UIDE
Powered by
Arizona State University

ING. MECATRÓNICA

Tesis previa a la obtención del título deIngeniero en Mecatrónica.

AUTOR: Milton Steven Garrido Egas

TUTOR: Ing. CristinaGiselle Oscullo Naranjo

IMPLEMENTACIÓN DE UN PROTOTIPO DE ROBOT MÓVIL PARA LA DETECCIÓN BIOMÉTRICA EN UNA PLANTA DE PIMIENTOS

QUITO – ECUADOR | 2023

APROBACIÓN DEL TUTOR

Yo, Cristina Giselle Oscullo Naranjo, certifico que conozco al autor del presente trabajo de Investigación "IMPLEMENTACIÓN DE UN PROTOTIPO DE ROBOT MÓVIL PARA LA DETECCIÓN BIOMÉTRICA EN UNA PLANTA DE PIMIENTOS", de Milton Steven Garrido Egas, siendo el responsable exclusivo tanto de su originalidad y autenticidad, como de su contenido.

.....

DIRECTOR DEL TRABAJO DE TITULACIÓN

Agradecimientos

Agradezco a mi familia principalmente a mis padres, por su amor incondicional, por todo el apoyo que me han brindado a lo largo de toda mi vida y su fe en mis capacidades. Este logro no es solo mío sino de ellos. Ellos me inspiran a cada día ser mejor. Y espero que me alcance la vida para darles tanto como ellos me han dado a mi. Agradezco a mis mejores amigos por brindarme su amistad y acompañarme en los buenos y malos momentos. También quisiera agradecer a mis compañeros, por su amistad y compañía en innumerables horas que trabajamos y estudiamos juntos. Agradezco a la Universidad Internacional Del Ecuador por brindarme las herramientas y oportunidades para crecer tanto académica como personalmente. Por último, pero no menos importante, agradezco a todos los que han formado parte de mi viaje académico y han contribuido, de una forma u otra

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

1. Tema

El tema del proyecto planteado es la construcción de un prototipo de robot móvil para la detección biométrica en una planta de pimientos.

2. Objetivos

2.1. General

Implementar un prototipo de robot móvil para la detección biométrica en una planta de pimientos.

2.2. Específicos

- Investigar los diferentes tipos de robots móviles.
- Realizar un dimensionamiento mecánico y electrónico, para la seleccion de componetes.
- Crear una ruta desde un punto inicial hasta la ubicación de la planta para que el robot la siga.
- Construir el prototipo de robot móvil.
- Indagar en métodos de detección de plantas.
- Ejecutar la rutina a la planta cada día o cuando el operador lo requiera.

3. Planteamiento del problema

Con el paso del tiempo la visión artificial y la inteligencia artificial se empiezan a utilizar con mayor frecuencia. Sin embargo, para tener un buen reconocimiento es necesario contar un un set de imágenes bastante amplio y dependiendo del objeto que se desea reconocer se puede encontrar o no el set buscando en internet. En el caso de los pimientos no se cuenta con esa información y dada la gran cantidad de imágenes que se requieren se vuelve un trabajo cansado y repetitivo.

4. Hipótesis

El prototipo es un robot móvil que sigue una ruta predefinida por el terreno en que se encuentra la planta hasta alcanzar su posición, una vez que llegue a la posición de la planta se activará una cámara integrada en el robot para obtener imágenes de la planta. Este proceso el robot lo realizará una vez al día o cuando un operador le de la orden de hacerlo.

5. Estudio teórico

5.1. Origen y clasificación del pimiento

El pimiento o Capsicum annuum L. es una hortaliza de la familia Solanaceae. Tiene como origen diferentes países de latinoamerica como lo son Bolivia, Peru, entre otros. Los primeros indicios del pimiento en Europa fueron gracias a Cristobal Colon quien llevaría dicha hortaliza en su primer viaje realizado en 1493. Actualmente el pimiento se ha extendido alrededor del mundo, pero su siembra y cosecha principal se encuentran en los países latinos donde luego se exporta a países en Europa, America y Asia. Dentro de la familia del pimiento existen diferentes tipos como lo son los picantes, dulces y dulce italiano. [1]

- Picante: Tienden a tener una forma alargada y fina y varia de color dependiendo de su maduración. Es una de las más cultivadas en Suramerica.
- Dulces: Este tipo de pimiento suelen ser los que se cultivan en invernadero y son de los mas consumidos en nuestro país. Tienen un tamaño grande de tres puntas o rectangular. [2]
- Dulce italiano: Tiene de igual manera una forma delgada y alargada que pueden llegar a medir hasta 35 cm y con espesor de pulpa variable. Al momento de madurar pueden tomar entre un color rojizo o amarillento. [1]

5.2. Cultivo de pimientos

El pimiento tiene un ciclo de cultivo anual y se da en los climas cálidos o en las épocas de verano y otoño, son plantas que soportan muy bien el calor pero no el frio es por esta

razón que en las otras fechas o épocas del año se las cultiva en invernadero.

En la Figura 1 [3] se muestran las diferentes etapas en el cultivo de pimientos. Se especifica que el crecimiento de la planta se obtiene en 45 días, en los siguientes 65 días se tiene la reproducción de la planta que es la etapa en la que crecen los frutos y en los últimos 70 días el pimiento tiene su maduración. [3]

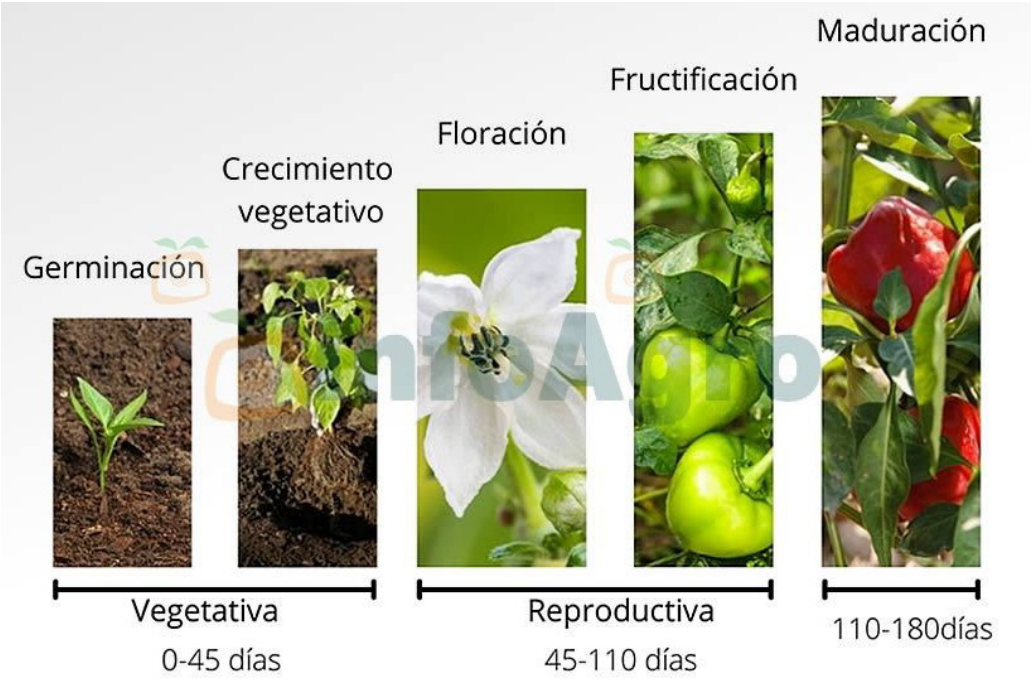


Figura 1. Etapas del ciclo de cultivo

Es necesario para un correcto cultivo de los pimientos tener en cuento los siguientes factores:

- Temperatura: El mejor rango de temperatura para el pimiento esta entre los 18 a 28 grados centígrados para un adecuado crecimiento del mismo. [2]
- Humedad: Se debe lograr una humedad relativa de entre el 50 al 70 por ciento. Cuando se supera este porcentaje se puede generar diferentes tipos de hongos lo que dificultaría la fecundacion. [2]
- Luz: En los primeros estados de reproducción y durante la floración requiere de una alta luminosidad pero se debe tener cuidado con los altos índices de radiación. [2]

5.3. Robots móviles

Los robots móviles se dividen en tres clases principales, cada una de ellas se subdivide en más pero por la naturaleza del proyecto no se hará énfasis en los robots móviles acúaticos. Las divisiones generales que se tienen de los robots móviles se encuentran en la Figura 2 [4].

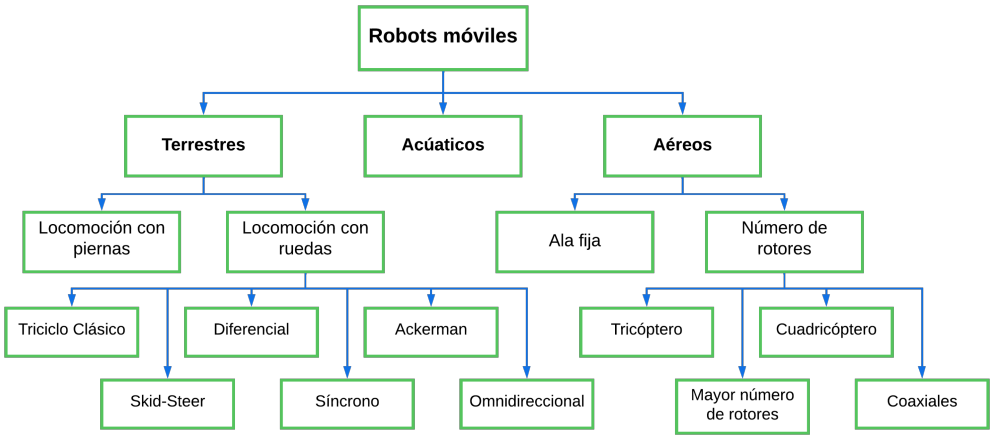


Figura 2. Tipos de robots móviles

5.4. Robots móviles terrestres con locomoción con ruedas

La configuración de un robot móvil con locomoción con ruedas hace referencia a la posición en la que están colocadas las ruedas y cuales cuentan con tracción, como se vio en la Figura 2, existen 6 configuraciones principales, a continuación se mostrará cada una de ellas con sus características de tracción y dirección y una imagen de referencia.

Triciclo clásico

Como su nombre lo indica cuenta con tres ruedas, una delantera y dos traseras como se puede ver en la Figura 3 [5], tanto la tracción como la dirección está dada por la llanta delantera, por lo que está cuenta con dos motores: uno para la tracción y un servomotor para controlar la dirección

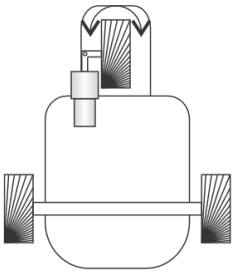


Figura 3. Configuración triciclo clásico

Diferencial

Al igual que con la configuración de triciclo clásico, la diferencial cuenta con tres ruedas, dos traseras y una delantera como se puede ver en la Figura 4 [5], la diferencia con la configuración de triciclo radica en la tracción y dirección. Tanto para la tracción como para la dirección se utilizan las dos llantas traseras, cada una de ellas cuenta con un motor propio para la tracción y el control de la dirección se lo realiza variando las direcciones y velocidades de giro de los motores.

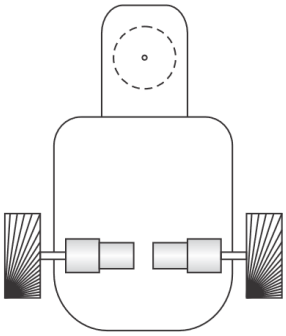


Figura 4. Configuración diferencial

Ackerman

La configuración Ackerman es la que se utiliza en la industria automotriz, cuenta con dos llantas traseras y dos llantas delanteras como se observa en la Figura 5 [5], las dos llantas trasera son las encargadas de la tracción por lo que puede constar de un único motor para las dos llantas o uno para cada llanta que se mueven de igual forma dependiendo de las necesidades, de la dirección se encarga las llantas delanteras estas giran dependiendo del lado al que se quiera direccionar, estas llantas se mueven únicamente con un solo

servomotor por lo cual cuenta con un mecanismo que mueve a las dos llantas a la vez para que estas siempre se encuentre paralelas la una de la otra.

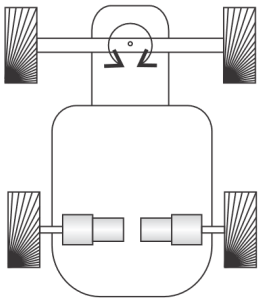


Figura 5. Configuración Ackerman

Skid-Steer

La configuración Skid-Steer es la que cuenta con el mayor número de llantas, además de la mayor fuerza de tracción ya que cada una de las llantas cuenta con su propio motor como se puede observar en la Figura 6 [5]. Esta configuración puede utilizar llantas aunque también se puede encontrar con una plataforma deslizante. Como se mencionó anteriormente cada una de las llantas cuenta con motores por lo que la tracción se encuentra en todas las llantas y para lograr el cambio de dirección los motores se agrupan, los del lado derecho y los del lado izquierdo y se los controla de igual forma que con la configuración diferencial.

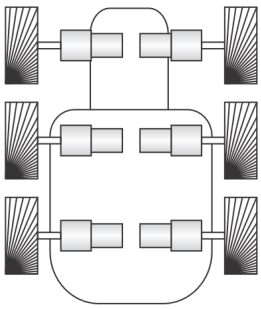


Figura 6. Configuración Skid-Steer

Sincrono

La configuración síncrona cuenta con tres o cuatro llantas como se ve en la Figura 7 [6], cada una de las llantas cuenta con un motor propio por lo que la tracción se encuentra en

todas las llantas. La peculiaridad de esta configuración radica en la dirección, pues todas las llantas se encuentra siempre de manera paralela, apuntando todas a la misma dirección, por lo que para dar dirección al robot todas las llantas se mueven al mismo tiempo, para lograr esto se suele utilizar un sistema de engranajes con correas concéntricas. Otra peculiaridad que tiene esta configuración es que el roto como tal no cambia de dirección, siempre el frente en la misma dirección, únicamente cabían las direcciones de las ruedas. [6]

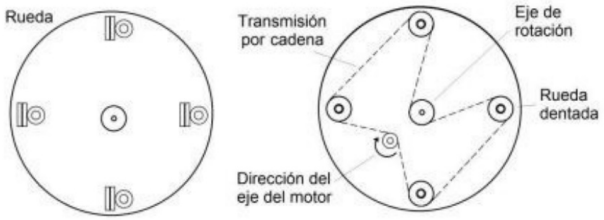


Figura 7. Configuración Sincrono

Omnidireccional

La configuración omnidireccional (Figura 8 [5]) es nombrada así por las llantas que se emplean, existen diferentes tipos de llantas omnidireccionales y la selección de estas se la realiza dependiendo del proyecto en que van a ser empleadas, pero su característica principal es que cuenta con un componente activo para proveer tracción en una dirección y componentes pasivos que proveen tracción en otras direcciones, esto da como resultado que la tracción activa en una de las llantas sea también la tracción pasiva de las demás llantas. Teniendo un control sobre lo mencionado se logra que el robot sea capaz de moverse en cualquier dirección sin tener que cambiar la dirección de las llantas. [7] [5]

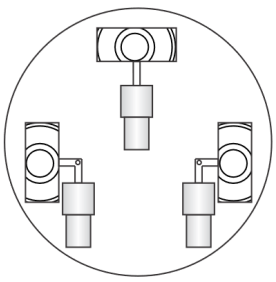


Figura 8. Configuración Omnidireccional

5.5. Requerimientos técnicos

Los requerimientos del usuario deben ser escuchados y solventados dentro del diseño conceptual, en la Tabla 1 se presenta la voz del usuario conjuntamente con la voz del ingeniero donde se establecen las principales especificaciones del prototipo.

Tabla 1. Análisis de I/O

Requerimientos	Especificaciones técnicas
Realizar la ruta cuando el operador lo requiera.	El prototipo cuenta con un sistema que permite al operador ordenar que realice la ruta.
El prototipo funciona sin necesidad de conexión a la red eléctrica, con una autonomía de dos días.	El prototipo cuenta con una batería incorporada permitiendo que se pueda movilizar sin tener una conexión a la red eléctrica.
Está preparado para subir pendientes con un ángulo de inclinación de 8,2°.	Los motores están dimensionados para que el prototipo sea capaz de subir una pendiente pronunciada.
Establecer rutas predefinidas.	La ruta que el prototipo debe seguir está predefinida en su programación.
El prototipo no interfiere con la distribucion de cultivo.	El funcionamiento del prototipo no interfiere con la normativa “Buenas Prácticas Agropecuarias” [8]

5.6. Parámetros de diseño

En base a los requerimientos técnicos, el robot móvil debe contar con las siguientes características:

- Configuración de robot debe ser capaz de atravesar terrenos irregulares.
- Los motores deben tener el torque suficiente para que el robot móvil pueda movilizarse por pendientes con normalidad.
- El robot móvil cuenta con su propia fuente de alimentación, por lo cual se puede movilizar sin conexión a la red eléctrica.
- La autonomía de la fuente de alimentación tendrá una duración mínima de dos días.
- El prototipo cuenta con una cámara que será empleada para la obtencion de imagenes de la planta.

En la Figura 9 se muestra un bosquejo del prototipo modelado en un software CAD.

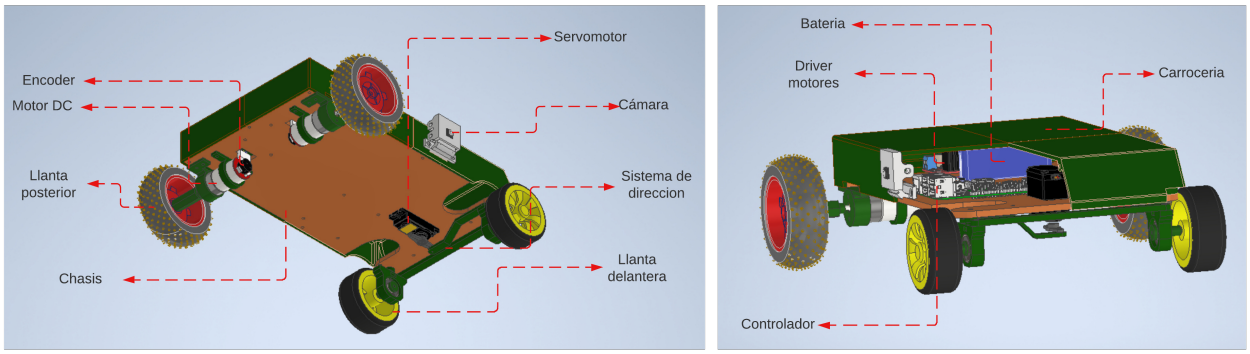


Figura 9. Diseño preliminar de robot móvil.

5.7. Diseño Electrónico

En la Figura 10 se observa el diagrama de bloques con las conecciones de los principales componentes electrónicos del prototipo, los componentes electrónicos a seleccionar son los drivers para los motores, encoder para el control de la velocidad de los motores, el microprocesador encargado del control de los motores y del procesamiento de las imágenes, regulador de voltaje para alimentar el microcontrolador, la cámara con la que se obtienen las imágenes y la batería que alimenta todo el prototipo.

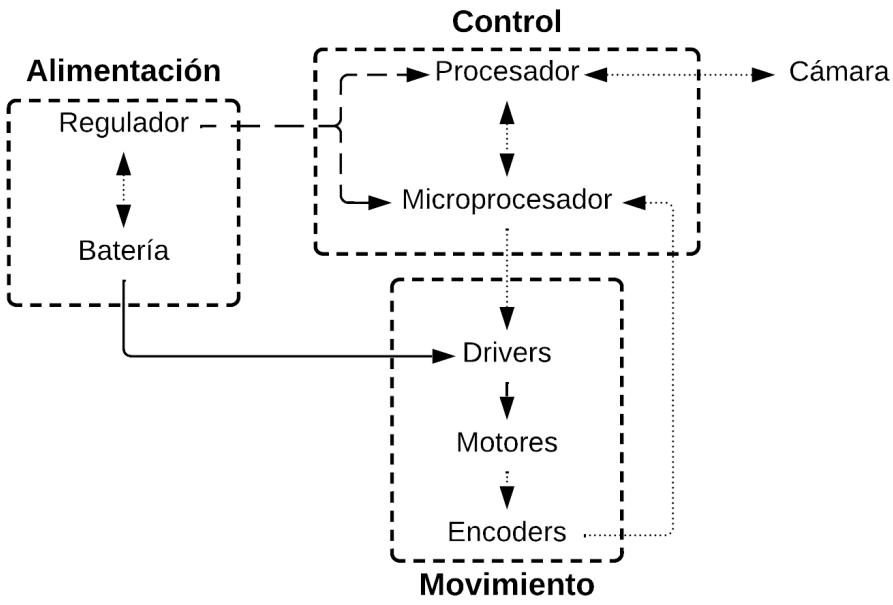


Figura 10. Diagrama de bloques electronico.

Procesador

El procesador es el encargado de la obtención de imágenes. Para realizarlo de forma automática se utilizará el lenguaje de programación python, por lo cual el procesador debe poder utilizarlo, también debe contar con pines programables para su comunicación con el microprocesador encargado de los motores. Con la descripción antes dada lo primero en que se piensa en es una Raspberry Pi ya que básicamente en un minicomputador y dado que se la puede alimentar con 5 V de DC hace que sea fácilmente transportable ideal para el proyecto. Sin embargo, no es la única elección. Existen varias alternativas entre ellas: ASUS Tinker Board S, LePotato, Rock64 Media Board, Odroid XU4, etc. Todas estas alternativas son minicomputadores en los que se puede instalar Linux y utilizar Python.

Pero se debe tener en cuenta que las opciones mencionadas no son encontradas en el país a diferencia de las Raspberry Pi y buscando precios en el exterior parecería que son más económicas las otras placa controladoras, sin embargo al momento de aumentar el coste de la importación y el tiempo que demora en llegar al país el precio se eleva demasiado, además de que no se encuentra facilmente documentación sobre estas placas controladoras.

Dicho esto la mejor alternativa para el procesador es la Raspberry Pi además de que se puede encontrar cámaras que se conectan directamente en la placa, por lo que es oportuno realizar pruebas para saber si cuenta con la potencia suficiente como para realizar el procesamiento de imágenes.

Microprocesador

Para la selección de las primeras se establecerá la forma en que se realizará el control de los motores, en los motores es necesario controlar la dirección de giro así como su velocidad. Para el control de dirección de giro se utilizará un puente H ya que este nos permite alimentar con 12V los motores, en cuanto al control de la velocidad de giro se utilizará una señal PWM. Para un mejor control en la velocidad los motores cuentan con un encoder.

Teniendo en cuenta las consideraciones anteriores, es necesario que el microcontrolador

cuenta con pines listos para trabajar con PWM, además de interrupciones para una correcta lectura de los encoders. Los principales microcontroladores que se podrían implementar son: ATtiny85, PIC16F877A, ESP32 DEVKIT, placa controladora Arduino nano.

En la Tabla 2 se obtiene que la mejor alternativa para el control de los motores es ESP32 DEVKIT.

Tabla 2. Alternativas de microprocesador

Factor	Pon.	ATtiny85		PIC16F877A		Arduino N		ESP32	
		Val		Val		Val		Val	
Cantidad de pines	0.20	4	0.8	9	1.8	8	1.6	9	1.8
Pines PWM	0.35	4	1.4	5	1.75	7	2.45	9	3.15
Facilidad de programación	0.35	6	1.5	7	1.75	9	2.25	9	2.25
Precio	0.20	9	1.8	8	1.6	8	1.6	8	1.6
TOTAL	1		5.50		6.90		7.90		8.80

En la Figura 11 [9] se muestra el pinout del ESP32 DEVKIT.

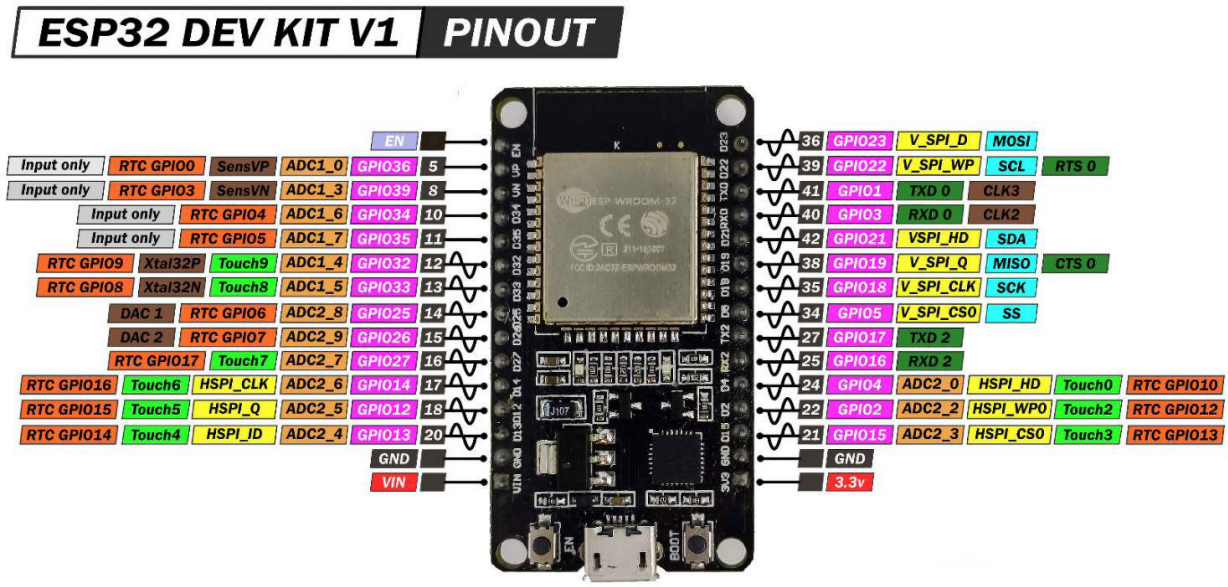


Figura 11. Pinout ESP32 DEVKIT

Drivers para motores

Para amplificar la señales desde la placa de control hacia las bobinas del motor, es necesario la utilización de drivers o controladores de los motores, estos permitirán energizar las bobinas de los motores. En el mercado se pueden encontrar varias placas con circuitos

integrados para el control de motores DC por ejemplo: TA6586, MX1508, VNH2SP30, D13-717-V14, l298n, etc.

En la Tabla 3 se observa la comparación que se realizó entre los diferentes drivers. Esto con el fin de seleccionar el adecuado. Como resultado se obtuvo que el L298N es el más adecuado para el prototipo.

Tabla 3. Alternativas de drivers

Factor	Pon.	TA6586		MX1508		VNH2SP30		D13-717-V14		L298N	
		Val		Val		Val		Val		Val	
Voltaje	0.25	9.5	2.38	5	1.25	9.5	2.38	10	2.5	9.5	2.38
Corriente	0.2	7	1.4	7	1.4	9	1.8	8.5	1.7	5	1
Refrigeración	0.15	3	0.45	3	0.45	3	0.45	3	0.45	8.5	1.28
Tamaño	0.5	8	0.8	7	0.7	7	0.7	3	0.3	5	0.5
Precio	0.3	4	1.2	9	2.7	4	1.2	4	1.2	8	2.4
TOTAL	1		6.23		6.5		6.53		6.15		7.5

En la Figura 12 [10] se muestra el pinout del módulo L298N, a continuación se explica el uso de cada uno de estos:

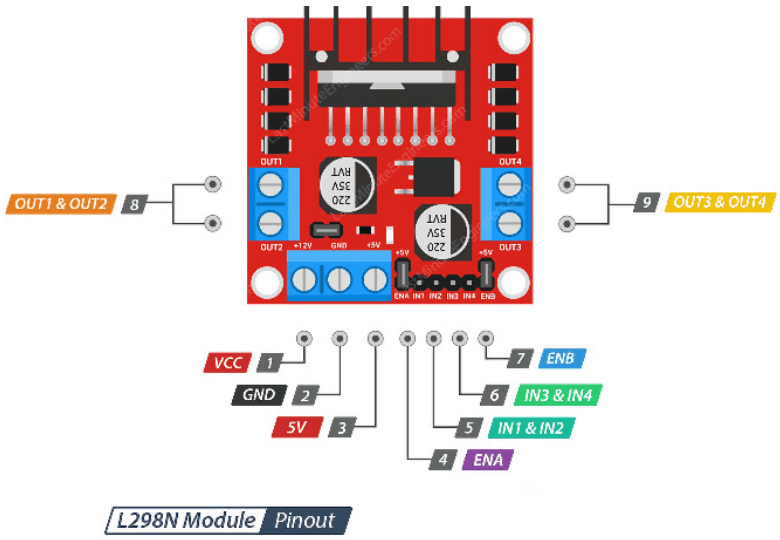


Figura 12. Pinout de módulo L298N

1. Vcc: Entrada de voltaje en un rango de 5 a 35 V. Este es el voltaje que alimenta a los motores.

2. GND: Tierra o voltaje de referencia.

3. 5V: Salida de voltaje, soporta máximo 1 A. Esta salida se puede deshabilitar si se retira el jump colocado justo encima de Vcc y GND.

4. ENA: Habilitado de Motor A. El módulo cuenta con un jump conectado a 5V, por lo que el motor siempre está habilitado, el jump es retirado para controlar la velocidad del motor con una señal PWM conectada en este pin.

5. IN1 e IN2: Control de dirección de motor A. Son dos pines que se activan uno a la vez, dependiendo de cual esté activado el giro de motor se realizará en una dirección u otra.

6. IN3 e IN4: Control de dirección de motor B. Son dos pines que se activan uno a la vez, dependiendo de cual esté activado el giro de motor se realizará en una dirección u otra.

7. ENB: Habilitado de Motor B. El módulo cuenta con un jump conectado a 5V, por lo que el motor siempre está habilitado, el jump es retirado para controlar la velocidad del motor con una señal PWM conectada en este pin.

8. OUT1 y OUT2: Salidas para motor A. El voltaje y GND se alternan en estos dos pines dependiendo de la señal se ingrese en IN1 e IN2, además de si se encuentra activado ENA.

9. OUT3 y OUT4: Salidas para motor B. El voltaje y GND se alternan en estos dos pines dependiendo de la señal se ingrese en IN3 e IN4, además de si se encuentra activado ENB.

Encoder

Los encoders son sensores que nos permiten medir tanto la dirección como el desplazamiento angular de un motor. En el prototipo es necesario conocer el desplazamiento angular de los motores para con ello conocer la distancia que ha recorrido el robot.

Por la necesidad de contar con estos sensores se busca que las alternativas de los motores ya cuenten con encoders integrados en su estructura.

En la Figura 13 [11] se muestra el pinout del encoder de cada uno de los motores, a continuación se explica el uso de estos:

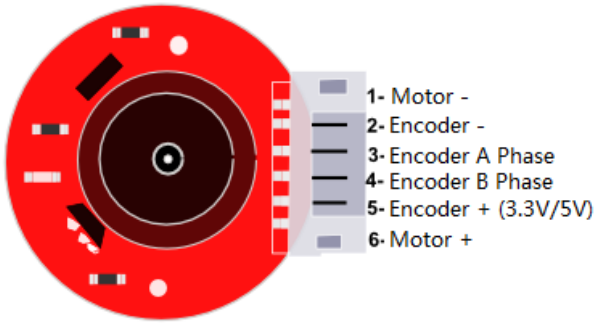


Figura 13. Pinout encoder de motor 12 V.

1. Motor -: Alimentación del motor. Pin conectado a OUT1 u OUT4, dependiendo de si es el motor A o el motor B.
2. Encoder -: GND del encoder.
3. Encoder fase A: Señal A de pulsos del encoder.
4. Encoder fase B: Señal B de pulsos del encoder. Ya que el prototipo solo requiere conocer la velocidad del motor mas no su dirección únicamente se hará uso de la fase A.
5. Encoder +: Alimentación del encoder de 3,3 V.
6. Motor +: Alimentación del motor. Pin conectado a OUT2 u OUT3, dependiendo de si es el motor A o el motor B.

Cámara

Con el uso de la Raspberry Pi como procesador, se tiene un gran número de cámaras que se conectan con la placa mediante USB, estas cámaras son principalmente webcams, sin embargo, el costo del prototipo se eleva si se adquiere este tipo de cámaras. Por lo que la mejor opción es utilizar la cámara de Raspberry que se puede ver en la Figura 14 [12].

Esta cámara se conecta directamente a la placa, además de que su costo es menor que una webcam.

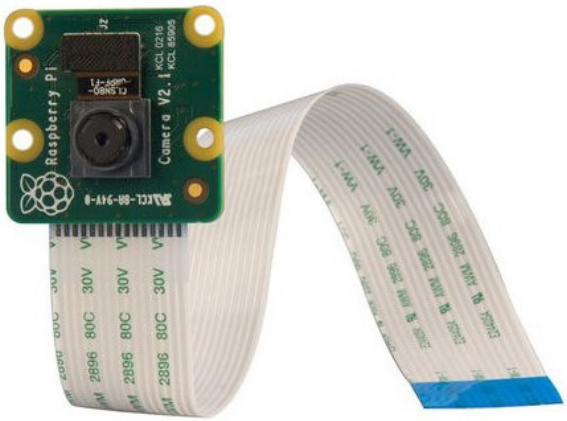


Figura 14. Cámara Raspberry Pi.

Batería

La alimentación del circuito es uno de los aspectos más importantes, ya que se debe encontrar una fuente que proporcione el voltaje y corriente requerido por los motores, además de contar con una gran capacidad para que el prototipo dure encendido por un tiempo prolongado. Asi mismo, se debe tomar en cuenta el tamaño y peso de la batería ya que una batería con dimensiones grandes proporciona una capacidad elevada pero por su peso los motores no logran mover el prototipo. Las principales baterías que se adaptan a estas necesidades son: batería de motocicleta, batería lipo 14,8V, baterías 18650.

Gracias a la Tabla 4 se deduce que la batería más adecuada para el prototipo es una batería de motocicleta por lo que se hará uso de esta (Figura 15).

Tabla 4. Alternativas de baterías

Factor	**Pon.**	**Batería motocicleta**		**Batería lipo 14.8V**		**Baterías 18650**	
		Val		Val		Val	
Voltaje	0.25	10	2.5	8	2	8	2
Corriente	0.15	10	1.5	10	1.5	9	1.35
Capacidad	0.15	8	1.2	8	1.2	9	1.35
Peso	0.20	6	1.2	8	1.6	9	1.8
Precio	0.25	8	2	7	1.75	7.5	1.87
TOTAL	1		8.4		8.05		8.38

Figura 15. Batería Nano-Gel

Regulador de voltaje

La batería escogida proporciona 12V de corriente continua, sin embargo, la Raspberry Pi utiliza 5V de corriente continua por lo que es necesario un regulador de voltaje para obtener los 5V que necesita. Por otro lado, la corriente que requiere la Raspberry Pi es superior a 1A que es la corriente máxima que suelen entregar la gran mayoría de reguladores de voltaje que se encuentran en el mercado. Por lo que en la Tabla 5 se procede a evaluar alternativas que se pueden emplear en el prototipo.

Tabla 5. Alternativas de reguladores de voltaje

Factor	**Pon.**	**Ams 1117**		**XL4015**		**LM2596S**	
		Val		Val		Val	
Corriente	0.45	3	1.35	10	4.5	8	3.6
Tamaño	0.10	7	0.7	9	0.9	9	0.9
Precio	0.45	10	1.5	6	2.7	8	3.6
TOTAL	1		6.55		8.10		8.10

Por los parámetros evaluados se obtuvo un empate entre dos reguladores, el XL4015 y LM2596S (Figura 16 [13]). Por lo cual los dos son aptos para el prototipo. Sin embargo el LM2596S es algo más económico por lo que se seleccionó esta alternativa.

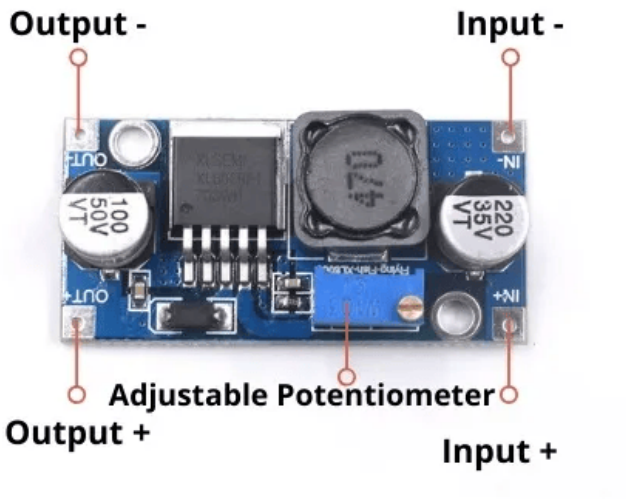


Figura 16. Pinout Lm2596S.

El LM2596S tiene una entrada de voltaje en el rango 4.5V $\leq V_{in} \leq$ 40V. Y una salida ajustable gracias a su potenciómetro, por lo que puede proporcionar los 5V requeridos, además de que el regulador soporta hasta 3A por lo que alimenta a la Raspberry Pi sin preocupación.

5.8. Diseño Mecánico

Alternativas de configuración de robot móvil

Anteriormente se especificaron las diferentes configuraciones con las que se cuenta tanto en el caso de los robots móviles aéreos como terrestres. Teniendo eso en cuenta se procederá a descartar algunas de estas alternativas por las dificultades que traería a futuro. Primero se descartan los robots móviles aéreos en todas sus configuraciones, un de motivo de esto es por la estabilidad de las imágenes. Si bien los drones cuentan con buena estabilidad, las plantas con las que se van a trabajar son pequeñas por lo que el robot debe moverse a poca distancia de las plantas, el viento que generan los robots móviles aéreos crea una mala calidad en la imagen por el movimiento que genera el viento. Otro motivo es la poca carga que puede llevar por lo que las imágenes que se obtengan deberán ser

transmitidas a una base lo cual también genera un elevamiento en el costo del prototipo, la limitación en la carga también crea una limitación en la capacidad de la batería la limitación en la capacidad de la batería ya que una batería con mayor capacidad también tiene un mayor peso.

En cuanto a las configuraciones de robots móviles terrestres se descartan las configuraciones: triciclo clásico, diferencial, síncrono, omnidireccional. Este descarte se lo realiza por la dificultad que tienen estas configuraciones para movilizarse por terrenos no uniformes en los que se encuentran las plantas. Descartando esas configuraciones quedan únicamente las configuraciones Ackerman y Skid-steer. De estas dos configuraciones las principales diferencias se encuentran en el control de la dirección y en la capacidad de carga que pueden llevar. Por lo cual se realizarán los cálculos pertinentes para conocer si únicamente con dos motores se puede movilizar el robot, en caso de que si se aplicara la configuración Ackerman, caso contrario se aplicará la configuración Skid-steer.

Selección de alternativas de materiales

También es necesario definir los materiales con los que se construirá el prototipo. Para ello se diferencia dos secciones: primera sección la carrocería y soportes que componen el recubrimiento de los componentes así como los soportes de motores, cámara e interruptores y segunda sección el chasis donde se sujetan todos los componentes.

Ya que las geometrías que se utilizaran en el diseño de las piezas de la sección 1 para adaptarse a los componente seleccionados en el diseño electrónico son bastante complejas, se hará uso de una impresora 3D para obtener esas geometrías, por lo cual en la Tabla se tiene como alternativas materiales con los cuales se puede imprimir en 3D. Además de que estas deben ser lo más livianas posible para que los motores hagan el menor esfuerzo posible. Además de que estas deben ser lo más livianas posible para que los motores hagan el menor esfuerzo posible.

Tabla 6. Alternativas de material sección 1

Factor	Pon.	PLA		ABS		PETG		Nailon	
		Val		Val		Val		Val	
Dureza	0.20	6	1.2	9	1.8	9	1.8	6	1.2
Resistencia al calor	0.25	6	1.5	9	2.25	9	2.25	7	1.75
Facilidad de impresión	0.20	9	1.8	6	1.2	7	1.4	4	0.8
Precio	0.35	9	3.15	7	2.45	7	2.45	5	1.75
TOTAL	1		7.65		7.7		7.9		5.5

Por otro lado en la Tabla 7 que hace referencia a las alternativas de la sección 2 se encarga de soportar el peso de todos los componentes por lo cual se tiene como alternativas materiales más resistentes aunque también se toma en cuenta el material seleccionado en la Tabla 6.

Tabla 7. Alternativas de material sección 2

Factor	Pon.	Acero		MDF		PETG	
		Val		Val		Val	
Carga	0.35	10	3.5	7	2.45	4	1.4
Resistencia al calor	0.15	7	1.05	9	1.35	9	1.35
Facilidad de impresión	0.20	3	0.6	9	1.8	10	2
Precio	0.30	6	1.8	9	2.7	5	1.5
TOTAL	1		6.95		8.3		6.25

Gracias a la Tabla 7 se tiene como resultado que el material más adecuado es el MDF por lo cual se realizará simulaciones con diferentes grosores para conocer cuál es el más adecuado.

Simulación de fuerzas

El material seleccionado para el chasis es el MDF, sin embargo, en el mercado se pueden encontrar diferentes espesores de este material, por lo cual se procederá a calcular la deflexión máxima para cada uno de los espesores y se los comparara con una simulación realizada en Autodesk Inventor para determinar el espesor adecuado.

Para calcular la deflexión máxima de la viga doblemente empotrada se usa la ecuación 1. [14]

$$Y_{max} = \frac{F \cdot L^3}{192 \cdot E \cdot I_{ST}} \tag{1}$$

Donde:

Y_{max} Deflexion máxima, en m;

F Fuerza máxima aplicada, en N;

L Longitud, en m;

E Módulo de elasticidad, en Pa;

I_{ST} Inercia de la sección transversal, en m^4;

Se procede a utilizar la ecuación para 4 diferentes espesores de MDF los datos fijos para las interacción son: la fuerza máxima aplicada $F = 23,61N$, la longitud $L = 0,223m$ y el módulo de elasticidad $E = 2,7x10^9Pa$. Dado que el espesor cambia la inercia también, en la Tabla 8 se muestran los resultados de los cálculos.

Tabla 8. Deflexión máxima con diferentes espesores de MDF

Espesor mm	**Inercia** mm^4	**Deflexión máxima** m	**deflexión máxima** mm
3	165	3,09E-03	3,09
4	392	1,30E-03	1,30
5	766	6,67E-04	0,67
6	1320	3.86E-04	0,38

Ya que esta pieza está sujeta a la carrocería en varios puntos y que las piezas de la carrocería no son flexibles como lo puede ser el MDF se espera que la deflexión máxima no supere 0,5 mm. Siendo ese el caso, la mejor opción es el espesor de 6mm. Por lo cual se procede a realizar una simulación de la pieza con los esfuerzos colocados en las posiciones adecuadas. En la Figura 17 se observan 4 zonas numeradas del 1 al 4 haciendo referencia a los anclajes de la simulación. Los puntos localizados en las zonas 1 y 2 son los puntos en los que se acoplan los soportes de motores y por ende llantas posteriores. Por otro lado las zonas 3 y 4 son los cuentos de acople a los soportes de rodamientos y por tanto las llantas frontales. En la Figura 17 también se observan 4 zonas nombradas de A a D haciendo referencia a los esfuerzos producidos por los componentes más significativos. En la zona A se encuentra el puente H L298N, en la zona B se encuentra la batería, en la zona C se encuentra la Raspberry Pi y en la zona D se encuentra la placa con el microcontrolador para el control de los motores.

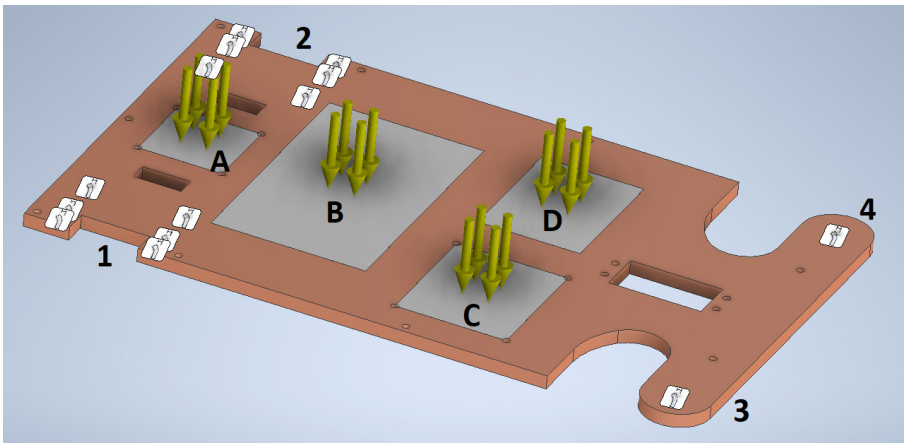


Figura 17. Esfuerzos y restricciones.

Con las restricciones y los esfuerzos colocados se procede a realizar la simulación dando como resultado la Figura 18 y gracias a ello nos podemos dar cuenta de que ya que los esfuerzos están distribuidos la deflexión máxima es de 0,1 mm.

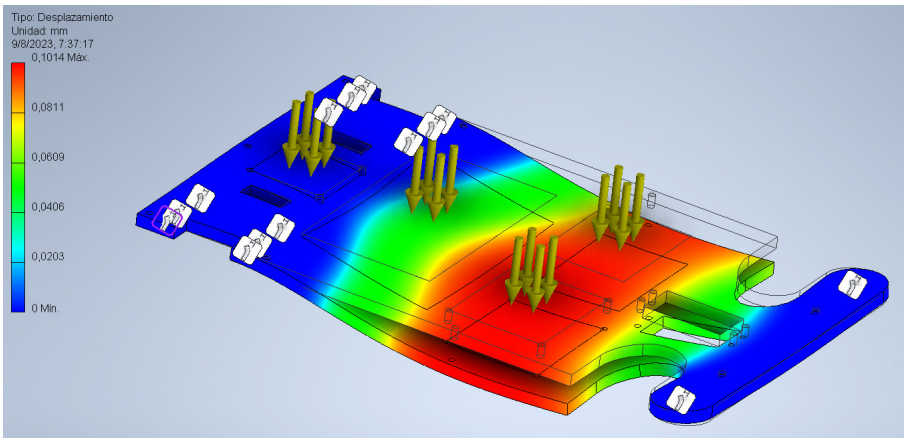


Figura 18. Simulación MDF con espesor 6mm.

Dimensionamiento de motores

Una vez determinados los componentes que se utilizaran en el prototipo se determinaron los pesos de cada uno de ellos y se realizará el cálculo para determinar el torque que se requiere en los motores para lograr movilizar el prototipo a través del terreno.

En la Tabla 9 se muestra un listado de los componentes junto a su peso en kilogramos. Esto es requerido para el cálculo de la fuerza para el movimiento del robot y el toque requerido para ello.

Tabla 9. Masa de componentes

Componente	Masa [Kg]
Módulo L298N	0,030
Motores 12V GA25-370	0,176
ESP 32	0,007
Servomotor MG996R	0,055
Rodamientos 60822	0,022
Regulador de voltaje	0,011
Bateria NANO-GEL KENDA	1,680
Raspberry Pi 4	0,05
Cámara Raspberry Pi	0,004
Interruptores	0,005
Chasis MDF	0,23
Carrocería PETG	0,178
Pernos y tuercas	0,168
Cables y placa	0,070
TOTAL	2,686

Por temas de seguridad se realiza un sobredimensionamiento de la masa del robot, haciendo que la masa de los componentes sea el 70 % de la masa con la que se realizan los cálculos dando como resultado la ecuación 2.

$$m_c = 0,7 \cdot m \tag{2}$$

Donde:

m_c masa de componentes, en kg;

m masa sobredimensionada, en kg;

Se despeja la ecuación 2 para obtener la masa sobredimensionada

$$m = \frac{m_c}{0,7}$$
$$m = \frac{2,686}{0,7}$$
$$m = 3,837 \ kg$$

Para el cálculo de la fuerza requerida es necesario establecer la aceleración máxima a que tendrá el robot. En el proyecto se establece una aceleración de 1 m/s^2. La fuerza se calcula empleando la ecuación 3. [15]

$$F = m \cdot a \tag{3}$$

Donde:

F fuerza requerida, en N;

m masa del robot, en kg;

a aceleración máxima del robot, en m/s^2;

Con la masa y la aceleración definidas se calcula la fuerza.

$$F = 3,837 \cdot 1$$

$$F = 3,837 \ \ N$$

Dado que el robot utiliza dos motores para su movimiento y que estos solo son utilizados únicamente para tracción, la fuerza que se requiere se divide entre ellos por lo tanto, cada robot debe hacer la mitad de la fuerza requerida, entonces se cumple la ecuación 4.

$$F_{izq} = F_{der} = F_{motor} = \frac{F}{2} \tag{4}$$

Donde:

F_{izq} fuerza requerida en motor izquierdo, en N;

F_{der} fuerza requerida en motor derecho, en N;

F_{motor} fuerza requerida en un motor, en N;

F fuerza requerida para mover el robot, en N;

$$F_{izq} = F_{der} = F_{motor} = \frac{3,837}{2}$$

$$F_{motor} = 1,919 \ \ N$$

Para dimensionar el motor es necesario calcular el torque utilizando la ecuación 5. [16]

$$\tau = F_{motor} \cdot r \tag{5}$$

Donde:

τ torque requerido, en Nm;

F_{motor} fuerza de motor, en N;

r radio de llanta, en m;

Sin embargo en las hojas de datos de los motores el torque se encuentra con las unidades $kgcm$, por lo cual hay que utilizar la ecuación 6 para tener el torque en esas unidades.

$$\tau_{kgcm} = \tau_{Nm} \cdot \frac{100}{9,81} \tag{6}$$

Donde:

τ_{kgcm} torque, en $kgcm$;

τ_{Nm} torque, en Nm;

La fuerza ya fue calculada previamente y en cuanto al radio, el prototipo cuenta con llantas de diametro 0,085 m. Por tanto el torque es igual a:

$$\tau = 1,919 \cdot \frac{0,085}{2}$$
$$\tau = 0,082 \ \ Nm$$

Empleando la ecuación 6 se tiene que el torque igual a:

$$\tau_{kgcm} = 0,082 \cdot \frac{100}{9,81}$$
$$\tau_{kgcm} = 0,836 \ \ kgcm$$

El toque calculado es el necesario para que el robot se mueva por una ruta sin pendientes. Sin embargo, en la ruta que debe seguir el robot existe un tramo con una pendiente de 8,2° por lo cual es necesario calcular una fuerza para superar la pendiente. Para ello se hace uso de la ecuación 7. [15]

$$F_{pendiente} = F + m \cdot g \cdot sin(\theta) \tag{7}$$

Donde:

$F_{pendiente}$ fuerza requerida para para que el robot suba la pendiente, en N;

F fuerza requerida para mover el robot sin pendiente, en N;

m masa del robot, en kg;

g gravedad, en m/s^2;

θ inclinación de la pendiente, en rad;

Con todos los datos se procede a realizar el cálculo, lo que da un resultado de:

$$F_{pendiente} = 3,837 + 3,837 \cdot 9,81 \cdot sin(0,143)$$
$$F_{pendiente} = 9,206 \ N$$

Como se explicó con anterioridad esta fuerza se divide para los dos motores con los que se cuenta, además se vuelven a aplicar las ecuaciones 4, 5 y 6 para obtener el torque que requeren los motores.

$$F_{izqpen} = F_{derpen} = F_{motpen} = F_{pendiente}/2$$
$$F_{motpen} = 4,603 \ N$$
$$\tau_{Nm} = 4,603 \cdot \frac{0,085}{2}$$
$$\tau_{Nm} = 0,196 \ Nm$$
$$\tau_{kgcm} = 0,196 \cdot \frac{100}{9,81}$$
$$\tau_{kgcm} = 1,994 \ kgcm$$

En el Apéndice A se puede observar las especificaciones de los motores de JGA25-370 los motores que se utilizaran son de 12V para no tener que transformar el voltaje de la batería. Entonces la caja reductora que debe tener el motor es de 399 rpm, ya que este cuenta con un torque de 2,2 $kgcm$ que es el inmediato superior a los 1,994 $kgcm$ que requiere el robot para moverse en la pendiente, que es el tramo que más torque requiere.

Con los cálculos se demuestra que con dos motores es más que suficiente para mover el prototipo por lo que se utilizará la configuración Ackerman.

Siendo de esta manera es necesario diseñar un mecanismo para el la dirección del robot móvil, este mecanismo tendrá su movimiento gracias a un servomotor. Las ruedas están sujetas a rodamientos para que su movimiento sea fluido, el rodamiento se encuentra en una pieza que permite estar conectado al prototipo y permite cambiar la dirección de la llanta. Las dos piezas que sostienen los rodamientos se encuentran conectadas entre ellas para lograr controlar la dirección con únicamente un servomotor, de esta manera si

se mueve una llanta la otra se mueve con ella. Por último una de las piezas que sostienen los rodamientos está conectada al servomotor. En la Figura 19 se muestra el mecanismo diseñado.

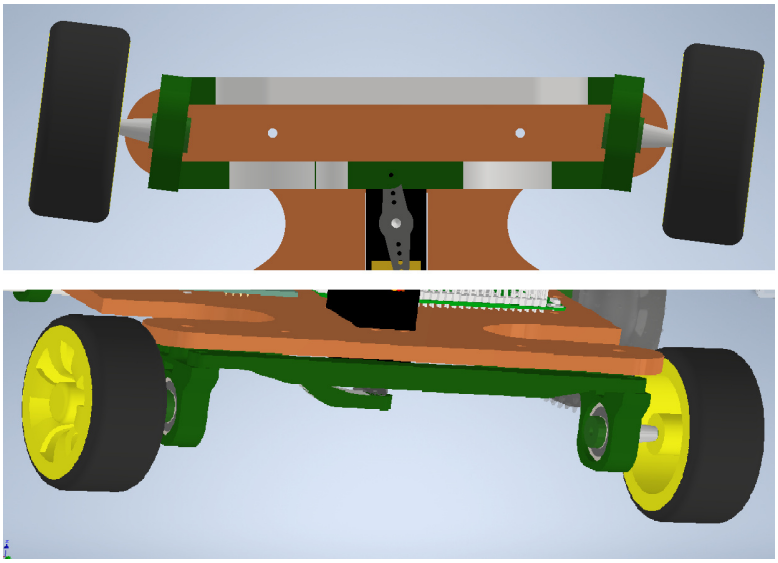


Figura 19. Diseño de direccion en Inventor

Para sostener los motores DC que generan la tracción se diseñaron dos piezas. La primera sostiene al motor en la mitad, La segunda pieza se encuentra en el inicio del motor, sujetada con pernos para que el motor quede fijo. Estas dos piezas se sujetan con pernos a la base. En la Figura 20 se puede observar las piezas que sostienen los motores.

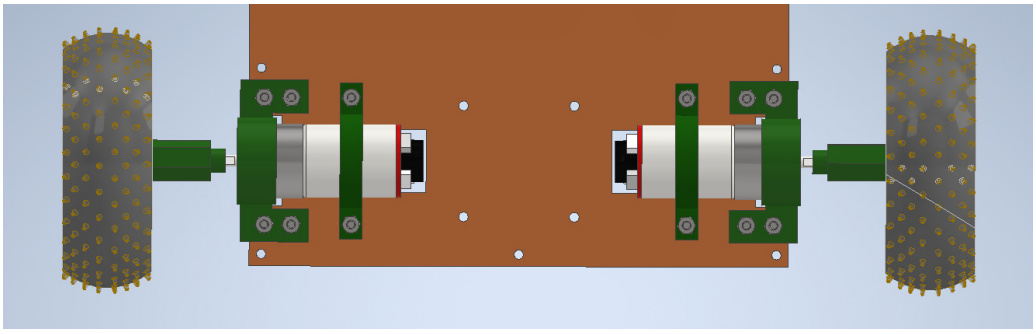


Figura 20. Soporte de motores DC

Las llantas que se utilizan para la tracción no cuentan con un acople para conectar el motor directamente a la llanta por lo que se diseñó una pieza que sirva de acople. Esta pieza entra con presión al motor y se sujeta con un perno a la llanta. En la Figura 21 se observa el acople diseñado.

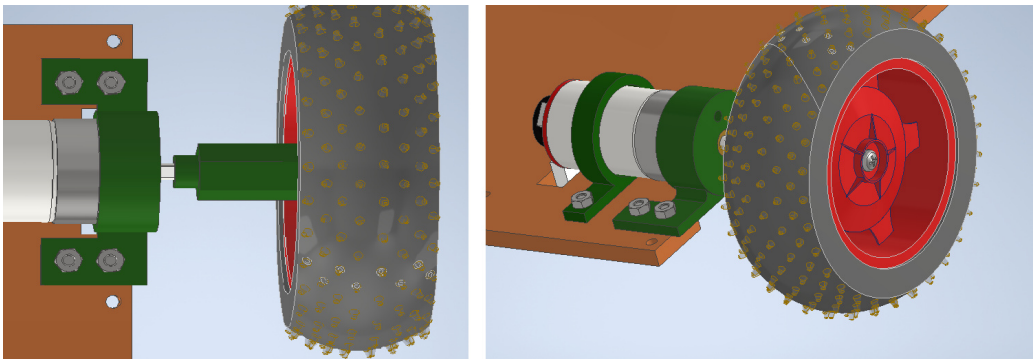


Figura 21. Acople de motor a la llanta

Todos los componentes están sujetos a una base de MDF de 6 mm de espesor, que fue diseñada con los agujeros necesarios para que los pernos que soportan todas las piezas puedan atravesar la base. En la Figura 22 se muestra el diseño de la base.

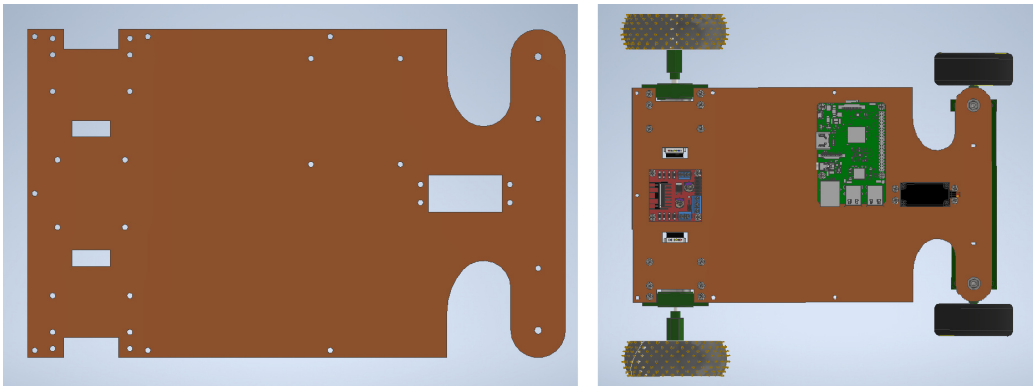


Figura 22. Diseño de base del robot.

El prototipo está cubierto por una carrocería de PETG como se observa en la Figura 23, esta fue diseñada para cubrir a todos los componentes, la carrocería está anclada a la base del prototipo con pernos. La carrocería cuenta con dos huecos en ambos lados en los cuales se colocan el soporte de la cámara y el soporte del switch de encendido y un botón. Estos huecos son idénticos por lo que la cámara se puede colocar en el lado de mayor conveniencia.

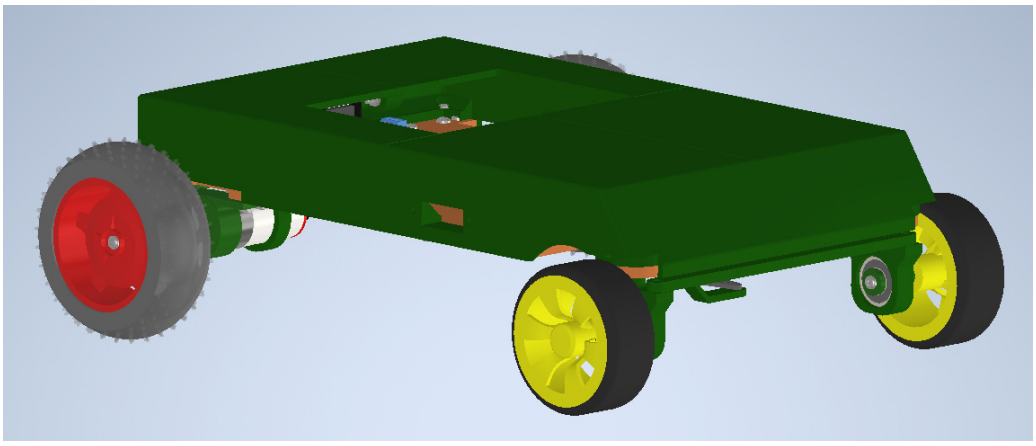


Figura 23. Carroceria de robot

La cámara se encuentra fija gracias a un soporte diseñado a medida de la cámara, este soporte se coloca en uno de los agujeros antes mencionados en la carrocería. Para el switch de encendido y el botón se tiene también un soporte que encaja en estos agujeros.

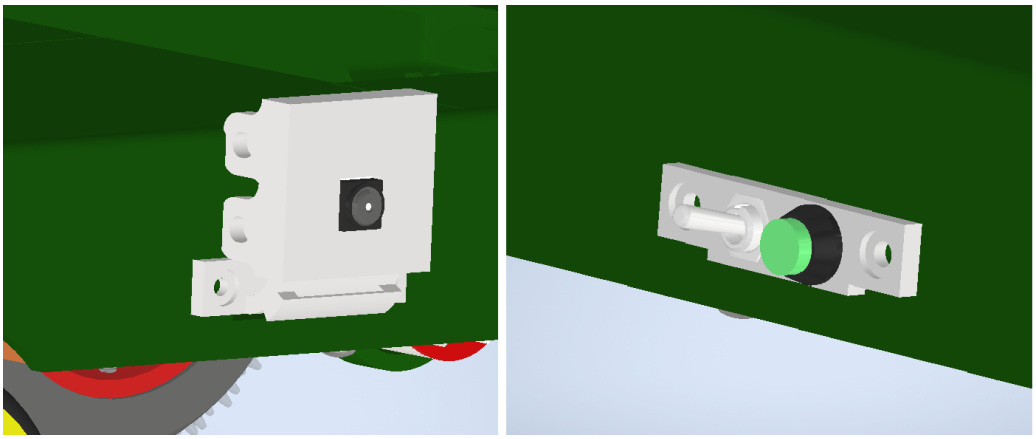


Figura 24. Soportes de cámara y switches.

Por último en la Figura 25 se observa una estructura que cubre la batería que alimenta el prototipo además de que tiene la función de cubrir todo el prototipo en caso de lluvia, de esta forma el prototipo sigue con su labor a pesar del mal clima.

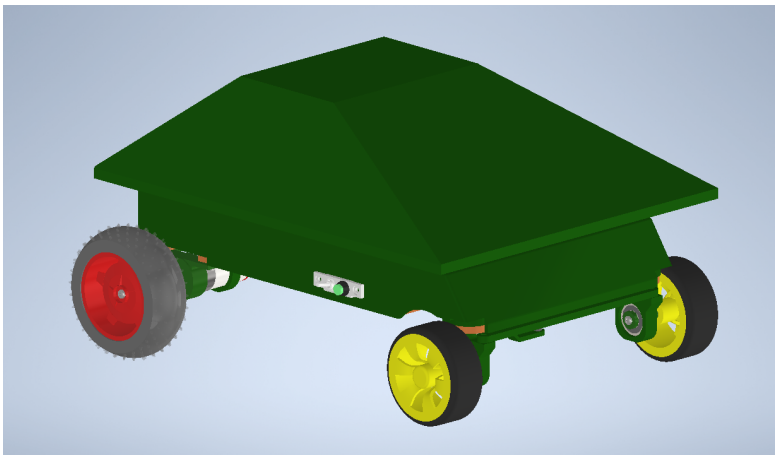


Figura 25. Estructura de prototipo final.

5.9. Desarrollo de la Programación

El desarrollo de la programación se lo dividió en tres partes principales que se listan a continuación:

1. Desarrollo del código de movilidad del robot: En este apartado se muestra el código compatible con ESP32 para poder realizar los movimientos del recorrido del robot para que cumpla con el recorrido planteado.

2. Creación del código de control de la cámara y almacenamiento de imágenes: El último apartado es el correspondiente a la incorporación de un código usando el lenguaje de programación Python para tomar las imágenes con la cámara y almacenarlas posteriormente poder utilizarlas en el entrenamiento del modelo.

3. Entrenamiento del modelo de reconocimiento:: Este apartado se centra netamente en el entrenamiento de un modelo de red neuronal convoluciones para la identificación de la planta de pimientos haciendo uso del modelo de YOLOv5.

Desarrollo del código de movilidad del robot

Como se mencionó previamente el control de los motores se lo realizará con una placa de desarrollo ESP32 el código empleado se lo puede observar en el Apéndice B pero se procede a explicar su flujo de trabajo en la Figura 26.

En primer lugar se declaran las constantes y variables que se utilizaran para el control de los motores, servomotor, encoders, pulsador. En este apartado se especifica los pines que se utilizarán así como los valores de los encoders para determinar el desplazamiento angular y los valores para señales PWM. Después los pines declarados en la parte anterior se los establece como entradas o salidas. Por último se realiza una prueba del funcionamiento de los motores.

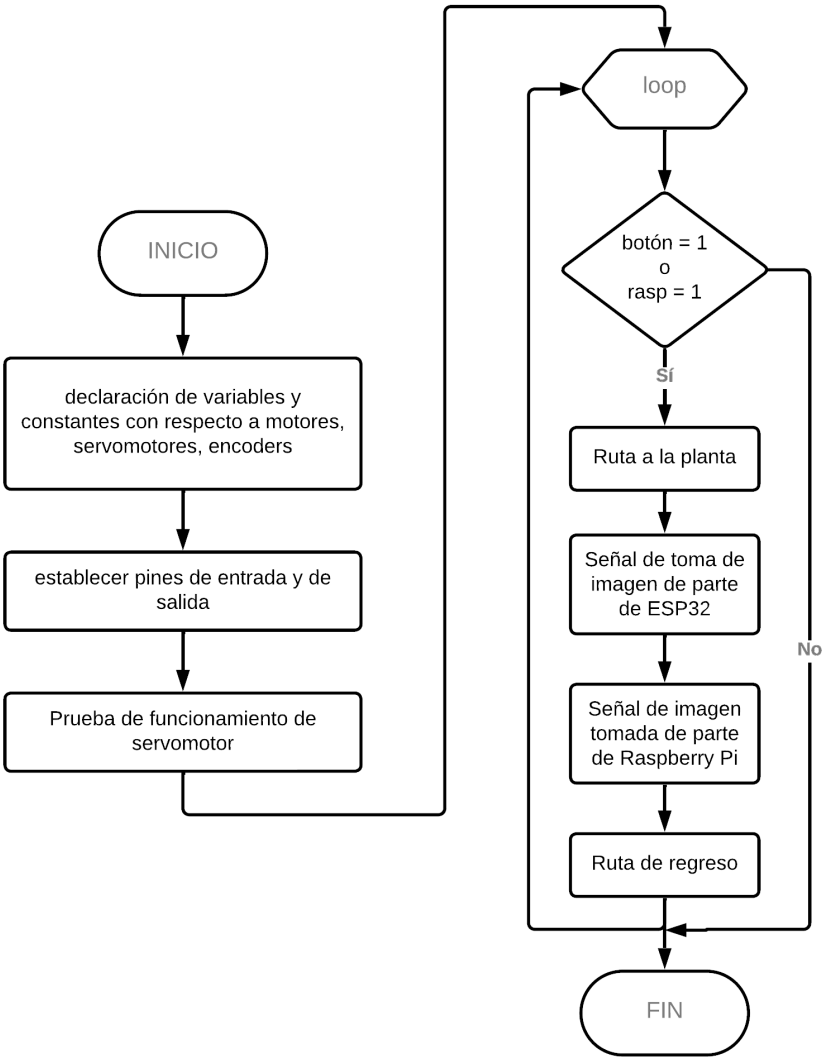


Figura 26. Diagrama de flujo control de motores.

Concluido la parte anterior empieza el bucle, en este el ESP32 DEVKIT se mantiene en espera hasta que el pulsador físico o las Raspberry Pi le den la señal de empezar el movimiento. El robot móvil avanzará hasta la locación de la planta donde enviará una señal a la Raspberry Pi para que ésta proceda a tomar la imagen y después regresar a la posición de partida.

Creación del código de control de la cámara y almacenamiento de imágenes

El código para el control de la cámara se encuentra en el Apéndice C y el flujo del código se observa en la Figura 27. Lo primero que se realiza es la importación de las librerías para el control de la cámara, el tiempo y el GPIO de la Raspberry Pi. Después se establece la forma de utilizar el GPIO, se determinan los pines que se utilizarán y si estos se utilizarán como entrada o como salida.

Dentro del bucle se tiene en espera hasta que sea la hora establecida para empezar la rutina o a que reciba una señal de parte del control de los motores para capturar una imagen y guardarla. Una vez capturada la imagen se envía una señal para que la rutina pueda continuar.

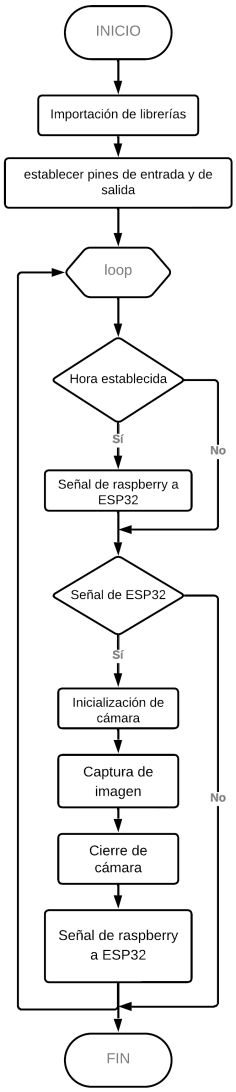


Figura 27. Diagrama de flujo control de cámara

Entrenamiento del modelo de reconocimiento

Las imágenes obtenidas gracias al robot móvil son almacenadas en la nube de google drive. Estas imágenes se utilizaran para el entrenamiento de la red para el reconocimiento de las plantas de pimiento, las imágenes se dividen en tres grupos diferentes: entrenamiento, validación y testeo, la cantidad de imágenes en cada grupo se especifica en la Tabla 10. También es necesario crear etiquetas de las imágenes, para ello se crea un documento de texto con el mismo nombre de la imagen, en el documento se enumeran las etiquetas, se colocan las coordenadas en xy del centro de un rectángulo, ancho y largo del rectángulo donde. El rectángulo debe estar ubicado de tal forma contenga el objeto que se desea reconocer.

Tabla 10. División en grupos

Grupo	Cantidad de imágenes
Entrenamiento	615
Validación	180
Testeo	41

Para el entrenamiento de las redes neuronales se puede recurrir a varios marcos de aprendizaje entre ellos: Tensorflow, Pythorch, Caffe model, Scikit-learn, YOLOv5, etc.

Habiendo definido los principales marcos de aprendizaje, se selecciona a YOLOv5 como la mejor opción, sin embargo, este no se lo puede definir como marco de aprendizaje, sino que es un conjunto de familias de redes pre entrenados lo cual facilita al entrenamiento con set de datos personalizados. ya definido eso también es necesario seleccionar qué red se utilizara. Para este proyecto se seleccionó YOLOv5x que es la red con mayor número de neuronas, lo hace que el entrenamiento se realice de mejor forma aunque también demora más en entrenarse, además de que requiere de más recursos computacionales. Pero no hay que olvidar que la velocidad de entrenamiento no es un factor determinante en este proyecto.

Dicho lo anterior y ya teniendo las imágenes, las etiquetas y haber seleccionado a YOLOv5x como red neuronal se procede a entrenar la red.

Procesamiento de nuevas imágenes.

Para el procesamiento de las nuevas imágenes se creó un programa sencillo con una interfaz gráfica (Figura 28) que solicita la ubicación del archivo resultado del entrenamiento con YOLOv5 el cual es un archivo con extensión pt, además solicita la ubicación de la carpeta donde se encuentran las imágenes a procesar y la ubcación donde se desea almacenar los resultados, también se tiene una entrada de texto en la se ingresa el porcentaje de confianza con la que se desea realizar la identificación de la planta de pimiento, este porcentaje debe estar en un rango de 0 a 100, por último se selecciona si se desea eliminar las imágenes originales. El código empleado para emplear este programa se encuentra en el Apéndice D.

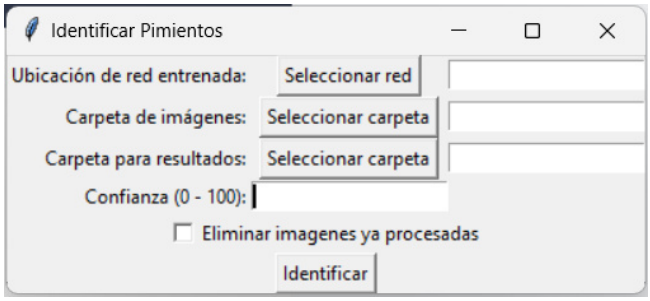


Figura 28. Interfaz gráfica.

6. Construcción

Luego de diseñadas las piezas fueron enviadas a impresión y corte dependiendo de la pieza. Ya teniendo las piezas se procede a acoplarlas, primero los motores con sus soportes y estos al chasis del robot. Ya unidas las piezas se acoplan las ruedas posteriores al motor, dando como resultado la Figura 29.

Figura 29. Soporte de motores y llanta posterior.

Después se colocan los rodamientos en sus soportes y se acoplan las llantas delanteras utilizando el eje diseñado. Los soportes de los rodamientos se unen al chasis y se coloca la unión entre ambos soportes además de la unión entre el servomotor y la dirección como se observa en la Figura 30.

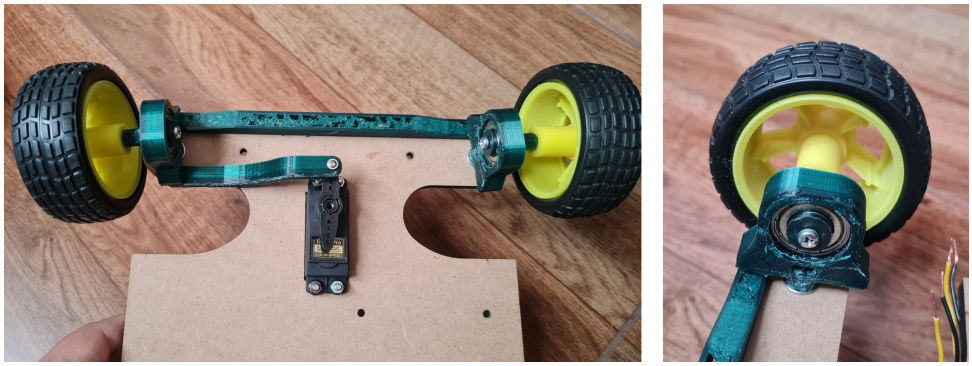


Figura 30. Dirección de robot móvil.

Posteriormente se acoplan los componentes electrónicos al chasis. Luego en la carrocería frontal se colocan el soporte para la cámara y el soporte para los interruptores como se ve en las Figuras 31 y 32.

Figura 31. Soporte interruptores.

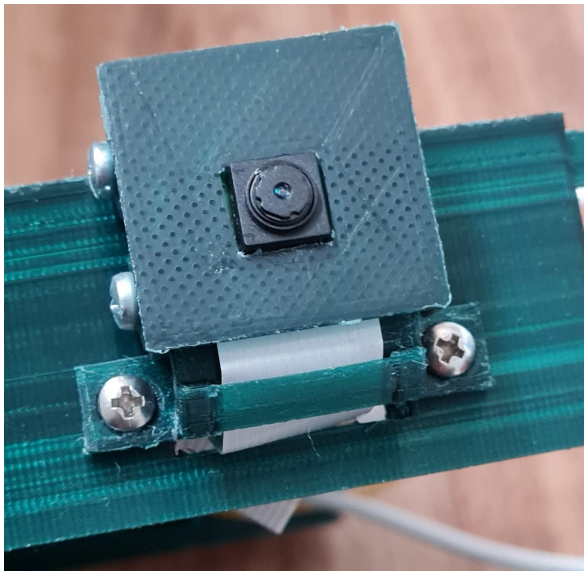


Figura 32. Soporte cámara.

Ya colocados ambos soportes se colocan las carrocerías frontales en el chasis (Figura 33).

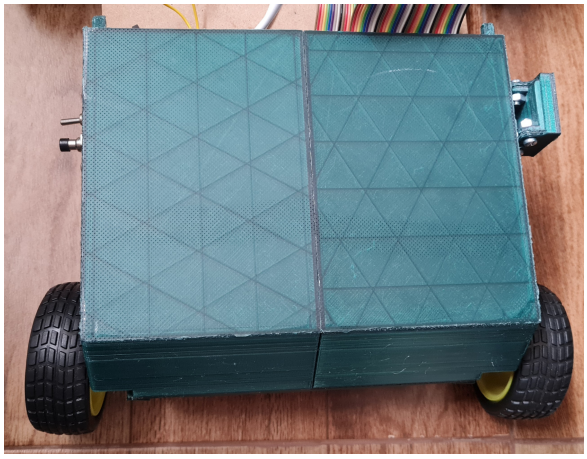


Figura 33. Carrocería frontal.

Posteriormente se colocan las carrocerías posteriores. Por último se coloca la batería y

la cobertura dando como resultado la Figura 34.

Figura 34. Robot móvil.

7. Pruebas y resultados

La ruta preprogramada para el robot cuenta con pequeños cambios tanto en la dirección como en las señales PWM que se envían al driver para controlar la potencia del motor. Estos cambios están programados de tal forma que el robot sigue una ruta en línea recta a pesar de las irregularidades del terreno. Además, por estas mismas irregularidades del terreno sumado a la inercia del robot en movimiento da como resultado que el robot siga su movimiento aunque los motores sean apagados.

A continuación en la Figura 35 se muestran el recorrido programado versus el recorrido real del robot camino a la planta, en la Figura 36 se muestra el recorrido de regreso. También se especifican los dos ciclos de trabajo empleados en la señal PWM de los motores la señal 1 con un ciclo de trabajo de 90,19 % y la señal dos con un ciclo de trabajo 74,51 %.

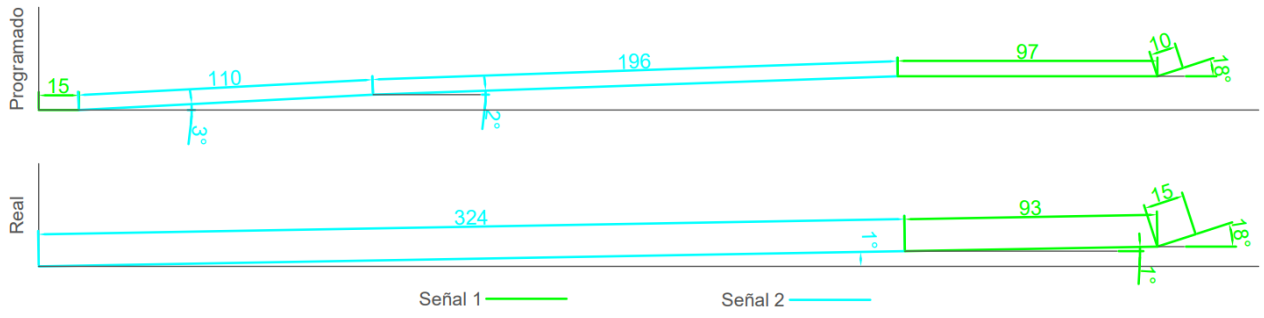


Figura 35. Desplazamiento programado vs real.

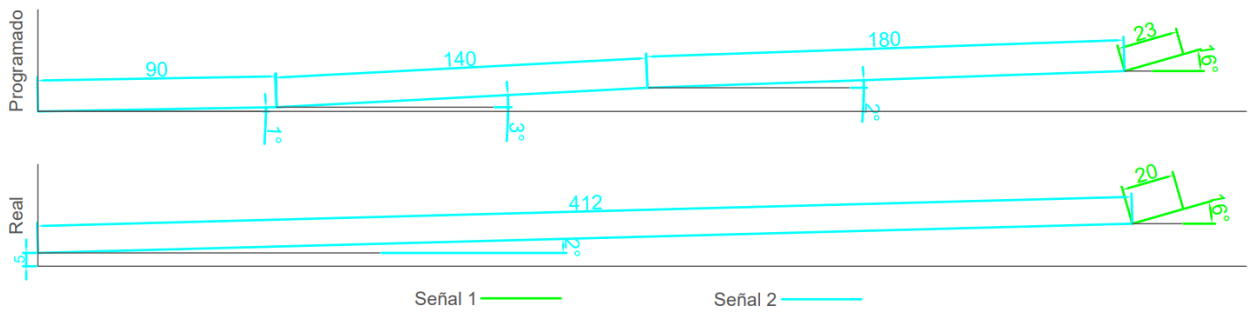


Figura 36. Desplazamiento programado vs real.

Con los desplazamientos expuestos se puede medir el error absoluto obtenido entre el los puntos objetivos y los puntos reales dando como resultado la Tabla 11 donde se especifican los puntos puntos objetivos programados y reales en cm, medidos tomando como referencia el punto inicial.

Tabla 11. Errores absolutos

	Programado		**Real**		**Error Absoluto**	
	x	y	x	y	x	y
Des. ida	427.24	15.69	432	11.98	4,76	3,71
Des. vuelta	0	0	3	5	3	5

Como se mencionó la red seleccionada es YOLOv5x con imágenes de 640 pixeles, lotes de 17 y 200 épocas, aunque cabe recalcar que el entrenamiento se detuvo en la época 177 ya que no hubo mejoría en el entrenamiento por 100 épocas. Con ello el entrenamiento dio los siguientes resultados.

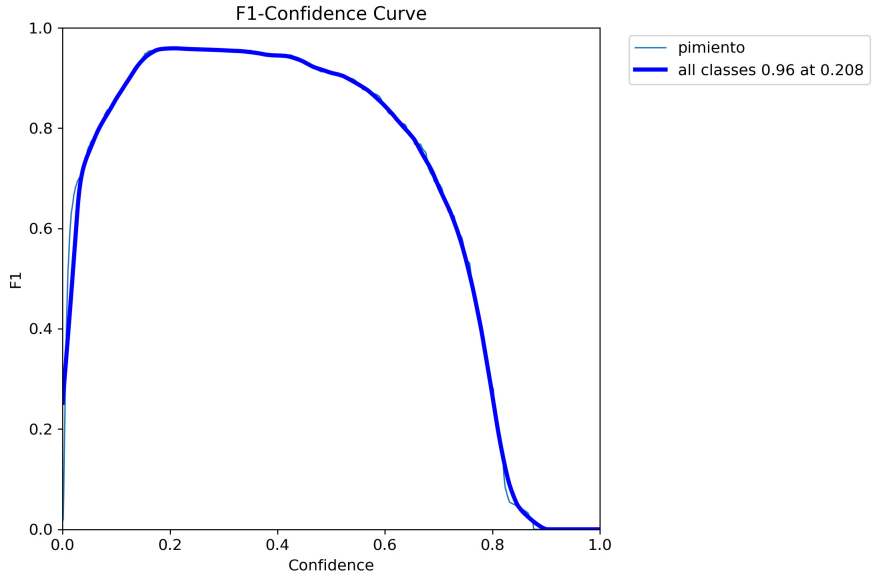


Figura 37. Curva F1 - Confianza.

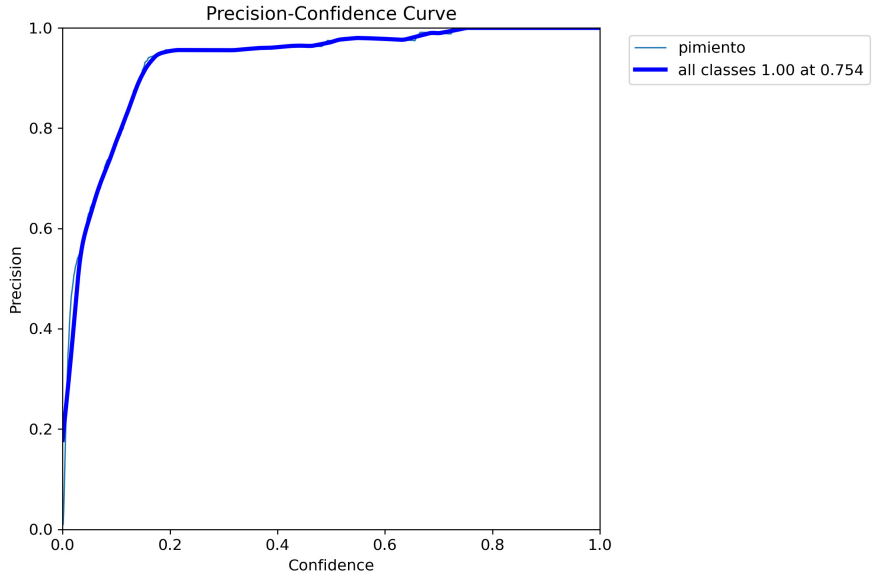


Figura 38. Curva Precisión - Confianza.

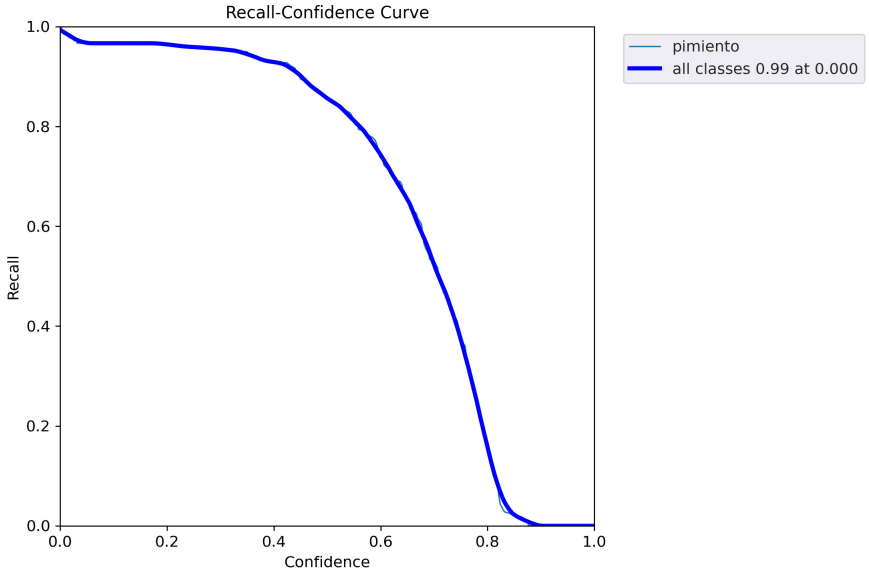


Figura 39. Curva Recall - Confianza.

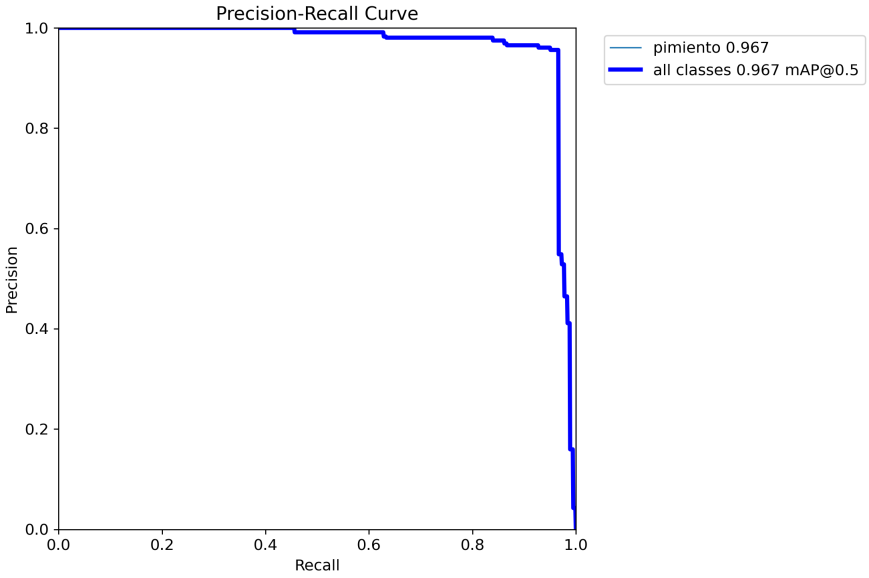


Figura 40. Curva Precisión - Recall.

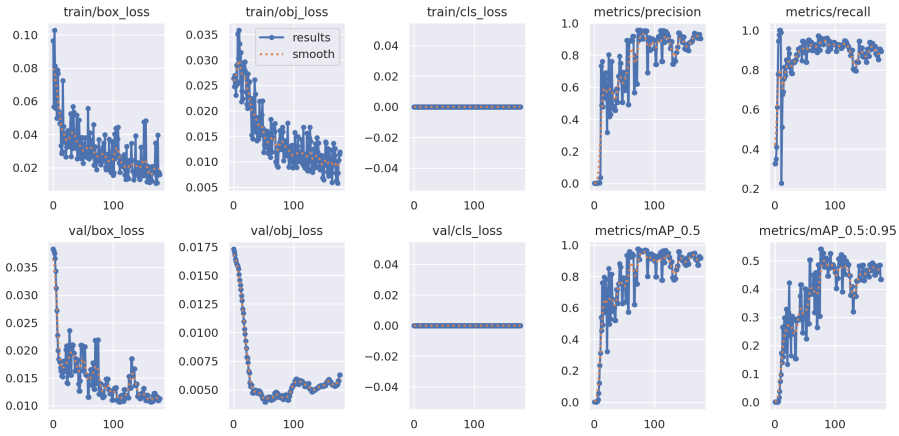


Figura 41. Resultados.

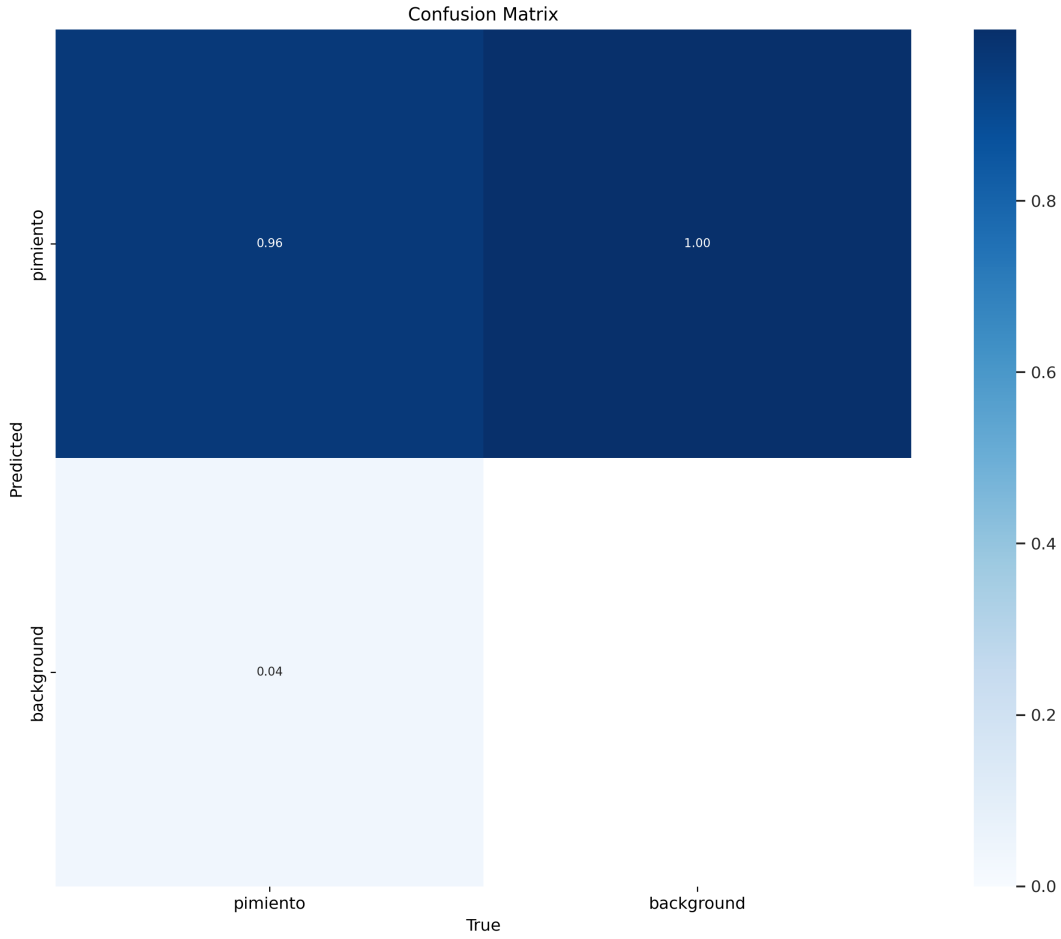


Figura 42. Matriz de confusión.

El entrenamiento fue validado con otro grupo de imágenes, ejemplo de ellos son las Figuras 43 y 44. Donde en la Figura 43 se muestra las imágenes con las etiquetas y en la Figura 44 se muestra los resultados de la predicción empleada en estas imágenes. El resto de las imágenes validadas tanto con las etiquetas como sin las etiquetas se las puede

observar en el Apéndice E.

Figura 43. Fotos de validación con etiquetas.

Figura 44. Fotos de validación con predicción.

El entrenamiento también fue testeado en un grupo de imágenes que contiene tanto plantas de pimientos como otro tipo de plantas ejemplo de ello lo podemos observar en las Figuras 45 y 46. Todas las imágenes testeadas con los resultados que la red entrenada se puede observar en el Apéndice F.

Figura 45. Foto testeada de planta de pimientos.

Figura 46. Foto testeada de planta que no es de pimientos.

8. Conclusiones

- De acuerdo con los cálculos realizados se llegó a la conclusión de que la configuración de robot móvil terrestre Ackerman es la más adecuada para su implementación en el prototipo, ya que el torque requerido para sus motores de 1,99 $kgcm$ y los motores más adecuados que brindan 2,2 $kgcm$, si se los puede encontrar en el mercado del país.

- El prototipo de robot móvil terrestre y su sistema de control fue diseñado, simulado y construido con materiales y tecnología disponibles en el mercado local.

- El control del robot móvil se realizó mediante el uso de los encoders integrados en los motores. Conociendo cuantos pulso genera al dar una vuelta del motor además de conocer la relación de la caja reductora y el diámetro de las llantas, se puede calcular el desplazamiento angular y con ello el desplazamiento lineal del robot. de esta manera controlando el desplazamiento del robot.

- El prototipo, con dimensiones generales de 322,1 x 304,8 x 177,1 mm (largo, ancho y altura) y un peso aproximado de 2,68 kg, consta de una chasis en MDF de 6 mm de espesor, que se moviliza en terrenos no uniformes con un mecanismo de ruedas de ABS y goma, accionado por dos motorreductores modelo JGA25-370, que funcionan con baterías de motocicleta de 12 VDC y transporta una cámara Raspberry Pi de 8 Megapíxeles que permite la toma de imágenes la planta de pimientos. Una carrocería en PETG cubre todos los componentes para proteger los componentes de la intemperie.

- En cuanto a la detección biométrica de las plantas se puede emplear diferentes marcos de aprendizaje para el reconocimiento de las plantas de pimientos entre ellos los más conocidos son: Tensorflow, Pythorch, Caffe AI, Scikit-learn, YOLOv5. Entre ellos se empleó YOLOv5 el cual aunque no se lo puede considerar un marco de aprendizaje como tal esta cuenta con redes pre entrenadas, las cuales se pueden volver a entrenar con un set de datos personalizado para lograr identificar objetos específicos, en este caso una planta de pimientos.

- Dentro del control de la cámara está integrada en el código la consulta de la hora actual, una vez que la hora sea igual a una predefinida en esta caso las 17:00 la Raspberry enviará una señal al ESP32 para que empiece la rutina movilizando al robot hacia la planta de pimientos. De igual forma el prototipo cuenta con un pulsador físico para que un operador pueda accionar la rutina en cualquier momento.

- Al momento de encender el prototipo este procede a realizar una pequeña rutina en la cual prueba el funcionamiento del servomotor, moviendo la dirección del robot. Además de accionar los motores hacia adelante y hacia atrás para probar el funcionamiento de los motores.

9. Recomendaciones

- Para mejorar la calidad de las imágenes se recomienda reemplazar la cámara de raspberry por una webcam, ya que estas cuentan con sensores y lentes de mejor calidad.

- Se puede agregar un mayor número de plantas para que la adquisición de las imágenes sea mayor además de que contar con plantas diferentes mejoran el entrenamiento.

- Para obtener una mayor confianza en la identificación de las plantas se debe contar con un mayor número de imágenes tomadas desde diversos ángulos.

- Se pueden agregar sensores alrededor del sensor para identificar si encuentra algún obstáculo y poder evitarlo.

- Ya que el prototipo está encendido por varios días pero cumple con su rutina solo una vez al día se pueden implementar modos de bajo consumo en los controladores.

BIBLIOGRAFÍA

[1] El cultivo del pimiento (1ª parte). [En línea]. Disponible: https://www.infoagro.com/hortalizas/pimiento.htm [Fecha de consulta: Julio 2023]

[2] DANE, "El cultivo del pimentón (capsicum annuum l) bajo invernadero," *INSUMOS Y FACTORES ASOCIADOS A LA PRODUCCIÓN AGROPECUARIA*, 2015.

[3] La absorción de nutrientes en pimiento (chile). [En línea]. Disponible: https://www.infoagro.com/documentos/la_absorcion_nutrientes_pimiento__chile_.asp [Fecha de consulta: Julio 2023]

[4] L. M. Enciso Salas, "Diseño de un sistema de navegación autónomo para robots móviles usando fusión de sensores y controladores neuro difusos," Proy. titulación, Pontificia Universidad Católica del Perú, Lima, Perú, 2015.

[5] R. A. Orozco Velázquez, "Robot móvil colector con visión artificial," Proy. maestría, Instituto Politécnico Nacional, D.F., México, 2014.

[6] D. E. Hernández Sánchez, "Diseño e implementacion de un robot miniatura para proveer estimulación iterativa a la extremidad posterior del mutante de mielina taiep," Proy. maestría, Benemérita Universidad Autónoma de Puebla, Puebla, México, 2017.

[7] H. Barcenas Díaz y C. Castañeda Nava, "Prototipo de robot móvil omnidireccional de cuatro llantas tele-operado bajo plataforma móvil," Proy. titulación, Instituto Politécnico Nacional, D.F., México, 2013.

[8] P. P. Israel Vaca, Jakeline Arias, "Buenas prácticas agrÍcolas para hortalizas y verduras," 2015.

[9] Doit esp32 dev kit v1 high resolution pinout and specs. [En línea]. Disponible: https://www.mischianti.org/2021/02/17/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/ [Fecha de consulta: Enero 2023]

[10] Interface l298n dc motor driver module with arduino. [En línea]. Disponible: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/ [Fecha de consulta: Octubre 2022]

[11] Motor dc jga25-370 - 12v/350rpm con encoder. [En línea]. Disponible: https://naylampmechatronics.com/motores-dc/616-motor-dc-jga25-370-12v350rpm-con-encoder.html (2022)

[12] CÁmara 8mpx v2 para raspberry. [En línea]. Disponible: https://grupoelectrostore.com/shop/placas-para-programacion/raspberry/accesorios-para-raspberry/camara-8mpx-v2-para-raspberry/

[13] pinout of lm2596 dc dc buck converter module adjustable power supply board in pakistan leave a comment. [En línea]. Disponible: https://electrobes.com/?attachment_id=42038

[14] E. D. T. Peñaloza, “DiseÑo de una mÁquina cnc de corte por plasma con control de altura de la antorcha,” 2020.

[15] S. C. V. Loaiza, “Obtención de los parámetros necesarios para el cálculo de la fuerza en rueda de vehículos eléctricos,” 2016.

[16] A. C. Ñontol, “DiseÑo de una maquina devanadora automÁtica para mejorar la calidad en fabricaciÓn de bobinas de motores elÉctricos en seltromind-cajamarca-2017,” 2018.

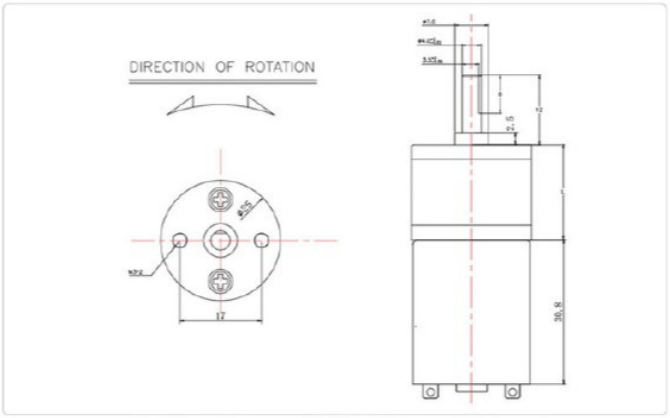
APÉNDICE A

Datasheet Motor JGA25-370

- JGA25-370 Dc motor with Gearbox Specs:

  - High quality and 100% brand new
  - Rated volt: 6v 12v 24v
  - No load speed: 12/16/26/35/60/77/130/170/280/620/1360 RPM

25MM dc geared motor Dimension:



Datasheet:

	No load		On Load				Stall			
Rated Volnge	rpm	Current(ma)	rpm	Current(ma)	Torque(Kg.cm)	Output(W)	Torque(Kg.cm)	Current(ma)	Ratio (1:1)	Size(L)
6v	1363	80	1090	380	0.1	2	0.8	900	4.4	18
6v	646	80	516	380	0.23	2	1.5	900	9.6	17.5
6v	281	80	238	380	0.5	2	4	900	21	19
6v	176	80	150	380	0.85	2	5	900	35	21
6v	133	80	113	380	1.1	2	6	900	45	21
6v	77	80	65	380	1.95	2	10	900	78	23
6v	58	80	46	380	3	2	15	900	103	25
6v	35	80	28	380	4	2	25	900	171	25
6v	26	80	20	380	5	2	35	900	226	27
6v	16	80	13	380	9	2	Forbidden	900	377	27
6v	12	80	10	380	10	2	Forbidden	900	500	18
12v	1931	50	1351	250	0.11	4	0.44	1200	4.4	17.5
12v	915	50	640	250	0.24	4	1	1200	9.6	19
12v	399	50	279	250	0.55	4	2.2	1200	21	21
12v	250	50	175	240	0.95	4	38	1200	35	21
12v	188	50	131	240	1.26	4	5	1200	45	23
12v	108	50	75	240	2.1	4	8.4	1200	78	23
12v	82	50	57	240	2.88	4	11	1200	103	25
12v	50	50	40	240	4.78	4	18	1200	171	25
12v	37	50	29	240	6.3	4	14	1200	226	27
12v	22	50	17	240	10	4	Forbidden	1200	377	27
12v	17	50	13	240	14	4	Forbidden	1200	500	18
24v	863	10	604	70	0.083	1	0.4	220	4.4	17.5
24v	409	10	280	70	0.17	1	0.88	220	9.6	19
24v	178	10	125	70	0.4	1	2	220	21	21
24v	111	10	80	70	0.64	1	3.2	220	35	21
24v	84	10	60	70	0.85	1	4.2	220	45	23
24v	48	10	35	70	1.4	1	7.4	220	78	23
24v	36	10	25	70	1.9	1	9.5	220	103	23
24v	22	10	15	70	3.2	1	16	220	171	25
24v	17	10	12	70	4.2	1	21	220	226	25
24v	10	10	7	70	7	1	35	220	377	27
24v	7.5	10	5	70	9.5	1	47	220	500	27

Figura A.1. Especificaciones de Motor

APÉNDICE B

Código de movilidad del robot

```
#include <Servo.h>
//Direcion
Servo direccion;
int dir = 13;
int DER = 55;
int IZQ = 125;
int CEN = 90;
//variables encoder Motor 1
int EncA = 35;
const int encoder_ticA = 206;
volatile int contadorA = 0;
int Obj_encoderA = 0;
//variables encoder Motor 2
int EncB = 34;
const int encoder_ticB = 206;
volatile int contadorB = 0;
int Obj_encoderB = 0;
//Motor A
int PWM_MotorA = 12;
int Dir_MotorA_1 = 27;
int Dir_MotorA_2 = 14;
int State_MA1 = LOW;
int State_MA2 = LOW;
int spA = 250;
//Motor B
int PWM_MotorB = 33;
int Dir_MotorB_1 = 26;
```

```
int Dir_MotorB_2 = 25;
int State_MB1 = LOW;
int State_MB2 = LOW;
int spB = 240;
//Boton
int Bot_inicio = 32;
int Pulso_Rasp1 = 19; // OUTPUT
int Pulso_Rasp2 = 18; // INPUT
int Led = 2;
const float Diametro = 8.3;
//float Circun = Diametro * PI;
int distancia =100;
int ruti = 0;
int Tramo = 0;
int aux_ruti = 0;

void setup() {
  direccion.attach(dir);
  direccion.write(CEN);
  pinMode(EncA, INPUT);
  pinMode(EncB, INPUT);
  attachInterrupt(EncA,EncoderA, RISING);
  attachInterrupt(EncB,EncoderB, RISING);
  pinMode(PWM_MotorA, OUTPUT);
  digitalWrite(PWM_MotorA,LOW);
  pinMode(Dir_MotorA_1, OUTPUT);
  pinMode(Dir_MotorA_2, OUTPUT);
  pinMode(PWM_MotorB, OUTPUT);
  digitalWrite(PWM_MotorB,LOW);
  pinMode(Dir_MotorB_1, OUTPUT);
```

```
  pinMode(Dir_MotorB_2, OUTPUT);
  pinMode(Bot_inicio, INPUT_PULLUP);
  pinMode(Led, OUTPUT);
  pinMode(Pulso_Rasp1, OUTPUT);
  pinMode(Pulso_Rasp2, INPUT);
  digitalWrite(Led,LOW);
  digitalWrite(Pulso_Rasp1,LOW);
  direccion.write(IZQ);
  delay(700);
  direccion.write(DER);
  delay(700);
  direccion.write(CEN+7);
  delay(2000);
}

void loop() {
  if (digitalRead(Bot_inicio) == 1){
    Tramo++;
    ruti = 1;
    delay(100);
    digitalWrite(Led,HIGH);
  }
  else if (digitalRead(Pulso_Rasp2) == 1 && aux_ruti == 0){
    Tramo++;
    ruti = 1;
    delay(100);
  }
  else if (digitalRead(Pulso_Rasp2) == 1 && aux_ruti == 1){
    ruti = 1;
    aux_ruti = 0;
```

```
}

if (ruti == 1){
  switch(Tramo){
    case 0:
      State_MA1 = LOW;
      State_MA2 = LOW;
      State_MB1 = LOW;
      State_MB2 = LOW;
      analogWrite(PWM_MotorA,0);
      analogWrite(PWM_MotorB,0);
      digitalWrite(Dir_MotorA_1,State_MA1);
      digitalWrite(Dir_MotorB_1,State_MB1);
      digitalWrite(Dir_MotorA_2,State_MA2);
      digitalWrite(Dir_MotorB_2,State_MB2);
    break;
    case 1:
      spA = 223;
      spB = 230;
      direccion.write(CEN);
      distancia = 15;
      Motores_ade();
      Tramo++;
    break;
    case 2:
      spA = 183;
      spB = 190;
      direccion.write(CEN+3);
      distancia = 110;
      Motores_ade();
```

```
    Tramo++;
  break;
  case 3:
    direccion.write(CEN+2);
    distancia = 196;
    Motores_ade();
    Tramo++;
  break;
  case 4:
    spA = 223;
    spB = 230;
    direccion.write(CEN);
    distancia = 97;
    Motores_ade();
    Tramo++;
  break;
  case 5: //Captura de imagen
    direccion.write(IZQ - 17);
    distancia = 10;
    Motores_ade();
    Parar();
    Tramo++;
    aux_ruti = 1;
    digitalWrite(Pulso_Rasp1,HIGH);
    delay(700);
    digitalWrite(Pulso_Rasp1,LOW);
  break;
  case 6:
    direccion.write(IZQ - 15);
    distancia = 18;
```

```
        Motores_ret();
        Tramo++;
        digitalWrite(Led,HIGH);
      break;
      case 7:
        spA = 183;
        spB = 190;
        direccion.write(CEN + 2);
        distancia = 180;
        Motores_ret();
        Tramo++;
      break;
      case 8:
        direccion.write(CEN + 3);
        distancia = 140;
        Motores_ret();
        Tramo++;
      break;
      case 9:
        direccion.write(CEN + 1);
        distancia = 90;
        Motores_ret();
        Parar();
      break;
    }
  }
  delay(50);
}

void EncoderA () {
```

```
  contadorA++;
}
void EncoderB () {
  contadorB++;
}

void Motores_ade(){
  contadorB = 0;
  contadorA = 0;
  float Circun = Diametro * PI;
  Obj_encoderA = (distancia / Circun) * encoder_ticA;
  Obj_encoderB = (distancia / Circun) * encoder_ticB;
  //Tramo++;
  State_MA1 = HIGH;
  State_MA2 = LOW;
  State_MB1 = HIGH;
  State_MB2 = LOW;
  digitalWrite(Dir_MotorA_1,State_MA1);
  digitalWrite(Dir_MotorB_1,State_MB1);
  digitalWrite(Dir_MotorA_2,State_MA2);
  digitalWrite(Dir_MotorB_2,State_MB2);
  analogWrite(PWM_MotorA,spA);
  analogWrite(PWM_MotorB,spB);
  while(Obj_encoderB >= contadorB || Obj_encoderA >= contadorA){  }
}

void Motores_ret(){
  contadorB = 0;
  contadorA = 0;
  float Circun = Diametro * PI;
```

```
  Obj_encoderA = (distancia / Circun) * encoder_ticA;
  Obj_encoderB = (distancia / Circun) * encoder_ticB;
  //Tramo++;
  State_MA1 = LOW;
  State_MA2 = HIGH;
  State_MB1 = LOW;
  State_MB2 = HIGH;
  digitalWrite(Dir_MotorA_1,State_MA1);
  digitalWrite(Dir_MotorB_1,State_MB1);
  digitalWrite(Dir_MotorA_2,State_MA2);
  digitalWrite(Dir_MotorB_2,State_MB2);
  analogWrite(PWM_MotorA,spA);
  analogWrite(PWM_MotorB,spB);
  while(Obj_encoderB >= contadorB || Obj_encoderA >= contadorA){  }
}

void Parar(){
  State_MB1 = LOW;
  State_MB2 = LOW;
  State_MA1 = LOW;
  State_MA2 = LOW;
  digitalWrite(Dir_MotorB_1,State_MB1);
  digitalWrite(Dir_MotorB_2,State_MB2);
  digitalWrite(Dir_MotorA_1,State_MA1);
  digitalWrite(Dir_MotorA_2,State_MA2);
  analogWrite(PWM_MotorB,0);
  analogWrite(PWM_MotorA,0);
  ruti = 0;
  digitalWrite(Led,LOW);
  if (Tramo == 9){
```

```
    Tramo = 0;
  }
}
```

APÉNDICE C

Control de la cámara y almacenamiento de imágenes

```
# Librerias
import RPi.GPIO as GPIO
import time
import datetime as dt
import subprocess as sp

# Se establece los pines fisicos de la Raspberry Pi
GPIO.setmode(GPIO.BOARD)
GPIO.setup(40, GPIO.IN)
GPIO.setup(38, GPIO.OUT)
GPIO.output(38, GPIO.LOW)

# print('Empieza')
# print(dt.datetime.now())
# print(GPIO.input(40))
# print(24)

while True:
    if (
            dt.datetime.now().hour == 17 and dt.datetime.now().minute == 0 and dt.dat

        # print('Es hora')
        GPIO.output(38, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(38, GPIO.LOW)
        time.sleep(1)
```

```
    if GPIO.input(40):
        nombre_img = "/home/acer/mnt/gdrive/Pimientos/Nuevas/" + str(dt.datetime.now
        sp.call(["raspistill", "-o", nombre_img])
        GPIO.output(38, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(38, GPIO.LOW)
        # print('Nombre de foto')
        # print(nombre_img)
        # break
    # print(GPIO.input(40))
    time.sleep(0.5)

# print('Nombre de foto')
# print(nombre_img)
```

APÉNDICE D

Reconocimiento

```
import tkinter as tk
from tkinter import filedialog
import os
import torch
from PIL import Image, ImageDraw, ImageFont

# Crear la ventana principal
window = tk.Tk()
window.title("Identificar Pimientos")

# Crear las etiquetas
label_red = tk.Label(window, text="Ubicación de red entrenada:")
label_img = tk.Label(window, text="Carpeta de imágenes:")
label_res = tk.Label(window, text="Carpeta para resultados:")
label_con = tk.Label(window, text="Confianza (0 - 100):")

# Crear los campos de texto
entry_red = tk.Entry(window)
entry_img = tk.Entry(window)
entry_res = tk.Entry(window)
entry_con = tk.Entry(window)

# Crear los botones
button_red = tk.Button(window, text="Seleccionar red")
button_img = tk.Button(window, text="Seleccionar carpeta")
button_res = tk.Button(window, text="Seleccionar carpeta")
button_det = tk.Button(window, text="Identificar")
```

```
# Crea boton check
var = tk.IntVar()
check_borr = tk.Checkbutton(window, text="Eliminar imagenes ya procesadas", variable=

# Fuente de texto
font = ImageFont.truetype("arial.ttf", 70)

# Mesajes
img_pro = 0
pim_idn = 0
t_mens = 0

def Ventana_mensaje(img_pro, pim_idn, t_mens):
    if t_mens == 0:
        titulo = "Error"
        mess_txt = "     No se ingreso archivo     "
        mess2_txt = "Red entrenada"
        mess3_txt = ""
    elif t_mens == 1:
        titulo = "Error"
        mess_txt = "     No se ingreso carpeta     "
        mess2_txt = "Imagenes tomadas"
        mess3_txt = ""
    elif t_mens == 2:
        titulo = "Error"
        mess_txt = "     No se ingreso carpeta     "
        mess2_txt = "Imagenes procesadas"
        mess3_txt = ""
    elif t_mens == 3:
```

```
            titulo = "Fin"
            mess_txt = "                Fin del procesamiento                "
            mess2_txt = "Imagenes procesadas: " + str(img_pro)
            mess3_txt = "Pimientos identificados: " + str(pim_idn)
        elif t_mens == 4:
            titulo = "Error"
            mess_txt = "     No se coloco porcentaje de confianza     "
            mess2_txt = ""
            mess3_txt = ""
        elif t_mens == 5:
            titulo = "Error"
            mess_txt = "     Confianza fuera de rango     "
            mess2_txt = ""
            mess3_txt = ""
        n_ventana = tk.Toplevel(window)
        n_ventana.title(titulo)
        mess = tk.Label(n_ventana, text=mess_txt)
        mess.pack()
        mess2 = tk.Label(n_ventana, text=mess2_txt)
        mess2.pack()
        mess3 = tk.Label(n_ventana, text=mess3_txt)
        mess3.pack()

    # Seleccionar la red
    def archivo():
        archivo_red = tk.filedialog.askopenfilename(title="Seleccionar red entrenada", fi
        # Actualizar el campo de texto con la dirección del archivo
        entry_red.delete(0, tk.END)
        entry_red.insert(0, archivo_red)
```

```
# Seleccionar la carpeta de imagenes
def carpeta_img():
    # Abrir una ventana de diálogo para elegir la carpeta
    directorio_imagenes = tk.filedialog.askdirectory(title="Selecciona carpeta de imá
    # Actualizar el campo de texto con la dirección de la carpeta
    entry_img.delete(0, tk.END)
    entry_img.insert(0, directorio_imagenes)

# Seleccionar la carpeta de resultados
def carpeta_res():
    # Abrir una ventana de diálogo para elegir la carpeta
    directorio_resultados = tk.filedialog.askdirectory(title="Selecciona carpeta de r
    # Actualizar el campo de texto con la dirección de la carpeta
    entry_res.delete(0, tk.END)
    entry_res.insert(0, directorio_resultados)

# Definir la función para mover el archivo
def proce_img():
    if len(entry_red.get()) == 0 or entry_red.get().isspace():
        img_pro = 0
        pim_idn = 0
        t_mens = 0
        Ventana_mensaje(img_pro, pim_idn, t_mens)
    elif len(entry_img.get()) == 0 or entry_img.get().isspace():
        img_pro = 0
        pim_idn = 0
        t_mens = 1
        Ventana_mensaje(img_pro, pim_idn, t_mens)
    elif len(entry_res.get()) == 0 or entry_res.get().isspace():
```

```
        img_pro = 0
        pim_idn = 0
        t_mens = 2
        Ventana_mensaje(img_pro, pim_idn, t_mens)
    elif len(entry_con.get()) == 0 or entry_con.get().isspace():
        img_pro = 0
        pim_idn = 0
        t_mens = 4
        Ventana_mensaje(img_pro, pim_idn, t_mens)
    else:
        num_conf = float(entry_con.get())

    if 0 > num_conf or 100 < num_conf:
        img_pro = 0
        pim_idn = 0
        t_mens = 5
        Ventana_mensaje(img_pro, pim_idn, t_mens)
    else:
        archivo_red = entry_red.get()
        directorio_imagenes = entry_img.get()
        directorio_resultados = entry_res.get()
        img_pro = 0
        pim_idn = 0
        t_mens = 3
        # Carga el modelo
        modelo = torch.hub.load('ultralytics/yolov5', 'custom', path=archivo_red)
        # Itera sobre cada imagen
        '''
        for nombre_imagen in os.listdir(directorio_imagenes):
            img_pro = img_pro + 1
```

```
        # Escala imagen
        ruta_imagen = os.path.join(directorio_imagenes, nombre_imagen)
        imagen = Image.open(ruta_imagen)
        n_eacala = (650, 650)
        imagen_redi = imagen.resize(n_eacala)
        imagen_redi.save(ruta_imagen)

    label_res.config(text="funciona")
    '''
    for nombre_imagen in os.listdir(directorio_imagenes):
        img_pro = img_pro + 1
        ruta_imagen = os.path.join(directorio_imagenes, nombre_imagen)
        ruta_resultados = os.path.join(directorio_resultados, nombre_imagen)

        # Carga la imagen
        imagen = Image.open(ruta_imagen)

        # Realiza la detección de Pimientos
        resultados = modelo(imagen)

        if len(resultados.xyxy[0]) == 0:  # Si no detecta Pimientos
            draw = ImageDraw.Draw(imagen)
            draw.text((20, 20), "No se detecto", fill=(255, 0, 0), font=font)
            imagen.save(ruta_resultados)
            continue
        else:
            conf = resultados.xyxy[0][0][4] * 100  # Porcentaje de confianza
            if conf <= num_conf:  # Si la confianza es menor o igual al 70 %
                draw = ImageDraw.Draw(imagen)
                draw.text((20, 20), "No se detecto", fill=(255, 0, 0), font=font)
```

```
                    imagen.save(ruta_resultados)
                    continue
                else:  # Si la confianza es mayor al 70 %
                    pim_idn = pim_idn + 1
                    # Imprime los resultados
                    # resultados.print()
                    ruta_resultados = ruta_resultados.rstrip(".jpg")
                    # Dibuja las cajas delimitadoras en la imagen y la guarda
                    resultados.save(save_dir=ruta_resultados)

                    # Se mueven las imágenes a una única carpeta
                    r = os.path.join(ruta_resultados, nombre_imagen)
                    rr = os.path.join(directorio_resultados, nombre_imagen)
                    os.rename(r, rr)
                    os.rmdir(ruta_resultados)
        Ventana_mensaje(img_pro, pim_idn, t_mens)
        if var.get() == 1:
            for archi in os.listdir(directorio_imagenes):
                ruta_imagen_ele = os.path.join(directorio_imagenes, archi)
                if os.path.isfile(ruta_imagen_ele):
                    os.remove(ruta_imagen_ele)



# Botones
button_red.config(command=archivo)
button_img.config(command=carpeta_img)
button_res.config(command=carpeta_res)
button_det.config(command=proce_img)
```

```
# Colocar los elementos en la ventana usando grid
label_red.grid(row=0, column=0, sticky="e")
button_red.grid(row=0, column=1)
entry_red.grid(row=0, column=2)

label_img.grid(row=1, column=0, sticky="e")
button_img.grid(row=1, column=1)
entry_img.grid(row=1, column=2)

label_res.grid(row=2, column=0, sticky="e")
button_res.grid(row=2, column=1)
entry_res.grid(row=2, column=2)

label_con.grid(row=3, column=0, sticky="e")
entry_con.grid(row=3, column=1)

check_borr.grid(row=4, column=0, columnspan=3)

button_det.grid(row=5, column=0, columnspan=3)

# Iniciar el bucle principal de la ventana
window.mainloop()
```

APÉNDICE E

Fotos validadas

Figura E.1. Fotos de validación con etiquetas 2.

Figura E.2. Fotos de validación con predicción 2.

Figura E.3. Fotos de validación con etiquetas 3.

Figura E.4. Fotos de validación con predicción 3.

APÉNDICE F

Fotos testeadas

Figura F.1. Test 1.

Figura F.2. Test 2.

Figura F.3. Test 3.

Figura F.4. Test 4.

Figura F.5. Test 5.

Figura F.6. Test 6.

Figura F.7. Test 7.

Figura F.8. Test 8.

Figura F.9. Test 9.

Figura F.10. Test 10.

Figura F.11. Test 11.

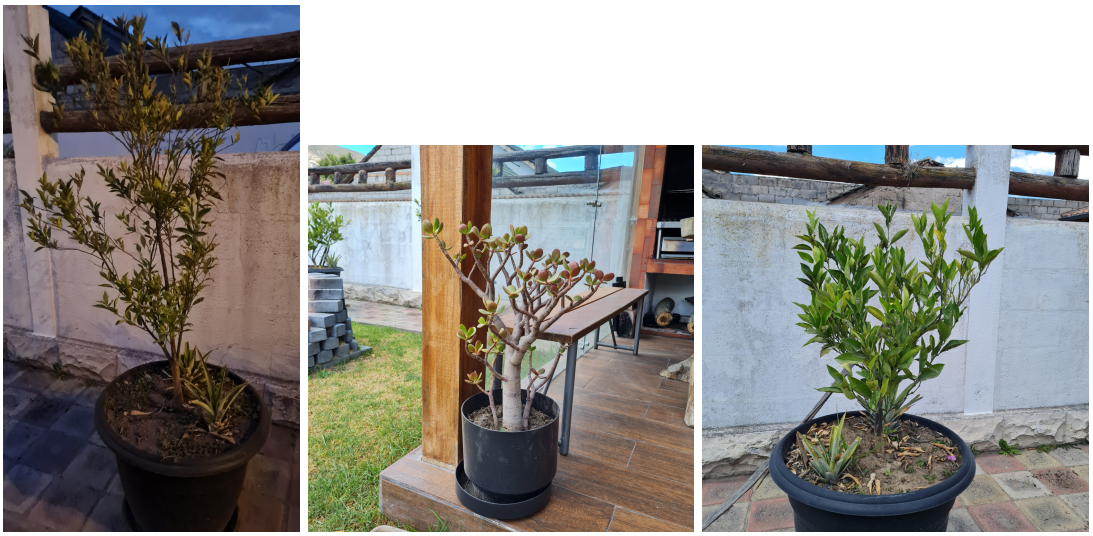


Figura F.12. Test 12.

Figura F.13. Test 13.

Figura F.14. Test 14.