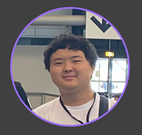
Edge

Soccer Light Weight

Japan





Kotaro Watanabe
渡邉　幸大朗
•PCB Design
•Team MGR
•X : @bbee7358

Yuta Nakagawa
中川　裕太
•Software
•X : @shiokara_rcj

Ryosuke Aoki
青木　良亮
•Hardware
•X : @makkasaaoki

Abstract & Concept

We are making robots as part of our after-school club in Japan. Last year, we achieved an **overall 4th place in the world tournament**, and this year, we won **1st place in the Japan tournament**. Last year, we were unable to achieve the ideal performance due to the instability of the **data output from the sensors**, the **low performance of the hardware**, and the **difficulty of making software changes**. Therefore, we strongly felt that stability was more important than adding new features. So, we aimed to **move quickly** and **accurately while ensuring stable operation**. This poster summarizes the added features, software, and experiments conducted for this purpose.

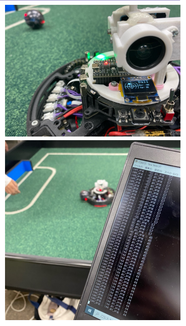
Lightning-Fast Debugging

UI

When the robot is behaving unexpectedly, we need to resolve the problem quickly. Therefore, we have made it possible to accurately **understand the information the robot is receiving without having to physically connect a PC**, by doing the following:

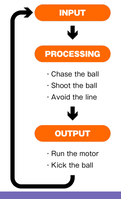
- Equipped the robot with a **buzzer** function to understand information audibly.
- Equipped it with **NeoPixels** to visually obtain information such as the angle of the ball and the angle of the line.
- Equipped it with a **display** for checking threshold adjustments, output adjustments, and bugs during the match.
- Equipped it with **Bluetooth** functionality to display information from the robot on a computer in real-time.

This has significantly sped up problem resolution.



Program Structure

To improve debuggability, We divided the entire program into three parts: **INPUT**, **PROCESSING** and **OUTPUT** of data. This dramatically improved debuggability, as it shortened the time to identify problematic code. Additionally, dividing the program made it easier to understand the overall flow of operations, making it easier to implement new features. This enabled us to efficiently perform the most important parts of software development: **debugging** and **implementing new functionalities**.



Reliable Circuit

We made the following improvements to achieve stable operation by creating a circuit that **does not break and malfunction** during matches.

Less Failure And Easier Troubleshooting
- Improved soldering skills and environment
- Created spare circuit boards
- Mechanism to replace line sensor boards in 30 seconds

Reduce Noise
- Converted to RS232C level for long-distance communication
- Used GND Solid pattern
- Keep wiring short and neat
- Added appropriate capacitors
- Added pull-up resistors to I2C

Experiment with pull-up resistors.

Avoid Human Error
- Implemented protection circuits
- Thoroughly performed insulation treatment
- Changed the shape and color of connectors based on the type of wiring

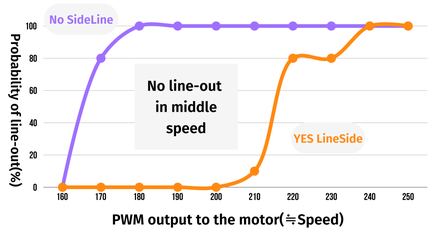
High-Speed Line Detection

When using a multiplexer to read multiple sensors with the main microcontroller, the line processing time was about 25 ms. By **adding a microcontroller and using a comparator**, all line sensors were **read simultaneously via registers**. Additionally, by converting the communication to RS-242C level, reliability was increased, and communication speed was improved, reducing the line processing time to approximately **2 ms (92% reduction)**. Currently, the mainstream shapes for line sensors are the Ring type and the Cross type, but to incorporate the advantages of both, we adopted the **Ring Cross type**.

	Ring type	Cross type	Ring Cross type
Location info	✓	✗	✓
Detection time	✗	✓	✓

Experiment To Verify The Effectiveness Of LineSide

To verify the effectiveness of the Ring Cross type, we investigated the relationship between speed and line-out when detecting lines using only the outer right line sensor (hereafter referred to as the LineSide) and the ring. The results showed that without the LineSide, the robot would go out of bounds when outputting a speed of 170 or higher. However, with the LineSide, the robot would not go out of bounds even at speeds of up to 210. Therefore, it was found that having the LineSide allows for an approximate **16% increase in speed**.



REPORT
MORE INFO

Capturing The Ball Using The Velocity Vector

Until now, we couldn't respond to fast ball movements, leading to situations where the ball was stolen or goals were scored against us. This time, we used the ball's velocity vector as a countermeasure. This allowed us to detect the ball's movement and respond to fast ball movements effectively. For example, for the defense robot, **when the absolute value of the velocity vector was large, we shortened the reaction time**, making it more responsive. By referencing the direction of the velocity vector in addition to the ball's position, we determined the optimal direction, thereby strengthening the defense. For the attack robot, its movements were determined by considering the ball's movement based on the velocity vector's angle.

The New Standard For Enemy Avoidance

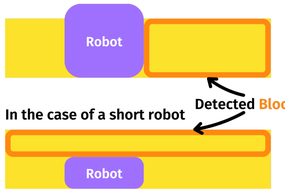
The previous system for bypassing enemy robots relied on shooting either left or right of the robot based on how much of the goal was visible in each. This assumes that the opposing robot was tall enough to separate our robot's view of the goal into two blocks. In the case of a shorter robot, this system was not able to recognize the goal properly.

This season, we implemented an enemy avoidance system that can **detect even low-height robots by utilizing the difference in color between the robots and the goal**. The details are as follows:

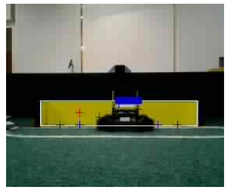
1. The color was acquired from the bottom of the block recognized as the goal.
2. The coordinates of the bottom surface of the goal color (marked with a black cross) were recorded.
3. The mark's position becomes higher where the robot is present.
4. The presence of a robot was determined where the height difference between adjacent marks is large (the part where the blue block on top of the goal is present).

This significantly improved the shooting accuracy.

Previous system

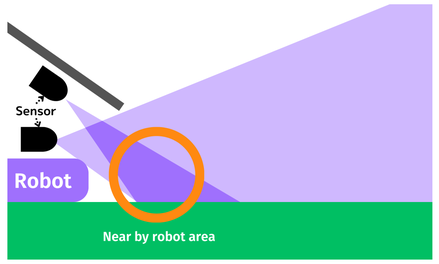


New System



Accurate Ball Position Estimation

In our competition, robots typically determine their actions based on the angle and distance to the ball. However, this method has a challenge: **the ball's rotation changes the infrared output angle, making it difficult to obtain accurate distance measurements**. This often leads to unstable robot movements. To address this issue in the current competition, we **added a sensor that detects the ball from above**, as shown in the diagram. This sensor reacts when the ball is nearby and doesn't react when the ball is far away, allowing for more accurate distance information. As a result, the speed of maneuvering around the ball **increased by approximately 40%.**



REPORT
Comparative experiment on the time it takes for the robot to circle around

Parts Diagram

Motor • RE16 19:1

With good responsiveness and fast initial speed, it was possible to **achieve movements that were in line with the program**.

MD PCB • BD63150AFM

Implementation of **multiple protection** circuits, capability to handle an instantaneous current of **6A**, ability to operate at a frequency of **100kHz** and **high cost-performance**.

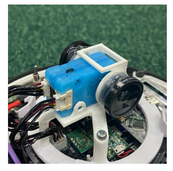
Battery
- Zeee 3S Lipo Battery 1500mAh 11.1V

Power PCB
- LM2576SX-5.0/NOPB (Generate 5V)
- Glass tube Fuse 10A

Line Sensor PCB
- Atmega2560(CPU)
- ADM3202(RC233C)
- B19H1LS(Sensor) x45
- OSHR1608C1A(LED) x48
- LM393(Comparator) x24

Kicker PCB
- XL6009(Generate 45V)
- AQW212(Photocoupler)
- TJ8S06M3L(MOSFET) x2
- Capacitor 2200uf x2
- Solenoid(CB10370100)

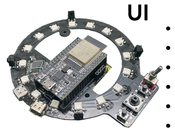
By using the AQW212 (photocoupler), which can operate at low current, it was possible to directly output from the Teensy, which can only supply low current, minimizing the number of parts.



Camera • UnitV x2

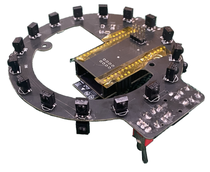
By switching from Pixy2 to Unit V, the **program's scalability increased**, enabling advanced techniques such as shooting to bypass enemies. Additionally, Unit V is relatively inexpensive, allowing for the attachment of two cameras. This provides **stable data values and ensures accurate information**.

UI
- ESP32(CPU)
- WS2812B(NeoPixel) x16
- PKLCS1212E4001(Buzzer)
- Display
- Tact-Switch ×3
- Toggle-Switch



Ball Sensor PCB
- Atmega32U4(CPU)
- TSSP4038(Ball sensor) x16 + 4

The ball sensor operates on pulse readings, so it was necessary to continuously monitor the sensor's value for a certain period of time. By adding a sensor microcontroller for this purpose, **the processing time for the ball on the main microcontroller was reduced**.



Main PCB
- Teensy4.1(CPU)
- BNO055(Orientation sensor)

By using the Teensy 4.1, development could be done in the Arduino environment, and it allowed for the use of **multiple UART** pins.

To Achieve Ideal Movements (Cost & Support)

	Costs
PCB	2,905
Circiut Parts	400
Motor	1,900
Other	500
Total	$5,705

We have spent about **two years evolving this robot**. The total cost incurred is approximately **$5700** (the breakdown is listed in the table).To achieve ideal performance, we have optimized hardware, software, and role distribution. However, to get even closer to our ideal, we faced financial challenges. **Achieving our ideal movements requires purchasing high-performance motors and spare parts**. Therefore, we emailed over 15 companies to request support. As a result, we received support from 9 companies. Now, **approximately 83% of the costs for the robot are covered by sponsors**. This support has allowed us to purchase high-performance motors and spare parts, enhancing our software and hardware mechanisms, and bringing our robots closer to ideal movements. We are deeply grateful for their support.

Ideal	To achieve it
Doesn't go out of line even at high speeds.	A highly responsive motor.
Multiple attempts	Funds to get failure.
Secure time for brushing up	Durable parts.

Access Me !

We are publishing data to improve our competitive skills. Please see the QR code below.

Circuit

Software

Python

Sponsor

DigiKey maxon TAKAHA Scramble JLCPCB HAKKO M5STACK ROHM SEMICONDUCTOR 株式会社ワイエムジー RITSUMEIKAN