



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

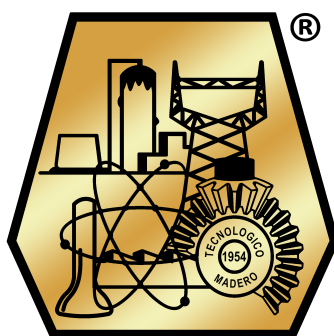


TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN INGENIERÍA ELÉCTRICA



"POR MI PATRIA Y POR MI BIEN"

TESIS

**ANÁLISIS DE LA CALIDAD DE LA ENERGÍA UTILIZANDO
SOFTWARE DE COMPUTACIÓN NUMÉRICA Y UN SISTEMA DE
ADQUISICIÓN DE DATOS**

Que para obtener el Grado de
Maestro en Ingeniería Eléctrica

Presenta

Ing. Luis Felipe Miranda Pérez

G17070913

CVU 1315671

Director de la Tesis

M.C. Hermenegildo Cisneros Villegas

CVU 63302

Co-director de la Tesis

Dr. Gastón Hernández Martínez



Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Ciudad Madero
Subdirección Académica
División de Estudios de Posgrado e Investigación

Ciudad Madero, Tamaulipas, **21/Noviembre/2025**

Oficio No.: U2.060/25

Asunto: Autorización de impresión de tesis

C. LUIS FELIPE MIRANDA PÉREZ
No. DE CONTROL G17070913
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ingeniería Eléctrica, se acordó autorizar la impresión de su tesis titulada:

"ANÁLISIS DE LA CALIDAD DE LA ENERGÍA UTILIZANDO SOFTWARE DE COMPUTACIÓN NUMÉRICA Y UN SISTEMA DE ADQUISICIÓN DE DATOS"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTE:	M.C. HERMENEGILDO CISNEROS VILLEGAS
SECRETARIO:	DR. GASTÓN HERNÁNDEZ MARTÍNEZ
VOCAL:	M.C. RAFAEL CASTILLO GUTIÉRREZ
SUPLENTE:	M.C. AARON GONZÁLEZ RODRÍGUEZ

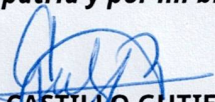
DIRECTOR DE TESIS:	M.C. HERMENEGILDO CISNEROS VILLEGAS
CO-DIRECTOR:	DR. GASTÓN HERNÁNDEZ MARTÍNEZ

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

ATENTAMENTE

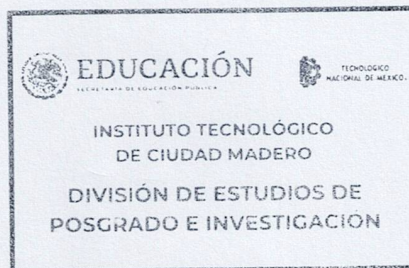
Excelencia en Educación Tecnológica®

"Por mi patria y por mi bien"®


RAFAEL CASTILLO GUTIERREZ
JEFE DE LA DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

ccp. Archivo

RCG/ RCI/aecr



2025
Año de
La Mujer
Indígena

Av. 1° de Mayo y Sor Juana I. de la Cruz S/N Col. Los Mangos
C.P. 89440 Cd. Madero, Tam. Tel. (833) 3574820 ext. 3110
e-mail: depi_cdmadero@tecnm.mx www.tecnm.mx
www.cdmadero.tecnm.mx



Dedicatoria

A mis padres, quienes con dedicación constante, cariño inagotable y renunciias generosas me han acompañado en cada momento de mi camino. Su modelo de perseverancia y entrega es el fundamento de mis éxitos. Esta obra es un homenaje a su esfuerzo y a su afecto, siempre presente. ¡Gracias!

Agradecimientos

Agradezco de corazón al Tecnológico Nacional de México de Ciudad Madero por abrirme las puertas para cursar este posgrado en sus instalaciones, y por la excelencia académica que distingue con orgullo su programa de maestría.

Un reconocimiento profundo a los profesores, quienes fueron pilares esenciales en mi desarrollo profesional. Con inmensa generosidad, compartieron su tiempo, experiencia y sabiduría en el campo de la Ingeniería Eléctrica, enriqueciendo mi formación tanto dentro como fuera del aula.

Extiendo mi más sincero agradecimiento a mi director de tesis, el M.C. Hermenegildo Cisneros Villegas, por su invaluable apoyo en este proyecto, así como al Dr. Gastón Hernández Martínez por sus valiosos consejos y orientación.

Un agradecimiento especial a Secithi por haberme otorgado un apoyo económico que hizo posible mi estancia en la maestría.

Por último, agradezco con todo el corazón a mis compañeros de maestría por su extraordinaria calidad humana y por los momentos tan enriquecedores que compartimos.

Resumen

En la presente investigación, se implementó un diseño modular de un dispositivo capaz de medir armónicos impares hasta la décima quinta armónica. Este enfoque modular ofrece una solución práctica y accesible para el análisis de armónicos.

Para garantizar la compatibilidad de las señales con el microcontrolador del sistema de adquisición de datos, se ensamblaron circuitos específicos de acondicionamiento para corriente y voltaje. Además, se integró un circuito de comunicación USB serial a TTL, que establece un enlace confiable entre el hardware de adquisición y el software de procesamiento.

Finalmente, se desarrolló una interfaz de usuario en Scilab que permite visualizar y analizar el contenido armónico en corriente de una red polifásica de hasta tres fases. Se logró con éxito la medición de voltaje y corriente, la conexión estable entre el sistema de adquisición y la interfaz, así como el procesamiento completo de los datos.

Abstract

In the present research, a modular design was implemented for a device capable of measuring odd harmonics up to the fifteenth harmonic. This modular approach offers a practical and accessible solution for harmonic analysis.

To ensure signal compatibility with the microcontroller of the data acquisition system, specific conditioning circuits for current and voltage were assembled. Additionally, a USB-to-TTL serial communication circuit was integrated, establishing a reliable link between the acquisition hardware and the processing software.

Finally, a user interface was developed in Scilab that enables the visualization and analysis of harmonic content in the current of a polyphase network with up to three phases. Successful measurement of voltage and current was achieved, along with a stable connection between the acquisition system and the interface, as well as complete data processing.

Índice general

Dedicatoria	I
Agradecimientos	II
Resumen	III
Abstract	IV
Índice general	V
Lista de acrónimos	IX
Lista de tablas	X
Lista de figuras	XI
1. Introducción	1
1.1. Introducción	1
1.2. Planteamiento del Problema	1
1.3. Justificación	3
1.4. Objetivos	3
1.4.1. Objetivo General	3
1.4.2. Objetivos específicos	3
1.5. Hipótesis	4
1.6. Alcances y Limitaciones	4
1.6.1. Alcances	4
1.6.2. Limitaciones	4
1.6.3. Metodología	4

1.6.4. Diagrama de bloques	5
2. Marco teórico	6
2.1. Sistema de adquisición de datos	6
2.2. Software de computación numérica	7
2.3. Transformada de Fourier discreta	8
2.4. Teorema de muestreo de Nyquist-Shannon	9
2.5. Clasificación de las armónicas	9
2.6. Parámetros de medida de armónicos	10
2.6.1. Voltaje armónico	10
2.6.2. Corriente armónica	10
2.6.3. Distorsión armónica total (THD)	10
2.6.4. Distorsión armónica total de corriente (THDi)	10
2.6.5. Distorsión armónica total de voltaje (THDv)	11
2.7. Potencias en el dominio armónico	11
2.7.1. Potencia activa	11
2.7.2. Potencia reactiva	11
2.7.3. Potencia aparente	11
2.8. Factor de potencia	11
2.9. Factor de potencia de desplazamiento	12
2.10. Factor de potencia de distorsión	12
2.11. Factor de potencia verdadero (total)	12
2.12. Calidad de la energía	13
2.12.1. Principales disturbios en la calidad de la energía	13
2.13. Factor de calidad Q	14
2.14. Resonancia en bancos de capacitores	14
2.15. Efectos de los armónicos	15
2.15.1. Tercera armónica	15
2.16. Instalación de capacitores en redes con armónicos	16
2.17. Causas y efectos de los armónicos	16
2.18. Soluciones para compensación en redes con armónicos	17
2.19. Estándares de calidad de la energía	17
2.19.1. CEI 61000-4-30	17

2.19.2. CEI 61000-4-7	17
2.19.3. CEI 61000-4-15	18
2.19.4. EN 50160	18
2.19.5. IEEE 1159	18
2.19.6. IEEE 519	18
2.20. Analizador de calidad de energía	18
3. Caracterización del Problema: Simulaciones y Mediciones	20
3.1. Clasificación y simulación de armónicos pares e impares	20
3.2. Simulaciones de resonancia en sistemas eléctricos	27
3.3. Mediciones de armónicos en redes reales utilizando analizador Fluke	32
4. Selección de Componentes y Tecnologías para el Sistema de Análisis de Calidad de Energía	35
4.1. Sondas de corriente y voltaje	35
4.2. Adecuación de señales	39
4.3. Sistema de adquisición de datos	41
4.4. Selección de software	42
5. Elaboración de la Solución para el Sistema de Análisis de Calidad de Energía	44
5.1. Prototipos en tablilla de experimentos: desarrollo inicial para acondicionamiento de señales y sistema DAQ con ATmega328P y ESP32	44
5.2. Diseño de PCB en KiCad: para ATmega328P, ESP32 y acondicionamiento de voltajes y corrientes	46
5.3. Maquinado de PCB con máquina CNC	53
5.4. Interfaz de usuario: desarrollo en Scilab con integración Python	54
5.5. Programa gui.sci	55
5.6. Programa filtrado.py	67
5.7. Programa comunicacion.py	69
5.8. Programa daq.c	73
5.9. Banco de pruebas y calibración	79
6. Resultados y Conclusiones	81
6.1. Medición en banco de pruebas	81

6.2. Conclusiones	85
7. Trabajos Futuros	86
Bibliografía	87

Lista de Acrónimos

ADC	Convertidor analógico - digital. 47, 50, 51, 53, 56, 66, 87, 91
CA	Corriente Alterna. 25, 49
CD	Corriente Directa. 49–52
CNC	Control Numerico Computacional. 67
CSV	Valores separados por comas. 75, 76, 81, 83, 85–87
CT	Transformador de corriente. 49, 50
DAQ	Adquisición de datos. 18, 53–56, 58, 63–66, 85, 99, 100, XI, XIII
DFT	Transformada discreta de Fourier. 20, 21
DPF	Factor de Potencia de Desplazamiento. 24
FFT	Transformada Rapida de Fourier. 52, 54, 55, 70, 81
FP	Factor de potencia. 24
GPL	Licencia Publica General de GNU. 55
GUI	Interfaz grafica de usuario. 55, 68, 98, 100, XIII
IDE	Entorno de Desarrollo Integrado. 55
IDFT	Transformada inversa de Fourier discreta. 20, 21
IEEE	Instituto de Ingenieros Elécticos y Electrónicos. 22, 39
LED	Diodo Emisor de Luz. 17, 93
op-amp	Amplificador Operacional. 52, 56, 66
PCB	Placa de Circuito Impreso. 57, 58, 67
PDF	Formato de Documento Portatil. 100
THD	Distorsión armónica total. 24, 30, 46, 52, 77, 98
UPS	Sistema de Alimentacion Ininterrumpida. 24
VFD	Variador de frecuencia. 24

Lista de Tablas

2.1. Cargas y sus armónicos generados.	16
5.1. Lista de componentes del circuito de adecuación de corrientes.	48
5.2. Lista de componentes del circuito de adecuación de voltajes.	49
5.3. Lista de componentes Adquisición de datos (DAQ) V2 con ATmega328P. . . .	51
5.4. Lista de componentes DAQ V3 con ESP32.	52
6.1. Valores de distorsión armónica y factor de potencia obtenidos con el prototipo.	85

Lista de Figuras

1.1. Efectos de los armónicos y el bajo factor de potencia.	2
1.2. Explosión de banco de capacitores por presencia de armónicos.	2
1.3. Diagrama de bloques	5
2.1. Tarjeta de adquisición de datos NI USB-6009.	7
2.2. Analizador de calidad de la energía eléctrica - C.A 8345.	19
3.1. Simulación de distorsión por la presencia de segunda armónica.	21
3.2. Forma de onda distorsionada por la presencia de segunda armónica.	21
3.3. Simulación de distorsión por la presencia de cuarta armónica.	21
3.4. Forma de onda distorsionada por la presencia de cuarta armónica.	22
3.5. Simulación de distorsión por la presencia de sexta armónica.	22
3.6. Forma de onda distorsionada por la presencia de sexta armónica.	22
3.7. Simulación de distorsión por la presencia de octava armónica.	23
3.8. Forma de onda distorsionada por la presencia de octava armónica.	23
3.9. Simulación de distorsión por la presencia de tercera armónica.	24
3.10. Forma de onda distorsionada por la presencia de tercera armónica.	24
3.11. Simulación de distorsión por la presencia de quinta armónica.	24
3.12. Forma de onda distorsionada por la presencia de quinta armónica.	25
3.13. Simulación de distorsión por la presencia de séptima armónica.	25
3.14. Forma de onda distorsionada por la presencia de séptima armónica.	25
3.15. Simulación de distorsión por la presencia de novena armónica.	26
3.16. Forma de onda distorsionada por la presencia de novena armónica.	26
3.17. Simulación de distorsión por la presencia de undécima armónica.	26
3.18. Forma de onda distorsionada por la presencia de undécima armónica.	27
3.19. Circuitos modelados con diferentes valores de resistencia.	28
3.20. Circuito modelado sin resistencia.	29
3.21. Gráfica de voltaje y corriente con resistencia de 100Ω	30
3.22. Gráfica de voltaje y corriente con resistencia de 50Ω	30
3.23. Gráfica de voltaje y corriente con resistencia de 10Ω	31
3.24. Gráfica de voltaje y corriente con resistencia de 0Ω	31
3.25. Instalación de las sondas de medición en el centro de carga.	32
3.26. Porcentaje de armónicos en corriente y voltaje.	33
3.27. Gráfico de armónicos en corriente.	33
3.28. Gráfico de armónicos en voltaje.	34
4.1. Transformador de corriente SCT013.	36
4.2. Señal obtenida por el sensor de corriente SCT013.	36
4.3. Bobina Rogowski ETCR50FA.	37

4.4. Señal obtenida por la bobina Rogowski ETCR50FA.	37
4.5. Sensor de corriente de efecto Hall ACS712.	38
4.6. Diagrama para la adecuación de señales de voltaje (una sola fase).	39
4.7. Diagrama para la adecuación de señales de corriente (una sola fase).	40
4.8. Diagrama para el sistema de adquisición de datos primera versión.	41
4.9. Diagrama para el sistema de adquisición de datos segunda versión.	42
5.1. Circuito experimental del sistema de adquisición de datos en su primera versión.	45
5.2. Circuito experimental del sistema de adecuación de señales de voltaje.	45
5.3. Circuito experimental del sistema de adecuación de señales de corriente.	46
5.4. Enrutado de pistas del sistema de adecuación de señales de corriente.	47
5.5. Vista 3D del sistema de adecuación de señales de corriente.	47
5.6. Enrutado de pistas del sistema de adecuación de señales de voltaje.	48
5.7. Vista 3D del sistema de adecuación de señales de voltaje.	49
5.8. Enrutado de pistas del sistema DAQ con ATmega328P.	50
5.9. Vista 3D del sistema DAQ con ATmega328P.	50
5.10. Enrutado de pistas del sistema DAQ con ESP32.	51
5.11. Vista 3D del sistema DAQ con ESP32.	52
5.12. Módulos ensamblados y soldados.	53
5.13. Diagrama de conexión entre software y hardware.	54
5.14. Primera versión de la interfaz gráfica de usuario.	55
5.15. Diagrama de flujo de la interacción entre códigos.	79
5.16. Banco de pruebas trifásico.	80
6.1. Medición de corrientes trifásicas en la red de pruebas con el Fluke 434.	81
6.2. Formas de onda de corriente capturadas por el Fluke 434.	82
6.3. Contenido armónico total registrado por el Fluke 434.	82
6.4. Distribución de armónicos individuales obtenida con el Fluke 434.	83
6.5. Factor de potencia por fase medido con el Fluke 434.	83
6.6. Medición de corrientes trifásicas con el prototipo desarrollado.	84
6.7. Interfaz gráfica de usuario (Interfaz grafica de usuario (GUI)) en Scilab mostrando resultados en tiempo real.	84

Capítulo 1

Introducción

1.1. Introducción

El análisis de la calidad de la energía es un tema importante en el campo de la ingeniería eléctrica. Se refiere al estudio de las condiciones normales de operación de los sistemas eléctricos en cuanto a estabilidad, distorsión de la forma de onda y continuidad del suministro de energía eléctrica. Asegurar altos niveles de calidad requiere el estudio de la influencia de la conexión adecuada de los sistemas y la detección de perturbaciones que puedan afectar el suministro eléctrico.

1.2. Planteamiento del Problema

La calidad de energía es crucial en la industria moderna, especialmente en procesos automatizados. La baja calidad de energía puede afectar los procesos automatizados, provocar interrupciones en el suministro de energía eléctrica, aumento de los costos en general, así como puede dañar los equipos y, en algunos casos, interrumpir procesos críticos.

Por lo tanto, se tiene la necesidad de disminuir esos problemas mediante el análisis de la red, es indispensable contar con un método flexible para medir la calidad de la energía que pueda ser adaptable a diversos tipos de escenarios.

Entre los problemas que afectan la calidad de la energía están los armónicos y el bajo factor de potencia, estos se pueden presentar en tiempos prolongados, provocando calenta-

miento en los conductores y daño en los equipos, entre otras afectaciones.

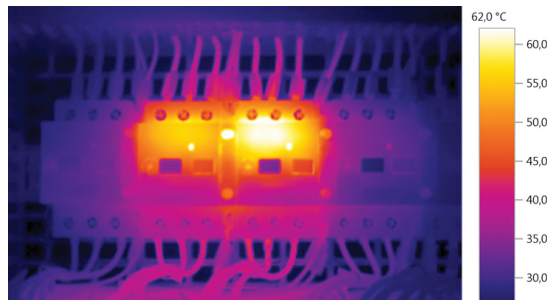


Figura 1.1: Efectos de los armónicos y el bajo factor de potencia.



Figura 1.2: Explosión de banco de capacitores por presencia de armónicos.

Para confirmar la presencia de estos problemas en la red eléctrica se utiliza un analizador de calidad de la energía; Este dispositivo permite detectar y registrar perturbaciones eléctricas, realizar análisis de tendencias y verificar la calidad del suministro eléctrico conforme a las normas aplicables en el código de red, el no cumplir con las normas implicaría recibir multas económicas o incluso la desconexión de la red.

1.3. Justificación

El mayor inconveniente del analizador de calidad de la energía es su elevado costo, dificultando la formación de personal que pueda manejar e interpretar los datos obtenidos por el dispositivo.

Se pretende buscar una alternativa económica que pueda ser útil para la formación de estudiantes.

Para poder realizar el análisis de la calidad de la energía se optó por una interfaz en software de computación numérica para hacer el procesamiento de los datos, un sistema de adquisición de datos, en conjunto con sondas no invasivas de corriente y sondas de voltaje.

1.4. Objetivos

1.4.1. Objetivo General

Medir el factor de potencia y la relación de armónicos de una red eléctrica utilizando una interfaz diseñada en software de computación numérica, en conjunto con una tarjeta de adquisición de datos, sondas de corriente y voltaje, y realizar un análisis de los datos obtenidos.

1.4.2. Objetivos específicos

- Realizar mediciones con el analizador de redes Fluke 434.
- Probar distintos tipos de sondas de corriente y voltaje.
- Investigar distintos sistemas de cómputo numérico.
- Diseñar una placa de circuito impreso para el sistema de adquisición de datos.
- Realizar la calibración del dispositivo usando como instrumento patrón el analizador de redes Fluke 434.
- Aprender a interpretar los datos obtenidos en las mediciones.

1.5. Hipótesis

Es posible realizar un análisis de la calidad de la energía utilizando software de computación numérica para el procesamiento de los datos obtenidos con un sistema de adquisición de datos y sondas de voltaje y corriente.

El sistema de adquisición de datos y el software de cómputo numérico son capaces de procesar el volumen de datos.

1.6. Alcances y Limitaciones

1.6.1. Alcances

Medir el factor de potencia y la relación de armónicos de una red eléctrica, utilizando focos tipo LED de bajo consumo como cargas de prueba.

1.6.2. Limitaciones

Se pretende medir una red de hasta tres fases, tomando en cuenta el sistema eléctrico disponible en el Laboratorio de la Maestría en Ingeniería Eléctrica del Instituto Tecnológico de Ciudad Madero.

Obtención de armónicos en corriente hasta la décimo quinta armónica.

1.6.3. Metodología

El proyecto se dividió en tres etapas.

En la primera etapa se tuvo la adquisición de señales, lo que comprenderá la elección de tarjeta de adquisición de datos a utilizar así como la realización de pruebas con diferentes tipos de sondas de voltaje y corriente y la selección del software de cómputo numérico en el

que se realizará procesamiento de los datos.

La segunda etapa del proyecto consistió en las pruebas de medición utilizando las herramientas elegidas en la primera etapa. Se realizará una serie de pruebas con Lámparas tipo Diodo Emisor de Luz (LED) de bajo consumo. Así mismo se contemplará la adecuación de las herramientas elegidas con el propósito de realizar la calibración del sistema usando como instrumento patrón el analizador de redes Fluke 434 perteneciente al Laboratorio de la Maestría en Ingeniería Eléctrica del Instituto Tecnológico de Ciudad Madero.

En la tercera etapa del proyecto, se realizó el análisis de los datos obtenidos y se comparó con un analizador de redes para evaluar el margen de error de los datos obtenidos.

1.6.4. Diagrama de bloques

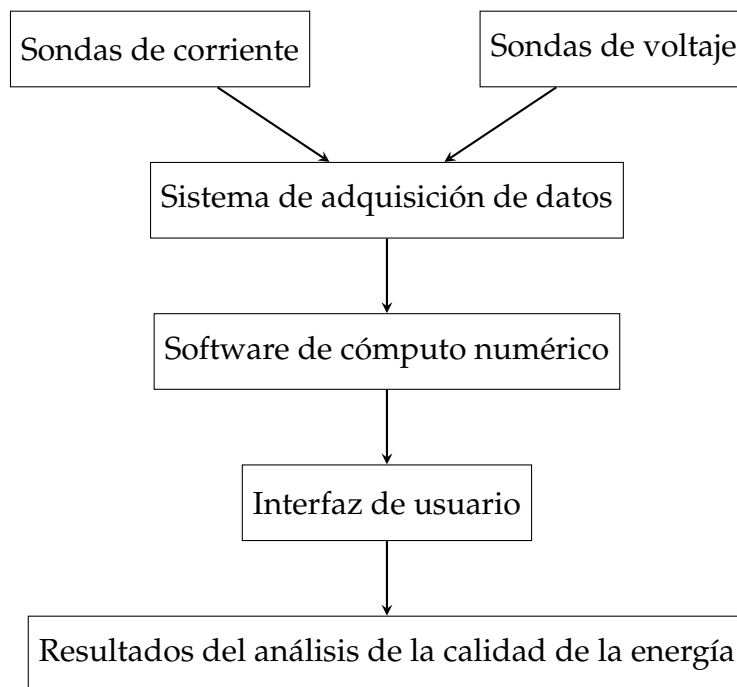


Figura 1.3: Diagrama de bloques

Capítulo 2

Marco teórico

2.1. Sistema de adquisición de datos

La adquisición de datos, comúnmente conocida por sus siglas en inglés DAQ (*data acquisition*), se refiere al proceso de medir variables físicas mediante una computadora. Un sistema DAQ se compone de tres elementos principales: los sensores, la tarjeta de adquisición de datos (hardware) y una computadora (software programable).

Gracias a su conexión con una computadora, se obtienen numerosos beneficios, como el aprovechamiento de la potencia del procesador, la visualización de datos y la conectividad, lo que ofrece soluciones más accesibles en comparación con equipos especializados, tales como los analizadores de red.

Las tarjetas de adquisición de datos (hardware) actúan como interfaz entre una computadora y señales analógicas. Es decir, la información recopilada por el sensor se transmite al DAQ, el cual transforma la señal analógica en una señal digital. [1]



Figura 2.1: Tarjeta de adquisición de datos NI USB-6009.

2.2. Software de computación numérica

El cómputo numérico se refiere al área de las ciencias de la computación que se ocupa de los métodos y algoritmos para realizar cálculos numéricos. Existen varios programas y bibliotecas de software especializados en este campo.

- **MATLAB:** Esta plataforma de programación está diseñada específicamente para ingenieros y científicos. Se trata de un lenguaje basado en matrices que permite la expresión más natural de las matemáticas computacionales. Combina un entorno de escritorio optimizado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa directamente las matemáticas de matrices y vectores. Los paquetes de herramientas de MATLAB se desarrollan de forma profesional, pasan pruebas rigurosas y están totalmente documentados. [2]
- **GNU Octave:** Se considera el equivalente libre de MATLAB. Entre las características compartidas, destaca que ambos ofrecen un intérprete que permite ejecutar órdenes en modo interactivo. Octave es un lenguaje de alto nivel para realizar cálculos numéricos en el ordenador y también un programa capaz de interpretar este lenguaje y ejecutar

los cálculos. Ofrece una interfaz de usuario interactiva orientada a línea de comandos, aunque también puede utilizarse en modo no interactivo, leyendo órdenes desde un fichero. Octave es software libre bajo licencia GNU, lo que significa que se puede usar y redistribuir libremente, y que cualquiera puede contribuir a su mejora. [3]

- Scilab: Este software de código abierto proporciona un entorno de programación para aplicaciones científicas y de ingeniería. Se utiliza para resolver problemas matemáticos y realizar cálculos numéricos, y ofrece un entorno interactivo para el análisis de datos y la modelización. Scilab es similar a MATLAB en muchos aspectos y se emplea para cálculos numéricos complejos, modelado y simulación. [4]

2.3. Transformada de Fourier discreta

La transformada discreta de Fourier, o Transformada discreta de Fourier (DFT) (*discrete Fourier transform* en inglés), es un tipo de transformada discreta utilizada en el análisis de Fourier. Esta transformada convierte una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, mientras que la función original se encuentra en el dominio del tiempo. Sin embargo, la DFT requiere que la función de entrada sea una secuencia discreta y de duración finita. Tales secuencias suelen generarse a partir del muestreo de una función continua, como la voz humana.

La secuencia de N números complejos $x_0, \dots, x_{N-1}$ se transforma en la secuencia de N números complejos $X_0, \dots, X_{N-1}$ mediante la DFT según la fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1. \quad (2.1)$$

Donde i es la unidad imaginaria y $e^{\frac{2\pi i}{N}}$ es la N -ésima raíz de la unidad. (Esta expresión también puede escribirse en términos de una matriz DFT; cuando se escala adecuadamente, se convierte en una matriz unitaria y X_k puede interpretarse como los coeficientes de x en una base ortonormal.)

La transformada se denota a veces por el símbolo $\mathcal{F}$, como en $\mathbf{X} = \mathcal{F}\{\mathbf{x}\}$ o $\mathcal{F}(\mathbf{x})$ o $\mathcal{F}\mathbf{x}$.

La transformada inversa de Fourier discreta, o Transformada inversa de Fourier discreta

(IDFT) (*inverse discrete Fourier transform* en inglés), se da por:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N} kn} \quad n = 0, \dots, N-1. \quad (2.2)$$

Una descripción simple de estas ecuaciones es que los números complejos X_k representan la amplitud y fase de diferentes componentes sinusoidales de la señal de entrada x_n . La DFT calcula X_k a partir de x_n , mientras que la IDFT muestra cómo calcular x_n como la suma de componentes sinusoidales $(1/N)X_k e^{\frac{2\pi i}{N} kn}$ con una frecuencia de k/N ciclos por muestra. [5]

2.4. Teorema de muestreo de Nyquist-Shannon

El teorema demuestra que la reconstrucción exacta de una señal continua en banda base a partir de sus muestras es matemáticamente posible si la señal está limitada en banda y la tasa de muestreo es superior al doble de su ancho de banda.

Dicho de otro modo, la información completa de la señal analógica original que cumple el criterio anterior está descrita por la serie total de muestras resultantes del proceso de muestreo. Por tanto, no hay nada en la evolución de la señal entre muestras que no esté perfectamente definido por la serie total de muestras.

Si la frecuencia más alta contenida en una señal analógica $x_a(t)$ es $F_{max} = B$ y la señal se muestrea a una tasa $F_s > 2F_{max} \equiv 2B$, entonces $x_a(t)$ se puede recuperar totalmente a partir de sus muestras mediante la siguiente función de interpolación:

$$g(t) = \frac{\sin 2\pi Bt}{2\pi Bt} \quad (2.3)$$

Así, $x_a(t)$ se puede expresar como:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a\left(\frac{n}{F_s}\right) g\left(t - \frac{n}{F_s}\right) \quad (2.4)$$

donde $x_a\left(\frac{n}{F_s}\right) = x_a(nT) \equiv x(n)$ son las muestras de $x_a(t)$. [6]

2.5. Clasificación de las armónicas

Las armónicas son señales de tensión y/o corriente cuya frecuencia es un múltiplo entero de la fundamental. Estas armónicas se suman a la señal fundamental y la distorsionan, generando lo que comúnmente se conoce como distorsión armónica, según la definición del

(Instituto de Ingenieros Eléctricos y Electrónicos (IEEE)).

Las interarmónicas son tensiones o corrientes cuya frecuencia es un múltiplo no entero de la frecuencia fundamental de suministro. La norma IEC 61000-2-1 define las interarmónicas de la siguiente manera: entre los armónicos de frecuencia de la tensión y de la corriente de la red de alimentación, se pueden observar otras frecuencias que no son un múltiplo entero de la fundamental. Estas frecuencias pueden aparecer como frecuencias discretas o como un espectro de banda ancha.

El término "subarmónica" no tiene una definición oficial; se trata de un caso particular de interarmónica cuya frecuencia es menor que la frecuencia fundamental.

2.6. Parámetros de medida de armónicos

2.6.1. Voltaje armónico

$$v(t) = \sum_{h=1}^{\infty} V_h \cos(h\omega_0 + \theta_h) \quad (2.5)$$

2.6.2. Corriente armónica

$$i(t) = \sum_{h=1}^{\infty} I_h \cos(h\omega_0 + \phi_h) \quad (2.6)$$

2.6.3. Distorsión armónica total (THD)

$$THD = \frac{1}{F_1} \sqrt{\sum_{h=2}^{\infty} F_h^2} \quad (2.7)$$

Este parámetro permite evaluar la distorsión de la forma de onda con un solo número.

2.6.4. Distorsión armónica total de corriente (THDi)

$$THDi = \frac{1}{I_1} \sqrt{\sum_{h=2}^{\infty} I_h^2} \quad (2.8)$$

$$I_{RMS} = I_{1_{RMS}} \sqrt{1 + THDi^2} \quad (2.9)$$

La THDi es un porcentaje del valor eficaz de las corrientes armónicas de rangos superiores a uno en relación con el valor eficaz de la corriente fundamental.

2.6.5. Distorsión armónica total de voltaje (THDv)

$$THDv = \frac{1}{V_1} \sqrt{\sum_{h=2}^{\infty} V_h^2} \quad (2.10)$$

$$V_{RMS} = V_{1RMS} \sqrt{1 + THDv^2} \quad (2.11)$$

La THDv es un porcentaje del valor eficaz de los voltajes armónicos de rangos superiores a uno en relación con el valor eficaz del voltaje fundamental.

2.7. Potencias en el dominio armónico

2.7.1. Potencia activa

$$P = \sum_{h=1}^{\infty} (V_{hRMS})(I_{hRMS}) \cos(\theta_h - \phi_h) \quad (2.12)$$

2.7.2. Potencia reactiva

$$Q = \sum_{h=1}^{\infty} (V_{hRMS})(I_{hRMS}) \sin(\theta_h - \phi_h) \quad (2.13)$$

2.7.3. Potencia aparente

$$S = V_{RMS} \cdot I_{RMS} \quad (2.14)$$

2.8. Factor de potencia

El factor de potencia de un circuito de corriente alterna se define como la relación entre la potencia activa y la potencia aparente. En forma de ecuación:

$$FP = \frac{\text{Potencia activa}}{\text{Potencia aparente}} \quad (2.15)$$

Si las ondas de corriente y tensión son perfectamente senoidales, el factor de potencia (FP) y el $\cos \phi$ coinciden. [7]

2.9. Factor de potencia de desplazamiento

Cuando las ondas de tensión y de corriente son senoidales y las cargas son lineales, como los motores de inducción o las resistencias, al factor de potencia se le denomina (Factor de Potencia de Desplazamiento (DPF), por sus siglas en inglés *displacement power factor*). [8]

2.10. Factor de potencia de distorsión

Sin embargo, los sistemas eléctricos modernos incluyen una gran cantidad de cargas pulsantes o no lineales, como equipos electrónicos (inversores, Variador de frecuencia (VFD) por *variable frequency drives*, Sistema de Alimentación Ininterrumpida (UPS) por *uninterruptible power supply*, rectificadores, etc.), en los que la potencia aparente excede considerablemente a la potencia activa. La definición anterior de Factor de potencia (FP) no es adecuada, ya que los armónicos impactan en el factor de potencia.

Por ello, se debe considerar la distorsión armónica total. Esta forma de factor de potencia se conoce como **factor de potencia de distorsión** (DPF, por sus siglas en inglés *distortion power factor*), y se define como la razón de la corriente de la frecuencia fundamental a la corriente real RMS, en forma de ecuación:

$$\text{FP de distorsión} = \frac{I_{\text{fundamental}}}{I_{\text{RMS}}} = \frac{1}{\sqrt{1 + \text{THD}^2}} \quad (2.16)$$

donde Distorsión armónica total (THD) se refiere a la distorsión armónica de corriente total. [9]

2.11. Factor de potencia verdadero (total)

El producto del factor de potencia de desplazamiento por el de distorsión da el **factor de potencia verdadero (total)**, en forma de ecuación:

$$\text{FP verdadero (total)} = \text{FP de desplazamiento} \times \text{FP de distorsión} \quad (2.17)$$

2.12. Calidad de la energía

La calidad de la energía se refiere a los fenómenos que afectan a los equipos cuando la energía en (Corriente Alterna (CA)) excede ciertos límites aceptables. Una calidad de energía deficiente puede causar problemas como el sobrecalentamiento de inductores, ruido en capacitores, fallas en sistemas de control basados en computadoras y paradas inesperadas de equipos sensibles. [7]

Los fabricantes deben diseñar equipos capaces de resistir niveles razonables de disturbios eléctricos. Sin embargo, los usuarios de equipos sensibles, como los utilizados en los siguientes sectores, requieren sistemas de acondicionamiento de energía para garantizar un suministro estable:

- Sistemas de información con equipos de cómputo.
- Departamentos de paramédicos y bomberos.
- Empresas públicas (agua, electricidad, etc.).
- Aeropuertos.
- Instituciones financieras.
- Entidades gubernamentales.

2.12.1. Principales disturbios en la calidad de la energía

Los disturbios más comunes que afectan la calidad de la energía incluyen: [8]

- **Pico de voltaje:** Incremento momentáneo del voltaje (en microsegundos) causado por fallas en la red, descargas atmosféricas o conmutación de grandes cargas.
- **Depresión de voltaje (sags):** Decremento momentáneo del voltaje (varios ciclos) debido a la conexión de cargas grandes o fallas en la red.
- **Dilatación de voltaje (swell):** Incremento prolongado del voltaje (varios ciclos) causado por la desconexión de cargas grandes.
- **Sobrevoltaje:** Condición de voltaje elevado sostenido, generalmente por una regulación deficiente.

- **Parpadeo (flicker):** Fluctuaciones en el nivel de voltaje causadas por cargas cíclicas, como hornos eléctricos, o por osciladores subarmónicos.
- **Interrupciones de energía:** Pérdida total de potencia, definida cuando el voltaje cae al 15 % o menos del valor nominal.
- **Ruido eléctrico:** Distorsión no periódica de la forma de onda senoidal, causada por conmutaciones o equipos industriales.
- **Distorsión armónica:** Distorsión periódica de la forma de onda, causada por cargas no lineales como rectificadores y hornos de arco eléctrico.

2.13. Factor de calidad Q

El factor de calidad Q describe el comportamiento de resonancia en un oscilador armónico subamortiguado. Un Q alto indica una resonancia con mayor amplitud, pero con un ancho de banda más estrecho. Se define como:

$$Q = \frac{\text{Potencia reactiva}}{\text{Potencia promedio}} = \frac{X_L}{R} \quad (2.18)$$

donde X_L es la reactancia inductiva y R es la resistencia en resonancia. [10]

2.14. Resonancia en bancos de capacitores

La resonancia ocurre en circuitos RLC cuando las reactancias capacitiva (X_C) e inductiva (X_L) son iguales, resultando en una impedancia puramente resistiva. Existen dos tipos principales:

- **Resonancia en paralelo:** Provoca amplificación de corriente.
- **Resonancia en serie:** Provoca amplificación de voltaje.

La frecuencia resonante f_o se calcula como:

$$f_o = \frac{1}{2\pi\sqrt{LC}} \quad (2.19)$$

Según la norma **IEC 60831-1**, la frecuencia de resonancia en bancos de capacitores se determina por:

$$FR = f \sqrt{\frac{S_{cc}}{Q}} \quad (2.20)$$

donde S_{cc} es la potencia de cortocircuito y Q es la potencia reactiva del banco de capacitores. [11]

2.15. Efectos de los armónicos

Los armónicos, generados por cargas no lineales, tienen efectos perjudiciales en los sistemas eléctricos: [12]

- **Efectos inmediatos:** Disparo de protecciones, vibraciones, ruidos anormales, sobrecarga térmica y fallos en cargas no lineales.
- **Equipos afectados:**
 - Capacitores: Pérdidas dieléctricas, vibraciones y desgaste mecánico.
 - Motores: Calentamiento, vibraciones y reducción de vida útil.
 - Interruptores automáticos: Disparos intempestivos.
 - Cables: Pérdidas dieléctricas, especialmente en el neutro por el armónico $H3$.
 - Ordenadores: Pérdida de datos.
 - Electrónica de potencia: Problemas de sincronización.

2.15.1. Tercera armónica

El armónico de tercer orden ($H3$) es particularmente problemático en sistemas con neutro, ya que puede causar sobrecargas en el conductor neutro. En cargas monofásicas con rectificadores, el $H3$ puede alcanzar hasta el 80 % de la corriente fundamental. [9] Ejemplos de equipos con rectificadores monofásicos incluyen bancos de baterías y sistemas de protección catódica.

2.16. Instalación de capacitores en redes con armónicos

Para mitigar los efectos de los armónicos en bancos de capacitores, se deben tomar precauciones: [13]

- Reducir la amplitud de los armónicos mediante filtros.
- Separar cargas generadoras de armónicos de las no generadoras.
- Realizar mediciones del espectro armónico durante 3 a 7 días.

En el sector industrial, los motores en 480 V o 4.16 kV deben conectarse en configuración delta para eliminar el armónico $H3$. Las mediciones deben realizarse en los puntos de mayor contenido armónico y cerca de los bancos de capacitores.

2.17. Causas y efectos de los armónicos

En la siguiente tabla se citan, a título orientativo, distintos receptores con indicaciones sobre el espectro armónico en intensidad inyectado.

Tabla 2.1: Cargas y sus armónicos generados.

Tipo de carga	Armónico generado	Comentarios
Transformador	Orden par e impar	Componente en CC
Motor asíncrono	Orden impar	Inter y subarmónicos
Lámpara de descarga	$3^\circ + \text{impar}$	Hasta 30 % de I_1
Soldadura arco	3°	
Horno arco CA	Espectro variable inestable	No lineal, asimétrico
Rectificadores con filtro inductivo	$h = KP \pm 1; I_h = I_1/h$	SAI, variadores de velocidad
Rectificadores con filtro capacitivo	$h = KP \pm 1; I_h = I_1/h$	Alimentación equipos electrónicos
Cicloconvertidores	Variables	Variadores de velocidad
Reguladores PWM	Variables	SAI, convertidor CC/CA

2.18. Soluciones para compensación en redes con armónicos

Dependiendo del nivel de distorsión armónica total de voltaje (THDV):

- **THDV <1.5 %:** Usar equipos estándar con tensión nominal igual a la de la red.
- **1.5 % <THDV <6 %:** Emplear equipos con filtros de rechazo sintonizados a 215 Hz para evitar la amplificación de armónicos.
- **THDV >6 %:** Utilizar filtros pasivos y activos para reducir el THDV por debajo del 3.2 %.

La norma **EN 61642** prohíbe añadir inductancias antiarmónicas a capacitores no dimensionados para soportar un 10 % adicional de la tensión de red. [13]

2.19. Estándares de calidad de la energía

Los estándares internacionales proporcionan directrices para medir, evaluar y mitigar problemas de calidad de la energía, asegurando la compatibilidad electromagnética y la confiabilidad de los sistemas eléctricos. [14] A continuación, se describen los principales estándares mencionados:

2.19.1. CEI 61000-4-30

Esta norma define métodos de medición de calidad de la energía, clasificando los instrumentos en Clase A (máxima precisión) y Clase S. Especifica algoritmos de medición, precisión y sincronización temporal para parámetros como voltaje, frecuencia y armónicos. [15]

2.19.2. CEI 61000-4-7

Esta norma establece requisitos para medir armónicos de corriente y voltaje en sistemas eléctricos, así como las emisiones armónicas de equipos. Define el rendimiento de instrumentos de medición para garantizar resultados consistentes. [16]

2.19.3. CEI 61000-4-15

Esta norma especifica técnicas de prueba para medir fluctuaciones de voltaje y parpadeo (*flicker*). Incluye requisitos para instrumentos que evalúan la severidad del parpadeo en función de su impacto en sistemas de iluminación. [17]

2.19.4. EN 50160

Esta norma europea define límites para características de voltaje en sistemas de distribución pública, como variaciones de frecuencia, caídas de voltaje y niveles de armónicos. Se utiliza para garantizar un suministro eléctrico uniforme. [14]

2.19.5. IEEE 1159

Esta norma proporciona prácticas recomendadas para el monitoreo de la calidad de la energía, incluyendo definiciones de disturbios y métodos de medición. Es ampliamente utilizada en América del Norte. [12]

2.19.6. IEEE 519

Esta norma establece límites recomendados para armónicos de voltaje y corriente en sistemas eléctricos. Define niveles aceptables de THD y especifica responsabilidades para usuarios y proveedores de energía. [18]

2.20. Analizador de calidad de energía

Un analizador de calidad de energía es un instrumento utilizado para realizar mediciones de calidad de energía. Este instrumento mide y analiza varios aspectos de un suministro de energía eléctrica durante largos períodos de tiempo. Registra datos sobre voltaje, corriente, frecuencia, armónicos, transitorios y otros parámetros para analizar la calidad de la energía. Esto permite diagnosticar problemas con la calidad de la energía, garantizar el cumplimiento de los estándares y optimizar el rendimiento de los sistemas eléctricos. [19]



Figura 2.2: Analizador de calidad de la energía eléctrica - C.A 8345.

Capítulo 3

Caracterización del Problema: Simulaciones y Mediciones

3.1. Clasificación y simulación de armónicos pares e impares

La realización de simulaciones en el software ATPDraw para modelar la inyección de armónicos pares e impares en una red eléctrica se justifica por la necesidad de caracterizar de manera controlada y diferenciada su impacto en la calidad de la energía, alineado con los objetivos del presente trabajo de desarrollar un método flexible para el análisis de perturbaciones en sistemas eléctricos.

Los armónicos pares (H2, H4, H6, H8), típicamente generados por cargas no lineales asimétricas o fallos en convertidores, como rectificadores de media onda o desbalances en sistemas trifásicos, provocan distorsiones que afectan la simetría de la onda sinusoidal y pueden inducir corrientes de secuencia cero, incrementando pérdidas por calentamiento en neutros y transformadores.

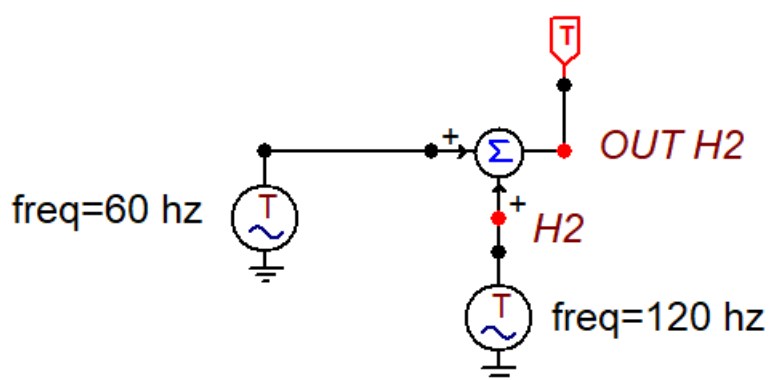


Figura 3.1: Simulación de distorsión por la presencia de segunda armónica.

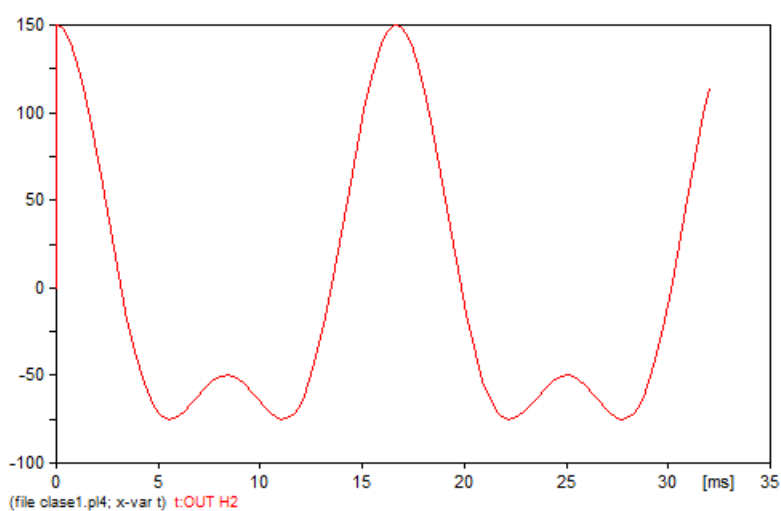


Figura 3.2: Forma de onda distorsionada por la presencia de segunda armónica.

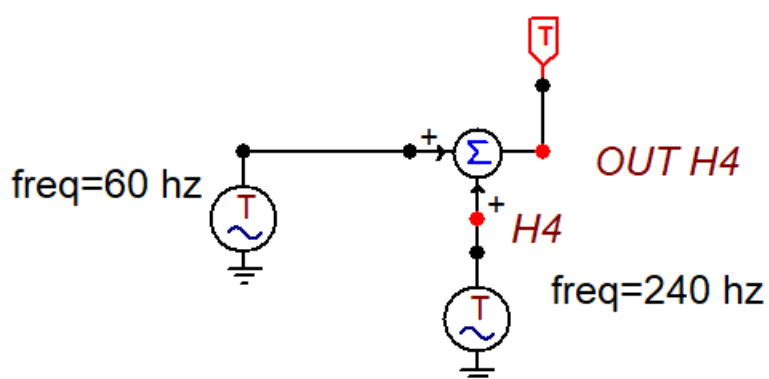


Figura 3.3: Simulación de distorsión por la presencia de cuarta armónica.

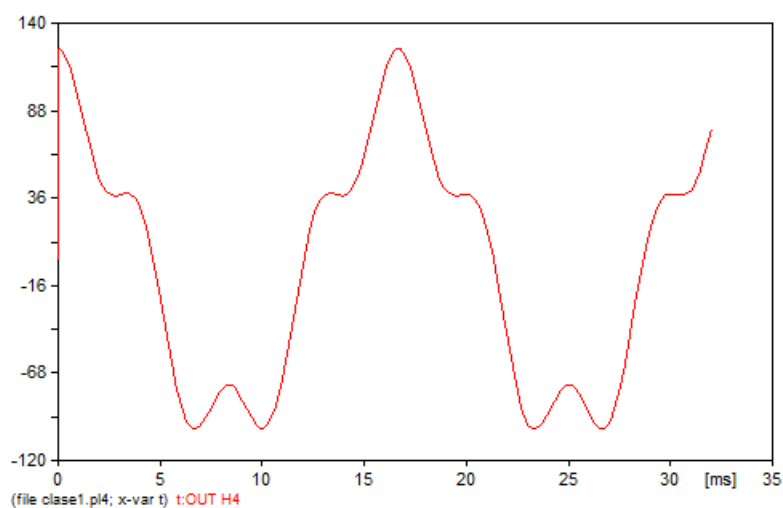


Figura 3.4: Forma de onda distorsionada por la presencia de cuarta armónica.

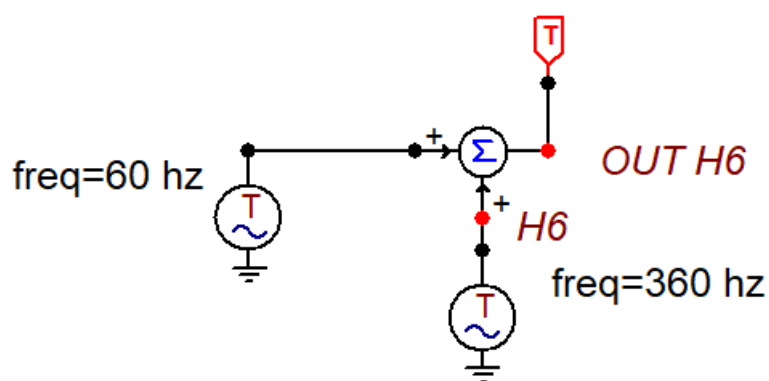


Figura 3.5: Simulación de distorsión por la presencia de sexta armónica.

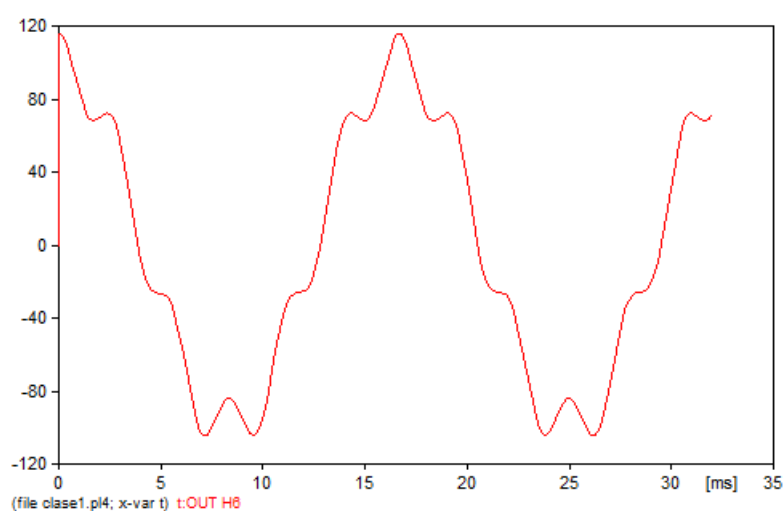


Figura 3.6: Forma de onda distorsionada por la presencia de sexta armónica.

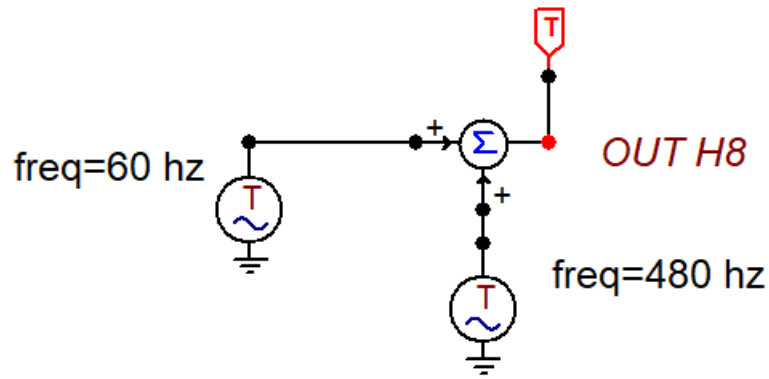


Figura 3.7: Simulación de distorsión por la presencia de octava armónica.

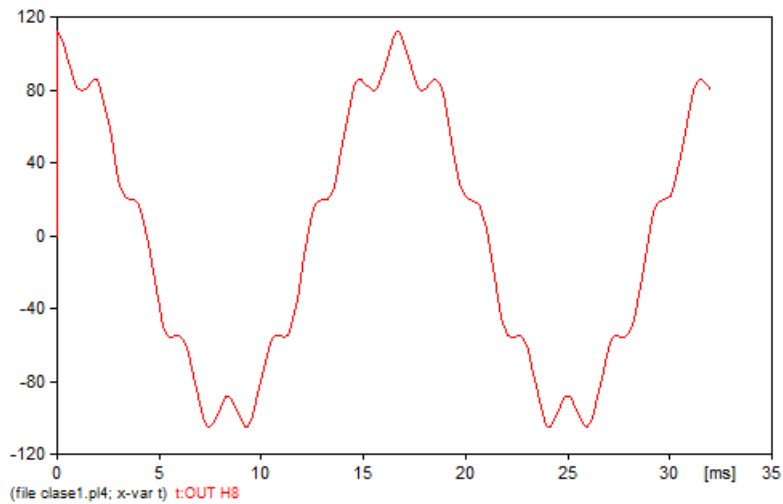


Figura 3.8: Forma de onda distorsionada por la presencia de octava armónica.

Por otro lado, los armónicos impares (H3, H5, H7, H9, H11), más comunes en cargas como rectificadores trifásicos o variadores de frecuencia, son predominantes en escenarios industriales y contribuyen a resonancias y sobrecargas en capacitores, como se evidencia en efectos reales tales como la explosión de bancos de capacitores por resonancia.

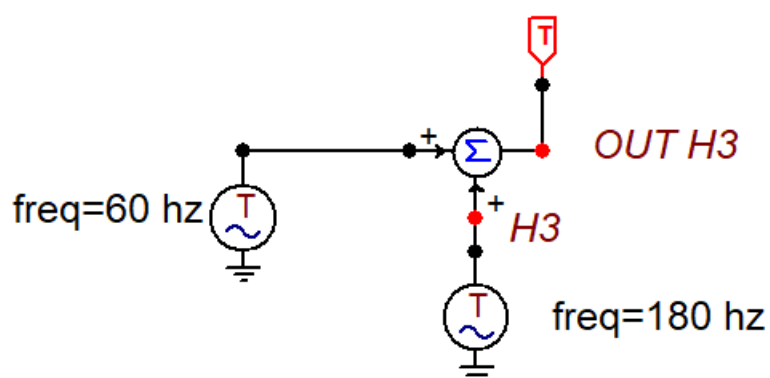


Figura 3.9: Simulación de distorsión por la presencia de tercera armónica.

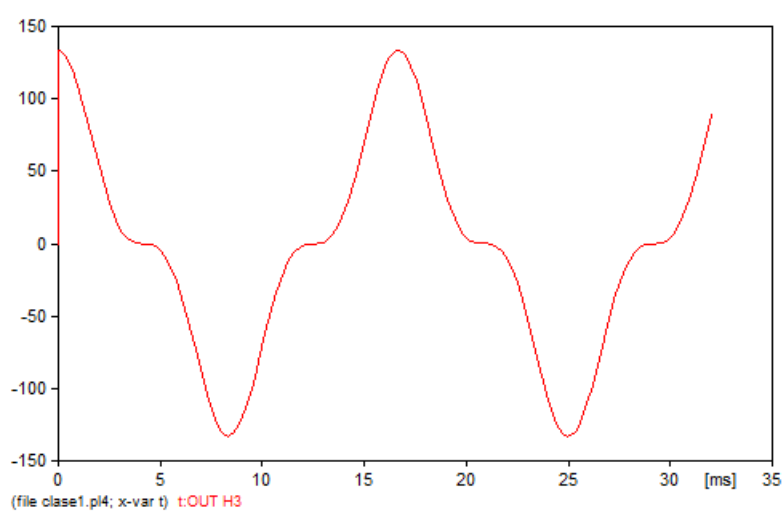


Figura 3.10: Forma de onda distorsionada por la presencia de tercera armónica.

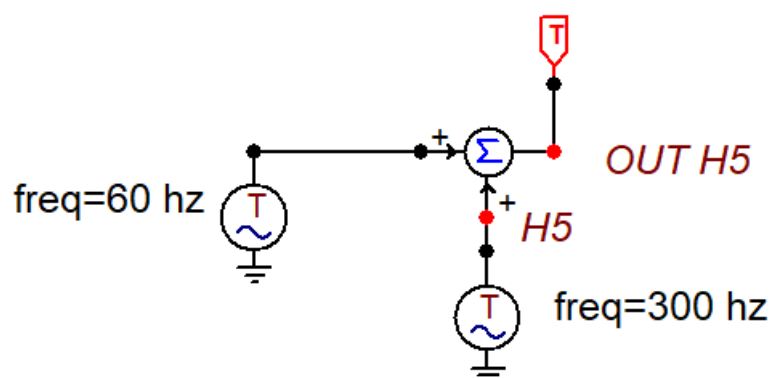


Figura 3.11: Simulación de distorsión por la presencia de quinta armónica.

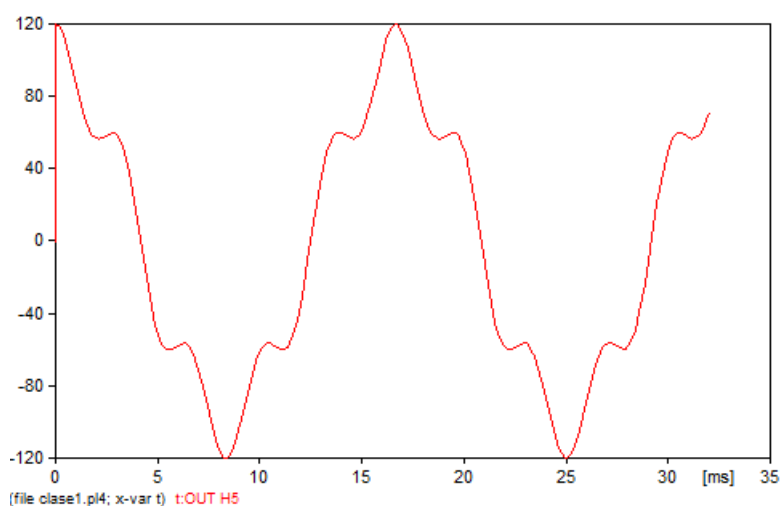


Figura 3.12: Forma de onda distorsionada por la presencia de quinta armónica.

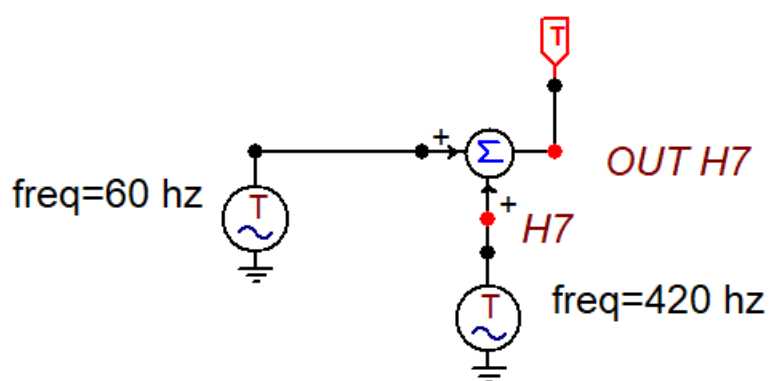


Figura 3.13: Simulación de distorsión por la presencia de séptima armónica.

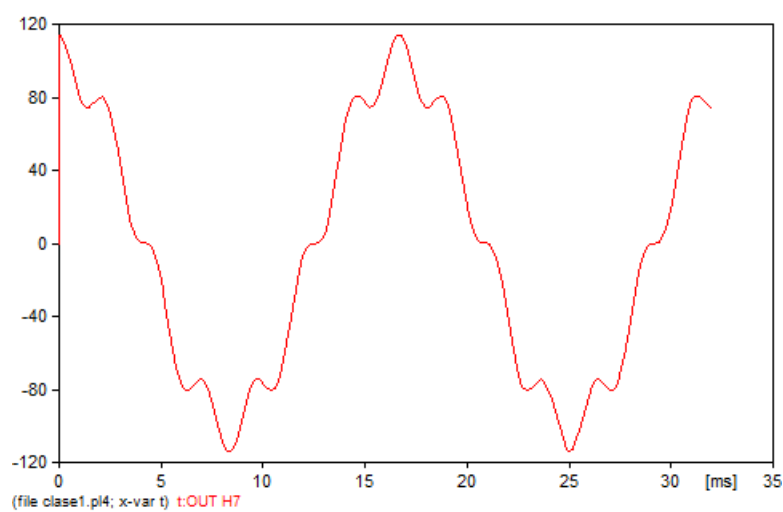


Figura 3.14: Forma de onda distorsionada por la presencia de séptima armónica.

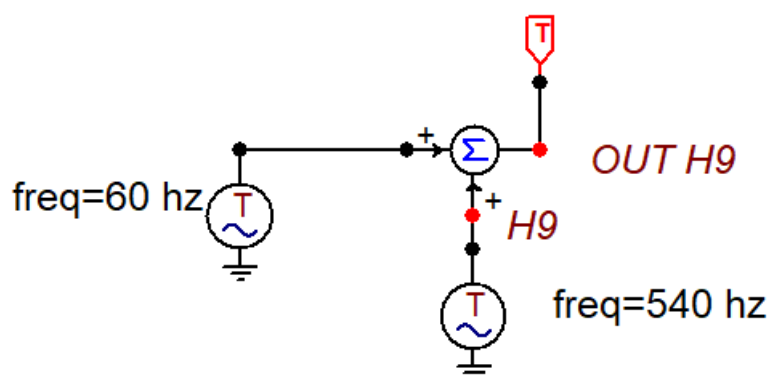


Figura 3.15: Simulación de distorsión por la presencia de novena armónica.

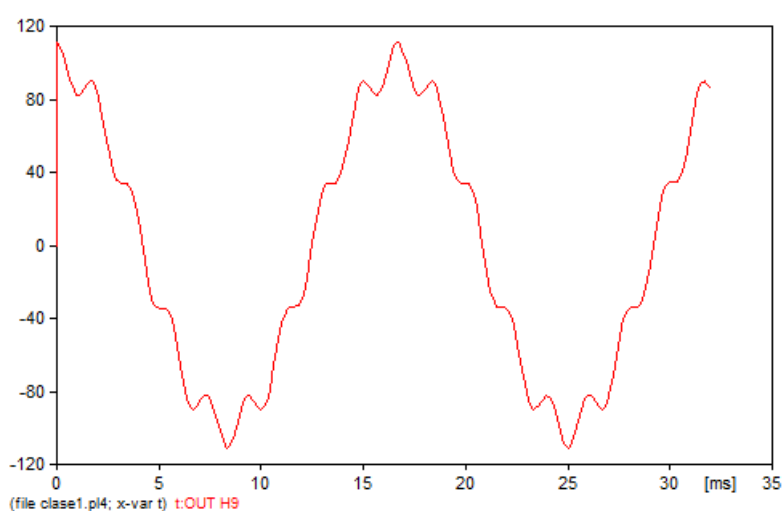


Figura 3.16: Forma de onda distorsionada por la presencia de novena armónica.

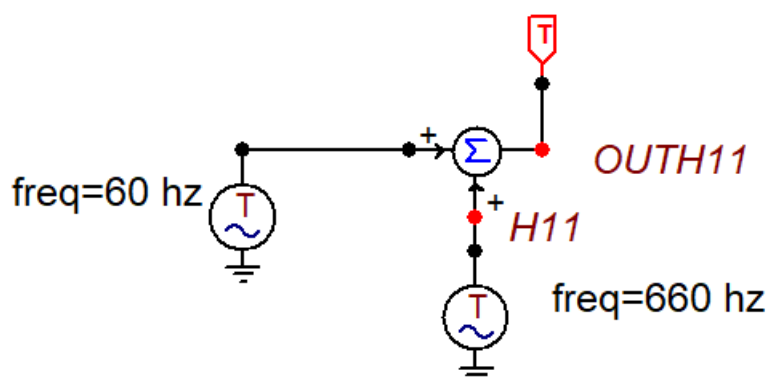


Figura 3.17: Simulación de distorsión por la presencia de undécima armónica.

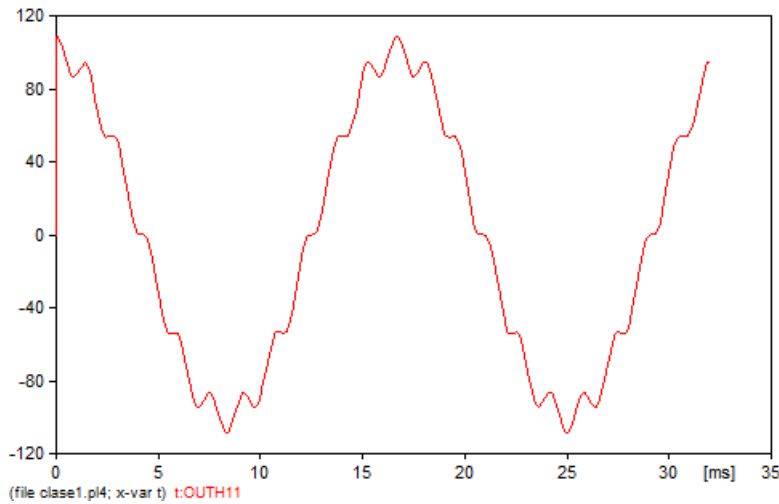


Figura 3.18: Forma de onda distorsionada por la presencia de undécima armónica.

Al simular estos escenarios en ATPDraw, con una onda fundamental de 100 V y 60 Hz como base, se permite una clasificación clara y la visualización de su propagación en circuitos equivalentes, facilitando la comprensión de cómo interactúan con elementos pasivos (resistencias, inductancias y capacitores) y reproduciendo condiciones reales observadas en redes industriales, según la norma IEEE 519-2014. Esta aproximación no solo valida teóricamente los parámetros de medida de armónicos, sino que también sirve como base para el diseño de sistemas de adquisición de datos en capítulos posteriores, minimizando riesgos en mediciones reales con equipos como el Fluke 434.

De esta forma, las simulaciones contribuyen a una metodología integral, adaptable a diversos escenarios, y apoyan la hipótesis de que un sistema combinado de software numérico y adquisición de datos puede mitigar problemas de calidad energética.

3.2. Simulaciones de resonancia en sistemas eléctricos

La resonancia en sistemas eléctricos representa uno de los fenómenos más críticos en el análisis de la calidad de la energía, ya que puede amplificar armónicos y generar sobrevoltajes o sobrecorrientes que comprometen la integridad de los componentes, como bancos de capacitores o transformadores. En el contexto de la presente tesis, las simulaciones de resonancia se realizan para caracterizar el comportamiento de los armónicos en escenarios reales, permitiendo identificar condiciones de riesgo antes de implementar mediciones físi-

cas. Utilizando el software ATPDraw, se modelaron circuitos RLC que replican configuraciones típicas en redes eléctricas industriales, enfocándose en la interacción entre inductancias y capacitancias que conducen a resonancias en serie o en paralelo.

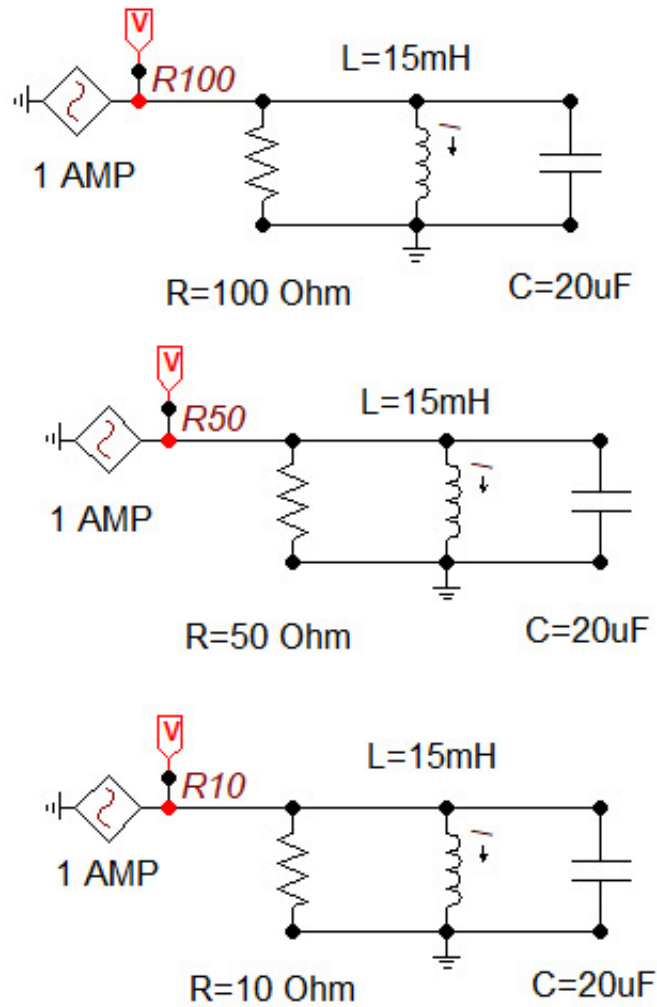


Figura 3.19: Circuitos modelados con diferentes valores de resistencia.

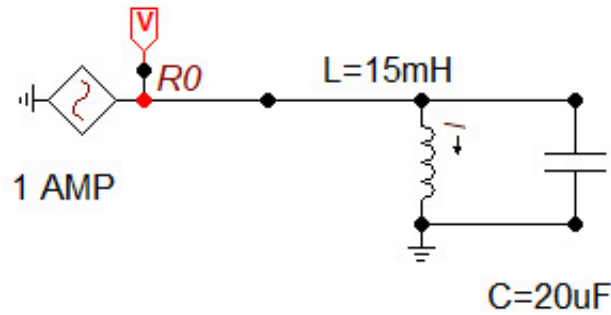


Figura 3.20: Circuito modelado sin resistencia.

El objetivo principal de estas simulaciones es visualizar los efectos de la resonancia en la red eléctrica, como el aumento en la amplitud de las señales o la distorsión de la forma de onda, lo cual se alinea con el planteamiento del problema en el Capítulo 1, donde se destaca la necesidad de métodos flexibles para detectar perturbaciones prolongadas. Por ejemplo, en presencia de armónicos, la resonancia puede provocar calentamiento excesivo en conductores o explosiones en capacitores, como se ilustra en las Figuras 1.1 y 1.2. Estas simulaciones sirven como base para validar el sistema de adquisición de datos propuesto en capítulos posteriores, asegurando que el dispositivo pueda detectar tales anomalías.

Se crearon una serie de circuitos en ATPDraw para simular diferentes escenarios de resonancia, basados en parámetros como la resistencia (R), inductancia (L) y capacitancia (C). El procedimiento seguido fue el siguiente:

- Modelado de circuitos básicos RLC en serie y en paralelo, ajustando valores para frecuencias de resonancia cercanas a múltiplos armónicos comunes (e.g., 5.^a, 7.^a armónica).
- Introducción de fuentes de voltaje senoidales con componentes armónicos para observar amplificaciones.
- Análisis de resultados mediante gráficas de señales temporales y espectros de frecuencia.

Ejemplos de circuitos simulados incluyen:

- Un circuito con $R = 100 \Omega$, $L = 15 \text{ mH}$, $C = 20 \mu\text{F}$, que genera resonancia en serie alrededor de 290.5 Hz.

- Variaciones con $R = 50\ \Omega$ o $R = 10\ \Omega$ para estudiar el impacto del factor de calidad Q en la amplitud de resonancia.

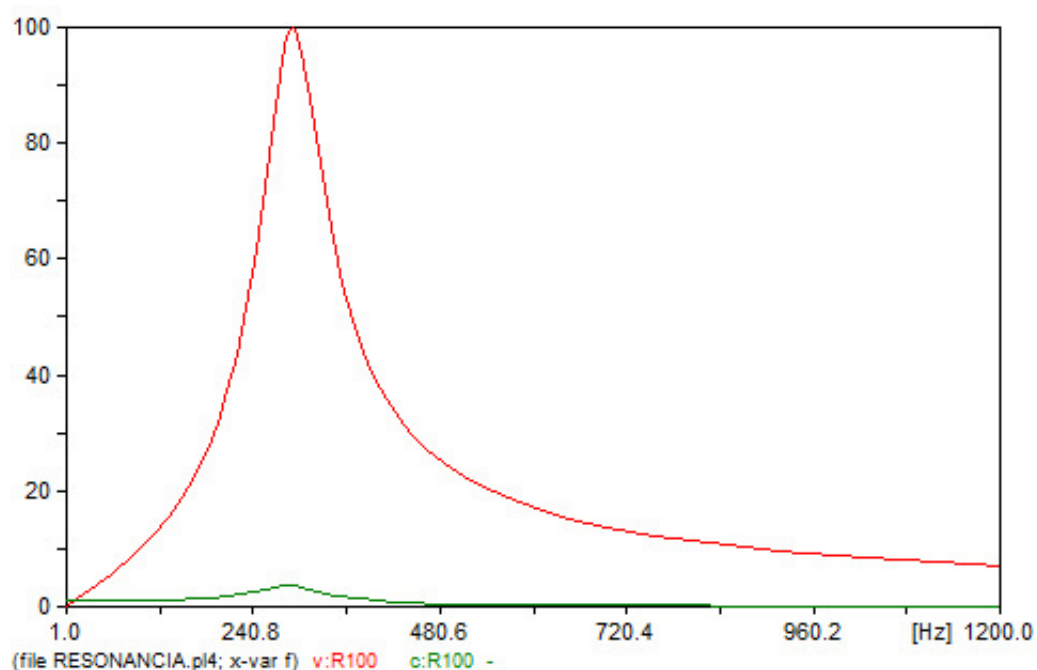


Figura 3.21: Gráfica de voltaje y corriente con resistencia de $100\ \Omega$.

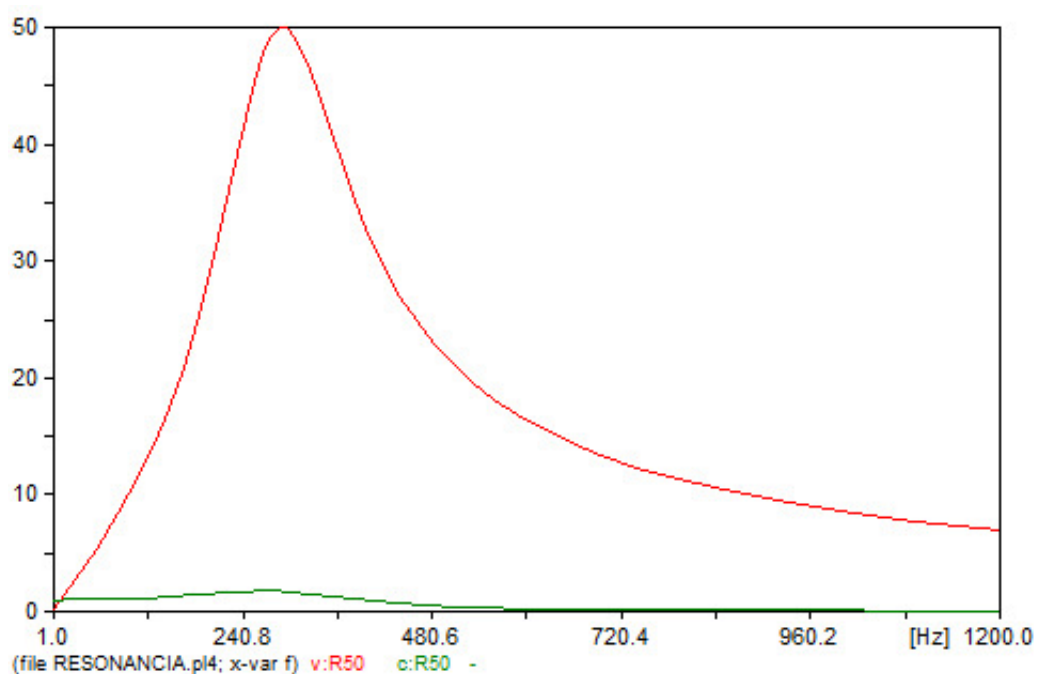


Figura 3.22: Gráfica de voltaje y corriente con resistencia de $50\ \Omega$.

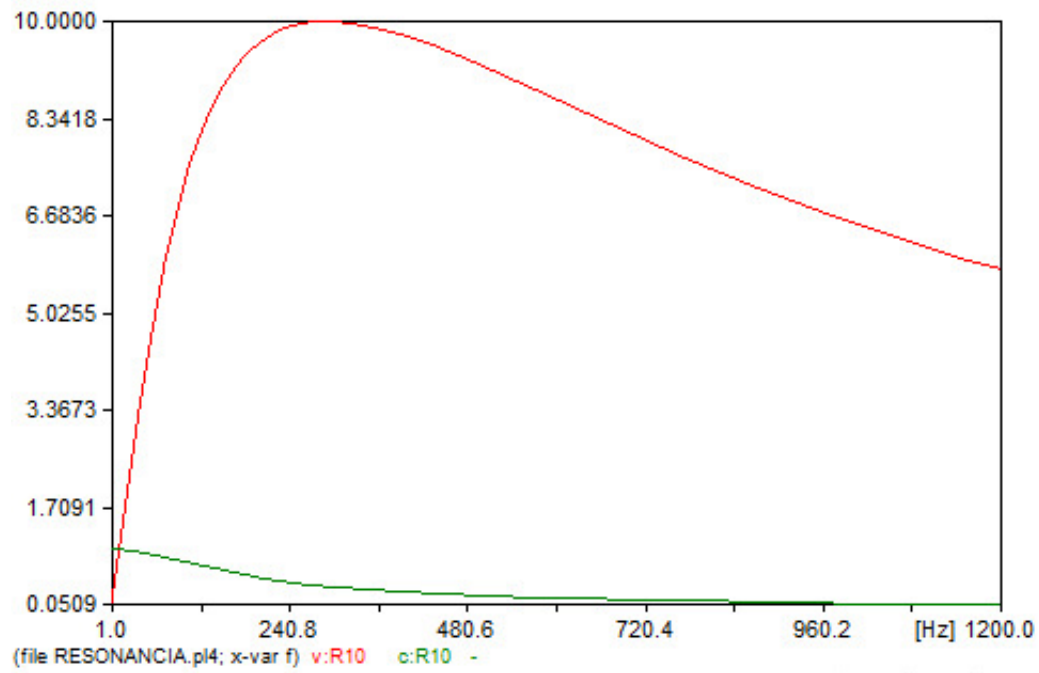


Figura 3.23: Gráfica de voltaje y corriente con resistencia de 10Ω .

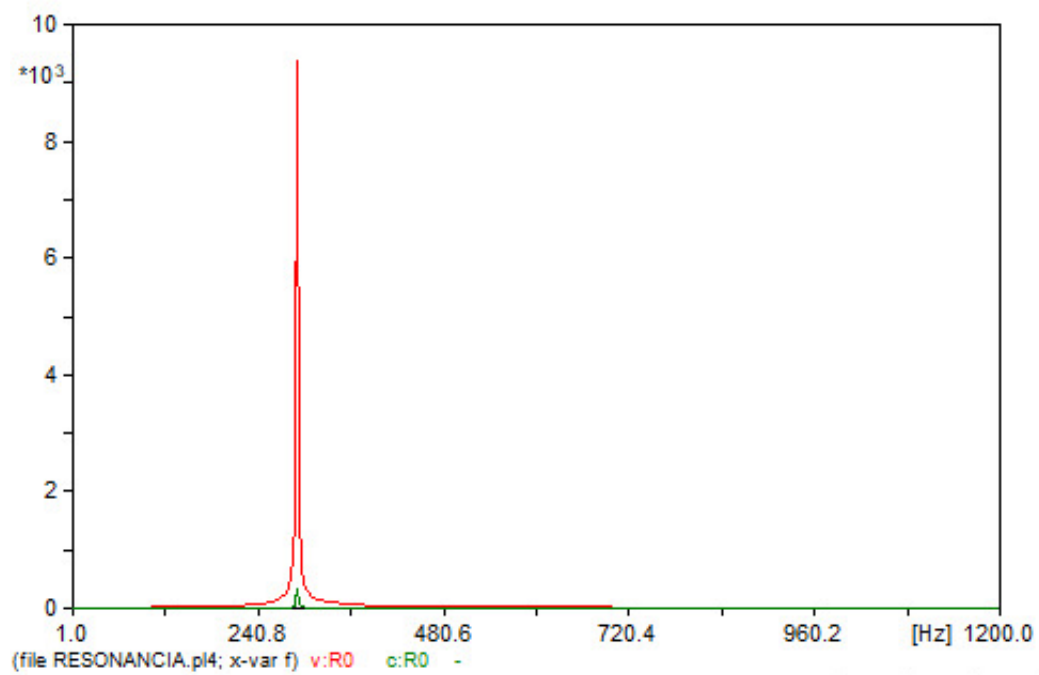


Figura 3.24: Gráfica de voltaje y corriente con resistencia de 0Ω .

3.3. Mediciones de armónicos en redes reales utilizando analizador Fluke

Las mediciones prácticas de armónicos en redes eléctricas contaminadas constituyen un paso esencial en el análisis de la calidad de la energía, ya que permiten validar los fenómenos teóricos y simulados en escenarios reales. En la presente tesis, se empleó un analizador de redes Fluke (modelo 434, según referencias del laboratorio) para evaluar la presencia de armónicos en una instalación eléctrica típica, específicamente en el centro de carga que alimenta los salones del edificio EE del campus 2.

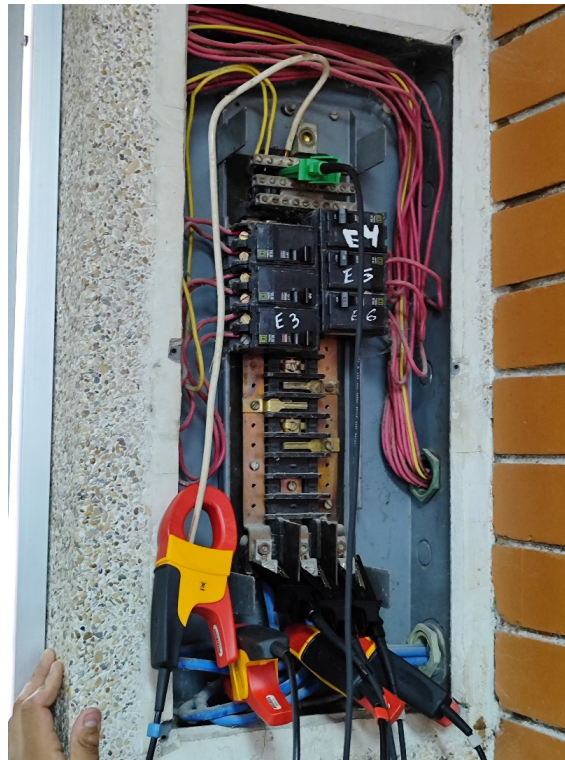


Figura 3.25: Instalación de las sondas de medición en el centro de carga.

Este enfoque complementa las simulaciones de resonancia realizadas en ATPDraw (Sección 2.2), proporcionando datos empíricos que ilustran cómo los armónicos afectan la forma de onda y la relación armónica en entornos industriales o educativos. El objetivo principal fue identificar armónicos comunes, como los de orden 3, 5 y 7, que pueden causar distorsiones prolongadas, calentamiento en conductores y daños en equipos, alineándose con el planteamiento del problema en el Capítulo 1, donde se enfatiza la necesidad de métodos flexibles para detectar perturbaciones que impactan la eficiencia y seguridad de los sistemas

eléctricos.

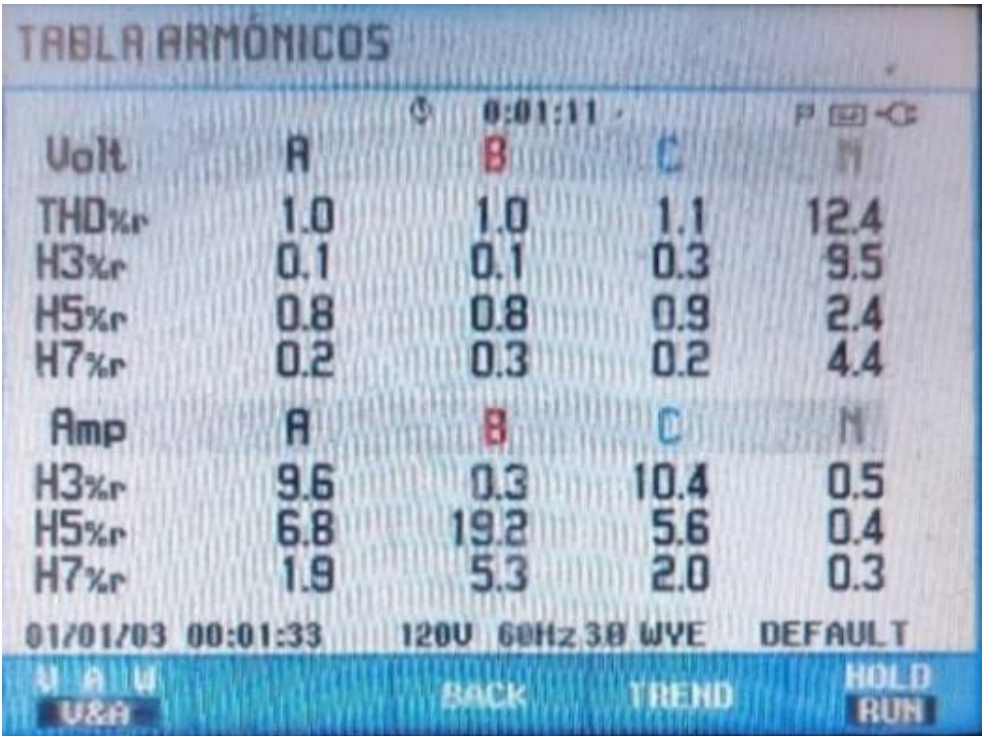


Figura 3.26: Porcentaje de armónicos en corriente y voltaje.



Figura 3.27: Gráfico de armónicos en corriente.

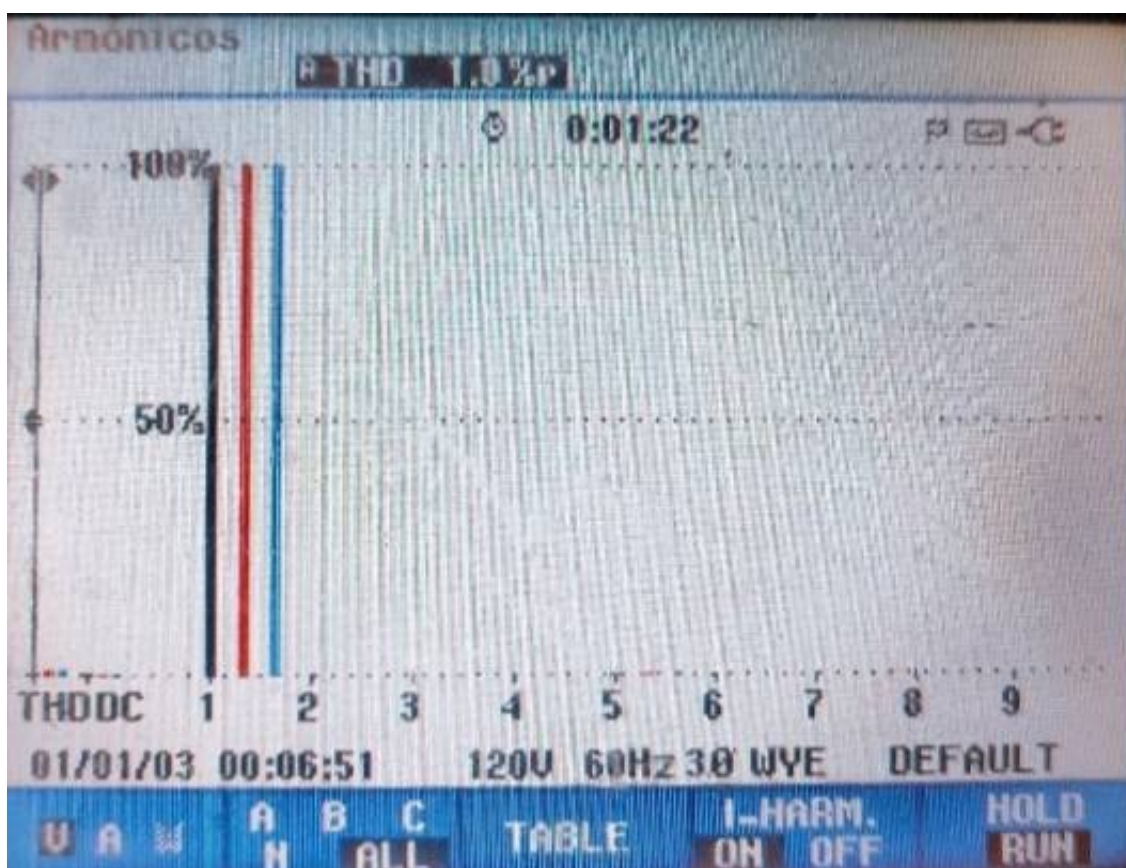


Figura 3.28: Gráfico de armónicos en voltaje.

Estas mediciones no solo cuantifican la THD y los componentes individuales, sino que también sirven como base para el diseño del sistema de adquisición de datos propuesto en capítulos posteriores, demostrando la relevancia práctica de herramientas como la transformada de Fourier discreta en el procesamiento de señales reales.

Capítulo 4

Selección de Componentes y Tecnologías para el Sistema de Análisis de Calidad de Energía

4.1. Sondas de corriente y voltaje

Los microcontroladores están diseñados principalmente para procesar señales de voltaje a través de un (Convertidor analógico - digital (ADC)). Estos ADC convierten variaciones de voltaje en valores digitales que el procesador puede manipular. Sin embargo, la corriente eléctrica no es una señal que un microcontrolador pueda leer directamente de manera segura y precisa. [20]

Los pines de entrada de un microcontrolador no están hechos para manejar corrientes altas o variables directamente. Intentar medir corriente pasando el cable directamente por un pin podría causar daños al hardware, generar sobrecalentamiento, quemar componentes o destruir el microcontrolador. [21]

No hay un mecanismo nativo para convertir corriente en una señal legible; los ADC solo responden a voltajes típicamente en un rango de 0-3.3 V o 0-5 V. Las sondas de corriente actúan como transductores que convierten la corriente (I) en un voltaje proporcional (V). [22] Su principio radica en la conversión de potenciales o flujos magnéticos: las sondas de voltaje detectan diferencias de potencial, mientras que las de corriente miden inducciones o resistencias. Se priorizan opciones no invasivas para evitar interrupciones en instalaciones como el centro de carga del edificio EE (Sección 2.3), diferenciándolas de las invasivas que

requieren inserción física en el circuito. [23]



Figura 4.1: Transformador de corriente SCT013.

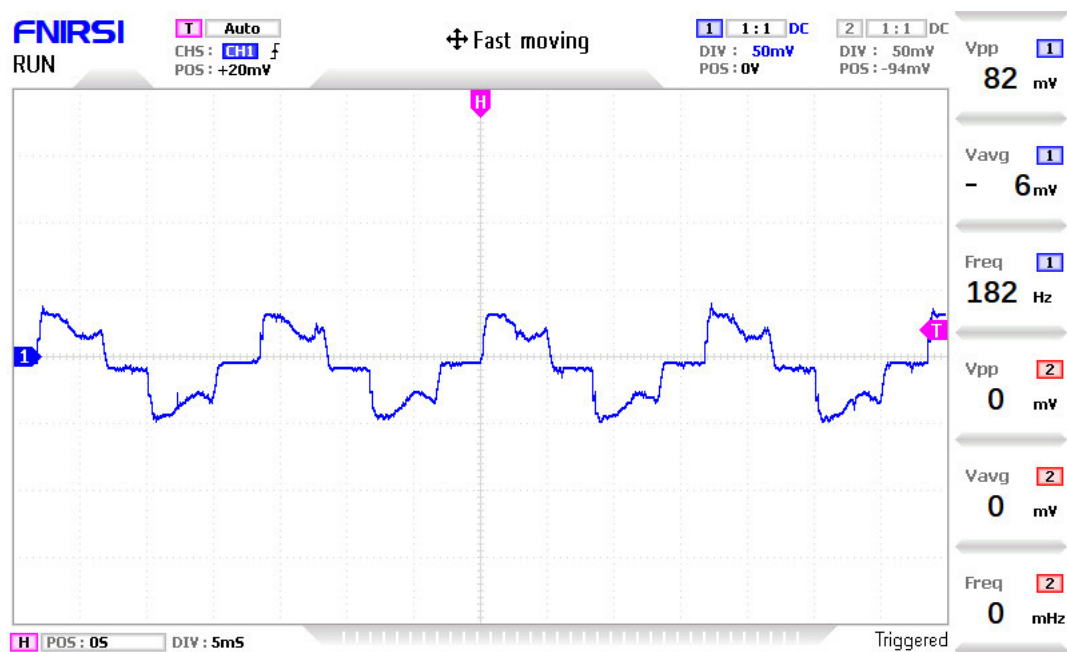


Figura 4.2: Señal obtenida por el sensor de corriente SCT013.



Figura 4.3: Bobina Rogowski ETCR50FA.



Figura 4.4: Señal obtenida por la bobina Rogowski ETCR50FA.

Los sensores de efecto Hall proporcionan medición bidireccional (CA/Corriente Directa (CD)) con aislamiento galvánico [24], el Transformador de corriente (CT) ofrece alta precisión en CA aunque pueden saturarse [25], y las bobinas Rogowski destacan por su flexibilidad y amplio ancho de banda para armónicos de alta frecuencia [26].

Los sensores Hall superan a los CT en CD, pero son vulnerables a campos magnéticos externos (errores hasta un 5 %) [28]; los CT resultan económicos, aunque se saturan [25], mientras que las bobinas Rogowski soportan corrientes de hasta 10 kA sin saturación [29]. No obstante, estas últimas requieren un circuito integrador para acondicionar la señal [30] y su disponibilidad en el mercado es limitada. Por ello, se optó por utilizar CT, dada su alta disponibilidad, costo accesible y prestaciones suficientes para la aplicación.

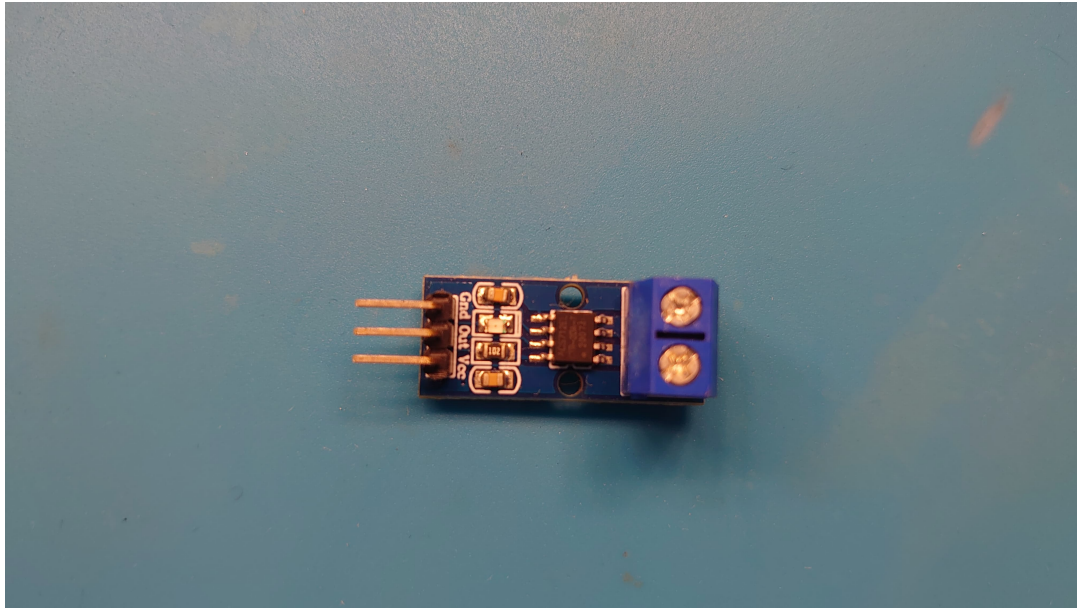


Figura 4.5: Sensor de corriente de efecto Hall ACS712.

Cuando la señal a medir es directamente un voltaje, el proceso es mucho más directo, no es necesaria una conversión de corriente a voltaje, pero sí es común ajustar el nivel de voltaje mediante amplificación o atenuación para que encaje en el rango de entrada del ADC. [31]

Si el voltaje de la señal está dentro de este rango permisible del microcontrolador, puede ser utilizada directamente. Sin embargo, las señales del mundo real, como las de sensores, baterías o redes eléctricas, pueden ser más altas (e.g., 12 V en un motor) o más bajas (e.g., 10 mV de un termistor), lo que requiere ajuste. [32]

The circuit diagram shows two LM358 op-amps, U1A and U1B, configured as precision rectifiers. U1A is a voltage follower with its non-inverting input (+) connected to the input signal through a 3.3M resistor (R1) and a 3.3M resistor (R2) to ground. Its output (pin 1) is connected to its inverting input (-) through a 50K resistor (RV1) and to a 100K resistor (R4) to the output of U1B. U1B is configured as a precision half-wave rectifier. Its non-inverting input (+) is connected to the output of U1A through a 680K resistor (R5) and to ground through a 470K resistor (R7). Its inverting input (-) is connected to ground through a 470K resistor (R6). The output of U1B (pin 7) is connected to the output of U1A through a 22K resistor (R3) and to ground through a 100nF capacitor (C1). The input signal is connected to the non-inverting input of U1A through a 3.3M resistor (R1) and to ground through a 3.3M resistor (R2).

Para las sondas de corriente se utiliza un circuito similar, con la diferencia de que se tiene una resistencia más baja en la entrada del amplificador operacional, ya que la sonda de corriente puede entregar como máximo 1V sin saturarse. [34]

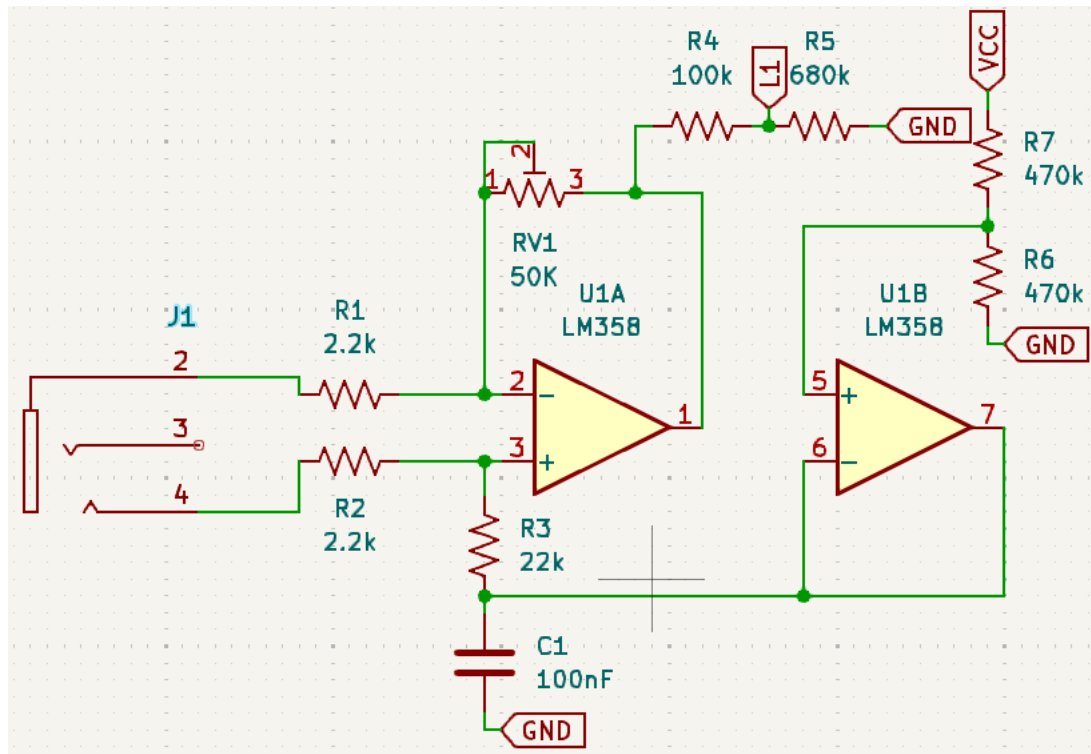


Figura 4.7: Diagrama para la adecuación de señales de corriente (una sola fase).

A continuación, se explica el funcionamiento del circuito por etapas: En la primera etapa (U1A), se cuenta con un Amplificador Operacional (op-amp) en modo amplificador inversor, la ganancia del circuito se ajusta mediante el potenciómetro RV1. En la segunda etapa (U1B), se tiene un op-amp en configuración de seguidor de voltaje, este en su entrada tiene un divisor de voltaje resistivo; esta etapa proporciona un nivel de voltaje CD que permite realizar el desplazamiento de nivel inyectando esa señal en la entrada no inversora del op-amp de la primera etapa. [35]

No se presenta inconveniente alguno el uso de la señal invertida. El análisis espectral armónico, típicamente realizado mediante la Transformada Rápida de Fourier (FFT), resulta invariante ante una inversión de fase de 180° [36]. A continuación, se explica el razonamiento, comparando ambos casos.

El espectro armónico descompone la señal en componentes sinusoidales (fundamental y armónicas), cuantificando magnitudes y fases por frecuencia (e.g., 60 Hz, 120 Hz) [37]. Para métricas estándar como la distorsión armónica total THD, potencia de armónicas o contenido espectral, solo las magnitudes son relevantes; las fases se consideran secundarias salvo

en análisis específicos (e.g., sincronización de fase) [38].

4.3. Sistema de adquisición de datos

El DAQ digitaliza señales analógicas para análisis armónico, donde el ADC es pivotal. La frecuencia de muestreo debe exceder el doble de la máxima para armónicos hasta 15.^a orden (900 Hz en 60 Hz). Comparando, ATmega328P (10-bit, 15 kSPS) es económico, ESP32 (12-bit, 200 kSPS) ofrece conectividad y bajo ruido. [39]

El ESP32 supera en frecuencia de muestreo, ruido mitigado con oversampling, con 5 % menos discrepancia. [40]

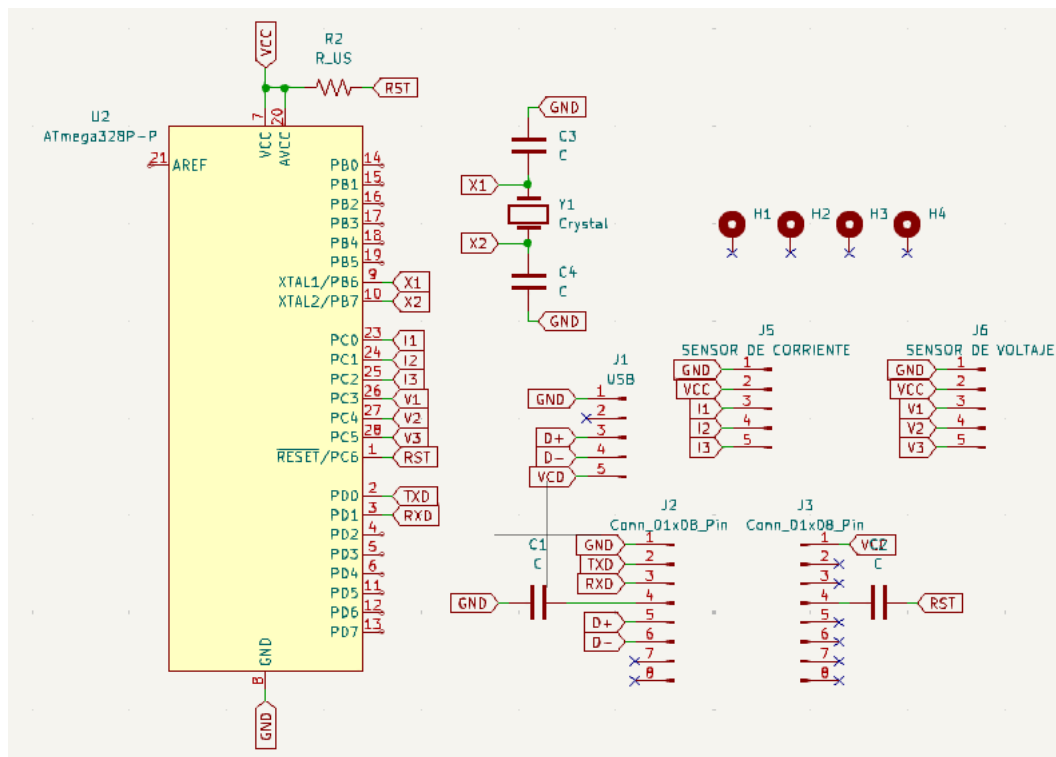


Figura 4.8: Diagrama para el sistema de adquisición de datos primera versión.

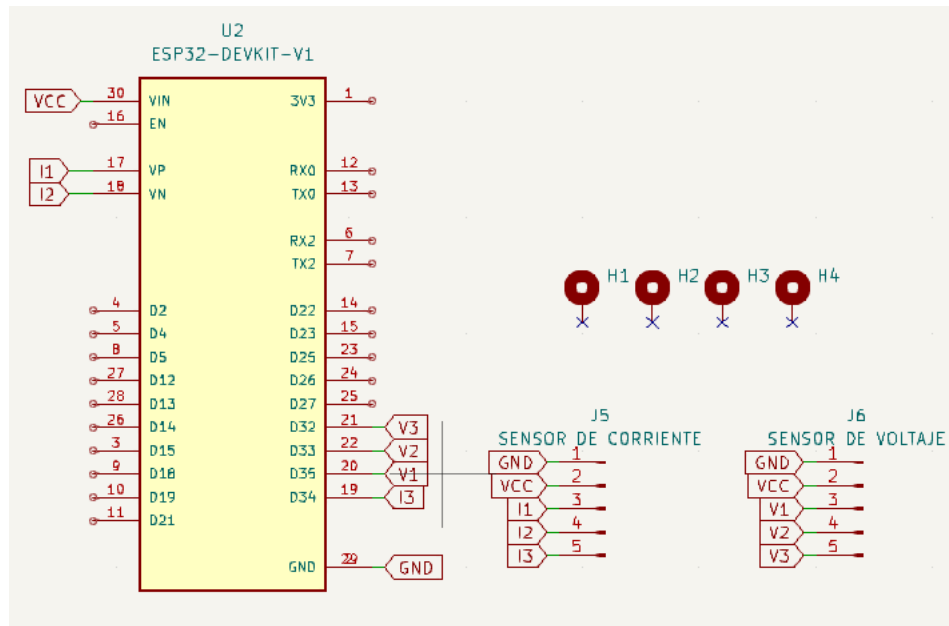


Figura 4.9: Diagrama para el sistema de adquisición de datos segunda versión.

4.4. Selección de software

MATLAB, como software propietario, destaca por su costo elevado —que va desde unos 49 dólares para estudiantes hasta 2150 dólares por licencia estándar, con suscripciones obligatorias para actualizaciones—, lo que lo hace menos accesible para proyectos académicos individuales. Su sintaxis es la base para las alternativas, con una compatibilidad perfecta con sus toolboxes oficiales, y ofrece un soporte excelente para procesamiento de señales y análisis de datos mediante módulos dedicados como el Signal Processing Toolbox (para FFT y filtrado) y Statistics (para análisis de energía), junto con visualización nativa en 2D y 3D. En integración con DAQ, brilla con su Data Acquisition Toolbox, que soporta hardware común como NI, aunque es propietaria y costosa. En rendimiento, lidera en simulaciones complejas y datasets grandes, siendo un 20-30 % más rápido que las alternativas en benchmarks. Funciona bien en Linux mediante instaladores propietarios, y su interfaz de usuario es robusta con App Designer, extensible en C++ y Java. En general, es ideal para entornos industriales, pero su precio lo penaliza para investigación individual. [41]

Por otro lado, GNU Octave es una opción gratuita y de código abierto bajo licencia GPL, sin costos ni restricciones. Su sintaxis es casi idéntica a la de MATLAB —con una compatibilidad directa del 95 % en scripts portables—, lo que facilita la migración. Para procesamiento

de señales y análisis de datos, ofrece un soporte bueno a través de paquetes del Octave-Forge como signal y statistics, suficiente para análisis de calidad de energía, aunque menos integrado que en MATLAB. La integración con DAQ es limitada, dependiendo de wrappers como DAQtoolbox, que pueden ser menos estables en tiempo real. En rendimiento, es comparable a MATLAB en tareas simples, pero más lento en matrices grandes. [42]

Finalmente, Scilab equilibra bien estos aspectos como software gratuito y de código abierto bajo Licencia Publica General de GNU (GPL), perfecto para proyectos académicos sin presupuesto. Su sintaxis es similar a la de MATLAB, con diferencias menores que requieren ajustes en solo el 10-20 % de los casos, lo que lo hace viable para migraciones. En procesamiento de señales y análisis de datos, es muy bueno gracias a módulos integrados para señales (como signal processing para FFT y wavelet) y estadísticas, superando a Octave en muestreo de datos. Para DAQ, destaca con toolboxes gratuitas como MicroDAQ para adquisición en tiempo real y VISA para control de instrumentos, más integradas que en Octave y sin los costos de MATLAB. Su rendimiento en cómputo numérico es similar al de Octave, eficiente para prototipos y análisis de energía con datasets medianos, aunque no tan rápido como MATLAB en simulaciones intensivas. El soporte en Linux es nativo y altamente estable, con instalación fácil vía repositorios, y su Entorno de Desarrollo Integrado (IDE) completa incluye editor, depurador y plotting integrado, extensible en C, Java y Python. [43]

Se optó por utilizar Scilab por ser un software libre de código abierto y tener herramientas como Xcos y funciones para la creación de GUI; para complementar, se utilizó Python. Como sistema operativo, se utilizó una distribución de Linux Xubuntu para mantener la sinergia del código libre. [44]

Capítulo 5

Elaboración de la Solución para el Sistema de Análisis de Calidad de Energía

5.1. Prototipos en tablilla de experimentos: desarrollo inicial para acondicionamiento de señales y sistema DAQ con ATmega328P y ESP32

Los prototipos en protoboard representan la fase inicial de validación práctica, permitiendo probar circuitos de acondicionamiento de señales (voltaje y corriente) y sistemas de adquisición de datos (DAQ) en configuraciones monofásicas y trifásicas, identificando problemas como ruido debido a contactos inestables. Se implementaron versiones con ATmega328P y ESP32, priorizando la captura de formas de onda armónicas sin soldadura permanente. Esto valida las selecciones del Capítulo 4, revelando limitaciones como la insuficiencia de memoria en ATmega para 5000 muestras, lo que motivó la migración a ESP32.

- Acondicionamiento: circuitos basados en op-amp para amplificar y filtrar señales, probados monofásica y trifásicamente con generadores de funciones.
- DAQ: implementación con ATmega328P y CH340C para comunicación serial, usando 6 ADC; luego, ESP32 DevKit V1 para mayor memoria y sampling simultáneo.
- Pruebas: medición de ruido y estabilidad, atribuyendo inestabilidades a contactos en protoboard.

El ATmega328P capturó señales básicas pero falló en buffers grandes, mientras que el ESP32 manejó datos estables con menor ruido, confirmando su superioridad para trabajar con armónicos. El ruido considerable en protoboard se atribuyó a conexiones sueltas, validando la necesidad de una Placa de Circuito Impreso (PCB).

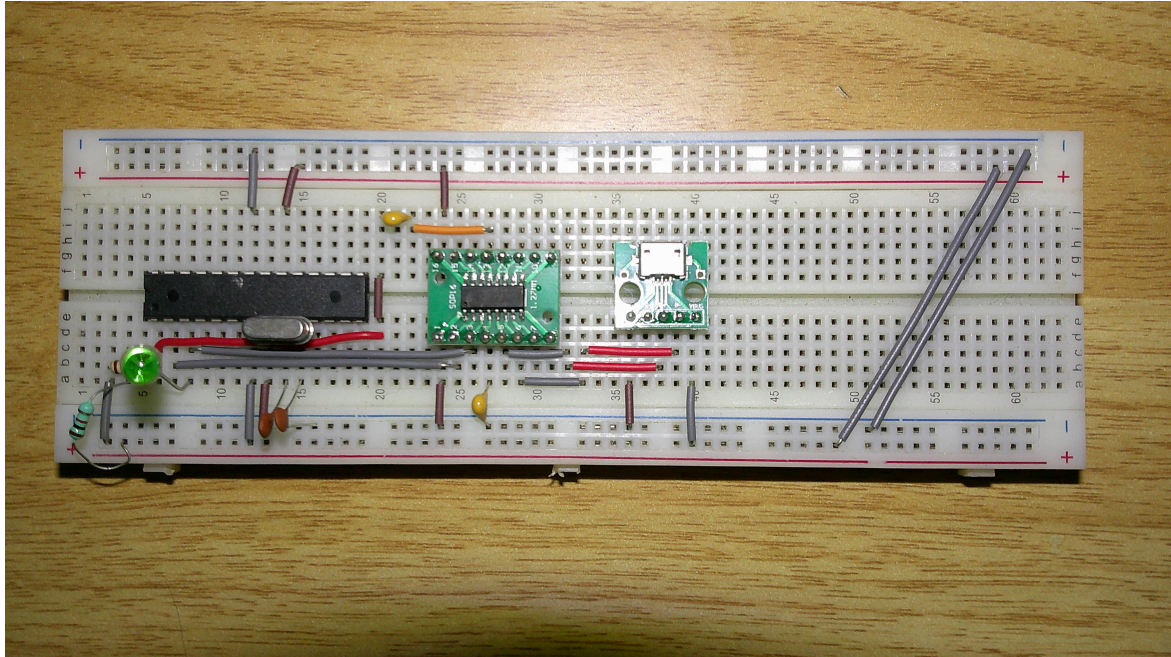


Figura 5.1: Circuito experimental del sistema de adquisición de datos en su primera versión.

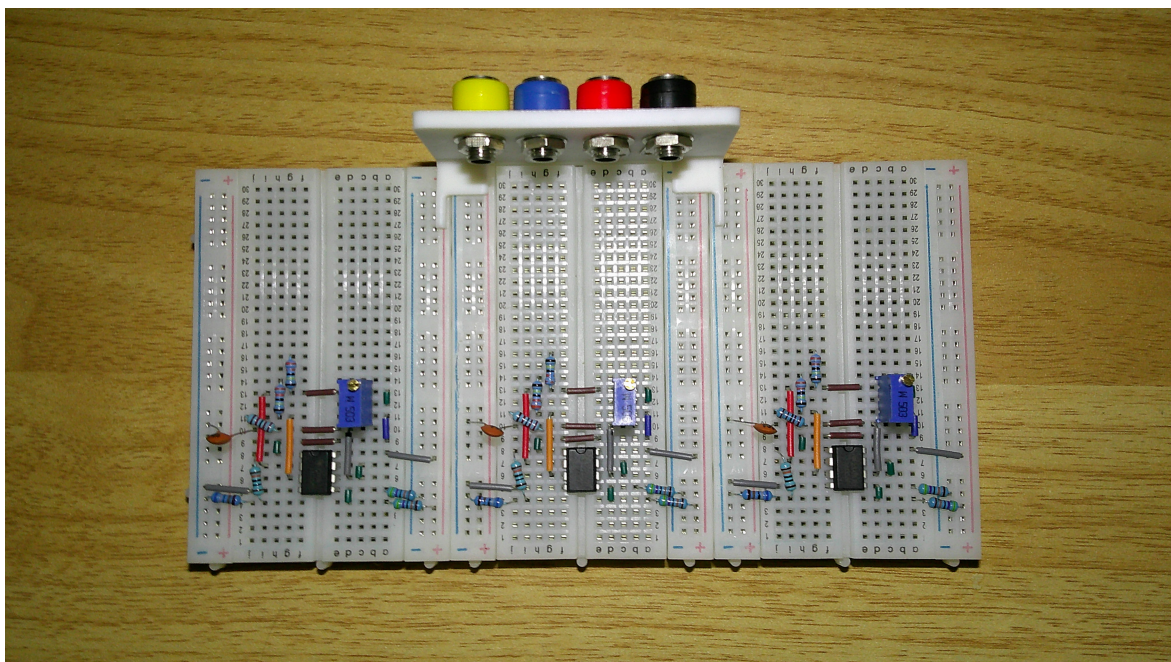


Figura 5.2: Circuito experimental del sistema de adecuación de señales de voltaje.

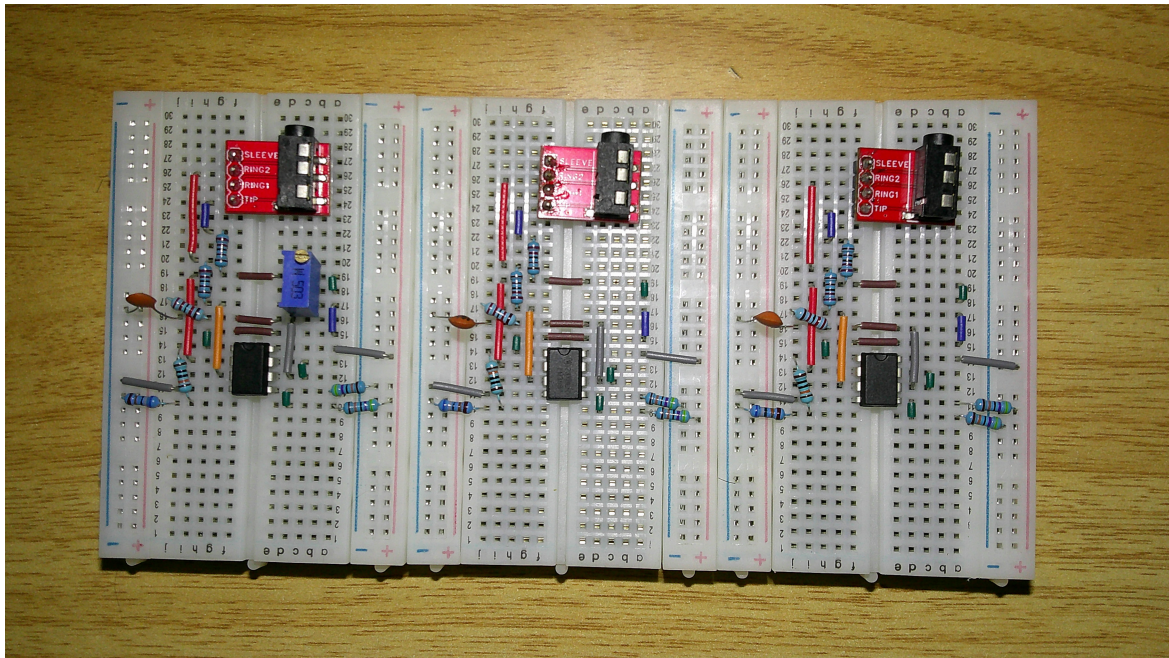


Figura 5.3: Circuito experimental del sistema de adecuación de señales de corriente.

5.2. Diseño de PCB en KiCad: para ATmega328P, ESP32 y acondicionamiento de voltajes y corrientes

El diseño de PCB en KiCad formaliza los prototipos, creando esquemáticos y layouts para bloques de acondicionamiento y DAQ, incluyendo tablas de componentes. Se diseñaron versiones para ATmega328P con CH340C y ESP32 DevKit V1 con conectores Molex para reducir ruido. Esto optimiza trazas para minimizar interferencias en señales armónicas, basado en las selecciones del Capítulo 4.

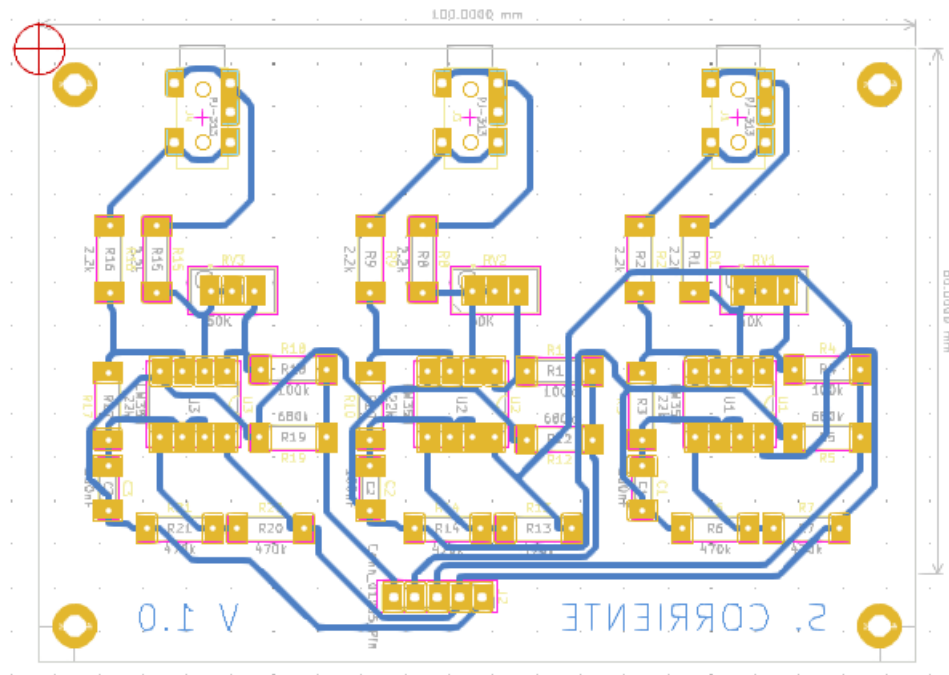


Figura 5.4: Enrutado de pistas del sistema de adecuación de señales de corriente.

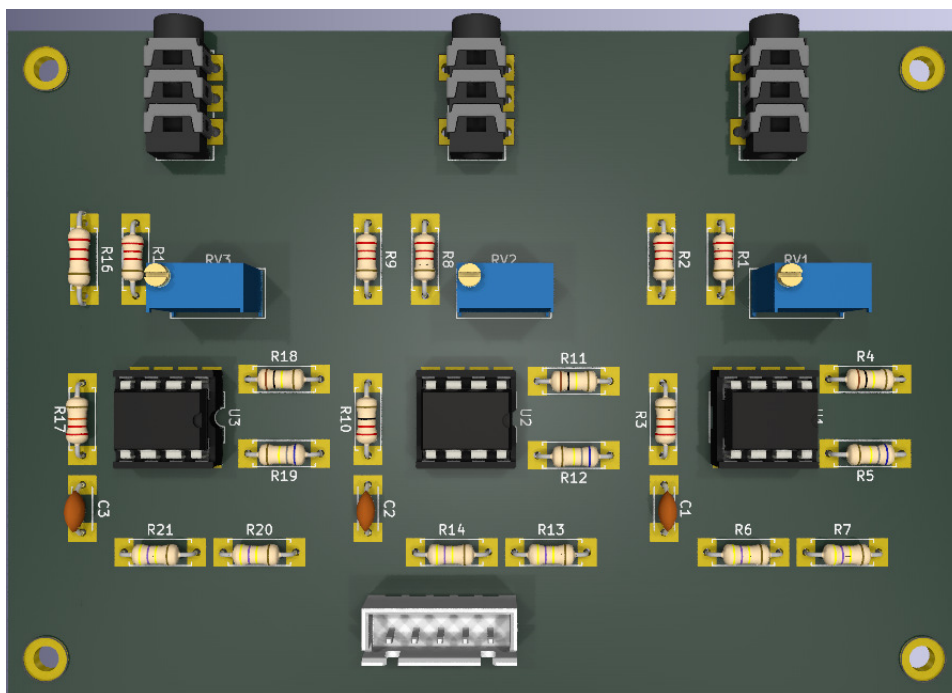


Figura 5.5: Vista 3D del sistema de adecuación de señales de corriente.

Tabla 5.1: Lista de componentes del circuito de adecuación de corrientes.

Referencia	Descripción	Cantidad
C1, C2, C3	100nF	3
R1, R2, R8, R9, R15, R16	2.2kΩ	6
R6, R7, R13, R14, R20, R21	470kΩ	6
R3, R10, R17	22kΩ	3
R4, R11, R18	106kΩ	3
R5, R12, R19	680kΩ	3
U1, U2, U3	LM358	3
RV1, RV2, RV3	50 kΩ	3
J1, J3, J4	PJ-313	3
J2	01 x 05 Pin	1

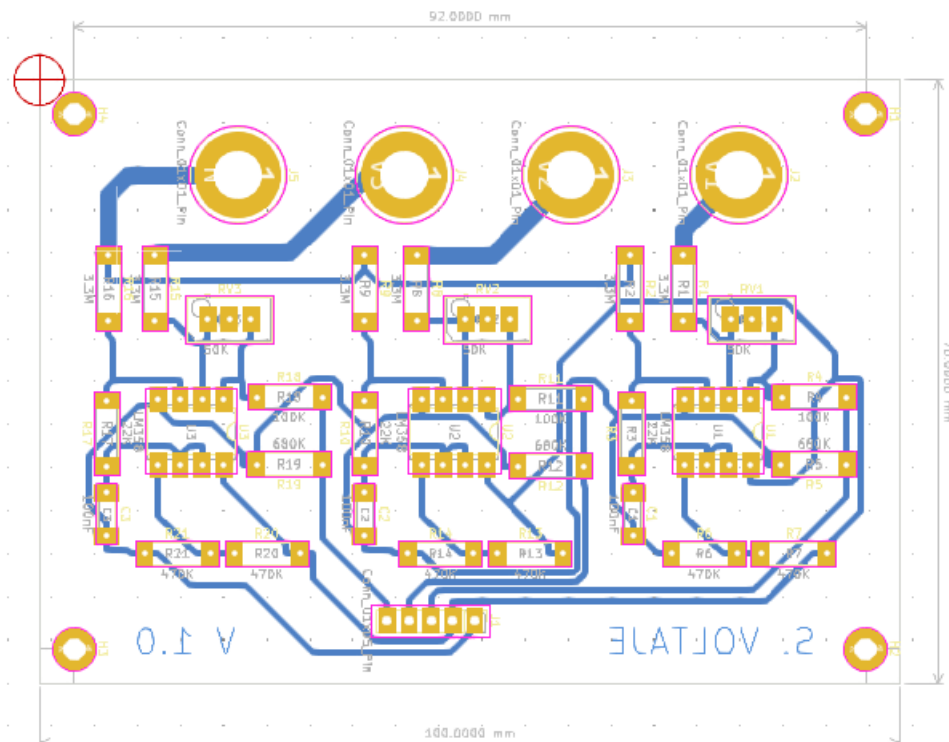


Figura 5.6: Enrutado de pistas del sistema de adecuación de señales de voltaje.

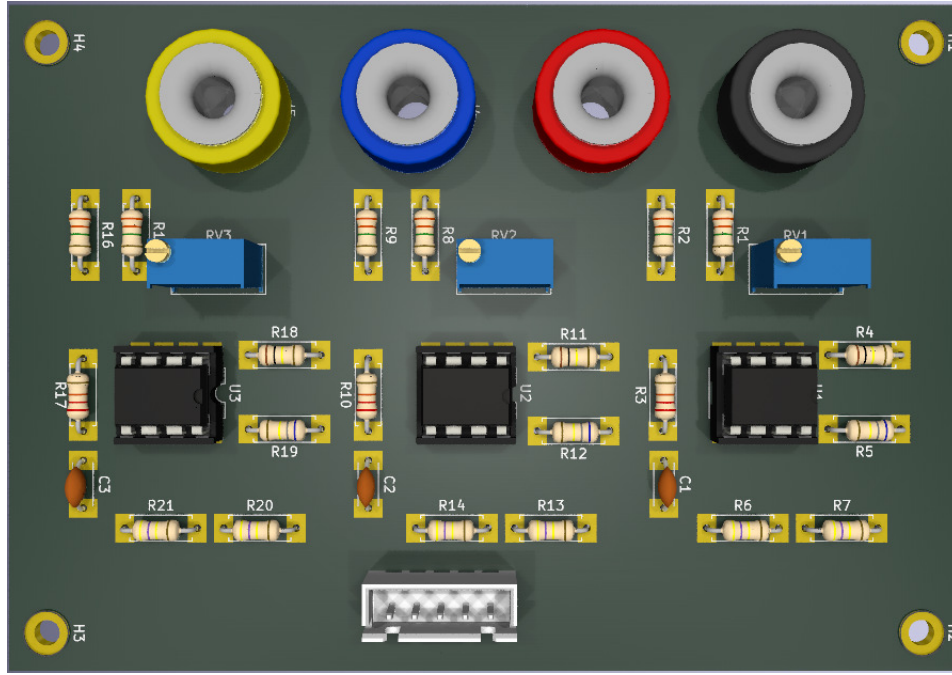


Figura 5.7: Vista 3D del sistema de adecuación de señales de voltaje.

Tabla 5.2: Lista de componentes del circuito de adecuación de voltajes.

Referencia	Descripción	Cantidad
C1, C2, C3	100 nF	3
R1, R2, R8, R9, R15, R16	3.3 M Ω	6
R6, R7, R13, R14, R20, R21	470 k Ω	6
R3, R10, R17	22k Ω	3
R4, R11, R18	100k Ω	3
R5, R12, R19	680k Ω	3
U1, U2, U3	LM358	3
RV1, RV2, RV3	50 k Ω	3
J2, J3, J4, J5	Banana hembra 4 mm	4
J1	01x05 Pin	1

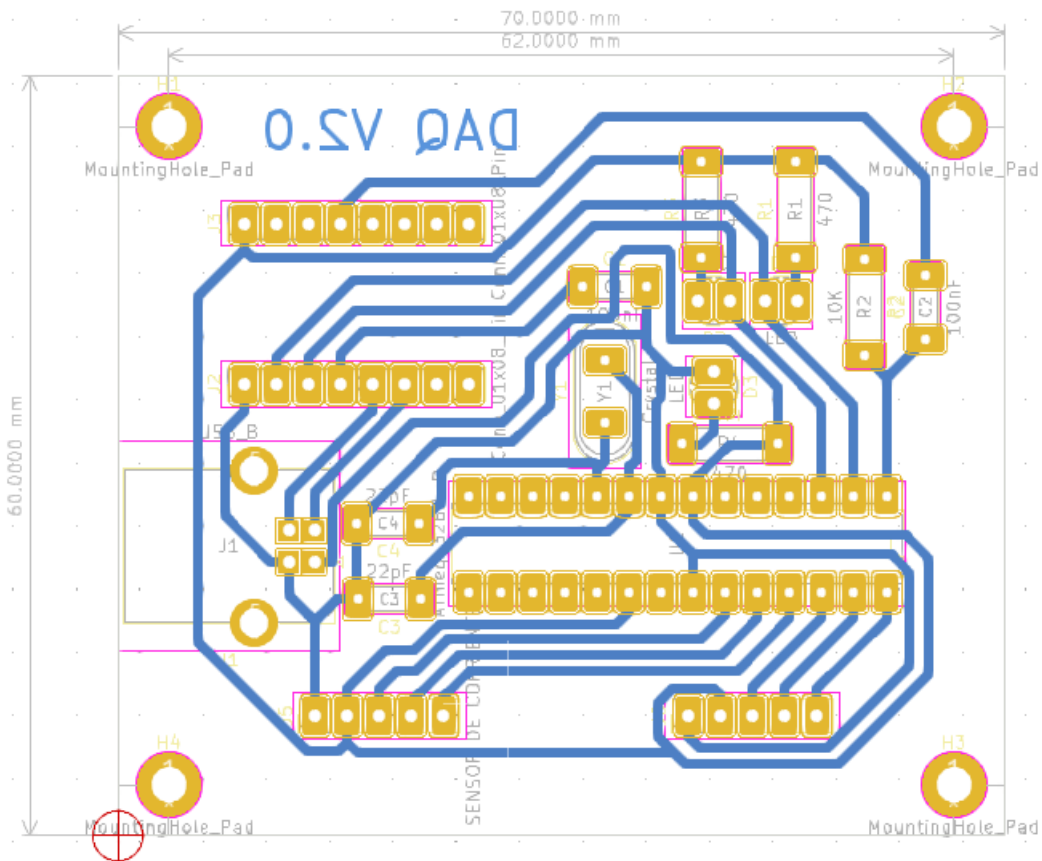


Figura 5.8: Enrutado de pistas del sistema DAQ con ATmega328P.

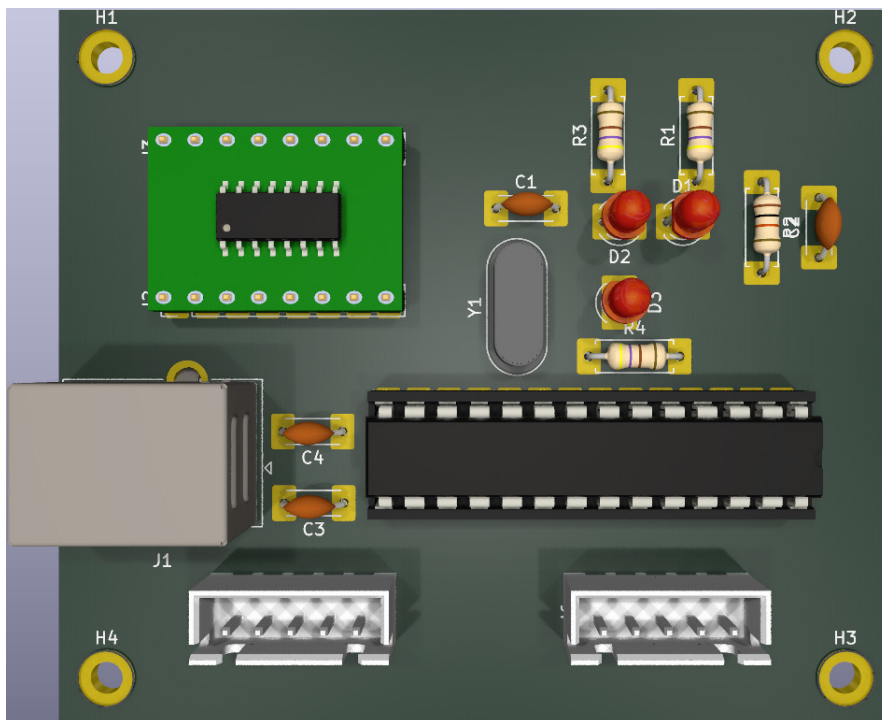


Figura 5.9: Vista 3D del sistema DAQ con ATmega328P.

Tabla 5.3: Lista de componentes DAQ V2 con ATmega328P.

Referencia	Descripción	Cantidad
C1, C2	100 nF	2
C3, C4	22 pF	2
R1, R3, R4	470 Ω	3
R2	10 k Ω	1
D1, D2, D3	LED	3
U2	ATmega328P-PU	1
Y1	Cristal de 16MHz	1
J2, J3	Conn 01x08 Pin	2
J1	USB B	1
J5	01x05 Pin	1
J6	01x05 Pin	1

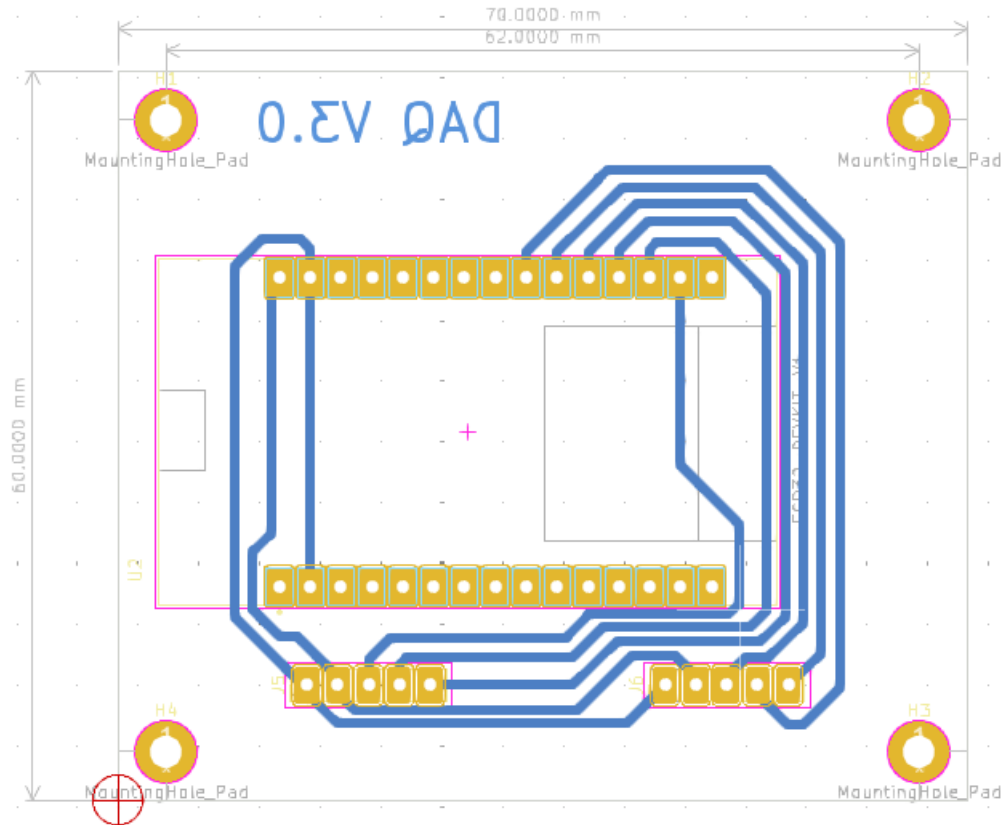


Figura 5.10: Enrutado de pistas del sistema DAQ con ESP32.

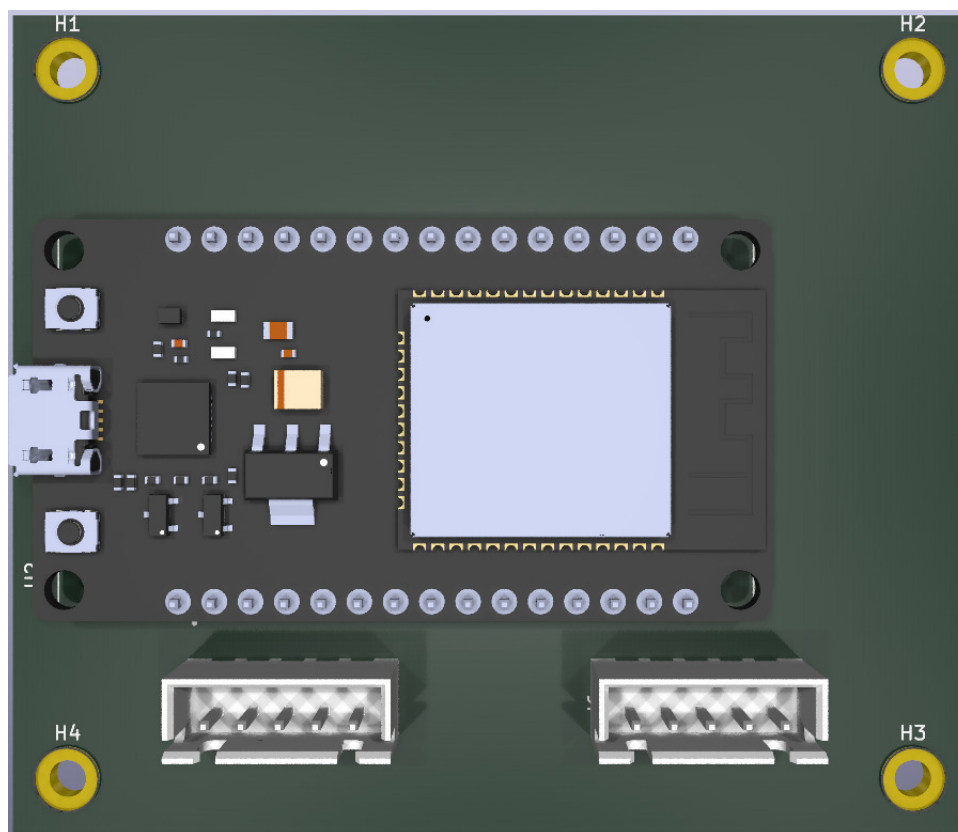


Figura 5.11: Vista 3D del sistema DAQ con ESP32.

Tabla 5.4: Lista de componentes DAQ V3 con ESP32.

Referencia	Descripción	Cantidad
U2	ESP32-DEVKIT-V1	1
J5	01x05 Pin	1
J6	01x05 Pin	1

- Esquemático: dibujo en KiCad de circuitos con op-amp para acondicionamiento y ADC en microcontroladores.
- Layout: enrutamiento de trazas, colocación de componentes.
- Tablas: listado de componentes.

Los diseños redujeron el ruido considerablemente, con el ESP32 ofreciendo mejor integración USB.

5.3. Maquinado de PCB con máquina CNC

Se realizó el maquinado de Control Numerico Computacional (CNC) de los PCB diseñados (Sección 5.2) usando la máquina del laboratorio de Maestría en Ingeniería Eléctrica del ITCM, incluyendo piezas impresas en 3D para sujeción. Esto permite prototipos físicos estables, reduciendo ruido de contactos.

- Preparación: exportar Gerber de KiCad y configurar CNC para fresado.
- Proceso: fresado de cobre, perforación y corte.
- Verificación: inspección visual y continuidad.

Las PCB maquinadas mostraron precisión de 0.1 mm, eliminando inestabilidades presentadas por la protoboard.

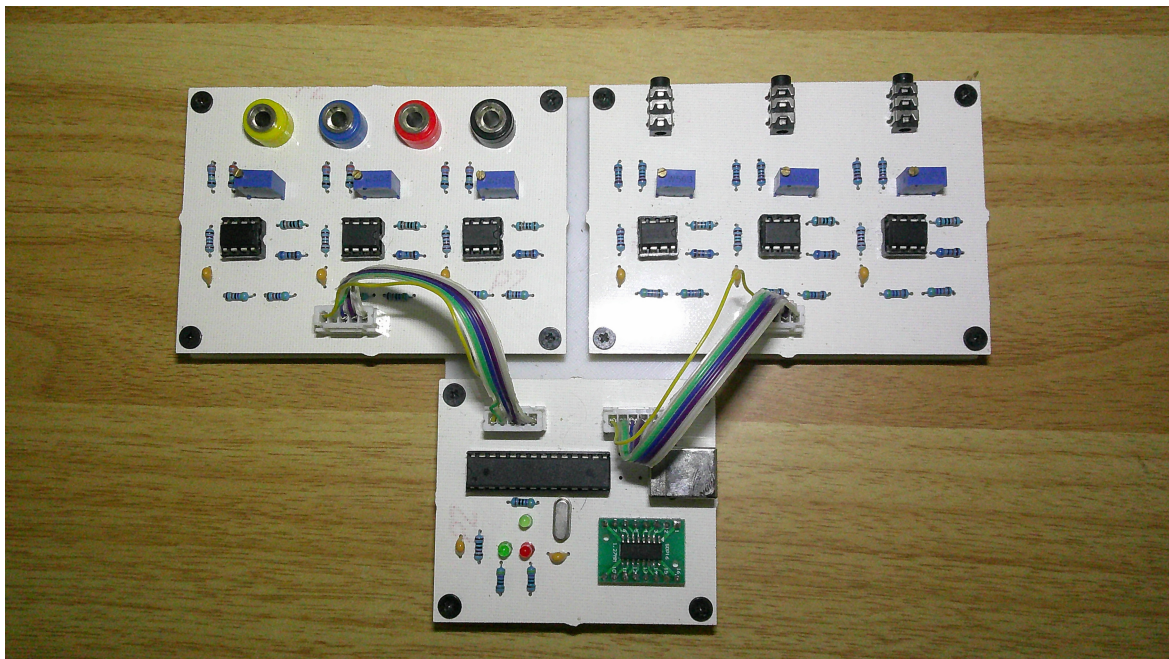


Figura 5.12: Módulos ensamblados y soldados.

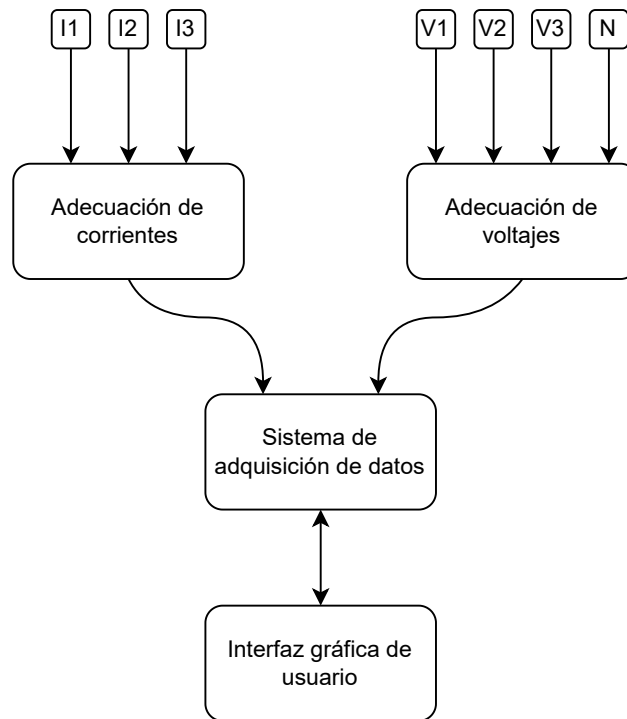


Figura 5.13: Diagrama de conexión entre software y hardware.

5.4. Interfaz de usuario: desarrollo en Scilab con integración Python

La GUI en Scilab visualiza armónicos, formas de onda, con menús y submenús. A continuación, se explica cada sección del código, dividiendo el script en bloques funcionales y destacando su propósito.

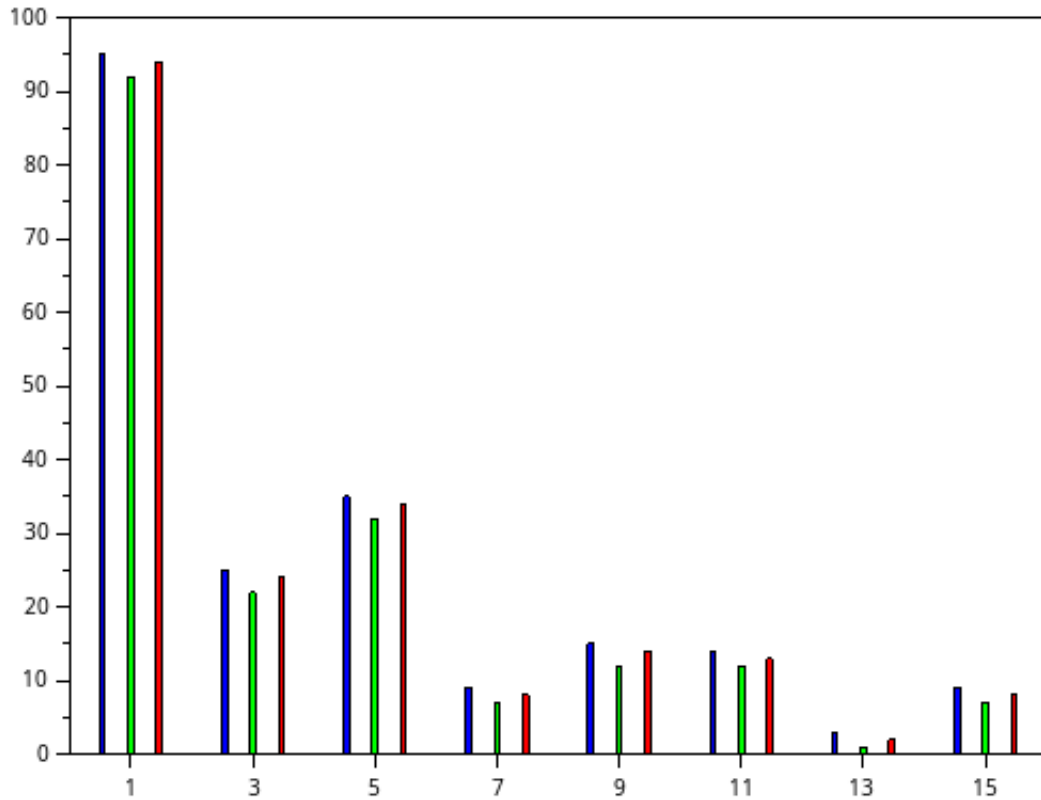


Figura 5.14: Primera versión de la interfaz gráfica de usuario.

5.5. Programa gui.sci

Primero, limpiamos la consola y cerramos ventanas gráficas, y definimos variables globales para la configuración.

```

1      clear;
2      close();
3      global param;
4      global %serial_port;
5      global data;
6      global margin_x margin_y;
7      global frame_w frame_h plot_w plot_h;
8      global handles1 handles2;
    
```

Estas variables globales permiten compartir datos entre funciones, como la configuración del puerto y handles de gráficos. Opciones de configuración serial, definimos arrays para paridad y estados de menús.

```
1 parity = ["n", "o", "e", "m", "s"];
2 pref = ["checked", "on", "off"];
```

La paridad incluye opciones como ninguna, impar, par, etc. Función para truncar a ciclos completos, esta función ajusta la señal a un número entero de ciclos para precisión en el análisis.

```
1 function [corriente_truncada, N_trunc, num_ciclos] =
    truncarCiclosCompletos(corriente, fs, f0)
2 N = length(corriente);
3 longitud_ciclo = round(fs / f0);
4 num_ciclos = floor(N / longitud_ciclo);
5 if num_ciclos < 5 then
6 disp("Advertencia: Menos de 5 ciclos; usando todas las muestras
    .");
7 corriente_truncada = corriente;
8 N_trunc = N;
9 num_ciclos = N * f0 / fs;
10 return;
11 end
12 N_trunc = num_ciclos * longitud_ciclo;
13 corriente_truncada = corriente(1:N_trunc);
14 disp("Truncado a " + string(num_ciclos) + " ciclos completos ("
    + string(N_trunc) + "/" + string(N) + " muestras, f0=" +
    string(f0) + " Hz)");
15 endfunction
```

Asegura al menos 5 ciclos para una FFT robusta. Función para estimar f0, estima la frecuencia fundamental usando FFT con ventana Hanning.

```
1 function f0_estimada = estimarF0(corriente, fs, N)
2 [corriente_trunc, N_trunc, _] = truncarCiclosCompletos(
    corriente, fs, 60);
3 ventana = window('hn', N_trunc);
4 corriente_ventanada = corriente_trunc .* ventana;
5 fft_corriente = fft(corriente_ventanada);
```

```

6      magnitudes = abs(fft_corriente(1:N_trunc/2+1)) / N_trunc;
7      frecuencias = (0:N_trunc/2) * fs / N_trunc;
8      rango_bajo = find(frecuencias >= 50 & frecuencias <= 70);
9      if isempty(rango_bajo) then
10         f0_estimada = 60;
11         disp("Advertencia: No se encontr pico en 50-70 Hz. Usando f0
            =60 Hz.");
12         return;
13     end
14     [max_mag, idx_rel] = max(magnitudes(rango_bajo));
15     idx_abs = rango_bajo(idx_rel);
16     f0_estimada = frecuencias(idx_abs);
17     if abs(f0_estimada - 60) > 5 then
18         f0_estimada = 60;
19         disp("Advertencia: Estimaci n f0=" + string(f0_estimada) + "
            Hz parece inusual. Usando 60 Hz.");
20     else
21         disp("f0 estimada: " + string(f0_estimada) + " Hz");
22     end
23     endfunction

```

Busca el pico en 50-70 Hz y usa 60 Hz como fallback. Función para conectar al puerto serial, abre y lee datos del puerto serial.

```

1      function conectar()
2      global param;
3      global %serial_port;
4      global data;
5      disp(param.conectado);
6      param_com = [string(param.baud) + ', ' + param.parity + ', ' +
            string(param.data_bits) + ', ' + string(param.stop_bits)];
7      %serial_port = openserial("/dev/ttyUSB" + string(param.port),
            param_com);
8      disp(%serial_port);
9      values = [ascii(0)];
10     data = [ascii(0)];

```

```
11     value = ascii(0);
12     while param.conectado
13         if (readserial(%serial_port, 1)) == ascii(36) then
14             while (value ~= ascii(42))
15                 value = readserial(%serial_port, 1);
16                 values = values + value;
17             end
18             disp(values);
19             data = strsplit(values, ',');
20             values = [ascii(0)];
21             value = ascii(0);
22         end
23     end
24 endfunction
```

Lee tramas delimitadas por \$ y *. Función para desconectar, cierra el puerto serial.

```
1     function desconectar()
2         global %serial_port;
3         closeserial(%serial_port);
4     endfunction
```

Simple cierre del handle serial. Función para cargar configuración, carga parámetros desde ini o usa defaults.

```
1     function param = cargarConfiguracion()
2         global param;
3         param = struct(...
4             'port', 0, ...
5             'baud', 9600, ...
6             'parity', 'n', ...
7             'data_bits', 8, ...
8             'stop_bits', 1, ...
9             'conectado', %f, ...
10            'directorio', pwd() ...
11        );
12        if ~isfile('configuracion.ini') then
```

```

13     disp("Usando valores por defecto - archivo no encontrado");
14     return;
15 end
16 try
17     contenido = mgetl('configuracion.ini');
18     for i = 1:size(contenido, 1)
19         linea = stripblanks(contenido(i));
20         if linea == "" | part(linea, 1) == "[" | part(linea, 1) == ";"
21             then
22                 continue;
23         end
24         partes = strsplit(linea, "=");
25         if size(partes, '*') >= 2 then
26             clave = stripblanks(partes(1));
27             valor = stripblanks(partes(2));
28             select clave
29             case "port" then param.port = evstr(valor);
30             case "baud" then param.baud = evstr(valor);
31             case "parity" then param.parity = valor;
32             case "data_bits" then param.data_bits = evstr(valor);
33             case "stop_bits" then param.stop_bits = evstr(valor);
34             case "conectado" then param.conectado = (valor == "T" | valor
35                 == "%T" | valor == "true");
36             case "directorio" then param.directorio = valor;
37         end
38     end
39     disp("Configuracion cargada correctamente");
40     catch
41         disp("Error al leer configuracion, usando valores por defecto")
42         ;
43     lasterror();
44 end
45 endfunction

```

Usa struct para parámetros y try-catch para errores. Función para guardar configuración, escribe parámetros en ini.

```
1      function guardarConfiguracion(param)
2          archivo = mopen('configuracion.ini', 'w');
3          fprintf(archivo, "[parametros]\n");
4          fprintf(archivo, "port=%d\n", param.port);
5          fprintf(archivo, "baud=%d\n", param.baud);
6          fprintf(archivo, "parity=%s\n", param.parity);
7          fprintf(archivo, "data_bits=%d\n", param.data_bits);
8          fprintf(archivo, "stop_bits=%d\n", param.stop_bits);
9          fprintf(archivo, "conectado=%s\n", string(param.conectado));
10         fprintf(archivo, "directorio=%s\n", param.directorio);
11         fclose(archivo);
12         disp("Configuracion guardada en configuracion.ini");
13     endfunction
```

Usa fprintf para formato ini. Función para actualizar menús, actualiza el estado checked en menús.

```
1      function updateChecked(menuHandle, selectedLabel)
2          children = get(menuHandle, "children");
3          for i = 1:length(children)
4              if get(children(i), "label") == selectedLabel then
5                  set(children(i), "checked", "on");
6              else
7                  set(children(i), "checked", "off");
8              end
9          end
10     endfunction
```

Itera sobre submenús para marcar la opción seleccionada. Funciones para configurar parámetros, actualizan param y menús.

```
1      function setPort(port)
2          global param;
3          param.port = port;
4          updateChecked(h20, "tyUSB" + string(port));
```

```

5     guardarConfiguracion(param);
6     endfunction
7     // Similar para setBaudRate, setParity, etc. (omitido por
        brevedad)

```

Llaman a updateChecked y guardan config. Función para tomar muestras, ejecuta Python, lee Valores separados por comas (CSV), gráfica y analiza.

```

1     function tomarMuestra()
2     global param;
3     global datos2;
4     global datos3;
5     global handles1;
6     global handles2;
7     // Ejecutar comunicacion.py (omitido por brevedad)
8     try
9     datos = csvRead(latest_csv);
10    corriente_filtrada = csvRead(param.directorio + "/"
        corriente_filtrada.csv");
11    N = size(datos, 1);
12    fs = 20000;
13    t = (0:N-1)' / fs;
14    datos3 = [t, datos(:,4:6), corriente_filtrada];
15    datos2 = [t, datos(:,1:3)];
16    guardarMuestra(datos);
17    subplot(3,1,1);
18    clf();
19    title("Señales de corriente");
20    plot(datos3(:,1), datos3(:,2), 'color', 'b', 'linestyle', '-');
21    plot(datos3(:,1), datos3(:,3), 'color', 'r', 'linestyle', '-');
22    plot(datos3(:,1), datos3(:,4), 'color', 'k', 'linestyle', '-');
23    handles1 = struct();
24    legend(['I1', 'I2', 'I3']);
25    // Similar para voltajes, analisis armónicos y gráficos
26    catch
27    disp("Error al leer los archivos CSV: " + lasterror());

```

```
28     end
29 endfunction
```

Función para guardar muestras, guarda datos en CSV.

```
1     function guardarMuestra(datos)
2     global param;
3     nombreArchivo = param.directorio + "/muestra.csv";
4     csvWrite(datos, nombreArchivo);
5     disp("Muestra guardada en: " + nombreArchivo);
6 endfunction
```

Función para seleccionar directorio, usa uigetdir para elegir ruta.

```
1     function seleccionarDirectorio()
2     global param;
3     directorioSeleccionado = uigetdir();
4     if directorioSeleccionado ~= "" then
5     param.directorio = directorioSeleccionado;
6     guardarConfiguracion(param);
7     disp("Directorio seleccionado: " + param.directorio);
8     else
9     disp("No se selecciono ning n directorio.");
10    end
11 endfunction
```

Función para cargar muestras, usa uigetfile para cargar CSV y actualizar gráficos.

```
1     function cargarMuestra()
2     global param;
3     // ... (selección de archivo, lectura, organización de datos,
4         gráficos, análisis armónicos)
5 endfunction
```

Función para mostrar about, mensaje simple sobre el autor.

```
1     function FigMain_About()
2     msg = sprintf(gettext("Análisis de calidad de la energía,
        prototipo experimental"));
```

```

3     messagebox(msg, gettext("Sobre"), "info", "modal");
4     endfunction

```

Función para análisis de armónicos, integra estimación f_0 y truncado, haciendo uso de las ecuaciones 2.7 ,2.16 , 2.17 para el cálculo del THD así como del factor de potencia.

```

1     function [porcentaje_armonicos, thd, DPF, PF_total] =
        analizarArmonicos(corriente, fs, N_orig)
2     f0 = estimarF0(corriente, fs, N_orig);
3     [corriente_trunc, N_trunc, num_ciclos] = truncarCiclosCompleto
        (corriente, fs, f0);
4     ventana = window('hn', N_trunc);
5     corriente_ventanada = corriente_trunc .* ventana;
6     fft_corriente = fft(corriente_ventanada);
7     magnitudes = abs(fft_corriente) / N_trunc;
8     frecuencias = (0:N_trunc-1) * fs / N_trunc;
9     indices_armonicos = round((1:15) * f0 * N_trunc / fs) + 1;
10    magnitudes_armonicos = magnitudes(indices_armonicos);
11    porcentaje_armonicos = (magnitudes_armonicos /
        magnitudes_armonicos(1)) * 100;
12    thd = sqrt(sum(magnitudes_armonicos(2:$).^2)) /
        magnitudes_armonicos(1) * 100;
13    thd_decimal = thd / 100;
14    DPF = 1 / sqrt(1 + thd_decimal^2);
15    PF_total = 1 * DPF;
16    disp("Porcentaje de armonicos:");
17    for n = 1:15
18    disp("Armonico " + string(n) + ": " + string(
        porcentaje_armonicos(n)) + "%");
19    end
20    disp("THD: " + string(thd) + "%");
21    disp("Factor de Potencia de Distorsion (DPF): " + string(DPF));
22    disp("Factor de Potencia Total (PF_total, asumiendo
        FP_desplazamiento=1): " + string(PF_total));
23    endfunction

```

Función para graficar armónicos, gráfico de barras con etiquetas.

```
1      function graficarArmonicos(orden_armonico, porcentaje)
2      bar(orden_armonico, porcentaje, 0.2, 'blue');
3      xlabel('Orden Armonico');
4      ylabel('Porcentaje (%)');
5      title('Porcentaje de Armonicos');
6      xgrid;
7      for i = 1:length(orden_armonico)
8      x = orden_armonico(i);
9      y = porcentaje(i);
10     texto = string(round(porcentaje(i))) + "%";
11     xstring(x, y + 1, texto);
12     end
13     endfunction
```

Función para toggle gráficas, oculta/muestra curvas.

```
1      function toggleGrafica(grafica, handles)
2      select grafica
3      case 1 then
4      if handles.h3.visible == "on" then
5      handles.h3.visible = "off";
6      else
7      handles.h3.visible = "on";
8      end
9      case 2 then
10     if handles.h2.visible == "on" then
11     handles.h2.visible = "off";
12     else
13     handles.h2.visible = "on";
14     end
15     case 3 then
16     if handles.h1.visible == "on" then
17     handles.h1.visible = "off";
18     else
19     handles.h1.visible = "on";
```

```
20     end
21     case 5 then
22     if handles.h3.visible == "on" then
23     handles.h3.visible = "off";
24     else
25     handles.h3.visible = "on";
26     end
27     case 6 then
28     if handles.h2.visible == "on" then
29     handles.h2.visible = "off";
30     else
31     handles.h2.visible = "on";
32     end
33     case 7 then
34     if handles.h1.visible == "on" then
35     handles.h1.visible = "off";
36     else
37     handles.h1.visible = "on";
38     end
39     end
40     endfunction
```

Inicialización gráfica y menús, crea ventana y menús personalizados.

```
1     frame_w = 550; frame_h = 1050;
2     plot_w = 600; plot_h = frame_h;
3     margin_x = 15; margin_y = 15;
4     defaultfont = "arial";
5     axes_w = 3 * margin_x + frame_w + plot_w;
6     axes_h = 2 * margin_y + frame_h;
7     FigMain = scf(1);
8     FigMain.background = -2;
9     FigMain.figure_position = [100 100];
10    FigMain.figure_name = gettext("Calidad de la energia");
11    // Eliminar men s y barra de herramientas predeterminados
12    delmenu(FigMain.figure_id, gettext("&File"));
```

```

13     delmenu(FigMain.figure_id, gettext("&Tools"));
14     delmenu(FigMain.figure_id, gettext("&Edit"));
15     delmenu(FigMain.figure_id, gettext("&?"));
16     toolbar(FigMain.figure_id, "off");
17     // Crear men s personalizados
18     h1 = uimenu("parent", FigMain, "label", gettext("Conexi n"));
19     // ... (men s completos omitidos por brevedad, pero incluidos
        en el c digo)

```

Carga inicial y gráficos de prueba, carga config y datos de prueba.

```

1     param = cargarConfiguracion();
2     disp(param);
3     datos3 = csvRead('/home/user/corrientes.csv');
4     datos2 = csvRead('/home/user/voltajes.csv');
5     N = size(datos3, 1);
6     fs = 20000;
7     t = (0:N-1)' / fs;
8     datos3 = [t, datos3(:,2:4)];
9     datos2 = [t, datos2(:,2:4)];
10    subplot(3,1,1);
11    title("Senales de corriente");
12    plot(datos3(:,1), datos3(:,2), 'color', 'b');
13    plot(datos3(:,1), datos3(:,3), 'color', 'r');
14    plot(datos3(:,1), datos3(:,4), 'color', 'k');
15    legend(['I1', 'I2', 'I3']);
16    subplot(3,1,2);
17    title("Senales de voltaje");
18    plot(datos2(:,1), datos2(:,2), 'color', 'b');
19    plot(datos2(:,1), datos2(:,3), 'color', 'r');
20    plot(datos2(:,1), datos2(:,4), 'color', 'k');
21    legend(['V1', 'V2', 'V3']);
22    subplot(3,1,3);
23    orden_armonico = [1 3 5 7 9 11 13 15];
24    [porcentaje_armonicos, thd, DPF, PF_total] = analizarArmonicos(
        datos3(:,2), fs, N);

```

```

25     porcentaje = [porcentaje_armonicos(1) porcentaje_armonicos(3)
                    porcentaje_armonicos(5) porcentaje_armonicos(7)
                    porcentaje_armonicos(9) porcentaje_armonicos(11)
                    porcentaje_armonicos(13) porcentaje_armonicos(15)];
26     graficarArmonicos(orden_armonico, porcentaje);
27     sleep(500);

```

5.6. Programa filtrado.py

Se presenta un script de Python diseñado para leer datos de corriente desde un archivo CSV, aplicar un filtro pasa-bajos, graficar la señal original y filtrada, y guardar los resultados. A continuación, se explica cada sección del código, dividiendo el script en bloques funcionales y destacando su propósito. Se importan las librerías necesarias para procesamiento numérico, señales y FFT.

```

1     import numpy as np
2     from scipy import signal
3     from scipy.fft import fft

```

numpy para cálculos, scipy.signal para filtros, y scipy.fft para transformada rápida de Fourier. Lectura y Normalización de Datos: leemos la cuarta columna (i1) y la normalizamos.

```

1     datos = np.genfromtxt('corrientes.csv', delimiter=',')
2     corriente = datos[:,3]
3     corriente = (corriente - np.mean(corriente)) * 0.0130709

```

La función np.genfromtxt lee el CSV. Restamos la media y escalamos por 0.0130709 para convertir a amperios. Estimación de Frecuencia Fundamental (f0), aplicamos ventana Hanning, calculamos FFT y buscamos el pico en 50–70 Hz.

```

1     fs = 20000
2     N = len(corriente)
3     ventana = np.hanning(N)
4     corriente_ventanada = corriente * ventana
5     fft_corriente = fft(corriente_ventanada)

```

```

6     magnitudes = np.abs(fft_corriente[:N//2+1]) / N
7     frecuencias = np.arange(N//2+1) * fs / N
8     rango_bajo = np.where((frecuencias >= 50) & (frecuencias <= 70)
9         )[0]
10    if len(rango_bajo) == 0:
11        f0 = 60
12        print("Advertencia: No se encontro pico en 50-70 Hz. Usando f0
13            =60 Hz.")
14    else:
15        idx_pico = rango_bajo[np.argmax(magnitudes[rango_bajo])]
16        f0 = frecuencias[idx_pico]
17        if abs(f0 - 60) > 5:
18            f0 = 60
19            print(f"Advertencia: Estimacion f0={f0:.2f} Hz parece inusual.
20                Usando 60 Hz.")
21        else:
22            print(f"f0 estimada: {f0:.2f} Hz")

```

Se usa una ventana Hanning para reducir fugas espectrales. Si no se detecta pico en el rango esperado, se usa $f_0 = 60$ Hz como respaldo. Ajustamos la señal para que contenga un número entero de ciclos.

```

1     longitud_ciclo = round(fs / f0)
2     num_ciclos = int(np.floor(N / longitud_ciclo))
3     if num_ciclos < 5:
4         print("Advertencia: Menos de 5 ciclos; usando todas las
5             muestras.")
6         corriente_truncada = corriente
7         N_trunc = N
8     else:
9         N_trunc = num_ciclos * longitud_ciclo
10        corriente_truncada = corriente[:N_trunc]
11        print(f"Truncado a {num_ciclos} ciclos completos ({N_trunc}/{N}
12            muestras, f0={f0:.2f} Hz)")

```

Esto mejora la precisión del análisis armónico al evitar truncamiento parcial de ciclos. Apli-

camos un filtro pasa-bajos de orden 4.

```
1      b, a = signal.butter(4, 2000/(fs/2), 'low')
2      filtrada = signal.filtfilt(b, a, corriente_truncada)
```

Frecuencia de corte: 2000 Hz. `filtfilt` aplica el filtro en ambas direcciones (sin desfase). Guardamos la señal filtrada para uso posterior (por ejemplo, en Scilab).

```
1      np.savetxt('corriente_filtrada.csv', filtrada, delimiter=',')
```

El archivo se guarda con comas como delimitador, listo para ser leído por otros scripts.

5.7. Programa comunicacion.py

Se presenta un script de Python diseñado para leer datos seriales desde un dispositivo ESP32, procesarlos y guardarlos en un archivo CSV. El script usa argumentos de línea de comandos para configurar el puerto serial y guarda los datos con un nombre de archivo que incluye la fecha y hora. A continuación, se explica cada sección del código, dividiendo el script en bloques funcionales. Comenzamos importando las librerías necesarias y configurando los argumentos de línea de comandos.

```
1      #!/usr/bin/python
2      import serial
3      import time
4      import csv
5      from datetime import datetime
6      import os
7      import argparse
```

El script inicia con la línea `#!/usr/bin/python` para ejecutarse como un script de Python. Las librerías `serial` manejan la comunicación serial, `time` gestiona pausas, `csv` permite escribir archivos CSV, `datetime` genera marcas de tiempo, `os` maneja directorios, y `argparse` procesa argumentos de línea de comandos. Definimos los argumentos para configurar el puerto serial y el directorio de salida.

```
1      parser = argparse.ArgumentParser(description='Lectura de datos
          seriales desde ESP32')
```

```

2 parser.add_argument('--port', type=int, default=1, help='Numero
    de puerto (por defecto: 1)')
3 parser.add_argument('--baud', type=int, default=115200, help='
    Baud rate (por defecto: 115200)')
4 parser.add_argument('--parity', type=str, default='n', choices
    =['n', 'e', 'o'], help='Paridad (n: none, e: even, o: odd)')
5 parser.add_argument('--data_bits', type=int, default=8, choices
    =[5, 6, 7, 8], help='Bits de datos (por defecto: 8)')
6 parser.add_argument('--stop_bits', type=int, default=1, choices
    =[1, 2], help='Bits de parada (por defecto: 1)')
7 parser.add_argument('--directorio', type=str, default="/home/
    luis-miranda/Documentos/silab_codes", help='Directorio para
    guardar los archivos')
8 args = parser.parse_args()

```

La clase ArgumentParser define argumentos como -port (número de puerto), -baud (velocidad en baudios), -parity (paridad), -data_bits (bits de datos), -stop_bits (bits de parada), y -directorio (ruta de guardado). Los valores por defecto permiten ejecutar el script sin argumentos. Configuramos el puerto serial usando los argumentos proporcionados y un mapeo de paridad.

```

1 parity_map = {'n': serial.PARITY_NONE, 'e': serial.PARITY_EVEN,
    'o': serial.PARITY_ODD}
2 ser = serial.Serial(
3     port=f'/dev/ttyUSB{args.port}',
4     baudrate=args.baud,
5     parity=parity_map[args.parity],
6     stopbits=args.stop_bits,
7     bytesize=args.data_bits,
8     timeout=2
9 )

```

El diccionario parity_map mapea las opciones de paridad a constantes de la librería serial. La función serial.Serial configura el puerto serial (por ejemplo, /dev/ttyUSB1) con los parámetros especificados y un tiempo de espera de 2 segundos. Creamos el directorio de salida si no existe y generamos un nombre de archivo con la fecha y hora.

```

1     if not os.path.exists(args.directorio):
2         os.makedirs(args.directorio)
3         fecha_hora = datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
4         nombre_archivo = f'datos_serial_{fecha_hora}.csv'
5         ruta_completa = os.path.join(args.directorio, nombre_archivo)

```

La función `os.path.exists` verifica si el directorio existe, y `os.makedirs` lo crea si es necesario. `datetime.now().strftime` genera una marca de tiempo y `os.path.join` construye la ruta completa del archivo CSV. Inicializamos el archivo CSV con un encabezado y preparamos una lista para almacenar los datos.

```

1     columna_actual = 0
2     filas = [[] for _ in range(5000)]
3     with open(ruta_completa, mode='w', newline='') as file:
4         writer = csv.writer(file)
5         writer.writerow(['v1', 'v2', 'v3', 'i1', 'i2', 'i3'])

```

La variable `columna_actual` rastrea la columna actual, y `filas` crea una lista de 5000 filas vacías. El bloque `with` abre el archivo CSV en modo escritura y escribe el encabezado con las columnas `v1`, `v2`, `v3`, `i1`, `i2`, `i3` (voltajes y corrientes). Enviamos comandos al DAQ para iniciar la transmisión de datos.

```

1     commands = [b'h', b'a', b'b', b'c', b'd', b'e', b'f', b'g']
2     for cmd in commands:
3         ser.write(cmd)
4         time.sleep(1)

```

La lista `commands` contiene comandos en bytes (de `h` a `g`) que se envían al ESP32 usando `ser.write`. La pausa de 1 segundo (`time.sleep(1)`) da tiempo al ESP32 para procesar cada comando. Leemos los datos del puerto serial, los procesamos y los almacenamos en la lista de filas.

```

1     current_array = None
2     current_row = 0
3     while True:
4         if ser.in_waiting > 0:
5             respuesta = ser.readline().decode('ascii').strip()

```

```

6      print("Respuesta del ESP32:", respuesta)
7      if respuesta in ['v1', 'v2', 'v3', 'i1', 'i2', 'i3']:
8          current_array = respuesta
9          current_row = 0
10         continue
11     if respuesta == 't':
12         columna_actual += 1
13     print(f"Cambiando a la columna {columna_actual}")
14     current_array = None
15     continue
16     if current_array:
17         try:
18             value = int(respuesta)
19             column_map = {'v1': 0, 'v2': 1, 'v3': 2, 'i1': 3, 'i2': 4, 'i3': 5}
20             while len(filas[current_row]) <= column_map[current_array]:
21                 filas[current_row].append('')
22                 filas[current_row][column_map[current_array]] = value
23                 current_row += 1
24             except ValueError:
25                 print(f"Dato invalido recibido, ignorando: {respuesta}")
26                 continue
27             if columna_actual >= 6:
28                 print("Todas las columnas recibidas. Saliendo del bucle...")
29                 break

```

El bucle `while True` lee datos del puerto serial. La función `ser.readline().decode('ascii')` lee y decodifica las respuestas. Identificamos el inicio de un arreglo (`v1`, `v2`, `v3`, `i1`, `i2`, `i3`), el cambio de columna (`t`), o datos numéricos, que se almacenan en filas según el mapeo de columnas. Los datos inválidos se ignoran con un bloque `try-except`. El bucle termina cuando se procesan las 6 columnas. Escribimos todas las filas en el archivo CSV.

```

1      with open(ruta_completa, mode='w', newline='') as file:
2          writer = csv.writer(file)
3          writer.writerow(['v1', 'v2', 'v3', 'i1', 'i2', 'i3'])

```

```
writer.writerows(filas)
```

El bloque with reabre el archivo CSV, escribe el encabezado y todas las filas de datos almacenadas en filas. Enviamos un comando para reiniciar el ESP32 y cerramos la conexión serial.

```
ser.write(b'z')
time.sleep(1)
ser.close()
```

El comando b'z' reinicia el ESP32, y ser.close() cierra la conexión serial.

5.8. Programa daq.c

Este código en C para ESP32 implementa la adquisición continua de 6 canales ADC (3 voltajes + 3 corrientes). Con una frecuencia de muestreo de 20 kHz por canal (120 kHz total), así como buffers de 5000 muestras por canal y transmisión por puerto serial controlada por comandos, haciendo uso de FreeRTOS, interrupciones, y calibración ADC.

Se explica el código a continuación: Se incluyen las librerías necesarias y se definen constantes.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/adc.h"
#include "esp_adc_cal.h"
#include "esp_log.h"

#define TAG "ADC_CONTINUOUS"

#define ADC_CHANNELS 6
#define ADC_SAMPLES_PER_CHANNEL 5000
#define ADC_FREQ_PER_CHANNEL 20000
#define BUFFER_SIZE (ADC_SAMPLES_PER_CHANNEL * ADC_CHANNELS *
2)
```

TAG se usa para logs. BUFFER_SIZE reserva espacio para 5000 muestras $\times$ 6 canales $\times$ 2 bytes.

Buffers y banderas de control.

```
1      static esp_adc_cal_characteristics_t adc_chars;
2      static uint16_t *v1, *v2, *v3, *i1, *i2, *i3;
3      static bool v1tx = false, v2tx = false, v3tx = false;
4      static bool i1tx = false, i2tx = false, i3tx = false;
5      static bool start = false;
```

Los punteros v1, i1, etc., almacenan los datos. Las banderas *tx controlan la transmisión.

Envía un canal completo por serial.

```
1      static void transmit_data(const char* label, uint16_t* data) {
2          printf("%s\n", label);
3          for (int i = 0; i < ADC_SAMPLES_PER_CHANNEL; i++) {
4              printf("%u\n", data[i]);
5          }
6          printf("t\n");
7      }
```

Formato: v1 valor1 valor2 ... t Procesa datos en tiempo real desde el buffer .

```
1      static bool IRAM_ATTR adc_continuous_callback(
2          adc_continuous_handle_t handle, const
3          adc_continuous_evt_data_t *evt, void *user_data) {
4          if (!start) return true;
5          uint8_t *data = evt->data;
6          uint32_t size = evt->size / 2;
7          static int sample_index = 0;
8          for (uint32_t i = 0; i < size; i += ADC_CHANNELS) {
9              if (sample_index < ADC_SAMPLES_PER_CHANNEL) {
10                 v1[sample_index] = (data[i * 2] | (data
11                     [i * 2 + 1] << 8)) & 0xFFFF;
12                 v2[sample_index] = (data[(i + 1) * 2] |
13                     (data[(i + 1) * 2 + 1] << 8)) & 0
14                     x0FFF;
15                 v3[sample_index] = (data[(i + 2) * 2] |
16                     (data[(i + 2) * 2 + 1] << 8)) & 0
```

```

11         x0FFF;
        i1[sample_index] = (data[(i + 3) * 2] |
        (data[(i + 3) * 2 + 1] << 8)) & 0
        x0FFF;
12        i2[sample_index] = (data[(i + 4) * 2] |
        (data[(i + 4) * 2 + 1] << 8)) & 0
        x0FFF;
13        i3[sample_index] = (data[(i + 5) * 2] |
        (data[(i + 5) * 2 + 1] << 8)) & 0
        x0FFF;
14        sample_index++;
15    }
16    }
17    if (sample_index >= ADC_SAMPLES_PER_CHANNEL) {
18        start = false;
19        adc_continuous_stop(handle);
20        ESP_LOGI(TAG, "Captura completada");
21        sample_index = 0;
22    }
23    return true;
24    }

```

IRAM_ATTR asegura ejecución en interna. Se desempaquetan 12 bits por muestra. Inicialización y bucle de comandos.

```

1    void app_main(void) {
2        // Asignar memoria
3        v1 = (uint16_t*)malloc(ADC_SAMPLES_PER_CHANNEL * sizeof
        (uint16_t));
4        v2 = (uint16_t*)malloc(ADC_SAMPLES_PER_CHANNEL * sizeof
        (uint16_t));
5        v3 = (uint16_t*)malloc(ADC_SAMPLES_PER_CHANNEL * sizeof
        (uint16_t));
6        i1 = (uint16_t*)malloc(ADC_SAMPLES_PER_CHANNEL * sizeof
        (uint16_t));
7        i2 = (uint16_t*)malloc(ADC_SAMPLES_PER_CHANNEL * sizeof

```

```

            (uint16_t));
8      i3 = (uint16_t*)malloc(ADC_SAMPLES_PER_CHANNEL * sizeof
            (uint16_t));
9      // Configuraci n del handle ADC
10     adc_continuous_handle_t handle = NULL;
11     adc_continuous_handle_cfg_t adc_config = {
12         .max_store_buf_size = BUFFER_SIZE,
13         .conv_frame_size = ADC_SAMPLES_PER_CHANNEL *
            ADC_CHANNELS,
14     };
15     ESP_ERROR_CHECK(adc_continuous_new_handle(&adc_config,
            &handle));

```

Se definen 6 canales con atenuación de 11 dB.

```

1      adc_continuous_channel_t channels[ADC_CHANNELS] = {
2          { .channel = ADC_CHANNEL_0, .atten =
            ADC_ATTEN_DB_11, .bit_width =
            ADC_BITWIDTH_12 }, // GPIO36
3          { .channel = ADC_CHANNEL_3, .atten =
            ADC_ATTEN_DB_11, .bit_width =
            ADC_BITWIDTH_12 }, // GPIO39
4          { .channel = ADC_CHANNEL_6, .atten =
            ADC_ATTEN_DB_11, .bit_width =
            ADC_BITWIDTH_12 }, // GPIO34
5          { .channel = ADC_CHANNEL_7, .atten =
            ADC_ATTEN_DB_11, .bit_width =
            ADC_BITWIDTH_12 }, // GPIO35
6          { .channel = ADC_CHANNEL_4, .atten =
            ADC_ATTEN_DB_11, .bit_width =
            ADC_BITWIDTH_12 }, // GPIO32
7          { .channel = ADC_CHANNEL_5, .atten =
            ADC_ATTEN_DB_11, .bit_width =
            ADC_BITWIDTH_12 }, // GPIO33
8      };

```

Configuración del modo continuo con frecuencia total: 120 kHz (6×20 kHz).

```
1      adc_continuous_config_t dig_cfg = {
2          .sample_freq_hz = ADC_FREQ_PER_CHANNEL *
3              ADC_CHANNELS,
4          .conv_mode = ADC_CONV_SINGLE_UNIT_1,
5          .format = ADC_DIGI_FORMAT_12BIT,
6          .pattern_num = ADC_CHANNELS,
7          .pattern = channels,
8      };
9      ESP_ERROR_CHECK(adc_continuous_config(handle, &dig_cfg)
10          );
```

Se calibra el ADC y se registra el callback.

```
1      esp_adc_cal_characterize(ADC_UNIT_1, ADC_ATTEN_DB_11,
2          ADC_WIDTH_BIT_12, 1100, &adc_chars);
3      adc_continuous_evt_cbs_t cbs = {
4          .on_conv_done = adc_continuous_callback,
5      };
6      ESP_ERROR_CHECK(adc_continuous_register_event_callbacks
7          (handle, &cbs, NULL));
```

Control por UART (comandos: a, b, ..., z).

```
1      while (1) {
2          if (uart_get_buffered_data_len(UART_NUM_0, (
3              size_t*)&start) > 0) {
4              char mensaje;
5              uart_read_bytes(UART_NUM_0, &mensaje,
6                  1, portMAX_DELAY);
7              switch (mensaje) {
8                  case 'z': start = false;
9                      adc_continuous_stop(handle);
10                      break;
11                  case 'a': start = true;
12                      adc_continuous_start(handle);
13                      break;
```

```

8           case 'b': v1tx = true; break;
9           case 'c': v2tx = true; break;
10          case 'd': v3tx = true; break;
11          case 'e': i1tx = true; break;
12          case 'f': i2tx = true; break;
13          case 'g': i3tx = true; break;
14          case 'h':
15              v1tx = v2tx = v3tx = i1tx =
16                  i2tx = i3tx = false;
17              break;
18      }
19      if (v1tx) { transmit_data("v1", v1); v1tx =
20                  false; }
21      if (v2tx) { transmit_data("v2", v2); v2tx =
22                  false; }
23      if (v3tx) { transmit_data("v3", v3); v3tx =
24                  false; }
25      if (i1tx) { transmit_data("i1", i1); i1tx =
26                  false; }
27      if (i2tx) { transmit_data("i2", i2); i2tx =
28                  false; }
29      if (i3tx) { transmit_data("i3", i3); i3tx =
30                  false; }
31      vTaskDelay(pdMS_TO_TICKS(10));
32  }
33  }

```

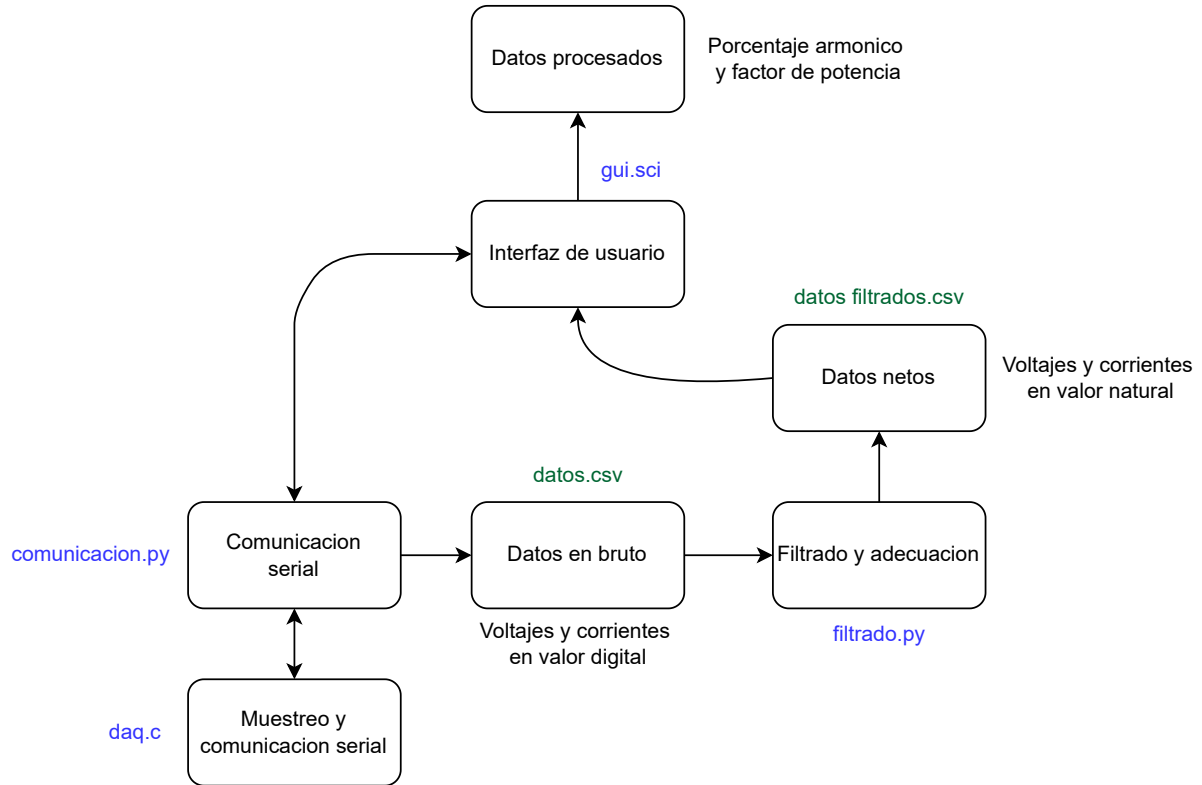


Figura 5.15: Diagrama de flujo de la interacción entre códigos.

5.9. Banco de pruebas y calibración

Se construyó un banco de pruebas que simula redes contaminadas con cargas no lineales utilizando focos LED ahorradores, evaluando el sistema en condiciones reales.

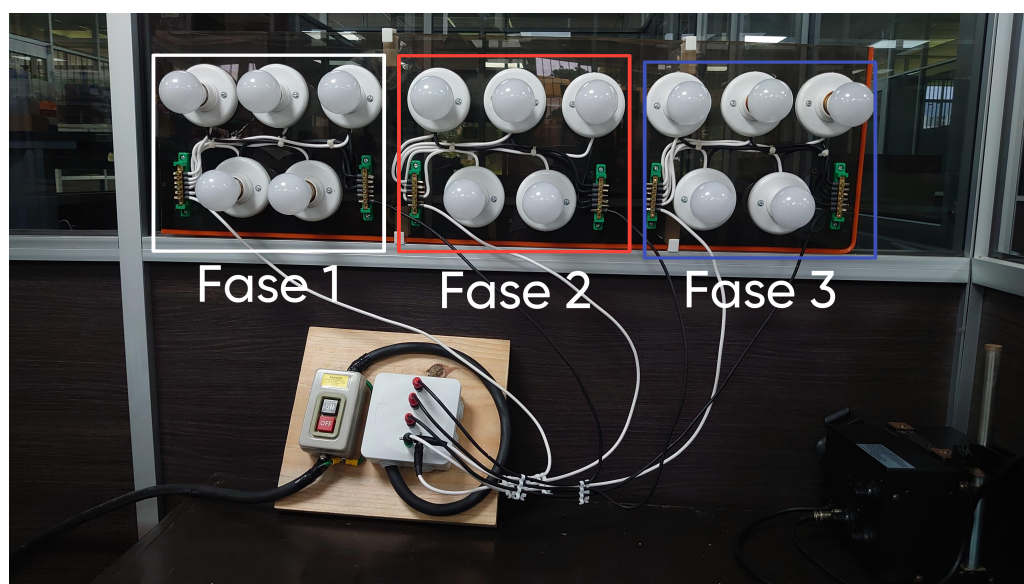


Figura 5.16: Banco de pruebas trifásico.

Capítulo 6

Resultados y Conclusiones

6.1. Medición en banco de pruebas

Se realizaron mediciones con el analizador de calidad de la energía Fluke 434 para establecer un marco de referencia. A continuación, se presentan las formas de onda y el contenido armónico registrados por dicho equipo.



Figura 6.1: Medición de corrientes trifásicas en la red de pruebas con el Fluke 434.



Figura 6.2: Formas de onda de corriente capturadas por el Fluke 434.

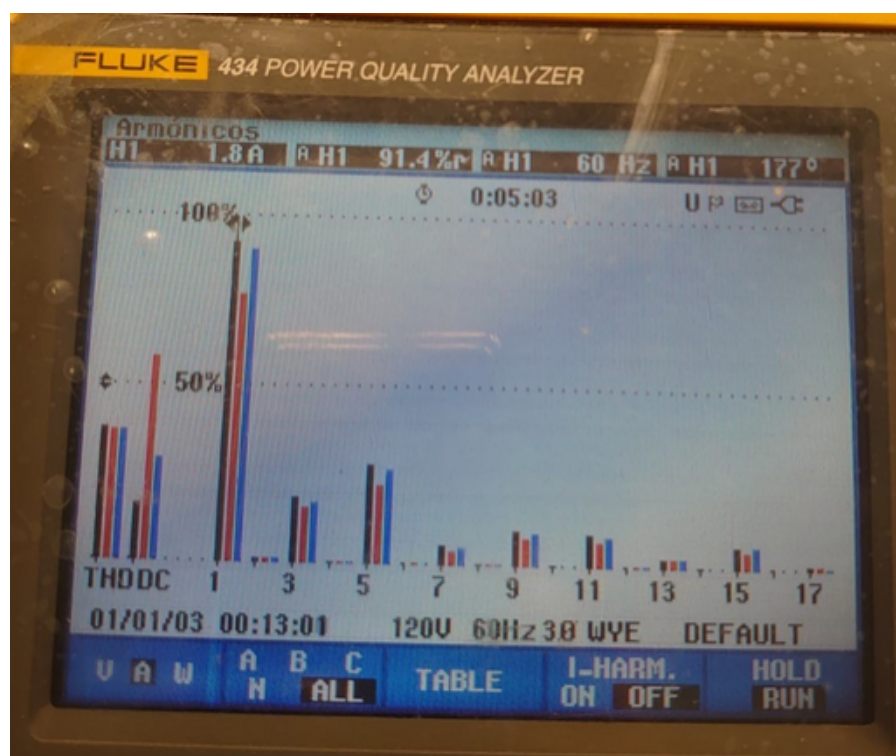


Figura 6.3: Contenido armónico total registrado por el Fluke 434.

Amp	A	B	C
THD%	30.9	31.2	31.4
H3%	16.3	12.9	15.2
H5%	20.8	18.8	21.2
H7%	2.8	2.2	1.0
H9%	10.8	9.4	10.5
H11%	4.7	4.6	4.3
H13%	4.8	3.8	4.7
H15%	4.3	4.2	4.2

Figura 6.4: Distribución de armónicos individuales obtenida con el Fluke 434.

	A	B	C
kW	0.2	0.3	0.3
kVA	0.3	0.4	0.3
kVAR	÷ 0.1	÷ 0.2	÷ 0.2
PF	0.88	0.74	0.86
DPF	0.98	0.99	0.99
A rms	2	3	2
	A	B	C
V rms	131.9	132.1	131.2

Figura 6.5: Factor de potencia por fase medido con el Fluke 434.

Posteriormente, se aplicó el dispositivo desarrollado sobre la misma red de pruebas para comparar resultados.

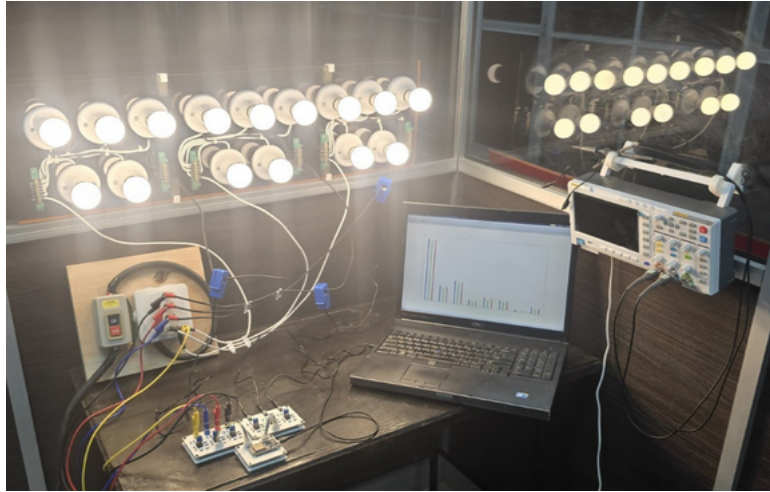


Figura 6.6: Medición de corrientes trifásicas con el prototipo desarrollado.

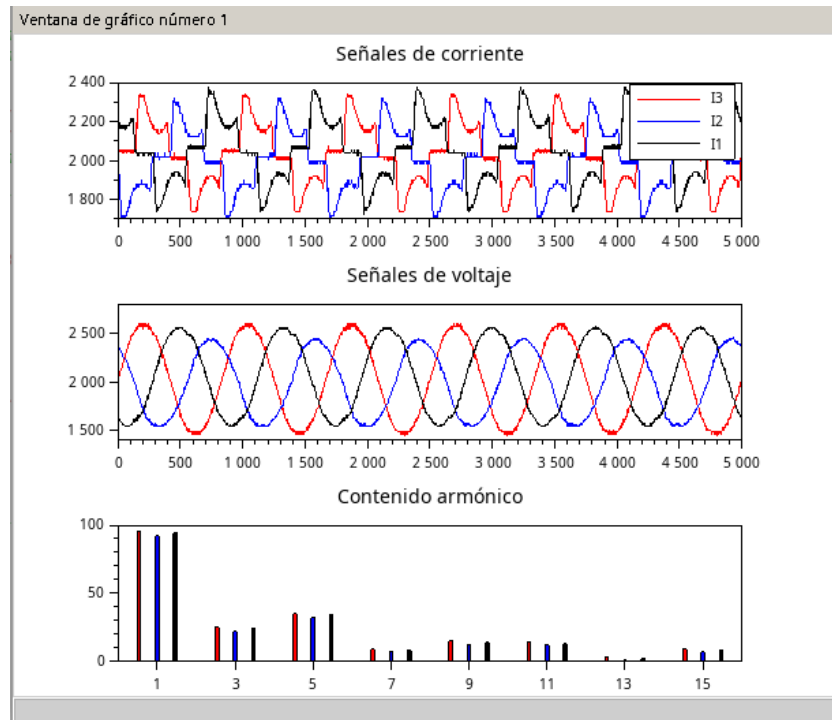


Figura 6.7: Interfaz gráfica de usuario (GUI) en Scilab mostrando resultados en tiempo real.

En la Tabla 6.1 se presentan los valores de THD, armónicos individuales y factor de potencia obtenidos mediante el sistema desarrollado. Se observa un error relativo promedio del 5 % respecto a las mediciones del Fluke 434, lo cual es aceptable para un prototipo educativo.

Parámetro	Fase A	Fase B	Fase C
THD (%)	29.44	29.64	29.87
H3 (%)	15.51	12.25	14.46
H5 (%)	19.08	17.86	20.14
H7 (%)	1.98	2.09	0.95
H9 (%)	10.34	8.93	10.02
H11 (%)	4.53	4.37	4.11
H13 (%)	4.64	3.61	4.46
H15 (%)	4.11	3.99	3.99
FP	0.84	0.70	0.82

Tabla 6.1: Valores de distorsión armónica y factor de potencia obtenidos con el prototipo.

6.2. Conclusiones

El dispositivo desarrollado cumple con el objetivo principal de adquirir señales de voltaje y corriente en un sistema trifásico, permitiendo el análisis de armónicos y cálculo del factor de potencia. Sin embargo, se requiere una calibración con un instrumento patrón certificado para mejorar la precisión de las mediciones.

Su diseño modular facilita la mejora de prestaciones mediante la sustitución de módulos individuales (acondicionamiento, DAQ, procesamiento). Aunque no sustituye a equipos profesionales de análisis de calidad de la energía, ofrece una herramienta valiosa en el ámbito educativo y de formación técnica, donde los requisitos normativos no son estrictos.

Este prototipo representa un primer acercamiento práctico al estudio de la calidad de la energía eléctrica, promoviendo el aprendizaje activo y la comprensión de fenómenos como la distorsión armónica y el impacto de cargas no lineales en redes trifásicas.

Capítulo 7

Trabajos Futuros

Dada la alta integración de los componentes de software —Python, Scilab y Linux—, se propone migrar la sección digital del proyecto a un sistema embebido, como la Raspberry Pi o sus equivalentes, con el fin de obtener un dispositivo portátil y autónomo.

Se sugieren mejoras en el módulo de acondicionamiento de señales de voltaje, ya que actualmente cuenta únicamente con protección por alta impedancia. Para mayor seguridad y aislamiento, se recomienda incorporar optoaisladores.

Dado que el proyecto consideró exclusivamente cargas no lineales y, por ende, se asumió un factor de potencia de desplazamiento igual a 1, se propone adaptar el código para calcular el desfase real entre las señales de corriente y voltaje.

Asimismo, se sugiere implementar un script en Python que genere reportes en Formato de Documento Portatil (PDF) a partir de los datos capturados por la GUI.

Finalmente, se recomienda optimizar la comunicación entre la DAQ y la GUI para soportar procesamiento en tiempo real.

Bibliografía

- [1] JM Industrial Technology SA de CV. Tarjetas de adquisición de datos, 2014.
- [2] Inc. The MathWorks. Matlab, 2022.
- [3] John W. Eaton. Gnu octave, 2020.
- [4] Dassault Systèmes. Scilab, 2023.
- [5] Eric W Weisstein. Discrete fourier transform. *<https://mathworld.wolfram.com/>*, 2002.
- [6] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.
- [7] Stephen J. Chapman. *Fundamentos de Máquinas Eléctricas*. McGraw-Hill, México, 5 edition, 2011. Traducción de la 5ª edición en inglés.
- [8] Theodore Wildi. *Máquinas Eléctricas y Sistemas de Potencia*. Pearson Educación, México, 6 edition, 2006.
- [9] James P. Dugan and Robert L. Stein. *Power Quality: Problems and Mitigation Techniques*. Wiley-IEEE Press, Hoboken, NJ, 2014.
- [10] Paul Horowitz and Winfield Hill. *El Arte de la Electrónica*. Cambridge University Press, Nueva York, 3 edition, 2015. Traducción de la 3ª edición en inglés.
- [11] International Electrotechnical Commission. Shunt power capacitors of the self-healing type for a.c. systems having a rated voltage up to and including 1 000 v – part 1: General – performance, testing and rating – safety requirements – guide for installation and operation, 2014.

- [12] A Rash. Harmonics-what are they, how to measure them and how to solve the problem (in connection with standards iee 1159-1995 and iee 519-1992). In *Proceedings of 19th convention of electrical and electronics engineers in Israel*, pages 83–86. IEEE, 1996.
- [13] European Committee for Electrotechnical Standardization. Industrial a.c. networks affected by harmonics – application of filters and shunt capacitors, 1997.
- [14] Amir Broshi. Monitoring power quality beyond en 50160 and iec 61000-4-30. In *2007 9th International Conference on Electrical Power Quality and Utilisation*, pages 1–6. IEEE, 2007.
- [15] Andres E Legarreta, Javier H Figueroa, and Julio A Bortolin. An iec 61000-4-30 class a—power quality monitor: Development and performance analysis. In *11th International Conference on Electrical Power Quality and Utilisation*, pages 1–6. IEEE, 2011.
- [16] Milad Dalali and Alireza Jalilian. Indices for measurement of harmonic distortion in power systems according to iec 61000-4-7 standard. *IET Generation, Transmission & Distribution*, 9(14):1903–1912, 2015.
- [17] Murat SİLSÜPÜR and Belgin Emre TÜRKAY. Flicker source detection methods based on iec 61000-4-15 and signal processing techniques—a review. *Balkan Journal of Electrical and Computer Engineering*, 3(2), 2015.
- [18] IEEE. Ieee recommended practice and requirements for harmonic control in electric power systems, 2022.
- [19] Christopher K Duffey and Ray P Stratford. Update of harmonic standard iee-519: Ieee recommended practices and requirements for harmonic control in electric power systems. *IEEE Transactions on Industry Applications*, 25(6):1025–1034, 1989.
- [20] Microchip Technology Inc. Current sensing circuit concepts and fundamentals, 2023. Accedido: 2025.
- [21] Embedded Lab. Safe current measurement with microcontrollers, 2024. Accedido: 2025.
- [22] Dewesoft. Current sensors: Hall effect, ct, rogowski coils, 2025. Accedido: 2025.
- [23] Enercorp Instruments. Non-invasive vs invasive current measurement, 2023. Accedido: 2025.

-
- [24] R. S. Popović. *Hall Effect Devices*. Taylor Francis, 2nd edition, 2004.
- [25] IEEE. IEEE standard requirements for instrument transformers. Technical Report IEEE Std C57.13-2016, IEEE, 2016.
- [26] W. Rogowski and W. Steinhaus. Die messung der magnetischen spannung. *Archiv für Elektrotechnik*, 1(4):141–150, 1912.
- [27] Electronics Stack Exchange. Comparison of current sensors: Ct, hall, rogowski, 2022. Accedido: 2025.
- [28] Honeywell International. Magnetic field sensitivity in hall effect sensors. Technical report, Honeywell Sensing and Control, 2019. Application Note AN218.
- [29] W. F. Ray and C. R. Hewson. High performance rogowski current transducers. *IEE Conference Publication*, 1:196–199, 2008.
- [30] C. R. Hewson. *Practical Rogowski Coil Design and Applications*. Pembridge Press, 2005.
- [31] EDN Network. Signal conditioning for adc inputs, 2023. Accedido: 2025.
- [32] Microchip Technology Inc. Signal conditioning for sensor interfaces, 2024. Accedido: 2025.
- [33] EDN Network. Preparing high-voltage signals for adc, 2022. Accedido: 2025.
- [34] Analog Devices. Signal conditioning for current sensors, 2023. Accedido: 2025.
- [35] Microchip Technology Inc. Maximizing adc performance with signal conditioning, 2025. Accedido: 2025.
- [36] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 2nd edition, 1999.
- [37] Frederic J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.
- [38] IEEE. Ieee recommended practice and requirements for harmonic control in electric power systems. Technical Report IEEE Std 519-2022, IEEE, 2022.
- [39] Utmel. Comparison: Rp2040, atmega328p, esp32, stm32, 2024. Accedido: 2025.

- [40] Maker Guides. Esp32 adc performance and oversampling, 2025. Accedido: 2025.
- [41] MathWorks. Matlab pricing and licensing, 2025. Accedido: 2025.
- [42] Educba. Scilab vs octave vs matlab: Performance comparison, 2023. Accedido: 2025.
- [43] EDN Network. Open-source alternatives to matlab: Octave and scilab, 2024. Accedido: 2025.
- [44] Young Professionals IEEE. Open-source tools for power quality analysis, 2025. Accedido: 2025.