

Frederick Sun, Aidan Zhu | iBot Academy | January 1, 2024

Robocup Asia Pacific

Bit Fusion

# 1 | Research

## Problem Statement

We had made the mistake in Robocup Singapore for not reading the updated rules for 2024, and hence we lost many points due to changes in hardware that needed to be made.

The floor may be either smooth or textured (like linoleum or carpet)

The following suggests that we must be versatile in our movement method, we cannot use a method that generates too much friction, nor do we want to remove all of it. Perhaps the use of an interchangeable system between normal tires and omniwheels?

The black line, 1-2 cm wide, …

Previously, our IR sensor modules had all been designed to fit the larger end of this spectrum. We need to have a modular system that can adapt during a tournament.

The line will be 10 cm away from any edge of the field, walls, pillars to support ramps, seesaws, and obstacles that do not lie ahead of the robot’s path.

This specifies that we should aim to have the robot’s width constrained to 100 or smaller.

Speed bumps will have a height of 1 cm …

There must be at least a 10 gap between the bottom of our robot to the line following module.

Obstacles may include bricks, blocks, weights, and other large, heavy items.

There are no predetermined dimensions of the obstacle, only the height is determined to being at least 15cm, we must include sensors to adapt to any shape.

Ramps will not exceed an incline of 25 degrees from the horizontal.

Previously, we were using a tilt sensor set to trigger at 25 degrees. However, it the ramp may be less than 25 degrees, and hence would not trigger our tilt. We need to implement a more precise sensor to measure the current position and orientation in 3d space.

Seesaw is a tile that can pivot around a hinge in the centre of a regular tile …

Our robot must have a low centre of gravity to minimize tipping.

The organizers may place an obstacle inside the evacuation zone. In the evacuation zone, organizers may put the obstacle anywhere with a minimum of 10 cm clearance from the wall.

Proves we need a mapping system, in combination with a pathfinding algorithm. We currently lack sensors to achieve either of these tasks, except for the touch at the front. Or we use a different strategy as opposed to random movement.

Robots are not permitted to move on their own while calibrating.

We need to find a reliable way to record values, instead of having the robot spin on the spot.

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 the tile, during Robocup Singapore. We must continue trying to get points for other elements.

## Movement

Navigating the course requires some sort of movement. Given that the boards can be different textures, (linoleum or carpet), or system must be robust to handle these different friction levels.

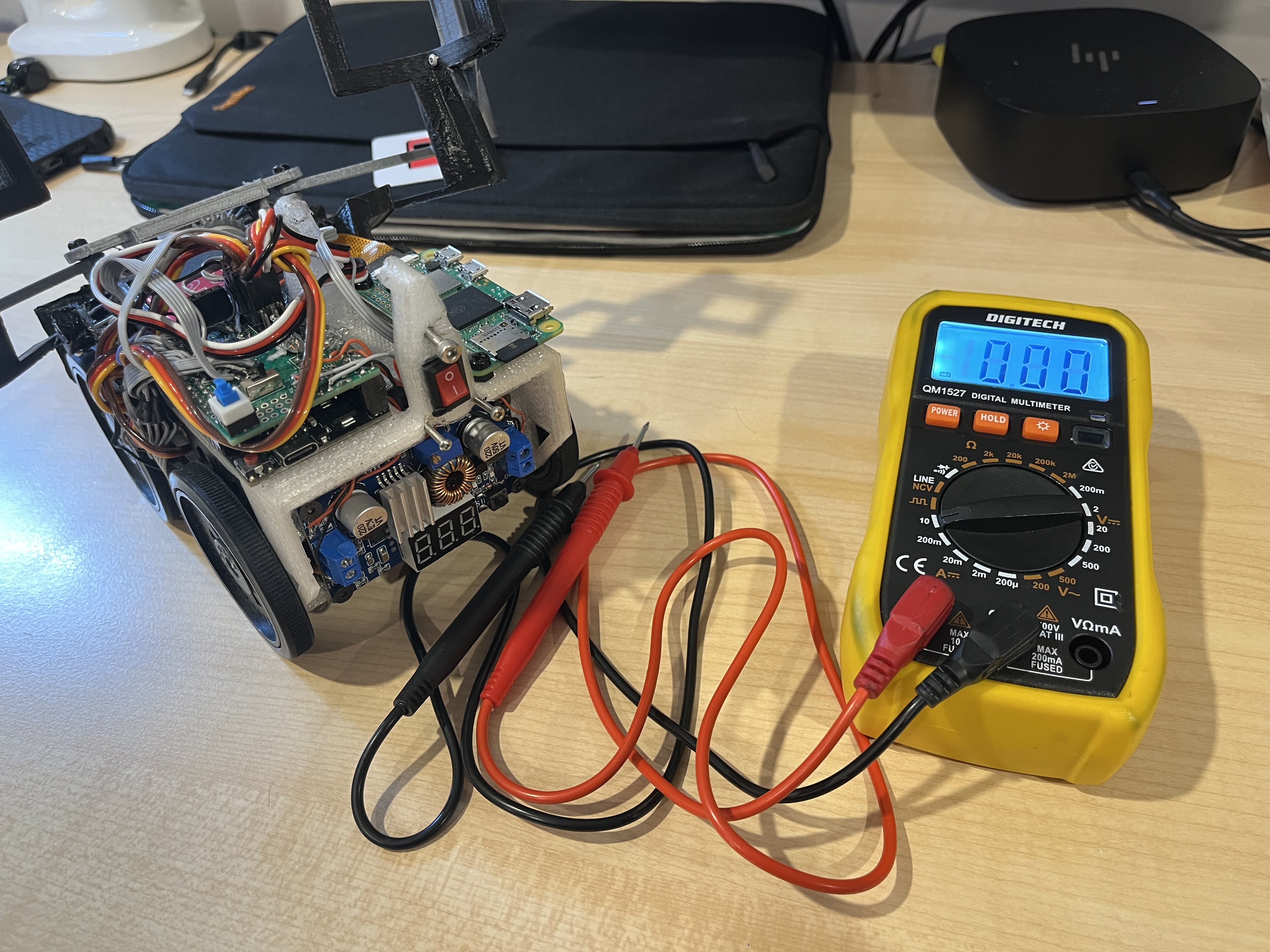
We will test these methods through constructing a test frame that holds the wheels in roughly the same positions and measure the velocity against percentage voltage. We should select the method that produces the widest range of velocities, consistently across different textures.

1. 4w servo
2. 4w dc
3. 2w servo + 2o servo
4. 2w dc + 2o dc
5. Tank treads with servo
6. Tank treads with dc

The wheel size and dimension will also play a large role in how our robot functions. They affect the grip strength, stability, and overall dimension of the robot.

We will investigate the ease of movement of different diameters and grip strengths, over a generic speed bump, toothpicks, balsa wood, pencils, and other objects, rated from 1-10. Calculating the mean score would suggest the best size to use.

Other considerations that may impact other elements of the robot design include current draw, and voltage requirements. For servos, the voltage requirements can be found through datasheets which provide a min and max value. The voltage will determine the battery we need to use. Thus, it could impact size, and weight distribution, which alters our car design. If stall current isn’t provided in the datasheet, it can be tested with a multimeter and power supply.

In this situation we used a multimeter that can handle up to 10A in the right mode, voltage regulator (regulating to the recommended voltage 5.8v), 2S 18650 batteries, a microcontroller (to turn the motors), 4 JX PDI-6221mg 360 servos, and a 3d printed car chassis to hold the components. High load for our use case would be the servos all changing from max speed in one direction to another, at the same time. This turning causes a sharp rise in current draw momentarily as all servo’s change direction.   
Fig 1.1 : Equipment ready for testing

We measured an average high of 2A which ensured that the 2.5mm pitched connectors would be enough with their 3A rating. To measure accurately we:

1. Interrupted the circuit where the current flows between the voltage regulator and the servo (an alternative is to probe between the battery and voltage regulator)
2. Connected the multimeter's positive side to voltage regulator following the current flow over to the negative side and then to the servo’s power input.

This method of measuring the increase of current draw works, and can be explained through physics, particularly Newton’s first law of motion, which states that an object at rest stays at rest and an object in motion stays in motion with the same speed and in the same direction unless acted upon by an external force. When the servos abruptly change direction, the inertia of the moving parts inside each servo resists this change. Overcoming this inertia requires additional force, which in electrical terms translates to an increased demand for current. This sudden spike in power demand can lead to a brief but significant increase in the current drawn from the batteries. Managing this transient current is crucial as it affects the electrical stability of the system and can potentially lead to voltage drops, affecting performance or even causing damage to the electronic components if not properly regulated. Therefore, designing a robust power management system that can handle these peaks is essential for ensuring the reliability and efficiency of our robot car.

## Line Tracking

The line width may vary – the difference between 10 and 20 is extremely large, especially given the uncertainty of ± 10%. There are several available methods that we can employ, each that we need to test. Results will be documented underneath each point.

Intersection must also be tested alongside line following as it is critical that we are not optimizing for one task but are able to complete every task.

We will measure the ease of use through attempting to program a reliable line follower program, then measure the time taken for a certain course that combines most variable elements.

1. Camera with filtering/tracking
2. Camera with machine learning
3. Variety of IR emitters + IR photodiodes
4. Bright LED to reduce noise + TCRT5000
5. Bright LED + Photodiodes

## Evacuation Zone

There are several strategies to moving through the evacuation zone, with the added complexity of obstacles, we must thoroughly test this as rescuing victims is crucial to being successful. Depending on which method, the sensors required differ.

Testing all these methods requires extensive time and research to get right, hence proper research done beforehand would greatly reduce the workload later.

### Preset Path – Snaking or Circular

The most basic task is highly adaptable as the starting point does not matter. This method would also ensure that every space is covered, no matter how obstacles are placed within the evacuation zone. A touch sensor at the front and the back is necessary, and any additional sensors.

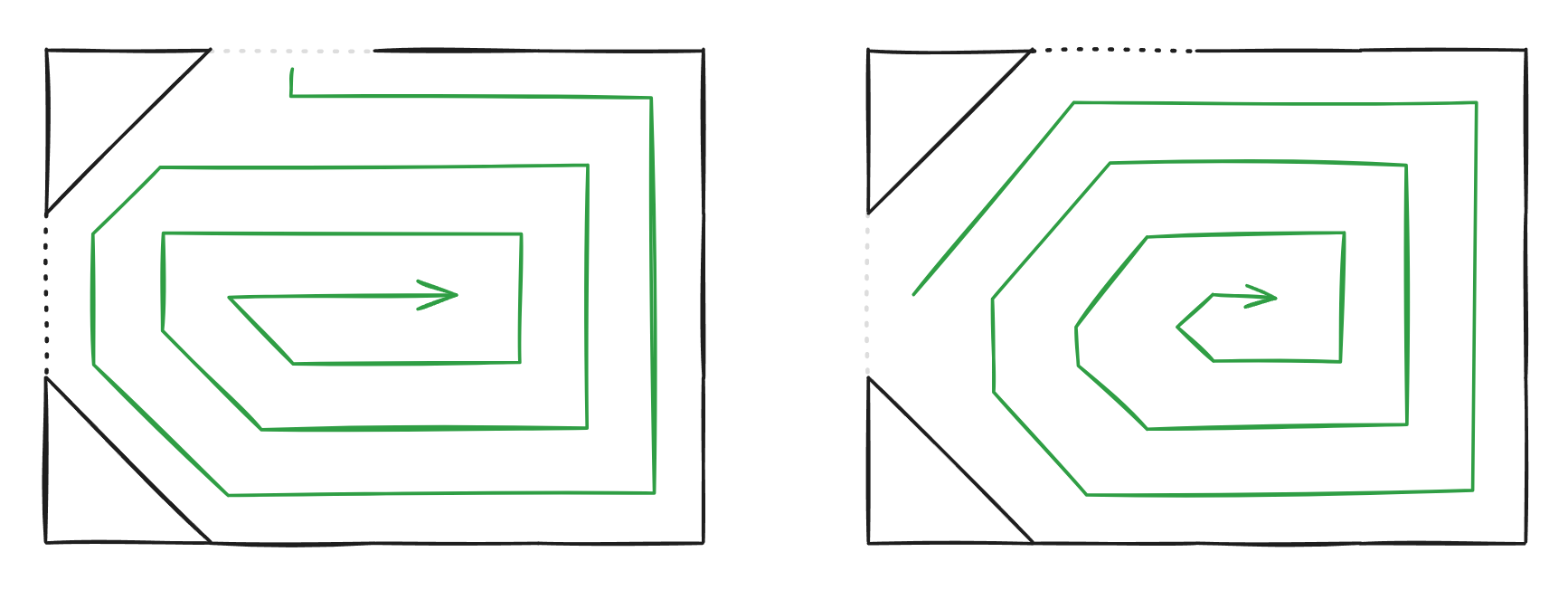


Fig 1.1: Possible paths that we could recognize and pre-program.

However, using this method requires extensive testing into every possibility in placement of the triangles and entrance, exit points.

### Wall following

Using a laser sensor placed on the side of the robot would allow us to measure the distance from walls. This is beneficial now we align ourselves parallel to the wall, ensuring that we cover all the ground possible, without the need to hard code every possibility. Furthermore, all the possible areas are covered, and navigating around obstacles are possible.

### Pathfinding

If we can gain a full map / understanding of the boundaries of the evacuation zone, we would be able to identify which areas we have not checked, and where they likely are. Having an obstacle located at the centre would suggest we have to check all the way around.

## Microcontrollers / Controllers

*Arduino Portenta H7*  
Checking YouTube videos for this, it seems overpriced for its capabilities, and we’d be better of using a single board computer, as it has much greater performance for its size.

*Google Coral Dev Mini*  
Having an integrated TPU, and being specifically designed to run TensorFlow Lite models, this seems like a great option, especially seeing as its dimensions are smaller than a RPI 5 model.

*Google Coral Dev Micro*  
This board is both a microcontroller and a single board computer. This allows us to control all the sensors and servos that are required to complete this challenge. Its size is even smaller compared to the google coral dev micro, which has no effect as this single board would be able to control everything. We would still have extra space for other components; hence efficiency is reduced. We would rather have a larger board with more computational power.

*Raspberry Pi 5*  
This was the most obvious choice for us. A step-up from the RPI Zero 2W model that we were running in Singapore. Bringing more computational power, and extra camera slots, would greatly benefit our robot.

*Raspberry Pi 5 compute module*This board, while being smaller and better overall compared to the RPI 5, is not yet released, and hence we cannot build and hope that it releases this year before the competition.

*ESP 32*Untested

*ESP32 CAM*Untested

*OpenMV Cam*This is a specifically designed Arduino board that carries a camera and enough processing power to run machine learning models on it. However, looking at the demonstration videos, it seems as though it is only compatible with boards within its own ecosystem, and functions with its own IDE, which greatly limits what we can achieve whilst increasing cost.

*Conclusion*We will be using the Google Coral Dev Mini, as it maximises computational power, but has a smaller volume, hence we can better balance the robot, fit more components on the robot.

# 2 | Design

# 3 | Coding

## Coding Conventions

A problem that we struggled with during Robocup Singapore was different coding styles, naming conventions, and debugging methods. As a team, we should have the same coding conventions so that code is uniform throughout all code documents.

**Indentation**: Use 4 spaces per indentation level. Avoid using tabs, and if using a mix of tabs and spaces, Python 3 disallows this.

**Line Length**: Limit all lines to a maximum of 79 characters. For flowing long blocks of text with fewer structural restrictions (docstrings or comments), the line length should be limited to 72 characters.

**Blank Lines**: Separate top-level function and class definitions with two blank lines. Method definitions inside a class are separated by a single blank line.

**Imports**: Imports should usually be on separate lines and are placed at the top of the file just after any module comments and docstrings and before module globals and constants.

**Whitespace in Expressions and Statements**: Avoid extraneous whitespace in the following situations:

* Immediately inside parentheses, brackets, or braces.
* Between a trailing comma and a following close parenthesis.
* Immediately before a comma, semicolon, or colon.
* However, add whitespace around operators and after commas to increase readability.

**Naming Conventions**:

* Function names should be lowercase, with words separated by underscores as necessary to improve readability.
* Variable names follow the same convention as function names.
* Class names should normally use the CapWords convention.
* Constants are usually defined on a module level and written in all capital letters with underscores separating words.

**Comments**: Comments should be complete sentences. If a comment is a phrase or sentence, its first word should be capitalized, unless it is an identifier that begins with a lower-case letter (never alter the case of identifiers!).

**Docstrings**: Pythonic code often includes docstrings that follow the conventions outlined in PEP 257. These provide a clear explanation of the function or module purpose and how to use it.

  
Fig 1.1: Code example using all coding conventions implemented.

# 4 | Testing

# 5 | Final Thoughts

We have released all our code, designs, and documentation within our GitHub Repository.

<https://github.com/D-uality/BitFusion-Robocup-Asia-Pacific>