RoboCup Junior Soccer LightWeight 2021

Team Description Paper Shinobi

Members: Rin Yoshida ,Yuki Okubo

Mentor: Taichi Shirayama

Institution: Nagoya Science City museum

Country: Japan

Contact details : taichirobo@gmail.com

Web page: https://robo2668.wixsite.com/shinobi-rcj

Abstract

All the parts of the robot we made this time were made of carbon to reduce weight, increase durability, and improve maneuverability. Also, the robot we made in last year's RoboCup Junior was programmed without using mathematics or physics, but this time we incorporated mathematics and physics to improve the maneuverability. The incorporation of physics and mathematics reduced the processing speed of the robot, so we installed a large number of microcontrollers to increase the clock frequency of the robot.

1.INTRODUCTION

Our team consists of two high school students. Rin Yoshida, creates the software and hardware, and Yuki Okubo, creates the software. Our team was created by Rin Yoshida four years ago. I participated in RoboCup Junior Soccer Lightweight three times in four years at the RoboCup Junior Japan Open. In this RoboCup, I made a robot as shown in Fig. 2. The robot in Fig. 1 is the robot created last year. In Fig.1, there was no uniformity or maintainability, and robot maintenance was extremely difficult. By creating a robot as shown in Fig. 2, the maintainability of the robot has been improved. Major changes from last year are the use of durable carbon for the plate and omni wheel that support the robot, the introduction of a circular line sensor, the introduction of an ultrasonic sensor to calculate the robot's self-position, and a large number of microcomputers. By installing a board, the clock frequency of the robot is not lowered. Last year, both robots were defensive, but this year we have divided them into defensive and offense. In order to create a robot suitable for defense and offense, we attached a cross-shaped line sensor to the defense and a circular line sensor to the offense.

Through these major modifications, the defense has made it possible to secure the ball with agile movements, control the attitude of the robot, and score reliable goals.

The offense made it possible to catch the ball reliably and prevent the goal coming from the opponent's team, and made it possible to trace the white line in front of the goal with a circular line sensor, making it possible to eliminate unnecessary movement.

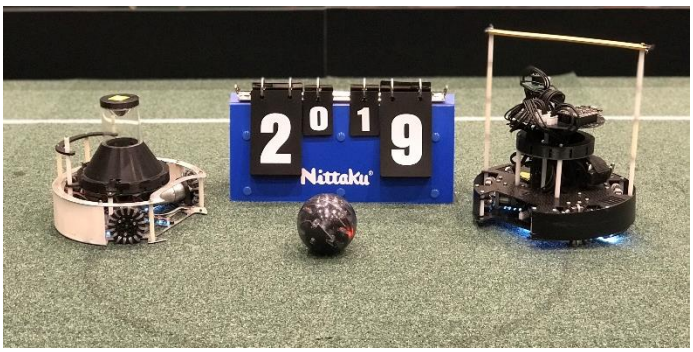


Fig.1

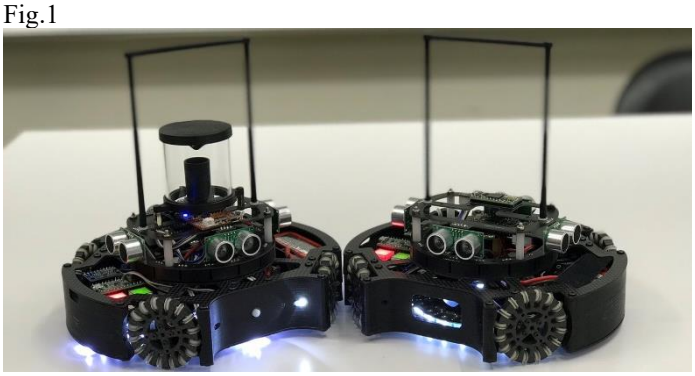


Fig.2

2. Hardware

Offensive	Individual
Teensy3.5	×2
Teensy4.0	×3
Mpu9250	×1
Jy901	×0
Hcwl-1601	×4
Tssp58038	×16
OpenMv H7	×1
CB1037	×1
Maxon 135531	×4
Pololu G2 Motor Driver	×4
Cross line	×1
Circular line	×0
Kypom 3S 1000mah	×1
Xbee3	×1

Defensive	Individual
Teensy3.5	×2
Teensy4.0	×3
Mpu9250	×1
Jy901	×1
Hcwl-1601	×4
Tssp58038	×16
OpenMv H7	×1
CB1037	×0
Maxon 135531	×4
Pololu G2 Motor Driver	×4
Cross line	×0
Circular line	×1
Kypom 3S 850mah	×1
Xbee3	×1

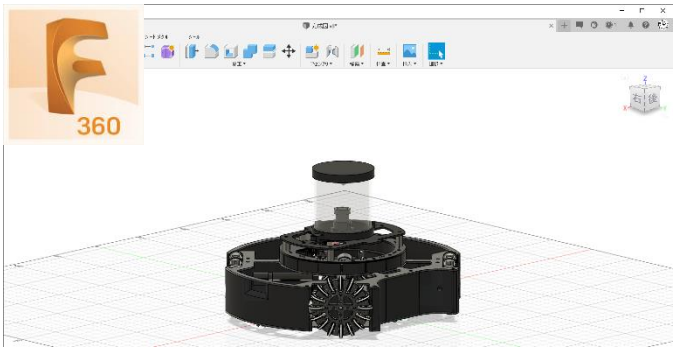


Fig.3,Fusion360

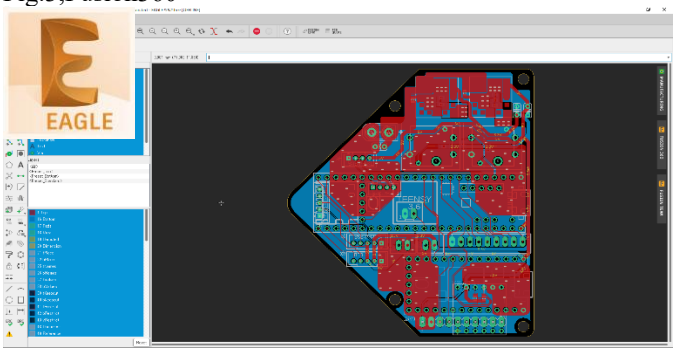


Fig.4, Eagle

Software used for design

Designed with Fusion 360 and Eagle for durability, maintainability and weight reduction.

As shown in Fig. 5 to Fig. 6, Fusion 360 applied a force from 0N to 100N to simulate durability and tested the durability, and thought about how light the carbon used in the robot could be.

By using Eagle, we have reduced the weight and size of the base. At Eagle, as a noise countermeasure, we made it through a bus line that does not cause crosstalk and made it ground solid.

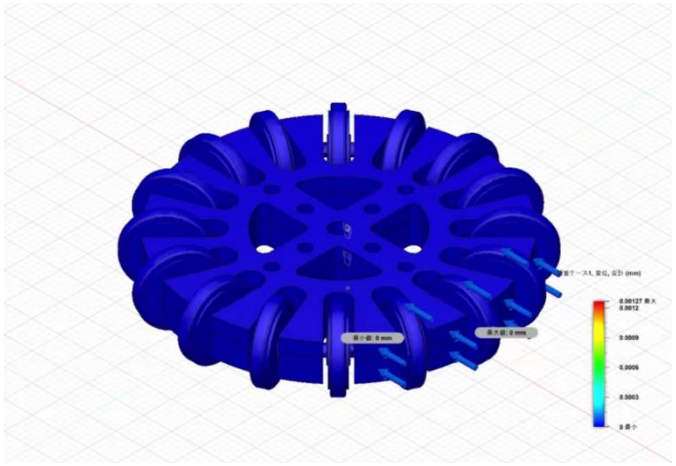


Fig.5 0N

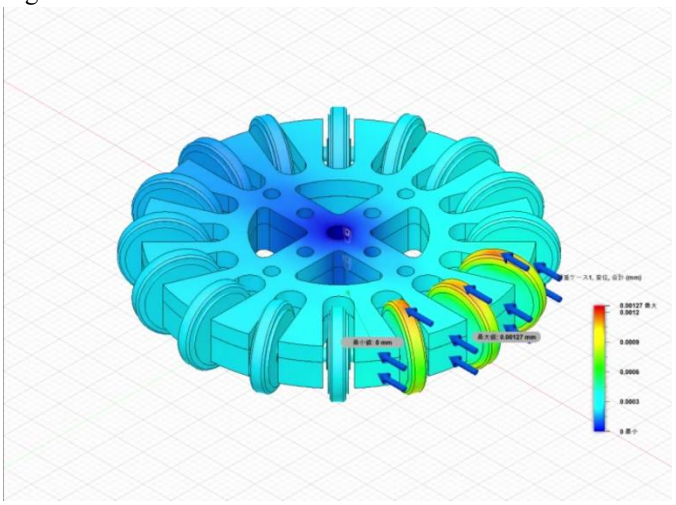
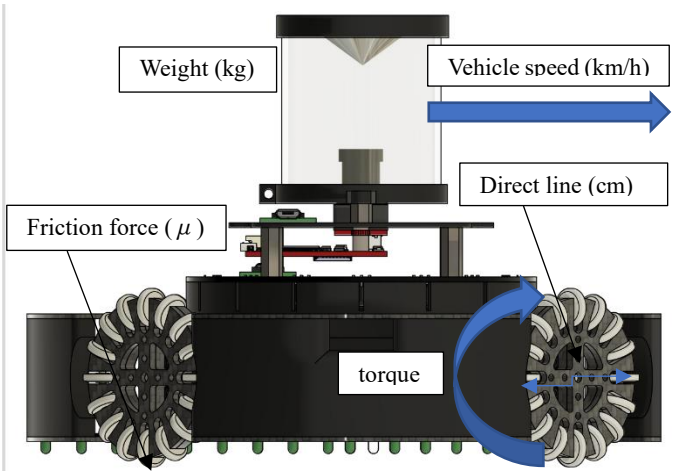


Fig.6 100N



Motor selection method

We will calculate the torque of the motor. Torque is a quantity (moment) expressed as the product of force distances (vector product).

The formula for producing torque can be calculated from the weight of the robot, the radius of the drive wheels, and the coefficient of friction. The formula is (1) listed below.

Our team created a four-wheeled robot, so the weight is distributed to four points. So divide the weight by 4.

Next is the speed of the motor. rpm is a value that indicates the amount of motor rotation per minute.

This is speculation from our experience, but I think that RoboCup will require about 600 to 1000 rpm. So we are using a 731rpm motor.

Torque(kgf.g)=(Weight(kg)/4)*Frication(μ)*(Direct(cm)/2)
・・・(1)

Battery selection

Calculate the total power consumption of the robot using the formula (1) shown below. Next, calculate the watt hour (Wh) using the formula (2). It can then be calculated using formula (3) to convert kWh to mah.

Voltage(V)* Current(A)= Electric power(W) ・・・(1)

Electric power(W)* Match time(s)= Watt hour(Wh) ・・・(2)

Watt hour(Wh)*1000/ Voltage(V)=mah ・・・(3)

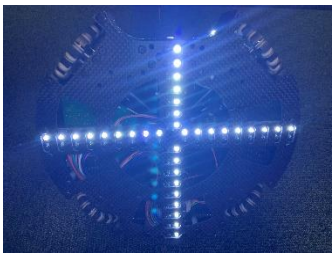


Fig.6 cross line

Fig.7 circular line

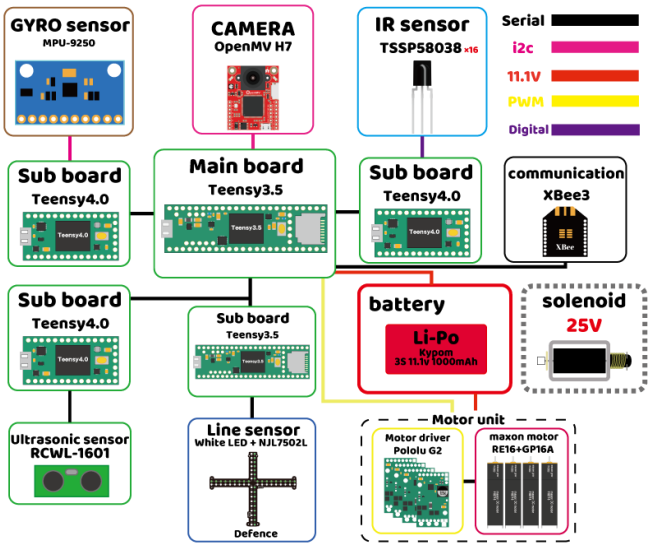
Line sensor

Our team divides the line sensor for reading the white line into a cross shape and a circular shape.

The advantage of the cross-shaped line sensor is that it reliably reads the line and does not autobahn even at high speeds.

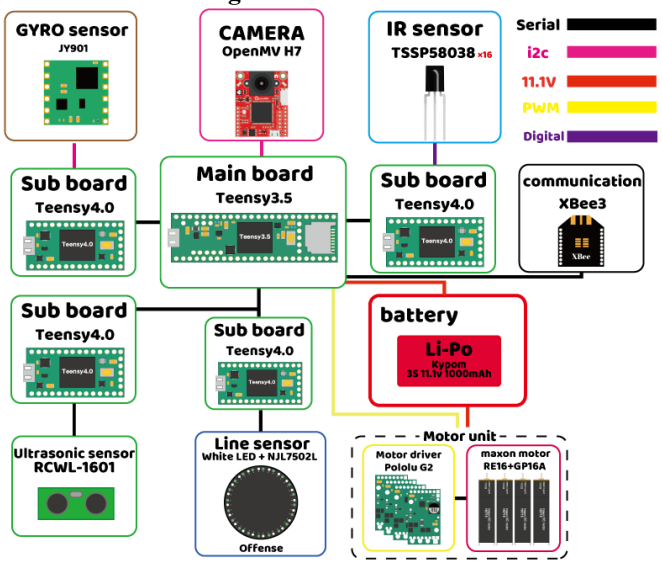
The advantage of the circular line sensor is that you can see the position of the white line you are touching. So, in order to take advantage of these advantages, our team has installed a cross-shaped line sensor on the fast offensive robot and a circular line sensor on the defensive robot that traces the line in front of the goal. By introducing a circular line sensor, it is possible to trace the line in front of the goal smoothly, and it is possible to cut the shot from the opponent at extremely high speed and the shortest distance. The cross-shaped line is based on the line that was touched first, and within a certain period of time, even if it touches any place on the front, back, left, or right, it moves to the target axis side of the position that was touched first, and the robot does not reliably autobahn Was created.

Offensive circuit diagram



We succeeded in increasing the speed of the robot by installing a large number of microcomputer boards. Between the sub-microcomputer and the main microcomputer of the cross-shaped line sensor, we created a robot that does not autobahn by enabling fast processing using interrupt processing.
Equipped with MPU9250 and OpenMV, it is possible to face the opponent's goal direction. The MPU9250 calculates the yaw axis of the robot's global coordinates so that the robot always faces forward, and OpenMV reads the color of the opponent's goal, making it possible to avoid the opponent's robot and reach the goal.
In addition, we combined a solenoid and OpenMV, measured the distance to the opponent's goal with OpenMv, decided the timing to operate the solenoid, and made it possible to hit a goal with high certainty.
The TSSP53038 reads the infrared ball, and the ultrasonic sensor can calculate the self-estimated position.

Defensive circuit diagram



Like the offensive robot, the defensive robot is equipped with a large number of microcomputers to ensure that the goal from the opponent can be cut.
The communication between the line sensor and the main microcomputer is different from the attacking robot, and serial communication is performed at a baud rate of 250,000 in order to send the angle of the white line touched by the robot to the main microcomputer.
The JY901 is installed to face the front like the MPU9250 installed in the offensive robot.
Open MV is reading the color of its coat. By reading the color of your coat, you can calculate the range of coats you protect.
The ultrasonic sensor and TSP53038 are installed for the same reason as the offensive robot.


Fig.8


Fig.9


Fig.10

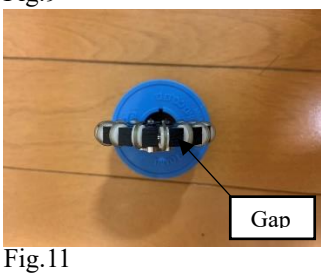


Fig.11

Omni wheel durability test

Our team created omni wheels from a variety of materials and investigated which material was suitable for RoboCup.
Fig. 8 shows a PLA omni wheel made with a 3D printer, Fig. 2 shows a ready-made duralumin (2017) omni wheel made with a CNC milling cutter, and Fig. 3 shows a carbon omni wheel made outsourced.
The first one I used, the PLA omni wheel shown in Fig. 8. This omni wheel was lightweight and easy to mass produce, but we stopped using it because it could crack during the match.
Next, we used the ready-made duralumin omni wheel shown in Fig. 9. This off-the-shelf omni wheel was light and cheap to buy, but we stopped using it because the part shown in Fig. 11 is easy to bend and the side wheels may come off.
The last one I used, the carbon omni wheel shown in Fig. 10. Carbon omni wheels were expensive because they were outsourced, but we used carbon to achieve both strength and weight reduction at the same time.
Although carbon has a lower density than aluminum, it has strong tensile strength and specific strength, so we thought that weight reduction and strength increase would be effective, so we adopted carbon.
The side wheels are durable, with durable urethane sandwiched between wear-resistant POM.


Omni parts


Carbon omni wheel


Fig.12

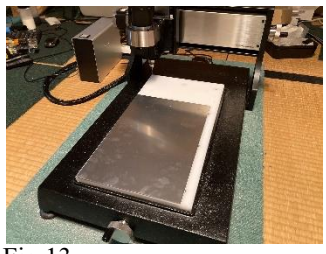


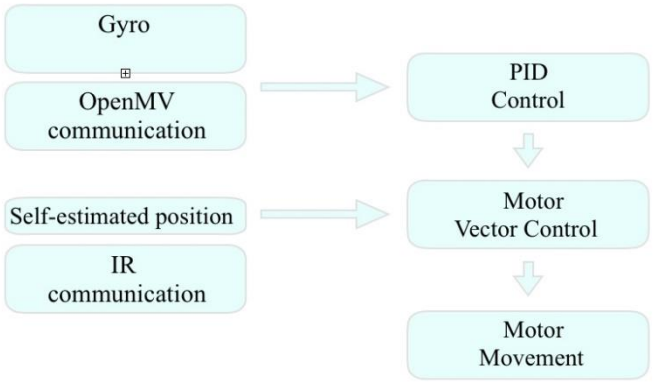
Fig.13

Work equipment

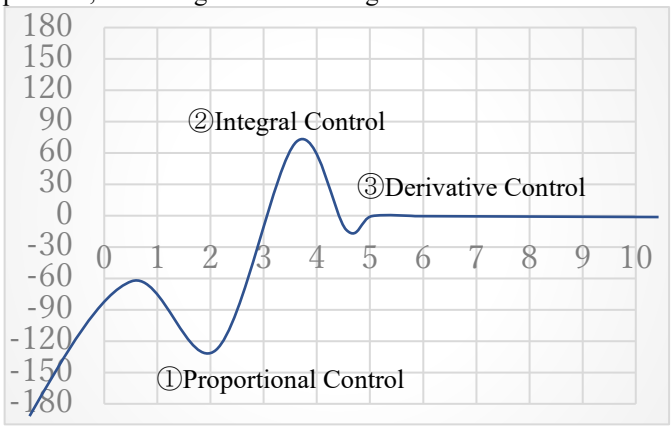
I have a 3D printer shown in Fig. 12 and a CNC milling cutter shown in Fig. 13 at home. Having these machines at home has greatly increased the speed of robot production. Our team uses 3D printers to create motor fixtures and robot covers. When introducing a new idea to an existing part, it is efficient because it can be immediately created with a 3D printer and mounted on a robot.

3. Software

Program flow chart



The robot software first communicates with the gyro via I2C and calculates the posture angle from 0 degrees' front to 180 degrees' rear and from 0 degrees' front to -180 degrees rear in a clockwise direction with the origin on the xy-plane as the robot's position, and sets 0 degrees as the robot's front.
PID control is performed based on this gyro threshold to quickly approach the target value (the front of the robot).
We use openMV to ensure that the goal is scored. The moment the robot catches the ball, it reads the color of the opponent's court and changes the setting of the attitude angle (target value).
The threshold value calculated by the PID control is assigned to the vector control of the motor so that the posture can be controlled while the robot is moving.
The local coordinates of the ball are calculated using a trigonometric function based on the pulse signal read by the infrared sensor.
The ultrasonic sensor calculates the robot's self-estimated position, extending the robot's range of motion.



Fig,14

PID Control

PID control is to approach the target value as quickly as possible.
PID control consists of proportional control + integral control-differential control.
Proportional control can be calculated from the constant of proportionality x target value-current value, and the manipulated variable is swiftly reacted to quickly approach the target value as shown in Graph 1 of Fig. 14.
Integral control can be calculated by integration constant x target value-current value x one cycle of the program, and the deviation is accumulated over time, and when the accumulated amount reaches a certain size, the operation amount is increased. Eliminate the deviation and operate it as shown in Graph 2 in Fig. 14.
Derivative control can be calculated by differential control × (target value-current value) -(target value-past current location) ÷ one cycle of the program, comparing the current deviation with the previous deviation, and the magnitude of the deviation The operation amount is swiftly reacted and the vibration phenomenon is suppressed as shown in Graph 3 of Fig. 14.

The proportional constant, integral constant, and differential coefficient of PID control set the gain suitable for the robot.
Our team substituted various values and repeated the experiment to find a value that quickly converged to the target value. The experimental results are shown in the table below.

P Gain	Overshoot
1.10	×
1.20	×
1.30	×
1.40	×
1.50	×
1.55	×
1.60	○
1.65	○

I Gain	overshoot and undershoot
0.10	×
0.20	○
0.30	○
0.40	○
0.50	×
0.60	×
0.70	×
×	×
×	×

D Gain	Convergence
0.05	×
0.10	×
0.15	×
0.20	○
0.25	○
0.30	○
0.35	○
0.40	×

Result

Proportional gain(Kp)=1.64

Integral gain(Ki)=0.12

Derivative gain(Kd)=0.23

Motor Vector Control

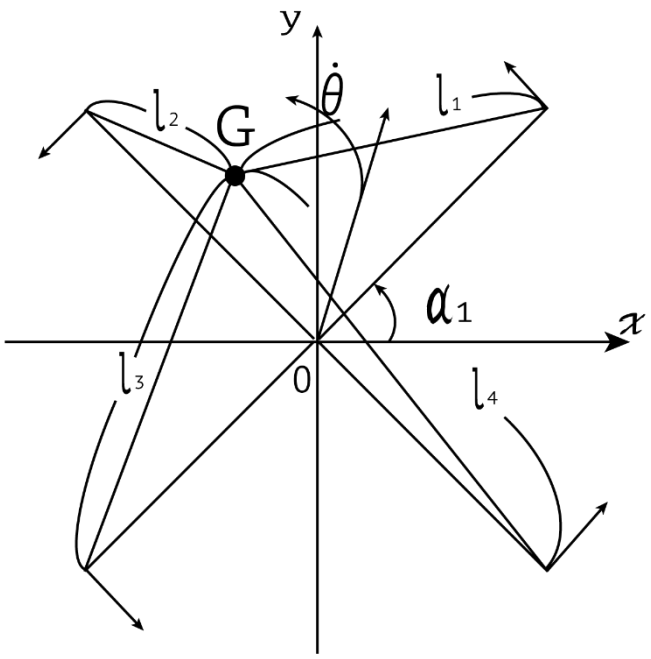


Fig.15

Fig.15 shows the coordinate system and velocity components of a mobile robot. Let G be the center of gravity of the mobile robot, O be the center of rotation around the center of gravity, li be the distance from the center to each omni wheel, O be the center of the main body, and l be the distance from the center to each omni wheel. Considering the xy plane with the center as the origin, let Vi be the translational speed generated by the rotation of the omni wheel placed in each quadrant. Also, let a1 be the angle formed from the x-axis with respect to the omni wheel 1. Vi-θ is positive counterclockwise.
Assuming that the velocity vector of the mobile robot shown in Fig. 15 is V, the velocity components in the x and y directions are Vx and Vy, and the relational expression of the velocity Vi of each omni wheel is as follows.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} -\sin\alpha_1 & \cos\alpha_1 & l_1\cos\beta_1 \\ -\cos\alpha_1 & -\sin\alpha_1 & l_2\cos\beta_2 \\ \sin\alpha_1 & -\cos\alpha_1 & l_3\cos\beta_3 \\ \cos\alpha_1 & \sin\alpha_1 & l_4\cos\beta_4 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \dot{\theta} \end{bmatrix}$$

queue

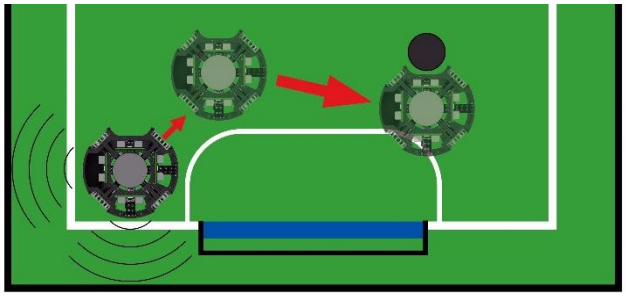


Fig.16

Ultrasonic self-estimated coordinates

This section describes how to calculate the estimated coordinates using the ultrasonic sensor.

First, in order to calculate the abscissa, we compare the left and right sides of the ultrasonic wave and find the threshold maximum value.

The important thing here is to calculate the threshold maximum value.

If it is simply a matter of calculation, comparing the distance of symmetrical ultrasonic waves and measuring the shortest value is effective in reducing the error. However, this estimation method does not take into account the other side's aircraft. When the other side's aircraft is next to yours, the distance to the wall cannot be measured correctly if the minimum value is calculated.

As a way to deal with this, the distance to the wall is calculated based on the maximum value.

Next is the vertical axis. The vertical axis is calculated based on the horizontal axis, because the coordinates of the vertical axis will be different depending on whether there is a goal area or not. From these coordinates, the estimated coordinates of the aircraft are calculated.

Fig.16 shows an example of an action using the estimated coordinates.

If the aircraft is at the edge of the court and the ball is on the target axis, the aircraft will try to move to the right. If it moves to the right, it will touch the line and return to the left. It keeps repeating this movement, so it can't follow the ball. This is where I used the estimated coordinates. When the ball touches the line, it changes its direction of movement based on its estimated coordinates.

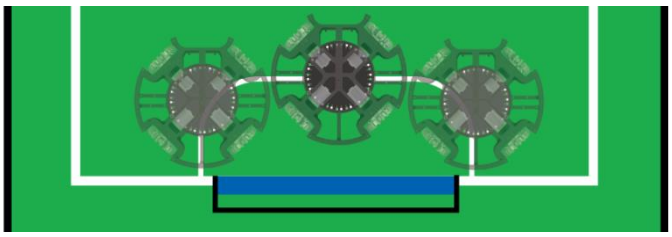


Fig.17

Overwhelming defense using self-estimated coordinates

We have created a defense machine that can cut off an opponent's solenoid shot.

No matter what kind of shot comes, the robot can run on the line in front of the goal to supplement the ball, as shown in Fig. 17

Even when the robot is pushed backward, it moves forward to prevent itself from being outburned.

Also, it will be pushed out of the way by the opponent. Or after an outburst, even if the robot is placed at the neutral point, we made it possible for the aircraft to return to the line based on the estimated coordinates.

However, just running on the line is against the rules, therefore we took countermeasures by making the robot move forward if there is a ball in front of it for a certain period of time.

By creating this robot, we were able to take advantage of pushing, which allowed us to place the ball at the opponent's neutral point, so that the attacking robot could supplement the ball and score a goal.

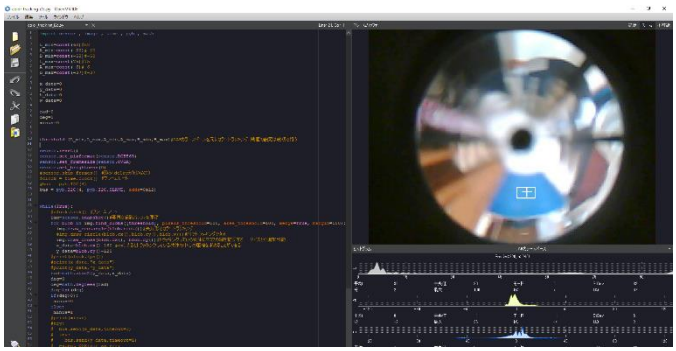


Fig.18

Color Tracking with Open MV

Fig.18is an image of color tracking using OpenMV.

By using an omnidirectional mirror, you can see the inside of the court 360 degrees. The coordinates to be tracked are calculated, the threshold is arctangent, and the angle is calculated.

The calculated angle binary is sent by I2C to the main microcomputer and assigned to the PID control expression. However, Open MV image processing is slow and the average output is only 30fps.

If the processing speed becomes slower, the binary transmission speed will decrease and the clock frequency of the main microcomputer will decrease. As a countermeasure against this, the main microcomputer requests the binary from OpenMV only when the robot catches the ball.

By doing this process, we were able to improve the clock frequency of the main microcontroller that runs the robot. By increasing the clock frequency, we were able to increase the mobility of the robot.

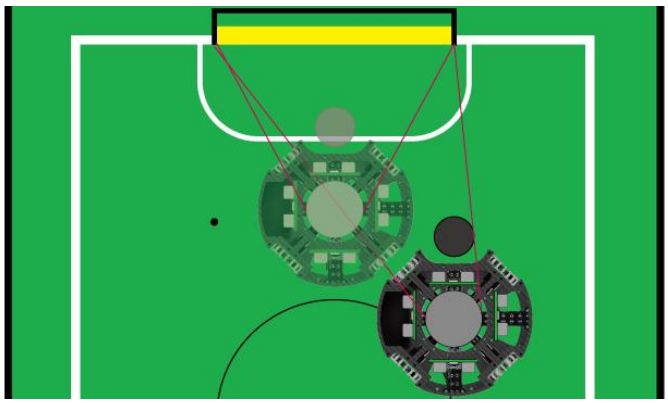


Fig.19

Fig. 19 shows the behavior using Open MV, which reads the color of the opponent's goal and recognizes the object. By doing this, the angle of the goal from the robot is calculated and the angle is assigned to the PID control. As shown in Fig. 19 the robot is able to score a goal from the edge of the court, where it would not normally be able to score a goal, by correcting the angle of the robot with posture control

We also made it possible for the robot to avoid the opponent's robot and score a goal even if the opponent's robot was in front of the goal, thus advancing the game.

Goal by solenoid

Our team introduced solenoids to the offensive robot and incorporated a mechanism that allows it to kick. By doing this, we were able to evade the defending robot and score a goal.

The timing to drive the solenoid was determined based on the area of the object recognized by Open MV color tracking.

In the case of larger area, the closer the object is, and in the case the farther the area, the farther the object.

Once the robot senses that it has caught the ball, it communicates with Open MV. At that time, the distance between the goal and the robot is measured, and the timing of the solenoid shot is determined.

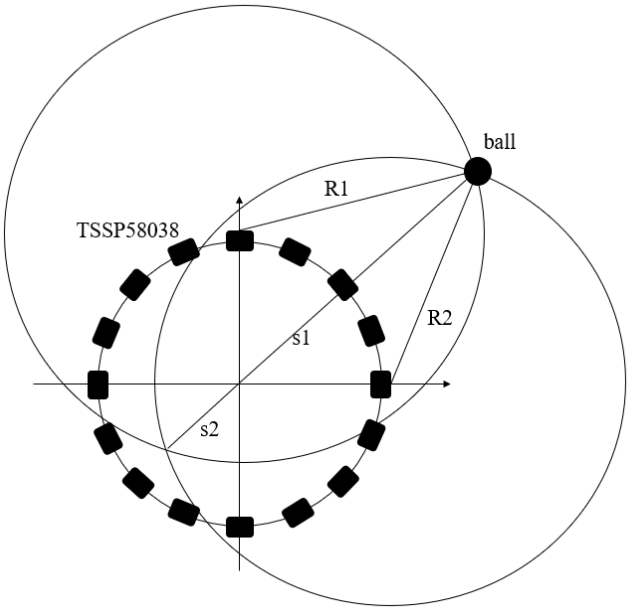


Fig.20

Robot approach to a ball experiment

We figured out a way to calculate the X and Y coordinates of the ball. Our team placed an infrared receiver (TSSP58038) in a circle in the XY plane as shown in Fig 20. 16 sensors and two long values R1 and R2 are calculated and the distance between the sensors and the ball is represented by a straight line. Then, with the infrared receiver, draw a circle with the sensor as the midpoint and the distance to the ball R1 and R2 as the radius. By doing so, the two circles will intersect at two points. The distance between the two points from the origin is S1 and S2, and the longer distance is the direction of the ball. This process should theoretically give us the X and Y coordinates of the ball, but since the distance between R1 and R2 was unstable, it didn't work and we never used it in the game.

The next method I thought of was to calculate the straight line distance between the ball and the 16 infrared receivers, divide the 16 decomposed straight line distances into X and Y components using trigonometric ratios, and then combine and arctangent them to calculate the angle.

This method of calculating the angle was more accurate than the first method of calculating the coordinates of the ball, so we decided to use this method for the match.

The ball angles and distances calculated from these robots were used to determine the robot's detour.

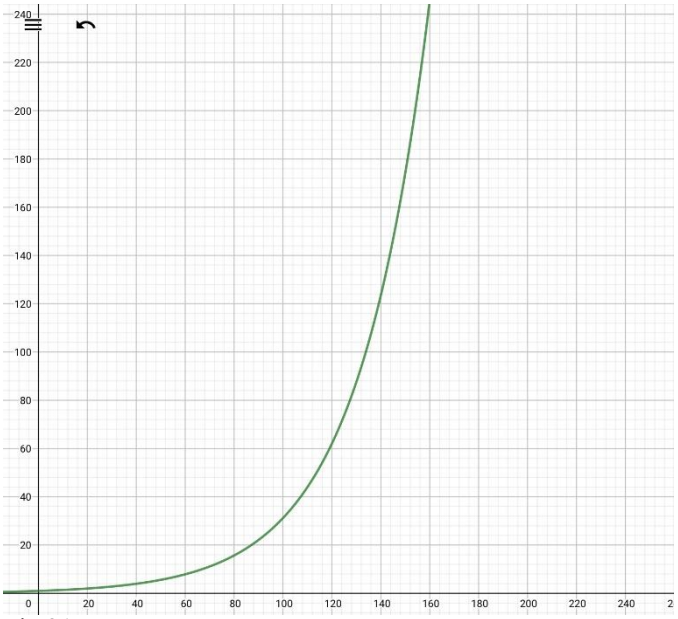


Fig.21

Robot's approach to the ball

The moment the robot sees the ball, it will turn around and go for it. Fig. 21 shows one of the examples of the wrap-around motion I am using. In our robot, the origin on the xy-plane is the position of the robot, and the position of the ball is calculated from 0 degrees in the front to 180 degrees in the rear, and from 0 degrees in the front to -180 degrees in the left, with 0 degrees being the front of the robot. The distance from the robot to the ball in a straight line is also calculated along with this angle. From these values, we calculate the robot's wraparound. Specifically, as shown in Fig. 20, the angle of the ball's position is multiplied by an exponential function to perform the wraparound. By doing this, when the angle of the ball is between 0 and 20 degrees, the angle in the specified direction in which the robot will move will be as close to zero as possible, and when the angle is between 20 and 180 degrees, the angle in the specified direction in which the robot will move will be larger, making it possible to perform a smooth wraparound. The adjustment of the exponent of this exponential function is calculated from the distance of the ball. When the ball is close, the exponent value is increased, and when it is far, the exponent value is decreased. By doing this, when the robot is close to the ball, it will perform a large wraparound, and when the robot is far from the ball, it will perform the wraparound in the shortest distance.

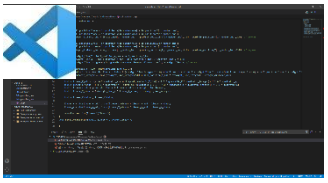


Fig.22

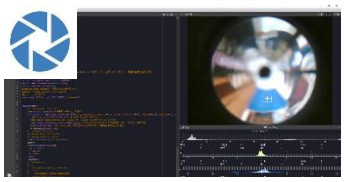


Fig.23

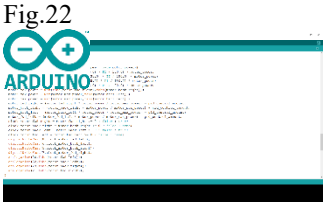


Fig.24

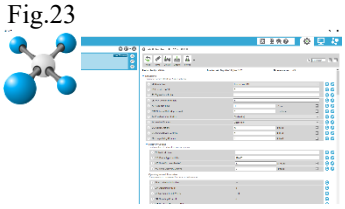


Fig.25

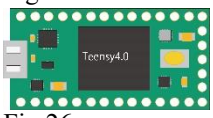


Fig.26

SoftWare used

Our team is using Vscode (Fig 22), OpenMV IDE (Fig 23), Arduino IDE (Fig 24), XCTU (Fig 25), and Teensyduino (Fig 26) to create the software. The Arduino program is the main development environment needed to run the robot, the OpenMV IDE is the development environment needed to incorporate the color tracking software, XCTU is the software used to configure the XBee3, and Teensyduino is an add-on to the Arduino IDE that allows you to make sketches for Teensy. Teensyduino is an add-on to the Arduino IDE that allows you to make sketches for Teensy.

Gamma di frequenza PWM

We experimented with PWM frequency bands for our motors. The results of our experiments showed that the lower the frequency band, the thicker the pattern width of the base, the larger the fluctuation of the current value, and the more noise is generated, resulting in lower power efficiency.

The higher the frequency band, the smaller the current value, and since there was no loss of speed or torque with the higher frequency band, our team set the PWM frequency band to 80Khz, thinking that we could use it without overloading the motor. The increased power efficiency meant that less power was consumed, allowing the robot to run longer and consume less battery power.

4. What I learned from making robots

In terms of hardware, what I learned in building this robot is the various communication methods of robots. I learned about serial communication, I2C, SPI, and other communication methods in the process of building the robot.
In building the robot body, I learned how to create the base and how to create the CAD. In software, I learned two things. First, I was able to learn about mathematics such as differential and integral calculus, matrices, vectors, trigonometric functions, exponential functions, and logarithmic functions in order to introduce PID control, vector control of motors, and the robot's approach to the ball. Secondly, I was able to learn the C programming language and Micro Python. In addition to learning these programming languages, I studied assembly too, which gave me a deeper understanding of embedded programming.

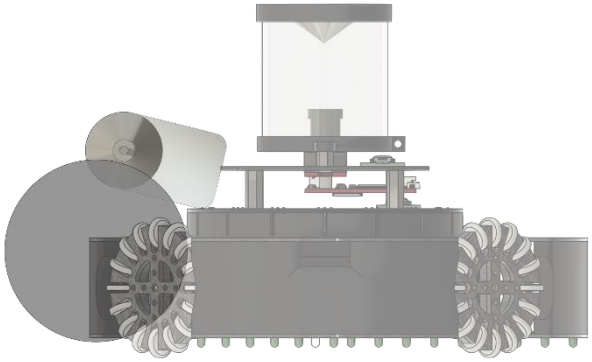


Fig.26

5. Future goals

Our future goal is to equip the robot with a dribbler, as shown in Fig. 26. A dribbler is a brushless motor that rotates a stick of rubber or other material with a high coefficient of friction at high speed. By attaching the dribbler to the part of the robot that holds the ball, the robot can move back and forth, left and right, while holding the ball.
By using the dribbler and the Open MV currently installed in the robot, we can ensure that the ball at the edge of the court is brought to the goal.
However, to make this happen, the weight of the robot has to be reduced. Our robot has a mass of 1050 grams. With 50g, the robot cannot be equipped with a dribbler. We are currently thinking of three ways to reduce the weight of the robot, the first being to make the edges of the carbon, the body of the robot, thinner. Current robots have thicker edges for strength. So by making the edges thinner, we can make the robot lighter.
The second is to consolidate the microcontroller of the robot and create an FPGA-like infrastructure. Currently, we are using up to five microcontroller boards, which makes the robot heavy. So we are planning to use 6 STM32 microcontrollers and integrate them into one board.
Third is the method of fixing the omnidirectional mirror. The current fixation method is using a cylindrical acrylic pipe. I don't want to use the acrylic pipes because they are heavy, although they can provide an all-round view. So next time I will install two thin pipes right next to the steering wheel and fix the omni-directional mirror.
We are also planning to develop a GUI for the robot, so that we can change the program pattern and set the threshold of the line sensor on the robot itself. If the detailed settings can be adjusted on the robot itself without using a PC, it will be easier to change the settings during the competition.
The third is to incorporate machine learning into the robot. Whenever a score is scored in the first half of the game, the way the robot moves when the score is scored is saved, and at halftime, the robot derives an advantageous move based on the analysis of the way it moves. We will use the analysis results to our advantage in the second half of the game.

6. References

https://www.jstage.jst.go.jp/article/jacc/53/0/53_0_323/_pdf

https://www.yukisako.xyz/entry/pid_control

https://docs.openmv.io/reference/constrained.html

https://www.pjrc.com/teensy/teensyduino.html

http://www.tokushudenso.co.jp/technicalGuide/pdf/reckoning2heitan.pdf

https://neophile.hatenablog.com/entry/2016/11/26/114316