Universidad Nacional de Ingeniería

Facultad de Ingeniería Mecánica

TESIS

Mejora del sistema productivo del cacao basado en el diseño e implementación de una estación climática automática y vehículo aéreo no tripulado

Para obtener el título profesional de Ingeniero Mecatrónico.

Elaborado por

Angel Alfredo Rios Villacís

0000-0003-4192-9652

Asesor

M.Sc. Alcides Guillermo Joo Aguayo

0000-0002-8459-8489

LIMA – PERÚ

2024

Citar/How to cite	Ríos Villacís [1]
Referencia/Reference Estilo/Style: IEEE (2020)	[1] A. Ríos Villacís, "*Mejora del sistema productivo del cacao basado en el diseño e implementación de una estación climática automática y vehículo aéreo no tripulado*" [Tesis]. Lima (Perú): Universidad Nacional de Ingeniería, 2024.

Citar/How to cite	(Apellido, 2022)
Referencia/Reference Estilo/Style: APA (7ma ed.)	Ríos, A. (2024). *Mejora del sistema productivo del cacao basado en el diseño e implementación de una estación climática automática y vehículo aéreo no tripulado*. [Tesis, Universidad Nacional de Ingeniería]. Repositorio institucional Cybertesis UNI.

Dedicatoria

A mis queridos padres, quienes han sido mi mayor apoyo y fuente de inspiración a lo largo de este viaje académico, les dedico mi tesis con profundo agradecimiento por su amor incondicional y constante motivación. Su sacrificio y dedicación han sido el motor que me impulsa a alcanzar mis metas.

A mis hermanos, cuya presencia y aliento han sido un faro en los momentos difíciles y una fuente inagotable de alegría en los momentos de celebración. Esta tesis es también su logro, pues cada uno de ustedes ha sido parte integral de mi camino hacia el éxito.

A mi alma mater y a mis profesores, les dedico este trabajo como un atributo a su invaluable guía y enseñanzas. La universidad ha sido el hogar donde he cultivado no solo conocimientos, sino también valores y habilidades que me han preparado para enfrentar los desafíos del mundo académico y profesional. A ustedes, profesores, les agradezco por su dedicación y por sembrar en mí la semilla del saber. Sin su orientación, este logro no sería posible.

A todos mis familiares y amigos que han sido mi roca y mi inspiración a lo largo de este camino, les dedico esta tesis con gratitud y cariño. Su apoyo y sus palabras de aliento han sido la fuerza que me impulsó a alcanzar ese logro. Gracias por creer en mí y por ser parte de este importante capítulo en mi vida.

Resumen

En esta tesis se aborda los desafíos enfrentados por el cultivo del cacao en el contexto del cambio climático y la prevalencia de la moniliasis, una enfermedad que puede ocasionar pérdidas significativas. En respuesta a estos problemas, se propone una solución integral y económica.

En primer lugar, se diseñó y desarrolló una Estación Climática Automática utilizando un ESP8266 y un Arduino Nano, que recopila datos de 10 sensores. Estos datos son transmitidos a una plataforma IoT creada desde cero en Django, permitiendo a agricultores y especialistas acceder a información en tiempo real para tomar decisiones informadas sobre la adaptabilidad climática del cultivo. La propuesta presenta una alternativa más asequible en comparación con opciones comerciales disponibles.

Adicionalmente, se aborda la amenaza de la moniliasis mediante la construcción de un Vehículo Aéreo No Tripulado basado en una Raspberry Pi. Este dron integra un algoritmo de inteligencia artificial entrenado para reconocer la moniliasis en los frutos de cacao, siendo capaz de funcionar en diversas variedades de plantaciones. El control del dron se realiza a través de una aplicación Android programada en Java, que también presenta el análisis del reconocimiento de la infección. Esta solución brinda a los agricultores la capacidad de analizar la moniliasis sin necesidad de desplazarse por toda la plantación ni depender de equipos especializados, ya que todo está integrado en un dispositivo accesible desde cualquier dispositivo Android. La propuesta no solo es más económica, sino que también ofrece la funcionalidad adicional de reconocimiento, mejorando la eficiencia y la toma de decisiones en la gestión del cultivo de cacao.

Palabras clave --- Cacao, estación climática, IoT, Dron, Inteligencia Artificial

Abstract

This thesis addresses the challenges faced by cocoa cultivation in the context of climate change and the prevalence of moniliasis, a disease that can cause significant losses. In response to these issues, a comprehensive and cost-effective solution is proposed.

Firstly, an Automatic Weather Station was designed and developed using an ESP8266 and an Arduino Nano, collecting data from 10 sensors. These data are transmitted to an IoT platform built from scratch in Django, enabling farmers and specialists to access real-time information for informed decision-making regarding the crop's climate adaptability. The proposal presents a more affordable alternative compared to available commercial options. Additionally, the threat of moniliasis is addressed through the construction of an Unmanned Aerial Vehicle based on a Raspberry Pi. This drone integrates an artificial intelligence algorithm trained to recognize moniliasis in cocoa fruits, capable of functioning in various plantation varieties. Drone control is done through an Android application programmed in Java, which also provides analysis of infection recognition. This solution gives farmers the ability to analyze moniliasis without the need to traverse the entire plantation or rely on specialized equipment, as everything is integrated into a device accessible from any Android device. The proposal is not only more cost-effective but also offers the additional functionality of recognition, improving efficiency and decision-making in cocoa crop management.

Keywords --- Cocoa, weather station, IoT, Drone, Artificial Intelligence

Tabla de Contenido

Lista de Tablas

Lista de Figuras

Lista de Gráficos

Introducción

El cultivo de cacao enfrenta desafíos significativos relacionados con el manejo climático y la identificación temprana de enfermedades, que afectan directamente la productividad y sostenibilidad del cultivo. En respuesta a estos retos, esta tesis presenta el diseño e implementación de una estación climática inteligente y un vehículo aéreo no tripulado (dron) basado en tecnología de Raspberry Pi, con el objetivo de mejorar el sistema productivo del cacao.

El proyecto se enfoca en integrar sensores y microcontroladores para la recolección y análisis de datos climáticos, facilitando la toma de decisiones informadas por parte de los agricultores. Además, el dron, equipado con algoritmos de inteligencia artificial, está diseñado para la detección y monitoreo de la moniliasis del cacao, una enfermedad que afecta gravemente a los cultivos.

La estación climática proporciona datos precisos y en tiempo real, permitiendo una mejor adaptabilidad climática del cultivo de cacao. Por otro lado, el dron ha demostrado ser una herramienta prometedora para la identificación de frutos infectados, contribuyendo a una gestión más eficaz de las enfermedades.

Este trabajo de investigación tiene como objetivo validar que la integración de tecnologías de monitoreo climático y reconocimiento automatizado de enfermedades puede transformar el cultivo de cacao, reduciendo costos de producción y empoderando a los productores con herramientas accesibles y de alta precisión. La implementación de esta tecnología no solo mejora la productividad, sino que también promueve prácticas agrícolas sostenibles y resilientes ante el cambio climático.

En las siguientes secciones de esta tesis, se detallarán los antecedentes y la problemática actual del cultivo del cacao, el desarrollo de la estación climática y el dron, así como los resultados obtenidos de su implementación y validación en el campo.

Capítulo I. Parte Introductoria del trabajo

1.1. Antecedentes Referenciales

La evolución del cultivo del cacao a lo largo del tiempo se enfrentó al desarrollo de plagas y enfermedades, estas limitaciones sanitarias han dado forma a la distribución geográfica de la producción a lo largo de los siglos. El cambio climático actual agrega una restricción adicional a las restricciones de salud de las plantas, haciendo que el futuro del cultivo del cacao sea más incierto. El cambio climático no solo afecta a las zonas donde el cacao se cultiva por razones fisiológicas, particularmente en relación con los cambios en los regímenes hídricos, sino también afecta la distribución de plagas y enfermedades que afectan a este cultivo. Los programas de árboles de cacao para la resistencia sostenible a las limitaciones de salud de las plantas y el cambio climático son, por lo tanto, desafíos particularmente importantes para el cultivo de cacao, con las otras prácticas de manejo de plantaciones.

En el Perú, la agricultura abarca aproximadamente el 30% de su vasto territorio y es un sector de gran prioridad. Debido a la diversidad de ecosistemas en el país, la agricultura se ha desarrollado de manera diferenciada en distintas regiones, lo que dificulta la implementación de soluciones uniformes. Esta alta especificidad agrava la vulnerabilidad del país ante el cambio climático, en comparación con otros sectores.

En la última década el Perú ha escalado al octavo puesto a nivel mundial como productor de cacao, con diversas áreas dedicadas a su cultivo. No obstante, el cambio climático, un desafío global de envergadura, está generando problemas graves en las zonas cacaoteras. Las variaciones climáticas propician plagas y enfermedades que causan pérdidas económicas considerables y amenazan la subsistencia de las comunidades rurales en estas regiones peruanas.

1.1.1. Estado del arte

Para poder entender qué tipos de técnicas se están estudiando y mejorando en la actualidad y para poder solucionar los problemas planteados se revisaron las siguientes publicaciones.

1.1.1.1. Productos y soluciones existentes

En el proyecto denominado "Monitoreo Climático Participativo: una estrategia para el manejo eficiente de cultivos de cacao en Santander", desarrollado por la Fundación Natura Colombia en 2015, se destaca la estrecha colaboración entre investigadores de la Fundación y los agricultores. Este proyecto se estructura en tres fases clave: la primera etapa se enfoca en proporcionar capacitación a los agricultores para que aprendan a manejar de manera adecuada la microestación. La segunda fase se centra en profundizar la comprensión de la relación entre las variables climáticas registradas. La última fase, por su parte, se basa en permitir que los agricultores tomen decisiones informadas para la gestión de sus cultivos, utilizando como referencia los datos climáticos recopilados.

1.1.1.2. Publicaciones científicas / ingenieriles

Publicaciones Internacionales

Huilca Salcedo, J. G., & Sichiqui Velecela, P. F. (2019), en la publicación titulada “Diseño e implementación de un sistema embebido de monitoreo de las variables climáticas para plantaciones de maíz”, presentan un sistema embebido para el monitoreo de las variables climáticas aplicadas al cultivo del maíz, en donde primero hacen un análisis de las variables climáticas que afectan la producción de maíz, y los requerimientos necesarios para su óptimo desarrollo; para el desarrollo del sistema embebido implementaron una red de sensores utilizando comunicación ZigBee; dentro de las variables que se monitorean están la humedad, temperatura y radiación solar; los datos adquiridos de los sensores son enviados al nodo coordinador, esta información se visualiza en una interfaz gráfica que ayuda a los agricultores a tomar decisiones en el cultivo del maíz.

Abad Buri, J. A., & Farez Sigcha, J. P. (2018), en la publicación titulada “Diseño e implementación de un sistema de monitoreo de variables climáticas que afectan al cultivo de café”, en la plantación asoproccsi ubicado en Santa Isabel”, presentan una red de 5 sensores los cuales estarán controlados por un Arduino nano, este enviará los datos hacia un nodo controlador utilizando tecnología zigbee, donde será almacenado en una base de datos realizada en MySQL, y las variables podrán ser visualizadas en un HMI utilizando un servidor local de Apache llamado XAMPP.

Gómez Zeballos, & Sichiqui Velecela. (2007), en la publicación titulada “Diseño e implementación de una estación meteorológica automática”, presenta un sistema de 7 sensores análogos conectados a un microcontrolador PIC, dentro del microcontrolador realizan una transformación de los bits a valores de temperatura, presión, humedad, etc. Estos valores se visualizan en una pantalla de cristal líquido (LCD), también los valores son dirigidos hacia un PC mediante módulos RF en protocolo RS232; los datos son almacenados para su posterior análisis.

Sm, J. (2016), en la publicación titulada “Implementation of Precision Agriculture Using IOT and Raspberry PI Along with LabVIEW Simulation”, se presenta el desarrollo de un método de cultivo utilizando los datos de diferentes sensores, además también incluye un sistema de riego automático. Para la realización de este proyecto utilizaron un Arduino Uno y una Raspberry Pi. La Raspberry conecta con la aplicación Iot de ThinkSpeak, y también hacen uso de un dashboard realizado en LabView.

Bohorquez Vergara, & Enciso Cala. (2019), en la publicación titulada “Diseño y construcción de un anemómetro para estudios de viabilidad de implementación de generadores eólicos en lugares remotos”, presenta el desarrollo de dos anemómetros electrónicos, uno de cazoletas y otro de hélices, los cuales se hicieron mediante impresión 3D de un modelo CAD realizado previamente, dentro de los componentes se utilizó un sensor de efecto hall para medir la velocidad, y un magnetómetro para la dirección del viento, que son alimentados mediante una batería de Litio y un panel solar, todo es controlado mediante Arduino y utilizando el módulo Goblin 2 son enviados a la plataforma en la nube Ubidots para poder visualizar los datos en tiempo real.

Caro Garrido W. (2019), en la publicación titulada “Diseño de un pluviómetro digital con tecnología sigfox de bajo costo, a través de un módulo Arduino”, presenta el diseño de un pluviómetro de cazoletas que además de medir la precipitación, también será capaz de medir la humedad y temperatura del suelo, realizaron el diseño CAD del pluviómetro teniendo en cuenta todas las normas necesarias impuestas en Manual del Sistema Mundial de Observaciones (OMN-NO-544).

Stanley Tan, Neil Leong, Balon Laguna & Angelyn Lao (2016), en la publicación titulado “A framework for measuring infection level on cacao pods” presenta un framework conteniendo algoritmos de machine learning y técnicas de procesamiento de imágenes para la detección de nivel de la enfermedad de Podredumbre parda en el fruto de cacao. Dentro del framework se utilizan imágenes en la región de color RGB que posteriormente se pasan a un algoritmo de clustering llamado K-means el cual secciona el fruto entre la parte infecta y la parte no infectada, después se aplica un algoritmo basado en una support vector machine para determinar el nivel de infección que tiene el fruto estudiado.

Stanley Tan, Neil Leong, Balon Laguna & Angelyn Lao (2018), en la publicación titulada “Automated Tool for Disease Detection and Assessment for Cacao Black Pod Rot”, se presenta una propuesta de aplicación móvil con la que todos los agricultores de cacao puedan medir el nivel de infección de un fruto causado por el hongo de *Phytophthora* spp. Para el estudio se usaron 60 frutos infectados de cacao con dicho hongo, para entrenar el algoritmo el cual consiste en 2 partes la primera clasifica la planta por partes de acuerdo con la semejanza de color, utilizando k-means, posteriormente se utiliza una máquina de soporte vectorial para así poder determinar que la parte clasificada en el paso anterior es infectada o no infectada y de esta manera poder determinar el nivel de infección de la planta.

López García, (2020), en la publicación titulada “Diseño y construcción de un dron basado en Raspberry Pi”, donde se presenta el diseño estructural de un UAV con todos los análisis de resistencia y costos, seguido de la configuración interna del dron, que contiene a la raspberry pi como computador de a bordo y la tarjeta controladora que se utiliza es la Acro Naze 32, para el control de los 4 rotores, todo el sistema electrónico estará alimentado mediante un batería de litio y el dron estará controlado mediante una aplicación móvil Android.

En su obra titulada "DronePi: Construcción de un dron basado en Raspberry Pi," Puertas Gayoso (2016) presenta un proyecto que involucra la concepción y creación de un cuadricóptero dirigido mediante un dispositivo móvil Android. Este innovador dron estará equipado con una cámara capaz de capturar tanto imágenes como videos, los cuales podrán ser transmitidos al terminal móvil en tiempo real. Además, este dispositivo llevará a cabo el procesamiento de imágenes directamente a bordo mediante una Raspberry Pi, que no solo se encargará de esta tarea, sino también de recibir las instrucciones procedentes del dispositivo móvil y transmitirlas a la controladora, que a su vez controlará los motores del cuadricóptero.

Publicaciones Nacionales

Ceccarelli V., Fremout T., Zavaleta D. & Lastra S. (2021), en la publicación titulada “Climate change impacto in cultivated and wild cacao in Perú and the search of climate change tolerant genotypes” presentan una revisión de cómo afectará el cambio climático en contra del cacao, en la cual preveen que habrá una contracción de área para el cacao cultivado en el Perú. Frente a esto recomiendan optar por genotipos tolerantes para facilitar la adaptabilidad climática del cultivo.

Paiva-Peredo E. (2016), en la publicación titulada “Modelado y control de un cuadricóptero” presenta un modelo matemático, la estimación de parámetros físicos, el diseño de controles PID para nuestro cuadricóptero, la validación del modelo matemático de la aerodinámica del cuadricóptero agregando la dinámica de los propulsores.

En su trabajo titulado "Sistema inteligente de detección de patógenos y enfermedades en plantaciones de arroz en el departamento de Lambayeque durante 2019," Galan Zapata J. L. (2019) introduce una aplicación móvil con un sistema inteligente que se respalda en un modelo de reconocimiento de imágenes basado en redes neuronales. El propósito fundamental de este sistema es proporcionar un reconocimiento altamente preciso de los patógenos identificados en los cultivos de arroz.

1.2. Planteamiento del problema

1.2.1. Descripción problemática

La utilización inadecuada y degradación de la base productiva de la actividad económica de la agricultura hace que su explotación no sea muy efectiva. Esta mala utilización se debe al mal monitoreo de las variables climáticas, mal control de plagas y al ineficiente sistema de riego con el que cuentan gran parte de los sectores de cultivo (MINAGRI, 2010).

Esto es generado por el desinterés político, abandonando el agro y de esta forma no ayudando a solucionar estos problemas. Debido a esto es que existe un mal monitoreo de las variables climáticas en la producción de las plantaciones de cacao, además también de una ineficaz detección de plagas y enfermedades (Eduardo Zegarra, 2020).

El cacao se encuentra dentro del grupo de los cultivos más vulnerables a los efectos del cambio climático y a la falta de conocimiento por parte de los agricultores sobre cómo actuar en su proceso de adaptabilidad climática; otras de las razones por las que el cacao se ve vulnerable es el bajo nivel tecnológico y carencia de información, así también como el difícil acceso a fuentes de financiamiento y a sistemas seguros y por último a la escasez de variedades resistentes al estrés climático según informa el MINAGRI, (2020).

Debido al mal manejo de enfermedades en el cultivo de cacao, el hongo de la moniliasis redujo drásticamente la producción de cacao en Costa Rica de unas 10.000 t en 1978 a unas 600 t ahora. La moniliasis ya está en todos los países de América Central. En México, la producción de cacao rondaba las 50.000 t en 2003, justo antes de la aparición de la moniliasis, y ha descendido a menos de 30.000 t desde 2007. En Perú se ha reportado que la moniliasis causada por el hongo *Moniliophthora roreri* es la principal enfermedad del cultivo de cacao y ocasiona pérdidas de hasta el 90% en áreas no manejadas y además causa abandono o sustitución del cultivo (Rios-Ruiz, 2004). Por el momento, la moniliasis no se ha trasladado a Brasil y Guyana, pero la enfermedad está en todos los demás países productores de América Latina. En el Caribe, solo

Jamaica se ve afectada por moniliasis. Otro ejemplo es la producción en el estado de Bahía (Brasil), que disminuyó de 242.000 t en 1990 a 69.000 t en 1995, principalmente por la llegada de la escoba de bruja al estado. La llegada de escoba de bruja habría sido intencional en este caso (Cilas & Bastide, 2020).

Otra de las enfermedades que más pérdidas causa en el cultivo de cacao es causado por el hongo de la *Phytopthora*, que cuando no es tratado correctamente puede causar pérdidas de hasta el 90% de la plantación, esta enfermedad tiene un impacto económico estimado en un costo de hasta 450 mil toneladas métricas por valor de 423 millones de dólares de pérdidas cada año (Bowers et al., 2001), como se visualiza en la **Tabla 1**.

Tabla 1 : Pérdidas de producción de cacao por regiones (Bowers, 2001)

Diseases	Pathogen	Region	Reduced Production	
			(tons x 1000)	($ million)*
Black Pod	*Phytophthora* spp.	Africa/Brazil/ Asia	450	423
Witches' Broom	*Crinipellis perniciosa*	Latin America	250	235
Frosty Pod Rot	*Moniliophthora roreri*	Latin America	30	47
Swollen Shoot	CSSV	Africa	50	28
Vascular-streak dieback	*Oncobasidium theobromae*	Asia	30	28

*January, 2001: value = $940.00/ metric ton.
Source: The World Cocoa Situation, M. Taylor, LMC International Ltd/Trade Discussions

1.2.2. Formulación del problema

¿En qué medida el inadecuado o nulo uso de técnicas de observación, medición y monitoreo de variables climáticas para la adaptabilidad climática y de la mala determinación de enfermedades para su manejo integrado afecta negativamente la producción del cultivo de cacao?

1.3. Objetivo

Objetivo General:

- Mejorar el sistema productivo de cacao basado en el diseño de una estación climática automática y un vehículo aéreo no tripulado.

Objetivos Específicos:

- Diseñar la arquitectura de hardware y software para la toma de datos climáticos y el sistema de vuelo del dron, con el fin de mejorar los procesos de cultivo de cacao y reducir sus costos.
- Diseñar una plataforma IoT en la que se puedan visualizar y almacenar los datos climáticos.
- Diseñar un algoritmo de inteligencia artificial para la detección de la moniliasis en el fruto del cacao.
- Validar los algoritmos y la arquitectura de hardware y software de la estación climática y vehículo aéreo no tripulado en la mejora del sistema productivo del cacao.

1.4. Hipótesis y operacionalización de variables

1.4.1. Hipótesis general y específicas

General

- La estación climática inteligente y el vehículo aéreo no tripulado basado en raspberry pi, serán de ayuda significativa para la mejora en el sistema productivo del cacao.

Específicas

- Los sensores, actuadores y módulos integrados a utilizarse en el diseño serán los óptimos en relación calidad-precio, además de lograr una buena sinergia juntos.
- La plataforma IoT será capaz de mostrar los datos climáticos en tiempo real y establecer una conexión con la base de datos para su almacenamiento.
- El algoritmo de inteligencia artificial diseñado será capaz de reconocer la moniliasis del cacao y notificar al agricultor para su respectiva extracción.
- La validación de los algoritmos y la arquitectura de hardware de la estación climática y vehículo aéreo no tripulado logrará servir de ayuda en la mejora del sistema productivo del cacao.

1.4.2. Operacionalización de variables

La **Tabla 2** contiene la operacionalización de las variables independientes de la toma de datos climáticos y de intervención en el vuelo de un vehículo aéreo no tripulado, mientras que la **Tabla 3** presenta la operacionalización de las variables dependientes para la mejora del sistema productivo del cultivo de cacao.

Tabla 2 : Operacionalización de variable independiente

VARIABLE INDEPENDIENTE	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	INDICADORES
Estación climática automática y vehículo aéreo no tripulado.	Estación climática automática para el monitoreo de variables climáticas y vehículo aéreo no tripulado basado en raspberry pi con algoritmo de reconocimiento de moniliasis	Microcontrolador programado para la toma de datos climáticos de una red de sensores y raspberry pi con algoritmo de control de vuelo y algoritmo de reconocimiento de moniliasis para un vehículo aéreo no tripulado	Voltajes de salida de los sensores para su interpretación en el microcontrolador seleccionado Colores y texturas características presentes en un fruto que presenta enfermedad

Tabla 3 : Operacionalización de variable dependiente

VARIABLE DEPENDIENTE	DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERACIONAL	INDICADORES
Sistema productivo de cacao	Medidas a tomar en cuenta para considerar la adaptabilidad climática del cultivo de cacao y la reducción de moniliasis	Crear una alerta si alguna de las variables climáticas sale del rango óptimo para el cultivo de cacao y la proporción de frutos infectados por moniliasis en un área de cultivo de cacao	Exceso o déficit en alguna unidad climática medida respecto al rango óptimo de operación para el cacao Cantidad de plantas que presentan frutos enfermos por área

Capítulo II. Marco teórico y conceptual

2.1. Bases teóricas

2.1.1. El Cacao

El cacao (*Theobroma cacao* L) es una planta que tiene su origen en los bosques de América del Sur, específicamente en las regiones del Amazonas y Orinoco, donde se encuentra de forma natural. Antes de la llegada de los colonizadores españoles, algunas tribus indígenas de Centro y Sudamérica ya estaban familiarizadas con el cacao y le atribuían una amplia gama de aplicaciones. De hecho, debido a su elevado valor, el cacao llegó a utilizarse como moneda en algunas culturas indígenas, como los Chichimecas, Toltecas y aztecas.

Fue en el año 1735 cuando el naturalista Carl Linneo llevó a cabo su primera clasificación del cacao, otorgándole el nombre de *Theobroma cacao*, que en latín significa "fruto de los dioses". Este nombre ha perdurado hasta la actualidad y refleja la importancia histórica y cultural que se le atribuye al cacao.

Se cree que la dispersión del cacao en las regiones de Centro y Sudamérica estuvo relacionada con el estilo de vida nómada de muchas tribus indígenas de la época. Cuando los colonizadores españoles llegaron a América, descubrieron la diversidad de usos que estas tribus le daban al cacao. Posteriormente, el cacao fue llevado a África, donde se cultivó de manera extensiva, aprovechando la mano de obra de los esclavos. Actualmente, África alberga las mayores plantaciones de cacao en todo el mundo.

Se ha indicado que el área principal de diversidad del cacao podría ubicarse en la región nororiental del Perú. No obstante, la presencia de poblaciones silvestres y autóctonas dispersas en la región central y sur de la Amazonía alta respalda la hipótesis de que el lugar de origen del cacao abarcaría la región centro y sureste del Perú, incluyendo las cuencas de los ríos Huallaga, Ucayali y Urubamba, como menciona García en el 2000.

2.1.2. El Cultivo de Cacao en el mundo

El árbol de cacao es cultivado en las regiones tropicales, y su cultivo comercial se extiende generalmente desde los 15° al norte hasta los 15° al sur de la línea ecuatorial. Aunque ocasionalmente se encuentra en latitudes subtropicales, desde los 23°26’ al norte (límite del Trópico de Cáncer) hasta los

23°26’ al sur (límite del Trópico de Capricornio) de la línea ecuatorial (ver **Figura 1**). La temperatura anual promedio que favorece su desarrollo oscila entre 23° y 30°C, siendo ideal alrededor de los 25°C. Su cultivo abarca desde el nivel del mar hasta altitudes de 1,200 metros sobre el nivel del mar, siendo la altitud óptima de 500 a 800 metros sobre el nivel del mar. Además, requiere una humedad relativa anual promedio que se sitúa entre el 70% y 80%.

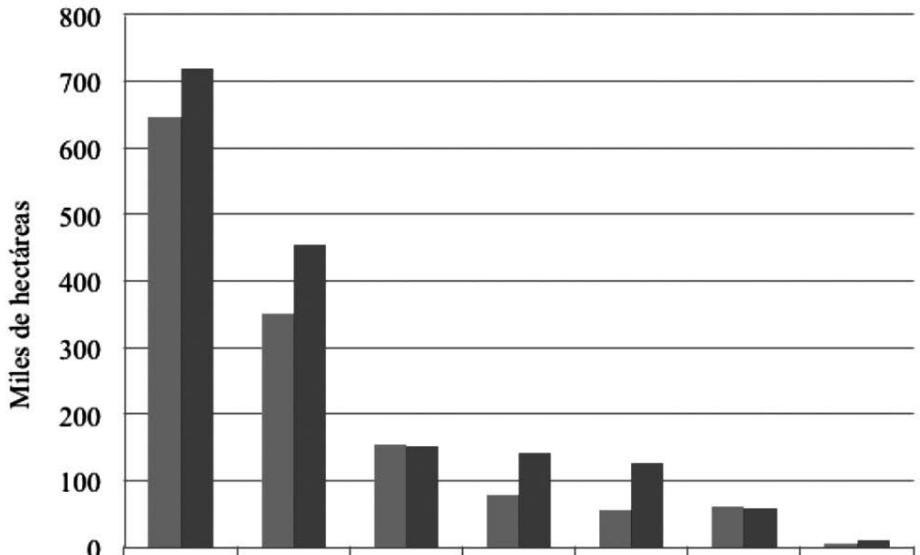
Figura 1 : Ubicación y Distribución de los países productores de Cacao (Batsita, 2009)

El grano del cacao desempeña un papel central en la elaboración de chocolates, y más allá de esto, se aprovecha en la creación de una amplia gama de derivados y subproductos, cada uno de los cuales atiende a mercados específicos. Estos productos incluyen, en primer lugar, el licor de cacao, que se presenta como una forma líquida de chocolate puro y está formulado a partir de una combinación de manteca de cacao y cacao seco. El licor de cacao desempeña un rol fundamental en la fabricación de chocolates y productos de confitería.

El continente africano lidera la producción global de cacao, destacando naciones como Costa de Marfil, Ghana, Nigeria y Camerún, que en conjunto generan aproximadamente 2.752 mil toneladas anuales de este cultivo. Asia se sitúa como el segundo continente más relevante en la producción de cacao, siendo Indonesia el principal país productor con alrededor de 738 mil toneladas. En América Latina y el Caribe (ALC), se ha observado un constante aumento en

la extensión de tierras dedicadas al cultivo de cacao. Este incremento se concentra en cinco naciones: Ecuador, Colombia, Brasil, Perú y República Dominicana, que en conjunto suman aproximadamente 354 mil hectáreas adicionales destinadas a este cultivo. Colombia y Perú, en particular, han experimentado un incremento en la superficie de cultivo de más del 100% en comparación con el año 2006 (Fontagro-BID, 2019) (**Figura 2**)

Figura 2 : Incremento de la superficie cultivada de cacao en los principales países de ALC (Fontagro-BID, 2019)



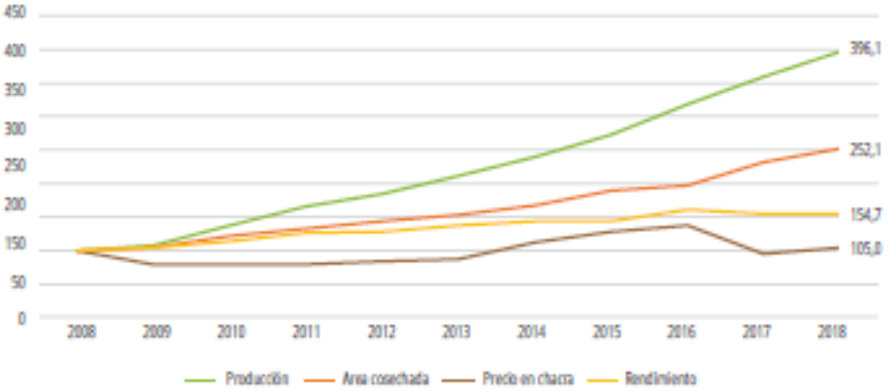
	Brasil	Ecuador	República Dominicana	Colombia	Perú	México	Nicaragua
2006	647,135	350,027	153,219	77,320	56,732	60,866	5,600
2016	720,053	454,257	150,943	141,143	125,580	58,734	9,344

2.1.3. Importancia del cultivo de cacao en el Perú

En Perú, los indicadores clave del sector cacaotero, que incluyen la extensión de áreas cultivadas, la producción, la productividad y la participación de las familias en la producción, han experimentado un crecimiento constante en la última década, llegando a niveles récord. En el año 2018, se alcanzó una producción de 134,000 toneladas de cacao, provenientes de unas 160,000 hectáreas de cultivo, logrando un rendimiento promedio de 851 kg/ha/año (según datos del MINAGRI, 2019a). Durante el período comprendido entre 2008 y 2018, la superficie destinada al cultivo de cacao aumentó significativamente en un 152%, la producción se incrementó en un impresionante 296%, la productividad mejoró en un 55%, y los precios en origen aumentaron un 5% (ver **Figura 3**). En

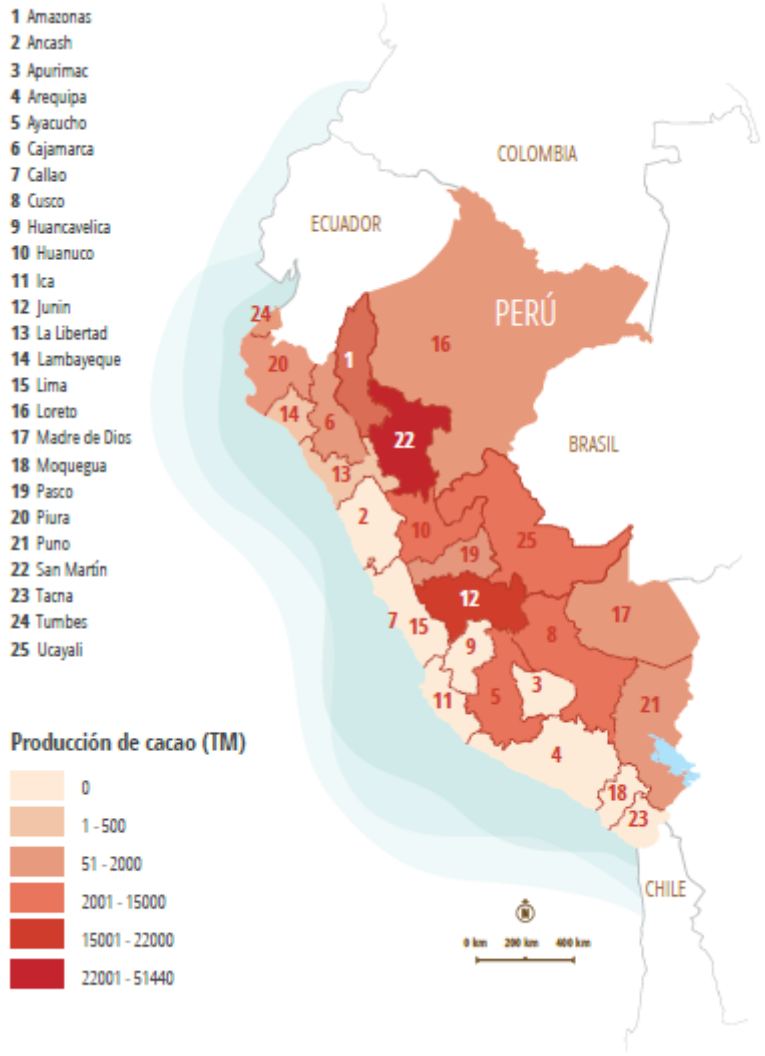
la actualidad, el sector involucra a aproximadamente 90,000 familias de pequeños agricultores, que poseen en promedio parcelas agrícolas de alrededor de 1.9 hectáreas cada una, según datos del MINAGRI (2019b).

Figura 3 : Índices de producción, precios, área y rendimiento de cacao en el Perú (MINAGRI, 2019)



A pesar de que el cacao se cultiva en 16 departamentos y 259 distritos de Perú, aproximadamente el 95% de la producción se concentra en 7 departamentos prominentes: San Martín lidera con un 42.2%, seguido por Junín con un 17.9%, Ucayali con un 10.9%, Huánuco con un 7.3%, Cusco con un 7.2%, Amazonas con un 5.2%, y Ayacucho con un 4.2% (como se muestra en la Figura 5.3). Las regiones con los rendimientos más elevados son Junín, La Libertad, Pasco, Tumbes, Loreto y San Martín, con productividades que oscilan entre 950 y 1,192 kg/ha/año, mientras que la productividad más baja se registra en Cusco, con 412 kg/ha/año (véase la Figura 5), según datos del MINAGRI (2019a).

Figura 4 : Producción de cacao en Perú por departamento (MINAGRI, 2019)



2.1.4. El sistema y producción del cultivo de cacao

El árbol de cacao, cuando crece en la sombra de árboles forestales más altos, tiene la capacidad de alcanzar una altura de hasta 10 metros. Sus frutos, conocidos como mazorcas, tienen dimensiones que varían entre 15 y 25 centímetros de longitud y albergan aproximadamente de 30 a 40 semillas, que después de un proceso de fermentación y secado se convierten en los granos de cacao. Estas mazorcas se desarrollan tanto en el tronco principal como en las ramas de la copa del árbol. El cacaotal comienza a generar cosechas en un período de cuatro a cinco años después de su plantación y puede mantener una producción constante durante muchas décadas (Hardy, 1961).

Actualmente, en condiciones de la selva peruana la siembra del plantón de cacao en terreno definitivo debe coincidir con las épocas de mayor

precipitación. Estas luego son injertadas con el material genético adecuado. Actualmente la mayor área está injertada con el clon CCN-51, un material de alta producción, pero susceptible a las enfermedades y plagas. La siembra se realiza a un distanciamiento de 3m x 3m, en cuadrado formando calles a fin de facilitar la realización de las labores culturales de podas, deshierbos, fertilización y control de plagas y enfermedades. Las cosechas son periódicas y durante casi todo el año.

El diseño de siembra en hileras regulares y el manejo de la plantación de cacao a través de la poda, también facilitaría la implementación del uso de nuevas tecnologías automatizadas de monitoreamiento del clima y de enfermedades, pudiendo ser a través de drones, requiriendo por tanto hacer las pruebas necesarias para determinar la eficiencia de esta.

2.1.5. Condiciones climáticas para el crecimiento del cacao

El cacao es una especie nativa de la floresta tropical húmeda americana, siendo su centro de origen, probablemente, las nacientes de los ríos Amazonas y Orinoco. Este cultivo está, por tanto, adaptada al clima tropical lluviosas, regiones esas de precipitación abundante y temperaturas elevadas. Se cultiva desde el nivel del mar hasta los 1 200 msnm, siendo el óptimo de 500 a 800 msnm (Hardy, 1961).

Así por ejemplo para el Perú, y según el Mapa de zonas de vida de Holdridge del Perú realizado por el SENAMHI, identifica la existencia de 16 zonas de vida principales (biomas) (Aybar-Camacho, et al, 2017). En estas se asocian que las mayores áreas sembradas de cacao pertenecen a los biomas de bosque húmedo, boque muy húmedo y bosque pluvial.

2.1.5.1. Temperatura

El rango de temperatura promedio anual va de 23° a 30° C, siendo el óptimo de 25 °C. (Gómez et al., 2014).

Respecto a este elemento meteorológico, Alvim (1977) informa que el cacao requiere, para el cultivo comercial, temperaturas medias anuales entre 22, 4º y 26,7ºC, tolerando amplitudes térmicas mensuales que van desde 18,8ºC en los meses más fríos hasta 27,9ºC en los más cálidos. Las temperaturas medias mensuales por debajo de los 22 °C inhibirían la formación de botones florales, con efectos negativos en la producción, de cinco a seis meses después.

En el trabajo de Erneholm (1948), citado por Alvim (1977), comparando las temperaturas extremas en las regiones cacaoteras, se concluyó que la temperatura mínima promedio para el cultivo no debe ser menor de 15 °C, y la temperatura mínima absoluta más baja debe ser no estar por debajo de 10 °C. Estos son los límites que se mencionan a menudo en la literatura como los factores que definen los límites de altitud y latitud del cultivo.

Otro aspecto de la temperatura que debe tenerse en cuenta, para Hardy (1961), es el rango diario. Se han encontrado correlaciones positivas entre rangos diarios superiores a 9 °C y excesiva brotación de yemas vegetativas, en detrimento de la producción.

2.1.5.2. Humedad Relativa

El cacao necesita una humedad relativa anual promedio de entre el 70% y el 80%.

Aun considerando que las producciones de cacao provienen de las denominadas regiones con poca variación de elementos meteorológicos, las variaciones diarias de temperatura determinan, en correspondencia, fluctuaciones en la humedad relativa (HR) y, consecuentemente, en el déficit de presión de vapor (DPV) en la atmósfera.

Raja Harun & Hardwick (1987b), trabajando con plántulas de 5 meses en un invernadero en Inglaterra, encontraron que la resistencia estomática obtenida era relativamente alta debido a la baja intensidad de luz utilizada. Sin embargo, pudieron observar que la resistencia de las estomas aumenta, de forma aproximadamente lineal, a medida que el DPV se hace más grande. El cierre de estomas debido a un aumento de la VPP o una reducción de la humedad relativa es una respuesta hidroactiva al estrés hídrico de la epidermis foliar. Los autores concluyen que con el cierre de estomas en respuesta al aumento de DPV, con la consecuente reducción de la tasa fotosintética, en lugares muy soleados y sin sombra, los resultados pueden ser aún más deletéreos con la fotodestrucción de la clorofila.

2.1.5.3. Precipitación

Las cantidades de precipitación pluvial que resultan adecuadas para el cultivo de cacao se sitúan en un rango mínimo de 1,400 mm y un máximo de 3,000 mm. Sin embargo, la precipitación óptima, que contribuye de manera significativa al ciclo del cultivo, se encuentra en el intervalo de 1,500 a 2,500 mm.

Es fundamental destacar que el cacao exhibe una baja tolerancia ante la falta de agua, especialmente en meses donde las precipitaciones son inferiores a 100 mm, lo que puede provocar deficiencias hídricas perjudiciales para la floración y la emergencia de hojas (Gómez et al., 2014).

Alvim (1977) considera las lluvias de la región como uno de los factores más importantes para realizar un cultivo comercial de cacao. Las precipitaciones deben estar entre 1400 y 2500 mm anuales, bien distribuidas a lo largo del año, Augusto (1997) advierte que este intervalo es demasiado amplio, argumentando que el límite superior adecuado rondaría los 1800 mm anuales. Estos valores deben superar las pérdidas anuales por evapotranspiración. Según Hardy (1961), la ausencia de una estación seca bien definida, es decir, con menos de 60 mm de lluvia, de dos a tres meses consecutivos durante el año, se acepta como índice de requerimiento mínimo para el cacao.

Cuando estos valores superan los 2500 a 3000 mm por año, la producción puede reducirse debido al encharcamiento y al desarrollo de enfermedades. Cuando la precipitación anual es menor a 1200 mm, es posible desarrollar cacao con el uso de riego, como, por ejemplo, en el norte de Venezuela, donde la precipitación está entre 700 y 800 mm por año.

Sin embargo, el cacao puede considerarse una planta resistente a la sequía y puede cultivarse en zonas con periodos de escasez de lluvias, siempre que estos periodos coincidan con una estación relativamente fría, cuando el consumo de agua por evapotranspiración es menor.

Augusto (1997), citando el informe anual de ESFIP - Estación Experimental Fillogônio Peixoto perteneciente al Centro de Investigación del Cacao (CEPEC), organismo de investigación del Comité Ejecutivo del Plan de Cultivo de Cacao (CEPLAC), en Linhares, estado de Espírito Santo, de 1996, informa que en 1988 llovió solo 943 mm, con precipitaciones superiores a 5 mm en solo 54 días. Ese año la productividad fue de 201 kg de cacao / ha. En 1992, con una precipitación de 1698 mm y una precipitación superior a 5 mm en 98 días, la productividad alcanzó los 1120 kg de cacao / ha. Sin embargo, para demostrar los efectos de la distribución temporal de las precipitaciones, menciona que en 1995 la precipitación total fue de 1167 mm. Sin embargo, como se produjeron precipitaciones superiores a los 5 mm en solo 53 días, el rendimiento de este año se redujo a 70 kg de cacao/ha, agravando la muerte del

20% de los árboles de cacao. El mismo autor refuerza el énfasis que se le debe dar al binomio (precipitación total) / (distribución a lo largo del año), atribuyendo, aún, los bajísimos niveles de producción en 1995, a la época en que ocurrieron los períodos secos y lluviosos.

2.1.5.4. Vientos

El cultivo del cacao requiere estar libre de vientos fuertes persistentes a lo largo del ciclo productivo: es importante la prevención con árboles forestales como cortina rompe viento.

Los manuales sobre el cultivo del cacao son unánimes al afirmar que la aparición de vientos es extremadamente perjudicial para el cultivo, especialmente para las plantas jóvenes. Para el cacao adulto, sin embargo, este requisito no es tan imperativo, especialmente cuando se encuentra en áreas sombreadas, debido a que las plantas de sombra ya ofrecen protección contra los efectos del viento. Estas afirmaciones son corroboradas por Alvim (1977), quien, en estudios realizados en Bahía, con plantas trasplantadas, verificó un aumento en el área foliar y, en consecuencia, la dinamización de las plantas, luego de un período de severa defoliación, en parcelas con cortavientos y defoliación seguida de la muerte del 72% de las plantas sometidas al viento.

En cuanto a la velocidad del viento, Alvim (1977) informa que, en el caso de Linhares, en el estado de Espírito Santo, la defoliación por el viento es frecuente en lugares donde se cultivan árboles de cacao sin al menos un mínimo de sombra. En el valle de Colatina, también en el estado de Espírito Santo, una región naturalmente protegida por montañas, sin embargo, no se ha recomendado ningún cuidado especial con el viento. Agrega que, si bien en ambas localidades del estado de Espírito Santo la precipitación media es similar (1200 a 1300 mm), mientras que en Colatina la velocidad media del viento es de alrededor de 1 m/s, en Linhares es de 4 m/s.

2.1.5.5. Irradiación solar y Luminosidad

Desde un punto de vista meteorológico, en un día típico de verano en las regiones tropicales, la intensidad de la luz puede alcanzar los 2000 µmol fotones.m-2. s-1, lo que lleva a la conclusión de que el árbol del cacao sería una planta de sombra, corroborado por las estructuras anatómicas y morfológicas de las hojas, glabras (sin pelo) o casi glabras, propias de las plantas tropicales de sombra (Cuatrecasas (1964), citado por Souza & Dias (2001)).

Para Alvim (1977), el árbol del cacao se comporta como una planta de sombra cuando es joven, en los primeros 2 a 3 años. Afirma categóricamente que en muchos lugares sería prácticamente imposible iniciar un cultivo comercial de cacao sin sombra. Sin embargo, con el tiempo, esta necesidad desaparece con el aumento del área foliar del árbol del cacao, debido al auto sombreado. El mismo autor cita varias fuentes que muestran que después de la etapa juvenil, el dosel mismo es capaz de proporcionar sombra a las plantas. A partir de entonces, la productividad aumenta, cuando la nutrición de la planta es satisfactoria, en condiciones sin sombra.

La luminosidad es variable dependiendo del ciclo productivo en el que se encuentre, siendo del 40% al 50% para el cultivo en crecimiento (menor de 4 años) y del 60 al 75% para plantación en producción (mayor de 4 años).

2.1.6. Condiciones de suelos

Desde una perspectiva de sus características físicas, el cacao prospera en suelos de considerable profundidad, que oscilan entre 0,60 y 1,50 metros. Estos suelos pueden tener una textura que varía entre franco, franco-arcilloso y franco arenoso, pero deben evitarse los suelos extremadamente finos o gruesos. Es esencial destacar que el cacao no se adapta bien a suelos arcillosos pesados, puesto que estos tienden a tener una baja capacidad de aireación y drenaje del agua. Los suelos ideales para el cacao exhiben un nivel de porosidad que oscila entre el 20% y el 60%, lo que les permite retener de manera efectiva la humedad. En este sentido, es crucial garantizar un drenaje adecuado en el suelo. Además, se debe asegurar que el nivel freático se encuentre a una profundidad superior a 1,5 metros, mientras que la topografía óptima para el cultivo de cacao comprende terrenos planos o ligeramente ondulados, con una pendiente que no exceda el 25%.

2.1.7. Planeación y gestión ambiental

Según la Federación de Cacaoteros de Colombia la planificación ambiental para el cultivo del cacao abarca una serie de objetivos fundamentales que buscan guiar y gestionar de manera efectiva esta actividad en relación con su impacto ambiental. Estos objetivos incluyen, entre otros:

- Adquirir un conocimiento exhaustivo de la normativa ambiental aplicable a la cacaocultura, garantizando así el cumplimiento de los requisitos legales vigentes.
- Reducir los efectos negativos que la actividad cacaotera pueda tener en el entorno ambiental, al mismo tiempo que se promueven y fomentan los impactos positivos que puedan surgir.
- Establecer cronogramas de trabajo basados en los diferentes procesos y etapas del cultivo, asegurando una planificación eficiente.
- Evaluar las distintas alternativas tecnológicas disponibles para la producción de cacao, con un enfoque particular en su influencia sobre el medio ambiente.
- Identificar y caracterizar los recursos naturales necesarios para respaldar el proceso productivo de manera sostenible.
- Implementar medidas destinadas a gestionar adecuadamente los posibles impactos ambientales negativos que puedan surgir a lo largo de la actividad.
- Definir, distinguir y cuantificar los productos, subproductos y residuos generados como resultado del proceso de producción del cacao.
- Asignar y garantizar los recursos necesarios para llevar a cabo las actividades previstas en los planes de manejo ambiental.
- Desarrollar proyectos cacaoteros en consonancia con los planes y esquemas de ordenamiento territorial específicos de la región en la que se llevan a cabo.
- Reconocer las condiciones naturales del ecosistema donde se implementará el proyecto, considerando su biodiversidad y características únicas.
- Realizar un análisis detallado de los actores sociales y económicos presentes en la región, entendiendo su influencia en el contexto cacaotero.
- Determinar el alcance del proyecto, incluyendo aspectos como la generación de empleo, la producción proyectada y su impacto en los mercados correspondientes.

- Establecer un cronograma de trabajo que se adapte a los procesos actuales o a las modificaciones planificadas, garantizando una ejecución eficiente y ordenada de las actividades.

2.1.7.1. Etapas de la planeación ambiental

La planeación ambiental se desarrolla a través de cinco etapas principales, las cuales son: preparación, evaluación, formulación, implementación y desarrollo, y seguimiento, como se ilustra en la **Figura 5**. A continuación, se detallan estas etapas:

a. Preparación

En esta fase se consideran las características generales del entorno ambiental, se analiza la normativa ambiental vigente y se delinean las medidas ambientales necesarias para la ejecución del proyecto.

b. Evaluación

Durante la etapa de evaluación, se examinan diversas alternativas tecnológicas viables para llevar a cabo el proyecto, se evalúan los recursos requeridos para la implementación de cada alternativa, se analizan los posibles beneficios que se derivan de cada opción y finalmente se elige una de las alternativas teniendo en cuenta aspectos ambientales, económicos y sociales.

c. Formulación

En esta fase, se inicia el proceso ante las entidades competentes para cumplir con todos los requisitos ambientales aplicables, como la obtención de licencias y permisos necesarios.

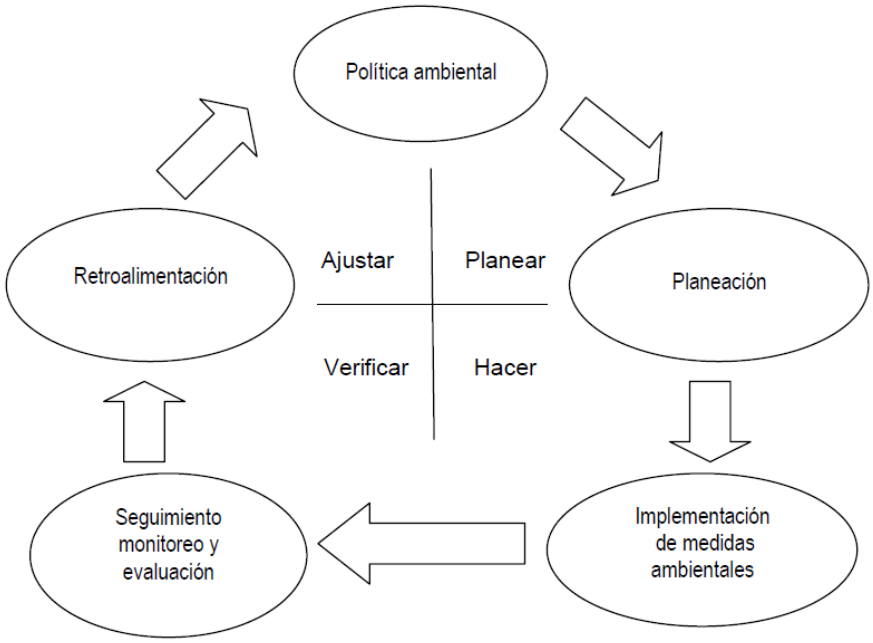
d. Implementación y desarrollo

Consiste en llevar a cabo las actividades y medidas ambientales previamente definidas en la etapa de preparación.

e. Seguimiento

El seguimiento implica una revisión continua del proceso, permitiendo la realización de ajustes y mejoras según sea necesario. Estos ajustes se centran en medidas relacionadas con la prevención, mitigación, control o compensación ambiental, garantizando así un enfoque adaptativo y sostenible en el proyecto.

Figura 5 : Esquema del proceso de Gestión Ambiental (MADR, 2013)



2.1.8. Enfermedades en el cacao

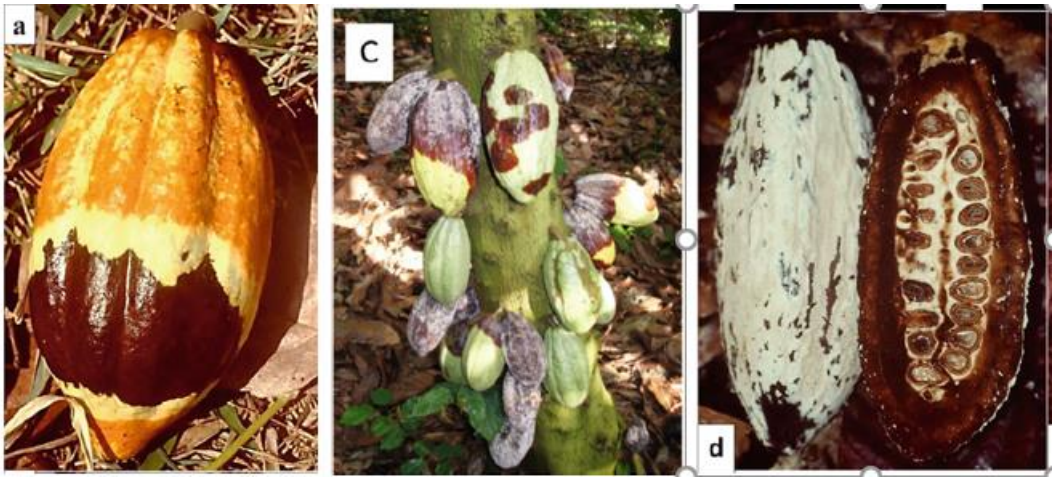
Uno de los problemas de reducción de la producción en cacao lo constituyen las enfermedades. Las enfermedades, en conjunto se estiman que hoy causan perdidas de entre el 30 a 50 %, y en algunas regiones hasta a pérdida total de la producción, cuando no se maneja la plantación, favorecidas por las condiciones favorables de factores climáticos, y por tanto la no aplicación de medidas o practicas agronómicas adecuadas y en momentos oportunos (Bailey y Meinhardt, 2016). Estas pérdidas, causadas por las enfermedades podrían ser reducidas, a través de un adecuado monitoreamiento de las condiciones climáticas, que definan periodos de condiciones climáticas extremas y adecuadas y por tanto permitan la puesta en práctica de medidas de control.

Las condiciones de clima que favorecen el ataque de enfermedades son las mismas de las que favorecen al cacao. Debido a su coevolución, el hospedero y el patógeno tienen similares requerimientos ambientales. (Leandro, 2017).

En este cultivo existen muchas enfermedades, pero tres de ellas son muy importantes, denominadas, moniliasis, escoba de bruja y pudrición parda (Figura 5.6) (Rios-Ruiz, 2004; Bailey y Meinhardt, 2016), Estas infectan los frutos del cacao, causando manchas externas de color pardo y café, seguido de una

esporulación blanquecina, causando la pudrición y pérdida total del fruto (Bailey y Meinhardt, 2016).

Figura 6 : Síntomas de frutos por la enfermedad: a. Escoba de bruja c. Pudrición Parda d. Moniliasis (Bailey y Meinhardt, 2016)



En Perú se ha reportado que la moniliasis causada por el hongo *Moniliophthora roreri* es la principal enfermedad del cultivo de cacao y ocasiona pérdidas de hasta el 90% en áreas no manejadas y además causa abandono o sustitución del cultivo (Rios-Ruiz, 2004).

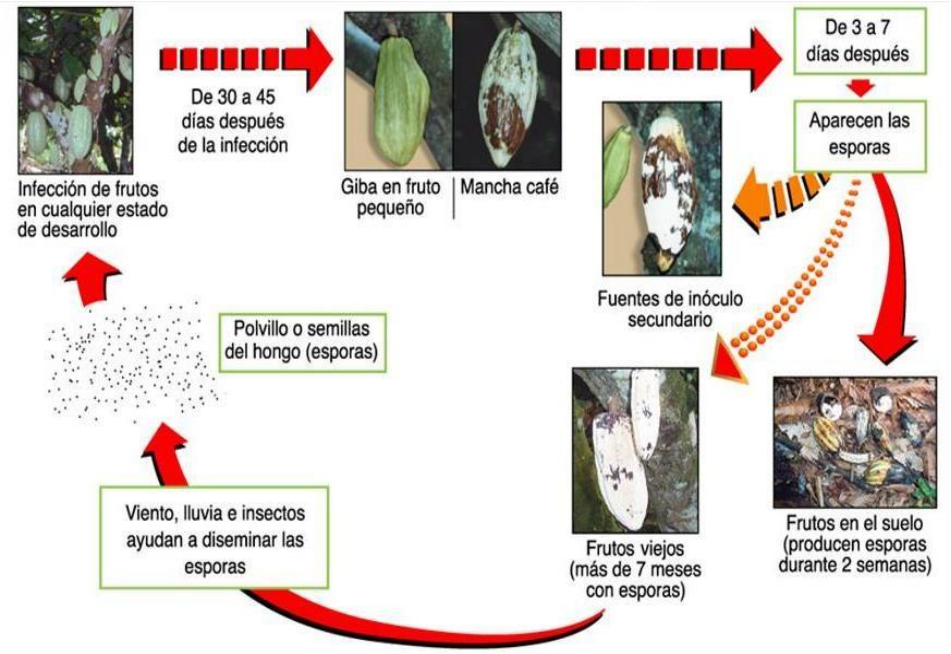
2.1.8.1. Moniliasis

Esta enfermedad en causada por el hongo *Moniliophthora roreri* es el principal patógeno que causa daños en el cacao. Puede causar la perdida de hasta el 100% de la cosecha sino se practica ningún control. El agente patógeno penetra en los frutos aún en proceso de desarrollo, dando origen a una secuencia de manifestaciones que incluyen la formación de manchas acuosas y untuosas, la distorsión de los tejidos frutales, la aparición de áreas pardas irregulares y la necrosis de los tejidos afectados. Además, induce la maduración temprana de las mazorcas y, en condiciones climáticas propicias, da lugar a la formación de extensas masas de esporas de color blanco. Este proceso culmina con la momificación de los frutos y la generación de pudriciones internas tanto en los frutos como en las semillas de cacao. Estas características, fácilmente reconocibles por los agricultores en el campo, son indicativas de la presencia del patógeno (Pérez, 2018).

El suceso de la infección de frutos por moniliasis de cacao es determinado por varios factores, incluyendo la humedad de los frutos. Es reportado que la presencia de una película de agua sobre la superficie del fruto es requerida para la germinación de esporas. Altas humedades relativas (80 a 100%) y temperaturas calientes (20 a 27 C) son condiciones adecuadas para la germinación de esporas y para la penetración del hongo. Condiciones favorables para la esporulación son similares a esos requeridos para infección. La esporulación es también dependiente de la humedad del fruto y temperaturas calientes (20 a 28 C), pero este último factor puede fluctuar en orden a estimular el proceso de esporulación. La luz es también otro importante factor en el proceso de esporulación. Periodos alternados de luz y oscuridad promueven la formación de esporas en condiciones in vitro (Leandro, 2017).

El tiempo de vida de moniliasis en condiciones de campo es de 68 a 85 días aproximadamente (**Figura 7**) (Murrieta y Palma, 2018).

Figura 7 : Ciclo de vida de moniliasis (Murrieta y Palma, 2018)



Es fundamental promover la implementación oportuna de prácticas de control cultural para contener la propagación de este patógeno. La eliminación de los frutos afectados constituye el enfoque más sencillo y efectivo para interrumpir el ciclo de la enfermedad, evitando que el hongo sobreviva entre las temporadas de cosecha. La estrategia de retirar los frutos enfermos de los

árboles de manera semanal y permitir que se descompongan de forma natural en el suelo ha demostrado ser altamente eficaz, reduciendo la incidencia de la enfermedad a niveles inferiores al umbral económico crítico. Esta labor debe centrarse en identificar y eliminar cualquier fruto que presente signos o síntomas de la enfermedad, otorgando prioridad a aquellos que presenten esporulación (Rios-Ruiz, 2004). Además, es esencial asegurar una adecuada visibilidad dentro de la copa de los árboles, lo cual se logra mediante una poda apropiada y manteniendo una altura de los árboles que no supere los 4.0 metros.

2.1.9. Monitoreamiento y evaluación de enfermedades.

La evaluación de enfermedades de plantas es uno de los más importantes y frecuentes tareas difíciles en la epidemiologia de enfermedades de plantas. La evaluación de una cantidad de enfermedad presente en un dado tiempo es la piedra angular para el análisis estadístico de los datos, esfuerzos de modelamiento e interpretación y control de un patosistema. Evaluar enfermedades usualmente consume mucho tiempo y por lo tanto es relativamente caro. El planeamiento cuidadoso de cómo, cuando, donde y cual enfermedad sería evaluado es esencial. El planeamiento requiere una clara concepción del objetivo específico de la investigación y un detallado conocimiento del patosistema (Campell & Madden, 1990).

La cuantificación de enfermedades es denominada de Patometría (pathos=enfermedad y metro=medir). La cuantificación de enfermedad es basada en los síntomas y signos (proporción del tejido enfermo del hospedero). La enfermedad puede ser cuantificada utilizándose dos parámetros: incidencia y la severidad. La incidencia es evaluada por el porcentaje de plantas, frutos, ramos, infectados. La severidad es evaluada por el porcentaje de área de tejido enfermo (síntomas y/o signos visibles).

La cuantificación de enfermedades envuelve dos etapas. La primera es la definición del muestreo de unidades que posee o tejido susceptible (enfermo con síntomas y/o signos) tales como mata, planta, ramos, hojas, frutos, etc. Generalmente se hace una caminata dentro del cultivo de modo se recorra en líneas rectas o se forme la letra W o X. Cuando se trata de parcelas experimentales lo correcto es marcar un determinado número de plantas (ese número varia con el tipo de experimento y cultivo) y hacer la cuantificación en

esas plantas. La segunda etapa es la evaluación de la incidencia o severidad de acuerdo con la enfermedad en estudio (Azevedo y Leite, 1995).

Como se ha manifestado la cuantificación correcta de enfermedades de plantas en la mayoría de las veces, es un proceso arduo y trabajoso. Además, envuelve una serie de procedimientos que deben ser seguidos y las personas deben ser entrenadas. En este contexto un estudio de investigación que intente adecuar un dispositivo que automatice este sistema servirá como herramienta de apoyo al trabajo de nuestros investigadores y productores agrícolas en la evaluación de enfermedades en tiempo real y con ello tener la información oportuna para orientar medidas de control de ellas, reducir las pérdidas que ocasionan y tener más producción en sus cultivos.

2.1.10. Estación Meteorológica

Una estación meteorológica se configura como un sitio meticulosamente seleccionado para la disposición de una variedad de instrumentos diseñados para la medición de diversas variables que inciden en el estado de la atmósfera. En otras palabras, representa un emplazamiento que facilita la observación de fenómenos atmosféricos y alberga dispositivos especializados para cuantificar las condiciones atmosféricas. Muchos de estos dispositivos requieren estar al aire libre, aunque en ciertos casos, es esencial protegerlos de la radiación solar para evitar que esta interfiera con la precisión de los datos recopilados; por lo tanto, se implementa una estructura que permite el flujo de aire en su interior. En cuanto a aquellos instrumentos que necesitan resguardarse de las inclemencias meteorológicas, se ubican dentro de una estructura denominada garita meteorológica. Esta garita se configura como una edificación elevada, con una altura mínima de 120 centímetros con respecto al suelo, y sus paredes adoptan una disposición de tipo persiana, estratégicamente orientada para bloquear la entrada de los rayos solares en su interior, preservando así la temperatura y la humedad de manera óptima. La entrada de la garita está orientada hacia el norte, y su techo se caracteriza por presentar una ligera inclinación. En su interior, se encuentran los instrumentos que requieren protección, resguardados de las condiciones climáticas mediante dispositivos registradores adecuados (Claudia Camptella, 2011).

Figura 8 : Estación Meteorológica automática (catálogo HOBO, 2020)

2.1.11. Vehículo aéreo no tripulado

Un vehículo aéreo no tripulado (UAV), conocido coloquialmente como dron, representa una categoría de aeronaves que opera sin la presencia de tripulación a bordo, desempeñando sus funciones de manera remota. Estos UAV son capaces de mantener un nivel de vuelo controlado y continuo de forma autónoma, empleando una variedad de tipos de motores, ya sea de explosión, eléctricos o de reacción, para su propulsión.

El diseño de UAVs abarca una amplia gama de configuraciones, tamaños y características, y se divide principalmente en dos variantes fundamentales: aquellos que son controlados de manera remota desde una ubicación externa y los que funcionan de manera autónoma, guiados por planes de vuelo previamente programados a través de sistemas de automatización dinámica.

Inicialmente, los UAVs encontraron sus aplicaciones principales en el ámbito militar, donde se les denominó vehículos aéreos no tripulados de combate. No obstante, con la creciente popularidad de los drones en usos civiles, sus aplicaciones se diversificaron significativamente, expandiendo la base de usuarios más allá del sector militar. Este vertiginoso crecimiento de su utilización ha llevado al surgimiento de diversas empresas, como Syma o DJI, que buscan

aprovechar este próspero nicho de mercado en el ámbito civil y comercial (Esther Salamí, 2014).

Figura 9 : Vehículo Aéreo no Tripulado de ala fija (Latin America in movement, 2015)

A lo largo de los años posteriores, se ha presenciado un notable y significativo avance en la evolución de los vehículos aéreos no tripulados (UAVs), también conocidos como drones, tanto en aplicaciones civiles como militares, hasta llegar a los dispositivos que son familiares en la actualidad. En la sociedad contemporánea, los drones han experimentado una profunda integración; más allá de su empleo con fines recreativos, se han consolidado como herramientas fundamentales en diversos sectores profesionales, abarcando desde la agricultura hasta la búsqueda y rescate de personas en situaciones de emergencia (S. L. Yang, 2017).

En el ámbito agrícola, los vehículos aéreos no tripulados (UAVs) se emplean de manera extensiva, especialmente para llevar a cabo tareas de monitoreo y control en los cultivos. Estos drones tienen la capacidad de cubrir extensas áreas de terreno de manera eficiente y veloz, al mismo tiempo que recopilan datos cruciales gracias a sus avanzados sensores. Dichos datos pueden incluir información relevante sobre aspectos como la humedad del suelo o la temperatura en los cultivos. Actualmente, existen numerosas empresas en el mercado dedicadas al diseño y fabricación de estos UAVs, logrando resultados de alto rendimiento en la agricultura moderna (Bejerano, s. f., 2021).

Las aplicaciones de los vehículos aéreos no tripulados (UAVs) en el ámbito agrícola presentan una amplia gama de posibilidades, dado que estos dispositivos son diseñados principalmente para realizar tareas de observación terrestre o para llevar a cabo la fumigación de los campos de cultivo.

En particular, los UAVs destinados a la fumigación agrícola están ganando creciente popularidad en países de América Latina, ofreciendo notables ventajas a los trabajadores del campo al contribuir significativamente en la aplicación de productos fitosanitarios esenciales para mantener cultivos saludables y libres de plagas.

Figura 10 : Drones de Fumigación (AGROPTIMA, 2020)

Por otro lado, los drones agrícolas de ala fija, equipados con cámaras para la captación de imágenes multiespectrales, desempeñan un papel fundamental al proporcionar información detallada sobre el estado de las plantaciones, permitiendo así una evaluación instantánea de los resultados obtenidos. Gracias a su capacidad de cobertura de terreno, estos dispositivos pueden captar datos precisos en un tiempo récord, llegando a abarcar hasta 500 hectáreas en una sola jornada.

En lo que respecta a los UAVs dotados de cámaras termográficas, desempeñan una función crucial en el estudio del terreno, debido a que generan mapas térmicos mediante imágenes y videos que ofrecen información esencial para evaluar factores como la humedad del suelo, el estrés hídrico en los cultivos y la temperatura foliar. Esto resulta esencial para la gestión de la siembra y el seguimiento de la salud de las plantas.

Figura 11. Dron con cámara multiespectral (Catálogo DJI Drones, 2022)

Por último, cuando se trata de estudiar la topografía del terreno, estos dispositivos son capaces de recopilar imágenes que permiten la elaboración de levantamientos topográficos extremadamente detallados. Esta tecnología de vanguardia ofrece una cobertura integral y de alta precisión, identificando características naturales, accidentes geográficos, cuerpos de agua y vías de comunicación en el terreno con un margen mínimo de error (Pino E. v., 2019).

2.2. Marco conceptual

2.2.1. Cacao

El término "cacao" se utiliza coloquialmente para hacer referencia al árbol conocido como cacaotero o *Theobroma cacao* en términos científicos. Este árbol es originario de las zonas tropicales y subtropicales de América y se clasifica botánicamente dentro de la familia Malvaceae. Se trata de una planta de hoja perenne que tiene un papel significativo en la producción de uno de los productos más apreciados en todo el mundo: el cacao.

2.2.2. Enfermedades

Una enfermedad agrícola se refiere a una población de organismos patógenos que infectan las plantas, y que tiene un impacto negativo en la producción del cultivo, lo que puede resultar en una disminución en el valor de la cosecha o un aumento en los costos de producción. Este concepto se centra en aspectos económicos. Las enfermedades agrícolas son ocasionadas por microorganismos que incluyen virus, bacterias, micoplasmas, viroides y hongos. Estas enfermedades pueden afectar de manera adversa la salud y el crecimiento de las plantas cultivadas.

2.2.3. Estación Meteorológica

Una estación meteorológica es una infraestructura diseñada específicamente para llevar a cabo mediciones periódicas y sistemáticas de múltiples variables relacionadas con las condiciones atmosféricas. La recopilación constante de estos datos es fundamental tanto para la generación de pronósticos meteorológicos basados en modelos computacionales como para la investigación y análisis de patrones climáticos a lo largo del tiempo.

2.2.4. Sensor

Un sensor es cualquier dispositivo que posee una característica que responde a un valor específico en su entorno y, a medida que esa magnitud particular cambia, también varía de manera proporcional su característica, de manera que detecta la existencia de esta magnitud y puede cuantificarla.

2.2.5. Vehículo Aéreo no Tripulado

Un vehículo aéreo no tripulado (UAV) o dron es una aeronave que puede operar sin necesidad de tripulación humana a bordo, y puede mantener un vuelo estable y controlado mediante la utilización de un motor que puede ser de explosión o de reacción.

2.2.6. Inteligencia Artificial

La inteligencia artificial (IA) en el ámbito de la informática se refiere a la capacidad de las máquinas, incluyendo sus procesadores y software correspondientes, para exhibir un nivel de inteligencia que se asemeja al funcionamiento de la mente humana. Esta inteligencia artificial se diferencia de la inteligencia natural que poseen los seres humanos y ciertos animales, la cual se basa en cerebros altamente complejos.

Capitulo III. Desarrollo del trabajo de investigación

3.1. Tipo y diseño de la investigación

Desde el punto de vista de la hipótesis, es de tipo inductiva debido a que se generalizará a partir de los datos tomados.

Desde el punto de vista de la medición, es de tipo experimental puesto que se tiene control sobre las condiciones del estudio.

Desde el punto de vista de la medición, es de tipo cuantitativa y longitudinal puesto que se basa en la medición objetiva y la recolección de datos cuantitativos a lo largo del tiempo.

El diseño de la investigación será aplicado debido a que se diseñará e implementará los dispositivos para el uso de un usuario final, los cuales pueden ser investigadores, profesores, alumnos o agricultores en general.

3.2. Unidad de análisis

La investigación utilizará como unidad de análisis una de las plantaciones de cacao de la Universidad Nacional Agraria de la Selva, ubicada en el centro experimental de Tulumayo en Tingo María.

3.3. Etapas de investigación

Etapa 1: Estudio de la problemática y búsqueda de información

- Búsqueda de información de trabajos similares realizados en el ámbito nacional e internacional: Se revisaron repositorios institucionales y revistas científicas con trabajos que se realizaron sobre los temas que estamos tratando, además también de proyectos realizados por los gobiernos provinciales y nacionales en Latinoamérica.
- Analizar la importancia del proyecto a realizar: Se buscaron literatura acerca de la problemática tratado y como está afectando tanto a grandes como pequeños productores, y que la propuesta que se está planteando disminuiría las pérdidas de producción debido al cambio climático e infección de plagas y enfermedades.
- Definir los beneficios del proyecto a realizar: Se realizó un estudio de cuanto podría ayudar esta propuesta a la producción de cacao a nivel nacional.

- Definición de los objetivos del proyecto y cronograma de actividades.

Etapa 2: Diseño del Circuito Electrónico y Primeras Pruebas

- Definir las variables climáticas a monitorear: Se tomaron dos factores esenciales, el primero es la búsqueda de literatura relacionada con cuales son las variables climáticas que más afectan o que son más importantes para el cultivo de cacao; y el segundo es la disponibilidad de los sensores para poder medir esas variables climáticas, y en caso de no haber una opción en el mercado, se verá si se diseña e implementa un nuevo sensor.
- Selección de los sensores y componentes electrónicos: Se hizo un estudio de mercado, de cuáles son las opciones actuales que están disponibles, tanto de sensores y módulos integrados comerciales, y se analizará cuáles podrían satisfacer las necesidades que queremos cubrir.
- Definición de la programación de la plataforma Iot: En este caso se programó una aplicación Web en donde se podrá observar en tiempo real el cambio en las variables climáticas, aquí se analizará que tecnología usar para la programación tanto del front end como del back end.
- Definir el número de rotores a utilizar en el dron: En esta parte se analizaron distintos estudios sobre la cantidad de rotores usados en drones comerciales, y cuál de estos se podría adecuar a lo que nosotros deseamos conseguir.
- Definir el controlador del dron: Se revisó distintas alternativas existentes y cuál sería el adecuado tanto para control de los rotores, así también como para que sea capaz de almacenar un algoritmo de inteligencia artificial.
- Definir el sistema de control del dron: Se revisaron todas las alternativas inalámbricas que existen para poder controlar un dron y de acuerdo con sus especificaciones se escogerá la que mejor se adecúe.
- Definir las tecnologías de software que se utilizarán en el dron: Se revisaron las tecnologías de software compatibles con el controlador y el sistema de control elegidos anteriormente y se elegirán en base a su eficiencia y facilidad de uso.

Etapa 3: Diseño de la placa electrónica y chasis del dron

- Definir los materiales a utilizar dado el entorno de trabajo: para ello se analizaron distintos materiales los cuales son usados comercialmente tanto en estaciones climáticas digitales y en drones, y se escogieron los que mejor se adapten a nuestras condiciones de campo.
- Analizar los costos y viabilidad del sistema mecánico desarrollado: Se escogieron los materiales y componentes que tengan la mejor relación calidad-precio y que además sean adecuados para nuestras condiciones de campo.

Etapa 4: Diseño de la programación interna de los dispositivos a implementar

- Búsqueda de información de las acciones desarrolladas por especialistas para la adaptabilidad climática del cultivo de cacao: Esto se realizó para poder aplicarlo al algoritmo interno de la estación meteorológica con el fin de poder brindar una alerta al agricultor dentro de la plataforma IoT.
- Desarrollar un código en Python para la manipulación de la información obtenida en las mediciones: Este algoritmo sirve para automatizar el proceso de adaptabilidad climática del cultivo de cacao, almacenando está información en una base de datos relacional por día, además de su visualización en tiempo real.
- Desarrollar el código controlador del dron, el cual es capaz de comunicarse con el control inalámbrico y enviar las respectivas instrucciones de movimiento a los rotores, además de implementar un lazo de control para mantener estable al dron en todo momento, el cual tomará los datos de sensores como giroscopio, magnetómetro y acelerómetro previamente seleccionados.

Etapa 5: Programación de la Plataforma IoT

- Programación de una Progressive Web App (PWA): Se realizó la programación de una PWA mediante el framework seleccionado anteriormente y se escogió esto debido a que es más accesible para verse tanto en PC como en el móvil, se realizó tanto la programación del front end como del back end, con la base de datos seleccionada, además

también de integrar los códigos de manipulación de datos en lenguaje Python.

- Elección de Servicio de Alojamiento Web: Se analizó las posibles plataformas de hosting disponibles en el mercado, que servirá para poner nuestra PWA a disposición de cualquier usuario que la solicite.

Etapa 6: Implementación de los diseños y Pruebas de campo

- Implementar los diseños propuestos: Se procedió a la compra de todos los sensores, componentes electrónicos y mecánicos a utilizar, y a su respectivo ensamblaje y prueba.
- Realizar pruebas de campo para analizar el funcionamiento del dispositivo: Se realizaron las pruebas en las plantaciones de Cacao de la Universidad Agraria de la Selva en el centro experimental de Tulumayo, Tingo María; para lo cual se cuenta con la ayuda de un profesional especialista en el tema y el cual es el encargado de esta plantación.
- Corregir los errores de diseño: En esta parte se corrigieron los errores que puedan presentarse durante las pruebas iniciales ejecutadas.

3.4. Matriz de Consistencia

En la **Tabla 4** se muestra la matriz de consistencia lógica.

Tabla 4 : Matriz de consistencia

PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLES		INDICADOR	METODOLOGÍAS EMPLEADAS
			DEPENDIENTE	INDEPENDIENTE		
¿Cómo influye el diseño de una estación climática automática y de un vehículo aéreo no tripulado en la mejora del sistema productivo del cacao?	Mejorar el sistema productivo en el cultivo de cacao basado en el diseño de una estación climática automática con aplicación IoT, además de un dron para el reconocimiento temprano de la moniliasis en el cultivo de cacao.	La propuesta de una estación meteorológica y de un dron de reconocimiento, junto con elementos de industria 4.0, ayudarán a mejorar el sistema de producción del cultivo de cacao automatizando los procesos de adaptabilidad climática del cultivo y la detección temprana de moniliasis.	Sistema productivo del cultivo de cacao.	Estación climática automática y vehículo aéreo no tripulado.	Voltajes de salida de los sensores para su interpretación en el microcontrolador seleccionado Colores y texturas características presentes en un fruto que presenta enfermedad Exceso o déficit en alguna unidad medida respecto al rango óptimo de operación para el cacao	Al ser una investigación del tipo aplicado y experimental se usará un área de ½ hectárea de cacao la cual contiene una muestra de alrededor de 160 plantas. En la definición de adaptabilidad climática se recolectarán los indicadores de 9 variables climáticas mediante sensores digitales de bajo coste, todos conectados a un microcontrolador que contiene un programa con el cual se convierte la señal analógica de los sensores a valores que puedan ser entendidos fácilmente por cualquier persona. Para la detección de frutos
¿Cómo influye el desarrollo de una plataforma IoT para la visualización de datos climáticos?	Diseñar una plataforma IoT que tome los datos de la estación climática, los almacene en una base de datos y puedan ser visualizados en tiempo real.	La plataforma IoT tendrá la capacidad de presentar de manera precisa los datos climáticos en tiempo real de la estación tanto en computadoras personales como en dispositivos móviles. Además, establecerá una conexión con la base de datos para almacenar y analizar estos datos de manera eficiente. Este entorno estará protegido por autenticación de usuario y contraseña para garantizar la seguridad del acceso.				
¿Cómo influye el diseño de hardware para la toma de datos climáticos y para el sistema de vuelo de un dron	Diseñar la arquitectura de hardware para la toma de datos climáticos y para el sistema de vuelo del dron, con el fin de	Los sensores, actuadores y módulos integrados a utilizarse en el diseño serán los óptimos en relación calidad-precio, además de lograr una buena sinergia				

de monitoreamiento?	mejorar los procesos de cultivo de cacao y minimizar sus costos	juntos.			Cantidad de plantas que presentan frutos enfermos por área	enfermos por medio de un dron se recolectarán imágenes de los frutos de diferentes plantas por medio de una cámara de alta definición y se almacenarán en la memoria interna de un microcomputador.
¿Cómo influye el diseño de una herramienta de detección de moniliasis en el cultivo de cacao?	Diseñar un algoritmo de inteligencia artificial para la detección de moniliasis en el fruto del cacao.	El algoritmo de inteligencia artificial diseñado será capaz de reconocer la moniliasis del cacao y notificar al agricultor para su respectiva extracción.				
¿Es posible validar los algoritmos y la arquitectura de hardware de la estación climática y vehículo aéreo no tripulado en las condiciones de la prueba de campo?	Validar los algoritmos y la arquitectura de hardware de la estación climática y vehículo aéreo no tripulado en la mejora del sistema productivo del cacao.	La validación de los algoritmos y la arquitectura de hardware de la estación climática y vehículo aéreo no tripulado logrará servir de ayuda en la mejora del sistema productivo del cacao.				

3.5. Estación climática automática

3.5.1. Visita a estaciones climáticas comerciales

Se llevaron a cabo visitas a las estaciones climáticas del Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI) ubicadas en la sede de Campo de Marte en Jesús María y en la estación climática de la Universidad Nacional Agraria de la Selva en Tingo María. El propósito principal de estas visitas fue adquirir un conocimiento detallado sobre los sensores y las metodologías actualmente empleados en la recopilación de datos climáticos a nivel nacional.

Figura 12 : Visita Estación Climática UNAS

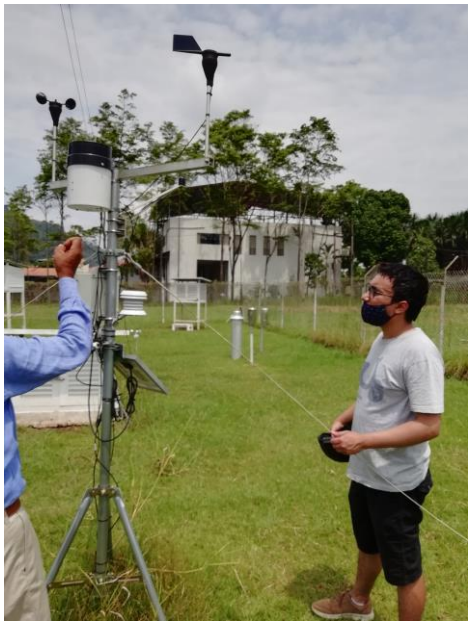


Figura 13 : Viendo Funcionamiento de Estación Climática Automática UNAS

Durante estas visitas, se pudo obtener información relevante acerca de las normativas y los parámetros que rigen la captura de datos climáticos, incluyendo aspectos como:

– La medición de la temperatura y la Humedad Relativa se efectúa en un entorno sombreado que permita la circulación del aire, y se realiza a una altura que oscila entre 1.25 metros y 2 metros del suelo. En el contexto de estas mediciones, se ha observado tanto el enfoque tradicional, en el cual estas condiciones se logran mediante una caseta de madera con aberturas diseñadas para el flujo de aire, como el enfoque moderno, que involucra la utilización de sensores digitales alojados en pequeñas estructuras con forma de vasos invertidos, las cuales cumplen con los estándares requeridos. Es importante destacar que, siguiendo las directrices del manual más reciente para la instalación de estaciones climáticas automáticas del Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI), se establece un rango de temperatura que debe estar comprendido entre -40 °C y 60 °C (OMM N° 305,1993, Suplemento Cap.6,2001). Además, se especifica que el sensor utilizado debe ofrecer una resolución de 0.1°C y una exactitud de 0.3°C para las mediciones de temperatura. En lo que respecta a la Humedad Relativa, se requiere que se encuentre en un rango del 0 al 100%, y el sensor empleado debe contar con una resolución de 1% y una exactitud de 3%.

Figura 14 : Método Tradicional de Medición de Humedad y Temperatura

Figura 15 : Método moderno de medición de Temperatura y Humedad

– Para llevar a cabo la medición de la precipitación, se ha considerado tanto el enfoque tradicional como el moderno. En el método tradicional, un experto realiza mediciones diarias de la precipitación utilizando una probeta. Por otro lado, el método moderno implica el empleo de un sensor de doble cubeta basculante, el cual se encuentra conectado a un software especializado para la captura de datos. Estas mediciones se efectúan a una altura que varía entre 1 metro y 1.5 metros sobre el nivel del suelo, con una altura típica de 1.3 metros seleccionada como estándar. Es relevante destacar que, siguiendo las pautas establecidas en el manual más reciente del Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI), se define un rango para las precipitaciones que debe situarse entre 0 y 401 mm/h (Galmarini, 2004). Además, se establece que el sensor utilizado para estas mediciones debe proporcionar una resolución de 0.2 mm y una exactitud de 0.2 mm.

Figura 16 : Método tradicional de medición de Precipitación con Probeta

Figura 17 : Método moderno de medición de Precipitación con doble Cubeta Basculante

– Para la medición de la dirección y velocidad del viento, se han observado diversas configuraciones de anemómetros y veletas. Estas configuraciones incluyeron tanto sistemas en un solo eje como aquellos con ejes separados y se consideraron diferentes alturas de medición. La elección de la altura de medición está condicionada por el propósito específico de adquirir datos sobre el viento. En el contexto de esta

investigación, se ha optado por una altura de 2.5 metros sobre el suelo, la cual se corresponde con la altura estándar en las plantaciones clonales de cacao. Esta elección se fundamenta en su capacidad para proporcionar soluciones óptimas en términos de adaptación climática del cultivo de cacao. Es importante mencionar que, de acuerdo con las directrices del Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI), el sensor utilizado para la medición de la velocidad del viento debe cubrir un rango de lectura que oscile entre 0 y 50 m/s, presentando una resolución de 0.5 m/s y una exactitud de 0.5 m/s. Por otro lado, el sensor destinado a la medición de la dirección del viento debe abarcar un rango de 0° a 360°, ofreciendo una resolución de 1° y una exactitud de 5°, siguiendo las pautas establecidas por el SENAMHI.

Figura 18 : Anemómetro y Veleta en Estación Climática Automática

– Para la medición de la temperatura y humedad del suelo, se han considerado tanto métodos tradicionales como enfoques modernos. En la aproximación tradicional, se emplean termómetros insertados en el suelo a diferentes profundidades para obtener datos sobre las condiciones a diferentes niveles. En contraste, en el método moderno se utiliza un único sensor de temperatura, posicionado a una profundidad específica según el fenómeno que se esté investigando. En el contexto del estudio del

cultivo de cacao, se ha seleccionado una profundidad de 30 cm, dado que esta ubicación se caracteriza por albergar la mayor densidad de raíces de las plantas y, por ende, es la zona principal de absorción de nutrientes por parte de estas.

Figura 19 : Sensores de Temperatura a diferentes profundidades

– En lo que respecta al sensor de radiación, su ubicación dentro de la estación meteorológica es de vital importancia. Debe estar estratégicamente posicionado de manera que no se proyecte ninguna sombra sobre él. La altura recomendada para su colocación oscila entre 1.5 metros y 3 metros, asegurando así una captación precisa de los datos de radiación solar incidente. Además, el sensor debe contar con un amplio rango de lectura, abarcando desde 0 hasta 1300 w/m2, lo que le permitirá captar una amplia gama de niveles de radiación. Asimismo, se espera que el sensor presente una alta resolución, con una capacidad de medición precisa de 1 w/m2, y una exactitud del 5%, garantizando la fiabilidad de los datos recopilados.

– El sensor encargado de medir la presión atmosférica desempeña un papel crucial en la estación meteorológica. Su rango de lectura, que va desde los 500 hPa hasta los 1050 hPa, le permite captar una amplia gama de variaciones en la presión atmosférica. Además, este sensor se caracteriza por su alta resolución, con una capacidad de medición precisa de 0.1 hPa, lo que asegura la obtención de datos detallados y confiables. Para garantizar su precisión, se espera que el sensor tenga una exactitud del orden de 0.3 hPa. En términos de ubicación, se recomienda que el sensor esté posicionado a una altura que oscile entre 1.5 metros y 2 metros sobre el suelo, permitiendo así la captación precisa de datos representativos de la presión atmosférica en la zona de estudio. Para preservar su funcionamiento y durabilidad, se coloca dentro de un gabinete de protección que lo resguarda de las condiciones climáticas extremas, asegurando su operatividad a lo largo del tiempo.

3.5.2. Elección de Componentes

Dentro del contexto del cultivo de cacao, y respaldado por investigaciones realizadas por expertos en la materia, se identifican como las principales variables climáticas que ejercen influencia sobre el cultivo de cacao las siguientes: la humedad y temperatura del aire, la dirección y velocidad del viento, la precipitación, la humedad y temperatura del suelo, y la intensidad luminosa. Teniendo en consideración esta premisa, se emprendió una búsqueda exhaustiva orientada a la identificación de sensores y componentes electrónicos de bajo costo que fueran capaces de proporcionar datos precisos y confiables para su posterior análisis. En el ámbito de las opciones disponibles en el mercado comercial, se destacan las siguientes alternativas como las más prometedoras para este propósito.

Para la temperatura del aire:

Tabla 5 : Sensores de Temperatura del aire

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Sensor ntc/Ptc	Resistencia	Variable	Variable	Variable	Variable
Ds18b20	Digital	3v - 5.5v	-55°c a +125°c	0.5°c	0.0625°c
Dht11/dht22	Digital	3v - 5v	-40°c a 80°c	0.5°c	0.1°c
Mlx90614	Infrarrojo	3.3v - 5v	-70°c a +380°c	0.5°c	0.002°c
Lm35	Analógico	4v - 30v	-55°c a +150°c	0.5°c	0.1°c
Bmp180/bmp280	Digital	1.8v - 3.6v	-40°c a +85°c	1°c	0.01°c
Max31855	Digital	3v - 5.5v	-270°c a 1372°c	2°c	0.25°c

Para la humedad del aire:

Tabla 6 : Sensores de Humedad del aire

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Dht11/dht22	Digital	3v - 5v	20% a 80%	2%	1%
Sht21/sht31	Digital	2.1v - 3.6v	0% a 100%	2%	0.01%
Bme280	Digital	1.8v - 3.6v	0% a 100%	3%	0.001%
Sensor capacitivo	Capacitivo	Variable	Variable	Variable	Variable
Sensor resistivo	Resistivo	Variable	Variable	Variable	Variable

Para la velocidad y dirección del viento:

Tabla 7 : Sensores de velocidad y dirección del viento

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Anemómetro de Copas	Mecánico	Variable	0.5m/s a 75 m/s	1 m/s	Variable
Anémometro de Paletas	Mecánico	Variable	0.5 m/s a 75 m/s	1 m/s	Variable
Veleta electrónica	Electrónico	5v	0° a 360°	2°	1°
Sensor de Efecto Hall	Electrónico	5v	Variable	0.1 m/s	Variable
Sensor ultrasónico	Ultrasónico	5v	0.2 m/s a 75 m/s	Variable	Variable
Sensor de Luz Solar	Electrónico	3.3v - 5v	Variable	Variable	Variable
Brújula electrónica	Electrónico	3v - 5v	0° a 360°	1°	1°

Para la precipitación:

Tabla 8 : Sensores de Precipitación

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Sensor de Inclinación	Mecánico / electrónico	5v	0.01mm a 5mm/h	Variable	Variable
Sensor Ultrasónico de Nivel de Agua	Ultrasónico	5v	0.1mm a 10mm/h	Variable	Variable
Sensores de Peso (Basculantes o de Báscula)	Mecánico / electrónico	Variable	0.01mm a 5 mm/h	Variable	Variable
Sensores de Caudal	Mecánico / electrónico	Variable	Variable	Variable	Variable

Para la temperatura del suelo:

Tabla 9 : Sensores de Temperatura del suelo

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Ds18b20	Digital	3v - 5.5v	-55°c a +125°c	0.5°c	0.0625°c
Termistor ntc	Analógico	Variable	-50°c a +150°c	Variable	Variable
Termistor ptc	Analógico	Variable	Variable	Variable	Variable
Termopar	Analógico	Variable	-270°c a +1800°c	Variable	Variable

Para la humedad del suelo:

Tabla 10 : Sensores de Humedad del Suelo

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Fc-28	Resistivo	3.3v - 5v	0% a 100%	Variable	Variable
YI-69	Resistivo	3.3v - 5v	0% a 100%	Variable	Variable
Capacitive soil moisture sensor	Capacitivo	3.3v - 5v	0% a 100%	Variable	Variable
Sensor de humedad del suelo de tensión matricial	Tensiómetro	Variable	Variable	Variable	Variable

Para la luminosidad

Tabla 11 : Sensores de Luminosidad

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Fotodiodo (Ldr)	Fotoresistor	3v - 5.5v	Variable	Variable	Variable
Fototransistor npn	Fototransistor	3v - 5.5v	Variable	Variable	Variable
Fototransistor pnp	Fototransistor	3v - 5.5v	Variable	Variable	Variable
Tsl2561	Sensor de luz	3v - 5.5v	0.1 lux a 40000 lux	Variable	Variable
Bht1750	Sensor de luz	2.4v - 3.6v	0.11 lux a 100000 lux	20%	1 lux
Tcs230	Sensor de color	2.7v - 5.5v	No aplicable	No aplicable	No aplicable

Dada la disponibilidad y accesibilidad de este tipo de sensores, así como en un esfuerzo por lograr una medición integral de un amplio conjunto de variables, se ha optado por incluir además las mediciones de radiación y presión atmosférica. Estas variables se encuentran intrínsecamente relacionadas con las anteriormente mencionadas, y como resultado, se han propuesto las siguientes incorporaciones:

Para medir la radiación:

Tabla 12 : Sensores de Radiación

Sensor	Tipo	Voltaje operacional	Rango de Medición	Exactitud	Resolución
Sensor geiger-Muller	Gas lleno	400v - 1200v	0.05-1000msv/h	20%	0.01 msv/h
Sensor de radiacion pin	Semiconductor	10v - 15v	0.01-10msv/h	10%	0.001 msv/h
Ml8511	Semiconductor	2.8v - 5.5v	0.01-10msv/h	5%	0.001 msv/h

Para medir la presión:

Tabla 13 : Sensores de Presión Atmosférica

Sensor	Tipo	Voltaje operacional	Rango de medición	Exactitud	Resolución
Bmp180	Piezoresistivo	3.3v	30 kpa a 110 kpa	1%	1 pa
Bmp280	Piezoresistivo	3.3v	30 kpa a 110 kpa	0.5%	0.1 pa
Bmp380	Piezoresistivo	3.3v	30 kpa a 110 kpa	0.25%	0.01 pa
Ms5611	Resistivo	3.3v	10 kpa a 1200 mbar	0.01%	0.001 pa
Lps25h	Capacitivo	3.3v	260 pa a 110 kpa	2%	1 pa
Bme280	Piezoresistivo y capacitivo	3.3v	30 kpa a 110 kpa	0.5%	0.01 pa

Dentro de todas las opciones que encontramos en el mercado, se eligieron las siguientes:

Temperatura y Humedad del Aire - DHT22:

La elección del sensor DHT22 se basa en su versatilidad y facilidad de uso para medir tanto la temperatura como la humedad en una única unidad, lo que resulta conveniente para aplicaciones que requieren el monitoreo de ambas variables. Además, se destaca por su compatibilidad y facilidad de conexión con placas como Arduino y otros microcontroladores, y se beneficia de la disponibilidad de bibliotecas que simplifican la programación, lo que lo convierte en una opción eficiente y efectiva para obtener mediciones precisas en diversas aplicaciones.

Figura 20 : Sensor de Temperatura DHT22

Temperatura del Suelo - DS18B20:

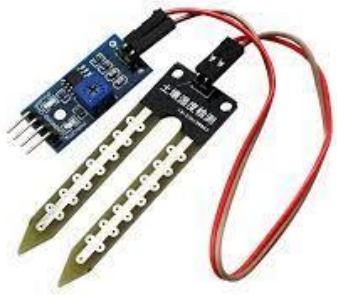
La elección del sensor DS18B20 se sustenta en su reconocida precisión en la medición de la temperatura, lo que lo convierte en una elección idónea para aplicaciones que demandan lecturas extremadamente precisas en el suelo. Además, su capacidad de ser resistente al agua, gracias a las variantes que se encuentran encapsuladas en sondas impermeables, habilita su empleo en entornos caracterizados por condiciones húmedas o de suelo, brindando así una versatilidad y fiabilidad adicionales en diversas situaciones de monitoreo ambiental.

Figura 21 : Sensor de Temperatura DS18B20

Humedad del Suelo - FC-28:

La elección del sensor FC-28 se fundamenta en su capacidad de llevar a cabo una detección directa de la resistencia eléctrica, permitiéndole medir eficazmente la humedad del suelo. Esto lo convierte en una selección apropiada para aplicaciones relacionadas con la agricultura y la jardinería, donde la medición precisa de la humedad es esencial. Además, su coste asequible lo posiciona como una opción rentable para la monitorización de la humedad del suelo, y su disponibilidad generalizada en el mercado facilita su adquisición y uso en proyectos de investigación y aplicaciones prácticas.

Figura 22 : Sensor de Humedad FC-28



Presión - BMP180:

La elección del sensor BMP180 se justifica en su capacidad para medir con alta precisión la presión atmosférica, lo que lo convierte en una selección idónea para aplicaciones que demandan información precisa sobre las variaciones en la presión atmosférica. Además, su diseño de tamaño compacto y su eficiente consumo de energía lo hacen altamente apropiado para su incorporación en proyectos portátiles y sistemas donde el espacio y la eficiencia energética son factores críticos a considerar, garantizando así una medición confiable y eficaz de la presión atmosférica en diversos contextos de investigación y aplicaciones prácticas.

Figura 23 Sensor de Presión BMP180

Luminosidad - BH1750:

La elección del sensor BH1750 se basa en su capacidad para proporcionar mediciones altamente precisas de la luminosidad en una amplia gama de condiciones de iluminación, lo que lo convierte en una opción altamente adecuada para aplicaciones que requieren un control preciso de la iluminación o monitoreo ambiental en entornos variables. Además, su interfaz de

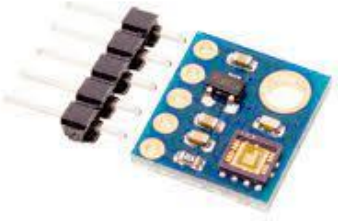
comunicación a través del protocolo I2C facilita significativamente su integración con microcontroladores y sistemas, lo que simplifica la implementación y el uso del sensor en una variedad de proyectos y aplicaciones. Esto asegura una medición confiable y efectiva de la luminosidad, contribuyendo al éxito de investigaciones y proyectos que dependen de datos precisos de iluminación.

Figura 24 : Sensor de Luminosidad BH1750

Radiación - ML8511:

La elección del sensor ML8511 se fundamenta en su especialización en la medición de radiación ultravioleta (UV), lo que lo convierte en una elección idónea para aplicaciones que requieren la detección y cuantificación precisa de la radiación UV, como la evaluación de la exposición solar y la identificación de riesgos relacionados con los rayos UV. Su capacidad para proporcionar una salida de voltaje analógica simplifica su interfaz con microcontroladores, lo que facilita la adquisición y el procesamiento de datos, lo que es esencial en investigaciones y aplicaciones que dependen de mediciones precisas de radiación UV para tomar decisiones informadas.

Figura 25 : Sensor de Radiación Ultravioleta ML8511



Velocidad de Viento - Sensor de Efecto Hall:

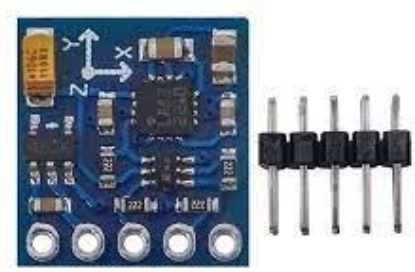
La elección del sensor de efecto Hall para la medición de la velocidad del viento se basa en sus ventajas significativas. Este sensor se caracteriza por su diseño sin piezas móviles, lo que implica una mayor durabilidad y una menor susceptibilidad al desgaste con el tiempo. Además, ofrece la ventaja de requerir un mantenimiento reducido en comparación con otros sensores de velocidad de viento, como los anemómetros mecánicos. Esta característica se traduce en una mayor confiabilidad y eficiencia a lo largo del tiempo, lo que lo convierte en una elección idónea para aplicaciones que demandan mediciones precisas y continuas de la velocidad del viento con una inversión mínima en mantenimiento.

Figura 26 : Sensor de Efecto Hall

Dirección de Viento - HMC5883:

La elección del sensor HMC5883 para la medición de la dirección del viento se basa en su capacidad para detectar el campo magnético terrestre, lo que posibilita estimar con precisión la dirección del viento. A través de su magnetómetro, este sensor es capaz de registrar las variaciones en el campo magnético, lo que se traduce en una valiosa información sobre la dirección del viento. Un aspecto destacado es que, al igual que los sensores de velocidad de viento basados en efecto Hall, el HMC5883 no contiene piezas móviles, lo que conlleva una mayor durabilidad y una reducción sustancial en la necesidad de mantenimiento. Esta característica, combinada con su capacidad para proporcionar mediciones fiables de la dirección del viento, lo convierte en una elección altamente adecuada para aplicaciones que requieren una medición precisa y constante de la dirección del viento.

Figura 27 : Sensor de Orientación HMC5883



Precipitación – Sensor de inclinación de doble cubeta basculante:

Se optó por el diseño de un sensor de inclinación de doble cubeta basculante que incorpora un sensor de efecto Hall para detectar y registrar una señal cada vez que el balancín experimenta una inclinación. Esta elección se fundamenta en la necesidad de desarrollar un dispositivo confiable y eficaz para la medición precisa de la precipitación. El sensor de inclinación, al utilizar un sensor de efecto Hall, asegura una detección sensible de los movimientos del balancín, lo que permite una representación precisa de la cantidad de precipitación. Este diseño se ha concebido con el propósito de proporcionar datos meteorológicos precisos y confiables, lo que lo convierte en una elección adecuada para aplicaciones que requieren un seguimiento preciso de las condiciones climáticas, especialmente en lo que respecta a la precipitación.

Luego de efectuar la selección de los sensores que serán empleados en la recopilación de datos relativos a las variables climáticas, se procedió a realizar un análisis exhaustivo con el propósito de determinar cuáles microcontroladores resultan más apropiados para la lectura y procesamiento de las señales emanadas de dichos sensores. Dentro del espectro de alternativas disponibles, se identificaron diversas opciones tales como:

Tabla 14 : Características de Microcontroladores

Característica	Arduino	Raspberry pi	Esp8266 / esp32	Pic
Tipo de Dispositivo	Microcontrolador	Sbc (singel board computer)	Microcontrolador	Microcontrolador
Potencia de Procesamiento	Baja	Media a alta	Media a alta	Baja a media
Sistema operativo	No	Si (linux)	No	No
Conectividad wifi	Requiere módulo adicional	Incorporada	Incorporada	Requiere módulo adicional
Memoria ram	Limitada (kb a mb)	1gb a más	Varía (kb a mb)	Limitada (kb a mb)
Almacenamiento interno	Limitado (kb a mb)	Tarjta microsd	Varía (kb a mb)	Limitado (kb a mb)
Interfaces de E/s	Gpio, analógicas	Gpio, usb, hdmi, más	Gpio, analógicas, más	Gpio, analógicas, más
Comunidad y soporte	Amplia	Amplia	Amplia	Varía
Consumo de Energía	Bajo	Medio	Bajo	Bajo a medio
Costo	Económico	Moderado a alto	Económico	Varía
Idoneidad para iot	Si	Si	Si	Si

En el proceso de elección entre diversas opciones de microcontroladores para la implementación de nuestra estación climática con funcionalidad IoT, se tomó la decisión de utilizar el microcontrolador NodeMCU ESP8266. Esta elección se basó en una serie de consideraciones. En primer lugar, el NodeMCU ESP8266 se destacó por su integración de módulo WiFi, lo que simplifica considerablemente las conexiones requeridas para habilitar la aplicación IoT. Además, su diseño compacto en comparación con el ESP32 contribuyó a su elección.

Figura 28 : NodeMCU ESP8266

Sin embargo, es importante señalar que el ESP8266 presenta ciertas limitaciones, como disponer únicamente de una entrada analógica. Para abordar este desafío, se optó por incorporar un multiplexor analógico (CD74HC4067), que permitirá la lectura secuencial de las diversas señales analógicas.

Figura 29 : Multiplexor Analógico CD74HC4067

Asimismo, se identificó la posibilidad de enfrentar problemas de espacio en la memoria del ESP8266, dada la necesidad de gestionar un total de diez variables climáticas. En respuesta a esta inquietud, se decidió incorporar otro microcontrolador en el sistema, en este caso, un Arduino Nano. La elección del Arduino Nano se fundamenta en la disponibilidad de librerías adecuadas para trabajar con los sensores seleccionados en su entorno de desarrollo. Además, su diseño compacto lo hace idóneo para la tarea. Este Arduino Nano se encargará de la lectura de los sensores que operan mediante comunicación I2C y transmitirá los datos correspondientes al ESP8266, donde se llevará a cabo la implementación de la aplicación IoT.

Figura 30 : Arduino Nano

3.5.3. Alimentación de la Estación Climática

Luego de haber seleccionado cuidadosamente los componentes electrónicos que conformarán la estación meteorológica automática surgen un factor crítico que demanda nuestra atención: la provisión de energía para mantener operativa la estación las 24 horas del día. Esta provisión energética es esencial para asegurar la obtención de datos precisos y fiables, fundamentales para análisis posteriores.

En este contexto, es importante señalar que, en las condiciones de campo específicas de este proyecto, no existe un suministro eléctrico disponible en las plantaciones de cacao. Por lo tanto, la alternativa más viable para superar este desafío es emplear una batería capaz de suministrar la corriente eléctrica requerida para el funcionamiento continuo de la estación. Dado que se realizarán pruebas en una zona que cuenta con abundante luz solar, se plantea la posibilidad de aprovechar esta fuente de energía natural para recargar la batería durante las horas diurnas mediante un panel solar.

En línea con este enfoque, se ha llevado a cabo un cálculo minucioso para determinar la selección adecuada tanto del panel solar como de la batería. Este cálculo se basa en el consumo eléctrico de cada componente electrónico empleado y en las horas de luz solar disponibles en la ciudad de Tingo María, Perú. Además, cabe destacar que se ha considerado la incorporación de un componente adicional en el sistema: un regulador de voltaje. La razón detrás de esta elección radica en que las baterías diseñadas para paneles solares generalmente suministran un voltaje estándar de 12V, mientras que los sensores y microcontroladores operan con un voltaje de alimentación que oscila entre 3.3V

y 5V. En este contexto, se han evaluado diversas opciones comerciales para seleccionar el regulador de voltaje más apropiado.

Tabla 15 : Reguladores de Voltaje

Regulador de Voltaje	Tensión de Entrada (v)	Tensión de Salida (v)	Corriente de Salida (a)	Eficiencia (%)	Tipo de Regulación
Lm7805	7v - 25v	5v	1a	60% - 80%	Regulación lineal
Lm317	Variable	Variable	1.5a	Variable	Regulación ajustable
Lm2596	4.5v - 40v	1.23v - 37v	3a	80% - 92%	Regulación conmutada
Mp1584	4.5v - 28v	Ajustable	3a	85% - 93%	Regulación conmutada

Dentro de las alternativas disponibles, se optó por el regulador de voltaje LM2596 debido a sus características destacadas. El LM2596 es un regulador de voltaje conmutado, también conocido como reductor de voltaje (step-down), que se destaca por su eficiencia energética. En contraste con los reguladores lineales tradicionales, como el LM7805, el LM2596 genera menos calor durante su operación y minimiza las pérdidas de energía en forma de calor. Esta cualidad adquiere una relevancia particular en situaciones donde se requiere alimentar dispositivos con baterías, debido a que contribuye significativamente a prolongar la duración de la batería.

Una ventaja significativa del LM2596 radica en su capacidad para manejar un amplio rango de voltajes de entrada, normalmente desde unos pocos voltios por encima de su voltaje de salida hasta varios voltios por encima de dicho voltaje. Esta característica lo hace versátil y adaptable a diversas fuentes de alimentación, como baterías cuyo voltaje puede variar a medida que se descargan.

Además de su eficiencia energética, el LM2596 está equipado con características de seguridad esenciales, incluyendo protección térmica y protección contra sobrecorriente, lo que lo convierte en una elección segura en situaciones donde puedan surgir condiciones de funcionamiento anormales. Estas cualidades hacen del LM2596 una elección acertada y confiable para la aplicación en cuestión.

Figura 31 : Regulador de Voltaje LM2596

Conociendo todos los elementos electrónicos que estarán involucrados en la estación climática automática se procedió a calcular el consumo eléctrico de cada componente y en su conjunto para poder hacer la elección de nuestra batería y panel solar.

Tabla 16 : Consumo Energético de los Componentes de la Estación Climática

Componente	Voltaje (v)	Corriente (a)	Potencia (w)	Potencia Total en 24h (Wh)
Nodemcu esp8266	5	0.1	0.5	12
Arduino nano	5	0.1	0.5	12
Cd74hc4067	3.3	0.025	0.0825	1.98
Fc-28	3.3	0.035	0.1155	2.772
Dht22	3.3	0.03	0.099	2.376
Sensor efecto hall - Anemómetro	5	0.01	0.05	1.2
Sensor efecto hall - Pluviómetro	5	0.01	0.05	1.2
Lm2596	5	0.05	0.25	0.6
Bh1750	5	0.02	0.1	2.4
Hmc5883	5	0.01	0.05	1.2
Bmp180	5	0.02	0.1	2.4
Lm8511	3.3	0.01	0.033	0.792
Ds18b20	3.3	0.01	0.033	0.792
Total		0.43		41.712

Con estos datos se calculó las horas solares pico en la localidad de Tingo María que es el número de horas que existe irradiación solar constante de valor 1000W/m2.

Según el atlas de minería y energía del Ministerio de Energía y Minas, Tingo María cuenta con una irradiación solar promedio anual de 4020 Wh/m2.

Para obtener las horas solares pico se dividió estas cantidades:

$$HPS = {}^{4020\ Wh/m2}/_{1000\ W/m2} = 4.02\ h$$

Para satisfacer las necesidades de todo el circuito se debería contar con un panel de potencia mayor o igual a 41.712Wh / 4.02 h = 10.38 W.

Tingo María al estar ubicado en una zona de Selva del Perú durante todo el año los días presentan alrededor de 10 horas de sol, teniendo como máxima irradiación 4.02h como ya vimos anteriormente. Teniendo en cuenta esto se tomó la corriente máxima que consume todo el circuito y se multiplicó por las horas de autonomía que tiene que proporcionar la batería al circuito 0.43A x 14h = 6.02Ah.

Habiendo hecho todos estos cálculos se optó por la utilización de un panel solar de 20W y una batería de 7Ah los cuales con el consumo de los componentes son más que suficientes para que el circuito funcione las 24 horas del día sin parar, está elección se basó tomando en cuenta en que la eficiencia de este componente no es del 100% y también como manera de prevención para días en los que las horas de sol sean menores. Estos valores también proporcionan la posibilidad de que la estación climática pueda ser usada en lugares donde la autonomía de la estación sin horas de sol sea mayor, garantizando su funcionamiento durante todo el día.

Figura 32 : Panel Solar de 20W

Figura 33 : Batería de 7Ah

3.5.4. Pruebas y Programación utilizando Arduino IDE

Con los componentes electrónicos seleccionados, el siguiente paso implica llevar a cabo pruebas de lectura de los sensores. Para llevar a cabo estas pruebas con precisión, se hace uso inicialmente de las librerías especializadas que desempeñan un papel fundamental al facilitar la lectura e interpretación de las señales procedentes de los sensores.

En el caso específico del Arduino Nano, que previamente se mencionó como responsable de la lectura de ciertos sensores, su labor se centra en capturar y procesar las señales de varios componentes clave. Estos componentes incluyen un magnetómetro para la medición de la dirección del

viento, un sensor de luminosidad BH1750, un sensor de presión BMP180 y sensores de efecto Hall que cumplen un papel vital en la medición de la precipitación y la velocidad del viento a través del pluviómetro y el anemómetro.

En cuanto al ESP8266, se requiere llevar a cabo la instalación de una librería específica. Esta acción es necesaria para habilitar la programación del microcontrolador mediante el entorno del Arduino IDE. Asimismo, es fundamental destacar que este microcontrolador desempeñará un rol central en la recolección de datos de diversas variables, incluyendo la medición de la humedad y temperatura ambiental, así como la humedad y temperatura del suelo, radiación y presión atmosférica. Además, para optimizar su funcionalidad, se empleará otra librería que facilitará la operación del multiplexor analógico conectado al ESP8266.

Puesto que las librerías han sido importadas con éxito, el siguiente paso consistió en abordar la elaboración del código principal destinado a cada uno de los microcontroladores.

Dentro del código destinado al Arduino Nano, se inicia con la definición de objetos relacionados a las librerías que previamente se han incorporado. Posteriormente, se procede a la creación de variables de control, cuyo propósito radica en gestionar las temporizaciones y determinar el momento oportuno para ejecutar acciones específicas dentro del código. Además, se establecen variables destinadas a almacenar las lecturas provenientes de los sensores. Entre estas, se destaca una variable de frecuencia, la cual se revela fundamental en el cálculo de la velocidad del viento con relación al anemómetro.

Siguiendo la metodología estipulada por Arduino, se procede con la creación de una función de inicio en la cual se llevarán a cabo múltiples tareas clave. En este contexto, se establece la inicialización de la comunicación serial, seguida de la ejecución de una función diseñada específicamente para configurar los pines de lectura dedicados a los sensores, además de las librerías y objetos previamente declarados.

Como se observa en el **Anexo 7**, se inicia la comunicación I2C, se cargan las librerías previamente importadas y se configuran los pines 6 y 5, destinados

a los sensores de efecto Hall ubicados en el pluviómetro y el anemómetro, respectivamente. Cabe mencionar que se lleva a cabo una breve verificación para asegurarse de que el sensor de presión funcione de manera óptima.

Como resultado de esta fase de inicio, se genera un mensaje de confirmación que se visualiza en el monitor serial, indicando que la inicialización ha transcurrido exitosamente.

Continuando con la programación en Arduino, llegamos a la función de bucle que desempeña un papel fundamental en el proceso. Aquí, se emplea la función 'milis' para registrar el tiempo de ejecución del programa, lo que permite una gestión temporal precisa.

En esta sección, se encuentra otra función esencial encargada de la lectura de los sensores de efecto Hall asociados al pluviómetro y al anemómetro. Esta función se ejecuta de forma constante en el bucle principal del programa. En el caso del pluviómetro, se configura para incrementar la variable "precipitación" en 1 unidad por cada flanco de bajada detectado. Por su parte, para el anemómetro, se recurre a una de las librerías previamente importadas para obtener la frecuencia y, en consecuencia, la velocidad del viento.

Posteriormente, se encuentra la función encargada de la lectura de los demás sensores conectados al Arduino. Esta función se activa cada 5 unidades de tiempo desde el inicio de la ejecución del programa. Además, se implementa una estructura condicional que, cada 24 horas, reinicia la variable "precipitación" para comenzar un nuevo conteo.

En resumen, la función de bucle representa una parte esencial de la programación en Arduino, coordinando la adquisición de datos y el cálculo de variables importantes para el monitoreo climático.

Con esto se completaría el código para el arduino imprimiendo las variables en el puerto serial (Véase en el **Anexo 7**).

En cuanto al ESP8266 cuyo código puede verse en el **Anexo 8**, siguiendo un procedimiento análogo al utilizado en Arduino, tras la importación de las librerías pertinentes, se procede a definir los objetos requeridos de dichas librerías. Asimismo, se establecen todas las variables necesarias para la lectura de los sensores, así como para el almacenamiento de los parámetros transmitidos a través del puerto serial desde el Arduino. Por último, se crean

variables adicionales que desempeñarán un papel crucial durante la fase de lectura del puerto serial.

Luego, en el contexto de la función de configuración (setup), se procede a realizar diversas tareas esenciales. En primer lugar, se inicializan la comunicación serial para garantizar un intercambio eficiente de datos. También, se configura la lectura I2C para habilitar la comunicación con ciertos sensores, y establecemos objetos específicos que facilitarán la adquisición de datos provenientes de dichos sensores. Además, se llevó a cabo la configuración de un pin destinado a la lectura analógica, lo cual resulta fundamental para nuestras operaciones de recolección de información

A continuación, dentro de la función de bucle, ejecutamos dos funciones adicionales que desempeñan roles cruciales. La primera función nos permite efectuar la lectura precisa de los sensores que están conectados al ESP8266. Con esta acción, se obtienen los datos relativos a la humedad y temperatura proporcionados por el DHT22, así como el valor analógico inicial que corresponde al sensor de radiación UV. Este último valor será procesado por otras dos funciones, cuya finalidad radica en su formateo y conversión en unidades de radiación según las normativas de registro de datos climáticos. Seguidamente, reconfiguramos el multiplexor para acceder al valor que denota la temperatura del suelo y lo convertimos en grados centígrados. Por último, se procedió a capturar el dato relativo a la humedad del suelo, mapeando los bits recibidos para obtener una cifra en el rango de 0 a 100, de acuerdo con la escala estipulada.

La segunda función, encargada de la lectura de los valores transmitidos a través del puerto serial desde el Arduino, se sirve de la función readBytesUntil. Esta función permite la captura de los datos que llegan al puerto serial, en busca de un carácter específico, limitando la lectura a un número máximo de caracteres. Los valores leídos son almacenados en variables previamente definidas al comienzo del programa. Posteriormente, se procede a la separación de los datos recibidos, con el propósito de asignarlos a variables individuales y convertirlos a los tipos de datos pertinentes. Gracias a este proceso, todas las variables se encuentran debidamente almacenadas y preparadas para su utilización en aplicaciones relacionadas con el Internet de las cosas (IoT).

Luego de completar meticulosamente cada uno de estos procedimientos, se llevó a cabo una evaluación integral. Esta fase incluyó la puesta a prueba simultánea de todos los sensores y el código desarrollado. Para realizar esta evaluación, se suministró alimentación a los microcontroladores mediante conexión USB desde un equipo portátil. Los sensores, a su vez, se conectaron de manera interconectada utilizando cables jumper, estableciendo conexiones con un protoboard y los microcontroladores correspondientes. Esta evaluación integral se realizó con el objetivo de verificar exhaustivamente el funcionamiento y la interacción de todos los componentes del sistema.

Figura 34 : Diagrama de Flujo de la Estación Climática con Arduino Nano y ESP8266

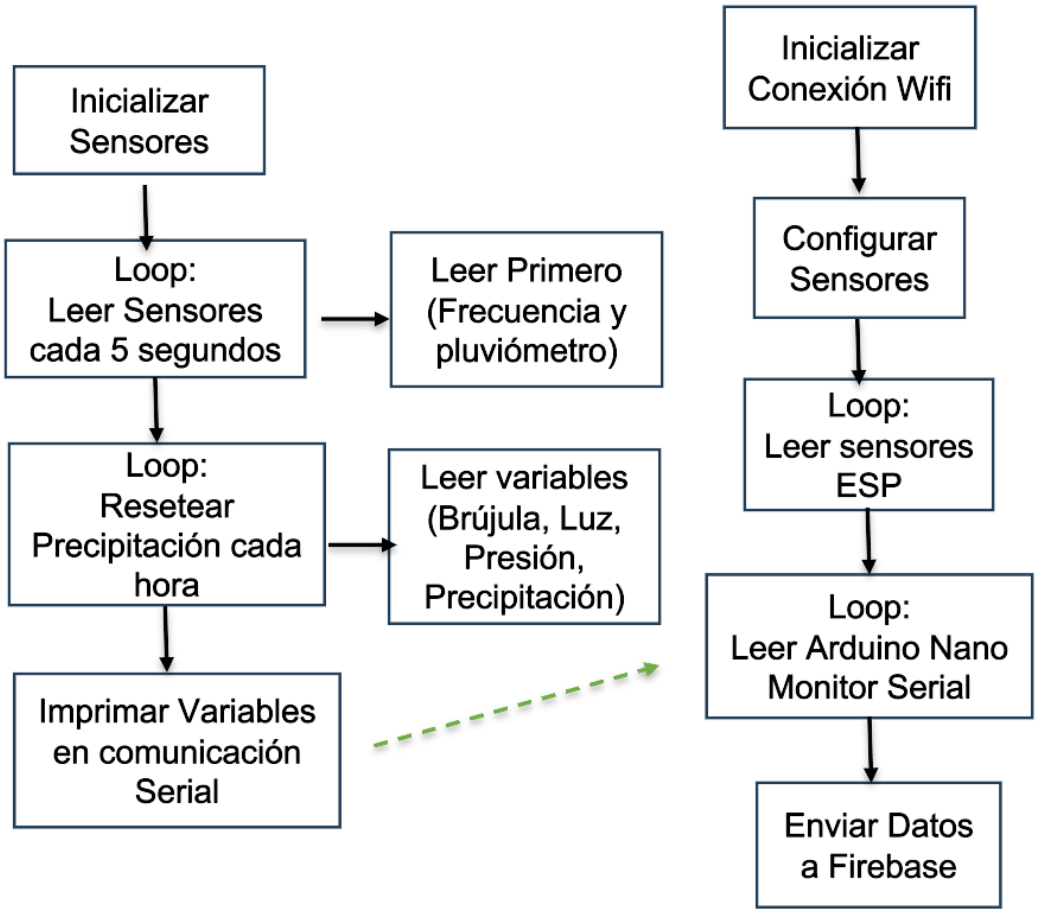
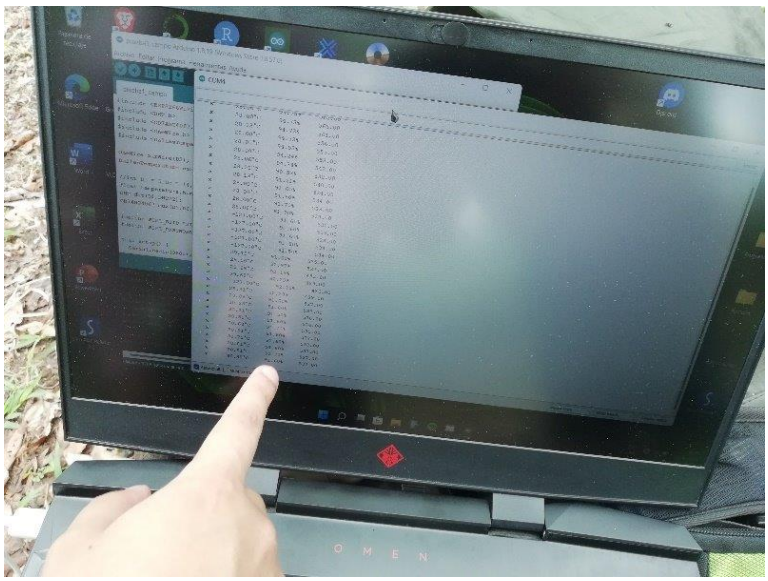


Figura 35 : Primeras pruebas en situación de campo

Figura 36 : Variables medidas visualizadas por monitor serial



3.5.5. Diseño de la placa PCB

Tras haber seleccionado cuidadosamente todos los componentes electrónicos y se ha completado con éxito la programación y validación de las lecturas junto con el código correspondiente, surge la necesidad de desarrollar una placa de circuito impreso (PCB) que acoja y asegure todos estos componentes electrónicos de manera organizada y segura. Esto es especialmente relevante dado que la estación climática estará ubicada en un entorno al aire libre, donde está expuesta a diversas condiciones ambientales. La creación de una PCB también desempeña un papel crucial para prevenir posibles fallos debido a la utilización de conexiones temporales y protoboard.

En el ámbito del diseño de PCB, existe una amplia gama de software comercial especializado diseñado específicamente para llevar a cabo este tipo de tareas. Estas herramientas proporcionan una plataforma integral para la creación, edición y optimización de las placas de circuito impreso, facilitando el proceso de diseño y garantizando un resultado de alta calidad, entre las opciones comerciales se tienen las siguientes:

Tabla 17 : Programas de Diseño de Placas PCB

Software de Pcb	Sistema operativo	Licencia	Capacidad multiplataforma	Comunidad / Soporte	Facilidad de Uso	Características avanzadas
Altium designer	Windows	Comercial	No	Buena	1 pa	Amplia
Kicad	Windows, macos, Linux	Código abierto	Si	Comunidad fuerte	0.1 pa	Amplia
Autodesk eagle	Windows, macos, linux	Comercial	Si	Buena	0.01 pa	Moderada
Orcad	Windows	Comercial	No	Buena	0.001 pa	Amplia
Easyeda	Basado en web	Gratuito con opciones de pago	Si	Comunidad emergente	1 pa	Limitada

De todos estos softwares se decidió utilizar Autodesk Eagle puesto que presenta grandes ventajas tales como el tener una gran biblioteca de componentes que incluye símbolos esquemáticos, modelos de huellas de PCB y 3D para una variedad de componentes electrónicos. Esto facilita la creación de diseños y reduce el tiempo necesario para crear símbolos y huellas personalizadas. El tener una interfaz de usuario intuitiva y su curva de aprendizaje relativamente suave. Esto lo convierte en una de las mejores opciones para esta clase de proyectos. Eagle tiene una comunidad activa de usuarios y una base de conocimientos sólida. Esto significa que es más probable encontrar tutoriales, consejos y soluciones a problemas comunes en línea. Eagle admite diseños de PCB multicapa, lo que lo hace adecuado para proyectos más complejos y avanzados. Autodesk Eagle ofrece una licencia gratuita que permite a los usuarios trabajar en proyectos pequeños y medianos sin costo. Esto es beneficioso para que estudiantes puedan utilizarlos en proyectos académicos. Además, este software ya ha sido utilizado en algunos proyectos para la universidad por lo que debido a todas estas razones se considera la mejor opción para el desarrollo de la placa PCB de la estación climática.

Para empezar a diseñar la placa PCB se eligió un grosor de ruta de 1mm debido a que permite una alta capacidad de manejo de corriente, este tipo de rutas reducen la resistencia y la caída de voltaje, lo que ayuda a evitar problemas de calentamiento y pérdida de voltaje. Además, estas pistas son menos propensas a dañarse debido a corrientes excesivas o fluctuaciones de temperatura. Esto mejora la robustez y la vida útil de la PCB en este tipo de aplicaciones en donde estará sometida a diferentes cambios de clima y a condiciones desafiantes, también al no ser tan pequeña facilita el proceso de soldadura y de fabricación de la placa PCB, además este grosor de pista también ayuda a la disipación de calor para ayudar a componentes como un regulador de voltaje. Tal como se ve es lo suficientemente ancho para garantizar un buen grosor de cobre y una resistencia eléctrica baja, así también como lo suficientemente estrecho como para no ocupar demasiado espacio y esto permite que toda la parte electrónica de la estación climática sea más compacta y ocupe el menor espacio.

Debido a este último aspecto también se ha elegido diseñar la placa con 2 capas de pistas una superior y una inferior para hace aún más compacta la

placa, estas tendrán que comunicarse entre sí para proporcionar el flujo eléctrico a todo el circuito, con esto en mente lo siguiente es elegir el material de fabricación de la placa PCB. Para esto se eligió los materiales de fibra de vidrio y agujeros metalizados que se usan para diseños más profesionales y robustos, la fibra de vidrio nos dará una alta resistencia y durabilidad en la placa, además de eficiencia eléctrica para las pistas, y los agujeros metalizados nos proporcionarán la conexión entre ambas capas diseñas por defecto, además que nos brindarán facilidad en la soldadura de los componentes, con ello en mente se empezó a elaborar la placa PCB.

Dentro del diseño de la placa se empezó por poner en primer lugar al regulador de voltaje, dado que este tiene que ser de fácil acceso para poder conectar los cables de alimentación. Después se tiene la distribución de los componentes electrónicos más grandes con los que cuenta la estación climática, que son los microcontroladores y el multiplexor analógico, en este caso se puede observar primero al arduino nano, luego al ESP8266 y finalmente el multiplexor CD74HC4067, después en la parte inferior tenemos la distribución de las entradas para los sensores, estos fueron distribuidos de acuerdo a los microcontroladores a los que se conectarán, en caso de los sensores que se conectarán al arduino están en la parte derecha y los que se conectarán al ESP8266 se encuentran en la parte izquierda, igual los que irán hacia el multiplexor analógico, además se agregaron una resistencia para que nos ayude a la lectura del sensor de humedad y temperatura del aire, y en el sensor para la lectura de humedad del suelo.

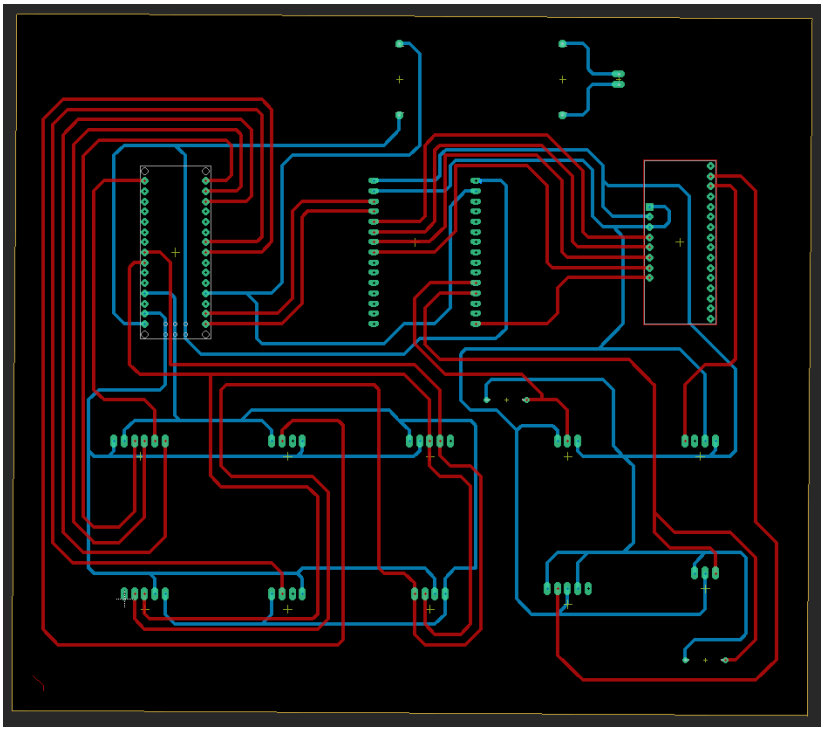
Dentro de este diseño se utilizó la capa superior representada en rojo para la conexión entre los sensores y los microcontroladores en la mayoría de los casos sin tener en cuenta la alimentación de estos, y la capa inferior de la placa se utilizó la pista para la alimentación de todo el circuito partiendo desde el regulador de voltaje, los microcontroladores y los sensores, además también se utilizó para completar algunas rutas que no fueron ser completadas en la capa superior.

Otro de los aspectos a destacar en este diseño de la placa son los ángulos utilizados en las pistas, los cuales son de 45°, siguiendo algunos estándares de diseño de PCB, como IPC-2221B (anteriormente conocido como IPC-D-275), como una buena práctica de diseño para mejorar la integridad de la señal y

reducir la EMI. Este valor de ángulo en diseño de placas PCB es muy utilizado por que aporta grandes beneficios, tales como:

- Menos reflexiones de señal: Las curvas de 45 grados reducen la posibilidad de reflexiones de señal en comparación con las curvas de 90 grados. Las reflexiones de señal causan interferencia y degradación de la calidad de la señal en líneas de alta velocidad y frecuencia. Con ángulos de 45 grados, las transiciones entre segmentos de pista son más suaves, lo que minimiza las reflexiones.
- Mejora de la integridad de la señal: Las curvas de 45 grados ayudan a mantener una integridad de señal más sólida, especialmente en señales de alta velocidad, como buses de datos y señales de reloj. Esto es fundamental para garantizar que los datos se transmitan y reciban de manera confiable sin errores.
- Menos interferencia electromagnética (EMI): Las curvas suaves reducen la probabilidad de generar campos electromagnéticos no deseados. Esto ayuda a mitigar problemas de EMI, lo que es especialmente importante en aplicaciones sensibles a la interferencia electromagnética.
- Optimización del espacio: Los ángulos de 45 grados permiten un uso eficiente del espacio en el diseño de PCB. Esto es importante cuando se trabaja en placas de circuito impreso de alta densidad donde el espacio es limitado.
- Facilita el enrutamiento: En muchos casos, las curvas de 45 grados son más fáciles de enrutar y ayudan a simplificar el diseño general de la PCB. Esto acelera el proceso de diseño y reducir la complejidad.

Figura 37 : Diseño en Eagle de la Placa PCB



Después de haber culminado el proceso de diseño de la placa PCB, se procedió a avanzar en la fase de fabricación de dicha placa. Para lograrlo, se envió el diseño previamente elaborado junto con las especificaciones y elecciones de materiales pertinentes a una empresa de impresión especializada. Una vez en posesión de la placa impresa, se dio inicio al proceso de soldadura de los componentes electrónicos seleccionados. En esta etapa, se optó por utilizar conectores hembra tipo espárrago, los cuales fueron soldados en la placa para facilitar la conexión de todos los componentes. Esta elección se hizo con la finalidad de permitir una manipulación sencilla de los componentes y, en caso de ser necesario, reemplazar cualquiera de ellos sin enfrentar dificultades significativas en caso de fallos o actualizaciones. Con la culminación de esta fase, se procedió a realizar una rigurosa validación para garantizar el correcto funcionamiento de la placa y sus componentes. Esta validación se realizó tomando datos locales utilizando una computadora.

Figura 38 : Placa Impresa

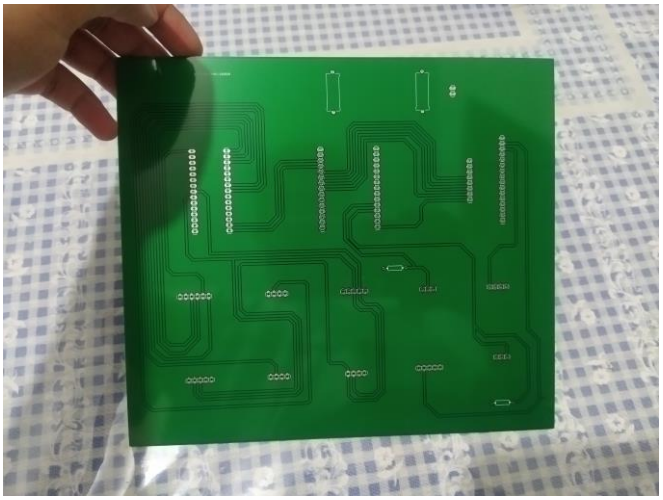


Figura 39 : Prueba de Funcionamiento de la placa PCB

3.5.6. Diseño en solidworks de anemómetro, veleta y pluviómetro

Para dar inicio al proceso de diseño de las piezas que serán fabricadas mediante impresión 3D, se procedió con la crucial etapa de seleccionar el software de diseño asistido por computadora (CAD, por sus siglas en inglés) que sería empleado para llevar a cabo este trabajo. Entre las diversas alternativas

disponibles en el mercado, se consideraron las siguientes opciones de software comercial:

Tabla 18 : Software de Diseño CAD

Software	Tipo	Facilidad de Uso	Funciones avanzadas	Plataformas	Precio
Autodesk fusion 360	Comercial	Intermedia	Amplias	Windows, macos, web	Suscripción mensual o anual
Solidworks	Comercial	Avanzada	Muy amplias	Windows	Licencia anual
Tinkercad	Gratuito	Fácil	Básicas	Navegador web	Gratuito
Blender	Gratuito	Avanzada	Muy amplias	Windows, macos, linux	Gratuito
Freecad	Gratuito	Intermedia	Amplias	Windows, macos, linux	Gratuito

Dentro de todas estas opciones se escogió Solidworks, debido a que es uno de los softwares más robustos para este tipo de tareas, cuenta con una amplia variedad de herramientas que facilitan el diseño, tiene una interfaz bastante intuitiva a pesar de la robustez del programa, y como es un software utilizado en algunos cursos dentro de la universidad ya existe familiaridad con este software.

Después elegido el software CAD se procedió a modelar las diferentes partes que tendrán cada una de las piezas, tomando como inspiración los diseños realizados por marcas comerciales.

En cuanto a la estructura física del anemómetro y la veleta, ambos comparten un diseño básico. Comienzan con una base que se asegura a la estructura de la estación meteorológica y proporciona una vía para los cables de los sensores ubicados en su interior. A continuación, se encuentra el cuerpo principal, que tiene forma de botella y ofrece suficiente espacio para albergar los sensores necesarios. También incluye una pieza cilíndrica móvil, que aloja los sensores del anemómetro y la veleta.

Dentro del cuerpo del anemómetro, se ubica un imán de neodimio fijo, que genera las señales necesarias para medir las revoluciones del anemómetro. Esta

pieza cilíndrica móvil se sujeta mediante un soporte en la parte superior que tiene una estructura similar a un paraguas. Para asegurar un movimiento suave y minimizar la fricción en estas partes móviles, se incorporan rodamientos axiales, lo que contribuye a obtener mediciones más precisas y fiables.

Por último, en la parte superior de cada instrumento, encontramos las características específicas de cada uno. En el caso del anemómetro, se utiliza un diseño de cazoletas en forma de copas, que sigue las medidas estándar de la industria. En cambio, la veleta se distingue por su estructura fina y una gran área expuesta al viento, lo que facilita la detección de la dirección del viento.

Figura 40 : Diseño del Anemómetro en SolidWorks

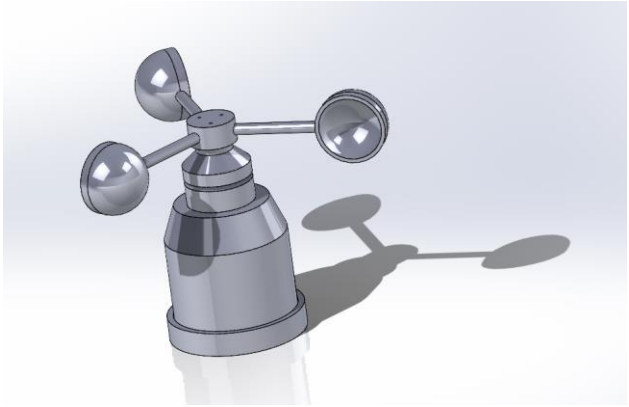


Figura 41 : Vista en corte del anemómetro en SolidWorks

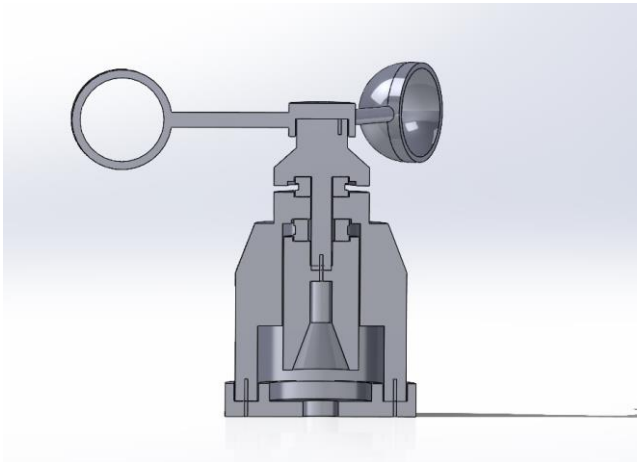


Figura 42 : Vista explosionada del anemómetro en Solidworks

Figura 43 : Diseño de la Veleta en SolidWorks

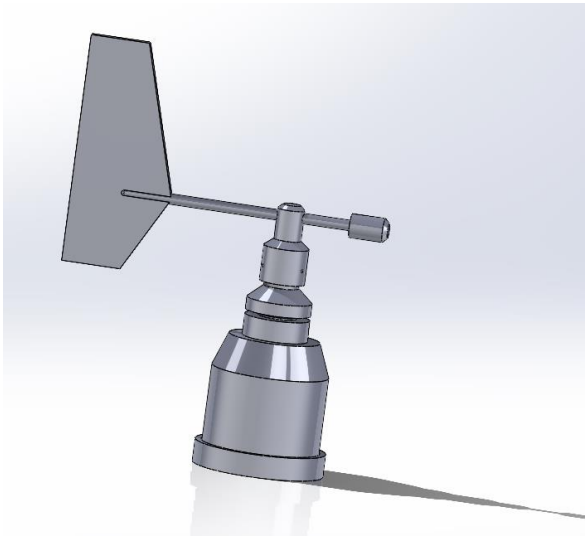


Figura 44 : Vista en Corte de la Veleta en SolidWorks

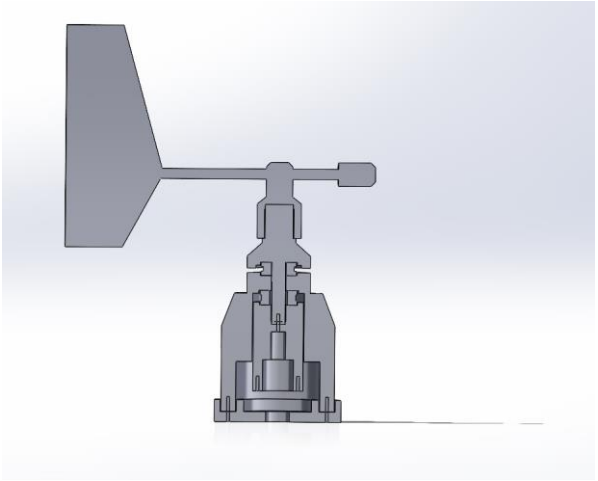
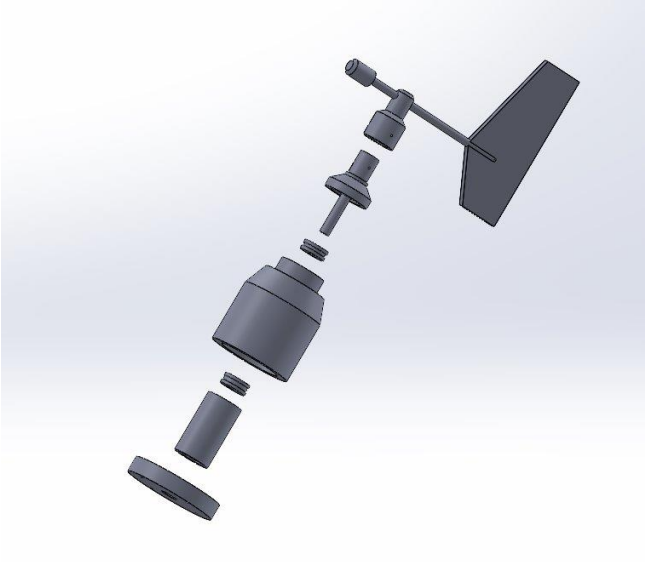


Figura 45 : Vista explosionada de la Veleta en Solidworks



Una vez que se completó el ensamblaje de todos los componentes, se llevó a cabo una validación de los esfuerzos tanto en el anemómetro como en la veleta utilizando la función de simulación estática de SolidWorks. Se configuraron todos los materiales, tanto fijos como móviles, junto con sus propiedades, y se incorporaron dos rodamientos en cada caso, simulando además la influencia de la fuerza de la gravedad. Los resultados indicaron que ambos diseños son consistentes, mostrando solo esfuerzos leves en las secciones más delgadas, los cuales se mantienen muy por debajo de su límite elástico.

Figura 46 : Simulación de esfuerzos del anemómetro en Solidworks

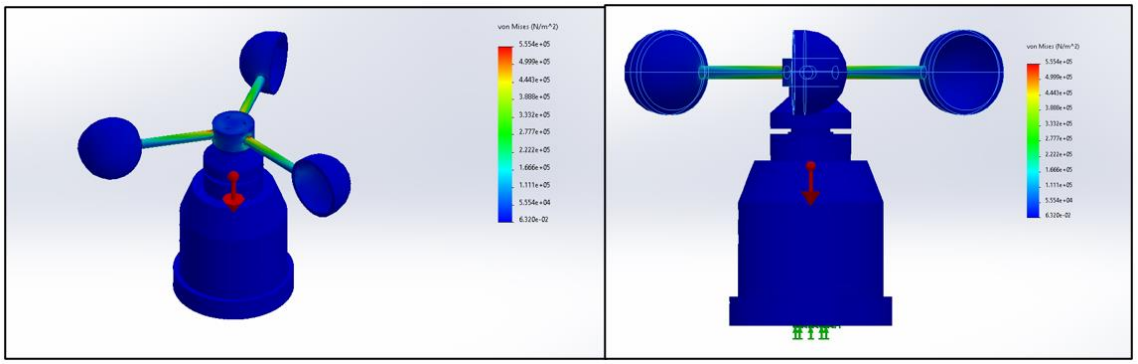
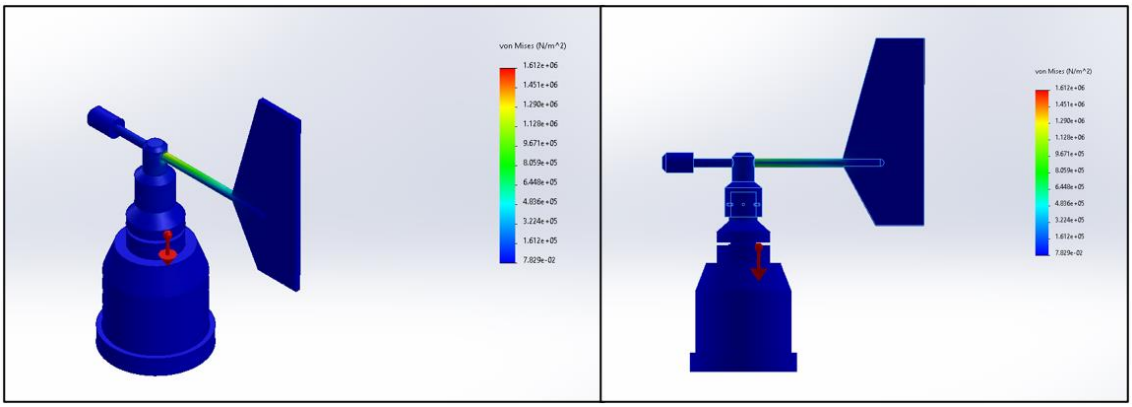


Figura 47 : Simulación de esfuerzos de la veleta en Solidworks



También se llevó a cabo una simulación dinámica tanto del anemómetro como de la veleta con el objetivo de analizar su respuesta ante la influencia del viento. Durante la simulación, se aplicó una fuerza transversal uniforme para representar las condiciones del viento, lo que permitió observar que ambos componentes reaccionaron de manera adecuada. Específicamente, se notó que el anemómetro mantuvo una velocidad angular constante, mientras que la veleta demostró una velocidad angular casi constante. Este hallazgo implica que el diseño de los dispositivos es robusto y capaz de mantener un rendimiento estable incluso bajo la influencia de condiciones variables del viento.

Figura 48 : Simulación de respuesta del anemómetro en presencia de viento

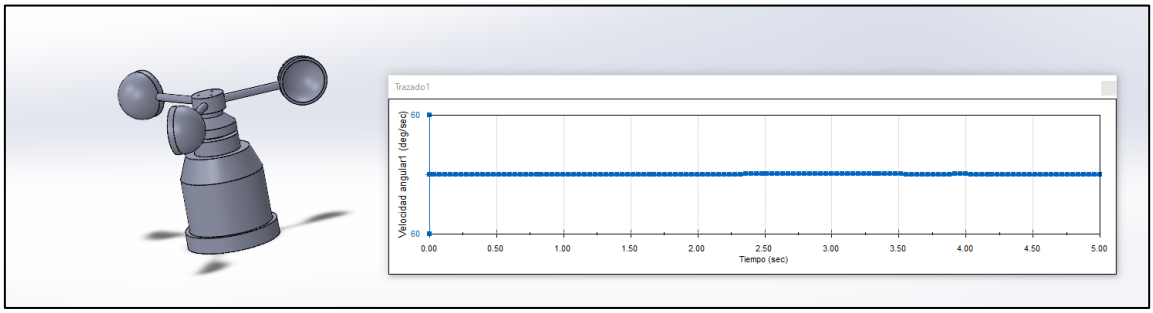
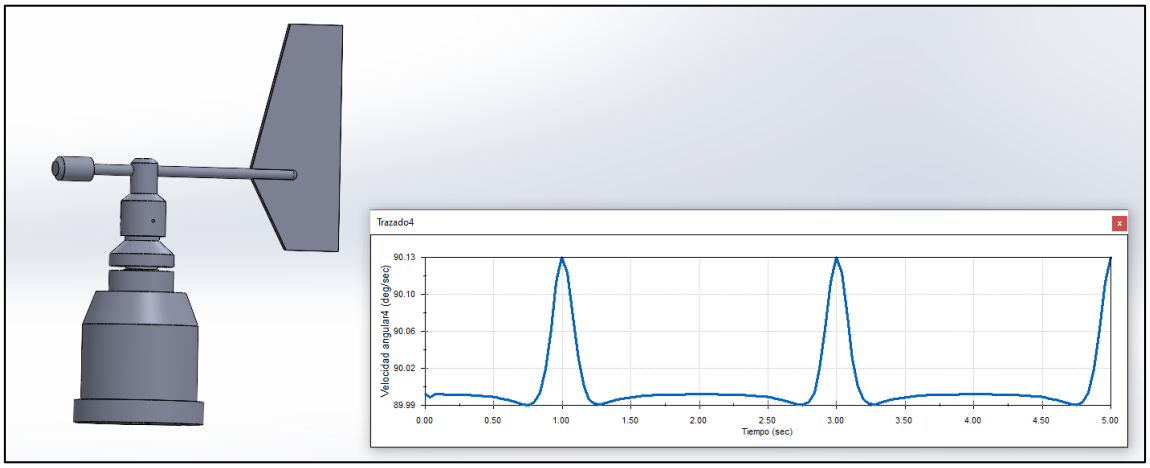


Figura 49 : Simulación de respuesta de la veleta en presencia de viento



En lo que respecta a las piezas del pluviómetro, estas se han segmentado en dos partes esenciales. La primera corresponde a la parte superior, diseñada en forma de un embudo que actúa como la entrada para el agua. En este diseño, se han tomado como referencia medidas convencionales utilizadas por fabricantes comerciales. Aunque no existe una norma estandarizada para el diámetro de este embudo, se ha procurado seleccionar medidas que sean comúnmente utilizadas y que han demostrado ser eficaces en la práctica. Además, se ha considerado la posibilidad de realizar conversiones matemáticas para generalizar estas medidas si fuese necesario.

La segunda parte esencial del pluviómetro es el mecanismo de balancín. En este componente, se observa una pieza de doble cubeta basculante, que se alojará en una base compuesta por dos piezas unidas mediante pernos. Asimismo, se incluirá una pieza cilíndrica que se encargará de proporcionar el movimiento necesario a la doble cubeta basculante. Esta pieza cilíndrica estará conectada a las dos partes de la base, donde se instalarán dos rodamientos radiales en los extremos para minimizar los efectos de la fricción.

Es importante destacar que, en este diseño, se han concentrado principalmente en la parte superior y el mecanismo de la cubeta basculante. Esto se debe a que es posible utilizar recipientes estándar y adaptar estos mecanismos para lograr un funcionamiento en conjunto eficiente y preciso.

Con ello ya se tienen listas todas las piezas para ser impresas, donde también se tienen la confirmación del ensamblaje de todas las piezas en solidworks.

Figura 50 : Diseño de Doble Cubeta Basculante en SolidWorks

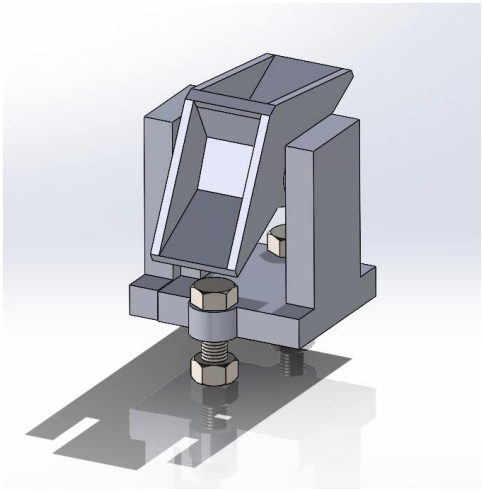
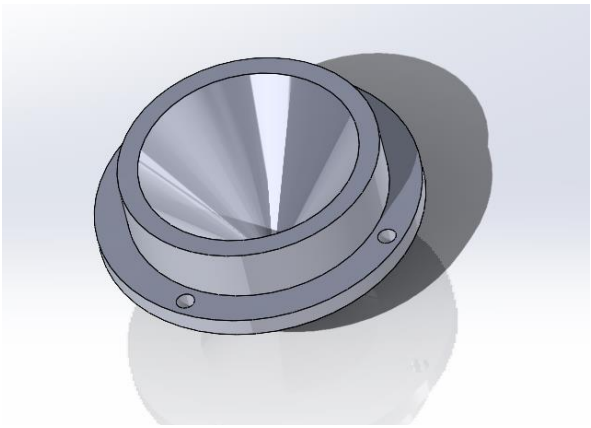


Figura 51 : Parte superior de Pluviómetro en SolidWorks



3.5.7. Impresión y ensamblaje del anemómetro, veleta y pluviómetro

Primero para la impresión es necesario elegir el mejor material para las condiciones a las que estarán expuestas estas piezas, que son condiciones de campo con clima bastante cambiante y expuesto a diferentes cambios de

humedad. Teniendo en cuenta esto se analizaron los materiales existentes en los que se pueden realizar este tipo de impresiones:

Tabla 19 : Materiales de Impresión 3D

Material	Pla	Abs	Petg	Asa	Tpu
Resistencia al Clima	Buena	Buena	Excelente	Excelente	Buena
Durabilidad	Buena	Buena	Excelente	Excelente	Buena
Resistencia al Agua	Media	Baja	Excelente	Excelente	Excelente
Flexibilidad	Baja	Baja	Baja	Baja	Alta
Facilidad de Impresión	Fácil	Moderada	Moderada	Moderada	Moderada
Tolerancia a Temperatura	Baja	Moderada	Alta	Alta	Moderada
Resistencia química	Sensible	Sensible	Buena	Buena	Sensible
Biodegradable	Si	No	No	No	No

Teniendo en cuenta los materiales descritos se decidió utilizar PLA debido a la facilidad de impresión que tiene este material para las piezas que se van a utilizar, es biodegradable lo cual hace este tipo de impresión amigable con el medio ambiente, también es un material de bajo costo que cuenta con una buena resistencia para aplicaciones al exterior debido a que resiste a la degradación por UV y es un buen aislante térmico.

Con esto ya definido se pasó al proceso de impresión de las empiezas y debido al material elegido y al tamaño de las piezas no demoró demasiado tiempo. Terminado el proceso de impresión y teniendo ya las piezas listas ahora se procede al ensamblaje de los mecanismos.

Para el anemómetro y la veleta se presentó un error de diseño en las tolerancias de la pieza cilíndrica del interior de los mecanismos debido a que no contaban con la tolerancia necesaria, para solucionar esto no fue necesario volver a imprimir las piezas sino lijar las piezas hasta que lleguen a una tolerancia aceptable y que no afecte el movimiento de las piezas.

Como se mencionó previamente, el anemómetro estará equipado con un sensor de efecto Hall, que responderá a la influencia de un pequeño imán de neodimio que permanecerá inmóvil. La parte que será móvil en este sistema será

el propio sensor de efecto Hall. En el caso de la veleta, el montaje seguirá un patrón similar, y el magnetómetro que se empleará se ubicará en la base de la pieza cilíndrica interna, asegurado de manera fija para rastrear el movimiento de la veleta.

Figura 52 : Cuerpo Impreso del Anemómetro y la Veleta

Figura 53 : Parte Superior del Anemómetro

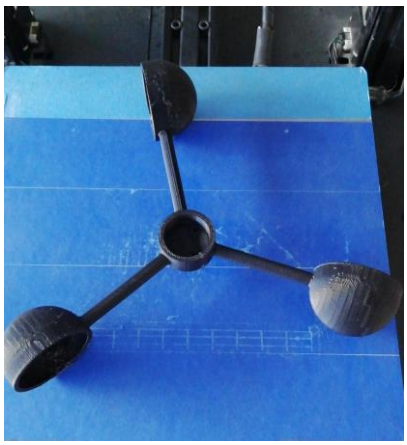


Figura 54 : Parte Superior de la Veleta

Figura 55 : Piezas Impresas de la Veleta

Figura 56 : Piezas Impresas del Anemómetro

En cuanto al pluviómetro, las diversas piezas se ensamblaron mediante pernos y tuercas de 3/4 a través de los orificios previamente diseñados. Este mecanismo también utilizará un sensor de efecto Hall; en este caso, el sensor estará anclado a un lado del balancín, y será activado por un imán de neodimio que se moverá de forma conjunta con el balancín. Esta configuración permitirá contar la cantidad de lluvia recolectada con precisión. En base a las dimensiones del pluviómetro construido se calculó la cantidad de “mm” de lluvia mínima que detecta el pluviómetro esta unidad viene dada por la relación del volumen de agua en litros que cae sobre un área expresada en metros cuadrados.

$$mm\ de\ lluvia\ = \frac{Litros}{metros^2}$$

Para el caso de este pluviómetro se tiene un área circular de 54mm de radio y el volumen de agua que se necesita para generar la señal de cambio en sistema de cubeta basculante se halló de manera experimental con pruebas locales puesto que para que la cubeta gire no llena todo el espacio disponible y se halló que este volumen es de 0.006 litros. Con estos datos se calcula la cantidad de precipitación que se registra por cada señal que envía el sensor de efecto hall.

$$mm\ de\ lluvia\ = \frac{0.006}{(0.054)^2 x\ \pi}$$

$$mm\ de\ lluvia\ = 0.6586\ mm$$

Figura 57 : Piezas Impresas del Pluviómetro

Figura 58 : Sistema de doble Cubeta Basculante del Pluviómetro

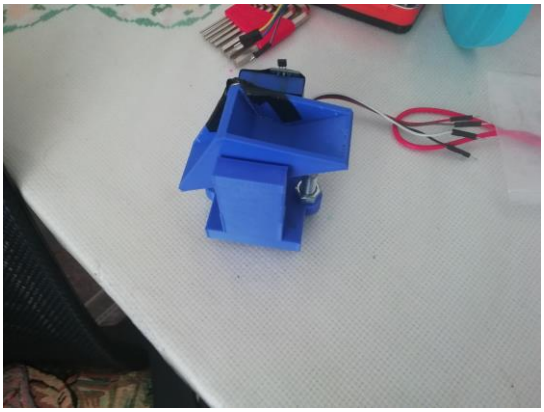


Figura 59 : Prueba Experimental del mecanismo del Pluviómetro

3.5.8. Conexión a Internet para la aplicación IoT

Como se mencionó anteriormente, las pruebas de campo se llevaron a cabo en una plantación de cacao que, al igual que muchas otras, carece de suministro eléctrico y, por lo tanto, no tiene acceso a Internet. Ante esta circunstancia, se ha emprendido un análisis detallado de las opciones disponibles para habilitar la conectividad a Internet, una necesidad fundamental para la implementación de la aplicación IoT en la estación climática, que representa el núcleo de esta investigación.

En las zonas rurales, a pesar de la ausencia de una infraestructura fija de Internet, suele estar disponible la conexión móvil a través de las redes de telefonía. Por lo tanto, una solución viable para acceder a Internet en este contexto es el uso de un módem portátil que sea compatible con la compañía de telefonía que brinde la mejor cobertura en la ubicación de la plantación. Esta elección estratégica permite que la estación climática pueda funcionar en diferentes ubicaciones, simplemente sustituyendo el chip de conexión a Internet según la región en la que se encuentre.

Figura 60 : Módem WiFi Móvil Huawei

En consonancia con esta estrategia, el suministro de energía para el módem también se gestionará mediante la batería y el panel solar. Para lograrlo, se ha calculado el consumo eléctrico del módem, que se estima en 5W, de acuerdo con el modelo específico utilizado. Esto significa que no afectará la elección de los componentes elegidos previamente debido a que el consumo energético es pequeño.

3.5.9. Problemas de hardware utilizando tarjeta SD y descarte

En la fase inicial del diseño, se contempló la incorporación de una tarjeta microSD como una medida de respaldo para los datos que se envían a la nube, especialmente en situaciones en las que la conexión pudiera fallar. Sin embargo, se encontró un desafío significativo debido a que el ESP8266 carece de suficientes direcciones de memoria para garantizar un almacenamiento adecuado de los datos en la tarjeta microSD cuando el circuito permanece en funcionamiento durante largos períodos de tiempo.

3.5.10. Programación de la plataforma IoT

La razón fundamental detrás de la decisión de programar una plataforma IoT personalizada para la estación climática en lugar de recurrir a una de las numerosas plataformas preexistentes radica en el deseo de ejercer un mayor grado de control sobre el contenido y la disponibilidad de la plataforma. Este enfoque de desarrollo se presenta como una alternativa específica, orientada al cultivo de cacao, en lugar de simplemente adaptarse a las soluciones ya existentes.

Para iniciar el desarrollo de la plataforma IoT destinada a la estación climática automática, el proceso inicial se inicia definiendo las necesidades concretas, explorando las tecnologías disponibles para satisfacer dichas necesidades y seleccionando las opciones más adecuadas para este contexto particular. A partir de ahí, se elabora la arquitectura de software y se traza el diagrama de flujo que guiará el proyecto en su conjunto.

Esta plataforma IoT tiene como objetivo principal ofrecer una visualización en tiempo real de todas las variables climáticas recopiladas por la estación climática. Además, debe posibilitar el acceso a datos históricos en un formato diario y permitir la descarga de esta información, facilitando así su análisis por parte de especialistas encargados de estudiar la adaptabilidad climática en el cultivo de cacao. Otra meta esencial es garantizar la accesibilidad de esta plataforma, de modo que pueda ser utilizada por una amplia gama de dispositivos conectados a Internet.

Para poder cubrir estas necesidades de la plataforma IoT, se requiere un entorno de desarrollo que pueda ser compilado para ser visualizado en múltiples plataformas, además de una base de datos para el almacenamiento de las variables medidas por la estación climática para luego ser presentadas dentro de la plataforma.

Dentro de las opciones para la primera necesidad tenemos las siguientes opciones:

Tabla 20 : Tipos de Aplicaciones

Caracteristicas	Aplicaciones multiplataformas	Aplicaciones web progresivas (Pwa)
Soporte multiplataforma	Compatible con múltiples plataformas como ios, android, y a veces también escritorio	Principalmente basadas en tecnologías web y ejecutadas en navegadores modernos pueden ser accesibles desde cualquier dispositivo con un navegado
Acceso sin Descargas	Requiere la instalación de un marco de trabajo o un entorno específico, lo que implica descargas para los usuarios.	No requieren descargas ni instalaciones; se ejecutan directamente en el navegador web del usuario
Experiencia de Usuario Nativa	Suelen ofrecer una experiencia de usuario más cercana a la nativa, con acceso a características del dispositivo.	Proporcionan una experiencia de usuario ismilar a la de una aplicación nativa, incluyendo notificaciones y acceso a hardware, através de api web.
Desarrollo más Complejo	El desarrollo puede ser más complejo debido a la necesidad de adaptar la aplicación para diferentes plataformas.	Más sencillo en términos de desarrollo ya que utiliza tecnologías web estándar y un único código base
Actualizaciones más Rápidas	Las actualizaciones pueden requerir aprobación de tiendas de aplicaciones, lo que puede retrasar la disponibilidad de nuevas características.	Las actualizaciones pueden implementarse de manera inmediata en el servidor y estarán disponibles para los usuarios la próxima vez que carguen la pwa.
Uso de Recursos del Dispositivo	Puede acceder a más recuros del dispositivo, como cámara, sensores y almacenamiento local.	Tiene acceso limitado a recursos del dispositivo, lo que puede ser más seguro en términos de privacidad.
Costos de Desarrollo	Puede implicar costos más altos de desarrollo debido a la necesidad de crear y mantener múltiples versiones de la aplicación.	Suelen ser más económicas de desarrollar y mantener, puesto que comparten un código base y aprovechan la infraestructura web existente.
Actualizaciones sin Friccion	Las actualizaciones pueden requerir que los usuarios descarguen e instalen nuevas versiones de la aplicación.	Las actualizaciones se realizan automáticamente en segundo plano, lo que proporciona una experiencia sin fricción para los usuarios.

Con el fin de hacer más simple el desarrollo y de poder ejecutar la plataforma en cualquier dispositivo mediante internet se eligió desarrollar la plataforma como una aplicación web progresiva, esto también tiene la ventaja de que no solo funciona como una aplicación web, sino que también puede instalarse en dispositivos móviles y ejecutarse como si fuera una aplicación nativa para móviles, esto hace que el alcance de la plataforma cubra todos los dispositivos electrónicos. Para el desarrollo de aplicaciones web progresivas existen diferentes frameworks que permiten crear estas aplicaciones en una gran variedad de lenguajes de programación, para este caso se eligió utilizar el framework Django el cual utiliza Python como lenguaje de programación, Django es un framework completo que proporciona una amplia gama de características integradas, como un sistema de autenticación, un sistema de administración, un ORM (Object-Relational Mapping), y más. Esto acelera el desarrollo y permite concentrarse en la lógica de negocio en lugar de construir componentes comunes desde cero. Además de ser compatible con múltiples sistemas de bases de datos, como PostgreSQL, MySQL, SQLite y otros. Todos estos aspectos hacen de Django una elección sólida para el desarrollo de la plataforma IoT debido a su versatilidad, seguridad, escalabilidad y conjunto integral de características. Además, su enfoque en la eficiencia y la calidad del código ayuda a mantener un proyecto de IoT bien organizado y mantenible a lo largo del tiempo.

Como hemos visto esta propuesta proporciona una buena integración con distintas bases de datos, entrando en el punto de la elección de la base de datos se decidió separar en 2, una base de datos que será la encargada de todo el manejo de los datos en tiempo real y otra base de datos más robusta que será la encargada de almacenar los datos históricos medidos.

Dentro de las opciones para la base de datos en tiempo real se tienen:

Tabla 21 : Base de Datos en Tiempo Real

Caracteristicas	Firebase	Dynamodb	Mongodb	Redis
Modelo de Datos	Json en tiempo real y NoSQL.	NoSQL y json flexible.	Documentos bson (formato binario json).	Estructura de datos clave-valor.
Escalabilidad horizontal	Sí, escalabilidad automática y global.	Sí, escalabilidad horizontal mediante particionamiento.	Sí, escalabilidad horizontal y distribución de datos.	Sí, clustering y particionamiento
Velocidad y latencia	Alta velocidad y baja latencia.	Baja latencia y alta velocidad de lectura / escritura.	Latencia baja a moderada, dependiendo de la configuración.	Muy baja latencia para operaciones en memoria.
Soporte para Datos en Tiempo Real	Soporte nativo para datos en tiempo real.	Limitado soporte en tiempo real a través de 90streams y websockets.	Puede implementarse para datos en tiempo real, pero no es su enfoque principal.	Soporte completo para datos en tiempo real con pub/sub.
Consistencia de Datos	Modelo de datos en tiempo real altamente consistente.	Modelo de datos altamente consistente.	Modelo de datos consistente eventualmente.	Modelo de datos altamente consistente en memoria.
Búsquedas y consultas	Búsquedas simples y filtrado en tiempo real.	Búsquedas y filtrados avanzados con consultas sql-like.	Consultas complejas y búsqueda flexible.	Búsqueda y filtrado sencillo.
Uso principal	Aplicaciones móviles, web y iot con datos en tiempo real.	Aplicaciones web y móviles con requerimientos de escalabilidad.	Aplicaciones web y móviles con necesidades de flexibilidad en el esquema.	Caché en memoria y aplicaciones que requieren alta velocidad
Integracion con Plataformas	Integración directa con otros productos de google cloud.	Puede integrarse con servicios de aws y aplicaciones de terceros.	Se integra con muchas bibliotecas y marcos web.	Amplias bibliotecas y soporte de cliente
Precio	Planes de precios flexibles, incluyendo un nivel gratuito.	Escalonamiento de precios con tarifas por capacidad y uso	Varias opciones de implementación, algunos con costo.	Precios basados en la memoria y almacenamiento utilizado.

Dentro de todas estas opciones se eligió Firebase debido a su facilidad de uso y su curva de aprendizaje suave. Firebase ofrece sincronización en tiempo real de datos, lo que significa que los cambios realizados en la base de datos se reflejan instantáneamente en todos los dispositivos conectados. Esto es ideal para este tipo de aplicaciones en tiempo real. Las APIs y las bibliotecas de Firebase están bien documentadas y son accesibles, lo que facilita su implementación. En el caso de las tecnólogías que se eligieron anteriormente estas se integran de muy buena manera con Firebase puesto que existen librerías que facilitan la conexión entre Firebase y Arduino IDE y Django. Dentro de todos estos beneficios también se encuentra que Firebase se puede utilizar de manera gratuita para trabajos de investigación como es este caso.

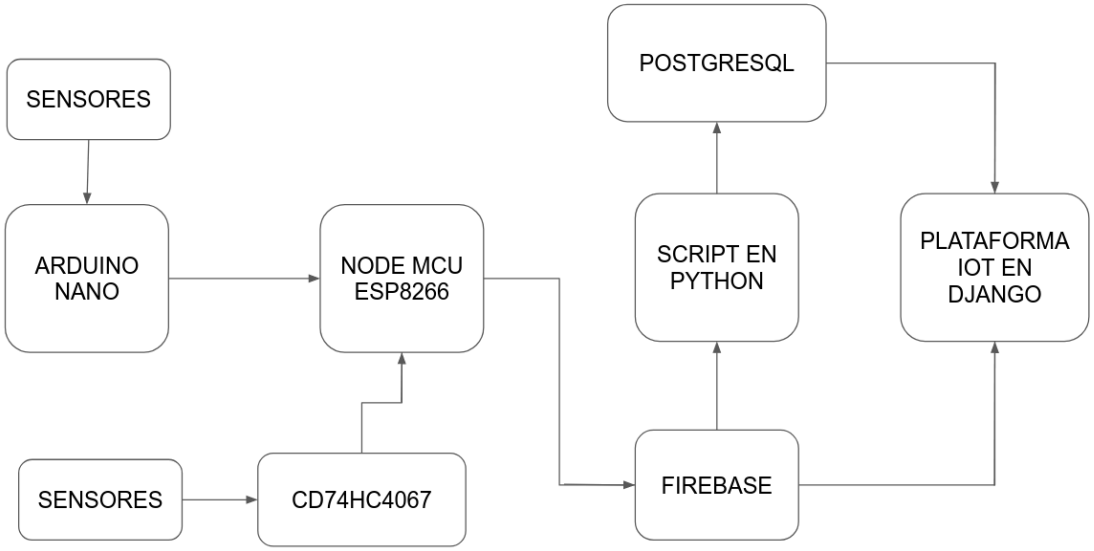
Esto facilita el almacenamiento y la visualización de todos los datos en tiempo real de la estación climática, pero a la hora del almacenamiento de los datos climáticos históricos requiere tener un orden en el almacenamiento, como en tablas, para este caso es más recomendable el uso de una base de datos relacional, entre las más populares en esta opción, existen: MySQL, PostgreSQL, Microsoft SQL Server, entre otros. La opción que se eligió para este caso es PostgreSQL, debido a que es una base de datos que admite el estándar SQL completo, y es una tecnología de código abierto que puede manejar grandes volúmenes de datos y consultas complejas de manera eficiente, también es una de las bases de datos recomendadas para trabajar con Django debido a que estas tecnologías son totalmente compatibles.

Con ello ya empezamos a plantear el flujo de trabajo para la plataforma IoT de la estación climática, el cual viene dado por:

Primero se tomarán los datos leídos por el arduino y pasados mediante comunicación serial al ESP8266, teniendo todas las variables medidas estás serán enviadas desde la estación hacia la base de datos en tiempo real en Firebase, desde aquí se tendrá un script en python el cual permita tomar los datos desde Firebase y guardarlos en PostgreSQL, este script será el encargado de guardar en una tabla de PostgreSQL los datos climáticos por hora, y llegado el final del día, este tendrá que promediar todos los valores de esta primera tabla y guardarlo en otra tabla donde serán guardados los datos históricos por día, después la plataforma IoT será la encargada de tomar los datos en tiempo real y mostrarlos, además también será capaz de tomar los datos almacenados en

PostgreSQL para que puedan ser visualizados. Todo esto se representa en el siguiente diagrama.

Figura 61 : Flujo de la Toma y Muestra de Datos de la Estación Climática



Para habilitar la transmisión de datos desde el ESP8266 a Firebase, es necesario extender el código previamente explicado para establecer conexiones tanto a Internet como a Firebase. Inicialmente, en el programa, se procederá a importar nuevas librerías que serán fundamentales para estas operaciones. Además, se configurarán constantes que actuarán como credenciales necesarias para acceder a Internet; estas credenciales se obtendrán del módem móvil integrado en la estación y de Firebase, y se obtienen al crear el proyecto y la base de datos en tiempo real en la plataforma Firebase. A continuación, completada esta configuración, se instanciará un objeto perteneciente a la librería de Firebase, el cual será utilizado para gestionar el envío y almacenamiento de los datos.

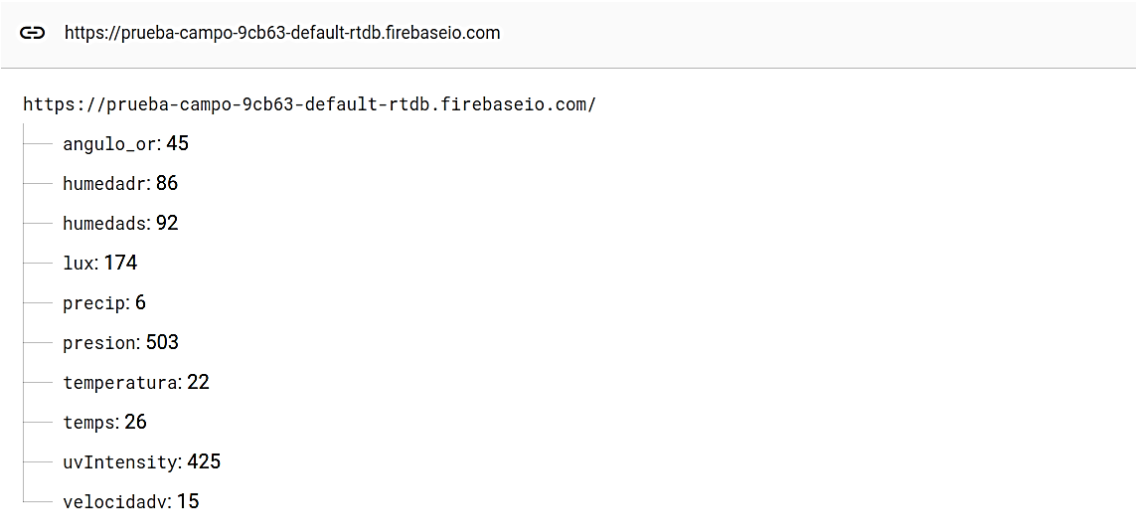
Dentro de la función setup del Arduino, se incorporan las etapas de inicialización tanto de la conexión con la red WiFi como de la verificación de esta conexión, lo que se refleja mediante una confirmación que se imprime en el monitor serial para notificar que la conexión a Internet se ha establecido de manera exitosa. Además, se lleva a cabo la configuración inicial para establecer la comunicación con nuestra base de datos en Firebase. Este proceso incluye la

transmisión de las credenciales necesarias y se ejecuta un procedimiento de reconexión, diseñado para prevenir problemas durante el proceso de inicialización.

Por último, después de recopilar todas las variables pertinentes de la estación climática, incorporamos el código necesario para transferir estos datos a la plataforma Firebase. Este procedimiento incluye una verificación para asegurarse de que exista una conexión operativa con Firebase. En caso de que no se detecte una conexión, se implementa un proceso de reinicio de esta. Luego, se procede a la transmisión de las variables medidas, utilizando el objeto de la librería de Firebase, especificando el nombre de la variable en Firebase donde se almacenarán y, finalmente, proporcionando el valor correspondiente que se almacenará (Véase en el **Anexo 8**).

Con esto ya se logra confirmar la recepción de los datos por parte de Firebase

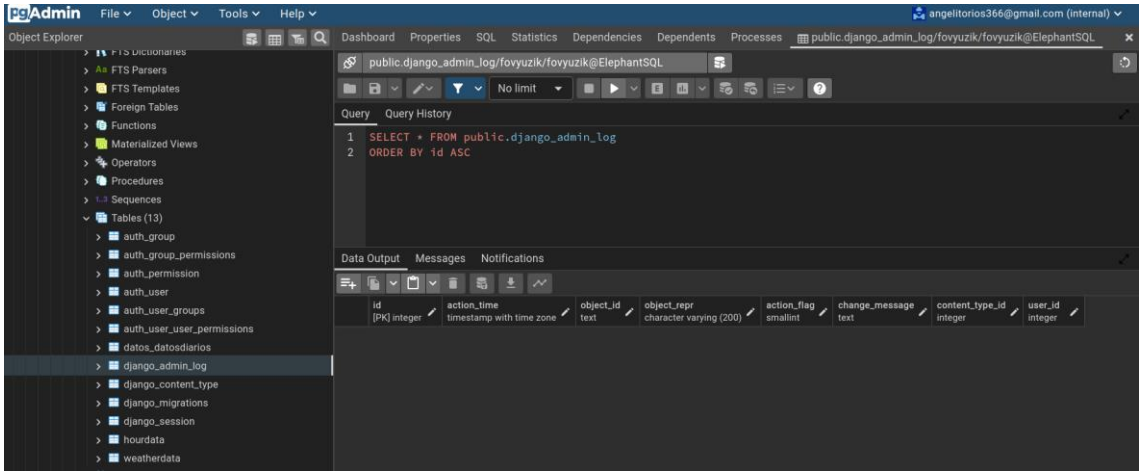
Figura 62 : Recepción de Datos en Firebase



Al cabo de confirmar el funcionamiento de Firebase con la estación climática, se desarrolló el script que será el encargado de tomar estos datos y agregarlos a la base de datos en PostgreSQL, además también será el encargado de promediarlos.

Este script inicia importando las librerías necesarias para conectarnos a Firebase y a PostgreSQL, se realiza un objeto de configuración donde van las credenciales para la conexión a firebase, después de realizada la conexión se guarda los valores en variables representadas con el mismo nombre que en firebase y tomando a lo más 2 decimales en cada variable. Después se tiene la conexión con PostgreSQL, también con nuestras credenciales, y obtenemos la hora actual que nos servirá para saber que fila de la base de datos actualizaremos, después ejecutamos la instrucción SQL para actualizar la base de datos ya con los valores tomados desde Firebase en la hora correspondiente. Finalizamos el script con la carga de los valores promediados de la primera tabla, en otra tabla donde irán los datos históricos y cerramos la conexión de postgreSQL (Véase en el **Anexo 9**).

Figura 63 : Tablas de PostgreSQL en PgAdmin



Luego de completar estas tareas iniciales, se procedió a llevar a cabo la programación de la plataforma IoT utilizando Django. Este proceso comenzó con la creación de un nuevo proyecto en Django, seguido de la creación de un usuario específico para nuestro proyecto. Como se mencionó previamente, una de las ventajas notables de este framework es su panel de administración incorporado, que simplifica significativamente la configuración y gestión del proyecto. En este panel, creamos nuestro primer usuario, denominado "cacao_iot", para simular un usuario real dentro de la plataforma. Es importante destacar que Django es altamente escalable, lo que significa que es posible

agregar más usuarios sin dificultades en el futuro, en caso de que el proyecto evolucione hacia una aplicación comercial.

A continuación que se estableció la base de nuestro proyecto, se procedió a configurar PostgreSQL como la base de datos principal para el proyecto, dado que Django, de manera predeterminada, utiliza SQLite. Para llevar a cabo esta configuración, se introdujeron las credenciales necesarias en el archivo de configuración de Django.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.postgresql_psycopg2',
    'NAME': 'xxxxxxx',
    'USER': 'xxxxxxx',
    'PASSWORD': 'XXXXXXXXXXXXXXXXXXXXXX',
    'HOST': 'postgresql-server',
    'DATABASE_PORT': '5432',
  }
}
```

Luego de tener la base de nuestro proyecto, hay que empezar a modelar la aplicación web antes de iniciar su desarrollo, en este sentido la propuesta es: La aplicación inicia con una ventana de login que luego nos redirige a un dashboard en donde podremos observar las variables climáticas en tiempo real, en este punto se podría poner en esta misma ventana la lista de los registros diarios desde que empezó el experimento, pero para tener un mayor control sobre esto se decidió hacerlo en una vista a parte donde solo se renderice este elemento, además de agregar un botón con el que puedas descargar esta data para su análisis más detallado utilizando otras herramientas.

Primero dentro de nuestro proyecto definimos las urls para las vistas, las cuales serán de:

- / : la ruta raíz de la aplicación es la que será utilizada por el login para poder ingresar a la aplicación.
- /index/ : esta será la ruta donde se podrán observar todas las variables climáticas en tiempo real y será la ruta principal de la aplicación.
- /data/ : esta será ruta donde se mostrarán los datos históricos diarios de las variables climáticas medidas por la estación.

– /data/generar_csv/ : esta ruta será la encargada de darnos la petición para hacer la descarga de los datos en formato csv para su posterior analisis en otros entornos.

Ya teniendo en cuenta las rutas, se pasó al desarrollo de las vistas que serán renderizadas en cada una de estas urls, esto lo haremos en 2 archivos de vistas, uno será utilizado para las 2 primeras rutas, y el segundo archivo view será el utilizado para la data histórica.

Para la programación de las dos primeras vistas, al estar utilizando Python, el código es similar al que se desarrolló para el script que conecta Firebase y postgreSQL, primero se crea el objeto de configuración para firebase y luego se guardan los valores de firebase en variables locales y esto se renderiza pasando como argumento para el archivo html de esta vista.

Después se tiene la vista de login a la que le pasamos únicamente el archivo html donde se renderizará el formulario para ingresar las credenciales, dentro de este archivo Django ya nos proporciona una variable la cual le dirá al formulario si el usuario ingresado es correcto o no, comparándolo con lo configurado en el panel de administración de Django.

```
def index (request):
    angulo_or = database.child('angulo_or').get().val()
    humedadr = database.child('humedadr').get().val()
    humedads = database.child('humedads').get().val()
    lux = database.child('lux').get().val()
    precip = database.child('precip').get().val()
    presion = database.child('presion').get().val()
    temperatura = database.child('temperatura').get().val()
    temps = database.child('temps').get().val()
    uvIntensity = database.child('uvIntensity').get().val()
    velocidadv = database.child('velocidadv').get().val()
    return render(request, 'index.html',{
        "angulo_or":round(angulo_or,2),
        "humedadr":round(humedadr,2),
        "humedads":round(humedads,2),
        "lux":round(lux,2),
        "precip":round(precip,2),
        "presion":round(presion,2),
        "temperatura":round(temperatura,2),
```

```
    "temps":round(temps,2),
    "uvIntensity":round(uvIntensity,2),
    "velocidadv":round(velocidadv,2)
  })


def login (request):
  return render(request, 'registration/login.html')
```

Tal como se ha indicado previamente, una de las fortalezas inherentes a Django radica en su capacidad para proporcionar un ORM (Mapeo Objeto-Relacional) integrado que simplifica significativamente la gestión de datos. De acuerdo con este enfoque, se procedió a la creación de un modelo de objeto destinado a representar las variables de la base de datos para el registro diario de datos. Este modelo fue definido en un archivo denominado "models", y se presenta como se detalla a continuación:

```
class DatosDiarios(models.Model):
  fecha = models.DateField()
  angulo_or = models.FloatField()
  humedadr = models.FloatField()
  humedads = models.FloatField()
  lux = models.FloatField()
  precip = models.FloatField()
  presion = models.FloatField()
  temperatura = models.FloatField()
  temps = models.FloatField()
  uvIntensity = models.FloatField()
  velocidadv = models.FloatField()

class Meta:
  verbose_name = 'datosDiarios'
  verbose_name_plural = 'datosDiarioss'
```

Con todo esto en su lugar, se ha completado la infraestructura para el desarrollo de la última vista. Como se ha mencionado anteriormente, en esta etapa se han definido dos URL distintas. En la primera URL, que será utilizada para mostrar todos los datos climáticos diarios, el proceso comienza con la

obtención de todos los datos almacenados en la base de datos. Esto se logra invocando el modelo previamente creado y almacenando los resultados en un objeto. Posteriormente, este objeto se pasa como parámetro al archivo HTML que se encargará de la representación visual de los datos, lo que permitirá su presentación en forma de tabla.

Luego, se encuentra la URL destinada a proporcionar un archivo CSV que contendrá todos los valores almacenados en la base de datos. Para llevar a cabo esta tarea, se emplea un enfoque similar al de la vista anterior. En primer lugar, se recopilan todos los datos en un objeto. A continuación, se genera una respuesta utilizando el protocolo HTTP. Durante este proceso, se crea el archivo CSV y se procede a escribir fila por fila todos los datos necesarios. Finalmente, se retorna el archivo CSV recién generado como resultado de la operación.

```
def dataview(request):
    datos = DatosDiarios.objects.all()
    return render(request, 'dataview.html', {'datos':datos})


def generar_csv(request):
    datos = DatosDiarios.objects.all()
    response = HttpResponse(content_type = 'text/csv')
    response['Content-Disposition'] = 'attachment; filename="datos.csv"'
    writer = csv.writer(response)

writer.writerow(['fecha','angulo_or','humedadr','humedads','lux','precip','presion','temperatura','temps','uvIntensity','velocidadv'])


    for dato in datos:
        writer.writerow([dato.fecha,dato.angulo_or,dato.humedadr,dato.humedads,dato.lux,dato.precip,dato.presion,dato.temperatura,dato.temps,dato.uvIntensity,dato.velocidadv])
    return response
```

Con todos estos componentes creados y configurados, el último paso es completar los archivos HTML. Estos archivos desempeñan un papel fundamental, debido a que se encargan de representar visualmente todas las vistas de la aplicación y aplicar estilos a través de hojas de estilo CSS.

En la aplicación IoT resultante, el flujo es el siguiente: en primer lugar, se presenta la pantalla de inicio de sesión, donde la aplicación se denomina "Cacao IoT". Una vez autenticados, se accede a la vista principal de la aplicación, donde se presentan las 10 variables obtenidas de la estación climática junto con sus respectivos nombres y unidades. Además, se incluye un menú que permite navegar a la vista donde se presentarán todos los datos históricos.

Dentro de esta vista, se conserva el menú que facilita el regreso a la vista anterior. Aquí, se muestran todos los datos históricos de las 10 variables climáticas, organizados por fecha. Por último, debajo del menú, se encuentra un botón que desencadena la solicitud de descarga del archivo CSV. De esta manera, se concluye la implementación de la plataforma IoT diseñada para su uso en la estación climática automática.

Figura 64 : Login de la plataforma IoT

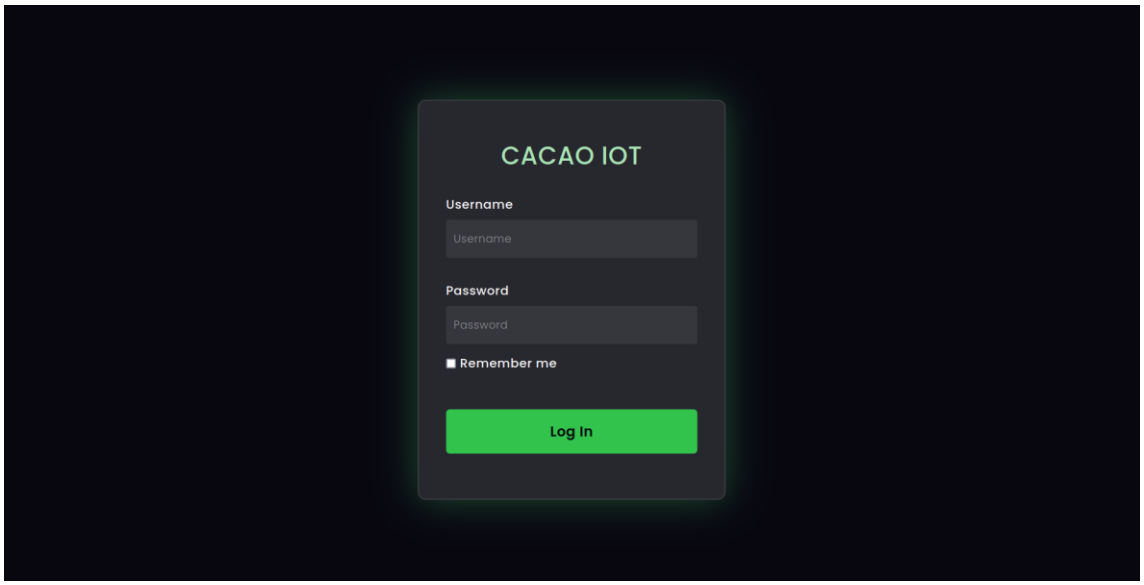


Figura 65 : Visualización de Datos en tiempo real en la plataforma IoT

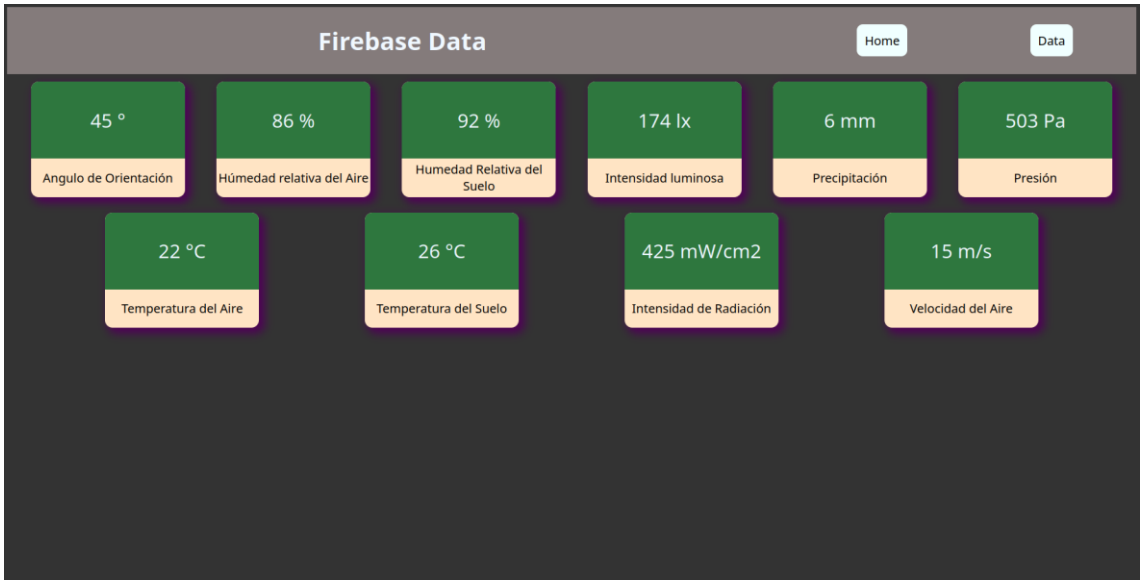


Figura 66 : Visualización de Histórico de Datos Tomados en la Plataforma IoT

PostgreSQL Data

Home | Data

DOWNLOAD CSV

Fecha	Orientación del viento (°)	Humedad Relativa del aire (%)	Humedad Relativa del suelo (%)	Luminosidad (lx)	Precipitación (mm)	Presión (Pa)	Temperatura del aire (°C)	Temperatura del suelo (°C)	Intensidad de radiación UV (mW/cm2)	Velocidad del viento (m/2)
Sept. 13, 2022	35.6	86.47	91.52	352.12	6.25	42659.35	25.3	28.3	134.58	6.35
Sept. 14, 2022	48.7	85.372	87.51	425.18	15.3	50659.35	22.8	26.7	345.178	12.35
Sept. 18, 2022	83.46	86.2	92.59	266.35	22.51	101756.14	26.56	31.25	329.18	2.18
Sept. 19, 2022	144.01	84.08	83.79	343.05	17.21	101880.6	23.75	29.75	413.76	0.7
Sept. 20, 2022	202.32	79.6	87.35	342.57	9.39	101831.71	27.44	25.22	336.51	1.73
Sept. 21, 2022	148.95	87.82	82.87	299.89	13.56	101894.69	23.86	27.0	485.59	2.48
Sept. 22, 2022	73.69	86.82	86.2	270.77	0.06	101813.06	23.79	24.89	412.85	1.14
Sept. 23, 2022	180.69	87.04	84.13	262.08	4.66	101894.6	27.82	30.82	331.15	2.87
Sept. 24, 2022	208.43	81.14	87.89	285.36	11.64	101799.82	26.91	32.61	345.49	1.04
Sept. 25, 2022	195.67	82.55	86.83	318.64	8.65	101718.73	26.03	31.94	433.85	0.88
Sept. 26, 2022	44.3	87.02	85.49	253.45	9.41	101847.67	23.8	31.85	477.8	1.09
Sept. 27, 2022	46.6	85.31	82.15	366.71	23.3	101905.52	25.96	30.29	446.9	0.56
Sept. 28, 2022	139.1	82.46	82.56	369.09	4.09	101725.28	22.1	28.03	334.43	0.76
Sept. 29, 2022	110.61	79.96	88.94	376.12	15.87	101831.47	21.14	32.56	478.64	1.85

3.5.11. Estructura y Montaje de la Estación Climática

Al termino de completar la implementación electrónica y el desarrollo de software de la estación climática, antes de llevar a cabo las pruebas en el campo, es necesario abordar la construcción de la estructura que albergará y sustentará todo el conjunto de la estación climática.

Después de un análisis exhaustivo de las estructuras utilizadas en estaciones climáticas comerciales, se optó por tomar estas estructuras como fuente de inspiración para el diseño. Esta elección se basa en la reconocida estabilidad y eficiencia que ofrecen en la recopilación de datos climáticos.

La estructura base adoptada para la estación climática es una robusta configuración de metal en forma de trípode, diseñada con el propósito de proporcionar una elevada estabilidad. En el extremo inferior de las patas y el comienzo del cuerpo central, se ha incorporado una caja metálica que albergará la placa electrónica, la batería, el módem de conexión a Internet y los cables que conectan con el panel solar. Asimismo, esta caja servirá como punto central desde donde se distribuyen todos los cables que se conectan a los sensores externos, incluyendo aquellos destinados a la medición de humedad y temperatura del suelo. Estos últimos, de acuerdo con las normativas meteorológicas pertinentes y las prácticas específicas para el cultivo de cacao, se encuentran enterrados en el suelo.

Figura 67 : Montaje de la Base de la Estructura de la Estación Climática

Continuando hacia arriba desde la caja metálica, se ubica el panel solar, el cual se asegura al eje de la estructura mediante abrazaderas. Esta disposición facilita el ajuste del panel solar en función de la posición del sol, lo que maximiza la eficiencia de carga de la batería. El eje de la estructura alberga una serie de discos invertidos que sostienen el sensor de humedad y temperatura del aire. Siguiendo las normativas establecidas, este sensor se encuentra a una altura de 1.4 metros del suelo, una medida representativa para el cultivo de cacao.

Figura 68 : Montaje del Panel Solar en la Estructura de la Estación Climática

Figura 69 : Estructura para la toma de Humedad y Temperatura

Figura 70 : Montaje de platos invertidos en la Estructura de la Estación Climática

La parte superior del eje de la estructura alberga el anemómetro y la veleta, dispuestos en los extremos de una barra metálica que forma una “T" con la estructura principal. Estos componentes se encuentran posicionados a una altura de 2.5 metros del suelo, una medida adecuada para la monitorización de las condiciones climáticas relevantes para el cultivo de cacao. En la sección central de esta estructura, se encuentran ubicados los sensores de luminosidad y radiación, protegidos por un material transparente que permite el paso de la luz mientras brinda resguardo contra las inclemencias del tiempo.

Figura 71 : Estructura de la Estación Climática con Sensores

Finalmente, en una estructura independiente al cuerpo principal de la estación climática, se encuentra el pluviómetro, dispuesto a una altura de 1.3 metros según las directrices normativas aplicables. Todos los cables que se extienden hacia los sensores externos han sido recubiertos con mangueras corrugadas con el propósito de minimizar las interferencias eléctricas causadas por las variaciones climáticas.

3.5.12. Pruebas de Campo

Conforme a lo previamente indicado en la propuesta de investigación, se ha elegido una de las plantaciones de cacao perteneciente a la Universidad Nacional Agraria de la Selva como el sitio para llevar a cabo las pruebas de campo. Esta plantación abarca una extensión de casi 1.5 hectáreas y engloba diversas variedades de cacao. La instalación de la totalidad de la estructura se ha efectuado en el centro de esta área de cultivo, asegurándose de que no se vea afectada por la presencia de árboles u otros elementos, en pleno cumplimiento con las directrices estipuladas por las normativas meteorológicas pertinentes.

Figura 72 : Montaje en Campo de la Estación Climática

Figura 73 : Ajuste de Componentes en la Estación Climática

Figura 74 : Verificación del Funcionamiento del Panel Solar

Figura 75 : Montaje de la Placa PCB

Las pruebas se dieron inicio en el mes de agosto de 2022 y se prolongaron hasta enero de 2023. Durante todo este período, se llevó a cabo un monitoreo constante en la plantación con el fin de verificar el funcionamiento de la estación climática y efectuar correcciones ante posibles fallos.

Figura 76 : Inspección de la Estación Climática

Entre los desafíos que se experimentaron durante este estudio, se encontró que, en ciertas ocasiones, la transferencia de datos entre la plataforma IoT y la estación se veía interrumpida debido a la presencia de intermitencias en la conexión a Internet en la ubicación de la plantación. Además, los cambios climáticos constantes, incluyendo lluvias intensas y exposición al sol, contribuyeron al desgaste gradual de ciertos componentes utilizados, como el pluviómetro.

Adicionalmente, se observaron desafíos en relación con la eficiencia del anemómetro y la veleta, puesto que, al ser componentes móviles, requieren una mayor sensibilidad y una velocidad del viento superior a los valores establecidos por las normativas para responder adecuadamente a las variaciones en la dirección y velocidad del viento.

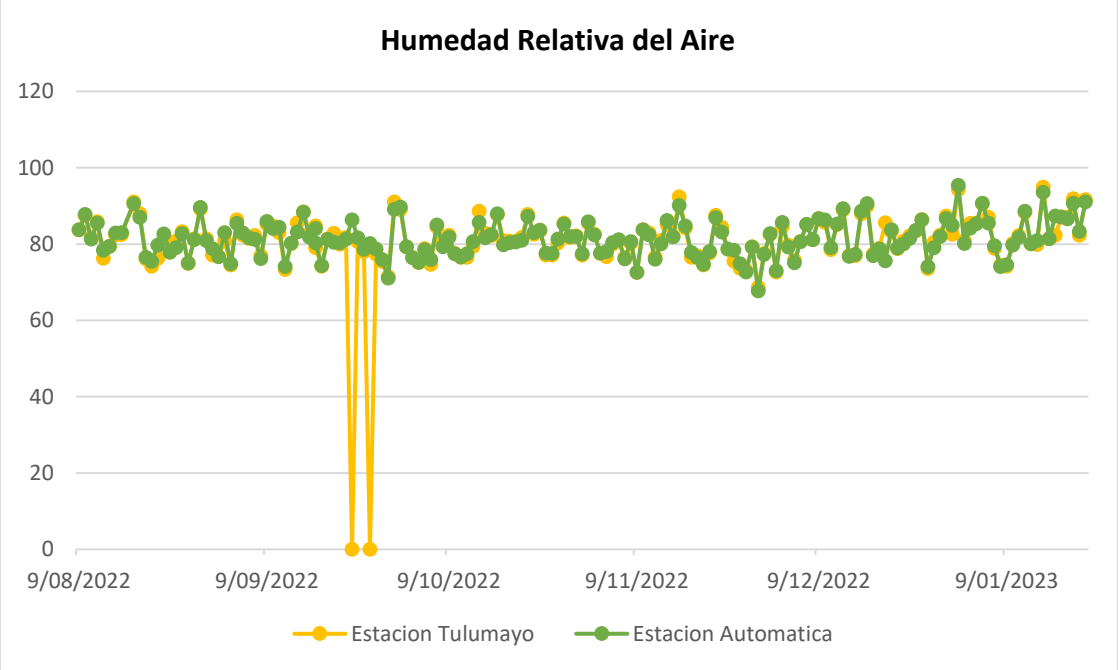
3.5.13. Resultados Obtenidos

Luego del período de prueba, se ha podido verificar que la estación climática proporciona información precisa y fiable para uso del productor. Esta información resulta de gran utilidad al momento de tomar decisiones más informadas en lo que respecta al cultivo de cacao, particularmente en relación con los microclimas que se manifiestan en la región de Tingo María.

A continuación, se muestran los gráficos comparativos de la estación del SENAMHI en la localidad de Tulumayo y de la estación implementada en las variables de humedad relativa del aire, temperatura del aire y precipitación, dado que son las únicas variables que registra el SENAMHI en esta zona. Para las demás variables se muestra el gráfico de los datos recolectados por la estación implementada.

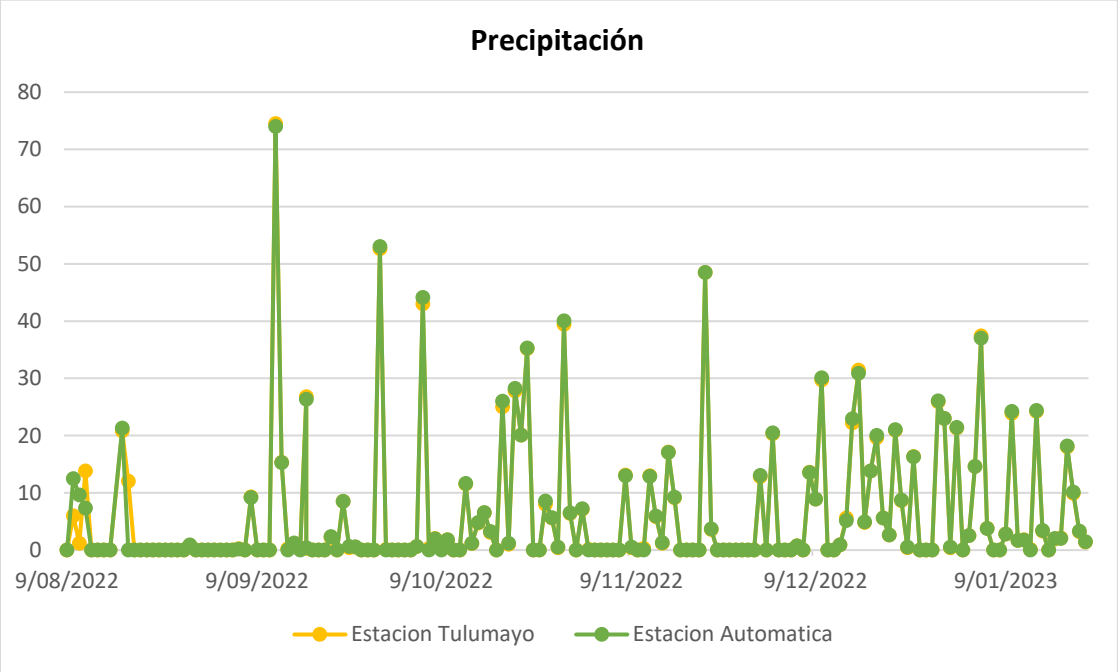
Estas variables también fueron sometidas a la correlación de Pearson con el fin de obtener un valor numerico de la correlación que existe entre los datos de la estación climática implementada y los datos que proporciona el SENAMHI.

Gráfico 1 : Datos de Humedad Relativa (%)



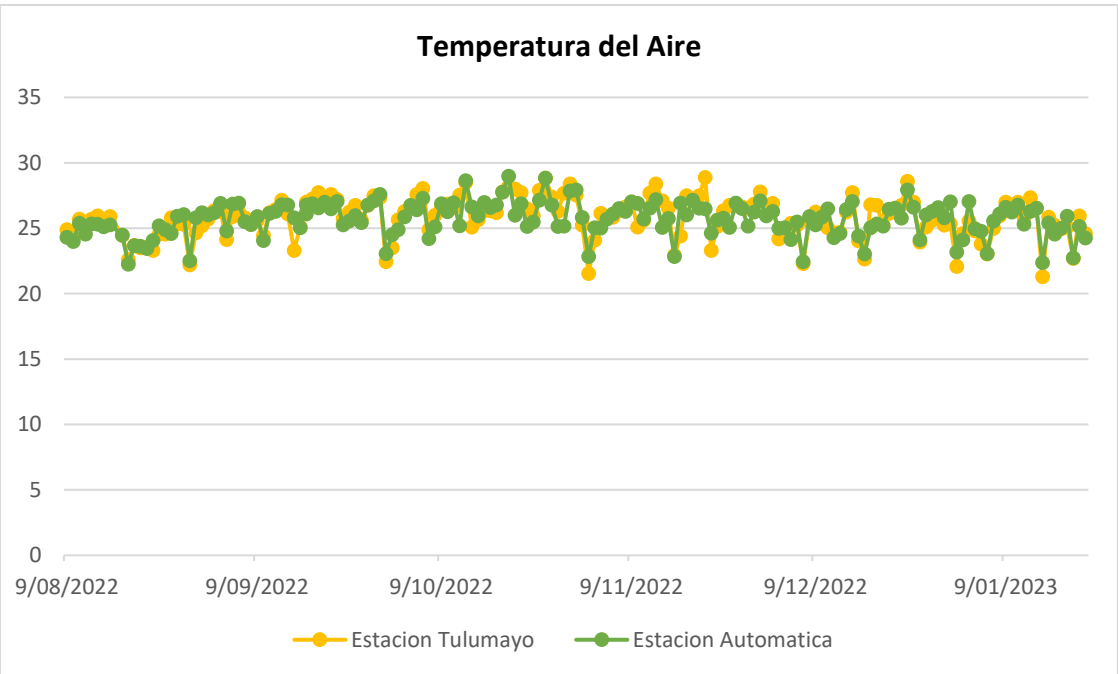
La correlación respecto al **Gráfico 1** es muy fuerte (0.96), lo que sugiere una relación lineal positiva significativa entre las mediciones de humedad entre ambos conjuntos de datos.

Gráfico 2 : Datos de Precipitación (mm)



La correlación respecto al **Gráfico 2** es casi perfecta (0.99), lo que indica una relación lineal muy fuerte y positiva entre las mediciones de precipitación entre ambos conjuntos de datos.

Gráfico 3 : Datos de Temperatura del Aire (°C)



Existe una correlación respecto al **Gráfico 3** positiva fuerte (0.83), lo que indica que hay una relación lineal positiva entre las mediciones de temperatura entre ambos conjuntos de datos.

Gráfico 4 : Datos de Orientación del Viento

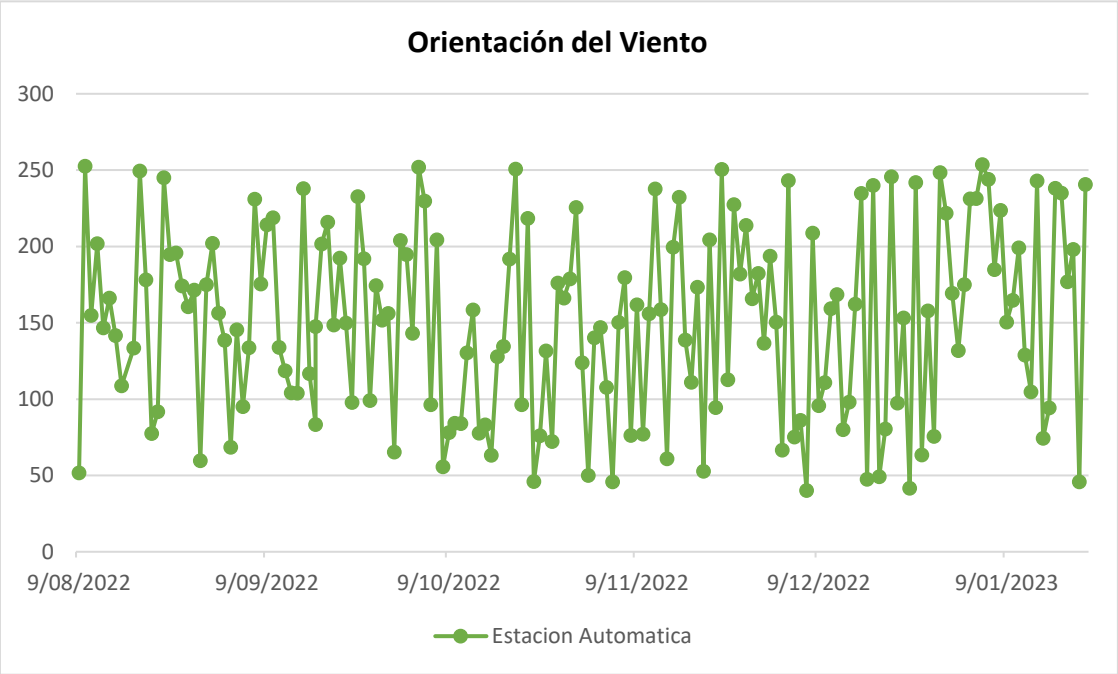


Gráfico 5 : Datos de Humedad Relativa del suelo (%)

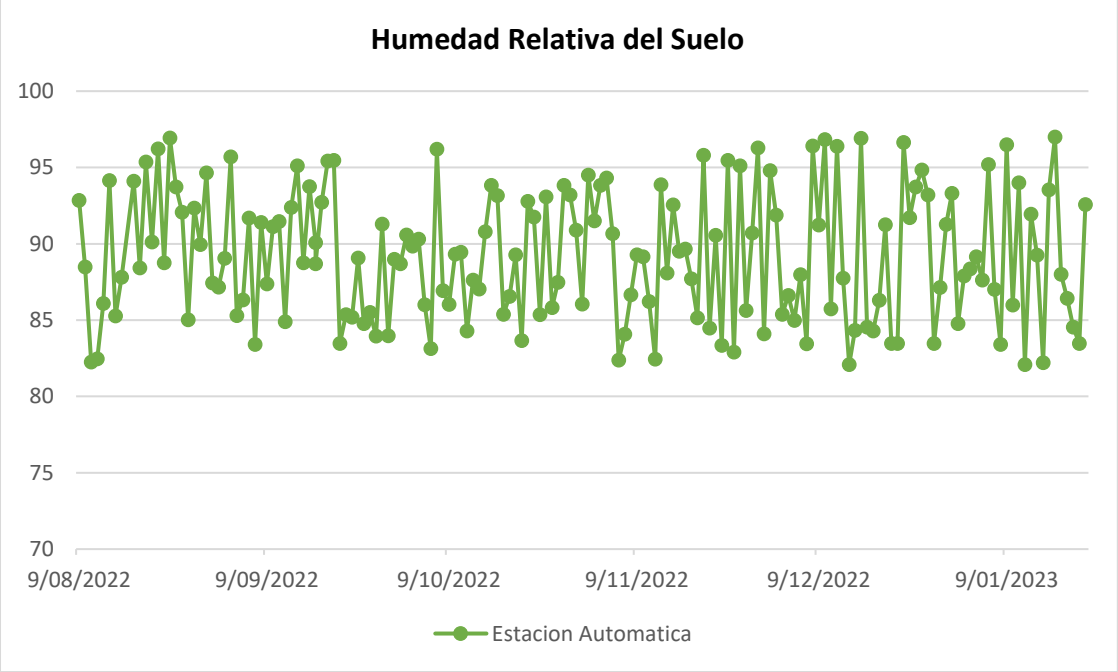


Gráfico 6 : Datos de Luminosidad (lux)

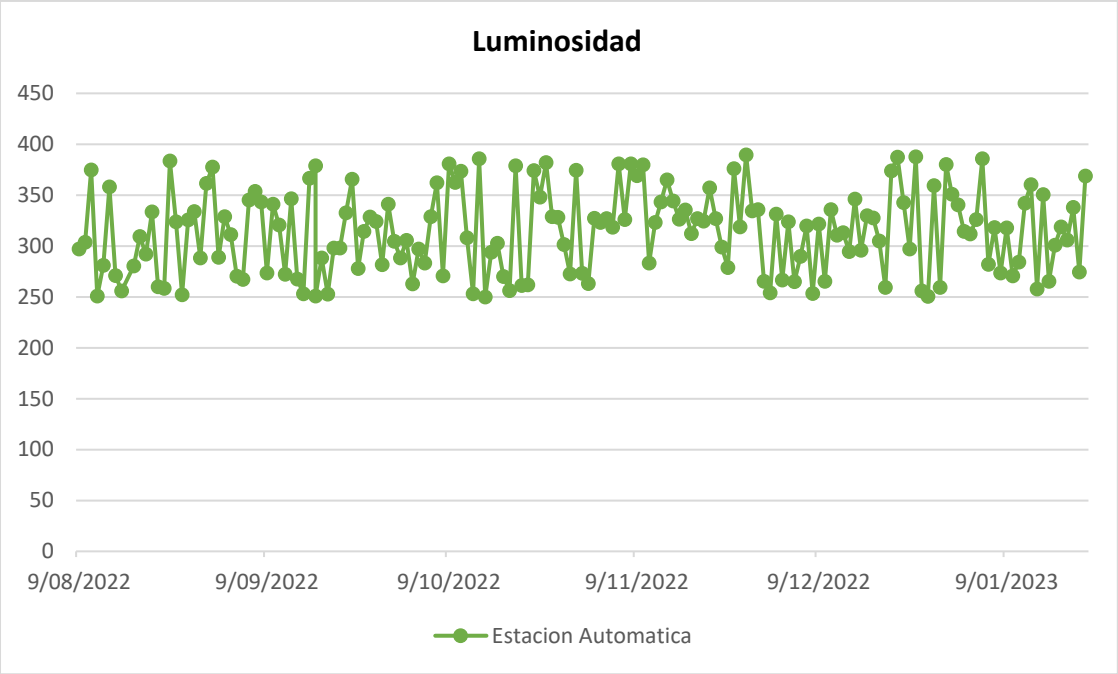


Gráfico 7 : Datos de Presión Atmosférica (Pa)

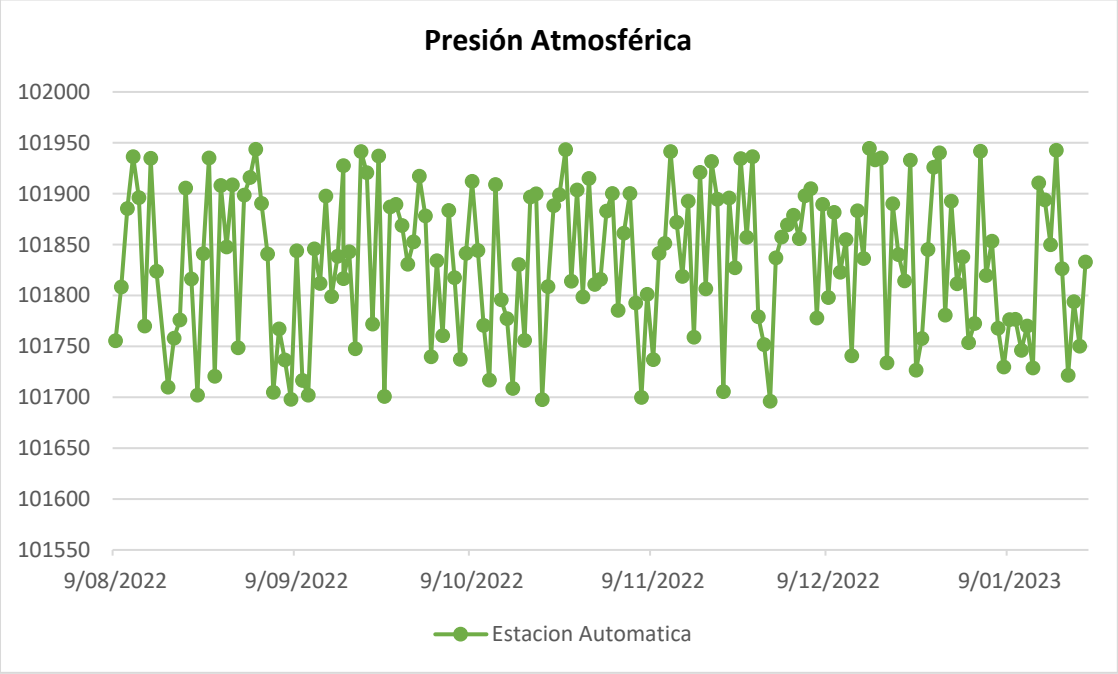


Gráfico 8 : Datos de Temperatura del Suelo (°C)

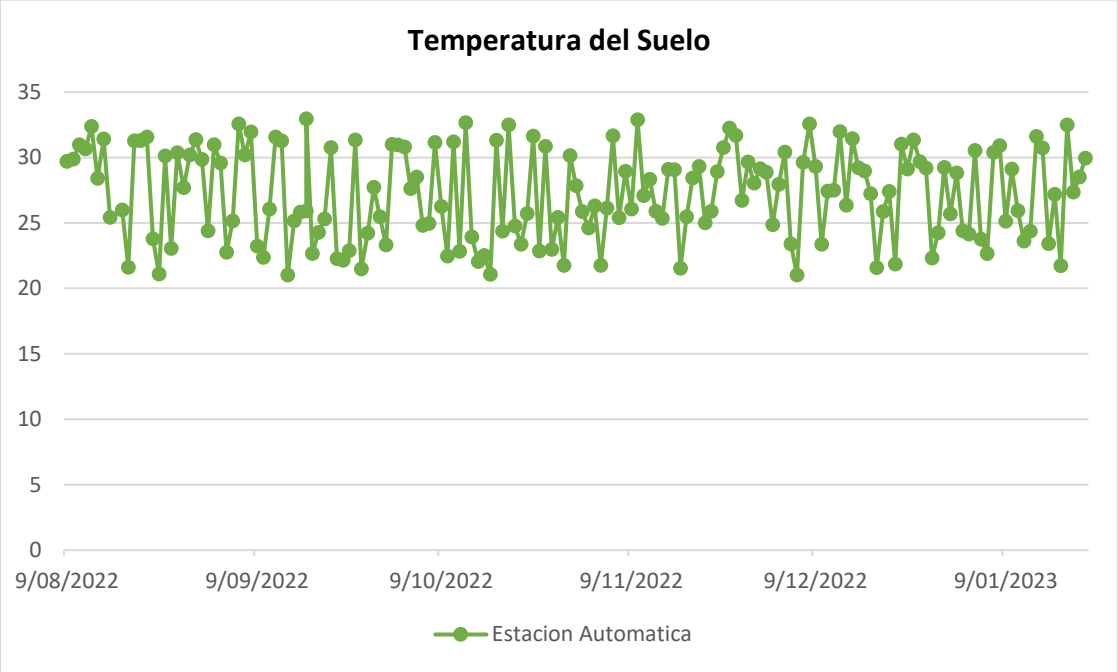
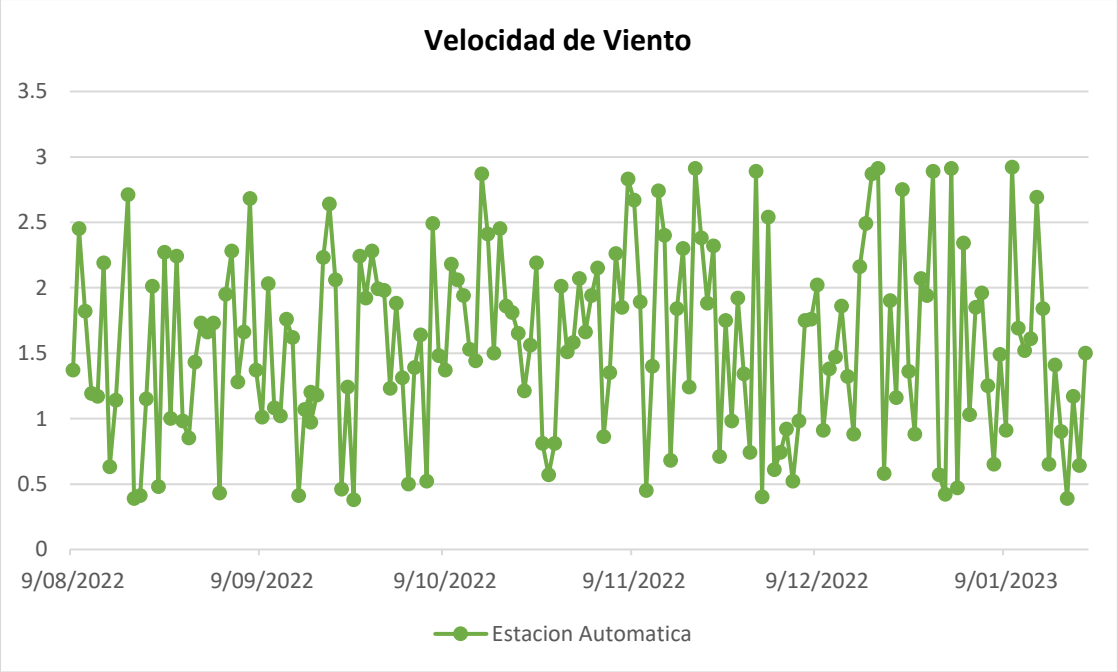


Gráfico 9 : Datos de Velocidad de Viento (m/s)



Es aconsejable emplear el mismo material en todas las partes del pluviómetro, debido a que durante la evaluación se observó que las piezas impresas en 3D se mantuvieron en buen estado, a diferencia de la estructura principal del pluviómetro que mostraba signos de fragilidad. Además, se

recomienda reconsiderar el material utilizado en la fabricación del anemómetro y la veleta, optando por un material menos denso y más resistente. Esto tiene el propósito de prevenir la deformación de estos componentes debido a la exposición al sol y, al mismo tiempo, reducir el peso para cumplir con los parámetros especificados por las normativas correspondientes.

Figura 77 : Pluviómetro luego del tiempo de prueba

Figura 78 : Veleta luego del tiempo de Prueba

Esta estación climática automática también demostró ser una excelente alternativa de bajo costo a diferencia de las soluciones comerciales cuyos costos

se elevan a 4000 dólares el modelo estándar, además que en esta propuesta se incluyen soluciones como las de aplicación IoT además de monitorear 10 variables climáticas que los modelos comerciales no incluyen. Como se muestra en la Tabla 7.18 esta es una propuesta robusta que está al alcance de cualquier persona que desee monitorear las variables climáticas con el fin de mejorar la adaptabilidad climática de su cultivo de cacao.

Tabla 22 : Costos de Componentes de la Estación Climática

Componente	Costo (Soles)
Esp8266	30
Arduino nano	30
Cd74hc4067	10
Dht22	30
Fc-28	10
Sensor de Efecto Hall x2	20
Hmc5883	15
Bh1750	10
Ml8511	35
Ds18b20	10
Bmp180	13
Impresión 3d anemometro	80
Impresión 3d veleta	80
Impresión 3d pluviómetro	60
Impresión placa pcb	200
Cables y tubos de protección	50
Panel solar	130
Batería 7ah	70
Estructura general	600
Otros gastos	120
Total	1603

3.6. Dron de reconocimiento de moniliasis en plantaciones de cacao

3.6.1. Elección de componentes y frame del dron

Para iniciar el proceso de desarrollo del dron, en primer lugar, se tomó la decisión de utilizar un cuadricóptero, basándose en las diversas alternativas disponibles en términos de drones y sus configuraciones de número de hélices. Esta elección se fundamentó en las ventajas inherentes a los cuadricópteros, como su excelente estabilidad y maniobrabilidad, así como su diseño simple y fácil mantenimiento. Además, los cuadricópteros son ampliamente reconocidos en el mercado y ofrecen una amplia gama de opciones en términos de componentes, lo que los convierte en una elección idónea con una relación calidad-precio sobresaliente para su aplicación en una plantación de cacao.

A continuación de determinar la configuración de cuadricóptero, se procedió a investigar las estructuras de drones disponibles en el mercado que se adaptan a esta disposición de hélices. Durante esta búsqueda, se identificaron los siguientes modelos de frames de drones:

Tabla 23 : Frames de Drones Cuadricopteros

Modelo	Tamño del Frame (Mm)	Material del Frame	Pedo del Frame (g)	Carga útil máxima (g)	Uso principal
Dji f450	450	Fibra de vidrio	282	1200	Uso recreativo, fotografía
Dji matrice 100	650	Aluminio	2380	1000	Desarrollo de aplicaciones
Dji s500	500	Fibra de carbono	800	1500	Fotografía, agricultura
Tarot t810	810	Fibra de carbono	1500	5000	Fotografía, cine, agricultura
Dji matrice 300 rtk	900	Aleación de magnesio	1500	2500	Aplicaciones industriales

Dentro de estas opciones se escogió el frame DJI S500, puesto que cuenta con un diseño robusto fabricado en fibra de carbono esto lo hace ideal para soportar condiciones de vuelo diversas, cuenta con facilidad de montaje y ensamblaje, también es muy adaptable lo que lo hace ideal para que pueda acomodarse a las necesidades de esta investigación, es compacto y de fácil transporte además de ser una de las opciones más económicas con todas estas características.

Figura 79 : Frame DJI S500

Dentro de las opciones comerciales para controlador del dron se encuentra DJI Naza, PixHawk 4, Ardupilot, entre otros. Todas estas opciones están diseñadas para controlar la velocidad de los motores en base a la inclinación del dron, pero ninguna de estas opciones está configurada para poder procesar ningún tipo de algoritmo o script externo, por lo que para poder completar la aplicación de reconocimiento de moniliasis en el cacao se requeriría otro dispositivo, por ello se optó por placas de desarrollo potentes pero pequeñas, que puedan ejecutar todas estas tareas, entre las opciones comerciales existe:

Tabla 24 : Microcomputadores

Modelo	Cpu	Gpu	Ram	Conectividad	Gpio
Nvidia jetson nano	Arm cortex a57	128-core maxwell gpu	4gb	Gigabit ethernet, wifi, bluetooth	40
Google coral dev board	Arm cortex a53	Powervr ge8300 gpu	1gb	Wifi, bluetooth	40
Raspberry pi 4	Arm cortex a72	Videocore vi gpu	8gb	Gigabit ethernet, wifi, bluetooth	40
Odroid xu4	Arm cortex a15	Mali-t628 mp6 gpu	2gb	Gigabit ethernet	30

Dentro de estas opciones se eligió la raspberry pi 4 de 8GB de RAM, debido a su alta capacidad de procesamiento para ejecutar modelos de aprendizaje automático también contiene la función de aceleración de hardware por GPU para este tipo de tareas, es compatible con las librerías de inteligencia artificial más usadas como Tensorflow, Keras o Pytorch de manera nativa, es una de las placas más eficientes con respecto al consumo energético, también se presenta como una solución económica frente a las otras placas de desarrollo vistas anteriormente, otra de sus características es que cuenta con módulos de cámara específicos para la raspberry pi lo que hace que la integración de una cámara sea mucho más fácil para la aplicación que estamos buscando en esta investigación, es por ello que dentro de las opciones existentes se eligió la Cámara de Raspberry PI V.2 de 8Mp, la cual es suficiente para el reconocimiento de enfermedades y específicamente de la moniliasis en el cacao.

Figura 80 : Raspberry Pi 4B 8Gb

Sin embargo, con el propósito de transformar la Raspberry Pi en un controlador de dron, es esencial tener la capacidad de regular la velocidad de los motores en función de la inclinación del dron y las instrucciones de manejo. Para cumplir con estos requisitos, se empleó en primer lugar el módulo MPU9250 como una unidad de medición inercial, que está equipado con un magnetómetro, un acelerómetro y un giroscopio. Estos componentes nos permitieron determinar los ángulos de inclinación del dron a lo largo de los tres ejes. Además, se utilizó el módulo PCA9685 para generar señales PWM que controlen la velocidad de los motores de manera precisa.

Figura 81 : Módulo MPU9250

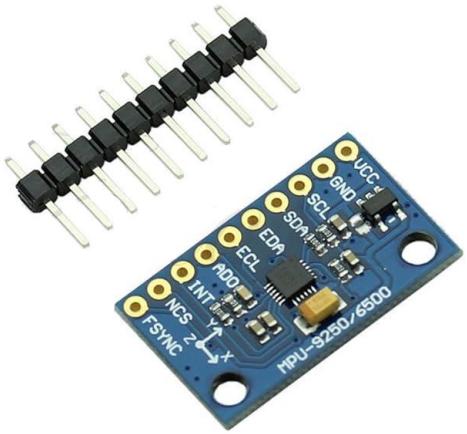


Figura 82 : PCA9685

Es común que los drones se alimenten mediante baterías de Polímero de Litio (LiPo). Dentro de esta categoría de baterías, existe una variedad de capacidades disponibles. Dado que el dron que estamos desarrollando tiene dimensiones y peso considerables, hemos decidido emplear una batería con una

capacidad de 5300mAh. Aunque esta capacidad es ligeramente superior al promedio típicamente utilizada en drones de este tipo, se seleccionó con la intención de proporcionar una mayor autonomía durante los vuelos de la aeronave.

Figura 83 : Batería LiPo 5300mAh

Considerando estos elementos, se llega a la última fase, que abarca los motores y los reguladores de velocidad (ESC) correspondientes. Al analizar las alternativas disponibles en cuanto a motores utilizados en drones, encontramos las siguientes opciones:

Tabla 25 : Motores para Drones

Tipo de motor	Principio de funcionamiento	Ventajas	Desventajas
Motores eléctricos de corriente continua	Convierten la energía eléctrica en energía mecánica girando un rotor dentro de un campo magnético creado por imanes.	Control de velocidad preciso. Fácil de controlar y de usar. Gran eficiencia energética.	Requieren mantenimiento de las escobillas. No son tan eficientes a altas velocidades como los motores sin escobillas.
Motores sin escobillas (brushless)	Utilizan imanes permanentes en el rotor y electroimanes en el estator para generar movimiento.	Mayor eficiencia energética. Menos mantenimiento. Mayor vida útil. Alta velocidad y potencia.	Requieren electrónica de control más avanzada. Pueden ser más costosos.
Motores de inducción	Generan movimiento al inducir corriente en un rotor sin necesidad de conexiones eléctricas directas.	Simple y robusto. Menos costoso. Requiere poco mantenimiento. Utilizado en electrodomésticos y sistemas de climatización.	Menor eficiencia que los motores sin escobillas. Limitado control de velocidad y torque
Motores de chorro (jet engines)	Comprimen aire y lo expulsan a alta velocidad para generar empuje.	Muy alta potencia y velocidad. Utilizado en aviación y propulsión de vehículos aéreos no tripulados.	Consumen grandes cantidades de combustible. Requieren sistemas de enfriamiento y control avanzados.

Dentro de estas opciones se escogió los motores brushless debido a que este tipo de motores son conocidos por su eficiencia y potencia a comparación de los motores con escobillas. Esto significa que proporcionan un mayor empuje con una menor demanda de energía, lo que se traduce en una mayor eficiencia y duración de la batería para el dron. Los motores brushless tienden a ser más duraderos y tener una vida útil más larga en comparación con los motores con escobillas. Los motores brushless permiten un control de velocidad y dirección

más preciso, lo que es esencial para el vuelo estable y la navegación precisa. Teniendo en cuenta los tipos de motores escogidos, habrá que escoger la potencia de estos motores que vienen dado en kv.

Para este tipo de drones, es común encontrar valores de potencia de los motores que oscilan entre 850 kv y 1400 kv. Sin embargo, en esta aplicación en particular, donde se utiliza una Raspberry Pi 4 y otros módulos adicionales junto con una batería ligeramente más grande, se ha tomado la decisión de optar por motores con una velocidad nominal de 920 kv. Para estos motores, se recomienda el uso de reguladores de velocidad (ESC) de 30 amperios. Se eligió esta velocidad nominal dado que se requiere que el motor genere el empuje necesario para poder levantar todo el peso del dron y también que genere las revoluciones necesarias para el manejo del dron.

Figura 84 : Motor Brushless 920kv

Figura 85 : ESC 30A

En cuanto a la selección de las hélices, se han elegido las hélices modelo F450, que son compatibles tanto con el frame como con los motores.

Luego que se han definido todos estos componentes para el dron, se procede a realizar los cálculos necesarios para la elección de la batería y los motores. Esto garantiza que los motores sean capaces de generar el empuje requerido para el vuelo del dron y permite el cálculo del tiempo de vuelo máximo al que el dron puede aspirar.

Primero se presentan los pesos de cada uno de los componentes a utilizar en la construcción del dron:

Tabla 26 : Peso de Componentes del dron

Componente	Peso unitario (g)	Cantidad	Peso total (g)
Frame	405.3	1	405.9
Imu	1.4	1	1.4
Helice	12.5	4	50
Bateria lipo	327.1	1	327.1
Raspberry	46.7	1	46.7
Servomotor	10.6	2	21.2
Esc	30.2	4	120.8
Motor brushless	52.3	4	209.2
Camara	4.5	1	4.5
Soporte de camara	22.5	1	22.5
Pca9685	10.3	1	10.3
Regulador de voltaje	10.7	1	10.7
Protoboard	38.2	2	76.4
Peso total			1306.7

Según los datos proporcionados por el fabricante para los motores escogidos, teóricamente cada motor logra generar un empuje de 1500g, lo que significa que con los 4 motores el dron teóricamente genera 4500g de empuje. Para elevar un dron mediano como el planteado es necesario que el empuje teórico de los motores sea de 2 a 3 veces el peso total del dron, esto proporciona suficiente empuje para superar la gravedad y mantener vuelo estable. En este caso si se cumple esta proporción debido a que el empuje generado por los motores es mayor a 3 veces el peso del dron ($1306.7gf\ x\ 3\ =\ 3920.1gf$).

Otro de los aspectos a calcular es el tiempo de vuelo estimado del dron, para ello se requiere el consumo energético de los componentes del dron, entre los que se considera:

Tabla 27 : Consumo Energético del dron

Componente	Consumo energético (w)
Raspberry pi	5
Sensores	0.5
Motores	112
Total	117.5

Teniendo esto se calcula la potencia que debe entregar la batería seleccionada:

$$Wh = V\ x\ Ah$$

$$Wh = 11.1V\ x\ 5.3Ah = 58.83Wh$$

Finalmente se calcula el tiempo de vuelo estimado del dron:

$$T = 58.86\ {}^{Wh}/_{117.5W} = 0.501h = 30.05min$$

Se calcula que el tiempo de vuelo estimado es de aproximadamente media hora, pero cabe mencionar que los motores no estarán funcionando a máxima potencia a causa de que el dron solo volará a una altura relativamente baja, esto hace que el consumo por parte de los motores sea mucho menor por consecuente el tiempo de vuelo aumenta, por lo que el dato real del tiempo de vuelo para este dron se obtendrá de las pruebas de campo realizadas.

3.6.2. Elección del Método de Manejo del Dron

Dentro de los modelos comerciales comúnmente se utiliza el método de manejo por control remoto el cual utiliza radiofrecuencia para transmitir la señal desde el control remoto hasta el controlador de vuelo, esta opción además de ser más costoso requiere intervención de más componentes y por lo que el dron completo se vuelve un objeto difícil de transportar para condiciones de campo.

Dada la elección de utilizar una Raspberry Pi como la unidad central del dron, se ha decidido emplear un smartphone como dispositivo de control del dron

a través de una aplicación móvil. Esta elección se basa en la amplia disponibilidad de smartphones en la actualidad, lo que los convierte en una opción accesible para la mayoría de las personas.

Al considerar las opciones de conectividad para la comunicación entre el smartphone y la Raspberry Pi, se han evaluado tres alternativas: WiFi, Bluetooth y el protocolo MQTT a través de internet o una red local. Entre estas opciones, se ha optado por utilizar la tecnología WiFi debido a sus ventajas en este contexto específico. El WiFi ofrece una alta velocidad de transmisión de datos y una baja latencia, aspectos cruciales en aplicaciones de control de drones. Además, su alcance de transmisión de datos es adecuado para su uso en plantaciones de cacao. El WiFi proporciona protocolos de seguridad sólidos, como el WPA2, que garantizan la protección de la comunicación, y su configuración en la Raspberry Pi es relativamente sencilla. Esta elección se fundamenta en la eficiencia y la fiabilidad que ofrece el WiFi para la comunicación entre el smartphone y la unidad central del dron en el entorno de una plantación de cacao.

Después de hecho esto se definió el protocolo que se seguirá para la transmisión de datos desde el teléfono móvil hacia la raspberry pi, para esta comunicación se utilizará el protocolo TCP/IP con comunicación Socket. Una conexión socket es un punto de conexión entre dos procesos para permitir la conexión entre ellos haciendo uso de una red, esta conexión se basa en el modelo cliente-servidor. La comunicación a través de sockets TCP es confiable y permite la transmisión de datos en tiempo real con una garantía de entrega. Es adecuada para aplicaciones en las que es importante que los datos se reciban y procesen de manera precisa. En el caso de nuestra aplicación la raspberry será la que funcionará como el servidor de escucha donde se recepcionarán los datos del cliente que será nuestra aplicación móvil.

3.6.3. Elección del entorno de desarrollo para la aplicación móvil de control del Dron.

Dentro de los principales entornos de desarrollo para aplicaciones móviles podemos encontrar los siguientes.

- Desarrollo Nativo Android (Java o Kotlin)
- Desarrollo Narivo IOS (Objective-C o Swift)

– Frameworks Multiplataforma (Flutter, React native, Xamarin)

De las opciones existentes se decidió utilizar el desarrollo nativo android utilizando java y android studio que es el entorno de desarrollo oficial de Android. Esto debido que este entorno te brinda acceso completo a las funcionalidades de android y a la API completa, el tiempo de respuesta de una aplicación nativa es mayor a comparación de los entornos para desarrollo multiplataforma. Además, android studio ofrece herramientas avanzadas de depuración, perfiles, y emulación, lo que facilita el desarrollo de la aplicación. Con esto también se tiene control total de la aplicación, además que cuenta con una amplia variedad de bibliotecas y recursos. Teniendo a java como lenguaje de programación frente a kotlin nos proporciona más recursos y documentación disponible puesto que kotlin es un lenguaje relativamente nuevo. Y el desarrollo nativo para Android, frente al de IOS permite tener una cuota de mercado mayor (alrededor del 72%), y es un entorno más amigable con el desarrollador debido a que no tienes limite sobre que hardware utilizar para el desarrollo.

3.6.4. Configuración de la Raspberry Pi

Como ya se tiene definido los protocolos de comunicación a utilizar, se tiene que configurar la raspberry pi como punto de acceso para que pueda generar una señal de WiFi donde se conectará nuestro teléfono celular para crear la conexión socket.

La raspberry pi 4B elegida está corriendo el sistema operativo oficial para raspberry, Raspbian OS basado en Debian. Si bien es cierto que existen otros sistemas operativos que funcionan en la raspberry pi, se eligió éste debido a que como se mencionó es el oficial y el que recomienda la fundación raspberry, en donde proporciona y amplia documentación sobre el hardware y el software de la raspberry pi.

Siguiendo esta documentación oficial, se procedió con la configuración de la raspberry pi primero instalando las librerías necesarias con los siguientes comandos escritos en la terminal:

```
sudo apt install hostapd
sudo apt install dnsmasq
```

sudo DEBIAN_FRONTEND=noninteractive apt install -y netfilter-persistent iptables-persistent

Se inicia el servicio que permite crear este punto de acceso en la raspberry pi:

sudo systemctl unmask hostapd
sudo systemctl enable hostapd

Después se escribió la configuración de la red WiFi y la ip que se utilizará en la raspberry pi, así también como el rango de ips que tomaran los dispositivos se conecten con las raspberry pi:

interface wlan0
static ip_address=192.168.4.1/24
nohook wpa_supplicant

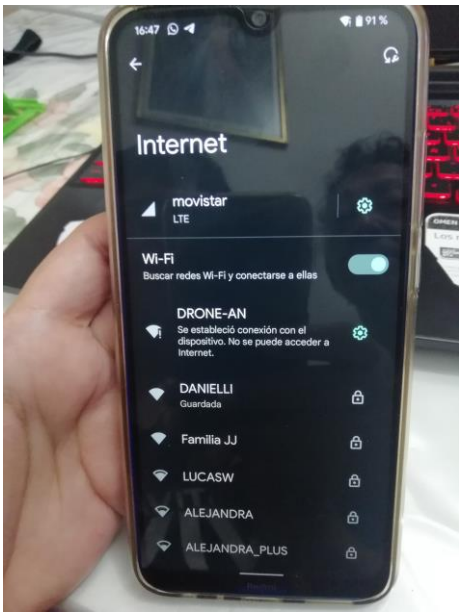
interface=wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
domain=wlan
address=/gw.wlan/192.168.4.1

Posteriormente se configura todos los parámetros relacionados a la red que se generó:

country_code=PE
interface=wlan0
ssid=DRONE-AN
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=drone-angel
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP

Con ello solo queda reiniciar la raspberry pi, y comprobar la conexión con un teléfono celular, y se vio que efectivamente se ha generado la red, y que se logró establecer conexión con la raspberry pi de manera correcta.

Figura 86 : Conexión WiFi entre Android y la Raspberry Pi



3.6.5. Programación de conexión socket

Como ya se vio anteriormente la transmisión de datos entre la raspberry pi y la aplicación Android será mediante sockets, para ello es necesario realizar el código para probar está conexión antes de empezar a implementar el código final tanto de la raspberry pi como de la aplicación móvil.

Dentro de la raspberry pi, primero se importan las librerías que son necesarias para crear la conexión socket, además se configura la dirección y el puerto de nuestro servidor de escucha y se inicia la conexión socket de la siguiente manera:

```
from ast import Try
from operator import truediv
from socket import *
HOST = '192.168.4.1'
PORT = 8080
BUFSIZE = 1024
ADDR = (HOST,PORT)
tcpSerSock = socket(AF_INET,SOCK_STREAM)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(1)
```

Después se programa un bucle que estará escuchando si se envían datos, cuando se recibe algún dato se decodifica esta información para obtener una cadena de texto que pueda ser interpretado de mejor manera en el código, después de esto se imprime la data para verificarla de la siguiente manera:

```
while True:
   print('Waiting for connection')
   tcpCliSock,addr = tcpSerSock.accept()
   try:
      while True:
         data = tcpCliSock.recv(BUFSIZE)
         data1 = data.decode('UTF-8')
         if not data:
            break
         print(data1)
except KeyboardInterrupt:
      print("Cerrando conexión")
      tcpSerSock.close()
```

Ya teniendo el script completo en la raspberry pi se programa el envío de datos desde Android, para esta prueba de conexión se realizó una aplicación sencilla, que únicamente tiene 2 botones para enviar diferentes datos a la raspberry pi. Para ello en un principio se importan todas las librerías que ayudarán a enviar datos mediante socket utilizando java:

```
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.Scoket.
import java.net.UnknownHostException;
```

Luego de completada la importación de librerías, se ha desarrollado una función denominada “Socket_AsyncTask," diseñada para facilitar la transmisión de datos a través de sockets. Como su nombre sugiere, esta función opera de

forma asincrónica en segundo plano, asegurando que no bloquee la interfaz de usuario principal.

Para utilizar la conexión de socket, se inicializa una variable de socket y se sobreescribe el método "doInBackground," que es el núcleo de AsyncTask. A continuación, se establece la conexión de socket especificando la dirección IP y el puerto al que nos conectaremos. Estos detalles están previamente definidos en variables al inicio de la aplicación. Posteriormente, se crea un flujo de salida de datos para transmitir información al socket utilizando la librería DataOutputStream. Los datos que se desean enviar se almacenan en la variable CMD, que actúa como un contenedor para los diversos parámetros que deseamos comunicar a la Raspberry Pi. Luego, se cierra el flujo de transmisión de datos y finalizamos la conexión del socket. También se ha incorporado excepciones para gestionar errores de comunicación que puedan surgir en este proceso.

```
public class Socket_AsyncTask extends AsyncTask<Void,Void,Void> {
    Socket socket;

    @Override
    protected Void doInBackground(Void... params){
      try {
        InetAddress inetAddress =
InetAddress.getByName(MainActivity.wifiModuleIp);
        socket = new java.net.Socket(inetAddress,MainActivity.wifiModulePort);
        DataOutputStream dataOutputStream = new
DataOutputStream(socket.getOutputStream());
        dataOutputStream.writeBytes(CMD);
        dataOutputStream.close();
        socket.close();
      }catch (UnknownHostException e) {
        e.printStackTrace();
      } catch (IOException e) {
        e.printStackTrace();
      }
      return null;
    }
  }
```

En seguida de realizar la función que permite enviar los datos a la raspberry pi se definen las funciones de los dos botones que se configuraron, la

función de ambos es bastante similar, para realizar estas pruebas por un botón se configuró para enviar un solo carácter, y con el segundo botón se configuró para el envío de una palabra, de esta manera poder observar cómo llegan los datos a la raspberry pi.

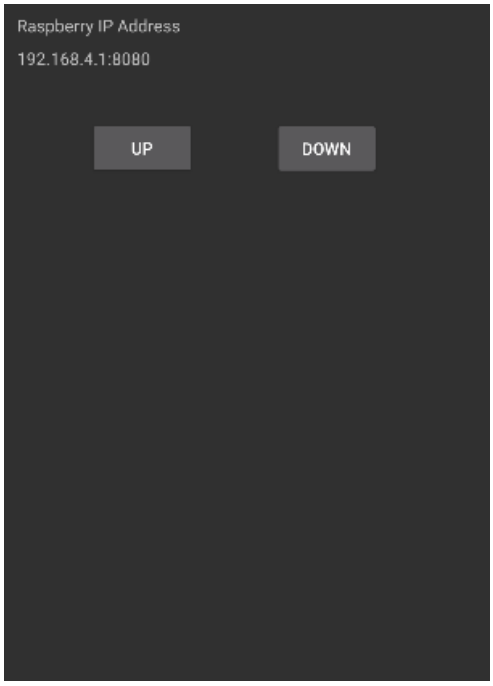
```
btn_1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        CMD = "1";
        Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
        cmd_increase_servo.execute();
    }
});
```

Una vez finalizada la programación de la aplicación, se procedió a realizar pruebas. En este proceso, se estableció la conexión a la red WiFi de la Raspberry Pi y se ejecutó el servidor correspondiente. Sin embargo, al presionar los botones destinados a visualizar los valores recibidos, se identificó un problema relacionado con el búfer de recepción de datos. Se constató que la recepción de un solo carácter se realizaba de manera exitosa, pero al intentar transmitir una cadena de caracteres más larga, esta llegaba fragmentada y dificultaba su lectura en la Raspberry Pi.

En vista de esta situación, se tomó la decisión de limitar la transmisión a un solo carácter. Esta elección se fundamentó en la búsqueda de una mayor eficiencia tanto en el control del dron como en la comunicación de datos entre la Raspberry Pi y la aplicación móvil. Para llevar a cabo esta optimización, se establecieron acciones específicas basadas en el carácter recibido, lo que simplificó el proceso y garantizó una comunicación más efectiva en el proyecto en su conjunto.

Figura 87 : Recepción de datos en la Raspberry Pi para ejecutar acciones

Figura 88 : Aplicación de envío de Datos por conexión Socket

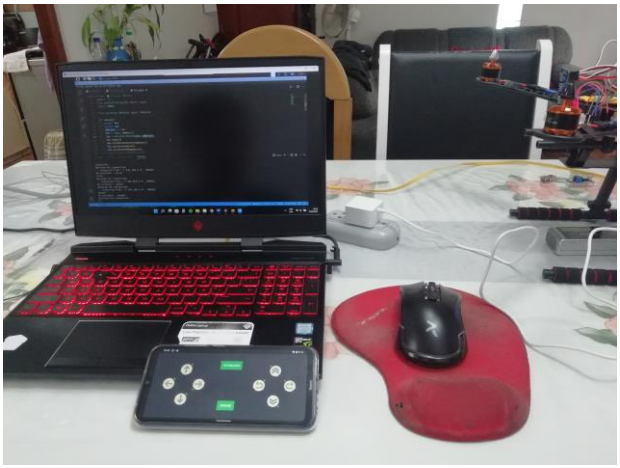


3.6.6. Programación de la aplicación Android

Seguido de haber definido el código de transmisión y recepción de datos tanto por parte de la aplicación móvil como de la raspberry pi., se procedió a realizar el diseño de la aplicación de control de dron como tal, al ser esta la primera versión de la aplicación se planteó realizar el control mediante botones, para ello se planteó 8 botones para los movimientos del dron, y dos botones de seguridad, uno para estabilizar la velocidad de los motores y otro para apagarlo por si acaso. Siguiendo el código mostrado anteriormente, todos los botones se programaron para enviar un carácter, y en base a esto realizar las acciones del movimiento del dron.

Figura 89 : Primera versión de la Aplicación de Control del dron

Figura 90 : Entorno de desarrollo de la primera versión de la Aplicación de Control del Dron



Al realizar las pruebas de esta primera versión de la aplicación se evidenció que el sistema de control por botones no es muy eficiente debido a que dentro de la aplicación es muy difícil detectar todo el tiempo que se mantiene presionado un botón y durante ese tiempo seguir enviando el carácter correspondiente, es por ello por lo que se buscó la opción de poder utilizar un joystick virtual dentro de la aplicación, simulando también los controles comunes para drones de manera física.

Con esta propuesta del manejo mediante un joystick virtual, la primera dificultad al elegir esta nueva interfaz es que Android Studio no cuenta con un objeto nativo que simule un joystick, por lo que para hacer uso de esta característica se empezó a buscar librerías que se puedan utilizar para este caso, se encontró el proyecto Open Source llamado JoystickView cuyo código se encuentra publicado en Github. Esta es una librería que nos permite importar

una vista que simula un joystick virtual, está librería se encontraba un poco desactualizada, por lo que después de integrarla al proyecto y actualizar las dependencias para que pueda funcionar con las últimas actualizaciones de Android, se crean las dos vistas de joystickview que reemplazarán a los botones anteriormente programados, estos servirán para los movimientos de arriba, abajo, girar hacia la derecha, girar hacia la izquierda, adelante, atrás, derecha e izquierda. Dentro de la librería del JoystickView se tiene una función que permanece en la escucha de que el joystick se mueva, dentro de esta función se tienen métodos de la propia librería que nos permiten ver el ángulo al cual se movió el joystcik y también el porcentaje que se movió respecto al centro, estas variables se utilizan también para tener un indicador visual debajo de los joysticks. También tenemos 2 variables, una en cada joystick, que nos permite saber si ambos joystick están o no en su estado inicial, en reposo, estas variables son “first_joy” y “second_joy”, que cada vez que se mueve el joystick le asignamos el valor de false, después entramos en una estructura de selección switch, la que permite saber cuál es la dirección que se le está dando al joystick, y en base a esta se envía un carácter diferente mediante socket a la raspberry pi utilizando la función asíncrona que se creó en la prueba de comunicación socket. Cuando se suelta el joystick y éste vuelve a su estado de reposo la variable de cambio se asigna a “true” y ejecuta la función “sendZero”, la cual envía el carácter designado para que el dron se mantenga en reposo, esta función solo se ejecuta cuando ambas variables de cambio sean true, esto para evitar que se esté ejecutando una de las funciones de algún joystick mientras el otro está en movimiento y evitar falsos datos transmitidos a la raspberry pi.

Después de agregar ambos Joysticks a la aplicación se agregaron 5 botones los cuales formaran parte del control para la cámara y también para el movimiento de la cámara que será anclada a una plataforma en “L” con 2 servomotores. Estos botones definen el movimiento de la cámara para las acciones de “arriba”, “abajo”, “derecha” e “izquierda”. Todos estos botones para la cámara únicamente envían un carácter por la comunicación socket y la raspberry es la encargada de ejecutar todos los movimientos en los servomotores.

Después, para la visualización de la cámara dentro de la aplicación móvil, se decidió utilizar un servidor web en la raspberry pi que sea el encargado de

transmitir la imagen de la cámara en el servidor y poder acceder a este servidor desde la aplicación móvil, mediante una vista web que es un componente nativo dentro del entorno de Android Studio. Para esto después de crear la vista Web en Android Studio se configuraron ciertos parámetros que nos ayudarán a que se pueda observar el servidor web sin ningún problema, además de ajustar esta vista al espacio disponible por la WebView en la aplicación. Como podemos ver se toma la dirección ip configurada para la raspberry y otro puerto, para no interferir con la transmisión de datos del control.

Finalmente se crearon 2 botones a los extremos de la vista web, uno se encarga de enviar un carácter para indicar a la raspberry que apague los motores y el otro botón es el encargado de recargar la vista web en caso de ocurrir algún error en la transmisión (Véase en el **Anexo 10**).

Figura 91 : Diagrama de flujo de la aplicación móvil para control del Dron

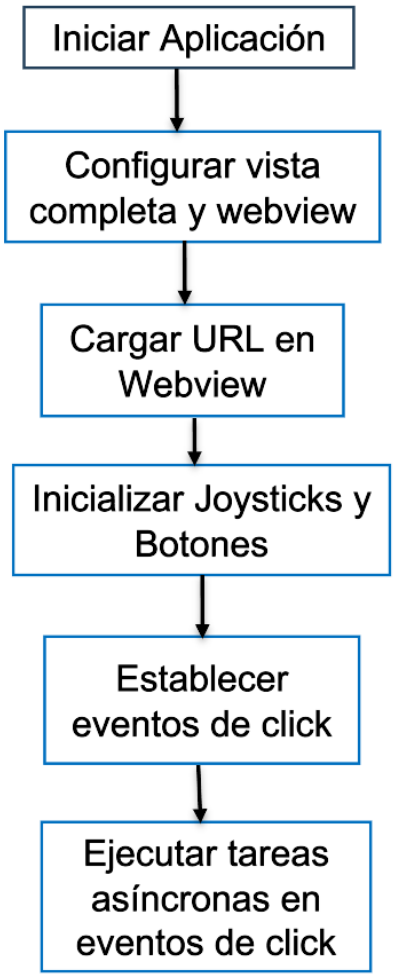


Figura 92 : Aplicación para Control del Dron

Los valores de los caracteres elegidos para cada acción se muestran a continuación:

Tabla 28 : Código de caracteres por acción para movimiento del Dron

Carácter	Acción
0	Adelante
1	Derecha
2	Atrás
3	Izquierda
4	Arriba
5	Giro derecho
6	Abajo
7	Giro izquierdo
8	Cámara
9	Servo izquierdo
A	Servo derecho
B	Servo abajo
C	Servo arriba
D	Reposo
F	Apagar motores

3.6.7. Ensamblaje del Dron

Se procedió al ensamblaje del dron siguiendo el diseño previamente mencionado. El frame seleccionado se caracteriza por su sencillo montaje, utilizando tornillos Allen. Inicialmente, se instaló la base del dron y se ubicaron los motores en sus posiciones correspondientes, prestando especial atención a su orientación y asegurándose de que estuvieran correctamente alineados. Además, se acoplaron los reguladores de velocidad (ESC), que estaban soldados a los motores, con los cables conectados al cuerpo principal del dron.

Figura 93 : Armado del Frame del dron

Figura 94 : Instalación de los motores en el Frame del dron

En el diseño original, se contemplaban dos proboards, uno en la parte delantera y otro en la parte trasera del dron. En la parte delantera se ubican la Raspberry Pi junto con la cámara, mientras que en la parte posterior se albergan los módulos previamente seleccionados. Esta disposición se debió a las

limitaciones en el espacio del cuerpo del dron, puesto que las capas de este estaban demasiado cercanas entre sí. Por lo tanto, la batería se posicionó en una estructura debajo del dron, entre las patas de soporte.

En esta fase inicial de montaje, se emplearon cables para conectar la parte trasera de los módulos con la parte delantera, donde se encontraba la Raspberry Pi. Esta ubicación se justificaba por la necesidad de mantener la Raspberry Pi cerca de la cámara, dado que el cable de conexión disponible no era lo suficientemente largo. Se intentó usar un cable más largo, pero esto generó fallas e interferencias cuando se movía la cámara, lo que llevó a descartar esta opción en favor del cable más corto. Esto permitió mantener una transmisión de imagen constante y estable desde la cámara.

Además, la posición de la Raspberry Pi en la parte frontal del dron se justificó por la necesidad de optimizar la conectividad WiFi, dado que el dispositivo no contaba con una antena externa y se basaba en la señal de la propia Raspberry Pi. Por lo tanto, se requería que no hubiera obstáculos entre la Raspberry Pi y su entorno, garantizando así una conexión estable con la aplicación Android. Esta configuración facilitó las pruebas y la programación de la Raspberry Pi en las etapas posteriores del proyecto.

Figura 95 : Armado del dron con los componentes electrónicos

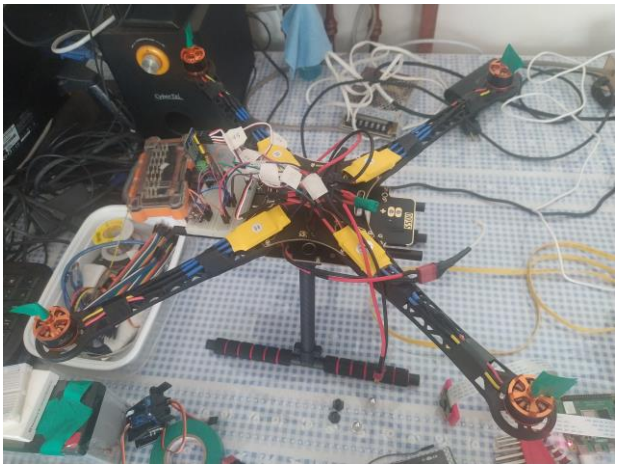
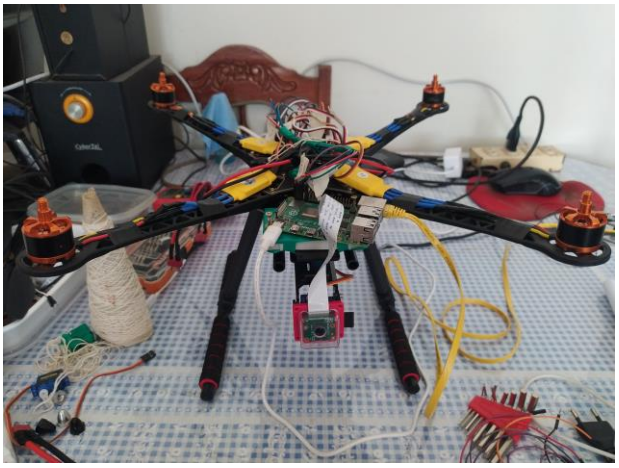


Figura 96 : Primer armado del dron



3.6.8. Prueba y elección de librerías para el manejo de los sensores y actuadores en el dron

En la Raspberry Pi, se hizo uso principalmente de las librerías provistas por los fabricantes, en particular, se optó por las librerías de Adafruit. Esta elección se basó en la búsqueda de una mayor compatibilidad y armonización entre el software y el hardware de la Raspberry Pi. Además, se incorporó una librería especializada diseñada para gestionar las funcionalidades de la cámara específicamente en la Raspberry Pi.

Adicionalmente, se integraron las librerías necesarias para la configuración de la conexión socket y la creación de servidores web en la Raspberry Pi, lo que resultó fundamental para el correcto funcionamiento de la aplicación y la transmisión de datos entre dispositivos.

3.6.9. Programa principal del dron

Primero se creó el script para la transmisión de imagen de la raspberry a un servidor web, cuyo código puede verse en el **Anexo 11**. Para esto se siguió los siguientes pasos:

Importación de bibliotecas y configuración de la página HTML:

- Se importa varias bibliotecas necesarias, como io, picamera, logging, socketserver, Condition y server del módulo http.
- Se define una cadena llamada PAGE que contiene código HTML para crear una página web simple que mostrará el video en tiempo real. Esta

página se mostrará en el navegador cuando se acceda al servidor o en nuestro caso desde la webview de la aplicación android.

Definición de clases:

- StreamingOutput: Esta clase se utiliza para capturar y almacenar el flujo de video de la cámara en un búfer de memoria. También contiene un objeto de condición que se utiliza para notificar a los clientes cuando hay nuevos datos disponibles.
- __init__: Inicializa las variables de la clase.
- write: Método para escribir datos en el búfer. Cuando se detecta el comienzo de un nuevo fotograma (marcado por ciertos bytes), se borra el búfer y se notifica a los clientes que el fotograma está disponible.
- StreamingHandler: Esta clase maneja las solicitudes HTTP que llegan al servidor.
- do_GET: Procesa las solicitudes GET y responde según la ruta de la solicitud.
- Si la ruta es /, redirige al cliente a /index.html.
- Si la ruta es /index.html, envía la página HTML definida en PAGE.
- Si la ruta es /stream.mjpg, comienza la transmisión de video en tiempo real.
- De lo contrario, responde con un error 404 si la ruta no se encuentra.

Creación de una instancia de cámara:

- Se crea una instancia de la cámara Raspberry Pi con una resolución de 640x480 píxeles y una velocidad de fotogramas de 24 fotogramas por segundo.
- Se crea una instancia de la clase StreamingOutput para capturar el video de la cámara y almacenarlo en un búfer.
- La cámara comienza a grabar video en formato Motion JPEG (mjpeg) y el resultado se envía al objeto output, que almacena el video en el búfer.
- Se define una dirección IP ('192.168.4.1') y un puerto (8081) en los que el servidor escuchará las solicitudes entrantes.

- Se crea una instancia de la clase StreamingServer, que es una clase que hereda de socketserver.ThreadingMixIn y server.HTTPServer. Esta instancia se encargará de administrar las conexiones entrantes y las solicitudes HTTP.
- El servidor se inicia llamando a server.serve_forever(). Esto hace que el servidor esté listo para recibir solicitudes HTTP y transmitir video en tiempo real.

Finalmente, cuando el servidor se detiene (por ejemplo, cuando se cierra el programa), se detiene la grabación de video llamando a camera.stop_recording().

Después se desarrolló un script que será el encargado de controlar el módulo para enviar la señal pwm tanto a los motores como a los servomotores, el código de este script puede verse en el **Anexo 13**.

Se declaran las variables globales a utilizarse

- init: Lista de valores de inicio para los motores, valores menores no se reconocen en el motor y se toman como 0.
- first: Lista de valores iniciales para los motores con los que empiezan a rotar.
- vel: Lista de velocidad reales de los motores.
- throt: Lista de valores de velocidad teóricos de los motores.
- serv: Lista de posiciones de servos.

Al último motor se le dió un poco más de duty cycle, debido a que éste es menos eficiente que los demás. Estos valores fueron probados individualmente para cada motor, y se encontró que estos motores funcionan con un valor de duty_cycle mayor o igual a 2400 y menor o igual 7400. Para el caso de los servomotores también se realizaron las pruebas individuales y se encontró que uno de los servos funciona en el rango 1700 a 5800 y el otro servo de 4500 a 8000, el valor de inició que se le dió a los servomotores es cuando la estructura en “L” en la que estarán se encuentra centrada al medio en ambas direcciones.

Después se realiza la inicialización del hardware donde se importan y configuran las librerías correspondientes.

- Se importan las bibliotecas necesarias, como time, adafruit_servokit, board, busio, y adafruit_pca9685.
- Se configura una frecuencia de 50 Hz para la placa PCA9685.

Después se crea la función setup donde se definen dos variables más, que son arreglos con las direcciones tanto de los motores como de los servomotores en la pca9685.

- Los servos se inicializan a las posiciones iniciales especificadas en serv.
- Se inicializan y calibran los motores, haciendo que los valores de los motores cambien de 0 a 2500 y luego vuelvan a 0, esto para garantizar que todos los motores se hayan inicializado correctamente y estén preparados para las señales de aumentar su velocidad para que el dron tome vuelo.

Después se pasó a definir las funciones del control de movimiento del dron:

- arriba(): Aumenta la velocidad de los motores para elevar el dron.
- abajo(): Disminuye la velocidad de los motores para descender el dron.
- adelante(): Controla el movimiento hacia adelante del dron.
- atras(): Controla el movimiento hacia atrás del dron.
- derecha(): Controla el movimiento hacia la derecha del dron.
- izquierda(): Controla el movimiento hacia la izquierda del dron.
- giro_der(): Controla el giro hacia la derecha del dron.
- giro_izq(): Controla el giro hacia la izquierda del dron.
- setvel(ml): Establece la velocidad real de los motores.
- setthrot(ml): Establece la velocidad patrón de los motores.
- cam_der(), cam_izq(), cam_arr(), y cam_aba(): Controlan la posición de la cámara.
- apagar(): Apaga los motores y restablece las velocidades iniciales.

Finalmente se definen las funciones para el control de los motores brushless que son utilizadas por las funciones anteriormente definidas y que éstas también puedan ser llamadas por otros scripts, como por ejemplo para imprimir el valor de las velocidades de cada motor para hacer su monitoreo.

- Las funciones elevar(index) y disminuir(index) aumentan y disminuyen respectivamente la velocidad de un motor individual en un step de 10 para evitar cambios tan lentos o bruscos de velocidad
- setmot(): Actualiza la velocidad de los motores.

Después se pasó con el desarrollo del script que hará uso de la MPU9250 para que nos de los valores de los ángulos de inclinación del dron, cuyo código puede verse en el **Anexo 14**.

Primero se importan las librerías necesarias para este script, una librería que nos ayudará con la comunicación I2C que es el protocolo que utiliza la MPU9250, y la librería que nos ayudará a trabajar con la MPU9250, de la siguiente manera:

Al cabo de haber hecho esto se definen varias funciones que podrán ser llamadas en otros scripts, tales como:

Función setup():

- Configura las conexiones con el sensor MPU9250 a través del bus I2C (SMBus).
- Inicializa el objeto imu de la clase MPU9250 para interactuar con el sensor.
- Inicializa el objeto sensorfusion de la clase kalman.Kalman() para fusionar datos de sensores y obtener una estimación más precisa de la orientación.
- Lee los datos iniciales del sensor y calcula la orientación inicial (startroll, startpitch, startyaw).
-

Después se tiene la definición de 3 funciones para leer los valores del acelerómetro, giroscopio y magnetómetro en los 3 ejes.

Función get_orien():

- Lee los datos del sensor, calcula la orientación actual (r, p, y) y la compara con la orientación inicial (startroll, startpitch, startyaw).

- Ajusta las diferencias para que estén dentro del rango de -180 a 180 grados.
- Devuelve la diferencia en la orientación en grados en los tres ejes.

Función get_orien_kal():

- Realiza una fusión de datos de sensores utilizando un filtro de Kalman para obtener una orientación más suave y precisa.
- Calcula la orientación actual (r, p, y) y la compara con la orientación inicial (startroll, startpitch, startyaw).
- Ajusta las diferencias para que estén dentro del rango de -180 a 180 grados.
- Devuelve la diferencia en la orientación en grados en los tres ejes.

Después se programó un script sencillo para tomar la fotografía de la cámara y almacenar esta imagen de manera local.

El script comienza importando la librería PiCamera desde picamera. Esta librería proporciona una interfaz para interactuar con la cámara de la Raspberry Pi.

Se crea una instancia de la cámara utilizando PiCamera(). Esto inicializa la cámara y la prepara para capturar imágenes, se configura la resolución de la imagen que se capturará. En este caso, la resolución se establece en 640x640 píxeles, después se ejecuta un comando para voltear verticalmente la imagen capturada con el fin de ajustar la orientación de la imagen. Finalmente se hace la captura de la imagen y se guarda en la ubicación especificada.

```
from picamera import PiCamera
camera = PiCamera()
camera.resolution = (640,640)
camera.vflip = True
camera.capture('/home/greenskull/Desktop/raspberry/test_images/image.jpg')
```

Después se creó un script para el control de estabilidad en el dron mediante un PID, se programó este script de la siguiente manera:

Definiendo las variables globales preverror, sumerror, kp, ki, kd, y dt que se utilizan para almacenar valores previos, sumar errores, coeficientes PID y el intervalo de tiempo entre las iteraciones. Estas variables posteriormente se ajustarán sus valores con las pruebas de todo el dron.

Función pidfunc(tar, cur, i): Esta es la función principal que calcula la salida del controlador PID para un eje específico (0, 1 o 2, que corresponden a x, y, z). Aquí está cómo funciona:

- tar: El valor deseado (setpoint) para el eje actual.
- cur: El valor actual (lectura del sensor) para el eje actual.
- i: Índice que indica el eje al que se aplica el control (0 para x, 1 para y, 2 para z).

  La función calcula tres componentes del control PID:

- P (Proporcional): Es proporcional al error actual (error). Se multiplica por el coeficiente proporcional kp[i].
- D (Derivativo): Es proporcional a la tasa de cambio del error (error - preverror[i]) dividida por el tiempo (dt). Se multiplica por el coeficiente derivativo kd[i].
- I (Integral): Es proporcional a la suma acumulada de errores (sumerror[i]). Se multiplica por el coeficiente integral ki[i].

Luego, se suma P, D e I para obtener la salida (out). Se aplican restricciones para asegurarse de que out esté dentro de un rango específico (-600 a 600 en este caso). También se actualizan las variables preverror[i] y sumerror[i] para su uso en la próxima iteración (Vease en el **Anexo 15**).

Después de realizar esto se programó el script principal el cual será el encargado de llamar y utilizar las funciones de todos los scripts anteriores que hemos venido programando, en este script es donde se encuentra el servidor de escucha para recepcionar los datos desde la aplicación móvil.

Primero se empezó con la importación de librerías, estas se utilizan para la realización de operaciones matemáticas, el módulo que nos sirve para la comunicación por sockets, también importamos las clases Popen y PIPE que se

utilizan para ejecutar procesos externos y redirigir la entrada/salida de estos procesos. Después se importan todos los módulos que se programó anteriormente. Y también importamos el módulo para trabajar con hilos en Python y las librerías para trabajar con el tiempo y en la comunicación con dispositivos I2C.

Luego, se establecen algunas variables globales que desempeñan un papel integral en todo el script. Entre estas variables se incluyen 'change', 'stable', 'pid_vel' y 'tar_get'. Estas variables cumplen diversas funciones en relación con la estabilidad del dron. Por ejemplo, 'change' registra cuándo se produce un cambio y cuál era la velocidad previa antes del cambio. 'pid_vel' almacena la velocidad de salida generada por la función de control PID para realizar correcciones necesarias. Asimismo, 'tar_get' guarda los ángulos deseados en cada uno de los ejes para cada instrucción enviada al dron. Es importante destacar que estas variables globales pueden ser accedidas y modificadas desde cualquier parte del script según sea necesario.

Después se llama a las funciones para inicializar los motores y servomotores, así también como la unidad de medida inercial (imu) que se está utilizando.

Se inicia un proceso externo para ejecutar el script 'cam_server' que es el que actúa como un servidor de transmisión de video para visualizar la imagen de la cámara del dron en tiempo real, tal como se explicó la programación de este script anteriormente.

```
cam_process = Popen(['python3',
'/home/greenskull/Desktop/raspberry/prueba_completa/cam_server.py'])
```

Se definen 2 clases de hilos que se utilizarán para ejecutar funciones en paralelo a intervalos regulares. Cada hilo espera durante un intervalo de tiempo antes de ejecutar la función de nuevo. Un hilo ('k') está destinado a la función de control PID ('pidfunc') y el otro ('k2') a la función de impresión de datos ('onlyprint'). Esto permite que estas funciones se ejecuten en segundo plano mientras que el script principal continúa su ejecución.

La función 'pidfunc' implementa el control PID para ajustar las velocidades de los motores de dron con el fin de mantenerlo estable. Calcula las correcciones

necesarias en función de la diferencia entre la inclinación deseada (‘tar_ang’) y la inclinación actual (‘rpy’). Luego, actualiza las velocidades de los motores (‘pid_vel’) para corregir cualquier desviación de la inclinación deseada en base a la disposición de los motores en la estructura del dron.

La función ‘onlyprint’ se encarga de imprimir datos relevantes en la consola, como el estado de los motores, las velocidades, las correcciones PID y la inclinación. Esto permite supervisar el estado del dron durante la ejecución.

Después se realiza la inicialización del socket, se configura la comunicación por socket para permitir que el dron reciba comandos desde la aplicación android. Esto se realiza configurando el host (‘HOST’) y el puerto (‘PORT’) en el que el dron escuchará los comandos enviados por socket. Y como podemos observar tenemos la variable de comandos en base a la tabla explicada anteriormente para cada acción de movimiento tanto del dron como de la cámara.

Por último, se ha implementado el bucle principal que permanece en constante escucha de los comandos que llegan a través del socket. Este bucle se asemeja al que se programó anteriormente, pero con una diferencia crucial: para cada carácter recibido, se invoca una función de movimiento específica para controlar los motores del dron. Se introduce el uso de tres variables adicionales para mejorar la precisión y eficacia del movimiento. En primer lugar, la variable 'stable' se encarga de almacenar el estado de estabilidad del dron antes de realizar cualquier movimiento. La variable 'change' actúa como una bandera que indica si se está ejecutando una función de movimiento en el dron o si este se encuentra en estado de reposo. Por último, la variable ‘tar_angle' determina los ángulos deseados en el dron con respecto al tipo de movimiento que se pretende llevar a cabo. Estos ángulos son esenciales para que la función de control PID pueda ajustar de manera precisa las velocidades de los motores y, de esta forma, ejecutar los movimientos de manera efectiva y precisa.

Una vez que se ha completado cualquier movimiento en el dron, la aplicación envía el carácter 'd' como indicativo de que el dron debe regresar a su estado de reposo. Para lograr esto, se cambia el valor de la variable 'change' a Verdadero (True), se restablecen los ángulos a 0 y se recuperan los valores de velocidad almacenados previamente en la variable de estabilidad.

Además, se ha implementado un segmento de código encargado de capturar imágenes utilizando la cámara. Dado que no es posible acceder simultáneamente a la cámara desde dos procesos distintos, se requiere detener el servidor de transmisión de video en tiempo real. Luego, se ejecuta el script de captura de imágenes, las cuales posteriormente serán sometidas a un algoritmo de inteligencia artificial para determinar si los frutos de cacao están infectados con moniliasis o se encuentran en buen estado (Vease en el **Anexo 17**).

3.6.10. Programación del algoritmo de reconocimiento y clasificación de imágenes, elección de versiones y librerías de inteligencia artificial

Para el inicio en el desarrollo del algoritmo de inteligencia artificial, fue necesario definir las versiones de las librerías y lenguajes de programación que se utilizó. Se escogió el lenguaje de programación python debido a que es compatible con Raspberry Pi, y este lenguaje está optimizado para trabajar junto a la raspberry, esto permite ejecutar modelos de inteligencia artificial de manera eficiente en hardware limitado, además de ser un lenguaje conocido por su facilidad de uso y legibilidad.

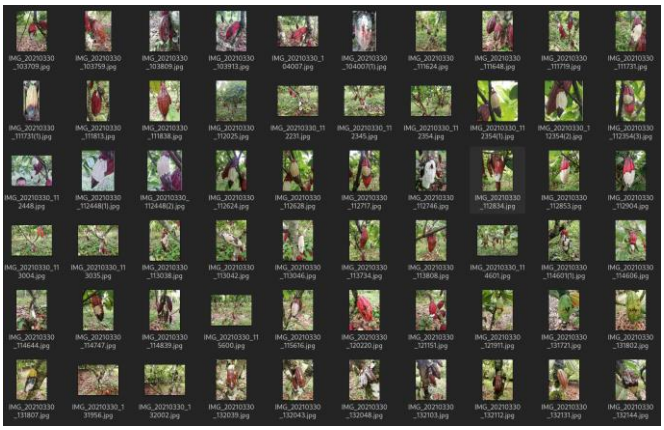
La librería que se usó para desarrollar este algoritmo es Keras, esta es una API de alto nivel para redes neuronales que se ejecuta sobre el backend de TensorFlow. Su diseño centrado en el usuario facilita la creación y experimentación con modelos de inteligencia artificial, lo que ahorra tiempo en el desarrollo y la depuración. Keras además permite la creación de modelos livianos y eficientes, lo que es crucial para ejecutar algoritmos de inteligencia artifial en un dispositivo como la raspberry pi, que no tiene la potencia de cálculo de una computadora de alto rendimiento.

Considerando esta situación, se ha observado que la Raspberry Pi viene preinstalada con Python 3.7. Para habilitar la funcionalidad necesaria para la inteligencia artificial, específicamente TensorFlow y Keras, es crucial instalar versiones específicas. En este contexto, la versión más estable de TensorFlow es la 2.1. Mientras es factible instalar versiones posteriores de Keras en la Raspberry Pi, la compatibilidad con la versión de TensorFlow seleccionada es una preocupación. En consecuencia, se ha optado por utilizar Keras 2.2.4, una versión que es plenamente compatible con la elección de TensorFlow.

Se dispone de un extenso conjunto de datos que consta de alrededor de 4000 imágenes que abarcan una variada colección de frutos de cacao. Estas imágenes se han recolectado a lo largo de un periodo de 4 meses mediante visitas a tres plantaciones de cacao distintas. La diversidad de variedades de cacao presentes en estas plantaciones garantiza que el algoritmo resultante sea versátil y capaz de identificar diversos patrones. Esto contribuye a asegurar su efectividad en un amplio rango de plantaciones de cacao.

Figura 97 : Recolección de Datos para el entrenamiento del Algoritmo de reconocimiento

Figura 98 : Datos para el entrenamiento del algoritmo



El desarrollo del algoritmo comenzó con la creación de un entorno virtual de Python que incluye todas las bibliotecas necesarias, todas ellas con las versiones adecuadas para garantizar la compatibilidad con el hardware de la Raspberry Pi. Es importante señalar que el proceso de entrenamiento del algoritmo se lleva a cabo en una computadora portátil con una tarjeta gráfica

dedicada, lo que permite agilizar significativamente los tiempos de entrenamiento.

El conjunto de datos disponible se divide en dos secciones principales: una sección que se utilizará para entrenar el algoritmo y otra destinada a las pruebas. Cada una de estas secciones se subdivide a su vez en dos carpetas, una con imágenes de frutos de cacao sanos y la otra con imágenes de frutos infectados con moniliasis.

La primera sección de este código se dedica a la inclusión de las bibliotecas necesarias para el proyecto. Estas bibliotecas abarcan tanto TensorFlow como Keras, que son herramientas fundamentales para la implementación de aprendizaje automático. Además, se incorpora una biblioteca que facilita la interacción con archivos y directorios en el sistema operativo. También se importa una clase especializada que posibilita la generación de lotes de datos de imágenes en tiempo real durante el proceso de entrenamiento de un modelo de inteligencia artificial.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import os
from tensorflow.keras import layers
from keras.preprocessing.image import ImageDataGenerator
```

Luego se procede a realizar un proceso de filtrado en nuestro conjunto de datos, donde se llevó a cabo la eliminación de imágenes que presenten corrupciones. Este código se encarga de examinar cada imagen en las carpetas correspondientes, aplicando la eliminación de aquellas imágenes que no sean archivos JPEG válidos, tanto en el conjunto de entrenamiento como en el de prueba.

En una primera instancia, se declara una variable con el propósito de llevar un registro del número de imágenes que se eliminarán. A continuación, se inicia un recorrido por ambas carpetas que componen el conjunto de datos, una destinada a las imágenes de frutos sanos y otra a las imágenes de frutos infectados. Se establece un proceso iterativo que permite obtener la ruta de cada imagen, seguido de la lectura de cada imagen en modo binario.

Posteriormente, se verifica si el archivo en cuestión es un archivo JPEG válido mediante la inspección de los primeros bytes del archivo, buscando la cadena de bytes "JFIF". Luego, sin importar si el archivo se considera válido o no, se procede a cerrar el archivo. En última instancia, en caso de que se determine que el archivo no es válido, se incrementa el valor de la variable de conteo y se procede a la eliminación del archivo.

```
skipped_imgs = 0
for f_name in ('monilia','sano'):
   f_path = os.path.join('training',f_name)
   for fname in os.listdir(f_path):
      fpath = os.path.join(f_path, fname)
      try:
         f_obj = open(fpath,'rb')
         is_jfif = tf.compat.as_bytes('JFIF') in f_obj.peek(10)
      finally:
         f_obj.close()

      if not is_jfif:
         skipped_imgs += 1
         os.remove(fpath)

print('Deleted %d images' %skipped_imgs)
```

En este contexto, tras ejecutar este procedimiento tanto en el conjunto de entrenamiento como en el conjunto de prueba, se llevó a cabo la eliminación de aproximadamente 100 imágenes en total.

A continuación, se procedió a desarrollar un código con el propósito de generar dos conjuntos de datos, uno destinado al entrenamiento y otro al conjunto de prueba. Estos conjuntos serán empleados en las etapas subsiguientes del proceso de entrenamiento y evaluación de un modelo de inteligencia artificial.

Primero se definen algunas variables, la primera es el tamaño en el que se redimensionarán todas las imágenes para poder tenerlas normalizadas, la segunda variable es el tamaño de lotes utilizados durante el entrenamiento, esto es, cuántas imágenes se procesan a la vez durante cada iteración de entrenamiento, y la última variable que se define es el tamaño de la entrada del

modelo de inteligencia artificial las dimensiones en pixeles con 3 canales de color.

Después se tiene el generador de datos tanto del entrenamiento y test en ambos se realiza una normalización de las imágenes, y para el caso del conjunto de entrenamiento se agregan algunos parámetros que aplica transformaciones a las imágenes de entrenamiento, con el fin de aumentar la variabilidad en el conjunto de datos.

Finalmente, ya se crea los conjuntos de datos de entrenamiento y test, mediante el generador de datos que ya definimos y le pasamos también las variables del tamaño de la imagen, tamaño de los lotes y se establece un modelo de clasificación binaria.

```
image_size = (180,180)
batch_size = 128
input_shape = (180,180,3)

train_datagen = ImageDataGenerator (
   rescale = 1./255,
   shear_range = 0.2,
   zoom_range = 0.2,
   horizontal_flip=True
)

test_datagen = ImageDataGenerator(
   rescale = 1./255
)


training_dataset = train_datagen.flow_from_directory(
   'training',
   target_size = image_size,
   batch_size = batch_size,
   class_mode = 'binary'
)

testing_dataset = test_datagen.flow_from_directory(
   'test',
   target_size = image_size,
```

```
    batch_size = batch_size,
    class_mode = 'binary'
)
```

Figura 99 : Visualización de Datos por parte del Algoritmo

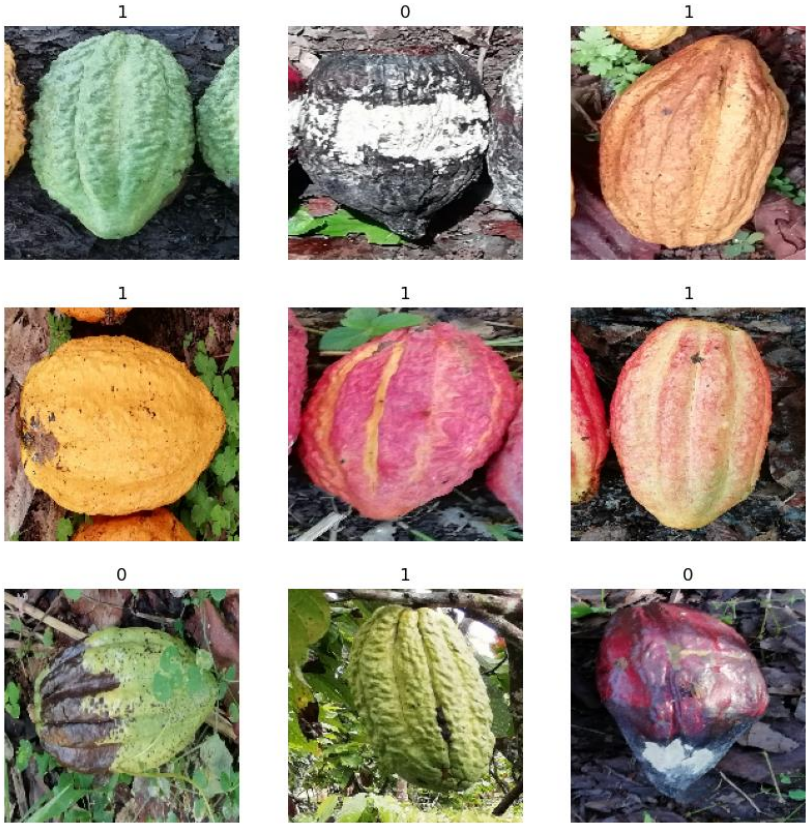


Figura 100 : Transformaciones lineales en el conjunto de Entrenamiento del Algoritmo

Después se pasó con la creación del modelo, existen varios algoritmos que pueden utilizarse para esta tarea, entre los más conocidos se tienen las redes neuronales convolucionales (CNN), redes neuronales recurrentes (RNN), redes neuronales profundas (DNN), redes neuronales generativas (GANs), etc. Dentro de estas opciones se escogió hacer uso de las redes neuronales convolucionales, debido a que es un algoritmo altamente efectivo en la extracción de características visuales de las imágenes, es un algoritmo que posee buena adaptabilidad a datos de imágenes, además de que las CNN han demostrado su eficacia en una amplia variedad de aplicaciones de visión por computadora, incluido el reconocimiento de objetos y patrones en imágenes.

Se procedió en la construcción de este modelo haciendo uso de la API funcional e Keras para crear una arquitectura de red neuronal convolucional (CNN). Se define un modelo secuencial, que es una pila lineal de capas. Las capas fluyen a través del modelo en el orden en que se definen las capas. Empezando el modelo se define la capa de entrada donde se especifica el tamaño de entrada de las imágenes que se estableció previamente. Luego de hecho esto se pasó con el desarrollo de las capas convolucionales, la primera

capa tiene 32 filtros de tamaño (3,3) y utiliza la función de activación de rectificador lineal unitario para introducir no linealidad, después de cada capa convolucional, hay una capa de reducción, que reduce el tamaño de las características extraídas y ayuda a reducir la cantidad de parámetros en el modelo; se tiene una segunda capa con 64 filtros de tamaño (3,3) y también utiliza la función de activación de rectificador lineal unitario. Después de las capas convolucionales se utiliza una función para aplanar la salida de las capas convolucionales en un vector unidimensional. Esto es necesario para conectar las capas densas que siguen. Después se tiene una capa para regularizar el modelo y reducir el sobreajuste, con una probabilidad del 50%, se “desactivan” aleatoriamente algunas neuronas durante el entrenamiento, lo que ayuda a evitar que el modelo se ajuste demasiado a los datos de entrenamiento. Finalmente tenemos las capas densas, la primera capa densa tiene 128 neuronas y utiliza la función de activación de rectificador lineal unitario, esta capa realiza la primera etapa de la clasificación y aprende características de nivel superior. La última capa densa tiene una sola neurona y utiliza la función de activación sigmoide. Esta capa produce la salida final del modelo y realiza la clasificación binaria, donde un valor cercano a 0 representa una clase y un valor cercano a 1 representa la otra clase. En resumen, el modelo que se presenta a continuación es una red neuronal convolucional simple pero efectiva para la clasificación binaria de imágenes. Comienza con capas convolucionales para extraer características, seguidas de reducción de dimensiones, aplanamiento y capas densas para realizar la clasificación final. La capa de dropout ayuda a prevenir el sobreajuste durante el entrenamiento.

```
model = keras.Sequential (
  [
    keras.Input(shape = input_shape),
    layers.Conv2D(32, kernel_size=(3,3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
    layers.MaxPooling2D(pool_size=(2,2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(128, activation='relu'),
```

```
    layers.Dense(1, activation='sigmoid')
  ])
```

Al cabo de haber realizado el modelo se pasó con su entrenamiento, primero se define el número de épocas, esto define cuántas veces se recorrerá todo el conjunto de datos durante el entrenamiento. Se realiza la compilación del modelo antes de entrenarlo y se definen algunos aspectos clave, se utiliza la función de pérdida de entropía cruzada binaria, que es comúnmente utilizada en problemas de clasificación binaria, esta función de pérdida mide la discrepancia entre las predicciones del modelo y las etiquetas reales; se utiliza un optimizador para ajustar los pesos del modelo durante el entrenamiento y se utiliza la métrica de precisión durante el entrenamiento para evaluar qué tan bien el modelo se está desempeñando en el conjunto de datos de entrenamiento y en el conjunto de datos de validación. Ya con esto se inicia el proceso de entrenamiento del modelo, donde se define el conjunto de datos de entrenamiento que se utiliza para entrenar el modelo, se pasa la variable definida anteriormente con el número de épocas de entrenamiento y finalmente se proporciona un conjunto de datos de validación para evaluar el rendimiento del modelo después de cada época. Después de completar todas las épocas de entrenamiento, se guarda el modelo con el fin de poder cargarlo y utilizarlo en la raspberry pi sin tener que volver a entrenarlo desde cero.

```
epochs = 25

model.compile(
  loss='binary_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

model.fit(
  training_dataset,
  epochs = epochs,
  validation_data = testing_dataset
)

model.save('p37_models/p37_m_25_adam.h5')
```

Este modelo fue entrenado usando diferentes parámetros de los mencionados anteriormente y se obtuvieron los siguientes resultados:

Para epochs=25 y optimizer=adam:

- Precisión en entrenamiento: 96.98%
- Pérdida en entrenamiento: 0.0805
- Precisión en validación: 96.36%
- Pérdida en validación: 0.1101

Para epochs=32, optimizer=adam:

- Precisión en entrenamiento: 97.84%
- Pérdida en entrenamiento: 0.0623
- Precisión en validación: 95.52%
- Pérdida en validación: 0.1421

Para epochs=25 y optimizer=rmsprop:

- Precisión en entrenamiento: 94.82%
- Pérdida en entrenamiento: 0.1350
- Precisión en validación: 83.75%
- Pérdida en validación: 0.4309

Para epochs=32 y optimizer=rmsprop:

- Precisión en entrenamiento: 94.77%
- Pérdida en entrenamiento: 0.1406
- Precisión en validación: 94.68%
- Pérdida en validación: 0.1304

De estos resultados se observa que los modelos entrenados con el optimizador "adam" tienden a tener una precisión más alta tanto en el conjunto de entrenamiento como en el conjunto de validación en comparación con los modelos entrenados con "rmsprop". El modelo con 32 épocas tiende a tener una precisión ligeramente más alta en el conjunto de entrenamiento en comparación

con el modelo con 25 épocas, pero la precisión en el conjunto de validación es ligeramente más baja. Esto puede indicar un ligero sobreajuste en el modelo con 32 épocas. El modelo con 25 épocas y "adam" parece ser una buena elección, debido a que tiene una alta precisión en ambos conjuntos y una pérdida relativamente baja en comparación con el modelo con 32 épocas. El modelo con "rmsprop" tiene una precisión significativamente menor en ambos conjuntos y una pérdida más alta, por lo que no parece ser la mejor opción en este caso.

Basándonos en estos resultados, el modelo con 25 épocas y optimizador “adam” parece ser la mejor opción para el reconocimiento de frutos de cacao infectados con moniliasis, dado que ofrece una alta precisión en el conjunto de validación y es menos propenso al sobreajuste. Este es el modelo que se integró en la raspberry pi.

3.6.11. Integración del Modelo de Reconocimiento a la Raspberry Pi

Luego de haber completado el algoritmo de reconocimiento de moniliasis se pasa a su integración dentro de la raspberry pi ya teniendo instalado correctamente las versiones de todas las librerías.

El modelo se agregó al script principal de nuestro programa principal que será el encargado de cargar el modelo al iniciarse el programa, para lograr esto, se importan las librerías necesarias, que en este caso son tensorflow y keras, y luego se importa el modelo que previamente ya guardamos en un archivo H5, también para esta importación se utiliza un iniciador para que el modelo pueda cargarse correctamente con todas las configuraciones originales y evitar errores. Además, también definimos el tamaño de imagen que se estará utilizando, que es el mismo que se usó para entrenar el modelo.

```
import tensorflow as tf
from tensorflow import keras
model =
tf.keras.models.load_model('/home/greenskull/Desktop/raspberry/p37_models/p3
7_m_25_adam.h5', custom_objects={'GlorotUniform':
tf.keras.initializers.GlorotUniform})
image_size = (180,180)
```

En esta etapa, se ha realizado la importación exitosa del modelo en la Raspberry Pi. Sin embargo, aún no se dispone de una interfaz para visualizar los resultados del análisis del modelo. Del mismo modo, al presionar el botón de la cámara en la aplicación móvil, la foto se toma, pero carece de cualquier indicador visual para el usuario. Este aspecto se abordará en detalle. Para solucionarlo, se establecerá un segundo servidor web en la Raspberry Pi. Este servidor operará en el mismo puerto que el servidor utilizado para visualizar las imágenes de la cámara de la Raspberry Pi. Esto se hace para mantener la integridad de la aplicación móvil y porque, como se mencionó anteriormente, la petición de la imagen de la cámara se puede utilizar en un único script.

Teniendo esto en cuenta, se modificó un poco el código cuando se recibe el comando de la ejecución de la cámara desde la aplicación android, para ello primero se define una variable booleana que se llamó process, esta nos indica si se está ejecutando el proceso de visualización de la imagen de la cámara en tiempo real o no, la cual se inicializó con el valor de 'true', también se tiene una variable llamada 'params' que se inicializa con el valor de 'MONILIA', esto es el parámetro que le pasamos al servidor con el fin de que sepa que valor mostrar, si el análisis después de pasar por el modelo de inteligencia artificial determina que es un fruto sano o un fruto infectado con moniliasis, si se está ejecutando la transmisión de imagen, primero se para la ejecución de este script, se corre el script para tomar la foto y almacenarla localmente, después de tener la imagen que se quiere analizar primero se convierte la imagen a un array debido a que la entrada del modelo debe ser un arreglo de numpy que representa la imagen, después se realiza una expansión de dimensiones de este arreglo que representa a la imagen, esto se hace para que la imagen tenga una dimensión adicional al principio, puesto que el modelo espera un lote de imágenes como entrada, y un lote generalmente tiene una dimensión adicional para la cantidad de imágenes en el lote, en este caso lo que se busca es expandir las dimensiones en el primer eje (lote), lo que crea un lote de una sola imagen. Después se realiza la predicción sobre la imagen y dependiendo de la predicción se cambia la variable de parámetros a 'SANO' en caso sea necesario y después se ejecuta el servidor para visualizar dentro de la aplicación Android el análisis de este modelo, este script lo programamos a continuación, después se cambia el valor de la variable 'process' para indicar que ya no se está ejecutando el servidor de

transmisión de imagen en tiempo real. En caso de que éste ya no se esté ejecutando terminamos el proceso del servidor de resultados y cambios de nuevo hacia el servidor de transmisión de la imagen.

Con esto se procedió con el desarrollo del servidor donde se muestran los resultados del análisis efectuado por el modelo importado. Para la creación de este script primero se importan las librerías necesarias y se guarda en una variable el parámetro recibido al levantar el servidor, se define la estructura HTML que se muestra en el servidor, en una parte se muestra la imagen analizada y en otra el resultado en base al parámetro recibido.

Después se crea el código para manejar las solicitudes que llegan al servidor. En este caso se tienen 2 tipos de solicitudes, la solicitud para devolver la página HTML que se definió anteriormente y la solicitud que devuelve la imagen que fue tomada para su respectivo análisis y para que ésta pueda ser utilizada por nuestra página HTML de la siguiente manera:

Finalmente se inicia el servidor web en la misma dirección y puerto que el servidor de transmisión de imagen para que pueda ser visualizado con el mismo componente webview en la aplicación Android (véase en el **Anexo 12**).

Después de haber terminado la integración del algoritmo de inteligencia artificial en la raspberry pi se procedió a realizar pruebas estáticas en 2 plantas de cacao de variedades diferentes, para ver la respuesta de todo el sistema frente a condiciones de campo.

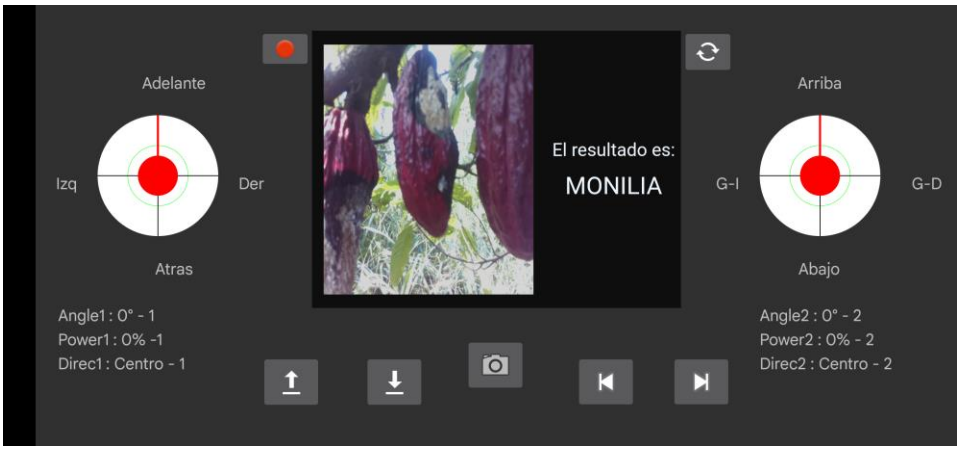
En estas pruebas se pudo evidenciar que el algoritmo responde correctamente, pero debido a la limitante en la resolución de la cámara de la raspberry pi, solo responde correctamente cuando existe buena iluminación en el fruto, si existe demasiada sombra el porcentaje de acierto del algoritmo baja considerablemente.

Figura 101 : Dron completo

Figura 102 : Plantas de cacao para prueba estática del algoritmo

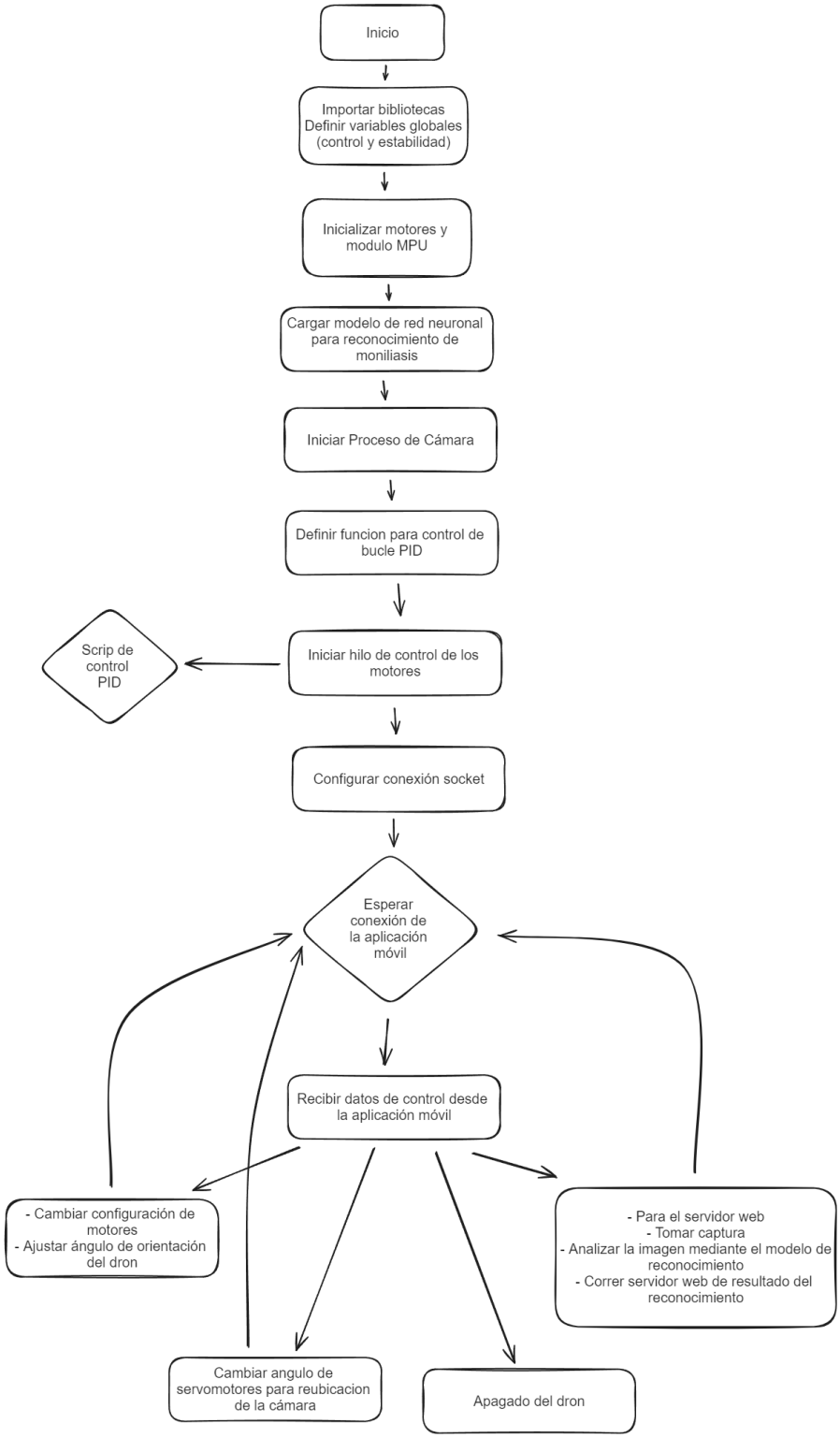
Figura 103 : Prueba estática del algoritmo

Figura 104 : Resultado de las pruebas desde la Aplicación Móvil



3.6.12. Pruebas del dron terminado

Figura 105 : Diagrama de Flujo del software del Vehículo Aéreo no Tripulado



Ya teniendo todo software y hardware que se utilizará se procedió con las primeras pruebas de vuelo para el dron.

El proceso de prueba inicial del dron en un entorno seguro reveló varios problemas sustanciales. En primer lugar, se evidenció que el dron no estaba adecuadamente calibrado para despegar. Además, se enfrentó a un desafío significativo al intentar levantar todo el peso del dron, lo que generó un alto nivel de esfuerzo por parte de los motores. Asimismo, se observó que la disposición de los componentes en el dron no era la más adecuada.

Figura 106 : Primera prueba de vuelo del dron

Como respuesta a estas dificultades, se implementaron ajustes importantes tanto en el hardware como en la composición del dron. En primer lugar, se optó por sustituir las hélices originales del dron por unas hélices DJI Phantom, las cuales resultaron ser compatibles con los motores seleccionados. Estas hélices, además de su compatibilidad, demostraron ser más resistentes y eficientes, lo que se tradujo en una mejora significativa en la capacidad de despegue del dron. Además de este cambio, se decidió reemplazar la batería original por una de 2200mAh, reduciendo así el peso total del dron.

Estos ajustes iniciales prepararon el terreno para una redistribución de los componentes en el dron. Con el objetivo de reducir aún más el peso y mejorar la estabilidad general, se retiró el protoboard trasero y se instaló uno de menor tamaño en el centro del dron, donde se alojaría la unidad de medida inercial (IMU). Además, se realizaron las conexiones necesarias para la PCA9685, que también se ubicaría en el centro del dron y estaría encargada de gestionar la

modulación por ancho de pulsos (PWM) de los sensores. Para mantener el peso reducido y al mismo tiempo mejorar la estabilidad del dron, se eliminó la base que anteriormente sostenía la batería y se optó por una configuración que anclaba la batería directamente a la estructura base en la parte central.

En resumen, estos cambios y ajustes colectivos contribuyeron a lograr un diseño más ligero y eficiente que, además de permitir un despegue más efectivo, garantizaba una mayor duración de vuelo del dron.

Figura 107 : Inicialización del programa con la carga del modelo de reconocimiento de moniliasis

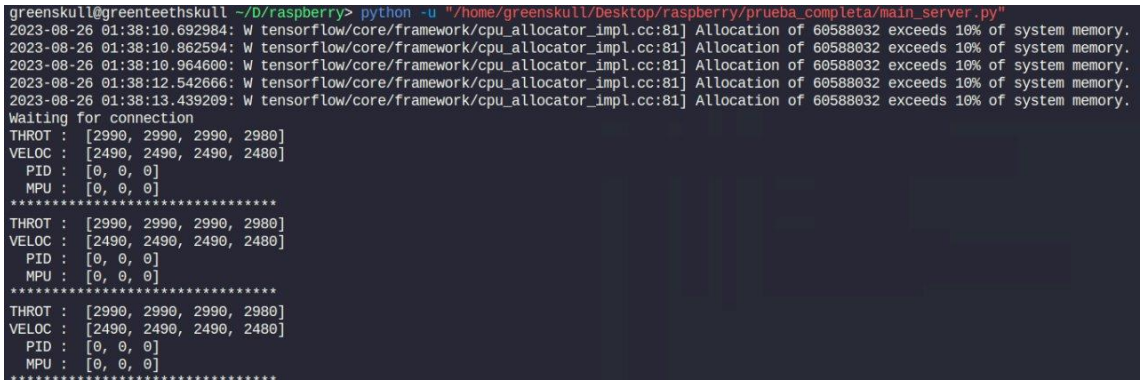
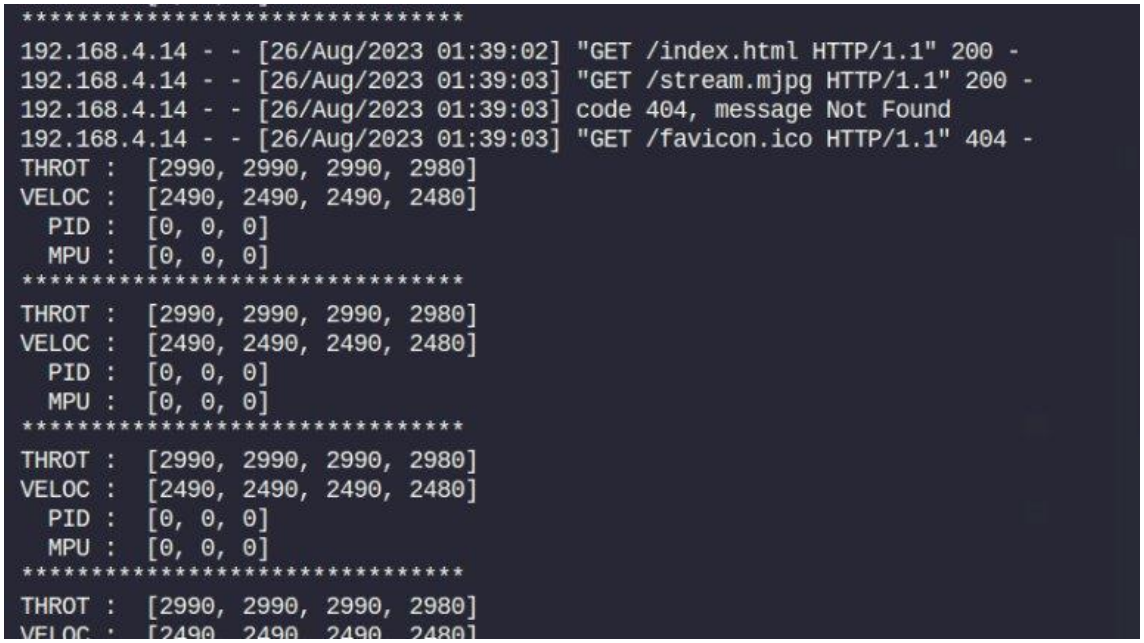


Figura 108 : Mensaje de conexión exitosa al servidor web por parte de la aplicación móvil



Se realizó las pruebas pertinentes empezando con valores de las constantes proporcional y derivativo respectivamente con valores de 1 y

cambiando ligeramente cada uno de estos valores en cada prueba, con el fin de mantener libre al dron se empezó probando únicamente con la mano tal y como se muestra a continuación.

Figura 109 : Prueba de respuesta a los controles del dron

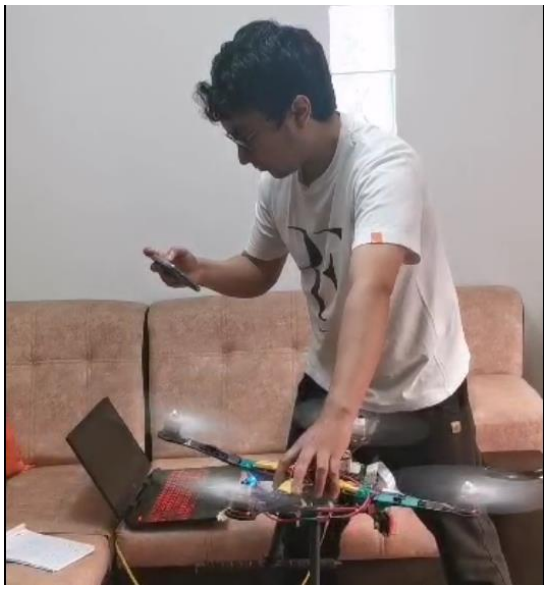


Figura 110 : Prueba de respuesta del algoritmo de estabilidad

Esta manera de probar representaba un peligro latente de corte y era muy insegura, por lo que empezó a probarse amarrado en una base a unos 60cm del suelo, y con una libertad para que pueda tomar vuelo a una baja distancia. Al realizar las pruebas tratando de tunear las constantes del PID, también monitoreamos algunas variables como el valor de los ángulos de inclinación del dron, la salida del PID, la velocidad base de los motores y la velocidad real de los mismos. Dentro de las pruebas al cambiar los valores de las constantes del PID el dron no se estabilizaba, la razón era que nuestra imu cambiaba

rotundamente los valores de los ángulos aun cuando el dron permanecía casi estático, con ligeras vibraciones propias de los motores; cuando los motores permanecían apagados pero todo el sistema funcionando la imu respondía correctamente, pero cuando los motores empezaban a rotar la imu se inestabilizaba, cuanto más rpm generaban los motores, más inestable era la imu. Esto sucede por el campo magnético que generan los motores que interviene con el magnetómetro incorporado en la imu.

Figura 111 : Posición de inicio para prueba de estabilidad

Figura 112 : Posición final después de prueba de estabilidad

Debido a esta dificultad se decidió cambiar de imu al módulo MPU6050 porque permite proporcionar lecturas más estables de los ángulos de inclinación puesto que es menos sensibles a interferencias magnéticas, además en este

módulo no es necesario realizar la calibración ni corrección del magnetómetro, también otra ventaja es que este módulo es más económico y proporciona una velocidad de lectura más rápida, todo esto hace que este módulo también se utiliza en varias soluciones de controladores de drones comerciales.

Figura 113 : Módulo MPU6050

Una vez elegido el nuevo módulo para reemplazar a la unidad de medida inercial, se tuvo que programar el script correspondiente para leer los ángulos de inclinación del dron, cuyo código puede verse en el **Anexo 16**.

Primero como todos los scripts anteriores, se inició importando las librerías necesarias, en este caso es la librería que contiene la clase que nos permite interactuar con el sensor y también una librería para realizar cálculos matemáticos necesarios en el código.

Después se definió una función setup() en la que se inicializan las variables globales a utilizar y también se configura el sensor MPU6050. Se crea una instancia de la librería importada anteriormente y como parámetro le pasamos la dirección I2C del sensor. También se inicializan los valores de estos ángulos de inclinación que servirán como una corrección del ángulo de inclinación propio del sensor, estos valores fueron obtenidos leyendo continuamente el sensor en reposo hasta la estabilización de la lectura.

Después se define una función de lectura del acelerómetro y del giroscopio del sensor y se devuelve como un par de diccionarios:

Después se creó la función que calculará los ángulos de inclinación a partir de los datos de aceleración y giroscopio. Estos ángulos se obtienen mediante un filtro pasa bajo que combina las lecturas del acelerómetro y el

giroscopio para obtener lecturas suavizadas y estables de los ángulos de inclinación.

El ángulo roll se calcula como el ángulo entre el eje Y y la componente Z del vector de aceleración. Esto proporciona el ángulo de inclinación en el plano XY con respecto al eje Z. El ángulo pitch se calcula de manera similar, utilizando el ángulo entre el eje X y la magnitud de la componente del vector de aceleración en los ejes Y y Z. Esto proporciona el ángulo de inclinación en el plano XZ con respecto al eje Y.

El filtro pasa bajo es esencialmente una medida ponderada entre las lecturas del giroscopio y las lecturas del acelerómetro donde el factor de ponderación es ‘alpha’ y se utiliza para determinar cuánto peso se les da a las lecturas del giroscopio en comparación con las lecturas del acelerómetro. El nuevo ángulo roll se calcula como una interpolación lineal entre el ángulo calculado a partir del giroscopio y el ángulo calculado a partir del acelerómetro de manera similar se aplica para el ángulo calculado a partir del giroscopio.

El ángulo Yaw se actualiza simplemente agregando las lecturas del giroscopio en el eje Z multiplicadas por el intervalo de tiempo determinado. Esto permite que el dron mantenga un seguimiento del ángulo de orientación alrededor del eje vertical. Además, se agrega la corrección de ángulos para que se encuentren en el intervalo de -180 a 180.

La elección de alpha 0.98 significa que se les da más peso a las lecturas del giroscopio, lo que proporciona una respuesta más rápida, este valor es común en aplicaciones de filtros pasa bajo, puesto que brinda una buena combinación de estabilidad y capacidad de respuesta.

Finalmente se programó la función principal que hace uso de las funciones anteriores para devolver los ángulos roll, pitch y yaw.

Como se observa en cuanto a los nombres de las funciones se siguió manteniendo el mismo patrón que para el MPU9250, esto para que sea más fácil su integración en el programa principal.

Ya habiendo hecho las modificaciones correspondientes, se procedió a realizar las pruebas de tuneo de las constantes del control PID, haciendo uso del mismo mecanismo anteriormente mencionado. Se empezó probando las

constantes proporcionales y derivativas, de donde se pudo observar los siguientes comportamientos en base a estos cambios:

Figura 114 : Pruebas de Estabilidad del dron

Figura 115 : Visualización de parámetros por consola para calibración

```
********************************
THROT :  [4110, 4110, 4110, 4100]
VELOC :  [4115, 4109, 4105, 4101]
  PID :  [-2, 3, 0]
  MPU :  [-3, 4, 9]
********************************
THROT :  [4110, 4110, 4110, 4100]
VELOC :  [4117, 4109, 4103, 4101]
  PID :  [-3, 4, 0]
  MPU :  [-3, 4, 10]
********************************
THROT :  [4110, 4110, 4110, 4100]
VELOC :  [4117, 4109, 4103, 4101]
  PID :  [-3, 4, 0]
  MPU :  [-3, 4, 10]
********************************
THROT :  [4110, 4110, 4110, 4100]
VELOC :  [4117, 4109, 4103, 4101]
  PID :  [-3, 4, 0]
  MPU :  [-3, 4, 10]
********************************
THROT :  [4110, 4110, 4110, 4100]
VELOC :  [4117, 4109, 4103, 4101]
  PID :  [-3, 4, 0]
  MPU :  [-3, 4, 10]
********************************
```

Tabla 29 : Pruebas de constantes PID

P	D	Acccion
1	1	No Realiza Ninguna Accion De Control
1.5	5	Tambaleo Suave
2	5	Tambaleo Brusco
2	7	Vuelo Inestable
2.5	6.5	Responde Correctamente De Manera Suave
3	6	Responde Inestablemente
3.5	4.5	Vuelo Con Respuesta Suave
3.5	6	Vuelo Con Respuesta Brusca
4	4	Tambaleo Brusco

En base a estos resultados se logró tunear el PID con los valores siguientes:

Constante proporcional = 3.67

Constante derivativa = 4.95

Constante integral = 0.05

Con ello ya se tuvo todo el sistema completo y únicamente faltaba la prueba completa en campo donde se evaluaría el funcionamiento tanto de todo el hardware y software implementado en el dron.

Estas pruebas se realizaron en una de las plantaciones de la Universidad Nacional Agraria de la Selva de donde también se sacaron las fotos para el entrenamiento del algoritmo de inteligencia artificial, las pruebas se realizaron con 4 variedades de cacao para poder comprobar cómo responde el algoritmo en base a frutos con diferentes colores, tamaños y formas. Además, también se lograron encontrar síntomas diferentes en cuanto a la infección de moniliasis, con manchas, esporulación o ambas.

Figura 116 : Dron en Campo para Realizar pruebas

Figura 117 : Frutos que se usaron para las pruebas del dron

Figura 118 : Visualización de la cámara del dron desde la aplicación móvil

Figura 119 : Prueba de vuelo del dron por medio de la plantación

3.6.13. Resultados Obtenidos

Después de realizadas las pruebas se pudo comprobar que:

El algoritmo de control por PID logró estabilizar de manera correcta el dron en vuelo, pero para poder obtener un vuelo estable también es necesario que el entorno cuente con las condiciones adecuadas, cuando existe mucho viento, el dron no logra estabilizarse y el vuelo se torna peligroso, también otro de los aspectos es que no se puede volar bajo condiciones de lluvia, debido a que esto dañaría los circuitos internos del dron.

En cuanto al tiempo de vuelo se logró confirmar que el dron mantiene un vuelo estable durante 25 minutos y este logra llegar a una duración total de 40 minutos, pero perdiendo eficiencia en los motores por lo que lo más recomendable es sobrevolar el dron solo durante 30 minutos.

Figura 120 : Dron volando en medio de la plantación de cacao

El algoritmo de reconocimiento logró reconocer con un gran porcentaje de acierto los frutos de cacao infectados con moniliasis y diferenciarlos con los frutos sanos. Como se mencionó estos debe tener la luz suficiente para que puedan ser reconocidos los colores y las formas de los frutos, pero tampoco la cámara tiene que estar expuesta directamente en el sol debido a que esto causaría una sobresaturación en la imagen.

Se probó el dron en 4 clones de cacao con presencia de moniliasis, estos clones presentan variabilidad en cuanto al color, forma y grado de infección del fruto. En total la prueba se realizó con 80 frutos de los 4 clones mencionados, y

en condiciones de campo el algoritmo logró reconocer 75 frutos de manera correcta dando una eficiencia del 93.75%, lo que demuestra la efectividad de este dispositivo.

Figura 121 : Reconocimiento de frutos sanos de cacao

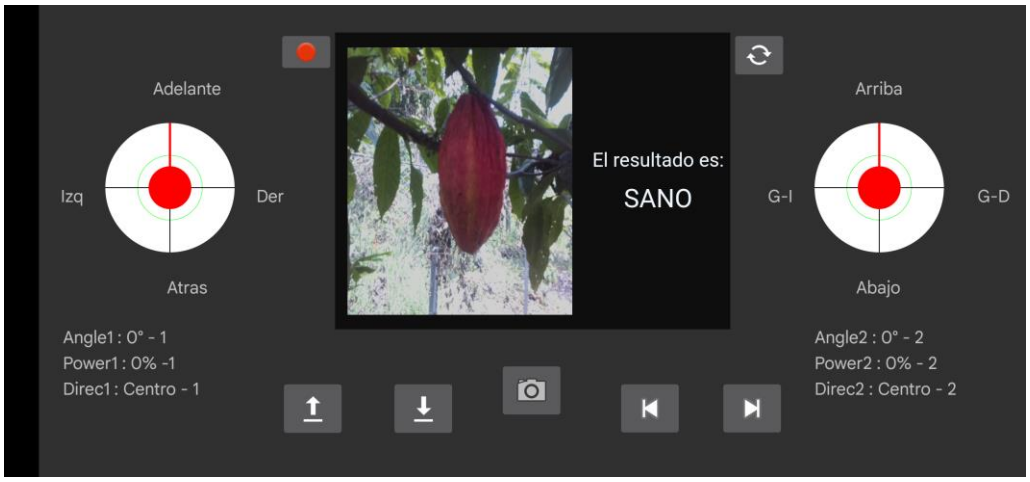
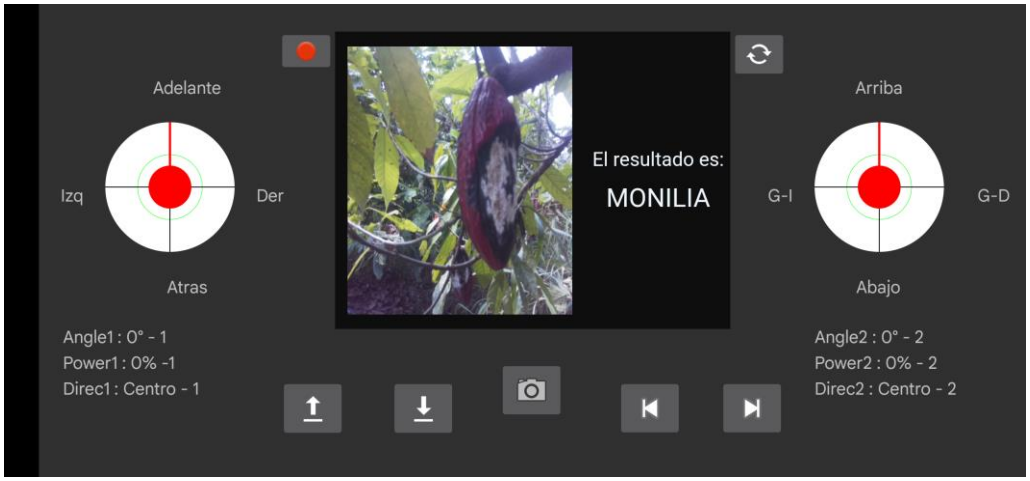


Figura 122 : Reconocimientos de frutos de cacao infectados con moniliasis



Todo el programa del dron mostró una buena eficiencia en conjunto, con todos los procesos ejecutándose, la tasa de ejecución del control PID de 100ms demostró ser eficiente para este tipo de aplicaciones, aunque si es cierto que en ocasiones debido a este proceso la imagen de la cámara mostrada en el servidor web presenta bajones de cuadros por segundo, pero en general la imagen es bastante estable por lo que la raspberry pi fue una buena elección para esta aplicación.

Además también se consiguió un diseño robusto de bajo precio como se muestra en la Tabla 7.24, que a diferencia de los drones comerciales cuyo costo ronda los 2000 dólares, esta se presenta como una alternativa accesible para cualquier persona además de incluir un sistema de inteligencia artificial para el reconocimiento de moniliasis en el cacao todo integrado, que en el caso de los drones comerciales este tipo de aplicaciones no suele venir con este tipo de aplicaciones y para los casos que lo incluyan este viene con un costo adicional, además de requerir otros componentes aumentando su precio original drásticamente.

Tabla 30 : Costos de Componentes en el Dron

Componente	Costo (Soles)
Frame	120
Raspberry Pi	400
Camara	137
Mpu6050	18
Pca9685	40
Bateria	130
ESC X4	120
MOTORES BRUSHLESS X4	280
HELICES X4	120
Otros Componentes	135
Total	1500

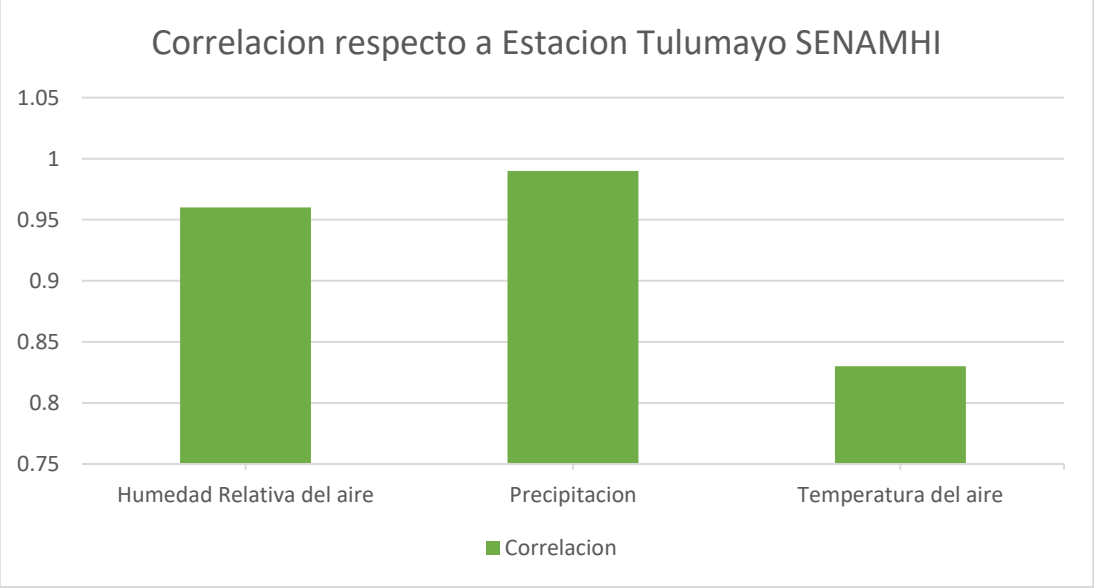
Capítulo IV. Análisis y discusión de resultados

4.1. Contrastación de hipótesis

La estación climática proporcionó información precisa y fiable para la toma de decisiones informadas en el cultivo de cacao, especialmente en microclimas de Tingo María.

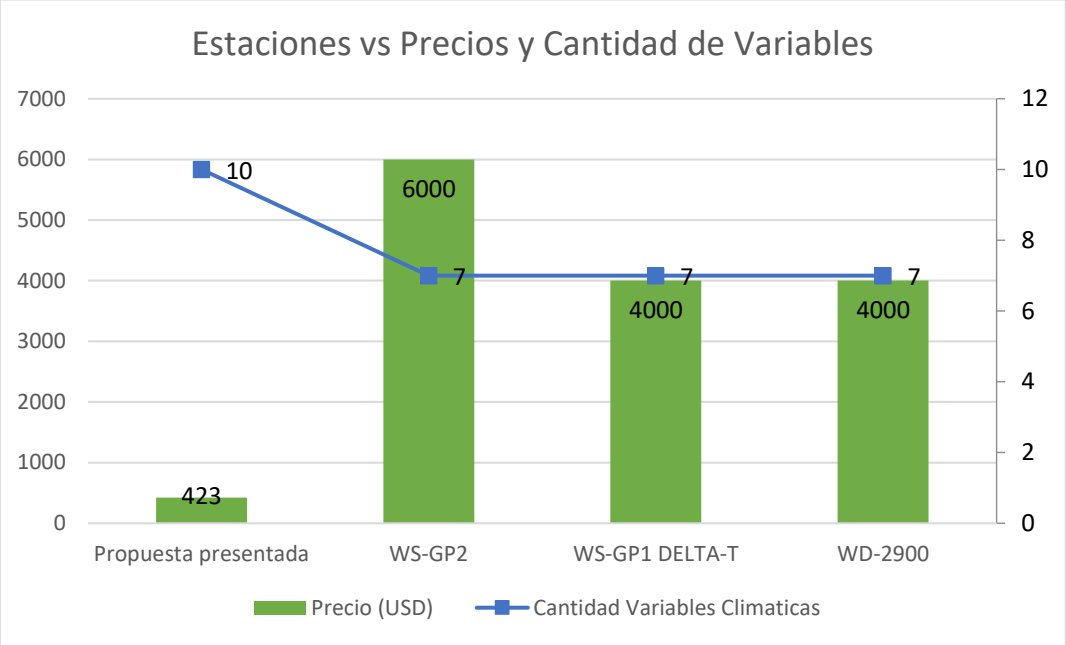
De acuerdo con los gráficos vistos en la sección 7.1.3 comparando esta propuesta con la del SENAMHI mostraron altas correlaciones de entre 0.83 y 0.96.

Gráfico 10 : Correlaciones de datos de la estación climática diseña y la estación Tulumayo del SENAMHI



La estación climática demostró ser una alternativa de bajo costo y robusta, abarcando 10 variables climáticas, frente a soluciones comerciales más costosas.

Gráfico 11 : Comparación entre la estación climática diseñada y estaciones climáticas comerciales



Se logra evidenciar la necesidad de estaciones climáticas localizadas debido a que en regiones como la selva peruana se crean microclimas a distancias muy cortas lo que afecta al cultivo de distintas maneras y en cada plantación es necesario aplicar métodos de adaptabilidad climática diferentes.

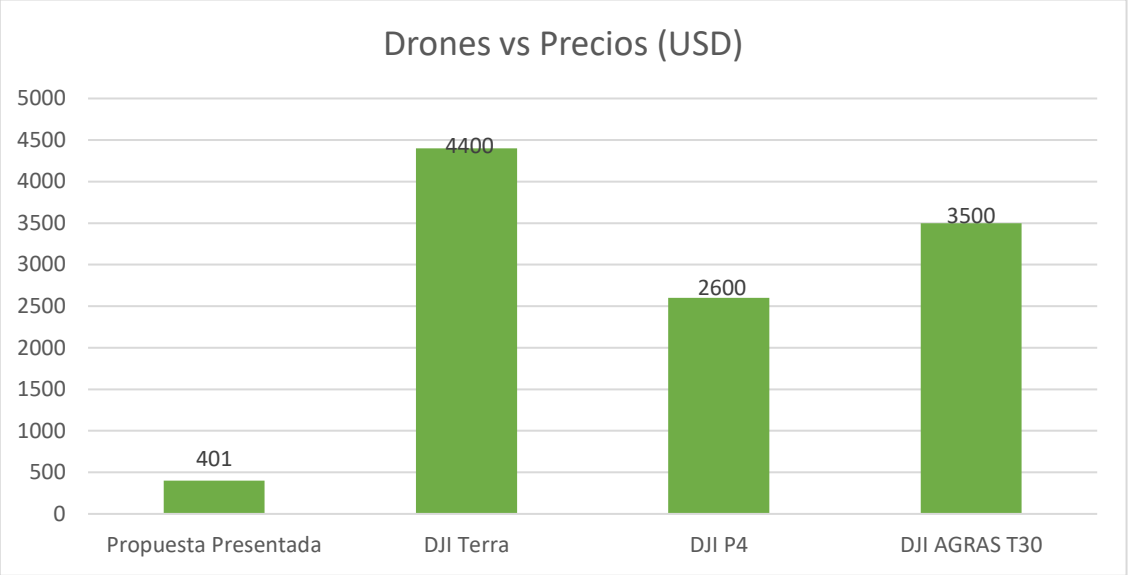
La integración de una plataforma IoT al monitoreo de datos climáticos facilita en gran medida la toma de decisiones para la adaptabilidad climática del cultivo de cacao.

También se evidencio que es mejor contar con una fuente de internet más estable para que no exista pérdida de información y los datos sean más reales.

Se recomienda reconsiderar el material del pluviómetro, anemómetro y veleta para mayor durabilidad y cumplimiento de normativas.

El vehículo aéreo no tripulado demostró ser una alternativa de bajo costo y robusta, además de integrar una solución de reconocimiento de moniliasis en el cacao mediante un algoritmo de inteligencia artificial.

Gráfico 12 : Comparación entre propuesta de vehículo aéreo no tripulado y opciones comerciales



El vehículo aéreo no tripulado demostró que se cuenta con una alta eficacia en el reconocimiento de frutos infectados con moniliasis en diferentes variedades de cacao.

El vehículo aéreo no tripulado también respondió de manera eficiente en base a la arquitectura de software y de hardware que se diseñó.

Se logró un tiempo de vuelo estimado de 30 minutos y también validar que cuando existe poca luz natural hacia los frutos el error del algoritmo de reconocimiento crece considerablemente, tal como muestra la **Figura 123**. Además, tampoco tiene que estar expuesta directamente en el sol para no causa sobresaturación de la imagen como el caso de la **Figura 124.**

Figura 123 : Reconocimiento en condiciones de mucha sombra

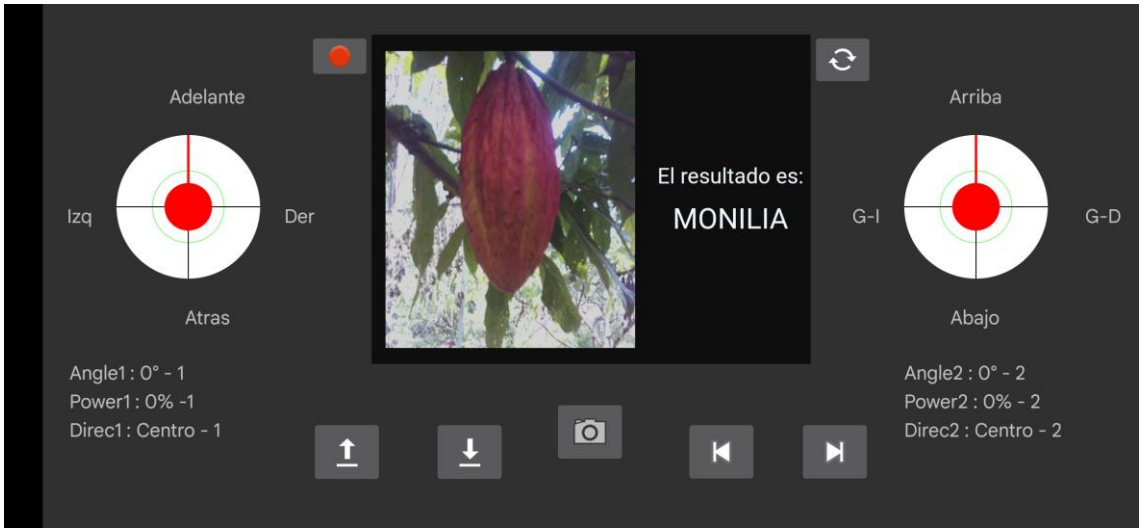


Figura 124 : Reconocimiento cuando la cámara es expuesta al sol directamente

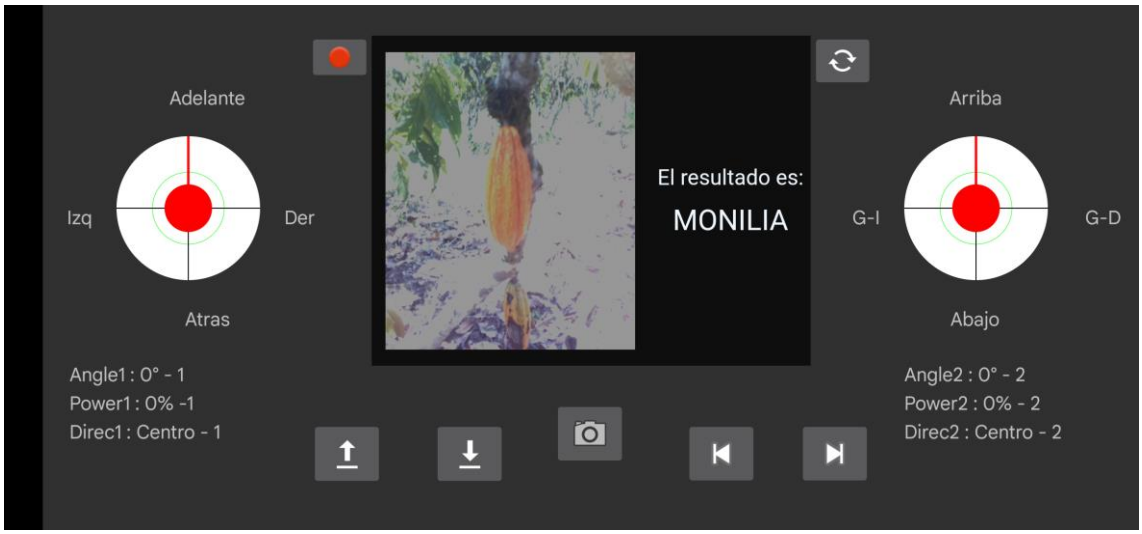
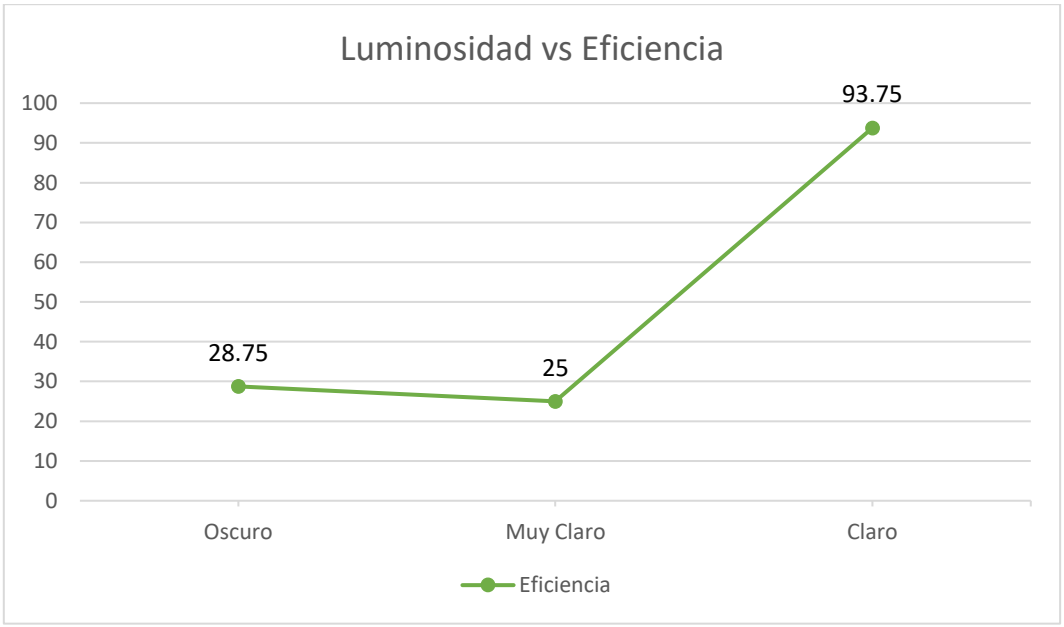


Gráfico 13 : Eficiencias del algoritmo de reconocimiento bajo diferentes condiciones climáticas



Se verificó la estabilidad y eficiencia de la estación climática automática con material recomendado, haciendo revisiones y registros continuos en todas las pruebas de campo.

Costo inferior a soluciones comerciales, manteniendo calidad y aplicaciones adicionales.

La estación climática proporcionó datos precisos y fiables en tiempo real para la mejora de la adaptabilidad climática del cultivo de cacao y una integración exitosa con la base de datos para almacenamiento.

El dron con algoritmo PID logró vuelo estable y reconocimiento eficiente de frutos infectados con moniliasis.

El dron logró una eficiencia del 93.75% en reconocimiento de frutos infectados.

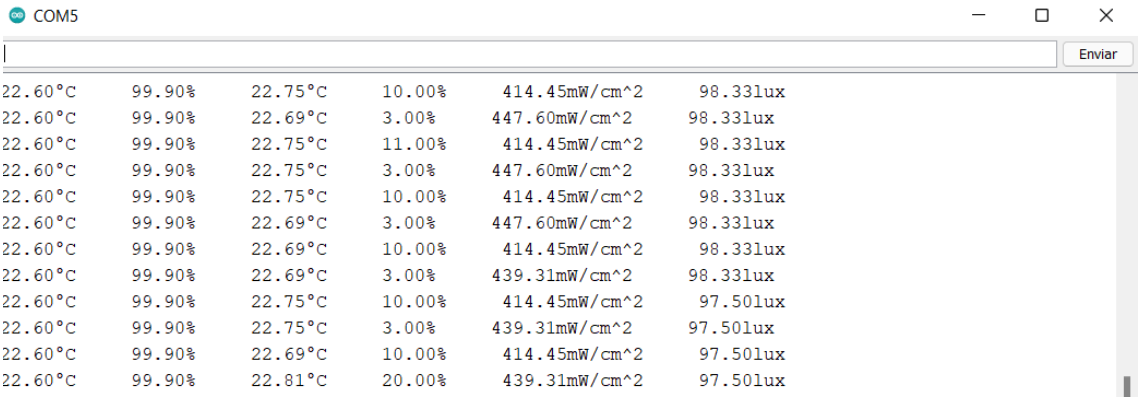
La estación climática y el vehículo aéreo no tripulado, validados, proporcionaron datos clave para la toma de decisiones y la mejora del sistema productivo del cacao.

Se verifica que la hipótesis es positiva debido a que la estación climática inteligente y el vehículo aéreo no tripulado basado en raspberry pi, representan una ayuda significativa para la mejora en el sistema productivo de cacao.

4.2. Validación de indicadores

Los sensores y microcontroladores seleccionados lograron proporcionar resultados estables y utilizables para los estudios de adaptabilidad climática del cultivo de cacao.

Figura 125 : Vista de toma de datos de los sensores de la estación en monitor serial de Arduino



El algoritmo de reconocimiento es eficiente en la diferenciación de frutos de cacao infectados y sanos, con clones de variados colores y texturas, tal como muestra las imágenes a continuación.

Figura 126 : Reconocimiento de fruto amarillo infectado

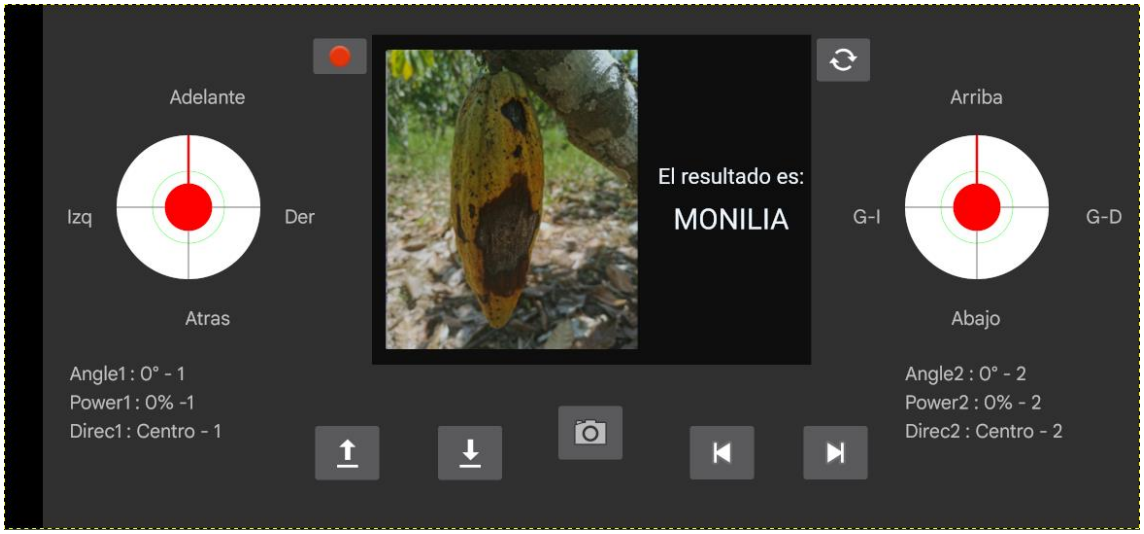


Figura 127 : Reconocimiento de fruto rojo infectado

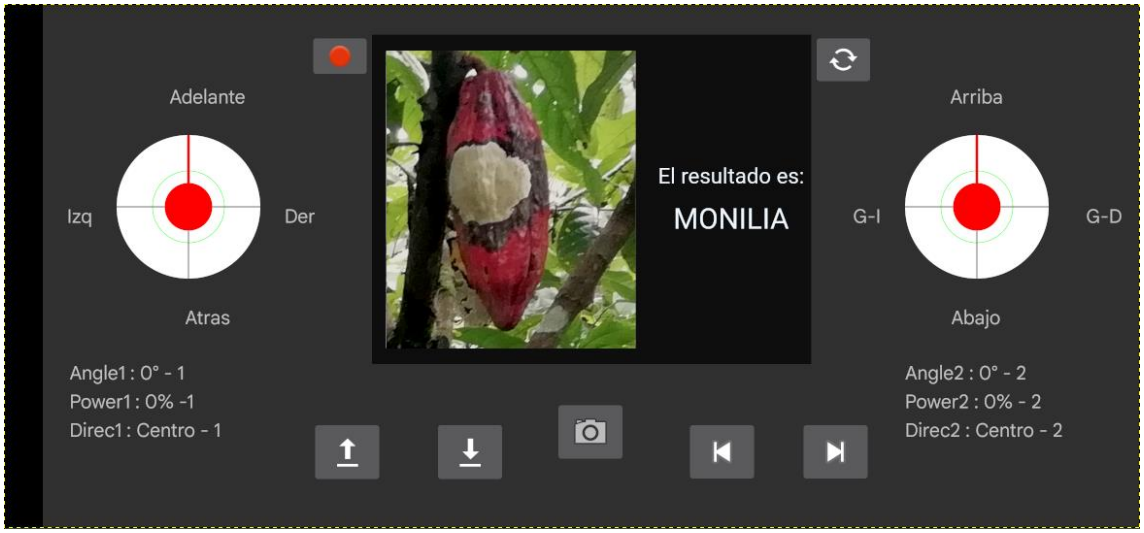
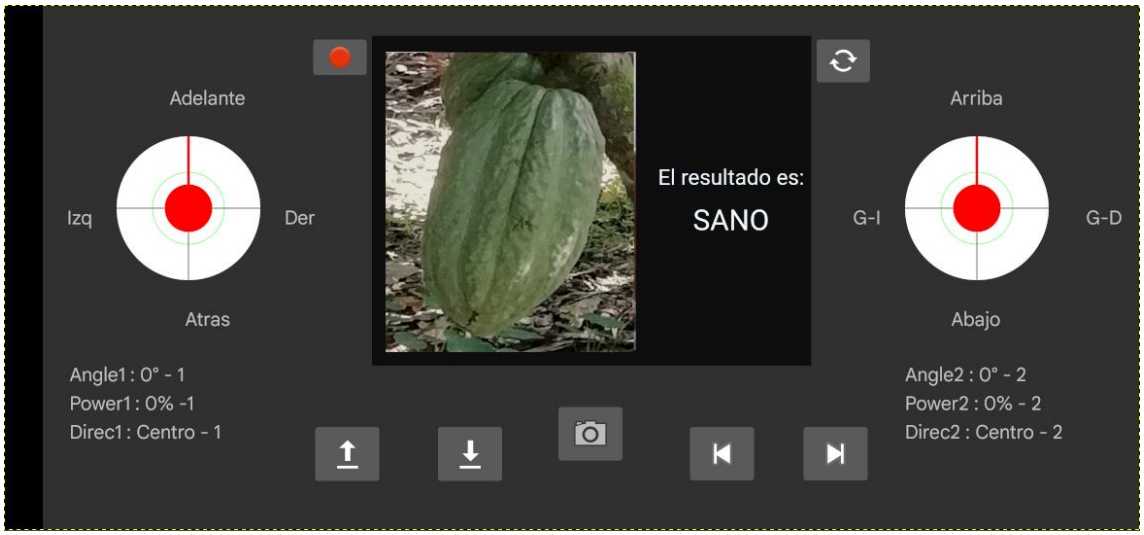


Figura 128 : Reconocimiento de fruto verde sano



Se logra llevar una mejor toma de decisiones para la adaptabilidad climática del cultivo de cacao mediante la recolección y visualización de 10 variables climáticas en una plataforma IoT.

El vehículo aéreo no tripulado logró reconocer con alta precisión frutos infectados, proporcionando datos sobre la cantidad de plantas infectadas.

Cabe mencionar que las pruebas de campo también pusieron en evidencia que esta investigación puede seguir evolucionando con el fin de obtener aún mejores resultados.

Conclusiones

- Se logró una mejora en el sistema productivo del cacao en base al diseño e implementación de una estación climática automática y un vehículo aéreo no tripulado, con la selección cuidadosa de componentes basada en la relación calidad-precio que ha permitido reducir los costos de producción de estas herramientas en un 86%, haciendo accesible a todos los productores de cacao empoderándolos con un control preciso sobre el entorno del cultivo.
- Se logró diseñar e implementar la arquitectura de hardware y software de la estación climática automática, en base a las pruebas de conexión entre los sensores y los microcontroladores, y también la conexión del microcontrolador con internet para el envío de datos a una base de datos en tiempo real, validando estos resultados comparándolos con datos oficiales del SENHAMI y obteniendo una correlación de los datos de más del 83%.
- Se logró diseñar una plataforma IoT para la visualización de datos climáticos en tiempo real, en base a la programación de la aplicación web progresiva y su conexión con una base de datos en tiempo real que recibe todos los datos leídos por cada sensor en la estación climática, contribuyendo así a la toma de decisiones informadas en la gestión de la adaptabilidad climática en el cultivo de cacao.
- Se logró diseñar e implementar la arquitectura de hardware y software del vehículo aéreo no tripulado en base a las pruebas del algoritmo PID para su estabilidad, alcanzando una autonomía de vuelo estable de 30 minutos y permitiendo un monitoreo más eficiente y preciso de vastas áreas de cultivo.
- Se logró diseñar y validar el algoritmo de inteligencia artificial para la detección de moniliasis en frutos de cacao, en base al entrenamiento del algoritmo con 4000 imágenes para el reconocimiento de colores, síntomas y texturas características del fruto de cacao obteniendo una precisión del 96%, validando este algoritmo en campo y obteniendo una precisión del 93.75%, esto elimina la necesidad de equipos especializados en reconocimiento de moniliasis y la exposición directa del agricultor a condiciones climáticas adversas.

- Se validó los algoritmos y la arquitectura de hardware y software de la estación climática y vehículo aéreo no tripulado, en base a pruebas de campo realizadas con distintas variedades de cacao, demostrando su eficacia en la mejora del sistema productivo del cultivo de cacao.

Recomendaciones

- Para el caso del anemómetro y la veleta en la estación climática estos al ser los más expuestos a los cambios constantes de clima, es preferible que sean de un material más resistentes que los materiales de impresión 3D, para que tengan mayor vida útil.
- Para el sistema electrónico de la estación climática se puede utilizar un microcontrolador que cuente con más pines y direcciones de memoria para poder leer todas las variables climáticas en un mismo controlador.
- Se debe contar con una conexión a internet estable y en caso de contar con una, se necesita crear alternativas que permitan esta conexión.
- Hacer uso de la mínima cantidad de cables para la construcción del vehículo aéreo no tripulado aprovechando las conexiones internas que vienen con los frames de desarrollo.
- Utilizar la misma cámara que se utilizará en el producto final para la toma de datos del algoritmo de reconocimiento, con el fin de no tener problemas de resolución o de iluminación.
- Buscar alternativas de cámaras compatibles con raspberry pi que cuenten con una mayor resolución y con un mayor campo de foco.
- Tener la opción de utilizar otro microcomputador en el dron con el fin de repartir los procesos de todo el software implementado.

Referencias bibliográficas

Abad Buri, J. A. & Farez Sigcha, J. P. (2018*). Diseño e implementación de un Sistema de monitoreo de variables climáticas que afectan al cultivo de café, en la plantación asoproccsi ubicado en santa Isabel.* [Tesis para optar por el Título de Ingeniero Electrónico]. Universidad Politécnica Salesiana. Cuenca, Ecuador.

Agroptima Blog (2020). Drones para fumigar. ¿Son viables para nuestros cultivos?. [https://www.agroptima.com/es/blog/drones-para-fumigacion-agricola-son-viables/]

Alcantara, C. (2013). *Ciclo biológico de Carmenta foraseminis Eichlin, en Theobroma cacao – en la zona de Satipo.* [Tesis para optar Título de Ingeniero en Ciencias Agrícolas]. Universidad Nacional del Centro del Perú. Satipo, Perú.

Alomia, J.; Alomia, C., & Vega, B. 2021. *Carmenta foraseminis Eichlin y Phytophthora palmivora en frutos de Theobroma cacao L. en Satipo, Perú.* Facultad Ciencias Agrarias, Universidad Nacional del Centro del Perú (UNCP), Huancayo, Perú.

Alvim, P. T. 1977. Cacao. In: Alvim, P.T; Koslowski, T.T. (eds.) *ECOPHYSIOLOGY OF TROPICAL CROPS*. New York: Academic Pres.

Anzules, V.; Borjas, R.; Alvarado, L.; Castro, V., & Julca, A. 2019. *Control cultural, biológico y químico de Moniliophthora roreri y Phytophthora spp en Theobroma cacao 'CCN-51'.* Scientia Agropecuaria, 10(4), 511-520, https://doi.org/10.17268/sci.agropecu.2019.04.08

Arshad Ali. (2020). *Labview and Internet of Things (IoT) Based Remote Monitoring of Lab Experiments to Enhance Collaboration between Universities*. International Journal of Innovative Technology and Exploring Engineering. https://doi.org/10.31449/inf.v46i2.3840

Aybar-Camacho, C.; Lavado-Casimiro, W.; Sabino, E.; Ramírez, S.; Huerta, J. & Felipe-Obando, O. (2017). *Atlas de zonas de vida del Perú – Guía Explicativa.* Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI). Dirección de Hidrología.

Bailey B. A. & Meinhardt L. W. (2016). *Cacao Diseases A History of Old Enemies and New Encounters.* Springer.

Batista, L; 2009. *Guía Técnica el Cultivo del Cacao en la República Dominicana. Santo Domingo, República Dominicana*. CEDAF.

Bejerano, P. (s. f.). (2021). *El uso de drones en agricultura.* ToDrone.

Bioversity International & Centro Internacional de Agricultura Tropical (CIAT) (2020). *Estrategia integral para el fortalecimiento del plan de competitividad de la cadena de cacao y chocolate en Ucayali 2020-2030.* Publicación CIAT No. 496.

Bohorquez Vergara, & Enciso Cala. (2019). *Diseño y construcción de un anemómetro para estudios de viabilidad de implementación de generadores eólicos en lugares remotos*. [Tesis para optar al título de Ingeniero Mecatrónico]. Universidad Autónoma de Bucaramanga. Bucaramanga, Colombia.

Campbell, C.L & Madden, L.V. 1990. *Introduction to plant disease Epidemiology*. John Wiley & Sons. New York.

Campetella C., Cerne B., Salió P. (2011). *Entornos invisibles de la ciencia y la tecnología. Estación Meteorológica.* Instituto Nacional de Educación Tecnológica. Buenos Aires, Argentina.

Caro Garrido W. (2019). *Diseño de un pluviómetro digital con tecnología sigfox de bajo costo, a través de un módulo Arduino*. [Tesis para optar por el título de Ingeniero Electrónico]. Universidad del Magdalena. Santa Marta, Colombia.

Ceccarelli V., Fremout T.,Zavaleta D., Lastra S., Imán S., Arévalo E., Rodriguez C., Cruz W. & Thomas E. (2021). *Climate change impacto in cultivated and wild cacao in Peru and the serch of climate change-tolerant genotypes.* CGIAR. https://doi.org/10.1111/ddi.13294

CENTE, F. 2019. *Preferencia y daño del Carmenta spp, relacionado al crecimiento del fruto de Theobroma cacao L. CCN-51 en dos épocas de producción, Pichari, Cusco, 2015.* [Tesis para optar Título profesional de Ingeniero Agroforestal]. Universidad Nacional de San Cristóbal de Huamanga. Ayacucho, Perú.

Cilas C. & Bastide P. (2020). *Challenges to cocoa production in the fase of climate change and the spread of pests and diseases.* MDPI Agronomy. https://doi.org/10.3390/agronomy10091232

D. S. Tan *et al*., "A framework for measuring infection level on cacao pods," *2016 IEEE Region 10 Symposium (TENSYMP)*, Bali, Indonesia, 2016, pp. 384-389, doi: 10.1109/TENCONSpring.2016.7519437.

De Azevedo, L.A. S. & Leite, O.M. 1995. *Manual de quantificacao de doencas de plantas.* Ciba Agro. Sao Paulo.

Dias, L.A.S. (Ed.). (2001). *Melhoramento genético do cacaueiro.* Viçosa. Funape.

DJI Drones Perú (2022). DJI Drone Phantom 4 Multiespectral. [http://djidroneperu.com/home/248-dji-drone-phantom-4-multiespectral.html]

Esther Salamí, Cristina Barrado, Enric Pastor (2014). *UAV flight experiments applied to the remote sensing of vegetated areas.* Univeritat Politecnica de Catalunya. Remote Sens. doi:10.3390/rs61111051.

Fontagro 2019. *La Cadena de Valor del Cacao en América Latina y El Caribe.*

Fundación Natura Colombia (2015). *Monitoreo Climático Participativo. Una estrategia basada en el enfoque de Ciencia Ciudadana.* Bogotá, Colombia.

Galan Zapata J. L. (2019). *Sistema inteligente de detección de patógenos y enfermedades en plantaciones de arroz en el departamento de Lambayeque durante 2019*. [Tesis para optar por el Título de Ingeniero de Sistemas y Computación]. Universidad Católica Santo Toribio de Mogrovejo. Chiclayo, Perú.

García, L.F. (2000). *Grupos y variedades de cacao. Cultivo del Cacao en la Amazonía Peruana. Arca, M. (ed.),* Instituto Nacional de Investigación Agraria (INIA), Lima.

Gomez J. & Lasluisa J. (2007). *Diseño e implementación de una estación meteorológica automática.* [Tesis para optar por el Título de Ingeniero Electromecánico]. Escuela Politécnica del Ejército. Latacunga, Ecuador.

Gomez, A. R., García B. R., Tong F. y Gonzales H. C. (2014). *Paquete Tecnológico del Cultivo del Cacao Fino de Aroma.* Oficina de las Naciones Unidas contra la Droga y el Delito para el Perú y el Ecuador – UNODC.

Hardy, F (1961). *Manual de cacao.* Turrialba, Costa Rica: Instituto Interamericano de Ciencias Agrícolas, 1961. 439 p.

HOBO by Onset (2020), Soluciones de Monitorización para Agua, Interior, Exterior y Agricultura. [Catálogo Distribuciones Industriales y Científicas S.L.].

Huilca Salcedo, J. G., & Sichiqui Velecela, P. F. (2019). *Diseño e implementación de un sistema embebido de monitoreo de las variables*

climáticas para plantaciones de maíz. [Tesis para obtar por el Título de Ingeniero Eléctrico]. Universidad Politécnica Salesiana. Cuenca, Ecuador.

John H. Bowers, Bryan A. Bailey, Prakash K. Hebbar, Soumaila Sanogo, and Robert D. Lumsden (2001). *The Impact of Plant Diseases on World Chocolate Production.* APS Publications. https://doi.org/10.1094/PHP-2001-0709-01-RV.

Latin America in movement (2015). Aeronaves militares extranjeras y aprobación previa de sobrevuelo.

Leandro Muñoz & Mariela E. (2017). *Biology and epidemiology of Moniliophthora roreri, causal agent of moniliophtora pod rot of cacao.* [Tesis de Doctorado]. CATIE, Turrialba, Costa Rica.

López García, J. (2020). PRON: *Diseño y construcción de un dron basado en Raspberry Pi*. Universitat Oberta de Catalunya. Barcelona, España.

MINAGRI Perú, cooperación alemana (2020). *Adaptación al cambio climático para la competitividad agraria: Experiencias exitosas en cultivos de algarroba, cacao y café.* Segundo Boletín. https://hdl.handle.net/20.500.12543/4585.

MINAGRI. 2019a. Sistema Integrado de Estadística Agraria. [Base de datos]. http://siea.minagri.gob.pe/siea/.

MINAGRI. 2019b. El Cacao en el Perú. [Presentación].

Ministerio de Agricultura y Desarrollo Rural – Colombia (2013). "*Guía Ambiental para el cultivo del cacao".* Corporación Colombiana de Investigación Agropecuaria.

Ministerio de Agricultura y Riego – Perú (2016). *"Estudio del CACAO en el Perú y el Mundo – un análisis de la producción y el comercio."* http://hdl.handle.net/20.500.13036/71.

Ministerio de Agricultura y Riego (MINAGRI) (2020). *Boletín de Publicación Trimestral Abril - Junio.*

Murrieta, E., y Palma, H. 2018. *Manejo integrado del mazorquero en el cultivo del cacao.* Del pueblo de los Estados Unidos de América (USAID), Alianza cacao Perú. [En línea] Issuu (https://issuu.com/comunicacionesalianzacacaoperu/docs/manual_mip_mazorquero documento en pdf, revisado el 27 dic 2021).

P. Patil, A. Narkhede, A. Chalke, H. Kalaskar and M. Rajput, "Real time automation of agricultural environment," *International Conference for*

Convergence for Technology-2014, Pune, India, 2014, pp. 1-4, doi: 10.1109/I2CT.2014.7092040.

Paiva-Peredo E. (2016). *Modelado y control de un cuadricóptero*. [Tesis para optar por el Título de Máster en Ingeniería Mecánico-Eléctrica con Mención en Automática y Optimización]. Universidad de Piura. Piura, Perú.

Pérez-Vicente, Luis F.. (2018). *Moniliophthora roreri H.C. Evans et al. and Moniliophthora perniciosa (Stahel) Aime: impact, symptoms, diagnosis, epidemiology and management*. Rev. Protección Veg. 33 (1). https://www.researchgate.net/publication/331012463_Moniliophthora_roreri_HC_Evans_et_al_and_Moniliophthora_perniciosa_Stahel_Aime_impact_symptoms_diagnosis_epidemiology_and_management

Phillips, W., y Cerda, R. 2011. *Catálogo de enfermedades de cacao en Centroamérica*. Centro Agronómico Tropical de Investigación y Enseñanza (CATIE) Turrialba, Costa Rica. 28 p.

Pino E. V. (2019). *Los drones una herramienta para una agricultura eficiente: un futuro de alta tecnología.* Universidad de Tarapacá. IDESIA. Arica, Chile.

Puertas Gayoso D. (2016). *DrondePi: Construcción de un dron basado en Raspberry Pi.* [Tesis para obtar por el Título Ingeniero de Tecnologías y Servicios de Telecomunicaciones]. Unversitat Politecnica de Valencia. Valencia, Perú.

Raja H, R. M., y Hardwick, K. (1987b). *The effect of different temperatures and water vapour pressure deficits on photosynthesis and transpiration of coca leaves.* pp. 211-214 En "Proceedings 10 International coca research conference".

Rios-Ruiz, R. 2004. *Epidemiologia e manejo da moniliase do cacaueiro no Peru.* Tese Doctor Scientiae. Minas Gerais, Brasil. Universidade Federal de Vicosa.

S. Galmarini, D. G. Steyn, B. Ainslie (2004). *The scaling law relating world point-precipitation records to duration.* Royal Meteorological Society. https://doi.org/10.1002/joc.1022

S.L. Yang, X.B. Yang, J.Y. (2017). *Optimization of control parameters of droplet density in citrus trees using UAVs and the Taguchi method.* International Journal of Agricultural and Biological Engineering (IJABE).

SANCHEZ, R. 2020. *Evaluar el comportamiento de Carmenta foraseminis (Busck) Eichlin en el cultivo de cacao (Theobroma cacao L.) en Sivia –Huanta*. [Tesis para optar Título de Ingeniero Agrónomo]. Universidad Nacional de Huancavelica. Huancavelica, Perú. 53 p.

Sena-Gomes, R. (1997). Informe anual de ESFIP- Estación Experimental Fillogônio Peixoto. Comité Ejecutivo del Plan de Cultivo de Cacao (CEPLAC). Linhares. Espírito Santo.

Sm, J. (2016*). Implementation of Precision Agriculture Using IOT and Raspberry PI Along with LabVIEW Simulation.* IEEE. India.

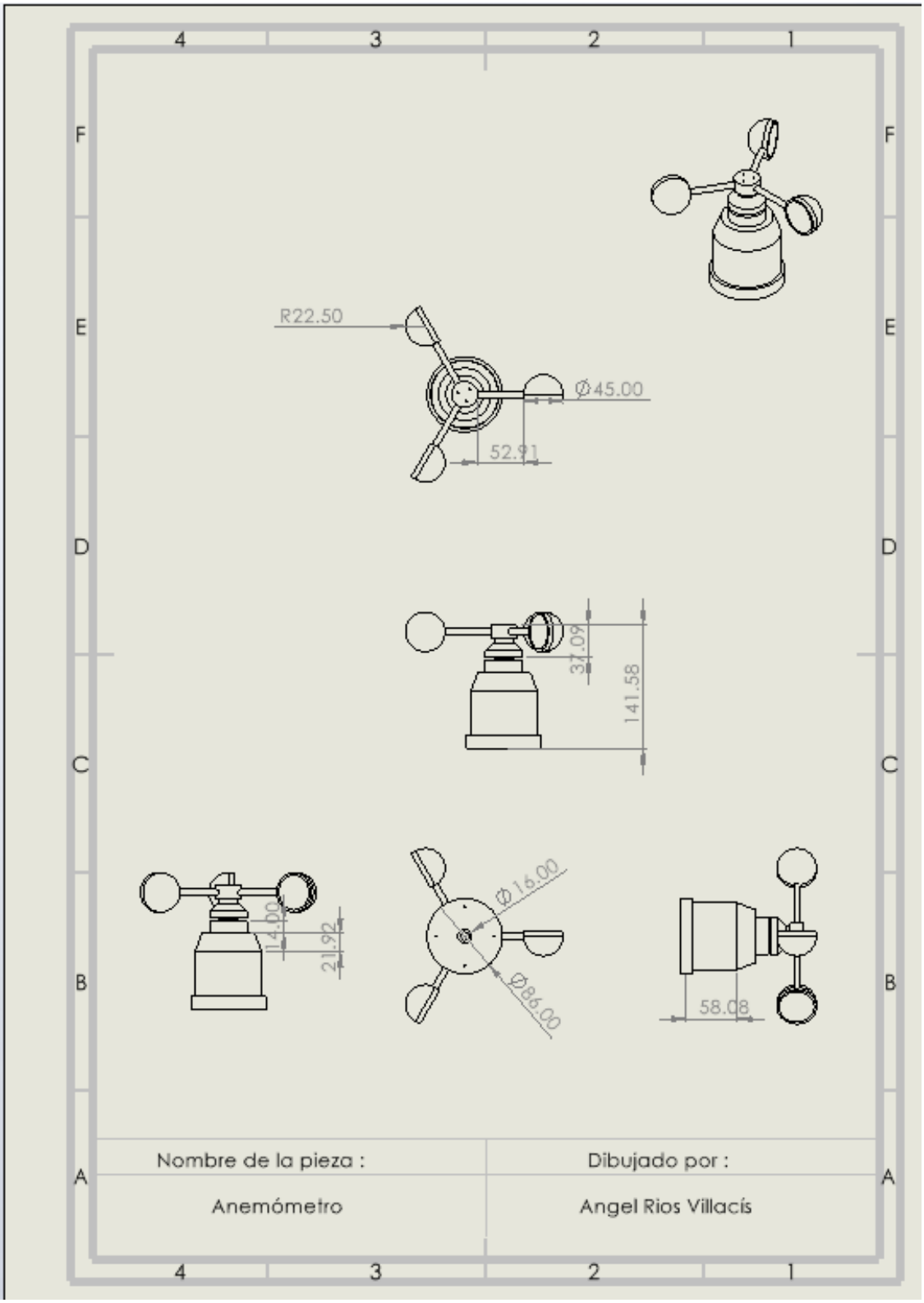
Stanley Tan, Neil Leong, Balon Laguna & Angelyn Lao (2018). *Automated Tool for Disease Detection and Assessment for Cacao Black Pod Rot*. Science Direct. https://doi.org/10.1016/j.cropro.2017.09.017.

Torres Arroyo, A. J. (2021). *Diseño de un vehículo aéreo cuadrirotor de bajo coste con plataforma raspberry pi*. [Tesis para optar por el Título de Ingeniero en Tecnologías Industriales]. Universidad Politécnica de Valencia. Valencia, España.

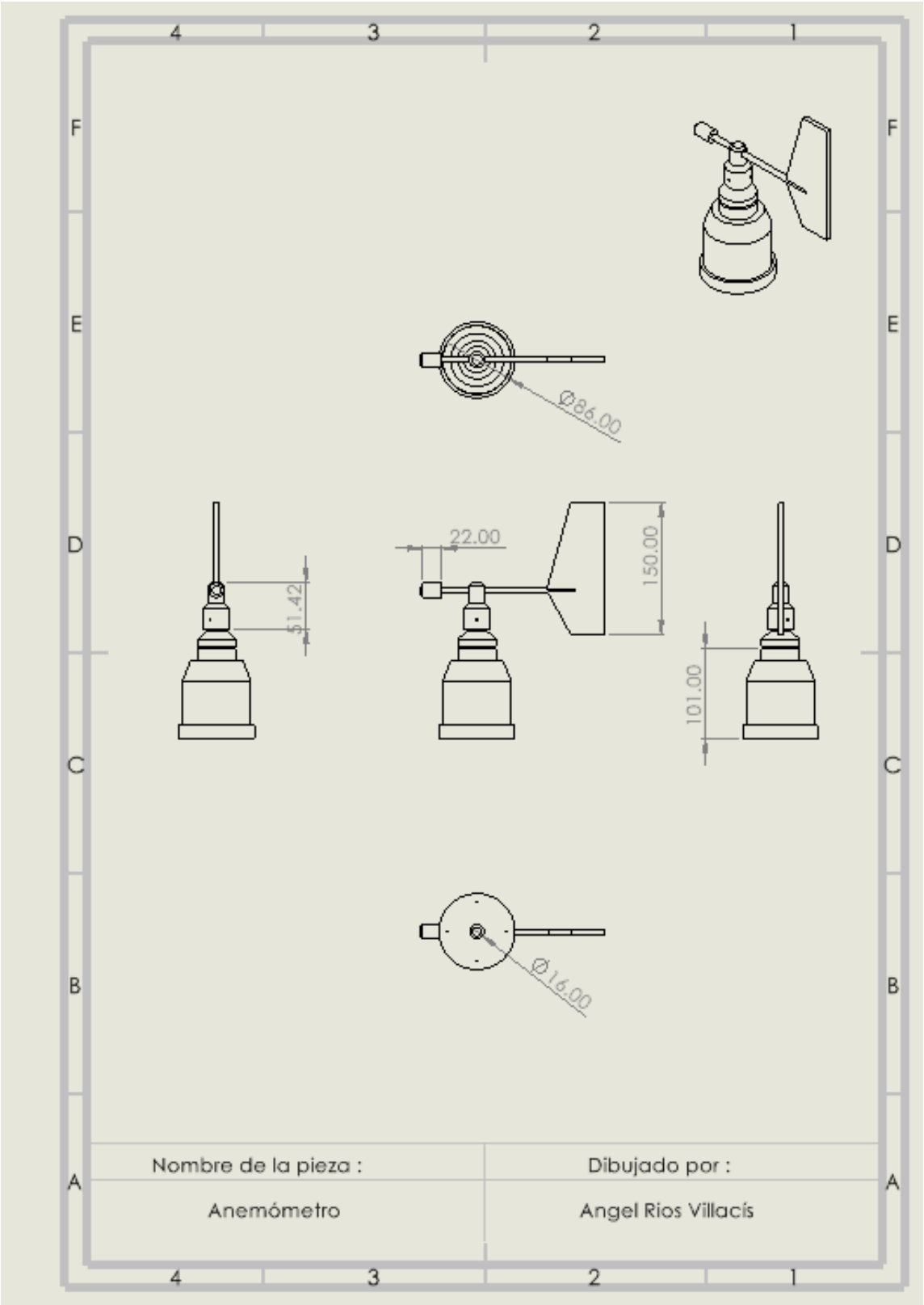
Zegarra E. (2020). *El gobierno de Sagasti y la agenda agraria en pandemia*. Portal de Noticias Ser.pe.

Anexos

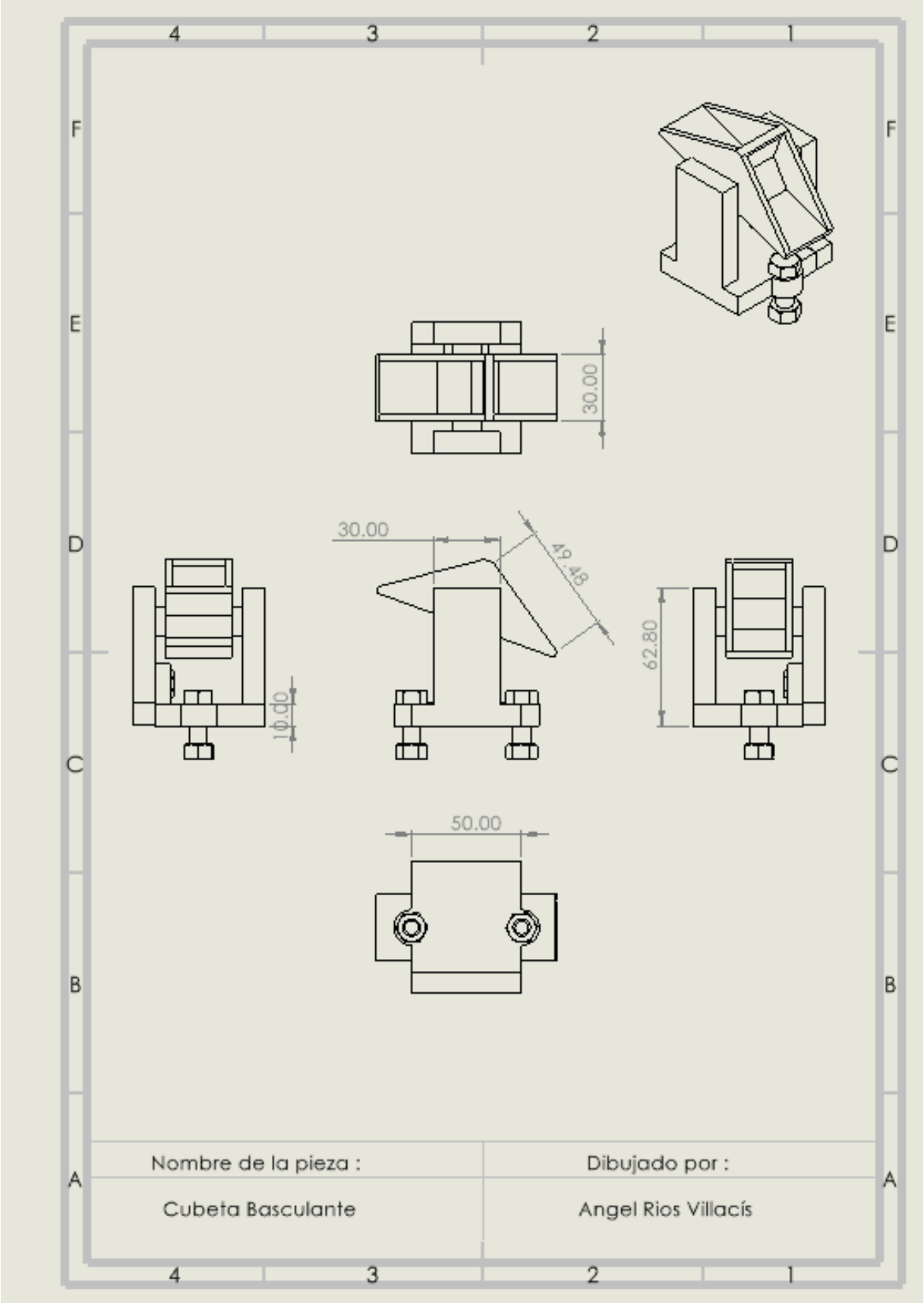
Anexo 1 : Plano de diseño para el Anemómetro

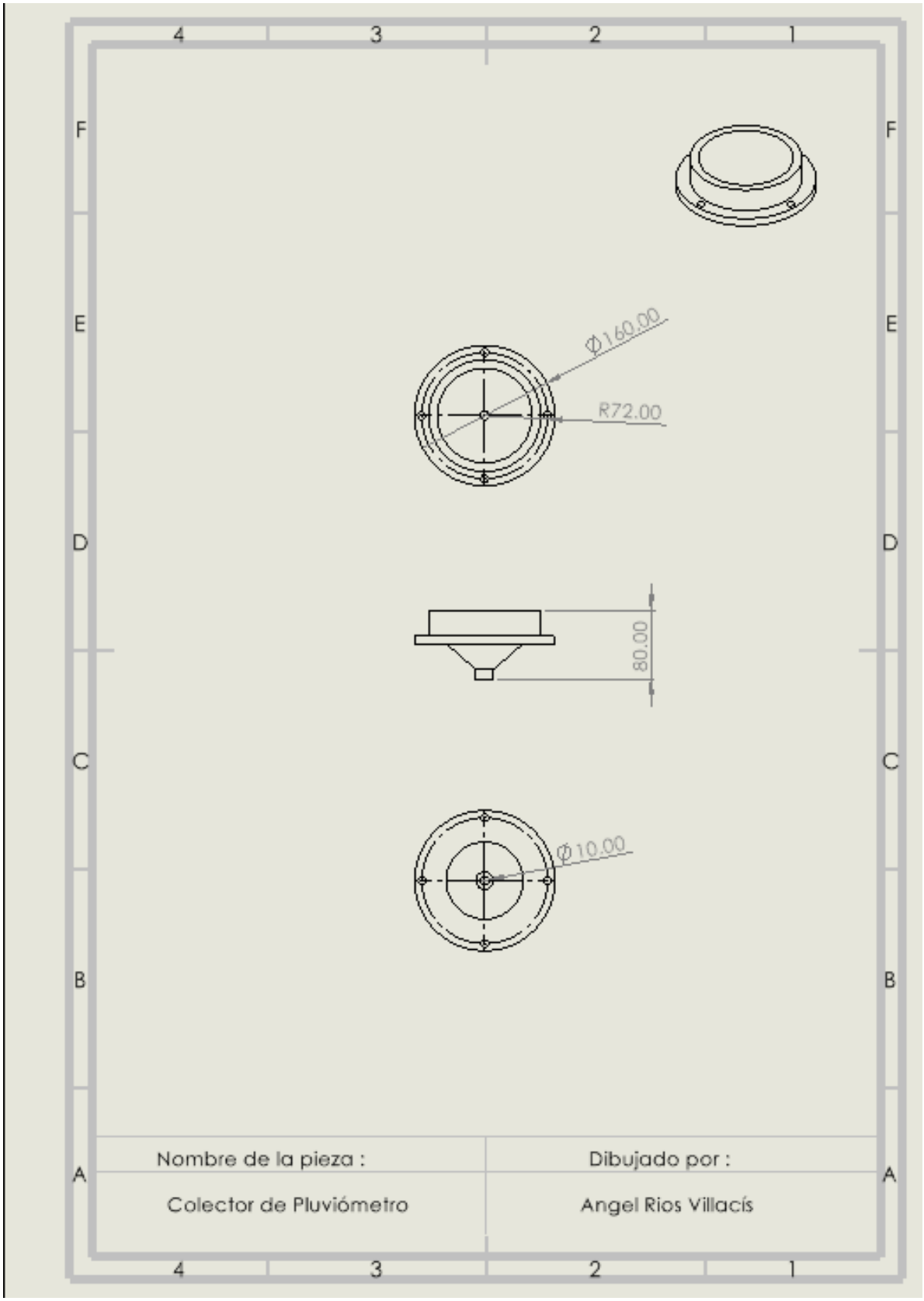


Anexo 2 : Plano de diseño para la Veleta

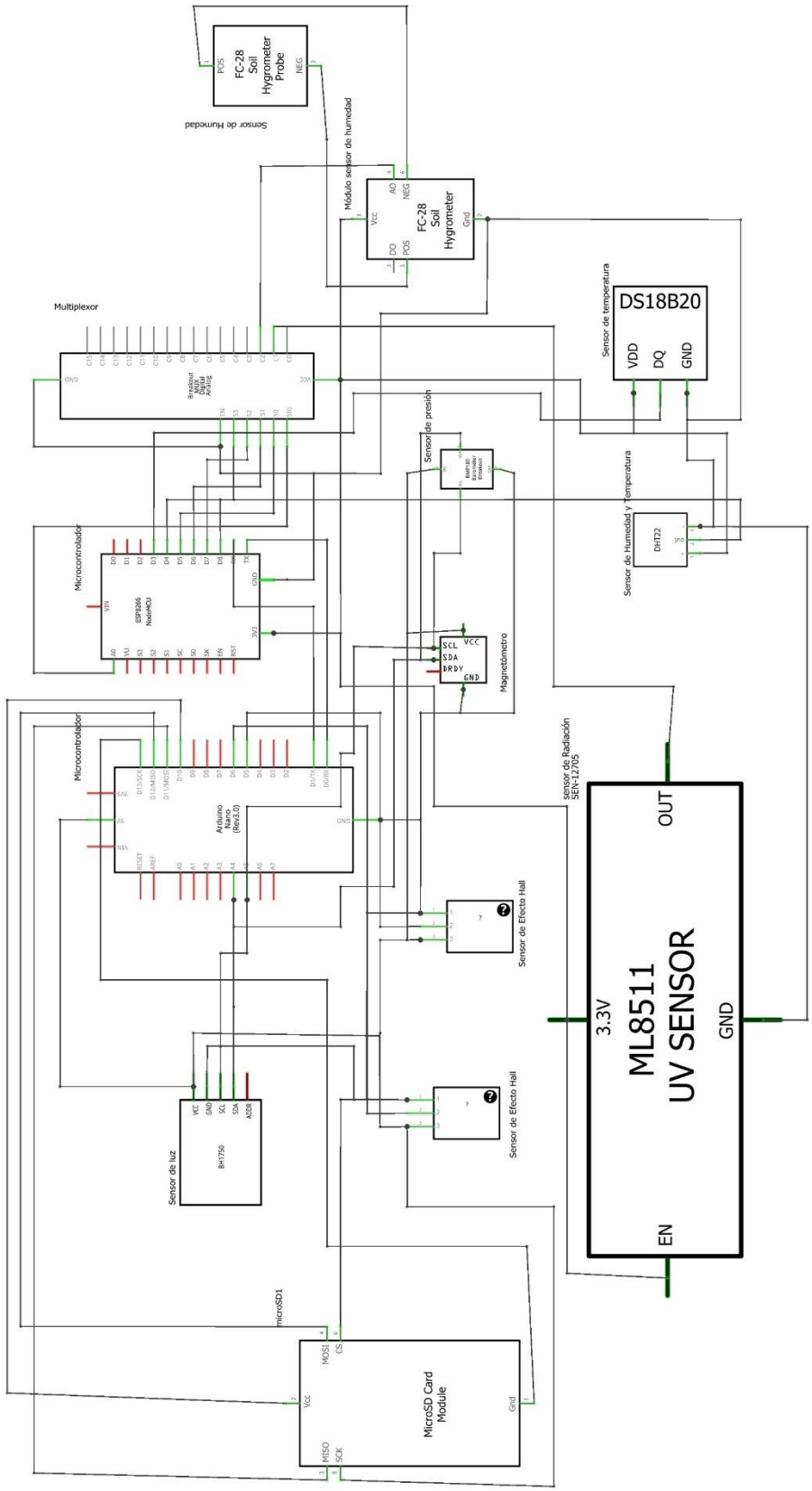


Anexo 3 : Plano de diseño para el Pluviómetro

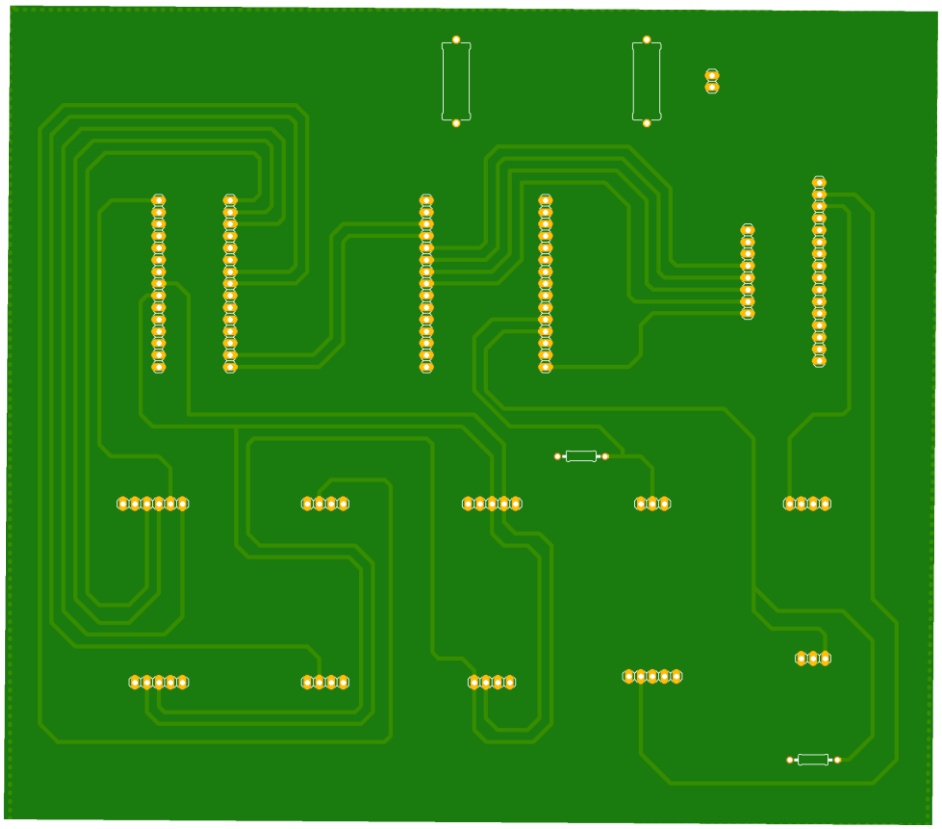


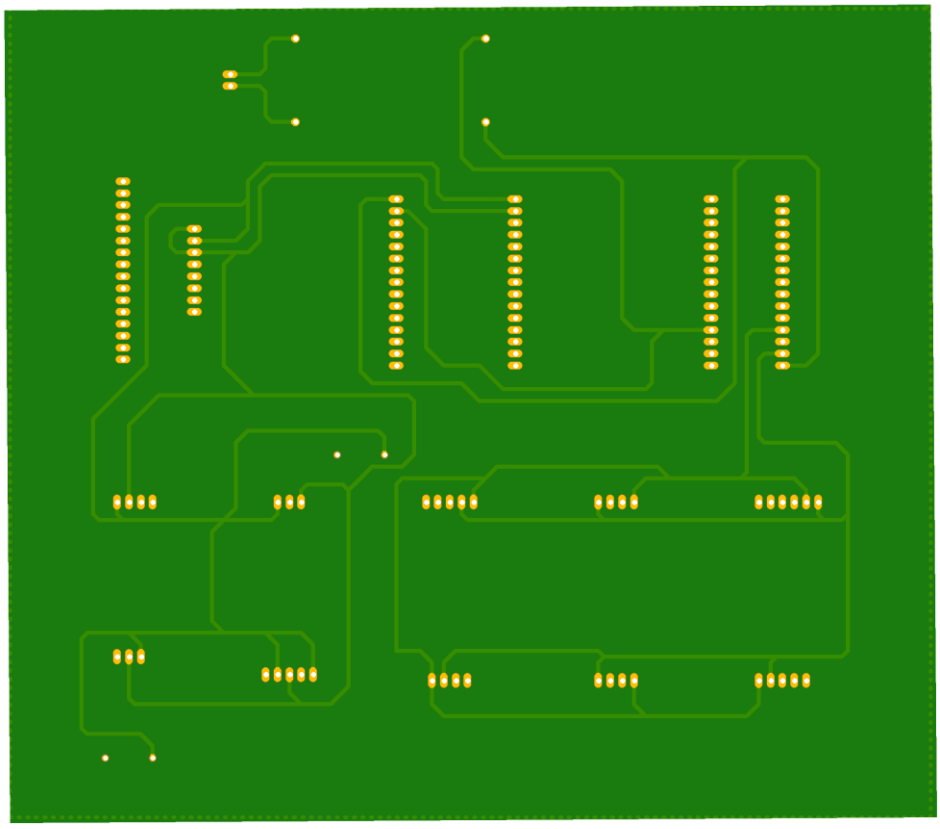

4
3
2
1
F
E
D
C
B
A
Ø160.00
R72.00
80.00
Ø10.00
Nombre de la pieza :
Colector de Pluviómetro
Dibujado por :
Angel Rios Villacís

Anexo 4 : Esquemático de la Estación Climática Automática

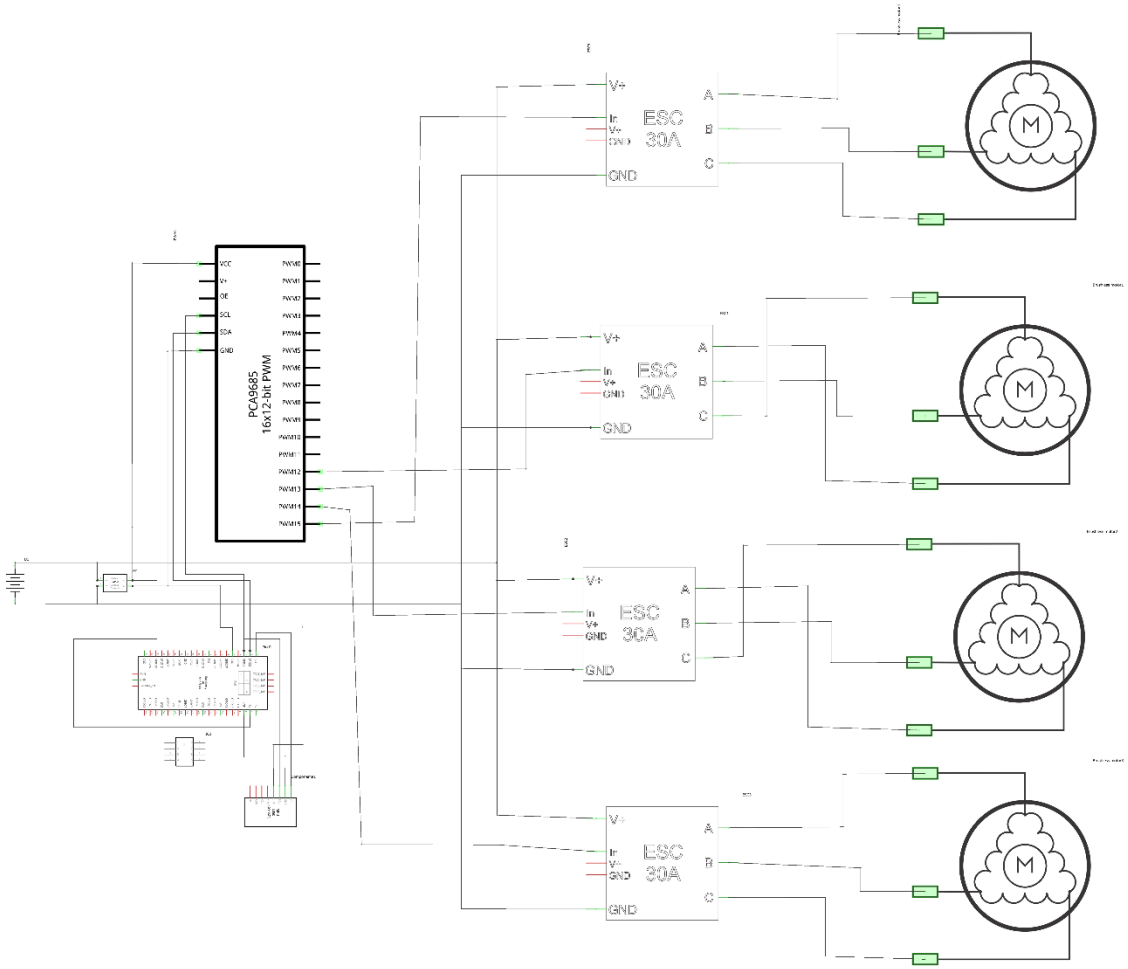


Anexo 5 : Planos de Impresión de la Estación climática automática





Anexo 6 : Esquemático Vehículo Aéreo no Tripulado



Anexo 7 : Código Arduino Nano

```
#include <QMC5883LCompass.h>
#include <FreqCount.h>
#include <BH1750.h>
#include <Wire.h>
#include <Adafruit_BMP085.h>

QMC5883LCompass compass;
BH1750 lightMeter;
Adafruit_BMP085 bmp;

unsigned long myTime = 0;
unsigned long myPreviousTime = 0;
unsigned long myPreviousPluvTime = 0;

float velocidadv = 0.0,lux = 0.0,presion=0.0,precipitacion=0.0;
long frecuencia = 0;
int precip = 0;
int angulo_or,pluv=1,pluv_anterior=1;

String cadena_primera = "";
char buff[] = "";

void setup() {

 Serial.begin(115200);

 inicializar_sensores();
}

void loop() {
 myTime = millis();

 leer_primero();

 if (myTime - myPreviousTime >= 5000) {
  myPreviousTime = myTime;
  leer_var();
 }

 if (myTime - myPreviousPluvTime >= 3600000) {
  myPreviousPluvTime = myTime;
  precip=0;
 }
}

void leer_primero () {
```

```
  pluv = digitalRead(6);

  if (FreqCount.available()){
    frecuencia = FreqCount.read();
    velocidadv = frecuencia*2*PI*0.017;
  }

  if (pluv == 0 && pluv_anterior == 1){
    precip++;
  }
  pluv_anterior = pluv;
}

void leer_var(){

  compass.read();
  angulo_or = compass.getAzimuth();
  lux = lightMeter.readLightLevel();
  presion = bmp.readPressure() / 100;
  precipitacion = precip * ((0.006) / (0.002916*PI)) ;

  cadena_primera =
String(angulo_or)+String(",")+String(precipitacion)+String(",")+String(velocidadv)+Strin
g(",")+String(lux)+String(",")+String(presion)+String(",")+String("\n");
  Serial.print(cadena_primera);

}

void inicializar_sensores () {

  Wire.begin();
  pinMode(6,INPUT);
  pinMode(5,INPUT);
  compass.init();
  lightMeter.begin();
  FreqCount.begin(1000);
  if (!bmp.begin()) {
    Serial.println("Could not find a valid BMP085 sensor, check wiring!");
    while (1) {}
  }

  Serial.println("primera linea serial fisico");
}
```

Anexo 8 : Código NodeMCU ESP8266

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <CD74HC4067.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <FirebaseESP8266.h>

OneWire ourWire(D3);
DallasTemperature sensors(&ourWire);

int uvLevel,refLevel,humedadss,angulo_or,precip;
float
temperatura,humedadr,humedads,luminosidad,temps,outputVoltage,uvIntensity,velo
cidadv,lux,presion;
DHT dht(D4,DHT22);
CD74HC4067 mux(D5,D6,D7,D8); //(s0,s1,s2,s3)

String cadena_enviar="";
char str1[]="";
String cadena_buff="";
String cadena_total="";

char caracter = '\n';

char *cadenas = NULL;
char *combinado[6];

#define _SSID "HUAWEI-D0BE"
#define _PASSWORD "xxxxxxxxxx"
#define FB_HOST "prueba-campo-xxxxx-default-rtdb.firebaseio.com"
#define FB_AUTH "XxxxxxXXXXxxxXXXXXxxxXXXXxxxXXXXx"

FirebaseData firebaseData;

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

  // Connect to WiFi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to: ");
  Serial.println(_SSID);
  WiFi.begin(_SSID, _PASSWORD);

  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print("-");
  }

  Serial.println("");
  Serial.println("WiFi Connected");

  Wire.begin();
  dht.begin();
  sensors.begin();

  Firebase.begin(FB_HOST,FB_AUTH);
  Firebase.reconnectWiFi(true);

  pinMode(A0,INPUT);
}

void loop() {
  leer_sensores_esp();

  leer_arduino_nano();
}

//Takes an average of readings on a given pin
//Returns the average
int averageAnalogRead(int pinToRead)
{
  byte numberOfReadings = 8;
  unsigned int runningValue = 0;

  for(int x = 0 ; x < numberOfReadings ; x++)
    runningValue += analogRead(pinToRead);
  runningValue /= numberOfReadings;

  return(runningValue);
}

float mapfloat(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

void leer_sensores_esp () {
  humedadr = dht.readHumidity();
  temperatura = dht.readTemperature();

  mux.channel(1);
```

```
  uvLevel = averageAnalogRead(A0);
 outputVoltage = uvLevel;

  uvIntensity = mapfloat(outputVoltage, 0.99, 2.8, 0.0, 15.0); //Convert the voltage to a
UV intensity level
  uvIntensity = 10*uvIntensity;

  mux.channel(2);
  humedadss = analogRead(A0);
  sensors.requestTemperatures();
  temps = sensors.getTempCByIndex(0);
  humedads = map(humedadss,1023,0,0,100);
}

void leer_arduino_nano () {
  if (Serial.available()) {
    Serial.readBytesUntil(caracter,str1,35);
    byte index = 0;
    cadenas = strtok(str1,",");
    while (cadenas != NULL) {
      combinado[index] = cadenas;
      index++;
      cadenas = strtok(NULL,",");
    }
    angulo_or = atoi(combinado[0]);
    precip = atoi(combinado[1]);
    velocidadv = atof(combinado[2]);
    lux = atof(combinado[3]);
    presion = atof(combinado[4]);

    if (Firebase.ready()){
      Firebase.setInt(firebaseData, "angulo_or", angulo_or);
      Firebase.setFloat(firebaseData, "humedadr", humedadr);
      Firebase.setFloatAsync(firebaseData, "humedads", humedads);
      Firebase.setFloat(firebaseData, "lux", lux);
      Firebase.setInt(firebaseData, "precip", precip);
      Firebase.setFloat(firebaseData, "presion", presion);
      Firebase.setFloat(firebaseData, "temperatura", temperatura);
      Firebase.setFloat(firebaseData, "temps", temps);
      Firebase.setFloat(firebaseData, "uvIntensity", uvIntensity);
      Firebase.setFloatAsync(firebaseData, "velocidadv", velocidadv);
    } else {
      Firebase.reconnectWiFi(true);
    }
  }
}
```

Anexo 9 : Script en Python para conexión Firebase – PostgreSQL

```
import psycopg2
import pyrebase
import datetime

# FIREBASE CONNECTION AND DATA
config = {
  "apiKey": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "authDomain": "prueba-campo-xxxxx.firebaseapp.com",
  "databaseURL": "https://prueba-campo-xxxxx-default-rtdb.firebaseio.com",
  "projectId": "prueba-campo-xxxxx",
  "storageBucket": "prueba-campo-xxxxx.appspot.com",
  "messagingSenderId": "xxxxxxxxxx",
  "appId": "1:xxxxxxxxxx:web:xxxxxxxxxxxxxxxxxxxxxx"
}

firebase = pyrebase.initialize_app(config)
authe = firebase.auth()
database = firebase.database()

print('connect to firebase complete')

angulo_or = round(database.child('angulo_or').get().val(), 2)
humedadr = round(database.child('humedadr').get().val(), 2)
humedads = round(database.child('humedads').get().val(), 2)
lux = round(database.child('lux').get().val(), 2)
precip = round(database.child('precip').get().val(), 2)
presion = round(database.child('presion').get().val(), 2)
temperatura = round(database.child('temperatura').get().val(), 2)
temps = round(database.child('temps').get().val(), 2)
uvIntensity = round(database.child('uvIntensity').get().val(), 2)
velocidadv = round(database.child('velocidadv').get().val(), 2)

print('Get data from firebase complete')


# Get the PostgreSQL database connection.
conn = psycopg2.connect(
  host="silly.db.elephantsql.com",
  port="5432",
  database="fovyuzik",
  user="fovyuzik",
  password="UCqDyKsOuZZWawMIOuq96RrmjQC6mvXe",
)

print('connect to psql complete')
```

```
# Put the data into the PostgreSQL database.
cur = conn.cursor()
horawf = datetime.datetime.now()
hora = horawf.strftime('%H:00')

query = '''
UPDATE hourdata
SET "angulo_or" = %s,
   "humedadr" = %s,
   "lux" = %s,
   "precip" = %s,
   "presion" = %s,
   "temperatura" = %s,
   "temps" = %s,
   "uvIntensity" = %s,
   "velocidadv" = %s,
   "humedads" = %s
WHERE hour = %s;
'''

cur.execute(
   query,
   (angulo_or, humedadr, lux, precip, presion, temperatura,
   temps, uvIntensity, velocidadv, humedads, hora),
)

conn.commit()

print('update psql complete')

if hora == '23:00':
   query1 = '''
      SELECT AVG("angulo_or"),
         AVG("humedadr"),
         AVG("lux"),
         AVG("precip"),
         AVG("presion"),
         AVG("temperatura"),
         AVG("temps"),
         AVG("uvIntensity"),
         AVG("velocidadv"),
         AVG("humedads")
      FROM hourdata;
   '''

   print('Get all averages from hourdata psql complete')
```

```
    cur.execute(query1)
    averages = cur.fetchone()
    avg = []

    for num in averages:
        avg.append(round(num, 2))

    today = datetime.date.today()
    avg.append(today)

    query2 = f'INSERT INTO datos_datosdiarios ("angulo_or", "humedadr", "lux",
"precip", "presion", "temperatura", "temps", "uvIntensity", "velocidadv", "humedads",
"fecha") VALUES ({avg[0]}, {avg[1]}, {avg[2]}, {avg[3]}, {avg[4]}, {avg[5]}, {avg[6]},
{avg[7]}, {avg[8]}, {avg[9]}, \'{avg[10]}\')'

    cur.execute(query2)
    conn.commit()

    print('Put the averages on datosdiarios psql complete')


conn.close()
```

Anexo 10 : Código Java de la aplicación móvil para el control del dron

```
package com.example.appstreamjoystick;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;

import com.zerokol.views.joystickView.JoystickView;

import org.w3c.dom.Text;

import java.io.DataOutputStream;
```

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;

public class MainActivity extends AppCompatActivity {

    WebView webView;
    JoystickView joystickView1,joystickView2;
    TextView angle1,power1,direc1,angle2,power2,direc2;
    ImageButton btn_der,btn_izq,btn_arr,btn_aba,btn_cam,btn_sync,btn_zero;

    Socket myAppSocket = null;
    public static String wifiModuleIp = "";
    public static int wifiModulePort = 0;
    public static String CMD = "";

    public static Boolean first_joy = true;
    public static Boolean second_joy = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

getWindow().getDecorView().setSystemUiVisibility(View.SYSTEM_UI_FLAG_HIDE_NAVI
GATION);

this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,Windo
wManager.LayoutParams.FLAG_FULLSCREEN);

        webView = (WebView) findViewById(R.id.webView);
        joystickView1 = (JoystickView) findViewById(R.id.joystickView1);
        joystickView2 = (JoystickView) findViewById(R.id.joystickView2);
        angle1 = (TextView) findViewById(R.id.angle1);
        angle2 = (TextView) findViewById(R.id.angle2);
        power1 = (TextView) findViewById(R.id.power1);
        power2 = (TextView) findViewById(R.id.power2);
        direc1 = (TextView) findViewById(R.id.direc1);
        direc2 = (TextView) findViewById(R.id.direc2);

        btn_aba = (ImageButton) findViewById(R.id.btn_aba);
        btn_arr = (ImageButton) findViewById(R.id.btn_arr);
        btn_izq = (ImageButton) findViewById(R.id.btn_izq);
        btn_der = (ImageButton) findViewById(R.id.btn_der);
        btn_cam = (ImageButton) findViewById(R.id.btn_cam);
        btn_sync = (ImageButton) findViewById(R.id.btn_sync);
```

```
btn_zero = (ImageButton) findViewById(R.id.btn_zero);

WebSettings webSettings = webView.getSettings();
webSettings.setJavaScriptEnabled(true);
webSettings.setUseWideViewPort(true);
webSettings.setLoadWithOverviewMode(true);

webView.loadUrl("http://192.168.4.1:8081/index.html");
getIpandPort();

btn_sync.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      webView.reload();
   }
});

btn_zero.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      CMD = "f";
      Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
      cmd_increase_servo.execute();
   }
});

btn_aba.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      CMD = "b";
      Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
      cmd_increase_servo.execute();
   }
});

btn_arr.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      CMD = "c";
      Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
      cmd_increase_servo.execute();
   }
});

btn_cam.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
```

```
            CMD = "8";
            Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
            cmd_increase_servo.execute();
            try {
               Thread.sleep(1500);
            } catch (InterruptedException e) {
               throw new RuntimeException(e);
            }
            //webView.reload();
            Intent intent = new Intent(MainActivity.this, MainActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            finish();
         }
      });

      btn_izq.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            CMD = "9";
            Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
            cmd_increase_servo.execute();
         }
      });

      btn_der.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            CMD = "a";
            Socket_AsyncTask cmd_increase_servo = new Socket_AsyncTask();
            cmd_increase_servo.execute();
         }
      });

      joystickView1.setOnJoystickMoveListener(new
JoystickView.OnJoystickMoveListener() {

         @Override
         public void onValueChanged(int angle, int power, int direction) {
            // TODO Auto-generated method stub
            angle1.setText(" " + String.valueOf(angle) + "° - 1");
            power1.setText(" " + String.valueOf(power) + "% - 1");
            first_joy = false;
            switch (direction) {
               case JoystickView.FRONT:
                  direc1.setText("Adelante-1");
```

```
            CMD = "0";
            if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
            }
            break;
        case JoystickView.FRONT_RIGHT:
            direc1.setText("Adelante-Izquierda-1");
            break;
        case JoystickView.RIGHT:
            direc1.setText("Izquierda-1");
            CMD = "3";
            if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
            }
            break;
        case JoystickView.RIGHT_BOTTOM:
            direc1.setText("Izquierda-Atras-1");
            break;
        case JoystickView.BOTTOM:
            direc1.setText("Atras-1");
            CMD = "2";
            if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
            }
            break;
        case JoystickView.BOTTOM_LEFT:
            direc1.setText("Atras-Derecha-1");
            break;
        case JoystickView.LEFT:
            direc1.setText("Derecha-1");
            CMD = "1";
            if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
            }
            break;
        case JoystickView.LEFT_FRONT:
            direc1.setText("Derecha-Adelante-1");
            break;
        default:
            first_joy = true;
            direc1.setText("Centro-1");
            sendZero();
    }
```

```
        }
    }, JoystickView.DEFAULT_LOOP_INTERVAL);

    joystickView2.setOnJoystickMoveListener(new
JoystickView.OnJoystickMoveListener() {

        @Override
        public void onValueChanged(int angle, int power, int direction) {
          // TODO Auto-generated method stub
          angle2.setText(" " + String.valueOf(angle) + "° - 2");
          power2.setText(" " + String.valueOf(power) + "% - 2");
          second_joy = false;
          switch (direction) {
            case JoystickView.FRONT:
              direc2.setText("Adelante-2");
              CMD = "4";
              if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
              }
              break;
            case JoystickView.FRONT_RIGHT:
              direc2.setText("Adelante-Izquierda-2");
              break;
            case JoystickView.RIGHT:
              direc2.setText("Izquierda-2");
              CMD = "7";
              if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
              }
              break;
            case JoystickView.RIGHT_BOTTOM:
              direc2.setText("Izquierda-Atras-2");
              break;
            case JoystickView.BOTTOM:
              direc2.setText("Atras-2");
              CMD = "6";
              if (power>=50) {
                Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                cmd_send.execute();
              }
              break;
            case JoystickView.BOTTOM_LEFT:
              direc2.setText("Atras-Derecha-2");
              break;
            case JoystickView.LEFT:
```

```
                direc2.setText("Derecha-2");
                CMD = "5";
                if (power>=50) {
                   Socket_AsyncTask cmd_send = new Socket_AsyncTask();
                  cmd_send.execute();
                }
                break;
            case JoystickView.LEFT_FRONT:
                direc2.setText("Derecha-Adelante-2");
                break;
            default:
                second_joy = true;
                direc2.setText("Centro-2");
                sendZero();
        }

      }
    }, JoystickView.DEFAULT_LOOP_INTERVAL);
  }

  public void getIpandPort(){
    wifiModuleIp = "192.168.4.1";
    wifiModulePort = 8080;
  }

  public void sendZero() {
    if (first_joy && second_joy){
      CMD = "d";
      Socket_AsyncTask cmd_send = new Socket_AsyncTask();
      cmd_send.execute();
    }
  }

  public class Socket_AsyncTask extends AsyncTask<Void,Void,Void> {
    Socket socket;

    @Override
    protected Void doInBackground(Void... params){
      try {
        InetAddress inetAddress =
InetAddress.getByName(MainActivity.wifiModuleIp);
        socket = new java.net.Socket(inetAddress,MainActivity.wifiModulePort);
        DataOutputStream dataOutputStream = new
DataOutputStream(socket.getOutputStream());
        dataOutputStream.writeBytes(CMD);
        dataOutputStream.close();
        socket.close();
```

```
            }catch (UnknownHostException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return null;
        }
    }
}
```

Anexo 11 : Servidor de visualización de cámara

```
import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html style="height: 100%;">
  <head>
    <meta name"viewport" content="width=device-width, minimum-scale=0.1">
    <title>Raspberry Pi - Surveillance Camera</title>
  </head>
  <body style="margin: 0px; background: #0e0e0e; height: 100%">
    <img style="display: block; transform: rotate(180deg); -webkit-user-select: none;
margin: auto; background-color: hsl(0,0%,25%); width: 100%; height: 100%"
src="stream.mjpg">
  </body>
</html>
"""

class StreamingOutput(object):
  def __init__(self):
    self.frame = None
    self.buffer = io.BytesIO()
    self.condition = Condition()

  def write(self, buf):
    if buf.startswith(b'\xff\xd8'):
      self.buffer.truncate()
      with self.condition:
        self.frame = self.buffer.getvalue()
        self.condition.notify_all()
      self.buffer.seek(0)
    return self.buffer.write(buf)
```

```
class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace;
boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()
```

```
  camera.start_recording(output, format='mjpeg')
  try:
    address = ('192.168.4.1',8081)
    server = StreamingServer(address, StreamingHandler)
    server.serve_forever()
  finally:
    camera.stop_recording()
```

Anexo 12 : Servidor de visualización del resultado del algoritmo de reconocimiento

```
from http import server
import sys

param = sys.argv[1] if len(sys.argv) > 1 else 'NaN'

PAGE="""\
<html style="height: 100%;">
  <head>
    <meta name"viewport" content="width=device-width, minimum-scale=0.1">
    <title>Raspberry Pi - Surveillance Camera</title>
  </head>
  <body style="margin: 0px; background: #0e0e0e; color: aliceblue; display: grid; grid-
template-columns:1.7fr 1fr; place-items: center; height:100%">
      <div style="height: 90%;width:90%; background-color: blue;">
        <img style="display: block; -webkit-user-select: none; margin: auto;
background-color: hsl(0,0%,25%); width: 100%; height: 100%" src="/image.jpg">
      </div>
      <div style="display: flex; align-items: center; justify-content: center; flex-
direction: column; width: 100%;">
        <div style="display: flex; justify-content: center; align-items: center; font-size:
3em;">El resultado es:</div>
        <div style="display: flex; margin-top: 8%; justify-content: center; align-items:
center; font-size: 4em;">{}</div>
      </div>
  </body>
</html>
""".format(param)

class ImageHandler(server.BaseHTTPRequestHandler):
  def do_GET(self):
    if self.path == '/index.html':
      content = PAGE.encode('utf-8')
      self.send_response(200)
      self.send_header('Content-type', 'text/html')
      self.send_header('Content-Length', len(content))
      self.end_headers()
```

```
            self.wfile.write(content)
        elif self.path == '/image.jpg':
            try:
                # Ruta local de la imagen
                image_path = "/home/greenskull/Desktop/raspberry/test_images/image.jpg"
                # Abrir y leer la imagen en modo binario
                with open(image_path, 'rb') as file:
                    image_data = file.read()
                # Enviar la respuesta con la imagen
                self.send_response(200)
                self.send_header('Content-type', 'image/jpeg')
                self.send_header('Content-Length', len(image_data))
                self.end_headers()
                self.wfile.write(image_data)
            except FileNotFoundError:
                self.send_error(404, "Imagen no encontrada")
        else:
            # return super().do_GET()
            self.send_error(404)
            self.end_headers()

if __name__ == '__main__':
    try:
        # Crea el servidor HTTP en el puerto 8081
        server = server.HTTPServer(('192.168.4.1', 8081), ImageHandler)
        print('Servidor en ejecución en http://192.168.4.1:8081/')
        server.serve_forever()
    except KeyboardInterrupt:
        print('Servidor detenido')
        server.server_close()
```

Anexo 13 : Script de Control de Motores

```
import time
from adafruit_servokit import ServoKit

global init
init = [2490, 2490, 2490, 2480]
global first
first = [3490, 3490, 3490, 3480]
global vel
vel = [2490, 2490, 2490, 2480]
global throt
throt = [2990, 2990, 2990, 2980]
global serv
```

```
serv = [7900, 4100]

import board
import busio
import adafruit_pca9685

i2c = busio.I2C(board.SCL, board.SDA)
pca = adafruit_pca9685.PCA9685(i2c)
pca.frequency = 50

def setup():
    global pca1
    pca1 = ServoKit(channels=16)
    global motores
    motores = [12, 13, 14, 15]
    global servos
    servos = [10, 11]
    pca.channels[servos[0]].duty_cycle = serv[0]
    pca.channels[servos[1]].duty_cycle = serv[1]
    for i in motores:
        pca.channels[i].duty_cycle = 0
        time.sleep(1.5)
        pca.channels[i].duty_cycle = 2500
        time.sleep(1.5)
        pca.channels[i].duty_cycle = 0
        time.sleep(1.5)
        pca.channels[i].duty_cycle = 2500
    setmot()

def arriba():
    for i in range(4):
        elevar(i)
    vel = throt
    setmot()

def abajo():
    for i in range(4):
        disminuir(i)
    vel = throt
    setmot()

def adelante():
    for i in range(5):
        elevar(2)
        elevar(3)
        disminuir(1)
        disminuir(0)
```

```
    vel = throt
    setmot()

def atras():
    for i in range(5):
        elevar(1)
        elevar(0)
        disminuir(2)
        disminuir(3)
    vel = throt
    setmot()

def derecha():
    for i in range(10):
        elevar(1)
        elevar(2)
        disminuir(0)
        disminuir(3)
    vel = throt
    setmot()

def izquierda():
    for i in range(10):
        elevar(0)
        elevar(3)
        disminuir(1)
        disminuir(2)
    vel = throt
    setmot()

def giro_der():
    for i in range(10):
        elevar(0)
        elevar(2)
        disminuir(1)
        disminuir(3)
    vel = throt
    setmot()

def giro_izq():
    for i in range(10):
        elevar(1)
        elevar(3)
        disminuir(0)
        disminuir(2)
    vel = throt
    setmot()
```

```
def setvel(ml):
    for i in range(4):
        vel[i] = ml[i]

def setthrot(ml):
    for i in range(4):
        throt[i] = ml[i]

def cam_der():
    serv[1] = serv[1] + 200
    if serv[1] > 5800:
        serv[1] = 5800
    pca.channels[servos[1]].duty_cycle = serv[1]

def cam_izq():
    serv[1] = serv[1] - 200
    if serv[1] < 1700:
        serv[1] = 1700
    pca.channels[servos[1]].duty_cycle = serv[1]

def cam_arr():
    serv[0] = serv[0] + 200
    if serv[0] > 8000:
        serv[0] = 8000
    pca.channels[servos[0]].duty_cycle = serv[0]

def cam_aba():
    serv[0] = serv[0] - 200
    if serv[0] < 4500:
        serv[0] = 4500
    pca.channels[servos[0]].duty_cycle = serv[0]

def apagar():
    for i in range(4):
        throt[i] = init[i]
        vel[i] = init[i]
    setmot()

def elevar(index):
    throt[index] = throt[index] + 10
    if throt[index] > 7400:
        throt[index] = 7400
    if throt[index] < first[index]:
        throt[index] = first[index]

def disminuir(index):
```

```
    throt[index] = throt[index] - 10
    if throt[index] < first[index]:
        throt[index] = init[index]

def setmot():
    for i in range(4):
        pca.channels[motores[i]].duty_cycle = vel[i]
```

Anexo 14 : Script del módulo MPU9250

```
import time
import smbus

from imusensor.MPU9250 import MPU9250
from imusensor.filters import kalman

def setup():
    global bus
    global imu
    address = 0x68
    bus = smbus.SMBus(1)
    imu = MPU9250.MPU9250(bus,address)
    imu.begin()
    global sensorfusion
    sensorfusion = kalman.Kalman()
    global startroll
    global startpitch
    global startyaw
    imu.readSensor()
    imu.computeOrientation()
    startroll = imu.roll
    startpitch = imu.pitch
    startyaw = imu.yaw
    imu.readSensor()
    imu.computeOrientation()
    sensorfusion.roll = imu.roll
    sensorfusion.pitch = imu.pitch
    sensorfusion.yaw = imu.yaw

def get_accel():
    imu.readSensor()
    imu.computeOrientation()
    return imu.AccelVals

def get_gyro():
    imu.readSensor()
    imu.computeOrientation()
    return imu.GyroVals
```

```
def get_mag():
    imu.readSensor()
    imu.computeOrientation()
    return imu.MagVals

def get_orien():
    imu.readSensor()
    imu.computeOrientation()
    r = imu.roll
    p = imu.pitch
    y = imu.yaw

    diffr = r - startroll
    diffp = p - startpitch
    diffy = y - startyaw
    if diffr < -90:
        diffr = diffr + 360
    if diffp < -90:
        diffp = diffp + 360
    if diffy < -90:
        diffy = diffy + 360
    if diffr > 90:
        diffr = 360 - diffr
    if diffp > 90:
        diffp = 360 - diffp
    if diffy > 90:
        diffy = 360 - diffy

    return [round(diffr),round(diffp),round(diffy)]

def get_orien_kal():
    imu.readSensor()
    imu.computeOrientation()
    dt = 0.1
    sensorfusion.computeAndUpdateRollPitchYaw(imu.AccelVals[0], imu.AccelVals[1],
imu.AccelVals[2], imu.GyroVals[0], imu.GyroVals[1], imu.GyroVals[2], imu.MagVals[0],
imu.MagVals[1], imu.MagVals[2], dt)
    r = sensorfusion.roll
    p = sensorfusion.pitch
    y = sensorfusion.yaw

    diffr = r - startroll
    diffp = p - startpitch
    diffy = y - startyaw
    if diffr < -180:
        diffr = diffr + 360
```

```
    if diffp < -180:
        diffp = diffp + 360
    if diffy < -180:
        diffy = diffy + 360
    if diffr > 180:
        diffr = 360 - diffr
    if diffp > 180:
        diffp = 180 - diffp
    if diffy > 180:
        diffy = 360 - diffy

    return [round(diffr),round(diffp),round(diffy)]
```

Anexo 15 : Script del control PID

```
global preverror
preverror = [0,0,0]
global sumerror
sumerror = [0,0,0]
global kp
kp=[2.5,2.5,0]
global ki
ki=[0,0,0]
global kd
kd=[6.5,6.5,0]
global dt
dt = 0.1

def pidfunc(tar,cur,i):
    error = cur - tar
    # if (error<5 and error>-5):
    #     return 0
    P = kp[i]*error
    D = kd[i]*(error-preverror[i])/dt
    I = ki[i]*sumerror[i]
    out = P+D+I
    # if out > 600:
    #     out = 600
    # if out < -600:
    #     out = -600
    preverror[i] = error
    sumerror[i] = sumerror[i]+error*dt
    return round(out)
```

Anexo 16 : Script del módulo MPU6050

```
from mpu6050 import mpu6050
import math

def setup():
    global sensor, roll, pitch, yaw, alpha, r0, p0
    sensor = mpu6050(0x68)
    roll = 0.79
    pitch = -0.59
    yaw = 0.0
    alpha = 0.98
    r0 = -1.85
    p0 = 2.05

def red_sensor_data():
    accel_data = sensor.get_accel_data()
    gyro_data = sensor.get_gyro_data()
    return accel_data, gyro_data

def get_angles(accel_data, gyro_data, dt):
    global roll, pitch, yaw, r0, p0

    roll_acc = math.degrees(math.atan2(accel_data['y'], accel_data['z'])) - roll
    roll = alpha * (roll + gyro_data['x'] * dt) + (1 - alpha) * roll_acc

    pitch_acc = math.degrees(math.atan2(-accel_data['x'], math.sqrt(accel_data['y'] ** 2
+ accel_data['z'] ** 2))) - pitch
    pitch = alpha * (pitch + gyro_data['y'] * dt) + (1 - alpha) * pitch_acc

    yaw += gyro_data['z'] * dt


    # Asegurarse de que los ángulos estén en el rango -180 a 180 grados
    roll = roll % 360
    if roll > 180:
        roll -= 360
    pitch = pitch % 360
    if pitch > 180:
        pitch -= 360

    return roll, pitch, yaw


def get_orien(dt):
    accel_data, gyro_data = red_sensor_data()
    r, p, y = get_angles(accel_data, gyro_data, dt)
    return [r, p, y]
```

Anexo 17 : Script principal

```
from ast import Try
from operator import truediv
from socket import *
from time import ctime, sleep
from subprocess import Popen, PIPE
import func_motor
import pid
import threading
import time
import func_mpu
change = False
process = True
global stable
stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
global pid_vel
pid_vel =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
global tar_ang
tar_ang = [0,0,0]

func_motor.setup()
func_mpu.setup()

import tensorflow as tf
from tensorflow import keras
model =
tf.keras.models.load_model('/home/greenskull/Desktop/raspberry/p37_models/p37_
m_25_adam.h5', custom_objects={'GlorotUniform':
tf.keras.initializers.GlorotUniform})
image_size = (180,180)

cam_process = Popen(['python3',
'/home/greenskull/Desktop/raspberry/prueba_completa/cam_server.py'])

global prevtime
prevtime = time.time()
global pidprint
pidprint = [0,0,0]
global rpy
rpy = [0,0,0]

class ThreadJob(threading.Thread):
    def __init__(self,callback,event,interval):
        self.callback = callback
        self.event = event
```

```
        self.interval = interval
        super(ThreadJob,self).__init__()

    def run(self):
        while not self.event.wait(self.interval):
            self.callback()

def pidfunc():
    global pidprint
    global rpy
    rpy = func_mpu.get_orien(0.1)
    r = pid.pidfunc(tar_ang[0],rpy[0],0)
    p = pid.pidfunc(tar_ang[1],rpy[1],1)
    y = pid.pidfunc(tar_ang[2],rpy[2],2)
    pidprint = [r,p,y]
    m1 = func_motor.throt[0] - r + p + y
    m2 = func_motor.throt[1] - r - p - y
    m3 = func_motor.throt[2] + r - p + y
    m4 = func_motor.throt[3] + r + p - y
    pid_vel = [m1,m2,m3,m4]
    func_motor.setvel(pid_vel)
    func_motor.setmot()

def threadfunc():
    if func_motor.throt[3] >= 3800:
        pidfunc()

def onlyprint():
    global prevtime
    global pidprint
    global rpy
    nowtime = time.time()
    dt = nowtime-prevtime
    if dt>=1:
        print('THROT : ',func_motor.throt)
        print('VELOC : ',func_motor.vel)
        print('  PID : ',pidprint)
        print('  MPU : ',[round(rpy[0]),round(rpy[1]),round(rpy[2])])
        print('********************************')
        prevtime = time.time()

event = threading.Event()
k = ThreadJob(threadfunc,event,0.1)
k.start()

event2 = threading.Event()
k2 = ThreadJob(onlyprint,event2,0.1)
```

```
k2.start()

CMD = ['0','1','2','3','4','5','6','7','8','9','a','b','c','d','e','f']
HOST = '192.168.4.1'
PORT = 8080
BUFSIZE = 1024
ADDR = (HOST,PORT)
tcpSerSock = socket(AF_INET,SOCK_STREAM)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(2)

print('Waiting for connection')

while True:
    tcpCliSock,addr = tcpSerSock.accept()
    try:
        while True:
            data=tcpCliSock.recv(BUFSIZE)
            data1 = data.decode('UTF-8')
            if not data:
                break;
            if data1 == CMD[0]:
                if change == True:
                    stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
                    change = False
            if data1 == CMD[1]:
                if change == True:
                    stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
                    change = False
            if data1 == CMD[2]:
                if change:
                    stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
# func_motor.atras()
                    change = False
            if data1 == CMD[3]:
                if change == True:
                    tar_ang = [-5,0,0]
                    stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
                    change = False
            if data1 == CMD[4]:
                func_motor.arriba()
                stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
```

```
        if data1 == CMD[5]:
            if change == True:
                stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
# func_motor.giro_der()
                change = False
        if data1 == CMD[6]:
            func_motor.abajo()
            stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
        if data1 == CMD[7]:
            if change == True:
                stable =
[func_motor.throt[0],func_motor.throt[1],func_motor.throt[2],func_motor.throt[3]]
                func_motor.giro_izq()
                change = False
        if data1 == CMD[8]:
            params = 'MONILIA'
            if process:
                cam_process.terminate()
                sleep(1)
                cam_shot = Popen(['python3',
'/home/greenskull/Desktop/raspberry/prueba_completa/cam_shot.py'])
                imagen =
keras.preprocessing.image.load_img('/home/greenskull/Desktop/raspberry/test_imag
es/image.jpg',target_size=image_size)
                imagen_array = keras.preprocessing.image.img_to_array(imagen)
                imagen_array = tf.expand_dims(imagen_array,0)
                pred = model.predict(imagen_array)
                score = float(pred[0])
                if score > 0.5:
                    params = 'SANO'
                server_process = Popen(['python3',
'/home/greenskull/Desktop/raspberry/prueba_completa/server_test.py', params])
                process = False
            else :
                server_process.terminate()
                sleep(1)
                cam_process = Popen(['python3',
'/home/greenskull/Desktop/raspberry/prueba_completa/cam_server.py'])
                process = True
        if data1 == CMD[9]:
            func_motor.cam_izq()
        if data1 == CMD[10]: # a
            func_motor.cam_der()
        if data1 == CMD[11]: # b
            func_motor.cam_aba()
```

```
        if data1 == CMD[12]: # c
            func_motor.cam_arr()
        if data1 == CMD[13]: # d
            change = True
            tar_ang = [0,0,0]
            func_motor.setthrot(stable)
            func_motor.setvel(stable)
            func_motor.setmot()
        if data1 == CMD[15]: # f
            func_motor.apagar()
except KeyboardInterrupt:
    print("Cerrando conexión")
    tcpSerSock.close()
```