Engineering Journal

Team ID: FE24_309_01

Team Name: SSTalwarts

Team Members:
Kevin Anthony Velasco Orjalo
Tan Siew Ling
Sim Jie Yi Kieren

Formal Photo:

Informal Photo:

Contents

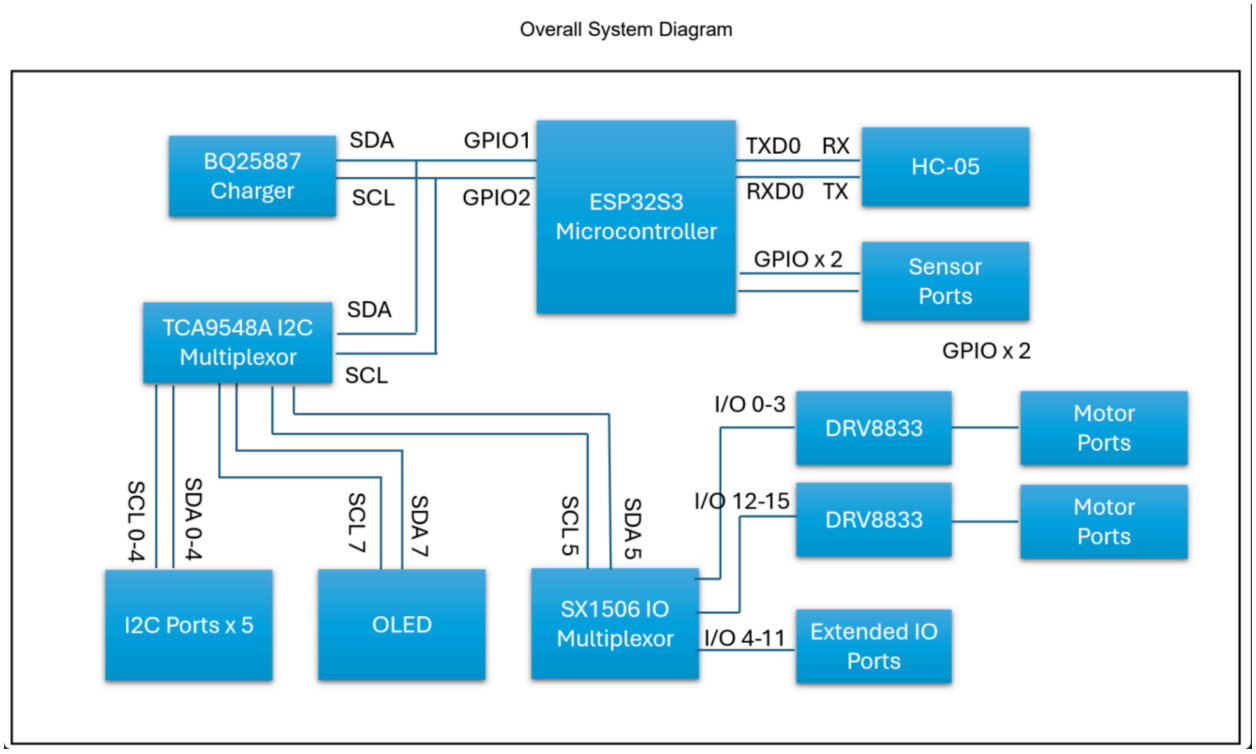
1. Hardware Overview
2. Drivetrain
3. Components
4. Strategy
5. Code

Hardware Overview

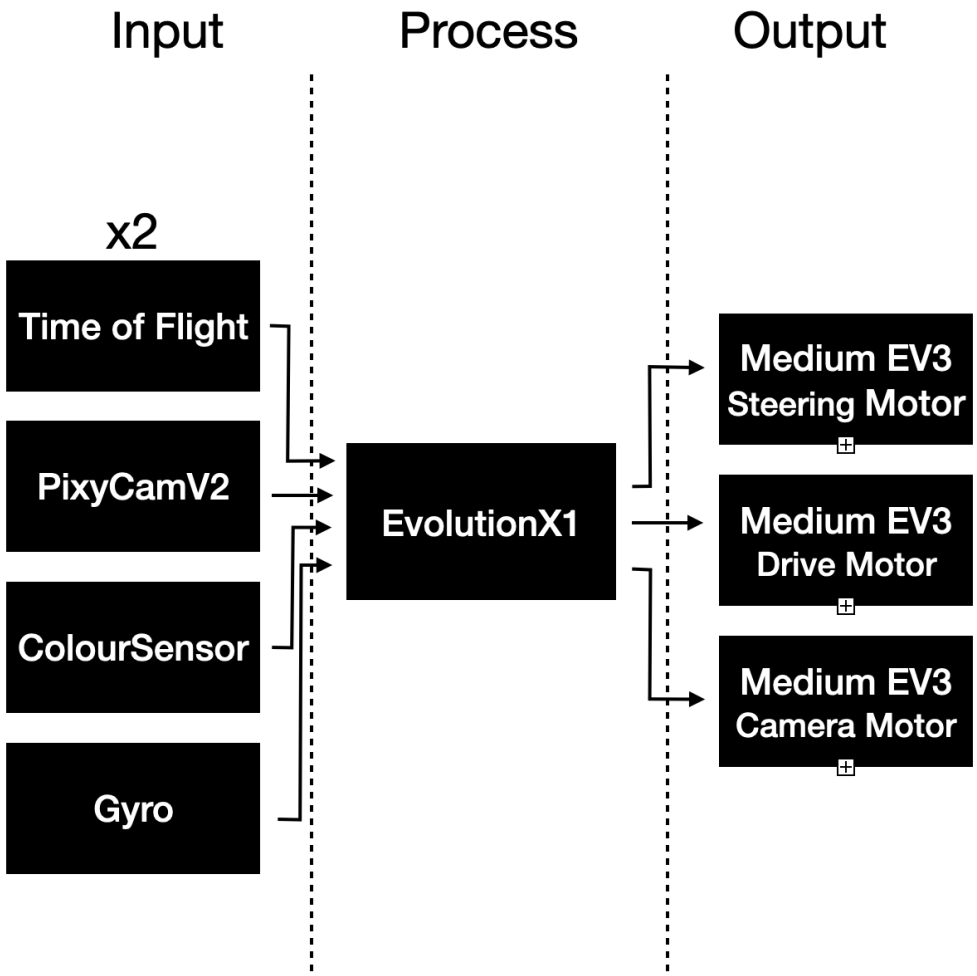
Table of Components used

Description	Model	Quantity	Others
EV3 Motor	Medium Motor	3	
Time of Flight	VL53L0X	2	
IMU	MPU9250	1	
Colour Sensor	TCS3472	1	
Camera	HuskeyLens	1	
Microcontroller	ESP32S3	1	
Charger	BQ25887	1	
I2C Multiplexor	TCA9548A	1	
IO Multiplexer	SX1506	1	
Battery	NCR18650B	2	

Microcontroller System Diagram



Block Diagram

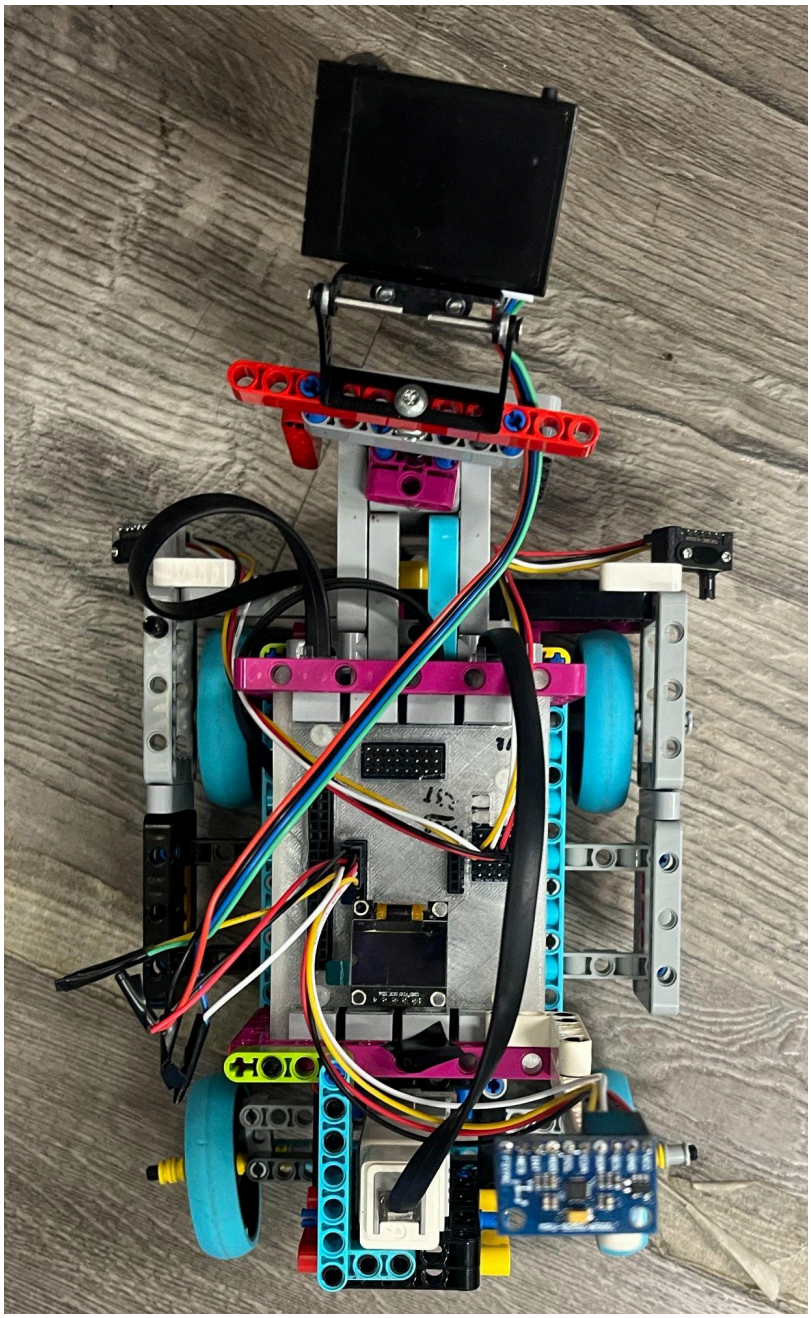


Drivetrain

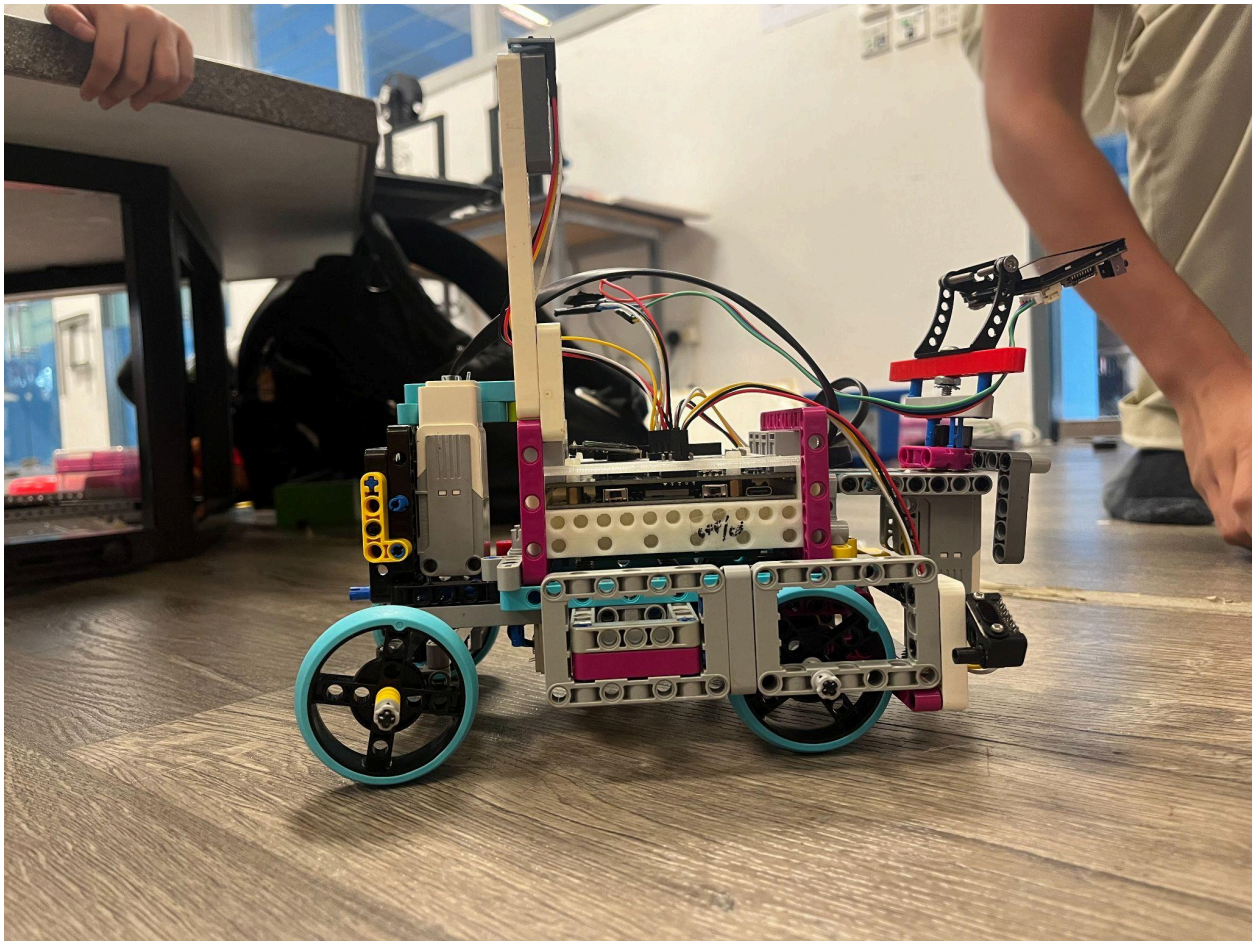
Chassis

Our chassis was built from Lego parts. We chose this material because we found it light and versatile to use, and we had abundant lego parts. Furthermore, it has a simple design that allows us to modify it and prototype it easily. We also 3D printed holders for our sensors, which have lego compatibility, so it allows us to install on our bot easily.

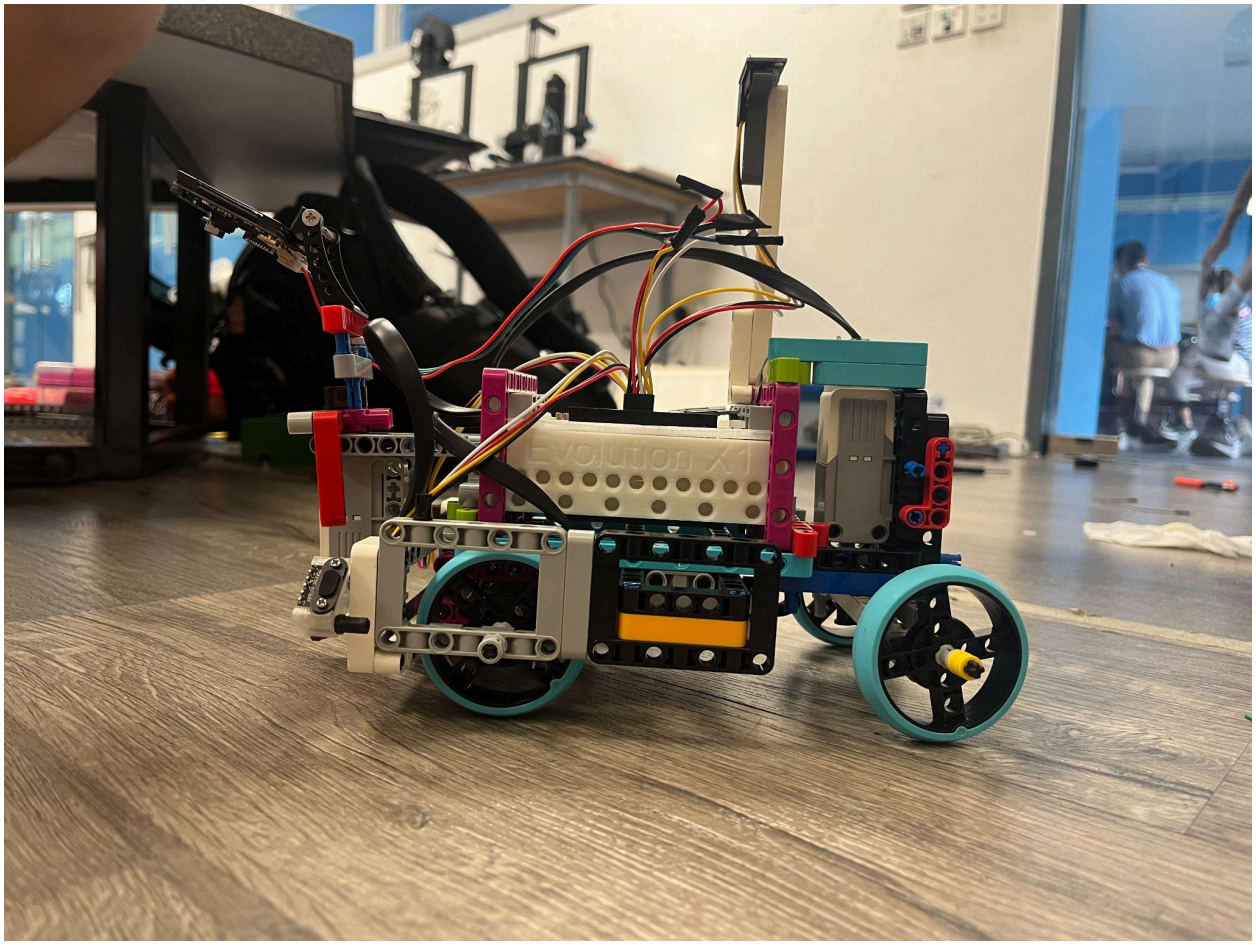
Top View:



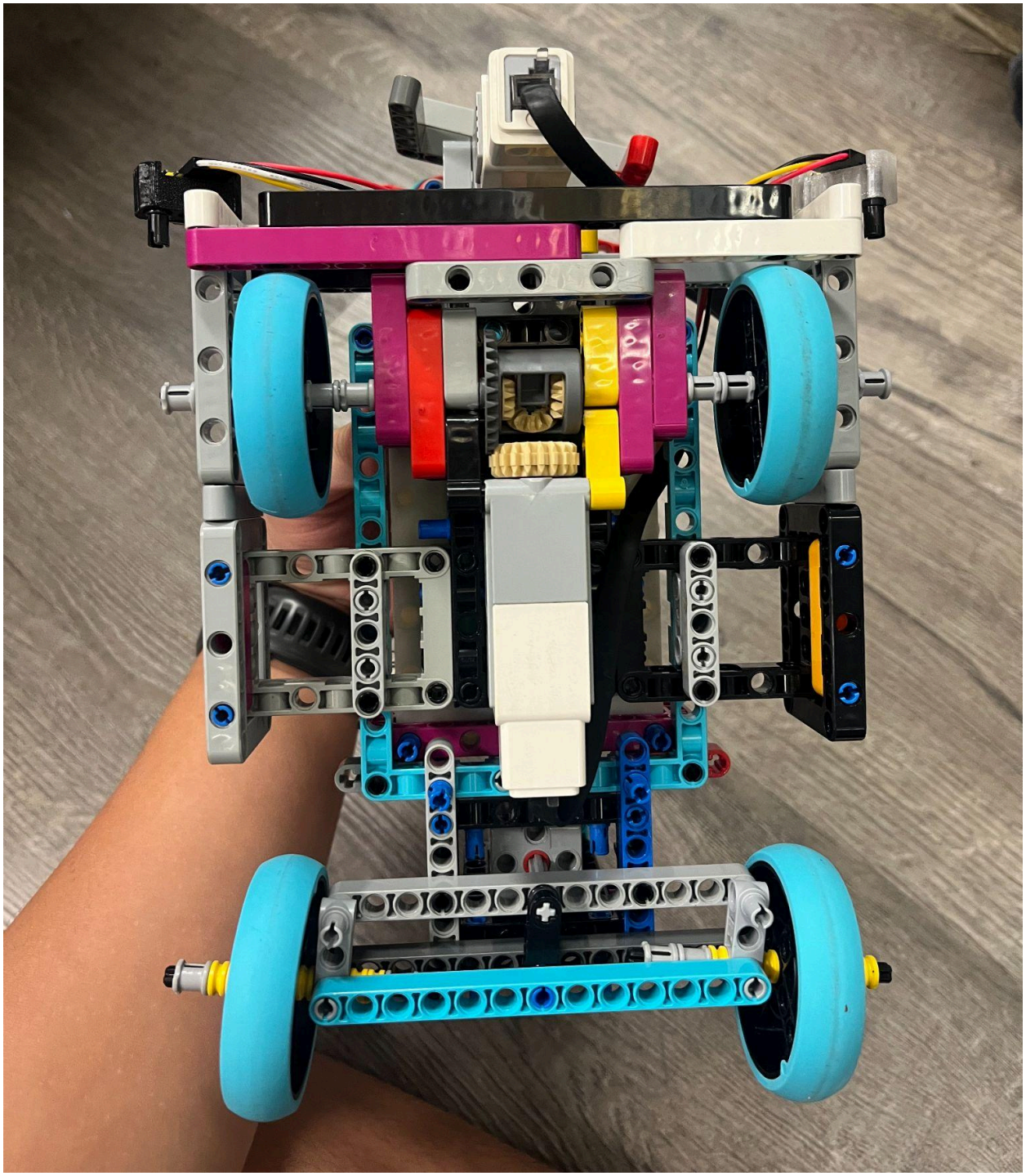
Right Side View:



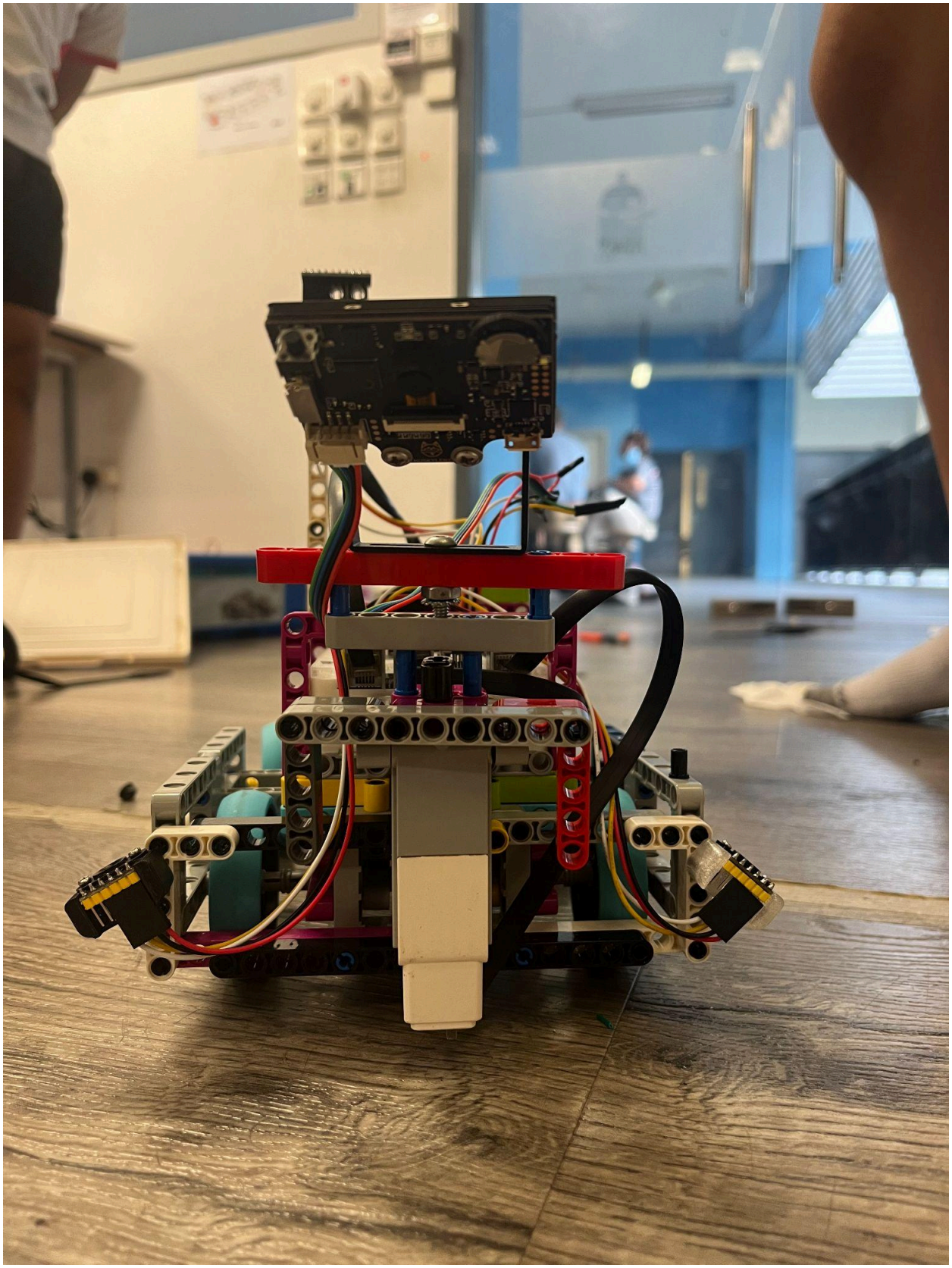
Left Side View:



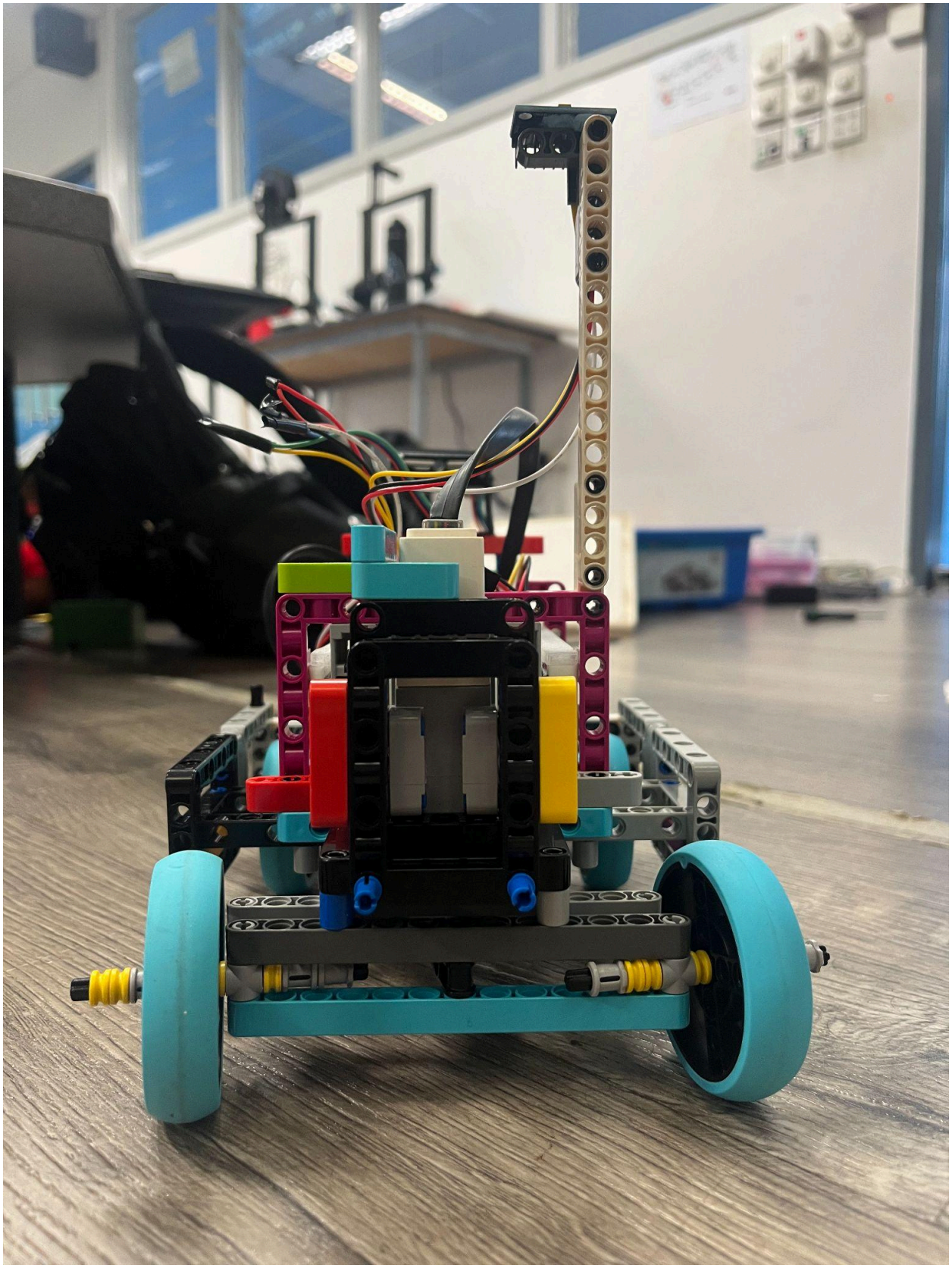
Bottom View:



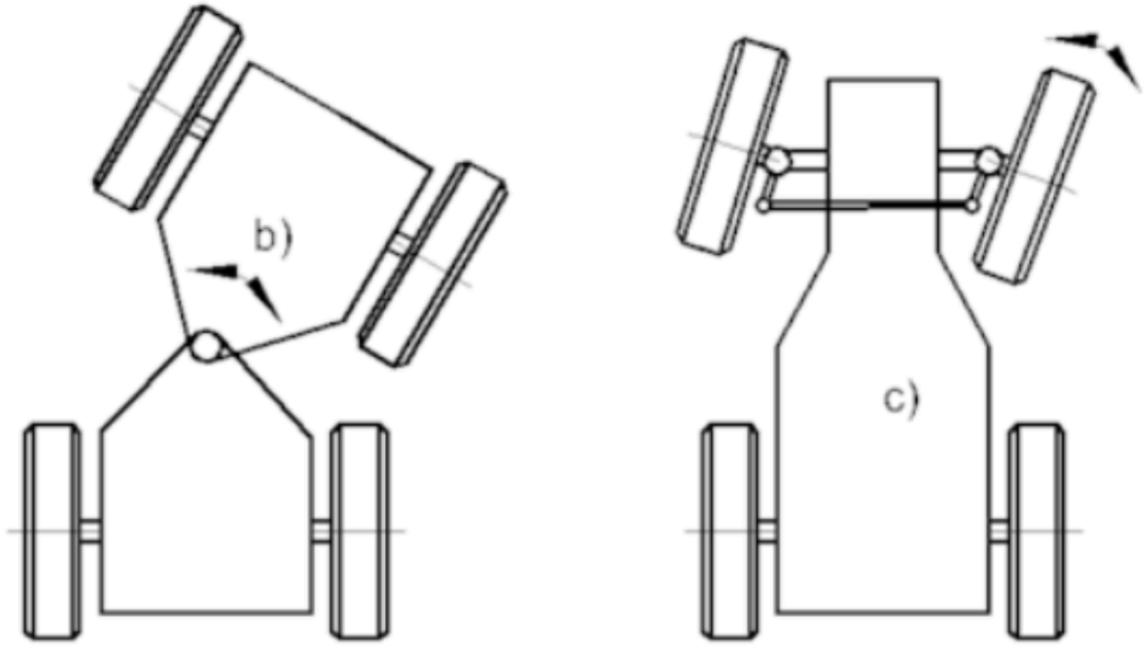
Front View:



Back View:



Steering



Our team has chosen the steering on the left, c) which is the ackermann steering. We chose it as it was very versatile and easy to build. We were also more familiar with it, so it was very easy for us to build it. However, lego had limits to building techniques, which caused us to be unable to go for other types of more advanced steering, like the one on the right, b) articulated steering.

Motors

The motor we used is the EV3 medium motor. We chose it as it had a balance of speed and torque. We initially considered the EV3 big motor, but however, we found out that the lot bigger size made it difficult to install it on the bot. Furthermore, it also focuses on stronger torque and slower.

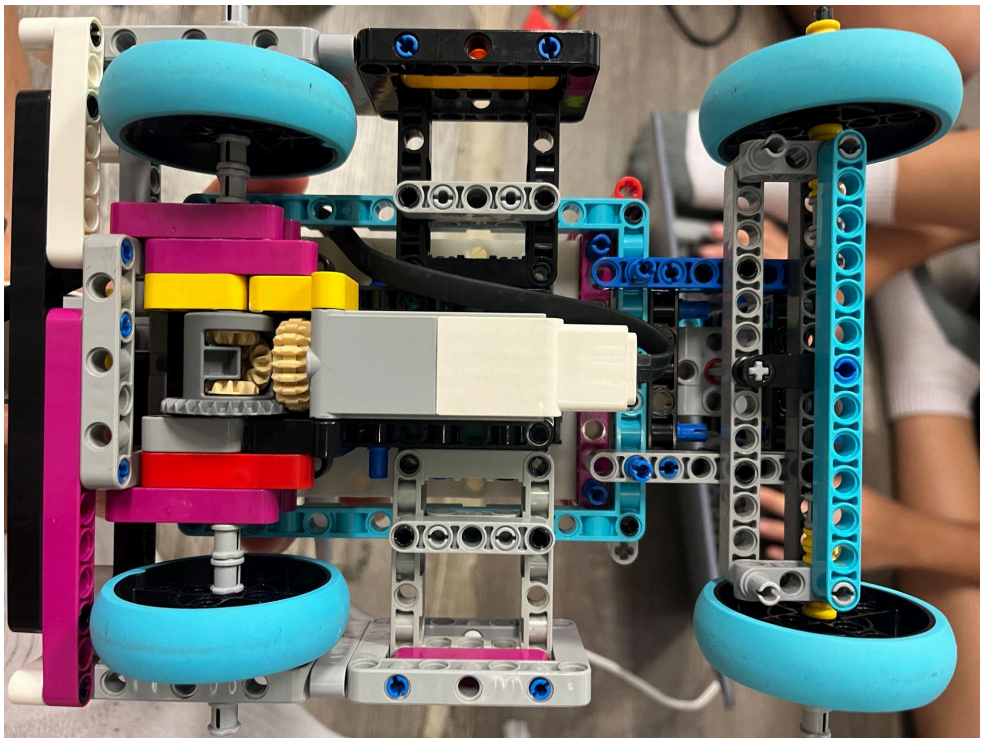
Medium Motor

Big Motor

Our motors are used for the drive, steering and camera.

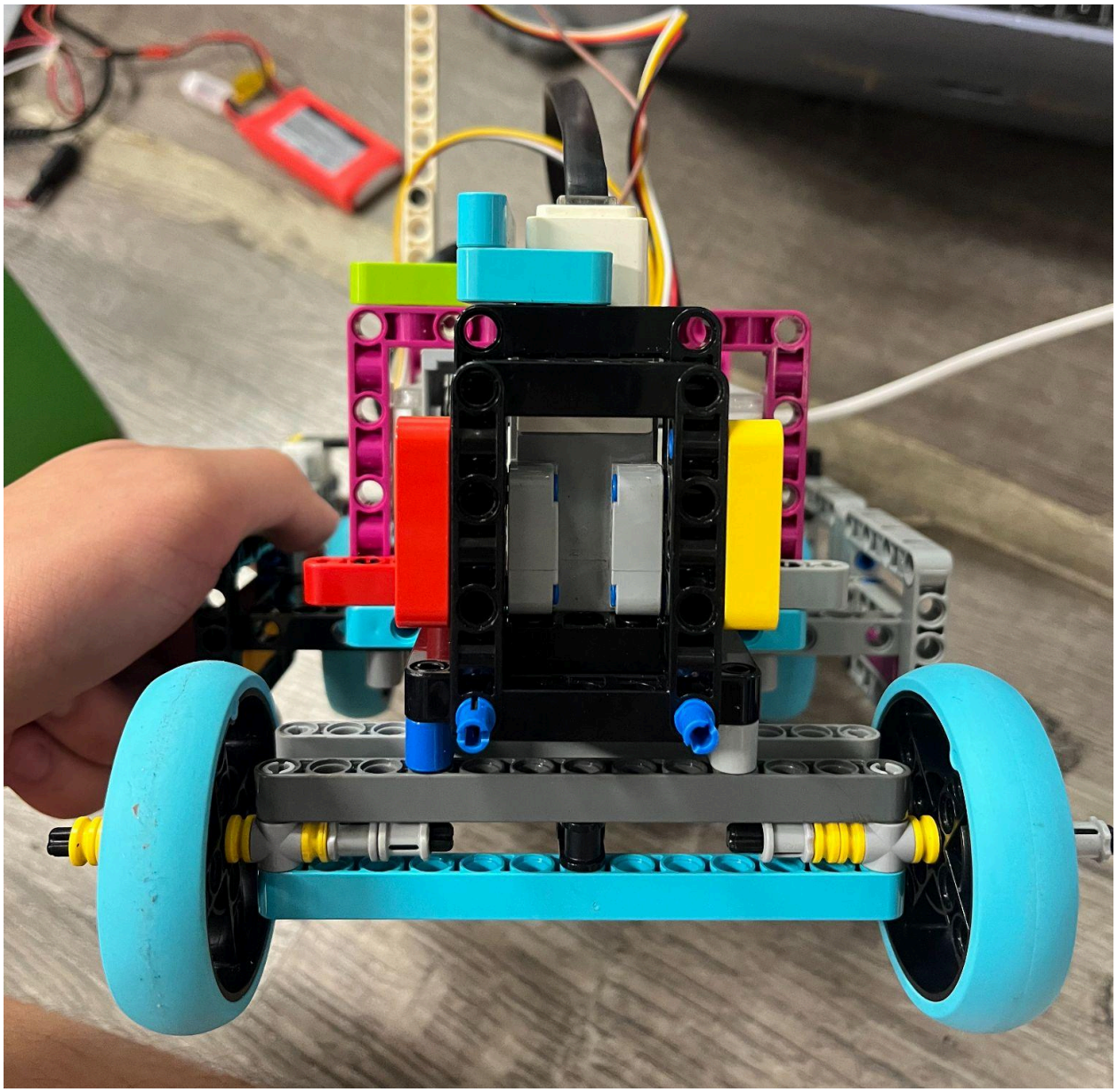
Differential Gearing

The drive motor is connected to the differential gearing at the bottom of the bot. The differential gearing allows the wheels to spin at different speeds, which is very important when we are turning.



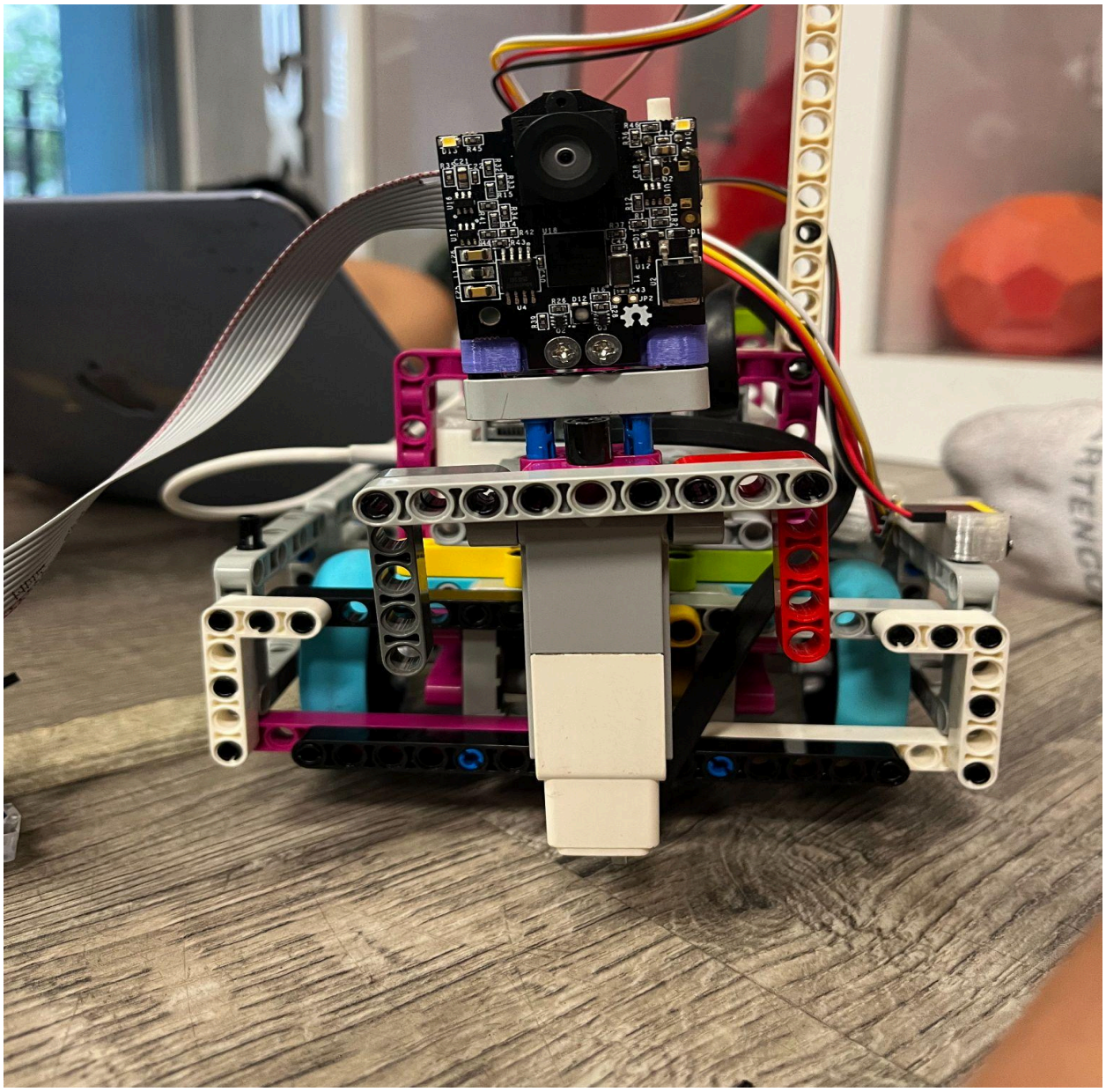
Steering

The steering motor is attached at the back of the bot. During testings, we found out that when the steering motor is attached at the front, the movement is very bad. However, when it is at the back, it is more consistent



Camera

The camera motor is attached to the front to control the camera position as this is to ensure that the camera is always facing the traffic blocks even if the bot is turning.



Components

EvolutionX1
The Evolution X1 Controller Platform is a versatile and powerful development board designed to seamlessly integrate with the LEGO EV3 Mindstorms ecosystem. Built around the robust ESP32-S3 microcontroller, this platform offers a unique combination of connectivity, performance, and flexibility. Below are the key features of evolutionX1 and some reasons why we chose it.

Key Features:
1. EV3 Mindstorms Compatibility:
The controller is equipped with specialized connectors that enable seamless connection and control of EV3 Mindstorms motors and sensors. This compatibility allows us to utilize the existing EV3 components while taking advantage of the advanced features of the ESP32-S3.

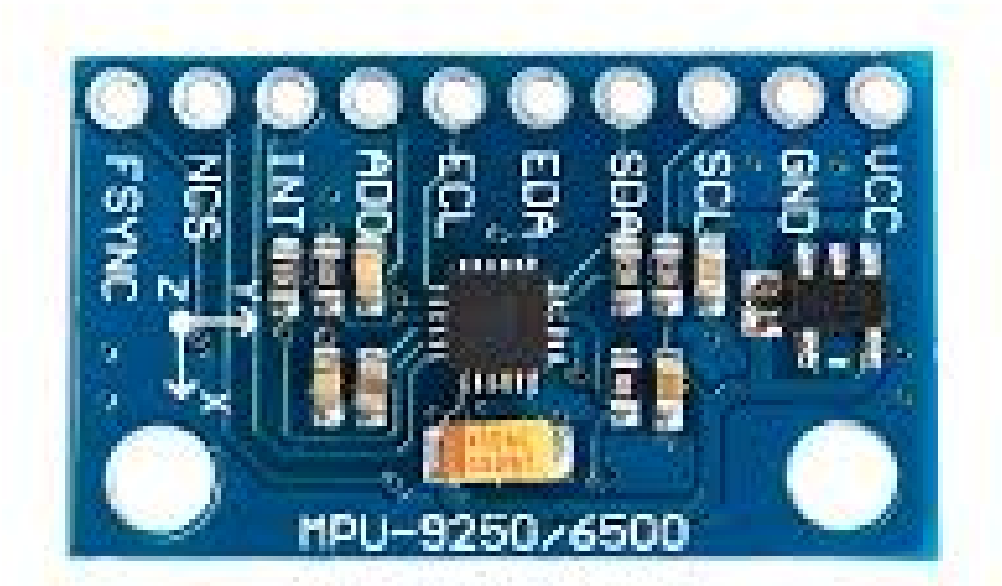
2. Extended GPIO Pinouts:
In addition to EV3 connectors, the platform offers an extensive set of GPIO pinouts. These pinouts allow us to connect various third-party sensors and actuators, broadening the scope for customization and experimentation.

3. Powerful ESP32-S3 Microcontroller:
At the core of the platform is the ESP32-S3 microcontroller, recognized for its high performance, low power usage, and extensive feature set. The ESP32-S3 provides dual-core processing, built-in Wi-Fi and Bluetooth, and a range of peripheral interfaces, making it an adaptable option for various applications.

4. User-Friendly Development Environment:
The platform is compatible with well-known development environments like Arduino, MicroPython, and Espressif's IDF, offering a familiar and user-friendly programming experience. Additionally, extensive documentation and strong community support further improve the development process.

Gyroscope

For our gyroscope, we chose the MPU-9250. It is a 9-axis motion tracking device that integrates a 3-axis gyroscope, a 3-axis accelerometer, and a 3-axis magnetometer. We chose it because, firstly, it combines 3 sensors into 1 chip, so it saves space and reducing the need for multiple components. Secondly, The gyroscope and accelerometer provide accurate measurements of angular velocity and linear acceleration, while the magnetometer adds orientation data, making it ideal for our bot motion tracking. Lastly, it supports standard communication protocols like I2C and SPI, which is what we are using.

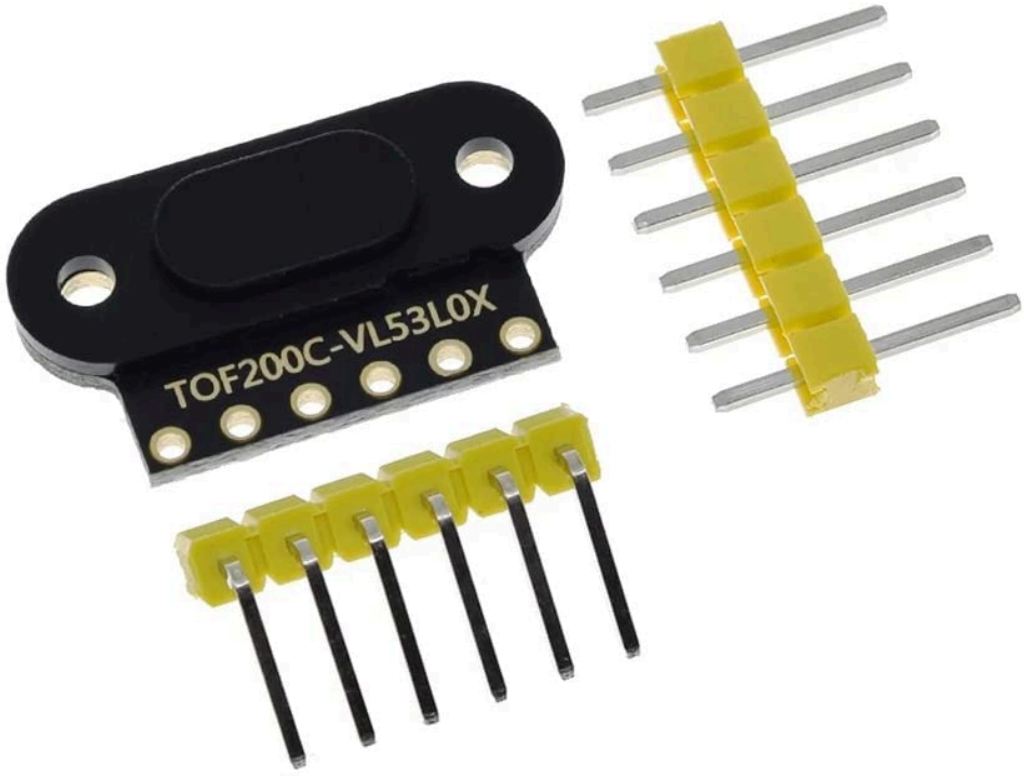


Time of Flight
The time of flight sensor is a distance sensor that measure the time it takes for a signal (typically infrared light) to travel to an object and reflect back to the sensor. This time is then used to calculate the distance between the sensor and the object.

We chose it as firstly, it had accurate distance measurements. ToF sensors can measure distances with high accuracy, often down to millimeters, making them ideal for applications requiring precise distance measurement. They can also accurately measure distances over a broad range, from a few centimeters to several meters, depending on the sensor.

Secondly, it has fast response time. ToF sensors are able to provide rapid measurements, enabling real-time applications like gesture recognition, object tracking, and obstacle detection. Furthermore, the speed of light allows for very quick distance calculations, resulting in low-latency data that's useful in dynamic environments.

Lastly, it has easy integration. ToF sensors often use standard communication protocols like I2C or SPI, making them easy to interface with microcontrollers and other digital systems. Some ToF sensors come with built-in processing, simplifying the implementation in complex systems.



Colour sensor

The colour sensor we chose is the TCS3472. It is a colour sensor is able to provide RGB (Red, Green, Blue) and clear light sensing capabilities. We chose it as firstly, it has accurate color sensing. The TCS3472 can accurately detect and measure the intensity of red, green, and blue light, along with clear (unfiltered) light, allowing for precise color recognition. It features a 16-bit analog-to-digital converter (ADC), providing high-resolution color data that enhances the accuracy of color measurements.

Secondly, it has a wide dynamic range. Iit is highly adaptable to various lighting conditions where it is capable of operating in a wide range of lighting conditions, from low-light to bright sunlight, due to its wide dynamic range. This ensures consistent performance regardless of the environment.

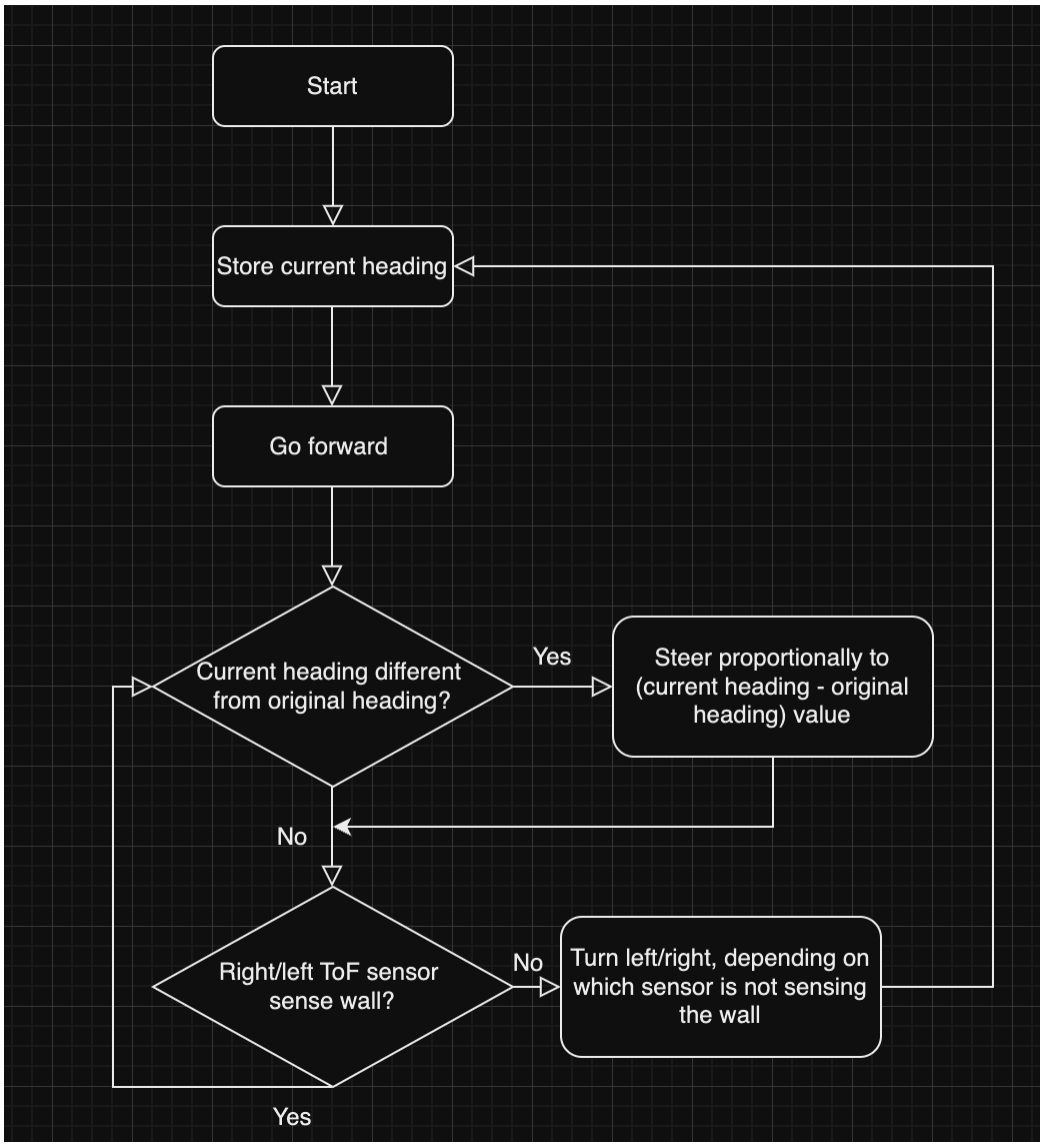
Lastly, it has a built-in IR blocking filter. The TCS3472 includes an integrated infrared (IR) blocking filter, which reduces the influence of IR light on color measurements, enhancing the accuracy of color detection.

Huskeylens

The huskeylens is an AI vision sensor that is able to detect objects, colours, etc. It features many AI algorithms like object tracking and colour recognition. We chose it firstly because the huskeylens has a built in object and colour recognition, which can be used to help detect the obstacles during the obstacle challenges. Secondly, it has a built in screen at the back of the sensor to allow easy use during testing. Lastly, it is very compact and portable, which allows easy installation on our bot.

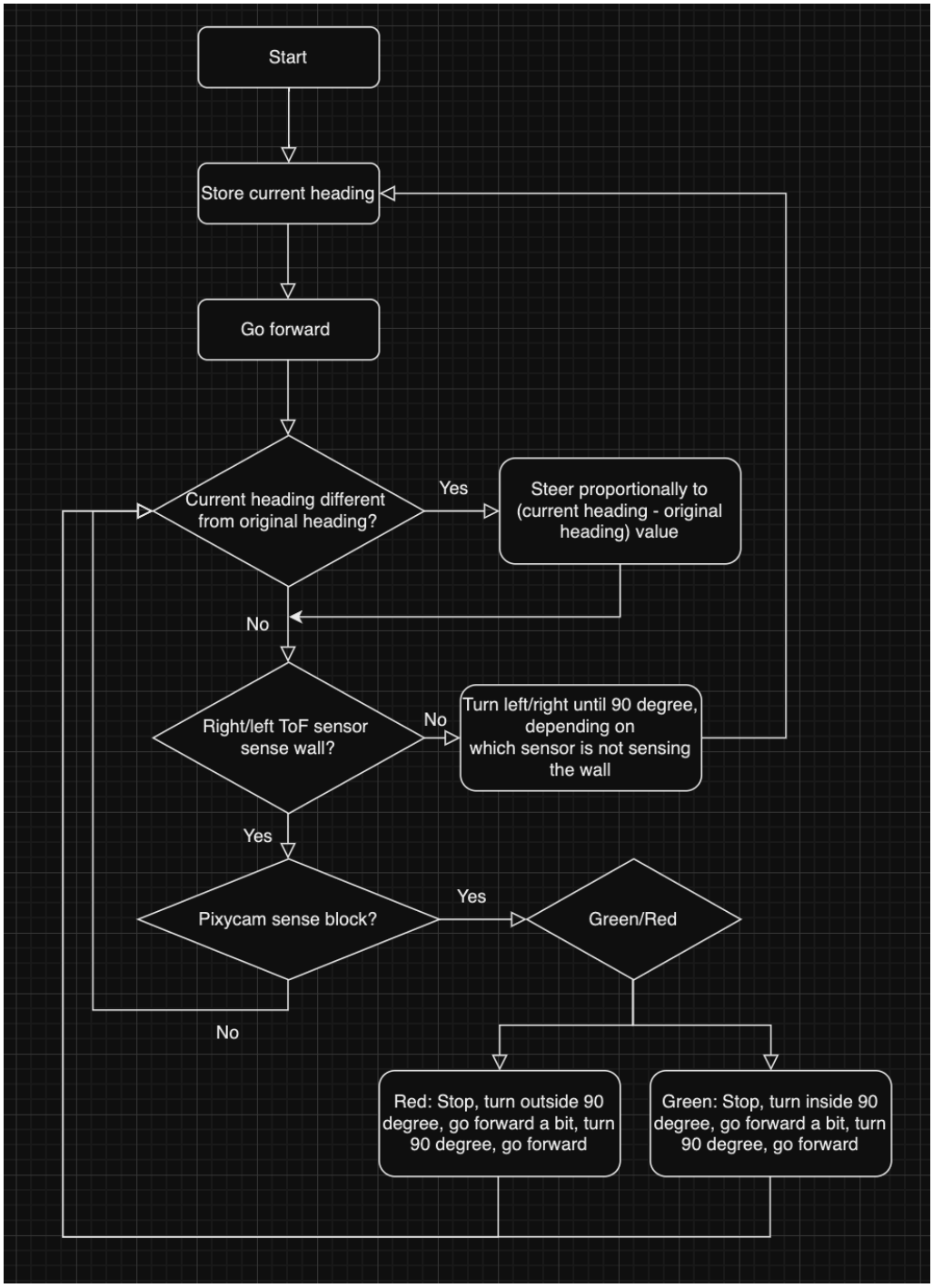
Strategy

Flowchart for preliminary rounds



For the preliminary rounds, we plan to make use of the gyroscope and time of flight sensor, which is attached to the front side of the bot. We would use the gyroscope to ensure that our heading is correct so to ensure that we would always go in a straight line. The time of flight sensor is used to sense the wall, when one of them do not sense the wall, the bot would turn in the direction of that sensor.

Flowchart for obstacle challenge



For the obstacle challenge, we would still use most of plans from the preliminary rounds. However, we added more such as pixycam. The pixycam is used to detect the colour of the block, where if it is red, it would turn outwards, and when it is green, it would turn inwards.

Code

In order to ensure that our code was understood by everyone by the team, we made readability our number one priority - and ensured that our code had comments that labelled each part. This ensured that despite the complexity of the autonomous robot system, it could be broken down into various segments.

With this, we ensured that our code maintained cleanliness by:

- Leave code with comments at every part
- If there are any large chunks of code, they are declared at the start but written at the bottom (i.e. functions).
- Utilise libraries and classes to declutter the main code and include them at the start
- Ensure that naming of variables and functions are clear as to what they do and what they are