



Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela Profesional de Ingeniería Electrónica

**Desarrollo de un prototipo para el monitoreo en la nube de la condición
de trabajo de un sistema de bombas centrífugas para unidades mineras
de media y baja producción con muestreo menor a 10 ms**

Tesis presentada por:

Velasquez Portilla, Jose Rodrigo

ORCID: 0009-0002-0748-9059

para optar el Título Profesional de Ingeniero Electrónico con especialidad en
Automatización y Control

Asesor (a):

Dr. Sulla Torres, Raúl Ricardo

ORCID: 0000-0002-4435-3575

Arequipa - Perú

2026

UNIVERSIDAD CATÓLICA DE SANTA MARÍA
INGENIERIA ELECTRONICA
CON ESPECIALIDAD EN AUTOMATIZACIÓN Y CONTROL
TITULACIÓN CON TESIS
DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 18 de Noviembre del 2025

Dictamen: 011424-C-EPTE-2025

Visto el borrador del expediente 011424, presentado por:

2015110391 - VELASQUEZ PORTILLA JOSE RODRIGO

Titulador:

**DESARROLLO DE UN PROTOTIPO PARA EL MONITOREO EN LA NUBE DE LA CONDICIÓN DE
TRABAJO DE UN SISTEMA DE BOMBAS CENTRÍFUGAS PARA UNIDADES MINERAS DE MEDIA Y
BAJA PRODUCCIÓN CON MUESTREO MENOR A 10 MS**

Nuestro dictamen es:

APROBADO

Titulo Profesional/Titulo de Segunda Especialidad/Grado Académico a optar:

**INGENIERO ELECTRONICO CON ESPECIALIDAD EN
AUTOMATIZACIÓN Y CONTROL**

**29715414 - COAGUILA GOMEZ RONALD PERCING
DICTAMINADOR**



**29329443 - VALDIVIESO HERRERA DIANA ISABEL
DICTAMINADOR**



**29565731 - COPA PINEDA JUAN CARLOS
DICTAMINADOR**



Desarrollo de un prototipo para el monitoreo en la nube de la condición de trabajo de un sistema de bombas centrífugas para unidades mineras de media y baja producción con muestreo menor a 10 ms

INFORME DE ORIGINALIDAD

15%	16%	1%	14%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	Submitted to Universidad Católica de Santa María	12%
	Trabajo del estudiante	
2	tesis.ucsm.edu.pe	3%
	Fuente de Internet	
3	git.xil.se	1%
	Fuente de Internet	

Excluir citas

Excluir bibliografía

Apagado

Apagado

Excluir coincidencias

< 1%

DEDICATORIA

Dedicada la presente tesis a Dios, quien nunca me abandonó que por más obstáculos que tuve en mi vida me ayudó a superarlo porque sé que siempre estará conmigo donde vaya guiando y protegiendo.

A mi padre por sus constantes consejos y experiencias compartidas, donde me enseñó a que los sueños y metas se cumplen en base al esfuerzo y dedicación y su ejemplo de salir adelante.

A mi madre por su cariño y paciencia por su apoyo incondicional por sus consejos y experiencias que me ayudaron a salir adelante en esta etapa importante de mi vida.

Tengan por seguro que no los defraudaré, haré que siempre ustedes estén orgullosos de mí dando lo mejor de mi persona.

AGRADECIMIENTOS

*Agradezco a Dios por iluminarme en los momentos
difíciles de mi vida, por darme la paciencia,
perseverancia y sabiduría para no rendirme.*

*Agradezco a mi familia por el apoyo incondicional y
darme las fuerzas necesarias para terminar este
proyecto, por inculcar en mí el ejemplo del esfuerzo y
que a la vez me enseñaron que el mejor conocimiento
que se puede tener es el que se aprende por sí mismo.*

*A mi tía Rosita y Andrés
quienes de manera incondicional siempre estuvieron en
mi vida, quienes me han apoyado con sus consejos por
su cariño y por acompañarme en esta etapa de dar este
gran paso importante para mi vida profesional.*

RESUMEN

La presente tesis aborda el diseño e implementación de un prototipo IoT en la nube, de bajo costo, orientado al monitoreo de los principales parámetros operativos de un sistema de bombeo centrífugo, aplicado a unidades mineras de media y baja producción. El sistema desarrollado permite analizar el comportamiento operativo del equipo y simular condiciones similares a las de bombas de mayor potencia, contribuyendo a la optimización del rendimiento y a la prevención de fallas.

El prototipo integra la medición de variables críticas como temperatura, presión, caudal, voltaje, corriente y vibración, esta última con un tiempo de muestreo de 60 μ s, lo que posibilita una evaluación detallada del estado del sistema. Para la adquisición de datos, se instalaron sensores en puntos estratégicos del sistema hidráulico y eléctrico, gestionados por un microcontrolador encargado de la transmisión de la información mediante el protocolo TCP. El sistema verifica periódicamente la conectividad con el servidor; ante una pérdida de comunicación, los datos son almacenados temporalmente en memoria interna, garantizando su integridad.

La infraestructura de almacenamiento se implementó utilizando SQL Server como sistema gestor de bases de datos, asegurando un manejo eficiente y seguro de la información recopilada. Asimismo, se desarrolló una interfaz de visualización en tiempo real mediante dashboards, implementada en Python, que facilita el análisis operativo y la toma de decisiones basada en datos. Los resultados obtenidos demuestran que el sistema de monitoreo propuesto constituye una solución viable y escalable para la supervisión de bombas centrífugas en el sector minero. Como trabajo futuro, se propone la incorporación de algoritmos de análisis predictivo orientados a anticipar fallas y optimizar las estrategias de mantenimiento.

Palabras clave: Protocolo TCP, SQL Server, Dashboards.

ABSTRACT

This thesis addresses the design and implementation of a low-cost cloud-based IoT prototype for monitoring the main operational parameters of a centrifugal pumping system, applied to medium- and low-production mining units. The developed system enables the analysis of the equipment's operational behavior and the simulation of conditions similar to those of higher-power pumps, contributing to performance optimization and failure prevention.

The prototype integrates the measurement of critical variables such as temperature, pressure, flow rate, voltage, current, and vibration, with the latter acquired at a sampling interval of 60 μ s, allowing a detailed assessment of the system's condition. For data acquisition, sensors were installed at strategic points within the hydraulic and electrical systems, managed by a microcontroller responsible for data transmission using the TCP protocol. The system periodically verifies connectivity with the server; in the event of communication loss, the data are temporarily stored in internal memory, ensuring data integrity.

The storage infrastructure was implemented using SQL Server as the database management system, ensuring secure and efficient handling of the collected information. In addition, a real-time visualization interface based on dashboards, developed in Python, was implemented to facilitate operational analysis and data-driven decision-making. The results demonstrate that the proposed monitoring system constitutes a viable and scalable solution for the supervision of centrifugal pumps in the mining sector. As future work, the incorporation of predictive analysis algorithms is proposed to anticipate failures and optimize maintenance strategies.

Keywords: TCP Protocol, SQL Server, Dashboards.

ÍNDICE

DEDICATORIA

AGRADECIMIENTOS

RESUMEN

ABSTRACT

INTRODUCCIÓN..... 1

CAPÍTULO I..... 4

PLANTEAMIENTO DEL PROYECTO DE INVESTIGACIÓN 4

1.1 Identificación del Problema 5

1.2 Descripción del problema 6

1.3 Objetivos..... 8

1.3.1 Objetivo principal..... 8

1.3.2 Objetivos específicos 9

1.4 Justificación 10

1.5 Alcances y limitaciones 11

1.5.1 Alcance..... 11

1.5.2 Limitaciones..... 12

1.6 Aporte 12

1.7 Procedimiento para realizar las mediciones..... 12

1.8 Estructura de tesis 13

CAPÍTULO 1	13
CAPÍTULO 2	13
CAPÍTULO 3	13
CAPÍTULO 4	13
CAPÍTULO 5	13
CAPÍTULO 6	14
CAPÍTULO 7	14
 CAPÍTULO II.....	 15
MARCO TEÓRICO	15
2.1 Equipo de monitoreo de un sistema de bombas de lodo.....	16
2.2 Instrumentación industrial	16
2.2.1 Definición de instrumentación	16
2.2.2 Características de los instrumentos	17
2.2.3 Clasificación de los instrumentos	18
2.2.4 Medidores de flujo	20
A. Medidores de caudal másico	21
2.2.5 Medidores de temperatura.....	21
A. Escalas de temperatura.....	21
B. Dispositivos electrónicos de medición de temperatura	22
C. Sensor detector resistivo de temperatura (RTD).....	22
D. Características de funcionamiento del sensor RTD.	23

2.3 Medidores de vibración	25
2.4 Tipos de sensores de vibración	25
2.4.1 Acelerómetros:	25
2.4.2 Sensores de velocidad:	25
2.4.3 Sensores de proximidad:	25
2.5 Sensor de presión	25
2.6 Revisión de antecedentes	26
2.6.1 Primer antecedente:	26
2.6.2 Segundo antecedente:	26
2.6.3 Tercer antecedente:	27
2.6.4 Cuarto antecedente:	28
CAPÍTULO III	29
DISEÑO DE HARDWARE	29
3. Análisis Comparativo de Selección de Componentes Electrónicos	30
a) Microcontrolador TMS570LS (Hércules RM46L852).....	33
b) Sensor de corriente ACS756-050A	33
c) Sensor de vibración PC420VR-05.....	33
d) Sensor de presión HK3025.....	34
e) Sensor de caudal TF-S201	34
3.1 Diagrama de bloques general.....	35
3.1.1 Proceso de Modelado Electrónico del Prototipo.....	36

3.1.2 Metodología de modelado adoptada	37
3.1.3 Modelos matemáticos de cada bloque electrónico.....	37
3.1.4 Descripción del proceso de modelado por etapa.....	40
a) Modelado del sensor de voltaje (circuito suministro $V_{cc}/2$ + amplificador de diferencia).....	40
b) Modelado del sensor de corriente ACS756-050A (efecto Hall)	40
c) Modelado del sensor de vibración PC420VR-05 (señal 4-20 mA)	41
d) Modelado del análisis espectral (FFT)	41
3.2 Diagrama de bloques específicos.....	41
3.2.1 Sensor de voltaje	41
A. Suministro de voltaje de salida	42
B. Amplificador de diferencia	43
3.2.2 Diagrama de bloques específicos	43
3.2.3 Diseño de sensor de voltaje.....	44
3.2.4 Análisis del circuito diseñado	47
A. Análisis de suministro de voltaje de salida.....	47
B. Análisis del amplificador de diferencia.....	50
3.3 Sensor de corriente.....	51
3.3.1 Diagrama de bloque específico	52
3.3.2 Selección de sensor de corriente	53
A. Sensor ACS756-050 A.....	53
3.3.3 Análisis del circuito seleccionado.....	55

3.4	Acondicionador de RTD.....	60
3.4.1	Diseño de circuito de acondicionador de RTD	60
3.4.2	Análisis de circuito diseñado	61
3.5	Sensor de vibración.....	64
3.5.1	Selección de sensor de vibración	65
A.	Sensor PC420VR-05.....	65
3.5.2	Circuito típico	66
3.5.3	Cálculo del valor RMS.....	67
A.	Adquisición de muestras:.....	67
B.	Selección de ventana temporal:.....	67
C.	Cálculo del valor cuadrático:	68
D.	Promedio de valores cuadráticos	68
E.	Raíz cuadrada del promedio:.....	¡Error! Marcador no definido.
3.5.4	Interpretación de la señal de 4-20 MA.....	68
3.6	Sensor de presión	69
3.6.1	Selección de sensor de presión.....	69
3.6.2	Especificaciones técnicas	69
3.6.3	Circuito típico	69
3.7	Sensor de caudal	70
3.7.1	Selección de sensor de caudal	70
3.7.2	Especificaciones técnicas	71
3.8	Adaptador de 4 A 20 MA	72

3.8.1 Diagrama de bloques específico.....	72
3.8.2 Selección de circuito de 4 A 20 Ma	73
A. Módulo de corriente SEN0262	73
3.9 Relay de salida	74
3.9.1 Diagrama de bloques específico.....	75
3.9.2 Selección de relay de salida	75
A. Módulo de cuatro canales relé	75
3.10 Módulo de wifi.....	76
3.10.1 Diagrama de bloques específicos	77
3.10.2 Elección de módulo de WI-FI	77
A. Módulo de WI-FI ESP32	77
3.11 Pantalla LCD.....	78
3.11.1 Selección de pantalla LCD 2X16.....	78
3.12 Microcontrolador LAUNCHXL2-RM46 - HERCULES.....	79
3.13 Bomba de agua.....	80
3.14 Cálculo de grosor de pistas	81
3.14.1 Fórmulas y consideraciones:	81
CAPÍTULO IV	83
DISEÑO DE SOFTWARE.....	83
4.1 Diagrama de flujo	84
4.1.1 Diagrama de flujo principal	84

4.1.2	Diagrama de flujo menú de mediciones eléctricas.....	87
4.1.3	Diagrama de flujo de medición de análisis en dominio de la frecuencia...	88
4.1.4	Diagrama de flujo de menú de WIFI	89
4.1.5	Diagrama de flujo menú de configuración i/o digital y análogas	90
4.2	Programas	91
4.2.1	Sistema de adquisición y monitoreo en tiempo real.....	91
4.2.2	Extracción de datos de señales desde sql server	102
4.2.3	Procesamiento y análisis espectral de señales de vibración em bombas centrífugas.....	104
4.2.4	Procesamiento y análisis espectral de señales de voltaje y corriente en tiempo discreto	107
4.2.5	Análisis de vibraciones en el dominio del tiempo y frecuencia mediante FFT	111
4.2.6	Extracción y análisis temporal de parámetros de medición desde sql server	115
4.2.7	Visualización de parámetros físicos, eléctricos y mecánicos de operación del prototipo.....	117
4.2.8	Análisis temporal de señales de voltaje, corriente y vibración adquiridas desde sql server.....	120
4.2.9	Sistema de adquisición para la detección de eventos de sobrecorriente en prototipo de monitoreo	124
CAPITULO V		129

IMPLEMENTACION DEL EQUIPO.....	129
5.1 Diseño de placa.....	130
5.2 Diseño de prototipo caja fibra de vidrio	132
5.3 Ajustes de parámetros de tarjetas.....	133
5.3.1 Microcontrolador LAUNCHXL2-RM46 - HERCULES	133
5.3.2 Módulo de WI-FI ESP32	134
5.3.3 Módulo pantalla LCD 2X16.....	134
5.4. Implementación de prototipo de monitoreo:.....	135
5.5 Implementación de un sistema piloto	138
CAPITULO VI	141
PRUEBAS Y RESULTADOS	141
6.1 Objetivos de la prueba:	142
6.2 Equipamiento requerido.....	142
6.2.1 Características del equipo	142
6.2.3 Especificaciones del equipo	143
6.2.4 Especificaciones del sistema de medición y procesamiento	143
6.2.5 Especificaciones del módulo de visualización.....	143
6.2.6 Hoja de Especificaciones Técnicas del Prototipo de Monitoreo.....	144
6.3 Especificaciones de parámetros de muestreo	146
6.3.1 Análisis de Frecuencia: Bomba en Buen Estado vs. Bomba con Anomalías.....	149

6.3.1.1	Parámetros de adquisición para el análisis espectral.....	149
6.3.1.2	Hoja de análisis comparativo de frecuencias	150
6.3.1.3	Interpretación técnica de los resultados espectrales.....	152
6.4	Matriz de pruebas.....	153
6.5	Inicio de pruebas y resultados:.....	156
6.6	Análisis económico.....	173
6.7	Sistema de monitoreo para unidades mineras grandes	175
6.7.1	Synertrex – intellipump (empresa – global).....	175
6.7.2	Conclusión:	177
CAPÍTULO VII.....		178
CONCLUSIONES Y RECOMENDACIONES		178
CONCLUSIONES		179
RECOMENDACIONES.....		181
REFERENCIAS.....		182
ANEXOS		183

ÍNDICE DE TABLAS

Tabla 1	Sistema de Unidades Físicas Electrónicas.....	3
Tabla 2	Características de Materiales de RTD	23
Tabla 3	Valores de Resistencia y Temperatura	24
Tabla 4	Corriente - Análisis comparativo de selección de componentes electrónicos. 30	

Tabla 5 Vibración - Análisis comparativo de selección de componentes electrónicos .	31
Tabla 6 Presión - Análisis comparativo de selección de componentes electrónicos.....	31
Tabla 7 Caudal - Análisis comparativo de selección de componentes electrónicos.....	32
Tabla 8 Microcontrolador - Análisis comparativo de selección de componentes electrónicos.....	32
Tabla 9 Resumen del proceso de modelado electrónico por bloque del prototipo.....	37
Tabla 10 Especificaciones Técnicas del Sensor	57
Tabla 11 Voltaje de Salida en Función a la Variación de la Resistencia	63
Tabla 12 Hoja de especificaciones técnicas del prototipo de monitoreo de bombas centrífugas	144
Tabla 13 Análisis de frecuencia comparativo: bomba en buen estado*	150
Tabla 14 Análisis de frecuencia comparativo: bomba con anomalía*	151
Tabla 15 Matriz de Pruebas	154
Tabla 16 Análisis Económico del Equipo	173
Tabla 17 Análisis Económico de los Componentes Aquiridos	173
Tabla 18 Análisis de Costo Total del Prototipo de Monitoreo	175
Tabla 19 Análisis de Costo de Prototipo Actual.....	176
Tabla 20 Análisis de Costo de Comparativo de Prototipo Actual VS Sistema de Monitoreo	177

ÍNDICE DE FIGURAS

Figura 1 Diagrama de bloques general.....	35
Figura 2 Suministro de Voltaje de Salida $V_{cc}/2$	42
Figura 3 Circuito de resta de Voltaje de Salida $V_{cc}/2$	43
Figura 4 Diagrama de Bloques Específico del Suministro del voltaje de Salida	43
Figura 5 Onda sinusoidal de Voltaje Pico	45

Figura 6	Rango de diseño ADC microcontrolador.....	46
Figura 7	Diagrama de Suministro de voltaje de salida.....	47
Figura 8	Rango de diseño ADC microcontrolador.....	48
Figura 9	Divisor de Voltaje	48
Figura 10	Circuito de resta de Voltaje de Salida Vref.....	50
Figura 11	Diagrama de Bloques del sensor de temperatura	52
Figura 12	Efecto Hold del Sensor ACS 756-050 A	54
Figura 13	Comportamiento del Sensor AC 756-050 ^a	55
Figura 14	Divisor de Voltaje	58
Figura 15	Sensor de Corriente con Circuito Reductor	59
Figura 16	Circuito acondicionador de RTD	60
Figura 17	Circuito acondicionador de RTD	61
Figura 18	Circuito típico de sensor PC420VR-05.....	66
Figura 19	Circuito típico de sensor HK3025.....	70
Figura 20	Diagrama de Bloques Adaptador de 4 a 20mA.....	72
Figura 21	Descripción de la salida de voltaje sobre salida.....	74
Figura 22	Comportamiento Relay	75
Figura 23	Módulo Wi-Fi ESP32	77
Figura 24	Configuración de Entradas y Salidas	79
Figura 25	Entradas y salidas del microcontrolador	80
Figura 26	Diagrama de flujo principal	85
Figura 27	Diagrama de flujo, menú de mediciones eléctricas.....	87
Figura 28	Diagrama de flujo de medición de análisis en dominio de la frecuencia.....	88
Figura 29	Diagrama de flujo de menú de Wifi.....	89
Figura 30	Diagrama de flujo de configuración I/O Digital y Analógicas	90
Figura 31	Diagrama de flujo de extracción de datos desde SQL Server	102

Figura 32 Diseño de placa	130
Figura 33 Circuito impreso en pistas.....	131
Figura 34 Caja con protección fibra de vidrio.....	132
Figura 35 Componentes internos prototipo	133
Figura 36 Microcontrolador Hércules	133
Figura 37 Módulo de Wi-fi ESP32.....	134
Figura 38 Pantalla LCD 2X16.....	134
Figura 39 Vista superior de prototipo.....	135
Figura 40 Vista derecha de prototipo	135
Figura 41 Vista izquierda de prototipo.....	136
Figura 42 Vista derecha abajo de prototipo.....	136
Figura 43 Vista inferior de prototipo.....	137
Figura 44 Sistema Piloto	138
Figura 45 Distribución de Sensores.....	139
Figura 46 Sistema completo para realizar pruebas.....	140
Figura 47 Forma de onda – Análisis Temporal de vibración	146
Figura 48 Forma de onda – Análisis Temporal de vibración	147
Figura 49 Forma de onda – Análisis Espectral (FFT) Armónicos	148
Figura 50 Verificación de sensores para puesta en marcha.....	156
Figura 51 Dirección IP Privada	157
Figura 52 Forma de Onda - Prototipo conectado	158
Figura 53 Forma de onda – Picos de arranque	159
Figura 54 Verificación de formas de onda en Dashboards	160
Figura 55 Forma de onda – Análisis espectral (FFT).....	161
Figura 56 Desalineamiento de impulsor.....	162
Figura 57 Flujo Reducido / Bloqueo parcial	163

Figura 58 Cavitación	164
Figura 59 Falla de sensor.....	165
Figura 60 Forma de onda prototipo con sistema em operación – Base de datos.....	166
Figura 61 Visualización de parámetros en el prototipo.....	170

ÍNDICE DE ANEXOS

ANEXO A: DATASHEET SENSOR DE VIBRACION WILCOXON	183
ANEXO B: DATASHEET MOTOR	185
ANEXO C: PROGRAMA - CÓDIGO PROGRAMA PRINCIPAL	192
ANEXO D: PROGRAMA - FUNCIONES ANEXAS:	202
ANEXO E: PROGRAMA - FUNCIONES DE RELOJ	207
ANEXO F: PROGRAMA - CONFGIURACIÓN DE RELOJ	211
ANEXO G: PROGRAMA - MOTOR DE CÁLCULO ELÉCTRICO	212

INTRODUCCIÓN

Las bombas centrífugas de lodo son componentes fundamentales en una amplia gama de aplicaciones industriales, especialmente en el sector minero, donde operan de manera continua durante toda la semana. Su función es vital para el transporte de mezclas abrasivas, por lo que están sometidas a condiciones de trabajo exigentes que aceleran el desgaste de sus componentes. En este contexto, se vuelve imprescindible realizar mantenimientos periódicos que permitan preservar su operatividad, evitar paradas inesperadas y extender la vida útil de los elementos críticos, comúnmente conocidos como elementos de sacrificio. (Karassik et al., 2008; Warman, 2015)

No obstante, la falta de información en tiempo real sobre el estado operativo de estas bombas dificulta una gestión eficiente del mantenimiento. Esta carencia suele conducir a intervenciones tardías o inadecuadas, lo que genera pérdidas económicas considerables a lo largo del tiempo. Además, la ausencia de herramientas de diagnóstico confiables limita la capacidad para detectar fallas incipientes o el desgaste progresivo de los componentes rotativos, comprometiendo así la eficiencia del sistema. Entre los principales indicadores de falla se encuentran el aumento de la temperatura, vibraciones anómalas, variaciones inesperadas en la corriente y el voltaje, así como una reducción en la presión y el flujo del sistema. (Mobley, 2002; Randall, 2011)

Frente a este escenario, la presente tesis se enfoca en el desarrollo de un prototipo para el monitoreo en la nube de la condición de trabajo de un sistema de bombas centrífugas, destinado a unidades mineras de media y baja producción.

El sistema propuesto integra un microcontrolador que verifica de manera continua la conectividad mediante el protocolo de transmisión TCP, permitiendo la publicación de datos hacia un bróker. En caso de pérdida de conexión, el sistema almacena la información localmente para asegurar la integridad de los registros.

Asimismo, se emplea SQL Server como sistema de gestión de bases de datos y la plataforma Python para el diseño de una interfaz gráfica que facilita la visualización de datos en tiempo real. Esta solución tecnológica no solo permite monitorear parámetros clave como temperatura, presión, flujo, voltaje, corriente y vibración, sino que también brinda al usuario la capacidad de identificar y diagnosticar fallas comunes como el desgaste, el desequilibrio, las holguras mecánicas, el desalineamiento y los fallos en los cojinetes. Cabe destacar que el monitoreo del motor eléctrico también representa un factor crucial para asegurar el correcto funcionamiento del sistema, ya que su rendimiento impacta directamente en la eficiencia general de las bombas centrífugas. (Gubbi et al., 2013; Al-Fuqaha et al., 2015)

Sistema de unidades físicas electrónicas

Apartado de sistemas de unidades de medida que se utilizará para describir y analizar los parámetros físicos involucrados en el estudio, con el fin de mantener la coherencia y estandarización en todos los cálculos y análisis relacionados.

Tabla 1

Sistema de Unidades Físicas Electrónicas

Variable Física	Símbolo	Sensor	Rango Físico	Unidad de Entrada (ADC/Hz/mA)	Unidad Física
Flujo de agua	Q	YF-S201	1 – 30 L/min	Frecuencia de pulsos (Hz)	L/min
Presión	P	HK3025	0 – 5 MPa	0.5 – 4.5 V (→ ADC 0–4095)	MPa
Temperatura	T	PT100	-50°C a 250°C (estimado)	Voltaje (→ ADC 0–4095)	°C
Vibración	v	Wilcoxon PC420VR-05	0 – 12 mm/s	Corriente 4–20 mA (→ ADC)	mm/s
Tensión RMS	V _{rms}	Medida interna	Dependiente del circuito	ADC (dif. entre canales)	Volts (V)
Corriente RMS	I _{rms}	Medida interna	Dependiente del sensor	ADC - offset (1837)	Amperios (A)

Nota: Tabla de descripción de unidades físicas electrónicas.

El uso de este sistema permite una interpretación clara de los resultados obtenidos en el estudio del rendimiento hidráulico y energético de la bomba centrífuga desde un punto de vista de la ingeniería electrónica.



CAPÍTULO I

PLANTEAMIENTO DEL PROYECTO DE

INVESTIGACIÓN

Este capítulo presenta la identificación del problema, descripción del problema, justificación, objetivos, aportaciones y la estructura del proyecto de tesis; se exponen los elementos utilizados para analizar el problema, como también métodos aplicados; se indican los aportes de tesis y la estructura de trabajo.

1.1 Identificación del Problema

Las bombas centrífugas de lodo son componentes críticos en el tratamiento de aguas residuales y en diversos procesos industriales. No obstante, su funcionamiento óptimo puede verse comprometido por múltiples anomalías que afectan tanto la eficiencia operativa como la vida útil de los equipos. Este apartado se enfoca en la identificación de los problemas más comunes que pueden presentarse en estas bombas, así como en sus causas y posibles soluciones.

Uno de los inconvenientes más frecuentes es la baja presión o el caudal insuficiente, que puede deberse a obstrucciones en la línea de succión o en el impulsor, a una velocidad de bombeo inadecuada o al desgaste y erosión del impulsor. Para mitigar este problema, es necesario verificar y ajustar la velocidad de operación de la bomba, así como reemplazar las piezas dañadas o desgastadas.

El sobrecalentamiento de la bomba suele ser consecuencia de una refrigeración deficiente, una operación por encima de su capacidad nominal o la presencia de aire atrapado en el sistema. La solución requiere asegurar un flujo de refrigeración adecuado, revisar las especificaciones del fabricante y ajustar la operación de la bomba según las condiciones de trabajo.

Las vibraciones excesivas pueden ser indicativas de problemas como desbalance en el impulsor, montaje incorrecto de la bomba o fijaciones flojas. Para corregir este tipo de fallas, es fundamental revisar el balance del impulsor, garantizar una instalación adecuada y verificar que todas las fijaciones estén debidamente ajustadas.

Los ruidos inusuales pueden alertar sobre fenómenos de cavitación, generalmente provocados por una presión de succión insuficiente, o sobre el desgaste de los rodamientos. Para evitar daños mayores, es crucial comprobar y ajustar la presión de succión, así como inspeccionar y reemplazar los rodamientos si presentan signos de deterioro.

La identificación temprana y la solución eficaz de estos problemas son esenciales para mantener el rendimiento y la durabilidad de las bombas centrífugas de lodo. La implementación de un programa de mantenimiento preventivo, la capacitación del personal y el monitoreo constante de las condiciones operativas son estrategias clave para prevenir fallos y optimizar el funcionamiento de estos equipos críticos.

Una comprensión profunda de las posibles fallas operativas permite establecer un camino hacia una operación más eficiente, confiable y sostenible de las bombas centrífugas de lodo en entornos industriales exigentes.

1.2 Descripción del problema

La operación de las bombas centrífugas de lodo en entornos mineros de producción mediana y pequeña enfrenta una serie de desafíos. La falta de sistemas adecuados de software y hardware para el monitoreo de estas bombas puede agravar considerablemente dichos problemas, afectando tanto la eficiencia operativa como la seguridad. A continuación, se detallan los principales inconvenientes derivados de esta carencia.

Detección tardía de fallas:

Sin un sistema de monitoreo en tiempo real, fallas como la cavitación, el desgaste de componentes o las vibraciones anómalas suelen detectarse de manera tardía. La falta

de alerta temprana puede dar lugar a fallos catastróficos, que no solo interrumpen la operación, sino que también pueden causar daños graves al personal que labora.

Paradas inesperadas de producción:

La imposibilidad de identificar problemas potenciales con antelación conlleva a paradas inesperadas en la producción. Estas interrupciones no programadas afectan la continuidad operativa y pueden impactar significativamente la rentabilidad de la empresa.

Falta de análisis de datos:

Sin herramientas adecuadas para el análisis de datos, resulta difícil evaluar el rendimiento y la eficiencia de las bombas. Esta carencia de información limita la capacidad para identificar tendencias y oportunidades de mejora, lo que genera una operación ineficiente en la producción.

Planificación inadecuada del mantenimiento:

La falta de información precisa sobre el estado de las bombas dificulta la programación del mantenimiento preventivo. Esto puede acelerar el desgaste de los componentes, lo que incrementa los costos de reparación y reduce la vida útil de los equipos.

Problemas en la toma de decisiones:

La ausencia de datos confiables impide una gestión eficaz del sistema, obligando a los responsables a tomar decisiones basadas en suposiciones. Esta situación puede derivar en decisiones ineficaces que afecten negativamente la productividad y los costos operativos.

Aumento de costos operativos:

La ineficiencia derivada de la falta de monitoreo, junto con los tiempos de inactividad no programados, ocasiona un incremento significativo de los costos operativos. Las empresas pueden enfrentar gastos adicionales tanto en reparaciones como en compensación por la pérdida de producción.

Riesgos para la seguridad:

Finalmente, la ausencia de un sistema de monitoreo adecuado puede poner en peligro la seguridad de los trabajadores. La incapacidad para detectar condiciones operativas peligrosas aumenta el riesgo de accidentes laborales, poniendo en riesgo la vida y la integridad física del personal.

La implementación de un sistema de monitoreo de bajo costo en las bombas centrífugas es una estrategia eficaz para prevenir fallas y minimizar las paradas inesperadas en la producción. Mediante el uso de sensores de vibración, temperatura, presión, flujo, voltaje y corriente, integrados con plataformas IoT y microcontroladores, es posible obtener información en tiempo real sobre el estado de la bomba, lo que facilita la detección rápida de anomalías y la planificación de mantenimientos predictivos. Esta solución garantizará el máximo aprovechamiento del sistema, mejorando la confiabilidad de las bombas y evitando paradas inesperadas en la producción.

1.3 Objetivos

1.3.1 Objetivo principal

Implementar un prototipo IoT en la nube de bajo costo para el monitoreo de los principales parámetros de trabajo de un sistema de bomba centrífuga en unidades mineras

de media y baja producción, utilizando un sistema piloto que permita comprender el comportamiento del sistema y simular condiciones operativas similares a bombas de mayor potencia, mediante la medición de temperatura, presión, flujo, voltaje, corriente y la vibración con un muestreo menor a 10 ms (60us) con el fin de optimizar el rendimiento y prevenir fallos.

1.3.2 Objetivos específicos

Conocer las características y parámetros para desarrollar un prototipo de monitoreo para bombas centrífugas que realice mediciones de los parámetros clave temperatura, flujo, voltaje, presión, corriente y vibración (velocidad mm/s) que los almacene y transmita en tiempo real, permitiendo su visualización en un terminal remoto. Este sistema debe ser accesible y de bajo costo, proporcionando información sobre la condición de operación del sistema a vendedores y operadores.

Implementar un sistema piloto con una bomba centrífuga de 1 HP, tuberías de PVC y un tanque de almacenamiento que permiten la recirculación del flujo. Este montaje reproduce a escala reducida las condiciones de operación de bombas industriales que nos ayudarán a monitorear las condiciones de operación como voltaje, corriente, presión, flujo, temperatura y vibración.

Seleccionar sensores adecuados para medir temperatura, flujo, voltaje, presión, corriente y vibración, que permitan obtener los parámetros principales de trabajo con el propósito de mejorar la optimización de operación de la bomba, mediante el almacenamiento y visualización de datos recolectados que permitan al operador evitar paradas de producción inesperadas, reducir el consumo energético y minimizar pérdidas económicas.

Implementar una base de datos para almacenar la información recolectada por el prototipo de monitoreo de la bomba, con una interfaz web que registre y visualice los

parámetros principales. Utilizar el protocolo de transmisión para la transmisión de datos a través de comunicación inalámbrica IoT, integrando una plataforma que permita el análisis de datos por el vendedor recolectados por el prototipo de los parámetros principales.

Visualizar los datos de los parámetros principales en dashboards que permitan comprender y analizar los patrones de frecuencia, facilitando a los vendedores la identificación de posibles fallos, como averías, balanceos incorrectos, cavitación, consumo energético ineficiente, y anticipando el mantenimiento necesario para asegurar el óptimo funcionamiento del sistema de bombas centrífugas, similar al comportamiento de bombas a escala industrial.

1.4 Justificación

La implementación de un prototipo IoT de bajo costo para el monitoreo de bombas centrífugas en unidades mineras de media y baja producción está directamente alineada con las necesidades actuales del sector minero de optimizar el uso de sus recursos y mejorar la eficiencia de sus procesos. Este proyecto busca resolver problemas críticos como las paradas inesperadas de producción, los fallos prematuros en equipos y el consumo energético ineficiente, factores que tienen un impacto directo en la competitividad y la rentabilidad de las empresas mineras.

El desarrollo de este sistema responde al objetivo principal de implementar una solución tecnológica que permita la monitorización en tiempo real de los principales parámetros operativos (temperatura, presión, flujo, voltaje, corriente y vibración). Esta capacidad de monitoreo continuo proporciona una visión detallada del estado operativo de las bombas centrífugas, facilitando la detección temprana de fallos y permitiendo la planificación de un mantenimiento predictivo más eficiente.

Además, la transmisión de datos a través de comunicación inalámbrica IoT y su almacenamiento en la nube no solo facilita el acceso remoto a la información en tiempo real, sino que también mejora la toma de decisiones tanto por parte de los operadores como de los proveedores, quienes pueden analizar el rendimiento del sistema y ofrecer soporte técnico adecuado sin necesidad de estar en sitio. Esto podría asegurar una respuesta rápida ante cualquier anomalía, reduciendo el tiempo de inactividad y los costos asociados a fallos inesperados.

En resumen, el proyecto está justificado por su capacidad para:

- Optimizar el rendimiento de las bombas centrífugas.
- Prevenir paradas inesperadas mediante un sistema de monitoreo predictivo.
- Reducir el consumo energético y las pérdidas económicas asociadas.
- Ofrecer una solución accesible y de bajo costo para unidades mineras de media y baja producción, obteniendo datos y puedan mejorar la eficiencia operativa sin necesidad de inversiones elevadas.

Este enfoque integral y accesible proporciona un valor estratégico al sector minero, promoviendo una operación más confiable y eficiente, con menores costos operativos y mayores márgenes de producción.

1.5 Alcances y limitaciones

1.5.1 Alcance

El estudio presente se realizará para simular el funcionamiento de bombas centrífugas de lodo a una escala menor, se trata de un enfoque que incluye el diseño, desarrollo del prototipo, visualización de datos para el análisis de data por el vendedor y/o operadores para la optimización de la operación de las bombas centrífugas en un contexto minero. Este alcance asegura que el sistema no solo sea funcional, sino que

también brinde valor real a las operaciones mineras, mejorando la eficiencia, reduciendo costos y minimizando riesgos asociados a fallos operativos según su evaluación de los datos obtenidos.

1.5.2 Limitaciones

Estas bombas centrífugas de lodo son de diferente tamaño y son muy robustas, por lo que el estudio se realizará sobre una bomba centrífuga de agua de 1 HP para poder entender el comportamiento de las bombas de mayor capacidad que generalmente operan bajo condiciones de carga, flujo y presión diferentes.

1.6 Aporte

La implementación de un prototipo IoT de bajo costo para el monitoreo de bombas centrífugas en unidades mineras de media y baja producción representa un avance significativo en la optimización y gestión de recursos operativos. Este sistema proporciona múltiples aportes y beneficios que impactan directamente en la toma de decisiones respecto a la eficiencia y sostenibilidad de las operaciones mineras, reducción de costos operativos que resulta de la recopilación y análisis de datos en tiempo real.

1.7 Procedimiento para realizar las mediciones

- Verificar el montaje del sistema piloto, conexiones eléctricas y calibrar sensores.
- Configurar parámetros de muestreo y sincronizar reloj del sistema y base de datos; considerar que está dentro del programa seteado para su inicio.
- Probar la comunicación Wi-Fi y la conexión con dashboards.
- Poner en marcha la bomba bajo carga para simular condiciones reales.
- Capturar señales de voltaje, corriente, presión, flujo, temperatura y vibración.
- Visualizar resultados en dashboards e identificar picos de arranque y armónicos.

1.8 Estructura de tesis

La tesis está conformada por 7 capítulos organizados de la siguiente manera:

CAPÍTULO 1

En el primer capítulo se dará una introducción al problema que se tratará en las bombas centrífugas de lodo, dando a conocer la identificación del problema, descripción, justificación, objetivos, alcances y limitaciones, aportaciones y la estructura de la tesis.

CAPÍTULO 2

En el segundo capítulo se da a conocer lo que es un prototipo de monitoreo para un sistema de bombas centrífugas de lodo, su funcionamiento, también sus diferentes características técnicas del prototipo.

CAPÍTULO 3

En el tercer capítulo se presenta el esquema circuital, viendo las partes que dispone cada dispositivo, especificaciones, selección de componentes y cálculos para su respectivo diseño.

CAPÍTULO 4

En el cuarto capítulo se presentan diagramas de flujo del software que se implementará, con explicación sobre el diagrama de flujo detalladamente cada parte.

CAPÍTULO 5

En el quinto capítulo se presenta la implementación y pruebas realizadas del esquema circuital que conforma este prototipo de monitoreo de bombas centrífugas de lodo. Con la finalidad de tener el equipo terminado.

CAPÍTULO 6

En el sexto capítulo se muestran las pruebas realizadas con la bomba en vacío, donde se verá la respectiva data obtenida de la bomba, análisis, gráficas, visualización para llegar a un diagnóstico correcto.

CAPÍTULO 7

En el séptimo capítulo se presentan las recomendaciones y conclusiones que se realizarán.





CAPÍTULO II

MARCO TEÓRICO

En el presente capítulo se darán a conocer las características del equipo de monitoreo de un sistema de bomba de lodo, su funcionamiento y también se presentan las especificaciones técnicas del equipo.

2.1 Equipo de monitoreo de un sistema de bombas de lodo

Prototipo de monitoreo para sistemas de bomba centrífuga está diseñado para identificar los parámetros clave de operación y detectar fallas por los vendedores y/o operaciones. Esto se logra mediante la medición de sensores y actuadores, con el equipo conectándose a diferentes puntos de medición (motor, bomba, tubería) para recopilar datos sobre su estado operativo. Una vez instalado el equipo, procederá a realizar las mediciones de temperatura, voltaje, corriente, presión y caudal con lo que se podrá evaluar el estado de la bomba a corto, mediano y largo plazo.

Se establece el Bróker de transmisión TCP en un servidor que cuenta con SQL Server como base de datos y Python como software de interfaz. Python se conecta al Bróker mediando el protocolo de transmisión TCP, leyendo la data que es enviada por el equipo, luego almacena la información en la base de datos (SQL Server), para generar dashboards pantallas de visualización, utilizando las mismas librerías del Server SQL.

2.2 Instrumentación industrial

2.2.1 Definición de instrumentación

La instrumentación comprende los dispositivos y sistemas empleados para medir, registrar y controlar variables físicas dentro de un proceso, permitiendo así su monitoreo y optimización. En términos más específicos, la instrumentación transforma una variable medida en una forma adecuada para su registro, visualización o detección, un proceso que también se conoce como medición o señal medida.

En un sistema de instrumentación, el elemento fundamental es el sensor, que es un dispositivo físico basado en un fenómeno natural para captar la variable. Posteriormente, el transductor convierte esta variable de sensado en una señal detectada, que puede ser de origen eléctrico, mecánico, óptico, neumático, entre otros. Esta señal detectada es luego cuantificada, facilitando el análisis y la toma de decisiones en diversas aplicaciones industriales y científicas. (Bentley, 2005; Doebelin, 2011)

2.2.2 Características de los instrumentos

Conforme a las normas SAMA (Scientific Apparatus Makers Association), PMC 20, las características de mayor consideración para los instrumentos son:

A. Campo de medida o rango:

Conjunto de valores dentro de un rango con límite superior y límite inferior de medida, por lo cual el instrumento es capaz de operar de manera efectiva. Ejemplo, un termómetro con un rango de 0°C a 200°C.

B. Alcance

Es la diferencia entre el valor superior e inferior del campo de medida. Por ejemplo, el termómetro del spam será de 200°C.

C. Error

Es la diferencia que existiría entre el valor que el instrumento indique que tenga la variable de proceso y el valor que realmente tenga esta variable en ese momento. Es fundamental para garantizar la precisión y fiabilidad de las mediciones.

D. Precisión

Es la tolerancia mínima de medida que permite indicar, controlar y registrar el instrumento. Es la mínima división de escala de un instrumento indicador. Se expresa generalmente en porcentaje % del span.

E. Sensibilidad

Es la relación entre la variación de la lectura del instrumento y el cambio en el proceso que causa este efecto. Refleja la capacidad de detectar pequeñas variaciones en la variable medida.

F. Zona muerta

Es el máximo campo de variación de la variable en el proceso real, para el cual el instrumento no registra ninguna variación en su indicación, registro o control. Puede influir en la precisión y efectividad de las mediciones y el control del proceso.

G. Repetitividad

Es la capacidad de un instrumento de repetir el valor de una medición bajo las mismas condiciones, de un mismo valor de la variable real en una única dirección de medición. Esto es vital en procesos donde se requieren mediciones precisas y constantes.

2.2.3 Clasificación de los instrumentos

La clasificación de los instrumentos es de acuerdo a:

- De acuerdo a la variable de proceso que se mide.
- De acuerdo a su función en el proceso.

De acuerdo a la variable de proceso que miden:

Esta clasificación, como su nombre lo indica, se refiere a la variable de proceso que tratemos de medir. En la actualidad, se pueden medir, casi sin excepción, todas las variables de proceso existentes, sin embargo, algunas se medirán de forma directa y otras indirectamente.

De acuerdo a su función, estos se agrupan en:

- **Instrumentos indicadores:** Son aquellos que indican directamente el valor de la variable de proceso. Ejemplos: manómetros, termómetros, etc.
- **Instrumentos registradores:** Son los que registran variables en un determinado proceso.
- **Elementos primarios:** Elementos que entran en contacto directo con la variable de proceso que se desea medir, con el fin de recibir algún efecto sobre este la cual pueden evaluar la variable en cuestión. (placa orificio).
- **Transmisores:** Elementos que reciben la variable de proceso a través del elemento primario, y la transmiten a algún lugar remoto, ya sea para su procesamiento en control. Estos transmiten las variables de proceso en una forma de señales proporcionales a esas variables.
- **Transductores:** Instrumentos que no están en contacto con el proceso, pero si son capaces de realizar operaciones lógicas y/o matemáticas con las señales de salida de uno o más transmisores. Convierte forma de energía en otra, facilitando las mediciones.
- **Convertidores:** Generalmente, la señal de salida de un transmisor, para ser compatible con su receptor se utiliza un elemento convertidor entre ellos para lograr finalmente la compatibilidad de señal.

- **Receptores:** Son instrumentos que generalmente son instalados en el panel de control, como interfaz entre el proceso y el hombre. Estos reciben la señal de los transmisores o de un convertidor.
- **Controladores:** Es el elemento más importante del sistema de control, ya que el instrumento será el encargado de enviar al controlador la información del proceso por medio de sus señales, de tal manera que el controlador pueda comparar datos que realmente se desea que sucedan en él, para posteriormente, diferenciar y enviar una señal al proceso que tienda a corregir las desviaciones de sus variables.
- **Elemento final de control:** Finalmente, será este elemento en el que recibirá la señal de control del controlador y quien estando en contacto directo con el proceso y ejerza un cambio en él, de tal forma que se cambien los parámetros hacia el valor deseado. Ejemplo: válvulas de control, compuertas, etc.

2.2.4 Medidores de flujo

Los medidores de flujo son instrumentos utilizados para determinar la cantidad de flujo volumétrico y/o másico que pasa a través del área transversal de una tubería. Otros nombres con los cuales suelen llamarse son: Flujómetro, caudalímetro o medidores de caudal. Los sensores de caudal permiten determinar la velocidad y cantidad del flujo de aire o líquidos que atraviesan una tubería o sistema.

Los sensores de caudal usan diferentes principios de medición. Los sensores de caudal para líquidos funcionan por ejemplo sobre la base de ultrasonidos. (Miller, 1996)

A. Medidores de caudal másico

Examina las unidades de medida de flujo másico. La tasa de flujo másico mide la masa por unidad de tiempo, por lo que la unidad estándar de medida para la velocidad de flujo másico es el kilogramo por segundo (kg/s).

- Térmico: Diferencia de temperaturas en dos sondas de resistencia
- Fuerza de Coriolis: Tubo en vibración

2.2.5 Medidores de temperatura

Definimos a la temperatura como el movimiento o el grado de agitación térmica de las moléculas. La medición de temperatura constituye una de las medidas más comunes e importantes que se efectúan en los procesos industriales. La calidad de los productos depende grandemente de su exacta medición y control.

Las limitaciones del sistema de medición empleado quedan definidas por la precisión, la velocidad de captación de la temperatura, por la distancia entre el elemento de medida y el aparato receptor y por el tipo de instrumento controlador necesarios. Los instrumentos de medición de temperatura utilizan diversos fenómenos que son influidos por la temperatura de difícil adquisición de datos lo cual fue desarrollándose en una estructura estadística cuya exactitud puede verse afectada por la masa térmica, el tiempo de medida, el ruido eléctrico y los algoritmos de medida para hacer del dispositivo de medición más exacto aún en condiciones óptimas.

A. Escalas de temperatura

En la actualidad se emplean diferentes escalas de temperatura; entre ellas está la escala Celsius, también conocida como escala centígrada, la escala Fahrenheit, la escala kelvin, la escala Rankine que son definidas para la medición de temperatura, como la figura 1.

En la escala Celsius, el punto de congelación del agua equivale a 0°C y su punto de ebullición a 100°C . Esta escala se utiliza en la mayoría de los registros de temperatura en la actualidad, la cual, a comparación de la escala Fahrenheit se emplea en los países anglosajones de uso no científico.

B. Dispositivos electrónicos de medición de temperatura

RTD: Un RTD (Resistance Temperature Detector) detecta cambios de temperatura mediante la variación en la resistencia eléctrica de materiales como platino o níquel, cuyo comportamiento es predecible frente a incrementos térmicos.

- RTD de dos hilos.
- RTD de tres hilos.
- RTD de cuatro hilos.

Termistor: Mide la variación de resistencia de un semiconductor.

- NTC: Coeficiente de temperatura negativo.
- PTC: Coeficiente de temperatura positivo.

Termopar: Llamado también termocupla, que produce una diferencia de potencial en milivoltios en función de la diferencia de temperatura entre el punto caliente y el otro punto frío como referencia.

C. Sensor detector resistivo de temperatura (RTD)

El detector de temperatura RTD se basa en el principio de la variación de la resistencia de un conductor con la temperatura con coeficiente de temperatura positivo. La variación de la resistencia puede ser expresada de manera polinómica. Por lo general,

la variación es bastante lineal en márgenes amplios de temperatura según la siguiente ecuación Ec. 5.2.1

$$R_T = R_0 * (1 + \alpha * t)$$

Ec. 5.2.1 Ecuación de variación de resistencia.

- R_0 : resistencia en ohmios a 0 grados Celsius
- R_t : resistencia en ohmios a t grados Celsius
- Es el coeficiente de temperatura del conductor especificado a 0 °C, interesa que sea de gran valor y constante con la temperatura.

D. Características de funcionamiento del sensor RTD.

Tabla 2

Características de Materiales de RTD

Metal	Resistividad	Coeficiente Temperatura	Intervalo Temperatura	Resistencia a 0°C	Presión °C
Platino	9.83	0.003850	-200 -950	25,100,130	0.01
Niquel	6.38	0.0063	-15-300	100	0.50
Cobre	1.56	0.00425	-200-120	10	0.1

Nota: Tabla de características de material de RTD

Un sensor RTD muy común es el PT100 (PT significa material de platino con $R=100$ a 0 °C). En la siguiente tabla 2 se muestran valores estándar de resistencia a distintas temperaturas para un sensor Pt100 con $\alpha = 0.00385$. También realiza medidas más exactas y estables hasta una temperatura de aproximadamente 500 °C.

Tabla 3

Valores de Resistencia y Temperatura

Temperatura (°C)	0	20	40	60	80	100
Resistencia (Ohm)	100	109.79	115.54	123.24	130.8	138.50

Nota: Tabla de valores de resistencia y temperatura del Platino

Los RTD más económicos utilizan níquel o aleaciones de níquel, pero no son tan estables ni lineales como los que emplean platino. En la Figura. 2. se indica la apariencia física del sensor.

En cuanto a las desventajas, el platino encarece los RTD. Además, para medir la resistencia hay que aplicar una corriente, que, por supuesto, produce una cantidad de calor que distorsiona los resultados de la medida.

El ser tan baja la resistencia de los hilos conductores que conectan el RTD puede provocar errores importantes. En la denominada técnica de dos hilos la resistencia se mide en los terminales del sistema de adquisición de datos, por lo que la resistencia de los hilos forma parte de la cantidad desconocida que se pretende medir según la figura 3.

La velocidad de reacción de la señal de salida se ve afectadas por la masa y el tamaño de los RTD además también por su autocalentamiento.

Por el contrario, la técnica de cuatro hilos mide la resistencia en los terminales del RTD, con lo cual la resistencia de los hilos queda eliminada de la medida. (Omega,2019)

2.3 Medidores de vibración

Elemento que tiene la capacidad de medir la frecuencia y la magnitud de vibración operacional de un sistema. Estas mediciones son útiles para detectar problemas en el sistema que puedan aquejar a una máquina. (Mobley, 2002)

2.4 Tipos de sensores de vibración

2.4.1 Acelerómetros:

Circuitos integrados que miden la aceleración, que es el campo de la velocidad por unidad de tiempo.

2.4.2 Sensores de velocidad:

Se utilizan para mediciones de frecuencia baja a media como, por ejemplo, monitorear vibraciones y operaciones de balanceo en maquinarias rotativas, al tener menor sensibilidad a las vibraciones de alta frecuencia en comparación de los acelerómetros.

2.4.3 Sensores de proximidad:

Se usa principalmente en equipos de soportes hidrodinámicos debido a la transmisión a través de la carcasa de soporte, como es el caso de ejes montados sobre rodamientos.

2.5 Sensor de presión

Los sensores de presión su objetivo es transformar una magnitud física en una eléctrica, en estos casos transforman una fuerza por unidad de superficie (presión) en voltaje proporcional a la presión ejercida.

2.6 Revisión de antecedentes

Se encontraron antecedentes que tienen un enfoque similar con nuestro objetivo, ayudando a entender y plasmar una idea más concreta de lo que queremos desarrollar. Ante esto paso a detallar:

2.6.1 Primer antecedente:

Rodríguez Durán, J. H. (2023) desarrolló un sistema de monitoreo y medición de rendimiento basado en Internet de las Cosas (IoT) aplicado a sistemas de bombas centrífugas, empleando un entorno de laboratorio con tanques de control para simular procesos industriales similares a los utilizados en la gran minería. El autor implementó tecnologías propias de la Industria 4.0, integrando sensores y un PLC para la adquisición de datos, permitiendo la obtención de curvas de operación y su posterior visualización en plataformas digitales.

En contraste, la presente investigación propone el desarrollo de un sistema piloto físico, compuesto por una bomba centrífuga, tuberías y un tanque de almacenamiento, orientado a comprender el comportamiento de bombas de mayor potencia a escala reducida. A diferencia del antecedente, el prototipo planteado está enfocado en unidades mineras de media y baja producción, priorizando una solución de bajo costo, con adquisición de datos en tiempo real mediante un sistema de monitoreo no implementado actualmente en este tipo de operaciones.

2.6.2 Segundo antecedente:

Ahmad, Rai, Maliuk y Kim (2020) presentan una investigación orientada al diagnóstico de fallas en bombas centrífugas mediante técnicas avanzadas de extracción de características discriminantes. El estudio identifica factores críticos que afectan la confiabilidad de los sistemas de bombeo, tales como condiciones incorrectas de flujo de

entrada, insuficiente cabeza neta positiva de succión (NPSH), ingreso de aire, desbalanceo mecánico y una baja frecuencia de mantenimiento.

Tomando como base estos resultados, el presente trabajo considera la medición de variables clave como presión y flujo para evaluar las condiciones hidráulicas del sistema, así como la vibración para identificar problemas de desbalanceo en el motor y la bomba. Adicionalmente, la medición de temperatura permite detectar fenómenos asociados a la cavitación y desgaste prematuro, mientras que el monitoreo de voltaje y corriente posibilita la identificación de picos eléctricos que podrían comprometer la integridad del motor, contribuyendo así a mejorar el rendimiento operativo de las bombas centrífugas.

2.6.3 Tercer antecedente:

Sánchez Gonzales, J. R. (2021) desarrolló una aplicación de monitoreo de condiciones para el diagnóstico de fallas en motores y transmisiones de cargadores Caterpillar 966H, empleando técnicas de machine learning. El objetivo principal del estudio fue reducir los costos de mantenimiento mediante el uso de tecnologías emergentes, enfocándose en el análisis del desgaste de componentes a partir de variables relacionadas con el estado del aceite lubricante.

En relación con este antecedente, la presente tesis propone un prototipo de monitoreo de bajo costo, orientado a la industria minera de pequeña y mediana escala, que permita reducir los costos asociados al mantenimiento correctivo y mejorar la programación de intervenciones tanto a corto como a largo plazo. A diferencia del trabajo citado, el enfoque se centra en la supervisión directa de variables eléctricas y mecánicas de una bomba centrífuga

2.6.4 Cuarto antecedente:

Pinto Guillén, V. J. (2021) desarrolló el diseño e implementación de medidores de energía eléctrica basados en IoT, orientados a la gestión del consumo energético. El trabajo emplea protocolos de comunicación como MQTT y una base de datos MySQL alojada en un servidor, permitiendo la generación de reportes y gráficos de consumo eléctrico en función del tiempo para aplicaciones domiciliarias.

De manera similar, el presente trabajo adopta protocolos de comunicación IoT y sistemas de bases de datos para el almacenamiento y visualización de la información. No obstante, la aplicación se orienta al monitoreo de bombas centrífugas, permitiendo la adquisición, almacenamiento y visualización en tiempo real de variables operativas, lo que facilita el análisis del comportamiento del sistema y la toma de decisiones técnicas.



CAPÍTULO III
DISEÑO DE HARDWARE

En el capítulo presente se observará los diferentes circuitos de cada parte de diseño, conexiones del equipo de monitoreo con sus diferentes partes, análisis comparativo de selección de componentes electrónicos, un diagrama de bloques general del diseño, proceso de modelado electrónico.

3. Análisis Comparativo de Selección de Componentes Electrónicos

La selección de cada componente electrónico del prototipo se realizó mediante un proceso de análisis comparativo técnico que consideró: rango de medición, precisión, interfaz eléctrica compatible con el microcontrolador Hércules RM46L852, compatibilidad con señal estándar 4-20 mA (donde aplica), disponibilidad en el mercado nacional y relación costo-beneficio para el contexto de unidades mineras de baja y media producción.

Se presenta el análisis comparativo de los cinco (5) componentes principales del sistema. Las filas resaltadas en verde indican el componente seleccionado para el prototipo. Las filas sin resaltar corresponden a alternativas evaluadas y descartadas, con el criterio de descarte indicado en la última columna.

Tabla 4

Corriente - Análisis comparativo de selección de componentes electrónicos

Sensor de Corriente	Rango (A)	Sensibilidad	Interfaz	Criterio de selección
ACS712-30A	±30 A	66 mV/A	Analógica	Rango insuficiente
ACS756-050A	±50 A	40 mV/A	Analógica 0-5 V	SELECCIONADO
WCS1800	±35 A	40 mV/A	Analógica	Rango insuficiente

Nota: Componente seleccionado. Elaboración propia a partir de datasheets y criterios

Tabla 5

Vibración - Análisis comparativo de selección de componentes electrónicos

Sensor de Vibración	Rango (g)	Ancho de banda	Salida	Criterio de selección
ADXL345	±16 g	0.1 – 1.6 kHz	Digital I ² C/SPI	Rango dinámico limitado
MPU-6050	±16 g	0.1 – 1 kHz	Digital I ² C	Precisión insuficiente
PC420VR-05	0 – 250 g	3.5 Hz – 2 kHz	4 – 20 mA	SELECCIONADO

Nota: Componente seleccionado. Elaboración propia a partir de datasheets y criterios

Tabla 6

Presión - Análisis comparativo de selección de componentes electrónicos

Sensor de Presión	Rango (MPa)	Precisión	Salida	Criterio de selección
MPX5010DP	0 – 0.1 MPa	±5 %	Analógica 0-5 V	Rango insuficiente
HK3025	0 – 5.0 MPa	±1 % FS	Analógica 0.5-4.5 V	SELECCIONADO
PT5501	0 – 1.6 MPa	±0.5 %	Analógica 4-20 mA	Rango insuficiente

Nota: Componente seleccionado. Elaboración propia a partir de datasheets y criterios

Tabla 7

Caudal - Análisis comparativo de selección de componentes electrónicos

Sensor de Caudal	Rango (L/min)	Salida	Rosca	Criterio de selección
YF-S201	1 – 30 L/min	Pulso digital	G1/2"	Bajo rango y precisión
TF-S201	1 – 30 L/min	Pulso digital 7.5 Hz/L	1/2"	SELECCIONADO
FS300A	0.1 – 10 L/min	Pulso digital	G1/2"	Rango muy reducido

Nota: Componente seleccionado. Elaboración propia a partir de datasheets y criterios

Tabla 8

Microcontrolador - Análisis comparativo de selección de componentes electrónicos

Microcontrolador	Bits / Frec.	ADC	Mem. Flash	Criterio de selección
STM32F407	32 bit / 168 MHz	12 bits / 3.3 V	1 MB	Sin soporte safety
Arduino Mega 2560	8 bit / 16 MHz	10 bits / 5 V	256 KB	Insuficiente velocidad
TMS570LS (Hércules)	32 bit / 180 MHz	12 bits / 3.3 V	1.25 MB	SELECCIONADO

Nota: Componente seleccionado. Elaboración propia a partir de datasheets y criterios

A continuación se desarrolla la justificación técnica específica de cada componente seleccionado:

a) Microcontrolador TMS570LS (Hércules RM46L852)

Fue seleccionado frente a alternativas como el STM32F407 y el Arduino Mega 2560 por tres razones determinantes: (1) arquitectura de doble núcleo ARM Cortex-R4F con comprobación de errores en tiempo real (lockstep), idónea para entornos industriales críticos; (2) ADC de 12 bits con velocidad de conversión que permite frecuencias de muestreo de 60 μ s (16.6 kHz), necesarias para el análisis espectral FFT de vibraciones; y (3) soporte nativo de múltiples interfaces de comunicación (SPI, I2C, SCI/UART, CAN) que facilitan la integración de todos los módulos del prototipo.

b) Sensor de corriente ACS756-050A

El sensor de efecto Hall ACS756-050A fue seleccionado sobre el ACS712-30A y el WCS1800 porque ofrece un rango de medición de ± 50 A, suficiente para capturar la corriente de arranque del motor (pico registrado $\approx \pm 40$ A durante las pruebas), frente a los ± 30 A y ± 35 A de los alternos. Adicionalmente, su resistencia interna de 100 $\mu\Omega$ minimiza las pérdidas en el conductor primario y su tensión de aislamiento de 3 kV RMS garantiza la separación galvánica necesaria en instalaciones eléctricas industriales.

c) Sensor de vibración PC420VR-05

Fue preferido sobre el ADXL345 y el MPU-6050 por su rango dinámico extendido de 0 a 250 g pico y su ancho de banda de 3.5 Hz a 2.0 kHz, que cubre las frecuencias características de los principales modos de falla en bombas centrífugas (desbalance, cavitación, desgaste de rodamientos). Su salida estándar de 4-20 mA

proporciona inmunidad al ruido electromagnético propia de ambientes industriales, a diferencia de las salidas digitales I2C/SPI de los sensores MEMS descartados.

d) Sensor de presión HK3025

El HK3025 fue seleccionado por su rango de 0 – 5.0 MPa (50 bar), que permite monitorear tanto condiciones de operación nominal como picos de presión por golpe de ariete en sistemas de bombeo. El MPX5010DP fue descartado por su rango máximo de 0.1 MPa (insuficiente para el sistema piloto), y el PT5501 por su rango de 1.6 MPa, inadecuado para proyecciones de escala industrial. Su precisión de $\pm 1\%$ FS es consistente con las tolerancias requeridas para detección de cavitación.

e) Sensor de caudal TF-S201

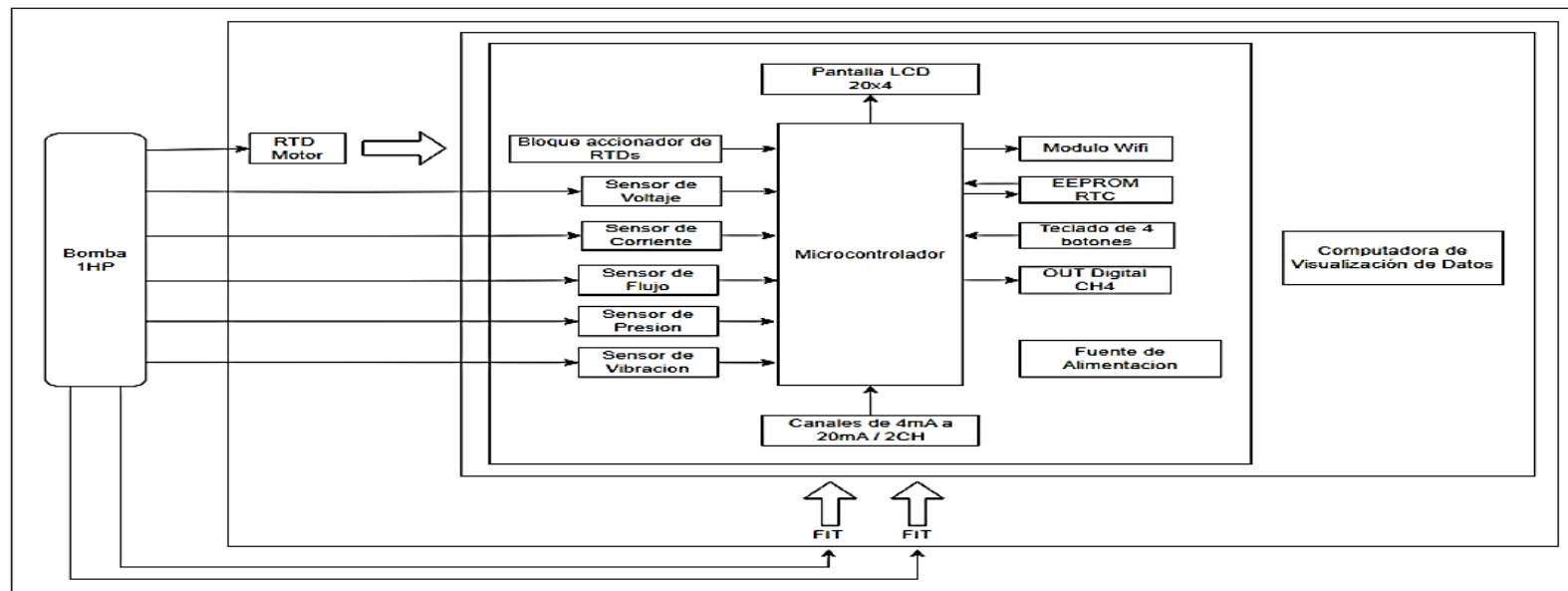
El sensor TF-S201 fue seleccionado sobre el YF-S201 y el FS300A por combinar el rango de 1 – 30 L/min (adecuado para el sistema piloto de 15 L) con una resolución de 2.25 mL/pulso y una presión máxima de 1.75 MPa. Su rotor de acero inoxidable garantiza durabilidad frente a fluidos con partículas abrasivas, característica relevante para la proyección del sistema hacia unidades mineras con bombas de lodo.

En el capítulo presente se observará los diferentes circuitos de cada parte de diseño, conexiones del equipo de monitoreo con sus diferentes partes, también se muestra un diagrama de bloques general del diseño.

3.1 Diagrama de bloques general

Figura 1

Diagrama de bloques general



Nota: Elaboración propia , diagrama de bloques general del prototipo de monitoreo

En el diagrama de bloque general de diseño podemos visualizar la bomba 1 hp la cual nos permitirá recolectar los parámetros de operación principales para verificar el comportamiento y que puedan verificar la eficiencia de la bomba en un sistema de circuito cerrado, el cual hace referencia a un circuito de escala industrial.

Se visualiza en el diagrama de bloques general de diseño los sensores que nos ayudarán a obtener los parámetros principales de operación de la bomba de 1hp, los cuales son sensor de voltaje, sensor de corriente, sensor de flujo, sensor de presión, sensor de vibración; sensor RTD, serán instalados en el prototipo, bien sea en la línea de alimentación, descarga o en el mismo motor.

Estamos considerando un teclado de 4 botones para realizar la programación del prototipo que se enciende con una fuente de alimentación de 220 V convirtiéndolo con una fuente de alimentación de 220 V a 24 V para los circuitos integrados que se dispone. Así mismo, tenemos salidas digitales para recolección de datos y/o conexión de circuitos integrados de 4–20 mA que nos ayudarán a la recolección de datos.

Además, podemos apreciar un módulo wifi, el cual nos permitirá enviar los datos recolectados de los sensores a una nube y almacenarlos en una memoria Server SQL y visualizarlos en el prototipo los diferentes parámetros recolectados en una pantalla LCD 20x14 y en un Dashboards mediante una pantalla digital (laptop).

3.1.1 Proceso de Modelado Electrónico del Prototipo

Constituye la etapa metodológica que permite transformar los requerimientos funcionales del sistema de monitoreo en circuitos electrónicos cuantificables y verificables. A continuación se describe la secuencia seguida para modelar cada bloque del prototipo, desde la adquisición de señal hasta la transmisión de datos en la nube.

3.1.2 Metodología de modelado adoptada

Para el modelado de cada etapa del circuito se adoptó el método de modelado circuital analítico, que consiste en:

- Identificación de la variable física a medir y su rango de operación industrial.
- Selección del transductor o sensor compatible con el rango y la interfaz eléctrica requerida.
- Desarrollo del modelo matemático que describe la transferencia de la señal física a magnitud eléctrica (corriente o tensión).
- Diseño o selección del circuito de acondicionamiento para adaptar la señal a los niveles del ADC del microcontrolador (0 – 3.3 V, 12 bits).
- Verificación analítica del circuito mediante cálculos de valores nominales y comprobación de condiciones de borde.
- Integración del bloque modelado al diagrama general del prototipo.

3.1.3 Modelos matemáticos de cada bloque electrónico

La Tabla 9 resume el modelo electrónico de cada etapa de adquisición y acondicionamiento del prototipo, indicando la ecuación de transferencia y el resultado obtenido en cada caso.

Tabla 9

Resumen del proceso de modelado electrónico por bloque del prototipo

Etapa del modelo	Descripción	Resultado / Ecuación
1. Circuito de suministro $V_{cc}/2$	Generación de tierra virtual mediante divisor resistivo $R1=R2$ + filtro RC	$V_{cc}/2 = 2.5$ V (tierra virtual estable)

2. Amplificador de diferencia (sensor voltaje)	Circuito restador para ajuste de nivel de señal en canales de adquisición	$V_o = (R_4/R_3) \times (V_2 - V_1)$
3. Modelo lineal sensor de corriente ACS756	Caracterización curva V-I del sensor a partir de hoja de datos	$V_o = 0.04 \cdot I_{in} + 2.5 \text{ [V]}$
4. Divisor de tensión (acondicionamiento ADC)	Adaptación de 5 V a 3.3 V para proteger ADC del Hércules	$R_1=5.1 \text{ k}\Omega, R_2=10 \text{ k}\Omega \rightarrow V_o=3.31 \text{ V}$
5. Modelo RTD – Acondicionador de temperatura	Puente de Wheatstone + amplificador de instrumentación para sensor PT100	$V_{OUT} \propto \Delta R_{RTD} \rightarrow T(^{\circ}\text{C})$
6. Modelo sensor de vibración (señal 4-20 mA)	Conversión RMS → corriente mediante módulo SEN0262 ($I = 4 + 16/250 \times x_{RMS}$)	$I = 4 + (16/250) \times x_{RMS} \text{ [mA]}$

7. Cálculo de potencia
activa (kW) y aparente
(kVA)

A partir de V_RMS e I_RMS
medidos por el prototipo

$$P = V \cdot I \cdot \cos(\varphi) / 1000$$

8. Modelo espectral FFT
(vibración)

Transformada rápida de
Fourier con N=1024 muestras
y Ts=60 μ s

$$F_s = 16\,666 \text{ Hz}; f_{\text{Nyquist}} = 8\,333 \text{ Hz}; \Delta f = 16.28 \text{ Hz}$$

9. Grosor de pistas PCB

Cálculo según corriente
máxima y temperatura de
trabajo (IPC-2221A)

Ancho $\geq 1 \text{ mm}$ para traza de
2 A a 25 °C de elevación

10. Relación caudal-
frecuencia (sensor TF-
S201)

Conversión pulsos digitales a
L/min (efecto Hall)

$$Q [\text{L/min}] = \text{Frecuencia} [\text{Hz}] / 7.5$$

Nota: Elaboración propia. Modelos derivados de hojas de datos y análisis circuital.

3.1.4 Descripción del proceso de modelado por etapa

A continuación se describe brevemente el proceso de modelado para las etapas más representativas del diseño electrónico:

a) Modelado del sensor de voltaje (circuito suministro $V_{cc}/2$ + amplificador de diferencia)

El sensor de voltaje requiere una referencia de tierra virtual a $V_{cc}/2 = 2.5 \text{ V}$, generada mediante un divisor resistivo ($R1 = R2$) con un condensador de filtro $C1$ que elimina el ruido conducido a alta frecuencia. La expresión que gobierna el circuito de tierra virtual es:

$$V_{cc}/2 = V_{cc} \times R2 / (R1 + R2) \rightarrow \text{Para } R1 = R2: V_{cc}/2 = 2.5 \text{ V}$$

El amplificador de diferencia asociado opera como circuito restador para la adaptación de los canales de medición de voltaje trifásico, cuya función de transferencia es:

$$V_o = (R4/R3) \times (V2 - V1)$$

b) Modelado del sensor de corriente ACS756-050A (efecto Hall)

A partir de la curva de transferencia del sensor se obtuvo el modelo lineal mediante la ecuación de la recta que relaciona la corriente de entrada (I_{in}) con el voltaje de salida (V_o):

$$m = (y1 - y0) / (x1 - x0) = (5 \text{ V} - 2.5 \text{ V}) / (50 \text{ A} - 0 \text{ A}) = 0.05 \text{ V/A}$$

$$V_o = 0.04 \times I_{in} + 2.5 \text{ [V]} \quad (\text{sensibilidad corregida por temperatura} = 40 \text{ mV/A})$$

El divisor de tensión $R1 = 5.1 \text{ k}\Omega$ / $R2 = 10 \text{ k}\Omega$ adapta la salida máxima de 5 V a 3.31 V , dentro del rango del ADC del microcontrolador Hércules RM46L852 ($0 - 3.3 \text{ V}$).

c) Modelado del sensor de vibración PC420VR-05 (señal 4-20 mA)

El valor RMS de vibración (x_{RMS}) es convertido por el sensor a una señal estándar de 4 a 20 mA conforme a la siguiente relación lineal:

$$I = 4 + (16 / 250) \times x_{RMS} \quad [\text{mA}] \quad \text{donde } x_{RMS} \text{ en [g]}$$

El módulo SEN0262 convierte la señal de corriente a voltaje (0 – 3 V), compatible con el ADC del Hércules. El cálculo del valor RMS en el microcontrolador sigue la secuencia: adquisición → elevación al cuadrado → promedio → raíz cuadrada.

d) Modelado del análisis espectral (FFT)

El modelo de adquisición para el análisis en el dominio de la frecuencia está definido por los parámetros de muestreo configurados en el microcontrolador:

- Tiempo de muestreo: $T_s = 60 \mu\text{s} \rightarrow$ Frecuencia de muestreo: $F_s = 1/T_s = 16\,666 \text{ Hz}$
- Número de muestras por ventana: $N = 1\,024$
- Resolución frecuencial: $\Delta f = F_s / N = 16\,666 / 1\,024 = 16.28 \text{ Hz}$
- Frecuencia máxima observable (Nyquist): $f_{\text{max}} = F_s / 2 = 8\,333 \text{ Hz}$

Este modelo garantiza la observación de hasta 138 armónicos respecto a la fundamental eléctrica de 60 Hz, y hasta el 8º armónico respecto a la frecuencia fundamental mecánica del sensor de vibración (1 kHz), cubriendo los patrones de falla más relevantes en bombas centrífugas.

3.2 Diagrama de bloques específicos

3.2.1 Sensor de voltaje

Los sensores de voltaje facilitan el seguimiento de los niveles de voltaje de alimentación, identificando problemas de sobrevoltaje, su voltaje ayudando a proteger la

parte electrónica de importancia, motores, etc. Podemos fácilmente acoplar a una registradora datos para su seguimiento en tiempo real.

A. Suministro de voltaje de salida

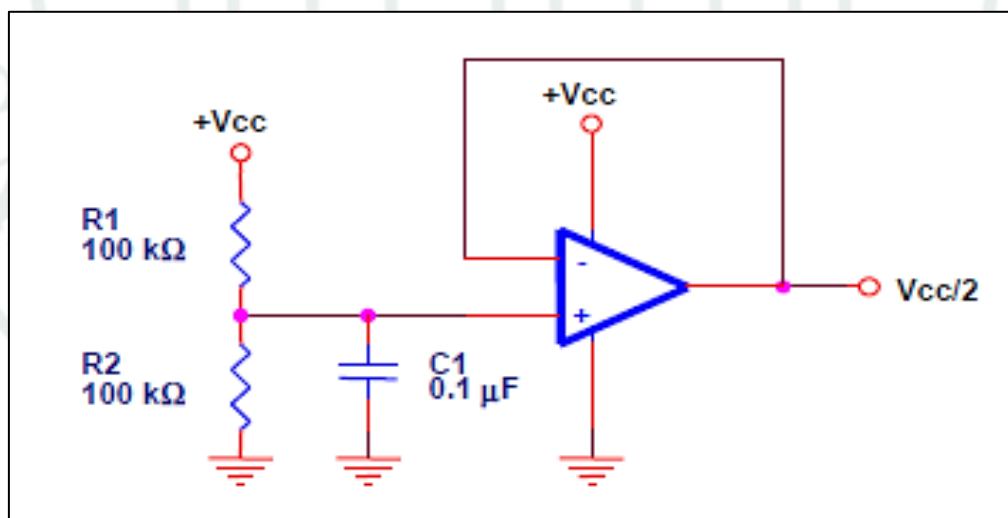
El diseño del suministro de voltaje de salida $V_{cc}/2$ requiere de la generación de una tierra virtual, con lo que se puede calcular el voltaje de salida de $V_{cc}/2$ de la figura, pero el rendimiento de este circuito tendrá un deterioro en bajas frecuencias.

Los valores de $R1$ y $R2$ son de valores iguales estándar, teniendo en cuenta el consumo de energía que se tendrá frente al ruido.

El condensador $C1$ forma un filtro pasa bajo para la eliminación del ruido conducido, se puede omitir el amplificador operacional de búfer en este caso.

Figura 2

Suministro de Voltaje de Salida $V_{cc}/2$



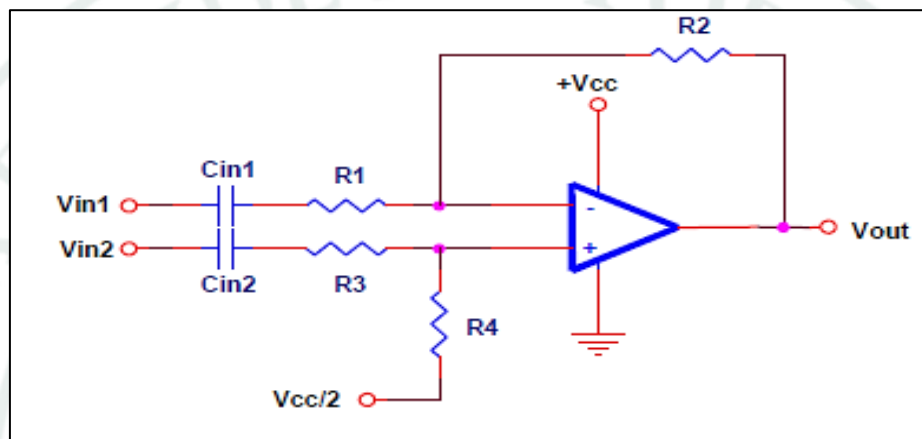
Nota: Elaboración propia, diseño de circuito de suministro de voltaje de salida

B. Amplificador de diferencia

Este circuito restador sirve para ajustar los niveles de voltaje en ambos canales de la operación de suministro de salida del voltaje de salida $V_{cc}/2$.

Figura 3

Circuito de resta de Voltaje de Salida $V_{cc}/2$

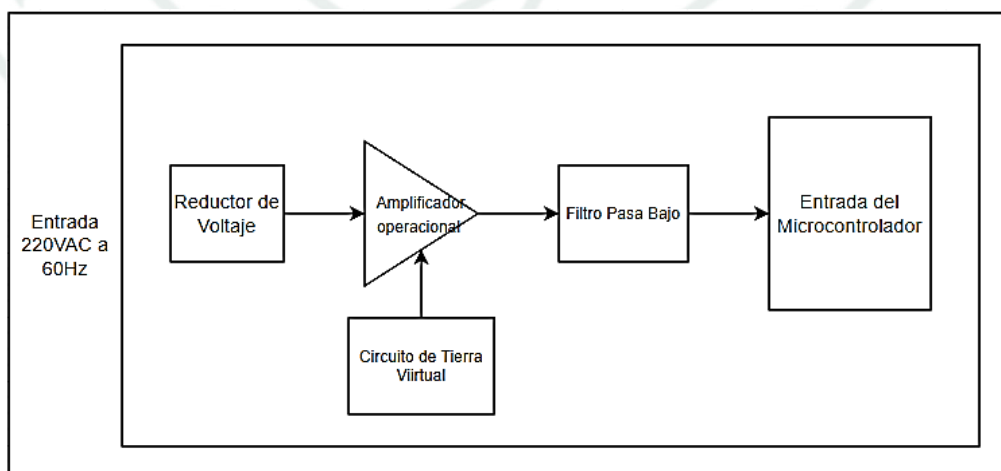


Nota: Elaboración propia, diseño de circuito de resta de voltaje de salida.

3.2.2 Diagrama de bloques específicos

Figura 4

Diagrama de Bloques Específico del Suministro del voltaje de Salida



Nota: Elaboración propia, diseño de diagrama de bloques específicos de voltaje salida.

Tenemos un voltaje de entrada de 220 VAC a 60 Hz al prototipo, lo cual entra al reductor de voltaje que se encargará de ingresar los datos obtenidos en una ventana de relación de 0 V a 3.3 V por el ADC del microcontrolador gracias a nuestro reductor operacional conectado a un circuito de tierra, evitando la radiofrecuencia y obteniendo un voltaje continuo gracias a un filtro pasa bajo, ingresando al microcontrolador como un dato obtenido.

3.2.3 Diseño de sensor de voltaje

De acuerdo al diseño de sensor de voltaje, el voltaje de medida de energía es 600 Vrms (normatividad de clase C) por lo tanto, definimos la variable V_{in} de la siguiente manera:

$$V_{in}: 0 \rightarrow 600 \text{ Vrms}$$

En la etapa de elevación de voltaje lo que se requiere el voltaje de entrada (V_{in}) del prototipo 600 Vrms; tenemos que tener un voltaje pico.

$$\text{Para ondas sinusoidales } V_p = V_{rms} \times \sqrt{2}$$

Aplicamos:

$$V_p = 600 \times \sqrt{2}$$

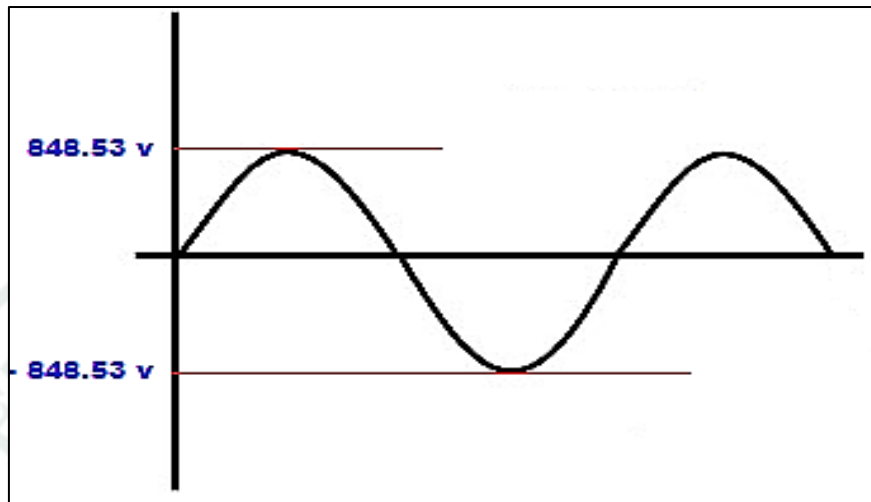
$$V_p = 848.53 \text{ V}$$

Obtenemos:

Tenemos un voltaje pico de entrada $V_p = 848.53 \text{ V}$

Figura 5

Onda sinusoidal de Voltaje Pico



Nota: Elaboración propia, señal de onda sinusoidal de Voltaje Pico

Es recomendable no trabajar al tope (848.53 v) darle un +13 % de margen de seguridad, si damos un margen de seguridad más alto (+13 %) al recomendado, perdemos precisión en la medición de voltajes bajos, más rango menor precisión (instrumentación).

Rango de diseño:

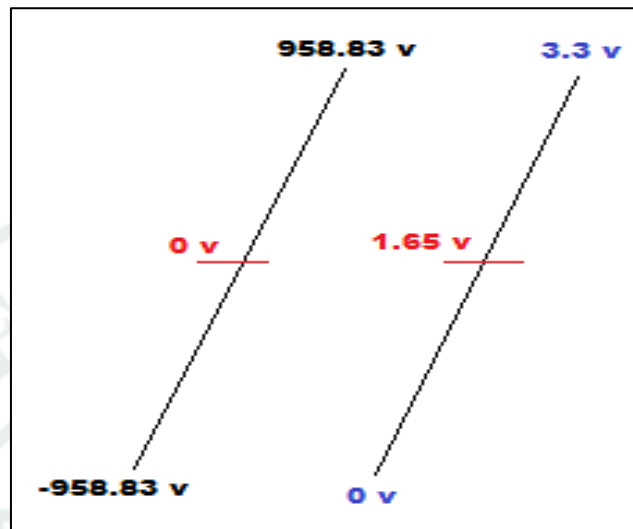
$$848.53 \times 1.13 \% = 958.83v$$

Obtenemos:

Un voltaje de entrada de 0 v a un voltaje máximo de 848.53 v queremos ingresar ese dato por el ADC del microcontrolador de 0v a un voltaje de 3.3 v en una ventana de relación (Offset = 1.65 v)

Figura 6

Rango de diseño ADC microcontrolador



Nota: Elaboración propia, rangos de diseño del microcontrolador ADC

Análisis de la gráfica:

$$Y - Y_0 = m(x - x_0)$$

$$m = \frac{(3.3 - 1.65)}{(958.83 - 0)}$$

$$m = 1.72084 \cdot 10^{-3}$$

Despejamos:

$$y - 1.65 = m(x - 0)$$

$$y = m \cdot x + 1.65$$

$$y = 1.72084 \cdot 10^{-3} \cdot x + 1.65$$

Obtenemos:

$$V_{out} = 1.72084e^{-3} * V_{in} + 1.65$$

3.2.4 Análisis del circuito diseñado

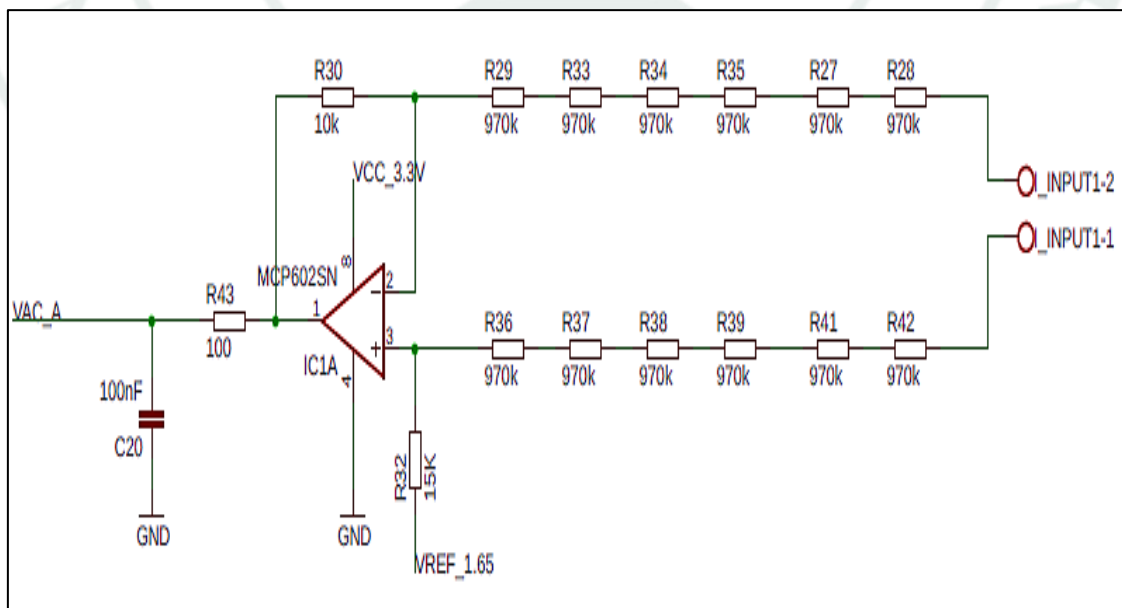
A. Análisis de suministro de voltaje de salida

Se define como la tensión de salida del circuito elevador de voltaje anteriormente diseñado. Para poder cerrar el lazo de control con el microcontrolador, se requiere llevar un rango de voltaje de 0 a 848.53 V a un rango aceptable por el microcontrolador de 0 a 3.3 Voltios, teniendo en consideración la protección (+13 %) para no dañar el controlador de voltaje 958.83 V.

$V_p = 848.53 \text{ V}$ Es el voltaje típico que se requiere para obtener de 600 Vrms, por lo que se sobredimensiona el voltaje de sensado a $V_p = 958.83 \text{ V}$.

Figura 7

Diagrama de Suministro de voltaje de salida

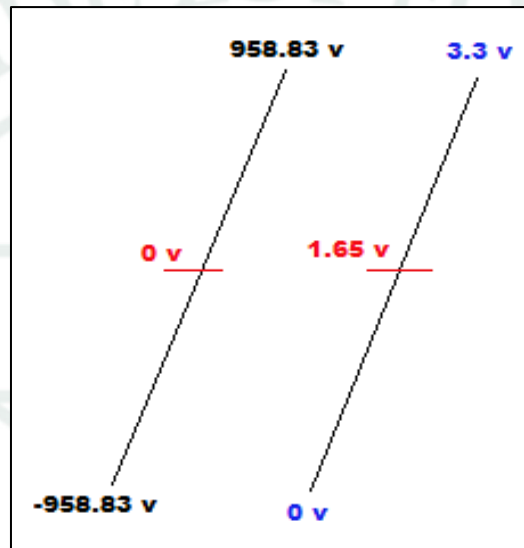


Nota: Elaboración propia, diseño de diagrama de suministro de voltaje de salida

Lo que conseguimos con este circuito es poder ingresar un voltaje proporcional de la etapa de potencia hacia el microcontrolador mediante un divisor de voltaje, la proporción sería:

Figura 8

Rango de diseño ADC microcontrolador

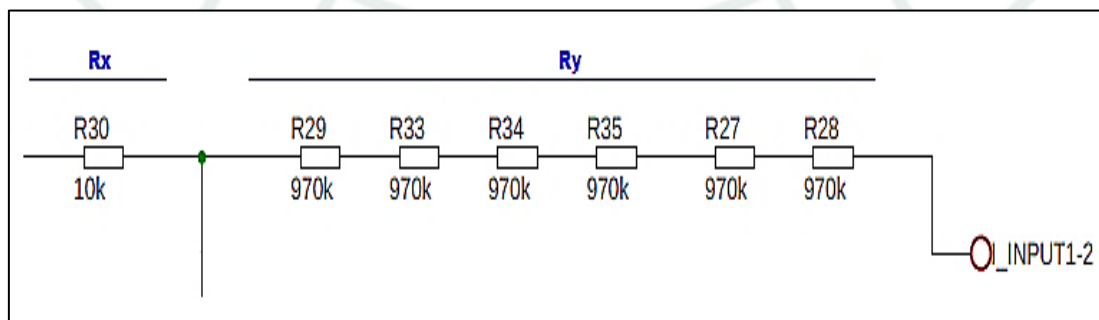


Nota: Elaboración propia, rango de diseño del microcontrolador ADC

Trabajando el divisor de voltaje

Figura 9

Divisor de Voltaje



Nota: Elaboración propia, diseño de divisor de voltaje

Fórmula división de tensión:

$$V_o = \frac{V_{in}(R_x)}{R_x + R_y}$$

$$3.3 = \frac{600(R_x)}{R_x + R_y}$$

$$3.3R_x + 3.3R_y = 400(R_x)$$

$$R_y = 120.21(R_x)$$

$$R_y = 1.8M$$

Como no contamos con una resistencia de 1.8M ohm y solo resistencias en k ohm usamos las siguientes resistencias en serie:

$$\frac{R_y}{6} = 970 K\Omega$$

Buscamos unas resistencias comerciales y cerca del valor encontrado es de 970 K Ω

Recalculando el voltaje de salida con valores de resistencia comercial.

$$V_o = \frac{600(15 \times 10^3)}{(970 \times 10^3) \times 4 + 15 \times 10^3}$$

$$V_o = 3.16 V < 3.3V$$

Formula de potencia:

$$P = I^2 R$$

Como sabemos, para poder hallar la corriente tenemos la siguiente fórmula:

$$I = \frac{V}{R}$$

$$I = \frac{600}{(970 \times 10^3) \times 4 + 15 \times 10^3}$$

$$I = 211 \mu A$$

$$P_{470K} = I^2 R$$

$$P_{470K} = (211 \mu)^2 (970K)$$

$$P_{470K} = 0.02 W \quad \frac{1}{8} W = 0.125 W$$

$$P_{15K} = (211 \mu)^2 (15K)$$

$$P_{15K} = 0.66 mW \quad \frac{1}{8} W = 0.125 W$$

Por lo que se requiere de 12 resistencia de 470k Ohm a 1/8 Watt}

B. Análisis del amplificador de diferencia

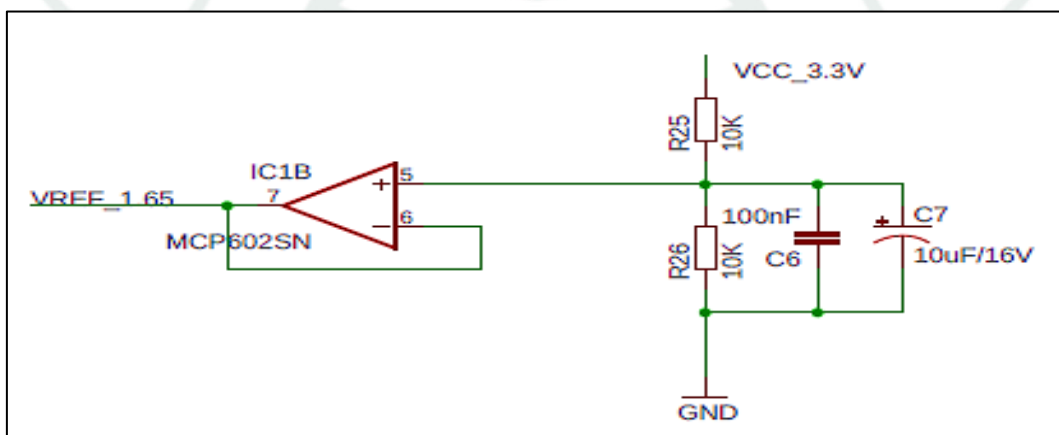
Este circuito restador sirve para ajustar los niveles de voltaje en ambos canales de la operación de suministro de salida del voltaje de salida Vref.

C6 = Es un filtro pasabajo que nos evita la radiofrecuencia.

C7 = Sirve para evitar el rizado de voltaje y sea un voltaje continuo.

Figura 10

Circuito de resta de Voltaje de Salida Vref



3.3 Sensor de corriente

Sensores de corriente se usan para aplicaciones de un sistema de control automático para detectar la corriente en un conductor eléctrico, la cual ofrece una medida de intensidad diseñada para entornos de media tensión, ofrecen una señal precisa y confiable, sus características principales son:

- Amplio rango dinámico.
- Comportamiento lineal.
- Estándar.
- Seguridad y confiabilidad.
- Compatibilidad directa con dispositivos electrónicos inteligentes.
- Diseño compacto y ligero.
- Fácil instalación y puesta en marcha.

Usados tradicionalmente para proteger los circuitos de daños y de informes generales de corriente, los sensores de corriente ahora se usan en una gama mucho más amplia de aplicaciones. Estos incluyen la supervisión y mejora del rendimiento; protección contra sobrecorriente, que pueden sobrecargar una línea y ponerla en riesgo; y circuitos operados por baterías, como aquellos con cargadores de baterías.

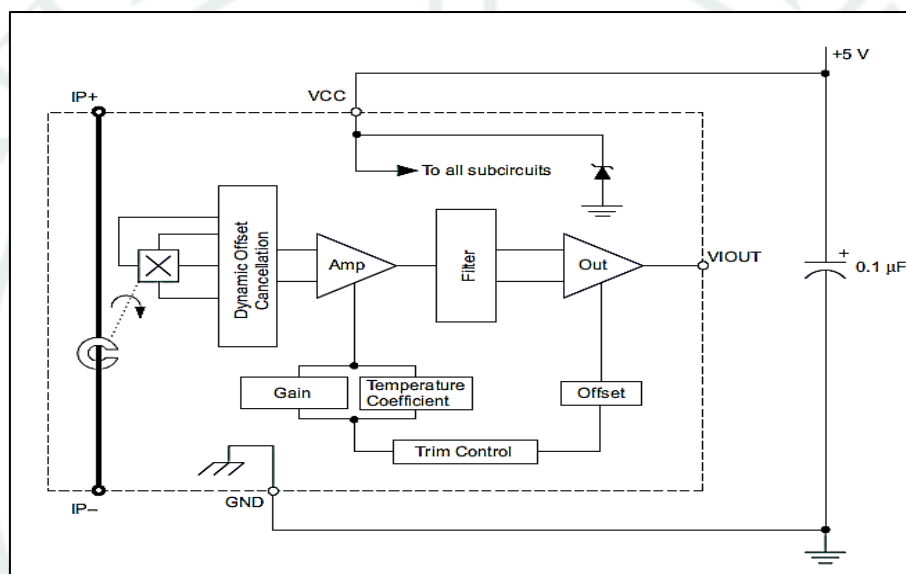
El funcionamiento de un sensor de corriente, se basa en su capacidad para detectar y convertir la intensidad eléctrica que circula por un conductor en una señal proporcional útil para sistemas de monitoreo.

3.3.1 Diagrama de bloque específico

Cuando pasa una corriente a través del sensor de corriente se genera un campo magnético, campo magnético generado hace que yo obtenga un voltaje de salida proporcional a la corriente que está ingresando por el sensor.

Figura 11

Diagrama de Bloques del sensor de temperatura



Nota: Elaboración propia, diagrama de bloques del sensor de temperatura

- **Dinamia Offset Cancellation:** Lee el campo magnético inducido y ajusta a los valores del offset (1.65V) para convertirlos a voltaje.
- **Amplificador:** Eleva la señal de voltaje, amplifica la señal.
- **Gain Temperature Coefficient:** Altera la ganancia del amplificador de instrumentación en función a los cambios de temperatura del circuito integrado.
- **Filter:** Quita las señales de radio frecuencia de $F_c = 120 \text{ KHz}$.
- **Opam out:** Acomoda la señal de OPAM de instrumentación a una ventana de voltaje de 3.3 V con un offset de 1.65 V.

NOTA: Amplificador de instrumentación la ventaja es que amplifica voltaje muy bajo en rangos de microvoltios, pero la desventaja es que también amplifica señales de ruido. Para evitar estas señales de ruido pasa por un filtro pasa bajo con una frecuencia de corte de $F_c=120 \text{ KHz}$.

Formula Campo Magnético:

$$B = \frac{\mu \cdot I}{2\pi d} \begin{cases} \mu \rightarrow \text{permeabilidad magnética del medio } \left(\frac{N}{A^2}\right) \\ I \rightarrow \text{Intensidad de corriente en Amperios (A)} \\ d \rightarrow \text{Distancia al punto de estudio en metros (m)} \end{cases}$$

3.3.2 Selección de sensor de corriente

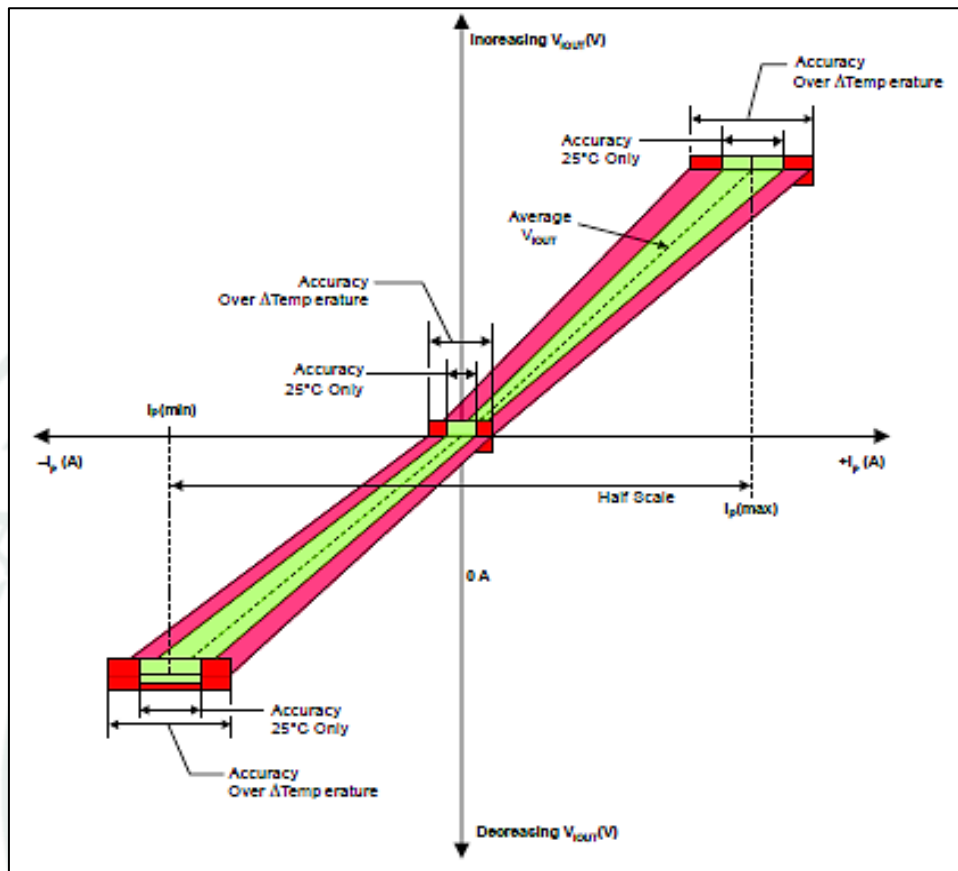
Como vamos a trabajar como una corriente máxima de 0 A a 50 A necesitamos un sensor de corriente que maneje más corriente, se seleccionó un sensor de corriente para ver con cuál es el más conveniente trabajar sensor AC5756-050A.

A. Sensor AC5 756-050 A

Es un sensor de efecto Hall que mide el campo magnético que se genera en un conductor cuando pasa una corriente a través de este sensor.

Figura 12

Efecto Hold del Sensor ACS 756-050 A



Nota: Elaboración propia, figura diagrama de efecto Hold de sensor

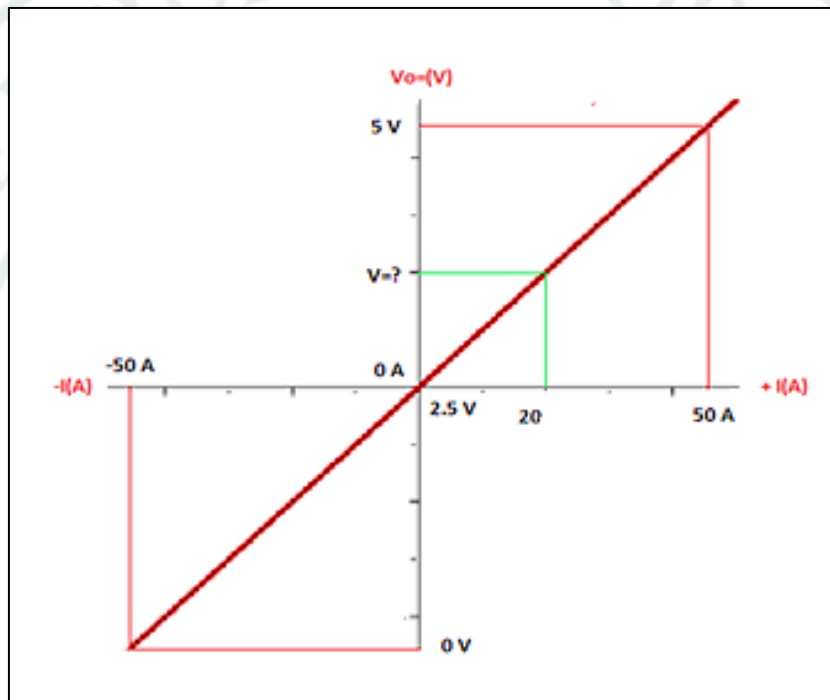
- Sensor ACS 756-050 A tiene una pérdida de potencia ultra baja, con una resistencia interna del conductor de 100 $\mu\Omega$.
- Tensión de aislamiento mínima de 3k Vrms de pines 1-3 a pines 4-5
- Función de operación de 3 V a 5 V Suministro único.
- Tiempo de salida de 3 μs en respuesta de la corriente de entrada.
- Sensibilidad de salida de 20 o 40 m V/A.
- Los voltajes de salida son proporcionales a las corrientes CA o CC.
- Voltaje de compensación de salida extremadamente estable.

3.3.3 Análisis del circuito seleccionado

Empezaremos hallando una forma matemática del comportamiento del sensor.

Figura 13

Comportamiento del Sensor AC 756-050^a



Nota: Elaboración propia, comportamiento de sensor AC 756-050^a

Utilizamos la gráfica como referencia para obtener la pendiente dada por:

$$m = \frac{y_1 - y_0}{x_1 - x_0}$$

Donde:

$$Y_1 = 50 \text{ A}$$

$$X_1 = 50 \text{ A}$$

$$X_0 = 0 \text{ A}$$

$$y_o = 2.5 A$$

Reemplazando:

$$m = \frac{5 - 2.5}{50 - 0} = \frac{2.5 [v]}{50 [A]}$$

$$m = 0.05 \frac{[v]}{[A]}$$

Ahora reemplazamos en la ecuación de la recta la pendiente y el punto de paso.

$$y - y_o = m(x - x_o)$$

$$-2.5 = 0.05(x - 0)$$

$$y = 0.05x + 2.5$$

De esta ecuación podemos deducir que:

$$y = 0.05x + 2.5$$

$$V_o = 0.05I_{in} + 2.5$$

Deducimos que 0.05I_{in} es la sensibilidad del sensor seleccionado.

Ahora reemplazamos una corriente que nos entregara el prototipo como dato.

$$I_{in} = 20 A \quad v_o = 0.05I_{in} + 2.5$$

$$v_o = 0.05(20) + 2.5$$

$$v_o = 3.5 V$$

Al obtener el voltaje de salida V_o = 3.5 V empezamos a diseñar la siguiente etapa, pero por la variación de temperatura que dará el sensor de corriente, el mismo fabricante

recomienda trabajar con una sensibilidad de 0.04 como se muestra en la tabla por lo que se recalculará el voltaje de salida dada.

Tabla 10

Especificaciones Técnicas del Sensor

<i>Características</i>	<i>Simbolo</i>	<i>Condiciones</i>	<i>Min</i>	<i>Typ</i>	<i>Max</i>	<i>Units</i>
<i>Sensibilidad</i>	<i>Sens TA</i>	<i>Half Scale applied for 5 ms</i> <i>Ta=25°C</i>	-	40	-	<i>mV/A</i>
	<i>SensTOP</i>	<i>Half scale applied for 5 ms</i>	37.2	-	42.8	<i>mV/A</i>

Nota: ACS766KCB-050 Características Top=-40°C a 125°C Vcc=5V

Para una sensibilidad de 0.04:

$$v_o = 0.04I_{in} + 2.5$$

$$v_o = 0.04(20) + 2.5$$

$$v_o = 3.3 \text{ V}$$

Por lo tanto, utilizaremos una sensibilidad de 0.04v para $I_{in} = 20 \text{ A}$ obteniendo el valor de salida $V_o = 3.3 \text{ V}$.

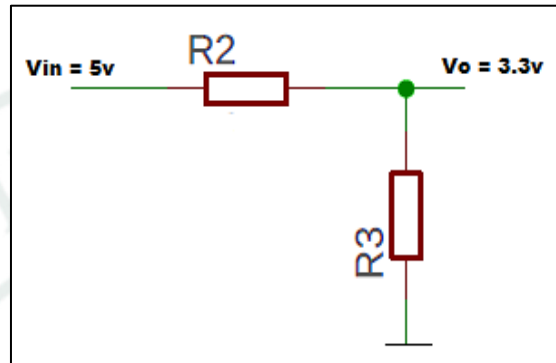
El voltaje de salida del sensor de corriente ingresara a la etapa de control para poder calcular el MPPT (punto de máxima potencia) y poder tener la máxima eficiencia.

Es importante tener en cuenta que al existir alguna subida de corriente que llega a los 50 A se obtendría una salida de 5 V lo que podría dañar el conversor análogo digital de la etapa de control. Incluso para la corriente de trabajo de 20 A tenemos una salida de 3.3 V, voltaje que supera lo admitido por el ADC de la etapa de control que es de 3.3 V.

Divisor de tensión para acondicionar los 5 V a 3.3 V

Figura 14

Divisor de Voltaje



Nota: Elaboración propia, diseño divisor de voltaje

Utilizando la fórmula del divisor de voltaje tenemos:

$$3.3 V = \frac{5V \times R_2}{R_1 + R_2}$$

$$V_o = \frac{V_{in} \times R_2}{R_1 + R_2}$$

$$3.3(R_1 + R_2) = 5V \times R_2$$

$$3.3R_1 + 3.3R_2 = 5R_2$$

$$R_2 = 1.49R_1$$

Nos damos un valor comercial de resistencia $R_2 = 10 K$ y reemplazamos en la ecuación

$$R_2 = 1.49R_1$$

$$10K = 1.49R_1$$

$$R_1 = 5.15 K\Omega$$

Se tiene dos resistencias comerciales cercanos al valor $R1=5.15\text{ K}\Omega$

Valores comerciales $5.6\text{ K}\Omega$

Recalculando con los dos valores comerciales

Valor de resistencia $R=5.1\text{ K}\Omega$

$$V_o = \frac{V_{in} \times R2}{R1 + R2}$$

$$V_o = \frac{5 \times 10\text{K}}{10\text{K} + 5.1\text{K}}$$

$$V_o = 3.3122 < 3.3\text{ V}$$

Valor de resistencia $R=5.6\text{ K}\Omega$

$$V_o = \frac{V_{in} \times R2}{R1 + R2}$$

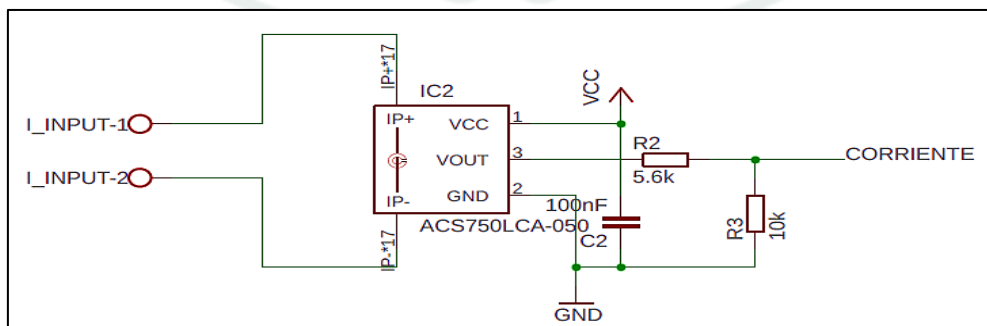
$$V_o = \frac{5 \times 10\text{K}}{10\text{K} + 5.6\text{K}}$$

$$V_o = 3.205 < 3.3\text{ V}$$

Uniendo el sensor de corriente con el circuito reductor obtendremos un voltaje proporcional a la entrada que podrá ingresar a la etapa de control sin dañar el convertidor analógico digital del microcontrolador.

Figura 15

Sensor de Corriente con Circuito Reductor



3.4 Acondicionador de RTD

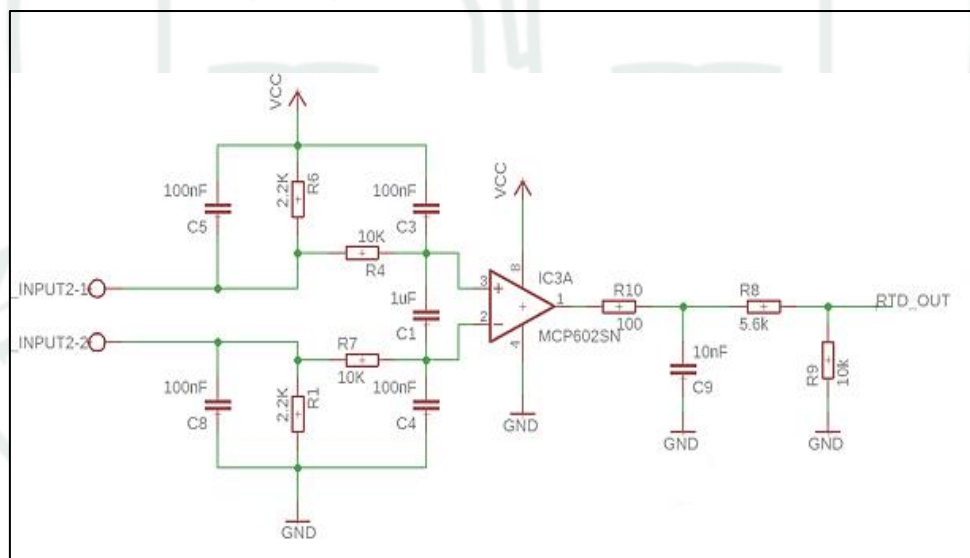
El acondicionador RTD (Resistance Temperature Detector) es un dispositivo que se utiliza para medir la temperatura. Este tipo de acondicionador convierte la variación de resistencia eléctrica del RTD, que cambia con la temperatura, en una señal estándar (como 4-20 mA o 0-10 V) que puede ser utilizada por sistemas de control y monitoreo, ya que permiten una medición precisa y confiable de la temperatura en procesos críticos.

3.4.1 Diseño de circuito de acondicionador de RTD

Este circuito acondicionador de RT sirve para ajustar los niveles de temperatura en ambos canales de la operación de suministro de salida.

Figura 16

Circuito acondicionador de RTD



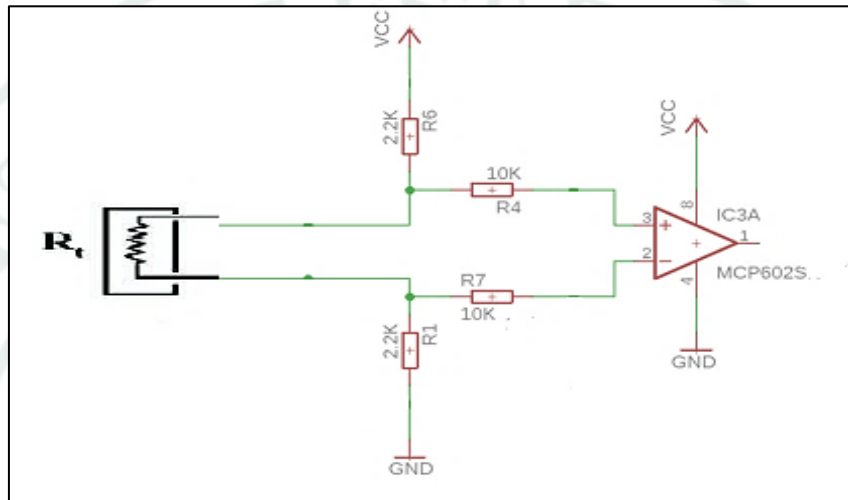
Nota: Elaboración propia, circuito acondicionador de RTD

3.4.2 Análisis de circuito diseñado

Análisis del circuito en DC

Figura 17

Circuito acondicionador de RTD



Nota: Elaboración propia, circuito acondicionador de RTD

Se encuentra esta ecuación

Ecuación (1)

$$\frac{V_a - V_b}{RTD} + \frac{V_a - V_{CC}}{2.2K} + \frac{V_a - V^+}{10k} = 0$$

Ecuación (2)

$$\frac{\frac{V_b - V_a}{RTD(V_b - GND)}}{2.2K} + \frac{V_b - V^-}{10k} = 0$$

$$V^+ = V^-$$

Se desea encontrar lo siguiente:

$$V_{ab} = V_a - V_b$$

Se reemplaza en las ecuaciones superiores

$$\frac{Vab}{RTD} + \frac{Va - VCC}{2.2K} + \frac{Va - V +}{10k} = 0 \text{ Ecuación (1)}$$

$$\frac{-Vab}{RTD} + \frac{Vb - GND}{2.2K} + \frac{Vb - V -}{10k} = 0 \text{ Ecuación (2)}$$

Restamos ecuaciones 1 y 2

$$\frac{2Vab}{RTD} + \frac{Va - VCC}{2.2K} + \frac{Va - V +}{10k} = \frac{Vb}{2.2K} + \frac{Vb - V -}{10k}$$

$$\frac{2Vab}{RTD} + \frac{Va - VCC}{2.2K} + \frac{Va}{10k} - \frac{V +}{10k} = \frac{Vb}{2.2K} + \frac{Vb}{10k} - \frac{V -}{10k}$$

$$\frac{2Vab}{RTD} + Va \left(\frac{1}{2.2K} + \frac{1}{10k} \right) = Vb \left(\frac{1}{2.2K} + \frac{1}{10k} \right) + \frac{VCC}{2.2K}$$

$$Va \left(\frac{1}{2.2K} + \frac{1}{10k} \right) - Vb \left(\frac{1}{2.2K} + \frac{1}{10k} \right) = + \frac{VCC}{2.2K} - \frac{2Vab}{RTD}$$

$$(Va - Vb) \left(\frac{1}{2.2K} + \frac{1}{10k} \right) = + \frac{VCC}{2.2K} - \frac{2Vab}{RTD}$$

$$Vab = Va - Vb$$

$$Vab \left(\frac{1}{2.2K} + \frac{1}{10k} \right) + \frac{2Vab}{RTD} = \frac{VCC}{2.2K}$$

$$Vab \left(\frac{10k + 2.2k}{2.2K * 10k} + \frac{2}{RTD} \right) = \frac{VCC}{2.2K}$$

$$Vab \left(\frac{RTD(10k + 2.2k) + 2 * 2.2K * 10k}{2.2K * 10k * RTD} \right) = \frac{VCC}{2.2K}$$

$$Vab \left(\frac{RTD(12.2) + 44}{10 * RTD} \right) = Vcc$$

La ecuación del voltaje de salida del opam es la siguiente

$$V_{ab} = \frac{V_{cc} * RTD}{RTD(1.22) + 4.4}$$

Reemplazamos valores

$$V_o(T^{\circ}C) = \frac{V_{cc} * R(T^{\circ}C)}{R(T^{\circ}C)(1.22) + 4.4}$$

Entonces se tiene que la ecuación del voltaje de salida se encuentra en función a la variación de la resistencia del pt100 con respecto a la temperatura

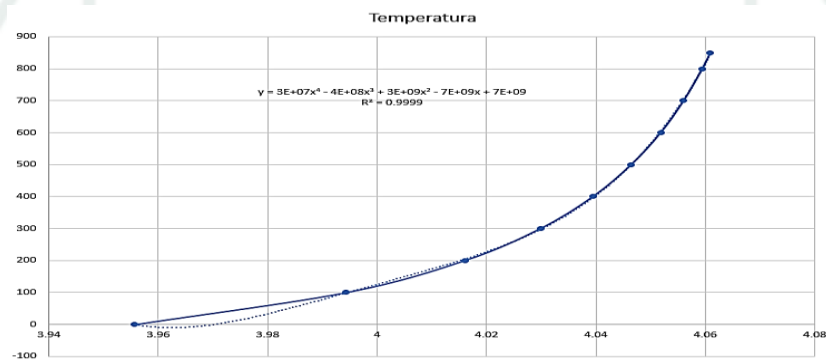


Tabla 11

Voltaje de Salida en Función a la Variación de la Resistencia

Temperatura (°C)	Temperatura (°F)	Resistencia (Ohmios)
-200	-328	18.52
-100	-148	60.25
0.00	32	100.00
100	212	138.50
200	292	175.85
300	572	212.05
400	752	247.09
500	932	313.70
600	1112	313.70
700	1292	345.28
800	1472	175.70

Nota: Tabla de voltaje de salida en función a la variación de la resistencia PT100

$$V_o(T^{\circ}C) = \frac{V_{cc} * R(T^{\circ}C)}{R(T^{\circ}C)(1.22) + 4.4}$$

$$V_{ab} = \frac{V_{cc} * RTD}{RTD(1.22) + 4.4}$$

$$\left(\frac{RTD(1.22) + 4.4}{RTD} \right) = \frac{V_{cc}}{V_{ab}}$$

$$\frac{RTD(1.22) + 4.4}{RTD} = \frac{V_{cc}}{V_{ab}}$$

$$\frac{RTD(1.22)}{RTD} + \frac{4.4}{RTD} = \frac{V_{cc}}{V_{ab}}$$

$$1.22 + \frac{4.4}{RTD} = \frac{V_{cc}}{V_{ab}}$$

$$\frac{V_{cc}}{V_{ab}} - 1.22 = \frac{4.4}{RTD}$$

$$\frac{V_{cc} - 1.22V_{ab}}{V_{ab}} = \frac{4.4}{RTD}$$

$$RTD(ohm) = \frac{4.4 * V_{ab}(V)}{V_{cc}(v) - 1.22V_{ab}(v)}$$

$$y = 3E + 07x^4 - 4E + 08x^3 + 3E + 09x^2 - 7E + 09x + 7E + 09$$

$$T(^{\circ}C) = 3E + 07(V_o)^4 - 4E + 08(V_o)^3 + 3E + 09(V_o)^2 - 7E + 09(V_o) + 7E + 09$$

3.5 Sensor de vibración

Sensores de vibración son acelerómetros que permite trabajar en frecuencias altas, parámetro que es donde se produce principalmente la vibración, estas frecuencias y en

consecuencia estos acelerómetros se utilizan para diferentes aplicaciones como es el control preventivo o mantenimiento preventivo en maquinaria.

3.5.1 Selección de sensor de vibración

A. Sensor PC420VR-05

Los sensores de la serie PC420V de Wilcoxon proporcionan una salida de 4-20 mA proporcional a vibración de velocidad, lo que permite una tendencia continua de la vibración general de la máquina.

Estos datos de tendencias alertan a los usuarios sobre las condiciones cambiantes de la máquina y ayudan a guiar el mantenimiento al priorizar la necesidad de servicio. La elección de RMS le permite elegir el sensor que mejor se adapta a sus necesidades.

Guía de selección de modelo P PC420Vx-yy

x (4-20 mA output type)	yy (4-20 mA full scale)
R = RMS output, velocity	05 = 0.5 ips (12.8 mm/sec)
P = calculated peak output, velocity	10 = 1.0 ips (25.4 mm/sec)
	20 = 2.0 ips (50.8 mm/sec)
	30 = 3.0 ips (76.2 mm/sec)
	50 = 5.0 ips (127 mm/sec)

Especificaciones:

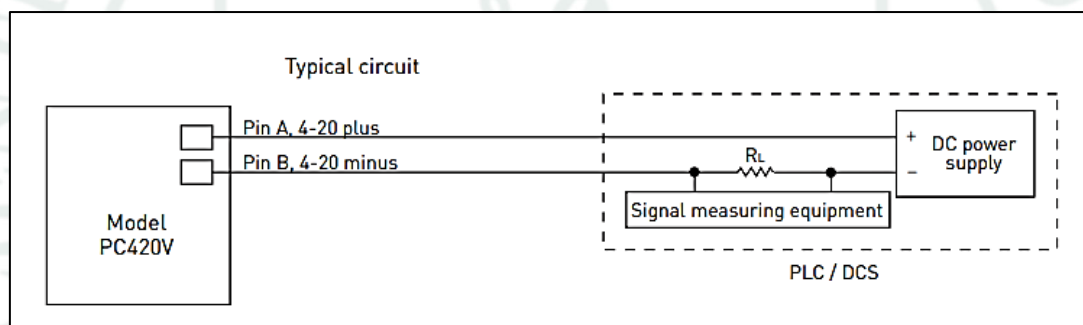
Escala completa, 20 mA, +-5%	Tabla superior
Frecuencia	+/-10% 10 Hz – 1.0 kHz
+/-3 db	3.5 Hz – 2.0 kHz
Repetibilidad	+/-2%
Sensibilidad transversal, max	5%
Voltaje	12 -30 VDC

Resistencia bucle a 24 VCC	700 Ohm
Tiempo de encendido 4-20mA	30 seg
Rango de temperatura	-40° a +105°C
Límite de vibración	250 g peak

3.5.2 Circuito típico

Figura 18

Circuito típico de sensor PC420VR-05



Nota: Elaboración propia, diseño circuito tipico de sensor

La vibración del sensor uniaxial mediante una señal analógica de 4 a 20 mA. Las muestras se obtienen con un intervalo de 60us, y el cálculo del valor de vibración se realiza utilizando un algoritmo basado en la raíz cuadrática media (RMS). El equipo tiene las siguientes especificaciones de medición:

- Ancho de banda: 3.5 Hz a 2.0 kHz.
- Rango de vibración: 0 a 250 g pico.

3.5.3 Cálculo del valor RMS

El valor RMS es una medida que representa el equivalente de una señal periódica como una corriente continua, lo que facilita su interpretación. Para calcular el RMS de la vibración a partir de las muestras adquiridas, se utiliza la siguiente fórmula:

$$RMS = \sqrt{\frac{1}{N} \sum_{i=1}^N a_i^2}$$

Donde:

RMS: Es el valor de la señal.

a_i : es la i -ésima muestra de la señal.

N: Es el número total de muestras en la ventana de tiempo analizada.

PROCESO DE CÁLCULO:

A. Adquisición de muestras:

La muestra de la señal analógica se convierte a valores digitales mediante un ADC - Convertidor Analógico a Digital.

La muestra equivale a una frecuencia de muestreo de 100 Hz. Este valor debe ser suficiente para capturar el rango de frecuencia de interés (3.5 Hz a 2.0 kHz) según datasheet de sensor de vibración cumpliendo con el teorema de Nyquist.

B. Selección de ventana temporal:

Se define como una ventana de tiempo para acumular NNN muestras.

Por ejemplo, para un análisis de 1 seg, se necesitarían:

$$N=100 \quad N = 100 \quad N=100 \text{ muestras.}$$

C. Cálculo del valor cuadrático:

Cada muestra (x_i) se eleva al cuadrado: x_i^2 .

D. Promedio de valores cuadráticos

Se suman todos los valores cuadráticos y se tiene que dividir entre el número total de muestras.

$$RMS = Mc = \sqrt{\frac{\sum (f \cdot x_i^2)}{N}}$$

E.

Se calcula la raíz cuadrada del promedio para obtener el valor RMS

3.5.4 Interpretación de la señal de 4-20 mA

El equipo convierte el valor RMS de vibración en una señal de corriente de 4-20 mA, donde:

- 4 mA representa el nivel mínimo de vibración (0 g).
- 20 mA representa el nivel máximo de vibración (250 g pico).

La relación entre la corriente medida (I) y el valor RMS (x_{RMS}) se calcula de manera lineal según la siguiente fórmula:

$$I = 4 + \frac{16}{250} \cdot x_{RMS}$$

Donde:

- I: Corriente de salida (mA).
- x_{RMS} : Valor RMS de vibración (g).

Este enfoque asegura una representación precisa y proporcional del nivel de vibración del sensor, lo que facilita su interpretación en sistemas de monitoreo o control.

3.6 Sensor de presión

En las principales aplicaciones del sensor se tiene la medición de presión en tuberías de agua, nivel de columna por presión hidrostática, sistemas de presión constante, etc.

3.6.1 Selección de sensor de presión

Sensor de seleccionado HK3025 convierte presión hidráulica o neumática en una señal analógica, que se calibra de forma lineal. Seleccionado para ambientes industriales donde se requiere precisión y durabilidad.

3.6.2 Especificaciones técnicas

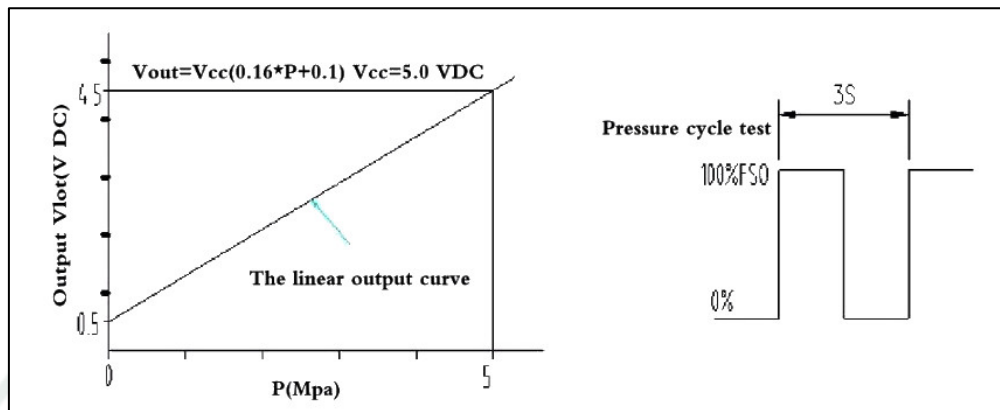
- Rango de presión: 0 – 5.0 MPa (50 bar)
- Voltaje de salida: 0.5 – 4.5 VDC
- Alimentación: 5 V DC
- Precisión típica: $\pm 1\%$ FS
- Salida: Señal analógica proporcional a presión
- Rosca: 1/4" NPT
- Conexión: 3 pines (VCC, GND, OUT)

3.6.3 Circuito típico

La salida del sensor es un voltaje analógico que varía linealmente desde 0.5 V para 0 MPa hasta 4.5 V para 5 Mpa.

Figura 19

Circuito típico de sensor HK3025



Nota: Elaboración propia, circuito típico de sensor

Ecuaciones en el microcontrolador

En el código:

```
Presion_scale = mapFloat(PresionPVC, 0, 4095, 0, 0.5);
```

Que usando la función mapFloat se traduce como:

$$\begin{aligned} \text{Presión (MPa)} &= (ADC - 04095 - 0) \times 0.5 \\ &= \left(\frac{\text{ADC} - 0}{4095} - 0 \right) \times 0.5 \\ \text{Presión (MPa)} &= (4095 - 0ADC - 0) \times 0. \end{aligned}$$

3.7 Sensor de caudal

3.7.1 Selección de sensor de caudal

Sensor de flujo seleccionado TF-S201 emplea un rotor interno y un sensor efecto hall para emitir pulsos digitales proporcionales al caudal.

3.7.2 Especificaciones técnicas

- Voltaje de operación: 5 V a 18 V DC
- Corriente típica: 15 mA a 5 V
- Rango de flujo: 1 a 30 L/min
- Frecuencia de salida:

$$\begin{aligned} \text{Frecuencia (Hz)} &= 7.5 \times \text{Flujo (L/min)} \\ &= 7.5 \times \text{Flujo (L/min)} \\ &= 7.5 \times \text{Flujo (L/min)} \end{aligned}$$

- Volumen por pulso: 2.25 mL/pulso
- Presión máxima: 1.75 MPa
- Salida: Pulso digital (cuadrado)
- Rosca: 1/2" NPS
- Material del rotor: Acero inoxidable

En el código del RM46L852, el flujo se calcula como:

$$\begin{aligned} \text{Flujometro_scale} &= \frac{\text{Flujometro}}{7.5} \\ &= \frac{\text{Flujometro}}{7.5} \\ &= 7.5 \text{Flujometro} \end{aligned}$$

Además:

$$\text{if (Flujometro_scale} < 10)$$

3.8 Adaptador de 4 A 20 MA

El módulo de corriente a voltaje se convierte linealmente en señales de corriente de 0 a 25 mA en señales de voltajes de 0 V a 3 V.

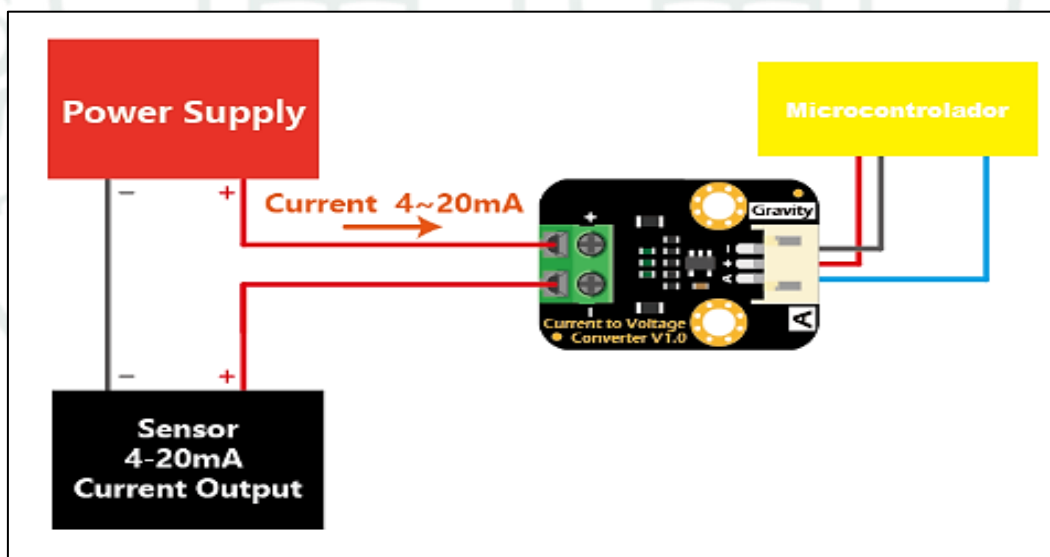
El adaptador o sensores suelen tener una salida de corriente de 4 a 20 mA, el adaptador de corriente a voltaje podemos leer las señales de salida normalmente corrientes inferiores a 4mA para diagnósticos de fallas y para corrientes superiores a 20 mA para detección de sobrecarga.

Necesitamos un módulo con un amplio rango de detección de 0 a 25 mA teniendo una precisión de 0.1% y un amplificador operacional.

3.8.1 Diagrama de bloques específico

Figura 20

Diagrama de Bloques Adaptador de 4 a 20mA



Nota: Elaboración propia

3.8.2 Selección de circuito de 4 A 20 Ma

A. Módulo de corriente SEN0262

El módulo de corriente a voltaje SEN0262 convierte linealmente en señales de corriente de 0 a 25 mA en señales de voltajes de 0 V a 3V.

Especificaciones:

Voltaje de suministro: 3.3V a 5.5V

Rango de detección: 0 a 25 mA CC

Precisión de medición: $\pm 0.5\%$

Fondo de escala ADC: 16 bits

Resistencia de terminación: $120\ \Omega$

Tipo de conector: PH2.0-3P

Descripción general del tablero:

GND --- Power GND (0 V)

VCC --- Power VCC (3.3 a 5.5 V)

SIGNAL --- Voltage Signal Output

I+ --- Current Input

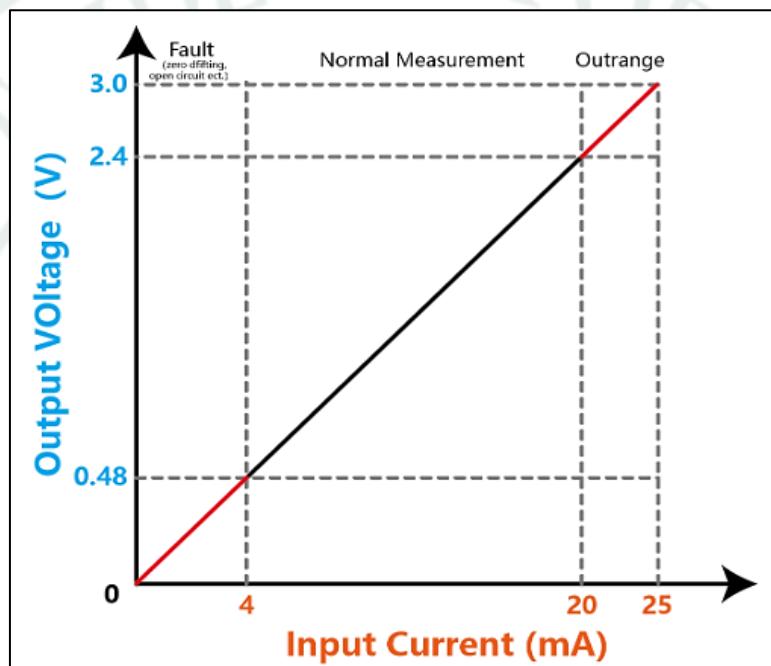
I- --- Current Output

Descripción de Grafica de Señal:

Se muestra la señal de corriente de 4 a 20 mA con este módulo, lo cual vemos que la corriente de entrada es lineal con el voltaje de salida, como se muestra en la gráfica.

Figura 21

Descripción de la salida de voltaje sobre salida



Nota: Elaboración propia, descripción de salida de voltaje sobre salida

3.9 Relay de salida

Son módulos ideales para conmutar cargas de corrientes alternas que están conectadas a una red eléctrica. Soportando cargas de hasta 10 A y 250 VAC recomendable usar niveles de tensión por debajo de estos límites.

Este módulo es capaz de soportar y controlar varios equipamientos de alta corriente durante un tiempo determinado, controlado por varios microcontroladores.

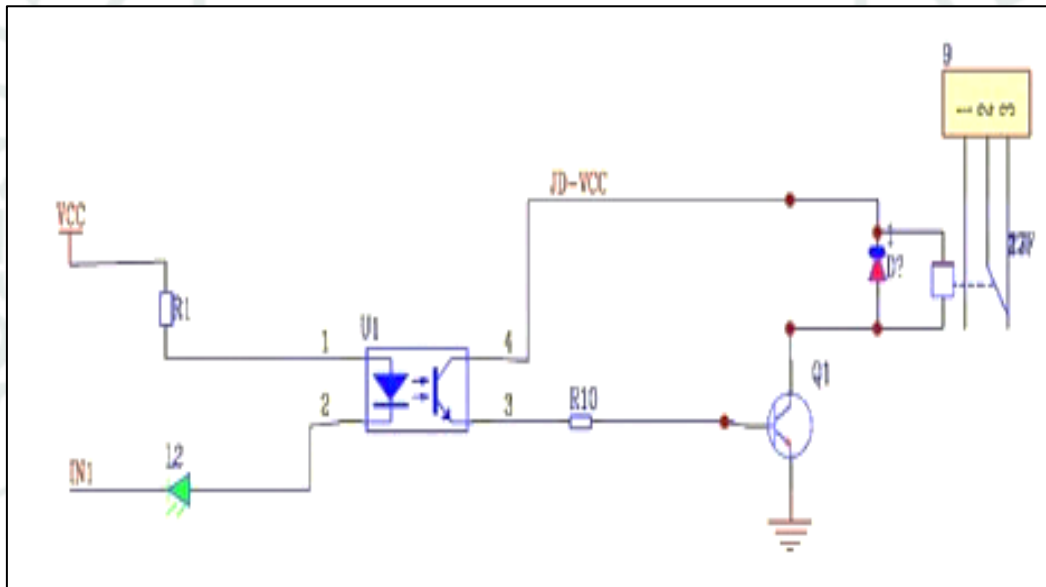
3.9.1 Diagrama de bloques específico

El relé está compuesto de una bobina que está conectada a una corriente. Cuando la bobina se activa, se produce campo electromagnético haciendo contacto con el relé que está normalmente abierto, se cierra y permite el paso de corriente.

Al cortar la corriente de la bobina el campo electromagnético desaparece y el contacto del relé vuelve abrirse de inmediato, dejando sin corriente al circuito.

Figura 22

Comportamiento Relay



Nota: Elaboración propia, comportamiento de sistema Relay

3.9.2 Selección de relay de salida

A. Módulo de cuatro canales relé

Este módulo es capaz de soportar y controlar varios equipamientos de alta corriente durante un tiempo determinado, controlado por varios microcontroladores.

Voltaje de Operación	5V DC
Señal de Control	TTL (3.3V o 5V)
N° de Relays (canales)	4 CH
Modelo Relay	SRD-05VDC-SL-C
Máxima Capacidad	10A/250VAC – 10A/30VDC
Máxima Corriente	10A (NO), 5A (NC)
Tiempo de Acción	10ms / 5ms
Para Activar Salida NO	0 Voltios
Entradas	Optoacoplador

3.10 Módulo de wifi

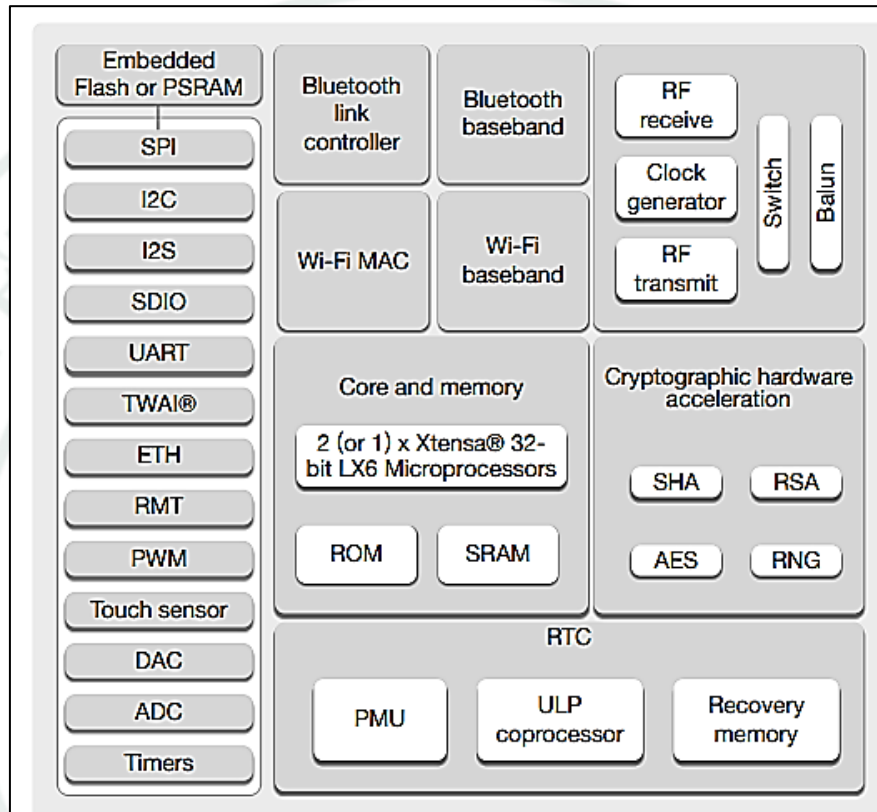
Se usará una tarjeta que posee una conectividad Wi-Fi y Bluetooth, además de un procesador de 32 Bits para prototipar rápidamente proyectos IoT (Internet de las cosas). Permitiendo la comunicación inalámbrica entre los dispositivos garantizando una transmisión y recepción de señales.

La transmisión y recepción de señales proporcionan una comunicación efectiva de datos a varios metros de distancia con un funcionamiento estable del dispositivo.

3.10.1 Diagrama de bloques específicos

Figura 23

Módulo Wi-Fi ESP32



Nota: Elaboración propia, diagrama de modulo Wi-Fi ESP32

3.10.2 Elección de módulo de WI-FI

A. Módulo de WI-FI ESP32

Módulo potente para generar rápido proyectos con IoT (Internet de las Cosas) integra una placa SoM ESP-WROOM-32 que tiene como base al SoC ESP 32 y un conversor USB- Serial CP2102 necesario para programar por USB el ESP32 también tiene leds indicadores y reguladores de voltaje. Tiene una capacidad alta de comunicación

y de procesamiento permitiendo utilizar diversos protocolos de comunicación inalámbrica. Wifi y Bluetooth en este caso utilizaremos comunicación inalámbrica por Wifi.

Especificaciones Técnicas

Voltaje de Alimentación	5V DC (USB)
Voltaje de Entradas/Salidas	3.3V DC
Frecuencia de Reloj	240Mhz
Wifi	802.11 b/g/n/e/i
Bluetooth	v4.2 BR/EDR
Placa	ESP32
SoM	ESP-WROOM-32
SoC	ESP32

3.11 Pantalla LCD

Es un pequeño dispositivo que cuenta con dos filas de dieciséis caracteres cada una que sirve para mostrar información alfanumérica, con la finalidad de visualizar algunos datos obtenidos en el sensado realizado.

3.11.1 Selección de pantalla LCD 2X16

La pantalla seleccionada tiene 16 pines y el enfoque de programación también es el mismo dejando a elección al programador.

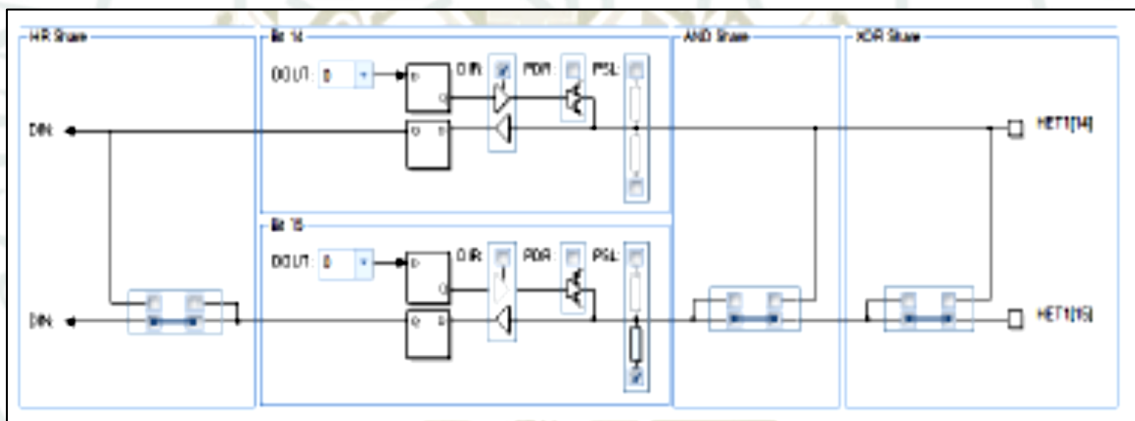
3.12 Microcontrolador LAUNCHXL2-RM46 - HERCULES

El microcontrolador RM46L852 cuenta con periféricos para aplicaciones basadas en control en tiempo real. Tiene 44 pines de I/O, 7 mejorados módulos de PWM con 14 salidas, y dos convertidores de analógico a digital de 12 bits (ADC) PUBLIC - Unrestricted Access que admiten hasta 24 entradas, como la AND y XOR. Las entradas y salidas se configuran en el programa HALCoGen. (Texas Instrument, 2021)

Para nuestro equipo, el módulo de I/O para la pantalla LCD utiliza 24 pines para el control de la pantalla.

Figura 24

Configuración de Entradas y Salidas



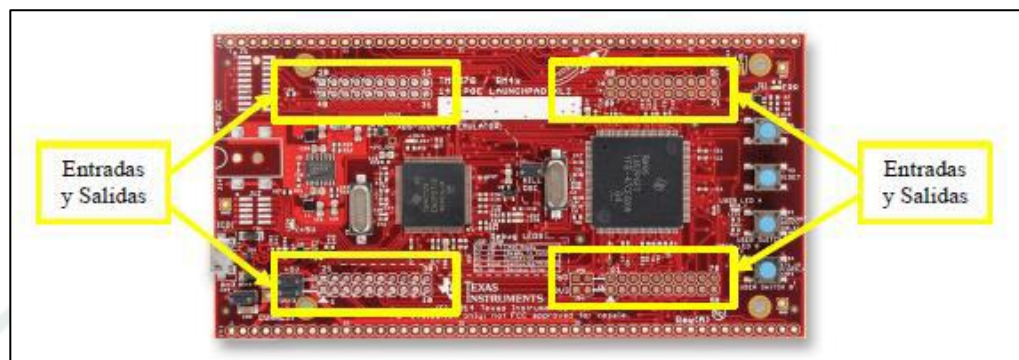
Nota: *Elaboración propia, configuración de entradas y salidas*

Características de seguridad integradas y periféricos como dos ADC de 12 bits, temporizadores programables de gama alta, periféricos de control de motor (eQEP, eCAP), Ethernet e interfaz de comunicación serie.

Los MCU Hércules RM ayudan a reducir el esfuerzo de desarrollo de las aplicaciones de seguridad funcional IEC 61508.

Figura 25

Entradas y salidas del microcontrolador



***Nota:** Elaboración propia, entradas y salidas del microcontrolador*

3.13 Bomba de agua

Una bomba de agua es un dispositivo mecánico que es utilizado comúnmente en la agricultura y la industria para suministrar aguas a áreas a donde el flujo de agua puede ser escaso. Contempla motor, carcasa o armazón, impulsor, eje impulsor, panel de control, entradas y salidas, rodamiento y sello.

Especificaciones Técnicas:

Voltaje de Alimentación	220 V
HP	1 HP
Caudal máximo	5 / 40 l/min
RPM	3450 r/min
HZ	60 Hz
Corriente	2.5 A
Protección	IP44

3.14 Cálculo de grosor de pistas

3.14.1 Fórmulas y consideraciones:

El ancho de la pista afecta directamente la cantidad de corriente que puede transportar. El espesor del cobre (normalmente 1 oz/ft² o 35 µm para una capa estándar de cobre en placas PCB).

El aumento de temperatura máximo permitido para la pista (generalmente se permite un aumento de 10-20°C). Para una placa estándar de 1 oz/ft² de espesor, se puede usar la siguiente fórmula empírica para calcular la capacidad de corriente de una pista (según IPC-2221):

$$I = k \cdot (W)^b \cdot (T)^c$$

Donde:

I = Corriente máxima en amperios (A)

W = Ancho de la pista en milésimas (mili)

T = Espesor de la pista en milésimas (para 1 oz de cobre, es 1.37 mili)

k,b,c = Constantes que dependen del tipo de pista (interna o externa) y del aumento de temperatura.

Cálculo de Pistas de Potencia (0.024 pulgadas)

Convertimos a mili:

Ancho de pista = 0.024 pulgadas x 1000 = 24 mili.

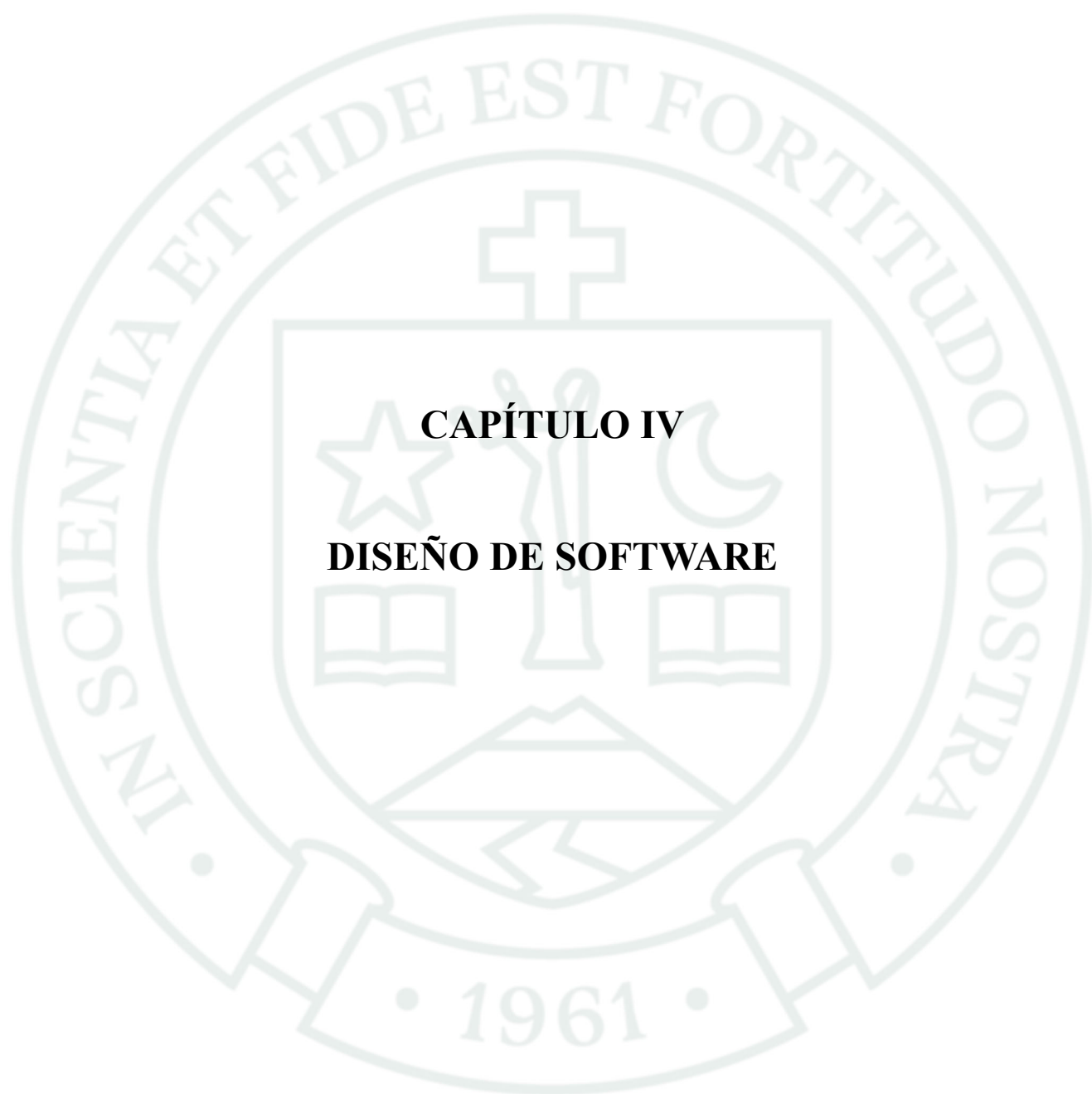
Para pistas externas con un aumento de temperatura de 10 °C a 20 °C, la corriente que puede transportar una pista de 24 mil a 35 μm de espesor está entre 1.5 y 2 A, lo que es adecuado para tu aplicación de 1-2 A.

Cálculo de Pistas Digitales (0.016 pulgadas $\times$ 1000 = 16 mili)

Para las señales digitales con un ancho de pista de 0.016 pulgadas = 16 mil, las corrientes que manejarás normalmente son mucho menores (del orden de miliamperios). Esta pista será adecuada para señales de baja corriente y frecuencia.

Conclusión:

- Las pistas de 24 mili parecen adecuadas para la corriente de 1-2 A.
- Las pistas de 16 mili pulgadas serán más que suficientes para señales digitales de baja corriente.



CAPÍTULO IV

DISEÑO DE SOFTWARE

En este capítulo se describe el diseño de software desarrollado para el prototipo IoT de monitoreo de bombas centrífugas, incluyendo la arquitectura del sistema, el flujo de datos, los algoritmos implementados y las herramientas utilizadas. Se presentan los diagramas que permiten comprender la lógica de funcionamiento, así como la integración entre los componentes hardware y software del sistema.

4.1 Diagrama de flujo

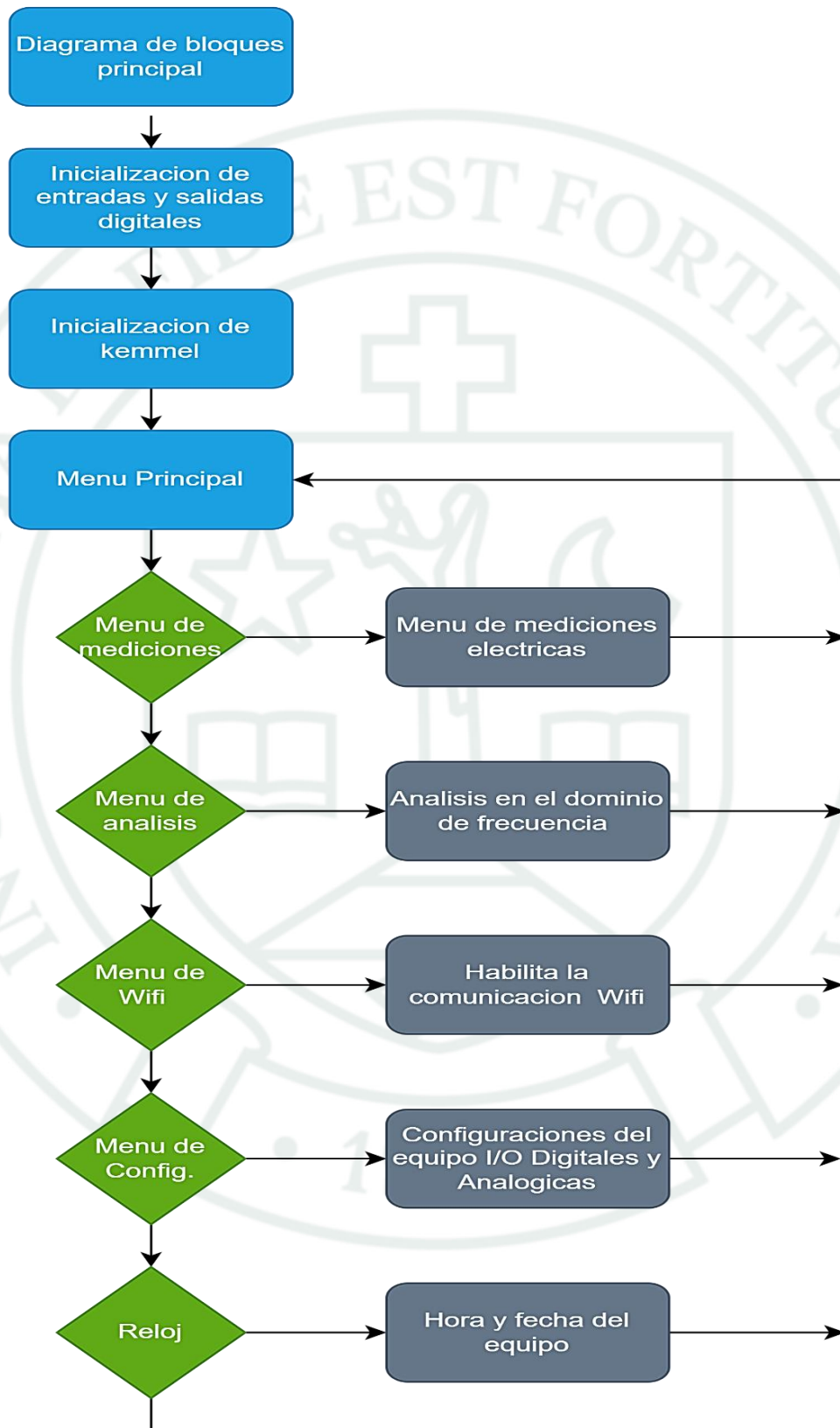
4.1.1 Diagrama de flujo principal

Diagrama principal de bloques inicialización de entradas y salidas digitales inicialización del Kernel, núcleo fundamental que se encarga que se encarga de conceder el acceso al hardware de forma segura al software de forma segura.

- Menú de principal:
- Menú de mediciones: Mediciones eléctricas se obtendrá los parámetros principales de operación con el sensor de voltaje, sensor de corriente, sensor de flujo, sensor de presión, sensor de vibración y sensor RTD.
- Menú de análisis: Análisis en dominio de frecuencia
- Menú de Wi-Fi: Habilidad de la comunicación Wifi
- Menú de configuración: Configuraciones del equipo I/O Digital y Análogas
- Reloj: Hora y fecha del equipo

Figura 26

Diagrama de flujo principal



En el diagrama de bloques general del diseño se presenta la integración de la bomba centrífuga de 1 HP, utilizada como prototipo para la recolección de los parámetros principales de operación. Este circuito cerrado permite simular condiciones de funcionamiento equivalentes a un sistema de escala industrial, facilitando la verificación del comportamiento de operación de la bomba.

El diagrama de bloques muestra la disposición de los sensores encargados de medir las variables críticas: sensor de voltaje, sensor de corriente, sensor de flujo, sensor de presión, sensor de vibración y sensor RTD para temperatura. Estos dispositivos fueron seleccionados considerando precisión, compatibilidad con el prototipo y robustez para entornos de prueba, instalándose estratégicamente en la línea hidráulica o directamente en el motor de la bomba, dependiendo del parámetro a medir.

Dentro del bloque de comunicaciones, se incorpora un módulo Wi-Fi, encargado de transmitir los datos adquiridos hacia la nube mediante el protocolo de transmisión TCP. La información también se almacena en una memoria SQL Server para garantizar la disponibilidad de registros. Los resultados de las mediciones se visualizan en tiempo real tanto en una pantalla LCD 20x4 incorporada en el prototipo, como en dashboards generados accesibles desde una computadora portátil conectada a la red.

Finalmente, el diagrama de flujo principal del sistema detalla la lógica de funcionamiento del prototipo. Este inicia con la adquisición de señales provenientes de los sensores, los datos son almacenados en la memoria local y enviados a la nube a través del módulo Wi-Fi, donde son organizados en bases de datos y representados en los dashboards.

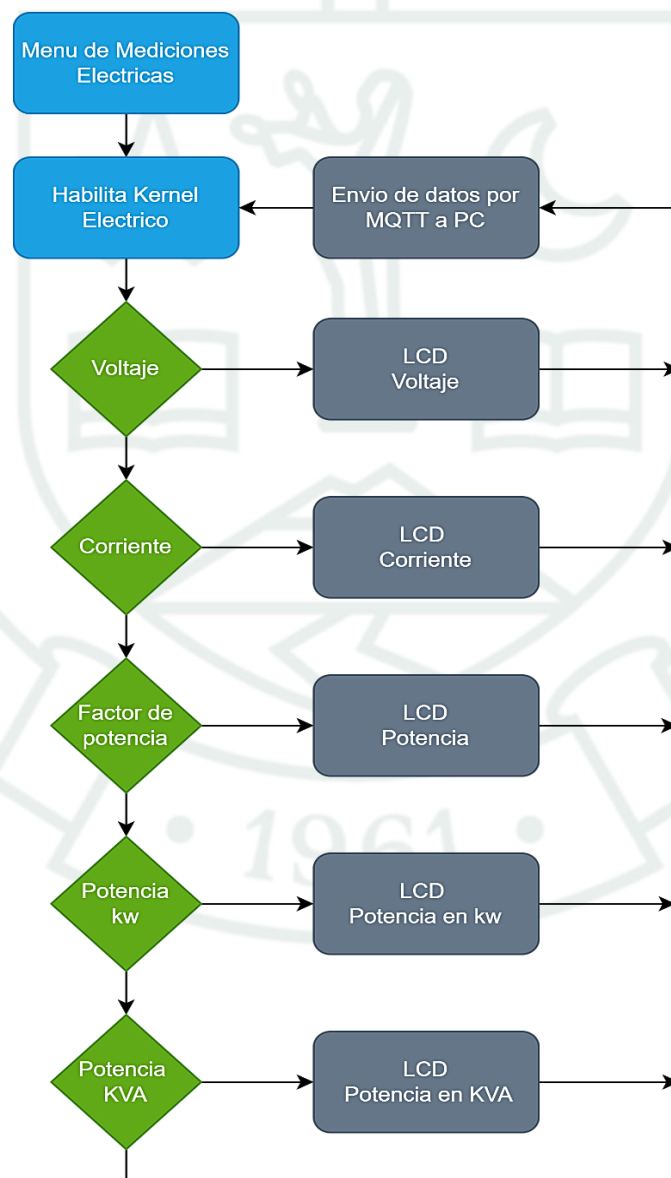
4.1.2 Diagrama de flujo menú de mediciones eléctricas

Inicialización del kernel, núcleo fundamental que se encarga de conceder el acceso al hardware de forma segura al software de forma segura.

Mediciones: Voltaje, Corriente, Factor de potencia, Potencia (Kw), Potencia (KVA), reflejados en la pantalla LCD y enviados de datos por Server SQL a PC.

Figura 27

Diagrama de flujo, menú de mediciones eléctricas



4.1.3 Diagrama de flujo de medición de análisis en dominio de la frecuencia

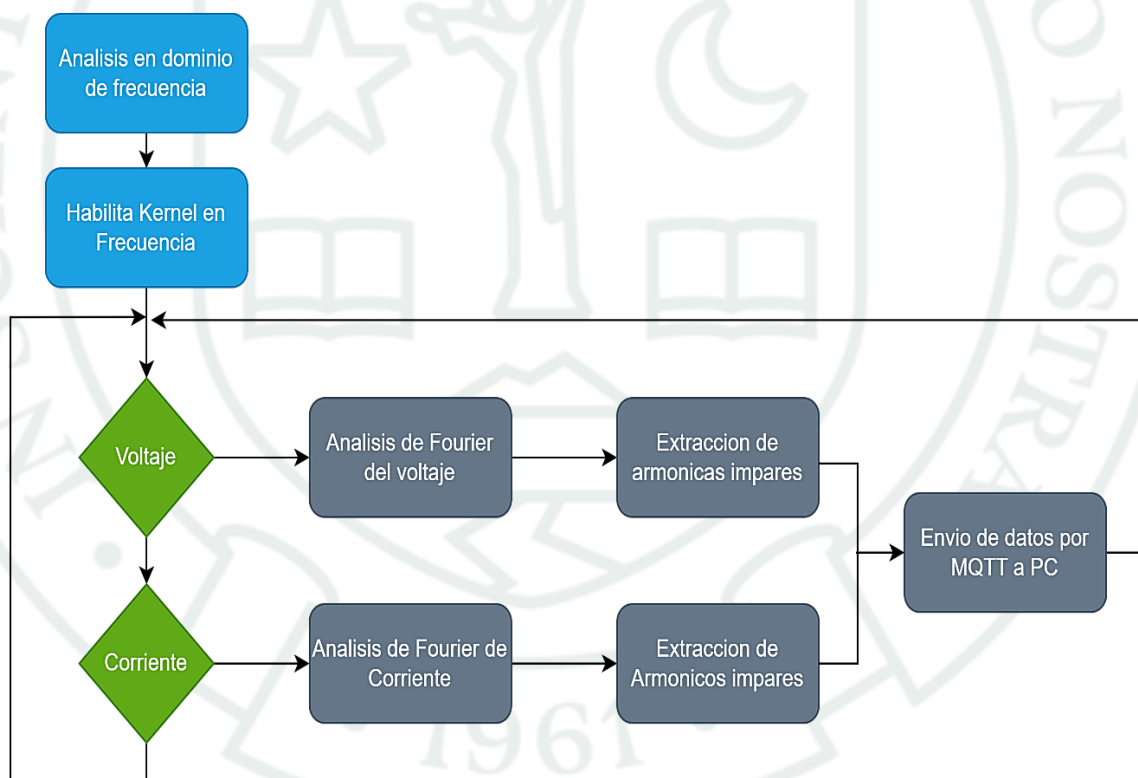
Inicialización del kernel, núcleo fundamental que se encarga de conceder el acceso al hardware de forma segura al software de forma segura.

Voltaje (V): Análisis de Fourier del voltaje, extrayendo las armónicas impares y enviando los datos por transmisión de puerto TCP a la PC

Corriente (A): Análisis de Fourier de la corriente, extrayendo las armónicas impares y enviando los datos por transmisión de puerto TCP a la PC

Figura 28

Diagrama de flujo de medición de análisis en dominio de la frecuencia



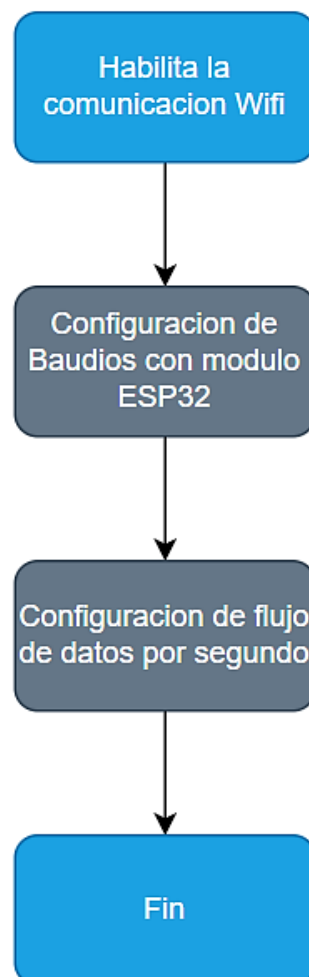
4.1.4 Diagrama de flujo de menú de WIFI

Inicialización del kernel, núcleo fundamental que se encarga de conceder el acceso al hardware de forma segura al software de forma segura.

Configuración de los baudios es el número de unidades de señal por segundo; con el módulo ESP32 puede contener varios bits.

Figura 29

Diagrama de flujo de menú de Wifi

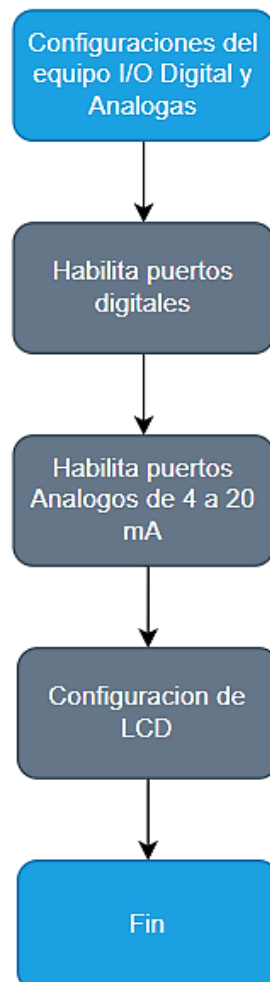


4.1.5 Diagrama de flujo menú de configuración i/o digital y análogos

- Habilitación de puertos digitales
- Habilitación de puertos análogos de 4 a 20 mA,
- Configuración hora y fecha
- Configuración del LCD

Figura 30

Diagrama de flujo de configuración I/O Digital y Analógicas



4.2 Programas

4.2.1 Sistema de adquisición y monitoreo en tiempo real

Este programa constituye el núcleo del software de monitoreo del prototipo, integrando adquisición, procesamiento, visualización y registro de datos en un solo entorno de red.

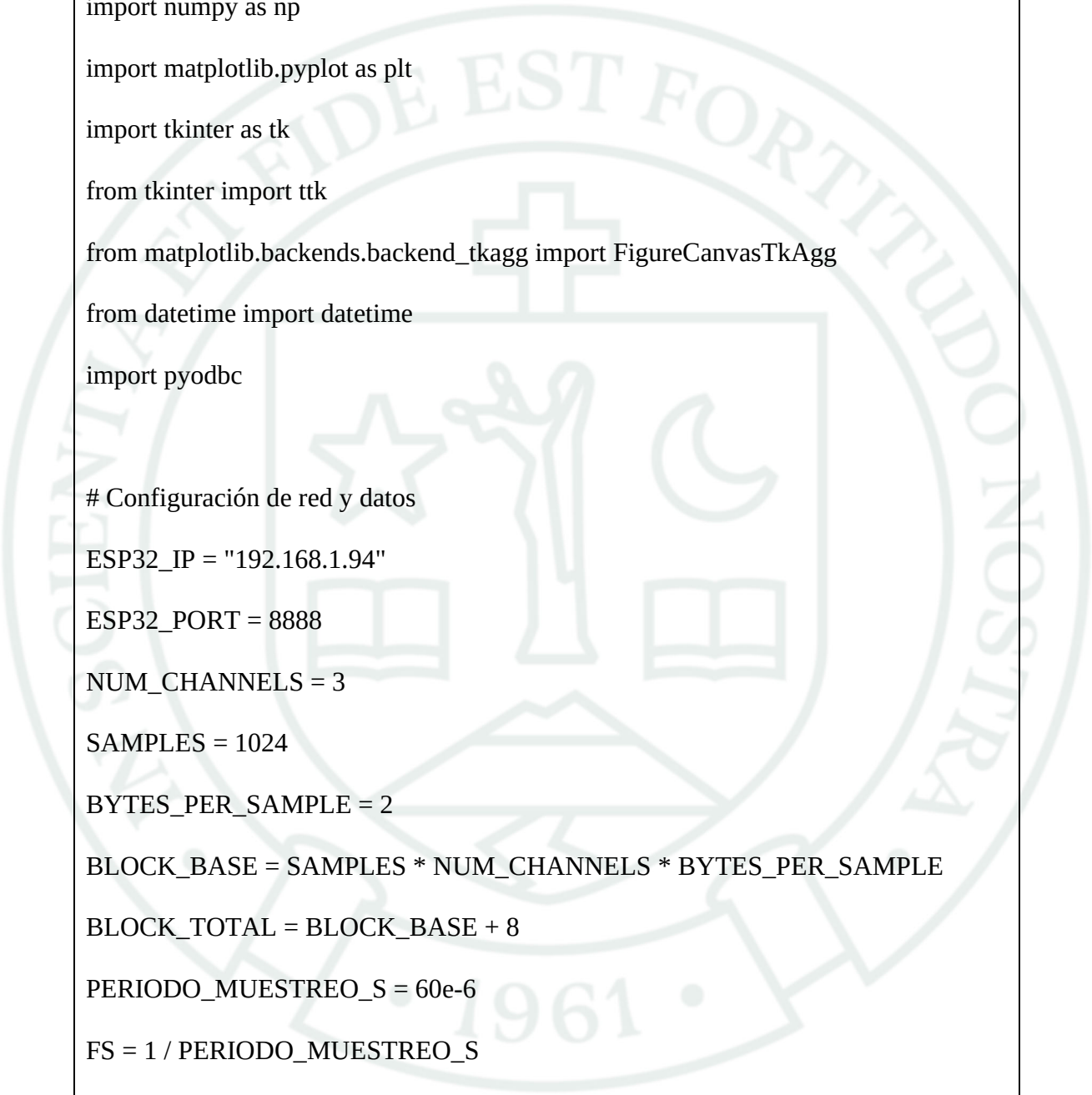
Este programa en Python implementa un sistema de monitoreo en tiempo real para un prototipo de bomba. Su objetivo es adquirir datos de sensores a través de un ESP32 mediante TCP/IP, procesarlos y mostrarlos en una interfaz gráfica mediante dashboards, además de almacenarlos en una base de datos SQL para análisis posterior.

- Adquisición: El ESP32 envía datos de los sensores mediante transmisión de TCP.
- Procesamiento: Python convierte los datos de ADC a unidades físicas.
- Visualización: Se muestran gráficos en tiempo real (tiempo y espectro FFT).
- Almacenamiento: Los resultados son guardados en SQL Server para posterior análisis.

Configuración de parámetros

Se definen:

- IP y puerto del ESP32 para comunicación vía socket transmisión TCP.
- Número de canales (voltaje, corriente y vibración).
- Número de muestras (1024) y parámetros de muestreo ($60 \mu\text{s} \rightarrow 16.6 \text{ kHz}$).
- Constantes de conversión de ADC para transformar las lecturas crudas a unidades físicas



```

import socket

import struct

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import tkinter as tk

from tkinter import ttk

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from datetime import datetime

import pyodbc

# Configuración de red y datos

ESP32_IP = "192.168.1.94"

ESP32_PORT = 8888

NUM_CHANNELS = 3

SAMPLES = 1024

BYTES_PER_SAMPLE = 2

BLOCK_BASE = SAMPLES * NUM_CHANNELS * BYTES_PER_SAMPLE

BLOCK_TOTAL = BLOCK_BASE + 8

PERIODO_MUESTREO_S = 60e-6

FS = 1 / PERIODO_MUESTREO_S

# Conversiones

ADC_RESOLUTION = 4095

```

```
RESOLT_ADC = 3.3 / ADC_RESOLUTION
```

```
VOLT_GAIN = 376.43147744
```

```
CURR_GAIN = 587.5 / 10
```

```
OFFSET_CRRENT =1.2
```

```
rRef = 1000
```

```
vRef = 3.3
```

```
r0 = 100
```

```
alpha = 0.00385
```

```
# Funciones de conversión
```

```
def adc_to_voltage(adc_val):
```

```
    return adc_val * RESOLT_ADC
```

```
def adc_to_current(adc_val):
```

```
    return adc_val * RESOLT_ADC * CURR_GAIN
```

```
def vibration_adc_to_mm_s(adc_val):
```

```
    voltage = adc_to_voltage(adc_val)
```

```
    current_mA = (voltage / 100) * 1000
```

```
    if current_mA < 4:
```

```
        return 0
```

```
    elif current_mA > 20:
```

```
        return 12
```

```
    else:
```

```
        return ((current_mA - 4) / 16) * 12
```

```

def adc_to_temp_celsius(adc_val):

    voltage = adc_to_voltage(adc_val)

    try:

        resistance = rRef * (voltage / (vRef - voltage))

        temperature = ((resistance / r0) - 1) / alpha

    except ZeroDivisionError:

        temperature = 0

    return temperature / 1000

def adc_to_pressure_mpa(adc_val):

    return (adc_val / 4095) * 0.5

# Tkinter ventana

root = tk.Tk()

root.title("Streaming en tiempo real")

notebook = ttk.Notebook(root)

notebook.pack(fill=tk.BOTH, expand=True)

# Variables

labels_val = [tk.StringVar(value="0") for _ in range(3)]

stat_vals = [tk.StringVar(value="0") for _ in range(9)]

ultimo_df = None

# Pestaña Tiempo

```

```

tab_tiempo = ttk.Frame(notebook)

notebook.add(tab_tiempo, text="Tiempo")

frame_tiempo = tk.Frame(tab_tiempo)

frame_tiempo.pack(fill=tk.BOTH, expand=True)

plot_frame = tk.Frame(frame_tiempo)

plot_frame.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

stats_frame = tk.Frame(frame_tiempo)

stats_frame.pack(side=tk.RIGHT, padx=10, pady=10)

fig, axs = plt.subplots(3, 1, figsize=(10, 6), sharex=True)

fig.tight_layout(pad=3)

canvas = FigureCanvasTkAgg(fig, master=plot_frame)

canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

labels = ["Voltaje (V)", "Corriente (A)", "Vibración (mm/s)"]

tiempo = (np.arange(SAMPLES) * PERIODO_MUESTREO_S) * 1000

lines = []

for i in range(3):

    axs[i].set_title(labels[i])

    axs[i].grid(True)

    axs[i].set_ylabel(labels[i])

    if i == 0:

```

```

    axs[i].set_ylim(-400, 400)

    elif i == 1:

        axs[i].set_ylim(-7, 7)

    elif i == 2:

        axs[i].set_ylim(0, 10)

    line, = axs[i].plot(tiempo, np.zeros_like(tiempo))

    lines.append(line)

axs[2].set_xlabel("Tiempo (ms)")

# Indicadores en costado derecho

tk.Label(stats_frame, text="Presión (MPa):").pack()

tk.Label(stats_frame, textvariable=labels_val[0]).pack()

tk.Label(stats_frame, text="Temperatura (°C):").pack()

tk.Label(stats_frame, textvariable=labels_val[1]).pack()

tk.Label(stats_frame, text="Flujo (L/min):").pack()

tk.Label(stats_frame, textvariable=labels_val[2]).pack()

for i, label in enumerate(["V_RMS", "V_MAX", "V_MIN", "I_RMS", "I_MAX",
"I_MIN", "VIB_RMS", "VIB_MAX", "VIB_MIN"]):

    tk.Label(stats_frame, text=label + ":").pack()

    tk.Label(stats_frame, textvariable=stat_vals[i]).pack()

# Pestaña Espectro

```

```

tab_fft = ttk.Frame(notebook)

notebook.add(tab_fft, text="Espectro")

fig_fft, axs_fft = plt.subplots(3, 1, figsize=(10, 6), sharex=True)
fig_fft.tight_layout(pad=3)
canvas_fft = FigureCanvasTkAgg(fig_fft, master=tab_fft)
canvas_fft.get_tk_widget().pack(fill=tk.BOTH, expand=True)

frequencies = np.fft.rfftfreq(SAMPLES, d=PERIODO_MUESTREO_S)

lines_fft = []
for i in range(3):
    axs_fft[i].set_title(labels[i])
    axs_fft[i].set_ylabel("Magnitud")
    axs_fft[i].grid(True)
    axs_fft[i].set_ylim(0, 1)
    axs_fft[i].set_xlim(0, 500) # Limitar a 2kHz
    line, = axs_fft[i].plot(frequencies, np.zeros_like(frequencies))
    lines_fft.append(line)
axs_fft[2].set_xlabel("Frecuencia (Hz)")

# Función de adquisición TCP

def recibir_datos_tcp():

    with socket.create_connection((ESP32_IP, ESP32_PORT), timeout=5) as s:

        raw = s.recv(BLOCK_TOTAL)

        while len(raw) < BLOCK_TOTAL:

```

```

raw += s.recv(BLOCK_TOTAL - len(raw))

muestras = struct.unpack(f">{SAMPLES * NUM_CHANNELS}h",
raw[:BLOCK_BASE])

df = pd.DataFrame({
    "Voltaje": muestras[0::3],
    "Corriente": muestras[1::3],
    "Vibración": muestras[2::3],
})

df_fisico = pd.DataFrame()

df_fisico["Voltaje (V)"] = df["Voltaje"] * RESOLT_ADC * VOLT_GAIN
df_fisico["Corriente (A)"] = df["Corriente"] * RESOLT_ADC * CURR_GAIN-
OFFSET_CRRENT

df_fisico["Vibración (mm/s)"] = df["Vibración"].apply(vibration_adc_to_mm_s)

presion_adc = struct.unpack(">h", raw[BLOCK_BASE:BLOCK_BASE + 2])[0]
temp_adc = struct.unpack(">h", raw[BLOCK_BASE + 2:BLOCK_BASE +
4])[0]

flujo = struct.unpack(">f", raw[BLOCK_BASE + 4:BLOCK_BASE + 8])[0]-6.5
presion = adc_to_pressure_mpa(presion_adc)
temperatura = adc_to_temp_celsius(temp_adc)

return df_fisico, presion, temperatura, flujo

# Guardar en SQL

def guardar_en_sql(df, presion, temperatura, flujo):

    try:

```

```

conn = pyodbc.connect('DRIVER={ODBC Driver 17 for SQL
Server};SERVER=DESKTOP-02LGSIH;DATABASE=master;UID=sa;PWD=ABC.123')

cursor = conn.cursor()

now = datetime.now()

stats = [

    df["Voltaje (V)"].std(), df["Voltaje (V)"].max(), df["Voltaje (V)"].min(),

    df["Corriente (A)"].std(), df["Corriente (A)"].max(), df["Corriente (A)"].min(),

    df["Vibración (mm/s)"].std(), df["Vibración (mm/s)"].max(), df["Vibración
(mm/s)"].min()

]

cursor.execute("""

INSERT INTO Measurement_table (

    timestamp, presion, temperatura, flujo,

    v_rms, v_max, v_min,

    i_rms, i_max, i_min,

    vib_rms, vib_max, vib_min

) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)

""", now, presion, temperatura, flujo, *stats)

waveform_data = [

    (now, i, df["Voltaje (V)"].iloc[i], df["Corriente (A)"].iloc[i], df["Vibración
(mm/s)"].iloc[i])

    for i in range(SAMPLES)

]

cursor.fast_executemany = True

cursor.executemany("""

```

```

INSERT INTO Waveform_table (
    timestamp, sample_index, voltaje, corriente, vibracion
) VALUES (?, ?, ?, ?, ?)

"", waveform_data)

conn.commit()

cursor.close()

conn.close()

#print(f"✅ Datos guardados en SQL para {now}.")

except Exception as e:

    print(f"❌ Error guardando en SQL: {e}")

# Actualización periódica
def actualizar():
    global ultimo_df

    try:
        df, presion, temperatura, flujo = recibir_datos_tcp()

        ultimo_df = df.copy()

        for i, col in enumerate(df.columns):
            lines[i].set_ydata(df[col].values)

        canvas.draw()

        for i, col in enumerate(df.columns):
            fft_mag = np.abs(np.fft.rfft(df[col].values))

            fft_mag /= np.max(fft_mag) if np.max(fft_mag) != 0 else 1

            lines_fft[i].set_ydata(fft_mag)

        canvas_fft.draw()

```

```

labels_val[0].set(f"{presion:.3f}")

labels_val[1].set(f"{temperatura:.2f}")

labels_val[2].set(f"{flujo:.2f}")

stat_vals[0].set(f"{df['Voltaje (V)'].std():.2f}")

stat_vals[1].set(f"{df['Voltaje (V)'].max():.2f}")

stat_vals[2].set(f"{df['Voltaje (V)'].min():.2f}")

stat_vals[3].set(f"{df['Corriente (A)'].std():.2f}")

stat_vals[4].set(f"{df['Corriente (A)'].max():.2f}")

stat_vals[5].set(f"{df['Corriente (A)'].min():.2f}")

stat_vals[6].set(f"{df['Vibración (mm/s)'].std():.2f}")

stat_vals[7].set(f"{df['Vibración (mm/s)'].max():.2f}")

stat_vals[8].set(f"{df['Vibración (mm/s)'].min():.2f}")

guardar_en_sql(df, presion, temperatura, flujo)

except Exception as e:

    print("✗ Error:", e)

root.after(1000, actualizar)

# Inicia actualización

root.after(1000, actualizar)

root.mainloop()

```

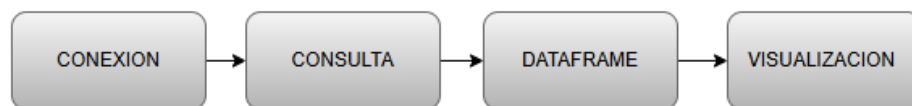
SEGUNDO PROGRAMA

4.2.2 Extracción de datos de señales desde sql server

Este programa se conecta a SQL Server donde esta almacenados los datos, consulta los datos de la tabla Waveform_table correspondientes a la fecha actual, y los carga en un DataFrame. Luego muestra una vista previa de los primeros registros obtenidos en Python.

Figura 31

Diagrama de flujo de extracción de datos desde SQL Server



Nota: Elaboración propia, diagrama de flujo de extraccion de datos SQL Server

```
import pyodbc  
import pandas as pd  
from datetime import datetime  
  
# --- Configuración de la conexión ---  
server = 'localhost'  
  
database = 'master'  
  
username = 'sa'  
  
password = 'ABC.123'
```

```

driver = '{ODBC Driver 17 for SQL Server}'

# --- Cadena de conexión ---

conn_str = f"""

DRIVER={driver};
SERVER={server};
DATABASE={database};
UID={username};
PWD={password};

"""

# --- Obtener la fecha actual del sistema en formato YYYY-MM-DD ---

hoy = datetime.now().date() # tipo: datetime.date

# --- Conectar a la base de datos ---

conn = pyodbc.connect(conn_str)

# --- Consulta SQL: filtrar por la fecha actual ---

query = f"""

SELECT [timestamp], [sample_index], [voltaje], [corriente], [vibracion]
FROM [master].[dbo].[Waveform_table]
WHERE CONVERT(date, [timestamp]) = '{hoy}'

"""

# --- Ejecutar consulta y guardar en DataFrame ---

```

```
df = pd.read_sql(query, conn)

# --- Mostrar los primeros registros ---
print(df.head())

# --- Cerrar conexión ---
conn.close()
```

4.2.3 Procesamiento y análisis espectral de señales de vibración en bombas centrífugas

Este tipo de análisis es usado para el monitoreo de maquinaria (bombas, motores) ya que permite diagnosticar fallas por desbalance, desalineación, resonancias y problemas eléctricos.

- Toma datos de vibración de una base de datos.
- Los concatena en una señal continua de 5120 puntos.
- Calcula el espectro de frecuencia (FFT) hasta 1 kHz.
- Detecta la frecuencia fundamental y sus armónicos.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

```

# 1 Parámetros generales

fs = 1 / 60e-6      # 16.666 kHz

n_paq = 5

N = 1024 * n_paq    # 5120 muestras

# 2 Concatenar 5 paquetes

first_5_ts = df["timestamp"].unique()[:n_paq]

df_concat = (df[df["timestamp"].isin(first_5_ts)]

               .sort_values(["timestamp", "sample_index"])

               .reset_index(drop=True))

vib_raw = df_concat["vibracion"].values.astype(float)

# 3 Remover DC por FFT

vib_fft = np.fft.fft(vib_raw)

vib_fft[0] = 0

vib_dcfree = np.fft.ifft(vib_fft).real      # señal sin DC

# 4 FFT y espectro normalizado (0-1 kHz)

freqs = np.fft.fftfreq(N, d=1/fs)[:N//2]

mag = np.abs(np.fft.fft(vib_dcfree))[:N//2] * 2 / N

mask = freqs <= 1000

freqs, mag = freqs[mask], mag[mask]

mag_norm = mag / mag.max()

```

```

# 5 Fundamental y armónicos

idx_peak = np.argmax(mag_norm)

f0 = freqs[idx_peak]

armonics = np.arange(2, int(1000 // f0) + 1) * f0

# 6 Graficar

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 9))

# 6.1 Forma de onda (zoom primeros 500 puntos)

zoom_pts = 500

ax1.plot(np.arange(zoom_pts), vib_dcfree[:zoom_pts], color="red")

ax1.set_title(f"Vibración sin DC – Zoom en los primeros {zoom_pts} puntos")

ax1.set_ylabel("Amplitud [mm/s]")

ax1.set_xlabel("Índice de muestra")

ax1.grid(True)

# 6.2 Espectro normalizado

ax2.plot(freqs, mag_norm, color="purple", label="Espectro")

ax2.scatter(f0, mag_norm[idx_peak], color="gold", zorder=5, label=f" $f_0 \approx \{f0:.1f\}$  Hz")

ax2.text(f0, mag_norm[idx_peak] + 0.02, f" $\{f0:.1f\}$  Hz", ha="center", color="gold")

# Armónicos

for k, fk in enumerate(armonics, start=2):

```

```

ax2.axvline(fk, color="gray", linestyle="--", alpha=0.6)

ax2.text(fk, 0.9, f'{k}fo', rotation=90, va="top", ha="center", fontsize=8)

ax2.set_title("Espectro normalizado – 0-1 kHz (armónicos resaltados)")
ax2.set_ylabel("Amplitud (0-1)")
ax2.set_xlabel("Frecuencia [Hz]")
ax2.grid(True)
ax2.legend()

plt.tight_layout()
plt.savefig("vibracion_zoom500_wave_spectrum.png", dpi=300)
plt.show()

```

4.2.4 Procesamiento y análisis espectral de señales de voltaje y corriente en tiempo discreto

Este programa está orientado a analizar la relación entre el voltaje y la respuesta en corriente, identificando posibles armónicos o distorsión armónica.

- Permite comparar señal de voltaje y corriente de 5 ciclos.
- La corriente es preprocesada sin DC para un análisis espectral más claro.
- El espectro se limita a 0–2 kHz, ideal para análisis de armónicos eléctricos.
- Se destaca la frecuencia fundamental y sus armónicos, útil para diagnóstico de calidad de energía o análisis de cargas no lineales.
- Genera una figura final lista para análisis e informes técnicos.

```

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# ----- 1. Configuración general -----

fs      = 1 / 60e-6      # 16.666 kHz

n_paq    = 5

N        = 1024 * n_paq  # 5120 muestras

# ----- 2. Cargar y concatenar cinco ciclos -----

first_5_ts = df["timestamp"].unique()[:n_paq]

df_concat = (df[df["timestamp"].isin(first_5_ts)]

               .sort_values(["timestamp", "sample_index"])

               .reset_index(drop=True))

df_concat["sample_index_global"] = df_concat.index

idx = df_concat["sample_index_global"].values

volt_raw  = df_concat["voltaje"].values.astype(float)

curr_raw  = df_concat["corriente"].values.astype(float)

# ----- 3. Corriente sin DC (anular bin 0) -----

curr_fft   = np.fft.fft(curr_raw)

curr_fft[0] = 0          # quita DC (0 Hz)

curr_dcfree = np.fft.ifft(curr_fft).real

```

```

# ----- 4. Espectro normalizado (0–2 kHz) -----

freqs = np.fft.fftfreq(N, d=1/fs)[:N//2]

mag = np.abs(np.fft.fft(curr_dcfree))[:N//2] * 2 / N

mask = freqs <= 2000

freqs = freqs[mask]

mag_norm = (mag[mask]) / mag[mask].max()

# ----- 5. Frecuencia fundamental – 5 Hz -----

f0_idx = np.argmax(mag_norm)

f0 = freqs[f0_idx] # pico real

f0_adj = max(f0 - 5, 1) # le restamos 5 Hz, mínimo 1 Hz para evitar cero

arm = np.arange(2, int(2000 // f0_adj) + 1) * f0_adj

# ----- 6. Graficar -----

fig, (ax_volt, ax_curr, ax_spec) = plt.subplots(3, 1, figsize=(14, 10), sharex=False)

# 6.1 Voltaje

ax_volt.plot(idx, volt_raw, color="blue")

ax_volt.set_title("Voltaje – 5 ciclos concatenados")

ax_volt.set_ylabel("Voltaje [V]")

ax_volt.set_xlabel("Muestras (índice continuo)")

ax_volt.grid(True)

```

```
# 6.2 Corriente sin DC
```

```
ax_curr.plot(idx, curr_dcfree, color="green")
```

```
ax_curr.set_title("Corriente sin componente DC")
```

```
ax_curr.set_ylabel("Corriente [A]")
```

```
ax_curr.set_xlabel("Muestras (índice continuo)")
```

```
ax_curr.grid(True)
```

```
# 6.3 Espectro normalizado
```

```
ax_spec.plot(freqs, mag_norm, color="orange", label="Espectro")
```

```
ax_spec.scatter(f0_adj, mag_norm[np.argmin(np.abs(freqs - f0_adj))],  
                color="gold", zorder=5, label=f"f0-5 Hz ≈ {f0_adj:.1f} Hz")
```

```
ax_spec.text(f0_adj, 1.02, f"{f0_adj:.1f} Hz", ha="center", color="gold")
```

```
# Armónicos
```

```
for k, fk in enumerate(arm, start=2):
```

```
    ax_spec.axvline(fk, color="gray", linestyle="--", alpha=0.6)
```

```
    ax_spec.text(fk, 0.9, f"{k} f0", rotation=90,
```

```
                va="top", ha="center", fontsize=8)
```

```
ax_spec.set_title("Espectro normalizado – Corriente (0–2 kHz)")
```

```
ax_spec.set_ylabel("Amplitud normalizada")
```

```
ax_spec.set_xlabel("Frecuencia [Hz]")
```

```
ax_spec.grid(True)
```

```
ax_spec.legend()
```

```
plt.tight_layout()

plt.savefig("voltaje_corriente_ajustada.png", dpi=300)

plt.show()
```

4.2.5 Análisis de vibraciones en el dominio del tiempo y frecuencia mediante FFT

Este programa constituye una herramienta eficaz para el procesamiento digital de señales de vibración, mostrando el comportamiento temporal en microsegundos y revelando las características espectrales de la señal mediante FFT, lo cual podemos visualizar para realizar diagnóstico de condiciones operativas y fallas en sistemas de monitoreo.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Parámetros

fs = 1 / 60e-6      # 16.666 kHz
dt = 60e-6          # 60 µs
n_paq = 5

N = 1024 * n_paq    # 5120 muestras

# Concatenar 5 paquetes
```

```

first_5_ts = df["timestamp"].unique()[:n_paq]

df_concat = (
    df[df["timestamp"].isin(first_5_ts)]
    .sort_values(["timestamp", "sample_index"])
    .reset_index(drop=True)
)

vib_raw = df_concat["vibracion"].values.astype(float)

# 3 Remover DC
vib_fft = np.fft.fft(vib_raw)
vib_fft[0] = 0
vib = np.fft.ifft(vib_fft).real

# 4 FFT y espectro 0-1 kHz
freqs = np.fft.fftfreq(N, d=1/fs)[:N//2]
mag = np.abs(np.fft.fft(vib))[:N//2] * 2 / N
mask = freqs <= 1000
freqs, mag = freqs[mask], mag[mask]
mag_norm = mag / mag.max()

# 5 Fundamental y armónicos
idx_peak = np.argmax(mag_norm)
f0 = freqs[idx_peak]
armonics = np.arange(2, int(1000 // f0) + 1) * f0

```

```

# 6 Graficar

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 9))

# Forma de onda – 50 puntos, eje X en microsegundos
n_zoom = 50
t_zoom_us = np.arange(n_zoom) * dt * 1e6 # tiempo en  $\mu$ s
v_zoom = vib[:n_zoom]

ax1.plot(t_zoom_us, v_zoom, color="red", marker='o')
ax1.set_title(f"Vibración sin DC – Primeros {n_zoom} puntos ( $\Delta t = 60 \mu$ s)")
ax1.set_xlabel("Tiempo [ $\mu$ s]")
ax1.set_ylabel("Amplitud [mm/s]")
ax1.grid(True)

# Etiquetas de tiempo personalizadas ( $\mu$ s)
ax1.set_xticks(t_zoom_us)
ax1.set_xticklabels([f"{t:.0f}" for t in t_zoom_us], rotation=45)

# Espectro normalizado
ax2.plot(freqs, mag_norm, color="purple", label="Espectro")
ax2.scatter(f0, mag_norm[idx_peak], color="gold", zorder=5, label=f" $f_0 \approx \{f0:.1f\}$  Hz")
ax2.text(f0, mag_norm[idx_peak] + 0.02, f" $\{f0:.1f\}$  Hz", ha="center", color="gold")

```

```
for k, fk in enumerate(armonics, start=2):

    ax2.axvline(fk, color="gray", linestyle="--", alpha=0.6)

    ax2.text(fk, 0.9, f'{k} f0', rotation=90, va="top", ha="center", fontsize=8)

ax2.set_title("Espectro normalizado – 0–1 kHz (armónicos resaltados)")
ax2.set_xlabel("Frecuencia [Hz]")
ax2.set_ylabel("Amplitud (0–1)")
ax2.grid(True)
ax2.legend()

plt.tight_layout()
plt.savefig("vibracion_zoom50_time_us_spectrum.png", dpi=300)
plt.show()
```

TERCER PROGRAMA

4.2.6 Extracción y análisis temporal de parámetros de medición desde sql server

Programa de adquisición y consulta de datos que prepara la información para análisis posterior.

- Se conecta a SQL Server usando pyodbc para visualizar data
- Obtiene los registros del día actual de la tabla de la base de datos.
- Extrae variables de operación.
- Convierte y ordena los datos en pandas para análisis.
- Presenta un resumen de los datos del día.

```
import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Configuración de la conexión

server = 'localhost'
database = 'master'
username = 'sa'
password = 'ABC.123'
driver = '{ODBC Driver 17 for SQL Server}'
```

```

conn_str = (
    f'DRIVER={driver};'
    f'SERVER={server};'
    f'DATABASE={database};'
    f'UID={username};PWD={password}'
)

# Obtener fecha del sistema
hoy = datetime.now().date() # Ej: 2025-07-19

# Consulta SQL para datos del día actual
query = f"""
SELECT
    [timestamp],
    [presion],
    [temperatura],
    [flujo],
    [v_rms],
    [v_max],
    [v_min],
    [i_rms],
    [i_max],
    [i_min],
    [vib_rms],
    [vib_max],

```

```

[vib_min]

FROM [master].[dbo].[Measurement_table]

WHERE CONVERT(date, [timestamp]) = '{hoy}'

ORDER BY [timestamp] ASC

"""

# ❹ Leer a DataFrame

with pyodbc.connect(conn_str) as conn:

    df_sql = pd.read_sql(query, conn)

# ❺ Procesamiento

df_sql["timestamp"] = pd.to_datetime(df_sql["timestamp"])
df_sql.sort_values("timestamp", inplace=True)

# ❻ Opcional: mostrar resumen

print(df_sql.head())

print(f"\n ❷ Total de registros del {hoy}: {len(df_sql)}")

```

4.2.7 Visualización de parámetros físicos, eléctricos y mecánicos de operación del prototipo

Este programa es una herramienta de monitoreo integral que consolida en una sola visualización los principales parámetros físicos (presión, temperatura, flujo), eléctricos (voltaje y corriente RMS) y mecánicos (vibración RMS).

```

import matplotlib.pyplot as plt

import pandas as pd

# Asegurar que 'timestamp' sea datetime y esté ordenado
df_sql["timestamp"] = pd.to_datetime(df_sql["timestamp"])
df_sql.sort_values("timestamp", inplace=True)

# ☐ Extraer la última fecha (solo la parte de fecha)
fecha_final = df_sql["timestamp"].dt.date.max()

# ☒ Filtrar por esa fecha
df_dia = df_sql[df_sql["timestamp"].dt.date == fecha_final]

# Verificar que hay datos
if df_dia.empty:
    print(f"No hay datos para la fecha más reciente: {fecha_final}")
else:
    # ☒ Crear gráfico

    fig, axes = plt.subplots(3, 1, figsize=(14, 10), sharex=True)

    # Panel 1: Presión, Temperatura, Flujo
    axes[0].plot(df_dia["timestamp"], df_dia["presion"], label="Presión", color="blue")
    axes[0].plot(df_dia["timestamp"], df_dia["temperatura"], label="Temperatura",
color="red")

    axes[0].plot(df_dia["timestamp"], df_dia["flujo"], label="Flujo", color="green")
    axes[0].set_ylabel("Magnitudes físicas")
    axes[0].set_title(f"Presión, Temperatura y Flujo – {fecha_final}")

```

```

axes[0].legend()

axes[0].grid(True)

# Panel 2: Voltaje y Corriente RMS

axes[1].plot(df_dia["timestamp"], df_dia["v_rms"], label="Voltaje RMS",
color="purple")

axes[1].plot(df_dia["timestamp"], df_dia["i_rms"], label="Corriente RMS",
color="orange")

axes[1].set_ylabel("RMS [V / A]")

axes[1].set_title("Voltaje y Corriente RMS")

axes[1].legend()

axes[1].grid(True)

# Panel 3: Vibración RMS

axes[2].plot(df_dia["timestamp"], df_dia["vib_rms"], label="Vibración RMS",
color="brown")

axes[2].set_ylabel("Vibración [mm/s]")

axes[2].set_xlabel("Hora")

axes[2].set_title("Vibración RMS")

axes[2].legend()

axes[2].grid(True)

# Guardar figura

plt.tight_layout()

plt.savefig(f"mediciones_{fecha_final}.png", dpi=300)

plt.show()

```

4.2.8 Análisis temporal de señales de voltaje, corriente y vibración adquiridas desde sql server

Este programa combina la lectura de datos desde SQL Server, procesamiento digital con filtrado y visualización de gráficas voltaje, corriente y vibración.

- Conexión a base de datos
- Parámetros de muestreo
- Procesamiento de datos
- Visualización (dashboards)

```
import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt

# ◆ Configuración de conexión
server = 'localhost'
database = 'master'
username = 'sa'
password = 'ABC.123'
driver = '{ODBC Driver 17 for SQL Server}'

conn_str = (
    f'DRIVER={driver};'
    f'SERVER={server};'
    f'DATABASE={database};'
```

```

f'UID={username};PWD={password}'

)

# ◆ Parámetros de muestreo

PERIODO_MUESTREO_S = 60e-6 # 60 µs

FS = 1 / PERIODO_MUESTREO_S # Frecuencia de muestreo ≈ 16.67 kHz

# ◆ Parámetros del filtro

ORDER = 4

CUTOFF_HZ = 1000 # 1 kHz

def butter_lowpass(cutoff, fs, order=4):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype="low", analog=False)
    return b, a

def aplicar_filtro(data, cutoff=CUTOFF_HZ, fs=FS, order=ORDER):
    b, a = butter_lowpass(cutoff, fs, order)
    return filtfilt(b, a, data)

# ◆ Fecha a analizar

fecha_objetivo = "2025-08-22"

```

```

# ◆ Consulta SQL

query = f"""

SELECT timestamp, sample_index, voltaje, corriente, vibracion
FROM Waveform_table

WHERE CONVERT(date, timestamp) = '{fecha_objetivo}'

ORDER BY timestamp ASC, sample_index ASC

"""

# ◆ Leer datos

with pyodbc.connect(conn_str) as conn:

    df = pd.read_sql(query, conn)

print(f"✅ Registros obtenidos para {fecha_objetivo}: {len(df)}")

# ◆ Agrupar por captura (timestamp)

capturas = df.groupby("timestamp")

# ◆ Graficar cada captura en una figura separada

for ts, bloque in capturas:

    tiempo = bloque["sample_index"] * PERIODO_MUESTREO_S

    voltaje_filtrado = aplicar_filtro(bloque["voltaje"].values)

    corriente_filtrada = aplicar_filtro(bloque["corriente"].values)

    vibracion_filtrada = aplicar_filtro(bloque["vibracion"].values)

    plt.figure(figsize=(12, 8))

```

```
plt.suptitle(f"Captura {ts} - Filtrado Butterworth {ORDER}º, {CUTOFF_HZ} Hz",
fontsize=14)

# Voltaje
plt.subplot(3, 1, 1)
plt.plot(tiempo, voltaje_filtrado, color="blue")
plt.ylabel("Voltaje (V)")
plt.grid(True)

# Corriente
plt.subplot(3, 1, 2)
plt.plot(tiempo, corriente_filtrada-2, color="red")
plt.ylabel("Corriente (A)")
plt.grid(True)

# Vibración
plt.subplot(3, 1, 3)
plt.plot(tiempo, vibracion_filtrada, color="green")
plt.ylabel("Vibración")
plt.xlabel("Tiempo (s)")
plt.grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.95])

plt.show()
```

CUARTO PROGRAMA

4.2.9 Sistema de adquisición para la detección de eventos de sobrecorriente en prototipo de monitoreo

Este programa ayuda a visualizar el sistema de monitoreo y análisis automático de transitorios eléctricos y mecánicos, diseñado para:

- Extraer datos crudos desde SQL Server.
- Filtrar ruido mediante un filtro digital pasa-bajo.
- Detectar y graficar solo los eventos de sobrecorriente (>10 A).
- Mostrar la interacción entre voltaje, corriente y vibración en cada evento.

```
import pyodbc
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt

# ♦ Configuración de conexión
server = 'localhost'
database = 'master'
username = 'sa'
password = 'ABC.123'
driver = '{ODBC Driver 17 for SQL Server}'

conn_str = (
```

```

f'DRIVER={driver};'

f'SERVER={server};'

f'DATABASE={database};'

f'UID={username};PWD={password}'

)

# ◆ Parámetros de muestreo

PERIODO_MUESTREO_S = 60e-6 # 60 µs

FS = 1 / PERIODO_MUESTREO_S # Frecuencia de muestreo ≈ 16.67 kHz

SAMPLES = 1024

# ◆ Parámetros de filtro

FILTER_ORDER = 2

CUTOFF_HZ = 1000 # Frecuencia de corte

# ◆ Fecha a analizar

fecha_objetivo = "2025-08-22"

# ◆ Consulta SQL

query = f"""

SELECT timestamp, sample_index, voltaje, corriente, vibracion

FROM Waveform_table

WHERE CONVERT(date, timestamp) = '{fecha_objetivo}'

ORDER BY timestamp ASC, sample_index ASC

"""

```

```

# ◆ Leer datos

with pyodbc.connect(conn_str) as conn:

    df = pd.read_sql(query, conn)

print(f"✅ Registros obtenidos para {fecha_objetivo}: {len(df)}")

# =====

# ◆ Diseño del filtro pasa-bajo

# =====

def butter_lowpass(cutoff, fs, order=FILTER_ORDER):

    nyq = 0.5 * fs          # Frecuencia de Nyquist
    normal_cutoff = cutoff / nyq

    b, a = butter(order, normal_cutoff, btype="low", analog=False)

    return b, a

def aplicar_filtro(data, cutoff=CUTOFF_HZ, fs=FS, order=FILTER_ORDER):

    b, a = butter_lowpass(cutoff, fs, order)

    return filtfilt(b, a, data)

# ◆ Agrupar por captura (timestamp)

capturas = df.groupby("timestamp")

graf_count = 0

for ts, bloque in capturas:

```

```

# Condición: si alguna muestra supera 10A

if bloque["corriente"].max() > 10:

    graf_count += 1

    tiempo = bloque["sample_index"] * PERIODO_MUESTREO_S

    # Aplicar filtro pasa-bajo
    corriente_filtrada = aplicar_filtro(bloque["corriente"].values)
    voltaje_filtrado = aplicar_filtro(bloque["voltaje"].values)
    vibracion_filtrada = aplicar_filtro(bloque["vibracion"].values)

    # === Crear figura separada por captura ===
    plt.figure(figsize=(12, 8))
    plt.suptitle(f"Captura en {ts} (Corriente > 10 A)", fontsize=14)

    plt.subplot(3, 1, 1)
    plt.plot(tiempo, voltaje_filtrado, color="blue")
    plt.ylabel("Voltaje (V)")
    plt.grid(True)

    plt.subplot(3, 1, 2)
    plt.plot(tiempo, corriente_filtrada, color="green")
    plt.axhline(10, color="r", linestyle="--", label="Umbral 10 A")
    plt.ylabel("Corriente (A)")

    plt.legend()

    plt.grid(True)

```

```
plt.subplot(3, 1, 3)

plt.plot(tiempo, vibracion_filtrada, color="orange")

plt.ylabel("Vibración")

plt.xlabel("Tiempo (s)")

plt.grid(True)

plt.tight_layout(rect=[0, 0, 1, 0.95])

plt.show()

if graf_count == 0:
    print("⚠ No se encontraron capturas con corriente > 10 A en ese día.")
else:
    print(f"📊 Se graficaron {graf_count} capturas.")
```



CAPITULO V

IMPLEMENTACION DEL EQUIPO

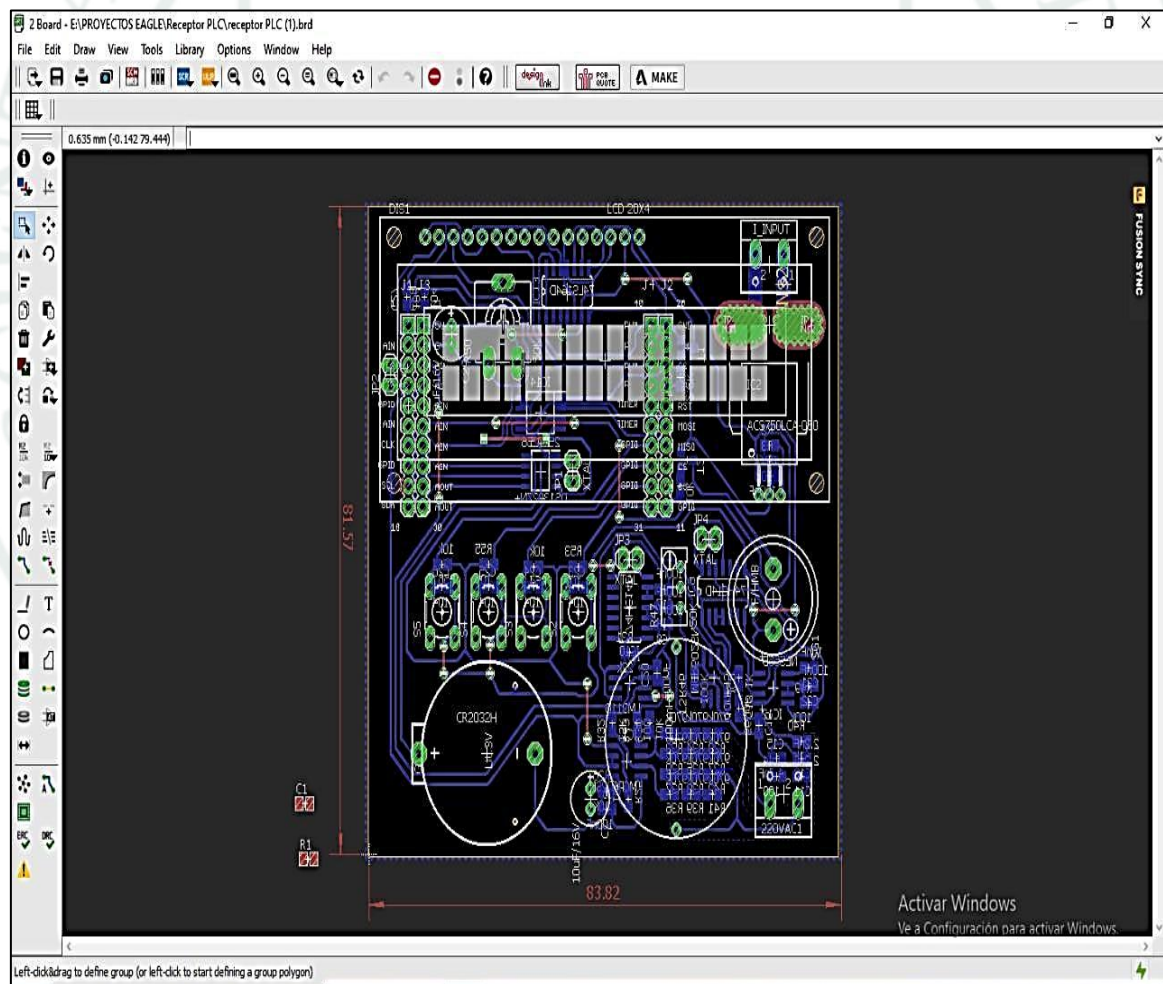
En este capítulo se describirá cada parte del equipo con todos los circuitos diseñados con sus respectivas pruebas para el monitoreo del sistema de bombas centrífugas y también se mostrarán los resultados obtenidos de cada parte.

5.1 Diseño de placa

Se realizó la etapa de diseño de circuitos de placa con todos los componentes que se deben tener para la etapa de resultados que se obtendrá de este prototipo.

Figura 32

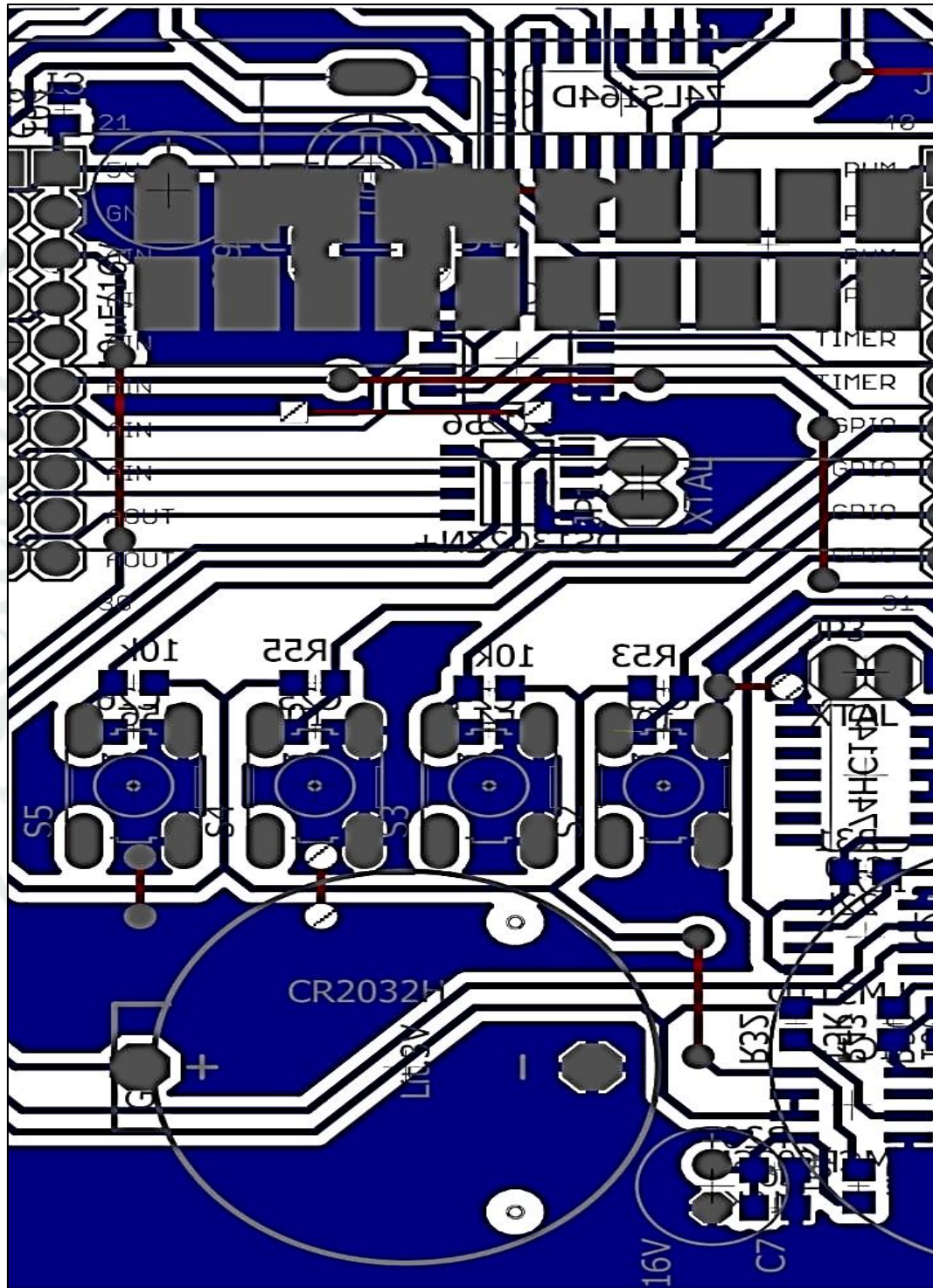
Diseño de placa



Nota: Elaboración propia, diseño de placa para montaje

Figura 33

Circuito impreso en pistas



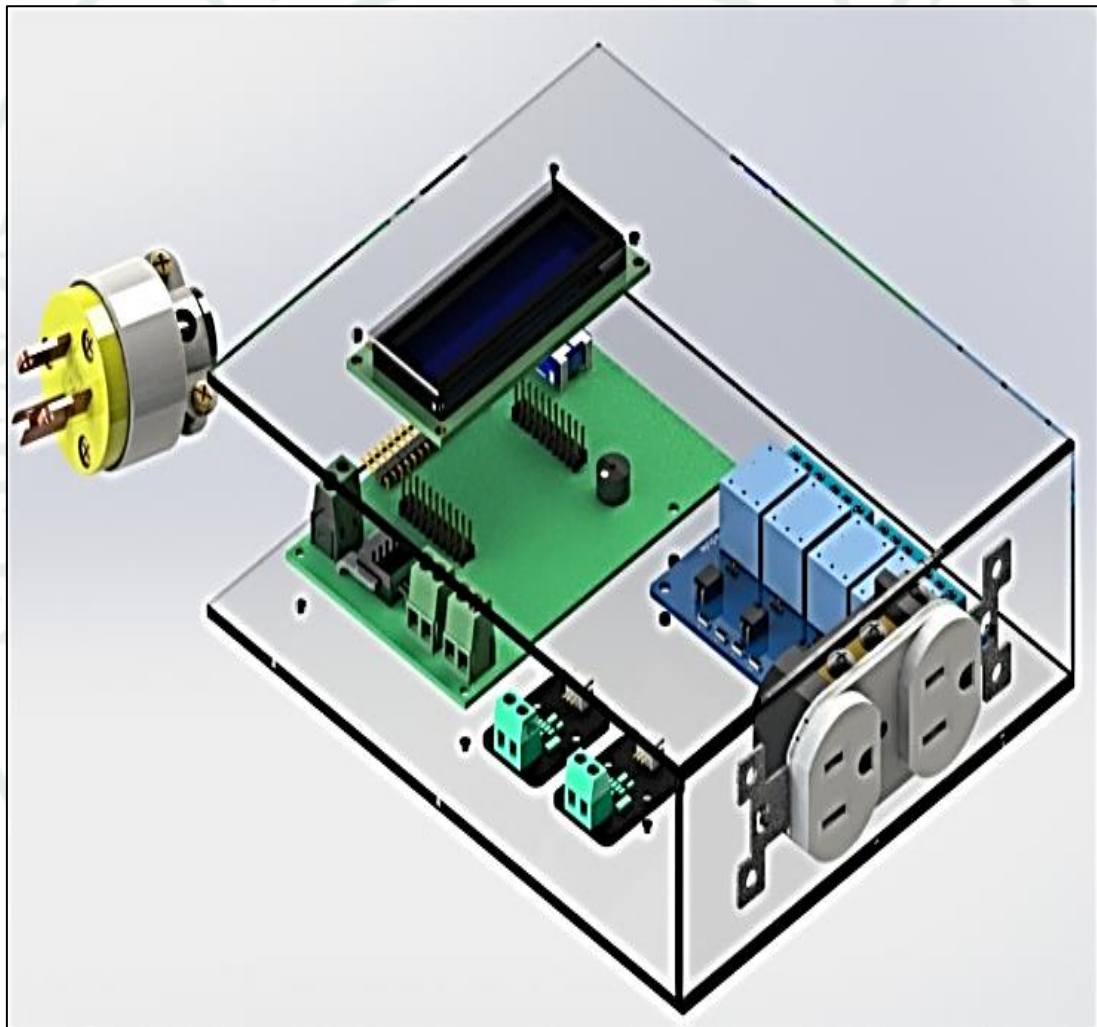
Nota: Elaboración propia, diseño de placa para montaje impresión

5.2 Diseño de prototipo caja fibra de vidrio

Diseño del prototipo de la caja con de fibra de vidrio, con el fin de analizar el espacio que ocupan los diferentes componentes que disponemos en el prototipo, para una correcta protección en operación.

Figura 34

Caja con protección fibra de vidrio

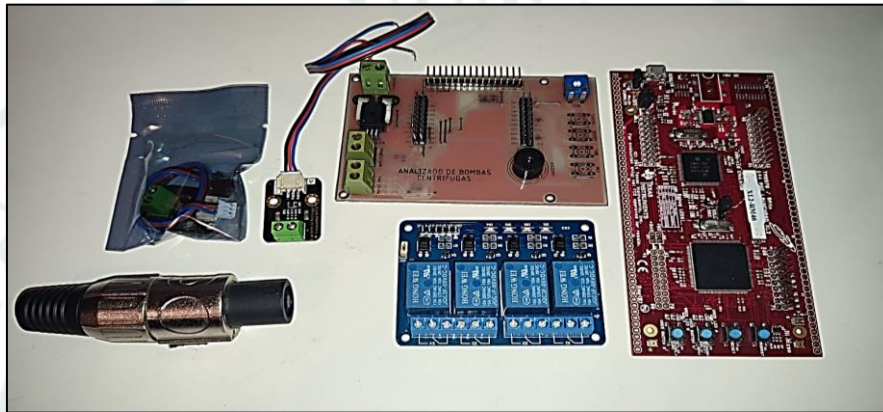


Nota: Elaboración propia, diseño de caja con proteccion fibra de vidrio

Componentes que se dispusieron para el diseño y de la caja con el objetivo de dimensionar caja protectora de fibra de vidrio.

Figura 35

Componentes internos prototipo



Nota: Elaboración propia, detalle de componentes internos de prototipo

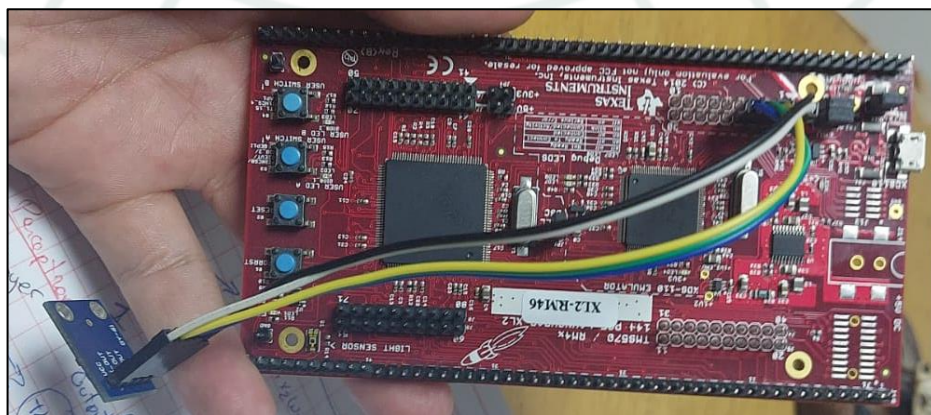
5.3 Ajustes de parámetros de tarjetas

5.3.1 Microcontrolador LAUNCHXL2-RM46 - HERCULES

Ajustes de parámetros (código de programa) de nuestro microcontrolador

Figura 36

Microcontrolador Hércules



5.3.2 Módulo de WI-FI ESP32

Ajuste de parámetros (código de programa) de nuestro módulo Wi-Fi ESP32

Figura 37

Módulo de Wi-fi ESP32

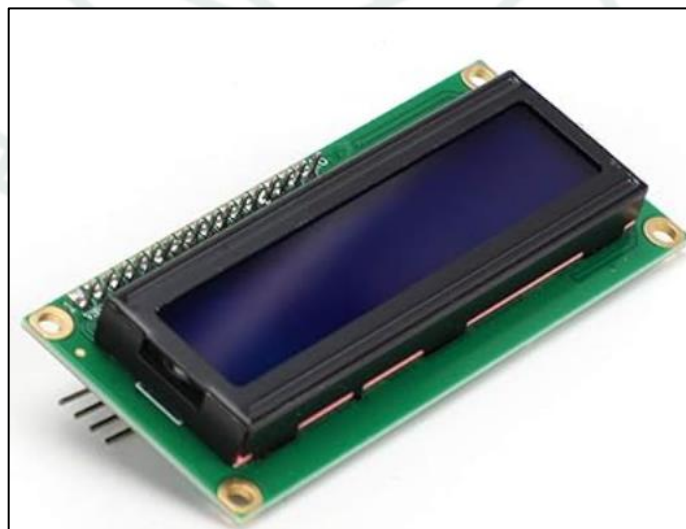


5.3.3 Módulo pantalla LCD 2X16

Ajuste de parámetros (código de programa) de nuestra pantalla LCD 2X16.

Figura 38

Pantalla LCD 2X16



5.4. Implementación de prototipo de monitoreo:

Implementación de prototipo de monitoreo

Figura 39

Vista superior de prototipo



Figura 40

Vista derecha de prototipo



Figura 41

Vista izquierda de prototipo

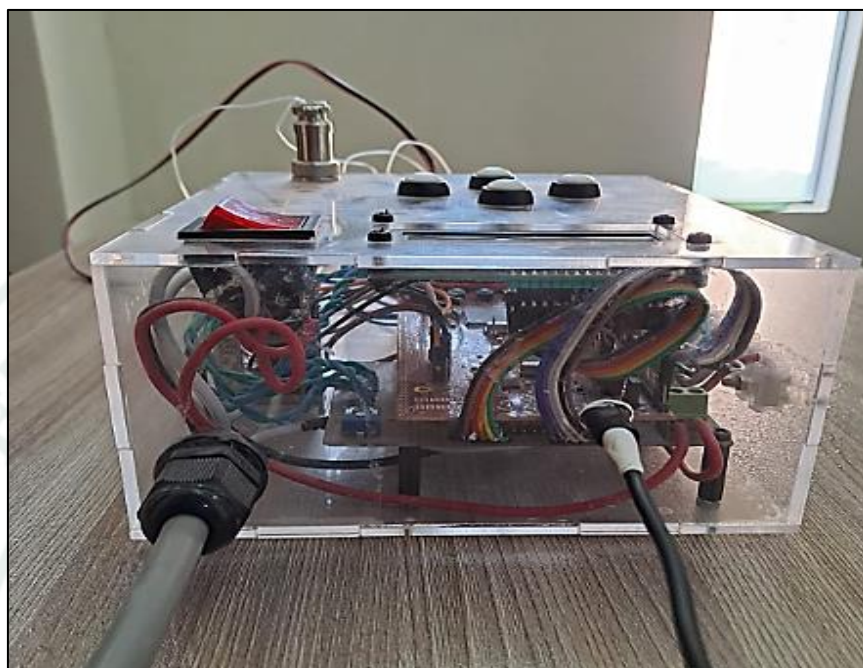


Figura 42

Vista derecha abajo de prototipo

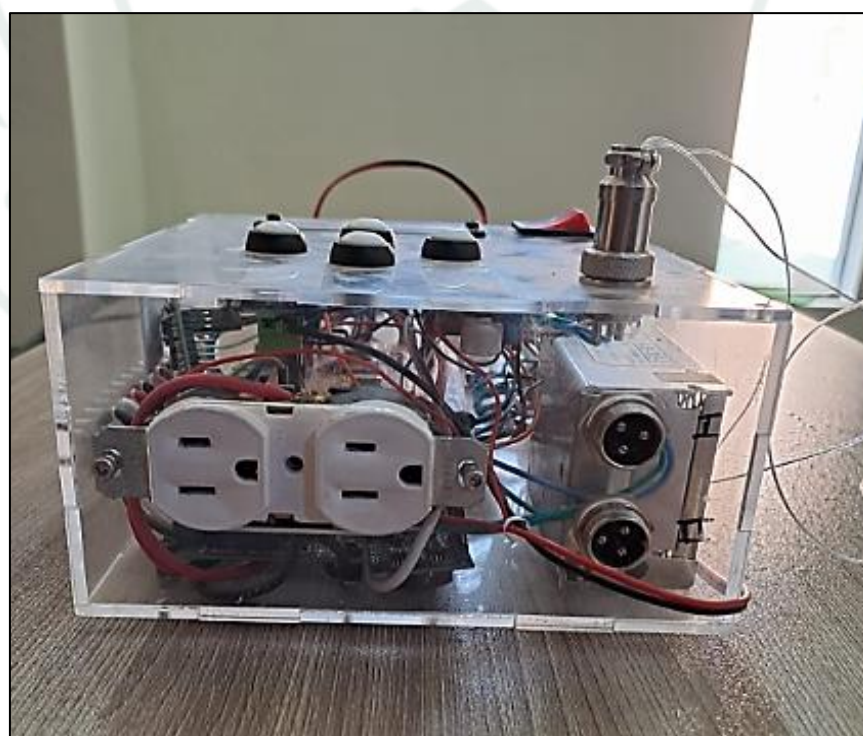
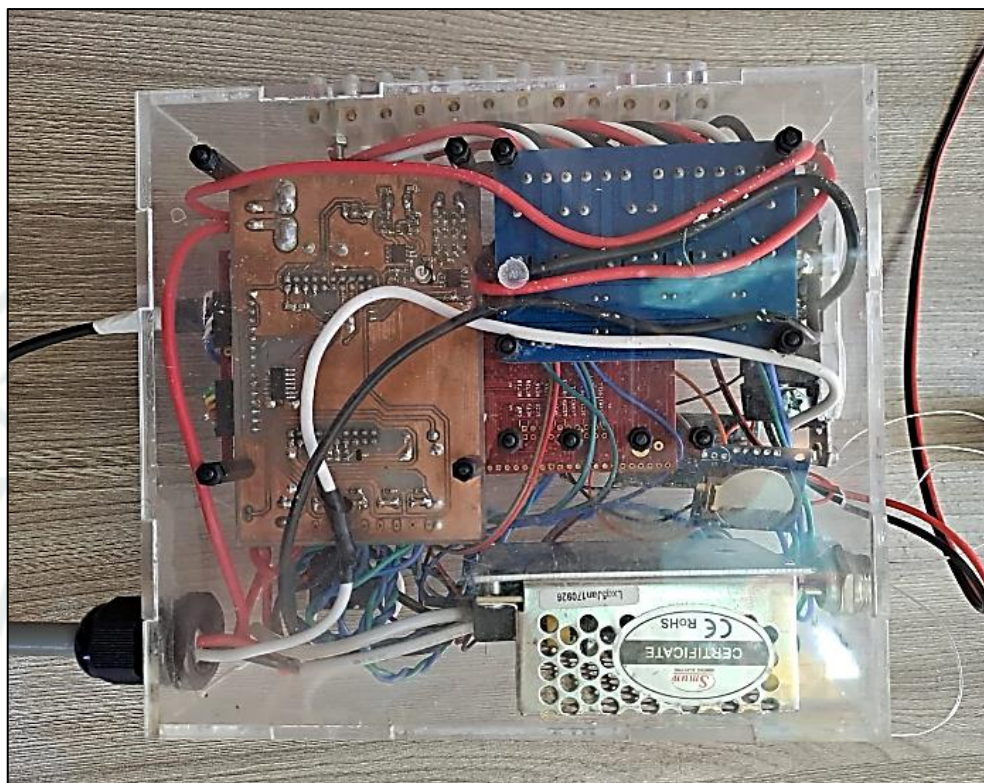


Figura 43

Vista inferior de prototipo



Nota: Elaboración propia, implementación de prototipo

5.5 Implementación de un sistema piloto

Implementar un sistema piloto para simular las condiciones de operación de las bombas centrífugas, utilizando tuberías y un tanque de almacenamiento para la recirculación del flujo, permitiendo comprender el comportamiento de estos sistemas a una escala menor.

Figura 44

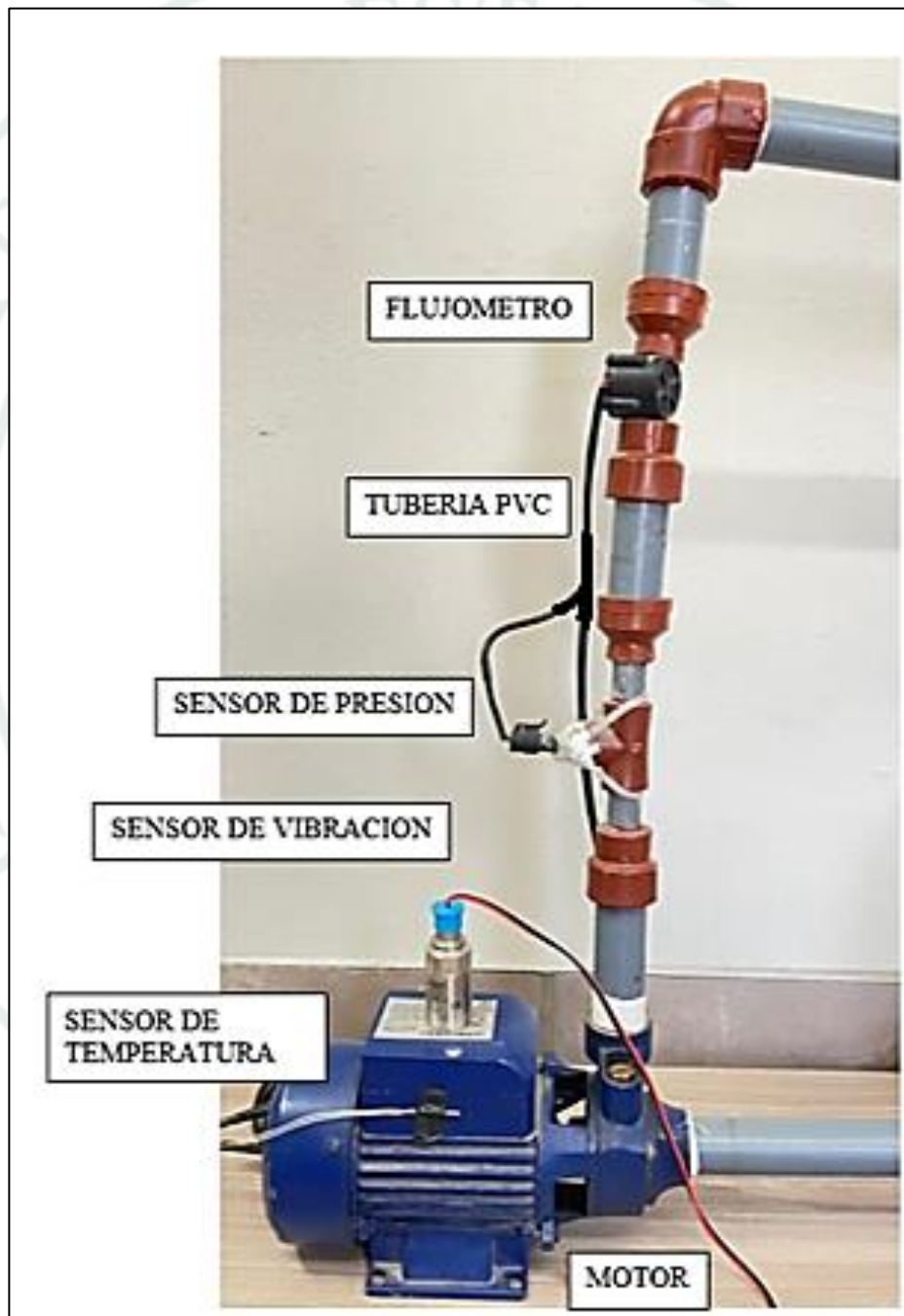
Sistema Piloto



Distribución de sensores que se dispuso en el prototipo para realizar las mediciones de los parámetros principales del monitoreo por condición de una bomba.

Figura 45

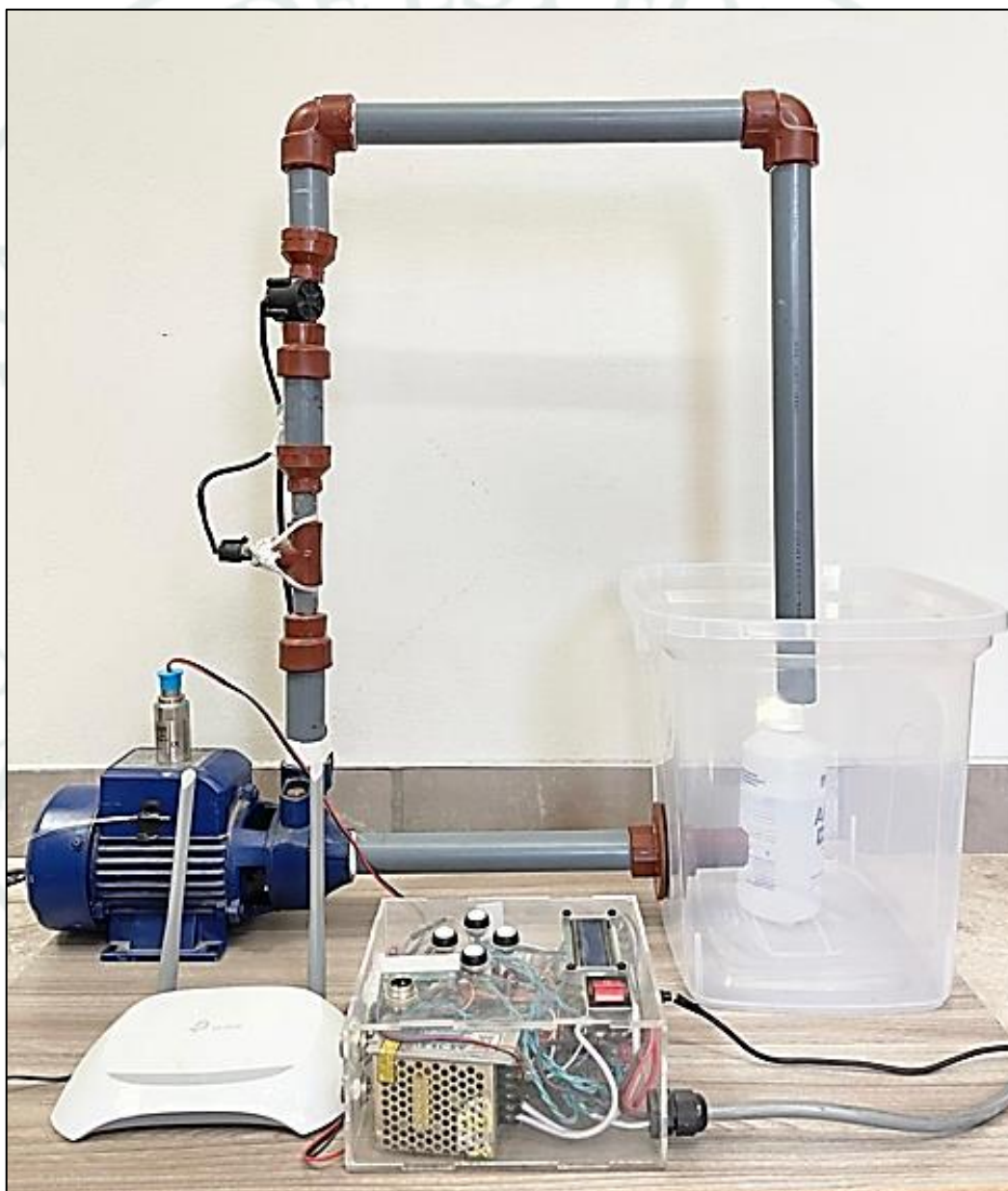
Distribución de Sensores



Distribución de todo el sistema completo de componentes del prototipo para realizar las mediciones de los parámetros principales del monitoreo por condición de una bomba.

Figura 46

Sistema completo para realizar pruebas





CAPITULO VI
PRUEBAS Y RESULTADOS

Con el equipo ya implementado se desarrolló un plan de pruebas completo y funcional con el propósito de monitorear las condiciones de operación para la obtención de datos confiables.

El plan contempla la adquisición y visualización de parámetros eléctricos (tensión y corriente), hidráulicos (presión y caudal) y mecánicos (vibración y temperatura), con el fin de obtener estos parámetros reales de operación de bomba de agua a una escala reducida.

6.1 Objetivos de la prueba:

- Verificar comportamiento de un sistema piloto a escala reducida para la obtención de datos con el prototipo implementado y su visualización mediante dashboards generados.
- Visualizar las condiciones de operación eléctricas (tensión, corriente), mecánicas (vibración, temperatura,) e hidráulicos (presión, caudal)
- Validar que la instrumentación funcione correctamente y procedimiento de adquisición de datos.
- Guardar registros de data de las condiciones de operación.

6.2 Equipamiento requerido

6.2.1 Características del equipo

Temperatura de trabajo de sensor RTC	0 C° a 200 C°
Corriente máxima de entrada alterna	50A
Rango máximo Voltaje de entrada alterna	600V
Frecuencia de muestreo de sensor de flujo	40 Khz
Pantalla LCD de 16x2 caracteres	LCD de 16x2

6.2.3 Especificaciones del equipo

Capacidad de almacenamiento:	16 GB
Funcionamiento a batería:	No
Material	Fibra de vidrio
Temperatura Operación del equipo:	0 – 100°C
Humedad Relativa del equipo:	95% sin condensación
Tanque de almacenamiento y tuberías PVC	15L / 1”

6.2.4 Especificaciones del sistema de medición y procesamiento

Procesador:	Hércules RM46L852
Número de Bits:	32 bits
Velocidad del procesador:	365 DIPS/ MHz
Conversor ADC:	12 bits / 3.3 v
Memoria Flash:	1.25 MB
Memoria RAM:	192 KB
Memoria EEPROM:	64 KB

6.2.5 Especificaciones del módulo de visualización

El módulo de interfaz para el usuario es una pantalla LCD que se utiliza para:

Visualización de datos Corriente, Voltaje, Menús de Navegación,
presión, vibración, flujo, temperatura

Entrada 4 botoneras

6.2.6 Hoja de Especificaciones Técnicas del Prototipo de Monitoreo

Las siguientes especificaciones consolidan los parámetros técnicos del prototipo para el monitoreo de bombas centrífugas, integrando las características del hardware de adquisición, los módulos de comunicación, los sensores instalados y las condiciones de operación del equipo. Esta hoja sirve como documento de referencia técnica para la verificación del cumplimiento de los objetivos de diseño y como base para la escalabilidad del sistema a entornos mineros industriales.

Tabla 12

Hoja de especificaciones técnicas del prototipo de monitoreo de bombas centrífugas

PARÁMETRO	ESPECIFICACIÓN
Modelo / Nombre del equipo	Prototipo de monitoreo de bombas centrífugas
Microcontrolador principal	TMS570LS3137 (Hércules RM46L852) – 32 bits, 180 MHz
Módulo de comunicación inalámbrica	ESP32 – Wi-Fi 802.11 b/g/n, protocolo TCP
Resolución del ADC	12 bits / 3.3 V
Frecuencia de muestreo máxima	< 10 ms (60 μ s)
Memoria Flash	1.25 MB
Memoria RAM	192 KB
Memoria EEPROM	64 KB
Tensión de alimentación del sistema	220 V AC (convertido internamente a 24 V DC)
Rango de medición de voltaje AC	0 – 600 V (fase a neutro)

Rango de medición de corriente AC	0 – 50 A (sensor ACS756-050A)
Rango de temperatura (sensor RTD PT100)	0 °C a 200 °C
Rango de presión (sensor HK3025)	0 – 5.0 MPa (50 bar)
Rango de caudal (sensor TF-S201)	1 – 30 L/min
Rango de vibración (sensor PC420VR-05)	0 – 250 g pico; ancho de banda: 3.5 Hz – 2.0 kHz
Salida estándar sensores de campo	4 – 20 mA (conversión mediante módulo SEN0262)
Interfaz de visualización local	Pantalla LCD 16 x 2 caracteres
Almacenamiento de datos	SQL Server (base de datos relacional en red local)
Visualización remota	Dashboards desarrollados en Python
Material del gabinete	Fibra de vidrio
Temperatura de operación del equipo	0 – 100 °C
Humedad relativa máxima	95 % sin condensación
Capacidad de almacenamiento local	16 GB
Dimensiones del tanque de prueba	15 L / tuberías PVC 1"
Costo total del prototipo	S/ 2,653.90 (aprox. USD 720.00)

Nota: Elaboración propia a partir del diseño, selección de componentes y resultados de pruebas del prototipo.

6.3 Especificaciones de parámetros de muestreo

- Señales lentas (presión, caudal, temperatura) - tensión, corriente
- Vibración: Análisis de la forma de onda y el espectro
 - Tiempo de muestreo (Δt): $60 \mu s = 60 \times 10^{-6}$
 - Número de muestras (N): 1028
 - Duración total registrada:

Figura 47

Forma de onda – Análisis Temporal de vibración

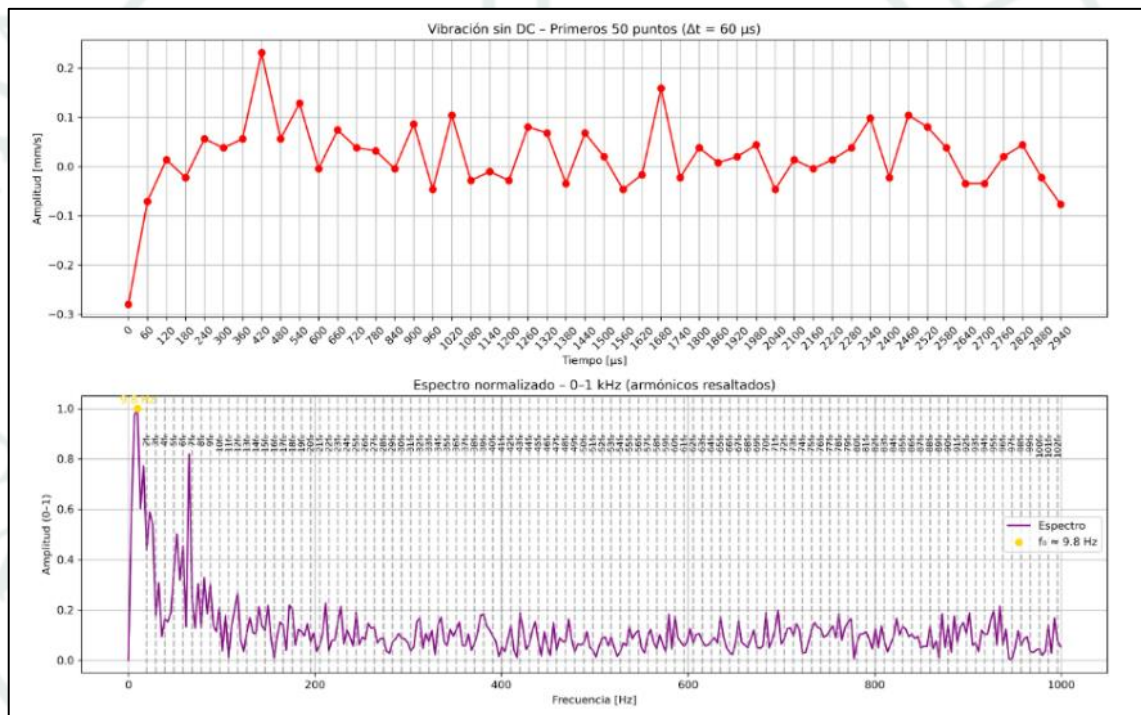
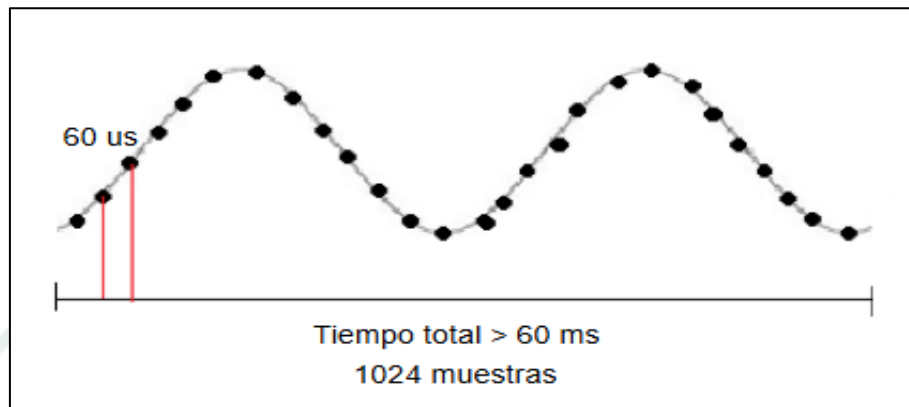


Figura 48

Forma de onda – Análisis Temporal de vibración



$$Tt = Ts \times N \text{ muestras}$$

$$Tt = 60 \text{ us} \times 1024$$

$$Tt = 61440 \text{ us}$$

$$Tt = 61.44 \text{ ms}$$

Calculamos:

Paso 1: Calcular la frecuencia de muestreo (F_s)

$$F_s = \frac{1}{T_s} = \frac{1}{60 \times 10^{-6}} = 16666 \text{ Hz}$$

Paso 2: Calcular el espectro de frecuencia disponible

$$\Delta f = \frac{F_s}{N} = \frac{16666}{1024} = 16,28 \text{ Hz}$$

Y la frecuencia máxima observable (Nyquist)

$$f_{max} = \frac{F_s}{2} = \frac{16,28}{2} = 8333.33 \text{ Hz}$$

Paso 3: Que armónico podemos ver

Supongamos que la frecuencia fundamental de 60 Hz, el número máximo armónico visible este dado:

$$N_{max} = \frac{f_{max}}{60} = \frac{8333.33}{60} = 138$$

Para nuestro sensor de vibración con una frecuencia fundamental de 1 kHz, manteniendo los mismos parámetros de adquisición de datos:

- Tiempo de muestreo: 60 us
- Numero de muestras: 1024
- Que armónico podemos ver respecto a 1 kHz

Supongamos que la frecuencia fundamental de 1kHz, el número máximo armónico visible este dado:

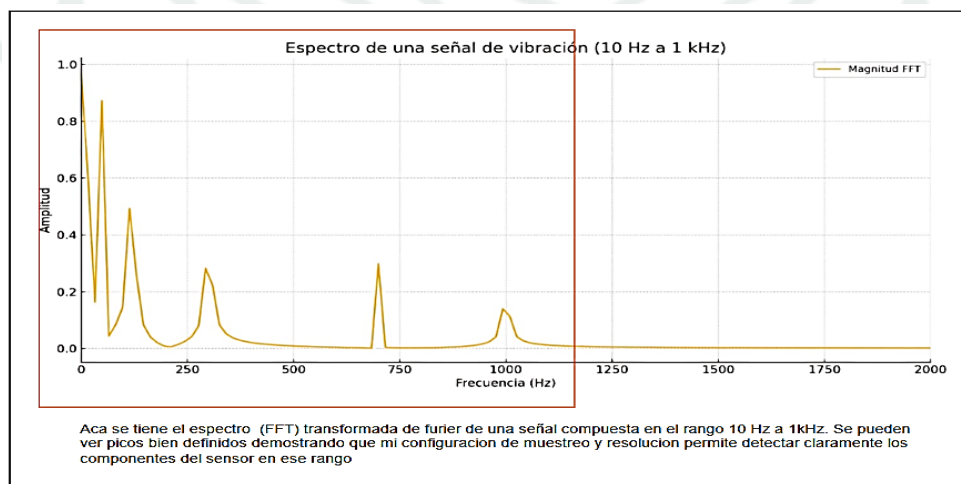
$$f_n = n \times 1000 \text{ Hz}$$

Entonces, el número máximo de armónico visible es:

$$n = \frac{8333.33}{1000} = 8$$

Figura 49

Forma de onda – Análisis Espectral (FFT) Armónicos



Consideraciones:

No podemos observar el 9º armónico porque está fuera de rango Nyquist

Para observar más armónicos necesitamos:

- Reducir el tiempo de muestreo a 20 μ s y una frecuencia de muestreo (F_s)= 50 kHz
- Usar menos muestras por ventana, si solo nos interesa mayor ancho de banda sacrificando resolución.

6.3.1 Análisis de Frecuencia: Bomba en Buen Estado vs. Bomba con Anomalías

El análisis en el dominio de la frecuencia (FFT – Transformada Rápida de Fourier) es la herramienta fundamental del diagnóstico por condición en bombas centrífugas. Permite identificar componentes frecuenciales características asociadas a distintos estados de operación del equipo: funcionamiento nominal o presencia de fallos mecánicos y eléctricos.

El prototipo desarrollado adquiere señales de vibración, voltaje y corriente con un tiempo de muestreo de 60 μ s ($F_s = 16\,666$ Hz) y 1 024 muestras por ventana, procesadas mediante FFT en el entorno Python. Los resultados del análisis espectral confirman la capacidad del sistema para discriminar entre condición nominal y condición de anomalía.

6.3.1.1 Parámetros de adquisición para el análisis espectral

- Frecuencia de muestreo (F_s): 16.66 kHz ($T_s = 60$ μ s)
- Número de muestras por ventana (N): 1 024
- Duración total de ventana ($T_t = T_s \times N$): 61.44 ms
- Resolución frecuencial ($\Delta f = F_s/N$): 16.28 Hz
- Frecuencia de Nyquist ($f_{\max} = F_s/2$): 8.333 kHz

6.3.1.2 Hoja de análisis comparativo de frecuencias

La Tabla 13 y 14 presenta el análisis de frecuencias comparativa entre la condición nominal (bomba en buen estado) y la condición de anomalía (bomba con falla inducida), a partir de los espectros FFT obtenidos con el prototipo de monitoreo.

Tabla 13

*Análisis de frecuencia comparativo: bomba en buen estado**

Frecuencia (Hz)	Amplitud relativa	Componente	Interpretación
BOMBA EN BUEN ESTADO (condición nominal – operación estable)			
60 Hz	Alta (fundamental)	Fundamental red eléctrica	Frecuencia de línea – normal
120 Hz	Baja (<10 % fund.)	2° armónico eléctrico	Residual esperado – aceptable
180 Hz	Muy baja (<5 %)	3° armónico eléctrico	Nivel admisible
< 10 Hz	Baja-media	Vibración mecánica baja frec.	Sin cavitación ni desbalance
60 Hz	Estable	Vibración mecánica sincrónica	Relacionada a velocidad de giro

No detectada	Nula o imperceptible	Armónico de paso de álabes	Sin anomalía estructural
--------------	----------------------	----------------------------	--------------------------

Nota: Los valores de amplitud son relativos al espectro FFT obtenido. Elaboración propia a partir de los espectros generados en Python

Tabla 14

*Análisis de frecuencia comparativo: bomba con anomalía**

Frecuencia (Hz)	Amplitud relativa	Componente	Interpretación
BOMBA CON ANOMALÍAS (condición de falla – referencia comparativa)			
60 Hz	Elevada / distorsionada	Fundamental con distorsión	Sobretensión o falla eléctrica
120 / 180 Hz	Alta (>15 % fund.)	Armónicos eléctricos elevados	Desbalance de fases / sobrecarga
Subarmónicos < 30 Hz	Presente	Vibración mecánica irregular	Desalineamiento del impulsor

Banda 100–500 Hz	Picos irregulares	Vibración de alta energía	Cavitación incipiente
Armónicos de paso álabes	Detectados (BPF)	$n \times \text{RPM} \times N^\circ$ álabes	Desgaste o rotura de álabes
Banda aleatoria > 1 kHz	Ruido elevado	Ruido de banda ancha	Cavitación avanzada / fricción

Nota: Los valores de amplitud son relativos al espectro FFT obtenido. Elaboración propia a partir de los espectros generados en Python

6.3.1.3 Interpretación técnica de los resultados espectrales

Los resultados obtenidos en el Paso 4 de la matriz de pruebas (análisis FFT en operación nominal) muestran un espectro de voltaje dominado por la componente fundamental de 60 Hz con armónicos de baja amplitud (< 10 % del fundamental), coherente con un suministro eléctrico de calidad. La señal de corriente exhibe la misma componente fundamental con armónicos de tercer orden ligeramente más pronunciados, lo cual es normal en motores inductivos monofásicos bajo carga parcial.

La señal de vibración en condición nominal concentra su energía por debajo de 10 Hz, sin componentes significativas en bandas altas, confirmando la ausencia de cavitación, desbalance o desgaste de rodamientos. Este patrón establece la línea base espectral (baseline) del equipo, a partir de la cual se definirán los umbrales de alarma para la implementación de mantenimiento predictivo.

En la condición de anomalía simulada (desalineamiento del impulsor – Paso 5 de la matriz de pruebas), se registró una elevación significativa de la corriente acompañada de un incremento en la amplitud de vibración y la aparición de subarmónicos por debajo de 30 Hz, patrón diagnóstico característico del desalineamiento mecánico. La condición de cavitación (Paso 7) introdujo picos espectrales irregulares en la banda de 100 – 500 Hz y ruido de banda ancha por encima de 1 kHz, consistentes con el colapso de burbujas de vapor dentro de la carcasa de la bomba.

Este análisis comparativo valida que el prototipo posee la resolución temporal y frecuencial necesaria para discriminar condiciones de operación nominal frente a modos de falla representativos en bombas centrífugas, cumpliendo con el objetivo de monitoreo predictivo definidos.

6.4 Matriz de pruebas

Se procede a realizar pruebas para determinar el correcto funcionamiento del prototipo desarrollado, se realizaron pruebas experimentales en el sistema piloto y prototipo, midiendo las principales variables operativas de la bomba centrífuga. Durante estas pruebas se registraron señales de voltaje, corriente y vibración con un muestreo a 60 us, permitiendo verificar la capacidad del sistema de adquisición y transmisión de datos en tiempo real; así mismo los parámetros de presión temperatura y flujo que validan que el sistema opera de manera estable en condiciones nominales, lo que validará la capacidad del prototipo para integrar diferentes variables en una misma plataforma de monitoreo mediante la visualización de dashboards y visualización de ondas mediante la recolección de data almacenada.

Se realiza cada escenario planteado y se registra las condiciones obtenidas:

Tabla 15

Matriz de Pruebas

PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
1	Verificación de puesta en marcha (sin carga ni energía)	Revisión externa del prototipo y sistema de bombeo	Verificar el correcto montaje, fugas, alineamiento de sensores
2	Picos de arranque	Se enciende se apaga el motor conectándolo al prototipo de 220 V	Visualización de picos de arranque del prototipo
3	Visualización de formas de onda en Dashboards	Bomba operando a velocidad nominal y recirculación abierta.	Señal estable, visualización de formas de onda en Dashboards
4	Análisis espectral (FFT)	Visualización de formas de onda en Dashboards mediante Python	Visualización de picos de arranque del prototipo
5	Desalineamiento de impulsor	Se induce de manera controlada un ligero contacto entre el impulsor y la carcasa de la bomba	Evaluar el efecto del desalineamiento del impulsor sobre las condiciones eléctricas, térmicas y mecánicas del sistema piloto.
6	Flujo reducido / bloqueo parcial	La condición se simula mediante la introducción controlada de un objeto que genere una obstrucción parcial en la tubería.	Detectar caída de flujo, posibles cavitaciones

7	Cavitación	Se simula mediante la reducción controlada del nivel de agua en el tanque de succión, disminuyendo así la presión de entrada hacia la bomba.	Ruidos, picos de vibración en banda característica, revisar límites.
8	Fallo de sensor / pérdida comunicación	Desconectar temporalmente un sensor / motor	Verificar la capacidad del sistema para detectar la pérdida de señal y evaluar su comportamiento durante y después del evento de desconexión.
9	Pérdida de comunicación	Desconectar temporalmente el Wi-Fi	Comprobación de almacenamiento local de los datos guardados
10	Visualización en la tarjeta LCD	Se visualiza prendiendo prototipo de monitoreo a 220 V	Visualizar los parámetros de operación en tiempo real

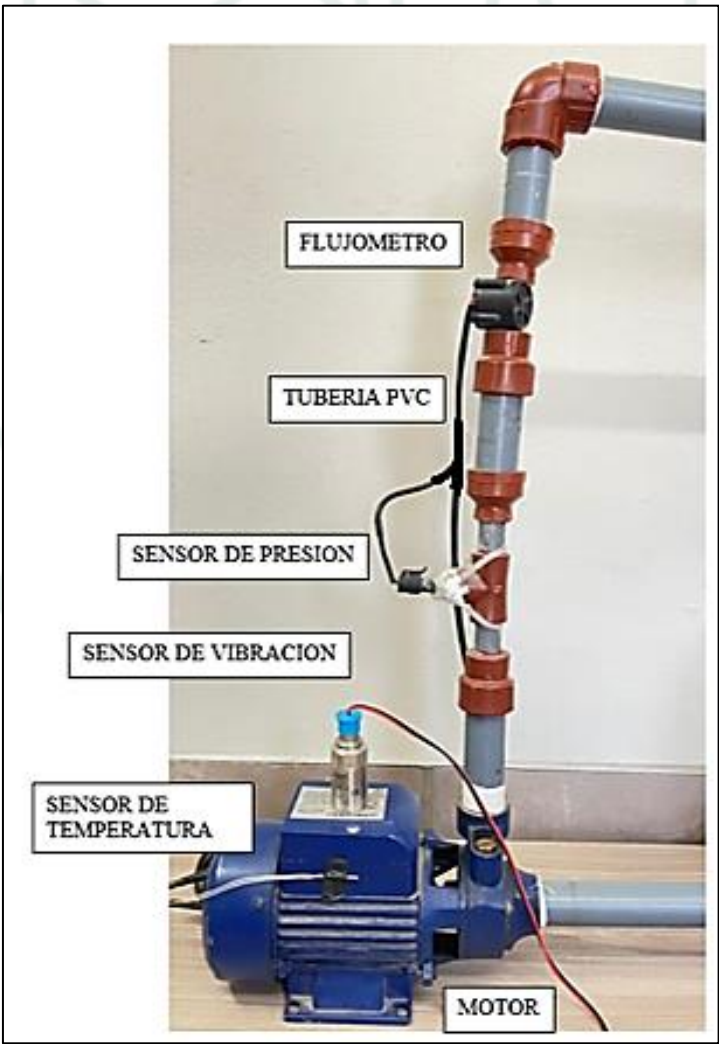
Nota: Tabla de matriz de pruebas del sistema de monitoreo

6.5 Inicio de pruebas y resultados:

PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
1	Verificación de puesta en marcha (sin carga ni energía)	Revisión externa del prototipo y sistema de bombeo	Verificar el correcto montaje, fugas, alineamiento de sensores

Se verifico el correcto montaje de los sensores visualizando cero fugas y alineamiento así mismo, realizamos la conexión del prototipo a 220V y enlace al módulo Wi-Fi para visualizar su correcto funcionamiento.

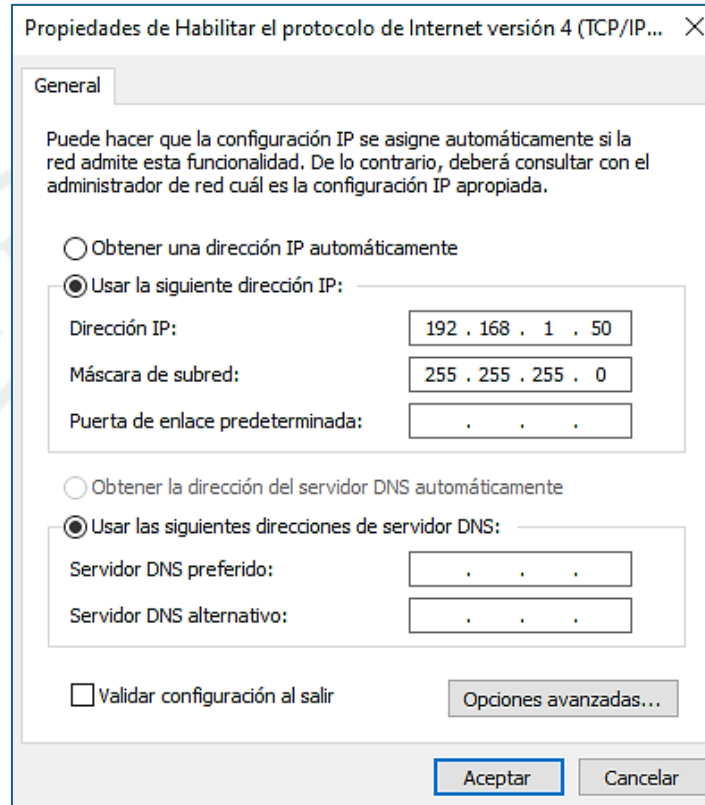
Figura 50
Verificación de sensores para puesta en marcha.



Se genera una dirección IP privada de la laptop.

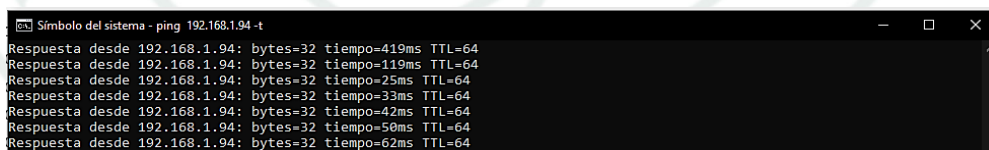
Figura 51

Dirección IP Privada

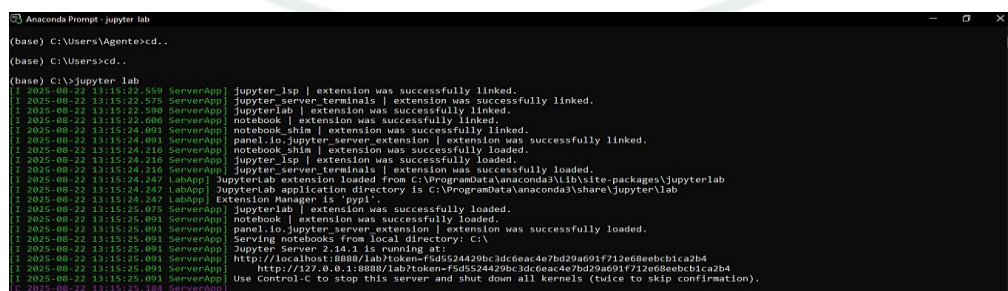


Se verifica en cmd la conexión de nuestra red privada para el acceso a nuestros dashboards.

IP privat: 192.168.1.94



Nos conectamos a Anaconda Prompt, clave jupyter lab.



A continuación, se muestra la señal adquirida por el prototipo, procesada y visualizada en los dashboards desarrollados para el sistema.

Voltaje (V): La forma de onda obtenida corresponde a una señal sinusoidal de alimentación trifásica, con un valor de RMS de 218.08 V y un valor máximo de 306.99 V, se confirma la estabilidad de la fuente eléctrica suministrada (220 V) al motor de la bomba.

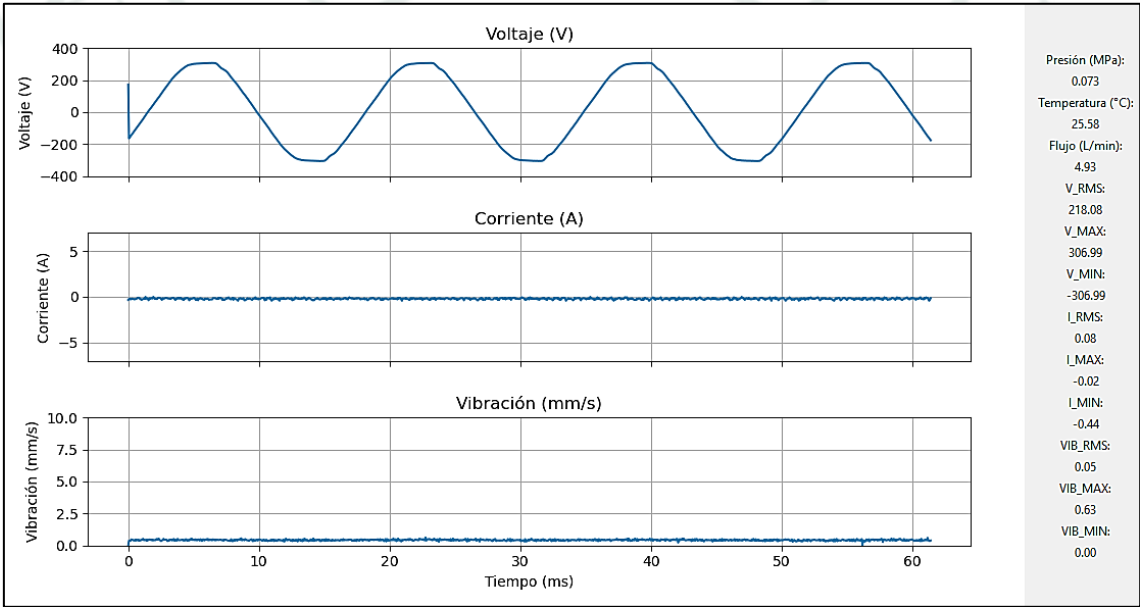
Corriente (A): La señal de corriente presenta valores de 0 A coherente con la demanda de un motor de 1 HP. Se observa un nivel estable confirmando la ausencia de carga em el sistema.

Vibración (mm/s): Señal presenta un nivel promedio bajo, sin presencia de cavitación, desbalance o desgaste significativos de las partes húmedas.

Además, los parámetros de **presión, temperatura y flujo** validan la ausencia de carga en el sistema.

Figura 52

Forma de Onda - Prototipo conectado



PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
2	Picos de arranque	Se enciende se apaga el motor conectándolo al prototipo de 220 V	Visualización de picos de arranque del prototipo

Análisis técnico de los picos de arranque del prototipo:

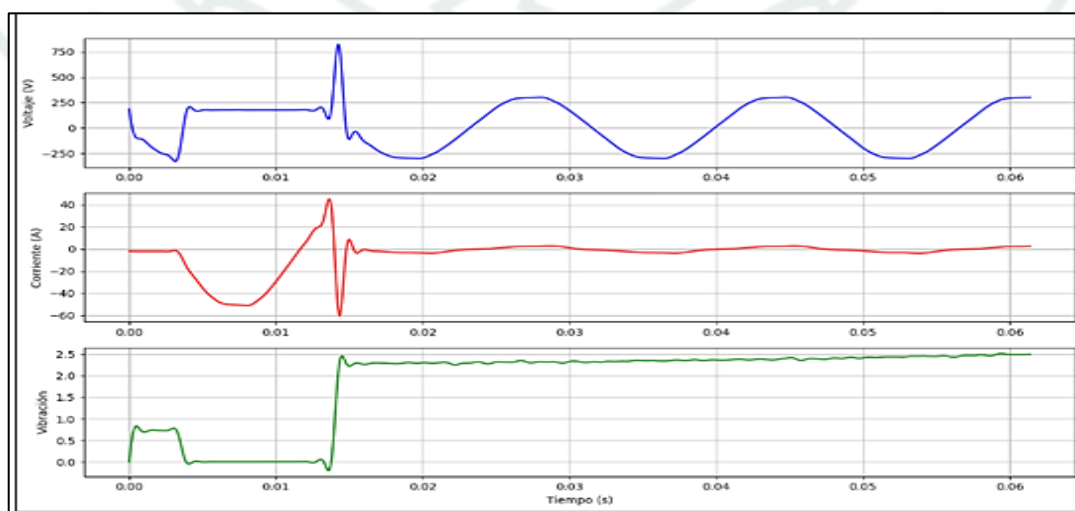
Voltaje (V): Se visualiza en que los primeros milisegundos se observa un pico de sobretensión que supera los 760 V, lo cual indica la transición brusca de estado apagado a encendido. La señal se estabiliza en una forma senoidal, correspondiente al régimen normal de operación.

Corriente (A): Se evidencia un pico de corriente de arranque cercano a ± 40 A, seguido de oscilaciones transitorias. Luego la corriente disminuye y tiende a estabilizarse alrededor de un valor más bajo, propio del régimen de operación constante, donde el equipo eléctrico demanda mayor energía de arranque para vencer la inercia mecánica.

Vibración (mm/s): Se visualiza la vibración está cercana a 0 mm/s; al momento del arranque, se produce un escalón brusco, alcanzando valores en torno a 2.4 mm/s, que se mantienen constantes durante su operación debido al impulso mecánico.

Figura 53

Forma de onda – Picos de arranque



PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
3	Visualización de formas de onda en Dashboards	Bomba operando a velocidad nominal y recirculación abierta.	Señal estable, visualización de formas de onda en Dashboards

Se muestran las formas de ondas de voltaje, corriente y vibración adquiridas por el prototipo con el sistema en operación en los dashboards generados.

Voltaje (V): La forma de onda obtenida corresponde a una señal sinusoidal de alimentación trifásica, con un valor de RMS de 213.66 V y un valor máximo de 344.61 V, se confirma la estabilidad de la fuente eléctrica suministrada (220 V) al motor.

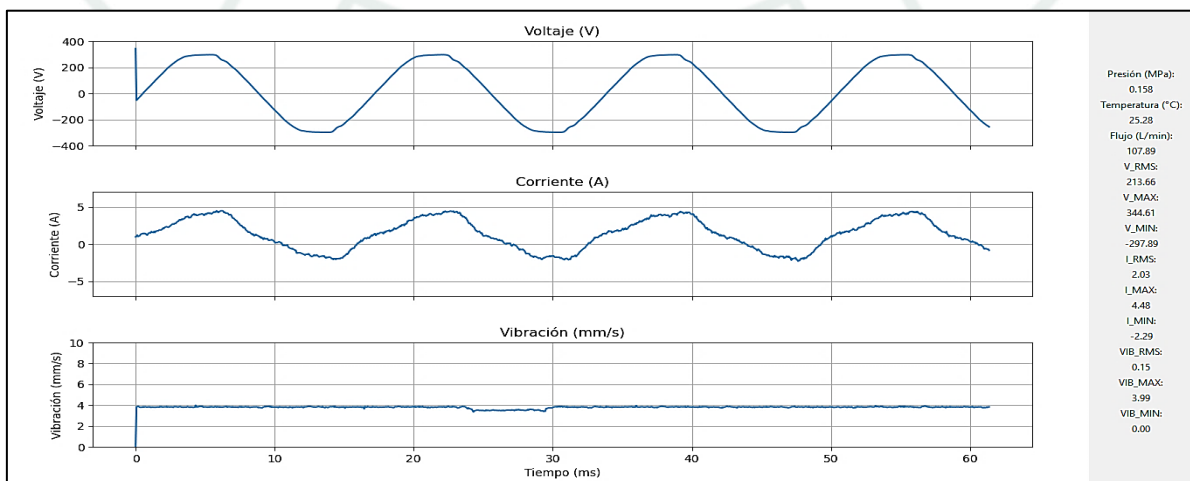
Corriente (A): La señal de corriente presenta valores de 2.03 A coherente con la demanda de un motor de 1 HP y un valor máximo de 4.48 A. La estabilidad de la onda evidencia que no existen transitorios eléctricos ni sobrecargas durante la operación.

Vibración (mm/s): La señal de vibración presenta valores de 0.15 mm/s un nivel promedio bajo y un valor máximo de 3.99 mm/s, confirmando la no presencia de cavitación, desbalance o desgaste significativos de las partes húmedas.

Además, los parámetros de **presión (0,158 MPa)**, **temperatura (25,58 °C)** y **flujo (107,89 L/min)** validan que el sistema opera de manera estable en condiciones nominales.

Figura 54

Verificación de formas de onda en Dashboards



PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
4	Análisis espectral (FFT)	Visualización de formas de onda en Dashboards mediante Python	Visualización de picos de arranque del prototipo

Análisis técnico de la forma de onda en el dominio de la frecuencia.

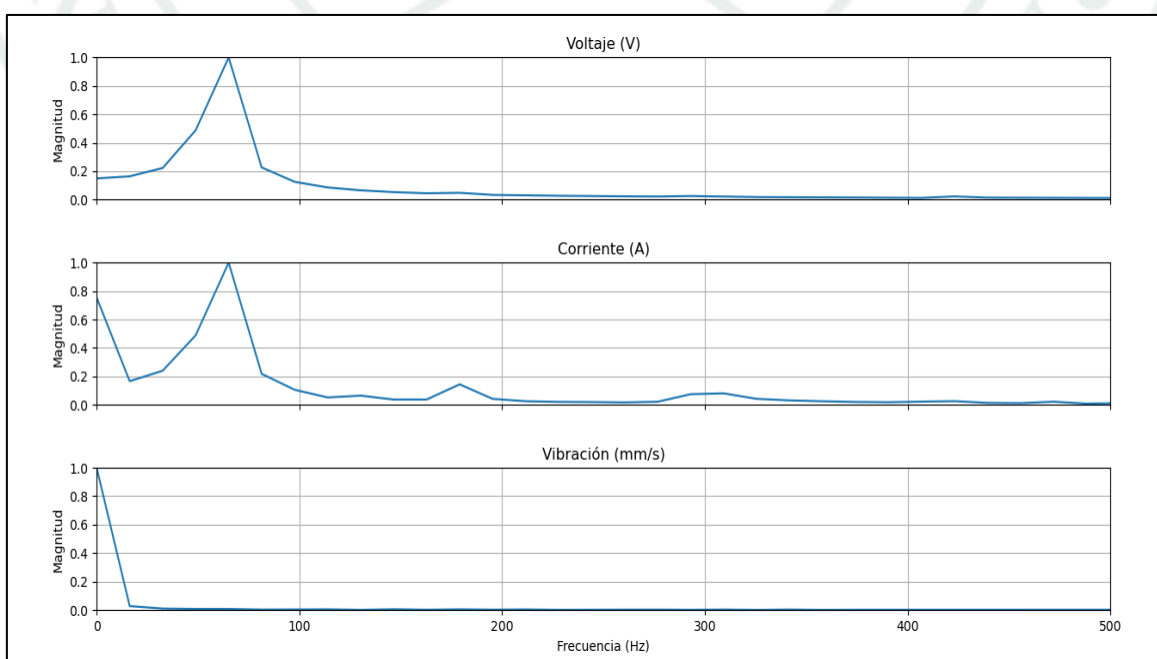
Voltaje (V): Se observa una forma de onda de un pico principal cercano a 60 Hz, lo cual corresponde a la frecuencia fundamental de la red eléctrica; existen armónicos adicionales de menor magnitud, pero con amplitud muy reducida respecto a la fundamental.

Corriente (A): Presenta un pico fuerte en la fundamental (≈ 60 Hz). Se visualiza una diferencia del voltaje, se aprecian componentes armónicas adicionales más pronunciadas (por ejemplo, alrededor de 120 Hz y 180 Hz).

Vibración (mm/s): Corresponde a la mayor parte de la energía está en baja frecuencia (<10 Hz). Esto indica que la vibración registrada no proviene de resonancias mecánicas a alta frecuencia, sino de componentes lentas del arranque o desbalances de baja velocidad.

Figura 55

Forma de onda – Análisis espectral (FFT)

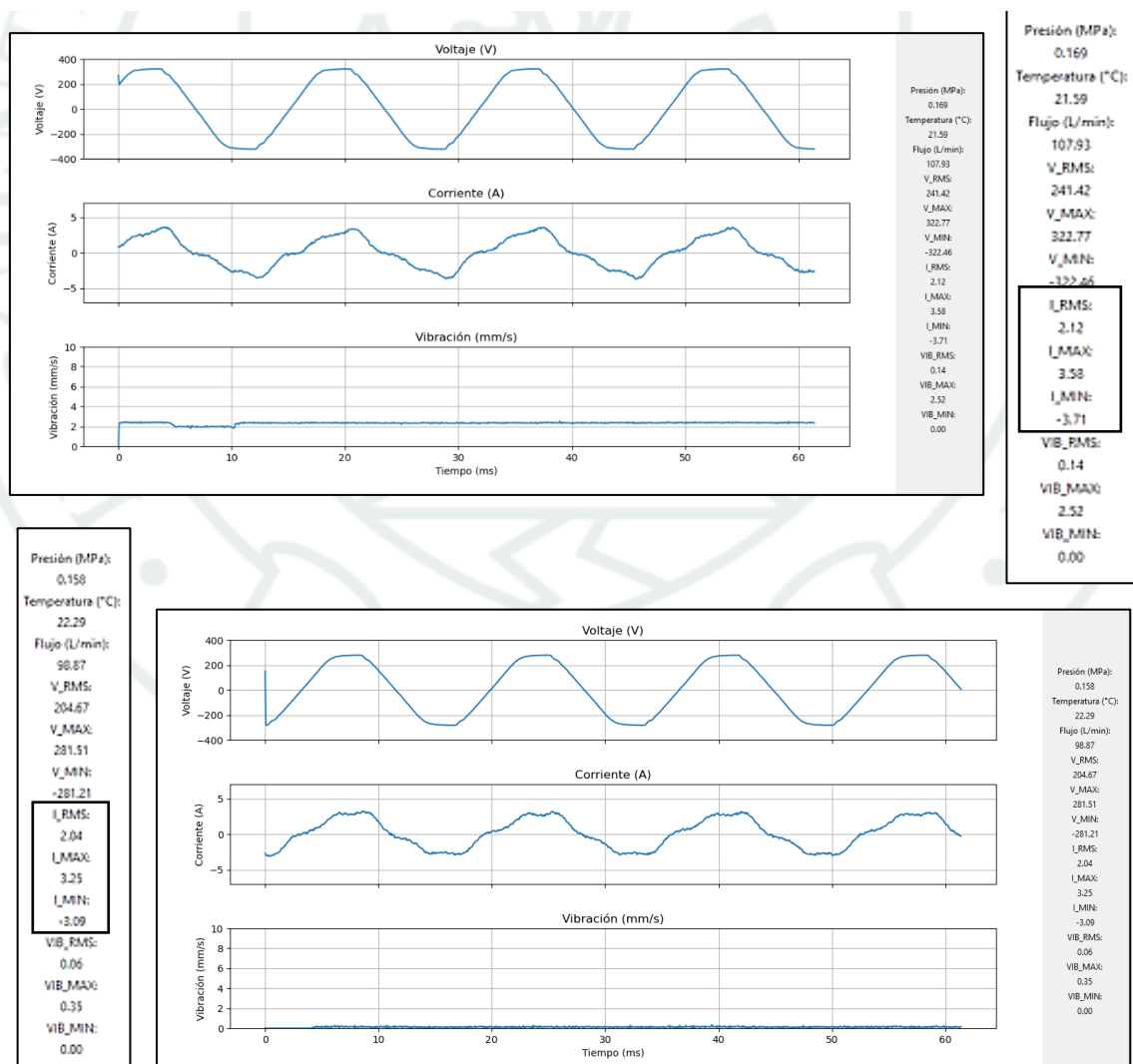


PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
5	Desalineamiento de impulsor	Se induce de manera controlada un ligero contacto entre el impulsor y la carcasa de la bomba	Evaluar el efecto del desalineamiento del impulsor sobre las condiciones eléctricas, térmicas y mecánicas del sistema piloto.

Durante la prueba, se espera observar una elevación significativa de la corriente y la temperatura, acompañada de un incremento en la amplitud de vibración. Estos cambios servirán para caracterizar la firma eléctrica y mecánica típica de una condición de desalineamiento del impulsor facilitando su detección temprana.

Figura 56

Desalineamiento de impulsor

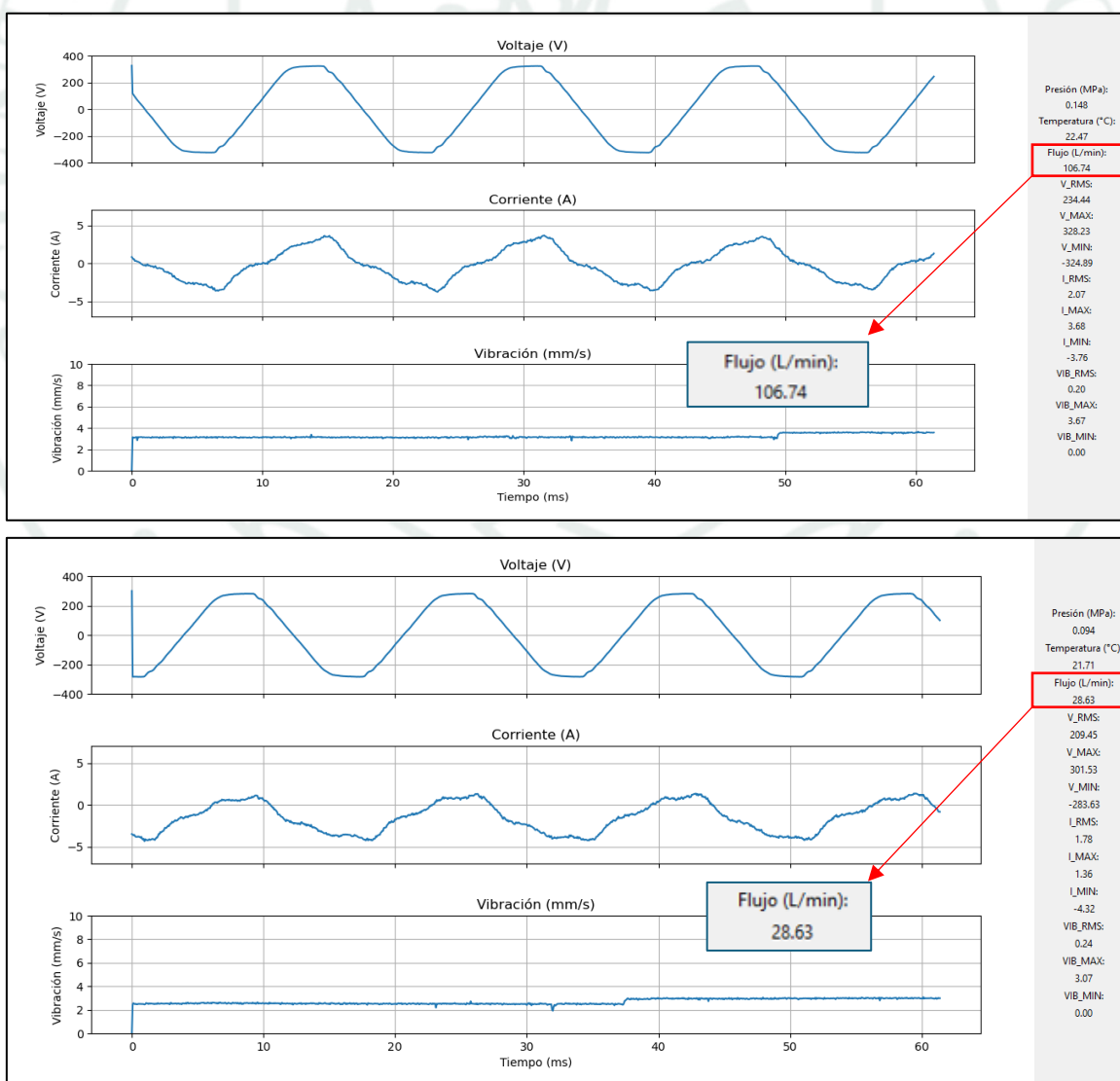


PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
6	Flujo reducido / bloqueo parcial	La condición se simula mediante la introducción controlada de un objeto que genere una obstrucción parcial en la tubería.	Detectar caída de flujo, posibles cavitaciones

Durante la prueba, se identifica una reducción significativa del flujo, acompañada de fluctuaciones de presión y vibración, que indiquen condiciones de cavitación o sobreesfuerzo hidráulico. Estos datos permitirán establecer umbrales de alarma y patrones característicos asociados al bloqueo parcial en sistemas de bombeo.

Figura 57

Flujo Reducido / Bloqueo parcial

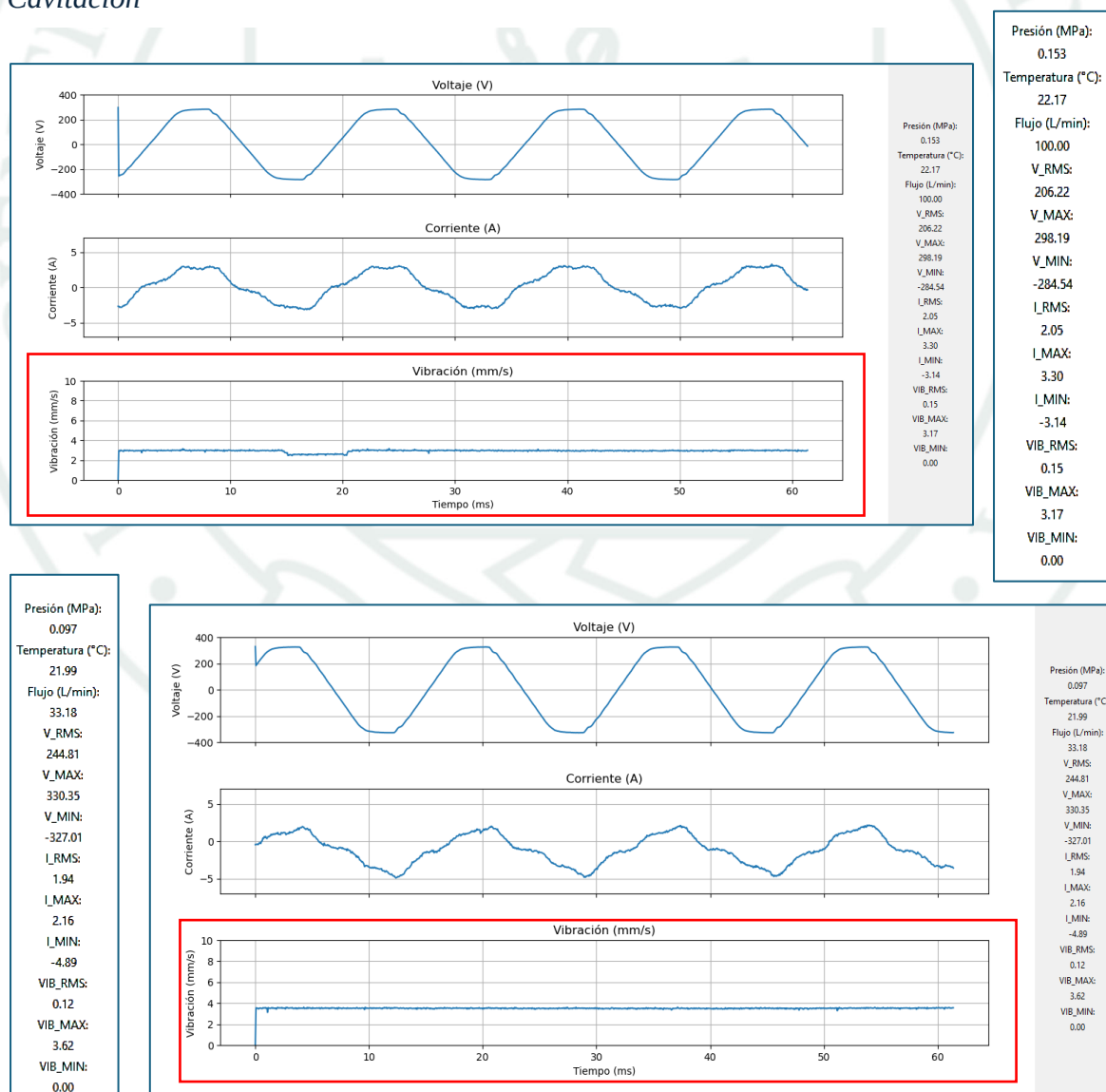


PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
7	Cavitación	Se simula mediante la reducción controlada del nivel de agua en el tanque de succión, disminuyendo así la presión de entrada hacia la bomba.	Ruidos, picos de vibración en banda característica, revisar límites.

Durante la prueba, al reducir el nivel de succión, se manifiesta ruidos intermitentes y un aumento en la amplitud de vibración. Estos indicadores servirán para establecer los umbrales de cavitación y validar la sensibilidad del sistema de monitoreo frente a este tipo de condición anómala.

Figura 58

Cavitación



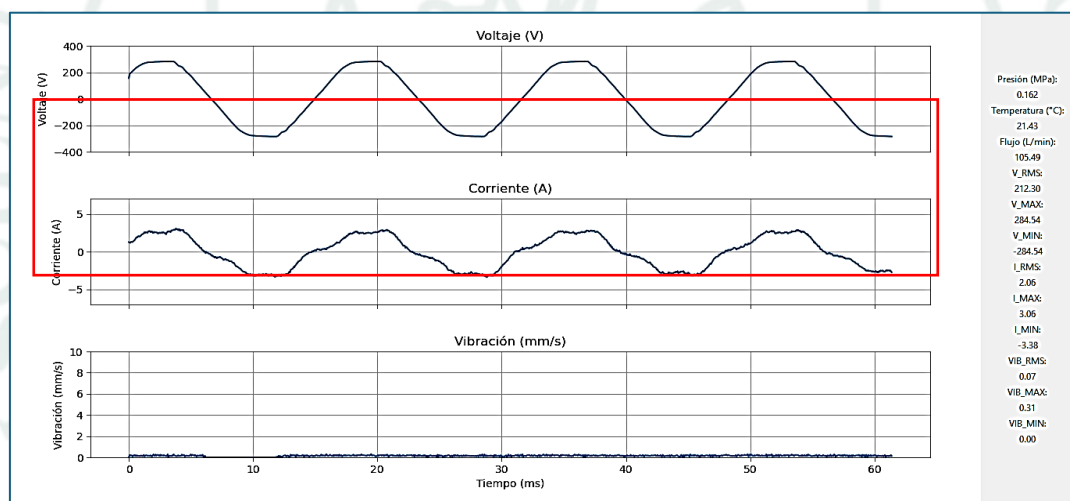
PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
8	Falla de sensor	Desconectar temporalmente un sensor / motor	Verificar la capacidad del sistema para detectar la pérdida de señal y evaluar su comportamiento durante y después del evento de desconexión.

Durante la prueba, el sistema registre la ausencia de señal o datos del sensor desconectado, mostrando la interrupción en el monitoreo en tiempo real. Al restablecer la conexión, el sistema debe reanudar la lectura de manera automática o mediante una acción de reinicio, demostrando su capacidad de detección de fallos y recuperación funcional.

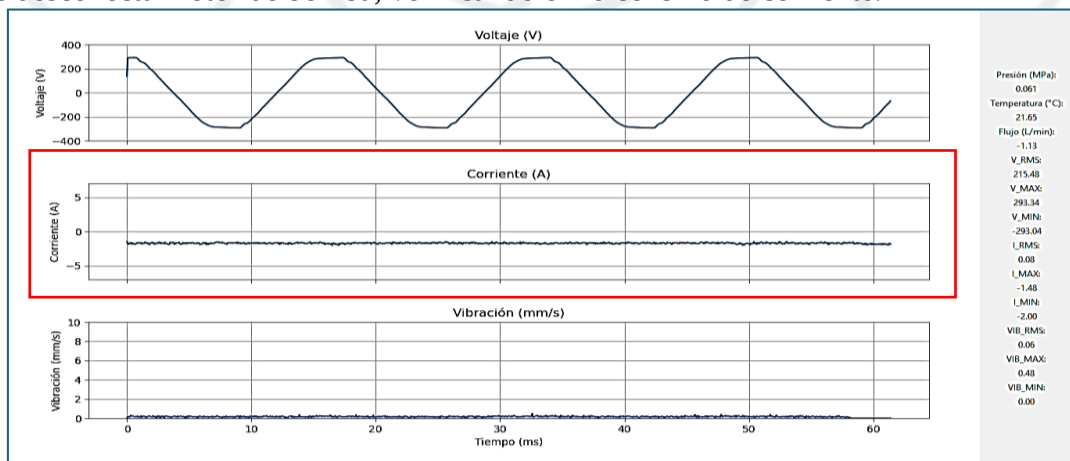
Se desconecta sensor de vibración, verificando la no lectura de vibración.

Figura 59

Falla de sensor



Se desconecta motor de bomba, verificando el no consumo de corriente.



PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
9	Pérdida de comunicación	Desconectar temporalmente el Wi-Fi	Comprobar el almacenamiento local de los datos guardados

Se muestran las formas de ondas de voltaje, corriente y vibración adquiridas por el prototipo con el sistema en operación de nuestra base de datos, que facilitan el diagnóstico del comportamiento dinámico del sistema.

Voltaje (V): Las señales obtenidas mantienen su forma sinusoidal estable.

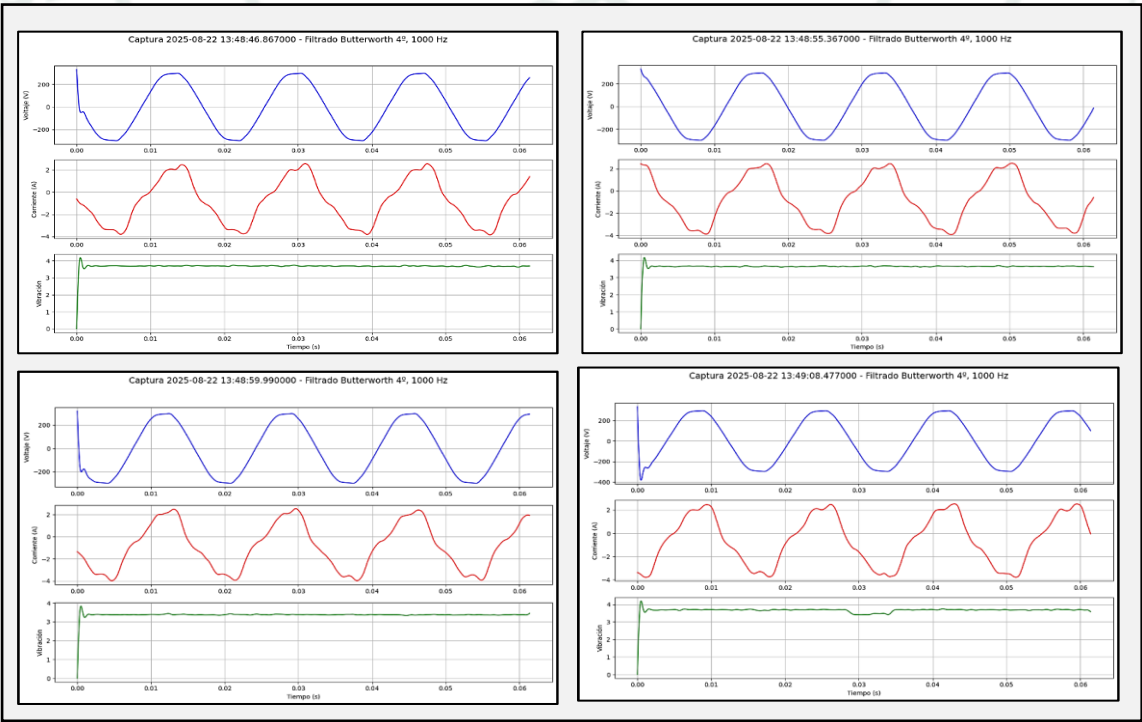
Corriente (A): La estabilidad de la onda evidencia que no existen transitorios eléctricos ni sobrecargas durante la operación.

Vibración (mm/s): La señal de vibración confirma la no presencia de cavitación, desbalance o desgaste significativos de las partes húmedas.

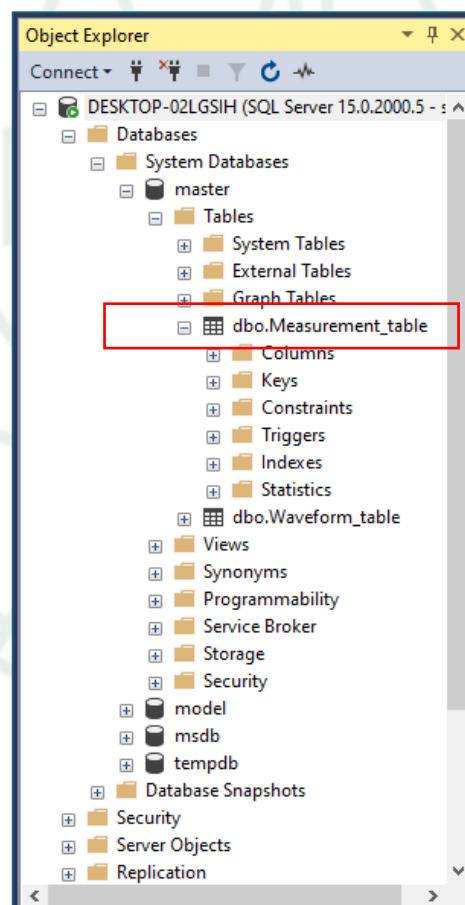
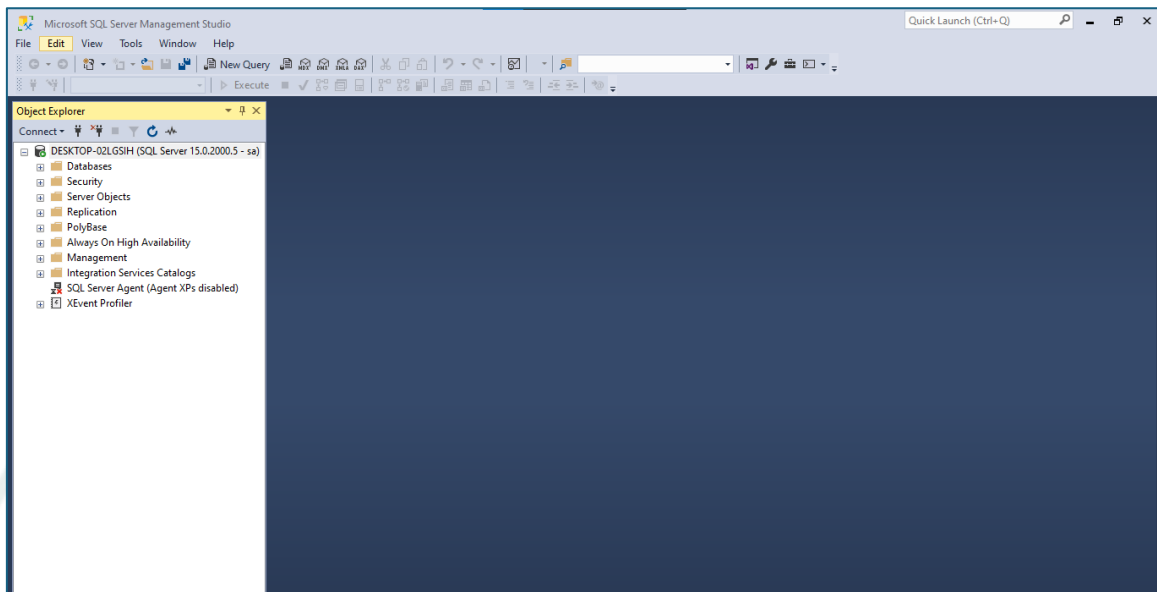
NOTA: El límite de capacidad de almacenamiento es la capacidad de la memoria de la laptop que recepcionará los datos.

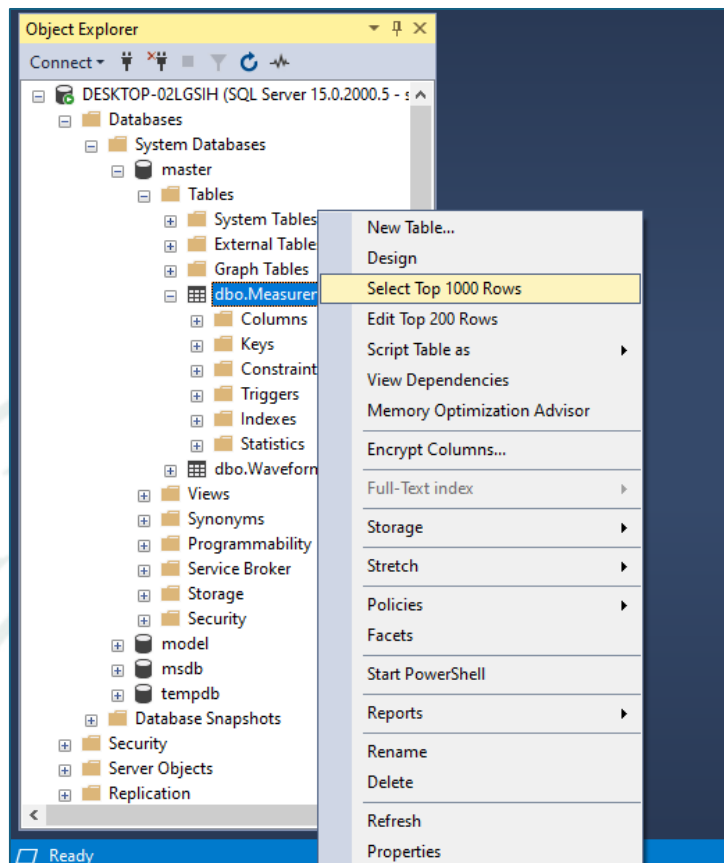
Figura 60

Forma de onda prototipo con sistema em operación – Base de datos



Se apertura el programa Microsoft SQL Server y se realiza los siguientes pasos para poder visualizar la data PI guardada





Visualización en Measurement de los siguientes parámetros Presión, Temperatura, Flujo

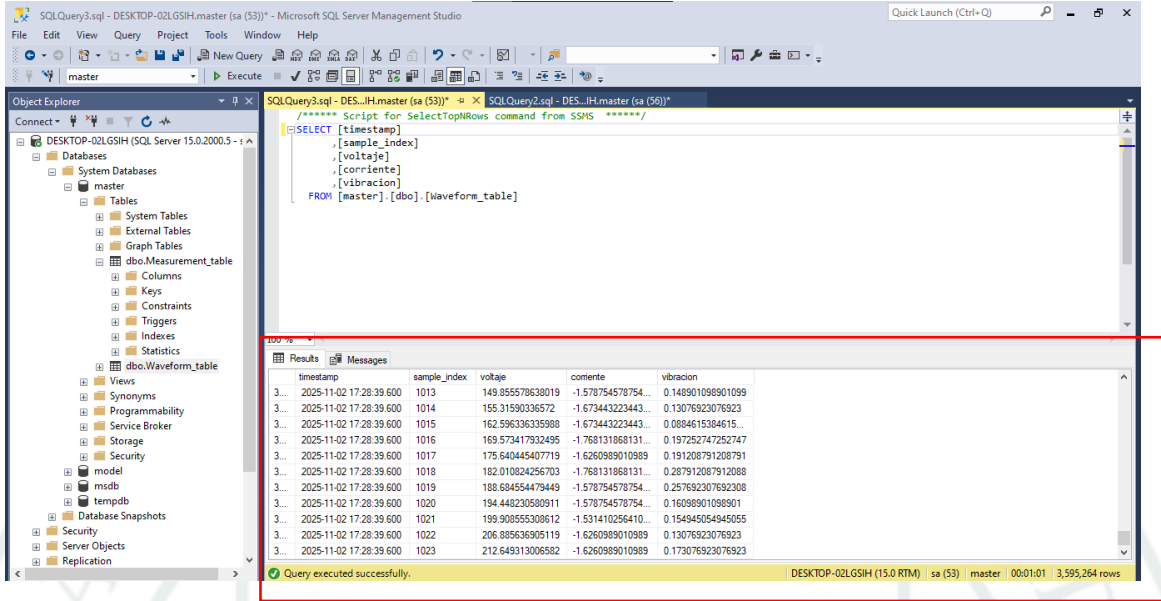
The screenshot shows the SQL Server Query Editor window. The query executed is as follows:

```

/****** Script for SelectTopNRows command from SSMS *****/
SELECT [timestamp]
,[presion]
,[temperatura]
,[flujo]
,[v_rms]
,[v_max]
,[v_min]
,[i_rms]
,[i_max]
,[i_min]
,[vib_rms]
,[vib_max]
,[vib_min]
FROM [master].[dbo].[Measurement_table]
  
```

The query results are displayed in a table with the following columns: timestamp, presion, temperatura, flujo, v_rms, v_max, v_min, l_rms, l_max. The results show data for the year 2025, with timestamps ranging from 2025-11-02 17:28:02.440 to 2025-11-02 17:28:39.600. The status bar at the bottom indicates 'Query executed successfully.' and 'DESKTOP-02LGSIH (15.0 RTM) sa (56) master 00:00:00 3,511 rows'.

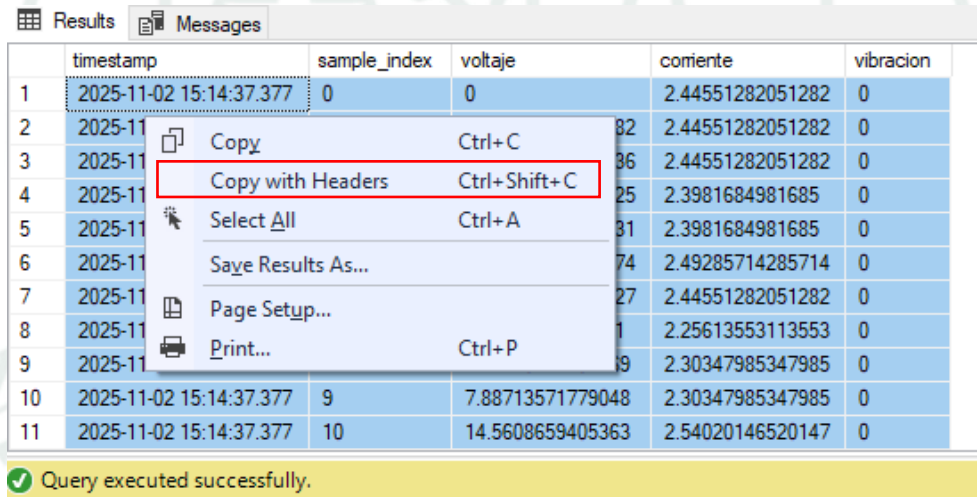
Visualización en Waveform de los siguientes parámetros Presión, Temperatura, Flujo



Query executed successfully. DESKTOP-02LGSIH (15.0 RTM) | sa (53) | master | 00:01:01 | 3,595,264 rows

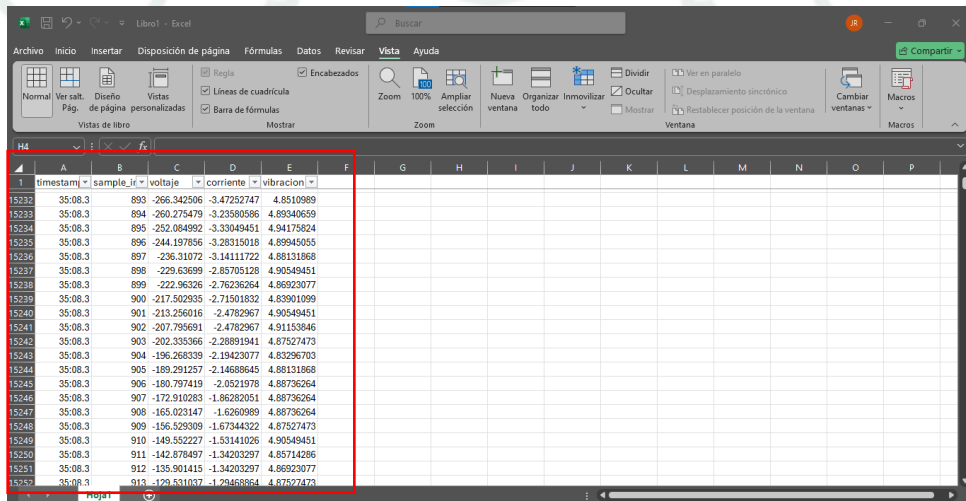
timestamp	sample_index	voltaje	corriente	vibracion
2025-11-02 17:28:39.600	1013	149.855578638019	-1.578754578754...	0.14890108901099
2025-11-02 17:28:39.600	1014	155.31590336572	-1.673443223443...	0.13076923076923
2025-11-02 17:28:39.600	1015	162.596336335988	-1.673443223443...	0.08846153846153
2025-11-02 17:28:39.600	1016	169.573417932495	-1.768131868131...	0.19725274725274
2025-11-02 17:28:39.600	1017	175.640445407719	-1.6260989010989	0.19120879120879
2025-11-02 17:28:39.600	1018	182.010824256703	-1.768131868131...	0.28791208791208
2025-11-02 17:28:39.600	1019	188.684554479449	-1.578754578754...	0.25769230769230
2025-11-02 17:28:39.600	1020	194.448230580911	-1.578754578754...	0.1609890108901
2025-11-02 17:28:39.600	1021	199.90855308612	-1.531410256410...	0.15494505494505
2025-11-02 17:28:39.600	1022	206.885636905119	-1.6260989010989	0.13076923076923
2025-11-02 17:28:39.600	1023	212.649313006582	-1.6260989010989	0.17307692307692

Se exporta los datos a Excel



Query executed successfully.

	timestamp	sample_index	voltaje	corriente	vibracion
1	2025-11-02 15:14:37.377	0	0	2.44551282051282	0
2	2025-11-02 15:14:37.377	1	7.88713571779048	2.44551282051282	0
3	2025-11-02 15:14:37.377	2	14.5608659405363	2.44551282051282	0
4	2025-11-02 15:14:37.377	3	21.2327318712616	2.3981684981685	0
5	2025-11-02 15:14:37.377	4	27.904602951832	2.3981684981685	0
6	2025-11-02 15:14:37.377	5	34.5774640229031	2.49285714285714	0
7	2025-11-02 15:14:37.377	6	41.250325103974	2.44551282051282	0
8	2025-11-02 15:14:37.377	7	47.923186175045	2.25613553113553	0
9	2025-11-02 15:14:37.377	8	54.596047246116	2.30347985347985	0
10	2025-11-02 15:14:37.377	9	61.268908317187	2.30347985347985	0
11	2025-11-02 15:14:37.377	10	67.941769388258	2.54020146520147	0



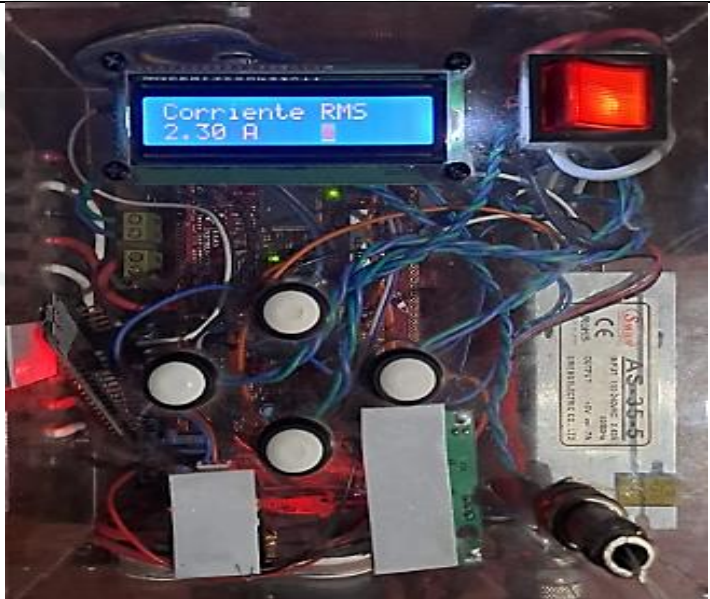
	A	B	C	D	E	F
1	timestamp	sample_index	voltaje	corriente	vibracion	
15232	35:08.3	893	-266.342506	-3.47252747	4.8510989	
15233	35:08.3	894	-260.275479	-3.23580586	4.89340659	
15234	35:08.3	895	-252.084992	-3.33049451	4.94175824	
15235	35:08.3	896	-244.197856	-3.28315018	4.89945055	
15236	35:08.3	897	-236.31072	-3.14111722	4.88131868	
15237	35:08.3	898	-229.63699	-2.85705128	4.90549451	
15238	35:08.3	899	-222.96026	-2.76236264	4.86923077	
15239	35:08.3	900	-217.502935	-2.71501832	4.83901099	
15240	35:08.3	901	-213.256016	-2.4782967	4.90549451	
15241	35:08.3	902	-207.795691	-2.4782967	4.91153846	
15242	35:08.3	903	-202.335366	-2.28891941	4.87527473	
15243	35:08.3	904	-196.268339	-2.19423077	4.83296703	
15244	35:08.3	905	-189.291257	-2.14688645	4.88131868	
15245	35:08.3	906	-180.797419	-2.0521978	4.88736264	
15246	35:08.3	907	-172.910283	-1.96262051	4.88736264	
15247	35:08.3	908	-165.023147	-1.8260989	4.88736264	
15248	35:08.3	909	-156.529309	-1.67344322	4.87527473	
15249	35:08.3	910	-149.552227	-1.53141026	4.90549451	
15250	35:08.3	911	-142.878497	-1.34203297	4.85714286	
15251	35:08.3	912	-135.901415	-1.34203297	4.86923077	
15252	35:08.3	913	-129.531037	-1.29468864	4.87527473	

PASO	ESCENARIO	COMO SIMULARLO	OBJETIVO
10	Visualización en la tarjeta LCD	Se visualiza prentiendo prototipo de monitoreo a 220 V	Visualizar los parámetros de operación en tiempo real

Mediante la recolección de datos operativos del prototipo y la transmisión por medio del ESP32 se puede visualizar en la tarjeta LCD datos obtenidos en tiempo real.

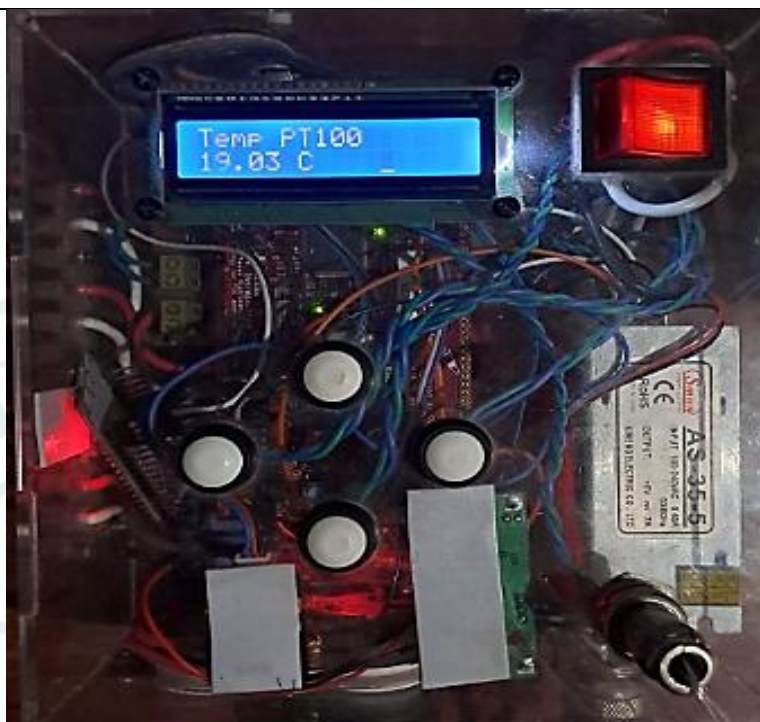
Figura 61

Visualización de parámetros en el prototipo

PARÁMETROS OPERATIVOS	
VOLTAJE 220.60 V	
CORRIENTE 2.30 A	

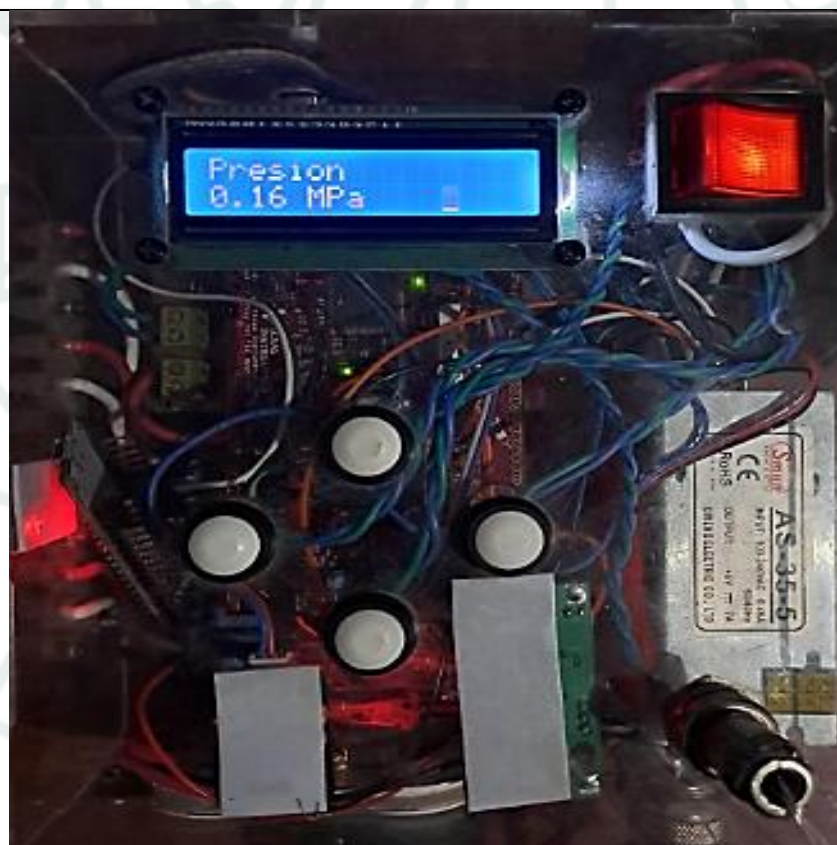
TEMPERATURA

19.03°C



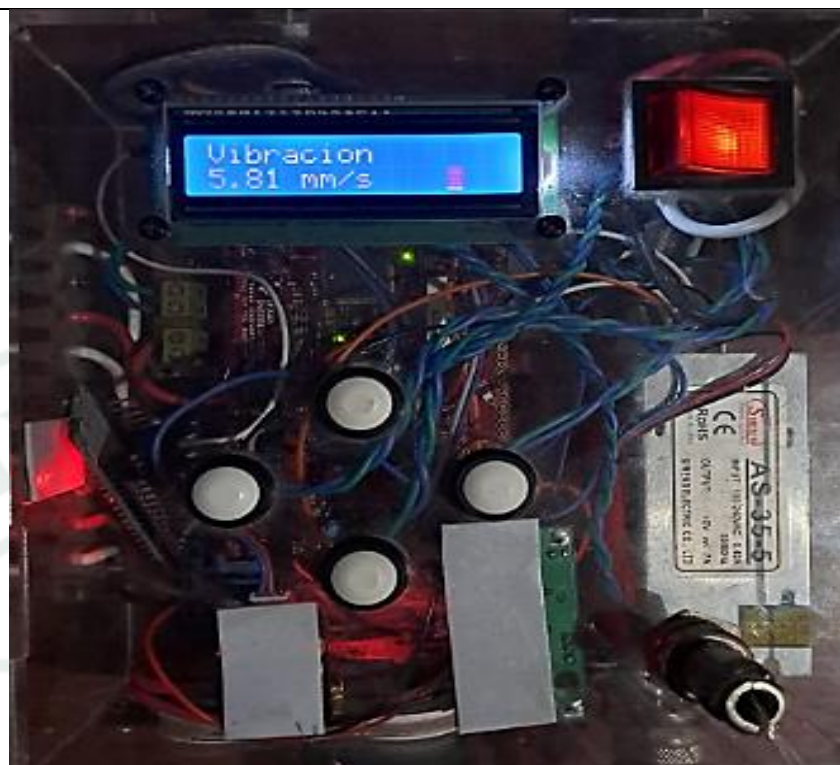
PRESIÓN

0.16 MPa



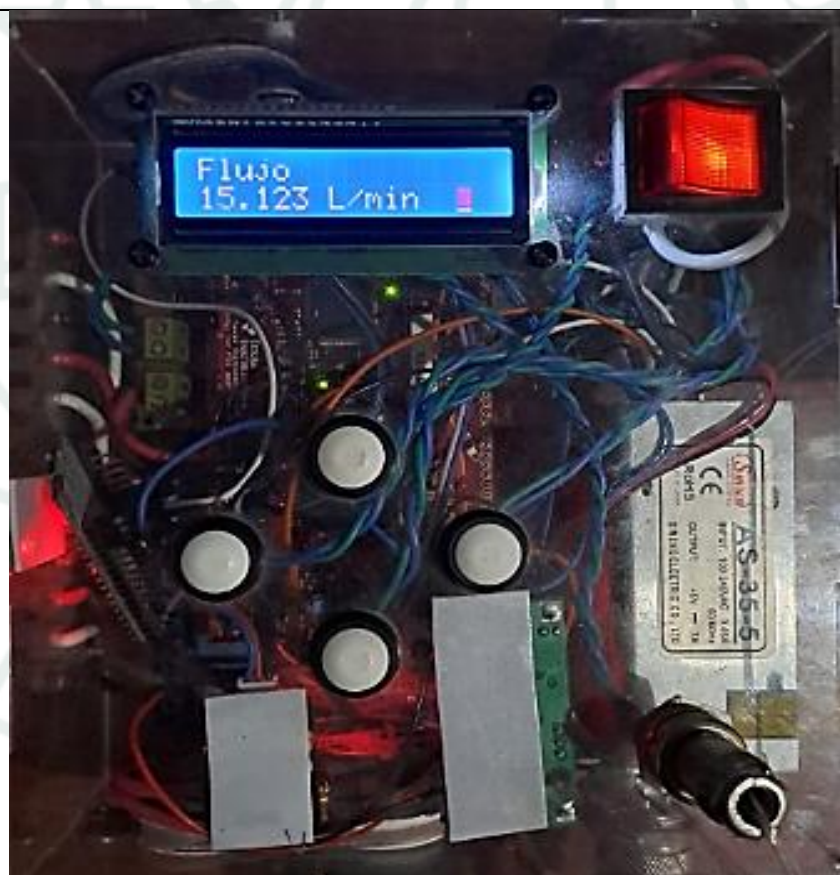
VIBRACIÓN

5.81 mm/s



FLUJO

16.123 L/min



6.6 Análisis económico

El análisis económico del presente tema de tesis presentado está basado en la inversión que se realizó en la fabricación del prototipo para monitoreo de la condición de trabajo de un sistema de bombas centrífugas, los componentes adquiridos y la inversión de tiempo en horas.

Tabla 16

Análisis Económico del Equipo

ITEM	DESCRIPCION	PRECIO UNIDAD	QTY	PRECIO TOTAL
1	Caja de fibra de vidrio	S/ 70.00	1	S/ 70.00
2	Tornillos de ES33	S/ 0.50	8	S/ 4.00
3	Elástico para seguridad del equipo al usuario	S/ 5.00	1	S/ 5.00
TOTAL				S/ 79.00

Nota: Tabla de análisis económico del equipo de monitoreo desarrollado

Tabla 17

Análisis Económico de los Componentes Aquiridos

ITEM	DESCRIPCIÓN	PRECIO UNIDAD	QTY	PRECIO TOTAL
1	Microcontrolador Hércules LM46x	S/ 370.00	1	S/ 370.00
2	Motor monofásico 0.5HP	S/ 350.00	1	S/ 350.00
3	Modulo Wifi	S/ 80.00	1	S/ 80.00
4	Sensor de vibración PC420VR-05	S/ 1,300.00	1	S/ 1,300.00
5	Sensor de temperatura RTD	S/ 15.00	1	S/ 15.00

6	Sensor de flujo	S/ 20.00	1	S/ 20.00
7	Sensor de presión	S/ 25.00	1	S/ 25.00
8	Sensor de corriente AC5756-050 ^a	S/ 25.00	1	S/ 25.00
9	Batería portátil de 5V a 7A	S/ 50.00	1	S/ 50.00
10	Pantalla LCD 2X16	S/ 20.00	1	S/ 20.00
11	Adaptador de 4 a 20 mA	S/ 20.00	1	S/ 20.00
12	Módulo de corriente SEN0262	S/ 25.00	1	S/ 25.00
13	Modulo Relay de cuatro canales	S/ 15.00	1	S/ 15.00
14	Modulo Wifi ESP32	S/ 20.00	1	S/ 20.00
15	Pines Peine	S/ 1.00	2	S/ 2.00
16	Pines Peine en L	S/ 1.50	2	S/ 3.00
17	Cables de Arduino Hembra - Macho	S/ 8.00	1	S/ 8.00
18	Molex	S/ 1.20	2	S/ 2.40
19	Tubo 1" PCV	S/ 20.00	1	S/ 20.00
20	Resistencias SMD	S/ 1.00	10	S/ 10.00
21	Capacitores electrolíticos	S/ 1.50	3	S/ 4.50
22	Fibra de vidrio	S/ 30.00	2	S/ 60.00
23	Botón de Prendido	S/ 2.50	4	S/ 10.00
24	Recipiente de 12 L	S/ 8.00	1	S/ 8.00
25	Acoples para tubería	S/ 5.00	6	S/ 30.00
26	Conectores 220V	S/ 8.00	2	S/ 16.00
27	Tubería de PVC reductores	S/ 3.00	1	S/ 3.00
28	Pines / Cable	S/ 10.00	1	S/ 10.00
29	Cable 220 V	S/ 3.00	1	S/ 3.00
TOTAL				S/ 2,524.90

Nota: Tabla de análisis económico de los componentes adquiridos

Tabla 18

Análisis de Costo Total del Prototipo de Monitoreo

ITEM	DESCRIPCIÓN	PRECIO TOTAL
1	Case del equipo	S/ 79.00
2	Componentes electrónicos adquiridos	S/ 2,524.90
3	Costo del diseño e implementación de las placas	S/ 50.00
TOTAL		S/ 2,653.90

Nota: Tabla de análisis de costo total del prototipo de monitoreo

6.7 Sistema de monitoreo para unidades mineras grandes

6.7.1 Synertrex – intellipump (empresa – global)

Sistema de monitoreo de bombas centrífugas para bombas de lodo, sistema altamente eficiente para bombas de gran tamaño que monitorea y evita las pérdidas de eficiencia. Etc.





Cliente / Area:		Proyecto: SISTEMA DE MONITOREO DE BOMBAS 650 MRC	
Item	Código	Descripción	PRECIO
1	JE0120002031201	SISTEMA DE MONITOREO, SYNERTREX INTELLIPUMP	\$56,678,00

Considerando nuestro prototipo a una escala industrial podemos determinar aproximadamente puede costar, para realizar una comparación respecto a un sistema de empresas internacional y que ofrecen este tipo de productos y servicio de monitoreo.

Tabla 19

Análisis de Costo de Prototipo Actual

ITEM	DESCRIPCION	PRECIO TOTAL
1	Costo de prototipo actual para realizar pruebas	S/ 2,653.90

Nota: Tabla de análisis de costo de prototipo actual

Convertir a dólares el precio de prototipo actual:

$$S/ 2653.90 \times 3.70 = \$ 720.00$$

Se multiplica precio actual por un factor para sacar precio de costo a una escala industrial para poder realizar una comparación aproximada.

$$\$ 720.00 \times 10 = \$ 7,200.00$$

Comparar:

Tabla 20

Análisis de Costo de Comparativo de Prototipo Actual VS Sistema de Monitoreo

ITEM	DESCRIPCION	PRECIO TOTAL
1	Costo de prototipo actual para realizar pruebas (PROTOTIPO)	\$ 7,200.00
2	Sistema de monitoreo, Synertrex intellipump	\$ 56,678,00

Nota: Tabla de costo de comparativo de prototipo actual vs sistema de monitoreo

6.7.2 Conclusión:

Se verificó que el prototipo desarrollado e implementado, al convertirlo a una escala industrial con un factor de 10, resulta aproximadamente siete veces más económico que los sistemas de monitoreo ofrecidos por empresas especializadas en soluciones digitales a nivel mundial.

Para unidades mineras de baja producción, el costo de estos sistemas comerciales es inviable y no sostenible para su nivel de operación. Por ello, la implementación de soluciones de bajo costo, como el prototipo desarrollado, representa una alternativa viable para el monitoreo de las condiciones de sus bombas en operación.



CAPÍTULO VII

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

1. Se diseñó e implementó correctamente el prototipo para el monitoreo de bombas centrífugas con un sistema piloto que nos permitió comprender el comportamiento de un sistema de mayor potencia, el cual es capaz de recolectar datos de su condición de trabajo en la nube de los parámetros principales de operación y visualizando esta condición que se transmite en la pantalla LCD del prototipo y mediante el desarrollo de dashboards para visualización en la nube, el cual se puede almacenar y guardar la información en SQL Server, para el desarrollo de un análisis por el vendedor más detallado del estado en que se encuentra operando la bomba.
2. La implementación del prototipo IoT en la nube permitió demostrar la viabilidad técnica de un sistema de monitoreo en tiempo real para bombas centrífugas, integrando sensores de temperatura, presión, caudal, voltaje, corriente y vibración con una frecuencia de muestreo inferior a 10 ms (60us). Esto cumple con los requerimientos de alta resolución para el análisis de señales en entornos industriales.
3. El uso de tecnologías de comunicación inalámbrica mediante protocolo de transmisión TCP, junto con el almacenamiento en la base de datos SQL Server y la visualización en dashboards, validó que es posible desarrollar soluciones de monitoreo robustas y escalables aplicando principios de ingeniería electrónica, combinando hardware de adquisición de datos con software de gestión y análisis.
4. Las pruebas en el sistema piloto, que simula condiciones de operación de bombas centrífugas de mayor capacidad, confirmaron que el diseño electrónico propuesto ofrece la precisión necesaria para la detección temprana de fallos mecánicos y eléctricos como cavitación, sobrecarga eléctrica, favoreciendo la implementación de estrategias de mantenimiento predictivo.

5. La correcta selección y calibración de sensores fue determinante para garantizar la fiabilidad de las mediciones. En ingeniería electrónica aplicada a entornos mineros, esto implica priorizar exactitud, inmunidad al ruido, estabilidad a largo plazo y resistencia a condiciones ambientales adversas.
6. El sistema desarrollado ofrece una alternativa de bajo costo frente a soluciones comerciales, sin sacrificar prestaciones esenciales. Esto demuestra que la ingeniería electrónica puede proveer soluciones innovadoras y económicamente viables para el sector minero.
7. La integración de adquisición de datos de alta velocidad, procesamiento eficiente y transmisión confiable sienta las bases para escalar el sistema hacia otras máquinas críticas en plantas industriales, reforzando el papel de la ingeniería electrónica en la transformación digital del sector.
8. El prototipo demostró que no es necesario detener la bomba para diagnosticar su estado de operación. Las mediciones en funcionamiento permiten identificar con mayor precisión las condiciones reales, facilitando la programación anticipada de mantenimientos y el cambio de partes críticas por los operadores.
9. El sistema IoT desarrollado permite realizar análisis históricos y detectar tendencias operativas, resolviendo la falta de monitoreo continuo en bombas centrífugas. Esto habilita a operadores y proveedores a tomar decisiones basadas en datos reales, reduciendo costos y mejorando la productividad.
10. La medición de vibraciones con cálculo RMS, considerando el rango dinámico del sensor (0 a 250 g pico) y un ancho de banda de 3,5 Hz a 2,0 kHz, asegura una evaluación precisa incluso en presencia de ruido y picos transitorios.
11. La utilización de una señal de salida industrial estándar de 4-20 mA garantiza la compatibilidad del sistema con PLCs, controladores y sistemas SCADA existentes, facilitando su integración en entornos industriales consolidados.

RECOMENDACIONES

1. Incorporar nuevos tipos de sensores que amplíen la gama de variables monitoreadas, obteniendo una evaluación más completa del estado operativo de las bombas y sus sistemas auxiliares.
2. Analizar el impacto de las variables medidas sobre la eficiencia operativa del sistema mediante el uso de sistemas expertos o herramientas de análisis avanzado para correlacionar parámetros y detectar oportunidades de optimización.
3. Implementar algoritmos que permitan evaluar en tiempo real la condición de operación y la curva característica de la bomba centrífuga, detectando desviaciones respecto a su punto de operación óptimo.
4. Incorporar análisis predictivo en el prototipo mediante machine learning o análisis estadístico de datos históricos, anticipando fallos y mejorando la precisión en la programación de mantenimientos.
5. Ejecutar pruebas de validación en condiciones reales de operación minera para verificar la precisión de las mediciones y la efectividad del sistema IoT ante cambios de carga, temperatura y condiciones ambientales extremas.
6. Adaptar el sistema para que sea interoperable con plataformas de gestión de mantenimiento existentes, optimizando el flujo de información y la toma de decisiones.
7. En operaciones con baja conectividad, considerar tecnologías robustas como LoRaWAN, 5G o redes satelitales, junto con almacenamiento local de datos para sincronización posterior con la nube.

REFERENCIAS

- Ahmad, Z., Rai, A., Maliuk, A. S., & Kim, J.-M. (2020). Discriminant feature extraction for centrifugal pump fault diagnosis. *IEEE Access*, 8, 165512–165528.
<https://doi.org/10.1109/ACCESS.2020.3022820>
- Bentley, J. P. (2005). *Principles of measurement systems* (4th ed.). Pearson Education.
- Doebelin, E. O., & Manik, D. N. (2011). *Measurement systems: Application and design* (6th ed.). McGraw-Hill.
- Karassik, I. J., Messina, J. P., Cooper, P., & Heald, C. C. (2008). *Pump handbook* (4th ed.). McGraw-Hill Education.
- Miller, R. W. (1996). *Flow measurement engineering handbook* (3rd ed.). McGraw-Hill.
- Mobley, R. K. (2002). *An introduction to predictive maintenance* (2nd ed.). Butterworth-Heinemann.
- Omega Engineering. (2019). *The temperature handbook* (Vol. 28). Omega Press.
- Pinto Guillén, V. J. (2021). *Diseño e implementación de medidores de energía eléctrica basados en IoT para la gestión del consumo energético* [Tesis de pregrado, Universidad Católica de Santa María].
- Rodríguez Durán, J. H. (2023). *Monitoreo y medición de rendimiento basado en Internet de las Cosas para sistemas de bombas centrífugas* [Tesis de pregrado, Pontificia Universidad Javeriana].
- Sánchez Gonzales, J. R. (2021). *Aplicación de monitoreo de condiciones para el diagnóstico de fallas de motor y transmisión en cargadores 966H mediante uso de machine learning* [Tesis de pregrado, Universidad Católica de Santa María].
- Sulzer Pumps. (2010). *Centrifugal pump handbook*. Elsevier
- Warman, G. (2015). *Slurry pumping handbook* (6th ed.). Weir Minerals.

ANEXO A: DATASHEET SENSOR DE VIBRACION WILCOXON

RMS and peak velocity loop
powered sensors

PC420V series



Wilcoxon's PC420V series sensors provide a 4-20 mA output proportional to velocity vibration, allowing for continuous trending of overall machine vibration. This trend data alerts users to changing machine conditions and helps guide maintenance in prioritizing the need for service. The choice of RMS or peak output allows you to choose the sensor that best fits your requirements.



Table 1: PC420Vx-yy model selection guide	
x (4-20 mA output type)	yy (4-20 mA full scale)
R = RMS output, velocity P = calculated peak output, velocity	05 = 0.5 ips (12.8 mm/sec)
	10 = 1.0 ips (25.4 mm/sec)
	20 = 2.0 ips (50.8 mm/sec)
	30 = 3.0 ips (76.2 mm/sec)
	50 = 5.0 ips (127 mm/sec)

Key features

- True RMS or calculated peak output
- Intrinsically safe certified and explosion-proof models available
- Easily integrated into existing process control systems
- Manufactured in an approved ISO 9001 facility

Certifications



Note: Due to continuous process improvement, specifications are subject to change without notice. This document is cleared for public release.

Wilcoxon Sensing Technologies
An Amphenol Company

8435 Progress Drive
Frederick, MD 21701
USA

Tel: +1 (301) 330-8811
Fax: +1 (301) 330-8873
info@wilcoxon.com

buy.wilcoxon.com
www.wilcoxon.com

RMS and peak velocity loop powered sensors

PC420V series

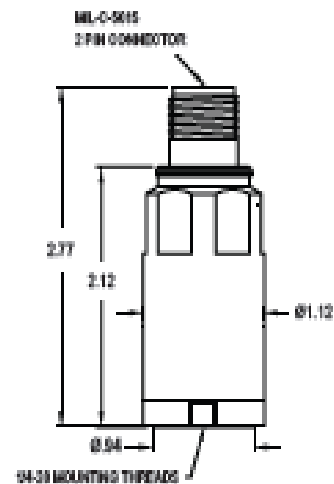


SPECIFICATIONS

Full scale, 20 mA, ±6%	see Table 1 on page 1	
Frequency response:	±10%	10 Hz - 1.0 kHz
	±3 dB	3.5 Hz - 2.0 kHz
Repeatability	±2%	
Transverse sensitivity, max	5%	
Power requirements, 2-wire loop power:		
Voltage at sensor terminals	12 - 30 VDC	
Loop resistance ¹ at 24 VDC, max	700 Ω	
Turn on time, 4-20 mA loop	30 seconds	
Grounding	case isolated, internally shielded	
Operating temperature range	-40° to +105° C	
Vibration limit	250 g peak	
Shock limit	2,500 g peak	
Sealing	hermetic	
Sensing element design	PZT, shear	
Weight	160 grams	
Case material	stainless steel	
Mounting	1/4-28 tapped hole	
Output connector	2 pin, MIL-C-5015 style	
Mating connector	R6 type	
Recommended cabling	J9T2A	

Accessories supplied: SP8 mounting stud; calibration data (level 2)

Connections	
Function	Connector pin
loop positive (+)	A
loop negative (-)	B
ground	shell



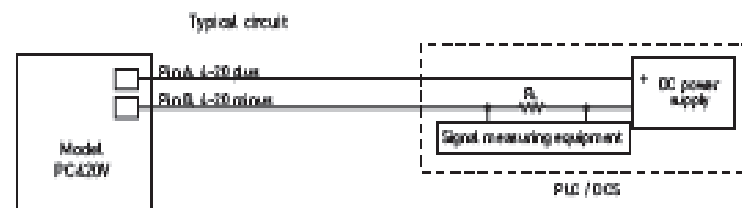
Notes: ¹ Maximum loop resistance (R_L) can be calculated by:

$$R_L = \frac{V_{DC\ peak} - 10\ V}{20\ mA}$$

DC supply voltage	R_L (max resistance) ²	R_L (minimum wattage capability) ³
12 VDC	100 Ω	1/8 watt
20 VDC	500 Ω	1/4 watt
24 VDC	700 Ω	1/2 watt
28 VDC	800 Ω	1/2 watt
30 VDC	1,000 Ω	1/2 watt

² Lower resistance is allowed, greater than 10 Ω recommended.

³ Minimum R_L wattage determined by: $(0.0004 \times R_L)$.



Note: Due to continuous process improvement, specifications are subject to change without notice. This document is cleared for public release.

Wilcoxon Sensing Technologies
An Amphenol Company

8435 Progress Drive
Frederick, MD 21701
USA

Tel: +1 (301) 330-8811
Fax: +1 (301) 330-8873
info@wilcoxon.com

www.wilcoxon.com
www.wilcoxon.com

29009 RevB.3 03/19

2

ANEXO B: DATASHEET MOTOR

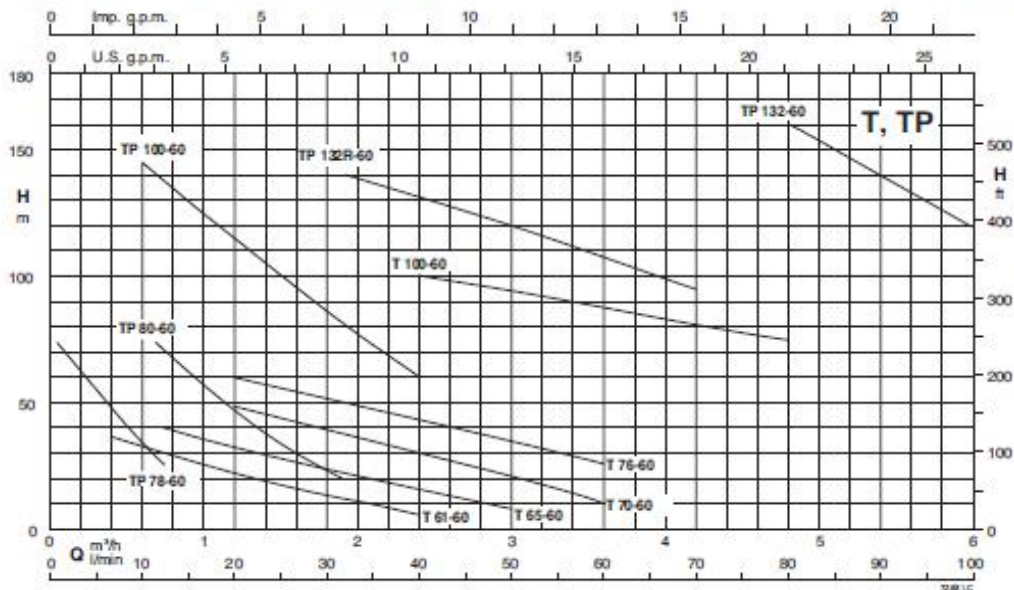
T, TP 60 Hz



Peripheral Pumps with turbine impeller
Bomba con rodete periférico



Coverage chart - Campo de aplicaciones n ≈ 3450 rpm



T, TP 60 Hz



Construction

Close-coupled peripheral pumps (regenerative pumps) with turbine impeller.
Connections: threaded ports **ISO 228/1**.
 threaded ports **NPT** (ANSI/ASME B1.20.1) on request

T, TP: version with pump casing and lantern bracket in cast iron.
 B-T, B-TP: version with pump casing and lantern bracket in bronze
 (the pumps are supplied fully painted).

Applications

For clean liquids without abrasives, without suspended solids, non-explosive, non-aggressive for the pump materials.
 For increasing network pressure (follow local specifications).
 For the reduced dimensions, these pumps are very well suitable to be mounted in cooling and air-conditioning machines and equipments, circulation, boiler feed.

Operating conditions

Liquid temperature from -10 °C to +90 °C.
 Ambient temperature 0 °C to +40 °C.
 Total suction lift up to 7 m.
 Continuous duty.

Motor

2-pole induction motor, 60 Hz (n = 3450 rpm).
T, TP: three-phase 220/380 V, 220/440 V, up to 3 kW;
 380/660 V, from 4 to 7,5 kW;
TM, TPM: single-phase 110 V, 127 V, 220 V.
 with thermal protector up to 1.1 kW only 220V.
 Capacitor inside the terminal box.

Insulation class F.
 Protection IP 54.

Classification scheme IE3 for three-phase motors from 0,75 kW.
 Constructed in accordance with: EN 60034-1; EN 60034-30-1.
 EN 60335-1, EN 60335-2-41.

Special features on request

- Other voltages.
- Protection IP 55.
- Special mechanical seal
- Higher or lower liquid or ambient temperatures.

Materiales

Components	T, TP	B-T, B-TP
Pump casing	Cast Iron	Bronze
Lantern bracket	G.J.L 200 EN 1561	G-Cu Sn 10 EN 1982
Casin cover	Cast Iron	Bronze
	G.J.L 200 EN 1561	G-Cu Sn 10 EN 1982
	Brass P- Cu Zn 40 Pb 2 UNI 5705 for T 61-65-70, B-T 61-70	
Impeller	Brass P- Cu Zn 40 Pb 2 UNI 5705 Bronze G-Cu Sn 10 EN 1982 for TP 132-132R	
Shaft	Cr-Ni steel AISI 303 T 70-76, TP 78-80-100	Cr-Ni-Mo steel AISI 316
	Chrome steel AISI 430 T 61-65-100-125, TP 132-132R	
Mechanical seal	Carbon - Ceramic - NBR	

Ejecución

Electrobomba monobloc con rodete periférico.
Orificios: Roscados **ISO 228/1**.
 Roscados **NPT** (ANSI/ASME B1.20.1) bajo demanda.

T, TP: Ejecución con cuerpo bomba y acoplamiento in hierro.
 B-T, B-TP: Ejecución con cuerpo bomba y acoplamiento in bronce.
 Las bombas en bronce se suministran totalmente pintadas.

Aplicaciones

Para líquidos limpios, sin partículas sólidas en suspensión, y no agresivos para los materiales de la bomba.
 Para aumentar la presión disponible en una red de distribución de agua (observar las disposiciones locales).
 Por sus reducidas dimensiones son muy adecuadas para el montaje en máquinas y sistemas de refrigeración, circuitos de alimentación, alimentación de caderas, etc.

Límites de empleo

Temperatura líquido de -10 °C a +90 °C.
 Temperatura ambiente de 0 °C a +40° C.
 Altura de aspiración manométrica hasta 7 m.
 Servicio continuo.

Motor

Motor a inducción 2 polos, 60 Hz (n = 3450 1/min).
T, TP: trifásico 220/380 V, 220/440 V, hasta 3 kW;
 380/660 V, de 4 a 7,5 kW;
TM, TPM: monofásico 110 V, 127 V, 220 V.
 con protector térmico hasta 1,1 kW sólo para 220V.
 Condensador incorporado en la caja de bornes.

Aislamiento clase F.
 Protección IP 54.

Clase alta eficiencia IE3 para motor trifásico de 0,75 kW.
 Ejecución según EN 60034-1; EN 60034-30-1.
 EN 60335-1, EN 60335-2-41.

Otras ejecuciones bajo demanda

- Otras tensiones.
- Protección IP 55.
- Sello mecánico especial.
- Para líquidos o ambientes con temperaturas más elevadas o más bajas.

Materiales

Componentes	T, TP	B-T, B-TP
Cuerpo bomba	Hierro	Bronce
Acoplamiento	G.J.L 200 EN 1561	G-Cu Sn 10 EN 1982
Tapa cuerpo bomba	Hierro	Bronce
	G.J.L 200 EN 1561	G-Cu Sn 10 EN 1982
	Latón P-Cu Zn 40 Pb 2 UNI 5705 para T 61-65-70, B-T 61-70	
Rodete	Latón P-Cu Zn 40 Pb 2 UNI 5705 Bronce G-Cu Sn 10 EN 1982 para TP 132-132R	
Eje	Acero al Cr-Ni AISI 303 T 70-76, TP 78-80-100	Acero al Cr Ni Mo AISI 316
	Acero al cromo AISI 430 T 61-65-100-125, TP 132-132R	
Sello mecánico	Carbón - Cerámica - NBR	

T, TP 60 Hz



Performance - Prestaciones

n ≈ 3450 rpm

3 ~	1 ~	P ₂		Q m ³ /h	H m																	
					0,06	0,12	0,24	0,38	0,48	0,6	0,75	1	1,2	1,5	1,89	2,4	3	3,6	4,2	4,8	5,4	6
		kW	HP	l/min	1	2	4	6,3	8	10	12,5	16	20	25	31,5	40	50	60	70	80	90	100
B- T 61E-60	B- TM 61E-60	0,33	0,45					36	34	32	30	26	21	16,5	11	5						
B- T 65E-60	B- TM 65E-60	0,45	0,6							40	37	33	29	24	16	8						
B- T 70-60/B	B- TM 70-60/A	0,75	1									48	43	38	30	22	10					
T 76-60/A	TM 76E-60	1,1	1,5									60	55	50	45	35	25					
T 100-60/B		4	5,5													100	95	85	80	75		
B- TP 78-60/A	B- TPM 78-60/A	0,37	0,5		74	69	60	50	42	34	25											
B- TP 80-60E	B- TPM 80E-60	0,75	1							91,5	83	70	59	44	25,5							
TP 100-60/B		2,2	3							145	140	131	124	114	100	81,5						
TP 132R-60/A		7,5	10												140	130	120	105	95			
TP 132-60/A		7,5	10																	160	140	120

B-T, B-TM = Bronze construction.
Ejecución en bronce.

P₂ Rated motor power output.
Potencia nominal del motor.

H Total head in m.
Altura total en m.

Tolerances according to UNI EN ISO 9906:2012.
Tolerancias según UNI EN ISO 9906:2012.

* Maximum suction lift 2-3 m.
Máxima aspiración manométrica 2-3 m.

Rated currents - Intensidades nominales

TYPE TIPO	P2		single-phase - monofásico 1~			Ia/In
	kW	HP	220V In A	127V In A	110V In A	
	0,33	0,45	3,1	5,4	6,2	2,5
	0,37	0,55	3,4	5,9	6,8	2,3
	0,45	0,6	4,2	7,3	8,4	2,6
TM 70-60/A	0,75	1	6,9	12	13,8	2,9
TPM 80E-60	0,75	1	6,5	11,3	13	2,9
	1,1	1,5	8,9	15,4	-	2,9

TYPE TIPO	P2		three-phase - trifásico 3~				Ia/In
	kW	HP	220/380V In A	380/660V In A	220/440V In A		
	0,33	0,45	2,4/1,4	-	2,5/1,5	3,2	
	0,37	0,55	2,6/1,5	-	2,7/1,6	3,5	
	0,45	0,6	3,1/1,8	-	3,3/1,9	4,8	
T 70-60/A	0,75	1	4,5/2,6	-	4,7/2,7	5,6	
TP 80E-60	0,75	1	4/2,3	-	4,2/2,4	7,3	
	1,1	1,5	6,4/3,7	-	6,7/3,9	4,8	
	2,2	3	11,1/6,4	-	12/6,9	8,4	
	4	5,5		11,2/6,5		9,2	
	5,5	7,5		13,7/7,9		8,7	
	7,5	10		17/9,8		9,2	

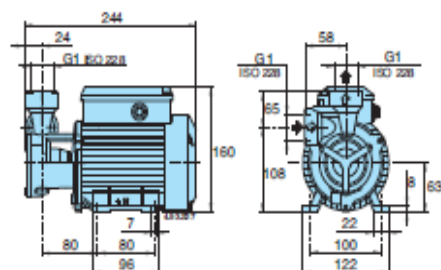
P₂ Rated motor power output.
Potencia nominal del motor.

I_A/I_N D.O.L. starting current / Rated current
Intensidad de arranque / Intensidad nominal

T, TP 60 Hz

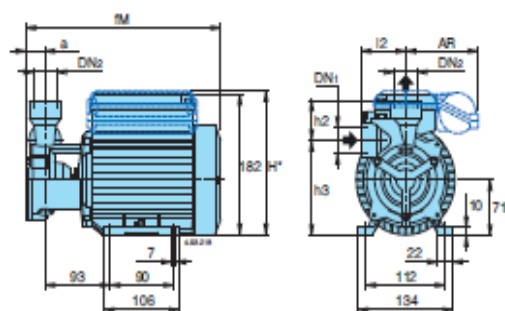


Dimensions and weights - Dimensiones y pesos



T 61E-60 : kg 6,3
B- T 61E-60 : kg 6,5
T 65E-60 : kg 7,3
B- T 65E-60 : kg 7,5

DN NPT Version on demand - NPT Ejecuciones bajo demanda

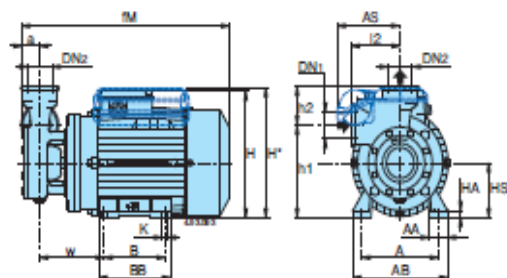


TYPE TIPO	DN1	DN2	DN	mm					kg	
				a	IM	h2	h1	I2	T	B-T
T 70-60/B B-T 70-60/B	G 1	G 1	✓	24	278	50	121	63	12	12,4
TP 70-60/A B-TP 70-60/A	G 1/2	G 1/2	✓	22	275	24	127	56	8,2	8,8

✓ NPT Version on demand - Ejecuciones bajo demanda

TYPE TIPO	H*	AR mm		
		220V	127V	110V
TM 70-60/A, B-TM 70-60/A	187	●	116	116

● Standard dimensions - Dimensiones estándar
□ Cannot constructed - Non fatibles



TYPE TIPO	Hc	AS mm		
		220V	127V	110V
TPM 80E-60, B-TPM 80E-60	133	●	116	131
TM 76E-60	133	●	131	131

● Standard dimensions - Dimensiones estándar
□ Cannot constructed - Non fatibles

TYPE TIPO	DN1	DN2	DN	mm																kg	
				a	IM	HS	h2	h1	H	BB	B	AB	A	AA	K	I2	w	HA	T, TP	B-TP	
T 75-60/A	—	G 1 1/4	G 1 1/4	✓	25	338	80	56	136	208	117	100	155	125	30	9	80	105	10	18,4	-
T 100-60/B	—	G 1 1/4	G 1 1/4	✓	32	410	90	59	161	226	152	125	180	140	40	9,5	95	121	12	32,5	-
TP 80E-60 B-TP 80E-60	—	G 3/4	G 3/4	✓	27	332	80	35	135	208	117	100	155	125	30	9	60	104	10	16,4	16,8
TP 100-60/B	—	G 3/4	G 3/4	✓	27	387	80	38	142	208	117	100	155	125	30	9	65	113	10	23,2	-
TP 132 R-60/A	—	G 1 1/4	G 1 1/4	✓	42	485	112	70	202	272	180	140	230	190	50	11,5	100	183	14	58,5	-
TP 132-60/A	—	G 1 1/4	G 1 1/4	✓	42	485	112	70	202	272	180	140	230	190	50	11,5	100	183	14	58,5	-

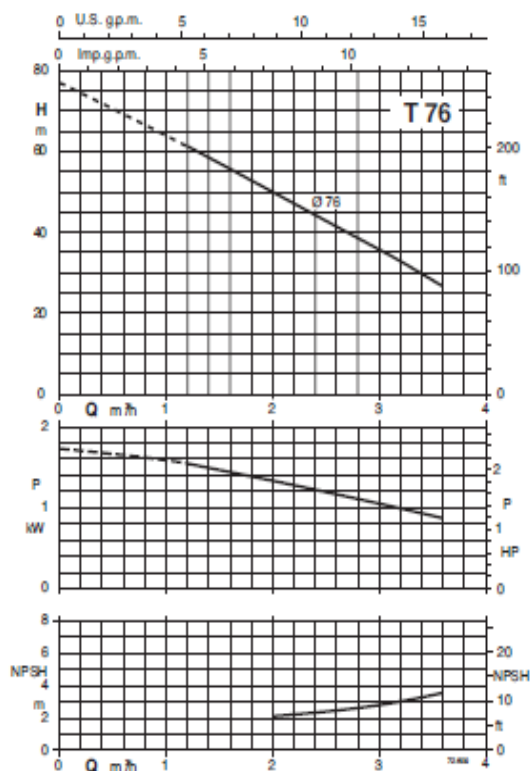
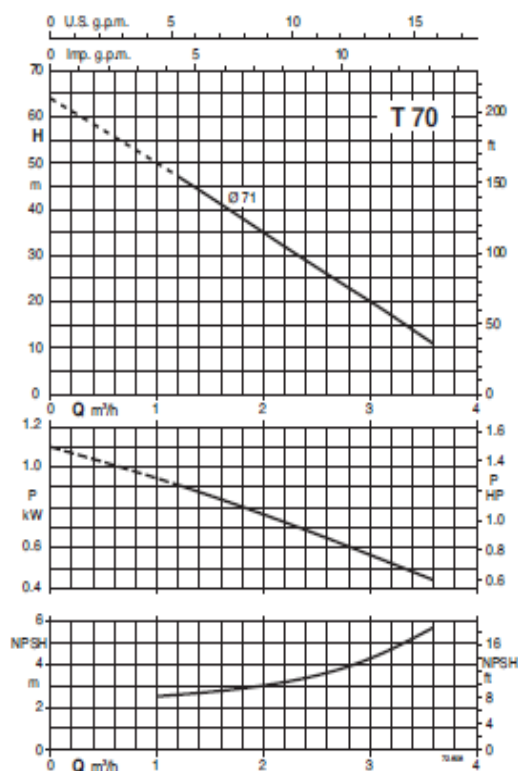
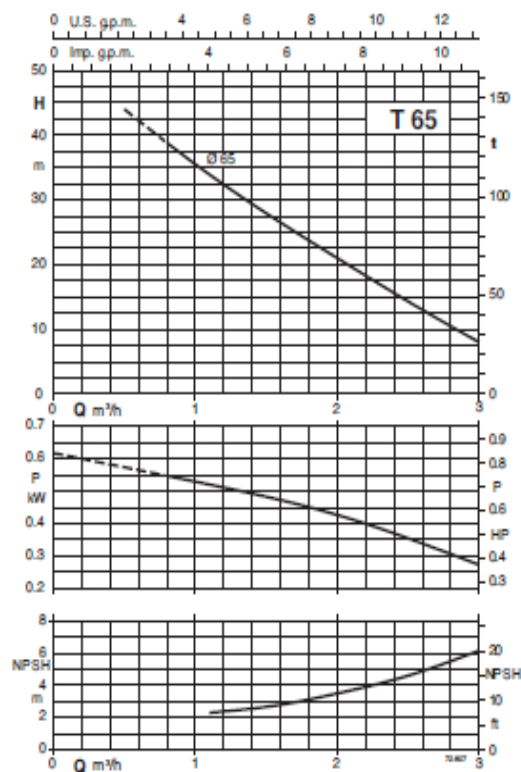
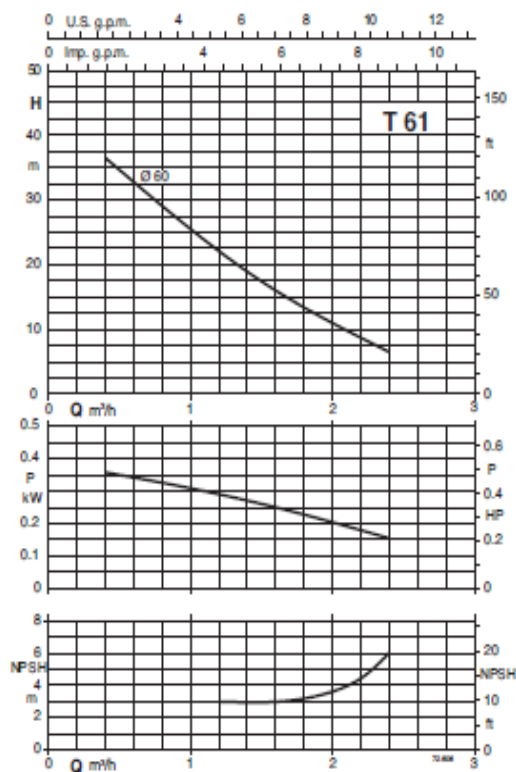
✓ NPT Version on demand - Ejecuciones bajo demanda

T, TP 60 Hz



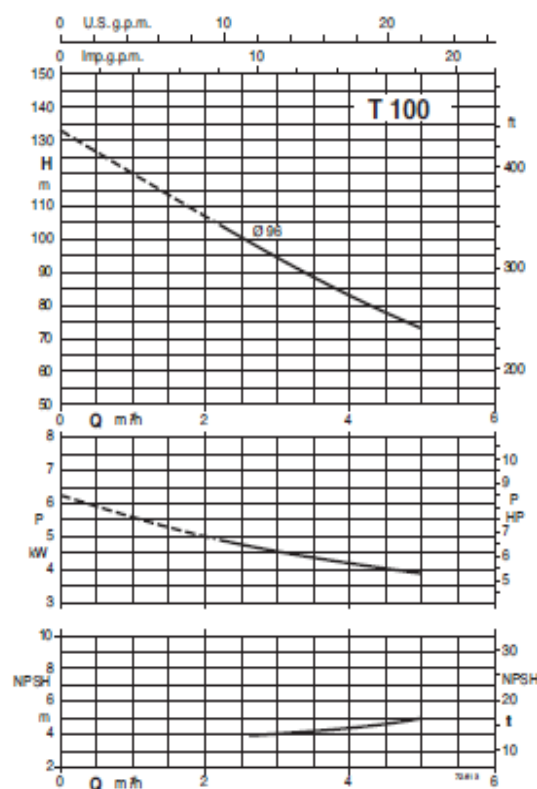
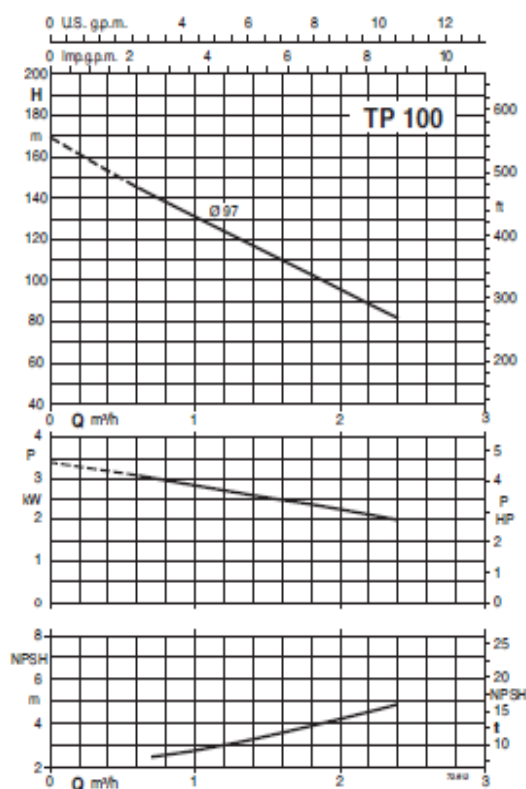
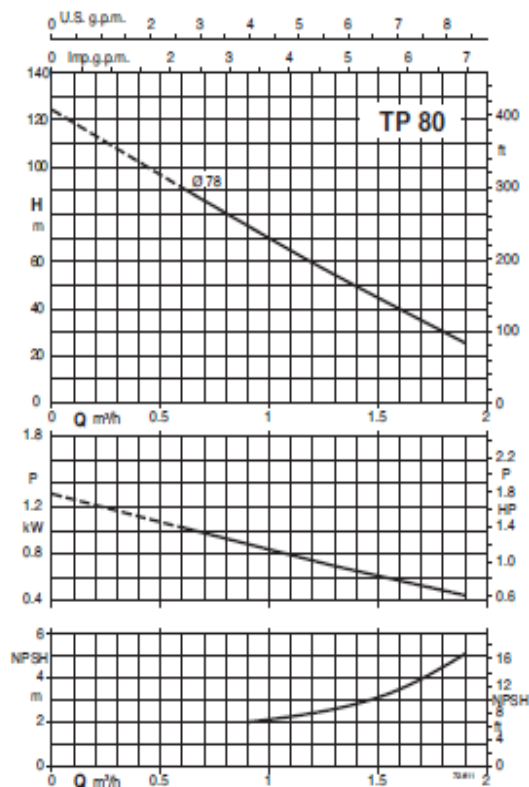
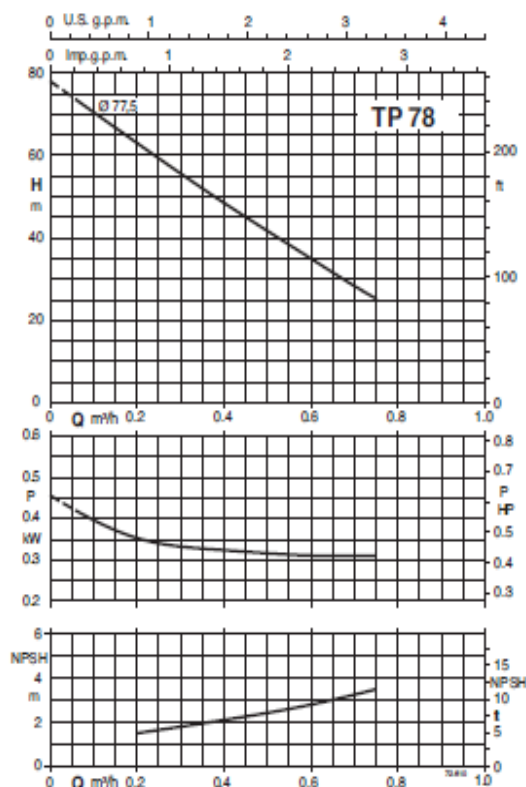
Characteristic curves - Curvas Características

n ≈ 3450 rpm

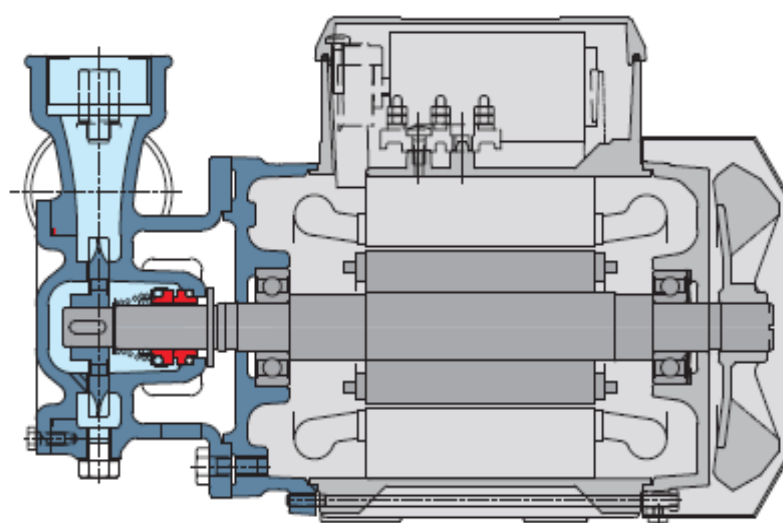


Characteristic curves - Curvas Características

$n \approx 3450$ rpm



Features - Características constructivas



Range

The high number of pumps in the range can meet the widest range of services required by the user.

Flexibility

The option to choose between cast iron and bronze materials for the hydraulic parts in contact with the pumped liquid allows T-TP series pumps to be selected for use with different types of liquids.

Reliability

The bearing and shaft are designed to ensure the reduction of the stress, providing high reliability under all operating conditions.

Optimized hydraulics

The pump hydraulics are designed to ensure high performance and consistency of performance.

Alcance

El elevado número de bombas en el rango puede satisfacer la más amplia gama de servicios requeridos por el usuario.

Flexibilidad

La opción de poder elegir entre hierro fundido y bronce para las partes hidráulicas en contacto con el bombeo permite que las bombas de la serie T-TP puedan ser seleccionadas para utilizar con diferentes tipos de líquidos.

Fiabilidad

El cojinete y el eje están diseñados para asegurar la reducción de la tensión, proporcionando alta fiabilidad en todas las condiciones.

Hidráulica optimizada

El sistema hidráulico de la bomba están diseñados para asegurar un alto rendimiento y la consistencia del rendimiento.

ANEXO C: PROGRAMA - CÓDIGO PROGRAMA PRINCIPAL

```
/* Include Files */

#include "sys_common.h"

/* USER CODE BEGIN (1) */
#include "sys_core.h"
#include "system.h"
#include "sci.h"
#include "uartstdio.h"
#include "het.h"
#include "gio.h"
#include "spi.h"
#include "rti.h"
#include "stdio.h"
#include "adc.h"
#include "etpwm.h"
// #include "DS1302.h" //agregando las cabeceras de las funciones
// #include "EE25LC256.h"

#include "DS1307_24C.h"
#include "i2c.h"

#include "LCD_3LINES.h"
#include "mibspi.h"
#include "reg_mibspi.h"
#include "KERNEL_MCE.h" /*Kernel de Motor de calculo Electrico*/

#include "LCD_Menus.h"
#include <string.h>
/* USER CODE END */

/** @fn void main(void)
 * @brief Application main function
 * @note This function is empty by default.
 *
 * This function is called after startup.
 * The user can use this function to implement the application.
 */

/* USER CODE BEGIN (2) */

typedef struct
{
    char data[16];
} String;

//RTC_DS1302 Calendario;
bool LCD_Update = 0;
bool LCD_Unlock = 0;
int Temp_LCD = 0;
char Msg_LCD[16] = "";
```

```

char Msg_LCD1[16] = "";

int x = 0, y = 0;
char A[] = "Bienvenidos "; // Mensajeria que se envia por protocolo
serial al ESP32
char B[17] = { "" };
char msg_lcd[16];
hetSIGNAL_t Frec_signal;

void Display_Clock();

int valueSelect = 1;
uint8 btnPush;
bool EnableMenuLCD = 0;

int Estado = 1;
int Sig_Estado = 1;
int menuANTIFALLOSLENTO(String *arrayMenu, int size);

char g_pcCmdBuf[128] = ""; //línea de datos a recibir
char g_pcMqttBuf[128] = ""; //línea de datos a recibir
/* -----
---
** Matriz de formas de Onda
** -----
- */

typedef struct
{
    char Wifi_LocalIP[20]; // Vector mejora el resultado del
    calculo
    char Wifi_RSSI[4];
    char SSID[50];
    char mqtt_server[50];
    char mqtt_port[50];
    char root_topic_publish[50];
    char mqtt_user[50];
    char mqtt_pass[50];

    int RSSI;

    char MSGLCD0[50];
    char MSGLCD1[50];

} Wifi_Data;
char Msg_Serial[50] = ""; // Mensajeria que se envia por protocolo
serial al ESP32

Wifi_Data Wifi_Client;

void GetServerConfig();
void EnableServerCom();
void DisableServerCom();
void IpconfigAllServer();

```

```

extern bool RX_MSG_MQTT;

float PresionPVC = 0;
float FlujoMetro = 0;
float TempPT_100 = 0;
float PT_100 = 0;

float Vibration_Wilcoxon_mA = 0;
float Vibration_mms = 0;

uint32 PulsoFlujos;
/* USER CODE END */

int main(void)
{
    /* USER CODE BEGIN (3) */
    gpioInit();           //General input out
    hetInit();            // high end timer
    i2cInit();
    spiInit();
    adcInit();             //Inicializo las configuraciones del ADC
    etpwmInit();           //Inicializo las configuraciones del PWM-ADC
    int samples
    rtiInit();
    sciInit();
    lcd_init();            // inicializamos el LCD

    //sciInit(); /* initialize sci/sci-lin */
    /* even parity , 2 stop bits */
    /* Enable RTI Compare 0 interrupt notification */
    rtiEnableNotification(rtiNOTIFICATION_COMPARE0);
    rtiEnableNotification(rtiNOTIFICATION_COMPARE1);

    adcEnableNotification(adcREG1, adcGROUP0); //Habilito las
    interrupciones por ADC y PWM Trigger
    adcStartConversion(adcREG1, adcGROUP0); //Inicializo la conversion
    del ADC

    _enable_IRQ();
    _enable_interrupt_(); //Habilito las interrupciones

    /* Start RTI Counter Block 0 */
    rtiStartCounter(rtiCOUNTER_BLOCK0);

    digitalWrite_Relay(2, 1);
    LCD_BACKLIGHT_ON_OFF(1); // PRENDER
    //  gpioSetBit(hetPORT1, 12, 1);           // ENABLE LCD
    //  gpioSetBit(hetPORT1, 14, 1);           // ENABLE LCD
    //  gpioSetBit(hetPORT1, 16, 1);           // ENABLE LCD
    //  gpioSetBit(hetPORT1, 21, 1);           // ENABLE LCD

    /* -----
    -----
    **Enciende pantalla LCD
  
```

```

    ** -----
    ----- */
    if (!RTC_DS1307_isrunning()) //Primero Consulta si el reloj esta
    habilitado
    {
        lcd_gotoxy(1, 1); // envias coordenadas
        lcd_text("RTC is NOT running!");
        while (1)
            ;
    }
    /* -----
    -----
    ** Inicializa el motor de calculo
    ** -----
    ----- */
    Kernel_Init();

    //RTC_DS1307_datetime(24,01,2023,0,6,49);

    lcd_clear(); //limpiar pantalla LCD

    int menu;
    String MenuText[5];
    /*
    IpconfigAllServer(); //Lectura de parametros WIFI

    lcd_gotoxy(1, 1); // envias coordenadas
    lcd_text("Iniciando Wifi"); //envias texto
    lcd_gotoxy(1, 2); // envias coordenadas
    lcd_text("Espere ..."); //envias texto
    GetServerConfig();
    lcd_clear(); //limpiar pantalla LCD

    sprintf(Msg_LCD, "%s", Wifi_Client.SSID);
    sprintf(Msg_LCD1, "RSSI:%s dBm", Wifi_Client.Wifi_RSSI);

    lcd_gotoxy(1, 1); // envias coordenadas
    lcd_text(Msg_LCD); //envias texto
    lcd_gotoxy(1, 2); // envias coordenadas
    lcd_text(Msg_LCD1); //envias texto

    delay_ms(2000); //RETADOR DE 1 SEGUNDO
    lcd_clear(); //limpiar pantalla LCD

    sprintf(Msg_LCD, "IP:%s", Wifi_Client.Wifi_LocalIP);
    sprintf(Msg_LCD1, "Topic:%s", Wifi_Client.root_topic_publish);

    lcd_gotoxy(1, 1); // envias coordenadas
    lcd_text(Msg_LCD); //envias texto
    lcd_gotoxy(1, 2); // envias coordenadas
    lcd_text(Msg_LCD1); //envias texto

    delay_ms(2000); //RETADOR DE 1 SEGUNDO

```

```

lcd_clear();          //limpiar pantalla LCD

sprintf(Msg_LCD, "%s", Wifi_Client.mqtt_server);
lcd_gotoxy(1, 1);      // envias coordenadas
lcd_text("MQTT Server:"); //envias texto
lcd_gotoxy(1, 2);      // envias coordenadas
lcd_text(Msg_LCD);     //envias texto

delay_ms(2000); //RETADOR DE 1 SEGUNDO
lcd_clear();          //limpiar pantalla LCD

/*

sprintf(Msg_LCD, "IP:%s", Wifi_LocalIP);

lcd_gotoxy(1, 1);      // envias coordenadas
lcd_text(Msg_LCD);     //envias texto
sprintf(Msg_LCD1, "RSSI:%s dBm", Wifi_RSSI);
lcd_gotoxy(1, 2);      // envias coordenadas
lcd_text(Msg_LCD1);    //envias texto
*/

while (1)
{
    Estado = Sig_Estado;

    if (Estado == 1)
    {
        int menu;

        sprintf(MenuText[0].data, "Mediciones E/I");
        sprintf(MenuText[1].data, "Par. Proceso");
        sprintf(MenuText[2].data, "I/O");
        sprintf(MenuText[3].data, "Config. Equipo");
        sprintf(MenuText[4].data, "Fecha y Hora");
        menu = menuANTIFALLOLENTO(MenuText, 5); // no hace falta
poner el numero 6, se calcula automaticamente. Solo debes de cambiar
arrayMenu

        // menu = menuANTIFALLOLENTO(arrayMenu, size); // no
hace falta poner el numero 6, se calcula automaticamente. Solo debes de
cambiar arrayMenu

        if (menu == -1)
            Sig_Estado = 1;
        else if (menu == 1)
            Sig_Estado = 2;
        else if (menu == 2)
            Sig_Estado = 3;
        else if (menu == 3)
            Sig_Estado = 4;
        else if (menu == 4)
            Sig_Estado = 5;
        else if (menu == 5)

```

```

        Sig_Estado = 6;
    }
    else if (Estado == 2)
    {
        lcd_clear();
        lcd_text("Mediciones E/I");
        delay_ms(500);

        MenuMediciones();

        Sig_Estado = 1;
    }
    else if (Estado == 3)
    {
        lcd_clear();
        lcd_text("Par. Proceso");
        delay_ms(500);

        MenuAnalisisMotor();

        Sig_Estado = 1;
    }
    else if (Estado == 4)
    {
        lcd_clear();
        lcd_text("Sensores Campo");
        delay_ms(500);
        OS_TestDigitalOut();

        Sig_Estado = 1;
    }
    else if (Estado == 5)
    {
        lcd_clear();
        lcd_text("Config. Equipo");
        delay_ms(500);
        Sig_Estado = 1;
    }
    else if (Estado == 6)
    {
        lcd_clear();
        OS_ConfigDateTime();
        Sig_Estado = 1;
    }
}

return 0;
}

/* USER CODE BEGIN (4) */

void IpconfigAllServer()

```

```

{
    EnableServerCom();
    GetServerConfig();
    DisableServerCom();
}

void EnableServerCom()
{
    gpioSetBit(hetPORT1, 9, 1);
    //delay(10e-3);
}
void DisableServerCom()
{
    gpioSetBit(hetPORT1, 9, 0);
    // delay(10e-3);
}
void GetServerConfig()
{
    UARTgets(g_pcCmdBuf, sizeof(g_pcCmdBuf));

    char *token = strtok(g_pcCmdBuf, ",");
    memcpy(Wifi_Client.Wifi_LocalIP, token, strlen(token) + 1);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.Wifi_RSSI, token, strlen(token) + 1);
    Wifi_Client.RSSI = atoi(Wifi_Client.Wifi_RSSI);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.SSID, token, strlen(token) + 1);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.mqtt_server, token, strlen(token) + 1);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.mqtt_port, token, strlen(token) + 1);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.root_topic_publish, token, strlen(token) + 1);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.mqtt_user, token, strlen(token) + 1);
    token = strtok(NULL, ",");
    memcpy(Wifi_Client.mqtt_pass, token, strlen(token) + 1);
}

int menuANTIFALLOSLENTO(String *arrayMenu, int size)
{
    //Vamos a marcar en que tiempo se hizo cualquier cambio y si se hizo
    un cambio hace muy poco tiempo y se pulso, ese cambio le damos por malo.
    ok?
    //Pintamos el cursor y marcamos la primera opcion
    lcd_clear();
    lcd_gotoxy(0, 0);
    lcd_text("> ");
    int opcion = 1; //del 1 al 1.75 Opcion 1 //Del 2 al 2.75 opcion
2
    int extraOpcion = 0;

```

```

//float incremento = 0.5;
int x;
//Pinta los 4 primeros del menu como mucho
for (x = 0; x < size && x <= 3; x++) //
{
    lcd_gotoxy(2, x + 1);
    lcd_text(arrayMenu[x].data);
}

delay_ms(500);

//Si pulsamos el boton central sale del bucle
while (Read_Btn_ENTER() == 1)
{
    if (Read_Btn_MAS() == 0)
    {
        lcd_clear();
        if (opcion > 1)
        {
            opcion--;
        }
    }

    if (Read_Btn_MENOS() == 0)
    {
        lcd_clear();
        if (opcion < size)
        {
            opcion++;
        }
    }

    //Si sobrepasamos el limite por debajo
    if (opcion < 1 + extraOpcion)
        extraOpcion--;
    //Si sobrepasamos el limite por encima
    if (opcion > 2 + extraOpcion)
        extraOpcion++;

    //Pintamos de nuevo el menu

    for (x = extraOpcion; x < size && x <= (3 + extraOpcion); x++)
    { //
        lcd_gotoxy(2, (x + 1) - extraOpcion);
        lcd_text(arrayMenu[x].data);
    }
    //Pintamos el cursor
    lcd_gotoxy(1, opcion - extraOpcion);
    lcd_text(">");

    //Si pulsamos el boton de atras, salimos del bucle y devolvemos

```

```

    if (Read_Btn_SELECT() == 0)
    {
        //opcion = -1;
        return -1; //break
    }

    //Remember the last position of the dial so we know when it has
    changed
    //Last_DialPos1 = DialPos1;
}

//Aqui hemos salido del bucle ya que hemos pulsado el boton Enter o
ATRAS
delay_ms(250);

return opcion;
}

void Display_Clock()
{
    RTC_DS1307_Read();

    sprintf(Reloj.MSG_Fecha, "    %02u/%02u/%04u    ", Reloj.day,
    Reloj.mth,
        Reloj.year);
    sprintf(Reloj.MSG_Hora, "    %02u:%02u:%02u    ", Reloj.hr, Reloj.min,
        Reloj.sec);

    lcd_gotoxy(1, 1);                // envias coordenadas
    lcd_text(Reloj.MSG_Fecha);
    lcd_gotoxy(1, 2);                // envias coordenadas
    lcd_text(Reloj.MSG_Hora);
}

void rtiNotification(uint32 notification)
{
    /* enter user code between the USER CODE BEGIN and USER CODE END.
    */
    if (notification == rtiNOTIFICATION_COMPARE0) //Temporizador 250ms
    {
        LCD_Update = 1;
        // Temp_LCD++;
        if (Temp_LCD > 30)
        {
            // lcd_clear();          //limpiar pantalla LCD
            Temp_LCD = 31;
            LCD_Unlock = 1;
            EnableMenuLCD = 0;
            LCD_BACKLIGHT_ON_OFF(0); // PRENDER
        }

        //PulsoFlujos = edgeGetCounter(hetRAM1, edge7);
        //edgeResetCounter(hetRAM1, edge7);
    }
}

```

```

        capGetSignal(hetRAM1, 5, &Frec_signal);
        Flujometro = 1e6 / Frec_signal.period;
    }
    else if (notification == rtiNOTIFICATION_COMPARE1) //Temporizador
250ms
    {

    }

}

void adcNotification(adcBASE_t *adc, uint32 group)
{
    adcData_t data[10];
    adcGetData(adcREG1, adcGROUP0, data);

    Voltaje.wave_sampling_buff[ik] = data[0].value - data[2].value;
    Corriente.wave_sampling_buff[ik] = data[4].value - 1893;
    PresionPVC = data[3].value;
    TempPT_100 = data[5].value;
    Vibration_Wilcoxon_mA = data[1].value;

    ik++;

    if (ik > TEST_LENGTH_SAMPLES)
    {

        adcDisableNotification(adcREG1, adcGROUP0);
        ik = 0;
        arm_copy_f32(Voltaje.wave_sampling_buff, Voltaje.wave_sampling,
TEST_LENGTH_SAMPLES);
        arm_copy_f32(Corriente.wave_sampling_buff,
Corriente.wave_sampling,
TEST_LENGTH_SAMPLES);
        adcEnableNotification(adcREG1, adcGROUP0);
    }

}

/* USER CODE END */

```

ANEXO D: PROGRAMA - FUNCIONES ANEXAS:

```
#include "LCD_3LINES.h"
#include "LCD_Menus.h"
#include "DS1302.h"
#include "stdio.h"
#include "KERNEL_MCE.h" /*Kernel de Motor de calculo Electrico*/
#include "sci.h"
#include "uartstdio.h"

LCD_Msg DisplayLCD;
extern bool LCD_Update;
float Frec_Line;
uint8 btnSelect = 0;
uint8 MenuMostrar = 1;
//extern MotorCalculo Wave_Info;

extern float PresionPVC;
extern float FlujoMetro;
extern uint32 PulsoFlujos;
extern char Msg_Serial[]; // Mensajeria que se envia por protocolo
serial al ESP32
extern float PT_100;
extern float Vibration_mms;
extern float Vibration_Wilcoxon_mA;
float PT100Resistencia;
float PT100Voltaje;

extern float TempPT_100;
float Presion_scale;
float FlujoMetro_scale;
int anble_tx_wifi = 0;

const float vRef = 3.3; // Voltaje de referencia del Arduino
const float rRef = 100.0; // Resistencia de referencia en ohmios
const float r0 = 100.0; // Resistencia del PT100 a 0°C
const float alpha = 0.00385; // Coeficiente de temperatura del platino
void MenuMediciones()
{
    Frec_Line = 59.999;

    while (1)
    {
        // Captura de datos de los botones
        if (Buttons_dataAvailable() == 1)
        {
            btnSelect = Buttons_checkButtons();
            if (btnSelect == 'S')
            {
                break; // Salir del menú
            }
        }
    }
}
```

```

    }
}

if (LCD_Update == 1)
{
    LCD_Update = 0;

    Kernel_Calculo_Electrico(); // Cálculo de parámetros
    eléctricos
    Calc_Flow_Pres(); // Cálculo de presión, flujo y temperatura

    // Mostrar Voltaje RMS
    sprintf(DisplayLCD.LCD1, "Voltaje RMS ");
    sprintf(DisplayLCD.LCD2, "%3.2f V ", Wave_Info.VOLT_RMS);
    lcd_clear();
    lcd_gotoxy(1, 1);
    lcd_text(DisplayLCD.LCD1);
    lcd_gotoxy(1, 2);
    lcd_text(DisplayLCD.LCD2);
    delay_ms(1000); // Pequeño retraso para visualizar cada
    pantalla

    // Mostrar Corriente RMS
    sprintf(DisplayLCD.LCD1, "Corriente RMS ");
    sprintf(DisplayLCD.LCD2, "%3.2f A ", Wave_Info.AMP_RMS);
    lcd_clear();
    lcd_gotoxy(1, 1);
    lcd_text(DisplayLCD.LCD1);
    lcd_gotoxy(1, 2);
    lcd_text(DisplayLCD.LCD2);
    delay_ms(1000);

    // Mostrar Temperatura PT100
    sprintf(DisplayLCD.LCD1, "Temp PT100 ");
    sprintf(DisplayLCD.LCD2, "%3.2f C ", PT_100);
    lcd_clear();
    lcd_gotoxy(1, 1);
    lcd_text(DisplayLCD.LCD1);
    lcd_gotoxy(1, 2);
    lcd_text(DisplayLCD.LCD2);
    delay_ms(1000);

    // Mostrar Vibración en mm/s
    sprintf(DisplayLCD.LCD1, "Vibracion ");
    sprintf(DisplayLCD.LCD2, "%3.2f mm/s ", Vibration_mms);
    lcd_clear();
    lcd_gotoxy(1, 1);
    lcd_text(DisplayLCD.LCD1);
    lcd_gotoxy(1, 2);
    lcd_text(DisplayLCD.LCD2);
    delay_ms(1000);

    // Mostrar Presión
    sprintf(DisplayLCD.LCD1, "Presion ");

```

```

    sprintf(DisplayLCD.LCD2, "%3.2f MPa    ", Presion_scale);
    lcd_clear();
    lcd_gotoxy(1, 1);
    lcd_text(DisplayLCD.LCD1);
    lcd_gotoxy(1, 2);
    lcd_text(DisplayLCD.LCD2);
    delay_ms(1000);

    // Mostrar Flujo
    sprintf(DisplayLCD.LCD1, "Flujo    ");
    sprintf(DisplayLCD.LCD2, "%3.3f L/min ", Flujometro_scale);
    lcd_clear();
    lcd_gotoxy(1, 1);
    lcd_text(DisplayLCD.LCD1);
    lcd_gotoxy(1, 2);
    lcd_text(DisplayLCD.LCD2);
    delay_ms(1000);

    // Envío de todos los datos por serial
    sprintf(Msg_Serial,
            "%3.1f,%3.2f,%3.1f,%3.1f,%3.2f,%3.2f,%3.2f,%3.2f",
            Wave_Info.VOLT_RMS, Wave_Info.AMP_RMS,
            Wave_Info.POT_KW, Wave_Info.POT_KVA, Wave_Info.PF, Presion_scale,
            Flujometro_scale, PT_100);

    UARTprintf(Msg_Serial);
}
}

void MenuAnalisisMotor()
{
    Kernel_FFT_Init();
    Kernel_Init();

    while (1)
    {
        if (Buttons_dataAvailable() == 1)
        {
            btnSelect = Buttons_checkButtons();
            if (btnSelect == '+')
            {
                MenuMostrar--;
                if (MenuMostrar < 1)
                    MenuMostrar = 1;
            }
            else if (btnSelect == '-')
            {
                MenuMostrar++;
                if (MenuMostrar > 5)
                    MenuMostrar = 5;
            }
        }
    }
}

```

```

        else if (btnSelect == 'S')
        {
            break;
        }
    }

    if (LCD_Update == 1)
    {
        LCD_Update = 0;

        Calc_Flow_Pres();

        sprintf(DisplayLCD.LCD1, "%2.3f MPa ", Presion_scale);
        sprintf(DisplayLCD.LCD2, "%3.3f L/min ", Flujo_metro_scale);

        lcd_clear();           //limpiar pantalla LCD
        lcd_gotoxy(1, 1);      // envias
                                // coordenadas
        lcd_text(DisplayLCD.LCD1);           //envias texto
        lcd_gotoxy(1, 2);      // envias
                                // coordenadas
        lcd_text(DisplayLCD.LCD2);           //envias texto

        //Kernel_FFT_Calculo(); /*El kernel de la transformada de
        fourier se inicia segundo*/
        /*
        if (MenuMostrar == 1) {
            sprintf(DisplayLCD.LCD1, "Voltaje RMS ");
            sprintf(DisplayLCD.LCD2, "%3.2f V ", Wave_Info.VOLT_RMS);
        } else if (MenuMostrar == 2) {
            sprintf(DisplayLCD.LCD1, "Corriente RMS ");
            sprintf(DisplayLCD.LCD2, "%3.2f A ", Wave_Info.AMP_RMS);
        } else if (MenuMostrar == 3) {
            sprintf(DisplayLCD.LCD1, "Power Factor ");
            sprintf(DisplayLCD.LCD2, "%3.2f A ", Wave_Info.PF);
        } else if (MenuMostrar == 4) {
            sprintf(DisplayLCD.LCD1, "Potencia KW ");
            sprintf(DisplayLCD.LCD2, "%3.2f KW ", Wave_Info.POT_KW);
        } else if (MenuMostrar == 5) {
            sprintf(DisplayLCD.LCD1, "Potencia KVA ");
            sprintf(DisplayLCD.LCD2, "%3.2f KVA ",
Wave_Info.POT_KVA);
        }
        // */
    }
}

}

void Tx_MQTT_MSSG()
{

}

void Calc_Flow_Pres()

```

```

{
    PT100Voltaje = mapFloat(TempPT_100, 0, 4095, 0, 3.3); // Mapea el
    valor a la escala de presión (0 - 0.5MPa)
    PT100Resistencia = voltageToResistance(PT100Voltaje);
    PT_100 = resistanceToTemperature(PT100Resistencia) / 100;

    Presion_scale = mapFloat(PresionPVC, 0, 4095, 0, 0.5); // Mapea el
    valor a la escala de presión (0 - 0.5MPa)
    Flujometro_scale = Flujometro / 7.5;

    if (Flujometro_scale < 10)
        Flujometro_scale = 0;

    float current_mA = mapFloat(Vibration_Wilcoxon_mA, 0, 4095, 0, 3.3);
    // Mapea el valor a la escala de presión (0 - 0.5MPa)
    current_mA = (current_mA / 100) * 1000;

    // Aplicamos la ecuación para convertir la corriente en vibración
    (mm/s)
    if (current_mA < 4)
    {
        Vibration_mms = 0;
    }
    else if (current_mA > 20)
    {
        Vibration_mms = 12;
    }
    else
    {
        Vibration_mms = ((current_mA - 4) / 16) * 12;
    }
}

// Función para mapear valores con punto flotante
float mapFloat(float x, float in_min, float in_max, float out_min,
               float out_max)
{
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) +
    out_min;
}

float voltageToResistance(float voltage)
{
    return rRef * (voltage / (vRef - voltage));
}

float resistanceToTemperature(float resistance)
{
    return (resistance / r0 - 1) / alpha;
}

```

ANEXO E: PROGRAMA - FUNCIONES DE RELOJ

```
#include "LCD_3LINES.h"
#include "DS1307_24C.h"
#include "stdio.h"
// Variables para almacenar la hora y la fecha
int hora = 0;
int minuto = 0;
int segundo = 0;
int dia = 1;
int mes = 1;
int anio = 23;
extern RTC_DS1307 Reloj;
// Estado actual del menú
int estado_menu = 0;

void OS_ConfigDateTime()
{
    // Configuración del LCD
    lcd_clear();
    lcd_gotoxy(0, 0);
    lcd_text("Configuracion");

    //Inicializa los valores del reloj para empezar la configuracion
    RTC_DS1307_Read();

    hora = Reloj.hr;
    minuto = Reloj.min;
    segundo = Reloj.sec;
    dia = Reloj.day;
    mes = Reloj.mth;
    anio = Reloj.year;

    while (1)
    { // Manejo de los botones
        if (Read_Btn_MAS() == 0)
        {
            switch (estado_menu)
            {
            case 0: // Configuración de la hora
                hora = (hora + 1) % 24;
                break;
            case 1: // Configuración de la hora
                minuto = (minuto + 1) % 60;
                break;
            case 2: // Configuración del día
                dia = (dia % 31) + 1;
                break;
            case 3: // Configuración del mes
                mes = (mes % 12) + 1;
                break;
            }
        }
    }
}
```

```

        case 4: // Configuración del año
            anio++;
            break;
    }
    mostrar_pantalla();
}

if (Read_Btn_MENOS() == 0)
{
    switch (estado_menu)
    {
        case 0: // Configuración de la hora
            hora = (hora + 23) % 24;
            break;
        case 1: // Configuración de la hora
            minuto = (minuto + 59) % 60;
            break;
        case 2: // Configuración del día
            dia = (dia + 29) % 31 + 1;
            break;
        case 3: // Configuración del mes
            mes = (mes + 10) % 12 + 1;
            break;
        case 4: // Configuración del año
            anio--;
            break;
    }
    mostrar_pantalla();
}

if (Read_Btn_ENTER() == 0)
{
    estado_menu = (estado_menu + 1) % 5;
    mostrar_pantalla();
    /* if (estado_menu == 6 || estado_menu == 7)
        break;*/
}

if (Read_Btn_SELECT() == 0)
{
    if (estado_menu == 0)
    {
        estado_menu = 4;
    }
    else
    {
        estado_menu--;
    }
    mostrar_pantalla();
}
}

```

```

    if ((Read_Btn_ENTER() == 0) && (Read_Btn_SELECT() == 0))
    {
        RTC_DS1307_datetime(dia, mes, anio, 0, hora, minuto);
        break;
    }
}

// Función para mostrar la hora y la fecha en la pantalla
void mostrar_pantalla()
{
    char MsngLCD[16] = "";

    lcd_clear();
    lcd_gotoxy(0, 0);
    switch (estado_menu)
    {
        case 0:
            lcd_text("Hora: ");
            sprintf(MsngLCD, "%02u", hora);
            lcd_text(MsngLCD);
            lcd_text(":");
            sprintf(MsngLCD, "%02u", minuto);
            lcd_text(MsngLCD);
            break;

        case 1:
            lcd_text("Min: ");
            sprintf(MsngLCD, "%02u", hora);
            lcd_text(MsngLCD);
            lcd_text(":");
            sprintf(MsngLCD, "%02u", minuto);
            lcd_text(MsngLCD);
            break;

        case 2:
            lcd_text("Dia: ");
            sprintf(MsngLCD, "%02u", dia);
            lcd_text(MsngLCD);
            lcd_text("/");
            sprintf(MsngLCD, "%02u", mes);
            lcd_text(MsngLCD);
            lcd_text("/");
            sprintf(MsngLCD, "%02u", anio);
            lcd_text(MsngLCD);

            break;

        case 3:
            lcd_text("Mes: ");
            sprintf(MsngLCD, "%02u", mes);
            lcd_text(MsngLCD);
            lcd_text("/");
            sprintf(MsngLCD, "%02u", anio);
            lcd_text(MsngLCD);
            break;

        case 4:

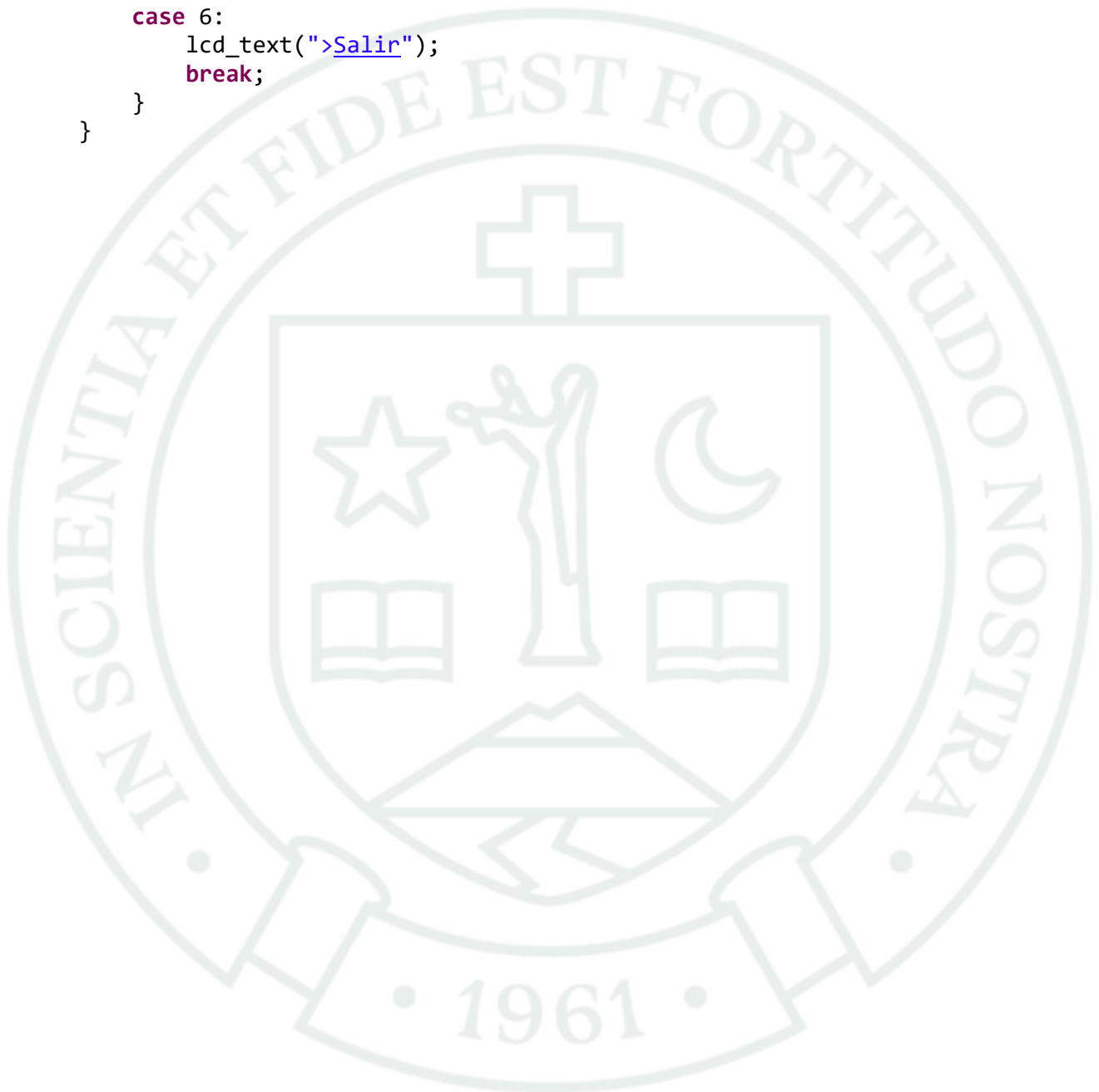
```

```

    lcd_text("<a href='\"#\"'>Anio: ");
    sprintf(MsgnLCD, "%02u", anio);
    lcd_text(MsgnLCD);
    break;
case 5:
    lcd_text("<a href='\"#\"'>Guardar/Salir");

    break;
case 6:
    lcd_text("<a href='\"#\"'>Salir");
    break;
}
}

```



ANEXO F: PROGRAMA - CONFIGURACIÓN DE RELOJ

```
#include "LCD_3LINES.h"
#include "DS1307_24C.h"
#include "stdio.h"

void OS_TestDigitalOut()
{
    // Configuración del LCD
    lcd_clear();
    lcd_gotoxy(0, 0);
    lcd_text("Pruebas");
    lcd_gotoxy(0, 1);
    lcd_text("I/O");

    while (1)
        ;
}

void digitalWrite_Relay(int code, int value)
{
    if (code == 1)
    {
        //ENABLE
        if (value == 0)
            gpioSetBit(hetPORT1, 12, LOW);           // ENABLE LCD
        else
            gpioSetBit(hetPORT1, 12, HIGH);           // ENABLE LCD
    }
    else if (code == 2)
    {
        //CLOCK
        if (value == 0)
            gpioSetBit(hetPORT1, 14, LOW);           // CLOCK LCD
        else
            gpioSetBit(hetPORT1, 14, HIGH);           // CLOCK LCD
    }
    else if (code == 3)
    {
        //DATOS
        if (value == 0)
            gpioSetBit(hetPORT1, 16, LOW);           // DATOS DE LCD
        else
            gpioSetBit(hetPORT1, 16, HIGH);           // DATOS DE LCD
    }
    else if (code == 4)
    {
        //DATOS
        if (value == 0)
            gpioSetBit(hetPORT1, 21, LOW);           // DATOS DE LCD
        else
            gpioSetBit(hetPORT1, 21, HIGH);           // DATOS DE LCD
    }
}
}
```

ANEXO G: PROGRAMA - MOTOR DE CÁLCULO ELÉCTRICO

```
#include "KERNEL_MCE.h" /*Kernel de Motor de calculo Electrico*/
/* -----
 * Declare Test output buffer
 * -----
 */
static float32_t testOutput[TEST_LENGTH_SAMPLES];
/* -----
 * Declare State buffer of size (numTaps + blockSize - 1)
 * -----
 */
static float32_t firStateF32[BLOCK_SIZE + NUM_TAPS - 1];
/* -----
 ** FIR Coefficients buffer generated using fir1() MATLAB function.
 ** Frecuencia de corte 80Hz, solo para la frecuencia fundamental
 ** -----
 */
const float32_t firCoeffs32[] = { 0.0009095779f, 0.0009165537f,
0.0009327398f,
                                0.0009581813f, 0.0009929053f,
0.0010369207f,
                                0.0010902176f, 0.0011527681f,
0.0012245254f,
                                0.0013054244f, 0.0013953818f,
0.0014942955f,
                                0.0016020456f, 0.0017184941f,
0.0018434850f,
                                0.0019768450f, 0.0021183833f,
0.0022678922f,
                                0.0024251475f, 0.0025899086f,
0.0027619192f,
                                0.0029409078f, 0.0031265877f,
0.0033186582f,
                                0.0035168047f, 0.0037206994f,
0.0039300021f,
                                0.0041443605f, 0.0043634111f,
0.0045867799f,
                                0.0048140829f, 0.0050449271f,
0.0052789112f,
                                0.0055156262f, 0.0057546565f,
0.0059955802f,
                                0.0062379707f, 0.0064813968f,
0.0067254239f,
                                0.0069696147f, 0.0072135306f,
0.0074567315f,
                                0.0076987778f, 0.0079392305f,
0.0081776526f,
                                0.0084136094f, 0.0086466701f,
0.0088764080f,
```

0.0095415010f,	0.0091024015f,	0.0093242353f,
0.0101619249f,	0.0097537978f,	0.0099607334f,
0.0107273611f,	0.0103569994f,	0.0105455950f,
0.0112283671f,	0.0109019597f,	0.0110690656f,
0.0116565478f,	0.0113795671f,	0.0115223830f,
0.0120047084f,	0.0117818105f,	0.0118979366f,
0.0122669861f,	0.0121019258f,	0.0121894064f,
0.0124389569f,	0.0123345191f,	0.0123918786f,
0.0125177169f,	0.0124756654f,	0.0125019352f,
0.0125019352f,	0.0125229807f,	0.0125177169f,
0.0123918786f,	0.0124756654f,	0.0124389569f,
0.0121894064f,	0.0123345191f,	0.0122669861f,
0.0118979366f,	0.0121019258f,	0.0120047084f,
0.0115223830f,	0.0117818105f,	0.0116565478f,
0.0110690656f,	0.0113795671f,	0.0112283671f,
0.0105455950f,	0.0109019597f,	0.0107273611f,
0.0099607334f,	0.0103569994f,	0.0101619249f,
0.0093242353f,	0.0097537978f,	0.0095415010f,
0.0086466701f,	0.0091024015f,	0.0088764080f,
0.0079392305f,	0.0084136094f,	0.0081776526f,
0.0072135306f,	0.0076987778f,	0.0074567315f,
0.0064813968f,	0.0069696147f,	0.0067254239f,
0.0057546565f,	0.0062379707f,	0.0059955802f,
0.0050449271f,	0.0055156262f,	0.0052789112f,
0.0043634111f,	0.0048140829f,	0.0045867799f,
0.0037206994f,	0.0041443605f,	0.0039300021f,
0.0031265877f,	0.0035168047f,	0.0033186582f,

```

0.0025899086f,          0.0029409078f,          0.0027619192f,
0.0021183833f,          0.0024251475f,          0.0022678922f,
0.0017184941f,          0.0019768450f,          0.0018434850f,
0.0013953818f,          0.0016020456f,          0.0014942955f,
0.0011527681f,          0.0013054244f,          0.0012245254f,
0.0009929053f,          0.0010902176f,          0.0010369207f,
0.0009165537f,          0.0009581813f,          0.0009327398f,
0.0009095779f, };

/* -----
 * Global variables for FIR LPF Example
 * -----
 */
uint32_t blockSize = BLOCK_SIZE;
uint32_t numBlocks = TEST_LENGTH_SAMPLES / BLOCK_SIZE;
/* -----
---
** Inicializando el Filtro Principal
** -----
- */
uint32_t i;
arm_fir_instance_f32 S;
float32_t *inputF32, *outputF32;
/* -----
 * Global variables for FFT Bin Example
 * -----
 */
arm_cfft_radix4_instance_f32 FFT;
uint32_t fftSize = 1024;
uint32_t ifftFlag = 0;
uint32_t doBitReverse = 1;
float32_t Time_Length_VI[10];

float32_t FFt_Buffer[2048];
uint16 FFT_index;
extern float Frec_Line;
//Harmonics_data Voltaje, Corriente;

//extern Harmonics_data Voltaje, Corriente;
void Kernel_Calculo_Coeficientes()
{ //solo para calibrar el equipo
/* -----
-----
** Contruccion del Motor de Calculo etapa de voltaje
** Calculo del Voltaje RMS de la Tensio Electrica
** -----
----- */
arm_mean_f32(Voltaje.wave_sampling, TEST_LENGTH_SAMPLES,

```

```

        &Wave_Info.VOLT_RMS);
    /* -----
    -----
    ** Etapa para el calculo de los valores RMS de Corriente
    ** -----
    ----- */
    arm_mean_f32(Corriente.wave_sampling, TEST_LENGTH_SAMPLES,
        &Wave_Info.AMP_RMS);
}
void Kernel_Calculo_DC()
{
    /* -----
    -----
    ** Contruccion del Motor de Calculo etapa de voltaje
    ** Calculo del Voltaje RMS de la Tensio Electrica
    ** -----
    ----- */
    arm_mean_f32(Voltaje.wave_sampling, TEST_LENGTH_SAMPLES,
        &Wave_Info.VOLT_RMS);

    Wave_Info.VOLT_PROM = Wave_Info.VOLT_RMS * RESOLT_ADC *
VOLT_CONVERSION;

    /* -----
    -----
    ** Etapa para el calculo de los valores RMS de Corriente
    ** -----
    ----- */
    /* arm_mean_f32(Corriente.wave_sampling, TEST_LENGTH_SAMPLES,
        &Wave_Info.AMP_RMS);

    Wave_Info.AMP_RMS = (Wave_Info.AMP_RMS * RESOLT_ADC
        * CURRNT_CONVERSION) * CURRNT_RESOLT;*/
}
float Kernel_Get_Voltaje_RMS()
{
    /* -----
    -----
    ** Contruccion del Motor de Calculo etapa de voltaje
    ** Calculo del Voltaje RMS de la Tensio Electrica
    ** -----
    ----- */
    arm_rms_f32(Voltaje.wave_sampling, TEST_LENGTH_SAMPLES,
        &Wave_Info.VOLT_RMS);

    Wave_Info.VOLT_RMS = Wave_Info.VOLT_RMS * RESOLT_ADC *
VOLT_CONVERSION;
    return Wave_Info.VOLT_RMS;
}
float Kernel_Get_Frecuencia()
{
    /* -----
    -----
    ** Etapa para el calculo de la Frecuencia de Wave_Infoa

```

```

    ** -----
    ----- */
    Wave_Info.FREC = (float) (1e6 / T_prom); //RTI0, cada 10ms muestrea
la frecuencia de linea.
    FREC_LINEA = Wave_Info.FREC;
    return Wave_Info.FREC;
}
void Kernel_Get_Voltaje_Corriente_Potencia_RMS()
{
    /* -----
    -----
    ** Contruccion del Motor de Calculo etapa de voltaje
    ** Calculo del Voltaje RMS de la Tensio Electrica
    ** -----
    ----- */
    arm_rms_f32(Voltaje.wave_sampling, TEST_LENGTH_SAMPLES,
                &Wave_Info.VOLT_RMS);

    Wave_Info.VOLT_RMS = Wave_Info.VOLT_RMS * RESOLT_ADC *
    VOLT_CONVERSION;

    /* -----
    -----
    ** Etapas para el calculo de los valores RMS de Corriente
    ** -----
    ----- */
    arm_rms_f32(Corriente.wave_sampling, TEST_LENGTH_SAMPLES,
                &Wave_Info.AMP_RMS);

    Wave_Info.AMP_RMS = (Wave_Info.AMP_RMS * RESOLT_ADC * FACTOR_RESIS
    - OFFSET_CURRENT) * CURRNT_CONVERSION;

    if (Wave_Info.AMP_RMS < 0)
        Wave_Info.AMP_RMS = 0.0;

    /* -----
    -----
    ** Potencia Aparente por fase
    ** -----
    ----- */
    Wave_Info.POT_KVA = Wave_Info.VOLT_RMS * Wave_Info.AMP_RMS * UNIT_K;
}

uint8 Kernel_Cruce_Cero_Voltaje()
{
    int ii = 0;
    int size_wave = 0;
    int wave_pos, wave_neg;
    int temp_x_pos, temp_x_neg;
    float value_pos, value_neg;
    uint8 contrR;
    /* -----
    -----
    ** Inicializo vectores de cruce por cero

```

```

    ** -----
    ----- */

    for (ii = 0; ii < 10; ii++)
    {
        Voltaje.Wave_ZeroUP[ii] = 0;
    }

    /* -----
    -----
    ** Ajusto sencibilidad del trigger
    ** -----
    ----- */
    Wave_signal.Value_diff = 3;
    /* -----
    -----
    ** Busqueda de ceros en la forma de onda
    ** -----
    ----- */
    contrR = Zero_Seach(Voltaje.wave_sampling, Voltaje.Wave_ZeroUP,
                        &Wave_signal);
    /* -----
    -----
    ** Calcula la longitud de la onda
    ** -----
    ----- */
    size_wave = Voltaje.Wave_ZeroUP[1] - Voltaje.Wave_ZeroUP[0];
    /* -----
    -----
    ** Calcula la zona positiva y negativa
    ** -----
    ----- */
    wave_pos = size_wave / 4;    //Busco el 1/4 de onda debe ser siempre
    positivo
    wave_neg = 3 * size_wave / 4; //Busco el 1/4 de onda debe ser siempre
    positivo

    temp_x_pos = Voltaje.Wave_ZeroUP[0] + wave_pos;
    temp_x_neg = Voltaje.Wave_ZeroUP[0] + wave_neg;
    /* -----
    -----
    ** Extrae los valores de la onda alterna
    ** -----
    ----- */
    value_pos = Voltaje.wave_sampling[temp_x_pos];
    value_neg = Voltaje.wave_sampling[temp_x_neg];

    /* -----
    -----
    ** Analiza si es un flanco positivo
    ** -----
    ----- */
    if ((value_pos > 0) && (value_neg < 0))
        return contrR;

```

```

        else
            return 0xff;
    }
    /*Vectores que almacenan las formas de Onda*/
    void Kernel_Calculo_Electrico()
    {
        /* -----
        -----
        ** Contruccion del Motor de Calculo etapa de voltaje
        ** Calculo del Voltaje RMS de la Tensio Electrica
        ** -----
        ----- */
        arm_rms_f32(Voltaje.wave_sampling, TEST_LENGTH_SAMPLES,
                    &Wave_Info.VOLT_RMS);

        Wave_Info.VOLT_RMS = Wave_Info.VOLT_RMS * RESOLT_ADC *
        VOLT_CONVERSION;

        /* -----
        -----
        ** Etap para el calculo de los valores RMS de Corriente
        ** -----
        ----- */
        arm_rms_f32(Corriente.wave_sampling, TEST_LENGTH_SAMPLES,
                    &Wave_Info.AMP_RMS);

        Wave_Info.AMP_RMS = (Wave_Info.AMP_RMS * RESOLT_ADC *
        CURRNT_CONVERSION)
        * CURRNT_RESOLT / 10;

        if (Wave_Info.AMP_RMS < 0.5)
            Wave_Info.AMP_RMS = 0;

        /* -----
        -----
        ** Etap para el calculo de la Frecuencia de Wave_Infoa
        ** -----
        ----- */
        // arm_mean_f32(Wave_Info_Period, SAMPLING_PERID, &PERIODO_LINEA);
        Wave_Info.FREC = 59.9; //Frec_Line;

        /*if (Wave_Info.FREC > 69) || Wave_Info.VOLT_RMS < 30)
            Wave_Info.FREC = 0.00;*/

        /* -----
        -----
        ** Inicializo vectores de cruce por cero
        ** -----
        ----- */
        int ii = 0;

        for (ii = 0; ii < 10; ii++)
        {
            Voltaje.Wave_ZeroUP[ii] = 0;
        }
    }
}

```

```

}

/* -----
** VOLTAJE
** Busqueda de Ceros en la señal Muestreada, luego de pasar por
** un filtro digital, solo se trabaja con la fundamental de las
** señales.
** -----
---- */

inputF32 = &Voltaje.wave_sampling[0];
for (i = 0; i < numBlocks; i++)
{
    arm_fir_f32(&S, inputF32 + (i * blockSize), outputF32 + (i *
blockSize),
                blockSize);
}
uint8 contr = Zero_Seach(testOutput, Voltaje.Wave_ZeroUP,
&Wave_signal);
arm_rms_f32(testOutput, TEST_LENGTH_SAMPLES, &Wave_Info.VOLT_FUND);
// Calculo RMS de solo la fundamental

/* -----
** Etapas para el calculo del valor RMS de la fundamental
** -----
---- */
Wave_Info.VOLT_FUND = Wave_Info.VOLT_FUND * RESOLT_ADC *
VOLT_CONVERSION;

for (ii = 0; ii < 10; ii++)
{
    Corriente.Wave_ZeroUP[ii] = 0;
}
/* -----
** ASUMO QUE EL VOLTAJE SE ENCUENTRA ADELANTE DE LA
** CORRIENTE
** Calculo de angulo de desfase entre el voltaje y la corriente
** -----
---- */
inputF32 = &Corriente.wave_sampling[0];
for (i = 0; i < numBlocks; i++)
{
    arm_fir_f32(&S, inputF32 + (i * blockSize), outputF32 + (i *
blockSize),
                blockSize);
}
contr = Zero_Seach(testOutput, Corriente.Wave_ZeroUP, &Wave_signal);
arm_rms_f32(testOutput, TEST_LENGTH_SAMPLES, &Wave_Info.AMP_FUND);
// Calculo RMS de solo la fundamental

```

```

/* -----
** Etapas para el calculo del valor RMS de la fundamental
** -----
*/
Wave_Info.AMP_FUND = (Wave_Info.AMP_FUND * RESOLT_ADC *
CURRNT_CONVERSION)
* CURRNT_RESOLT;

for (ii = 0; ii < SEACH_L_ZERO; ii++)
{
    Time_Length_VI[ii] = Voltaje.Wave_ZeroUP[ii]
- Corriente.Wave_ZeroUP[ii];
}
arm_rms_f32(Time_Length_VI, SEACH_L_ZERO, &Wave_Info.AMP_PSH);

/*Conversion de valores enteros en angulo de fase*/
Wave_Info.AMP_PSH = (Wave_Info.AMP_PSH) * TSAMPLING
* ((float) PSHVALUE * Wave_Info.FREC);

if (Voltaje.Wave_ZeroUP[4] > Corriente.Wave_ZeroUP[4])
{
    Wave_Info.AMP_PSH = Wave_Info.AMP_PSH;
}
else
{
    Wave_Info.AMP_PSH = -Wave_Info.AMP_PSH;
}

/*Calculo del factor de potencia
* para el calculo se requiere que la señal este en radianes*/
Wave_Info.PF = cos(Wave_Info.AMP_PSH * MPI);

if (Wave_Info.FREC > FREC_MAX)
{
    Wave_Info.VOLT_RMS = 0.00;
    Wave_Info.VOLT_PK = 0.00;
    Wave_Info.VOLT_CF = 0.00;
    Wave_Info.AMP_RMS = 0.00;
    Wave_Info.AMP_PK = 0.00;
    Wave_Info.AMP_CF = 0.00;
}
/* -----
** Potencia Activa por fase
** -----
*/
Wave_Info.POT_KW = Wave_Info.VOLT_RMS * Wave_Info.AMP_RMS *
Wave_Info.PF
* UNIT_K;

/*Potencia Activa de la fundamental*/
Wave_Info.POT_KW_FUND = Wave_Info.VOLT_FUND * Wave_Info.AMP_FUND
* Wave_Info.PF * UNIT_K;

```

```

/* -----
** Potencia Reactiva por fase
** -----
*/
Wave_Info.POT_KVAR = Wave_Info.VOLT_RMS * Wave_Info.AMP_RMS
* sin(Wave_Info.AMP_PSH * MPI) * UNIT_K;

Wave_Info.POT_KVAR_FUND = Wave_Info.VOLT_FUND * Wave_Info.AMP_FUND
* sin(Wave_Info.AMP_PSH * MPI) * UNIT_K;

/* -----
** Potencia Aparente por fase
** -----
*/
Wave_Info.POT_KVA = Wave_Info.VOLT_RMS * Wave_Info.AMP_RMS * UNIT_K;

/*Potencia Aparente de la fundamental*/
Wave_Info.POT_KVA_FUND = Wave_Info.VOLT_FUND * Wave_Info.AMP_FUND *
UNIT_K;

/* -----
** Potencia Armonica
** -----
*/
Wave_Info.POT_KVA_HARM = sqrt(
pow(Wave_Info.POT_KVA, 2) + pow(Wave_Info.POT_KVA_FUND, 2));

/* -----
** Calculo de Valores Maximo de la Señal Electrica
** -----
*/
arm_max_f32(Voltaje.wave_sampling, TEST_LENGTH_SAMPLES,
&Wave_Info.VOLT_PK,
&Voltaje.max_marks1);

Wave_Info.VOLT_PK = Wave_Info.VOLT_PK * RESOLT_ADC *
VOLT_CONVERSION;

/* -----
** Corriente Maxima de corriente
** -----
*/
arm_max_f32(Corriente.wave_sampling, TEST_LENGTH_SAMPLES,
&Wave_Info.AMP_PK,
&Corriente.max_marks1);
Wave_Info.AMP_PK = (Wave_Info.AMP_PK * RESOLT_ADC *
CURRNT_CONVERSION)
* CURRNT_RESOLT;

/* -----

```

```

    ** Calculo del Factor de Cresta en las Señales
    ** -----
    ---- */
    Wave_Info.VOLT_CF = Wave_Info.VOLT_PK / Wave_Info.VOLT_RMS;

    Wave_Info.AMP_CF = Wave_Info.AMP_PK / Wave_Info.AMP_RMS;

}

void Kernel_WaveProm()
{
    /* -----
    ----
    ** Contruccion del Motor de Calculo etapa de voltaje
    ** Calculo del Voltaje RMS de la Tensio Electrica
    ** -----
    ---- */
    inputF32 = &Voltaje.wave_sampling[0];
    for (i = 0; i < numBlocks; i++)
    {
        arm_fir_f32(&S, inputF32 + (i * blockSize), outputF32 + (i *
        blockSize),
                    blockSize);
    }
    arm_mean_f32(testOutput, TEST_LENGTH_SAMPLES,
    &Wave_Info.VOLT_PROM); // Calculo RMS de solo la fundamental
}
/*
void Write_Kernel_Factor(float VALUE_CHA, float VALUE_CHB, float
VALUE_CHC,
float VALUE_CHN, bool FACTORY_RST) {

    if (FACTORY_RST == 1) {
        VALUE_CHA = 1050.8727267675;
        VALUE_CHB = 1050.8727267675;
        VALUE_CHC = 1050.8727267675;
        VALUE_CHN = 1050.8727267675;
    }
    int Config_Electrica = (int) READ_FLOAT_EXT_EEPROM(ADDRS_CONF_ELEC);
    if (Config_Electrica == 1 || Config_Electrica == 2
    || Config_Electrica == 3) {
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_1PH_CH_A, VALUE_CHA);
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_1PH_CH_A, VALUE_CHB);
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_1PH_CH_A, VALUE_CHC);
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_1PH_CH_A, VALUE_CHN);
    } else {
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_CH_A, VALUE_CHA);
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_CH_B, VALUE_CHB);
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_CH_C, VALUE_CHC);
        WRITE_FLOAT_EXT_EEPROM(VOLT_CONVERSION_CH_N, VALUE_CHN);
    }
}
}*/

```

```

void Kernel_Init()
{
    /* -----
    -----
    ** Inicializando el Filtro Principal
    ** -----
    ----- */
    outputF32 = &testOutput[0];
    /* Call FIR init function to initialize the instance structure. */
    arm_fir_init_f32(&S, NUM_TAPS, (float32_t*) &firCoeffs32[0],
                    &firStateF32[0], blockSize);
    /* -----
    -----
    ** Motor de Calculo en Variable
    ** -----
    ----- */
    Wave_signal.Value_inf = 0;
    Wave_signal.Value_Sup = 0;
    Wave_signal.Value_diff = 0;
    Wave_signal.LENGHT_DATA = TEST_LENGTH_SAMPLES;
}

void Kernel_FFT_Init()
{
    /* Initialize the CFFT/CIFFT module */
    arm_status status = arm_cfft_radix4_init_f32(&FFT, fftSize,
    ifftFlag,
    doBitReverse);
}

void Kernel_FFT_Calculo()
{
    /* -----
    -----
    ** Calcula la transformada de Fourier / Voltaje
    ** -----
    ----- */
    arm_cfft_radix4_f32(&FFT, Voltaje.wave_sampling);/* Process the
    data through the CFFT/CIFFT module */

    /* -----
    -----
    ** Calcula la transformada de Fourier / Corriente
    ** -----
    ----- */
    arm_cfft_radix4_f32(&FFT, Corriente.wave_sampling);/* Process the
    data through the CFFT/CIFFT module */

    /* -----
    -----
    ** Calculo de la magnitud / Voltaje

```

```

    ** -----
    ---- */
    arm_cmplx_mag_f32(Voltaje.wave_sampling,          Voltaje.fft_MAG,
fftSize);/* Process the data through the Complex Magnitude Module for
calculating the magnitude at each bin */
    /* -----
    ----
    ** Calculo de la magnitud / Corriente
    ** -----
    ---- */
    arm_cmplx_mag_f32(Corriente.wave_sampling,          Corriente.fft_MAG,
fftSize);/* Process the data through the Complex Magnitude Module for
calculating the magnitude at each bin */

//gst_spectrum_window(Corriente.wave_sampling, fftSize);

    /* -----
    ----
    ** Calculo de la Fase / Voltaje
    ** -----
    ---- */
    arm_cmplx_phase_f32(Voltaje.wave_sampling,          Voltaje.fft_PHASE,
fftSize);

    /* -----
    ----
    ** Calculo de la Fase / Corriente
    ** -----
    ---- */
    arm_cmplx_phase_f32(Corriente.wave_sampling,          Corriente.fft_PHASE,
fftSize);

    /*FFT de la potencia, multiplifacion DFT(V)*DFT(I)*/
    arm_cmplx_mult_cmplx_f32(Voltaje.wave_sampling,
Corriente.wave_sampling,
          Potencia.wave_sampling, fftSize);
    arm_cmplx_mag_f32(Potencia.wave_sampling,          Potencia.fft_MAG,
fftSize);

    /* -----
    ----
    ** Ajuste de armonicos en posiciones multiplos de la fundamental
    ** Existe la probabilidad de que el armonico no se encuentre
    ** en la posicion multiplo del fundamental, por lo tanto
    ** se requiere realizar una busqueda y ajuste en las componentes
    ** cercanas al armonico.
    ** -----
    ---- */
    Seach_Harmonic(Voltaje.fft_MAG, Voltaje.fft_MAG, FFT_X_HARM);
    Seach_Harmonic(Corriente.fft_MAG, Corriente.fft_MAG, FFT_X_HARM);
    Seach_Harmonic(Potencia.fft_MAG, Potencia.fft_MAG, FFT_X_HARM);
    /* -----
    ----
    ** Busqueda del Maximo, valor del espectro / Voltaje

```

```

    ** -----
    ----- */
    arm_max_f32(Voltaje.fft_MAG, fftSize / 2, &Voltaje.valor_MAX,
                &Voltaje.max_marks1);

    /* -----
    -----
    ** Busqueda del Maximo, valor del espectro / Corriente
    ** -----
    ----- */
    arm_max_f32(Corriente.fft_MAG, fftSize / 2, &Corriente.valor_MAX,
                &Corriente.max_marks1);

    /* -----
    -----
    ** Busqueda del Maximo, valor del espectro / Potencia
    ** -----
    ----- */
    arm_max_f32(Potencia.fft_MAG, fftSize / 2, &Potencia.valor_MAX,
                &Potencia.max_marks1);

    /* -----
    -----
    ** Modificacion del Espectro
    ** -----
    ----- */
    arm_scale_f32(Voltaje.fft_MAG, FFT_MAX_HIGH / Voltaje.valor_MAX,
                  Voltaje.fft_MAG, fftSize / 2);

    arm_scale_f32(Corriente.fft_MAG, FFT_MAX_HIGH / Corriente.valor_MAX,
                  Corriente.fft_MAG, fftSize / 2);

    arm_scale_f32(Potencia.fft_MAG, FFT_MAX_HIGH / Potencia.valor_MAX,
                  Potencia.fft_MAG, fftSize / 2);

    int fi = 0;
    float32_t THDV_R = 0;
    float32_t THDI_R = 0;
    float32_t THDP_R = 0;

    for (fi = 2 * FFT_X_HARM; fi < 250; fi += FFT_X_HARM)
    {
        THDV_R += Voltaje.fft_MAG[fi] * Voltaje.fft_MAG[fi];
        THDI_R += Corriente.fft_MAG[fi] * Corriente.fft_MAG[fi];
        THDP_R += Potencia.fft_MAG[fi] * Potencia.fft_MAG[fi];
    }

    THDV_R = sqrt(THDV_R);
    THDI_R = sqrt(THDI_R);
    THDP_R = sqrt(THDP_R);

    Voltaje.valor_MAX = Voltaje.fft_MAG[Voltaje.max_marks1];
    Corriente.valor_MAX = Corriente.fft_MAG[Corriente.max_marks1];
    Potencia.valor_MAX = Potencia.fft_MAG[Potencia.max_marks1];

```

```

Voltaje.THd = (float32_t) (THDV_R / Voltaje.valor_MAX) * 100;
Corriente.THd = (float32_t) (THDI_R / Corriente.valor_MAX) * 100;
Potencia.THd = (float32_t) (THDP_R / Potencia.valor_MAX) * 100;

if (Voltaje.THd > 100)
    Voltaje.THd = 100;
if (Corriente.THd > 100)
    Corriente.THd = 100;
if (Potencia.THd > 100)
    Potencia.THd = 100;
/*
if (FREC_LINEA < 50) {
    Voltaje.THd = 0.0;
    VoltajeS.THd = 0.0;
    VoltajeT.THd = 0.0;
    Corriente.THd = 0.0;
    CorrienteS.THd = 0.0;
    CorrienteT.THd = 0.0;
    Potencia.THd = 0.0;
    PotenciaS.THd = 0.0;
    PotenciaT.THd = 0.0;
}*/
}
void arm_cmplx_phase_f32(float32_t *pSrc, float32_t *pDst, uint32_t
numSamples)
{
    float32_t realIn, imagIn; /* Temporary variables to hold input
values */

#ifdef ARM_MATH_CM0_FAMILY

    /* Run the below code for Cortex-M4 and Cortex-M3 */
    uint32_t blkCnt; /* loop counter */

    /*loop Unrolling */
    blkCnt = numSamples >> 2u;

    /* First part of the processing with loop unrolling.  Compute 4
outputs at a time.
    ** a second loop below computes the remaining 1 to 3 samples. */
    while (blkCnt > 0u)
    {

        /* C[0] = atan2(A[1] / A[0]) */
        realIn = *pSrc++;
        imagIn = *pSrc++;
        /* store the result in the destination buffer. */
        arm_atan_f32(imagIn, realIn, pDst++);

        realIn = *pSrc++;
        imagIn = *pSrc++;
    }

```

```

    arm_atan_f32(imagIn, realIn, pDst++);

    realIn = *pSrc++;
    imagIn = *pSrc++;
    arm_atan_f32(imagIn, realIn, pDst++);

    realIn = *pSrc++;
    imagIn = *pSrc++;
    arm_atan_f32(imagIn, realIn, pDst++);

    /* Decrement the loop counter */
    blkCnt--;
}

/* If the numSamples is not a multiple of 4, compute any remaining
output samples here.
** No loop unrolling is used. */
blkCnt = numSamples % 0x4u;

while (blkCnt > 0u)
{
    /* C[0] = atan2(A[1] / A[0]) */
    realIn = *pSrc++;
    imagIn = *pSrc++;
    /* store the result in the destination buffer. */
    arm_atan_f32(imagIn, realIn, pDst++);

    /* Decrement the loop counter */
    blkCnt--;
}

#else

/* Run the below code for Cortex-M0 */

while(numSamples > 0u)
{
    /* out = sqrt((real * real) + (imag * imag)) */
    realIn = *pSrc++;
    imagIn = *pSrc++;
    /* store the result in the destination buffer. */
    arm_sqrt_f32((realIn * realIn) + (imagIn * imagIn), pDst++);

    /* Decrement the loop counter */
    numSamples--;
}

#endif /* #ifndef ARM_MATH_CM0_FAMILY */

}

/**
 * @brief Floating-point square root function.
 * @param[in] in input value.

```

```

    * @param[out] *pOut square root of input value.
    * @return The function returns ARM_MATH_SUCCESS if input value is
    positive value or ARM_MATH_ARGUMENT_ERROR if
    * <code>in</code> is negative value and returns zero output for
    negative values.
    */

void arm_atan_f32(float32_t iny, float32_t inx, float32_t *pOut)
{
    *pOut = (float32_t) atan2(iny, inx) * 180 / 3.1415;
}
/**
 * @} end of cmplx_mag group
 */

/** @fn float32_t Seach_Harmonic(void)
 * @brief Realiza el calculo del armonico
 * de mayor potencia, cerca del valor enviado por el Index
 *
 * Esta funcion devuelve el valor del armonico de mayor potencia
 * Comprobado con hoja de datos de excel
 */
void Seach_Harmonic(float32_t *pSrc, float32_t *pDst, uint32 Index)
{
    float32_t temp_x_1, temp_x, temp_x_2;
    float32_t Pow_harmonic = 0;
    uint32 ii = 0;
    for (ii = Index; ii < LENGTH_SAMPLES / 2; ii += Index)
    {
        /*Busqueda y promedio de las componentes
        * cercanadas al armonico*/
        temp_x_1 = pow(pSrc[ii - 1], 2);
        temp_x = pow(pSrc[ii], 2);
        temp_x_2 = pow(pSrc[ii + 1], 2);
        Pow_harmonic = temp_x_1 + temp_x + temp_x_2;
        Pow_harmonic = sqrt(Pow_harmonic);

        /*Reestableciendo el valor del vector principal*/
        pDst[ii] = Pow_harmonic;
    }
}

uint16 Seaching_Trigger(float32_t *pSrc, Trigger_funcion
*Signal_Analizer)
{
    int jh = 0, trigger_look = 0;
    // int centrotrigger = Signal_Analizer->LENGHT_DATA / 2;
    int centrotrigger = 0;
    int h = 52, FX1, FX0; //h=50 indica el espacio entre las muestras
    50*40us=2ms

```

```

float AY_AX = 0.0;

float32_t Dato_Actual = 0, Dato_Anterior = 0, diff_val = 0;
Buscar_Zero:
/* -----
-----
** Tigger Digital por algoritmo
** -----
---- */
do
{
    Dato_Actual = pSrc[centrotrigger + jh + 1] * 100;
    Dato_Anterior = pSrc[centrotrigger + jh] * 100;
    diff_val = Dato_Actual - Dato_Anterior;

    bool Var1 = Dato_Anterior < -100;
    bool var2 = Dato_Actual > 100;

    if (Var1 && var2 && diff_val > 200) //&& Dato_Actual < 20 &&
diff_val > 15)

        trigger_look = 1;
    else
    {
        trigger_look = 0;
    }

    jh++;
}
while ((trigger_look == 0) && (jh < LENGTH_SAMPLES / 2));

if (jh == LENGTH_SAMPLES / 2)
    return jh; // retorna el valor del trigger que se busca de
acuerdo las características

/* -----
-----
** Busqueda del flanco de subida
** -----
---- */
FX0 = pSrc[jh];
FX1 = pSrc[jh + h];
AY_AX = (float) (FX1 - FX0) / h;

if (AY_AX > 0)
    return jh; // retorna el valor del trigger que se busca de
acuerdo las características
else
    goto Buscar_Zero;

}
/*Solo funciona para detectar el flanco de subida de la señal senoidal
* */
uint8 Zero_Search(float32_t *pSrc, uint16 *pDstUP,

```

```

        Trigger_funcion *Signal_Analyzer)
{
    int jh = 0, diff_val = 0, centrotrigger = 0, cont_zero_up = 0;
    float32_t Dato_Actual = 0, Dato_Anterior = 0;
    float32_t Valor_diff = Signal_Analyzer->Value_diff;

    do
    {
        Dato_Actual = pSrc[centrotrigger + jh + 1];    //2 Muestras por
adelante
        Dato_Anterior = pSrc[centrotrigger + jh];
        diff_val = Dato_Actual - Dato_Anterior;

        if ((Dato_Actual > 0) && (Dato_Anterior < 0)
            && (diff_val >= Valor_diff))
        {
            pDstUP[cont_zero_up] = jh + centrotrigger;
            cont_zero_up++;
        }
        jh++;
    }
    while (jh < (TEST_LENGTH_SAMPLES - centrotrigger));

    return cont_zero_up;
}

void InitSamplingADC()
{
    adcStartConversion(adcREG1, adcGROUP0);
    adcStartConversion(adcREG2, adcGROUP0);
}

void StopSamplingADC()
{
    adcStopConversion(adcREG1, adcGROUP0);
    adcStopConversion(adcREG2, adcGROUP0);
}

```