UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MONITOREO DE VIBRACIONES EN UN MOTOR, UTILIZANDO TARJETA EMBEBIDA Y TRANSMISIÓN DE DATOS A LA NUBE

Trabajo de titulación previo a la obtención del

Título de Ingeniero en Electrónica

AUTORES: Sebastián Alfredo Román Espinoza

Alberto Andrés Pesantes Morales

TUTOR: Ing. Rafael Franco Reina, MSc

GUAYAQUIL – ECUADOR

2025 - 2026

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Sebastián Alfredo Román Espinoza con documento de identificación N°0957261613 y Alberto Andrés Pesantes Morales con documento de identificación N°0953862125, manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación

Guayaquil, 12 de septiembre del año 2025.

Atentamente,

Sebastián Alfredo Román Espinoza
0957261613

Alberto Andrés Pesantes Morales
0953862125

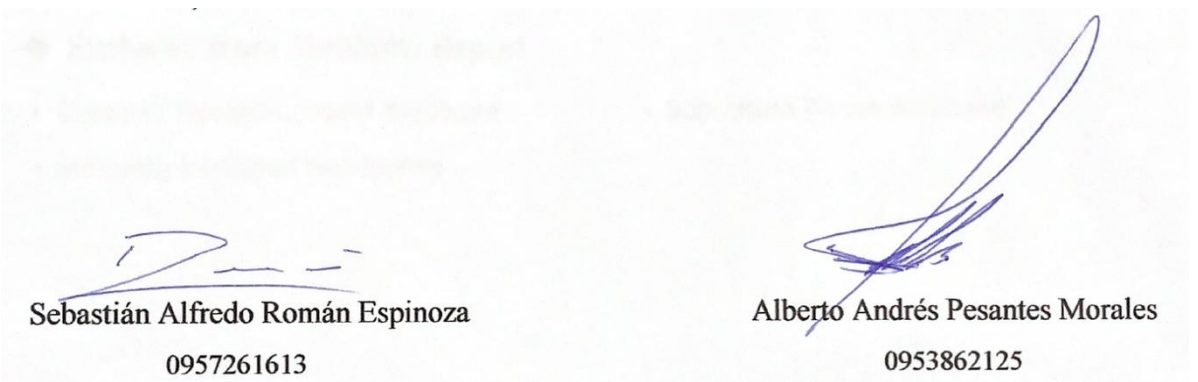
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Sebastián Alfredo Román Espinoza con documento de identificación N°0957261613 y Alberto Andrés Pesantes Morales con documento de identificación N°0958045189, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MONITOREO DE VIBRACIONES EN UN MOTOR, UTILIZANDO TARJETA EMBEBIDA Y TRANSMISIÓN DE DATOS A LA NUBE", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica**,** quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 12 de septiembre del año 2025.

Atentamente,



Sebastián Alfredo Román Espinoza
0957261613

Alberto Andrés Pesantes Morales
0953862125

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Rafael Christian Franco Reina con documento de identificación N°0923328629, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación**:** DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MONITOREO DE VIBRACIONES EN UN MOTOR, UTILIZANDO TARJETA EMBEBIDA Y TRANSMISIÓN DE DATOS A LA NUBE , realizado por Sebastián Alfredo Román Espinoza con documento de identificación N°0957261613 y Alberto Andrés Pesantes Morales con documento de identificación N°0958045189, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 12 de septiembre del año 2025.

Atentamente,

Ing. Rafael Christian Franco Reina, MSc.
0923328629

DEDICATORIA

Primeramente, a Dios por darme la fuerza, la sabiduría y la perseverancia para seguir adelante, incluso en los momentos más difíciles. Sin su guía, este logro no había sido posible

Dedico este trabajo de titulación a mi familia, especialmente a mis padres, por ser mi pilar fundamental. Gracias por su amor incondicional, por cada sacrificio silencioso y por enseñarme el valor del esfuerzo y la dedicación.

A mis seres queridos, por su apoyo constante, sus palabras de aliento y por estar siempre a mi lado. Este logro es también de ustedes.

Con todo mi corazón, gracias.

Sebastián Alfredo Román Espinoza

Dedico este trabajo de titulación a mis padres que siempre estuvieron ahí para darme la fuerza y el apoyo constante sin importar cuantos tropiezos haya tenido en el proceso.

También esta dedicatoria va para una de las personas más importantes en mi vida que ya no está con nosotros Alicia Elfida Duran Salinas mi abuela, desde el cielo este logro es para ti.

Alberto Andrés Pesantes Morales

AGRADECIMINETO

Quiero expresar mi agradecimiento a Dios por guiarme y darme fuerza en todo momento.

A mi familia, por su apoyo incondicional, paciencia y amor que me han impulsado a seguir adelante. También agradezco a la universidad, que me brindo los conocimientos y recursos necesarios para crecer académica y personalmente. Sin ellos, este logro no habría sido posible.

Sebastián Alfredo Román Espinoza

Primero que todo agradecerle a Dios para darme la fuerza para seguir y no desistir, a su vez agradecerles a mis padres por su apoyo incondicional esto no hubiera sido posible sin ustedes, a su vez a la institución por los conocimientos obtenidos y por último también agradecerle a mis compañeros y futuros colegas por siempre apoyarme cuando más uno los necesita.

Alberto Andrés Pesantes Morales

RESUMEN

En el presente trabajo de titulación se propone el diseño e implementación de un prototipo funcional que permite monitorear las vibraciones en un motor eléctrico, con el objetivo de detectar posibles fallas de forma temprana y práctica. Este sistema está basado en el uso de una tarjeta embebida ESP32 y un sensor acelerómetro, integrando además la capacidad de transmitir los datos recolectados a una plataforma en la nube para su almacenamiento y análisis remoto.

Durante el desarrollo del proyecto, se identifica que las vibraciones anormales en motores eléctricos pueden deberse a múltiples causas como desalineaciones, desequilibrios o fallas estructurales, las cuales, si no se detectan a tiempo, pueden comprometer el funcionamiento del motor y generar daños mayores. A partir de esta problemática, diseñe un prototipo que no solo sirve como herramienta técnica, sino también como apoyo educativo para estudiantes que deseen comprender el comportamiento mecánico de los motores y como enfrentarlos mediante tecnología actual.

Como resultado, se desarrolló un prototipo que permita medir las vibraciones de los motores mediante el (Acelerómetro MPU6050), este sensor realiza un monitoreo del estado en el que se encuentra el motor, además, el sensor proporciona los datos en tiempo real sobre las aceleraciones que ejerce el motor y detecta cualquier tipo de anomalías como desbalances, desalineaciones y desgastes de componentes mecánicos. Para el procesamiento de los datos se escogió la tarjeta embebida ESP32 ya que es de bajo consumo, fácil de programar y con conectividad Wi-Fi, los datos recolectados se suben a la nube mediante el protocolo de comunicación IOT, generando una carpeta de Excel de las señales registradas, con el fin de aplicar la transforma de Fourier a estos datos recolectados, dando como resultado que el Motor 1 muestra un comportamiento estable, con valores constantes en el tiempo (~60) y una FFT plana (~0.8), indicando un buen balance, en cambio el Motor 2 presenta inestabilidad, con oscilaciones entre 65 y 100 y picos marcados en 10, 20 y 30 Hz, dando a reflejar a posible desbalance o desgaste mecánico. De esta manera se determina que el prototipo implementado y el análisis de la data cumplen con el objetivo del proyecto.

Palabras Claves

Motores eléctricos, vibraciones, ESP32, acelerómetro, datos en la nube, mantenimiento predictivo.

ABSTRACT

This thesis proposes the design and implementation of a functional prototype that allows the monitoring of vibrations in an electric motor, with the aim of detecting possible faults early and practically. This system is based on the use of an ESP32 embedded card and an accelerometer sensor, also integrating the ability to transmit the collected data to a cloud platform for storage and remote analysis.

During the development of the project, it was identified that abnormal vibrations in electric motors can be caused by multiple factors such as misalignment, imbalance, or structural failures, which, if not detected in time, can compromise the operation of the motor and cause major damage. Based on this problem, I designed a prototype that not only serves as a technical tool but also as an educational aid for students who want to understand the mechanical behavior of motors and how to address it using current technology.

As a result, a prototype was developed to measure motor vibrations using the MPU6050 accelerometer, which monitors the motor's condition in real time by providing acceleration data and detecting anomalies such as imbalances, misalignments, and mechanical wear. For data processing, the ESP32 embedded board was selected due to its low power consumption, ease of programming, and Wi-Fi connectivity, enabling the collected data to be uploaded to the cloud through the IoT communication protocol and stored in Excel files for further analysis. Applying the Fourier Transform to the recorded signals showed that Motor 1 exhibited stable behavior, with constant values over time (~60) and a flat FFT (~0.8), indicating good balance, while Motor 2 displayed instability, with oscillations between 65 and 100 and marked peaks at 10, 20, and 30 Hz, suggesting possible imbalance or mechanical wear. Therefore, it is concluded that the implemented prototype and the data analysis successfully meet the project's objective.

Keywords

Electric motors, vibrations, ESP32, accelerometer, cloud data, predictive maintenance

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLA

I INTRODUCCIÓN

Las vibraciones en motores son provocadas por múltiples factores, como desequilibrio o problema en los rodamientos, con eso se plantea la necesidad de tener una herramienta practica para diagnosticar y monitorear posibles fallas. Es así que, el prototipo permite captar señales de vibración generadas con un motor en funcionamiento.

Por tanto, el objetivo es diseñar e implementar un prototipo funcional que permita monitorear las vibraciones de un motor de forma práctica y accesible. Para lograrlo, se utiliza una tarjea embebida ESP32 junto con sensores acelerómetro, los cuales permiten medir en tiempo real y enviar los datos a la nube. De esta manera, poder visualizar la información de forma remota, detectar cambios en los patrones de vibraciones y anticipar posibles fallas.

Este trabajo no solo busca aportar una solución técnica, sino también convertirse en una herramienta didáctica para que otros estudiantes pueden aprender y experimentar con tecnologías actuales como los sistemas embebidos, el internet de las Cosa (IoT) y el análisis de datos a la nube. La implementación de este prototipo fomenta el aprendizaje práctico y la capacidad de aplicar soluciones innovadoras en contexto reales, acercando la formación académica a las necesidades de la industria.

II PROBLEMA

En la actualidad, los motores eléctricos son componentes importantes en la mayoría de los procesos industriales y educativos, de acuerdo con su buen rendimiento, durabilidad y también bajo costo operativo. Sin embargo, a pesar de su fiabilidad, estos tipos de motores normalmente funcionan en las industrias de formas exigentes, la cual influyen en su rendimiento y provoca fallos precipitados (Arias & Sánchez, 2024).

Los datos obtenidos de informes y estudios, los problemas comunes de los motores eléctricos son las vibraciones que afectan directo al motor. Pueden sufrir desalineaciones, desequilibrios del rotor o problema en la índole estructural y si no se detectan a tiempo, puede causar daños permanentes en los sistemas mecánicos y eléctricos del motor (Auqui & Farez, 2024).

A pesar del que el estudio de vibraciones es también un método que se puede utilizar para identificar el tipo de problema, ya que en lo universitario no se utiliza de manera adecuada. Por lo tanto, es importante como explorar o cómo se puede implementar de manera práctica para el estudio de vibraciones en los motores existente en los laboratorios de la universidad. Ya que las vibraciones de un motor deben de estar con una magnitud aceptables y constante, pero al deterioro de estos componentes la vibración aumentara con el tiempo. (Mejía, 2009).

III OBJETIVOS

3.1 Objetivo general

Implementar un prototipo que permita monitorear las vibraciones de un motor mediante una tarjeta embebida y transmisión de datos en la nube.

3.2 Objetivos específicos

- Diseñar un sistema funcional de monitoreo de vibraciones en motores eléctricos, utilizando una tarjeta embebida ESP32 y un sensor acelerómetro.
- Implementar la transmisión de datos recolectados a una plataforma en la nube, para el almacenamiento y visualización remota de las vibraciones registradas.
- Evaluar el funcionamiento del prototipo mediante dos practicas experimentales de los estudiantes, acercándolos al fortalecimiento de sus habilidades.

IV FUNDAMENTO TEÓRICO

4.1 Módulo didáctico

Un módulo didáctico, también denominado modulo institucional, es un recurso educativo que integra todos los elementos necesarios para el aprendizaje de conceptos y destrezas del estudiante, sin intervención presencial continua del del instructor. Esta herramienta responde a una planificación pedagógica estructurada que facilita el aprendizaje autónomo, como se muestra en la figura 1 (Fajardo, Andino, & Fernández, s.f.).

Figura 1
Modulo didáctico

Nota. La gráfica muestra modulo para el proyecto (EA-AUTOMATAS, 2024).

4.2 PLC S7 - 1500 de Siemens

S7-1500 representa una solución de automatización industrial de alta gama, caracterizada por su elevado rendimiento, seguridad integrada y gran adaptabilidad. Equipado con CPU de alta velocidad como el CPU 1513-1 PN, con frecuencia de hasta 1,2 GHz, está preparada para abordad aplicaciones industriales exigentes. Sus avanzadas funciones de diagnósticos y mantenimientos predictivo contribuyen a mejorar la fiabilidad del sistema y minimizar los tiempos de parada. Asimismo, este controlador destaca por su conectividad robusta mediante protocolos como PROFINET Y PROFIBUS, además de su capacidad de integración con plataformas en la nube, lo que hace compatible con los principios de la industria 4.0. El entorno de programación TIA Portal simplifica la puesta en marcha y configuración del sistema, mientras que su facilidad de expansión y durabilidad aseguran una solución escalable y de larga vida útil para distintas aplicaciones industriales, como se muestra en la figura 2 (Peciña, 2019).

Figura 2
PLC S7 - 1500

Nota. La ilustración muestra el modelo del PLC S7 - 1500 de Siemens (AG., 2025).

4.3 HMI KTP700 Basic

El uso de interfaces HMI en Siemens no solo implica la conexión entre el PLC y la pantalla, sino también el desarrollo estructurado de proyectos dentro de un entorno de programación. El TIA portal resulta esencial, ya que permite integrar tanto la lógica del PLC como el diseño de las interfaces HMI en una misma plataforma, lo cual facilita la gestión de proyectos completos. (Bee, 2022).

Además de la parte técnica, el diseño visual y la experiencia del operador tienen un papel relevante. Siemens ofrece un cuaderno de trabajo con recomendaciones, plantillas y buenas prácticas que contribuyen a optimizar la claridad y usabilidad de las pantallas HMI, asegurando así una interacción más eficiente con los sistemas de automatización. (Siemens, 2025).

Siemens ha desarrollado manuales especializados, como la guía de diseño HMI para el sistema BRAUMAT. Este documento abordado la configuración de alarmas, mensaje, tendencias y soporte multilingüe, lo que evidencia la importancia de las HMI como herramientas avanzadas para la supervisión y control en procesos industriales complejos, a continuación, se mostrara el HMI en la figura 3. (Siemens, 2014).

Figura 3
HMI KTP700

Nota. HMI se mostrará los botones de encendido y apagado del motor (GmbH, 2025).

Características del HMI

- Tipo de display: TFT panorámica, retroiluminación LED
- Conexión Profinet
- Teclas función: 8
- Tensión de alimentación: 24VDC
- Consumo (valor nominal): 125mA
- Configurable con Wincc. TIA Portal >13
- Consumo de potencia activa: 3W
- Resolución: 800 * 480 PIXELES.
- Diagonal de pantalla: 7’’
- Número de colores: 65536
- Procesador ARM
- MTBF de la retroiluminación: 20.000 horas
- Con memoria flash y RAM
- Interfaz industrial Ethernet
- Interfaz USB
- Ventana de Avisos, de recetas, de alarmas

4.4 Fuente Simatic pm 1507

Es un módulo de alimentación Power Module diseñado para integrarse con los controladores SIMATIC S7-1500 y los sistemas ET 200Mp. Se detecta por su conmutación automática de rango de entrada, lo que permite conectarlo a redes monofásicas de 120v o 230v AC sin necesidad de configuración manual. Aporta una salida regulada de 24v DC, ideal para alimentar CPU, circuitos de entrada- salidas y sensores o actuadores auxiliares. Además,

tiene capacidad para buffering durante cortes de red y cuenta con opciones de salida de 3A o 8A, dependiendo de la variante utilizada, como se muestra en la figura 4 (Axmatic, 2025).

Figura 4
Fuente de alimentación

Nota. *Esta* fuente alimentara al PLC Y HMI (Electrónicas, 2025).

4.5 Variador de frecuencia v20

Un variador de frecuencia VFD, por sus siglas en ingles es un dispositivo electrónico que permite controlar la velocidad de un motor de corriente alterna mediante la variación de las frecuencias y el voltaje suministrado al motor. Este control se logra a través de un proceso que incluye la conversión de corriente alterna con las características deseadas. Los VDF son esenciales en aplicaciones industriales donde se requiere un control preciso de velocidad del motor, como en bombas, ventiladores y compresores, en la figura 5 se muestra (Domínguez, 2025).

Figura 5
Variador V20

Nota. Variador de frecuencia del motor 1 y 2 (Eléctricos, 2025).

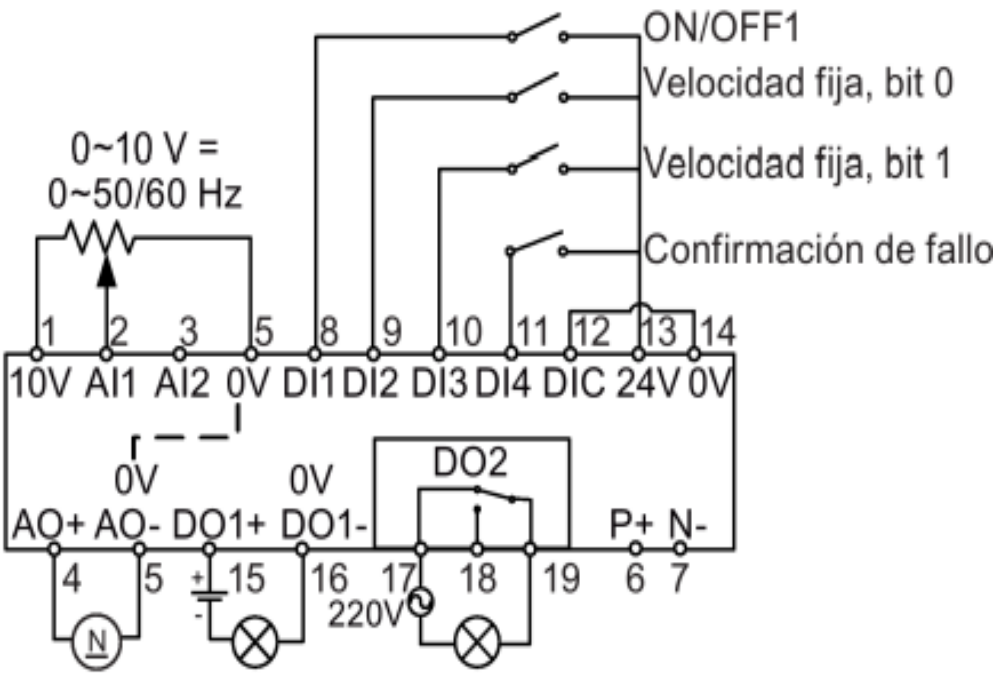
4.5.1 Características del Variado V20

- Rango de potencia: 0,12 kW hasta 30 kW.
- Tensión de red: 1AC 200–240 V y 3AC 380–480 V.
- Frecuencia de salida: 0 a 650 Hz.
- Grado de protección: IP20.
- Temperatura de operación: –10 °C a +60 °C.
- Sobrecarga: 150 % durante 60 s.
- Incluye macros de conexión predefinidas.
- Funciones de ahorro de energía (modo ECO y hibernación).
- Opciones de comunicación: USS y Modbus RTU.
- Panel básico integrado para puesta en marcha rápida.

4.5.2 Configuración

El variador SINAMICS V20 puede configurarse fácilmente para control de velocidad mediante un potenciómetro conectado a la entrada analógica AI1 (0-10). Para ello, se debe establecer P0700= 2 y P1000=2, como se muestra en la figura 6 (Siemens, 2016).

Figura 6
Configuración SINAMICS V20

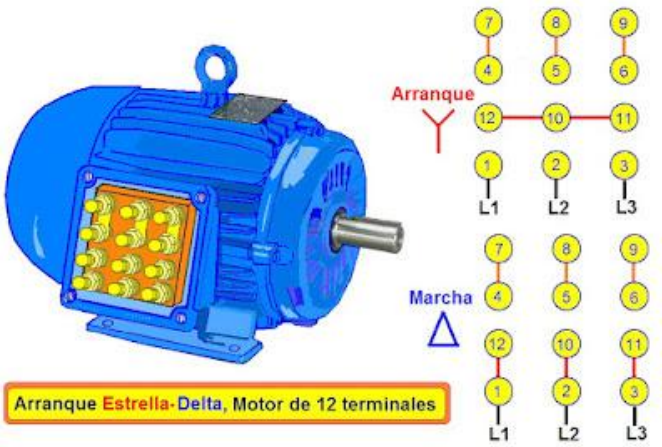


*Nota***.** Parámetros que se utilizaron en el proyecto (Siemens, 2020).

4.6 Motor Trifásico de 12 terminales

Los motores trifásicos tienen como objetivo que al trabajar con corriente alterna trifásica o corriente trifásica mandan su propia tensión que varía o cambia de manera periódica en tres conductores separados, cuyo tiempo se traslada a 120° hacia delante o hacia atrás de las tensiones de los demás conductores. Si alimentan las tres bobinas de electroimán, cada tensión con una conductora de la fase del sistema trifásico, en cada una de la bobina y eso hace que genere un campo magnético cuya temporización, al igual que la progresión de la tensión, se compensa en parte de período con respecto a los otros campos de bobina, en la figura 7 se mostrara el motor trifásico (Escuela universitaria de oficios UNLP, 2020).

Figura 7
Motor trifásico



Nota. Parte interna del motor trifásico (Coparoman, 2016).

4.6.1 Características técnicas

- Los 12 terminales corresponden a las salidas de los seis devanados del estator.
- Permiten realizar conexiones en estrella (Y), doble estrella (YY), triángulo (Δ) y doble triángulo
- Permite trabajar en dos tensiones diferentes según la conexión: Ejemplo 220/380 V, 380/660 V o 440/760 V.
- Frecuencia y velocidad Normalmente 50/60 Hz, con velocidades síncronas de 3000, 1500 o 1000 rpm
- Se fabrican bajo normas IEC o NEMA, con grados de protección típicos IP55 / IP65 y clase de aislamiento F.

4.6.2 Conexión estrella (Y)

La conexión estrella (Y) es una configuración comúnmente empleada en el arranque de motores eléctricos trifásicos. En esta disposición, los terminales de las bobinas del estator se conectan en un punto común, formando una figura geométrica similar a la estrella. Esta configuración permite que cada bobina reciba una tensión de fase reducida, lo que resulta en una corriente de arranque significativamente menor en comparación con la conexión en triangulo (Δ). Este método es especialmente útil para reducir el impacto de la corriente de arranque en la red eléctrica y en los componentes del motor, como se muestra en la siguiente figura 8 (Fernández, 2023).

Figura 8
Conexión estrella

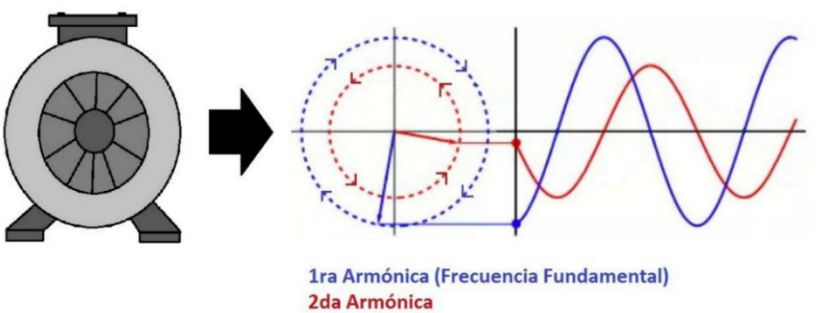
Nota. Conexiones que se usaron en el motor AC (Tecnología, 2025).

4.7 Armónicos

Es importante saber que los equipos o componentes electrónicos de potencia suelen a ser muy delicados y tienden a ser muy susceptibles al punto de que tengan un mal funcionamiento si hay niveles significativos de distorsión armónica los motores se pueden ver afectados negativamente por los armónicos en las redes a las que están conectados (Grajales, Ramírez, & Cadavid, 2004).

La distorsión armónica suele provocar un desplazamiento de los puntos de cruce de voltaje cero, y estas variaciones suelen ser críticas para muchos circuitos de control electrónico. Además, si se llega a producir una conmutación incorrecta, se pueden producir más armónicos, agravando el problema en los equipos, a continuación, en la figura 9 se muestra la variación de los filtros armónicos (Ramírez, 2024).

Figura 9
Motor y las características de los armónicos



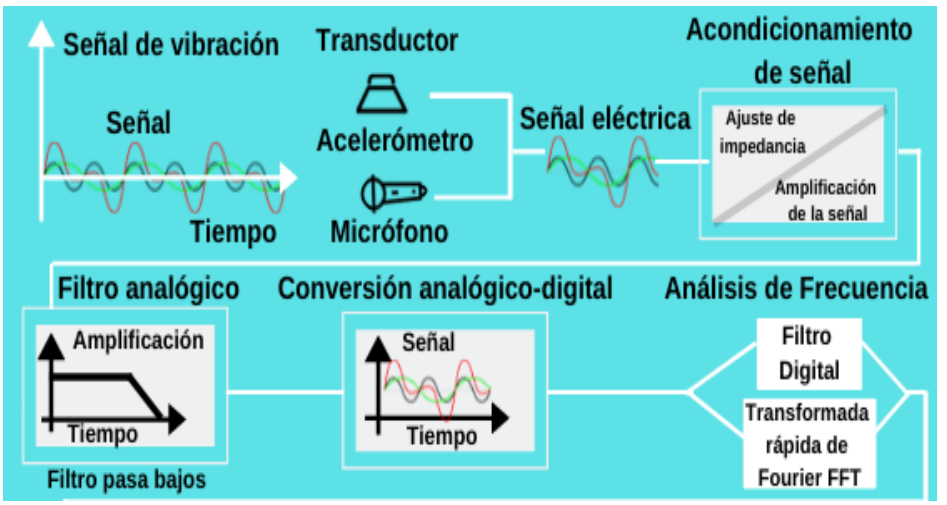
Nota. La gráfica muestra los filtros armónicos del motor (Motores y Generadores, 2023).

4.8 Vibraciones

La vibración se entiende como la oscilación de una masa respecto a su posición de equilibrio, generada por fuerzas dinámicas internas o externas, ya que toda máquina rotativa presenta vibraciones propias que pueden verse alteradas por factores como desalineaciones, fricción, desgaste o desbalance, por ello, los analizadores de vibraciones resultan esenciales para monitorear niveles, ya que su correcta interpretación permite anticipar fallas y tomar decisiones de mantenimiento más efectivas, como se muestra en la figura 10 (Mobley, 2002).

Figura 10

Análisis de vibraciones



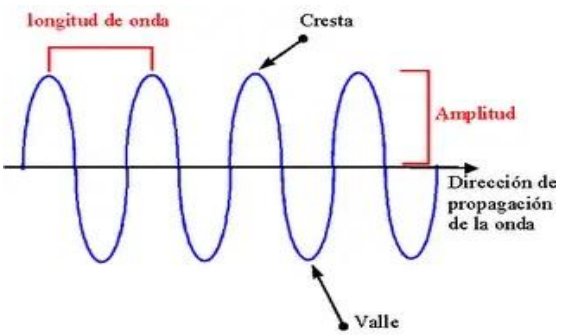
Nota. Análisis de vibraciones que se pueden utilizar (Comunitario, 2021).

4.8.1 Características de una onda de vibración

Esta se caracteriza por tres parámetros principales: frecuencia, número de ciclos por segundo medidos en Hertz (Hz); período, tiempo requerido para completar un ciclo, siendo el inverso de la frecuencia; y amplitud, magnitud máxima del desplazamiento respecto al equilibrio, como se muestra en la figura 11 (Meirovitch, 2001)

Figura 11

Característica de vibración

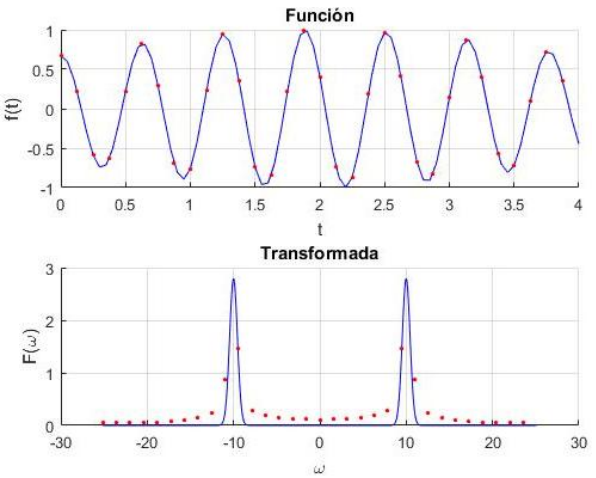


Nota. Estas con las 3 características de la vibración (Comunitario, 2021).

4.9 Transformada de Fourier

La Transformada de Fourier es una herramienta matemática fundamental para el análisis de señales y sistemas. Que permite descomponer cualquier señal en sus componentes de frecuencia, representando una señal en el dominio de la frecuencia en lugar del dominio del tiempo. Esta descomposición facilita el estudio de la energía y el comportamiento de señales periódicas o no periódicas, permitiendo identificar las frecuencias que componen una señal compleja, como se muestra en la siguiente figura 12 (Martínez, 2008).

Figura 12
Fourier



Nota. Ejemplo de gráfica aplicando la transformada de Fourier (Vasco, 2025).

4.9.1 Formula de la Transformada Discreta de Fourier

$$x[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j} \frac{2\pi}{N} kn$$

X[k] Componente espectral en la frecuencia k. Representa la amplitud y fase de la frecuencia k en la señal.

x[n] Señal discreta en el tiempo, con n=0,1, 2, …, N−1.

N Número total de muestras de la señal.

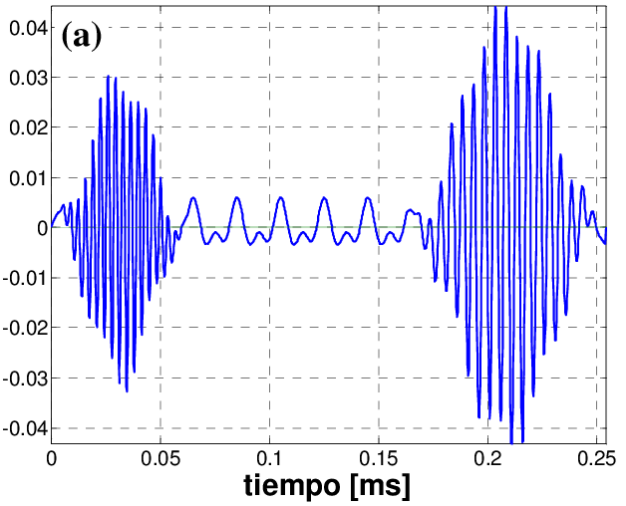
$\mathbf{e^{-j}\frac{2\pi}{N}kn}$ Factor complejo que descompone la señal en sus componentes de frecuencia, parte de los coeficientes de la base exponencial compleja.

La sumatoria $\sum_{n=0}^{N-1}$ recorre todas las muestras de la señal.

4.9.2 Dominio del tiempo

Un análisis de vibraciones en el dominio del tiempo se basa en estudiar cómo cambia la vibración de un motor o maquinas a lo largo del tiempo, ya que esto permite reconocer patrones o picos que pueden señalar fallas tempranas, a diferencia del análisis en frecuencia, este enfoque es útil para detectar problemas como impactos, desbalanceo y desalineaciones con comportamientos irregulares, como se muestra en la figura 13 (Randall, 2011)

Figura 13
Dominio del tiempo

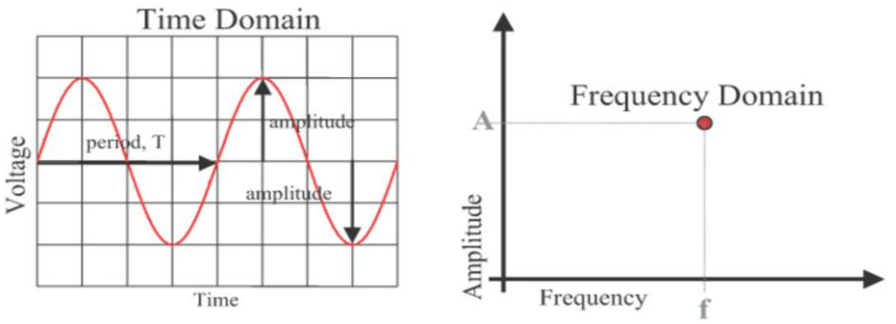


Nota. Ejemplo de gráfica aplicando dominio del tiempo. (Montejo, 2007)

4.9.3 Dominio de la frecuencia

El análisis en el dominio de la frecuencia estudia cómo se reparte la energía de una señal entre distintas frecuencias, ya que ayuda a identificar patrones o componentes repetitivos en las vibraciones de los motores, lo que permite detectar problemas como desbalanceo, desalineación o fallas en los rodamientos, a diferencia del análisis en el tiempo, que muestra cómo cambia la vibración con el tiempo, el enfoque en frecuencia revela las causas de estas vibraciones, facilitando un diagnóstico más preciso, como se muestra en la figura 14 (Pruftechnik, 2011).

Figura 14
Dominio de la frecuencia



Nota. Ejemplo de gráfica aplicando dominio de la frecuencia (Wiki, 2025)

4.10 Acelerómetro MPU6050

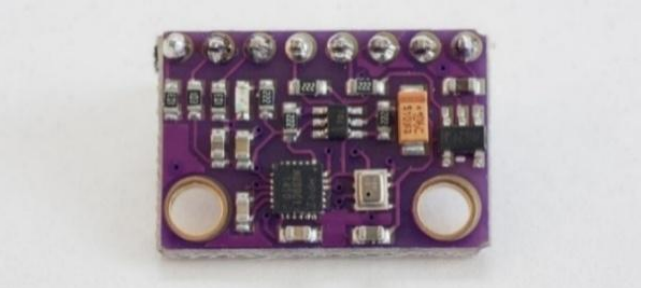
Los sensores de vibración para los motores son muy comunes ya que estos se encuentran en una variedad de diferentes campos entre las fuentes eléctricas y mecánicas. Y si bien es cierto los sensores de vibración se centran más en las fuentes mecánicas, ya que ayuda a detectar algunas de las fuentes eléctricas y prevé futuras fallas (Manufactura Latam, 2024).

Es importante saber que las máquinas vibran, ya que un cambio mínimo en sus patrones de vibración puede mostrar una variedad de posibles problemas. Medir los cambios en los parámetros de vibración ayuda a los equipos a identificar desajustes, desequilibrios, holgura, desalineación o un sin número de diferentes fallas de los rodamientos en los equipos y prever de que se produzca alguna más. La vibración anormal o excesiva ocasiona un gran problema a que los componentes sufran y afectar vida útil del equipo. Los sensores de vibración para motores se instalan en las maquinarias para monitorear los cambios en diferentes frecuencias (López, 2024).

A su vez pueden recopilar información de diferentes fuentes ya que los datos recopilados se analizan para identificar fallas y determinar la gravedad del equipo, por tanto, la misión de los acelerómetros radica en la medida de aceleraciones gravitacionales estáticas, lo que les permite la medida de la desviación del ángulo al que el objeto que se mide se aparta de la vertical, siendo posible también realizar otro tipo de medida para que el instrumento tome medidas en relación con aceleraciones dinámicas y modificaciones en los comportamientos de un objeto bajo golpes, desplazamientos, impactos o vibraciones sustanciales; es decir, vibraciones de baja amplitud y frecuencia, las que pueden alcanzar a

decenas de Hz, como se muestra en la figura 15 se observa el acelerómetro (Transfer Multisort Elektronik, 2025).

Figura 15
Acelerómetro



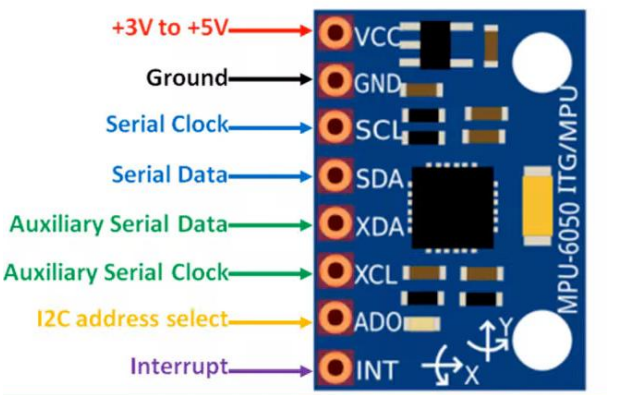
Nota. Acelerómetro utilizado para censar las vibraciones (Transfer Multisort Elektronik, 2025).

4.11 PINES DEL ACELEROMETRO

El módulo MPU6050 dispone de pines esenciales para su funcionamiento y comunicación mediante I²C. Los pines VCC y GND suministran alimentación, mientras que SCL y SDA gestionan el reloj y los datos, como se muestra en la siguiente figura 16.

- PIN Función
- VCC Alimentación (3.3V o 5V)
- GND Tierra
- SCL Línea de reloj I²C
- SDA Línea de datos I²C
- XDA Salida de datos auxiliar I²C (opcional)
- XCL Reloj auxiliar I²C (opcional)
- AD0 Selección de dirección I²C (0 = 0x68, 1 = 0x69)
- INT Salida de interrupción (opcional, para eventos)

Figura 16
Pines del acelerómetro



Nota. Acelerómetro pines que se utilizaran. (Electrónica, 2022).

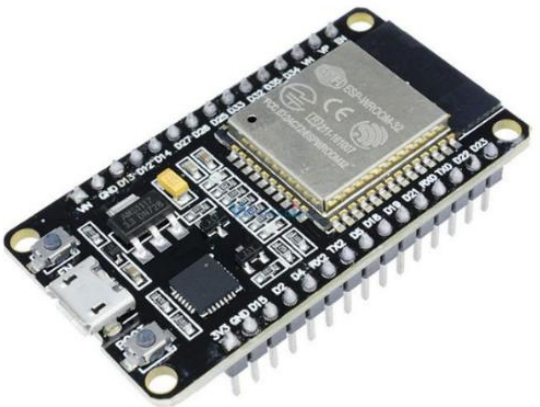
4.12 Sistemas embebidos (ESP32)

Los sistemas embebidos están desarrollados para integrarse en un sistema de mayor tamaño. Eso quiere decir, en lugar de ser un sistema independiente como una computadora tradicional, estos sistemas están destinados a ejecutar una cantidad de tareas exactas de un sistema más amplio. Su naturaleza de estar "embebidos" en la forma que operan, formando parte integral de la función global de ese dispositivo o sistema informático (Universidad Fidélitas, 2024).

4.12.1 Tarjeta ESP32

Poder describir el ESP32 como un microcontrolador que integra tecnologías Wifi y Bluetooth con las que permitirá conectarlo a internet o bien conectarlo con otros dispositivos. Se describe la principal funcionalidad y características de este, pero también los principales entornos de desarrollo y los lenguajes de programación que servirán para conseguir programar el dispositivo, se observa en la figura 17 el módulo de comunicación ESP32 *(Ortiz & Valencia, 2025)*.

Figura 17
Tarjeta ESP32



*Nota***.** Dispositivo electrónico ESP32 para la comunicación (Novatronic Ecuador, 2025).

4.12.2 Pines del ESP32

- **GPIO, ADC y DAC**

  El ESP32 cuenta con hasta 34 pines GPIO disponibles en placas de desarrollo. Ofrece además 18 canales ADC de alta resolución (12 bits) y 2 canales DAC de 8 bits para conversión de señal digital-analógica.

- **Sensores capacitivos (Touch Inputs)**

  Dispone de 10 pines táctiles capacitivos que permiten detectar toques, sirviendo también como fuente para despertar el ESP32 desde modo de suspensión profundo (deep sleep).

- **Interfaces de comunicación**

  El ESP32 integra múltiples protocolos de comunicación: 3 UART, 2 I²C, 4 SPI, 2 I²S, y un controlador SDIO/SD, lo que facilita la conexión con una amplia variedad de periféricos.

- **PWM y audio**

  Ofrece 16 canales PWM para control flexible de motores o iluminación. También incluye interfaces I²S para audio digital.

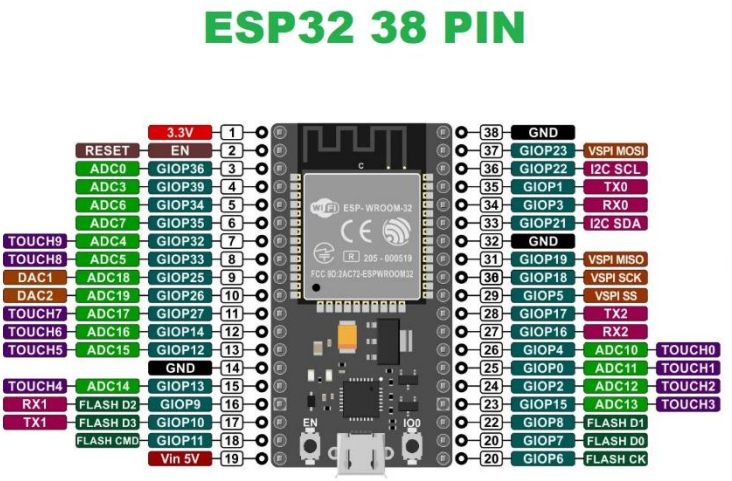
- **Pines restringidos (strapping y flash)**

  Algunos pines, como GPIO6 a GPIO11, están vinculados internamente a la memoria flash y no deben usarse para aplicaciones generales. Otros, como GPIO0, GPIO2, GPIO12, entre otros, son pines de arranque (strapping) y afectan el modo de inicio del dispositivo.

- **Uso de ADC y Wi-Fi**

  Se recomienda utilizar canales ADC1 cuando el Wi-Fi está activo, ya que los canales ADC2 pueden presentar conflictos o funcionalidad limitada en ese modo, como se muestra en la figura 18 (Boyko, 2024)

Figura 18
Pines del ESP 32

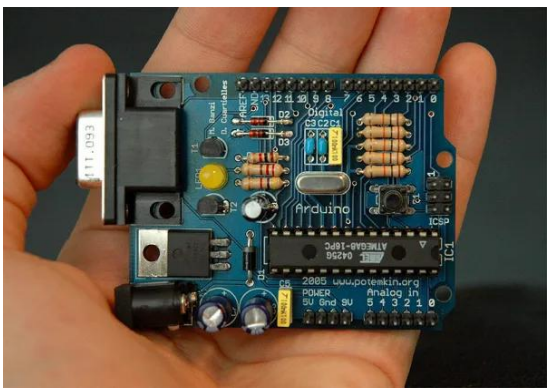


Nota. Dispositivo electrónico ESP32 con sus Pines *(Megatrónica, 2025).*

4.13 Arduino

Arduino es un componente electrónico de código abierto, sus objetivos son contar con software y hardware fáciles de usar. Lo que permite el dispositivo es una infinidad de tipos de microordenadores de una sola placa, que pueden tener una variedad de usos según lo que requiera la persona que lo programe, a su vez permite desarrollar elementos autónomos para conectarse a otros dispositivos o interactuar con otros programas del hardware como con el software, en la figura 19, se observa el dispositivo de comunicación ARDUINO (arduino, 2018).

Figura 19
Dispositivo Arduino



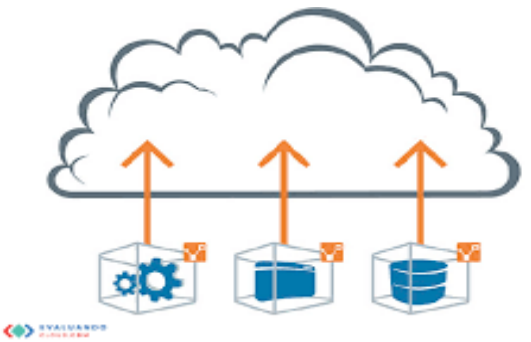
Nota. Arduino que hará la comunicación entre los dispositivos (Aprendiendo Arduino, 2019).

4.14 Transmisión de datos a la nube

La transmisión de datos a la nube habla de que su función es definir, implementar y supervisar un marco que mejora el rendimiento de los sistemas y los servicios de la nube. Dado los parámetros de la nube se expanden rápidamente más de lo que podrían manejar las personas de forma manual, la automatización hace posible y capaz un control más eficiente (Red Hat, 2022).

La nube brinda un proceso más organizado hacia la automatización de las tareas o actividades que se requieren en una gestión de la nube en flujos de trabajo más complejos. Al separar las diversas tareas de automatización de los distintos equipos y dominios en flujos de trabajo completos más eficientes, en la figura 20 se muestra la transmisión de datos a la nube (Kurduban, 2024).

Figura 20
Trasmisión de datos nube



Nota. Serie de datos ingresando a la nube (Evaluando Software, 2025).

4.14.1 Arduino IoT Cloud

Es una plataforma end-to-end que permite a los desarrolladores desplegar y monitorizar soluciones IoT sin necesidad de infraestructuras complejas. Facilita la gestión de dispositivos, sincronización de datos en tiempo real, creación de dashborards y automatización avanzada mediante OTA y programación desde la nube. Proporciona una compresión profunda del ecosistema Arduino Cloud, incluyendo la configuración de proyectos desde cero, el uso de APIs y SDKs, y la implementación de comunicaciones avanzadas como LoRaWAN y OTA, lo que lo convierte en un recurso valioso para soluciones escalables y prácticas en áreas como smart-home, agricultura inteligente y monitoreo ambiental, como se muestra en la figura 21 (Kurniawan, 2021)

Figura 21
IoT Cloud

Nota. Frecuencias ingresando en la nube *(how2electronics, 2025)*.

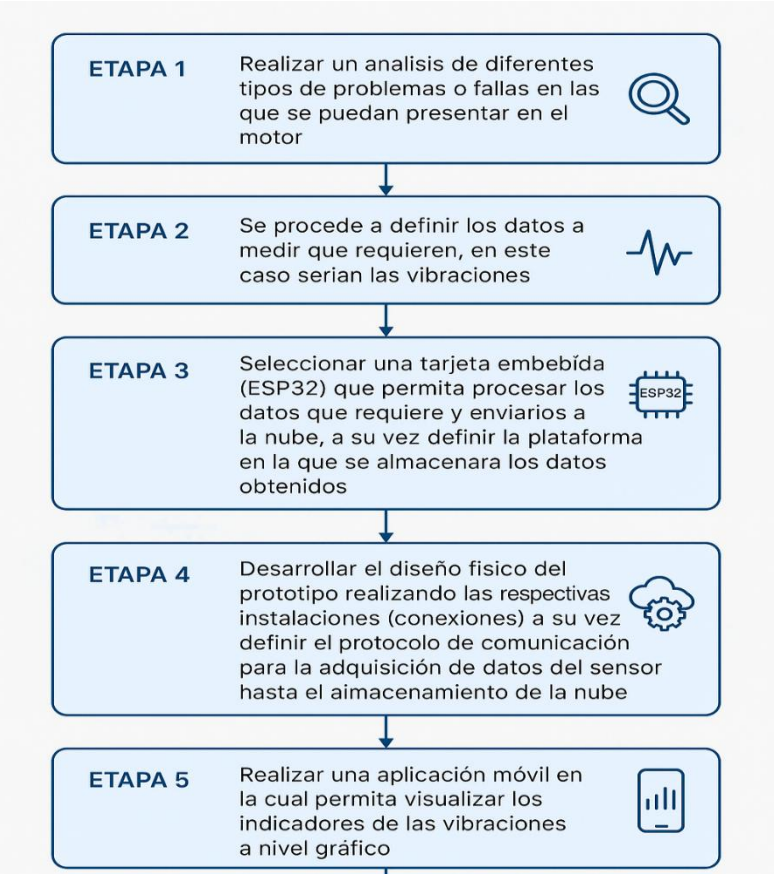
V MARCO METODOLÓGICO

El prototipo desarrolla un sistema para la detección y análisis de vibraciones en motores, combinando una tarjeta embebida con un acelerómetro para analizar datos en tiempo real, añadiendo la transformada de Fourier para identificar desequilibrios en las frecuencias facilitando un diagnóstico de manera clara y sencilla.

5.1 Etapas de la metodología

Se muestra el diagrama estructural que se va a implementar en el proyecto, en la cual se realiza etapa a etapa el análisis e implementación del prototipo, la etapa que más resalta es la 3 ya que parte principal en el proyecto porque es la comunicación del todo el proyecto, en la figura 22 se muestra.

Figura 22
Diagrama estructural del proyecto

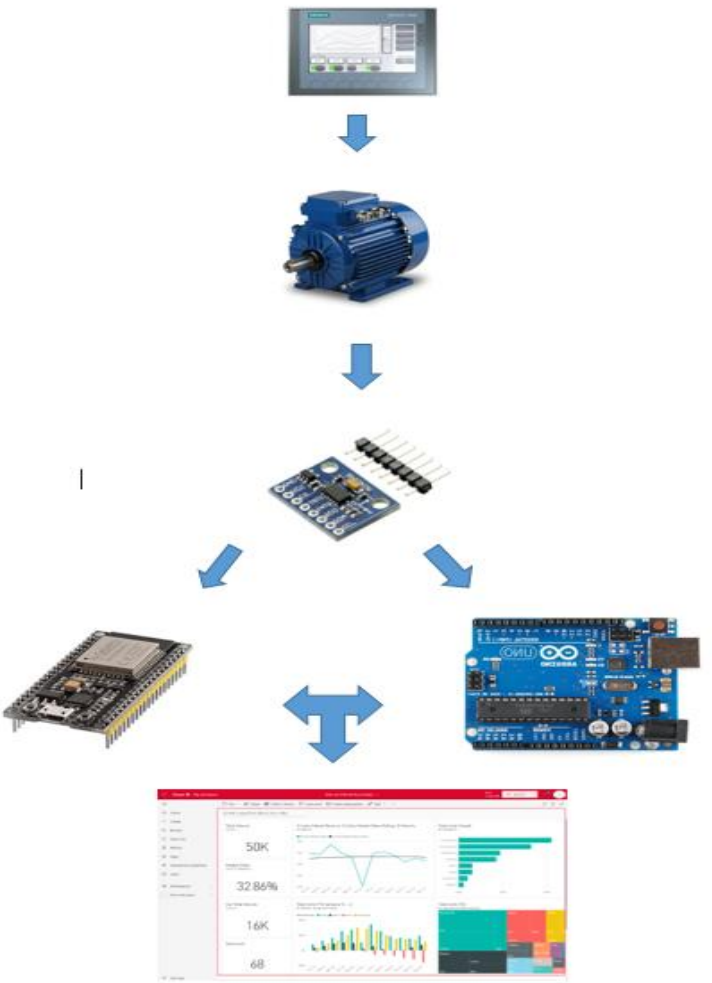


Nota. Estas son las etapas del proceso del proyecto.

5.2 Prototipo esquemático

Se muestra el prototipo esquemático en la cual se realizará la implementación del proyecto de manera física paso a paso como se observa en la figura 23.

Figura 23
Diagrama esquemático

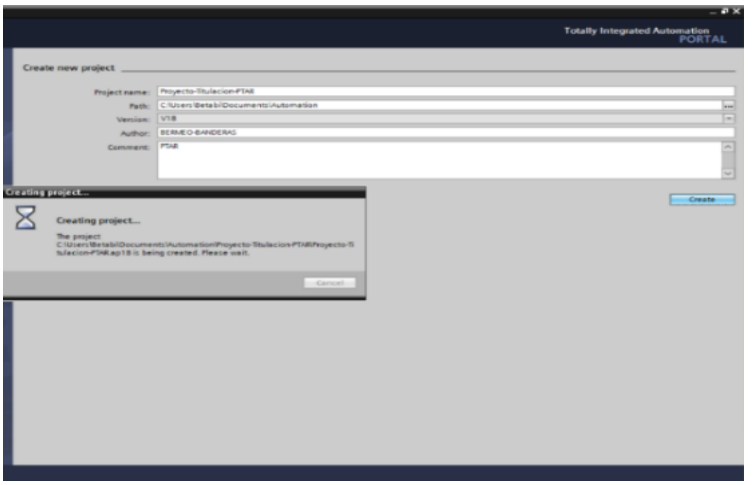


Nota. Estos son los dispositivos que se van a usar en su respectivo orden.

5.3 Creación de proyecto

Al acceder a la pantalla principal de TIA Portal, se observará un botón identificado como crear nuevo proyecto. Es así como, se desplegará una ventana en la que será necesario introducir el nombre que se asignará al proyecto, como se ilustra en la figura 24.

Figura 24
Creación nuevo proyecto

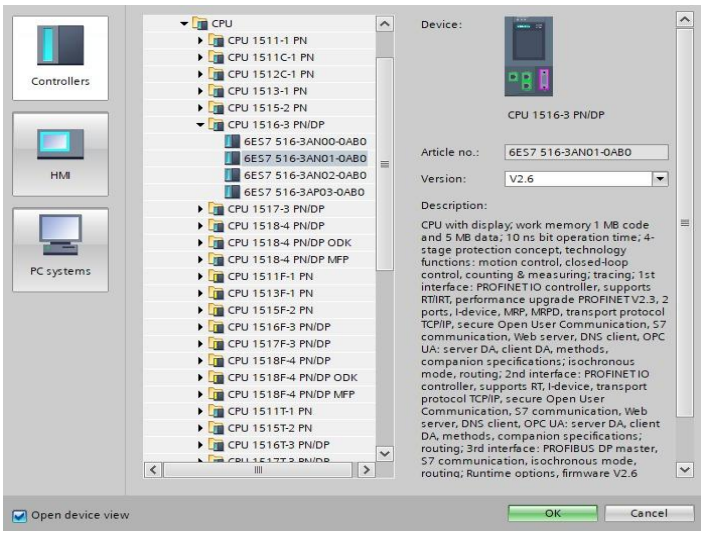


Nota. Se realizo la creación del proyecto con su respectivo nombre.

5.4 Selección del tipo de control

Este paso consiste en definir el tipo de PLC a emplear. Para este proyecto se usó el PLC S7-1500 como dispositivo encargado del control, mientras que para la supervisión y visualización del proceso se utilizó la HMI KTP700 Basic, tal como se aprecia en la figura 25.

Figura 25
Selección del plc s7-1500



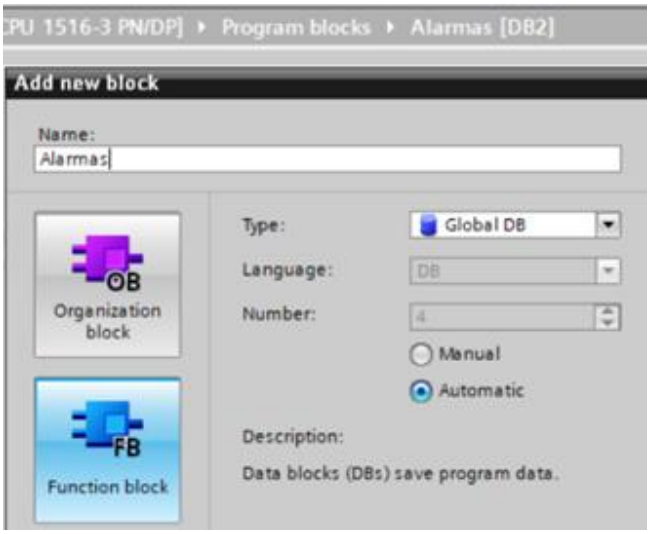
Nota. En este tipo de control se seleccionó el PLC S7-1500.

5.5 Configuración de bloque de datos (DB_Alarmas)

Para un DB llamado Alarmas, lo habitual es crear bits BOOL para cada alarma y, si hace falta, tiempos/contadores o estructuras por máquina o área. Al ser Global DB, cualquier OB/FB/FC puede leer y escribir esas señales, como se muestra en la figura 26.

Figura 26

Program blocks

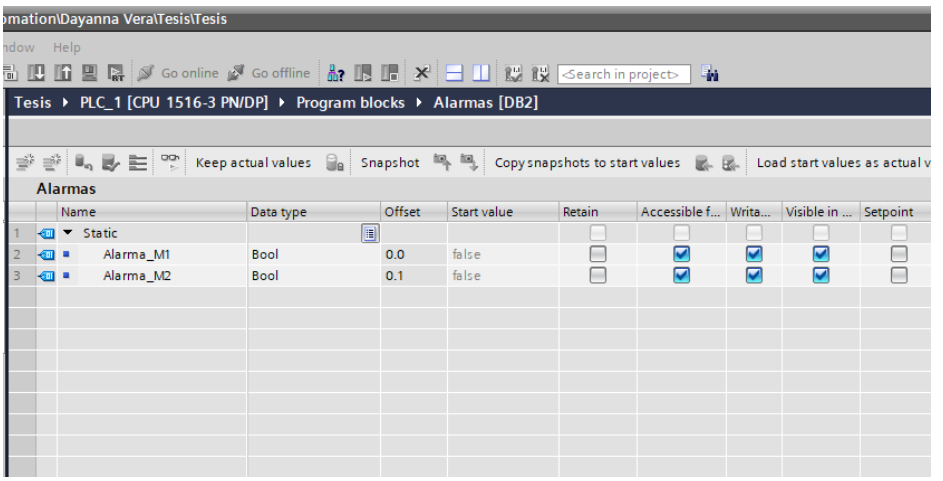


Nota. Creación de la data block.

Se crea un bloque de datos Data Block desde la sección Program Blocks, asignándole el nombre DB_Alarmas. Dentro de este bloque se definen las variables alarma_M1 y alarma_M2, ambas de tipo Bool, correspondientes a los estados de alarma de cada motor, como se muestra en la figura 27.

Figura 27

DB_Alarmas

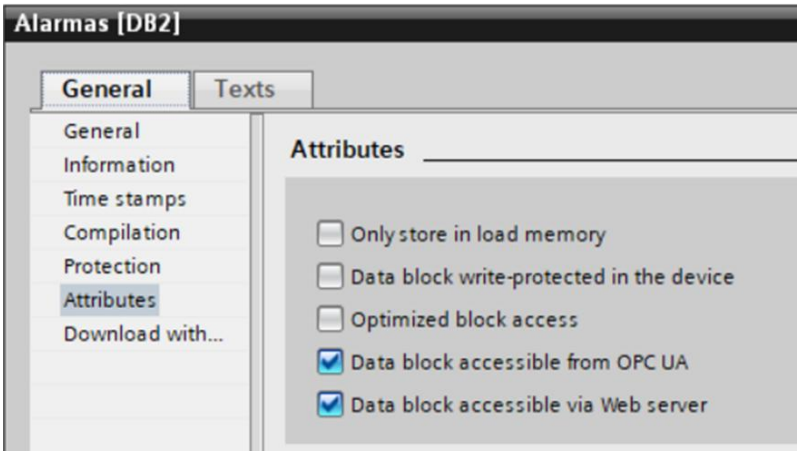


Nota. Asignación de alarma en el data block.

Luego de haber creado el bloque de datos, se configuró sus propiedades desactivando la optimización de acceso del bloque, esto para poder crear las variables y compilar sin problemas, como se muestra en la figura 28.

Figura 28

Desactivación optimización



Nota. configuró sus propiedades desactivando la optimización.

5.6 Configuración PLC TAG

En la programación del PLC se utilizan diferentes variables y señales que permite gestionar el arranque, paro y supervisión de los motores desde entradas físicas y la interfaz HMI. Entre ellas se incluyen los bits de reloj, como el Clock_Byte y las señales Clock_10Hz a Clock_0.5Hz, que generan pulsos cíclicos a distintas frecuencias para sincronización de procesos, como se muestra en la figura 29.

- Clock_Byte %MB100: Es un byte completo que cambia de estado de forma cíclica en función del reloj interno del PLC.
- Clock_10Hz a Clock_0.5Hz %M100.0 a %M100.7: Son bits que oscilan automáticamente con frecuencias fijas generadas por el PLC 10 Hz, 5 Hz, 2.5 Hz, etc.
- Marcha (%I0.0): Entrada digital para arrancar el proceso o motor.
- Paro %I0.1: Entrada digital para detener el proceso o motor.
- Paro_HMI Memoria global: Señal de paro activada desde la interfaz HMI pantalla táctil.
- Marcha_M2 / Paro_M2: Entradas para controlar un segundo motor o sección del sistema.
- Marcha_HMI_M2 / Paro_HMI_M2: Variables globales controladas desde la HMI para el segundo motor.

- Marcha %I0.0: Entrada digital para arrancar el proceso o motor.
- Paro %I0.1: Entrada digital para detener el proceso o motor.
- Paro_HMI Memoria global: Señal de paro activada desde la interfaz HMI pantalla táctil).
- Marcha_M2 / Paro_M2: Entradas para controlar un segundo motor o sección del sistema.
- Marcha_HMI_M2 / Paro_HMI_M2: Variables globales controladas desde la HMI para el segundo motor.

Figura 29
PLC TAG

PLC tags

	Name	Tag table	Data type	Address	Retain	Acces...	Writa...	Visibl...
4	Clock_Byte	Default tag table	Byte	%MB100	☐	☑	☑	☑
5	Clock_10Hz	Default tag table	Bool	%M100.0	☐	☑	☑	☑
6	Clock_5Hz	Default tag table	Bool	%M100.1	☐	☑	☑	☑
7	Clock_2.5Hz	Default tag table	Bool	%M100.2	☐	☑	☑	☑
8	Clock_2Hz	Default tag table	Bool	%M100.3	☐	☑	☑	☑
9	Clock_1.25Hz	Default tag table	Bool	%M100.4	☐	☑	☑	☑
10	Clock_1Hz	Default tag table	Bool	%M100.5	☐	☑	☑	☑
11	Clock_0.625Hz	Default tag table	Bool	%M100.6	☐	☑	☑	☑
12	Clock_0.5Hz	Default tag table	Bool	%M100.7	☐	☑	☑	☑
13	Marcha	Entradas	Bool	%I0.0	☐	☑	☑	☑
14	Paro	Entradas	Bool	%I0.1	☐	☑	☑	☑
15	Paro_HMI	Memorias globales	Bool	%M0.2	☐	☑	☑	☑
16	Motor_2	Salidas	Bool	%Q0.1	☐	☑	☑	☑
17	Marcha_M2	Entradas	Bool	%I0.2	☐	☑	☑	☑
18	Marcha_HMI_M2	Memorias globales	Bool	%M0.3	☐	☑	☑	☑
19	Paro_M2	Entradas	Bool	%I0.3	☐	☑	☑	☑
20	Paro_HMI_M2	Memorias globales	Bool	%M0.4	☐	☑	☑	☑
21	Led_M2	Salidas	Bool	%Q0.2	☐	☑	☑	☑
22	Led_M1	Salidas	Bool	%Q0.3	☐	☑	☑	☑
23	Activar_M2	Default tag table	Bool	%M0.5	☐	☑	☑	☑
24	Motor_animado	Default tag table	Word	%MW2	☐	☑	☑	☑
25	Motor_animado_M2	Memorias globales	Int	%MW4	☐	☑	☑	☑
26	<Add new>				☐	☑	☑	☑

Nota. Asignación de los Tag del PLC.

Para las entradas físicas fueron creadas las variables que simplemente envían la señal a un bloque set y reset, activando el bloque llamado Activar_M1 y Activar_M2 haciendo referencia a la activación del primer motor. También están las memorias globales que son fundamental para el uso interactivo con el HMI, como se muestra en la figura 30.

Figura 30
Entradas físicas

Entradas

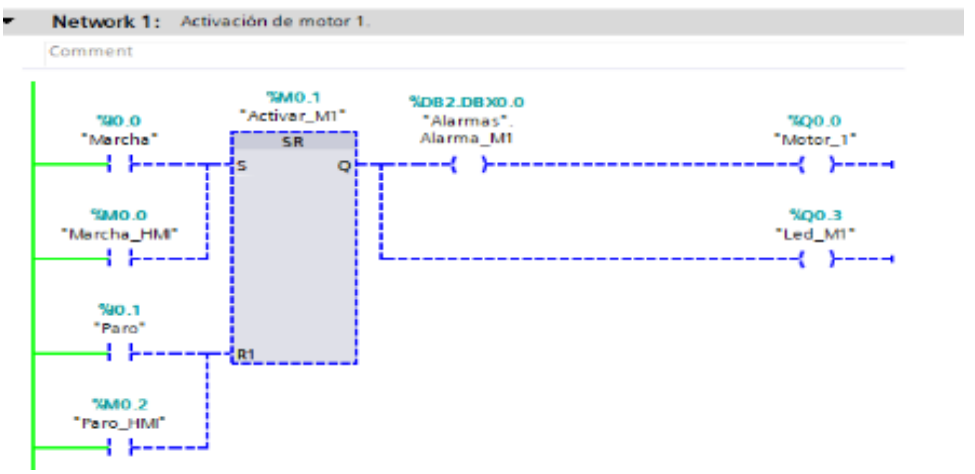
	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...
1	Marcha	Bool	%I0.0	☐	☑	☑	☑
2	Paro	Bool	%I0.1	☐	☑	☑	☑
3	Marcha_M2	Bool	%I0.2	☐	☑	☑	☑
4	Paro_M2	Bool	%I0.3	☐	☑	☑	☑
5	<Add new>			☐	☑	☑	☑

Nota. Asignación de las entradas físicas.

5.7 Programación de control del Motor 1 y 2

El motor 1 se controla por un bloque SR (set reset) que actúa como encendido y apagado, además, este se enciende cuando la entrada de set se activa ya sea por el botón físico de marcha I0.0 o la virtual marcha HMI M0.0, mientras que para el apagado se enciende la entrada reset la cual activa el botón Paro I0.1 o Paro HMI %M0.2, de manera automática se activa la señal de Alarma M1 %DB2.DBX0.0. La salida del bloque SR controla el encendido del motor %Q0.0 y la luz indicadora Led HMI %Q0.3 que muestra su estado en funcionamiento, como se muestra en la figura 31.

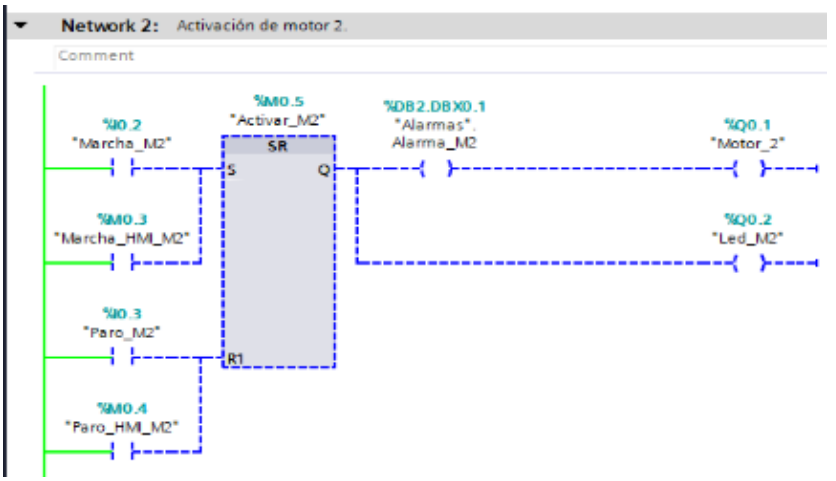
Figura 31
Activación del Motor 1



Nota. Diagrama de bloque para la activación del motor 1.

El motor 2 es el mismo funcionamiento del motor 1 utilizando el bloque SR %M0.5, el motor se enciende cuando se activa la entrada SET la cual controla el botón Marcha M2 %I0.2 o el Marcha HMI M2 %M0.2, el motor se detiene si se activa la entrada Reset ya sea por el botón Paro M2 o Paro HMI M2, si hay una falla la señal Alarma M2 %DB2.DBX0.1 se activa. Cuando el bloque SR esta activado el motor %Q0.1 y el Led indicador Led M2 %Q0.2 se activan para mostrar el funcionamiento., como se muestra en la figura 32.

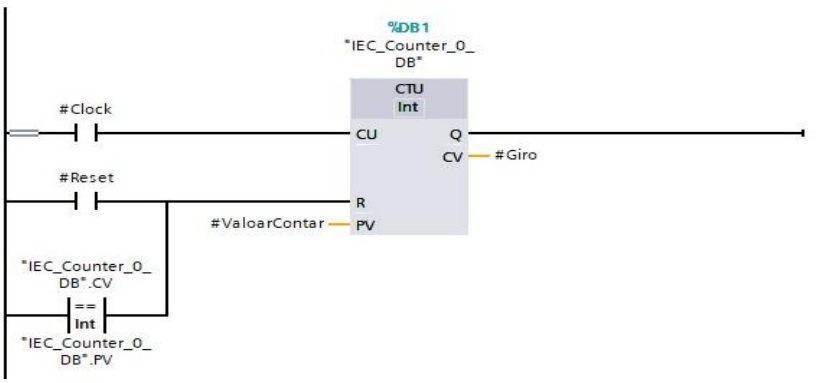
Figura 32
Activación de Motor 2



Nota. Diagrama de bloque para la activación del motor 2.

Esta animación giratoria se controla con un contador ascendente CTU que crea un bucle cíclico, la entrada CU activa con pulso el CLOCK interno del sistema haciendo que el contador eleve el valor CV a un ritmo constante ya que cuando el valor del contador llegue al valor preestablecido PV la salida Q se activa. La salida conectada a la entrada Reset de mismo contador lo que reiniciara automáticamente a 0., como se muestra en la figura 33.

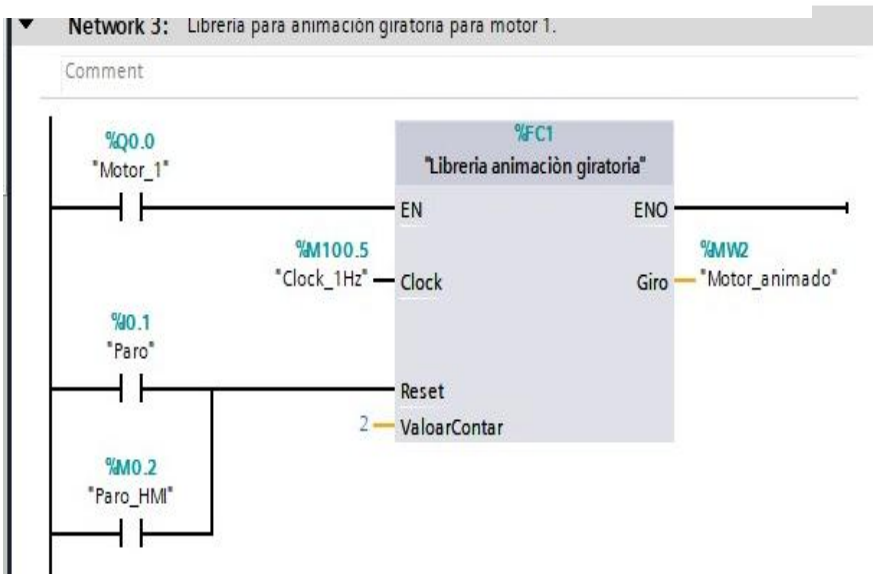
Figura 33
Animación giratoria



Nota. Bloque que se va a usar para la animación giratoria.

En este bloque de función FC1 Librería animación giratoria cuya entrada EN está controlada por el contacto %Q0.0 "Motor_1, la librería sólo opera cuando el motor está encendido, la entrada Clock recibe pulsos dé %M100.5 Clock_1Hz para avanzar el conteo interno; las señales de paro %I0.1 Paro y %M0.2 Paro_HMI están puestas en paralelo para generar el Reset si cualquiera se activa; el parámetro ValorContar está fijado a 2 el comportamiento se dispara tras 2 pulsos del clock y cuando se cumple la condición el bloque pone su salida Giro que se mapea a %MW2 Motor_animado y ENO refleja el estado de ejecución con lo cual se produce la señal que hace la animación giratoria del motor., como se muestra en la figura 34.

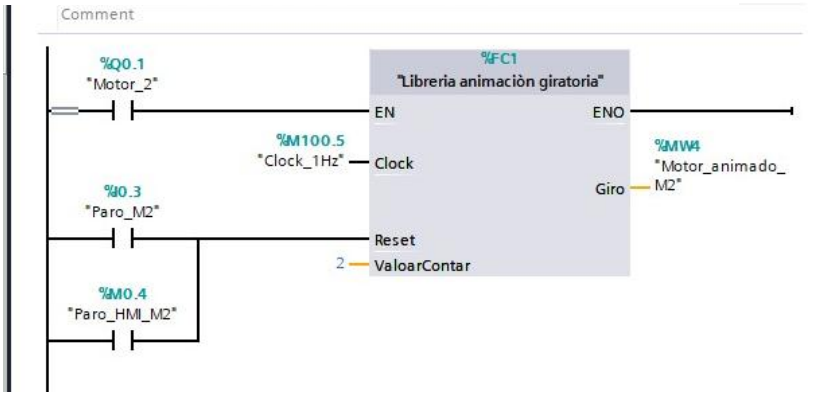
Figura 34
Animación giratoria motor 1



Nota. Bloque que se va a usar para la animación giratoria del motor 1.

Se muestra en este el bloque de función “Librería animación giratoria” FC1 cuando el contacto %Q0.1 Motor_2 está verdadero; el pin EN del bloque queda energizado y el pin Clock recibe el pulso de %M100.5 Clock_1Hz, que hace avanzar los pasos de la animación. En la rama inferior se cablean las paradas %I0.3 Paro_M2 y %M0.4 Paro_HMI_M2 de forma que, si cualquiera se activa, entra señal al pin Reset del bloque y la animación se reinicia. Además, el parámetro Valor a Contar del FC se fija con la constante 2 número de estados/pasos por ciclo. La salida Giro del bloque se escribe en %MW4 Motor_animado_M2 para mostrar el giro del motor en la HMI, mientras que ENO simplemente indica la ejecución correcta hacia el riel derecho, en la siguiente figura 35 se mostrará.

Figura 35
Animación giratoria del Motor 2

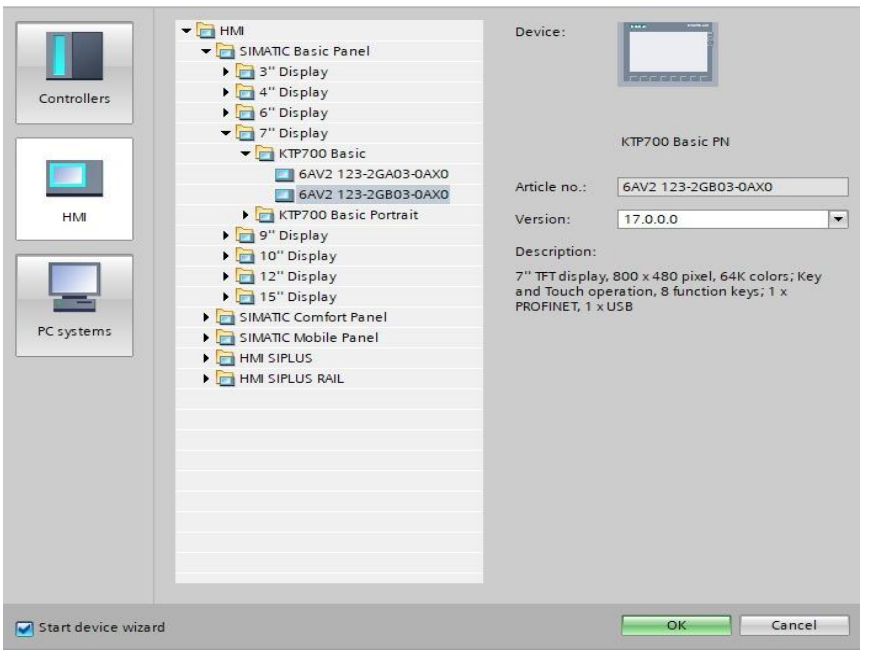


Nota. Bloque que se va a usar para la animación giratoria del motor 2.

5.8 Pantalla de presentación

A continuación, se selecciona el panel HMI KTP700 Basic, en su versión 15.1.0.0, con el fin de llevar a cabo la visualización del proceso, tal como se muestra en la figura 36.

Figura 36
Selección del HMI KTP700

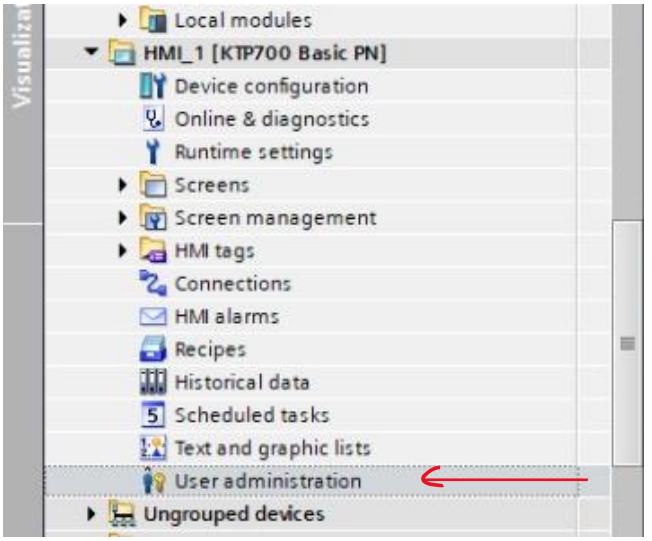


Nota. Creación del HMI con su respectiva versión.

En esta parte de la programación se navega en el árbol del proyecto hasta Security Settings y luego doble clic en Users administration, aquí, se puede crear nuevos usuarios, asignarles contraseñas y definir sus roles y permisos para controlar qué partes del proyecto pueden ver o modificar, como se muestra en la figura 37.

Figura 37

Users administration

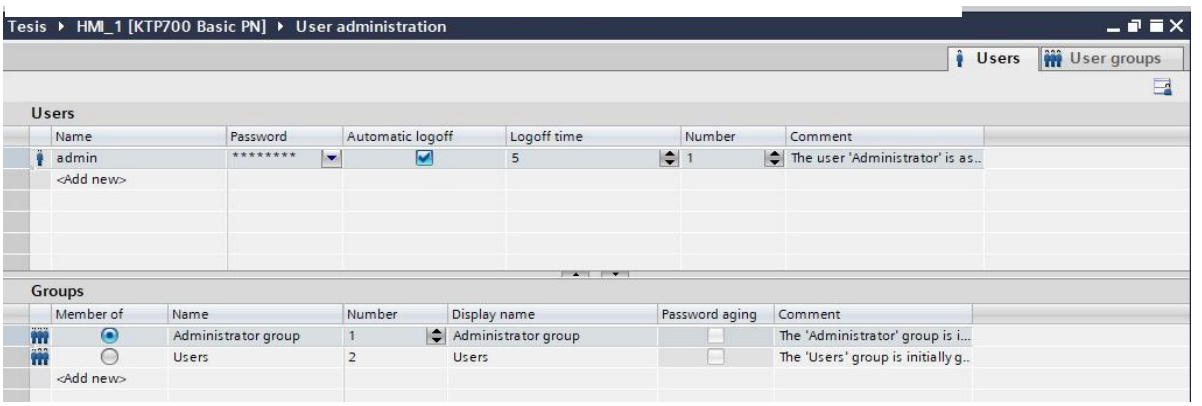


Nota. Se lo otorgara un usuario y contraseña al HMI.

En esta ventana de administración de usuarios en TIA Portal se observa el usuario admin, con contraseña, cierre de sesión automático en 5 min y asignado al grupo Administrator group; además, existen dos grupos creados: Administrator group y Users, a los cuales se pueden añadir más usuarios según los permisos que se necesiten en el HMI, como se muestra en la figura 38.

Figura 38

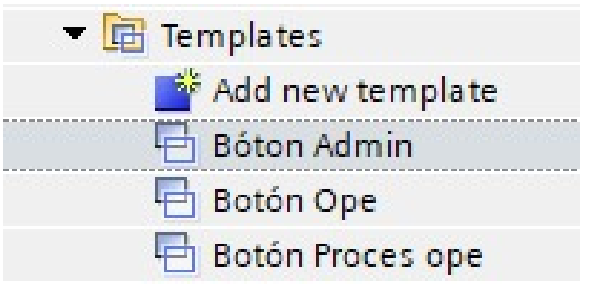
Administración de usuarios



Nota. Usuarios que se utilizaran en el HMI.

También se crearon dos diferentes Templates que serán llamados en los HMI el Boton Admin, Boton Ope, Boton Proces Ope, en el Templates Boton Admin se incluyó el ingreso de usuario, por esta razón se agregó en la pantalla de INICIO para que solo las personas autorizadas puedan manipular la interfaz, figura 39.

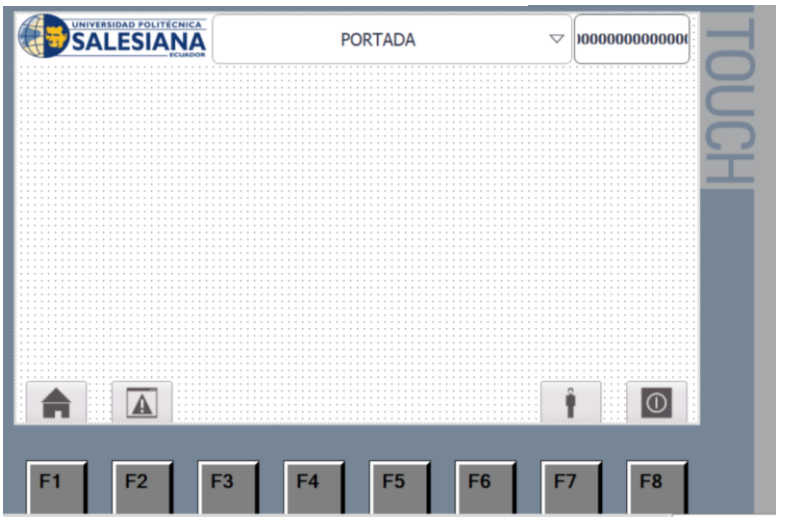
Figura 39
Templates



Nota. Templates que se utilizaran en el HMI.

El Template-ACCESS cuenta con una barra en la parte superior que permite navegar entre los distintos Screens, además incluye una sección donde una vez ingresado se puede visualizar si el usuario tiene el rol de administrador o de operador, por último, contiene un botón para el ingreso de datos como usuario y contraseña destinado al acceso del usuario, como se muestra en la figura 40.

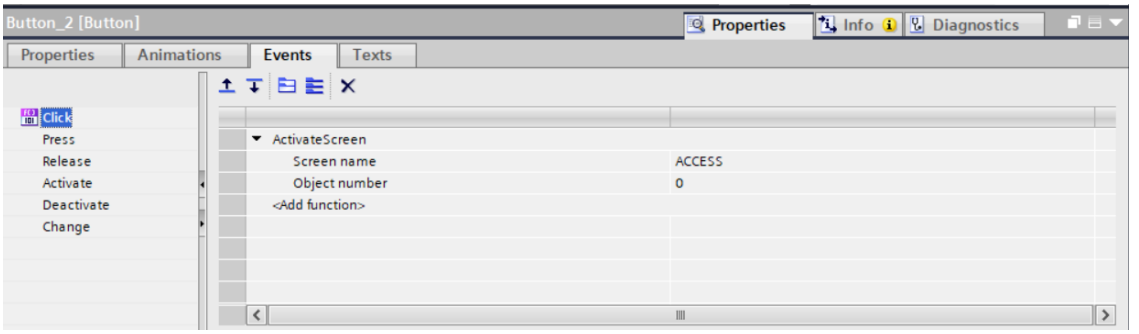
Figura 40
Template-ACCESS



Nota. Usuario como administrador.

Entonces, para poder colocarle una imagen a ese botón se le configuro sus propiedades, se le asigno un evento al darle clic a dicho botón, el cual es que se active o presente la pantalla que se le asigne, en esta ocasión es la pantalla ACCESS, como se muestra en figura 41.

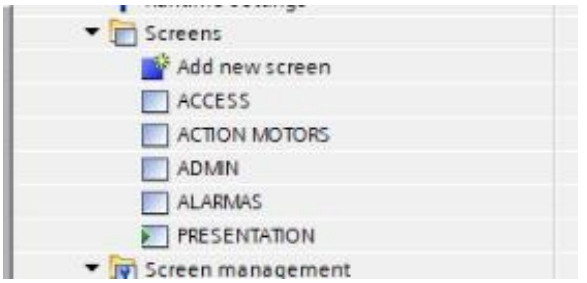
Figura 41
Actívate screen



Nota. Activación al botón siguiente para pasar a la pantalla de motores.

En la figura 42 se muestra la pantalla que se crean en el HMI, como se puede observar las que más se van a utilizar es la de presentación y acción motores.

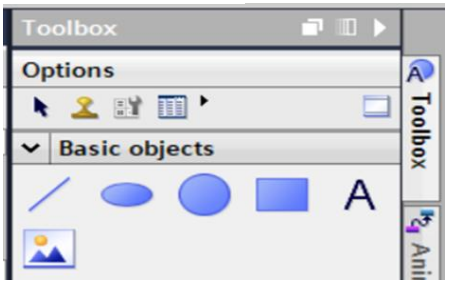
Figura 42
Pantallas HMI



Nota. Pantallas que se utilizaran en el HMI

Para colocar la caja de texto, lo primero que se hizo fue usar la pestaña "Volvox" que está hacia la derecha, como se muestra en la figura 43.

Figura 43
Toolbox



Nota. Modificar pantalla de presentación.

Así mismo, en la misma sección de “Basic objeto” se encuentra la opción de insertar figurar obtenidas de la web y finalmente, para el último recurso de la pantalla presentación se lo obtuve de la sección “Elementas”, como se mestra en la figura 44.

Figura 44
Elements

Nota. Modificar pantalla de presentación.

La pantalla HMI fue creada en TIA Portal añadiendo un panel KTP700 Basic PN y generando una nueva pantalla donde se diseñó un fondo con formas y colores, se insertaron imágenes como el logotipo de la Universidad, el motor, las tarjetas electrónicas y el ícono de Ío, además de cuadros de texto para mostrar el título del proyecto, el nombre del tutor y de los integrantes, configurando tamaños y estilos de letra para dar mayor énfasis; posteriormente se organizaron los elementos de manera clara y estética, se añadieron botones de navegación en la parte inferior para permitir el acceso a otras pantallas y finalmente se probó en el simulador antes de transferirlo al panel físico, como en la siguiente figura 45 se muestra.

Figura 45
Pantalla de presentación del HMI

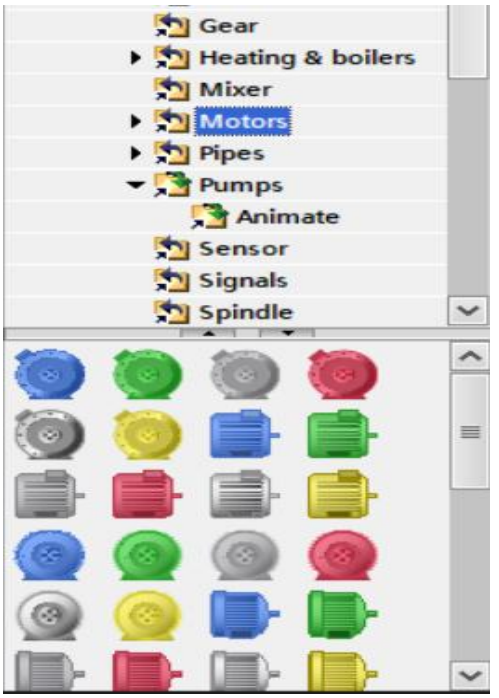
Nota. Pantalla HMI que se va a utilizar en el proceso.

5.9 Pantalla de proceso del HMI

Para la pantalla ACCION MOTORS se le asigno desde la pestaña tolos, sección Graphics el grupo WinCC graphic folder, luego en Equipment, después en Automation EMF para finalmente elegir Motors, en dicho grupo se eligió uno de color verde y otro de rojo, esto para realizar un efecto de encendido y apagado del motor, para ello se colocó el rojo encima del verde, como se muestra en la figura 46

Figura 46

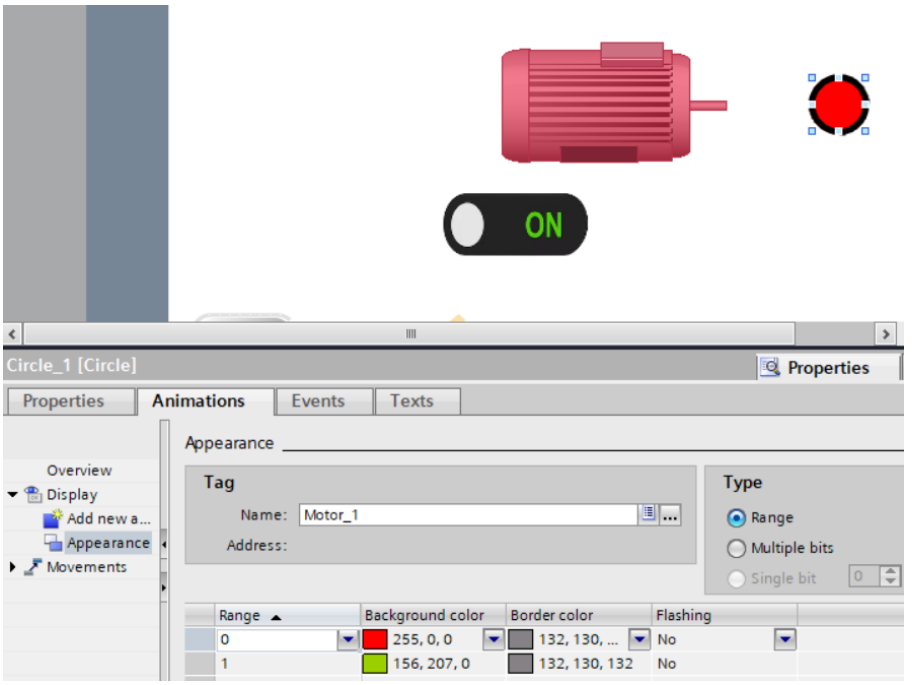
Creación del motor 1 y 2



Nota. Figuras que se van a utilizar en el HMI.

Luego se colocó un interruptor deslizante tipo ON/OFF (abajo a la izquierda) y se vinculó al bit de arranque Motor2_Start; a su lado se añadió una lámpara piloto circular (arriba a la derecha) asociada al estado Motor2_Status para indicar “Encendido/Apagado como se muestra en la figura 47.

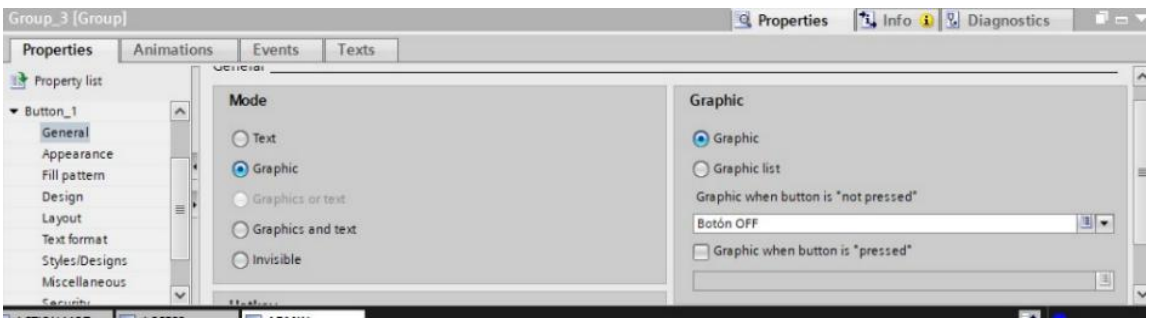
Figura 47
ON-OFF



Nota. Programación del on-off.

En esta pantalla HMI se configuró el control de dos motores: cada panel muestra la imagen del motor, una lámpara indicadora de estado y un botón gráfico ON/OFF. En las propiedades se definió el modo Graphic y la imagen para el estado apagado, mientras que en la pestaña de animaciones y eventos se vinculan los botones, pilotos y giro del motor con las variables del PLC. Finalmente, se agrupan los elementos para ordenarlos y se verifica su funcionamiento en runtime, como se muestra en la figura 48.

Figura 48
Control de los 2 motores

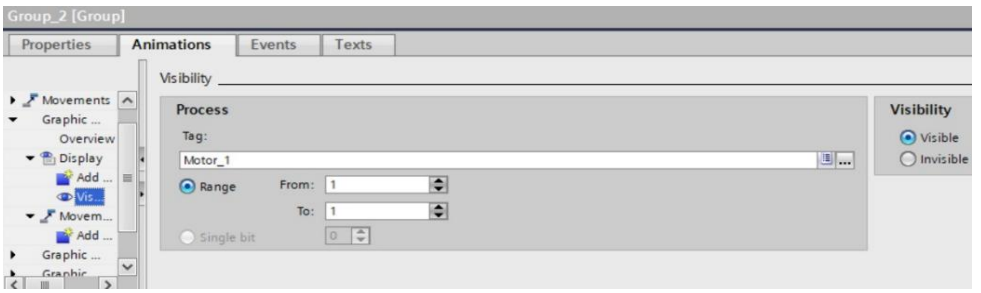


Nota. Pantalla HMI implementada ON-OFF.

En esta imagen corresponde a la programación de una pantalla HMI para el control y monitoreo de dos motores eléctricos, donde se configuran animaciones de visibilidad para mostrar su estado (ON/OFF) en tiempo real, como se muestra en la figura 49.

Figura 49

Activación del On-Off

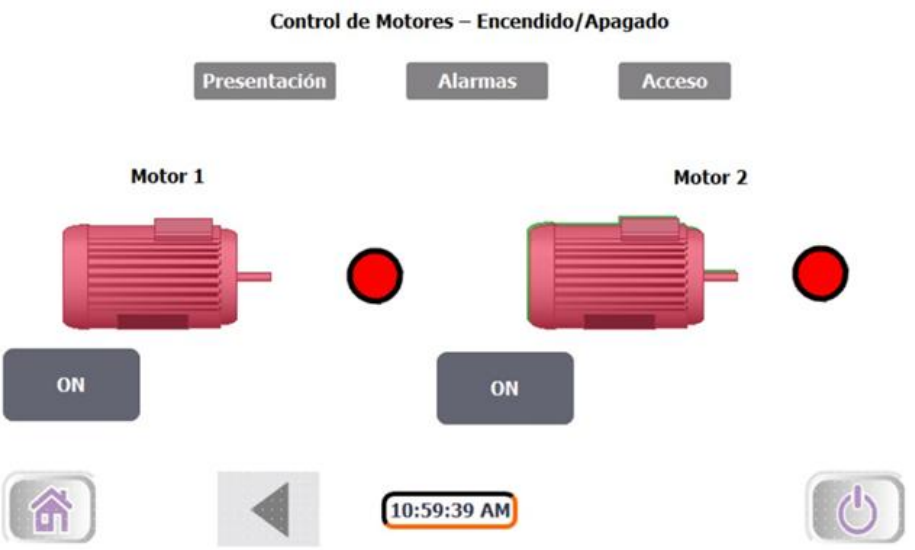


Nota. Pantalla donde se manipulará la activación de motores en el HMI.

Se diseñó una interfaz gráfica en el HMI que permite el control de Motores Encendido/Apagado y las pestañas Presentación, Alarmas y Acceso para cambiar de pantalla; en el área central están dos tarjetas, Motor 1 y Motor 2, cada una con la gráfica del motor, una lámpara circular y la leyenda Encendido / Apagado, además de un control grande con la etiqueta ON que funciona como interruptor, como se muestra en la figura 50.

Figura 50

Insertar graficas de los motores

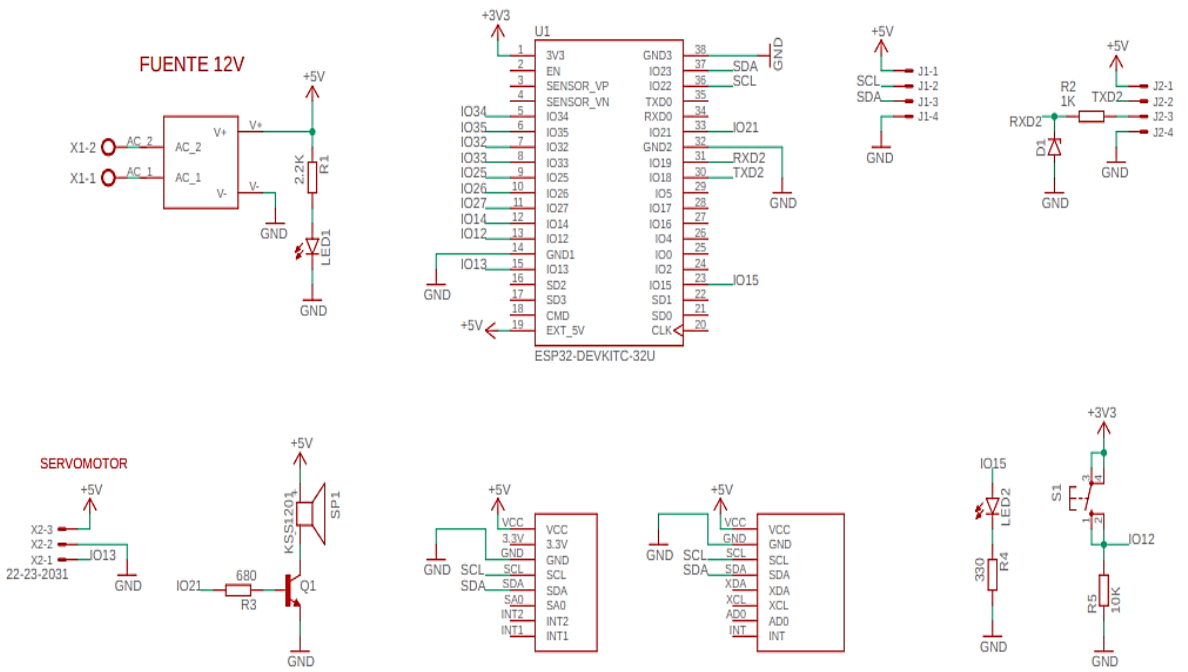


Nota. Pantalla donde se manipulará la activación de motores.

5.10 Diseño de placa PCB

Se diseña una placa donde se montan los componentes, como el ESP 32 para comunicación con el acelerómetro que también estaría montado con una fuente de 5 volteos, en la figura 51 se muestra el diseño de la placa.

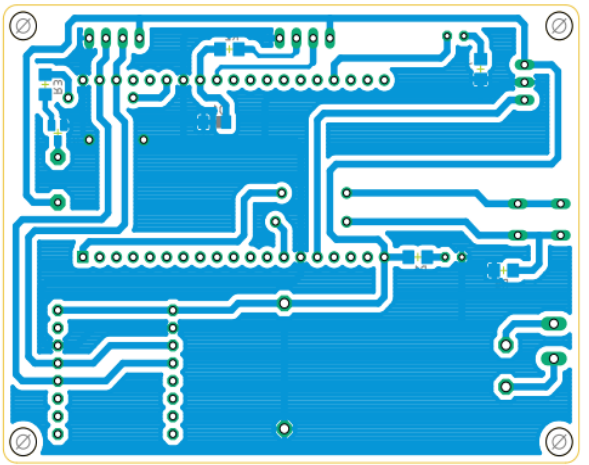
Figura 51
Elaboración de diagrama de conexiones



Nota. Placa donde se montaron el acelerómetro y ESP 32.

Diseño de la tarjeta electrónica, donde se lleva la simulación y configuración de los componentes, en ella se trazaron las pistas del circuito, asegurando una conexión óptima y minimizando el riesgo de interferencias, todo con el fin de obtener la funcionalidad, como se muestra en la figura 52.

Figura 52
Diseño de la tarjeta electrónica

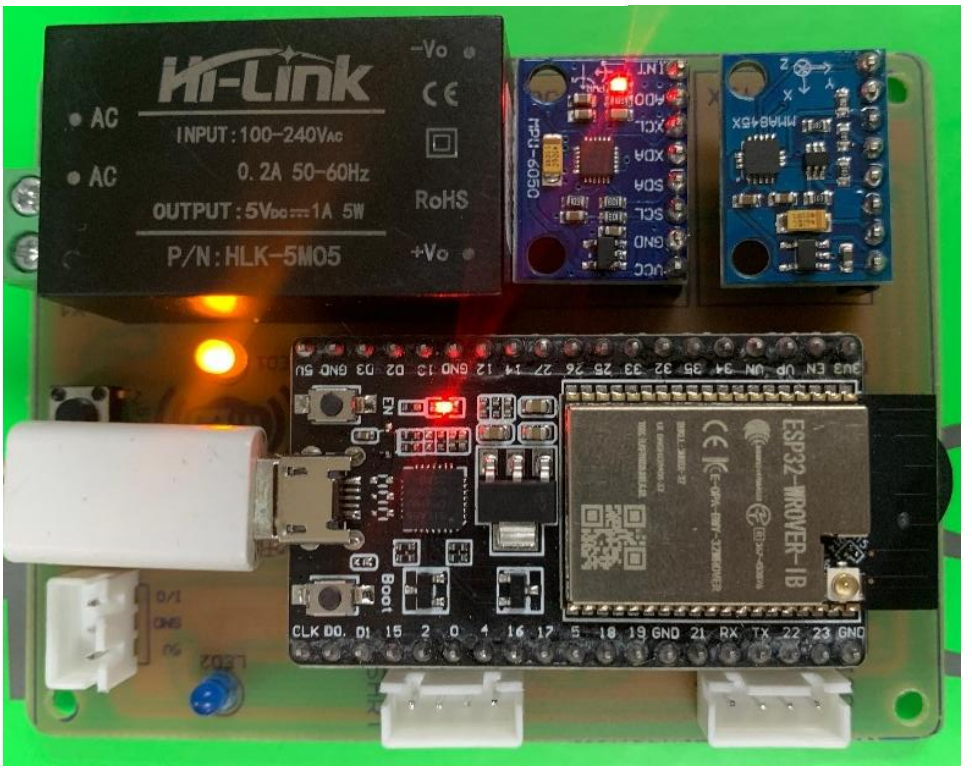


Nota. Diagramas de conexiones en Proteus.

5.11 Implementación de elementos electrónicos en la PCB

En la figura 53 se observa la implementación del prototipo utilizando cada elemento y dispositivo electrónico junto a su placa PCB, que utilizara con el motor.

Figura 53
Implementación



Nota. instalación de los componentes en la tarjeta.

5.12 Parámetros del variador para conexión del motor 1 y 2

Se procede a la conexión de los Motore 1 y Motor 2, con las entradas y las salidas digitales que se configuró en el PLC, para el funcionamiento de los motores, también se usó 2 variadores de frecuencia con los parámetros P0100=1, P0304=230, P0305=1.89, P0307=0.5 P0309=0, P0310=60, P0311=615, P1900=2 (El equipo debe estar conectado al motor antes de dar ok en este parámetro), estos datos son depende a la característica del motor, Tabla 1

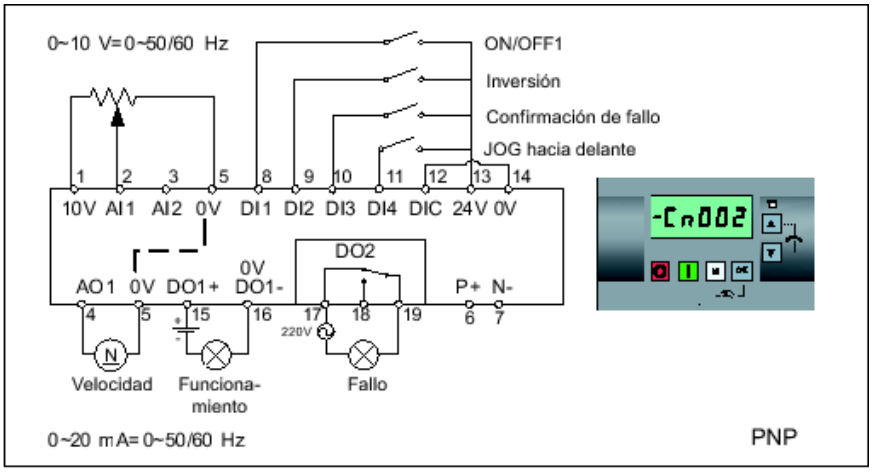
Tabla 1
Configuración en los parámetros

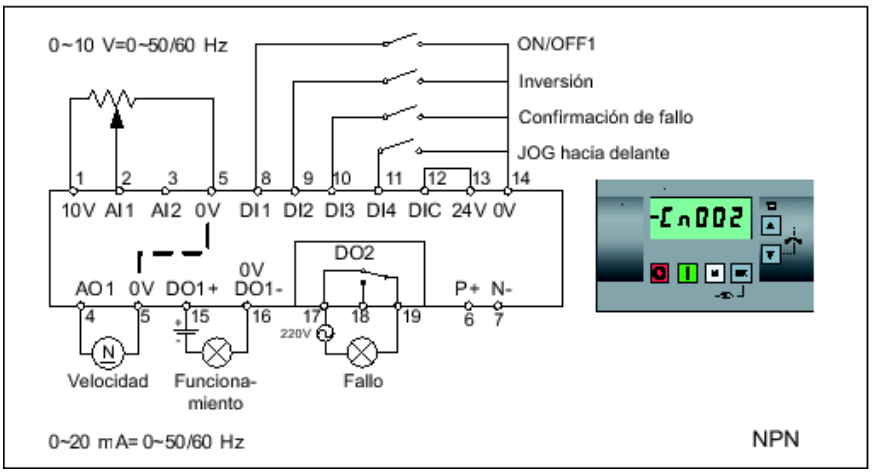
Parámetros	Nivel de acceso	Función
P0100=1	1	Selección de 50/60 Hz =0: Europa [kW], 50 Hz (valor predeterminado de fábrica) =1: Norteamérica [hp], 60 Hz =2: Norteamérica [kW], 60 Hz
P0304=230	1	Tensión nominal del motor [V] Tenga en cuenta que la entrada de los datos de la placa de características tiene que corresponder con el cableado del motor (en estrella/triángulo).
P0305=1.89	1	Corriente nominal del motor [A] Tenga en cuenta que la entrada de los datos de la placa de características tiene que corresponder con el cableado del motor (en estrella/triángulo).
P0307=0.5	1	Potencia nominal del motor [kW/hp] Si P0100 = 0 o 2, unidad de potencia del motor = [kW]
P0309=0	1	Si P0100 = 1, unidad de potencia del motor = [hp]Eficiencia nominal del motor [%]. Visible solamente cuando P0100 = 1. El ajuste 0 produce el cálculo interno del valor.
P0310=60	1	Frecuencia nominal del motor [Hz].
P0311=615	1	Velocidad nominal del motor [RPM].
P1900=2	2	Selección de la identificación de datos del motor. = 0: Deshabilitada = 2: Identificación de todos los parámetros en parada

Nota. Parámetros para la configuración del variador.

La Macro de conexión utilizada en la práctica Cn002 (PNP/NPN), ya que esta configuración realiza un control externo con el potenciómetro, figura 54.

Figura 54
Parámetro del variador

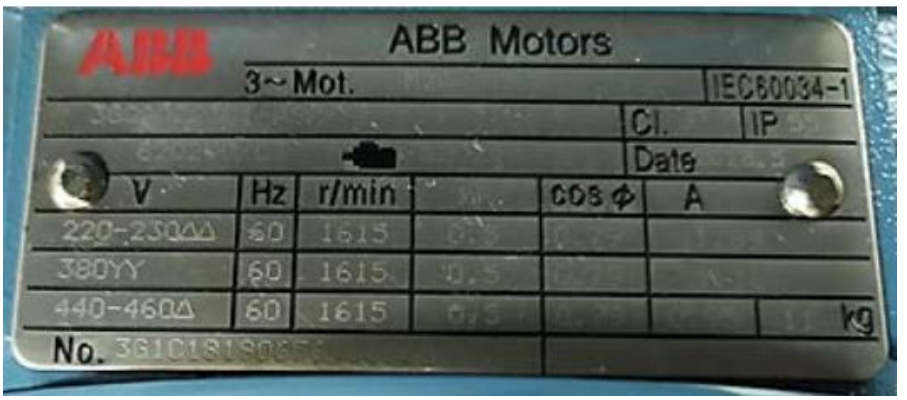




Nota. Parámetros que se utilizaron en el proyecto.

Se realiza una conexión estrella en el motor trifásico para alimentarlo a como indica la placa del fabricante en la Figura 55 colocada en el motor trifásico.

Figura 55
características del motor



Nota. Placa con sus especificaciones.

En la figura 56 se muestran seis terminales del motor trifásico U1, V1, W1 arriba y U2, V2, W2 abajo. Los puentes conectan U1 con W2, V1 con U2 y W1 con V2, formando una conexión en estrella, que permite al motor funcionar a su tensión nominal de manera eficiente.

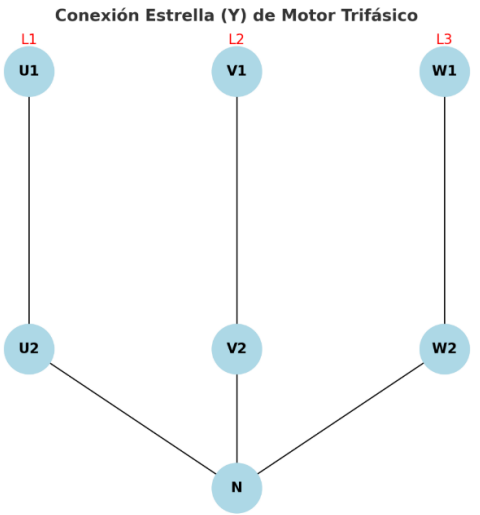
Figura 56
Conexión de los motores

Nota. Se observa la conexión estrella apropiada para el funcionamiento.

Por tanto, el diagrama eléctrico en conexión estrella (Y) del motor trifásico queda como se muestra en la figura 57.

- Las fases L1, L2, L3 se conectan a U1, V1, W1.
- Los extremos U2, V2, W2 se unen formando el punto neutro (N).
- La conexión a tierra (PE) va aparte al chasis del motor.

Figura 57
Conexión estrella

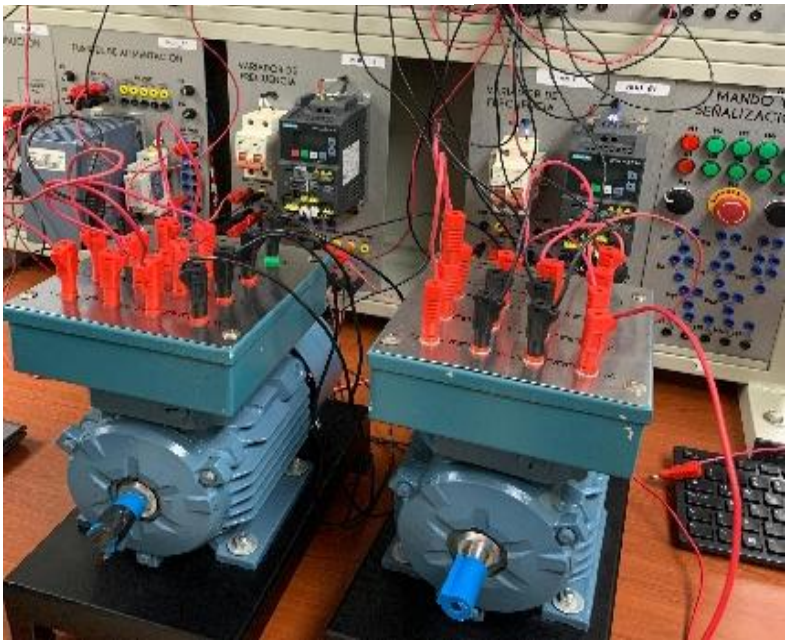


Nota. Se observa la conexión estrella en un diagrama para el funcionamiento.

5.13 Motores conectados a los variadores

Se procede la conexión de los Motore 1 y Motor 2, con las entradas y las salidas digitales que se configuro en el PLC, para el funcionamiento de los motores, la cual también se usó 2 variadores para los 2 motores y probar dos velocidades diferentes, en la figura 58 se mostrara.

Figura 58
Conexión de los motores

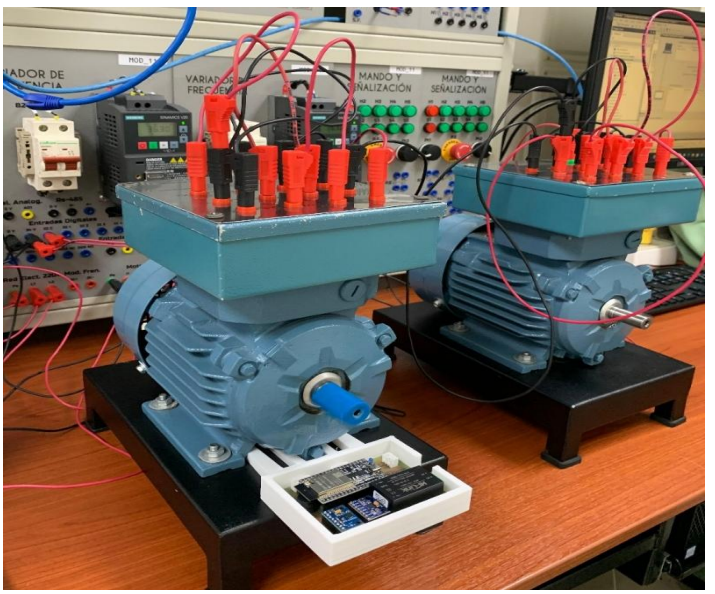


Nota. Conexión de los motores al Modulo

5.14 Implementación general

Se realizaron las respectivas pruebas para poder censar los movimientos del motor colocándolo encima del motor, pero una vez probado se realizará la base para que pueda ir implementado estéticamente, como se muestra en la figura 59.

Figura 59
Implementación general



Nota. Pruebas de frecuencia de los motores.

VI RESULTADOS

La primera práctica permitió implementar de manera adecuada un sistema IoT basado en Arduino y ESP32 para el monitoreo en tiempo real de vibraciones utilizando el sensor MPU6050. Se comprobó la capacidad del sistema para capturar, procesar y transmitir datos hacia la nube mediante Arduino IoT Cloud, lo que facilitó la visualización de la vibración. Además, en general la integración de hardware y software demostró ser eficiente, confiable y escalable, validando la importancia del IoT en la optimización de procesos de monitoreo.

En la segunda parte, se implementó el monitoreo de vibraciones utilizando el acelerómetro MPU6050 integrado con ESP32 y Arduino IoT Cloud lo que permitió registrar y exportar datos en tiempo real, con la aplicación de la Transformada Rápida de Fourier (FFT) en MATLAB se obtuvieron análisis de frecuencia que revelaron la presencia de armónicos y concentraciones de energía en bajas frecuencias, se realizó un análisis comparativo entre dos motores mostrando diferencias, el primero motor presentó vibraciones intermitentes y de menor magnitud, mientras que el segundo motor registró amplitudes más altas y frecuentes, sugiriendo posibles problemas de desbalance o desgaste.

6.1 PRÁCTICA 1

TEMA:

"Implementación de un sistema IoT con Arduino para la recolección de datos"

NÚMERO DE ESTUDIANTES 2

Integrantes:

Sebastián Alfredo Román Espinoza

Alberto Andrés Pesantes Morales

Docente: MSc. Rafael Franco Reina

TIEMPO ESTIMADO 2 HORAS

1. OBJETIVO GENERAL

- Implementar un sistema IoT basado en Arduino para la recolección y monitoreo en tiempo real de datos provenientes del acelerometro, con el fin de optimizar la obtención de información para su análisis.

2. OBJETIVOS ESPECÍFICOS

- Seleccionar e integrar los sensores adecuados según las variables a monitorear la vibración.
- Desarrollar y programar un algoritmo en Arduino para la captura, procesamiento y envío de datos.
- Implementar la comunicación IoT entre Arduino y la plataforma de almacenamiento o visualización de datos en la nube.
- Verificar y validar el correcto funcionamiento del sistema mediante pruebas de adquisición de datos en tiempo real.

3. NORMAS DE SEGURIDAD

- No consumir alimentos dentro del laboratorio.
- Utilizar únicamente software autorizado para la simulación de los módulos.
- Uso de Mandil y equipo adecuado durante las prácticas en el Laboratorio.

4. RECURSOS UTILIZADOS (EQUIPOS, Y ACCESORIO)

- Arduino IoT
- Cable micro USB
- ESP32
- Acelerómetro

5. MARCO TEÓRICO

5.1. Arduino

Arduino es un componente electrónico de código abierto, sus objetivos son contar con software y hardware fáciles de usar. Lo que permite el dispositivo es una infinidad de tipos de microordenadores de una sola placa, que pueden tener una variedad de usos según lo que requiera la persona que lo programe, a su vez permite desarrollar elementos autónomos para conectarse a otros dispositivos o interactuar con otros programas del hardware como con el software, en la figura 60, se observa el dispositivo de comunicación ARDUINO (arduino, 2018).

Figura 60
Dispositivo Arduino

Nota. Arduino que hará la comunicación entre los dispositivos (Arduino, 2025).

5.2.Transmisión de datos a la nube

La transmisión de datos a la nube habla de que su función es definir, implementar y supervisar un marco que mejora el rendimiento de los sistemas y los servicios de la nube. Dado los parámetros de la nube se expanden rápidamente más de lo que podrían manejar las personas de forma manual, la automatización hace posible y capaz un control más eficiente (Red Hat, 2022).

La nube brinda un proceso más organizado hacia la automatización de las tareas o actividades que se requieren en una gestión de la nube en flujos de trabajo más complejos. Al separar las diversas tareas de automatización de los distintos equipos y dominios en flujos de trabajo completos más eficientes, en la figura 61 se muestra la transmisión de datos a la nube (Kurduban, 2024).

Figura 61

Trasmisión de datos nube

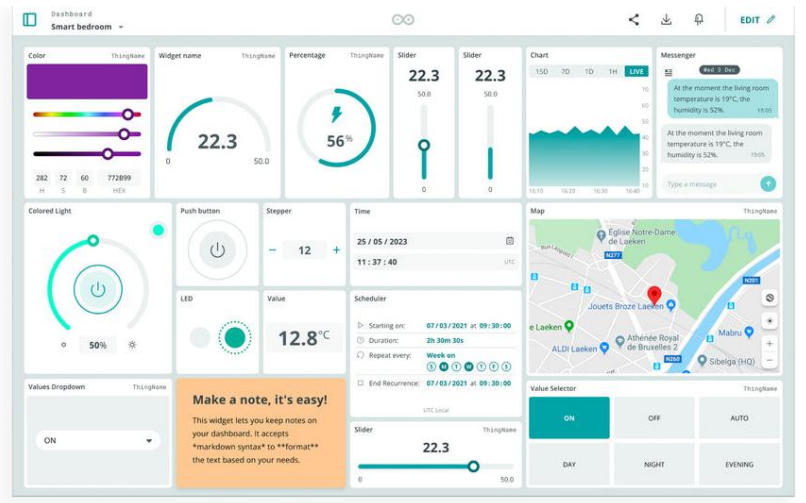
Nota. Serie de datos ingresando a la nube *(Garatu, 2017).*

5.3. Arduino IoT Cloud

Es una plataforma end-to-end que permite a los desarrolladores desplegar y monitorizar soluciones IoT sin necesidad de infraestructuras complejas. Facilita la gestión de dispositivos, sincronización de datos en tiempo real, creación de dashborards y automatización avanzada mediante OTA y programación desde la nube. Proporciona una compresión profunda del ecosistema Arduino Cloud, incluyendo la configuración de proyectos desde cero, el uso de APIs y SDKs, y la implementación de comunicaciones avanzadas como LoRaWAN y OTA, lo que lo convierte en un recurso

valioso para soluciones escalables y prácticas en áreas como smart-home, agricultura inteligente y monitoreo ambiental, como se muestra en la figura 62 (Kurniawan, 2021).

Figura 62
IoT Cloud



Nota. Frecuencias ingresando en la nube *(Arduino, 2025).*

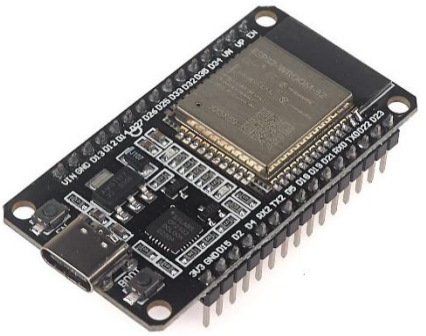
5.4. Sistemas embebidos (ESP32)

Los sistemas embebidos están desarrollados para integrarse en un sistema de mayor tamaño. Eso quiere decir, en lugar de ser un sistema independiente como una computadora tradicional, estos sistemas están destinados a ejecutar una cantidad de tareas exactas de un sistema más amplio. Su naturaleza de estar "embebidos" en la forma que operan, formando parte integral de la función global de ese dispositivo o sistema informático (Universidad Fidélitas, 2024).

5.5. Tarjeta ESP32

Poder describir el ESP32 como un microcontrolador que integra tecnologías Wifi y Bluetooth con las que permitirá conectarlo a internet o bien conectarlo con otros dispositivos. Se describe la principal funcionalidad y características de este, pero también los principales entornos de desarrollo y los lenguajes de programación que servirán para conseguir programar el dispositivo, se observa en la figura 63 el módulo de comunicación ESP32 *(Ortiz & Valencia, 2025)*.

Figura 63
Tarjeta ESP32



Nota. Dispositivo electrónico ESP32 para la comunicación *(Electrostore, 2025).*

5.6. PINES DEL ESP32

- **GPIO, ADC y DAC**

  El ESP32 cuenta con hasta 34 pines GPIO disponibles en placas de desarrollo. Ofrece además 18 canales ADC de alta resolución (12 bits) y 2 canales DAC de 8 bits para conversión de señal digital-analógica.

- **Sensores capacitivos (Touch Inputs)**

  Dispone de 10 pines táctiles capacitivos que permiten detectar toques, sirviendo también como fuente para despertar el ESP32 desde modo de suspensión profundo (deep sleep).

- **Interfaces de comunicación**

  El ESP32 integra múltiples protocolos de comunicación: 3 UART, 2 I²C, 4 SPI, 2 I²S, y un controlador SDIO/SD, lo que facilita la conexión con una amplia variedad de periféricos.

- **PWM y audio**

  Ofrece 16 canales PWM para control flexible de motores o iluminación. También

incluye interfaces I²S para-audio digital.

- **Pines restringidos (strapping y flash)**

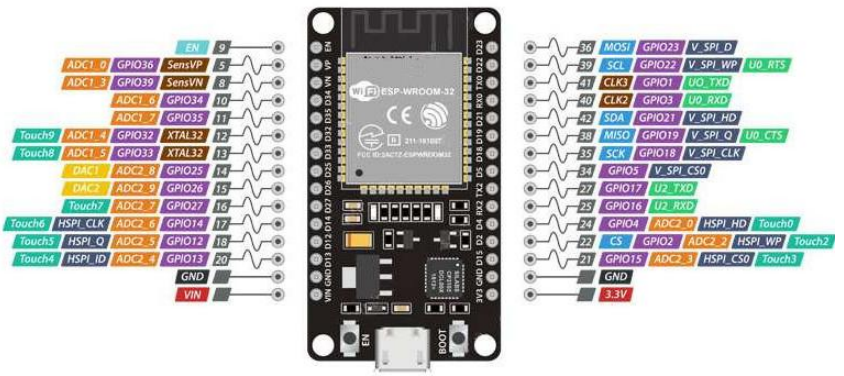
  Algunos pines, como GPIO6 a GPIO11, están vinculados internamente a la memoria flash y no deben usarse para aplicaciones generales. Otros, como GPIO0, GPIO2, GPIO12, entre otros, son pines de arranque (strapping) y afectan el modo de inicio del dispositivo.

- **Uso de ADC y Wi-Fi**

  Se recomienda utilizar canales ADC1 cuando el Wi-Fi está activo, ya que los canales ADC2 pueden presentar conflictos o funcionalidad limitada en ese modo, como se muestra en la figura 64 (Boyko, 2024)

Figura 64

Pines del ESP 32



Nota. Dispositivo electrónico ESP32 con sus Pines *(Electrostore, 2025).*

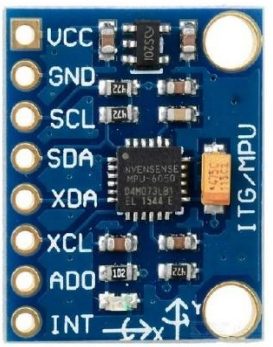
5.7. Acelerómetro MPU6050

Los sensores de vibración para los motores son muy comunes ya que estos se encuentran en una variedad de diferentes campos entre las fuentes eléctricas y mecánicas. Y si bien es cierto los sensores de vibración se centran más en las fuentes mecánicas, ya que ayuda a detectar algunas de las fuentes eléctricas y prevé futuras fallas (Manufactura Latam, 2024).

Es importante saber que las máquinas vibran, ya que un cambio mínimo en sus patrones de vibración puede mostrar una variedad de posibles problemas. Medir los cambios en los parámetros de vibración ayuda a los equipos a identificar desajustes, desequilibrios, holgura, desalineación o un sin número de diferentes fallas de los rodamientos en los equipos y prever de que se produzca alguna más. La vibración anormal o excesiva ocasiona un gran problema a que los componentes sufran y afectar vida útil del equipo. Los sensores de vibración para motores se instalan en las maquinarias para monitorear los cambios en diferentes frecuencias (López, 2024).

A su vez pueden recopilar información de diferentes fuentes ya que los datos recopilados se analizan para identificar fallas y determinar la gravedad del equipo, por tanto, la misión de los acelerómetros radica en la medida de aceleraciones gravitacionales estáticas, lo que les permite la medida de la desviación del ángulo al que el objeto que se mide se aparta de la vertical, siendo posible también realizar otro tipo de medida para que el instrumento tome medidas en relación con aceleraciones dinámicas y modificaciones en los comportamientos de un objeto bajo golpes, desplazamientos, impactos o vibraciones sustanciales; es decir, vibraciones de baja amplitud y frecuencia, las que pueden alcanzar a decenas de Hz, como se muestra en la figura 65 se observa el acelerómetro (Transfer Multisort Elektronik, 2025).

Figura 65
Acelerómetro



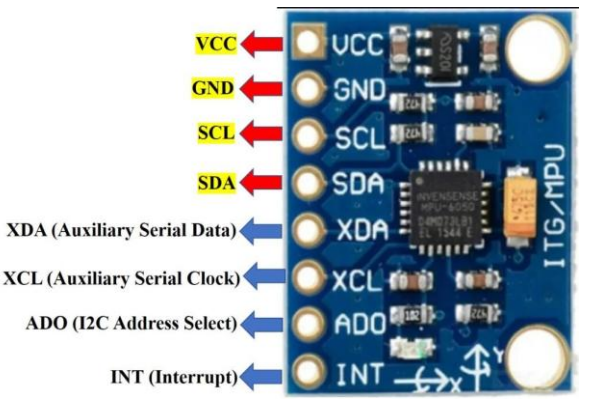
Nota. Acelerómetro utilizado para censar las vibraciones *(Sandorobotics, 2025)*.

5.8. PINES DEL ACELEROMETRO

El módulo MPU6050 dispone de pines esenciales para su funcionamiento y comunicación mediante I²C. Los pines VCC y GND suministran alimentación, mientras que SCL y SDA gestionan el reloj y los datos, como se muestra en la siguiente figura 66.

- PIN Función
- VCC Alimentación (3.3V o 5V)
- GND Tierra
- SCL Línea de reloj I²C
- SDA Línea de datos I²C
- XDA Salida de datos auxiliar I²C (opcional)
- XCL Reloj auxiliar I²C (opcional)
- AD0 Selección de dirección I²C (0 = 0x68, 1 = 0x69)
- INT Salida de interrupción (opcional, para eventos)

Figura 66
Pines del acelerómetro



Nota. Acelerómetro pines que se utilizaran (Nohuto, 2025).

6. INSTRUCCIONES PARA EL DESARROLLO DE LA PRÁCTICA

- Conectar el acelerómetro MPU6050 al Arduino/ESP32 mediante I²C.
- Configurar librerías y conexión Wi-Fi en el ESP32.
- Programar y cargar el código para leer y enviar datos a la nube.
- Probar vibraciones y observar resultados en Arduino IoT Cloud.
- Guardar lecturas y evidencias del funcionamiento.

7. MARCO PROCEDIMENTAL

7.1. Creación de Código en la aplicación Arduino

Se implementa el siguiente código para la comunicación del ESP32 con el acelerómetro MPU6050, permitiendo la adquisición de las señales de vibración generadas por el motor, los datos procesados se transmiten y se visualizan en la nube, facilitando el monitoreo remoto del comportamiento.

7.2. Librerías

Se utiliza una librería específica que permiten integrar distintas funcionalidades en el desarrollo. ArduinoIoTCloud y ConnectionHandler facilitan la conexión del dispositivo con la nube y la gestión de sus propiedades en línea, mientras que la librería arduinoFFT permite realizar el cálculo de la transformada rápida de Fourier, como se muestra en la siguiente figura 67.

Figura 67
Librería

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
#include <Wire.h>
#include <arduinoFFT.h>
```

Nota. Librería de Arduino que se va a utilizar.

7.3. Credenciales y WiFi

En este fragmento del código se configuran los parámetros necesarios para establecer la conexión del dispositivo con el Arduino IoT Cloud y con la red WIFI local. Se definen credenciales como el DEVICE_LOGIN_NAME, la SSID y la contraseña, además de la DEVICE_KEY que autentica el Thing creado en la plataforma. Finalmente, se declara el objeto WiFiConnectionHandler, encargado de gestionar la conexión inalámbrica del dispositivo con la nube, como se muestra en la figura 68.

Figura 68
Conexión con la red

```
const char DEVICE_LOGIN_NAME[]  = "c96b6d74-fe9c-4e34-a822-8f40621f01f9";
const char SSID[]               = "Galaxy A14 AAPM";    // Network SSID (name)
const char PASS[]               = "Nicolas12345";    // Network password (use for WPA,
const char DEVICE_KEY[]  = "1DA7EI84DSUK?HysIn87sswXZ";    // Secret device password
```

Nota. Conexión a con la red de Arduino que se va a utilizar

7.4. Registros del MPU6050

En este código se define la dirección I^2C del sensor MPU6050 y algunos registros clave, como la de gestión de energía y la lectura de aceleración en el eje X. Además, para el caso del ESP32, se especifican de forma explícita los pines de comunicación I^2C, asignado SDA=23 y SCL=22, lo que asegura una correcta comunicación entre modulo y el sensor, como se muestra en la figura 69.

Figura 69
Conexión con el MPU6050

Nota. Pines de Arduino que se va a utilizar.

7.5. Adquisición de muestras (eje X, en "g")

Se realiza la lectura de los datos crudos del acelerómetro en el eje X, los cuales se convierten a valores en g utilizando la escala predeterminada de ±2g (16384 LSB/g). Los datos obtenidos se almacenan en el arreglo vReal para su posterior análisis con la FFT, mientras que en vImag se inicializan en cero. Además, se implementa un retardo en microsegundos para espaciar las muestras y lograr una frecuencia de muestreo cercana a 1KHz, como se muestra en la figura 70.

Figura 70
Muestras (eje X, en "g")

```
106  // --- Adquisición de muestras en eje X (en 'g') ---
107  double sum = 0.0;
108  static double tbuf[SAMPLES]; // copia para RMS en tiempo
109
110  for (int i = 0; i < SAMPLES; i++) {
111    double g = (double)readAccelX_raw() / 16384.0; // ±2g → 16384 LSB/g
112    tbuf[i]  = g;
113    vReal[i] = g;
114    vImag[i] = 0.0;
115
116    sum += g;
117    delayMicroseconds((int)(1000000UL / SAMPLING_FREQUENCY));
118  }
```

Nota. Muestras de eje x y en g

7.6. Lectura cruda del acelerómetro

La función readAccelX_raw () y la función readAccelX_raw() posiciona el puntero en el registro ACCEL_XOUT_H del MPU6050 y realiza la lectura de dos bytes consecutivos que corresponden a la parte alta y baja del dato. Posteriormente, los combina en un valor int16_t con signo, que representa la medida cruda de aceleración en el eje X, como se muestra en la figura 71.

Figura 71
Lectura con el sensor

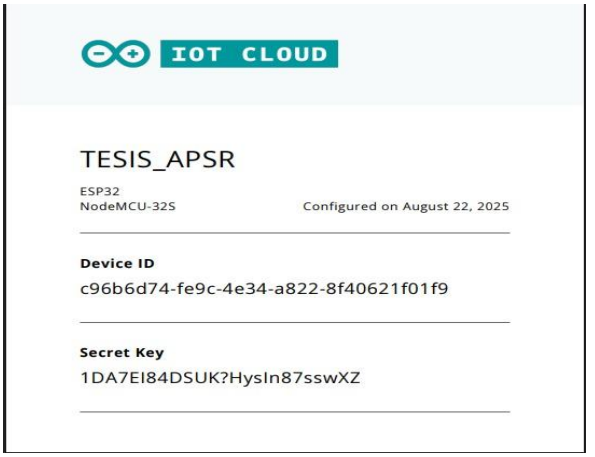
```
int16_t readAccelX_raw() {
  Wire.beginTransmission(MPU6050_ADDRESS);
  Wire.write(REG_ACCEL_XOUT_H);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU6050_ADDRESS, 2, true);
  int16_t raw = (Wire.read() << 8) | Wire.read();
  return raw;
}
```

Nota. Conexión con el acelerometro.

7.7. Conexíon con ESP32

Se descargó la aplicación Iot para la comunicación ESP32, una vez instalado se coloca un usuario y contraseña de la red, la cual permita la conexión con el módulo, en la figura 72 se mostrará el proceso.

Figura 72
Aplicación IOT



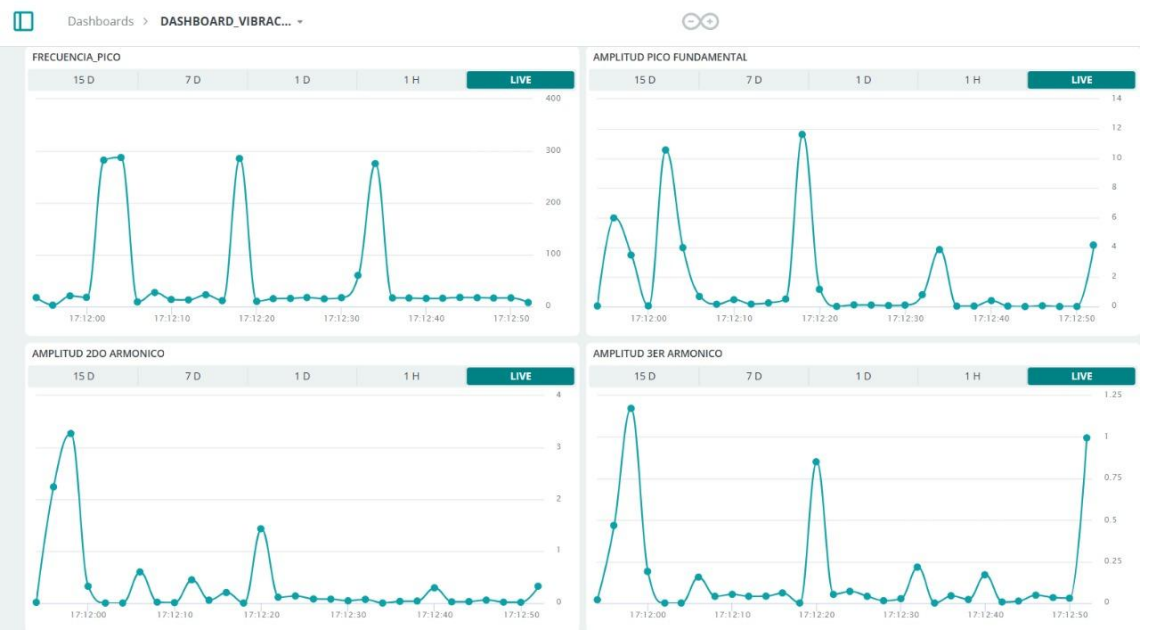
Nota. Configuración para comunicar con el ESP 32.

7.8. Resultado de las frecuencias del Motor 1

Se presenta un dashboard de monitoreo de vibraciones en tiempo real, figura 73, donde muestra la frecuencia pico y las amplitudes fundamentales, segundo y tercer armónico. Esta información permite detectar variaciones anormales y apoyar el diagnóstico predictivo del sistema analizado.

Figura 73

Armónico del Motor 1

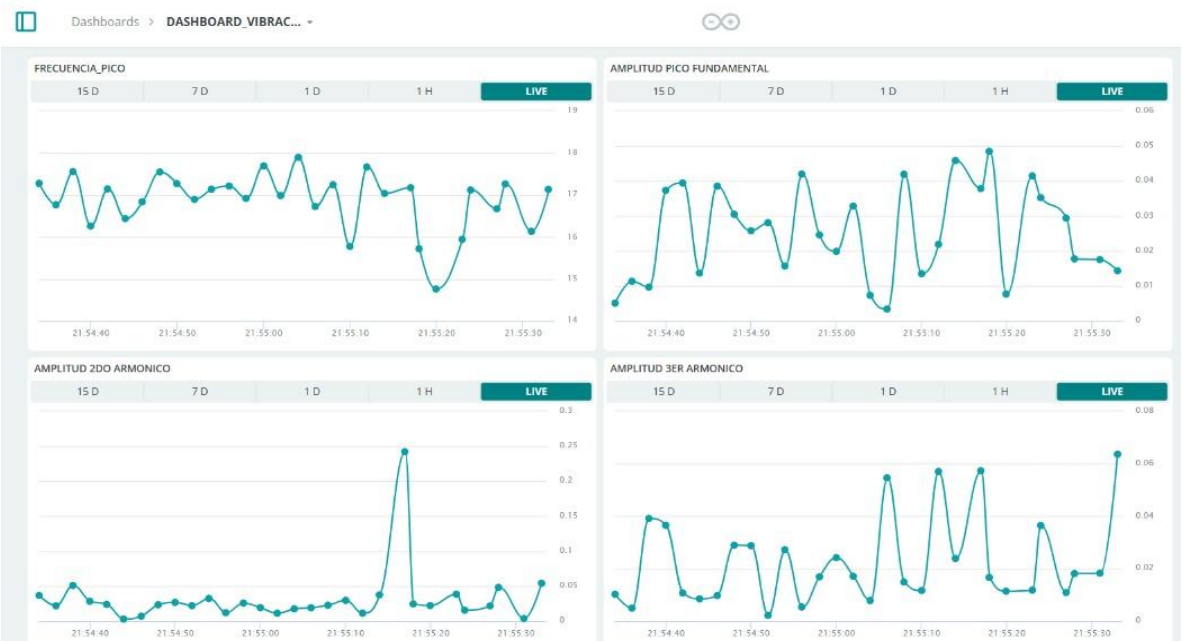


Nota. Detecciones de vibraciones normales.

7.9 Resultado de la frecuencia del Motor 2

La figura 74 muestra un dashboard de vibraciones en tiempo real, donde se registran la frecuencia pico y las amplitudes del pico fundamental, segundo y tercer armónico. Estos gráficos permiten observar el comportamiento dinámico del sistema y detectar posibles anomalías o variaciones que pueden indicar fallas incipientes.

Figura 74
Armónicos del motor 2



Nota. Detecciones de vibraciones anormales.

8. CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones

- Se logró implementar un sistema IoT funcional con Arduino y ESP32 para el monitoreo de vibraciones en tiempo real.
- La comunicación a través de Arduino IoT Cloud permitió visualizar los datos de manera remota y organizada.
- El acelerómetro MPU6050 demostró ser un sensor confiable para la detección de vibraciones de baja amplitud y frecuencia.
- El uso de librerías específicas facilitó la programación, procesamiento y transmisión de datos hacia la nube.

8.2. Recomendaciones

- Verificar siempre las conexiones I²C y parámetros de red antes de iniciar las pruebas.
- Realizar un filtrado digital de señales para mejorar la precisión de las mediciones.
- Probar el sistema en diferentes condiciones de vibración para validar su desempeño en distintos escenarios.
- Mantener actualizadas las librerias del ESP32 y Arduino IDE para asegurar la compatibilidad y estabilidad en la comunicación con la nube.

9. BIBLIOGRAFÍA

6.2 PRÁCTICA 2

TEMA:

"Análisis del espectro de frecuencia con datos obtenidos de Arduino IoT Cloud para aplicar la transformada de Fourier mediante Matlab"

NÚMERO DE ESTUDIANTES 2

Integrantes:

Sebastián Alfredo Román Espinoza

Alberto Andrés Pesantes Morales

Docente: MSc. Rafael Franco Reina

TIEMPO ESTIMADO 2 HORAS

1. OBJETIVO GENERAL

- Aplicar la Transformada de Fourier a los datos registrados por el prototipo en Arduino IoT Cloud, con el fin de analizar el comportamiento de las señales de vibración en el dominio de la frecuencia e identificar patrones relevantes como sus armónicos.

2. OBJETIVOS ESPECÍFICOS

- Obtener y exportar los datos generados por el acelerómetro MPU6050 a través de la plataforma Arduino IoT Cloud.
- Implementar la Transformada Rápida de Fourier (FFT) en MATLAB para representar y analizar la señal armónica.
- Generar representaciones gráficas y tablas que permitan interpretar los resultados del análisis espectral.

3. NORMAS DE SEGURIDAD

- No consumir alimentos dentro del laboratorio.
- Utilizar únicamente software autorizado para la simulación de los módulos.
- Uso de Mandil y equipo adecuado durante las prácticas en el Laboratorio.

4. RECURSOS UTILIZADOS (EQUIPOS, Y ACCESORIO)

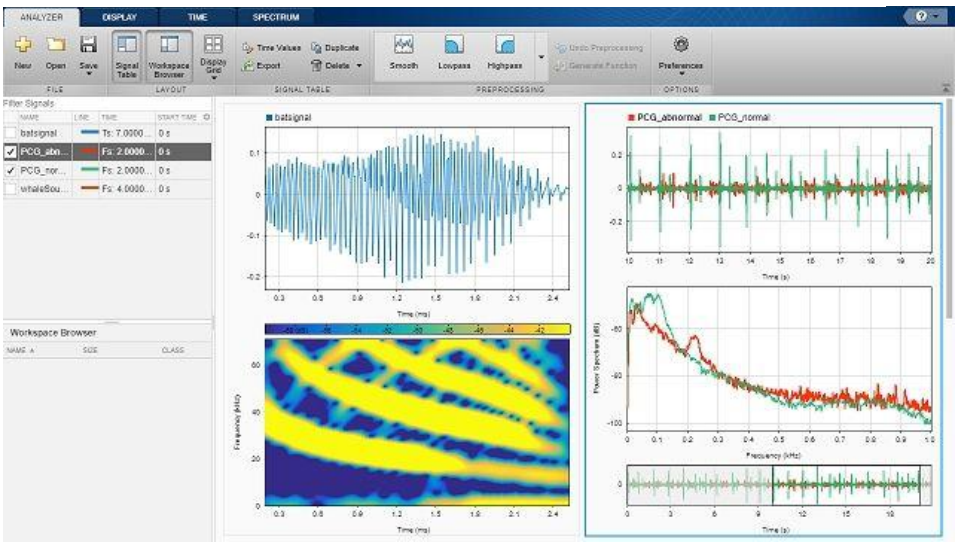
- Matlab
- Vibraciones
- Transformada de Fourier

5. MARCO TEÓRICO

5.1. Matlab

MATLAB es un programa que se usa mucho en ingeniería y ciencias porque ayuda a trabajar con cálculos numéricos y programación. Permite resolver operaciones matemáticas, hacer simulaciones, analizar señales, diseñar sistemas de control y graficar datos. Lo que lo hace muy útil es que combina un lenguaje de programación fácil de usar con herramientas ya listas para matemáticas, álgebra y análisis numérico, como se muestra en la figura 75 (MathWorks, 2021).

Figura 75
Matlab

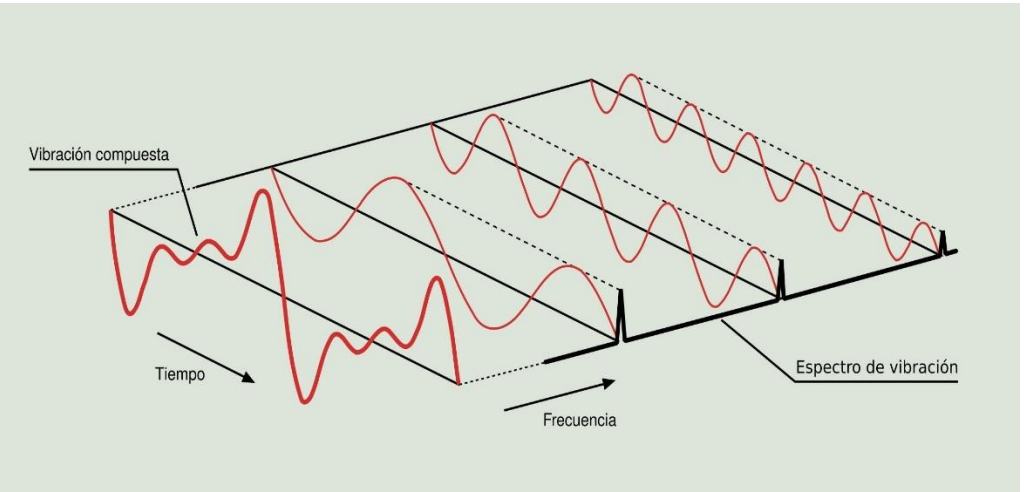


Nota. Se utilizo Matlab para la transformada de Fourier (MathWorks, 2025).

5.2. Vibraciones

La vibración se entiende como la oscilación de una masa respecto a su posición de equilibrio, generada por fuerzas dinámicas internas o externas, ya que toda máquina rotativa presenta vibraciones propias que pueden verse alteradas por factores como desalineaciones, fricción, desgaste o desbalance, por ello, los analizadores de vibraciones resultan esenciales para monitorear niveles, ya que su correcta interpretación permite anticipar fallas y tomar decisiones de mantenimiento más efectivas, como se muestra en la figura 76 (Mobley, 2002).

Figura 76
Análisis de vibraciones

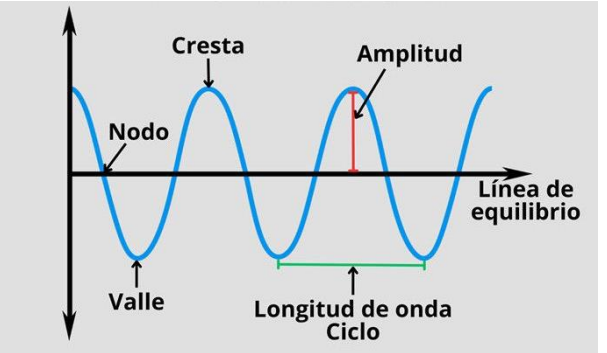


Nota. Análisis de vibraciones que se pueden utilizar (Fernández, 2017).

5.3. Características de una onda de vibración

Esta se caracteriza por tres parámetros principales: frecuencia, número de ciclos por segundo medidos en Hertz (Hz); período, tiempo requerido para completar un ciclo, siendo el inverso de la frecuencia; y amplitud, magnitud máxima del desplazamiento respecto al equilibrio, como se muestra en la figura 77 (Meirovitch, 2001).

Figura 77
Característica de vibración

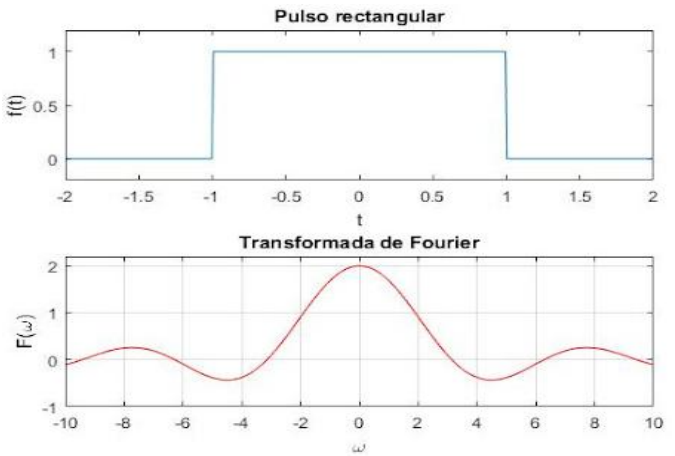


Nota. Estas con las 3 características de la vibración (Rhoton, 2025).

5.4. Transformada de Fourier

La Transformada de Fourier es una herramienta matemática fundamental para el análisis de señales y sistemas. Que permite descomponer cualquier señal en sus componentes de frecuencia, representando una señal en el dominio de la frecuencia en lugar del dominio del tiempo. Esta descomposición facilita el estudio de la energía y el comportamiento de señales periódicas o no periódicas, permitiendo identificar las frecuencias que componen una señal compleja, como se muestra en la siguiente figura 78 (Martínez, 2008).

Figura 78
Fourier



Nota. Ejemplo de gráfica aplicando la transformada de Fourier (Vasco, 2025).

5.5. Formula de la Transformada Discreta de Fourier

$$x[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi}{N}kn}$$

X[k] Componente espectral en la frecuencia k. Representa la amplitud y fase de la frecuencia k en la señal.

x[n] Señal discreta en el tiempo, con n=0,1, 2, …, N−1.

N Número total de muestras de la señal.

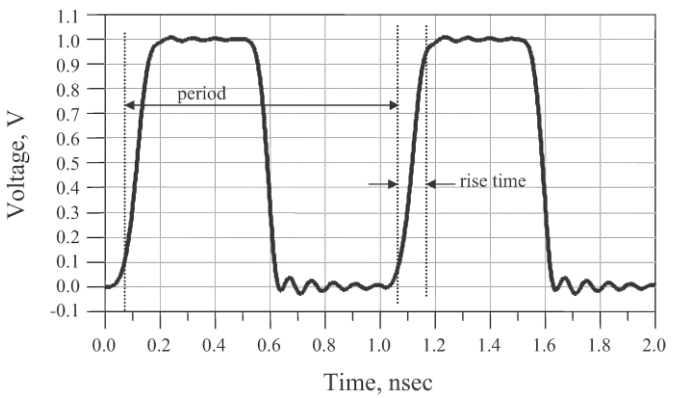
$\mathbf{e^{-j\frac{2\pi}{N}kn}}$ Factor complejo que descompone la señal en sus componentes de frecuencia, parte de los coeficientes de la base exponencial compleja.

La sumatoria $\sum_{n=0}^{N-1}$ recorre todas las muestras de la señal.

5.6. Dominio del tiempo

Un análisis de vibraciones en el dominio del tiempo se basa en estudiar cómo cambia la vibración de un motor o maquinas a lo largo del tiempo, ya que esto permite reconocer patrones o picos que pueden señalar fallas tempranas, a diferencia del análisis en frecuencia, este enfoque es útil para detectar problemas como impactos, desbalanceo y desalineaciones con comportamientos irregulares, como se muestra en la figura 79 (Randall, 2011)

Figura 79
Dominio del tiempo

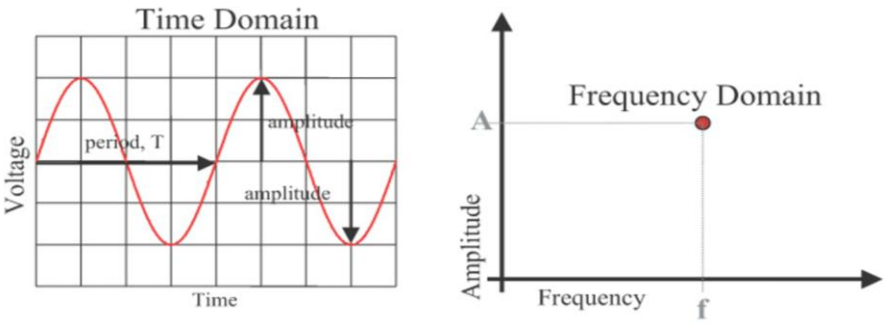


Nota. Ejemplo de gráfica aplicando dominio del tiempo. (Montejo, 2007).

5.7. Dominio de la frecuencia

El análisis en el dominio de la frecuencia estudia cómo se reparte la energía de una señal entre distintas frecuencias, ya que ayuda a identificar patrones o componentes repetitivos en las vibraciones de los motores, lo que permite detectar problemas como desbalanceo, desalineación o fallas en los rodamientos, a diferencia del análisis en el tiempo que muestra cómo cambia la vibración con el tiempo, el enfoque en frecuencia revela las causas de estas vibraciones, facilitando un diagnóstico más preciso, como se muestra en la figura 80 (Pruftechnik, 2011).

Figura 80
Dominio de la frecuencia



Nota. Ejemplo de gráfica aplicando dominio de la frecuencia (Wiki, 2025).

6. MARCO PROCEDIMENTAL

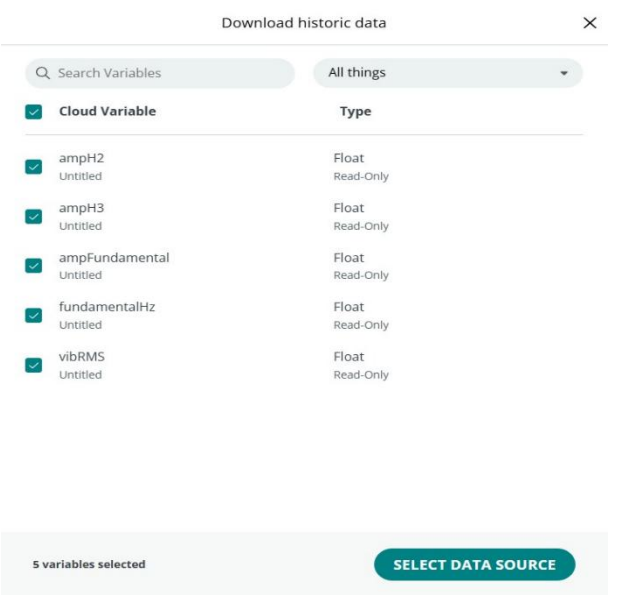
- Registrar y exportar los datos de vibración del motor desde la nube hacia MATLAB.
- Preparar los datos en MATLAB y graficar la señal en el dominio del tiempo para observar patrones o anomalías.
- Aplicar la Transformada Rápida de Fourier (FFT) para convertir la señal al dominio de la frecuencia.
- Analizar el espectro resultante para identificar armónicos, frecuencias principales y posibles fallas del equipo.
- Guardar gráficos, tablas y observaciones para documentar los resultados del análisis.

7. MARCO PROCEDIMENTAL

7.1. Download data

Se muestra una ventana para descargar datos históricos de la nube, donde se seleccionan variables de amplitud, frecuencia fundamental y vibración en formato float de solo lectura, listas para su análisis posterior, como se observa en la figura 81.

Figura 81
Datos Excel

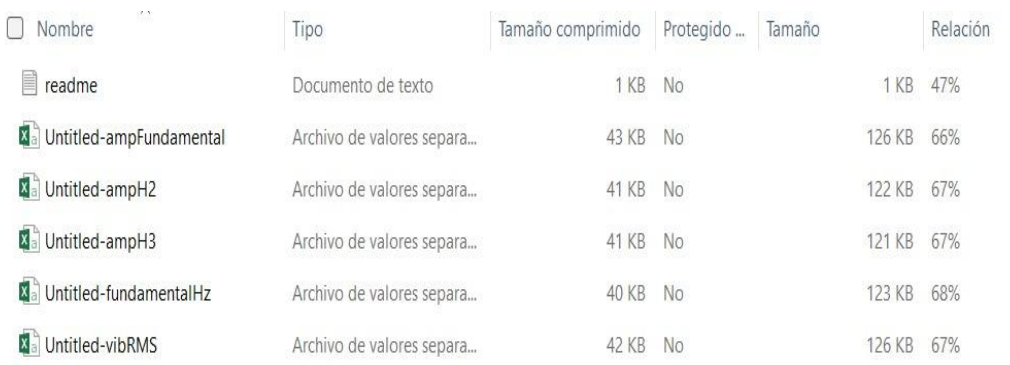


Nota. Datas de las vibraciones.

7.2. Carpetas de datos

Se muestra todos los resultados de la frecuencia se reflejan en carpetas que se descargan por variable de la data, como se muestra en la figura 82.

Figura 82
Carpeta de datos

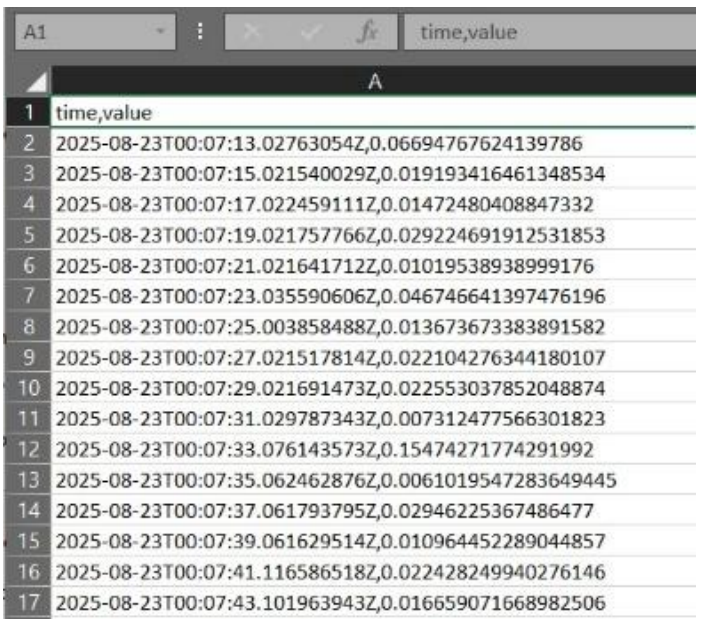


Nombre	Tipo	Tamaño comprimido	Protegido ...	Tamaño	Relación
readme	Documento de texto	1 KB	No	1 KB	47%
Untitled-ampFundamental	Archivo de valores separa...	43 KB	No	126 KB	66%
Untitled-ampH2	Archivo de valores separa...	41 KB	No	122 KB	67%
Untitled-ampH3	Archivo de valores separa...	41 KB	No	121 KB	67%
Untitled-fundamentalHz	Archivo de valores separa...	40 KB	No	123 KB	68%
Untitled-vibRMS	Archivo de valores separa...	42 KB	No	126 KB	67%

Nota. Todos los datos de las ondas se pueden descargar en Excel.

Los datos del motor 1 recopilados de la data por tiempo, para ejecutarlos en Matlab, como se muestra en la figura 83.

Figura 83
Datos recopilados



	A
1	time,value
2	2025-08-23T00:07:13.02763054Z,0.0669476762413978
3	2025-08-23T00:07:15.021540029Z,0.019193416461348534
4	2025-08-23T00:07:17.022459111Z,0.01472480408847332
5	2025-08-23T00:07:19.021757766Z,0.029224691912531853
6	2025-08-23T00:07:21.021641712Z,0.01019538938999176
7	2025-08-23T00:07:23.035590606Z,0.046746641397476196
8	2025-08-23T00:07:25.003858488Z,0.013673673383891582
9	2025-08-23T00:07:27.021517814Z,0.022104276344180107
10	2025-08-23T00:07:29.021691473Z,0.022553037852048874
11	2025-08-23T00:07:31.029787343Z,0.007312477566301823
12	2025-08-23T00:07:33.076143573Z,0.15474271774291992
13	2025-08-23T00:07:35.062462876Z,0.0061019547283649445
14	2025-08-23T00:07:37.061793795Z,0.02946225367486477
15	2025-08-23T00:07:39.061629514Z,0.010964452289044857
16	2025-08-23T00:07:41.116586518Z,0.022428249940276146
17	2025-08-23T00:07:43.101963943Z,0.016659071668982506

Nota. Datos de Excel que se descargaron del ARDUINO CLOUD.

Los datos del motor 2 recopilados de la data por tiempo, para ejecutarlos en Matlab, como se muestra en la figura 84.

Figura 84

Datos Recopilados

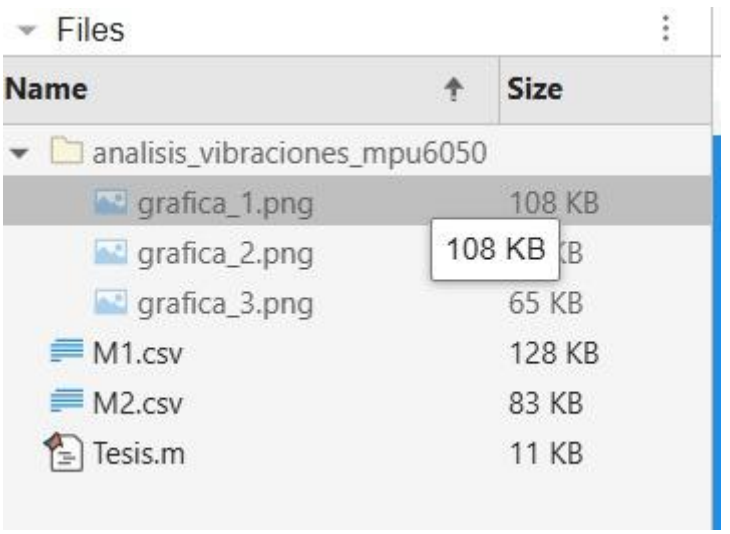
	A
1	time,value
2	2025-09-03T22:10:46.258582775Z,16.572608947753906
3	2025-09-03T22:10:48.265828572Z,17.42607879638672
4	2025-09-03T22:10:50.218524041Z,17.21927261352539
5	2025-09-03T22:10:52.259761451Z,17.48847770690918
6	2025-09-03T22:10:54.259220045Z,16.78023910522461
7	2025-09-03T22:10:56.258565642Z,16.984783172607422
8	2025-09-03T22:10:58.258374224Z,16.89009666442871
9	2025-09-03T22:11:00.258353364Z,17.339927673339844
10	2025-09-03T22:11:02.299237342Z,17.411840438842773
11	2025-09-03T22:11:04.298449055Z,17.00434684753418
12	2025-09-03T22:11:06.299099853Z,16.94566535949707
13	2025-09-03T22:11:08.298552839Z,17.58220863342285
14	2025-09-03T22:11:10.474387341Z,16.992429733276367
15	2025-09-03T22:11:12.306107099Z,17.172147750854492
16	2025-09-03T22:11:14.299137328Z,17.1751708984375
17	2025-09-03T22:11:16.298823468Z,17.564090728759766
18	2025-09-03T22:11:18.338435889Z,17.086557388305664
19	2025-09-03T22:11:20.338275155Z,17.168703079223633
20	2025-09-03T22:11:22.338767025Z,16.516132354736328

Nota. Datos de Excel que se descargaron del ARDUINO CLOUD.

7.3. Transformada de Fourier

En esta sección se importan los datos recopilados del IOT al MATLAB de ambos motores que están guardados como CSV para su respectivo análisis, como se muestra en la figura 85.

Figura 85
Datos a Matlab

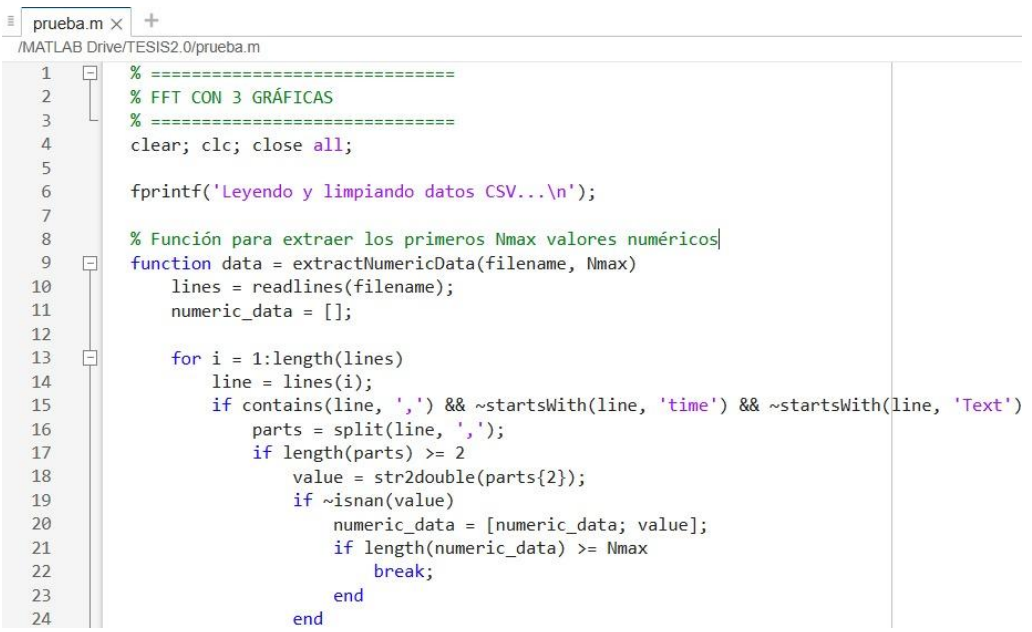


Nota. Datos de Excel al MATLAB.

7.4. Ejecución de los datos en Matlab

Esta línea de código define con el comando "extractNumericData" cuya función es leer los datos importados que se descargaron para leer las columnas y las filas del archivo, el proceso continuo hasta recopilar la cantidad de datos máximo indicada por el parámetro Nmax listos para analizar, como se muestra la figura 86.

Figura 86
Parte 1 Matlab



Nota. Código de Matlab.

Esta parte del código se lee los primeros 20 datos de dos archivos (Motor1.csv y Motor2.csv), guarda esos valores en vectores, cuenta cuántos datos tiene cada motor y luego muestra en pantalla un resumen indicando la cantidad de valores obtenidos y el rango entre el primero y el último dato, como se muestra en la figura 87.

Figura 87
Parte 2 Matlab

```
% Extraer primeros 20 datos
y1 = extractNumericData("Motor1.csv", 20);
y2 = extractNumericData("Motor2.csv", 20);

% Número de datos
N = length(y1);

fprintf('✓ Datos extraídos correctamente:\n');
fprintf('Motor 1: %d valores, desde %.4f hasta %.4f\n', length(y1), y1(1), y1(end));
fprintf('Motor 2: %d valores, desde %.4f hasta %.4f\n', length(y2), y2(1), y2(end));
```

Nota. Código de Matlab.

El código calcula la Transformada de Fourier de las señales de los motores, primero define la frecuencia de muestreo, elimina el valor medio, aplica la FFT con 1024 puntos y obtiene el espectro de frecuencias y su magnitud para analizar en qué frecuencias se concentran las vibraciones, como se muestra en la figura 88.

Figura 88
Parte 3 Matlab

prueba.m
/MATLAB Drive/TESIS2.0/prueba.m

```
% ==============================
% FRECUENCIA DE MUESTREO
% ==============================
% Si no tienes timestamps, asumimos fs=100 Hz como ejemplo
fs = 100;  % <<-- AJUSTA ESTO SEGÚN TU SISTEMA REAL

% ==============================
% FFT
% ==============================
y1_centered = y1 - mean(y1);
y2_centered = y2 - mean(y2);

Nfft = 1024;
Y1 = fft(y1_centered, Nfft);
Y2 = fft(y2_centered, Nfft);

f = (0:Nfft-1)*(fs/Nfft);   % Vector de frecuencias
mag1 = abs(Y1);
mag2 = abs(Y2);
```

Nota. Código de Matlab.

El código dibuja una gráfica comparando el espectro de frecuencia de los dos motores, mostrando las magnitudes normalizadas frente a la frecuencia, con etiquetas, leyenda y cuadrícula para facilitar la interpretación, como en la figura 89.

Figura 89

Parte 4 Matlab

prueba.m

/MATLAB Drive/TESIS2.0/prueba.m

```
62  % ==============================
63  % GRÁFICAS
64  % ==============================
65  figure('Position', [100, 100, 1500, 500]);
66
67  % (1) Espectro FFT normalizado
68  subplot(1,3,1);
69  plot(f, mag1/max(mag1), 'b-', 'LineWidth', 2); hold on;
70  plot(f, mag2/max(mag2), 'r-', 'LineWidth', 2);
71  xlabel('Frecuencia (Hz)');
72  ylabel('Magnitud Normalizada');
73  title('FFT Dominio de la frecuencia');
74  legend('Motor 1', 'Motor 2');
75  grid on;
76  xlim([0 fs/2]);
77  xticks(0:10:fs/2);
```

Nota. Código de Matlab.

En el código se realizó una gráfica en un subplot de la magnitud completa de la FFT de ambos motores, mostrando los valores sin normalizar frente a la frecuencia (f), con etiquetas, título, leyenda, cuadrícula y rango de frecuencia limitado a la mitad de la frecuencia de muestreo, como se muestra en figura 90.

Figura 90

Parte 5 Matlab

prueba.m ×

/MATLAB Drive/TESIS2.0/prueba.m

```
89
90    % (3) Magnitud compleja tipo MATLAB oficial
91    subplot(1,3,3);
92    plot(f, mag1, 'b-', 'LineWidth', 1.5); hold on;
93    plot(f, mag2, 'r-', 'LineWidth', 1.5);
94    xlabel('f (Hz)');
95    ylabel('|fft(X)|');
96    title('Complex Magnitude of FFT Spectrum');
97    legend('Motor 1', 'Motor 2');
98    grid on;
99    xlim([0 fs/2]);
100   xticks(0:10:fs/2);
101
```

Nota. Código de Matlab.

En esta sección de códigos se calcula cómo cambia la frecuencia dominante de cada motor a lo largo del tiempo, analizando ventanas pequeñas de la señal y guardando las frecuencias principales en cada segmento., como en la figura 91.

Figura 91

Parte 6 Matlab

prueba.m ×

/MATLAB Drive/TESIS2.0/prueba.m

```
102   % ============================
103   % (4) Frecuencia en el tiempo (simplificado)
104   % ============================
105   window = 5;  % tamaño de ventana (número de muestras por análisis)
106   step = 1;    % avance entre ventanas
107   freq_dom1 = [];
108   freq_dom2 = [];
109   time_axis = [];
110
111   for i = 1:step:(N-window)
112       seg1 = y1_centered(i:i+window-1);
113       seg2 = y2_centered(i:i+window-1);
114
115       % FFT de cada segmento
116       Yseg1 = abs(fft(seg1, 256));
117       Yseg2 = abs(fft(seg2, 256));
118       fseg = (0:255)*(fs/256);
119
120       % Pico dominante (descartando DC en f=0)
121       [~, idx1] = max(Yseg1(2:end));
122       [~, idx2] = max(Yseg2(2:end));
123
124       freq_dom1(end+1) = fseg(idx1+1);
125       freq_dom2(end+1) = fseg(idx2+1);
126       time_axis(end+1) = i/fs;
127   end
128
```

Nota. Código de Matlab.

En el código se genera dos tipos de grafica que muestran la distribución de amplitudes de vibración por cada motor, como se muestra en la figura 92.

Figura 92

Parte 7 Matlab

Tesis.m

/MATLAB Drive/TESISAAPM/Tesis.m

```
187  %% Figura 3: Análisis estadístico
188  figure('Position', [150, 150, 1200, 600], 'Name', 'Análisis Estadístico');
189
190  subplot(1,2,1);
191  histogram(valores_M1, 50, 'FaceColor', 'blue', 'FaceAlpha', 0.7);
192  hold on;
193  histogram(valores_M2, 50, 'FaceColor', 'red', 'FaceAlpha', 0.7);
194  title(' Distribución de Amplitudes', 'FontSize', 14, 'FontWeight', 'bold');
195  xlabel('Amplitud de Vibración', 'FontSize', 12);
196  ylabel('Frecuencia', 'FontSize', 12);
197  legend('Motor 1', 'Motor 2');
198  grid on;
199
200  subplot(1,2,2);
201  % Crear boxplot para vectores de diferente tamaño
202  datos_combinados = [valores_M1, valores_M2];
203  grupos = [ones(1, length(valores_M1)), 2*ones(1, length(valores_M2))];
204  boxplot(datos_combinados, grupos, 'Labels', {'Motor 1', 'Motor 2'});
205  title(' Comparación de Distribuciones', 'FontSize', 14, 'FontWeight', 'bold');
206  ylabel('Amplitud de Vibración', 'FontSize', 12);
207  grid on;
```

Nota. Código de Matlab.

Esta línea del código realiza una gráfica que muestra cómo varía la frecuencia dominante de cada motor en el tiempo, usando los datos calculados previamente freq_dom1 y freq_dom2, con colores distintos, etiquetas, título, leyenda y cuadrícula., como se muestra en figura 93.

Figura 93
Parte 8 Matlab

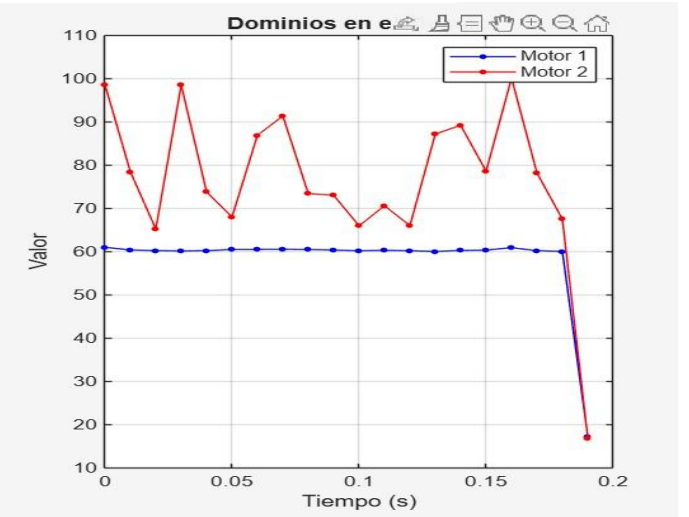
```
% Gráfica
figure;
plot(time_axis, freq_dom1, 'b.-', 'MarkerSize', 12); hold on;
plot(time_axis, freq_dom2, 'r.-', 'MarkerSize', 12);
xlabel('Tiempo (s)');
ylabel('Frecuencia dominante (Hz)');
title('Frecuencia en el tiempo (simplificado)');
legend('Motor 1','Motor 2');
grid on;
```

Nota. Código de Matlab.

7.5. Monitoreo de vibraciones

Se muestra cómo se comportan los motores a lo largo del tiempo. El Motor 1 se mantiene en un valor constante de las vibraciones cercanas a 60 Hz, lo que indica un comportamiento estable. En cambio, el Motor 2 presenta valores que oscilan de manera excesiva entre aproximadamente 65hz y 100hz, evidenciando un comportamiento inestable, figura 94.

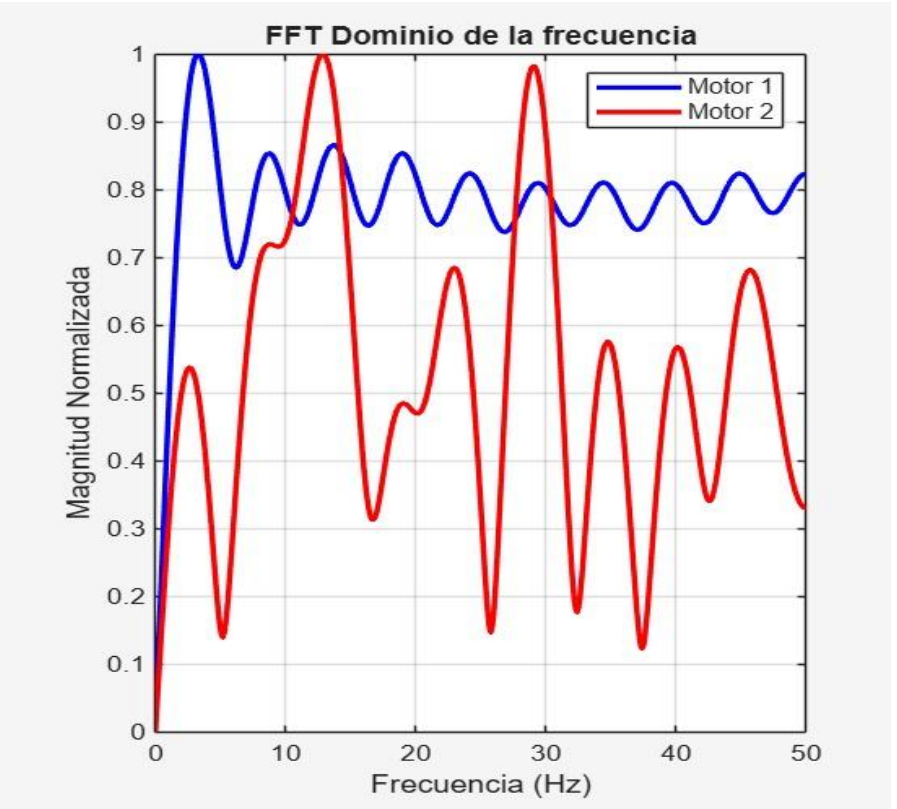
Figura 94
Dominio de tiempo



Nota. Vibración en el dominio de tiempo del Motor 1 y Moto 2.

En esta gráfica se muestra la intensidad de la vibración en distintas frecuencias, en la cual el Motor 1 mantiene una vibración baja y estable, con valores que oscilan entre 0.75 y 0.85 de magnitud normalizada a lo largo del rango de frecuencias 0–50 Hz, en cambio el Motor 2 presenta vibraciones más altas e inestables, alcanzando picos cercanos a 1.0 en 10 Hz, 20 Hz y 30 Hz, además de valores variables entre 0.2 y 0.95 en el resto de las frecuencias. Esto quiere decir que existe un problema de resonancia o desbalance en el Motor 2, figura 95.

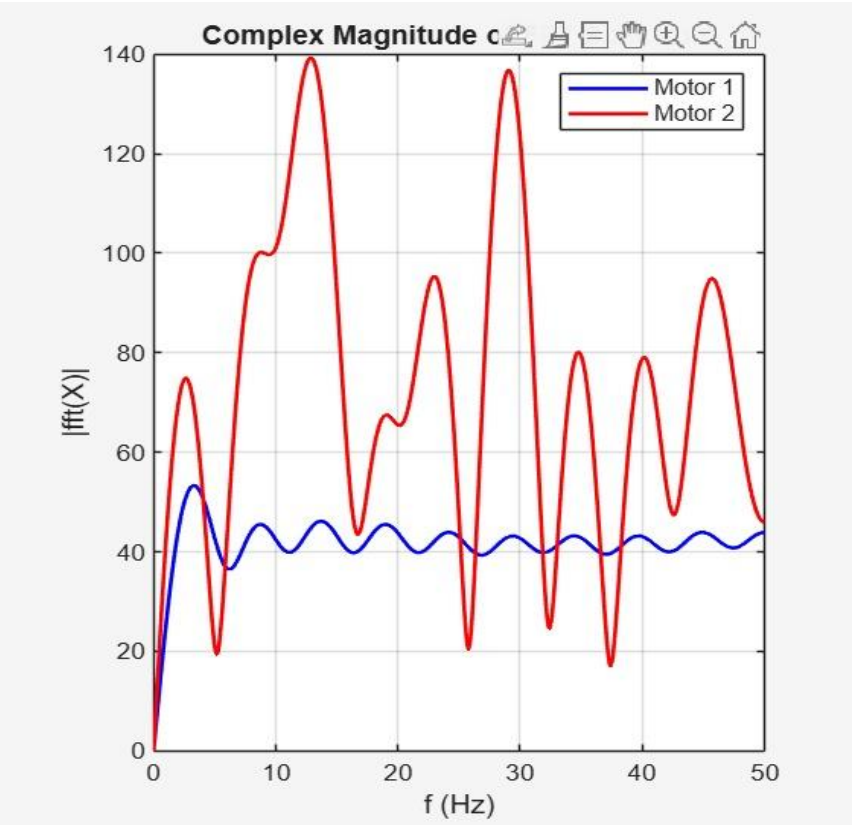
Figura 95
Comparación de vibraciones



Nota. Dominio de la frecuencia del Motor 1 y Motor 2.

Mediante la siguiente grafica se muestra la magnitud real de las vibraciones, el motor 1 tiene vibraciones que no superan a los 50hz, a diferencia del motor 2 que superan a valores de vibración hasta 140hz lo cual muestra picos más altos, eso quiere decir que sus vibraciones son más altas e indica una posible falla, figura 96.

Figura 96
Distribución de amplitudes



Nota. Distribución de amplitudes del Motor 1 y Motor 2.

El análisis de vibraciones con el sensor MPU6050 muestra diferencias claras entre el Motor 1 y el Motor 2 como el dominio de la frecuencia, dominio del tiempo y el espectro FFT estabilidad, permitiendo identificar el estado dinámico de cada equipo, como se muestra en la Tabla 2.

Tabla 2
Comparación de frecuencias

Parámetro	Motor 1	Motor 2
FFT Dominio de la Frecuencia	Magnitud casi constante ≈ 0.8	Magnitud variable, con picos cercanos a 0.95 en 10 Hz, 20 Hz y 30 Hz
Dominio en el Tiempo	Señal estable, valores ~60	Señal fluctuante entre 65 y 100, con picos frecuentes
Espectro FFT (Magnitud Compleja)	Magnitud entre 35 y 45 (casi plana)	Magnitud variable, con picos hasta ~140 en 10 Hz, 20 Hz y 30 Hz
Comportamiento General	Estable, poca variación en frecuencia y tiempo	Inestable, alta variación en frecuencia y tiempo

Nota. En la tabla comparativa de frecuencia se muestra el análisis de ambos motores.

8. CONCLUSIONES Y RECOMENDACIONES

8.1. Conclusiones

- La adquisición de la señal del acelerómetro mediante Arduino IOT Cloud permitió recopilar y exportar los registros de aceleración de los ejes y el total N 20 número de muestras por archivo ya que su integración e MATLAB muestra sus cálculos y graficas.
- La implementación de la Transformada de Fourier (FFT) en Matlab contribuyó en representar, mostrar y analizar la señal armónica en el dominio de la frecuencia, facilitando la interpretación del comportamiento vibratorio del sistema.
- El análisis en el dominio del tiempo y frecuencia evidenció que el motor 1 tiene un comportamiento estable con amplitudes reducidas entre 59hz a 61hz en las oscilaciones vibratorias a diferencia del motor 2 que presenta de 18hz a 105hz oscilaciones y picos más elevados, lo que refleja la magnitud normalizada y el espectro es que una energía mayor vibratoria y un posible desequilibrio.

8.2. Recomendaciones

- Revisar mecánicamente Motor 2: rodamientos, alineación y fijaciones, ya que sus picos FFT (hasta 1.0) y valores temporales (>100) superan ampliamente los del Motor 1.
- Definir umbral de alarma: si FFT >0.9 o señal temporal ±90, programar inspección inmediata del Motor 2.
- Tomar más muestras (≥200 ms) y ampliar el análisis >50 Hz para detectar posibles armónicos o resonancias adicionales.
- Aplicar mantenimiento predictivo periódico al Motor 2, comparando con la línea base del Motor 1; intervenir si magnitud >130 unidades o aumenta >10 % entre mediciones.

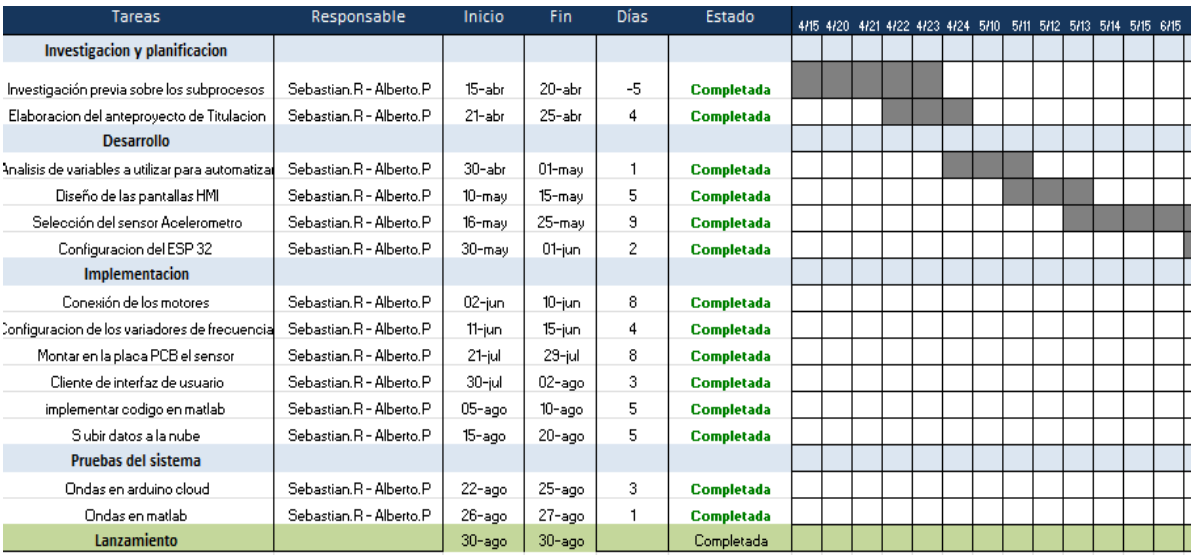
9. BIBLIOGRAFÍA

VII CRONOGRAMA DE ACTIVIDADES

Se observa cada actividad que se ha seguido cronológicamente de una forma adecuada y coherente, esto mediante diferentes fases del proyecto en las fechas fijadas para su cumplimiento, como se muestra en la figura 97 y 98.

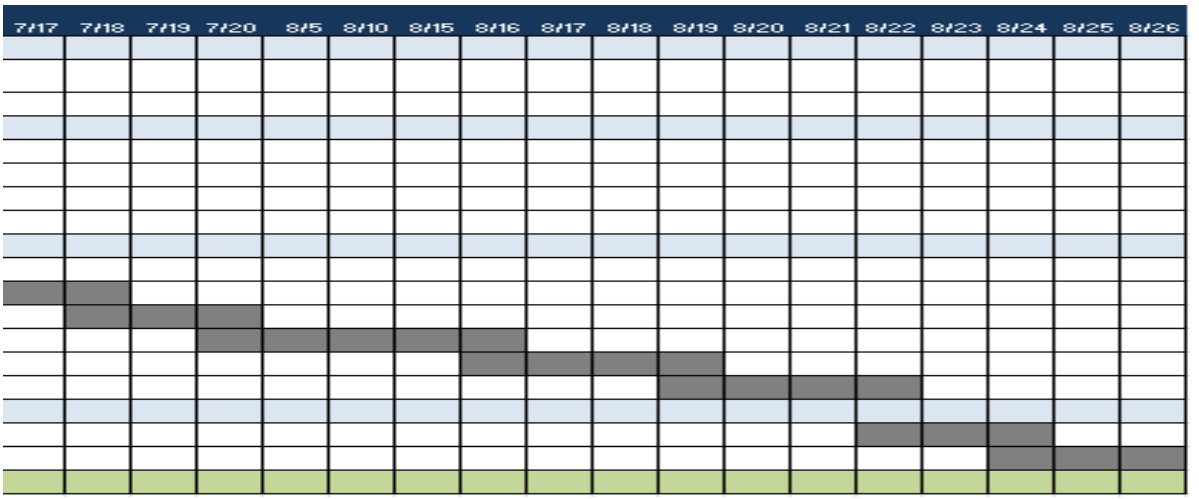
Figura 97
Cronograma de actividades

Tareas	Responsable	Inicio	Fin	Días	Estado
Investigacion y planificacion					
Investigación previa sobre los subprocesos	Sebastian.R - Alberto.P	15-abr	20-abr	-5	**Completada**
Elaboracion del anteproyecto de Titulacion	Sebastian.R - Alberto.P	21-abr	25-abr	4	**Completada**
Desarrollo					
Analisis de variables a utilizar para automatizar	Sebastian.R - Alberto.P	30-abr	01-may	1	**Completada**
Diseño de las pantallas HMI	Sebastian.R - Alberto.P	10-may	15-may	5	**Completada**
Selección del sensor Acelerometro	Sebastian.R - Alberto.P	16-may	25-may	9	**Completada**
Configuracion del ESP 32	Sebastian.R - Alberto.P	30-may	01-jun	2	**Completada**
Implementacion					
Conexión de los motores	Sebastian.R - Alberto.P	02-jun	10-jun	8	**Completada**
Configuracion de los variadores de frecuencia	Sebastian.R - Alberto.P	11-jun	15-jun	4	**Completada**
Montar en la placa PCB el sensor	Sebastian.R - Alberto.P	21-jul	29-jul	8	**Completada**
Cliente de interfaz de usuario	Sebastian.R - Alberto.P	30-jul	02-ago	3	**Completada**
implementar codigo en matlab	Sebastian.R - Alberto.P	05-ago	10-ago	5	**Completada**
S ubir datos a la nube	Sebastian.R - Alberto.P	15-ago	20-ago	5	**Completada**
Pruebas del sistema					
Ondas en arduino cloud	Sebastian.R - Alberto.P	22-ago	25-ago	3	**Completada**
Ondas en matlab	Sebastian.R - Alberto.P	26-ago	27-ago	1	**Completada**
Lanzamiento		30-ago	30-ago		Completada



Nota. Se establecen diferentes fases del proyecto para la distribución de los meses de trabajo

Figura 98
Parte 2 del cronograma



Nota. Fases en base a las fechas establecidas.

VIII PRESUPUESTO

El presupuesto total del proyecto tiene un estimado de $379,10, valor que incluye tanto los componentes electrónicos como el trabajo de ingeniería. Los costos más representativos corresponden a la elaboración de la placa PCB y a las horas de ingeniería empleadas, que representan la mayor parte del gasto. Los demás elementos, como el ESP32, el Arduino y el acelerómetro, tienen un costo más bajo en comparación. En conjunto, la inversión realizada asegura la implementación completa del sistema de control y monitoreo, integrando tanto el hardware como el diseño técnico necesario para su funcionamiento, como se refleja en la tabla 3.

Tabla3
Presupuesto

CANTIDAD	DESCRIPCION	VALOR UNITARIO	VALOR TOTAL
1	ESP32	$ 15	$ 15
1	ARDUINO	$ 27,10	$ 27,10
1	ACELEROMETRO	$ 7	$ 7
1	PLACA PCB	$ 300	$ 300
120 HORAS	BASE 3D	$ 30	$ 0
TOTAL			$ 379.10

*Nota***.** Esta tabla de presupuesto se muestra los valores de los costos unitario y totales para realizar el prototipo.

IX CONCLUSIONES

En este proyecto se logró diseñar e implementar un prototipo funcional para el monitoreo de vibraciones en motores eléctricos, utilizando una tarjeta embebida ESP32, sensores acelerómetro MPU6050, el sistema permitió medir en tiempo real la frecuencia fundamental, sus armónicos y el valor RMS de la vibración, demostrando que los motores operan dentro de parámetros normales y que el prototipo es capaz de detectar cambios que podrían indicar fallas futuras.

Se implementó un sistema de transmisión de datos hacia la nube, en la cual permitió el almacenamiento y visualización remota de las vibraciones registradas en el dominito del tiempo y las guardado en un Excel, gracias a esta integración los datos obtenidos por el sensor MPU6050 y procesados por la tarjeta ESP32 pudieron enviarse de manera eficiente a la plataforma en línea, lo que facilita su análisis en cualquier momento y desde diferentes ubicaciones.

Comprando ambos motores se analizó que el Motor 1 presenta un comportamiento estable, con valores en el tiempo alrededor de 60 unidades, una FFT casi plana (~0.8) y magnitudes complejas entre 35 y 45, lo que indica un funcionamiento balanceado y sin vibraciones relevantes en contraste, el Motor 2 muestra inestabilidad con valores en el tiempo que oscilan entre 65 y 100, picos importantes en la FFT a 10 Hz ~0.95, 20 Hz ~0.9 y 30 Hz ~0.85, y magnitudes complejas que alcanzan hasta 140, evidenciando resonancias y vibraciones significativas que sugieren desbalance o desgaste mecánico.

X RECOMENDACIONES

Se recomienda optimizar el tiempo de transmisión de datos hacia la nube reduciendo el rango de 7 a 10 segundo observado en las pruebas, ya que una menor latencia permitirá obtener una respuesta más inmediatas antes anomalías y mejorar la capacidad de reacción frente a posibles fallas en los motores.

Es aconsejable ampliar el sistema incorporando más sensores, no solo acelerómetros, sino también de temperatura y corrientes, para contar con un diagnóstico más completos de los estados de motores y lograr un análisis predictivo más robusto. Ampliar el prototipo para monitorear varios motores a la vez.

Desde la perspectiva académica, se sugiere seguir utilizando este prototipo como recursos didácticos, pero complementarlo con guías prácticas y simulaciones que permiten a los estudiantes profundizar en temas como programación embebida, análisis de vibraciones y conectividad IoT aplicada a la industria.

También sería conveniente escalar el proyecto hacia entornos industriales más exigentes, reforzando la protección del hardware, mejorando la interfaz HMI con visualizaciones más avanzadas y evaluando protocolos de comunicación más seguros, garantizando así la confiabilidad y la adaptabilidad del sistema en aplicaciones reales.

XI BIBLIOGRAFÍA

Mejía, J. (2009). *ANÁLISIS DE VIBRACIONES EN MOTORES ELÉCTRICOS ASÍNCRONOS TRIFÁSICOS*. Guatemala: Biblioteca USAC. Obtenido de http://biblioteca.usac.edu.gt/tesis/08/08_0158_ME.pdf

AG., S. (2025). *IMATIC S7-1500 [página de producto]*. Obtenido de https://www.siemens.com/es/es/productos/automatizacion/sistemas/simatic/controladores-simatic/simatic-s7-1500.html

Aprendiendo Arduino. (2019). *Placas Arduino*. Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/06/19/placas-arduino-2/

arduino. (2018). *¿Qué es Arduino?* Obtenido de https://www.arduino.cc/en/Guide/Introduction/

Arduino. (2025). *¿Qué es Arduino?* Obtenido de https://arduino.cl/que-es-arduino/?srsltid=AfmBOopQdS6_6i2hWZRNEmknAdoFGeu-IjUNdptJKiNYDg8xJUhWgf-q

Arduino. (2025). *Arduino Cloud Maker Plan – 1-Year Subscription*. Obtenido de https://store-usa.arduino.cc/products/cloud-maker-plan-one-year

Arias, K., & Sánchez, R. (2024). *Diseño e implementación de un prototipo para reconocimiento de fallos en rodamientos de motores eléctricos por medio del sonido utilizando Teachable Machine*. DSPACE. Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/29089/1/UPS-GT005781.pdf

Auqui, C., & Farez, C. (2024). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO IOT DE TEMPERATURA Y VIBRACIÓN PARA PROTECCIÓN DE MOTORES ELÉCTRICOS*. Guayaquil: DSPACE UPS. Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/29087/1/UPS-GT005779.pdf

Axmatic. (2025). *SIMATIC S7-1500 – PM 1507 – Siemens Power Supply*. Obtenido de https://axmatic.com/product/simatic-s7-1500-pm-1507-siemens-power-supply

Bee, L. (2022). *PLC and HMI Development with Siemens TIA Portal: Develop PLC and HMI programs using standard methods and structured approaches with TIA Portal V17. Packt Publishing.* Obtenido de https://books.google.com/books/about/PLC_and_HMI_Development_with_Siemens_TIA.html?id=QF5oEAAAQBAJ

Boyko. (2024). The ESP32 Beginner's Handbook: Your Complete Guide to Microcontroller Mastery. MonkMakes Press. https://bookshop.org/p/books/esp32-beginner-s-handbook-your-complete-guide-to-microcontroller-mastery-aden-m-boyle/22816560?utm_source.

Comunitario, S. (2021). *Ondas*. Obtenido de https://servicio-comunitario-45.webnode.es/l/tipos-de-ondas/

Coparoman. (2016). *Arrancador estrella delta para un motor de 12 puntas*. Obtenido de https://coparoman.blogspot.com/2016/10/arrancador-estrella-delta-para-un-motor.html

Domínguez. (2025). *Variadores 101: Técnicas para la instalación, programación y detección de fallas en variadores de frecuencia (VFD)*. Obtenido de https://es.scribd.com/document/667352040/VARIADORES-101

EA-AUTOMATAS. (2024). *Diseño de módulos educativos*. Obtenido de https://ea-automatas.com/diseno-de-modulos-educativos/

Eléctricos, M. (2025). *Variador de frecuencia SINAMICS V20 Siemens*. Obtenido de https://motores-electricos.com.ar/variadores-de-frecuencia/variador-frecuencia-sinamics-v20-siemens/?srsltid=AfmBOoriUkSovwoxxPxiMig6BDektxSpjDiCQxzh_JB_MzogA4fkZcSr

Electrónica. (2022). *Giroscopio MPU6050 con Arduino*. Obtenido de https://www.electronicadiy.com/blogs/tutoriales-y-blog/giroscopio-mpu6050-con-arduino

Electrónicas, D. (2025). *SIMATIC PM 1507 fuente de alimentación estabilizada para SIMATIC S7-1500. 24 VDC/8 A [Ficha técnica]*. Obtenido de https://www.didacticaselectronicas.com/shop/6ep1333-4ba00-simatic-pm-1507-fuente-de-alimentacion-estabilizada-para-simatic-s7-1500-24vdc-8a-21109#attr=5053,5054,5055,5056,5057

Electrostore. (2025). *MÓDULO ESP32 ESP-32 WIFI BLUETOOTH 30 PINES PLUG TIPO C*. Obtenido de https://www.grupoelectrostore.com/producto/modulo-esp32-esp-32-wifi-bluetooth-30-pines-plug-tipo-c/

Electrostore, G. (2025). *Módulo ESP32 ESP-32 WiFi Bluetooth DevKit V1 (30 pines)*. Obtenido de https://www.grupoelectrostore.com/producto/modulo-esp32-esp-32-wifi-bluetooth-devkit-v1-30-pines/

Escuela universitaria de oficios UNLP. (2020). *Motores Eléctricos.* Obtenido de JS Technik: https://unlp.edu.ar/wp-content/uploads/32/33732/cbe4aba99c3a4eccc904dd2c666d1f03.pdf

Evaluando Software. (2025). *Privacidad de datos en la nube ¿Qué analizar antes de migrar a la nube?* Obtenido de Evaluando Software: https://www.evaluandosoftware.com/ciberseguridad/privacidad-datos-la-nube-analizar-migrar-la-nube/

Fajardo, Z., Andino, A., & Fernández, L. (s.f.). *Diseño de un módulo instucional PARA PROMOVER LA EFECTIVIDAD DE LA COMPETENCIA DE COMUNICACIÓN ESCRITA EN LAS ESCUELAS DE NEGOCIOS*. Obtenido de file:///C:/Users/AAA/Downloads/manager,+art4.pdf

Fernández. (2017). *Estudio de las vibraciones*. Obtenido de https://power-mi.com/es/content/estudio-de-las-vibraciones

Fernández, A. (2023). *Motores eléctricos trifásicos: de 9 y 12 terminales (1.ª ed.)*. Obtenido de https://www.amazon.com/-/es/Jes%C3%BAs-Abel-Arriero-Fern%C3%A1ndez-ebook/dp/B0CLKXVW8L

Garatu, G. (2017). *El futuro en el éxito de nuestras empresas está en las bases de datos en la nube*. Obtenido de https://grupogaratu.com/dbaas-bases-datos-la-nube/

GmbH, R. G.-S. (2025). *HMI Siemens – soluciones de caja y brazos portantes para dispositivos HMI SIMATIC de Siemens.* Obtenido de https://rolec.com/es/caja/brazos-portantes-hmi/hmi-siemens

Grajales, J., Ramírez, J., & Cadavid, D. (2004). *El contenido armónico en motores eléctricos: causas, efectos y soluciones.* Obtenido de https://www.redalyc.org/pdf/430/43003110.pdf

how2electronics. (2025). *Getting started with Arduino IoT Cloud with ESP8266*. Obtenido de https://how2electronics.com/getting-started-with-arduino-iot-cloud-with-esp8266/

Kurduban, V. (2024). *Diferencias y relación*. Obtenido de https://es.digi.com/blog/post/edge-computing-vs-cloud-computing?utm_source

Kurniawan, A. (2021). *Beginning Arduino Nano 33 IoT: Step-By-Step Internet of Things Projects. Apress*. Obtenido de https://link.springer.com/book/10.1007/978-1-4842-6446-1?utm_source

López, J. (2024). *Vibraciones en equipos rotativos: Un indicador clave de posibles fallas*. Obtenido de Vibraciones en equipos rotativos: Un indicador clave de posibles

fallas: https://inspenet.com/articulo/vibraciones-equipos-rotativos-como-indicador/

Manufactura Latam. (2024). *Todo sobre sensores de vibración: Qué son, cómo funcionan, tipos y beneficios*. Obtenido de Manufactura Latam: https://www.manufactura-latam.com/es/noticias/todo-sobre-sensores-de-vibracion-que-son-como-funcionan-tipos-y-beneficios

Martínez, C. (2008). *Análisis de Fourier: Series y transformadas. 25 problemas útiles*. Obtenido de https://www.casadellibro.com/libro-analisis-de-fourier-series-y-transformadas-25-problemas/9788493601874/1183586?srsltid=AfmBOop-tFdyN1iJl5garopmtEiOTMQQd9rPV2jgDfrcXB544Fc28SBQ&utm_source

MathWorks. (2021). *The Language of Technical Computing*. Obtenido de https://www.mathworks.com/help/matlab/index.html?utm_source

MathWorks. (2025). *Signal Processing Toolbox*. Obtenido de https://la.mathworks.com/products/signal.html

Megatrónica. (2025). *ESP32S 38 pines WiFi + Bluetooth IoT (ESP32 Wroom)*. Obtenido de https://megatronica.cc/producto/esp32s-38-pines-wifi-bluetooth-iot/?srsltid=AfmBOoqUabXWKIMplWLKFHegG3fVEM-58K6nsbbpBM01GQtPl8IjC45z

Meirovitch. (2001). *undamentals of Vibrations. McGraw-Hill*. Obtenido de https://books.google.com.ec/books/about/Fundamentals_of_Vibrations.html?id=u358QgAACAAJ&redir_esc=y

Mobley. (2002). *An Introduction to Predictive Maintenance (2nd ed.)*. Obtenido de https://doi.org/10.1016/B978-075067531-4/50005-9

Montejo, S. (2007). *eñal en el dominio del tiempo (a) y su respectivo espectro de Fourier (b). En Aplicaciones de la transformada ondícula ("Wavelet") en ingeniería estructural*. Obtenido de https://www.researchgate.net/figure/Figura-4-Senal-en-el-dominio-del-tiempo-a-y-su-respectivo-espectro-de-Fourier-b_fig1_272829611

Motores y Generadores. (2023). *Consecuencias del contenido armónico en motores eléctricos*. Obtenido de Motores y Generadores: https://motoresygeneradores.com/consecuencias-del-contenido-armonico-en-motores-electricos/

Nohuto. (2025). *Módulo MPU-6050 MPU6050 Módulo de Acelerómetro de 3 Ejes Convertidor AD de 16 Bits*. Obtenido de https://www.nohuto.org/producto-p-52094.html

Novatronic Ecuador. (2025). *ESP32 Placa de Desarrollo V1.* Obtenido de Novatronic Ecuador: https://novatronicec.com/index.php/product/esp32-placa-de-desarrollo-v1/

Ortiz, J., & Valencia, F. (2025). *ESP32:MANUAL BÁSICO PARA ESTUDIANTE*. Obtenido de https://alumnieditora.com/omp/index.php/home/catalog/download/9/12/33?inline=1

Peciña, B. (2019). *Programación de controladores avanzados SIMATIC S7-1500 con TIA Portal, AWL/KOP y SCL (3ª ed.).* Obtenido de https://www.libreriajuridicaandaluza.com/libro/programacion-de-controladores-avanzados-simatic-s7-1500-con-tia-portal-awlkop-y-scl_126712

Pruftechnik. (2011). *¿Qué es el análisis de vibraciones?* Obtenido de https://www.pruftechnik.com/es/%C2%BFQu%C3%A9-es-el-an%C3%A1lisis-de-vibraciones-Una-gu%C3%ADa-completa/

Ramírez, E. T. (2024). *DISTORSION ARMONICA.* Obtenido de https://agngroup.net/papers/Distorsion%20Armonica.pdf

Randall. (2011). *Vibration-based condition monitoring: Industrial, automotive and aerospace applications (2nd ed.)*. Obtenido de https://www.wiley.com/en-us/Vibration%2Bbased%2BCondition%2BMonitoring%3A%2BIndustrial%2C%2BAutomotive%2Band%2BAerospace%2BApplications%2C%2B2nd%2BEdition-p-9781119477556

Red Hat. (2022). *¿Qué es la automatización de la nube?* Obtenido de Red Hat: https://www.redhat.com/es/topics/automation-and-management/que-es-la-automatizacion-de-la-nube

Rhoton, S. (2025). *Onda*. Obtenido de https://www.significados.com/onda/

Sandorobotics. (2025). *Módulo Acelerómetro y Giroscopio MPU6050*. Obtenido de https://sandorobotics.com.mx/producto/hr0044/

Siemens. (2014). *BRAUMAT HMI Design Guide. Siemens Industry Online Support*. Obtenido de https://cache.industry.siemens.com/dl/files/364/80142364/att_3056/v1/80142364_braumat_hmi_design_guide_en.pdf

Siemens. (2016). *SINAMICS V20 – Instrucciones de servicio compacto*. Obtenido de https://support.industry.siemens.com/cs/attachments/109736382/SINAMICS_V20_Compact_Operating_Instructions_022016_en-US.pdf

Siemens. (2020). *SINAMICS V20 Operating Instructions (Manual técnico, núm. 109780683)*. Obtenido de https://support.industry.siemens.com/cs/mdm/109780683?c=126956317707&lc=es-AR

Siemens, A. (2025). *HMI Design Workbook. Siemens.* Obtenido de https://www.siemens.com/us/en/products/automation/simatic-hmi/hmi-design-workbook.html

Tecnología, Á. (2025). *Arranque estrella-triángulo de motores trifásicos*. Obtenido de https://www.areatecnologia.com/electricidad/arranque-estrella-triangulo.html

Transfer Multisort Elektronik. (2025). *¿Cómo funciona y qué hace el acelerómetro?* Obtenido de Transfer Multisort Elektronik: https://www.tme.eu/es/news/library-articles/page/22568/Como-funciona-y-que-hace-el-acelerometro/

Universidad Fidélitas. (2024). *Explorando los sistemas embebidos: inteligencia en cada dispositivo.* Obtenido de Universidad Fidélitas: https://ufidelitas.ac.cr/que-son-los-sistemas-embebidos/

Vasco, U. d. (2025). *Transformada de Fourier*. Obtenido de http://www.sc.ehu.es/sbweb/fisica3/datos/fourier/fourier_1.html

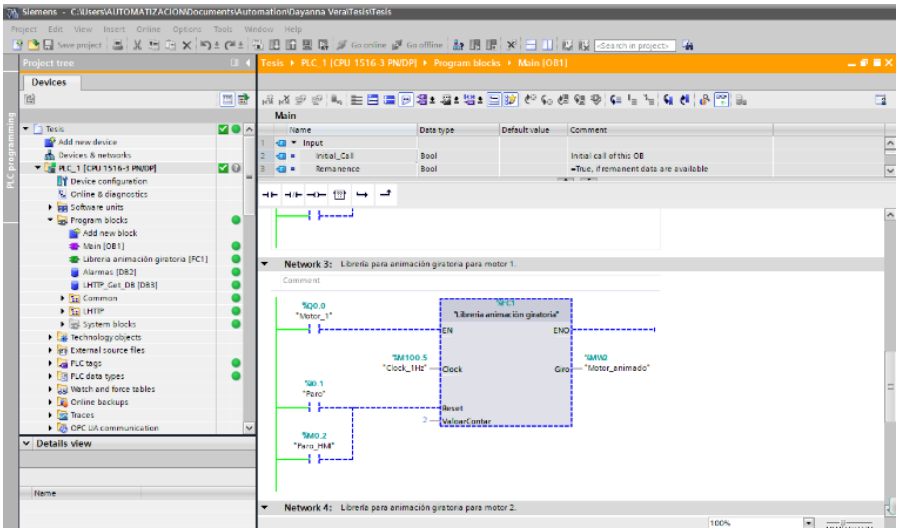
Vasco, U. d. (2025). *Transformada de Fourier*. Obtenido de https://www.sc.ehu.es/sbweb/fisica3/simbolico/fourier/fourier_1.html

Wiki, P. (2025). *ntegridad de la señal: Dominio del tiempo y dominio de frecuencia*. Obtenido de https://wiki-power.com/es

XII ANEXOS

12.1 Programación del TIA Portal

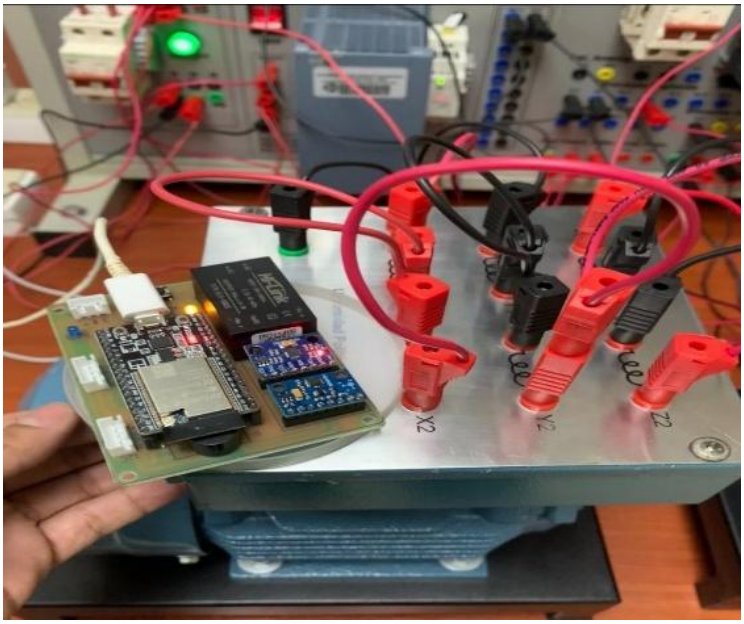
Figura 99
Programación de Motores 1 y 2



Nota. Avance de diagrama bloque para el arranque del motor.

12.2 Pruebas con ESP32

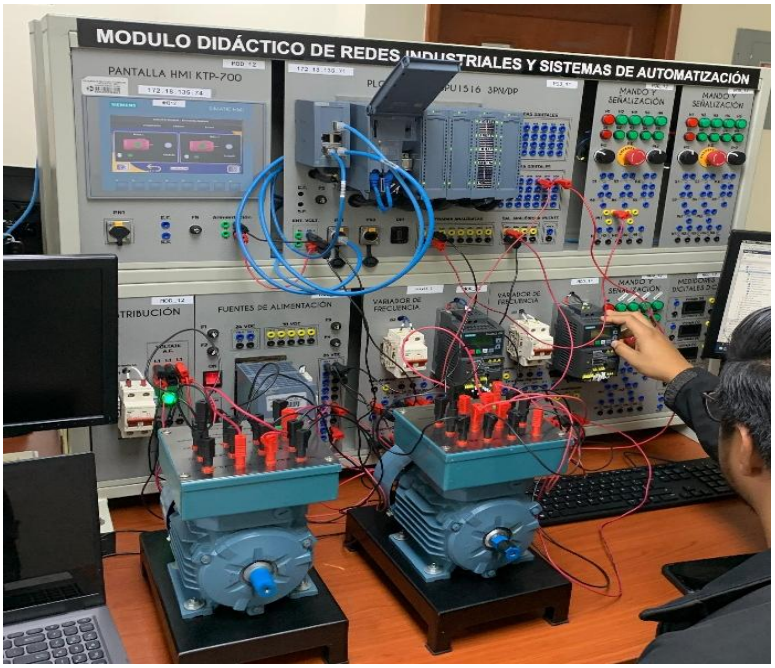
Figura 100
Funcionamiento del sensor



Nota. Pruebas censando las vibraciones del motor.

12.3 Configuración de motores con los variadores

Figura 101
Pruebas de los variadores



Nota. Pruebas conectado los motores con los variadores.

12.4 Comunicación de TIA porta con el PLC

Figura 102
Probando la comunicación del PLC

Nota. Realizando comunicación del PLC.

12.5 Código completo de Arduino

```
#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>
#include <Wire.h>
#include <arduinoFFT.h>

// ============================
// Credenciales Arduino IoT Cloud + WiFi
// ============================
const char DEVICE_LOGIN_NAME[]  = "13dcaa47-132f-4946-a9cf-950d0e87613f";
const char SSID[]               = "Galaxy A14 AAPM";
const char PASS[]               = "Nicolas12345";
const char DEVICE_KEY[]         = "zuhmely6nErZs712s3jov0a9L";

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);

// ============================
// Direcciones / registros I2C
// ============================
#define MPU6050_ADDRESS   0x68
#define REG_PWR_MGMT_1    0x6B
#define REG_ACCEL_XOUT_H  0x3B

#define I2C_SDA 23
#define I2C_SCL 22

// ============================
// Pines PCB
// ============================
```

```
#define BUTTON_PIN  12
#define BUZZER_PIN  21
#define LED_PIN     15

// ============================
// FFT
// ============================
#define SAMPLES           256
#define SAMPLING_FREQUENCY  1000

double vReal[SAMPLES];
double vImag[SAMPLES];
ArduinoFFT<double> FFT(vReal, vImag, SAMPLES, SAMPLING_FREQUENCY);

// ============================
// Variables en IoT Cloud
// ============================
float fundamentalHz  = 0;
float ampFundamental  = 0;
float ampH2 = 0;
float ampH3 = 0;
float vibRMS = 0;

// ============================
// IoT Cloud
// ============================
void initProperties() {
  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
```

```
  ArduinoCloud.addProperty(fundamentalHz,   READ, 2 * SECONDS, NULL);
  ArduinoCloud.addProperty(ampFundamental,  READ, 2 * SECONDS, NULL);
  ArduinoCloud.addProperty(ampH2,           READ, 2 * SECONDS, NULL);
  ArduinoCloud.addProperty(ampH3,           READ, 2 * SECONDS, NULL);
  ArduinoCloud.addProperty(vibRMS,          READ, 2 * SECONDS, NULL);
}

// ============================
// Setup
// ============================
void setup() {
  Serial.begin(115200);
  delay(500);

  Wire.begin(I2C_SDA, I2C_SCL);

  pinMode(BUTTON_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);

  Wire.beginTransmission(MPU6050_ADDRESS);
  Wire.write(REG_PWR_MGMT_1);
  Wire.write(0x00);
  Wire.endTransmission();

  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
```

```
  ArduinoCloud.printDebugInfo();

  Serial.println("Listo: I2C + MPU6050 + FFT + IoT Cloud (Debug)");
}

// ============================
// Loop
// ============================
void loop() {
  ArduinoCloud.update();

  double sum = 0.0;
  static double tbuf[SAMPLES];

  unsigned long t0 = micros();  // medir tiempo real de adquisición

  for (int i = 0; i < SAMPLES; i++) {
    double g = (double)readAccelX_raw() / 16384.0;
    tbuf[i]  = g;
    vReal[i] = g;
    vImag[i] = 0.0;

    sum += g;
    delayMicroseconds((int)(1000000UL / SAMPLING_FREQUENCY));
  }

  unsigned long t1 = micros();
  double fs_real = (double)SAMPLES / ((t1 - t0) / 1e6);

  // RMS AC
```

```
double mean = sum / SAMPLES;
double acc = 0.0;
for (int i = 0; i < SAMPLES; i++) {
  double ac = tbuf[i] - mean;
  acc += ac * ac;
}
vibRMS = sqrt(acc / SAMPLES);

// FFT
FFT.windowing(FFT_WIN_TYP_HAMMING, FFT_FORWARD);
FFT.compute(FFT_FORWARD);
FFT.complexToMagnitude();

fundamentalHz = FFT.majorPeak();

// índice del pico
int idxFund = (int)(fundamentalHz * SAMPLES / fs_real + 0.5);
int maxBin = (SAMPLES / 2) - 1;
if (idxFund < 1) idxFund = 1;
if (idxFund > maxBin) idxFund = maxBin;

ampFundamental = vReal[idxFund];

int idx2 = idxFund * 2;
int idx3 = idxFund * 3;
ampH2 = (idx2 <= maxBin) ? vReal[idx2] : 0.0;
ampH3 = (idx3 <= maxBin) ? vReal[idx3] : 0.0;

// --- Debug extra ---
double freqFromIndex = idxFund * (fs_real / SAMPLES);
```

```
  Serial.print("fs_real: "); Serial.print(fs_real, 2); Serial.print(" Hz | ");
  Serial.print("idxFund: "); Serial.print(idxFund);
  Serial.print(" | majorPeak(): "); Serial.print(fundamentalHz,  2);
  Serial.print(" Hz | freqFromIndex:  "); Serial.print(freqFromIndex,  2);
  Serial.print(" Hz | A0: "); Serial.print(ampFundamental,  3);
  Serial.print(" | RMS: "); Serial.println(vibRMS, 4);

  delay(250);
}

// ============================
// Lectura cruda eje X
// ============================
int16_t readAccelX_raw() {
  Wire.beginTransmission(MPU6050_ADDRESS);
  Wire.write(REG_ACCEL_XOUT_H);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU6050_ADDRESS,  2, true);
  int16_t raw = (Wire.read() << 8) | Wire.read();
  return raw;
}
```

12.6 Código completo de Matlab

```
clear; clc; close all;


fprintf('Leyendo y limpiando datos CSV...\n');


% Función para extraer los primeros Nmax valores numéricos
function data = extractNumericData(filename, Nmax)
    lines = readlines(filename);
    numeric_data = [];


    for i = 1:length(lines)
        line = lines(i);
        if contains(line, ',') && ~startsWith(line, 'time') && ~startsWith(line, 'Text')
            parts = split(line, ',');
            if length(parts) >= 2
                value = str2double(parts{2});
                if ~isnan(value)
                    numeric_data = [numeric_data; value];
                    if length(numeric_data) >= Nmax
                        break;
                    end
                end
            end
        end
```

```
    end
    data = numeric_data;
end

% Extraer primeros 20 datos
y1 = extractNumericData("Motor1.csv", 20);
y2 = extractNumericData("Motor2.csv", 20);

% Número de datos
N = length(y1);

fprintf('✓ Datos extraídos correctamente:\n');
fprintf('Motor 1: %d valores, desde %.4f hasta %.4f\n', length(y1), y1(1), y1(end));
fprintf('Motor 2: %d valores, desde %.4f hasta %.4f\n', length(y2), y2(1), y2(end));

% ==============================
% FRECUENCIA DE MUESTREO
% ==============================
% Si no tienes timestamps, asumimos fs=100 Hz como ejemplo
fs = 100; % <<-- AJUSTA ESTO SEGÚN TU SISTEMA REAL

% ==============================
% FFT
% ==============================
```

```
y1_centered = y1 - mean(y1);
y2_centered = y2 - mean(y2);

Nfft = 1024;
Y1 = fft(y1_centered, Nfft);
Y2 = fft(y2_centered, Nfft);

f = (0:Nfft-1)*(fs/Nfft);    % Vector de frecuencias
mag1 = abs(Y1);
mag2 = abs(Y2);

% ============================
% GRÁFICAS
% ============================
figure('Position', [100, 100, 1500, 500]);

% (1) Espectro FFT normalizado
subplot(1,3,1);
plot(f, mag1/max(mag1), 'b-', 'LineWidth', 2); hold on;
plot(f, mag2/max(mag2), 'r-', 'LineWidth', 2);
xlabel('Frecuencia (Hz)');
ylabel('Magnitud Normalizada');
title('FFT Dominio de la frecuencia');
legend('Motor 1', 'Motor 2');
```

```
grid on;
xlim([0 fs/2]);
xticks(0:10:fs/2);

% (2) Señales en el tiempo
subplot(1,3,2);
t = (0:N-1)/fs;
plot(t, y1, 'b.-', 'MarkerSize', 10); hold on;
plot(t, y2, 'r.-', 'MarkerSize', 10);
xlabel('Tiempo (s)');
ylabel('Valor');
title('Dominios en el Tiempo');
legend('Motor 1', 'Motor 2');
grid on;

% (3) Magnitud compleja tipo MATLAB oficial
subplot(1,3,3);
plot(f, mag1, 'b-', 'LineWidth', 1.5); hold on;
plot(f, mag2, 'r-', 'LineWidth', 1.5);
xlabel('f (Hz)');
ylabel('|fft(X)|');
title('Complex Magnitude of FFT Spectrum');
legend('Motor 1', 'Motor 2');
grid on;
```

```
xlim([0 fs/2]);
xticks(0:10:fs/2);

% ============================
% (4) Frecuencia en el tiempo (simplificado)
% ============================
window = 5;  % tamaño de ventana (número de muestras por análisis)
step = 1;    % avance entre ventanas
freq_dom1 = [];
freq_dom2 = [];
time_axis = [];

for i = 1:step:(N-window)
    seg1 = y1_centered(i:i+window-1);
    seg2 = y2_centered(i:i+window-1);

    % FFT de cada segmento
    Yseg1 = abs(fft(seg1, 256));
    Yseg2 = abs(fft(seg2, 256));
    fseg = (0:255)*(fs/256);

    % Pico dominante (descartando DC en f=0)
    [~, idx1] = max(Yseg1(2:end));
    [~, idx2] = max(Yseg2(2:end));
```

```
    freq_dom1(end+1) = fseg(idx1+1);
    freq_dom2(end+1) = fseg(idx2+1);
    time_axis(end+1) = i/fs;
end

% Gráfica
figure;
plot(time_axis, freq_dom1, 'b.-', 'MarkerSize', 12); hold on;
plot(time_axis, freq_dom2, 'r.-', 'MarkerSize', 12);
xlabel('Tiempo (s)');
ylabel('Frecuencia dominante (Hz)');
title('Frecuencia en el tiempo (simplificado)');
legend('Motor 1','Motor 2');
grid on;
```