

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

Robocup Asia Pacific

Bit Fusion

# 0 | Abstract

Robocup AsiaPacific brings forth the international rules – of which is our second time competing with. BitFusion is a team that combines innovative strategies in both software and hardware approaches. Maintaining a central focus on completion of the event, our robot combines camera vision with omnidirectional mirrors, custom sensor boards, and controllers with custom PCBs.

# 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. This time, we ensure that we carefully read the rules and highlight the key areas that we missed.

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

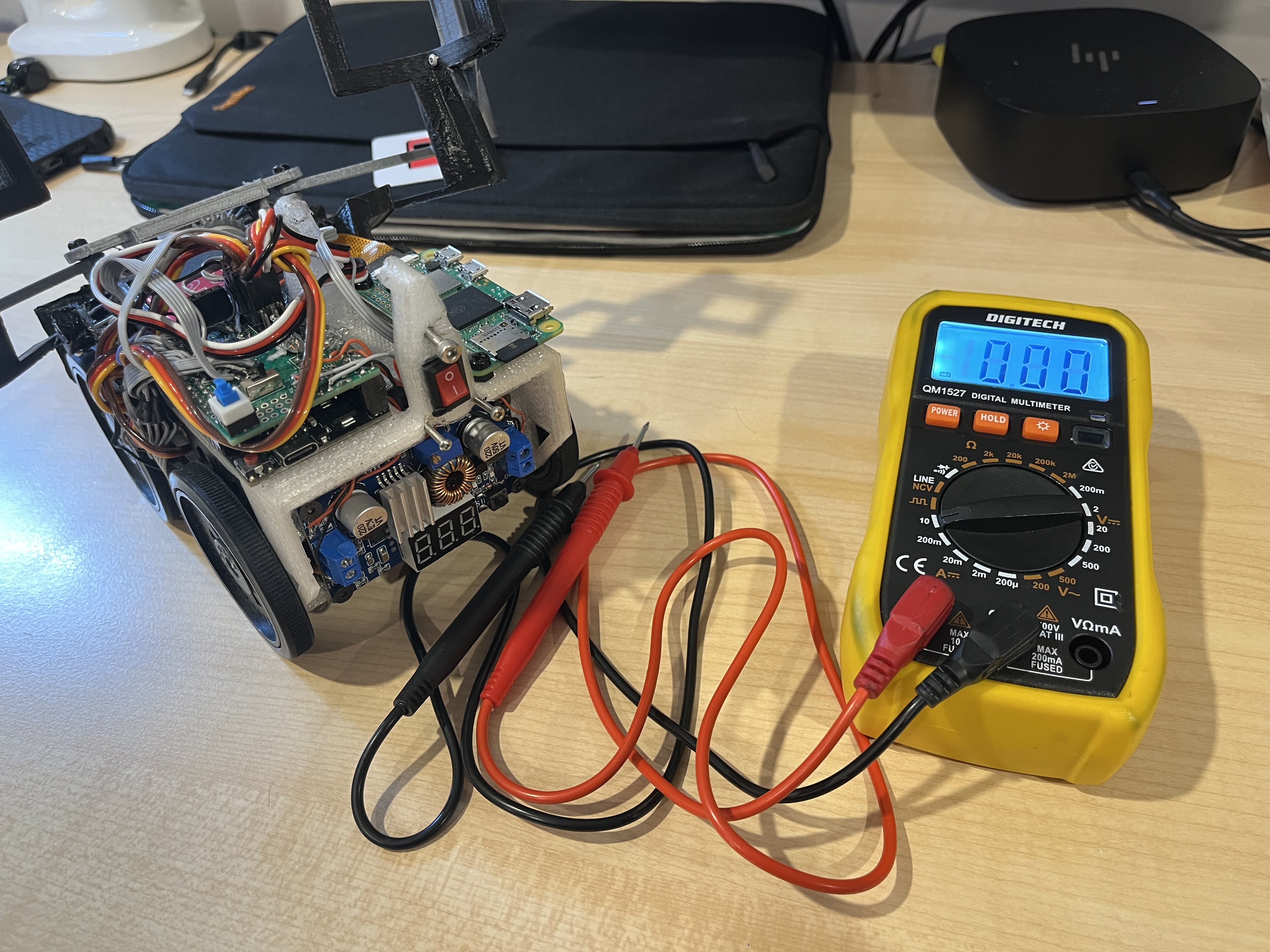
Navigating the course requires some sort of movement. Given that the boards can be different textures, (linoleum or carpet), or system must be robust to handle these different friction levels.

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.

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.

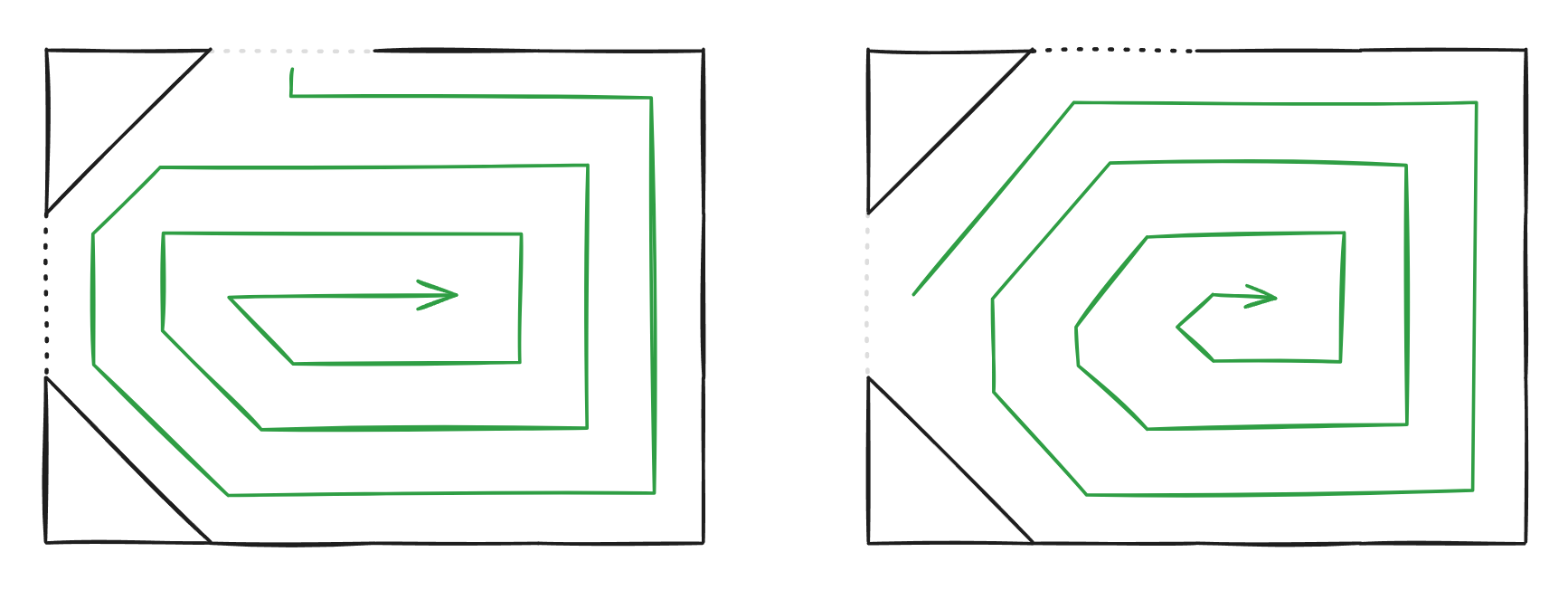


Fig 1.1: Possible paths that we could recognize and pre-program.

However, using this method requires extensive testing into every possibility in placement of the triangles and entrance, exit points. Furthermore, the code structure would be extremely nested if we want to determine the exact scenario we in in given limited information.

### 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. The only potential reason we’d use this is for it’s integration with a microcontroller, which means we do not need to manually create one ourselves to carry the ADC (Analogue to Digital converter), PWM signal generators, and other functionality that a SBC wouldn’t have by itself.

*Google Coral Dev*  
Following Robocup Singapore, seeing Catbot-Jet utilise a Nvidia Jetson Orion, giving it processing

*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. Furthermore, there is still a TPU processor on board so that we can run TensorFlow Lite models reasonably quickly. 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. This board has the most community support, tutorials, and fully fledged guides, in case we get stuck (especially seeing as it is our first time using Linux).

*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. Creating a shield that contains a microcontroller, will handle the analogue input and output, as well as generating PWM signals for the variety of servos.

In this case, communication between the SBC and the MC must be established. Several options are available: USB (UART), I2C, and SPI. For short distance communication, SPI and I2C are much faster compared to USB. However, during development of the Robot for Robocup Singapore, I2C proved to be unreliable, and we had to switch to USB.

This time, we will implement SPI, which has many variants for synchronous communication. This allows us to reliably send data between the different boards. USB has been implemented as well, just as a backup in case SPI does not behave as expected.

# 2 | Design

Design is extremely important as solving a problem with hardware is infinitely easier compared to solving it with software – which is sometimes impossible. Having a clear, clean design is critical to success.

## Custom PCB

For our SBC, handling all the processing tasks is easy and doable. However, to control sensors, generate PWM signals, and controlling our robot is not. Hence there is a requirement of a microcontroller. Possible microcontrollers include:

**Arduino Uno**The Arduino Uno is a classic choice. However, it is easily out shadowed by other alternatives.

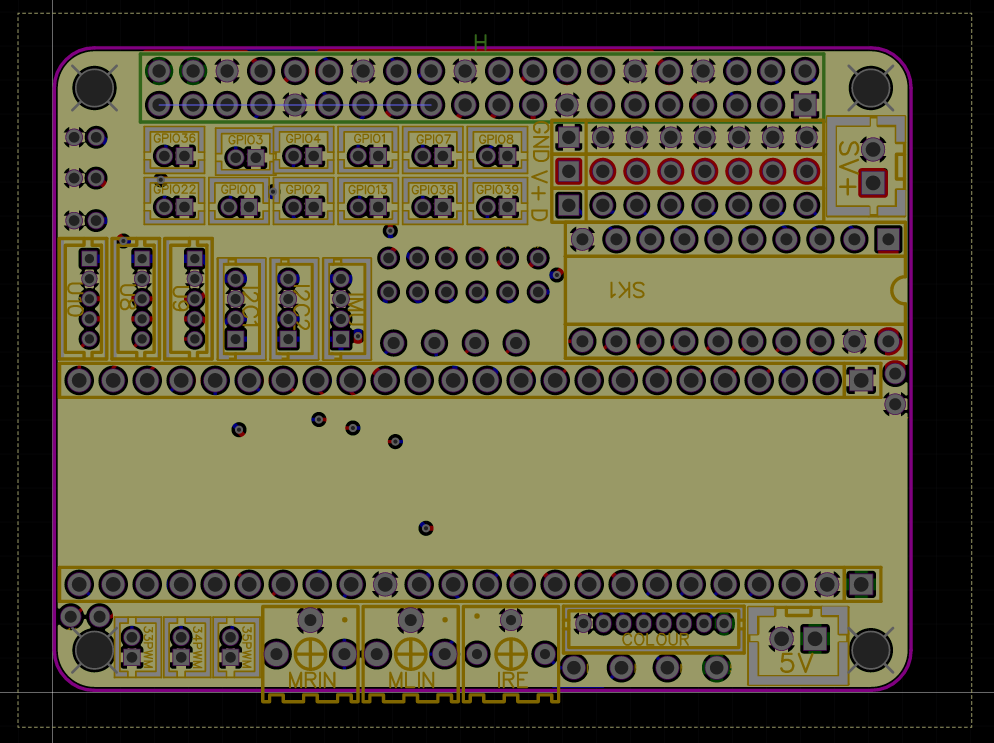
**Arduino Mega Mini**A straight upgrade from the Arduino Uno – smaller volume, more pins, faster processor – this is what we used during Robocup Singapore.

**Teensy 4.1**This is what many other teams use in conjunction with their own SBC. This board has an even smaller volume, whilst maintaining enough digital and analogue pins for various functions.

**Arduino Nano**

|  |  |  |  |
| --- | --- | --- | --- |
| Microcontroller | Digital Pins | Analogue Pins | Dimensions |
| Arduino Uno | **14** | **6** | **101.6 x 53.3** |
| Arduino Mega | **54** | **16** |  |
| Arduino Mega Pro Mini | **54** | **16** |  |
| Arduino Mini |  |  |  |
| Arduino Nano |  |  |  |
| Teensy 4.0 |  |  |  |
| Teensy 4.1 |  |  |  |
| ESP32 Dev Board |  |  |  |

To have everything fit on the Google Coral, however, we must implement the smallest microcontroller possible, as we still have other components to place. Hence, we decided to go with the Teensy 4.1, which has ample digital and analogue pins to support all our servos, sensors, and having backup for testing.

Our PCB contains a header that connects directly to the 40 Pin GPIO input/output. This allows us to directly place the shield over the computer, which minimizes lost space, and provides a more robust system.

This board has standoff points that link up with the dimensions of the Coral, which allows for secure attachment.

We have chosen to use a Teensy 4.1 as our microcontroller – linked with SPI or UART communications. This further allows for analogue inputs, as well as generate PWM signals.  
Fig 2.1: PCB layout

A computer screen shot of a computer

Description automatically generatedOn the shield, we have JST connectors, which is a great improvement compared to using jumper wires, as it provides stronger connections, smaller connection points, and easy removability.

On board, we have I2C pinouts, servo pinouts, a teensy, colour sensor boards, Teensy 4.1 boards, VIN, VCC, Ground, spare digital pins, MPU6050 pinout. Many connections will be done through the JST connectors.

Fig 2:1: PCB Components

## Coral Mini Shortage

However, our chosen SBC (Google Coral Development Board Mini) is a hard board to find, especially in New Zealand. Lack of stock, and lack of official retailers makes the closest – and only reliable one – a company from Germany. However, shipping costs as well as shipping time have to be compensated for. We do not have an extra month just to wait for our computer to arrive, hence we quickly made the decision to switch from the Coral to Raspberry Pi 5 instead.

A computer screen shot of a computer

Description automatically generatedA computer screen shot of a computer

Description automatically generatedThis decision was not made easily – there was still the Google Coral Development Board. At the same dimensions, it carries more processing power compared to the RPi 5, with a TPU capable of carrying 4 trillion operations per second. However, there are two downsides: the first being that   
the shield would be incompatible as the cooling system would be too high to attach anything to the exposed GPIO pins. Furthermore, reviews suggested that there were problems with the software, with Authentication Key locks present, requiring extra work to setup.

With the number of online tutorials, forums, and videos for the RPi – much more support is offered, which helps with our ability to debug any problems that we find.   
Fig 2. : Image of the final SBC/Microcontroller together with the shield that links them.

## 1st Adaptation – Raspberry Pi 5 Power failures

We had designed a basic prototype for the RPi 5 within a week of receiving the item.

## Design Focuses

During Robocup Singapore:

1. Our robot was too small – everything was too compact which led to inefficiencies in building and designing. Furthermore, this exacerbated the problem with tolerance in multiple parts of the model.
2. The centre of gravity, while okay to run on normal tiles, were not optimised. Going uphill and downhill required more control. In addition, going over speedbumps on ramps were challenging, and caused or robot to tip.
3. Our wheels are too small. Traversing over oddly placed speed bumps resulted in struggles, especially when placed around turning points.
4. Battery removal was difficult, and required taking off a wheel, some screws, and moving the battery pack.
5. No consideration for tolerance lead to problems with 3d printing, gear systems working reliability, and overall construction and ease of assembly.­­­­­­­ This led to having dysfunctional systems when gears and other linkages broke.
6. We had no spare pieces, save for a few for the important pieces such as the claw, and a single base board. Spare pieces and components were disassembled.
7. Everything was, in the end, a single piece. Whilst this saves weight, it also reduces are design mobility, as if something needs to change, it will affect everything, instead of just a single, related, part

This time, we will:

1. Increase the robot size by a bit, to increase the space we have to work with. This allows us to accommodate the larger SBC, as well as extra wriggle room to build and connect components with ease.
2. Lower and centre the centre of gravity, to avoid tipping, especially whilst on ramps and seesaws.

# 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

During Robocup Singapore, our team was split into two different groups:

* Aidan
  + Hardware management
  + 3D printing
  + Line following + Obstacle + Evacuation Zone pathing
  + Overall coding
* Frederick
  + Lead CAD designer
  + Claw design
  + Evacuation Zone identification + TensorFlow Lite model + Image processing
  + Raspberry Pi setup

While this played to the strengths of our team, it lead to a situation where Frederick didn’t have a completed robot by the time we flew to Singapore, hence we had no replacement parts, nor were we able to provide assistance with the other during stressful situations, due to our limited knowledge said field. This lead to many issues with code and design, as we were working with one approach, and not two. Hence we missed out on many possible solutions until it was too late.

These few months, we will strategically plan our timeline and track all our progress, so we are not overwhelmed and left scrambling.

## Timeline

We have from 8th of April till — to fully develop and test a robot with all features installed and working.

During our preparation to Robocup Singapore, our team relied on our teachers PVC Foam boards. This severely limited us in what combinations and different scenario we could practice. For example, the exit was placed in such a way that following the wall clockwise would always let the robot leave the evacuation zone, hence we never prepared for the situation in which the exit did not allow for that.

As a solution this time we will be creating a more, robust system that tests both the normal cases and the extremely hard ones. Our evacuation zone should now support modular entrance and exit locations, as well as our evacuation points.

### Boards

# 5 | Conclusion

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

Visual Studio Code, OnShape, Fusion 360, Github, Arduino IDE, , TinkerCAD, EasyEDA, Linkage, Fritzing, ChatGPT, Perplexity AI, has been used during the process of creating and deploying our robot.

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

<https://cad.onshape.com/documents/t/66503fd4285ce43accbee3c1>