Robocup Junior Rescue Line 2025

Team Description Paper

FusionZero

---

0 | Abstract

FusionZero, as the 3rd iteration, includes a condensed set of features that has appeared throughout its short lifetime. Our robots maintain a small footprint, taking only 187 mm by 141 mm whilst running a single actuator claw. Outstandingly, our unique solution to locating victims with a cost-effective solution takes advantage of the game field and expected behaviours. Furthermore, line following is most importantly, stable and reliable.

Our consistency is what sets us apart from other competitors: understanding that speed is not everything. Playing to the point system, touches should be avoided at all cost. Absolute data from a multitude of sensors ensures we have perfect control over any situation, thus decreasing the chance of incurring a lack of progress deduction. From custom PCBs with an IR sensor array, to a shield for the Raspberry Pi Zero 2W, to integrated camera vision, we believe we have an alternative to the heavy AI inferencing approach, retaining core principles of solving problems with the least complexity.

1 | Introduction

A | Team

FusionZero is a team previously established in Auckland, New Zealand. Having competed under iBot Academy, this is the first year of full independence. Upon completing the regional events, the international ruleset provides much more difficulty, greatly expanding the skillsets of both members.

Frederick Sun (team leader) has experience in a multitude of events. During these years, he has always worked on the overall design of the robot, as well as solving the chemical spill with efficiency and accuracy. Thus, leading up to this event, he has used Onshape of CAD for the robot, simulated claw linkages with Linkage, and 3d printing. In addition, he introduced the vision-based solution to victim identification, and provided the basic approach to the evacuation-zone.

- [Regionals] 2018 Auckland Secondary, 3rd
- [Regionals] 2018 Auckland Premier, 2nd
- [Nationals] 2018 New Zealand Secondary, N/A
- [Nationals] 2018 New Zealand Premier, 2nd
- [Nationals] 2019 Australia Open Tertiary, 3rd
- [Nationals] 2024 Singapore U19 International, Judges Award, 3rd Internationally
- [Nationals] 2024 Australia U19 International, 2nd

Aidan Zhu became involved in robotics through after school lessons, and began competing in Robocup when he was 11 years old. During his early years he was involved in line following, “chemical spill”, and design aspects. Later when he joined with Frederick he became more focused on line following, whilst assisting in evacuation zone testing. Since then he has developed skills in PCB design for controllers and sensor arrays.

- [Nationals] 2021, New Zealand Self Driving Car Senior, 1st
- [Regionals] 2022, Auckland Premier, 2nd
- [Nationals] 2023, New Zealand Premier, 1st
- [Nationals] 2023, Australia Open Tertiary, Top 5
- [Nationals] 2024 Singapore U19 International, Judges Award, 3rd Internationally
- [Nationals] 2024 Australia U19 International, 2nd

2 | Project Planning

A | Overall Project Plan

We must ensure we have a clear understanding of the ruleset, a problem that cost us in the 2024 Robocup Singapore Open and in the 2024 Robocup Australia Open. Here are all the key points that we will work on:

The team captain may make further attempts at the course to earn additional points from scoring elements that have not already been earned before reaching the next checkpoint.
We did not understand that we could continue going to points for other scoring elements, not just tile, during Robocup Singapore. We must continue trying to get points for other elements.

Time limit of 8 minutes ...
We should aim to use all 8 minutes, and leave parts of the course unfinished. This is a better approach compared to finishing early, but having LoP/skipped parts of the course, guaranteeing lost points.

The line along the ramps can contain gaps, speed bumps, intersections, obstacles and debris [...] exit [...] entrance [...]
Last time we lost focus on all the small details, and these components were rushed and therefore done extremely poorly. We incorrectly assumed that our robot would simply “handle” these “minor” challenges, which really took us by surprise.

B | Timeline

December 2024

1. Consolidate what went wrong and consult changes that are required. Begin designing new components to solve the issues. Define the constraints that the new robot must have.

January 14th 2025 - February 1st 2025

1. Execute exhaustive testing for line-follow and evacuation-zone respectively. This ensures that all components for each section are properly accounted for, and work in isolation. This means that all variations are accounted

Gather new ideas, any new methods or designs that we should incorporate into the new iterations.
2. Produce a document that identifies all the errors, regions for improvement, and provide the appropriate solutions.

December 2024 - January 2025

1. Individually expand on the ideas and methods gathered, allowing for a more streamlined process, as rapid application development may take place.
2. Finalise the new component, and then test, assuming perfect conditions, to gather data and results.
3. Compare and contrast the new idea to our current robot? Define which is better through measuring different metrics.

January 1st 2025- January 14th 2025

1. Create modules of each component, and perform white-box testing where all modules are tested in isolation to one another, hence we assume perfect input.
2. This aims to bridge the gap between hardware and software: we want all sensors, actuators, to code, and be able to interact with the robot.
3. Frederick will work on the CAD, whilst Aidan will work on creating code snippets for interacting with the various components.

for, so nothing may surprise us during the competition.
2. Frederick will work on structuring the basic evacuation-zone code structure, with a focus on victim identification. Aidan will develop the overall code structure, including the line-follow, calibration of sensors, intersections.

February 2025

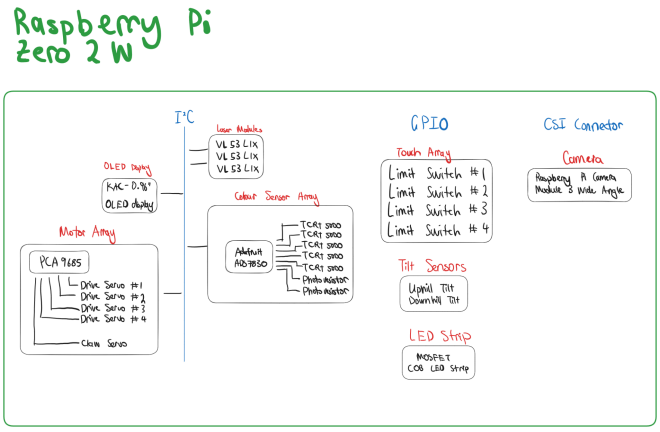
1. Combine both sections, such that full runs are possible. Complete black box testing, and then review afterwards. This way we can identify recurring problems in a batch that have not occurred to untimely situations, but a persistent problem with the robot design or code.
2. Aidan will perform full runs more, we both assume a role as a judge, trying to score the boards live. This way we can learn how scoring works in depth, thus deepening our gameplan. Frederick will begin finalising the designs, and writing up documentation.

March 2025

1. Travel to Japan, finalise documents, finalise robot design and 3d print backup sets for parts in case of accidents.

2 | Integration

These should be all the components and sensors that are required to complete all major tasks.

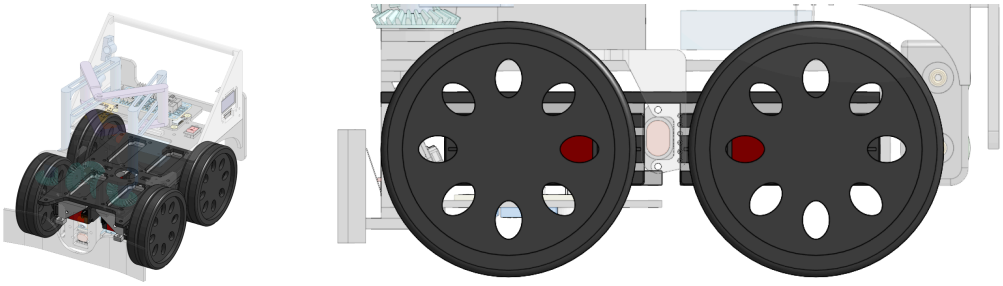


Basic overview of all components and how they are connected

Line following will be completed with a custom PCB, servo control (drivetrain + claw) with the PCA9685, capable of generating PWM signals, taking input via the I2C channel. For detecting ramps, we have tilt sensors. For distance measuring in tasks such as obstacle routing, we have lasers to aid with the movement. Victim identification may utilize the camera. Finally, we can have real time information about the program through the OLED display, which will be helpful for debugging.

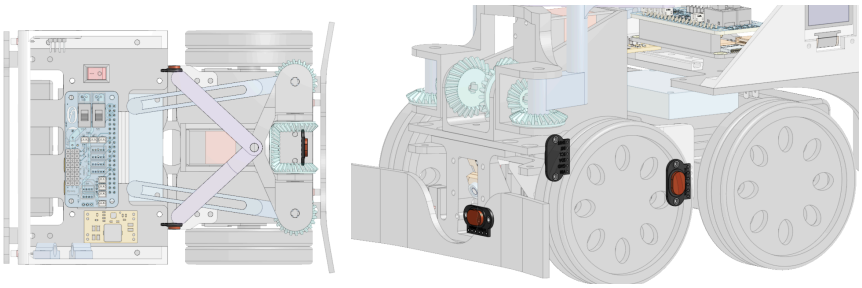
Our main processor will be a Raspberry Pi Zero 2W, as this provides a strong CPU and a CSI connector, whilst being able to run numpy, opencv2 for basic image operations, whilst having the smallest current draw and footprint. All the components listed above will be connected through a custom shield, featuring JST connectors for secure connections.

Movement will be controlled through the PCA9685, with the servos positioned such that the minimal gap was left for the ToF sensor, as to minimise space.



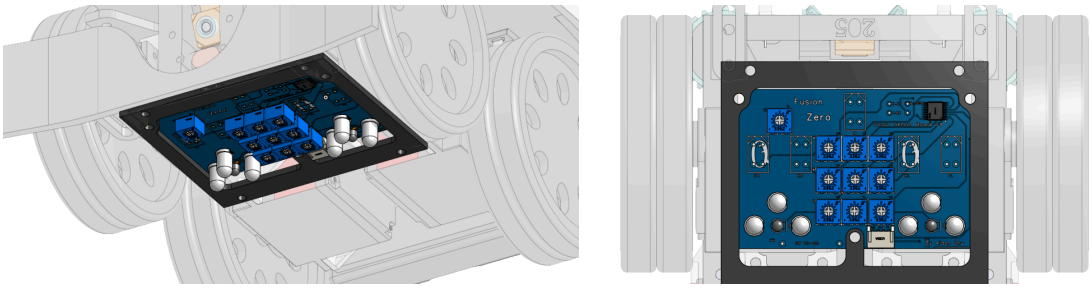
Drivetrain servo positioning

For navigation within the evacuation-zone, and whilst going around obstacles, we use 3 x ToF sensors. These are placed in the front and the sides such that we have a lot of information for where the robot is, in relation to objects in the environment. These sensors use I2C for communication, and we automatically reassign I2C addresses via the use of the XSHUT pins.



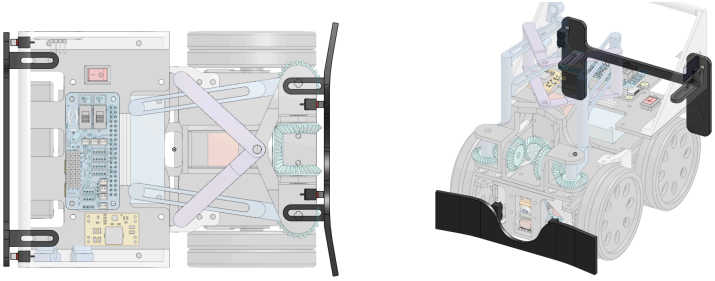
ToF positions (left, front, right)

For line following, we have positioned TCRT5000s and photoresistors in line with the centre of rotation for the first set of drive servos. This ensures the line we are following is accurately reflected with the centre of rotation of the robot. These are then connected to an ADS7830, which converts 8 analog inputs into an I2C data stream, which can also be read by the RPi.



Colour Sensor Array positioned beneath the drive servos

Our touch sensors detect collisions with walls and obstacles. These are absolute sensors, and would allow us to detect collisions over a wide area as opposed to a small area offered by the ToF sensors. We have placed them at both the front and the back, such that the most common collision spaces are monitored. These sensors may be read through a simple connection to a GPIO pin and ground, provided the GPIO pin is set to a pullup resistor.

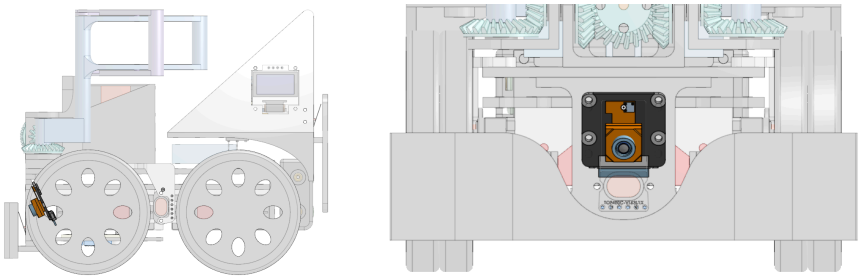


Touch sensors (front, back)

The Pi Camera Module 3 is positioned in such a way that the images that it takes naturally cut off the top section of the evacuation zone (10cm minimum walls). Furthermore, this angle allowed us to try line follow (partially successful). Furthermore, the camera must be able to see the entirety of the evacuation-zone, as our search pattern is random. The camera and the Raspberry Pi communicate through Raspberry Pi's inbuilt CSI cable, with libcamera2.

Live victims are found through using the repeatable environment to our advantage. There are always going to be studio lights above the field, where due to the reflective nature of the silver balls, creates a bright "spectral highlight" in the top half of the victim.

Dead victims are found through filtering for low values in the HSV colour space, finding all contours, and then validating through running a circularity check and size check.



Pi Camera Module 3 Wide Angle, angled downwards

3 | Hardware

Our primary design focus is keeping the robot as lightweight as possible. We should aim to use the least complex mechanisms to solve any problem, thus removing the uncertainty as we should have much more control over what happens.

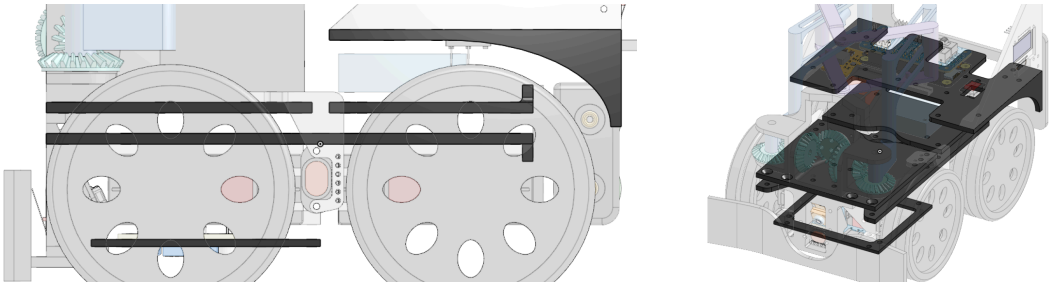
Our robot is different; as opposed to using DC motors, we have opted for using 360 degree continuous servos as our drive train. Furthermore, while most teams opt for a dual actuator claw for victim rescue, we have reduced it into a single actuator, freeing up space. Our power source comes from 2 x 18650 LiPo batteries in series, outputting 8.4V. Our servos all operate at 6V, and may draw up to 1A per servo, for a total maximum of 4A. We obtained a DC-DC buck converter rated at 5A, stepping down the 8.4V into a safe 6V for the motors. Furthermore, the Raspberry Pi requires 5V 0.5A, hence we have a Mini650 Pro, converting 8.4V to 5V.

Our claw actuator works because of a linkage developed to minimise space, whilst maintaining strength, and utilizes a 3 bar mechanism for gripping victims, more secure than a simple gear claw.

All components of the robot are printed with PLA plastic. Despite PLA's reputation as a weak plastic, we think it is suitable for our task - nothing should experience a force to compress or bend, nor will the environment exceed 60 degrees celsius.

A | Mechanical Design

Our robot is composed of "bases", wherein each base serves a different purpose. This allows for a multi-layered, modular robot design that requires less printing during the development stage. These are the building blocks, to which other components are connected to, providing a sturdy anchor point. Furthermore, these act as channels for wires to run through, allowing for a much easier time managing wires, as nothing gets in the way, and may always be hidden.



"Base" boards used to mount other components | Printed at 3 mm thickness | 1 mm for colour sensor module

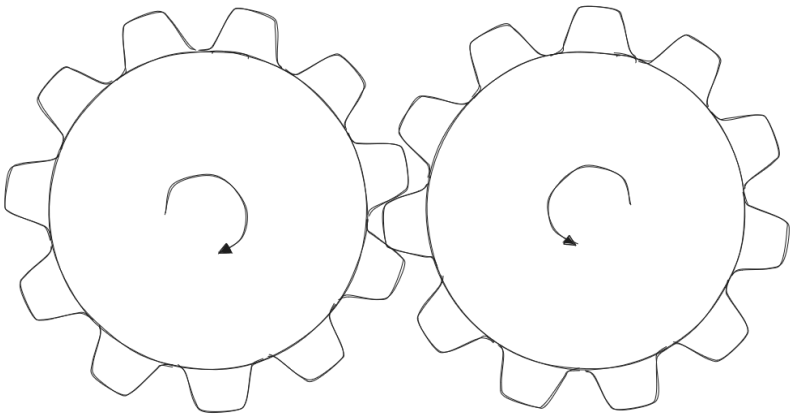
Spacing between each layer is done with a set of standard spacers, of 6mm and 18mm. They are placed 5 mm away from the edges, allowing for modular designs for this base frame.

Placed on the bases, we have the components for the powertrain. All our power stems from two 18650 Lithium ion batteries, outputting 8.4V, converted into 5V and 6V using appropriate step-down converters, that power the Raspberry Pi and all servos respectively. The raspberry pi is connected to an PCA9685 via the I2C protocol, which drives the servos with various PWM frequencies. This system is reliable, as the buck converter is rated at 5A maximum current flow, and from testing, our servos may draw up to 1A each, perfectly matching the rated current draw.

Refer to the power subsystem [link].

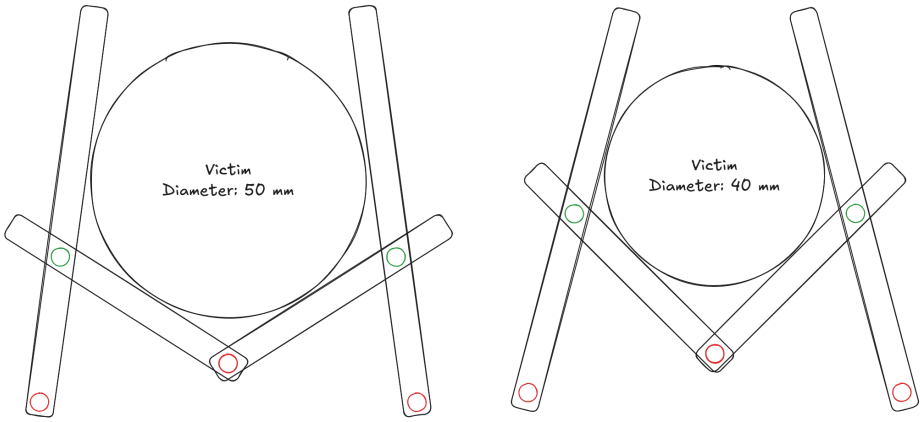
Our compact rescue mechanism employs a singular actuator, integrating an efficient system of linkages, oblique gears, and optimized gear ratios. This innovative design significantly reduces spatial requirements while maintaining operational efficacy.

The rescue mechanism may be split into two distinct motions: the first being securing the victim. For this section, as opposed to using a simple dual gear setup, such as:



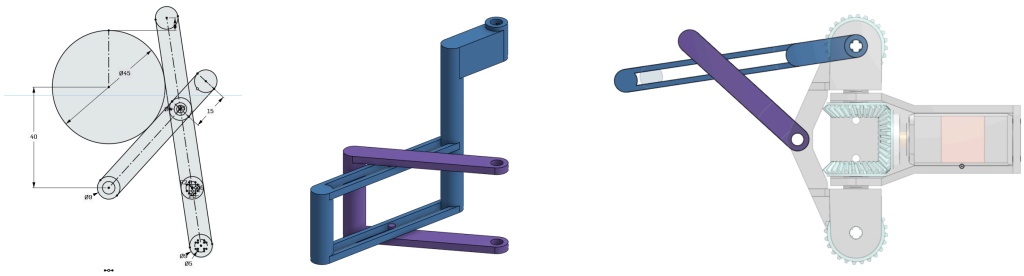
A simple two gear setup rotating in opposite directions

We chose not to as the ball size may vary. Recall the rule: *a victim represents a person and is in the form of a 4-5 cm diameter sphere [...].* Thus, this rule renders claws of this style impractical as they always have a set closing distance and motion. Hence, we decided to use a 3 bar linkage as our claw. The major benefit is that this has an adaptable shape, depending on the size of the victim, the claw should always maintain 4 points of contact with the victim.



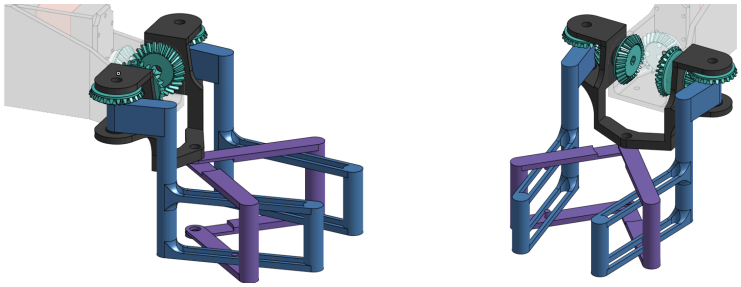
3 Bar Linkage | RED are grounded pivot points | GREEN is a slider between the linkages | All lengths remain the same

To verify if this idea works, we used two simulation software. Firstly, Linkage, then built into an onshape sketch with loose constraints such that it also works in CAD. When transforming into a 3D model, multiple test claws were constructed demonstrating this idea in action, each stage reducing its size and complexity, finally arriving at a small gear transmission mechanism.



From linkage to CAD design | Shows half the completed claw structure

Moving onto the second part of this mechanism: lifting the victim above the 6cm required for dumping. This may be simply achieved by rotating the entire system, if it were all attached by an axle. In our design, this is achieved through a piece known as the “rotating frame”, where the entire 3 bar linkage resides.



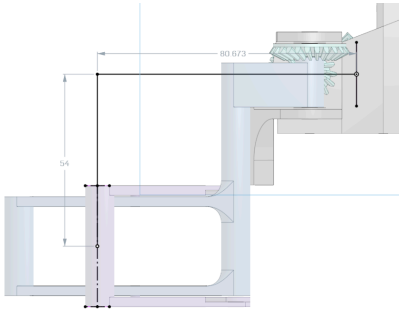
“Rotating frame” attached to the 3 bar linkage

Finally, the last problem to solve is only using a single actuator to run this entire system. The order of operations whilst rescuing should be grab then lift. When depositing the victim, it should be in reverse - drop then open. Our mechanism supports both this through using a passively actuated 3 bar linkage, where due to gravity, it takes less energy to actuate the 3 bar linkage instead of lifting the 3 bar mechanism. However, when the 3 bar linkage cannot move anymore (when the victim is securely grabbed or the claw pushes against itself), then the linkage moves upwards, as that’s the only degree of freedom remaining.

Conversely, when depositing victims, less energy is required to lower the claw than to open the claw, thus the claw moves downwards until it no longer can, and then releases the victim in a controlled manner into the rescue points.

We consider this to be an innovative mechanical design, as it takes two different motions and combines it into one. The major benefit of this is that it saves space, and the strong approach to securely grabbing victims.

Thorough testing was conducted about the strength, regarding this line of the rules: *[...] off-center center of mass and a maximum weight of 80 g.* By simply calculating the torque required and with real tests, we can confirm any design.



$$\text{Manufacter stall torque for 6V} = 20\text{kg} \cdot \text{cm}$$

$$\begin{aligned} \tau_{servo} &= 20 \cdot 9.81 \cdot 1 \cdot 10^{-2} \\ &= 1.962 \text{ Nm} \end{aligned}$$

$$\begin{aligned} \tau_{required} &= mg \cdot d_{\perp} \\ &= 80 \cdot 10^{-3} \times 9.81 \times 80.673 \cdot 10^{-3} \\ &= 0.633121704 \text{ Nm} \end{aligned}$$

$$\tau_{servo} > \tau_{required}$$

Calculations demonstrating exceeding torque

B | Electronic Design and Manufacturing

Choosing the correct computing hardware to use in our system to control all our peripherals is critical to success. Considering the previous constraints that we have applied:

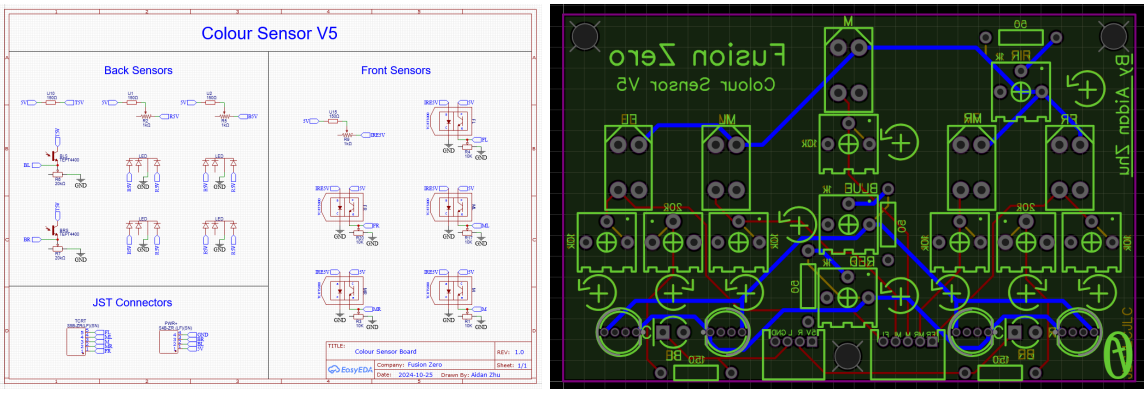
Hardware	Arduino Portenta H7	Google Coral Dev	Google Coral Dev Mini	Google Coral Dev Micro	Pi 5 + Microcontroller	Raspberry Pi 5	Raspberry Pi 4 B // B+	Raspberry Pi Zero 2W	OpenMV
Processing Speed	1	9	8	7	6	5	4	3	2
I/O	2	9	9	9	9	9	9	9	1
Power	9	1	2	5	3	4	6	7	8
Size	9	1	3	6	1	5	1	7	9
Information	2	4	3	1	9	9	9	9	5
Total	23	24	25	28	28	32	29	35	25

The raspberry pi zero 2W has all the capabilities to run I2C modules, GPIO, and most importantly a camera. Whilst its computing power is not the best, it is more than capable of running OpenCV at low cost functions, which our victim identification is based around.

There are a variety of sensors that we use, all combine to create a smooth experience. Some of these sensors provide more information than others, but have less reliability. A high level overview of the sensors may be summarized in the table underneath:

Sensor Type		**Output**	**Information Level**	**Count**
IR Sensor Array	TCRT5000	Analog	Absolute	5
	RGB LED	N/A	Absolute	4
	Photoresistors	Analog	Absolute	2
	PCA7830	I2C	Absolute	1
ToF sensors	VL53L1X	I2C	Relative	3
Switches	Limit switches	Digital	Relative	4
	Mercury beaded	Digital	Absolute	2

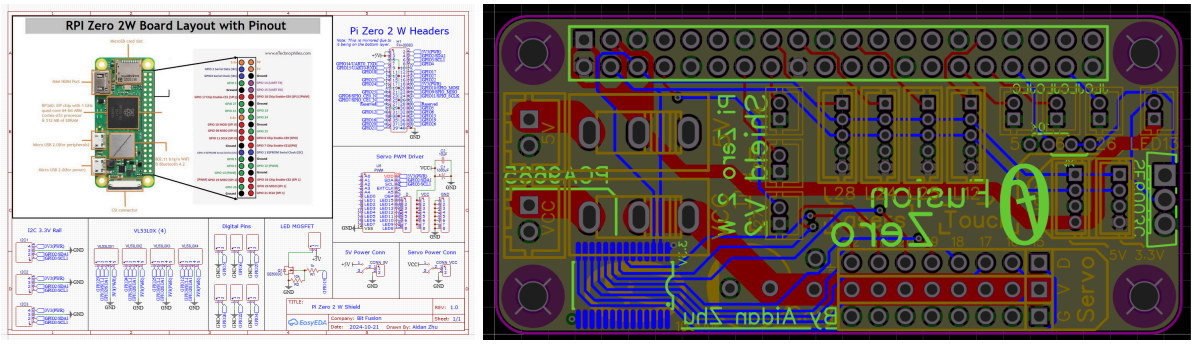
Our power subsystem involves two voltage converters, used to power different things. Use reference [1] as a guide. As many other top teams, we have utilized custom PCBs across the robot to ensure we have total control in an orderly, compact manner. We have developed shields for both the Raspberry Pi Zero 2W, as well as our most complex sensor: the IR Sensor Array.



PCB Schematic | PCB design and routing

For line following, we are using standard TCRT5000s. These work exceptionally well with identifying the difference between white and black, but not so much green and other colours. Thus we use photoresistors detecting visible light for intersection parsing, in a controlled environment with RGB LEDs. This combination uses the benefits from two separate methods to handle line following with accuracy.

From previous experiences, we used to hardwire connections with solder (singapore), then we switched to dupont (australia). However, those were still unstable, so this time we upgraded further, using JST connections that are one way, and have a click and can't come loose. Thus we created a custom shield for the raspberry pi to support all our peripherals.



Raspberry Pi Zero 2W custom shield

Take a look at 7 | Reference for the power supply diagram.

4 | Software

The Pi runs a legacy 32-bit Raspberry Pi OS Lite. This is due to reaching performance issues with CV2 and stability issues due to high loads. This is achieved through using the raspberry pi image flasher.

Our software is fully coded with python, locally on the Pi Zero 2W. Through the VSCode Remote extension and setting up a VSCode remote SSH server, we are able to code on the Pi's system via the command line, while maintaining the basic VSCode structure. Through community libraries, we are able to program high-level logic instead of being bogged down with interfacing with various sensors.

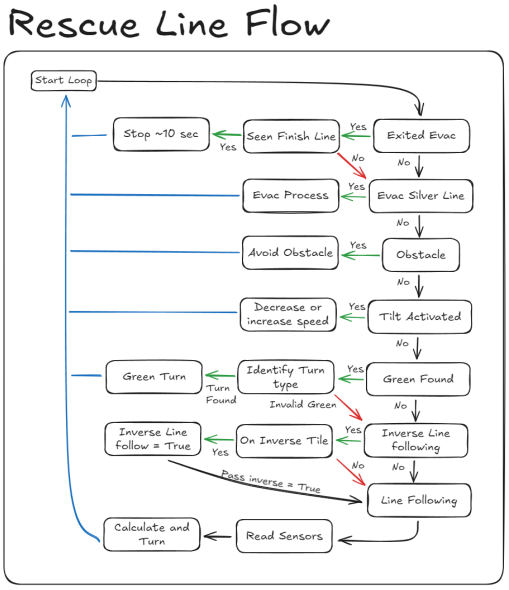
A | General Software Architecture

On the Pi, our code may be separated into the different components, combined together into either line following, evacuation zone, and user interface. Firstly, we have a file simply containing all the configuration variables and preferences, referenced by other files in a program. We include things such as: if X11 forwarding is enabled (impacts performance), image resolution, time spent in the evacuation zone, and victims found (for debugging).

When running the main program, whilst in development, we have a menu that has various modes available to select from, depending on what we wish to achieve. Modes that are included are: calibration, sensor reading, evacuation zone only, line follow only, and run (everything).

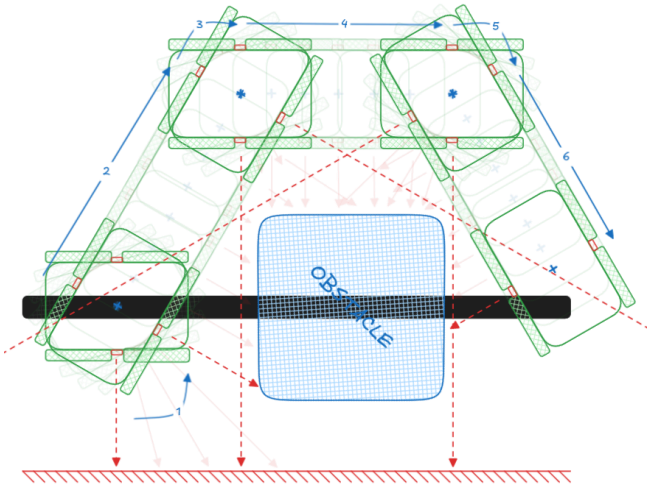
The main loop begins with reading the IR sensor array, as well as the touch states. We next perform colour checks for red and silver. This is done through incrementing a loop counter, such that after a certain number of passes in a loop (around ~100ms), the colour is confirmed and not seen by accident.

Afterwards, we check the touch states for collisions with obstacles. When the states change, we trigger the obstacle avoidance loop.



Flow chart for line following

1. Turn until the laser opposite of the side it turned towards sees an obstacle.
2. Moves forwards until the laser sees over the obstacle.
3. Turns opposite the direction it went in initially until the laser sees nothing.
4. Repeat steps 2 and 3 until the front sensor on the opposite side of the initial turn sees black.
5. Turn onto black line.

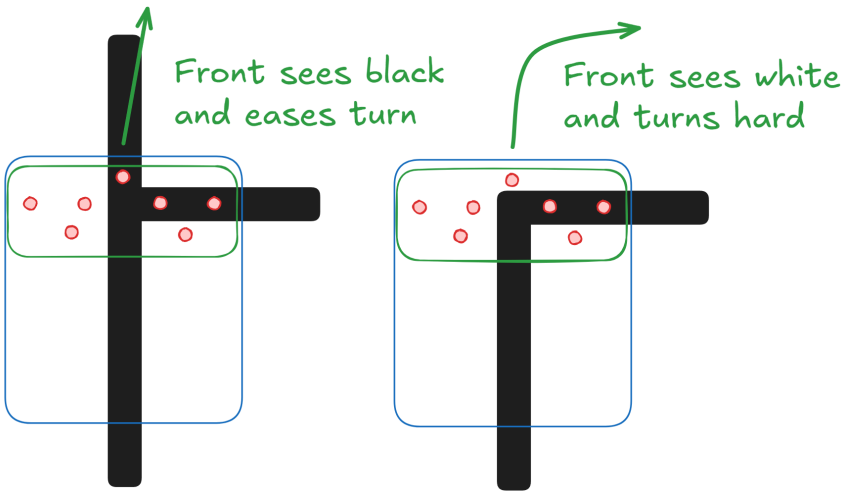


Obstacle avoidance

If there is no obstacle, then we check the tilt sensors. If a tilt sensor detects that we're going uphill, then an uphill loop counter is incremented. If the loop counter passes a threshold, after about ~100ms, then we increase the speed of the line-follower. Conversely, the same is true for the downhill tilt sensor, only this time decreasing the speed.

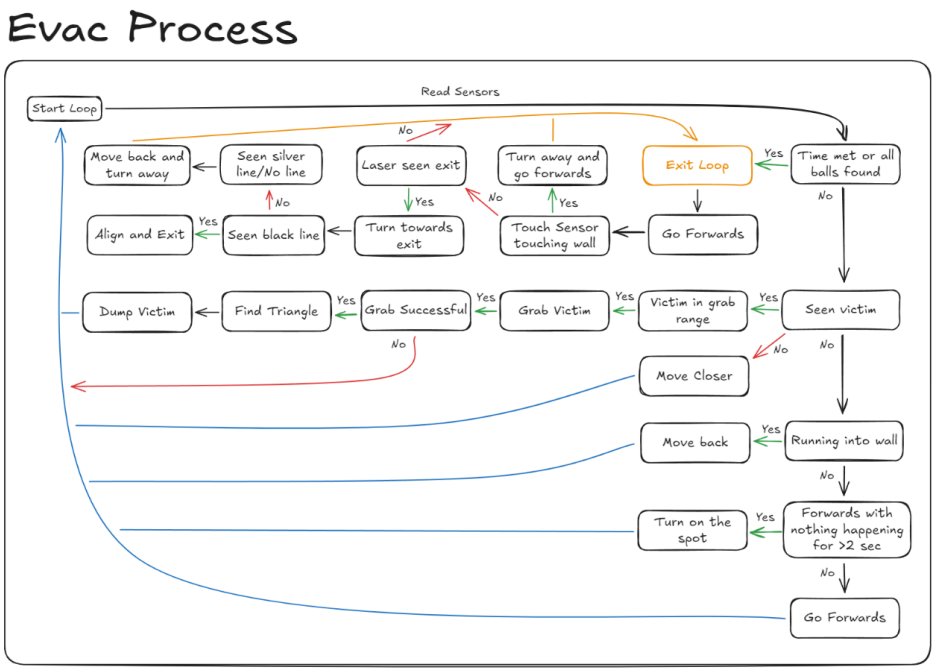
These are all the modifiers completed; checking the intersections is next in priority. This is done through checking the back sensors for green and front sensors for black, and proceeding forth. Following this, we check for inverse tiles by checking if front IR sensors read black and back IR sensors read white, thus incrementing an inverse tile count over a loop. Once this counter passes a threshold, line following directions are inverted.

Finally, when all modifications and extra processes are complete, we follow the line with a closed-loop feedback algorithm, known as PID. However, we have further developed this through a use of a middle IR sensor, creating a unique "easing" system. This allows us to control the intensity of turns: sharp or smooth. Finally the loop arrives at the line following process which uses proportional line following along with our unique "easing" multiplier. First, like any proportional line follower it calculates the error and then multiplies the proportional or "Kp" constant. After that we have a colour sensor in the middle where if it's on black it will ease the turn and reduce it, and if it's on white it will make it turn sharply.



Demonstration of the easing mechanic

When the main loop detects the entrance to the evacuation zone (silver line), we enter the evacuation zone loop. We align the robot to the silver line to ensure we are properly facing inside the evacuation zone and begin a timer.



Flow chart for evacuation zone

Firstly, the loop checks whether the number of victims found is less than 3, and whether the time spent inside the evacuation zone loop exceeds a preferred time. If this condition is met, we enter a subroutine, where we find the exit by moving forwards until we arrive at a wall, then following the wall, using a combination of the laser sensors on the sides and the IR sensors to find gaps (exit or entrance). We keep following the wall until we find the black line (exit). If the condition fails, we are able to search for victims. Depending on the number of victims found, we switch the searching function, but the proceeding steps are the same. Both search functions for live and dead victims return the x coordinate from the camera, or None if there are no victims in sight.

When a victim has been found within an image, the robot tries moving close to it, whilst aligning itself closer to the victim as well, done through a simple proportional feedback loop. After this, the front laser is used for confirmation of distances, before a grab sequence is initiated.

During this sequence, we run something known as a “presence check” that allows for us to verify that we have successfully grabbed a victim, such that when we don’t due to external factors, we still have perfect information on the game state. This check is only possible with no external sensors due to how our claw runs off a single actuator, the claw height is different depending on whether we successfully lift a victim or not. Following this, we do a fine alignment, spinning on the spot, then dropping off the victim, and incrementing our victim count variable.

B | Innovative Solutions

VSCode Remote Extension, SSH, X11 | USB OTG Ethernet Device

We chose this option as we are already severely limited in the computational power and system resources aboard the raspberry pi - VNC servers would cause too much lag as the entire desktop environment would be forwarded. Thus, only forwarding GUI applications when they are used greatly benefits our performance, and ability to code and debug.

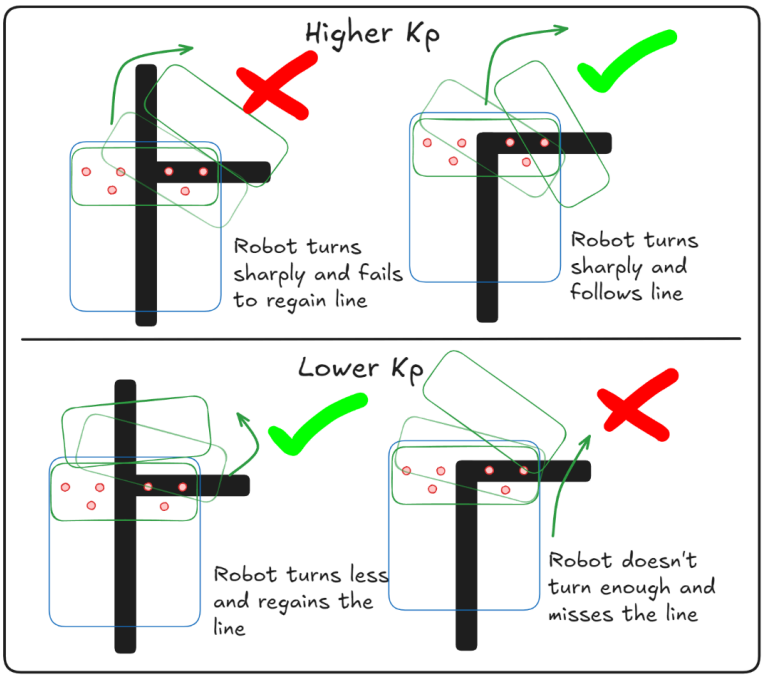
As a backup option, we have converted the Pi into a USB OTG device recognised by windows machines. This treats the raspberry pi as a local ethernet device, which may be used as a backup.

Github

Sharing code between the members, as well as the various raspberry pis that we were using is vital. A streamlined approach is to simply use github. Offering extensive versioning and history features and being streamlined for projects such as this, we embarked on learning this tool.

Using the branch features were extremely important, as it allowed us to work in parallel, without having conflicting code, especially since we were developing two robots in sync.

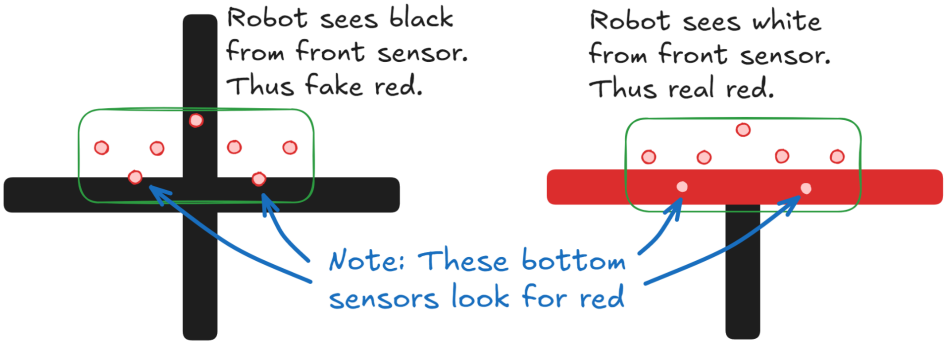
Middle IR Sensor



Demonstration of the impact of different Kp values

For as long as we've been doing robotics we've been taught to use 2 color sensors for line following. Either that or to use 1 color sensor on the side of the line. However, after placing 2nd in Australia last year we decided that we needed to seriously improve our line following. Luckily after Robocup Australia the end of year break commenced, providing us with ample time to think of possible ways to improve line following. To begin this improvement planning, we identified our weak points. The part we struggled with the most was for turns like the image to the left where our bot would turn away and veer off the course. Of course we had already tried many methods with our old colour sensor array but no algorithm would work. This is because when we have a higher kP the robot can make sharp turns just fine but it won't pass the fork in the road. Consequently we would try to decrease Kp to pass the fork but then not have enough kp for more difficult turns.

Our solution was actually found not through this problem but through another. We had an issue with red detection failing due to false red's so we realised we could fix this by checking for a black line in front of the finish line. We had the same issue with detecting fake silvers on ramps which could also be fixed by checking if there was a black line in front of the silver. By chance, this front sensor also benefited line following as it solved the fork issue as shown prior.



"Fake red" check

Claw Design

Our claw design requires a single actuator - completing both a grab and lift operation, and a drop then open action. This complexity is solved simply by a passive gearage, and a 3 bar linkage. We consider these to be largely beneficial to the accuracy and adaptability to the environment, and allow the claw to become much more robust, where consistency is always supreme compared to speed.

Victim Detection

We use the environment to our advantage, accounting for the bright spotlights up overhead that are most likely guaranteed due to the indoor nature of this competition and the need for consistent lighting conditions. Thus we are able to abuse the reflective quality of the live victims, and create a computationally low cost algorithm that is efficient and reliable.

5 | Performance Evaluation

A | Testing procedure

Line Following

When testing line following, and other assorted components, it is clear that we needed variation in our board. Thus, using PVC foam boards, we created all our boards by hand. Using electrical tape and a craft knife allowed us to create black lines of varying widths. Using green electrical tape from different brands got different shades of green, cut using the craft knife. This meant that our tiles replicated the conditions found in competition. We modelled our own ramps, seesaw, obstacles in Onshape, and 3d printed them.

Evacuation Zone

By developing a jigsaw pattern, we were able to laser cut / 3d print pieces that fit together, and provided the base for a transportable evacuation zone. Furthermore, the walls of the evacuation zone are fully 3d printed, and can be slid out, allowing for customizable entrance/exit combinations, something that we struggled with in our previous competitions.

Results

We employ an interactive development cycle, where from an initial plan, we deviate and adapt to new and unexpected problems. We were able to do this as both members of the team had an individual robot, enabled by the low cost of components, and thus allowed for synchronous development and testing.

6 | Conclusion

TO BE TESTED

7 | References