ROBOCUPJUNIOR RESCUE LINE 2025

TEAM DESCRIPTION PAPER

Airborne

André Pinto and Francisco Gaspar

Contents

Abstract

Our robot was designed to reliably complete all challenges of the RoboCup Junior Rescue Line competition through a combination of computation, custom electronics, and specific mechanical design. Its architecture is centered on a dual-processor setup: a Raspberry Pi 5 handles high-level logic and AI inference, while a Raspberry Pi Pico manages real-time motor and sensor control. A Google Coral USB Accelerator allows onboard execution of two YOLOv8 AI models, one for victim detection and another for identifying the silver evacuation zone entrance, outperforming traditional computer vision methods in accuracy and speed.

The robot's chassis, fully designed in SolidWorks and 3D printed, features over 70 custom parts and a modular architecture that simplifies maintenance and upgrades. A tilt-mounted wide-angle camera, passive suspension system, and dual-servo rescue arm provide great performance on complex terrain. Custom PCBs, developed in-house, manage power, sensors, and actuators while keeping internal wiring clean and reliable. The software runs in parallel processes for camera input, sensor reading, and decision-making for fast and consistent performance. A custom HTML interface, powered by a WebSocket server, allows real-time monitoring and debugging. The result is a fast, reliable, and intelligent system engineered to perform at high level across all stages of Rescue Line.

1. Introduction

a. Francisco Gaspar

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

I am Francisco, and I have 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.

b. André Pinto

Team Co-Captain; 3D Modelling & Structure; Budget & Sponsorship Manager

I am 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.

2. Project Planning

a. Overall Project Plan

Objective: Our team's primary objective for the RoboCup Junior Rescue Line League competition is to design and build a robot capable of navigating the rescue scenarios efficiently and reliably. To achieve this, we have outlined several key requirements and constraints that our robot and team must meet, based on the competition rules and the physical and time deadlines inherent to the event, Table 1.

Requirement	Tools/Components	Final Solution
Navigate and turn within the 25x25x25 (cm) space in between pillars	Custom 3D Model; 12V DC Motors paired with ESCs	We designed a compact 3D printable chassis with dimensions smaller than the ones required for ease of navigation. Also benefits having a lower center of gravity
Overcome speed bumps, ramps and seesaws	Silicone Wheels (traction); IMU ICM-20948 (ramp detection)	The robot has high-traction custom made silicone wheels and a low center of gravity for stability. It uses a hard suspension setup to soften the load on 3D printed parts during high impact situations
Follow the black line; detect green intersection markers and red strip	Raspberry Pi Camera Module V3 Wide; Raspberry Pi 5; Raspberry Pi Pico	We chose a camera as it was the sensor with less limitations and a lot of possibilities, being able to deal with multiple tasks solely relying on it. More specifically, we chose the Raspberry Pi Camera Module V3 Wide. The camera is mounted on a tilt platform that points down on Line Following and forward on the Evacuation Zone
Detect and work around obstacles	Time of Flight VL53L0X	With the use of time of flights (ToFs) we are able to detect any object that is in front of the robot. After detection we can move around it based on side-pointing ToFs.
Detect reflective silver strip at the entrance of the evacuation zone	USB Google Coral TPU; Raspberry Pi Camera Module V3 Wide	After attempting multiple traditional camera algorithms, we decided on using a YoloV8 small classification model, trained on a custom dataset for the silver strip detection. The values are averaged out over a period of time to prevent false positive from new scenarios for the model
Detect victims and evacuation points	USB Google Coral TPU; Raspberry Pi Camera Module V3 Wide	A YoloV8 small detection model trained on a custom dataset was our final solution, as it proved itself to be much more reliable than OpenCV Hough Transform for Circles with basic color detections.
Pick, store and deposit victims	SG90 Servos (Robotic Arm & Ball Storage)	The robot has a robotic arm that is deployed once a ball is within reach. This arm is capable of sorting the balls to different compartments according to their color/status. We opted to have a ball storage capable of storing a total of 4 balls.
Handle and Lack of Progress Procedure	Lack of Progress ON/OFF Switch	As the robot is already quite tall and has a complex 3D design, we chose to incorporate a flexible handle made out of zip-ties. Additionally, we have a main power switch for quick power offs and a Lack of Progress switch.

Table 1: Overview of Requirements and Final Solutions for Rescue Line

Project Schedule: Right after our participation in the European Championship in June 2024, Germany, we decided that we wanted to try rescue line for another year. So, as soon as we finished our final exams, we started planning and preparing. With the plan of what needs to be changed, added and improved, we created a schedule for the year, that can be seen in Figure 1.

For the start of the year, we were able to stay relatively close to the schedule we had set up previously. Unfortunately, due to some problems with our power driver choices and camera positioning, we ended up taking longer than expected trying to integrate some software solutions because the robot was not as responsive as expected. Eventually, we caught up to the schedule with the majority of the problems solved.

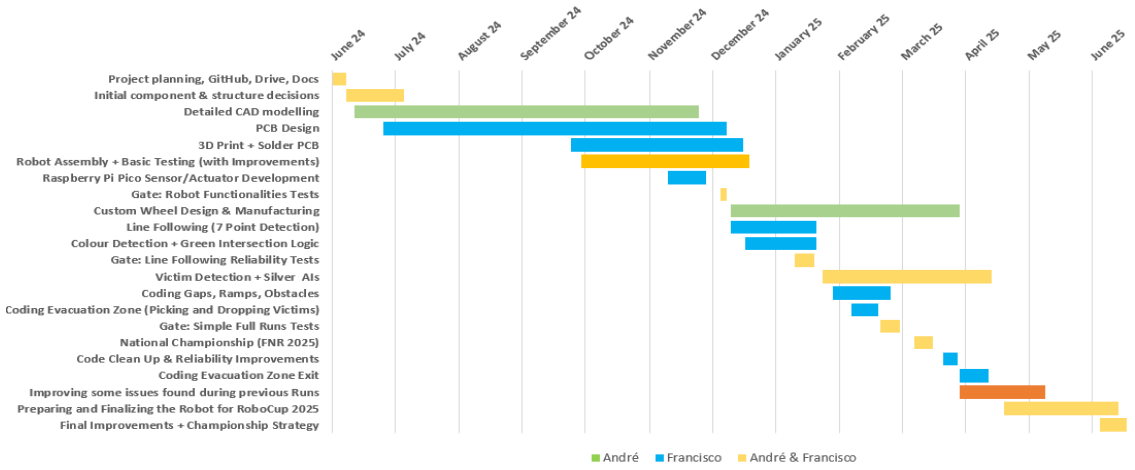

Figure 1: Gantt Chart of our Project Schedule

b. Integration Plan

According to the requirements of this competition, defined in Table 1, we designed our robot with the integration of specific sensors and components to meet those expectations. Each component we integrated underwent a selection of two main phases. The first was a testing phase, during which we developed a simple setup to evaluate and try out whether the component could reliably perform the required functionalities. If it met our expectations, we proceeded to the second phase: integrating the component into the main system. This included implementing the necessary code in the main program and conducting further validation tests such as compatibility in between sensors. An explanation of how each component addresses the competition requirements is presented in Table 1. The overall architecture, including connections and interactions between components, is illustrated in Figure 2.

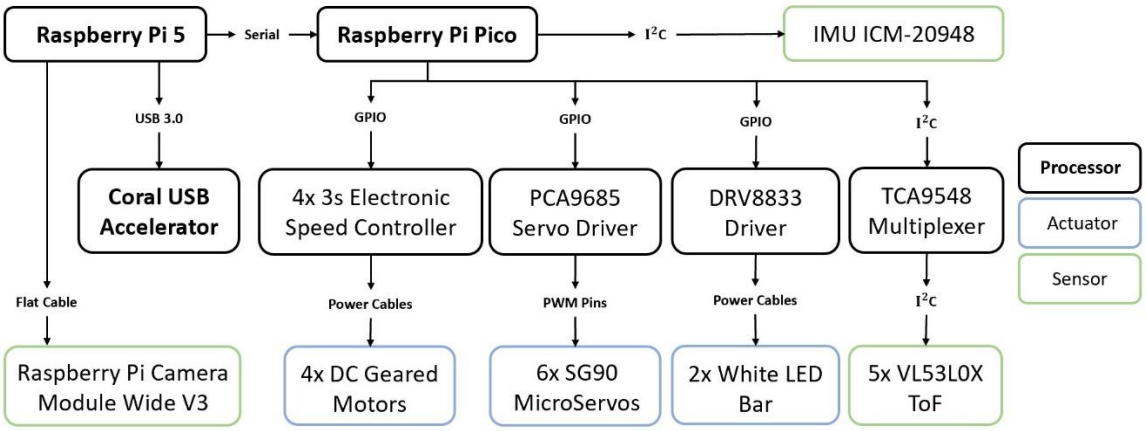

Figure 2: Integration Plan

3. Hardware

Our robot hardware was designed to be fully custom and as much as possible home-manufactured. We opted to design and 3D print the entire structure, enabling a chassis tailored specifically for this competition, with optimized sensor placement, robustness, and modularity in mind. This approach also allows for rapid replacement of individual components when necessary.

Based on the requirements outlined in Table 1, we selected the components shown in the hardware diagram (Figure 3). We then designed the full CAD model of the robot and a Custom PCB, ensuring that all connections are both mechanically stable and easily removable for maintenance/replacement. As a result, our robot features a modular three-level hardware architecture, assemble with screws and developed to prioritize stability and reliability. Each level corresponds to a specific electro-mechanic, computational or mechanical subsystem and collectively contributes to the robot's effectiveness in the Rescue Line environment.

The lower level contains the drive train system, composed of four 12V DC motors with custom silicone wheels, and is designed with a central axis that allows left or right side tilting to improve maneuverability. The middle level houses the core electronics, including the Raspberry Pi 5, a custom PCB, and the Google Coral, and supports the servo-driven arm used to manipulate victim balls. Finally, the top level integrates a dual-compartment storage system with a servo-controlled gate to sort and release balls based on classification.

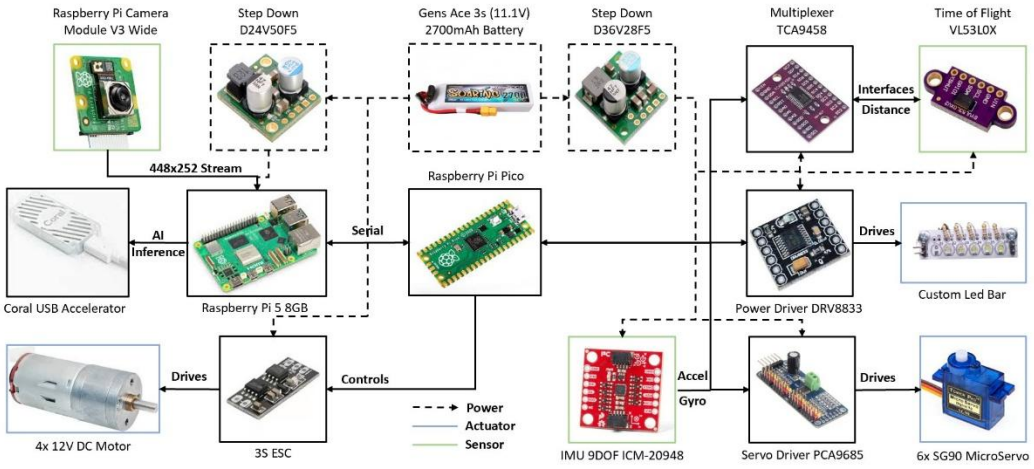

Figure 3: Hardware Diagram

a. Mechanical Design and Manufacturing

Robot Chassis: Our robot's chassis and all mechanical components were fully designed in SolidWorks, allowing for tight integration of submodules and optimization of internal space. All structural parts were manufactured using FDM 3D printing with PLA filament, chosen for its simplicity and mechanical strength. The design comprises over 30 unique 3D-printed parts, totaling more than 70 components. This modular architecture based on many small parts allows easy maintenance and upgrades, as individual parts can be replaced and redesigned without affecting the full assembly.

The robot has compact dimensions of approximately 25 × 20 × 20 cm, weighing 1.8 kg. This compact size and low center of mass, provide strong stability on ramps, seesaws, and speed bumps. The robot has rounded corners to prevent getting stuck on pillars and the zone entrance/exit. The structure prioritizes function over aesthetics, featuring a simple zip-tie handle for flexible handling during testing and setup. All design decisions were driven by the requisites on table 1. See the full CAD chassis design and the fully built robot in Figure 4.

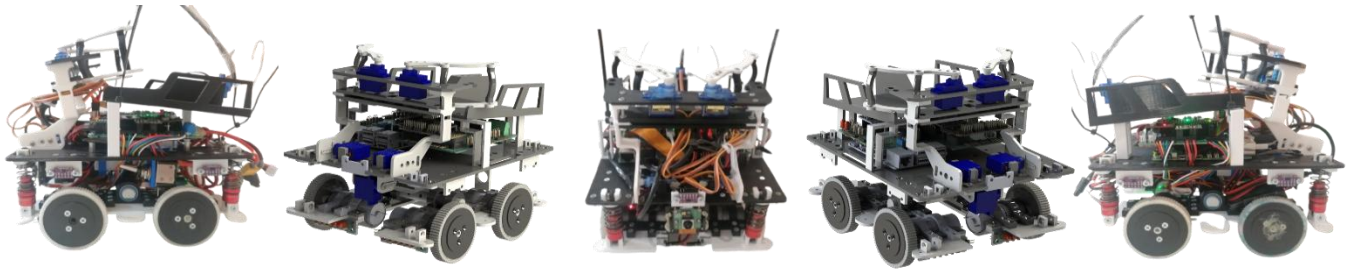

Figure 4: Robot Chassis Design Rendering and Fully Finished Robot

Rescue Mechanism: Our rescue arm is mounted on the second level of the robot and is controlled via SG90 micro servos, selected for their availability, light weight, and ease of use. The arm is designed to grasp and lift spherical victims, using a dual-servo gripping mechanism that adjusts the claw width during operation. While no encoder-based feedback system is in place, the mechanical simplicity ensures repeatable and reliable pickups within the expected rescue area dimensions.

The actuation mechanism has been fine-tuned to ensure smooth release without high acceleration movements and minimal accidental drops. Additionally, the robotic arm is able to position the balls on the left or right storage side, therefore allowing it to sort between alive and dead victims.

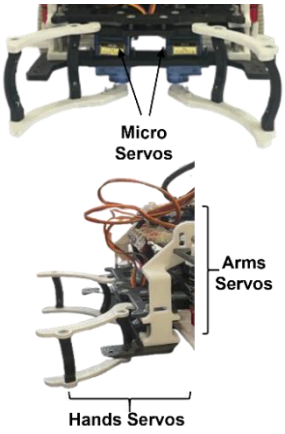

Figure 5: Robotic Arm

Suspension System: To tackle uneven terrain such as speed bumps and debris (see Table 1), we developed an innovative passive suspension system that enhances both stability and durability. Rather than rigidly connecting the top and bottom structural levels, the lower plate is split into two independent sections (left and right), each able to tilt forward and backward via bearings. These "floating" halves (Figure 6) are connected to the main chassis using semi-flexible dampers, allowing vertical displacement and increased flexibility. This setup ensures that almost always four wheels remain in contact with the ground, even on irregular surfaces. Beyond improving traction and obstacle traversal, the suspension absorbs shocks that previously caused mechanical failures — notably, the main axle's connection to the frame. In testing over rice, toothpicks, plastic fragments, and speed bumps, it significantly reduced vibration and preserved structural integrity.

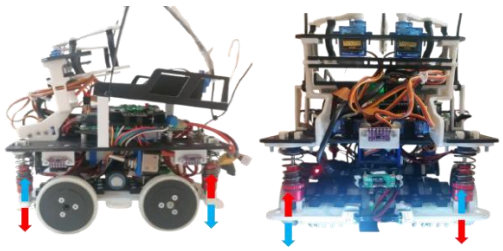

Figure 6: Suspension Movement Diagram

Camera Tilt Mechanism: Instead of relying on a conventional setup with two rigidly mounted cameras, we designed our robot with a single camera mounted on a tilt mechanism (see Figure 7). This innovative solution allows us to dynamically adjust the viewing angle based on the robot's current situation, compensating for ramps, improving line visibility, and even enabling the robot to anticipate or skip gaps by looking further ahead.

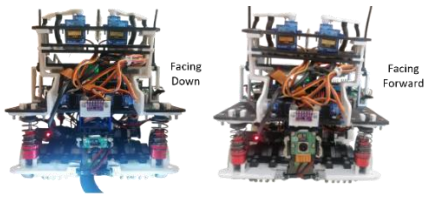

Figure 7: Camera Tilt Mechanism

These design choices proved to be highly reliable during our testing. The robot operated for extended periods without overheating or wiring issues and successfully handled inclines up to 30 degrees. It also withstood small shock and collision tests, maintaining full functionality even after repeated transport and setup cycles.

b. Electronic Design and Manufacturing

Processors, Sensors and Actuators: The electronic architecture of our robot is built around a dual-processor system to balance performance with real-time reliability. We use a Raspberry Pi 5 (8GB) for high-level processing tasks such as AI inference, computer vision, and actuator output control. For responsive low-level handling, we use a Raspberry Pi Pico, which manages sensor polling and communication with actuators. These two systems are connected over a USB serial interface, ensuring robust communication while distributing computational tasks efficiently between them. A wide-angle Pi Camera streams video to the Pi 5, assisted by a Google Coral USB Accelerator for fast onboard YOLOv8 AI inference—used for both victim detection and silver line recognition.

Sensor input is provided by five VL53L0X Time-of-Flight sensors via an I2C multiplexer, and an ICM-20948 IMU for 3D acceleration and 3D rotation speed observation. Four 12V DC motors are driven by ESCs, while six SG90 servos handle the robotic arm, camera tilt, and ball storage. Two custom LED bars, controlled by a repurposed DRV8833 motor driver, ensure consistent line illumination. Power is supplied by an 11.1V LiPo battery, stepped down using two regulators to separately power the Pi 5 and all other electronics, ensuring system stability.

Custom PCBs: To ensure a compact and reliable system, we designed two custom PCBs using KiCad: the Main Control PCB and the Custom LED Bar PCB (see Figure 8). The Main PCB connects the Raspberry Pi 5, Raspberry Pi Pico, ESCs, sensors, and power rails, serving as the central hub for routing and integration. It includes easily replaceable connectors, a DIP switch for runtime configuration, and onboard RGB LEDs and a buzzer for debugging.The Custom LED Bar PCB was developed for high-brightness signaling using a DRV8833 motor driver, chosen for its ability to handle higher currents with simple digital control. The board features current-limiting resistors and precise layout to meet our size requirements.

Both boards were designed and soldered in-house, manufactured by JLCPCB. A test-driven workflow was followed, beginning with breadboard validation and progressing through schematic simulations and field testing. Our current revision, version 4, reflects multiple iterations based on integration feedback. Final validation included power stability tests, component calibration, and real-world tests across ramps and uneven surfaces. These PCBs were critical for system modularity, reliability, and clean internal wiring, directly supporting our robot's design goals.

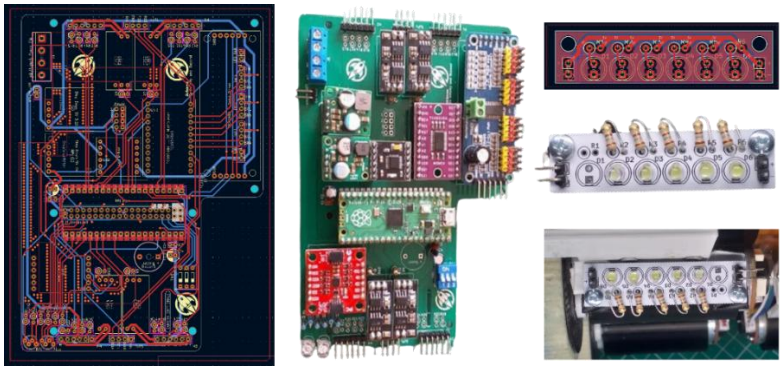

Figure 8: Main PCB and LED Bar PCB

To ensure electronic reliability, we conducted multiple validation tests. Battery autonomy was verified to support over 20 minutes of full operation, with voltage consistently staying above 12V to maintain motor performance. We monitored and logged voltage levels during test runs and confirmed that the system remains stable even under peak loads. Our design includes capacitors for noise reduction and protection, and both step-down converters feature built-in diodes to prevent reverse current damage.

4. Software

a. General Software Architecture

Our robot's software was developed primarily in Python running on a Raspberry Pi 5, while a Raspberry Pi Pico runs C++ code to handle low-level motor and sensor control. The two communicate over USB serial. We wrote most of our code using Visual Studio Code with remote SSH into the Pi, and used GitHub for version control and WinSCP to transfer videos, images and logs. The Pico was programmed using Arduino IDE. Libraries like OpenCV and NumPy are our mainly used libraries. For object detection and classification, specifically victim recognition and silver strip, we use AI models running with the Ultralytics API, accelerated with a Coral USB TPU.

The RPi program is organized into separate processes to run tasks in parallel, see Figure 9. One handles the main camera for line following and intersection detection. A second process reads sensor data from the Pico. A third process controls the robot's decision-making. This multiprocessing setup allows the robot to analyze visuals, collect sensor inputs, and take actions without delays or slowdowns, easily staying within the respective refresh rates.

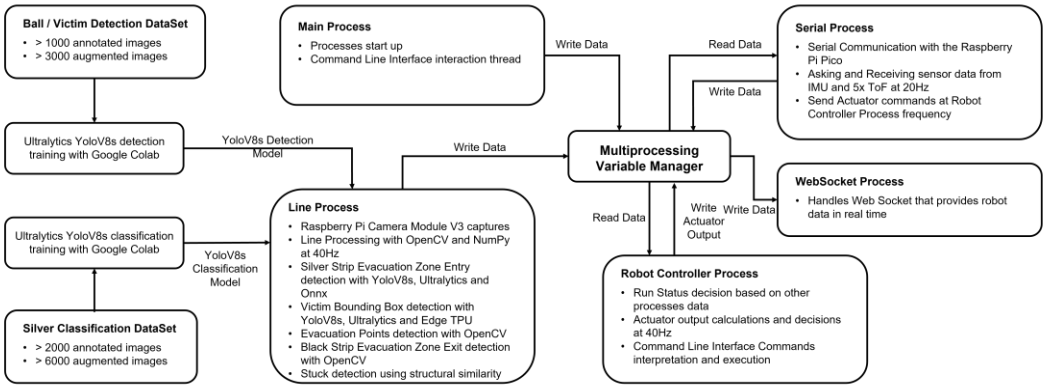

Figure 9: General Software Flow Diagram

Line Following: The line following is done entirely through computer vision, except for the obstacles. Images from the front-facing camera are processed in real time at 40FPS. In the figure below, Figure 10, is a simplified flowchart of our robot handles the line following, accounting for all possible scenarios described in Rescue Line Rules and requirements shown in table 1. Thanks to our custom LED bar, we reduced unidentified colors by approximately 85% with general calibration, and with environment-specific tuning, color detection accuracy reaches around 95% surpassing our reliability requirements.

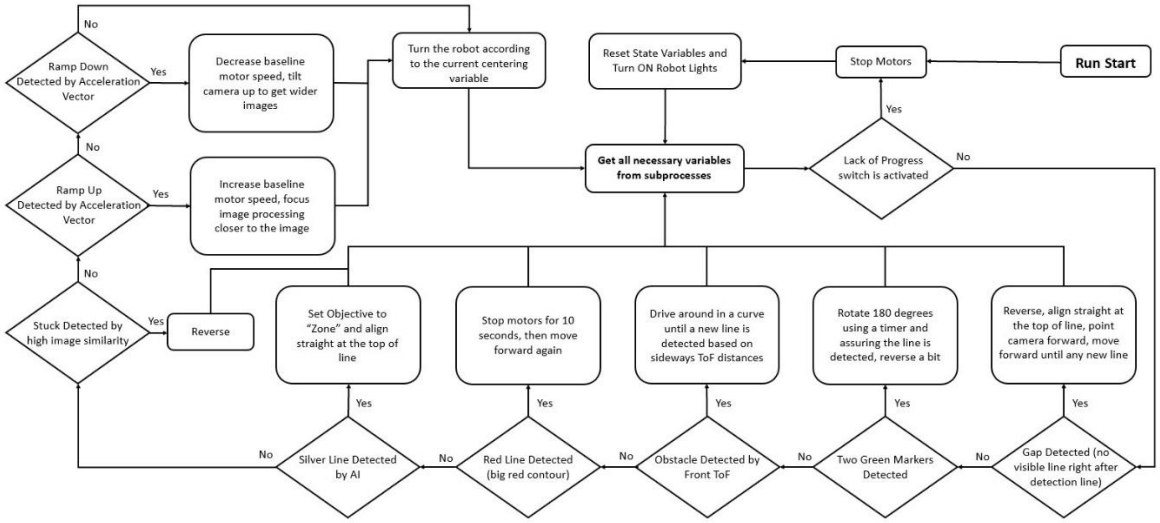

Figure 10: Line Following Flow Diagram

Evacuation Zone: When the robot detects a silver strip, it switches to the evacuation zone mode, shown in Figure 11. The tilt mechanism points the camera forward, the line following lights are turned off (helps with Victim Detection AI), and the software searches for silver and black balls. A YOLOv8s model detects the balls, allowing the robot to collect and sort the silver and black balls. Our strategy is to first collect all victims before delivering them, but if the situation gets difficult, meaning we are in the evacuation zone for too long or we have used lack of progresses during it, we deposit the balls as soon as possible. To ensure reliability, we tested this system under varying lighting conditions and found that the robot detects victims with over 90% confidence in most cases.

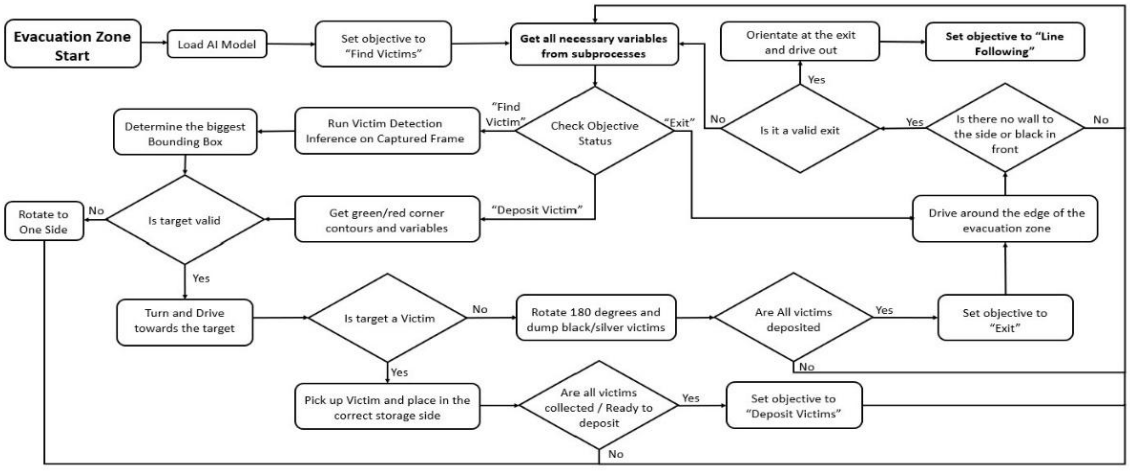

Figure 11: Evacuation Zone Flow Diagram

b. Innovative Solutions

Line Detection: Our line detection algorithm relies on a seven-point of interest (POI) system to determine the position and direction of the black line in each frame (see Figure 12). Inspired by a method first introduced by team Overengineering[2], we expanded on it by adding an extra directional bias for sharp turns. This means that turns sharper than 90 degrees are treated with increased weight, even when the chosen POI lies on the image's edge in both standard and sharp turn cases. This adjustment significantly improved reliability. The robot uses a wide-angle camera near the ground to capture the path ahead and selects the most appropriate POI, typically the highest one in a cropped region, for smooth steering. In Figure 12, the largest red circle shows the point the robot aims to follow.

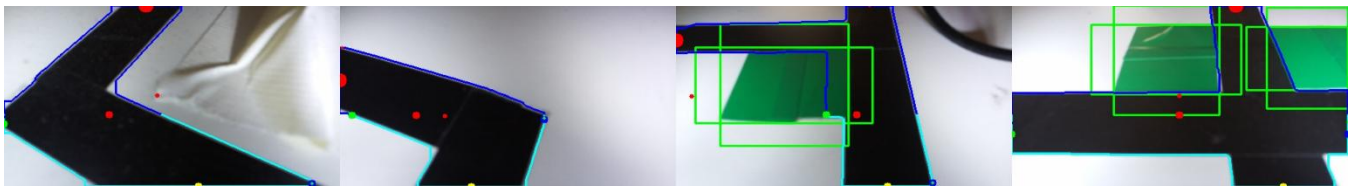

Figure 12: Line Detection with 7 POIs

AI YoloV8s for Victim and Silver Strip Detections: We use two custom-trained YOLOv8 small models to improve reliability in key vision tasks (see Figure 13). One model detects and classifies victims in the evacuation zone, distinguishing silver (alive) from black (dead) balls with high accuracy (around 90%), even under difficult lighting or cluttered backgrounds. The second model is used to detect the silver line at the entrance of the evacuation zone, replacing earlier OpenCV-based methods that failed under certain lighting conditions. Both models were trained using annotated datasets collected during tests and our national competition in 2025. Inference is accelerated using a Coral USB Accelerator, allowing us to run these models in real time without slowing down the rest of the system.

Figure 13: Victim Detection with YoloV8s model

WebSocket Process and HTML Page: To simplify debugging and better understand the robot's behaviour during tests, we developed a 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 (see Figure 14), 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.

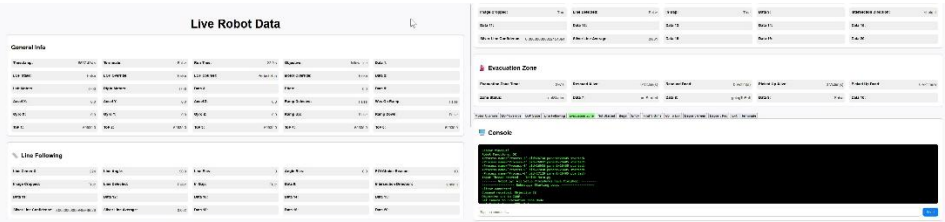

Figure 14: Custom HTML Page for debugging

5. Performance Evaluation

To ensure reliable performance under competition conditions, we carried out continuous testing throughout development. We built and maintained a growing test course, adding new challenges without removing previously working tasks. This allowed us to consistently verify that updates did not break earlier functionality. Tiles included sharp turns, tight intersections, silver markers, gaps, and ramps to test both image processing and mechanical stability.

Hardware systems like the suspension and drivetrain were tested on various surfaces, such as rice, tape, plastic, and speed bumps; to simulate difficult terrain. Ramp testing was conducted in our school gym using a large adjustable ramp to determine safe incline limits.

To improve software reliability, we developed debugging scripts and protocols. Most importantly, we can easily switch our code to Debug Mode, where instead of relying on live robot information and actuation, we use a pre-acquired dataset. Additionally, a WebSocket-based HTML interface provided live data during runs, making it easier to trace logic errors and spot unusual behavior. Screen recordings were also used to analyze edge cases and rare failures.

One significant challenge we faced was inconsistent line detection caused by variable lighting conditions across different environments. Reflections, shadows, or low contrast between the line/green markers and the floor often confused our image processing pipeline, leading to unstable navigation. To mitigate this, we designed custom LED bar PCBs to ensure uniform lighting directly in front of the camera, significantly improving reliability during line following. These lights can be toggled in software, allowing us to disable them when entering the evacuation zone, where our object detection models perform better without artificial lighting.

6. Conclusion

After competing at the European Championship last year and winning the national championship this season, we are proud of how far this project has come. Building everything from the ground up, from structure to software, was a challenging but rewarding journey. While no system is perfect and improvements are always possible, we are satisfied with the reliable and competitive robot we've created. We now look forward to the World Championship not just to compete, but to celebrate what we have built and learn from the global community.