UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL
CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MINI ROBOT MÓVIL BIDIRECCIONAL CON VIDEO CÁMARA PARA INSPECCIÓN EN TUBERÍAS DE DIÁMETROS REDUCIDOS

Trabajo de titulación previo a la obtención del
Título de Ingeniero Electrónico

AUTOR: BRYAM ESTALIN CUESTA CORREA
LEONARDO XAVIER MURILLO LEÓN
TUTOR: ORLANDO GIOVANNI BARCIAL AYALA

Guayaquil - Ecuador
2025

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Bryam Estalin Cuesta Correa con documento de identificación N° 0943315762 y Leonardo Xavier Murillo León con documento de identificación N° 0952848471; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 1 de septiembre del año 2025 Atentamente,

Bryam Cuesta

Bryam Estalin Cuesta Correa
0943315762

Leonardo Murillo

Leonardo Xavier Murillo León
0952848471

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Bryam Estalin Cuesta Correa con documento de identificación No. 0943315762 y Leonardo Xavier Murillo León con documento de identificación No. 0952848471, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "Diseño e implementación de un prototipo de mini robot móvil bidireccional con video cámara para inspección en tuberías de diámetros reducidos", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 1 de agosto del año 2025

Atentamente,

Bryam Estalin Cuesta Correa
0943315762

Leonardo Xavier Murillo León
0952848471

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Orlando Giovanni Barcia Ayala con documento de identificación N° 1309445714, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE MINI ROBOT MÓVIL BIDIRECCIONAL CON VIDEO CÁMARA PARA INSPECCIÓN EN TUBERÍAS DE DIÁMETROS REDUCIDOS, realizado por Bryam Estalin Cuesta Correa con documento de identificación N° 0943315762 y por Leonardo Xavier Murillo León con documento de identificación N° 0952848471, obteniendo como resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 1 de septiembre del año 2025

Atentamente,

Orlando Giovanni Barcia Ayala
1309445714

DEDICATORIA

Dedicado a mis padres; William y Laura, familia y amigos que me brindaron su apoyo en todo mi proceso educativo.

Bryam Estalin Cuesta Correa

DEDICATORIA

A todo el esfuerzo personal que deposité en este camino. A las horas de estudio, a los momentos de cansancio en los que aún, así decidí continuar, y a cada paso dado con perseverancia, incluso cuando las circunstancias no fueron fáciles. Este logro representa la suma de la disciplina, la paciencia y la voluntad de seguir adelante a pesar de las dudas. Este logro representa no solo un objetivo alcanzado, sino la prueba de que la constancia y la determinación abren caminos donde antes solo había incertidumbre.

A mi familia, especialmente a mi madre, por ser el pilar firme que me ha sostenido a lo largo de este camino. Por su amor incondicional, sus consejos llenos de sabiduría y el apoyo que me brindaron. Por su paciencia, y por ser esa fuente constante de inspiración y fortaleza que me impulsó a no rendirme.

Leonardo Xavier Murillo León

AGRADECIMIENTO

Mi más grande agradecimiento a mi madre Laura y a mi padre William, familia y amigos por su apoyo durante todo mi proceso educativo.

Bryam Estalin Cuesta Correa

AGRADECIMIENTO

Quiero expresar mi agradecimiento a mi madre, Elizabeth, por ser mi ejemplo constante de sacrificio, valentía y entrega. Gracias por tu apoyo inquebrantable, por creer en mí y por sostenerme incluso cuando dudaba de mis propias fuerzas.

A Dios, por darme fortaleza en los momentos más difíciles y guiarme con sabiduría durante todo este proceso académico.

Finalmente, agradezco a mi propia constancia, que me permitió mantenerme firme a pesar de las dificultades. Este trabajo no habría sido posible sin el conjunto de cada uno de estos apoyos.

Leonardo Xavier Murillo León

RESUMEN

En el presente Trabajo de Titulación hace el diseño e implementación de un prototipo de mini robot móvil bidireccional con video cámara para inspección de tuberías de diámetros reducidos. Para la inspección visual dentro de tuberías es común el uso de robots inspectores sin embargo estos robots por su tamaño dejan de ser útiles para esta tarea en tuberías de diámetros reducidos dejando como única alternativa el uso de cámaras endoscópicas.

El desarrollo de este prototipo de mini robot inspector con cámara suma como una alternativa para la inspección visual dentro de tuberías en los que por su pequeño diámetro antes si o si se debía usar una cámara endoscópica.

El desarrollo de este mini robot se desarrolla en varias etapas, incluyendo una fase de investigación bibliográfica y desarrollo de un primer prototipo de prueba previo al prototipo final. Las etapas de desarrollar van desde diseñar la estructura y mecanismo de desplazamiento y acople, el desarrollo del software y su integración con el hardware hasta su implementación en entornos ideales previamente preparados para los que fue diseñado.

*Palabras Clave**:** ESP32 CAM, Motor reductor, TBA6612FN, Arduino IDE, HTML,*

ABSTRACT

In this thesis, a prototype of a bidirectional mobile mini robot with a video camera is designed and implemented for the inspection of small-diameter pipes. Inspection robots are commonly used for visual inspection inside pipes; however, due to their size, these robots are no longer useful for this task in small-diameter pipes, leaving the use of endoscopic cameras as the only alternative.

The development of this prototype of a mini-inspection robot with a camera offers an alternative for visual inspection inside pipes where, due to their small diameter, an endoscopic camera was previously required.

The development of this mini robot is carried out in several stages, including a bibliographic research phase and the development of an initial test prototype prior to the final prototype. The development stages range from the design of the structure and the movement and coupling mechanism, software development and integration with the hardware, to its implementation in the ideal environments previously prepared for which it was designed.

Keywords: *ESP32 CAM, Gearmotor, TBA6612FN, Arduino IDE, HTML,*

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Muchos procesos productivos industriales están dotados de redes y sistemas tuberías de distintos tipos, tamaños y funcionalidades. Al ser parte de los distintos procesos es fundamental que estén en óptimas condiciones, por ello el mantenimiento preventivo y la detección de daños son necesarios, para esto existen equipos especializados en la inspección visual y detección de fallas dentro de tuberías como; las cámaras endoscópicas y los robots móviles inspectores, consiguiendo con este último un mejor control y operabilidad respecto a las cámaras endoscópicas. Los robots inspectores presentan una limitante, su tamaño, que les impide entrar a tuberías de diámetros reducidos.

El presente Trabajo de Titulación tiene como objetivo principal el diseño y la implementación de un prototipo de mini robot bidireccional con cámara de video para inspección de tuberías de diámetros reducidos. Este prototipo es una nueva alternativa dentro del campo de la inspección visual de tuberías de diámetro reducido.

Al ser un prototipo presenta limitaciones por lo que no tiene como finalidad ser implementado en un entorno real, más bien ofrece la oportunidad para ser usado como referencia en el desarrollo de robots capaces se ser usados en el entorno real objetivo. También se busca enriquecer la formación académica tanto dentro como fuera de la comunidad Universitaria, aportando con nuevas soluciones a los desafíos en el área.

PROBLEMA

El transporte de combustibles en refinerías y el transporte de agua potable comparten una cosa en común, para que ello suceda debe existir una red de tuberías. No solo forman parte de procesos industriales sino también están presentes en casas, edificios, departamentos y centros comerciales. En palabras de Gonzales (2021) los sistemas de tuberías son una parte esencial e integral de la civilización moderna, al igual que las arterias y las venas son esenciales para el cuerpo humano.

Es fundamental mantener estas redes en perfectas condiciones, para esto existen varias formas de corroborar el correcto de tuberías. Las ubicadas a grandes distancias bajo y sobre la superficie requieren el uso de sensores de presión, que detectan cambios entre la presión de entrada del sistema y la salida, otro método de detección usado son los sensores de vibración que detectan movimientos en las tuberías. La inspección visual es útil en tuberías superficiales, la visualización directa de grietas o fugas es relativamente sencilla, pero para tuberías ubicadas a grandes distancias sobre o bajo la superficie una inspección directa que a simple vista es imposible (Instituto de ingenieria de la UNAM, 2018).

Uno de los dispositivos usados para la inspección visual en tuberías son los robots móviles, estos al igual que las cámaras endoscópicas permiten una visualización interna de las tuberías por medio de una cámara de video. Una de sus ventajas es el alcance, que puede llegar a ser de miles de metros gracias a que pueden ser operados de forma remota e inalámbrica, también, pueden ser controlados con mayor precisión gracias a sus movimientos de desplazamiento y giros. (Díaz & Sarmiento, 2015).

Los robots móviles tradicionales pueden inspeccionar internamente tuberías de gran tamaño con las de aguas residuales sin inconvenientes como lo hacen algunos robots tele operados, pero

para tuberías de por ejemplo 6 pulgadas, estos robots móviles dejan de ser útiles, su tamaño se los impide (Estrella C, 2024).

Para los sistemas de tuberías de diámetros reducidos existen pocas alternativas en lo que a robots móviles inspectores se refiere, se han desarrollado robots móviles totalmente autónomos u operados diseñados para la exploración de tuberías de diámetros reducidos, por ejemplo, con sistemas de desplazamiento tipo oruga o sistemas de enlace extensible (Kim, 2010).

JUSTIFICACIÓN

El diseño y la implementación de un mini robot bidireccional preparado para la inspección visual en sistemas de tuberías les permite a los estudiantes de la Universidad Politécnica Salesiana en la carrera de Electrónica y Automatización aprender acerca de la cinemática de robots móviles de tamaño pequeño, la miniaturización electrónica, el uso de distintos sensores para el monitoreo de parámetros físicos en campos de aplicación industrial y la programación en placas de desarrollo y microcontroladores modernos.

Es uno de los primeros mini robot inspector de tuberías en la comunidad Universitaria Salesiana del Ecuador, aunque se orienta a un campo industrial puntual, podrá ser usado como punto de referencia en el campo de investigación e innovación en el área de robótica en la carrera de Electrónica y Automatización y a fines.

OBJETIVOS

Objetivo general

Implementar un prototipo de mini robot móvil bidireccional con videocámara para inspección en tuberías de diámetro reducido usando un microcontrolador ESP32.

Objetivos específicos

- Diseñar el cuerpo y mecanismo de desplazamiento bidireccional para el prototipo de mini robot móvil en un programa CAD para su posterior fabricación.
- Implementar el hardware de control y de video cámara para el prototipo de mini robot móvil.
- Desarrollar el software de control y de videocámara para el prototipo de mini robot móvil.
- Validar el mini robot móvil en un entorno de pruebas preparado para comprobar su funcionamiento en tuberías de diámetros reducidos.

FUNDAMENTO TEÓRICO

Daños en tuberías

Los problemas en los sistemas de tuberías están directamente relacionados a las funciones específicas que cumple el sistema, suponiendo que se cumplieron con todos los estándares durante el desarrollo e instalación de mismo, a continuación, se mencionan algunos de los problemas con mayor recurrencia en las tuberías:

Corrosión

La corrosión es un proceso de deterioro a los materiales, especialmente metales, por reacciones con el entorno. Este daño puede afectar la estructura de sistemas y presentar riesgos para la salud, como filtraciones y contaminación del agua. Factores que aumentan la corrosión incluyen un pH bajo, baja alcalinidad y altas temperaturas, que aceleran el deterioro y reducen la vida útil del material (Reyes, 2024).

Obstrucción

La obstrucción se refiere a la dificultad total o parcial para el movimiento de un material o líquido, este fenómeno puede deberse a residuos orgánicos, incrustaciones minerales, acumulación de grasa, o incluso objetos extraños que ingresan al conducto. Esto reduce el flujo del agua (Yang, y otros, 2019).

Fisuras y grietas

Una fisura se define como una pequeña y superficial abertura, en cambio, una grieta es más honda y puede atravesar por completo el material con el tiempo tienden a expandirse, generando fugas. Estas fallas suelen originarse por factores como el envejecimiento del material (De Miguel, 2015).

Sistemas para detección de fallas en tuberías

Video cámara

Una cámara es un equipo típicamente transportable que posibilita capturar fotos, transformándolas en señales eléctricas que pueden ser mostradas por un dispositivo específico, un ejemplo de cámara robotizada se muestra en la Figura 1. La habilidad de las cámaras de inspección es hallar características físicas normales y anormales; grietas, bloqueos y deterioro en la tubería, sin recurrir a excavaciones (Rico, 2012).

Figura 1

Cámara de vídeo inspección robotizada

Nota. Comúnmente empleada en pequeños robots, con capacidad de visión amplia y ajuste de ángulo, que optimiza la inspección gracias a su zoom ajustable y su extenso rango de inclinación. Tomado de **(Panatec agua, 2018)**.

Robots móviles para inspección de tuberías

Actualmente los métodos más utilizados para inspeccionar tuberías son métodos magnéticos, ultrasónicos o cámara oscilo giratoria. Los robots móviles destinados a la revisión de tuberías están diseñados para moverse de manera independiente o bajo control en tubos cerrados. Por lo general, estos aparatos están provistos de cámaras de video, sensores que indican su ubicación, luz incorporada y, en ciertas ocasiones, sensores que miden gases, humedad o temperatura,

según la aplicación específica, algunos robots (Urdaneta, 2012). En la Figura 2 se observan algunos modelos de robots móviles para trabajos de inspección.

Figura 2

Robots móviles para inspección de tuberías iPEK ROVIO.

Nota. La ilustración representa sistemas de inspección móvil ROVION son fáciles de mover a lugares inaccesibles para vehículos. Tomado de (Panatec agua, 2018).

Mini robots

La micro robótica desarrolla e investiga robots de tamaño reducido o escala pequeña. Según Rodríguez (2013) se tratan de robots que tienen como objetivo llevar a cabo tareas simples que puedan realizar con el menor número de fallos posible. Son robots que desempeñan funciones concretas. La Figura 3 muestra un mini robot creativo programable.

Figura 3

Mini Robot

Nota. La Figura representa muestra un robot móvil mBot es un autómata diseñado para educar en robótica, programación y electrónica. Tomado de (creativaKids, 2023)

Componentes de mini robot móvil inspector de tuberías

Un dispositivo robótico en movimiento creado para revisar tuberías necesita incluir varios elementos electrónicos, mecánicos y de comunicación que le posibiliten moverse de manera independiente o controlada a distancia, recolectar información visual del ambiente, ajustarse a situaciones físicas difíciles y enviar datos al instante. (Díaz & Sarmiento, 2015).

Sistema electrónico

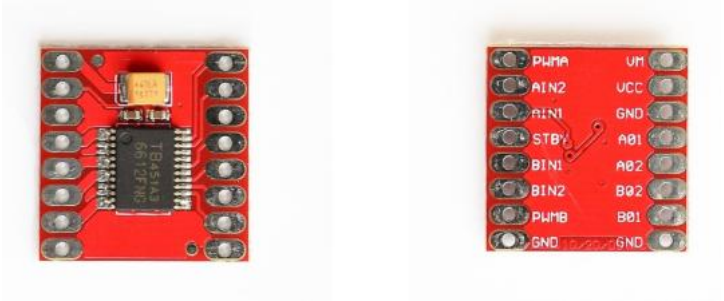
Los sistemas electrónicos se organizan en tres fases esenciales: la fase de entrada; que se ocupa de recolectar datos del ambiente como luz, temperatura, humedad, movimiento o presionar un botón, la fase de procesamiento o control; donde se examina e interpreta esa información y la fase de salida; en la que se decide si se activan o no ciertos actuadores o dispositivos de acuerdo con el procesamiento. (Floyd & Buchla, 2014).

Driver para motores

Los drivers para motores o módulos de potencia para motores de corriente continua permiten el accionamiento y el control de la velocidad de giro de motores gobernados por un microcontrolador. El módulo TB6612FNG mostrado en la Figura 4 tiene unas dimensiones de 20.32 mm x 15.24 mm y puede manejar voltajes de para alimentación de motores hasta 15V y una corriente promedio de salida de hasta 3A. Cuenta con 16 pines: VCC para alimentación del driver, VM para alimentación de motores, tres GND comunes, AIN1, AIN2, BIN1 e BIN2 para recepción de las señales provenientes desde el microcontrolador para inversión de giro, STBY (stand by) para encender y apagar el driver desde el microcontrolador, PWMA y PWMB para la señal PWM proveniente del microcontrolador y salidas de accionamiento de dos motore; A01, A02, B01 Y B02 (Naylamp Mechatronics, 2023).

Figura 4

Módulo TBA6612FNG



Nota. Modulo que permite controlar dos motores con una potencia máxima de 1.5A por motor. Tomada de (Naylamp Mechatronics, 2023).

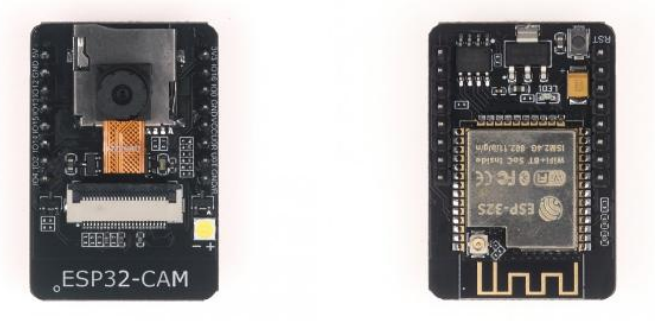
Placa de desarrollo ESP32-CAM, video cámara y modulo serial CH340

Esta placa desarrollo tiene como microcontrolador un ESP-32S lo que la dota de conectividad WiFi 802. 11b y Bluetooth 4.2. La placa de desarrollo de la Figura 5 tiene 16 pines, un slot para

tarjetas microSD, led indicador, botón reset, conector de antena y lo más importante, un sócalo para la conexión de cámaras OV2640 y OV3660 (NayLamp mechatronics, 2023).

Figura 5

Placa de desarrollo ESP32-CAM

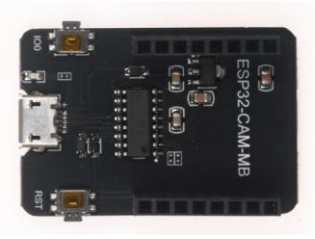


Nota. Vista frontal y trasera de placa de desarrollo ESP32-CAM con una cámara de video incluida. Tomado de (NayLamp mechatronics, 2023).

Para poder conectarse al módulo atreves de una interfaz USB para su programación y obtener las imágenes de video es necesario el módulo CH340 (NayLamp mechatronics, 2023). La Figura 6 es uno de los módulos comerciales de este tipo.

Figura 6

Módulo CH340



Nota. Vista superior de módulo CH340 con interfaz USB-C. Tomado de (NayLamp mechatronics, 2023).

Sistema mecánico

Los sistemas mecánicos mantienen estructuras o convierten energía, ya sea para incrementar la velocidad, cambiar la dirección del movimiento o provocar una alteración en la fuerza ejercida. (López, 2023).

Motor reductor

Combina un motor eléctrico con un reductor de velocidad. Su propósito fundamental es disminuir la velocidad de giro del motor, mientras que incrementa su torque, la Figura 7 muestra un motor reductor de doble eje. Esto es esencial en robótica, donde se necesita exactitud y potencia en los movimientos (Romero & Unamuno, 2020).

Figura 7

Mini motor reductor de 2 ejes.

Nota. Motor reductor de corriente continua de doble eje D con engranajes y una velocidad de 850 RMP con un voltaje de funcionamiento de 5V. Tomada de (Grupo ElectroStore, 2019).

Chasis (cuerpo de robot)

Constituye la parte estructural del robot, donde se encuentra el sistema de propulsión y que le permite moverse gracias a ruedas, orugas o diferentes mecanismos. Además, ofrece un soporte para montar herramientas como brazos, garras, elevadores, arados, sistemas de transporte,

dispositivos de recogida y otros elementos de diseño que se utilizan para interactuar con objetos (VEX V5, 2025). La Figura 8 muestra un tipo de chasis para un robot omnidireccional.

Figura 8

Chasis para robot.

Nota. El chasis de un robot cuenta con tres ruedas. Tomada de (Thido Electrónica, 2024).

Ruedas

Brindan la tracción esencial que necesita el robot para desplazarse. Tienen 3 partes, el eje, el aro y el neumático. El eje puede ser con o sin rodamientos, el aro puede construirse en distintos tipos de materiales como aluminio, acero, plástico, de igual forma el neumático que puede ser macizo o con recamara. En la Figura 9 muestra una rueda para un robot móvil. Su manejo es más exacto en comparación con los robots que utilizan orugas o patas. (Rivera, 2016).

Figura 9

Ruedas para robot.

Nota. Ruedas para robots móviles de pequeño tamaño. Tomada de (Amazon, 2025).

Router TP-LINK

Es un equipo que permite establecer comunicación entre varias redes y equipos informáticos permitiendo el control del tráfico de paquetes de datos para que lleguen al destino correcto. Su uso más común es permitir a dispositivos establecer una conexión con internet, los dispositivos pueden ser conectados por cable Ethernet o de forma inalámbrica WiFi. El modelo TL-WR740N tiene un interfaz de 4 puertos LAN 10/100Mbps y un puerto LAN 10/100Mbp, una antena omnidireccional de 5dBi, estándares inalámbricos IEEE 802.11n, 802.11G y 802.11b. Se alimenta con una fuente externa 5VDC/0.6A. Puede manejar una velocidad de transferencia máxima de 150Mbps (TP-LINK, 2025). En la Figura 10 se muestra una imagen de este modelo de router.

Figura 10

Router TP-LINK WR7040N de 150Mbps

Router TP-LINK WR7040N de 150Mbps. Tomado de (TP-LINK, 2025)

MARCO METODOLÓGICO

En este trabajo de titulación se usa una metodología investigativa y experimental, que se integran de manera complementaria para obtener los resultados deseados y conclusiones. Para empezar, se hace uso de la metodología investigativa con la búsqueda de información acerca de modelos de mini robots inspectores, mecanismos de desplazamiento, componentes electrónicos, protocolos de comunicación y entornos de desarrollo de software como programas CAD e IDE's para microcontroladores, todo esto proporciona una base teórica solida que permitirá cumplir con todas las fases de desarrollo de este mini robot.

Con los conocimientos previos se empieza a hacer uso de la metodología experimental con el desarrollo del software de control y video cámara, los diseños de los mecanismos, y la interfaz de control. Al ser un prototipo que se implementa de forma física es necesario hacer un diseño inicial, evaluar sus limitaciones y a partir de eso crear un nuevo prototipo mejor.

La elaboración de este trabajo de titulación se divide en las siguientes fases:

- Diseño CAD y mecánico de cuerpo, mecanismo de desplazamiento, mecanismo de auto acoplamiento e impresión 3D de las partes que conforman el mini robot.
- Interconexión entre CAM-ESP32, sistema de alimentación y modulo para control de motores.
- Programación de placa CAM-ESP32 S.
- Diseño de interfaz de control y video cámara del mini robot.
- Integración de todos los elementos del mini robot móvil bidireccional para sus respectivas pruebas en un entorno controlado.

Diseño CAD y mecánico de cuerpo, mecanismo de desplazamiento, mecanismo de auto acoplamiento e impresión 3D de las partes que conforman el mini robot.

Para el diseño CAD se usó el programa FreeCAD, que es un modelador en 2D y 3d que permite diseñar cualquier tipo de objeto para posteriormente imprimirlo, hacer pruebas de funcionabilidad o implementarlo con otros objetos. Se crearon dos prototipos.

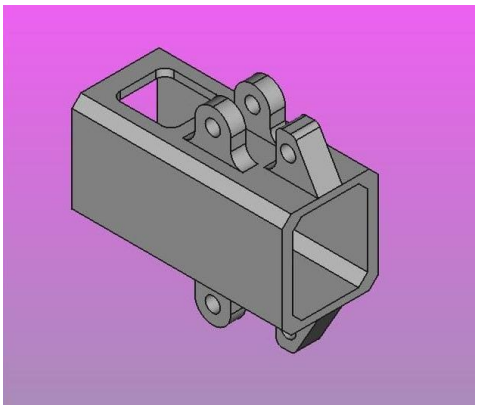
Primer prototipo

El trabajo de Tamara et al. (2015) con su micro-robot para inspección de tuberías es usado de guía para el desarrollo del primer prototipo de este mini robot inspector. La disposición de patas en forma de trípode fue adaptada a este mini robot con una disposición de solo dos patas, mientras que la disposición de los dos pares de muelles laterales para cada pata se mantuvo en el primer prototipo.

El primer prototipo fue un robot de dos brazos auto ajustables que podía ingresar dentro de tuberías de entre 90 mm hasta 230 mm. Para el desarrollo de este prototipo se diseñaron un total de dos piezas en FreeCAD, una de ellas era el cuerpo del robot y las otras dos los soportes para los dos motores que incorporaba. En la Figura 11 se muestra el diseño del cuerpo de este primer prototipo, y en la Figura 12 el diseño de los soportes para los motores.

Figura 11

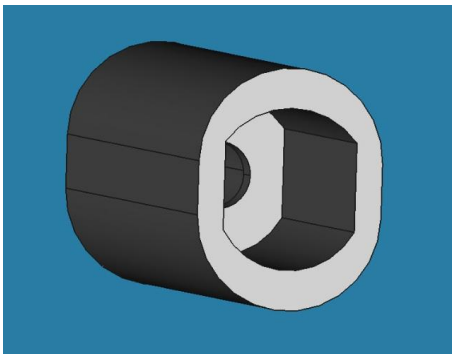
Cuerpo de primer prototipo, diseñada en FreeCAD



Nota. El cuerpo consta de tres estructuras de soporte, dos de ellas para los brazos auto acoplables y una de ellas para el soporte de los muelles.

Figura 12

Soporte para motor, diseñada en FreeCAD



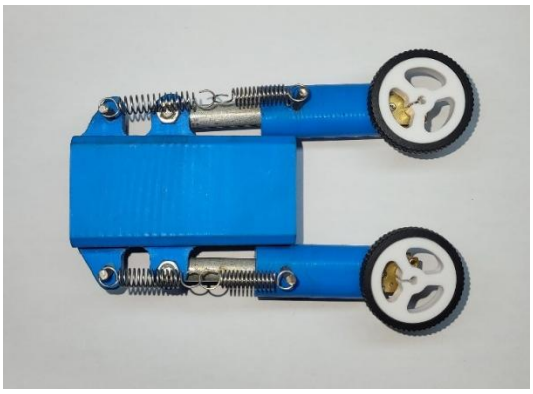
Nota. Se imprimieron dos de estas piezas, una para cada motor.

Para unir los soportes de los motores con el cuerpo del robot por medio de las estructuras de soporte se usó un tubo de aluminio de 45 mm de largo y 8mm de diámetro. Para que exista el auto acople fue necesario añadir 4 muelles entre la estructura de soporte de los muelles y el soporte para el motor, esto para cada brazo, dando un total de 8 muelles utilizados.

A los motores se les colocó un par de ruedas por motor, estas ruedas tienen un diámetro de 30 mm y un ancho de eje de 9 mm, estas ruedas son de plástico y caucho. Finalmente se imprimieron las piezas en materia PLA con una impresora 3D y se unieron sus partes para dar como resultado el primer prototipo de mini robot que se muestra en la Figura 13 y 14.

Figura 13

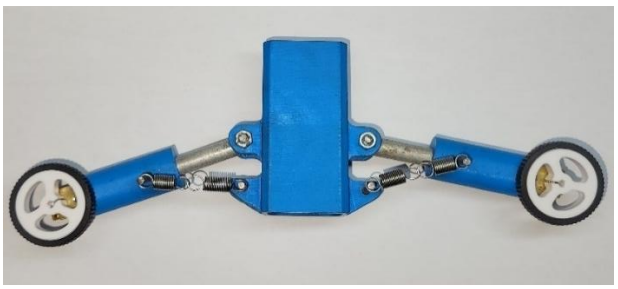
Primer prototipo terminado.



Nota. Mini robot en su mínima distancia de rueda a rueda

Figura 14

Primer prototipo terminado



Nota. Mini robot en su máxima distancia de rueda a rueda

Segundo prototipo (prototipo final)

Para este segundo prototipo es necesario considerar que, aunque sigue el diseño funcional del micro-robot la ausencia de una pata y un tamaño mayor no permiten la uniformidad de la sujeción del robot en las pardes de la tubería. Para solucionar esto se cambia la disposición de las ruedas y se incrementa el número de ruedas. El sistema de acople basado en muelles también se modifica ubicando los muelles cerca del centro del mini robot, aunque se mantiene el mismo principio de funcionamiento del micro-robot de referencia y el mismo número de patas del primer prototipo. El prototipo final y consta de siete piezas impresas y 4 eslabones, que dan como resultado un robot de dimensiones compactas como se muestra en Tabla 1.

Tabla 1

Dimensiones operativas del mini robot

DIMENSIONES DE OPERACIÓN	
Altura máxima	185 mm
Altura mínima	115 mm
Altura ideal de funcionamiento	125 mm – 175 mm

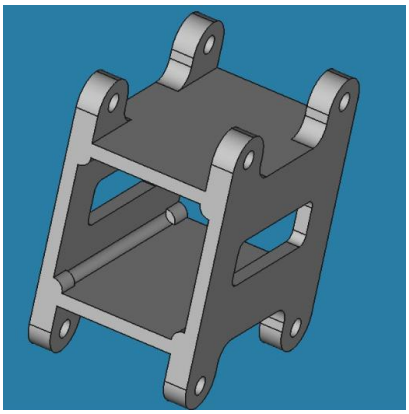
Nota. La altura es medida desde rueda a rueda. La altura corresponde al diámetro de la tubería por la que el mini robot puede navegar. La altura puede aumentar o disminuir cambiando el largo de los eslabones.

El cuerpo de este prototipo final del mini robot tiene forma de prisma rectangular y dimensiones de 36 mm de ancho, 51 mm de largo y 60 mm de alto contando sus soportes superiores e inferiores. En la parte superior cuenta con cuatro soportes con un agujero de 4 mm cada uno, estos sirven para colocar los eslabones que conectan el cuerpo y el par de estructuras que son

el soporte de las ruedas y motores del robot, lo mismo con la parte inferior del cuerpo. Dentro del cuerpo hay 4 soportes internos en sus esquinas, estos fueron sirven para incrustar tuercas de inserción de 3.5 mm que son útiles para colocar las tapas del cuerpo del robot para acceder al interior de forma sencilla. En la Figura 15 se muestra el diseño del cuerpo de este segundo prototipo, en la Figura 16 se muestra el cuerpo con las tuercas de inserción.

Figura 15

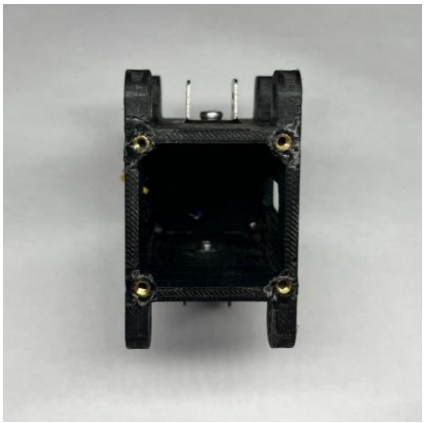
Cuerpo de segundo prototipo, diseñada en FreeCAD



Nota. En sus lados cuenta con una ranura que ayuda a la ventilación.

Figura 16

Cuerpo de segundo prototipo impreso e incorporada las tuercas de inserción.

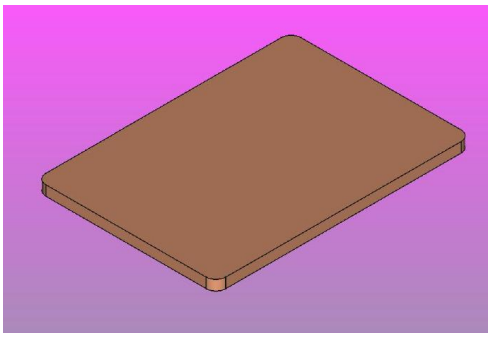


Nota. Las tuercas de inserción se colocaron tanto en la parte frontal como la trasera.

Para el cuerpo se diseñaron dos tapas cuadradas de 36 mm y 2 mm de grosor, se muestran en la Figura 17. Una vez impresas a estas tapas se les realizaron 4 agujeros por los que pasan los tornillos de cabeza estrella de 2.5 mm y 6 mm de largo. A una de las tapas se le hicieron dos agujeros cerca del centro, uno para el diodo led y otro para la cámara, esta será la tapa frontal como se muestra en la Figura 18. A la otra tapa se le realizo un agujero en el centro por el cual pasa el pin de energía para el robot como se muestra en la Figura 19.

Figura 17

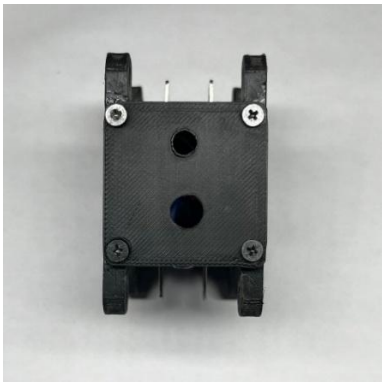
Tapas de segundo prototipo, diseñada en FreeCAD.



Nota. Para este prototipo se imprimieron dos de estas piezas.

Figura 18

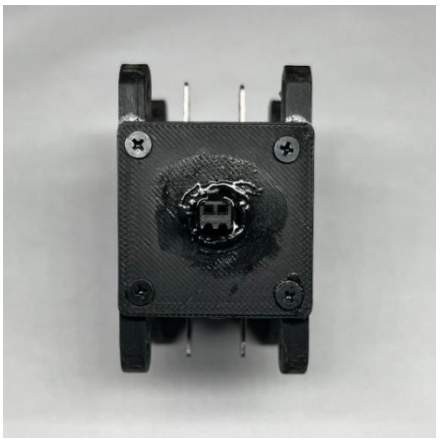
Tapa frontal de segundo prototipo impresa y con sus respectivos agujeros.



Nota. La tapa se encuentra ya colocada en el cuerpo del robot con sus respectivos tornillos.

Figura 19

Tapa trasera de segundo prototipo impresa y con sus respectivos agujeros.

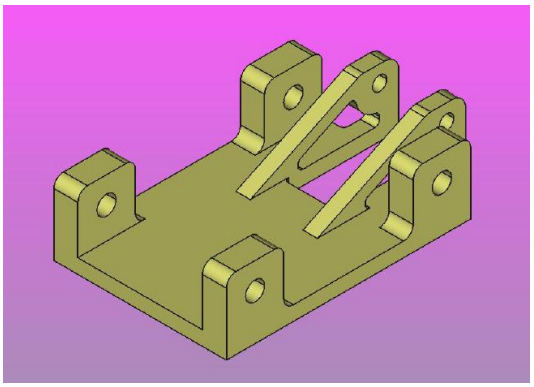


Nota. La tapa se encuentra ya colocada en el cuerpo del robot con sus respectivos tornillos y conector de energía.

Un par de estructuras son las que hacen de soporte para los motores y ruedas y a su vez se conectan a al cuerpo del robot por medio de eslabones. Estas estructuras son una plataforma plana rectangular de 3 mm de grosor, 51 mm de largo y 36 mm de ancho, sobre cada hay 4 soportes que sirven para la unión de los eslabones que conectan con el cuerpo. Una muesca de 11 mm da el espacio necesario para albergar una rueda guía centrada respecto a la plataforma, un pequeño brazo en forma de triangulo rectángulo sostiene la rueda guía de cada una de las plataformas. En esta estructura también se colocarán los soportes para los motores. En la Figura 20 se muestra el diseño de estas plataformas.

Figura 20

Plataforma rectangular de segundo prototipo, diseñada en FreeCAD

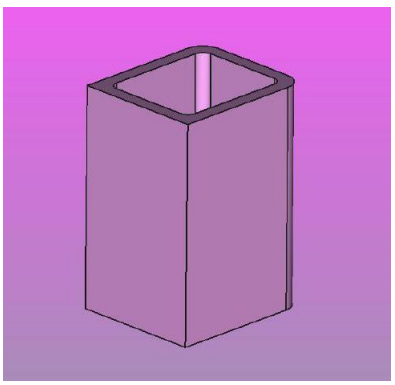


Nota. Para este prototipo se imprimieron dos de estas piezas.

Para colocar en la estructura los motores se diseñaron un par de piezas en forma de pequeños prismas rectangulares cuyas dimensiones internas son la de los motores reductores usados que son 12 mm de alto por 10 mm de ancho y 16 mm de profundidad y un tope de 6 mm. El largo exterior de estas piezas es de 22 mm, 15 mm de alto y 13 mm de ancho. Por la cara superior ingresa el motor, y por la inferior los cables de los, en la Figura 21 se muestra esta pieza de soporte para los motores reductores.

Figura 21

Soporte para motores de segundo prototipo, diseñado en FreeCAD.

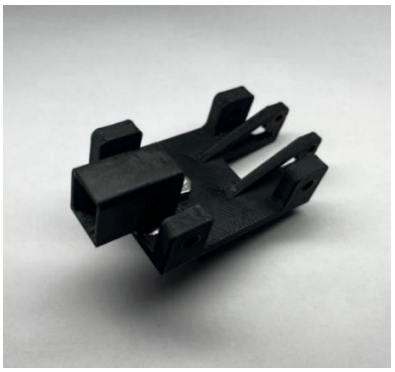


Nota. Para este prototipo se imprimieron dos de estas piezas.

Los soportes para los motores como las plataformas planas rectangulares son parte de la misma estructura. Para unirlas se usó pegamento bi-componente transparente, obteniendo una mayor resistencia entre estas piezas. En la Figura 22 se muestra el resultado final. Con los motores y las ruedas en su lugar la estructura que da movimiento a este robot se muestra en la Figura 23

Figura 22

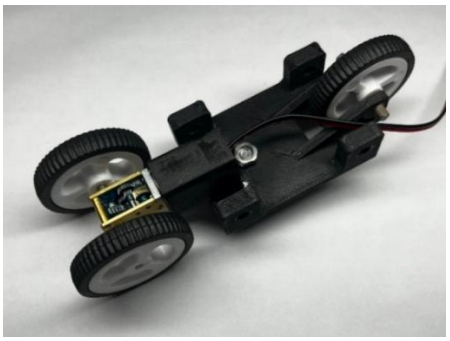
Estructura completa que da soporte a los motores y ruedas.



Nota. Para este prototipo se hicieron dos de estas piezas.

Figura 23

Motor y ruedas en la estructura que da movimiento a este robot.

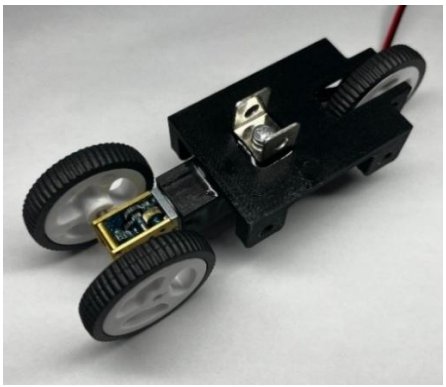


Nota. Para este prototipo se hicieron dos de estas estructuras cada una con su respectivo motor y ruedas.

Para el sistema de auto acoplamiento se usaron dos pares de muelles, 2 veces menos muelles que el primer prototipo. Para conectar los muelles entre la estructura que contiene las ruedas y el cuerpo del robot se hace uso de cuatro láminas de metal de 36 mm de largo, 10 mm de ancho y 1 mm de grosor, estas laminas tienen 3 agujeros de 5 mm distribuidos de forma uniforme, para hacerlas funcionales fueron dobladas en forma de ‘U’ y colocadas en el lado reverso de ambas estructuras que contienen las ruedas (ver Figura 24) y en la parte superior e inferior del cuerpo del robot (ver Figura 25). Para fijarlas fue necesario hacer una perforación de 4 mm en los lugares donde se ubicaron para colocar un tornillo de 4 mm y reforzar con pegamento bi componente. Tornillos pasarán de lado a lado en forma horizontal para hacer de eje a los muelles.

Figura 24

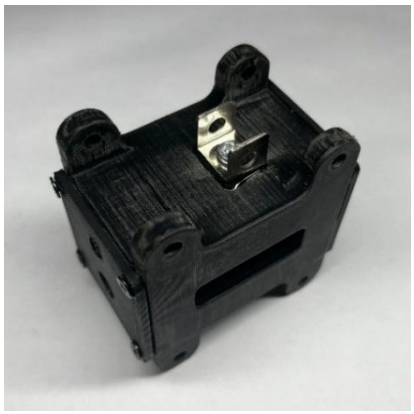
Motor y ruedas en la estructura que da movimiento a este robot



Nota. Se fijaron en el lado reverso de ambas estructuras de soporte para las ruedas.

Figura 25

Laminas incorporadas en el cuerpo del robot.

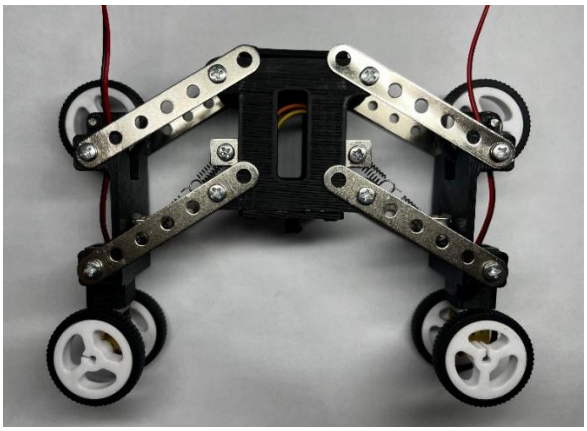


Nota. Se fijaron tanto en la parte superior del cuerpo como en la inferior

Para unir y fijar el cuerpo con la estructura que sostiene las ruedas y motor se hizo uso de 8 eslabones de 70 mm de largo, 10 mm de ancho y 1 mm de grosor, estos eslabones cuentan con 7 perforaciones de 5 mm. Se hizo uso de las perforaciones de los extremos de cada eslabón para por medio de un tornillo fijarlos a los soportes del cuerpo y de la estructura que contiene las ruedas como se muestra en la Figura 26.

Figura 26

Unión con eslabones de estructura para el sistema de auto acople.

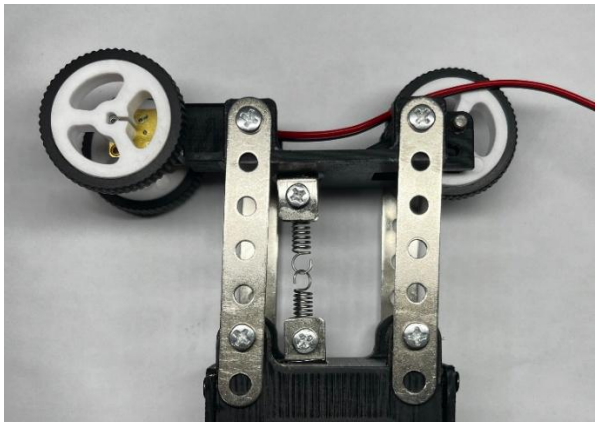


Nota. Esto se hizo para la otra estructura de igual forma.

En pares fueron unidos los muelles y se fijaron de cada extremo a las láminas que se colocaron en el cuerpo y en la estructura que contiene las ruedas del robot como se muestra en la Figura 27.

Figura 27

Unión de los muelles para el sistema de auto acople.



Nota. Esto se hizo para la otra estructura de igual forma.

En los anexos se encuentran los dibujos técnicos, perspectivas y medidas de todas las piezas diseñadas en FreeCAD e impresas.

Interconexión entre CAM-ESP32, sistema de alimentación, modulo para control de motores e iluminación led

En esta sección se diseña la interconexión del hardware del mini robot. Como resultado de la implementación de la parte eléctrica y electrónica del mini robot se obtiene como resultado las características eléctricas del mini robot descritas en la Tabla 2.

Tabla 2

Características eléctricas del mini robot

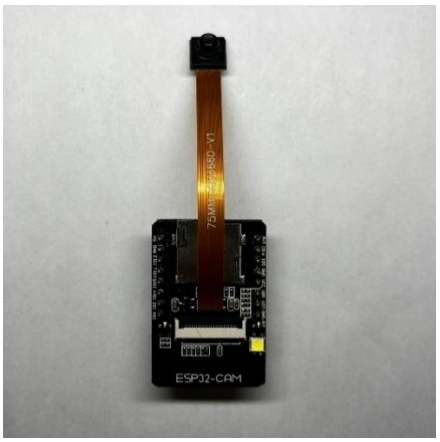
ENERGIA	
Voltaje de funcionamiento	4,5 v – 5v
Voltaje ideal de funcionamiento	5v
Potencia de consumo máxima	400mW

Nota. La potencia de consumo del mini robot está directamente relacionada con el grado de inclinación de la tubería y la distancia entre el mini y el router.

Una de las partes más importantes es la placa de desarrollo CAM-ESP32-S, cuenta con 16 pines; alimentación de 3,3v o 5v, pines de entradas y salidas analógicas y digitales, más que suficientes para el control del diodo led y los motores. Incorpora conexión WI-FI a 2,4GHz esencial para la transmisión de video streaming y control de forma remota, también dispone de un sócalo al cual se conecta la cámara OV3660. La cámara de 3 mega pixeles cuenta con un lente de 160 grados con enfoque manual y su flexor de conexión tiene una longitud de 75 mm que permite ubicarla de mejor forma tanto el CAM-ESP32-S como los demás componentes dentro del cuerpo del mini robot. En la Figura 28 se muestra el módulo y cámara reales usados en este robot.

Figura 28

CAM-ESP32-S con cámara OV3660.



*Nota.*CAM-ESP32-S con cámara OV3660.

El módulo TBA6612FN es el intermediario entre el CAM-ESP32-S y los motores. Sin él es imposible controlar los motores directamente pues la corriente de salida que es capaz de entregar el CAM-ESP32-S por sus pines es del orden los 20mA y los motores reductores N20 usados en este robot tienen un consumo de 120mA en promedio cada uno.

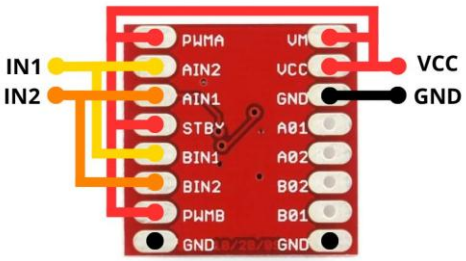
Para esta aplicación el módulo TBA6612FN se configura como siempre activo, esto se logra conectando STBY (Stand by) a VCC. No se controla la velocidad de los motores por lo que los pines PWMA y PWMB deben estar conectados a VCC para poder hacer uso del 100% la velocidad de los motores. El voltaje máximo que se entrega a los motores depende del suministrado en VM, el mini robot usará el VCC para alimentar los motores por lo que VM se conecta a VCC.

Para encender, apagar y cambiar sentido de giro de los motores es necesario que el CAM-ESP32-S envíe señales y que el módulo TBA6612FN las reciba, para ello se usan los pines AIN1, AIN2 para controlar las salidas AO1 y AO2 y BIN1, BIN2 para controlar las salidas BO1 y BO2. Como ambos motores trabajan al unisonó las señales enviadas desde el CAM-

ESP32-S a las entradas AIN1 e AIN2 son las mismas para BIN1 e BIN2 respectivamente, por eso, para ahorrar pines y conexiones se interconectaron AIN1 con BIN1 y AIN2 con BIN2. En la Figura 29 se muestra el esquema de conexiones y en la Figura 30 las conexiones reales.

Figura 29

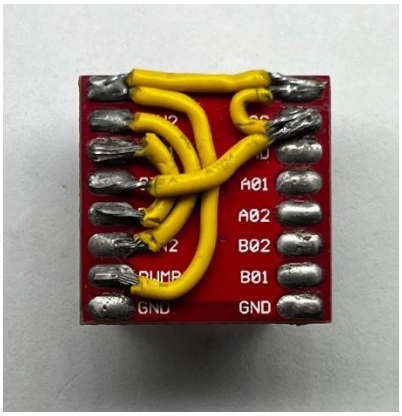
Esquema de conexiones de modulo TBA6612FN.



Nota. Esquema de conexión de modulo TBA6612FN.

Figura 30

Conexiones de modulo TBA6612FN



Nota. Conexiones de modulo TBA6612FN.

Los pines 12 y 13 del CAM-ESP32-S se usan para controlar el encendido/apagado de los motores y el sentido de giro por medio del módulo TBA6612FN. El pin 12 se conecta al

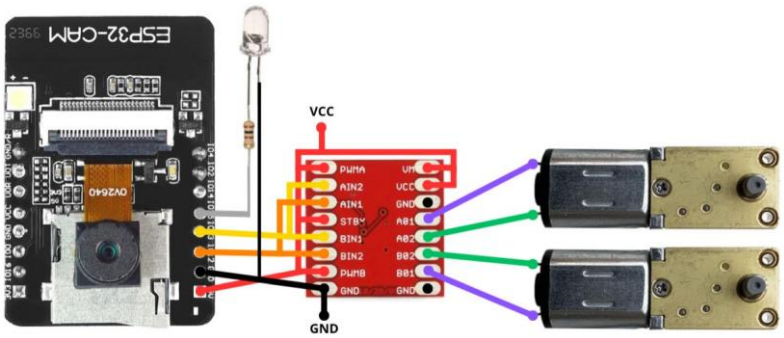
AIN2/BIN2, el pin 13 al AIN1/BIN1. El GND y el VCC se conectan entre si con la fuente de alimentación respectivamente.

El cátodo del led de luz blanca se conecta en serie con una resistencia de 100 ohms y el pin 15 a la resistencia y el ánodo del led se conecta al GND común.

A01, A02 se conectan a uno de los motores mientras que el B01, B02 al otro motor. El VCC del modelo se conecta VCC de la fuente y cualquier de los tres GND al GND de la fuente puesto que todos están conectados entre sí. En la Figura 31 se muestra el diagrama de conexiones de todos los componentes de este robot.

Figura 31

Esquemas de conexiones de modulo TBA6612FN, CAM-ESP32-S, Diodo LED y motores.



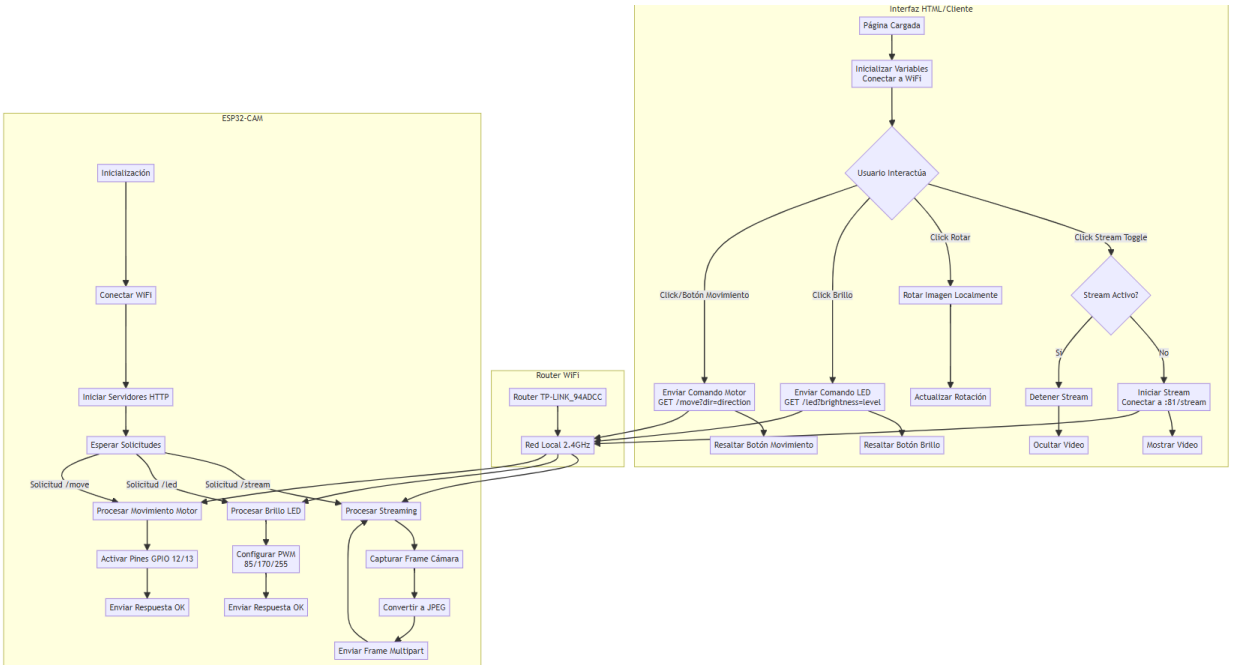
Nota. Esquemas de conexiones de modulo TBA6612FN, CAM-ESP32-S, LED y motores.

Diagrama de flujo de comunicación entre el CAM ESP32 S, router y HTML.

El diagrama de flujo de la Figura 32 muestra tres secciones, ESP32 CAM, Router y HTML, la interacción entre estos tres elementos es lo que permite controlar y ver el streaming del mini robot. El ESP32 CAM y HTML cuentan con una programación particular que permite la interacción entre ambas de manera eficaz.

Figura 32

Diagrama de flujo de comunicación



Nota. Diagrama de flujo de información entre el interfaz, el ESP32 CAM.

Diseño de código de programación de placa CAM ESP32 S

Los elementos de hardware usados son: El CAM-ESP32-S, el dispositivo cliente y un router. Para el control del mini robot se usa el protocolo de transferencia de hipertexto (HTTP) que tiene como cliente al dispositivo de control que puede ser un smartphone, Tablet, PC o laptop y como servidor el CAM-ESP32-S.

El router es el intermediario entre el cliente y el servidor, permite hacer uso de su red inalámbrica WI-FI 2,4GHz, asigna y gestiona las IPs de los clientes y habilita la comunicación bidireccional entre los dispositivos de la red local.

Para este mini robot la CAM-ESP32-S incluye código de control de cámara para streaming de video, control de motores, control de iluminación LED y servidor web para comandos y

streaming. Primero es importante agregar las librerías que serán útiles para hacer uso de funciones importantes dentro de la programación; *esp_camera.h* para control de cámara, *WiFi.h* para conexión WIFI, *esp_http_server.h* para servidor HTTP y *driver/ledc.h* para control PWM de LED. Segmento de código en la Figura 33.

Figura 33

Librerías, segmento de código de programación de CAM-ESP32-S

```
1   // Librerias
2   #include "esp_camera.h"
3   #include <WiFi.h>
4   #include "esp_http_server.h"
5   #include "driver/ledc.h"
```

Nota. Librerías, segmento de código de programación.

En la sección “configuración de pines para AI-Thinker ESP32-CAM” se definen variables y se le asignan pines físicos del chip que se usan para la configuración de la cámara.

Figura 34

Asignación de pines de cámara, segmento de código de programación de CAM-ESP32-S

```
8   #define CAMERA_MODEL_AI_THINKER
9   #define PWDN_GPIO_NUM     32
10  #define RESET_GPIO_NUM    -1
11  #define XCLK_GPIO_NUM      0
12  #define SIOD_GPIO_NUM     26
13  #define SIOC_GPIO_NUM     27
14  #define Y9_GPIO_NUM       35
15  #define Y8_GPIO_NUM       34
16  #define Y7_GPIO_NUM       39
17  #define Y6_GPIO_NUM       36
18  #define Y5_GPIO_NUM       21
19  #define Y4_GPIO_NUM       19
20  #define Y3_GPIO_NUM       18
21  #define Y2_GPIO_NUM        5
22  #define VSYNC_GPIO_NUM    25
23  #define HREF_GPIO_NUM     23
24  #define PCLK_GPIO_NUM     22
```

Nota. Asignación de pines de cámara, segmento de código de programación.

El sentido de giro de los motores se controla por medio de los pines 12 y 13 mientras el control del brillo del led se hace por medio del pin 15, en la Figura 35 se muestra este segmento de código.

Figura 35

Asignación de pines de control de sentido de giro y brillo del led

```
25  // Pines de control
26  #define MOTOR_PIN1 12
27  #define MOTOR_PIN2 13
28  #define LED_PIN    15
```

Nota: Asignación de pines de control de sentido de giro y brillo del led, segmento de código de programación.

Para el control del brillo led se hace uso del periférico de control LEDC que genera señales PWM para varias aplicaciones entre ellas el control de brillo de leds, por ejemplo. Se definen variables a cada elemento del periférico de control, a *LEDC_TIMER* se le asigna el temporizador hardware *LEDC_TIMER_0* que se usa para generar la PWM, a *LEDC_MODE* se le asigna el modo de baja velocidad del controlador *LEDC_LOW_SPEED_MODE* que ahorra energía y es adecuado para esta aplicación simple, a *LED_CHANNEL* se le asigna el canal que se usa para el control del led *LEDC_CHANNEL_0*, a *LEDC_DUTY_RES* se le asigna la resolución del duty cycle del PWM *LEDC_TIMER_8_BIT* con una resolución de 8 bits. a *LEDC_FREQ* se le asigna 5000 que establece la frecuencia de la señal PWM en 5kHz, a *LEDC_INTR* se le asigna *LEDC_INTR_DISABLE* que desactivación de las interrupciones del módulo *LEDC*. En la Figura 36 se muestra el segmento de código que permite esta tarea.

Figura 36

Asignación de variables para uso de periférico de control LEDC

```
30  // Configuración PWM para LED
31  #define LEDC_TIMER          LEDC_TIMER_0
32  #define LEDC_MODE           LEDC_LOW_SPEED_MODE
33  #define LEDC_CHANNEL        LEDC_CHANNEL_0
34  #define LEDC_DUTY_RES       LEDC_TIMER_8_BIT
35  #define LEDC_FREQ           5000
36  #define LEDC_INTR           LEDC_INTR_DISABLE
```

Nota: Asignación de variables para uso de periférico de control LEDC, segmento de código de programación.

Para la captura de video es necesario establecer algunos parámetros como *FRAME_SIZE* con *FRAMESIZE_VGA (resolución de 640x480), JPEG_QUALITY* define la calidad de compresión JPEG en un rango de 0 a 63 siendo 0 más compresión, este parámetro se establece en 12 para obtener un equilibrio en calidad y tamaño de archivo. *FB_COUNT* define el número de buffers de imagen en memoria, el valor de 2 permite procesar video sin retrasos. En la Figura 37 se muestra el segmento de código.

Figura 37

Asignación de variables para configuración de cámara

```
38  // Configuración de cámara
39  #define FRAME_SIZE FRAMESIZE_VGA
40  #define JPEG_QUALITY 12
41  #define FB_COUNT 2
```

Nota: Asignación de variables para configuración de cámara, segmento de código de programación.

Para acceder al router se definen el nombre de la red WiFi y la contraseña por medio de una cadena de texto constante *const char* ssid = "TP-LINK_94ADCC"* y *const char* password = ""* respectivamente. El segmento de código se muestra en la Figura 38.

Figura 38

Declaración de variables para conexión con router

```
43  // Configuración WiFi
44  const char* ssid = "TP-LINK_94ADCC";
45  const char* password = "";
```

Nota: Declaración de variables para conexión con router, segmento de código de programación.

Para el manejo del servidor HTTP se declaran dos variables globales *httpd_handle_t* que es un manejador *(handle)* usado por el servidor web HTTP *(parte del componente esp_http_server),* la *httpd_handle_t stream_httpd = NULL* que declara un manejador para un servidor HTTP dedicado al streaming y se inicializa como NULL (aun no creado), la *httpd_handle_t cmd_httpd = NULL* que declara un manejador para un servidor HTTP dedicado a comandos o *API REST* e inicia como *NULL*. Se crean dos servidores separados para optimizar recursos y tener configuraciones independientes. El segmento de código se muestra en la Figura 39.

Figura 39

Declaración de variables para manejo de servidores HTTP independientes

```
47  // Servidores HTTP
48  httpd_handle_t stream_httpd = NULL;
49  httpd_handle_t cmd_httpd = NULL;
```

Nota: Declaración de variables para manejo de servidores HTTP independientes, segmento de código de programación.

Dentro del void setup se establecen las condiciones iniciales de este programa. De la línea 56 a la 60 se establece el modo de los pines que controlan el giro de los motores y como inician estas variables, estos pines son de salida e inician en LOW. En la Figura 40 se muestra este segmento de código.

Figura 40

Configuración de pines para control de motores

```
56  // Configuración de pines del motor
57  pinMode(MOTOR_PIN1, OUTPUT);
58  pinMode(MOTOR_PIN2, OUTPUT);
59  digitalWrite(MOTOR_PIN1, LOW);
60  digitalWrite(MOTOR_PIN2, LOW);
```

Nota: Configuración de pines para control de motores, segmento de código de programación.

Se crea una estructura (struct) de tipo *ledc_timer_config_t* que tiene contiene las variables y sus valores establecidos son las que se asignan aquí. Con *ledc_timer_config(&ledc_timer) se* aplica la configuración de la memoria al hardware. En la Figura 41 se muestra este segmento de código.

Figura 41

Asignación de las variables que contienen los valores para el uso del LEDC

```
62  // Configuración PWM para LED
63  ledc_timer_config_t ledc_timer = {
64    .speed_mode       = LEDC_MODE,
65    .duty_resolution  = LEDC_DUTY_RES,
66    .timer_num        = LEDC_TIMER,
67    .freq_hz          = LEDC_FREQ,
68    .clk_cfg          = LEDC_AUTO_CLK
69  };
70  ledc_timer_config(&ledc_timer);
```

Nota: Asignación de las variables que contienen los valores para el uso del LEDC, segmento de código de programación.

La estructura *ledc_channel_config_t* contiene los parámetros para la configuración de un canal del LEDC, a cada parámetro se le asigna su variable correspondiente que ya fue definida en segmentos pasados, a excepción de *.duty = 85* que inicia el canal con un brillo del 33%, *hpoint = 0* que es el punto de fase que comúnmente se deja en 0. Como puntero a esta estructura se tiene la variable *ledc_channel* y en la línea 81 se aplica la configuración de la memoria al hardware. En la Figura 42 se muestra este segmento de código.

Figura 42

Asignación de las variables que contienen los valores para el uso del LEDC

```
72  ledc_channel_config_t ledc_channel = {
73      .gpio_num       = LED_PIN,
74      .speed_mode     = LEDC_MODE,
75      .channel        = LEDC_CHANNEL,
76      .intr_type      = LEDC_INTR,
77      .timer_sel      = LEDC_TIMER,
78      .duty           = 85,  // Brillo bajo inicial (33%)
79      .hpoint         = 0
80  };
81  ledc_channel_config(&ledc_channel);
```

Nota: Asignación de las variables que contienen los valores para el uso del LEDC, segmento de código de programación.

Para la configuración de la cámara se usa una estructura de tipo *camera_config_t* que contiene todos los parámetros para hacer funcionar el hardware de la cámara. Las variables de asignación de pines y de configuración de cámara son las que se asignan aquí junto a otras que reciben los valores que se muestran en la Figura 43. En la línea 106 se aplica la configuración de la memoria al hardware e inicializa la cámara por medio de la función *esp_camera_init()* y el puntero *config* donde *esp_err_t err* almacena el resultado de inicialización que ayuda a identificar en el serial el tipo de error en caso que haya.

Figura 43

Asignación de las variables que contienen los valores para el uso de la cámara

```
83  // Configuración de la cámara
84  camera_config_t config;
85  config.pin_d0 = Y2_GPIO_NUM;
86  config.pin_d1 = Y3_GPIO_NUM;
87  config.pin_d2 = Y4_GPIO_NUM;
88  config.pin_d3 = Y5_GPIO_NUM;
89  config.pin_d4 = Y6_GPIO_NUM;
90  config.pin_d5 = Y7_GPIO_NUM;
91  config.pin_d6 = Y8_GPIO_NUM;
92  config.pin_d7 = Y9_GPIO_NUM;
93  config.pin_xclk = XCLK_GPIO_NUM;
94  config.pin_pclk = PCLK_GPIO_NUM;
95  config.pin_vsync = VSYNC_GPIO_NUM;
96  config.pin_href = HREF_GPIO_NUM;
97  config.pin_sscb_sda = SIOD_GPIO_NUM;
98  config.pin_sscb_scl = SIOC_GPIO_NUM;
99  config.pin_pwdn = PWDN_GPIO_NUM;
100 config.pin_reset = RESET_GPIO_NUM;
101 config.xclk_freq_hz = 20000000;
102 config.pixel_format = PIXFORMAT_JPEG;
103 config.frame_size = FRAME_SIZE;
104 config.jpeg_quality = JPEG_QUALITY;
105 config.fb_count = FB_COUNT;
106 esp_err_t err = esp_camera_init(&config);
```

Nota: Asignación de las variables que contienen los valores para el uso de la cámara, segmento de código de programación.

La función WiFi.begin inicia la conexión WIFI con las credenciales *ssid* y *password* que se definieron en las líneas 44 y 45. *WiFi.setAutoReconnect(true)* reactiva la conexión automática en caso de desconexión. En la Figura 44 se muestra el segmento de código explicado antes.

Figura 44

Funciones para establecer conexión WiFi

```
113  // Conectar WiFi
114  WiFi.begin(ssid, password);
115  WiFi.setAutoReconnect(true);
```

Funciones para establecer conexión WiFi, segmento de código de programación.

Para la configuración del servidor que recibe los comandos en la línea 135 se crea una variable *cmd_config* de tipo *httpd_config_t* y se inicializa con valores por defecto mediante la macro *HTTPD_DEFAULT_CONFIG().* En la línea 136 *cmd_config.server_port = 80* establece el puerto TCP donde el servidor escuchara las peticiones HTTP, *cmd_config.ctrl_port = 80* define el puerto para le canal de control interno del servidor que normalmente es el mismo puerto que el *server_port*, *cmd_config.max_opwn_socket = 3* limita el número máximo de conexiones simultaneas (sockets) que el servidor aceptará. En la Figura 45 se muestra el segmento de código explicado antes.

Figura 45

Configuración de servidor para los comandos

```
134  // Configurar servidor de COMANDOS (puerto 80)
135  httpd_config_t cmd_config = HTTPD_DEFAULT_CONFIG();
136  cmd_config.server_port = 80;
137  cmd_config.ctrl_port = 80;
138  cmd_config.max_open_sockets = 3;
```

Nota: Configuración de servidor para los comandos, segmento de código de programación.

Las funciones *httpd_start(&cmd_httpd,&cmd_config)* inicia el servidor HTTP con la configuración predefinida *(cmd_config)* y el puntero del servidor *(cmd_http)* que se usa para gestionar el servidor. Para manejar las solicitudes HTTP GET se usan dos endpoints (rutas),

por medio de las estructuras *httpd_urit_t move_uri* y *httpd_urit_t led_uri* que, cuando el cliente acceda a *http://[192.168.0.100]/move* se ejecuta la función move_handler encargada del accionamiento de los motores, de la misma forma si el cliente accede a *http://[192.168.0.100]/led* se ejecuta la función led_handler encargada del control del brillo led. Los *httpd_register_uri_handler()* de la linea 153 y 154 asocian una ruta URL como */move* o */led* con una función manejadora que se ejecuta cuando el cliente accede a esa ruta, el parámetro *&cmd_httpd* es el manejador del servidor web y *&move_uri* o *&led_uri* los punteros a las estructuras que definen la ruta y su manejador. La Figura 46 muestra el segmento de código explicado antes.

Figura 46

Configuración de servidor para los comandos e inicio del servidor HTTP y su función manejadora encargada de los comandos para el streaming

```
140   if (httpd_start(&cmd_httpd, &cmd_config) == ESP_OK) {
141     httpd_uri_t move_uri = {
142       .uri = "/move",
143       .method = HTTP_GET,
144       .handler = move_handler,
145       .user_ctx = NULL
146     };
147     httpd_uri_t led_uri = {
148       .uri = "/led",
149       .method = HTTP_GET,
150       .handler = led_handler,
151       .user_ctx = NULL
152     };
153     httpd_register_uri_handler(cmd_httpd, &move_uri);
154     httpd_register_uri_handler(cmd_httpd, &led_uri);
155     Serial.println("Servidor de comandos iniciado en puerto 80");
156   }
```

Nota: Configuración de servidor para los comandos e inicio del servidor HTTP y su función manejadora encargada de los comandos para el streaming, segmento de código de programación, segmento de código de programación.

De igual forma que para el servidor de comandos se configura este para el manejo del streaming, pero en el puerto 81. Solo se usa un endpoint *stream_uri* que al cliente acceder a *http://[192.168.0.100]/stream* se ejecuta la función stream_handler. El *httpd_register_uri_handler()* de la linea 170 se asocia a la ruta */stream*, el parámetro *&stream_httpd* es el manejador del servidor web y *&stream_uri* el puntero a la estructura que definen la ruta y su manejador. La Figura 47 muestra el segmento de código explicado antes.

Figura 47

Inicia el servidor HTTP y su función manejadora encargada de los comandos para motores y led

```
158   // Configurar servidor de STREAMING (puerto 81)
159   httpd_config_t stream_config = HTTPD_DEFAULT_CONFIG();
160   stream_config.server_port = 81;
161   stream_config.ctrl_port = 81;
162
163   if (httpd_start(&stream_httpd, &stream_config) == ESP_OK) {
164     httpd_uri_t stream_uri = {
165       .uri = "/stream",
166       .method = HTTP_GET,
167       .handler = stream_handler,
168       .user_ctx = NULL
169     };
170     httpd_register_uri_handler(stream_httpd, &stream_uri);
171     Serial.println("Servidor de streaming iniciado en puerto 81");
172   }
173 }
```

Nota: Inicia el servidor HTTP y su función manejadora encargada de los comandos para motores y led, segmento de código de programación.

La línea 179 contiene la declaración del handler que controla el brillo del led, recibe como parámetros el puntero a la estructura que contiene los datos de la solicitud HTTP. En la línea 180 se guarda en una variable el resultado de convertir el request (req) en su longitud en bytes, la línea siguiente comprueba si existen parámetros en el req y si los hay asigna memoria dinámica para almacenar el query string (parte de req después del ?) y luego convertirlo en char, todo esto por medio de la línea 182.

La función *httpd_req_get_url_query_str* copia el query string en el buffer *(buf)* y retorna un *ESP_OK* , si se cumple la == la función *strstr* busca la subcadena *"brightness=low"* dentro de *buf* y si la encuentra retorna la posición del puntero a esa posición. Ledc_set_duty() establece el ciclo de trabajo a 85 que corresponde a un 33% del brillo, y ledc_update_duty() aplica el cambio inmediatamente, de la misma forma para cuando la subcadena *"brightness=medium"* o *"brightness=high"* pero con los valores de 170 (un 66% de brillo) y 255 (un 100% de brillo) respectivamente. Free(buff) libera la memoria dinámica reservada anteriormente por malloc() para el buf, la línea 202 envía una respuesta de confirmación al cliente con el mensaje "OK" de que la solicitud se procesó. En la Figura 48 se muestra el segmento de código descrito anteriormente.

Figura 48

Handler para control de brillo del led

```
179  static esp_err_t led_handler(httpd_req_t *req) { char* buf;
180    size_t buf_len = httpd_req_get_url_query_len(req) + 1;
181    if (buf_len > 1) {
182      buf = (char*)malloc(buf_len);
183      if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
184        if (strstr(buf, "brightness=low")) {
185          ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, 85);
186          ledc_update_duty(LEDC_MODE, LEDC_CHANNEL);
187          Serial.println("LED: Brillo bajo (33%)");
188        }
189        else if (strstr(buf, "brightness=medium")) {
190          ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, 170);
191          ledc_update_duty(LEDC_MODE, LEDC_CHANNEL);
192          Serial.println("LED: Brillo medio (66%)");
193        }
194        else if (strstr(buf, "brightness=high")) {
195          ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, 255);
196          ledc_update_duty(LEDC_MODE, LEDC_CHANNEL);
197          Serial.println("LED: Brillo alto (100%)");
198        }
199        free(buf);
200      }
201    }
202    httpd_resp_send(req, "OK", HTTPD_RESP_USE_STRLEN); return ESP_OK;
203  }
```

Nota: Handler para control del brillo del led, segmento de código de programación.

El handler para el control del sentido de giro de los motores tiene la misma estructura que la del control del led pero las query string que busca son *"dir=forward"* y pone en alto el MOTOR_PIN1 y bajo el MOTOR_PIN2, si es *"dir=backward* pone en bajo el MOTOR_PIN1 y alto el MOTOR_PIN2. En la Figura 49 se muestra el segmento de código descrito anteriormente.

Figura 49

Handler para control de sentido de giro de motores

```
206  static esp_err_t move_handler(httpd_req_t *req) {char* buf;
207    size_t buf_len = httpd_req_get_url_query_len(req) + 1;
208    if (buf_len > 1) {
209      buf = (char*)malloc(buf_len);
210      if (httpd_req_get_url_query_str(req, buf, buf_len) == ESP_OK) {
211        if (strstr(buf, "dir=forward")) {
212          digitalWrite(MOTOR_PIN1, HIGH);
213          digitalWrite(MOTOR_PIN2, LOW);
214          Serial.println("Motor: Adelante");
215        }
216        else if (strstr(buf, "dir=backward")) {
217          digitalWrite(MOTOR_PIN1, LOW);
218          digitalWrite(MOTOR_PIN2, HIGH);
219          Serial.println("Motor: Atrás");
220        }
221        else {
222          digitalWrite(MOTOR_PIN1, LOW);
223          digitalWrite(MOTOR_PIN2, LOW);
224          Serial.println("Motor: Detenido");
225        }
226        free(buf);
227      }
228    }
229    httpd_resp_send(req, "OK", HTTPD_RESP_USE_STRLEN); return ESP_OK;
230  }
```

Nota: Handler para control de sentido de giro de motores, segmento de código de programación.

Para el control del video se declara el handler stream_handler, básicamente se sigue el patrón de estructura de los handlers anteriores. Las variables locales manejadas aquí son las de las líneas 234 a 238; frame buffer de la cámara, validación por parte de cliente, longitud de buffer

JPEG, buffer para imagen JPEG y buffer para partes del mensaje multipart. En la línea 239 se procesa el tipo de respuesta, *multipart/x-mixed-replace* es el protocolo de streaming donde cada frame reemplaza al anterior, *boundary=....* es la cadena única que separa cada frame en el stream. En la Figura 50 se muestra el segmento de código descrito anteriormente.

Figura 50

Handler para control de streaming

```
233  static esp_err_t stream_handler(httpd_req_t *req) {
234    camera_fb_t *fb = NULL;
235    esp_err_t res = ESP_OK;
236    size_t _jpg_buf_len = 0;
237    uint8_t *_jpg_buf = NULL;
238    char *part_buf[64];
239    httpd_resp_set_type(req, "multipart/x-mixed-replace;boundary=123456789000000000000987654321");
```

Nota: Handler para control de streaming, segmento de código de programación.

Dentro de un ciclo while, la variable fb obtiene un frame de la cámara, si falla (fb==NULL) se sale del bucle, de la línea 246 a 255 se hace la conversión a JPEG, si el formato no es JPEG se hace la conversión usando frame2jpg() al valor definido previamente en la variable JPEG_QUALITY, jpg_buf y _jpg_buf_len almacenan el buffer y la longitud de JPEG resultante, si el frame ya está en JPEG usa directamente el buffer y longitud del frame.

Figura 51

Handler para control de sentido de streaming

```
240   while (true) {
241     fb = esp_camera_fb_get();
242     if (!fb) {
243       res = ESP_FAIL;
244       break;
245     }
246     if (fb->format != PIXFORMAT_JPEG) {
247       bool jpeg_converted = frame2jpg(fb, JPEG_QUALITY, &_jpg_buf, &_jpg_buf_len);
248       if (!jpeg_converted) {
249         res = ESP_FAIL;
250         break;
251       }
252     } else {
253       _jpg_buf_len = fb->len;
254       _jpg_buf = fb->buf;
255     }
```

Nota: Handler para control de streaming, segmento de código de programación.

El encabezado HTTP es contiene los metadatos que dan información sobre cómo deben ser tratada la información al cliente, específicamente los multipart indica por medio del delimitador boundary que el cuerpo de la solicitud contiene varios datos. De la línea 256 a la 261 con spnrintf se formatea la variable que contiene los datos multipart cada vez que se procesa un frame en el stream, part_buf contiene el buffer de 64 bytes para almacenar el encabezado. Si los datos del part_buf que indican el tamaño y tipo del JPEG, _jpg_buf los bytes de la imagen, y el delimitador de 38 bytes son ==ESP_OK los datos se envían al cliente sino se sale del bucle.

Figura 52

Handler para control de sentido de streaming

```
256     size_t hlen = snprintf((char *)part_buf, 64, "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n", _jpg_buf_len);
257     if (httpd_resp_send_chunk(req, (const char *)part_buf, hlen) != ESP_OK ||
258         httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len) != ESP_OK ||
259         httpd_resp_send_chunk(req, "\r\n--123456789000000000000987654321\r\n", 38) != ESP_OK) {
260       res = ESP_FAIL;
261     }
262     if (fb->format != PIXFORMAT_JPEG) {
263       free(_jpg_buf);
264     }
265     esp_camera_fb_return(fb);
266     if (res != ESP_OK) {
267       break;
268     }
269   }
270   return res;
271 }
```

Nota: Handler para control de streaming, segmento de código de programación.

Diseño de interfaz de control y video cámara del mini robot

La interfaz que permite el control y visualización del streaming del mini robot es una interfaz HTML que contiene el código en JavaScript. Esta interfaz HTML cuenta con un <head>, este el segmento CSS que define el diseño visual de la interfaz. La paleta de colores se define en la sección de código de la Figura 53 que marca una tendencia a colores oscuros con azules profundos y los colores para el estado de activo o inactivo del stream.

Figura 53

Sección HTML que define los colores de la interfaz HTML

```
:root {
    --primary-dark: #1a1a2e;
    --secondary-dark: #16213e;
    --button-color: #0f3460;
    --button-active: #00b4d8;
    --text-light: #f1f1f1;
    --stream-off: #ff0000;
    --stream-on: #00ff00;
}
```

Nota: Sección de HTML que define los colores de la interfaz HTML. Segmento de código

La interfaz está distribuida en una proporción de 70% para el video y 30% para los controles, los colores asignados también forman parte de la paleta de colores general, también el HTML es responsivo y se ajusta a dispositivos móviles cambiando de 60vh a 40vh. En la Figura 54 se muestra el segmento de código que contiene lo descrito.

Figura 54

Sección de HTML con la distribución del dashboard

```
.dashboard {
    display: grid;
    grid-template-columns: 70% 30%;
    height: 100vh;
}

/* Responsividad */
@media (max-width: 768px) {
    .dashboard {
        grid-template-columns: 1fr;
        grid-template-rows: 60vh 40vh;
    }
}

body {
    background-color: #0d0d1a;
}
```

Nota: Sección de HTML con la distribución del dashboard. Segmento de código

El panel de video contiene un fondo negro puro (#000) para generar un mejor contraste, un flexbox centrado para el video, transiciones suaves para las rotaciones del video stream y animaciones de pulso para el estado de conexión. En la Figura 55 se muestra el segmento de código que contiene lo descrito.

Figura 55

Sección de HTML que muestra el video streaming

```
.video-panel {
    background-color: #000;
    display: flex;
    flex-direction: column;
    position: relative;
}

.video-container {
    flex-grow: 1;
    display: flex;
    justify-content: center;
    align-items: center;
    background-color: #000;
    overflow: hidden;
}

.video-feed {
    transition: transform 0.3s ease;
    transform-origin: center center;
}

.video-feed.connecting {
    opacity: 0.5;
    animation: pulse 1.5s infinite;
}

@keyframes pulse {
    0% { opacity: 0.6; }
    50% { opacity: 1; }
    100% { opacity: 0.6; }
}
```

Nota: Sección de HTML que muestra el video streaming. Segmento de código.

La sección del panel de control contiene colores oscuros, sobras que dan un efecto de profundidad y un scroll vertical por si existe un overflow, los botones cuentan con esquinas redondeadas, sombras por efecto de profundidad y bordes sutiles. En la Figura 56 se muestra el segmento de código que contiene lo descrito.

Figura 56

Sección de estilo de los botones

```
.control-panel {
    background: linear-gradient(145deg, var(--primary-dark), var(--secondary-dark));
    padding: 15px;
    display: flex;
    flex-direction: column;
    box-shadow: -5px 0 15px rgba(0,0,0,0.5);
    overflow-y: auto;
}
.controller {
    background-color: var(--secondary-dark);
    border-radius: 15px;
    padding: 15px;
    margin: 8px 0;
    box-shadow: inset 0 0 10px rgba(0,0,0,0.3);
    border: 2px solid rgba(255,255,255,0.05);
}
```

Nota: Sección de HTML da estilo a los botones y panel de control. Segmento de código.

Los botones son interactivos, tienen efectos de hover/active lo que cambia su estética ante interacciones, en este caso existen transiciones al cambiar de estética, también sus colores cambian, pasando de un gris a un celeste en respuesta al clic. En la Figura 57 se muestra el segmento de código que contiene lo descrito.

Figura 57

Sección de HTML da estilo a los botones y los hace interactivos visualmente

```
.btn {
    border: none;
    color: var(--text-light);
    font-weight: bold;
    cursor: pointer;
    transition: all 0.2s ease;
    border-radius: 10px;
    box-shadow: 0 3px 6px rgba(0,0,0,0.3);
    position: relative;
    overflow: hidden;
}

.btn:active {
    transform: translateY(2px);
    box-shadow: 0 2px 3px rgba(0,0,0,0.3);
}

.btn-toggle {
    background-color: var(--button-color);
    padding: 12px;
    margin: 8px 0;
    font-size: 0.95rem;
    width: 100%;
}

.btn-toggle.active {
    background-color: var(--button-active);
}
```

Nota: Sección de HTML da estilo a los botones y los hace interactivos visualmente. Segmento de código.

Los botones de control de movimiento son de un tamaño más grande en comparación a los demás, de 70x70 y para dispositivos móviles de 100x100, esto contienen un feedback táctil con scale 0.98 al activarlo. Los botones para el control de brillo tienen características similares a las descrita anteriormente.

Los demás segmentos de código que se encargan por ejemplo de enviar las variables de acción e información de video se encuentran completos en la sección de anexos de este documento.

Integración de todos los elementos del mini robot móvil bidireccional para sus respectivas pruebas en un entorno controlado.

Para que el HTML y el ESP32 CAM puedan funcionar en conjunto es fundamental generar un medio por el cual puedan comunicarse, y esta es la función que cumple el router. En la Figura 52 se muestra un diagrama de flujo con tres secciones, de izquierda a derecha todas las funciones y líneas de flujo que componen las funcionalidades del ESP32 CAM, el flujo de trabajo empieza con la inicialización del ESP32 CAM seguido de su conexión a red por medio de WiFi, inicialización de servidores HTTP en los puertos 80 y 81, a partir de allí el ESP32 CAM espera las solicitudes de *move*, *led* o *stream* que provienen de la interfaz HTML y según la solicitud ejecuta las acciones de movimiento, brillo de led y transmisión de video, las dos primeras envían una respuesta ESP OK que es propia del ESP32 CAM, la transmisión de video realiza tres procesos antes de enviar el frame de video hacia el HTML, captura el frame, lo convierte a JPGE y lo envía multipart.

Por el lado del HTML se inicia la página y previamente el equipo cliente (en este caso la computadora) debe estar ya conectado a la misma red. El usuario desde el HTML puede interactuar con cuatro parámetros, tres manejadas en el ESP32 CAM y una dentro del mismo HTML. Si se da clic en cualquier de los botones de la sección de movimiento se envía la solicitud *move* hacia el ESP32 CAM quien la compara y valida dando paso a que los motores

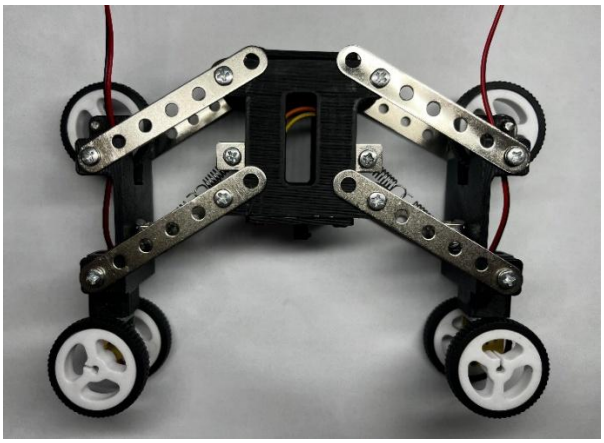
se mueven en un sentido o en otro. Al dar clic en cualquier de los botones de la sección del brillo led se envía la solicitud *led* hacia el ESP32 CAM quien la compara y valida dando paso al encendido del led en el nivel de brillo deseado. Para el streaming se envía la solicitud *stream* al ESP32 CAM al dar clic al botón “INICIAR STREAM” y se detiene la transmisión al dar clic al mismo botón quien cambia su estado a “DETENER STREAM”. El parámetro de rotación no necesita enviar ninguna solicitud al ESP32 CAM pues este se maneja dentro del HTML, por lo que al dar clic al botón “ROTAR” la imagen se rota 90 grados en sentido horario y el estado actual en grados se muestra en la misma interfaz.

RESULTADOS

Al desarrollar e imprimir cada una de las partes y mecanismos diseñado en CAD los cuales se encuentran descritos en la sección de metodología y cuyas dimensiones están acotadas en los planos adjuntos en el anexo y, posteriormente ensamblar cada una de ellas se obtiene como resultado toda la estructura de este mini robot. En la Figura 58 se muestra el cuerpo y mecanismo de desplazamiento y auto acople de este.

Figura 58

Cuerpo y mecanismo de desplazamiento y auto acople del mini robot móvil

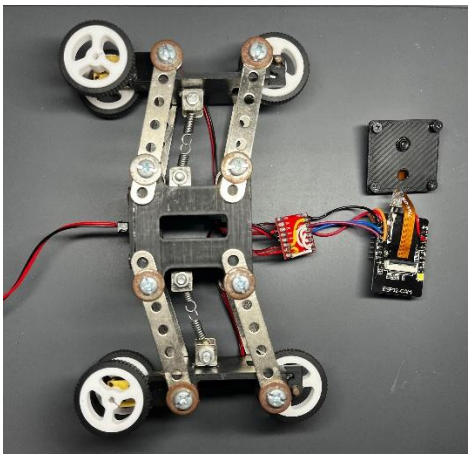


Nota. Resultado posterior al ensamble de cada una de las piezas mecánicas y de estructura del mini robot.

Tomando como referencia el esquema de conexiones de las Figura 31 se implementa el hardware de control y de video para el mini robot, conectando cada uno de los pines del ESP32 CAM con el módulo TBA6612FN y este con cada uno de los dos motores y todo esto se coloca dentro del cuerpo del mini robot. En la Figura 59 se muestra todo el hardware fuera del robot para poder visualizarlo en su totalidad.

Figura 59

Implementación de hardware de control y video para el mini robot

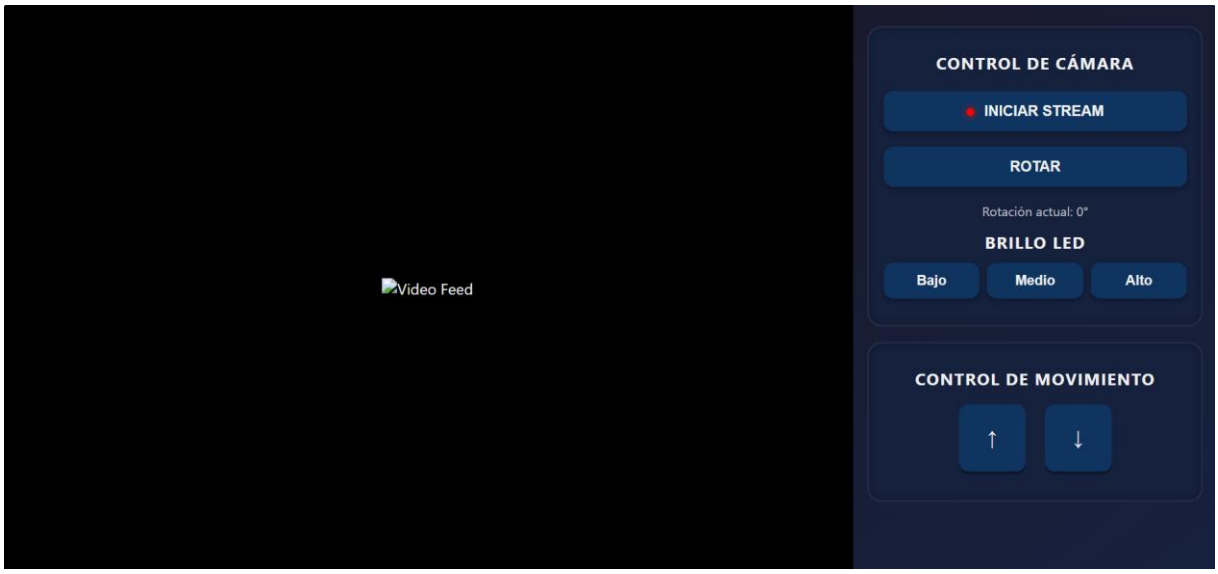


Nota. El hardware de control y video va dentro del mini robot, pero para su mejor visualización se lo extrajo cuidando no desconectar ninguna conexión entre módulos y elementos.

Para el control del mini robot es necesario un entorno que permita una interacción intuitiva y amigable con el operador. El proceso de creación de esta interfaz es explicado paso a paso en la metodología y, en la Figura 60 se muestra el resultado de ello, una interfaz simple pero funcional con botones que cambian su estado y aspecto al interactuar con ellos, una sección amplia donde se puede ver el video en vivo capturado por el mini robot.

Figura 60

Interfaz HTML para el control y monitoreo del mini robot

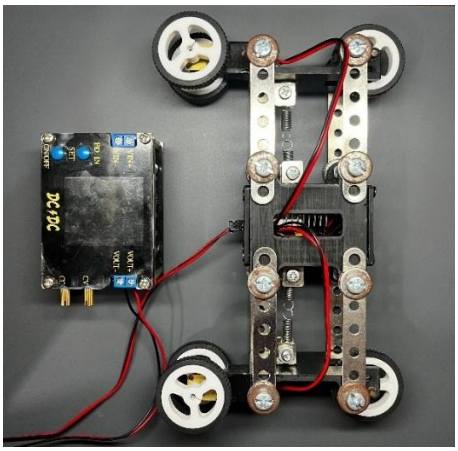


Nota. Esto es lo primero que se muestra la abrir la interfaz HTML en cualquier navegador.

Para validar el funcionamiento del mini robot móvil se lo hace dentro de un tubo de PVC de uso común, unos 150 mm de diámetro externo y 147 mm de diámetro interno con una longitud de un metro de largo. Para la puesta en marcha del robot primero, para introducirlo, se ubica su parte frontal, esta es donde se ubica la cámara y el led, seguido, con ambas manos se comprimirme el robot lo suficiente como para que entre dentro del tubo. Ahora, se deben conectar las terminales de alimentación (cable negro y rojo) a la fuente DC respetando la polaridad establecida por el código de colores, rojo VCC y negro GND, si la fuente es externa como en este caso esta debe estar apagada durante el proceso de conexión.

Figura 61

Conexión física entre fuente DC y mini robot



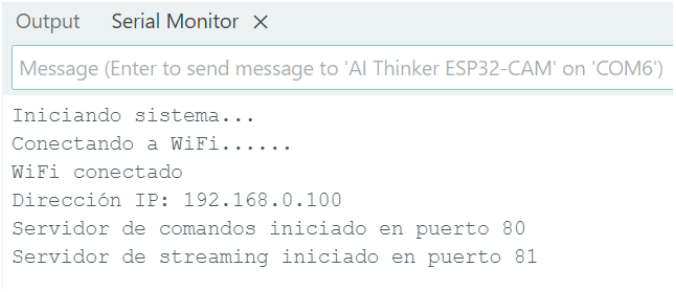
Nota: Se usa una fuente DC como alimentación para el mini robot, pero puede ser sustituida por cualquier otra fuente de alimentación DC de 5V como una batería.

Tanto el mini robot con el equipo que lo controla debe estar en la misma red, para ello se enciende el router y la fuente DC, en unos segundos el mini robot ya está conectado a la red WiFi automáticamente, luego se debe conectar el dispositivo de control que para este caso es una laptop a la misma red, ya sea por WiFi o mediante cable Ethernet directamente a una de las terminales RJ45 del router. Hacerlo en este orden es importante pues el router debe asignar una IP primero al robot y luego al dispositivo de control, la IP asignada corresponde a la 192.168.0.100.

Para verificar si el mini robot se encuentra conectado a la red y si se asignó la IP correcta se puede usar el módulo serial CH340. Se coloca sobre el socket del módulo serial el ESP32 CAM, por medio de un cable de puerto USB C a USB B/C que permita transmisión de datos se conecta hacia un puerto de la computadora y con la aplicación Arduino IDE por medio del monitor serial se puede verificar si el ESP32 CAM se conectó a la red y que dirección se le asignó como se muestra en la Figura 62.

Figura 62

Mensaje de confirmación de conexión de a la red del ESP32 CAM.



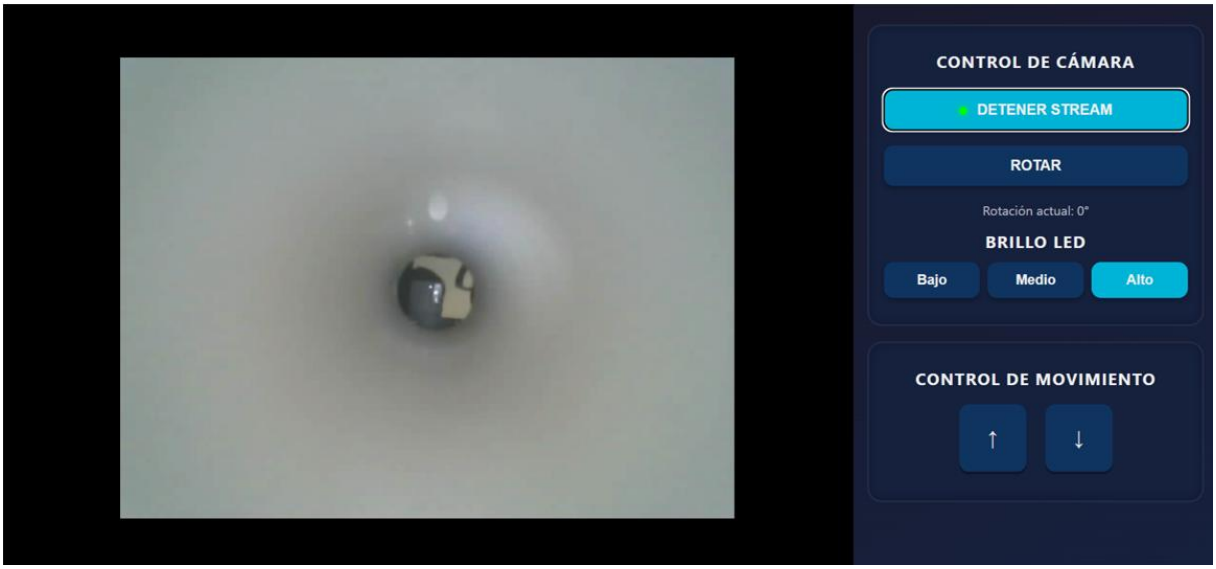
Nota: Mensaje de confirmación de conexión de a la red del ESP32 CAM en Arduino IDE.

Abriendo el archivo .html en cualquier navegador se puede acceder a la interfaz de control del mini robot. Una vez dentro se puede controlar el desplazamiento del mini robot con los dos botones interactivos, de las misma forma el brillo del led puede ser controlado por medio de tres botones que ajustan el brillo en tres niveles: bajo, medio y alto y del lado, el botón “ROTAR” permite la rotación de la imagen del streaming en intervalos de 90 grados en sentido horario, el botón “INICIAR STREAM ” habilita o deshabilita la transmisión de video en tiempo real que captura la cámara del mini robot y la sección donde se muestra el video capturado por el robot.

El mini robot cuenta con un led de luz blanca que puede ser controlado por medio de los botones “Bajo”, “Medio” y “Alto”. Antes de iniciar el stream se debe encender el led para poder observar lo que está dentro de la tubería. Al presionar “INICAR STREAM”, este cambia de estado a “DETENER STREAM” y su indicador rojo pasa a estar en verde, inmediatamente en la sección izquierda se muestra el video en tiempo real de lo que la cámara del mini robot está viendo. En la Figura 63 se muestra la interfaz con lo que está capturando la cámara en ese momento.

Figura 63

Interfaz de control y monitoreo de mini robot

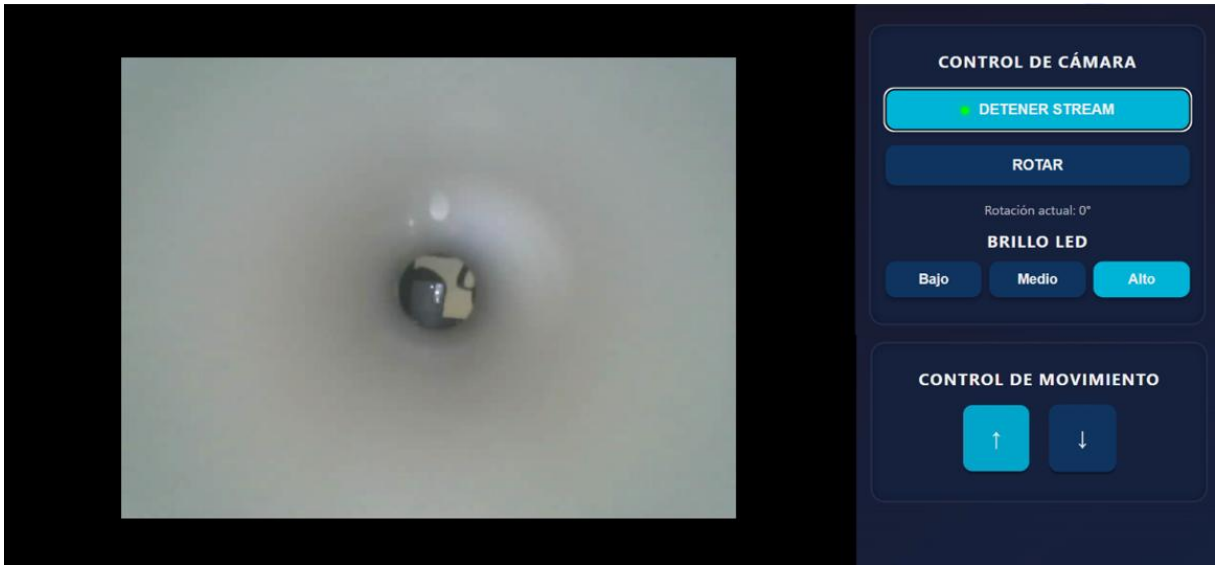


Nota: Interfaz de control y monitoreo de mini robot con la transmisión del streaming y led encendido al brillo más alto.

Al presionar los botones de la sección de movimiento el robot avanza hacia delante al presionar el botón con la flecha hacia arriba y hacia atrás si se presiona la flecha hacia abajo como se muestra en la Figura 64.

Figura 64

Interfaz de control y monitoreo de mini robot.

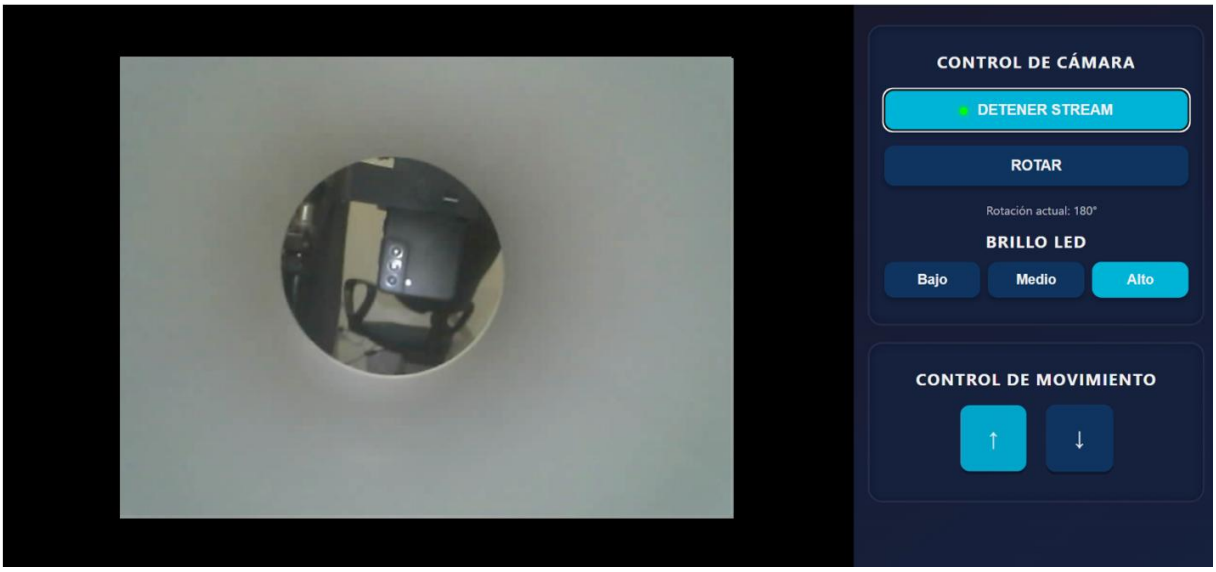


Nota: Interfaz de control y monitoreo con la transmisión del streaming, el brillo del led alto y el desplazamiento hacia delante del mini robot.

Otra función útil en la interfaz es la rotación del video stream, al no tener una parte superior o inferior definida el mini robot puede ser ubicado dentro de la tubería de la cualquier forma y desde la interfaz se puede rotar el streaming presionando el botón “ROTAR” en intervalos de 90 grados y en sentido horario a la vez que se puede saber cuánto se encuentra rotado el stream con el texto que se encuentra bajo el botón, en la Figura 65 se muestra cómo se ve el stream al hacer una rotación de 180 grados.

Figura 65

Interfaz de control y monitoreo de mini robot.



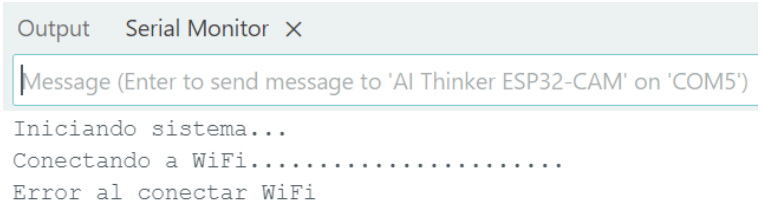
Nota: Interfaz de control y monitoreo con la transmisión del streaming rotada a 180 grados mientras el robot desplazamiento hacia delante y el led está encendido a brillo alto.

Cuando el robot haya cumplido con su función simplemente se puede apagar desde la fuente que lo alimenta, no existe un orden a diferencia de cuando se enciende. El robot se puede desplazar con o sin el streaming encendido, aunque mientras está el stream activado pueden presentarse pequeños retardos desde que se presionan los botones del control de movimiento hasta que el mini robot ejecuta el desplazamiento, lo mismo sucede durante la transmisión del video puede presentar intermitencias o desconexiones que ameritan un reinicio del sistema o simplemente una detención y encendido del stream desde el botón en la interfaz del HTML.

Según las pruebas realizadas; tres de cada 10 inicializaciones del sistema presentan problemas al intentar conectarse a la red WiFi, en la Figura 66 se muestra en el monitor serial de Arduino IDE el mensaje de error al ESP32 CAM intentar conectarse al WiFi.

Figura 66

Mensaje de error del ESP32 CAM al intentar conectarse a la red.



Nota: Mensaje de error del ESP32 CAM al intentar conectarse a la red en Arduino IDE.

De cada 15 órdenes de control de movimiento tres no son ejecutadas o accionan la orden en un bucle lo que hace que el robot se desplace, aunque se haya dejado de pulsar los botones de control de movimiento. El stream al ser la tarea más exigente es la que más presenta problemas, en las pruebas, seis de cada 10 intentos para encender y obtener el video desde la cámara del CAM ESP32 fallan y de los seis aciertos los seis presentan intermitencias, lag, desconexiones y reconexiones periódicas, es decir, que es 100% probable que se presenten durante la operación del mini robot.

CRONOGRAMA DE ACTIVIDADES

En la Tabla 3 se mencionan cada una de las actividades ejecutadas para la creación del mini robot.

Tabla 3

Cronograma de actividades

Actividades	**Tiempo**							
	Meses	**Mayo**				**Junio**		
	Semanas	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Elección de tema de proyecto de titulación.		✓						
Definición de objetivos generales y específicos.		✓						
Búsqueda de elementos mecánicos y electrónicos para el mini robot.			✓					
Redacción de justificación, marco teórico, metodología y presupuesto.				✓	✓			
Diseño CAD del cuerpo y mecanismo de desplazamiento y acoplamiento del robot.						✓		
Impresión 3D de mecanismos y piezas del robot.							✓	
Programación de control y de video en la placa CAM-ESP32.							✓	

Actividades	Tiempo							
	Meses	**Mayo**				**Junio**		
	Semanas	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Pruebas con la video cámara.							✓	
Ensamble de elementos mecánicos, estructurales y electrónicos del robot.								✓	
Pruebas electromecánicas del robot.								✓	
Pruebas finales del robot en entorno preparado y controlado.									✓

Nota: Estimación del momento en el que se ejecutaron cada una de las actividades durante la creación de este mini robot.

PRESUPUESTO

En la Tabla 4 se especifica lo invertido en el desarrollo de este mini robot.

Tabla 4

Presupuesto del proyecto del mini robot

DESCRIPCION	COSTO UNIDAD	CANTIDAD	COSTO TOTAL
Horas de ingeniería empleadas	$2,94	120 horas	$352,80
Filamento PLA+ para impresión 3D	$20,00	0,5 kg	$10
Módulo DRV8833	$1,90	1	$1,90
Módulo TB6612FNG	$1,52	1	$1,50
10 muelles de 20x0,6x6mm	$1,72	2	$3,44
10 muelles de 15x0,6x5mm	$1,70	2	$3,40
Modulo ESP32-CAM	$7,21	1	$7,21
Motor reductor N20 DC 5V 850RPM de doble eje	$3,91	2	$7,82
10 ruedas de goma	$3,59	1	$3,59
5 metros de cable de cobre Cable dual 26AWG	$2,26	1	$2,26
2 ruedas de goma pequeñas	$2,10	2	$4,20
Cámara de 75mm y 120 grados	$4,39	1	$4,39
		TOTAL	$402,51

Nota: Descripción, costo, cantidad y valor total de los componentes y tiempo de trabajo empleados en este proyecto.

CONCLUSIONES

- Aunque es mínima, en unas cuentas ocasiones las ruedas que se encuentra frente al robot no tienen contacto con la superficie de la tubería pues durante el diseño no se consideró que las llantas traseras hacen contacto primero con el tubo, esto provoca unos cuantos atascos.
- Mientras más cerca o menos obstáculos existan entre el mini robot y el router mejor será el control y transmisión de video hacia el dispositivo de control y monitoreo.
- Una buena disipación en la cámara OV3660 y el chip del CAM-ESP32-S mejora notablemente la estabilidad y rendimiento del mini robot en el video y la ejecución de los comandos.
- Al ser un robot modular todas sus piezas son fácilmente reemplazables, sobre todo sus eslabones que son los que permiten cambiar la altura del robot dándole la capacidad para navegar dentro de tuberías de distintos diámetros.
- Aunque el cuerpo del mini robot es capaz de albergar una batería de litio no se optó por ello pues es un prototipo inicial e incorporarla incrementaba el tiempo de implementación y el valor de la inversión.
- El módulo CAM-ESP32-S tiene el rendimiento justo para transmitir video en calidad hasta SVGA sin perder estabilidad, para esta aplicación se lo limita a VGA pues también se debe encargar del manejo de comandos para la ejecución del accionamiento de los motores y de la luz led, pero aun así la demanda de recursos se ve sobrepasada por lo que ver el streaming y controlar el robot al mismo tiempo genera inestabilidad en ambas tareas.
- Gracias a que opera con 5v sus motores alcanzan velocidades altas y con la caja reductora de alto torque y bajas RPM’s el mini robot podría desplazarse por tuberías verticales.

RECOMENDACIONES

- La solución a los problemas de atascos es: colocar dos ruedas delanteras o subir unos tres milímetros el eje de las ruedas frontales.
- Para evitar pérdidas de información y mejorar la distancia de operación se puede incorporar una antena al conector IPX que está en el chip.
- Colocar disipadores de temperatura al chip del ESP32 CAM y a la cámara.
- Siempre asegurarse de que tanto el mini robot como el dispositivo de control y monitoreo estén conectados al mismo router y respetando el orden de conexión (primero se debe conectar el mini robot al router y luego el dispositivo).
- Se puede modificar el código de programación del robot como de la interfaz HTML para que usen IP's estáticas, con esto no es necesario seguir un orden al momento de conectarse a la red WiFi del router.
- El cuerpo del mini robot tiene espacio suficiente para albergar una batería de litio lo que le daría la capacidad de ser 100% inalámbrico
- El uso de un módulo más potente como un ESP32-S3 CAM puede dotar y mucho mejor rendimiento al mini robot que podría operar sin conflictos entre el video streaming y los controles de desplazamiento o control del brillo del led o subir la resolución de la transmisión de video hasta en HD lo que sí es posible gracias al módulo de cámara OV3660.
- Cambiar las ruedas por unas de mejor agarre y cambiar los muelles por unos más fuertes le darían al mini robot el suficiente agarre como para desplazarse por tuberías fácilmente.
- Las fallas en la transmisión de video se deben a: distintas direcciones IP asignadas por el router al no seguir el orden de conexión de los dispositivos. Dispositivo de control (laptop) no conectada al router. Robot no conectado al router. Saturación de

información. Distancia u obstáculos de mini robot con el router o dispositivo de control y router.

REFERENCIAS BIBLIOGRÁFICAS

Amazon. (2025). *Amazon*. Obtenido de https://www.amazon.com/-/es/agujero-adecuado-neum%C3%A1tico-juguetes-piezas/dp/B09P46WMBJ

Canelo, T., Gamboa, E., Brunete, A., & Hernando, M. (2015). Micro-robot para inspeccion de tuberias. *XXXVII Jornadas de Automáticas*, (pág. 6). Madrid.

creativaKids. (2023). *creativaKids*. Obtenido de creativaKids: https://creativakids.com/mbot.php

De Miguel, J. (24 de Diciembre de 2015). *Archivo Digital UPM.* Obtenido de FISURAS Y GRIETAS: https://oa.upm.es/74693/1/Fisuras_Grietas.pdf

Díaz, M., & Sarmiento, T. (2015). *Estudio y Diseño de un robot explorador de ductos mediante captura de imagenes en tiempo real utilizando interface grafico remoto [Trabajo de titulacion de Grado, Universidad del Azuay].* Repositorio institucional, Cuenca, Ecuador. Obtenido de https://dspace.uazuay.edu.ec/handle/datos/4860

Estrella C, Q. J. (2024). *Implementacion de un robot teleoperado para inspeccion de tuberias [Trabajo de titulacion de Grado, Universidad Politecnica Salesiana].* Repositorio Institucional, Guayaquil, Ecuador. Obtenido de https://dspace.ups.edu.ec/handle/123456789/27767

Floyd, T., & Buchla, D. (2014). *Electronics Fundamentals Circuits, Devices and Applications.* Pearson Education Limited.

Gonzales, J. (5 de Agosto de 2021). *ENGI-LEARN*. Obtenido de https://www.engi-learn.com/post/sistemas-de-tuber%C3%ADas-piping

Grupo ElectroStore. (2019). *Electrostore*. Obtenido de Electrostore: https://grupoelectrostore.com/shop/motores/micromotores/micro-motorreductor-n20-2-ejes-6v-190rpm/

Instituto de ingenieria de la UNAM. (7 de Mayo de 2018). *Instituto de ingenieria UNAM*. Obtenido de https://www.iingen.unam.mx/es-mx/Investigacion/Proyecto/Paginas/Diagnosticoparatuberias.aspx

Kim, J.-H. a. (2010). FAMPER: A fully autonomous mobile robot for pipeline exploration. *Researchgate,*, (517 - 523),.

López. (04 de Agosto de 2023). *Colecciones Digitales UDLAP, Catarina, Mx.* Obtenido de CAPITULO 3 implementacion de un sistema mecanico: https://catarina.udlap.mx/u_dl_a/tales/documentos/lep/lopez_r_lc/capitulo3.pdf

Naylamp Mechatronics. (2023). *Naylamp Mechatronics*. Obtenido de Naylamp Mechatronics: https://naylampmechatronics.com/drivers/200-driver-puente-h-tb6612fng.html

NayLamp mechatronics. (2023). *NayLamp mechatronics*. Obtenido de NayLamp mechatronics: https://naylampmechatronics.com/espressif-esp/700-esp32-cam-con-camara-ov2640-esp32-wifi-base-ch340.html

Panatec agua. (25 de Mayo de 2018). *Panatec agua*. Obtenido de Robot de inspección de tuberías iPEK ROVVER: https://www.panatec-agua.com/robot-inspeccion-ipek.php

Reyes. (11 de Julio de 2024). *Inspenet*. Obtenido de Inspenet: https://inspenet.com/articulo/soluciones-corrosion-en-tuberias-enterradas/

Rico, E. (2012). *Diseño de un robot de inspección y vigilancia [Trabajo de titulacion de Grado, Instituto Politénico Nacional].* Repositorio institucional, Mexico D.F.,

México. Recuperado el 29 de Mayo de 2025, de https://tesis.ipn.mx/bitstream/handle/123456789/13236/%E2%80%9CDISE%C3%91O%20DE%20UN%20ROBOT%281%29.pdf?sequence=3&isAllowed=y

Rivera, D. (2016). *Control de un robot con ruedas de giro limitado [Trabajo de Grado, Universidad Politécnica de Madrid]*. Repositorio institucional, Madrid. Obtenido de https://oa.upm.es/43777/

Rodriguez, M., & Sandobalin, S. (Septiembre de 2013). *[Trabajo de titulacion de Grado, Escuela Politécnica Nacional]*. Repositorio institucional, Quito. Obtenido de Diceño y construccion de tres mini robots exploradores cooperativos.: https://bibdigital.epn.edu.ec/bitstream/15000/6692/1/CD-5089.pdf

Romero, D., & Unamuno, B. (2020). *Diseño e implementación del algoritmo de control para un Robot explorador con tecnología Pitsco Tetrix y myRio [Trabajo de titulacion de Grado, Universidad Politécnica Salesiana]*. Repositorio institucional, Guayaquil. Obtenido de diceño e implementacion del algoritmo de control para un robot explorador con tecnologia pitsco: https://dspace.ups.edu.ec/

Thido Electrónica. (2024). *Thido Electrónica*. Obtenido de Chasis De Carro Con 3 Ruedas: https://electronicathido.com/

TP-LINK. (2025). *Router Inalámbrico N a150Mbps*. Obtenido de Router Inalámbrico N a150Mbps: https://www.tp-link.com/latam/home-networking/wifi-router/tl-wr740n/

Urdaneta, M. (2012). *Diseño y desarrollo de un robot de inspeccion de tuberías [Tesis doctoral, Universidad Politécnica de Madrid]*. Repositorio institucional, Madrid. Obtenido de Diseño y desarrollo de un robot de inspección de tuberias: https://oa.upm.es/14776/1/MARIA_ALEJANDRA_URDANETA_LIMA.pdf

VEX V5. (2025). *kb.vex.com*. Obtenido de kb.vex.com: https://kb.vex.com/hc/es/articles/360035590992-Refuerzo-del-chasis-de-un-V5#:~:text=El%20chasis%20es%20el%20componente,como%20el%20marco%20del%20robot.

Yang, L., Fu, H., Liang, H., Yanbo, W., Guoqing, H., & Kegang, L. (2019). Detection of pipeline blockage using lab experiment and computational fluid dynamic simulation. *Journal of Petroleum Science and Engineering*, 183. Obtenido de https://www.sciencedirect.com/science/article/pii/S0920410519308423

ANEXOS

Algunas otras características técnicas que no fueron mencionadas son las que se muestran en la Tabla 5.

Tabla 5

Características generales del mini robot

VIDEO	
Resolución	VGA (640x480)
Audio	No
ILUMINACION	
Niveles de brillo	Bajo, Medio, Alto
Color	Luz Blanca
DESPLAZAMIENTO	
Tipo	Bidireccional
Velocidad con voltaje ideal	110 cm/min

Nota: Información básica sobre las características técnicas.

Código de programación en lenguaje C++ desarrollado en Arduino IDE.

```
#include "esp_camera.h"
#include <WiFi.h>
#include "esp_http_server.h"
#include "driver/ledc.h"

// Configuración de pines para AI-Thinker ESP32-CAM
#define CAMERA_MODEL_AI_THINKER
#define PWDN_GPIO_NUM     32
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM      0
#define SIOD_GPIO_NUM     26
#define SIOC_GPIO_NUM     27
#define Y9_GPIO_NUM       35
#define Y8_GPIO_NUM       34
#define Y7_GPIO_NUM       39
#define Y6_GPIO_NUM       36
#define Y5_GPIO_NUM       21
#define Y4_GPIO_NUM       19
#define Y3_GPIO_NUM       18
#define Y2_GPIO_NUM        5
#define VSYNC_GPIO_NUM    25
#define HREF_GPIO_NUM     23
#define PCLK_GPIO_NUM     22

// Pines de control
#define MOTOR_PIN1 12
#define MOTOR_PIN2 13
#define LED_PIN    15

// Configuración PWM para LED
#define LEDC_TIMER          LEDC_TIMER_0
#define LEDC_MODE           LEDC_LOW_SPEED_MODE
#define LEDC_CHANNEL        LEDC_CHANNEL_0
#define LEDC_DUTY_RES       LEDC_TIMER_8_BIT  // Resolución de 8 bits (0-
255)
#define LEDC_FREQ           5000
#define LEDC_INTR           LEDC_INTR_DISABLE

// Configuración de cámara
#define FRAME_SIZE FRAMESIZE_VGA
#define JPEG_QUALITY 10
#define FB_COUNT 1

// Configuración WiFi
const char* ssid = "TP-LINK_94ADCC";
const char* password = "";

// Servidores HTTP
httpd_handle_t stream_httpd = NULL;
httpd_handle_t cmd_httpd = NULL;

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println("\nIniciando sistema...");
```

```
// Configuración de pines del motor
pinMode(MOTOR_PIN1, OUTPUT);
pinMode(MOTOR_PIN2, OUTPUT);
digitalWrite(MOTOR_PIN1, LOW);
digitalWrite(MOTOR_PIN2, LOW);

// Configuración PWM para LED
ledc_timer_config_t ledc_timer = {
  .speed_mode       = LEDC_MODE,
  .duty_resolution  = LEDC_DUTY_RES,
  .timer_num        = LEDC_TIMER,
  .freq_hz          = LEDC_FREQ,
  .clk_cfg          = LEDC_AUTO_CLK
};
ledc_timer_config(&ledc_timer);

ledc_channel_config_t ledc_channel = {
  .gpio_num       = LED_PIN,
  .speed_mode     = LEDC_MODE,
  .channel        = LEDC_CHANNEL,
  .intr_type      = LEDC_INTR,
  .timer_sel      = LEDC_TIMER,
  .duty           = 85,  // Brillo bajo inicial (33%)
  .hpoint         = 0
};
ledc_channel_config(&ledc_channel);

// Configuración de la cámara
camera_config_t config;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
config.frame_size = FRAME_SIZE;
config.jpeg_quality = JPEG_QUALITY;
config.fb_count = FB_COUNT;
esp_err_t err = esp_camera_init(&config);

if (err != ESP_OK) {
  Serial.printf("Error al inicializar cámara: 0x%x", err);
  return;
}
```

```
// Conectar WiFi
WiFi.begin(ssid, password);
WiFi.setAutoReconnect(true);

Serial.print("Conectando a WiFi...");
int attempts = 0;
while (WiFi.status() != WL_CONNECTED && attempts < 20) {
  delay(500);
  Serial.print(".");
  attempts++;
}

if (WiFi.status() != WL_CONNECTED) {
  Serial.println("\nError al conectar WiFi");
  return;
}

Serial.println("\nWiFi conectado");
Serial.print("Dirección IP: ");
Serial.println(WiFi.localIP());

// Configurar servidor de COMANDOS (puerto 80)
httpd_config_t cmd_config = HTTPD_DEFAULT_CONFIG();
cmd_config.server_port = 80;
cmd_config.ctrl_port = 80;
cmd_config.max_open_sockets = 3;

if (httpd_start(&cmd_httpd, &cmd_config) == ESP_OK) {
  httpd_uri_t move_uri = {
    .uri = "/move",
    .method = HTTP_GET,
    .handler = move_handler,
    .user_ctx = NULL
  };
  httpd_uri_t led_uri = {
    .uri = "/led",
    .method = HTTP_GET,
    .handler = led_handler,
    .user_ctx = NULL
  };
  httpd_register_uri_handler(cmd_httpd, &move_uri);
  httpd_register_uri_handler(cmd_httpd, &led_uri);
  Serial.println("Servidor de comandos iniciado en puerto 80");
}

// Configurar servidor de STREAMING (puerto 81)
httpd_config_t stream_config = HTTPD_DEFAULT_CONFIG();
stream_config.server_port = 81;
stream_config.ctrl_port = 81;

if (httpd_start(&stream_httpd, &stream_config) == ESP_OK) {
  httpd_uri_t stream_uri = {
    .uri = "/stream",
    .method = HTTP_GET,
    .handler = stream_handler,
    .user_ctx = NULL
```

```
// Conectar WiFi
WiFi.begin(ssid, password);
WiFi.setAutoReconnect(true);

Serial.print("Conectando a WiFi...");
int attempts = 0;
while (WiFi.status() != WL_CONNECTED && attempts < 20) {
  delay(500);
  Serial.print(".");
  attempts++;
}

if (WiFi.status() != WL_CONNECTED) {
  Serial.println("\nError al conectar WiFi");
  return;
}

Serial.println("\nWiFi conectado");
Serial.print("Dirección IP: ");
Serial.println(WiFi.localIP());

// Configurar servidor de COMANDOS (puerto 80)
httpd_config_t cmd_config = HTTPD_DEFAULT_CONFIG();
cmd_config.server_port = 80;
cmd_config.ctrl_port = 80;
cmd_config.max_open_sockets = 3;

if (httpd_start(&cmd_httpd, &cmd_config) == ESP_OK) {
  httpd_uri_t move_uri = {
    .uri = "/move",
    .method = HTTP_GET,
    .handler = move_handler,
    .user_ctx = NULL
  };
  httpd_uri_t led_uri = {
    .uri = "/led",
    .method = HTTP_GET,
    .handler = led_handler,
    .user_ctx = NULL
  };
  httpd_register_uri_handler(cmd_httpd, &move_uri);
  httpd_register_uri_handler(cmd_httpd, &led_uri);
  Serial.println("Servidor de comandos iniciado en puerto 80");
}

// Configurar servidor de STREAMING (puerto 81)
httpd_config_t stream_config = HTTPD_DEFAULT_CONFIG();
stream_config.server_port = 81;
stream_config.ctrl_port = 81;

if (httpd_start(&stream_httpd, &stream_config) == ESP_OK) {
  httpd_uri_t stream_uri = {
    .uri = "/stream",
    .method = HTTP_GET,
    .handler = stream_handler,
    .user_ctx = NULL
```

```
      free(buf);
    }
  }
  httpd_resp_send(req, "OK", HTTPD_RESP_USE_STRLEN); return ESP_OK;
}

// Handler para streaming de video
static esp_err_t stream_handler(httpd_req_t *req) {
  camera_fb_t *fb = NULL;
  esp_err_t res = ESP_OK;
  size_t _jpg_buf_len = 0;
  uint8_t *_jpg_buf = NULL;
  char *part_buf[64];
  httpd_resp_set_type(req, "multipart/x-mixed-
replace;boundary=123456789000000000000987654321");
  while (true) {
    fb = esp_camera_fb_get();
    if (!fb) {
      res = ESP_FAIL;
      break;
    }
    if (fb->format != PIXFORMAT_JPEG) {
      bool jpeg_converted = frame2jpg(fb, JPEG_QUALITY, &_jpg_buf,
&_jpg_buf_len);
      if (!jpeg_converted) {
        res = ESP_FAIL;
        break;
      }
    } else {
      _jpg_buf_len = fb->len;
      _jpg_buf = fb->buf;
    }
    size_t hlen = snprintf((char *)part_buf, 64, "Content-Type:
image/jpeg\r\nContent-Length: %u\r\n\r\n", _jpg_buf_len);
    if (httpd_resp_send_chunk(req, (const char *)part_buf, hlen) != ESP_OK
||
        httpd_resp_send_chunk(req, (const char *)_jpg_buf, _jpg_buf_len) !=
ESP_OK ||
        httpd_resp_send_chunk(req, "\r\n--
123456789000000000000987654321\r\n", 38) != ESP_OK) {
      res = ESP_FAIL;
    }
    if (fb->format != PIXFORMAT_JPEG) {
      free(_jpg_buf);
    }
    esp_camera_fb_return(fb);
    if (res != ESP_OK) {
      break;
    }
  }
  return res;
}
```

Código de programación para interfaz HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Control de Robot Inspector</title>
    <style>
        :root {
            --primary-dark: #1a1a2e;
            --secondary-dark: #16213e;
            --button-color: #0f3460;
            --button-active: #00b4d8;
            --text-light: #f1f1f1;
            --stream-off: #ff0000;
            --stream-on: #00ff00;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #0d0d1a;
            color: var(--text-light);
            overflow: hidden;
        }

        .dashboard {
            display: grid;
            grid-template-columns: 70% 30%;
            height: 100vh;
        }

        .video-panel {
            background-color: #000;
            display: flex;
            flex-direction: column;
            position: relative;
        }

        .video-container {
            flex-grow: 1;
            display: flex;
            justify-content: center;
            align-items: center;
            background-color: #000;
            overflow: hidden;
        }

        .video-feed {
            max-width: 100%;
            max-height: 100%;
            object-fit: contain;
            transition: transform 0.3s ease;
            transform-origin: center;
        }

        .video-feed.rotated-portrait {
```

```
            max-width: 100vh;
            max-height: 100vw;
        }

        .video-feed.connecting {
            opacity: 0.5;
            animation: pulse 1.5s infinite;
        }

        .control-panel {
            background: linear-gradient(145deg, var(--primary-dark), var(--
secondary-dark));
            padding: 15px;
            display: flex;
            flex-direction: column;
            box-shadow: -5px 0 15px rgba(0,0,0,0.5);
            overflow-y: auto;
        }

        .controller {
            background-color: var(--secondary-dark);
            border-radius: 15px;
            padding: 15px;
            margin: 8px 0;
            box-shadow: inset 0 0 10px rgba(0,0,0,0.3);
            border: 2px solid rgba(255,255,255,0.05);
        }

        .section-title {
            color: var(--text-light);
            font-size: 1.1rem;
            margin: 10px 0;
            text-align: center;
            text-transform: uppercase;
            letter-spacing: 1px;
        }

        .btn {
            border: none;
            color: var(--text-light);
            font-weight: bold;
            cursor: pointer;
            transition: all 0.2s ease;
            border-radius: 10px;
            box-shadow: 0 3px 6px rgba(0,0,0,0.3);
            position: relative;
            overflow: hidden;
        }

        .btn:active {
            transform: translateY(2px);
            box-shadow: 0 2px 3px rgba(0,0,0,0.3);
        }

        .btn-toggle {
            background-color: var(--button-color);
            padding: 12px;
       margin-top: 8px;
       text-align: center;
```

```
        font-size: 0.85rem;
        color: rgba(255,255,255,0.7);
    }

    .brightness-control {
        margin: 12px 0;
        text-align: center;
    }

    .brightness-levels {
        display: flex;
        justify-content: space-between;
        margin-top: 8px;
    }

    .brightness-btn {
        flex: 1;
        margin: 0 4px;
        padding: 10px;
        background-color: var(--button-color);
        border-radius: 10px;
        transition: all 0.2s;
        font-size: 0.9rem;
        font-weight: bold;
    }

    .brightness-btn.active {
        background-color: var(--button-active);
    }

    .brightness-btn:first-child {
        margin-left: 0;
    }

    .brightness-btn:last-child {
        margin-right: 0;
    }

    @keyframes pulse {
        0% { opacity: 0.6; }
        50% { opacity: 1; }
        100% { opacity: 0.6; }
    }

    @media (max-width: 768px) {
        .dashboard {
            grid-template-columns: 1fr;
            grid-template-rows: 60vh 40vh;
        }

        .movement-btn {
            width: 100px;
            height: 100px;
            font-size: 2rem;
        }
    }
</style>
```

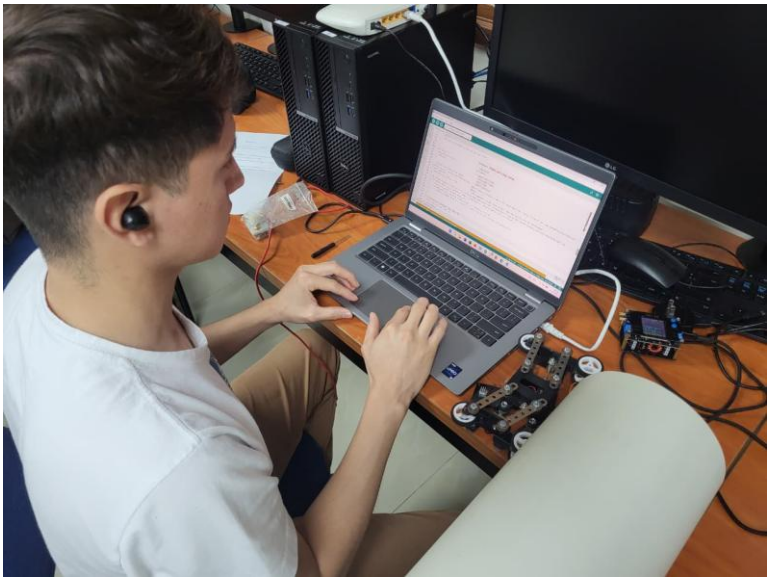
```
</head>
<body>
    <div class="dashboard">
        <div class="video-panel">
            <div class="video-container">
                <img id="videoFeed" class="video-feed" src="" alt="Video
Feed">
            </div>
        </div>

        <div class="control-panel">
            <div class="controller">
                <h2 class="section-title">Control de Cámara</h2>

                <button id="streamToggleBtn" class="btn btn-toggle">
                    <span class="status-indicator stream-off"></span>
                    INICIAR STREAM
                </button>

                <button id="rotateBtn" class="btn btn-
toggle">ROTAR</button>
                <div class="rotation-info" id="rotationInfo">Rotación
actual: 0°</div>

                <div class="brightness-control">
                    <h2 class="section-title">Brillo LED</h2>
                    <div class="brightness-levels">
                        <button id="brightLow" class="btn brightness-
btn">Bajo</button>
                        <button id="brightMedium" class="btn brightness-
btn">Medio</button>
                        <button id="brightHigh" class="btn brightness-
btn">Alto</button>
                    </div>
                </div>
            </div>

            <div class="controller">
                <h2 class="section-title">Control de Movimiento</h2>
                <div class="movement-controls">
                    <button id="forwardBtn" class="btn movement-
btn">↑</button>
                    <button id="backwardBtn" class="btn movement-
btn">↓</button>
                </div>
            </div>
        </div>
    </div>

    <script>
        // Configuración
        const esp32IP = window.location.hostname || "192.168.0.100";
        let isStreaming = false;
        let currentRotation = 0;
        let currentDirection = null;
        let lastCommandTime = 0;
        const COMMAND_THROTTLE = 50; // 50ms entre comandos
```

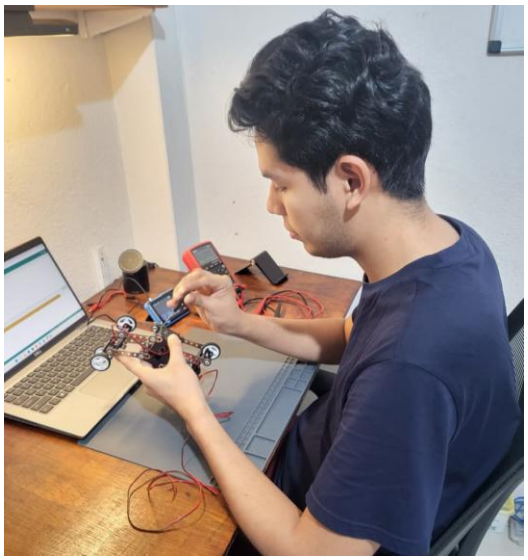
```
        // Elementos del DOM
        const streamToggleBtn = document.getElementById('streamToggleBtn');
        const videoFeed = document.getElementById('videoFeed');
        const forwardBtn = document.getElementById('forwardBtn');
        const backwardBtn = document.getElementById('backwardBtn');
        const rotateBtn = document.getElementById('rotateBtn');
        const rotationInfo = document.getElementById('rotationInfo');
        const brightLowBtn = document.getElementById('brightLow');
        const brightMediumBtn = document.getElementById('brightMedium');
        const brightHighBtn = document.getElementById('brightHigh');

        // Función para controlar el brillo del LED
        function setBrightness(level) {
            // Actualizar interfaz
            document.querySelectorAll('.brightness-btn').forEach(btn => {
                btn.classList.remove('active');
            });
            event.target.classList.add('active');

            // Enviar comando al ESP32
            const controller = new AbortController();
            const timeout = setTimeout(() => controller.abort(), 200);

            fetch(`http://${esp32IP}/led?brightness=${level}`, {
                cache: 'no-store',
                signal: controller.signal
            })
            .catch(e => console.log("Error ajustando brillo"))
            .finally(() => clearTimeout(timeout));
        }

        // Función optimizada para control del motor
        function controlMotor(direction) {
            const now = Date.now();
            if (direction !== currentDirection || now - lastCommandTime >=
COMMAND_THROTTLE) {
                currentDirection = direction;
                lastCommandTime = now;

                const controller = new AbortController();
                const timeout = setTimeout(() => controller.abort(), 200);

                fetch(`http://${esp32IP}/move?dir=${direction}&t=${now}`, {
                    cache: 'no-store',
                    signal: controller.signal
                })
                .catch(e => console.log("Motor command skipped"))
                .finally(() => clearTimeout(timeout));
            }
        }

        // Configurar controles táctiles y de ratón
        function setupMotorControl(button, direction) {
            const startMotor = () => {
                button.classList.add('active');
                controlMotor(direction);
```

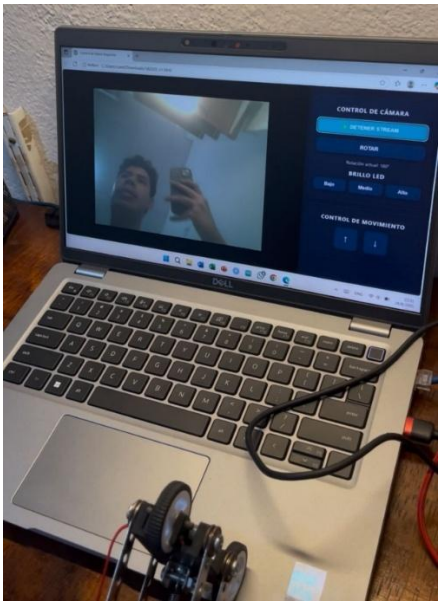
```
            streamToggleBtn.addEventListener('click', toggleStream);
            rotateBtn.addEventListener('click', rotateImage);

            // Configurar efecto de presión para el botón ROTAR
            rotateBtn.addEventListener('mousedown', () => {
                rotateBtn.classList.add('active');
            });

            rotateBtn.addEventListener('mouseup', () => {
                rotateBtn.classList.remove('active');
            });

            rotateBtn.addEventListener('mouseleave', () => {
                rotateBtn.classList.remove('active');
            });

            // Controles de brillo
            brightLowBtn.addEventListener('click', () =>
setBrightness('low'));
            brightMediumBtn.addEventListener('click', () =>
setBrightness('medium'));
            brightHighBtn.addEventListener('click', () =>
setBrightness('high'));

            // Eventos de teclado
            document.addEventListener('keydown', (e) => {
                if (e.repeat) return;

                if (e.key === 'ArrowUp') {
                    forwardBtn.classList.add('active');
                    controlMotor('forward');
                    e.preventDefault();
                }
                if (e.key === 'ArrowDown') {
                    backwardBtn.classList.add('active');
                    controlMotor('backward');
                    e.preventDefault();
                }
                if (e.key === 'r' || e.key === 'R') {
                    rotateBtn.classList.add('active');
                    rotateImage();
                    e.preventDefault();
                }
            });

            document.addEventListener('keyup', (e) => {
                if (e.key === 'ArrowUp') {
                    forwardBtn.classList.remove('active');
                    controlMotor('release');
                    e.preventDefault();
                }
                if (e.key === 'ArrowDown') {
                    backwardBtn.classList.remove('active');
                    controlMotor('release');
                    e.preventDefault();
                }
```

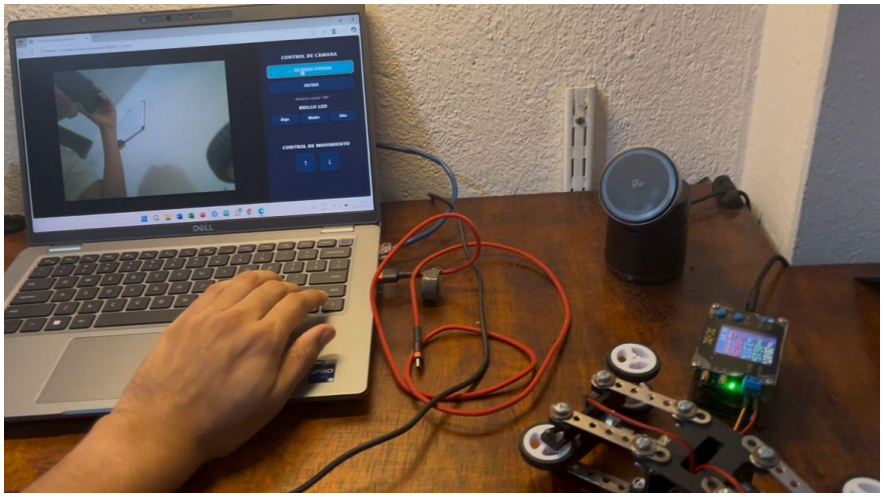
```
                if (e.key === 'r' || e.key === 'R') {
                    rotateBtn.classList.remove('active');
                    e.preventDefault();
                }
            });
        });

        // Manejo de errores del video feed
        videoFeed.onerror = function() {
            if(isStreaming) {
                videoFeed.classList.add('connecting');
                setTimeout(() => {
                    videoFeed.src =
`http://${esp32IP}:81/stream?t=${Date.now()}`;
                }, 2000);
            }
        };

        videoFeed.onload = function() {
            videoFeed.classList.remove('connecting');
            videoFeed.style.transform = `rotate(${currentRotation}deg)`;
        };
    </script>
</body>
</html>
```

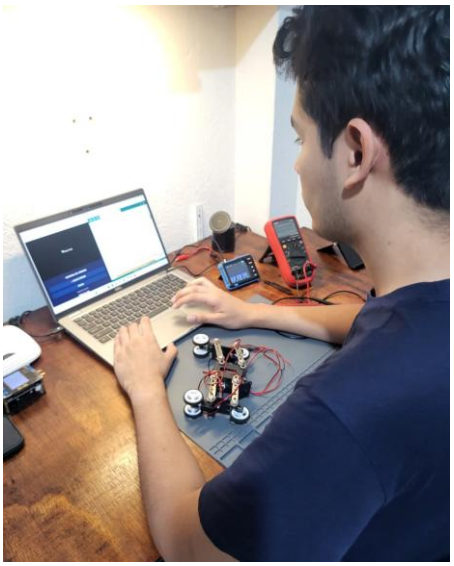
Fotografías capturadas durante el proceso de desarrollo y pruebas de validación del mini robot.

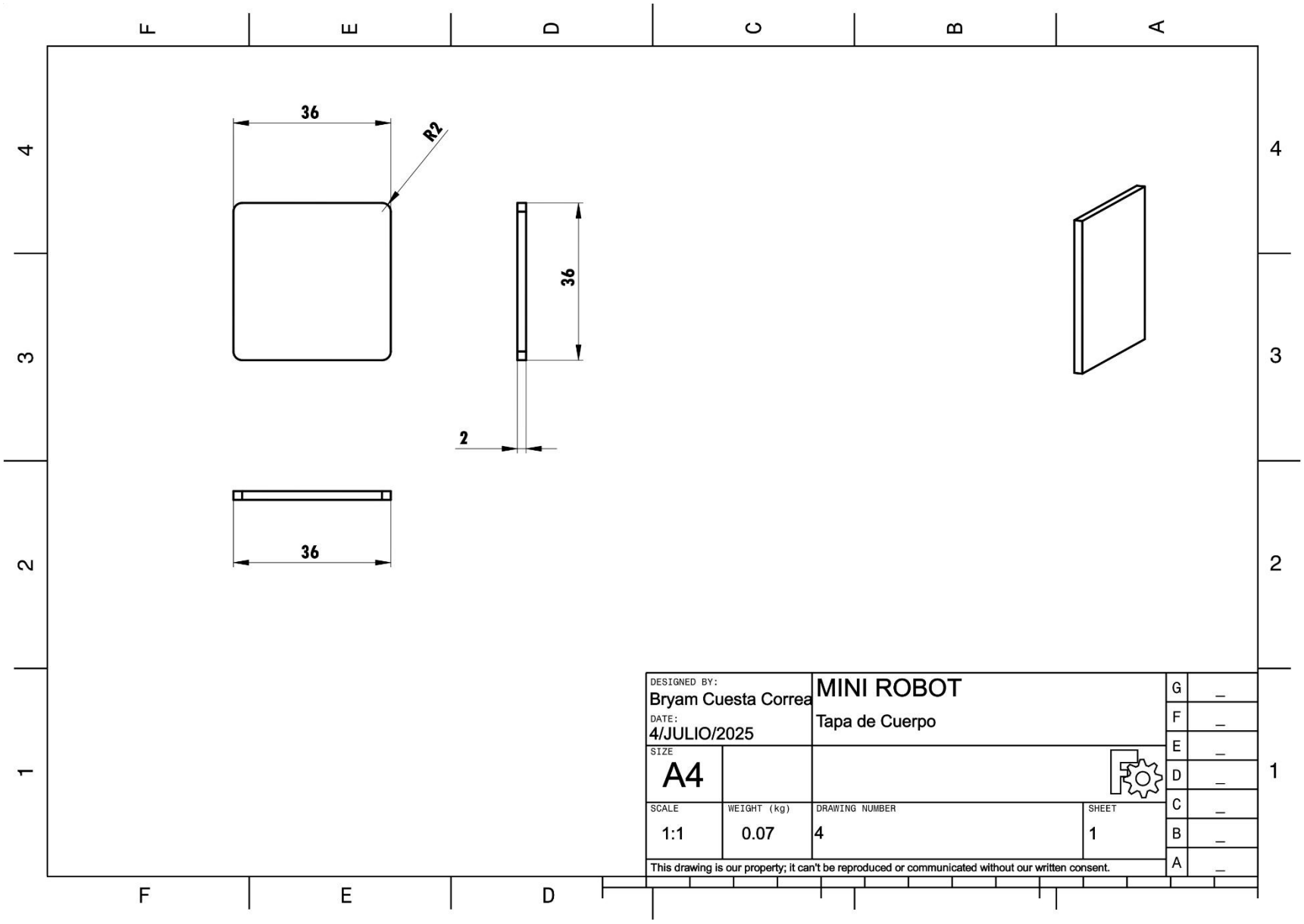




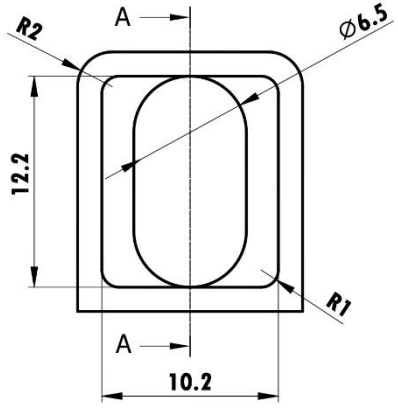

CONTROL DE CÁMARA
ROTAR
BRILLO LED
CONTROL DE MOVIMIENTO

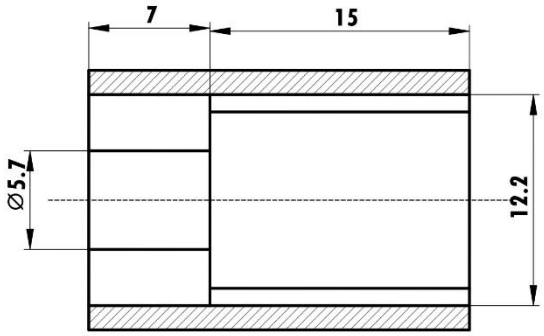

CONTROL DE CÁMARA
BRILLO LED
CONTROL DE MOVIMIENTO



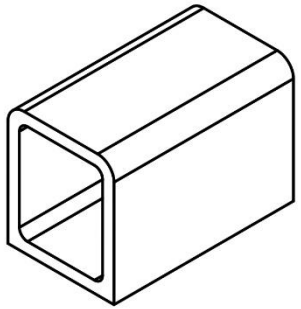


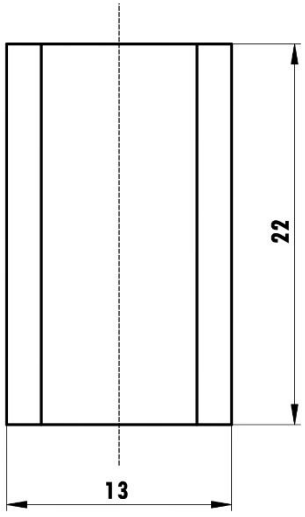
DESIGNED BY: Bryam Cuesta Correa		MINI ROBOT		G _
DATE: 4/JULIO/2025		Tapa de Cuerpo		F _
SIZE A4				E _
				D _
SCALE 1:1	WEIGHT (kg) 0.07	DRAWING NUMBER 4	SHEET 1	C _
				B _
This drawing is our property; it can't be reproduced or communicated without our written consent.				A _



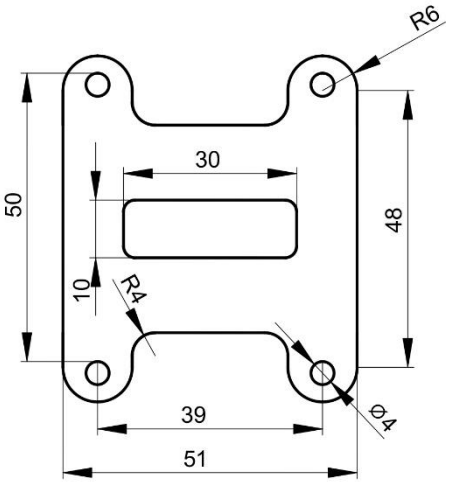


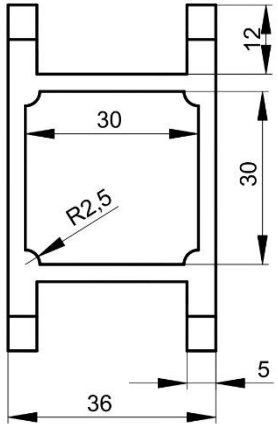
SECCIÓN A-A

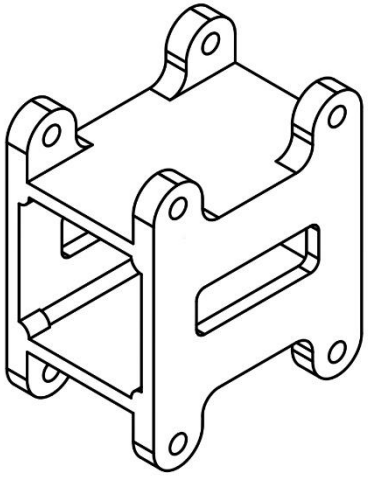


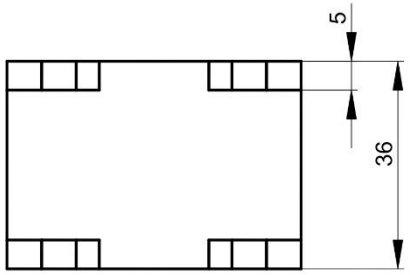


DESIGNED BY: Bryam Cuesta Correa DATE: 4/JULIO/2025		MINI ROBOT Base de Motor		
SIZE A4				
SCALE 3:1	WEIGHT (kg) 0.09	DRAWING NUMBER 3	SHEET 1	

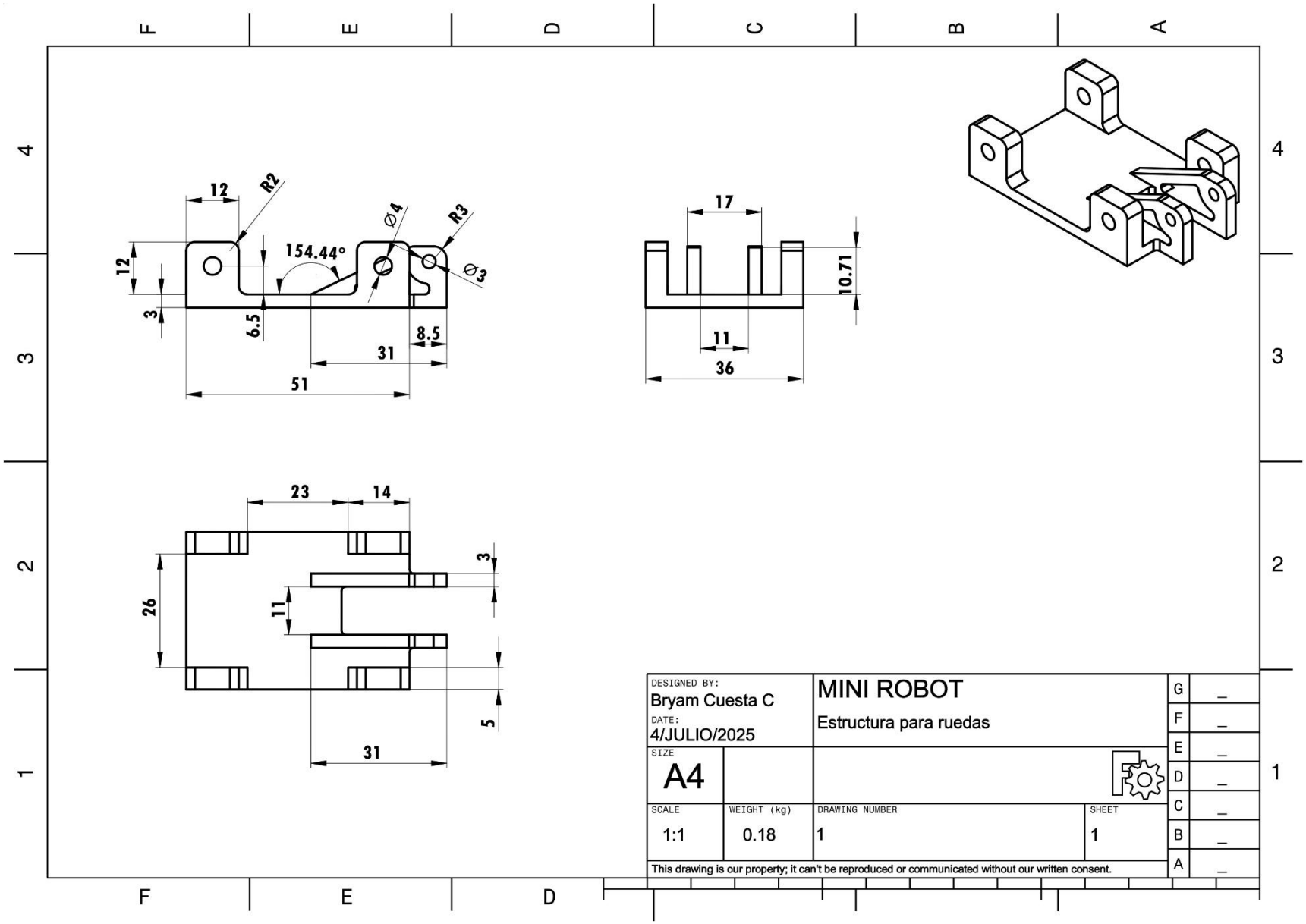








DESIGNED BY: CUESTA BRYAM DATE: 4/7/2025		MINI_ROBOT Cuerpo	
SIZE A4			
SCALE 1:1	WEIGHT (kg) 0.10	DRAWING NUMBER 1	SHEET 1


F
E
D
C
B
A
4
3
2
1
12
R2
154.44°
Ø4
R3
Ø3
12
3
6.5
8.5
31
51
17
10.71
11
36
23
14
3
26
11
5
31
DESIGNED BY:
Bryam Cuesta C
DATE:
4/JULIO/2025
MINI ROBOT
Estructura para ruedas
SIZE
A4
SCALE
1:1
WEIGHT (kg)
0.18
DRAWING NUMBER
1
SHEET
1
This drawing is our property; it can't be reproduced or communicated without our written consent.
G
F
E
D
C
B
A