UNIVERSIDAD ESTATAL
PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

Propuesta Tecnológica, previo a la obtención del Título de:

INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

“Implementación de un rover automatizado con tecnología de visión artificial para la clasificación de tres tipos de plantas sanas/enfermas en cultivos de ciclo corto”

AUTOR

STEVEN FLORENCIO TIGRERO BACILIO

PROFESOR TUTOR

ING. LUIS ENRIQUE CHUQUIMARCA JIMÉNEZ, MSC.

LA LIBERTAD – ECUADOR

2024

AGRADECIMIENTO

A **Dios** por darme guía, fuerza y sabiduría, para poder tomar buenas decisiones al transcurso de toda mi vida universitaria y culminar con éxito mis estudios y mi proyecto de grado.

A la universidad Estatal Península de Santa Elena, por permitirme adquirir todos los conocimientos para poder culminar mis estudios, quedando así en deuda con tal prestigiosa universidad.

A mi madre, **Maritza Bacilio** por día a día cuidarme y esforzándose por siempre darme un mejor futuro, quien con cariño y paciencia creo un ser humano de bien con disciplina, perseverancia y creatividad, que es una de las cualidades que me caracteriza, ya que ella siempre me apoyo para desarrollar mi imaginación lo cual me permite hoy en día generar diferentes soluciones a un problema planteado, por aquello y miles de cosas más gracias.

A mi padre, **Florencio Tigrero** quien siempre me brinda su consejo y ayuda con algún problema que se me presenta en situaciones de trabajo y cosas técnicas, brindándome muchas opciones y guiándome siempre con paciencia hacia un mejor futuro.

A mis hermanos, **Ivan Tigrero** y **Cecilia Tigrero**, por estar presentes para mi y brindarme su comprensión y paciencia para apoyarme en cualquier proyecto que se me presentaba.

Al **Ing. Luis Chuquimarca, MSc.** por ser mi tutor en este proyecto, por brindarme todo su apoyo, conocimientos y paciencia, al otorgarme de su valioso tiempo, por ayudarme a culminar mi proyecto de grado.

Mis más sinceros agradecimientos

Steven Florencio Tigrero Bacilo

DEDICATORIA

El presente trabajo de titulación se lo dedico a mi madre. Se que se esforzó en darme todo su apoyo, sus enseñanzas se quedaran en mi mente y corazón, siendo siempre una mujer fuerte y sensible a la vez forjando así a este ser humano, que hoy en día logra culminar una etapa más como son los estudios universitarios para poder comenzar de manera independiente mi vida laboral. Espero con todo mi ser que en algún punto pueda devolverte todo el amor y comprensión que me diste al transcurso de toda mi vida y que estoy seguro me seguirás brindando durante mucho tiempo.

A mi padre que quien con su palabra y consejo pude salir adelante en varias situaciones en las cuales me sentía perdido y con poca experiencia para afrontarlos. Quien me guio y me sigue guiando para ser un mejor líder y persona que es una de las cualidades que el ayudo a crear.

Y finalmente a mi hermana y mi hermano, quienes siempre estuvieron dispuestos a ayudarme en todo momento y a quienes siempre acudo por algún consejo y pedido de ayuda.

Steven Florencio Tigrero Bacilio

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del trabajo de titulación denominado: "Implementación de un rover automatizado con tecnología de visión artificial para la clasificación de tres tipos de plantas sanas/enfermas en cultivos de ciclo corto", elaborado por el estudiante Tigrero Bacilio Steven Florencio, de la carrera de Electrónica y Automatización de la Universidad Estatal Península de Santa Elena, me permito declarar que luego de haber orientado, estudiado y revisado, la apruebo en todas sus partes y autorizo al estudiante para que inicie los trámites legales correspondientes.

La libertad, 21 junio del 2024.

Ing. Luis Chuquimarca Jiménez, MSc.

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, Ph. D.
DIRECTOR DE LA CARRERA DE
ELECTRÓNICA Y AUTOMATIZACIÓN

Ing. Junior Figueroa Olmedo, Mgt.
DOCENTE ESPECIALISTA

Ing. Luis Chuquimarca Jiménez, MSc.
DOCENTE TUTOR
DOCENTE GUÍA UIC

Ing. Corina Gonzabay De La A, Mgt.
SECRETARIA GENERAL

RESUMEN

El proyecto consiste en el desarrollo de un rover automatizado equipado con tecnología de visión artificial para la identificación y clasificación de plantas sanas y enfermas en cultivos de lechuga. El objetivo principal es mejorar la eficiencia y precisión del monitoreo agrícola, permitiendo una detección temprana de problemas de salud en las plantas y facilitando la toma de decisiones por parte de los agricultores.

El rover está equipado con cámaras y sensores especializados que capturan imágenes de alta resolución de las plantas en el campo. La tecnología de visión artificial integrada procesa estas imágenes para identificar patrones y características asociadas con la salud de las plantas. Se emplean algoritmos avanzados de aprendizaje automático para entrenar al sistema en la diferenciación entre plantas sanas y enfermas, utilizando conjuntos de datos previamente etiquetados.

El rover se desplaza de manera autónoma por el campo de lechugas, utilizando un sistema de navegación que evita obstáculos y optimiza la cobertura del área de cultivo. A medida que avanza, realiza análisis en tiempo real de las imágenes capturadas, identificando cualquier signo de enfermedad, estrés o deficiencia nutricional en las plantas. Los resultados de la clasificación se almacenan y se pueden visualizar a través de una interfaz de usuario amigable para que los agricultores puedan tomar decisiones informadas sobre la gestión de sus cultivos.

Este enfoque innovador proporciona beneficios significativos, como la reducción de los costos asociados con la mano de obra de monitoreo, la detección temprana de problemas en las plantas, y la optimización de los recursos agrícolas al dirigir tratamientos específicos solo a las áreas afectadas. Además, el sistema puede ser escalable y adaptable a otros cultivos, contribuyendo así a la sostenibilidad y eficiencia en la agricultura.

Palabras claves: Rover automatizado, visión artificial, clasificación de planta, detección temprana, agricultura sostenible.

ABSTRACT

The project consists of the development of an automated rover equipped with artificial vision technology for the identification and classification of healthy and diseased plants in lettuce crops. The main objective is to improve the efficiency and accuracy of agricultural monitoring, allowing early detection of health problems in plants and facilitating decision-making by farmers.

The rover is equipped with specialized cameras and sensors that capture high-resolution images of plants in the field. Integrated machine vision technology processes these images to identify patterns and characteristics associated with plant health. Advanced machine learning algorithms are used to train the system to differentiate between healthy and diseased plants, using previously labeled data sets.

The rover moves autonomously through the lettuce field, using a navigation system that avoids obstacles and optimizes coverage of the growing area. As it progresses, it performs real-time analysis of the captured images, identifying any signs of disease, stress or nutritional deficiency in the plants. The classification results are stored and can be viewed through a friendly user interface so that farmers can make informed decisions about managing their crops.

This innovative approach provides significant benefits, such as reducing costs associated with monitoring labor, early detection of plant problems, and optimizing agricultural resources by targeting specific treatments only to affected areas. Furthermore, the system can be scalable and adaptable to other crops, thus contributing to sustainability and efficiency in agriculture.

Keywords: Automated rover, artificial vision, plant classification, early detection, sustainable agriculture.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

ÍNDICE DE ECUACIONES

INTRODUCCIÓN

El sector agrícola enfrenta constantemente la necesidad de optimizar recursos y reducir las pérdidas relacionadas con enfermedades y plagas de los cultivos. La detección temprana y precisa de los problemas sanitarios de los cultivos es esencial para tomar medidas preventivas y reducir las pérdidas significativamente. En este contexto, la visión artificial está surgiendo como una herramienta prometedora para la captura y el análisis automatizado de imágenes, facilitando la detección temprana de síntomas de enfermedades y anomalías en las plantas.

El robot autónomo propuesto se moverá automáticamente a través de los campos de lechugas, capturando imágenes de las plantas en movimiento. La introducción de algoritmos avanzados de visión artificial permitirá clasificar en tiempo real las plantas en categorías saludables, distinguiendo las ensaladas saludables de las afectadas por enfermedades o condiciones desfavorables. Esta capacidad de evaluación inmediata no sólo acelerará el proceso de seguimiento, sino también facilitará la toma de decisiones informadas para una gestión eficaz de los cultivos.

La eficiencia y la precisión son factores clave que garantizan una producción de alimentos de alta calidad. El uso de tecnologías avanzadas juega un papel fundamental en la mejora de los procesos agrícolas, y la visión artificial se considera una herramienta novedosa para solucionar los problemas de salud de los cultivos. Este proyecto se centra en el despliegue de un vehículo todoterreno autónomo equipado con tecnología de visión artificial.

CAPÍTULO I

1. FUNDAMENTACIÓN.

1.1. Antecedentes.

El desarrollo de las tecnologías, en las últimas décadas ha impulsado de manera positiva a varios sectores productivos como el agrícola. Como lo menciona la Organización de las Naciones Unidas (ONU) en uno de sus artículos llamado "Influencia de las tecnologías digitales", en la actualidad las tecnologías digitales, como son el agrupamiento de datos y la inteligencia artificial se utilizan para rastrear y detectar problemas en varios sectores importantes como la agricultura, la salud y el medio ambiente o simplemente para realizar tareas cotidianas como el pago de una factura o el desplazamiento de un automóvil [1].

La tecnología es uno componentes esenciales en el desarrollo de nuestra sociedad la cual le ha permitido mejorar con el paso del tiempo, en especial desde la revolución industrial donde uno de los sectores que presentó un cambio radical ante su forma de proceso tradicional fue el sector agrícola.

Existen varios enfoques tecnológicos que a lo largo de los años se han venido desarrollando y creando de manera secuencial y exponencial permitiéndonos así el incremento del rendimiento en la producción agrícola a nivel global [2]. Es así como surge el desarrollo de la agricultura de precisión y la agricultura inteligente las cuales utilizan las tecnologías de la información y comunicación pasando a ser considerada "la segunda revolución verde" [3].

Así como en todos los sectores existen factores que afectan su productividad, el sector agrícola no es una excepción ya que es uno de los que tiene mayor número de factores bióticos los cuales afectan a su productividad y rendimiento. En la última reunión que tuvo la organización de las Naciones Unidas que fue el 05 de abril del 2022 se trató el tema de cambio climático y medio ambiente haciendo mención de que las plagas y enfermedades de las plantas ocasionan un 40% de las pérdidas de los cultivos alimentarios y los daños que causan a la agricultura y en la producción de alimentos se acentúan en el hambre del mundo y amenazan los medios de vida rurales [4].

Es por eso que en varios países se están desarrollando nuevas tecnologías en torno al tema de Agricultura de precisión ya que está nos permite ser más eficientes al momento de hacer el cultivo de una planta, además de que gracias a la precisión se pueden ahorrar varios recursos tanto económicos como ambientales [5]. En México al norte del estado de Chihuahua se realizó la implementación de una aplicación móvil que identifica a las plantas de su sector, siendo este un trabajo para la Licenciatura en Ingeniería en Sistemas Computacionales, el cual ayuda a la identificación de plantas [6].

En torno a la agricultura de precisión existen varios artículos, documentos y trabajo de titulación que profundiza en este tema como es el caso del artículo científico publicado el 29 de septiembre de 2017 llamado "procesamiento de imágenes de plantas ornamentales multi escala para calcular su crecimiento" en el cual tenía como objetivo desarrollar un algoritmo que permita calcular el crecimiento de una planta por medio del escalamiento de sus dimensiones, en este artículo también se utilizó un lenguaje de programación accesible como lo es Python y una librería llamada OpenCV [7].

1.2. Descripción del Proyecto.

En este trabajo se tiene planteado crear un dispositivo el cual pueda identificar a la planta por medio de su hoja y clasificarla según corresponda e identificar si posee alguna plaga, para esto vamos a necesitar desarrollar un algoritmo que nos permita identificar a la planta por medio de las dimensiones y colores de su hoja dándonos así información suficiente como para poder clasificarla. El problema aquí es que no muchas personas que están en el sector agrícola saben identificar a un tipo de plaga o de planta específica y sería de mucha ayuda un dispositivo que tenga disponible esta información además de datos como qué tipo de planta es, su lugar de origen y si tiene algún uso medicinal, una de las causas principales de nuestro problema sería el poco conocimiento de herbolaria que existe en personas del sector agropecuario que están iniciando en el mismo creando así un desaprovechamiento del producto además de una pérdida económica considerable. Uno de los factores que beneficia a nuestro problema es que las fuentes de información acerca de estos

temas no son muy accesibles o simplemente no son de fácil entendimiento de modo que la persona opta por métodos empíricos poniendo en riesgo el cultivo y su sustento. Es por eso que para realizar este trabajo de implementación se plantean las preguntas de investigación cómo cuantos tipos de plantas existen en un sector específico en este caso el sector será los huertos urbanos de Santa Elena para esto se va a investigar las plantas nativas de dicho sector, las introducidas e invasoras, resaltando características como de qué lugar son originarias y si tienen algún tipo de uso. Otra pregunta de investigación es cuál es la plaga más común de las plantas de ese sector para esto también se tendrá que realizar una investigación acerca de las plagas que son más frecuentes para luego separar a las más leves de las más fuertes y averiguar información relevante como su nombre, el porqué de su aparición y un método para eliminarla. Y por último tendremos al tratamiento para dichas plagas.

1.3. Objetivos del proyecto.

1.3.1. Objetivo General.

Implementar un rover automatizado con tecnología de visión artificial para la clasificación de tres tipos de plantas sanas/enfermas en cultivos de ciclo corto.

1.3.2. Objetivos Específicos.

- Implementar un rover autónomo que pueda movilizarse en campo agrónomo para recorrer cultivos de ciclo corto.
- Desarrollar un modelo de visión artificial para la clasificación de tres tipos de plantas de cultivos de ciclo corto.
- Desarrollar un modelo de visión artificial para la detección de enfermedades en plantas de cultivos de ciclo corto.
- Implementar en el rover un sistema electrónico para la adquisición de las imágenes y su procesamiento.

1.4. Justificación.

La implementación de este proyecto tiene como motivo principal informar a la gente acerca de los beneficios de una planta, es decir, poder aportar conocimientos herbolarios de suma importancia para las personas que carecen de los mismos. Además se podrá identificar al tipo de planta por medio de su hoja y a su vez poder determinar si ésta tiene algún tipo de plaga o si está enferma, esto resulta muy conveniente para las personas que tienen huertos urbanos que se dedican a la venta de productos orgánicos, como por ejemplo, entre los más comunes tenemos al tomate y a la lechuga que son famosas en el ámbito de compra y venta de productos orgánicos sin embargo dichos productos también atraen plagas y si no se realiza un control de las mismas, esto podría desencadenar a una pérdida de capital invertido. Este proyecto implementará un dispositivo capaz de reconocer tipos de plantas y tipos de plagas para así poder solucionar el problema de plagas en los huertos urbanos.

1.5. Alcance del Proyecto.

En este proyecto se tiene pensado ayudar a las personas que tienen un huerto urbano o que se dediquen a la agricultura. Tomando en cuenta que se deberá hacer un estudio de campo que contemplará las plantas que se contemplan en este documento que son de la provincia y las introducidas para realizar una comparativa entre ellas y poder cultivar las que sean convenientes. Este proyecto tiene un tiempo estimado de implementación de cuatro a 6 meses en los cuales se tiene contemplado el tiempo de investigación, la búsqueda y cotización de los elementos físicos y equipos electrónicos, además el levantamiento de información con respecto a las plantas nativas para la creación de la base de datos para nuestro dispositivo. Cumpliendo así todos los activos planteados en este documento.

1.6. Metodología.

Se realiza la investigación de las librerías y comandos necesarios para cumplir los objetivos planteados, además de revisar documentación con objetivos similares

para la identificación de plantas por medio de la visión artificial, o por medio del entrenamiento de un neurona, como primer paso se realiza un código base, donde se utiliza dos librerías la primera es “Open CV” que nos ofrece una amplia gama de comandos y funciones que nos ayudan con el procesamiento e interpretación tanto de imágenes como de los vídeos [8]. Como en este trabajo hablamos de un sistema en tiempo real, se ampliará la captura de vídeo mediante la cámara de la rashpberry y con la ayuda de la librería anteriormente mencionada, nos proporcionará una perspectiva en dos dimensiones de la hoja a identificar. Como podemos observar en el código, vamos a guardar 3 variables, las dos primeras van a ser para guardar los tonos altos y los tonos bajos del color que vamos a utilizar, en este caso, utilizaremos el color verde ya es el que se presenta de forma más constante en las plantas. En la primera variable se guardan los tonos bajos utilizando un arreglo y coordenadas de los colores HSV, y de la misma manera se hace en la segunda variable, pero con los tonos claros. Por defecto, el programa trabajará con colores RGB, sin embargo, el programa al procesar los datos de manera matemática no lo podrá interpretar de una buena manera, de modo que, en este caso haremos la conversión de colores RGB a HSV, porque nos brindan la facilidad de poder hacer operaciones matemáticas y reducir el porcentaje de error [9]. En la tercera variable llamada “captura”, vamos a guardar la captura del vídeo que hará la cámara de la rashpberry, después de haber declarado todas las variables, abrimos un bucle para que se esté ejecutando de forma infinita, luego utilizamos el comando “.read”, que nos devolverá dos respuestas que tendremos que almacenar en dos variables; en una variable se guardará la respuesta booleana y en la otra se guardará la imagen que devuelve [10]. Abrimos una condición que nos dice que, si la variable booleana es verdadera, ejecutaremos nuestro código principal. En primer lugar, tendríamos que declarar una variable para guardar la conversión de color, luego de haber hecho eso, se realiza 3 máscaras para poder fijar un rango conforme se realizan pruebas, para que una vez estén definidos los límites poder mostrar 4 ventanas que en este caso serían: la ventana con la captura original, la máscara que sólo mostrará una imagen binaria para identificar de mejor manera el color que se quiere extraer de la imagen original, la máscara cortada, que es donde se visualiza solamente el objeto del color que se pide y la máscara “Canny” que realiza un trazo alrededor del borde del

objeto. Se modifica el código base y se aumentan condiciones para que no sólo detecte el color verde dentro de los rangos ya definidos, de manera que también pueda identificar a la planta por medio del área de su hoja. Esto se logra gracias a los comandos de la librería "OpenCV" que nos permiten no sólo marcar los contornos del objeto a identificar, sino que también, puede calcular el área de este utilizando el comando "cv2.contourArea()" [11].

Realizamos una función aparte para poder llamarla cuando la necesitemos y para eso utilizamos "def" que es el análogo de "void" del lenguaje C++ [12]. Una vez realizada esa función identificaremos el área con el comando ya mencionado, ubicando las áreas correspondientes a cada, o en este caso, la hoja de lechuga que es la de mayor área, así que dentro de la cadena de condiciones se ubica primero, luego le sigue la hoja de tomate, que estará entre un rango de área de 2000 a 7000, y, por último, tenemos a la hoja de perejil que tiene un rango de área de 500 a 1000.

En cada una de estas condiciones se declara las variables "X" y "Y" para poder ubicar el centro de la imagen identificada y agregarle un círculo que la haga notar, para esto utilizamos el comando "Circle" de la librería "cv2" u "Open CV" [13]. Luego de esto procederemos a crear un contorno que encierre netamente la figura con él color que se requiere; para que este contorno sea más suave utilizamos el comando "convexHull()" de la librería "cv2", y el comando "putText" para insertar texto que en este caso serían los nombres de las plantas lo cual dependerá de su área [14].

Posteriormente, se procederá a copiar el código base, pero esta vez retiraremos las máscaras que usábamos en ese código, dejando únicamente la que me detectaba el color, dentro de este nuevo bucle se llama a la función dibujar, que es la que tiene las condiciones que detectan el tipo de planta.

Una vez realizadas las primeras pruebas con los códigos base se implementan dentro de la raspberry y se desarrolla el entrenamiento de un modelo de neurona para poder realizar una identificación mas precisa que no dependa de rangos de colores determinados ingresados por los usuarios, si no, que se guie de un modelo pre entrenado para la identificación de las plantas sanas y enfermas.

CAPÍTULO II

2.1. Marco Contextual.

En los sectores de senderos de Nin ubicado en la parte posterior del Autódromo Teófilo Bucaram, que esta atrás de la Universidad Estatal Península de Santa Elena donde se encuentran varias personas y lotes destinados al cultivo de plantas de ciclo corto como lechuga, bróculi y coliflor.

Se plantea la iniciativa para la implementación de este proyecto de titulación en base a la clasificación de tres tipos de plantas sanas/enfermas dentro de cultivos de ciclo corto, contribuyendo a las personas con huertos urbanos afectados por plagas, un dispositivo con herramientas que aporten al buen desarrollo de la planta cultivada.

De modo que el dispositivo a implementar tendrá que procesar las imágenes de la planta a evaluar, tendrá una cámara compatible con la placa de microordenador, además de una estructura lo suficientemente grande para poder recorrer el cultivo sin dañar las plantas.

Por consiguiente, la elaboración de este rover consiste en la evaluación de parámetros como: tamaño de la hoja, color e intensidad; que serán captados mediante la cámara del dispositivo para ser clasificado por la codificación ingresada en el microordenador.

2.2. Marco Conceptual.

2.2.1. Plantas

Las plantas son organismos pluricelulares, con células eucarióticas, y autótrofos, es decir, capaces de producir su propio alimento a través de la fotosíntesis. Junto a los animales, son uno de los grupos más conspicuos y reconocibles de los que componen la biodiversidad [15].

2.2.2. Clases de plantas según duración

Las plantas se clasifican con respecto al tiempo de sembrado hasta el momento de la cosecha, y se denominan de la siguiente manera según [16]:

- **Anuales:** Las plantas anuales son aquellas que germinan, florecen y mueren en el curso de un año.
- **Bienales:** Son plantas que necesitan por lo general dos estaciones o periodos vegetativos desde que se siembra hasta que florecen. Estas plantas crecen vegetativamente el primer año mientras que el segundo florece y dan los frutos.
- **Perennes:** Son las plantas que viven durante más de dos años.

2.2.3. Tipos de planta

Existen diferentes criterios para clasificarlas. Sin embargo, solo se tiene en cuenta dos criterios muy simples como el tamaño y el tiempo de vida, además de tener a la subdivisión por especies como se describe en [17]:

- Árboles: Los árboles son plantas con tallo leñoso que se ramifica a cierta altura del suelo. En general el término hace referencia a las plantas que en su madurez superan una cierta altura, aunque esta altura varía según las fuentes, de los 2 a los 6 metros [18].

- Arbustos: Llamamos arbusto a una planta perenne leñosa que, a diferencia del árbol, no se eleva sobre un solo tronco o fuste, sino que se ramifica desde la misma base [18].

- Matas: La mata es un subarbusto o arbusto enano que se distingue del arbusto por la disposición de las ramas a ras del suelo, y por tener menor altura (no suelen superar los 20 cm) [16].

- Hierbas: Es una planta que no presenta órganos decididamente leñosos. Los tallos de las hierbas son verdes y generalmente mueren al acabar la buena estación, siendo luego reemplazados por otros nuevos [16].

2.2.4. Lechuga

La lechuga es una hortaliza muy popular que se cultiva por sus grandes hojas que en algunos casos se aprietan formando repollos más o menos compactos. Existen muchísimas variedades de esta verdura, y el número aumenta cada año. Las propiedades nutritivas de esta planta son escasas, siendo un alimento bajo en calorías que aporta algunas vitaminas y minerales. Está formada por grandes hojas que se disponen unas sobre otras formando en algunos casos un repollo. Las hay de diferentes formas, tamaños y colores, habiendo lechugas verdes y rojas. Pueden tener las hojas más o menos dentadas. Existen muchas dudas sobre su origen, aunque se sabe que era conocida por las civilizaciones antiguas. Los romanos extendieron su cultivo por el resto de Europa [19].

- **Ciclo de vida de la lechuga:** La lechuga de manera general es una planta anual de la familia de las compuestas, su duración de cultivo o ciclo de vida radica entre los 50 hasta los 60 días para las variedades de lechuga temprana, a diferencia de las variedades tardías, donde, su ciclo de vida es aproximadamente entre 70 y 80 días, desde la plantación hasta la recolección del producto [20].
- **Dimensiones:** Según [21] la planta es de un tamaño pequeño con textura suave, su longitud promedio es de 12 cm y de diámetro mide 7 cm.
- **Modo de siembra:** La distancia de plantación depende en la mayoría de las plantas de este tipo de la envergadura que alcance la variedad, por ejemplo, para las lechugas foliares se utilizan distancias de siembre entre 20 a 30 cm entre cada planta y de 20 a 30 cm entre cada uno de los surcos [21].

La lechuga se consume cruda, formando parte de ensaladas y acompañada de tomate y cebolla entre otros ingredientes. También puede consumirse cocida o asada, aunque esto es menos frecuente. Además, puede formar parte de menestras y ensaladas.

Es un alimento pobre en nutrientes, con muy poco contenido graso, lo que la hace indicada para dietas de adelgazamiento. Aporta algunas vitaminas, como la C y ácido fólico, y pequeñas cantidades de fósforo, potasio, hierro y calcio. Se

considera que tiene propiedades calmantes y sedantes [19]. Un ejemplar se puede observar en la figura1.

Figura 1. Lechuga.

Fuente: Autoría propia.

2.2.5. Plagas en la lechuga

Conjunto de especies implicadas en la transmisión de enfermedades infecciosas para el hombre y en el daño o deterioro del hábitat y del bienestar urbano, cuando su presencia es continua en el tiempo y está por encima del umbral de tolerancia [22]. Tanto si trabajas en el sector de la horticultura o agrícola, como si tienes un pequeño huerto en tu jardín par tu disfrute, debes poner atención a las plagas del campo más frecuentes [23].

Existen una gran cantidad de plagas de las plantas, vamos a contarte las más habituales para que puedas estar prevenido:

- Pulgón: son la plaga principal de la mayoría de los cultivos, causando deformaciones en las hojas, tallos, debilitando de manera general a la planta, sin embargo, cuenta con una mayor diversidad de enemigos naturales, como consecuencia de su rápida propagación [24].

Figura 2. Pulgón.

Fuente: [24].

- Mosca blanca: Pueden causar perdidas en los cultivos de hasta un 100%, sin embargo, en la provincia de Manabí, la Península de Santa Elena y el Valle del Chota, este insecto provoca entre el 25% y 50% de perdida [25].

Figura 3. Mosca blanca.

Fuente: [25].

- Trips: Son insectos que se desarrollan en las hojas y brotes, se dispersan desde las hojas viejas donde se encuentran los ejemplares más longevos, hasta los brotes de la planta en donde se permanecen las larvas de esta plaga [26].

Figura 4. Trips.

Fuente: [26].

- Minadores: Los minadores de hoja pueden provocar danos directos o indirectos que dependen de la gravedad de las heridas causadas por esta plaga, ya que se expone a hongos o bacterias del ambiente, esta plaga realiza su recorrido desde el punto de inversión del peciolo can el limbo de la hoja [27].

Figura 5. Plaga de minador.

Fuente: [27].

2.2.6. Normativas de Agrocalidad del Ecuador

Según la Agrocalidad, Agencia de Regulación y control de Fito y Zoosanitario del Ecuador, en el capítulo VI detalla el establecimiento del cultivo con sus respectivas normativas desglosadas en el Articulo 10 denominado "Siembra", el cual contiene los siguientes apartados:

a) Se recomienda sembrar variedades de hortalizas y/o verduras con características adaptables a la zona agroecológica y que satisfagan las exigencias del mercado.

b) La densidad de siembra debe ser adecuada al medio, a la especie y a la variedad.

c) Se recomienda utilizar semillas y plántulas de calidad, para asegurar la producción y alcanzar rendimientos óptimos.

d) En el caso de realizar la obtención o propagación del material vegetal para uso interno en la UPA, se debe asegurar la calidad del proceso y sus resultados.

e) La preparación de los sustratos para los semilleros, deben considerar las condiciones físicas, nutricionales y fitosanitarias para el óptimo desarrollo de las plántulas.

2.2.7. Visión artificial

La visión artificial entrena a las máquinas para realizar estas funciones, pero tiene que hacerlo en mucho menos tiempo con cámaras, datos y algoritmos en lugar de retinas, nervios ópticos y una corteza visual. Como un sistema entrenado para inspeccionar productos o ver un activo de producción puede analizar miles de

productos o procesos por minuto, detectando defectos o problemas imperceptibles, puede superar rápidamente las capacidades humanas [28].

Al igual que la inteligencia artificial, no es nada nuevo en la práctica industrial. Por ejemplo, la industria manufacturera ha estado utilizando sistemas de visión artificial durante varios años y los beneficios son fáciles de ver. El uso de robots en la agricultura ha seguido creciendo en los últimos años a medida que nuevos avances mejoran la tecnología y facilitan su adopción por parte de los agricultores. Sin embargo, los avances en los sistemas de visión por computadora pueden hacer que los sistemas robóticos sean mucho más independientes y brindarles los medios para recopilar datos valiosos durante las operaciones de una empresa.

Por ejemplo, ahora se están integrando sistemas de reconocimiento facial en sistemas de visión robótica para permitir la identificación individual de animales y mejorar el seguimiento del rebaño minimizando al mismo tiempo la necesidad de intervención humana. El análisis de fenotipado de plantas de alto rendimiento es otro ejemplo de cómo los sistemas de visión artificial pueden recopilar y medir características de las plantas para crear bases de datos que pueden ser vitales para la genética vegetal y el desarrollo de cultivos resilientes [29].

2.2.8. Identificación en visión artificial

Identificación es la acción y efecto de identificar o identificarse, reconocer si una persona o una cosa es la misma que se busca, hacer que dos o más cosas distintas se consideren como una misma, llegar a tener las mismas creencias o propósitos que otra persona, dar los datos necesarios para ser reconocido [30].

Identificar una especie requiere reconocer una o más características de la planta y asociarlas al nombre, independientemente de si se trata del nombre común o del llamado nombre científico. La gente suele prestar atención a una o más de las siguientes características: toda la planta (tamaño, forma, etc.), flores (color, tamaño, lugar de crecimiento, inflorescencia, etc.), tallo (forma, botón, exterior), patrones de corteza, etc.), frutos (tamaño, color, calidad, etc.) y hojas (forma, bordes, patrones, textura, venas) [31].

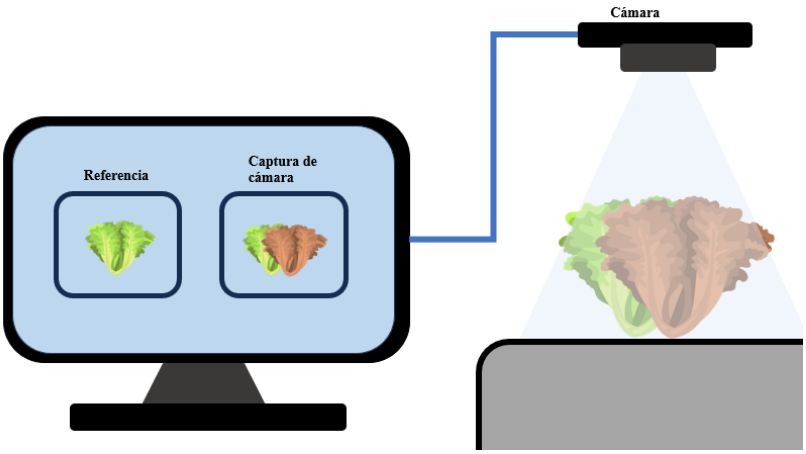


Figura 6. Identificación en la visión artificial.

Fuente: Autoría propia.

2.2.9. Clasificación en visión artificial

Es un ordenamiento o una organización de cosas en una serie de categorías o clases. Se pueden clasificar ideas, objetos o cualquier tipo de referente. De hecho, clasificar se define en el diccionario de la Real Academia Española como "disponer por clases algo", o sea, organizar y dividir un conjunto de cosas según un criterio escogido de antemano [32].

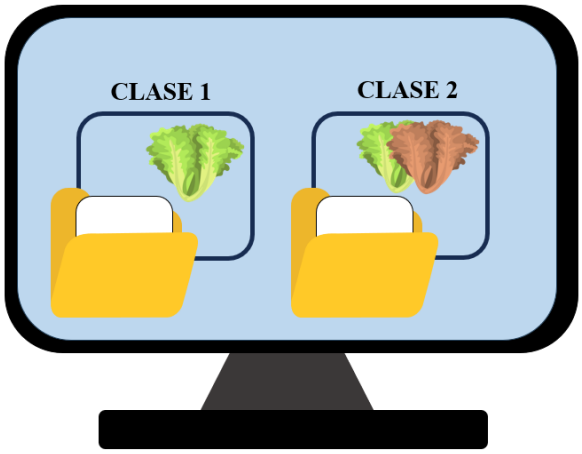


Figura 7. Clasificación en la visión artificial.

Fuente: Autoría propia.

2.2.10. Arquitectura del modelo utilizado en visión artificial

Una de las arquitecturas más utilizadas es la Perceptrón multicapa para la clasificación de patrones, además su aprendizaje es supervisado, lo que quiere decir, es que la red reconoce la salida que debe generar, este tipo de entrenamiento consiste en actualizar los pesos de sus conexiones cuando se presente un error entre la salida obtenida y la salida deseada [33].

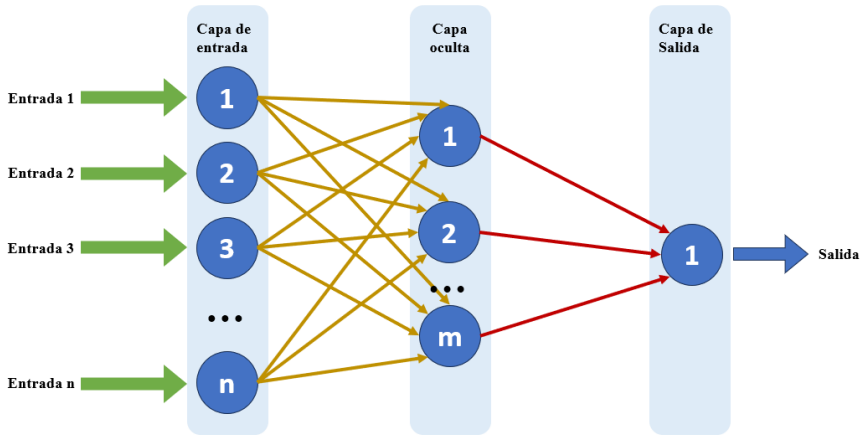


Figura 8. Arquitectura de un perceptrón multicapa.

Fuente: Autoría propia.

2.2.11. Microprocesador

Es un dispositivo electrónico digital, integrado, programable y de actuación secuencial que constituye una base tangible más elemental de un sistema de cómputo, como lo es una laptop o un ordenador de escritorio, además es capaz de ejecutar programas a partir de instrucciones en lenguaje de máquina para realizar operaciones aritméticas, lógica con un acceso a la memoria en tiempos cortos [34].

2.2.12. Robots móviles

Los robots móviles son robots que tiene la capacidad de moverse desde un punto a otro autónomamente, es decir, que pueden moverse sin la asistencia externa de un controlador u operador humano. A diferencia de la mayoría de los robots industriales que pueden moverse en un área específica de trabajo [35]. Existen diferentes tipos de robots móviles como:

- **Robots terrestres:** En su mayoría sus movimientos son a causa de ruedas impulsadas por un motor, sin embargo, existen algunos robots que imitan las piernas de humanos o las patas de algún animal [36].

- **Robots aéreos:** Los más comunes dentro de esta categoría se encuentran los UAV (vehículos aéreos no tripulados) [37].

- **Robots acuáticos:** Este tipo de robot son los más adecuados para la movilidad en entornos marinos, con fines exploratorios, son más conocidos como vehículos marinos autónomos [38].

A los robots terrestres se los clasifica por el medio de locomoción que utiliza para desplazarse dentro de un entorno, los sistemas de locomoción más usados en la actualidad son: ruedas, patas, orugas [39].

2.2.13. Robots móviles terrestres con ruedas

Los robots móviles con ruedas son vehículos capacitados para movilizarse sobre una superficie u entorno elegido por el usuario, obteniendo tracción de ruedas sujetadas al chasis del robot móvil. Utilizar estos tipos de robot representan un menor consumo de energía y se mueven más rápido que otros mecanismos de locomoción, además desde el punto de vista del control, es mucho más fácil, debido a que posee mecanismos más simples y no presenta problemas en la estabilidad [40].

Los tipos de ruedas están atados a la restricción de la movilidad que presenta el robot, es decir, que cada rueda que esté presente en el prototipo contribuye al movimiento del robot, además de aplicar restricciones en su movimiento reduciendo los grados de movilidad del vehículo [39].

- **Rueda fija:** La rueda fija no tiene ninguna articulación para que pueda cambiar su dirección, su nombre se da porque su posición es fija con respecto al robot móvil, por lo general, este es uno de los sistemas de movimiento más simples [41].

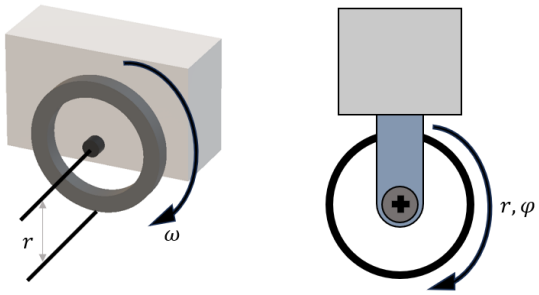


Figura 9. Rueda fija.

Fuente: Autoría propia.

- **Rueda de centro orientable:** Este tipo de rueda tiene una articulación para la dirección, esto le permite que exista movimiento independiente con respecto a la estructura o chasis del robot móvil, una de sus características es que el eje de dirección pasa por el centro de rotación de la rueda [39].

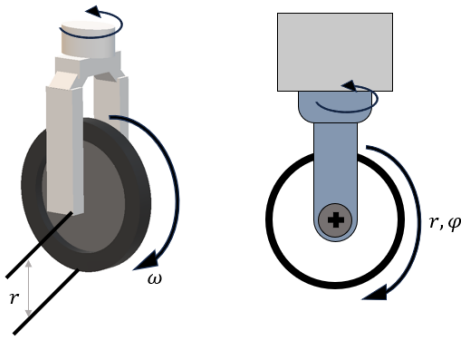


Figura 10. Rueda de centro orientable.

Fuente: Autoría propia.

- **Rueda de centro orientable desplazado:** Es un tipo de rueda orientable que posee una articulación de dirección con respecto a la estructura del robot, con la diferencia que el eje vertical no pasa por el centro de la rueda como se mencionó en la rueda de centro orientable [42].

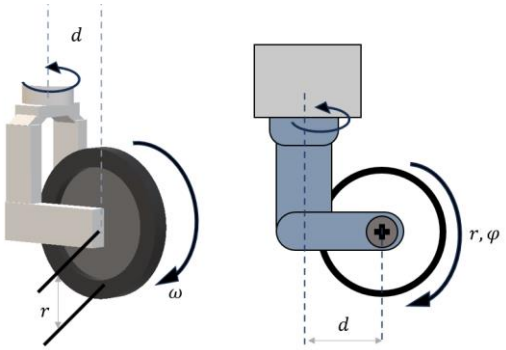


Figura 11. Rueda de centro orientable desplazado.

Fuente: Autoría propia.

- **Rueda omnidireccional o sueca:** La rueda se caracteriza por poseer varias ruedas direccionales pequeñas que se desplazan a 45 o 90 grados con respecto a la superficie de rodado, con el objetivo de que la rueda más grande se mueva hacia delante y hacia atrás mientras que las ruedas pequeñas se mueven indirectamente, otorgando una movilidad en las 4 direcciones de un plano [39].

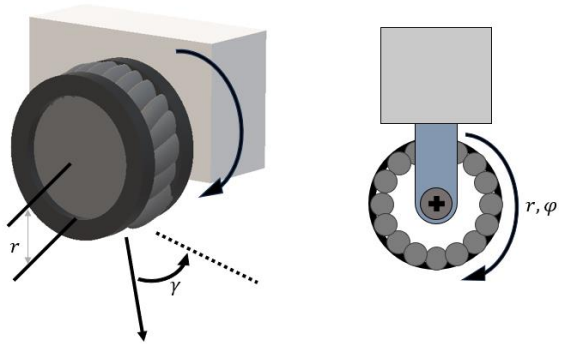


Figura 12. Rueda omnidireccional.

Fuente: Autoría propia.

2.2.14. Configuración de robots móviles terrestres

Según la disposición y los tipos de rueda se pueden detallar una variedad de configuraciones para los robots móviles con distintos tipos de maniobrabilidad, esto se refleja a la selección y la ubicación del tipo de rueda en la estructura del robot, además de tener en cuenta la determinación de los parámetros cinemáticos. De manera general se clasifican por el número de ruedas que tiene [43].

- **Robot con una rueda:** Este tipo de robot es considerado inestable debido a su incapacidad de mantener el equilibrio, sin embargo, existen varios como el robot monociclo o robots esféricos. Este tipo de robots no son muy utilizados ya que requieren de mecanismos de equilibrio adicionales y su control dinámico es difícil [44].

- **Robot con dos ruedas:** Dentro de la figura 6 se observa la primera configuración de tipo bicicleta, la que cuenta con una rueda que genera movimiento y una rueda delantera que se encarga de la dirección, la estabilidad dinámica aumenta a medida que aumenta su velocidad, es decir, que no puede mantener la estabilidad si se detiene su movimiento [45].

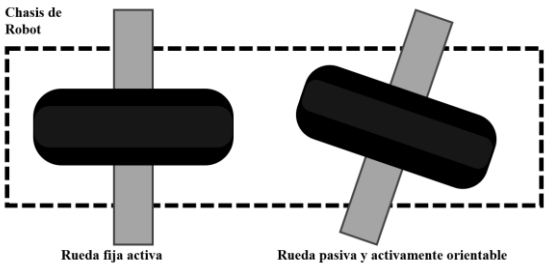


Figura 13. Robot tipo bicicleta.

Fuente: Autoría propia.

En la segunda configuración tenemos un robot tipo péndulo invertido como se observa en la figura 7, también se conoce como robot de accionamiento diferencial de dos ruedas, este robot presenta una desventaja que es la complejidad para realizar el control del equilibrio dinámico [46].

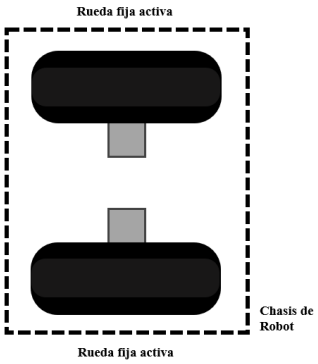


Figura 14. Robot tipo péndulo invertido.

Fuente: Autoría propia.

- **Robot con tres ruedas:** Este tipo de configuración tipo triciclo es más común con dos ruedas fijas en un mismo eje y una rueda de centro orientable que realiza la función de dar la dirección al robot móvil [47].

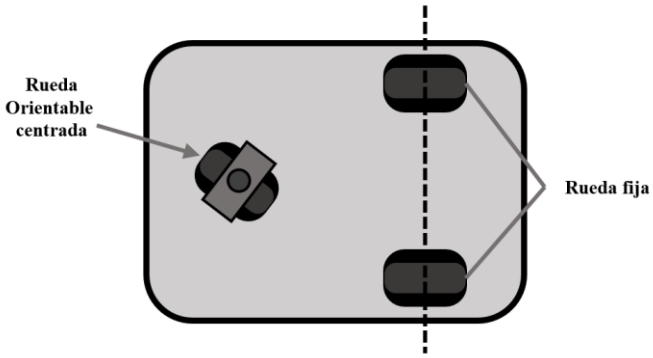


Figura 15. Robot con configuración de triciclo.

Fuente: Autoría propia.

- **Robot con cuatro ruedas:** Existen varias configuraciones para el robot de cuatro ruedas, una de las más usadas es la configuración Ackerman, este diseño cuenta con una buena estabilidad debido a que tiene dos ruedas traseras fijas sobre un mismo eje y dos ruedas delanteras que se dirigen sincrónicamente con respecto al centro instantáneo de rotación ICR [48].

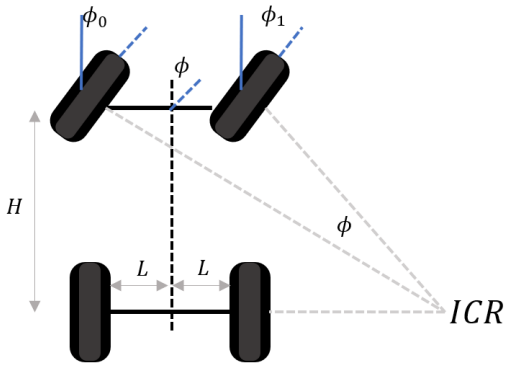


Figura 16. Configuración Ackerman.

Fuente: Autoría propia.

2.2.15. Tracción y dirección de robots móviles terrestres

Estas características de los robots móviles se clasifican por la ubicación de las ruedas, los algoritmos de control de los motores y la mecánica asociada [43]. Dentro de [43] se plantean tres configuraciones de tracción y dirección:

- **Tracción y dirección en ejes independientes:** Dentro de esta primera configuración la tracción se efectúa en las ruedas traseras, mientras que la dirección de robot móvil está destinada para las ruedas delanteras [42].

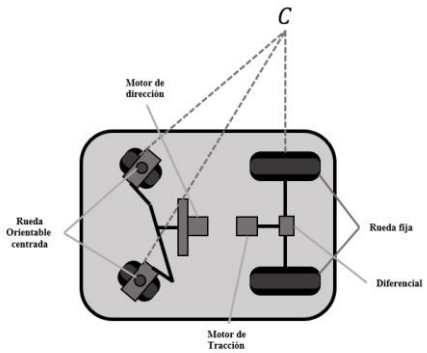


Figura 17. Tracción y dirección en eje diferentes.

Fuente: Autoría propia.

Tracción y dirección en un mismo eje: En esta configuración la tracción se logra ubicando motores independientes en las ruedas de un mismo eje, es muy utilizada para fines prácticos, sin embargo, una de sus desventajas en que los motores deben tener las mismas características con el fin de que el control sea simple [49].

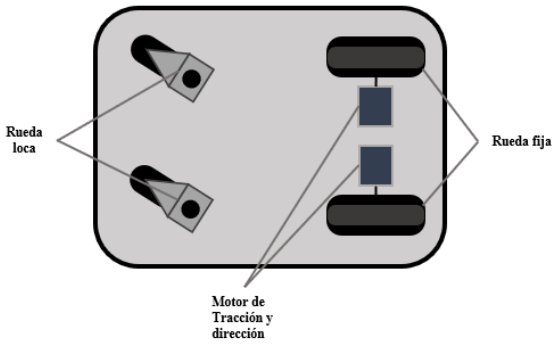


Figura 18. Tracción y dirección en un mismo eje.

Fuente: Autoría propia.

Tracción y dirección sobre todos los ejes: En la figura 19 se observa una de las configuraciones más utilizadas de tracción sobre todos sus ejes, esta se aplica cuando se trabaja en un entorno de difícil acceso en donde la velocidad no es imprescindible, de manera que pasa a un segundo plano ya que se requiere de mejor adherencia del terreno [50].

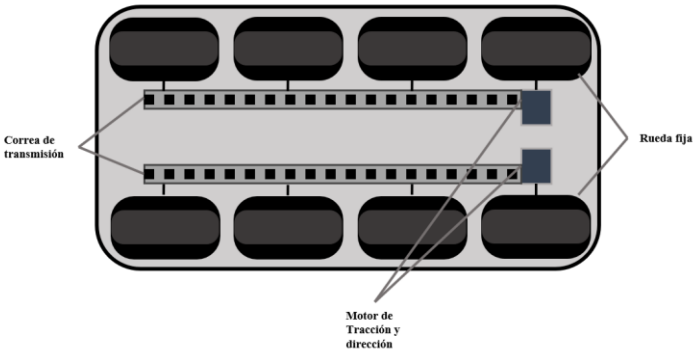


Figura 19. Tracción y dirección sobre todos los ejes.

Fuente: Autoría propia.

2.2.16. Cinemática de un robot móvil terrestre

La cinemática se encarga de estudiar y analizar la configuración de los robots en el espacio de trabajo que se le asigne, además de tener en cuenta sus parámetros geométricos, así como las limitaciones definidas en su trayectoria. Las ecuaciones cinemáticas dependen de la estructura geométrica del robot, por ejemplo, un robot fijo puede tener una estructura cartesiana, esférica o cilíndrica, además de que puede tener una o más ruedas, con o sin restricciones en su movimiento. Dentro de la cinemática se encuentran diferentes fórmulas como se muestra en [51]:

Cinemática directa e inversa

Es la definición de las coordenadas generalizadas q en el espacio articular y p en el espacio de tareas:

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}, p = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Ecuación 1. Ecuación de cinemática directa e inversa.

En donde se plantea que el problema de cinemática directa es:

$$p = f(q)$$

Ecuación 2. Problema de cinemática directa.

Y el problema de cinemática inversa es:

$$q = f^{-1}(p)$$

Ecuación 3. Problema de cinemática inversa.

Cinemática diferencial

$$d\boldsymbol{p} = \boldsymbol{J} d\boldsymbol{q}$$

Ecuación 4. Relación diferencial directa.

Donde **J** es la matriz jacobiana que relaciona las velocidades articulares con las vectoriales en el espacio.

2.2.17. Dinámica de robots móviles terrestres

La dinámica estudia las fuerzas y torques que causan el movimiento de los robots móviles, además implica una relación entre las fuerzas que actúan sobre el robot y su movimiento, teniendo en cuenta su masa y momento de inercia. Dentro de la dinámica de robots móviles tenemos el siguiente modelo dinámico general, según la segunda Ley de Newton para sistemas de múltiples grados de libertad [51]:

$$\mathbf{M}(q)\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{G}(q) = \tau$$

Ecuación 5. Modelo dinámico general.

Donde:

- $\mathbf{M}(q)$ es la matriz de inercia.
- $\mathbf{C}(q, \dot{q})$ es la matriz de Coriolis y fuerzas centrífugas.
- $\mathbf{G}(q)$ es el vector de fuerzas gravitacionales.
- τ es el vector de torques o fuerzas actuadoras.

2.3. Marco Teórico.

La organización de las Naciones Unidas nos dice que las plagas pueden traspasar fronteras por medio del comercio de diferentes tipos de plantas que se exporta e importa a varios países, y que este tipo de propagación es muy peligrosa y dañina tanto para los agricultores del sector como para las especies nativas del mismo lugar.

Los árboles pueden ser susceptibles a llenarse de plagas cuando están en un ambiente que no es muy favorable para su crecimiento, por ejemplo, la falta de agua que no necesariamente debe ser por un ambiente seco sino por la abundante vegetación existente en el lugar, también tenemos a la falta de luz como uno de estos factores. Esto no quiere decir que la planta no se desarrollara en varios casos las plantas se adaptan y crecen con los recursos disponibles en el sector donde crecieron, de aquí vienen los problemas y comienzan a llegar las plagas y enfermedades que afectan más a las plantas que están estresadas [52]. En uno de los artículos de Red agrícola resalta las causas bióticas y abióticas que favorecen a las plagas en plantas de fruto cítrico, explica que mientras las plagas tengan un ambiente húmedo, su mortalidad será baja, además de que, si la planta esta sana provee de un buen alimento a la plaga lo que hace difícil su control, es por eso que plantea varios escenarios en los cuales va variando los factores ambientales para ver un escenario óptimo para el control de las plagas [53].

En el artículo informativo publicado en la página oficial del diario "El universo", plantea que las plagas y lluvias afectan el sector agrícola de Santa Elena causando la perdida de la próxima cosecha, aquí se resaltan los cultivos más afectados que son los de sandia, melón y maíz, los cuales al perderse por el mal tiempo provocaron una pérdida económica considerable ya que en el cuidado de estos cultivos se realizó una mayor inversión. También presenta una de las plagas que ataca mayormente al maíz el cual es el gusano cogollero [54].

Dentro de este trabajo [55] se plantea una contraparte para el uso de los plaguicidas. Nos dice que dichos productos afectan en diferentes formas al ambiente y nos presenta de manera individual los efectos adversos a corto plazo en el ambiente cercano, efectos adversos a largo plazo en el ambiente lejano, efectos en el ambiente abiótico y biótico. En este trabajo de titulación de la universidad de

Valladolid, se presenta una propuesta innovadora que trata de una aplicación que permitirá guiar de manera intuitiva a las personas que trabajan en el sector agrícola o a las personas que van a incorporarse en el mismo por medio de una aplicación para teléfono móvil [56].

Este articulo realiza una comparación entre diferentes métodos de identificación de hojas para la extracción de sus características como los las texturales, cromáticas y geométricas. Esto nos ayudara a tener una vista más amplia de los métodos para identificar una planta por medio de su hoja [57].

Existe un software que sirve para la identificación y localización de una planta y sus similares, utilizando inteligencia artificial que buscara en repositorios de la comunidad científica para analizarlos y encontrar patrones que lo identifiquen como una variedad de esa planta que en este caso se centró en la quinua, presentado en [58].

En [6] se presenta una aplicación que identifica plantas del norte del estado de Chihuahua justificando que existen herramientas que se usan para la identificación de plantas pero que son complejas y no son intuitivas, la aplicación tiene su data base propio recolectado por los autores de la propuesta junto con otras fuentes para la implementación de su aplicación.

La propuesta utiliza visión artificial como media para la identificación de plantas sansas o enfermas, de modo que la parte de procesamiento de imágenes con OpenCV y Python es necesaria ya que serán conceptos útiles para desarrollar el proyecto de investigación debido a que usan dicho método para calcular la maduración de un tomate por medio de los colores del fruto en cuestión en [59].

En [60] se plantean los problemas de la clasificación de imágenes para la detección de maleza en cultivos de cereales además nos muestra las diferentes metodologías y procesos que se ocupó como el “machine learning”, redes neuronales, “Deep learning”, clasificación “Pixel by pixel”, etc. Que nos ayudaran a tener una visión más amplia de los métodos a utilizar dentro de la propuesta.

En este trabajo se desarrolla una aplicación que pretende clasificar la hoja según la especie de árbol a la que pertenezca usando técnicas de visión artificial que

analicen la foto de la hoja. Esta aplicación tendrá una base de datos ampliable pero que en un principio tendría 6 especies registradas.

Este articulo propone una metodología para el análisis y detección de enfermedades que se presentan en las hojas de las plantas por medio de técnicas de procesamiento de imágenes. Los resultados experimentales demuestran que el sistema puede detectar con éxito a cuatro principales enfermedades además de clasificarlas [61].

En el artículo [62] se muestra el desarrollo de un sistema que detecta malezas y aplica herbicidas en un cultivo de pina, usando conceptos de agricultura de precisión. Este prototipo que se plantea procesa imágenes en tiempo real y es implementado en un vehículo que lo hará paras por los surcos del cultivo creando así un herbicida selectivo.

Se presentan dos algoritmos de procesamiento de imágenes para la identificación del café que tiene mejor calidad para su producción. El primero es el algoritmo clasificador que tiene imágenes para la comparación de café maduro y verde. Por otra parte, se tiene al algoritmo de detección de broca el cual fue desarrollado a base de un criterio de bancarización para detectar las zonas negras de la imagen que serán interpretadas como agujeros que le dejo la plaga al grano de café en [63].

CAPÍTULO III

3.1. Componentes de la propuesta

En la siguiente sección se realiza la selección de cada uno de los componentes físicos y lógicos, que se utilizaron para la implementación del presente proyecto, detallando las características técnicas importantes, además de una explicación breve de su funcionamiento.

3.1.1. Componentes físicos

Para llevar a cabo la implementación del rover automatizado, que detecta e identifica plantas de ciclo corto sanas o enfermas mediante el uso de visión artificial, se utilizaron varios equipos y componentes electrónicos que se detallan en la siguiente sección.

3.1.1.1. Raspberry Pi 4 - 8GB RAM

Es una computadora de bajo costo y con un tamaño compacto, del porte de una tarjeta de crédito, puede ser conectada a un monitor de computador o un TV, y usarse con un mouse y teclado estándar. Además, cuenta con un sistema operativo Linux capaz de permitirle a las personas de todas las edades explorar la computación y aprender a programar lenguajes como Scratch y Python. Es capaz de hacer la mayoría de las tareas típicas de un computador de escritorio, desde navegar en internet, reproducir videos en alta resolución, manipular documentos de ofimática, hasta reproducir juegos [64].

Figura 20. Raspberry pi 4.

Fuente: [64].

Tabla 1. Características técnicas del Raspberry Pi 4.

CARACTERÍSTICAS TÉCNICAS	
Modelo	Raspberry Pi 4 B
CPU	4 núcleos de 1.5 Hz
GPU	
Memoria	8GB LPDDR4
Conectividad	LAN inalámbrica IEEE 802.11b / g / n / ac de 2.4 GHz y 5.0 GHz, Bluetooth 5.0, BLE
Video y sonido	2 puertos micro HDMI (hasta 4Kp60 admitidos)
Puertos	2 puertos USB 3.0, 2 puertos USB 2.0
Alimentación	5V DC a través del conector USB-C (mínimo 3A)
Expansión	Estándar de 40 pines

3.1.1.2. Motorreductores

Este motorreductor es un pequeño dispositivo electromecánico que conecta un motor de CC a una caja de cambios que reduce la velocidad del motor y aumenta su par o fuerza. Este tipo de motorreductor es ideal para proyectos de robótica, electrónica y automatización utilizando un Arduino porque permite controlar el movimiento de ruedas, palancas, elevadores y otros mecanismos. Estos motorreductores Arduino se caracterizan por su tamaño compacto, bajo coste, buena adherencia en muchas superficies diferentes y se pueden reconocer fácilmente por su color amarillo. También están disponibles con diferentes relaciones de transmisión, como 48:1 o 120:1, lo que significa que el motor gira 48 o 120 veces más rápido que el eje de salida de la transmisión. De esta forma, podrá elegir el motorreductor más adecuado en función de la velocidad y el par requerido para cada aplicación [65].

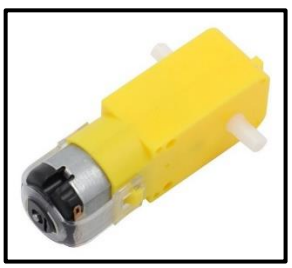


Figura 21. Motorreductor.

Fuente: [65].

Tabla 2. Características técnicas de Motorreductor.

CARACTERÍSTICAS TÉCNICAS	
Torque	1 Kg/cm
Voltaje de operación	3 a 12 V
Velocidad de giro	200 RPM a 5V
Consumo de corriente sin carga	70 mA
Consumo de corriente con carga	140 mA a 250 mA
Relación de reducción	48:1

3.1.1.3.Cámara

Esta cámara de 5 megapíxeles puede grabar vídeos e imágenes de 1080p (1920x1080) cuando se conecta a la placa Raspberry Pi. Para usarlo, simplemente se conecta con un cable plano directamente al conector SCI (Interfaz serie de cámara) de Raspberry Pi, se descarga la última versión de Raspbian y luego, después de reiniciar el Pi, se debe habilitar la compatibilidad con video a través de "raspi-config". La placa de circuito es muy pequeña, mide aproximadamente 25 x 20 x 9 mm y pesa sólo 3 gramos, lo que la hace ideal para aplicaciones móviles u otras aplicaciones donde el tamaño es fundamental. Para imágenes, el sensor es capaz de capturar hasta 2592x1944. Sin embargo, las siguientes resoluciones son compatibles en el modo de vídeo: 1080p30, 720p60 y 640x480p60/90 [66].

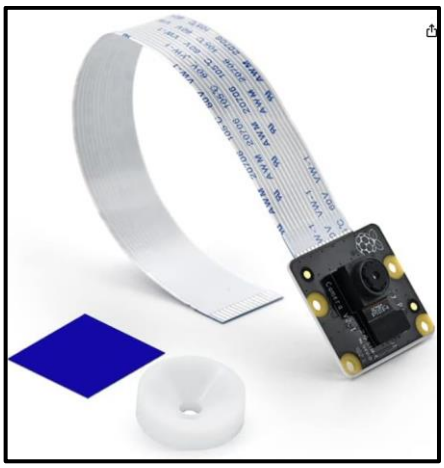


Figura 22. Cámara de 5 MPX.

Fuente: [66].

Tabla 3. Características técnicas de la Cámara de Raspberry.

CARACTERÍSTICAS TÉCNICAS	
Modelo	Cámara Raspberry Pi 4 genérica
Resolución	¼ 5M
Angulo de visión	54 x 41 grados
Video	1080p a 30 fps
Tamaño de pixel	1.4 x 1.4 um

3.1.1.4.Puente H L298N

El módulo de control de motores puente H L298N nos permite controlar la velocidad y dirección de dos motores DC o motores paso a paso de forma muy sencilla gracias a los dos puentes H instalados en él.

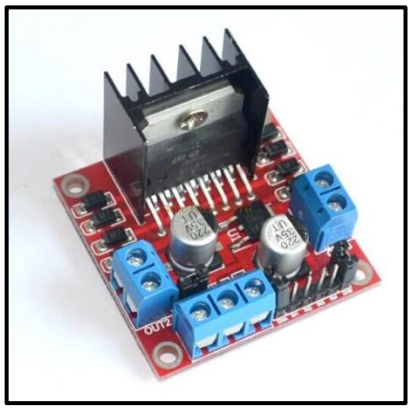


Figura 23. Puente H L298N.

Fuente: Autoría propia.

Tabla 4. Características técnicas de Puente H L298N.

CARACTERÍSTICAS TÉCNICAS	
Alimentación	4.5V a 36V
Voltaje de entrada	4.5 V a 7V
Corriente por canal	600 mA
Corriente por salida máxima	1.2 A
Numero de pines	16

3.1.1.5.Batería lipo

Las baterías lipo (abreviatura de Litio y Polímero) son baterías recargables, que a veces constan de varias celdas, que se utilizan en aplicaciones que requieren una corriente superior a 1 A y se caracterizan por su peso ligero y su tamaño pequeño.

Figura 24. Batería lipo de 3500 mA.

Fuente: Autoría propia.

Tabla 5. Características técnicas de Batería Lipo.

CARACTERÍSTICAS TÉCNICAS	
Capacidad	3500mAh
Voltaje	7.4V
Descarga constante	25C

3.1.1.6.MPU6050

En la Figura se encuentra el sensor, el cual contiene un acelerómetro y un giroscopio MEMS (Micro Electro Mechanical Systems) integrado en un solo chip que permite medir el movimiento en 6 grados de libertad según el sistema de coordenadas cartesiano. La comunicación del módulo es por I2C (protocolo de comunicación de dos vías), esto permite trabajar con la mayoría de los microcontroladores. Los pines SCL y SDA (vías de comunicación I2C) tienen una resistencia pull-up en placa para una conexión directa al microcontrolador sin necesidad de una protección extra [67].

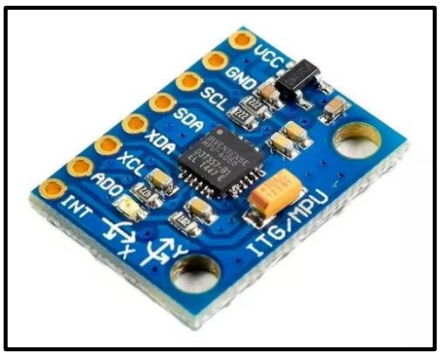


Figura 25. MPU6050.

Fuente: [67].

Tabla 6. Características técnicas de MPU6050.

CARACTERÍSTICAS TÉCNICAS	
Voltaje de alimentación	3V/3.3V ~ 5VDC
Rango de Acelerómetro	2g/4g/8g/16g
Rango de Giroscopio	200Grad/seg, 500Grad/seg, 1000Grad/seg
Sensibilidad Giroscopio	131 LSBs/dps
Interfaz	I2C

3.1.1.7.Regulador de voltaje LM2596 DC-DC

El convertidor LM2596 es un dispositivo que reduce el voltaje de entrada a un nivel de voltaje de salida constante, sin importar los cambios en la entrada o la carga. Admite hasta 3A de corriente de salida, con un rango de entrada de 4.5 a 40 voltios y un rango de salida de 1.23 a 37 voltios. El voltaje de salida se ajusta mediante un potenciómetro de múltiples vueltas [68].

Figura 26. Regulador de voltaje LM2596 DC-DC.

Fuente: [68].

Tabla 7. Características técnicas de Regulador de voltaje LM2596 DC-DC.

CARACTERÍSTICAS TÉCNICAS	
Voltaje de entrada	4.5V a 40V DC
Voltaje de salida	1.23 V a 37V DC
Corriente de salida	máx. 3A, 2.5A recomendado
Potencia de salida	25W
Frecuencia de trabajo	150KHZ

3.1.2. Componentes lógicos

En la siguiente sección se muestra a los softwares que se utilizaron en el presente proyecto, para realizar la implementación del mismo.

3.1.2.1.Raspberry Pi Imager

Raspberry Pi Imager es una herramienta diseñada para facilitar la instalación del sistema operativo en tarjetas microSD utilizadas con dispositivos Raspberry Pi. Esta aplicación permite crear rápida y fácilmente un dispositivo de arranque preparando una tarjeta microSD para usar con Raspberry Pi [69]. Con esta herramienta, se puede instalar muchos sistemas operativos diferentes disponibles para Raspberry Pi, sin embargo, para el desarrollo de este proyecto instalaremos una versión que tenga disponible las dependencias y librerías que se requieran utilizar para nuestro sistema.

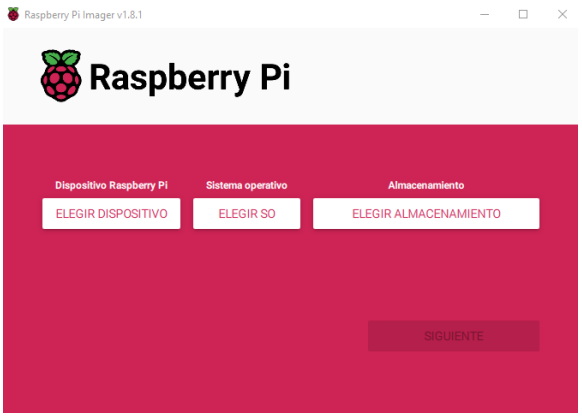


Figura 27. Software Raspberry Pi Imager v1.8.1

Fuente: Autoría propia.

3.1.2.2.Advanced IP Scanner

Es un escáner de red gratuito diseñado para escanear de forma rápida y fiable en redes locales. Esta herramienta le permite restaurar información sobre todos los dispositivos conectados a su red, brinda acceso a carpetas compartidas y admite la administración de recursos [70]. Será fundamental debido a que, una vez que conectemos nuestra Raspberry pi 4 a la red, el modem le asignará una ip única, que se requerirá para la configuración del mando remoto.

Figura 28. Software Advanced IP Scanner.

Fuente: [70].

3.1.2.3.VNC Viewer

Es una aplicación que forma parte del paquete de tecnología Virtual Network Computing (VNC). VNC es un sistema gráfico para compartir escritorio que utiliza el protocolo Remote Frame Buffer (RFB) para permitir el control remoto de una computadora desde otra ubicación [71]. En este caso de obtiene el control de la placa Raspberry pi 4 para realizar las diferentes configuraciones iniciales de la misma.

Figura 29. Software VNC Viewer.

Fuente: [71].

3.1.2.4.Python

Es un lenguaje de programación interpretado, cuyo mayor objetivo es la legibilidad. Este lenguaje es /multiparadigma ya que soporta programación orientada a objetos, imperativa y funcional [72].

Figura 30. Lenguaje de programación Python.

Fuente: [72].

3.1.2.5.Solidworks

El programa es una herramienta de diseño asistido por computadora en 3D que permite crear piezas y ensamblajes tridimensionales, así como planos bidimensionales. Proporciona una amplia gama de soluciones que abarcan todas las etapas del proceso de desarrollo de productos, desde la creación y el diseño hasta la simulación, fabricación, publicación y gestión de los datos relacionados con el proceso de diseño [73].

Figura 31. Software Solidworks.

Fuente: [73].

3.1.2.6. Google Colab

Colab es una plataforma en línea que ofrece acceso sencillo a Jupyter Notebook, sin necesidad de configuración previa. Además, brinda acceso gratuito a recursos informáticos avanzados, como GPU y TPU. Esta herramienta es especialmente útil para el aprendizaje automático, la ciencia de datos y la educación [74].

Figura 32. Google Colab.

Fuente: [74].

3.1.2.7.Open CV

Por sus siglas en inglés Open Computer Vision (Visión Artificial Abierta), es una librería de visión artificial de código abierto, enfocada en el tratamiento de

imagen, escrita en C y C++, es ejecutada en los sistemas operativos como Linux, Windows, Mac OS X, y tiene interfaces en Matlab, Java y Python [75].

Figura 33. Librería de Código abierto Open CV.

Fuente: [75].

3.1.2.8.Tensorflow

Es una biblioteca de código abierto desarrollada por Google para realizar cálculos numéricos y ejecutar procesos de aprendizaje automático de manera eficiente [76]. Con esta herramienta se entrena el modelo de aprendizaje para el sistema de visión artificial del presente proyecto.

Figura 34. Librería de código abierto Tensroflow.

Fuente: [76].

3.2. Diseño de la propuesta

Para el diseño del robot móvil terrestre se utilizó el software SolidWorks para el chasis y la protección de los elementos físicos de modo que no estén expuestos a factores ambientales que puedan ser contraproducentes para la implementación de la propuesta, en el siguiente apartado se detallan los elementos y su ubicación dentro de la carcasa del robot móvil.

3.2.1. Esquema de la propuesta

En la figura 35 se puede apreciar la parte interna de la carcasa del robot móvil en donde se encuentran todos los elementos electrónicos de la propuesta, los puentes H están ubicados en los extremos de la figura hexagonal que conforma la base del robot para que puedan estar más cerca de los servomotores y disminuir la distancia y el uso excesivo de cable además se deja un espacio para poder colocar la batería, que es la fuente de alimentación de todo el circuito tanto de control como de fuerza.

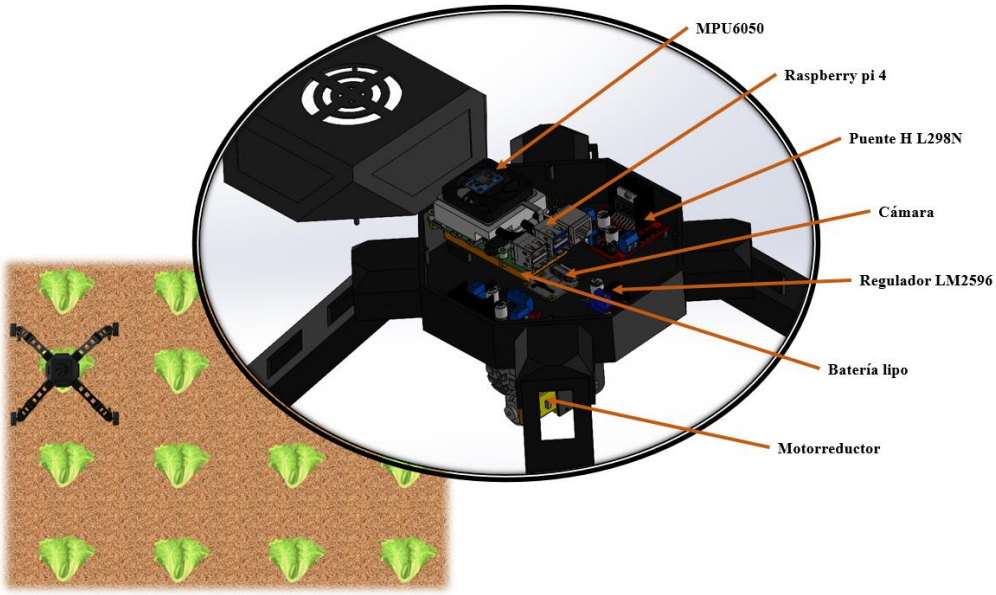


Figura 35. Esquema del rover automatizado.

Fuente: Autoría propia.

Sobre la batería estarán ubicados los elementos como la raspberry pi 4 modelo B y en su parte superior estará ubicado el sensor MPU6050 en la parte céntrica del modelo en 3D ya que será el encargado de los giros del robot y controlarlos por medio de los cálculos de los ángulos obtenidos por el mismo. A un costado se encuentra el regulador de voltaje DC – DC que administrara el voltaje de alimentación para la tarjeta controladora que en este caso es la raspberry pi 4, que contara con una alimentación de 3V, cada uno de los elementos fue medido previo a la fabricación de la carcasa del rover para poder ubicarlos de mejor manera y evitar fallos al momento de armar la propuesta.

3.2.2. Diagrama de flujo del algoritmo

En la figura 36 se muestra el diagrama de flujo se representa el proceso que realiza el rover automatizado iniciando con el cálculo de los valores del MPU6050 para realizar el giro controlado de toda la estructura, de esta manera el rover mantiene su posición durante 5 segundos con los motores apagados, luego comienza con la adquisición de imágenes por medio del modelo entrenado en Google colab.

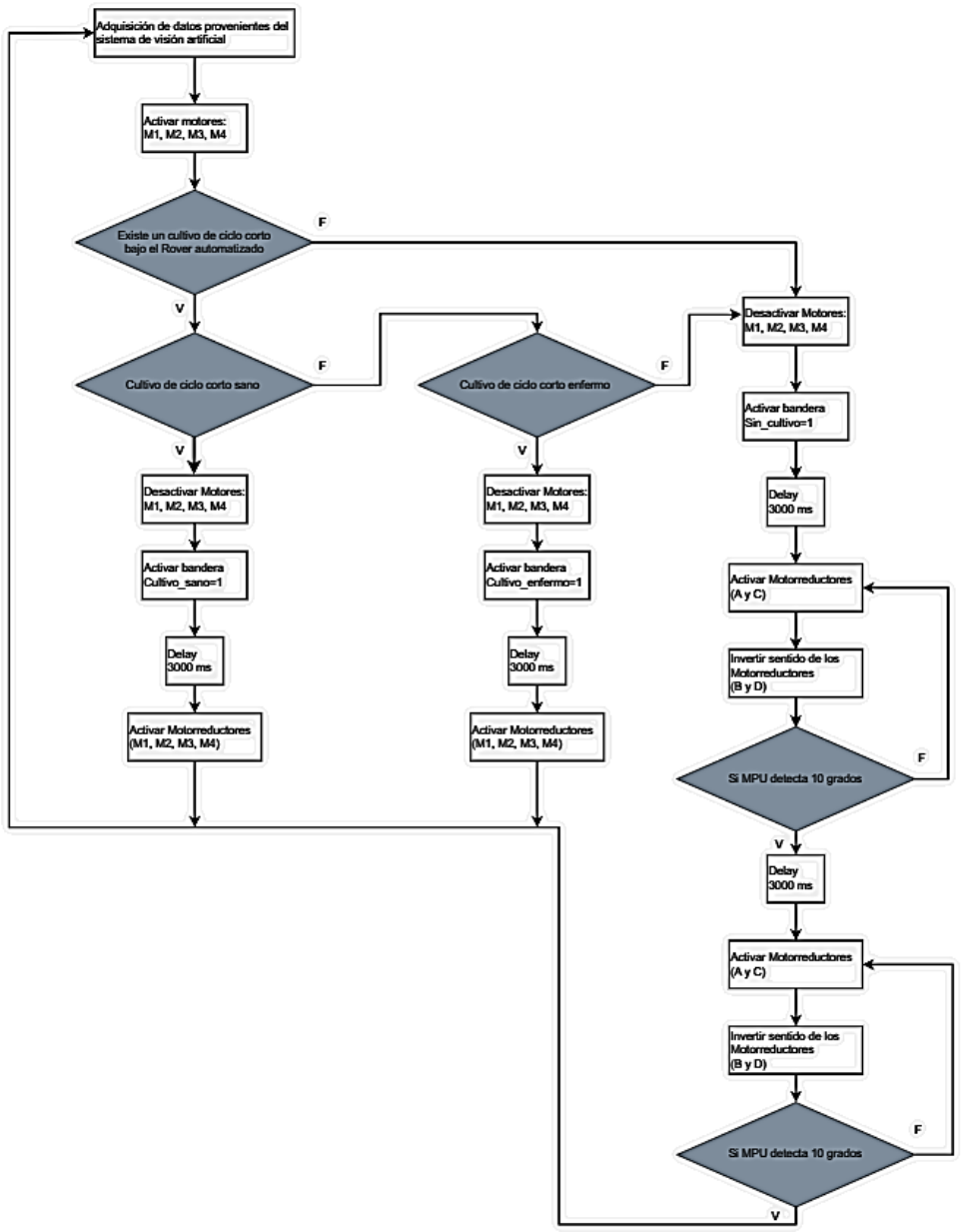


Figura 36. Diagrama de flujo de la propuesta.

Fuente: Autoría propia.

Como siguiente paso, el rover activará sus cuatro motores para avanzar en una dirección hasta que detecte una planta, de modo que cuando esté sobre el cultivo desactive los motores durante un tiempo de 3 segundos para que el sistema de visión artificial pueda identificar al cultivo y clasificarlo como un cultivo sano o un cultivo enfermo, dentro del diagrama de flujo se presentan estas condiciones tipo "IF", si el sistema detecta que el cultivo está sano activa una bandera llamada "Cultivo_sano", luego el rover seguirá avanzando hasta que detecte el siguiente cultivo, como son cultivos de ciclo corto, de manera general se siembran en parcelas, donde las plantas están divididas por surcos, una vez que el rover termine de detectar y clasificar la primera hilera de cultivos pasara a la siguiente, para esto el rover ya no debe detectar plantas al final de la hilera, de esta manera se activará una bandera llamada "Sin_cultivos", seguido de unas instrucciones que permitirán el giro del rover hacia la siguiente hilera de cultivos, para eso se activará los motores M1 y M2, mientras que, M3 y M4 invertirán su sentido de giro haciendo que el rover pueda girar en su propio eje, además, el MPU6050 realizara su cálculo para realizar el giro con precisión, para que los motores solo se desactiven cuando el MPU detecte que realizo el giro con los grados ingresados en el código, luego, el dispositivo avanzará durante 2 segundos y volverá a girar para entrar en la siguiente hilera de cultivos de ciclo corto, continuando con la detección y clasificación de los cultivos.

3.2.3. Modelado de piezas 3D para la estructura del rover

Se opto por el diseño de un rover con 4 extremidades alargadas y separadas del centro con el objetivo de que cuando realice el recorrido no afecte a ninguna planta y procese las imágenes con la cámara de manera efectiva.

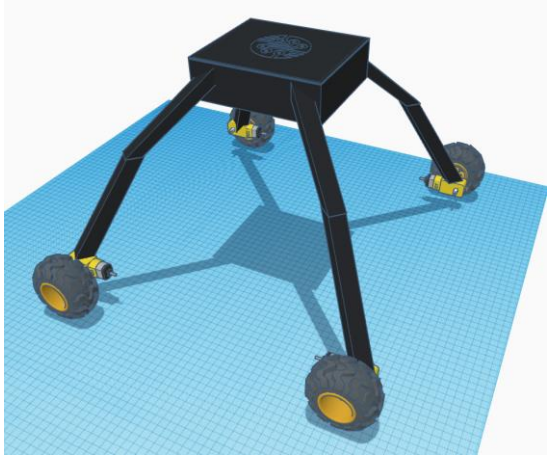


Figura 37. Diseño 3D de la estructura del rover.

Fuente: Autoría propia.

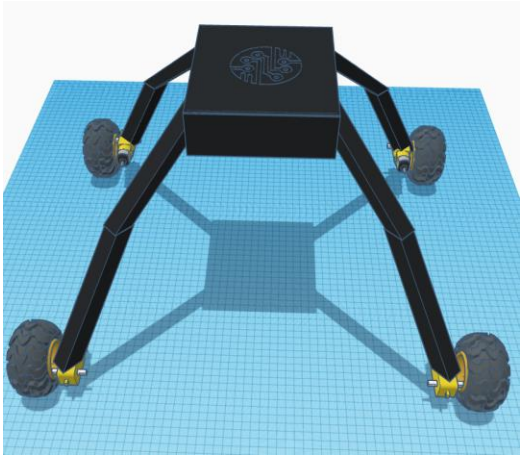


Figura 38. Vista frontal del diseño 3D del rover.

Fuente: Autoría propia.

Dentro de la estructura se observa la ubicación de cada uno de los elementos electrónicos que realizan el control del rover, en donde encontramos los dos Puente H la batería que está ubicada en el fonde de la estructura, y por último tenemos a la Raspberry Pi 4 que está en la parte superior de la batería, la ubicación de cada uno de estos dispositivos es para que puedan estar nivelados los pesos del rover y no presente irregularidades al momento de que los motores avancen.

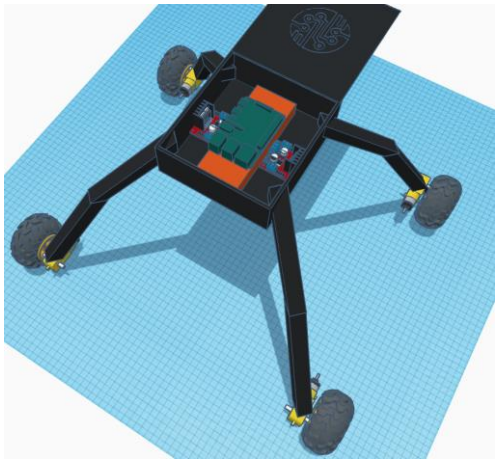


Figura 39. Vista interna de los elementos electrónicos descritos en la propuesta.

Fuente: Autoría propia.

Se realiza la estructura con un derivado de aluminio como es una plancha de aluminio compuesto, en donde encontramos sus extremidades y la caja central donde se resguarda la Raspberry pi 4 con los demás elementos de control, como se observa en figura 39, y dentro de la figura 40 se visualiza la estructura provisional para colocar los elementos de fuerza y lógica, para un refinamiento dentro del software Solidworks donde se realizan las piezas de manera individual para materializarlas con una impresora 3D .

Figura 40. Implementación de la estructura.

Fuente: Autoría propia.

3.2.3.1.Pieza 1

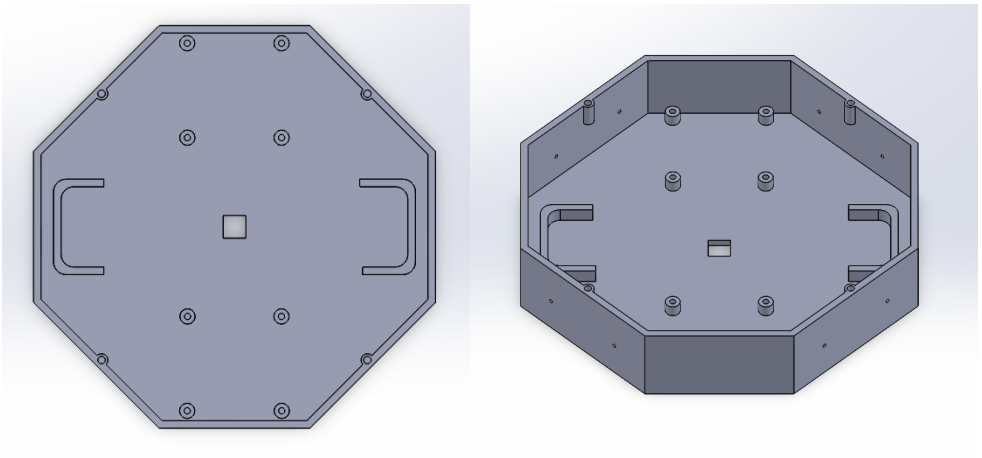


Figura 41. Base del rover.

Fuente: Autoría propia.

Esta parte es fundamental, ya que es donde se encontrarán los componentes físicos de la propuesta, contiene diferentes relieves en la base, como se observa en la figura 42, para que puedan colocarse los puentes H y un espacio cuadrado en medio de la base para que la cámara pueda detectar el cultivo, además en la parte lateral de esta pieza, en las cuatro paredes que se encuentran de manera perpendicular con respecto a la otra, se ubican orificios pequeños donde se fijará por medio de tornillos y tuercas la articulación qué hará que el rover eleve su posición con respecto al suelo, además de proporcionarle estabilidad y con la ayuda de los actuadores obtiene movimiento.

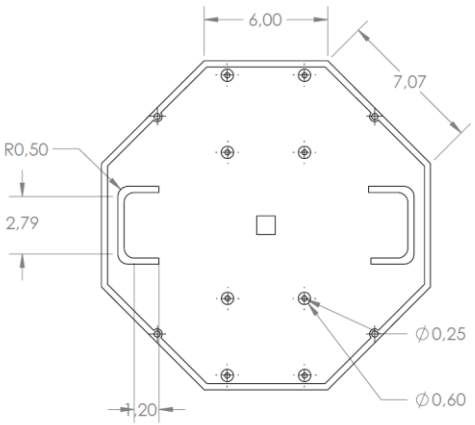


Figura 42. Planos con medidas de la pieza 1.

Fuente: Autoría propia.

En la figura 43 se visualizan las medidas más importantes de la pieza 1, ya que previo a la realización de las piezas se tomó medidas de cada uno de los elementos electrónicos de control y fuerza para ubicarlos de manera que no representen un problema para otro componente, además, de tomar en cuenta el tamaño que iba a ocupar todos los elementos, en el caso de esta primera pieza los que irán ubicados en la base de manera fija son los puente H, los cuales tienen 8 relieves en total para ambos, con un tamaño de 0.6 centímetros con un grosor de 0.35 centímetros para que pueda entrar el tornillo. Por otro lado, se observan a los lados unos relieves que tienen la forma de una "C" los cuales tienen como función elevar a la batería de manera estratégica y obtener espacio para la conexión de la cámara con la raspberry y que no sufra daño alguno debido al roce o al contacto directo.

Al momento de ubicar las mediciones de los elementos electrónicos se realizaron bocetos para definir la forma de la base del rover, de modo que se optó por un hexágono ya que aprovechaba de mejor manera todo el espacio y era conveniente también para la ubicación de los brazos del robot móvil terrestre que se implementan después del armado total de la pieza uno.

3.2.3.2.Pieza 2

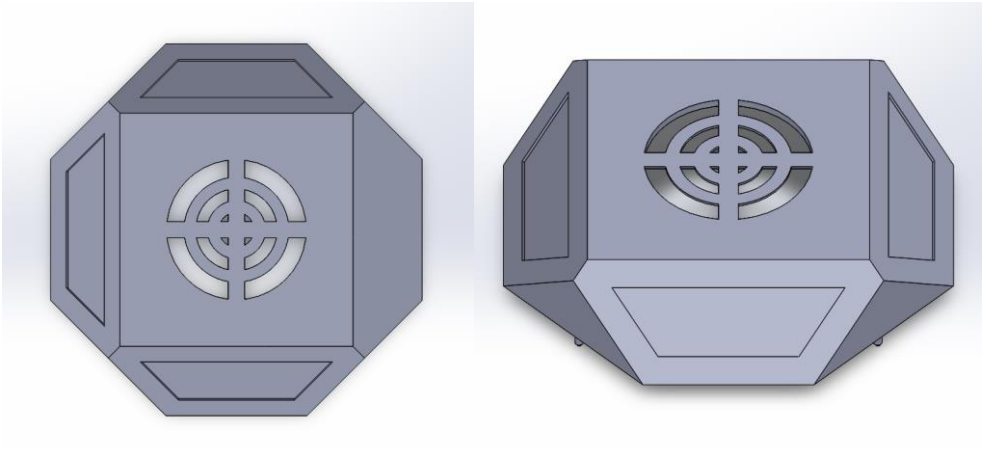


Figura 43. Cubierta de rover.

Fuente: Autoría propia.

Esta parte es la tapa de la base qué protegerá a los componentes actuador es y lógicos de la propuesta, realizado con las mismas medidas de la base para que encaje de manera uniforme, además, en medio de la parte superior de la pieza tiene la ventilación qué le permitirá mantener una temperatura óptima para que los

componentes funcionen dentro de manera más eficiente dentro de la carcasa cerrada como se observa en la figura 44.

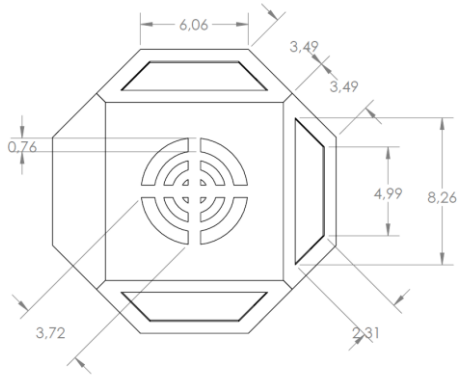


Figura 44. Planos con medidas de la pieza 2.

Fuente: Autoría propia.

Se realizaron modificaciones al diseño de la tapa para poder reducir la cantidad de filamento que se utiliza en la impresora 3D, esto se evidencia de mejor manera en la figura 45, en el lado derecho donde están ubicadas las mediciones de 4.99 y 8.26 centímetros los cuales forman un trapecio que se ubica en tres caras de la tapa.

3.2.3.3.Pieza 3

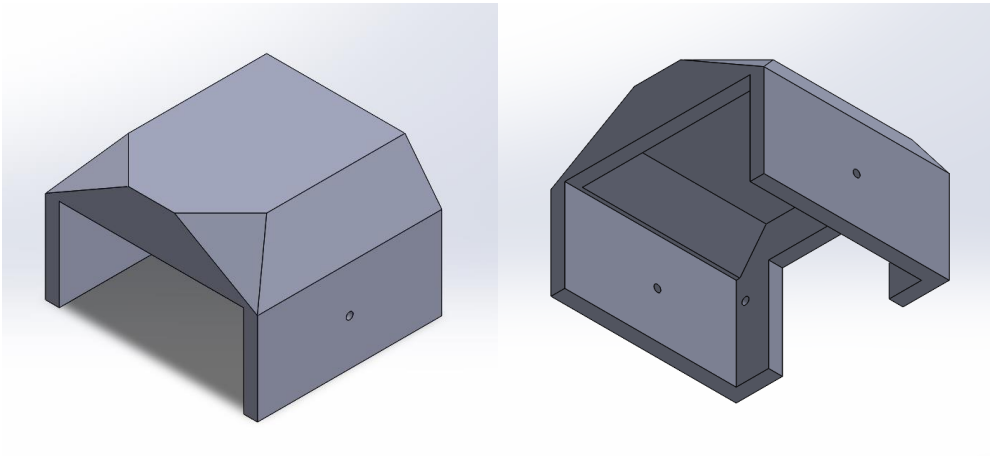


Figura 45. Primera pieza de brazo articulado.

Fuente: Autoría propia.

La siguiente pieza es la primera parte de la articulación para el rover, como se visualiza en la figura 46, que permite que la articulación pueda tener diferentes ángulos para controlar la altura del rover de manera manual, dependiendo de los requerimientos del usuario. Esta pieza tiene dos orificios para la sujeción de la

misma, con la base de la estructura y dos más en las partes laterales para sostener la siguiente pieza de la articulación.

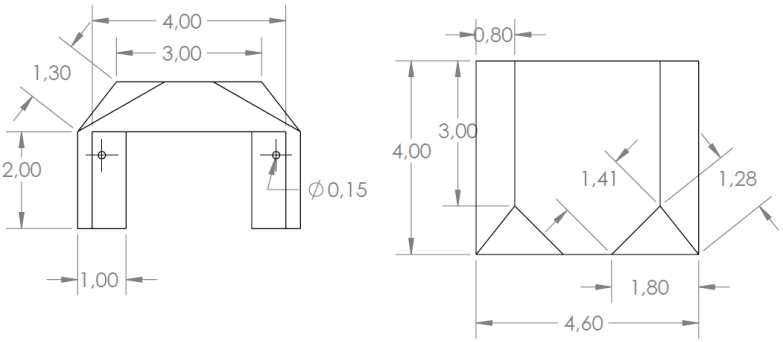


Figura 46. Planos con medidas de la pieza 3.

Fuente: Autoría propia.

Esta pieza se fija a las paredes laterales de la base o pieza 1 para que se direccionen de manera diagonal con respecto al eje de la base, estos orificios tienen un diámetro de 0.15 centímetros, en la figura 47, esto debido a que el tornillo debe entrar con un poco de presión para poder sujetar bien a la articulación que simulara un brazo, y si se mantiene esta comparativa esta pieza se denominaría el hombro de la articulación del robot móvil terrestre.

3.2.3.4.Pieza 4

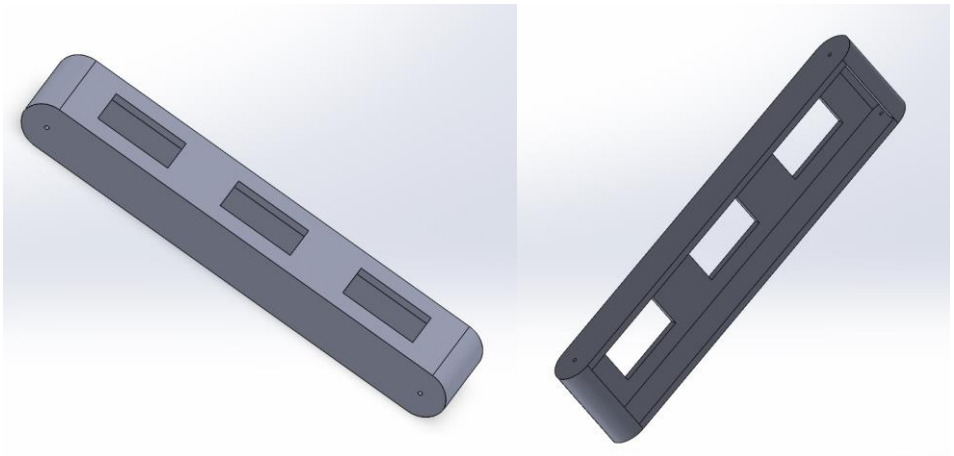


Figura 47. Segunda pieza de brazo articulado.

Fuente: Autoría propia.

Como se aprecia en la figura 48 la segunda pieza de la articulación tiene la función de elevar la estructura además de ampliar la distancia entre las demás articulaciones creando el espacio suficiente para que el cultivo de ciclo corto no

sufra daño cuando el rover realice su recorrido por el medio de las parcelas y los demás sembradíos.

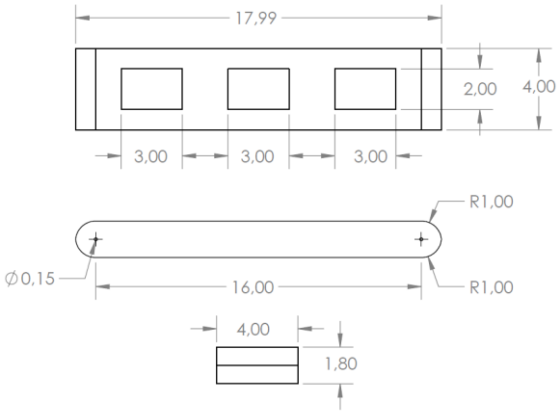


Figura 48. Planos con medidas de la pieza 4.

Fuente: Autoría propia.

Esta pieza cuenta con un largo de 17.99 centímetros y un ancho de 4 centímetros, además como se observa en la figura 49 se realizan cortes rectangulares a lo largo de la superficie más grande de la pieza, esto se debe a que por medio de la extracción de aquellas partes se reduce el peso de la pieza y a su vez disminuye la cantidad de filamento a utilizar, tomando en cuenta que estos cortes no representan un peligro para la función de soporte de esta pieza.

3.2.3.5.Pieza 5

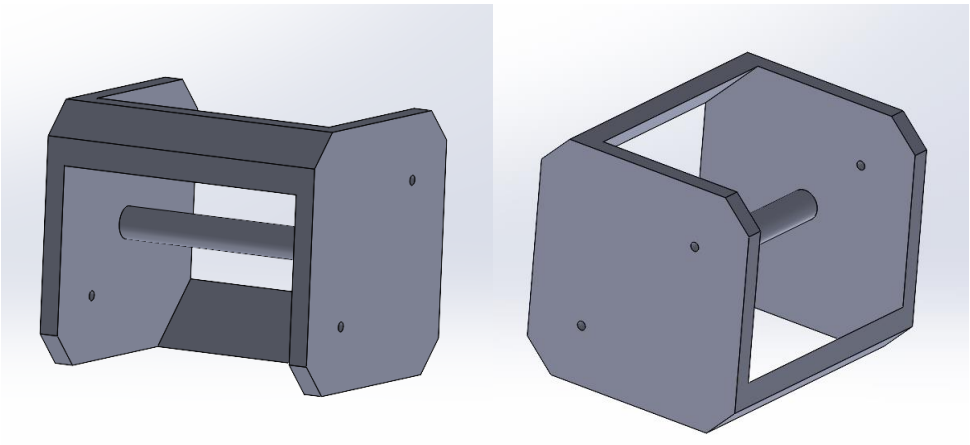


Figura 49. Tercera pieza de articulación, unión de piezas 4 y 6.

Fuente: Autoría propia.

En la figura 50 se encuentra la segunda articulación permite la unión de las piezas 4 y 6, aumentando la distancia que existe entre cada uno de los brazos

articulados, además de que en el otro extremo tiene un relieve de forma cilíndrica que sirve para fortalecer la pieza ya que en ella irán unidas dos piezas alargadas.

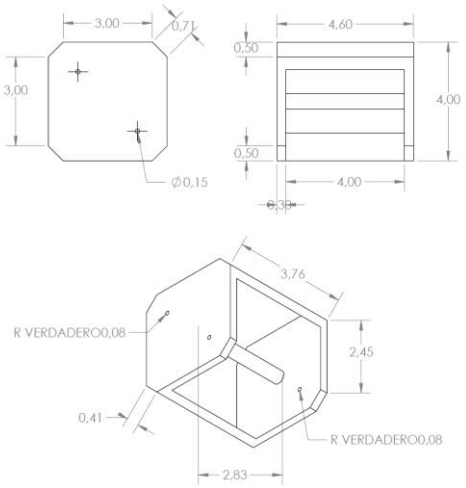


Figura 50. Planos con medidas de la pieza 5.

Fuente: Autoría propia.

Dentro de la figura 51 se encuentran las mediciones de todas las caras de la pieza 5, resaltando que debe ser mucho más ancha que las piezas 4 y 6 debido a que estarán dentro de ella de manera fija. Al realizar el diseño de esta pieza se tomó en cuenta que las lechugas tienen diferentes tamaños, entonces se optó por un diseño el cual pueda adaptarse a esas dimensiones, por eso esta pieza no es tan grande ni abarca todo el espacio disponible.

3.2.3.6.Pieza 6

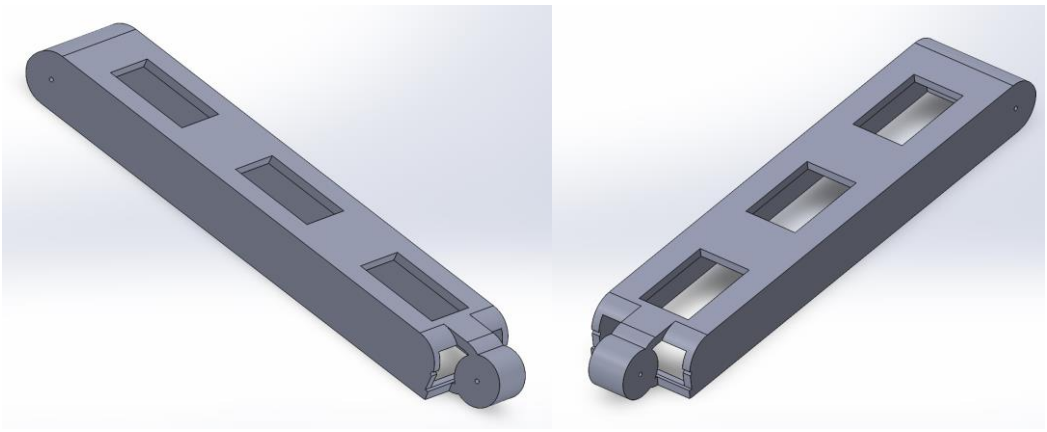


Figura 51. Cuarta pieza de articulación.

Fuente: Autoría propia.

La pieza mostrada en la figura 52 es similar a la pieza 4, sin embargo, esta en particular tiene un sobresaliente en forma de cilindro hueco, que permite la unión

de la pieza 7, otorgando la libertad de colocar en distintos ángulos conforme lo requiera la identificación que desea realizar el usuario.

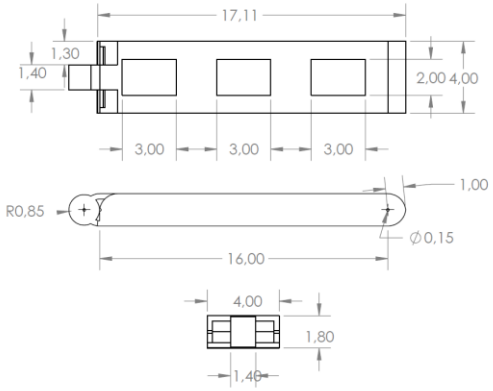


Figura 52. Planos con medidas de la pieza 6.

Fuente: Autoría propia.

Uno de los objetivos de esta pieza y su tamaño es que las plantas puedan mantenerse a salvo de ser dañadas y darle altura a la base para que la cámara pueda tener un mayor campo de captura al momento de procesar la imagen, esta sea más legible y se puedan reducir los errores al momento de identificar si la planta está enferma o si esta sana. Por otro lado, esta pieza es más grande que la pieza 4, en la figura 49 se logra visualizar que la medición de punto a punto de cada orificio de unión es de 16 centímetros, a diferencia de lo que se observa en la figura 53 donde las medidas difieren y tiene una distancia de 17.11 centímetros.

3.2.3.7.Pieza 7

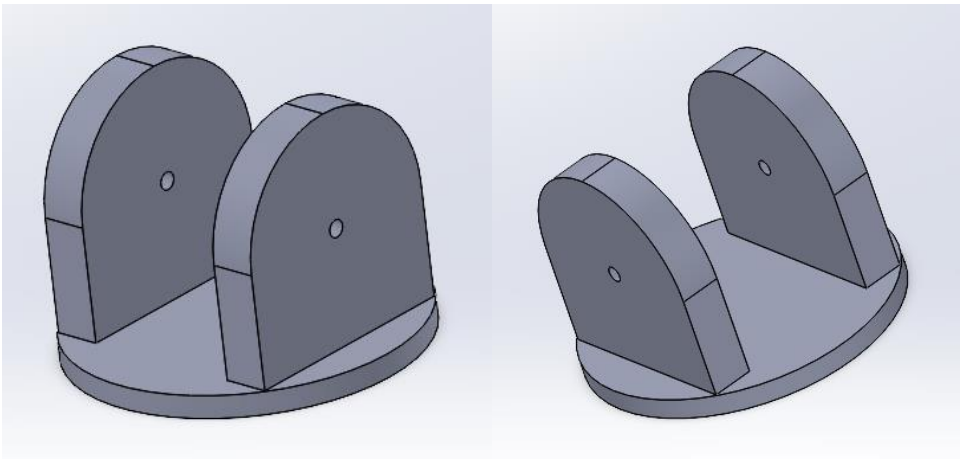


Figura 53. Quinta Pieza de articulación.

Fuente: Autoría propia.

Esta pequeña pieza permite la conexión entre el brazo articulado al motor reductor que le permitirá al rover automatizado moverse a través de todo el campo y sembríos. En la figura 54 se observa que la pieza 5 le permite ser graduada con respecto a los deseos del usuario tomando en cuenta el ángulo de inclinación de las demás piezas que componen a esta articulación.

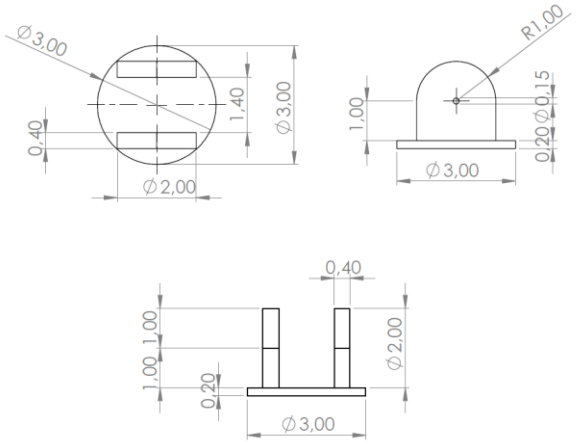


Figura 54. Planos con medidas de la pieza 7.

Fuente: Autoría propia.

Dentro de la figura 55 se encuentran las medidas de la pieza 5 donde se puede destacar que su base tiene un diámetro de 3 centímetros y tiene una altura de 2.2 centímetros que coinciden con las medidas de la pieza cilíndrica hueca de la pieza 6, además de los orificios para fijar las piezas con un diámetro de 0.15 centímetros.

3.2.3.8.Pieza 8

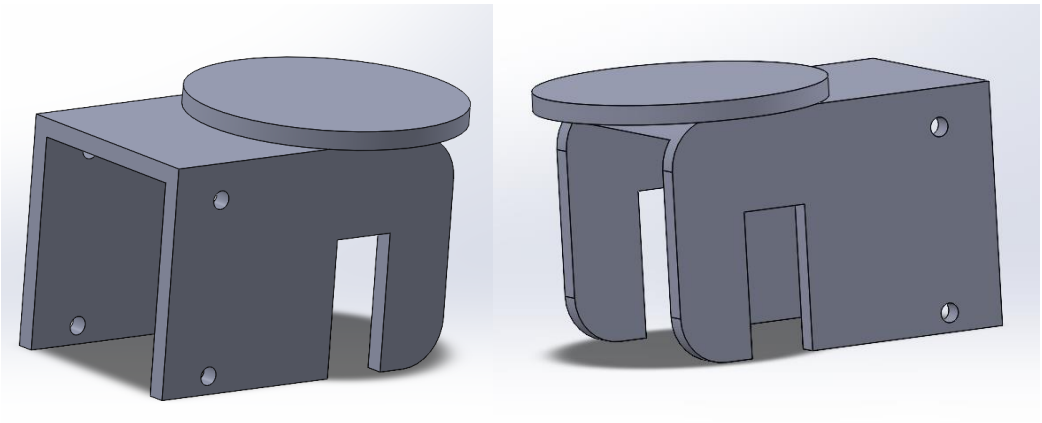


Figura 55. Sexta pieza de articulación.

Fuente: Autoría propia.

En la figura 56 se observa la pieza que permite la sujeción del motorreductor, para esto se tomaron las medidas respectivas al actuador para diseñar esta pieza y en la implementación no exista ningún tipo de imperfecto ya que la pieza una vez impresa si se le aplica mucha fuerza tiende a romperse.

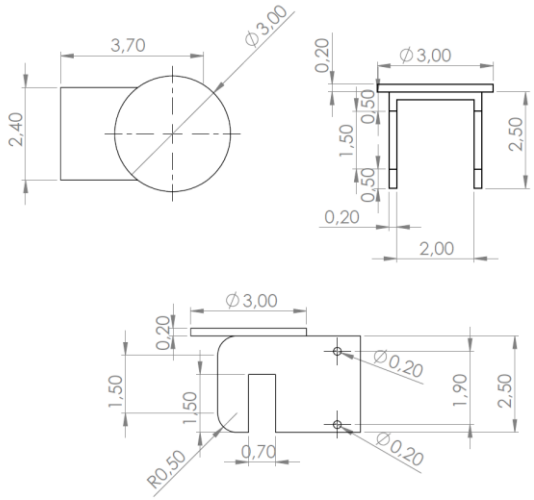


Figura 56. Planos con medidas de la pieza 8.

Fuente: Autoría propia.

3.2.3.9.Ensamble de piezas

Con ayuda del software de diseño 3D Solidworks realizamos la exportación de las piezas en un nuevo archivo llamado “Ensamble” donde colocaremos las piezas necesarias para la construcción de rover automatizado.

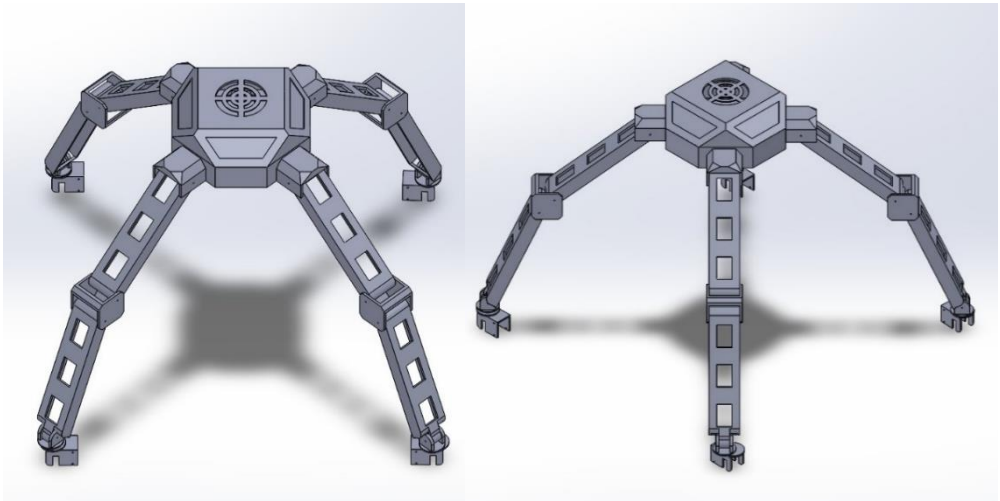


Figura 57. Ensamble de todas las piezas del rover.

Fuente: Autoría propia.

El software permite exportar varias veces una misma pieza, en este caso se utiliza esta opción en las piezas 3, 4, 5, 6, y 7, para obtener 4 brazos articulados y unirlos a la pieza 1 donde se encuentran los componentes de la propuesta. Para realizar la unión de estas piezas, el software permite crear relaciones entre las caras de las piezas que se requieran, esto permite que las piezas se unan y crean un límite dentro del espacio de simulación del entorno de Solidworks, limitando su desplazamiento, facilitando la unión de cada una de las piezas de la estructura como se puede apreciar en la Figura 58.

Figura 58. Vista superior del interior de la base del rover.

Fuente: Autoría propia.

En la figura 59 se observa las piezas del modelo de rover unidas, dando una vista previa de la estructura antes de imprimir las piezas por separado, este paso es importante ya que se al unir las piezas podemos observar si los orificios por donde pasarán lo tornillos uniendo las piezas coinciden.

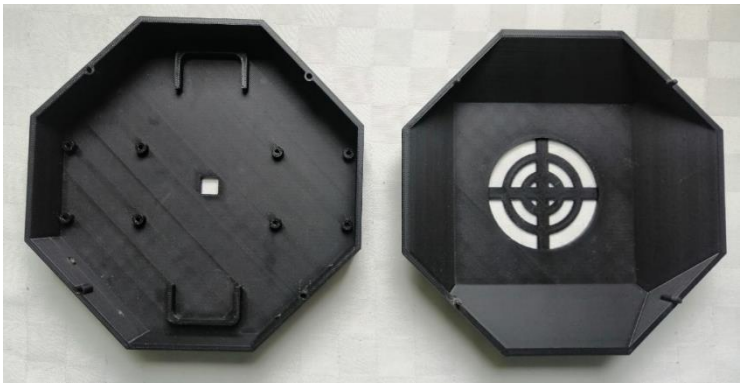


Figura 59. Impresión 3D de piezas 1 y 2.

Fuente: Autoría propia.

Una vez impresas las partes de la estructura se procede a quitar los excesos de material que la impresora 3D utiliza para realizar las piezas, en la Figura se observan las piezas 1 y 2 qué en la figura 60 se mostraban en el entorno del software Solidworks.

Se realizaron ajustes a la pieza de la tapa ya que las puntas sobresalientes qué se añadieron para que pueda acoplarse a la base eran muy gruesa y se procedió a desgarrar un poco dichos relieves para que pueda encajar de mejor manera.

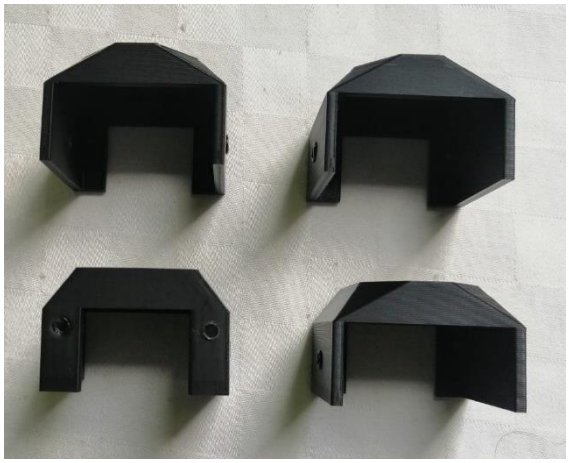


Figura 60. Impresiones 3D de la pieza 3.

Fuente: Autoría propia.

La pieza 3 se observa en la figura 61, donde las cuatro piezas se les retira el exceso de material, residuo de la impresión 3D, además, se verifica que los orificios tengan el tamaño óptimo para realizar las diferentes uniones de las partes articuladas.

Figura 61. Impresiones 3D de la pieza 4.

Fuente: Autoría propia.

En las impresiones de la pieza 4 se retira el exceso de material para reducir el peso de la estructura, esa es una de las razones por las que se realizó esta pieza con orificios.

Figura 62. Impresiones 3D de la pieza 5.

Fuente: Autoría propia.

Se verifica que las piezas de la figura 63 tengan sus uniones reforzadas, ya que esta pieza está en medio de las piezas más grandes de la estructura, por otro lado, es la encargada de sostener a las piezas mencionadas en la posición en la que el usuario la ajuste.

Figura 63. Impresiones 3D de la pieza 6.

Fuente: Autoría propia.

En la siguiente figura 65 se presenta a la pieza 6 impresa, tiene una similitud con la pieza 4 sin embargo, en uno de sus extremos tiene un relieve en forma de cilindro que le permite encajar con la siguiente pieza para formar la parte articulada del rover.

Figura 64. Impresiones 3D de la pieza 7.

Fuente: Autoría propia.

La pieza 7 que unen los soportes para los motores y el brazo articulado se muestran en la figura donde se aprecian 4 de estos ejemplares.

Figura 65. Impresiones 3D de la pieza 8.

Fuente: Autoría propia.

En la Figura 67 se puede visualizar a todas las piezas en 3D impresas, de modo que se obtiene un total de 26 piezas.

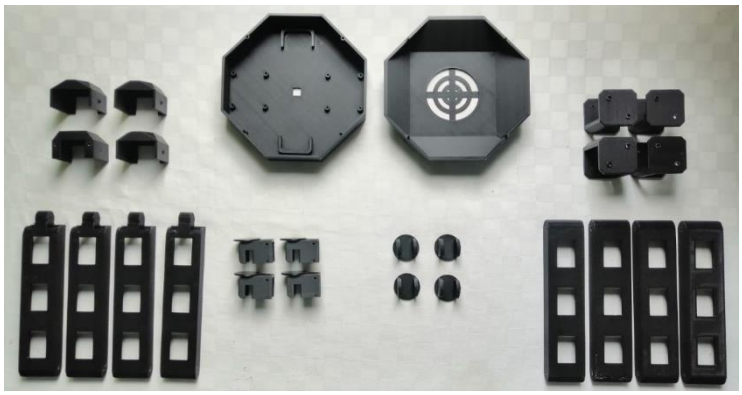


Figura 66. Vista general de todas las piezas impresas en 3D.

Fuente: Autoría propia.

Estas piezas tienen cortes circulares en algunas de sus superficies que permiten la unión por medio de tornillos y tuercas para su correcta sujeción entre cada una de ellas, además de proporcionarle más estabilidad al robot móvil y evitar falsos positivos al momento de realizar el respectivo procesamiento de la imagen capturada por la cámara del sistema.

Figura 67. Ensamble de las piezas impresas en 3D.

Fuente: Autoría propia.

Una vez ensambladas las piezas impresas en 3D se verifica los ángulos de las piezas que conforman la articulación para que el modelo sea estable como se puede observar en las figuras 68 y 69, en donde el modelo 3D tiene instalados los actuadores, que en este caso son los cuatro servomotores con sus respectivas ruedas.

Figura 68. Vista superior del rover ensamblado.

Fuente: Autoría propia.

3.3.Configuración del sistema de la propuesta

Instalación de Picamera

Para la instalación de esta librería indispensable para la propuesta se necesita el siguiente comando con permisos de administrador “sudo apt – get install python3-picamera”, luego de ingresar el comando se comenzará a descargar todos los archivos y a crear las dependencias necesarias.

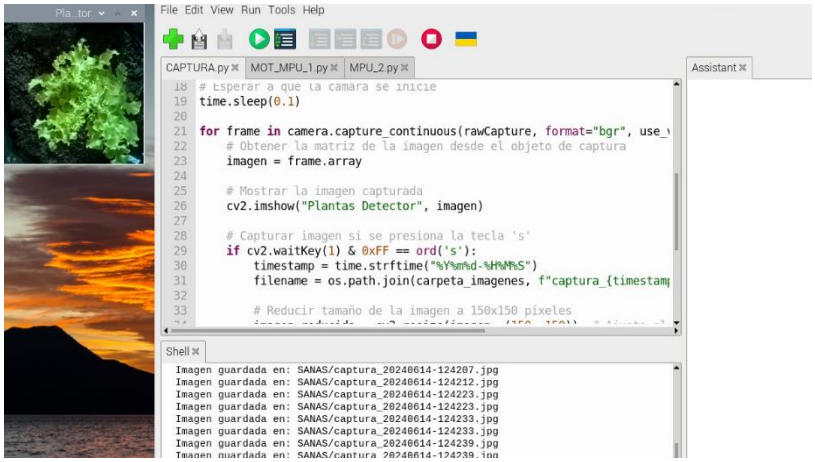


Figura 69. Captura de imágenes para el entrenamiento del modelo.

Fuente: Autoría propia.

Posteriormente se realiza un código para la captura de imágenes, con las que se va a entrenar el modelo, para luego subirlas a Google drive, en donde se trabajara con Colab para importar estas imágenes a su entorno de trabajo y procesar las imágenes capturadas, redimensionándolas para un óptimo proceso como se observa en la figura 71 donde se realizan las capturas de imágenes.

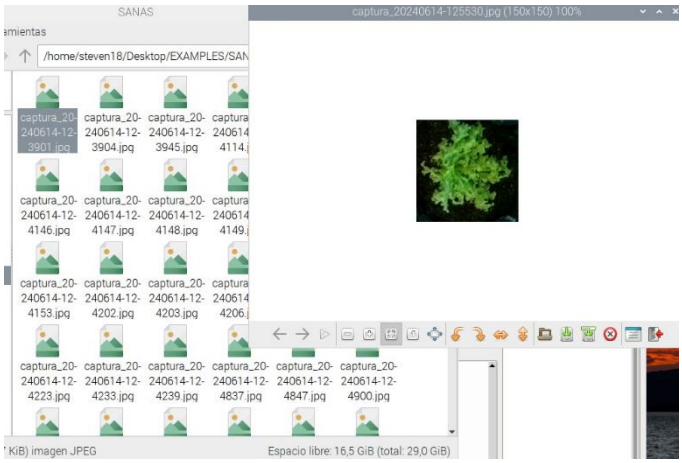


Figura 70. Almacenamiento de las imágenes tomadas por la cámara de la raspberry pi 4.

Fuente: Autoría propia.

Dentro de la figura 72 se observa que las imágenes se almacenan dentro de una carpeta en el escritorio de la raspberry pi 4, sin embargo, una vez tomadas y subidas al entorno de Google colab, no son necesarias dentro de la tarjeta, de modo que se procede a borrarlas para que la raspberry pi 4 tenga más almacenamiento. El código utilizado para la captura de imágenes y su redimensionamiento se encuentra en el Anexo H.

3.3.1. Configuración de la raspberry pi 4

Para realizar la configuración de la tarjeta controladora primero se debe realizar el entrenamiento del modelo con Google colab, esta acción se realiza en un ordenador aparte, esto es para que la raspberry no tenga tanta carga además de disminuir el tiempo de entrenamiento, porque la raspberry tiene capacidades limitadas y para optimizar el proceso de entrenamiento se realiza el entrenamiento en un dispositivo con mayores capacidades, para posteriormente exportar dicho modelo a la raspberry y usarlo en el código para la identificación de las plantas.

3.3.2. Entrenamiento de modelo de perceptrón multicapa en Google Colab.

Se importan las librerías que utilizaremos para el proyecto en la primera parte del código en donde se realizara el preprocesamiento de las imágenes, se importan las librerías que se visualizan en la figura. En el segundo bloque de código tenemos dos variables declaradas en donde se almacenarán las imágenes y categorías respectivas de la planta sana y enferma.

Luego ejecutamos el siguiente bloque de código para subir los archivos en donde se encuentran las carpetas con las imágenes de cada planta, además de que con el comando “files.upload()” se despliega una ventana que permite seleccionar el documento con extensión ZIP donde están los archivos necesarios para el entrenamiento de la neurona.

En la figura 75 se muestran las carpetas “ENFERMAS” y “SANAS”, las misma están comprimidas en el archivo ZIP que se encuentra en la parte inferior.

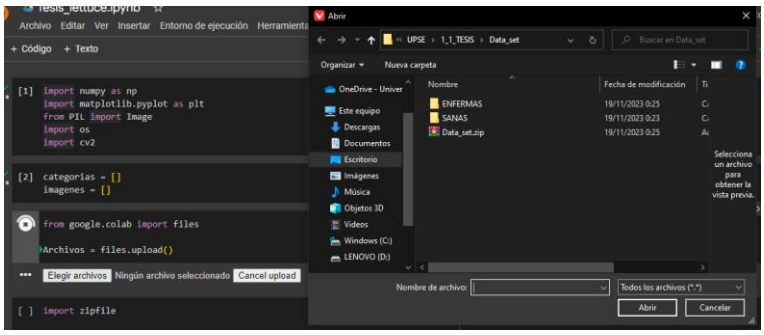


Figura 71. Selección de carpetas para el entrenamiento de neurona.

Fuente: Autoría propia.

Una vez finalizado el proceso de importar el archivo, verificamos en las carpetas de Google colab, dentro del directorio de “content”, como se muestra en la figura.

Luego de importar las carpetas con las imágenes necesarias, se procede a descomprimir el archivo ZIP para obtener las carpetas que contiene, que en este caso son las carpetas “SANAS” y “ENFERMAS”. Para esto importamos una librería llamada “zipfile”, que permite extraer las carpetas de un archivo con extensión ZIP.

Declaramos la variable “lechuga_zip” donde se descomprime el archivo ZIP, y se extrae el directorio en la variable “directorio_extract”. Posteriormente dentro de la variable “categorias” se almacena una lista que contiene los nombres de todas las carpetas dentro del directorio extraído, para eso se utiliza los módulos “os” para filtrar solamente a los directorios. Cuando se finalice este proceso debemos verificar que se extrajeron las carpetas requeridas para eso con el comando “print” mostramos las carpetas existentes en el directorio.

Al momento de descomprimir todas las carpetas de archivo ZIP, empezamos a procesar las imágenes que contiene cada una de ellas por separado, empezando con la carpeta “SANAS”. Iniciamos definiendo la ruta de esta carpeta además de definir el tamaño de imagen, que en este caso es de 150. Se utiliza un ciclo “for” para poder redimensionar todas las imágenes extraídas y finalmente mostrar los elementos procesados.

Se verifica que todos los archivos se extrajeron de manera correcta utilizando las siguientes líneas de código, que permiten observar el número de imágenes, el tamaño de la imagen y el canal de colores.

Graficamos una de las imágenes que se encuentra dentro de la carpeta “SANAS” para visualizar la imagen con el canal de colores escogido como se muestra en la figura.

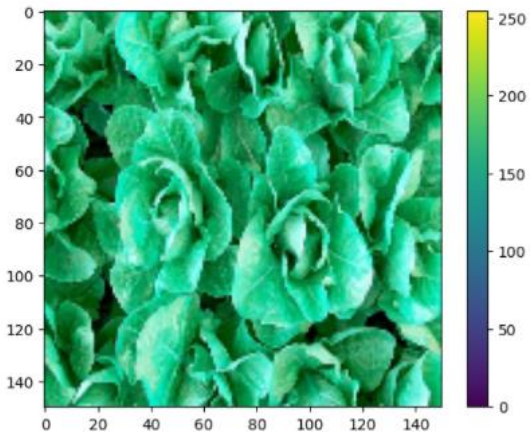


Figura 72. Gráfica de una de las figuras almacenadas en la carpeta “SANAS” con el canal de color escogido.

Fuente: Autoría propia.

Del mismo modo se realiza el proceso para el procesamiento de las imágenes de la carpeta “ENFERMAS” donde el resultado de ejecución del código se observa dentro la figura 74.

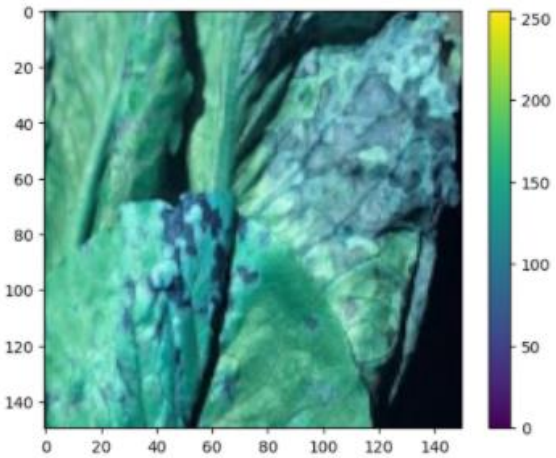


Figura 73. Gráfica de una de las figuras almacenadas en la carpeta “SANAS” con el canal de color escogido.

Fuente: Autoría propia.

Al finalizar el procesamiento de ambas imágenes se juntan las dos carpetas y con ayuda del comando “np.concatenate” unimos todas las imágenes para tenerlas en un solo directorio , utilizamos el comando “print” para poder verificar cuántas imágenes tenemos, y posteriormente almacenar la lista de imágenes en un arreglo.

En el siguiente código lo que se realiza es el etiquetado de las imágenes de la carpeta “SANAS” para que puedan identificarse como tal, se utiliza el comando “np.repeat” para poder asignarle un valor a esta etiqueta que en este caso es cero, además de agregarle como segundo argumento el número de imágenes con el cual estamos trabajando.

De la misma manera realizamos las etiquetas para la carpeta “ENFERMAS”, utilizando el mismo comando sin embargo para poder identificar a estas imágenes le asignaremos uno, más el número de imágenes que contiene la carpeta.

Se ejecuta en la siguiente línea de código para poder definir las clases que tenemos que en este caso son las plantas sanas con el número cero y las plantas enfermas con el número uno.

Posteriormente con el comando “np.concatenate” juntamos todas las etiquetas, y almacenamos la lista en un arreglo para al ejecutar la línea de código poder observar cuántas etiquetas tenemos con el valor que se le asignó a cada una.

Una vez realizadas las etiquetas para cada una de las carpetas se procede a graficar las imágenes junto con las etiquetas correspondientes mediante la creación de una nueva figura con “matplotlib” con un tamaño de 10 x 10 HP pulgadas, donde se utiliza un ciclo “for” para iniciar un bucle que se itera 50 veces y gráfica 10 filas y 5 columnas para que se puedan observar la cuadrícula de las imágenes junto con la etiqueta en la parte inferior de cada una de ellas.

Como se puede observar en la figura 75, cuadrícula de las imágenes con su respectiva etiqueta, separando así las plantas sanas de las plantas enfermas.

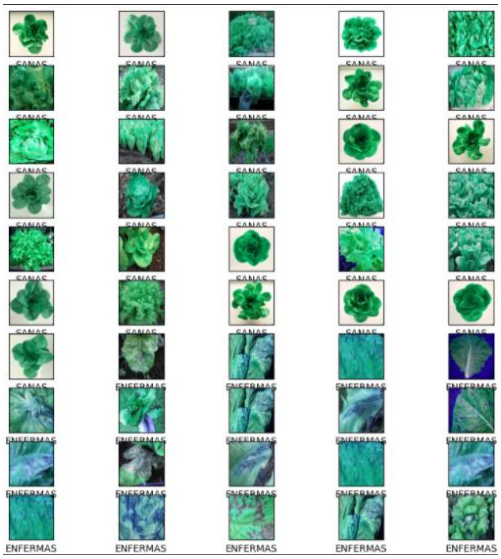


Figura 74. Gráfica de clasificación de imágenes de plantas enfermas y sanas.

Fuente: Autoría propia.

Luego inicializamos las librerías de "Tensorflow" para poder realizar el entrenamiento de la neurona, con ayuda de los comandos del Anexo D.

Como siguiente paso tenemos el entrenamiento de la neurona y la creación de nuestro modelo, para el cual utilizamos "keras.Sequential", el cual define un modelo de red neuronal secuencial, es decir, que las capas van a apilarse una encima de la otra en secuencia, la capa de aplanado transforma la entrada tridimensional que en este caso serían las imágenes de 150x150 pixeles con 3 canales de color en un vector unidimensional antes de que pase a las capas que están completamente conectadas.

Con el comando "keras.layers.Dense(128, activation='relu')" pasa a la capa completamente conectada con 128 neuronas y con una función de activación relu, posteriormente pasa a la capa de salida que contiene dos neuronas ya que va a realizar una tarea de clasificación binaria.

Pasamos a la compilación del modelo donde se realizan las configuraciones y se declaran algunos argumentos cómo, la optimización Adam que es un algoritmo de optimización popular, la función de pérdida de optimización, y por último argumento, se tiene a la métrica que va a seguir durante el entrenamiento comúnmente se la conoce como la precisión o "accuracy".

Luego se entrena al modelo con los datos de entrada y las etiquetas correspondientes, además de especificar el número de épocas para que el modelo pueda entrenar, en este caso se asignan 100 épocas.

Para realizar pruebas y compara su efectividad al momento de detectar si una planta esta sana o enferma, se procede a entrenar dos modelos más, con la diferencia de que con cada modelo aumentaran el número de muestras y épocas de entrenamiento, para que así se pueda evidenciar la mejora del modelo conforme más datos reciba y más tiempo tarde en entrenar. El código utilizado para el preprocesamiento de las imágenes se encuentra en los Anexos I, para cada uno se tiene las mismas clases, pero con más muestras para cada una de ellas.

De la misma manera se concatenan todas las etiquetas en un solo arreglo, donde en total indica que tenemos 204 muestras para el entrenamiento del otro modelo para el sistema de visión artificial por perceptrón multicapa.

Luego procedemos a realizar la clasificación de las imágenes que se almacenaron en el arreglo para poder verificar si el etiquetado esta realizado de la manera correcta, como se evidencio en el anterior modelo entrenado con 50 muestras. A diferencia del otro código en este caso se modificaron los parámetros para que se pueda visualizar la mayor parte de imágenes del modelo ya que ahora son más de 100 muestras, de modo que se redimensiono las imágenes para graficarlas como se ve en la figura 76.

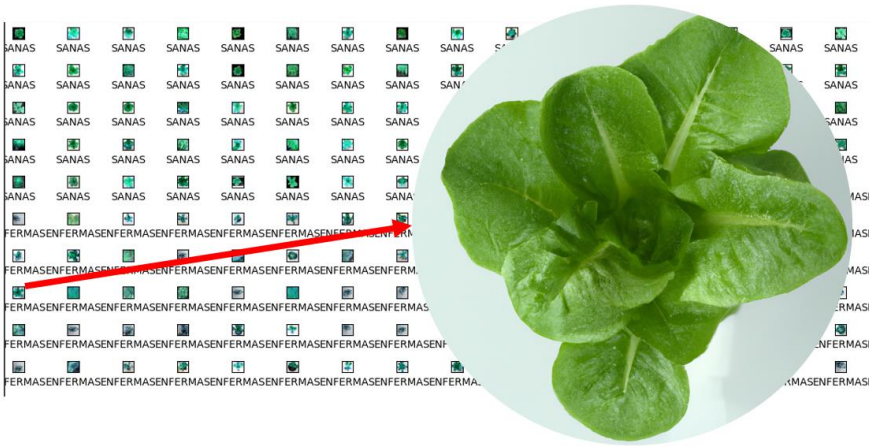


Figura 75. Gráfica de clasificación de imágenes de plantas enfermas y sanas del nuevo modelo para el sistema de visión artificial.

Fuente: Autoría propia.

Y como último paso se realiza el entrenamiento del respectivo modelo con 204 muestras y con 200 épocas de manera que el modelo pueda garantizar un equilibrio entre el ajuste preciso de los datos de entrenamiento y la capacidad de generación del modelo a nuevos datos. (Revisar Anexo J)

3.3.3. Entrenamiento de la red neuronal convolucional en Google Colab.

Para mejorar la detección de las plantas sanas y enfermas se realiza un último entrenamiento, pero esta vez se realiza con 1347 imágenes, y se aumenta una clase más al entrenamiento, denominada "SIN_PLANTA", dentro de esta clase irán las imágenes del suelo donde se plantan los cultivos, y pueda detectar de mejor manera que en ese espacio no hay planta sana o una planta enferma y así reducir aun mas el error de que detecte planta enferma, mientras la cámara apunta al suelo.

Para esto se realizó un nuevo script para entrenar el modelo de CNN (Convolutional Neuronal Network), sin embargo, se siguen utilizando los códigos del Anexo I, para el preprocesamiento de las imágenes además del ajuste en su tamaño, antes de este último entrenamiento.

En el Anexo K se encuentra el código para el entrenamiento del modelo para la red neuronal convolucional, en donde se tienen las librerías de "Tensorflow" para construir y entrenar al modelo y "train_test_split" de "sklearn" para dividir los datos en conjuntos de entrenamiento y validación. Como se mencionó que se utilizarían los códigos para el preprocesamiento de imágenes usados para los otros modelos, una vez terminado ese proceso, estas imágenes se combinan en un solo "array" llamado "images" y se crean las etiquetas correspondientes para cada clase, como son "PLANTA_SANA", "PLANTA_ENFERMA" y "SIN_PLANTA", 0 para sanas, 1 para enfermas, 2 para sin planta y estos se almacenan en el "array" llamado "labels".

Posteriormente se utiliza la línea de código "train_images, val_images, train_labels, val_labels = train_test_split(Images, labels, test_size=0.2, stratify=labels, random_state=42)" para la división de datos en conjuntos de entrenamiento y validación, ya que es una etapa crucial en la construcción de modelos de "machine learning", donde el 80% se utiliza para el entrenamiento y el

20 % para la validación, además con “stratify=labels” se asegura que la proporción de cada clase sea la misma en ambos conjuntos.

Luego con “ImageDataGenerator” se aumenta los datos de entrenamiento mediante transformaciones aleatorias, es decir, con rotaciones, desplazamientos, cambios de zoom y volteos horizontales, esto ayuda a mejorar el entrenamiento del modelo.

En la siguiente parte del código se define a la red neuronal convolucional utilizando “Sequential”, este modelo incluye varias capas convolucionales “Conv2D” que se encargan de extraer características de las imágenes, y las capas pooling “MaxPooling2D” reducen la dimensionalidad de las características extraídas. Después de estas capas, se aplanan los datos con “Flatten” y se pasan por capas densas “Dense”, en la última capa se encuentran 3 unidades con activación “softmax” para producir una probabilidad para cada una de las tres clases establecidad en esta propuesta.

El modelo se compila utilizando el optimizador “adam” y la función “sparse_categorical_crossentropy”, que es adecuada para los problemas de clasificación con etiquetas enteras, además se monitorea la precisión con “accuracy” como métrica de rendimiento.

Se realiza el entrenamiento del modelo utilizando los generadores de datos de entrenamiento y validación, además el entrenamiento se realiza durante 50 épocas y se ajusta el modelo en cada época utilizando los datos aumentados mencionados.

Finalmente, el modelo se evalúa en el conjunto de validación y se imprime la precisión de validación.

3.2.3. Respaldo de los modelos entrenados en Google Colab

Una vez que se realizó el entrenamiento respectivo de todos los modelos a utilizar en esta propuesta se descargan y exportan a la tarjeta Raspberry Pi 4 para iniciar con el proceso de pruebas, para esto se utiliza Google Colab para guardar los modelos en el ordenador se usa el comando “model.save()”, lo cual guarda el

modelo dentro del ambiente de Google Colab, como se observa en los anexos.(Revisar Anexo L)

3.2.4. Conversión de modelos entrenados en Tensorflow a Tflite

Debido a que en la tarjeta Raspberry Pi 4 no está disponible la versión completa de TensoFlow ya que no existe una compatibilidad en ninguna de sus versiones, se instala tflite que es una extensión de la librería TensorFlow y que cumple con los requerimientos de esta propuesta, sin embargo, dicha extensión no trabaja con los modelos ".h5", sino con modelos "tflite". (Revisar Anexo M)

3.2.5. Configuración de pines de la Raspberry Pi 4

Dentro de la Raspberry Pi 4 existen diferentes pines con objetivos diferentes, como se observa en la figura 77, sin embargo, en esta propuesta solo se utilizan los pines con PWM para poder controlar la velocidad de los actuadores como son los cuatro motorreductores, para esto se conectan los pines GPIO correspondientes, como son GPIO 18, 19, 20 y 21. (Ver Figura 88)

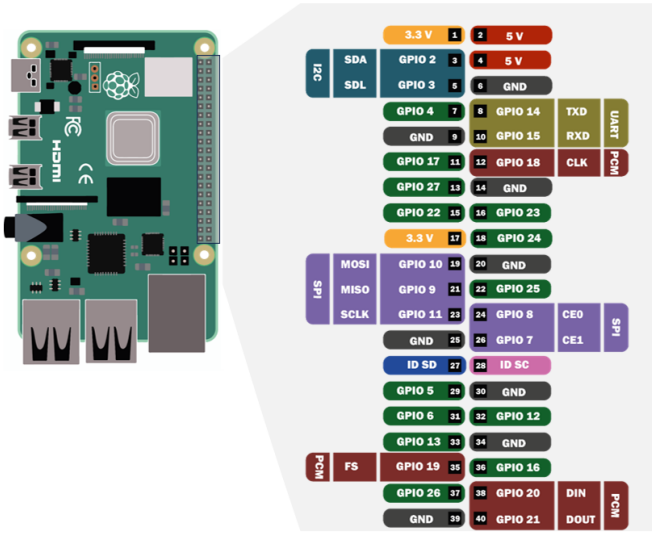


Figura 76. Pines de la Raspberry pi 4.

Fuente: Autoría propia.

Para la conexión de cada uno de los componentes de la propuesta, se realizaron en primer lugar las conexiones de alimentación, y el dimensionamiento correspondiente para cada uno de los actuadores, luego se conectaron la alimentación que en este caso es la batería Lipo de 7.4V a 3.3A a un variador de voltaje para energizar a la Raspberry Pi 4 con 5V, los fabricantes de Raspberry pi recomiendan usar la tarjeta con una fuente que supere los 3 A para un funcionamiento optimo del proceso a ejecutar, debido a que la tarjeta no tiene pines de alimentación, se adapta un cable con terminal tipo C para energizar a la tarjeta.

Posteriormente una vez realizadas las conexiones VCC y GND, se procede a conectar los pines de control de los puentes H L298N a la tarjeta Raspberry Pi 4, como se mencionó, para controlar el parámetro de velocidad se debe utilizar pines PWM para la habilitación de los motorreductores. (Ver Figura 78 a 80)

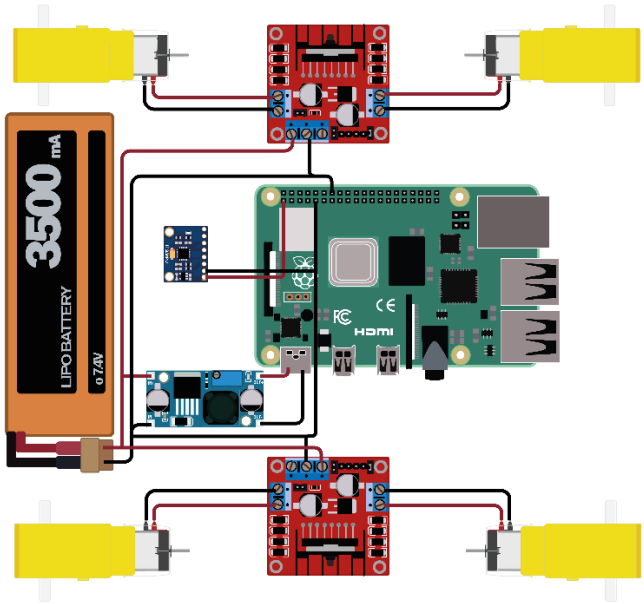


Figura 77. Conexiones VCC y GND de la propuesta.

Fuente: Autoría propia.

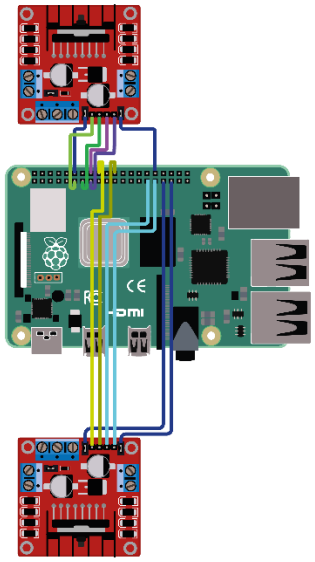


Figura 78. Conexión de pines habilitadores y direccionamiento de los motores.

Fuente: Autoría propia.

Estas conexiones son importantes para definir cuáles de los cuatro motores se les asignara los nombres de A, B, C y D, para en el momento de pruebas de movimiento se puedan invertir correctamente el sentido de dirección y girar el rover.

Tabla 8. Pines GPIO a utilizar para el control de los motorreductores 1 y 2.

Motorreductor	**Pines GPIO Raspberry Pi 4**	**Puente H L298N 1**	**Descripción**
Motorreductor 1	GPIO 18	ENA	Pin para habilitar el control de controlador del Motorreductor 1
	GPIO 23	IN1	Pines para el sentido de giro del Motorreductor 1
	GPIO 24	IN2	
Motorreductor 2	GPIO 20	ENB	Pin para habilitar el control de controlador del Motorreductor 2
	GPIO 25	IN3	Pines para el sentido de giro del Motorreductor 2
	GPIO 12	IN4	

Tabla 9. Pines GPIO a utilizar para el control de los motorreductores 3 y 4.

Motorreductor	Pines GPIO Raspberry Pi 4	Puente H L298N 2	Descripción
Motorreductor 3	GPIO 19	ENA	Pin para habilitar el control de controlador del Motorreductor 3
	GPIO 4	IN1	Pines para el sentido de giro del Motorreductor 3
	GPIO 17	IN2	
Motorreductor 4	GPIO 21	ENB	Pin para habilitar el control de controlador del Motorreductor 4
	GPIO 27	IN3	Pines para el sentido de giro del Motorreductor 4
	GPIO 22	IN4	

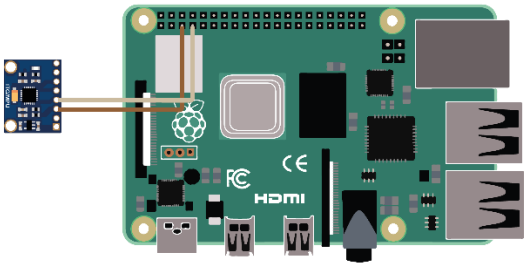


Figura 79. Conexión de pines SCL y SDA.

Fuente: Autoría propia.

Tabla 10. Pines GPIO a utilizar para el control de giro del rover.

Pines GPIO Raspberry Pi 4	MPU6050	Descripción
GPIO 2	SDA	Permite que el maestro como el esclavo puedan enviar datos.
GPIO 3	SCL	El maestro genera la señal de reloj en la línea SCL

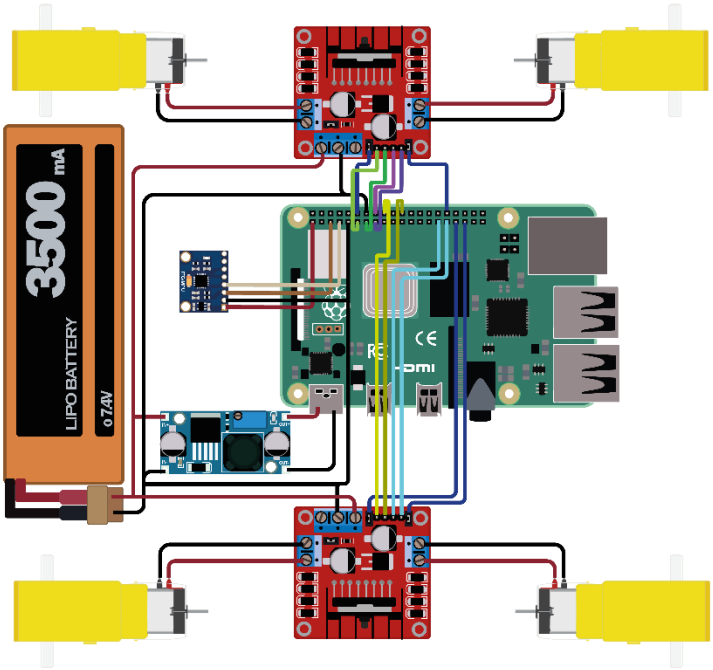


Figura 80. Vista general de todas las conexiones de la propuesta.

Fuente: Autoría propia.

Implementación de los modelos entrenados en la raspberry pi 4

El código realizado en el Anexo N implementa un sistema de control para un robot móvil que utiliza una Raspberry Pi. Primero, se importa las bibliotecas necesarias: “RPi.GPIO” para controlar los pines GPIO de Raspberry Pi, luego se usa “time” para funciones relacionadas con el tiempo, “smbus” para comunicación con dispositivos I2C, “numpy” para operaciones con matrices y “arrays”, “tflite_runtime.interpreter” para realizar inferencias con “TensorFlow Modelos Lite”, picamera para capturar imágenes con la cámara Raspberry Pi y “cv2” para procesar imágenes usando “OpenCV”. A continuación, se configura los pines GPIO para controlar cuatro motores utilizando PWM y señales digitales. Posteriormente, se establece la comunicación I2C con el sensor MPU-6050 y sus registros se configuran para leer datos del giroscopio y el acelerómetro. Se realiza una calibración inicial del giroscopio para obtener un desplazamiento en el eje Z. Se definen funciones para controlar el sentido de rotación (horario/antihorario) y detener los cuatro motores.

El modelo TensorFlow Lite se carga y configura para la detección de plantas, y la cámara se inicializa con la resolución y la velocidad de fotogramas establecidas. En el bucle principal, las imágenes se capturan continuamente, se preprocesan para la inferencia del modelo y se clasifican en "PLANTA SALUDABLE", "PLANTA ENFERMA" o "SIN PLANTA". En base a la clasificación, el robot realiza diferentes acciones: si se detecta “SIN PLANTA”, el robot avanza y monitorea el tiempo sin planta; Si se detecta una planta, el robot avanza, se detiene y espera. Además, el giroscopio se lee continuamente para controlar el ángulo de giro. Si se detecta "NO HAY PLANTA" durante más de 7 segundos, el robot gira 10 grados hacia la izquierda y avanza. Al finalizar, los recursos se cierran: se cierra la ventana OpenCV, se detiene la cámara y se limpian los pines GPIO. Este código combina control de hardware y procesamiento de imágenes para crear un robot autónomo capaz de identificar y reaccionar ante la presencia de plantas.

Una vez realizado el código principal se realiza el cálculo de consumo de cada elemento electrónico utilizado en la implementación del rover, empezando con los encargados de otorgarle movimiento al robot móvil, para aquello se utilizaron 4 motorreductores, generalmente esta clase de motores operan a 3 hasta 12 voltios, en la siguiente tabla se observan sus valores de consumo tanto para voltaje como amperaje.

Tabla 11. Valores de consumo del motorreductor.

Voltaje de operación	6V
Corriente sin carga	120 mA
Corriente máxima bajo carga	250 mA

En este caso se utilizaron 4 motorreductores, de modo que multiplicamos dichos valores por cuatro.

Tabla 12. Valores de consumo de cuatro motorreductores.

Corriente sin carga total	Corriente máxima bajo carga total
$4 \times 120mA = 280mA$	$4 \times 250mA = 1000mA$

El consumo de la raspberry pi 4 modelo B tiene la siguiente table de requisitos para su alimentación.

Tabla 13. Valores de consumo de la Raspberry pi 4.

Voltaje	5V
Corriente típica	700 mA
Corriente máxima	3 A

Donde la corriente típica se refiere al consumo de corriente que consume la raspberry pi 4 cuando opera sin los demás elementos periféricos, y la corriente máxima cuando utiliza todos sus periféricos a carga máxima.

Para el sensor MPU6050 se tiene las siguientes cifras de consumo.

Tabla 14. Valores de consumo del sensor MPU6050.

Voltaje de operación	3.3V
Corriente típica	3.9 mA

Luego tenemos las especificaciones para los puentes H L298N.

Tabla 15. Valores de consumo del puente H L298N.

Voltaje de operación	7 - 12V
Corriente de reposo	36 mA
Corriente máxima por canal	2 A

Para dos puentes H con cuatro canales el cálculo se hace a continuación.

Tabla 16. Consumo total de corrientes de dos Puentes H L298N.

Corriente de reposo total	Corriente máxima total
$4 \times 36mA = 144mA$	$4 \times 2\ A = 8\ A$

Y por último tenemos al regulador de voltaje sin embargo entre sus especificaciones tiene una eficiencia típica de 90%.

Una vez definido los voltajes y corrientes que consumen cada uno de los elementos electrónicos, calculamos el consumo total de energía, y en primer lugar se convierten los valores a vatios (W).

Motorreductores

Sin carga

$$P_{motor\_\text{sin }\_carga} = V_{motor} \times A_{motor\_\text{sin }\_carga}$$

Ecuación 6. Cálculo de potencia del motorreductor sin carga.

$$P_{motor\_\text{sin }\_carga} = 6V \times 0.48A = 2.88W$$

Con carga máxima

$$P_{motor\_carga\_max} = V_{motor} \times A_{motor\_carga\_max}$$

Ecuación 7. Cálculo de potencia del servomotor con carga máxima.

$$P_{motor\_carga\_max} = 6V \times 1.0A = 6W$$

Raspberry pi 4 modelo B

Típico

$$P_{rasp\_tipico} = V_{rasp} \times A_{rasp\_tipico}$$

Ecuación 8. Cálculo de potencia de Raspberry pi 4.

$$P_{rasp\_tipico} = 3V \times 0.42A = 1.26W$$

Agregando la eficiencia del regulador

$$P_{rasp\_tipico\_reg} = \frac{P_{rasp\_tipico}}{0.9}$$

Ecuación 9. Potencia de raspberry pi 4 agregando la eficiencia del regulador.

$$P_{rasp\_tipico\_reg} = \frac{1.26W}{0.9} \approx 1.4W$$

Máximo

$$P_{rasp\_max} = V_{rasp} \times A_{rasp\_max}$$

Ecuación 10. Cálculo de potencia de paspberry pi 4 a su máximo rendimiento.

$$P_{rasp\_max} = 3V \times 1.8A = 5.4W$$

Agregando la eficiencia del regulador

$$P_{rasp\_max\_reg} = \frac{P_{rasp\_max}}{0.9}$$

Ecuación 11. Cálculo de potencia de paspberry pi 4 a su máximo rendimiento agregando la eficiencia del regulador.

$$P_{rasp\_max\_reg} = \frac{5.4W}{0.9} \approx 6W$$

Sensor MPU6050

Típico

$$P_{mpu} = V_{mpu} \times A_{mpu\_tipico}$$

Ecuación 12. Cálculo de consumo típico del sensor MPU6050.

$$P_{mpu} = 3.3V \times 0.0039A = 0.01287W$$

Puentes H L298N

Reposo

$$P_{l298n\_\text{sin }\_carga} = V_{l298n} \times A_{l298n\_reposo}$$

Ecuación 13. Cálculo de consumo del puente H en reposo.

$$P_{l298n\_\text{sin }\_carga} = 7.4V \times 0.144A = 1.0656W$$

Carga máxima

$$P_{l298n\_carga\_max} = V_{l298n} \times A_{l298n\_carga\_max}$$

Ecuación 14. Cálculo de consumo del puente H con carga máxima.

$$P_{l298n\_carga\_max} = 7.4V \times 8A = 59.2W$$

Consumo total de energía

Consumo mínimo

$$P_{total\_min} = P_{motor\_\text{sin }\_carga} + P_{rasp\_tipico\_reg} + P_{mpu} + P_{l298\_\text{sin }\_carga}$$

Ecuación 15. Cálculo total del consumo mínimo del sistema.

$$P_{total\_min} = 2.88W + 1.4W + 0.01287W + 1.0656W$$

$$P_{total\_min} = 5.35847W$$

Consumo máximo

$$P_{total\_max} = P_{motor\_carga\_max} + P_{rasp\_\max\ \_reg} + P_{mpu} + P_{l298\_carga\_max}$$

Ecuación 16. Cálculo total del consumo máximo del sistema.

$$P_{total\_max} = 6W + 6W + 0.01287W + 59.2W$$

$$P_{total\_max} = 5.35847W$$

Tiempo de operación

Para determinar el tiempo que dura nuestro rover en operación se deben realizar los siguientes cálculos de:

Corriente total requerida

Consumo mínimo

$$I_{min} = \frac{P_{total\_min}}{7.4V}$$

Ecuación 17. Cálculo de corriente total con un consumo mínimo del sistema.

$$I_{min} = \frac{5.36W}{7.4V} \approx 0.72A$$

Consumo máximo

$$I_{max} = \frac{P_{total\_max}}{7.4V}$$

Ecuación 18. Cálculo de corriente total con un consumo máximo del sistema.

$$I_{max} = \frac{71.21W}{7.4V} \approx 9.63A$$

Duración de la batería

Consumo mínimo

$$Duración = \frac{Capacidad}{Corriente\ mínima}$$

Ecuación 19. Cálculo de tiempo de duración del sistema con un consumo mínimo.

$$Duración = \frac{3.5Ah}{0.72A} \approx 4.86\ horas$$

Consumo máximo

$$Duración = \frac{Capacidad}{Corriente\ máxima}$$

Ecuación 20. Cálculo de tiempo de duración del sistema con un consumo máximo.

$$Duración = \frac{3.5Ah}{9.63A} \approx 0.36\ horas$$

3.2. Estudio de Factibilidad

Para determinar el costo de la propuesta para la Implementación de un rover automatizado con tecnología de visión artificial para la clasificación de tres tipos de plantas sanas/enfermas en cultivos de ciclo corto, de detallan los costos en la tabla 17.

Tabla 17. Valores para la implementación de la propuesta.

CANT.	DESCRIPCION	COSTO
1	Raspberry pi 4	$155.00
1	Cámara Raspberry de 5 MPX	$12.00
2	Puente H L298N	$9.00
4	Motorreductor	$6.00
1	Batería Lipo de 3500 mA	$45.00
1	Otros Gastos	$50.00
1	Regulador de Voltaje DC -DC	$5.00
1	MPU6050	$7.00
26	Piezas 3D	$75.00
TOTAL		$364.00

3.3. Resultados

En esta sección se muestran las diferentes pruebas realizadas en este documento, en total se realizaron 3 pruebas para el sistema de visión artificial implementada en el rover, utilizando tres diferentes modelos de entrenamiento de IA, cuyos resultados se muestran en la siguiente tabla con el modelo entrenado con 50 muestras.

Tabla 18. Pruebas para el modelo de visión artificial de 50 muestras.

MODELO DE IA CON 50 MUESTRAS					
Numero de pruebas	Detección de la IA (%)		Detección Humana	Aciertos	
	Lechuga				
	Sana	Enferma		Correctos	Incorrectos
1	8.888867e-07	5.88899e-01	Sana		✓
2	4.387029e-09	9.66544e-01	Enferma	✓	
3	7.666867e-01	6.88844e-06	Enferma		✓
4	7.888867e-01	6.55543e-06	Enferma		✓
5	9.999976e-01	2.67674e-07	Sana	✓	

Como se muestra en la tabla 18, se observa que existe una inclinación del primer modelo hacia la clase de “Enfermas”, proporcionando datos erróneos y falsos positivos como se visualiza en las pruebas 1, 3 y 4. Para la realización de las pruebas de esta propuesta se tenían 4 plantas de lechuga sanas y una enferma.

Para estas primeras pruebas también se utilizan imágenes de las lechugas enfermas y sanas tomadas de manera externa y procesadas por la computadora en el código realizado en el entorno de Google Colab con el fin de tener una visión más amplia y comparar el procesamiento de la tarjeta Raspberry Pi 4 con un ordenador.

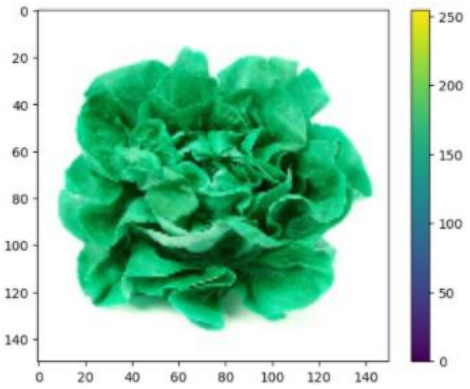


Figura 81. Verificación de la imagen que se va a analizar.

Fuente: Autoría propia.

Una vez realizada la prueba en el entorno de un ordenador, se procede a probar en la tarjeta Raspberry Pi 4 para la respectiva comparación. Dentro de una de las pruebas realizadas se presentan errores debido a los espectros de luz y las sombras generadas en el ambiente ya que el modelo de 50 muestras lo toma como una planta enferma y en general presenta variaciones en los resultados de inferencia dentro de cada una de las pruebas. (Ver Figura 82 a 86)

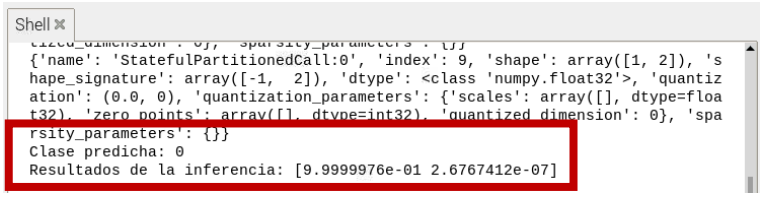


Figura 82. Resultados de inferencia con la clase 0 del modelo entrenado 1.

Fuente: Autoría propia.

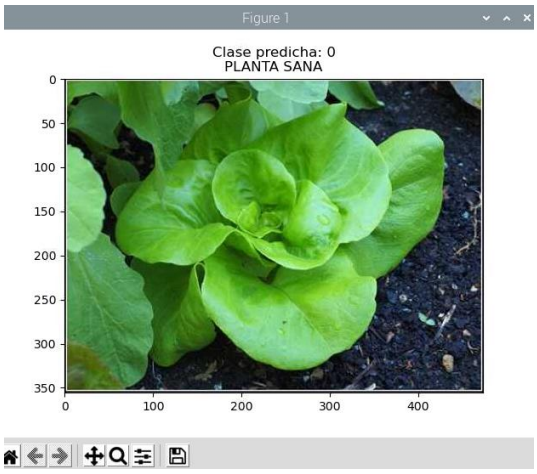


Figura 83. Gráfica de la imagen procesada con la clase 0.

Fuente: Autoría propia.

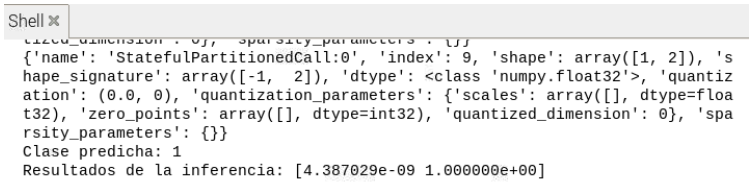


Figura 84. Resultados de inferencia con la clase 1 del modelo entrenado 1.

Fuente: Autoría propia.

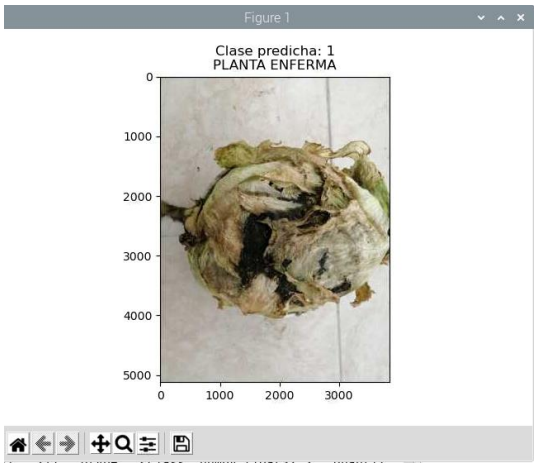


Figura 85. Gráfica de la imagen procesada con la clase 1.

Fuente: Autoría propia.

Tabla 19. Pruebas para el modelo de visión artificial de 100 muestras.

MODELO DE IA CON 100 MUESTRAS					
Numero de pruebas	Detección de la IA (%)		Detección Humana	Aciertos	
	Lechuga				
	Sana	Enferma		Correctos	Incorrectos
1	9.98865 e-01	1.03934e-04	Sana	✓	
2	8.332267e-05	8.77799e-01	Enferma		✓
3	0.087864	0.922135	Enferma	✓	
4	5.666544e-01	6.88899e-06	Sana	✓	
5	4.856333e-04	7.88899e-01	Sana		✓

Culminadas las pruebas del primero modelo entrenado con 50 muestras, se cambia la ruta del modelo y se la reemplaza con el que tiene 100 muestras y 100 épocas de entrenamiento, se observa que en la tabla 14 los falsos positivos se regularon moderadamente con valores más cercanos al correcto, sin embargo, sigue

existiendo un porcentaje de error, el cual se corrige con el aumento de muestras que el modelo entrenado pueda procesar y el aumento de las épocas para que se realice un aprendizaje profundo.(Ver figuras 87 a 90)

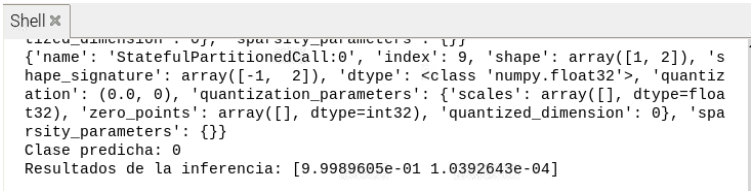


Figura 86. Resultados de inferencia con la clase 0 del modelo entrenado 2.

Fuente: Autoría propia.

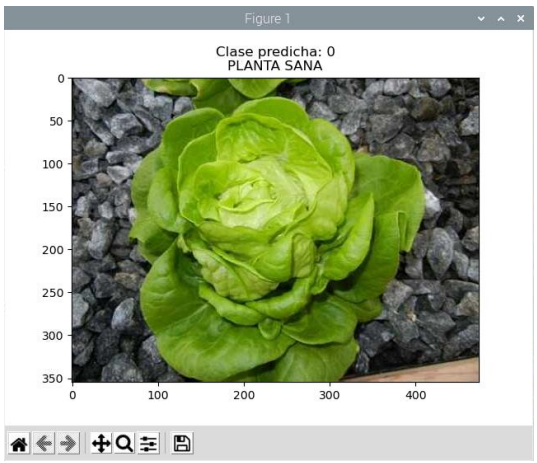


Figura 87. Gráfica de la imagen procesada con la clase 0 con el modelo entrenado 2.

Fuente: Autoría propia.

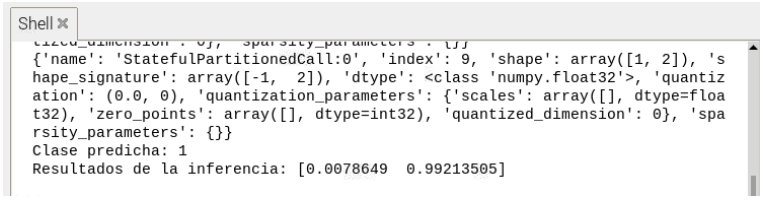


Figura 88. Resultados de inferencia con la clase 1 del modelo entrenado 2.

Fuente: Autoría propia.

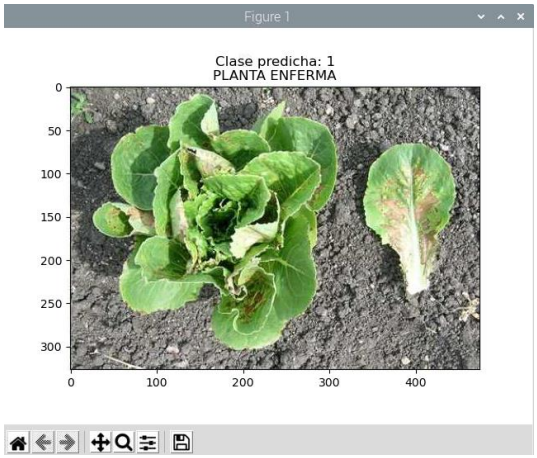


Figura 89. Gráfica de la imagen procesada con la clase 1 con el modelo entrenado 2.

Fuente: Autoría propia.

Tabla 20. Pruebas para el modelo de visión artificial de 1347 muestras de la CNN.

MODELO DE IA CON 1347 MUESTRAS PARA LA CNN					
Numero de pruebas	Detección de la IA (%)		Detección Humana	Aciertos	
	Lechuga				
	Sana	Enferma		Correctos	Incorrectos
1	1.623400e-10	1.00000e-00	Enferma	✓	
2	8.83278e-01	8.86899e-01	Enferma		✓
3	1.000000e-00	9.76743e-11	Sana	✓	
4	9.999979e-11	1.25303e-09	Sana	✓	
5	1.000000e-00	5.88899e-06	Sana	✓	

Dentro de las pruebas con el modelo con 1347 muestras tenemos la tabla 20, que nos permite visualizar un cambio con respecto a las primeras pruebas realizadas con el modelo entrenado con 50 y 100 muestras del perceptrón multicapa.

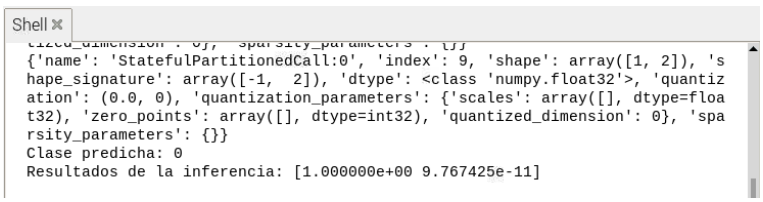


Figura 90. Resultados de inferencia con la clase 0 del modelo entrenado 3.

Fuente: Autoría propia.

Donde se observa que los resultados de inferencia son cercanos a la unidad y en las pruebas 1, 3 y 5 son uno, como se observa en la figura 91, esto nos indica

que el modelo de aprendizaje profundo nos permite realizar mejoras con respecto a la identificación de varios objetos en este caso plantas. (Ver Figuras 91 a 94)

Figura 91. Gráfica de la imagen procesada con la clase 0 con el modelo entrenado 3.

Fuente: Autoría propia.

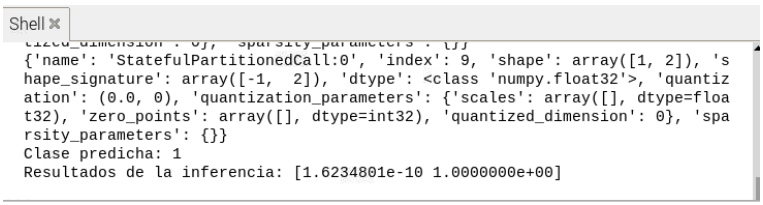


Figura 92. Resultados de inferencia con la clase 1 del modelo entrenado 3.

Fuente: Autoría propia.

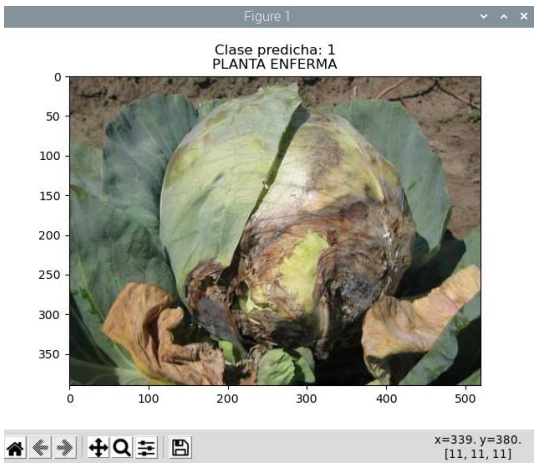


Figura 93. Gráfica de la imagen procesada con la clase 1 con el modelo entrenado 3.

Fuente: Autoría propia.

Finalizada las etapas de pruebas del modelo de aprendizaje profundo entrenado con 3 números de muestras diferentes, se realiza pruebas con el último modelo

entrenado para verificar su funcionamiento en un proceso en tiempo real, el cual es el que se pretende realizar en esta propuesta.

Mediante el uso de las librerías mencionadas en las tablas de los Anexos C y D, las cuales se descargaron en la tarjeta Raspberry Pi 4, se procede a utilizar picamera para los permisos y captura de imágenes con la cámara compatible con la tarjeta, posteriormente se usa Cv2 para el procesamiento de las imágenes proporcionadas por picamera, se realiza el debido procesamiento y redimensionamiento de la cámara. Iniciamos el modelo con la ruta en la que se encuentra guardado dentro de la tarjeta y finalizamos con la condición de las clases detectadas que en este caso son dos, la clase Sanas, para las plantas de lechuga que no presenten anomalías ni daño en su superficie, y la clase Enfermas, para las plantas cuyas hojas están afectadas por un virus, enfermedad, u hongos.

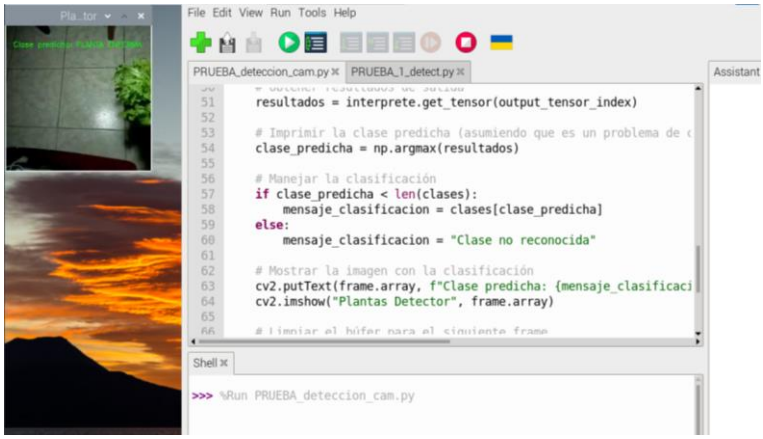


Figura 94. Prueba de modelo en tiempo real para la identificación de la clase 1.

Fuente: Autoría propia.

Figura 95. Ventana de visualización de la cámara en tiempo real con la clase 1.

Fuente: Autoría propia.

Dentro de la figura 97 se observa al cultivo de lechuga con una luz direccionada al centro del frame con la mitad de la planta dentro del mismo con el objetivo de detectar algún defecto en el modelo, sin embargo, el modelo entrenado detecta un cultivo sano, acertando con la comparación de la vista humana como se observan en las tablas de prueba que se presentan en resultados.

Figura 96. Ventana de visualización de la cámara en tiempo real con la clase 0.

Fuente: Autoría propia.

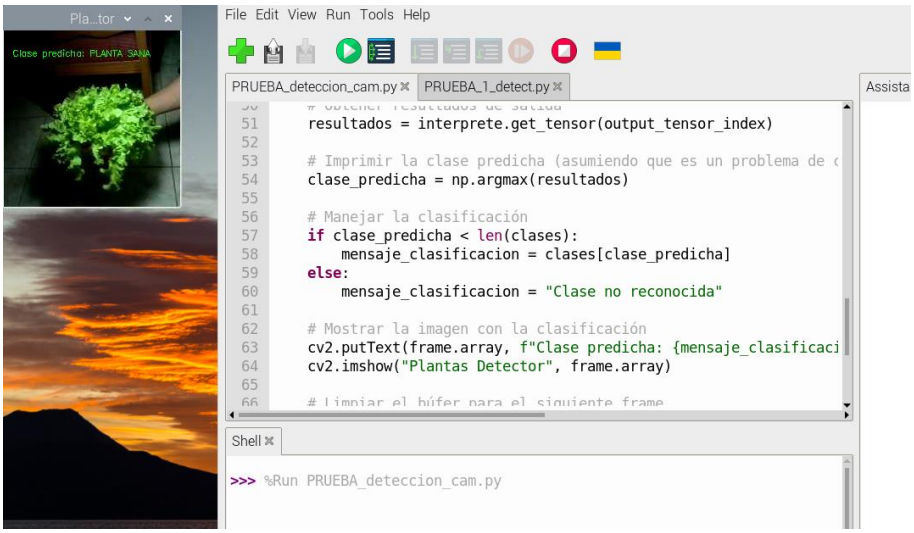


Figura 97. Prueba de modelo en tiempo real para la identificación de la clase 0.

Fuente: Autoría propia.

Dentro de las figuras 97 y 98 se visualiza la captura de la cámara con la ejecución del código con el modelo entrenado dentro de un ambiente controlado, para poder realizar pruebas con distintos direccionamientos de la luz además de probar con diferentes tipos de colores, ya que dentro del campo se presentan

diferentes variables como lo es la luz presente en el día, la cual va variando con respecto al tiempo, es por eso que al realizar el entrenamiento de este modelo, se capturo imágenes con diferentes calidades de luz, como luz cálida y blanca, para poder reducir los posibles errores que se puedan presentar.

Pruebas en Campo

Figura 98. Rover analizando el cultivo.

Fuente: Autoría propia.

En las siguientes figuras 99 hasta la 102 se observa al robot móvil desplazándose por el campo de cultivo, analizando cada una de las plantas, en donde el rover pasa analizándolas y determinando si es una planta sana o una planta enferma, asignándole una categoría según corresponda para cada uno de los cultivos, estas pruebas se las realizaron al medio día, para observar el rendimiento del robot, y su desenvolvimiento dentro de una superficie porosa como se observa en la figura 100, donde se encuentran diferentes relieves.

Figura 99. Vista superior del rover en campo.

Fuente: Autoría propia.

Figura 100. Vista frontal del rover en campo.

Fuente: Autoría propia.

Dentro de las pruebas de campo se presentaron retrasos al momento de realizar el giro en el propio eje del robot móvil ya que la superficie donde se está desplazando es porosa, sin embargo, el robot móvil completa su giro al final de la fila de cultivo como se visualiza en la figura 126 y continua con la siguiente hilera de cultivos, dentro de algunas observaciones realizadas en las pruebas de campo se destacan la del tiempo de trabajo del rover implementado que dura un aproximado de 4 horas, respetando los cálculos realizados en el desarrollo, ya que no realiza el consumo de todos los recursos disponibles de la raspberry pi 4.

Figura 101. Rover analizando la última planta de la fila de cultivo.

Fuente: Autoría propia.

CONCLUSIONES

En la etapa de visión artificial implementada en la tarjeta Raspberry Pi 4, se desarrolló un algoritmo en Python, empleando la biblioteca de OpenCV y el Framework de TensorFlow Lite para el reconocimiento en tiempo real según el estado de la planta que en este caso es la lechuga, estas bibliotecas permitieron adquirir fotogramas en tiempo real de un cultivo de ciclo corto para realizar el procesamiento adecuado de la imagen, logrando así implementar una solución para la identificación de una planta enferma, de modo que las bibliotecas utilizadas en la presente propuesta se puede adicionar instrucciones para robustecer el sistema de detección del presente trabajo además de poder adicionar otras plantas para su detección.

La implementación del rover autónomo se realizo exitosamente luego de ser diseñado previamente con las medidas de cada elemento electrónico en el software Solid Works, además de tener un sistema electrónico compuesto principalmente por una cámara que permite la captura en tiempo real de los cultivos de ciclo corto y junto a la raspberry pi 4 que realiza el procesamiento de las imágenes.

Se logró una precisión aceptable en la clasificación de plantas sanas y enfermas gracias a la introducción de modelos de visión por ordenador entrenados con la libreria TensorFlow. El rendimiento de la clasificación dependerá en gran medida de la calidad del conjunto de datos de entrenamiento y de la complejidad del modelo utilizado como en el caso de la propuesta.

La implementación de TensorFlow Lite en Raspberry Pi 4 ayuda a implementar de manera eficiente modelos de visión por computadora. La capacidad de convertir modelos entrenados en TensorFlow al formato tflite permite una integración más fluida con el hardware Raspberry Pi.

Con las pruebas realizadas con los 3 modelos entrenados se observó como con cada modelo realizado se iba mejorando el proceso de identificación de plantas sanas y enfermas, esto es debido al aumento de muestras y épocas utilizadas en cada uno de los entrenamientos realizados al transcurso de esta propuesta.

RECOMENDACIONES

Para poder obtener un porcentaje de fiabilidad superior al obtenido con la implementación de este rover con un sistema de detección de visión artificial, se deben realizar modificaciones en el modelo y sus componentes, como una cámara con características mejoradas para una mejor detección de plagas debido a que las mismas pueden llegar a iniciar con una pequeña área de la planta en donde la resolución de la cámara marcaria una diferencia al momento de capturar y procesar la imagen del cultivo, de modo que tendría una mayor capacidad para detectar tonalidades en los colores de la planta.

Se requeriría de un mayor número de muestras o imágenes para mejorar el entrenamiento del modelo a utilizar, además de usar otros frameworks en la placa Raspberry pi 4 como Tensorflow en su versión normal ya que en la versión del sistema operativo que se instaló en la placa no era compatible con las librerías y dependencias que necesitaba Tensorflow para poder ejecutarse, sin embargo, debido a que los repositorios de Raspian están en constante actualización, es posible que creen un nuevo SO (Sistema Operativo) o nuevas actualizaciones para los actuales que permitan utilizar las herramientas de visión artificial con más libertad.

Al utilizar la tarjeta Raspberry pi 4 con el sistema de visión artificial, esta tarjeta consume la mayor parte de los recursos disponibles de la memoria RAM, lo que provoca un aumento de temperatura en el CPU, aunque tenga instalado disipadores de calor dentro de la misma, además de tener un ventilador que amenora su temperatura con las condiciones en la que el prototipo trabaja no suele ser suficiente, por eso se recomienda utilizar un método de refrigeración como puede ser un ventilador con motores brushless que no consumen un amperaje excesivo.

Si se desea reforzar o agregar otro actuador al rover implementado en este proyecto es recomendable realizar el dimensionamiento de las baterías para poder alimentar de manera eficiente a la placa Raspberry pi 4, ya que si no le llega la energía suficiente puede presentar fallos al momento de detectar alguna plaga y en el sistema de visión artificial en general.

BIBLIOGRAFÍA

[1] Organizacion de las Nacines Unidas, «Naciones Unidas,» [En línea]. Available: https://www.un.org/es/un75/impact-digital-technologies#:~:text=Los%20avances%20digitales%20pueden%20apoyar,logro%20de%20la%20alfabetizaci%C3%B3n%20universal.. [Último acceso: 9 junio 2022].

[2] South-South and Resources Mobilization Division, «Política de desarrollo agrícola,» 2004.

[3] A. Derlys Collado y M. Del Pilar Délano, «La agricultura del futuro: cambios y desafíos,» 7 Diciembre 2018. [En línea]. Available: https://comunidades.cepal.org/ilpes/sites/default/files/2018-12/Cierre%20Alfredo%20D.%20Collado.pdf.

[4] Organizacion de las Naciones Unidas, «Cambio climático y medioambiente,» 2021.

[5] C. A. Cásares Farías , N. Farías Mendoza, N. García Días y A. García Rebolledo, «Procesamiento de imágenes de plantas ornamentales multiescala para calcular su crecimiento,» *3C TIC: Cuadernos de desarrollo aplicados a las TIC,* vol. 6, nº 3, pp. 14-17, 2017.

[6] V. A. Morales Carreón y L. A. Rodríguez Castrejón, «Aplicacion movil para la identificacion de plantas semidesierticas ubicadas en el norte del estado de Chihuahua,» pp. 2-4, 2016.

[7] N. F. M. N. G. D. A. G. R. Carlos Alberto Cásares Farías, «Procesamiento de imágenes de plantas ornamentales multi-escala para calcular su crecimiento,» *Ciencias,* vol. 6, nº 3, 2017.

[8] J. G. G. A. V. M. Arévalo, «LA LIBRERÍA DE VISIÓN ARTIFICIAL OPENCV,» pp. 1-3.

[9] M. W. Schwarz, W. B. Cowan y J. C. Beatty, «An Experimental Comparison of RGB, YIQ, LAB, HSV, and Opponent Color Models,» *ACM Transactions on Graphics,* vol. 6, pp. 126-127, 1987.

[10] G. Solano, «MES,» 3 Septiembre 2019. [En línea]. Available: https://omes-va.com/basicvideo/.

[11] G. Solano, «MES,» 2 Septiembre 2019. [En línea]. Available: https://omes-va.com/leerimagen/.

[12] D. reservados, «Extendiendo Python con C o C++,» [En línea]. Available: https://docs.python.org/es/3.8/extending/extending.html.

[13] R. Marín, «¿Qué es OpenCV? Instalación en Python y ejemplos básicos,» *Revista Digital INESEM,* 2020.

[14] L. Del Valle, «Programacion facil,» 2015. [En línea]. Available: https://programarfacil.com/blog/vision-artificial/detector-de-bordes-canny-opencv/.

[15] D. Alercia, «DefinicionABC,» [En línea]. Available: https://www.definicionabc.com/ciencia/planta.php.

[16] M. Gago, «Ecologia verde,» 15 diciembre 2017. [En línea]. Available: https://www.ecologiaverde.com/clasificacion-de-las-plantas-segun-su-duracion-889.html. [Último acceso: 18 junio 2024].

[17] «Tipos de plantas: ¿cómo se clasifican? - Pequeocio,» Pequeocio, [En línea]. Available: https://www.pequeocio.com/tipos-de-plantas/.

[18] R. Bellefontaine, S. Petit, M. Pain-Orcet, P. Deleporte y J.-G. Bertault, «Los árboles fuera del bosque,» [En línea]. Available: https://www.fao.org/4/y2328s/y2328s00.htm#toc. [Último acceso: 18 junio 2024].

[19] Anónimo, «Lechuga - Información general,» Listado de frutas, [En línea]. Available: https://www.frutas-hortalizas.com/Hortalizas/Presentacion-Lechuga.html.

[20] J. Japon Quintero, «Hojas Divulgadoras del Ministerio de Agricultura de España,» [En línea]. Available: https://www.mapa.gob.es/ministerio/pags/biblioteca/hojas/hd_1977_10.pdf. [Último acceso: 2024 junio 13].

[21] K. . A. Reinoso Rodríguez, «DESARROLLO MORFOFISIOLÓGICO Y PRODUCTIVO DE DOS VARIEDADES DE LECHUGA (Lactuca sativa) CON DIFERENTES DISTANCIAS DE PLANTACIÓN EN LAS CONDICIONES DEL CENTRO DE INVESTIGACIÓN, POSGRADO Y CONSERVACIÓN AMAZÓNICA (CIPCA), PROVINCIA DE NAPO.,» 2019.

[22] «Inicio - Ayuntamiento de Madrid,» [En línea]. Available: https://www.madrid.es/portales/munimadrid/es/Inicio/El-Ayuntamiento/Calidad-y-Evaluacion/Modelos-de-Calidad-y-Excelencia/Cartas-de-Servicios/Cartas-de-Servicios-vigentes/Servicios-prestados-Prevencion-y-Control-de-Plagas/?vgnextfmt=default&vgnextoid=7eda3. [Último acceso: 12 06 2023].

[23] «Ecoplagas,» 29 05 2020. [En línea]. Available: https://ecoplagas.es/tipos-de-plagas/. [Último acceso: 12 06 2023].

[24] A. Dughetti, «PULGONES CLAVE PARA IDENTIFICAR LAS FORMAS ÁPTERAS QUE ATACAN A LOS CEREALES,» diciembre 2012. [En línea]. Available: http://www.ucv.ve/fileadmin/user_upload/facultad_agronomia/Zoologia_Agricol

a/Manejo_Integrado/Material_Interes/Pulgones_clave_para_identificar_las_formas_apteras_que_atacan_cereales_by_dughetti_arturo_inta.pdf. [Último acceso: 18 junio 2024].

[25] O. Valarezo C., E. Cañarte B., B. Navarrete C., J. Ma. Guerrero y B. Arias, «Diagnóstico de la "mosca blanca" en Ecuador,» 2008. [En línea]. Available: https://dspace.ups.edu.ec/bitstream/123456789/8720/1/Diagnostico%20de%20la%20mosca%20blanca%20en%20Ecuador.pdf. [Último acceso: 18 junio 2014].

[26] S. Garcés, L. Lomas y E. Peralta, «Manejo de trips en el cultivo de frejol,» noviembre 2008. [En línea]. Available: https://repositorio.iniap.gob.ec/bitstream/41000/2625/1/iniapscpl297.pdf. [Último acceso: 18 junio 2024].

[27] M. D. M. Téllez Navarro, «Enemigos naturales para el control de minador en cultivos hortícolas,» 2007. [En línea]. Available: https://www.mapa.gob.es/ministerio/pags/Biblioteca/Revistas/pdf_SH%2FSH_2007_15_127_137.pdf. [Último acceso: 18 junio 2024].

[28] IBM, «IBM - Deutschland | IBM,» [En línea]. Available: https://www.ibm.com/es-es/topics/computer-vision. [Último acceso: 2023 06 12].

[29] Lanner Electronics Canada Ltd., «Lanner,» [En línea]. Available: https://www.lanner-america.com/es/blog-es/la-ia-y-la-vision-artificial-redefinen-la-agricultura-de-precision/. [Último acceso: 1 12 2023].

[30] J. Pérez Porto y M. Merino, «Definición.de,» 8 11 2010. [En línea]. Available: https://definicion.de/identificacion/#:~:text=Identificaci%C3%B3n%20es%20la%20acci%C3%B3n%20y,datos%20necesarios%20para%20ser%20reconocido).. [Último acceso: 12 06 2023].

[31] C. A. S. Ríos, «Detección de enfermedades en plantas de crisantemo, por medio de visión artificial aplicada a imágenes multiespectrales,» Medellín, Colombia, 2020.

[32] Etecé, «Concepto,» 23 11 2021. [En línea]. Available: https://concepto.de/clasificacion/. [Último acceso: 12 06 2023].

[33] E. C. Andrade Tepán, «Estudio de los principales tipos de redes neuronales y las herramientas para su aplicación,» Cuenca, 2013.

[34] Universidad de Oviedo Area de Ingenieria de Sistemas y Automatica., «El microprocesador,» [En línea]. Available: http://www.isa.uniovi.es/~alonsog/Microcontrolador/T2%20El%20Microprocesador.pdf. [Último acceso: 18 junio 2024].

[35] J. F. Ortega Noroña y L. M. Yapo Pillajo, «Construcción de un robot móvil híbrido omnidireccional,» Quito, 2017.

[36] M. Alatise y G. Hancke, «A Review on Challenges of Autonomous Mobile,» IEEEAccess, 2020.

[37] D. Ortega, T. Moposita, W. Aguilar, M. Paredes, G. León y A. Jara Olmedo, «Math Model of UAV Multi Rotor Prototype with Fixed Wing Aerodynamic Structure for a Flight Simulator,» 2017.

[38] B. Pollard y P. Tallapragada, «An aquatic robot propelled by an internal rotor,» 2016.

[39] J. F. Reyes Cortés, «Robótica Control De Robots Manipuladores,» ALFAOMEGA, 2011.

[40] B. S. a. O. Khatib, «Springer Handbook of Robotics,» 2008.

[41] K. Shabalina, «Comparative Analysis of Mobile Robot Wheels Design,» de *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*, 2018.

[42] J. Bok Song y K. Seok Byun, «Design and Control of an Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels,» 2006.

[43] D. Roa Ramos, B. Báez Ruiz y D. Mosquera Osuna, «Desarrollo de control e implementación de robot móvil con la capacidad de solucionar laberintos complejos».

[44] W. Nukulwuthiopas y S. Laowattana, «Dynamic modeling of a one-wheel robot by using Kane's method,» de *Industrial Technology, 2002. IEEE ICIT '02. 2002 IEEE International Conference onVolume: 1*, 2002.

[45] B. Siliciano y O. Khatib, «Robotics and the Handbook».

[46] A. Stefek, T. Van Pham, V. Krivanek, H. Lam Pham y K. Lam Pham, «Energy Comparison of Controllers Used for a Differential Drive Wheeled Mobile Robot,» IEEE, 2020.

[47] J. Heon Chung, B. Ju Yi, W. Kuk Kim y H. Lee, «The dynamic modeling and analysis for an omnidirectional mobile robot with three caster wheels,» 2003.

[48] E. Maulana, M. Aziz Muslim y V. Hendrayawan, «Inverse kinematic implementation of four-wheels mecanum drive mobile robot using stepper motors,» 2015.

[49] R. Martinez-Clark, C. Cruz-Hernández, J. Pliego-Jimenez y A. Arellano-Delgado, «Control algorithms for the emergence of self-organized behaviours in swarms of differential-traction wheeled mobile robots,» 2018.

[50] P. Zhang, L. Gao y Y. Zhu, «Study on control schemes of flexible steering system of a multi-axle all-wheel-steering robot,» 2016.

[51] J. F. Ortega Noroña y L. M. Yapo Pillajo, «Construccion de un robot movil hibrido omnidireccional,» 2017.

[52] «Home,» [En línea]. Available: https://www.isahispana.com/portals/0/docs/treecare/insect_disease_spanish.pdf. [Último acceso: 19 06 2023].

[53] «Redagrícola Chile,» [En línea]. Available: https://www.redagricola.com/cl/factores-agronomicos-y-ambientales-que-favorecen-las-plagas-en-citricos/. [Último acceso: 19 06 2023].

[54] W. Narea, «El Universo | Noticias de Ecuador y del mundo,» 07 04 2017. [En línea]. Available: https://www.eluniverso.com/noticias/2017/04/07/nota/6126698/plagas-lluvias-afectan-1000-hectareas-peninsula/. [Último acceso: 19 06 2023].

[55] «El portal Ãºnico del gobierno. | gob.mx,» [En línea]. Available: https://www.gob.mx/cms/uploads/attachment/file/26576/riesgos.pdf. [Último acceso: 19 06 2023].

[56] M. Mulero Lorenzo, «Desarrollo de una aplicacion Android de ayuda en la identificacion de plagas y enfermedades de huerto,» 2020.

[57] J. T. J. G. L. F. R. C. J. S. Y. R. A. &. J. L. D. Cervantes, «Análisis comparativo de las técnicas utilizadas en un sistema de reconocimiento de hojas de planta.,» Revista Iberoamericana de Automática e Informática industrial, 2017.

[58] C. A. Vallejos Fonseca, «Software para identificar y localizar diversas variedades de plantas de quinua en el distrito de Querocoto-Cajamarca.,» 2019.

[59] J. A. Chica, Sistema de reconocimiento de maduración del tomate, mediante el procesamiento de imágenes con OpenCV y Python en una Raspberry Pi 3, DESARROLLO E INNOVACIÓN EN INGENIERÍA, 2018.

[60] A. Montano Rodríguez, «Problemas de clasificación de imágenes para la detección de malezas en los cultivos de cereales a partir de imágenes aéreas tomadas por drones.,» 2019.

[61] Y. M. &. H. N. C. Oo, «lant Leaf Disease Detection and Classification using Image Processing.,» International Journal of Research and Engineering, 2018.

[62] A. F. C. P. D. A. &. G. R. D. Y. JIMÉNEZ LÓPEZ, «Sistema Inteligente para el manejo de Malezas en el cultivo de Piña con Conceptos de Agricultura de Precisión.,» Ciencia y Agricultura, 2020.

[63] K. D. J. BELEÑO SAENZ, «Clasificación de los frutos de café según su estado de maduración y detección de la broca mediante técnicas de procesamiento de imágenes.,» Prospectiva, 2016.

[64] raspberrypi, «Concepto,» [En línea]. Available: https://raspberrypi.cl/que-es-raspberry/. [Último acceso: 12 06 2023].

[65] D. reservados, «TECmikro,» [En línea]. Available: https://tecmikro.com/motores/482-motorreductor-amarillo-para-arduino.html. [Último acceso: 2 12 2023].

[66] AFEL, «AFEL,» [En línea]. Available: https://afel.cl/producto/camara-raspberry-rev-1-3/. [Último acceso: 3 12 2023].

[67] L. A. V. GABRIELA, 02 diciembre 2019. [En línea]. [Último acceso: 3 junio 2024].

[68] «NAYLAMPMECHATRONICS,» [En línea]. Available: https://naylampmechatronics.com/conversores-dc-dc/196-convertidor-voltaje-dc-dc-step-down-3a-lm2596.html. [Último acceso: 3 junio 2024].

[69] Raspberry Pi, «Raspberry Pi,» [En línea]. Available: https://www.raspberrypi.com/documentation/computers/os.html. [Último acceso: 06 12 2023].

[70] Famatech Corp., «Advanced Ip Scanner,» [En línea]. Available: https://www.advanced-ip-scanner.com/es/about/#:~:text=Explorador%20de%20redes%20gratuito&text=Advance%20IP%20Scanner%20ha%20demostrado,prueban%20cada%20nueva%20versi%C3%B3n%20rigurosamente.. [Último acceso: 07 12 2023].

[71] RealVNC Limited. Todos los derechos reservados., «REALVNC,» [En línea]. Available: https://discover.realvnc.com/what-is-vnc-remote-access-technology#:~:text=VNC%20stands%20for%20Virtual%20Network,right%20in%20front%20of%20it.. [Último acceso: 07 12 2023].

[72] Python Software Foundation, «Python,» [En línea]. Available: https://www.python.org/doc/. [Último acceso: 05 12 2023].

[73] SOLIDWORKS, «SOLIDBI Inspira tu innovacion,» [En línea]. Available: https://solid-bi.es/solidworks/?v=5b61a1b298a0. [Último acceso: 3 junio 2024].

[74] GOOGLE, [En línea]. Available: https://colab.google/. [Último acceso: 3 junio 2024].

[75] OpenCV, «OpenCV,» [En línea]. Available: https://opencv.org/about/. [Último acceso: 09 12 2023].

[76] TensorFlow, «TensorFlow,» [En línea]. Available: https://www.tensorflow.org/overview?hl=es-419. [Último acceso: 10 12 2023].

[77] «StackPath,» [En línea]. Available: https://bloglatam.jacto.com/plagas-agricultura/. [Último acceso: 19 06 2023].

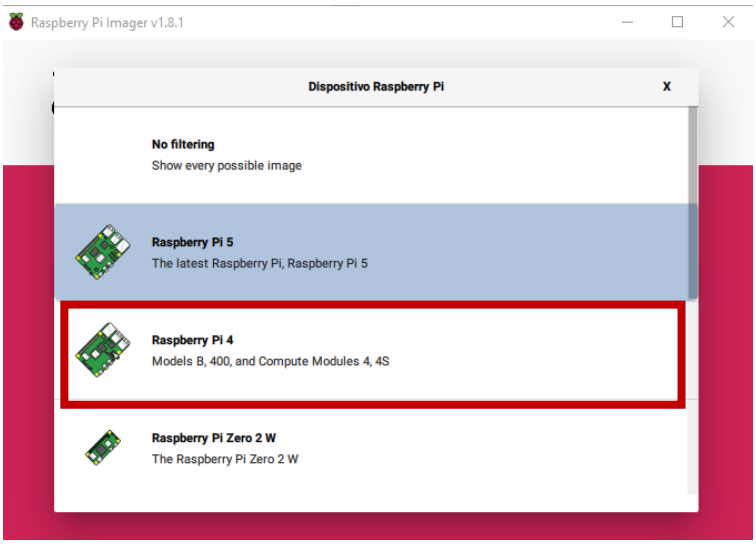
[78] A. G. González, «SERVOMOTORES Y SU FUNCIONAMIENTO ¿Qué es y cómo funciona un servomotor? Definición de servomotor diagrama de bloques del servomotor gasto de energia,» 2016.

79] V. G. LECHÓN ANZULES , 02 diciembre 2019. [En línea]. [Último acceso: 3 Junio 2024].

ANEXOS

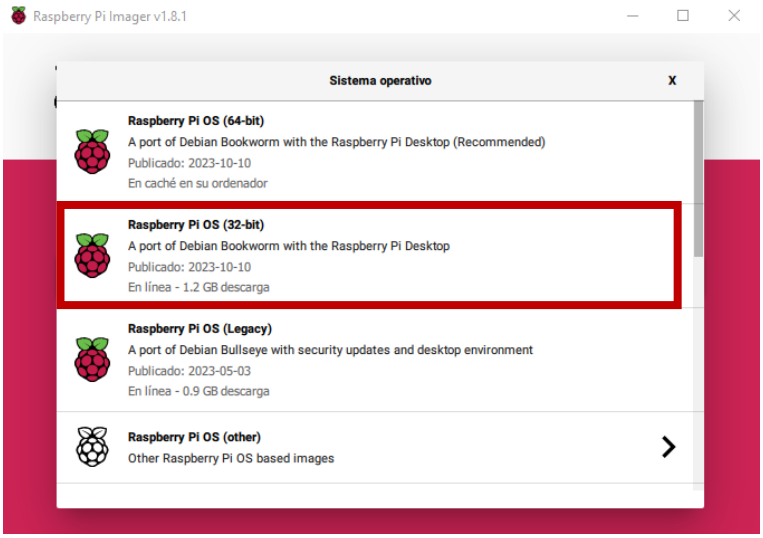
Anexo A. Instalación del sistema operativo a la Raspberry pi 4

Para poder realizar la instalación de este sistema operativo en la tarjeta Raspberry Pi tenemos que instalar previamente en un computador la herramienta de Raspberry Pi Imager. Al ejecutar como administrador se procede a seleccionar dispositivo en donde se instala el sistema operativo, para este proyecto se utiliza la tarjeta Raspberry pi 4.



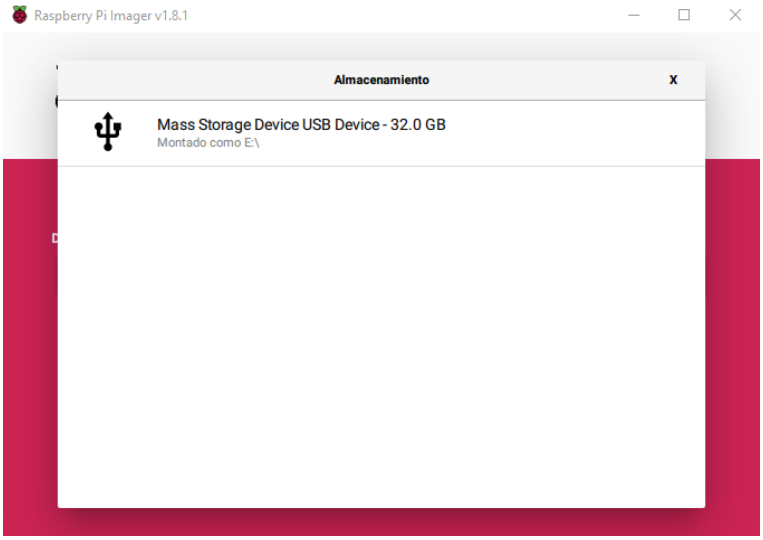
Selección del dispositivo Raspberry pi 4.

Al seleccionar el dispositivo, se continua con la selección del sistema operativo, para lo cual el fabricante recomienda que se instale el sistema operativo llamado Raspberry Pi OS de 32 bits, sin embargo, esta selección dependerá del tipo de aplicaciones que se requiera instalar y de la capacidad de almacenamiento de la tarjeta micro SD que se utiliza para el proyecto, en este caso se selecciona el sistema operativo que recomienda el fabricante (Ver Figura 14), de modo que se instalan los puertos, aplicaciones básicas y el escritorio de la tarjeta Raspberry Pi ya que posteriormente de manera manual se instalan las librerías y aplicaciones necesarias para usar la visión artificial.



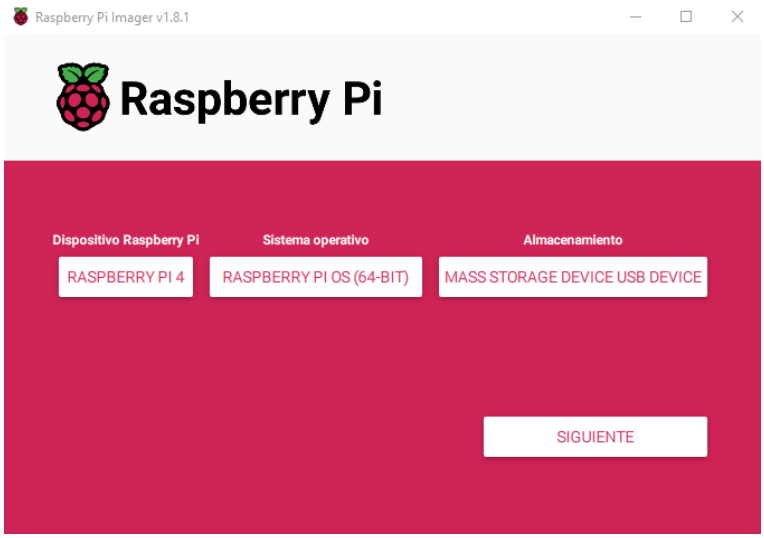
Selección del sistema operativo desde el entorno de Raspberry Pi Imager.

Posteriormente se procede a colocar la tarjeta SD con su respectivo adaptador en la computadora, de manera que la herramienta Raspberry Pi Imager pueda reconocerla y mostrarla dentro de su propio entorno, donde observamos la capacidad de almacenamiento de la tarjeta SD, que necesariamente debe estar dentro del rango de 16 GB a 64 GB, dicha tarjeta también deberá contener un soporte para el sistema de archivos FAT16 o FAT32 de manera obligatoria ya que Estas son las extensiones que puede soportar la Raspberry Pi cuatro para los archivos del cargador de arranque. Se selecciona la tarjeta que muestra el entorno de la herramienta (Ver Figura 15).



Selección de la unidad SD, para la instalación del sistema Raspberry Pi OS 32 bit.

Una vez culminado los pasos anteriores se procede a escribir OA cargar el sistema operativo para lo cual se empezará a visualizar una slider hoy con el mensaje de "Writing %", en donde el número que acompaña a este símbolo de porcentaje indicará la escritura del sistema operativo en la tarjeta microSD, una vez culminado la fase de escritura se procede a extraer la tarjeta microSD del computador, para insertarla en la ranura micro SD que posee la tarjeta Raspberry Pi 4.



Instalación del sistema operativo Raspberry Pi OS 32 bit en la tarjeta microSD.

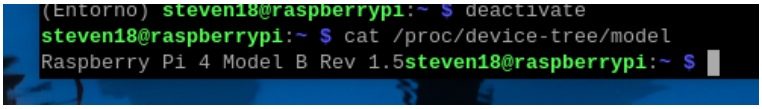
Posteriormente se procede a energizar a la placa Raspberry pi 4 para poder ingresar a su entorno inicial y configurar los datos del usuario requeridos para utilizarla de manera remota con el uso del programa VNC Viewer e instalar las librerías que se requieren para la correcta ejecución del código usado en este documento.

Entorno inicial de la Raspberry pi 4.

Anexo B. Revisión de las características de la Raspberry pi 4 modelo B

Anexo B.1. Versión de Raspberry pi

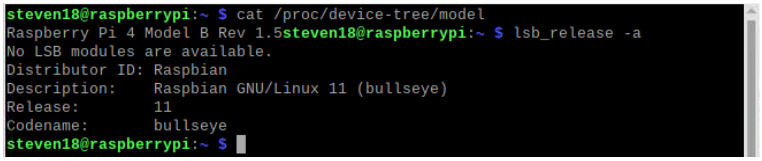
Con el comando que se visualiza en la siguiente figura, se puede verificar la versión y modelo de la Raspberry pi con el fin de determinar que comando y bibliotecas tenemos disponibles para la ejecución del código que se emplea en este proyecto.



Verificación de la versión de la Raspberry pi 4.

Anexo B.2 Versión del sistema operativo

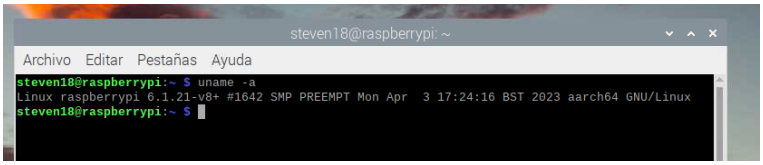
Luego de ejecutar el comando anterior se verifica la versión del sistema operativo y sus características correspondientes para tener más información a cuáles librerías y opciones de las cuales dispone Raspian tenemos al alcance para poder desarrollar el proyecto.



Verificación del Sistema operativo de la Raspberry pi 4.

Anexo B.3 Versión de Kernel

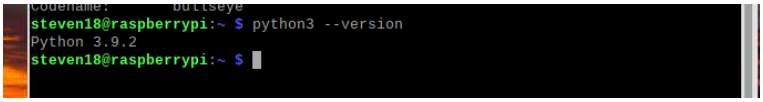
El kernel es una parte importante del sistema operativo ya que realiza la interacción entre el hardware y el software, ante la gestión de recursos como, el CPU, los dispositivos de entrada y salida que en la implementación de este proyecto será de mucha ayuda y con la memoria. De manera que ejecutando la siguiente línea de código en el cmd de nuestra placa Raspberry pi 4 podremos obtener datos acerca del Kernel.



Verificación de la versión de Kernel en la placa Raspberry pi 4.

Anexo B.4 Versión de Python instalada

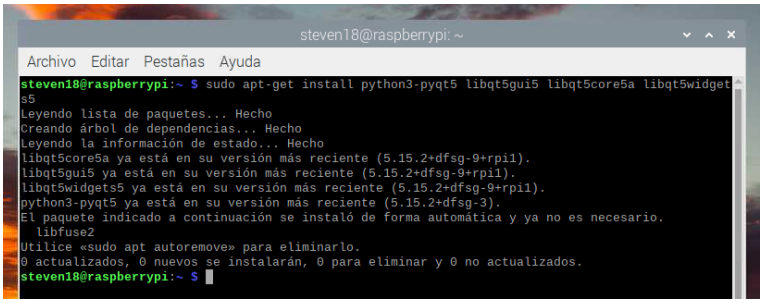
Otro aspecto importante es la versión de Python que está instalada por defecto en la placa, ya que, dependiendo de cada uno de los sistemas operativos, vienen instaladas diferentes versiones y dependiendo de nuestras necesidades se debe descargar el adecuado para compilar el código requerido con las librerías y dependencias necesarias para un buen procesamiento.



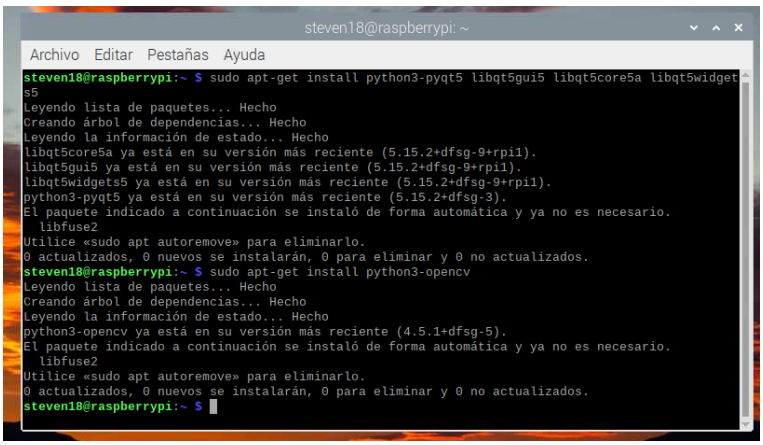
Verificación de la versión de Python instalada en el SO (Sistema Operativo).

Anexo B.5 Instalación de OpenCV

OpenCV es una librería de código abierto que es sumamente importante para el desarrollo de la propuesta ya que mediante esta herramienta se podrá realizar la adquisición de imágenes en tiempo real proveniente de una cámara compatible con la tarjeta Raspberry pi 4 como lo es la cámara Raspberry Pi camera Rev1.3, para posteriormente tratar la imagen y redimensionarla.



Instalación de dependencias necesarias para la librería Opencv.

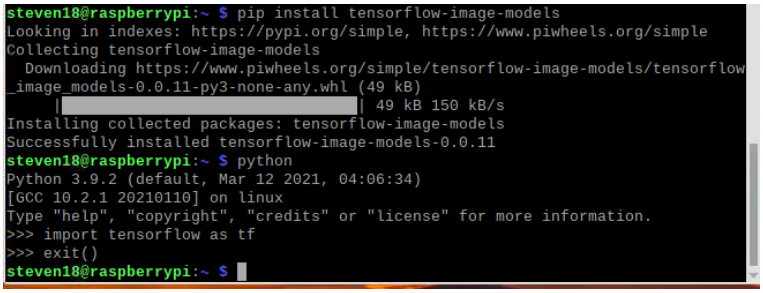


Instalación de la librería Opencv en la placa Raspberry pi 4.

Verificación de la librería Opencv.

Anexo B.6 Instalación de tflite-runtime

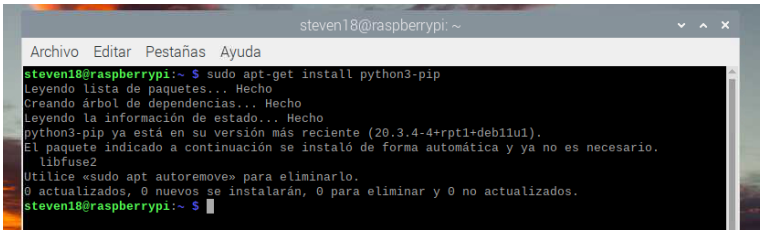
Una vez instalada las versiones y dependencias compatibles de "OpenCV", se procede a instalar una de las variantes de las versiones de "TensorFlow", debido a que la herramienta original es muy pesada para nuestra tarjeta, sin embargo, "TensorFlow image models" cumple la misma función además de ser más compacta y compatible con la versión de Sistema Operativo que se tiene instalado. De modo que se compila el siguiente código para instalar sus dependencias y su librería como se muestra en la siguiente figura.



Instalación de TensorFlow image models.

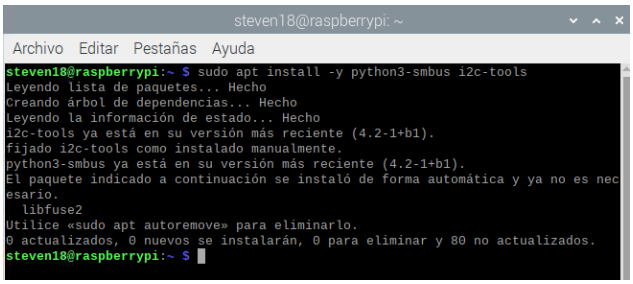
Anexo B.7 Instalación de PIP para la instalación de la librería de python3-picamera

Para la instalación de la librería de pi cámara se utiliza el siguiente comando "sudo apt – get install python3 - pip", una vez ejecutado se instalarán las dependencias necesarias que requiere esta librería, si es posible realizar el reinicio del sistema para que se instalen todas sus dependencias.

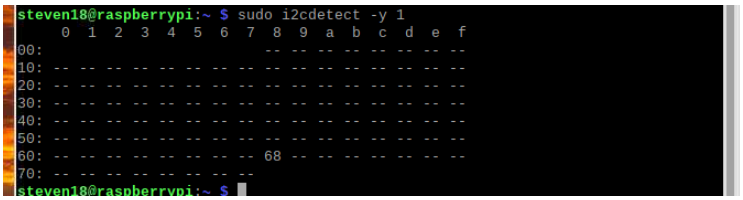


Instalación de PIP en la Raspberry Pi 4.

Anexo B.8 Instalación de librerías para MPU6050



Una vez instalada la librería para utilizar el sensor, se realiza una prueba en el propio cmd, teniendo en cuenta que el MU6050 debe estar conectado para que funcione el siguiente comando que es “sudo 12cdetect -y 1”, al ejecutar esta línea de código se presentara el numero 68 en la fila 8, esto nos indica que la tarjeta raspberry pi 4 reconoce al sensor y está recibiendo señales.



Anexo C. Comandos para la raspberry pi 4.

Anexo C.1 Dependencias necesarias para la instalación de Opencv.

Numero	Comandos a emitir en la ventana de comandos de la Raspberry Pi 4
1	sudo apt install build-essential
2	sudo apt install libjpeg-dev
3	sudo apt install libtiff5-dev
4	sudo apt install libpng-dev
5	sudo apt install libavcodec-dev sudo apt install libavformat-dev sudo apt install libswscale-dev
6	sudo apt install libv4l-dev
7	sudo apt install libxvidcore-dev
8	sudo apt install libx264-dev
9	sudo apt install libjpeg8-dev
10	sudo apt install libv4l-dev
11	sudo apt install libjasper-dev
12	sudo apt install libpng12-dev

Anexo C.2 Descripción de dependencias necesarias para la instalación de Opencv.

Numero de comando	Descripción
1	Instala las herramientas y bibliotecas esenciales para la compilación de programas desde código fuente.
2	Paquetes de desarrollo para trabajar con imágenes en formato JPEG.
3	Paquetes de desarrollo para trabajar con imágenes en formato TIFF.

4	Paquetes de desarrollo para trabajar con imágenes en formato PNG.
5	Desarrollo de paquetes para la biblioteca de códecs de audio y video.
6	Desarrollo de paquetes para la biblioteca de códecs de audio y video V4L.
7	Desarrollo de paquetes para la biblioteca de códecs de video Xvid.
8	Desarrollo de paquetes para la biblioteca de códecs de video x264.
9	Paquetes de Desarrollo para trabajar con imágenes JPEG con versión 8.
10	Desarrollo de paquetes para la biblioteca de códecs de audio y video V4L, de la versión 8.
11	Desarrollo de paquetes para la biblioteca de procesamiento de imágenes Jasper, de versión 12.
12	Paquetes de Desarrollo para trabajar con imágenes PNG, de versión 12.

Anexo D. Librerías usadas en Google Colab

Anexo D.1 Librerías utilizadas en Google Colab para el procesamiento de imágenes y el entrenamiento del modelo.

Numero	Librerías utilizadas en proyecto
1	import numpy as np
2	import Matplotlib.pyplot
3	From PIL import Image
4	import os
5	import cv2
6	from google.colab import files
7	import zipfile
8	from __future__ import absolute_import, division, print_function, unicode_literals
9	Import tensorflow as tf
10	From tensorflow import keras
11	Import tensorflow.keras.optimizers as Optimizer

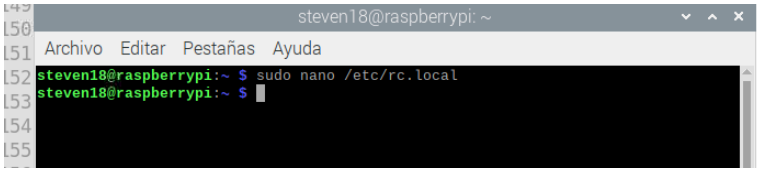
Anexo D.2 Descripción de las librerías utilizadas en Google Colab para el procesamiento de imágenes y el entrenamiento del modelo.

Numero	Descripción
1	Librería para el proceso de computación científica de Python, proporcionando arreglos multidimensionales y funciones matemáticas de alto nivel.
2	Librería de visualización 2D para Python
3	Librería de imágenes de Python para manipular y guardar varios formatos de imágenes.
4	Proporciona una manera fácil de usar funcionalidades dependientes de sistema operativo.
5	Es una Librería de Visión por computadora que proporciona herramientas para procesar imágenes y video.

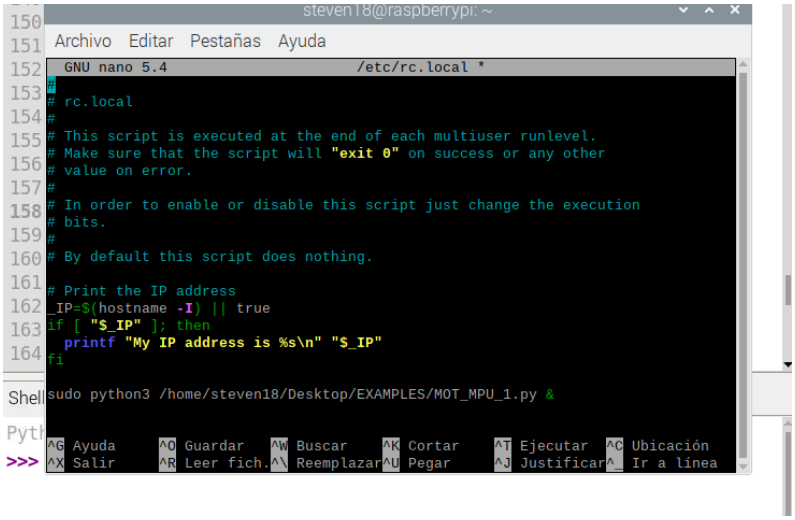
6	Es un módulo que proporciona funciones para cargar y descargar archivos ZIP.
7	Permite trabajar con archivos ZIP, como comprimir y descomprimir.
8	Este módulo permite usar funciones y características de versiones futuras de Python en versiones anteriores.
9	Es una librería de código abierto para aprendizaje automático y aprendizaje profundo, además proporciona herramientas para construir y entrenar modelos de aprendizaje profundo.
10	Es una interfaz de alto nivel que permite construir y entrenar modelos de aprendizaje profundo.
11	Contiene optimizadores para ajustar modelos durante el proceso de entrenamiento.

Anexo E. Abrir un script de Python después del arranque automáticamente

Se utiliza el comando “sudo nano /etc/rc.local” para acceder al archivo “rc.local” y agregamos la dirección donde esta ubicado el script que se desea ejecutar después del arranque de la raspberry pi 4.



Una vez realizado el paso anterior se procede a guardar los cambios realizados en el archivo actual, presionando “Ctrl + O” y posteriormente con un “Enter”, se guarda en el archivo, luego con “Ctrl + X” salimos del modo de edición, y se reinicia la raspberry pi, para verificar que después del arranque se ejecuta el script con la ruta que le agregaste al archivo “rc.local”.



Anexo F. Código para la recepción de datos del MPU6050

```
import smbus
import time

# Dirección I2C del MPU-6050
MPU6050_ADDR = 0x68

# Registros del MPU-6050
PWR_MGMT_1 = 0x6B
SMPLRT_DIV = 0x19
CONFIG = 0x1A
GYRO_CONFIG = 0x1B
ACCEL_CONFIG = 0x1C
GYRO_ZOUT_H = 0x47

def MPU6050_Init():
    # Inicializa el MPU-6050
    bus.write_byte_data(MPU6050_ADDR, SMPLRT_DIV, 7)
    bus.write_byte_data(MPU6050_ADDR, PWR_MGMT_1, 1)
    bus.write_byte_data(MPU6050_ADDR, CONFIG, 0)
    bus.write_byte_data(MPU6050_ADDR, GYRO_CONFIG, 24)
    bus.write_byte_data(MPU6050_ADDR, ACCEL_CONFIG, 0)

def read_raw_data(addr):
    # Lee datos sin procesar de la dirección I2C
    high = bus.read_byte_data(MPU6050_ADDR, addr)
    low = bus.read_byte_data(MPU6050_ADDR, addr + 1)
    value = ((high << 8) | low)
    if value > 32768:
        value = value - 65536
    return value
```

```
def calibrate_gyro():
    print("Calibrando el giroscopio... Mantén el sensor quieto.")
    num_samples = 1000
    z_offset = 0
    for _ in range(num_samples):
        z_offset += read_raw_data(GYRO_ZOUT_H)
        time.sleep(0.002)
    z_offset /= num_samples
    print(f"Offset del giroscopio en Z: {z_offset}")
    return z_offset

bus = smbus.SMBus(1)  # or bus = smbus.SMBus(0) para Raspberry Pi 1
MPU6050_Init()

# Calibrar el giroscopio
z_offset = calibrate_gyro()

print("Leyendo el ángulo de giro (Yaw) del sensor MPU-6050 en el eje Z con corrección de offset")

# Inicialización de variables
gyro_z_angle = 0
previous_time = time.time()

while True:
    # Leer el dato del giroscopio en el eje Z
    gyro_z = read_raw_data(GYRO_ZOUT_H)

    # Aplicar la corrección de offset
    gyro_z -= z_offset

    # Convertir a velocidad angular en grados/segundo
```

```
Gz = gyro_z / 131.0

# Calcular el tiempo transcurrido
current_time = time.time()
elapsed_time = current_time - previous_time
previous_time = current_time

# Calcular el ángulo de giro (integrando la velocidad angular)
gyro_z_angle += Gz * elapsed_time

# Imprimir el ángulo de giro
print(f"Ángulo de giro en el eje Z (Yaw): {gyro_z_angle:.2f}°")

# Detectar giros de 90 grados
if gyro_z_angle >= 90:
    print("Giro de 90 grados a la derecha detectado")
    gyro_z_angle = 0  # Reiniciar el ángulo
elif gyro_z_angle <= -90:
    print("Giro de 90 grados a la izquierda detectado")
    gyro_z_angle = 0  # Reiniciar el ángulo

time.sleep(0.1)
```

Anexo G. Código del movimiento de motorreductores controlado por el MPU6050

```
import RPi.GPIO as GPIO
import time
import smbus

# Configuración de los pines GPIO para los motores A, B, C y D
ENA = 12  # GPIO 12 para PWM0
IN1 = 17
IN2 = 27
ENB = 18  # GPIO 18 para PWM0
IN3 = 22
IN4 = 23
ENC = 13  # GPIO 13 para PWM1
IN5 = 24
IN6 = 25
END = 19  # GPIO 19 para PWM1
IN7 = 5
IN8 = 6

# Dirección I2C del MPU-6050
MPU6050_ADDR = 0x68

# Registros del MPU-6050
PWR_MGMT_1 = 0x6B
SMPLRT_DIV = 0x19
CONFIG = 0x1A
GYRO_CONFIG = 0x1B
ACCEL_CONFIG = 0x1C
GYRO_ZOUT_H = 0x47

# Configuración de los pines GPIO
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(ENA, GPIO.OUT)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(ENB, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)
GPIO.setup(ENC, GPIO.OUT)
GPIO.setup(IN5, GPIO.OUT)
GPIO.setup(IN6, GPIO.OUT)
GPIO.setup(END, GPIO.OUT)
GPIO.setup(IN7, GPIO.OUT)
GPIO.setup(IN8, GPIO.OUT)

# Configuración PWM
pwmA = GPIO.PWM(ENA, 1000)
pwmA.start(50)  # Reduce la velocidad al 50%
pwmB = GPIO.PWM(ENB, 1000)
pwmB.start(50)  # Reduce la velocidad al 50%
pwmC = GPIO.PWM(ENC, 1000)
pwmC.start(50)  # Reduce la velocidad al 50%
pwmD = GPIO.PWM(END, 1000)
pwmD.start(50)  # Reduce la velocidad al 50%

# Inicializar bus I2C
bus = smbus.SMBus(1)

def MPU6050_Init():
    bus.write_byte_data(MPU6050_ADDR, SMPLRT_DIV, 7)
    bus.write_byte_data(MPU6050_ADDR, PWR_MGMT_1, 1)
    bus.write_byte_data(MPU6050_ADDR, CONFIG, 0)
    bus.write_byte_data(MPU6050_ADDR, GYRO_CONFIG, 24)
```

```
    bus.write_byte_data(MPU6050_ADDR, ACCEL_CONFIG, 0)

def read_raw_data(addr):
    high = bus.read_byte_data(MPU6050_ADDR, addr)
    low = bus.read_byte_data(MPU6050_ADDR, addr + 1)
    value = ((high << 8) | low)
    if value > 32768:
        value = value - 65536
    return value

def calibrate_gyro():
    print("Calibrando el giroscopio... Mantén el sensor quieto.")
    num_samples = 1000
    z_offset = 0
    for _ in range(num_samples):
        z_offset += read_raw_data(GYRO_ZOUT_H)
        time.sleep(0.002)
    z_offset /= num_samples
    print(f"Offset del giroscopio en Z: {z_offset}")
    return z_offset

def girar_motorA_sentido_horario():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)

def girar_motorA_sentido_antihorario():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)

def detener_motorA():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
```

```
def girar_motorB_sentido_horario():
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)

def girar_motorB_sentido_antihorario():
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)

def detener_motorB():
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)

def girar_motorC_sentido_horario():
    GPIO.output(IN5, GPIO.HIGH)
    GPIO.output(IN6, GPIO.LOW)

def girar_motorC_sentido_antihorario():
    GPIO.output(IN5, GPIO.LOW)
    GPIO.output(IN6, GPIO.HIGH)

def detener_motorC():
    GPIO.output(IN5, GPIO.LOW)
    GPIO.output(IN6, GPIO.LOW)

def girar_motorD_sentido_horario():
    GPIO.output(IN7, GPIO.HIGH)
    GPIO.output(IN8, GPIO.LOW)

def girar_motorD_sentido_antihorario():
    GPIO.output(IN7, GPIO.LOW)
    GPIO.output(IN8, GPIO.HIGH)
```

```
def detener_motorD():
    GPIO.output(IN7, GPIO.LOW)
    GPIO.output(IN8, GPIO.LOW)

# Inicialización del MPU-6050 y calibración del giroscopio
MPU6050_Init()
z_offset = calibrate_gyro()

# Mover hacia adelante durante 3 segundos
girar_motorA_sentido_horario()
girar_motorB_sentido_horario()
girar_motorC_sentido_horario()
girar_motorD_sentido_horario()
time.sleep(3)

# Detener los motores
detener_motorA()
detener_motorB()
detener_motorC()
detener_motorD()
time.sleep(0.5)

# Variables para el ángulo y el tiempo
gyro_z_angle = 0
previous_time = time.time()

# Iniciar giro de 10 grados a la izquierda
girar_motorA_sentido_antihorario()
girar_motorC_sentido_antihorario()
girar_motorB_sentido_horario()
girar_motorD_sentido_horario()
```

```
try:
    while True:
        # Leer el dato del giroscopio en el eje Z
        gyro_z = read_raw_data(GYRO_ZOUT_H)
        gyro_z -= z_offset
        Gz = gyro_z / 131.0

        current_time = time.time()
        elapsed_time = current_time - previous_time
        previous_time = current_time

        # Calcular el ángulo de giro
        gyro_z_angle += Gz * elapsed_time

        if gyro_z_angle <= -10:
            print("Giro de 10 grados a la izquierda detectado")
            detener_motorA()
            detener_motorB()
            detener_motorC()
            detener_motorD()
            break

        time.sleep(0.1)
finally:
    # Detener los PWM y limpiar GPIO
    pwmA.stop()
    pwmB.stop()
    pwmC.stop()
    pwmD.stop()
    GPIO.cleanup()
```

Anexo H. Código de captura y reducción de imágenes para el entrenamiento de la neurona.

```
import numpy as np
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2
import time
import os

# Crear una carpeta para guardar las imágenes capturadas
carpeta_imagenes = "SANAS"
os.makedirs(carpeta_imagenes, exist_ok=True)

# Iniciar la cámara de la Raspberry Pi
camera = PiCamera()
camera.resolution = (208, 208)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(208, 208))

# Esperar a que la cámara se inicie
time.sleep(0.1)

for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
    # Obtener la matriz de la imagen desde el objeto de captura
    imagen = frame.array

    # Mostrar la imagen capturada
    cv2.imshow("Plantas Detector", imagen)

    # Capturar imagen si se presiona la tecla 's'
    if cv2.waitKey(1) & 0xFF == ord('s'):
```

```
        timestamp = time.strftime("%Y%m%d-%H%M%S")
        filename = os.path.join(carpeta_imagenes, f"captura_{timestamp}.jpg")

        # Reducir tamaño de la imagen a 150x150 píxeles
        imagen_reducida = cv2.resize(imagen, (150, 150))  # Ajusta el tamaño según lo que consideres necesario
        cv2.imwrite(filename, imagen_reducida)
        print(f"Imagen guardada en: {filename}")

    # Limpiar el búfer para el siguiente frame
    rawCapture.truncate(0)

    # Romper el bucle si se presiona la tecla 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Cerrar la ventana y detener la captura
cv2.destroyAllWindows()
camera.close()
```

Anexo I. Código para el procesamiento de las imágenes en Google Colab

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import os
import cv2

categorias = []
imagenes = []

from google.colab import files
Archivos = files.upload()

import zipfile

lechuga_zip = 'Data_set.zip'

directorio_extract = '/content/Data_set'

with zipfile.ZipFile(lechuga_zip,'r') as zip_ref:
  zip_ref.extractall(directorio_extract)

categorias = [carpeta for carpeta in os.listdir(directorio_extract) if os.path.isdir(os.path.join(directorio_extract, carpeta))]

print("Carpetas existentes en el directorio:")
for carpeta in categorias:
    print(carpeta)

print(categorias)

ruta_p_sanas = "/content/Data_set/SANAS"
```

```
SANAS=[]

img_size =150

for img in os.listdir(ruta_p_sanas):
    img = cv2.imread(os.path.join(ruta_p_sanas,img))
    img_resize = cv2.resize(img,(img_size,img_size))
    SANAS.append(img_resize)

    print(len(SANAS))
    print(SANAS)

SANAS = np.array(SANAS)
print(SANAS.shape)

print(SANAS[4].shape)
plt.figure()

plt.imshow(np.squeeze(SANAS[4]))
plt.colorbar()
plt.grid(False)
plt.show()

ruta_p_enfermas = "/content/Data_set/ENFERMAS"
ENFERMAST=[]

img_size =150

for img in os.listdir(ruta_p_enfermas):
    img = cv2.imread(os.path.join(ruta_p_enfermas,img))
    img_resize = cv2.resize(img,(img_size,img_size))
    ENFERMAST.append(img_resize)
```

```
    print(len(ENFERMAST))
    print(ENFERMAST)

ENFERMAST = np.array(ENFERMAST)
print(ENFERMAST.shape)

print(ENFERMAST[1].shape)
plt.figure()

plt.imshow(np.squeeze(ENFERMAST[1]))
plt.colorbar()
plt.grid(False)
plt.show()

images = np.concatenate([SANAS,ENFERMAST])
print(len(images))
Images = np.array(images)
print(Images.shape)

etiquetas_SANAS = np.repeat(0,90)
print(len(etiquetas_SANAS))
print(etiquetas_SANAS)

etiquetas_ENFERMAST = np.repeat(1,114)
print(len(etiquetas_ENFERMAST))
print(etiquetas_ENFERMAST)

#nombres de clases para SANAS = 0 y para enfermas = 1
class_names = ['SANAS','ENFERMAS']

labels = np.concatenate([etiquetas_SANAS,etiquetas_ENFERMAST])
```

```
print(len(labels))
print(labels)
labels = np.array(labels)
print(labels.shape)

plt.figure(figsize=(15,15))
for i in range(204):
  plt.subplot(30,20,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.grid(False)
  plt.imshow(Images[i])
  plt.xlabel(class_names[labels[i]])
plt.tight_layout()
plt.show()
```

Anexo J. Código de entrenamiento del modelo con la librería Tensor Flow para el perceptrón multicapa.

```
from __future__ import absolute_import, division, print_function, unicode_literals

variable_name = ""
#try:
#  %tensorflow_version 2.x
#  except Exception:
#    pass

from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import tensorflow.keras.optimizers as Optimizer

import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(150,150,3)),
    keras.layers.Dense(128, activation='relu'),

    keras.layers.Dense(2, activation='softmax'),

])

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

```
#numero de epocas
model.fit(Images, labels, epochs=200)
trained=model.fit(Images, labels, epochs=200)
```

Anexo K. Código de entrenamiento del modelo con la librería Tensor Flow para la red neuronal convolucional.

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split

# Asumiendo que ya se tiene las imágenes y etiquetas como arrays NumPy
# SANAS, ENFERMAS, y SIN_PLANTA deberían ser arrays NumPy con tus imágenes correspondientes
images = np.concatenate([SANAS, ENFERMAST, SIN_PLANTA])
labels = np.concatenate([np.repeat(0, len(SANAS)), np.repeat(1, len(ENFERMAST)), np.repeat(2, len(SIN_PLANTA))])

# Convertir a arrays NumPy
Images = np.array(images)
labels = np.array(labels)

# Dividir los datos en entrenamiento y validación
train_images, val_images, train_labels, val_labels = train_test_split(
    Images, labels, test_size=0.2, stratify=labels, random_state=42)

# Aumento de datos
datagen = ImageDataGenerator(
    rescale=1./255,  # Normalización de las imágenes
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
```

```
    fill_mode='nearest'
)

# Generadores de datos
train_generator = datagen.flow(train_images, train_labels, batch_size=32)
validation_generator = datagen.flow(val_images, val_labels, batch_size=32)

# Definición del modelo CNN
model = models.Sequential()

# Capas convolucionales y de pooling
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# Aplanamiento y capas densas
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(3, activation='softmax'))  # Cambiado a 3 para las tres clases

# Compilación del modelo
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Resumen del modelo
```

```
model.summary()

# Entrenamiento del modelo
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_images) // 32,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=len(val_images) // 32
)

# Evaluar el modelo en el conjunto de validación
val_loss, val_acc = model.evaluate(validation_generator)
print(f"Validation accuracy: {val_acc}")
```

Anexo L. Código de prueba para los modelos entrenados en Google colab

```
img = Images[3]
print(img.shape)
img = (np.expand_dims(img,0))
print(img.shape)

plt.figure()
plt.imshow(Images[3])
plt.colorbar()
plt.grid(False)
plt.show()

predictions_single = model.predict(img)
print(predictions_single)
print(np.sum(predictions_single))
print(np.argmax(predictions_single))
print(class_names[np.argmax(predictions_single)])
```

Anexo M. Código para guardar el modelo entrenado y convertirlo de Tensorflow a tf

```
model.save('modelo_entrenado2.h5')
from google.colab import files

files.download('modelo_entrenado2.h5')

import tensorflow as tf

# Ruta al modelo entrenado en formato .h5
modelo_h5 = "/content/drive/MyDrive/UPSE/TESIS/modelo_entrenado2.h5"

# Cargar el modelo entrenado
modelo = tf.keras.models.load_model(modelo_h5)

# Convertir el modelo a formato TensorFlow Lite (.tflite)
convertidor = tf.lite.TFLiteConverter.from_keras_model(modelo)
modelo_tflite = convertidor.convert()

# Guardar el modelo TensorFlow Lite en un archivo .tflite
with open("modelo_entrenado2.tflite", "wb") as archivo:
    archivo.write(modelo_tflite)
```

Anexo N. Código para la detección y movimiento del rover autónomo.

```
import RPi.GPIO as GPIO
import time
import smbus
import numpy as np
import tflite_runtime.interpreter as tflite
from picamera.array import PiRGBArray
from picamera import PiCamera
import cv2

# Configuración de los pines GPIO para los motores A, B, C y D
ENA = 12  # GPIO 12 para PWM0
IN1 = 17
IN2 = 27
ENB = 18  # GPIO 18 para PWM0
IN3 = 22
IN4 = 23
ENC = 13  # GPIO 13 para PWM1
IN5 = 24
IN6 = 25
END = 19  # GPIO 19 para PWM1
IN7 = 5
IN8 = 6

# Dirección I2C del MPU-6050
MPU6050_ADDR = 0x68

# Registros del MPU-6050
PWR_MGMT_1 = 0x6B
SMPLRT_DIV = 0x19
CONFIG = 0x1A
GYRO_CONFIG = 0x1B
```

```
ACCEL_CONFIG = 0x1C
GYRO_ZOUT_H = 0x47

# Configuración de los pines GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(ENA, GPIO.OUT)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(ENB, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)
GPIO.setup(ENC, GPIO.OUT)
GPIO.setup(IN5, GPIO.OUT)
GPIO.setup(IN6, GPIO.OUT)
GPIO.setup(END, GPIO.OUT)
GPIO.setup(IN7, GPIO.OUT)
GPIO.setup(IN8, GPIO.OUT)

# Configuración PWM
pwmA = GPIO.PWM(ENA, 1000)
pwmA.start(40)  # Reduce la velocidad al 50%
pwmB = GPIO.PWM(ENB, 1000)
pwmB.start(40)  # Reduce la velocidad al 50%
pwmC = GPIO.PWM(ENC, 1000)
pwmC.start(40)  # Reduce la velocidad al 50%
pwmD = GPIO.PWM(END, 1000)
pwmD.start(40)  # Reduce la velocidad al 50%

# Inicializar bus I2C
bus = smbus.SMBus(1)
```

```
def MPU6050_Init():
    try:
        bus.write_byte_data(MPU6050_ADDR, SMPLRT_DIV, 7)
        bus.write_byte_data(MPU6050_ADDR, PWR_MGMT_1, 1)
        bus.write_byte_data(MPU6050_ADDR, CONFIG, 0)
        bus.write_byte_data(MPU6050_ADDR, GYRO_CONFIG, 24)
        bus.write_byte_data(MPU6050_ADDR, ACCEL_CONFIG, 0)
        print("MPU6050 inicializado correctamente")
    except OSError as e:
        print(f"Error al inicializar el MPU6050: {e}")
        exit()

def read_raw_data(addr):
    high = bus.read_byte_data(MPU6050_ADDR, addr)
    low = bus.read_byte_data(MPU6050_ADDR, addr + 1)
    value = ((high << 8) | low)
    if value > 32768:
        value = value - 65536
    return value

def calibrate_gyro():
    print("Calibrando el giroscopio... Mantén el sensor quieto.")
    num_samples = 1000
    z_offset = 0
    for _ in range(num_samples):
        z_offset += read_raw_data(GYRO_ZOUT_H)
        time.sleep(0.002)
    z_offset /= num_samples
    print(f"Offset del giroscopio en Z: {z_offset}")
    return z_offset

def girar_motorA_sentido_horario():
```

```
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)

def girar_motorA_sentido_antihorario():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)

def detener_motorA():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)

def girar_motorB_sentido_horario():
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)

def girar_motorB_sentido_antihorario():
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)

def detener_motorB():
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)

def girar_motorC_sentido_horario():
    GPIO.output(IN5, GPIO.HIGH)
    GPIO.output(IN6, GPIO.LOW)

def girar_motorC_sentido_antihorario():
    GPIO.output(IN5, GPIO.LOW)
    GPIO.output(IN6, GPIO.HIGH)

def detener_motorC():
```

```
    GPIO.output(IN5, GPIO.LOW)
    GPIO.output(IN6, GPIO.LOW)

def girar_motorD_sentido_horario():
    GPIO.output(IN7, GPIO.HIGH)
    GPIO.output(IN8, GPIO.LOW)

def girar_motorD_sentido_antihorario():
    GPIO.output(IN7, GPIO.LOW)
    GPIO.output(IN8, GPIO.HIGH)

def detener_motorD():
    GPIO.output(IN7, GPIO.LOW)
    GPIO.output(IN8, GPIO.LOW)

# Inicialización del MPU-6050 y calibración del giroscopio
MPU6050_Init()
z_offset = calibrate_gyro()

# Ruta al modelo TensorFlow Lite (.tflite)
ruta_modelo_tflite = "/home/steven18/Descargas/modelo_entrenadoCNN2.tflite"

# Crear un intérprete TensorFlow Lite
interprete = tflite.Interpreter(model_path=ruta_modelo_tflite)

# Inicializar el intérprete
interprete.allocate_tensors()

# Entradas y salidas del modelo
input_tensor_index = interprete.get_input_details()[0]['index']
output_details = interprete.get_output_details()
output_tensor_index = output_details[0]['index']
```

```
output_shape = output_details[0]['shape']

# Iniciar la cámara de la Raspberry Pi
camera = PiCamera()
camera.resolution = (208, 208)  # Ajusta la resolución según las necesidades de tu modelo
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(208, 208))

# Esperar a que la cámara se inicie
time.sleep(0.1)

# Clases de tu modelo
clases = ["PLANTA SANA", "PLANTA ENFERMA", "SIN PLANTA"]

# Variables para el ángulo y el tiempo
gyro_z_angle = 0
previous_time = time.time()

# Variables de control
sin_planta_timer = 0
planta_detectada = False
start_time = None

# Función para avanzar
def avanzar():
    girar_motorA_sentido_horario()
    girar_motorB_sentido_horario()
    girar_motorC_sentido_horario()
    girar_motorD_sentido_horario()

# Función para detener
```

```
def detener():
    detener_motorA()
    detener_motorB()
    detener_motorC()
    detener_motorD()

# Función para girar a la izquierda
def girar_izquierda():
    girar_motorB_sentido_antihorario()
    girar_motorC_sentido_antihorario()
    girar_motorA_sentido_horario()
    girar_motorD_sentido_horario()

# Bucle principal
while True:
    # Activar la detección del modelo TF
    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):
        # Obtener la matriz de la imagen desde el objeto de captura
        imagen = frame.array

        # Preprocesar la imagen para la inferencia
        imagen = cv2.resize(imagen, (150, 150))  # Ajusta el tamaño según las necesidades de tu modelo
        imagen = imagen.astype(np.float32) / 255.0
        imagen = np.expand_dims(imagen, axis=0)  # Añadir una dimensión de lote

        # Cargar datos de entrada en el tensor de entrada
        interprete.set_tensor(input_tensor_index, imagen)

        # Realizar inferencia
        interprete.invoke()
```

```
        # Obtener resultados de salida
        resultados = interprete.get_tensor(output_tensor_index)
        clase_predicha = np.argmax(resultados)

        # Manejar la clasificación
        mensaje_clasificacion = clases[clase_predicha]

        # Mostrar la imagen con la clasificación
        cv2.putText(frame.array, f"Clase predicha: {mensaje_clasificacion}", (10,
30), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 255, 0), 1)
        cv2.imshow("Plantas Detector", frame.array)

        # Limpiar el búfer para el siguiente frame
        rawCapture.truncate(0)

        # Acción basada en la clasificación
        if mensaje_clasificacion == "SIN PLANTA":
            if planta_detectada:
                planta_detectada = False
                sin_planta_timer = 0
                start_time = None

            if start_time is None:
                start_time = time.time()

            avanzar()
            time.sleep(1)
            detener()
            sin_planta_timer = time.time() - start_time
            print(f"Tiempo sin planta: {sin_planta_timer:.2f} segundos")
        else:
```

```
            sin_planta_timer = 0
            planta_detectada = True
            start_time = None
            avanzar()
            time.sleep(1)
            detener()
            time.sleep(6)
            print(f"Clase predicha: {mensaje_clasificacion}")

        # Leer el dato del giroscopio en el eje Z
        gyro_z = read_raw_data(GYRO_ZOUT_H)
        gyro_z -= z_offset
        Gz = gyro_z / 131.0

        current_time = time.time()
        elapsed_time = current_time - previous_time
        previous_time = current_time

        # Calcular el ángulo de giro
        gyro_z_angle += Gz * elapsed_time

        if sin_planta_timer > 7:
            # Girar 10 grados a la izquierda si se detecta SIN PLANTA durante más de
7 segundos
            while sin_planta_timer > 7:
                girar_izquierda()
                while True:
                    # Recalibrar y reiniciar el cálculo del ángulo
                    z_offset = calibrate_gyro()
                    gyro_z_angle = 0

                    gyro_z = read_raw_data(GYRO_ZOUT_H)
```

```
            gyro_z -= z_offset
            Gz = gyro_z / 131.0

            current_time = time.time()
            elapsed_time = current_time - previous_time
            previous_time = current_time

            gyro_z_angle += Gz * elapsed_time

            if gyro_z_angle <= -10:
                print("Giro de 10 grados a la izquierda detectado")
                detener()
                break
            time.sleep(0.1)

        # Avanzar 5 segundos
        avanzar()
        time.sleep(5)
        detener()

        # Resetear ángulo y timer
        gyro_z_angle = 0
        sin_planta_timer = 0
        start_time = None

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# Romper el bucle si se presiona la tecla 'q'
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

```
# Cerrar la ventana y detener la captura
cv2.destroyAllWindows()
camera.close()

# Detener los PWM y limpiar GPIO
pwmA.stop()
pwmB.stop()
pwmC.stop()
pwmD.stop()
GPIO.cleanup()
```