocol Modbus a un dels PLCs del laboratori, per a posteriorment poder mostrar els valors mesurats en el terminal tàctil programat prèviament.

Abstract

This project has the objective the design, programming and mounting of a three-phase analyzer at a competitive price. This analyzer has been integrated into the renewable energy based microgrid of the LabDER laboratory, so values that have been measured by it could be added to the laboratory automation system using Modbus protocol. Furthermore, it is described the programming of a touch screen that allows to see not only values measured by this analyzer, but also all energy parameters that are being picked up in the laboratory. Moreover, the PLC is programmed in order to communicate and get information from the analyzer, and exchange data with the touch screen. The document is divided in three parts:

In the first one, it is described the complete development of a three-phase analyzer, from the assembly of its physical parts to the development of the software that allows its correct functioning. This analyzer consists of an Arduino board, an Ethernet module for communication with the microgrid automation system and a LCD local screen that shows measured values.

The second part is focused on the programming of a HMI touch screen, whose principal function is the monitoring, in a visual manner, of all relevant values that are being measured live from the microgrid. This is possible thanks to the connection among this touch screen and the two PLCs that are included in the local network, which contain all information about sensors and meters that are installed on the microgrid.

The third part has the objective of linking the two previous parts. This last section is focused on the connection of the analyzer through Modbus protocol to one of the PLCs of the laboratory. So, after that, we could show the measured values on the touch screen that was programmed previously.

Índice

Capítulo 1 - Introducción

Antecedentes

Descripción general

Este apartado se va a centrar en explicar el concepto de microrred, cómo se utiliza en el laboratorio de investigación de energías renovables LabDER de la Universidad Politécnica de Valencia y cómo se diseñan los sistemas de automatización y control en una microrred de estas características. Además, se incluye una breve introducción a la plataforma Arduino, dado que es parte fundamental del analizador a realizar.

Generación distribuida y microrredes

El principal problema de las energías renovables es la incertidumbre que se tiene sobre características clave en la generación: no se conoce con exactitud la cantidad de energía que se va a generar, ni durante cuánto tiempo. Esta es una de las mayores dificultades para cambiar el modelo energético actual basado en combustibles fósiles, pues es más complicado generar lo requerido por la demanda en cada momento.

La generación distribuida tiene como concepto pasar de tener grandes plantas de producción de energía, inmersas en grandes infraestructuras de transporte; a crear un número más elevado de centros de generación, con una menor potencia nominal, pero con una menor distancia entre generación y consumo.

Para operar en un sistema de estas características, es necesario crear una microrred, compuesta por el sistema de generación local y la carga o consumo. Aun así, existen problemas asociados a alimentar una microrred con una única fuente de energía: si ésta no es capaz de generar, se debería tener una gran capacidad de almacenamiento, o la carga quedaría desabastecida.

Dado que, hoy en día, el almacenamiento de energía en grandes cantidades no es ni barato ni eficiente, se han intentado buscar otras soluciones.

Hybrid Renewable Energy System (HRES)

El objetivo de un sistema híbrido es paliar los defectos que puedan tener sus componentes por separado. En este ámbito, un sistema híbrido renovable tiene como objetivo la alimentación de una carga mediante la combinación de dos o más fuentes de energía renovables (1). De este modo, se pretende llegar a alimentar a la carga de manera ininterrumpida, poniendo a funcionar una u otra fuente según las condiciones de cada momento.

Esto se consigue mediante la combinación de dos o más fuentes de energía diferentes. Las más utilizadas son la eólica, la solar fotovoltaica, la pila de hidrógeno y la biomasa.

Al crear una microrred aislada de estas características, se cambia el paradigma de cómo se entiende el modelo de producción y consumo de energía en la actualidad. Las infraestructuras de generación pasan de ser parte de grandes empresas, a ser propiedad del usuario final.

Esto conlleva como principal inconveniente la elevada inversión inicial que se necesita realizar para el sistema de generación, pero a cambio se deja de pagar una mensualidad a una empresa por el término de potencia, término de energía y alquiler de equipos de medida.

Ventajas frente al modelo clásico de producción en centrales

- Se elimina la inversión y mantenimiento en infraestructuras de transporte, pues ya no son necesarias.
- Se eliminan todas las pérdidas de potencia asociadas a la transformación de baja a alta tensión y viceversa, al trabajarse a una tensión intermedia, y se eliminan también todas las pérdidas de transporte de la potencia.
- Se adapta mejor la generación al consumo, al diseñarse específicamente la microrred para abastecer a un tipo de carga, ya sea industrial, doméstica o de servicios.
- Se reducen las emisiones contaminantes, debido al tipo de energías utilizadas.
- Se elimina la dependencia de los combustibles fósiles

Inconvenientes de este nuevo modelo

- Requiere una alta inversión inicial, que dependerá de la potencia que debe suministrar la microrred.
- Requiere un estudio personalizado de la carga que se va a alimentar.
- Requiere un complejo sistema de automatización y control, para que todos los elementos funcionen correctamente de manera conjunta
- El mantenimiento es más costoso, debido a que aumenta la complejidad del sistema de generación

Sistema de adquisición y control de datos SCADA

SCADA es una arquitectura de control que utiliza ordenadores, PLCs e interfaces gráficas para la monitorización y control de procesos industriales. Son sistemas que permiten la gestión automática de dichos procesos, junto con funciones de monitorización para visualizar el estado general del sistema.

Existen varios niveles de monitorización y control del sistema, teniendo elementos que interactúan directamente con el proceso industrial y otros que se encargan del control de los mismos. Los elementos más representativos de un sistema SCADA son los siguientes:

- Remote terminal units (RTUs): son aquellos elementos conectados al sistema cuya función reside en interactuar con el entorno físico en el que están instalados. Existen tanto RTUs de entrada (sensores), como RTUs de salida (actuadores). Estos elementos están comunicados con los PLC, de manera que puedan enviar o recibir información de los mismos. Un ejemplo de RTU de entrada podría ser un termómetro que envíe información sobre la temperatura medida al PLC, y un RTU de salida podría ser un relé que, tras una señal del PLC, abra o cierre un circuito.
- Programmable logic controllers (PLCs): son dispositivos que actúan como controladores de los RTU. Son capaces de recibir información de los RTU, almacenarla en memoria y ejecutar acciones en función de la información recibida. El tratamiento que se da a esta información es configurable pues, como su nombre indica, se pueden programar para adaptar su funcionamiento a las necesidades del proceso.
- Ordenador de supervisión: es el nivel superior de un sistema SCADA. Es el controlador que recibe toda la información del proceso y el lugar donde se ejecuta el programa

principal, que comanda al resto de equipos. Está conectado a todos los PLC del sistema, enviando y recibiendo señales de los mismos para realizar el control integral del proceso.

- Human-Machine Interface (HMI): es la interfaz de comunicación entre el sistema SCADA y los usuarios del mismo. Es una interfaz de comunicación bidireccional, pues es el elemento que se encarga de mostrar la información del sistema al usuario, pero también existe la posibilidad de comunicación desde el usuario hacia el sistema. Por tanto, un HMI puede ser tanto una pantalla que muestre los datos más relevantes del SCADA, como un botón que dispare una acción dentro del sistema.
- Infraestructura de comunicación: es el modo de conexión de los equipos en el sistema. Para que dos equipos se comuniquen, es necesario que empleen el mismo protocolo de comunicación, aunque no es necesario que se utilice el mismo protocolo en la totalidad del sistema. Hay diferentes formas de conexión entre equipos, desde la conexión serie, que permite conectar directamente dos dispositivos, hasta la conexión por Ethernet, mediante la cual se crea una red local, en la que se asigna una IP a cada elemento, lo cual permite a cada unidad intercambiar información con cualquier equipo de la red.

Habiendo explicado las características de los elementos principales que componen un sistema SCADA, se deduce que el núcleo de control está en los PLC, al ser los dispositivos que ejecutan los programas de automatización del sistema. La programación de los mismos se realiza mediante el lenguaje Ladder, un lenguaje de programación gráfico basado en diagramas de contactos que, mediante elementos y operaciones lógicas (booleanas), es capaz de tomar decisiones y ejecutar acciones en el sistema.

A pesar de que este tipo de sistemas son aplicables a multitud de escenarios de aplicación, tales como un sistema de regulación de la temperatura en una sala, o un proceso industrial de producción, se van a detallar las ventajas del SCADA en el ámbito que nos ocupa, que en este caso es la gestión de una microrred.

Descripción de la microrred del LabDER

El LabDER es un centro de investigación de la Universidad Politécnica de Valencia cuyo objetivo es la creación de un HRES (Hybrid Renewable Energy System) que sea capaz de alimentar una carga mediante la combinación de varias fuentes de energía renovables (eólica, solar y biomasa), y apoyado por un sistema de almacenamiento de energía basado en baterías recargables (2).

Además, dispone de un sistema SCADA como el descrito en el apartado anterior, formado por un servidor central, dos PLC, múltiples RTU y varios sistemas HMI para monitorizar y controlar la microrred. Este sistema de control está programado para alimentar una carga de manera ininterrumpida utilizando en la medida de lo posible energías renovables, para evitar tomar energía de la red eléctrica e intentar conseguir una estructura de generación lo más aislada posible.

Una vez se ha descrito de manera general la composición y objetivos de esta microrred, se procede a detallar los elementos que la integran. El esquema de la instalación aparece en la siguiente figura:

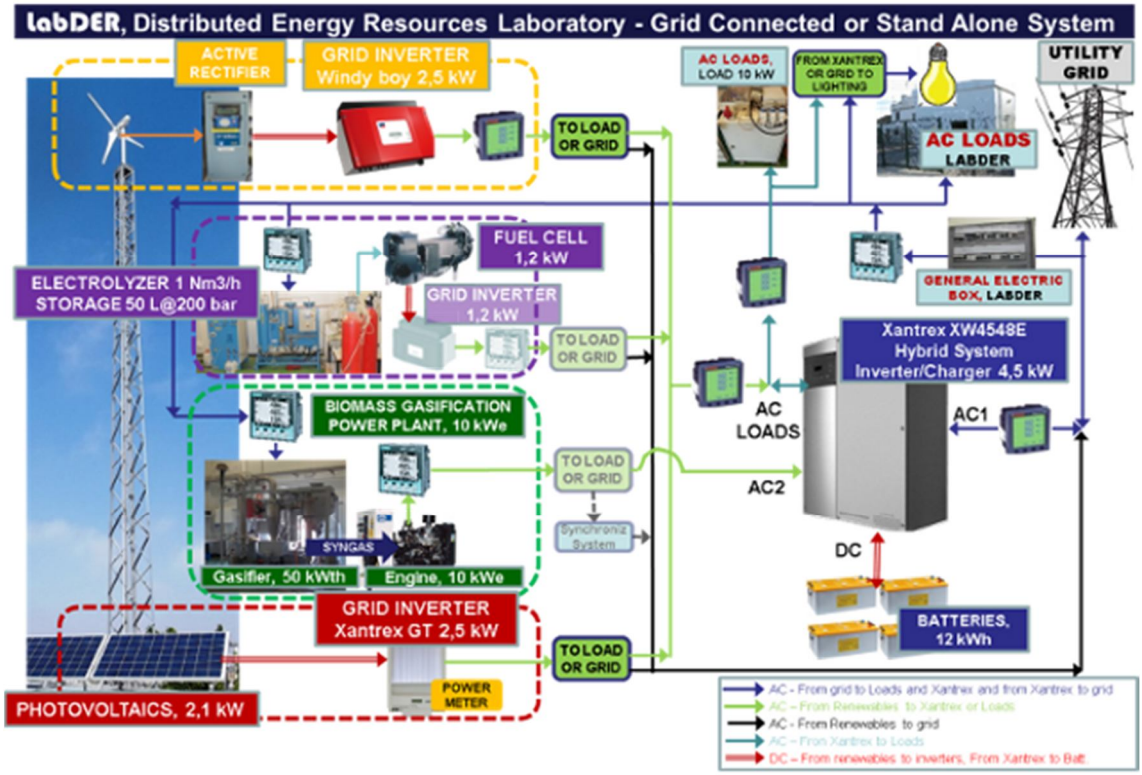


Figura 1.1: Esquema general de la microrred del LabDER

Como se puede observar, se trata de un sistema conectado a red, que capta energía de la misma cuando la generación producida por los generadores de la microrred no es suficiente para cubrir la demanda impuesta por las cargas.

El equipo que se encarga de conectar, sincronizar y desconectar la microrred a la red eléctrica es un Xantrex XW4548E. Es un equipo muy completo, que además dispone de un inversor para cargar las baterías o para obtener energía de las mismas.

La generación en la microrred está compuesta por 4 sistemas:

- Paneles fotovoltaicos: se dispone de un conjunto de paneles en el tejado del laboratorio, que captan energía de la radiación solar mediante el efecto fotoeléctrico. La corriente continua generada se transforma en alterna en el inversor Xantrex GT, para posibilitar su conexión a la microrred.
- Aerogenerador: aprovecha la velocidad del viento para generar energía eléctrica en forma de corriente alterna. Esta corriente aún no es apta para inyectarla a la red, por lo que debe pasar primero por un rectificador y luego por un inversor, para generar una onda alterna con la forma adecuada.
- Planta de gasificación de biomasa: hace pasar a los pellets de biomasa por un proceso de gasificación, en el cual se libera el poder calorífico de la biomasa en forma de gases combustibles (el llamado gas de síntesis), el cual se quema después en un motor para generar energía eléctrica.
- Generador auxiliar: se utiliza este generador cuando se quiere utilizar la microrred como un sistema aislado, es decir, se compensa el déficit de generación con el motor de combustión interna, en lugar de cubrir la demanda a partir de energía de la red eléctrica.

Quedan por nombrar los elementos que componen las cargas del laboratorio, siendo utilizados para este fin tanto el sistema de iluminación del laboratorio como un radiador eléctrico. Estas cargas, al ser regulables, permiten realizar pruebas con curvas de carga reales, para así obtener información más precisa acerca del comportamiento de la microrred en un escenario más práctico.

Regulación de la microrred mediante un sistema SCADA

El control y monitorización de la microrred, tal y como se ha comentado anteriormente, están gestionados por un sistema SCADA, cuyos elementos y sistema de conexión se describen a continuación.

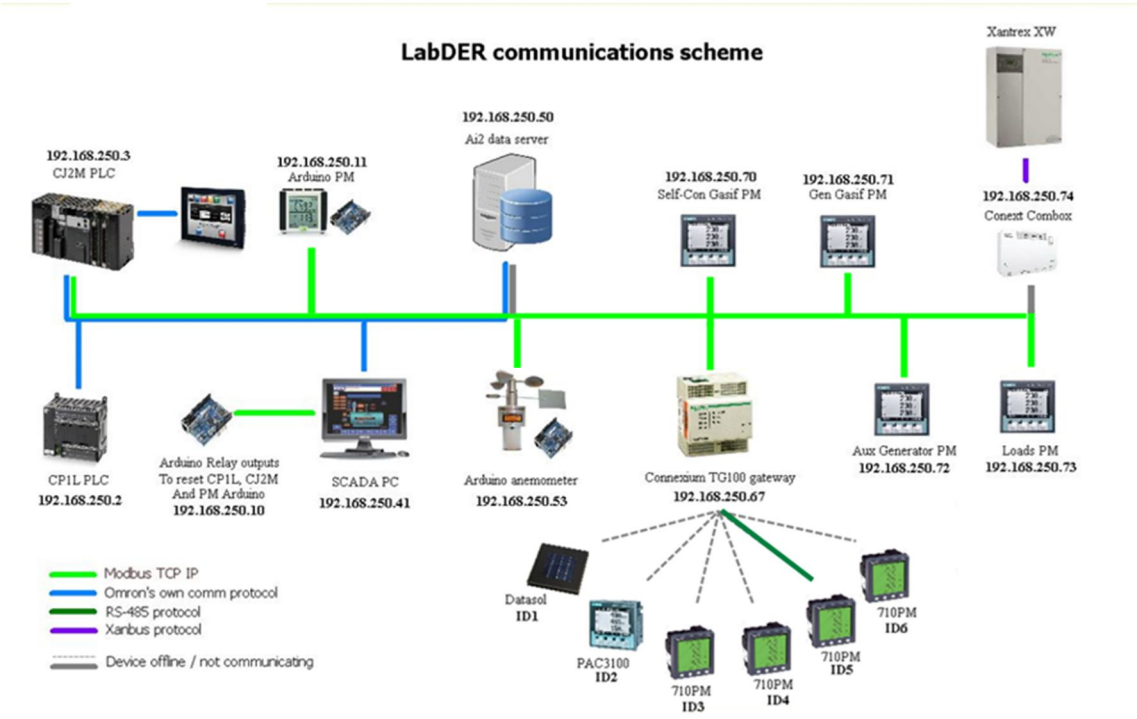


Figura 1.2: Esquema de comunicaciones del LabDER

El esquema muestra una gran cantidad de equipos conectados entre sí. Se va a dividir estos dispositivos en las categorías vistas anteriormente, según el nivel de control que poseen:

- RTUs: son los dispositivos en contacto directo con las variables a analizar, cuya función es recoger esa información y enviarla a un PLC. En esta categoría entran todos los termómetros, analizadores de redes y otros sensores instalados en la microrred.
- PLCs: son los elementos que intercambian información con los RTUs, recibiendo variables de los mismos o comandando acciones, actuando según el programa de automatización instalado en su memoria. En el laboratorio se dispone de dos PLCs instalados:
    - OMRON CJ2M: recoge información de los analizadores colocados en los elementos de generación y consumo del laboratorio, pero también de los termómetros, piranómetros y flujómetros equipados en los diferentes sistemas del laboratorio.

    - OMRON CP1L: recibe información de la placa Arduino que se utiliza, junto con un ánemometro, para registrar la velocidad del viento.
- Ordenador de supervisión: se tiene un servidor central que almacena toda la información recogida por los PLCs en una base de datos. Este equipo también funciona como HMI, lo cual se describe en el siguiente punto.
- HMI: el propio ordenador de supervisión incorpora un software que permite visualizar y controlar los valores que se están midiendo en el laboratorio. Además, también se dispone de una página web, accesible desde cualquier dispositivo, que funciona como HMI, conectándose a la base de datos almacenada en el servidor para mostrar la información deseada por pantalla.
- Infraestructura de comunicación: los equipos instalados en el laboratorio utilizan diversos protocolos de comunicación. Para las conexiones entre equipos de la marca OMRON, se utiliza el protocolo propio de la marca, pues ahorra problemas de conexión y asegura un buen funcionamiento. Sin embargo, para el resto de dispositivos se utiliza un protocolo más flexible y estandarizado: se trata de Modbus TCP/IP, un protocolo que trabaja sobre la conexión Ethernet y permite el intercambio de información con cualquier equipo de la red local. Por último, queda nombrar el protocolo RS-485, que posibilita la conexión serie entre dos equipos.

Estos elementos son los que, al funcionar entre sí, posibilitan el control automático de la microrred del laboratorio. Además, el comportamiento de este sistema es totalmente configurable a través de la programación del mismo, por lo que se puede adaptar a cualquier escenario de uso.

Plataforma Arduino

Arduino es una plataforma basada en una placa controladora, la cual se puede programar mediante el software Arduino IDE. A esta placa se pueden conectar una gran cantidad de componentes, tanto sensores como actuadores, para ampliar sus funcionalidades y adaptarse a la aplicación requerida.

El software Arduino IDE permite la creación de *sketch* en lenguaje de programación C (3). Un *sketch* está dividido en dos funciones principales, una de configuración y otra de ejecución, dentro de las cuales se ejecutan las instrucciones que darán funcionalidad a la placa.

Al ser una plataforma de código abierto, no es necesario el desarrollo de algunas funciones, pues muchas de ellas ya han sido programadas y subidas a Internet por otros usuarios del sistema. Así, no es necesario escribir un programa para conectarse por Ethernet, o en el caso que nos ocupa, para analizar las ondas de tensión y de corriente. Estas funcionalidades se incluyen en librerías, que los autores comparten libremente, ahorrando una gran cantidad de tiempo al resto de programadores de la plataforma Arduino.

Objetivo

El objetivo de este trabajo de fin de grado es el diseño, programación y montaje de un analizador de redes trifásico de bajo coste utilizando la plataforma Arduino. Este analizador contará con comunicaciones Modbus, mediante las cuales se integrará a la microrred del LabDER. Además, se programará un terminal táctil desde el cual monitorizar todas las mediciones del laboratorio en tiempo real. Por último, se programará el PLC para que recoja los valores medidos por el analizador y de esta manera centralizar toda la información.

El analizador trifásico que se pretende crear está basado en una placa programable Arduino, para la cual se desarrollará un software específico que permitirá que los sensores conectados trabajen como un analizador trifásico. El apartado de hardware también será creado desde cero: desde el diseño del circuito hasta la construcción y montaje de todos los componentes. Los datos recogidos por el analizador se mostrarán en una pantalla integrada, además de enviarse por Ethernet a la red del laboratorio.

Por otro lado, el terminal táctil HMI será un NB de OMRON en el cual se programará una aplicación con varias pantallas, en la que cada pantalla tendrá la descripción de un sistema diferente del laboratorio: meteorología, baterías, comunicaciones y gasificación. En cada sistema, se mostrarán valores relativos a la energía y potencia intercambiados, así como parámetros más específicos como voltajes y corrientes de fase y línea, factor de potencia y frecuencia.

Justificación

El LabDER, como centro de investigación, es una entidad que se desarrolla día a día, por lo que el presente Trabajo de Fin de Grado se enmarca como una parte del desarrollo del mismo. Esto significa que se parte de conceptos y resultados ya obtenidos en fases anteriores, y se utilizará este documento como apoyo para próximas investigaciones y mejoras. En este caso, debido a que el alcance del proyecto es limitado, se van a realizar únicamente las mejoras comentadas durante el apartado anterior.

La creación del analizador trifásico con Arduino está justificada por la necesidad de optimizar los costes de adquisición, pues es una de las medidas para hacer la microrred más competitiva, ya que los equipos comerciales tienen un precio elevado. En el capítulo 2 se demuestra este hecho.

La programación e integración de la pantalla táctil en el sistema de control de la microrred está justificada por el hecho de que únicamente se disponía de un terminal táctil en el que se reflejaban los valores de la planta de gasificación del laboratorio. El objetivo es trasladar la monitorización del LabDER a la nueva pantalla táctil, y no sólo colocar los datos de la planta de gasificación, sino de todos los equipos pertenecientes al laboratorio.

Estructura del documento

La estructura del documento se divide en varios capítulos:

- Capítulo 1 - Introducción
- Capítulo 2 - Analizador trifásico
- Capítulo 3 - Terminal táctil de monitorización del LabDER
- Capítulo 4 - Programación del autómata
- Capítulo 5 - Resultados
- Capítulo 6 - Conclusiones
- Capítulo 7 - Bibliografía
- Capítulo 8 - Anexos

El primer capítulo es una toma de contacto con el proyecto, en el que se explican los antecedentes del mismo, el objetivo general y la justificación del mismo.

Los tres capítulos siguientes contienen la descripción general del trabajo. Cada uno de ellos está dividido en tres apartados:

- Objetivo: se detallan los resultados que se pretenden obtener en el capítulo.
- Soporte material: se describen tanto los elementos físicos (hardware, herramientas, cableado…) como el software utilizado en la realización de cada capítulo.
- Procedimiento: se trata de una memoria donde se describen las tareas realizadas para completar cada apartado, los problemas que han surgido, y las soluciones que se han adoptado para resolverlos.

El capítulo 5 se sitúa una vez terminada la descripción de los procedimientos utilizados para llevar a cabo este trabajo. En él, se detallan los resultados del mismo: en particular, el análisis del dispositivo Arduino construido, para comprobar su diferencia en precio y funcionalidades con el modelo comercial, y la integración del terminal táctil en la microrred.

El capítulo 6 muestra las conclusiones, en las que se valoran y sintetizan los conocimientos adquiridos y los resultados obtenidos tras este trabajo.

El capítulo 7 contiene la bibliografía: manuales, páginas web y artículos consultados con el objetivo de vincular los resultados expuestos con evidencias encontradas en estos documentos.

Por último se encuentra el capítulo de anexos, donde se incluyen apartados que complementan y dan más información acerca del trabajo realizado.

Capítulo 2 - Analizador trifásico Arduino

Objetivo

El objetivo de esta parte es la construcción de un analizador trifásico mediante sus componentes básicos, de manera que nos permita evaluar el consumo energético de una red, obteniendo valores como tensión, corriente y potencia consumida por fase; además de calcular la energía total consumida.

El montaje se compondrá principalmente de una placa Arduino, que actuará como controlador; y de varios sensores que le proporcionarán a la misma datos sobre corriente y tensión. Esta información será adquirida y procesada por la placa, la cual devolverá los resultados enviándolos mediante ModBus a un autómata o bien mostrándolos en una pantalla LCD integrada en el montaje.

La construcción de este analizador está justificada, además del aprendizaje y experiencia adquirida, para demostrar que se puede conseguir un analizador trifásico mucho más barato que los disponibles actualmente en el mercado. Los analizadores con comunicaciones Modbus con los que cuenta la microrred son Siemens Sentron PAC 3200 y Schneider PM710, equipos con un rango de precios de entre 400 y 700€.

Además, para el PM710 se requiere una pasarela para convertir el protocolo de comunicaciones Modbus sobre RS485 a Modbus sobre TCP/IP. El coste de dicha pasarela es de 500€, y aunque se podrían utilizar hasta 255 analizadores con esa única pasarela, es un precio elevado, teniendo en cuenta que se podrían realizar las mismas funciones con un analizador basado en Arduino, cuyo coste de fabricación ronda los 100€.

Nota: Debido a la relevancia que tiene en este trabajo el precio de los componentes y equipos, se adjunta en el Anexo 9 - Referencia de precios los detalles y origen de cada uno de los costes descritos en este documento.

Soporte material

Equipos

- Multímetro
- Soldador de estaño

Componentes

- Placa Arduino MEGA
- Arduino Ethernet Shield
- Arduino LCD Keypad Shield
- Resistencias de diferentes valores
- 3 transformadores 220/12V
- 7 condensadores de 10 microfaradios
- Placa de baquelita
- Cables macho-macho y macho-hembra
- Cable Ethernet
- Cable USB – Micro USB
- 4 pinzas amperimétricas
- Caja de montaje
- 2 postes hembra

Software

- Arduino IDE
- Microsoft Excel

Procedimiento

El primer paso en la realización de un proyecto de estas características es tener claro el diseño del equipo que se va a construir. Se necesita un dispositivo que sea capaz de medir la tensión y la corriente instantánea, obtener la forma de onda y el desfase entre las ondas de tensión y de corriente. A partir de estas mediciones, se realizan los cálculos necesarios para calcular valores de potencia, frecuencia, factor de potencia y energía.

El objetivo final es el diseño de un circuito que permita la medición del voltaje mediante una placa Arduino, y otro que lo haga con la corriente. Gracias a que la plataforma Arduino es de código abierto, se ha encontrado un modelo con estas características (4), el cual ha servido de base para el desarrollo del presente analizador. El procedimiento de diseño y realización se describe a continuación.

Nota: se han estudiado posibles mejoras para incluir en el analizador Arduino, pero debido a las limitaciones de tiempo y espacio del proyecto, no han sido llevadas a la práctica. El resultado de estas investigaciones, cuyo objetivo es la ampliación de las funcionalidades del analizador, se puede encontrar en el Anexo 5 - Determinación de la frecuencia de muestreo del analizador Arduino

Un factor relevante a la hora de realizar mediciones es conocer cada cuánto tiempo el analizador Arduino recoge datos de la red. Una rápida frecuencia de muestreo permite una medición mucho más exacta de los valores, además de ser fundamental para otras aplicaciones, como la medición de armónicos.

Consultando la hoja de características del analizador Siemens , se observa que su velocidad de muestreo es de 64 samples por ciclo, es decir, por periodo. Dado que la frecuencia es 50Hz, si se multiplican ambos valores se concluye que la velocidad de muestreo es de 3.2KHz.

Es momento de comparar este valor con el del analizador construido. Este valor dependerá tanto de las capacidades hardware de la placa, como del programa que se está ejecutando. Al consultar en la página oficial , se puede comprobar que la velocidad de reloj es muy superior a la del analizador Siemens, pues trabaja a 16MHz. Aun así, primero se debe comprobar la eficiencia del código empleado, que será determinante si se desea conseguir un tiempo de ejecución más reducido.

Para comprobar el tiempo de ejecución de un bucle en Arduino, simplemente se inicializa a 0 una variable llamada *last_time* (de tipo *long int*) en la cabecera del sketch, y se añade la siguiente función al comienzo del loop:

```
Serial.println(millis()-last_time);
last_time = millis();
```

Código 8.1: Comprobación de la velocidad de ejecución del sketch del analizador

Al acceder al Serial Plotter, se obtienen los siguientes resultados:

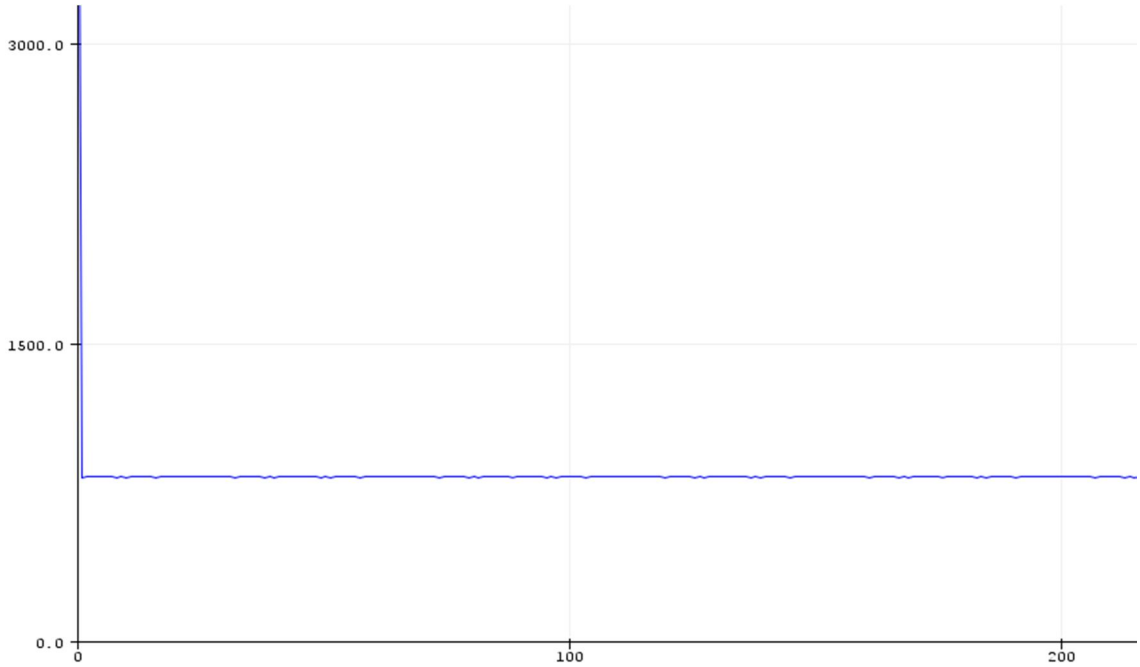


Figura 8.1: Tiempo de ejecución del sketch del analizador Arduino

Es decir, cada loop tarda en ejecutarse unos 827 milisegundos (valor obtenido del Monitor Serie). Al convertir este valor en frecuencia, se obtienen 1.2Hz, demasiado baja para la aplicación que se desea utilizar.

En este momento cabe preguntarse cuál sería la disminución del tiempo de ejecución si se optimizase el código lo suficiente. Se va a realizar una prueba para un código muy sencillo, para comprobar su velocidad de ejecución.

```
void setup()
{
  Serial.begin(115200);
}
long int last_time = 0;
void loop()
{
  Serial.println(micros()-last_time);
  last_time = micros();
}
```

Código 8.2: Comprobación de la velocidad de ejecución de un código sencillo

El parámetro más importante es la velocidad de comunicación Serial, que se ha establecido en 115200, un valor muy superior al utilizado por defecto, para evitar que la velocidad de comunicación sea un factor limitante en la velocidad de ejecución del bucle. Se compila y sube el sketch, obteniendo los siguientes resultados en el Serial Plotter:

Figura 8.2: Tiempo de ejecución de un programa sencillo en la placa Arduino

En este gráfico se puede observar que el programa tarda entre 4 y 12 microsegundos en ejecutarse, por lo que suponiendo un valor medio de 8 se tendría una frecuencia de 125KHz, un valor muy superior al que ofrece el modelo comercial.

Por tanto, se puede concluir que la velocidad de muestreo de una placa Arduino depende enteramente del programa que se ejecuta. Por motivos de tiempo y extensión, no se ha estudiado hasta qué punto se podría optimizar el programa analizador de tensión para mejorar su velocidad de ejecución, pero queda patente que se pueden llegar a alcanzar velocidades muy elevadas, por lo que se pondría por delante del analizador Siemens en este aspecto.

Anexo 6 - Funciones adicionales del analizador Arduino.

Diseño general del analizador

El objetivo de este analizador es la medición, captura y envío de los datos medidos a la microrred. Para ello, se necesitarán algunos componentes físicos que, al trabajar en conjunto, posibiliten la realización de estas funciones.

La base del diseño del analizador está presente en el proyecto Open Energy Monitor (4), en el cual se describe el circuito y los componentes necesarios para la medición de la corriente y la tensión. En el caso que nos ocupa, se realizará esta medición de la corriente y de tensión en un sistema trifásico, por lo que se han de tener en cuenta algunos factores para tomar decisiones acerca del diseño final.

Uno de los más importantes es el hecho de que se necesita un transformador para cada tensión que se desee medir. En un sistema trifásico, se pueden medir seis tensiones diferentes: tres de línea y tres de fase, lo que implicaría el uso de seis transformadores.

El problema reside en el elevado peso, tamaño y coste de los transformadores requeridos, por lo que se plantea la cuestión de si sería posible reducir el número de transformadores. Esto implicaría la medición de menos tensiones, pero dado que las tensiones de línea se pueden calcular a partir de las tensiones de fase, finalmente se decidió limitar el número de transformadores a tres, para posteriormente calcular los valores restantes.

En el apartado de medición de la corriente, debido a que se disponía del suficiente material, se decidió incluir cuatro pinzas amperimétricas, para la medición de todas las corrientes de línea y neutro.

Para la visualización de los datos medidos, se incluye una pantalla LCD de 16x2 caracteres, que permite mostrar la información en varias pantallas. El apartado de conexiones se completa con la presencia del shield Ethernet, que posibilita la conexión del analizador a una red local.

Circuito medidor de corriente

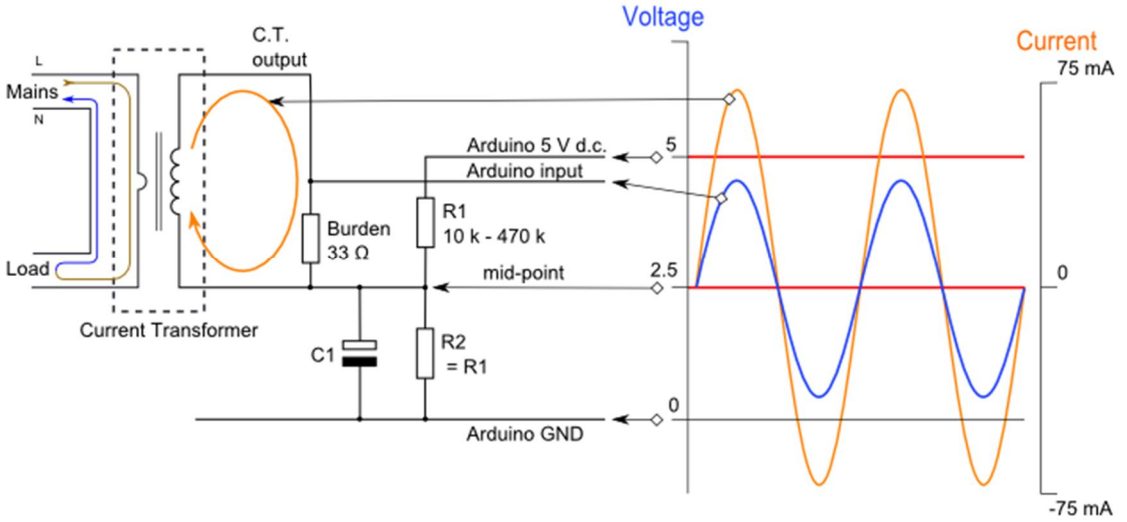


Figura 2.1: Circuito medidor de corriente (5)

El objetivo principal de este circuito es adaptar la onda de corriente a una que la placa Arduino sea capaz de medir, para lo que se realiza esta conversión en dos pasos. Primero, se aísla el circuito de medición del de potencia, rebajándose la tensión mediante un transformador. Posteriormente, se ajusta la onda a los niveles de tensión (1-4V) de la entrada analógica de la placa mediante un circuito divisor de tensión.

La entrada de tensión al circuito se realiza mediante un transformador. Al haberse decidido utilizar un elemento de medición no invasiva, se ha elegido una pinza amperimétrica para desempeñar esta función. La misma consta de un transformador reductor y de una resistencia interna, en la figura denominada *Burden*.

Se procede a describir los cálculos a realizar para conocer la tensión a la salida de la pinza. Como paso previo, se consultan los datos de relación de transformación, rango de corriente de entrada y valor de la resistencia integrada, disponibles en la lista de características del dispositivo (6).

Para averiguar la tensión a la salida, se toma el valor de corriente pico de entrada para calcular la corriente máxima que podría circular (ecuación 1). Después, se convierte la corriente de entrada, mediante la relación de transformación, a la corriente que circula por el circuito de medición (ecuación 2). Para finalizar, se calcula el valor de tensión a la salida aplicando la ley de Ohm (ecuación 3).

$$Corriente\ pico\ ent. = \sqrt{2} * Corriente\ ent.máx.\ (RMS) = \sqrt{2} * 30 = 42.43\ A \quad (1)$$

$$Corriente\ pico\ sal. = \frac{Corriente\ pico\ ent.}{Rel.transformación} = \frac{42.43}{1800} = 0.02357\ A \quad (2)$$

$$Tensión\ pico\ sal. = Corr.pico\ sal. * R_{Burden} = 0.02357 * 62 = 1.46\ V \quad (3)$$

En este punto, se tiene una onda de tensión con valores comprendidos entre -1.46 y +1.46V. Aunque la entrada analógica de la placa Arduino permite la entrada de ondas de tensión en un rango de entre 0 y 5V (7), se va a establecer un rango más restringido (1 a 4V) para evitar errores en las mediciones debidos a picos de tensión.

Para ello, se utilizará un circuito divisor de tensión, el cual consta de dos resistencias iguales conectadas a una diferencia de potencial constante de 5V, suministrada por la placa Arduino entre los pines GND y 5V. Al conectar ambas resistencias en serie, se obtiene un valor de voltaje en el punto medio igual a la mitad del original.

Este hecho posibilita el aumento del valor medio de la onda de tensión a la salida de la pinza amperimétrica, a los valores descritos en las ecuaciones (4) y (5).

$$Tensión\ pico\ sup.\ Arduino = 2.5 + 1.46 = 3.46V \quad (4)$$

$$Tensión\ pico\ inf.\ Arduino = 2.5 - 1.46 = 1.04V \quad (5)$$

Al estar comprendidos entre 1 y 4V, se da por válida la resistencia interna que incorpora la pinza amperimétrica. Si el valor de la tensión de pico fuese superior a 4V, habría que añadir otra resistencia extra para evitarlo.

Por último, también se observa que en el circuito se tiene un condensador, cuya función es evitar que la señal alterna del medidor desplace ligeramente el valor de referencia de los 2,5 V. Debido a que los condensadores únicamente dejan pasar la corriente alterna a través de ellos, bloqueando la continua, se consigue así desacoplar la señal del medidor de la de referencia, de forma que su valor no se vea afectado por la señal de entrada. Se recomienda que el condensador tenga una reactancia de unos pocos cientos de ohmios, por lo que se ha elegido un condensador de 10 µF, calculándose su valor de reactancia en la ecuación (6).

$$X_C = \frac{1}{2 * \pi * f * C} = \frac{1}{2 * \pi * 50 * 10 * 10^{-6}} = 318.3\ \Omega \quad (6)$$

Circuito medidor de tensión

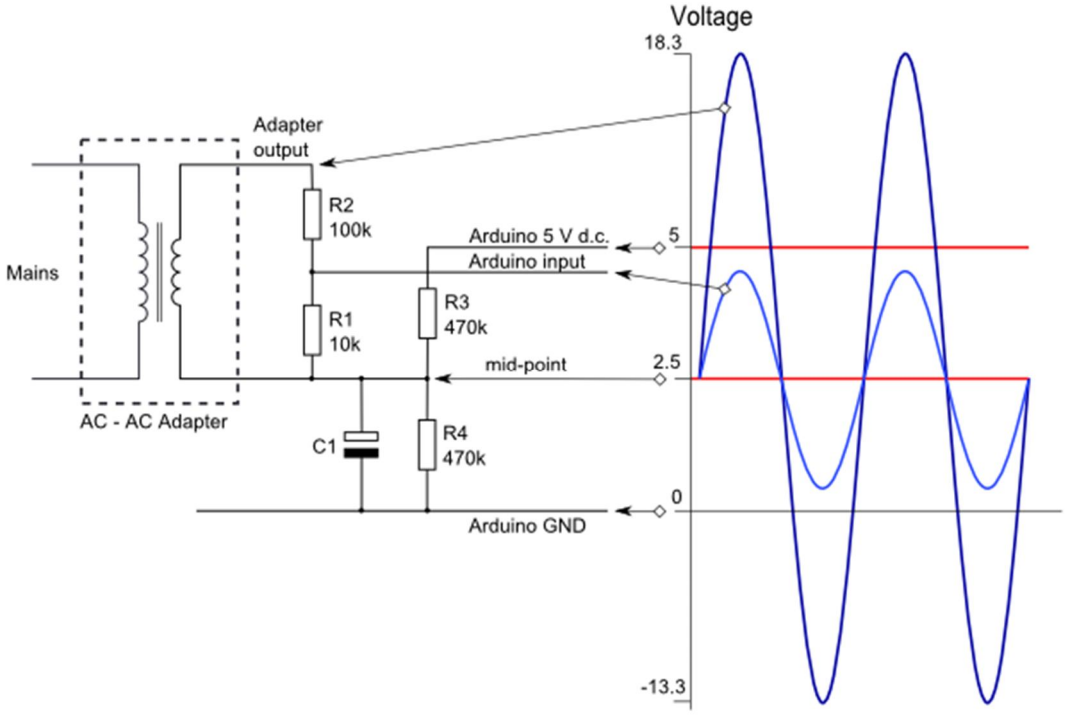


Figura 2.2: Circuito medidor de tensión (8)

Como se puede observar, el circuito medidor de tensión es muy parecido al de corriente, con la única diferencia de que dispone de otra resistencia. Las resistencias R3 y R4, de igual valor, se utilizan como divisor de tensión, y el condensador cumple la misma función que en el circuito anterior.

En este caso, para la entrada de tensión al circuito de medición se han utilizado transformadores de 220/12V, por lo que se dispone de una onda cuya tensión eficaz son 12V. En las ecuaciones (7) y (8) se detalla el cálculo de la tensión de salida para ese valor de tensión eficaz de entrada.

$$Tensión\ pico\ ent. = \sqrt{2} * Tensión\ eficaz = \sqrt{2} * 12 = 16.97\ V \quad (7)$$

$$Tensión\ pico\ a\ pico = 2 * Tensión\ pico\ ent. = 2 * 16.97 = 33.94\ V \quad (8)$$

El objetivo es que la onda de tensión esté comprendida dentro del rango 1 a 4V, por lo se utilizan resistencias para reducir esta tensión a una admisible. Utilizando la Ley de Ohm, se llega a la ecuación (9). Se despeja el factor k, calculándose su valor máximo en la ecuación (10).

$$Tensión\ pico\ sal. = k * Tensión\ pico\ ent.; \qquad k = \frac{R_1}{R_1 + R_2} \quad (9)$$

$$k_{max} = \frac{Tensión\ pico\ sal.}{Tensión\ pico\ ent.} = \frac{2.5}{33.94} = 0.07366 \quad (10)$$

El siguiente paso consiste en combinar las resistencias disponibles en el laboratorio de manera que el valor de *k* sea menor que k_{max}. Se ha utilizado finalmente el factor k calculado en la ecuación (11), que resulta en una tensión de salida descrita en la ecuación (12).

$$R_1 = 220\Omega; \qquad R_2 = 10k\Omega; \qquad k = \frac{R_1}{R_1 + R_2} = 0.02153 \tag{11}$$

$$Tensión\ pico\ sal. = k * Tensión\ pico\ ent. = 0.73V \tag{12}$$

Dado que este valor es el valor de pico, y considerando que la señal oscila alrededor del valor medio 2.5V, los valores pico de la tensión se calculan en las ecuaciones (13) y (14).

$$Tensión\ pico\ sup. Arduino = 2.5 + 0.73 = 3.23\ V \tag{13}$$

$$Tensión\ pico\ inf.\ Arduino = 2.5 - 0.73 = 1.77V \tag{14}$$

Valores que se encuentran dentro del rango de los 5V, aunque bastante alejados del mismo. Una puntualización a realizar es el grado de precisión que se alcanza al estar tan distantes estos valores de tensión de pico de los valores límite de entrada analógica de la placa.

Para comprender mejor esto, se procede explicar el modo de utilización de las señales analógicas por parte de la placa Arduino. Al ser una placa digital, no puede operar con ondas analógicas, por lo que realiza una conversión para transformarla en una señal digital. Esto es, simplifica una onda analógica con infinitos valores en una onda digital con valores finitos. Cuanto mayor sea el número de valores finitos en los que se divide la onda, mayor será la precisión, pero también mayor el tiempo de cálculo y la memoria utilizada para almacenar estos datos.

Arduino cuenta con una resolución de 10 bits, es decir, 1024 valores diferentes. Se transforma la onda analógica con infinitos valores entre 0 y 5V a una onda digital comprendida en un rango de entre 0 y 1024. La precisión que tendría si se aprovechase todo el rango de tensión sería de aproximadamente 5 mV, significando esto que con un cambio de solo 6 mV la placa detectaría que el valor leído ha cambiado.

Pero con las resistencias instaladas, el rango de tensión analógico va de 1.77 a 3.23V, lo cual significa que, al convertirlo a digital, únicamente se tendrá un rango de entre 362 y 662. Es decir, la resolución del analizador es algo inferior a la máxima posible (17mV frente a 5mV), lo cual le resta precisión al analizador construido.

Montaje del circuito

Tras el diseño teórico del circuito, se procede a su construcción física. Dado que se ha decidido utilizar como fuente de información tres mediciones de tensión y cuatro de corriente, se montan tres circuitos de medición de tensión y cuatro de medición de corriente en una placa de pruebas.

Para ello, se colocan los componentes según la Figura 2.1: Circuito medidor de corriente y la Figura 2.2: Circuito medidor de tensión, utilizando los valores de resistencias y condensadores descritos anteriormente.

La conexión de cada uno de estos circuitos con la placa Arduino se realiza a través de los diferentes pines Analog Input de la placa, mientras que todos estos circuitos comparten los mismos pines de tensión a 5V y de puesta a tierra GND. En este caso, se ha decidido colocar los tres medidores de voltaje en los pines 2, 3 y 4 y los medidores de corriente en los pines 8, 9, 10 y 11, aunque podría utilizarse cualquiera de ellos con la misma funcionalidad.

Conexión de la placa al ordenador

Para la creación del programa de la placa, se necesita el software Arduino IDE, el cual se puede descargar desde su página oficial (9). Una vez instalado, se comprueba la correcta comunicación entre el software y la placa, para lo que se ha de consultar el menú *Herramientas* y asegurarse de que el modelo de placa y el puerto seleccionado coincidan con la placa física. Un ejemplo de una correcta conexión se observa en la figura siguiente.

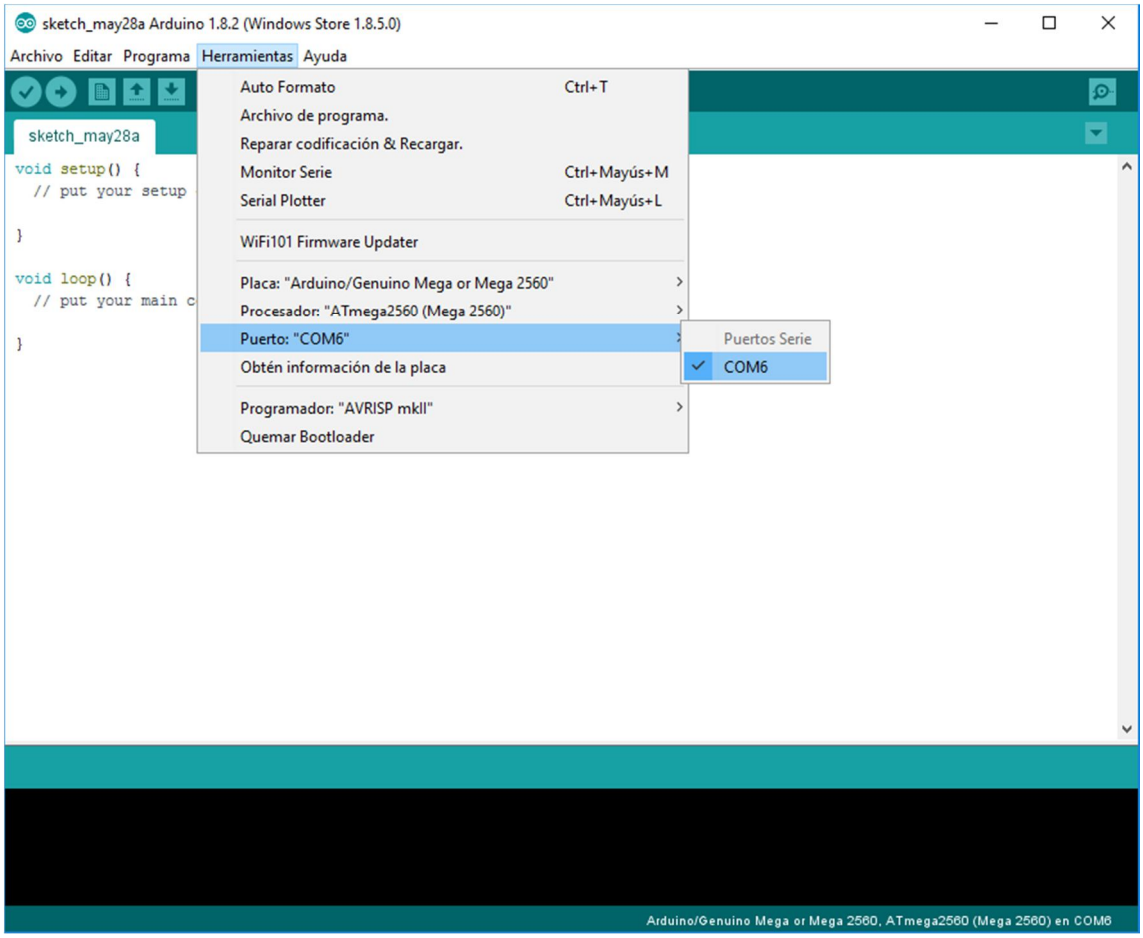


Figura 2.3: Comprobación de la correcta conexión de la placa

Si no se ha conseguido una correcta conexión, el motivo más frecuente es la falta de drivers. Dado que muchas de las placas que se comercializan hoy en día son placas compatibles, pero no originales, hay ocasiones en las que se requiere la instalación de drivers adicionales para el correcto funcionamiento de la conexión entre el IDE y la placa. Para la placa utilizada, al ser una placa compatible, se necesitan los drivers del chip CH340 (10).

Creación del software de la placa

En este trabajo se va a hacer uso de la librería EmonLib, la cual se descarga de su repositorio oficial en GitHub (11). Esta librería contiene un conjunto de funciones útiles para la medición de corriente y tensión, además del cálculo de parámetros como la potencia, energía o factor de potencia. Se va a mostrar un ejemplo sencillo del funcionamiento de esta librería:

```
#include <EmonLib.h>                   // Include Emon Library
EnergyMonitor emon1;                   // Create an instance

void setup()
{
  Serial.begin(9600);

  emon1.voltage(2, 234.26, 1.7);  //V: input pin, calibration, phase_shift
  emon1.current(8, 111.1);       // I: input pin, calibration.
}

void loop()
{
  emon1.calcVI(20,2000);         // Calculate all. No.of half wavelengths
  emon1.serialprint();           // Print out all variables
}
```

Código 2.1: Ejemplo básico del sketch para la medición de corriente y tensión

Primero, se incluye la librería EmonLib, y se crea un objeto *emon1* de la clase EnergyMonitor.

En el bucle de configuración, se inicia la comunicación serial a una tasa de 9600bps, para que posteriormente se puedan consultar los resultados en un monitor integrado en el IDE.

En las siguientes dos líneas, se configura el objeto emon1 para que lea la onda de tensión del pin 2 y la onda de corriente del pin 8. El resto de parámetros son parámetros de calibración, que se revisarán más adelante.

En el bucle principal, se calculan todos los valores (voltaje, corriente, potencia activa, reactiva y aparente, factor de potencia) y se envían al monitor serie para su visualización por pantalla.

Tras la escritura del código, se compila y se sube el sketch a la placa. Para comprobar el correcto funcionamiento del programa, se accede al monitor serie, en el que se han de poder visualizar los valores que el analizador mide en tiempo real.

Calibración del analizador

Para este apartado, se ha de contar con un multímetro, para comprobar cuál es el valor real al que ha de aproximarse el valor medido por el analizador. Se van a modificar los parámetros de calibración de las funciones vistas anteriormente, de manera que se consiga el mínimo error posible.

Primero se realizará la calibración de la tensión. Para ello, se necesita modificar ligeramente el código, con el objetivo de mostrar únicamente los valores de voltaje medido:

```
#include <EmonLib.h>             // Include Emon Library
EnergyMonitor emon1;             // Create an instance

void setup()
{
  Serial.begin(9600);

  emon1.voltage(2, 234.26, 1.7);  //V: input pin, calibration, phase_shift
  emon1.current(8, 111.1);       // I: input pin, calibration.
}

void loop()
{
  emon1.calcVI(20,100);         // Calculate all. No.of half wavelengths
  Serial.println(emon1.Vrms);    // Print only voltage
}
```

Código 2.2: Calibración de la tensión

Se conectan los terminales de medida del analizador a la diferencia de tensión que se desee medir, y se hace lo propio con el multímetro. Esta diferencia de tensión debe de ser lo más constante posible, para evitar errores en la calibración.

Se debe de tener en cuenta que los mejores resultados se obtienen cuando se calibra el analizador para el rango de valores en los que va a trabajar, pues el coeficiente de calibración es ligeramente diferente a medida que varían los niveles de tensión. Dado que el escenario real en el que se instala este analizador trabaja con tensiones de fase de alrededor de 220V, se calibra el analizador utilizando ese nivel de tensión como referencia.

Además, se ha modificado también el segundo parámetro de calcVI, en el que se indica el tiempo de espera entre una medición y la siguiente. Se ha reducido de 2000 a 100 milisegundos, para obtener rápidamente una gran cantidad de valores y poder realizar una calibración más precisa.

Se ejecuta el programa, y se calcula el valor medio de los valores medidos. Al comparar este valor con el valor mostrado por el multímetro, se comprueba si éste es menor o mayor al mismo, para corregir el coeficiente de calibración. Este coeficiente aparece como segundo parámetro de la función *emon1.voltage.*

Una vez corregido, se repite el proceso hasta que el valor medio de los valores obtenidos se acerque lo suficiente al valor real medido por el multímetro. El valor de este coeficiente de calibración, entre otros factores, depende del valor de las resistencias utilizadas en el circuito, por lo que será diferente para cada analizador construido. Para este analizador, el valor obtenido es 518.

Tras calibrar la tensión, se debe de hacer lo propio con la corriente, modificando el código para que se muestre en el monitor serie el valor de *emon1.Irms* en lugar de *emon1.Vrms*. El coeficiente de calibración que se debe ajustar es el que se encuentra dentro de la función *emon1.current*. El valor obtenido para este analizador es 29.6.

Por último, se realiza la calibración del factor de potencia, que se regula mediante el tercer parámetro de *emon1.voltage.* Para evitar complicaciones, se recomienda conectar el analizador a una carga puramente resistiva, y modificar este valor hasta que el factor de potencia sea la unidad. El nombre de la variable a mostrar es *emon1.powerFactor.*

Se repite el proceso anterior con las tres fases, teniendo en cuenta que los coeficientes que se obtienen para las fases son iguales o muy parecidos entre sí, dado que los circuitos contienen los mismos componentes.

Ajuste de la tensión y corriente medidas

A pesar de que utilizando las funciones de EmonLib se consigue un ajuste bastante exacto de la tensión y corriente medidas, en ambos casos existen fluctuaciones en las mediciones instantáneas, como se puede observar en la *Figura 2.4: Comportamiento errático de la tensión medida*.

Dado que el problema es el mismo tanto en el apartado de la tensión como en el de la corriente, se procede a explicar la solución utilizada para el cálculo de la tensión, para después aplicarla también en el cálculo de la corriente.

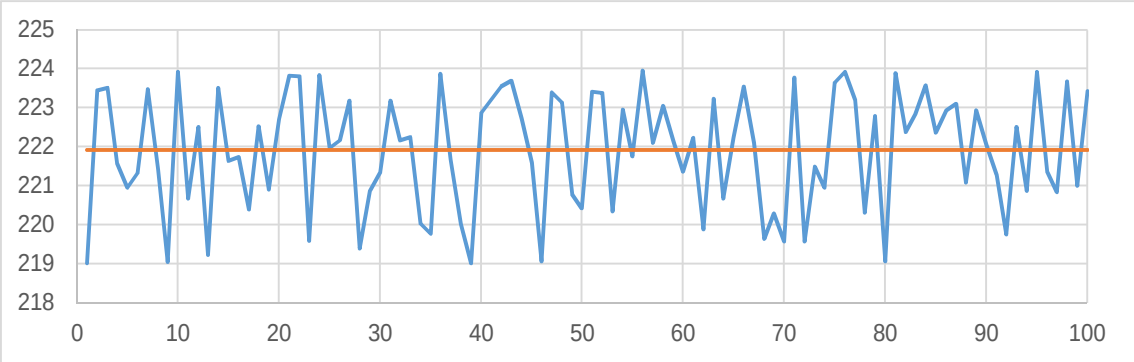


Figura 2.4: Comportamiento errático de la tensión medida

Se tiene un valor de tensión, medido por el multímetro, de 222V. Como se puede observar, el valor medio de las mediciones realizadas por el analizador se aproxima bastante a ese valor, pero las mediciones discretas fluctúan en un rango comprendido entre 219V y 224V.

Las razones de este comportamiento se han atribuido a la alimentación del analizador mediante conexión USB. Este conector no asegura una diferencia de potencial fija, haciendo variar ligeramente el voltaje de 5V del pin de la placa y afectando al comportamiento del circuito divisor de tensión.

Se procede a reducir este error de medición, utilizando para ello el valor medio de las tensiones medidas. Dado que éste se aproxima bastante al valor del multímetro, se incorpora al programa un cálculo del valor medio de los últimos valores medidos para mejorar la precisión.

En la cabecera se añade la definición de 4 vectores, uno para el voltaje de cada fase, y otro temporal para realizar operaciones. La longitud de este vector determina el número de mediciones con las que se calcula el valor medio.

Una longitud elevada implica una gran cantidad de valores de medición, por lo que se consigue un valor más preciso. En cambio, una longitud reducida favorece la respuesta del analizador ante un cambio en el nivel de tensión. Al probar con diferentes valores, se llega a la conclusión de que 5 es un valor de longitud que alcanza el compromiso entre precisión de medida y rapidez ante un cambio de tensión.

Además, se definen 3 variables en las que se guarda el acumulado de los valores anteriores.

```
double V1[5], V2[5], V3[5], Vtemp[5];
double V1_tot, V2_tot, V3_tot;
```

Código 2.3: Cabecera del ajuste de la tensión medida

El contenido del bucle setup es simplemente un vaciado de los vectores vistos anteriormente, para evitar errores en el primer paso del bucle.

```
for(i=0;i<5;i++){ V1[i] = 0; V2[i] = 0; V3[i] = 0; }
```

Código 2.4: Bucle setup del ajuste de la tensión medida

Por último, se muestra el contenido del bucle loop. En él, se guarda el último valor medido de tensión en el vector, a la vez que se descarta el más antiguo. Se calcula el valor medio de los componentes del vector, obteniéndose el valor de tensión requerido.

```
V1_tot = 0; V2_tot = 0; V3_tot = 0;

for(i=0;i<5-1;i++){  Vtemp[i+1] = V1[i]; }
for(i=0;i<5;i++){ V1[i] = Vtemp[i]; V1_tot = V1_tot + V1[i]; }

for(i=0;i<5-1;i++){  Vtemp[i+1] = V2[i]; }
for(i=0;i<5;i++){ V2[i] = Vtemp[i]; V2_tot = V2_tot + V2[i]; }
```

```
for(i=0;i<5-1;i++){  Vtemp[i+1] = V3[i]; }
for(i=0;i<5;i++){ V3[i] = Vtemp[i]; V3_tot = V3_tot + V3[i]; }


V1[0] = emon_ph1.Vrms;
V2[0] = emon_ph2.Vrms;
V3[0] = emon_ph3.Vrms;
V1_tot = V1_tot + V1[0];
V2_tot = V2_tot + V1[0];
V3_tot = V3_tot + V1[0];


Vf_ph1 = V1_tot/5;
Vf_ph2 = V2_tot/5;
Vf_ph3 = V3_tot/5;
```

Código 2.5: Bucle loop del ajuste de la tensión medida

Este procedimiento se aplica de manera idéntica para aplanar el valor de la medida realizada por la pinza amperimétrica, utilizando los valores de corriente de las líneas y el neutro en lugar de los de tensión.

Cálculo de todos los valores de la red

A pesar de que EmonLib calcula muchos de los valores relativos a la medición, para esta aplicación en concreto se requiere de cálculos posteriores para conseguir algunos de los valores necesarios.

Un grupo de información del que no se dispone todavía es el de las tensiones de línea, debido a que no se han realizado mediciones acerca de estos valores. Dado que se están calculando valores eficaces, y el sistema a medir se considera un sistema trifásico en estrella, se puede utilizar la ecuación (15) para el cálculo de la tensión de línea:

$$U_L = \sqrt{3} * U_F \tag{15}$$

Y para un sistema trifásico en estrella, se cumple la ecuación (16) para el cálculo de la corriente eficaz de línea:

$$I_L = \sqrt{3} * I_F \tag{16}$$

Se describe el cálculo de la potencia reactiva mediante la ecuación (17):

$$Q = \sqrt{S^2 - P^2} \tag{16}$$

Por último, los valores de potencia instantánea se transforman a unas unidades (KW, KVar, KVA) que son más cómodas de utilizar en el rango de potencia en el que se trabaja en este proyecto.

Se van a detallar las instrucciones necesarias, todas ellas incluidas dentro del bucle loop:

```
double Vf_ph1, Vf_ph2, Vf_ph3;
double Vl_ph1, Vl_ph2, Vl_ph3;
double Il_ph1, Il_ph2, Il_ph3, Il_neu;
double If_ph1, If_ph2, If_ph3, If_neu;

Vl_ph1 = sqrt(3)*Vf_ph1;
Vl_ph3 = sqrt(3)*Vf_ph3;
Vl_ph2 = sqrt(3)*Vf_ph2;

Il_ph1 = emon_ph1.Irms;
Il_ph2 = emon_ph2.Irms;
Il_ph3 = emon_ph3.Irms;
Il_neu = emon_neu.Irms;

If_ph1 = Il_ph1;
If_ph2 = Il_ph2;
If_ph3 = Il_ph3;
If_neu = Il_neu;

double P_ph1 = emon_ph1.realPower/1000;
double P_ph2 = emon_ph2.realPower/1000;
double P_ph3 = emon_ph3.realPower/1000;

double S_ph1 = emon_ph1.apparentPower/1000;
double S_ph2 = emon_ph2.apparentPower/1000;
double S_ph3 = emon_ph3.apparentPower/1000;

double Q_ph1 = sqrt(pow(S_ph1,2)-pow(P_ph1,2));
double Q_ph2 = sqrt(pow(S_ph2,2)-pow(P_ph2,2));
double Q_ph3 = sqrt(pow(S_ph3,2)-pow(P_ph3,2));

double pf_ph1 = emon_ph1.powerFactor;
double pf_ph2 = emon_ph2.powerFactor;
double pf_ph3 = emon_ph3.powerFactor;
```

Código 2.6: Cálculo de todos los valores de la red

Cálculo de la energía consumida

Se va a calcular la energía consumida con la ecuación (17) para la energía activa, y la ecuación (18) para la reactiva.

$$E_A = \sum P_i * t_i \tag{17}$$

$$E_R = \sum Q_i * t_i \tag{18}$$

Donde P_i y Q_i representan las potencias instantáneas, y t_i el tiempo transcurrido entre una medición y la siguiente.

Dado que ya se conocen las potencias instantáneas, únicamente se requiere conocer el tiempo que existe entre cada medición. Trasladando este concepto al del analizador Arduino, este tiempo será igual al tiempo de ejecución del bucle principal.

Para ello, se procede a almacenar el valor de la energía en una variable que acumulará el valor instantáneo de cada bucle, calculado a partir del producto entre la potencia instantánea y el tiempo de ejecución del bucle.

El código a añadir en la cabecera es una declaración de variables:

```
int seconds = 0;
short int minutes = 0;
short int hours = 0;
int days = 0;
int lasttime = 0;
int offset = 0;
int e_act = 0;
int e_rea = 0;
```

Código 2.7: Cabecera del cálculo de la energía consumida

Mientras que en el bucle loop se coloca lo siguiente:

```
//tiempo
if(lasttime!=0)
  offset = millis() - lasttime;
seconds = millis()/1000 - minutes*60;
if(seconds>=59)
  minutes++;
if(minutes>=59){
  minutes = 0;
  hours++;
```

```
}
if(hours>=23){
  hours = 0;
  days++;
}
lasttime = millis();
char time_char[12] = " ";
sprintf(time_char, "%2dd%02d:%02d:%02d", days, hours, minutes, seconds);

//calculo energía activa y reactiva
e_act = e_act + (P_ph1+P_ph2+P_ph3)*1000*offset/3600;
e_rea = e_rea + (Q_ph1+Q_ph2+Q_ph3)*1000*offset/3600;
char e_act_char[10]=" ";
sprintf(e_act_char, "%9d", e_act);
char e_rea_char[10]=" ";
sprintf(e_rea_char, "%9d", e_rea);
```

Código 2.8: Bucle loop del cálculo de la energía consumida

El código se basa en la utilización de la función *millis()*, que devuelve el valor de milisegundos que han pasado desde que se puso en marcha el analizador. En la variable *lasttime* se almacena el valor de esta función en la pasada anterior del bucle, mientras que la variable *offset* resta el tiempo actual menos el anterior, teniendo así el tiempo de ejecución del bucle.

Después, se realizan diversas conversiones entre segundos, minutos, horas y días, y se almacenan estos valores en una única variable de tiempo de tipo char. Esto se utiliza como reloj en la pantalla del analizador, para poder conocer el tiempo en el que se ha consumido la cantidad de energía que se está midiendo.

Posteriormente, se calculan las energías consumidas durante este bucle, multiplicando la suma de la potencias de cada fase por el tiempo *offset,* y sumándolas al acumulado de energías.

Por último, esta energía total se convierte a tipo char con el objetivo de dar formato al número para su posterior visualización en la pantalla.

Conexión y diseño de la pantalla LCD

En este momento se tiene un analizador de redes plenamente funcional, pero su mayor limitación reside en que se necesita de un ordenador para poder visualizar los valores que se están midiendo. La solución consiste en integrar una pequeña pantalla LCD al equipo para que el analizador sea independiente del PC y se puedan realizar las mediciones en cualquier lugar.

Para ampliar las funcionalidades de la placa Arduino, existen añadidos llamados shields (12), que se conectan encima de los pines de la placa y quedan perfectamente montados. En este proyecto, se van a utilizar dos de ellos: el Arduino Ethernet Shield, que le permite a la placa conectarse mediante este tipo de puerto; y la LCD Keypad Shield, la cual incorpora una pantalla LCD de dos filas de 16 caracteres y varios botones con los que realizar acciones.

Conectar los shields es muy sencillo, pero en este caso en particular se ha de prestar atención en el montaje para que no surjan problemas. El primero que se debe colocar es el Arduino Ethernet Shield, teniendo cuidado de que todos los pines se hayan introducido correctamente.

Una vez se haya conectado el Ethernet Shield, se puede observar que no se puede colocar correctamente la LCD Keypad Shield, debido a que el puerto Ethernet sobresale más que los pines de conexión. Para ello, se necesitará utilizar un grupo de postes hembra como los que aparecen en la *Figura 2.5: Conexión de los shields Arduino*, o será imposible la conexión de la pantalla.

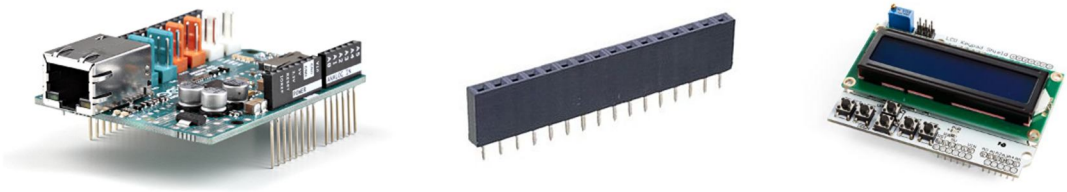


Figura 2.5: Conexión de los shields Arduino

Pero aún queda un detalle con el que se debe tener precaución, pues es causante de errores que afectan a ambos shields. Se trata de que la placa Ethernet utiliza como propios algunos pines, entre ellos el pin 10, por lo que su comportamiento será errático si se conecta cualquier componente en ese pin.

Normalmente, al conectar un shield se conectan todos los pines, pero en este caso se debe dejar el pin 10 sin conectar. Bastará con no colocar un poste hembra en ese pin, para que el pin de la placa de la pantalla no llegue a conectar con el de la placa Ethernet.

Una vez se tengan los shields instalados, se debe pensar acerca del diseño que va a tener la información descrita en pantalla. Dado que se tiene que presentar una gran cantidad de información (tensiones y corrientes de línea y fase, potencias, factor de potencia y energías), resulta imposible condensar toda esa información en una sola pantalla.

Para solucionar este problema, se recurrirá a los botones que el shield trae integrados, con los que podremos avanzar y retroceder entre las diferentes pantallas del programa. En este caso, se ha optado por realizar un diseño que consiste en nueve pantallas, que mostrarán la siguiente información:

1. Voltaje por línea
2. Voltaje por fase
3. Corriente por línea
4. Potencia activa por fase
5. Potencia reactiva por fase
6. Potencia aparente por fase
7. Factor de potencia por fase
8. Energía activa consumida total
9. Energía reactiva consumida total

El diseño de las todas las pantallas es similar: se dividen los 32 caracteres disponibles en 4 bloques de 8 caracteres, para colocar en cada uno de esos bloques los valores medidos o a la descripción de la propia pantalla.

Figura 2.6: Diseño de la pantalla del analizador

Para visualizar el diseño de cada una de ellas, se puede consultar el

Anexo 2 - Diseño de las pantallas del analizador.

Programación de la pantalla LCD

En este apartado, se utiliza la librería LiquidCrystal (13), la cual es una librería desarrollada para facilitar la impresión de caracteres en una pantalla LCD.

Se detalla el código que se agrega en cada parte del sketch, comenzando por el código a incluir en la cabecera:

```
#include <LiquidCrystal.h>


//define screens
int screen      = 0;
int i;


// define some values used by the panel and buttons
int lcd_key     = 0;
int adc_key_in  = 0;


//define left and up as back; right and down as forward
#define btnBACK     0
#define btnFORWARD  1
#define btnNONE     5


// select the pins used on the LCD panel
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);


// read the buttons
int read_LCD_buttons()
{
 adc_key_in = analogRead(0);
 if (adc_key_in > 1000) return btnNONE
 if (adc_key_in < 50)   return btnFORWARD;
 if (adc_key_in < 650)  return btnBACK;
}
```

Código 2.9: Cabecera de la pantalla LCD

De esta parte, se destaca la función *read_LCD_buttons()*, la cual devuelve un valor dependiendo del botón que se haya pulsado. Solamente se van a utilizar tres de los botones incluidos en la

pantalla: el botón Left y Right para navegar entre las pantallas, y el botón de Reset, que reinicia el analizador.

El siguiente lugar en el que incluir código es el bucle setup, en el que simplemente se necesitará inicializar la pantalla. Los dos parámetros que incluye son los pines a los que está conectada en el shield:

```
lcd.begin(16, 2);
```

Código 2.10: Bucle setup de la pantalla LCD

Por último, se detalla el código que se ha incluido en el bucle loop:

```
lcd_key = read_LCD_buttons();  // read the buttons
 int number_of_screens = 10;
switch (lcd_key) {
   case btnBACK:
     {
     if(screen>0) screen--; else screen = number_of_screens;
     break;
     }
   case btnFORWARD:
     {
     if(screen<number_of_screens) screen++; else screen = 0;
     break;
     }
 }
 switch (screen)
 {
    case 0:
      {
        //voltajes de línea
        lcd.setCursor(0,0);
        lcd.print("1 ");
        lcd.print(Vl_ph1,1);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(Vl_ph2,1);
        lcd.setCursor(0,1);
        lcd.print("3 ");
```

```
        lcd.print(Vl_ph3,1);
        lcd.setCursor(7,1);
        lcd.print(" VL (V) ");
      }
    case 1:
      {
        //voltajes de fase
        lcd.setCursor(0,0);
        lcd.print("1 ");
        lcd.print(Vf_ph1,1);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(Vf_ph2,1);
        lcd.setCursor(0,1);
        lcd.print("3 ");
        lcd.print(Vf_ph3,1);
        lcd.setCursor(7,1);
        lcd.print(" VF (V) ");
        break;
      }
}
```

Código 2.11: Bucle loop de la pantalla LCD

Primero, se lee el botón que ha sido pulsado y se introduce su valor en la variable *lcd_key*. Después, dependiendo de si se ha pulsado el botón Left o el Right, se disminuye o aumenta el valor de la variable *screen*. Cada valor de *screen* representa una pantalla, por lo que utilizando la instrucción *switch* se elige la pantalla que se va a mostrar.

Dentro de cada pantalla, se utilizan dos instrucciones fundamentales, que son *lcd.setCursor(),* con la cual elegimos la posición de la pantalla (columna, fila) en la que se van a comenzar a escribir los caracteres, y *lcd.print()*, con la que se muestra un texto por pantalla.

Con esto, se puede dar por finalizada la programación del analizador. Para visualizar el código en su totalidad, éste se ha incluido en el

Anexo 1 - Código fuente del analizador trifásico.

Utilización de resistencias en lugar de transformadores para la reducción de la tensión

En el analizador de redes descrito a lo largo del capítulo, la tensión medida no es la que realmente existe entre los bornes que se están midiendo, sino que es una tensión que ha sido

transformada previamente, tanto por seguridad de cara al usuario, como para la de la placa en la que se montan los componentes.

Además, la placa Arduino solamente puede medir tensiones comprendidas entre 0 y 5V, por lo que se necesita transformar la tensión alterna en bornes, que opera en el rango desde -325V hasta +325V (pico-pico), en una que esté entre los márgenes anteriores.

Para ello, se suele utilizar un transformador, pero también se puede realizar un circuito divisor de tensión, el cual tiene como ventajas:

- Se puede conseguir la relación de transformación que se necesite, simplemente cambiando los valores de las resistencias.
- El transformador induce en la tensión de salida un desfase de 10º respecto de la tensión de entrada. Esto afecta al cálculo del factor de potencia, pues el método de cálculo de la librería *EmonLib.h* se basa en la medición del desfase entre las ondas de tensión y corriente medidas. Así, si la onda de la tensión está desfasada respecto a su valor real, la medición del desfase entre tensión y corriente no resultará correcta, siendo erróneo también el factor de potencia calculado a posteriori. El circuito divisor de tensión aplica la reducción de tensión sobre la onda original, por lo que se evita este problema.
- Se utilizan únicamente 2 resistencias, que son componentes mucho más baratos que un transformador.
- El peso y el tamaño del circuito divisor son más reducidos que los de un transformador, por lo que el analizador de redes resultaría más compacto y ligero.

Aun así, se debe valorar el principal inconveniente de este circuito, que es la seguridad, tanto del usuario como de la placa. Un transformador aísla el circuito de potencia del de medición, por lo que en la placa se tiene siempre una tensión reducida, además de que los cambios en el circuito de la red no afectan directamente al del medidor, y viceversa.

Al seguir la metodología del divisor de tensión, no se obtienen dos circuitos separados, por lo que se debe de prestar especial atención a la protección en las partes del circuito en las que existe una tensión superior a 220V. Esto se consigue aislando cualquier parte metálica de la parte de alta tensión que pudiera ponerse en contacto con el usuario, y revisando bien el circuito antes de cada conexión para evitar cortocircuitos, que a tensiones tan elevadas podrían producir el quemado de los componentes.

Finalmente, a pesar de todas las ventajas que ofrece este concepto, se decidió no aplicarlo en el analizador construido, priorizando la seguridad del usuario y del propio analizador frente a las ventajas que se obtendrían.

Justificación de la utilización del analizador únicamente con conexión en estrella

Dado que solamente se dispone de tres transformadores de tensión de 220/12V, se había descartado la idea de poder medir las tensiones de línea en caso de que fuese necesario, pues éstas tienen valores de entre 380 – 400V. Por tanto, se conecta siempre el analizador en estrella, para medir las tensiones de fase.

Este apartado tiene como objetivo comprobar si se podrían utilizar los transformadores de 220V para medir tensiones de línea. El problema de utilizar una tensión superior a la nominal es que, debido a que la resistencia interna del transformador se mantiene constante, circulará una corriente más elevada. El transformador utilizado tiene una potencia de 4VA, por lo que utilizando la definición de potencia aparente, se puede calcular la corriente nominal mediante la ecuación (19).

$$S = V * I \rightarrow I = \frac{S}{V} = \frac{4VA}{220V} = 0{,}018\ A \quad (19)$$

Esta es la corriente máxima que puede circular por el transformador sin que afecte al aislamiento de los devanados. Dado que no se está utilizando el transformador en carga, sino prácticamente en vacío (solamente estamos realizando mediciones de tensión), es posible que la corriente de vacío a 400V sea inferior a la corriente a carga nominal a 220V (18mA). Se procede a comprobarlo.

Para realizar la prueba, primero se conecta uno de los transformadores a la red trifásica, entre línea y línea. Se verifica que el transformador sigue funcionando, midiendo a la salida una tensión de 27V, lo cual equivale a una relación de transformación de 390/27 = 14,45. Esta relación es superior a la calculada para la tensión nominal (223,6/19,79 = 11,29), lo cual ya induce a sospechar que está habiendo pérdidas de energía en el transformador.

Ahora se comprueba qué ocurre al conectar el analizador. Se conectan los 3 transformadores entre línea y línea, y se mira el valor que aparece en la pantalla del mismo. El valor de esta medida está en torno a los 350V, inferior al valor de 390V que debería medir.

Además, al medir la corriente que circula por el lado de alta del transformador, la medida está entre unos 20 y 50mA. Esta imprecisión se debe a utilizar una pinza amperimétrica calibrada para rangos superiores de corriente pero, en todo caso, se demuestra que esta corriente es superior a los 18mA que deberían circular de corriente máxima.

En efecto, al tocar los transformadores se comprueba que el calor que disipan es demasiado elevado. Esto ocurre debido al efecto Joule, que postula que si en un conductor circula corriente eléctrica, parte de la energía cinética de los electrones se transforma en calor (14). La potencia disipada por efecto Joule se calcula mediante la ecuación (20).

$$P = R * I^2 \quad (20)$$

Dado que la resistencia interna del transformador es constante, si la potencia disipada aumenta se puede concluir que es debido a un aumento de la corriente circulante.

Al tener una corriente en vacío tan elevada, el transformador entra en zona de saturación, cuando la zona recomendada de trabajo es la zona lineal (15):

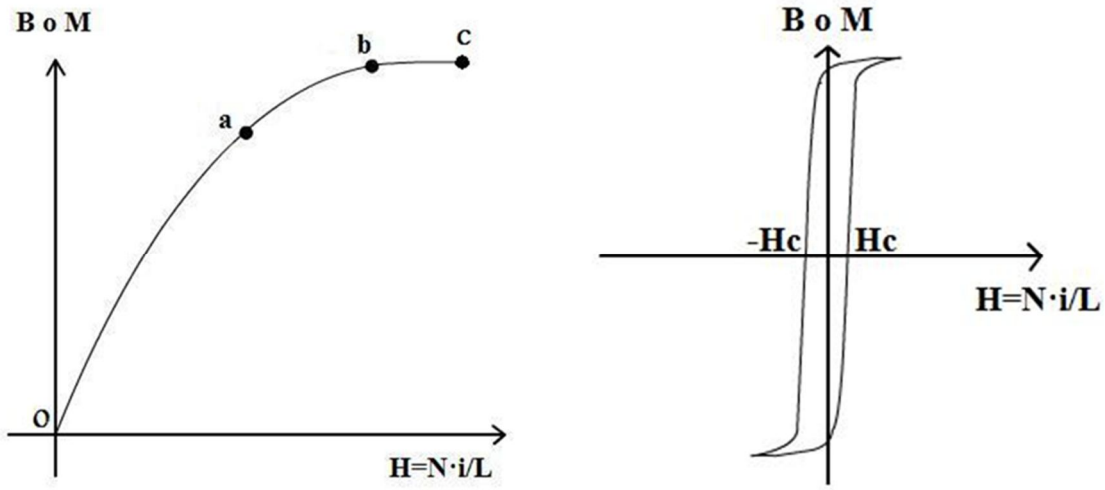


Figura 2.7: Zonas de trabajo y ciclo de histéresis del transformador

Es decir, el núcleo del transformador utilizado ha entrado en la zona bc, por lo que, aunque aumenta el valor de la corriente, no aumenta prácticamente nada el valor del flujo magnético.

Así, al entrar el material en zona de saturación, el área del ciclo de histéresis es más grande, y dado que el área encerrada por el ciclo de histéresis es proporcional a la energía de pérdidas, ésta aumenta.

Al aumentar la energía de pérdidas, se disminuye el rendimiento del transformador, por lo que la energía transmitida al lado secundario se reduce. Esto es lo que ocasiona el reducido valor de voltaje medido por el analizador.

La solución consiste en medir el voltaje entre línea y neutro. Así, al conectar el analizador en estrella, la tensión que existe en el lado de alta del transformador es más cercana a los 220V nominales, por lo que el transformador trabaja en zona lineal. Con esto, se descarta la utilización de estos transformadores para medir tensiones de red en triángulo.

Capítulo 3 - Terminal táctil de monitorización del LabDER

Objetivo

El objetivo principal de esta parte del Trabajo de Fin de Grado es la realización de un sistema de monitorización, llevado a cabo mediante un terminal táctil que permita controlar la microrred y visualizar todas las variables medidas.

El LabDER tiene equipos de medición tales como analizadores de redes, termopares, anemómetros, piranómetros y flujómetros. Cada uno de esos sensores/medidores, por medio de señales analógicas o protocolos de comunicación envía las medidas a uno de los PLCs (CJ2M o CP1L), que guarda esos datos en su memoria interna. La función del terminal HMI es visualizar los datos que están almacenados en la memoria del PLC, para lo cual ambos dispositivos se comunican mediante un protocolo propio de OMRON.

La interfaz mostrada por la pantalla es totalmente configurable, gracias al software OMRON NB-Designer, que permite al programador crear varias pantallas, y colocar imágenes, botones y casillas en las que aparezcan los valores del PLC.

En el caso que nos ocupa, la programación va a tener como objetivo el desarrollo de una aplicación multipantalla, con menús que permitan acceder a las diferentes partes del programa y visualizar así todos los datos que están midiéndose en el laboratorio en tiempo real.

Soporte material

Elementos físicos

- Terminal táctil HMI OMRON NB7W01B
- PLC OMRON CJ2M

Software

- OMRON NB-Designer
- OMRON CX-Supervisor
- OMRON CX-Programmer

Procedimiento

Introducción

Dado que los autómatas programables que posee el laboratorio son modelos comerciales de la marca OMRON (CJ2M y CP1L), se ha elegido el terminal táctil OMRON NB7W-TW01B para realizar esta tarea.

Una de las principales ventajas de este terminal es que posee un software gráfico, llamado NB Designer, en el cual se puede crear de forma fácil la interfaz que se necesite para el terminal. El proceso es muy parecido a la programación de la placa Arduino: primero se diseña el software en el ordenador, para más tarde cargarlo al terminal mediante un cable USB.

Este software se estructura en las llamadas pantallas: cada pantalla es un conjunto de información y gráficos, en las que se configurar diferentes botones para la navegación entre las mismas. Es un concepto bastante fácil e intuitivo, que busca facilitar la tarea de visualización y control de la información de los PLCs.

Instalación del software y primera toma de contacto

El primer paso consiste en descargar e instalar el software NB Designer desde la página oficial de OMRON, donde también se puede encontrar su manual de operación (16), una guía muy útil para la utilización del programa. La interfaz principal se puede observar en la siguiente figura:

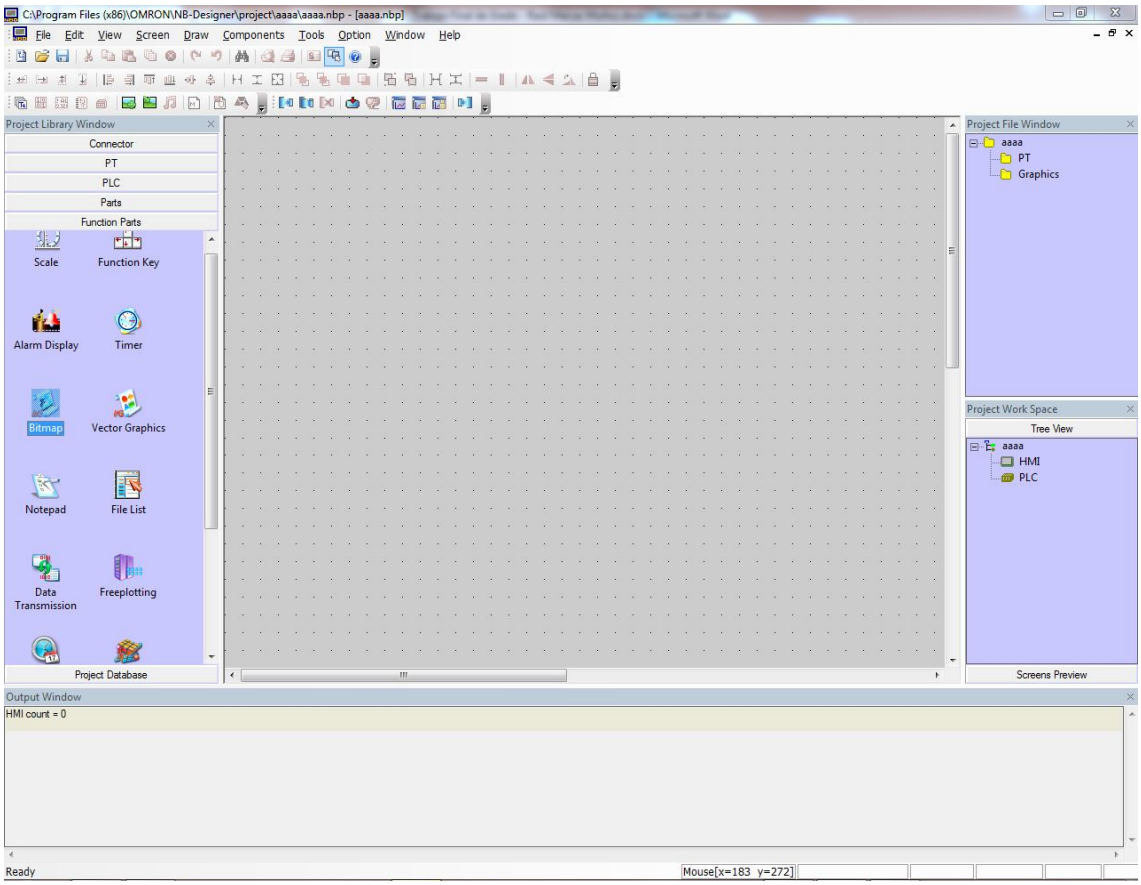


Figura 3.1: Interfaz general del NB Designer

La ventana está dividida en varias partes:

- Barra superior (menús y botones): da acceso a todas las funciones del programa. Para este trabajo se utilizan principalmente los botones de guardado, compilación y descarga al terminal.
- Barra izquierda (elementos de la pantalla): se muestran todos los elementos que se pueden agregar a cada una de las pantallas: display numérico, texto, imágenes, botones de navegación, botones de comando del PLC, gráficos...
- Barra derecha (estructura de la pantalla): se listan todos los elementos agregados al proyecto, como macros y gráficos. En la parte inferior se pueden visualizar los terminales y PLCs conectados, así como las pantallas incluidas dentro de cada terminal.
- Barra inferior (mensajes): se muestran los mensajes de error, advertencias o éxito de la compilación cuando se realiza la transferencia del software al terminal.
- Parte central (diseño de la interfaz): es la parte principal del programa, donde se arrastran los elementos a colocar dentro de cada pantalla.

Creación del proyecto y conexión de los PLCs

Para la creación de un nuevo proyecto, se ha de pulsar en File>New y dar un nombre al mismo.

Posteriormente, se ha de realizar el esquema de conexiones. Los elementos que se conectan a este esquema son el terminal HMI y los PLCs de los cuales el terminal obtendrá la información a mostrar. Por tanto, se insertan desde la barra de elementos un PT (modelo OMRON NB-Series) y los dos PLCs del laboratorio (CP1L y CJ2M), como se puede observar en la figura siguiente.

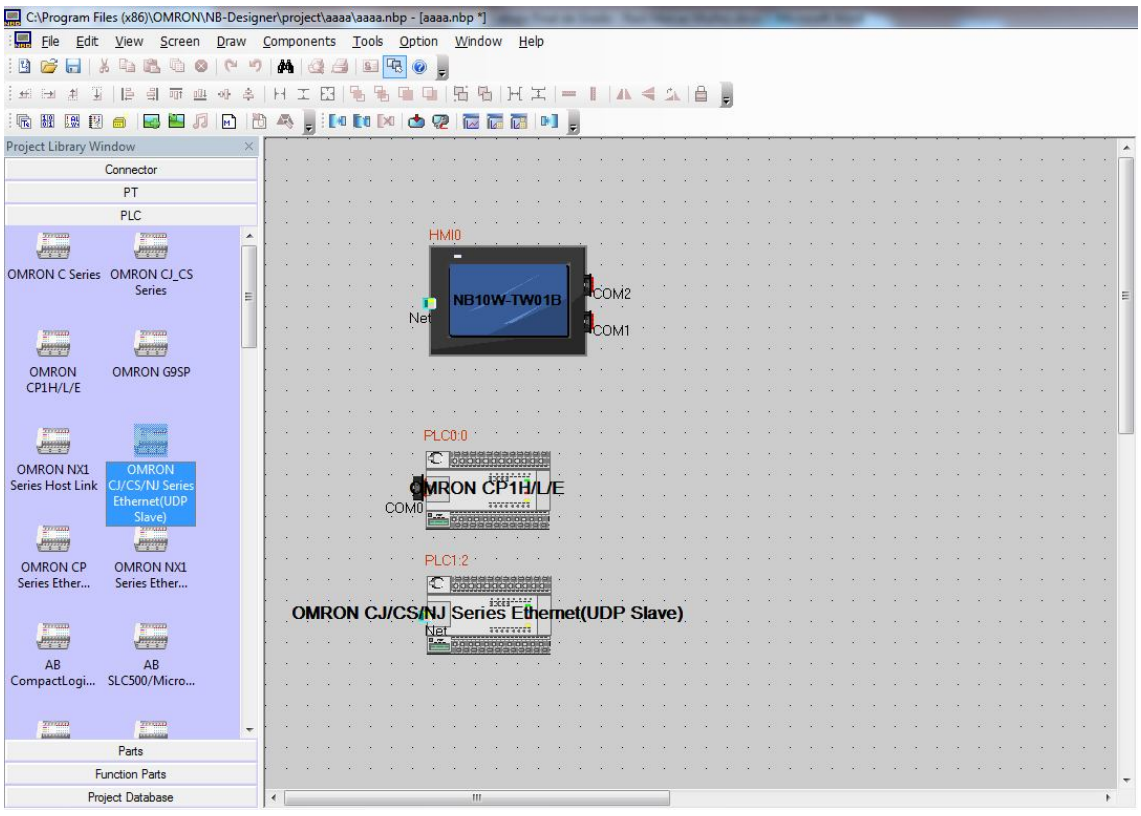


Figura 3.2: Detalle de las conexiones entre PLCs y HMI

Una vez se han añadido los tres elementos que van a formar parte de la comunicación, es momento de conectarlos. Se pulsa en Connector, y se añade una conexión Ethernet. Aparece el siguiente diálogo, donde se muestra un resumen de la configuración de la conexión:

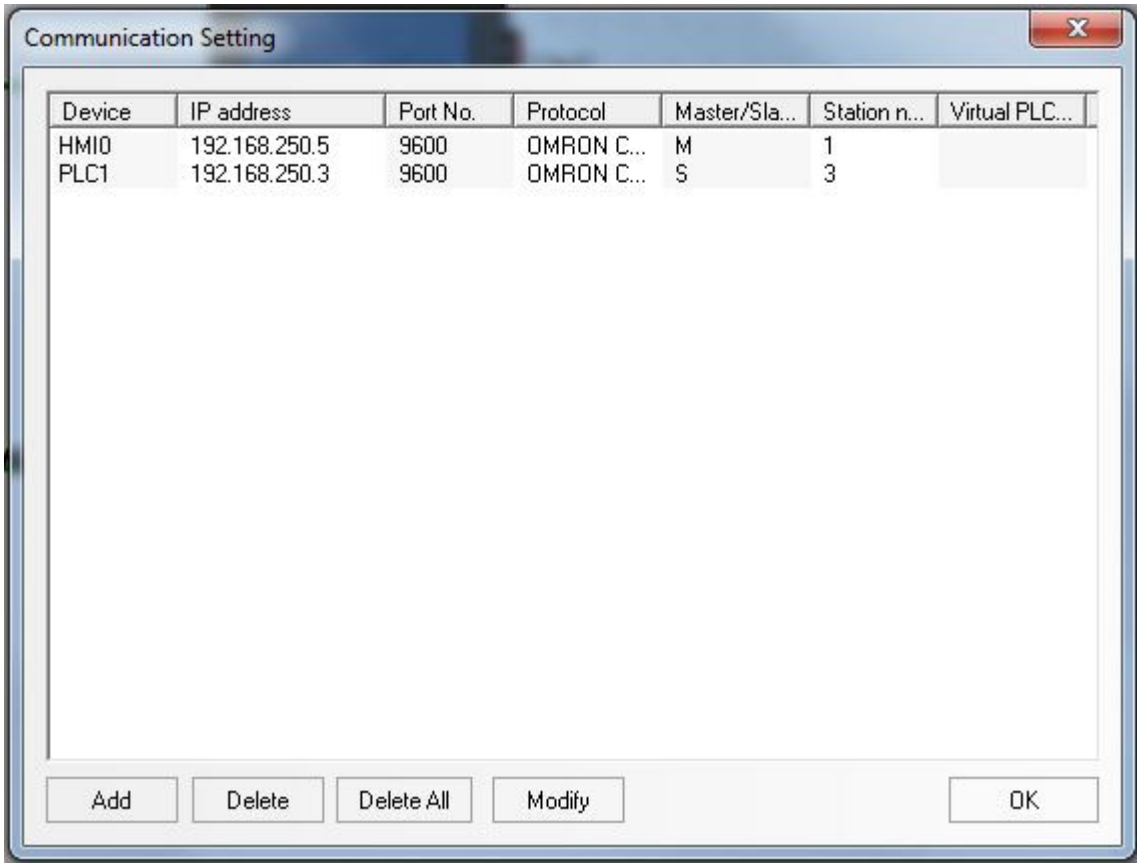


Figura 3.3: Communication Setting PLCs y HMI

Al pulsar en Add, se puede agregar la conexión de estos elementos de manera sencilla. Para ello, se debe conocer con exactitud qué IP y qué número de nodo tiene cada PLC, pues de otra manera no se podrá conectar. La pantalla, al ser un nuevo elemento en el laboratorio, necesitará una dirección IP que no esté repetida en la misma red, y un número de nodo único, también como identificación.

Creación de las pantallas

Una vez se han realizado todas las conexiones, se procede a la visualización de la información del PLC en la pantalla del HMI. Para ello, en la barra derecha, se pulsa sobre HMI0, que es como NB Designer denomina al terminal táctil.

Como se puede observar, el programa ya ha creado automáticamente algunas pantallas. Es de especial importancia la pantalla Frame0, pues es la que aparece en primer lugar al encenderse el terminal. Debido a ello, se va a aprovechar para mostrar un resumen de los valores más relevantes del laboratorio, además de un menú que facilite el acceso al resto de pantallas.

La creación de una nueva pantalla se realiza haciendo clic derecho sobre HMI0 > Add screen. En este caso, la estructura está formada por diez pantallas, por lo que se repite este proceso diez veces.

La primera pantalla está compuesta por un esquema general del LabDER, y un menú superior que permanece en todas las pantallas para facilitar la navegación entre ellas. El objetivo final es el siguiente:

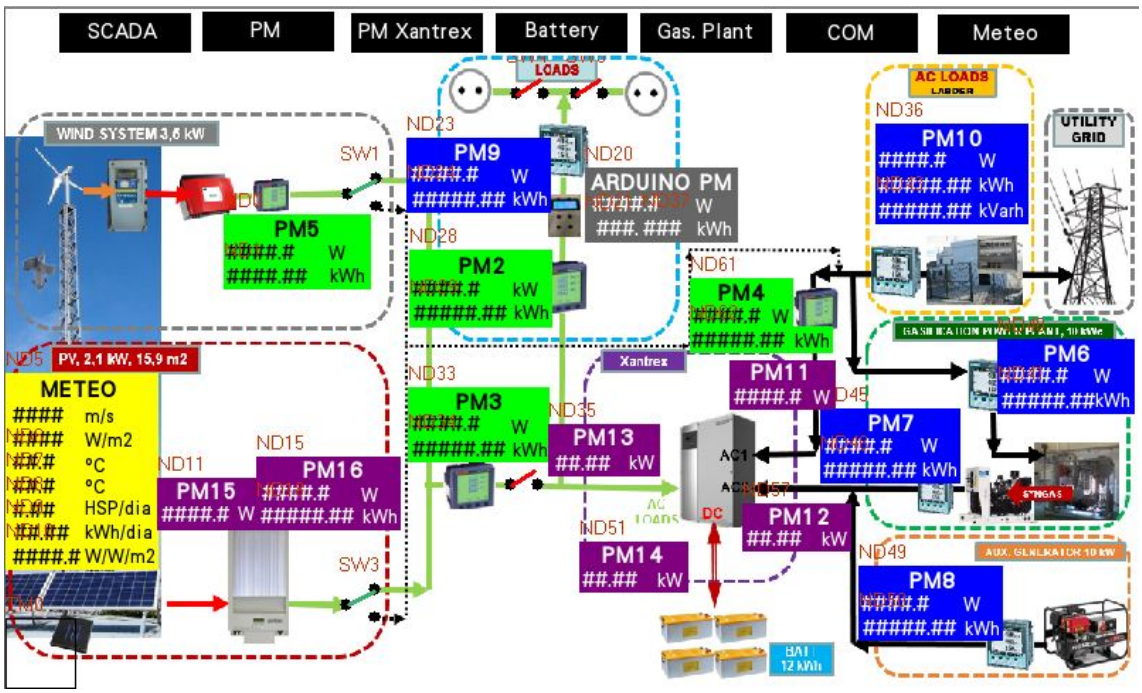


Figura 3.4: Pantalla general de información del LabDER

Para explicar la realización de esta pantalla, primero se van a describir los elementos que la componen, junto con la función que desempeña cada uno.

Imagen de fondo

Draw > Add Graphics… conduce a una pantalla en la que se debe seleccionar crear un gráfico tipo bitmap, pues la imagen que se pretende poner de fondo es un PNG, el cual es un formato en mapa de bits. No es necesario establecer una altura y anchura, pues más tarde se pueden modificar. Lo mismo ocurre con el número de estados, que son de utilidad en otro tipo de elementos, como los botones, pero por ahora se va a dejar en 1.

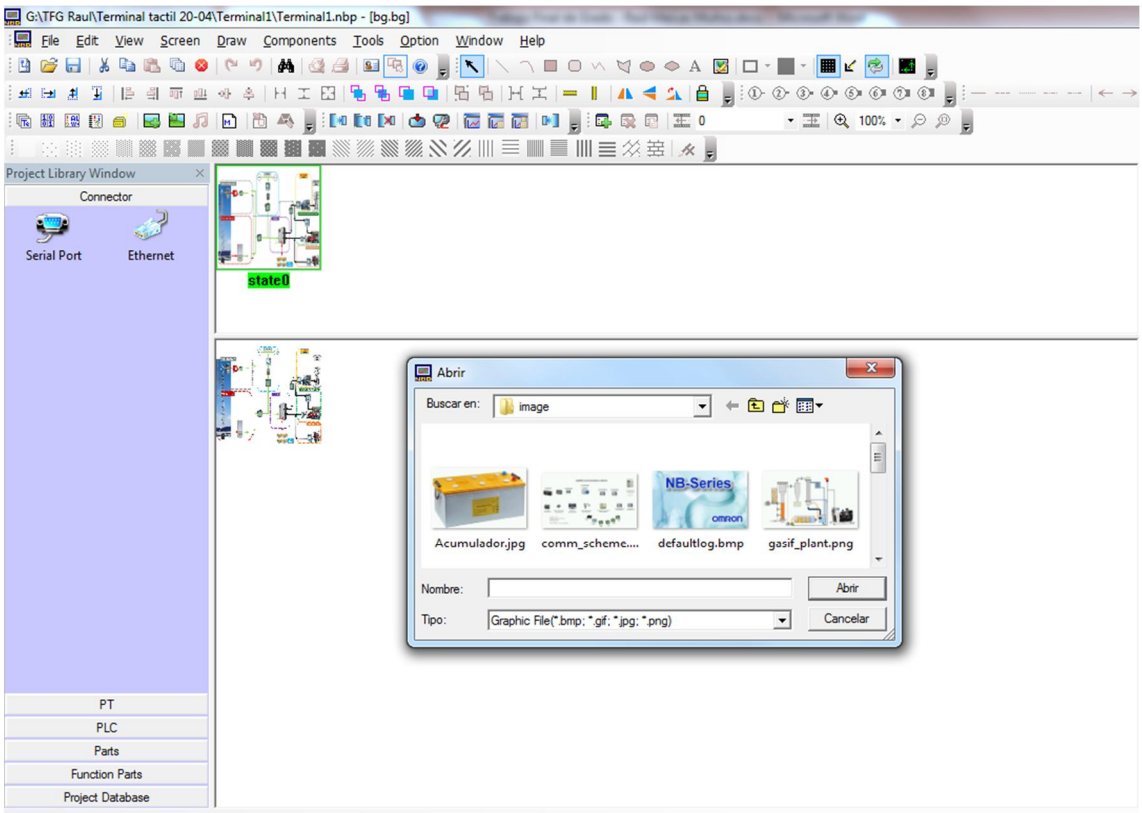


Figura 3.5: Edición de bitmap en NB Designer

Aparece una ventana como la de la figura superior, en la que en la parte de arriba se listan todos los estados disponibles para esa imagen, y abajo se permite la modificación de la imagen haciendo doble clic y buscándola en el equipo.

Tras guardar los cambios, se procede a insertar la imagen de fondo en la pantalla Frame0. Se añade el elemento Bitmap, que se encuentra dentro de Function Parts, al espacio de trabajo. Aparece una pantalla en la cual se selecciona el gráfico que se desee utilizar, y después se mostrará el mismo insertado en nuestra pantalla.

Arrastrando los tiradores que tiene la imagen en sus esquinas, se puede ajustar el tamaño de la misma para que ocupe toda la pantalla y así cumplir la función de imagen de fondo.

Botones de navegación

Antes de la creación del botón propiamente dicho, se va a diseñar la imagen de fondo que poseerá el mismo. Para ello, se pulsa en Draw > Add graphics... pero esta vez se selecciona Vector Graphics, pues se va a dibujar en formato vectorial el fondo del botón.

Además, se necesitará que tenga una forma alargada, por lo que se establecen sus dimensiones en 150 de ancho y 50 de alto.

El botón va a tener dos estados principales: pulsado y sin pulsar. Por ello, se deben crear dos estados del botón, para lo que se hace clic derecho en status0 > Add Status. Se selecciona el primer estado, para poder modificarlo en la parte inferior del programa.

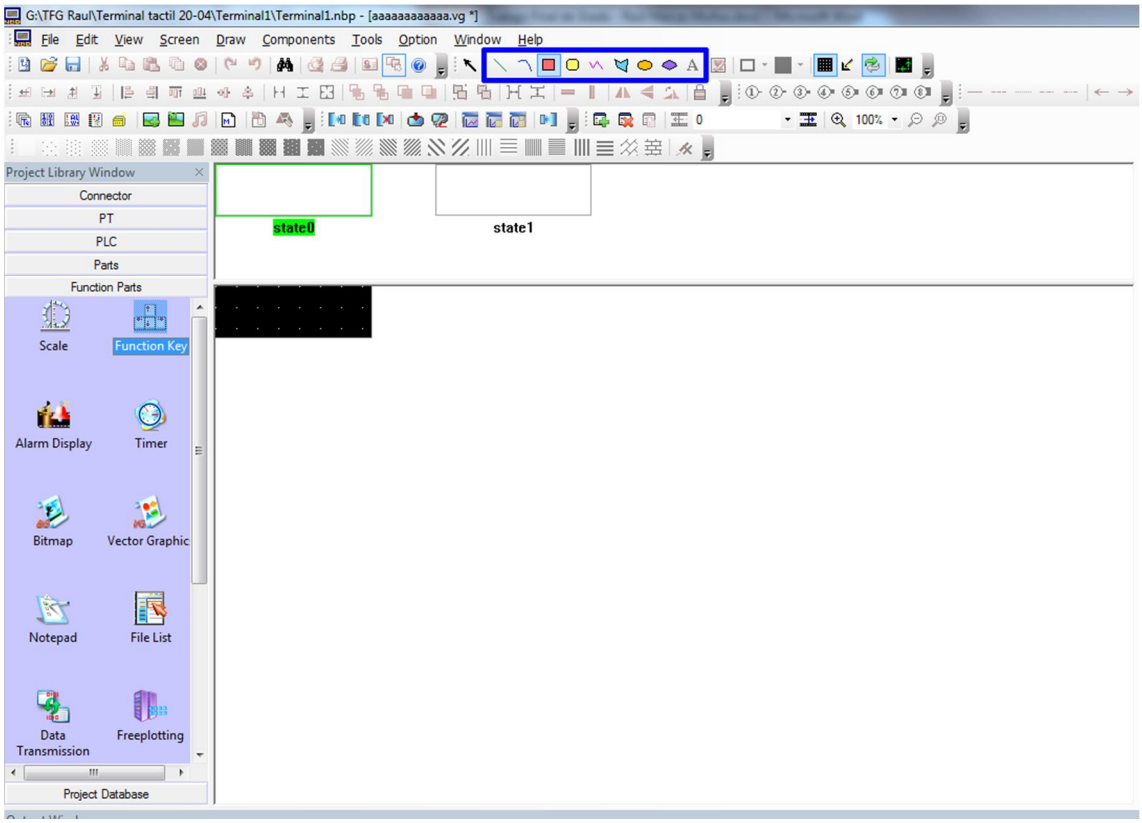


Figura 3.6: Edición de elementos vectoriales en NB Designer

Se va a realizar un botón básico, con un fondo negro y marco blanco, pero el programa posee más opciones en la barra de herramientas remarcada en la figura anterior. En este caso, se va a seleccionar la herramienta rectángulo y se va a rellenar el área de dibujo completamente.

Al hacer doble clic sobre el rectángulo recién creado, se abre un menú de configuración de colores y etiquetas, en el que se le puede dar el aspecto que se prefiera. Para el botón actual, se ha elegido un fondo negro y un borde blanco, para mejorar su visibilidad.

Tras modificar el primer estado, se accede al segundo estado del botón, y se repite el procedimiento anterior para añadir un rectángulo. Esta vez se ha dejado de color gris, para indicar visualmente que se está pulsando. Se guarda y se vuelve a la configuración de la pantalla.

Dentro de Function Parts, se selecciona Function Key y se arrastra hacia el espacio de trabajo. Este es un elemento en el que se puede configurar la respuesta al ser pulsado, por lo que resulta perfecto para un botón. Se abre una ventana en la que se ha de dejar seleccionado Switch

Screens en la lista desplegable, y seleccionar la pantalla a la que se desee acceder tras pulsar ese botón.

Sin cerrar esa ventana, se cambia a la pestaña Label, donde se escribe el texto que posteriormente se visualizará en el botón. Después, se accede a la pestaña Graphics y se selecciona el gráfico de botón que se ha creado anteriormente. Al aceptar, ya se tendrá un botón plenamente funcional.

Se repite este proceso tantas veces como pantallas se tengan, para poder acceder a todas ellas con facilidad. De esta forma, ya estará creado el menú de navegación.

Display numérico

El último elemento que falta por incluir en la pantalla es el más importante: un display numérico que permita visualizar, en tiempo real, el valor que posee la memoria del PLC en un determinado registro.

Para añadirlo a la pantalla, se pulsa y se arrastra el elemento Number Display sobre el espacio de trabajo. Aparecerá un diálogo semejante al de la siguiente figura:

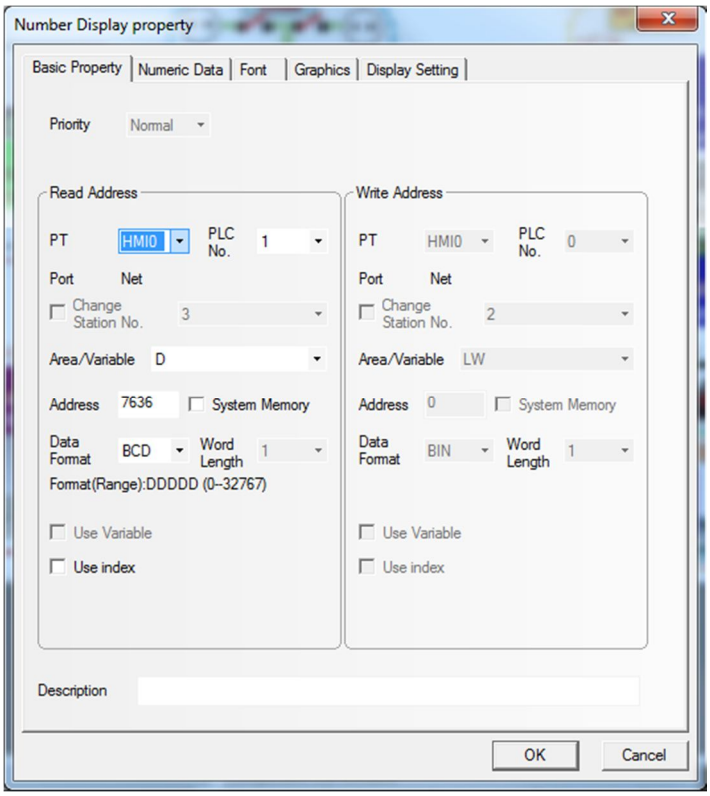


Figura 3.7: Propiedades generales de Number Display

Esta pestaña es donde se realiza la configuración relativa al registro al que se desea acceder. Por ello, se han de revisar diversos parámetros:

- PLC No.: aquí se ha de seleccionar el número del PLC del cual se extrae la información.
- Area/Variable: en esta casilla se selecciona la parte de la memoria del PLC donde está almacenado el valor que se quiere visualizar. En este caso, está en el sector D.
- Address: Es la dirección del registro donde está guardado el valor.

Una vez configurado esto, se cambia a la pestaña Numeric Data, donde aparece lo siguiente:

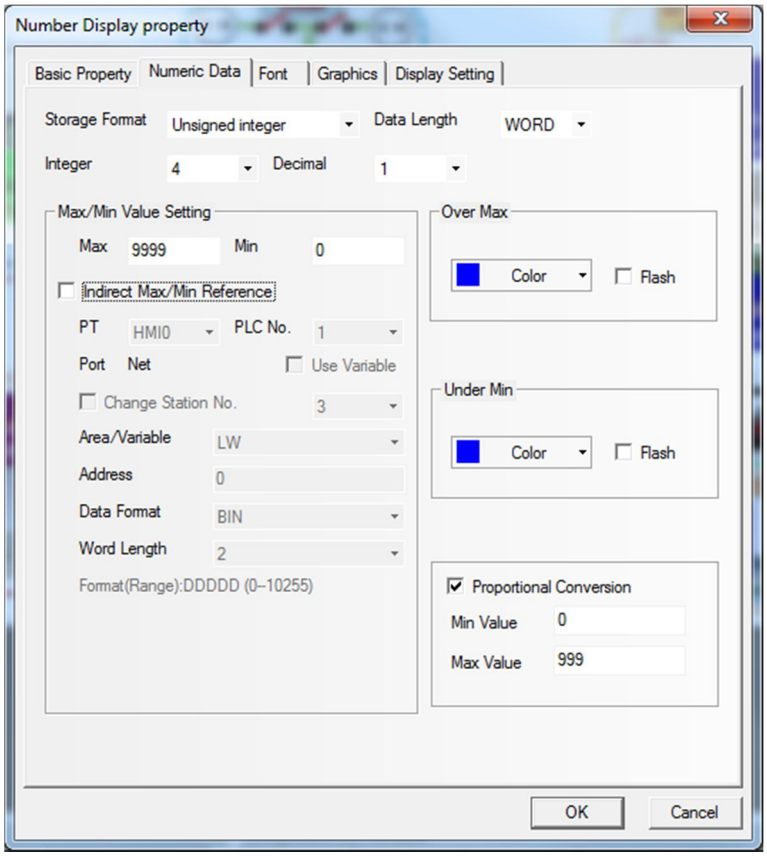


Figura 3.8: Propiedades numéricas del Number Display

El primer campo que se ha de modificar es *Storage Format*. Este es un campo al que se debe de prestar mucha atención, pues el tipo de formato que se escoge aquí debe de ser exactamente el mismo que el que guarda el PLC en su memoria.

Proceso de comprobación del formato numérico

A priori, es difícil saber qué formato tiene un registro, y no es posible averiguarlo desde este programa. Para obtener esta información, se accede a CX-Programmer, donde se ha de establecer conexión con el PLC y colocarlo en modo Monitor.

Una vez hecho esto, se accede a la dirección de memoria que se quiera utilizar, para posteriormente realizar pruebas con diferentes tipos numéricos hasta que el valor se muestre correctamente.

Para realizar esto, se hace clic con el botón derecho en el apartado de la barra inferior donde se pueden añadir variables a visualizar, y se selecciona la opción Edit.

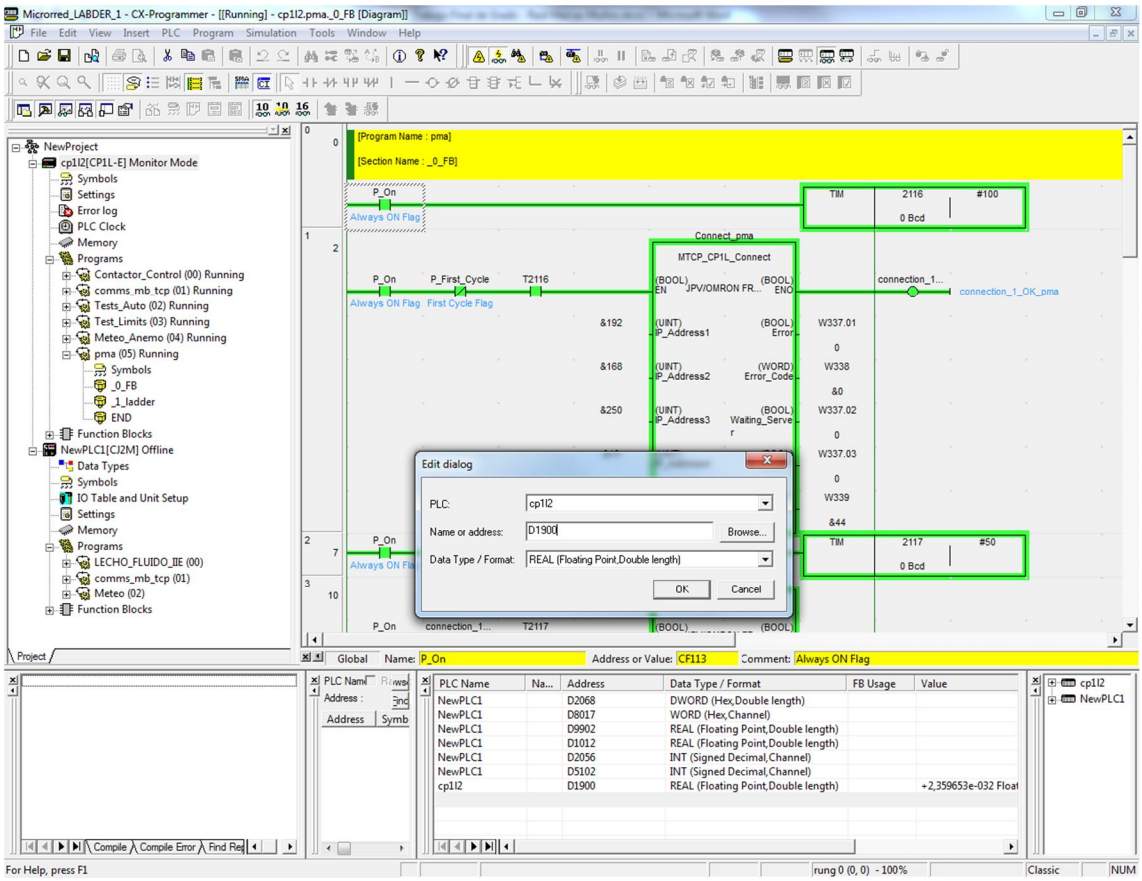


Figura 3.9: Comprobación del formato de número en CX Programmer

En la ventana que aparece, se selecciona el PLC donde está la variable, y se escribe el registro al cual queremos acceder.

El siguiente campo es el que indica el formato del registro. Si no se dispone de esta información, la única opción es probar los diferentes tipos de datos hasta que se obtenga un valor coherente.

Antes de comenzar a probar, se va a realizar una breve explicación de los formatos más utilizados en el PLC (17):

- Float (REAL): es un formato de punto decimal, que tiene un tamaño de 4 bytes. En el laboratorio, ha sido usual encontrar este tipo de registro en equipos de la marca Siemens.
- Double Float (LREAL): es otro formato de punto decimal, que permite almacenar más dígitos que el Float. A causa de esto, ocupa el doble de memoria (8 bytes).
- Unsigned Integer (UINT): es un formato que guarda números enteros sin signo. También ocupa 4 bytes, y ha sido habitual encontrarlo en la mayoría de los equipos: Xantrex, Schneider Electric y Arduino.
- Boolean (BOOL): es un formato que simplemente guarda un bit, un cero o un uno. Habitual en todos los equipos.

Dado que el PLC tiene una memoria con registros de 16 bits, y sabiendo que un byte contiene 8 bits, se puede saber qué espacio ocupa cada tipo de variable:

- Float: ocupa 2 registros.
- Double Float: ocupa 4 registros.
- Unsigned Integer: ocupa 2 registros.
- Boolean: cada registro puede contener 16 variables.

Esta información servirá también para detectar el tipo de variable. Así, si una variable está contenida en el registro W140.1, se deduce que se trata de un booleano.

Tras esta explicación, se procede a averiguar el tipo de variable analizada. Se deja el tipo de variable en REAL y aceptamos, obteniendo un valor bastante cercano a 0, lo cual no concuerda con el rango de valores que debería mostrar. Se continúa repitiendo este proceso hasta que se obtenga un valor coherente:

PLC Name	Na...	Address	Data Type / Format	FB ...	Value
cp1l2		D1900	REAL (Floating Point,Double length)		+2,359653e-032 Float
cp1l2		D1900	UINT (Decimal,Channel)		&2644
cp1l2		D1900	LREAL (Double Floating Point,Qua...		+1,73229897222775e-222 Float

Figura 3.10: Prueba de varios formatos numéricos en CX Programmer

Una vez hallado el tipo de variable, se retorna a NB Designer y se selecciona el mismo en el apartado Storage Format.

Configuración de los decimales

Solamente queda configurar la posición del punto decimal, que dependerá del formato en el que esté almacenado el número en el registro. Esto se controla mediante los menús desplegables Integer y Decimal.

Por último, para los registros en los que el valor de la variable esté multiplicado por un factor, se activa la casilla Proportional Conversion y, estableciendo los máximos y los mínimos en cada apartado, se puede simular un factor de conversión.

Esto suele utilizarse en los registros de tipo Integer, por ejemplo, si se está almacenando una corriente de 0.543 amperios, lo habitual es guardar el valor en el registro multiplicado por 1000, para no tener problemas con los decimales. En este caso, se tendría el valor 543 en el registro, por lo que para mostrar el valor real deberíamos establecer los máximos y mínimos como aparece en la siguiente figura:

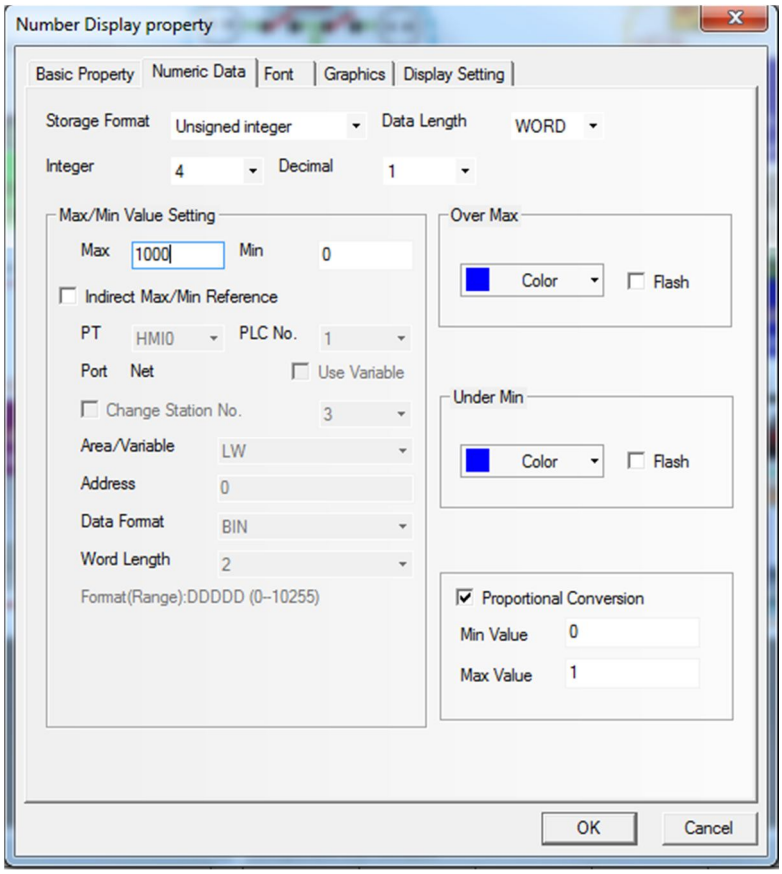


Figura 3.11: Conversión proporcional de valores en el Number Display

Así, el valor máximo de 1000 será convertido a 1, por lo que en realidad se está aplicando un factor de conversión 1:1000.

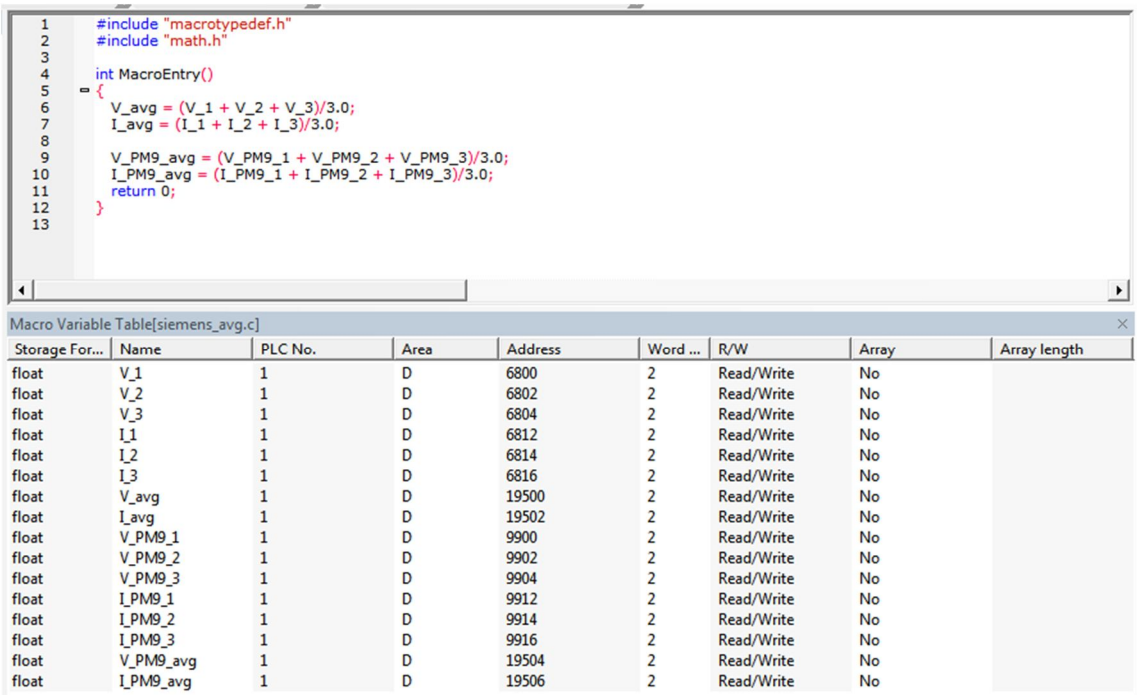
Creación y utilización de macros

A pesar de que las macros son una herramienta muy potente, que permite realizar cualquier tipo de cálculo e incluso tomar decisiones dependiendo de los valores medidos, en este caso se han utilizado únicamente para calcular valores medios de las tensiones y corrientes de fase.

Por ejemplo, en la pantalla PM Siemens, en la que no existe un registro del PLC que recoja la corriente e intensidad medias del PM9 y PM10, se realizará este cálculo mediante una macro.

Para crear una nueva macro, se pulsa en el menú Options > Macrocode... y se asigna un nombre. Después, se abrirá una pantalla semejante a la de la *Figura 3.12: Pantalla de edición de macros*, donde se puede editar el código de la macro.

Antes de escribir código, se deben añadir todas las variables que se vayan a utilizar en la parte inferior de la ventana, que en este caso son relativas a voltajes e intensidades. Además, se deberá encontrar algún registro de memoria vacío en el PLC, para guardar los valores medios que se calculen. En este caso, se han elegido los registros del D19500 en adelante.



```
#include "macrotypedef.h"
#include "math.h"

int MacroEntry()
{
  V_avg = (V_1 + V_2 + V_3)/3.0;
  I_avg = (I_1 + I_2 + I_3)/3.0;

  V_PM9_avg = (V_PM9_1 + V_PM9_2 + V_PM9_3)/3.0;
  I_PM9_avg = (I_PM9_1 + I_PM9_2 + I_PM9_3)/3.0;
  return 0;
}
```

Macro Variable Table[siemens_avg.c]

Storage For...	Name	PLC No.	Area	Address	Word ...	R/W	Array	Array length
float	V_1	1	D	6800	2	Read/Write	No	
float	V_2	1	D	6802	2	Read/Write	No	
float	V_3	1	D	6804	2	Read/Write	No	
float	I_1	1	D	6812	2	Read/Write	No	
float	I_2	1	D	6814	2	Read/Write	No	
float	I_3	1	D	6816	2	Read/Write	No	
float	V_avg	1	D	19500	2	Read/Write	No	
float	I_avg	1	D	19502	2	Read/Write	No	
float	V_PM9_1	1	D	9900	2	Read/Write	No	
float	V_PM9_2	1	D	9902	2	Read/Write	No	
float	V_PM9_3	1	D	9904	2	Read/Write	No	
float	I_PM9_1	1	D	9912	2	Read/Write	No	
float	I_PM9_2	1	D	9914	2	Read/Write	No	
float	I_PM9_3	1	D	9916	2	Read/Write	No	
float	V_PM9_avg	1	D	19504	2	Read/Write	No	
float	I_PM9_avg	1	D	19506	2	Read/Write	No	

Figura 3.12: Pantalla de edición de macros

Para añadir las variables, se pulsa con el botón derecho del ratón en la parte inferior de la pantalla y se selecciona Add Variable. Se abre un cuadro de diálogo en el cual se configura el nombre, dirección y PLC en el cual se almacena la variable, para después aceptar.

Una vez añadidas todas las variables necesarias, el código simplemente se encarga de efectuar la media entre los tres valores que se proporcionan:

```
#include "macrotypedef.h"
#include "math.h"


int MacroEntry()
{
	V_avg = (V_1 + V_2 + V_3)/3.0;
	I_avg = (I_1 + I_2 + I_3)/3.0;


	V_PM9_avg = (V_PM9_1 + V_PM9_2 + V_PM9_3)/3.0;
	I_PM9_avg = (I_PM9_1 + I_PM9_2 + I_PM9_3)/3.0;
	return 0;
}
```

Código 3.1: Macro para calcular el valor medio

Una vez se ha guardado la macro, se incluye la misma en la pantalla en la que la queramos ejecutar. Para ello, se accede a la pantalla de PM Siemens, y se añade un temporizador (Timer). El menú de configuración que aparece es el siguiente:

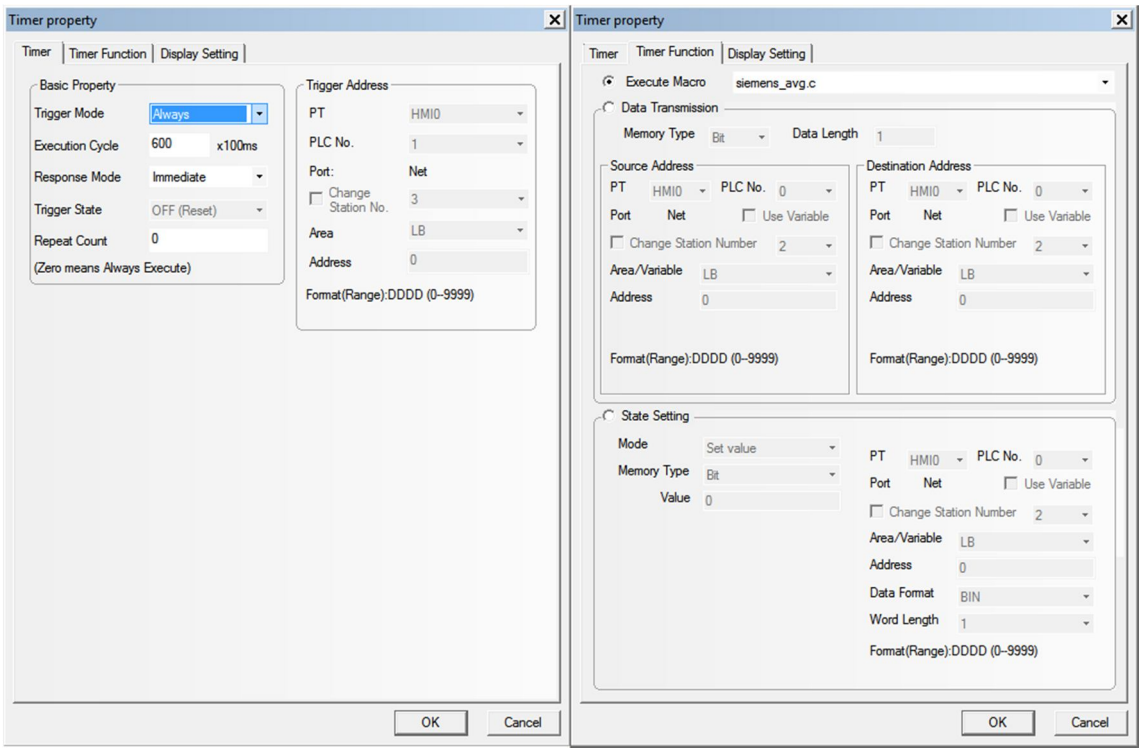


Figura 3.13: Configuración del temporizador para la ejecución de la macro

En la pestaña *Timer*, se debe comprobar que Trigger Mode se mantiene en Always, para posteriormente pasar a la pestaña Timer Function. Es en esta pestaña donde se puede elegir la función que se ejecutará al iniciarse el temporizador: en este caso, se selecciona Execute Macro y se elige la macro recién creada de la lista desplegable.

Lo último que falta por hacer es añadir los Number Display necesarios para mostrar los valores de los registros D19500 en adelante, operación que ya se ha explicado con anterioridad.

Finalización de la pantalla y adición de nuevas pantallas

El proceso de realización del resto de la pantalla es simplemente ir combinando los elementos vistos anteriormente.

La creación de las demás pantallas es muy parecida, por lo que no se va a entrar en más detalles. Sin embargo, se ha realizado una descripción de la información que se muestra en cada pantalla del terminal táctil, en el *Anexo 2: Descripción de las pantallas del terminal táctil.*

Capítulo 4 - Programación del autómata

Objetivo

El objetivo de esta última parte del trabajo es la consecución, por parte del PLC, de la recogida y guardado en memoria de los valores medidos por el analizador. Este proceso se realiza con el ánimo de que el terminal HMI acceda posteriormente a los registros guardados y los muestre por pantalla.

La conexión de la placa Arduino con el PLC se realizará mediante ModBus. El PLC se conectará a la placa, leyendo de manera cíclica los registros almacenados en la misma, y copiándolos en su memoria interna. Para visualizar estos valores, se conectará el terminal táctil HMI al PLC y, mediante un protocolo propietario de OMRON, estos valores se enviarán desde el PLC al terminal táctil, permitiendo su visualización de manera sencilla.

Soporte material

Elementos físicos

- Analizador trifásico Arduino
- Cable Ethernet
- HMI OMRON NB-7WTW01B
- PLC OMRON CP1L

Software

- Arduino IDE
- OMRON Multiway
- OMRON CX-Programmer
- OMRON NB-Designer

Procedimiento

Conexión del analizador a la red local

La red local del laboratorio está formada por varios PLC y equipos medidores, que operan comunicándose mediante diferentes protocolos. Una descripción más detallada puede encontrarse en el apartado de la *introducción Regulación de la microrred mediante un sistema SCADA*.

Este esquema es útil porque relaciona cada uno de los equipos con su dirección IP local. Dado que una dirección IP no puede estar repetida, se ha de escoger una IP que no esté en uso para identificar al analizador construido. En la red local del laboratorio, los equipos se asocian a una IP del tipo 192.168.250.*xxx*, en la que *xxx* representa el identificador único de cada equipo.

Se decide tomar 13 como el identificador del analizador, pues es una dirección que no está repetida. Una vez decidida la IP del analizador, es momento de actualizar su software para posibilitar la comunicación mediante el protocolo Modbus.

Modbus es un protocolo estandarizado de comunicación entre PLCs y otros equipos, que permite el intercambio de información entre un equipo conectado como maestro y otros como esclavos. Concretamente, se va a utilizar el Modbus TCP/IP, que opera en el puerto 502, pues la comunicación se realiza mediante Ethernet.

Adición de comunicaciones mediante Modbus al software Arduino

Para conseguir la comunicación del analizador con el resto de la red, primero es necesario modificar el software del mismo. Para conseguir la conexión, se incluyen las siguientes librerías en el sketch:

- Ethernet.h: controla las funciones que permiten a la placa conectarse a la red.
- Mudbus.h: permite a la placa comunicarse mediante este protocolo (18).

Tras la inclusión de ambas librerías en el programa, el siguiente paso es configurar la conexión Ethernet. Para ello, se han de establecer sus direcciones MAC e IP.

La IP, como ya se ha comentado anteriormente, es una dirección que identifica al equipo que la posee dentro de una red. Por ello, un equipo puede tener direcciones IP diferentes dependiendo de la red a la que se conecte, incluso dependiendo del momento de conexión, gracias al servicio DHCP.

Por otra parte, la MAC es una dirección única para cada equipo. Es su identificador en todas las redes, y es un parámetro que no puede modificarse.

Una vez se tiene clara la teoría, ya se puede comenzar a escribir código. Antes del bucle de setup se colocarán las siguientes instrucciones:

```
Mudbus Mb;


byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 250, 13);
```

Código 4.1: Cabecera de la programación de la conexión del analizador mediante Modbus

Las cuales sirven para crear una instancia de Mudbus, y para almacenar en variables la MAC y la IP del analizador.

Una vez dentro del setup, se inicializa la conexión Ethernet:

```
Ethernet.begin(mac, ip);
```

Código 4.2: Bucle setup de la programación de la conexión del analizador mediante Modbus

Y por último, se inicializa la instancia de Mudbus que se ha creado anteriormente y se añade un valor al registro:

```
Mb.Run();
Mb.R[0]  = Vf_ph1;
```

Código 4.3: Bucle loop de la programación de la conexión del analizador mediante Modbus

Con esta instrucción se crea el registro 0 y se le asigna el valor de *Vf_ph1,* apto para ser leído a través de Modbus por cualquier equipo que se conecte al analizador.

Simplemente repitiendo la última instrucción, se pueden crear todos los registros que se necesiten y asignarlos a las variables que representan los valores medidos por el analizador; para así poder comunicar toda la información que está obteniendo al PLC.

Comprobación de la conexión

Una vez se ha actualizado el software de la placa, primero se ha de comprobar que la comunicación funciona correctamente, antes de programar el PLC.

Para ello, se ha utilizado el software OMRON Multiway, que es una aplicación de comunicación multiprotocolo. Entre ellos, se encuentra el protocolo Modbus, mediante el cual se va a intentar la conexión con el analizador.

Una vez abierto Multiway, en la pestaña Modbus, se rellena el campo IP con la dirección asignada previamente a la placa Arduino (192.168.250.13), comunicándose a través del puerto 502, que es el que se utiliza para el protocolo Modbus sobre TCP/IP.

Al pulsar el botón de PING, Multiway envía un paquete de datos al analizador. Si el botón se vuelve verde, es que la comunicación ha sido correcta y el paquete ha sido recibido. Si no lo es, se ha de comprobar que el cable Ethernet del analizador esté correctamente conectado y que el software de la placa haya sido actualizado.

Dado que el PC desde el que se utiliza el Multiway es el que va a solicitar información del analizador, se ha de colocar el PC como cliente, por lo que se ha de seleccionar esa casilla en el programa. Una vez hecho esto, se pulsa en Connect.

Por último, se ha de solicitar la información al analizador mediante un comando Modbus. La estructura de estos comandos (19) es parecida, tanto para la solicitud como para la respuesta.

El comando que se va a utilizar para solicitar uno de los registros del analizador es el siguiente, con su posterior explicación:

Comando	01 03 0037 0001
Código	**Descripción**
01	La dirección del esclavo (suele ser 1)
03	Código de función: cada par numérico está asociado a una función diferente, por lo que, dependiendo del número, se realizará una acción u otra: ya sea leer un registro, escribirlo u otras opciones.
0037	Número de registro, en hexadecimal
0001	Número de registros a leer a partir del registro solicitado

Tabla 4.1: Códigos de solicitud Modbus

Al enviar el comando de solicitud o pregunta, si no existe ningún problema se recibirá poco después la respuesta del analizador, con un formato parecido:

Comando	01 03 02 0197
Código	**Descripción**
01	La dirección del esclavo (suele ser 1)
03	Código de función, en este caso leer registros de salida analógicos
02	Número de bytes que devuelve a continuación
0197	Valor, en hexadecimal, que contiene el registro

Tabla 4.2: Códigos de respuesta Modbus

Si el valor que devuelve el Multiway coincide con el valor que muestra la pantalla del analizador, significa que la conexión está configurada correctamente y se puede proceder a la conexión del analizador con el PLC.

Programación del PLC

Ahora que el analizador ya está conectado a la red local, se ha denumodificar el software del PLC para que se comunique con el Arduino y le solicite la información que está recogiendo. Para ello, se necesita un programa que se ejecute de manera cíclica, cuya función consistirá en solicitar los valores que mide el analizador en tiempo real y guardarlos en su memoria interna.

Como se va a utilizar el PLC OMRON CP1L, la programación del mismo se va a realizar mediante la aplicación que OMRON ha diseñado para sus PLC: el software CX-Programmer.

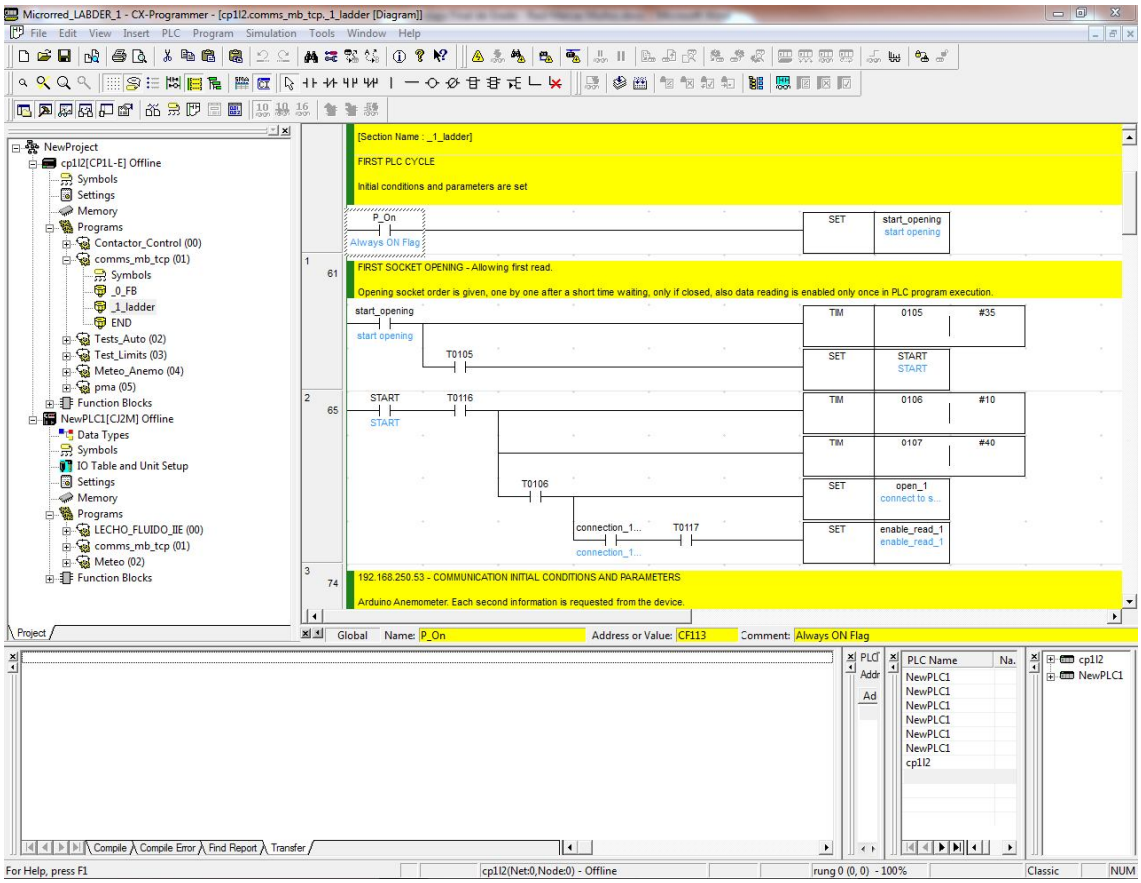
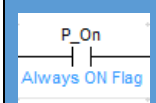


Figura 4.1: Pantalla general del CX-Programmer

Una vez abierto el programa, se puede observar que está dividido en diferentes paneles, como se muestra en la figura superior. La columna izquierda presenta especial relevancia, pues muestra los PLCs disponibles en este proyecto (CP1L y CJ2M), y dentro de cada uno, se listan los registros de memoria, los programas y los bloques de función que contiene.

Los programas que se pueden crear con este software se basan en el uso de diagramas de contactos o lenguaje Ladder. Este es un lenguaje de programación gráfico muy popular dentro de los autómatas programables, debido a que se basa en los esquemas eléctricos de control clásicos (20). Se van a describir a continuación los componentes principales de los diagramas de contactos:

Elemento	Descripción
Contacto 	Un contacto es un elemento asociado a una dirección de tipo BOOL del PLC, que se activa o se desactiva dependiendo del estado o forma de cambio del valor del bit. Hay varios tipos: • Normalmente abierto: cuando el estado del bit es 0 el circuito permanece abierto, y cuando es 1 se cierra. • Normalmente cerrado: cuando el estado del bit es 1 el circuito permanece abierto, y cuando es 0 se cierra. • Bajada: cuando el bit pasa de 1 a 0, se activa. • Subida: cuando el bit pasa de 0 a 1, se activa.

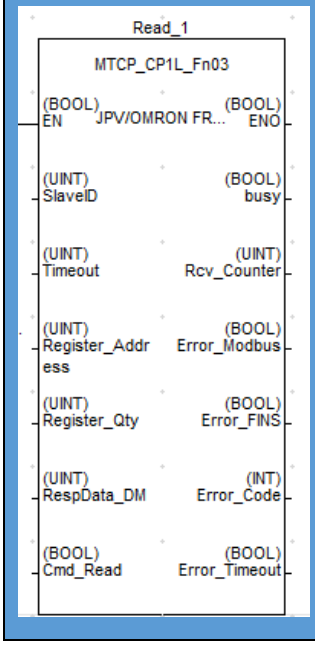
Instrucción 	Una instrucción es una orden que se ejecuta al recibir un bit positivo. Las instrucciones que se utilizan en este programa son: • SET: Coloca un bit como positivo o 1. • RSET: Coloca un bit como negativo o 0. • TIM: Crea un temporizador de un periodo configurable. • MOV: Mueve información entre registros del PLC.
Bobina 	Se activa cuando la instrucción recibida es un bit positivo.
Bloque de función 	Un bloque de función es un programa Ladder encapsulado. Son partes complejas de programa que se pueden instanciar las veces que sean necesarias, evitando tener diagramas de contactos repetidos. Además, los bloques de función se pueden guardar por separado y compartir, por lo que es posible utilizar funciones desarrolladas por otras personas e integrarlas en el proyecto, consiguiendo un ahorro importante de tiempo. En el diagrama de contactos realizado más abajo se han utilizado dos bloques de función: uno para conectar el PLC al analizador y otro para leer los valores del registro.

Tabla 4.3: Descripción de los elementos del diagrama Ladder

Para realizar el programa, se necesitará almacenar variables durante la ejecución del mismo, por lo que se debe encontrar un espacio de la memoria del PLC que no esté ocupado todavía. Para ello, una vez se esté online, se puede acceder a los registros de memoria del PLC haciendo doble clic sobre su icono del panel izquierdo, y colocándolo en modo monitor pulsando Online > Monitor. En esta pantalla se pueden visualizar qué registros están ocupados y cuáles no, facilitando la tarea de elegir en qué registros se van a almacenar las variables.

Una vez se ha terminado la introducción teórica, y se han elegido los registros en los que se van a almacenar los valores del programa, se procede a realizar el diagrama de contactos requerido. Para ello, se han de colocar los contactos, instrucciones y bloques de función a partir de la barra de herramientas superior de CX-Programmer. El diagrama de contactos completo, que se procede a explicar a continuación, puede consultarse con más detenimiento en el *Anexo 4 - Diagrama de contactos del PLC* CP1L.

Descripción del diagrama de contactos

Al encenderse el PLC y transcurrir un tiempo definido por el temporizador, se accede al bloque de función de conexión. Este bloque realiza la conexión a partir de varios parámetros de entrada: los 4 números que forman la IP del analizador, además de un bit que permite el reinicio de la conexión si surge algún problema. Si la conexión tiene éxito, se activa la bobina *connection_1_OK_pma*, en el caso contrario devolviendo un error y almacenándose el código del mismo en el espacio de memoria *W338*.

Se activa otro pequeño temporizador, para no empezar a leer justo después de conectarse.

Una vez el PLC se ha conectado al analizador, se procede a leer los valores del registro. Para ello, se utiliza otro bloque de función, al que se le suministran como parámetros de entrada:

- El número de identificación del esclavo
- El tiempo de espera para considerar que la lectura ha resultado fallida.
- El registro del analizador desde el que se empieza a leer, en este caso, a partir del 0.
- La cantidad de registros que se desean leer, en este caso son 50.
- La parte de la memoria del PLC donde se van a almacenar esos registros. Como se ha descrito anteriormente, previamente a colocar una dirección de memoria cualquiera se ha de comprobar que está vacía, para evitar errores en el funcionamiento del PLC y los equipos conectados al mismo. En este caso se guardarán registros a partir del 1900.
- Un bit que, al activarse, indica el inicio de la lectura. Este bit es útil para hacer el proceso de lectura un proceso cíclico.

Este bloque de función devolverá como parámetros de salida:

- Un bit que indique si ha tenido éxito el proceso de lectura.
- Un bit que indica si el proceso está ejecutándose (busy).
- Un entero que almacena el total de registros recibidos.
- Un bit que indica error en la comunicación Modbus.
- Un bit que indica error en la comunicación FINS (otro tipo de comunicación de OMRON).
- Un entero que guarda el código de error.
- Un bit que indica error porque el tiempo de espera ha terminado.

Una vez el programa haya recorrido hasta aquí el diagrama de contactos, si no ha surgido ningún error, el PLC habrá leído todos los valores del registro del analizador y los habrá introducido en su memoria, en el lugar que indicado en el bloque de función de lectura.

Aun así, se ha de tener en cuenta que ha sido un proceso que solamente se ha ejecutado una vez, por lo que es necesario convertir este programa en cíclico para que recoja los valores del analizador a medida que se van midiendo.

Para ello, se van a utilizar diferentes instrucciones y temporizadores, que básicamente poseen la tarea de reiniciar el proceso de lectura, por lo que se ha prestar atención a los bits que almacenan información sobre el mismo. En concreto, existen dos bits muy relevantes en este proceso:

- START_1_pma: Es el bit que controla el inicio de la lectura.
- BUSY_1_pma: Cuando está ON, indica que está leyendo, y cuando está OFF, que ha terminado el proceso de lectura.

En general, el comportamiento que tiene el programa es que, cuando detecta que *BUSY_1_pma* pasa de ON a OFF, significa que ha terminado de leer, por lo que activa el bit *START_1_pma*. Este bit inicia el proceso de lectura del bloque de función, por lo que se consigue un bucle infinito, que lee los valores del registro continuamente si no aparece ningún error.

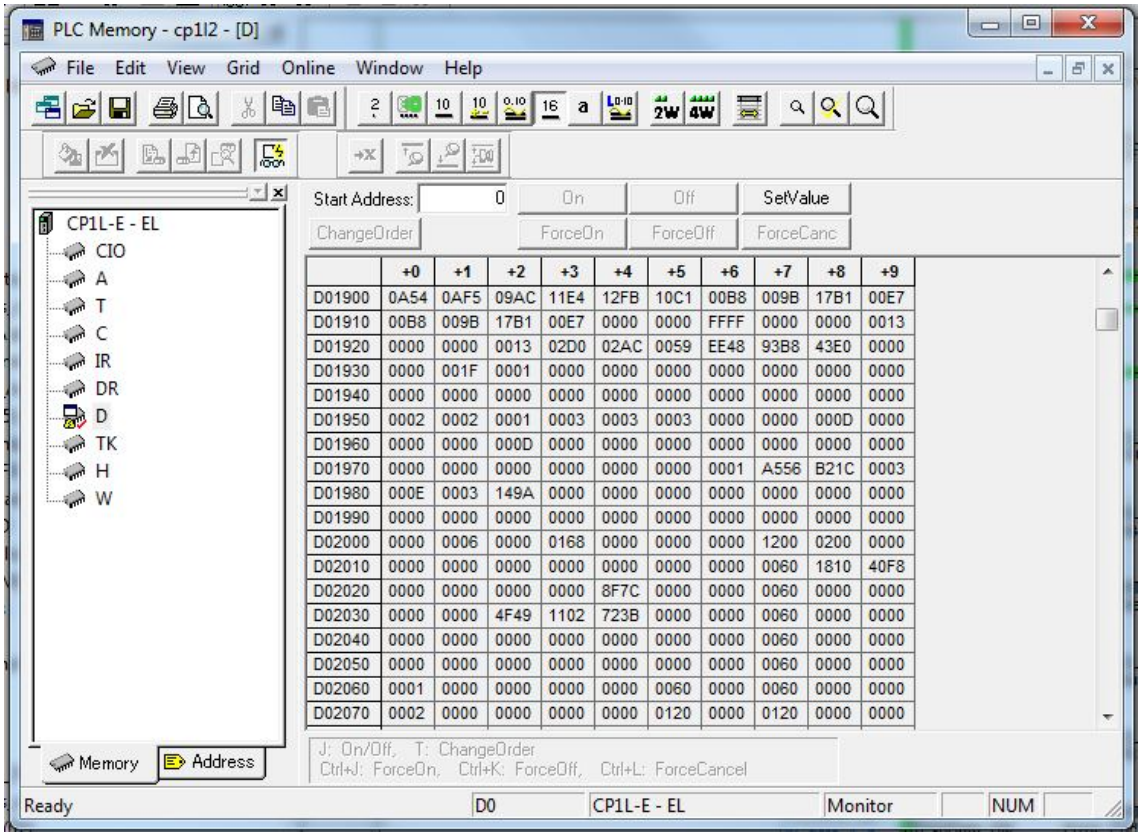
Carga del programa al PLC y comprobación de funcionamiento

Para finalizar, se debe subir este programa al PLC. Al hacer clic en PLC > Work online, el ordenador se conecta con PLC, y después se pulsa en PLC > Transfer > Transfer to PLC...

Al finalizar este proceso, se tendrá el diagrama de contactos cargado en el PLC.

Para comprobar su correcto funcionamiento, una vez se está online, se accede a la memoria del PLC haciendo doble clic en su icono del panel izquierdo. Se abrirá una ventana como la siguiente, en la que aparecen todos los sectores de memoria del PLC. En este caso, se entra al sector D, ya que se han almacenado los registros a partir de D1900.

Para monitorizar los valores en tiempo real, se pulsa en Online > Monitor. Se busca el registro de memoria en el que se deberían estar guardando los datos del analizador, y se comprueba el valor registrado. Si el proceso ha tenido éxito, el valor de este registro debería permanecer en constante actualización, y coincidir con el mostrado en la pantalla del analizador.



	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
D01900	0A54	0AF5	09AC	11E4	12FB	10C1	00B8	009B	17B1	00E7
D01910	00B8	009B	17B1	00E7	0000	0000	FFFF	0000	0000	0013
D01920	0000	0000	0013	02D0	02AC	0059	EE48	93B8	43E0	0000
D01930	0000	001F	0001	0000	0000	0000	0000	0000	0000	0000
D01940	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
D01950	0002	0002	0001	0003	0003	0003	0000	0000	000D	0000
D01960	0000	0000	000D	0000	0000	0000	0000	0000	0000	0000
D01970	0000	0000	0000	0000	0000	0000	0001	A556	B21C	0003
D01980	000E	0003	149A	0000	0000	0000	0000	0000	0000	0000
D01990	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
D02000	0000	0006	0000	0168	0000	0000	0000	1200	0200	0000
D02010	0000	0000	0000	0000	0000	0000	0000	0060	1810	40F8
D02020	0000	0000	0000	0000	8F7C	0000	0000	0060	0000	0000
D02030	0000	0000	4F49	1102	723B	0000	0000	0060	0000	0000
D02040	0000	0000	0000	0000	0000	0000	0000	0060	0000	0000
D02050	0000	0000	0000	0000	0000	0000	0000	0060	0000	0000
D02060	0001	0000	0000	0000	0000	0060	0000	0060	0000	0000
D02070	0002	0000	0000	0000	0000	0120	0000	0120	0000	0000

Figura 4.2: Visualización de los registros de la memoria del PLC

Terminal HMI

Una vez se ha configurado correctamente la conexión entre PLC y analizador, y se poseen los datos actualizados constantemente en la memoria del PLC, se va a proceder a mostrar los datos registrados en la pantalla del terminal HMI.

Los pasos a seguir son prácticamente los mismos que en la configuración realizada en el capítulo anterior, pero con la diferencia de que se va a conectar otro modelo de PLC.

Se inicia el software NB-Designer, abriendo el proyecto de la pantalla realizado previamente. En la pantalla de conexión de los terminales y PLCs, se procede a añadir otro PLC a la red. Para ello, se añade el modelo OMRON CP Series Ethernet, que es el modelo que representa la familia del PLC CP1L. Al hacer doble clic sobre él, se accede a su pantalla de configuración. El único dato a modificar es la dirección IP del PLC, la cual se puede encontrar en el apartado de la introducción *Regulación de la microrred mediante un sistema SCADA*.

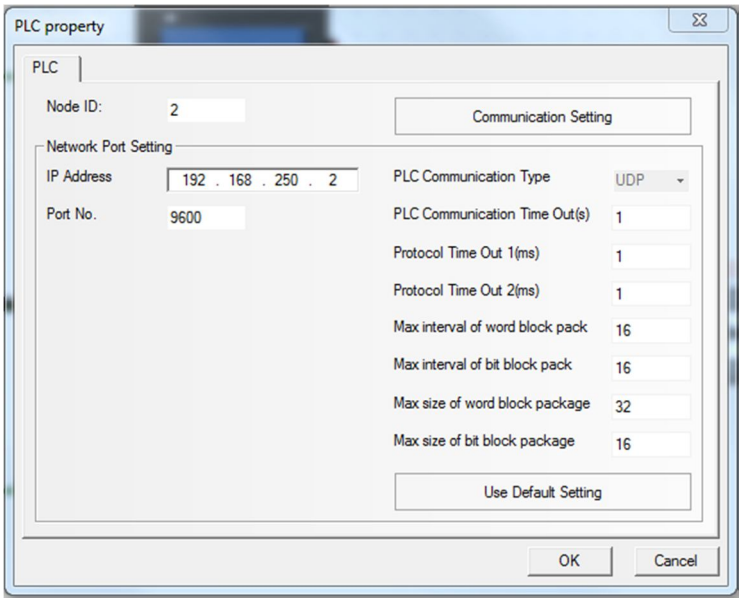


Figura 4.3: Adición de un PLC en NB Designer

Después, se ha de conectar el PLC mediante Ethernet en el esquema del programa. Tras hacer doble clic en la línea verde de la izquierda, aparece el diálogo Communication Setting. Al pulsar en Add, aparece una pantalla para añadir un nuevo elemento a la red. Se selecciona PLC, dejando el resto de la configuración como viene por defecto.

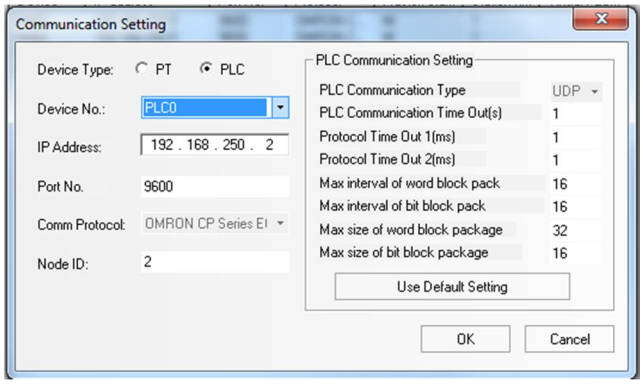


Figura 4.4: Configuración de la conexión Ethernet del PLC en NB Designer

El problema reside en que OMRON dispone de varios protocolos de comunicación dependiendo de la familia a la que pertenecen. La familia CJ y la CP no comparten el mismo protocolo de comunicación, por lo que no pueden transferirse información directamente entre sí. No obstante, esto no quiere decir que el terminal táctil no sea capaz de mostrar la información relativa a ambos PLC: la solución reside en que se ha de indicar a la pantalla qué protocolo debe de utilizar con cada uno de ellos.

Se accede de nuevo a la ventana Communication Setting, y se añade el protocolo de comunicación OMRON CP Series Ethernet a la pantalla, como se observa en la siguiente figura.

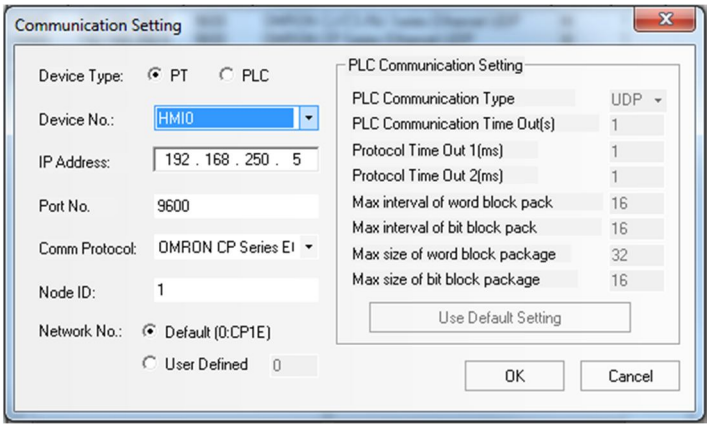


Figura 4.5: Configuración de la conexión Ethernet del terminal HMI en NB Designer

Una vez finalizado el proceso, se obtendrá un esquema de comunicaciones como el de la siguiente figura. Como se puede observar, se listan 4 entradas en este registro, a pesar de que solamente se dispone de 3 equipos. Esto ocurre porque la pantalla establece dos protocolos de comunicación independientes: uno con el PLC CP1L y otro con el PLC CJ2M.

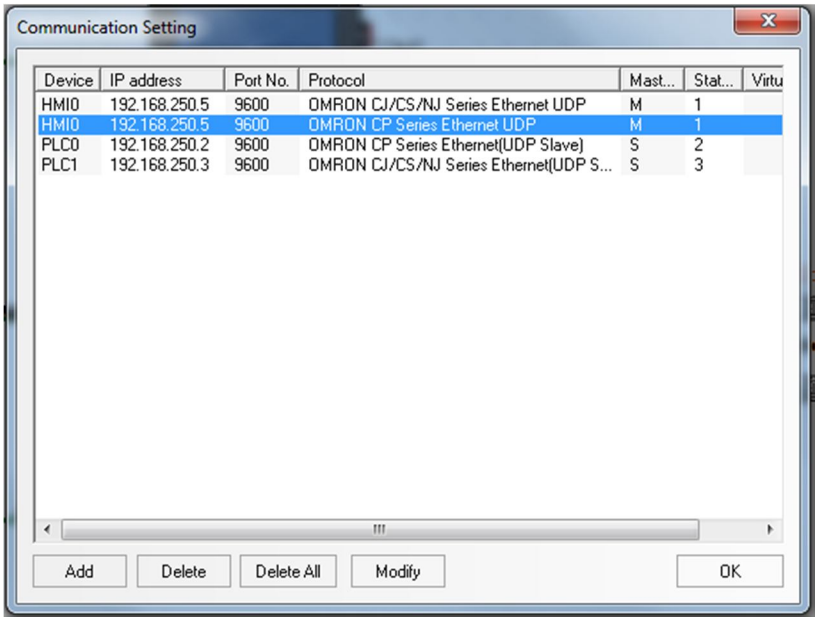


Figura 4.6: Communication Setting - Visualización general de las conexiones entre PLC y HMI

Una vez se haya conectado el PLC, se ha de crear otra pantalla en el terminal para la visualización de los valores del analizador. Este proceso se explica detalladamente en el *Procedimiento del Capítulo 3- Terminal táctil de monitorización del LabDER.*

Capítulo 5 – Resultados

En este apartado se van a describir con detalle todos los resultados que se han obtenido, comenzando por la descripción del dispositivo Arduino construido. Por otro lado, también se comentará el resultado de la creación del terminal táctil para la monitorización de los valores del laboratorio, y la conexión e integración en la microrred del analizador Arduino.

Analizador trifásico Arduino

En esta parte se describen los resultados del diseño, montaje y programación del dispositivo construido a lo largo de capítulos anteriores, explicando en profundidad tanto su diseño y conexiones físicas, como el funcionamiento del mismo y el presupuesto de su construcción.

Construcción física

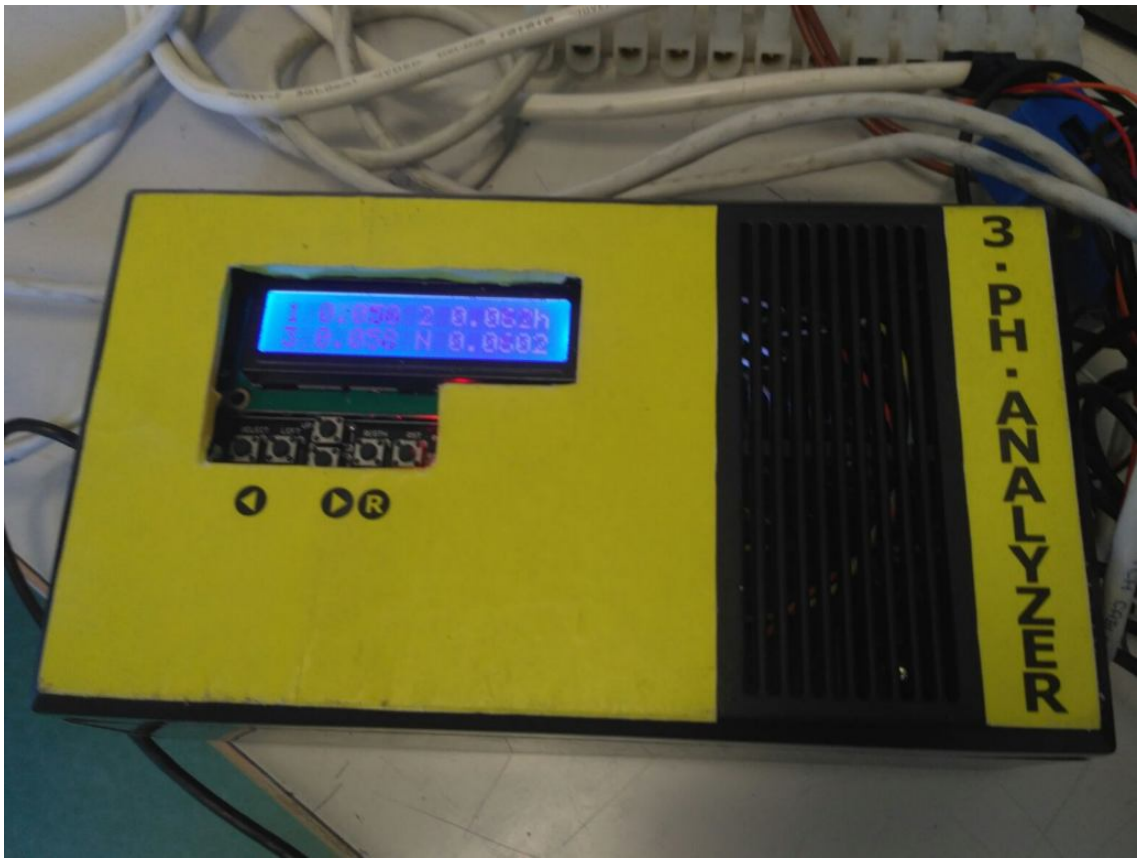


Figura 5.1: Aspecto final del analizador Arduino

En la figura superior, se puede observar el aspecto que tiene el analizador trifásico Arduino una vez finalizada su construcción. El resultado final es un dispositivo compuesto por una caja de montaje de dimensiones 200x112x71mm, provisto de una pantalla LCD en la parte frontal y de los componentes necesarios para su funcionamiento en su interior.

La distribución de los componentes en el interior de la caja se puede observar en la *Figura 5.2: Interior del analizador Arduino*. Dado que el elemento que disipa más energía en forma de calor

en este analizador son los transformadores, se han colocado en la zona de la rejilla de ventilación para facilitar su refrigeración.

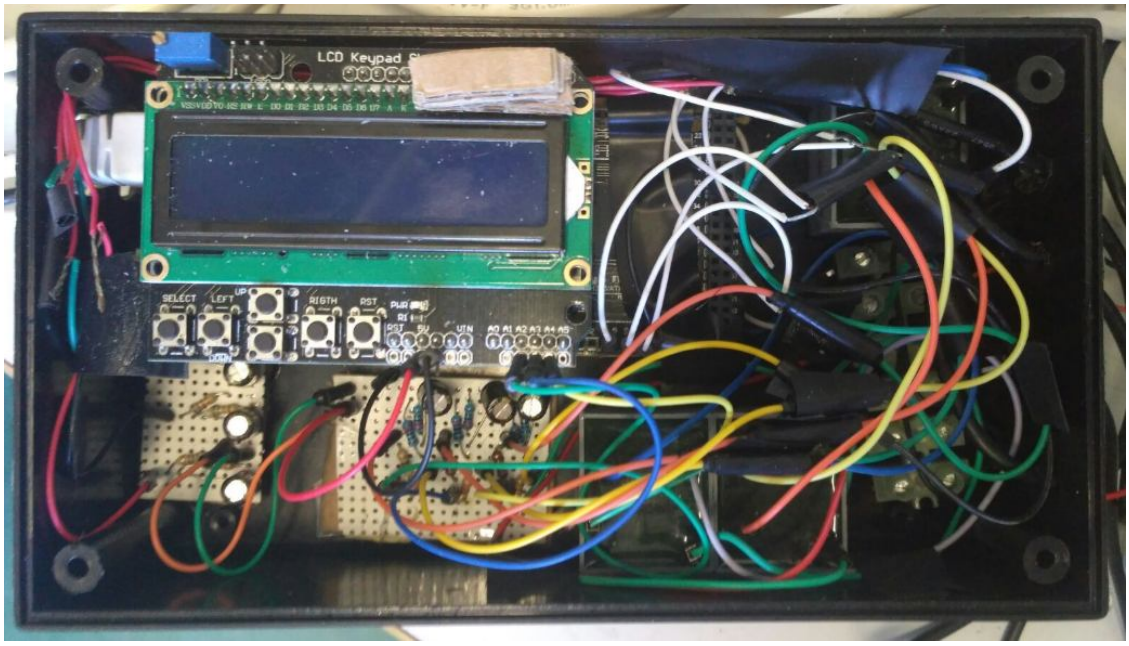


Figura 5.2: Interior del analizador Arduino

En la parte izquierda, se tiene el conjunto formado por la placa Arduino y los dos shields que se le han colocado para ampliar sus funcionalidades. En la parte inferior, se observan los circuitos de medición de tensión y de corriente, soldados en una placa de baquelita.

Por último, el apartado de conexiones del analizador se puede dividir en dos partes: conexiones para medición y conexiones para alimentación y comunicación. En la *Figura 5.3: Conexiones de medición del analizador* y en la *Figura 5.4: Conexiones de comunicación y alimentación del analizador* se muestra el diseño de las mismas.

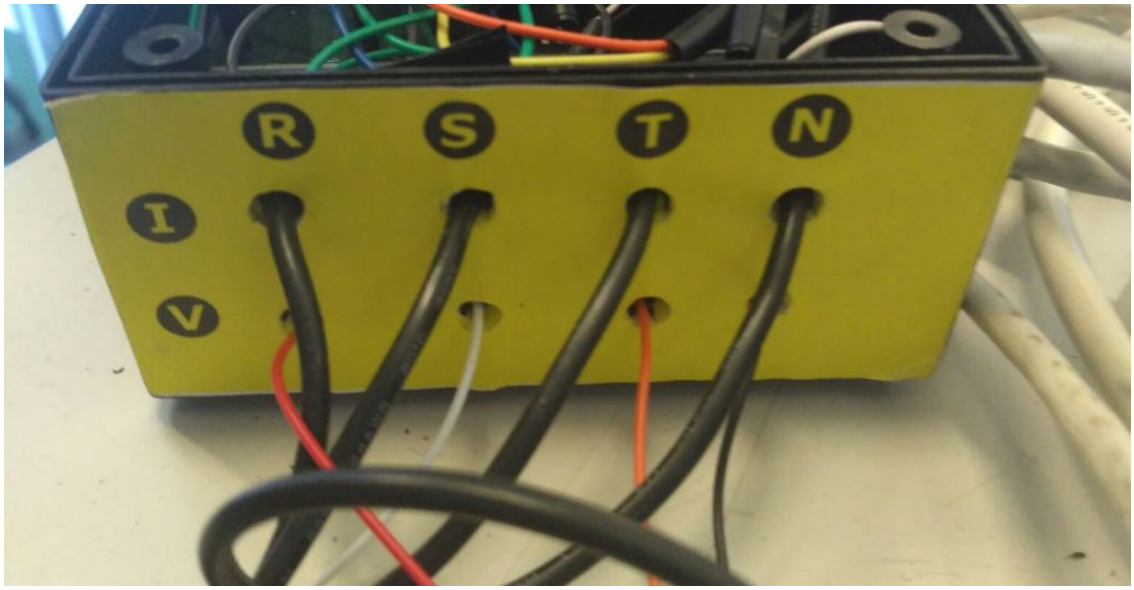


Figura 5.3: Conexiones de medición del analizador

Figura 5.4: Conexiones de comunicación y alimentación del analizador

Como se puede observar, el dispositivo cuenta con 4 cables para la medición de la corriente (tres para cada una de las líneas, y otro para el neutro) y otros 4 cables que permiten la evaluación de la tensión entre cada una de las fases y el neutro.

El apartado de comunicaciones está representado por la conexión Ethernet de la que dispone el analizador en su costado izquierdo, junto con las dos conexiones de alimentación integradas: mediante un cable USB - micro USB o mediante una fuente de alimentación de 9V.

Funcionamiento

Para poner en funcionamiento el analizador, se conecta a una fuente de tensión, ya sea a un ordenador mediante un cable USB o a una fuente de alimentación. El tiempo de arranque del analizador está en torno a los 5 segundos, como se demuestra al ejecutar la función *millis()* en la primera iteración del bucle principal.

Para la medición de la corriente, se cierran las pinzas amperimétricas alrededor de cada una de las líneas y del neutro. Para la tensión, es necesario conectar cada uno de los terminales del analizador entre cada una de las fases y el neutro, pues se ha de tener en cuenta que el presente dispositivo realiza las mediciones para una conexión en estrella.

Una vez se conecta todo, los valores medidos se muestran en la pantalla LCD. Esta pantalla es un display de 16x2 caracteres que cuenta con una botonera en su parte inferior para facilitar la navegación entre los diferentes valores que se están midiendo. Se utilizan los botones de izquierda y derecha para avanzar y retroceder entre las diferentes pantallas, y el botón de reset para efectuar el reinicio del analizador. Para más detalles acerca del diseño y de la información contenida en cada una de estas pantallas, se recomienda dirigirse al

Anexo 2 - Diseño de las pantallas del analizador.

Prueba de precisión del analizador

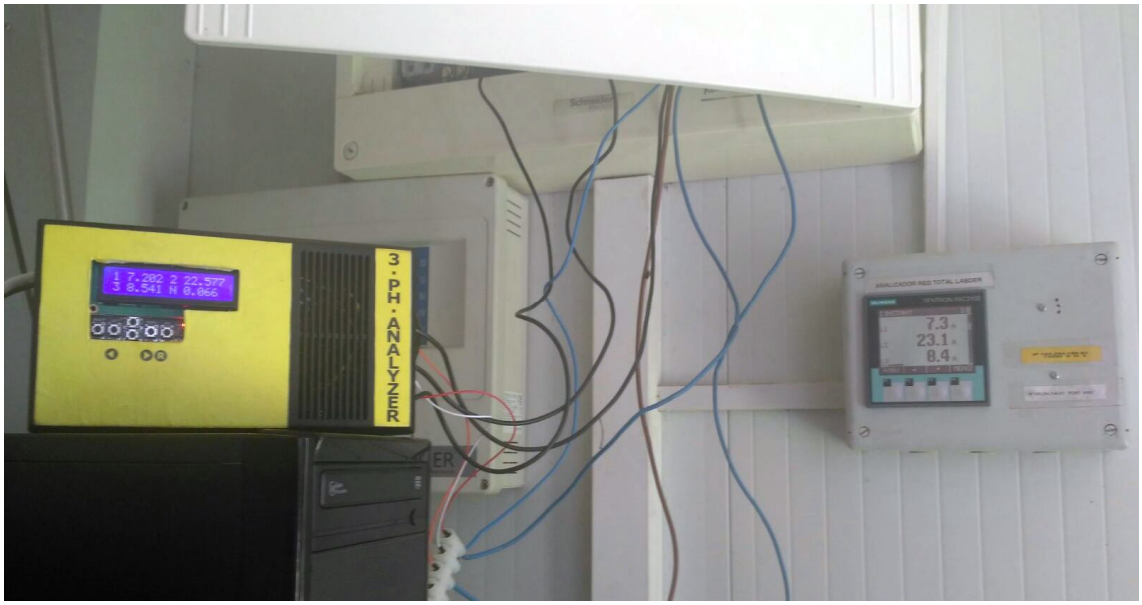


Figura 5.5: Ambos analizadores realizando mediciones simultáneamente

En este apartado, se van a comparar las mediciones que realiza el analizador Arduino con las del Siemens PAC3200, con el objetivo de hallar el error relativo que.

Para ello, se han realizado varias pruebas con diversos niveles de carga, para observar el comportamiento del analizador Arduino en diferentes condiciones de operación. Las cargas que se han utilizado son las siguientes:

- Sistema de automatización: es el consumo mínimo del laboratorio, con todos los equipos de la red local conectados.
- Iluminación: el conjunto de puntos de luz instalados en el laboratorio.
- Dos estufas de 1.8KW: se trata de estufas eléctricas monofásicas, que se utilizan para realizar pruebas de carga. Al estar equipadas con resistencias, su conexión implica el aumento del factor de potencia de la instalación.
- Estufa de 5KW: es una estufa eléctrica trifásica, que también se usa para pruebas de carga. Como las estufas del punto anterior, también son elementos resistivos.

Estas cargas se han combinado de la siguiente manera para realizar las pruebas de medición:

	Sistema de automatización	**Iluminación**	**Dos estufas de 1.8KW**	**Estufa de 5KW**
Medición 1	Sí	No	No	No
Medición 2	Sí	Sí	No	No
Medición 3	Sí	No	Sí	No
Medición 4	Sí	Sí	Sí	No
Medición 5	Sí	Sí	Sí	Sí
Medición 6	Sí	No	Sí	Sí

Tabla 5.1: Configuraciones de las pruebas de medición

Se van a describir en profundidad los resultados de la primera medición, para después hacer una comparativa general que evalúe la evolución del error relativo en las diversas condiciones de utilización.

Prueba de medición 1

En este caso, se tiene conectado únicamente el sistema de automatización del LabDER. Dado que los equipos que forman parte del mismo tienen características muy diversas, la carga no es enteramente resistiva, por lo que el factor de potencia se aleja de la unidad (0,74). Debido a que la calibración se ha realizado para cargas resistivas, se pretende observar cómo afecta este hecho a la precisión de las medidas realizadas.

Los gráficos que se muestran a continuación siguen el siguiente esquema de representación:

- Serie naranja: representa la medida tomada como referencia en el analizador Siemens.
- Serie azul: muestra las mediciones instantáneas realizadas por el analizador Arduino.
- Eje X: representa el número de mediciones realizadas.
- Eje Y: aparece el valor medido.

Tensiones de fase

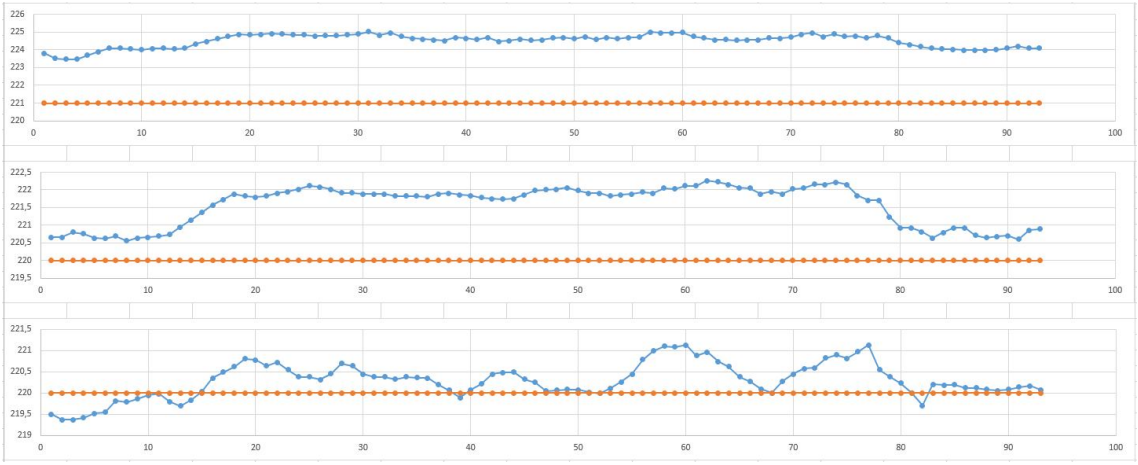


Figura 5.6: Comparativa entre las mediciones de las tensiones de fase entre el equipo Arduino y el comercial

Como se puede observar en la figura, existe un error en la calibración de la tensión en las fases 1 y 2. Los valores de error relativo medios son, para cada una de las fases, 1,58%, 0,71% y 0,18%.

Tensiones de línea

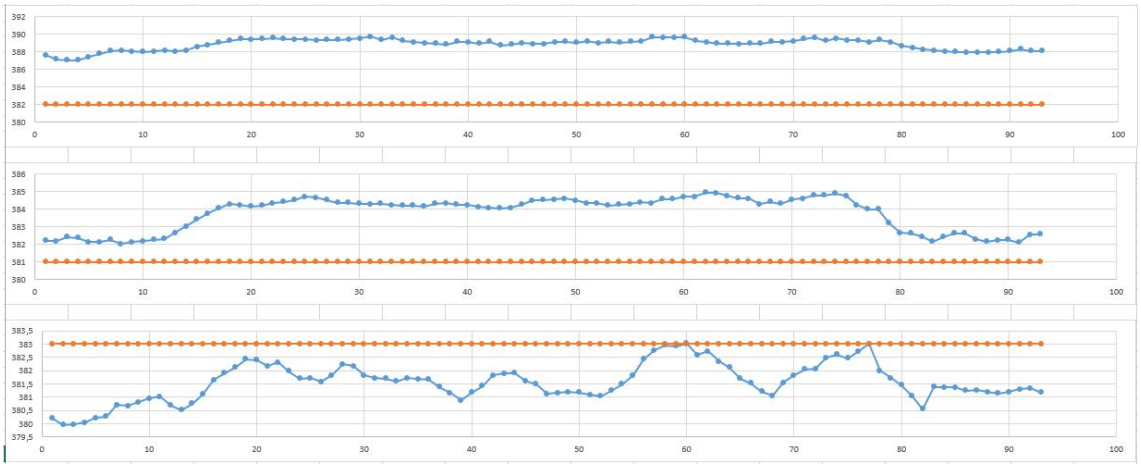


Figura 5.7: Comparativa entre las mediciones de las tensiones de línea entre el equipo Arduino y el comercial

Al obtenerse los valores de tensión de línea mediante un cálculo a partir de las tensiones de fase, la forma de la señal medida es idéntica a la anterior. Al existir errores de calibración en las tensiones de fase, éstos se trasladan también a las tensiones de línea, resultando un error relativo medio del 1,79%, 0,72% y 0,37%; valores muy parecidos a los obtenidos en el apartado anterior.

Corrientes

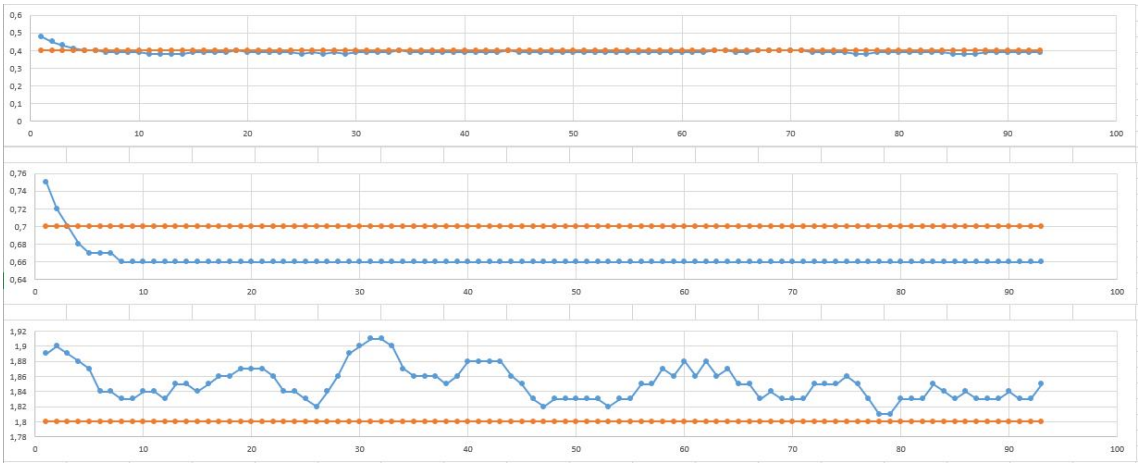


Figura 5.8: Comparativa entre las mediciones de las corrientes entre el equipo Arduino y el comercial

En esta figura se observa la calibración casi exacta de la corriente de la línea 1, mientras que existe un ligero error en las otras dos corrientes. Además, el hecho de que el valor de la corriente sea tan bajo produce que el error relativo aumente. En concreto, el valor del error relativo medio es de 2,85%, 5,56% y 2,81%, resultados un tanto elevados, que se reducirán al aumentar la corriente medida, como se comprobará posteriormente.

Factor de potencia

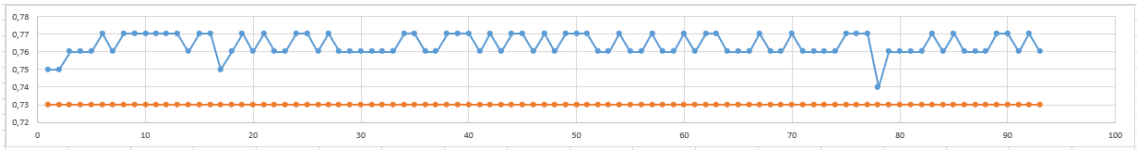


Figura 5.9: Comparativa entre las mediciones del factor de potencia entre el equipo Arduino y el comercial

Debido a que la calibración del factor de potencia se realizó en condiciones de carga puramente resistivas, y teniendo en cuenta que los coeficientes de calibración varían con las condiciones de operación, se explica el elevado valor de error medio que éste presenta (4,63%). En las siguientes pruebas, debido a la conexión de cargas de gran componente resistivo, el factor de potencia aumentará y con ello la precisión de la medición.

Potencia aparente total

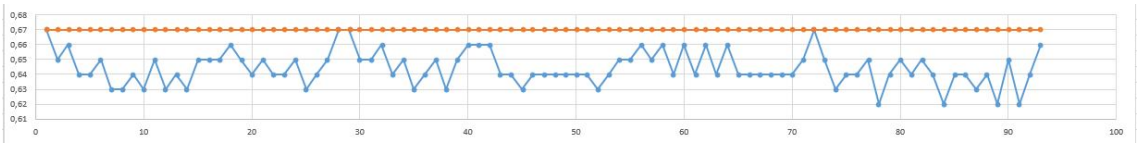


Figura 5.10: Comparativa entre las mediciones de potencia aparente entre el equipo Arduino y el comercial

La librería EmonLib realiza el cálculo de la potencia aparente mediante el producto de voltaje y tensión, por lo que los errores en la calibración de las mismas se trasladan también a este apartado. En este caso, el error relativo medio resulta ser del 3,83%.

Potencia activa total

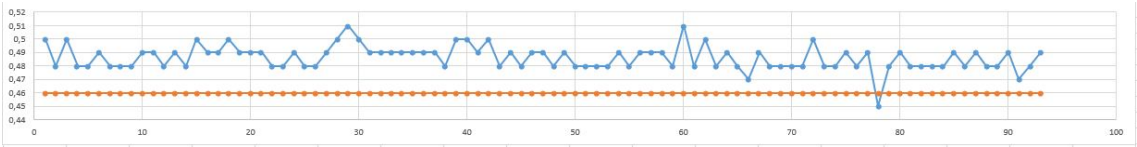


Figura 5.11: Comparativa entre las mediciones de la potencia activa entre el equipo Arduino y el comercial

Este es el valor que más errores acumula, pues es resultado del producto entre la potencia aparente y el factor de potencia. Por este motivo, el error relativo medio resulta ser del 5,77%.

Evolución del error relativo

En este apartado, se describen y explican los valores de error relativo a medida que se han ido realizando las diversas pruebas.

En este caso, el eje X de los gráficos representa el número de prueba de medición realizada (consultar la *Tabla 5.1: Configuraciones de las pruebas de medición*), mientras que en el eje Y aparece el valor de error relativo medio, expresado en porcentaje.

Tensiones de fase

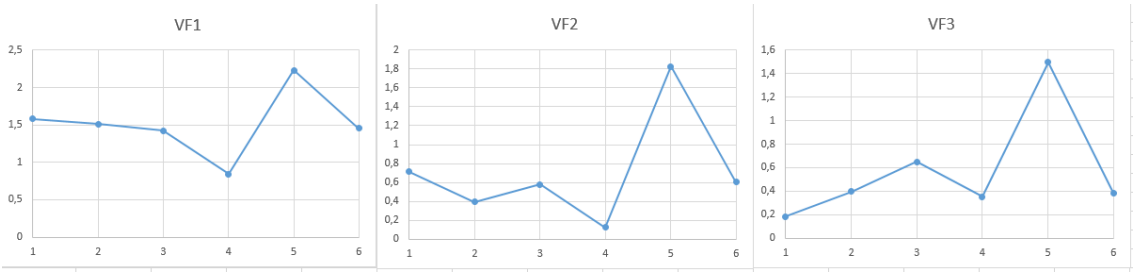


Figura 5.12: Evolución del valor medio del error relativo en las tensiones de fase

Como ya se ha podido observar al analizar la prueba 1, la calibración de la tensión de la fase 1 es menos exacta que las otras dos, resultando un error medio del 1,51%. En cambio, las otras dos tensiones mantienen un valor parecido (0,71% y 0,58%), lo cual representa de mejor manera el grado de precisión alcanzado.

Tensiones de línea

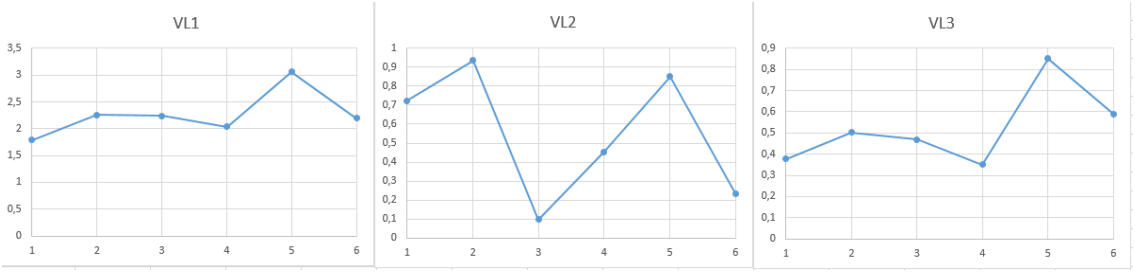


Figura 5.13: Evolución del valor medio del error relativo en las tensiones de línea

Las tensiones de línea, como ya se ha hecho notar anteriormente, son valores calculados directamente a partir de las tensiones de fase, por lo que su error relativo es también parecido: 2,26%, 0,55% y 0,52%.

Corrientes

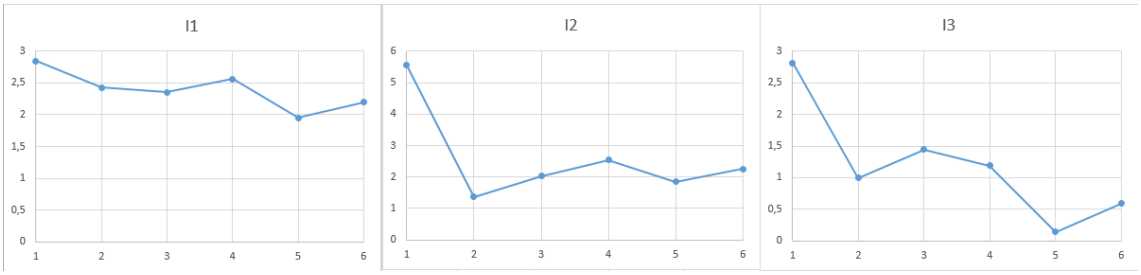


Figura 5.14: Evolución del valor medio del error relativo en las corrientes

En este caso, se puede apreciar que la medición de la corriente 3 posee un mejor grado de calibración (1,19%) que las otras dos (2,39% y 2,60%). No obstante, el comportamiento del error relativo durante las pruebas de medición es equiparable a las tres corrientes: el valor medio del error se reduce a medida que aumenta la corriente. Esto es debido a diversos factores:

- A la propia definición de error relativo, pues a medida que el valor de la corriente se aproxima a 0, éste aumenta, al estar el valor real en el denominador
- A la reducción del rendimiento del transformador cuando opera a un factor de carga muy bajo (21). Ya que el transformador de corriente equipado tiene un rango de utilización de 0 a 30A, trabajar con valores tan reducidos de corriente repercutirá negativamente en el rendimiento de este elemento.
- Algo parecido sucede con la lectura de la señal analógica por parte de la placa Arduino. El pin de entrada analógica se ha configurado también para leer valores en un rango de entre 0 a 30A, por lo tras la conversión a una señal digital, la resolución de medida es de 30000/1024 = 29mA. Para solucionar este problema, se podría utilizar un sistema parecido al de los multímetros, los cuales permiten seleccionar el rango de corrientes que se van a medir, adaptando su resolución según este criterio.

Factor de potencia

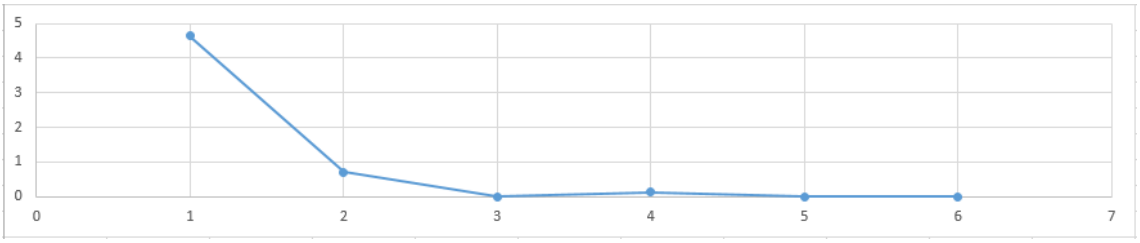


Figura 5.15: Evolución del valor medio del error relativo en el factor de potencia

En la figura superior, se observa cómo el error relativo del factor de potencia cae a medida que aumenta la carga resistiva de la instalación. Esto es debido a las condiciones en las que se realizó la calibración del analizador (con cargas puramente resistivas).

Aun así, el valor medio del error (0,91%) resulta aceptable para el rango de factores de potencia al que suele trabajar el laboratorio, comprendido entre 0,80 y 1. No obstante, se podría ajustar mejor al utilizar coeficientes de calibración diferentes dependiendo del rango de factor de potencia a medir.

Potencia aparente total

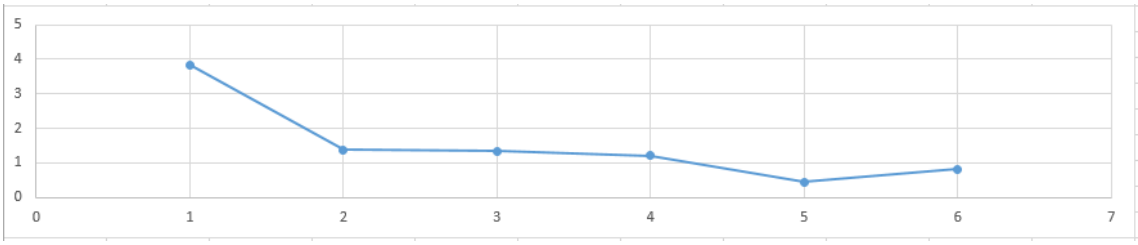


Figura 5.16: Evolución del valor medio del error relativo en la potencia aparente total

La potencia aparente, al ser proporcional a la corriente que circula por las líneas, tiene un comportamiento muy parecido a ésta, reduciéndose su error medio hasta un valor del 0,44% cuando circulan corrientes elevadas (de 7 a 22A).

No obstante, su precisión decrece a medida que decrece la carga, resultando en un valor medio del error relativo del 1,5%.

Potencia activa total

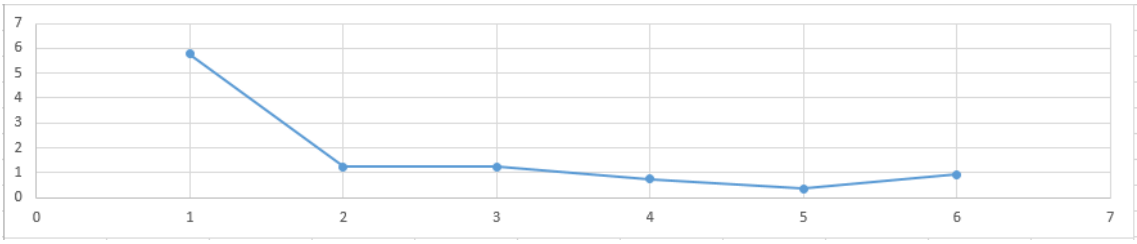


Figura 5.17: Evolución del valor medio del error relativo en la potencia activa total

La forma de esta curva es prácticamente igual a la de la potencia aparente total, debido al elevado factor de potencia utilizado en la mayoría de las pruebas de medición. El error relativo medio se mantiene en un 1,71%, valor también similar al de la potencia aparente.

Comparación con el analizador Siemens

Consultando en el manual del equipo Siemens PAC3200 (22), se pueden conocer los valores de precisión del equipo, clasificados según la magnitud a medir. El error relativo medio de cada una de ellas es el siguiente:

- Tensión: 0,3%.
- Corriente: 0,2%.
- Factor de potencia y potencias: 0,5%.

Para poder comparar esta información con el analizador, se han calculado también estos valores para el dispositivo Arduino, dando como resultado:

- Tensión: 1,11%.
- Corriente: 2,15%.
- Factor de potencia y potencias: 1,37%.

Como se puede observar, el dispositivo construido no dispone de tanta precisión como el analizador comercial. Aun así, se debe destacar que existe margen de mejora, pues se podría

llevar a cabo proceso de calibración más exhaustivo. Es decir, realizar pruebas de calibración para cada una de las tensiones, corrientes y factor de potencia en diferentes rangos de utilización, asignando a cada uno de esos rangos un coeficiente de calibración distinto.

Por tanto, se concluye que la precisión del modelo comercial es superior a la del analizador construido, pero que los valores de error medio se podrían reducir realizando un adecuado proceso de calibración.

Presupuesto

A continuación, en la *Tabla 5.2: Presupuesto del analizador trifásico* se detallan todos los componentes que integran el analizador, junto con su precio y el coste total del dispositivo.

En el *Anexo 9 - Referencia de precios* se listan las referencias de precios utilizados para este presupuesto.

Componente	**Precio p.u. (€)**	**Cantidad**	**Precio total (€)**
Transformador 230/12V	5,2	3	15,6
Pack de resistencias	9,52	1	9,52
Pack de condensadores	3,51	1	3,51
40 cables macho-macho	3,07	1	3,07
40 cables macho-hembra	3,07	1	3,07
Arduino MEGA (compatible)	19,95	1	19,95
Arduino Ethernet Shield	7,17	1	7,17
Arduino LCD Keypad Shield	5,17	1	5,17
Cable Ethernet	2,99	1	2,99
Placa de baquelita	2,58	1	2,58
Pinza amperimétrica	5,05	4	20,20
Poste hembra	1,07	2	2,14
Caja montaje	8,6	1	8,6
Fuente de alimentación	3,58	1	3,58
TOTAL			**107,15€**

Tabla 5.2: Presupuesto del analizador trifásico

A priori, el precio del analizador Arduino es bastante más reducido (107,15€) que el del modelo comercial (670,48€). Más adelante, en el capítulo de conclusiones, se valorará este dato, junto con las funcionalidades que incluyen cada uno de los modelos, para elaborar una conclusión sobre si se puede sustituir el dispositivo comercial por uno Arduino sin pérdida de funcionalidad.

Terminal HMI

El resultado de la segunda parte de este trabajo de fin de grado es una interfaz HMI para la monitorización de los valores del laboratorio mediante una pantalla táctil OMRON.

Una vez conectados los PLCs y los sensores del laboratorio, el terminal es capaz de solicitar información de los registros del PLC para su visualización en pantalla. Para una descripción detallada de los registros utilizados, se puede revisar el *Anexo 8 - Mapa de direcciones del PLC*.

Principalmente, los registros que se visualizan son los del PLC CJ2M, que es el que centraliza la información proveniente de la mayoría de sensores y medidores del laboratorio. Un esquema completo de los dispositivos conectados se puede consultar en el apartado de la introducción *Regulación de la microrred mediante un sistema SCADA*.

La visualización de la información del laboratorio se lleva a cabo a través de varias pantallas. Al iniciar el terminal aparece una pantalla, a modo de resumen, con un esquema general del laboratorio. En cada uno de los sistemas, se muestra el valor de potencia entregada o consumida, y la energía total acumulada.

Para navegar entre el resto de información disponible, se dispone de un menú de navegación en la parte superior de la pantalla, para permitir la visualización de los datos de cada sección. Cada uno de los valores se sitúa al lado del elemento a describir, junto con su unidad correspondiente, para su correcta comprensión. Para una descripción detallada de las pantallas creadas, se puede consultar el *Anexo 3 - Descripción de cada pantalla del terminal.*

Una vez se ha conectado y configurado todo correctamente, se observa un comportamiento muy estable del terminal, sin presentar ningún problema de funcionamiento. El equipo muestra las variables medidas en tiempo real, actualizándose constantemente, y la respuesta de la pantalla táctil es inmediata.

Programación del PLC

Este último apartado pretende describir las funciones programadas en el PLC CP1L, las cuales han posibilitado la integración del analizador Arduino a la red local para la posterior visualización de sus valores en el terminal HMI.

El analizador Arduino se conecta a la red mediante un cable Ethernet, para después comunicarse con el resto de equipos utilizando el protocolo Modbus. Para ello, se incorporan las instrucciones necesarias al sketch de la placa para asignar cada valor a un registro Modbus.

Después, el PLC se conecta al analizador Arduino y, mediante el programa en lenguaje Ladder instalado, cumple las funciones de conexión, lectura de valores del registro y almacenamiento de esos valores en su memoria interna. Esta memoria interna, al estar organizada en registros, permite guardar cada valor en una dirección determinada por el usuario.

Por último, la conexión del terminal HMI se realiza también mediante Modbus, pero en este caso es el propio terminal el que se conecta al PLC CP1L utilizando un protocolo propio de OMRON, para la lectura de los valores almacenados en los registros del PLC. Estos valores son mostrados por pantalla con una presentación y estructura adecuada, para su mejor visualización.

Capítulo 6 – Conclusiones

Esta parte final pretende elaborar una valoración general de todos los aspectos implicados en este Trabajo de Fin de Grado. Las conclusiones no solamente se centran en realizar un repaso por las características técnicas de los dispositivos desarrollados, sino que también se realiza una valoración económica de los mismos. Además, se incluye un apartado que describe los contenidos que el alumno ha interiorizado durante el periodo de realización de este proyecto. Por último, se realiza un repaso general, describiendo las conclusiones finales del trabajo.

Conclusiones técnicas

En este apartado, se van a analizar los resultados obtenidos en el capítulo anterior, para valorar si se han cumplido los objetivos propuestos para este proyecto. Como en toda la estructura del trabajo, se van a dividir estas conclusiones en tres partes:

Analizador Arduino

El objetivo de la realización de este analizador era demostrar que se puede obtener un analizador plenamente funcional con un presupuesto reducido, lo cual contrasta con el elevado precio al que se pueden encontrar los analizadores comerciales.

Para ello, se va a realizar una comparativa entre el dispositivo Arduino que se ha construido y un equipo comercial. Se van a contrastar las funciones que tiene cada uno, para posteriormente poder decidir en qué casos es una buena opción utilizar un analizador Arduino en lugar de uno comercial.

El analizador con el que se va a realizar la comparativa es un Siemens SENTRON PAC3200 (es uno de los dispositivos presentes en el laboratorio), para valorar si éste podría sustituirse por el analizador Arduino, valorando su reducido coste pero también las funcionalidades de las que dispone o carece.

En la *Tabla 6.1: Tabla comparativa entre el analizador trifásico Arduino y un modelo comercial,* se hace un resumen general de las características que comparten y distinguen a cada uno de estos dispositivos.

	Arduino 3PH	**SENTRON PAC3200**
Precio	107,15€	670,48€
Medición		
Voltaje de línea	No	Sí
Voltaje de fase	Sí	Sí
Corriente de línea y fase	Sí	Sí
Potencia activa, reactiva y aparente	Sí	Sí
Factor de potencia	Sí	Sí

Frecuencia	No	Sí
Histórico de energía acumulada	Sí	Sí
Distorsión armónica	No	Sí
Comunicaciones		
Ethernet	Sí	Sí
Funciones adicionales		
Menú de configuración	No	Sí
Pantalla de visualización de datos, resolución	Sí, 16x2 caracteres	Sí, 130x100 píxeles
Función de alarma	No	Sí

Tabla 6.1: Tabla comparativa entre el analizador trifásico Arduino y un modelo comercial

A simple vista, se pueden observar las funciones de las que el analizador Arduino carece en relación con el modelo comercial, tanto en medición como en funciones adicionales.

Se va a realizar una descripción de las diferencias encontradas entre ambos analizadores, con el objetivo de describir con detalle cada una de estas carencias, para determinar si son decisivas a la hora de elegir entre un analizador y otro. El resto de funciones se consideran cubiertas de la misma manera tanto por el Arduino como por el modelo comercial.

Antes de pasar a la descripción de las mismas, y para hacer hincapié en la naturaleza modular y ampliable de la plataforma Arduino, se ha preparado un anexo en el que se describen las funciones que el analizador construido sería capaz de tener si se le añadiesen el software y los componentes adecuados. Este apartado se denomina *Anexo 6 - Funciones adicionales del analizador Arduino*, el cual se tiene en cuenta en la valoración de las siguientes características:

- La medición directa de la tensión de línea: a pesar de que el dispositivo Arduino no realiza esta medición de forma empírica, el valor calculado a partir de las tensiones de fase es muy aproximado al valor real (error medio relativo del 0,52%), por lo que se puede obviar esta característica en determinadas situaciones. Además, se podría añadir esta característica mediante la instalación de otros tres transformadores, como se describe en el apartado *Medición de voltaje de línea.*
- La medición de la frecuencia es una característica de la que carece el dispositivo Arduino, pero en el anexo *Medición de la frecuencia* se demuestra que, con una actualización del software de la placa, esta funcionalidad podría ser integrada de manera sencilla.
- La medición de la distorsión armónica no está incluida tampoco dentro de las capacidades del analizador Arduino, pero se ha hallado un proyecto realizado por un usuario de la comunidad (23) que es capaz de realizar esta función. No obstante, se descarta la adición de esta funcionalidad al analizador, pues la correcta medición de los armónicos depende estrechamente de la velocidad de ejecución del sketch. Esto es debido a que, sin una demostración empírica, no se puede determinar con exactitud si al integrar esa funcionalidad en el software del analizador, se ejecutaría a la suficiente velocidad como para realizar una correcta medición.

- No existe menú de configuración en el analizador construido, principalmente debido a la limitación que supone el pequeño tamaño de la pantalla a la hora de mostrar una cantidad elevada de información. Por ello, se ha investigado si sería posible incorporar una pantalla de mayor calidad y tamaño, resultando en el apartado *Pantalla TFT de 3.5"*.
- La función de alarma, útil para casos en los que exista un dispositivo o una parte de la red funcionando en un rango de tensión o corriente inusual, es otra característica de la que no dispone el analizador construido, pero que se puede agregar mediante una actualización de software y un pequeño componente denominado *buzzer*. Se detalla este punto en el apartado .

- Alarma.

Por la investigación llevada a cabo en los puntos anteriores, se puede concluir que el analizador Arduino sería capaz de llevar a cabo todas las funciones adicionales descritas, exceptuando la característica de medición de la distorsión armónica.

Aun así, en general se puede observar que las capacidades del analizador Arduino están incluso más allá que las que ofrece el modelo comercial. La flexibilidad de la plataforma Arduino, con la gran cantidad de componentes que pueden ser añadidos, junto con la capacidad de programación de la placa, permite una personalización y una adaptación perfecta para cualquier tipo de necesidades.

No obstante, el modelo comercial también tiene ventajas. Esencialmente, el principal punto a favor es su facilidad de instalación, operación y mantenimiento: simplemente se compra el dispositivo, se conecta a la red eléctrica y comienza a medir. Además, cualquier problema que sufra el equipo puede ser revisado por el servicio de mantenimiento de la marca, por lo que se asegura un correcto funcionamiento del equipo durante la vida útil del mismo.

En cambio, el analizador Arduino, al estar diseñado y construido a partir de sus componentes elementales, hace más difícil hallar los problemas que surjan durante su operación. Además, el proceso de conexión y programación de la placa es una tarea que requiere un tiempo y unos conocimientos determinados, por lo que en entornos de aplicación inmediata, o en casos dirigidos a usuarios finales sin conocimientos técnicos, no sería recomendable la instalación de un dispositivo de estas características.

Por tanto, se puede observar que el analizador Arduino es superior en el apartado de funcionalidades, no obstante, se debe tener en cuenta la mayor complejidad de este sistema y el tiempo que se debería invertir en su construcción y corrección de errores, problemas que se encuentran en mucha menor medida en un modelo comercial.

No obstante, la decisión final depende del escenario de utilización: se deberá valorar si se prefiere un analizador a un precio muy razonable con una gran cantidad de funciones, o un dispositivo que operará desde el momento de su adquisición, y continuará con su correcto funcionamiento hasta el final de su vida útil.

Terminal táctil

En este apartado, se puede concluir que el equipo OMRON NB 7W01B cumple con los objetivos que se habían determinado para su correcto funcionamiento.

El terminal HMI es capaz de acceder a los dos PLCs presentes en la red local de manera simultánea, recogiendo valores de sus registros de memoria, una vez conocida la dirección donde se alojan esos registros. Además, es compatible con todos los formatos numéricos utilizados en esos registros, permitiendo conversiones proporcionales para una mejor visualización de los valores almacenados.

El tiempo de puesta en marcha del terminal es muy reducido: una vez conectado, aparece la pantalla del esquema general del LabDER, e inmediatamente se comienzan a mostrar los valores medidos en tiempo real. La respuesta de la pantalla táctil es impecable, además de que la organización de la información en diversas pantallas permite una mejor monitorización de los valores medidos, al estar subdividida en diferentes apartados que se pueden consultar individualmente.

Por tanto, se puede concluir que el dispositivo construido es apto para el reemplazo del terminal HMI que el laboratorio posee, pues no solamente sustituye su función de visualización de los datos relativos a la planta de gasificación, sino que además es capaz de mostrar la información captada por los sensores y medidores en todos los sistemas del laboratorio.

Programación del PLC

En este último apartado, se pretende destacar la facilidad en la programación del dispositivo PLC, junto con la correcta integración y capacidad de comunicación de este equipo con el resto de la red. También se describe la comunicación mediante Modbus del analizador Arduino con el resto de la microrred.

Dado que el lenguaje Ladder es un lenguaje de programación gráfico, basado en diagramas de contactos, el proceso de programación del comportamiento del PLC resulta una tarea muy sencilla. De manera visual, se puede comprender el esquema general de funcionamiento del programa, de una forma mucho más intuitiva que mediante instrucciones de código.

Además, a través de la utilización de bloques de función, se aligera el proceso de programación del PLC, al no ser necesario crear instrucciones para la conexión y la lectura de registros mediante Modbus. Todo este proceso se realiza mediante el software CX-Programmer, que facilita enormemente tareas como la creación del programa o la monitorización de los registros de la memoria del PLC.

La posterior integración del analizador Arduino a la red local resultó también satisfactoria. Se utiliza el protocolo Modbus sobre TCP/IP para la conexión entre el analizador y el PLC el cual, tras guardar los registros del analizador en su memoria interna, transmite la información a un terminal táctil HMI mediante un protocolo propio de OMRON. Esto demuestra que el PLC es capaz no sólo de mantener una comunicación simultánea con varios equipos diferentes dentro de la red local, sino que lo hace a través de protocolos distintos para cada dispositivo, lo cual permite agregar una gran variedad de equipos a la red local.

Conclusiones económicas

Una vez se han revisado las características técnicas de ambos analizadores, tanto del Arduino como del modelo comercia, es momento de realizar una valoración de su relación funcionalidades/precio, para poder realizar una comparación con el coste de un dispositivo Arduino.

Como se ha visto en apartados anteriores, el coste total del analizador Arduino se mantiene alrededor de los 107,15€. Dado que el precio del modelo comercial, aunque depende del vendedor, está en torno a los 670,48€, se ha de valorar en profundidad si se requieren las características propias de este modelo. Por el contrario, la elección resulta clara: el analizador Arduino es un dispositivo mucho más rentable, capaz de realizar las funciones de medición de tensión, corriente y potencia que se requieran a un precio muy bajo.

No obstante, se ha querido demostrar que el analizador Arduino sigue siendo un dispositivo mucho más barato que el modelo comercial, aun añadiendo todos los componentes necesarios para la adición de las funcionalidades adicionales descritas anteriormente. Para ello, se ha elaborado un presupuesto que incluye todos estos componentes.

Componente	**Precio p.u. (€)**	**Cantidad**	**Precio total (€)**
Transformador 230/12V	5,20	3	15,60
Transformador 400/12v	17,25	3	51,75
Pack de resistencias	9,52	1	9,52
Pack de condensadores	3,51	1	3,51
40 cables macho-macho	3,07	1	3,07
40 cables macho-hembra	3,07	1	3,07
Arduino MEGA (compatible)	19,95	1	19,95
Arduino Ethernet Shield	7,17	1	7,17
Pantalla TFT 3,5”	9,66	1	9,66
Buzzer	1,00	1	1,00
Cable Ethernet	2,99	1	2,99
Placa de baquelita	2,58	1	2,58
Pinza amperimétrica	5,05	4	20,20
Poste hembra	1,07	2	2,14
Caja montaje	8,60	1	8,60
Fuente de alimentación	3,58	1	3,58
TOTAL			**164,39€**

Tabla 6.2: Presupuesto de un analizador Arduino con funciones añadidas

Esto demuestra que, aun añadiéndole al analizador Arduino todas las funcionalidades comentadas anteriormente, el coste del equipo seguiría siendo más bajo (164,39€) que el del modelo Siemens (670,48€). Así, el dispositivo Arduino no solamente superaría al analizador comercial en su precio, sino que estaría prácticamente equiparado en funciones, por lo que sería la elección a realizar en la mayoría de ámbitos de aplicación.

Conocimientos adquiridos

A pesar de que este trabajo está relacionado con la temática del grado, y que muchas de las capacidades y conocimientos necesarios para su realización formaban parte del plan formativo de la titulación, se requería una mayor profundización en algunas áreas para la correcta consecución del trabajo. Este hecho está motivado por el tipo de documento realizado, catalogado como proyecto de investigación.

El aprendizaje práctico adquirido durante el desarrollo del mismo ha ayudado a interiorizar nuevos conceptos y capacidades que no se poseían antes de su realización. Así, se ha complementado la formación recibida durante el grado, especialmente en las áreas de Electrónica y Sistemas Automáticos.

La primera parte, que se centra en el analizador, aporta conocimientos tanto de electrónica como de la plataforma Arduino. Esta parte subraya la necesidad de la correcta comprensión y diseño de los circuitos, para que no se cometan errores en el posterior montaje. Se adquiere fluidez en el uso del multímetro, especialmente para medir las resistencias y para comprobar la correcta conexión de las diferentes partes del circuito. Además, se requiere el aprendizaje de la técnica de la soldadura con estaño, con el objetivo de poder soldar los componentes a la placa de baquelita.

Otro ámbito en el que se adquiere una gran cantidad de conocimientos es en el de la plataforma Arduino, descubriendo el enorme número de funcionalidades y aplicaciones que pueden ser desarrolladas para su placa. En este caso, es de agradecer haber recibido formación relativa a la programación en lenguaje C durante el grado. Además, la gran comunidad que apoya esta plataforma hace muy sencillo encontrar soluciones para los problemas que surgen, pues es una comunidad muy activa que comparte en Internet los resultados de sus aplicaciones.

La segunda y tercera parte, las cuales se centran en Sistemas Automáticos, son apartados principalmente prácticos. Dado que la formación adquirida durante el grado en este ámbito fue de carácter teórico, se han de interiorizar una gran cantidad de conceptos para la comprensión de su funcionamiento. Se requieren conocimientos acerca de la estructura de una red local y de los elementos que la forman, en especial acerca de las características y el funcionamiento de un PLC: desde cómo se comunican los PLCs con el resto de equipos dentro de la red, hasta cómo se programa un autómata, pasando por cómo funciona su memoria. Todos ellos conocimientos que se consideran de gran utilidad en el ámbito de la ingeniería práctica.

En resumen, considero el Trabajo de Fin de Grado como una experiencia que ha posibilitado el aprendizaje de una gran cantidad de conocimientos prácticos, además de la capacidad de organización, esfuerzo y dedicación necesaria para poder realizar un proyecto de este tamaño.

Conclusión final

Tras haber revisado en detalle todos los aspectos relativos a este Trabajo de Fin de Grado, se van a exponer sus conclusiones finales, repasando cada una de las partes realizadas y dando una valoración global del conjunto.

Para empezar, se ha demostrado la capacidad de un dispositivo Arduino de realizar prácticamente las mismas funciones que un analizador comercial. El diseño y construcción del mismo no resulta complicado, además de que la flexibilidad de la plataforma Arduino permite la adición de funcionalidades sin un incremento sustancial en el coste del equipo.

La precisión de la medición, tras realizarse una correcta calibración, es suficiente para la mayoría de ámbitos de utilización. Además, el apartado de conexiones está cubierto por el shield Ethernet que incorpora, pues a través de esta conexión física es capaz de establecer comunicaciones con otros equipos mediante el protocolo Modbus.

Sin embargo, se determina que no son dispositivos aptos para cualquier caso de utilización, pues existen ámbitos en los cuales se requiere una elevada facilidad de instalación y operación, dentro de los cuales un dispositivo comercial justificaría su elevado precio. Por el contrario, el dispositivo Arduino tiene un coste muy reducido, quedando demostrado que la construcción de un analizador Arduino resulta mucho más barata que la adquisición de su equivalente comercial.

En la segunda parte del trabajo, se ha realizado la programación de un terminal HMI, un dispositivo que, al conectarse al sistema de automatización, permite el acceso y visualización por pantalla de toda la información disponible en los PLC del laboratorio. Por tanto, se considera apto para la sustitución del terminal HMI que operaba previamente en el LabDER.

En la tercera parte, la integración del analizador Arduino en la microrred también ha resultado satisfactoria, desempeñando de forma correcta sus funciones de medición y posterior envío de información a la red local, tal y como lo hace un dispositivo comercial.

Como conclusión, se puede afirmar que la elección de la plataforma Arduino para la realización del analizador trifásico ha resultado ser un acierto, pues no sólo se ha demostrado que es superior en coste, sino que las funciones que es capaz de realizar son equiparables a las de un equipo comercial.

Capítulo 7 - Bibliografía

1. **Wikipedia.** Hybrid Renewable Energy Systems. [En línea] 2017. https://en.wikipedia.org/wiki/Hybrid_renewable_energy_system.

2. **Instituto de Ingeniería Energética.** LabDER. [En línea] 2017. http://iie.webs.upv.es/labder/.

3. **Arduino.cc.** Arduino FAQ. [En línea] 2017. https://www.arduino.cc/en/Main/FAQ#toc11.

4. **Open Energy Monitor.** Electricity Monitoring. [En línea] 2017. https://learn.openenergymonitor.org/electricity-monitoring/.

5. **OpenEnergyMonitor.** CT Sensor - Interfacing with an Arduino. [En línea] 2017. https://learn.openenergymonitor.org/electricity-monitoring/ct-sensors/interface-with-arduino?redirected=true.

6. **Amazon.** SODIAL Transformador de Corriente CA SCT 013-030 3.5mm Salida No invasiva. [En línea] 2017. https://www.amazon.es/SODIAL-Transformador-Corriente-013-030-invasiva/dp/B00H3CTFIQ/ref=sr_1_1?ie=UTF8&qid=1496083410&sr=8-1&keywords=SCT013.

7. **Arduino.cc.** AnalogRead. [En línea] 2017. https://www.arduino.cc/en/Reference/AnalogRead.

8. **Open Energy Monitor.** Measuring AC Voltage with an AC to AC power adapter. [En línea] 2017. https://learn.openenergymonitor.org/electricity-monitoring/voltage-sensing/measuring-voltage-with-an-acac-power-adapter?redirected=true.

9. **Arduino.cc.** Arduino Software. [En línea] 2017. https://www.arduino.cc/en/Main/Software.

10. **Driver Scape.** USB-SERIAL CH340. [En línea] 2016. http://www.driverscape.com/download/usb-serial-ch340.

11. **GitHub.** openenergymonitor/EmonLib: Electricity monitoring library. [En línea] 2014. https://github.com/openenergymonitor/EmonLib.

12. **Aprendiendo Arduino.** Shields para Arduino. [En línea] 2015. https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/.

13. **Arduino.cc.** LiquidCrystal. [En línea] 2017. https://www.arduino.cc/en/Reference/LiquidCrystal.

14. **Wikipedia.** Efecto Joule. [En línea] https://es.wikipedia.org/wiki/Efecto_Joule.

15. **Universidad de Vigo.** Curso multimedia de electromagnetismo. [En línea] 2011. http://quintans.webs.uvigo.es/recursos/Web_electromagnetismo/magnetismo_perdidasmagneticas.htm.

16. **OMRON.** Manual de operación de NB Designer. [En línea] 2016. https://downloads.omron.es/IAB/Products/Automation%20Systems/HMI/Compact%20HMI/NB7/V106/V106-E1-14+NB-series+OperManual.pdf.

17. **Programando paso a paso.** Espacios de memoria por tipo de variable. [En línea] 2010. https://eperdomo89.wordpress.com/2009/11/11/espacios-de-memoria-por-tipo-de-variable/.

18. **Ragnall.** Mudbus - GitHub. [En línea] 09 de 02 de 2012. https://github.com/luizcantoni/mudbus/blob/master/Mudbus/Mudbus.h.

19. **Simply Modbus.** Read Holding Registers (FC=03). [En línea] 2015. http://www.simplymodbus.ca/FC03.htm.

20. **Wikipedia.** Lenguaje Ladder. [En línea] 2017. https://es.wikipedia.org/wiki/Lenguaje_Ladder.

21. **Universidad de Carabobo.** Máquinas Eléctricas. [En línea] 2008. http://www.ing.uc.edu.ve/electrica/potencia/maqelec1/index.php?option=com_content&view=article&id=94&Itemid=88.

22. **Siemens.** Manual de referencia del analizador Siemens PAC3200. [En línea] 2011. https://www.downloads.siemens.com/download-center/Download.aspx?pos=download&fct=getasset&id1=BTLV_38467.

23. **duino4projects.** Power Quality Meter using Arduino. [En línea] 2014. http://duino4projects.com/power-quality-meter-using-arduino/.

24. **Avtanski, Alexander.** LCD Screenshots Generator. [En línea] 2014. http://avtanski.net/projects/lcd/.

25. **Siemens.** Rapid Online. [En línea] 2011. https://www.rapidonline.com/pdf/543861_v1.pdf.

26. **arduino.cc.** Arduino Mega 2560. [En línea] 2017. https://www.arduino.cc/en/main/arduinoBoardMega2560.

27. **openenergymonitor Forums.** Measuring frequency from the EmonLib. [En línea] 2014. https://openenergymonitor.org/forum-archive/node/3791.html.

28. **Foro Arduino.** Velocidad de ejecución de instrucciones. [En línea] 2012. http://forum.arduino.cc/index.php?topic=86962.0.

29. **GitHub.** Open Energy Monitor - EmonLib. [En línea] 2014. https://github.com/openenergymonitor/EmonLib.

30. **Omron.** Serie NB. [En línea] 2017. https://industrial.omron.es/es/products/nb.

Capítulo 8 – Anexos

Anexo 1 - Código fuente del analizador trifásico

Este apartado contiene el código completo del sketch Arduino que permite a la placa Arduino actuar como analizador trifásico.

```
#include <SPI.h>
#include <Ethernet.h>
#include <Mudbus.h>

#include <EmonLib.h>
#include <LiquidCrystal.h>

EnergyMonitor emon_ph1;
EnergyMonitor emon_ph2;
EnergyMonitor emon_ph3;
EnergyMonitor emon_neu;

double V1[5];
double V2[5];
double V3[5];
double Vtemp[5];
double V1_tot, V2_tot, V3_tot;

int seconds = 0;
short int minutes = 0;
short int hours = 0;
int days = 0;
int lasttime = 0;
int offset = 0;
int e_act = 0;
int e_rea = 0;

int screen      = 0;
int i;

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

int lcd_key     = 0;
int adc_key_in  = 0;

#define btnBACK     0
#define btnFORWARD  1
#define btnNONE     5

int read_LCD_buttons()
{
 adc_key_in = analogRead(0
 if (adc_key_in > 1000) return btnNONE;
 if (adc_key_in < 50)   return btnFORWARD;
 if (adc_key_in < 650)  return btnBACK;
}

Mudbus Mb;
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
IPAddress ip(192, 168, 250, 13);
```

```
void setup()
{
  for(i=0;i<5;i++){ V1[i] = 0; V2[i] = 0; V3[i] = 0; }

  delay(5000);

  Serial.begin(9600);

  emon_ph1.voltage(2, 518, 1.15);
  emon_ph2.voltage(3, 518, 1.15);
  emon_ph3.voltage(4, 525, 1.15);

  emon_ph1.current(8 , 29.6);
  emon_ph2.current(9 , 29.6);
  emon_ph3.current(10, 29.6);
  emon_neu.current(11, 29.6);

  Ethernet.begin(mac, ip);

  lcd.begin(16, 2);
}

void loop()
{
  emon_ph1.calcVI(20,100);
  emon_ph2.calcVI(20,100);
  emon_ph3.calcVI(20,100);
  emon_neu.calcVI(20,100);

double Vf_ph1, Vf_ph2, Vf_ph3;
double Vl_ph1, Vl_ph2, Vl_ph3;
double Il_ph1, Il_ph2, Il_ph3, Il_neu;
double If_ph1, If_ph2, If_ph3, If_neu;

V1_tot = 0; V2_tot = 0; V3_tot = 0;

for(i=0;i<5-1;i++){  Vtemp[i+1] = V1[i]; } for(i=0;i<5;i++){ V1[i] =
Vtemp[i]; V1_tot = V1_tot + V1[i]; }
for(i=0;i<5-1;i++){  Vtemp[i+1] = V2[i]; } for(i=0;i<5;i++){ V2[i] =
Vtemp[i]; V2_tot = V2_tot + V2[i]; }
for(i=0;i<5-1;i++){  Vtemp[i+1] = V3[i]; } for(i=0;i<5;i++){ V3[i] =
Vtemp[i]; V3_tot = V3_tot + V3[i]; }

  V1[0] = emon_ph1.Vrms;
  V2[0] = emon_ph2.Vrms;
  V3[0] = emon_ph3.Vrms;
  V1_tot = V1_tot + V1[0];
  V2_tot = V2_tot + V1[0];
  V3_tot = V3_tot + V1[0];

  Vf_ph1 = V1_tot/5;
  Vf_ph2 = V2_tot/5;
  Vf_ph3 = V3_tot/5;

  Vl_ph1 = sqrt(3)*Vf_ph1;
  Vl_ph2 = sqrt(3)*Vf_ph2;
  Vl_ph3 = sqrt(3)*Vf_ph3;

  Il_ph1 = emon_ph1.Irms;
  Il_ph2 = emon_ph2.Irms;
  Il_ph3 = emon_ph3.Irms;
  Il_neu = emon_neu.Irms;

  If_ph1 = Il_ph1;
  If_ph2 = Il_ph2;
  If_ph3 = Il_ph3;
  If_neu = Il_neu;
```

```
double P_ph1 = emon_ph1.realPower/1000;
double P_ph2 = emon_ph2.realPower/1000;
double P_ph3 = emon_ph3.realPower/1000;

double S_ph1 = emon_ph1.apparentPower/1000;
double S_ph2 = emon_ph2.apparentPower/1000;
double S_ph3 = emon_ph3.apparentPower/1000;

double Q_ph1 = sqrt(pow(S_ph1,2)-pow(P_ph1,2));
double Q_ph2 = sqrt(pow(S_ph2,2)-pow(P_ph2,2));
double Q_ph3 = sqrt(pow(S_ph3,2)-pow(P_ph3,2));

double pf_ph1 = emon_ph1.powerFactor;
double pf_ph2 = emon_ph2.powerFactor;
double pf_ph3 = emon_ph3.powerFactor;

if(lasttime!=0)
  offset = millis() - lasttime;
seconds = millis()/1000 - minutes*60;
if(seconds>=59)
  minutes++;
if(minutes>=59){
  minutes = 0;
  hours++;
}
if(hours>=23){
  hours = 0;
  days++;
}
lasttime = millis();
char time_char[12] = " ";
sprintf(time_char, "%2dd%02d:%02d:%02d", days, hours, minutes, seconds);

e_act = e_act + (P_ph1+P_ph2+P_ph3)*1000*offset/3600;
e_rea = e_rea + (Q_ph1+Q_ph2+Q_ph3)*1000*offset/3600;
char e_act_char[10]=" ";
sprintf(e_act_char, "%9d", e_act);
char e_rea_char[10]=" ";
sprintf(e_rea_char, "%9d", e_rea);

Mb.Run();
Mb.R[0]  = Vf_ph1*1000;
Mb.R[1]  = Vf_ph2*1000;
Mb.R[2]  = Vf_ph3*1000;
Mb.R[3]  = Vl_ph1*1000;
Mb.R[4]  = Vl_ph2*1000;
Mb.R[5]  = Vl_ph3*1000;
Mb.R[6]  = If_ph1*1000;
Mb.R[7]  = If_ph2*1000;
Mb.R[8]  = If_ph3*1000;
Mb.R[9]  = If_neu*1000;
Mb.R[10] = Il_ph1*1000;
Mb.R[11] = Il_ph2*1000;
Mb.R[12] = Il_ph3*1000;
Mb.R[13] = Il_neu*1000;
Mb.R[14] = P_ph1*1000;
Mb.R[15] = P_ph2*1000;
Mb.R[16] = P_ph3*1000;
Mb.R[17] = Q_ph1*1000;
Mb.R[18] = Q_ph2*1000;
Mb.R[19] = Q_ph3*1000;
Mb.R[20] = S_ph1*1000;
Mb.R[21] = S_ph2*1000;
Mb.R[22] = S_ph3*1000;
Mb.R[23] = pf_ph1*1000;
Mb.R[24] = pf_ph2*1000;
Mb.R[25] = pf_ph3*1000;
```

```
 Mb.R[26] = e_act*1000;
 Mb.R[27] = e_rea*1000;
 Mb.R[29] = days;
 Mb.R[30] = hours;
 Mb.R[31] = minutes;
 Mb.R[32] = seconds;

lcd_key = read_LCD_buttons();
int number_of_screens = 9;

switch (lcd_key)
{
  case btnBACK:
    {
    if(screen>0) screen--; else screen = number_of_screens;
    break;
    }
  case btnFORWARD:
    {
    if(screen<number_of_screens) screen++; else screen = 0;
    break;
    }
}
switch (screen)
{
   case 0:
     {
       lcd.setCursor(0,0);
       lcd.print("1 ");
       lcd.print(Vl_ph1,1);
       lcd.setCursor(7,0);
       lcd.print(" 2 ");
       lcd.print(Vl_ph2,1);
       lcd.setCursor(0,1);
       lcd.print("3 ");
       lcd.print(Vl_ph3,1);
       lcd.setCursor(7,1);
       lcd.print(" VL (V) ");
     }
   case 1:
     {
       lcd.setCursor(0,0);
       lcd.print("1 ");
       lcd.print(Vf_ph1,1);
       lcd.setCursor(7,0);
       lcd.print(" 2 ");
       lcd.print(Vf_ph2,1);
       lcd.setCursor(0,1);
       lcd.print("3 ");
       lcd.print(Vf_ph3,1);
       lcd.setCursor(7,1);
       lcd.print(" VF (V) ");
       break;
     }
   case 2:
     {
       lcd.setCursor(0,0);
       lcd.print("1 ");
       lcd.print(Il_ph1,3);
       lcd.setCursor(7,0);
       lcd.print(" 2 ");
       lcd.print(Il_ph2,3);
       lcd.setCursor(0,1);
       lcd.print("3 ");
       lcd.print(Il_ph3,3);
       lcd.setCursor(7,1);
       lcd.print(" N ");
```

```
        lcd.print(Il_neu,3);
        break;
      }
    case 3:
      {
        lcd.setCursor(0,0);
        lcd.print("1 ");
        lcd.print(If_ph1,3);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(If_ph2,3);
        lcd.setCursor(0,1);
        lcd.print("3 ");
        lcd.print(If_ph3,3);
        lcd.setCursor(7,1);
        lcd.print(" N ");
        lcd.print(If_neu,3);
        break;
      }
    case 4:
      {
        lcd.setCursor(0,0);
        lcd.print("1 ");
        lcd.print(P_ph1,3);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(P_ph2,3);
        lcd.setCursor(0,1);
        lcd.print("3 ");
        lcd.print(P_ph3,3);
        lcd.setCursor(7,1);
        lcd.print(" P (KW)  ");
        break;
      }
    case 5:
      {
        lcd.setCursor(0,0);
        lcd.print("1 ");
        lcd.print(Q_ph1,3);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(Q_ph2,3);
        lcd.setCursor(0,1);
        lcd.print("3 ");
        lcd.print(Q_ph3,3);
        lcd.setCursor(7,1);
        lcd.print(" Q (KVar)");
        break;
      }
    case 6:
      {
        lcd.setCursor(0,0);
        lcd.print("1 ");
        lcd.print(S_ph1,3);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(S_ph2,3);
        lcd.setCursor(0,1);
        lcd.print("3 ");
        lcd.print(S_ph3,3);
        lcd.setCursor(7,1);
        lcd.print(" S (KVA) ");
        break;
      }
    case 7:
      {
        lcd.setCursor(0,0);
```

```
        lcd.print("1 ");
        lcd.print(pf_ph1,3);
        lcd.setCursor(7,0);
        lcd.print(" 2 ");
        lcd.print(pf_ph2,3);
        lcd.print(" ");
        lcd.setCursor(0,1);
        lcd.print("3 ");
        lcd.print(pf_ph3,3);
        lcd.setCursor(7,1);
        lcd.print(" P.Factor");
        break;
      }
    case 8:
      {
        lcd.setCursor(0,0);
        lcd.print("  ");
        lcd.print(e_act_char);
        lcd.setCursor(11,0);
        lcd.print("   Wh");
        lcd.setCursor(0,1);
        lcd.print("Time:");
        lcd.setCursor(5,1);
        lcd.print(time_char);
        break;
      }
    case 9:
      {
        lcd.setCursor(0,0);
        lcd.print("  ");
        lcd.print(e_rea_char);
        lcd.setCursor(11,0);
        lcd.print(" varh");
        lcd.setCursor(0,1);
        lcd.print("Time:");
        lcd.setCursor(5,1);
        lcd.print(time_char);
        break;
      }
}
```

Anexo 2 - Diseño de las pantallas del analizador

En este anexo se van a visualizar todas las pantallas de la LCD Keypad Shield que dispone el analizador construido, entre las que se puede navegar pulsando los botones Left y Right incluidos en la pantalla. Para una mejor presentación, se ha decidido utilizar un software (24) para generar las imágenes de la pantalla LCD.

Voltaje de línea

Voltaje de fase

Corriente

Potencia activa

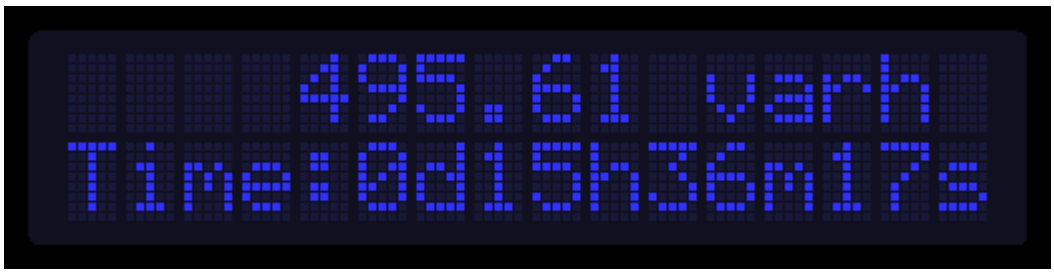
Potencia reactiva

Potencia aparente

Factor de potencia

Energía activa

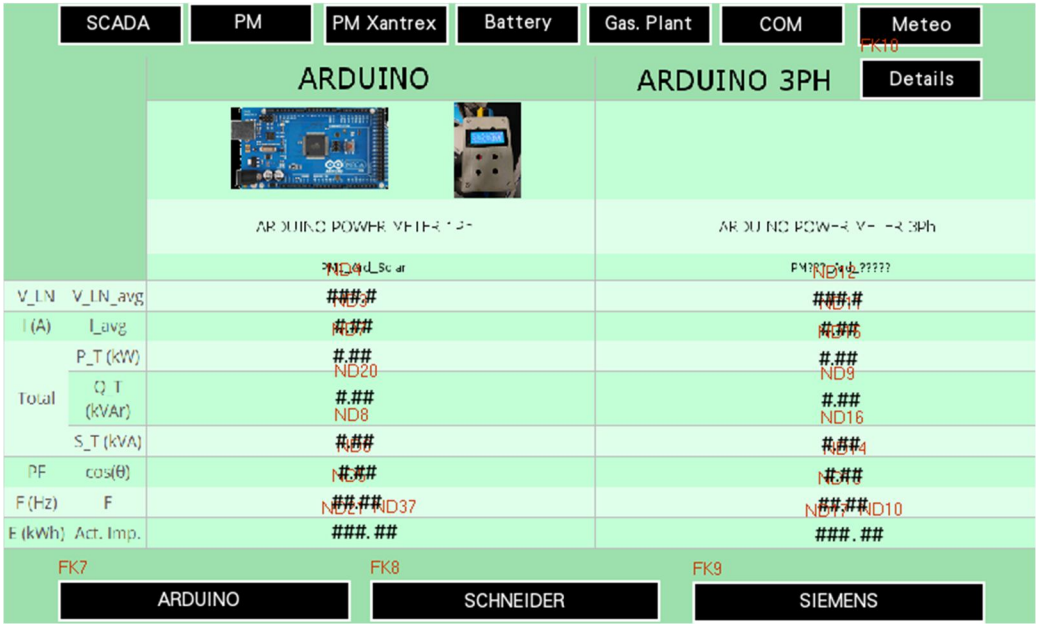
Energía reactiva



Anexo 3 - Descripción de cada pantalla del terminal

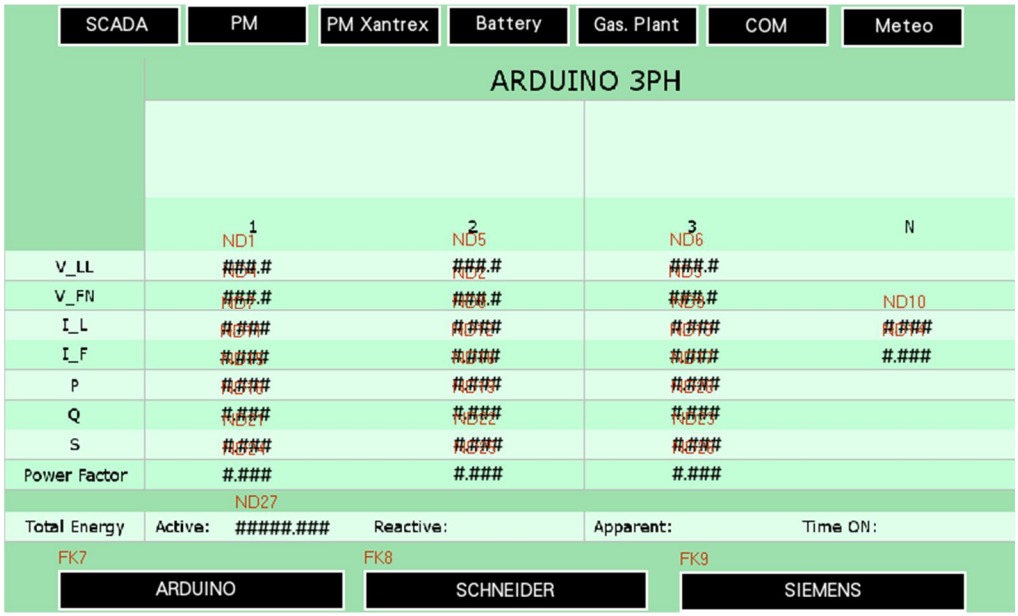
En este anexo, se van a enumerar y describir cada una de las pantallas diseñadas en el terminal táctil HMI, evaluando especialmente qué valores son los que se muestran en cada apartado.

PM Arduino

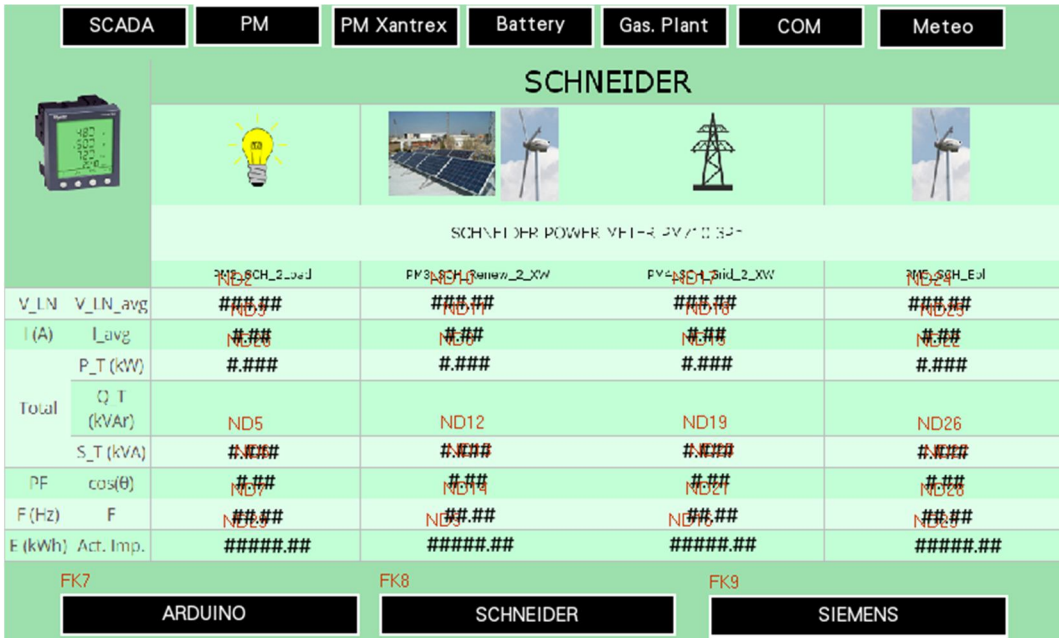


En esta pantalla, a la que se accede tras pulsar el botón PM, aparecen los valores relativos a los dos analizadores Arduino que están instalados en la microrred. El primer analizador mide los valores consumidos por las cargas, mientras que el segundo es el analizador que se ha desarrollado durante este trabajo.

Con los botones Schneider y Siemens se accede a los datos de los analizadores de estas marcas, y con el botón Details se accede a todos los valores que está registrando el analizador Arduino que se ha desarrollado como parte de este trabajo. Aparecen voltajes, intensidades, potencias y factor de potencia en cada una de las líneas, además del acumulado de energía activa, reactiva y aparente desde que se puso en marcha el analizador.



PM Schneider

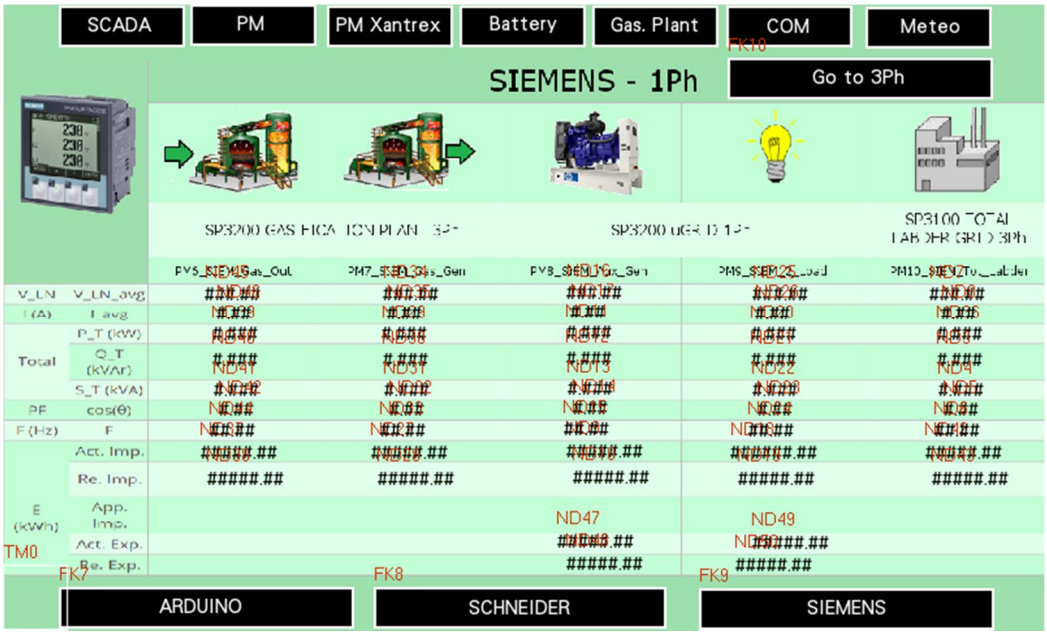


Existen 4 analizadores Schneider conectados en la microrred. Están conectados en los siguientes puntos de la instalación:

- La salida de las cargas
- La salida de ambas renovables hacia el Xantrex
- La conexión a la red eléctrica
- La salida del aerogenerador

En todos ellos se ha realizado un resumen de los datos medidos: voltaje medio de línea, corriente media de línea, potencia activa, aparente, factor de potencia y acumulado de energía.

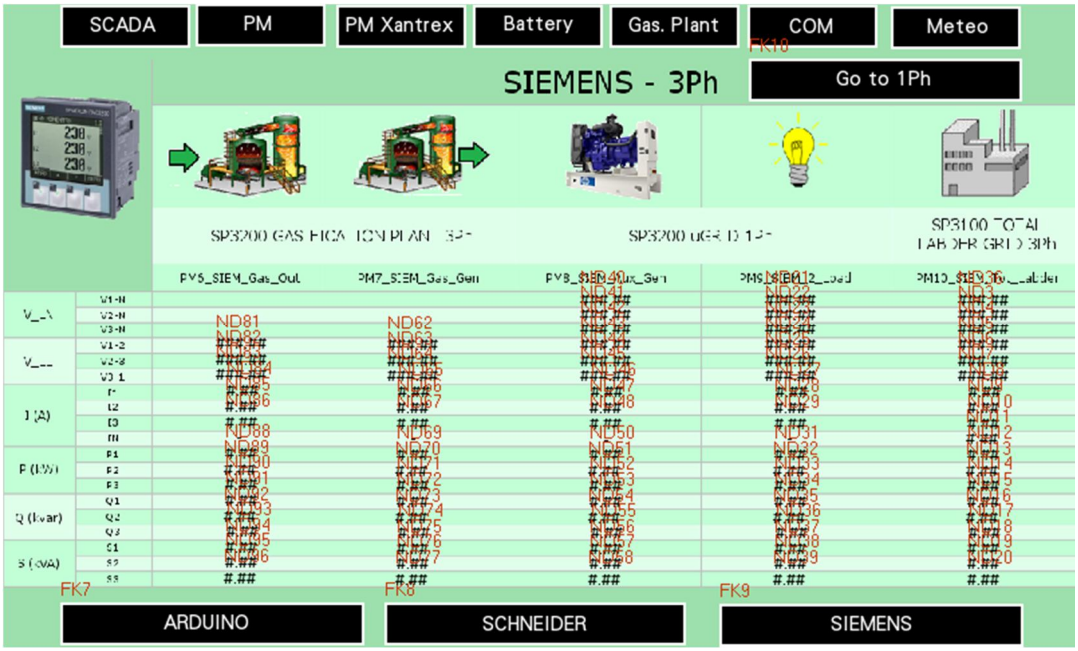
PM Siemens



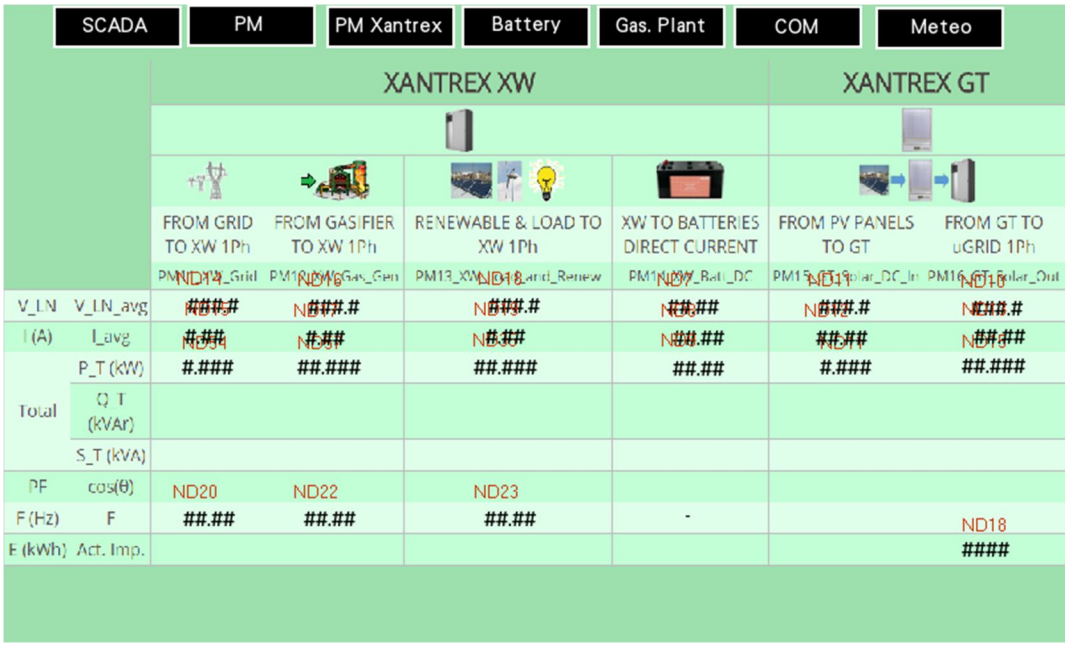
En la pantalla PM Siemens aparecen muchos de los valores medidos por los analizadores de esta marca, que están colocados:

- A la entrada de la planta de gasificación
- A la salida de la planta de gasificación
- A la salida del generador auxiliar
- A la salid de las cargas
- En la conexión general de la microrred a la red eléctrica

Pulsando Arduino o Schneider, se pueden consultar las pantallas de dichos analizadores, y con el botón Go to 3Ph se pueden ver con más detalle los datos que están midiendo estos equipos.



PM Xantrex



En esta pantalla aparecen los dos equipos de la marca Xantrex: el Xantrex XW y el Xantrex GT.

Xantrex XW es un equipo que principalmente es un inversor/cargador cuya función es cargar, conectar y desconectar las baterías dependiendo del estado del resto de la instalación. Además, lleva incorporados 4 analizadores, que miden datos de corriente y tensión en los siguientes puntos:

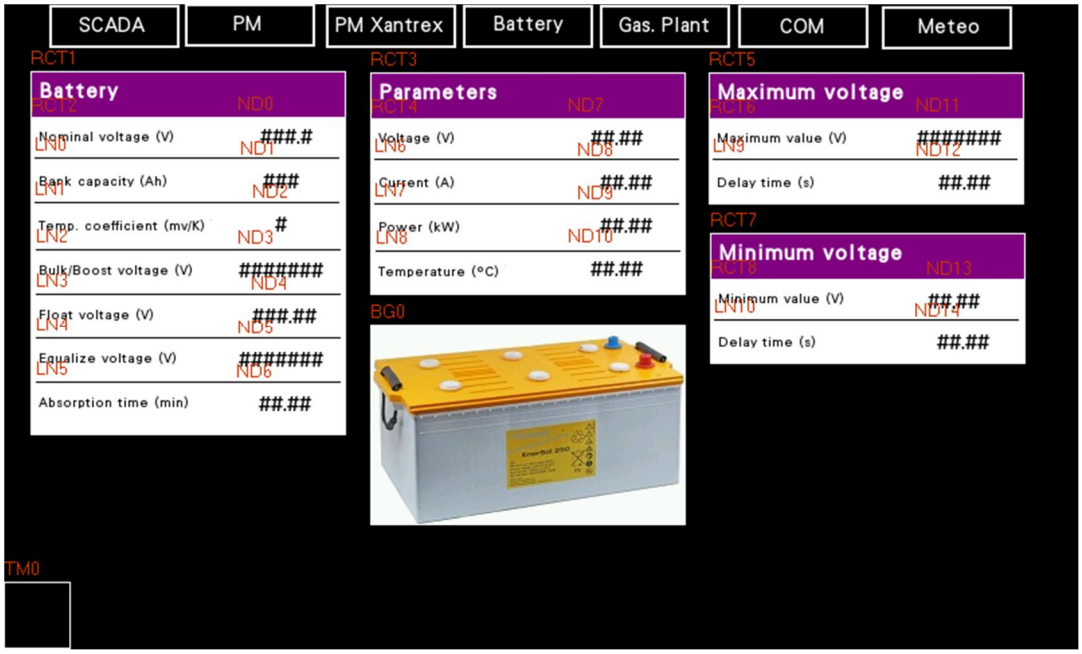
- PM11: En la conexión con la red eléctrica
- PM12: En la entrada proveniente de la planta de gasificación y el generador auxiliar
- PM13: A la salida de las renovables
- PM14: En la conexión con las baterías

Xantrex GT es también un inversor, cuya función es convertir la corriente continua de los paneles fotovoltaicos en corriente alterna para su incorporación a la microrred. Incorpora dos medidores:

- PM15: Colocado a la entrada del inversor
- PM16: Situado a la salida del inversor

Así, dividiendo la potencia de salida entre la de entrada, se puede hallar el rendimiento del inversor, lo cual proporciona información acerca de la calidad de esta conversión energética.

Battery

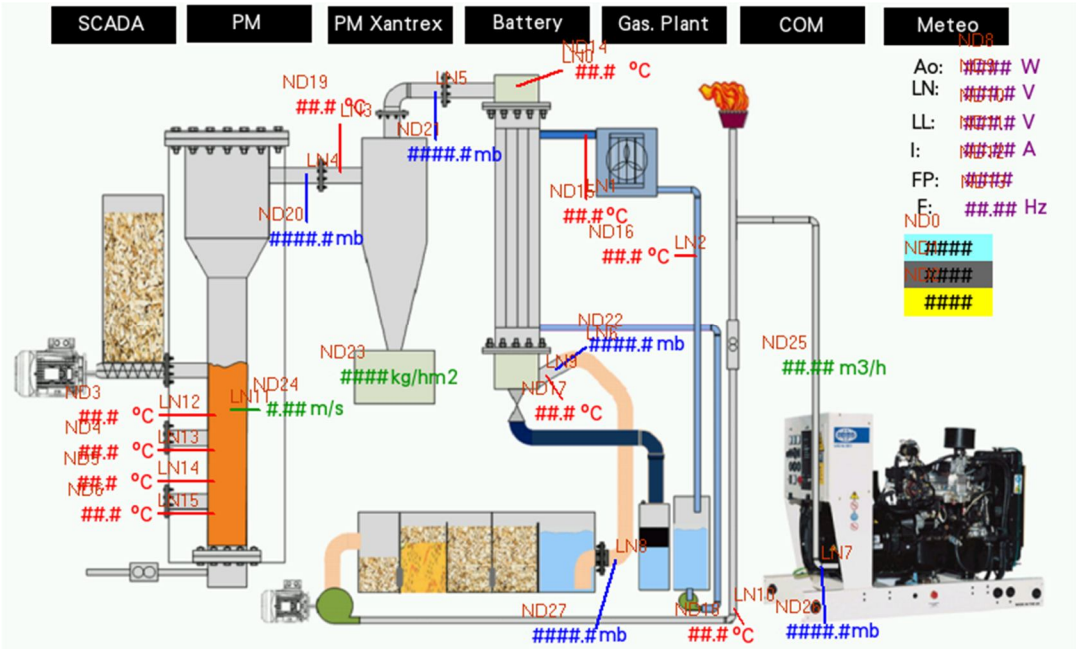


En esta pantalla aparece toda la información relativa a las baterías. En la parte izquierda se muestran valores con las características de la batería, tales como voltaje nominal, capacidad, tiempo de absorción o coeficiente de temperatura.

En la columna central, aparecen los valores que variarán dependiendo de la utilización de la batería, tales como voltaje, corriente y potencia entregada, además de la temperatura a la que se encuentra el equipo.

En la columna derecha, aparecen valores relativos a los máximos y mínimos alcanzados en el voltaje de la batería.

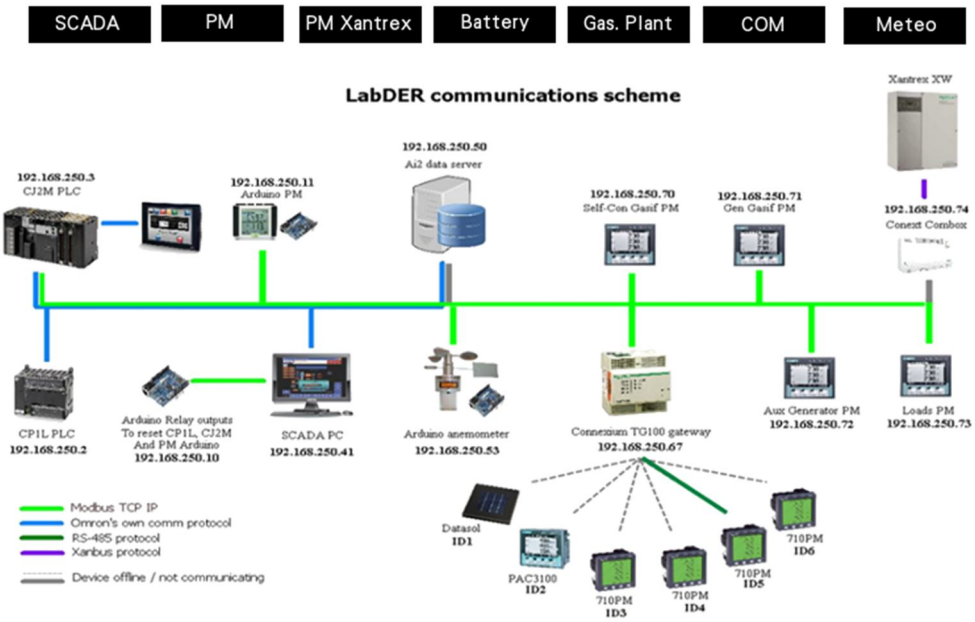
Gasification Plant



Esta pantalla representa el esquema de la planta de gasificación instalada en el laboratorio. Así, se puede monitorizar de forma general el funcionamiento de la planta, con información relativa a temperaturas, presiones y caudales en los diferentes puntos de la instalación.

Además, aparecen también valores relativos a tensiones, corrientes y potencia generada por esta planta de gasificación, lo cual permite tener una información muy completa sobre el funcionamiento de la misma.

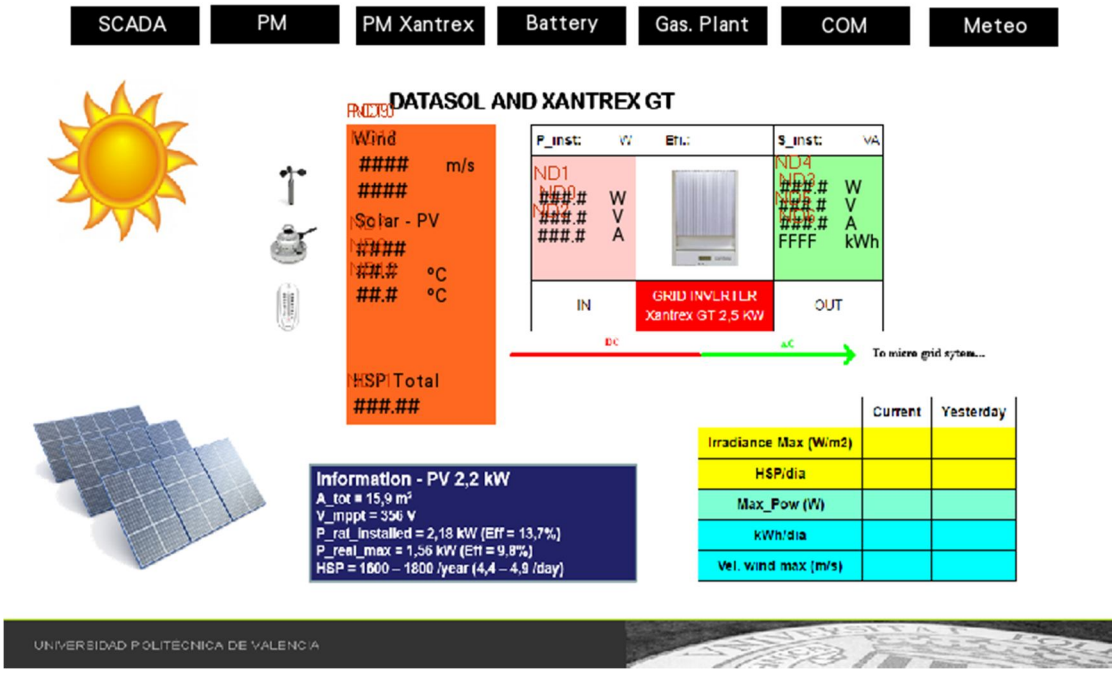
Communications



El apartado de comunicaciones representa la conexión de los equipos de la microrred en tiempo real. Cada línea de conexión tiene un bit asignado, el cual está encendido o apagado dependiendo de si existe comunicación entre ambos equipos.

Así, se puede monitorizar de manera visual qué equipos están conectados y cuál es su esquema de conexiones.

Meteo



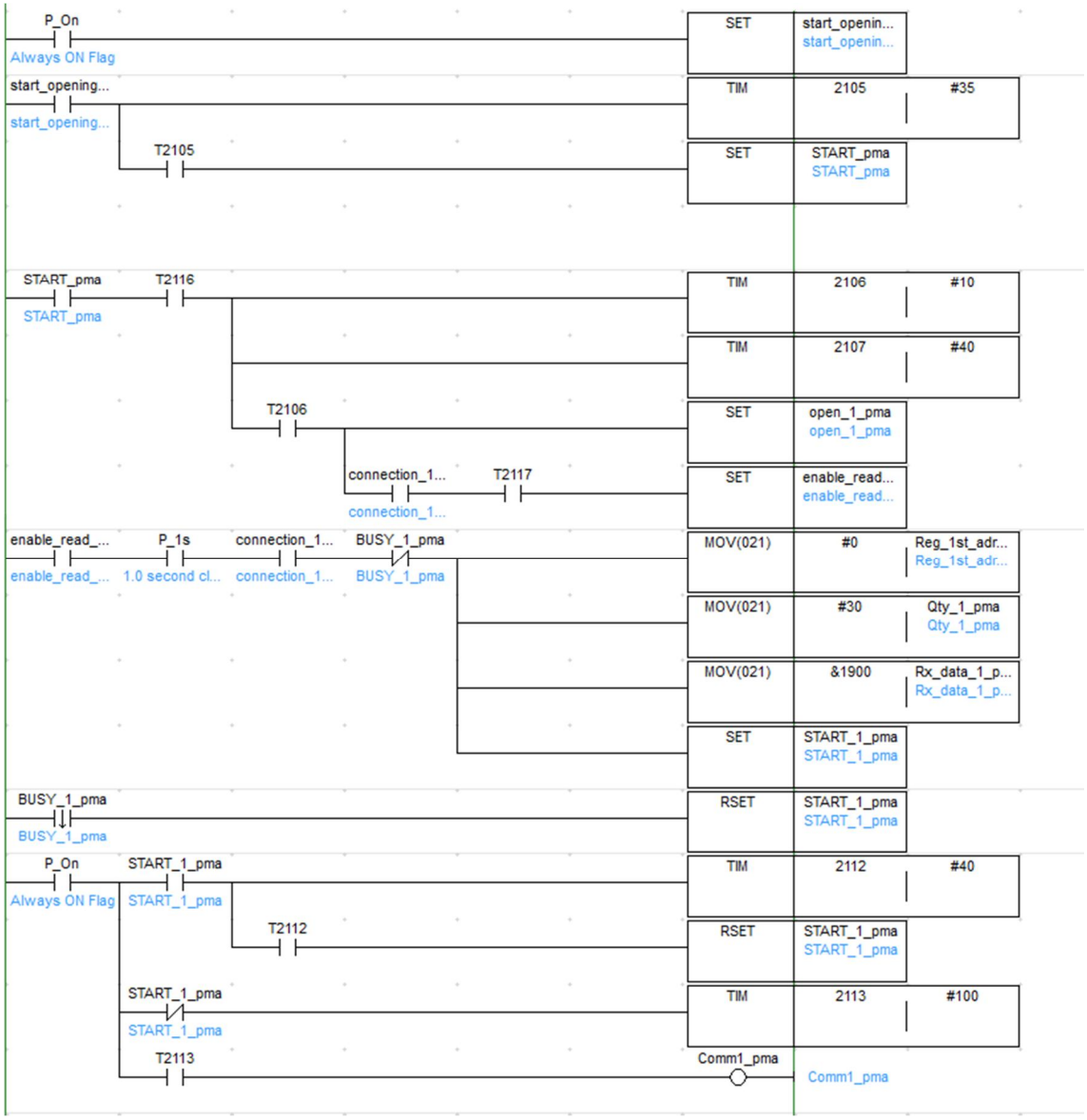
En esta última pantalla aparecen los datos relativos a la meteorología, la cual es relevante porque el laboratorio dispone de dos fuentes de energía renovables que dependen enteramente de ella.

El laboratorio dispone de un anemómetro, un piranómetro y dos termómetros que proporcionan información sobre la velocidad del viento, la irradiación solar y la temperatura ambiente. Todo ello queda reflejado en esta pantalla, además de las mediciones relativas al inversor Xantrex GT que opera a la salida de las placas solares.

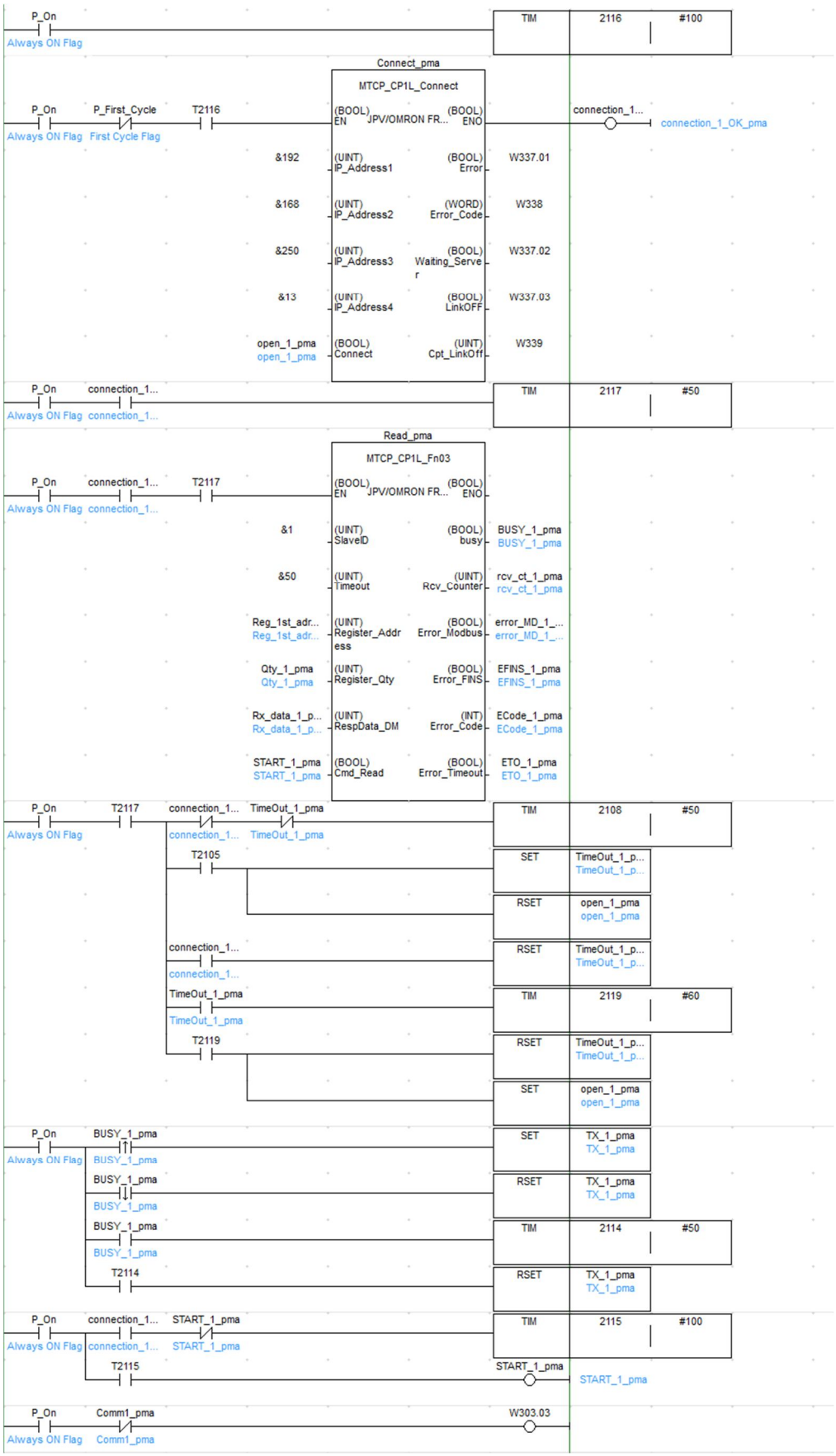
Anexo 4 - Diagrama de contactos del PLC CP1L

En este anexo se muestran los dos programas que se ejecutan en el PLC CP1L para realizar la lectura de los valores del analizador Arduino.

Sección _1_ladder



Sección _0_FB


P_On
Always ON Flag
TIM
2116
#100
Connect_pma
MTCP_CP1L_Connect
P_First_Cycle
First Cycle Flag
T2116
(BOOL) EN
JPV/OMRON FR...
(BOOL) ENO
connection_1...
connection_1_OK_pma
&192
(UINT) IP_Address1
(BOOL) Error
W337.01
&168
(UINT) IP_Address2
(WORD) Error_Code
W338
&250
(UINT) IP_Address3
(BOOL) Waiting_Server
W337.02
&13
(UINT) IP_Address4
(BOOL) LinkOFF
W337.03
open_1_pma
(BOOL) Connect
(UINT) Cpt_LinkOff
W339
2117
#50
Read_pma
MTCP_CP1L_Fn03
T2117
&1
(UINT) SlaveID
(BOOL) busy
BUSY_1_pma
&50
(UINT) Timeout
(UINT) Rcv_Counter
rcv_ct_1_pma
Reg_1st_adr...
(UINT) Register_Address
(BOOL) Error_Modbus
error_MD_1_...
Qty_1_pma
(UINT) Register_Qty
(BOOL) Error_FINS
EFINS_1_pma
Rx_data_1_p...
(UINT) RespData_DM
(INT) Error_Code
ECode_1_pma
START_1_pma
(BOOL) Cmd_Read
(BOOL) Error_Timeout
ETO_1_pma
TimeOut_1_pma
2108
SET
RSET
T2105
2119
#60
T2119
BUSY_1_pma
TX_1_pma
2114
T2114
2115
START_1_pma
Comm1_pma
W303.03

Anexo 5 - Determinación de la frecuencia de muestreo del analizador Arduino

Un factor relevante a la hora de realizar mediciones es conocer cada cuánto tiempo el analizador Arduino recoge datos de la red. Una rápida frecuencia de muestreo permite una medición mucho más exacta de los valores, además de ser fundamental para otras aplicaciones, como la medición de armónicos.

Consultando la hoja de características del analizador Siemens (5), se observa que su velocidad de muestreo es de 64 samples por ciclo, es decir, por periodo. Dado que la frecuencia es 50Hz, si se multiplican ambos valores se concluye que la velocidad de muestreo es de 3.2KHz.

Es momento de comparar este valor con el del analizador construido. Este valor dependerá tanto de las capacidades hardware de la placa, como del programa que se está ejecutando. Al consultar en la página oficial (6), se puede comprobar que la velocidad de reloj es muy superior a la del analizador Siemens, pues trabaja a 16MHz. Aun así, primero se debe comprobar la eficiencia del código empleado, que será determinante si se desea conseguir un tiempo de ejecución más reducido.

Para comprobar el tiempo de ejecución de un bucle en Arduino, simplemente se inicializa a 0 una variable llamada *last_time* (de tipo *long int*) en la cabecera del sketch, y se añade la siguiente función al comienzo del loop:

```
Serial.println(millis()-last_time);
last_time = millis();
```

Código 8.1: Comprobación de la velocidad de ejecución del sketch del analizador

Al acceder al Serial Plotter, se obtienen los siguientes resultados:

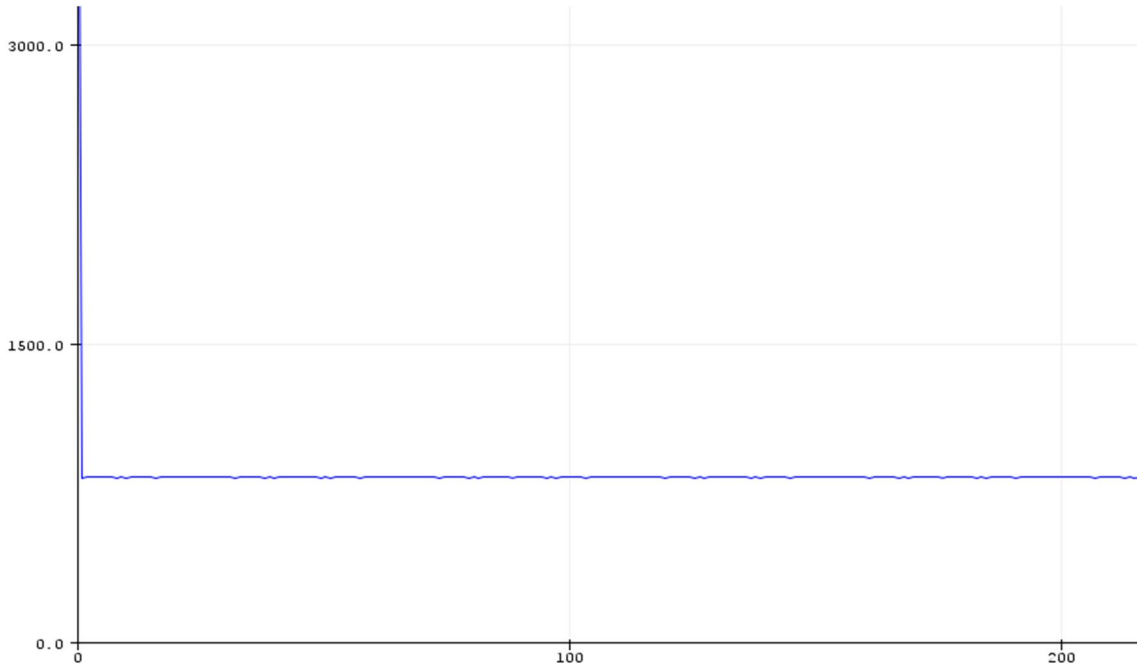


Figura 8.1: Tiempo de ejecución del sketch del analizador Arduino

Es decir, cada loop tarda en ejecutarse unos 827 milisegundos (valor obtenido del Monitor Serie). Al convertir este valor en frecuencia, se obtienen 1.2Hz, demasiado baja para la aplicación que se desea utilizar.

En este momento cabe preguntarse cuál sería la disminución del tiempo de ejecución si se optimizase el código lo suficiente. Se va a realizar una prueba para un código muy sencillo, para comprobar su velocidad de ejecución.

```
void setup()
{
  Serial.begin(115200);
}
long int last_time = 0;
void loop()
{
  Serial.println(micros()-last_time);
  last_time = micros();
}
```

Código 8.2: Comprobación de la velocidad de ejecución de un código sencillo

El parámetro más importante es la velocidad de comunicación Serial, que se ha establecido en 115200, un valor muy superior al utilizado por defecto, para evitar que la velocidad de comunicación sea un factor limitante en la velocidad de ejecución del bucle. Se compila y sube el sketch, obteniendo los siguientes resultados en el Serial Plotter:

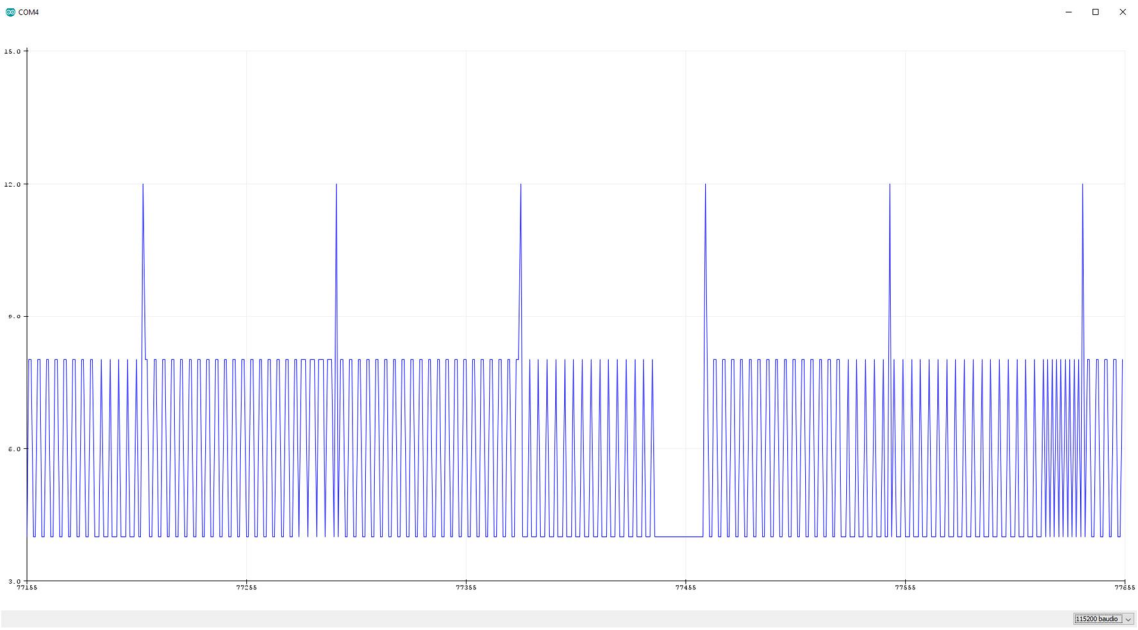


Figura 8.2: Tiempo de ejecución de un programa sencillo en la placa Arduino

En este gráfico se puede observar que el programa tarda entre 4 y 12 microsegundos en ejecutarse, por lo que suponiendo un valor medio de 8 se tendría una frecuencia de 125KHz, un valor muy superior al que ofrece el modelo comercial.

Por tanto, se puede concluir que la velocidad de muestreo de una placa Arduino depende enteramente del programa que se ejecuta. Por motivos de tiempo y extensión, no se ha estudiado hasta qué punto se podría optimizar el programa analizador de tensión para mejorar su velocidad de ejecución, pero queda patente que se pueden llegar a alcanzar velocidades muy elevadas, por lo que se pondría por delante del analizador Siemens en este aspecto.

Anexo 6 - Funciones adicionales del analizador Arduino

Medición de voltaje de línea

Uno de los más importantes defectos del analizador Arduino es que no incorpora medición directa de voltaje de línea, a pesar de que es capaz de calcularla a partir de los voltajes de fase.

Esta limitación viene impuesta por la cantidad de transformadores que se decidió integrar en el analizador. Para ser capaz de medir simultáneamente voltajes de línea y de fase, el analizador debería contar con 6 transformadores. Aumentar la cantidad de transformadores encarece el equipo, además de volverlo más pesado y voluminoso, por lo que se decidió utilizar solamente la mitad.

Además, esta decisión está motivada porque la tensión de línea es un parámetro que se puede calcular a través de las tensiones de fase, por lo que, a pesar de que no se obtenga un valor real, sí que se consigue un valor calculado muy aproximado.

Aun así, si se desea obtener información acerca de las mediciones exactas y no calcular las tensiones de línea a partir de otros valores, se debe modificar el diseño del analizador Arduino para incluir otros tres transformadores.

Al ser voltajes de línea, que para una red trifásica de baja tensión oscilan en el rango de entre 380 y 400V, los transformadores a utilizar serán modelos con una relación de transformación superior a los utilizados para la medición del voltaje de fase. En concreto, se requieren dispositivos con un voltaje primario/secundario de 400/12V.

El incremento de coste que supondría esta adición de componentes, dado que el precio de un transformador de estas características está alrededor de 17.25€, es de 51.75€. Esto demuestra que, aun añadiendo el precio de los transformadores, el coste total del analizador Arduino seguiría siendo inferior (164.39€) al del modelo comercial (670.48€).

Pantalla TFT de 3.5”

Si se observan las pantallas integradas por ambos equipos, se puede determinar que la del modelo comercial es superior a la del Arduino, pues es una pantalla monocroma de 130x100 píxeles, mientras que la del Arduino es una pantalla de 16 caracteres por fila (7x5 píxeles por carácter) y dos filas.

Aunque técnicamente es superior, la información que son capaces de mostrar ambas pantallas es prácticamente la misma, por lo que la funcionalidad es parecida. Además, la placa Arduino tiene la ventaja de que se le pueden conectar otras pantallas, aparte de la que se ha utilizado para este analizador.

Un ejemplo se puede encontrar en la tienda online DX.com, en la que se pone a la venta una pantalla TFT de 3.5” (resolución 320x480) compatible con Arduino por 9.66€, con una librería que facilita su integración en el software. Gracias a ello, se tendría una pantalla de más calidad que la integrada en el modelo comercial, y a un menor coste.

Medición de la frecuencia

A pesar de que la función de determinación de la frecuencia no ha sido incluida en el analizador construido, se ha realizado un estudio sobre la misma (27), concluyendo que, si se necesitase esta función, no se necesitaría agregar ningún componente hardware extra al analizador. La solución reside en incorporar a la librería EmonLib el código necesario para su funcionamiento, y añadir algunas instrucciones al sketch Arduino.

Primero, se muestra el código que se debe añadir a la librería EmonLib:

```
void EnergyMonitor::calcF(int crossings)  {
boolean stf = false;
unsigned long first, second;
unsigned long delta = 0;
while(stf == false)                              //wait for first crossing
{
  startF = analogRead(inPinV);                  //using the voltage waveform
  if ((startF > 512) && (startF < 520))  {
   stf = true;                                     //check its within range
   first = millis();
   delay(4);                     //Make sure no second read is in this range
  }
}
stf = false;
for(int i = 0 ; i < crossings ; i++)   {
  while(stf == false)  {                           //wait for next crossing
   startF = analogRead(inPinV);                //using the voltage waveform
   if ((startF > 512) && (startF < 528))   {
    stf = true;
    second = millis();
    delta += (second - first);                  //Add time between crossings
    first = second;
    delay(4);                     //Make sure no second read is in this range
   }
  }
  stf = false;
}
frequency = (crossings / 2) / ((double)delta / 1000);
}
```

Código 8.3: Cálculo de la frecuencia en el analizador Arduino

Y el aspecto que tendría un programa sencillo de determinación de la frecuencia:

```
#include "EmonLib.h"             // Include Emon Library
EnergyMonitor emon1;             // Create an instance

void setup()
{
  Serial.begin(38400);

  emon1.voltage(4, 200.8, 0.5);  // Voltage: input pin, calibration,
phase_shift

void loop()
{
  emon1.calcF(50);     // 50 zero crossings
  float freq = emon1.frequency;

  Serial.print("Frequency: ");
  Serial.println(freq);

  delay(1000);
}
```

Código 8.4: Sketch para la determinación de la frecuencia

Una vez visto el código, se puede proceder a describir el algoritmo que utiliza para realizar el cálculo de la frecuencia. Dado que es la inversa del periodo, se va a medir el tiempo que transcurre entre un paso por cero y otro de la onda de tensión.

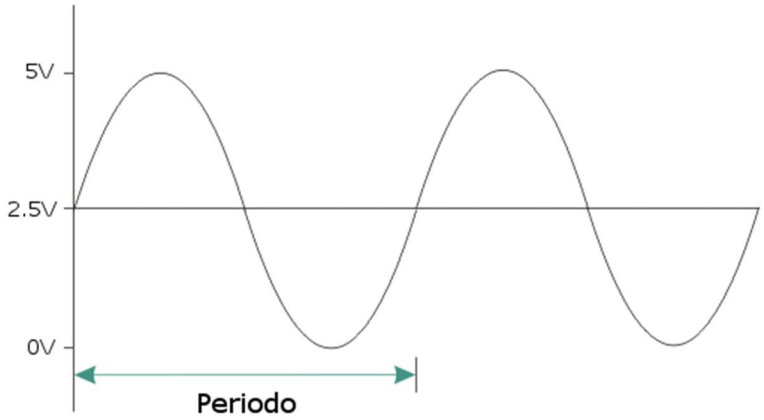


Figura 8.3: Descripción del paso por cero de la tensión

Para determinar este tiempo entre ceros, se realiza el siguiente procedimiento: se toma de uno de los inputs analógicos de Arduino la señal de tensión que se está midiendo. A continuación, lee su valor utilizando la función *analogRead*, la cual convierte la amplitud de la onda de tensión analógica que se está midiendo (en un rango de 0 a 5V) a un valor digital que está comprendido entre 0 y 1023.

Dado que la onda de tensión tendrá un aspecto parecido al de la *Figura 2.1: Circuito medidor de corriente* el paso por cero de la onda se realizará a una amplitud de aproximadamente 2.5V. Convirtiendo ese valor a digital, se tendría una amplitud de 512.

Por ello, cada vez que se detecte que la amplitud está comprendida entre 512 y 520, se considerará que la onda ha pasado por cero. A partir de ese instante, se medirá el tiempo transcurrido hasta el siguiente paso por cero. Este proceso se repetirá una vez más, dado que cada periodo tiene dos pasos por cero.

Al sumar ambos tiempos, tendremos el periodo de la onda de tensión. Esta medición se realizará tantas veces como se necesite, pudiendo configurar el parámetro *crossings* para este fin. Cuanto mayor sea este parámetro, más tardará el programa en dar una respuesta, pero a cambio la medición será más precisa.

Finalmente, una vez obtenido el periodo, realizando la inversa del mismo se consigue el valor deseado de frecuencia. Así, se demuestra que esta característica no está integrada en el analizador construido, pero se podría implementar con una actualización de software.

Alarma

La última función característica que incluye el analizador comercial es la función de alarma, mediante la cual se puede configurar el analizador para que emita una señal acústica cuando un valor se sale de un rango configurado durante un determinado periodo de tiempo.

Programar un código en el sketch Arduino para comparar los valores leídos con un máximo o mínimo es una tarea muy sencilla, por lo que lo único que faltaría sería encontrar un elemento capaz de reproducir un sonido cuando se lo ordene la placa.

Investigando sobre esto, se ha encontrado un elemento que es capaz de producir sonidos a partir de pulsos eléctricos. Se denomina *buzzer,* el cual es un dispositivo electrónico pasivo, que utiliza el efecto piezoeléctrico como base de su funcionamiento. Al ser atravesado por una tensión eléctrica variable, el cristal de cuarzo que incorpora vibra a la frecuencia de la señal, generando un sonido a esa frecuencia.

Por tanto, este elemento resultaría perfecto para esta aplicación, pues se podrían configurar varios tonos de alarma. Además, su bajo precio (1€) posibilita su adición a la placa sin un incremento relevante del coste.

La parte que respecta a la programación de la placa se describe a continuación. Arduino dispone de una función llamada *tone()*, con la cual se puede generar una señal de tensión con la frecuencia y duración deseadas. El código de funcionamiento sería parecido al siguiente:

```
int speakerPin = 9;
int freq = 294;

void setup()  { }

void loop()   {
    for (int i = 0; i < 10; i++)  {
        tone(speakerPin, freq);
        delay(500);
    }
    noTone(speakerPin);
}
```

Con este código, se ejecuta la función *tone()* durante 10 intervalos. Esta función tiene como parámetros el pin en el que está conectado el *buzzer* y la frecuencia que se necesite reproducir.

Así, se demuestra que el analizador Arduino puede equiparar sus funciones de alama a las del modelo comercial, únicamente ampliando la programación de la placa y añadiéndole este componente de bajo coste.

Anexo 7 - Índice de figuras y fragmentos de código

En este anexo se listan todas las figuras (imágenes, gráficos…) y fragmentos de código que se han utilizado a lo largo de este documento, junto con una referencia a la página en la que aparecen.

El formato de numeración es parecido al de la bibliografía: un número que indica el número de capítulo en el que se encuentra, un punto y otro número identificador ascendente único para cada tipo de elemento.

Figuras

Fragmentos de código

Tablas

Anexo 8 - Mapa de direcciones del PLC

En este anexo se van a detallar las direcciones de los registros del PLC utilizados en el terminal HMI, con el objetivo de relacionar esta dirección con el valor y formato que contiene. Se ha subdividido el contenido en bloques para una mejor lectura.

SCADA

Dirección CX Supervisor	Dirección PLC	PLC utilizado	Medidor	Formato	Descripción
wind_vel_ms_CX	D1801	CP1L	-	Unsigned integer	Velocidad del viento
Radiacion_Media	D6600	CJ2M	-	Unsigned integer	Irradiación solar
Temp_Ambiente	D6610	CJ2M	-	Unsigned integer	Temperatura ambiente detrás de la placa
Temp_Sup_Panel	D6611	CJ2M	-	Unsigned integer	Temperatura superficie del panel
E_dia	D8006	CJ2M	-	Unsigned integer	Producción del día del inversor
P_in_DC	D8009	CJ2M	PM15	Unsigned integer	Potencia de entrada del inversor
P_out_AC	D8012	CJ2M	PM16	Unsigned integer	Potencia de salida del inversor
kWh_hist	D8017	CJ2M	PM16	Unsigned integer	Energía total acumulada producida por las placas solares
ID6_Real_Power_CX	D7636	CJ2M	PM5	Unsigned integer	Potencia de salida del aerogenerador
ID6_Real_Energy_CX	D7720	CJ2M	PM5	Unsigned integer	Energía total acumulada producida por el aerogenerador
Total_act_pow_73_CX	D9964	CJ2M	PM9	Unsigned integer	Potencia consumida por las cargas
Act_en_imp_73_CX	D13200	CJ2M	PM9	Unsigned integer	Energía total acumulada consumida por las cargas
ID3_Real_Power_CX	D7036	CJ2M	PM2	Unsigned integer	Potencia proveniente de fuentes renovables
ID3_Real_Energy_CX	D7120	CJ2M	PM2	Unsigned integer	Energía total acumulada proveniente de fuentes renovables
ID4_Real_Power_CX	D7236	CJ2M	PM3	Unsigned integer	Potencia consumida por las cargas
ID4_Real_Energy_CX		CJ2M	PM3	Unsigned integer	Energía total acumulada consumida por las cargas
real_pow	D14305	CJ2M	ARDUINO PM	Unsigned integer	Potencia consumida por las cargas
act_en_imp_A	D14309	CJ2M	ARDUINO PM	Unsigned integer	Parte entera de la energía total acumulada consumida por las cargas

act_en_imp_B	D14310	CJ2M	ARDUINO PM	Unsigned integer	Parte decimal de la energía total acumulada consumida por las cargas
XW_Grid_AC_Power	D5102	CJ2M	PM11	Unsigned integer	Potencia intercambiada con la red por parte del Xantrex
XW_Gener_Power	D5172	CJ2M	PM12	Unsigned integer	Potencia intercambiada con la planta de gasificación y el generador auxiliar por parte del Xantrex
XW_Load_Power	D5154	CJ2M	PM13	Unsigned integer	Potencia intercambiada con las energías renovables y las cargas por parte del Xantrex
XW_Battery_Power	D5084	CJ2M	PM14	Unsigned integer	Potencia intercambiada con las baterías por parte del Xantrex
TOT_ACTI_POWER_AUTOC	D1064	CJ2M	PM6	Float	Potencia de la red consumida por la planta de gasificación
ACT_EN_IMP_T1_B	D12000	CJ2M	PM6	Float	Energía total acumulada de la red consumida por la planta de gasificación
TOT_ACTI_POWER_GENER	D2064	CJ2M	PM7	Float	Potencia entregada por la planta de gasificación
ACT_EN_IMP_T1	D11000	CJ2M	PM7	Float	Energía total acumulada entregada por la planta de gasificación
Total_act_pow_72_CX	D9564	CJ2M	PM8	Float	Potencia entregada por el generador auxiliar
Act_en_imp_72_CX	D13000	CJ2M	PM8	Float	Energía total acumulada entregada por el generador auxiliar
Tot_act_pow	D6848	CJ2M	PM10	Float	Potencia intercambiada con la red
Active_Energy	D6990	CJ2M	PM10	Float	Energía activa intercambiada con la red
Reactive_Energy	D6994	CJ2M	PM10	Float	Energía reactiva intercambiada con la red

Battery

Dirección CX Supervisor	Dirección PLC	PLC utilizado	Medidor	Formato	Descripción
XW_Bat_Nominal_Volt	D5372	CJ2M	PM14	Unsigned integer	Tensión nominal de la batería
XW_Bat_Bank_Capacity	D6374	CJ2M	PM14	Unsigned integer	Capacidad nominal de la batería
XW_Bat_Temp_Coeff	D6375	CJ2M	PM14	Unsigned integer	Coeficiente de temperatura de la batería
XW_Bat_Buck_Volt	D6410	CJ2M	PM14	Unsigned integer	Tensión buck de la batería
XW_Bat_Float_Volt	D6416	CJ2M	PM14	Unsigned integer	Tensión float de la batería

XW_Bat_Equalize_Volt	D6406	CJ2M	PM14	Unsigned integer	Tensión de ecualización de la batería
XW_Bat_Absorpt_Time	D6414	CJ2M	PM14	Unsigned integer	Tiempo de absorción de la batería
XW_Battery_Voltage	D5080	CJ2M	PM14	Unsigned integer	Tensión actual de la batería
XW_Battery_Current	D5082	CJ2M	PM14	Unsigned integer	Corriente que circula por la batería
XW_Battery_Power	D5084	CJ2M	PM14	Unsigned integer	Potencia intercambiada con la batería
XW_Battery_Temperat	D5086	CJ2M	PM14	Unsigned integer	Temperatura de la batería
High_Battery	D6394	CJ2M	PM14	Unsigned integer	Tensión máxima alcanzada por la batería en operación
Battery_Delay_High	D6398	CJ2M	PM14	Unsigned integer	Tiempo que la batería opera a esta tensión máxima
Low_Battery	D6380	CJ2M	PM14	Unsigned integer	Tensión mínima alcanzada por la batería en operación
Battery_Dellay_Low	D6382	CJ2M	PM14	Unsigned integer	Tiempo que la batería opera a la mínima máxima

PM Arduino

Dirección CX Supervisor	**Dirección PLC**	**PLC utilizado**	**Medidor**	**Formato**	**Descripción**
nominal_volt	D14302	CJ2M	ARDUINO PM	Unsigned integer	Tensión media medida por el analizador
nominal_curr	D14304	CJ2M	ARDUINO PM	Unsigned integer	Corriente media medida por el analizador
real_pow	D14305	CJ2M	ARDUINO PM	Unsigned integer	Potencia activa
react_pow	D14307	CJ2M	ARDUINO PM	Unsigned integer	Potencia reactiva
app_pow	D14306	CJ2M	ARDUINO PM	Unsigned integer	Potencia aparente
pow_fact	D14308	CJ2M	ARDUINO PM	Unsigned integer	Factor de potencia
freq	D14303	CJ2M	ARDUINO PM	Unsigned integer	Frecuencia
act_en_imp_A	D14309	CJ2M	ARDUINO PM	Unsigned integer	Energía activa importada
-	D1900	CP1L	ARDUINO PM 2	Unsigned integer	Tensión de la fase 1
-	D1902	CP1L	ARDUINO PM 2	Unsigned integer	Tensión de la fase 2
-	D1904	CP1L	ARDUINO PM 2	Unsigned integer	Tensión de la fase 3
-	D1906	CP1L	ARDUINO PM 2	Unsigned integer	Tensión de línea 1

-	D1908	CP1L	ARDUINO PM 2	Unsigned integer	Tensión de línea 2
-	D1910	CP1L	ARDUINO PM 2	Unsigned integer	Tensión de línea 3
-	D1912	CP1L	ARDUINO PM 2	Unsigned integer	Corriente de fase 1
-	D1914	CP1L	ARDUINO PM 2	Unsigned integer	Corriente de fase 2
-	D1916	CP1L	ARDUINO PM 2	Unsigned integer	Corriente de fase 3
-	D1918	CP1L	ARDUINO PM 2	Unsigned integer	Corriente de línea 1
-	D1920	CP1L	ARDUINO PM 2	Unsigned integer	Corriente de línea 2
-	D1922	CP1L	ARDUINO PM 2	Unsigned integer	Corriente de línea 3
-	D1924	CP1L	ARDUINO PM 2	Unsigned integer	Potencia activa 1
-	D1926	CP1L	ARDUINO PM 2	Unsigned integer	Potencia activa 2
-	D1928	CP1L	ARDUINO PM 2	Unsigned integer	Potencia activa 3
-	D1930	CP1L	ARDUINO PM 2	Unsigned integer	Potencia reactiva 1
-	D1932	CP1L	ARDUINO PM 2	Unsigned integer	Potencia reactiva 2
-	D1934	CP1L	ARDUINO PM 2	Unsigned integer	Potencia reactiva 3
-	D1936	CP1L	ARDUINO PM 2	Unsigned integer	Potencia aparente 1
-	D1938	CP1L	ARDUINO PM 2	Unsigned integer	Potencia aparente 2
-	D1940	CP1L	ARDUINO PM 2	Unsigned integer	Potencia aparente 3
-	D1942	CP1L	ARDUINO PM 2	Unsigned integer	Factor de potencia 1
-	D1944	CP1L	ARDUINO PM 2	Unsigned integer	Factor de potencia 2
-	D1946	CP1L	ARDUINO PM 2	Unsigned integer	Factor de potencia 3
-	D1948	CP1L	ARDUINO PM 2	Unsigned integer	Energía activa
-	D1950	CP1L	ARDUINO PM 2	Unsigned integer	Energía reactiva
-	D1952	CP1L	ARDUINO PM 2	Unsigned integer	Energía aparente
-	D1954	CP1L	ARDUINO PM 2	Unsigned integer	Segundos desde que se inició la medición del analizador
-	D1956	CP1L	ARDUINO PM 2	Unsigned integer	Minutos desde que se inició la medición del analizador

-	D1958	CP1L	ARDUINO PM 2	Unsigned integer	Horas desde que se inició la medición del analizador
-	D1960	CP1L	ARDUINO PM 2	Unsigned integer	Días desde que se inició la medición del analizador

PM Xantrex

Dirección CX Supervisor	Dirección PLC	PLC utilizado	Medidor	Formato	Descripción
XW_Grid_AC_Voltage	D5098	CJ2M	PM11	Unsigned integer	Tensión media
XW_Grid_AC_Current	D5100	CJ2M	PM11	Unsigned integer	Corriente media
XW_Grid_AC_Power	D5102	CJ2M	PM11	Unsigned integer	Potencia activa
XW_Grid_AC_Frequency	D5097	CJ2M	PM11	Unsigned integer	Frecuencia
XW_Gener_Voltage	D5162	CJ2M	PM12	Unsigned integer	Tensión media
XW_Gener_Current	D5164	CJ2M	PM12	Unsigned integer	Corriente media
XW_Gener_Power	D5172	CJ2M	PM12	Unsigned integer	Potencia activa
XW_Gener_Frequency	D5166	CJ2M	PM12	Unsigned integer	Frecuencia
XW_Load_Voltage	D5140	CJ2M	PM13	Unsigned integer	Tensión media
XW_Load_Current	D5150	CJ2M	PM13	Unsigned integer	Corriente media
XW_Load_Power	D5154	CJ2M	PM13	Unsigned integer	Potencia activa
XW_Load_Frequency	D5152	CJ2M	PM13	Unsigned integer	Frecuencia
XW_Battery_Voltage	D5080	CJ2M	PM14	Unsigned integer	Tensión actual de la batería
XW_Battery_Current	D5082	CJ2M	PM14	Unsigned integer	Corriente que circula por la batería
XW_Battery_Power	D5084	CJ2M	PM14	Unsigned integer	Potencia intercambiada con la batería
V_in_DC	D8007	CJ2M	PM15	Unsigned integer	Tensión media
I_in_DC	D8008	CJ2M	PM15	Unsigned integer	Corriente media
P_in_DC	D8009	CJ2M	PM15	Unsigned integer	Potencia activa
V_out_AC	D8010	CJ2M	PM16	Unsigned integer	Tensión media
I_out_AC	D8011	CJ2M	PM16	Unsigned integer	Corriente media

P_out_AC	D8012	CJ2M	PM16	Unsigned integer	Potencia activa
kWh_hist	D8017	CJ2M	PM16	Unsigned integer	Energía activa importada

PM Schneider

Dirección CX Supervisor	Dirección PLC	PLC utilizado	Medidor	Formato	Descripción
ID3_Voltage_CX	D7033	CJ2M	PM2	Unsigned integer	Tensión media
ID3_Current_CX	D7020	CJ2M	PM2	Unsigned integer	Corriente media
ID3_Real_Power_CX	D7036	CJ2M	PM2	Unsigned integer	Potencia activa
ID3_App_Power_CX	D7039	CJ2M	PM2	Unsigned integer	Potencia aparente
ID3_Power_Factor_CX	D7036	CJ2M	PM2	Unsigned integer	Factor de potencia
ID3_Frequency_CX	D7013	CJ2M	PM2	Unsigned integer	Frecuencia
ID3_Real_Energy_CX	D7120	CJ2M	PM2	Unsigned integer	Energía activa importada
ID4_Voltage_CX	D7233	CJ2M	PM3	Unsigned integer	Tensión media
ID4_Current_CX	D7220	CJ2M	PM3	Unsigned integer	Corriente media
ID4_Real_Power_CX	D7236	CJ2M	PM3	Unsigned integer	Potencia activa
ID4_App_Power_CX	D7239	CJ2M	PM3	Unsigned integer	Potencia aparente
ID4_Power_Factor_CX	D7236	CJ2M	PM3	Unsigned integer	Factor de potencia
ID4_Frequency_CX	D7213	CJ2M	PM3	Unsigned integer	Frecuencia
ID4_Real_Energy_CX	D7320	CJ2M	PM3	Unsigned integer	Energía activa importada
ID5_Voltage_CX	D7433	CJ2M	PM4	Unsigned integer	Tensión media
ID5_Current_CX	D7420	CJ2M	PM4	Unsigned integer	Corriente media
ID5_Real_Power_CX	D7436	CJ2M	PM4	Unsigned integer	Potencia activa
ID5_App_Power_CX	D7439	CJ2M	PM4	Unsigned integer	Potencia aparente
ID5_Power_Factor_CX	D7436	CJ2M	PM4	Unsigned integer	Factor de potencia
ID5_Frequency_CX	D7413	CJ2M	PM4	Unsigned integer	Frecuencia

ID5_Real_Energy_CX	D7520	CJ2M	PM4	Unsigned integer	Energía activa importada
ID6_Voltage_CX	D7633	CJ2M	PM5	Unsigned integer	Tensión media
ID6_Current_CX	D7620	CJ2M	PM5	Unsigned integer	Corriente media
ID6_Real_Power_CX	D7636	CJ2M	PM5	Unsigned integer	Potencia activa
ID6_App_Power_CX	D7639	CJ2M	PM5	Unsigned integer	Potencia aparente
ID6_Power_Factor_CX	D7636	CJ2M	PM5	Unsigned integer	Factor de potencia
ID6_Frequency_CX	D7613	CJ2M	PM5	Unsigned integer	Frecuencia
ID6_Real_Energy_CX	D7720	CJ2M	PM5	Unsigned integer	Energía activa importada

PM Siemens

Dirección CX Supervisor	**Dirección PLC**	**PLC utilizado**	**Medidor**	**Formato**	**Descripción**
MEDIUM_LN_AUTO	D1056	CJ2M	PM6	Float	Tensión media
MEDIUM_I_AUTO	D1060	CJ2M	PM6	Float	Corriente media
TOT_ACTI_POWER_AUTOC	D1064	CJ2M	PM6	Float	Potencia activa
Tot_ract_pow	D6850	CJ2M	PM6	Float	Potencia reactiva
TOT_APP_POWER_AUTOC	D1062	CJ2M	PM6	Float	Potencia aparente
FP_AUTOC	D1068	CJ2M	PM6	Float	Factor de potencia
FREQUENCY_AUTOC	D1054	CJ2M	PM6	Float	Frecuencia
ACT_EN_IMP_T1	D11000	CJ2M	PM6	Double float	Energía activa importada
REAC_EN_IMP_T1	D11016	CJ2M	PM6	Double float	Energía reactiva importada
MEDIUM_LN_GENER	D2056	CJ2M	PM7	Float	Tensión media
MEDIUM_I_GENER	D2060	CJ2M	PM7	Float	Corriente media
TOT_ACTI_POWER_GENER	D2064	CJ2M	PM7	Float	Potencia activa
TOT_REAC_POWER_GENER	D2066	CJ2M	PM7	Float	Potencia reactiva
TOT_APP_POWER_GENER	D2062	CJ2M	PM7	Float	Potencia aparente
FP_GEN	D2068	CJ2M	PM7	Float	Factor de potencia
FREQUENCY_GENER	D2054	CJ2M	PM7	Float	Frecuencia
ACT_EN_IMP_T1_B	D12000	CJ2M	PM7	Double float	Energía activa importada
REAC_EN_IMP_T1_B	D12016	CJ2M	PM7	Double float	Energía reactiva importada

V_an_72	D9500	CJ2M	PM8	Float	Tensión media
Current_a_72	D9512	CJ2M	PM8	Float	Corriente media
Total_act_pow_72	D9564	CJ2M	PM8	Float	Potencia activa
Total_ract_pow_72	D9566	CJ2M	PM8	Float	Potencia reactiva
Total_app_pow_72	D9562	CJ2M	PM8	Float	Potencia aparente
Pow_factor_a_72	D9536	CJ2M	PM8	Float	Factor de potencia
Freq_72	D9554	CJ2M	PM8	Float	Frecuencia
Act_en_imp_72	D13000	CJ2M	PM8	Double float	Energía activa importada
Ract_en_imp_72	D13016	CJ2M	PM8	Double float	Energía reactiva importada
Act_en_exp_72	D13008	CJ2M	PM8	Double float	Energía activa exportada
Ract_en_exp_72	D13024	CJ2M	PM8	Double float	Energía reactiva exportada
-	D19504	CJ2M	PM9	Float	Tensión media
-	D19506	CJ2M	PM9	Float	Corriente media
Total_act_pow_73	D9964	CJ2M	PM9	Float	Potencia activa
Total_ract_pow_73	D9966	CJ2M	PM9	Float	Potencia reactiva
Total_app_pow_73	D9962	CJ2M	PM9	Float	Potencia aparente
Pow_factor_a_73	D9968	CJ2M	PM9	Float	Factor de potencia
Freq_73	D9554	CJ2M	PM9	Float	Frecuencia
Act_en_imp_73	D13200	CJ2M	PM9	Double float	Energía activa importada
Ract_en_imp_73	D13216	CJ2M	PM9	Double float	Energía reactiva importada
Act_en_exp_73	D13208	CJ2M	PM9	Double float	Energía activa exportada
Ract_en_exp_73	D13224	CJ2M	PM9	Double float	Energía reactiva exportada
-	D19500	CJ2M	PM10	Float	Tensión media
-	D19502	CJ2M	PM10	Float	Corriente media
Total_act_pow	D6848	CJ2M	PM10	Float	Potencia activa
Total_ract_pow	D6850	CJ2M	PM10	Float	Potencia reactiva
Total_app_pow	D6846	CJ2M	PM10	Float	Potencia aparente
Tot_pow_factor	D6852	CJ2M	PM10	Float	Factor de potencia
Freq	D6838	CJ2M	PM10	Float	Frecuencia
Active_Energy	D6990	CJ2M	PM10	Double float	Energía activa importada
Reactive_Energy	D6994	CJ2M	PM10	Double float	Energía reactiva importada
V_L12_AUTO	D1006	CJ2M	PM6	Float	Tensión de línea 1
V_L23_AUTO	D1008	CJ2M	PM6	Float	Tensión de línea 2
V_L13_AUTO	D1010	CJ2M	PM6	Float	Tensión de línea 3
I1_AUTO	D1012	CJ2M	PM6	Float	Corriente de línea 1
I2_AUTO	D1014	CJ2M	PM6	Float	Corriente de línea 2
I3_AUTO	D1016	CJ2M	PM6	Float	Corriente de línea 3

P_ACT_L1_AUTO	D1024	CJ2M	PM6	Float	Potencia activa 1
P_ACT_L2_AUTO	D1026	CJ2M	PM6	Float	Potencia activa 2
P_ACT_L3_AUTO	D1028	CJ2M	PM6	Float	Potencia activa 3
P_REAC_L1_AUTO	D1030	CJ2M	PM6	Float	Potencia reactiva 1
P_REAC_L2_AUTO	D1032	CJ2M	PM6	Float	Potencia reactiva 2
P_REAC_L3_AUTO	D1034	CJ2M	PM6	Float	Potencia reactiva 3
P_APP_L1_AUTO	D1018	CJ2M	PM6	Float	Potencia aparente 1
P_APP_L2_AUTO	D1020	CJ2M	PM6	Float	Potencia aparente 2
P_APP_L3_AUTO	D1022	CJ2M	PM6	Float	Potencia aparente 3
V_L12_GENER	D2006	CJ2M	PM7	Float	Tensión de línea 1
V_L23_GENER	D2008	CJ2M	PM7	Float	Tensión de línea 2
V_L13_GENER	D2010	CJ2M	PM7	Float	Tensión de línea 3
I1_GENER	D2012	CJ2M	PM7	Float	Corriente de línea 1
I2_GENER	D2014	CJ2M	PM7	Float	Corriente de línea 2
I3_GENER	D2016	CJ2M	PM7	Float	Corriente de línea 3
P_ACT_L1_GENER	D2024	CJ2M	PM7	Float	Potencia activa 1
P_ACT_L2_GENER	D2026	CJ2M	PM7	Float	Potencia activa 2
P_ACT_L3_GENER	D2028	CJ2M	PM7	Float	Potencia activa 3
P_REAC_L1_GENER	D2030	CJ2M	PM7	Float	Potencia reactiva 1
P_REAC_L2_GENER	D2032	CJ2M	PM7	Float	Potencia reactiva 2
P_REAC_L3_GENER	D2034	CJ2M	PM7	Float	Potencia reactiva 3
P_APP_L1_GENER	D2018	CJ2M	PM7	Float	Potencia aparente 1
P_APP_L2_GENER	D2020	CJ2M	PM7	Float	Potencia aparente 2
P_APP_L3_GENER	D2022	CJ2M	PM7	Float	Potencia aparente 3
V_an_72	D9500	CJ2M	PM8	Float	Tensión de fase 1
V_bn_72	D9502	CJ2M	PM8	Float	Tensión de fase 2
V_cn_72	D9504	CJ2M	PM8	Float	Tensión de fase 3
V_ab_72	D9506	CJ2M	PM8	Float	Tensión de línea 1
V_bc_72	D9508	CJ2M	PM8	Float	Tensión de línea 2
V_ca_72	D9510	CJ2M	PM8	Float	Tensión de línea 3
Current_a_72	D9512	CJ2M	PM8	Float	Corriente de línea 1
Current_b_72	D9514	CJ2M	PM8	Float	Corriente de línea 2
Current_c_72	D9516	CJ2M	PM8	Float	Corriente de línea 3
Act_pw_a_72	D9524	CJ2M	PM8	Float	Potencia activa 1
Act_pw_b_72	D9526	CJ2M	PM8	Float	Potencia activa 2
Act_pw_c_72	D9528	CJ2M	PM8	Float	Potencia activa 3
React_pw_a_72	D9530	CJ2M	PM8	Float	Potencia reactiva 1
React_pw_b_72	D9532	CJ2M	PM8	Float	Potencia reactiva 2
React_pw_c_72	D9534	CJ2M	PM8	Float	Potencia reactiva 3
App_pw_a_72	D9518	CJ2M	PM8	Float	Potencia aparente 1
App_pw_b_72	D9520	CJ2M	PM8	Float	Potencia aparente 2
App_pw_c_72	D9522	CJ2M	PM8	Float	Potencia aparente 3
V_an_73	D9900	CJ2M	PM9	Float	Tensión de fase 1
V_bn_73	D9902	CJ2M	PM9	Float	Tensión de fase 2
V_cn_73	D9904	CJ2M	PM9	Float	Tensión de fase 3

V_ab_73	D9906	CJ2M	PM9	Float	Tensión de línea 1
V_bc_73	D9908	CJ2M	PM9	Float	Tensión de línea 2
V_ca_73	D9910	CJ2M	PM9	Float	Tensión de línea 3
Current_a_73	D9912	CJ2M	PM9	Float	Corriente de línea 1
Current_b_73	D9914	CJ2M	PM9	Float	Corriente de línea 2
Current_c_73	D9916	CJ2M	PM9	Float	Corriente de línea 3
Act_pw_a_73	D9924	CJ2M	PM9	Float	Potencia activa 1
Act_pw_b_73	D9926	CJ2M	PM9	Float	Potencia activa 2
Act_pw_c_73	D9928	CJ2M	PM9	Float	Potencia activa 3
React_pw_a_73	D9930	CJ2M	PM9	Float	Potencia reactiva 1
React_pw_b_73	D9932	CJ2M	PM9	Float	Potencia reactiva 2
React_pw_c_73	D9934	CJ2M	PM9	Float	Potencia reactiva 3
App_pw_a_73	D9918	CJ2M	PM9	Float	Potencia aparente 1
App_pw_b_73	D9920	CJ2M	PM9	Float	Potencia aparente 2
App_pw_c_73	D9922	CJ2M	PM9	Float	Potencia aparente 3
V_an	D6800	CJ2M	PM10	Float	Tensión de fase 1
V_bn	D6802	CJ2M	PM10	Float	Tensión de fase 2
V_cn	D6804	CJ2M	PM10	Float	Tensión de fase 3
V_ab	D6806	CJ2M	PM10	Float	Tensión de línea 1
V_bc	D6808	CJ2M	PM10	Float	Tensión de línea 2
V_ca	D6810	CJ2M	PM10	Float	Tensión de línea 3
Current_a	D6812	CJ2M	PM10	Float	Corriente de línea 1
Current_b	D6814	CJ2M	PM10	Float	Corriente de línea 2
Current_c	D6816	CJ2M	PM10	Float	Corriente de línea 3
Neutral_current	D6836	CJ2M	PM10	Float	Corriente del neutro
Act_pow_a	D6824	CJ2M	PM10	Float	Potencia activa 1
Act_pow_b	D6826	CJ2M	PM10	Float	Potencia activa 2
Act_pow_c	D6828	CJ2M	PM10	Float	Potencia activa 3
Ract_pow_a	D6830	CJ2M	PM10	Float	Potencia reactiva 1
Ract_pow_b	D6832	CJ2M	PM10	Float	Potencia reactiva 2
Ract_pow_c	D6834	CJ2M	PM10	Float	Potencia reactiva 3
App_pow_a	D6818	CJ2M	PM10	Float	Potencia aparente 1
App_pow_b	D6820	CJ2M	PM10	Float	Potencia aparente 2
App_pow_c	D6822	CJ2M	PM10	Float	Potencia aparente 3

Anexo 9 - Referencia de precios

En este último anexo, se detallan los precios de cada componente que se ha nombrado en capítulos anteriores, junto con la entidad que lo ofrece a la venta y la dirección URL en la que se podría adquirir.

Artículo	Precio (€)	Vendedor	Dirección URL
OMRON NB7W-TW01B	1002,79	RS Online	http://uk.rs-online.com/web/p/touch-screen-hmi-displays/8211807/
Siemens SENTRON PAC3200	670,48	RS Online	http://es.rs-online.com/web/p/medidores-digitales-de-potencia/8347556/
Pantalla TFT táctil 7" 800x480px	42,51	AdaFruit	https://www.adafruit.com/product/2354
Placa controladora RA8875 para pantallas táctiles TFT de 40 pines	31,27	AdaFruit	https://www.adafruit.com/product/1590
Arduino MEGA (compatible)	19,95	Garaje Geek	https://garajegeek.com/impresoras-3d/43-arduino-mega-2560-r3-compatible.html?gclid=CjsKDwjw6qnJBRDpoonDwLSeZhIkAIpTR8KyrgaaI5TJ0rzx6p2IDFA5wlEhEgh5Uy9V24ui4pckGgI2IfD_BwE
Transformador encapsulado 400/24VAC	17,25	eBay	http://www.ebay.com/itm/1-pc-Transformer-encapsulated-2-6VA-400VAC-24V-108-3mA-IP00-120g-/172688470776?hash=item283508f2f8:g:tg4AAOSwjDZYf9wD.
Pantalla TFT de 3.5" 480x320px	9,96	DX.com	http://www.dx.com/es/p/3-5-inch-tft-color-screen-module-320-x-480-ultra-hd-for-arduino-uno-r3-mega2560-black-425809?tc=EUR&gclid=CjwKEAjwja_JBRD8idHpxaz0t3wSJAB4rXW5sHNUCBrOctjYTsOh8Waty09u5fYC8Ym8Lo9sQChWshoC66fw_wcB#.WS2vnWjyiUl
Pack de resistencias de diferentes valores	9,52	Electrónica Gimeno	http://www.electronicagimeno.com/store/juego-surtido-de-resistencias-14w-5-61valores-krese12
Caja ABS 200 x 112 x 71 (mm) con aireación PP-60C Negro	8,60	Electrónica Gimeno	http://www.electronicagimeno.com/store/caja-abs-200-x-112-x-71-mm-con-aireaci%C3%B3n-pp-60c-negro

Artículo	Precio (€)	Vendedor	Dirección URL
Arduino Ethernet Shield	7,17	DX.com	http://www.dx.com/p/ethernet-shield-with-wiznet-w5100-ethernet-chip-tf-slot-for-arduino-383470#.WSsjo2jyiUk
Transformador 230/12V	5,20	Todo Electrónica	https://www.todoelectronica.com/transformador-encapsulado-jesiva-220v-1212v-6va-p-69833.html
Arduino LCD Keypad Shield	5,17	DX.com	http://www.dx.com/p/2-6-lcd-keypad-shield-for-arduino-green-black-161359#.WSsjyGjyiUk
Pinza amperimétrica	5,05	Amazon	https://www.amazon.es/SODIAL-Transformador-Corriente-013-030-invasiva/dp/B00H3CTFIQ/ref=sr_1_1?ie=UTF8&qid=1496083410&sr=8-1&keywords=SCT013
Fuente de alimentación AC 100-240V To DC 9V 1000mA	3,58	Banggood	https://www.banggood.com/AC-100-240V-To-DC-9V-1A-1000mA-Power-Supply-Adapter-Charger-EUUS-Plug-p-937975.html?p=HT25115024762014O3SN
25 condensadores de 10 microfaradios	3,51	Todo Electrónica	https://www.todoelectronica.com/condensador-electrolitico-10microf-25v-25-unidades-p-9914.html
40 Cables Macho-Hembra	3,07	Electrónica Gimeno	http://www.electronicagimeno.com/store/cable-de-puente-40-polos-30-cm-macho-hembra-cable-plano
40 Cables Macho-Macho	3,07	Electrónica Gimeno	http://www.electronicagimeno.com/store/cable-de-puente-40-polos-30-cm-macho-macho-cable-plano
Placa de baquelita	2,58	Electrónica Gimeno	http://www.electronicagimeno.com/store/placa-topos-baquelita-ct4-77x90-p254
Comparador LM311	2,13	eBay	http://www.ebay.es/itm/like/201931628911?chn=ps
Poste Hembra 40C P2,54 Recto	1,07	Electrónica Gimeno	http://www.electronicagimeno.com/store/poste-hembra-40c-p254-recto
Zumbador buzzer pasivo	1,00	eBay	http://www.ebay.es/itm/2-x-Zumbador-Buzzer-pasivo-Universal-3-a-12Vdc-Electronica-arduino-prototipos-/281532948832?_trksid=p2141725.m3641.l6368.