RoboCupJunior Rescue (Line/Maze) 2025

Team Description Paper

STEN

Hampus Mollgård, Malte Robért & Oscar Bergh

Abstract

STEN is a robot designed to compete in Robocup Junior Rescue Line. The current version (STEN 11) is built with the final goal of winning the world championship 2025 held in Salvador, Brazil between the 15 and 21 of July. Our robot consists of a custom designed three-wheeled chassi and drives according to data from two individual wide angle cameras. We use OpenCV and NumPy for image processing, alongside multiple object detection models. The physical construction is developed in Fusion 360 and manufactured via 3d-printing. This is our third year competing, and we have throughout our entire development found multiple innovative solutions that might increase our chances this summer. Setting us apart from many other teams is our miniscule size. Our counterintuitive decision to make our robot tiny is highlighted more thurroly in section 2a. We have also designed to sacrifice a large amount of our limited space to a server fan integrated in the center. While helping to cool all components, this fan's main purpose is actually to remove debris from the track. This theory is described further in section 3. Setting us apart further from other teams is our three-wheeled drivetrain. This decision allows us to constantly remain in full control of the robots every movement keeping a consistent point of rotation.

1. Introduction

a. Team

Malte Robért

With almost three years being involved in RoboCup Junior Rescue Line and an even longer background of building and coding robotics I am responsible for programming the robot. Ever since I was a kid I have been interested in making autonomous and moving robots that ”think”. I love to take on the difficult challenges and create my own ways of solving a problem, therefore during the development of the current generation of STEN I have been responsible for the software and algorithms.

Oscar Bergh

After getting third place with a randomized group in a local RoboCup competition 2022, me and three others believed that we could take this further and formed a new team called STEN. Before that I had only been working with simple electronic circuits, 3d printers and lego robots. During development of STEN I have gained a lot of experience in CAD and programming. This year I am mainly responsible for the chassis and electrical system.

Hampus Mollgård

Having grown up on a farm, I have always loved making and constructing. Initially this consisted of physical wood/metal fabrication in my own tiny workshop, but the outlet for my interest has evolved throughout my life. My love for inventing has led to a developed interest in 3D-printing, my entrepreneurial personality has contributed to an ability to sink completely into my work, and my relatively newfound love for electronics has been highly developing personally. With this background, I have mostly been responsible for the mechanical construction while simultaneously challenging my electronic competence ever since we started our robot development.

2. Project Planning

a. Overall Project Plan

Objective: Our goal is simple, we aim to win the World Championship of RoboCup Junior Rescue Line 2025. However, achieving this goal is not a simple task but with two years in the competition and a competent team we will always strive for redundancy and perfection. To achieve this goal we have not only implemented the RoboCup federations rules but also our own constraints to give us the best possible solution.

- Maximum footprint size of 180x160 mm - In order to ensure easy navigation without the risk of getting stuck due to faulty alignment, we will keep the robot footprint to a minimum. This will make the robot more agile and increase mobility.
- Handling of 30° slopes - With the new rules for 2025 awarding points per navigated ramp tile we want to ensure with great redundancy that our robot can handle more than the rules requirement of 27.5° including the 10% tolerance.
- Easy GUI for debugging - We need an easy GUI to debug and find errors in the code. The GUI should therefore show us all the relevant data and the core decisions being made.
- Not being affected by external lighting - Glare and bad lighting can cause the line following algorithm to underperform and therefore we need to make sure that the lighting is even and correct for our application.
- Minimum of 20 fps in linefollowing - To ensure that no details are missed during runs we have set a requirement of 20 fps during live runs.
- Completion of rescue zone in under 2 minutes - We have estimated that our robot could complete the rescue zone in 1.5 minutes with perfect conditions therefore we want to complete it consistently in under 2 minutes to save time for other moments.

Project schedule: After two years in the RoboCup Junior Rescue Line competitions we have a pretty good way to structure the work. We have evaluated our previous runs and used our experiences to build a reasonable timeline for our robot. As shown in table 1, tasks are split up between all three team members based on the first letter of their names. Malte => M, Hampus => H, and Oscar => O.

Table 1: Project schedule

Milestone	Deadline
Implemented a camera based system on STEN 8 to test visual navigation. (M)	2024-09
Having decided on components for the next generation of STEN based on the previous testing. (M, O and H)	2024-10
Developed a design that includes the new components. (O and H)	2024-11
New prototype for next generation of STEN done. (STEN 9) (O and H)	2024-11
Checkpoint: A prototype of STEN 9 has been built and is going through testing at our newly developed training course.	2024-11
Performance of prototype tested and evaluated. (M)	2024-12
Checkpoint: Tank/Belt steering has been decided on as the drivetrain. Active mapping whilst the robot is driving around on the course has been implemented to assist in tricky situations and dead angles from the camera.	2025-01
Design of STEN 9 settled. Methods used for navigation and core functionalities should be established. (M, O and H)	2025-02
Both STEN 8 and STEN 9 ready for the local district championship. (STEN 8 as a backup) (M, O and H)	2025-03

Checkpoint: Vicktory at the district championship. Large rooms for improvement noticed. Tank steering abandoned in favor of three-wheeled robot	2025-03
Revisions of STEN completed incorporating lessons learned from the local district championship. STEN should be ready for the Swedish Championship and sponsors should have been established. (M and H)	2025-04
Checkpoint: Victory at the Swedish championship. Still areas with room for improvement.	2025-04
Revisions of STEN incorporating lessons from the Swedish Championship. (M)	2025-05
Final touches and optimizations done. Sponsors for the World Championship should be established. We should have spare parts for every single part on STEN. (O and H)	2025-06
STEN should be ready for the World Championship. (M, O and H)	2025-07

This schedule was mainly decided upon in order to allow all team members to work simultaneously on different tasks without interfering with each other. We have also chosen to include multiple iterations in our schedule to rapidly develop the robot. Testing many different versions, with new parts, new drivetrains, and new methods allows us to find new and great concepts faster than a complete development. Following this development we have also staggered software- with hardware development ensuring both fields are always improving alongside each other while still remaining compatible. This method of development requires more work, building and coding multiple robots, but with the benefit that our first plan does not have to be perfect and that we always can keep learning from our mistakes.

In general we have been able to follow our schedule quite closely, and since we have been competing for multiple years this far, we knew what to expect from each of the different competitions. For the local district championship we did not expect any difficult competitors, and were quite certain that a robot capable of basic line following, intersections, 25° ramps, and a mediocre rescue zone would be able to win. This theory was correct and we were able to win, being the only team gathering points in the rescue zone. The Swedish championship 2024 consisted in the largest part only of advanced line following, and we expected this year to be similar. One aspect about the Swedish finals was however a great fear for our team. The Swedish Robocup federation invested in new rescue-line tiles 2024 and accidentally bought shiny, mirror-like tiles. They are fully allowed to use these tiles according to §3.2.1, and §3.3.1. We did however fear the increased reflectiveness would cause difficulties for our camera, and therefore decided to make both STEN 8 (without camera) and STEN 10 ready for the competition. We ended up using STEN 10 and won this championship as well, rescuing all possible victims during both our scoring-runs.

For the world championship we expect a large presence of slopes and ramps following our experience in Eindhoven 2024. We also expect a lot of debris, and intersections present on the course, alongside a cluttered evacuation zone filled with obstacles and debris. Therefore we know we need to keep the last few months dedicated to one robot and doing a lot of final adjustments. From the district championship to the world championship, we decided to cease our iterative processes, as a rapid prototype only provides results limited to the scope of the method being tested. Instead, we require the advantages of a well-conceived and comprehensive design. We will only practice (mostly on slopes, i.e. intersection in

slopes, obstacles in slopes, etc.), adjust, and tweak during the last few months, with the ultimate goal of achieving first place.

b. Integration Plan

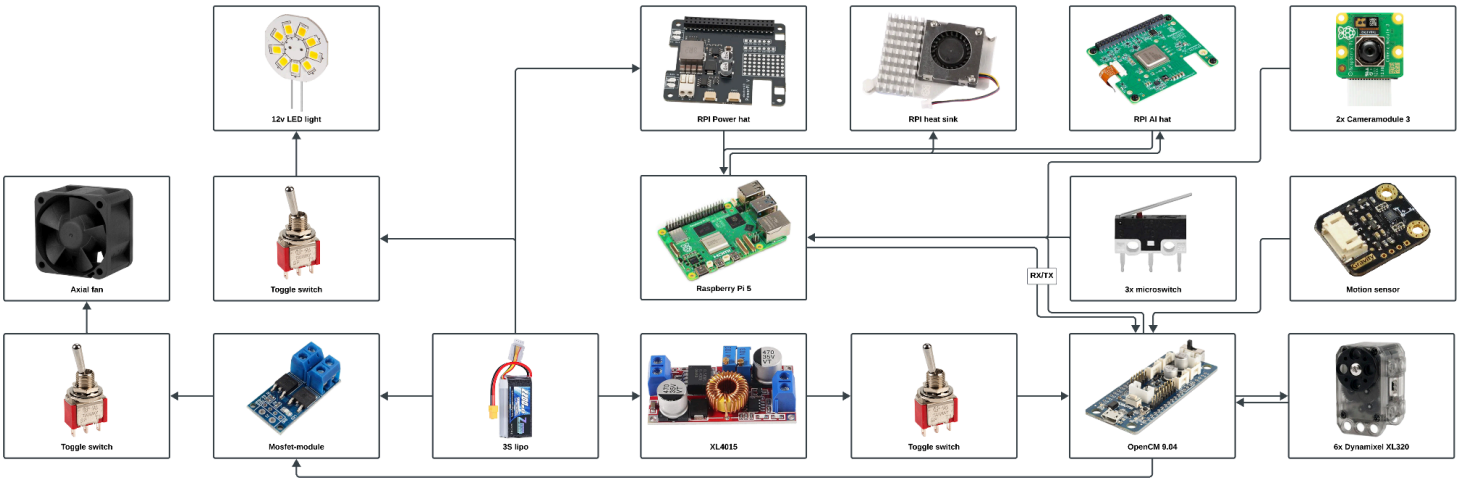
Our early development process of constant brainstorming and continuous testing has given us a large insight into different solutions and requirements for the robot. More than complying with the official rules, we had a clear objective to fulfill our own requirements highlighted under section 2a. These challenges and our solutions are highlighted in table 2.

Table 2: Integration of solutions

Requirement	Solution
Maximum footprint size of 180x160 mm	Solved by always keeping size and space in mind during development. Compromising with smaller and fewer components, and optimizing the design.
Handling of 30° slopes	Solved by using lego wheels for their high friction and a drivetrain that allows full control without needing any wheels to move sideways.
Not being affected by external lighting	Solved by installing a dedicated lightsource capable of overpowering any external light sources
Minimum of 20 fps in linefollowing	Solved by thoroughly considering all components, using more expensive parts, and optimizing all code. The implementation of a 26 TOPS AI accelerator also helps with performance

All components in STEN have been thoroughly tested throughout our iterative process, and components we deemed useful or necessary in previous robots have been carried over into the present version. With this method of development we not only reuse old components (keeping costs down), but also develop a great familiarity with each part. All current parts, and along with the connections are provided in figure 1. For more information on the parts functionality/use-case and the robots general construction see section 3, and for resources on the individual parts follow the parts list provided under references, or information on our github.

Figure 1: Hardware parts



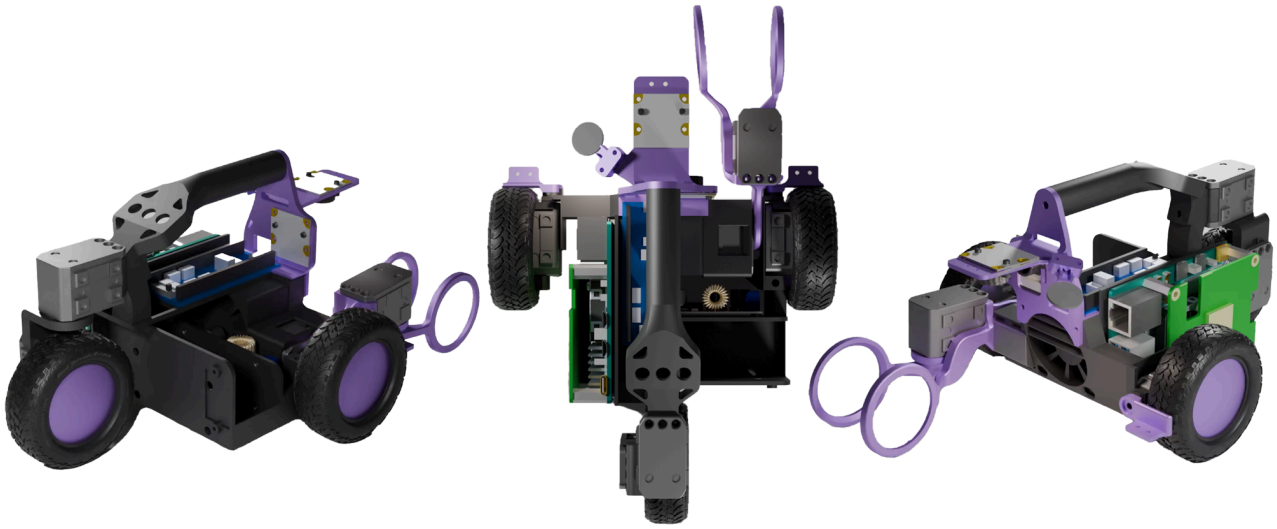
3. Hardware

Since all past generations of STEN have had structural parts based on 3D printing, this generation also follows that concept. The custom design allows for precise integration of components and simplifies maintenance. STEN 11 uses a three-wheeled drivetrain with two driven front wheels and one driven rear swerve wheel. This setup improves turning precision and maneuverability compared to previous omni-wheel designs. The chassis is compact and designed to maintain a low center of gravity while still keeping a small footprint. The robot includes two wide-angle cameras: one downward-facing for line following and one front-facing for detecting victims. Both are connected to a Raspberry Pi 5, which handles image processing and AI inference using a 26 TOPS AI accelerator. Motor control is managed by an OpenCM9.04 board connected to Dynamixel XL-320 continuous servos. Power is supplied by a 3-cell LiPo battery, with voltage regulation for the Pi and OpenCM. Furthermore, the robot includes a small built-in server fan to clear debris, along with a LED light and claw to improve visibility and complete the rescue zone.

a. Mechanical Design and Manufacturing

Provided our sizeconstraint the construction of STEN 11 is compact and highly thought out. All structural parts are 3D printed from PLA in order to allow rapid manufacturing and the ability to create a complex and tightly fitting skeleton around each component. Furthermore, the main structure is built with both ease of disassembly and cable management in mind. This includes among other things the use of heated inserts, and thurow design considerations to limit the amount of cables present. For space efficiency we have also modified a set of lego-wheels allowing us, not only to conceal our motors partly inside the wheel, but also the ability to build in custom made metal weights in the bottom of our tires lowering our center of mass. Our powertrain is thus quite compact, consisting of three driven lego 62.4 x 20 S tires, one of which being a powered swerve wheel capable of steering our robot with great control. For driving we use Dynamixel xl320 continuous servos, chosen for their small size, easy wiring, and provided feedback loop. These servos are also used to control the swerve wheel, and actuate our robotic arm, giving us a total of six dynamixel motors. The rescue mechanism is limited to only a tiny but robust claw mounted in the front of our robot and actuated by two dynamixel servos. Despite our compact design we still managed to fit a tiny server fan in the center of our robot, and a compact but powerful raspberry pi assembly. This assembly consists of a raspberry pi 5 as the base. We have then attached a heatsink to the Pi and stacked both the raspberry pi AI-hat and power-hat on top. See full cad design, including our unique powertrain and claw mechanism in figure 2.

Figure 2: Robot design in CAD



b. Electronic Design and Manufacturing

STEN 11 is built with three different voltage-levels. 11,1 volts is fed directly from our 3s lipo battery to our axial fan, and LED light. Furthermore the battery voltage is stepped down in two different places. Our OpenCM9.04 can run on a wide range of voltages, but in order to allow that board the ability to power our Dynamixel xl320 servos the board's input voltage needs to be stepped down to 7,2 V. This is done via an adjustable buck converter, the XL4015. We have previously used a second identical XL4015 to supply the required 5 volts for our raspberry pi, but after a multitude of problems related to our pi constantly warning about “low voltage”, and a few instances when the board unexpectedly shutdowned we decided to use a dedicated power hat for the pi. We have generally tried to keep the electrical system as simple as possible, utilizing only cameras and microswitches to sense our surroundings, and dynamixel motors for driving. This limits the amount of required wiring, and the space needed for all sensors and components. For a better understanding of this system refer to figure 1.

4. Software

The programming language used is Python 3.11 to support libraries like OpenCV, NumPy, and Ultralytics. A PID algorithm is used during simple line following whilst all more advanced tasks are performed via different object detection models. To keep the electronic and software design simple we run all processing on the Raspberry Pi 5, only outsourcing motor-control to the OpenCM9.04.

The software of STEN 11 is built around a dual-camera system to ensure that there is never a lack of information. During line following we utilice the downward facing camera to constantly run a custom made object detection model. This model can identify both sharp turns, X/T-intersections and green direction-markers, as well as the evacuation-zone entry. If no special object is detected then the robot drives according to a simple PID-based algorithm (utilizing only the proportional and integral factors). Any object detection too close to the robot will however trigger a sequence of instructions. Using a predefined sequence allows us to clear an intersection or turn in one motion with information based on a frame where the whole obstacle is visible. When approaching a turn or intersection we therefore

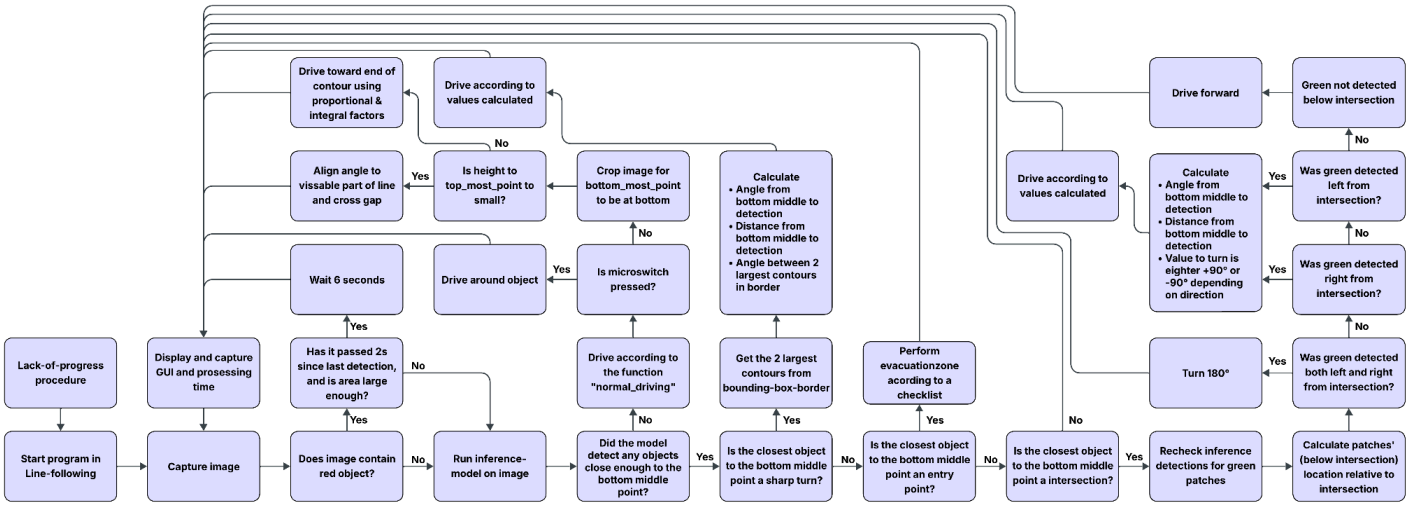
calculate, using trigonometry, the angle to the object, distance to object, and angle required to turn when over the object.

When in the evacuation zone, our system utilizes only the front facing camera and a second custom made detection model to identify victims, evacuation points, and entry/exit points. This allows us to drive straight to and from any object in the evacuation zone.

a. General software architecture

To make the structure of our software easy to understand and make the debugging process easier, the code is mostly built around a few fundamental functions. The code consists of one main function that is called upon boot. The main function calls other functions that complete different tasks such as displaying our GUI or turning on the spot. During line following we utilize a simplified PID algorithm for easy tasks, and custom trained AI-models for the more advanced tasks. A large overview of our robots driving behavior during line following is outlined in the flowchart in figure 3. When entering the evacuation zone STEN drives according to a checklist of objects to visit and act at. This method of clearing the evaluation zone is highlighted more thoroughly in section B.

Figure 3: Flowchart line following



To enable our fast paced development process we have found a multitude of tools and methods to speed up our workflow. We have also tried creating broader functions that can transcend multiple generations of our robot. All code for the Raspberry pi is written in Visual studio code as opposed to locally on the Raspberry pi. This workflow gives us a large amount of advantages. The coding experience gets a lot easier in a faster IDE with easier and smoother access to valuable tools like google, stackoverflow, and github coopilot. Working on our personal computers also allows us the ability to review and develop software without access to the physical robot, and ensures that any backups we take of the code gets stored in the cloud rather than having them stored on the Raspberry pi itself. Coding the OpenCM9.04 is more of a hassle, recruiting specific versions of Arduino IDE depending on our operating system.

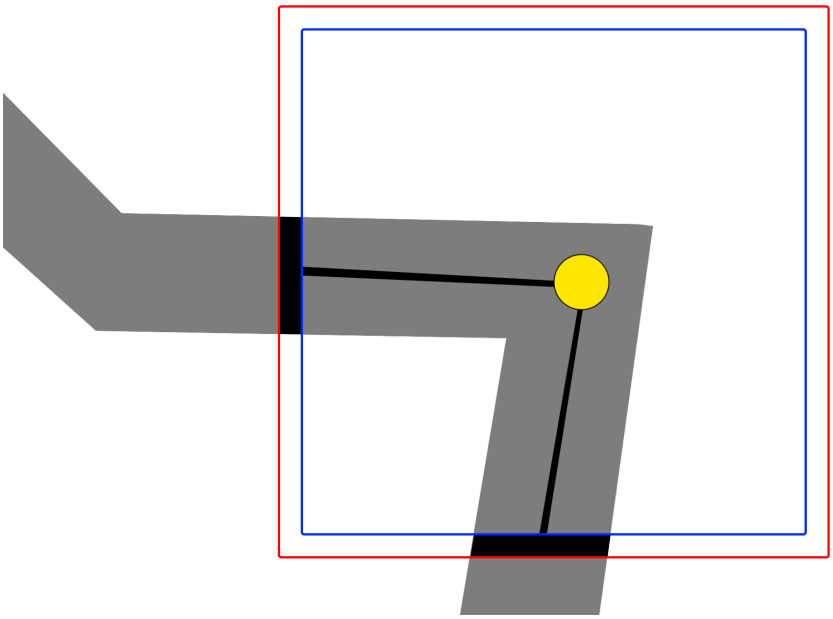
The OpenCM9.04 is programmed to act as an advanced motor controller. It has the ability to receive two different types of messages via rx/tx from the Raspberry pi. When receiving three values it calculates the

swerve wheel angle and all wheel-speeds in order to drive with a desired speed along some determined radius. If the OpenCM receives only two values, this indicates an instruction to perform some special sequence of actions. This sequence could for example be to pick up a ball, or turn a desired amount on the spot.

b. Innovative solutions

Usage of a "Bounding-box-border": To calculate the desired angle and direction to turn when on top of a sharp turn we crop and modify the image to only show a thin border around the detected 90° turn. This method gives us two separate contours illustrated in black in figure 4. Using this information we can determine what contour is the entry and exit from the turn, as well as what amount we should turn.

Figure 4: Illustration of bounding-box

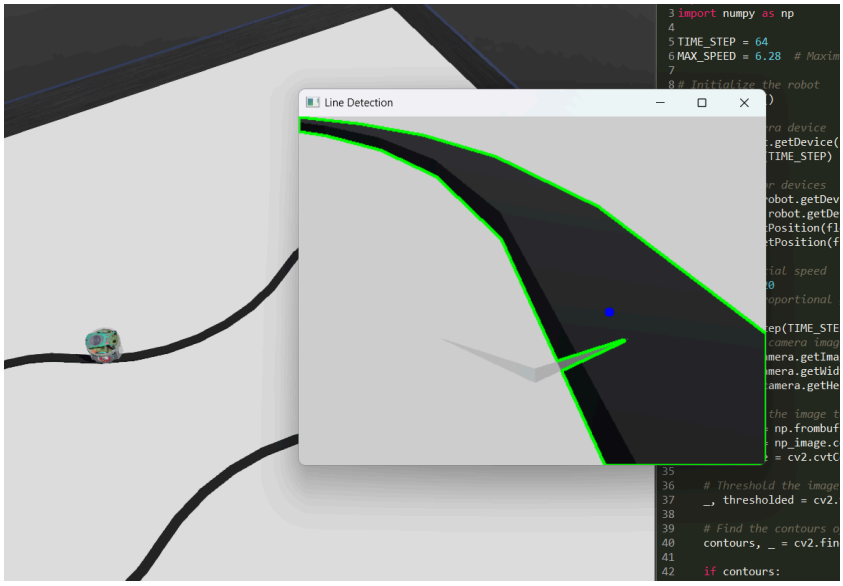


Custom trained AI models: Being early adopters of the newly released Raspberry Pi AI hat+ capable of 26 TOPS meant we had to explore this board's possibilities ourself. During our testing we experimented both with object detection models capable of identifying interesting features on the track, and image segmentation models capable of distinguishing the line with great accuracy despite glare and debris. Despite this we settled on only using the detection model to save on performance, a decision that allows us to achieve 200 fps running inference on blank images, and about 20 fps in real scenarios, including the rest of our algorithms. Using custom trained AI models makes it easy for us to identify advanced features both in the evacuation zone and during line following. We have also further developed a custom workflow for creating these models. In order to create a new model we use a separate script on the raspberry pi to record while we mode it along the track, split the video into frames, annotate all frames in roboflow, train the model in google collab using YoloV8, and convert it from ".onnx" to ".hef" in a docker container running via anydesk on a custom built PC. This process of creating a whole new model fully accustomed to new lightning conditions takes just over one hour if done correctly with a small dataset.

Checklist in evacuation Zone: For the purpose of simplification, our raspberry pi regards all objects in the evacuation zone in the same way. At the start we declare a checklist of objects to visit and act at. This list simply states the order of all objects to visit in the evacuation zone: ["Live victim", "Green", "Live victim", "Green", "Dead victim", "Red", "Exit"]. Using this list we can drive toward the currently desired object, perform a sequence of commands to either pick up or release a victim, or center itself on the exit. This is possible because we use the same detection model to detect victims as we use to detect the exit from the evacuation zone. For the exit we actually detect the rims around the exit, and therefore need to center our robot before exiting.

Testing in simulated environments: The ability to test, practice, and develop our robot in simulated environments is something we've researched and tested multiple times. Our robot has the ability to record and save the camera feed, sensordata, and decision-history from any live run, allowing us to rewatch and identify problems. Previous iterations of STEN have also had the ability to run a simulated run on the recorded data and see how our adjustments affect the robots' decisions. Despite this robot not possessing that ability, it is still built on insights gathered from that simulated environment. Furthermore, the introduction of AI in our robot has led us to investigate the possibility of running and training STEN on a fully simulated track using software like CoppeliaSim or Isaac Sim. We have not fully explored this possibility but did achieve a proof of concept that future teams may be able to build upon. This proof of concept is shown in figure 5.

Figure 5: Simulation



5. Performance evaluation

Following the construction of our own robot-course, highlighted in table 1 under section 2a, we have been running many test runs ensuring the functionality of STEN. During each test run we record and save all relevant data noting every mistake on a physical whiteboard. We have also implemented a GUI with the ability to record and playback every decision made by the robot. With this system we always have a list of possible areas of improvement. All fundamental testing has either been done at home on our own redamentry course or, following our victory in the district championship, on the course located at the Science Center organizing the local competition. Our main testing of both the robot and team structure has however occurred in live situations when attending the competitions leading up to the World Championship 2025.

As mentioned earlier we are now competing for our third year and have attended both the district championship and Swedish championship thrice, along with the world championship once. During these competitions we have learned from our own mistakes and made many changes in our development. These changes are highlighted in Table 3.

Table 3: Impact on development by previous testing

Competition	Mistake	Lesson learned
District 2023	Filled Lego spike storage fully.	Stopped using lego despite the simplicity.
Sweden 2023	Used Nema 17 stepper motors that was too weak for the 26° ramp	Train on exaggerated obstacles. Always bring tools/machines for quick fixes. Add planetary gearbox to motors.
District 2024	Raspberry pi 3 got corrupted the day before competition	Always have a backup (hence our continued development of STEN 8). Spare parts, backupped code, and emergency robot
Sweden 2024	Faulty color sensors, and lacking strength for slopes. Strings were also used as debris (tangles into wheel-hub)	Quit being cheap. Buy better sensors, motors, and better quality wire. Consider wire management and simplicity. Consider more than two driven wheels. Be informed on the rules.
World 2024	Coded based on failsafes and “happy accidents” instead of planned and informed decisions.	Use a camera to increase available data. Reduce dependency on C++/arduino IDE by switching to Python/VS code. More leniency with our size constraint
Sweden 2025	Stayed up developing new AI-models and 3d printing new parts throughout the night	Come as prepared as possible. The robot can always be improved, no matter what.

6. Conclusion

After three years of continuous development, redesign, and improvement we can finally say that we have a robot capable of competing at the world championship. From the very beginning, we have always encouraged rapid prototyping and testing new concepts while learning from our mistakes. The knowledge and experience we’ve gained over the past two years have allowed us to build and test a robot that can now compete on an international level. By optimizing teamwork and collaboration, we have also accelerated our development process. Last year’s robot struggled with both under dimensioned ground clearance and sensor restrictions that impacted its performance, but we have learned from those challenges. This time, we focused on making the robot far more reliable by completely replacing the sensor system and making the design a lot more reliable in situations far beyond the game rules. We are excited to compete in the World Championship, where we will not only challenge our robot on more demanding courses but also share knowledge and ideas with teams from around the world.

References

a. Software tools and platforms

- OpenCM 9.04
- XL-320
- Raspberry Pi 5 GPIO Pinout
- Roboflow
- Convert ONNX Model to HEF | Dylan Tintenfich

b. Libraries

- OpenCV - Open Computer Vision Library
- NumPy

c. Hardware

- Raspberry Pi 5 8GB
- Raspberry Pi AI Hat 26 TOPS
- Active cooler for Raspberry Pi 5
- Raspberry Pi cameramodule 3
- Power Supply HAT for Raspberry Pi 5 - 6-36V 5A
- LED-lamp 1.5W G4
- SanDisk 64GB microSDXC card
- Heated inserts
- 24 AWG wire
- Microswitch
- Mosfet-module
- XT30 connector
- 3S Lipo 50C 2200 mAh
- Screws M2-M5
- XL4015
- 40x40x28 mm axial fan
- OpenCM9.04
- DYNAMIXEL XL-320
- ELEGOO PLA
- Lego wheel
- Toggle Switch
- Motion Sensor