Airborne

To infinity and Beyond

Team

Francisco Gaspar

Team Captain; Lead Programmer; Electronics & PCB Design; Ai

I'm Francisco, and I've been leading the team since the start in 2022. I designed our custom PCBs and handled most of the programming, from low-level control to AI integration. I trained and deployed our YOLOv8 models and built the logic that helps the robot make decisions on the field. I love solving problems by mixing hardware and software.

André Pinto

Team Co-Captain; 3D modelling; Sponsorship Manager

I'm André, and I focused on designing the robot's structure. I created all the 3D models in SolidWorks and made sure everything, from suspension to the camera tilt mechanism, fit and worked reliably. I also took care of getting sponsorships and managing our budget. I really enjoy the mechanical side of things and making sure we're prepared both on and off the field.

European Championship 2024

National Robotics Festival 2025

Airborne Robotics was founded in 2023 with the goal of building a CanSat for the national competition.
Nevertheless, we ended up competing in our first Robotics National Championship (FNR 2024), securing 1st Place and qualifying for the European Championship in Germany.
Although our results weren't perfect, the experience motivated us to completely redesign the robot from the ground up. In 2025, with a new dual-processor architecture and our first reliable custom PCBs, we won 1st place at the National Championship Rescue Line Competition, earning us a spot at the RoboCup World Championship in Brazil.

1st Place FNR 2024

1st Place FNR 2025

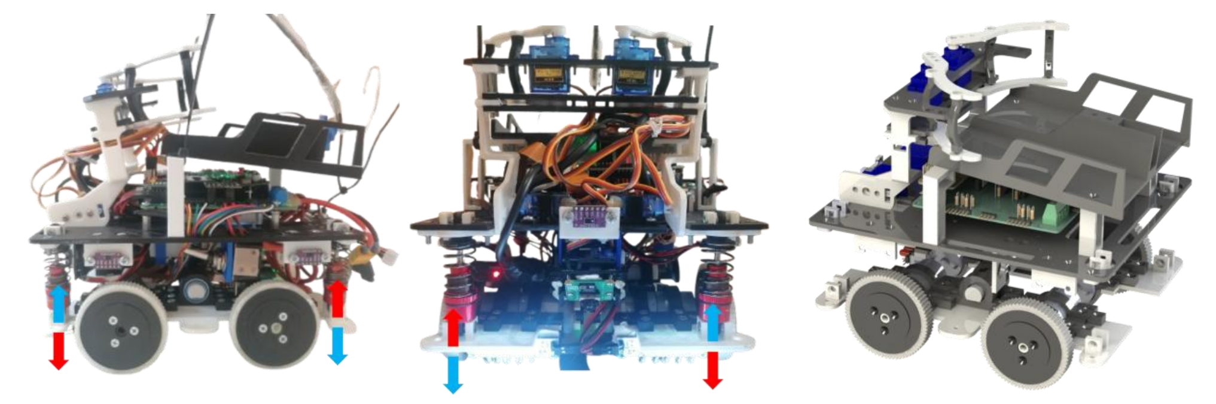
Hardware

Robot Chassis

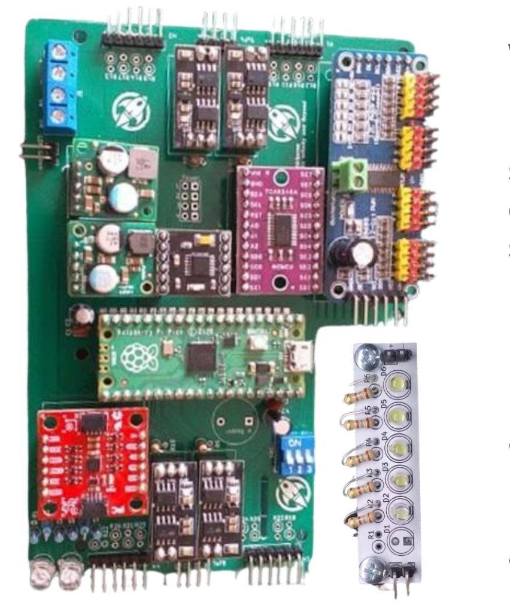
Our robot's chassis was entirely designed in SolidWorks and manufactured using FDM 3D printing with standard PLA, chosen for its rigidity and ease of use. The design features over 70 components, including 30+ unique parts, assembled into a compact 25×20×20 cm frame. Its modular three-level structure – dedicated to victim recovery; sensors & logic; actuation – makes it easy to access, repair, and upgrade. Rounded corners and a low center of mass ensure smooth movement across ramps and more complex terrain.

Passive Suspension

To improve traction and durability, we implemented a passive suspension system: the bottom layer is split into two floating halves connected by bearings and flexible dampers. This allows the chassis to adapt to uneven surfaces and reduces shock from obstacles. The system ensures four wheels almost always stay grounded, improving stability over elements like speed bumps, debris or toothpicks.



Custom PCBs



We designed and soldered two custom PCBs in-house: the Main Control PCB and the Custom LED Bar PCB. The Main PCB integrates the Raspberry Pi 5 and Pico with ESCs, sensors, power rails, and peripherals using clean, socketed connectors. It includes RGB LEDs, a buzzer, and DIP switches for debugging and runtime configuration. The LED Bar PCB drives high-brightness RGB LEDs using a repurposed DRV8833 motor driver for efficient, noise-free illumination.
Both boards were developed using a test-driven approach and are currently on their fourth revision and second versions, respectively. Their compact layout and reliable connections play a key role in simplifying internal wiring, reducing noise, and ensuring system stability.

Sponsors:

ORION TECHNIK MAINTENANCE & ENGINEERING

MAUSER.PT TUDO PARA ELECTRÓNICA

LARANJEIRO · FEIJÓ JUNTA DE FREGUESIA

EID A COHORT PLC COMPANY

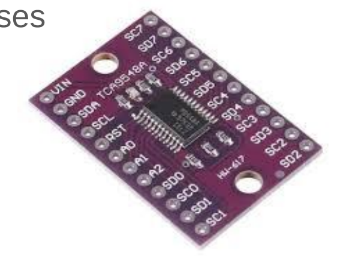
Pololu Robotics & Electronics

CMA CÂMARA MUNICIPAL DE ALMADA

CLUBES CIÊNCIA VIVA NA ESCOLA

TCA9458 Multiplexer

Responsible for integrating the Time of Flights with different I2C addresses



Raspberry Pi Camera Module V3 Wide

Used to acquire visual data; main sensor

Raspberry Pi Pico

Chosen for its versatility and general compatibility; responsible for sensor and actuator integration

Raspberry Pi 5

Main computing device, responsible for image processing and decision making

Google Coral USB Accelerator

This USB Accelerator functions as a high speed AI inference processor, helping the Raspberry Pi running the Victim Detection Ai

5x VL53L0X Time of Flights

These ToFs are used to detect obstacles and help guide the robot during the evacuation zone

ICM-20948 9DOF IMU

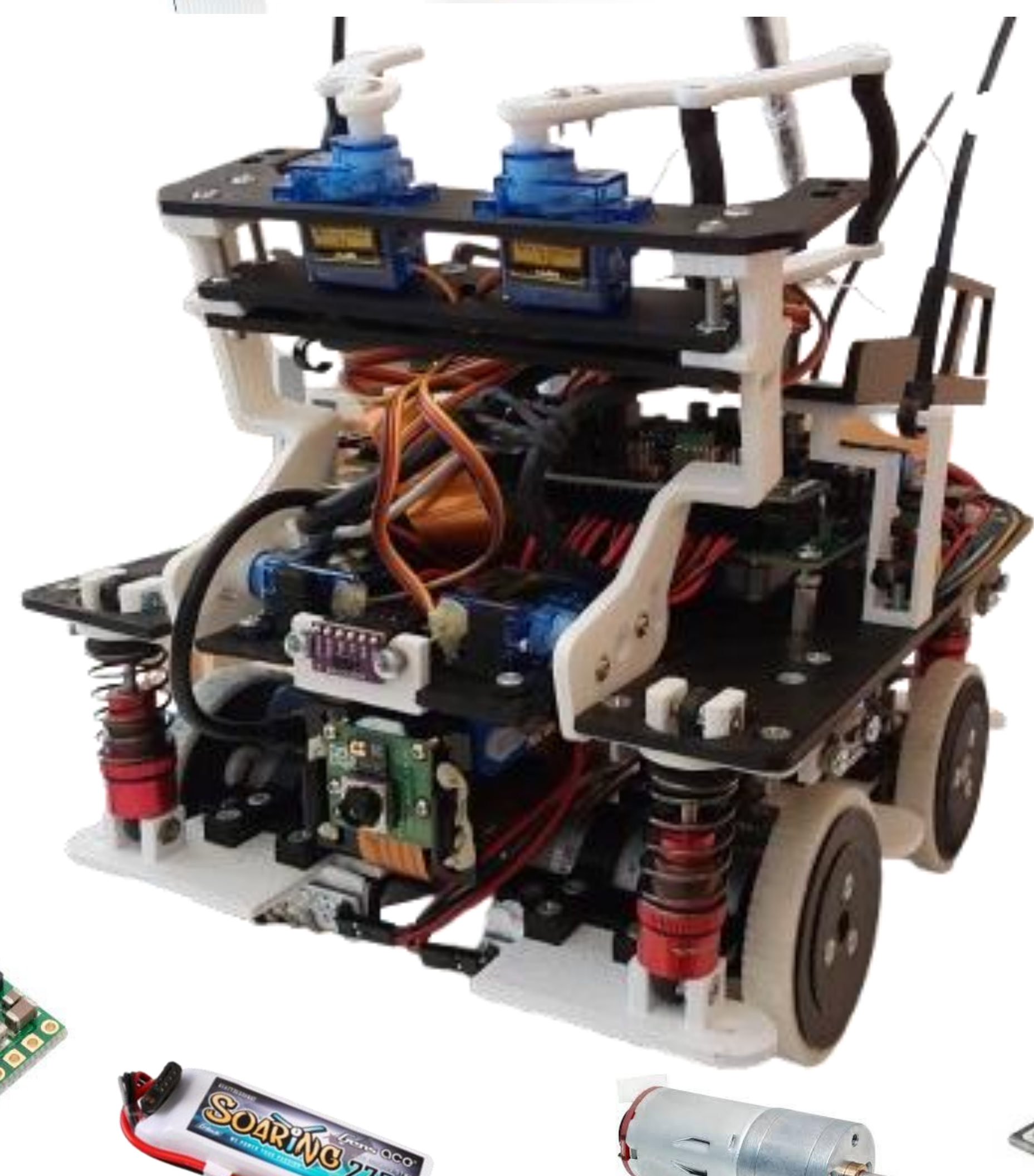
The Inertial Measurement Unit is used to acquire 3D data of acceleration and rotational speed to estimate robot pitch (ramp inclination

D24V50F5 Step Down

This step down provides stable power (5V; 5A) to our Raspberry Pi 5

D36V28F5 Step Down

This step down provides power (5V; 3.2A) to our Raspberry Pi Pico, SG90 Servos, Time of Flights and IMU



Gens Ace 3S Lipo Battery

This battery was selected based on their capacity, power output and dimensions

4x 12V DC Geared Motor

These motors were chosen because of their remarkable power at a decent speed

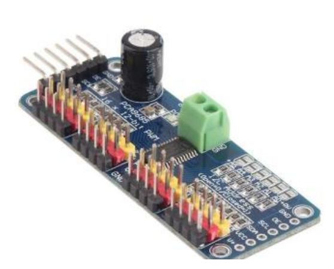
DRV8833 Power Driver

We are using this power driver to be able to control our custom PCB Led Bars luminosity intensity

2x Custom Led Bar PCB

We designed this Led Bar PCB so we could have a modular light source with the dimensions and led number required

PCA9685 Servo Driver



This driver is used to control our servo motors with a safe and stable power supply

6x SG90 Servos

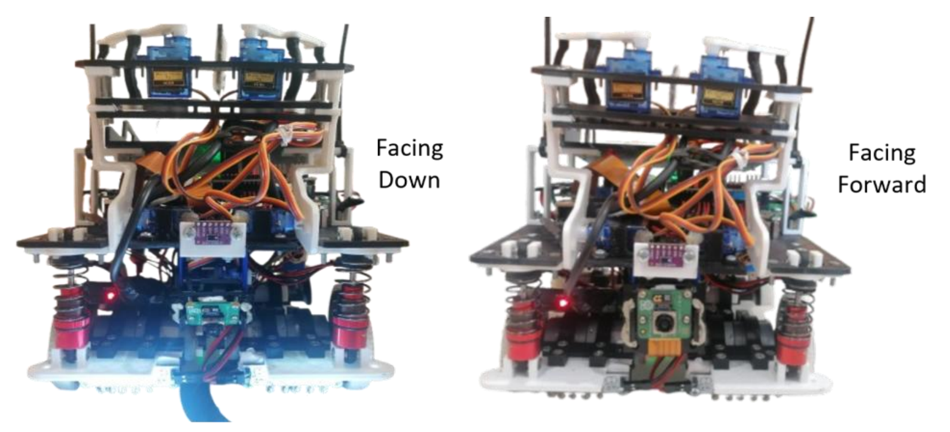
Chosen for their availability, ease of use and reliability; used in our Robotic Arm, Ball Storage and Camera Tilt Mechanism

4x 3S Electronic Speed Controllers

To control our motors we opted for these cheap electronic speed controllers

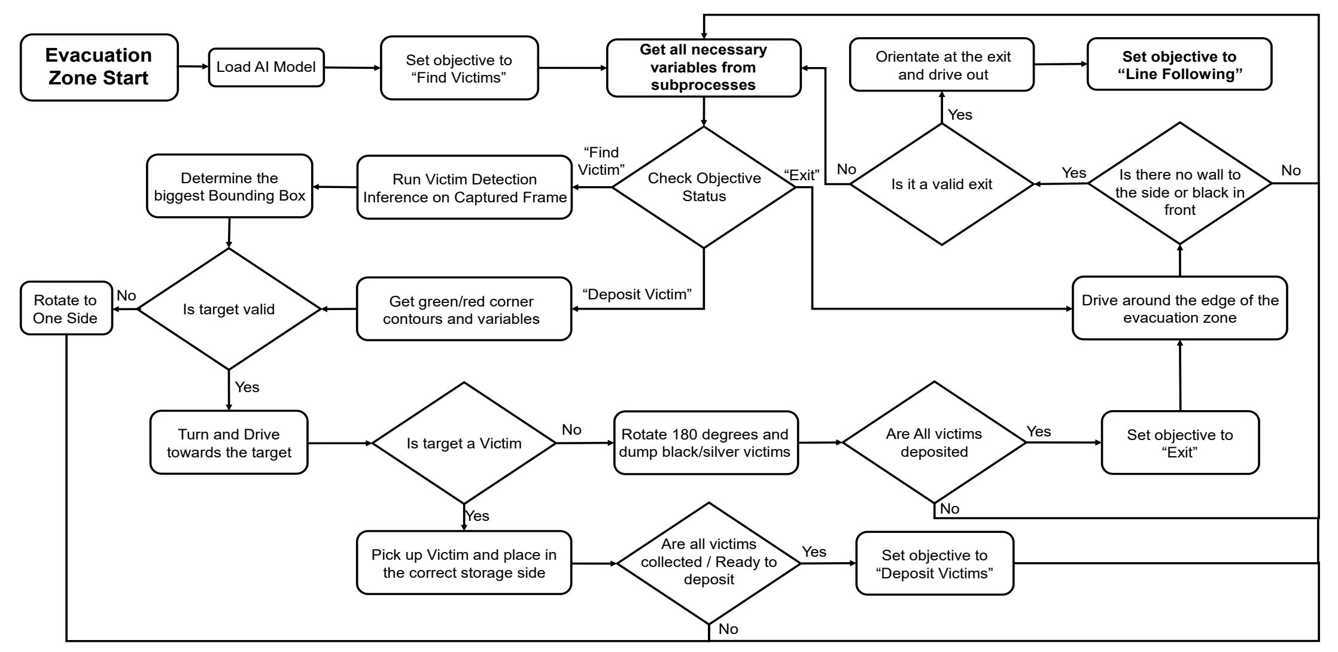
Camera Tilt Mechanism

To reduce complexity and improve versatility, our robot uses a single wide-angle camera mounted on a tilt mechanism driven by a lightweight SG90 servo. This design replaces the typical dual-camera setup, avoiding fragile second flat cables and saving space.
The servo adjusts the camera's angle during a run, improving performance on slopes and allowing the robot to anticipate line gaps by gradually tilting forward. This flexible vision system improves reliability and makes the robot more adaptable to dynamic field conditions.

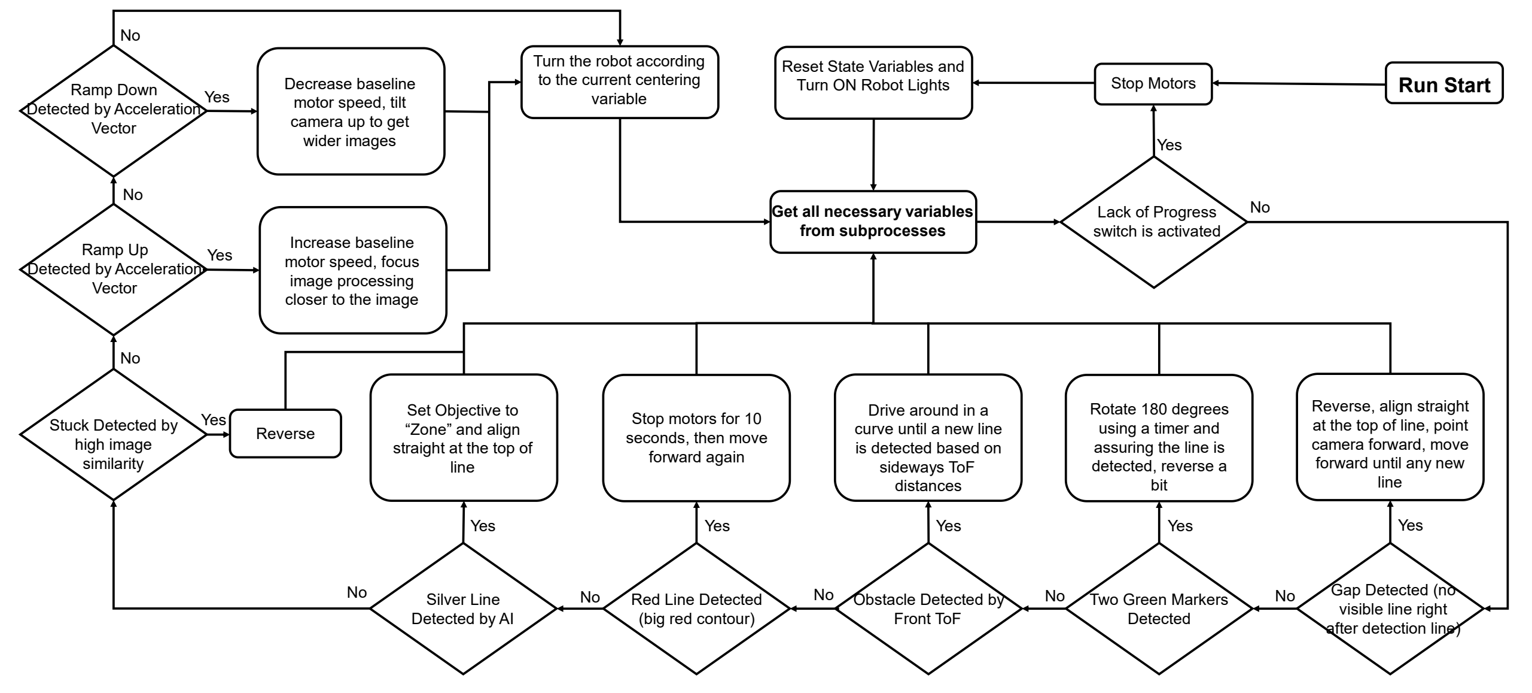


Software

Evacuation Zone


Evacuation Zone Start
Load AI Model
Set objective to "Find Victims"
Get all necessary variables from subprocesses
Orientate at the exit and drive out
Set objective to "Line Following"
Check Objective Status
"Find Victim"
"Exit"
"Deposit Victim"
Is it a valid exit
Is there no wall to the side or black in front
Yes
No
Run Victim Detection Inference on Captured Frame
Determine the biggest Bounding Box
Is target valid
Rotate to One Side
Get green/red corner contours and variables
Drive around the edge of the evacuation zone
Turn and Drive towards the target
Is target a Victim
Rotate 180 degrees and dump black/silver victims
Are All victims deposited
Set objective to "Exit"
Pick up Victim and place in the correct storage side
Are all victims collected / Ready to deposit
Set objective to "Deposit Victims"

Line Following


Run Start
Stop Motors
Reset State Variables and Turn ON Robot Lights
Get all necessary variables from subprocesses
Lack of Progress switch is activated
Turn the robot according to the current centering variable
Ramp Down Detected by Acceleration Vector
Decrease baseline motor speed, tilt camera up to get wider images
Ramp Up Detected by Acceleration Vector
Increase baseline motor speed, focus image processing closer to the image
Stuck Detected by high image similarity
Reverse
Set Objective to "Zone" and align straight at the top of line
Stop motors for 10 seconds, then move forward again
Drive around in a curve until a new line is detected based on sideways ToF distances
Rotate 180 degrees using a timer and assuring the line is detected, reverse a bit
Reverse, align straight at the top of line, point camera forward, move forward until any new line
Silver Line Detected by AI
Red Line Detected (big red contour)
Obstacle Detected by Front ToF
Two Green Markers Detected
Gap Detected (no visible line right after detection line)
Yes
No

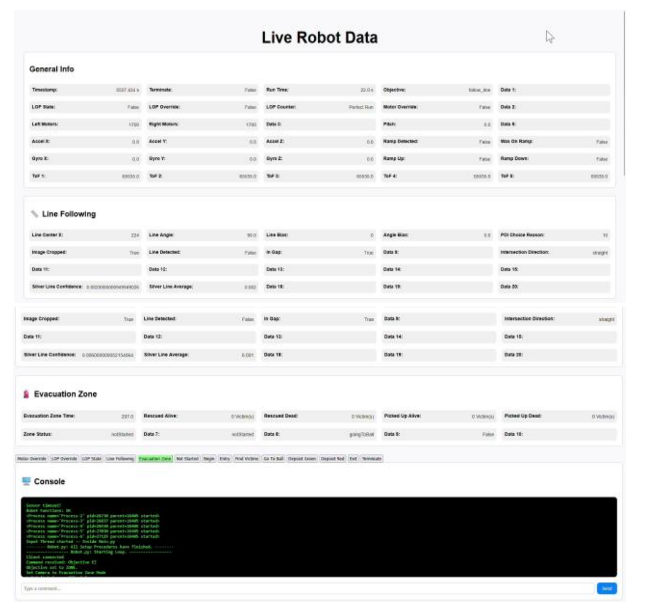
Visual Studio Code: We used Visual Studio Code paired with Remote SSH to seamlessly develop and debug code directly on our Raspberry Pi. This setup allowed editing and testing in a Linux environment while maintaining access to powerful development tools and a direct integration with GitHub. Additionally, by having a raspberry pi standing alone, we could program the robot without being close to it.

Python: Python served as the backbone for our high-level logic and image processing tasks. Its simplicity and extensive libraries made it ideal for rapid development and handling complex computations. Libraries like OpenCV and NumPy enabled advanced vision capabilities crucial for our robot's performance.

Arduino IDE: The Arduino IDE was our choice for programming the Raspberry Pi Pico in C++. Its user-friendly environment and robust library support simplified sensor interfacing and actuator control. Using Arduino IDE, we implemented real-time hardware communication and control critical to the robot's responsiveness.

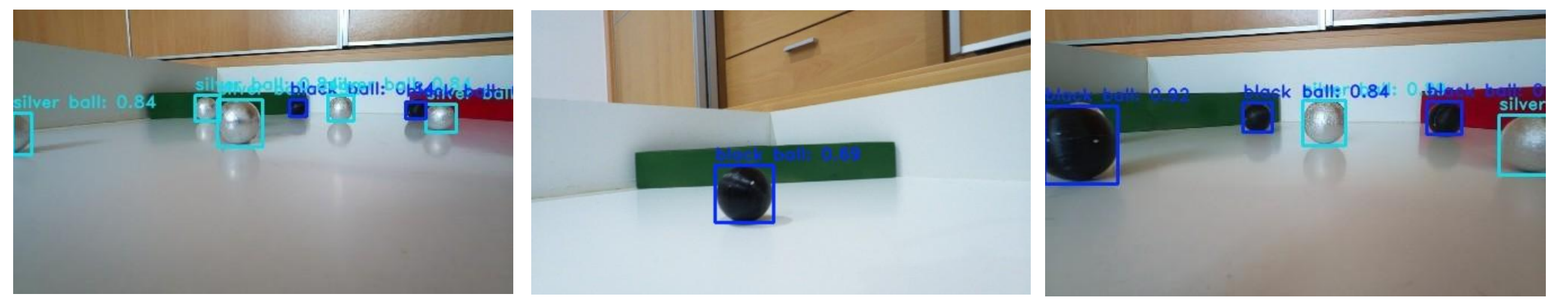
WebSocket and HTML Page

To simplify debugging and better understand the robot's behavior during tests, we developed an innovative WebSocket-based monitoring system. A dedicated process on the Raspberry Pi updates a set of predefined variables at 20 Hz, which can be accessed remotely from our computers. These values are displayed on a custom HTML page designed to replace the traditional console output with a more readable and organized interface.
Although still a work in progress, this tool has already proven extremely useful for logging and real-time analysis, helping us identify bugs, monitor internal states, and tune parameters more efficiently.



Yolo AI Models

To handle critical vision tasks with higher reliability, we trained two custom YOLOv8 small models, both running in real time on a Coral USB Accelerator. The first model is responsible for victim detection in the evacuation zone. This model was trained on a dataset of over 3,000 labeled instances from more than 1,000 annotated images collected during tests and the 2025 national competition.
The second YOLOv8 model detects the silver strip at the entrance of the evacuation zone. This model was trained on over 6,000 labeled examples across 2,000+ images, helping the robot recognize the zone entrance with much greater consistency.



Line Following Algorithm

Our robot's line-following algorithm is based on a seven-point of interest (POI) system that analyzes each frame captured by a wide-angle camera. The camera feed is cropped to a horizontal strip closer to the bottom of the image, where the line is closer to the robot, allowing faster and more relevant processing. From both the full image and the cropped region, the algorithm identifies the leftmost, rightmost, and highest points on the black line. Additionally, the lowest point is considered from the full image. These seven POIs give the robot multiple options for determining the best path forward.

This method, originally introduced by team OverEngineering², was further improved by us with a sharp-turn biasing system. When the robot detects turns sharper than 90 degrees, it adjusts its path planning accordingly. Even if the POI location appears visually similar to a gentler turn. This added logic greatly improves the robot's ability to maintain the line in complex tile configurations. In most cases, the robot follows the highest POI in the cropped region for stable and predictive steering. This approach has proven both fast and reliable, and it handles sudden directional changes better than traditional centroid/moments-following methods.