ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO ELECTROMECÁNICO

DISEÑO Y CONSTRUCCIÓN DE UN EQUIPO DE MEDIDA DE ENERGÍA ELÉCTRICA DE BAJO COSTE

Autor: David Egido Nieto
Director: Pablo Frías Marín e Ignacio Egido Cortés

Madrid
Junio 2015


PONTIFICIA UNIVERSITAS COMILLENSIS
MATRITI

Proyecto realizado por el alumno/a:

David Egido Nieto

(Poner el nombre del alumno/a)

Fdo.: ..... Fecha: 16 / 06 / 2015

Autorizada la entrega del proyecto cuya información no es de carácter confidencial

NACHO EGIDO CORTÉS

EL DIRECTOR DEL PROYECTO

(poner el nombre del Director del Proyecto)

Fdo.: ..... Fecha: 16 / 06 / 2015

Autorizada la entrega del proyecto cuya información no es de carácter confidencial

EL DIRECTOR DEL PROYECTO

PABLO FRÍAS MARÍN

(poner el nombre del Director del Proyecto)

Fdo.: ..... Fecha: 16 / 6 / 2015

Vº Bº del Coordinador de Proyectos

(poner el nombre del Coordinador de Proyectos)

Fdo.: ..... Fecha: ..... / ..... / .....

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO ELECTROMECÁNICO

DISEÑO Y CONSTRUCCIÓN DE UN EQUIPO DE MEDIDA DE ENERGÍA ELÉCTRICA DE BAJO COSTE

Autor: David Egido Nieto
Director: Pablo Frías Marín e Ignacio Egido Cortés

Madrid
Junio 2015


PONTIFICIA UNIVERSITAS COMILLENSIS
MATRITI

DISEÑO Y CONSTRUCCIÓN DE UN EQUIPO DE MEDIDA DE ENERGÍA ELÉCTRICA DE BAJO COSTE

Autor: Egido Nieto, David.

Directores:

Egido Cortés, Ignacio.

Frías Marín, Pablo.

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

1. Introducción

La monitorización del consumo eléctrica en la industria puede derivar en ahorros económicos importantes. Permite conocer el consumo de reactiva y los excesos de la potencia contratada por la que se cobra en exceso. Además permite ajustar el factor de potencia de las instalaciones y disminuir así el cobro de la potencia reactiva, potencia no útil.

2. Objetivos

El objetivo de este proyecto es diseñar un equipo de medida que pueda ser usado para obtener un registro en el tiempo del consumo en el edificio del ICAI. Para ello es preciso que el valor de los parámetros sean lo suficientemente fiables.

En la tabla 1 se muestran las especificaciones objetivo, fijadas y cumplidas para este trabajo.

Características		
Rango de tensiones admisibles en medida directa		UN= 3x220/380V
Tipos de medida	Tensión	Directa e indirecta
	Intensidad	Indirecta $I_{secundario}$=5A
Periodo de actualización de medidas		[1,10]s
Intensidad nominal		5A

Almacenamiento de datos			Sí
Tipo de almacenamiento			USB
Precisión (objetivo)	Intensidad		<10%
	Tensión		<10%
	Potencias		<10%
	Factor de potencia		<10%
	Energía activa		<10%
	Energía reactiva		No
Alimentación	AC		No
	DC		batería 9V DC
Método de medida potencia activa	Un vatímetro	4 hilos	
		3 hilos	
	Tres vatímetros	4 hilos	×
		3 hilos	
	Dos vatímetros	4 hilos (Dual)	
		3 hilos (Aron)	×
Mediciones monofásicas			Sí

Tabla 1: resumen de objetivos fijados para el TFG

3. Estado del arte

En el estado del arte se analizan dispositivos disponibles en el mercado desde los aparatos más sencillos, como los contadores de energía, a los más sofisticados como los analizadores de red.

Como referencia de precisiones se han tomado varios equipos comerciales para realizar la comparación, especialmente el DIRIS A40 [1], y un proyecto de código libre llamado Open Energy Monitor, que explica cómo llevar a cabo un medidor de energía barato aunque con precisiones dudosas.

También se han analizado todos los microcontroladores Arduino [2] con sus características principales y se ha seleccionado el que mejor cumpliría los objetivos fijados a sus funciones.

4. Arquitectura del sistema

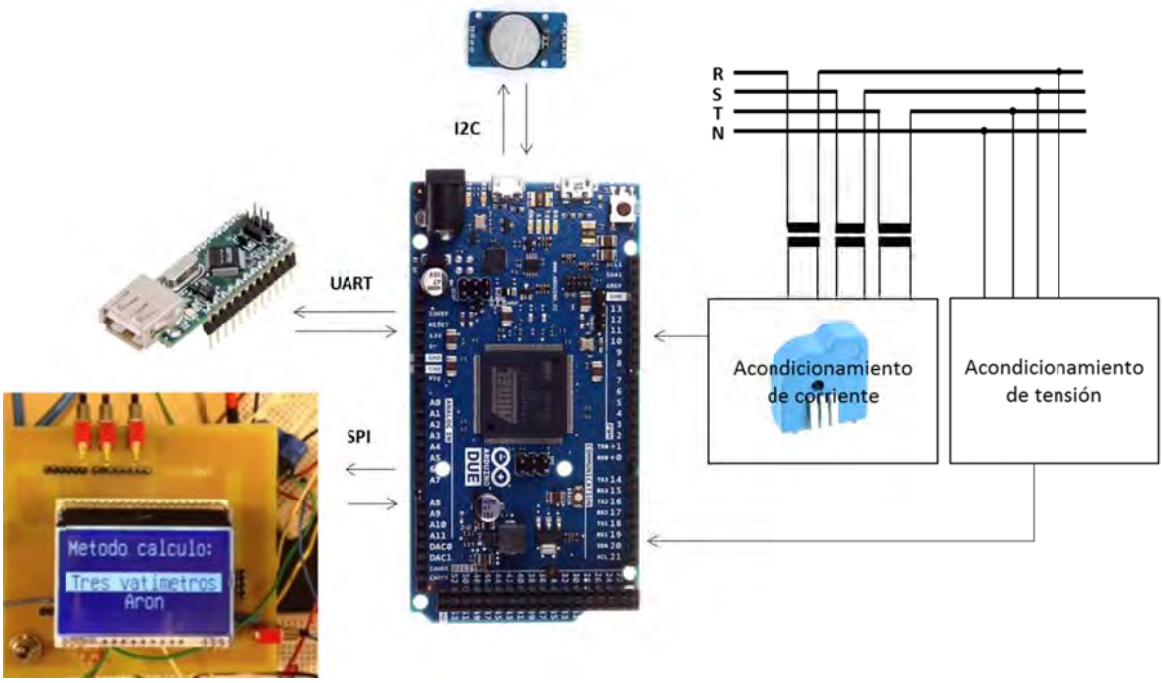


Figura 1: Esquema del equipo de medida

El esquema del proyecto se muestra en la figura 1 y se puede apreciar las características principales buscadas y alcanzas en este trabajo.

Por un lado, la parte de la interfaz del usuario está formada por el LCD y los pulsadores que permiten elegir el método de cálculo deseado y el puerto USB que permite almacenar los parámetros calculados en una memoria USB con formato FAT 32.

Otra de las características era la incorporación de un reloj no volátil, característica tomada de todos los registradores de energía analizados.

El microcontrolador, como se puede apreciar en la figura 1, se ha elegido el Arduino Due. Éste es el que mayores prestaciones ofrecía para la consecución de los objetivos. Es el encargado de procesar los datos obtenidos de las señales de corriente y tensión, así como, de comunicarse con los elementos que aparecen en la figura 1. En dicha figura también se aprecian los tipos de comunicación empleados con los integrados que forman la medida de tiempo y la interfaz de usuario.

Y, por último, el acondicionamiento de señales cuyas características principales se enumeran a continuación:

- Entradas independientes de tensión e intensidad, ofreciendo versatilidad en las conexiones.
- Posibilidad de medida directa en entradas de tensión y de medida indirecta a través de transformadores. Esto permite al equipo ser empleado en todo tipo de redes tanto de baja como de alta tensión.
- Medición indirecta de corriente con entrada nominal de 5A y entrada máxima admisible de 6A.
- Aislamiento inductivo en acondicionamiento de corriente por medio de transductores de corriente.
- Aislamiento óptico en acondicionamiento de tensión a través de conversión de dicha magnitud a frecuencia.
- Uso de referencia flotante independiente de la red y del Arduino respecto de la cual acondicionar las señales de tensión.

5. Resultados

La viabilidad del proyecto está justificada en base a su bajo coste en relación con su alta precisión, inferior al 1% en medidas de corriente, tensión y en otros parámetros como potencia activa para factores de potencia cercanos a la unidad. El coste total de proyecto se ha estimado en unos 192€.

Entre las mejoras introducidas respecto al proyecto de Open Energy Monitor, cabe destacar la adición de un filtro paso banda con compensación de fase para frecuencia de 50Hz y que mejora la respuesta ante armónicos de altas frecuencias. Otra mejora introducida es la utilización de métodos diferentes en el acondicionamiento de las señales y que tiene una importante mejora en la precisión.

6. Conclusiones

Se concluye haciendo énfasis en la consecución de un diseño de un equipo de medida cuyas características son muy cercanas a la de equipos comerciales. Además la precisión del aparato es superior a la del proyecto de Open Energy Monitor donde los mejores valores de precisión están entorno al 2% para un rango menos amplio de potencia respecto al de este dispositivo. El dispositivo diseñado en este TFG mantiene su precisión en todos los parámetros en más de un 75% del rango de tensión, más concretamente, desde el 25% de la tensión nominal, hasta un 110% de la tensión nominal.

7. Referencias

[1] SOCOMEC, «http://www.socomec.es/gama-analizadores-de-red-monopunto_es.html?product=/diris-a40_es.html,» [En línea].

[2] Arduino, «http://www.arduino.cc/,» [En línea].

[3] Open Energy Monitor, [En línea]. Available: http://openenergymonitor.org/emon/.


UNIVERSIDAD PONTIFICIA
ICAI ICADE
COMILLAS
M A D R I D

DESIGN AND MANUFACTURE OF A LOW COST EQUIPMENT OF ELECTRICAL ENERGY MEASUREMENT

1. Introduction

The monitoring of the electrical consumption in the industry might involve an economic saving in the invoice of electricity. It allows knowing the reactive power consumption and the excess of active power regarding the contracted power, excess which carries a strong increase in the invoice paid. Furthermore, it enables companies to adjust the power factor of their electrical installation so their reactive power charges can be reduced.

2. Objectives

The principal objective is to design an equipment of electrical energy measurement to be used in ICAI, taking a register of the long-term consumption. It is on that purpose that the device must deliver reliable and accurate quality measurement of the main parameters.

In the table 1 are showed the target specifications, fixed and fulfilled, for this Final Degree Project.

Characteristic		
Permitted voltage range in direct measuring		UN= 3x220/380V
Types of measurement	Voltage	Direct and indirect
	Current	Indirect $I_{secondary}$=5A
Period of update of the parameters		[1,10]s
Nominal current		5A
Storage of data		Yes
Type of storage		USB
Accuracy target	Current	<10%
	Voltage	<10%
	Power	<10%
	Power factor	<10%
	Active energy	<10%
	Reactive energy	No
Supply	AC	No
	DC	9V DC battery

Method of measuring active power	One wattmeter	4 wires	
		3 wires	
	Three wattmeter	4 wires	×
		3 wires	
	Two wattmeter	4 wires (Dual)	
		3 wires (Aron)	×
Single face measures			Yes

Table 1: Fixed objectives for the Final Degree Project

3. State of the art

In the current section it is analyzed a wide range of available devices, from the simplest Smart meters to the very sophisticated network analyzers.

As a reference of accuracy it has taken different commercial products in order to compare with the current design, especially the DIRIS A40 [1], and a project of a free and open source software called Open Energy Monitor. The last one explains how to carry out an inexpensive electric meter but with very doubtful accuracy.

It has been analyzed each microcontroller board of Arduino from the official website [2] with their main characteristic selecting the suitable one to fulfilled all the fixed objectives with regard to its functions.

4. System architecture

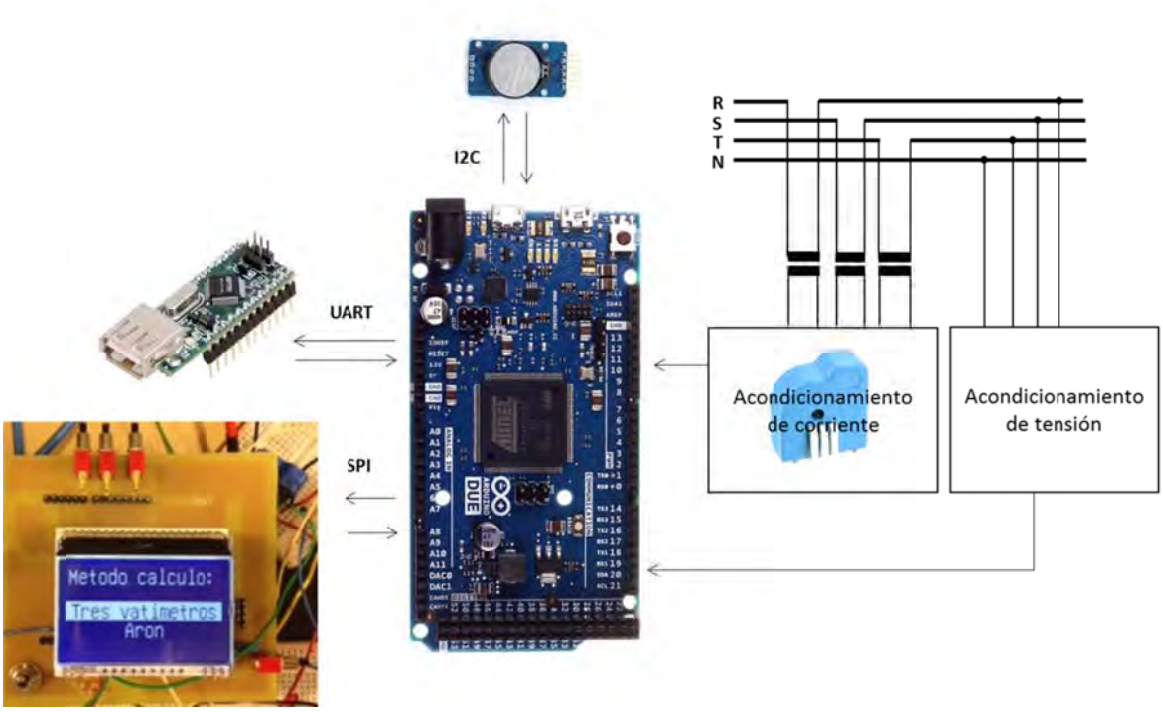


Figure 1: Simple schematic of the complete system

The architecture of this work is showed in the figure 1 and it could be noticed the main functional units developed.

On the one hand, the user interface consist of the Liquid Crystal Display and the push buttons to select the desirable method of calculations, and the USB port that enable to fill a USB stick, only formatted in FAT32, with the calculated parameters.

Otra de las características era la incorporación de un reloj no volátil, característica tomada de todos los registradores de energía analizados.

The chosen microcontroller, as it can be noticed in the figure 1, was the Arduino Due. It is the one with the most suitable performances for this Final Degree Project in order to achieve the objectives. It is in charge of process data acquired from the voltage and current signals, and also, to stablish communication with other peripherals that appeared in figure 1. In this figure it can also be noticed the types of communication stablished with peripherals.

And, to finish, the characteristics of the signal conditioning are listed below:

- Independent entrances of voltage and current, providing versatility in connections.

- Possibility of direct measures and indirect measure via voltage transformers in voltage entrances. This enables the equipment to be connected in low and high voltage systems.
- Indirect measurement of the current signal with 5A and 6A of nominal and maximum permissible current respectively.
- Inductive insulation in current conditioning via current transducer.
- Optical insulation in voltage conditioning via voltage to frequency conversion.
- Use of a floating ground independent from the electrical system and from the reference of the Arduino.

5. Results

The viability of the Project is justified based on its low cost with regards to the high accuracy achieved, lesser tan 1% in current, voltage and other parameter calculations, but only in case where power factor is close to the unity. The total cost of the project has been estimated in 192€.

Among the improves regarding Open Energy Monitor Project, it should be noted the addition of a band-pass filter which phase compensation leaves phase lag or lead equal to the real phase shift. This filter improve the temporal response to high frequency harmonics but only Works well with 50Hz. Another improve is the addition of a completely different method of signal conditionings resulting in very accurate measures.

6. Conclusions

It should be emphasized on the results obtained from this design achieving accurate measures, very close to commercial devices. There is also an improvement on the accurate with regard to the Open Energy Monitor Project where the power accuracy is about 2% in a smaller range of power values. In the current prototype designed, the accuracy achieved in power measurement is lower than 1% in a range of voltage from the 25% to a 110% of the nominal voltage.

7. References

[1] SOCOMEC, «http://www.socomec.es/gama-analizadores-de-red-monopunto_es.html?product=/diris-a40_es.html,» [En línea].

[2] Arduino, «http://www.arduino.cc/,» [En línea].

[3] Open Energy Monitor, [En línea]. Available: http://openenergymonitor.org/emon/.

DEDICATORIA

Aprovecho esta página para agradecer de corazón su apoyo a aquellas personas de mi entorno cercano.

En especial a mi familia por su apoyo incondicional.

A mis directores de proyecto, Pablo Frías Marín e Ignacio Egido Cortés por guiarme para afrontar todos los innumerables problemas surgidos.

Agradezco de forma sincera a Ignacio su paciencia y disponibilidad continua durante todo el periodo de realización del TFG.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

Parte I MEMORIA

Capítulo 1 INTRODUCCIÓN

Optimizar la potencia contratada en función del consumo puede derivar en ahorros económicos de gran trascendencia, principalmente para los consumos industriales. La monitorización de la energía y su medida puede ser determinante para este objetivo.

Esta monitorización se puede llevar desde el consumo más pequeño hasta la propia generación pasando por el transporte y la distribución de energía eléctrica y conocer la eficiencia del sistema.

Los aparatos capaces de medir la potencia consumida, además de otros parámetros eléctricos, son los analizadores de red. Por otro lado, permite a las compañías distribuidoras una gestión remota de la medida de la energía eléctrica.

El contexto en el que se introducirá el presente PFG es el de medida de la potencia consumida por el edificio de ICAI. Este edificio dispone de su propio centro de transformación donde se tomarán todas las medidas. Este dispositivo podrá ser empleado para optimizar la potencia a contratar con la compañía distribuidora correspondiente por parte de esta escuela de ingeniería y conocer el consumo de energía reactiva.

1 Estudio de los trabajos existentes / tecnologías existentes

Este proyecto de final de grado, en adelante PFG, se enmarca dentro del área de medida y control de los circuitos eléctricos y su objetivo principal es diseñar y construir un aparato de medida eléctrica trifásica de baja tensión.

Como punto de partida del proyecto se mostrará la tecnología presente en este mercado y que modelo se ajusta más al prototipo que se pretende.

El presente proyecto se centrará en los medidores digitales de redes trifásicas. En el mercado podemos encontrar una gran variedad de ofertas, desde aparatos de bajo coste como los contadores inteligentes (Smart Meter), a otros más sofisticados como el analizador de red multifunciones, PM850MG de Schneider.

A continuación se presentan varios modelos y sus características, situando este PFG con referencia a estos modelos.

			Aparato				
Característica			ZMXe310CR	DIRIS A40	PM850MG	N10 / N10A	ZMD310CTSCD
Rango de tensiones admisibles en medida directa			UN=3x127/220V ó 3x230/400V	50-700 V AC fase-fase/28-404 V AC fase-neutro	0-600 V AC fase-fase/ 0-347 V AC fase-neutro	100V ó 400V	58-240 V AC fase-neutro/100-415 V AC fase-fase
Tensión nominal en medida a través de transformador (indirecta)		Primario	No	500kV	0-3,2 MV AC	0-400kV	No
		Secundario		60, 100, 110, 115, 120, 173 y 190 V AC	0-347 V AC	100V ó 400V	
Periodo de actualización de medidas			1s	1s	1s	ND	ND
Intensidades nominales			Transformador interno 80A/5A	1, 5 A	Transformador interno 5 A-32.700 A/1 ó 5 A	Transformador interno 0 A-20.000 A/1 ó 5 A	Conexión directa: 10(Imax=80A) A
Consumo de alimentación			<15 VA	< 10VA	15VA	≤ 12 VA	≈ 4,5VA
Almacenamiento de datos			Sí	Opcional	Sí	No	Sí
Precisión	Intensidad		ND	0,5 % de 10 a 110 % de In	0.325 % de 1 A a 10 A	± (0.2% Valor medido + 0.1% del rango)	ND
	Tensión		ND	0,5 % de 140 a 700 V AC	0.375 % de 50 V a 277 V	± (0.2% Valor medido + 0.1% del rango)	ND
	Potencias		ND	1 % de la plena escala (-90° a + 90°)	0.2 %	± (0.5% Valor medido + 0.2% del rango)	ND
	Factor de potencia		ND	1 % para 0,5 < FP < 1	0,1 % desde 1A a 10A	± 1% Valor medido ± 2 cuentas	ND
	Energía activa		Clase B (EN 50470-3)	Clase 1 (IEC 62053-21)	Clase 0.5 S (IEC 62053-22)	± (0.5% Valor medido + 0.2% del rango)	Clase 1 (IEC 62053-22)
	Energía reactiva		Clase 2 (EN 62052-23)	Clase 1 (IEC 62053-23)	Clase 2 (IEC 62053-23)	± (0.5% Valor medido + 0.2% del rango)	Clase 2 (IEC 62053-23)
	Frecuencia		ND	0,1% de 45Hz a 65Hz	± 0,02 % de 47 a 67 Hz	± 0.5% Valor medido de 15.0-500.0 Hz	ND
Alimentación	AC		Desde la propia red donde mide	110 a 440 V AC en 50/60 Hz ± 10 %	100 a 415 V AC	85 a 250 V AC	Desde la propia red donde mide
	DC		No	120 a 350 V DC ± 20 %	125 a 250 V DC	60 a 177 V DC	40-140V DC (opcional)
Método de medida potencia activa	Un vatímetro	4 hilos		×	×	×	
		3 hilos		×	×	×	
	Tres vatímetros	4 hilos	×	×	×	×	×
		3 hilos		×	×	×	
	Dos vatímetros	4 hilos (Dual)		×	×	×	
		3 hilos (Aron)		×	×	×	
Mediciones monofásicas			No	Sí	Sí	Sí	Sí

Tabla 1: Características destacables de varios modelos de analizadores de redes eléctricas

Todos ellos admiten un gran rango de medidas de tensión. En el presente PFG se analizará cual es el valor mínimo de tensión para el cual el aparato mantiene un nivel de precisión en el cálculo de los parámetros aceptable. El nivel máximo de tensión será la tensión máxima que se pueda presentar en este punto de la red y para la cual se fijará el alcance. Ese alcance será de 380V para redes de baja tensión de ese voltaje entre fases.

Todos estos modelos permiten medir las tensiones sin necesidad de emplear transformadores de tensión que permitan aislar la red del equipo de medida. En el modelo a diseñar se incorporará un método de medida directa de igual forma que los aparatos analizados y que ofrezca aislamiento galvánico sin necesidad de emplear transformadores. Además, la incorporación del método de medida directa conlleva la posibilidad de la medida indirecta a través de transformador. Por tanto, el aparato a diseñar incorporará ambos.

También se busca que el consumo sea el menor posible, al ser un medidor de energía, este requiere una larga autonomía.

La mayoría de productos analizados disponen de una gran flexibilidad a la hora de conectarlos a la red de la cual se quiere obtener los distintos parámetros. En función de las distintas conexiones, estos aparatos realizan diferentes métodos de cálculo en base a los datos que se obtienen de dicha red. Por ejemplo, en caso de conexionado en red de tres hilos, uno de los métodos de cálculo empleados por sus softwares es el de Aron por el cual no necesitan la referencia del neutro.

En base a las características analizadas en los aparatos anteriores dispuestas en la tabla 1 se extraen las características *objetivo* de este PFG. Estas especificaciones se muestran a continuación en la tabla 2.

Características			
Rango de tensiones admisibles en medida directa			UN= 3x220/380V
Tipos de medida		Tensión	Directa e indirecta
		Intensidad	Indirecta $I_{secundario}$=5A
Periodo de actualización de medidas			>1s
Intensidades nominales			5A
Consumo de alimentación			ND
Almacenamiento de datos			Sí
Tipo de almacenamiento			USB
Precisión (Desconocida)	Intensidad		<10%
	Tensión		<10%
	Potencias		<10%
	Factor de potencia		<10%
	Energía activa		<10%
	Energía reactiva		No
Alimentación	AC		No
	DC		batería 9V DC
Método de medida potencia activa	Un vatímetro	4 hilos	
		3 hilos	
	Tres vatímetros	4 hilos	✕
		3 hilos	
	Dos vatímetros	4 hilos (Dual)	
		3 hilos (Aron)	✕
Mediciones monofásicas			Sí

Tabla 2 Características principales del dispositivo de este PFG

El dispositivo, dispondrá de una memoria permanente para almacenar todos aquellos parámetros que se pueda, sin que estos datos se pierdan una vez que se quite la alimentación del dispositivo.

Se incorporará una característica que poseen todos los aparatos analizados con almacenamiento de datos, esto es, un reloj no volátil. Este reloj evitará su desconfiguración ante una pérdida de alimentación.

La alimentación se obtendrá mediante unas baterías de corriente continua. Será necesario hacer una estimación del consumo del dispositivo, de manera que proporcione una autonomía suficiente y conocer el periodo de recarga necesario.

Una posible conexión del dispositivo a la red a monitorear se presenta en la figura 1 a continuación, extraída de la hoja técnica del DIRIS A40.

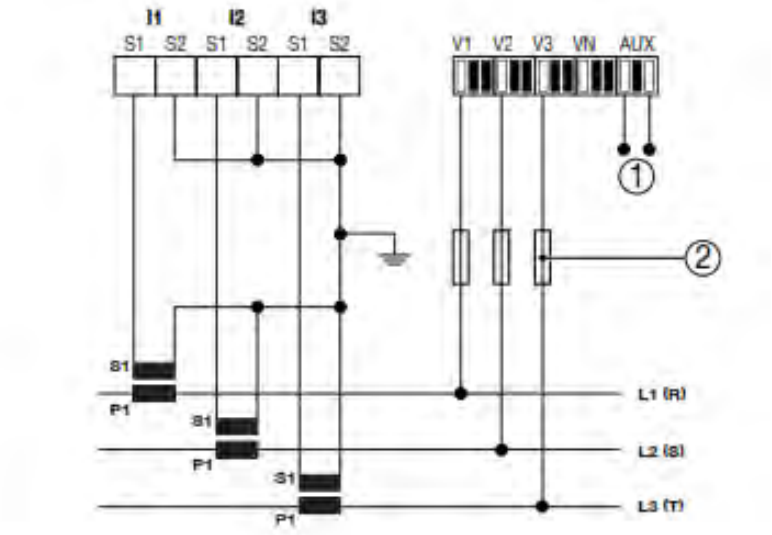


Figura 1: Conexionado del DIRIS A40 en red trifásica equilibrada o desequilibrada de cuatro hilos

En la figura 1 se representa uno de los distintos métodos de conexión de los que dispones el DIRIS A40. Esta conexión es para una red sin neutro accesible y con medida directa de tensión e indirecta de corriente. Este método de conexión se implementará en este trabajo, además junto a esta conexión se realizará el cálculo mediante el método de Aron.

El cálculo de los valores de potencia se realizará de dos formas distintas en función de la disposición de la red donde se conectará el aparato. Estas dos formas de conexión son las siguientes:

- Red de tres hilos: debido a la inexistencia del cable de neutro será necesario emplear el método de cálculo de Aron.
- Red de cuatro hilos: el cálculo será el del método de los tres vatímetros.

Esta forma de cálculo será seleccionable por parte del usuario por medio de unos botones y una pantalla.

Los dispositivos DIRIS A40, PM850MG y N10 / N10A analizados tienen sus entrada de tensión y corriente independientes entre sí. Esta es una de las características más importantes de un medidor de energía, y por ello, se incorporará al presente trabajo. Las entradas independiente de tensión y corriente permiten una gran versatilidad a la hora de conexionar, es decir, permite medir tensión y corrientes incluso de circuitos distintos.

Al igual que todos los productos analizados, se implementará una pantalla para poder visualizar los datos calculados, siendo estos de gran utilidad durante el desarrollo del aparato. Además de mostrar los distintos parámetros, solicitará al usuario la conexión realizada, es decir, si la conexión es a tres hilos o a cuatro hilos.

Para el método de medida directa se diseñará un circuito de acondicionamiento de las señales de tensión que ofrezcan un aislamiento galvánico. Para el circuito de acondicionamiento de corriente también se incorporará aislamiento galvánico que asegure el microcontrolador en caso de contingencia.

En este PFG se utilizará un microcontrolador de bajo coste, un Arduino. Para poder seleccionar aquel Arduino cuyas prestaciones sean suficientes para la consecución de los objetivos se hace un análisis de los distintos modelos disponibles. Se parte de la consideración de que el Arduino a seleccionar ha de ser lo suficientemente rápido para que el procesamiento de todos los cálculos se haga en el menor tiempo posible, esto es crítico debido a que se pretende tener una frecuencia de muestreo lo suficientemente elevada.

Los Arduinos disponibles en el mercado junto con sus características se presentan a continuación:

Modelo	Tensión de operación	Tensión de alimentación	Memoria Flash [Kb]	Memoria RAM estática [Kb]	Entradas/Salidas digitales/PWM	Entradas /Salidas analógicas	Resolución Entradas /Salidas analógicas	Velocidad de reloj	Microprocesador
Uno	5V	7-12V	32	2	14/6	6/0	10bits/0	16MHz	ATmega 328
Pro	3.3V (versión 1) / 5V (versión 2)	3.35 -12 V (versión 1) / 5-12V (versión 2)	16/32	1/2	14/6	6/0	10bits/0	8 MHz (Versión3.3V) / 16 MHz (Versión 5V)	ATmega 168/ ATmega 328
Nano	5 V	7-12 V	16/32	1/2	14/6	8/0	10bits/0	16MHz	ATmega 168/ ATmega 328
Pro Mini	3.3V (versión 1) / 5V (versión 2)	3.35 -12 V (versión 1) / 5-12V (versión 2)	16	1	14/6	8/0	10bits/0	8 MHz (Versión3.3V) / 16 MHz (Versión 5V)	ATmega 168
Leonardo	5V	7-12V	32	2,5	20/7	12/0	10bits/0	16MHz	ATmega32u4
Micro	5V	7-12V	32	2,5	20/7	12/0	10bits/0	16MHz	ATmega32u4
Mega	5V	7-12V	128	8	54/15	16/0	10bits/0	16MHz	ATmega 1280
Mega ADK	5V	7-12V	256	8	54/15	16/0	10bits/0	16MHz	ATmega2560
Due	3.3V	7-12V	512	96	54/12	12/2	12bits/ 12bits	84 MHz	Atmel SAM3X8E ARM Cortex-M3 CPU
Ethernet	5V	7-12V	32	2	14/4	6/0	10bits/0	16MHz	ATmega328
Esplora	5V	5V	32	2.5	N/A	N/A	N/A	16MHz	ATmega32u4

Tabla 3: Modelos de Arduino y sus características principales

El tiempo que los conversores analógico digitales de los Arduinos necesitan para leer las entradas analógicas son las siguientes:

- Arduino Due con CPU AT91SAM3X8E es de 1MHz.
- Arduinos con los distintos CPU´s de ATmega está en torno a 3.8kHz y 77kHz.

Esta frecuencia de muestreo se verá reducida debido a la complejidad del programa a implementar y al número de entradas analógicas muestreadas en cada ciclo de muestreo.

Como se puede ver en la tabla el Arduino Due es el que mejor precisión ofrece en la conversión de las señales analógicas, posee una resolución de su ADC de 12bits a diferencia de los 10 bits de los demás. Se tiene un error de cuantización, suponiendo que la tensión de referencia Vcc=3.3V, de aproximadamente un 0.01%.

Dado que para la medida de tensión se hace una conversión de esta a frecuencia, es necesario conocer cómo se puede medir frecuencia con el Arduino. Uno de los requerimientos que aunque no es imprescindible, es bastante conveniente, es el de que disponga de tres contadores independiente para llevar a cabo la medida de frecuencia. El hecho de que sean tres y no menos, permite hacer el muestreo de las tres tensiones de fase, en caso de red de cuatro hilos, al mismo tiempo. Esto es una gran ventaja dado que asegura que las medidas de tensión corresponden al mismo instante. Esto no sucede así con las medidas de corriente dado que el muestreo se lleva a cabo con el ADC del Arduino que muestrea una tras otra.

El Arduino Due es el de mayores prestaciones para este PFG, tiene el procesador más rápido de todos en cuanto a muestreo y procesamiento de datos, además tiene un alto número de entradas analógicas.

Es necesario advertir que el Arduino Due no dispone de memoria EEPROM que aunque los otros Arduinos sí disponen de ella, su capacidad es muy limitada para el almacenaje de datos. En cualquier caso, aún con la elección de un Arduino distinto, la necesidad de una mayor capacidad de memoria hace imprescindible la implementación de una memoria externa. La solución adoptada es la comentada anteriormente, la implementación de un puerto USB para introducir lápices de memorias que permite un almacenamiento de datos de hasta varios GBytes.

En cuanto al procesamiento de los datos, es decir, la parte del software, se acudirá a un proyecto de código libre llamado Open Energy Monitor [1]. En cuanto a su diseño del hardware, se considerará aunque se empleará unos métodos totalmente distintos a los empleados en Open Energy Monitor. Esta diferencia se debe a las expectativas de mejora de dicho proyecto de código libre.

Para el diseño del código se partirá de una estructura parecida a la empleada en su sistema, sin embargo, se adaptará a las características de este trabajo y en muchos casos se modificará.

En las figuras 2 y 3 se muestra el método empleado por Open Energy Monitor para acondicionar las señales de tensión e intensidad.

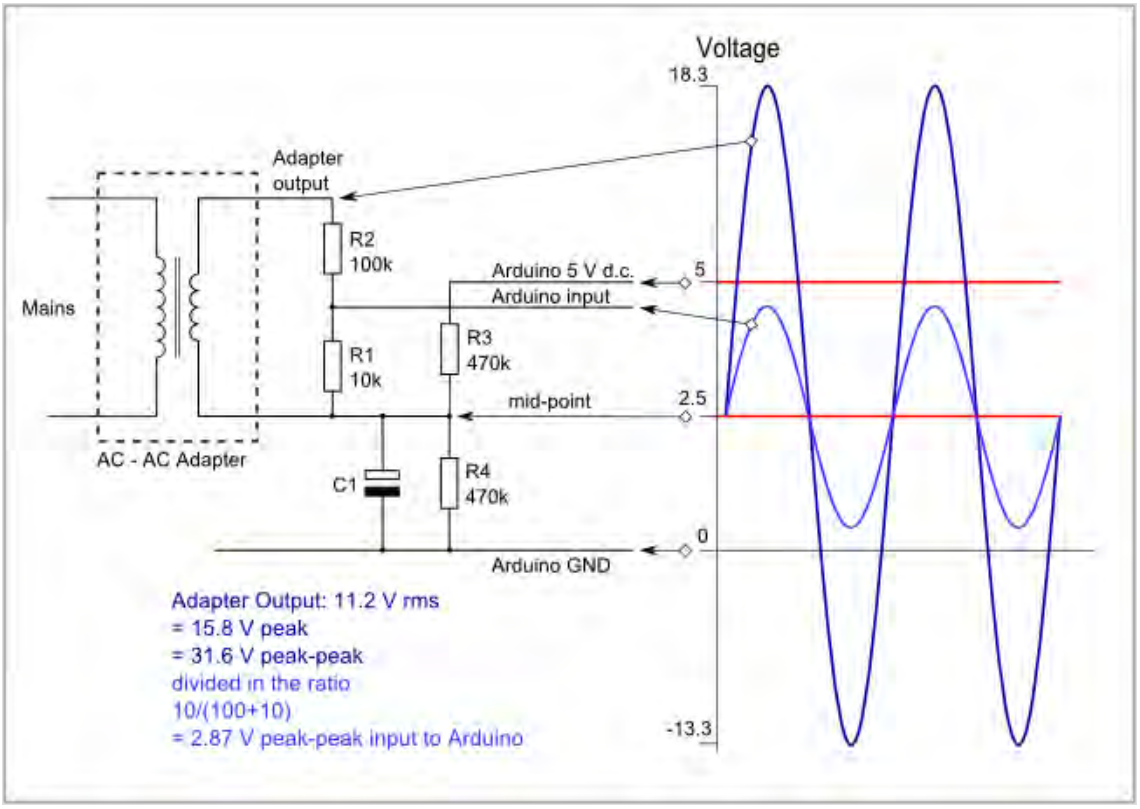


Figura 2: Acondicionamiento de la señal de tensión mediante transformador de tensión

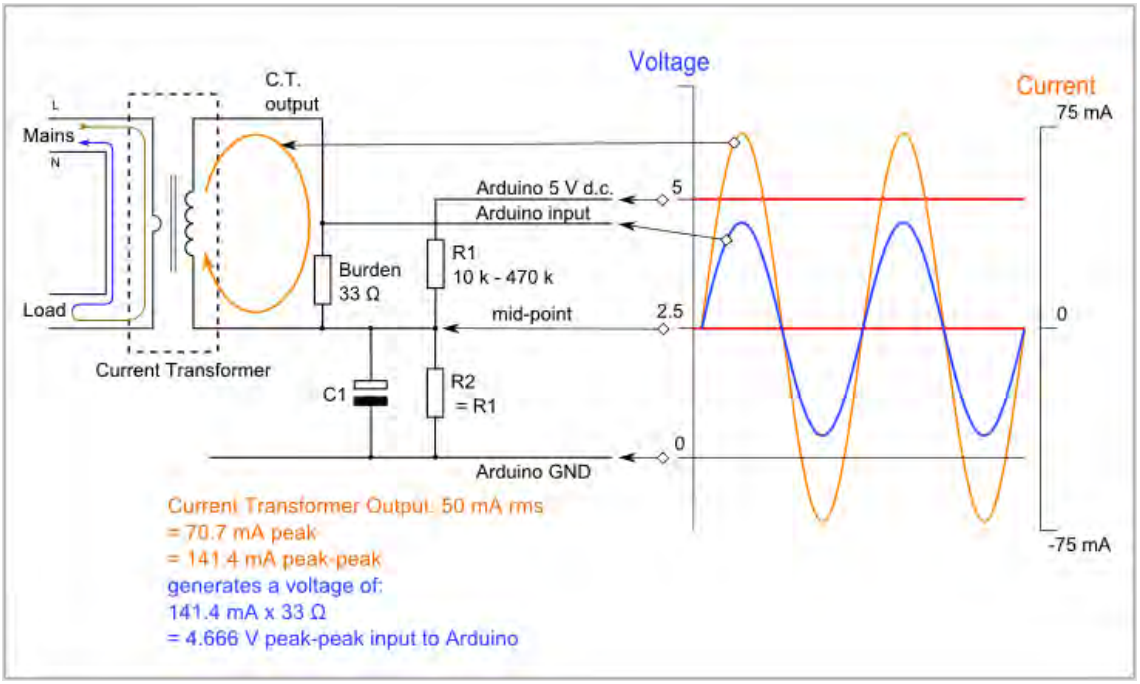


Figura 3: Acondicionamiento de la señal de corriente mediante transformador de corriente

Tras el análisis de las posibles soluciones actuales de equipos similares y junto a los distintos modelos de Arduino, se extraen los objetivos que han de alcanzarse para el presente trabajo.

2 Motivación del proyecto

El objetivo de este proyecto es el de construir un medidor de energía bajo coste para poder medir consumos en baja tensión de 220V/380V. Este aparato es de gran utilidad para consumos industriales. En este tipo de consumos no existe un interruptor de control de potencia, sino que, disponen de un maxímetro que mide los excesos de potencia consumida respecto a la contratada. Estos excesos se penalizan con cobros muy superiores a la consumida por debajo de dicho umbral. Este es el caso del edificio de ICAI. Además, también es muy útil conocer el consumo de energía reactiva, energía no útil, por la que la compañía eléctrica cobra al consumidor.

Este proyecto fin de grado tiene por finalidad construir un dispositivo para permitir en un futuro el uso de este para analizar el consumo en dicho edificio, pudiendo definir una posible potencia a contratar cuya factura pueda compensar los excesos o, por el contrario, sea de interés evitarlos y conocer el consumo de energía reactiva.

3 Objetivos

El presente trabajo pretende diseñar e implementar un medidor de parámetros eléctricos, para ello, se empleará un microprocesador Arduino junto a los distintos recursos de acondicionamientos de señales a las entradas del Arduino, buscando siempre el método menos costoso.

Se pretenden resolver problemas como:

- Interfaz de usuario para elección del método de conexionado.
- Entradas de corriente y tensión independientes entre sí para ofrecer versatilidad de conexión.
- Se decidirá los parámetros eléctricos a medir en función del número de entradas, la velocidad de muestreo del Arduino y tiempo de cálculo de parámetros.
- Elección de los elementos que permiten acondicionar las magnitudes medidas a las entradas del microprocesador.
- Selección del Arduino más conveniente.
- Diseño del código a implementar en el microprocesador.
- Continuación en la medida de fecha y tiempo cuando la alimentación, evitando así, la reconfiguración del reloj incorporado.
- Visualización de parámetros mediante pantalla.
- Necesidad de emplear memoria externa por limitación por parte del microcontrolador.
- Compromiso entre tiempo de muestreo de medidas y cálculo de parámetros, almacenamiento de datos y mostrar datos en pantalla.
- Búsqueda de la mejor precisión en las medidas optimizando el presupuesto.

4 Metodología / Solución desarrollada

La metodología a seguir será la del diseño paralelo de hardware y software de forma separada. También, se irán diseñando de forma individual de cada una de las unidades funcionales del PFG, como por ejemplo la implementación del reloj de tiempo real o del puerto USB. Este método de trabajo permite detectar las causas de error a medida que se van incorporando unidades funcionales y consiste en solventar errores individuales para prevenir el fallo en la implementación conjunta de estas partes.

La elección de los elementos a implementar en el PFG se eligen tras un concienzudo examen de todos los componentes disponibles y de sus ventajas e inconvenientes. Para ello será necesario proveerse de los conocimientos necesarios para tomar las decisiones adecuadas.

Se buscará en todo momento aquellos elementos que permitan alcanzar los objetivos fijados de manera óptima. Lo primordial en la elección de los elementos será el precio de estos, puesto que se busca la construcción de un analizar de red de bajo coste.

En cuanto al bloque de programación del Arduino se empleará el propio Software libre de Arduino junto con sus librerías. Se buscará en todo momento programar por bloques subdividiendo el programa siguiendo el método de trabajo descrito.

5 Recursos / herramientas empleadas

Las siguientes herramientas serán usadas para la consecución de los objetivos del trabajo:

- Placa con microcontrolador de Arduino.
- Arduino IDE (Integrated Development Environment), es el entorno de programación que proporciona Arduino para programación de la placa.
- Fritzing, se trata de un programa gratuito que permite crear esquemas de circuitos impresos con Arduino. Es una herramienta de diseño electrónico para la creación de prototipos.
- Conversores de tensión a frecuencia y optoacopladores.
- Transductores de intensidad de núcleo abierto.
- Resistencias y condensadores.
- Pantalla para visualizar los datos.
- Botones para visualizar los distintos parámetros y para inicialización del sistema.
- Baterías de corriente continua de 9V y 12V.
- Aparatos disponibles en el laboratorio de medidas y máquinas eléctricas.
- Microsoft office.
- Matlab.

Capítulo 2 DISEÑO Y FABRICACIÓN DEL ANALIZADOR DE RED

1 Descripción global

Mediante el mismo método de trabajo, por medio del cual se ha desarrollado este PFG, se irán explicando cada una de las distintas unidades funcionales que completan este analizador de red. Se puede dividir en las siguientes unidades funcionales:

- Acondicionamiento de las señales de tensión
- Acondicionamiento de las señales de intensidad.
- Medida de tiempo y temperatura mediante reloj externo.
- Almacenamiento de datos.
- Visualización de datos e interfaz.
- Procesamiento de datos y muestreo de señales. Esta parte es la que concierne al microcontrolador y por tanto, al software del PFG.

Dentro de cada uno de estos bloques se podrán encontrar la justificación de cada unidad funcional, la justificación de los componentes elegidos para realizar dicha función y de sus ventajas e inconvenientes.

2 Acondicionamiento de señales de tensión

El acondicionamiento de las señales se refiere a todo método implementado capaz de compatibilizar las señales de corriente y tensión con las formas de medida implementadas en el microcontrolador.

Para acondicionar las señales se buscará en todo momento proporcionar aislamiento galvánico que de seguridad a la parte de la interfaz de usuario, así como del microcontrolador y demás circuitería.

2.1 Alternativas

Durante la consecución del semestre se analizaron distintas posibilidades para alcanzar los objetivos definidos en la medida de tensión. Todas las alternativas fueron analizadas a conciencia y eliminadas en base al valor añadido que estas pudieran dar al PFG y a los medios de los que se disponían. Las distintas alternativas se barajaron en función del tipo de aislamiento galvánico que interesaba introducir y se detallan a continuación:

2.1.1 Aislamiento inductivo

El aislamiento galvánico inductivo es aquel que se consigue mediante el fenómeno de inducción electromagnética y permite que la corriente no fluya directamente entre circuitos. Para los métodos de medición de tensión se emplean trasformadores de tensión de alta precisión, figura 4.

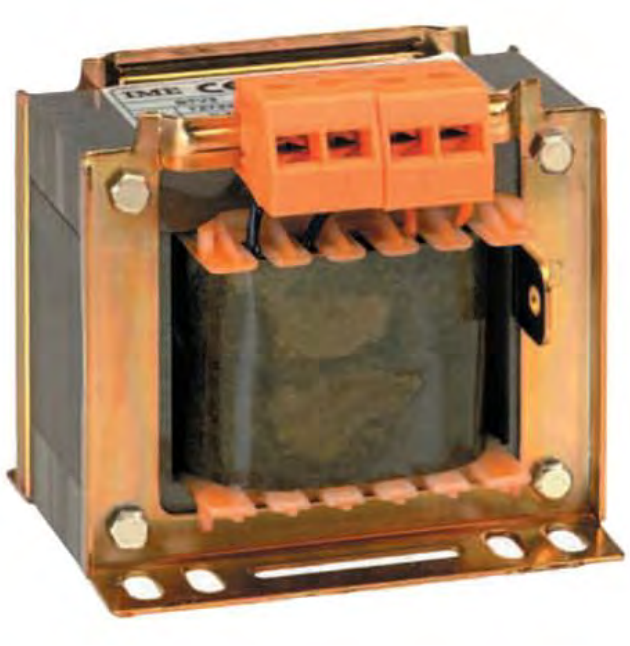


Figura 4: transformador de núcleo ferromagnético monofásico de baja tensión de clase 1 de IME

Una de las alternativas descartadas fue la de medida indirecta de tensión por medio de transformador de precisión de tensión. Este método es, sin duda, el más empleado por su sencillez, sin embargo, tiene el inconveniente de que las medidas están afectadas por un error elevado introducido por el transformador de tensión. Este error, tanto de módulo como de fase, es mayor cuanto menor es el precio del transformador.

Uno de los objetivos de este proyecto era medir tanto tensión simples como compuestas empleando el mismo circuito para ambas. El hecho de que los transformadores de precisión de tensión solo aseguran su precisión hasta el 80% de la nominal no nos permitiría emplear un transformador para reducir los 380V y los 220V que se presentan en la red. Para la última medida el transformador tendría unos excesivos errores que hacen inviable esta opción. Se tendría que calibrar cada transformador tanto en error de módulo como de ángulo por medio del software.

Por otro lado, el uso de transformadores haría un dispositivo pesado y de grandes dimensiones, algo que no interesa dado que se busca un aparato compacto.

Por todos estos inconvenientes se decidió descartar esta alternativa.

2.1.2 Aislamiento óptico

Este tipo de aislamiento se lleva a cabo por medio de los llamados optoacopladores, es decir, componentes que acoplan circuitos ópticamente en el rango del espectro infrarrojo y a la vez ofrecen aislamiento galvánico. En caso de producirse un defecto en el circuito que se conecta a la red, la corriente de cortocircuito no afectaría al microcontrolador que se encuentra en la parte posterior y que lee la salida del optoacoplador. Por lo tanto, se trata de una medida de protección del Arduino.

En la figura 5 se muestra una imagen de la configuración interna de un optoacoplador. Como se puede ver contiene un diodo emisor de luz y un dispositivo detector de luz, llamados optodetectores, que actúan como receptores.

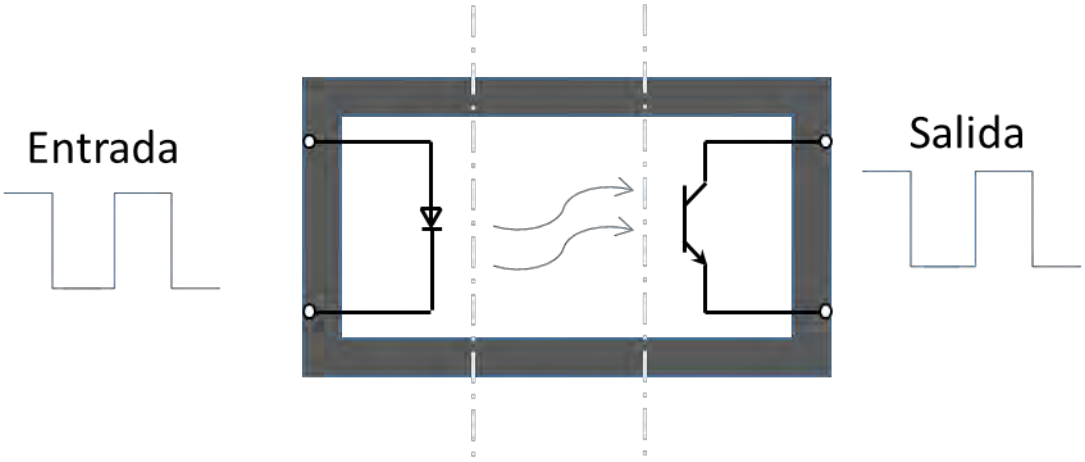


Figura 5: optoacoplador

El tipo de señal con la que alimentamos el optoacoplador es una señal discreta y no analógica. De aquí surge la necesidad de discretizar la señal de tensión para poder transmitirla a través de este. Para llevar a cabo esto, surgió la idea de convertir la señal de tensión a frecuencia, es decir, el valor de tensión en un instante convertirlo a una señal de pulsos cuyo periodo sea proporcional al valor de tensión en ese instante. De esta manera el optoacoplador nos permitiría transmitir esta señal.

Uno de los inconvenientes de este método es que el aislamiento galvánico no cubre por completo la circuitería y el conversor de tensión a frecuencia previo al optoacoplador no disfruta de dicha protección. Se explicará posteriormente cómo se soluciona por medio de fusibles y un diodo zener a la entrada del dispositivo.

La elección del optoacoplador fue posterior a la del conversor de tensión a frecuencia, en adelante VFC (Voltage to Frequency Converter), dado que este tenía que ser capaz de transmitir la señal que el VFC le enviaba. Fueron dos las características que se atendieron a la hora de buscar un optoacoplador:

- La velocidad de transmisión de la señal recibida debía ser lo suficientemente rápida. Esta característica se puede comprobar en el tiempo de subida y bajada y en el desfase del optoacoplador [2].
- La corriente de salida del VFC debía ser lo suficientemente grande para que el optoacoplador funcionase. Es necesario que esta corriente sea la suficiente para hacer que el diodo que contiene luzca [2].

Más adelante se mostrará una tabla comparativa entre las características del VFC y el optoacoplador para comprobar su compatibilidad.

2.1.2.1 Conversores tensión- frecuencia y frecuencia –tensión

Una de las alternativas barajeadas y finalmente rechazada, fue la de realizar la conversión a frecuencia para aislar los circuitos con el optoacoplador y de nuevo

convertir dicha señal a tensión. De esta manera se emplearían los pines analógicos del Arduino para hacer el muestreo.

La principal ventaja de este método es el uso del convertidor analógico del Arduino, en adelante ADC (Analog-to-Digital Converter), que posee una resolución de 12 bits y el tiempo de lectura de las entradas es de 1µs. Por lo tanto, el error asociado a esta medida muestreada es el error de cuantización. Este error se debe a que a los valores de una señal continua se le asocian valores digitales, por tanto, no todos los valores son posibles. El máximo error cometido, llamado error de cuantización, es decir, la diferencia entre el valor real y el valor asignado, será la mitad de la resolución:

$$\varepsilon_{cuantización} = \frac{V_{cc}/2^{12}}{2} = \frac{3.3[V]/2^{12}}{2} \cdot \frac{1\ [mV]}{1000\ [V]} \approx 0.4mV \qquad (1)$$

Como se puede observar en la figura 6, el error de cuantización es inapreciable para una señal de 50Hz de valor medio 1,65V y valor de pico 3.3V, que es el máximo que los pines del Arduino admite. Además también se puede observar que no existe apenas diferencia entre muestrear con una resolución de 8 bits. Una señal similar sería la que se introduciría una vez acondicionada desde la red. Para este ensayo se ha utilizado una frecuencia de muestreo de 250Hz, muy inferior a la que se consigue con este dispositivo.

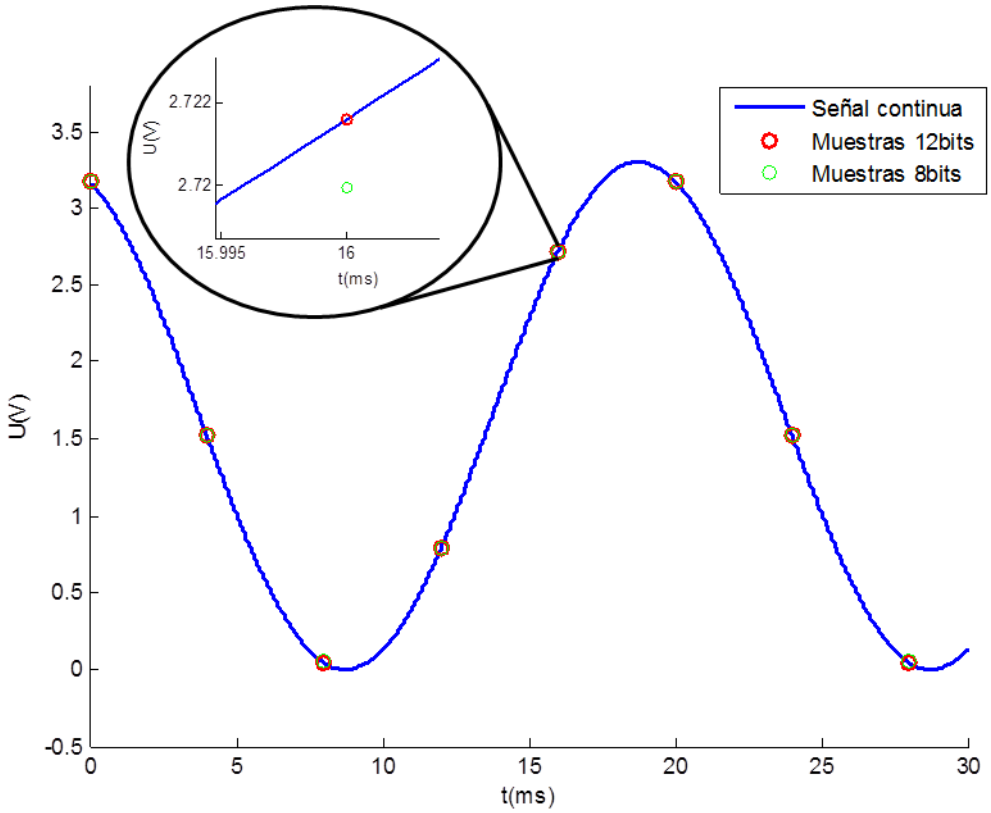


Figura 6: Señal de entrada analógica y muestreo con resoluciones de 12 y 8 bits

El error cometido con la resolución de 12 bits es muy pequeño y permite tomar medidas muy exactas. Se puede sacar en conclusión que los errores que se puedan llegar a cometer en las medidas provendrán de los circuitos y no tanto del muestreo.

El inconveniente de este método es la disponibilidad de VFC y conversores de frecuencia a tensión, en adelante FVC (Frequency-to-Voltage Converter), cuyos valores de frecuencia sean lo suficientemente elevados para poder transmitir nuestra señal de 50 Hz. Tras una intensa búsqueda el componente que era tanto VFC y FVC ofrecía salidas de rango de frecuencias insuficientes, $\Delta f \in [1,100.000]$ Hz. Transmitir una señal de 50Hz convirtiéndola a una señal de pulsos de menos de esa frecuencia hace inválido este método. Se necesitaría transmitir esa señal a través de frecuencia de al menos, cientos de kHz.

Este método sería el que mayor precisión ofreciese en caso de existir un VFC Y FVC que convirtiese dicha tensión a valores de frecuencia del orden de MGhz. El que sea el más preciso se debe a que el muestreo de tensión se realiza con el ADC del Arduino que, como ya se ha visto, muestrea de forma precisa. Se han encontrado VCOs que permiten realizar esta conversión, sin embargo, no sucede lo mismo con FVC. La dificultad estriba en que este método de acondicionamiento de tensión no es muy común entre los aparatos digitales de medida de tensión.

2.1.2.2 Conversor tensión- frecuencia y conteo de pulsos

Finalmente la elección definitiva fue la de emplear un VCO, el AD7741 [3], y el optoacoplador [2] para después medir la frecuencia mediante el microcontrolador. La diferencia respecto al anterior estriba en el uso del microcontrolador como frecuencímetro. Esta diferencia se aprecia en el error de cuantización medido en V, ecuación 2.

$$\varepsilon_{cuantización} = \frac{1\ [nº\ periodos]}{T_{conteo}[s] \cdot 2} \cdot \tau_{f-U}[V/Hz] \qquad (2)$$

El error de cuantización depende del periodo escogido para contar el número de pulsos, a mayor periodo menor será el error de cuantización. La resolución se entiende como el mínimo número de pulsos que se pueden contar para la mínima frecuencia del VCO, por tanto, a mayor periodo de conteo mejor será la resolución y menor el error de cuantización.

Para el cálculo de la frecuencia se emplea la ecuación 3 que se muestra a continuación, en ella, cada pulso equivale a un periodo y su división entre el periodo de conteo de pulsos permite obtener la frecuencia.

$$f\ [1/s] = \frac{nº\ pulsos\ en\ el\ intervalo}{T_{conteo}[s]} \qquad (3)$$

El valor de $\tau_{f\text{-}U}$ de la ecuación 2 indica la relación entre la frecuencia y la tensión dado por la hoja de especificaciones del VCO [3]. En la ecuación 4 se muestra dicha relación y en la figura 7 se representa gráficamente.

$$\tau_{f-U}[V/Hz] = \frac{2.5V}{(0.45 - 0.05) \cdot 6144kHz} \qquad (4)$$

Existe un compromiso entre el periodo de muestreo del dispositivo y el tiempo dedicado al conteo de pulsos, limitando el primero al segundo, esto se debe a que al aumentar el tiempo de conteo aumenta el periodo de muestreo. Otra limitación es la de la variación de tensión durante el periodo en el que se cuentan los pulsos, la frecuencia se modifica y se produce un error, cada vez mayor cuanto mayor sea el periodo de conteo.

El valor máximo de tiempo admisible para muestrear la señal de frecuencia no podrá ser superior a 1ms. De esta forma tendremos 20 muestras por periodo de señal, aunque se buscará aumentar este número al máximo para que el cálculo sea más preciso.

Para entender la conversión tensión-frecuencia de manera gráfica se muestra la figura 5. Se observa la conversión del VCO para los instantes en los que la tensión es mínima y máxima apreciándose como la frecuencia varía. Para las gráficas de los pulsos se ha

elegido el mismo tiempo en el eje de abscisas, 5µs. Estas gráficas se han obtenido mediante simulación, a través de Matlab, de la conversión realizada por el AD7741, el bloque de simulink se puede ver en la figura 8.

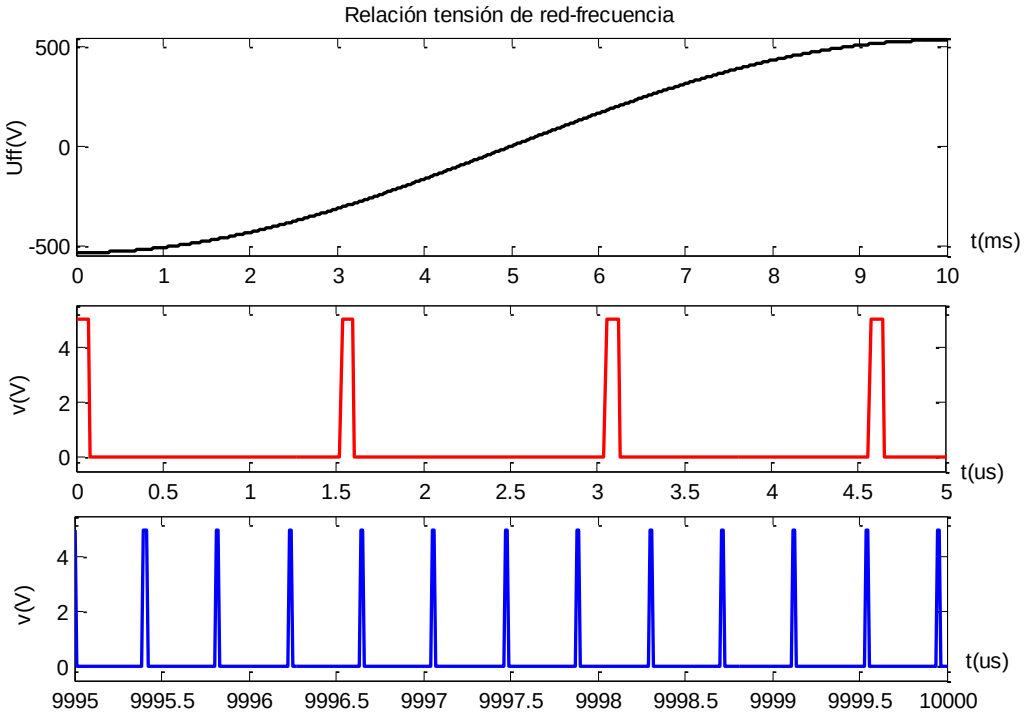


Figura 7: Conversión de tensión a frecuencia en el instante t=0ms y t=10ms

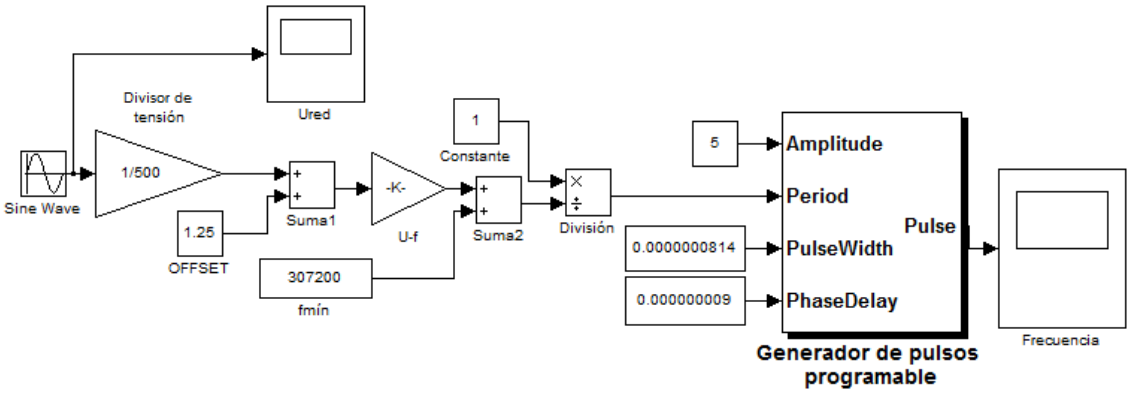


Figura 8: Modelo de Simulink del circuito de acondicionamiento de tensión

En la figura 7 se muestra una conversión de la tensión de red para el valor mínimo, gráfica roja, y el valor máximo, gráfica azul. La relación entre tensión de entrada al VCO y frecuencia de salida se muestra en la figura 9. La mínima frecuencia obtenida para el valor mínimo de tensión es de aproximadamente 479.4kHz, lo que permite transmitir la señal de tensión de 50Hz sin apenas error. La mínima frecuencia que puede dar el VCO es de 307kHz, esta frecuencia se obtiene aproximadamente para sobretensiones algo superiores al 10% de tensión compuesta. Este

sobredimensionamiento tiene por objetivo evitar posibles daños en los pines de entrada del VCO y demás componentes electrónicos.

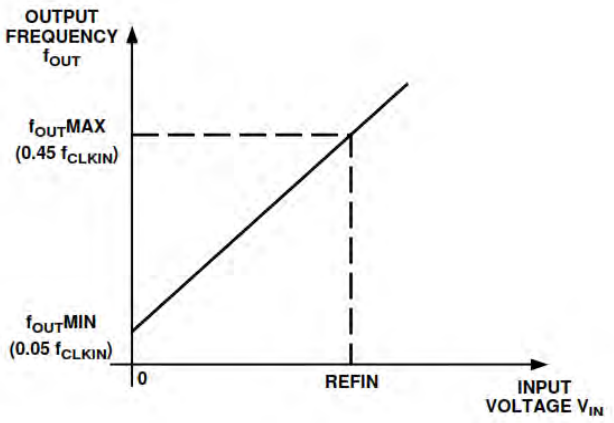


Figura 9: Relación U-f del VCO, gráfica obtenida de su hoja de especificaciones

El valor de f_{CLKIN} es el valor de frecuencia del oscilador de cristal que es necesario incorporar entre dos de los pines del VCO y oscila gracias a un resonador interno que es realimentado desde el VCO. Este cristal es de f_{CLKIN} = 6144kHz y por tanto, el rango de frecuencias a la salida es de $\Delta f \in$ [307.2, 2764.8] kHz. Como ya se comentó, este rango de frecuencias es bastante aceptable para transmitir la señal de 50Hz.

Para evitar que el cristal vibre con armónicos superiores a su frecuencia nominal es necesario introducir dos condensadores con respecto a la referencia, como se muestra en la figura 10.

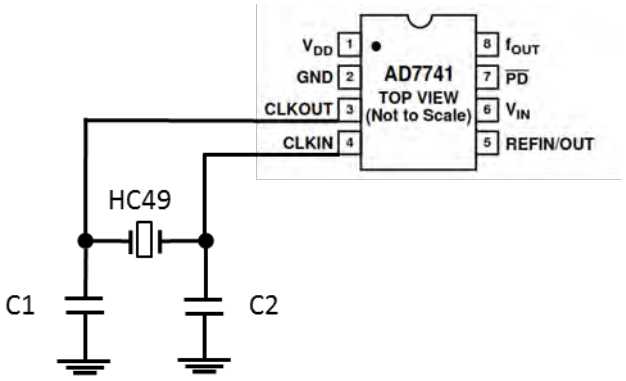


Figura 10: Oscilador de cristal del VCO

Para el cálculo del valor se recurre a la ecuación 5. Para poder hacer los cálculos es necesario consultar en la hoja de especificaciones técnicas de este oscilador de cristal, el HC49 [4] , el valor de la capacidad de carga es de 30pF mientras que la capacidad parásita es de 7pF. Haciendo que ambos condensadores sean del mismo valor, el resultado es que ambas capacidades deben valer 46 pF.

$$C_L = \frac{C_1 \cdot C_2}{C_1 + C_2} + C_S \qquad (5)$$

Para determinar el rango de tiempo dedicado a contar los pulsos es necesario estudiar el intervalo alrededor del punto t=5ms de la figura 5, es decir, aquella zona donde la tensión de entrada presente la mayor pendiente. Es el punto crítico de la conversión dado que la pendiente y el cambio de tensión es máxima, por tanto, afecta a la conversión. Además para determinar este tiempo es necesario estudiar el caso más desfavorable en el que la tensión de entrada sea la compuesta. En la figura 11 se ha representado el punto crítico y se ha determinado el tiempo de conteo asumiendo admisible la variación de 25.32V para el caso más desfavorable.

Para hacer la conversión a frecuencia de la tensión de entrada se diseñó un divisor de tensión de relación 0.002 y se añade un valor medio de 1.25V. Esto se explicará más adelante, sin embargo, es necesario conocer dichos datos para realizar los cálculos que viene a continuación.

Este incremento de tensión de 25.32V es equivalente a un Δf=49.7kHz, lo que para la medida en ese instante t= 5.075 ms de f=1560.95 kHz, supone un 3.2% de cambio de la frecuencia, porcentaje derivado de la ecuación 6.

$$\Delta f[\%] = \frac{39\ [kHz]}{1228.8\ [kHz]} \cdot 100 = 3.2\% \qquad (6)$$

El valor de dicha frecuencia de salida en el instante t=5.075ms se obtiene de la ecuación 6 para una tensión de entrada de 12.69V.

$$f_{salida} = \left[U_{entrada}[V] \cdot \frac{1k\Omega}{500k\Omega + 1k\Omega} + 1.25V\right] \cdot \frac{(0.45 - 0.05) \cdot 6144kHz}{2.5V} + 307.2kHz \qquad (7)$$

Sin embargo, esa cantidad no es equivalente al error cometido puesto que la frecuencia obtenida sería la media de ese periodo, lo que equivaldría a un 1.56% de error cometido. Además, este error no se traduce en un error de módulo, si no, en un error de fase. Es decir, al obtener la media al finalizar el periodo de conteo se obtendrá un valor inferior para pendientes positivas y superior para pendientes negativas. Esto último se puede apreciar en la figura 9.

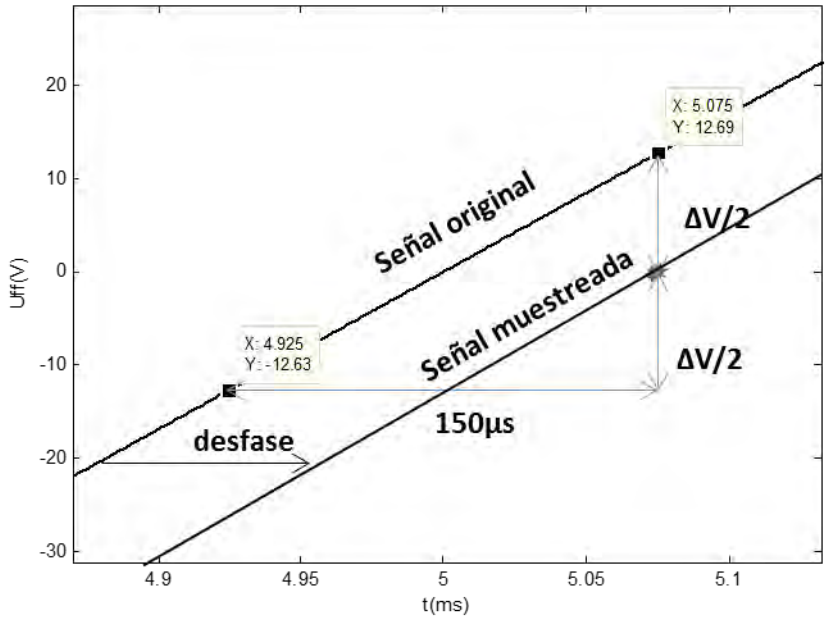


Figura 11: máxima variación de tensión para T_{conteo}

De la figura 11 y del razonamiento anterior se determina el periodo dedicado a contar los pulsos, este periodo se ha fijado en 150 µs y se implementará por medio del software. Este valor se ha obtenido de un acuerdo entre la búsqueda del máximo número de puntos muestreados por periodo y una mínima variación de la tensión de entrada. Además el valor límite inferior del tiempo de conteo estaba determinado por el error de cuantización, que como se ha visto, cuanto menor sea e periodo de conteo mayor será el error.

Otra de las conclusiones obtenidas de la figura 11 es que el desfase introducido sobre la señal original es de un valor teórico de 75µs, equivalente a 1º 21'.

Una vez determinado el tiempo de conteo, es posible calcular el error de cuantización.

El pequeño error de cuantización es debido a que los pulsos contados equivalen al número de periodos según el cálculo que se realiza representado en la ecuación 3 anterior. Sin embargo, el número de pulsos no coincide exactamente con el número de periodos y, por tanto, se produce un pequeño error, esto se aprecia fácilmente en el ejemplo de la figura 12. En ella se aprecia como el número de periodos existentes en ese intervalo es de 8 más ½ , sin embargo, el método de cálculo supondría solo 8 pulsos cometiéndose un error.

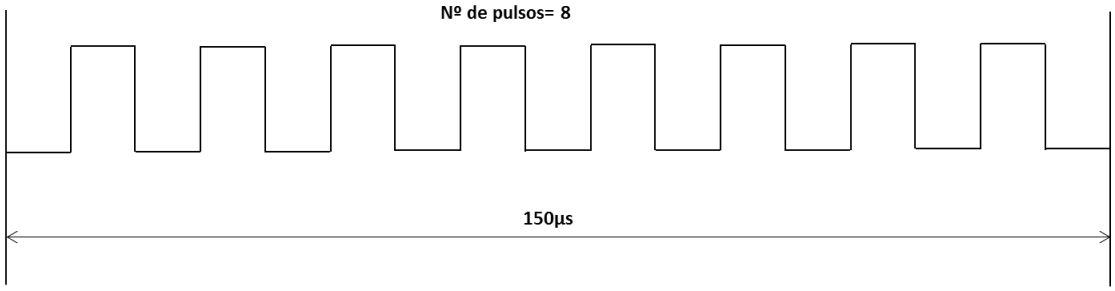


Figura 12: Ejemplo de conteo de pulsos en el periodo de muestreo fijado de 150µs

Este error es el de cuantización y se reduce cuanto mayor sea el número mínimo de pulsos que se vayan a contar. Dado que la frecuencia mínima que se obtiene con el VCO es de 307.2kHz el menor número de pulsos es de 46 para los 150µs. Al ser tan elevado dicho número, el error de cuantización relativo es muy pequeño y mejora el de las entradas analógicas del Arduino.

El error máximo que se puede cometer es la mitad de un periodo, es decir, medio pulso de error. Esto equivale a $3.\widehat{3}$ kHz, es decir, el valor del error de cuantización calculado que en valor relativo respecto a la mínima frecuencia es de 1.08%. Este error tan pequeño para la mínima frecuencia medida se traducirá en una precisión en las medidas de tensión muy buena como se podrá ver en los experimentos.

Dado que el tiempo de conteo es de 150 µs, la resolución obtenida de este método, según ecuación 2, es de 3.4mV. Este es superior a la obtenida de los pines analógicos del Arduino, 0.8mV, pero que como ya se acaba de comentar en valores relativos mejora las de éste. Como el conversor analógico-digital del Arduino mide valores mínimos de 0V este error de cuantización relativo para tensiones mínimas es muy superior al conseguido con el método de este PFG.

Otra consideración a tener en cuenta son los niveles de tensión entre los elementos. El VCO trabaja a un nivel de tensión de 5V, es decir, los pulsos de frecuencia tienen ese valor de tensión, sin embargo, los pines digitales del Arduino destinados a medir frecuencia solo admiten 3.3V máximo. El uso del optoacoplador además de ofrecer aislamiento galvánico permite ajustar a su salida el nivel de tensión y transmitir los pulsos a 3.3V.

2.1.2.3 Acoplamiento entre VCO y optoacoplador

Del apartado 2.1.2 acerca de las características que debía cumplir el optoacoplador, se concluye que una de las dos no se consiguió debido a inmensidad de la oferta de optoacopladores y la dificultad para encontrar el adecuado. Sin embargo, se adquirió a pesar de ello debido a que la solución a este problema era relativamente sencilla. El problema del que se habla es la falta de corriente para encender el diodo que contiene el optoacoplador. De la tabla 5 obtenida de la hoja de especificaciones del optoacoplador [2], se obtiene que la corriente necesaria debía estar en el rango $\Delta I \in [4 , 8]$ mA

mientras que el VCO ofrece corrientes en el rango de ΔI Є [0.8 , 1.6] mA, datos reflejados en la tabla 4.

Para solucionar este problema se incorporó un buffer de tensión, según figura 13, aunque previamente se barajeo la posibilidad de incorporar un transistor, como se puede ver en el esquema de la figura 14. Ambos métodos son válidos siempre y cuando la velocidad de transmisión de la señal se cumpla.

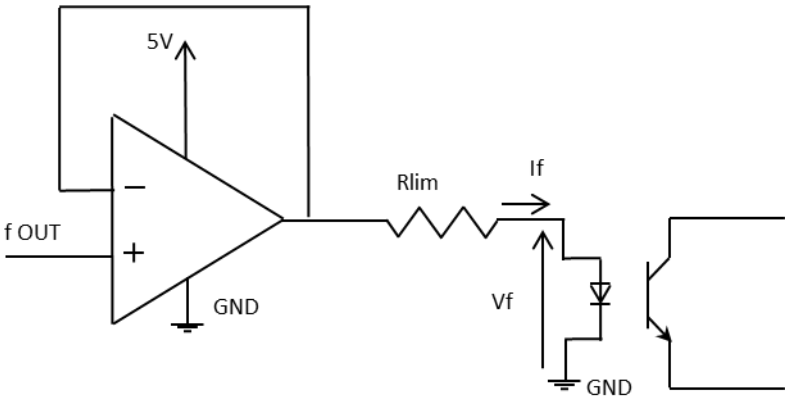


Figura 13: Alimentación optoacoplador mediante buffer de tensión

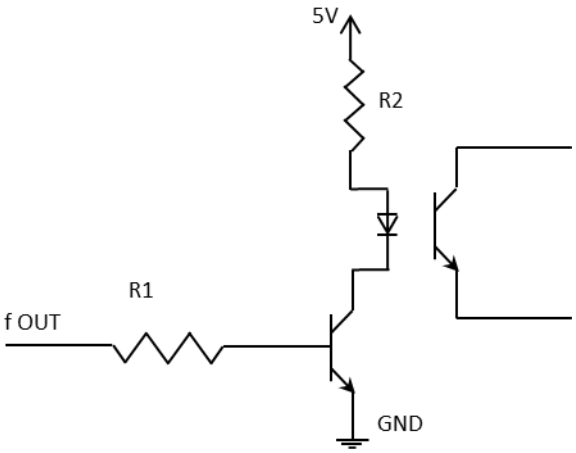


Figura 14: Alimentación optoacoplador mediante transistor

Para determinar la R_{lim}, que fija la corriente para que conduzca el optoacoplador, se recurre a la ecuación 8 y se toman los datos de la tabla 5 del optoacoplador.

$$Rlim = \frac{Vcc - Vf}{If} = \frac{5 - 1.5}{0.006} = 583.3\Omega \qquad (8)$$

Las características de este buffer se pueden encontrar en la tabla 6, el buffer empleado es el 74HC244. Este buffer dispone de 6 entradas y salidas de las cuales se emplearán 3 para cada una de las tensiones a acondicionar.

Los requisitos que se buscaron para este componente eran que los tiempos de subida y bajada fueran los suficientemente bajos para transmitir una señal de pulsos que, a la máxima frecuencia, 2764.8 kHz, es decir, 362 ns entre pulsos, tuviera 4ns de subida y de bajada proveniente del VCO, véase tabla 4. No era necesario que fueran inferiores, algo que es difícil de conseguir de un buffer.

Como se puede ver los tiempos de subida y bajada son ligeramente superiores a los del VCO, sin embargo, la señal sigue siendo admisible aunque un poco retrasada. Las características de tiempos del optoacoplador son muy inferiores a los de los demás componentes, sin embargo, con la introducción del buffer de tensión bastaba con que fueran inferiores a los del buffer.

VCO, AD7741		
Parámetro	**Valor**	**Unidad**
fmín	307.2	kHz
fmáx	2764.8	kHz
Retardo	9	ns
Ancho pulso	81.4	ns
t subida	4	ns
t bajada	4	ns
Vdd	5	V
Vin	0 a 2.5	V
Consumo máx. (1)	8	mA

Tabla 4: Características del VCO destacables [3]

(1) Consumo máximo cuando la salida está sin carga

De forma gráfica se representa en la figura 15 algunos de estos parámetros. La señal de CLKIN es la del oscilador de cristal respecto del cual se toman las medidas temporales de la salida del VCO.

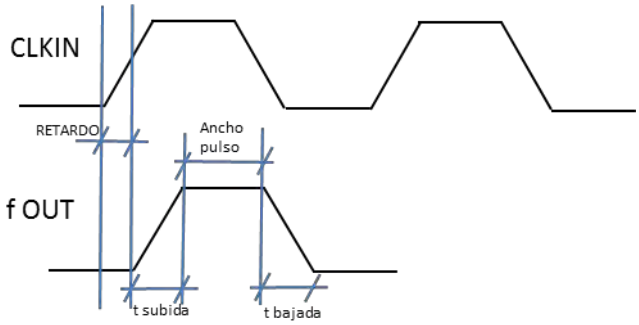


Figura 15: CLKIN, señal del cristal y f OUT, salida del VCO. Parámetros temporales del VCO

OPTOACOPLADOR, ACPL-W70L		
Parámetro	**Valor**	**Unidad**
If	4 a 8	mA
Vf	1.2 a 1.85	V
Retardo máx.	55	ns
t subida	3,5	ns
t bajada	3,5	ns
Aislamiento mín.	5000	Vrms
Rentrada-salida	10^{12}	TΩ

Tabla 5: Características del optoacoplador destacables [2]

Buffer, 74HC244		
Parámetro	**Valor**	**Unidad**
Vcc	2 a 6	V
Retardo	18 a 27	ns
t subida	12 a 18	ns
t bajada	12 a 18	ns
Consumo[1]	< 1	mA

Tabla 6: Características del buffer destacables [5]

La suma de los retardos introducidos por el optoacoplador y el buffer es inferior al mínimo tiempo entre pulsos, lo que evita que los pulsos se unan entre sí y se cuente mal la frecuencia. El retardo total máximo introducido por ambos es de 82ns mientras que el mínimo tiempo entre pulsos es de 362ns.

Otra consideración a tener en cuenta era la de que el buffer debía trabajar a 5V dado que la salida del VCO estaba en ese nivel de tensión.

[1] El consumo considerado es para las tres entradas empleadas en el PFG. Se considerará 1mA para dimensionar la batería recargable empleada.

2.2 Reducción de tensión

Para reducir la tensión se implementa un divisor de tensión dimensionado para la máxima tensión que pueda aparecer en la red. A la tensión reducida es necesario sumarle un valor medio de 1.25V para introducirlo posteriormente en el pin del VCO. La salida del divisor nunca podrá ser superior a 1.25V de pico dado que la entrada del VCO no admite más de 2.5V. La tensión más elevada que puede aparecer en la red es de un 10% superior a la nominal. máxima tensión es de 591.1V según la ecuación 9.

$$Umáx = 380 \cdot \sqrt{2} \cdot 1.1 = 591.1\,V \qquad (9)$$

El divisor por tanto se diseña según la relación de la ecuación 10.

$$Relación_{divisor} = \frac{R_1}{R_1 + R_2} = \frac{1.25}{591.1} = \frac{1}{472.91} \qquad (10)$$

Por lo tanto, las resistencias seleccionadas serán de $R_1 = 500k\Omega$ y $R_2 = 1k\Omega$. Se ha dejado un margen superior al que dictaba la ecuación 6 debido a que al añadir posteriormente un valor medio a dicha señal, este puede tener un error y ser superior o inferior a 1.25V y, por tanto, superar el límite de la entrada del VCO.

La máxima potencia que debe soportar cada una de ellas es de 348mW y 0.7mW respectivamente, según la ecuación 11. Para el cálculo en la ecuación 11 se ha considerado un 10% de sobretensión ya que es la sobretensión real que puede aparecer en sus bornas.

$$Pmáx\ de\ Rx = \frac{(U_{máx\,rms} \cdot \frac{R_x}{R_y + R_x})^2}{R_x} \qquad (11)$$

2.3 Referenciación de entradas de tensión y conexionado con red

Uno de los principales problemas surgidos durante este PFG fue el de implementar una referencia flotante respecto de la cual medir las diferencias de tensión. En caso de no ser flotante y emplear la tierra común a la red, la corriente de cortocircuito tendría un camino por el que circular atravesando componentes conectados a dicha tierra.

Tras buscar diferentes soluciones al problema se encontró un dispositivo capaz de referenciar una entrada diferencial a una referencia determinada, el AD623ANZ [6]. Este dispositivo es un amplificador de instrumentación, es muy utilizado en medida de corriente a través de la caída de tensión en una resistencia de valor pequeño conocida como en instrumentación eléctrica como ‘shunt’.

En este caso lo que se pretende es medir la caída de tensión en la resistencia del divisor de tensión. El esquema de este método se puede ver en la figura 16.

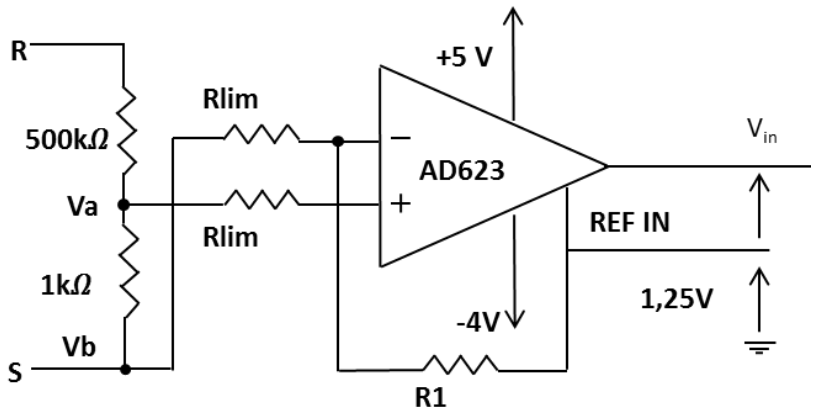


Figura 16: Esquema de referenciación de la señal de tensión diferencial

Este amplificador es un dispositivo que internamente realiza la función del esquema restador realizado mediante amplificadores operacionales. En un primer momento se diseñó el circuito mediante el esquema de la figura 17, sin embargo, para facilitar los circuitos impresos y reducir el espacio ocupado de estos se prefirió emplear este amplificador de instrumentación. Además, éste proporciona unas mejoras prestaciones en cuanto que está diseñado específicamente para esta función, como pueden ser la linealidad, la estabilidad con la temperatura y un consumo bajo menor de 3mW.

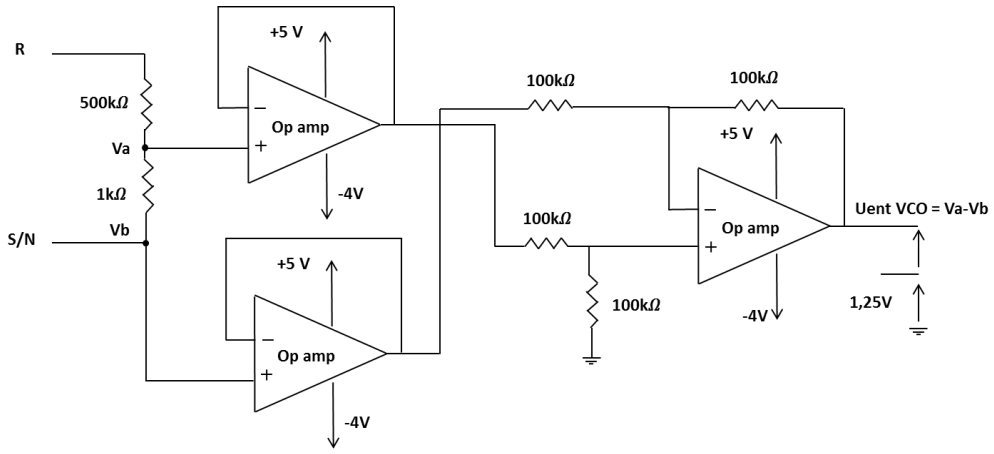


Figura 17: esquema diferencial con amplificadores operacionales

La realimentación del pin de referencia con la entrada inversora del amplificador de instrumentación, apreciable en la figura 16, da un camino a las corrientes de polarización. Estas corrientes son necesitadas por los transistores que se encuentran en las entradas del amplificador y que les permiten trabajar en una determinada región de la curva de transconductancia[2]. Esta región de trabajo es la que les permite a los

[2] Transconductancia, es la relación entre el incremento de corriente de la salida del amplificador y el incremento de tensión a la entrada. Cuando la corriente de polarización no retorna hacía los

transistores trabajar de forma lineal. En caso de faltar esta realimentación de la corriente de polarización, los transistores trabajarían en zonas extremadamente no lineales causando una distorsión de la salida.

Además de evitar la circulación de la corriente de cortocircuito, otra necesidad de referenciar la entrada de tensión a una tierra flotante se debe a que la entrada del VCO no era diferencial, es decir, mide la tensión respecto una referencia. Existen VCOs diferenciales que hubieran evitado la necesidad de llevar a cabo esta referenciación, sin embargo, aquellos conversores diferenciales encontrados no tenía las características que interesaban tales como un rango de frecuencia de salida muy superior a los 50Hz de la red.

Dado que el dispositivo es trifásico, tres circuitos idénticos se instalan en paralelo llevando las medidas la misma referencia. La dificultad y el peligro de emplear la misma referencia es la posibilidad de aparición de cortocircuitos que destruyan los componentes al conectar inadecuadamente las fases. Para evitar este hecho es necesario considerar los puntos del conexionado entre los cuales las fases podrían llegar a unirse. Véase la figura 18.

transistores la transconductancia de estos varía considerablemente afectando a la linealidad de la salida.

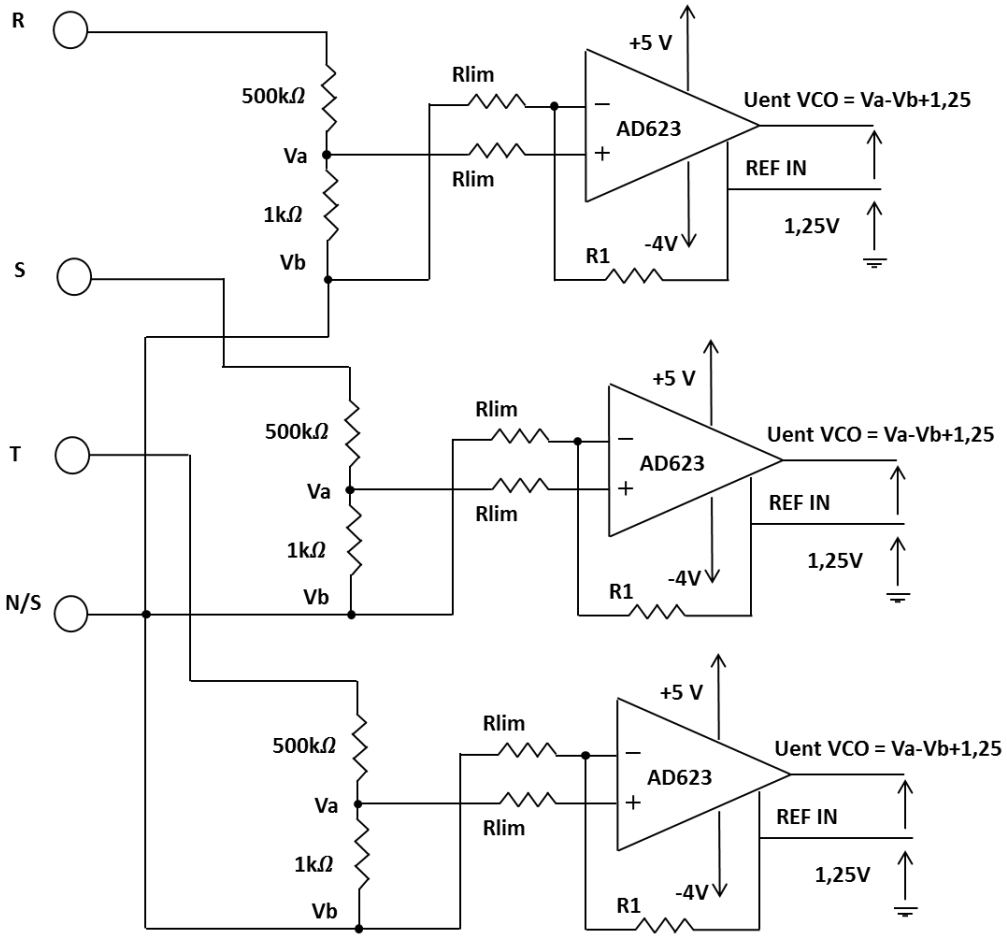


Figura 18: Conexionado circuitos de tensión

Como los pines inversores de los amplificadores están unidos al mismo punto todas aquellas conexiones desde los hilos de la red a este pin han de ser el mismo punto en la red.

Según el método elegido, Aron o tres vatímetros, la conexión es diferente. En la tabla se muestra la conexión según cada método.

MÉTODO		ARON	TRES VATÍMETROS
BORNA	R	R	R
	S	S	S
	T	T	T
	N/S	S	N

Tabla 7: Conexión según método

Este tipo de conexionado evita que se produzcan cortocircuitos indeseados a menos que la conexión de los bornes no sea la indicada por el manual de usuario. De la tabla 7 se deduce que cuando el método seleccionado a través de la interfaz del usuario es el de

Aron, entonces, el usuario tiene la obligación de conectar la borna N/S junto a la borna S y seguido de la conexión a la fase S.

En cualquier caso, todas las bornas deben estar conectadas para un correcto funcionamiento.

Es de gran importancia que el método elegido en la interfaz coincida con la conexión de la tabla 7. Los cálculos realizados por software dependen del conexionado.

La referencia flotante de la que se habla ha de ser distinta de las de los demás circuitos del dispositivo. Esto es debido a que parte de los circuitos de tensión no están aislados galvánicamente y el emplear una referencia común para todo el dispositivo puede ser peligroso para todos los componentes conectados a dicha referencia. Este peligro estriba de la posibilidad de ofrecer un camino a la corriente de cortocircuito a través de dicha referencia hacia todos los componentes.

2.4 Alimentación y referencia

Para emplear referencias distintas entre los circuitos de acondicionamiento de tensión y los demás circuitos de intensidad, LCD, USB y reloj se emplean dos baterías en este PFG, una para alimentar la parte no aislada galvánicamente de los circuitos de tensión y la otra para alimentar al Arduino, desde el que se alimentan todo lo demás.

La alimentación requerida por los componentes de los circuitos de tensión es tanto positiva, 5V, como negativa, -4V. Esos valores han sido escogidos para facilitar los circuitos. El valor de 5V era necesario para alimentar componentes como el AD7741 cuyo rango de alimentación es $\Delta V_{CC} \in [4.75, 5.25]V$ lo que obligaba a que la alimentación negativa fuera de -4V al tener una tensión de alimentación de 9V. Para obtener dichas tensiones y además una referencia flotante se empleó una batería recargable cuya tensión debía ser superior a la tensión de entrada del regulador de tensión. Este mantiene una tensión de 9V entre un rango muy reducido de variación, $\Delta V_{CC} \in [8.55, 9.45]V$ [7].

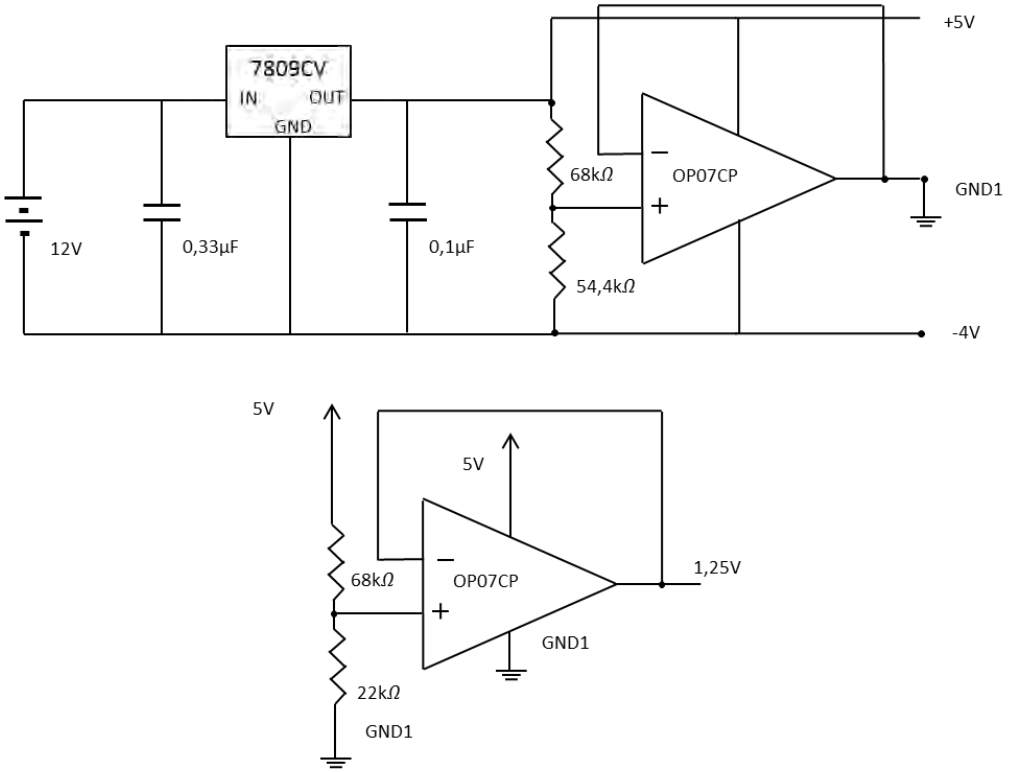


Figura 19: Esquema completo de alimentación en circuitos de tensión

El componente de la figura 19 llamado 7809CV es un regulador de tensión cuya finalidad es la de fijar 9V de tensión sin apenas variación. Esto se debe a que las baterías a medida que se descargan pierden valor de tensión nominal y dado que el AD7741 tiene un rango de alimentación muy restrictivo, la incorporación de este regulador de tensión se hizo imprescindible.

Otro de los peligros a evitar mediante este regulador de tensión es la posible variación del valor medio introducido a la señal de tensión, este error podría llegar a superar el valor límite inferior de 0V del AD7741. El punto crítico es el valor de 0V debido a la descarga de la batería. Sin el uso del regulador, al descargase ese valor medio se reduciría proporcionalmente hasta que la tensión de entrada al VCO tomase valores negativos y la conversión a frecuencia no sería satisfactoria aumentando los errores en la medida. En cambio, empleando el regulador de tensión solo es peligroso cuando la batería se descarga y no da la suficiente tensión requerida por el regulador, en ese caso, el regulador no daría tensión y la tensión a la entrada del VCO sería negativa durante el semiciclo de tensión inferior a 0V.

El AD7741 aguanta valores de tensión inferior a 0V de hasta -5V por lo que no habría peligro de dañar el componente. En cuanto a tensiones superiores a 2.5V soportaría tensiones de 5.3V, por lo que en ningún caso se dañaría el componente.

La función realizada por el buffer de tensión de la figura 20, es evitar que al conectar aparatos a la alimentación las tensiones del divisor se modifiquen debido a un cambio en la corriente que circula por dicho divisor. De esta manera se asegura que la tensión del divisor no se modifique.

De la misma manera que el anterior, para obtener el valor medio de tensión continua que es necesario añadir a la señal para introducirla en el VCO, se emplea un buffer de tensión que evite una modificación en dicho valor.

Para dimensionar la batería empleada en los circuitos de tensión fue necesario emplear una fuente de alimentación que simulará la batería a comprar y determinar así la corriente drenada.

2.5 Perspectiva completa del esquema de tensión

Al haber analizado los componentes del circuito de tensión siguiendo cronológicamente su diseño, ahora es necesario hacer una síntesis de lo que se ha visto hasta ahora. En la figura 20 se muestra el esquema completo de una de las fases del circuito de acondicionamiento de tensión.

La secuencia es la siguiente:

1. Reducción de la tensión de red a un nivel admisible para la entrada del AD7741 (VCO).
2. Referenciación de la señal de tensión a una tierra flotante obtenida de la alimentación de los circuitos de acondicionamiento de tensión.
3. Conversión de tensión a frecuencia.
4. Regulador de corriente mediante buffer y resistencia limitadora (R_{lim2}).
5. Aislamiento galvánico mediante optoacoplador, se transmite la señal idéntica aunque un poco desfasada.
6. Lectura de frecuencia a través del Arduino.

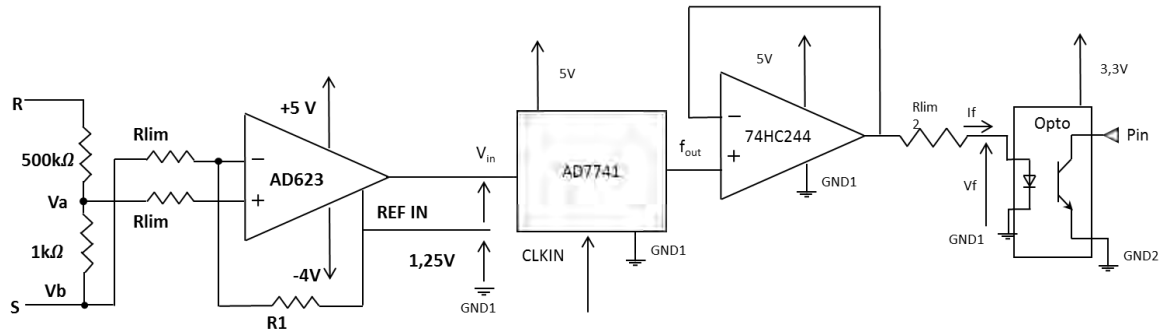


Figura 20: Esquema del circuito de acondicionamiento de la señal de tensión

En la figura 20 se aprecia la existencia de dos referencias, GND1 y GND2. La primera es la obtenida de la alimentación como ya se ha explicado en el apartado 2.4, mientras que la segunda proviene del pin del Arduino.

3 Acondicionamiento de corriente

El circuito de intensidad elegido viene condicionado por la presencia de transformadores de intensidad ya instalados a la salida del transformador de potencia del edifico de ICAI. La salida de estos transformadores es de intensidad nominal 5A. Para dar aislamiento galvánico se emplean transductores de corrientes que ofrecen una tensión de salida con un valor medio de 2.5V y que sigue la ecuación 12. Relación que también se puede ver gráficamente en la figura 21. Estos transductores de corriente

posee diferentes modos en los que la intensidad nominal del primer es posible modificarla. Para la aplicación del presente trabajo se realizará la conexión para cuya intensidad nominal, I_{PN} es de 6A.

$$V_{OUT} = 2.5 + 0.625 \cdot \frac{I_P}{I_{PN}} \qquad (12)$$

La máxima tensión que puede dar a la salida es de 3.38V, para una corriente de pico instantánea de 8.48A, lo cual supera por apenas unos 80mV la tensión que es capaz de leer las entradas analógicas del Arduino Due. Este sobrevoltaje no tiene efectos negativos en los pines analógicos del Arduino, solo provocaría que se leyese una tensión de 3.3V cuando en realidad es superior a este valor.

El transductor de corriente posee un pin de referencia que permite cambiar el valor de 2.5V a otro que interese, siempre y cuando se encuentre en el rango de tensión [1.9, 2.7] V. Para nuestra aplicación se introducirá aproximadamente 2.25V empleando un divisor de tensión desde el pin de 3.3V. La tensión para alimentar a dicho pin ha de salir de un buffer dado que la intensidad introducida a dicho pin según su hoja de especificaciones es de al menos 1mA [8].

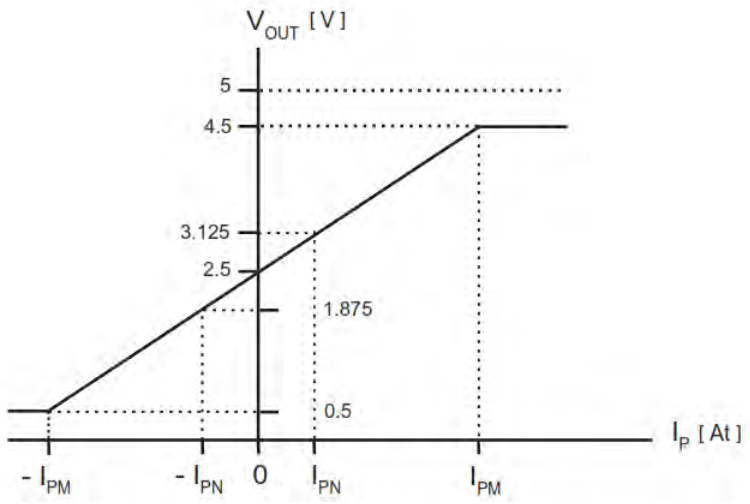


Figura 21: Conversión del transductor de corriente [8]

La conexión de los tres transductores es idéntica y la salida de cada uno de ellos va a tres de los pines analógicos del Arduino. La conexión de una de las fases se muestra a continuación en la figura 22.

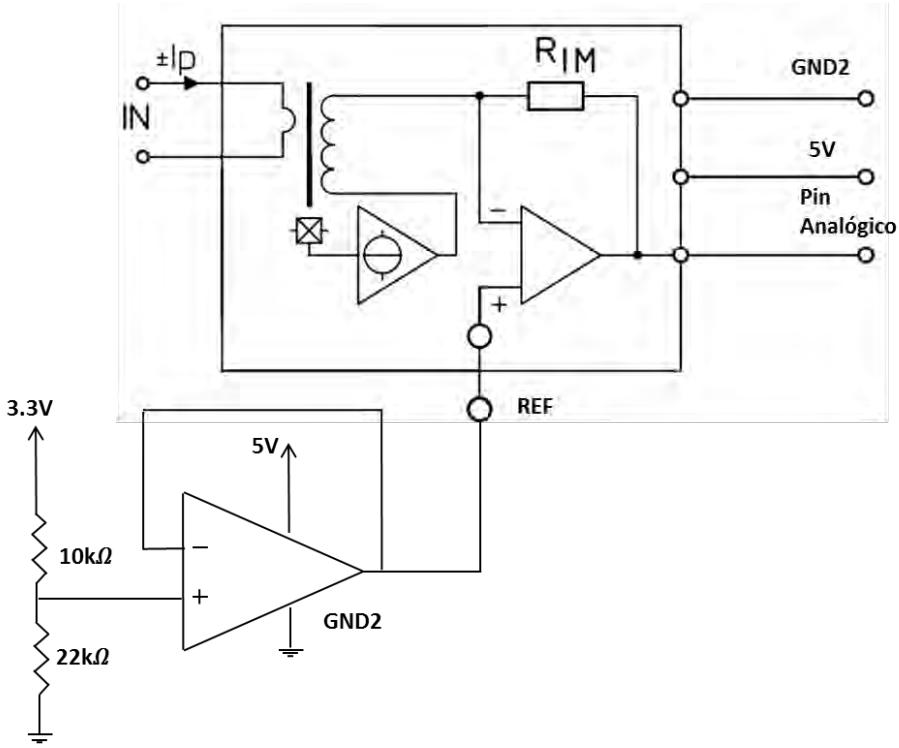


Figura 22: Conexionado de los transductores de corriente

El valor medio de 2.25V obtenido del buffer que se muestra en la figura 22 es compartido por los tres transductores dado que, a pesar de conectar tres cargas a la misma salida del amplificador operacional, dicha tensión no se ve modificada. Cada uno de estos pines de referencia consumirá 1mA de la alimentación.

4 Reloj de Tiempo Real

Módulo RTC (Reloj de Tiempo Real, siglas en inglés: "Real Timer Clock") del que se obtienen las mediciones de tiempo y temperatura con una precisión en tiempo buena y no tanto para la temperatura.

Este módulo es el DS3231BNZ, dispone de un bus de comunicación en serie I2C. Este tipo de comunicación emplea dos líneas para la transmisión de datos, una para la señal de reloj y otra para los datos. Estas líneas necesitan resistencias pull-up como se muestra en la figura 23.

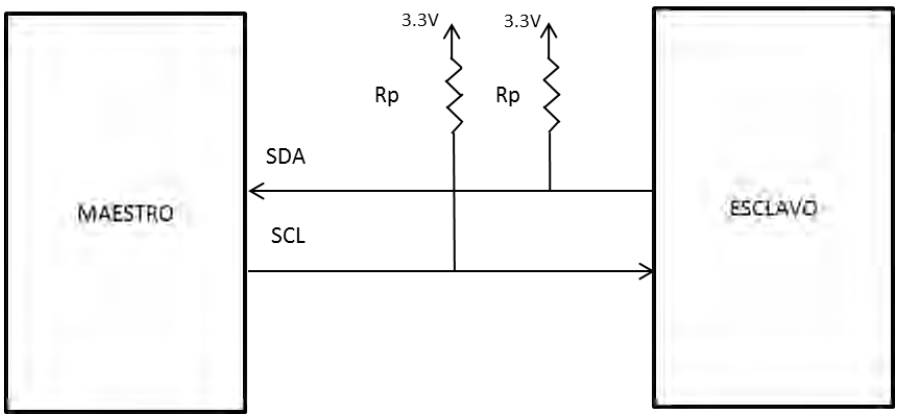


Figura 23: Buses de comunicación I2C con un solo esclavo

Las dos líneas de comunicación son SDA y SCL, la primera es por donde se transmiten los datos y la segunda es el bus del reloj. Este último permite que la comunicación sea satisfactoria haciendo que tanto el maestro como el esclavo se sincronicen y la información no se pierda.

El valor de las resistencias pull-up se obtiene con las ecuaciones 13 y 14 [9] y que dependen de la frecuencia a la que se pretenda enviar la información. El valor de C_{BUS} es la capacitancia total de cada línea que es de 400 pF.

$$Modo\ rápido\ \rightarrow Rmín = \frac{t_R}{C_B} = \frac{300 \cdot 10^{-9}[s]}{400 \cdot 10^{-12}} = 750\ \Omega \qquad (13)$$

$$Modo\ estándar\ \rightarrow\ Rmáx = \frac{t_R}{C_B} = \frac{1000 \cdot 10^{-9}[s]}{400 \cdot 10^{-12}} = 2500\ \Omega \qquad (14)$$

Este tipo de protocolo permite conectar a estas dos líneas más dispositivos llamados esclavos e identificados por unas direcciones determinadas. En este caso el dispositivo maestro será el Arduino, es el encargado de enviar un byte con la dirección del circuito integrado al que se quiere dirigir para después mandar o pedir los datos que precise. El byte que identifica al RTC como esclavo es el 0x68 en código Hexadecimal.

La trama de bites del protocolo es como se muestra en la figura 24.

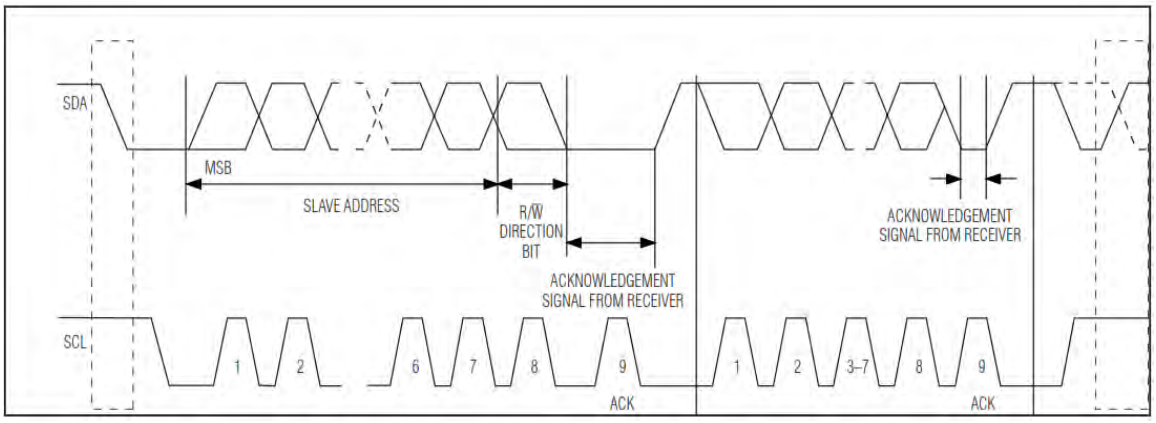


Figura 24: trama de comunicación I2C obtenida de la hoja de especificaciones del DS3231 [9]

En primer lugar, la condición de comienzo se fija cuando el maestro pone a cero el bus SDA cuando la línea de reloj está a tensión. Una vez comenzada la comunicación el maestro manda un byte (ADD, de 'Address') por el cual identifica el esclavo con el que se quiere comunicar. Después todos los datos que sean enviados deben ser enviarse cuando la línea de reloj este a cero y a cada pulso del reloj se transmite un bit de información. Cuando el receptor lea un byte confirma dicha lectura con un noveno bit. Toda esta secuencia es administrada por el propio Arduino y controlada en este caso mediante la librería Wire.h.

El módulo opera a 3.3V por lo que es compatible con el Arduino Due.

Dispone de un pin para alimentación desde una pila, lo que le permite llevar la cuenta del tiempo a pesar de que la alimentación del Arduino se haya perdido, esto fue el principal motivo de la elección de dicho elemento. La función de esta pila es la de alimentar al reloj en caso de que la alimentación desde el pin de 3.3V del Arduino falle.

Otro motivo por el cual se eligió este reloj de tiempo real es su precisión, ±2ppm. El Arduino Due dispone de un RTC incorporado pero su precisión no es tan buena como el DS3231 y tiene otro inconveniente, la desconfiguración del reloj ante una pérdida de alimentación.

Además este circuito integrado ofrece la posibilidad de monitorear la temperatura, aunque su precisión no es muy buena, ±3ºC.

Para comunicarnos con el dispositivo se hará uso de la librería que implementa el software de Arduino. Esta librería permite emplear de forma sencilla la comunicación en serie I^2C evitando la programación de niveles más bajos.

Los datos que proporciona este RTC y su disposición en los registros se puede ver en la tabla 8.

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Tabla 8: Registros del DS3231BNZ que contiene la fecha y hora. Tabla extraída de la hoja de características [9]

5 Almacenamiento de datos

Todos los analizadores de redes y contadores de energía analizados en el estado del arte tienen la posibilidad de guardar los parámetros que calculan, ya sea bien por medio de una memoria interna, externa o por transmisión de datos.

En este trabajo se eligió la posibilidad de almacenar datos en una memoria externa y se analizaron dos posibilidades expuestas a continuación.

5.1 Módulo de tarjeta SD

Se analizó la posibilidad de incorporar un módulo que permitiese introducir tarjetas SD y mediante la librería de Arduino SD controlar el almacenamiento de los datos [10]. Esta era la opción más sencilla puesta que esta librería simplifica mucho la programación del módulo.

Se desechó la opción debido a la incomodidad del uso de tarjetas SD para almacenamiento de los datos. Además estas tarjetas tienen menos memoria que los lápices USB.

5.2 Módulo con puerto USB

La opción seleccionada fue la del módulo VDIP1 [11] que aparece en la figura 25.

Figura 25: módulo VDIP1

Este módulo dispone de un firmware, el VNCIL, de propósitos de lectura/escritura que establece la lógica de más bajo nivel para el control del módulo. Este firmware ofrece una serie de comandos para su control, en la tabla 9 se muestran aquellas empleadas para el presente trabajo y su función.

Todas las versiones de este firmware solo soportan memorias USB cuyo sistema de archivo sea FAT32 con sectores de 512 bytes. Por tanto, antes de emplear un USB para este módulo, será necesario formatearlo para hacer que sea compatible.

Comandos	*Función*
IPA↵	Establece el código ASCII para los valores transmitidos
OPW.file.datetime↵	Abre un archivo de texto con la fecha deseada
CLF.file↵	Cierra el archivo de texto con el nombre especificado
WRF.dword↵	Escribir los números de bytes especificados

Tabla 9: Comando empleados del firmware del VDIP1 [12]

Dado que la comunicación con este integrado escogida fue UART [13], se empleó las funciones del IDE (en español, Ambiente de Desarrollo Integrado) de Arduino para establecer este protocolo a través de los pines destinados a ello. En este caso se utilizan los pines 18 y 19 que corresponden con el puerto serie 1 del Arduino.

Para enviar los parámetros y los datos se emplea el comando Serial.print1() que envía los datos introducidos entre paréntesis a través del pin TX, el pin 18, y que por defecto están en código ASCII.

Para configurar el integrado VDIP1 se establece, en primer lugar, el tipo de código numérico para el intercambio de información entre el integrado y el Arduino. Este podrá ser ASCII o binario. En este trabajo se prefirió trabajar, por comodidad, en código ASCII empleando el comando IPA de la tabla 9.

Con el segundo comando se consigue abrir un archivo de texto seguido del nombre del archivo y la fecha de creación. Dicha fecha es obtenida a través del RTC explicado anteriormente. Al final de cada sentencia es necesario enviar un retorno de carro

(símbolo ↲ y valor 13 en decimal). Una vez que se ha abierto dicho archivo todos los datos enviados se escribirán en él y no es necesario abrir de nuevo el archivo para seguir escribiendo.

El comando CLF seguido del nombre del archivo permite cerrar el archivo de texto sobre el que se está escribiendo. Este comando se emplea cuando no es necesario seguir escribiendo en él.

Y por último, el comando WRF (en inglés Write To File) permite escribir datos en el archivo que previamente se ha abierto. Seguido de dicho comando se envían el número de caracteres que se van a escribir a continuación. Por ejemplo si se quisiese escribir la temperatura medida por el RTC el código sería tal que así:

```
Serial1.print("WRF ");        //El espacio seguido del comando determina el fin del comando
Serial1.print(7);    //Siete es el número de caracteres enviados
Serial1.print(19.00);                    //Temperatura de 19ºC
Serial1.print("ºC");
Serial1.write(13);  //Retorno de carro que finaliza la serie de datos
```

El nombre de los archivos tiene como límite ocho caracteres, seguido de un punto, más otros tres caracteres opcionales para la extensión. En caso de superar la extensión de 8 caracteres el noveno carácter en adelante no se escibirá.

El módulo VDIP pose los pines RTS/CTS (Request To Send y Clear To Send) para el control de flujo de la comunicación UART. Estos es, el pin RTS es empleado por el módulo VDIP1 para indicar cuando no es capaz de recibir más datos, por ello, mediante Arduino es necesario leer el estado de dicha línea de comunicación. Si la línea está en 'Alto', es decir que está a tensión, entonces el VDIP1 es capaz de leer los datos recibidos. Por ejemplo, cuando el circuito integrado que recibe la información no tiene tiempo suficiente para asimilar esta, momento en el que entra en juego el control de flujo, informa al emisor de que detenga el envío de datos.

En la fase de prueba se conectaron ambos pines de control de flujo entre sí en el integrado. Para este caso, si coincide en el tiempo la indisponibilidad del módulo para leer y el envío de datos desde Arduino, estos datos no se almacenarían.

Se concluyó que el control del flujo es vital si se quiere una comunicación satisfactoria, evitar la pérdida de datos y optimizar el tiempo dedicado al USB.

Arduino no provee del uso del control de flujo en su librería, por lo tanto, es necesario leer el estado de la línea RTS cada vez que se envíen datos. La línea CTS es prescindible dado que es la que emplea el VDIP1 para saber si puede enviar datos al Arduino. Como el uso que se da de este integrado es solo de escritura no es necesario introducir dicha línea. A pesar de ello se introducirá en el circuito debido a que durante la fase de experimentación es necesario leer las respuestas que envía el módulo cada vez que se escribe en él.

En el código final se prescinde de la lectura, se trata de una pérdida de tiempo crítica para la precisión del cálculo de energía como se explica más adelante.

6 Visualización de datos

La visualización de datos es una característica que todos los aparatos de medida han de tener. En este PFG se analizaron dos posibilidades: una pantalla alfanumérica o una pantalla gráfica.

6.1 Pantalla alfanumérica

En primer lugar fue la opción que se barajeó, sin embargo, las pantallas alfanuméricas, como la de la figura 26, carecían de espacio suficiente para mostrar todos los parámetros que se podían calcular mediante software en la misma pantalla.

Figura 26: Imagen del LCD alfanumérico de 16x2 caracteres

Generalmente, todas las pantallas de este tipo que se encontraron en el mercado eran de 16x2, es decir, 16 caracteres por línea y dos líneas.

La principal ventaja de estos, es su ahorro de energía, sin embargo, para la aplicación de este PFG no es necesario que dicha pantalla luzca todo el tiempo. Dado que se empleará como registrador de parámetros eléctricos del consumo eléctrico del edificio de esta escuela de ingeniería, no existe necesidad de su continuo funcionamiento.

En la página principal de Arduino se puede encontrar una librería que permite controlar este tipo de pantallas facilitando su uso [14]. Según dicha página solo las pantallas cuyo controlador sea compatible con el HD44780 de Hitachi pueden ser empleados con esta librería, aunque también sirven controladores compatibles como el SPLC780D. Este último controlador es el que dispone la pantalla mostrada en la figura 27, la cual fue la que en un primer momento se iba a emplear [15].

6.2 Pantalla LCD gráfica

La pantalla de cristal líquido elegida es una pantalla de tipo gráfica que permite mostrar 8x21 caracteres o 4x10 caracteres de mayor tamaño, más que suficiente para mostrar los datos calculados de la red. Estas pantallas se diferencian de las anteriores

principalmente en que el número de bits es muy superior. En el caso de la pantalla elegida esta es de 128x64 bits y se muestra en la figura 27.

Figura 27: LCD DOGM128S-6

Las principales características del LCD DOGM128S-6 se muestran en la tabla 10 [16].

Características principales LCD gráfico	
Tipo de comunicación	SPI
Tensión de operación/alimentación	3,3V/3V
Píxeles	128X64
Consumo retroiluminado	[5,40] mA

Tabla 10: Características principales del LCD empleado

La alimentación empleada para esta pantalla es de 3.3V y se suministrará por el propio pin de alimentación del Arduino.

La transferencia de datos es unidireccional lo que significa que solo se pueden enviar datos al display. Los niveles de tensión máximos son compatibles con el Arduino del que se dispone, por lo que no es necesario emplear buffers de tensión para adecuar estas tensiones. La tensión de operación del display es de 3.3V.

Para comunicarnos con esta pantalla se empleará una librería existente que permite establecer los parámetros del protocolo de comunicación SPI (Serial Peripheral Interface) y hacerlo compatible con la pantalla en cuestión. Dicha librería proviene de una fuente de código abierto disponible en la página de 'Universal Graphics Library for 8 bit Embedded Systems' [17].

Esta pantalla requiere de un retroiluminado de LEDs para poder visualizar los datos adecuadamente. El mayor consumo se produce debido a esta iluminación y se regula mediante potenciómetro, sin embargo, se ha preferido emplear un valor de resistencia fija que determina una iluminación determinada y suficiente. El retroiluminado se ha escogido de color blanco [16].

Para seleccionar el tipo de cálculo, Aron o tres vatímetros, se ha implementado un menú para seleccionar dicho método así como unos pulsadores asociados. Una imagen de este menú se muestra en la figura 28.

Figura 28: Menú de selección del método de cálculo

Para crear el código de este menú se ha recurrido a un ejemplo interno de la librería anteriormente mencionada y se ha adaptado al interés de este trabajo.

Una vez seleccionado el método, existe un botón que activa una interrupción interna en el código de Arduino. El uso de las interrupciones está destinado a evitar la lectura de un pin de entrada cada cierto tiempo para saber si el usuario ha pulsado el botón. En el momento que se pulsa, la interrupción se activa y el código se detiene para atender a la función en que deriva dicha interrupción. La declaración de la interrupción es como sigue a continuación:

```
attachInterrupt(PinActivar,LCDActivado,RISING);
```

El primer valor introducido en la sentencia corresponde al pin donde se produce la interrupción, el segundo es la función a la que llama cuando la interrupción se produce. El tercer parámetro indica que la interrupción se produce cuando el estado del pin pasa de un valor de 'LOW', es decir 0V, a un valor de 'HIGH', es decir, a la tensión superior al límite inferior en el que el Arduino entiende que está a tensión.

Un problema derivado de estas interrupciones es que cuando se pulsa el botón se produce un efecto conocido como 'debouncing' y que produce el efecto de la figura 29.

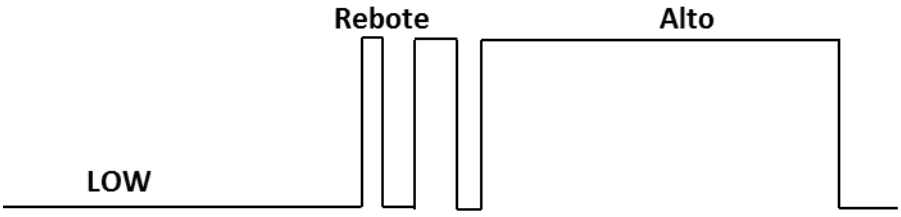


Figura 29: Efecto del rebote al pulsar

Cuando esto se produce la interrupción no es activada, por lo tanto es necesario evitar este efecto. La única forma de eliminar este efecto para el caso de una interrupción es mediante hardware, empleando un circuito paso bajo como se muestra en la figura 30.

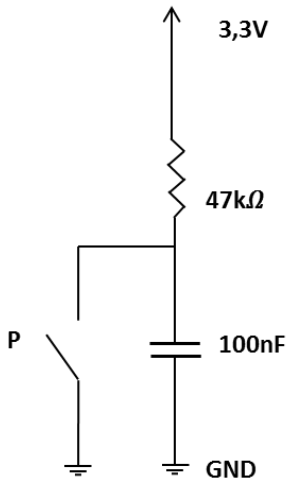


Figura 30: Circuito anti rebote

La constante de tiempo ha de ser lo suficientemente baja para durante el tiempo que se pulsa el botón el valor de tensión llegue a su valor mínimo de 'HIGH'. Este valor para el Arduino Due es de

En el circuito de la figura 30 la constante de tiempo es de 5ms aproximadamente, tiempo en el que la tensión que llega al pin del Arduino es superior a 2V. Por tanto, en menos de 5ms tras pulsar la interrupción se activa. Esa constante de tiempo era necesario que fuera muy inferior al tiempo que se emplea en pulsar el botón.

Dentro de la función de la interrupción, en inglés llamada Interrupt Service Routine (ISR), se almacena en una variable de tipo volátil el valor del tiempo leído del contador de Arduino. A continuación se muestra la parte del código de la función en cuestión:

```
void LCDActivado() //FUNCIÓN DE LA INTERRUPCION PARA ACTIVAR PANTALLA LCD MIENTRAS EL ARDUINO FUNCIONA
{
 Inicio=micros();
 LCDActivar=1;
}
```

Ambas variables se deben definir como volátiles. Este tipo de variable ayuda al software a entender que se trata de una variable cuyo valor puede cambiar en cualquier momento, por tanto, todas las variables modificadas en una ISR han de ser volátiles. Mediante dicha variable se conoce el momento en el que se activó dicho botón y mediante una variable 'boolean', llamada LCDActivar, de verdadero o falso, se consigue entrar en una sentencia 'if' que muestra los valores en el LCD. Esto se puede observar en el código a continuación:

```
if(LCDActivar==1){
   if((millis()-Inicio/1000) <= 60000){
      LCDValores(N);
   }
   if((millis()-Inicio/1000) >= 60001)
```

```
  {
   BorrarPantalla();
   LCDActivar=0;
  }
 }
```

En este código, cuando el tiempo transcurrido es de 60001µs se borra la pantalla LCD. Por tanto, siempre que se quiera visualizar los datos se ha de pulsar el botón correspondiente.

La función LCDValores(), es una función que permite mostrar por pantalla los valores de los parámetros calculados más recientemente. El valor N es el tamaño de los vectores de los distintos parámetros y al enviar el valor de N se muestran únicamente aquellos parámetros calculados en último lugar.

Se puede apreciar que el número de dígitos depende del parámetro. Este número de dígitos se eligió en base a la precisión y al rango de valores de dichos parámetros. Por ejemplo, para el factor de potencia se eligieron 3 valores decimales puesto que eran necesarios para determinar de forma más precisa dicho valor.

El tiempo de actualización de estos valores depende del tiempo de cálculo y muestreo, así como del tiempo dedicado a escribir los datos en el USB. En la sección de experimentos se mostrará cómo queda finalmente los tiempos de actualización de datos en pantalla.

7 Circuitos impresos y esquemas

Para diseñar los circuitos impresos que forman el conjuntos de la circuitería de este proyecto sea ha empleado el programa libre Fritzing. Este programa permite crear diseños de circuitos impresos a tamaño real para después fabricarlos.

En la figura 31 y 32 se muestran los circuitos impresos “espejo” del reloj y puerto USB y del LCD junto con los pulsadores.

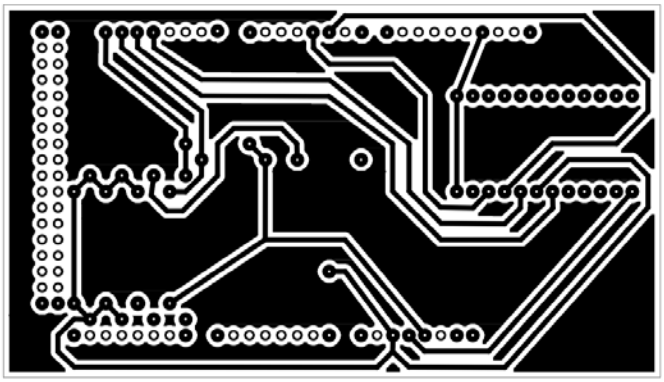


Figura 31: Circuito impreso de los componentes del RTC y puerto USB

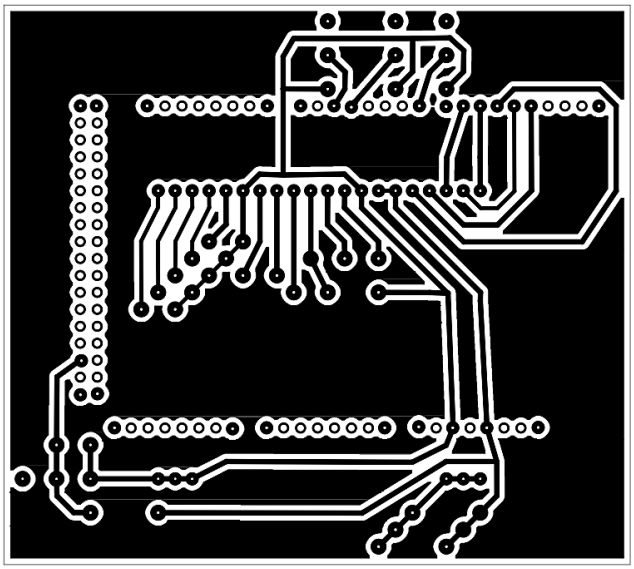


Figura 32: circuito impreso del LCD y sus pulsadores

Al usar el circuito impreso "espejo" se consigue que los circuitos estén en la parte posterior de la placa y los componentes en la superior. Las resistencias y condensadores se situarán por encima o por debajo según convenga.

Los pines del Arduino se puede ver en varias de las placas, la intención de situar dichos pines era la de situar las placas una tras otra encima del propio Arduino y de esta manera reducir al máximo el espacio, en la figura 33 se ve la placa de la figura 33 con el LCD instalado y en la parte inferior la placa con el USB.

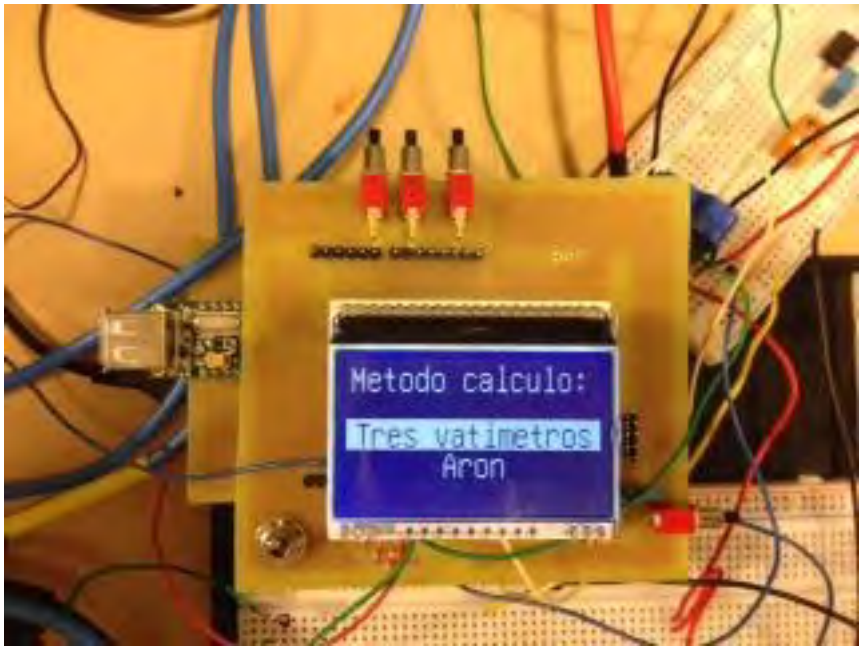


Figura 33: Imagen de las placas diseñadas del LCD y del USB

Entre las consideraciones tomadas en cuenta durante el diseño están las siguientes:

- Búsqueda de chaflanes a 45º de las vías de conexión.

- Maximizar el grueso de las conexiones y pines.
- Cubrir el circuito impreso con cobre para evitar exceso de tiempo en ácido.

Los circuitos impresos anteriores derivan del esquema del anexo I de la parte IV de este documento.

La placa de los circuitos de acondicionamiento de señales y la alimentación de los componentes de la circuitería de las fases de tensión se situaron conjuntamente en la misma placa como se puede observar en la figura 34.

De la misma manera que el anterior este circuito impreso deriva de los esquemas completos mostrados en los anexos III para el de tensión y la alimentación y el II para el circuito de corriente.

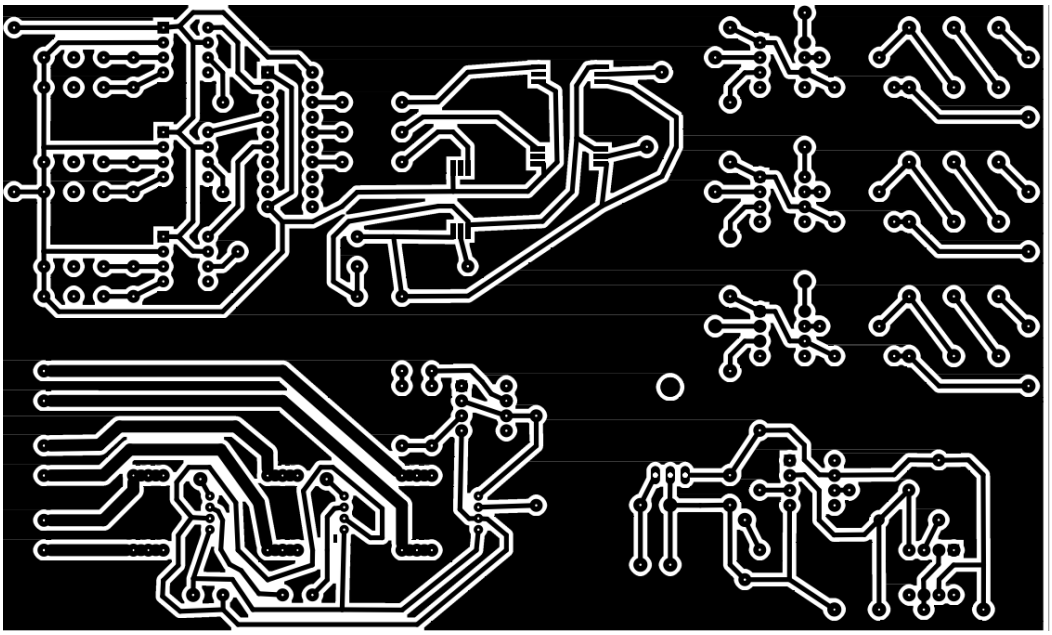


Figura 34: Circuito impreso de acondicionamiento de señales y alimentación

8 Procesamiento de datos

En este apartado se describen los métodos seleccionados para procesador los datos y las alternativas barajeadas durante la consecución de trabajo.

8.1 Alternativas

8.1.1 Circuito integrado de medida de energía polifásico

Se descubrió un circuito integrado de gran precisión en los cálculos de los diferentes parámetros eléctricos. Este es el ADE7758 [18], permite muestrear las señales acondicionadas previamente a sus rangos admisibles de tensión.

La parte anteriormente explicada de acondicionamiento de las señales de corriente y tensión se puede acoplar a este integrado. La ventaja de este método es la precisión que aporta y su diminuto tamaño, todas sus características se pueden apreciar en la tabla 11.

Características principales ADE7758		
Tipo de comunicación		SPI
Tensión de operación/alimentación		5V
Tipos de red		4 hilos/ 3 hilos
Consumo		70mW
Precisión	Energía activa por fase	0,10%
	Intensidad RMS	0,50%
	Tensión RMS	0,50%

Tabla 11: Características principales del ADE7758

A pesar de ser un aparato de grandísima precisión, con posibilidades de calibración de errores de ángulo y módulo en los circuitos de acondicionamiento de cada fase, no pudo emplearse debido al pequeño tamaño del integrado. Además presente la posibilidad de calibrar los errores de offset de los cálculos RMS que realiza internamente. El encapsulado disponible en las principales empresas distribuidoras de componentes electrónicos era el de MLP y no se disponían de métodos suficientes para conectar sus pines, en la figura 35 se aprecia este hecho.

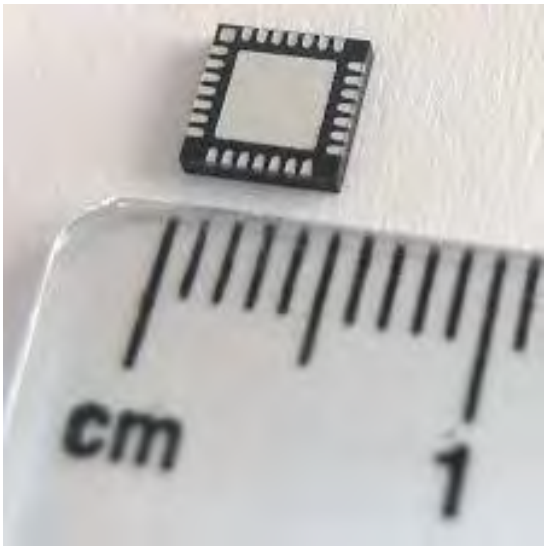


Figura 35: Encapsulado MLP

Para emplear dicho integrado es necesario utilizar adaptadores de niveles lógicos puesto que el Arduino Due opera a 3.3V y el ADE7758 a 5V.

8.1.2 Arduino

Finalmente se opta por emplear el Arduino para procesar y muestrear las señales. Para muestrearlas se emplean dos métodos según la señal:

- Señal procedente de las tensiones: Se emplea los tres contadores de los que dispone el Arduino Due para contar los pulsos emitidos por el optoacoplador. Es decir, el método de medida es de un frecuencímetro.
- Señal procedente de las corrientes. Para muestrear estas señales procedentes de los transductores de corriente, se emplean los pines analógicos que muestrean mediante el conversor analógico-digital del Arduino.

Para implementar los cálculos y el esquema general de un medidor de energía se ha recurrido a código del proyecto de Open Energy Monitor [1]. La programación del dispositivo se ha basado en el código de este proyecto de código abierto y se han introducido modificaciones sustanciales tales como el método de cálculo de Aron. También se ha intentado optimizar el filtrado del valor medio realizado por dicho código y además se ha llevado a cabo un filtro paso alto que mejora aún más las medidas tomadas filtrando el ruido.

Una de las principales diferencias entre este proyecto de Open Energy Monitor es el prescindir de un transformador de tensión que introduce errores muy elevados a medida que nos alejamos de su tensión nominal. Además, en lugar de emplear los pines analógicos del Arduino para muestrear la tensión, se ha empleado, como ya se ha comentado, los contadores del Arduino muestreando indirectamente la tensión a partir de la frecuencia del VCO.

Estas diferencias sustanciales han hecho posible mejorar la precisión del dispositivo y poder tomar medidas de potencia, tensión y energía para un rango de valores de tensión muy elevado.

8.2 Medida de frecuencia

Basándose en la hoja de especificaciones del microcontrolador del Arduino Due [19], en la sección 37, se programó los tres contadores que posee el microcontrolador para contar los pulsos durante el tiempo determinado anteriormente. Para ello el IDE de Arduino dispone de algunas funciones que permiten modificar los registros de los contadores.

```
// Activa el contador a través del Power Management Controller (PMC)
pmc_set_writeprotect(false);  //Desactivación de la protección de este registro

pmc_enable_periph_clk (ID_TC0);  //activación de los tres contadores
pmc_enable_periph_clk (ID_TC3);
pmc_enable_periph_clk (ID_TC8);

TC_Configure(TC0,0,TC_CMR_EEVTEDG_RISING | TC_CMR_TCCLKS_XC0 ); //pin 22 equivale a tensión R o RS
TC_Configure(TC1,0,TC_CMR_EEVTEDG_RISING | TC_CMR_TCCLKS_XC0 ); //pin analógico 3 equivale a tensión S
TC_Configure(TC2,2,TC_CMR_EEVTEDG_RISING | TC_CMR_TCCLKS_XC2 ); //pin 30 equivale a tensión T o TS
```

Los que se consigue con estas sentencias es programar cada uno de los contadores del Arduino Due para que cuente el número de pulsos positivos producidos en el pin donde se introduce su señal externa de reloj. En la tabla 12 se distribuyen las nueve posibles entradas de reloj externas de las cuales usaremos solo tres y cada una de ellas ha de estar asociada a un contador diferente [20].

Contador	*Canal*	*Señal externa*	*Pin asociado*
TC0	***0***	***TCLK0***	***Pin digital 22***
TC0	1	TCLK1	Pin analógico 5
TC0	2	TCLK2	Pin digital 31
TC1	***0***	***TCLK3***	***Pin analógico 3***
TC1	1	TCLK4	Pin analógico 2
TC1	2	TCLK5	DAC1
TC2	0	TCLK6	/
TC2	1	TCLK7	LED 'RX'
TC2	***2***	***TCLK8***	***Pin digital 30***

Tabla 12: Listado de señales de reloj externas asociadas a los contadores

Como solo existen tres contadores, las nueve fuentes externas de reloj están asociadas por paquetes de tres, es decir, solo se pueden leer tres señales externas al mismo tiempo.

Gracias a la independencia entre sus contadores podremos contar el número de pulsos en el mismo periodo de esta manera tendremos los valores de tensión correspondientes al mismo instante. Esto es una de las grandes ventajas de este método al no haber apenas desfase de tiempo entre muestras de tensión. No ocurre lo mismo con las muestras de corrientes, estas son desfasadas entre sí apenas 1µs, que es lo que tarda en muestrear el conversor analógico-digital. La parte del código en la que se produce el muestreo se presenta a continuación:

```
//MEDIDA DE TENSIÓN
TC_Start(TC0,0);
TC_Start(TC1,0);
TC_Start(TC2,2);
delayMicroseconds(150);
TC_Stop(TC0,0);
TC_Stop(TC1,0);
TC_Stop(TC2,2);

//MEDIDA DE CORRIENTE
muestra_I_fase_R = analogRead(A0); //TARDA 1 µs
muestra_I_fase_S = analogRead(A1);
muestra_I_fase_T = analogRead(A2);

muestra_V_fase_R = TC_ReadCV(TC0,0);
muestra_V_fase_S = TC_ReadCV(TC1,0);
muestra_V_fase_T = TC_ReadCV(TC2,2);
```

En primer lugar, se inician los tres contadores asi simultáneamente con la función TC_Start y especificando el contador y el canal que conecta con el pin correspondiente. Por ejemplo, la primera sentencia activa el contador TC0 y lo conecta con la señal entrante en el canal 0, lo que corresponde con el pin digital 22 como se observa en la tabla 12.

8.3 Alimentación

Para la alimentación del Arduino es necesario conocer el consumo de la circuitería antes de comprar la batería adecuada.

Esta batería se ha de diseñar de la manera más óptima intentando que tanto la batería de los circuitos de tensión como está se descarguen al mismo tiempo. Esta forma de dimensionar las baterías se puede entender como una optimización, buscando que el aparato tenga la misma autonomía para ambas alimentaciones. De esta manera, y como ambas baterías son recargables, en el momento en que la batería esté cerca de su vaciado será necesario recargar ambas a la vez sabiendo que están cerca de su límite de descarga.

El usar dos alimentaciones distintas puede llegar a ser incómodo, sin embargo, debido a la necesidad de realizar una referencia flotante en el circuito de acondicionamiento de tensión de la parte no aislada, ambas son fundamentales.

8.4 Cálculo de parámetros

A continuación se explican los métodos de cálculo implementados en el código según la disponibilidad del neutro. Los cálculos son indiferentes de la tipología de la red, es decir, son válidos tanto para redes en configuración δ como Y.

8.4.1 Cálculos de Aron

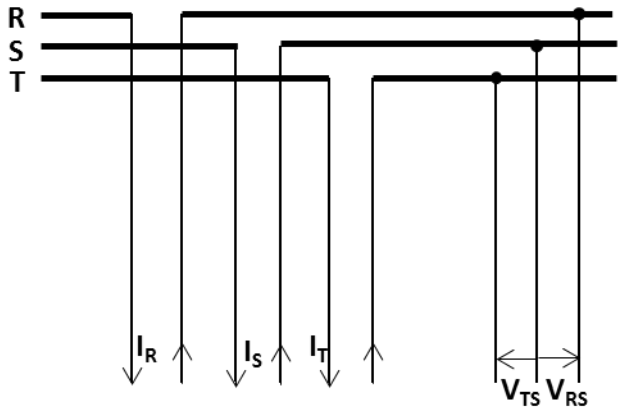


Figura 36: Red δ o Y sin neutro accesible

El cálculo de Aron está diseñado para medir potencias en redes cuyo neutro no es accesible o sencillamente, la red, no dispone de él, véase la figura 36. En ella se han señalado los valores muestreados.

Los cálculos realizados no se realizan sobre variables continuas dado que se dispone únicamente de muestras. Por tanto, los cálculos implementados son sobre variables discretas. Todos los cálculos implementados para este método se pueden apreciar en la tabla 13.

Parámetro	***Cálculo***
RMS	$\mathbf{X_{RMS}} = \sqrt{\mathbf{\frac{1}{n}} \cdot \sum_{\mathbf{1}}^{\mathbf{n}} (\mathbf{X_i})^2}$
P_{III}	$P_{III} = \frac{1}{n} \cdot \sum_{1}^{n} (V_{iR} - V_{iS}) \cdot I_{iR} + (V_{iT} - V_{iS}) \cdot I_{iT}$
Q_{III}	$Q_{III} = \left[\frac{1}{n} \cdot \sum_{1}^{n} \left((V_{iT} - V_{iS}) \cdot I_{iT} - (V_{iR} - V_{iS}) \cdot I_{iR} \right) \right] \cdot \sqrt{3}$
S_{III}	(1) $S_{III} = \sqrt{{P_{III}}^2 + {Q_{III}}^2}$ ó (2) $S_{III} = \sqrt{3} \cdot U_{ff} \cdot I_f$
cos ϕ	$\mathbf{\cos \varphi = \frac{P_{III}}{S_{III}}}$
$\mathbf{E_{act}}$	$\boldsymbol{E_{act} = P_{III} \cdot \Delta t}$

Tabla 13: Cálculos realizados para obtener los diferentes parámetros

En la tabla 13, aquellos cálculos señalados en negrita son comunes para ambos métodos. El método de Aron puede ser el más preciso de los dos, esto se debe a la forma de cálculo de la reactiva y al diseño del software para medir tensión compuesta.

Dado que la fórmula de la reactiva es una fórmula directa de cálculo se evitan errores introducidos al obtener dicho valor indirectamente a través de la ecuación 15.

$$Q_{III} = \sqrt{\left(\sqrt{3} \cdot U_{ff} \cdot I_f\right)^2 + {P_{III}}^2} \qquad (15)$$

En cuanto al cálculo de la potencia aparente se ha de determinar de manera empírica calibrando el aparato con otro de buena precisión. Para determinar cuál de ellas es más precisa, se ha de conocer primero la incertidumbre de la potencia activa y reactiva y de la incertidumbre de la tensión compuesta e intensidad RMS. El valor de la incertidumbre del cálculo de la potencia aparente con el método (1) se aprecia en la ecuación 16, mientras que el método (2) en la ecuación 17.

$$\alpha\left(S_{III}(1)\right) = \frac{P_{III} \cdot \alpha(P_{III}) + Q_{III} \cdot \alpha(Q_{III})}{\sqrt{{P_{III}}^2 + {Q_{III}}^2}} \qquad (16)$$

$$\alpha\left(S_{III}(2)\right) = \sqrt{3} \cdot U_{ff} \cdot \alpha\left(I_f\right) + \sqrt{3} \cdot I_f \cdot \alpha\left(U_{ff}\right) \qquad (17)$$

Otro de los motivos por los cuales el método de Aron puede ser más preciso que el método de los Tres Vatímetros, es que el hardware de la parte de acondicionamiento de tensión se ha diseñado para la máxima tensión. Esta tensión es la compuesta y dado que el método de los Tres Vatímetros mide tensión simple, el error relativo es superior.

A pesar de que el cálculo sea más preciso que el de los Tres Vatímetros, este método tiene un error asociado a los desequilibrios en las tensiones. Se trata de un método que muy preciso para sistema de tensiones equilibradas. Por lo tanto, a priori no se puede determinar qué método de los dos es el más preciso, a menos que se haga el ensayo de calibración para los dos métodos.

8.4.2 Cálculos de los Tres Vatímetros

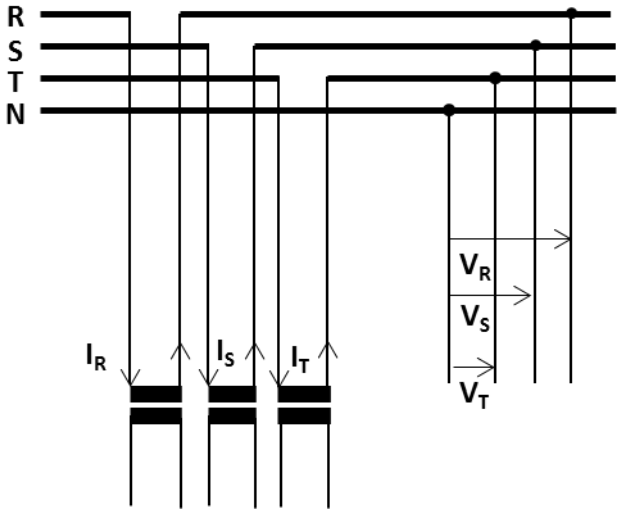


Figura 37: Red δ o Y con neutro accesible

El método de los Tres Vatímetros está diseñado para redes con neutro accesible como se aprecia en la figura 37. En ella se han señalado los valores muestreados.

Los cálculos de los parámetros implementado en este método de muestran en la tabla 14. Los parámetros comunes a ambos métodos se señalaron en negrita en el apartado anterior.

Parámetro	***Cálculo***
P_I	$P_f = \frac{1}{n} \cdot \sum_{1}^{n} V_i \cdot I_i$
P_{III}	$P_{III} = P_{fR} + P_{fS} + P_{fT}$
S_{III}	$S_{III} = U_{fR} \cdot I_{fR} + U_{fS} \cdot I_{fS} + U_{fT} \cdot I_{fT}$
Q_{III}	$Q_{III} = \sqrt{{S_{III}}^2 - {P_{III}}^2}$

Tabla 14: Cálculos implementados para el método de los Tres Vatímetros.

La ventaja de este método de cálculo es que sirve para redes desequilibradas en tensiones e intensidades. Por otro lado, el cálculo de la potencia reactiva es un cálculo indirecto a través del cálculo previo de la potencia aparente, lo que produce errores de medida, en especial, para factores de potencia cercanos a la unidad.

Como se ha podido observar en ninguno de los métodos se mide corriente de neutro, se debe a que no aporta información para el cálculo de los parámetros. Podría servir para añadir funciones de protección diferenciales al dispositivo.

8.5 Muestreo síncrono

Una de las características que ha de tener el código implementado es el muestreo síncrono. Esto es, muestrear cuando el número de periodos totales obtenidos de la señal es un número natural, es decir, la señal empieza donde acaba.

El efecto de un muestreo asíncrono se puede explicar sencillamente con el siguiente ejemplo: el cálculo del valor RMS de la señal de la ecuación 18.

$$v(t) = 2.333 \cdot \sqrt{2} \cdot \sin(2 \cdot \pi \cdot 50 \cdot t) \qquad (18)$$

Teóricamente el valor es de 2.333V, el cálculo de dicho valor cuando el muestreo es síncrono se aprecia en la figura 38.

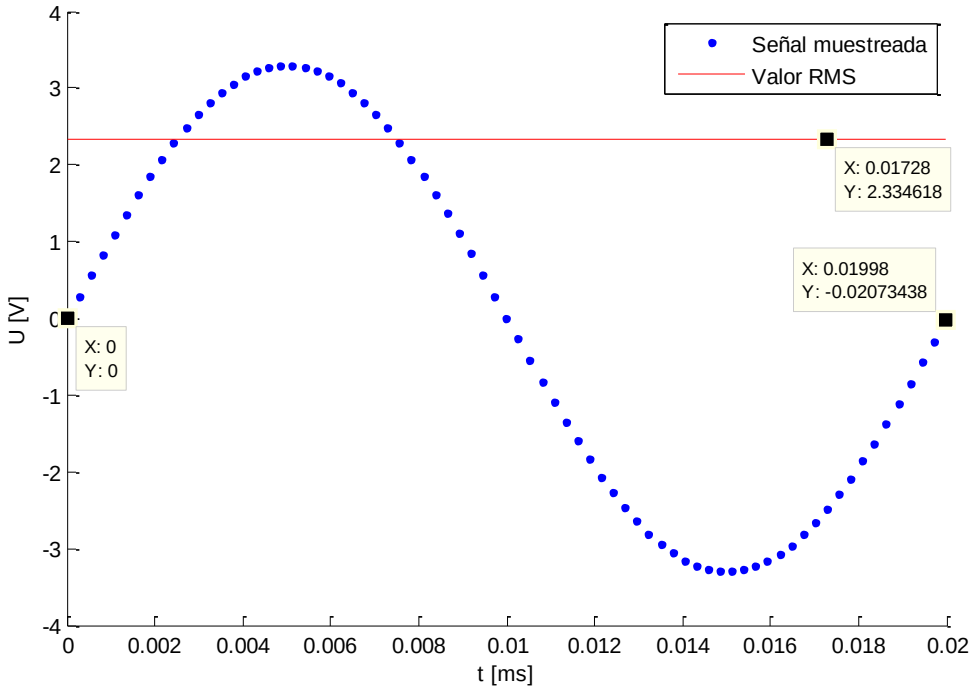


Figura 38: muestra de señal simulada y valor RMS

Como se puede ver en la figura 39 no ocurre lo mismo cuando no se ha tomado un número entero de periodos.

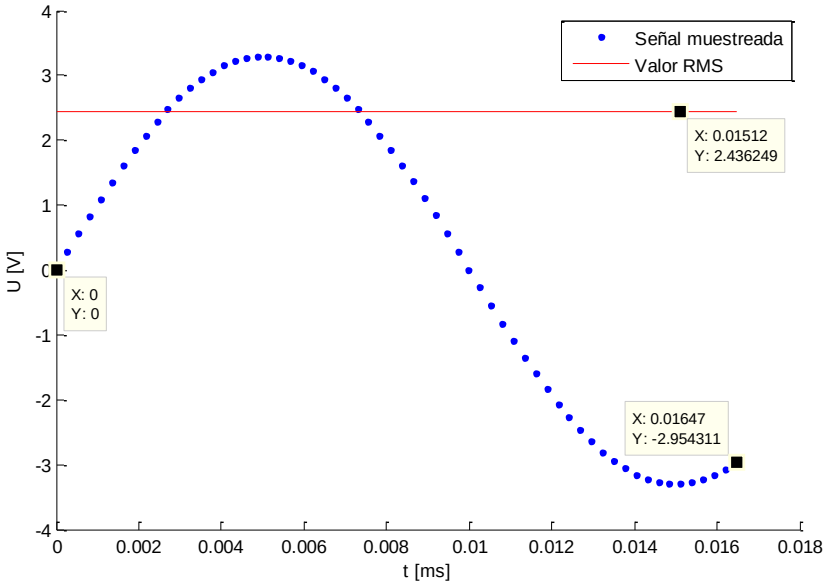


Figura 39: Cálculo de RMS con muestreo asíncrono

En la figura 39 el valor RMS tiene un error del 4.41% mientras que la de la figura 38 el error es de 0.07%.

Otro de los factores que influyen en la disminución del error cometido por el muestreo asíncrono es emplear el mayor número de periodos posibles.

Para llevar a cabo un muestreo síncrono en este trabajo se recurre a la ecuación 18.

$$Nº\ Muestras = \frac{T_{fundamental\ señal}}{T_{muestreo}} \cdot nºPeriodos\ \ muestreados \qquad (19)$$

El valor total del número de muestras para el caso a analizar se ha fijado en 2439 muestras, que para el periodo de muestreo que se tiene de 270µs, equivale a casi 33 periodos. Como se puede ver en la señal de corriente muestreada de la figura 40, la primera muestra coincide con la última.

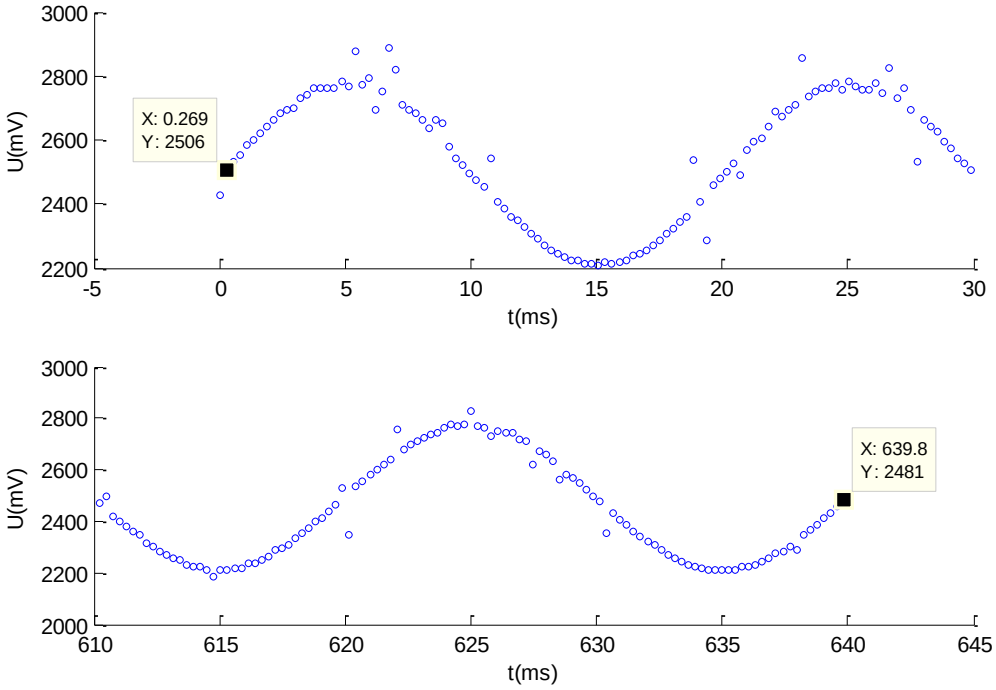


Figura 40: Principio y final de señal de corriente muestreada en pin analógico del Arduino

En la figura 40 solo se ha mostrado el principio y el final debido a que no se aprecian correctamente los 33 periodos seleccionados. Al emplear 33 periodos el error en los cálculos medios es inapreciable.

8.6 Filtros digitales y calibración

En este apartado se presentan los métodos de cálculo de las constantes de los filtros implementados en el código y la calibración necesaria para alcanzar la precisión correspondiente.

8.6.1 Filtrado del valor medio y calibración del desfase

Las entradas analógicas del Arduino encargadas de medir la tensión de salida de los transductores, así como, las entradas de los conversores de tensión a frecuencia requerían la adición de un valor medio puesto que no permitían la lectura de valores negativos.

Los cálculos realizados por software han de eliminar primero dicho valor medio, para después, llevar a cabo los cálculos pertinentes. Para eliminarlo se han analizado dos procedimientos distintos:

- Resta del valor teórico del valor medio introducido. En el caso de la frecuencia 1536kHz, mientras que en el de corriente 2.25V. Este método carece de precisión debido a que se incluyen inevitablemente los errores producidos en el Hardware. La solución a ese problema era el de medir cual era el valor medio que se introducía, sin

embargo, esto solo es válido para los circuitos de corriente dado que se podía medir que tensión se introducía exactamente al pin de referencia del transductor, también afectado por el error cometido al muestrearlo. Para el VCO el valor medio solo podía substraerse por medio de software, mediante la ecuación 19. Para ello primero sería necesario determinar dichos valores máximo y mínimos cambiantes con el valor de tensión de red.

$$f_{medio} = \frac{f_{máx} + f_{mín}}{2} \qquad (20)$$

Se trata de un método muy impreciso que añadiría errores significativos en todos los cálculos de los parámetros.

- Filtro digital. La solución óptima y más cómoda para eliminar el valor medio era la introducción de un filtro digital al código durante el muestreo de las señales. Este código se ha diseñado por analogía con la ecuación del circuito RC paso alto de la figura 41. De este filtro analógico se derivan la ecuación implementada y mostrada en la ecuación 25.

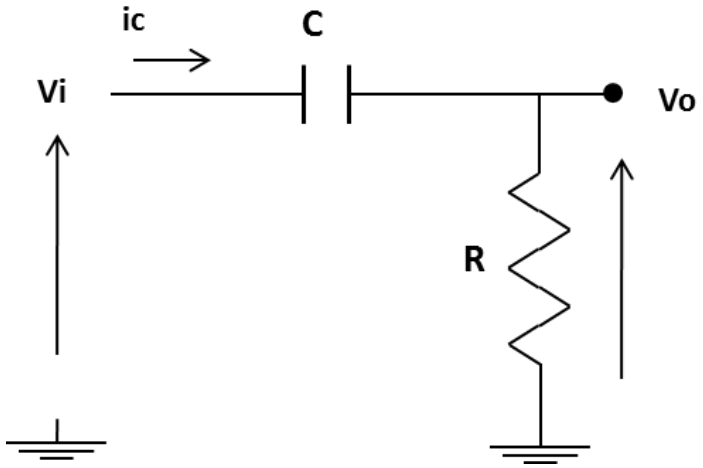


Figura 41: Filtro analógico paso alto

Los cálculos pertinentes para obtener la ecuación final se muestran secuencialmente de las ecuaciones 20 a la 25.

$$i_c = C \cdot \frac{d(v_i - v_o)}{dt} = C \cdot \left[\frac{d(v_i)}{dt} - \frac{d(v_o)}{dt}\right] \qquad (21)$$

$$v_o = i_c \cdot R = R \cdot C \cdot \left[\frac{d(v_i)}{dt} - \frac{d(v_o)}{dt}\right] \qquad (22)$$

El cálculo realizado por software de las derivadas solo se puede llevar a cabo mediante valores discretos como se muestra en la ecuación 22. Aunque se trata de una aproximación, el error cometido es pequeño gracias al periodo de muestreo tan bajo que se alcanza en este trabajo. El periodo de muestreo se obtuvo de uno de los experimentos realizados y explicados más adelante, éste es de 253µs.

$$v_o \cong R \cdot C \cdot \left[\frac{\Delta(v_i)}{\Delta t} - \frac{\Delta(v_o)}{\Delta t}\right] \qquad (23)$$

La constante de tiempo del paso alto es RC y determinará el tiempo que tarda el filtrado en eliminar el valor medio. Este tiempo interesa que sea lo más pequeño posible, sin embargo, ha de ser lo suficientemente alto como para evitar que la señal quede excesivamente desfasada.

Continuando con el desarrollo en la ecuación 22 se obtiene la ecuación 23.

$$v_o(i) = \tau \cdot \left[\frac{v_i(i) - v_i(i-1)}{\Delta t} - \frac{v_o(i) - v_o(i-1)}{\Delta t}\right] \qquad (24)$$

Y despejando…

$$v_o(i) \cdot \left[1 + \frac{\tau}{T_{muestreo}}\right] = \frac{\tau}{T_{muestreo}} \cdot [v_i(i) - v_i(i-1) + v_o(i-1)] \qquad (25)$$

Finalmente la ecuación 25 es la implementada en el software del Arduino.

$$v_o(i) = \frac{\tau}{\tau + T_{muestreo}} \cdot (v_i(i) - v_i(i-1) + v_o(i-1)) \qquad (26)$$

Para determinar el valor de la constante de tiempo se recurrió a Matlab exportando los datos muestreados por el Arduino. En la figura 42 se representan la respuesta del filtro ante una constante de tiempo de 40ms, es decir, una frecuencia de corte de 4Hz.

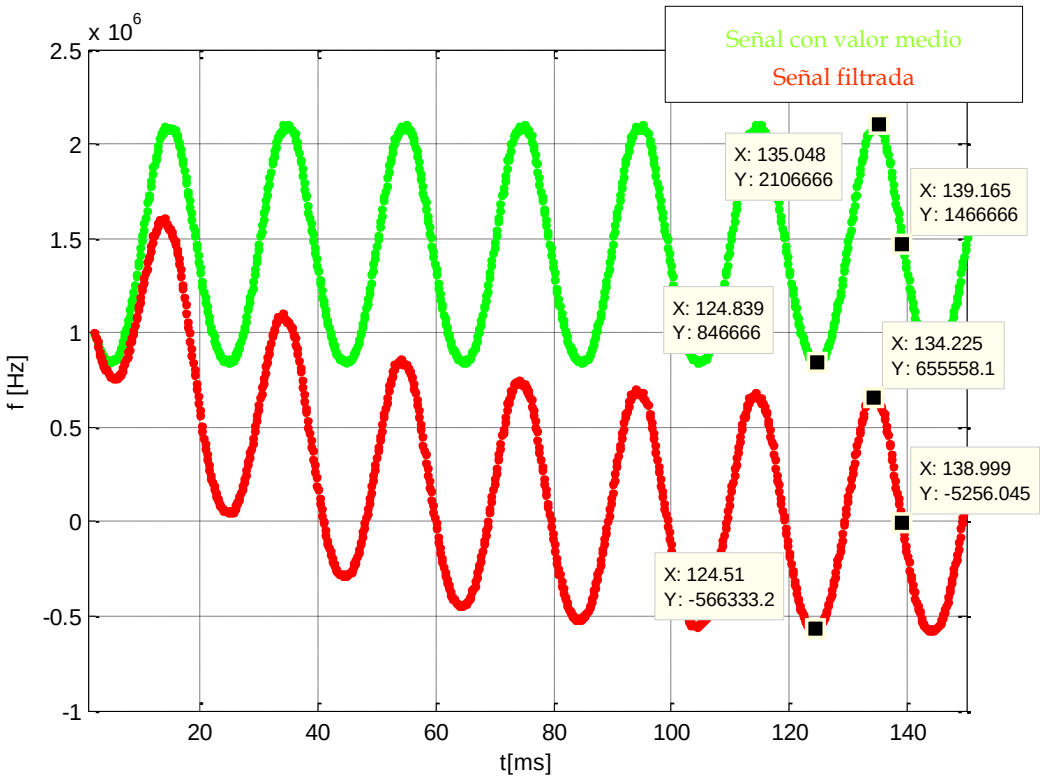


Figura 42: Respuesta del filtrado del valor medio para las muestras de frecuencia

Dos son las conclusiones que se obtienen de dicha figura, el filtro introduce un desfase que es necesario corregir y disminuye la amplitud.

La disminución de amplitud se puede corregir fácilmente calibrando la amplitud de las muestras. Este factor de calibración de la amplitud se obtiene de la ecuación 26.

$$K = \frac{2107 - 846.7}{655.6 + 566.3} = 1.0314 \qquad (27)$$

El desfase producido se ha tomado entre los puntos en los que las señales cruzan su valor medio, es decir, en la señal sin filtrar será de 1476.85 kHz y la otra será de 0Hz. Es necesario advertir que dicho desfase no solo se debe a la intrusión del filtrado, sino que también, tiene influencia lo hablado en la figura 12 de este documento. Como existe un error de cuantización estos valores no coinciden con las muestras y se han tomado valores lo más cercanos posibles. La ecuación 27 determina la calibración necesaria en º.

$$\alpha[º] = (139.165 - 138.999) \cdot \frac{360º}{20\ [ms]} = 3º \qquad (28)$$

Es decir, será necesario calibrar este desfase por medio la fórmula de la ecuación 28. Esta ecuación ha sido obtenida del código abierto de Open Energy Monitor [1].

$$v_{CAL}(i) = CAL \cdot (v_o(i) - v_o(i-1)) + v_o(i-1) \qquad (29)$$

La respuesta a esta calibración se puede ver en la figura 43, obtenida de nuevo mediante Matlab. Para esta figura se ha intentado corregir ese desfase de 3º que equivale a 0.166ms. El desfase es positivo, es decir, desplaza la tensión hacia la izquierda, por lo tanto, para corregir dicho desplazamiento el valor de CAL, de la ecuación 28, debe ser menor que la unidad.

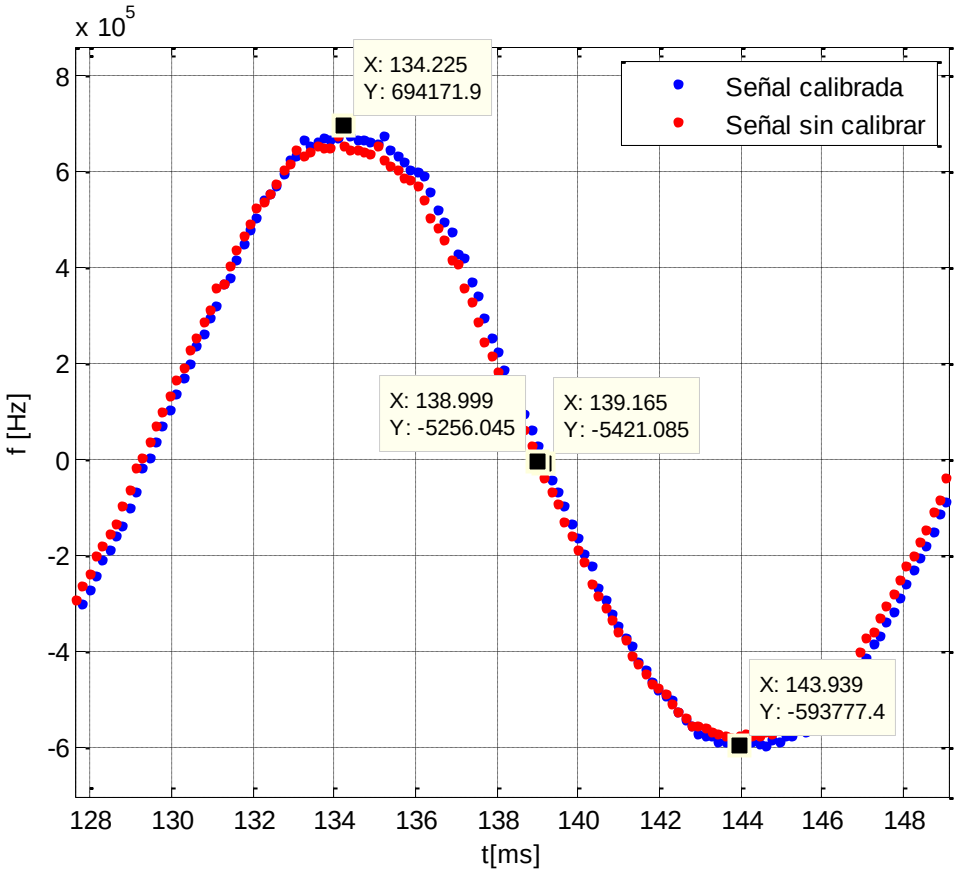


Figura 43: Respuesta de la calibración

Como se puede apreciar también se ha calibrado en amplitud. Esta calibración ha conseguido reproducir perfectamente la señal original sin ese valor medio. El desfase corregido es exactamente el mismo que introdujo el filtro y la corrección de amplitud elimina el error introducido y calculado anteriormente.

8.6.2 Filtrado del ruido

Para filtrar el ruido se introduce un filtro paso bajo cuya frecuencia de corte y respuesta temporal sean admisibles. Para valorarlo se ha procedido de la misma manera que en el apartado anterior. En primer lugar, es necesario diseñar la ecuación del filtro en forma discreta. Para ello, de nuevo, se hace una analogía con el circuito de la figura 44 que corresponde con un filtro paso bajo.

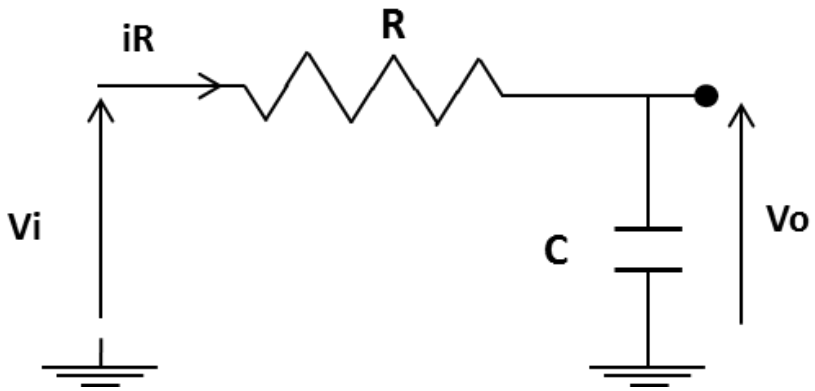


Figura 44: Circuito paso bajo de constante de tiempo RC

Desarrollando la relación entre ambas tensiones, procediendo de la misma manera que el filtro paso alto, se obtiene la ecuación implementada en el código, la ecuación 29.

$$v_o(i) = \left(v_o(i-1) \cdot \frac{\tau}{T_{muestreo}} - v_i(i-1) \right) \cdot \frac{1}{1 + \frac{\tau}{T_{muestreo}}} \qquad \textbf{(30)}$$

La respuesta de la entrada de tensión a este filtro es la que se representa en la figura 45.

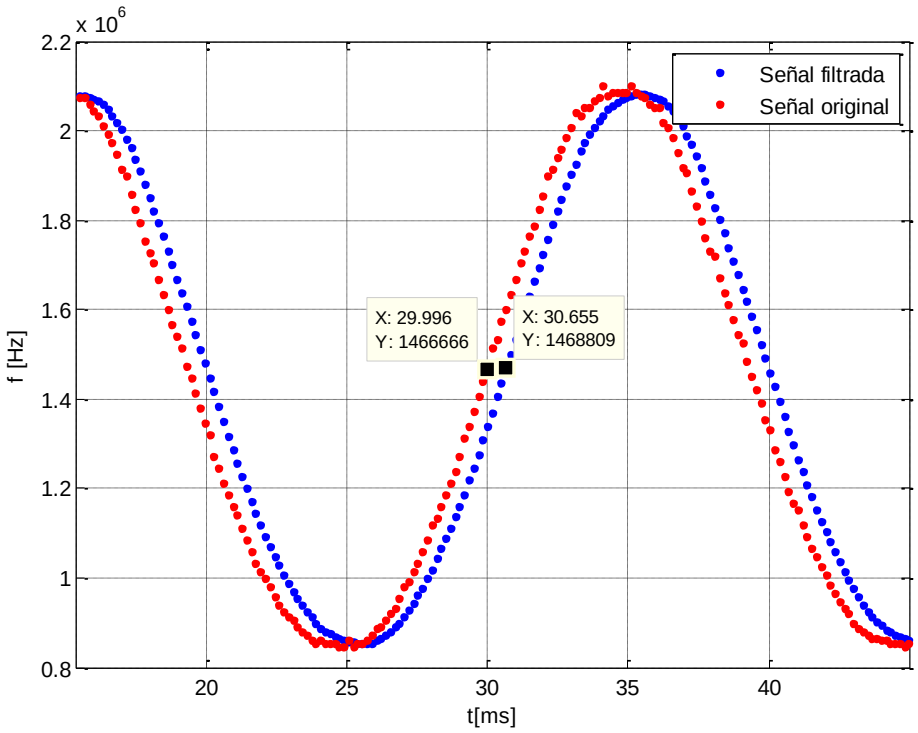


Figura 45: Respuesta de la señal de frecuencia a un filtro paso bajo

La frecuencia de corte escogida ha sido de 200Hz para eliminar el poco ruido que la señal de tensión tenía. Se puede ver la mejoría de la señal en la gráfica anterior.

De nuevo se produce un desfase que es necesario corregir mediante el código del apartado anterior. Por otro lado, la amplitud de la señal apenas se ha modificado por lo tanto no será necesario modificar el valor de calibración de la amplitud.

El desfase producido entre las señales es de -11.862º, que al sumarse al desfase del filtro anterior daría un desfase total de -8.86º. El desfase introducido es demasiado elevado respecto a la mejoría producida con respecto a la filtración del ruido.

La respuesta conjunta de ambos filtros produce un filtro paso banda entre las frecuencias de rango $\Delta f \in [4,200]$Hz. Esta señal final se aprecia en la figura 46.

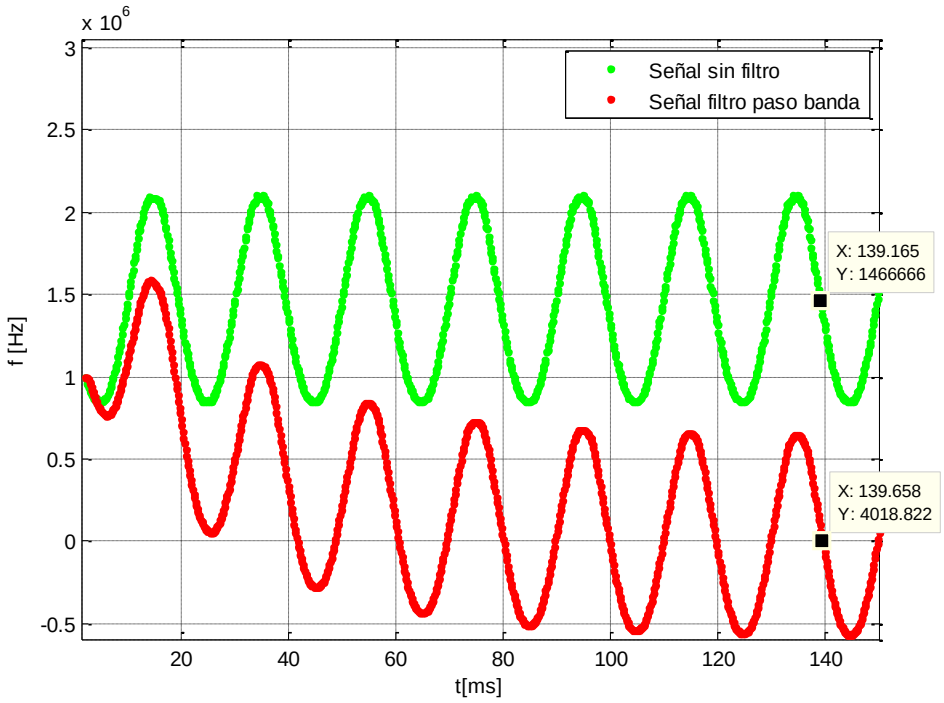


Figura 46: Respuesta del filtro paso banda en la señal de frecuencia

Como se puede ver el desfase total equivale al calculado previamente y es, según la gráfica, de -8.87º, muy parecido a los -8.86º previstos.

Aplicando la calibración angular obtenemos la respuesta mostrada en la figura 47 y calibrada respecto a la salida del filtro paso banda. Se ha mostrado las gráficas continuas para aplicar de forma más precisa la corrección.

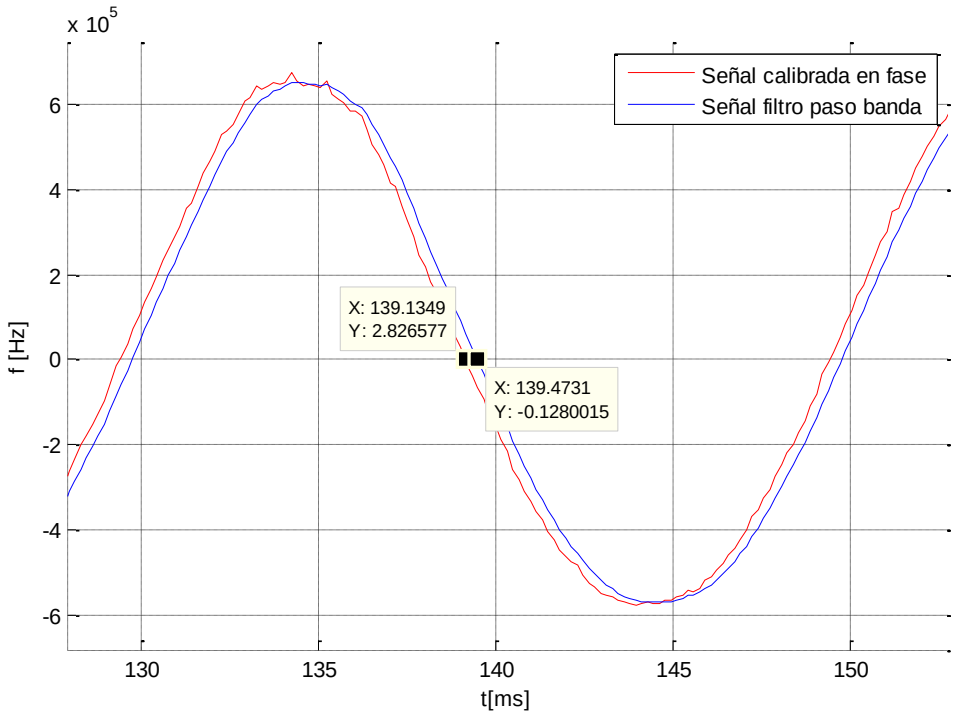


Figura 47: Respuesta del filtro, su calibración de la fase y la señal de frecuencia original

Se puede apreciar como la señal roja, que es la calibrada en fase, se ve de nuevo distorsionada por el ruido que se intentó corregir. Esto es debido a que la respuesta en frecuencia de la ecuación encargada de calibrar la fase incrementa la amplitud de los armónicos que producían el ruido. Por tanto, como conclusión, se prefiere prescindir de dicha calibración y corregir el desfase angular modificando las frecuencias de corte de manera que el desfase introducido por los filtros se compense entre sí.

Finalmente, a la señal de frecuencia leída por el Arduino se le aplicará un filtrado paso banda que elimine el valor medio y los armónicos del ruido. La respuesta elegida para la señal de frecuencia se representa en la figura 48.

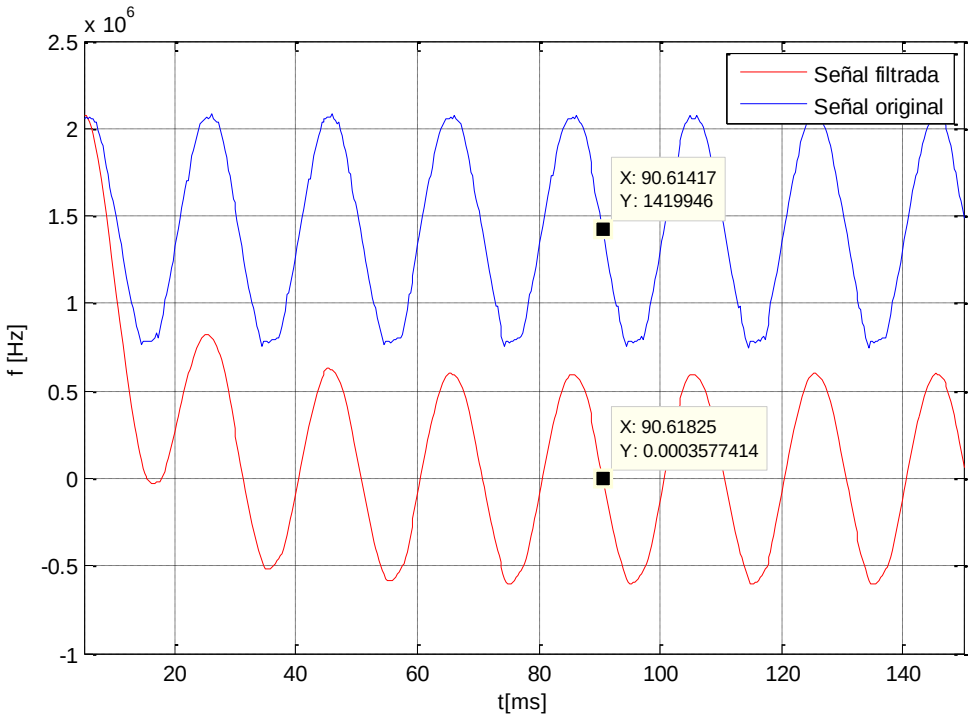


Figura 48: Respuesta del filtro paso banda a la señal de frecuencia muestreada

En la figura 48 se han mostrado los puntos en los que las señales pasan por sus correspondientes valores medios. En caso de la señal original muestreada de frecuencia esta es de 1.4199 MHz. Se puede apreciar como el desfase introducido por el filtro paso banda para una frecuencia de 50Hz como es la señal de la red introduce un desfase angular de casi 0º. Esto se ha conseguido, como ya se ha comentado, compensando el desfase introducido de ambos filtros.

El rango de frecuencias del filtro paso banda diseñado es de $\Delta f \in [16.93,221]$Hz.

La señal de corriente traducida a tensión mediante el transductor está afectada por ruido, se muestra en la figura 48.

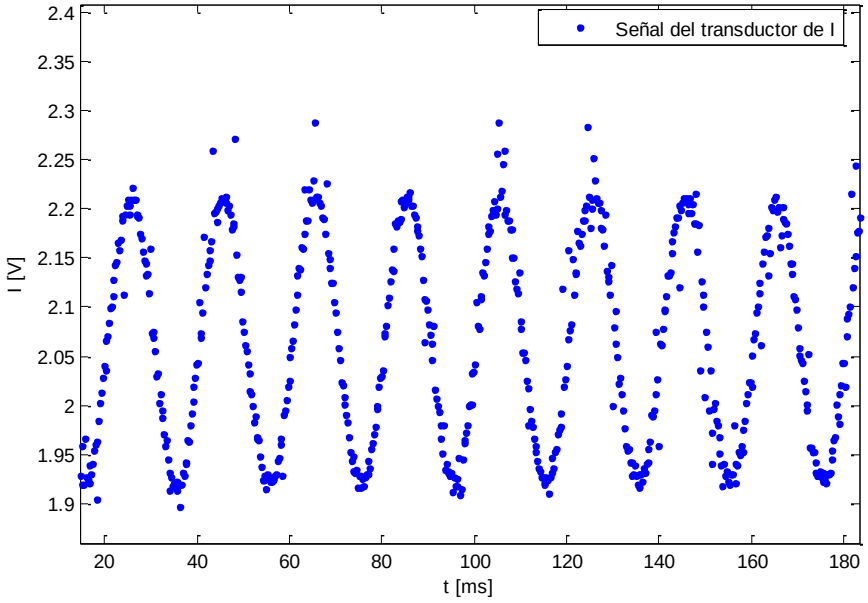


Figura 49: Señal muestreada de la señal de tensión producida por el transductor de corriente

En este caso si también será necesario aplicar el filtro paso banda como el diseñado anteriormente. En la figura 50, se representan conjuntamente la señal muestreada, la verde, la señal aplicándole el filtro paso alto, la señal roja, y finalmente, la señal azul con filtro paso banda.

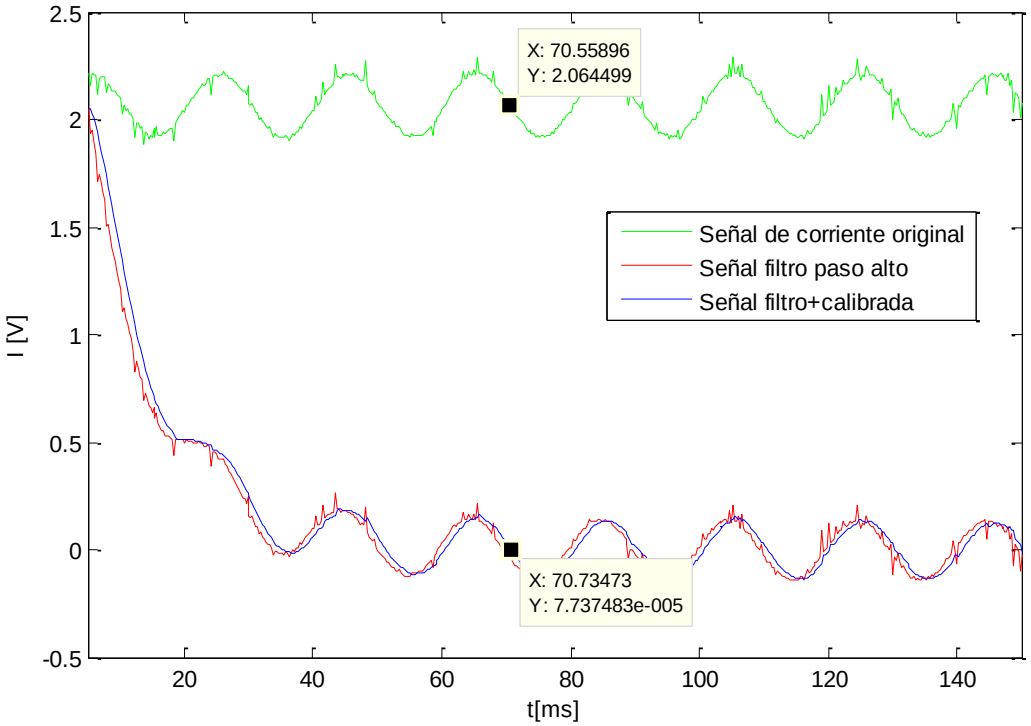


Figura 50: Respuesta conjunta de los filtros paso alto y paso banda

También en la figura 50 se muestra el desfase entre la señal original de valor medio 2.06V, calculado mediante Matlab tras sumar todas las muestras y dividir entre dicho número de muestras, y la señal con el filtro paso banda. Este desfase se muestra en la ecuación 30.

$$\alpha[º] = (70.73473 - 70.55896) \cdot \frac{360º}{20\ [ms]} = 3º\ 9'50'' \qquad (31)$$

Al ser un desfase pequeño es sencillo calibrar la señal de corriente mediante la fórmula de la ecuación 28.

En la figura 51 se representa la respuesta temporal producida por el filtro paso banda y la calibración del desfase calculado en la ecuación 30.

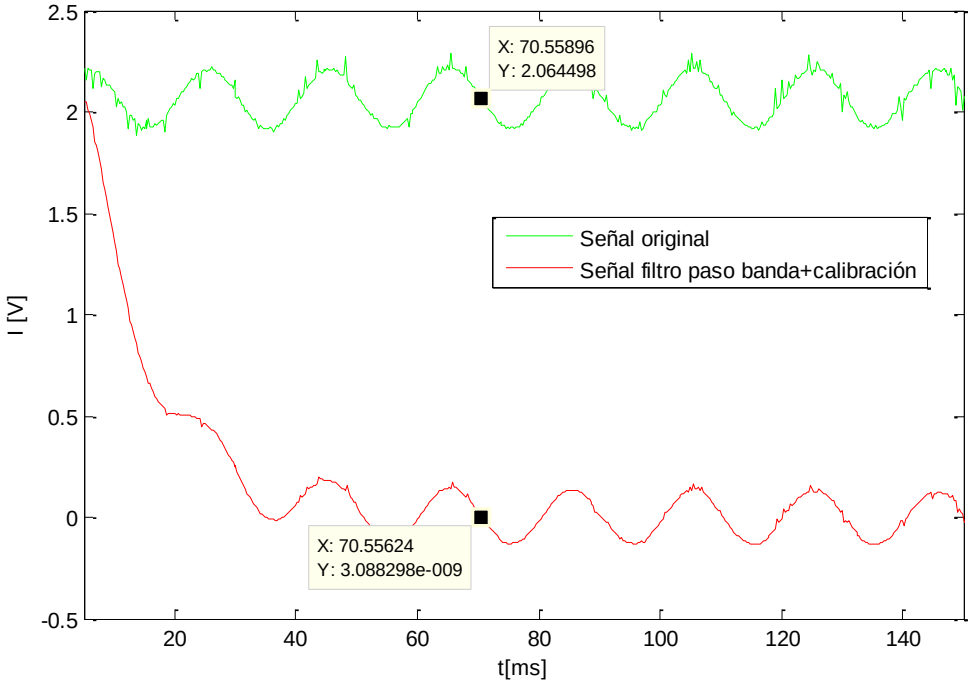


Figura 51: Respuesta de la señal muestreada del transductor filtrada y calibrada

A pesar de que la calibración aumente el módulo de los armónicos del ruido, debido al pequeño desfase introducido y, por tanto, gracias a un valor pequeño del parámetro CAL de 1.61, este ruido se ha conseguido reducir respecto a la señal original.

El rango del filtro paso banda aplicado es de Δf Є [15.9, 200] Hz.

Otra manera de conseguir el desfase deseado sin necesidad de recurrir a la calibración del ángulo, es la de modificar las frecuencias de corte de ambos filtros para que sus desfases se compensen entre sí, como ya se ha realizado para la señal muestreada de frecuencia.

La respuesta obtenida de forma empírica más satisfactoria es la de la figura 52. El rango de las frecuencias del filtro paso banda son Δf = [17.88, 200] Hz. Además a esta señal también se le ha aplicado una corrección de la amplitud de un 9.2% superior. Este valor ha sido obtenido calculando el valor pico-pico de la señal original dividido entre el valor pico-pico de la señal filtrada.

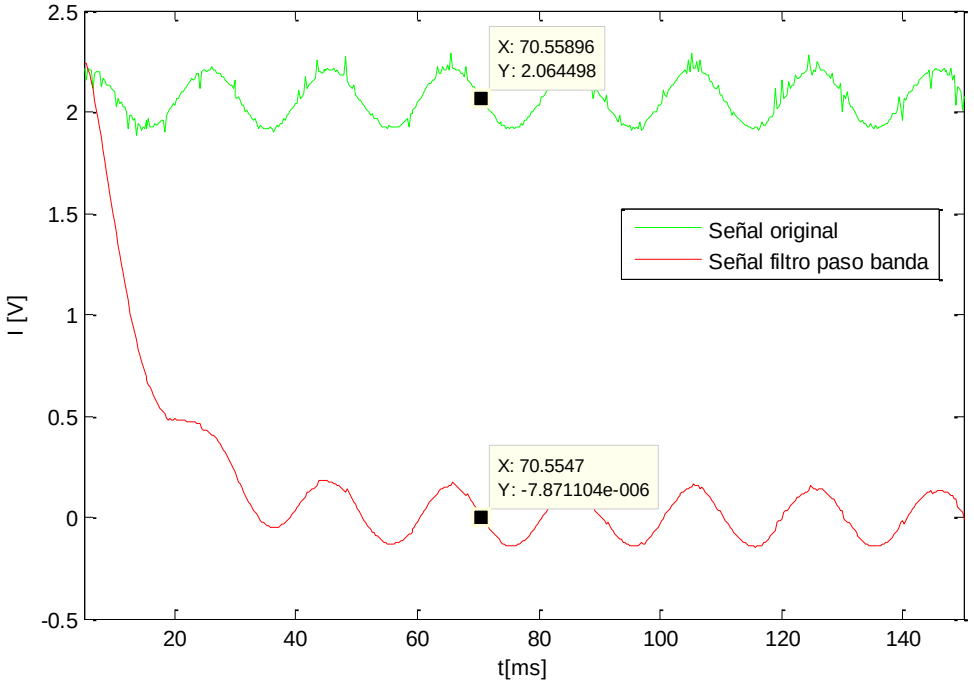


Figura 52: Respuesta de la señal muestreada del transductor con filtro paso banda

Se puede observar como se ha eliminado el ruido y además el desfase se ha corregido. Finalmente es este método el que se va aplicar en el código del Arduino. Como conclusión se deshecha el uso de la calibración de la fase debido al incremento de amplitud de los armónicos del ruido. Por tanto, el uso de esta calibración por parte del proyecto de Open Energy Monitor es dudoso cuando las señales introducidas al Arduino son señales con armónicos de alta frecuencia, este método incrementa su valor y distorsiona las medidas como se ha comprobado.

9 Especificaciones técnicas el aparato

En este apartado se presenta en la tabla 15 todas las características principales del aparato diseñado. Para comparar los objetivos fijados previamente al inicio del trabajo se puede comparar dicha tabla con la tabla 2 de la introducción.

Características		
Rango de tensiones admisibles en medida directa		20-380 V
Valor máximo admisible en medida indirecta	Tensión	Un secundaria=380V
	Intensidad	6A
Periodo de actualización de medidas		≈5s
Intensidades nominales		5A
Consumo de alimentación		ND
Almacenamiento de datos		Sí
Tipo de almacenamiento		USB

Precisión	Intensidad		≈1%
	Tensión		≈1%
	Potencia Activa		<5%
	Potencia Reactiva		<6%
	Factor de potencia		<5%
	Energía activa		ND
	Energía reactiva		No Calcula
Alimentación	AC		No
	DC		batería 9V DC y batería de 12 DC
Método de medida potencia activa	Un vatímetro	4 hilos	
		3 hilos	
	Tres vatímetros	4 hilos	✕
		3 hilos	
	Dos vatímetros	4 hilos (Dual)	
		3 hilos (Aron)	✕
Mediciones monofásicas			Sí

Tabla 15: Tabla resumen con las características del equipo de medida diseñado

Lo más llamativo de este trabajo es la mejora de algunos de los objetivos que se fijaron. No se esperaba poder medir con la precisión que se ha alcanzado. Estos valores de precisión se han obtenido calibrando el dispositivo con el DIRIS A40 que posee las incertidumbres mostradas en la tabla 16.

PARÁMETRO	*PRECISIÓN*	*RANGO*
TENSIÓN	0.2%	30-400V
INTENSIDAD	0.2%	50mA-5A
POTENCIA ACTIVA	0.5%+2 DÍGITOS	Hasta 10kW de alcance
$\cos\varphi$	1%	0.5-1

Tabla 16: especificaciones de precisión del DIRIS A40

Capítulo 3 RESULTADOS/EXPERIMENTOS

En este capítulo se presentan los ensayos realizados al equipo diseñado y que ayudan a determinas las especificaciones técnicas de prototipo.

1 Medidas de tiempos

La medida de tiempos de procesamiento de los datos muestreados permite llevar a cabo un análisis determinante para el correcto funcionamiento del aparato. Se entiende correcto por una actualización de los datos en el LCD no demasiado elevado, un tiempo mínimo de almacenamiento y monitorización visual de los parámetros que a su vez evite que el cálculo de energía no se vea influenciado.

En la figura 53 se muestran los tiempos característicos y representados de forma gráfica, mientras que en la tabla 17 se muestran los valores medidos del ensayo.

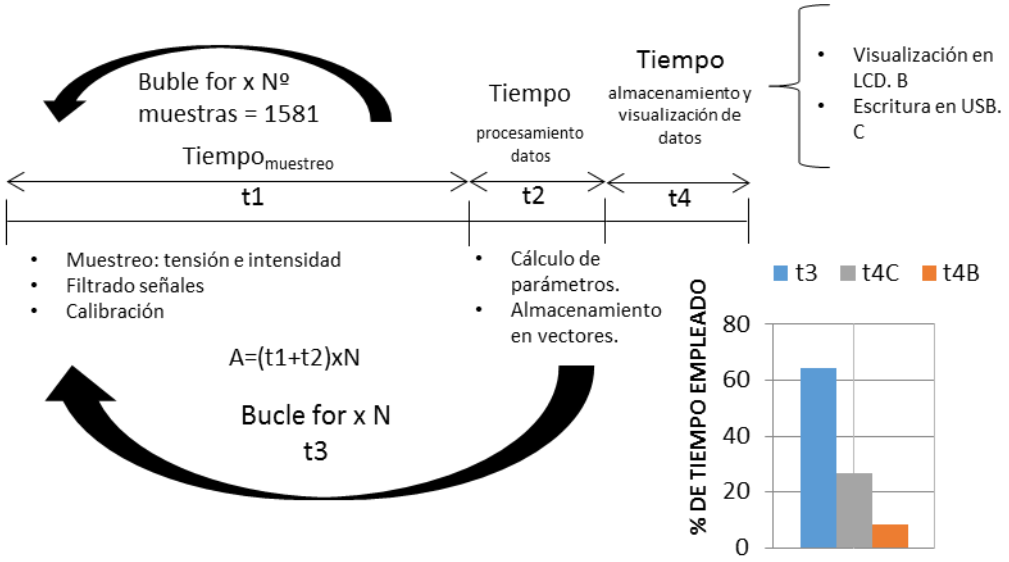


Figura 53: Definición gráfica de tiempos del código

Tiempos software		
t2	6,45	ms
t1	400	ms
t3	3251,6	ms
t4	1792	ms
t4B	431	ms
t4C	1361	ms

Tabla 17: Diferentes medidas de tiempos requeridos por software

El cálculo de energía se calcula multiplicando la potencia calculada durante t2 por el tiempo transcurrido de t1+t2, sin embargo, durante el tiempo de almacenamiento y visualización de parámetros se supone que la potencia no ha variado durante dicho periodo. En caso de no ser así se produce un error en el cálculo al suponer que la potencia en constante. Por ello interesa reducir al máximo el tiempo dedicado al almacenamiento y visualización.

Por todo esto se busca que el tiempo t3 sea muy superior a t4, algo que es posible modificar. Debe haber un acuerdo entre el tiempo de actualización de los datos visualizados y el tiempo de muestreo, y además, que el tiempo t3 sea superior a t4. Para el presente trabajo interesaba que durante la fase de experimentación los valores mostrados en el LCD se actualizasen rápidamente y por ello se fija el valor de t3 un 65% del tiempo total de manera que el LCD se actualizase cada 5s, es decir t3+t4.

Para implementar unas medidas de energía más adecuadas podría ser conveniente aumentar el número de veces N que recorre el bucle dado que el tiempo de almacenamiento de los datos no varía demasiado al escribir más tandas de parámetros. La mayor cantidad de tiempo pérdida en el USB es la de establecimiento de la comunicación, y una vez realizada, el envío de los datos es más veloz. Es por esto que conviene que una vez establecida la comunicación se escriba el mayor número de parámetros posible. De esta manera se aumenta el tiempo t3 de manera muy superior al tiempo t4. El inconveniente es que el tiempo de actualización de los datos del LCD aumentaría considerablemente, pero para aplicaciones en las que no sea necesario comprobar los parámetros a través de pantalla es más conveniente.

Es necesario aclarar que los datos mostrados en el LCD solo se muestran cada vez que el Arduino recorre todo su código, es decir, un tiempo de t3+t4. Esto no sucede así con el USB dado que los parámetros se guardan en vectores de tamaño N que se escriben seguidamente y aunque los valores escritos se hagan en un instante determinado, dichos parámetros corresponde a los calculados durante t2.

El esquema de tiempos de la figura 53, muestra que el tiempo en el que el Arduino no muestrea la señales está formado por los cálculos de los parámetros, la visualización de los datos por pantalla en caso de ser activada mediante pulsador y la escritura de parámetros en lápiz USB. Estos tiempo corresponde a un 35% del tiempo total, pero como ya se ha explicado, esta distribución es para la fase de experimentación.

La secuencia de cálculo de la energía se muestra esquemáticamente en la figura 54.

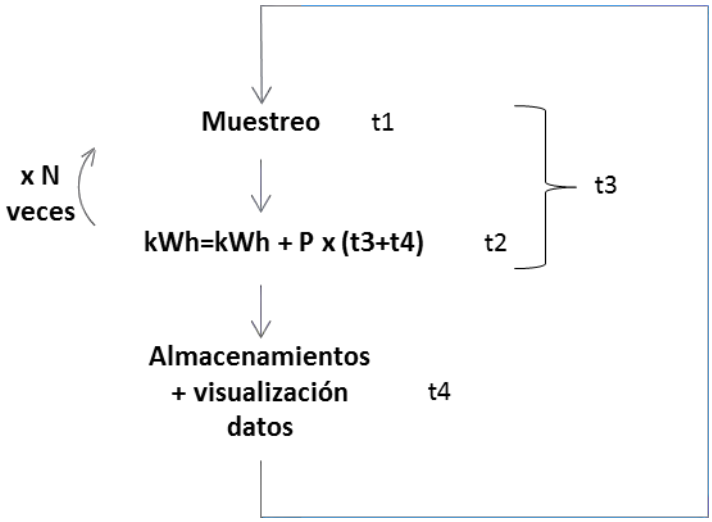


Figura 54: Secuencia de cálculo de los kWh

Durante el tiempo t4, el cálculo de energía que se realiza supone que la próxima potencia calculada es la que había durante dicho intervalo. El valor de t4 es un valor muy elevado de 1.7s lo que supone que consumos con variaciones continuas de carga provocarían una incertidumbre elevada en el cálculo.

2 Calibración del desfase entre tensión e intensidad

Para calibrar el desfase entre la tensión y la intensidad se ha procedido de la misma manera que para determinar el desfase entre la señal sin filtrar respecto de la señal filtrada.

Se ha preferido calibrar el desfase entre tensión e intensidad modificando únicamente la fase de la tensión. Esto tiene su justificación, dado que el transductor de corriente es de elevada precisión y el desfase introducido por el es casi inapreciable, se asume que la calibración de la corriente realizada en el apartado 8.6.2 es más precisa que la de tensión. Por tanto, casi todo error de fase se debería a una mala calibración angular de la tensión.

Para comprobar la calibración angular del dispositivo se recurre al DIRIS A40 que posee una precisión de 1% en el cálculo del factor de potencia.

El ensayo realizado es el de la figura 55. El valor de factor de potencia determinado por el DIRIS A40 es de 0.887 lo que equivale a un desfase de 27º 30’.

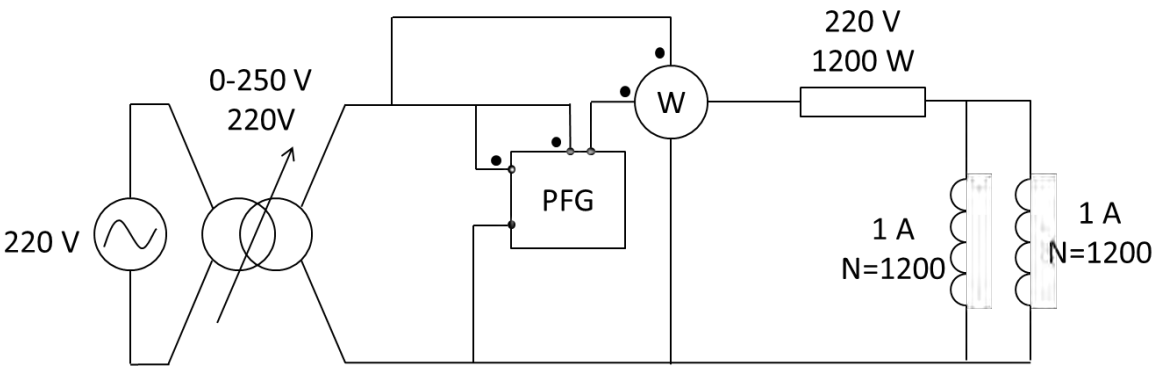


Figura 55: Ensayo de calibración de fase del dispositivo

La máxima corriente de este ensayo es de 2A puesto que el límite de intensidad de ambas bobinas es de 1A. Los valores de tensión e intensidad son de 159.7V y 1.893A.

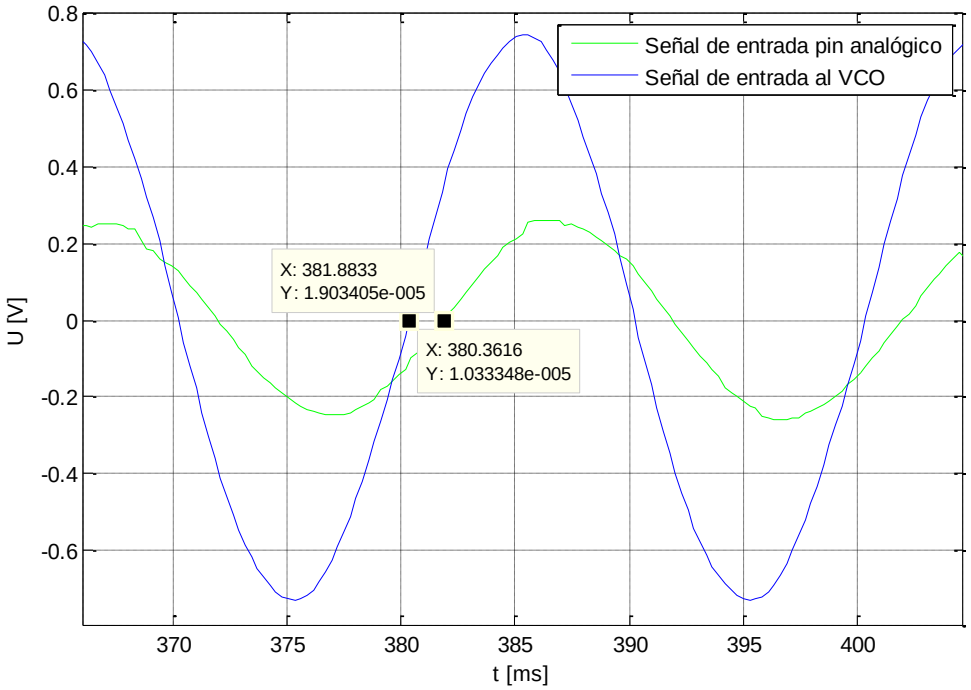


Figura 56: Señales filtradas de tensión e intensidad con desfase angular corregido

En la figura 56 se muestra la señal de tensión introducida al VCO y la señal de corriente traducida a tensión. Se ha preferido esta conversión dado que ambas magnitudes son parecidas y facilitan en análisis del desfase.

Como podemos ver en la ecuación 31 el desfase conseguido entre ambas señales es casi igual que el calculado por el DIRIS A40 que es de 27º 30'.

$$\alpha[º] = (381.883 - 380.362) \cdot \frac{360º}{20\ [ms]} = 27º\ 22' \qquad (32)$$

Por tanto, los valores de las frecuencias de corte para cada señal se muestran en la tabla 18.

Señal	F_{corte} filtro paso alto	F_{corte} filtro paso bajo
Corriente	17.88Hz	200Hz
Tensión	20.4Hz	221Hz

Tabla 18: Frecuencias de corte de filtros paso banda de señales de tensión y corriente.

La calibración entre las fases no es necesaria puesto que depende del ritmo de variación de la potencia. En caso de que la pendiente de cambio del valor de potencia fuera elevada entonces el desfase entre las fases sería crítico, sin embargo, el ritmo al que cambia este valor no es lo suficientemente rápido como para tener que considerar este desfase.

3 Medida monofásica

3.1 Ensayo resistivo

Para el ensayo resistivo se ha empleado una carga de 1200W, 220V.

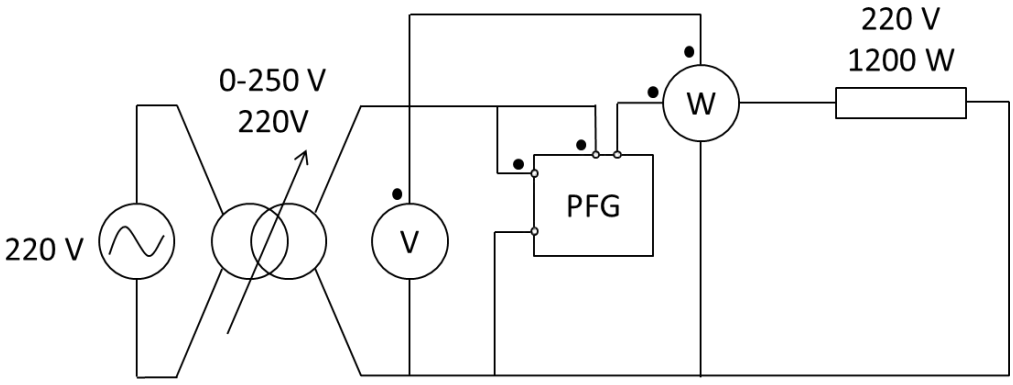


Figura 57 Esquema del ensayo de medida monofásica resistivo

Mediante el DIRIS A40 se ha determinado la precisión del dispositivo tomando este medidor como patrón. En la tabla 19 se muestra de nuevo la precisión de este aparato.

Tanto el DIRIS A40 como el dispositivo diseñado se han conectado en conexión larga procurando que ambos dispositivos midan los mismos valores de corriente y tensión.

PARÁMETRO	*PRECISIÓN*	*RANGO*
TENSIÓN	0.2%	30-400V
INTENSIDAD	0.2%	50mA-5A
POTENCIA ACTIVA	0.5%+2 DÍGITOS	Hasta 10kW de alcance
$\cos\varphi$	1%	0.5-1

Tabla 19: Precisión del DIRIS A40

En este ensayo se han tomado medidas para 10 valores distintos de tensión.
Los resultados obtenidos de precisión se pueden apreciar en la figura 58.

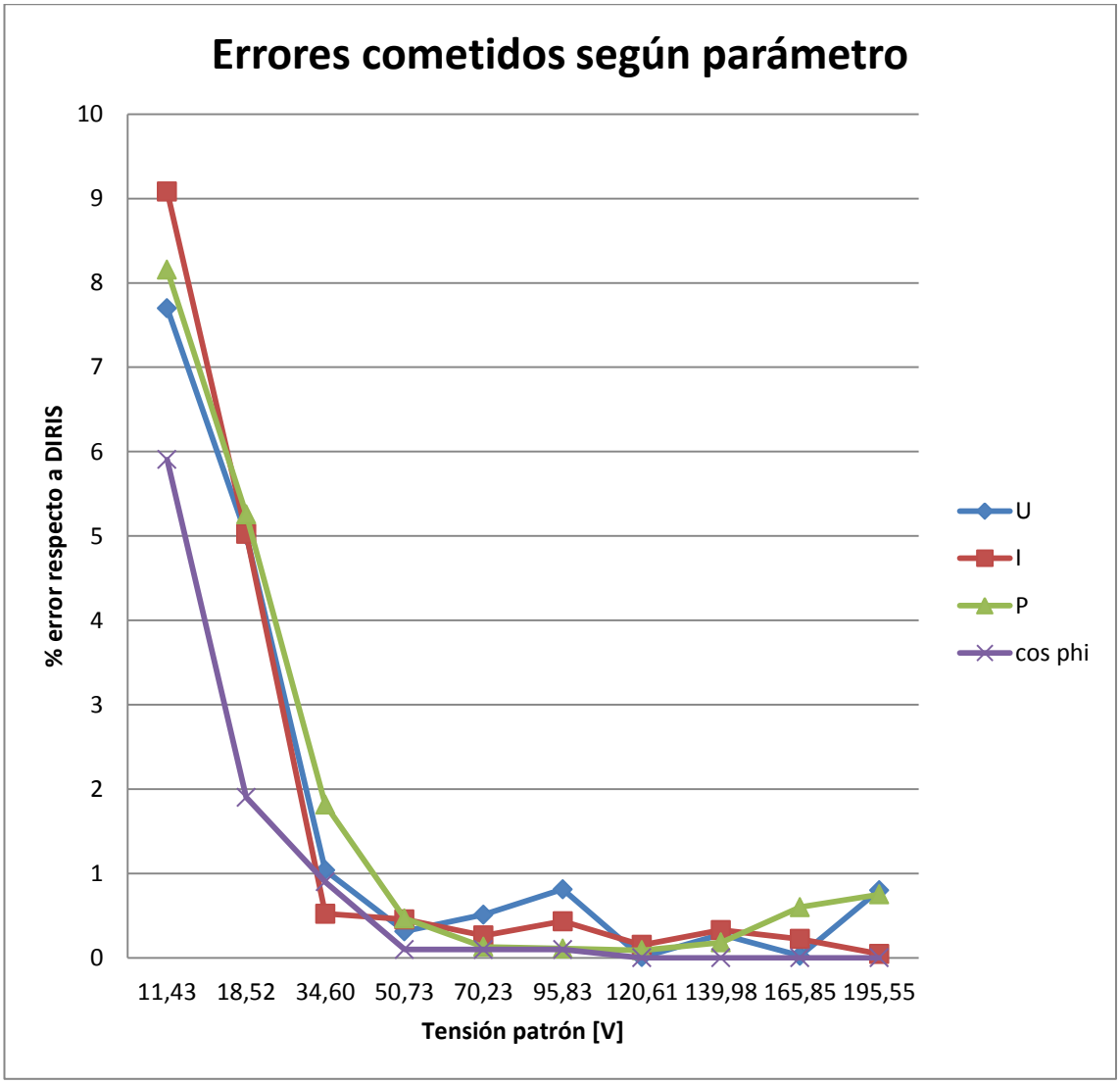


Figura 58: Distribución de errores según parámetro para distintos valores de tensión con factor de potencia 1

El desfase angular entre tensión e intensidad se ha calibrado para este ensayo. Esto se debe a que la aplicación principal de este trabajo es la medida del consumo en entornos

industriales y, por tanto, el factor de potencia que se puede encontrar estará en el rango de 0.9-1. La calibración partió de los valores de las frecuencias de corte determinadas en el apartado de filtrado y calibración.

Como se puede observar se han alcanzado con mucha holgura los objetivos buscados para este trabajo. Para un rango de valores de tensión entre 50V y 200V el aparato mantiene un precisión inferior al 1% en todos sus parámetros.

A pesar de fijar el alcance del dispositivo en 200V, este se puede aumentar hasta los 380V. Dado que las tensiones disponibles en el laboratorio no eran de tanta magnitud se prefirió fijar dicho alcance en ese valor.

3.2 Ensayo inductivo

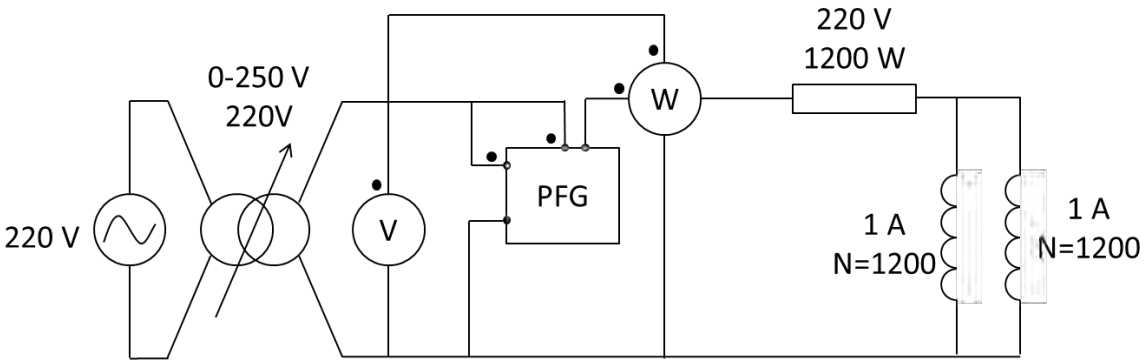


Figura 59 Esquema del ensayo de medida monofásica inductivo

Este ensayo representa el otro caso extremo en el que el factor de potencia es muy inferior a 0.9, para este ensayo se ha fijado en 0.725. En cualquier caso, en un entorno industrial donde la reactiva se cobra a un precio excesivo no se encontrarán factores de potencia tan bajos como los que se tiene en este ensayo. Sin embargo, sirve para probar hasta qué punto y con qué precisión es capaz de medir el aparato.

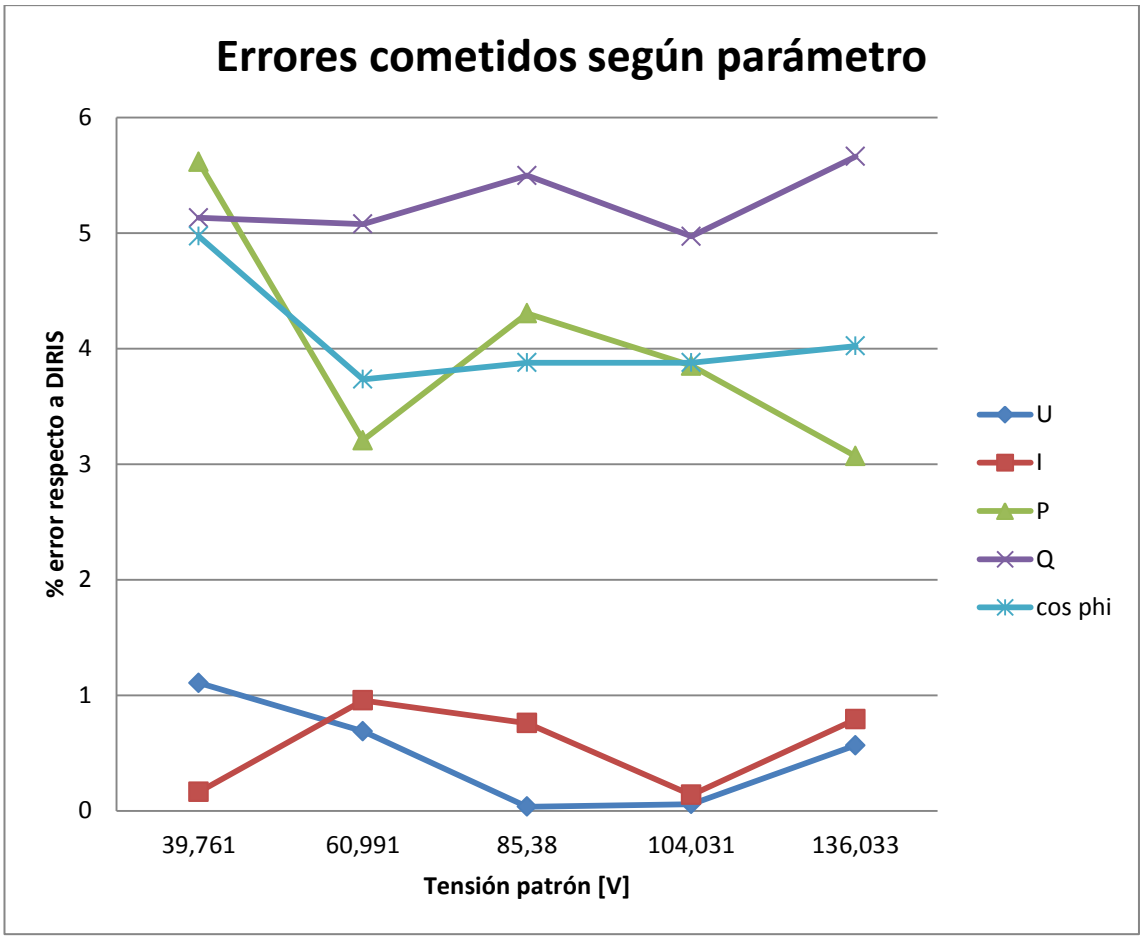


Figura 60: Distribución de errores según parámetro para distintos valores de tensión con factor de potencia 0.725

Como se puede apreciar en la figura 60, los valores concernientes al desfase angular entre tensión e intensidad pierden precisión cuando el factor de potencia no es la unidad. Dado que el aparato está diseñado para medir factor de potencia cercano a la unidad, un error ante consumos muy inductivos apenas importa.

En la figura 61 se muestra como varía el factor de potencia en función del ángulo. Ante un factor de potencia cercano a la unidad, un desfase elevado en el ángulo apenas provoca error en la medida del factor de potencia, mientras que ante un valor más alejado de la zona plana de la curva, un cambio de fase provocaría un error alto. Esto es lo que sucede pero extrapolado también a los valores de potencia activa y reactiva.

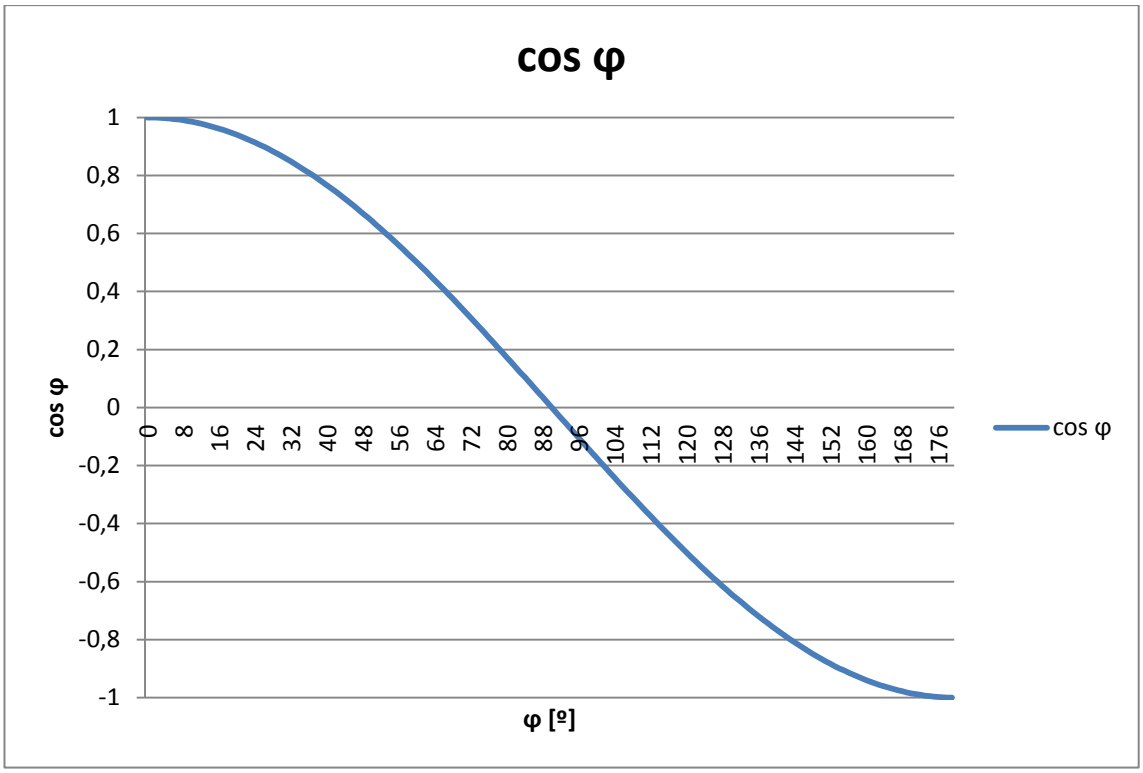


Figura 61: curva del factor de potencia respecto al ángulo

Capítulo 4 CONCLUSIONES

Para concluir este documento, se hace énfasis en la consecución más que satisfactoria de casi todos los objetivos fijados. Además, no solo se han cubierto expectativas si no que se han conseguido características cercanas a equipos comerciales. Como ejemplo, las precisiones obtenidas para tipologías de consumos como la que se preveía para emplear este dispositivo, han sido inferiores al 1%, es decir, valores de precisión muy cercanos a los del DIRIS A40, un equipo comercial de alta calidad.

Por otra parte, se han mejorado con creces las prestaciones en cuanto a precisión en medida de potencia del proyecto de código abierto de Open Energy Monitor, donde los mejores valores de precisión están entorno al 2% para un rango menos amplio de potencia respecto al de este dispositivo. El dispositivo diseñado en este documento mantiene su precisión en todos los parámetros en más de un 75% del rango de tensión, es decir, desde el 25% de la tensión nominal, hasta un 110% de la tensión nominal.

Además, las medidas de corriente son de una precisión magnífica. Al emplear este aparato para una tensión fija de red, la corriente podrá ser medida con alta precisión en un amplio rango de valores. Esto hará que las medidas de potencia mantengan un rango de precisión inferior al 1% hasta corrientes de 400mA, lo que supone un 8% de la intensidad nominal.

Para aplicaciones donde el consumo de energía varía ampliamente entre el pico y el valle de demanda, los valores obtenidos durante este último serán muy fiables.

Capítulo 5 FUTUROS DESARROLLOS

Las posibles mejoras a implantar en este PFG se desarrollan a continuación en los siguientes apartados.

1 Cálculo directo de reactiva en método de Tres Vatímetros

Para el cálculo de la reactiva en el método de los tres vatímetros existe un método de cálculo directo que se muestra en la ecuación 32.

$$Q = \frac{media(U_{RS} \cdot i_T + U_{TR} \cdot i_S + U_{ST} \cdot i_R)}{\sqrt{3}} \qquad (33)$$

Para ello es necesario conseguir un registro de todas las muestras y desfasar las tensiones simples 90º para poder hacer el cálculo.

Este método proveería al cálculo de la potencia reactiva de una incertidumbre menor a la alcanzada con el método implementado.

2 Protecciones

Debido a que el aparato de medida se ha de conectar de forma directa a los hilos de la red, se propone como futura implementación incorporar un fusible que limitase la corriente. En conjunto con ese fusible podría instalarse un diodo de disparo bidireccional, DIAC, cuya función es la de dar un camino a la corriente en caso de que la tensión sobrepase los límites que este fije.

3 Conversor de tensión a frecuencia diferencial

Posteriormente al diseño y fabricación del PFG se encontró un VCO del mismo modelo que el empleado en este proyecto pero con la diferencia de que las entradas son diferenciales, por tanto, esto evitaría el uso del amplificador de instrumentación y la necesidad de añadir un valor medio a la salida de esta para adecuarlo a la entrada del VCO empleado. Esto significaría ahorro de coste y tamaño del circuito.

Este VCO es el AD7742 [21], pertenece a la misma compañía, Analog Devices, y el comportamiento es prácticamente igual al AD7741.

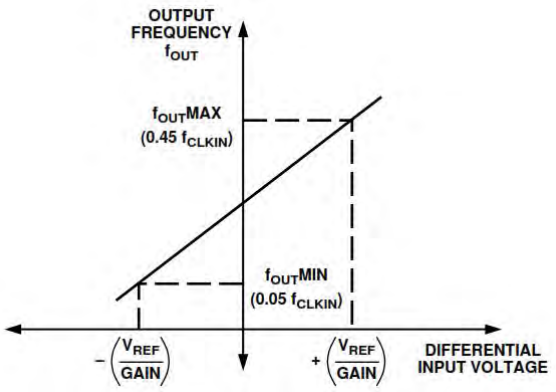


Figura 62: Conversión AD7742

Como se puede observar en la figura 62 la salida de frecuencia es la misma que la del empleado con la diferencia de que su entrada admite valores negativos.

Capítulo 6 BIBLIOGRAFÍA

[1] Open Energy Monitor, [En línea]. Available: http://openenergymonitor.org/emon/.

[2] Avago Technologies, «ACPL-W70L datasheet,» 2012.

[3] Analog Devices, «AD7741 Datasheet,» Norwood, Massachusetts.

[4] IQD Frequency Products, «HC49, 6.144MHz,» 2012.

[5] ON Semiconductor, «74HC244 datasheet,» Denver, Colorado, 2007.

[6] Analog Devices, «AD623ANZ Datasheet».

[7] STMicroelectronics, «L7800 Series, L7809CV,» 2004. [En línea]. Available: http://www.alldatasheet.com/datasheet-pdf/pdf/22639/STMICROELECTRONICS/L7809CV.html.

[8] LEM, «Current Transducer LTSR 6-NP».

[9] Maxim Integrated, «DS3231 datasheet,» San Jose, California, 2013.

[10] Arduino, «Librería para contol de tarjetas SD,» [En línea]. Available: http://www.arduino.cc/en/Reference/SD.

[11] VINCULUM, «VDIP1 Datasheet,» Glasgow, 2010.

[12] Vinculum, «Manual del usuario del firmware VNC1L,» Glasgow, 2008.

[13] Wikipedia, «Protocolo de comunicación UART,» [En línea]. Available: https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter.

[14] Arduino, «Librería para contol de LCDs alfanuméricos,» [En línea]. Available: http://www.arduino.cc/en/pmwiki.php?n=Tutorial/LiquidCrystal.

[15] Fordata Electronic, «FDCC1602B Datasheet».

[16] Electronic Assembly, «LCD Gráfico DOGM128S-6,» 2009.

[17] Universal Graphics Library for 8 bit Embedded Systems, [En línea]. Available: https://code.google.com/p/u8glib/.

[18] Analog Devices, «ADE7758 Datasheet,» Norwood, 2006.

[19] Atmel, «AT91SAM Datasheet,» 2012.

[20] I. Seidel, «Github,» [En línea]. Available: https://github.com/ivanseidel/DueTimer/blob/master/TimerCounter.md.

[21] Analog Devices, «AD7742 Datasheet,» Norwood, Massachusetts.

Parte II CÓDIGO FUENTE

```
// LIBRERIA PARA COMUNICACIÓN I2C. PARA COMUNICACIÓN CON RTC
#include "Wire.h"
// LIBRERIA PARA COMUNICACIÓN, CONFIGURACIÓN Y CONTROL DE LA PANTALLA
LCD
#include "U8glib.h"

// Registros del Reloj
#define DIRECCION_DS3231 0x68
#define DIRECCION_TEMPERATURA 0x11

// CONSTANTE QUE FIJA EL NÚMERO DE VECES QUE LLEVA A CABO LOS CÁLCULOS
#define N 6

//Configura el LCD y las comunicaciones con Arduino
U8GLIB_DOGM128 u8g(13, 11, 10, 9, 12);

//VARIABLES PARA LA INTERFAZ Y CREACIÓN DE MENÚ DEL LCD
#define KEY_NONE 0
#define KEY_PREV 1
#define KEY_NEXT 2
#define KEY_SELECT 3
#define KEY_BACK 4

uint8_t uiKeyPrev = 7;  //uint8_t entero de 8bits, definición empleada
por los ejemplos de la librería para estandarizar el tamaño de la
variable
uint8_t uiKeyNext = 3;
uint8_t uiKeySelect = 2;

uint8_t uiKeyCodeFirst = KEY_NONE;
uint8_t uiKeyCodeSecond = KEY_NONE;
uint8_t uiKeyCode = KEY_NONE;

//VARIABLES PARA DIBUJO DE MENÚ EN PANTALLA LCD
#define MENU_ITEMS 2
char *menu_strings[MENU_ITEMS] = { "Tres vatimetros", "Aron" };

uint8_t menu_current = 0;
uint8_t menu_redraw_required = 0;
uint8_t last_key_code = KEY_NONE;

//Variables de selección de menú en LCD
boolean Aron;
boolean TresVat;

 // Defino número de muestras necesarias para cada ciclo de cálculos
int numero_muestras=1852;

 // Variables de apagado automático de pantalla
volatile unsigned long Inicio=0;
volatile int LCDActivar=0;  // volatile le dice al compilador que esas
variables pueden cambiar en cualquier momento
int PinActivar=51; //Pin con interrupción asociada para activar la
pantalla durante un tiempo determinado

 // VARIABLES PARA EL CÁLCULO DEL PERIODO DE MUESTREO
unsigned long Suma_periodo=0;
unsigned long Periodo_muestreo[N];
unsigned long periodo_actual;
```

```
 // Factores de calibración
double Cal_V_R=1.034;
double Cal_I_R=1.085;
double Cal_V_S=1;
double Cal_I_S=1;
double Cal_V_T=1;
double Cal_I_T=1;


 // VARIABLES PARA CONVERSION
double Ratio_V_R=0.0002888*Cal_V_R;
double Ratio_I_R=0.007736*Cal_I_R;
double Ratio_V_S=0.0002888*Cal_V_S;
double Ratio_I_S=0.007736*Cal_I_S;
double Ratio_V_T=0.0002888*Cal_V_T;
double Ratio_I_T=0.007736*Cal_I_T;

double Ratio_V_RS
double Ratio_V_TS

 // Variables muestreadas
unsigned long ultima_muestra_V_fase_R;
unsigned long ultima_muestra_I_fase_R;
unsigned long muestra_V_fase_R;
unsigned long muestra_I_fase_R;

unsigned long ultima_muestra_V_fase_S;
unsigned long ultima_muestra_I_fase_S;
unsigned long muestra_V_fase_S;
unsigned long muestra_I_fase_S;

unsigned long ultima_muestra_V_fase_T;
unsigned long ultima_muestra_I_fase_T;
unsigned long muestra_V_fase_T;
unsigned long muestra_I_fase_T;

 // Variables sin OFFSET
double SinOffset_V_fase_R=0;
double SinOffset_I_fase_R=0;

double SinOffset_V_fase_S=0;
double SinOffset_I_fase_S=0;

double SinOffset_V_fase_T=0;
double SinOffset_I_fase_T=0;

double ultima_SinOffset_V_fase_R;
double ultima_SinOffset_I_fase_R;
double ultima_SinOffset_V_fase_S;
double ultima_SinOffset_I_fase_S;
double ultima_SinOffset_V_fase_T;
double ultima_SinOffset_I_fase_T;

double ultima_CAL_SinOffset_V_fase_R=0;

double ultima_CAL_SinOffset_I_fase_R=0;
double CAL_SinOffset_I_fase_R;
double CALFase_SinOffset_V_fase_R;

 // VARIABLES CALIBRADAS EN DESFASE
```

```
double CAL_SinOffset_V_fase_R;
double CAL_SinOffset_V_fase_S;
double CAL_SinOffset_V_fase_T;

 // VALOR DE LA CALIBRACIÓN EN DESFASE
double CAL_R=1;
double CAL_S=1;
double CAL_T=1;

 // VECTORES DE LA FECHA
byte segundos[N];
byte minutos[N];
byte horas[N];
byte diaSemana[N];
byte diaMes[N];
byte mes[N];
byte ano[N];

// VECTOR DE TEMPERATURA
byte Temp[N];

 // Variables para el cálculo de energía
unsigned long ms=0;
unsigned long ultimosms=0;
unsigned long Intervalo;
double kWhAcumulado[N];

//Variables calculadas
double V_R_RMS[N];
double I_R_RMS[N];
double P_Fase_R[N];
double S_Fase_R[N];
double Q_Fase_R[N];
double Cos_phi_R[N];

double V_S_RMS[N];
double I_S_RMS[N];
double P_Fase_S[N];
double S_Fase_S[N];
double Q_Fase_S[N];
double Cos_phi_S[N];

double V_T_RMS[N];
double I_T_RMS[N];
double P_Fase_T[N];
double S_Fase_T[N];
double Q_Fase_T[N];
double Cos_phi_T[N];

double P_Total[N];
double Q_Total[N];
double S_Total[N];
double Cos_phi[N];

//VARIABLES PARA CÁLCULOS
double  Suma_Cuadrado_V_R=0;
double  Suma_Cuadrado_I_R=0;
double  Suma_Cuadrado_V_S=0;
double  Suma_Cuadrado_I_S=0;
double  Suma_Cuadrado_V_T=0;
```

```
double  Suma_Cuadrado_I_T=0;

double Acumulado_potencia_R=0;
double Acumulado_potencia_S=0;
double Acumulado_potencia_T=0;

double S_Fase_R_Cuadrado;
double P_Fase_R_Cuadrado;
double S_Fase_S_Cuadrado;
double P_Fase_S_Cuadrado;
double S_Fase_T_Cuadrado;
double P_Fase_T_Cuadrado;

double  Cuadrado_V_R;
double  Cuadrado_V_S;
double  Cuadrado_V_T;
double  Cuadrado_I_R;
double  Cuadrado_I_S;
double  Cuadrado_I_T;

unsigned long Potencia_instantanea_R;
unsigned long Potencia_instantanea_S;
unsigned long Potencia_instantanea_T;

//Constante de los filtros paso banda
double Cte= 0.01/(0.01+256*0.000001);
double CteV=0.01/(0.01+256*0.000001);
double kk=0.0008/(266*0.000001);
double kkI=0.0008/(266*0.000001);

//VARIABLES USB
int UltimaDia=60;  //Valor que ha de ser superior a 31, el nº máx. de
días
int x=0;
int RTSPin = 5; //AD2
int CTSPin = 4; //AD3

void Calculos(int m)
{
  Periodo_muestreo[m]=micros();
  for(int i=0 ; i<numero_muestras ; i++ )
  {
    Filtrado();

    //Suma del cuadrado de las muestras de tensión
    Cuadrado_V_R= CALFase_SinOffset_V_fase_R * CALFase_SinOffset_V_fas
e_R;
    Suma_Cuadrado_V_R += Cuadrado_V_R;

    Cuadrado_V_S= CALFase_SinOffset_V_fase_S* CALFase_SinOffset_V_fase
_S;
    Suma_Cuadrado_V_S += Cuadrado_V_S;

    Cuadrado_V_T= CALFase_SinOffset_V_fase_T * CALFase_SinOffset_V_fas
e_T;
    Suma_Cuadrado_V_T += Cuadrado_V_T;

    //Suma del cuadrado de las muestras de intensidad
    Cuadrado_I_R= CAL_SinOffset_I_fase_R * CAL_SinOffset_I_fase_R;
    Suma_Cuadrado_I_R += Cuadrado_I_R;
```

```
    Cuadrado_I_S=CAL_SinOffset_I_fase_S * CAL_SinOffset_I_fase_S;
    Suma_Cuadrado_I_S += Cuadrado_I_S;

    Cuadrado_I_T=CAL_SinOffset_I_fase_T * CAL_SinOffset_I_fase_T;
    Suma_Cuadrado_I_T += Cuadrado_I_T;

    //Cálculo de los valores potencia instantanea y su valor acumulado
    Potencia_instantanea_R = CALFase_SinOffset_V_fase_R * CAL_SinOffse
t_I_fase_R;
    Acumulado_potencia_R += Potencia_instantanea_R;

    Potencia_instantanea_S = CALFase_SinOffset_V_fase_S * CAL_SinOffse
t_I_fase_S;
    Acumulado_potencia_S += Potencia_instantanea_S;

    Potencia_instantanea_T = CALFase_SinOffset_V_fase_T * CAL_SinOffse
t_I_fase_T;
    Acumulado_potencia_T += Potencia_instantanea_T;

    periodo_actual = micros() - Periodo_muestreo[m];
    Suma_periodo += periodo_actual;
    Periodo_muestreo[m] = micros();

}
  Periodo_muestreo[m]=Suma_periodo/numero_muestras;

  if( Aron==true){
    Aaron(m);
    }
  if( TresVat==true){
    tresVatimetros(m);
  }


  //Cálculo del factor de potencia
  Cos_phi[m]=P_Total[m]/S_Total[m];

  //Cálculo de energía en kWh
  ultimosms = ms;
  ms = millis();
  Intervalo = ms - ultimosms;

  kWhAcumulado[m] = kWhAcumulado[m] + P_Total[m]*Intervalo*(1/1000/100
0/3600);

  //LECTURA DE TIEMPO
  LecturaTiempo(&segundos[m], &minutos[m], &horas[m], &diaSemana[m], &
diaMes[m], &mes[m], &ano[m]);

  //LECTURA DE TEMPERATURA
  Temp[m]=Temperatura();

  //Reiniciar acumuladores
  Suma_periodo=0;
  Acumulado_potencia_T=0;
  Acumulado_potencia_S=0;
  Acumulado_potencia_R=0;
  Suma_Cuadrado_I_R=0;
  Suma_Cuadrado_I_S=0;
```

```
  Suma_Cuadrado_I_T=0;
  Suma_Cuadrado_V_R=0;
  Suma_Cuadrado_V_S=0;
  Suma_Cuadrado_V_T=0;
}


void Aaron(int m)
{
  //Cálculo de valores RMS de tensión e intensidad
  V_R_RMS[m]= Ratio_V_R * sqrt(Suma_Cuadrado_V_R/numero_muestras);
  I_R_RMS[m]= Ratio_I_R * sqrt(Suma_Cuadrado_I_R/numero_muestras);

  V_S_RMS[m]= 0;  //La tensión medida en este pin en mét. Aron es
entre las fases SS
  I_S_RMS[m]= Ratio_I_S * sqrt(Suma_Cuadrado_I_S/numero_muestras);

  V_T_RMS[m]= Ratio_V_T * sqrt(Suma_Cuadrado_V_T/numero_muestras);
  I_T_RMS[m]= Ratio_I_T * sqrt(Suma_Cuadrado_I_S/numero_muestras);

  //Cálculo de potencia activa como la media de la potencia
instantanea
  P_Fase_R[m] = Ratio_V_RS * Ratio_I_R * Acumulado_potencia_R/numero_m
uestras;

  P_Fase_S[m] = 0;

  P_Fase_T[m] = Ratio_V_TS * Ratio_I_T * Acumulado_potencia_T/numero_m
uestras;

  P_Total[m]= P_Fase_R[m] ;

  //Cálculo de tensiones simples suponiendo U neutro cero
  double V_media;
  V_media=(V_R_RMS[m]+V_T_RMS[m])/2;
  V_R_RMS[m]=V_media/sqrt(3);
  V_S_RMS[m]=V_media/sqrt(3);
  V_T_RMS[m]=V_media/sqrt(3);

  //Cálculo de la potencia reactiva
  Q_Total[m]= sqrt(S_Total[m]*S_Total[m]-P_Fase_R[m]*P_Fase_R[m]);

  //Cálculo de potencia aparente a partir de potencia reactiva y
activa
  S_Total[m]= sqrt(P_Total[m]*P_Total[m]+ Q_Total[m]*Q_Total[m]);
}

void tresVatimetros(int m)
{
  //Cálculo de valores RMS de tensión e intensidad
  V_R_RMS[m]= Ratio_V_R  * sqrt(Suma_Cuadrado_V_R/numero_muestras);
  I_R_RMS[m]= Ratio_I_R  * sqrt(Suma_Cuadrado_I_R/numero_muestras);

  V_S_RMS[m]= Ratio_V_S * sqrt(Suma_Cuadrado_V_S/numero_muestras);
  I_S_RMS[m]= Ratio_I_S * sqrt(Suma_Cuadrado_I_S/numero_muestras);

  V_T_RMS[m]= Ratio_V_T * sqrt(Suma_Cuadrado_V_T/numero_muestras);
  I_T_RMS[m]= Ratio_I_T * sqrt(Suma_Cuadrado_I_T/numero_muestras);
```

```
  //Cálculo de potencia activa como la media de la potencia
instantanea
  P_Fase_R[m] = Ratio_V_R * Ratio_I_R  * Acumulado_potencia_R/numero_m
uestras;

  P_Fase_S[m] = Ratio_V_S * Ratio_I_S * Acumulado_potencia_S/numero_mu
estras;

  P_Fase_T[m] = Ratio_V_T * Ratio_I_T * Acumulado_potencia_T/numero_mu
estras;

  P_Total[m]= P_Fase_R[m] + P_Fase_S[m] + P_Fase_T[m];

  //Cálculo de potencia aparente a partir de valores RMS de tensión e
intensidad
  S_Fase_R[m] = V_R_RMS[m] * I_R_RMS[m];
  S_Fase_S[m] = V_S_RMS[m] * I_S_RMS[m];
  S_Fase_T[m] = V_T_RMS[m] * I_T_RMS[m];


  //Cálculo de energía reactiva a partir de potencia aparente y activa
  S_Fase_R_Cuadrado = S_Fase_R[m] * S_Fase_R[m];
  P_Fase_R_Cuadrado = P_Fase_R[m] * P_Fase_R[m];
  Q_Fase_R[m] = sqrt(S_Fase_R_Cuadrado - P_Fase_R_Cuadrado);

  S_Fase_S_Cuadrado = S_Fase_S[m] * S_Fase_S[m];
  P_Fase_S_Cuadrado = P_Fase_S[m] * P_Fase_S[m];
  Q_Fase_S[m] = sqrt(S_Fase_S_Cuadrado - P_Fase_S_Cuadrado);

  S_Fase_T_Cuadrado = S_Fase_T[m]*S_Fase_T[m];
  P_Fase_T_Cuadrado = P_Fase_T[m]*P_Fase_T[m];
  Q_Fase_T[m] = sqrt(S_Fase_T_Cuadrado - P_Fase_T_Cuadrado);

  Q_Total[m]= Q_Fase_R[m]+Q_Fase_S[m]+Q_Fase_T[m];

  S_Total[m]=S_Fase_R[m]+S_Fase_S[m]+S_Fase_T[m];

  //Cálculo del factor de potencia por fase
  //Cos_phi_R[m]= P_Fase_R[m]/S_Fase_R[m];
  //Cos_phi_S[m]= P_Fase_S[m]/S_Fase_S[m];
  //Cos_phi_T[m]= P_Fase_S[m]/S_Fase_T[m];

}

void Filtrado()
{
    //Eliminación de OFFSET
    ultima_muestra_V_fase_R= muestra_V_fase_R;
    ultima_muestra_I_fase_R= muestra_I_fase_R;

    ultima_muestra_V_fase_S=muestra_V_fase_S;
    ultima_muestra_I_fase_S=muestra_I_fase_S;

    ultima_muestra_V_fase_T=muestra_V_fase_T;
    ultima_muestra_I_fase_T=muestra_I_fase_T;

    TC_Start(TC0,0);
    TC_Start(TC1,0);
    TC_Start(TC2,2);
    delayMicroseconds(160);
```

```
    TC_Stop(TC0,0);
    TC_Stop(TC1,0);
    TC_Stop(TC2,2);

    muestra_I_fase_R = analogRead(A0); //TARDA 1 us
    muestra_I_fase_S = analogRead(A1);
    muestra_I_fase_T = analogRead(A2);
    muestra_V_fase_R = TC_ReadCV(TC0,0)*6250; //LECTURA Y CONVERSIÓN A
HZ
    muestra_V_fase_S = TC_ReadCV(TC1,0)*6250;
    muestra_V_fase_T = TC_ReadCV(TC2,2)*6250;

    //FILTRO PASO ALTO
    ultima_SinOffset_V_fase_R = SinOffset_V_fase_R;
    ultima_SinOffset_I_fase_R = SinOffset_I_fase_R;
    ultima_SinOffset_V_fase_S = SinOffset_V_fase_S;
    ultima_SinOffset_I_fase_S = SinOffset_I_fase_S;
    ultima_SinOffset_V_fase_T = SinOffset_V_fase_T;
    ultima_SinOffset_I_fase_T = SinOffset_I_fase_T;

    SinOffset_V_fase_R = CteV * (ultima_SinOffset_V_fase_R + muestra_V
_fase_R - ultima_muestra_V_fase_R);
    SinOffset_I_fase_R = Cte * (ultima_SinOffset_I_fase_R + muestra_I_
fase_R - ultima_muestra_I_fase_R);

    SinOffset_V_fase_S = CteV * (ultima_SinOffset_V_fase_S + muestra_V
_fase_S - ultima_muestra_V_fase_S);
    SinOffset_I_fase_S = Cte * (ultima_SinOffset_I_fase_S + muestra_I_
fase_S - ultima_muestra_I_fase_S);

    SinOffset_V_fase_T = CteV * (ultima_SinOffset_V_fase_T + muestra_V
_fase_T - ultima_muestra_V_fase_T);
    SinOffset_I_fase_T = Cte * (ultima_SinOffset_I_fase_T + muestra_I_
fase_T - ultima_muestra_I_fase_T);

    //FILTRO PASO BAJO
    ultima_CAL_SinOffset_V_fase_R=CAL_SinOffset_V_fase_R;
    CAL_SinOffset_V_fase_R = 1/(1+kk)*(SinOffset_V_fase_R+kk*ultima_CA
L_SinOffset_V_fase_R);

    ultima_CAL_SinOffset_I_fase_R=CAL_SinOffset_I_fase_R;
    CAL_SinOffset_I_fase_R = 1/(1+kkI)*(SinOffset_I_fase_R+kkI*ultima_
CAL_SinOffset_I_fase_R);

    ultima_CAL_SinOffset_V_fase_S=CAL_SinOffset_V_fase_S;
    CAL_SinOffset_V_fase_S = 1/(1+kk)*(SinOffset_V_fase_S+kk*ultima_CA
L_SinOffset_V_fase_S);

    ultima_CAL_SinOffset_I_fase_S=CAL_SinOffset_I_fase_S;
    CAL_SinOffset_I_fase_S = 1/(1+kkI)*(SinOffset_I_fase_S+kkI*ultima_
CAL_SinOffset_I_fase_S);

    ultima_CAL_SinOffset_V_fase_T=CAL_SinOffset_V_fase_T;
    CAL_SinOffset_V_fase_T = 1/(1+kk)*(SinOffset_V_fase_T+kk*ultima_CA
L_SinOffset_V_fase_T);

    ultima_CAL_SinOffset_I_fase_T =CAL_SinOffset_I_fase_T;
    CAL_SinOffset_I_fase_T = 1/(1+kkI)*(SinOffset_I_fase_T+kkI*ultima_
CAL_SinOffset_I_fase_T);
```

```
    //CALIBRACIÓN FASE DE TENSIONES, NO APLICADO EN ESTE CÓDIGO
VALORES DE CAL_X=1
    CALFase_SinOffset_V_fase_R = ultima_CAL_SinOffset_V_fase_R + CAL_R
 * (CAL_SinOffset_V_fase_R - ultima_CAL_SinOffset_V_fase_R);
    CALFase_SinOffset_V_fase_S = ultima_CAL_SinOffset_V_fase_S + CAL_S
 * (CAL_SinOffset_V_fase_S - ultima_CAL_SinOffset_V_fase_S);
    CALFase_SinOffset_V_fase_T = ultima_CAL_SinOffset_V_fase_T + CAL_T
 * (CAL_SinOffset_V_fase_T - ultima_CAL_SinOffset_V_fase_T);
}



void setup()
{
  //CONFIGURACION USB
  ConfigUSB();

  //PARTE DEL RELOJ
  Wire.begin();
  //ConfigTiempo(30,01,16,4,10,06,15); //Una vez configurado el reloj
no volátil y alimentado con pila aux. no es encesario reconfigurar.


  //PARTE DEL LCD
  uiSetup();    //Configuración de botones para selección de método de
cálculo
  menu_redraw_required = 1;     //Fuerza el inicio del dibujo
  SeleccionMetodo();
  attachInterrupt(PinActivar,LCDActivado,RISING);   //Defino
interrupción para activar pantalla en el pin 51
  LCDActivar=0;
  Inicio=0;
  BorrarPantalla();

  //Las entradas analógicas ya estan definidas por defecto

  //Modifico a 12 bits la resolución del ADC dado que por defecto
vienen 10 bits
  analogReadResolution(12);

  // Activa el contador a través del Power Management Controller (PMC)
  pmc_set_writeprotect(false);  //Desactivación de la protección de
este registro
  pmc_enable_periph_clk (ID_TC0) ;  //activación de los tres
contadores
  pmc_enable_periph_clk (ID_TC3) ;
  pmc_enable_periph_clk (ID_TC8) ;

  TC_Configure(/* Reloj */TC0,/* Canal */0,  TC_CMR_EEVTEDG_RISING |
TC_CMR_TCCLKS_XC0 ); //pin 22 equivale a tensión R o RS
  TC_Configure(/* Reloj */TC1,/* Canal */0,  TC_CMR_EEVTEDG_RISING |
TC_CMR_TCCLKS_XC0 ); //pin analógico 3 equivale a tensión S
  TC_Configure(/* Reloj */TC2,/* Canal */2,  TC_CMR_EEVTEDG_RISING |
TC_CMR_TCCLKS_XC2 ); //pin 30 equivale a tensión T o TS

}

void loop()
{
  for(int m=0 ; m<N ; m++ )
```

```
  {
    Calculos(m);
  }
  if(LCDActivar==1)
  {
    LCDValores(N);
  }
  if((millis()-Inicio/1000) >= 60001)
  {
    BorrarPantalla();
  }
 }
 for(int m=0 ; m<N ; m++ )
 {
   AbrirTXT();
   EscribirTiempoTXT(m);
   EscribirTXT(m);
   CambioParrafo();
   CerrarTXT();
   if(CambioDia()==true)
   {
     x++;
   }
 }
}

// RELOJ DE TIEMPO REAL

byte decToBcd(byte val)  //CONVERSION DE DECIMAL A BINARIO
{
  return( (val/10*16) + (val%10) );
}
byte bcdToDec(byte val)  //CONVERSION DE BINARIO A DECIMAL
{
  return( (val/16*10) + (val%16) );
}

void ConfigTiempo(byte segundos, byte minutos, byte horas, byte
diaSemana, byte diaMes, byte mes, byte ano)
{
  // sets time and date data to DS3231
  Wire.beginTransmission(DIRECCION_DS3231);
  Wire.write(0); // set next input to start at the seconds register
  Wire.write(decToBcd(segundos)); // set seconds
  Wire.write(decToBcd(minutos)); // set minutes
  Wire.write(decToBcd(horas)); // set hours
  Wire.write(decToBcd(diaSemana)); // set day of week (1=Sunday,
7=Saturday)
  Wire.write(decToBcd(diaMes)); // set date (1 to 31)
  Wire.write(decToBcd(mes)); // set month
  Wire.write(decToBcd(ano)); // set year (0 to 99)
  Wire.endTransmission();
}
void LecturaTiempo(byte *segundos, byte *minutos, byte *horas, byte
*diaSemana, byte *diaMes, byte *mes, byte *ano)
{
  Wire.beginTransmission(DIRECCION_DS3231);
  Wire.write(0); //  SE FIJA PUNTERO AL REGISTRO EN LA POSICIÓN 00h
  Wire.endTransmission();
  Wire.requestFrom(DIRECCION_DS3231, 7); // SE PIDEN 7 BYTES DE DATOS
```

```
  *segundos = bcdToDec(Wire.read() & 0x7f);
  *minutos = bcdToDec(Wire.read());
  *horas = bcdToDec(Wire.read() & 0x3f);
  *diaSemana = bcdToDec(Wire.read());
  *diaMes = bcdToDec(Wire.read());
  *mes = bcdToDec(Wire.read());
  *ano = bcdToDec(Wire.read());
}

double Temperatura()
{
  Wire.beginTransmission(DIRECCION_DS3231);
  Wire.write(DIRECCION_TEMPERATURA); // set DS3231 register pointer to
11h
  Wire.endTransmission();
  Wire.requestFrom(DIRECCION_DS3231, 2);
  double byte1 = bcdToDec(Wire.read());
  int byte2 = bcdToDec(Wire.read()>>6);
  switch(byte2){
    case 0:
    byte1=byte1+0.0;
    break;
    case 1:
    byte1=byte1+0.25;
    break;
    case 2:
    byte1=byte1+0.5;
    break;
    case 3:
    byte1=byte1+0.75;
    break;
    }
  return byte1;
}

// FUNCIONES PARA PANTALLA LCD

void LCDActivado() //FUNCIÓN DE LA INTERRUPCION PARA ACTIVAR PANTALLA
LCD MIENTRAS EL ARDUINO FUNCIONA
{
  Inicio=micros();
  LCDActivar=1;
}

void BorrarPantalla(){ //ESTA FUNCIÓN BORRA LA PANTALLA LCD CUANDO HA
PASADO 1 MIN
  u8g.firstPage();
  do{
  }while(u8g.nextPage());
  u8g.sleepOff();
  LCDActivar=0;
}

void uiSetup(void) {  //FUNCIÓN PARA CONFIGURAR LOS PULSADORES DEL
MENU
  pinMode(PinActivar, INPUT);           // set pin to input
  digitalWrite(PinActivar, HIGH);       // turn on pullup resistors
  pinMode(uiKeyPrev, INPUT);           // set pin to input
  digitalWrite(uiKeyPrev, HIGH);       // turn on pullup resistors
  pinMode(uiKeyNext, INPUT);           // set pin to input
```

```
  digitalWrite(uiKeyNext, HIGH);        // turn on pullup resistors
  pinMode(uiKeySelect, INPUT);            // set pin to input
  digitalWrite(uiKeySelect, HIGH);        // turn on pullup resistors}
}

void uiStep(void) {  //Evita el bounde de los botones al cambio el
valor de uiKeyCode a la segunda pasada
  uiKeyCodeSecond = uiKeyCodeFirst;
  if ( digitalRead(uiKeyPrev) == LOW )
    uiKeyCodeFirst = KEY_PREV;
  else if ( digitalRead(uiKeyNext) == LOW )
    uiKeyCodeFirst = KEY_NEXT;
  else if ( digitalRead(uiKeySelect) == LOW )
    uiKeyCodeFirst = KEY_SELECT;
  else
    uiKeyCodeFirst = KEY_NONE;

  if ( uiKeyCodeSecond == uiKeyCodeFirst )
    uiKeyCode = uiKeyCodeFirst;
  else
    uiKeyCode = KEY_NONE;
}


void draw(void) {
  int h,w;
  u8g.setFont(u8g_font_unifont);
  u8g.setFontRefHeightExtendedText();
  u8g.setFontPosTop();
  u8g.drawStr( 2, 2, "Metodo calculo:");
}

void drawMenu(void) {
  uint8_t i, h;
  u8g_uint_t w, d;

  u8g.setFont(u8g_font_unifont);
  u8g.setFontRefHeightExtendedText();
  u8g.setFontPosTop();

  h = u8g.getFontAscent()-u8g.getFontDescent();
  w = u8g.getWidth();

  for( i = 0; i < MENU_ITEMS; i++ ) {
    d = (w-u8g.getStrWidth(menu_strings[i]))/2;
    u8g.setDefaultForegroundColor();
    u8g.drawStr( 2, 2, "Metodo calculo:");
    if ( i == menu_current ) {
      u8g.drawBox(0, i*h+2*h, w, h);
      u8g.setDefaultBackgroundColor();
    }
    u8g.drawStr(d, i*h+2*h, menu_strings[i]);
  }
}

void updateMenu(void) {
  if ( uiKeyCode != KEY_NONE && last_key_code == uiKeyCode ) {
    return;
  }
  last_key_code = uiKeyCode;
```

```
  switch ( uiKeyCode ) {
    case KEY_NEXT:
      menu_current++;
      if ( menu_current >= MENU_ITEMS )
        menu_current = 0;
      menu_redraw_required = 1;
      break;
    case KEY_PREV:
      if ( menu_current == 0 )
      menu_current = MENU_ITEMS;
      menu_current--;
      menu_redraw_required = 1;
      break;
    case KEY_SELECT:
       if(menu_current == 0
       ){
         Aron=true;
       }
       else{
         Aron=false;
       }
       if(menu_current == 1){
         TresVat=true;
       }
       else{
         TresVat=false;
       }
      break;
  }
}

void SeleccionMetodo()
{
  do{
  uiStep();  //Detecta presión de botón

  if(menu_redraw_required != 0){
  u8g.firstPage();
  do{
    draw();
    drawMenu();
  }while(u8g.nextPage());
  menu_redraw_required = 0;
  }

  updateMenu();  //Actualiza menú o selecciona método

  }while(uiKeyCode!= KEY_SELECT);  //No sale del blucle hasta haber
seleccionado el método
}

void LCDValores(int m)
{
  Serial.println("entra en dinujar");
  u8g.firstPage();
  do {
    int h,w,hh,ww;
    u8g.setFont(u8g_font_baby);
    u8g.setFontRefHeightExtendedText();
```

```
u8g.setFontPosTop();

u8g.drawStr( 1, 1, "U R=");
h = u8g.getFontAscent()-u8g.getFontDescent();
w = u8g.getStrWidth("U R=");

u8g.setPrintPos(1+w, 1);
u8g.print(V_R_RMS[m],2);//2 decimales

u8g.drawStr( 64, 1, "U S=");

u8g.setPrintPos(64+w, 1);
u8g.print(V_S_RMS[m],2); //2 decimales

u8g.drawStr( 1, h, "U T=");

u8g.setPrintPos(1+w, h);
u8g.print(V_T_RMS[m],2); //2 decimales

u8g.drawStr( 64, h, "I R=");

u8g.setPrintPos(64+w, h);
u8g.print(I_R_RMS[m],3); //2 decimales

u8g.drawStr( 1, h*2, "I S=");

u8g.setPrintPos(1+w, h*2);
u8g.print(I_S_RMS[m],3); //2 decimales

u8g.drawStr( 64, h*2, "I T=");

u8g.setPrintPos(64+w, h*2);
u8g.print(I_T_RMS[m],3); //2 decimales

u8g.drawStr( 1, h*3, "P W=");

u8g.setPrintPos(1+w, h*3);
u8g.print(P_Total[m],2); //2 decimales

u8g.drawStr( 64, h*3, "Q Var=");
ww = u8g.getStrWidth("Q Var=");

u8g.setPrintPos(64+ww, h*3);
u8g.print(Q_Total[m],2); //2 decimales

u8g.drawStr( 1, h*4, "S VA=");
ww = u8g.getStrWidth("S VA=");

u8g.setPrintPos(1+ww, h*4);
u8g.print(S_Total[m],2); //2 decimales

u8g.drawStr( 64, h*4, "CosPhi=");
ww = u8g.getStrWidth("CosPhi=");

u8g.setPrintPos(64+ww, h*4);
u8g.print(Cos_phi[m],2); //2 decimales

u8g.drawStr( 1, h*5, "kWh=");
ww = u8g.getStrWidth("kWh=");
```

```
    u8g.setPrintPos(1+ww, h*5);
    u8g.print(kWhAcumulado[m],2); //2 decimales

  } while( u8g.nextPage() );
}

//PARTE DEL USB
void ConfigUSB()
{
  pinMode(RTSPin, INPUT);  //pines de control de flujo
  pinMode(CTSPin, OUTPUT);
  Serial1.begin(9600);  //se fija la velocidad de comunicación con USB

  digitalWrite(CTSPin, LOW); // Clear to Send, quiere decir que el USB
siempre puede mandar información al Arduino sin tener que esperar a la
respuesta

  SendToUsb("IPA");   // Se configura en modo ASCII
  SendToUsbTermCmd();  //Se envía retorno de carro para terminar el
envío anterior

}

void AbrirTXT()
{
  if (Serial1.available() > 32) {
    digitalWrite(CTSPin, HIGH);
  } else {
    digitalWrite(CTSPin, LOW);
  }
  delay(50);

  byte seg, m, h, diaSem, diaM, mes, a;
  // retrieve data from DS3231
  LecturaTiempo(&seg, &m, &h, &diaSem, &diaM, &mes,
  &a);
  a=a+20;
  seg=seg/2;
  unsigned long fecha=(a<<9)|(mes<<5)|(diaM);
  unsigned int fechaDos=(h<<11)|(m<<5)|(seg);
  fecha=(fecha<<16);
  fecha=(fecha)|(fechaDos);

    delay(5);
    SendToUsb("OPW ");
    SendToUsbInteger(x); //Nombre del archivo fijado numericamente
    SendToUsb(".TXT");
    SendToUsb(" ");
    SendToUsbHex(fecha);
    SendToUsbTermCmd();
}

void CerrarTXT()
{
  if (Serial1.available() > 32) {
    digitalWrite(CTSPin, HIGH);
  } else {
    digitalWrite(CTSPin, LOW);
  }
  delay(5);
```

```
    delay(5);
    SendToUsb("CLF ");
    SendToUsbInteger(x);
    SendToUsb(".TXT");
    SendToUsbTermCmd();
    delay(5);
}
void EscribirTXT(int m)
{
  int nCaracteres;
  if (Serial1.available() > 32) {
    digitalWrite(CTSPin, HIGH);
  }
  else{
    digitalWrite(CTSPin, LOW);
  }
  delay(5);

    nCaracteres= numCharInDouble(I_R_RMS[m]) +
    numCharInDouble(I_S_RMS[m]) +
    numCharInDouble(I_T_RMS[m]) +
    numCharInDouble(V_R_RMS[m]) +
    numCharInDouble(V_S_RMS[m]) +
    numCharInDouble(V_T_RMS[m]) +
    numCharInDouble(P_Total[m]) +
    numCharInDouble(Q_Total[m]) +
    numCharInDouble(S_Total[m]) +
    numCharInDouble(Cos_phi[m]) +
    numCharInDouble(kWhAcumulado[m]) ;

    nCaracteres= 11 +nCaracteres ;    //caracter espacio entre valores
    SendToUsb("WRF ");
    SendToUsbInteger(nCaracteres);
    SendToUsbTermCmd();
    delay(5);
    SendToUsb(" ");
    SendToUsbDouble(V_R_RMS[m]);
    SendToUsb(" ");
    SendToUsbDouble(I_R_RMS[m]);
    SendToUsb(" ");
    SendToUsbDouble(V_S_RMS[m]);
    SendToUsb(" ");
    SendToUsbDouble(I_S_RMS[m]);
    SendToUsb(" ");
    SendToUsbDouble(V_T_RMS[m]);
    SendToUsb(" ");
    SendToUsbDouble(I_T_RMS[m]);
    SendToUsb(" ");
    SendToUsbDouble(P_Total[m]);
    SendToUsb(" ");
    SendToUsbDouble(Q_Total[m]);
    SendToUsb(" ");
    SendToUsbDouble(S_Total[m]);
    SendToUsb(" ");
    SendToUsbDouble(Cos_phi[m]);
    SendToUsb(" ");
    SendToUsbDouble(kWhAcumulado[m]);
    SendToUsbTermCmd();
```

```
    delay(5);
}

void EscribirTiempoTXT(int m){
   if (Serial1.available() > 32) {
     digitalWrite(CTSPin, HIGH);
   } else {
     digitalWrite(CTSPin, LOW);
   }
   delay(5);

   int CaracteresTemp;
   CaracteresTemp=numCharInDouble(Temp[m]);
   CaracteresTemp= 3 + CaracteresTemp; //Caracteres de °C y espacio
   CaracteresTemp= 3 + CaracteresTemp; //Caracteres espacio y barras
   CaracteresTemp= 6 + CaracteresTemp; //Caracteres de horas, minutos
y segundos

   SendToUsb("WRF ");
   SendToUsbInteger(CaracteresTemp);
   SendToUsbTermCmd();
   delay(5);
   SendToUsbDouble(Temp[m]);
   SendToUsb("°C ");
   if (horas[m]<10)
   {
     SendToUsb("0");
   }
   SendToUsbInteger(horas[m]);
   SendToUsb("/");
   if (minutos[m]<10)
   {
     SendToUsb("0");
   }
   SendToUsbInteger(minutos[m]);
   SendToUsb("/");
   if (segundos[m]<10)
   {
     SendToUsb("0");
   }
   SendToUsbInteger(segundos[m]);
   SendToUsb(" ");
   SendToUsbTermCmd();
   delay(5);

}

void ReadFromVDIP1(){
  int incounts=0;
  char Incomings[31];

  while(Serial1.available()){
    Incomings[incounts]=Serial1.read();
    delay(5);
    if(Incomings[incounts]==13 || incounts==31)break;
      incounts++;
  }

  Incomings[incounts]=0;
  if(incounts!=0){
```

```
    Serial.println(Incomings);
  }

}
void CambioParrafo(){

    SendToUsb("WRF ");
    SendToUsbInteger(3);
    SendToUsbTermCmd();
    delay(5);
    while (digitalRead(RTSPin) == HIGH) { }
    Serial1.println(" ");
    digitalWrite(CTSPin, LOW);
}

void SendToUsbTermCmd() {
  while (digitalRead(RTSPin) == HIGH) { }
  Serial1.write(13);
  digitalWrite(CTSPin, LOW);

}

void SendToUsbInteger(int value){
  while (digitalRead(RTSPin) == HIGH) { }
  Serial1.print(value);
  digitalWrite(CTSPin, LOW);
}

void SendToUsbDouble(double value){
  while (digitalRead(RTSPin) == HIGH) { }
  Serial1.print(value);
  digitalWrite(CTSPin, LOW);
}
void SendToUsb(String val) {
  while (digitalRead(RTSPin) == HIGH) { }
  Serial1.print(val);
  digitalWrite(CTSPin, LOW);
}
void SendToUsbHex(unsigned long val){
    while (digitalRead(RTSPin) == HIGH) { }
    Serial1.print("0x");
    Serial1.print(val,HEX);
    digitalWrite(CTSPin, LOW);
}

boolean CambioDia()
{
  byte dia ;
  boolean confirma=false;
  Wire.beginTransmission(DIRECCION_DS3231);
  Wire.write(4); // set DS3231 register pointer to 04h
  Wire.endTransmission();
  Wire.requestFrom(DIRECCION_DS3231, 1);
  dia = bcdToDec(Wire.read());
  if(dia!=UltimaDia)
  {
    confirma=true;
  }
  UltimaDia=dia;
  return confirma;
```

```
}

int numCharInDouble(double value)
{

   //Minimum number is 4 -> 0.00
   int charnum = 4;

   if (value>=10.00) charnum = 5;
   if (value>=100.00) charnum = 6;
   if (value>=1000.00) charnum = 7;
   if (value>=10000.00) charnum = 8;
   if (value>=100000.00) charnum = 9;
   if (value>=1000000.00) charnum = 10;
   if (value>=10000000.00) charnum = 11;
   if (value>=100000000.00) charnum = 12;
   if (value>=1000000000.00) charnum = 13;

   return charnum;
}
```

Parte III PRESUPUESTO

La tabla siguiente muestra todos los componentes empleados y el precio asociado juntos con el coste total.

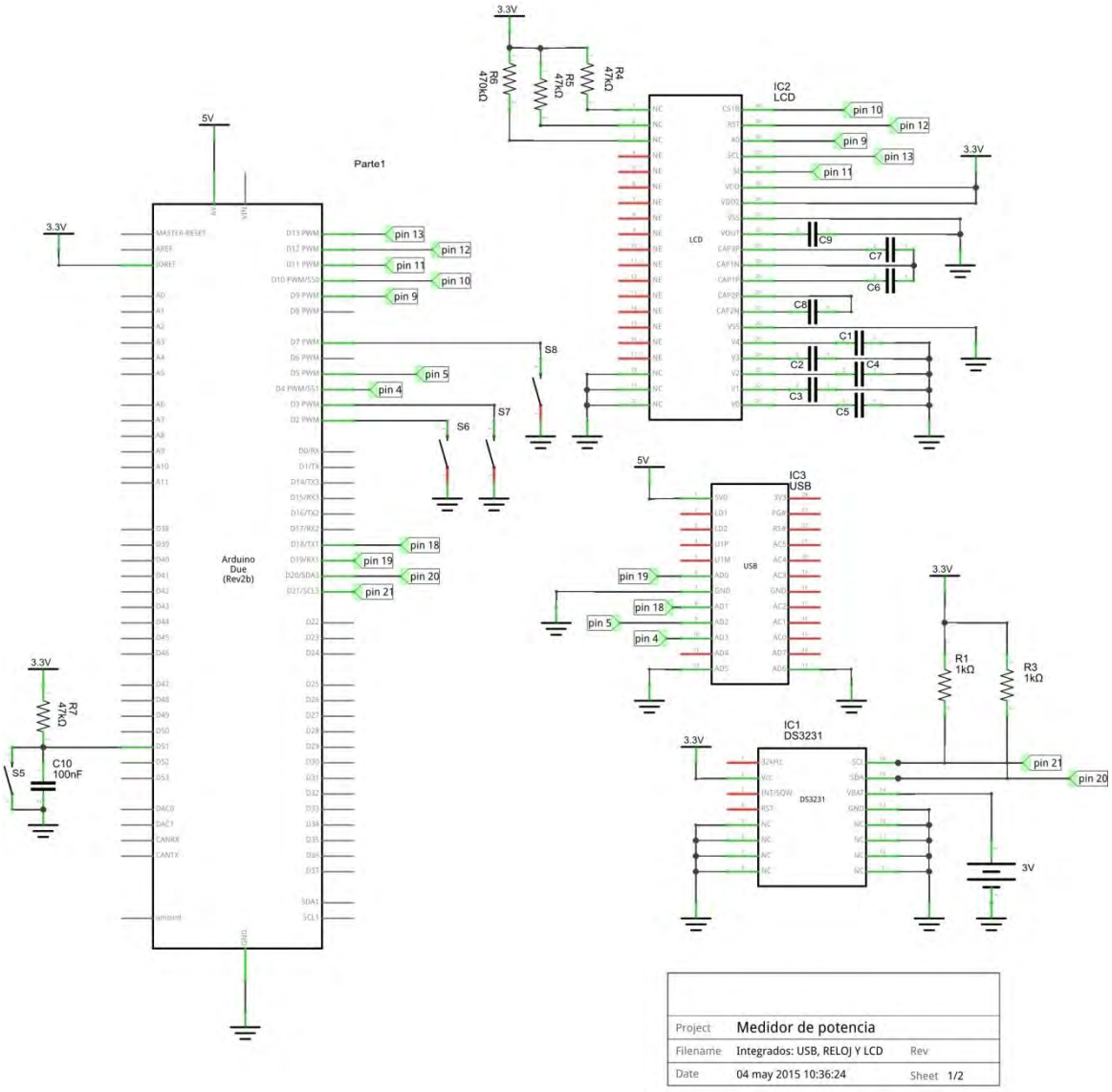
Circuito de acondicionamiento de tensión				
Dispositivo	Componente	Unidades	Precio/Unidad €	Coste total €
Amp. Instrumentación	AD623ANZ	3	4,41	13,23
VCO	AD7741BNZ	3	5,24	15,72
Buffer	74HC244	1	0,39	0,39
Optoacoplador	ACPL-W70L	3	1,83	5,49
Cristal	Oscilador	3	0,65	1,95
Resistencias	-	26	0,05	1,3
Condensadores	-	13	0,025	0,325
	TOTAL €			38,405
Visualización, almacenamiento de datos y reloj				
Dispositivo	Componente	Unidades	Precio/Unidad €	Coste total €
LCD Gráfico	EA DOGM128W-6	1	15,9	15,9
Retroiluminado	EA LED55x46-W	1	16,44	16,44
Puerto USB	VDIP1	1	22,43	22,43
Reloj	DS3231SN#	1	8,53	8,53
Pulsador	8MS9P1B07M7RES	4	1,14	4,56
Resistencias	-	6	0,05	0,3
Condensadores	-	10	0,025	0,25
Pila 3V	CR-2032/GUN	1	1,2	1,2
	TOTAL €			69,61
Circuito de acondicionamiento de corriente				
Dispositivo	Componente	Unidades	Precio/Unidad €	Coste total €
Transductor de corriente	LTSR 6-NP	3	14,32	42,96
Procesamiento de datos				
Dispositivo	Componente	Unidades	Precio/Unidad €	Coste total €
Microcontrolador	Arduino Due	1	41,16	41,16
PRESUPUESTO TOTAL €				192,135

Tabla 20: Presupuesto general del proyecto

Parte IV ANEXOS

1 Anexo I

Circuitos de los módulos de USB, LCD y RTC.

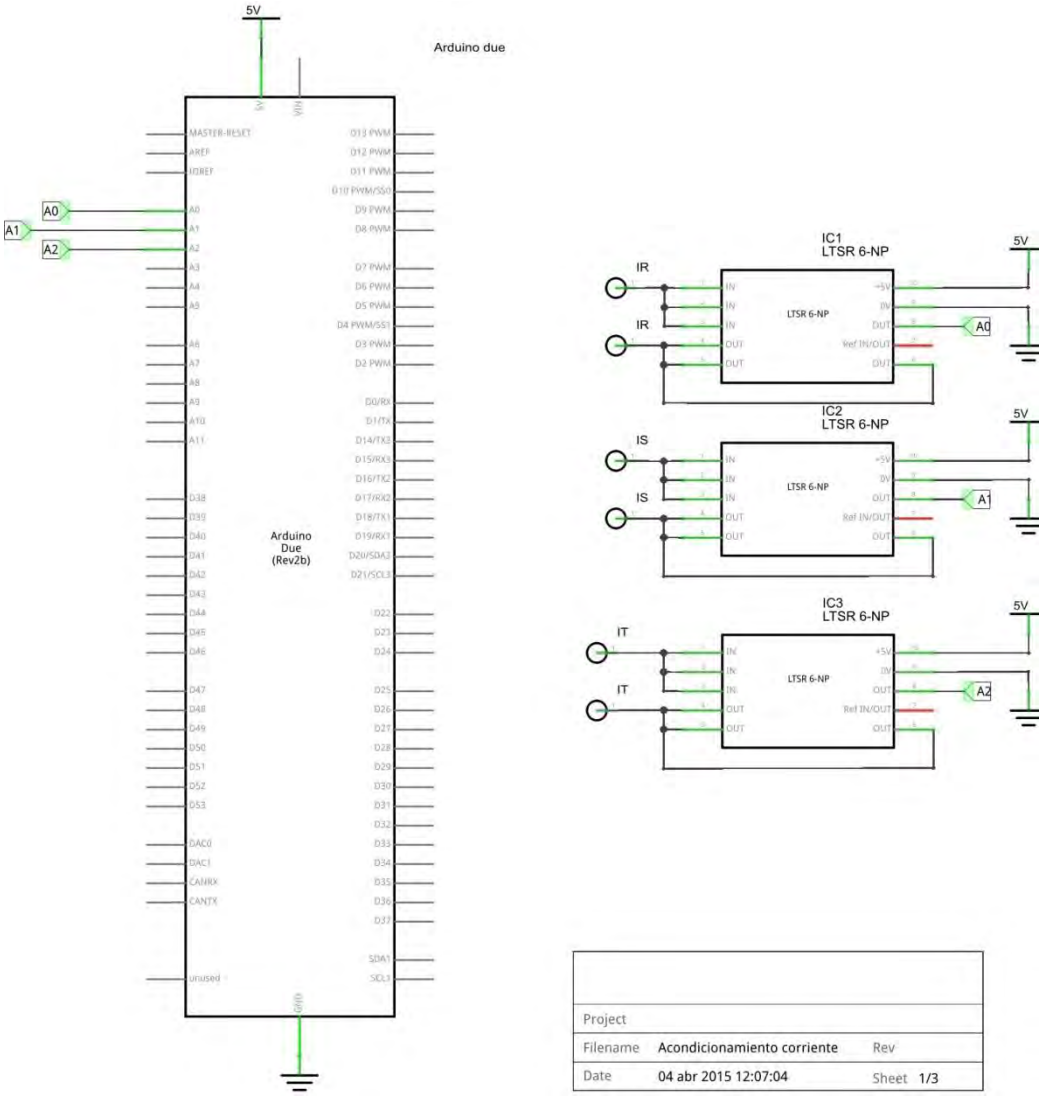


Lista de elementos:

- S5: Pulsador de activación de pantalla.
- S6: Pulsador de selección de método de cálculo en menú.
- S7: Pulsador de desplazamiento inferior en menú.
- S8: Pulsador de desplazamiento superior en menú.
- C1…C9: Condensadores cerámicos de valor 1µF.

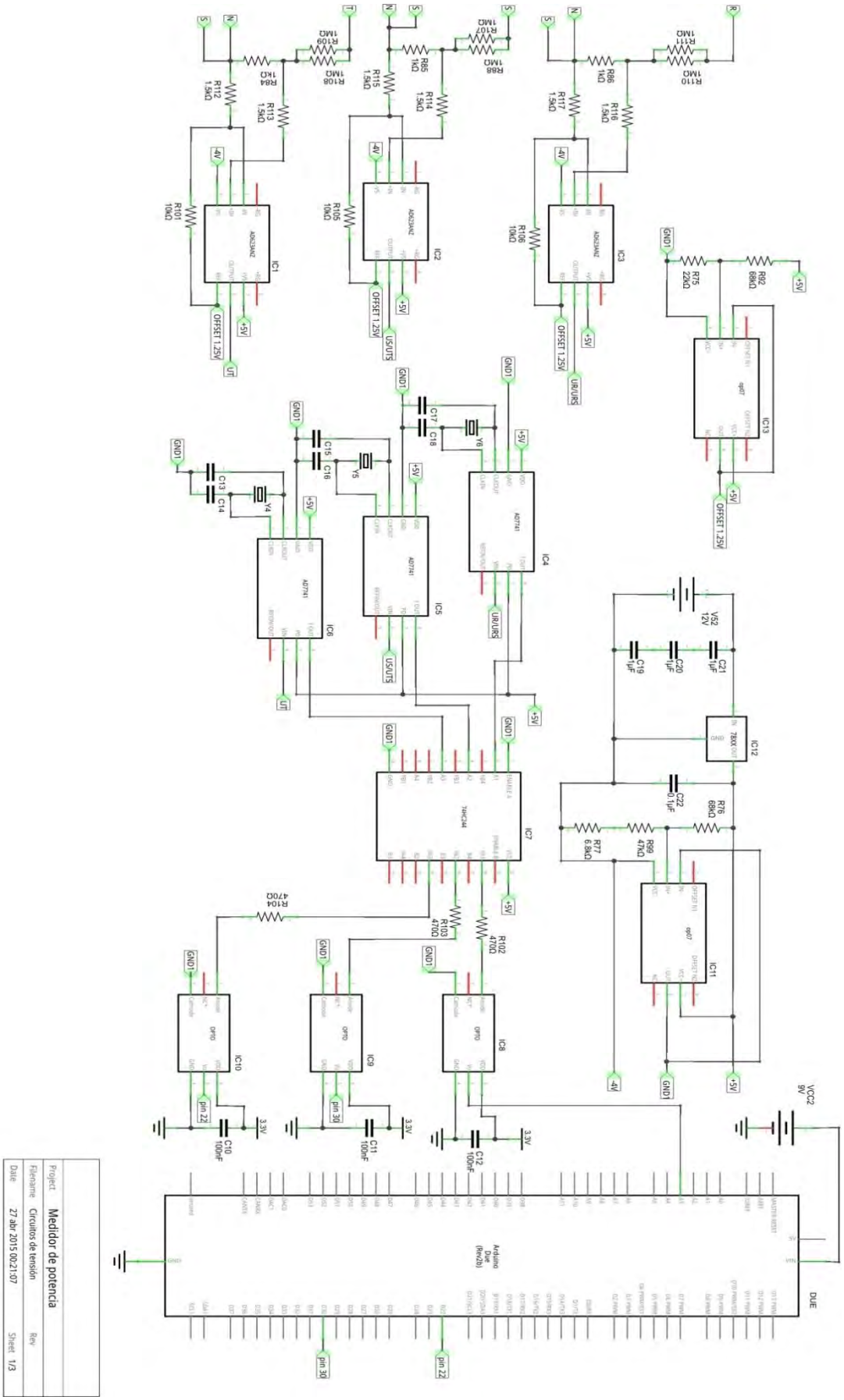
2 Anexo II

Circuito del acondicionamiento de la señal de corriente.



3 Anexo III

Circuito de acondicionamiento de las señales de tensión.


Project: Medidor de potencia
Filename: Circuitos de tensión
Rev:
Date: 27 abr 2015 00:21:07
Sheet 1/3
IC1
IC2
IC3
IC4
IC5
IC6
IC7
IC8
IC9
IC10
IC11
IC12
IC13
OFFSET 1.25V
US/UTS
UR/URS
UT
+5V
-4V
GND1
3.3V
VCC2
9V
DUE
Arduino Due (Rev2b)
pin 22
pin 30