UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Diseño y desarrollo de sistema de detección temprana de fallos para un motor eléctrico

Memoria para optar al título de Ingeniero Civil en Computación

Dante Antu Mardones Huala

Profesor Guía:
Luciano Radrigan Figueroa

Profesor Co-guía:
Alejandro Hevia Angulo

Comisión:
Andrés Abeliuk Kimelman

Santiago de Chile
2024

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL EN
COMPUTACIÓN
POR: DANTE ANTU MARDONES HUALA
FECHA: 2024
PROF. GUÍA: LUCIANO RADRIGAN

DISEÑO Y DESARROLLO DE SISTEMA DE DETECCIÓN TEMPRANA DE FALLOS PARA UN MOTOR ELÉCTRICO

El objetivo de este trabajo es diseñar e implementar un sistema de monitoreo y mantenimiento predictivo para motores eléctricos, aplicando técnicas de Machine Learning para la generación de diagnósticos y pronósticos.

Normalmente para este tipo de problemas, un equipo de expertos se encarga de analizar los datos generados por el motor. Sin embargo, tener personal con este perfil no es sencillo, ya que implica tener una amplia experiencia y conocimiento sobre la naturaleza del problema. Dada la naturaleza del problema, se propone aplicar modelos de clasificación de ML para determinar si el motor presenta un correcto funcionamiento o un desbalance/problema.

El sistema propuesto abarca todo el proceso del problema, es decir; la recolección de datos, el guardado de datos, el procesamiento de los datos para generar diagnósticos y pronósticos, además de la visualización de los resultados.

Se desarrollaron tres módulos; recolección de datos, procesamiento de datos y visualización de datos. Los cuales funcionan de manera independiente y se comunican entre si para traspasar la información.

Para la recolección de datos se utilizó un sensor de inercia BMI270 junto a un microcontrolador ESP32, este envía los datos recolectados mediante TCP/IP a un servidor hecho en Python, para ser guardados y procesados por el clasificador Support Vector Classifier. Luego, se envían los resultados mediante WebSockets a una interfaz desarrollada usando el framework React, con el lenguaje de programación Javascript.

Finalmente, se logró desarrollar un sistema con capacidades de monitoreo y diagnósticos en tiempo real. Se logró cumplir con el objetivo de diagnosticar y presentar datos en tiempo real con una interfaz simplificada con datos clave. Los diagnósticos entregados tienen un alto nivel de confiabilidad gracias al modelo elegido, aunque no se alcanzó la confiabilidad esperada, según se detalla a continuación.

Agradecimientos

Por este trabajo quiero agradecer a varias personas que me apoyaron de distintas formas en el proceso. A la Francys por su compañía y constancia en los *horarios de tesis*. A Neichon por preocuparse de mi progreso y ayudarme con los trámites. Al Mati por apoyarme en el camino y responderme dudas. A la Javiera por su compañía en general.

Por mi paso a en la universidad, agradecerle a las personas que conocí en el Eolian ya que con ellos pasé la mayor parte de mi estadía en la universidad y me ayudaron a crecer en el ámbito profesional y personal.

A mi familia, en especial a mis hermanos que siempre me apoyan y me aconsejan.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

La electromovilidad es un tema que cada día toma más fuerza en Chile, según el informe “Estrategia Nacional de Electromovilidad” [4] elaborado en 2017, un tercio del consumo energético en chile se debe a transporte y de esa parte un 98 % corresponde a derivados del petróleo. Esto, sumado a que se tiene como meta al 2050 contar con un 100 % de transporte público y un 40 % de vehículos particulares abastecidos por energía eléctrica, implica que existe potencial de crecimiento con respecto a los motores eléctricos en Chile, ya que se puede inferir que gran parte del transporte va a tener que ser reemplazado o reconvertido a una alternativa que utilice motores eléctricos.

Un motor eléctrico combina un sistema eléctrico y mecánico (es decir, un sistema electromecánico), por esto, prevenir los distintos tipos de fallos de un motor se vuelve un problema complejo, se requiere personal con expertiz sobre las distintas áreas y equipos especializados para hacer diagnósticos o mediciones sobre el motor.

Existen varias causas de fallo en un motor, para dar algunos ejemplos; el sobrecalentamiento y/o exceso de voltaje, desgaste de rodamientos, fallas en la insulación de los cables dentro del motor, entre otros [6]. En el contexto de este trabajo, solo se considerarán los fallos relacionados a información inercial y de variables ambientales relacionadas al motor.

Actualmente existen dos principales enfoques de mantención, el mantenimiento correctivo y el mantenimiento preventivo. Que en resumen, constan de reparar algo cuando falla, y tratar de reparar algo antes de que falle, respectivamente. En este trabajo se utilizará principalmente mantenimiento preventivo, en particular se aplicará mantenimiento predictivo, en donde se predice en cuanto tiempo se va a presentar un fallo, además de entregar un monitoreo de las variables generales del motor, todo en tiempo real.

El objetivo de este trabajo es desarrollar un sistema de bajo costo y fácil instalación, cuyo propósito se basa en la detección temprana de fallos para motores eléctricos, utili-

zando algoritmos de Machine Learning (ML), Internet of Things (IoT), sensores inerciales y sensores de variables ambientales para la caracterización del funcionamiento del motor. El sistema contempla la recolección, el envío, el procesamiento y la visualización de los datos. Ofreciendo un sistema completo y simple para medir las condiciones de fallo y el monitoreo de un motor.

Capítulo 2

Objetivos

2.1. Objetivo general

Este trabajo busca diseñar e implementar un sistema de detección temprana de fallos y monitoreo en tiempo real del estado de un motor eléctrico. De forma más específica, el elemento estudiado es un motor trifásico modelo M2BAX de fabricante ABB. Este motor tiene una potencia de 0.52KW y llega a las 1000 revoluciones por minuto. Para el accionamiento se utiliza un variador de frecuencia modelo ACS150 de fabricante ABB. Este variador utiliza una fuente de alimentación de entrada monofásica y salida trifásica.

2.2. Objetivos específicos

1. Montar un sistema con el motor y los sensores para generar datos históricos sobre su funcionamiento con y sin fallas.

2. Asegurar integridad de los datos, tanto en el proceso de recolección como en el de transmisión.

3. Elegir algún algoritmo de ML capaz de generar diagnósticos confiables, tomando los datos del problema estudiado. Se definen diagnósticos confiables como resultados que alcancen un 98 % de precisión, valor obtenido como referencia del trabajo estudiado [13]: Este objetivo se subdividió en dos metas secuenciales. Primero desarrollar un modelo capaz de diagnosticar el estado del motor en tiempo real (es decir, lograr CM y diagnósticos del estado actual) y como segunda meta, poder pronosticar valores de aceleración para el futuro.

   Combinando ambos se logra un pronóstico de los datos en el futuro, generando diagnósticos sobre los datos predichos por la segunda meta.

4. Mostrar el estado general del motor y los pronósticos hechos sobre el mismo de una forma intuitiva para el usuario en el servidor.

Capítulo 3

Estado del arte

Uno de los ejes principales de la revolución industrial 4.0 es maximizar la eficiencia de los procesos. En el contexto de componentes y/o maquinarias significa utilizarlos/las el mayor tiempo posible, maximizando la vida útil (*Remaining Useful Life*, RUL) de cada componente y en consecuencia, reduciendo los tiempos de mantención.

Cuando se habla de mantenimiento se puede separar en dos grandes áreas [5] [9]:

- Corrective maintenance: Esperar que algo falle para repararlo.
- Predictive maintenance (PdM): Entregar monitoreo en tiempo real y predecir en cuanto tiempo algo va a necesitar mantención.

Dependiendo del tipo de problema se aplican mezclas de ambos enfoques. Aunque los dos tienen como base el *Reliability Centered Maintenance* (RCM) [3] [5], básicamente priorizar el monitoreo y la mantención sobre las partes más críticas de un sistema.

La industria está adoptando cada vez más soluciones de PdM para disminuir costos. Por otro lado, las soluciones de PdM están mejorando y adoptando cada vez más responsabilidades.

El PdM consiste en generar diagnósticos y pronósticos sobre la maquinaria usando los datos generados por la misma. Con el objetivo de determinar el estado actual de funcionamiento y en cuanto tiempo se requiere hacer una mantención.

Para el procesamiento de los datos se utilizan varias opciones, como por ejemplo, métodos probabilísticos, redes neuronales, técnicas de procesamiento masivo de datos, técnicas de procesamiento de señales, aprendizaje de máquina, algoritmos de agrupamiento, etc [3] [11].

Para abordar problemas de mantenimiento sobre motores eléctricos, uno de los acercamientos principales es el análisis de vibraciones para una ventana de tiempo. De esa

ventana de datos se pueden observar tendencias sobre el funcionamiento del motor, ya que por ejemplo si un motor esta des balanceado, presenta una determinada frecuencia de vibración. [11]

El análisis sobre la ventana de datos se suele hacer usando las aceleraciones directamente o la transformada de Fourier de la misma ventana. Cada vez se adopta más utilizar machine learning (ML) para esta etapa, ya que califica como un problema donde se buscan patrones en los datos para determinar un diagnóstico.

Dentro del ML, se pueden encontrar distintas categorías de aplicaciones, como se puede ver en la figura 3.1, existe el ML con:

- Aprendizaje supervisado: Cuando se conoce que debería entregar el modelo al recibir los datos. A su vez, este se puede subdividir en problemas de; clasificación, cuando la salida del modelo corresponde a estados discretos y regresión, cuando la salida del modelo entrega resultados dentro de un rango de valores.
- Aprendizaje no supervisado, cuando no se conoce que debería entregar el modelo al recibir los datos. Este enfoque se puede subdividir en problemas de agrupamiento y asociación.

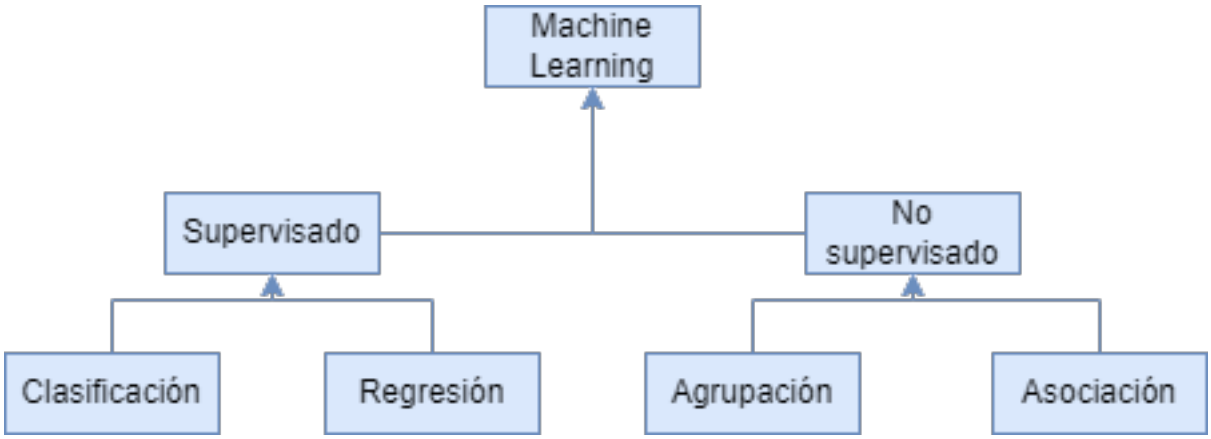

Figura 3.1: Tipos de ML

Para este trabajo se cuentan con datos sobre las vibraciones del motor y su estado de funcionamiento (balanceado o des balanceado).

Generar diagnósticos es un problema de aprendizaje supervisado de clasificación, ya que se reciben los datos de vibración y se entrega el estado diagnosticado. Algunos algoritmos de clasificación comunes para este tipo de problemas, son Naive Bayes (NB), Random Forest (RF), Support Vector Machine (SVM), K-Nearest Neighbors (KNN) [12].

Para este trabajo se utilizaron:

- Support Vector Classifier (SVC)
- KNN.
- NV
- Decision Tree.
- Gradient Boosting.
- RF.

Para generar pronósticos, un enfoque común es predecir las señales de vibración en el tiempo y luego generar diagnósticos sobre esos datos. Esto se puede hacer con redes neuronales como Long Short Term Memory (LSTM) [13] [12].

Con respecto a trabajos similares, se estudió un trabajo [10] donde se detalla la implementación de un sistema de monitoreo para un motor, utilizando machine learning. En él, se alcanza una precisión de diagnósticos del 100 % usando RF, 99 % usando NB, 98 % usando KNN, 84 % usando SVM y 82 % usando regresión lineal. Como referencia para este trabajó, se determinó como objetivo el 98 % alcanzado por KNN.

Capítulo 4

Sistema propuesto

El sistema propuesto aborda todo el proceso desde la lectura de datos por parte del sensor de inercia hasta la visualización de los resultados obtenidos en una aplicación web. Para el desarrollo completo se separó la solución en módulos independientes que se comunican entre ellos, como se puede ver en la figura 4.1 estos sistemas son: Recolección de datos; Procesamiento de datos; Visualización de resultados.

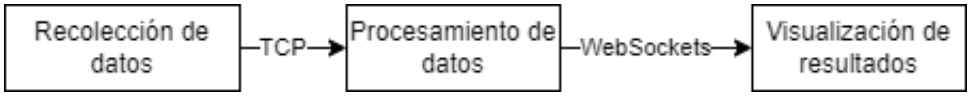

Figura 4.1: Sistema general.

A continuación se presenta en detalle cada módulo.

4.1. Recolección de datos

El objetivo de este módulo es hacer lecturas de aceleración del motor y dejarlas disponibles para el servidor donde se procesarán los datos.

Se dividió esta tarea entre 2 responsables, el emisor, quien recolecta los datos y se los envía al servidor de procesamiento, y el receptor, quien recibe los datos y los almacena en una base de datos para que el servidor de procesamiento pueda acceder a ellos.

El emisor consta de un microcontrolador ESP32, debido a su capacidad de conexión mediante los protocolos I2C y WiFi, y un sensor de inercia BMI270, compatible con el protocolo I2C. El sensor va acoplado a la superficie del motor mientras está conectado por I2C a la ESP32. En la tabla 4.1 y la figura 4.2 se muestran las conexiones utilizadas para comunicar el sensor y la ESP32 mediante I2C. La ESP32 puede estar en cualquier lugar, mientras que su cableado no estorbe el funcionamiento del motor. La ESP32 consulta los datos leídos por el BMI270 y luego los envía mediante IP/TCP al receptor.

Tabla 4.1: Conexiones I2C entre ESP32 y BMI270.

Pin	ESP32	BMI270
GND	GND	SDO
V	3v3	3.3v
SDA	22	SDA
SCL	21	SCL

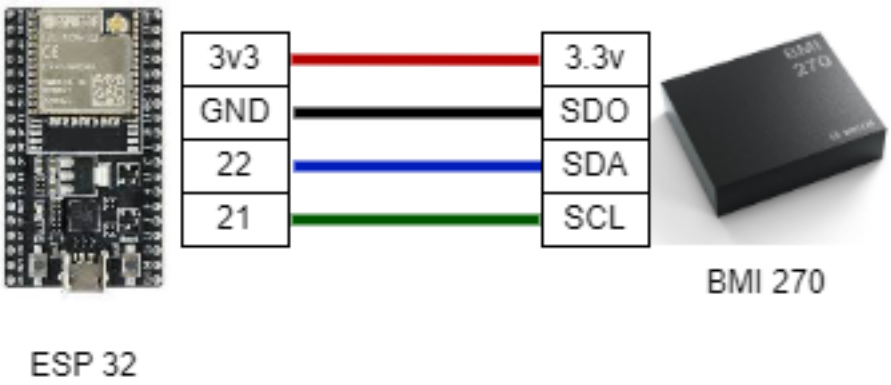

Figura 4.2: Cableado entre ESP32 y BMI270

Con respecto a su funcionamiento, al inicio la ESP32 establece una conexión con el receptor e inicia su reloj, luego comienza su ciclo de lectura de datos. Para leer datos la ESP32 le consulta al sensor (mediante I2C) si hay nuevos datos disponibles, si la respuesta es afirmativa, se sigue el procesamiento descrito en la figura 4.3. Es decir, el BMI270 devuelve una lista de 12 bytes, de los cuales 6 corresponden a las aceleraciones en los distintos ejes (x,y,z) y 6 corresponden a la velocidad angular en los distintos ejes. Se utilizan solo los valores asociados a aceleraciones, la ESP32 crea un paquete con estos datos y le agrega 2 valores; el tiempo en donde se recibió la lectura que corresponde a un epoch de Unix guardado en 8 bytes; y el número de la medición (comienza en 0) guardado en 4 bytes. Finalmente envía ese paquete de 24 bytes al servidor TCP con quién estableció una conexión inicialmente, es decir, el receptor. Este proceso ocurre 1600 veces por segundo, es decir, que se tiene una frecuencia de lectura de 1600 Hz, se hace énfasis en este número ya que tendrá relevancia en todo el resto de los módulos.

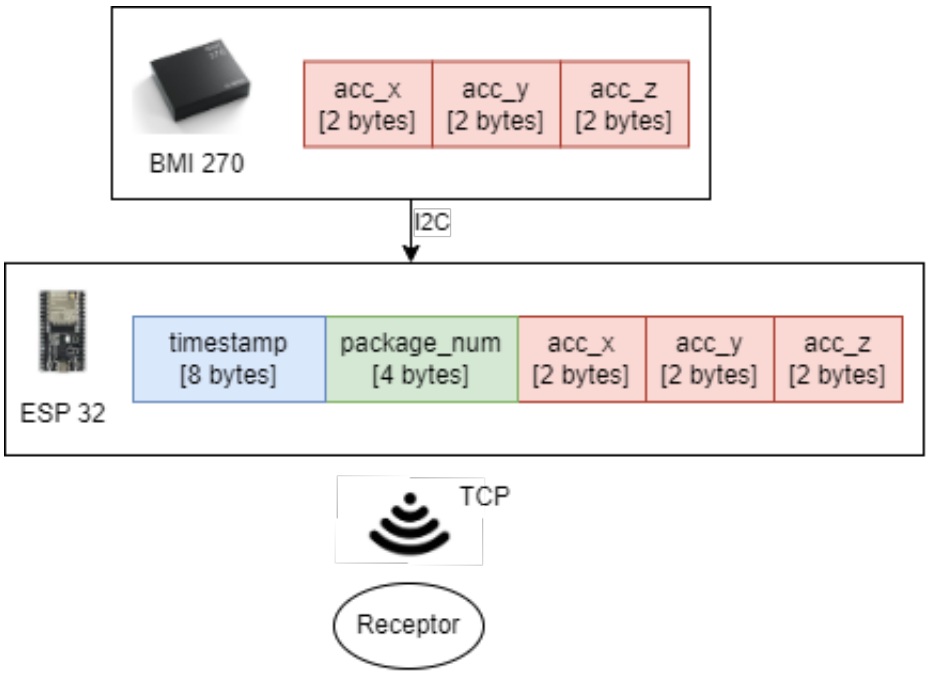

Figura 4.3: Funcionamiento del emisor

El receptor consta de un script desarrollado en Python. Como se puede ver en la figura 4.4, su responsabilidad es recibir los datos del emisor mediante TCP, procesarlos a magnitudes conocidas y guardarlos en una base de datos. Este script debe ser ejecutado en un ambiente compatible con TCP y con permisos de lectura y escritura a la base de datos.

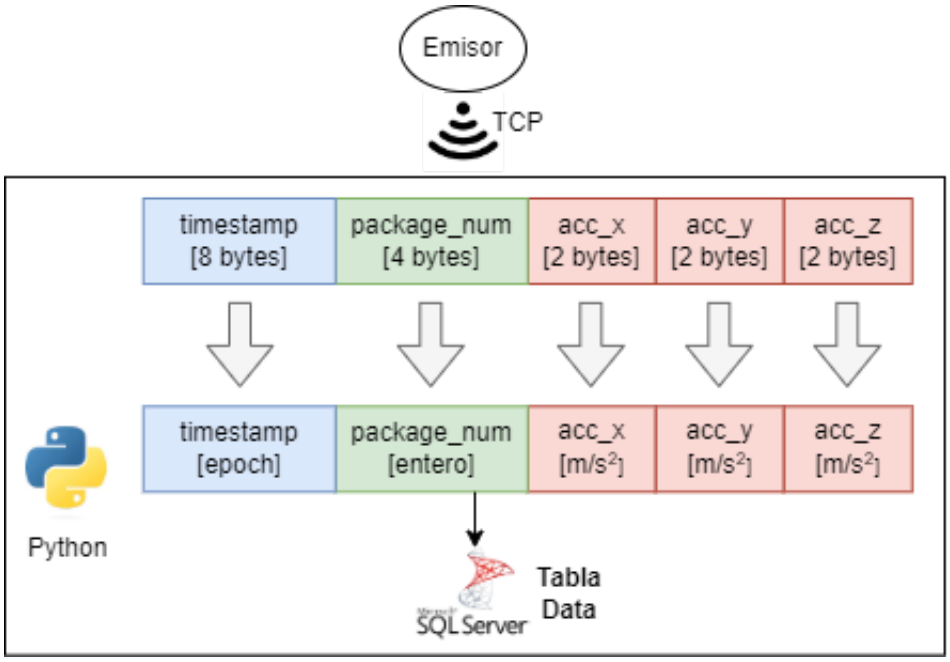

Figura 4.4: Funcionamiento del receptor

El receptor recibe el paquete mediante TCP, que consta de una lista de 24 bytes. Para

interpretar los valores del paquete se hace lo siguiente; el tiempo de lectura corresponde a un natural de 8 bytes, el cual representa un tiempo en epochs; el número de paquete es un entero de 4 bytes; cada aceleración se interpreta como un entero de 2 bytes (usando la codificación de complemento de 2) aplicándolas a la fórmula 4.1.

$$A = A_b * \frac{RA * g}{c} \tag{4.1}$$

Donde RA representa el rango de las aceleraciones medidas, en este caso $RA = 2$ (dada por la configuración del sensor), g representa la constante de gravedad, c es la constante 32769, A_b representa el valor de la aceleración en bytes. Todo esto entrega A, que representa el valor de la aceleración en $\frac{m}{s^2}$, esta fórmula viene dada por el documento de ficha técnica del sensor.

El receptor guarda los datos procesados en la base de datos, en una tabla llamada Data, cuyo esquema se puede ver en la figura 4.5.

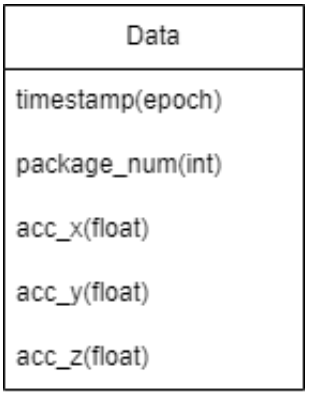

Figura 4.5: Esquema de guardado de datos

Donde timestamp, package_num, acc_x, acc_y, acc_z corresponden a la hora de la lectura, el número de la medición, la aceleración en el eje x, aceleración en el eje y, aceleración en el eje z, respectivamente.

La base de datos utilizada fue SQL Server 2022 Developer porque al ser relacional cumplía con las necesidades del problema y se tenía familiaridad trabajando con ella.

4.2. Procesamiento de datos

El desarrollo de este módulo consiste en 2 partes: definir y entrenar un modelo de ML para generar diagnósticos; y desarrollar un software que aplique el modelo sobre los datos recibidos y que envié los resultados obtenidos al módulo de visualización.

4.2.1. Modelo para generar diagnósticos

Este es un problema de clasificación binaria donde se tienen datos etiquetados. Por lo que se consideraron algoritmos de clasificación para aprendizaje supervisado.

Para el problema se facilitaron datos etiquetados descritos en la tabla 4.2, que constan de 540800 mediciones (310400 mediciones balanceadas y 230400 des balanceadas), a 1600 Hz, es decir a una velocidad de 1600 mediciones por segundo. Se usaron datos del motor funcionando a distintas frecuencias de giro, cada medición consta de filas como las descritas en la tabla 4.3; las aceleraciones en los tres ejes (x, y, z) como un número real entre distintos intervalos (especificados en la tabla 4.3), el tiempo de la medición como un epoch unix, el número de la medición como un entero y el estado del motor (balanceado o des balanceado). Cuando se dice que el motor está balanceado, se refiere a que está en correctas condiciones de funcionamiento, mientras que cuando está des balanceado, significa que presenta algún defecto/error de funcionamiento que está causando el des balance.

Tabla 4.2: Datos utilizados para entrenar los modelos

	Datos etiquetado como balanceados	Datos etiquetados como des balanceados
Cantidad	310400	230400

Tabla 4.3: Descripción de los registros utilizados para entrenar los modelos

acc_x	acc_y	acc_z	timestamp	n	estado
[-3, 3]	[-1, 1]	[-7,7]	epoch unix	entero	{balanceado,des balanceado}

Siguiendo la estrategia común de analizar una ventana de datos, el problema se planteó como, dada una ventana de mediciones, diagnosticar si el motor está balanceado o des balanceado. Para los experimentos se fue variando el tamaño de esta ventana, desde 50 hasta 1600, en intervalos de 50, es decir, 50, 100, 150, ..., hasta 1600.

Para el entrenamiento de los modelos, se utilizó como entrada 2 opciones; una lista con todas las mediciones de la ventana a analizar (utilizando directamente los valores de aceleración) y la transformada rápida de Fourier de la misma ventana.

Para ambas opciones, se comparó el desempeño de los algoritmos más utilizados para este tipo de problemas como; Support Vector Machine, Random Forest, Naive Bayes, Gradient Boosting además de otros utilizados para problemas de aprendizaje supervisado como: K-neighbors y Decision Trees.

Para comparar los distintos algoritmos, se utilizó la métrica del puntaje F1 (F1 score). El cálculo de este puntaje esta dado por la fórmula 4.2

$$F1 = \frac{2 * VP}{2 * VP + FN + FP} \tag{4.2}$$

donde VP representa los verdaderos positivos, FP los falsos positivos y FN los falsos negativos, se eligió esta métrica por sobre precisión, ya que considera todos los casos mencionados.

Como en este problema existen 2 posibles etiquetas, se utilizó la variante del puntaje F1 ponderado (*weighted*), que calcula el puntaje F1 para cada clase y luego lo pondera por la cantidad de datos de cada clase. El cálculo final del puntaje utilizado se puede ver en la fórmula 4.3.

$$F1 = \frac{F1_b + F1_u}{T} = \frac{\frac{2*VP_b}{2*VP_b+FN_b+FP_b} * V_b + \frac{2*VP_u}{2*VP_u+FN_u+FP_u} * V_u}{T} \tag{4.3}$$

Donde T representa el total de datos, V representa el número de etiquetas presentes inicialmente del estado respectivo, el subíndice b hace referencia al estado balanceado y u al estado des balanceado.

Para el entrenamiento y validación de los modelos, se utilizó *k-fold cross validation* con un valor de 5 para k. Es decir, separando el conjunto de datos en 5 grupos para cada validación.

Como se mencionó anteriormente, para entrenar el modelo se probó usando la ventana de aceleraciones y luego la fft de una ventana de aceleraciones. Además para cada opción se probó entrenar los modelos con los valores originales de aceleración y normalizando los valores entre 0 a 1.

Para el proceso de normalización se tomó el rango de valores original y se llevó a escala al rango entre 0 y 1. Es decir, para cada valor a se le aplicó la siguiente transformación:

$$\delta = (a - a_m)/(a_M - a_m) \tag{4.4}$$

$$A = \delta * (1 - 0) + 1 \tag{4.5}$$

Donde a_M representa el máximo valor en el intervalo de a, a_m representa el mínimo valor en el intervalo de a y A es el valor de a normalizado.

A continuación se muestran los resultados obtenidos para cada modelo y tipo de entrada de datos. En la figura 4.6 están los resultados procesando las aceleraciones normalizadas y en la figura 4.8 están los resultados procesando las aceleraciones sin normalizar. Luego en la figura 4.7 se muestran los resultados recibiendo la fft de las aceleraciones normalizadas y finalmente, en la figura 4.9 se muestran los resultados al procesar la fft de las acerelaciones sin normalizar.

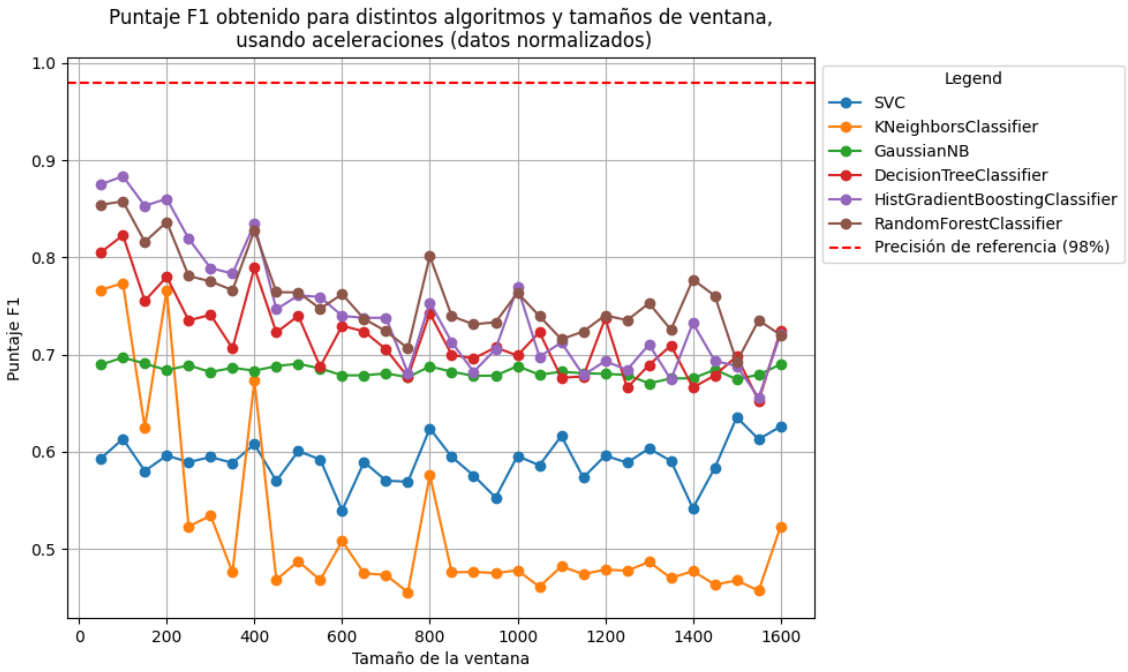

Figura 4.6: Resultados usando aceleraciones normalizadas

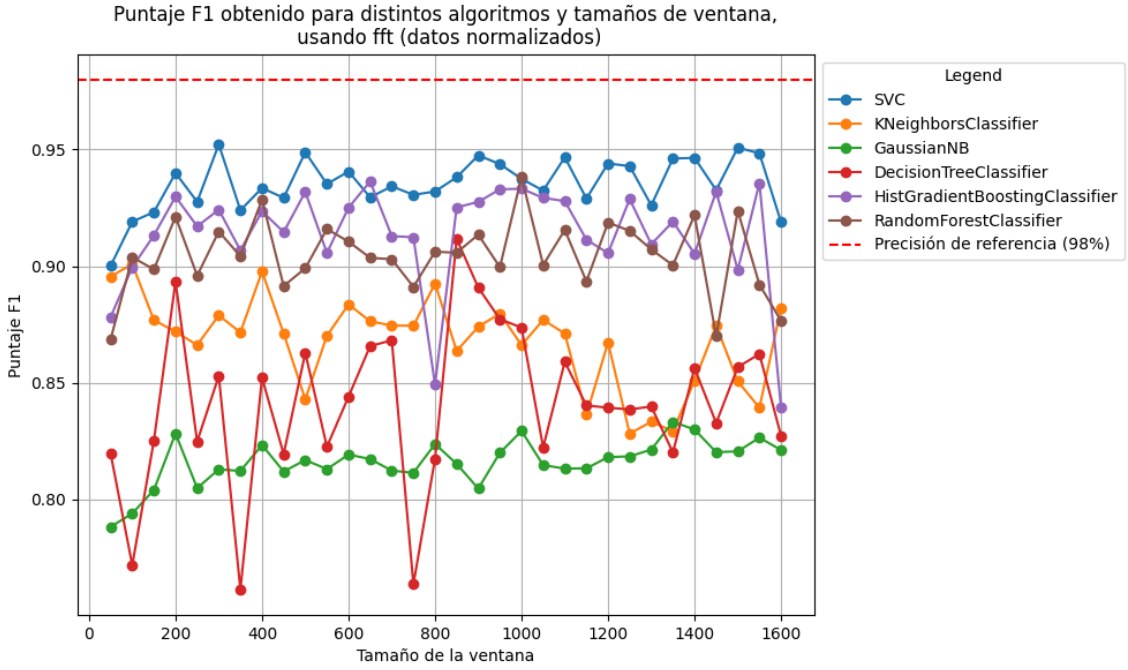

Figura 4.7: Resultados usando la transformada rápida de Fourier de aceleraciones normalizadas

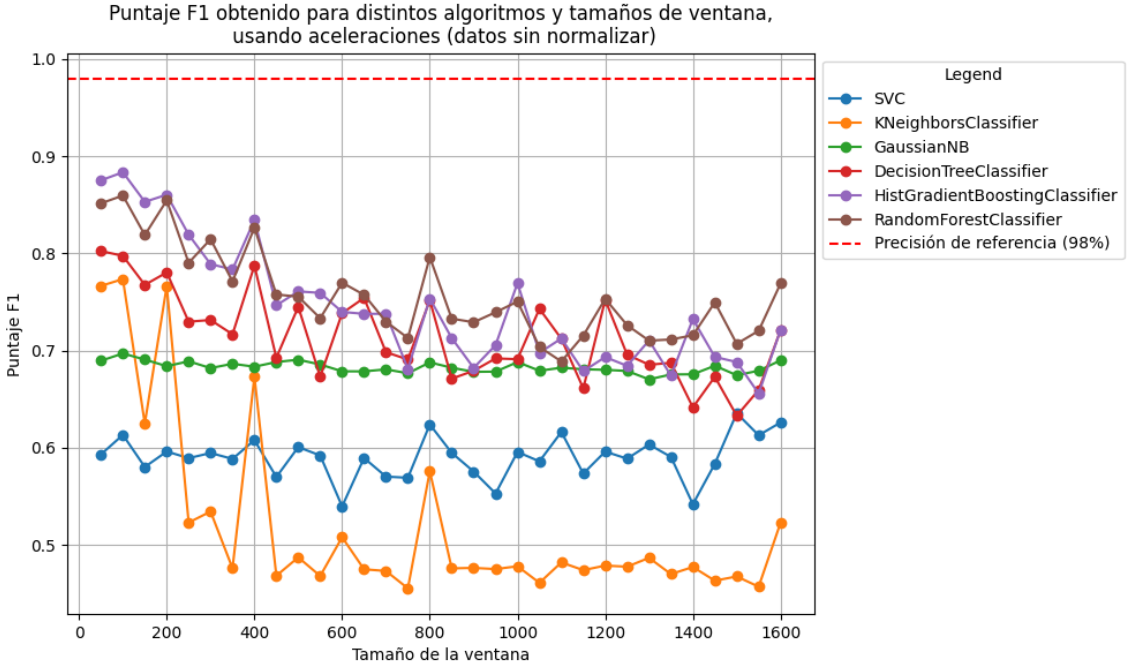

Figura 4.8: Resultados usando aceleraciones sin normalizar

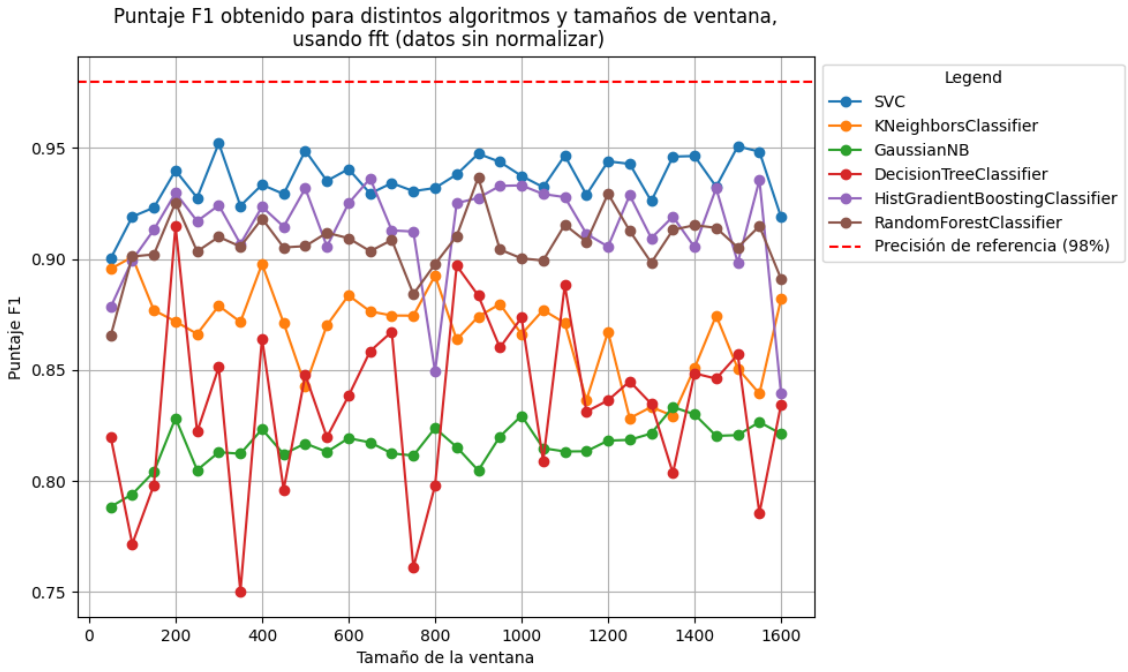

Figura 4.9: Resultados usando la transformada rápida de Fourier de aceleraciones sin normalizar

A partir de los gráficos se puede ver que se obtuvieron mejores resultados en general al procesar la transformada rápida de Fourier de las ventanas. Ya que la mayoría entregaba resultados por sobre el 75 %. Sin embargo, no se alcanzó el valor objetivo de referencia.

Los tres modelos que entregaron mejores resultados fueron SVC, Boosting y RF. Se puede deducir que Boosting y RF entregaron mejores resultados dada su naturaleza de algoritmos de ensamble, que los hace más robustos en general. Mientras que para SVC se puede deducir que funciona bien ya que al aplicar la transformada rápida de Fourier sobre los datos de aceleración, se elimina la no linealidad de las señales de aceleración y esto permite modelos como SVC entreguen mejores resultados.

Considerando los tres modelos mencionados, su comportamiento no cambio significativamente en el caso de la normalización o no normalización de las aceleraciones. En el caso de SVC, que es el que entregó mejores resultados en general, no cambió significativamente sus resultados para distintos tamaños de ventana.

Por lo mismo se decidió utilizar Support Vector Classifier para ventanas de tamaño 800, se eligió SVC porque presenta un mejor rendimiento independiente del tamaño de la ventana escogida.

Los experimentos fueron ejecutados usando Python y las implementaciones de cada

algoritmo fueron usadas de la librería scikit learn. Finalmente se exporta el modelo ya entrenado para ser usado por la siguiente etapa de este módulo.

4.2.2. Procesamiento y envío de datos

El objetivo de esta etapa es generar diagnósticos usando el modelo ya entrenado y enviar aquella información y otras variables de monitoreo al módulo de visualización. Consiste de un código desarrollado en Python, que tenga permisos de lectura a la base de datos. Este se comunica mediante WebSockets con el módulo de visualización.

Cada medio segundo, el programa extrae la última ventana disponible de 800 mediciones de aceleración de la base de datos, les calcula la transformada rápida de Fourier y se la entrega al modelo. Luego genera un arreglo con el diagnóstico generado y las mediciones analizadas y lo envía al módulo de visualización por medio de WebSockets. Este proceso se puede ver en la figura 4.10

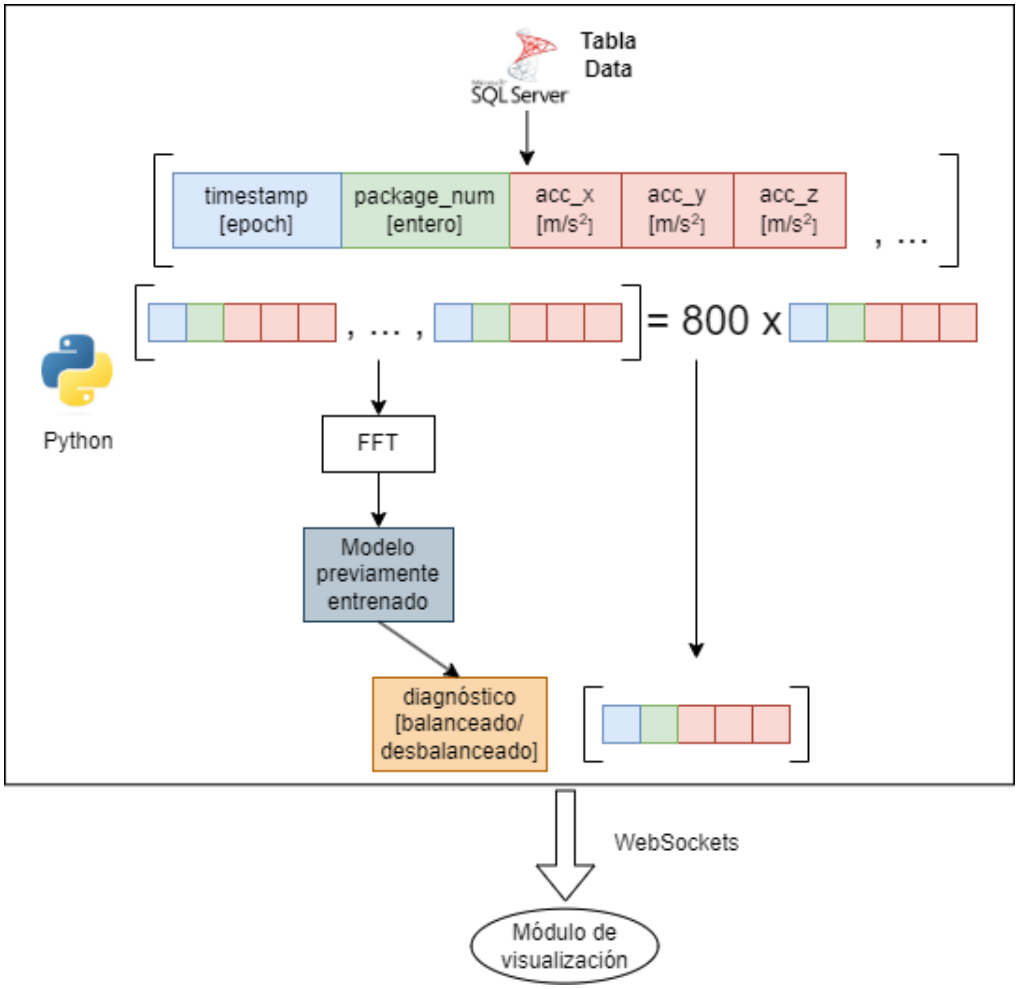

Figura 4.10: Funcionamiento del módulo de procesamiento

Cabe mencionar que se utilizó una comunicación por WebSockets debido a su capacidad de transmitir información con muy baja latencia, algo ideal para sistemas como este que deben funcionar en tiempo real.

4.3. Visualización de resultados

El objetivo de este módulo es recibir los diagnósticos generados y los datos leídos para luego mostrárselos al usuario. Consiste de una aplicación web escrita en React (javascript) que utiliza WebSockets para recibir los elementos a mostrar.

El funcionamiento de este módulo se puede ver en la figura 4.11. Consiste en esperar notificaciones de WebSockets del módulo de procesamiento. Estas llegan aproximadamente cada medio segundo (notar que este tiempo es el mismo tiempo que abarca la última ventana de mediciones analizada), y contienen la última ventana de 800 mediciones analizada junto al diagnóstico asociado a la misma, elementos como la transformada de Fourier o RMS son procesados una vez llegan al cliente de visualización, esto se diseño así para no colapsar el ancho de banda de la comunicación por WebSockets. Cada vez que recibe una actualización por el WebSocket, actualiza los gráficos correspondientes de la interfaz.

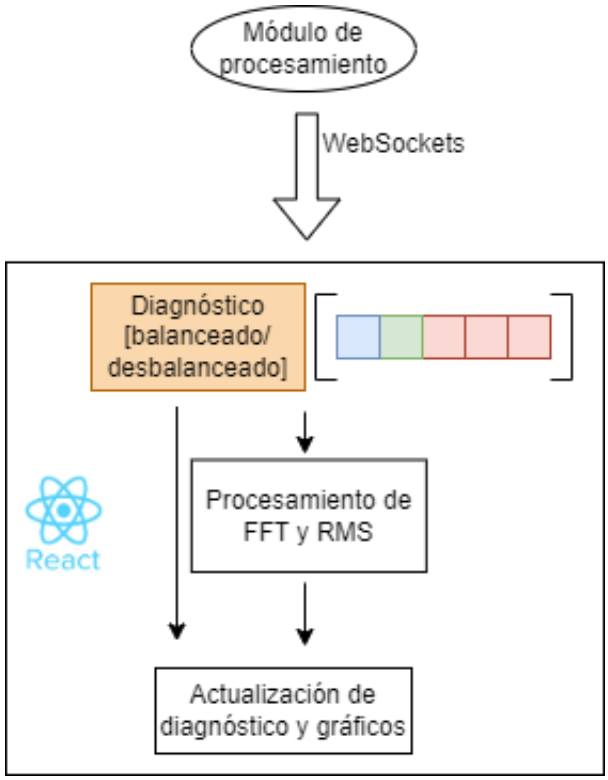

Figura 4.11: Funcionamiento del módulo de visualización

A continuación se explica más en detalle como se presentan los resultados y como se comporta cada elemento al actualizar sus datos. Se harán las descripciones en orden

de aparición de los distintos elementos en la página, desde arriba a abajo. La interfaz completa se puede ver en la figura 6.1 del anexo.

La idea detrás de esta interfaz, es ofrecer de forma rápida y simple el diagnóstico generado, y en el caso de que el usuario quiera indagar en más detalle sobre la razón detrás del diagnóstico, que pueda hacerlo visualizando los datos de forma más directa.

En la parte superior se presenta un cuadro con el estado actual del motor, que cambia a rojo si se nota un des balance (como se ve en la figura 4.12) y a verde si el motor está funcionando correctamente (como se ve en la imagen 4.13. Este se actualiza cada vez que se recibe una notificación. Este elemento se muestra primero ya que es lo más importante de toda la visualización.

Figura 4.12: Visualización de diagnóstico des balanceado

Figura 4.13: Visualización de diagnóstico balanceado

Luego, como se ve en la figura 4.14, se presenta un gráfico de linea que muestra las aceleraciones en los 3 ejes para una ventana de tiempo de 10 segundos. Por cada nueva notificación este gráfico desplaza la ventana de tiempo que abarca hacia adelante en X segundos. Este elemento se ubicó como el segundo más importante ya que es el estado más directo que se puede presentar sobre el sistema.

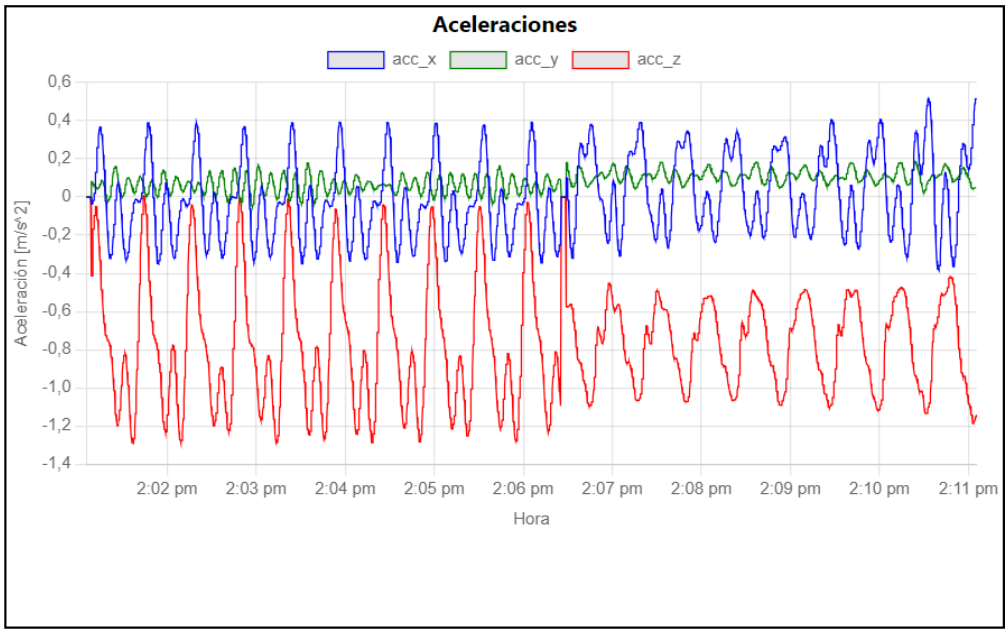

Figura 4.14: Gráfico de aceleraciones en los tres ejes

Le sigue otro gráfico de linea que muestra la RMS de las aceleraciones, funciona de la misma manera que el gráfico anterior, pero tiene el propósito de poder entregar información sobre las tendencias de las aceleraciones de forma rápida. Este se puede ver en la figura 4.15.

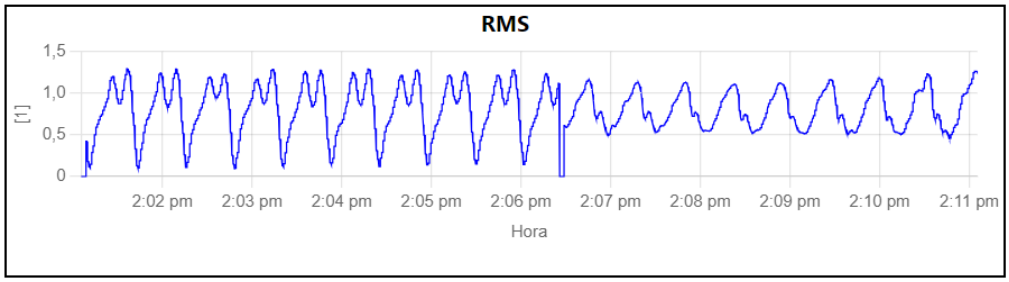

Figura 4.15: Gráfico de RMS de las aceleraciones en los tres ejes

Luego se incluyen tres gráficos de linea como el de la figura 4.16. Estos gráficos muestran la transformada de Fourier para las aceleraciones en los tres ejes respectivamente, sobre la última ventana de datos analizada. Cada vez que se recibe una notificación estos gráficos actualizan completamente sus datos. Se incluyó esta visualización ya que en el contexto de mantenimiento de motores, un experto puede obtener bastante información del estado del motor, con solo mirar la transformada de Fourier, fijándose en las frecuencias presentes y más predominantes.

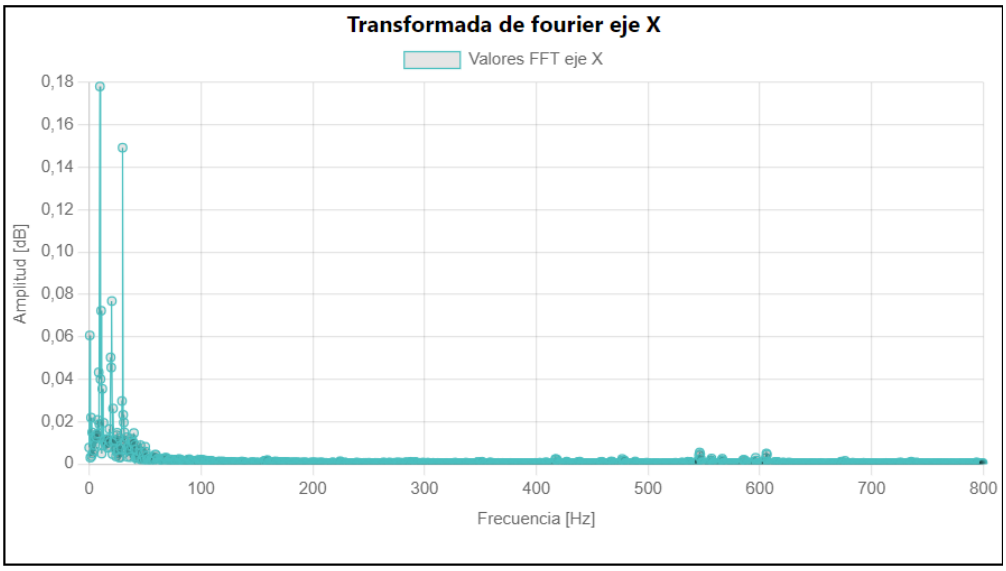

Figura 4.16: Gráfico de la transformada de Fourier de las aceleraciones

Capítulo 5

Evaluación y Resultados

A modo general, los modelos entrenados para este trabajo no pueden ser aplicados a otro tipo de motor. Ya que el funcionamiento correcto o en falla de un motor esta fuertemente ligado a sus frecuencias propias en distintas condiciones. Estas se obtienen generando una señal de impulso. Que consiste en alterar el motor estando inmóvil y medir como se comporta en términos de vibración.

Fuera de eso, el resto de elementos como la recolección, transmisión, procesamiento y visualización de los datos si pueden ser adaptados a otro tipo de motor.

5.1. Objetivos específicos

A continuación se presentan los resultados para cada uno de los objetivos propuestos, ordenado según el listado presentado en dicha sección de *Objetivos específicos*.

5.1.1. Montar un sistema con el motor y los sensores para generar datos históricos sobre su funcionamiento con y sin fallas.

Los datos históricos del funcionamiento del motor fueron proveídos por el profesor guía, que ya contaba con un conjunto de datos etiquetados para distintas condiciones de funcionamiento del motor.

Generar este tipo de datos implica una planificación de tiempo y logística considerable. Por lo que no se pudo completar este objetivo usando el módulo de recolección del sistema propuesto.

5.1.2. Asegurar integridad de los datos, tanto en el proceso de recolección como en el de transmisión.

Para procesar los datos entregados por el sensor, se siguió la fórmula del documento de su ficha técnica. Para asegurar la integridad en la transmisión entre la ESP32 y el

programa con acceso a la base de datos, se utilizó el protocolo TCP, ya que este asegura que no habrán perdida ni corrupción de paquetes.

5.1.3. Elegir algún algoritmo de ML capaz de generar diagnósticos confiables, tomando los datos del problema estudiado. Se definen diagnósticos confiables como resultados que alcancen un 98 % de precisión, valor obtenido como referencia del trabajo estudiado [13]

Resumiendo los resultados obtenidos en la sección 4.2.1. Para la generación de diagnósticos en tiempo real, el clasificador Support Vector Classifier (también conocido como Support Vector Machine o SVM/SVC) entregó resultados con un nivel de precisión de a lo más 95 %.

Con respecto a la generación de pronósticos, usando redes neuronales, se planteó predecir las aceleraciones del motor a partir de los datos históricos, para luego generar diagnósticos sobre esos datos predichos usando el modelo mencionado en el párrafo anterior.

La idea era la siguiente, generar una red neuronal de largo-corto plazo (Long Short-Term Memory o LSTM) que reciba como entrada señales de aceleración para una ventana de tiempo t. Para que luego entregue como salida una nueva señal de aceleración estimada para un tiempo $t+1$. La idea era replicar esto para las aceleraciones en los tres ejes.

Se utilizó una red LSTM con la función de activación *Rectified Linear Activation Function* (RelU) por la dependencia no lineal de las aceleraciones. Como se puede ver en la figura 5.1, las aceleraciones estimadas se alejan de el valor original. Por lo mismo no se pudo continuar con la parte de generar diagnósticos sobre los datos estimados.

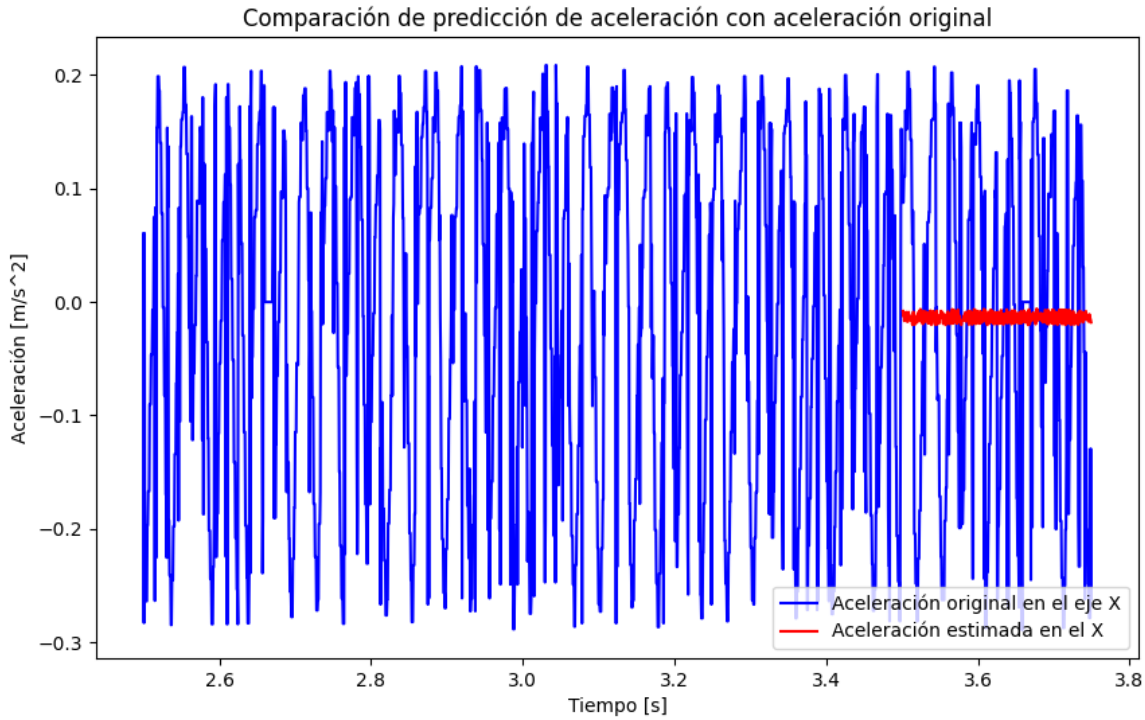

Figura 5.1: Resultados de estimación de aceleraciones usando red LSTM

5.1.4. Mostrar el estado general del motor y los pronósticos hechos sobre el mismo de una forma intuitiva para el usuario en el servidor.

Para la interfaz se utilizó una encuesta llamada Escala de Usabilidad del Sistema (*System Usability Scale* o SUS, en inglés) para poder cuantificar la percepción de los usuarios sobre varios aspectos de la interfaz. Esta encuesta consta de 10 afirmaciones, para cada afirmación los usuarios contestaron que tan de acuerdo están con la afirmación hecha, en un rango desde 1 que representa “Estoy muy en desacuerdo”, hasta un 5 que representa “Estoy muy de acuerdo”. Las afirmaciones son las siguientes:

1. Creo que me gustaría usar el sistema con frecuencia.
2. Encontré el sistema innecesariamente complejo.
3. Encontré el sistema de la página fácil de usar.
4. Creo que necesitaría el apoyo de una persona técnica para poder usar el sistema
5. Encontré que las diversas funciones en el sistema estaban bien integradas.
6. Pensé que había demasiada incoherencia en el sistema usado.

7. Me imagino que la mayoría de la gente aprendería a usar este sistema muy rápidamente.
8. Encontré el sistema muy engorroso de usar.
9. Me sentí muy seguro usando el sistema.
10. Tuve que aprender muchas cosas antes de poder comenzar con el uso de este sistema.

Además se dejo la pregunta abierta “¿Qué otros datos/formas de procesamiento le serían útiles al momento de analizar el estado del sistema?” y un espacio para comentarios generales.

Se encuestó a diez personas con conocimientos de ingeniería civil mecánica, en particular a estudiantes y egresados de la carrera de la Universidad de Chile.

Para la pregunta abierta, 7 personas encontraron que sería útil incorporar mediciones sobre la temperatura del motor, su alimentación (voltaje y/o corriente) y velocidad en revoluciones por minuto (rpm).

A continuación se muestran los resultados obtenidos para la encuesta SUS, se separaron en dos gráficos distintos ya que para las preguntas 1,3,5,7 y 9 es positivo responder con un 5, mientras que para las preguntas 2,4,6,8 y 10 es negativo.

Primero, en la figura 5.2, se muestra el gráfico donde un 5 representa una percepción positiva.

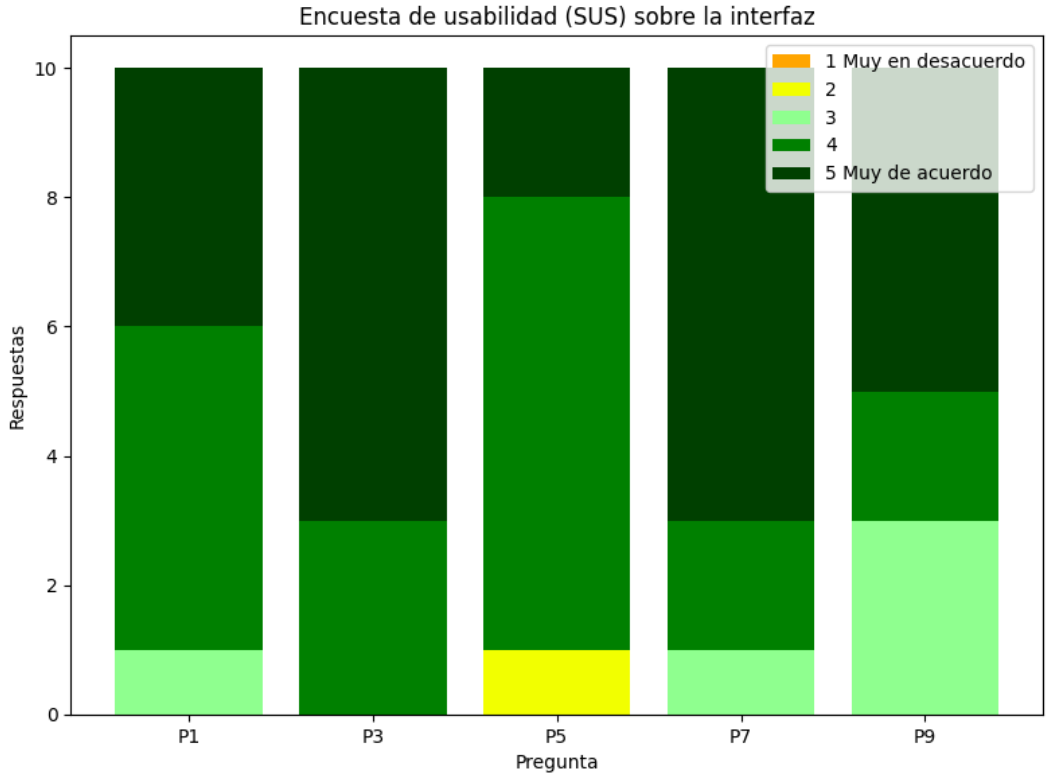

Figura 5.2: Resultados de la encuesta para las preguntas 1,3,5,7 y 9

Para las afirmaciones 1,3 y 7 se puede observar un rango de respuestas mayormente ubicado entre 4 o 5, es decir, están satisfechos con el sistema con respecto a esas afirmaciones.

Con respecto a la afirmación 5 “Encontré que las diversas funciones en el sistema estaban bien integradas.” un usuario comentó que le serviría “Garantizar si es a tiempo real o tiempo de muestreo”. En el caso del problema los datos se actualizan en tiempo real se podría mejorar agregando algún elemento o indicador visual para darle una retroalimentación al usuario cada vez que se haga una actualización

Las respuestas del resto de las preguntas se pueden ver en la figura 5.3.

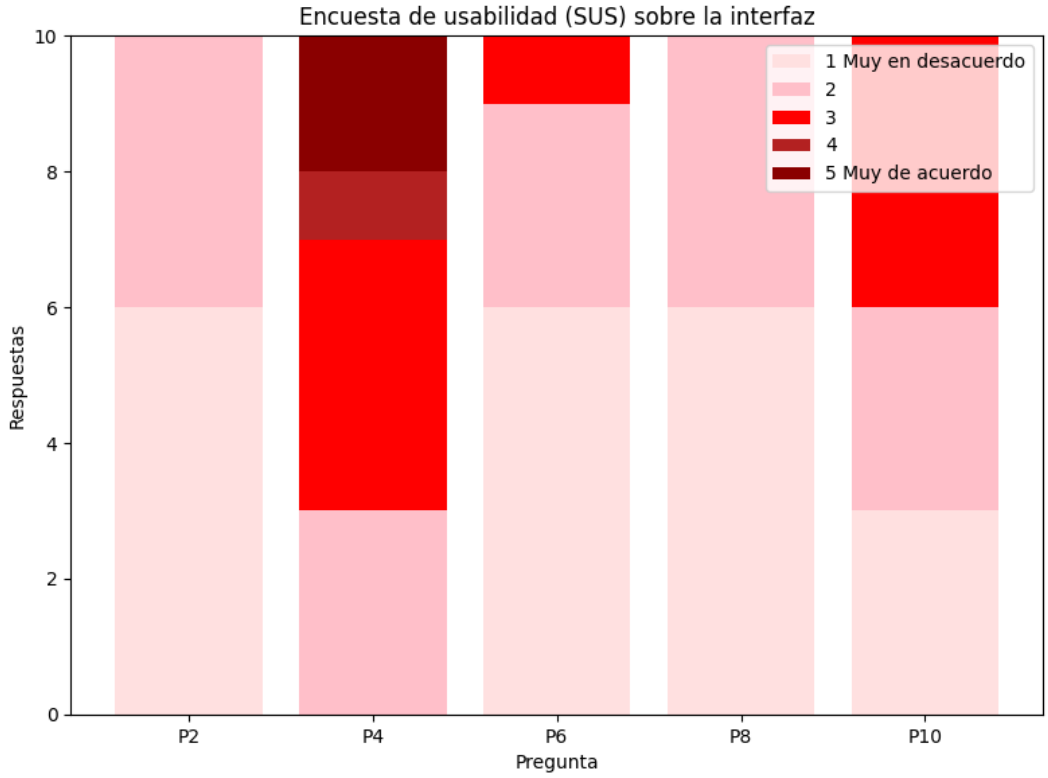

Figura 5.3: Resultados de la encuesta para las preguntas 2,4,6,8 y 10

Para las afirmaciones 2,6, 7 y 10 se puede observar respuestas mayormente en el rango de 1 a 3, que para este caso significa satisfacción con el sistema.

5.2. Objetivo general

Cada uno de los módulos del sistema propuesto cumple con su objetivo planteado, y la comunicación entre los distintos módulos también funcionó como se esperaba (justificar más).

Sin embargo no se pudo ejecutar una prueba del sistema funcionando completamente en el laboratorio del motor.

Capítulo 6

Conclusiones

Este trabajo muestra el diseño y desarrollo de un sistema para monitoreo de condiciones junto a generación de diagnósticos, sobre la condición actual de funcionamiento de un motor eléctrico.

Con respecto al trabajo logrado, se logró desarrollar un sistema capaz de recolectar y transmitir datos desde un motor eléctrico, para luego generar diagnósticos en tiempo real con una precisión de 94 %, junto a una interfaz que en su mayoría le pareció intuitiva a los usuarios encuestados. En teoría el sistema cumple con mantener la integridad de los datos, pero no se llegó a comprobarlo en un experimento con el sistema completo funcionando.

Con respecto a la parte de ML, la razón por la que SVM entregó mejores resultados en general en comparación a otros clasificadores, es que al aplicar la transformada de Fourier sobre las aceleraciones, se elimina la no linealidad de las señales de aceleraciones, donde para este tipo de problemas SVM entrega buenos resultados. SVM además funciona mejor para problemas de clasificación binaria, como es el caso de diagnosticar si el motor está balanceado o des balanceado.

Con el trabajo descrito en este informe, se entrega un sistema simplificado para la replicación con otro motor, teniendo sus datos históricos. Ya que se resolvieron bastantes desafíos a tener en cuenta para un sistema de este tipo, como por ejemplo, el pre-procesamiento para llevarlo a distintos algoritmos de clasificación y los protocolos y formas de comunicación escogidos para interactuar entre cada módulo.

Desarrollar un sistema de este tipo, donde se involucran varias partes independientes que se tienen que comunicar entre sí, implica bastante trabajo adicional, ya que para cada parte se tiene que hacer una etapa individual de investigación, diseño y pruebas. Además de que la comunicación entre las partes debe funcionar de una manera muy fluida, ya que cualquier problema o retraso en la comunicación de los datos, genera una

mala experiencia en todo el sistema.

6.1. Trabajos a futuro

Para posibles desarrollos futuro, se puede considerar abarcar un mantenimiento predictivo, después de los trabajos realizados, se plantean dos formas en las cuales esto se podría lograr:

1. Realizar una predicción de las señales de aceleración para una ventana próxima de tiempo, en base a las condiciones previas y/o actuales de funcionamiento del motor, para luego generar diagnósticos sobre esos datos futuros. Se sugiere usar una red LSTM para hacer este pronósticos de datos, pero cualquier modelo regresor podría cumplir este trabajo.

2. Conseguir datos históricos que indiquen en más detalle en que condiciones está funcionando el motor, por ejemplo que junto a los datos se incluya un indicador de salud en forma de porcentaje, donde si este baja de un 20 %, se pueda considerar que el motor entra en fallo. De esta forma se puede tener un monitoreo más detallado sobre la condición del motor y se podría estimar, dadas las condiciones actuales de funcionamiento, en cuanto tiempo se llegará a determinado porcentaje de salud.

Para mejorar las precisiones alcanzadas por los distintos modelos, se podría hacer un proceso de ajustes de hiperparámetros para cada uno.

En base a los resultados de las encuestas, se podría mejorar la interfaz de visualización al incluir otro tipo de variables al sistema, como la temperatura, la fuente de energía del motor o su velocidad, lo cual complementaría los datos totales a visualizar, permitiendo diagnósticos adicionales de otras variables de operación.

Por último, ya que las condiciones funcionamiento de un motor dependen fuertemente de sus frecuencias naturales, se podría desarrollar una forma de incluir esta (u otras) características específicas de un motor al modelo encargado de generar diagnósticos, de esta forma se podría aplicar este sistema a cualquier tipo de motor, con tal de conocer sus características físicas específicas.

Anexo

Anexo A

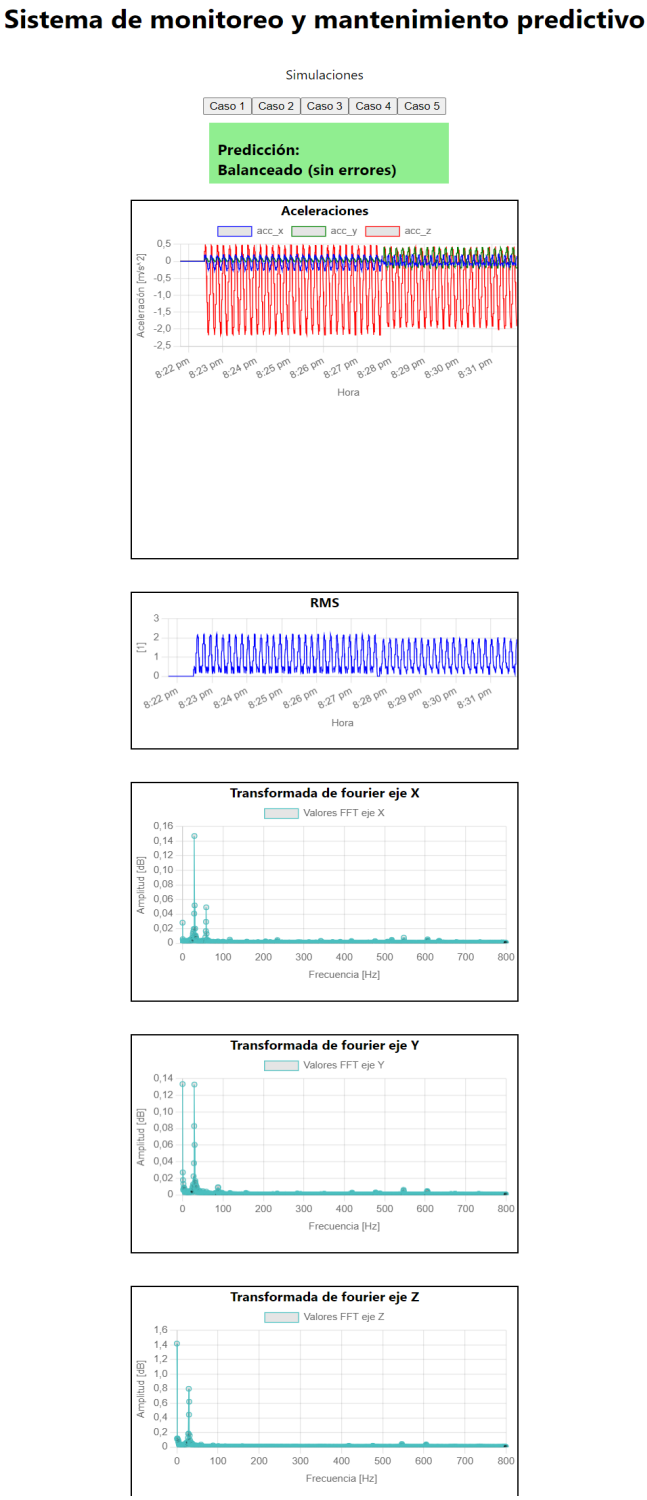

Figura 6.1: Interfaz completa de visualización

Bibliografía

[1] E. Brown , “Using the IoT for Predictive Maintenance” , Tech Briefs, vol. 44, no. 9, Sep. 2020. Accessed: Jul. 15, 2023. [Online]. Available: https://www.techbriefs.com/component/content/article/tb/supplements/st/features/applications/37661

[2] S. Guo,“Data-Driven Predictive Maintenance In a Nutshell” towardsdatascience.com https://towardsdatascience.com/data-driven-predictive-maintenance-in-a-nutshell-ccc65a13b998 (accessed Jul. 15, 2023).

[3] N. Sakib and T. Wuest, “Challenges and Opportunities of Condition-based Predictive Maintenance: A Review”, 6th CIRP Global Web Conference – Envisaging the future manufacturing, design, technologies and systems in innovation era (CIRPe 2018), vol. 78, pp 267-272, 2018.

[4] “Estrategia Nacional de Electromovilidad”, Ministerio de Energía, Ministerio de Transportes y Telecomunicaciones, Ministerio del Medio Ambiente, Dic. 13 de 2017.

[5] M. Compare, P. Baraldi and E. Zio, “Challenges to IoT-Enabled Predictive Maintenance for Industry 4.0”, IEEE INTERNET OF THINGS JOURNAL, vol. 7, no. 5, pp 4595-4597, May 2020.

[6] A. J. Bazurto, E. C. Quispe and R. C. Mendoza, “Causes and failures classification of industrial electric motor”, 2016 IEEE ANDESCON, Arequipa, Peru, 2016, pp. 1-4, doi: 10.1109/ANDESCON.2016.7836190.

[7] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni and J. Loncarski, “Machine Learning approach for Predictive Maintenance in Industry 4.0”, 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Oulu, Finland, 2018, pp. 1-6, doi: 10.1109/MESA.2018.8449150.

[8] A. A. Manjare y B. G. Patil, “A Review: Condition Based Techniques and Predictive Maintenance for Motor”, en 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), Coimbatore, India: IEEE, mar. 2021, pp. 807–813. doi: 10.1109/ICAIS50930.2021.9395903.

[9] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. D. P. Francisco, J. P. Basto, y S. G. S. Alcalá, “A systematic literature review of machine learning methods applied to predictive maintenance”, Computers Industrial Engineering, vol. 137, p. 106024, nov. 2019, doi: 10.1016/j.cie.2019.106024.

[10] N. A. Mohammed, O. F. Abdulateef, y A. H. Hamad, “An IoT and Machine Learning-Based Predictive Maintenance System for Electrical Motors”, JESA, vol. 56, nº 4, pp. 651–656, ago. 2023, doi: 10.18280/jesa.560414.

[11] M. Compare, P. Baraldi, y E. Zio, “Challenges to IoT-Enabled Predictive Maintenance for Industry 4.0”, IEEE Internet Things J., vol. 7, nº 5, pp. 4585–4597, may 2020, doi: 10.1109/JIOT.2019.2957029.

[12] J. Dalzochio et al., “Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges”, Computers in Industry, vol. 123, p. 103298, dic. 2020, doi: 10.1016/j.compind.2020.103298.

[13] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni, y J. Loncarski, “Machine Learning approach for Predictive Maintenance in Industry 4.0”, en 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Oulu: IEEE, jul. 2018, pp. 1–6. doi: 10.1109/MESA.2018.8449150.