MELBOURNE HIGH SCHOOL


Starr Yang
Mechanical
Software


Brendan Liang
Software
Strategy


Taj Gordon
Mechanical
Documentation


Bao Nguyen
Software
Vision

GitHub

光 HIKARI 光

Australian Team for RoboCup Junior Open Soccer 2025

Melbourne High School was supported by the Australian Government Department of Industry, Science and Resources through the Sponsorship Grants for Student Science Engagement and International Competitions.

> Abstract

We are HIKARI, a team of four students from Melbourne High School in Australia, participating in RCJ Open Soccer. The team was formed near the start of this year, when Starr - having entered the 2024 Australian Open by himself - realised he needed a team to be able to compete at the international competition. The team came together and bonded over a shared love for robotics, not having known each other beforehand.

We used Fusion 360 to design our bot and programmed it in Python. The design was heavily inspired by our design from last year, with similar key components - such as the Raspberry Pi 5 at the heart of both bots.

Key improvements we made to the design this year include a hyperbolic mirror - rather than a sky-mounted camera, which proved to be unreliable in last year's competition - and the usage of a solenoid kicker.

To accommodate the smaller robot size (180mm rather than 210mm), the four DJI RoboMaster M2006 motors we were using last year for the drive base were replaced with modified DJI RoboMaster M3508 (see Testing section for further details), which required us to create custom omni wheels.

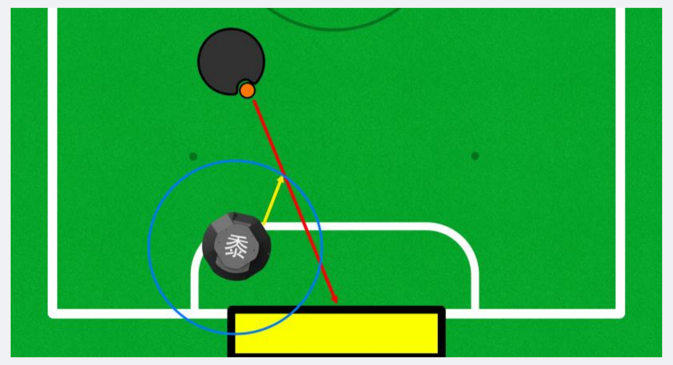
We additionally refactored and modularized the code, which allowed us to create a web interface to debug the robots - allowing for rapid iteration in design.

> Tools Used

We 3D printed the robot using designs created the robot in Fusion 360. The robot is coded in Python which we wrote in VSCode.

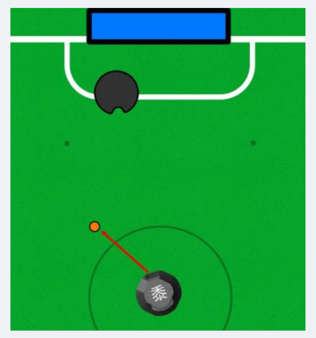
> Strategy

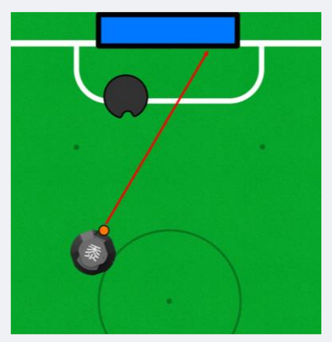
Defence



Our defence strategy is to stay inline with the ball, allowing the bot to quickly respond to any changes in the ball's direction.
Once the ball reaches a certain distance from the bot, the bot will begin chasing the ball until it is too far from its own goal.

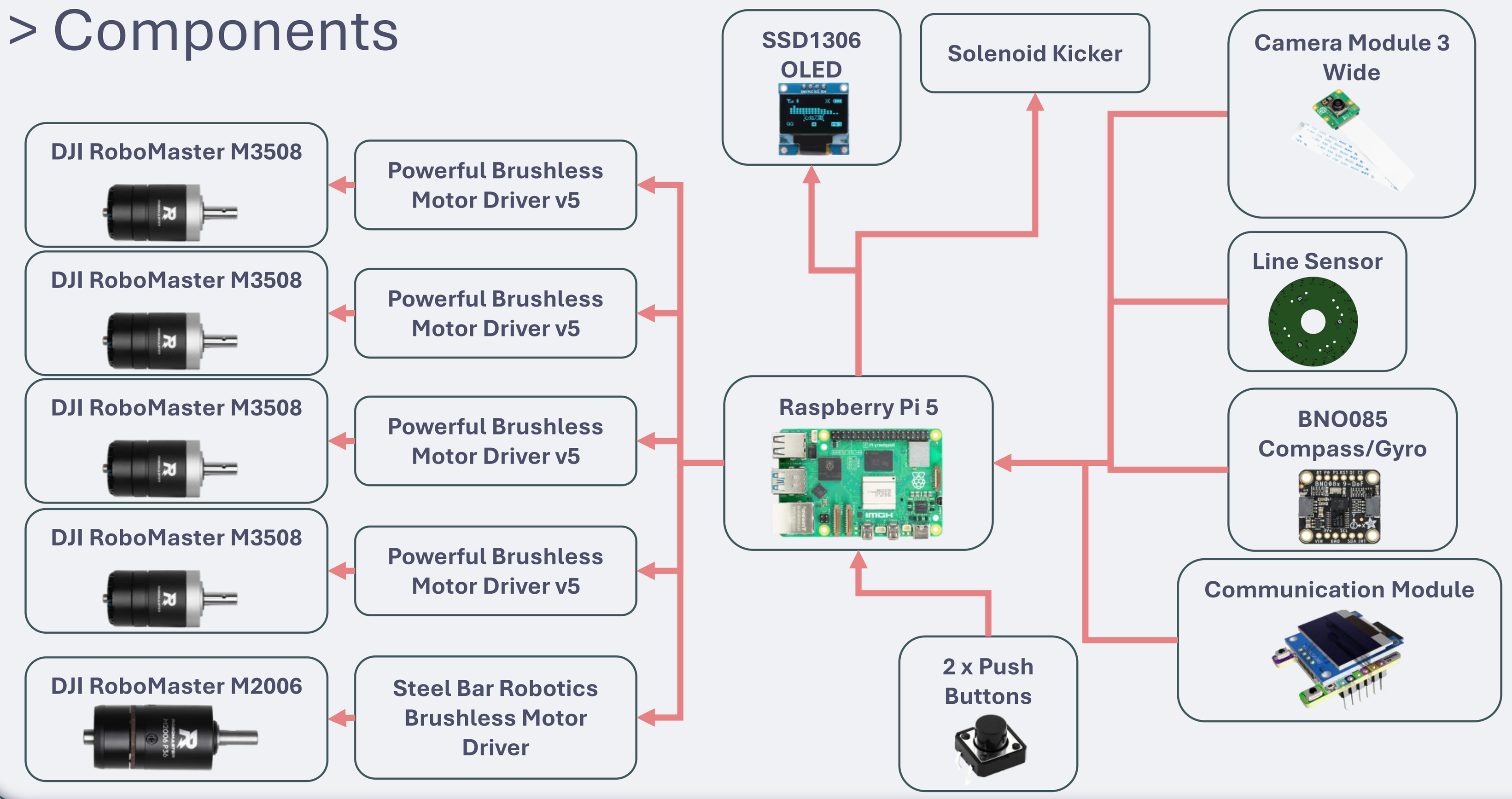
Attack

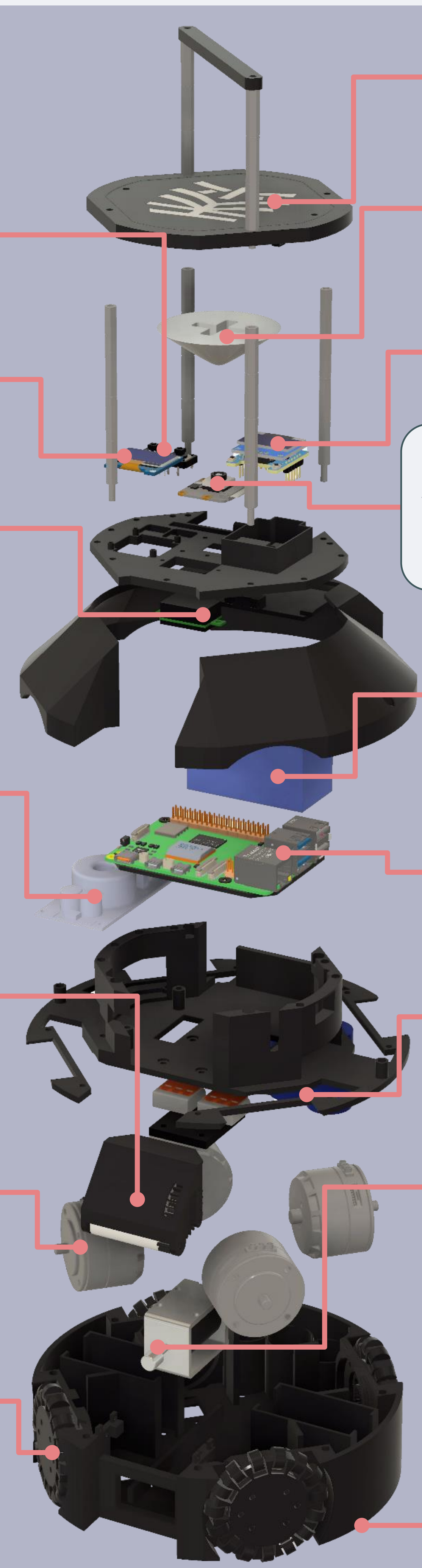




Our attack strategy is to simply chase the ball and gain possession with the dribbler. Once the camera sees that we have the ball, the bot turns to face the goal, keeping the ball close with the dribbler. The kicker then fires, taking a shot at the goal.

> Components





Buttons
For starting/stopping in regular environments

OLED Screen
Communicates useful information for debugging

Voltage Meter
Indicates the level of the connected battery

Boost Converter
Charges up capacitors to power the kicker

Dribbler
Silicone drum spun by a small motor and gears, used to control the ball

Modified DJI M3508 Motors
Powerful large motors with removed gearboxes for extra space

3D-Printed Omni-wheels
Custom wheels to fit modified motors while allowing omnidirectional movement

BNO085 IMU
Identifies important heading and acceleration information.

Hyperbolic Mirror
Polished out of aluminium for 360 degrees of vision

Communication Module
For starting/stopping in SuperTeam games

Raspberry Pi Camera Module 3
Works well with the Raspberry Pi 5

Turnigy 850mAh LiPo Battery
Saves space through small size, while having sufficient capacity

Raspberry Pi 5
Low-cost and high-performance computer

Capacitors
Builds up charge to fire the solenoid with a BANG!

Solenoid
Kicks the ball

Line Sensor
A ring of colour sensors under the bot to detect out-of-bounds areas

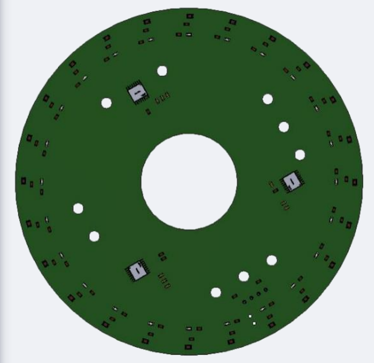
> Hardware

/ Custom Omni Wheels

We took out the gearbox off the motor to save space, exposing only a gear to mount wheels on. We designed and printed our own wheels, along with adapters with precise tolerances that we friction fit onto the motor's gear. The adapters have two grub screws to further secure them in place by pinching the gear.

* While this can work well, applying insufficient force to secure the wheels can lead to them flying off. Overall, this method is **not recommended.**

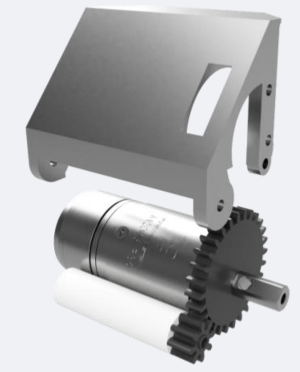
/ Custom Line Sensor



The line sensor is made of red LED/light sensor pairs on a PCB, multiplexed to send their brightness readings one after another via I2C.

/ Dribbler

Our dribbler uses a silicone drum, which provides high levels of friction to effectively apply backspin for ball control.



/ Kicker

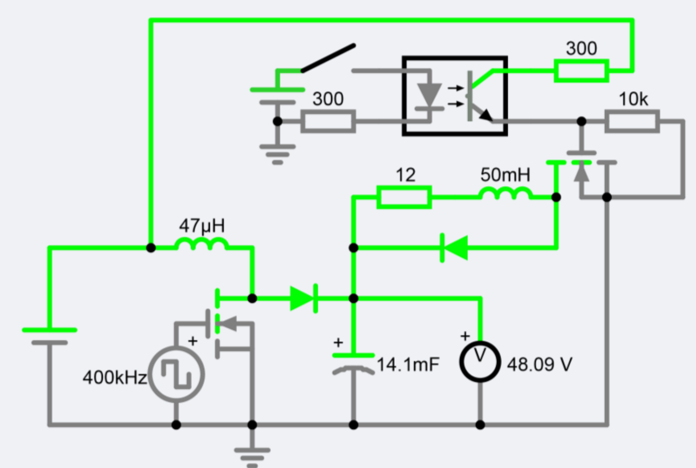


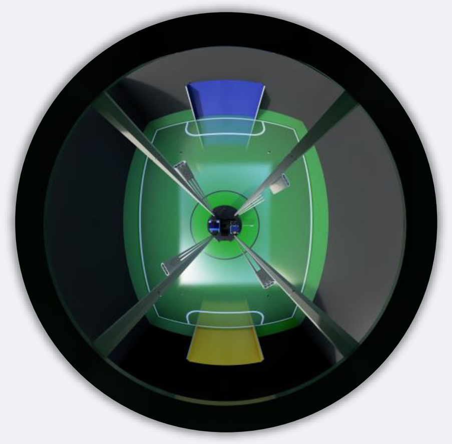
Figure 1: Kicker boost circuit to power the solenoid

Our kicker is a solenoid controlled by a Pi GPIO pin, which is represented by the switch in the circuit diagram (fig 1). We used an STPS10L60 diode to counter flyback, IRF1405 MOSFET and Adafruit's large push-pull solenoid. To power the MOSFET's gate pin with the Pi GPIO's weak voltage, we used the 4N25 optocoupler to drive it with battery voltage.

/ Vision

We use a Raspberry Pi Camera Module 3 Wide which rests on the top plate and faces up into a polished CNC mirror ordered from JLCPCB.

In the camera's code interface, OpenCV is used to find the centres of objects (coloured blobs) and a regression function is called to convert in-image distance to on-field distance.



Simulated view of the field through the camera and hyperbolic mirror

> Testing

We tested several different configurations before settling on our current drive system.

Table 1 shows the various iterations of motor and gearbox combinations to find a solution that was fast enough to be competitive, small enough to allow a large solenoid to be fitted, and reliable.

The custom gearboxes with the DJI M2006 proved unreliable and the M2006 by itself too slow, so the DJI M3508 was chosen and an improved omni wheel designed.

Motor	Needs Custom Omni Wheel	Size of solenoid supported	Speed
M2006	No	Small	Slow
M2006 with 2:1 external gear ratio	No	Small	Moderate
M2006 with 3:1 external gear ratio	No	Small	Fast
M2006 with custom plastic gearbox	No	Small	Fast
M2006 with custom CNC gearbox	No	Small	Fast
M3508 with factory gearbox	Yes	Small	Slow
M3508 without gearbox	Yes	Large	Fast

Table 1: Motors & gearbox combinations qualitative performance

Upon selecting the M3508, we needed to find a compatible motor driver. Table 2 summarises the configurations we tested.

The C620 motor driver communicates with the Raspberry Pi via CAN and supports two main functions: reading motor speed and setting motor current. Frequent speed readings were needed to control the motor using PID on current. Although the C620 offers a PWM mode with built-in PID, it's poorly tuned and lacks feedback.

We therefore tried standard CAN mode with USB-CAN-A adapters. The Waveshare adapter caused dropped and corrupted frames, while the OSHWLab adapter couldn't read fast enough for effective PID. Ultimately, we switched to the OSHWLab Powerful Brushless Motor Driver v5.

Connection to motor	Can read data	Reading speed	Packet loss
C620 in PWM mode	No	N/A	No
C620 with Waveshare USB-CAN-A	No	Slow	Yes
C620 with OSHWLab USB-CAN-A	Yes	Slow	No
Powerful Brushless Motor Driver v5	Yes	Fast	No

Table 2: Motor driver qualitative performance

Because our motors and solenoid take up a lot of space, we had little space on the robot left for the capacitors for the solenoid. To optimize the size of the components, we tested the solenoid with various capacitances. The results of the experiment in figure 2 indicate that there are diminishing returns and little further improvement beyond about 15000uF, so chose to use 3x 4700uF electrolytic capacitors, for a total of 14100uF of capacitance.

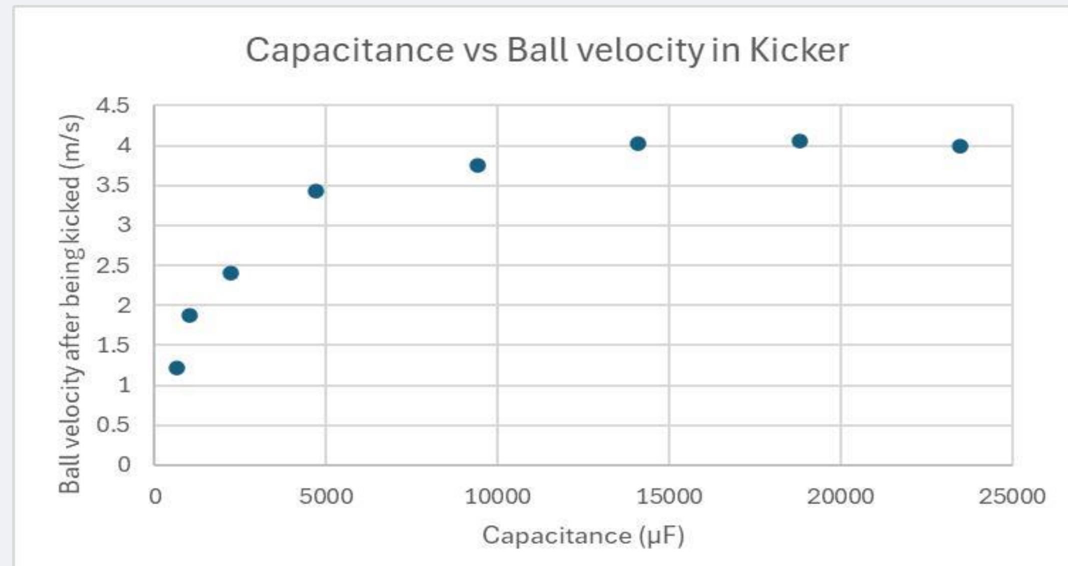


Figure 2: Kicker performance and capacitance relationship