Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

Licenciatura en Ingeniería en Sistemas Automotrices

"Diseño y construcción de una estructura con base tipo trípode de un aerogenerador offshore para la obtención de una base de datos en estado saludable"

T E S I S

Presentada para obtener el grado de:
Licenciado en Ingeniería en Sistemas Automotrices

Presenta:
Miguel Angel Cordero Romero

Directores de Tesis:

Dr. José Eligio Moisés Gutiérrez Arias (FCE-BUAP, México)

Dra. Gabriela Pérez Osorio (FIQ-BUAP)

Dr. Donaldo García Juanillo (CCT-UACM)

Puebla, Puebla. Octubre 2025

Agradecimientos

A Dios, por darme salud y acompañarme en cada paso de este camino.

A mi padre Juan, por enseñarme a no rendirme y recordarme siempre que todo problema tiene solución si se enfrenta con determinación.

A mi madre Rufi, por su apoyo incondicional y por enseñarme el valor de la perseverancia.

A mi hermana Esmeralda, por sacarme una sonrisa en los momentos de tensión y ayudarme a continuar con mi tesis.

A mi novia Marlene, por su comprensión, compañía y apoyo incondicional a lo largo de este proceso.

A mi prima Rosa, por sus comentarios acertados y su disposición para ayudarme siempre.

Al Dr. Moisés Gutiérrez, por su apoyo, dedicación, paciencia y compartir generosamente su conocimiento; la forma en la que explica es admirable.

Agradezco al Dr. Marco Antonio Vásquez, el Dr. Erick Javier Jiménez y la Mtra. Gabriela Aquino por sus observaciones y comentarios expuestos que enriquecieron el trabajo realizado.

Y a todos los compañeros y amigos que me brindaron su ayuda y respaldo durante la elaboración de esta tesis.

A todos, mi más sincero agradecimiento.

Resumen

En esta tesis se llevó a cabo la construcción de una estructura de aerogenerador tipo *"offshore"* con base tipo trípode, diseñado como banco de pruebas en estado saludable para la obtención de una base de datos en crudo. Una de sus principales características es que es completamente desarmable, lo cual facilita su transporte, instalación y la posibilidad de representar fallas estructurales en futuras pruebas de investigación.

Para el monitoreo del sistema, se caracterizaron sensores tipo acelerómetro (modelo MPU9250) y se instalaron en puntos estratégicos identificados en la literatura como zonas de mayor desgaste. En específico, se colocaron en la columna central, en cada uno de los tres refuerzos laterales de la base, en la torre y en el cojinete donde va el eje que acopla con el disco de las aspas, sumando un total de seis sensores.

Además, se desarrolló una interfaz gráfica en un archivo ejecutable (.exe), compatible con sistemas operativos Windows, que permite la visualización en tiempo real de los datos adquiridos de forma inalámbrica. Esta herramienta también permite cronometrar el tiempo de adquisición y guardar automáticamente las lecturas en un archivo Excel.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El aerogenerador es un dispositivo que genera electricidad a partir del movimiento de las aspas accionadas por el viento.

A lo largo de los años, los aerogeneradores han experimentado una evolución significativa, pasando de los "*onshore*" término que se refiere a las instalaciones terrestres a *"offshore"* para instalaciones marinas, los cuales se han convertido en el enfoque principal de la industria debido a la creciente demanda global de energía y la necesidad de aprovechar los recursos eólicos en alta mar. Este cambio responde a la mayor disponibilidad de viento en áreas marinas, lo que permite una producción energética más eficiente y sostenible, mientras se reduce el impacto visual y acústico en zonas terrestres.

Por lo que el problema principal es que los costos de mantenimiento son elevados ya que representan valores que van desde el 25 % al 35 % del total del proyecto en el tipo *offshore* debido a la relación de tamaño, donde, a mayor tamaño de la turbina, mayor es el daño; además de tener en cuenta que cierta cantidad de la infraestructura que existe ha llegado a su vida útil de 20 años Artigao, Martín-Martínez, Honrubia-Escribano, y Gómez-Lázaro (2018).

Entonces, profundizando en algunos componentes donde se presentan fallas, encontramos el cojinete de la turbina, roturas estructurales, caja de cambios que presenta cierto tipo de vibraciones, las cuales son importantes para su análisis B. Chen, Xie, Li, y Gao (2020) , Dao (2022).

Las técnicas de monitoreo convencionales, como el análisis de vibraciones, han sido comunes en la industria; no obstante, estas metodologías requieren inversiones en sistemas costosos por los sistemas de adquisición especializados, lo que puede limitar su uso en instalaciones ubicadas en zonas aisladas. El análisis de vibraciones se utiliza para identificar problemas mecánicos; sin embargo, carece de la inmediatez necesaria para una detección eficaz de fallos como lo señala Pozo, Vidal, y Salgado (2018).

Además, con el crecimiento en tamaño y complejidad de los parques eólicos *"offshore"* en Europa, Estados Unidos de América y China, de acuerdo con Artigao y cols. (2018), el monitoreo del estado de salud de sus componentes se ha vuelto crítico para garantizar la integridad estructural y el funcionamiento eficiente de los aerogeneradores. En este sentido, se han desarrollado diversos estudios e investigaciones centrados en mejorar las técnicas de monitoreo y detección de fallas. Un ejemplo de ello es el trabajo de Aquino González (2020), quien desarrolló y validó, tanto numérica como experimental, un nuevo método

para la detección de daños en aerogeneradores *"offshore"* en un banco de pruebas aplicando la técnica PCA (Análisis de Componentes Principales). Dicho estudio propone un enfoque novedoso para identificar daños estructurales, utilizando modelos de análisis de vibraciones y sistemas de monitoreo avanzados para prevenir fallos críticos y aumentar la confiabilidad de los aerogeneradores marinos.

Para complementar estos avances, se plantea la necesidad de desarrollar un modelo a escala de un aerogenerador, incorporando sensores en puntos estratégicos, como lo realizó Aquino González (2020) en su investigación, para obtener mediciones más precisas de parámetros sobre la vibración, por el esfuerzo mecánico o fatiga de los materiales.

Con la construcción del aerogenerador se podrá obtener una base de datos en crudo, que con el apoyo de inteligencia artificial, permitirá obtener diagramas en estado saludable, tanto de la estructura como de los componentes en la góndola del aerogenerador. Este enfoque no solo optimizará los procedimientos de mantenimiento preventivo, sino que también prolongará la vida útil de las instalaciones *"offshore"*, mejorando la eficiencia y reduciendo los costos operativos a largo plazo.

1.1. Estado del arte

Se han realizado análisis de frecuencia natural en la cimentación de la estructura trípode como lo plantea Zhang, Wang, Zhang, y Gao (2019), enfocándose en las propiedades del material de la base trípode mediante el uso de elemento finito. En este estudio, las variables de diseño y los materiales se consideran aleatorios, lo que permite evaluar la influencia de estos factores. La frecuencia natural se utiliza como criterio clave para optimizar el diseño de manera eficaz.

Otro estudio Plodpradit, Dinh, y Kim (2019) se centran en determinar la fatiga de la estructura, donde se comparan los resultados obtenidos por análisis de elementos finitos, tanto de la base como de la torre de forma individual. Posteriormente, se utiliza un super elemento para representar la interacción entre los pilotes de cimentación y el suelo marino. Esta técnica permite reducir significativamente el tiempo de cálculo sin comprometer la precisión de los resultados.

El Análisis de Componentes Principales Multidireccional (MPCA) con Pruebas de Hipótesis Estadísticas Multivariadas (MSHT), utiliza datos del sistema Supervisión, Control y Adquisición de Datos (SCADA) de las turbinas, lo que facilita la detección de fallos en tiempo real y aprovecha la información existente. La metodología ha mostrado una precisión del 100 % en diversas simulaciones de fallos, lo que resalta su efectividad.

Las ventajas del sistema propuesto incluyen una detección más ágil de fallos, una notable reducción en los costos operativos y la capacidad de funcionar eficazmente en condiciones de viento variable. Esto no solo incrementa la fiabilidad operativa de las turbinas, sino que también promueve una mayor inversión en energía eólica, posicionando este enfoque como una alternativa viable para el monitoreo en el sector Pozo y cols. (2018)

El mantenimiento de las turbinas eólicas, particularmente de componentes críticos como la caja de cambios, es esencial para evitar fallos costosos y mantener su funcionamiento continuo. Tradicionalmente, estas fallas se detectaban a través de inspecciones físicas, pero

estas requieren mucho tiempo y recursos. Con el avance de las tecnologías de monitoreo, se han desarrollado metodologías más eficientes basadas en el análisis de vibraciones y el procesamiento avanzado de señales, lo que permite identificar fallos de manera más rápida y precisa.

El análisis de vibraciones se ha convertido en una herramienta clave para monitorear el estado de la caja de cambios de las turbinas eólicas. Detectar cambios sutiles en la frecuencia y amplitud de las vibraciones ayuda a identificar fallos incipientes antes de que se conviertan en problemas mayores. Esto proporciona una ventaja importante, ya que permite realizar mantenimientos preventivos y evitar fallas que podrían interrumpir el funcionamiento de las turbinas.

Una mejora significativa en este campo es el uso de los coeficientes cepstrales de frecuencia Mel (MFCC), una técnica que permite extraer características clave de las señales de vibración. Estos coeficientes son especialmente útiles para identificar patrones anómalos en señales no estacionarias, como las que generan las turbinas eólicas en operación. Esto mejora notablemente la detección de fallas en comparación con los métodos tradicionales.

Además, el uso de aprendizaje profundo, en particular de redes neuronales convolucionales (CNN) Rangel (2022), ha permitido un análisis más eficiente y preciso de los datos de vibración. Estos algoritmos son capaces de identificar patrones complejos que podrían pasar desapercibidos con enfoques más convencionales, facilitando la detección automática de fallas sin intervención humana constante.

Dicho enfoque no solo mejora la fiabilidad de las turbinas; también reduce el tiempo de inactividad no planificado y los costos de mantenimiento. Al poder recolectar y analizar los datos de manera remota, se elimina la necesidad de inspecciones físicas frecuentes, minimizando aún más los costos y riesgos asociados al mantenimiento, como lo señala Velandia-Cardenas, Vidal, y Pozo (2024).

También, la detección de fallas en rodamientos principales de turbinas eólicas utiliza datos de SCADA y el análisis de componentes principales (PCA), lo que permite identificar errores de manera anticipada, incluso meses antes de que ocurran daños graves. La metodología está enfocada en detectar fugas de calor en los rodamientos, las cuales son señales tempranas de deterioro que podrían afectar otros componentes clave, como la caja de cambios y las aspas.

El análisis de componentes principales (PCA) procesa los datos de SCADA, comparando el comportamiento actual de la turbina con el histórico para detectar cualquier anomalía. Cuando se observa una desviación significativa en los datos, el sistema genera alertas que permiten al operador actuar antes de que el fallo se agrave. Este enfoque facilita el mantenimiento predictivo y reduce la probabilidad de fallas críticas.

Una de las mayores ventajas de este sistema es que no requiere la instalación de nuevos sensores ni equipos adicionales, lo que lo hace rentable y aplicable incluso en turbinas eólicas más antiguas. Además, no es necesario contar con un historial de fallas previas, ya que el sistema puede entrenarse únicamente con datos de operación normales, simplificando su implementación.

La efectividad de este enfoque se probó en un parque eólico en Polonia, donde se observó que el sistema era capaz de generar alertas con hasta seis meses de antelación respecto a la falla de los rodamientos principales en una turbina. Esto permitió a los operadores planificar el mantenimiento con tiempo suficiente, reduciendo los costos de reparación y

minimizando los tiempos de inactividad Campoverde-Vilela, Feijóo, Vidal, Sampietro, y Tutivén (2023).

1.2. Justificación

Con la creciente tendencia hacia el uso de la energía eólica como una fuente de energía renovable, los aerogeneradores se están volviendo cada vez más eficientes y, en consecuencia, más caros. Esto hace que la prevención de situaciones críticas sea de suma importancia para maximizar la vida útil y la rentabilidad de estos equipos. Al obtener una base de datos detallada y precisa, es posible diagnosticar fallas de manera proactiva, lo que a su vez reduce significativamente los costos de reparación y mantenimiento en aerogeneradores *offshore* tipo Trípode.

La implementación de planes de monitorización y estrategias de mantenimiento eficientes ya no es opcional, sino una necesidad. La monitorización continua permite la detección temprana de problemas, lo que reduce el tiempo de inactividad y evita reparaciones costosas. Además, mejora la seguridad operativa y alarga la vida útil de los componentes.

En esta tesis se busca obtener una base de datos en crudo del aerogenerador de tipo trípode en tres puntos clave: la base, la torre y la góndola. Con esta información, en investigaciones posteriores se pretende desarrollar un sistema de diagnóstico que permita la detección de fallas y el mantenimiento predictivo, lo que contribuirá a una operación más eficiente y económica de los parques eólicos *offshore*.

Además, el uso de tecnologías avanzadas de monitorización y análisis de datos en aerogeneradores no solo optimiza los costos, sino que también contribuye a la sostenibilidad ambiental al asegurar un funcionamiento más eficiente y reducir la huella de carbono asociada con las reparaciones y reemplazos de componentes. En un contexto global donde la demanda de energía limpia sigue creciendo, proyectos como este son esenciales para asegurar la viabilidad y competitividad de la energía eólica en el futuro.

En el mapa de la **Figura 1.1** se observan las instalaciones de parques eólicos en el mundo, tanto *offshore* como *onshore*. Asimismo, se enumeran los países con mayor producción de energía eólica en gigavatios (GW), destacando a China como uno de los principales líderes, seguido por Estados Unidos y Alemania.

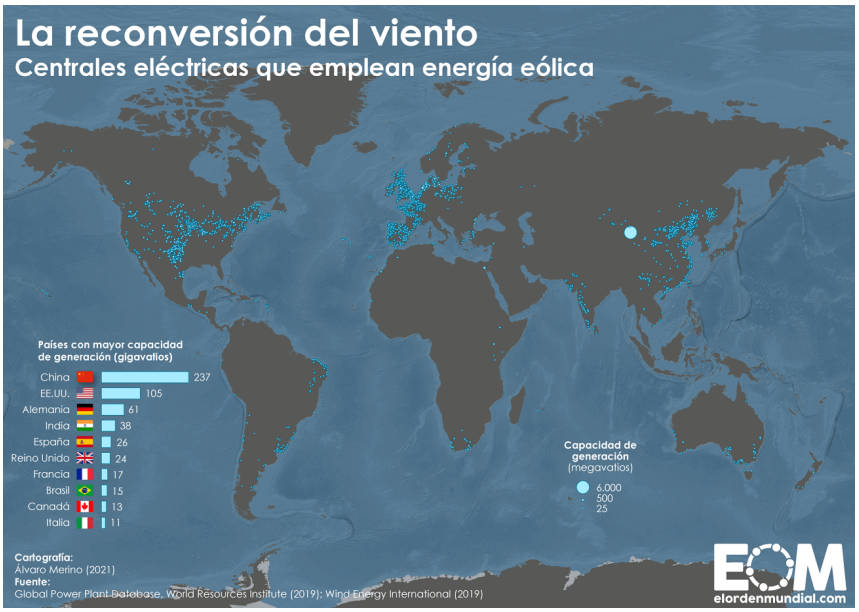


Figura 1.1: Mapa de parques eólicos en el mundo. Recuperado de Merino (2021).

Cabe destacar que China ha desarrollado el aerogenerador GWH252-16MW, lo que demuestra su capacidad para generar más energía con un menor número de aerogeneradores. Esto representa un avance significativo en la eficiencia y evolución tecnológica del sector eólico, así como lo describe Elton y Euronews (2024).

Europa se encuentra como líder en instalaciones *offshore* como se ve en la **Figura 1.2**.

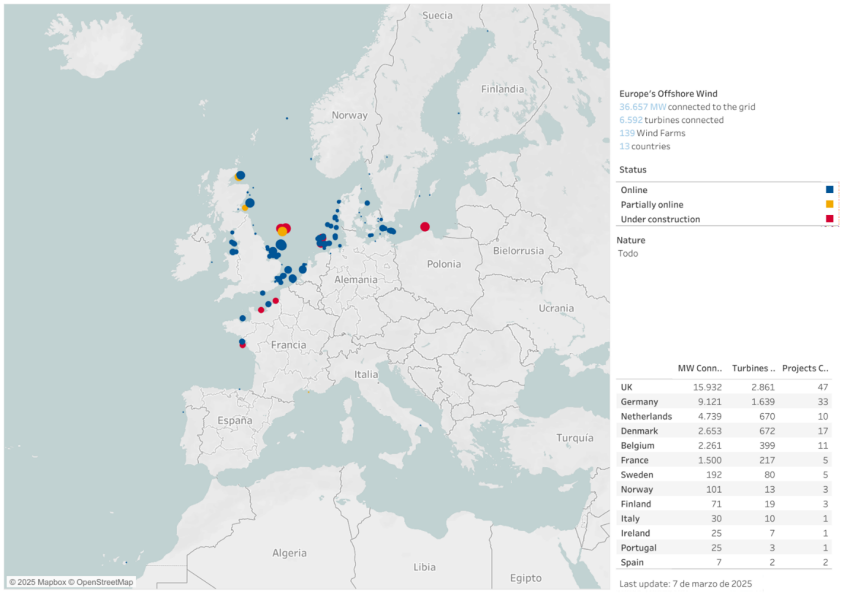


Figura 1.2: Mapa de parques eólicos *offshore* en Europa. Recuperado de WindEurope (2025).

Para el 2030 invertirá 6.5 mil millones de euros en infraestructura, esto de acuerdo con WindEurope (2022).

Este crecimiento se ve reflejado en un mayor porcentaje en Europa, ya que las estadísticas de Wind Europe muestran que hubo un crecimiento de 3.8 GW en el año 2023, como se muestra en la **Figura 1.3**. Comparando esto con los años anteriores, este tipo de estructuras *offshore* está en continuo crecimiento.

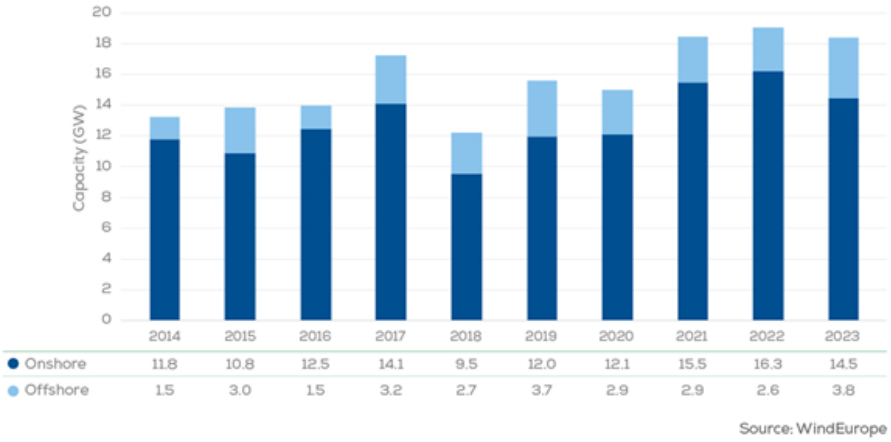


Figura 1.3: Capacidad anual de energía *onshore* y *offshore* instalada en Europa. Recuperado de WindEurope (2024).

En Europa, los países principales con mayor crecimiento en estructuras *offshore* son Alemania, generando en el año 2023 329 MW , Holanda (NL) con 1906 MW y Reino Unido con 833 MW. Esto representa un crecimiento importante en Holanda, como se muestra en la **Figura 1.4**, tal como lo señala WindEurope (2024).

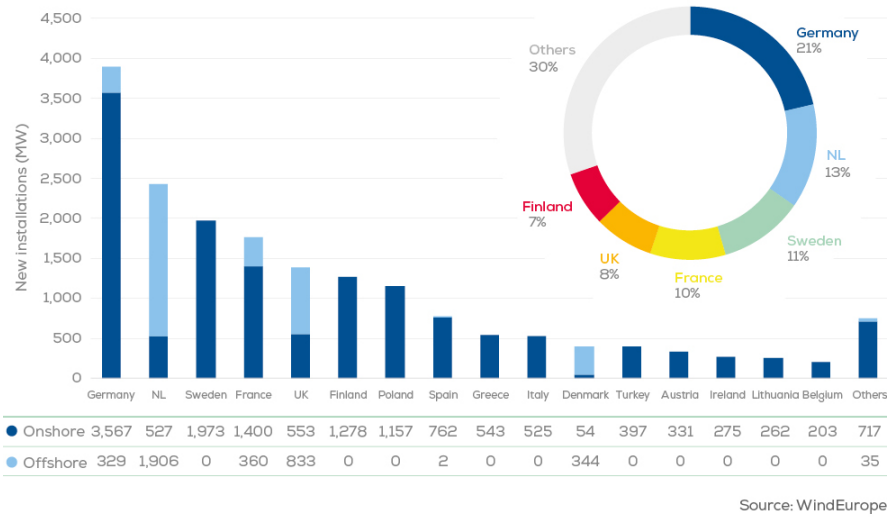


Figura 1.4: Nuevas instalaciones eólicas *onshore* y *offshore* en Europa año 2023. Recuperado de WindEurope (2024).

Por lo tanto, es un tema que permite la investigación constante para el desarrollo de aerogeneradores eficientes y duraderos.

1.3. Objetivos

Objetivo general

Diseñar y construir la estructura de un prototipo de aerogenerador *offshore* con base tipo trípode para la obtención de una base de datos en condiciones saludables de operación.

Objetivos específicos

1. Analizar la metodología que existe en la bibliografía para la detección de vibraciones en los aerogeneradores.
2. Diseñar prototipo en CAD para adecuarlo como banco de pruebas.
3. Construir el prototipo con base tipo trípode, torre y góndola.
4. Seleccionar los sensores y módulos de comunicación adecuados para la captura de datos.
5. Caracterizar los sensores y módulos de comunicación para transmisión de datos de forma inalámbrica.
6. Realizar una interfaz gráfica de usuario que permita monitorizar los sensores en tiempo real.
7. Obtener la base de datos en bruto de los sensores ubicados en el aerogenerador, en estado saludable de forma inalámbrica.

1.4. Organización de la tesis

Este trabajo se estructura en siete capítulos. En el Capítulo 1 se presentan los antecedentes de esta investigación, incluyendo una revisión de las instalaciones *offshore* en el mundo, los elementos que experimentan mayor desgaste en estas estructuras y la metodología basada en el análisis de vibraciones, así como la justificación del desarrollo de este trabajo.

El Capítulo 2 corresponde al marco teórico, donde se abordan los conceptos fundamentales para el análisis de aerogeneradores de base tipo trípode, además de los principios que sirven como fundamento para la metodología planteada.

En los Capítulos 3 a 5 se describe de manera detallada la metodología implementada para la obtención de la base de datos en bruto, correspondiente al estado saludable del prototipo de aerogenerador construido. El Capítulo 6 expone los resultados obtenidos a partir de dicha metodología, mientras que el Capítulo 7 presenta las conclusiones derivadas del estudio, así como las posibles líneas de trabajo futuro que pueden desarrollarse a partir de esta investigación.

Capítulo 2

Marco teórico

2.1. Definiciones

A continuación se presentan algunas definiciones necesarias para el desarrollo de este trabajo:

2.1.1. Aerogenerador

Es un sistema de conversión de energía, ya que genera energía eléctrica a partir del movimiento mecánico de las aspas accionadas por el viento hasta llegar a un generador, Romero y Juárez (2024).

Características de un aerogenerador

Para producir energía eólica, la velocidad promedio del viento debe ser mayor a 10 km/h (2.7 m/s); cuando hay vientos que superan los 90 km/h (25 m/s), el aerogenerador se frena por motivos de seguridad. Cada aerogenerador puede producir energía durante más de 25 años. Las aspas se mueven por acción del viento y transmiten el movimiento al generador a través de un sistema de engranajes que se denomina caja multiplicadora. La electricidad producida en el generador como corriente continua es conducida por el interior de la torre hasta su base; ahí un convertidor la transforma en corriente alterna. Luego, un transformador eleva la tensión para poder transportarla por el interior del parque; dedicada al aerogenerador, la corriente alterna es conducida por cables subterráneos hasta la subestación. En ella se eleva nuevamente la tensión para poder incorporarla a la red eléctrica nacional y transportarla a los puntos de consumo, Angarita Jaimes y cols. (2018).

Las aspas de los aerogeneradores suelen girar entre 12 y 20 rpm, esto solo es un promedio ya que depende de la tecnología, el diseño y tamaño del aerogenerador ya que entre mayor es la altura, menor son las rpm, además de que cuando la velocidad del viento es de 3.5 m/s es cuando comienzan a moverse las aspas y cuando llega a los 11 m/s alcanza su máxima potencia según Acciona (s.f.).

Componentes

Los componentes de un aerogenerador, en general, son los siguientes:

- Góndola: Es el lugar donde se concentran los siguientes elementos, los cuales son importantes para la obtención de energía eléctrica.
  - Aspas del rotor: son accionadas por el viento haciendo rotar el eje.
  - Eje: transfiere energía cinética al generador.
  - Caja de cambios (multiplicador): eleva el número de revoluciones de eje.
  - Generador: produce energía eléctrica a partir del movimiento del eje a altas revoluciones.
  - Freno: disminuye el giro del eje y de la caja de cambios.
- Torre: sostiene la góndola y eleva toda la instalación a una mayor altura para aprovechar el viento para mover las aspas.
- Base: parte inferior del aerogenerador, la cual se encuentra sumergida y fijada para que pueda sostener toda la estructura, soportando las fuerzas del viento y del agua, así como la masa de la estructura.
- Equipo eléctrico: transporta electricidad desde el generador hasta el transformador.

Así como se muestra en la figura **Figura 2.1**

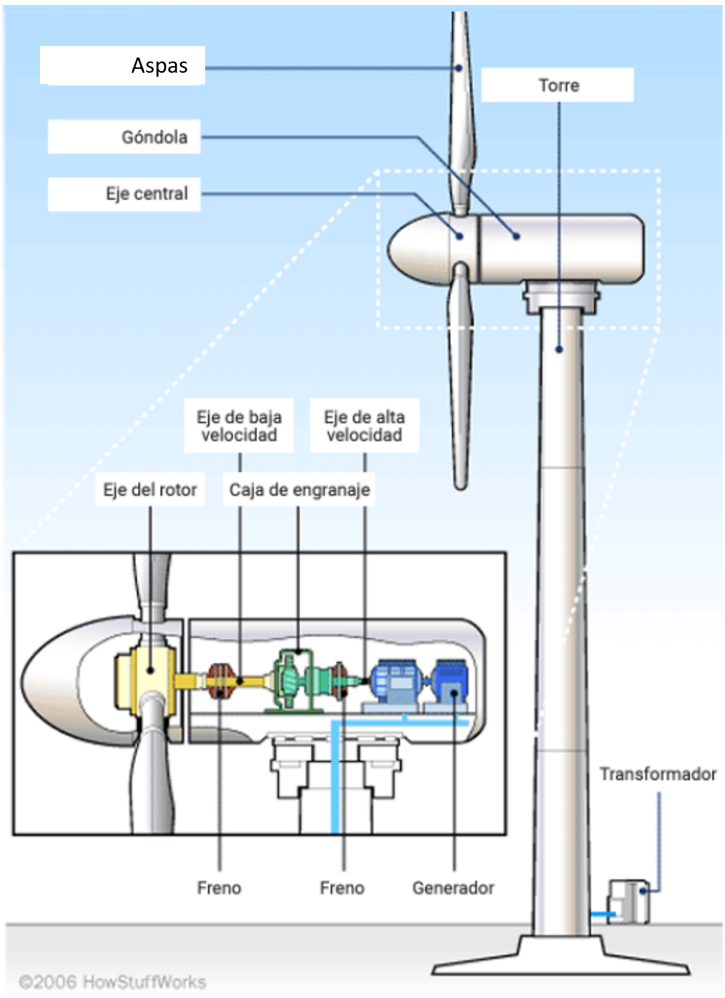


Figura 2.1: Partes de un aerogenerador. Recuperado de Layton (2006).

Análisis y diseño del soporte trípode

De acuerdo a la tesis de Bahamonde García y cols. (2019) vemos que en la Tabla 2.1 se muestra que en la actualidad los soportes tipo monopilote son los más utilizados por su bajo costo, pero solo pueden instalarse a profundidades no mayores de 25 metros; los de tipo trípode están en mayor crecimiento por instalarse hasta 50 metros. Esto limita las instalaciones en mar abierto, donde las profundidades son mayores, por lo que los tipos flotante por el momento están en fases de investigación por su capacidad de instalarse a profundidades mayores de 50 metros.

En la **Figura 2.2** se aprecia mejor la comparativa de profundidades de los diferentes tipos de estructuras *offshore*.

Denominación	**Profundidades**	**Aplicaciones en pp.ee.mm.** *(parques eólicos marinos)*
Base por gravedad (*gravity base*)	Hasta 15 m	Frecuentes
Monopilote (*monopile*)	Hasta 25 m	**Muy frecuentes**
Trípode (*tripod*)	De 20 a 50 m	**Crecientes**
Celosía (*jacket*)	De 20 a 50 m	**Crecientes**
Tripila (*tripile*)	De 25 a 40 m	Reducidas
Cámara de succión (*suction bucket*)	Hasta 30 m	Reducidas
Soportes flotantes (*floating platforms*)	Mayor de 50 m	En fase de investigación

Tabla 2.1: Tipos de cimentaciones *offshore* y sus aplicaciones. Recuperado de Bahamonde García y cols. (2019).

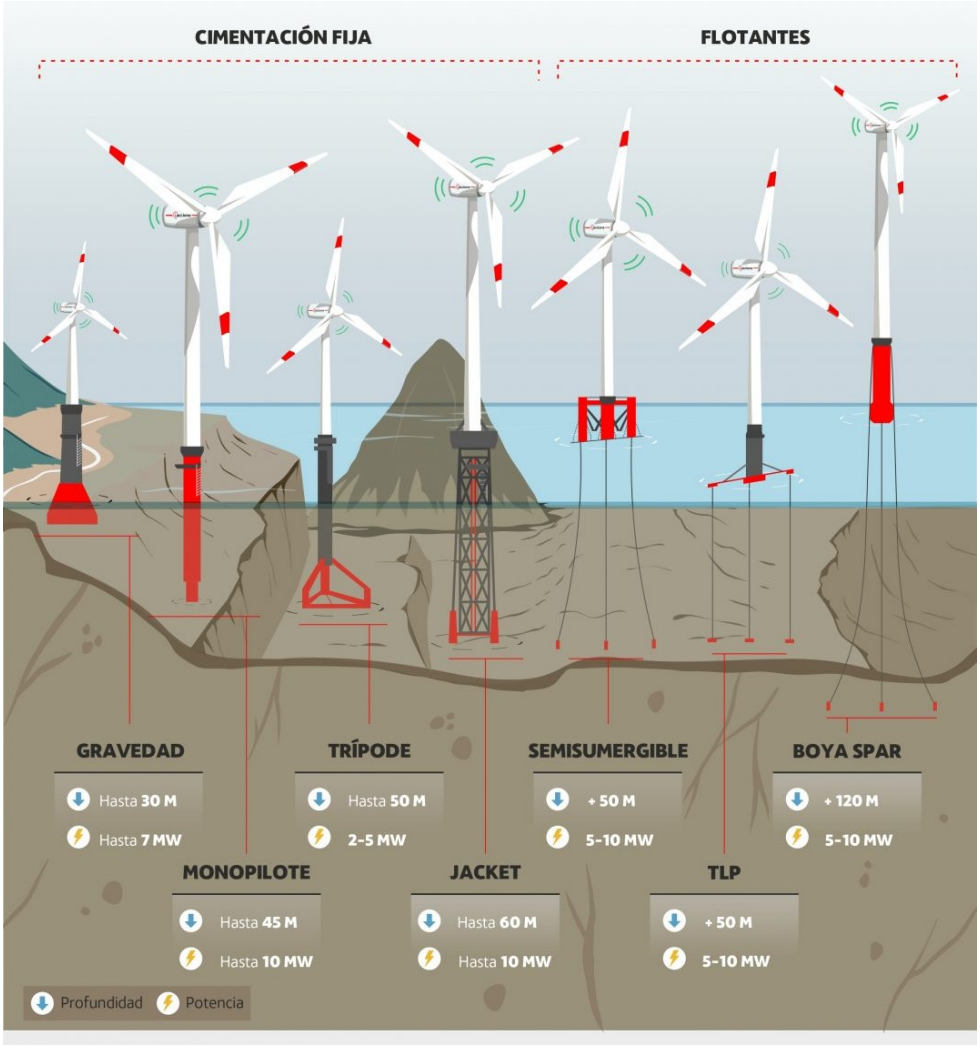


Figura 2.2: Tipos de estructuras *offshore*. Recortado y recuperado de Corporación de Desarrollo Tecnológico (2023).

Tomando como ejemplo uno de los parques pioneros en eólica *offshore* en Europa, tenemos el *"Alpha Ventus"* ubicado a 45 kilómetros mar adentro a profundidades mayores a 30 metros, iniciando con 12 aerogeneradores de 5 MW. Este parque contaba con 2 modelos, el tipo *Jacket* denominado REpower 5M y el tipo trípode M5000, con alturas de 90 metros sobre el nivel del mar y un peso de 1,000 toneladas. En su construcción se contó con el trabajo de 350 personas y 25 buques, pero para instalar estas estructuras se contó con 56 personas para tensar estas estructuras del tipo trípode, ocupando un área triangular de $255m^2$. Estas estructuras tienen la característica de que sus aspas pueden llegar a girar a una velocidad de 320 kilómetros por hora, con una separación de 800 metros entre cada aerogenerador, como lo describe Energetica21 (s.f.).

Por lo anterior vemos que la estructura trípode es una estructura muy usada en los parques eólicos, por su resistencia a profundidades de 50 metros, fácil ensamble frente a otro tipo de estructuras *offshore*, por lo que es importante tomarla como objeto de estudio y para nuestro propósito, como se va a utilizar en tierra para obtener la base de datos, sigue siendo funcional y de fácil instalación a diferencia de un tipo monopilote donde tendríamos que fijarla para cada prueba.

Partes de la estructura tipo trípode

Una estructura tipo trípode consiste en tres tubos o refuerzos en diagonal de acero soldados entre sí en un ángulo de 120°, los cuales proporcionan soporte a un tubo central que sostiene la torre que lleva la góndola **Figura 2.3**, los tubos del trípode en instalaciones *offshore* tienen un diámetro que varía entre 1 y 2.5 metros, abarcando una superficie triangular de entre $200m^2$ y $300m^2$ por lo que proporcionan buena rigidez y estabilidad contra el vuelco, recordar que un pilote es un elemento estructural que se coloca en el suelo para soportar cargas, como se menciona en Asociación Empresarial Eólica (AEE) (2022).

Un ejemplo donde se usan este tipo de estructuras son las que se encuentran en el parque eólico Borkum West II **Figura 2.3** de 400 MW situado a 45 km de la costa de la isla de Borkum en el Mar del Norte, alemán mencionado en Power Technology (2014).

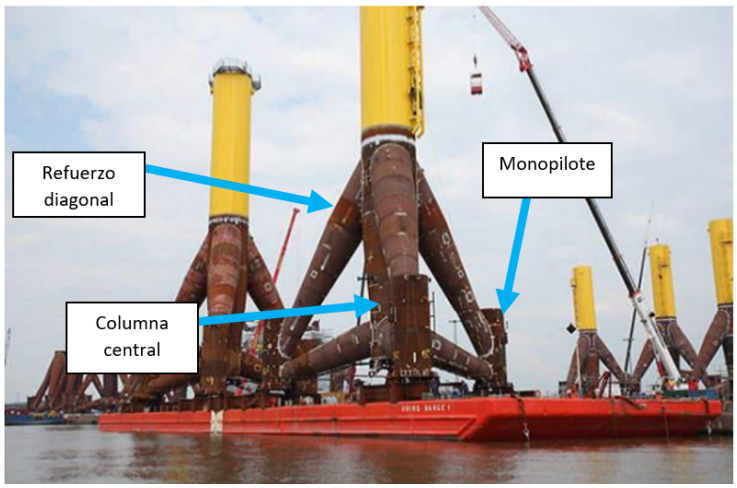


Figura 2.3: Partes de una estructura trípode real. Recuperado de Power Technology (2014).

Mencionar que es una estructura costosa en materiales y transporte por ser demasiado pesada de hasta 1,000 toneladas **Figura 2.4** pero en su ensamble y logística los costos son menores comparándola con otros tipos por ejemplo la tipo *Jacket*, entonces es considerada apropiada para turbinas de 4 a 5 MW. Otro lugar donde se pueden encontrar este tipo de estructuras tipo trípode es en los parques eólicos *offshore* de *"Bard en Dinamarca"*, mencionadas por Igartua San Segundo y cols. (2021).

Figura 2.4: Trasporte de base trípode. Recuperado de Alma de herrero (2010).

2.1.2. Estado saludable del aerogenerador

Condición en la que este se encuentra libre de desperfectos y conserva las características para las cuales fue diseñado y construido, en este estado, el equipo no presenta fisuras ni elementos sueltos, está correctamente instalado sobre una superficie nivelada y no exhibe vibraciones atribuibles a una mala instalación. Asimismo, se considera que no existen factores externos o interferencias de terceros que puedan alterar su equilibrio estructural y funcional, manteniéndose así dentro de los parámetros de diseño original.

2.1.3. Base de datos en bruto o crudo

Obtención de información tal como fue recopilada y almacenada, sin ningún tipo de procesamiento. Es decir, se trata del dato original, sin clasificación, limpieza ni análisis, correspondiente al estado en que se encuentra antes de ser interpretado para un propósito específico, Sanz Gómez (2018).

2.1.4. Caracterización de sensores

Proceso mediante el cual se evalúa y analiza su comportamiento para adecuarlo al propósito deseado, identificando las condiciones en las que funciona de manera óptima y asegurando su correcta integración en el sistema en el que se desea implementar. Este

proceso permite observar cómo responde el sensor, detectar posibles errores en su salida u operación y, en caso necesario, realizar su calibración para garantizar mediciones confiables, EstudioSmarter (2024).

2.1.5. Acelerómetro

Dispositivo electromecánico que es capaz de cuantificar los cambios de aceleración de un objeto y, dependiendo de su uso, tiene niveles de sensibilidad; la unidad de medida que se utiliza para estos dispositivos es en (g), Montes Fonseca (2019).

Por lo tanto, es un dispositivo que se usa para medir las vibraciones en un cuerpo de estudio, como lo menciona Manríquez Herrejón (2020), pues entre sus aplicaciones más populares se encuentran la detección de sismos, monitorización de máquinas rotatorias para localizar, detectar y/o prevenir grietas u otras fallas de las máquinas.

Como se mencionaba al principio de este trabajo, lo que se necesita monitorizar son las vibraciones presentes en el aerogenerador, debido a condiciones reales como el viento, la fatiga estructural y el desgaste por funcionamiento de sus componentes.

2.1.6. Vibración

Movimiento que presenta un cuerpo que está en equilibrio estático, provocando que salga de este estado. Si estas vibraciones se acumulan, con el tiempo el objeto o estructura llegará a un fallo prematuro, por lo que al monitorear estas vibraciones se pueden realizar predicciones de fallo, planificar mantenimientos que permitan la seguridad y reducción de costos en la estructura u objeto de estudio, National Instruments (2025).

2.2. Selección de sensores

Para elegir el sensor del tipo acelerómetro que más se adecuaba al propósito de esta tesis, se tomaron en cuenta las siguientes características:

Sensibilidad

Es la relación entre la señal de salida digital y la cantidad física que está midiendo el sensor (aceleración). Indica cuánto cambia el valor digital en respuesta a un cambio en la cantidad medida.

Expresado en unidades de LSB/g (*Least Significant Bit* por g) donde LSB es el bit menos significativo en el valor de salida digital del sensor, es decir, el valor más pequeño que puede representar el sensor en su salida digital; mientras que ”g” se refiere a la gravedad es decir $1\text{g} = 9.81\ m/s^2$, esto se usa en acelerómetros digitales y representa cuantos digitales (valores discretos) corresponden a 1g de aceleración.

Por ejemplo un acelerómetro con una sensibilidad de 16,384 LSB/g significa que un valor de 16,384 en su registro interno equivale a una aceleración de 1g.

Número de ejes

Esto se refiere a los ejes (X, Y, Z), en un acelerómetro ya que se puede encontrar de uno o más de tres ejes, pero tener en cuenta que si dice 6 ejes, los tres ejes de más son para el giroscopio o para alguna función extra independiente del acelerómetro.

Rango ajustable

En un acelerómetro, los valores de ±8g, ±16g, ±32g, se refieren al rango de medición de aceleración que el dispositivo puede detectar; como se mencionó anteriormente el símbolo "g" representa la aceleración debida a la gravedad de 9.81 m/s^2. Entonces si se tiene por ejemplo ±8g significa que el acelerómetro puede medir aceleraciones entre -8g y +8g, lo que corresponde a un rango de -78.48 m/s^2 a + 78.48 m/s^2.

Por lo tanto el rango ajustable permite cambiar la sensibilidad del acelerómetro para adaptarse a diferentes aplicaciones. Para situaciones que involucran movimientos suaves o pequeñas aceleraciones, se puede usar un rango más bajo (±8g) para obtener mediciones más precisas. En cambio, para movimientos bruscos o impactos, es mejor usar un rango más alto (±64g) para evitar saturar el sensor.

Protocolo de comunicación

- **I^2C**

Es un protocolo de comunicación serial que permite que varios dispositivos conectados compartan información utilizando solo dos líneas:

- SDA (Serial Data Line): para transmitir los datos.
- SCL (Serial Clock Line): para la sincronización entre dispositivos.

I^2C es muy común en aplicaciones de sensores, como los acelerómetros, ya que permite la comunicación sencilla, eficiente entre un microcontrolador y conectar uno o más dispositivos periféricos.

En este caso, al transmitir datos de un acelerómetro de manera inalámbrica, I^2C es una buena opción para que el sensor se comunique con un microcontrolador que luego envíe esa información. Es un protocolo de baja velocidad pero muy eficiente para entornos con distancias cortas y múltiples dispositivos. I^2C es flexible porque soporta diferentes velocidades de transmisión (por ejemplo, desde 100 kHz hasta 3.4 MHz en modos avanzados) y permite conectar varios dispositivos en el mismo bus, identificados por sus direcciones únicas como menciona Carletti (2007).

Ventajas del protocolo I^2C

Este protocolo permite la conexión de una amplia variedad de periféricos sin la necesidad de direccionamiento independiente o señales de habilitación del chip, como lo menciona NXP Semiconductors (2021).

- **SPI**

Es un protocolo de corto alcance ya que solo permite de 20 a 30 cm usando 4 hilos de comunicación, tiene una velocidad de hasta 50 MHz como lo expresa J. Chen y Huang (2023).

El *Serial Peripheral Interface* (SPI) es un protocolo de comunicación serial síncrono ampliamente utilizado para la transferencia de datos de alta velocidad entre microcontroladores y dispositivos periféricos como sensores, pantallas y memorias. Funciona con una arquitectura maestro-esclavo, donde el dispositivo maestro controla la comunicación generando señales de reloj y seleccionando los dispositivos esclavos.

Señales principales de SPI:

- SCLK (*Serial Clock*): Generado por el maestro para sincronizar la transmisión de datos.
- MOSI (*Master Out Slave In*): Transporta datos del maestro al esclavo.
- MISO (*Master In Slave Out*): Transporta datos del esclavo al maestro.
- SS (*Slave Select*): Señal activa en bajo que el maestro usa para seleccionar un esclavo específico.

SPI es una opción popular para sistemas embebidos que requieren comunicación rápida y de doble vía en distancias cortas (Pini (2019)).

Bits de resolución

Los bits en un acelerómetro se refieren a la resolución del convertidor analógico-digital (ADC) que convierte la señal analógica del sensor en datos digitales que pueden ser interpretados por un microcontrolador o una computadora. Estos sirven para la resolución de medición por ejemplo

- Un acelerómetro con una resolución de **16 bits** puede representar $2^{16} = 65,536$ valores distintos.
- Uno de **12 bits** solo puede representar $2^{12} = 4,096$ valores.
- Mientras más bits tenga, **más preciso será el valor** de la aceleración medido.

También sirve para la precisión en los valores de aceleración, por lo que un acelerómetro con más bits puede detectar cambios más pequeños en la aceleración.

Ejemplo: Si un acelerómetro de 16 bits mide un rango de $\pm 2g$, la resolución es:

$$\frac{4g}{65,536} = 0.000061g \quad \text{(61 µg por bit)}$$

En cambio, un acelerómetro de 12 bits con el mismo rango tiene una resolución menor:

$$\frac{4g}{4,096} = 0.00098g \quad \text{(980 µg por bit)}$$

Un sensor con más bits generalmente tiene **menos ruido** en la medición, lo que mejora la estabilidad de los datos.

Por lo tanto, es mejor elegir un acelerómetro con mayor resolución (más bits) esto para obtener medidas más precisas, ya que los bits en un acelerómetro afectan la precisión, sensibilidad y capacidad de detectar pequeñas variaciones de aceleración.

Teniendo como referencia lo anterior, se tomaron las siguientes propuestas existentes en el mercado **Anexo A.9**, pero los más sobresalientes son los mostrados en la tabla 2.2.

Carac.	**ADXL345**	**MPU9250**	**ADXL362**	**MMA8451**	**ADXL355Z**
Sensibilidad (LSB/g)	256	16,384	1,000	4096	256,000
Número de ejes	3	3	3	3	3
Rango ajustable (g)	±2, ±4, ±8, ±16	±2, ±4, ±8, ±16	±2, ±4, ±8	±2, ±4, ±8	±2, ±4, ±8
Protocolo de comunicación	SPI, I2C	SPI, I2C	SPI	I2C	SPI, I2C
Resolución	10 a 13 bits	16 bits	12 bits	14 bits	20 bits

Tabla 2.2: Comparativa de sensores del tipo acelerómetro

Se decidió trabajar con el sensor MPU9250 **Figura 2.5** ya que como se observa en la tabla, presenta buenas características y ademas es económico en comparación del ADXL355Z.

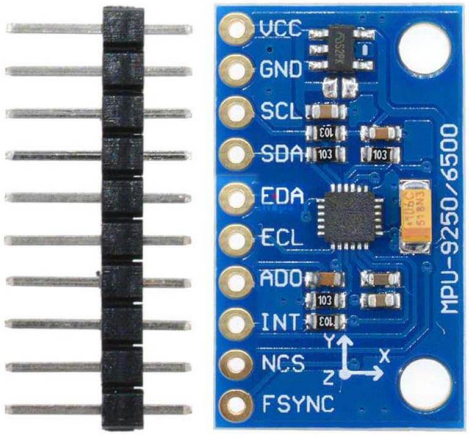


Figura 2.5: Acelerómetro MPU9250. Recuperado de UElectronics (s.f.).

Capítulo 3

Diseño y construcción de la estructura del aerogenerador con base trípode

El aerogenerador se compone de una base tipo trípode, una torre y una góndola, como se aprecia en la **Figura 3.1**.

La construcción del aerogenerador tipo trípode contará con una altura de 4.30 m, tomando en cuenta o buscando un equilibrio entre materiales comerciales, costo de los mismos, fácil ensamble y espacio para manipularla, pero sin dejar de lado la calidad de funcionamiento; además, apoyándonos de las medidas de la plataforma usada por Aquino González (2020), vemos que consta de una altura de 2.7 m, por lo que se aumentará esta altura además de convertirlo propiamente en un aerogenerador funcional, favoreciendo condiciones más reales para realizar lecturas de vibraciones.

Figura 3.1: Modelo a construir. Recuperado de Cathwell (s.f.).

3.1. Base trípode en CAD

El diseño asistido por computadora (CAD) puede ahorrar tiempo, esfuerzo y dinero, ya que nos permite realizar prototipos, para visualizar cómo quedaría el producto final, saber si el diseño es bueno, tanto funcional como en resistencia; esto se hace por medio de análisis estático, lo que es una relación entre el esfuerzo y la deformación, denominada módulo de elasticidad, así como el límite de elasticidad, los cuales están determinados por la estructura molecular del material.

La base trípode se construyó a una escala de 1:38, como se muestra en la Tabla 3.1, en dicha tabla, la columna correspondiente a la escala 1:38 presenta las dimensiones del prototipo, mientras que la columna con relación 1:1 muestra las dimensiones de un aerogenerador *offshore* a tamaño real.

BASE TRÍPODE		
Relación	**01:38**	**01:01**
Prototipo	1.3 m	50 m
Equivalencia	1 m	38.46 m

Tabla 3.1: Relación de escalado para base trípode.

3.2. Justificación de la carga aplicada a base trípode

Se evaluó la base para una carga vertical de 1000 N aplicada normal a la cara superior de la placa. Esta carga representa el peso de la torre y la góndola de $\approx 100\,\text{kg}$.

Entonces aplicando la siguiente formula se tiene como resultado:

$$W = m \cdot g = (100\,\text{kg})(9.81\,\text{m/s}^2) = 981\,\text{N} \tag{3.1}$$

Redondeado a 1000 N para cubrir variaciones e incertidumbres. La carga se distribuyó uniformemente sobre el área de apoyo, los apoyos en las bases se modelaron como fijaciones, y se incluyó la gravedad para el peso propio de la base.

3.2.1. Versión uno

La primera versión que se realizó **Figura 3.2** tenia los siguientes elementos:

1. Columna central: tubular de 4 pulgadas con 1.30 m de longitud
2. Refuerzos laterales: 3 barras de ángulo de pulgada y media con longitud de 1.50 m.
3. Refuerzos inferiores interiores: 3 barras de ángulo de 1 m de longitud.
4. Refuerzos inferiores exteriores: 3 barras en ángulo de longitud 1.5 m.
5. Aletas: 6 placas en triángulo colocadas en la parte inferior y superior para unir los refuerzos laterales e inferiores internos.
6. Soporte para torre: placa de 5 mm de espesor con dimensiones de 32 cm x 32 cm con barrenos de 3/4 de pulgada.

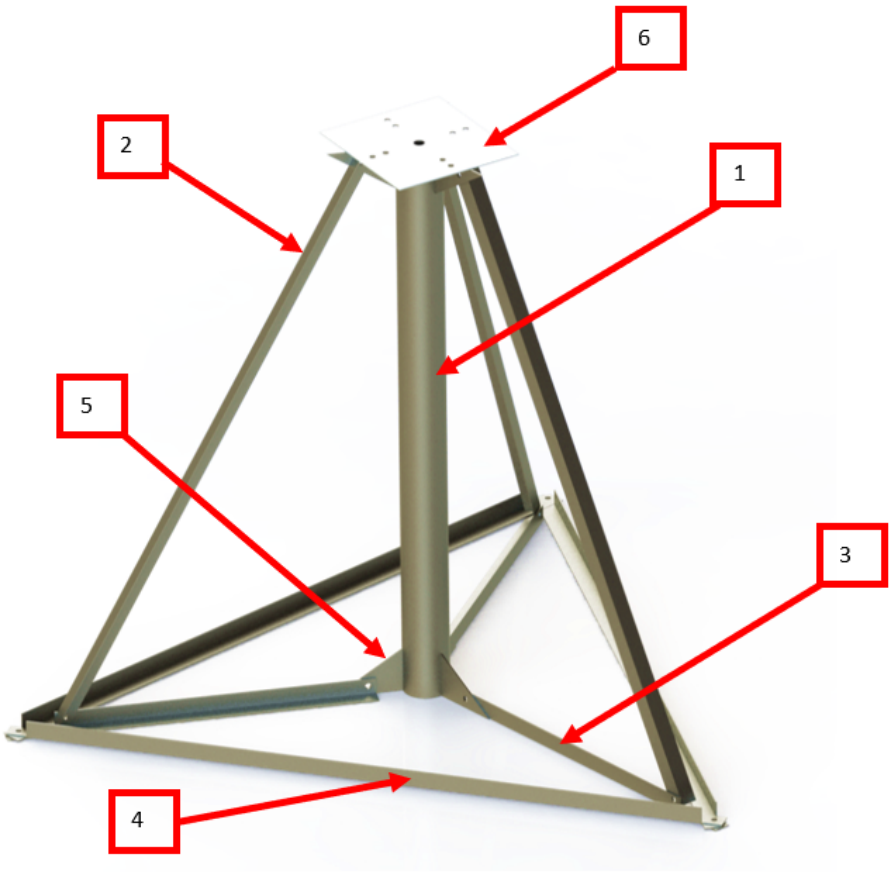


Figura 3.2: Elementos de la versión 1, base trípode.

3.2.2. Análisis estático de Versión uno

Al realizar el análisis se obtuvieron los siguientes resultados, primero se le aplicaron 1000 N de carga sobre el soporte, en donde se sostendrá la torre y la góndola, el material se deforma en lo que es la placa donde se coloca la torre por lo que se reforzó esa placa para reducir el desplazamiento y deformación.

El material usado fue acero A36, por ser el más comercial, además de contar con las siguientes propiedades en SolidWorks Tabla 3.2 y tener un buen límite elástico.

Comparando estas propiedades con la tabla del libro de mecánica de materiales del autor Beer y cols. (2010), vemos que coinciden en el límite elástico Tabla 3.3 lo que quiere decir que es un material con buena rigidez.

Tensiones

Al ejecutar el análisis estático obtenemos los siguientes resultados en la sección de tensiones como se observa en la **Figura 3.3**, la mayor tensión se presenta en el soporte para la torre pero además vemos que el valor máximo en límite elástico es de $2.500e+08N/m^2$ pero en el diseño solo llega hasta 4.598e+07N/m^2, por lo que no supera el límite elástico del material.

Propiedad	Valor	Unidades
Módulo elástico	200000	N/mm^2
Coeficiente de Poisson	0.26	N/D
Módulo cortante	79300	N/mm^2
Densidad de masa	7850	kg/m^3
Límite de tracción	400	N/mm^2
Límite de compresión	—	N/mm^2
Límite elástico	250	N/mm^2
Coeficiente de expansión térmica	—	/K
Conductividad térmica	—	$W/(m \cdot K)$
Calor específico	—	$J/(kg \cdot K)$

Tabla 3.2: Propiedades de acero A36 en SolidWorks.

Material	Densidad kg/m^3	Res. última Tensión (MPa)	Cedencia Tensión (MPa)	Módulo de elasticidad (GPa)
Acero Estructural (A36)	7860	400	250	200

Tabla 3.3: Propiedades mecánicas de acero A36. Recuperado de Beer y cols. (2010).

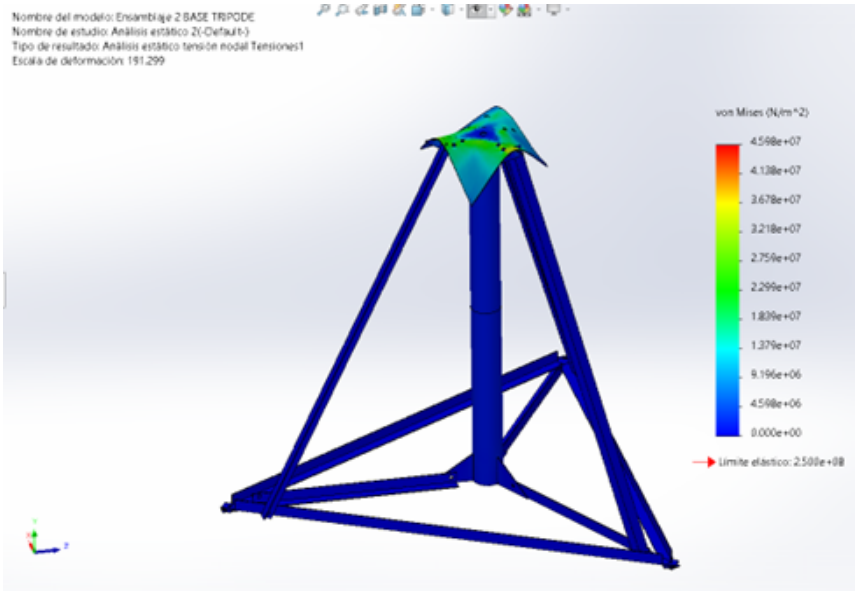


Figura 3.3: Versión 1 tensión.

Desplazamientos

En la **Figura 3.4** se muestra el desplazamiento máximo, que es de 8.417e-01 mm (0.8417 mm), por lo que es razonablemente bajo para ser acero con una carga de 1000 N.

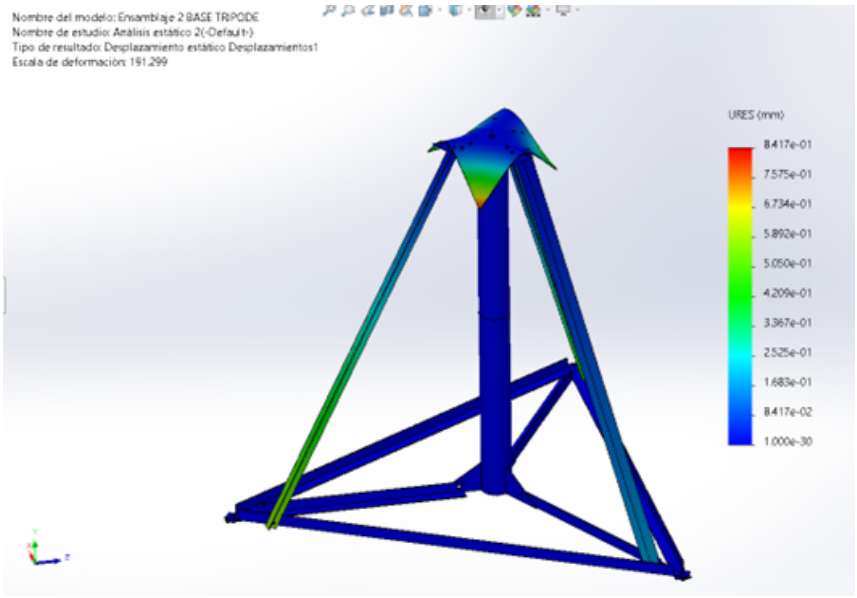


Figura 3.4: Tensión en la base trípode.

3.2.3. Versión dos

En la versión 2 **Figura 3.5** con sus respectivas medidas en la **Figura 3.6** se mejoró el soporte de la torre, esto para que fuera una estructura más robusta y donde fuera fácil colocar la torre ya que contaba con mayor área en el soporte para la torre.

Figura 3.5: Versión 2.

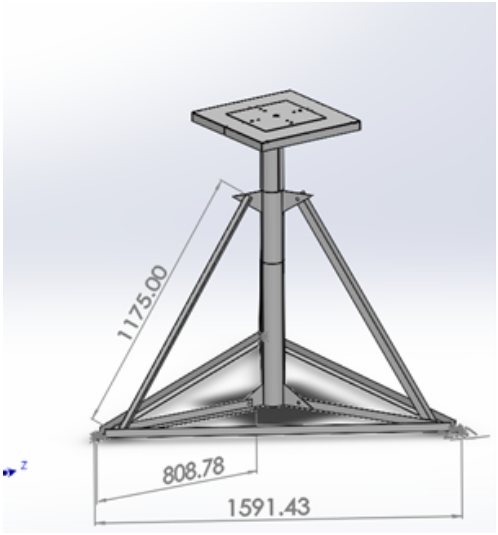


Figura 3.6: Medidas de la versión 2.

El peso de la estructura de la versión 2 es de 40.42 Kg como se muestra en la **Figura 3.7**.

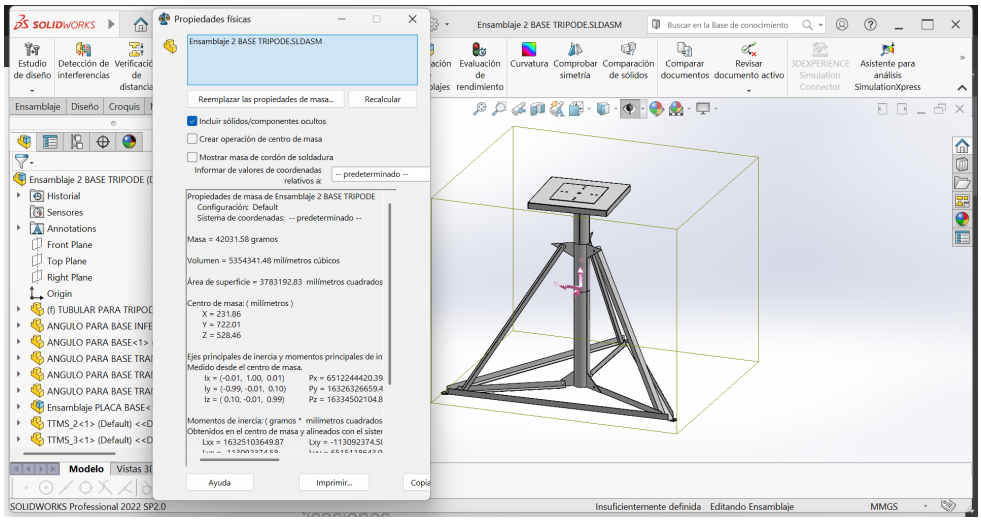


Figura 3.7: Peso de la versión 2 de la estructura trípode.

3.2.4. Análisis estático de Versión dos

Tensiones

Se aplicó 1,000 N sobre el soporte donde se acopla la torre, entonces se obtuvieron los siguientes resultados **Figura 3.8**, primero tenemos el valor del límite elástico de 2.500e+08 N/m^2 igual a 250,000,000 N/m^2, el cual indica el valor en que el material sufrirá una deformación permanente comparándolo con el valor de la simulación el cual es 9.202e+06 N/m^2 igual a 9,202,000 N/m^2, vemos que es menor que el del límite elástico por lo que el material está dentro de la región elástica, es decir, el material se deformará, pero recuperará su forma cuando se retire la carga. Esto indica que la pieza está soportando la carga de 1000 N sin problema y aún tiene margen antes de llegar al punto donde comenzaría la deformación plástica o también llamada deformación permanente.

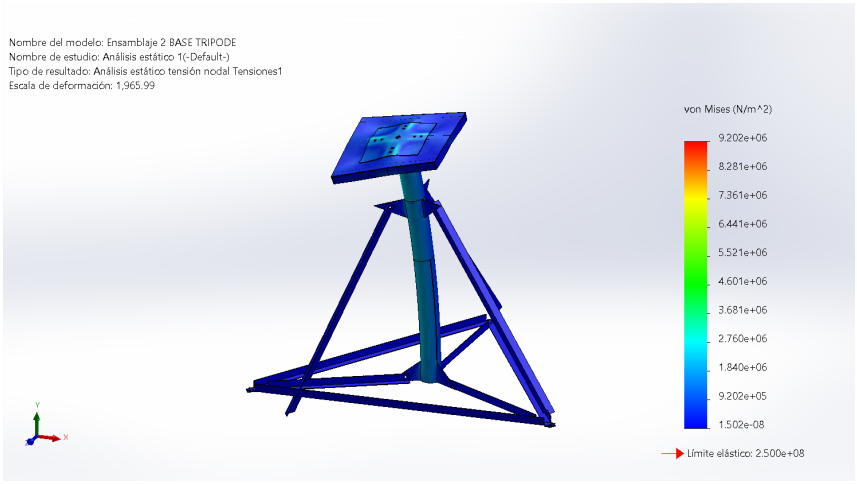


Figura 3.8: Tensión en la base trípode, versión 2.

Desplazamientos

Para la sección de desplazamiento se obtuvieron los siguientes resultados **Figura 3.9** donde encontramos el siguiente valor de 1.160e-01 mm igual a 0.1160 mm, que comparándola con la versión uno vemos que hay una disminución de 0.7257 mm, por lo que esta versión es mejor porque se desplaza menos bajo la misma carga.

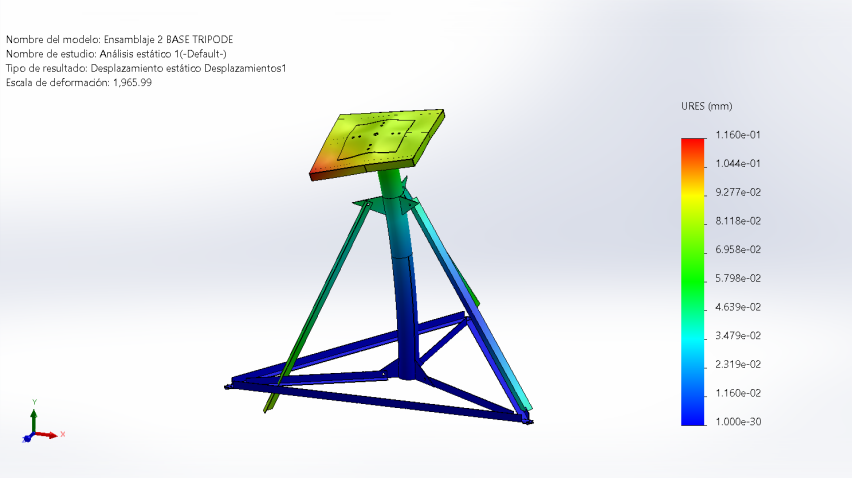


Figura 3.9: Desplazamiento en la base trípode.

3.2.5. Versión tres

Se tuvo que realizar una nueva versión ya que por disponibilidad en el mercado se utilizó el siguiente material, para la columna central se usó cuadrado de 4 pulgadas, tubulares de 1 ½ pulgadas para los refuerzos laterales, tubo de 1 ¼ de pulgada para los inferiores interiores y exteriores **Figura 3.10**.

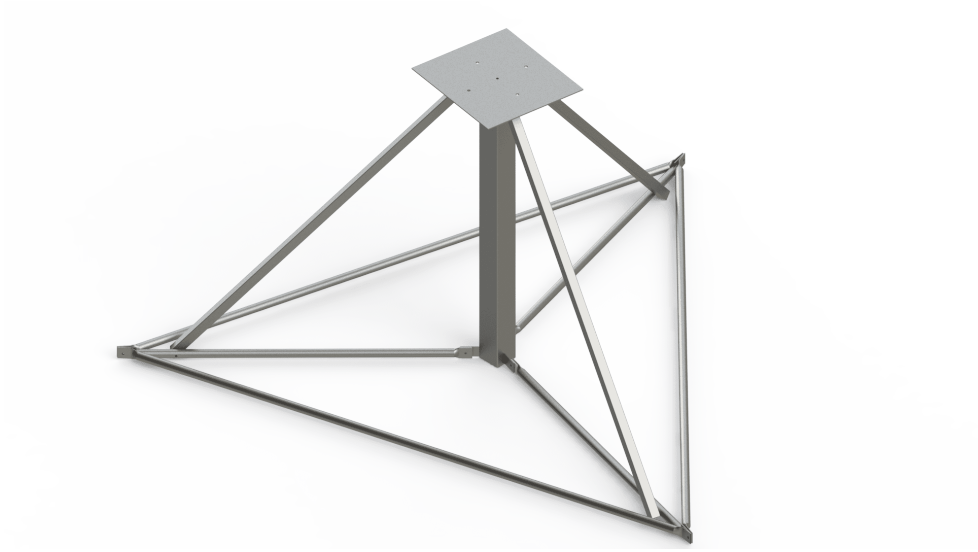


Figura 3.10: Versión 3 de base trípode.

Por lo que esta versión de acuerdo al software nos indica que tenia un peso de 57484.97 gramos o 57.484 Kg **Figura 3.11**

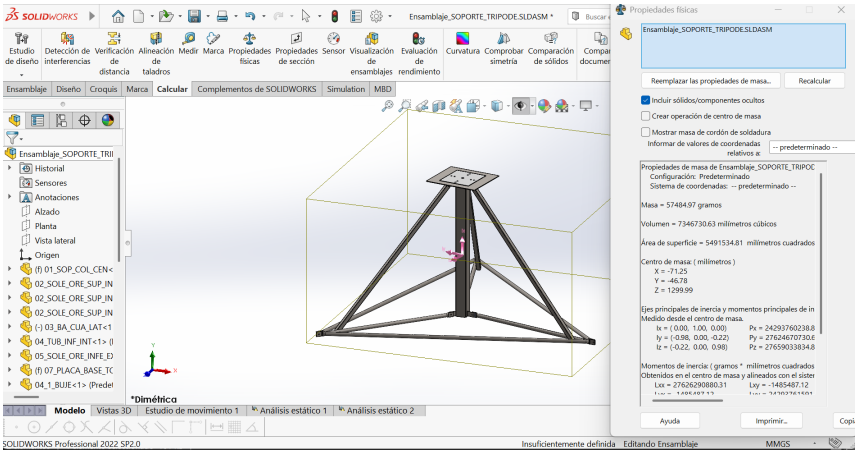


Figura 3.11: Peso de la versión 3 de base trípode.

3.2.6. Análisis estático de Versión tres

Tensiones

Al aplicar la misma carga de 1000 N se obtuvo un valor de 1.933e+07 N/m^2 lo que es igual a 19,330,000 N/m^2 esto quiere decir que la estructura es más resistente porque está trabajando con menos esfuerzo respecto a su capacidad máxima, indicando que la estructura está en condiciones más seguras bajo la carga aplicada. **Figura 3.12**

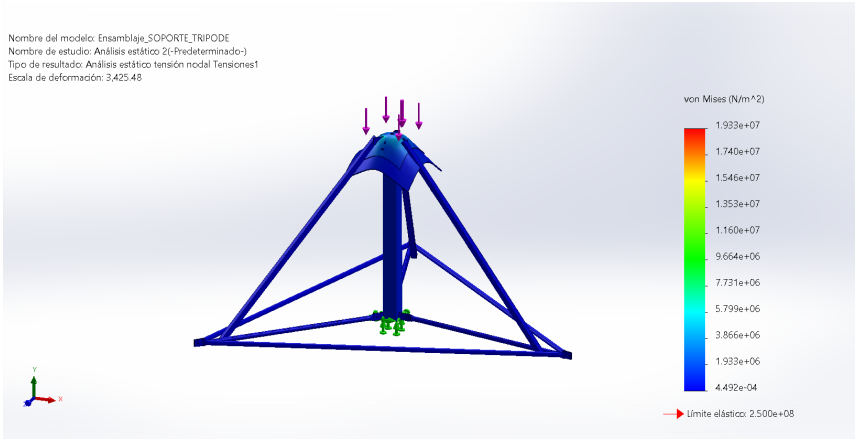


Figura 3.12: Tensión en la versión 3 de base trípode.

Desplazamientos

Esta versión presenta menos desplazamiento ya que en la simulación se obtuvo un valor de 8.759e-02 mm igual a 0.08759 mm **Figura 3.13**, lo que la hace una estructura estable

ya que se desplaza 0.02841 mm menos con respecto a la versión 2, lo que se traduce en menor deformación.

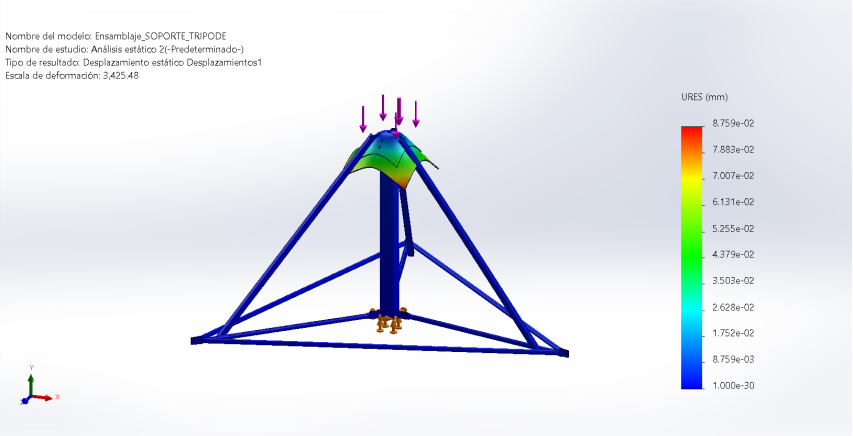


Figura 3.13: Desplazamiento de base trípode en la versión 3.

3.3. Aerogenerador en CAD

Después de tener la base en CAD, se continuó con el ensamble completo del aerogenerador, es decir, torre y góndola **Figura 3.14**.

Figura 3.14: Aerogenerador terminado.

3.4. Justificación de la carga aplicada a torre

Modelo

Se modela la torre como un cilindro vertical de diámetro exterior D y altura expuesta h, sometido a viento uniforme de velocidad V perpendicular al eje de la torre. Se adopta coeficiente de arrastre C_d constante a lo largo de la altura. El flujo se considera estacionario

(análisis estático).

Presión dinámica del viento

La presión dinámica del viento se define como:

$$q \;=\; \tfrac{1}{2}\,\rho\,V^2 \tag{3.2}$$

Área proyectada del cilindro

El área proyectada frontal del cilindro ("lo que ve el viento") es:

$$A \;=\; D\,h \tag{3.3}$$

Fuerza lateral total sobre la torre

La fuerza aerodinámica lateral se obtiene multiplicando la presión dinámica por el área proyectada y por el coeficiente de arrastre del cilindro:

$$F \;=\; q\,C_d\,A \;=\; \left(\tfrac{1}{2}\,\rho\,V^2\right)C_d\,(D\,h) \tag{3.4}$$

La carga de **500 N** aplicada en la torre se justifica como la fuerza lateral equivalente del viento. De acuerdo con la ecuación de presión dinámica del viento:

$$q = \frac{1}{2}\rho V^2 \tag{3.5}$$

Considerando la torre tubular de **4.3 m** de altura y **0.1016 m** de diámetro (4 pulgadas), con coeficiente de arrastre $C_d = 1.2$ de acuerdo a White (2011), $\rho = 1.225\ kg/m^3$ (densidad del aire) y un viento extremo de **35 m/s** (126 km/h), se obtiene:

Presión dinámica del viento

$$q \;=\; \tfrac{1}{2}\,\rho\,V^2 \tag{3.6}$$

$$q \;=\; \tfrac{1}{2}\,(1.225)\,(35)^2 = 750.31 N/m^2 \tag{3.7}$$

Área proyectada

$$A \;=\; D\,h = (0.1016m)(4.3m) = 0.43688m^2 \tag{3.8}$$

Sustituyendo en la fórmula de fuerza se obtiene:

$$F = q \cdot C_d \cdot A \tag{3.9}$$

$$F = (750.31 N/m^2)(1.2)(0.4368m^2) \;= 393.28\ \text{N} \tag{3.10}$$

Por lo tanto, se consideró aplicar **500 N** en lugar de 393.28 N como condición de carga crítica para representar la acción del viento sobre la torre en el análisis estático.

Tensiones

Con el ensamble completo se realizó un análisis estático aplicando una carga axial de 500 N en la torre, simulando la carga por el viento, el cual el resultado en tensión es el mostrado en la **Figura 3.15**, como se ve el límite elástico es de 2.5e+08 N/m^2 y al aplicarle la carga, se tiene un límite de 1.18e+08 N/m^2 lo cual indica que aún está dentro del rango del límite elástico antes de sufrir una deformación permanente.

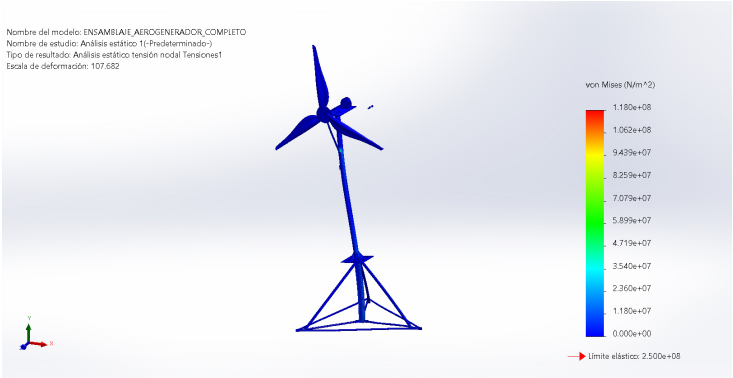


Figura 3.15: Tensión en el aerogenerador.

Desplazamientos

En el desplazamiento se tuvo un resultado de 5.538e+00 mm, en donde se observa en la **Figura 3.16** que el mayor desplazamiento se encuentra en la parte superior donde va la góndola.

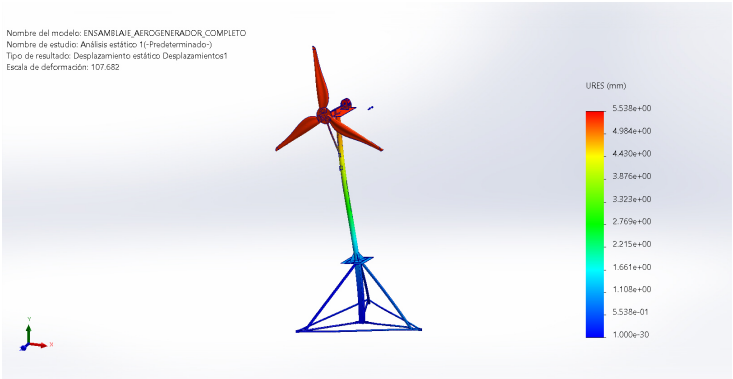


Figura 3.16: Desplazamiento en aerogenerador.

3.5. Construcción de la base trípode

Columna central

El plano de la columna central, se observa en el **Anexo A.1**, debe tener una altura de 1300 mm y un ancho de 101.60 mm o 4 in, con espesor de 4 mm, se eligió por ser resistente de acuerdo a nuestra tabla de materiales vista anteriormente, además de ser PTR comercial **Figura 3.17**.

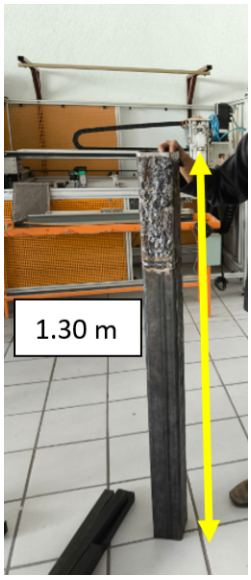


Figura 3.17: Resultado de la columna central.

Refuerzos laterales

Las tres refuerzos tienen una longitud de 1.70 m como se muestra en el plano **Anexo A.2**.

Se marcó con un punzón en donde se hicieron los barrenos en los tres cuadrados que se usaron para los refuerzos laterales con las siguientes medidas:

- 2 cm desde la punta y 1.8 cm a lo ancho para centrar la marca, con la finalidad de que al barrenar no se mueva la broca de 11/32 de pulgada **Figura 3.18**.

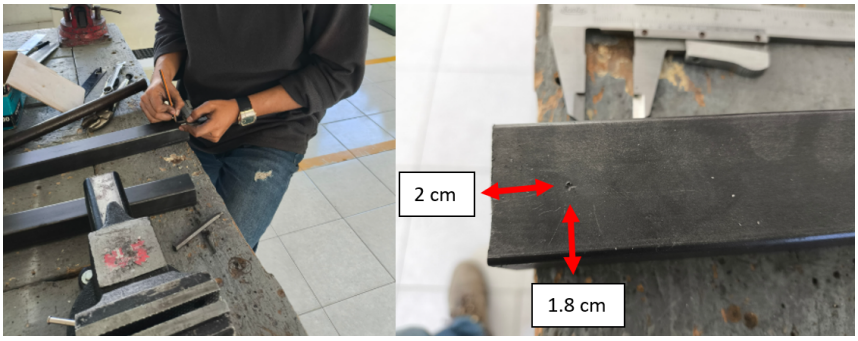


Figura 3.18: Marca para barreno.

Se barrenó con una broca 11/32 de pulgada en el taladro de banco, sin olvidar poner el nivel, esto para que el barreno saliera recto y a la misma distancia en ambas caras del tubular por donde pasa la broca, ya que si no esta nivelado existe el riesgo de que el barreno no esté centrado, provocando que no entre el tornillo con naturalidad.

Solera tipo aleta

Se elaboraron 6 soleras tipo aleta, las cuales sujetan los refuerzos laterales y tubulares inferiores internos.

Se marcaron con las siguientes medidas, 7.5 cm de alto por 4.5 cm de ancho, el barreno se situó a 2 cm desde la orilla y 2.25 cm a lo ancho, como se muestra en el plano **Anexo A.3**.

Siguiendo el proceso, primero se realizó la marcación sobre el metal con lápiz y punzón; después, el barrenado en taladro de banco y, por último, cortar con arco de segueta.

Con esto se obtienen las seis soleras iguales donde 3 se soldaron en la parte superior y las otras 3 restantes en la parte inferior de la columna central **Figura 3.19**.

Figura 3.19: Solera tipo aleta para barras laterales.

Tubulares inferiores internos

Estos tres tubulares tienen una longitud de 1.28 m y un diámetro de 3.4 cm como se muestra en el **Anexo A.4**.

Con un marro se prensa un lado del tubular, tomando 5 cm desde la orilla. Después se barrenan a 2.5 cm desde la orilla **Figura 3.20**.

Figura 3.20: Barrenado en tubular.

Etapa de ensamble base trípode

Para poder soldar las soleras tipo aleta a la columna, se marcó en una tabla un círculo con marcas a 120 grados.

Después, la columna central se colocó dentro del círculo y se procedió a soldar las aletas, las inferiores se colocaron a una altura de 5 mm desde el nivel del piso y los superiores a 1 cm, **Figura 3.21**.

Figura 3.21: Aletas unidas.

Ya que se tienen unidas las soleras con la columna central, se procede a colocar los refuerzos laterales y tubulares inferiores internos con sus respectivos tornillos 5/16 de pulgada, en la parte superior los tornillos tienen un largo de 2 1/2 de pulgada y para los inferiores 1 pulgada de largo, **Figura 3.22** y **Figura 3.23**.

Figura 3.22: Ensamble parte superior.

Figura 3.23: Ensamble parte inferior.

Para centrar la columna, se mide desde la cara de la columna central hacia la esquina del cuadrado que asienta sobre el piso. En este caso se midieron las tres distancias, las sumamos y al dividirlas da como resultado 116.16 cm, esta medida es la distancia para que la columna estuviera centrada, después se colocan unas prensas para que no se muevan de la medida y procedemos a realizar un molde de papel para que el barreno quede en medio, tanto del refuerzo lateral como el del tubular inferior interno como se ve en la **Figura 3.24**.

Figura 3.24: Medida correcta.

El siguiente molde se hace de la siguiente manera: se mide el perfil cuadrado y se plasma en papel. como se sabe el diámetro o ancho del tubo (34 mm), se coloca una línea a la cual se le saca la mitad y ahí es donde se barrena, **(Figura 3.25**).

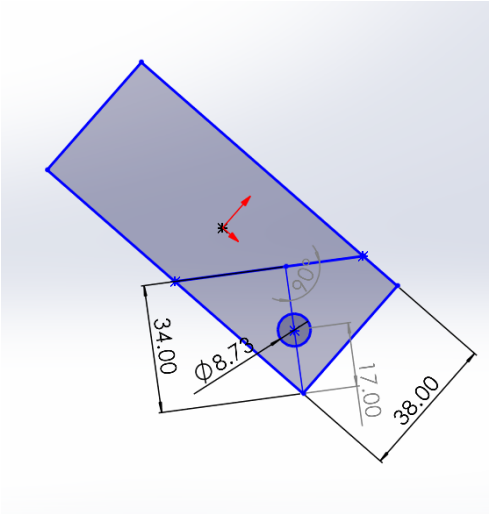


Figura 3.25: Medidas del molde.

Dicho lo anterior, se coloca el papel sobre el PTR cuadrado (refuerzo lateral) y con un punzón hacemos la marca para barrenar con una broca 11/32 de pulgada, esto se hace para los tres refuerzos.

Ya que se ha realizado el barreno, se coloca el refuerzo lateral en la columna. Ahora, como ya tiene barreno, el refuerzo lateral permite marcar con un lápiz dónde irá el barreno para el tubular inferior interno, **Figura 3.26**, esto respetando los 116.16 cm, como se menciona anteriormente, de la cara de la columna central a la arista que toca el suelo del PTR cuadrado.

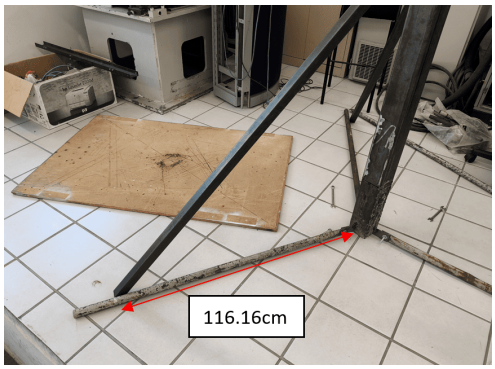


Figura 3.26: Refuerzo lateral unido a la columna central.

Se realiza un molde de cartón con el diámetro del tubular el cual es 34 mm, después se retira el centro del cartón, ya que ahí entrará el tubular, esto para que el barreno salga lo más centrado posible, **Figura 3.27**.

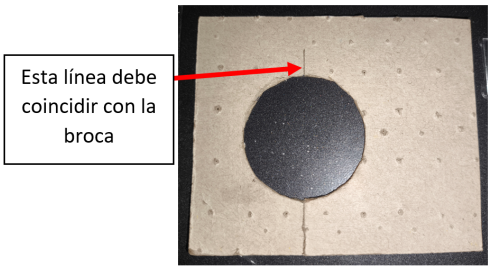


Figura 3.27: Molde para el tubular interno.

De igual forma, se coloca el nivel para barrenar, véanse **Figura 3.28**. Esto se realiza en los tres tubulares inferiores interiores. Asimismo, se puede marcar el lugar donde estaban las prensas, ya que al barrenar no es posible dejarlas por falta de espacio. De esta manera, al marcar su posición, al colocar los tornillos se podrá verificar si hubo algún desplazamiento o si el barreno se encuentra en la ubicación correcta.

Figura 3.28: Barrenado de tubular.

Una vez que se han barrenado los tubulares internos, se colocan en sus respectivos

lugares con sus tornillos, para estos tubulares inferiores internos y barras laterales se colocan tornillos de diámetro 5/16 de pulgada por 4 pulgadas de largo, **Figura 3.29**.

Figura 3.29: Tubular unido al PTR con tornillo.

Soleras tipo 2 para tubular inferior externo

Se realizaron las siguientes 3 soleras con las siguientes medidas, **Figura 3.29** y **Anexo A.5**.

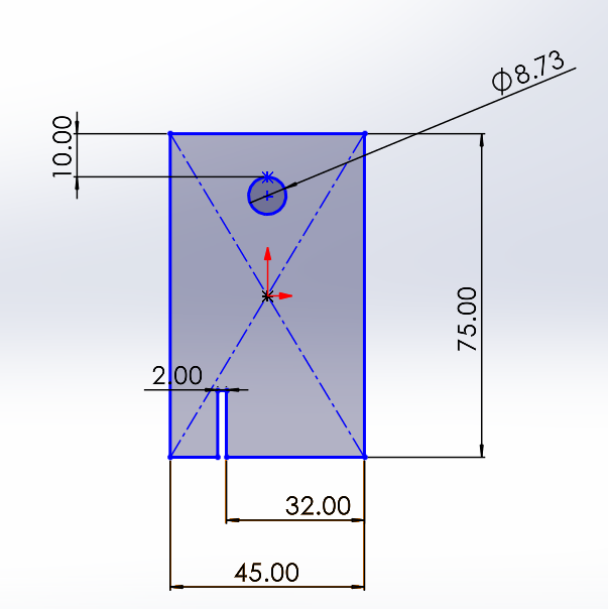


Figura 3.30: Plano de solera tipo 2.

Primero se cortaron las soleras como se muestra en la **Figura 3.31** en donde se ve que debe entrar hasta donde se ha cortado para que entre el espesor del tubular .

Figura 3.31: Ensamble de comprobación.

Después se marca la solera con punzón y se barrena, se deja 1 cm de margen para el barreno y se usa una broca de 11/32 de pulgada (8.73 mm) como se muestra en la **Figura 3.32**.

Figura 3.32: Resultado.

Ya que se tienen las 3 soleras cortadas, se procede a soldar cada una a un extremo de cada tubular inferior interno, esto se refiere a los que miden 1.28 m, **Figura 3.33**.

Figura 3.33: Solera vista frontal.

Tubulares inferiores externos

Son tres tubulares de 2.525 m de largo ya con el doblez de 5 cm en cada lado a 150 grados, como se ve en el plano, **Anexo A.6**.

Para ajustar los tubulares inferiores externos, primero se mide la distancia que hay entre las soleras que se encuentran en los tubulares inferiores internos, se suman esas distancias, el resultado se divide entre 3 y se obtiene que el largo debe ser de 2.525 m, como se muestra en la **Figura 3.34**.

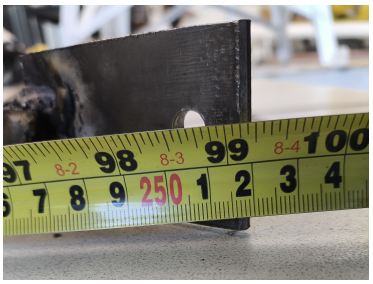


Figura 3.34: Distancia final.

Se continua marcando el tubular a la medida correcta y cortando los tres tubulares inferiores externos a la mista medida de 2.53 m.

Después se prensan con un marro ambos extremos del tubular, a una distancia de 5 cm, **Figura 3.35**, esto se hace para los tres tubulares externos.

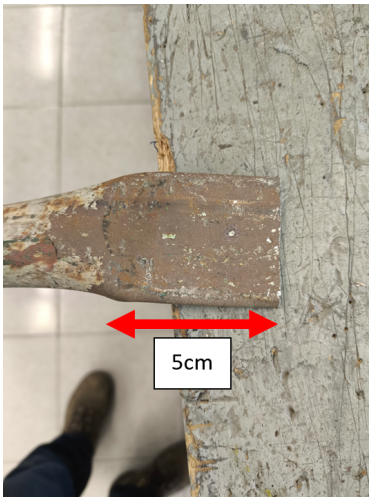


Figura 3.35: Tubular después del prensado.

Ya que se realizó el paso anterior se procede a realizar el doblez a 150 grados, **Figura 3.36**.

Figura 3.36: Tubular con dobles a 150 grados.

Después de que se hizo el doblez se presenta el tubular en la estructura para marcar con un lápiz el lugar de los barrenos donde se unirá con la solera y el tubular opuesto, **Figura 3.37**.

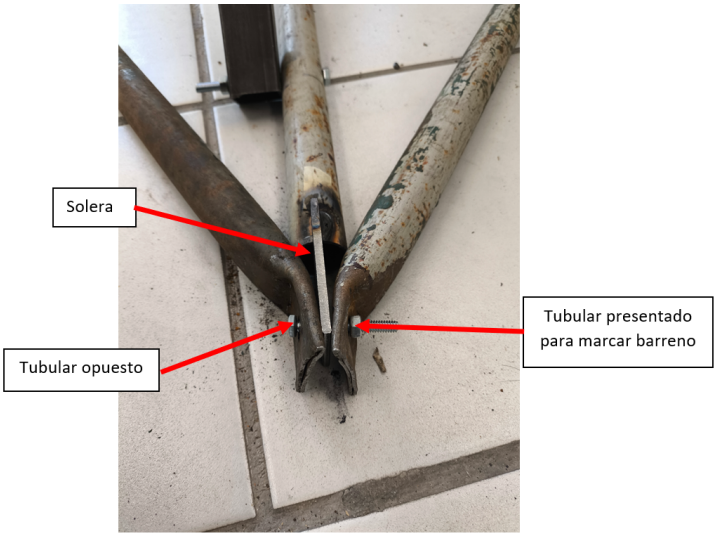


Figura 3.37: Unión de los tres tubulares.

Una vez determinado el punto donde se requiere el barreno, se miden el ancho y alto de dicha ubicación (**Figura 3.38**) y se replican estas medidas en la cara opuesta del tubular externo para marcar con un punzón (**Figura 3.39**). Debido al ángulo de 150 grados y a la longitud del tubo, no es posible realizar el barreno en esta cara, ya que no se podría asentar correctamente en la prensa del taladro de banco.

Figura 3.38: Ancho de cara de donde se saca la medida para la cara contraria.

Figura 3.39: Alto de cara de donde se saca la medida para la cara contraria.

Se procede con el barreno de 11/32 de pulgada en la ubicación marcada con el punzón, situada en los extremos doblados del tubular inferior externo.

Se presenta en la estructura y se coloca el tornillo que sujeta a los tres tubulares, los cuales son exterior izquierdo, exterior derecho e inferior interior, **Figura 3.40** y así se puede comprobar que estén a 2.525 m, **Figura 3.41**. Esto se hace para los tres tubulares inferiores externos.

Figura 3.40: Resultado de la unión.

Figura 3.41: Comprobación de la distancia de los tubulares externos.

Después de realizar estos pasos, la estructura queda como se muestra en la **Figura 3.42**.

Figura 3.42: Estructura ensamblada.

Placa para la base de la torre

La placa base cuenta con las medidas que se muestran en el **Anexo A.7**. Esta placa, al adquirirse de esas medidas, solo se realizaron los barrenos para que se acoplara con la torre, por lo que se procedió a marcar el centro de la placa marcando dos diagonales; posteriormente, se marcó el cuadro donde se coloca la torre.

Ya que se marca el cuadrado de la base que va en la parte inferior de la torre, se centra y sujeta con una prensa en Ç"para que no se mueva, de preferencia usar dos prensas para mayor sujeción, **Figura 3.43**. Con un lápiz marcamos el lugar donde se realizará el barreno.

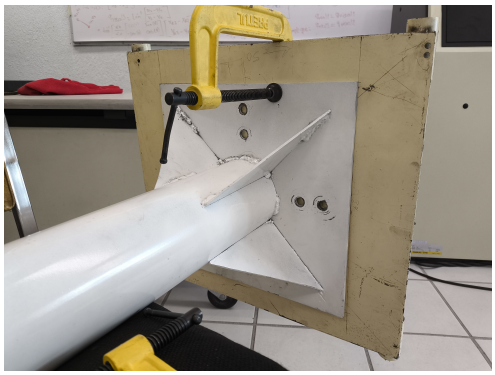


Figura 3.43: Unión de placas con prensa en Ç".

Además de marcar con el lápiz, también se marca con el taladro de mano y una broca de 1/2 pulgadas, esto sirve de guía para la broca en el taladro de banco y así no se mueva el lugar donde se necesita el barreno.

En el taladro de banco se usó la broca de 1/2 pulgada y con la ayuda del nivel se mantuvo centrada la placa para que los barrenos quedaran rectos, **Figura 3.44** y no inclinados, esto para que los tornillos no entren forzados.

Figura 3.44: Barrenado de la placa.

Como ya quedó centrada la columna central, ademas de que todos los tubulares tienen la medida exacta y la placa esta barrenada, se refuerzan los puntos de soldadura que se hicieron en las soleras, esto en las 6 que van en la columna, las 3 que van en los tubulares inferiores internos, la unión que se hizo en la columna central y por último colocar la placa sobre la columna central, **Figura 3.45**, antes de desarmar se colocan muescas con arco de segueta en cada pieza para no perder el orden.

Figura 3.45: Reforzando soldadura en toda la estructura.

Etapa de pintado

Se desarman todos los elementos de la estructura y se procede a pintar por piezas.

Se usó un litro de pintura y dos litros de thinner.

Para pintar se usó un compresor de aire y una pistola de gravedad de abanico chico o tobera de 1.5 mm.

Para todas las piezas se le aplicaron 2 capas de pintura, dejando secar entre cada aplicación por lo menos 30 minutos en lo que se secaban, esto para una mejor textura al aplicar la segunda capa de pintura.

Primero se pintaron los 3 tubulares inferiores externos.

Después se continuó con la columna central y tubulares inferiores internos **Figura 3.46**.

Figura 3.46: Tubulares inferiores internos pintados.

Finalizado lo anterior se colocaron los tubos en una posición en donde se pudieran secar sin maltratarse la textura de la pintura (por rayones, polvo, basura, huellas de dedos) y esperamos 24 horas para el secado de la pintura.

Ensamble después de pintar

Al pasar 24 horas ya se puede manipular las piezas pintadas pero no estaba de más tener cuidado de no maltratar la pintura, esta estructura se puede armar con una sola persona ya que las piezas no son pesadas.

Cada pieza está identificada con una muesca correspondiente (I, II, III), realizada con arco de segueta.

En la columna el número uno siempre será el que tiene la solera sobre la arista y no sobre las caras de la columna, ya que se ubicó la solera uno, se coloca enfrente de la columna y en sentido antihorario se ubica el 2, también se puede hacer de la siguiente forma ya que se coloca enfrente de la solera uno, la solera del lado derecho será el 2 y la de la izquierda el 3.

El orden en que se ensamblan los tubulares es, solera, tubular y tornillo, **Figura 3.47**.

Para apretar el tornillo se usan dos llaves combinadas de 1/2 pulgada o 13 mm.

Figura 3.47: Orden para ensamble.

Primero se para la columna central, como está bien centrado todo, no se cae, por lo que permite colocar los tubulares inferiores internos, **Figura 3.48**.

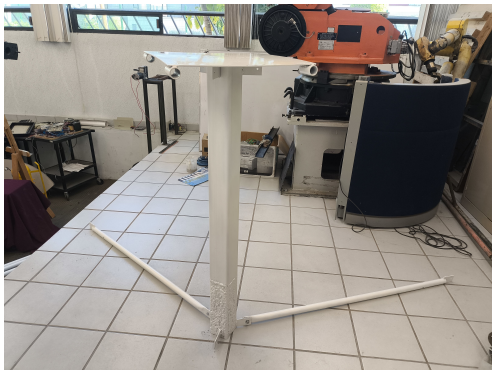


Figura 3.48: Ensamble de tubulares inferiores internos.

Cada tubular lateral tiene su número con plumón el cual quedó debajo de la pintura blanca pero se alcanza a ver por lo que no hay problema de saber qué posición va. Para estos tubulares el orden en la parte superior es el siguiente, primero es tubular cuadrado, solera y tornillo, **Figura 3.49**.

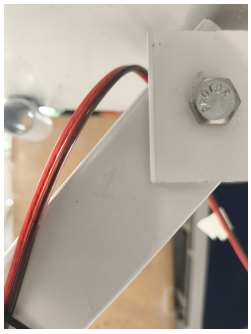


Figura 3.49: Orden de ensamble tubular lateral.

Ya que se sabe el orden, se colocan los tornillos 5/16 de pulgada por 4 pulgadas de largo, **Figura 3.50** y **Figura 3.51**.

Figura 3.50: Ensamble de tubulares laterales con inferiores internos.

Figura 3.51: Orden de ensamble.

Por último se colocan los tubulares inferiores externos (2.525 m) y la estructura trípode queda ensamblada, **Figura 3.52**.

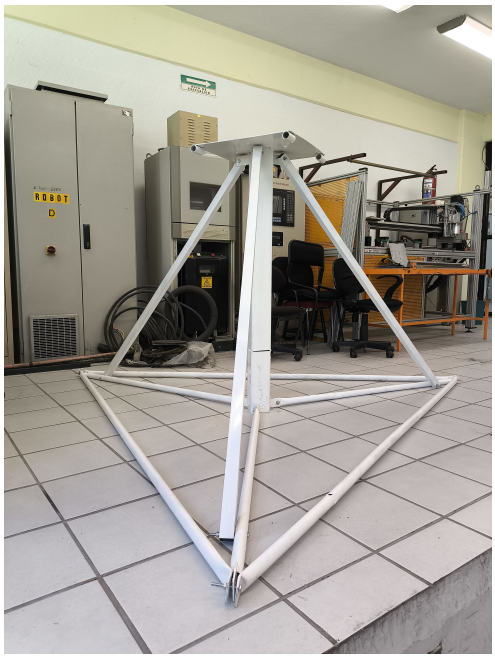


Figura 3.52: Estructura trípode ensamblada.

3.6. Construcción de Torre

Para el diseño de la torre se usó tubular de 4 pulgadas de diámetro, cédula 30 y altura de 3 metros, como se ve en el plano del **Anexo A.8**.

Esto de acuerdo a la escala mostrada en la Tabla 3.4.

TORRE		
Relación	**01:30**	**01:01**
Construido	3 m	90 m
Equivalencia	1 m	30 m

Tabla 3.4: Relación de escalado para la torre.

Para la parte que acopla con la base tipo trípode, se realizó el siguiente diseño en la parte inferior, **Figura 3.53**.

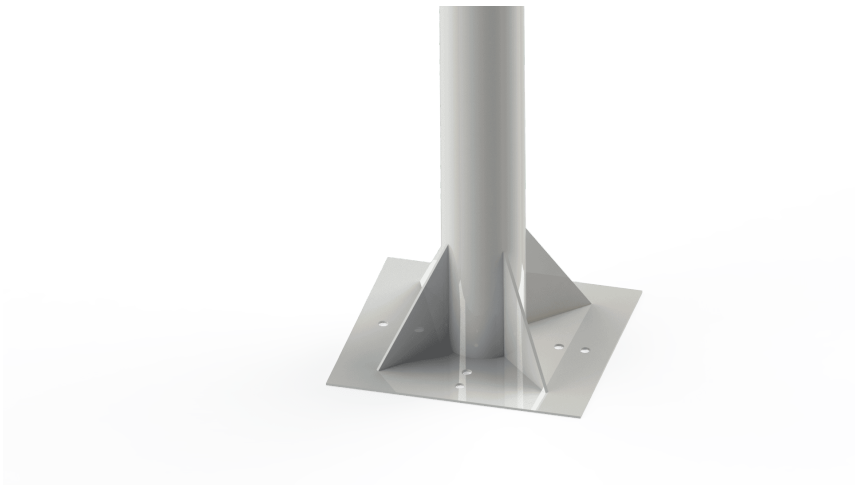


Figura 3.53: Parte inferior de la torre.

Primero se realizaron los barrenos con broca de 1/2 pulgada en la placa que va en la parte inferior de la torre; las medidas de esta placa son 34 cm por 34 cm, **Figura 3.54**.

Figura 3.54: Barrenos en la placa.

Después se procede a soldar la placa inferior, la que permitirá la unión con la base trípode, también las 4 escuadras en triángulo, las cuales refuerzan la unión del tubular con la placa, **Figura 3.55**.

Figura 3.55: Placa unida al tubular con soldadura.

3.7. Construcción de Góndola

Se utilizaron los siguientes elementos :

- Ángulo de 1.5 pulgadas y 29.5 cm de largo, para extender el largo de la placa base de la góndola.
- Tubular de 1 pulgada de diámetro y 96 cm de largo, para sostener el disco de aluminio.
- Cojinete de 1 pulgada de diámetro interior, para soporte del eje que viene de las aspas.
- Tubular de 1 pulgada de diámetro y 44 cm de largo, con dirección a las aspas.
- Tubular de 1 pulgada de diámetro y 37.5 cm de largo, con dirección al generador.
- Balero de 1 pulgada de diámetro interior, para el eje que viene de las aspas y el que va al generador.
- Disco de aluminio de 34.5 cm de diámetro y 8 mm de espesor, para colocar las aspas.
- 3 arandelas para sujetar las aspas.

Esto se visualiza en la **Figura 3.56**.

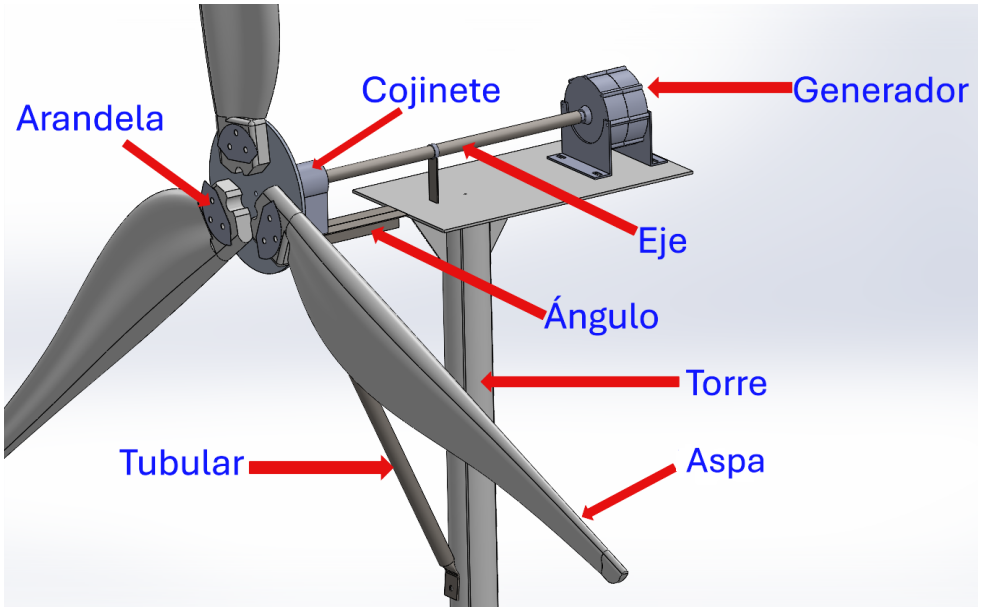


Figura 3.56: Componentes en la góndola.

Para la base de la góndola se utilizó una placa de 6 mm de espesor, 53 cm de largo por 34 cm de ancho; primero marcamos con plumón los barrenos que se necesitan para fijar el motor generador, **Figura 3.57** y **Figura 3.58**.

Figura 3.57: Posición del motor para marcar el lugar de los barrenos.

Figura 3.58: Puntos marcados en la placa.

Con un punzón y el martillo se marcó el lugar de los barrenos.
Con una broca 5/16 de pulgada se realizan los 4 barrenos, **Figura 3.59** y **Figura 3.60**.

Figura 3.59: Barrenado en la placa.

Figura 3.60: Barrenos terminados.

Luego se procedió a soldar la placa con la torre en la parte superior y se refuerza con dos escuadras en triángulo, **Figura 3.61**.

Figura 3.61: Placa unida a la torre.

Etapa de pintura

Se utilizaron los mismos materiales y técnicas que se emplearon para pintar la base tipo trípode.

En la preparación de la pintura, por cada 250 mililitros (ml) de pintura, se agregan 500 ml de thinner, se llena el contenedor de la pistola y se comienza a pintar.

Se comienza con la primera capa de pintura y hasta que la torre quede pintada de manera uniforme, **Figura 3.62**.

Figura 3.62: Torre pintada de manera uniforme.

Resultado de la construcción del aerogenerador

Terminado el proceso de pintado, el resultado fue el que se muestra en la **Figura 3.63** y con esto se continúa a colocar los sensores para realizar pruebas.

Figura 3.63: Aerogenerador terminado.

Capítulo 4

Caracterización de sensores y módulos de comunicación para transmisión de datos de forma inalámbrica

4.1. Selección de los módulos para obtener los datos de manera inalámbrica

Para obtener los datos de forma inalámbrica se usó el protocolo ESP NOW, el cual permite una transmisión de 250 bytes, es de consumo bajo y en condiciones de zona abiertas puede tener un rango de hasta 200 metros de alcance, como lo expresa Martínez Fernández (2021).

Las tarjetas de desarrollo que cuentan con este protocolo es la ESP32 S2 MINI, **Figura 4.1** , la cual tiene suficientes pines para conectar el MPU9250, entonces fue usado en la versión 1 y contando con las siguientes características.

- Chip integrado: ESP32-S2FN4R2
- Memoria:
- Flash: 4 MB
    - Flash: 4 MB
    - PSRAM: 2 MB
- Número de entradas/salidas: 27
- Voltaje de entrada: 5 V USB tipo C
- Voltaje de operación: 3.3 V
- Interfaces compatibles:

- ADC
- DAC
- Entradas/Salidas de detección táctil
- SPI
- I2S
- I2C
- UART
- RMT(TX/RX)
- Controlador,LED PWM
- USB OG

- • Sensor de temperatura interno
- • Velocidad de reloj: 240MHz
- • Firmware por defecto: MicroPython
- • Dimensiones: 34.3 mm x 25.4 mm
- • Peso: 5.31 g

En Arduino IDE se encuentra como LOLIN S2 MINI.

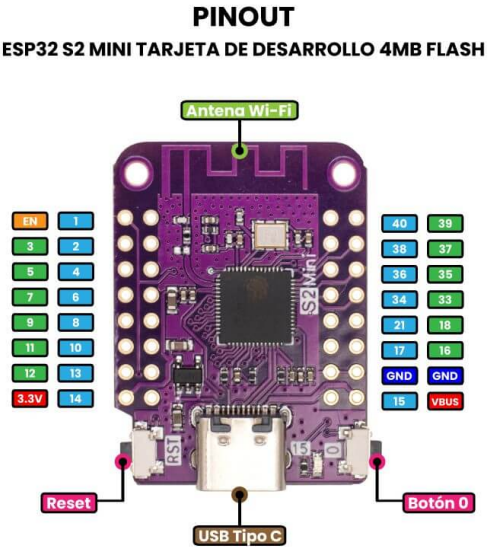


Figura 4.1: Tarjeta ESP32 S2 Mini. Recuperado de UElectronics (s.f.).

Para la versión 2 se utilizó la tarjeta ESP32 C3 super mini, **Figura 4.2** ya que es una tarjeta más compacta que cuenta con las siguientes características:

- CPU: ESP32-C3 32-bit RISC-V 160 MHz
- Puerto de comunicación: Tipo C
- Voltaje de operación: 3 V a 5 V
- Temperatura de operación -40° a 85° C
- WIFI: IEEE 802.11 b/g/n 2.4 GHz
- Bluetooth: Bluetooth 5, Bluetooth mesh
- Memoria:
  - 384 KB ROM
  - 400 KB SRAM
  - 4 MB flash
  - GPIO:
  - 1xI2C
  - 1xSPI
  - 2xUART
  - 11xPWM
  - 4xADC
- Dimensiones: 22.5 mm x 18 mm
- Peso: 3.05g

En el programa Arduino IDE, al seleccionar la placa para subir el código, se encuentra como ESP32 C3 Dev module.

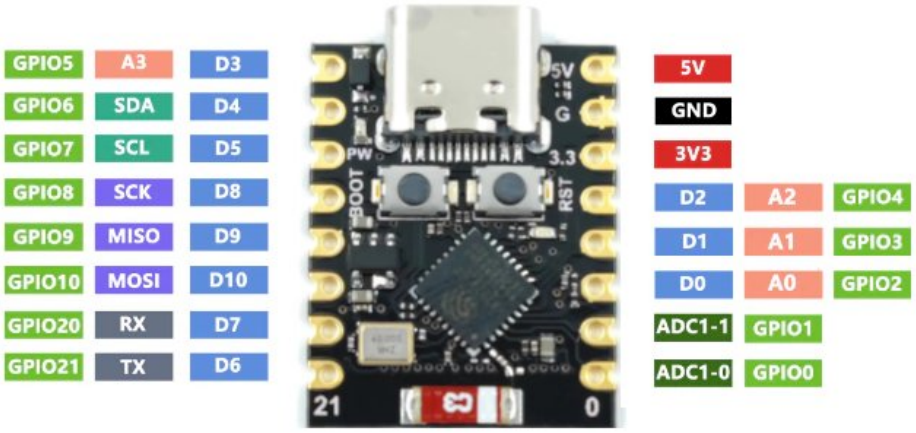


Figura 4.2: Tarjeta ESP32 C3 super mini. Recuperado de Megatrónica (s.f.).

La tabla 4.1 muestra la conexión de pines entre la tarjeta ESP32 S2 Mini y el sensor MPU9250.

ESP32 S2 Mini Pines	MPU9250 Pines
3V3	VCC
GND	GND
13	SCL
14	SDA

Tabla 4.1: Conexión de pines.

La **Figura 4.3** muestra la conexión física del sensor MPU9250 con el microcontrolador ESP32 S2 Mini, esto se realiza para los seis sensores, usando placas fenólicas como se ve en la **Figura 4.4**.

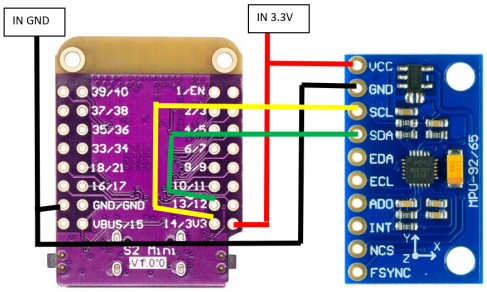


Figura 4.3: Diagrama de conexión física.

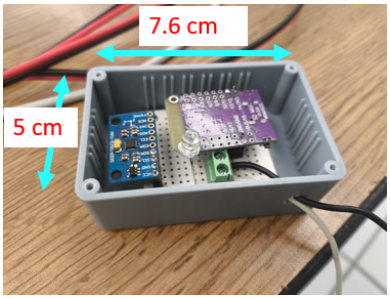


Figura 4.4: Deposito con sensor y microcontrolador.

Para la versión dos, se usaron tarjetas ESP32 C3 Super mini esto debido a un problema de alimentación que provocó que se quemaran las tarjetas S2 mini, esto se explica más adelante en la sección de alimentación de los sensores. Pero al cambiar de modelo la conexión fue similar ya que solo se cambiaron los pines de SDA y SCL como se muestra en la tabla 4.2, ademas una ventaja de estas tarjetas es que se pueden alimentar a 5 V lo que hace que ya no sea necesario usar un regulador externo en la fuente de alimentación.

ESP32 C3 Super Mini Pines	MPU9250 Pines
3V3	VCC
GND	GND
D5	SCL
D4	SDA

Tabla 4.2: Conexión de pines.

La **Figura 4.5** muestra la conexión física del sensor MPU9250 con el microcontrolador ESP32 S2 Mini, esto se realiza para los seis sensores, usando placas fenólicas como se ve en la **Figura 4.6**.

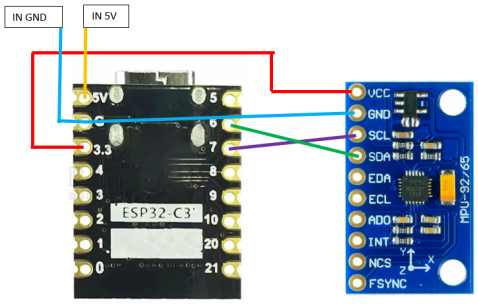


Figura 4.5: Diagrama de conexión física.

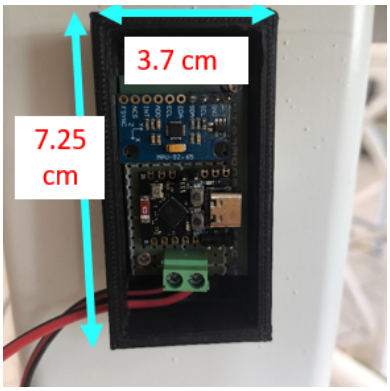


Figura 4.6: Deposito con sensor y microcontrolador.

4.2. Caracterización de sensores

Offset

Para calibrar estos sensores es necesario saber que el offset se refiere al valor de salida que tiene el sensor cuando se encuentra en reposo y en una orientación determinada. En teoría, si el acelerómetro estuviera perfectamente calibrado, en reposo debería medir exactamente 0g en los ejes (X, Y) y 1g en el eje (Z) debido a la gravedad de la Tierra. Sin embargo, en la práctica, hay pequeños errores en la fabricación y condiciones ambientales que provocan que el sensor entregue valores diferentes a los esperados como se describe en Mechatronics (s.f.).

Por lo tanto se debe compensar el offset del acelerómetro realizando una calibración manual, esto se hace colocando el acelerómetro en una superficie plana, se mide los valores de salida de cada eje, después se restan esos valores de las lecturas futuras para compensar el error, cabe mencionar que esto se hace antes de colocarlos en el aerogenerador.

Tomando en cuenta lo anterior se usó el código del **Anexo A.10.1** , para ajustarlo por software esto para obtener lecturas precisas. Al cargarlo en Arduino IDE arrojó los valores que corresponden al acelerómetro conectado, esto se hace para cada sensor ya que cada sensor tiene un offset diferente, por lo que si se usa el mismo, se tendrán lecturas erróneas.

Para el sensor 1 se obtuvo los valores:

```
1 int16_t ax_o = 5856, ay_o = 6231, az_o = 8946;
```

Por lo que estos valores se sustituyeron en el código de transmisión del sensor 1; este mismo procedimiento se hizo con los otros 5 sensores para obtener los offset correctos y poder calibrarlos, no importa si se usa la S2 Mini o la C3 Super mini, solo cuidar que se indiquen los pines correctos en el código.

Dirección MAC

Para poder enviar los datos de forma inalámbrica al receptor que se conecta a la computadora se necesita usar una dirección MAC (Media Access Control), la cual es un

identificador único del dispositivo o interfaz de red de una computadora. Se representa como una serie de 12 dígitos hexadecimales agrupados en pares, tal como se describe en Centro de Ciencias de la Atmósfera (s.f.).

Para el módulo receptor se utiliza una tarjeta ESP32 S2 Mini, esta permite recibir la lectura de los seis sensores y enviarlas al puerto serial, donde con ayuda de la interfaz se pueden visualizar y almacenar en una base de datos en Excel. Pero primero se debe obtener su dirección MAC con el siguiente código **Anexo A.11.1**.

Después de cargar el código en el receptor con Arduino IDE muestra la siguiente dirección:

```
uint8_t broadcastAddress[] = { 0x80, 0x65, 0x99, 0x4B, 0x13, 0xBC };
```

Una vez obtenida la dirección MAC, se coloca en el código de cada uno de los transmisores que tengan conectado el sensor MPU9250, ya que esta es la dirección del receptor el cual recibe los datos.

Caracterización de los sensores para realizar lecturas y transmitir datos al receptor

Ya que se tienen los valores (offset y MAC) se colocan en el código del **Anexo A.12.1** como se muestra en la **Figura 4.7**, el cual permite caracterizar el acelerómetro, para así obtener las lecturas y mandarlas al módulo receptor, por lo tanto esto se hace con las tarjetas que serán transmisoras o que tengan conectado un acelerómetro, solo se les va cambiando el valor del offset para cada acelerómetro, aclarar que la dirección MAC no se cambia ya que es la del receptor a donde van a enviar las lecturas. Se verifica que los pines coincidan tanto en el código como en la tarjeta S2 Mini o C3 super mini según sea el caso.

El procedimiento anterior se hace de la misma forma para los otros 5 sensores; en el apartado de Anexos se puede encontrar el código que le corresponde a cada sensor del 1 al 6. Solo es cuestión de cambiar el valor del offset de cada acelerómetro, el id (número que le corresponde a cada sensor) y verificar que la dirección MAC del receptor sea la correcta.

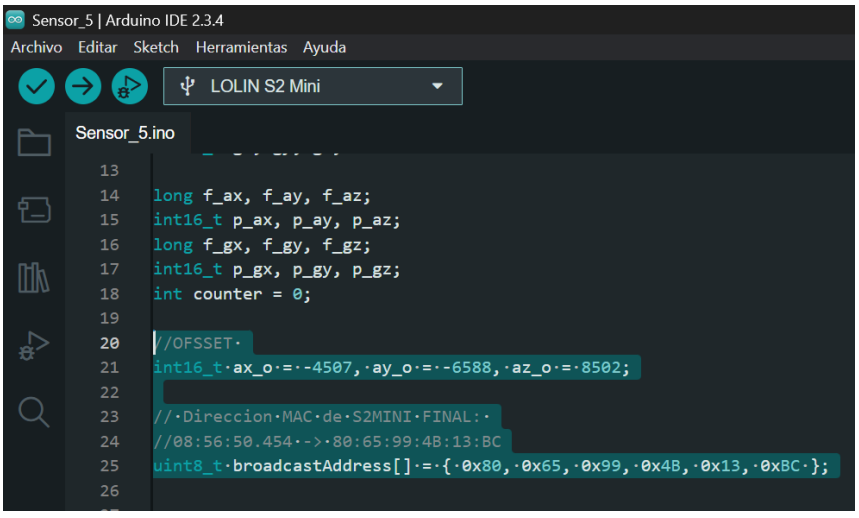


Figura 4.7: Offset y dirección MAC en el código para los transmisores.

Modulo receptor

Una vez caracterizado el microcontrolador con el sensor, se emplea como transmisor. A continuación, se procede a configurar el módulo receptor, que previamente solo se utilizó para obtener su dirección MAC. Ahora se le carga el código correspondiente para que funcione como receptor (**Anexo A.13.1**).

4.3. Colocación de sensores en el aerogenerador construido

Los sensores del uno al tres van en cada refuerzo lateral a una altura de 5 cm tomando el piso como referencia, el número cuatro en la columna central, el sensor 5 se ubica en la torre del aerogenerador a una altura de 1.70 cm a partir de donde inicia la torre, esto indicara las vibraciones que presente por turbulencias, ya sean por el viento, o daños en la estructura y por último el sensor 6 en el cojinete, en la parte de la góndola donde va el eje de las aspas, ya que como se mencionó al inicio son los elementos que sufren mayor desgaste, esto se visualiza mejor en la **Figura 4.8**.

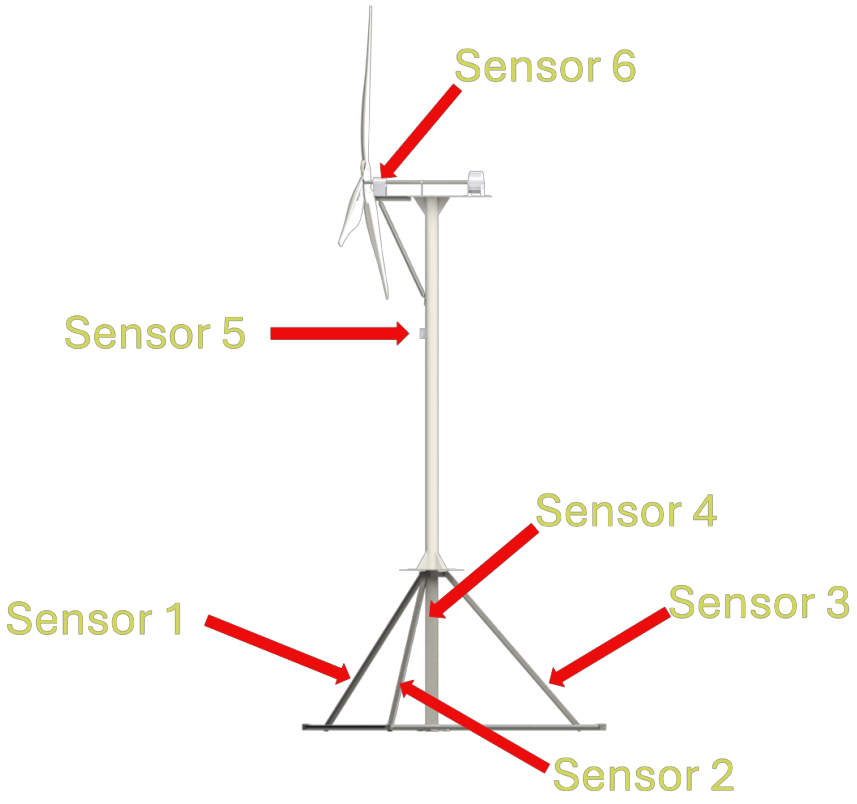


Figura 4.8: Ubicación de los sensores en el aerogenerador.

4.4. Alimentación de sensores y módulos de comunicación

Versión 1

Se utilizó una batería de 12 V y 80 000 mAh para alimentar los seis sensores MPU9250 a 3.3 V, junto con sus respectivos microcontroladores ESP32-S2 Mini también a 3.3 V, además de un actuador lineal de 12 V con su microcontrolador ESP32-S2 Mini a 3.3 V.

Por lo que se usó el siguiente circuito simulado en LTspice, usando el regulador LM1117T para reducir el voltaje de 12 a 3.3V, **Figura 4.9**, el cual de acuerdo al datasheet indica que da una salida de 3.3V, con una temperatura de operación de 45°C a 125°C por lo que fue necesario usar un disipador de calor, aplicándole pasta térmica, para una mayor transferencia de calor al disipador.

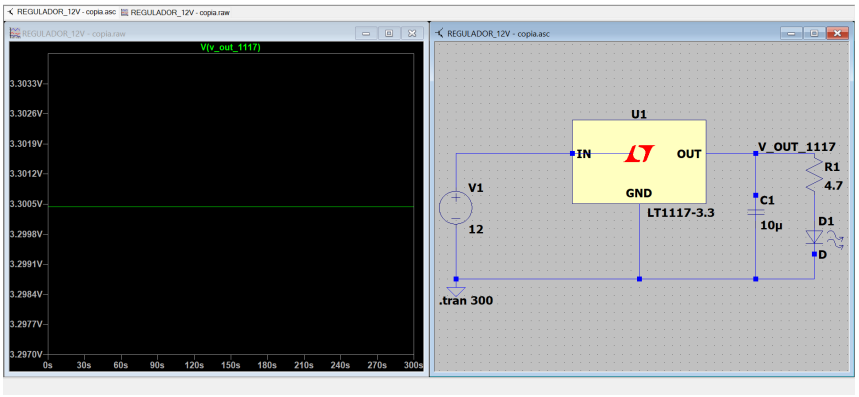


Figura 4.9: Circuito del regulador en LTspice.

En la práctica se usaron los materiales siguientes:

- Placa fenólica de 45mm x 45 mm
- 2 Clemas
- Regulador a 3.3V modelo LM1117T
- Switch de botón
- Led ultrabrillante de 3 volts sin resistencia
- Fusible de 1 Ampere
- Capacitor de 10 uF

Por lo tanto, se ensambló el circuito, como se observa en la **Figura 4.10**, montado en la estructura tipo trípode en una de las caras de la columna central.

Figura 4.10: Fuente de alimentación versión 1.

Durante las primeras pruebas, se alimentaron los sensores con la fuente de 12 V regulada a 3.3 V durante 15 minutos, registrando un consumo de 227.6 mA. En este periodo, el regulador no presentó fallas relacionadas con cortocircuitos, sobretensión en la salida ni sobrecalentamiento.

Después el circuito con el regulador 3.3 V se probó solo con el actuador lineal como freno (solo para alimentar la tarjeta de control), presentando un consumo de 500 mA, por lo que la alimentación era suficiente.

En las primeras pruebas, los sensores y el actuador no se evaluaron de manera conjunta, por lo que el sistema funcionó correctamente. Sin embargo, al realizar la prueba con los seis sensores y el actuador lineal alimentados desde la fuente de 12 V mediante el regulador LM1117T, el conjunto solo operó durante cinco minutos antes de que se dañaran todos los sensores y sus respectivos microcontroladores.

Esto se debió a un sobrecalentamiento en el regulador LM1117T, provocando que la salida dejara de regular, haciendo que el voltaje de entrada de 12 V fuera igual que el de salida (12 V) , en lugar de 3.3 V. Por lo que la potencia del actuador hace que el regulador se vuelva inestable, provocando sobrecalentamiento hasta fallar, ya que el datasheet del regulador nos indica un rango de temperatura de operación de los 25°C hasta 125°C y en este caso, con el uso del actuador, llegó a superar ese rango, comenzando desde los 110°C hasta fallar.

Versión 2

Por lo ocurrido, se realizó la versión 2, en donde se decidió alimentar de manera independiente los sensores y seguir con la batería de 12 V solo para el actuador lineal con su microcontrolador; se cambiaron los sensores por nuevos, se cambió la S2 mini por la C3 super mini ya que este modelo permite la alimentación a 5 V por lo que ya no era necesario usar un regulador externo en la entrada, además de que es más eficiente usar la power bank que entrega 5 volts continuos con capacidad de 20,000 mAh, **Figura 4.11** al no desperdiciar la energía en calor como lo hacía el LM1117T, por lo tanto no hubo desequilibrio en la fuente, permitiendo un correcto funcionamiento.

Figura 4.11: Fuente de alimentación para versión 2.

4.5. Motor reductor para el accionamiento de las aspas

Para accionar las aspas se utilizó un motor reductor, debido a que el viento disponible a la altura en la que se encuentra el aerogenerador no es suficiente para ponerlas en movimiento. Además, al no contar con un túnel de viento como en el trabajo de Rangel (2022), se optó por esta alternativa, la cual permite mover las aspas a diferentes rpm, simulando el efecto del viento sobre ellas, el cual fue girar las aspas de 10 a 22 rpm.

El acoplamiento del motor reductor al eje de las aspas se realizó mediante un mecanismo similar al de una bicicleta. Para ello, se colocó una estrella de 4 pulgadas en el eje de las aspas y otra en el eje del motor, dado que ambos ejes se encuentran en paralelo, se instaló una cadena que transmite el movimiento de manera conjunta, como se aprecia en la **Figura 4.12** y **Figura 4.13**.

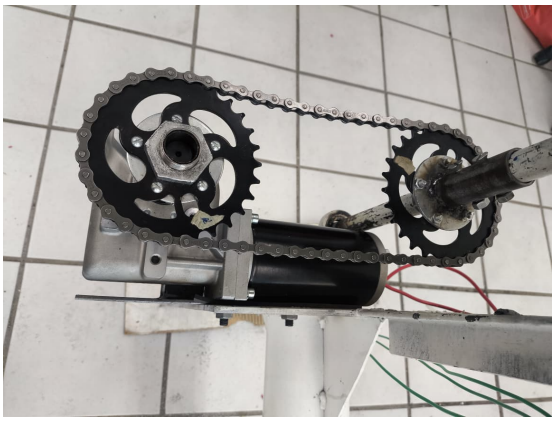


Figura 4.12: Motor reductor ensamblado al mecanismo.

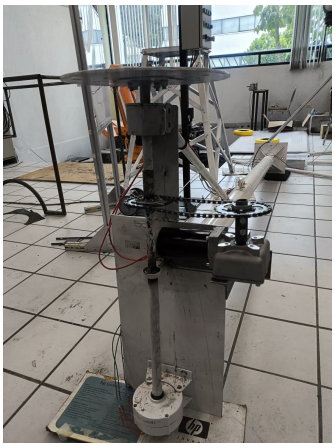


Figura 4.13: Mecanismo para aspas.

4.5.1. Caracterización del motor reductor

Se utilizó una ESP32-S2 Mini como transmisor utilizando el código que se muestra en el **Anexo A.15.1**, este transmisor se conecta directamente a la laptop, permitiendo que la interfaz gráfica la detecte y envíe las instrucciones correspondientes.

Para accionar el sistema se utilizó una ESP32 receptora con el código del **Anexo A.16.1**, empleando el protocolo ESP-NOW para recibir de forma inalámbrica los comandos para el control del motor.

En la **Figura 4.14** se muestra el módulo receptor, ubicado dentro de su respectivo compartimiento. Este incluye una fuente de alimentación de 12 V - 40 A dedicada exclusivamente al motor reductor y un puente H, debido a que el sistema presenta consumos de corriente de hasta 10 amperes, por lo que para evitar sobrecargar dicha fuente, se empleó una segunda fuente de 12 V - 20 A independiente destinada a la ESP32 receptora y a un ventilador encargado de refrigerar el puente H.

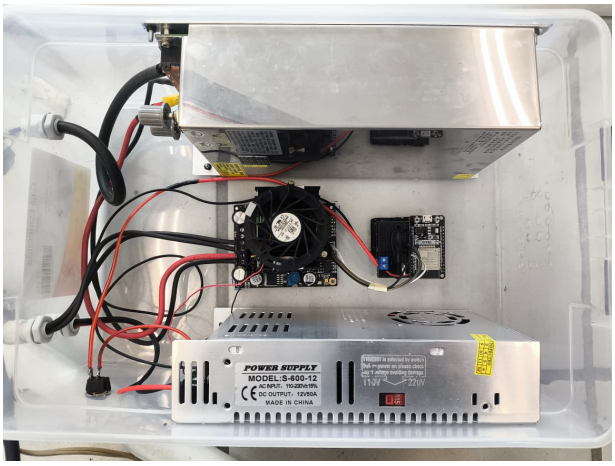


Figura 4.14: Compartimiento del modulo receptor.

Capítulo 5

Interfaz gráfica de usuario (GUI)

Esta parte complementa lo que se ha realizado en los capítulos anteriores, ya que la interfaz gráfica es el intérprete que permite la comunicación amigable entre la máquina y el usuario, así como lo hace el sistema operativo de cualquier computadora y las diversas aplicaciones que se encuentran en ella.

Por ello, en este enfoque, la interfaz nos permite obtener la base de datos en Excel de una forma rápida y sencilla, poder ejecutarla en cualquier computadora y además de que cualquier persona puede usarla sin necesidad de conocer el código fuente (lenguaje de programación y lineas de comandos en la que se desarrolló la interfaz de usuario).

5.1. Desarrollo

Como la comunicación es inalámbrica, la interfaz permite de una forma amigable, visualizar y guardar las lecturas que los sensores MPU9250 captan, esto se logró utilizando un receptor conectado a un puerto USB de la computadora, para establecer comunicación con los transmisores conectados a los sensores.

Al realizarse en software libre como lo es Python tenemos acceso al código fuente para realizar futuras actualizaciones sin depender de una licencia, además esta interfaz se pueda ejecutar en cualquier computadora que el usuario tenga disponible, ya que es un archivo ejecutable (.exe) para Windows, **Figura 5.1**, con un tamaño de 40.9 MB.

Figura 5.1: Icono del programa ejecutable.

Diagrama de flujo

Se realizó el siguiente diagrama de flujo antes de comenzar la programación en Python, el cual nos muestra los pasos que sigue la interfaz para ver el estado de los sensores, así como las lecturas y cronometrar el tiempo en el que se esté generando la base de datos, **Figura 5.2**.

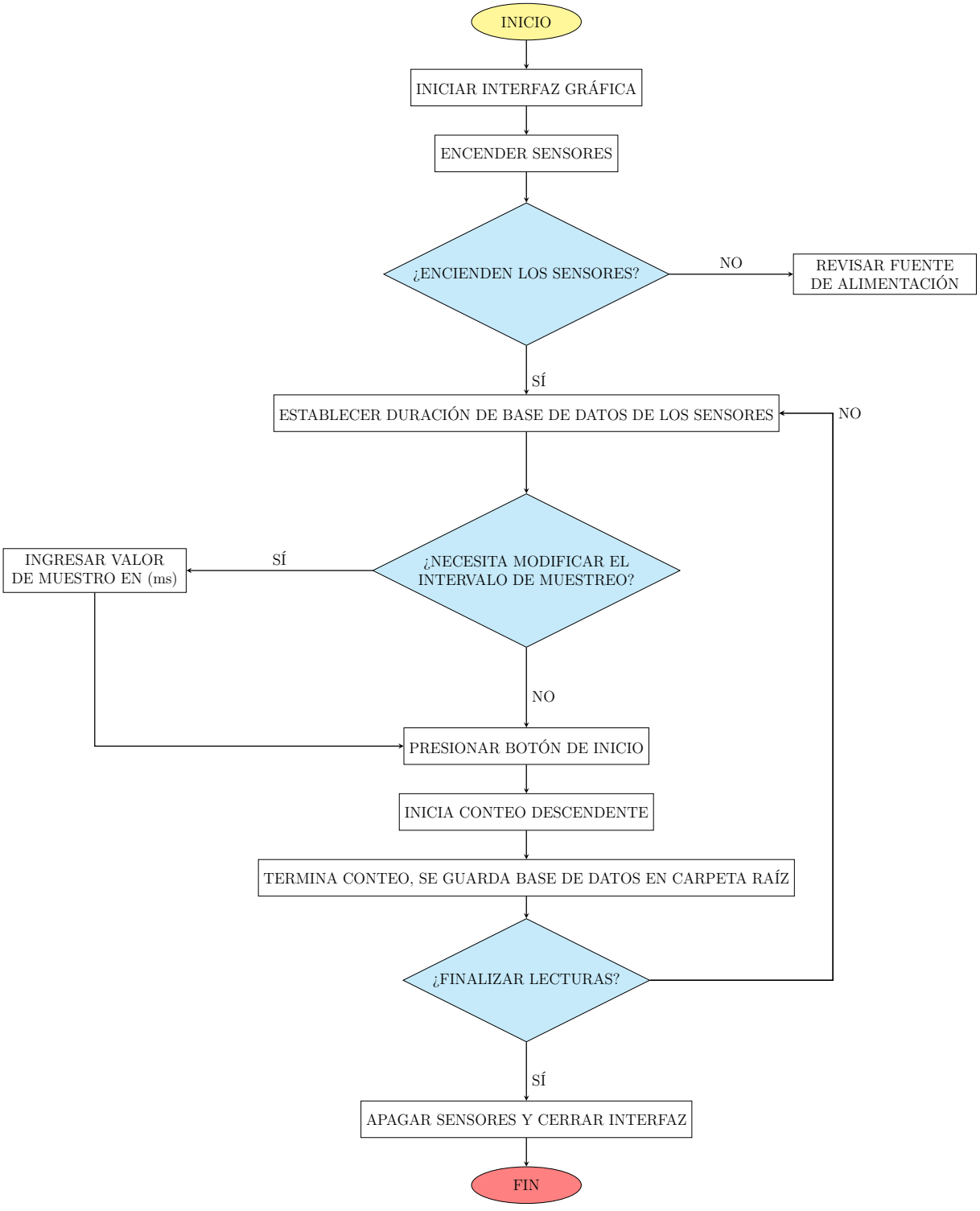


Figura 5.2: Diagrama de flujo para interfaz de usuario.

Una vez definido el funcionamiento requerido de la interfaz, se utilizó Visual Studio como entorno de desarrollo integrado. Este, en conjunto con el intérprete de CPython, permite ejecutar las líneas de código escritas en Visual Studio.

Se utilizó la librería CustomTkinter para el desarrollo de la interfaz gráfica, ya que permite implementar diseños más estéticos y modernos en comparación con la librería estándar Tkinter.

Para la versión 1 se usó la librería Tkinter, **Figura 5.3**, como se observa tiene un estilo clásico, por ejemplo tiene esquinas cuadradas en los botones, en las etiquetas y los tonos de los colores son opacos.

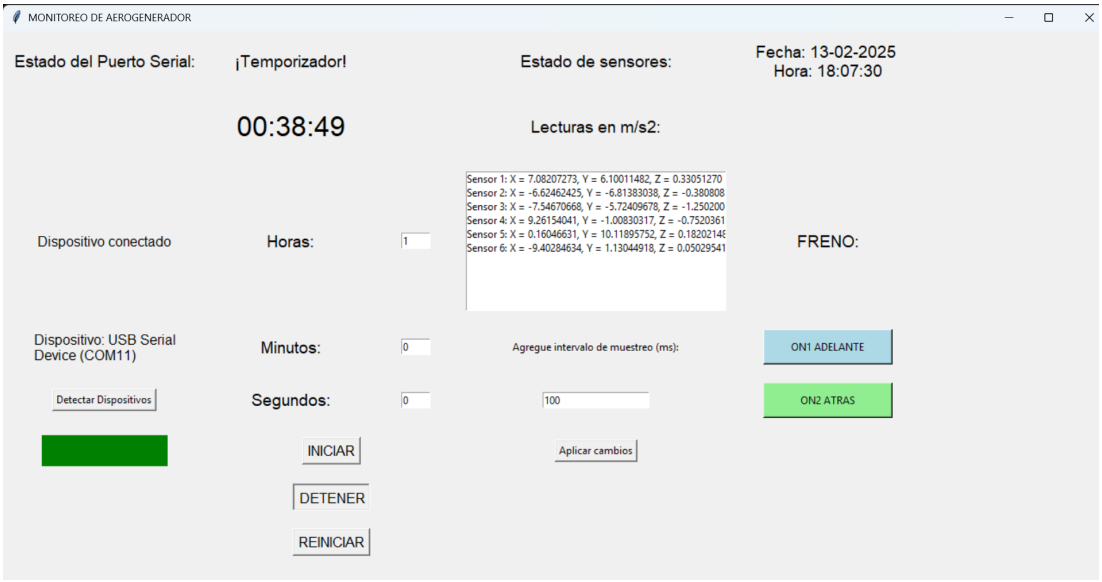


Figura 5.3: Versión 1 de interfaz gráfica.

5.2. Secciones de la interfaz gráfica de usuario

En la versión 2 con el código que se muestra en el **Anexo A.14.1** usando la librería Customtinker hay una actualización, donde se usan esquinas redondeadas, colores brillantes y fondos llamativos, la cual consta de las siguientes secciones comenzando de izquierda a derecha como se ve en la **Figura 5.4**, por lo que este es el orden para empezar a usarla.

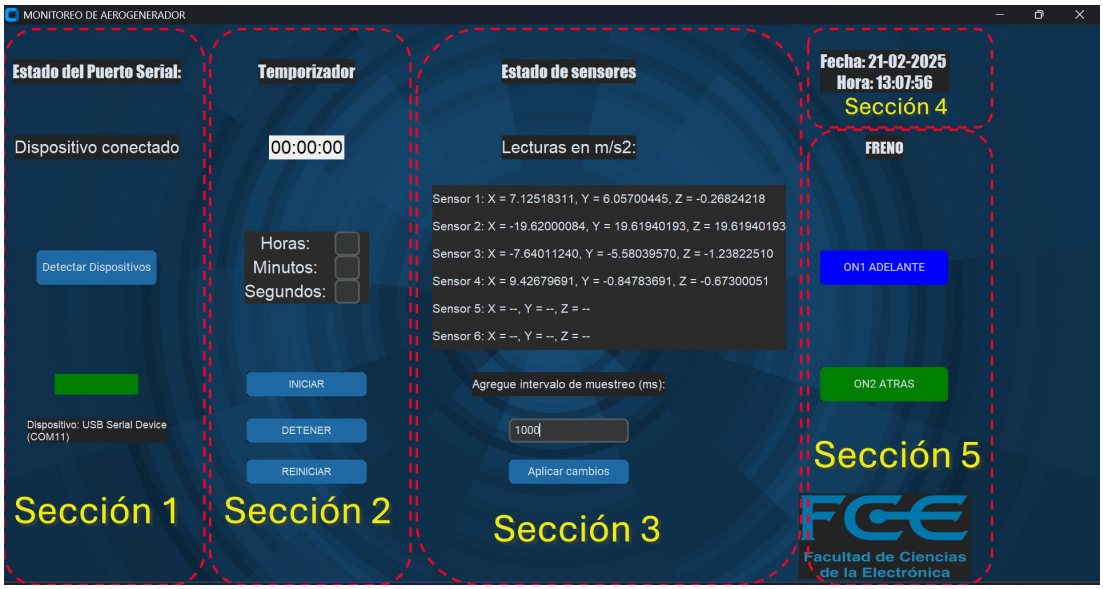


Figura 5.4: Versión 2, secciones de la interfaz gráfica.

- **Sección 1: Estado del puerto serial**

Aquí encontramos el estado del puerto serial, esta parte consta de un botón "Detectar dispositivos", el cual al presionarlo busca establecer comunicación del módulo receptor con el puerto serial de la computadora, si este módulo realiza la comunicación de forma exitosa con la interfaz, muestra una etiqueta que dice "Dispositivo conectado" y la etiqueta se mostrara de color verde, de lo contrario permanecerá en rojo con la leyenda "No se encontró dispositivo".

- **Sección 2: Temporizador**

Si en la sección anterior no hay problema se puede continuar con la sección 2, aquí se puede programar por cuanto tiempo se genera la base de datos en un archivo excel, esto se hace ingresando un valor en los recuadros de horas, minutos y segundos. Si solo se requiere unos segundos para obtener la base de datos en los recuadros de horas y minutos se debe ingresar un cero ya que si no se hace, el programa mostrara un mensaje de error, en caso de este mensaje presionar el botón reiniciar para volver a ingresar los valores de forma correcta.

Una vez ingresado el tiempo se presiona el botón iniciar, el cual comenzara a generar un archivo Excel en el cual se guardan las lecturas de los sensores en tiempo real.

Si se requiere interrumpir la base de datos se presiona el botón detener, este suspende las lecturas en el Excel y guarda el documento generado.

El botón reiniciar permite repetir el tiempo ingresado, sin necesidad de volver a llenar los recuadros de horas, minutos y segundos.

- **Sección 3: Estado de sensores**

Aquí se muestran las lecturas de los sensores en tiempo real, cada sensor ocupa un renglón del 1 al 6, cada sensor muestra la lectura en los ejes (X, Y, Z), estas lecturas se muestran en m/s^2, por defecto las muestras se generan cada 500 milisegundos (ms), lo que lleva a la parte de agregar el periodo de muestreo; si se necesita que los datos se guarden y visualicen en un tiempo específico, podemos ingresar desde 10 ms en adelante, para ello una vez ingresado el número para el intervalo, presionar el botón aplicar los cambios.

- **Sección 4: Fecha y hora**

Esta pequeña sección solo permite visualizar la fecha con el día, mes, año, horas, minutos y segundos; todo esto en tiempo real, en un formato amplio, a diferencia de la fecha de la computadora.

- **Sección 5: Freno**

Esta sección permite mover un actuador lineal de forma inalámbrica, esto se hace enviando el comando ON1 al presionar el botón ADELANTE, si se suelta deja de enviar el comando lo que se traduce en que el actuador lineal dejará de recibir el comando y se detendrá, recibiendo el comando OFF. Al presionar el botón ATRÁS, envía otro comando el cual es ON2, moviéndolo en sentido opuesto. Esto se desarrolló para futuras actualizaciones, además comprueba que la interfaz con el receptor conectado también se puede volver transmisor y que esta tarjeta ESP32 S2 mini es capaz de realizar estas funciones a pesar de su tamaño.

5.3. Interfaz gráfica de usuario para el motor reductor

Para controlar la velocidad desde la laptop se utilizó una interfaz de usuario desarrollada en Python el cual se muestra en el **Anexo A.17.1**, el archivo tiene un tamaño de 1.83 MB y puede ejecutarse en cualquier computadora con sistema operativo Windows.

5.3.1. Elementos de la interfaz del motor reductor

La interfaz se organiza en diferentes secciones, como se muestra en la **Figura 5.5**. A continuación, se describen cada una de ellas:

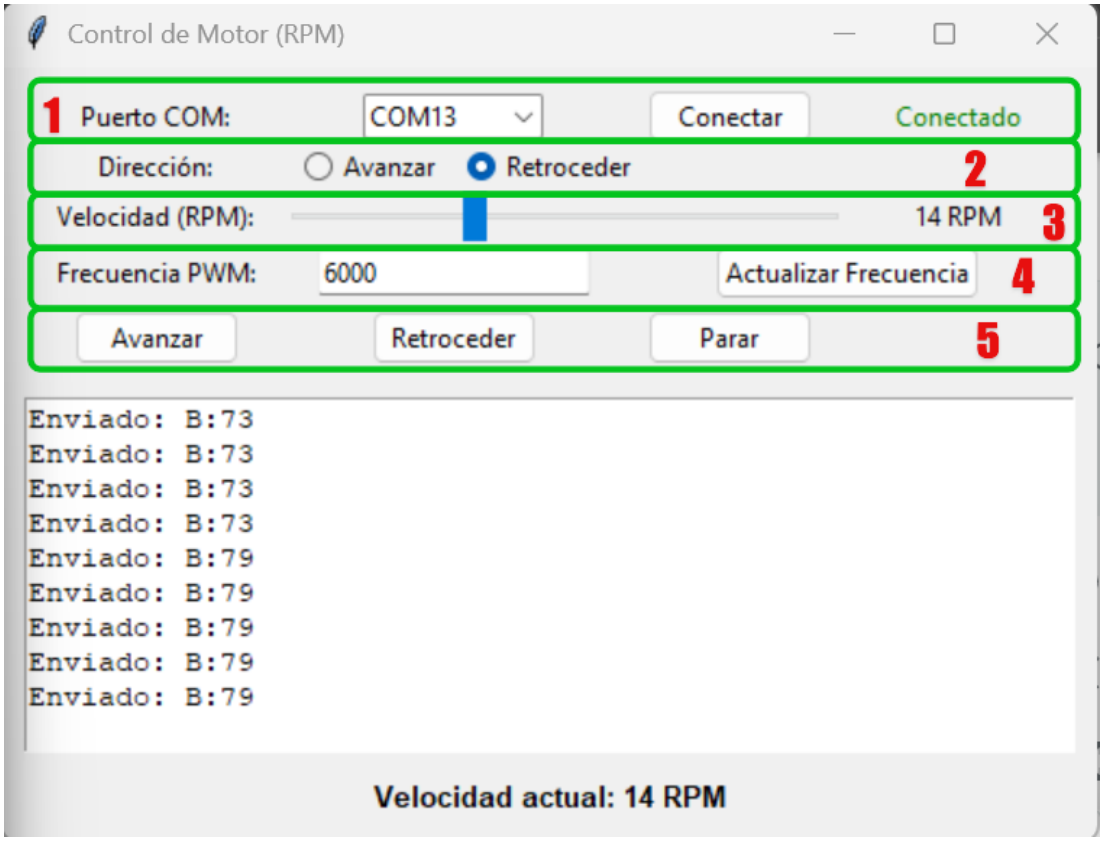


Figura 5.5: Interfaz gráfica para motor reductor.

1. Puerto COM: despliega una lista con los dispositivos conectados. Al conectar el transmisor a la PC, debe seleccionarse en la lista y presionar el botón Conectar, si la conexión es exitosa, aparecerá una etiqueta que cambia de "*Desconectado*" a "*Conectado*", como se observa en la imagen de referencia.

2. Dirección: en esta sección se selecciona el sentido de giro del motor.

3. Velocidad (RPM): cuenta con un control deslizante (*slider*), mediante el cual se ajustan las revoluciones por minuto. Conforme se desplaza el control, la etiqueta ubicada a la derecha indica la velocidad actual, esto permite realizar arranques y paradas suaves del motor.

4. Frecuencia PWM: en este apartado se introduce la frecuencia de operación del motor. Esta característica permite optimizar el consumo de corriente eléctrica, pudiendo ajustarse según la prueba para obtener mayor torque o velocidad; una vez ingresado el valor deseado, se debe presionar el botón "$Actualizar\,frecuencia$".

5. Botones de control: incluyen opciones para cambiar el sentido de giro del motor y un botón de paro, que puede emplearse tanto para detener el motor en una situación de emergencia como para un apagado normal.

Capítulo 6

Resultados

6.1. Obtención de base de datos en estado saludable

Para las pruebas realizadas en este estudio, se aplicará un período de muestreo de 10 ms, equivalente a una frecuencia de 100 Hz. Esta elección resulta suficiente por el momento, ya que permite evitar la saturación de la comunicación inalámbrica garantizando un funcionamiento estable y confiable durante la recolección de datos.

6.1.1. Primera etapa

Se realizaron 4 pruebas afuera del edificio FCE 6 de la Facultad de Ciencias de la Electrónica, **Figura 6.1**, con una duración de 4 minutos cada una **(Figura 6.2)** e intervalo de muestreo de 100 ms, obteniendo 4456 filas de datos, **Figura 6.3**, que al multiplicar por los 18 ejes de los seis sensores se obtuvieron 80,208 lecturas en cada prueba, todo esto a 8 metros de distancia del aerogenerador.

Figura 6.1: Aerogenerador en funcionamiento.

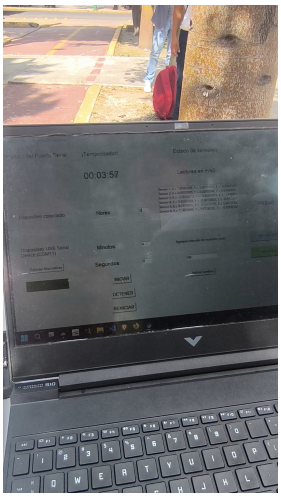


Figura 6.2: Programación del temporizador.

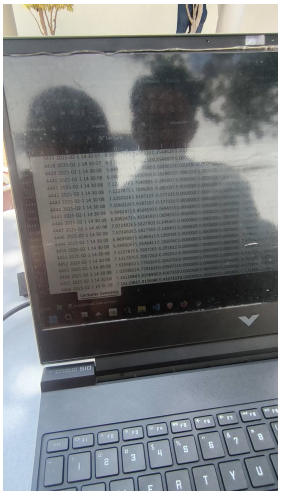


Figura 6.3: Base de datos obtenida.

Base de datos

Como se observa en la **Figura 6.4**, la base de datos en excel se organiza de la siguiente forma.

- **Primera columna**

De izquierda a derecha, encontramos el número de lectura; esto nos ayuda a ver cuántas lecturas se realizaron durante el tiempo programado.

- **Segunda columna Fecha**

Muestra el año, mes y día que se realizó cada lectura de los sensores.

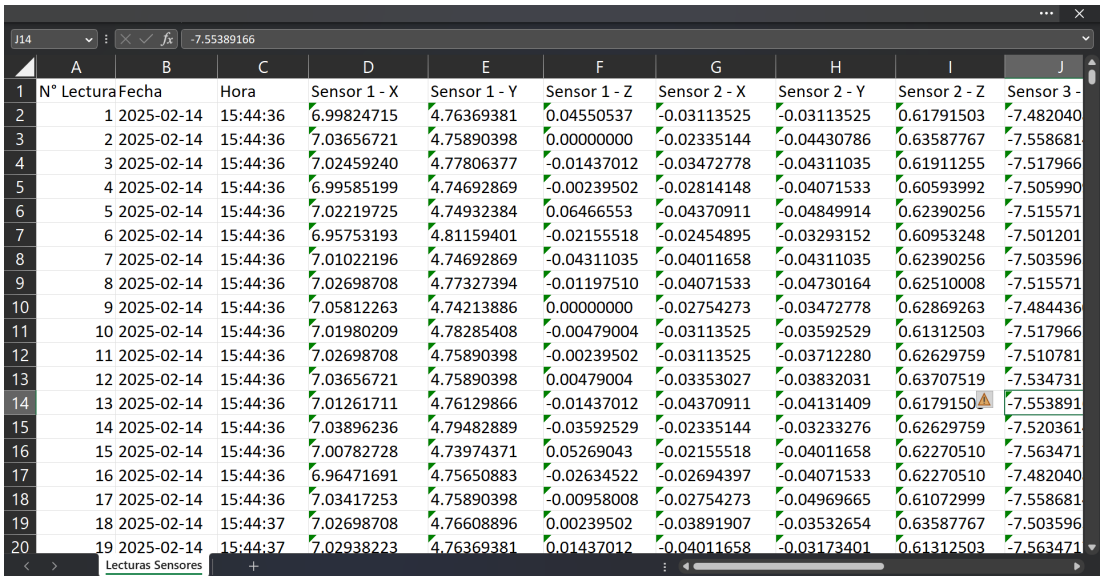
- **Tercera columna Hora**

Muestra la hora en formato 24 horas de cada lectura realizada por los sensores, comenzando por horas, minutos y segundos (hh:mm:ss).

- **Cuarta columna en adelante**

Cada sensor ocupa 3 columnas, una columna por cada eje (X, Y, Z), por ejemplo, la columna con la etiqueta “sensor 1 – X”, quiere decir que solo muestra las lecturas en el eje X, solo del sensor 1; ”Sensor 1 – Y”, indica que coloca las lecturas del eje Y, ”Sensor 1 – Z”, solo coloca las lecturas del eje Z del sensor 1. Esto se repite con los demás sensores del 1 al 6.

Cada vez que se genera una base de datos en excel, se guarda con la fecha y hora, por ejemplo “Lecturas Sensores 2025-02-14 15-48-38” y se guarda en automático en la carpeta del programa donde se encuentra el archivo ejecutable.



J14 -7.55389166

	A	B	C	D	E	F	G	H	I	J
1	N° Lectura	Fecha	Hora	Sensor 1 - X	Sensor 1 - Y	Sensor 1 - Z	Sensor 2 - X	Sensor 2 - Y	Sensor 2 - Z	Sensor 3 -
2	1	2025-02-14	15:44:36	6.99824715	4.76369381	0.04550537	-0.03113525	-0.03113525	0.61791503	-7.482040
3	2	2025-02-14	15:44:36	7.03656721	4.75890398	0.00000000	-0.02335144	-0.04430786	0.63587767	-7.558681
4	3	2025-02-14	15:44:36	7.02459240	4.77806377	-0.01437012	-0.03472778	-0.04311035	0.61911255	-7.517966
5	4	2025-02-14	15:44:36	6.99585199	4.74692869	-0.00239502	-0.02814148	-0.04071533	0.60593992	-7.505990
6	5	2025-02-14	15:44:36	7.02219725	4.74932384	0.06466553	-0.04370911	-0.04849914	0.62390256	-7.515571
7	6	2025-02-14	15:44:36	6.95753193	4.81159401	-0.02155518	-0.02454895	-0.03293152	0.60953248	-7.501201
8	7	2025-02-14	15:44:36	7.01022196	4.74692869	-0.04311035	-0.04011658	-0.04311035	0.62390256	-7.503596
9	8	2025-02-14	15:44:36	7.02698708	4.77327394	-0.01197510	-0.04071533	-0.04730164	0.62510008	-7.515571
10	9	2025-02-14	15:44:36	7.05812263	4.74213886	0.00000000	-0.02754273	-0.03472778	0.62869263	-7.484436
11	10	2025-02-14	15:44:36	7.01980209	4.78285408	-0.00479004	-0.03113525	-0.03592529	0.61312503	-7.517966
12	11	2025-02-14	15:44:36	7.02698708	4.75890398	-0.00239502	-0.03113525	-0.03712280	0.62629759	-7.510781
13	12	2025-02-14	15:44:36	7.03656721	4.75890398	0.00479004	-0.03353027	-0.03832031	0.63707519	-7.534731
14	13	2025-02-14	15:44:36	7.01261711	4.76129866	-0.01437012	-0.04370911	-0.04131409	0.6179150	-7.553891
15	14	2025-02-14	15:44:36	7.03896236	4.79482889	-0.03592529	-0.02335144	-0.03233276	0.62629759	-7.520361
16	15	2025-02-14	15:44:36	7.00782728	4.73974371	0.05269043	-0.02155518	-0.04011658	0.62270510	-7.563471
17	16	2025-02-14	15:44:36	6.96471691	4.75650883	-0.02634522	-0.02694397	-0.04071533	0.62270510	-7.482040
18	17	2025-02-14	15:44:36	7.03417253	4.75890398	-0.00958008	-0.02754273	-0.04969665	0.61072999	-7.558681
19	18	2025-02-14	15:44:37	7.02698708	4.76608896	0.00239502	-0.03891907	-0.03532654	0.63587767	-7.503596
20	19	2025-02-14	15:44:37	7.02938223	4.76369381	0.01437012	-0.04011658	-0.03173401	0.61312503	-7.563471

Lecturas Sensores

Figura 6.4: Base de datos en excel.

6.1.2. Segunda etapa

Para la segunda etapa del experimento se realizaron cinco pruebas con las siguientes condiciones:

- Primera prueba: tuvo una duración de 3 minutos, con un giro constante de 10 rpm en el motor que mueve las aspas y un periodo de muestreo de 10 milisegundos. Se obtuvieron 22,360 datos por cada eje de los sensores, resultando en un total de 402,480 datos de los seis sensores, como se muestra en la **Figura 6.5**.

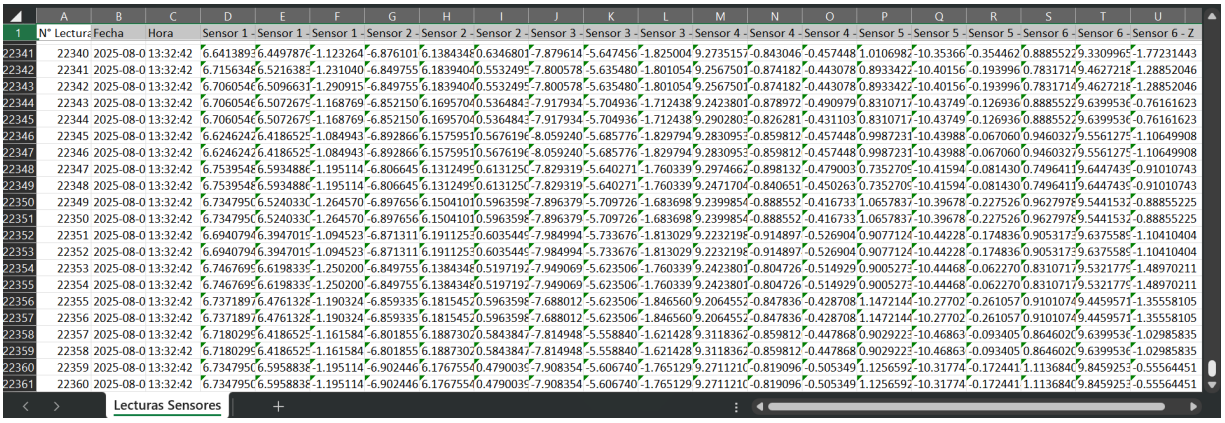


Figura 6.5: Base de datos, primera prueba.

Para una mejor interpretación de la base de datos obtenidas, se presentan las gráficas de aceleración (m/s²) en función del tiempo, donde este último corresponde a la hora en que se registraron las lecturas, mostrada en formato (hh:mm:ss).

En cada figura se incluyen las gráficas de los seis sensores por cada prueba realizada. Cada sensor cuenta con su respectiva gráfica, en la cual se representan sus tres ejes de aceleración: X en azul, Y en naranja y Z en verde.

La **Figura 6.6** muestra las gráficas de cada sensor de la base de datos a 10 rpm. En estas gráficas se observa que, dependiendo de la ubicación de cada sensor en la estructura, existe un desfase entre los valores positivos y negativos. Esto se debe a que cada sensor se encuentra en una posición distinta respecto a la referencia inicial, la cual corresponde al rango de valores (m/s²) X = 0, Y = 0, Z = 9.81.

Tomando la gráfica del sensor 1 se ve un rango de valores en los ejes X, Y de entre 6.6 a 6.9 m/s² y para el eje Z un rango de valores entre -1.1 a -1.4 m/s². Al ver esto como una señal, se observa que los valores se mantienen dentro de un rango constante, en el cual se puede traducir como un comportamiento lineal para los seis sensores.

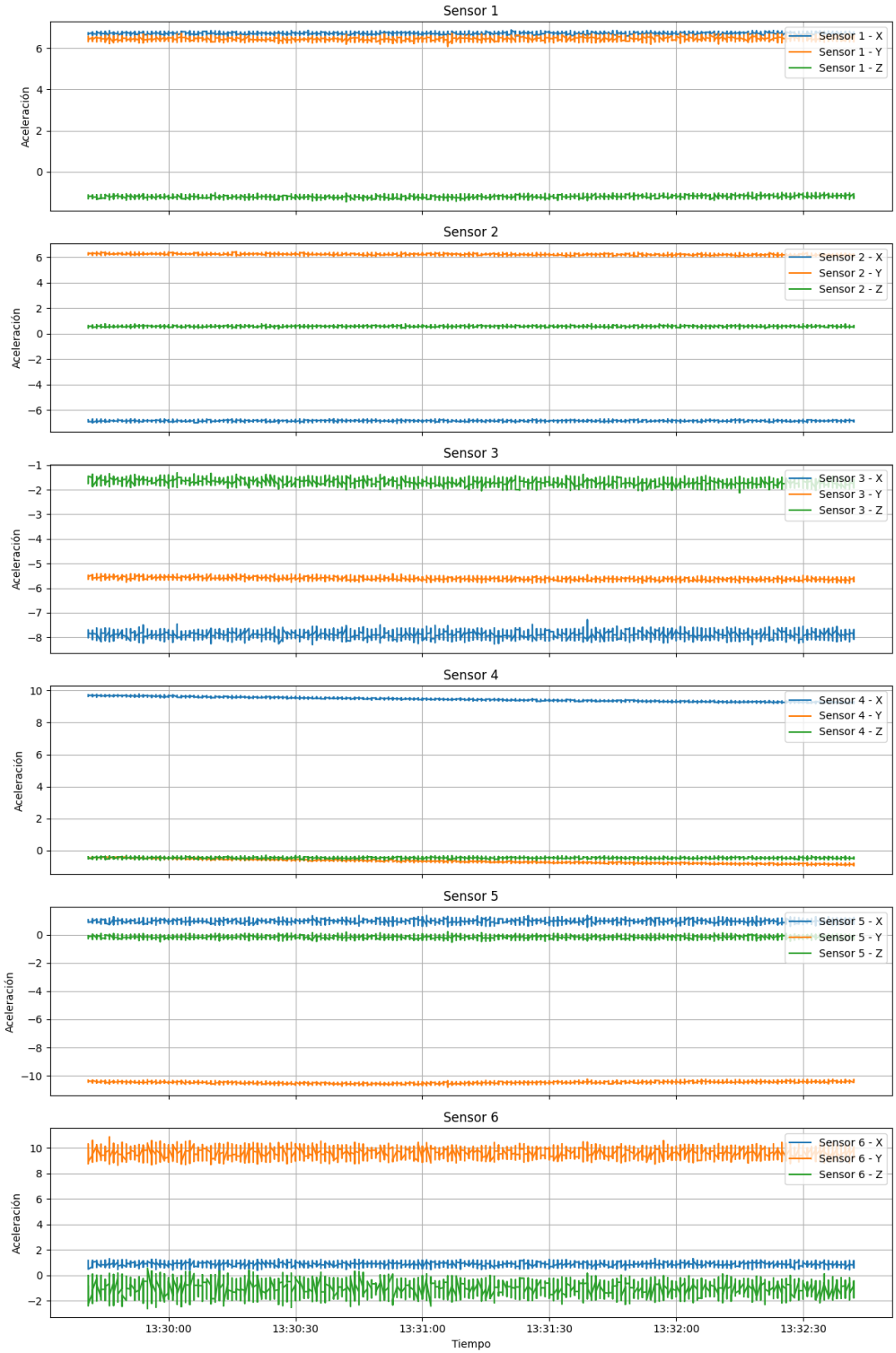


Figura 6.6: Gráficas de los 6 sensores en primera prueba (10 rpm).

- Segunda prueba: con una duración de 3 minutos, giro de las aspas a 14 rpm, un periodo de muestreo de 10 milisegundos se obtuvo una recolección de 383,076 datos en total por todos los ejes de los sensores; ver **Figura 6.7**.

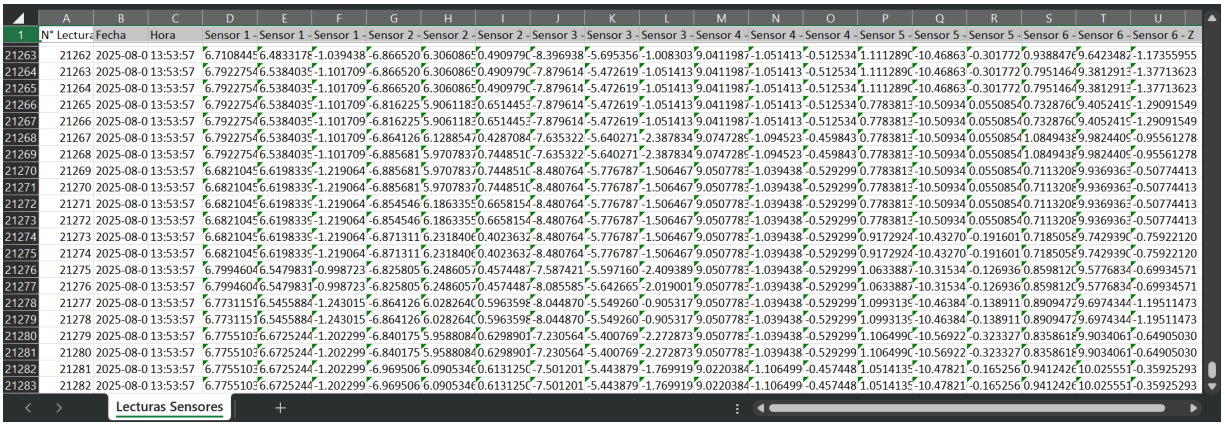


Figura 6.7: Base de datos, segunda prueba.

En la **Figura 6.8** se muestran las gráficas de la segunda prueba a 14 rpm. Se puede ver que al aumentar las rpm de las aspas, algunos sensores muestran variaciones en el rango de valores registrados. Por ejemplo, en la gráfica correspondiente al sensor 3 se observa que el rango de medición va de –7.3 a –8.5 en el eje X, –5.3 a –5.8 en el eje Y y –0.7 a –2.7 en el eje Z, mientras que en la prueba anterior los rangos eran de –7.7 a –8.1 en el eje X, –5.4 a –5.7 en el eje Y y –1.4 a 1.7 en el eje Z. No obstante, en ambos casos la señal mantiene un comportamiento lineal.

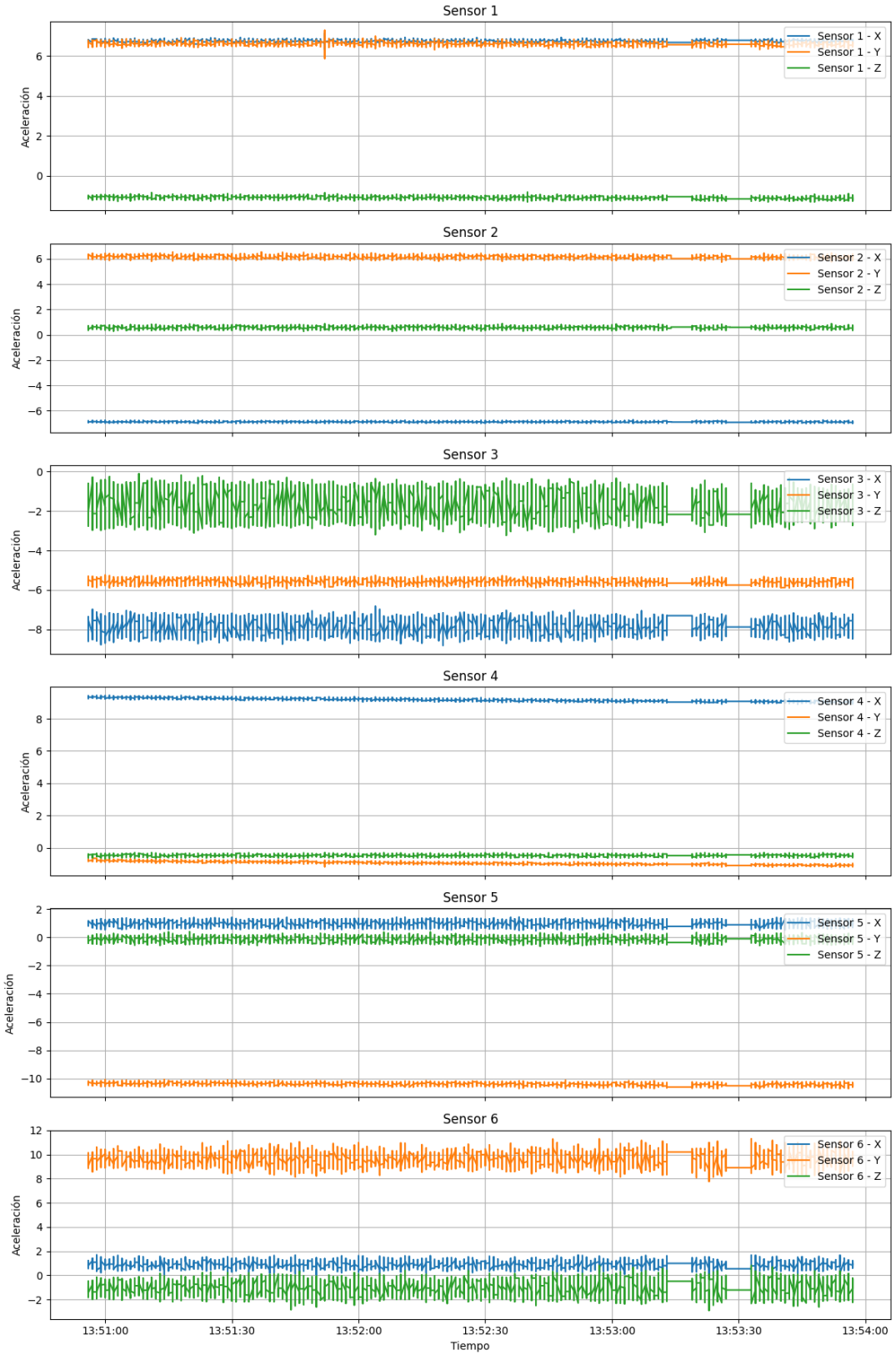
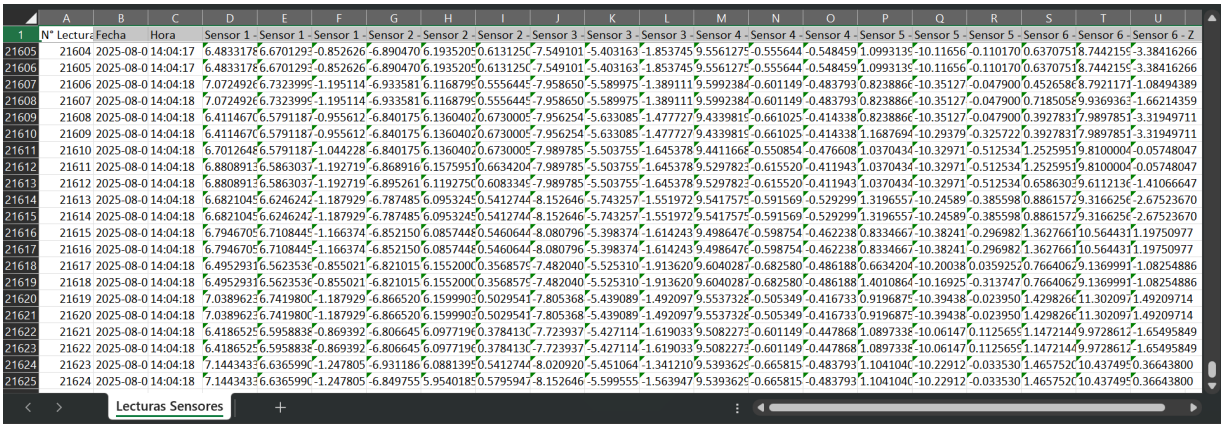


Figura 6.8: Gráficas de los 6 sensores en segunda prueba (14 rpm).

- Tercera prueba: duración de 3 minutos, giro de las aspas de 18 rpm, periodo de muestreo de 10 milisegundos y registro de 389,232 datos por todos los ejes de los sensores; ver **Figura 6.9**.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	N° Lectura	Fecha	Hora	Sensor 1 -	Sensor 1 -	Sensor 1 -	Sensor 2 -	Sensor 2 -	Sensor 2 -	Sensor 3 -	Sensor 3 -	Sensor 3 -	Sensor 4 -	Sensor 4 -	Sensor 4 -	Sensor 5 -	Sensor 5 -	Sensor 5 -	Sensor 6 -	Sensor 6 -	Sensor 6 -	Z
21605	21604	2025-08-0	14:04:17	6.483317	6.670129	-0.852626	-6.890470	6.193520	0.613125	-7.549101	-5.403163	-1.853745	9.556127	-0.555644	-0.548459	1.099313	-10.11656	-0.110170	0.637075	8.744215	-3.38416266
21606	21605	2025-08-0	14:04:17	6.483317	6.670129	-0.852626	-6.890470	6.193520	0.613125	-7.549101	-5.403163	-1.853745	9.556127	-0.555644	-0.548459	1.099313	-10.11656	-0.110170	0.637075	8.744215	-3.38416266
21607	21606	2025-08-0	14:04:18	7.072492	6.732399	-1.195114	-6.933581	6.116879	0.555644	-7.958650	-5.589975	-1.389111	9.599238	-0.601149	-0.483793	0.823886	-10.35127	-0.047900	0.452658	8.792117	-1.08494389
21608	21607	2025-08-0	14:04:18	7.072492	6.732399	-1.195114	-6.933581	6.116879	0.555644	-7.958650	-5.589975	-1.389111	9.599238	-0.601149	-0.483793	0.823886	-10.35127	-0.047900	0.718505	8.936936	-1.66214359
21609	21608	2025-08-0	14:04:18	6.411467	6.579118	-0.955612	-6.840175	6.136040	0.673000	-7.956254	-5.633085	-1.477727	9.433981	-0.661025	-0.414338	0.823886	-10.35127	-0.047900	0.392783	7.989785	-3.31949711
21610	21609	2025-08-0	14:04:18	6.411467	6.579118	-0.955612	-6.840175	6.136040	0.673000	-7.956254	-5.633085	-1.477727	9.433981	-0.661025	-0.414338	1.168769	-10.29379	0.325722	0.392783	7.989785	-3.31949711
21611	21610	2025-08-0	14:04:18	6.701264	6.579118	-1.044228	-6.840175	6.136040	0.673000	-7.989785	-5.503755	-1.645378	9.441166	-0.550854	-0.476608	1.037043	-10.32971	-0.512534	1.252595	9.810000	-0.05748047
21612	21611	2025-08-0	14:04:18	6.880891	6.586303	-1.192719	-6.868916	6.157595	0.663420	-7.989785	-5.503755	-1.645378	9.529782	-0.615520	-0.411943	1.037043	-10.32971	-0.512534	1.252595	9.810000	-0.05748047
21613	21612	2025-08-0	14:04:18	6.880891	6.586303	-1.192719	-6.895261	6.119275	0.608334	-7.989785	-5.503755	-1.645378	9.529782	-0.615520	-0.411943	1.037043	-10.32971	-0.512534	0.658630	9.611213	-1.41066647
21614	21613	2025-08-0	14:04:18	6.682104	6.624624	-1.187929	-6.787485	6.095324	0.541274	-8.152646	-5.743257	-1.551972	9.541757	-0.591569	-0.529299	1.319655	-10.24589	-0.385598	0.886157	9.316625	-2.67523670
21615	21614	2025-08-0	14:04:18	6.682104	6.624624	-1.187929	-6.787485	6.095324	0.541274	-8.152646	-5.743257	-1.551972	9.541757	-0.591569	-0.529299	1.319655	-10.24589	-0.385598	0.886157	9.316625	-2.67523670
21616	21615	2025-08-0	14:04:18	6.794670	6.710844	-1.166374	-6.852150	6.085744	0.546064	-8.080796	-5.398374	-1.614243	9.498647	-0.598754	-0.462238	0.833466	-10.38241	-0.296982	1.362766	10.56443	1.19750977
21617	21616	2025-08-0	14:04:18	6.794670	6.710844	-1.166374	-6.852150	6.085744	0.546064	-8.080796	-5.398374	-1.614243	9.498647	-0.598754	-0.462238	0.833466	-10.38241	-0.296982	1.362766	10.56443	1.19750977
21618	21617	2025-08-0	14:04:18	6.495293	6.562353	-0.855021	-6.821015	6.155200	0.356857	-7.482040	-5.525310	-1.913620	9.604028	-0.682580	-0.486188	0.663420	-10.20038	0.035925	0.766406	9.136999	-1.08254886
21619	21618	2025-08-0	14:04:18	6.495293	6.562353	-0.855021	-6.821015	6.155200	0.356857	-7.482040	-5.525310	-1.913620	9.604028	-0.682580	-0.486188	1.401086	-10.16925	-0.313747	0.766406	9.136999	-1.08254886
21620	21619	2025-08-0	14:04:18	7.038962	6.741980	-1.187929	-6.866520	6.159990	0.502954	-7.805368	-5.439089	-1.492097	9.553732	-0.505349	-0.416733	0.919687	-10.39438	-0.023950	1.429826	11.30209	1.49209714
21621	21620	2025-08-0	14:04:18	7.038962	6.741980	-1.187929	-6.866520	6.159990	0.502954	-7.805368	-5.439089	-1.492097	9.553732	-0.505349	-0.416733	0.919687	-10.39438	-0.023950	1.429826	11.30209	1.49209714
21622	21621	2025-08-0	14:04:18	6.418652	6.595883	-0.869392	-6.806645	6.097719	0.378413	-7.723937	-5.427114	-1.619033	9.508227	-0.601149	-0.447868	1.089733	-10.06147	0.112565	1.147214	9.972861	-1.65495849
21623	21622	2025-08-0	14:04:18	6.418652	6.595883	-0.869392	-6.806645	6.097719	0.378413	-7.723937	-5.427114	-1.619033	9.508227	-0.601149	-0.447868	1.089733	-10.06147	0.112565	1.147214	9.972861	-1.65495849
21624	21623	2025-08-0	14:04:18	7.144343	6.636599	-1.247805	-6.931186	6.088139	0.541274	-8.020920	-5.451064	-1.341210	9.539362	-0.665815	-0.483793	1.104104	-10.22912	-0.033530	1.465752	10.43749	0.36643800
21625	21624	2025-08-0	14:04:18	7.144343	6.636599	-1.247805	-6.849755	5.954018	0.579594	-8.152646	-5.599555	-1.563947	9.539362	-0.665815	-0.483793	1.104104	-10.22912	-0.033530	1.465752	10.43749	0.36643800

Figura 6.9: Base de datos, tercera prueba.

En la **Figura 6.10** se presenta la gráfica correspondiente a la tercera prueba realizada a 18 rpm. En ella se aprecia un rango de valores con comportamiento lineal, evidenciando incrementos asociados al aumento de la velocidad de giro de las aspas.

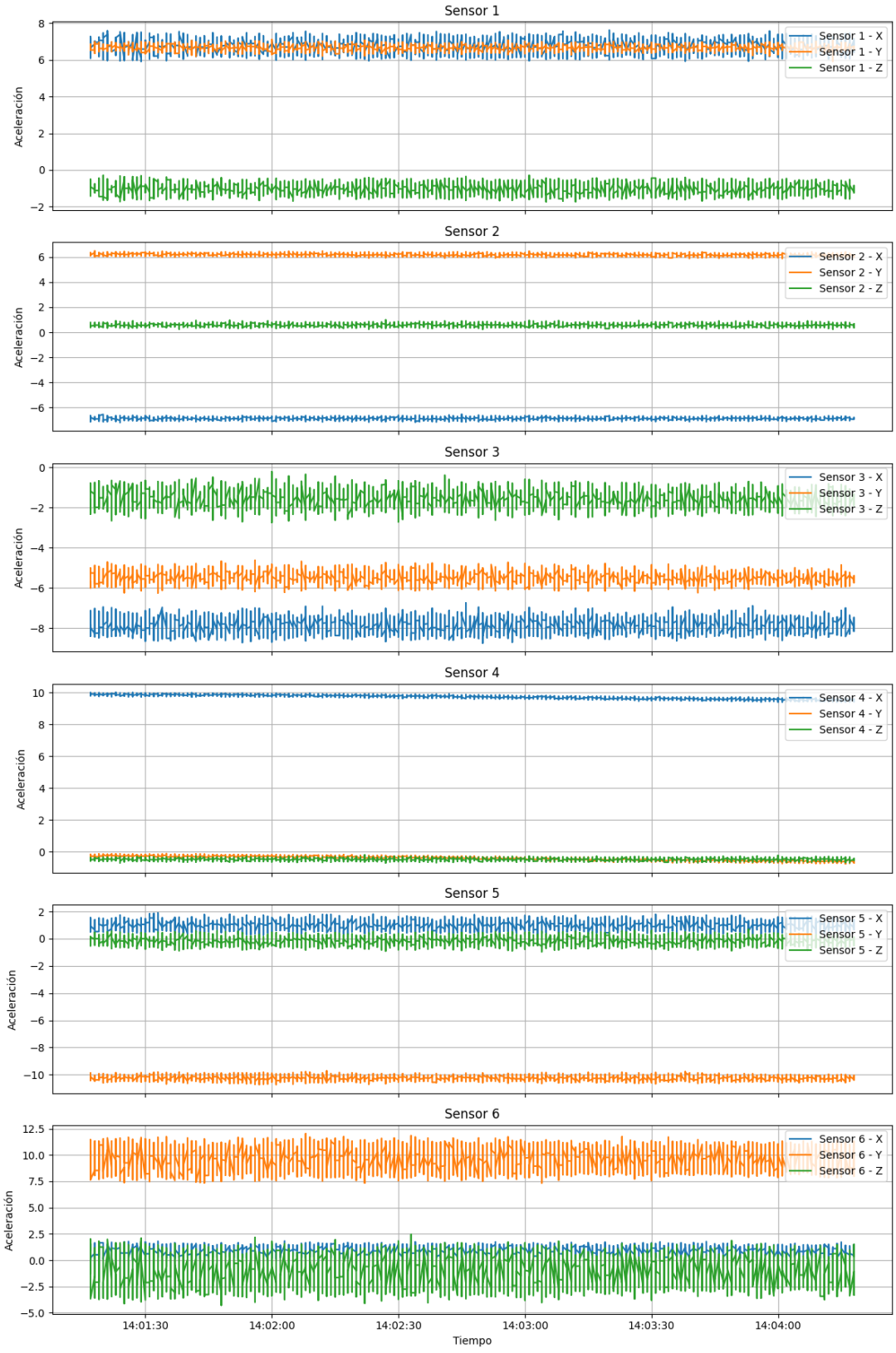
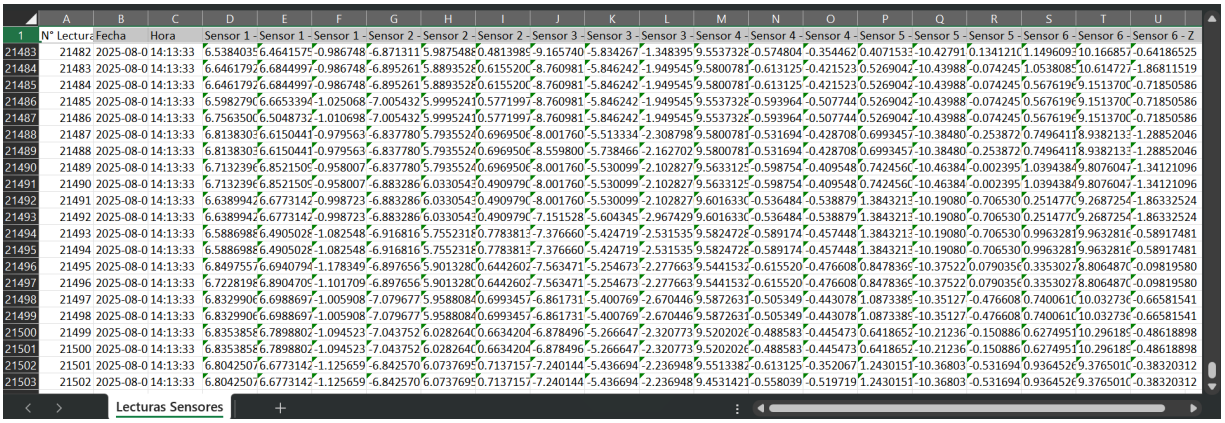


Figura 6.10: Gráficas de los seis sensores en tercera prueba (18 rpm).

- Cuarta prueba: duración de 3 minutos, giro de las aspas a 22 rpm, periodo de muestreo de 10 milisegundos y recolección de 387,036 datos por todos los ejes de los sensores; ver **Figura 6.11**.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	N° Lectura	Fecha	Hora	Sensor 1 - Sensor 1 - Sensor 1 - Sensor 2 - Sensor 2 - Sensor 2 - Sensor 3 - Sensor 3 - Sensor 3 - Sensor 4 - Sensor 4 - Sensor 4 - Sensor 5 - Sensor 5 - Sensor 5 - Sensor 6 - Sensor 6 - Z																		
21483	21482	2025-08-0	14:13:33	6.538403	6.464157	-0.986748	-6.871311	5.987548	0.481398	-9.165740	-5.834267	-1.348395	9.553732	-0.574804	0.354462	0.407153	-10.42791	0.134121	1.149609	10.16685	-0.64186525
21484	21483	2025-08-0	14:13:33	6.646179	6.684499	-0.986748	-6.895261	5.889352	0.615520	-8.760981	-5.846242	-1.949545	9.580078	-0.613125	-0.421523	0.526904	-10.43988	-0.074245	1.053808	10.61472	-1.86811519
21485	21484	2025-08-0	14:13:33	6.646179	6.684499	-0.986748	-6.895261	5.889352	0.615520	-8.760981	-5.846242	-1.949545	9.580078	-0.613125	-0.421523	0.526904	-10.43988	-0.074245	0.567619	9.151370	-0.71850586
21486	21485	2025-08-0	14:13:33	6.598279	6.665339	-1.025068	-7.005432	5.999524	0.577199	-8.760981	-5.846242	-1.949545	9.553732	-0.593964	-0.507744	0.526904	-10.43988	-0.074245	0.567619	9.151370	-0.71850586
21487	21486	2025-08-0	14:13:33	6.756350	6.504873	-1.010698	-7.005432	5.999524	0.577199	-8.760981	-5.846242	-1.949545	9.553732	-0.593964	-0.507744	0.526904	-10.43988	-0.074245	0.567619	9.151370	-0.71850586
21488	21487	2025-08-0	14:13:33	6.813830	6.615044	-0.979563	-6.837780	5.793552	0.696950	-8.001760	-5.513334	-2.308798	9.580078	-0.531694	-0.428708	0.699345	-10.38480	0.253872	0.749641	8.938213	-1.28852046
21489	21488	2025-08-0	14:13:33	6.813830	6.615044	-0.979563	-6.837780	5.793552	0.696950	-8.559800	-5.738466	-2.162702	9.580078	-0.531694	-0.428708	0.699345	-10.38480	-0.253872	0.749641	8.938213	-1.28852046
21490	21489	2025-08-0	14:13:33	6.713239	6.852150	-0.958007	-6.837780	5.793552	0.696950	-8.001760	-5.530099	-2.102827	9.563312	-0.598754	-0.409548	0.742456	-10.46384	-0.002395	1.039438	9.807604	-1.34121096
21491	21490	2025-08-0	14:13:33	6.713239	6.852150	-0.958007	-6.883286	6.033054	0.490979	-8.001760	-5.530099	-2.102827	9.563312	-0.598754	-0.409548	0.742456	-10.46384	-0.002395	1.039438	9.807604	-1.34121096
21492	21491	2025-08-0	14:13:33	6.638994	6.677314	-0.998723	-6.883286	6.033054	0.490979	-8.001760	-5.530099	-2.102827	9.601633	-0.536484	-0.538879	1.384321	-10.19080	-0.706530	0.251477	9.268725	-1.86332524
21493	21492	2025-08-0	14:13:33	6.638994	6.677314	-0.998723	-6.883286	6.033054	0.490979	-7.151528	-5.604345	-2.967429	9.601633	-0.536484	-0.538879	1.384321	-10.19080	-0.706530	0.251477	9.268725	-1.86332524
21494	21493	2025-08-0	14:13:33	6.588698	6.490502	-1.082548	-6.916816	5.755231	0.778381	-7.376660	-5.424719	-2.531535	9.582472	-0.589174	-0.457448	1.384321	-10.19080	-0.706530	0.996328	9.963281	-0.58917481
21495	21494	2025-08-0	14:13:33	6.588698	6.490502	-1.082548	-6.916816	5.755231	0.778381	-7.376660	-5.424719	-2.531535	9.582472	-0.589174	-0.457448	1.384321	-10.19080	-0.706530	0.996328	9.963281	-0.58917481
21496	21495	2025-08-0	14:13:33	6.849755	6.694079	-1.178349	-6.897656	5.901328	0.644260	-7.563471	-5.254673	-2.277663	9.544153	-0.615520	-0.476608	0.847836	-10.37522	0.079035	0.335302	8.806487	-0.09819580
21497	21496	2025-08-0	14:13:33	6.722819	6.890470	-1.101709	-6.897656	5.901328	0.644260	-7.563471	-5.254673	-2.277663	9.544153	-0.615520	-0.476608	0.847836	-10.37522	0.079035	0.335302	8.806487	-0.09819580
21498	21497	2025-08-0	14:13:33	6.832990	6.698869	-1.005908	-7.079677	5.958808	0.699345	-6.861731	-5.400769	-2.670446	9.587263	-0.505349	-0.443078	1.087338	-10.35127	-0.476608	0.740061	10.03273	-0.66581541
21499	21498	2025-08-0	14:13:33	6.832990	6.698869	-1.005908	-7.079677	5.958808	0.699345	-6.861731	-5.400769	-2.670446	9.587263	-0.505349	-0.443078	1.087338	-10.35127	-0.476608	0.740061	10.03273	-0.66581541
21500	21499	2025-08-0	14:13:33	6.835385	6.789880	-1.094523	-7.043752	6.028264	0.663420	-6.878496	-5.266647	-2.320773	9.520202	-0.488583	-0.445473	0.641865	-10.21236	-0.150886	0.627495	10.29618	-0.48618898
21501	21500	2025-08-0	14:13:33	6.835385	6.789880	-1.094523	-7.043752	6.028264	0.663420	-6.878496	-5.266647	-2.320773	9.520202	-0.488583	-0.445473	0.641865	-10.21236	-0.150886	0.627495	10.29618	-0.48618898
21502	21501	2025-08-0	14:13:33	6.804250	6.677314	-1.125659	-6.842570	6.073769	0.713715	-7.240144	-5.436694	-2.236948	9.551338	-0.613125	-0.352067	1.243015	-10.36803	0.531694	0.936452	9.376501	-0.38320312
21503	21502	2025-08-0	14:13:33	6.804250	6.677314	-1.125659	-6.842570	6.073769	0.713715	-7.240144	-5.436694	-2.236948	9.453142	-0.558039	0.519719	1.243015	-10.36803	0.531694	0.936452	9.376501	-0.38320312

Lecturas Sensores

Figura 6.11: Base de datos, cuarta prueba.

En la **Figura 6.12** se muestra la gráfica de la cuarta prueba a 22 rpm.

Retomando los datos de los sensores 1 y 3, se observa que a 10 rpm los seis sensores se mantienen dentro de rangos más estrechos y con oscilaciones de menor magnitud. En cambio, a 22 rpm se aprecia en casi todos los sensores un incremento en la amplitud de las variaciones, lo cual indica que el aumento en la velocidad de giro de las aspas genera mayores vibraciones dinámicas en la estructura.

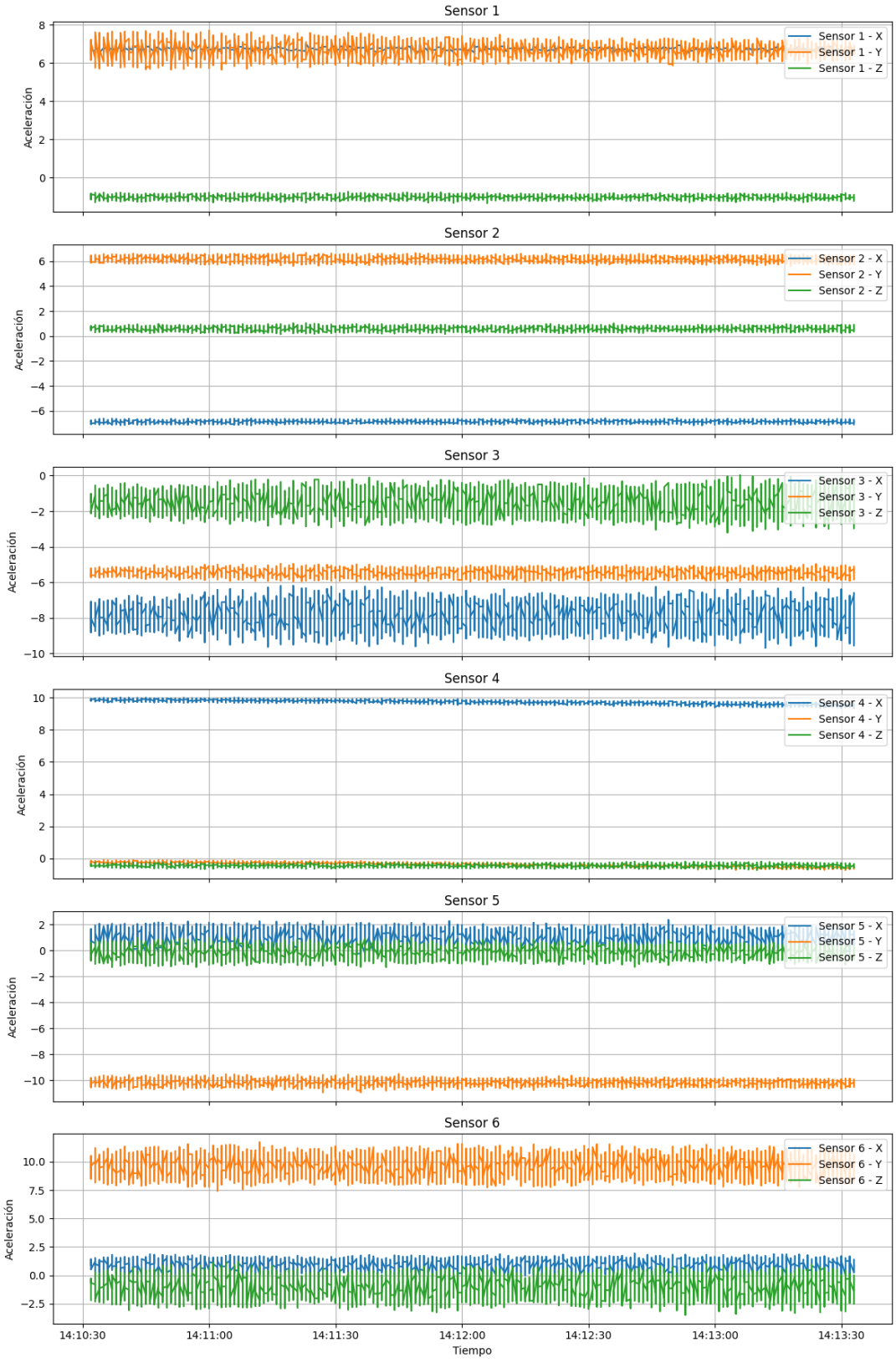
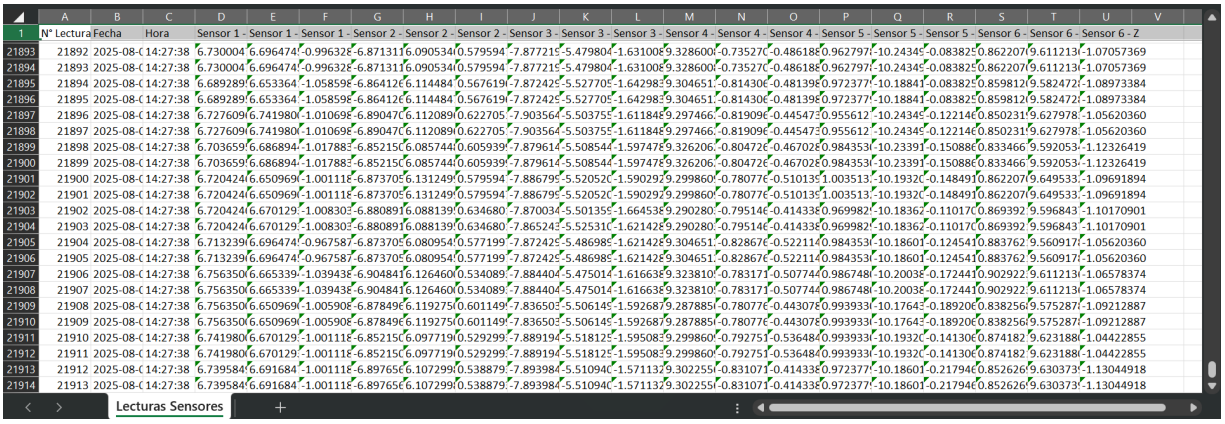


Figura 6.12: Gráficas de los seis sensores en cuarta prueba (22 rpm).

- Quinta prueba: a diferencia de las anteriores, se mantuvo una duración de 3 minutos, pero con un giro variable entre 10 y 22 rpm, aumentando y disminuyendo las revoluciones durante el tiempo establecido. El periodo de muestreo también fue de 10 milisegundos, obteniéndose 394,434 datos por todos los ejes de los sensores, ver **Figura 6.13** .



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	N° Lectura	Fecha	Hora	Sensor 1 - Sensor 1	Sensor 1 - Sensor 1	Sensor 1 - Sensor 2	Sensor 2 - Sensor 2	Sensor 2 - Sensor 3	Sensor 3 - Sensor 3	Sensor 3 - Sensor 4	Sensor 4 - Sensor 4	Sensor 4 - Sensor 5	Sensor 5 - Sensor 5	Sensor 5 - Sensor 6	Sensor 6 - Sensor 6	Sensor 6 - Z						
21893	21892	2025-08-(	14:27:38	6.730004	6.696474	-0.996328	6.871311	6.090534	0.579594	-7.877215	-5.479804	-1.631006	9.328600	-0.735270	-0.486188	0.962797	-10.24345	-0.083825	0.862207	9.611213	-1.07057369	
21894	21893	2025-08-(	14:27:38	6.730004	6.696474	-0.996328	6.871311	6.090534	0.579594	-7.877215	-5.479804	-1.631006	9.328600	-0.735270	-0.486188	0.962797	-10.24345	-0.083825	0.862207	9.611213	-1.07057369	
21895	21894	2025-08-(	14:27:38	6.689289	6.653364	-1.058598	6.864126	6.114484	0.567619	-7.872425	-5.527705	-1.642983	9.304651	-0.814306	-0.481398	0.972377	-10.18841	-0.083825	0.859812	9.582472	-1.08973384	
21896	21895	2025-08-(	14:27:38	6.689289	6.653364	-1.058598	6.864126	6.114484	0.567619	-7.872425	-5.527705	-1.642983	9.304651	-0.814306	-0.481398	0.972377	-10.18841	-0.083825	0.859812	9.582472	-1.08973384	
21897	21896	2025-08-(	14:27:38	6.727609	6.741980	-1.010698	6.890470	6.112089	0.622705	-7.903564	-5.503755	-1.611846	9.297466	-0.819096	-0.445473	0.955612	-10.24345	-0.122146	0.850231	9.627978	-1.05620360	
21898	21897	2025-08-(	14:27:38	6.727609	6.741980	-1.010698	6.890470	6.112089	0.622705	-7.903564	-5.503755	-1.611846	9.297466	-0.819096	-0.445473	0.955612	-10.24345	-0.122146	0.850231	9.627978	-1.05620360	
21899	21898	2025-08-(	14:27:38	6.703659	6.686894	-1.017883	6.852150	6.085744	0.605939	-7.879614	-5.508544	-1.597478	9.326206	-0.804726	-0.467028	0.984353	-10.23391	-0.150886	0.833466	9.592053	-1.12326419	
21900	21899	2025-08-(	14:27:38	6.703659	6.686894	-1.017883	6.852150	6.085744	0.605939	-7.879614	-5.508544	-1.597478	9.326206	-0.804726	-0.467028	0.984353	-10.23391	-0.150886	0.833466	9.592053	-1.12326419	
21901	21900	2025-08-(	14:27:38	6.720424	6.650969	-1.001118	6.873705	6.131249	0.579594	-7.886795	-5.520520	-1.590292	9.299860	-0.780776	-0.510135	1.003513	-10.19320	-0.148491	0.862207	9.649533	-1.09691894	
21902	21901	2025-08-(	14:27:38	6.720424	6.650969	-1.001118	6.873705	6.131249	0.579594	-7.886795	-5.520520	-1.590292	9.299860	-0.780776	-0.510135	1.003513	-10.19320	-0.148491	0.862207	9.649533	-1.09691894	
21903	21902	2025-08-(	14:27:38	6.720424	6.670129	-1.008303	6.880891	6.088139	0.634680	-7.870034	-5.501355	-1.664536	9.290280	-0.795146	-0.414338	0.969982	-10.18362	-0.110170	0.869392	9.596843	-1.10170901	
21904	21903	2025-08-(	14:27:38	6.720424	6.670129	-1.008303	6.880891	6.088139	0.634680	-7.865243	-5.525310	-1.621426	9.290280	-0.795146	-0.414338	0.969982	-10.18362	-0.110170	0.869392	9.596843	-1.10170901	
21905	21904	2025-08-(	14:27:38	6.713239	6.696474	-0.967587	6.873705	6.080954	0.577199	-7.872425	-5.486985	-1.621426	9.304651	-0.828676	-0.522114	0.984353	-10.18601	-0.124541	0.883762	9.560917	-1.05620360	
21906	21905	2025-08-(	14:27:38	6.713239	6.696474	-0.967587	6.873705	6.080954	0.577199	-7.872425	-5.486985	-1.621426	9.304651	-0.828676	-0.522114	0.984353	-10.18601	-0.124541	0.883762	9.560917	-1.05620360	
21907	21906	2025-08-(	14:27:38	6.756350	6.665339	-1.039438	6.904841	6.126460	0.534089	-7.884404	-5.475014	-1.616636	9.323810	-0.783171	-0.507744	0.986748	-10.20038	-0.172441	0.902922	9.611213	-1.06578374	
21908	21907	2025-08-(	14:27:38	6.756350	6.665339	-1.039438	6.904841	6.126460	0.534089	-7.884404	-5.475014	-1.616636	9.323810	-0.783171	-0.507744	0.986748	-10.20038	-0.172441	0.902922	9.611213	-1.06578374	
21909	21908	2025-08-(	14:27:38	6.756350	6.650969	-1.005908	6.878496	6.119275	0.601149	-7.836503	-5.506145	-1.592687	9.287885	-0.780776	-0.443078	0.993933	-10.17643	-0.189206	0.838256	9.575287	-1.09212887	
21910	21909	2025-08-(	14:27:38	6.756350	6.650969	-1.005908	6.878496	6.119275	0.601149	-7.836503	-5.506145	-1.592687	9.287885	-0.780776	-0.443078	0.993933	-10.17643	-0.189206	0.838256	9.575287	-1.09212887	
21911	21910	2025-08-(	14:27:38	6.741980	6.670129	-1.001118	6.852150	6.097719	0.529299	-7.889194	-5.518125	-1.595083	9.299860	-0.792751	-0.536484	0.993933	-10.19320	-0.141306	0.874182	9.623188	-1.04422855	
21912	21911	2025-08-(	14:27:38	6.741980	6.670129	-1.001118	6.852150	6.097719	0.529299	-7.889194	-5.518125	-1.595083	9.299860	-0.792751	-0.536484	0.993933	-10.19320	-0.141306	0.874182	9.623188	-1.04422855	
21913	21912	2025-08-(	14:27:38	6.739584	6.691684	-1.001118	6.897656	6.107299	0.538879	-7.893984	-5.510940	-1.571132	9.302255	-0.831071	-0.414338	0.972377	-10.18601	-0.217946	0.852626	9.630373	-1.13044918	
21914	21913	2025-08-(	14:27:38	6.739584	6.691684	-1.001118	6.897656	6.107299	0.538879	-7.893984	-5.510940	-1.571132	9.302255	-0.831071	-0.414338	0.972377	-10.18601	-0.217946	0.852626	9.630373	-1.13044918	

Lecturas Sensores

Figura 6.13: Base de datos, quinta prueba.

En la figura **Figura 6.14** se muestra la gráfica de la quinta prueba. Estas gráficas respaldan el comportamiento observado en las pruebas anteriores, mostrando que al variar la velocidad de giro de las aspas el rango de valores de aceleración se incrementa, pero sin perder la linealidad asociada a un estado saludable. Se aprecian tanto fases transitorias, donde los valores presentan mayores oscilaciones, como fases más estables en las que la señal se mantiene en un rango constante (aspas detenidas). Este comportamiento es coherente con la dinámica real de un aerogenerador, en el que la velocidad de giro no siempre permanece fija. Cabe destacar que, a diferencia de un escenario de falla, donde se esperaría una pérdida de linealidad o la aparición de patrones irregulares, las señales aquí obtenidas confirman que el sistema opera bajo condiciones normales.

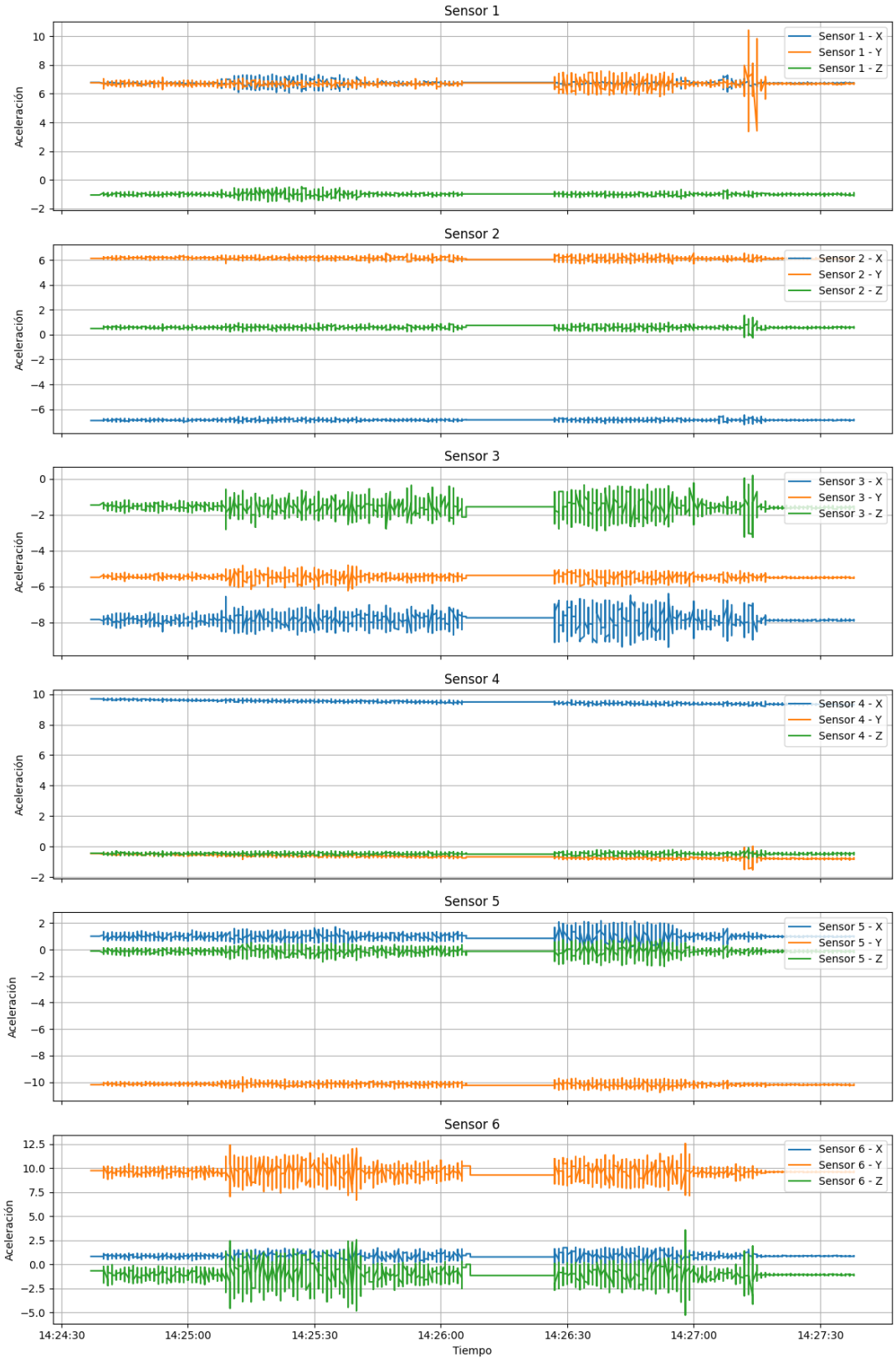


Figura 6.14: Gráficas de los seis sensores en quinta prueba (variable).

6.2. Producción de electricidad en el aerogenerador

Para el aerogenerador construido se usó el motor generador, el cual puede generar 12,000 W **Figura 6.15** y **Figura 6.16**, con una velocidad de operación de 500 rpm, frecuencia de 50 Hz, eficiencia de 85 % y vida útil de 20 años.

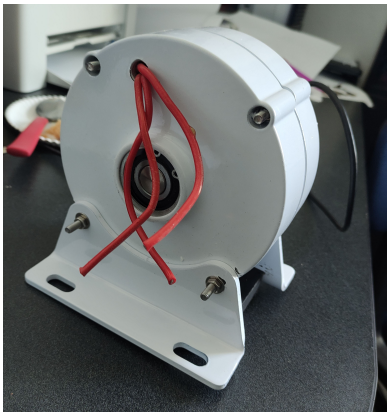


Figura 6.15: Motor generador.

Figura 6.16: Especificaciones del generador.

También se usó el controlador de 12 V el cual recibe la energía del generador y la convierte a voltaje continuo para poder almacenarlo en una batería de 12 V.

Realizando pruebas en el laboratorio, antes de ensamblar el aerogenerador, se conectó una tira de 5 LEDs con especificaciones de funcionamiento de 12 V y 1.75 A. Se hicieron girar manualmente el disco donde van montadas las aspas, como se muestra en la **Figura 6.17**. Al no tener peso adicional que ofreciera resistencia, el disco podía alcanzar mayores revoluciones, lo que permitió que los LEDs se encendieran, como también se observa en la **Figura 6.18**.

Figura 6.17: Girando disco con la mano.

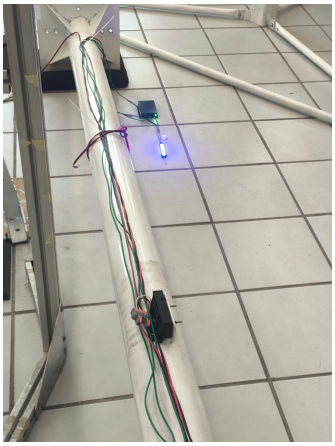


Figura 6.18: LEDs encendidos.

En las pruebas realizadas afuera del edificio FCE6 sin el multiplicador solo se produjo 4.67 V en DC, **Figura 6.19**, lo que no fue suficiente para encender una tira de luces LED que funcionan a 12 V.

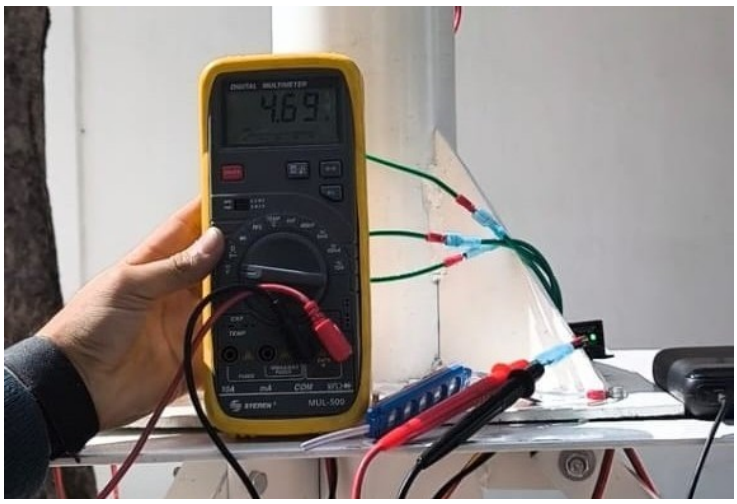


Figura 6.19: Prueba de generar energía.

Capítulo 7

Conclusiones

El prototipo de aerogenerador desarrollado demostró ser completamente funcional como banco de pruebas, permitiendo la obtención de bases de datos en estado saludable de manera inalámbrica y cumpliendo con los alcances planteados al inicio de esta tesis.

En las gráficas de aceleración registradas por cada sensor se aprecia un comportamiento lineal en términos generales, lo cual corresponde a datos en crudo que aún requieren un análisis posterior para filtrar ruido y otros factores externos. No obstante, se observa que a medida que aumenta el giro de las aspas (de 10 rpm hasta 22 rpm), la amplitud de las variaciones en las lecturas también incrementa, manteniendo al mismo tiempo un rango de valores relativamente constante. Este comportamiento indica que el sistema se encuentra en un estado saludable, pues no se aprecian anomalías o variaciones aleatorias que pudieran sugerir fallas estructurales.

El banco de pruebas no solo valida la adquisición de datos en condiciones normales, sino que abre la posibilidad de inducir fallas controladas (por ejemplo, aflojando pernos, retirando una barra o generando una fisura) con el fin de comparar las respuestas dinámicas frente a diferentes escenarios. Esto permitirá en estudios futuros establecer patrones característicos de fallas y avanzar hacia la predicción temprana de problemas estructurales en aerogeneradores.

La metodología de adquisición de datos (DAQ) empleada presenta la ventaja de ser fácilmente actualizable y de bajo costo, lo que constituye una herramienta accesible para proyectos de investigación y desarrollo. La comunicación inalámbrica fue validada con éxito, aunque con un alcance limitado a aproximadamente 8 metros. Este aspecto representa una oportunidad de mejora mediante la implementación de tecnologías de mayor cobertura, como LoRa, que permitirían extender la distancia de transmisión de datos sin sacrificar confiabilidad.

7.1. Trabajo a futuro

Con la base de datos obtenida se pueden desarrollar algoritmos de entrenamiento de la inteligencia artificial, para proceder a diagnosticar fallas en el aerogenerador realizando mantenimientos planificados.

Por lo tanto, el trabajo realizado en esta tesis es el fundamento que proporciona la

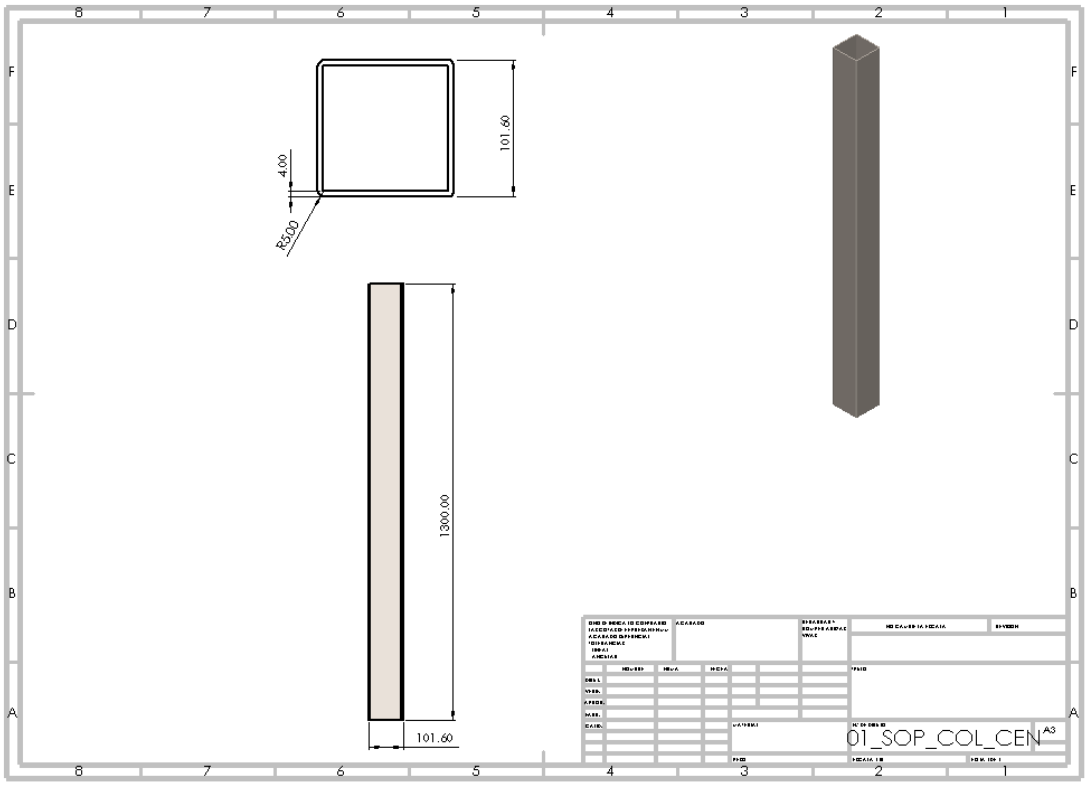
base de datos para obtener el diagrama en estado saludable.

Otro de los trabajos derivado de esta tesis es el diseño y la implementación del multiplicador. Como se observó en las pruebas de laboratorio, al hacer girar manualmente la base de las aspas a la velocidad necesaria fue posible generar la energía para encender las luces LED, pero en la práctica será necesario incorporar el multiplicador para obtener una mayor producción de energía. También es importante desarrollar el sistema de freno como medida de seguridad para el generador.

Anexos A

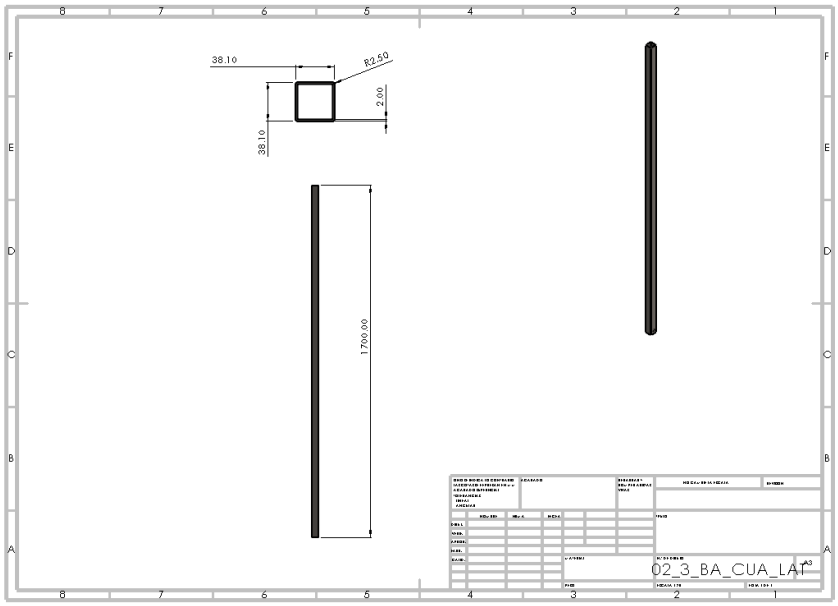
Anexos

A.1. Columna central



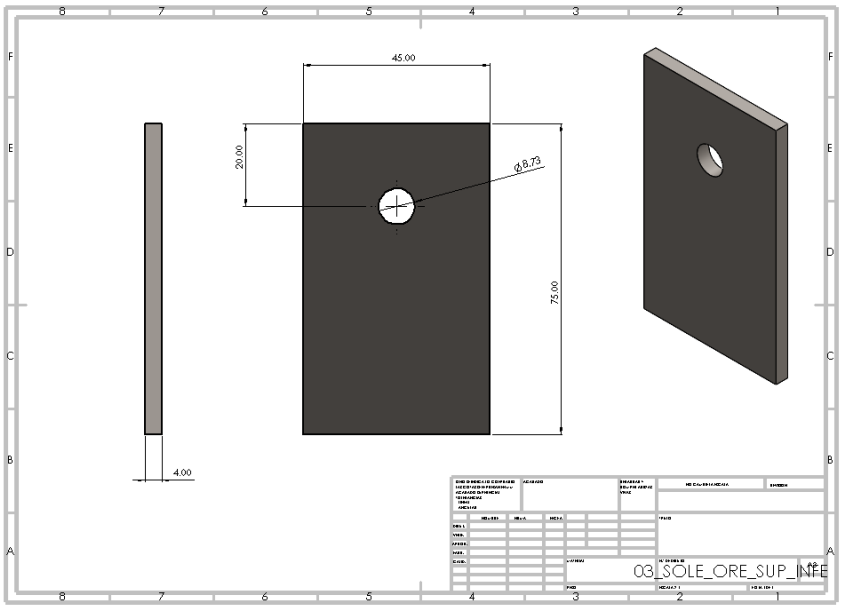
Anexo A.1: Plano de columna central.

A.2. Tubular refuerzo lateral



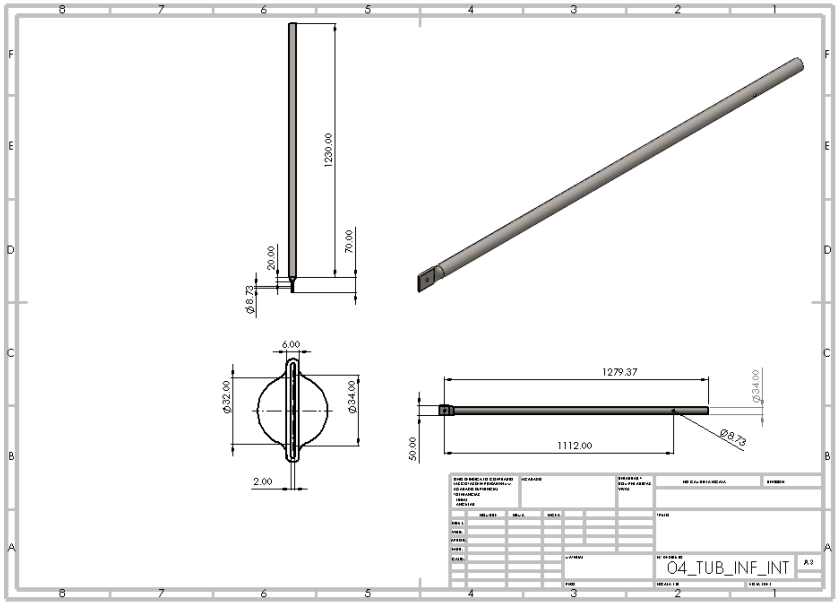
Anexo A.2: Plano de tubular refuerzo lateral.

A.3. Solera tipo aleta



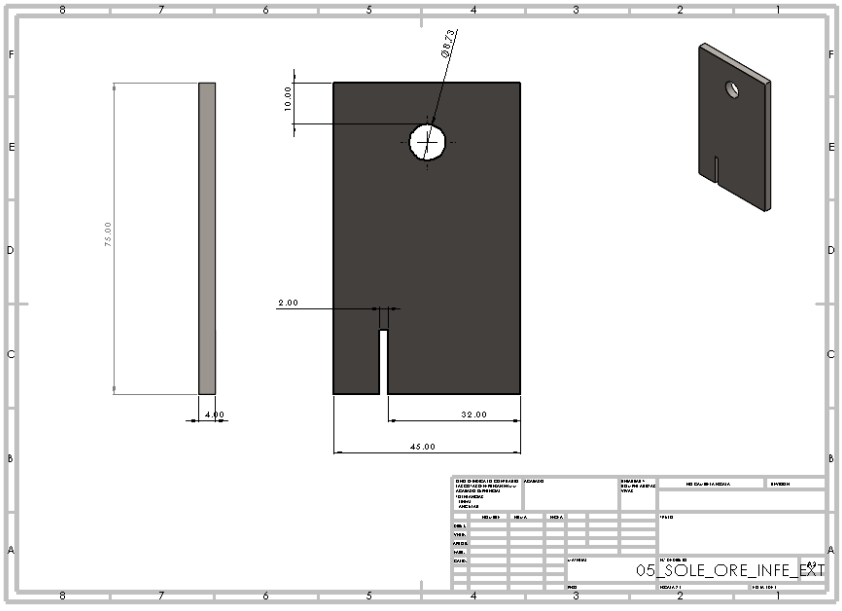
Anexo A.3: Plano de Solera tipo aleta.

A.4. Tubular inferior interno



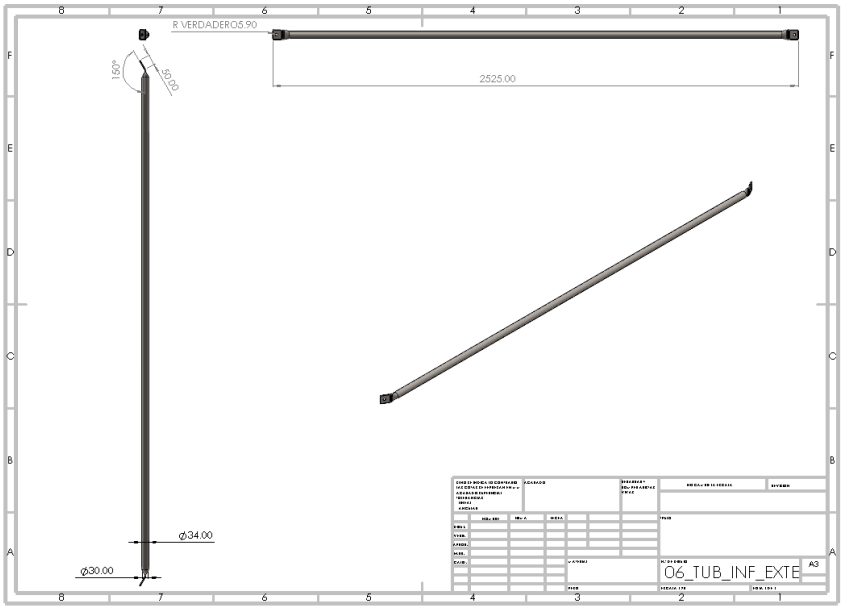
Anexo A.4: Plano de Tubular inferior interno.

A.5. Solera inferior externa



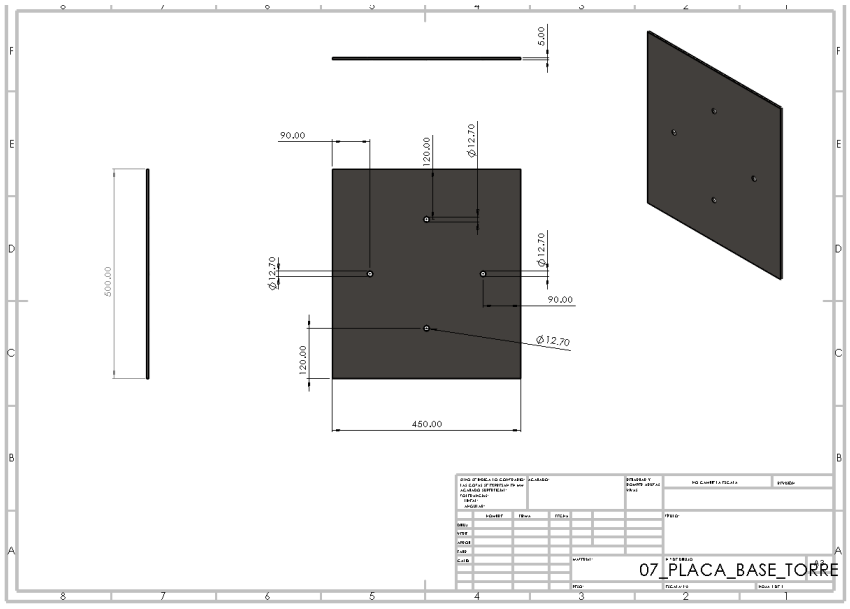
Anexo A.5: Plano de Solera inferior externa.

A.6. Tubular inferior externo



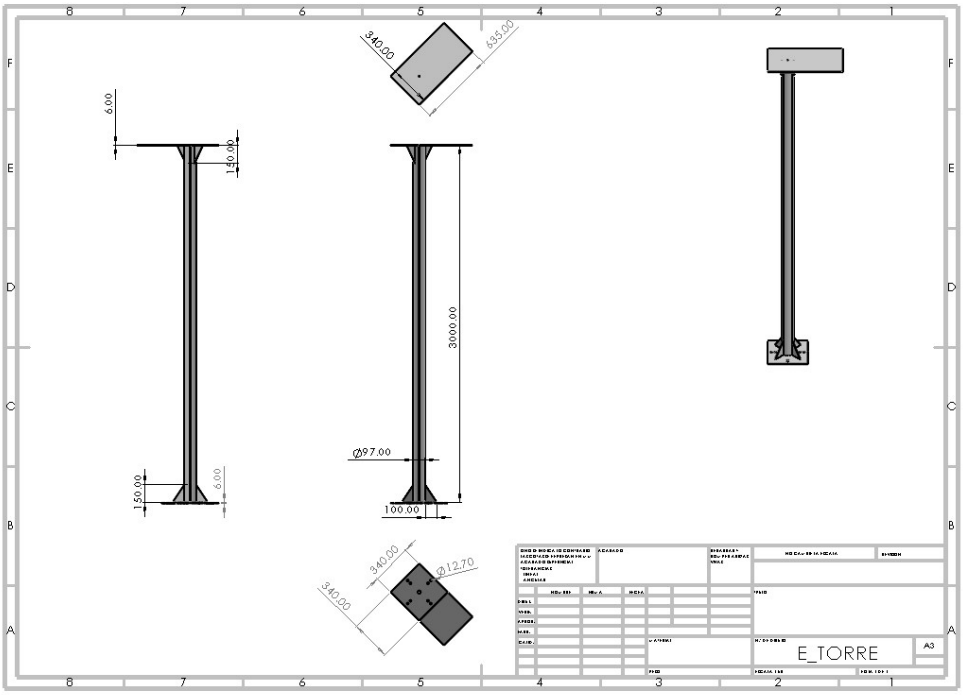
Anexo A.6: Plano de Tubular inferior externo.

A.7. Placa base



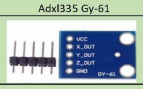
Anexo A.7: Plano de Tubular inferior externo.

A.8. Torre



Anexo A.8: Plano de torre.

A.9. Crossreference de sensores del tipo acelerómetro

CROSSREFERENCE ACELEROMETRO									
TIPO	PIEZOELÉCTRICO	ACELEROMETRO	ACELEROMETRO	ACELEROMETRO	INTERRUPTOR DE VIBRACION MECÁNICO	ACELEROMETRO	ACELEROMETRO	PIEZOELECTRICO	LASER,DESPLAZAMIENTO
MODELO	PCB Piezotronics 353B33	ADXL345	MPU-6050	ARD-394	Sensor Vibración KY-002	Acelerometro Analogico Adxl335 Gy-61	MPU-9250	Wilcoxon 786A	Keyence LK-G3000 Series
IMAGEN									
SENSIBILIDAD	(±5%)100 mV/g (10.19 mV/(m/s²))	40 mg/LSB	131 bits LSB / dps	////	NA	300 mV/g por eje	///	100 mV/g	100 mV/g
RANGO DE MEDICIÓN	±50 g pk (±491 m/s² pk)	± 2 g a ± 16 g	acelerometro± 2 g a ± 16 g grioscopio+/-250 a +/- 2000° /s(dps)	acelerometro± 2 g a ± 16 g grioscopio+/-250 a +/- 2000° /s(dps)	NA	±3g	± 2 ± 4 ± 8 ± 16 g	80 g	±0.5 mm hasta ±200 mm
RESOLUCIÓN DE BANDA ANCHA	0,0005 g rms (0.005 m/s² rms)	///	//	///	NA	NA	NA	0.5Hz ~ 14kHz	0.01 µm
RANGO DE FRECUENCIA	(±5%)1 a 4000 Hz	///	///	Acelerometro:1 kHz Giroscopio 200- 500 Hz	NA	1.6 kHz (X, Y) y 550 Hz (Z)	NA	30 kHz	50 kHz
PESO	27 g	2g	2g	2g	1g	1.5g	2g	90 g	480 g
DIMENCIONES	23.6 mm x 19.1 mm	3 mm × 5 mm × 1 mm	2.1 cm x 1.7 cm	2 cm x 1,6 cm	18.5 mm x 15 mm	21 x 15 mm	26 mm x 15.5 mm x 2.6 mm	///	///
VOLTAJE	18 - 30 VDC	3-5 V	3.3V a 5V	3,3 / 5 Vcc	3.3 V – 5 V	1.8 ~ 3.6 V	3V a 5V	18 a 30 V	24V
AMPERAJE	2 a 20 mA	0.1-40 µA		3.9 mA	5 mA	350 µA	3.2mA	///	500mA
DIRECCIONALIDAD (EJES DE MEDIDA)		3	6	6	NA	3	9	1	1
TEMPERATURA DE OPERACIÓN		(–40°C to +85°C)	-40°C a 85°C.	-40°C a 85°C.	////	-40°C a 85°C.	-40°C a 85°C.	-55°C ~ 120°C	0 a +50 °C
INTERFAZ		SPI,I2C	I2C			Analogica	I2C Y SPI		usb
PROVEEDOR	EBAY	MERCADO LIBRE , UNIT ELECTRONICS	MERCADO LIBRE	STEREN	UNIT ELECTRONICS	Tecneu	UNIT ELECTRONICS	Digi Key	EBAY
COSTO PESO MEXICANO	1572	65, 154 CON ENVIO	89	89	12	139	183	6925.41	33,992.86
DURABILIDAD O VIDA ÚTIL	/////	///	////	///	///	////	///	25 AÑOS	///
PAIS DE ORIGEN	USA	CHINA	CHINA	CHINA	CHINA	CHINA	CHINA	USA	Japón
MICROCONTROLADOR COMPATIBLE	Arduino,esp32 pero se debe usar un acondicionamiento para filtara y adaptar la señal	Arduino,Esp32, STM32,Raspberry Pi Pico (RP2040)	Arduino,Esp32, STM32,Raspberry Pi Pico (RP2040)	Arduino,Esp32, STM32,Raspberry Pi Pico (RP2040)	Arduino,Raspberry Pi	Arduino,Esp32, STM32,Raspberry Pi Pico (RP2040)	Arduino,Esp32, STM32,Raspberry Pi Pico (RP2040)	Arduino,Esp32, STM32	LK-G3001V
LINK DE DATASHEET	https://www.pcb.com/products?m=353b33	https://www.analog.com/media/en/technical-documentation/data-sheets/adxl345.pdf	https://uelectronics.com/producto/imu-mpu6050-6-grados-de-libertad/?srsltid=AfmBOop8fZNiW29xTsrSO6YGhc5VMaWheXyzGwInfr1tun_pBiEfYGul	https://www.steren.com.mx/sensor-con-acelerometro-y-giroscopio.html	https://uelectronics.com/producto/ky-002-sensor-vibracion/?srsltid=AfmBOopZjYZX0Us-DU12gVOxC0UjrBpylsLCZFwtAxNPMJIDNM-9kMip	https://www.tecneu.com/products/acelerometro-analogico-adxl335-gy-61-3-ejes?srsltid=AfmBOopnl_VxV8s7maGpy8lgMc-QQVPvSLVigv9Z6vUbWL6k0qx2r3cS	https://uelectronics.com/producto/mpu-9250-imu-de-9dof-9250/?srsltid=AfmBOoql-z35khkdgEJD-7lRpraYEV54CQVehK_zYiOhvT79mFAts-Cf	https://wilcoxon.com/786a/	https://www.keyence.com.mx/products/measure/laser-1d/lk-g3000/specs/
LINK DE COMPRA	https://www.ebay.com/itm/116307204481?_nkw=pcb+piezotronics&itmmeta=01J71W2PENRHQ8GSF2H61ANHPD&hash=item1b14731981:g:o4UAAOSwQ7lmz3~1	https://articulo.mercadolibre.com.mx/MLM-1400481699-acelerometro-adxl345-gy-291-_JM?matt_tool=74125127&matt_word=&matt_source=google&matt_campaign_id=19641074588&matt_ad_group_id=	https://uelectronics.com/producto/imu-mpu6050-6-grados-de-libertad/?srsltid=AfmBOop8fZNiW29xTsrSO6YGhc5VMaWheXyzGwInfr1tun_pBiEfYGul	https://www.steren.com.mx/sensor-con-acelerometro-y-giroscopio.html	https://uelectronics.com/producto/ky-002-sensor-vibracion/?srsltid=AfmBOooKv39JvxPeGrRQDBjbHZSdLvEqHYkmSTN0DY312_4GYqhosaC	https://www.tecneu.com/products/acelerometro-analogico-adxl335-gy-61-3-ejes?srsltid=AfmBOopnl_VxV8s7maGpy8lgMc-QQVPvSLVigv9Z6vUbWL6k0qx2r3cS	https://uelectronics.com/producto/mpu-9250-imu-de-9dof-9250/?srsltid=AfmBOoql-z35khkdgEJD-7lRpraYEV54CQVehK_zYiOhvT79mFAts-Cf	https://www.digikey.com.mx/es/products/detail/amphenol-wilcoxon-sensing-technologies/786A/9947286	https://www.ebay.com/itm/175082514165?chn=ps
NOTA	Es demaciado caro,pero es de alta precision, pero la adquisision de datos no es open source				Este sensor es útil en aplicaciones donde solo se necesita saber si hubo movimiento o vibración sin necesidad de medir la intensidad de la vibración.	La salida analógica facilita el uso en aplicaciones donde no se requieran interfaces digitales complejas como I2C o SPI, pero requiere la capacidad de manejar señales analógicas en el sistema donde se integre.			Necesitan controlador de la misma serie

Anexo A.9: Crossreference de sensores.

A.10. Código para calibración del MPU6050

El siguiente código fuente muestra la implementación para calibrar el sensor MPU6050 utilizando una ESP32.

Listing A.10.1: Código para calibración del MPU6050

```
*/// I2Cdev and MPU6050 must be installed as libraries
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
//////////////////  CONFIGURATION //////////////////////
//Change this 3 variables if you want to fine tune the skecth to your needs.
int buffersize=1000;     //Amount of readings used to average, make it higher to get more precision but sketch will be slower  (default:1000)
int acel_deadzone=8;     //Acelerometer error allowed, make it lower to get more precision, but sketch may not converge  (default:8)
int giro_deadzone=1;     //Giro error allowed, make it lower to get more precision, but sketch may not converge  (default:1)

// default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
//MPU6050 accelgyro;
MPU6050 accelgyro(0x68); // <-- use for AD0 high

int16_t ax, ay, az,gx, gy, gz;

int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz,state=0;
int ax_offset,ay_offset,az_offset,gx_offset,gy_offset,gz_offset;

///////////////////////  SETUP  ////////////////////////////
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
Wire.begin(21, 18);  // SDA en GPIO21, SCL en GPIO22 (ESP32)
  // COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE
  //TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz). Leonardo measured 250kHz.

  // initialize serial communication
  Serial.begin(115200);

  // initialize device
  accelgyro.initialize();

  // wait for ready
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available()){
    Serial.println(F("Send any character to start sketch.\n"));
    delay(1500);
  }
  while (Serial.available() && Serial.read()); // empty buffer again

```

```
44    // start message
45    Serial.println("\nMPU6050 Calibration Sketch");
46    delay(2000);
47    Serial.println("\nYour MPU6050 should be placed in horizontal position,
          with package letters facing up. \nDon't touch it until you see a
          finish message.\n");
48    delay(3000);
49    // verify connection
50    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful
          " : "MPU6050 connection failed");
51    delay(1000);
52    // reset offsets
53    accelgyro.setXAccelOffset(0);
54    accelgyro.setYAccelOffset(0);
55    accelgyro.setZAccelOffset(0);
56    accelgyro.setXGyroOffset(0);
57    accelgyro.setYGyroOffset(0);
58    accelgyro.setZGyroOffset(0);
59  }
60  ////////////////////   LOOP   ////////////////////////////
61  void loop() {
62    if (state==0){
63      Serial.println("\nReading sensors for first time...");
64      meansensors();
65      state++;
66      delay(1000);
67    }
68
69    if (state==1) {
70      Serial.println("\nCalculating offsets...");
71      calibration();
72      state++;
73      delay(1000);
74    }
75
76    if (state==2) {
77      meansensors();
78      Serial.println("\nFINISHED!");
79      Serial.print("\nSensor readings with offsets:\t");
80      Serial.print(mean_ax);
81      Serial.print("\t");
82      Serial.print(mean_ay);
83      Serial.print("\t");
84      Serial.print(mean_az);
85      Serial.print("\t");
86      Serial.print(mean_gx);
87      Serial.print("\t");
88      Serial.print(mean_gy);
89      Serial.print("\t");
90      Serial.println(mean_gz);
91      Serial.print("Your offsets:\t");
92      Serial.print(ax_offset);
93      Serial.print("\t");
94      Serial.print(ay_offset);
```

```
95      Serial.print("\t");
96      Serial.print(az_offset);
97      Serial.print("\t");
98      Serial.print(gx_offset);
99      Serial.print("\t");
100     Serial.print(gy_offset);
101     Serial.print("\t");
102     Serial.println(gz_offset);
103     Serial.println("\nData is printed as: acelX acelY acelZ giroX giroY
            giroZ");
104     Serial.println("Check that your sensor readings are close to 0 0 16384 0
             0 0");
105     Serial.println("If calibration was succesful write down your offsets so
            you can set them in your projects using something similar to mpu.
            setXAccelOffset(youroffset)");
106     while (1);
107   }
108 }
109
110 ///////////////////////  FUNCTIONS   ////////////////////////////
111 void meansensors(){
112   long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;
113
114   while (i<(buffersize+101)){
115     // read raw accel/gyro measurements from device
116     accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
117
118     if (i>100 && i<=(buffersize+100)){ //First 100 measures are discarded
119       buff_ax=buff_ax+ax;
120       buff_ay=buff_ay+ay;
121       buff_az=buff_az+az;
122       buff_gx=buff_gx+gx;
123       buff_gy=buff_gy+gy;
124       buff_gz=buff_gz+gz;
125     }
126     if (i==(buffersize+100)){
127       mean_ax=buff_ax/buffersize;
128       mean_ay=buff_ay/buffersize;
129       mean_az=buff_az/buffersize;
130       mean_gx=buff_gx/buffersize;
131       mean_gy=buff_gy/buffersize;
132       mean_gz=buff_gz/buffersize;
133     }
134     i++;
135     delay(2); //Needed so we don't get repeated measures
136   }
137 }
138
139 void calibration(){
140   ax_offset=-mean_ax/8;
141   ay_offset=-mean_ay/8;
142   az_offset=(16384-mean_az)/8;
143
144   gx_offset=-mean_gx/4;
```

```
145   gy_offset=-mean_gy/4;
146   gz_offset=-mean_gz/4;
147   while (1){
148     int ready=0;
149     accelgyro.setXAccelOffset(ax_offset);
150     accelgyro.setYAccelOffset(ay_offset);
151     accelgyro.setZAccelOffset(az_offset);
152 
153     accelgyro.setXGyroOffset(gx_offset);
154     accelgyro.setYGyroOffset(gy_offset);
155     accelgyro.setZGyroOffset(gz_offset);
156 
157     meansensors();
158     Serial.println("...");
159 
160     if (abs(mean_ax)<=acel_deadzone) ready++;
161     else ax_offset=ax_offset-mean_ax/acel_deadzone;
162 
163     if (abs(mean_ay)<=acel_deadzone) ready++;
164     else ay_offset=ay_offset-mean_ay/acel_deadzone;
165 
166     if (abs(16384-mean_az)<=acel_deadzone) ready++;
167     else az_offset=az_offset+(16384-mean_az)/acel_deadzone;
168 
169     if (abs(mean_gx)<=giro_deadzone) ready++;
170     else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);
171 
172     if (abs(mean_gy)<=giro_deadzone) ready++;
173     else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);
174 
175     if (abs(mean_gz)<=giro_deadzone) ready++;
176     else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);
177 
178     if (ready==6) break;
179   }
180 }
```

A.11. Código para obtener la dirección MAC del receptor

Listing A.11.1: Código para obtener la dirección MAC del receptor

```
1
2  #include "WiFi.h"
3
4  void setup() {
5    Serial.begin(115200);
6    WiFi.mode(WIFI_STA);  // Configura el modo estación
7    delay(1000);          // Pequeña espera para asegurar la inicialización
8
9    Serial.println("Direccion MAC de la ESP32: ");
10   Serial.println(WiFi.macAddress());  // Obtener dirección MAC
11 }
12
13 void loop() {
14   delay(1000);
15 }
```

A.12. Código para lectura y transmisión de datos del sensor 1

Listing A.12.1: Código para lectura y transmisión de datos del sensor 1

```
1
2  ////SENSOR ID1
3  #include "I2Cdev.h"
4  #include "MPU6050.h"
5  #include "Wire.h"
6  #include "esp_now.h"
7  #include "WiFi.h"
8  #define SDA_PIN 8//3
9  #define SCL_PIN 9//4
10 MPU6050 sensor;
11
12 int16_t ax, ay, az;
13 int16_t gx, gy, gz;
14
15 long f_ax, f_ay, f_az;
16 int16_t p_ax, p_ay, p_az;
17 long f_gx, f_gy, f_gz;
18 int16_t p_gx, p_gy, p_gz;
19 int counter = 0;
20 int16_t ax_o = 5856, ay_o = 6231, az_o = 8946;
21 int16_t gx_o = 90, gy_o = -39, gz_o = -34;
22
23 //Direcion MAC del receptor // Direccion MAC de S2MINI FINAL:
24 //08:56:50.454 -> 80:65:99:4B:13:BC
25 uint8_t broadcastAddress[] = { 0x80, 0x65, 0x99, 0x4B, 0x13, 0xBC };
26
27 //Estructura del mensaje a enviar
28 typedef struct struct_message {
29   int id;
30   float x,y,z;
31 } struct_message;
32
33 struct_message myData;
34
35 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
36   Serial.print("\r\nEstado del ultimo paquete enviado\t");
37   Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Envio exitoso" : "Envio
       fallido");
38 }
39
40 void setup() {
41   Serial.begin(115200);
42   Wire.begin(SDA_PIN, SCL_PIN);
43   sensor.initialize();
44
45   sensor.setXAccelOffset(ax_o);
46   sensor.setYAccelOffset(ay_o);
47   sensor.setZAccelOffset(az_o);
```

```
48
49    sensor.setXGyroOffset(gx_o);
50    sensor.setYGyroOffset(gy_o);
51    sensor.setZGyroOffset(gz_o);
52
53    Serial.println("Offsets asignados correctamente");
54
55     WiFi.mode(WIFI_STA);
56
57    if (esp_now_init() != ESP_OK) {
58      Serial.println("Error de inicializacion del protocolo ESP-NOW");
59      return;
60    }
61
62    esp_now_register_send_cb(OnDataSent);
63
64    esp_now_peer_info_t peerInfo;
65    memcpy(peerInfo.peer_addr, broadcastAddress, 6);
66    peerInfo.channel = 1;
67    peerInfo.encrypt = false;
68
69    if (esp_now_add_peer(&peerInfo) != ESP_OK){
70      Serial.println("No se pudo agregar el par");
71      return;
72    }
73  }
74
75  void loop() {
76    sensor.getAcceleration(&ax, &ay, &az);
77    sensor.getRotation(&gx, &gy, &gz);
78
79    // Convertir valores a unidades de aceleración (m/s^2) y velocidad angular
        (grados/s)
80    float accelerationX = (float)ax / 16384.0 * 9.81;
81    float accelerationY = (float)ay / 16384.0 * 9.81;
82    float accelerationZ = (float)az / 16384.0 * 9.81;
83
84   //* Mostrar los valores
85    Serial.print("Aceleración (m/s^2): ");
86    Serial.print(accelerationX); Serial.print("\t");
87    Serial.print(accelerationY); Serial.print("\t");
88    Serial.print(accelerationZ); Serial.println();
89
90    myData.id =1;
91    myData.x=accelerationX  ;
92    myData.y=  accelerationY;
93    myData.z=  accelerationZ;
94    esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *) &myData,
        sizeof(myData));
95
96    delay(10); // Esperar un poco entre lecturas
97  }
```

A.13. Código para modulo receptor

Listing A.13.1: Código para modulo receptor

```
#include <WiFi.h>
#include <esp_now.h>
// Dirección MAC del actuador
uint8_t actuadorMAC[] = {0xCC, 0x8D, 0xA2, 0x88, 0xB6, 0xB2}; //ANTERIOR
//uint8_t actuadorMAC[] = {0x80, 0x65, 0x99, 0xFB, 0x0D, 0xFA};

// Estructura para recibir datos de los sensores
typedef struct struct_message {
  int id;
  float x, y, z;
} struct_message;

struct_message myData;
String command = "";

// Callback para recibir datos de sensores
void OnDataRecv(const esp_now_recv_info *recv_info, const uint8_t *data, int
    len) {
  memcpy(&myData, data, sizeof(myData));

   //Mostrar datos recibidos de sensores anterior
  //String datosFormateados = "ID: " + String(myData.id) +
   //                          ", X (m/s²): " + String(myData.x, 2) +
    //                        ", Y (m/s²): " + String(myData.y, 2) +
    //                         ", Z (m/s²): " + String(myData.z, 2);

   String datosFormateados = String(myData.id) + "," +
                          "X: " + String(myData.x, 8) + "," +
                          "Y: " + String(myData.y, 8) + "," +
                          "Z: " + String(myData.z, 8);

  Serial.println(datosFormateados);
}

// Callback para confirmar envío al actuador
void onSentConfirmation(const uint8_t *macAddr, esp_now_send_status_t status
    ) {
  Serial.print("Estado de envío al actuador: ");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Éxito" : "Fallo");
}

void setup() {
  Serial.begin(115200);

  // Configurar WiFi en modo estación
  WiFi.mode(WIFI_STA);

  // Inicializar ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error inicializando ESP-NOW");
```

```
    return;
  }

  // Registrar callback para recepción de datos
  esp_now_register_recv_cb(OnDataRecv);

  // Configurar el peer del actuador
  esp_now_peer_info_t peerInfo;
  memcpy(peerInfo.peer_addr, actuadorMAC, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Error al agregar el actuador como peer");
    return;
  }
  // Registrar callback para confirmar envío
  esp_now_register_send_cb(onSentConfirmation);

  Serial.println("Sistema listo. Esperando datos de sensores y comandos
      desde la interfaz.");
}

void loop() {
  // Leer comandos desde el puerto serial
  if (Serial.available() > 0) {
    char receivedChar = Serial.read();
    if (receivedChar == '\n') {
      sendCommand(command); // Enviar comando al actuador
      command = "";
    } else {
      command += receivedChar; // Agregar caracteres al comando
    }
  }
}
// Función para enviar comandos al actuador
void sendCommand(String cmd) {
  const char *msg = cmd.c_str();
  esp_err_t result = esp_now_send(actuadorMAC, (uint8_t *)msg, strlen(msg));
  if (result == ESP_OK) {
    Serial.println("Comando enviado al actuador: " + cmd);
  } else {
    Serial.println("Error enviando comando al actuador.");
  }
}
```

A.14. Código para la interfaz de usuario

Listing A.14.1: Código desarrollado en Python

```
1
2  import tkinter as tk
3  from tkinter import ttk
4  import customtkinter as ctk
5  from PIL import Image, ImageTk
6  from datetime import datetime
7  import serial
8  import serial.tools.list_ports
9  import pandas as pd
10 import threading
11 import time
12 from openpyxl import load_workbook
13 from openpyxl.styles import Alignment
14 import queue #LIBRERIA PARA VER DATOS DE SENSORES
15 from openpyxl import Workbook
16
17 ###///////////
18 ###### 1 FUNCIONES PARA CREAR LA VENTANA DE LA INTERFAZ GRAFICA
19
20 # Crear la ventana principal
21 ctk.set_appearance_mode("dark")  # Modo oscuro (opcional) dark
22 ctk.set_default_color_theme("blue")  # Tema azul (puedes cambiarlo a "green"
       o "dark-blue")
23
24 root = ctk.CTk() #crear ventana
25
26 root.geometry("1920x1080")  # Ajusta al tamaño deseado  # Ancho x Alto
27 root.state("zoomed")  # Maximiza la ventana
28
29 # Obtener el tamaño de la ventana
30 ancho_ventana = root.winfo_screenwidth()
31 alto_ventana = root.winfo_screenheight()
32
33 # Cargar la imagen y ajustarla al tamaño de la ventana
34 imagen_fondo = ctk.CTkImage(Image.open("BUAP.jpg"), size=(ancho_ventana,
       alto_ventana))
35
36 # Crear un Label con la imagen
37 label_fondo = ctk.CTkLabel(root, image=imagen_fondo, text="")  # Sin texto
38 label_fondo.place(relwidth=1, relheight=1)  # Ocupa todo el espacio
       disponible
39
40 # Título de la ventana
41 root.title("MONITOREO DE AEROGENERADOR")
42
43
44 #Carga imagen CFE
45 cfe= ctk.CTkImage(light_image=Image.open("CFE.png"), size=(200, 100))
46 label_cfe = ctk.CTkLabel(root, image=cfe, text="")
47 label_cfe.grid(row=7, column=6, padx=1, pady=1)
```

```
48
49 #//////////////////////////////
50
51
52 #/////////////
53 #2 IDENTIFICAR PUERTO SERIAL
54
55 # Función para detectar los dispositivos conectados
56 def detectar_dispositivos_serial():
57     dispositivos = serial.tools.list_ports.comports()
58     dispositivos_detectados = []
59
60     for dispositivo in dispositivos:
61         nombre = dispositivo.description
62         puerto = dispositivo.device
63         dispositivos_detectados.append((nombre,  puerto))
64
65     return dispositivos_detectados
66
67
68 # Función para actualizar la lámpara y conectar al dispositivo
69 def actualizar_lampara():
70     dispositivos = detectar_dispositivos_serial()
71     if dispositivos:  # Si hay dispositivos conectados
72         lampara.config(bg="green")
73         estado_label.configure(text="Dispositivo conectado")
74         nombre_dispositivo.configure(text=f"Dispositivo: {dispositivos
              [0][0]}")  # Muestra el nombre del primer dispositivo
75         global ser
76         ser = conectar_dispositivo(dispositivos[0][1])
77         if ser:
78             # Iniciar un hilo para recibir los datos del dispositivo
79             threading.Thread(target=recibir_datos, args=(ser,), daemon=True)
                  .start()
80     else:  # Si no hay dispositivos conectados
81         lampara.config(bg="red")
82         estado_label.configure(text="No hay dispositivos conectados")
83         nombre_dispositivo.configure(text="")
84         ser = None
85
86 # Puertos seriales (lado izquierdo)
87 label_estado = ctk.CTkLabel(root, text="Estado del Puerto Serial", font=("
     Impact", 20) , bg_color="transparent")#
88 label_estado.grid(row=0, column=0, padx=10, pady=10)
89
90 # Crear la lámpara (label que cambia de color)
91 lampara = tk.Label(root, width=20, height=2, bg="red")
92 lampara.grid(row=3, column=0, padx=10, pady=10)
93
94 # Etiqueta de estado
95 estado_label = ctk.CTkLabel(root, text="No hay dispositivos conectados",
     font=("Arial", 20))
96 estado_label.grid(row=1, column=0, padx=10, pady=10)
97
```

```
98  # Etiqueta para mostrar el nombre del dispositivo
99  nombre_dispositivo = ctk.CTkLabel(root, text="",
100                                font=("Arial", 12), wraplength=200, justify="
                                       left")
101 nombre_dispositivo.grid(row=4, column=0, padx=10, pady=10)
102
103 # Crear un botón para actualizar la lámpara y conectar
104 boton_deteccion = ctk.CTkButton(root, text="Detectar Dispositivos", command=
        actualizar_lampara, width=40, height=40)
105 boton_deteccion.grid(row=2, column=0, padx=1, pady=1)
106 #///////////////
107
108 # 3Función para establecer comunicación serie con el dispositivo
        seleccionado
109 def conectar_dispositivo(puerto):
110     try:
111         # Establecer conexión con el puerto serie
112         ser = serial.Serial(puerto, 115200, timeout=1)  # Cambia la
                velocidad de baudios si es necesario
113         return ser
114     except Exception as e:
115         print(f"Error al conectar al puerto {puerto}: {e}")
116         return None
117 #//////////
118
119 ####### 4 FUNCION PARA VER DATOS DE SENSORES Y MUESTREO PART 1 2
120
121 # Intervalo de muestreo inicial (en ms)
122 intervalo_muestreo = 500  # Intervalo por defecto de 500 ms
123 ultimo_tiempo_muestreo = 0  # Marca de tiempo para el control de muestreo
124
125 # Función para procesar datos desde la cola
126 def procesar_datos_desde_cola():
127     global ultimo_tiempo_muestreo
128     tiempo_actual = time.time() * 1000  # Convertir tiempo a milisegundos
129     if tiempo_actual - ultimo_tiempo_muestreo >= intervalo_muestreo:
130         ultimo_tiempo_muestreo = tiempo_actual
131         if not datos_queue.empty():
132             datos = datos_queue.get()
133             procesar_datos(datos)
134         actualizar_frame_lista()#lista_datos
135     root.after(50, procesar_datos_desde_cola)  # Revisar la cola cada 50 ms
136
137 # Procesar datos recibidos
138 def procesar_datos(datos):
139     try:
140         if ',' in datos:
141             partes = datos.split(',')
142             if len(partes) == 4:
143                 id_sensor = partes[0].strip()
144                 lectura_x = partes[1].split(':')[1].strip()
145                 lectura_y = partes[2].split(':')[1].strip()
146                 lectura_z = partes[3].split(':')[1].strip()
147                 if id_sensor in lecturas_sensores:
```

```
148                     lecturas_sensores[id_sensor] = (lectura_x, lectura_y,
                            lectura_z)
149         else:
150             print(f"Formato de datos incorrecto: {datos}")
151     except Exception as e:
152         print(f"Error al procesar los datos: {e}")
153
154
155 # Etiqueta y campo de entrada para ingresar el intervalo de muestreo
156 etiqueta_muestreo = ctk.CTkLabel(root, text="Agregue intervalo de muestreo (
        ms):",  font=("Arial", 14))
157 etiqueta_muestreo.grid(row=3, column=4, padx=10, pady=5)
158
159 entrada_muestreo = ctk.CTkEntry(root)
160 entrada_muestreo.grid(row=4, column=4, padx=10, pady=5)
161
162 # Botón para aplicar los cambios
163 def aplicar_muestreo():
164     global intervalo_muestreo
165     try:
166         intervalo = int(entrada_muestreo.get())  # Obtener el valor
                ingresado
167         if intervalo > 0:
168             intervalo_muestreo = intervalo
169             print(f"Intervalo de muestreo actualizado a {intervalo} ms")
170         else:
171             print("Por favor ingrese un valor positivo para el intervalo de
                    muestreo.")
172     except ValueError:
173         print("Por favor ingrese un número válido para el intervalo de
                muestreo.")
174
175 boton_aplicar = ctk.CTkButton(root, text="Aplicar cambios", command=
        aplicar_muestreo)
176 boton_aplicar.grid(row=5, column=4, padx=10, pady=10)
177
178 # Recibir datos desde el puerto serial
179 def recibir_datos(ser):
180     while True:
181         if ser.in_waiting > 0:
182             try:
183                 datos = ser.readline().decode('utf-8', errors='ignore').
                        strip()
184                 if datos_queue.qsize() < 10:  # Limitar el tamaño de la cola
                         para evitar desbordamiento
185                     datos_queue.put(datos)
186             except Exception as e:
187                 print(f"Error al procesar los datos: {e}")
188
189 # Llamada inicial para procesar datos
190 root.after(50, procesar_datos_desde_cola)
191 #//////////////////////////////
192
193
```

```
194 #//////////////////
195 #5 FUNCION GUARDAR DATOS EN EXCEL PARA ESTO SE NECESITO DATOS MOSTRADOS EN
        LA INTERFAZ CON MUESTREO Y LAS FUNCIONES DEL TEMPORIZADOR
196
197 # Crear la cola para almacenar lecturas
198 datos_queue = queue.Queue()
199
200 # Diccionario para almacenar lecturas de sensores
201 sensor_ids = ['1', '2', '3', '4', '5', '6']  # IDs de sensores tambien le
        quite el 7
202 lecturas_sensores = {id_sensor: ('--', '--', '--') for id_sensor in
        sensor_ids}
203
204 # Variables globales para el temporizador y Excel
205 running = False
206 counter = 0
207 workbook = None
208 worksheet = None
209 excel_row = 2
210 numero_lectura = 1
211
212 # Función para procesar datos desde la cola
213 def procesar_datos_desde_cola():
214     global excel_row, numero_lectura
215     while not datos_queue.empty():
216         datos = datos_queue.get()
217         procesar_datos(datos)
218     actualizar_frame_lista()#lista_datos
219
220     # Guardar los datos en el archivo Excel
221     if running and workbook:
222         fecha_actual = datetime.now().strftime("%Y-%m-%d")
223         hora_actual = datetime.now().strftime("%H:%M:%S")
224
225         # Escribir datos en una fila
226         worksheet[f"A{excel_row}"] = numero_lectura
227         worksheet[f"B{excel_row}"] = fecha_actual
228         worksheet[f"C{excel_row}"] = hora_actual
229
230         # Agregar lecturas por cada sensor
231         col = 4  # Inicia después de las columnas de número, fecha y hora
232         for id_sensor in sensor_ids:
233             lectura_x, lectura_y, lectura_z = lecturas_sensores.get(
                    id_sensor, ('--', '--', '--'))
234             worksheet.cell(row=excel_row, column=col, value=lectura_x)
235             worksheet.cell(row=excel_row, column=col + 1, value=lectura_y)
236             worksheet.cell(row=excel_row, column=col + 2, value=lectura_z)
237             col += 3
238
239         excel_row += 1
240         numero_lectura += 1
241
242     root.after(intervalo_muestreo, procesar_datos_desde_cola)  # Actualizar
        cada `intervalo_muestreo` ms
```

```
243
244 # Función para actualizar el Listbox
245 def actualizar_frame_lista():
246     # Eliminar etiquetas antiguas dentro del frame
247     for widget in frame_lista.winfo_children():
248         widget.destroy()
249
250     # Agregar las nuevas lecturas de sensores
251     for id_sensor in sensor_ids:
252         lectura_x, lectura_y, lectura_z = lecturas_sensores.get(id_sensor, (
                '--', '--', '--'))
253         texto = f"Sensor {id_sensor}: X = {lectura_x}, Y = {lectura_y}, Z =
                {lectura_z}"
254
255         label = ctk.CTkLabel(frame_lista, text=texto, font=("Arial", 16))#14
256         label.pack(anchor="w", pady=2)  # Alineado a la izquierda con
                espaciado
257 #////
258
259 # Procesar datos recibidos
260 def procesar_datos(datos):
261     try:
262         if ',' in datos:
263             partes = datos.split(',')
264             if len(partes) == 4:
265                 id_sensor = partes[0].strip()
266                 lectura_x = partes[1].split(':')[1].strip()
267                 lectura_y = partes[2].split(':')[1].strip()
268                 lectura_z = partes[3].split(':')[1].strip()
269                 if id_sensor in lecturas_sensores:
270                     lecturas_sensores[id_sensor] = (lectura_x, lectura_y,
                            lectura_z)
271         else:
272             print(f"Formato de datos incorrecto: {datos}")
273     except Exception as e:
274         print(f"Error al procesar los datos: {e}")
275
276 # Función para iniciar el temporizador
277 def iniciar_temporizador():
278     global running, counter, workbook, worksheet, excel_row, numero_lectura
279     if not running:  # Solo iniciar si no está en ejecución
280         try:
281             horas = int(entry_horas.get())
282             minutos = int(entry_minutos.get())
283             segundos = int(entry_segundos.get())
284             counter = horas * 3600 + minutos * 60 + segundos  # Convertir a
                    segundos
285             if counter > 0:
286                 running = True
287
288                 # Crear un nuevo archivo Excel
289                 timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
290                 filename = f"Lecturas_Sensores_{timestamp}.xlsx"
291
```

```
292                 workbook = Workbook()
293                 worksheet = workbook.active
294                 worksheet.title = "Lecturas Sensores"
295 
296                 # Crear encabezados
297                 encabezados = ["N° Lectura", "Fecha", "Hora"]
298                 for id_sensor in sensor_ids:
299                     encabezados += [
300                         f"Sensor {id_sensor} - X",
301                         f"Sensor {id_sensor} - Y",
302                         f"Sensor {id_sensor} - Z",
303                     ]
304 
305                 # Escribir encabezados
306                 for col_num, encabezado in enumerate(encabezados, start=1):
307                     worksheet.cell(row=1, column=col_num, value=encabezado)
308 
309                 excel_row = 2
310                 numero_lectura = 1
311 
312                 print(f"Archivo Excel creado: {filename}")
313 
314                 update_timer()
315         except ValueError:
316             timer_label.config(text="ERROR: Tiempo no válido")
317 
318 # Función para detener el temporizador
319 def detener_temporizador():
320     global running, workbook
321     running = False
322     if workbook:
323         timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
324         filename = f"Lecturas_Sensores_{timestamp}.xlsx"
325         workbook.save(filename)
326         workbook.close()
327         print(f"Archivo Excel guardado como '{filename}'")
328         workbook = None
329 
330 # Función para reiniciar el temporizador
331 def reiniciar_temporizador():
332     global counter, running
333     detener_temporizador()
334     counter = 0
335     timer_label.config(text="00:00:00")
336 
337 # Función para actualizar el temporizador en cuenta regresiva
338 def update_timer():
339     global counter
340     if running:
341         if counter > 0:
342             counter -= 1
343             timer_label.config(text=time_format(counter))
344             root.after(1000, update_timer)
345         else:
```

```
346             detener_temporizador()
347
348 # Función para convertir los segundos a formato hh:mm:ss
349 def time_format(seconds):
350     hrs = seconds // 3600
351     mins = (seconds % 3600) // 60
352     secs = seconds % 60
353     return f"{hrs:02}:{mins:02}:{secs:02}"
354
355 # Etiquetas y campos de entrada del temporizador
356 label_temp = ctk.CTkLabel(root, text="Temporizador", font=("Impact", 20))
357 label_temp.grid(row=0, column=1, padx=80, pady=30)  # Mostrar la etiqueta en
        la ventana
358 timer_label = tk.Label(root, text="00:00:00", font=("Helvetica", 24))  #
    tamaño de la fuente
359 timer_label.grid(row=1, column=1, padx=80, pady=30)
360
361
362 frame_col2 = ctk.CTkFrame(root)
363 frame_col2.grid(row=2, column=1, padx=5, pady=5)
364
365 # Configurar subcolumnas dentro del frame
366 for i in range(2):
367     frame_col2.grid_columnconfigure(i, weight=200)
368
369 # Entradas de horas, minutos y segundos
370 label_horas = ctk.CTkLabel(frame_col2, text="Horas:", font=("Helvetica", 20)
    , bg_color="transparent")
371 label_horas.grid(row=0, column=0, sticky="nsew")
372 entry_horas = ctk.CTkEntry(frame_col2, width=30)
373 entry_horas.grid(row=0, column=1, padx=10)
374
375
376 # Crear una etiqueta con fondo transparente en lugar de usar grid
377 #label_horas = ctk.CTkLabel(root, text="Horas:", font=("Helvetica", 20),
    bg_color="transparent")
378 #label_horas.place(x=200, y=300)  # Posiciona la etiqueta en el lugar
    deseado
379
380 #entry_horas = ctk.CTkEntry(root, width=30)
381 #entry_horas.place(x=300, y=500)  # Posiciona la entrada en el lugar deseado
382 #////////
383
384 label_minutos = ctk.CTkLabel(frame_col2, text="Minutos:", font=("Helvetica",
        20))
385 label_minutos.grid(row=1, column=0, sticky="nsew")
386 entry_minutos = ctk.CTkEntry(frame_col2, width=30)
387 entry_minutos.grid(row=1, column=1, padx=5)
388
389 label_segundos = ctk.CTkLabel(frame_col2, text="Segundos:", font=("Helvetica
    ", 20))
390 label_segundos.grid(row=2, column=0, sticky="nsew")
391 entry_segundos = ctk.CTkEntry(frame_col2, width=30)
392 entry_segundos.grid(row=2, column=1, padx=3)
```

```
393
394
395 button_iniciar = ctk.CTkButton(root, text="INICIAR", command=
        iniciar_temporizador, font=("Helvetica", 12))
396 button_iniciar.grid(row=3, column=1, columnspan=2, pady=10)
397
398 button_finalizar = ctk.CTkButton(root, text="DETENER", command=
        detener_temporizador, font=("Helvetica", 12))
399 button_finalizar.grid(row=4, column=1, columnspan=2, pady=10)
400
401 button_reiniciar = ctk.CTkButton(root, text="REINICIAR", command=
        reiniciar_temporizador, font=("Helvetica", 12))
402 button_reiniciar.grid(row=5, column=1, columnspan=2, pady=10)
403
404
405 #Etiqueta del estado de sensores
406 # Etiqueta de visualización en tiempo real de datos del sensor
407 label_estado = ctk.CTkLabel(root, text="Estado de sensores", font=("Impact",
         20))
408 label_estado.grid(row=0, column=4, padx=20, pady=10)
409
410 # Listbox y muestreo
411 label_estado = ctk.CTkLabel(root, text="Lecturas en m/s2:", font=("Helvetica
        ", 20))
412 label_estado.grid(row=1, column=4, padx=20, pady=10)
413
414 #//
415 # Frame de lista de lecturas de sensores
416 frame_lista = ctk.CTkFrame(root, width=400, height=250)
417 frame_lista.grid(row=2, column=4, columnspan=2, padx=10, pady=10)
418
419 # Crear una lista de labels una sola vez, esto permite que no parpade la
        interfaz en cada actualizacion de lectura
420 labels = []
421
422 # Función para actualizar los datos sin parpadeo
423 def actualizar_frame_lista():
424     # Actualizar solo el texto de cada etiqueta
425     for i, id_sensor in enumerate(sensor_ids):
426         lectura_x, lectura_y, lectura_z = lecturas_sensores.get(id_sensor, (
            '--', '--', '--'))
427         texto = f"Sensor {id_sensor}: X = {lectura_x}, Y = {lectura_y}, Z =
            {lectura_z}"
428
429         # Actualizamos el texto de la etiqueta correspondiente
430         labels[i].configure(text=texto)
431
432 # Inicializar los labels solo una vez
433 def inicializar_labels():
434     for i, id_sensor in enumerate(sensor_ids):
435         lectura_x, lectura_y, lectura_z = lecturas_sensores.get(id_sensor, (
            '--', '--', '--'))
436         texto = f"Sensor {id_sensor}: X = {lectura_x}, Y = {lectura_y}, Z =
            {lectura_z}"
```

```
437
438         label = ctk.CTkLabel(frame_lista, text=texto, font=("Arial", 14))
439         label.pack(anchor="w", pady=2)  # Alineado a la izquierda con
                espaciado
440         labels.append(label)
441
442 # Llamar a la función de inicialización al inicio
443 inicializar_labels()
444
445 #///////
446
447 etiqueta_muestreo = ctk.CTkLabel(root, text="Agregue intervalo de muestreo (
        ms):",  font=("Arial", 14))
448 etiqueta_muestreo.grid(row=3, column=4, padx=10, pady=5)
449
450 entrada_muestreo = ctk.CTkEntry(root)
451 entrada_muestreo.grid(row=4, column=4, padx=10, pady=5)
452
453 boton_aplicar = ctk.CTkButton(root, text="Aplicar cambios", command=
        aplicar_muestreo)
454 boton_aplicar.grid(row=5, column=4, padx=10, pady=10)
455
456 # Intervalo de muestreo inicial (en ms)
457 intervalo_muestreo = 500  # Intervalo por defecto de 500 ms
458 root.after(intervalo_muestreo, procesar_datos_desde_cola)  # Llamada inicial
        para el primer muestreo
459
460 #////////////
461 ##6 ENVIAR COMANDOS PARA ACTUADOR
462 # Función para enviar comandos al dispositivo
463 def enviar_comando(comando):
464     global ser
465     try:
466         if ser and ser.is_open:  # Verifica si el puerto serie está abierto
467             ser.write((comando + '\n').encode())  # Incluye '\n' al final
                    del comando
468             print(f"Comando enviado: {comando}")
469         else:
470             print("No hay conexión con el dispositivo.")
471     except Exception as e:
472         print(f"Error al enviar el comando: {e}")
473
474 # Crear un botón para enviar el comando ON1
475 boton_on1 = ctk.CTkButton(
476     root,
477     text="ON1 ADELANTE",
478     width=150,
479     height=40,
480     fg_color="blue"
481     #activebackground="blue",
482 )
483
484 boton_on1.bind("<ButtonPress>", lambda event: enviar_comando("ON1"))  #
        Enviar ON1 al presionar
```

```
485 boton_on1.bind("<ButtonRelease>", lambda event: enviar_comando("OFF"))  #
        Enviar OFF al soltar
486 boton_on1.grid(row=2, column=6, padx=10, pady=10)
487
488 # Crear un botón para enviar el comando ON2
489 boton_on2 =ctk.CTkButton(
490     root,
491     text="ON2 ATRAS",
492     width=150,
493     height=40,
494     fg_color="green",
495     #activebackground="green",
496 )
497
498 boton_on2.bind("<ButtonPress>", lambda event: enviar_comando("ON2"))  #
        Enviar ON2 al presionar
499 boton_on2.bind("<ButtonRelease>", lambda event: enviar_comando("OFF"))  #
        Enviar OFF al soltar
500 boton_on2.grid(row=3, column=6, padx=10, pady=10)
501
502 # Etiqueta de BOTONES DE FRENO ACTUADOR LINEAL
503 label_estado = ctk.CTkLabel(root, text="FRENO", font=("Impact", 18))
504 label_estado.grid(row=1, column=6, padx=20, pady=10)
505
506 #///////////////
507 #7 FECHA
508 # Función para actualizar la etiqueta de fecha y hora en tiempo real
509 def update_datetime():
510     current_time = datetime.now().strftime("Fecha: %d-%m-%Y \nHora: %H:%M:%S
            ")
511     datetime_label.configure(text=f"{current_time}")
512     root.after(1000, update_datetime)  # Actualizar cada segundo
513
514 #ETIQUETAS
515 # Crear la etiqueta para mostrar la fecha y hora
516 datetime_label = ctk.CTkLabel(root, text="", font=("Impact", 20))
517 datetime_label.grid(row=0, column=6, padx=30, pady=30)
518
519 # Iniciar la actualización de fecha y hora
520 update_datetime()
521 #///////////////
522
523 root.mainloop() #ESTE COMANDO PERMITE ABRIR LA VENTANA DE LA INTERFAZ
```

A.15. Código transmisor ESP32 S2 Mini para motor reductor

Listing A.15.1: Código transmisor ESP32 S2 Mini

```
1
2  #include <WiFi.h>
3  #include <esp_now.h>
4
5  // Dirección MAC del receptor (¡YA ACTUALIZADA!)
6  uint8_t direccionMAC_Receptor[] = {0x08, 0xD1, 0xF9, 0xE8, 0x62, 0x50};
7
8  typedef struct ComandoMotor {
9    char texto[32];
10 } ComandoMotor;
11
12 ComandoMotor comando;
13
14 void setup() {
15   Serial.begin(9600);
16   WiFi.mode(WIFI_STA);
17   WiFi.disconnect();
18
19   if (esp_now_init() != ESP_OK) {
20     Serial.println("Error al iniciar ESP-NOW");
21     return;
22   }
23
24   esp_now_peer_info_t peerInfo = {};
25   memcpy(peerInfo.peer_addr, direccionMAC_Receptor, 6);
26   peerInfo.channel = 0;
27   peerInfo.encrypt = false;
28
29   if (!esp_now_add_peer(&peerInfo)) {
30     Serial.println("Receptor emparejado correctamente");
31   } else {
32     Serial.println("Error al emparejar receptor");
33   }
34
35   Serial.println("Listo para enviar comandos:");
36   Serial.println("Ejemplos:");
37   Serial.println("  A:150 → Sentido 1");
38   Serial.println("  B:100 → Sentido 2");
39   Serial.println("  P →     Parar suavemente");
40   Serial.println("  F:8000 → Cambiar frecuencia PWM");
41 }
42
43 void loop() {
44   if (Serial.available()) {
45     String texto = Serial.readStringUntil('\n');
46     texto.trim();
47
48     if (texto.length() > 0 && texto.length() < sizeof(comando.texto)) {
```

```
49        texto.toCharArray(comando.texto, sizeof(comando.texto));
50        esp_now_send(direccionMAC_Receptor, (uint8_t*)&comando, sizeof(comando
              ));
51        Serial.println("Comando enviado: " + texto);
52      } else {
53        Serial.println("Error: comando vacío o demasiado largo");
54      }
55    }
56  }
```

A.16. Código receptor ESP32 para motor reductor

Listing A.16.1: Código receptor ESP32

```
1
2  // RECEPTOR ESP32 ARRANQUE, PARADO Y CAMBIO DE SENTIDO SUAVE (loop)
3  #include <esp_now.h>
4  #include <WiFi.h>
5
6  // Pines del puente H
7  #define A1 16    // Dirección GPIO RX2
8  #define A2 17    // Dirección GPIO TX2
9  #define PA 18    // PWM (velocidad) GPIO D18
10
11 const int canalPWM = 0;
12 int frecuencia = 6000;
13 const int resolucion = 8;
14
15 int velocidad = 0;
16 String direccionActual = "";  // "A", "B" o vacío
17
18 typedef struct ComandoMotor {
19   char texto[32];
20 } ComandoMotor;
21
22 ComandoMotor comando;
23
24 // Función para rampas suaves
25 void rampaPWM(int desde, int hasta, int paso = 5, int retardo = 10) {
26   if (desde < hasta) {
27     for (int i = desde; i <= hasta; i += paso) {
28       ledcWrite(canalPWM, i);
29       delay(retardo);
30     }
31   } else {
32     for (int i = desde; i >= hasta; i -= paso) {
33       ledcWrite(canalPWM, i);
34       delay(retardo);
35     }
36   }
37 }
38
39 // Callback cuando se recibe un dato
40 void onReceive(const uint8_t *mac, const uint8_t *incomingData, int len) {
41   memcpy(&comando, incomingData, sizeof(comando));
42   String texto = String(comando.texto);
43   texto.trim();
44
45   if (texto.startsWith("A:")) {
46     int nuevaVelocidad = texto.substring(2).toInt();
47
48     if (direccionActual == "B") {
49       rampaPWM(velocidad, 0);
50       digitalWrite(A1, HIGH);
```

```
51        digitalWrite(A2, LOW);
52        rampaPWM(0, nuevaVelocidad);
53      } else {
54        digitalWrite(A1, HIGH);
55        digitalWrite(A2, LOW);
56        rampaPWM(velocidad, nuevaVelocidad);
57      }
58
59      direccionActual = "A";
60      velocidad = nuevaVelocidad;
61      Serial.println("Giro A a velocidad " + String(velocidad));
62    }
63
64    else if (texto.startsWith("B:")) {
65      int nuevaVelocidad = texto.substring(2).toInt();
66
67      if (direccionActual == "A") {
68        rampaPWM(velocidad, 0);
69        digitalWrite(A1, LOW);
70        digitalWrite(A2, HIGH);
71        rampaPWM(0, nuevaVelocidad);
72      } else {
73        digitalWrite(A1, LOW);
74        digitalWrite(A2, HIGH);
75        rampaPWM(velocidad, nuevaVelocidad);
76      }
77
78      direccionActual = "B";
79      velocidad = nuevaVelocidad;
80      Serial.println("Giro B a velocidad " + String(velocidad));
81    }
82
83    else if (texto == "P") {
84      rampaPWM(velocidad, 0);
85      digitalWrite(A1, LOW);
86      digitalWrite(A2, LOW);
87      velocidad = 0;
88      direccionActual = "";
89      Serial.println("Motor detenido suavemente");
90    }
91
92    else if (texto.startsWith("F:")) {
93      frecuencia = texto.substring(2).toInt();
94      ledcSetup(canalPWM, frecuencia, resolucion);
95      ledcAttachPin(PA, canalPWM);
96      ledcWrite(canalPWM, velocidad);
97      Serial.println("Frecuencia PWM actualizada a " + String(frecuencia));
98    }
99
100   else {
101     Serial.println("Comando no reconocido: " + texto);
102   }
103 }
104
```

```
105 void setup() {
106   Serial.begin(9600);
107   pinMode(A1, OUTPUT);
108   pinMode(A2, OUTPUT);
109   pinMode(PA, OUTPUT);
110 
111   digitalWrite(A1, LOW);
112   digitalWrite(A2, LOW);
113 
114   ledcSetup(canalPWM, frecuencia, resolucion);
115   ledcAttachPin(PA, canalPWM);
116   ledcWrite(canalPWM, 0);
117 
118   WiFi.mode(WIFI_STA);
119   if (esp_now_init() != ESP_OK) {
120     Serial.println("Error al iniciar ESP-NOW");
121     return;
122   }
123   esp_now_register_recv_cb(onReceive);
124   Serial.println("Receptor listo para recibir comandos");
125 }
126 
127 void loop() {
128   // No es necesario hacer nada aquí porque todo se maneja con ESP-NOW
129 }
```

A.17. Código para la interfaz de usuario del motor reductor

Listing A.17.1: Código interfaz motor reductor en Python

```
1
2  import tkinter as tk
3  from tkinter import ttk, messagebox
4  import serial
5  import serial.tools.list_ports
6
7  class MotorControlApp:
8      def __init__(self, root):
9          self.root = root
10         self.root.title("Control de Motor (RPM)")
11         self.serial_port = None
12         self.rpm_maxima = 45  # RPM máxima del motor
13
14         self.direccion = tk.StringVar(value="A")  # A: Avanzar, B:
               Retroceder
15
16         self.create_widgets()
17
18     def create_widgets(self):
19         frame = ttk.Frame(self.root, padding=10)
20         frame.grid(row=0, column=0)
21
22         # Puerto COM
23         ttk.Label(frame, text="Puerto COM:").grid(row=0, column=0)
24         self.combo_ports = ttk.Combobox(frame, values=self.get_serial_ports
               (), width=10)
25         self.combo_ports.grid(row=0, column=1)
26
27         self.btn_conectar = ttk.Button(frame, text="Conectar", command=self.
               conectar_serial)
28         self.btn_conectar.grid(row=0, column=2)
29
30         self.lbl_estado = ttk.Label(frame, text="Desconectado", foreground="
               red")
31         self.lbl_estado.grid(row=0, column=3, padx=10)
32
33         # Dirección
34         ttk.Label(frame, text="Dirección:").grid(row=1, column=0)
35         radio_frame = ttk.Frame(frame)
36         radio_frame.grid(row=1, column=1, columnspan=2, sticky="w")
37         ttk.Radiobutton(radio_frame, text="Avanzar", variable=self.direccion
               , value="A").pack(side="left", padx=5)
38         ttk.Radiobutton(radio_frame, text="Retroceder", variable=self.
               direccion, value="B").pack(side="left", padx=5)
39
40         # Slider RPM
41         ttk.Label(frame, text="Velocidad (RPM):").grid(row=2, column=0)
```

```
42          self.slider_rpm = ttk.Scale(frame, from_=0, to=self.rpm_maxima,
                orient="horizontal", command=self.actualizar_label_rpm)
43          self.slider_rpm.set(0)
44          self.slider_rpm.grid(row=2, column=1, columnspan=2, sticky="we")
45
46          self.lbl_rpm_valor = ttk.Label(frame, text="0 RPM")
47          self.lbl_rpm_valor.grid(row=2, column=3)
48
49          # Frecuencia PWM
50          ttk.Label(frame, text="Frecuencia PWM:").grid(row=3, column=0)
51          self.entry_frecuencia = ttk.Entry(frame)
52          self.entry_frecuencia.insert(0, "6000")
53          self.entry_frecuencia.grid(row=3, column=1)
54
55          self.btn_frecuencia = ttk.Button(frame, text="Actualizar Frecuencia"
                , command=self.enviar_frecuencia)
56          self.btn_frecuencia.grid(row=3, column=2, columnspan=2)
57
58          # Botones manuales
59          self.btn_avanzar = ttk.Button(frame, text="Avanzar", command=self.
                enviar_avanzar)
60          self.btn_avanzar.grid(row=4, column=0, pady=5)
61
62          self.btn_retroceder = ttk.Button(frame, text="Retroceder", command=
                self.enviar_retroceder)
63          self.btn_retroceder.grid(row=4, column=1, pady=5)
64
65          self.btn_parar = ttk.Button(frame, text="Parar", command=self.
                enviar_parar)
66          self.btn_parar.grid(row=4, column=2, pady=5)
67
68          # Log
69          self.txt_log = tk.Text(frame, height=10, width=60)
70          self.txt_log.grid(row=5, column=0, columnspan=4, pady=10)
71
72          # Indicador actual
73          self.lbl_rpm_actual = ttk.Label(frame, text="Velocidad actual: 0 RPM
                ", font=("Arial", 10, "bold"))
74          self.lbl_rpm_actual.grid(row=6, column=0, columnspan=4)
75
76      def get_serial_ports(self):
77          ports = serial.tools.list_ports.comports()
78          return [port.device for port in ports]
79
80      def conectar_serial(self):
81          try:
82              puerto = self.combo_ports.get()
83              self.serial_port = serial.Serial(puerto, 9600, timeout=1)
84              self.lbl_estado.config(text="Conectado", foreground="green")
85              self.txt_log.insert(tk.END, f"Conectado a {puerto}\n")
86          except Exception as e:
87              messagebox.showerror("Error", f"No se pudo conectar: {e}")
88
89      def calcular_pwm_desde_rpm(self, rpm):
```

```
90          return int((rpm / self.rpm_maxima) * 255)
91
92      def actualizar_label_rpm(self, val):
93          rpm = int(float(val))
94          self.lbl_rpm_valor.config(text=f"{rpm} RPM")
95
96          pwm = self.calcular_pwm_desde_rpm(rpm)
97          direccion = self.direccion.get()
98          self.enviar_comando(f"{direccion}:{pwm}")
99          self.lbl_rpm_actual.config(text=f"Velocidad actual: {rpm} RPM")
100
101     def enviar_comando(self, comando):
102         if self.serial_port and self.serial_port.is_open:
103             self.serial_port.write((comando + "\n").encode())
104             self.txt_log.insert(tk.END, f"Enviado: {comando}\n")
105             self.txt_log.see(tk.END)
106         else:
107             messagebox.showwarning("Atención", "Puerto serial no conectado."
                )
108
109     def enviar_avanzar(self):
110         rpm = int(self.slider_rpm.get())
111         pwm = self.calcular_pwm_desde_rpm(rpm)
112         self.enviar_comando(f"A:{pwm}")
113         self.lbl_rpm_actual.config(text=f"Velocidad actual: {rpm} RPM")
114
115     def enviar_retroceder(self):
116         rpm = int(self.slider_rpm.get())
117         pwm = self.calcular_pwm_desde_rpm(rpm)
118         self.enviar_comando(f"B:{pwm}")
119         self.lbl_rpm_actual.config(text=f"Velocidad actual: {rpm} RPM")
120
121     def enviar_parar(self):
122         self.enviar_comando("P")
123         self.lbl_rpm_actual.config(text="Velocidad actual: 0 RPM")
124
125     def enviar_frecuencia(self):
126         try:
127             freq = int(self.entry_frecuencia.get())
128             self.enviar_comando(f"F:{freq}")
129         except ValueError:
130             messagebox.showerror("Error", "Frecuencia inválida")
131
132 # Ejecutar app
133 if __name__ == "__main__":
134     root = tk.Tk()
135     app = MotorControlApp(root)
136     root.mainloop()
```

Referencias

Acciona. (s.f.). *Aerogeneradores.* https://www.acciona.com/es/energias-renovables/energia-eolica/aerogeneradores. (Acciona Energías Renovables. Recuperado el 24 de agosto de 2025)

Alma de herrero. (2010, 2 de agosto). *Parque eólico marino alpha ventus.* Descargado de https://almadeherrero.blogspot.com/2010/08/parque-eolico-marino-alpha-ventus.html (Entrada de blog)

Angarita Jaimes, E. X., y cols. (2018). Control de ángulo de calaje para una turbina eólica.

Aquino González, G. (2020). *Desarrollo, validación numérica y experimental de un nuevo método de detección de daños en aerogeneradores offshore* (Tesis de maestría, Benemérita Universidad Autónoma de Puebla). Descargado de https://scholar.google.com/citations?view_op=view_citation&hl=es&user=6K92UOYAAAAJ&citation_for_view=6K92UOYAAAAJ:u-x6o8ySG0sC (Repositorio BUAP)

Artigao, E., Martín-Martínez, S., Honrubia-Escribano, A., y Gómez-Lázaro, E. (2018). Wind turbine reliability: A comprehensive review towards effective condition monitoring development. *Applied energy*, *228*, 1569–1583.

Asociación Empresarial Eólica (AEE). (2022). *Preguntas frecuentes sobre la eólica marina en España.* Descargado de https://aeeolica.org/wp-content/uploads/2022/03/2202-FAQ-EOLICA-MARINA-2022-v5.pdf (Obtenido de Asociación Empresarial Eólica)

Bahamonde García, M. I., y cols. (2019). *Estudio del potencial eólico en mar abierto y optimización de la producción energética en la implantación de parques eólicos marinos* (Tesis Doctoral no publicada). Universidad de Huelva.

Beer, F. P., Johnston, E. R., DeWolf, J. T., y Mazurek, D. F. (2010). *Mecánica de materiales* (X ed.). McGraw-Hill.

Campoverde-Vilela, L., Feijóo, M. d. C., Vidal, Y., Sampietro, J., y Tutivén, C. (2023). Anomaly-based fault detection in wind turbine main bearings. *Wind Energy Science Discussions*, *2023*, 1–29.

Carletti, E. J. (2007). Comunicación-bus i2c. *Robots Argentina*.

Cathwell. (s.f.). *Wind turbine design.* Descargado 2025-05-12, de https://cathwell.com/tech-categories/need-design/wind-turbines/wind-turbine-design/

Centro de Ciencias de la Atmósfera, U. (s.f.). *¿cómo obtengo la dirección física (mac address) de mi computadora?* Descargado de https://www.ccg.unam.mx/uati/faqs/como-obtengo-la-direccion-fisica-mac-address-de-mi-computadora/ (Re-

cuperado el 21 de marzo de 2025)
Chen, B., Xie, L., Li, Y., y Gao, B. (2020). Acoustical damage detection of wind turbine yaw system using bayesian network. *Renewable Energy*, *160*, 1364–1372.
Chen, J., y Huang, S. (2023). Analysis and comparison of uart, spi and i2c. En *2023 ieee 2nd international conference on electrical engineering, big data and algorithms (eebda)* (pp. 272–276).
Corporación de Desarrollo Tecnológico. (2023). *¿fija o flotante? la eólica marina podría ser el futuro de la energía renovable.* Descargado de https://www.cdt.cl/fija-o-flotante-la-eolica-marina-podria-ser-el-futuro-de-la-energia-renovable/ (CDT)
Dao, P. B. (2022). Condition monitoring and fault diagnosis of wind turbines based on structural break detection in scada data. *Renewable Energy*, *185*, 641–654.
Elton, C., y Euronews. (2024). *Récord mundial: el aerogenerador chino que ha conseguido generar energía para cubrir la demanda de 170 000 hogares.* Euronews. Descargado de https://es.euronews.com/green/2024/06/16/record-mundial-un-aerogenerador-genera-en-un-dia-energia-suficiente-para-abastecer-a-17000 (Recuperado el 18 de junio de 2025)
Energetica21. (s.f.). *Documento sobre energía.* Descargado 2024-11-12, de https://www.energetica21.com/descargar.php?seccion=articulos&archivo=EaU2HOQk7JXDtndAERgktXuq4I7iLRmmldt1uS1qcMtL1LL5csdLFN.pdf (Recuperado de Energetica21)
EstudioSmarter. (2024, 5 de septiembre). *Caracterización de sensores: Ejemplos & tipos.* Descargado de https://www.studysmarter.es/resumenes/ingenieria/ingenieria-electrica/caracterizacion-de-sensores/ (Recuperado de StudySmarter)
Igartua San Segundo, Á., y cols. (2021). Diseño y cálculo de una plataforma flotante para un aerogenerador marino de 15 mw.
Layton, J. (2006, January). *How wind power works.* https://science.howstuffworks.com/environmental/green-science/wind-power.htm. HowStuffWorks.com. (Accessed: 2025-08-19)
Manríquez Herrejón, E. (2020). *Acelerómetros y vibrómetros.* PDF. Descargado de https://www.academia.edu/download/62849278/Acelerometros_y_vibrometros_-_Manriquez_Herrejon_Edgar20200406-54813-o62exd.pdf (Recuperado de Academia.edu)
Martínez Fernández, Á. (2021). *Red de sensores para la gestión de aparcamiento en vía urbana* (Tesis Doctoral no publicada). Universitat Politècnica de València.
Mechatronics, N. (s.f.). *Tutorial mpu6050, acelerómetro y giroscopio.* Descargado de https://naylampmechatronics.com/blog/45_tutorial-mpu6050-acelerometro-y-giroscopio.html (Accedido el 20 de marzo de 2025)
Megatrónica. (s.f.). *ESP32-C3 Super Mini – ESP32-C3.* Descargado de https://megatronica.cc/producto/esp32-c3-super-mini-esp32-c3/ (Accedido el 19 de mayo de 2025)
Merino, . (2021). *El mapa de las centrales de energía eólica.* https://elordenmundial.com/mapas-y-graficos/mapa-centrales-energia-eolica/. (El Orden Mundial. Recuperado el 17 de junio de 2025)

Montes Fonseca, F. (2019). *Acelerómetros triaxiales: análisis de señales y aplicaciones físicas* (Tesis doctoral). Universidad Autónoma de Chihuahua.

National Instruments. (2025). *Medir vibración con acelerómetros.* Sitio web. Descargado de https://www.ni.com/es/shop/data-acquisition/sensor-fundamentals/measuring-vibration-with-accelerometers.html (Recuperado de NI.com)

NXP Semiconductors. (2021). Um10204: I²c-bus specification and user manual (Rev. 7 ed.) [Manual de software informático]. Descargado de https://www.nxp.com/docs/en/user-guide/UM10204.pdf

Pini, A. (2019, febrero). *Por qué y cómo usar la interfaz periférica en serie (spi) para simplificar las conexiones entre múltiples dispositivos.* Descargado de https://www.digikey.com.mx/es/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices (DigiKey Electronics)

Plodpradit, P., Dinh, V. N., y Kim, K.-D. (2019). Tripod-supported offshore wind turbines: modal and coupled analysis and a parametric study using x-sea and fast. *Journal of Marine Science and Engineering*, *7*(6), 181.

Power Technology. (2014). *Borkum west ii offshore wind farm.* https://www.power-technology.com/projects/borkum-farm/. (Recuperado de Power Technology website)

Pozo, F., Vidal, Y., y Salgado, Ó. (2018). Wind turbine condition monitoring strategy through multiway pca and multivariate inference. *Energies*, *11*(4), 749.

Rangel, H. (2022). *Análisis tiempo-frecuencia de vibraciones y redes neuronales convolucionales para detectar daño en aerogeneradores de baja potencia* (Tesis de maestría, Universidad Autónoma de Querétaro). Descargado de https://ri-ng.uaq.mx/bitstream/123456789/3672/1/RI006696.pdf (Repositorio Institucional UAQ)

Romero, R., y Juárez, E. (2024). Simulación de un aerogenerador casero. *Revista ELECTRO*, *46*(1), 138–141.

Sanz Gómez, V. (2018). *Learning analytics: Propuesta metodológica y caso de estudio en educación no universitaria* (Trabajo Fin de Máster, Universidad de Valladolid). Descargado de https://uvadoc.uva.es/bitstream/handle/10324/33352/TFM-G959.pdf

UElectronics. (s.f.). *ESP32-S2 mini tarjeta de desarrollo 4MB Flash.* Descargado de https://uelectronics.com/producto/esp32-s2-mini-tarjeta-de-desarrollo-4mb-flash/ (Accedido el 19 de mayo de 2025)

UElectronics. (s.f.). *MPU-9250 IMU de 9DOF 9250.* Descargado de https://uelectronics.com/producto/mpu-9250-imu-de-9dof-9250/ (Accedido el 19 de mayo de 2025)

Velandia-Cardenas, C., Vidal, Y., y Pozo, F. (2024). Wind turbine gearbox early fault detection using mel-frequency cepstral coefficients of vibration data. *Structural Control and Health Monitoring*, *2024*(1), 7733730.

White, F. M. (2011). *Fluid mechanics* (7nd ed.). McGraw-Hill.

WindEurope. (2022). *Offshore wind energy | WindEurope.* WindEurope. Descargado de https://windeurope.org/policy/topics/offshore-wind-energy/

WindEurope. (2024). *Wind energy in Europe: 2023 Statistics and the outlook for 2024-2030.* WindEurope. Descargado de https://windeurope.org/

intelligence-platform/product/wind-energy-in-europe-2023-statistics-and-the-outlook-for-2024-2030/

WindEurope. (2025). *European offshore wind farms map (public).* Descargado 2025-05-07, de https://windeurope.org/intelligence-platform/product/european-offshore-wind-farms-map-public/

Zhang, J., Wang, Q., Zhang, C., y Gao, D. (2019). Natural frequency and sensitivity analysis of tripod foundation for offshore wind turbine: Jianhua zhang & qiyu wang chunwei zhang. En *Mechanics of structures and materials xxiv* (pp. 924–931). CRC Press.