Análisis de una Arquitectura de Procesamiento Distribuido (Jetson/Pico) Aplicada a un Robot RCJ Soccer Open

Technical Description Paper
Robocup Junior Soccer Open

Iker Cossío García
Fátima Chantal González Solís
Alejandro Salinas Salas

7 de mayo de 2025

PrepaTec Eugenio Garza Lagüera
Grupo Estudiantil AIZER

Contenido

1. Abstracto 5
2. Introducción 5
2.1 Contextos y motivaciones 5
2.2 Objetivos 6
2.3 Estructura del artículo 6
3. Trabajo relacionado 7
4. Descripción general 8
4.1 Propósito y enfoque 8
4.2 Presentación de subsistemas principales 8
4.3 Aspectos claves 9
5.1 Enfoque de diseño mecánico 10
5.2 Chasis 10
5.2.1 Sistema de locomoción 10
5.2.2 Batería 10
5.2.3 PCBs 10
5.3 Sistema de manejo del balón 11
5.3.1 Discusión de sistema de manejo del balón 11
5.3.2 Pateador 11
5.3.3 Zona de recepción 11
5.4 Sensores 12
5.4.1 Sensores de luz 12
5.4.2 Sensores de navegación 12
5.5 Sistema de Visión 12
5.5.1 Visión frontal 12
5.5.2 Visión omnidireccional 13
6. Electrónica 14
6.1 Unidades de procesamiento 14
6.2 Sistema de alimentación 14
6.3 Regulación de voltaje 15
6.4 Pateador 15
6.4.1 Función del relay 15
6.4.2 Activación del relay con juego de transistores 15
6.4.3 Carga del capacitor 15
6.5 Controlador de motores 16

6.6 Interfaz de sensores 17
6.6.1 Sensores de luz 17
6.6.2 Entradas analógicas (ADC) 17
6.7 Comunicación entre componentes 17
6.8 Placas de circuito impreso 17
6.8.1 Placa de control principal 17
6.8.2 Placas de sensores de luz 18
6.8.3 Placas de controlador de motores 18
6.8.4 Placa de pateador 18
6.9 Diagrama/Esquema eléctrico 19
6.9.1 Esquemático de la placa de control 19
6.9.2 Esquemático de la placa de motores 20
6.9.3 Esquemático de la placa de sensores de luz 20
6.9.4 Esquemático de la placa del pateador 21
7. Software para interacción con hardware 21
7.1 Arquitectura de software 21
7.2 Sensor de navegación (IMU) 23
7.3 Sensores de luz y entradas analógicas 23
7.4 Control de motores 24
7.4.1 Control de movimiento 25
7.4.2 Retroalimentación de movimiento 26
7.4.3 Freno 26
7.5 Control PID para dirección del robot 27
7.5.1 Objetivos y metas 27
7.5.2 Metodología 27
7.5.3 Implementación 27
7.5.4 Calibración 30
7.5.5 Resultados 34
7.6 Pateador 34
7.7 Arranque del robot 34
8. Software para visión 34
8.1 Arquitectura de software 35
8.2 Detección de objetos 35
8.3 Cámara frontal 37
8.3.1 Calibración de cámara frontal 37
8.3.2 Estimación de ángulo y distancia 39

8.4 Cámara omnidireccional 43
8.4.1 Calibración de cámara omnidireccional 43
8.4.2 Estimación para ángulo 46
9. Software para comunicación 47
9.1 ROS Melodic entre nodos 47
9.2 Comunicación Jetson - Raspberry Pi Pico 48
10. Software para implementación de estrategias 49
10.1 Descripción de estrategia 50
10.2 Uso de cámara frontal 50
10.3 Uso de cámara omnidireccional 50
10.4 Uso de pateador 51
10.5 Detección de líneas 51
11. Especificaciones técnicas 51
11.1 Dimensiones físicas 52
11.2 Limitaciones de hardware 52
11.2.1 Controladores de motores 52
11.2.2 Velocidad máxima 52
11.2.3 Rango y precisión de sensores 52
12. Discusión 53
13. Conclusión y trabajo futuro 54
14. Agradecimientos 54
15. Referencias 55
16. Apéndice 57
16.1 Esquemas electrónicos detallados 57
16.1.1 Placa de control 57
16.1.2 Placa de motores 59
16.1.3 Placa de sensores de luz 61
16.1.4 Placa del pateador 63
16.2 Arquitectura de software completa 64
16.3 Lista completa de materiales (Bill of materials - BOM) 64

1. Abstracto

La competición de Robocup Junior Soccer Open plantea el desafío de crear robots autónomos que compiten en un juego de fútbol a escala. El reto busca promover e impulsar a jóvenes innovadores a explorar la vanguardia de la robótica y de la IA a través de desafíos prácticos. En el reto de Soccer Open, se busca demostrar trabajo en equipo y precisión. [1]
Este artículo documenta exhaustivamente el proceso de diseño, desarrollo, programación y pruebas del desarrollo de los robots de AIZER para Soccer Open.
Mecánicamente, el robot está construido sobre un chasis compacto impreso en 3D y cortado en MDF, con un sistema de locomoción omnidireccional de 4 llantas y un sistema de control de la pelota que involucra un pateador y una zona de recepción.
Para la percepción del entorno se integran sensores de luz, IMU para la orientación y un sistema de dos cámaras gran angular con un espejo omnidireccional para obtener una visión 360 del campo.
La arquitectura electrónica utilizada es una Raspberry Pi Pico montada sobre una PCB personalizada y una Jetson Nano, gestionando la alimentación usando una pila LiPo de 3 celdas y 2 reguladores de voltaje a 5V y 3.3V.
El software de la Raspberry Pi Pico recibe las señales de los sensores como sensores de luz, encoders e IMU, además de recibir comandos de la Jetson Nano. La Jetson Nano por su parte analiza las imágenes de las dos cámaras, utiliza ROS Melodic como middleware para unir los nodos y usar estos datos para crear estrategias complejas de movimiento. La Jetson Nano y la Raspberry Pi Pico se comunican a través de USB para compartir datos clave que se necesitan para el funcionamiento de ambas computadoras.
Este trabajo culmina en un robot funcional y competitivo para RCJ Soccer Open, integrando soluciones innovadoras en sus subsistemas clave.

2. Introducción

Este documento presenta el diseño, desarrollo y pruebas detalladas de un robot autónomo para la competencia Robocup Junior (RCJ) Soccer Open. La introducción que sigue establece el contexto de esta competición, define los objetivos específicos de nuestro proyecto y describe la organización del presente artículo.

2.1 Contextos y motivaciones

El objetivo de la competencia es crear dos robots completamente autónomos que juegan en una cancha cerrada de mini fútbol, con una pelota de golf naranja. [2] Para el grupo estudiantil AIZER este reto es el más relevante, ya que se alinea con la visión y objetivo del grupo estudiantil, que es innovar y desarrollar soluciones en robótica autónoma.
Este proyecto es la actividad central del grupo estudiantil AIZER y busca preparar a los miembros para las carreras de Ingeniería y los equipos de robótica que ofrece el Tecnológico de Monterrey.
En los dos años que lleva el grupo estudiantil AIZER hemos buscado activamente soluciones para la visión por computadora con dos cámaras, la programación de sistemas embebidos con Raspberry Pi Pico, el uso de ROS Melodic en una Jetson Nano, el diseño de PCB personalizadas, el control de movimiento omnidireccional, entre otros.

Las habilidades que queremos lograr a través de esta competencia es el trabajo en equipo, el pensamiento computacional, el liderazgo, programación avanzada, diseño electrónico y diseño mecánico.
La motivación competitiva proviene de los dos años de experiencia previos en el TMR de Iker y Alejandro, y el año de experiencia previa de Fátima. El reto de enfrentarse a otros equipos y poner en práctica nuestras habilidades es muy llamativo para nosotros y es por eso que decidimos participar en esta competencia.
Nuestro objetivo para esta competencia es obtener un resultado positivo en el torneo, mostrar nuestras capacidades a través de los robots y mostrar la progresión que llevamos haciendo desde hace dos años que el grupo estudiantil comenzó.

2.2 Objetivos

En este año queremos lograr un sistema de locomoción con un buen control omnidireccional que nos permita implementar estrategias avanzadas.
Queremos además implementar un sistema de percepción visual que detecte y rastree objetos clave con rapidez y precisión.
Además, desarrollaremos un sistema de pateo de la pelota que nos permita experimentar con nuevas maneras de interactuar con la pelota no vistas en la categoría de Soccer Open en el Torneo Mexicano de Robótica (TMR).
Un alto grado de fiabilidad en todos los subsistemas es un objetivo ambicioso que queremos lograr en esta competencia.

2.3 Estructura del artículo

El resto del documento está organizado de la siguiente manera:

- La sección 3 presenta una revisión de trabajos previos con robots para la competencia Robocup Junior Soccer Open
- La sección 4 ofrece una descripción general de la arquitectura del sistema robótico propuesto
- La sección 5 detalla el diseño mecánico del robot, incluyendo el chasis, el sistema de locomoción omnidireccional, el manejo del balón y los sistemas de percepción física como la visión.
- La sección 6 describe la electrónica implementada, abarcando las unidades de procesamiento (Raspberry Pi Pico y Jetson Nano), el sistema de alimentación, los controladores, las placas del circuito impreso personalizadas y el pateador.
- Las secciones 7, 8, 9 y 10 se dedican al software, detallando la interacción con el hardware, procesamiento de visión, comunicación entre componentes e implementación de estrategias de juego respectivamente.
- La sección 11 resume las especificaciones técnicas clave alcanzadas por el robot
- La sección 12 discute los aspectos relevantes del diseño y posibles limitaciones
- La sección 13 presenta las conclusiones generales del trabajo y esboza posibles líneas de desarrollo futuro.

3. Trabajo relacionado

La competencia Robocup Junior (RCJ) Soccer Open desafía a los equipos a desarrollar robots autónomos capaces de jugar fútbol con un balón pasivo de color naranja, presentando retos significativos en percepción, control y estrategia. A lo largo de los años, han surgido diversas aproximaciones para abordar estos desafíos.

En cuanto a la locomoción, si bien la tracción diferencial es una opción robusta y común por simplicidad, muchos equipos de la liga de Open optan por sistemas omnidireccionales, para maximizar agilidad y espacio en el campo [3]. El año pasado AIZER optó por un sistema omnidireccional de 3 llantas [4]. Nuestro diseño actual, sin embargo, usa un sistema de cuatro ruedas omnidireccionales, buscando una mayor estabilidad, y potencialmente un control más preciso de movimiento lateral simultáneo. Además, nos proporcionará más fuerza en los motores y más peso lo que nos ayudará más a empujar a otros robots en situaciones donde los robots se juntan.

El principal reto en la liga Open es la detección fiable del balón naranja pasivo utilizando visión por computador. Un enfoque común es usar una única cámara frontal con un amplio campo de visión, mientras que otros implementan sistemas de visión omnidireccional con espejos para una cobertura de 360° [3]. Nuestro robot adopta un enfoque híbrido, utilizando un sistema de visión dual: una cámara frontal gran angular y una cámara adicional con espejo omnidireccional. Esta arquitectura busca combinar la visión detallada de la escena frontal con una percepción completa del entorno, permitiendo, en teoría, una mejor localización y una detección de objetos en el campo simultáneamente.

Para el manejo del balón, muchos equipos optan por implementar dribladores activos (como rodillos giratorios) para mantener el balón mientras se mueven [3], el primer año en el que Iker y Alejandro participaron en Robocup Junior Soccer Open (antes de AIZER), se implementó un sistema de driblador activo [5]. En las últimas dos ediciones de Robocup en las que AIZER ha participado nuestro diseño se desmarca de esta tendencia al optar por un diseño sin driblador activo, confiando en una zona de recepción y un pateador para diseñar nuestras estrategias. Esta decisión busca simplificar el diseño y permitir una transición más rápida entre la recepción y el disparo, además de una mayor velocidad de desarrollo.

Finalmente, en términos de arquitectura de control, es común encontrar robots basados en un único microprocesador de altas prestaciones o una single-board computer (SBC) que maneja todas las tareas [3]. Nuestra propuesta utiliza una arquitectura distribuida con una Jetson Nano dedicada al procesamiento de visión y tareas de alto nivel, y una Raspberry Pi Pico para el control de bajo nivel (Motores, IMU, Sensores de luz), comunicadas mediante ROS Melodic sobre USB. Este enfoque distribuido, es más complejo en términos de integración, pero nos permite un mayor nivel de paralelismo además de modularidad que nos facilita el desarrollo y depuración.

4. Descripción general

Esta sección presenta una visión general de la arquitectura y los principios del diseño del robot de AIZER, describiendo sus principales componentes funcionales y características distintivas antes de detallar cada subsistema en las secciones posteriores.

4.1 Propósito y enfoque

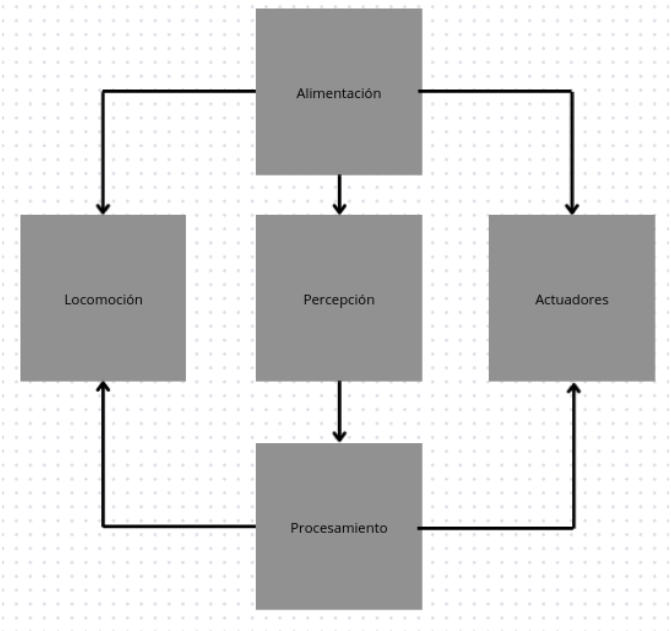
El propósito del robot es jugar un partido de fútbol a escala [2]. Utilizando principalmente la percepción visual y una arquitectura distribuida. El robot se enfoca en la modularidad, está construida utilizando hardware que nos permita dividir el desarrollo en secciones y con eso lograr un progreso acelerado. Nos enfocamos además en la robustez y fiabilidad, utilizando algoritmos que aproximen las instrucciones que les damos lo más cerca posible. Nos enfocamos en desarrollar un sistema de visión dual que nos permita desarrollar estrategias más avanzadas que somos capaces de implementar por la gran velocidad de desarrollo que podemos lograr a través de la modularización.
La arquitectura propuesta tiene como objetivo resolver el desafío de la navegación autónoma y la toma de decisiones en tiempo real con una pelota pasiva en un entorno real dinámico.

4.2 Presentación de subsistemas principales

Los principales subsistemas del robot se dividen en

- Alimentación (Véase sección 6): Provee de alimentación a los subsistemas del robot
- Locomoción (Véase sección 7.4 y 7.5): Maneja de manera precisa el control del robot lo que permite un movimiento limpio
- Percepción (Véase sección 7 y 8): Analiza de manera fiable las imágenes de las cámaras y detecciones de los sensores para crear una imagen del ambiente y tomar decisiones sólidas
- Actuadores (Véase sección 7.7): Interactúa con actuadores que agregan mayores posibilidades de estrategias
- Procesamiento (Véase sección 7, 8, 9 y 10): Procesa los datos del ambiente y responde con movimientos precisos.

El siguiente diagrama muestra la arquitectura general del sistema y la conexión entre los subsistemas principales.



4.3 Aspectos claves

- Locomoción omnidireccional de 4 ruedas: Seleccionado para maximizar la maniobrabilidad del robot y el control preciso del chasis.
- Sistema de visión dual: Combina una cámara frontal gran ángulo con una cámara colocada debajo de un espejo para obtener información detallada tanto del frente como contextual de 360°
- Procesamiento distribuido (Jetson/Pico/ROS): La arquitectura de control distribuida, que emplea una Jetson Nano para tareas de alta carga computacional como visión y una Raspberry Pi Pico para control en tiempo real. Comunicadas entre sí a través de ROS Melodic, es fundamental para el subsistema de procesamiento.
- Diseño sin driblador: Un diseño sin driblador nos da más espacio para instalar una visión frontal fiable y con un gran campo de visión, además que reduce la carga de trabajo y nos permite enfocarnos en el control del robot y desarrollo de estrategias.
- Integración de PCBs personalizadas: Nos permite llevar la electrónica al siguiente nivel que permite optimizar el espacio y asegurar conexiones fiables entre componentes.

5. Diseño mecánico

En esta sección nos enfocaremos en el diseño mecánico y las decisiones que se tomaron al momento de la creación del robot para optimizar su desempeño. En los apartados siguientes se detallarán las características técnicas y mecánicas que se implementaron en el robot, tanto para el aprovechamiento de espacio para la instalación de los componentes y a su vez una serie de estrategias mecánicas que facilitan y permiten su funcionamiento. Para la creación del diseño se utilizó la página web OnShape, una plataforma que se enfoca en la creación de modelos 3D [36]. Para las impresiones de diferentes piezas y soportes de la estructura del robot se utilizó una impresora 3D Ender 3 V3.

5.1 Enfoque de diseño mecánico

El diseño mecánico se centró en una integración de componentes efectiva, lo cual permitió un ahorro de espacio sin afectar el funcionamiento del robot. Se cuidaron diversos factores acerca de la estructura como la altura total, el peso y la distribución proporcional del peso del robot, manteniendo su equilibrio en todo momento. Siempre cuidando y garantizando que el robot cumpla fuertemente con las medidas establecidas en el reglamento internacional de RoboCup Soccer Junior, específicamente en el apartado 6.2.A. Otro elemento clave que se tuvo en cuenta en la creación del robot fue la fácil capacidad de acceso a la estructura del robot permitiendo así ajustes rápidos en caso de alguna corrección o ajuste.

5.2 Chasis

El diseño del chasis de este año se centró en mejorar y potenciar el posicionamiento de los motores, ofreciendo mejor estabilidad y mejorando el movimiento y fuerza del robot al cambiar nuestro sistema de movimiento omnidireccional de 3 motores a 4 motores en el nuevo diseño del robot, como se mencionó anteriormente en el apartado número 3 de Trabajo Relacionado. El diseño actual cuenta con 4 motores con ruedas omnidireccionales posicionadas en el perímetro de nuestro chasis, lo cual nos permite mayores capacidades de movimiento a comparación del diseño pasado. Nuestro chasis también fue desarrollado con la idea de la integración de las placas de los motores a un cercano alcance del motor para una mejor organización del cableado y para permitir un fácil acceso a este.

5.2.1 Sistema de locomoción

Nuestro sistema de locomoción está conformado por 4 motores conectados a llantas omnidireccionales distribuidos alrededor del chasis cada 90 grados. Gracias a esto se cuenta con un movimiento más preciso y muestra una estabilidad aún mayor. Por el cambio de 3 motores a 4 motores, véase apartado 3 de Trabajo relacionado para más información, ahora contamos con una distribución de la carga del robot más equitativa lo cual nos permite mayor estabilidad al momento de que se realicen movimientos.

5.2.2 Batería

La estructura de la posición de la batería está diseñada específicamente para ser de fácil alcance, posicionada en una de las bases superiores lo cual facilita su extracción e integración de manera rápida y sin obstáculos o riegos de afectar alguno de los componentes externos e internos al constante movimiento de la batería.

5.2.3 PCBs

Diseño estándar, que ambos robots tengan ambos lo mismo, porque el posicionamiento de la PCB, que están protegidas por una pared. Las PCB (palabra)
Las PCBs (Placa de Circuito Impreso) fueron diseñadas específicamente para su integración a la estructura interna del robot. Se optó por PCBs con un diseño estándar e irregular para acomodarse en el robot. Cada placa protegida por una pared para garantizar su seguridad sin importar movimientos bruscos que pueda realizar el robot. Para el diseño de las placas se

buscó seguir medidas específicas para garantizar el ahorro de espacio y la implementación mínima de cables para mantener organización y de esa manera tener fácil acceso a cualquier componente interno en caso de ajustes.

5.3 Sistema de manejo del balón

El sistema de manejo del balón en la estructura del robot gira en base al pateador, el cual cuenta con un área de recibimiento, todo el sistema ofrece mayor precisión y fuerza al momento de ejecutar una acción en el balón mientras a su vez permite el ahorramiento de espacio para más elementos a comparación de un driblador.

5.3.1 Discusión de sistema de manejo del balón

Para la construcción del robot se optó por la utilización e instalación de un pateador a diferencia del primer año que se optó por un driblador como se menciona brevemente en el apartado 3 de Trabajos Relacionados. Se menciona el uso de un driblador activo, sin embargo, se tomó la decisión de utilizar pateadores solenoides, los dispositivos solenoides son perfectos para la creación de movimiento linear de empuje [21], se utilizaron modelos JF-0826 y JF-0826B con una zona de recepción para cada zona. Al usar un pateador en lugar de un driblador permite el ahorro de espacio en la estructura del robot, a su vez aporta fuerza y precisión en la dirección en la que se patea el balón. Al ser un pateador en lugar de un driblador facilita la creación de movimientos y estrategias del robot para que sean más efectivos y reduce la velocidad en la que se ejecuta la acción, dándonos una ventaja contra el driblador.

5.3.2 Pateador

Los modelos que se utilizaron para el pateador son los modelos JF-0826 y el JF-0826B los cuales cuentan con un voltaje de operación de 12V, los cuales al ser solenoides lineales ofrecen un golpe fuerte lineal de 20 Newtons [21], el cual permite impulsar el balón a una velocidad suficiente a través de la cancha. Al tener un diseño compacto permite que su integración a la estructura del robot sea sencilla. Gracias a su bajo costo y su rápida respuesta de acción lo vuelven ideal para su uso en el robot como pateador. A su vez se destaca que no requiere un mantenimiento complejo lo cual al momento de competencia lo vuelve conveniente y facilita su uso.

5.3.3 Zona de recepción

La zona de recepción está diseñada específicamente para el pateador y para facilitar su uso, dejando un acceso despejado hacia el pateador en donde reposa el balón y la cámara lo detecta para activar el pateador. La zona de recepción no solo facilita el uso del pateador y su posicionamiento si no también ayuda a guiar el balón directamente al pateador, aunque el robot se encuentre en movimiento. Conformada por cortes diagonales en la estructura del robot que se abren hacia afuera, guiando cualquier objeto directamente hacia el pateador.

5.4 Sensores

Para que el robot pueda ubicarse y desplazarse de manera precisa dentro de la cancha se utilizaron múltiples encoders, los cuales permiten estimar en tiempo real la posición del robot gracias a su habilidad de convertir el movimiento generado por el giro de las llantas en una señal eléctrica que luego se utiliza para determinar su posición en la cancha [28]. Además de también integrar un sistema de navegación conformado por una BNO055 que obtiene datos de la dirección del robot de manera precisa. Gracias a la combinación de los datos obtenidos por los encoders y nuestro sistema de navegación (IMU) nos garantiza una mayor precisión y estabilidad en los datos de la posición y dirección del robot en la cancha, los cuales facilitan su trayectoria autónoma.

5.4.1 Sensores de luz

Para la instalación de los sensores de luz se optó por la implementación de fotorresistencias combinadas con leds rojos. Esta combinación permite que la luz emitida por los leds al tener contacto con una línea reflectiva, tal y como lo es la línea blanca, genere que la luz rebote hacia la fotorresistencia, activándola [29]. Gracias a esta técnica el robot es capaz de identificar los límites de la cancha y ajustar su trayectoria, manteniéndose en el área de juego.

5.4.2 Sensores de navegación

Para el sensor de navegación se utilizó el sensor BNO055, el cual es un componente que integra un giroscopio, acelerómetro y un magnetómetro en un solo componente, la cual es capaz de proporcionar datos de manera precisa y clara acerca de la dirección y posición del robot en tiempo real, lo cual resulta esencial para la coordinación de movimientos del robot de manera precisa [27]. La BNO055 utiliza un protocolo de comunicación I2C, como se puede observar en el apartado 7.2 del sistema de navegación. Su tamaño compacto permite una fácil integración a la estructura del robot, lo cual lo vuelve en una manera de navegación eficiente. La BNO055 al estar sujetada firmemente a la estructura interna del robot hace que se garantice una estabilidad en las lecturas aún mayor, al reducir el riesgo de errores en la lectura causado por vibraciones o movimientos bruscos.

5.5 Sistema de Visión

El sistema de visión del robot está conformado por dos cámaras Arducam IMX 708 gran angular en cada robot, una que nos permiten una visión omnidireccional y a su vez una cámara enfocada en la zona de recepción del pateador. Anteriormente el robot consistía de una cámara Arducam IMX 708 gran angular para la visión omnidireccional y una Raspberry Pi Camera v3 para el enfoque en la visión de la zona de recepción, sin embargo, con la Raspberry Pi Camera v3 se notó la falta de campo de visión a comparación de la Arducam IMX 708 gran angular [24], por lo cual se optó por un cambio de implementar que todas las cámaras sean Arducam IMX 708 gran angular en ambos robots, lo cual nos ofrece un campo de visión bastante amplio.

5.5.1 Visión frontal

La visión frontal está conformada por una cámara Arducam IMX 708 gran angular que está apuntando hacia la zona de recepción del pateador y se encuentra posicionada en la

estructura que se encuentra en la parte superior del pateador. La visión frontal aparte de que permite tener un campo de visión más amplio también le permite al robot saber la posición de la pelota en el área de recepción para poder activar el pateador. En la plantación de los elementos del robot se decidió realizar el cambio de la cámara frontal de una cámara Raspberry Pi camera V3 a una Arducam IMX 708 gran angular, como se mencionó anteriormente en el apartado 5.5 de Sistema de Visión, se decidió el cambio de la cámara al decidir ampliar el cambio de visión frontal para un mejor manejo y movimiento del robot en la zona de recepción.

5.5.2 Visión omnidireccional

La visión omnidireccional está compuesta por una Arducam IMX 708 gran angular, que está apuntada hacia un espejo cónico de aluminio en la parte de arriba, el cual está sujeto por un tubo de acrílico transparente cortado a medida.
El proceso para la elaboración del espejo se creó utilizando primeramente un modelo personalizado tomando en cuenta la altura del robot y la posición de la cámara para abarcar la mayor cantidad de espacio posible.
Una vez teniendo un modelo del espejo omnidireccional, se pidieron las piezas a JLCPCB.
Ya teniendo control de la pieza de aluminio, se le tuvo que dar una superficie de espejo, para eso probamos dos distintos métodos.

- Cromado: El cromado en aluminio es un proceso en el cual se recubre una pieza con una fina capa de cromo, lo que le da un acabado similar a un espejo [31]. El proceso completo requiere de una gran cantidad de pasos y maquinaria especializada, es por eso que para la creación de un espejo omnidireccional utilizando el método de cromado, se prefirió contactar a una empresa especializada en cromado, Cromanorte.
- Pulido: El aluminio es un material propenso a rayaduras, oxidación, manchas e imperfecciones. El proceso de pulido busca eliminar estas imperfecciones y darle un acabado espejo [32]. El proceso para pulir es más amigable que el cromado, e incluso se puede realizar de manera “casera”, sin embargo, por seguridad se decidió contactar a una persona con el conocimiento necesario para pulir el espejo.

Una vez terminado el proceso para ambos espejos, se realizó una comparación para encontrar cual espejo funciona mejor para nuestro objetivo.

Tipo de espejo	Costo	Tiempo	Precisión	Color	Resistencia
Cromado	$350 MXN	1 semana	+1.5mm	Colores brillantes y menos deformación de imagen	Resistente a rayones
Pulido	$250 MXN	2 días	Sin pérdida de precisión	Colores más opacos y con más deformación	No resistente a rayones

El cromado regresó un mejor resultado en términos de color, el acabado de espejo era más cercano a la realidad, además que la capa de cromo era más resistente a los rayones que el espejo pulido. El espejo pulido fue un proceso mucho más rápido y barato, sin embargo, no

tenía una resistencia a los rayones, después de unas semanas de uso se empezaron a formar rayones bastante visibles, y que si bien para el robot son imperceptibles, el cromado no tiene este defecto.

El cromado tuvo una pérdida de precisión, si bien es pequeña, puede ser notada si se presta la suficiente atención, aunque para nuestro robot no significa un problema, ya que el control de nuestro robot tiene menos precisión que la que se pierde durante el proceso.

6. Electrónica

La electrónica del robot recae principalmente en las placas de circuito impreso (PCB) las cuales son: la placa de control principal, la placa de motores, la placa de sensores de luz y la placa del pateador. (Para descripción específica de estas, véase la sección 6.8). Estas placas fueron diseñadas usando el software de Fusion 360, de Autodesk.

6.1 Unidades de procesamiento

Las unidades de procesamiento que usamos son la Raspberry Pi Pico y la Jetson Nano. La Raspberry Pi Pico es en el modelo estándar, al cual se le tuvieron que soldar headers macho ya que no venían incluidos, a diferencia de la pico H. [7].
Decidimos usar la Raspberry Pi Pico en vez de Arduino Uno R3 por su capacidad de procesamiento: Dual-core ARM Cortex-M0+ a 133 MHz en la Raspberry Pi Pico, contra el AVR ATmega328P a 16 MHz del Arduino Uno R3 y 264 KB de RAM, 2 MB de Flash, contra 2 KB de RAM, 32 KB de Flash del Arduino Uno R3 [7], [8]. Esto permite tener programas más complejos y un mejor manejo de datos.
Otro punto fue que también es más barata que el Arduino Uno R3, por lo cual se han podido comprar repuestos con mayor facilidad.
Adicionalmente, la Raspberry Pi Pico tiene más Pines de Entrada/Salida de Uso General (GPIOs) (26 contra 14) de los cuales todos son utilizados y con menos se necesitan dos unidades en vez de solo una. También se consideró que todos los GPIOs de la Raspberry Pi Pico cuentan con Modulación por Ancho de Pulso en 8 canales independientes, pero pudiendo usar cualquier GPIO, con lo cual se puede simular señales analogicas usando pines digitales y con este principio se manejan los datos de los encoders de motores, de los sensores de luz y de la brújula, mientras que en Arduino solo 6 GPIOs cuentan con función de PWM y no se puede cambiar cuáles, y también solo cuenta con 6 entradas análogas, lo cual suma 12 pines que pueden usarse para manejar un rango de datos más que solo un estado de apagado o encendido, contra los 26 distintos de la Raspberry Pi Pico. [7], [8].
La Jetson Nano se encarga principalmente de la visión del robot, para el algoritmo que es usado para esta puesto a que es bastante demandante, y la raspberry pi pico no podría con esa carga. Contamos con el modelo Jetson Nano Developer Kit B01, el cual cuenta con 4 GB de RAM 64 bits LPDDR4 a 25.6 GB/s, una GPU NVIDIA Maxwell con 128 núcleos CUDA de NVIDIA, una CPU ARM 4 núcleos Cortex-A57 a 1.43Ghz y rendimiento de 0.5 teraflops [34].

6.2 Sistema de alimentación

El robot tiene un sistema de alimentación mediante una batería LiPo de 11.1V con 2200 mAh y 3 celdas, de marca Supulse con 50C (potencia de carga-descarga). Se usa una batería LiPo

en vez de alternativas como baterías Li-ion, LiFePO4 o baterías AA NiMH, ya que las baterías LiPo tienen más efectividad para este tipo de aplicaciones en comparación a los otros tipo de baterías, debiéndose a varios factores como su peso, balance, potencia y costo, estas baterías mantienen mucha carga en un espacio compacto, mientras que las otras tienen tamaños más pesados y menos capacidad de carga y rendimiento, pero cabe mencionar que otros tipos de baterías como LiFePO4 son más estables y durables gracias a su composición química de fosfato de hierro [9], por lo cual son buenas para otro tipo de aplicaciones, pero no para nuestras necesidades específicas.

6.3 Regulación de voltaje

Para regular el voltaje se utilizan 2 módulos (nombre) que son capaces de recibir desde X Volts para reducirlo hasta una cantidad deseada. Usando los 12 Voltios de la pila, usamos dos de estos módulos, con uno se regula el voltaje de 12 a 3.3 voltios para darle voltaje a la Raspberry Pi Pico, sensores de luz, convertidor de señales análogas a digital, brújula, botones, encoders de los motores y poder lógico de los controladores de motores. Con el otro se regula hasta 5 voltios, los cuales se utilizan para darle poder a la Jetson Nano y también para el amplificador operacional del circuito externo del pateador.

6.4 Pateador

6.4.1 Función del relay

El relay SRD-05VDC-SL-C tiene la función de ser un switch electromecánico, para poder activar el pateador cuando sea requerido mediante una señal 3.3 voltios de un GPIO, lo cual provoca que el resorte electromagnético que tiene incluido haga que el switch interno del relay alterne entre estado de NA y NC (Normalmente Abierto y Normalmente Cerrado), en los cuales cuando está en estado NC 38 voltios estarán cargando un capacitor de 10000 micro faradios (µF), y cuando se reciba la señal de 3.3 voltios del GPIO de la Raspberry Pi Pico el switch cambia a estado NA y la carga almacenada del capacitor se libera hacia el pateador, el cual es un solenoide JF-0826, así logrando su función de patear la pelota efectivamente.

6.4.2 Activación del relay con juego de transistores

Debido a que la Raspberry Pi Pico no puede dar los suficientes mAh para la activación del resorte electromagnético del relay mediante un GPIO solo, se tuvo que buscar una solución la cual fue hacer un juego de transistores para simular un amplificador operacional (OPAMP), con el cual teniendo 5 voltios obtenidos desde un regulador de voltaje, se logró hacer el efecto de controlar el relay con el GPIO mediante esta ayuda extra, ya que ahora cuando el GPIO manda la señal de 3.3 voltios, esta pasa en el transistor NPN 337 y este está conectado a el transistor PNP 327, del cual sale con un diodo flyback 1N4007 la señal que se manda al relay, activándolo exitosamente puesto que ya tiene el amperaje suficiente gracias a este sistema.

6.4.3 Carga del capacitor

El capacitor tarda aproximadamente 8.219 segundos en cargar hasta 20 voltios, lo cual es suficientemente para tener una buena potencia para liberar al pateador (suficiente para llegar de un extremo de la cancha al otro) y tarda aproximadamente 50.657 segundos en llegar a

cargarse completo (99%) en 37.62 voltios aproximadamente. Los cálculos se muestran a continuación:

$$V = 38V$$
$$C = 0.01F$$
$$R = 1100\Omega$$

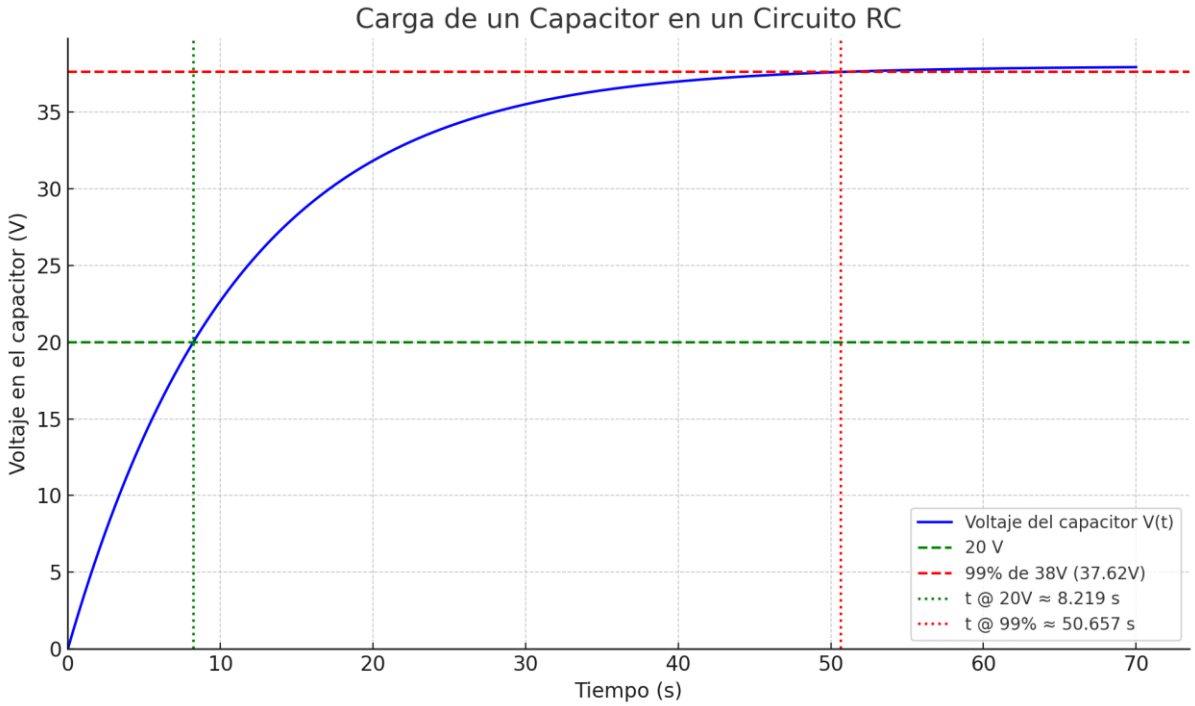
$$V(t) = v(1 - e^{-t/RC})$$
$$t = -RC \times ln(1 - \frac{v(t)}{v})$$

$v(t) = 20V$:

$$t = -1100 \times 0.01 \times ln(1 - \frac{20}{38})$$
$$t = 8.219s$$

$$v(t) = 37.62V$$
$$t = -1100 \times 0.01 \times ln(1 - \frac{37.62}{38})$$
$$t = 50.657s$$

La diferencia tan grande del tiempo de carga es por el cómo funciona la carga de un capacitor, y por el hecho de que esto mismo pasa de forma exponencial, entonces entre más acercado se esté a finalizar, más tiempo tomará, como se muestra en la siguiente gráfica:



6.5 Controlador de motores

Se utilizan los controladores de motor DRV8256E, los cuales son una muy buena alternativa a otros controladores de motores DC populares como los puentes H L298N. Los DRV8256E tienen ventajas como emplear MOSFET en vez de transistores bipolares NPN (BJT) como los

que usan los L298N [18] y esto reduce la caída de voltaje interna y provee mejor eficiencia energética, ya que los transistores BJT utilizan más energía por ser controlados mediante corriente contra el MOSFET que es controlado con voltaje [19]. Otras ventajas también incluyen: protección contra sobre-corriente, sobre-temperatura y subcorriente también [20]. Otro punto es que el control de estos es más eficaz, ya que para el puente H se necesitan dos pines para controlar dirección y otros dos para controlar velocidad, necesitando 4 en total por motor, el DRV8256E solo necesita dos por motor. Y como última y principal mejora, el espacio que estos utilizan es mucho menor (0.6″ × 0.6″) [20], y se pueden integrar fácilmente en una PCB.

6.6 Interfaz de sensores

6.6.1 Sensores de luz

Los sensores de luz que se utilizan son fotorresistencias (LDRs) de los cuales se usan 8 por robot, acomodados en dos PCB. Funciona detectando cambios de luz, cuando hay poca luz la resistencia sube y cuando la luz sube, la resistencia baja, y esto se debe a el material del que está hecho usualmente, sulfuro de cadmio, un semiconductor con el cual se genera el efecto fotoeléctrico. El componente cuenta con dos electrodos entre los cuales está el material sensible a la luz. Cuando detectan luz, los electrones se activan, aumentando la conductividad y disminuyendo resistencia, y cuando no hay luz, la resistencia sube nuevamente [22].

6.6.2 Entradas analógicas (ADC)

Debido a que la Raspberry Pi Pico no contaba con suficientes GPIOs tuvimos que integrar un convertidor analógico a digital (MCP3008) para poder usar 8 fotorresistencias por robot, el cual es un convertidor analógico a digital que permite ampliar hasta ocho canales analógicos mediante comunicación UART, utilizando únicamente cuatro pines digitales del microcontrolador (SCK, RX, TX y CSN). Este convertidor ADC ofrece una resolución de 10 bits [23], adecuada para capturar los cambios sutiles en la resistencia de las LDRs causados por variaciones en la intensidad lumínica. Esta integración permite una lectura eficiente y precisa de todos los sensores de luz, optimizando el uso de los recursos de hardware disponibles en el sistema.

6.7 Comunicación entre componentes

La Raspberry Pi Pico y la Jetson nano se comunican mediante puerto micro usb para compartir información relevante del manejo de los motores respecto a la información que la Jetson procesa de la cámara omnidireccional y la cámara frontal.

6.8 Placas de circuito impreso

6.8.1 Placa de control principal

En esta placa se incluye: 2 headers hembra 20x1 (para insertar la raspberry pi pico), 6 headers macho 8x1, 1 header 4x1 macho, 1 header 6x1 hembra, 1 header 4x1 hembra (ambos para sostener la BNO 055), 6 borneras, 2 botones, una resistencia 220, 1 convertidor

análogo a digital, 2 capacitores 22uF, 2 capacitores 10uF, 2 transistores LM317, 2 resistencias de 0 ohm, 2 resistencias de 560 ohm y 2 resistencias de 330 ohm.
El tamaño de las rutas es de 0.508 mm para 3.3 voltios, 1.651 mm para 12 voltios y 0.254 mm para señales digitales.
Esta placa se encarga de controlar las señales que llegan de los motores y sensores de luz, mediante la Raspberry Pi Pico, y también de activar el relay para el pateador cuando sea requerido.

6.8.2 Placas de sensores de luz

Cada placa de sensores de luz incluye: 4 fotorresistencias, 3 LEDs rojos, 4 resistencias de 1K Ω, 3 resistencias de 220 Ω y una tira de 6x1 headers macho. (al final las resistencias de 220 Ω fueron reemplazadas por resistencias de 1K Ω para reducir la intensidad de luz de los LEDs) y se usan dos placas por robot.
El tamaño de las rutas es de 0.508 mm para la línea de 3.3 voltios y 0.254 mm para las rutas de señales analógicas.
Esta placa tiene la función de mandar señales analógicas a la Raspberry Pi Pico con las 4 fotorresistencias por placa que incluye para luego pasar por el convertidor análogo a digital antes de llegar a la Raspberry Pi Pico, y después de pasar por este convertidor, detectar las líneas blancas de la portería, y en caso de que estas se detecten detener y hacer el robot para atrás.

6.8.3 Placas de controlador de motores

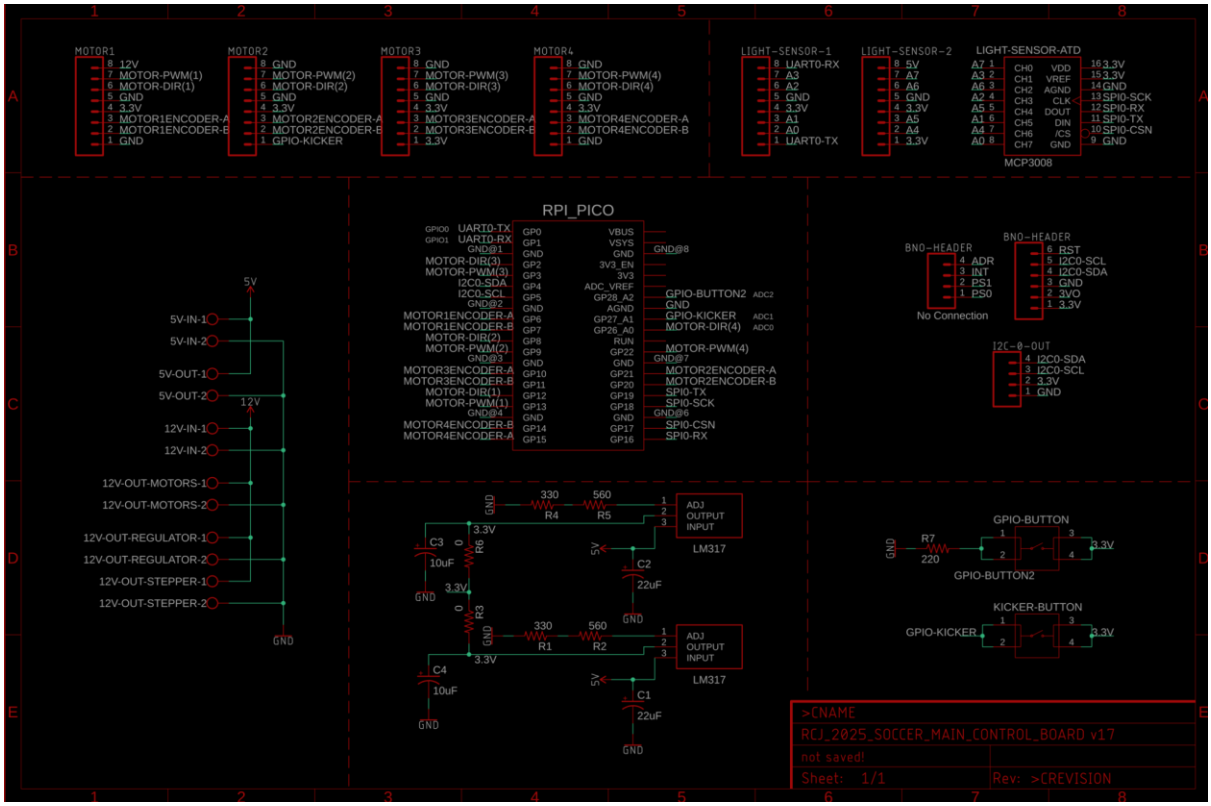
La PCB de motores cuenta con 4 headers 6x1 macho, 4 headers 6x1 hembra, 2 borneras, 2 LEDs blancos, 2 resistencias 220 ohm y 2 DRV 8256E por placa. (se insertan en los headers hembra estos).
El tamaño de las rutas es de 0.508 mm para la línea de 3.3 voltios, 1.651 mm para la línea de 12 voltios, y las líneas de poder para los motores y 0.254 para señales digitales.
Esta placa se encarga de manejar las señales de los motores DC 12 voltios de polulu, mediante un sistema de encoders, para controlar su velocidad y dirección, estos datos vienen de la Raspberry Pi Pico para controlar estas variables, y que los controladores de motores DRV 826E envíen estos datos para manejar los motores de la forma deseada.

6.8.4 Placa de pateador

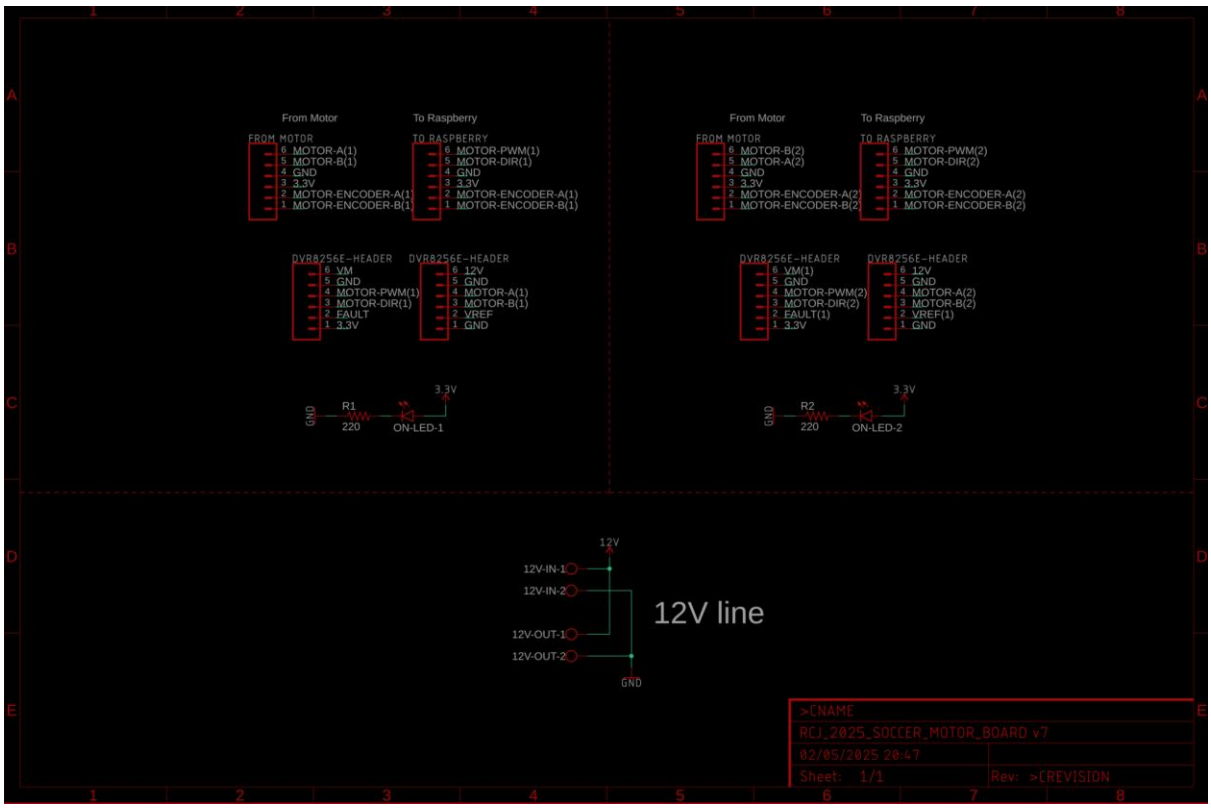
La PCB del pateador incluye un relay SRD-05VDC-SL-C, un capacitor 10000 μF, un LED blanco, una resistencia 2.5K ohm, dos borneras y un header de 3x1 macho.
El tamaño de las rutas es de 2.54 mm para la ruta de 38 voltios, y también para la ruta que conecta con el solenoide, de 1.27 mm para un LED que descarga el capacitor, y 0.508 mm para el voltaje que entra para activar el relay.
Esta placa se encarga de activar el solenoide (pateador) cuando reciba una señal de la Raspberry Pi Pico mediante el sistema del amplificador operacional, y cuando esto suceda el capacitor que ya se había estado cargando previamente a 38 voltios manda su carga hacia el solenoide, haciendo la función de patear.

6.9 Diagrama/Esquema eléctrico

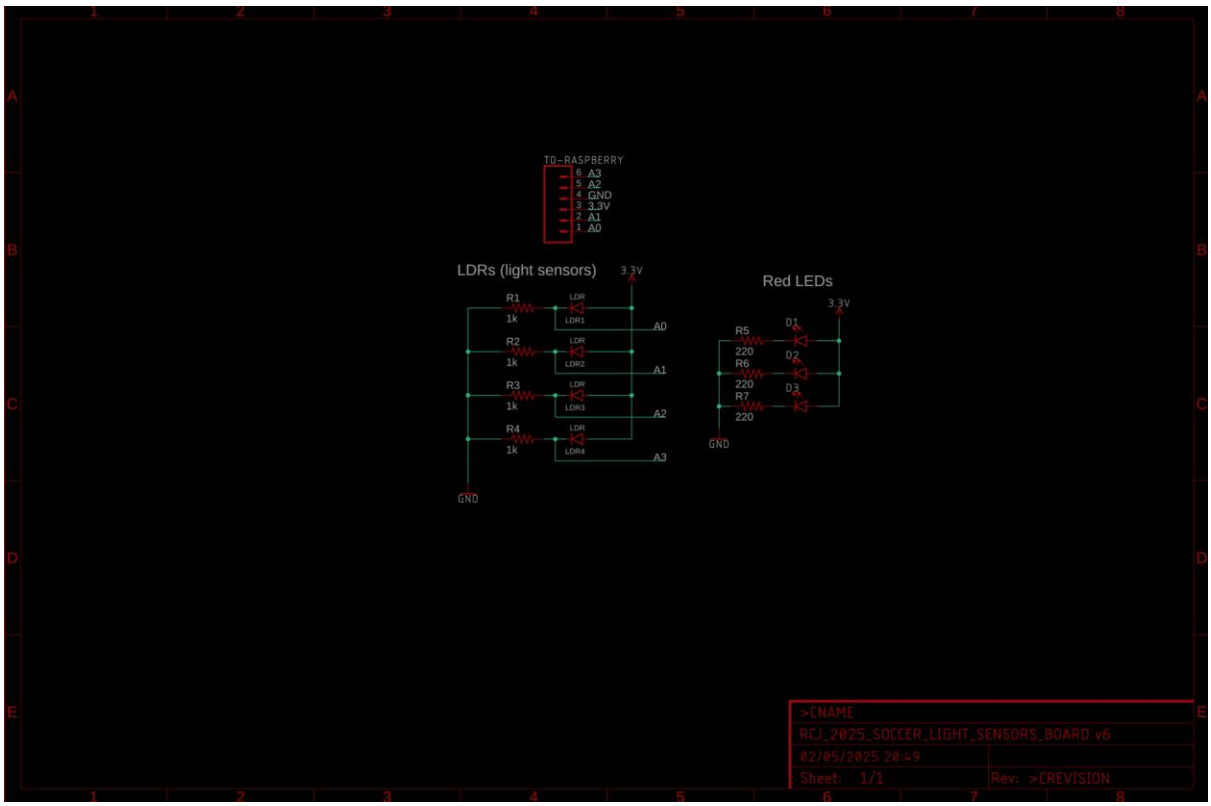
6.9.1 Esquemático de la placa de control



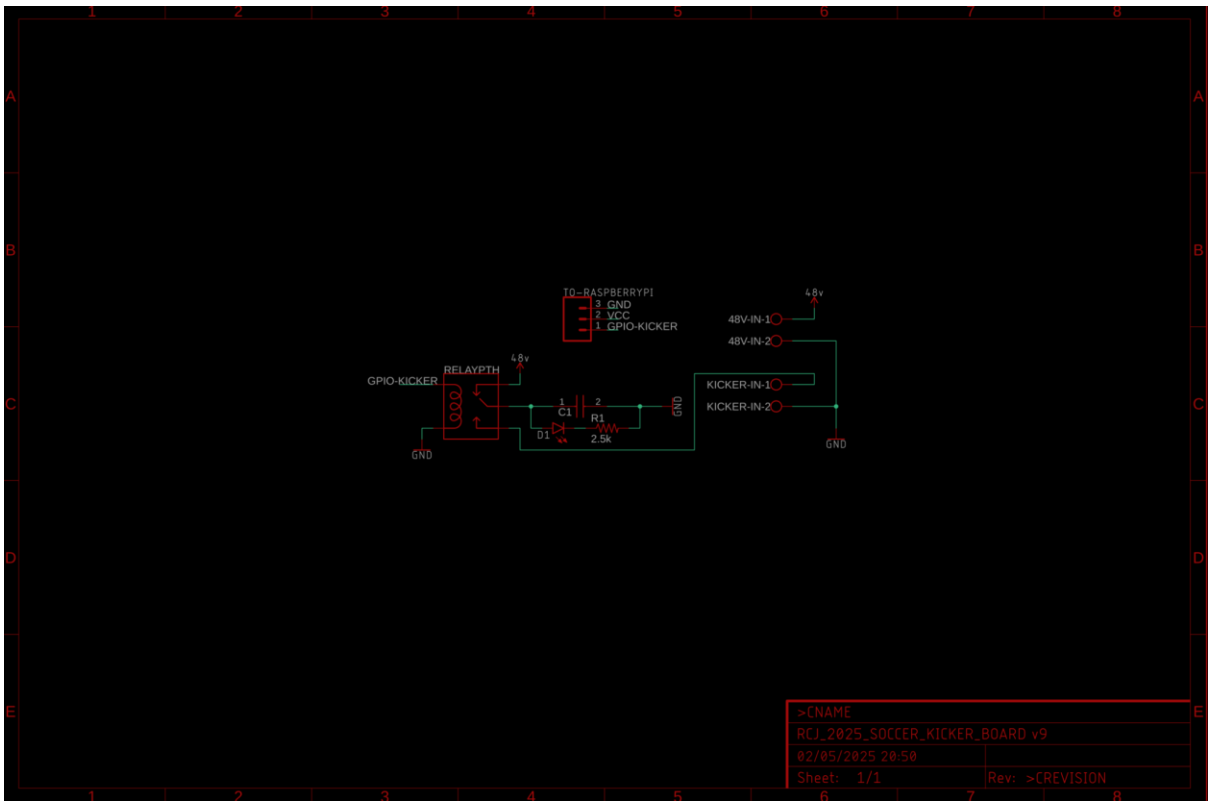
6.9.2 Esquemático de la placa de motores



6.9.3 Esquemático de la placa de sensores de luz



6.9.4 Esquemático de la placa del pateador

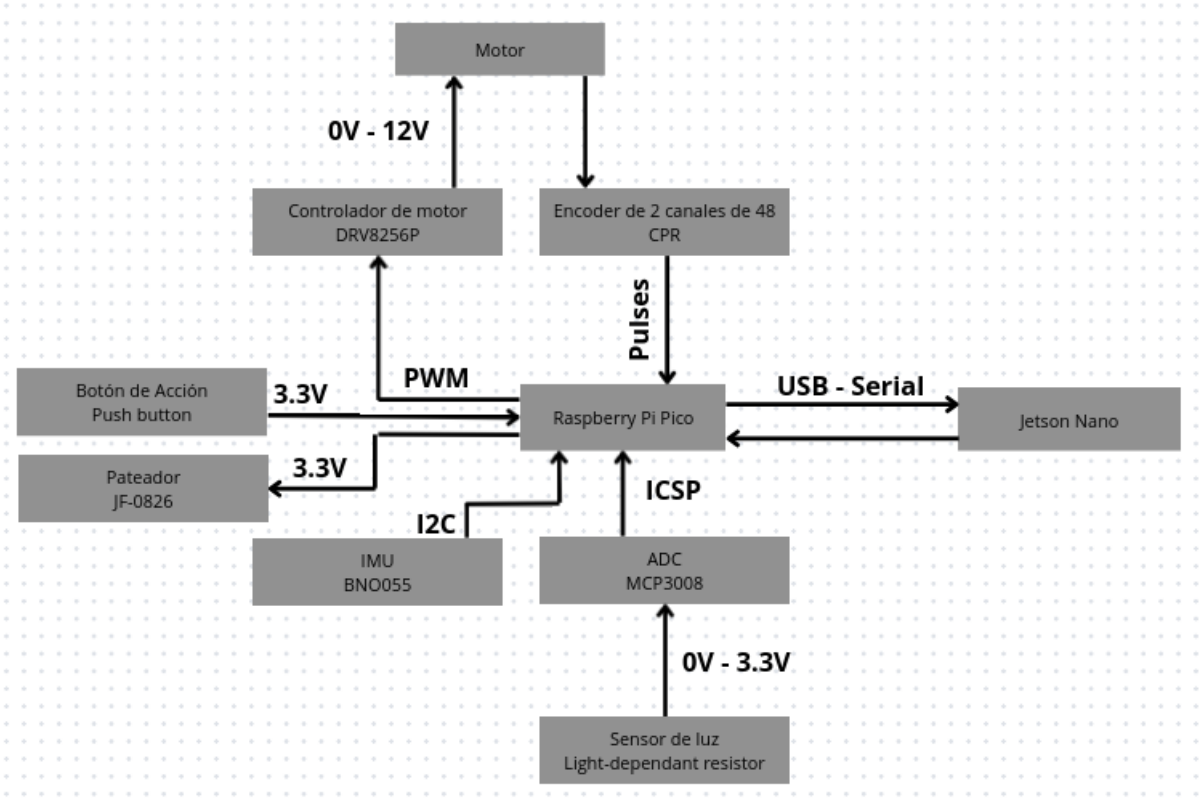


7. Software para interacción con hardware

El software para la interacción con el hardware recae en el microprocesador Raspberry Pi Pico. Este recibe la señal de componentes como IMU, Sensores de Luz y encoders, es también el encargado de controlar la locomoción y recibir comandos de la Jetson Nano.

7.1 Arquitectura de software

En el siguiente diagrama se puede apreciar cómo se conectan los componentes con el Raspberry Pi Pico



El código del Raspberry Pi Pico está programado en C++, utilizando el SDK C/C++ de Raspberry. Cada módulo del robot está encapsulado en su propia clase y tiene esta estructura

```
#ifndef MODULE_H
#define MODULE_H

#include "binarySerializationData.h"

class Module {
public:
   Module();
   void tick();
   void begin(BinarySerializationData& data);

private:
   BinarySerializationData* dataPtr;
   unsigned long long previous_tick{};
};

#endif //MODULE_H
```

En el código anterior se puede apreciar una librería llamada binarySerializationData.h (Véase sección 9). Esta contiene un objeto para la serialización de datos. Este objeto es el que se intercambia entre la Jetson Nano y la Raspberry Pi Pico.

El método tick() se ejecuta cada vez que la Raspberry Pi Pico entra en el bucle principal y actualiza el objeto de *dataPtr si es necesario enviar información a la Jetson Nano o recibir información de la misma.
(Para usos específicos de la clase, véase la sección 7.2, 7.3, 7.4 y 7.6)

7.2 Sensor de navegación (IMU)

Para el sensor de navegación IMU utilizamos una BNO055. Esta se comunica utilizando el protocolo I2C.
Para recibir el valor de Yaw (que es la única medida importante de la BNO055 utilizada en el programa). Se envían valores de acuerdo al datasheet de la BNO055 [6].

```
uint8_t reg_buf[1] = {0X1A}; // EUL_Heading_LSB [6, p. 52]
i2c_write_blocking(_port, _address, &reg_buf,  1, false);
i2c_read_blocking(_port, _address, &reg_buf, 1, false);

std::vector<double> xyz(3);
int16_t x = 0;
int16_t y = 0;
int16_t z = 0;

x = ((int16_t)buffer[0]) | (((int16_t)buffer[1]) << 8);

xyz[0] = ((double)x) / 16.0;
```

En el fragmento anterior se puede observar la lógica para obtener el valor de YAW desde la BNO055. el valor estará guardado en forma de un double.

En el método tick(), se ejecuta el fragmento de código anterior, y su resultado es convertido a un formato de -180 a +180, además de que se le resta un “offset”, este offset es calculado como el ángulo al que el robot está viendo en el momento que se enciende el robot utilizando el botón de acción.

7.3 Sensores de luz y entradas analógicas

Para la detección de líneas se utilizan fotorresistencias como sensores de luz. La Raspberry Pi Pico contiene 3 entradas con convertidor analógico a digital (ADC) [7]. Sin embargo, el robot contiene 8 fotorresistencias que dan una salida digital, por lo que se utiliza un componente llamado MCP3008, que es un convertidor analógico a digital de 10 bits y 8 canales [15].
Para recibir señales del MCP3008 se utiliza el siguiente fragmento de código

```
uint8_t tx_buf[3];
uint8_t rx_buf[3];
```

```
tx_buf[0] = 0x01; // Start bit
tx_buf[1] = (0x08 | channel) << 4; // Single-Ended + Channel Number
tx_buf[2] = 0x00; // Dummy byte

gpio_put(MCP_SPI_CSN, false);
sleep_us(1);

spi_write_read_blocking(MCP_SPI_PORT, tx_buf, rx_buf, 3);

sleep_us(1);
gpio_put(MCP_SPI_CSN, true);

const uint16_t result = ((rx_buf[1] & 0x03) << 8) | rx_buf[2];
```

El resultado del canal “channel” es luego guardado en un arreglo junto con los valores de los otros 8 canales. Este fragmento de código es llamado dentro de la función tick() que se ejecuta con una frecuencia de 1 vez cada $1 \times 10^{-3} s$.
El proceso de calibración para cada uno de los sensores de luz es de la siguiente manera

$$threshold = value + CALIBRATION\ OFFSET$$

Calibration offset es un número entero que puede cambiar dependiendo del material de la cancha, por lo general este valor se encuentra entre el 30 y 50.
Dentro del método tick se ejecuta este fragmento de código para hacer la detección.

```
if (channel_just_read < NUM_LDR_SENSORS) {
   ldr_readings[channel_just_read] = value;

   if (value > ldr_thresholds[channel_just_read]) {
      detection_active = true;
   }
}
```

Se utiliza este enfoque en lugar de analizar con un bucle cada uno de los resultados para encontrar si algún valor está por encima del threshold ya que es más eficiente y reduce la cantidad de operaciones que se deben realizar, ya que solamente se actualiza un único valor de sensores de luz por cada llamada al método tick(). Con este enfoque de realizar una única comprobación al único sensor que su valor cambia, se activa una variable llamada detection_active, la cual tiene que ser manualmente configurada a false después de que se haya tomado acción al encontrar la línea, que en este caso, es retroceder.

7.4 Control de motores

El control de motores se hace utilizando un driver para motores de DC DRV8256E que es un puente H que permite el control utilizando una señal PWM para controlar la velocidad y una señal de GPIO para controlar la dirección [12][16]. Se utilizan además motores de DC de pololu con un encoder de 48 CPR [17].

7.4.1 Control de movimiento

El movimiento se controla mediante una señal PWM para la velocidad y una señal GPIO para la dirección.
Este comportamiento se puede encontrar en el código de la Raspberry Pi Pico dentro de una estructura del motor individual.

```
void Motor::individualMotor::setSpeed(int16_t new_speed, bool
save_direction) {
   new_speed = std::clamp(new_speed, static_cast<int16_t>(-
MAX_SPEED), static_cast<int16_t>(MAX_SPEED));

   if (std::abs(new_speed) < MIN_SPEED) {
      if (std::abs(new_speed) < (MIN_SPEED / 2)) {
         new_speed = 0;
      } else {
         new_speed = MIN_SPEED * (new_speed > 0 ? 1 : -1);
      }
   }

   analogWrite(pwm_pin, std::abs(new_speed));
   digitalWrite(dir_pin, new_speed > 0);

   if (save_direction) {
      speed = std::abs(new_speed);
      if (new_speed != 0) direction = new_speed > 0 ? 1 : -1;
   }
}
```

En el fragmento anterior se puede ver el método setSpeed que forma parte de la estructura individualMotor, que a su vez forma parte de la clase Motor. En la función se aceptan dos argumentos, la velocidad y una variable que establece si se va a guardar la dirección.
El argumento de new_speed contiene la velocidad en forma de un número entero y la dirección en forma de un signo, ya sea positivo o negativo. Antes de enviar la señal al controlador se realizan dos guardias, para asegurarse que la velocidad no sea mayor a la MAX_SPEED que es establecida en la compilación (Normalmente 255). Y en caso de que la velocidad sea menor a MIN_SPEED se realizará una lógica para determinar si el motor debería moverse a MIN_SPEED o enviar 0.
El motivo de esta guardia es evitar se envíe una velocidad menor que la que el motor necesita para moverse.

Si $|new\ speed| < \frac{MIN\ SPEED}{2}$ entonces $new\ speed = 0$

Si $|new\ speed| > \frac{MIN\ SPEED}{2}$ entonces $new\ speed = MIN\ SPEED$

Las siguientes instrucciones envían a través de un pulso PWM la velocidad y a través de una señal digital la dirección.

La siguiente guardia de save_direction se utiliza para guardar o no guardar la información del movimiento. Los casos en los que save_direction podría deshabilitarse puede ser al momento de frenar (véase sección 7.4.3).

En el caso que la velocidad nueva sea igual a 0, no se actualizará la dirección ya que existe la posibilidad de que algún controlador PID esté utilizando la dirección para calcular el error. Ya que, en caso de no tener esta guardia, si un PID da como resultado un output de 0 (Lo que sucede normalmente en un PID con un objetivo de RPS muy bajo), la dirección se convertirá siempre en negativa y puede alterar los cálculos de las revoluciones por segundo.

7.4.2 Retroalimentación de movimiento

Para la retroalimentación del movimiento de cada motor individual se utilizan los encoders con 48 CPR de los motores [17]. Los encoders contienen dos canales A y B, contando la subida y bajada de cada canal regresarán 48 pulsaciones por rotación. Si se utiliza un único canal y se lee únicamente la subida, las pulsaciones por rotación se reducen a únicamente 12 CPR.
En el robot de Aizer se utiliza únicamente un canal y solamente se detecta la señal de `GPIO_IRQ_EDGE_RISE`, esto resulta en 12 pulsaciones por rotación. El motivo de esta elección es la velocidad de desarrollo, además que la precisión adicional que pueden proveer el resto de las pulsaciones por rotación no provee una diferencia significativa y solamente representa una carga más en el microprocesador de Raspberry Pi Pico.

La fórmula para determinar la cantidad de revoluciones por minuto es la siguiente:

$$RPS_{salida} = \frac{pulses \times 1x10^6}{\Delta T_{\mu s} \times CPR \times GEAR\ RATIO}$$

Donde

- RPS_{salida} indica la cantidad de revoluciones por segundo del segundo eje de salida del motor (la llanta).
- $pulses$ Es el número de pulsaciones detectados en el intervalo de tiempo $\Delta T_{\mu s}$
- $\Delta T_{\mu s}$: El intervalo de tiempo en microsegundos en la cual se contaron los pulses recibidos del encoder
- CPR: Es una constante que indica la cantidad de pulsos por revolución. Para el caso de los motores del robot de aizer, es de 12 CPS [17].
- $GEAR\ RATIO$: Es una constante que indica la relación exacta de transmisión del motor. Para el caso de los motores de aizer, es de 9.68 [17].

7.4.3 Freno

El controlador de motores de DC DRV8256E no contiene una lógica de freno activo como sí lo contienen otros controladores de motores como el puente H L298 [18]. Es por eso que se utiliza una lógica de freno que busca “contrarrestar” el movimiento del motor enviando una señal inversa a los motores para frenar.
La lógica del freno se ejecuta enviando la señal máxima MAX_SPEED en la dirección contraria a la que el motor se esté moviendo. La lógica de freno dejará de enviar la señal de MAX_SPEED:

- Si el motor se está moviendo a una velocidad de $1.5\ RPS$ o menor
- Si $RPS_{anterior} + 0.2 < RPS_{actual}$ lo que indica que la llanta ya está girando en el sentido opuesto.

7.5 Control PID para dirección del robot

El control para la dirección del robot es esencial para mantener un control estable y predecible del robot. La decisión de utilizar un control PID/PI sobre otro algoritmo de control como el control en cascada, proviene del hardware necesario para poder realizar estos algoritmos de control y sobre la facilidad y recursos en línea para implementarlos. El control en cascada necesita de dos procesos, rápido y lento, los cuales requieren su propio hardware específico para la retroalimentación de ambos procesos [10]. Para el caso de un control en cascada para el control de un motor se podría utilizar un encoder (proceso principal lento) y un sensor de corriente (proceso secundario rápido). Para el control PID únicamente es necesario un sistema de retroalimentación [11]. En el caso del robot de AIZER, se utiliza un encoder como único sistema de retroalimentación para el control directo de un motor.
El control del chasis completo, es decir, utilizando 4 ruedas, utiliza otro control de PID, que toma como retroalimentación el ángulo de la BNO055 para alinear el robot en el caso que este sufra alguna perturbación grande, como lo puede ser un golpe por otro robot, el choque con alguna parte de la cancha o incluso alguna falla en el controlador de un motor en específico, permitirán seguir manteniendo un control, aunque menos estable, predecible del robot [11].

7.5.1 Objetivos y metas

Para la calibración de los motores individuales, se utiliza un algoritmo de PID, la manera en la que se calibran los motores en un control PID pueden ser de distintas maneras, incluyendo un prueba y error, o el método de Ziegler Nichols. Por simplicidad y tiempo se utilizará el método de prueba y error ya que requiere menos conocimiento técnico y la precisión que necesitamos no debe ser tan exacta [11]. La meta del motor es acercarse lo más posible a las revoluciones por segundo (RPS) que se establezcan.

7.5.2 Metodología

Para la calibración del PID del control de los motores de DC se utilizarán 3 medidas.

1. Promedio: Se promedian las revoluciones por segundo en un plazo de aproximadamente 300 mediciones. (Cada medición se realiza cada aproximadamente 25ms) y se buscará que estén ±0.15 puntos del objetivo
2. Variabilidad del output: La variabilidad del output se medirá únicamente de manera visual, se buscará que no hayan caídas o subidas pronunciadas. Esto se debe a una limitación de hardware ya que el cambio rápido de velocidades puede provocar un sobrecalentamiento y que los motores se calen [12] [13].
3. Error integral sobre el tiempo: Se busca que el error integral se vuelva estable, no se necesita una precisión muy alta por lo que únicamente con un análisis visual para asegurarse que el error integral no se acumule infinitamente.

7.5.3 Implementación

La implementación del PID en el código se compone de una estructura que contiene los datos necesarios para el cálculo del PID.

```
struct  PidParameters {
   float kp{};
```

```
    float ki{};
    float kd{};

    double max_output = 0.0;
    double min_output = 0.0;
    float error_threshold = 0.0;

    double integral_error = 0.0;
    double previous_error = 0.0;
    unsigned long long last_iteration_ms = 0;

    bool first_run = true;

    double target = 0.0;
    double error = 0.0;
    double output = 0.0;

    bool one_direction_only = false;
    bool accept_type = PID_ACCEPT_POSITIVES_ONLY;

    bool reset_within_threshold = false;
};
```

En el fragmento de código anterior, se muestra la estructura PidParameters, la cual contiene constantes como K_p, K_i y K_d, contiene además constantes como max_output, min_output y error_threshold.

Las constantes de max_output, min_output y error_threshold contienen información acerca de los límites del control pid.

- max_output contiene el output tanto positivo como negativo máximo que puede alcanzar el output del PID. En caso que el output sea mayor que este límite, el output se reemplazará por max_output.
- min_output contiene la misma lógica que max_output pero para el límite inferior.
- error_threshold muestra cual es el error permitido, si el error se encuentra entre $0 < |error| < error\ threshold$, output se calculará sin tomar en cuenta la integral y si la variable reset_within_threshold, la estructura del PID será reiniciada.

El cálculo para un PID continuo se puede lograr con la siguiente fórmula

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

donde:

- $u(t)$ es la salida del controlador en el tiempo t
- $e(t)$ es el error (diferencia entre el valor deseado y el valor medido) en el tiempo t
- $K_p\ K_i\ K_d$ son las ganancias proporcional, integral y derivativa.
- $\int$ la integral, y $\frac{d}{dt}$ La derivada son operaciones de cálculo que asumen tiempo y valores continuos.

La naturaleza de un microcontrolador como la raspberry pi pico es que es un dispositivo digital, es decir que opera a pasos de tiempo discretos definidos por su reloj [7][14]. Esta limitación, aunque permite simular continuidad, no puede medir ni actuar de forma verdaderamente continua.

Para aplicar las matemáticas discretas se hará de la siguiente manera:

- Tiempo discreto: En lugar de un tiempo continuo t, se trabajará con pasos de tiempo discretos k, donde cada paso está separado por un intervalo de muestreo T_s
- Muestreo: Se medirá el error $e(t)$ solo en estos instantes discretos, resultando en una secuencia de errores $e(k)$
- Aproximación de la integral: La integral $\int_0^t e(t)dt$ se aproxima mediante la suma de los errores medidos, multiplicado por el intervalo de tiempo. Se aplicará de la siguiente manera:

$$Término\ integral \approx K_i T_s \sum_{i=0}^{k} e(i)$$

- Aproximación de la derivada: La derivada $\frac{de(t)}{dt}$ se aproxima mediante la diferencia entre el error actual y el error anterior, dividida por el intervalo de tiempo. Se aplicará de la siguiente manera:

$$Término\ derivativo \approx K_d \frac{e(k) - e(k-1)}{T_s}$$

La aplicación discreta del controlador es entonces de la siguiente manera:

$$u(k) = K_p e(k) + K_i T_s \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k-1)}{T_s}$$

Además de la implementación base del PID, se han agregado modificaciones para seguridad o mejor funcionamiento del control PID. Estas modificaciones son:

- implementación de $first\ run$: La primera vez que se ejecuta el control PID después de un reinicio o después de la inicialización, $u(t)$ se calculará de la siguiente manera.

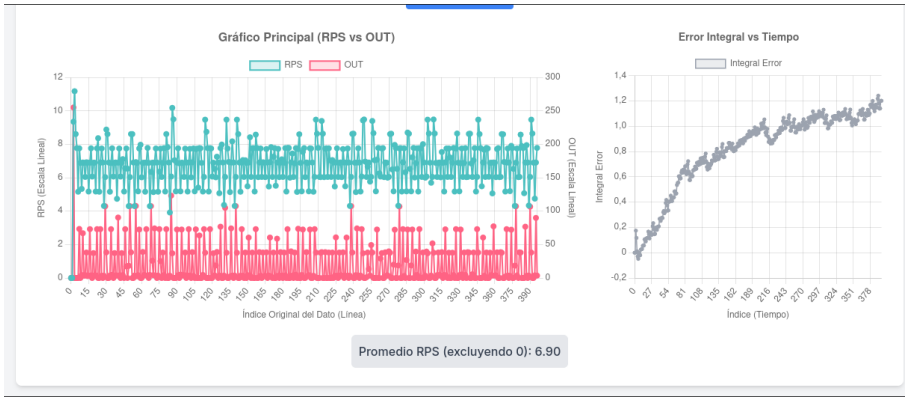
  $$u(t) = K_p e(k)$$

  La integral y la derivada no son tomadas en cuenta en la primera iteración, ya que no hay suficientes datos para calcular la derivada ni la integral, como pueden ser el error previo o T_s
- Banda muerta (deadband): Si el error es menor que el error_threshold, no se aplicará la integral. El objetivo de esta banda muerta es evitar que pequeños errores residuales o ruido no se sigan acumulando. Esto permite reducir la cantidad de sobreimpulsos u oscilaciones causadas por la integral innecesaria cerca del objetivo (target)
- Anti-Wind up: Si existe un límite definido superior max_output y el output del ciclo anterior del pid es mayor o igual al max_output y el signo del error intenta empujar la salida aún más de ese límite, se deshabilitará la integral ya que la acumulación de error es tan grande que podría llegar a ser perjudicial.
- One Direction Only: Esta implementación tiene como objetivo evitar el sobrecalentamiento de los controladores de los motores de DC, ya que el cambio brusco entre salidas con signo positivo y negativo, pueden aumentar significativamente la temperatura del controlador. Es por eso que la constante de one_direction_only se activa para el controlador PID de los motores individuales, que están en contacto directo con el controlador del motor DC. Si esta constante está

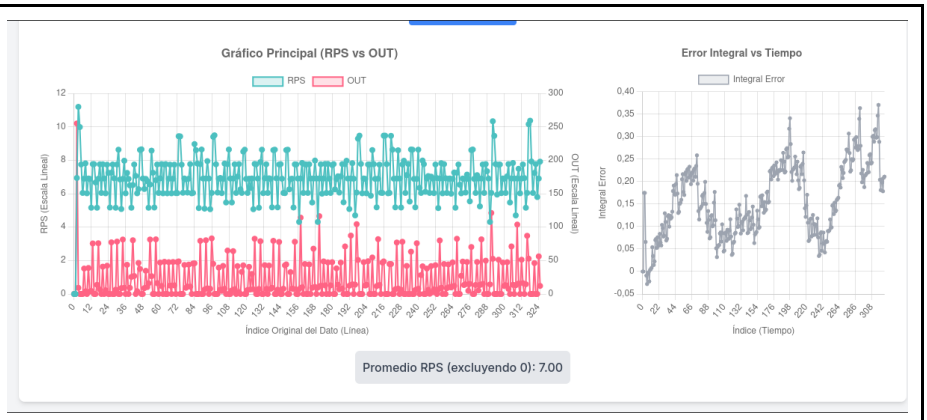
activada con true, se espera una actualización de accept_type en tiempo real, esta variable decide la dirección de valores que aceptar y cuales rechazar y regresar a 0.

7.5.4 Calibración

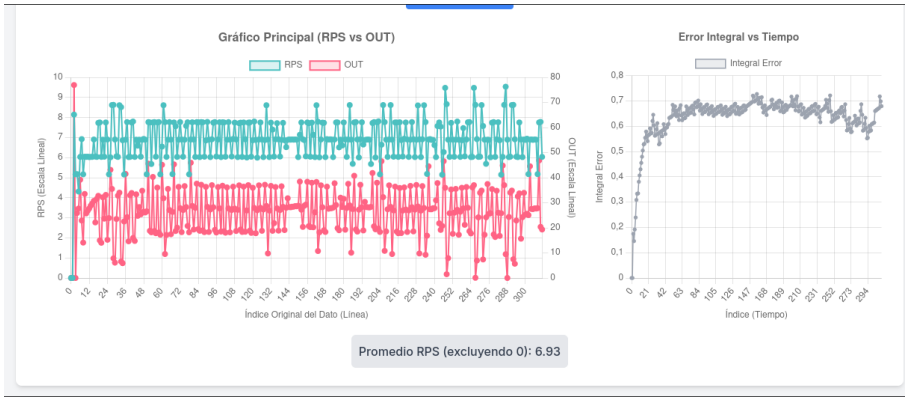
La calibración que se realizará en esta sección es dedicada al PID para el movimiento individual de un motor.
Para comenzar la calibración, se iniciará con “dummy values” que servirán como punto de partida para comenzar la calibración. Se utilizará un programa para graficar que muestra sobre el tiempo, las revoluciones por segundo (RPS), el output (PWM) que es enviado a los controladores y el error integral.

Utilizando un controlador proporcional únicamente se puede ver como el promedio de los RPS es igual a 6.90 que está dentro del límite de $\pm 0.15 RPS$ que se espera de los motores. Sin embargo, el error integral sobre el tiempo no parece estabilizarse de manera rápida y continúa creciendo dentro de los primeros 300 ciclos de ejecución del PID. Además, que el output varía considerablemente entre el 0 y valores cercanos al 50 e incluso llegada hasta 100. Este comportamiento es esperado sin una ganancia integral, ya que una vez que el objetivo de RPS se llega, el output será igual a 0 y el motor comenzará a frenar. Para la siguiente calibración se modificará el valor de K_i para intentar resolver estas observaciones.	
$K_p = 40$ $K_i = 0$ $K_d = 0$ $Target = 7\ RPS$	

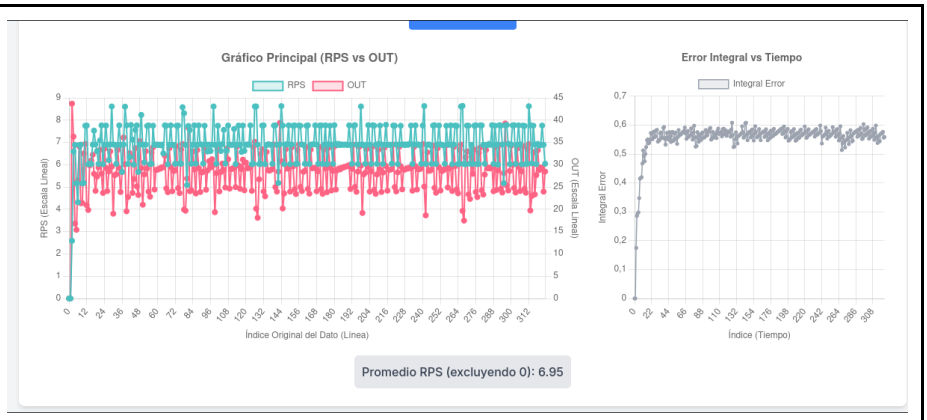
Se puede ver en esta nueva calibración que el objetivo de RPS se cumple correctamente y está dentro del rango de $\pm 0.15 RPS$, sin embargo, el error integral sobre el tiempo muestra saltos grandes y el output sigue variando entre 0 y 50, lo cual no es esperado.
De igual manera, si prestamos atención a los valores de out se puede observar como no siempre tocan el 0 como en la calibración anterior, sino que se mantienen arriba del 0 por unos cuantos puntos.
El sobre-impulso y los cambios de RPS y OUT rápidos provienen de una ganancia proporcional muy alta. Es por eso que, para la siguiente calibración, se reducirá el valor de K_p

$K_p = 40$ $K_i = 40$ $K_d = 0$ $Target = 7\ RPS$	

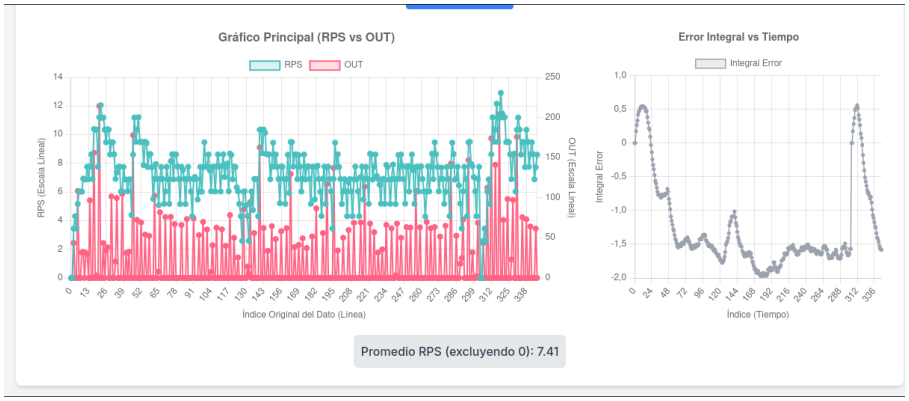
En esta calibración se puede ver un mejor desempeño del control, reduciendo la ganancia proporcional se han eliminado gran cantidad de los sobreimpulsos en el OUT dando un control más estable y predecible, además que el error integral sobre el tiempo se muestra estable y el promedio de RPS está dentro del rango de $\pm 0.15\ RPS$. Para la siguiente calibración podría reducirse aún más la ganancia proporcional y aumentar la ganancia integral para ver si es un cambio favorable.

$K_p = 10$ $K_i = 40$ $K_d = 0$ $Target = 7\ RPS$	

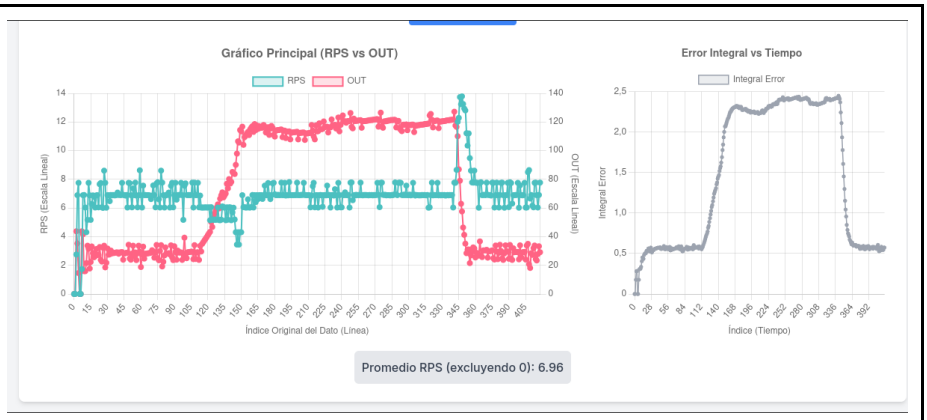
Con las nuevas ganancias proporcionales e integrales se ve un desempeño aún mejor del error integral sobre el tiempo, ya que se mantiene estable y las variaciones en el OUT son cada vez menos, y el promedio de RPS sigue manteniéndose dentro del objetivo de $\pm 0.15\ RPS$.
Este es un controlador PI, en la siguiente calibración se intentará un valor de K_d únicamente para ver el comportamiento de la gráfica con esa nueva variable. Sin embargo esta calibración es prometedora ya que cumple los 3 requisitos que se pedían para poder aceptarla.

$K_p = 5$ $K_i = 50$ $K_d = 0$ $Target = 7\ RPS$	

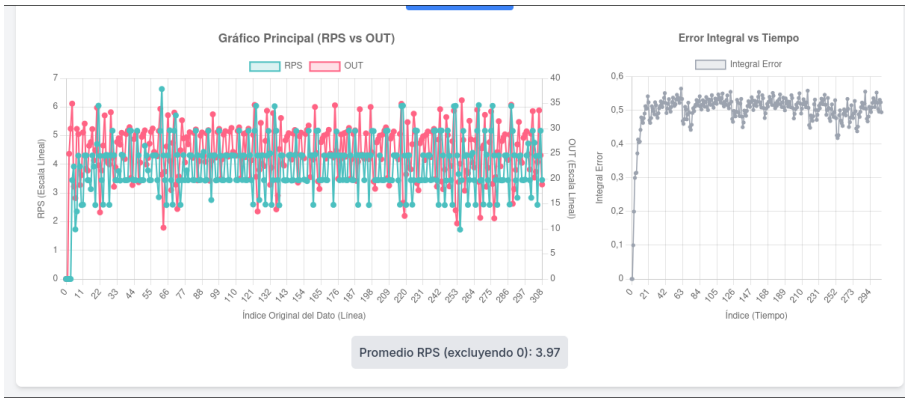
En esta nueva calibración se agregó un valor a la ganancia derivativa, sin embargo, esta provoca un efecto no deseado ya que es posible que la ganancia derivativa haya sido muy alta. Ninguno de los 3 requisitos se cumplen por lo que se rechazará este cambio y se quedará la calibración anterior.

$K_p = 5$ $K_i = 50$ $K_d = 2$ $Target = 7\ RPS$	

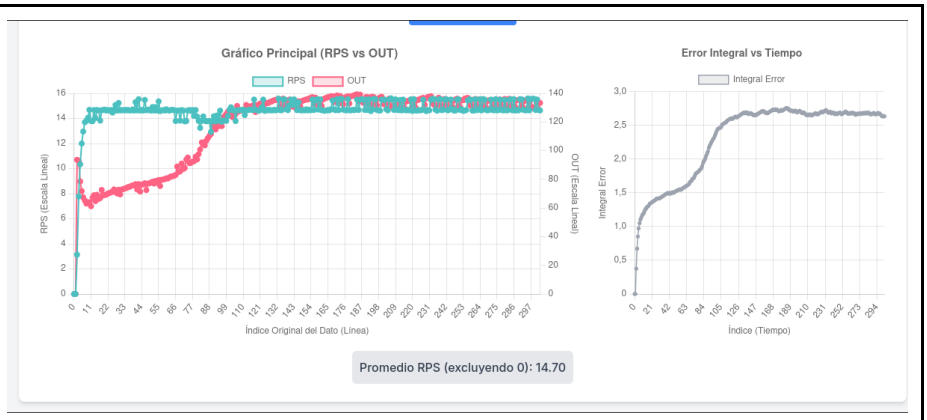
Utilizando la calibración que mejor cumplió los requisitos, se agregó una perturbación en a partir del ciclo número 100. La perturbación fue manualmente tocar la llanta para forzar un pequeño frenado. Se puede ver como el PID reaccionó de manera rápida y fue capaz de mantener los RPS estables incluso con la perturbación. y al momento de eliminar la perturbación cerca del ciclo 350, el PID tiene un sobreimpulso que lo lleva hasta los 14 RPS pero es rápidamente contrarrestado el rápido movimiento de los motores.

$K_p = 5$ $K_i = 50$ $K_d = 0$ $Target = 7\ RPS$	

Para la siguiente prueba se mantendrá la calibración, sin embargo se probará con un objetivo más difícil que es de 4 RPS. Se puede ver como el OUT y los RPS oscilan de una manera más agresiva que con un objetivo de 7 RPS, sin embargo el error integral y el objetivo de $\pm 0.15\ RPS$ se cumplen de manera correcta.

$K_p = 5$ $K_i = 50$ $K_d = 0$ $Target = 4\ RPS$	

Como última prueba se intentará con un objetivo aún más ambicioso de 15 RPS, se puede ver un OUT y un RPS muy estable a lo largo del tiempo, sin embargo el promedio de los RPS no están dentro del requisito de $\pm 0.15\ RPS$, sin embargo esto es un escenario poco realista ya que en el juego el robot nunca se moverá a una velocidad tan rápida.

$K_p = 5$ $K_i = 50$ $K_d = 0$ $Target = 15\ RPS$	

7.5.5 Resultados

La calibración se realizó de una manera exitosa y se logró un resultado esperado que cumplía con los 3 requisitos establecidos en la metodología, se utilizará este método de calibración para cada uno de los 4 motores del robot.

7.6 Pateador

El pateador utiliza un complejo sistema de transistores, relays, capacitores y elevadores de voltaje para su activación de lado de la electrónica (Véase sección 6.4.2). De lado de software se requiere únicamente una señal de GPIO para controlar su movimiento.
El pateador tiene una guardia integrada en la Raspberry Pi Pico que limita cada cuanto se puede activar el pateador. El tiempo entre cada activación está limitado por la cantidad de tiempo que toma cargar el capacitor a un voltaje mínimo para poder atravesar la cancha de portería a portería (Véase sección 6.4.3).

7.7 Arranque del robot

El arranque del robot se compone de un botón denominado “Botón de acción” ya que cambia su comportamiento dependiendo del estado del robot.

- Si el robot está apagado: El robot permitirá el movimiento de los motores, reiniciará todos los PIDs, todas las variables, calibrará los sensores de luz y colocará un offset en el ángulo de la brújula
- Si el robot está encendido: El robot bloqueará todas las señales de movimiento y disparará el kicker como señal de apagado

8. Software para visión

La visión del robot se basa en un sistema dual de cámaras que están posicionadas de manera que obtengan una visión casi completa de la cancha (Véase sección 5.5). La elección de las

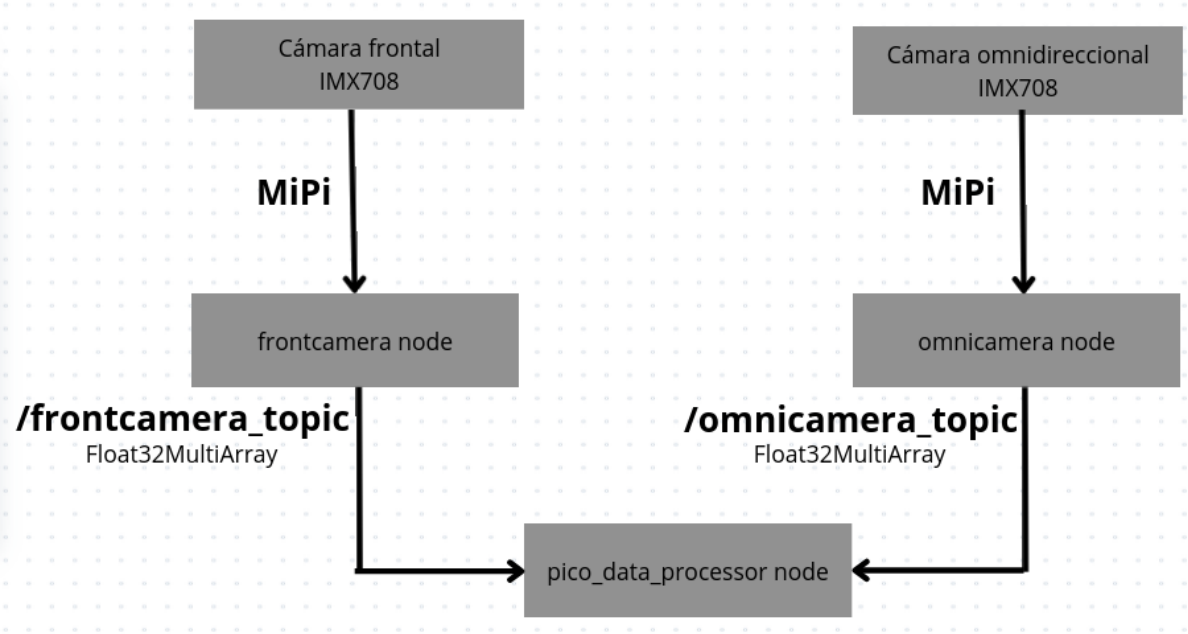
cámaras fue un modelo de Arducam que integra un sensor IMX708 [24]. Se eligió la Jetson Nano para el procesamiento de las imágenes para hacer pruebas utilizando modelos de detección de objetos.
Se probaron dos métodos distintos para la detección de los objetos en la cancha:

- Detección de objetos usando un modelo de YOLOv8n
- Detección de blobs utilizando thresholds.

La detección de objetos usando un modelo de YOLOv8n es la más atractiva, y fue el motivo para la elección de una Jetson Nano como procesador de imágenes. El objetivo de utilizar dos cámaras que detectan la pelota a por lo menos 14 frames por segundo por cámara es una meta ambiciosa y que si bien podría llegar a ser posible [26], con la limitación de tiempo decidimos descartar la idea y utilizar detección de blobs utilizando thresholds HSV.

8.1 Arquitectura de software

En el siguiente diagrama se observa la comunicación entre nodos y las cámaras.



8.2 Detección de objetos

Como discutido anteriormente, la detección de los objetos dentro de la cancha se basa en una búsqueda de blobs que estén dentro de un rango HSV.
La detección de los blobs se hace utilizando funciones nativas de OpenCV 4.1.1, de la siguiente manera [33].

```
struct Blob {
   int x;
   int y;
   int width;
   int height;

   long area() const {
```

```
        if (width < 0 || height < 0) return 0;
        return static_cast<long>(width) * height;
    }


cv::Mat frame; // Frame contiene la imagen de la cámara.

cv::Mat hsv_frame;
cv::cvtColor(frame, hsv_frame, cv::COLOR_BGR2HSV);

cv::Mat mask;
cv::inRange(hsv_frame, lower_bound, upper_bound, mask);

std::vector<std::vector<cv::Point>> contours;
std::vector<cv::Vec4i> hierarchy;

cv::findContours(mask,   contours,   hierarchy,   cv::RETR_EXTERNAL,
cv::CHAIN_APPROX_SIMPLE);

std::sort(contours.begin(),         contours.end(),         [](const
std::vector<cv::Point> &a, const std::vector<cv::Point> &b) {
    return cv::contourArea(a) > cv::contourArea(b);
});

std::vector<Blob> blobs;

for (const auto &contour : contours) {
    double area = cv::contourArea(contour);

    if (area > min_area && area < max_area) {
        cv::Rect rect = cv::boundingRect(contour);
        Blob blob = {
            rect.x,
            rect.y,
            rect.width,
            rect.height
        };

        blobs.push_back(blob);
    }
}
```

En el fragmento de código anterior se puede ver el proceso para detectar los blobs que estén dentro del rango HSV lower_bound y upper_bound, además son ordenados por tamaño y guardados dentro de un vector de objetos Blob llamado ‘blobs’.

8.3 Cámara frontal

La cámara frontal utiliza un sensor IMX708 con un lente gran angular de Arducam [24]. La cámara frontal está posicionada de manera en la que se pueda ver cuando el robot tiene posesión de la pelota y se puede ver la portería contraria estando pegado al otro extremo de la cancha.

8.3.1 Calibración de cámara frontal

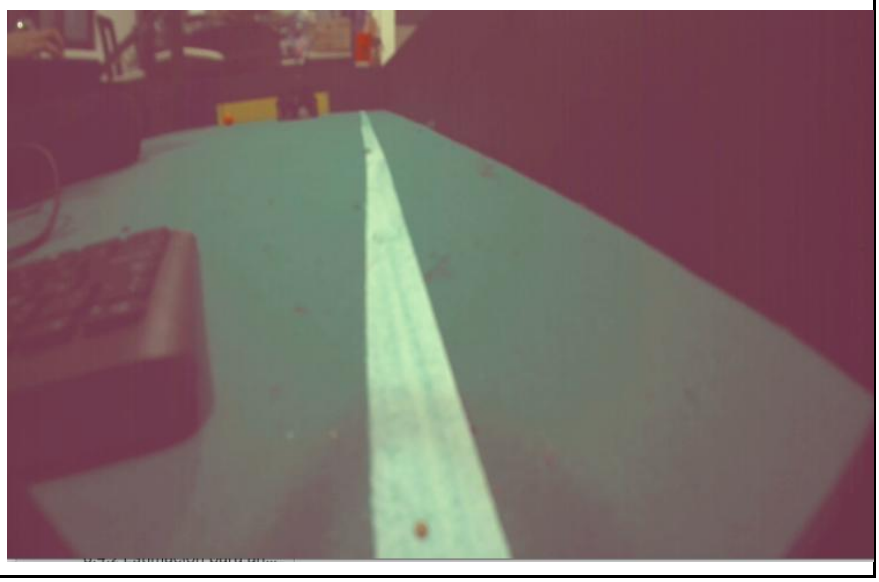
El sensor IMX708 regresa un esquema de colores bastante básico, se necesitan herramientas de post-procesamiento para hacer el color de la pelota más reconocible y diferenciable para la detección de la pelota.
El método que se utilizó fue de prueba y error, buscando obtener

En la siguiente imagen se puede ver la imagen directamente obtenida de la cámara, sin pasar por ningún tipo de filtro. En la imagen se puede apreciar una pelota, una portería y un robot de AIZER. Los colores no son del todo claros y tampoco se acercan a los colores reales. Sin embargo, es posible diferenciarlos. Para la siguiente calibración se colocará una saturación de 2 para ver si los colores se vuelven más brillantes, y por ende, más fáciles de detectar.	
Preprocesamiento: ● Ninguno	

En la siguiente imagen, se alcanza a ver el mismo escenario, con la diferencia de que los colores están más resaltados y se alcanza a ver una diferencia más notable entre la pelota, la portería y el resto de objetos en la cancha. Esta calibración es buena ya que a pesar de que los colores no son cercanos a los colores que se pueden percibir en la vida real, la diferencia de colores con el resto de objetos en la cancha permite fácilmente su detección.

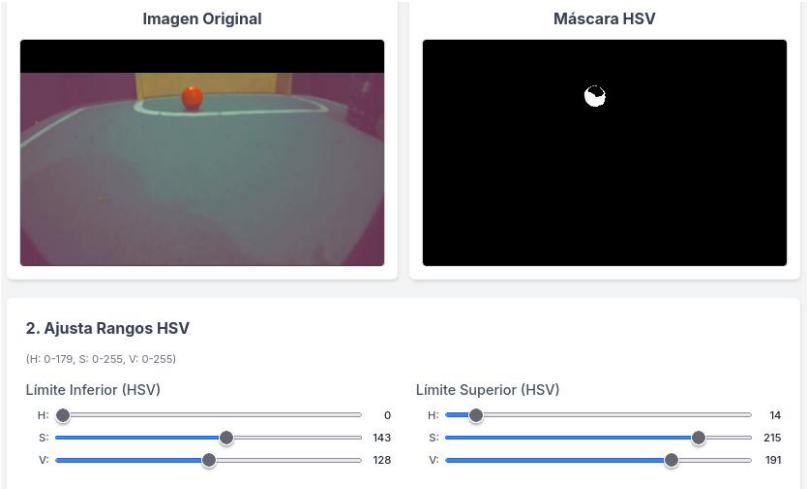
Preprocesamiento ● Saturación: 2	

Para el siguiente ejemplo, se posicionó el robot pegado a la esquina derecha de la portería azul de la cancha. En la imagen se puede apreciar el mismo escenario, sin embargo por la distancia y el lente gran angular, se pueden percibir objetos externos a la cancha, incluyendo un objeto extraño de color naranja que puede llegar a interferir con la detección de colores. Es por eso que para la siguiente calibración se colocará una barrera utilizando software para bloquear posibles objetos extraños ajenos a la cancha.

Preprocesamiento ● Saturación: 2	

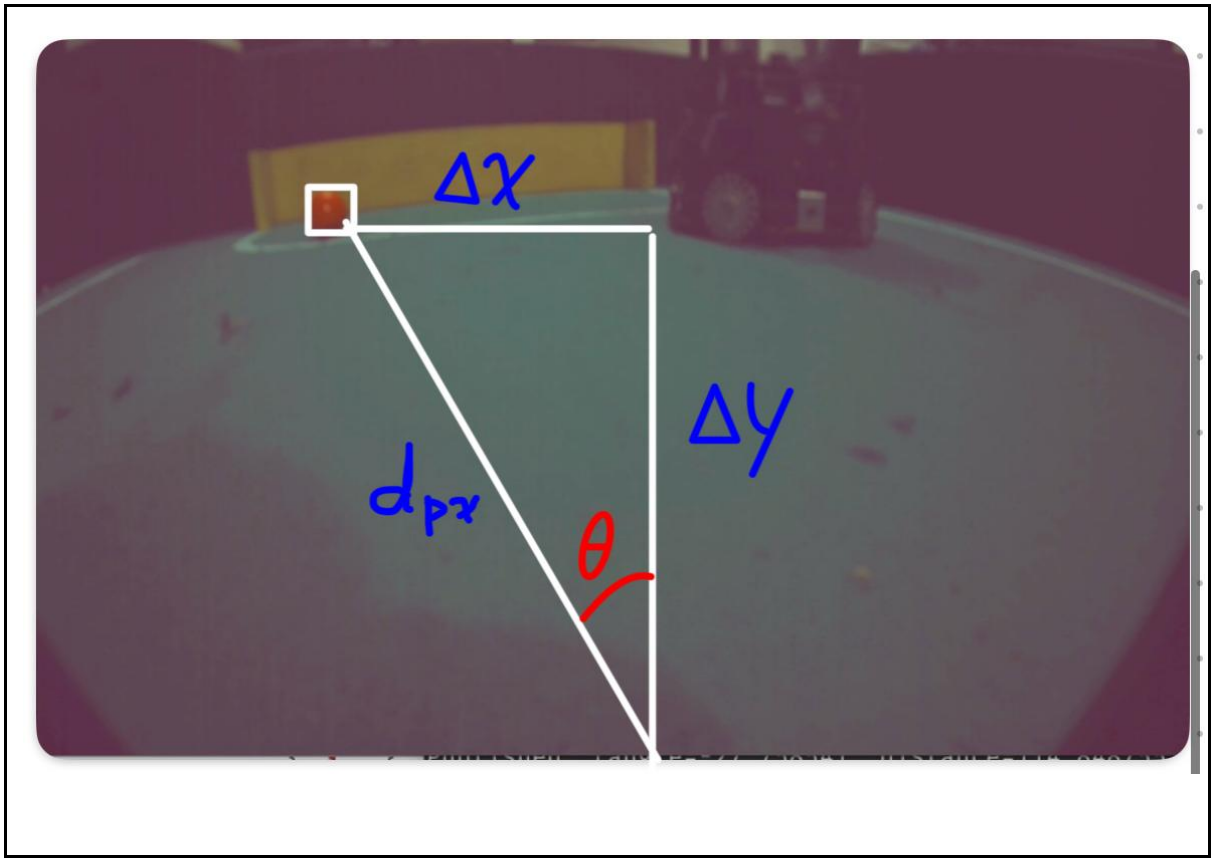
Utilizando OpenCV, se dibujo un rectángulo de color negro que abarca todo el ancho de la imagen y se posiciona desde $y_{start} = 0$ hasta $y_{end} = 69$. y_{end} contiene la coordenada del eje Y en donde se dejan de ver objetos externos a la cancha, esta variable tiene que ser manualmente calibrada antes de cada partido.

Preprocesamiento ● Saturación: 2 ● Rectángulo negro ajustado manualmente	

Por último, se acercará de nuevo el robot a la primer escena para buscar una combinación de valores de HSV que mejor representen el color de la pelota	
Preprocesamiento: ● Saturación: 2 ● Rectángulo negro ajustado manualmente Procesamiento: ● Lower bound: (0, 143, 128) ● Upper bound (14, 215, 191)	

8.3.2 Estimación de ángulo y distancia

El ángulo y la distancia de los objetos son esenciales para el desarrollo de una buena estrategia (Véase sección 10). Para estimar el ángulo de utilizan ecuaciones trigonométricas de la siguiente manera



Para la estimación del ángulo se plantean tres líneas imaginarias donde:

- $\Delta x = \left(ball_x + \frac{ball_{width}}{2}\right) - CAMERA\ ORIGIN\ X$
- $\Delta y = CAMERA\ ORIGIN\ Y - \left(ball_y + \frac{ball_{height}}{2}\right)$

Donde CAMERA ORIGIN X, Y contienen las coordenadas del origen de la cámara, comúnmente el valor se define de la siguiente manera

- $CAMERA\ ORIGIN\ X = \left(\frac{IMAGE\ WIDTH}{2}\right)$
- $CAMERA\ ORIGIN\ Y = IMAGE\ HEIGHT$

Para encontrar el ángulo θ se utilizará la función trigonométrica *arctan* de la siguiente manera:

$$\theta = arctan(\frac{\Delta x}{\Delta y}), \forall \Delta y \neq 0$$

Para encontrar la distancia de la pelota se puede utilizar la siguiente fórmula

- $d_{px} = \sqrt{\Delta x^2 + \Delta y^2}$

La distancia está expresada en píxeles, gracias al ángulo de inclinación de la cámara, se puede asumir que entre mayor sea Δy la distancia será mayor. Para convertir la distancia en una unidad lineal como centímetros, se utiliza una fórmula que aproxima d_{px} a d_{cm}.

Para eso se tomaron 15 puntos de referencia, se tomó la distancia real en centímetros y d_{px}, y utilizando una herramienta de curve-fitting, se buscó una función que aproxime lo más posible d_{px} a d_{cm}.

Se utilizó primeramente un modelo exponencial, que se puede ver como aproxima d_{px} a d_{cm}, sin embargo falla completamente en los datos cercanos a 0, lo que no es deseado ya que de acuerdo a las estrategias (Véase sección 10), es mucho más importante tener precisión en valores cercanos a 0.

Exponential

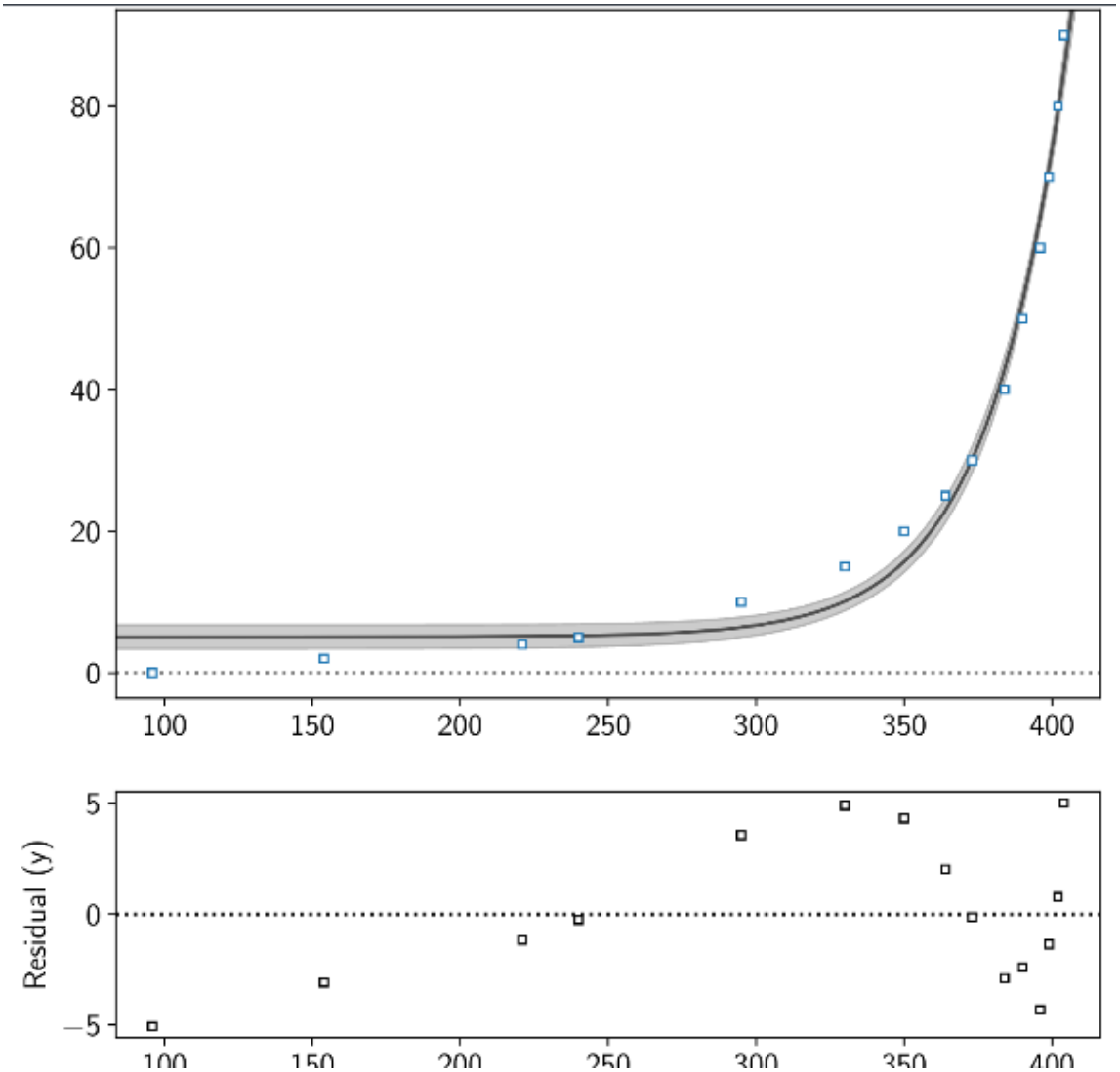
$y = Ae^{bx} + C$

δx ☐

δy ☐

Parameter	*Value*	*Uncertainty*
A	2.178e-05	2.809e-05
b	3.741e-02	3.204e-03
C	5.078e+00	1.601e+00

SSR = **1.57e+02**



Para la siguiente prueba se usó un modelo de Power Law, que si bien su SSR (Sum of squared residuals) es mayor, la precisión en los valores más cercanos a 0 es mejor.

Power Law

$y = Ax^B$

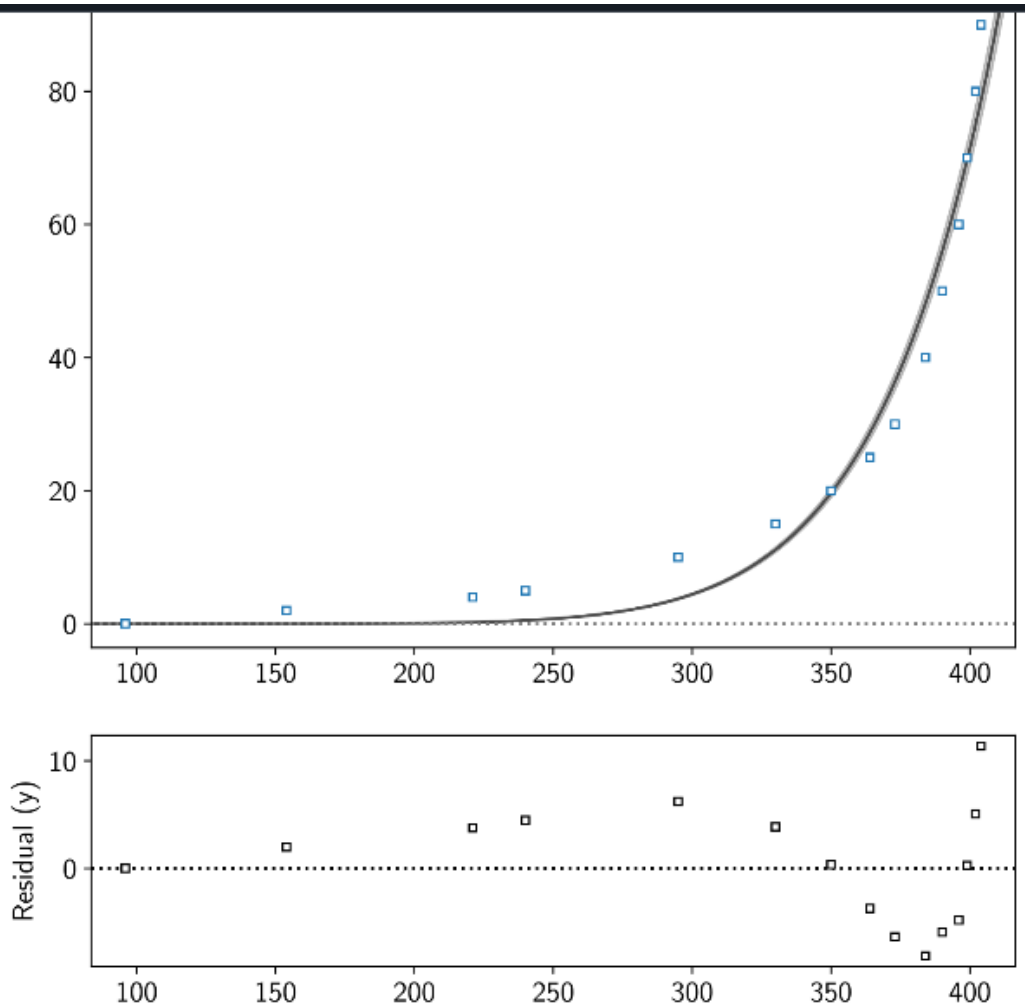
δx ☐

δy ☐

Parameter	*Value*	*Uncertainty*
A	5.105e-24	1.654e-25
B	9.664e+00	0

Jacobian not full rank, uncertainties may be zero

SSR = **4.26e+02**



Como última prueba, se utilizó el modelo de Lorenz, y a pesar que su función no se relaciona mucho con el propósito en esta calibración, obtuvo un mejor resultado, como se puede observar en su SSR y la precisión en los primeros valores es bastante acertada.

Lorentzian

$$y = \frac{A}{\sqrt{(\omega_o^2 - x^2)^2 + 4x^2\beta^2}} + C$$

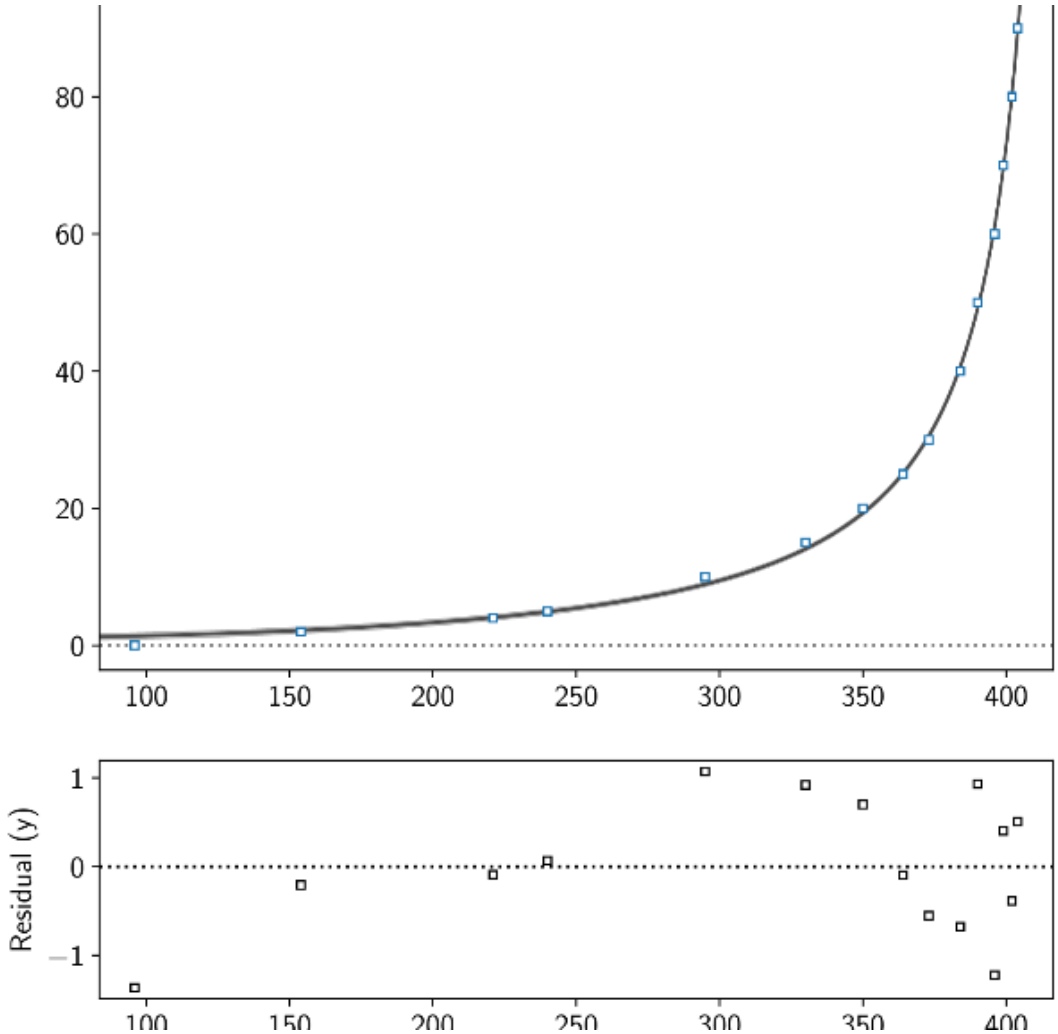
δx ☐

δy ☐

SSR = **8.11e+00**

Parameter	*Value*	*Uncertainty*
A	1.510e+06	4.873e+04
ω_o	-4.228e+02	5.670e-01
β	4.322e-06	0
C	-7.529e+00	6.094e-01

Jacobian not full rank, uncertainties may be zero



Utilizando este último modelo y combinándolo con la implementación anterior, se puede obtener una aproximación de la distancia de la pelota.

8.4 Cámara omnidireccional

8.4.1 Calibración de cámara omnidireccional

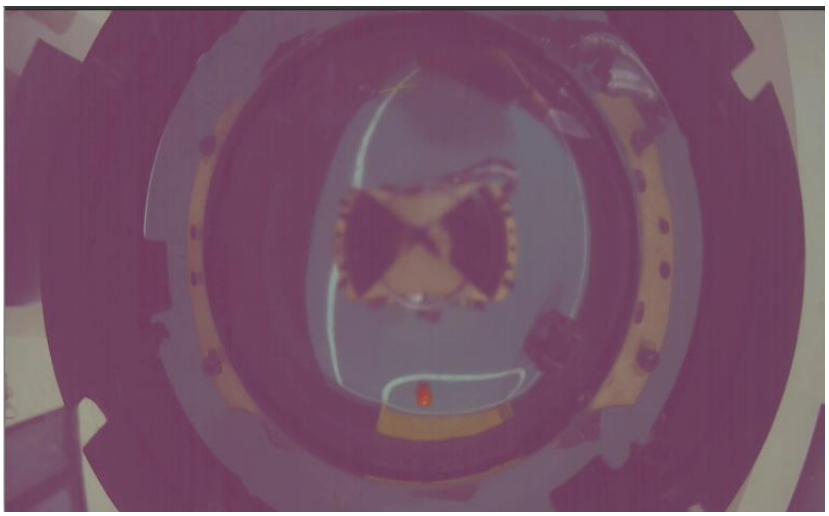
Para comenzar con la calibración, se extrajo la imagen cruda de la cámara, sin ningún tipo de filtro. Se alcanza a notar en la escena una portería amarilla, una pelota naranja, un robot

de AIZER y un humano. Al igual que en la cámara frontal (Véase sección 8.3.1), los colores que se muestran son muy pobres y no son cercanos a la realidad. Sin embargo, siguen siendo distinguibles, por lo que elevar el contraste dará una mayor diferencia en los colores, facilitando su detección.

Preprocesamiento:
- Ninguno

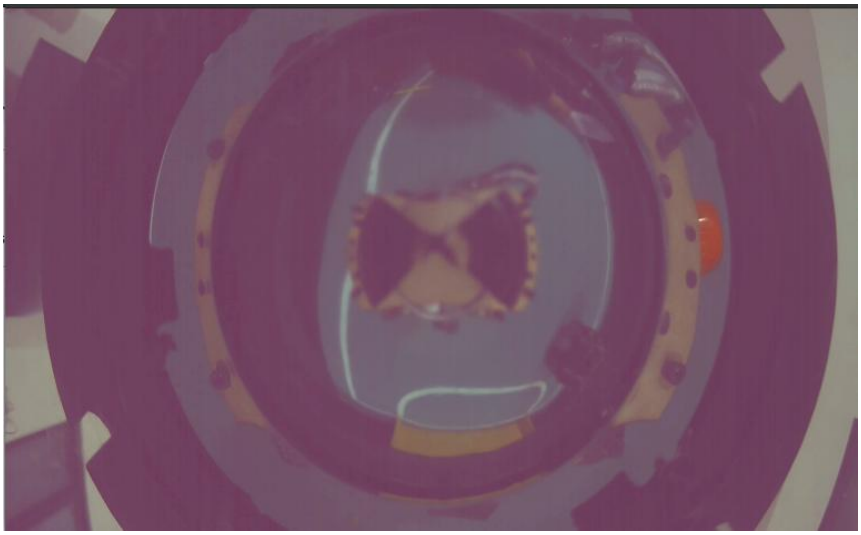
En la siguiente imagen se muestra la misma escena, utilizando una saturación de 2.
Este cambio muestra con mayor fuerza la diferencia entre los colores de la portería y la pelota, lo que es ideal para detectar ambos objetos utilizando un rango de HSV.

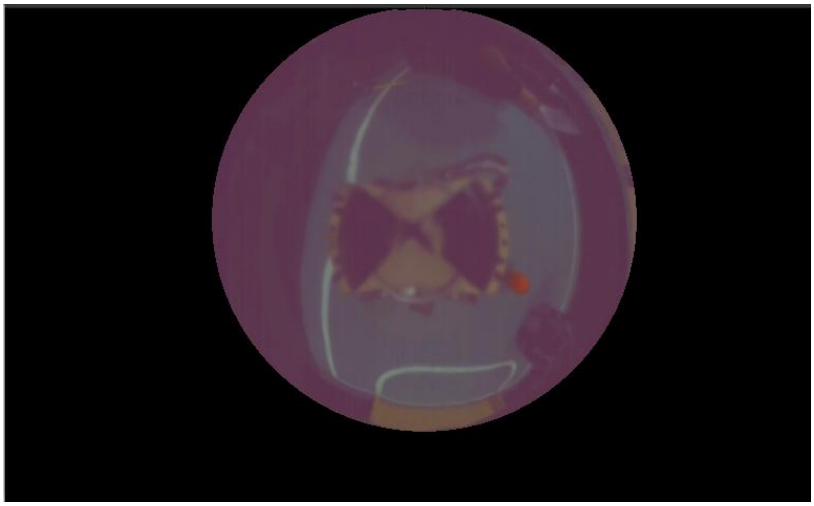
Preprocesamiento:
- Saturación: 2



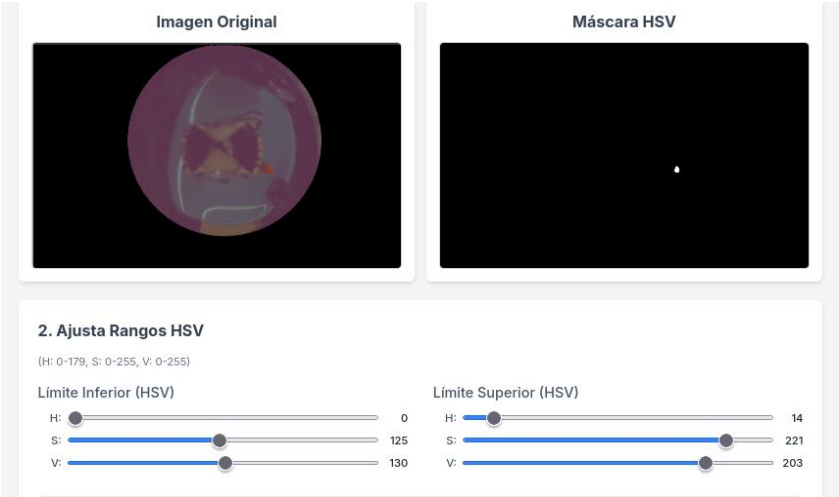
Un efecto interesante que se puede observar utilizando los espejos omnidireccionales, es que después alrededor de la imagen principal, hay un aro en donde se muestra el alrededor del robot de cerca. Esto puede ser ideal si la pelota está demasiado cerca al robot y se necesita encontrar. Como se puede ver en la imagen, el robot es capaz de ver la pelota naranja a pesar de que en el círculo principal no se alcance a ver la pelota.

Además de este efecto, la cámara es capaz de ver fuera de la cancha, viendo los extremos

que no están tapados por el espejo, esto puede ser una desventaja ya que encontrarse con algún color extraño puede confundir los objetos.	
Preprocesamiento: ● Saturación: 2	

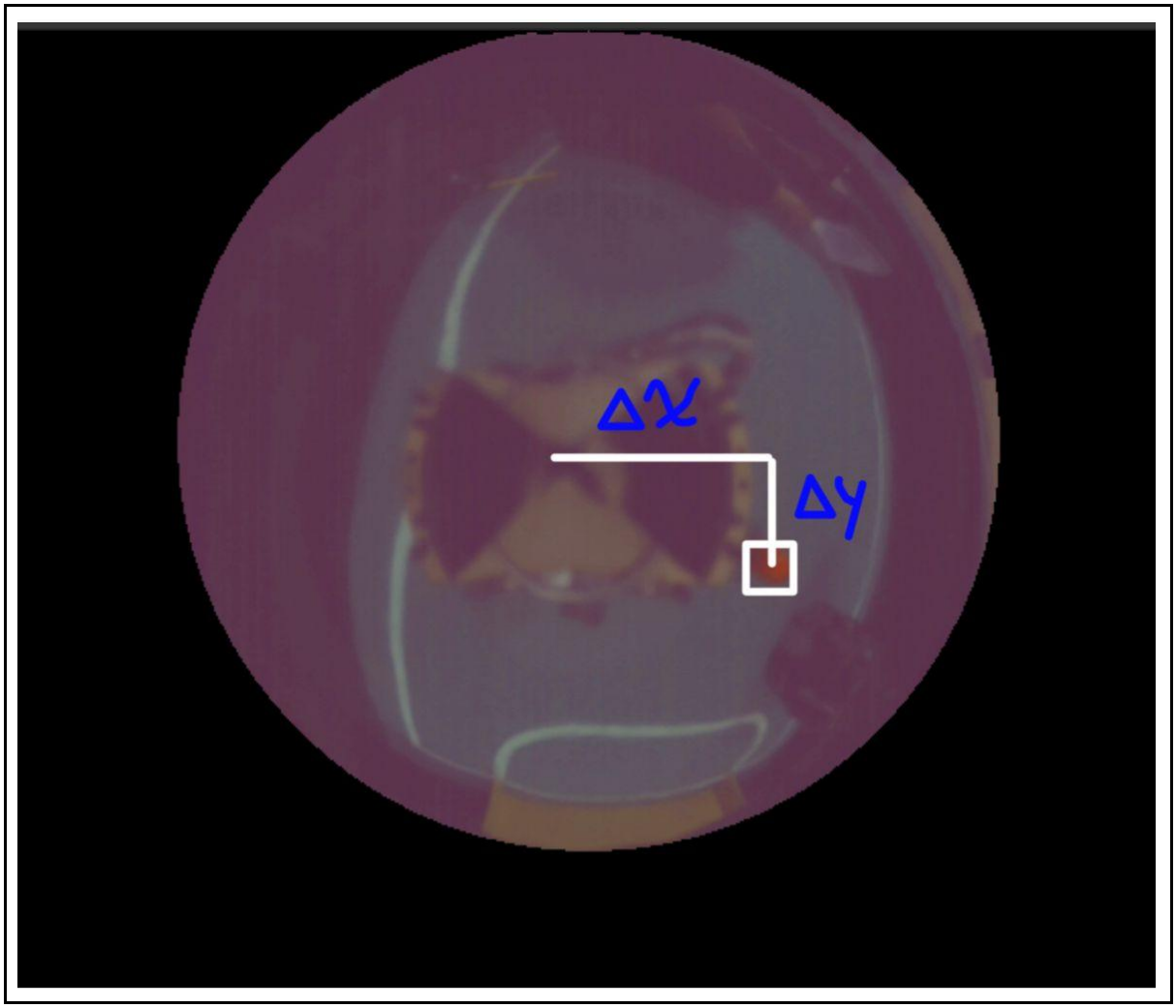
Se optó por dibujar un círculo negro, dejando únicamente la imagen principal, esto con el objetivo de simplificar la visión y tener un desarrollo más rápido.	
Preprocesamiento: ● Saturación: 2 ● Circulo negro ajustado manualmente	

Utilizando la imagen resultante, se utilizará un programa para obtener los valores HSV a través de entradas deslizantes que permiten la calibración. Después de prueba y error se llego a una máscara que muestra únciamente la pelota de color naranja.
Con esta información es posible utilizar el código presentado en la sección 8.2.

Preprocesamiento: ● Saturación: 2 ● Círculo negro ajustado manualmente Procesamiento: ● Lower bound: (0, 125, 130) ● Upper bound (14, 221, 203)	

8.4.2 Estimación para ángulo

Para la estimación de ángulo, se realiza de una manera muy similar a la cámara frontal, se plantean líneas imaginarias de la siguiente manera.



- $\Delta x = \left(ball_x + \frac{ball_{width}}{2}\right) - CAMERA\ ORIGIN\ X$
- $\Delta y = \left(ball_y + \frac{ball_{height}}{2}\right) - CAMERA\ ORIGIN\ X$

Donde CAMERA ORIGIN X, Y contienen las coordenadas del origen de la cámara, esto se calibra manualmente.

Para encontrar el ángulo θ se utilizará la función trigonométrica *arctan* de la siguiente manera:

$si\ x > 0$:

$\theta = arctan(\frac{\Delta y}{\Delta x})$

$si\ x < 0$:

y $y \geq 0$:

$\theta = arctan(\frac{\Delta y}{\Delta x}) + \pi$

y $y \leq 0$:

$\theta = arctan(\frac{\Delta y}{\Delta x}) - \pi$

$si\ x = 0$:

y $y > 0$:

$\theta = \frac{\pi}{2}$

y $y < 0$:

$\theta = -\frac{\pi}{2}$

y $y = 0$:

$\theta = 0$

9. Software para comunicación

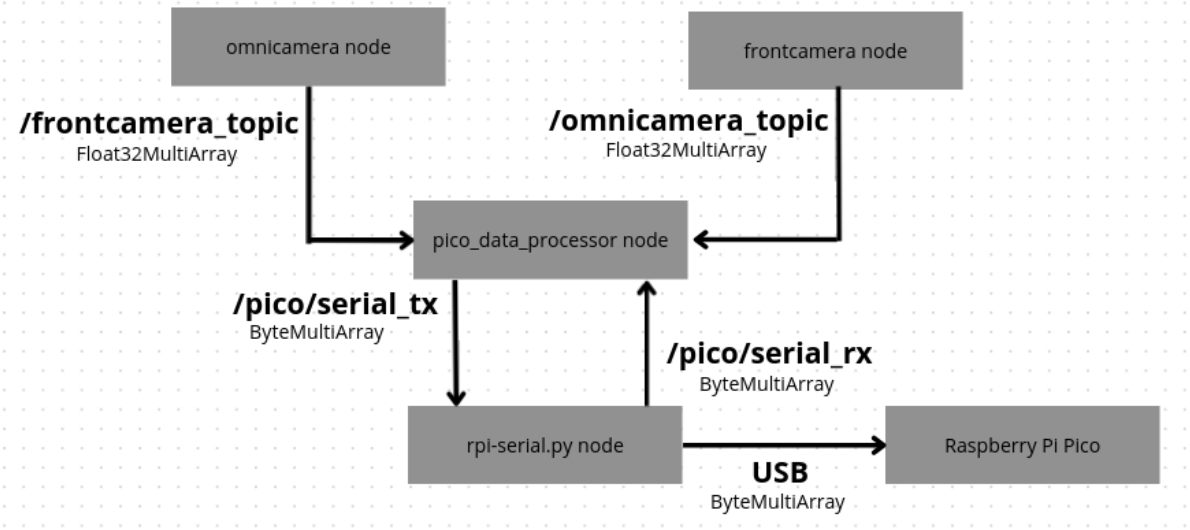
El robot está construido en una arquitectura modular, con cada sección del robot enfocados en lograr un objetivo específico. La comunicación entre estos módulos permite el funcionamiento correcto del robot. Se utiliza ROS Melodic para comunicación entre nodos de la Jetson Nano, y se utiliza un sistema personalizado para el intercambio de datos por USB entre la Jetson Nano y la Raspberry Pi Pico.

9.1 ROS Melodic entre nodos

Se eligió utilizar ROS Melodic ya que es la versión más común para Ubuntu 18.04, que es el sistema operativo de Jetpack 4.6.4, que utiliza el robot [35]. Se utilizan 4 modos de la siguiente manera:

- omnicamera: Analiza la imagen de la cámara omnidireccional, obtiene aproximaciones de ángulo. (Véase la sección 8.4).
- frontcamera: Analiza la imagen de la cámara frontal, obtiene aproximaciones del ángulo y la distancia. (Véase la sección 8.3).
- rpi-serial.py: Actúa como puente para la comunicación entre la Jetson Nano y la Raspberry Pi Pico.
- pico_data_processor: Analiza los datos recibidos por la rpi-serial.py, omnicamera y frontcamera, toma decisiones y envía los comandos de vuelta a rpi-serial.py

Se pueden entender los nodos de la siguiente manera:



9.2 Comunicación Jetson - Raspberry Pi Pico

La comunicación entre la Jetson Nano y la Raspberry Pi Pico es un reto que involucra sincronización, serialización y deserialización de datos. Se serializan datos clave en un objeto que se intercambia de manera uno y uno entre la Jetson Nano y la Raspberry Pi Pico.
El objeto que se intercambia se define como BinarySerializationData y se ve de la siguiente manera:

```
struct BinarySerializationData {
   int16_t compass_yaw;
   uint16_t robot_speed;
   int16_t robot_direction;
   int16_t robot_facing;
   bool robot_stop;
   uint16_t ldr_value;
   bool kicker_active;
};
```

Este objeto contiene distintas variables donde:

- compass_yaw: Es el ángulo al cual el robot está mirando, expresado en grados en un rango de -180 a 180. Este valor es modificado por la Raspberry Pi Pico e interpretado por la Jetson.

- robot_speed: Es la velocidad a la cual el robot se debe mover, se expresa en revoluciones por decisegundo, tiene un rango de 0 (es decir, no movimiento) hasta 167 (Véase sección 11.2.2). Este valor es modificado por la Jetson e interpretado por la Raspberry Pi Pico.

- robot_direction: Especifica la dirección en la cual el robot se debe mover, expresada en ángulos. Este valor es modificado por la Jetson e interpretado por la Raspberry Pi Pico.

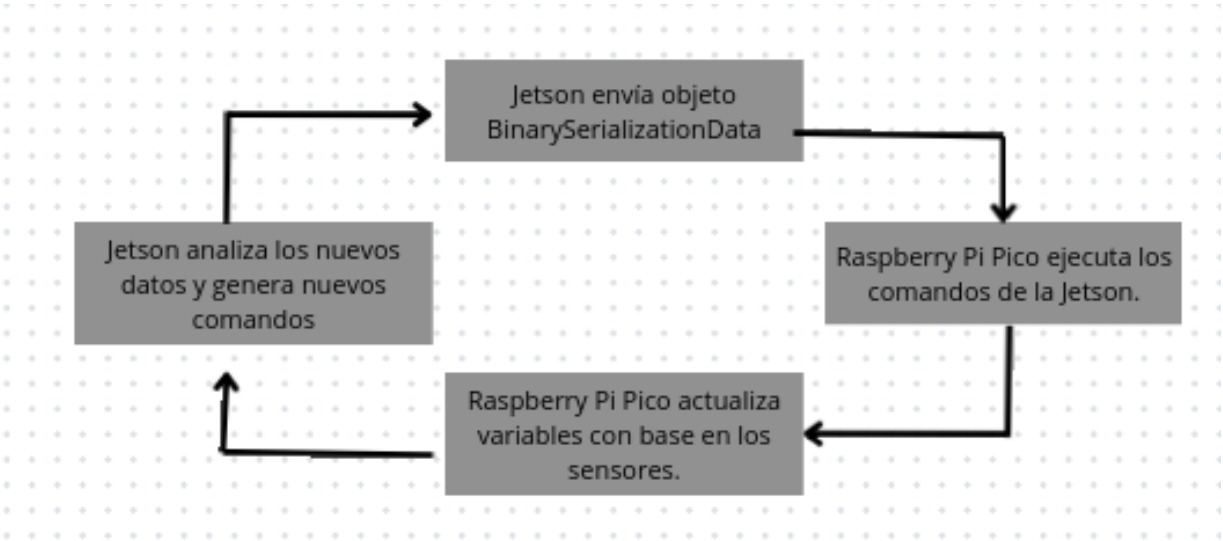
- robot_facing: Indica qué ángulo el robot debe ver cuando se mueva. Este valor es modificado por la Jetson e interpretado por la Raspberry Pi Pico.

- robot_stop: Un simulador de freno activo que contrarresta el movimiento de los motores para llevarlos a una parada. (Véase sección 7.4.3). Este valor es modificado por la Jetson e interpretado por la Raspberry Pi Pico.

- ldr_value: Contiene valores de un sensor de luz, su propósito es únicamente de calibración y no cumple una función específica en el programa. Este valor es modificado por la Raspberry Pi Pico.

- kicker_active: Cuando esta variable es cambiada a positivo, se activará el circuito del pateador. Este valor es modificado por la Jetson e interpretado por la Raspberry Pi Pico.

El sistema de intercambio del objeto BinarySerializationData se puede entender de la siguiente manera:



10. Software para implementación de estrategias

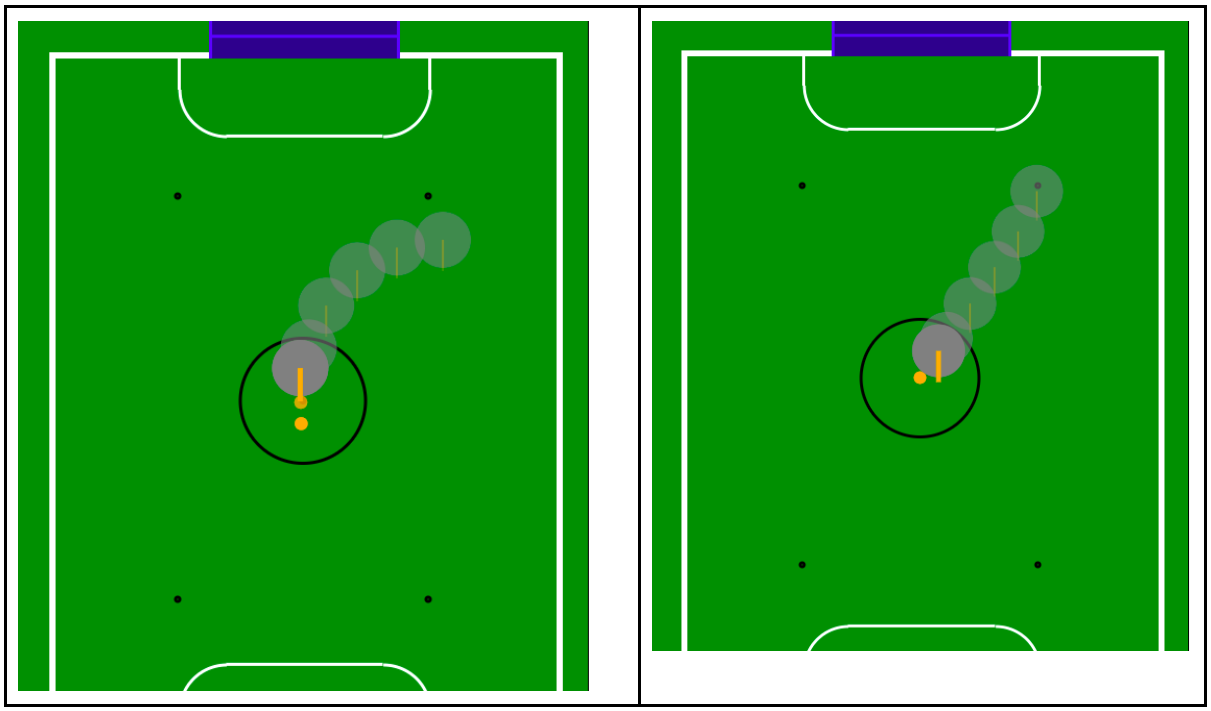
Las estrategias definen como el robot se controlará dependiendo de las circunstancias. La implementación de las estrategias se basa mayoritariamente en la Jetson Nano, en donde se tiene acceso a los datos de las cámaras y a los sensores.

10.1 Descripción de estrategia

La estrategia para la competencia del TMR 2025, se basa únicamente en lanzar la pelota hacia la portería contraria, esperando hacer suficientes intentos para que alguno sea gol. Para planes a futuro, véase la sección 13.

10.2 Uso de cámara frontal

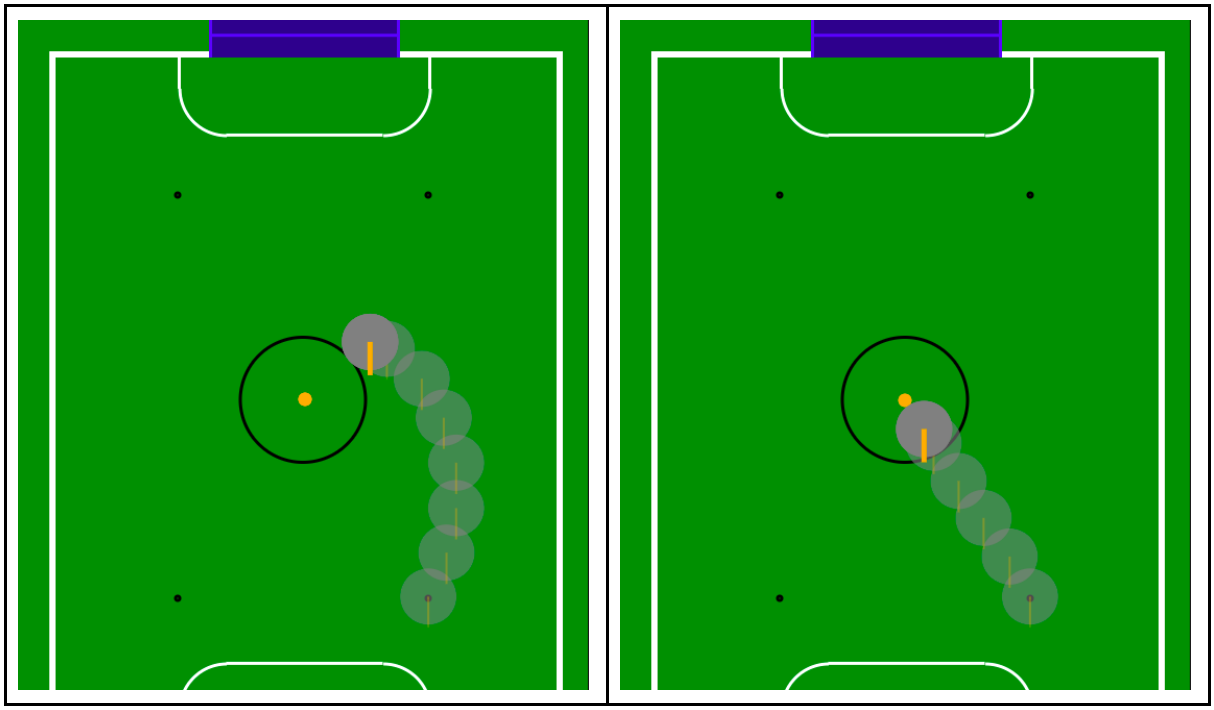
El uso de la cámara frontal tiene prioridad sobre la cámara omnidireccional ya que provee de una mayor precisión. El objetivo de la cámara frontal es posicionarse detrás de la pelota. Para esto se mueve al ángulo de la pelota, agregando un offset que permita posicionarse detrás de la pelota, en lugar de golpearla.



En la simulación de la izquierda, se puede ver el comportamiento del robot con este offset, es capaz de posicionarse detrás de la pelota y golpearla verticalmente, además que de esta manera la pelota nunca se pierde de vista.
En la simulación de la derecha sin embargo, el robot se mueve en una línea diagonal, hace que la pelota sea golpeada de manera diagonal evitando la portería, además que la pelota puede perderse de la cámara si la pelota está parcialmente en la visión.

10.3 Uso de cámara omnidireccional

El uso de la cámara omnidireccional es para obtener contexto, si la cámara frontal no es capaz de detectar la pelota, entonces la cámara omnidireccional buscará la pelota, y si existe, el robot se moverá hacia el ángulo de la pelota con un offset, de la misma manera que con la cámara frontal.



En la simulación de la izquierda, el movimiento tiene un offset que permite al robot hacer un movimiento de elipse, lo que permite posicionarse de frente a la pelota y pasarle la responsabilidad a la cámara frontal.
En la simulación de la derecha, el robot golpea directamente la pelota, botando la pelota hacia su propia portería y no logrando pasar la responsabilidad a la cámara frontal.

10.4 Uso de pateador

La responsabilidad del pateador recae en la Raspberry Pi Pico, evitando que este se active sin que esté cargado. Esto se logra colocando un tiempo de espera de 7.5 segundos antes de volver a permitir la activación del pateador.
El programa de la Jetson Nano integra el pateador en la estrategia, si este se encuentra a menos de 1.5 centímetros del robot, lo que asegura un golpe fuerte que pueda llevar la pelota de portería a portería.

10.5 Detección de líneas

La responsabilidad de detección de líneas recae totalmente en la Raspberry Pi Pico, tiene prioridad sobre cualquier instrucción de movimiento enviada por la Jetson Nano, el movimiento consiste en mover los motores a máxima velocidad en dirección contraria a la del movimiento del robot.

11. Especificaciones técnicas

Los dos robots de AIZER se construyeron utilizando el mismo diseño y las mismas especificaciones, durante el armado es posible que hayan existido pequeñas diferencias. Sin embargo, estas diferencias son poco significativas y no representan un motivo de preocupación importante.

11.1 Dimensiones físicas

Robot 1:

- Diámetro: 179.5 mm
- Altura: 176 mm
- Peso 1847 g

Robot 2:

- Diámetro: 179.2mm
- Altura: 178mm
- Peso: 1866g

La explicación para el cambio de altura entre los robots se explica a través de el tubo de acrílico, que fue cortado de manera manual y un error de un par de milímetros es esperado
La explicación para el cambio de diámetro entre los robots tiene motivo en la cortadora láser de MDF, ya que esta vibraba al momento de cortar haciendo que los cortes no fueran precisos por fracciones de milímetros.

11.2 Limitaciones de hardware

El hardware utilizado para los robots de AIZER, cuentan con sus propios límites y thresholds para salvaguardar la integridad del hardware, evitar sobrecalentamiento, sobrecorriente, polarización inversa, y más posibles alteraciones que pueden llegar a dañar las piezas.

11.2.1 Controladores de motores

Los controladores de motores DRV8256E tienen un sensor anti-sobrecalentamiento, lo que es ideal para mantener el tiempo de vida de los controladores [20]. Este límite, sin embargo, es posible alcanzarlo si se oscila de manera muy rápida con cambios de direcciones entre positivo y negativo. Es posible también alcanzar este límite si el motor se cala, es decir que no se mueve, incluso si se le está suministrando energía.

11.2.2 Velocidad máxima

La velocidad máxima de los motores 25Dx63L mm HP 12V de pololu es de 16.66 revoluciones por segundo [17].

11.2.3 Rango y precisión de sensores

El sensor de luz (LDR) opera en un rango de 0V a 3.3V, la señal analógica es convertida en una señal digital a través del ADC MCP3008 de 10 bits, esto significa que la señal digital recibida del ADC es de 0 a 1023.
La precisión del IMU BNO055 es de $\pm 2°$ [6]. La precisión del sensor depende también de las condiciones del ambiente.

12. Discusión

En la sección 2.2 se establecen objetivos con relación al resultado esperado en el desarrollo de los robots de AIZER, durante este documento se describió el proceso de implementación para lograr estos objetivos como el control omnidireccional y un sistema de visión dual.
En gran medida, se lograron los objetivos, se logró un control estable del sistema omnidireccional y un sistema de visión dual que permiten ubicar fiablemente la pelota.

El sistema de locomoción obtuvo un resultado sobresaliente, tiene un movimiento predecible y nos permite ejecutar la gran mayoría de nuestras estrategias, la única dificultad del sistema de locomoción es el cambio brusco de direcciones, el robot no es capaz de mantener su posición mirando hacia enfrente después de hacer un cambio rápido, aunque se equilibra en unos cuantos segundos.

La cámara logra un buen resultado al detectar la pelota, pero tiene dificultad para distinguir entre la portería azul y las paredes, ya que no se logró hacer una buena calibración de los colores, especialmente los colores obscuros o azulados, no fuimos capaces de encontrar la raíz de ese problema y, por ende, adaptamos nuestra estrategia a no utilizar las porterías.

La comunicación entre los nodos de ROS funciona como se esperaba, son capaces de comunicarse entre sí de una manera muy rápida y precisa. Las dificultades vienen al comunicar la Jetson con la Raspberry Pi Pico, tiene una poca cantidad de intercambio de mensajes por segundo, lo que puede hacer que el robot no reciba datos actualizados de los sensores y tome decisiones atrasadas.

El chasis al momento de su ajuste resultó como lo planeado, pero al haber creado las medidas demasiado exactas tuvimos problemas ya que el chasis terminó resultando en un diámetro exacto de 180 mm, lo cual al momento de pasarlo por el tubo de medición tuvo cierta dificultad. Para resolver el problema con el tubo de medición se tuvo que mover los motores unos milímetros más hacia el interior y realizar ciertos cortes en los bordes de la mdf de los extremos de los motores, lo cual resultó en pasar exitosamente el tubo de medición.

Al momento de hacer la modificación de mover los motores no tuvimos en cuenta cómo afectaría esto a los componentes cercanos a estos, por lo cual hubo un error al momento de intentar cerrar el chasis, se tuvo que hacer uso del dremel en los soportes del motor para crear espacio suficiente para cerrar el chasis.

En la placa de control principal, los LM317 tenían el objetivo de reducir un voltaje de 5 voltios a 3.3 voltios para los componentes que ocuparan este voltaje, y el circuito si funcionaba en pruebas de protoboard, pero debido a que en la placa no funcionó, tuvimos que instalar otro regulador de voltaje externo para dar los 3.3 voltios, dejando sin uso esos transistores en la placa. También otra observación es que en esta misma placa el SDA y SCL de la brújula BNO 055 están volteados, ya que se consideró un modelo distinto (uno donde la BNO es azul en vez de negra) donde aun teniendo el mismo nombre, cambia el lugar de estos pines.

13. Conclusión y trabajo futuro

El trabajo descrito en este documento cumplió con la gran mayoría de nuestras expectativas, fuimos capaces de desarrollar un control sobresaliente, un diseño robusto, una electrónica fiable y una comunicación suficiente para poder implementar estrategias básicas.
En el caso que podamos participar en la competencia internacional de Robocup Junior Soccer Open, investigaremos a fondo los problemas de la visión y de la comunicación entre la Raspberry Pi Pico y la Jetson Nano.
Se podría decir que el objetivo de una arquitectura de procesamiento distribuido comienza a dar resultados positivos, sin embargo, nos sigue faltando desarrollo para pulir todos los detalles y poder crear estrategias mucho más efectivas.

14. Agradecimientos

Al momento de terminar este proyecto no nos quedó más que reflexionar sobre todo lo que hemos aprendido en el transcurso de estos últimos años. Como el Torneo Mexicano de Robótica fue el motivo del nacimiento de un grupo que se guiaría con una motivación única y especial de dejar huella en el mundo de la robótica, estamos próximos a nuestra graduación donde cada integrante del equipo tomará su propio camino, pero el hecho que logramos trabajar juntos para participar e innovar en el TMR es algo que se nos quedará en la memoria. Este Technical Description Paper para nosotros tiene un significado aún mucho más profundo, es nuestra manera de ver y mostrar nuestro progreso, lo que años de trabajo en equipo y unas ganas inmensas de innovar en la tecnología dieron de resultado. No nos queda más que agradecer al TMR, por ser el inicio de nuestra motivación que guiará el camino de nuestras carreras en robótica y nuestro desarrollo personal.

De igual manera, solo nos queda agradecer de igual manera a cada miembro del equipo, Iker Cossio, Alejandro Salinas y Fátima Chantal. Juntos logramos cosas que parecían inimaginables, guiándonos mutuamente para seguir avanzando como equipo. Aunque los torneos representan una excelente oportunidad de aprendizaje y motivación, suelen estar cargados de estrés. Sin embargo, al enfocarse no solo en el desarrollo del robot, sino también en la fomentación de una convivencia sana y una buena organización, logramos transformar ese ambiente que usualmente está lleno de tensión en una experiencia positiva y un recuerdo memorable para todos. También hay que considerar que los integrantes del equipo no son solo los que construyen el robot, o los que están en la cancha, sino que son todos los que aportan al equipo. Queremos agradecer profundamente a nuestro equipo de Marketing, Roberta G. y Andrea L., que cuando nuestra moral estaba baja nos enseñaron lo mucho que podemos llegar a inspirar a otros, gracias a su desempeño logramos establecer una presencia en las redes sociales contando con una audiencia alta que está interesada en la robótica tanto como nosotros. Su apoyo no solo nos ayudó a promover la robótica a nivel mundial, sino que también nos mostró que hay personas allá afuera que comparten nuestro entusiasmo y celebran nuestros avances. Gracias a ellas, seguimos adelante con un profundo agradecimiento y una nueva motivación.

Reconocemos que nuestro progreso no habría sido posible sin nuestros patrocinadores, MEHD. Desde el inicio confiaron en nuestro potencial y en nuestra visión para este proyecto, no solo brindándonos apoyo financiero, sino también su confianza. Gracias a ellos, hemos

podido alcanzar nuestras metas y continuar avanzando. Su contribución ha sido fundamental para que nuestro equipo crezca y se fortalezca. Estamos profunda y eternamente agradecidos por su apoyo.

15. Referencias

[1] Robocup Junior, “Welcome to RobocupJunior”, Robocup Junior. [Online]. Available: https://junior.robocup.org/. [Accessed: Apr. 30, 2025].
[2] Femex Robótica, “RobocupJunior Soccer”, TMR 2025. [Online]. Available: https://femexrobotica.org/tmr2025/robocup-junior-soccer/. [Accessed: Apr. 30, 2025]
[3] robocup-junior, “awesome-rcj–soccer”. [Online]. Available https://github.com/robocup-junior/awesome-rcj-soccer. [Accessed: Apr. 30, 2025]
[4] AIZER “Technical description paper RCJ Soccer Open 2024”, Grupo estudiantil AIZER CEGL, 2024. [Online]. Available: https://github.com/AIZER-egl/RCJ_Soccer_Open_2024/blob/main/assets/Technical%20description%20paper.pdf. [Accessed: Apr. 30, 2025]
[5] PrepaTec Voltec Robotics “VOLTEC Robotics”, PrepaTec Eugenio Garza Lagüera, 2023. [Online]. Available: https://drive.google.com/file/d/1S7_5_M_BAXc_r8C6eipkUY8qNfNcUUKU/view. [Accessed: Apr. 30, 2025]
[6] Bosch, “BNO055 Intelligent 9-axis absolute orientation sensor”, BOSCH. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf. [Accessed: Apr. 30, 2025]
[7] Raspberry Pi, “Pico-series Microcontrollers”, Raspberry Pi documentation. [Online]. Available: https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html. [Accessed: May. 2, 2025]
[8] Arduino, “UNO R3,” Arduino Documentation. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3/#tech-specs. [Accessed: May. 2, 2025].
[9] Shielden, “Are LiFePO4 Batteries Safe?”, Shielden Blog, May. 28, 2024. [Online]. Available: https://es.shieldenchannel.com/blogs/battery/are-lifepo4-batteries-safe . [Accessed: May. 3, 2025].
[10]. Control Automático Educación, “Control en Cascada”, Control Retroalimentado. [Online]. Available: https://controlautomaticoeducacion.com/control-realimentado/control-en-cascada/. [Accessed: May 3, 2025]
[11]. Ni, “Explicación sobre el controlador PID y la teoría”, Ni. [Online]. Available: https://www.ni.com/es/shop/labview/pid-theory-explained.html. [Accessed: May 3, 2025]
[12]. Pololu, “DRV8256E Single Brushed DC Motor Driver Carrier”, Brushed DC Motor Drivers. [Online]. Available: https://www.pololu.com/product/4038. [Accessed: May 3, 2025]
[13]. Ayala, F. “¿QUÉ ES UN MOSFET?”, UNIT Electronics. [Online]. Available: https://uelectronics.com/transistores-mosfet. [Accessed: May 4, 2025].
[14]. Hayes, A., Parvej, R., Zhi Vee, H., Nelson, A., Khim, J., “Discrete Mathematics”, Brilliant. [Online]. Available: https://brilliant.org/wiki/discrete-mathematics/. [Accessed: May 4, 2025]
[15]. Microchip, “MCP3004/3008”, Microchip. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP3004-MCP3008-Data-Sheet-DS20001295.pdf. [Accessed: May 4, 2025].

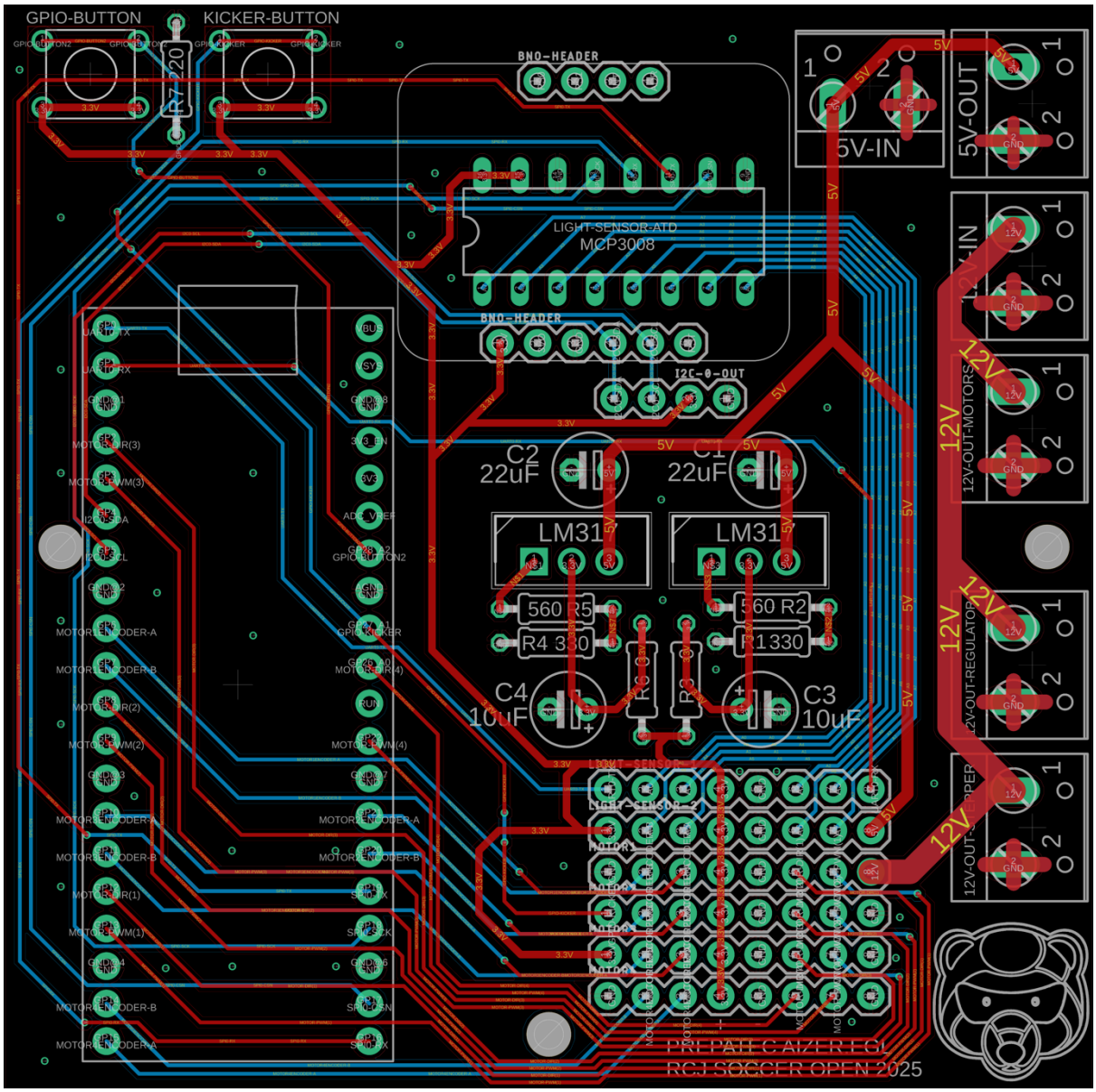
[16]. Texas Instruments, “DRV8256E/P H-Bridge Motor Driver With Integrated Current Sense and Smart Tune Technology”, Texas Instruments. [Online]. Available: https://www.ti.com/lit/ds/symlink/drv8256.pdf. [Accessed: May 4, 2025].
[17]. Pololu, “9.7:1 Metal Gearmotor 25Dx63L mm HP 12V with 48 CPR Encoder”, Pololu. [Online]. Available: https://www.pololu.com/product/4842. [Accessed: May 4, 2025].
[18]. STMICROELECTRONICS, “L298”, alldatasheet.es. [Online]. Available: https://www.alldatasheet.es/datasheet-pdf/download/22440/STMICROELECTRONICS/L298N.html. [Accessed: May 4, 2025].
[19] Electrical Classroom, “Difference between BJT and MOSFET,” Electrical Classroom. [Online]. Available: https://www.electricalclassroom.com/difference-between-bjt-and-mosfet/. [Accessed: May. 4, 2025].
[20] Pololu, “DRV8256E Single Brushed DC Motor Driver Carrier,” Pololu, [Online]. Available: https://www.pololu.com/product/4038. [Accessed: May 4, 2025].
[21] Grobotronics, “Solenoid 12V JF-0286B”, Grobotronics. [Online]. Available: https://grobotronics.com/solenoid-12v-jf-0826b.html?sl=en. [Accesed: May 4,2025].
[22] Ingenierizando, “Fotorresistencia o LDR,” Ingenierizando, [Online]. Available: https://www.ingenierizando.com/electronica/fotorresistencia-o-ldr/. [Accessed: May 4, 2025].
[23] Microchip Technology Inc., MCP3004/MCP3008 2.7V 10-Bit A/D Converters with SPI Interface, DS20001295E, 2022. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/MSLD/ProductDocuments/DataSheets/MCP3004-MCP3008-Data-Sheet-DS20001295.pdf. [Accessed: May 4, 2025].
[24] Arducam Wiki, “12MP IMX708”, Arducam Wiki, [Online]. Available: https://docs-arducam-com.translate.goog/Nvidia-Jetson-Camera/Native-Camera/imx708. [Accessed: May 4, 2025].
[25] Borg, M., “Chapter 2- Omnidirectional Vision”, Github, [Online]. Available: https://mark-borg.github.io/assets/omniapp-chapter2.pdf. [Accesed: May 4, 2025].
[26]. Ultralytics, “Explorar Ultralytics YOLOv8”, Ultralytics YOLO Docs. [Online]. Available: https://docs.ultralytics.com/es/models/yolov8/. [Accessed: May, 5. 2025].
[27] BOSCH, “Smart Sensor: BNO055”, Bosch Sensortec. [Online]. Available: https://www.bosch-sensortec.com/products/smart-sensor-systems/bno055/ [Accessed: May 5, 2025].
[28] ENCODER, “¿Que es un encoder?”, Encoder Products Company. [Online]. Available: https://www.encoder.com/article-que-es-un-encoder. [Accessed: May 5, 2025].
[29] La electrónica. “Fotorresistencia”, La electrónica, 2021. [Online] Available: https://laelectronica.com.gt/extras/que-es-un-ldr-y-como-funciona. [Accessed: May 5, 2025].
[30] Matan, “¿Qué es una placa de circuito impreso (PCB)?”, Electricity and Magnetism. [Online] Available: https://www.electricity-magnetism.org/es/que-es-un-pcb-multicapa/. [Accessed: May 4, 2025]
[31]. Ulbrinox, “¿Qué es el cromado?”, Ulbrinox. [Online]. Available: https://www.ulbrinox.com.mx/blog/que-es-el-cromado. [Accessed: May 6, 2025].
[32]. Lead, “Conceptos básicos del pulido de aluminio: cómo pulir aluminio”, Leadrp. [Online]. Available: https://leadrp.net/es/blog/aluminum-polishing-basics-how-to-polish-aluminum/. [Accessed: May 6, 2025].
[33]. OpenCV, “Structural Analysis and Shape Descriptors”, OpenCV. [Online]. Available: https://docs.opencv.org/4.1.1/d3/dc0/group_imgproc_shape.html. [Accessed: May, 6. 2025].
[34] Seeed Studio, “NVIDIA Jetson Nano Developer Kit - B01,” Seeed Studio, [Online]. Available: https://www.seeedstudio.com/NVIDIA-Jetson-Nano-Development-Kit-B01-p-4437.html. [Accessed: May 6, 2025].

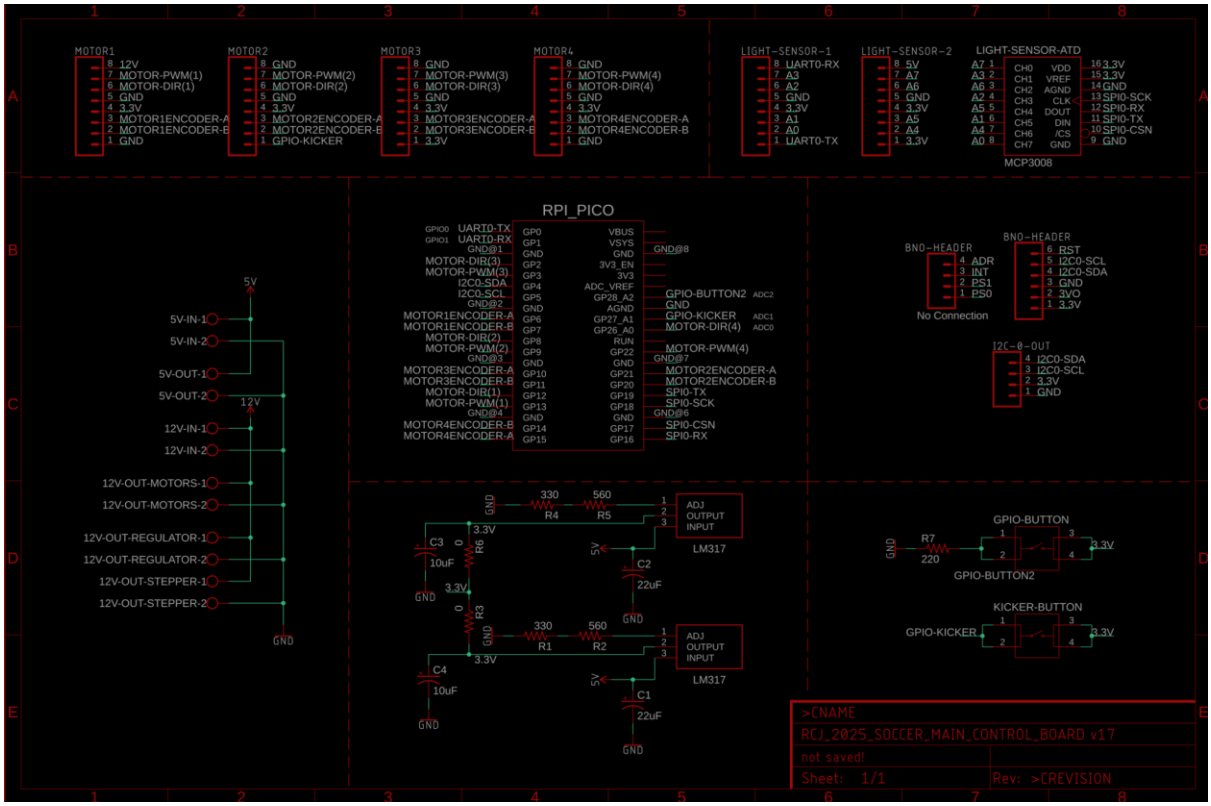
[35]. Nvidia, “JetPack SDK 4.6.4”, Nvidia. [Online]. Available: https://developer.nvidia.com/jetpack-sdk-464. [Accessed: May 6, 2025].
[36] Onshape, “why Onshape?”, Onshape. [Online]. Available: https://www.onshape.com/en/why-onshape [Accessed: May 6, 2025].

16. Apéndice

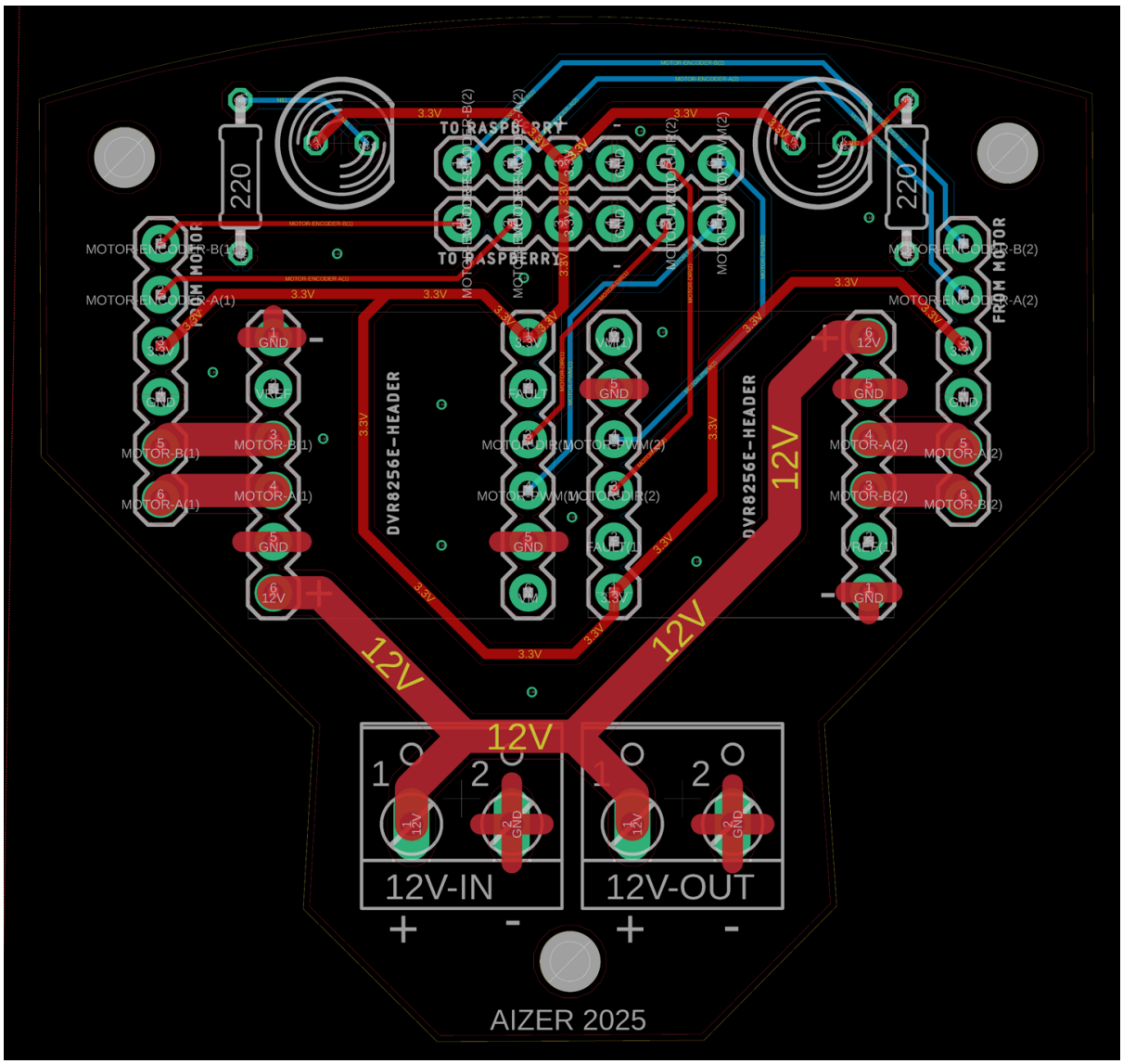
16.1 Esquemas electrónicos detallados

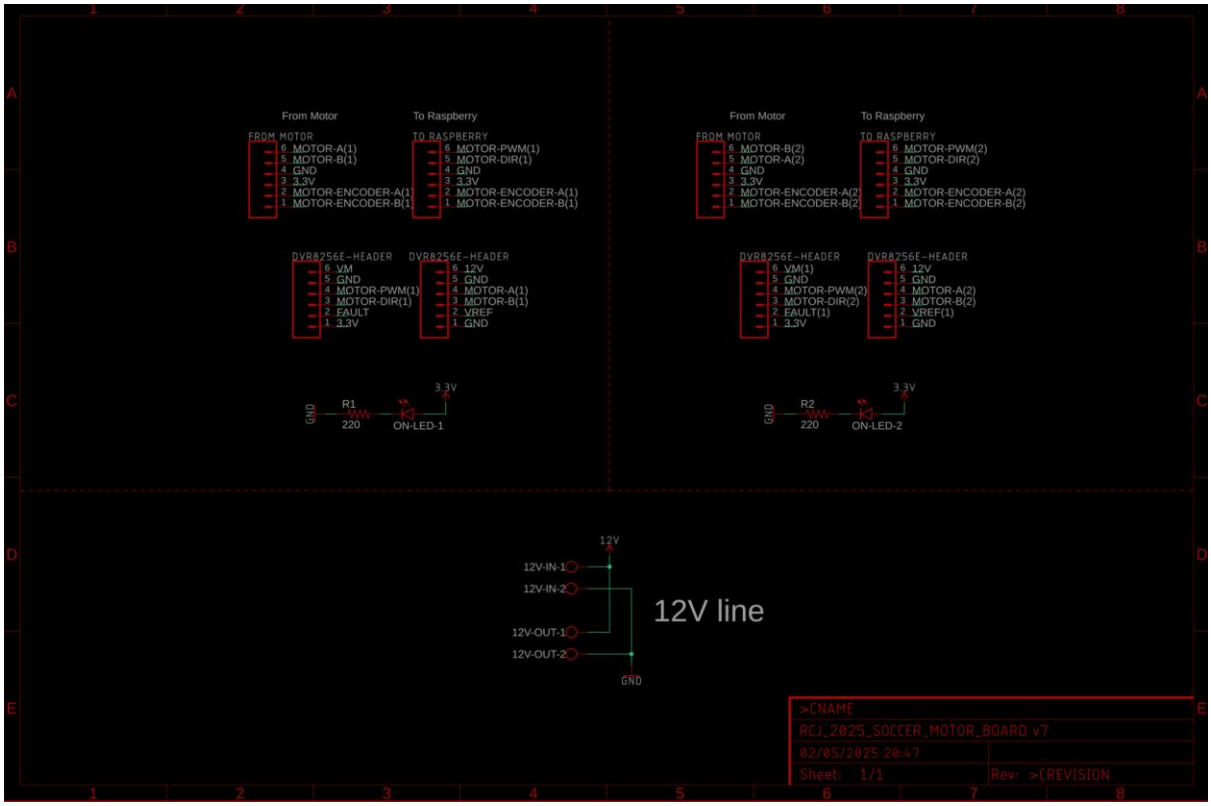
16.1.1 Placa de control



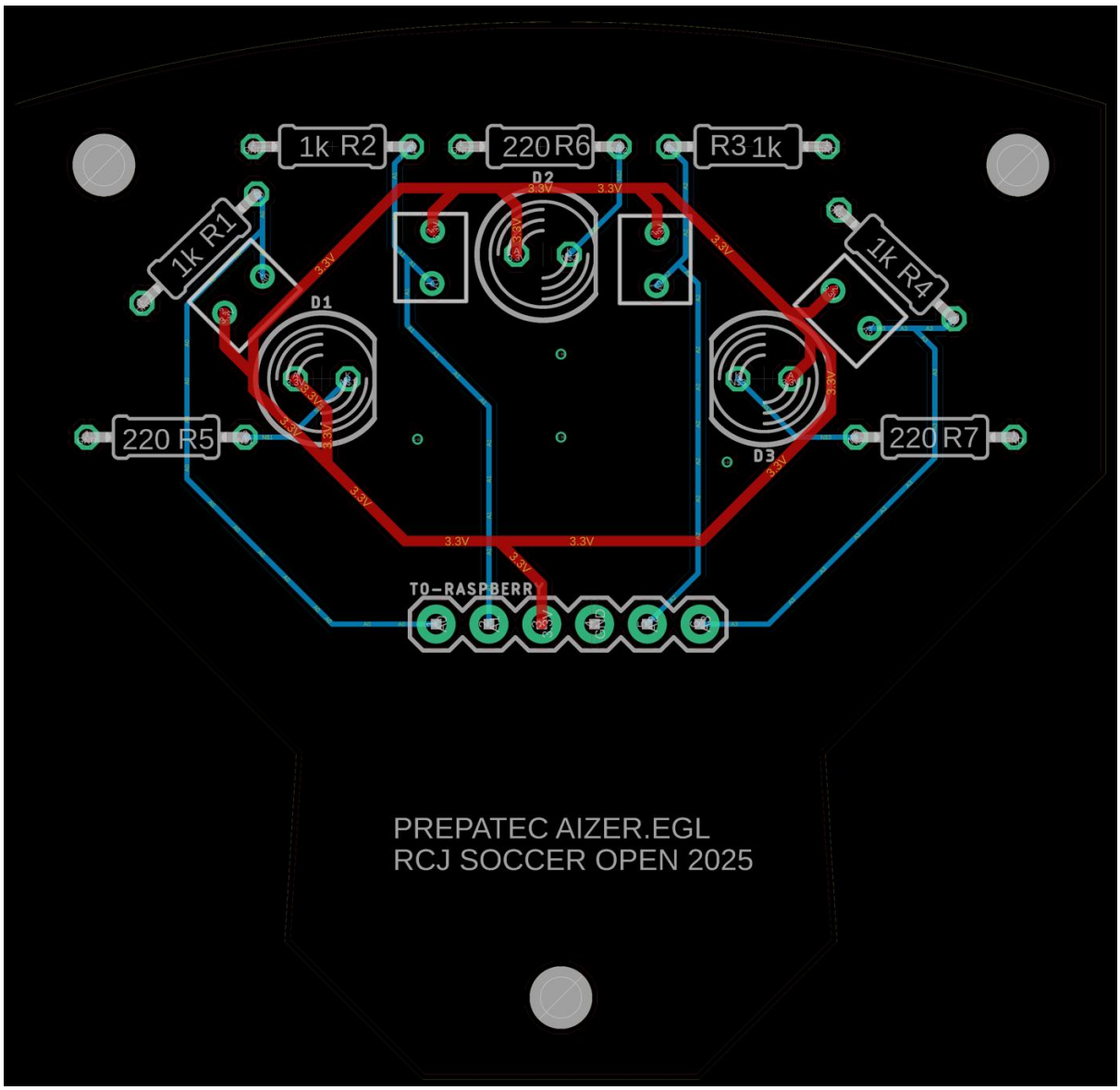

MOTOR1
8 12V
7 MOTOR-PWM(1)
6 MOTOR-DIR(1)
5 GND
4 3.3V
3 MOTOR1ENCODER-A
2 MOTOR1ENCODER-B
1 GND
MOTOR2
8 GND
7 MOTOR-PWM(2)
6 MOTOR-DIR(2)
5 GND
4 3.3V
3 MOTOR2ENCODER-A
2 MOTOR2ENCODER-B
1 GPIO-KICKER
MOTOR3
8 GND
7 MOTOR-PWM(3)
6 MOTOR-DIR(3)
5 GND
4 3.3V
3 MOTOR3ENCODER-A
2 MOTOR3ENCODER-B
1 3.3V
MOTOR4
8 GND
7 MOTOR-PWM(4)
6 MOTOR-DIR(4)
5 GND
4 3.3V
3 MOTOR4ENCODER-A
2 MOTOR4ENCODER-B
1 GND
LIGHT-SENSOR-1
8 UART0-RX
7 A3
6 A2
5 GND
4 3.3V
3 A1
2 A0
1 UART0-TX
LIGHT-SENSOR-2
8 5V
7 A7
6 A6
5 GND
4 3.3V
3 A5
2 A4
1 3.3V
LIGHT-SENSOR-ATD
MCP3008
RPI_PICO
GPIO0 UART0-TX
GPIO1 UART0-RX
GND@1
MOTOR-DIR(3)
MOTOR-PWM(3)
I2C0-SDA
I2C0-SCL
GND@2
MOTOR1ENCODER-A
MOTOR1ENCODER-B
MOTOR-DIR(2)
MOTOR-PWM(2)
GND@3
MOTOR3ENCODER-A
MOTOR3ENCODER-B
MOTOR-DIR(1)
MOTOR-PWM(1)
GND@4
MOTOR4ENCODER-B
MOTOR4ENCODER-A
GND@8
GPIO-BUTTON2
GND
GPIO-KICKER
MOTOR-DIR(4)
MOTOR-PWM(4)
GND@7
MOTOR2ENCODER-A
MOTOR2ENCODER-B
SPI0-TX
SPI0-SCK
GND@6
SPI0-CSN
SPI0-RX
BNO-HEADER
No Connection
BNO-HEADER
I2C-0-OUT
5V
5V-IN-1
5V-IN-2
5V-OUT-1
5V-OUT-2
12V
12V-IN-1
12V-IN-2
12V-OUT-MOTORS-1
12V-OUT-MOTORS-2
12V-OUT-REGULATOR-1
12V-OUT-REGULATOR-2
12V-OUT-STEPPER-1
12V-OUT-STEPPER-2
GND
330
560
R4
R5
ADJ
OUTPUT
INPUT
LM317
C3
10uF
R6
C2
22uF
R3
R1
R2
C4
C1
GPIO-BUTTON
R7
220
GPIO-BUTTON2
KICKER-BUTTON
GPIO-KICKER
>CNAME
RCJ_2025_SOCCER_MAIN_CONTROL_BOARD v17
not saved!
Sheet: 1/1
Rev: >CREVISION

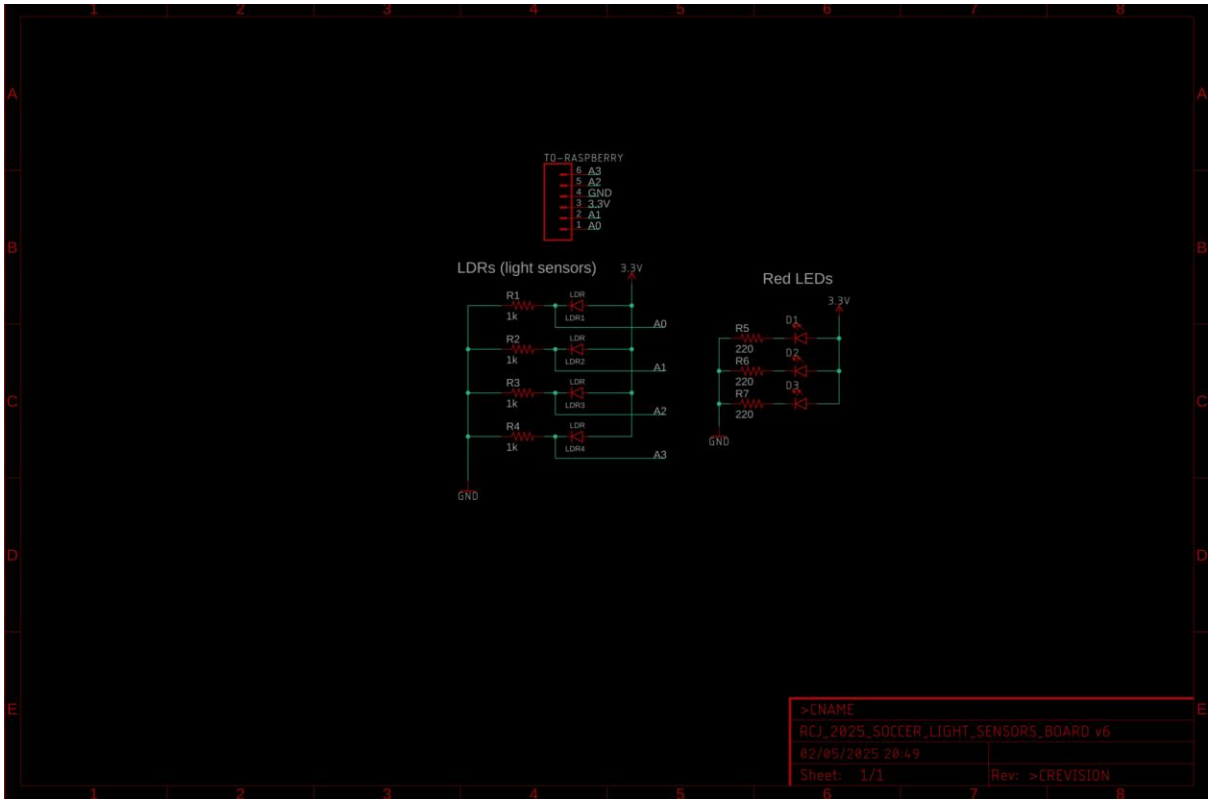
16.1.2 Placa de motores



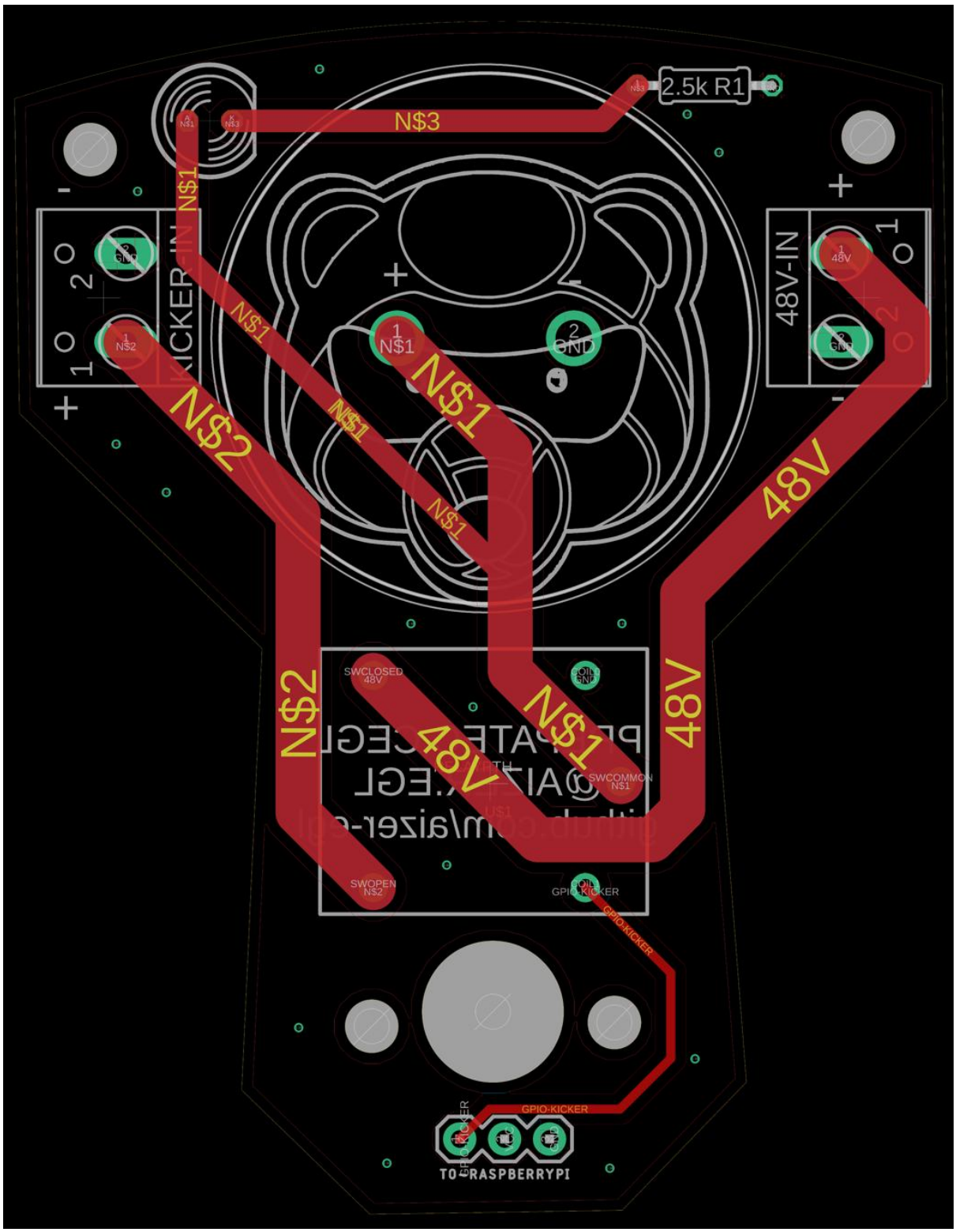

From Motor
To Raspberry
FROM MOTOR
6 MOTOR-A(1)
5 MOTOR-B(1)
4 GND
3 3.3V
2 MOTOR-ENCODER-A(1)
1 MOTOR-ENCODER-B(1)
TO RASPBERRY
6 MOTOR-PWM(1)
5 MOTOR-DIR(1)
4 GND
3 3.3V
2 MOTOR-ENCODER-A(1)
1 MOTOR-ENCODER-B(1)
From Motor
To Raspberry
FROM MOTOR
6 MOTOR-B(2)
5 MOTOR-A(2)
4 GND
3 3.3V
2 MOTOR-ENCODER-A(2)
1 MOTOR-ENCODER-B(2)
TO RASPBERRY
6 MOTOR-PWM(2)
5 MOTOR-DIR(2)
4 GND
3 3.3V
2 MOTOR-ENCODER-A(2)
1 MOTOR-ENCODER-B(2)
DVR8256E-HEADER
6 VM
5 GND
4 MOTOR-PWM(1)
3 MOTOR-DIR(1)
2 FAULT
1 3.3V
DVR8256E-HEADER
6 12V
5 GND
4 MOTOR-A(1)
3 MOTOR-B(1)
2 VREF
1 GND
DVR8256E-HEADER
6 VM(1)
5 GND
4 MOTOR-PWM(2)
3 MOTOR-DIR(2)
2 FAULT(1)
1 3.3V
DVR8256E-HEADER
6 12V
5 GND
4 MOTOR-A(2)
3 MOTOR-B(2)
2 VREF(1)
1 GND
GND
R1
220
3.3V
ON-LED-1
GND
R2
220
3.3V
ON-LED-2
12V
12V-IN-1
12V-IN-2
12V-OUT-1
12V-OUT-2
GND
12V line
>CNAME
RCJ_2025_SOCCER_MOTOR_BOARD v7
02/05/2025 20:47
Sheet: 1/1
Rev: >CREVISION

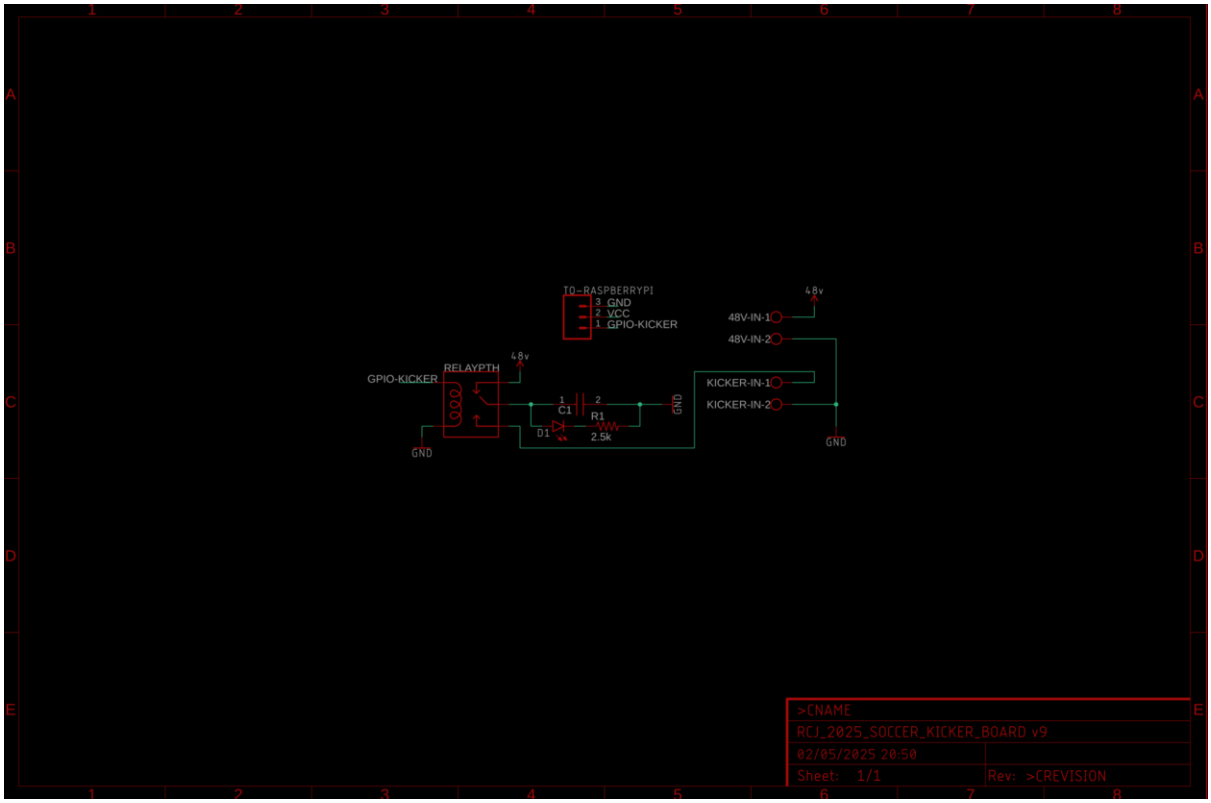
16.1.3 Placa de sensores de luz



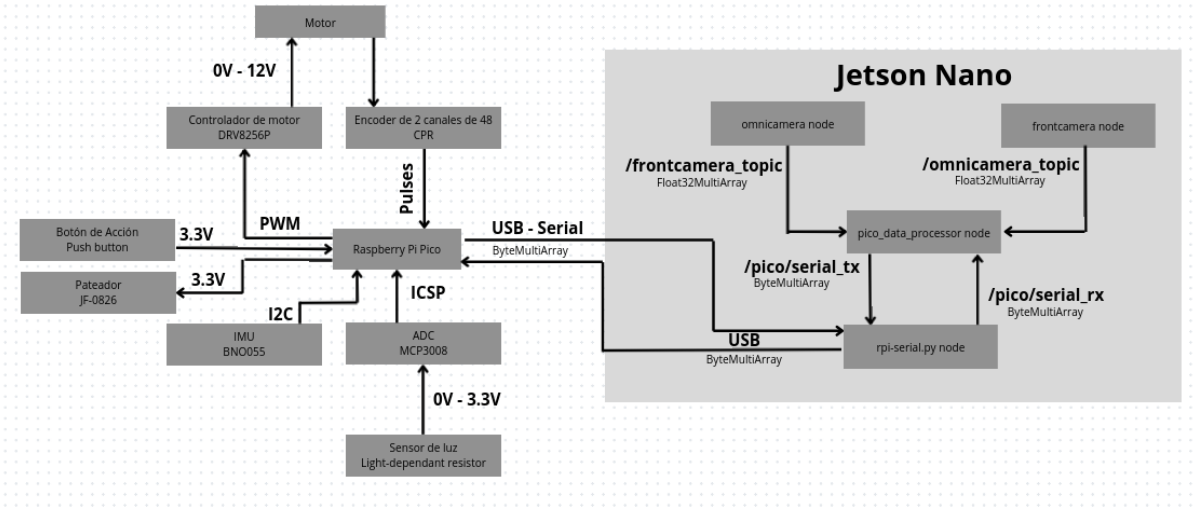

TO-RASPBERRY
6 A3
5 A2
4 GND
3 3.3V
2 A1
1 A0
LDRs (light sensors)
3.3V
R1
1k
LDR
LDR1
A0
R2
1k
LDR
LDR2
A1
R3
1k
LDR
LDR3
A2
R4
1k
LDR
LDR4
A3
GND
Red LEDs
3.3V
R5
220
D1
R6
220
D2
R7
220
D3
GND
>CNAME
RCJ_2025_SOCCER_LIGHT_SENSORS_BOARD v6
02/05/2025 20:49
Sheet: 1/1
Rev: >CREVISION

16.1.4 Placa del pateador





16.2 Arquitectura de software completa



16.3 Lista completa de materiales (Bill of materials - BOM)

A continuación, se mostrará la lista de materiales completos que fueron utilizados para la creación de ambos robots. Cada material y componente que fue necesario para una estructura y un sistema estable y efectivo. **(Cabe aclarar que la lista a continuación no incluye: Piezas o componentes usados al momento de pruebas, repuestos, costos de envío, impuestos en el total. Los precios a continuación pueden haber sido afectados por la conversión de dólares a pesos)**

Material/Componente	Cantidad	Precio Unitario	Precio total
Polymaker - Filamento PLA PRO 1,75mm	1	$619.00	$619.00
Panel MDF 3mm (1.22 x 2.44)	2	$165.00	$330.00
Raspberry Pi Pico W2	2	$170.00	$340.00
Jetson Nano 4 GB	2	$5,560.00	$11,120.00
PCBs Personalizadas	1	$660.00	$660.00
llantas omnidireccionales GTF de 50 milímetros	8	$800.00	$6,400.00
Tubo de acrílico transparente	1	$133.75	$133.75
Electroimán solenoide JF-0826B	1	$108.00	$108.00
Electroimán solenoide JF-0826	1	$108.00	$108.00
Arducam imx 708	4	$750.00	$3,000.00
Espejo omnidireccional	2	$1,500.00	$3,000.00
motores pololu HP 9.68:1 12V 48CPR Encoder	8	$800.00	$6,400.00
BNO055	2	$900.00	$1,800.00
batería lipo 2200mAh 50C 11.1 V	2	$500.00	$1,000.00
Cable micro USB 15cm	2	$100.00	$200.00
Regulador de Voltaje MT3608	4	$68.40	$273.60
Elevador de Voltaje Xl6009	2	$48.00	$96.00
Mini switch dos posiciones	4	$12.00	$48.00
Capacitor 10000uF 63V	2	$80.25	$160.50
LED Rojo ultra brillante	12	$3.00	$36.00
LDR	16	$2.00	$32.00
Controladores de motor DRV8256E	8	$306.00	$2,448.00
Convertidor analogico a digital MCP3008	2	$235.00	$470.00
resistencias de 1K Ω	38	$3.41	$129.58
Relay PTH	2	$24.00	$48.00
Jumpers	338	$0.90	$304.20

Borneras	36	$1.50	$54.00
Botón Pulsador	4	$1.30	$5.20
Total			$39,323.83