Radian 2023: Robocup Soccer Lightweight

Jieruei Chang
jierueic@gmail.com

Shrey Khetan
shrey@khetans.com

Yangwenbo Yao
william.us35@gmail.com

June 2023

Abstract

We, Team Radian from Princeton High School of the United States of America, present our fully autonomous soccer robot for the 2023 International Robocup Junior Lightweight competition, which features a solenoid kicker, role switching, and ultrasound-based localization. We discuss the electrical, mechanical, and software aspects of our robots, the features used to increase reliability and give a competitive edge, as well as the development process.

1 Introduction

Hailing from Princeton, we intertwine the disciplines of computer science, mechanical engineering, and electrical design to create autonomous soccer robots. We are Jieruei Chang, Shrey Khetan, and Yangwenbo (William) Yao, specializing in software, electrical, and mechanical design respectively. This is our second year competing in Robocup Junior: last year, we won second place in the US National competition, as well as the Best Engineering Design award. This year, we won the national Robocup Junior competition and will be representing the United States in the international competition in France.

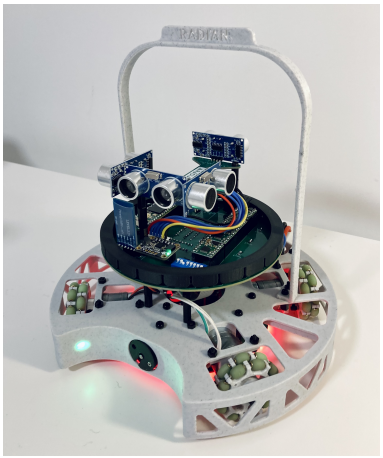

Figure 1: One of our two identical robots for 2023 Soccer Lightweight.

Our robots are built from the ground up with reliability in mind: our software is written to handle broken sensors, our custom four PCBs have a modular design to facilitate swapping, and our 3D-printed parts are deliberately simple to eliminate points of failure. Our four-wheeled robots are equipped with a 48V solenoid kicker, a lightgate, Pololu

motors equipped with custom omnidirectional wheels, 4 ultrasound sensors for localization, 24 infrared sensors for ball detection, a circular line PCB with 24 sensors, and HC-05 Bluetooth modules for dynamic role assignment.

2 Electrical Design

2.1 Overview

This year we designed our own custom PCBs to increase the modularity and reliability of our robot. We used the KiCAD software as it is free and open source, allowing for easier collaboration with teammates. We decided to make 4 custom PCBs this year: Main, Line, Kicker, and Ultrasound.

2.2 Main PCB

Our main PCB (Figure 2) is situated atop our robot and contains a diverse range of components, including 24 TSSP4038 infrared sensors, a BNO055 IMU, a voltage regulator, Teensy 4.1 microcontroller, HC-05 bluetooth module, DRV8874 motor drivers, dip switches, and attachment ports for the light gate (consisting of an LED and a photoresistor), kicker circuit connector, ultrasound board connector, and line sensor board connector. To achieve comprehensive 360-degree coverage, we employed a circular design, evenly distributing the infrared sensors at 15 degree increments around the board. In order to handle the large number of sensors, we use three ADCs (analog-to-digital converters) that can each handle eight analog sensor signals. These values are sent to the teensy using the SPI (Serial Peripheral Interface) communication network. In addition, we decided to add a Schottky diode in the latest version for reverse voltage protection, as well as five DIP switches that can be used to modify the behaviour of the robot without changing the code.

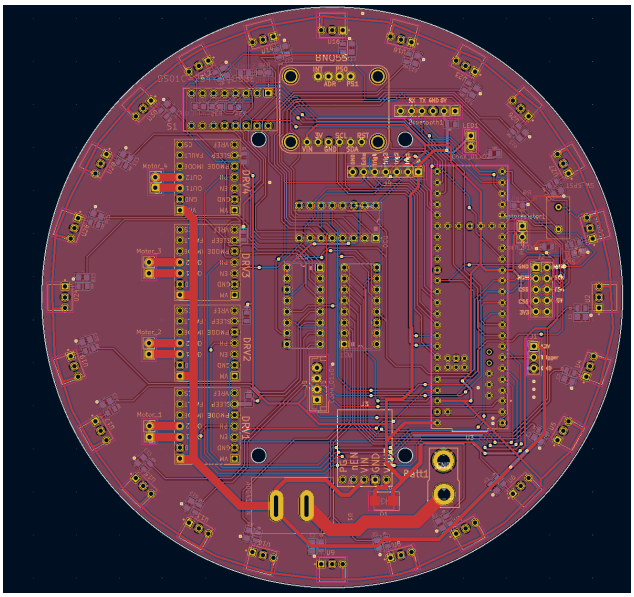

Figure 2: KiCAD Image of Main PCB

This year, we placed particular emphasis on facilitating component interchangeability while keeping the design relatively simple. As a result, we concentrated on consolidating the majority of our components onto the main board itself mounted on header pins, rather than

connected in many discrete parts. This approach streamlines maintenance and enhances convenience when replacing or upgrading elements of the system.

By adhering to these design considerations, we aim to create a more efficient and seamless integration of components on our main PCB, ensuring optimal functionality and adaptability for our robot.

2.3 Line PCB

The line PCB (Figure 3), positioned at the bottom of our robot, contains 24 ambient light sensors and 24 red LEDs. To ensure precise line sensing from all angles, we once again adopted a circular design, equidistantly placing the LEDs and ambient light sensors. The PCB connects to our main PCB through a ribbon cable IDC connector. This allows for easy interchangeability and debugging ability. We chose a circular design over a cross-shape design since the sensors on a circular PCB can hit the line earlier (for example when the robot hits a corner) and a circular PCB functions equally well across all possible robot orientations. Since our ground clearance is rather low, we designed this PCB to be as thin as possible with SMD components, and cut holes into the PCB where our motors and solenoid would be positioned to both reduce weight and ensure enough ground clearance. We once again incorporated ADCs in our line PCB design for ease of communication. Like most of the other PCBs, this PCB went through multiple iterations; our first version ran into some problems with the routing done for the CS (Chip Select) pins, and we had to completely redesign the board to solve these issues.

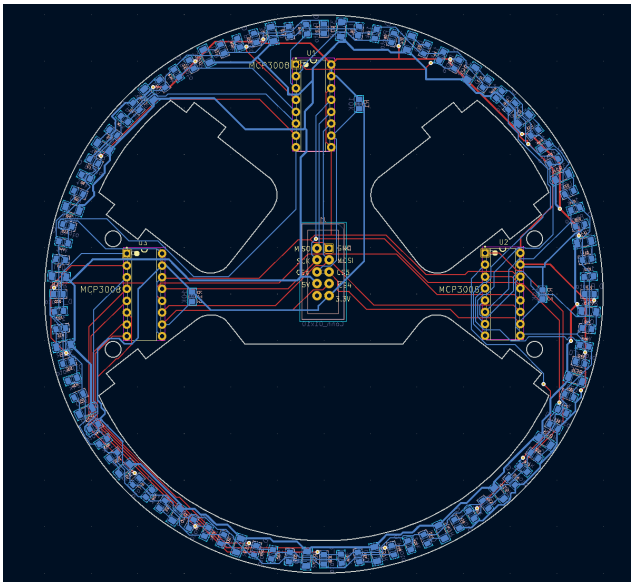

Figure 3: KiCAD Image of Line PCB

2.4 Kicker PCB

The solenoid PCB (Figure 4) is electrically the most complex PCB we designed. For optimal solenoid performance, we had to use a boost converter to increase the voltage fed to the capacitor. To accomplish this we used a boost convertor that takes in 12V and outputs 48v. We use a 2200uF electrolytic capacitor to store and rapidly release current to power a Takaha CB1037 open-frame solenoid.

To control the high voltages safely with the Teensy microcontroller, we added MOSFETs

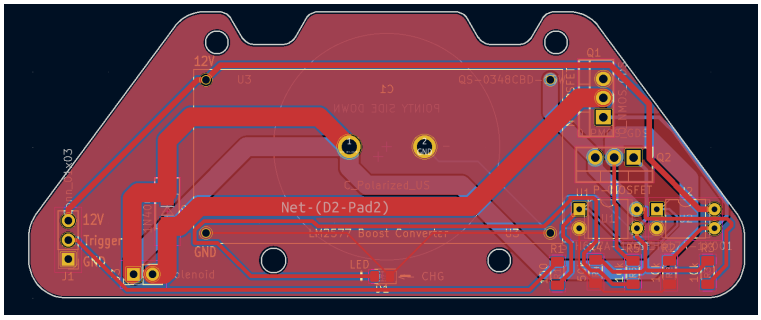

Figure 4: KiCAD Image of Kicker PCB

and optocouplers to the circuit, electrically isolating the low-voltage and high-voltage components, as well as the activated solenoid and the 12v power source. In addition, a flyback diode across the solenoid dampens the voltage spike generated from its activation. Without these safeguards, the sudden power drain during solenoid activation could cause damage to components, or cause the other robot components to momentarily lose power.

2.5 Ultrasound PCB

Our fourth and final PCB we designed for this year's robot was the Ultrasound PCB (Figure 5), containing four ultrasonic sensors set 90 degrees apart to cover all four cardinal directions. We decided to make this a separate PCB to mount the sensors higher, to see over the goalposts and accurately detect the walls as well as above any potentially shorter opponent robots. The PCB is designed to reduce the number of wires coming from all the ultrasonic sensors. Each sensor requires a trigger and echo pin, which if directly wired to the Teensy would take up 8 of its digital pins. Our PCB allows the ultrasonic sensors to share echo pins, reducing the pins being used on the Teensy to only 5. We use diodes to prevent echo signals from reaching any other sensors (in effect creating an electrical OR gate on the echo line), reducing the possibility of electrical damage and interference.

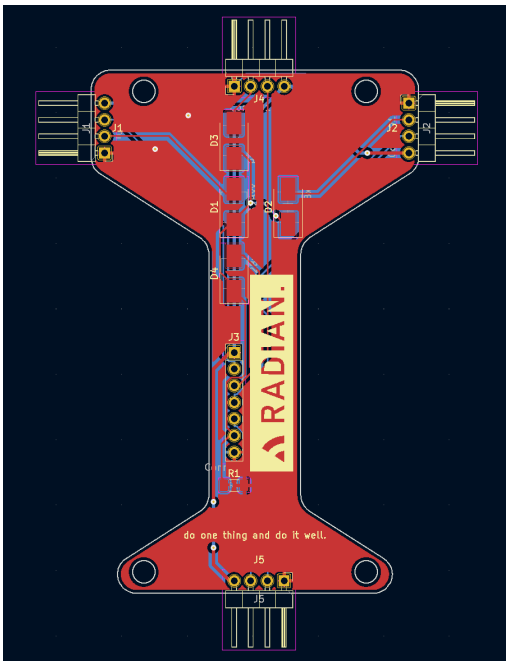

Figure 5: KiCAD Image of Ultrasound PCB

3 Mechanical Design

3.1 Overview

The mechanical aspects of the robot were designed with Onshape, an online CAD platform that allowed

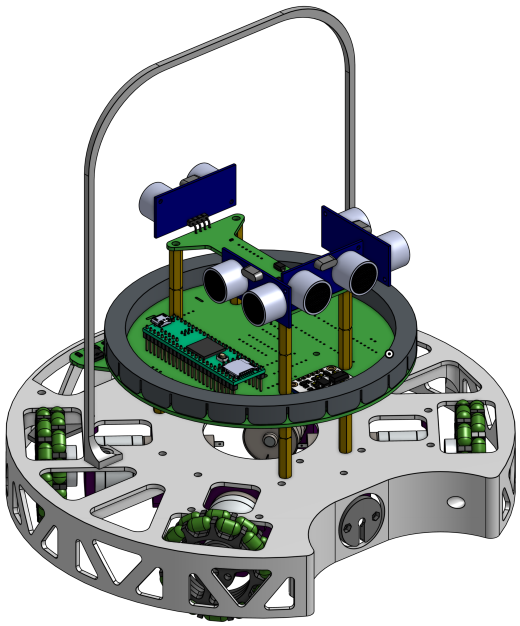

Figure 6: CAD design in Onshape

for easy collaboration. The 3D printed pieces consist of a main chassis and handle, an infrared sensor shield that narrows the field of view of the sensors to prevent interference and make them more accurate, a solenoid plate, a battery holder, and motor mounts.

3.2 Materials

Different mechanical parts of the robot are subjected to different amounts and forms of stress and have different material requirements. We utilize PETG and carbon fiber-infused PLA as our 3D printing materials. We initially planned to use carbon fiber PLA as the primary structural material, but quickly found that while it had high rigidity, that was not necessarily a desirable attribute because it is very brittle. Instead, polyethylene terephthalate glycol (PETG) is used for the most of the high-stress pieces; while it actually has a lower tensile strength than PLA, it is flexible and deformable. This means that our PETG chassis can withstand minor crashes without shattering (which occurred on last year's iteration of the robot). Carbon fiber is used in places where rigidity is more important, such as the thin battery holder plate.

3.3 Design

Unlike most chassis we have seen, our chassis is a single piece of PETG. While this reduces its modularity, it makes the chassis more mechanically resilient. There are no screws that could come loose, and the outer crash ring supports the flat mounting plate more evenly. In addition, this design makes the chassis very easy to fabricate, so spares can be easily

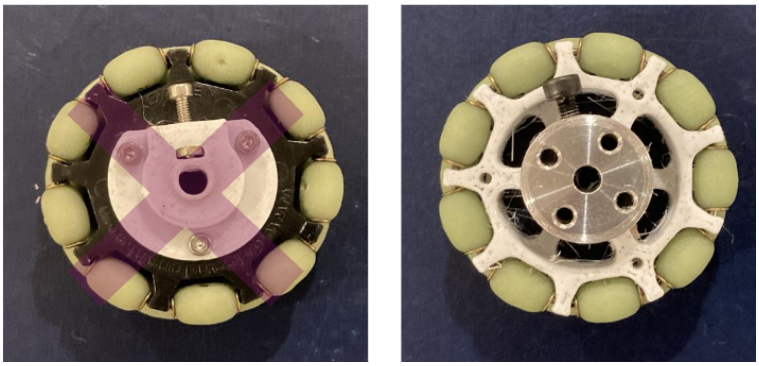

Figure 7: Left: old omniwheel design; Right: new omniwheel design

constructed. While the lack of a bottom plate may at first seem to significantly reduce the robot's structual integrity, its internal support ribs are adequate to maintain rigidity. The lack of a bottom plate also has the advantage of increasing our ground clearance.

This year, we also designed custom omniwheels; we initially used off-the-shelf omniwheels with custom 3D printed adapters (Figure 7) but these wore out rapidly. Instead, we designed custom PETG wheels that attach to metal adapters that can securely clamp onto the motors' D-shaft.

3.4 Fabrication

All 3D-printed pieces were fabricated on a Prusa i3 MK3S with a hardened nozzle for the printing of abrasive materials and a satin powder-coated steel spring sheet to enable easy removal of PETG prints (which is difficult to do on traditional print beds due to its strong layer adhesion).

4 Software

4.1 Movement

The robots use four driven omniwheels for unconstrained lateral translation. Trigonometric equations are used to determine how quickly each of the four motors must rotate in order to move in any given direction. Rotational force (typically used to point the robot forward) can be applied simply by adding an additional constant speed to each motor. We further run a normalization procedure to ensure that at least two motors are always spinning at maximum power, to increase the robot's speed.

An interesting (but very frustrating) problem we encountered was in setting the PWM signal, which is the method by which the Teensy microcontroller sends commands to the motor drivers. A PWM signal is intended to be in the range from 0 to 255 inclusive, where 255 is the maximum power. However, values over that range are not simply clamped to 255; they roll over such that a signal of 256 is equivalent to 0 (no power). This caused some motors to

abruptly shut off while the robot was moving, since floating-point errors in the trigonometric calculations caused the robot to occasionally send values that were over 255.

4.2 Sensing

4.2.1 Line

The primary line algorithm, used by the offense robot to avoid leaving the play area, operates on the principle of redirecting the robot's motion vector to clamp it against the line; that is, if the robot's proposed motion would move it further into the line, it is redirected such that moves along the line instead. The line algorithm first applies thresholding to each of the line sensor readings, then runs a routine to detect regions of activated sensors. Then, we can use these detected regions to find an angle that is pointing towards the line: if only one region is detected, the median direction of that region is used as the line direction; if two regions are detected, the line direction is the angle bisector of the median directions of the activated regions (Figure 8).

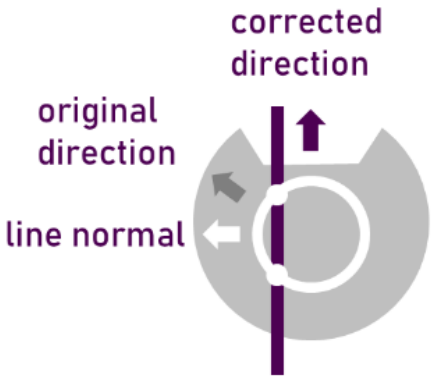

Figure 8: Basic Line Redirection Algorithm

If the proposed motion angle is less than 90 degrees from the line angle, the angle is clamped to the nearest of the two angles perpendicular to the line angle (Figure 8). This behaviour enables a smooth following of the line, rather than an oscillating motion where the robot repeatedly collides with the line and then backs off. However, testing showed that this algorithm is not always sufficient to stop the robot (especially for a horizontal line when the robot is moving forward at fill speed), and so "emergency back off" is sometimes necessary. Additionally, if three regions are detected, then the robot is likely in a corner, and backs off immediately.

As a further safeguard, the robot keeps track of the initial line vector calculated when it first enters a line. If the detected line vector suddenly flips by 180 degrees, then the robot is likely over halfway across the line, which can occur if it is pushed or if it is moving at high speed. In this case, the robot maneuvers to remove itself from the line before continuing.

However, rather than avoiding the line, the goalie robot is designed to "straddle" the goalie-box line, to keep it in front of the goal. By constantly moving in the direction of the line, the robot maintains a position in the middle of the line. Additionally, the line angle is used to detect the vertical lines from the edges of the goalie box, in order to prevent it from moving beyond the goal area.

4.2.2 Infrared

Detection of the ball is conducted through analyzing the outputs of the infrared sensors positioned equidistantly around the main PCB. The data from each sensor can be interpreted as a vector whose direction is that of the sensor's orientation, and whose magnitude is the strength of the infrared signal received by that sensor. However, our testing showed that when placed sufficiently close to the wall, the wall can reflect IR light, affecting the detected location of the ball by up to 10 degrees, though it is possible that this is a problem unique to our testing walls. To remedy this, we apply a weighting to each sensor k to favor the maximum-valued sensor:

$$p_{k_{\text{weighted}}} = p_{k_{\text{raw}}} \cdot \frac{n - 2(n - \text{abs}(\text{abs}(m - k) \bmod n - \frac{n}{2})}{n}$$

where n is the number of infrared sensors (24 for our robot) and m is the index of the maximum-valued sensor.

Since a reflection creates a secondary region of less-activated sensors that is some distance away from the primary region of more-active sensors pointing to the ball, such a weighting diminishes the importance of the secondary region in the ball angle calculation and thus limits the impact of the walls' interference.

The ball angle θ can then be calculated with

$$\theta = \tan^{-1}\left(\frac{\sum_{k=1}^{n} p_k \sin\left(k\frac{2\pi}{n}\right)}{\sum_{k=1}^{n} p_k \cos\left(k\frac{2\pi}{n}\right)}\right)$$

where p_k is the weighted sensor reading of the kth sensor. Because the TSSP4038 sensors are prone to randomly failing, in each cycle we selectively ignore sensor readings that appear anomalous, i.e. they are drastically different from either of their neighboring sensors. The sensors that are ignored are replaced with "virtual" sensor readings that are simulated with the averages of neighboring functional sensors.

4.2.3 Inertial Measurement Unit

To maintain a forward heading, we use a PID controller that applies an angular correction, following the formula

$$u(\theta) = K_p \cdot \theta + K_i \cdot \int_0^t \theta \cdot dt + K_d \cdot \frac{d\theta}{dt}$$

where θ is the error angle and K_p, K_i and K_d are the experimentally determined proportional, integral and derivative gains respectively. The proportional term applies a corrective force

that is matched by the resistive force of the derivative force (to prevent overshoot). In practice, the derivative term is unused as it tends to create instability within the robot since the derivative value is subject to a substantial amount of noise.

4.3 Strategy

4.3.1 Role Switching

The robots are designed to dynamically switch between orbiting (attacker) and defender roles by communicating with each other over Bluetooth with two HC-05 modules. This enables more complicated strategies, as the goalie automatically switches to attack mode after taking possession of the ball while the other robot switches to defend the goal.

We ran into the Two Generals problem while testing this role-switching functionality: the Bluetooth link is not always reliable, and a single "switch" message may be missed. To guard against both robots turning into goalies or orbiters, the robots instead constantly broadcast their current roles to each other, and both assign their roles as the opposite of the other. To avoid problems similar to race conditions where a robot switches its mode, receives a message from its twin, and because of that switches its mode back, a cooldown timer prevents the robots from switching roles too quickly.

Inter-robot communication also increases the reliability of the full system. If one robot is incapacitated, the other robot can recognize this and change its behaviour accordingly. A dead man's switch or watchdog timer is used to determine if a robot is broken: the robots are constantly broadcasting their current roles to each other; if a robot fails to send messages for a certain amount of time (about five seconds) the robot is deemed incapacitated. While we experimented with a hybrid attack-goalie mode for single-robot scenarios, we found that generally it is more reliable to simply convert the remaining robot to a goalie or attacker, depending on the opponent.

4.3.2 Orbit

The orbit algorithm is designed to maneuver the robot behind the ball and charge at it from behind. It takes into account the ball's angle relative to the robot, as well as an estimated proximity to the ball, in order to calculate an efficient route:

$$\theta_{\text{orbit}} = \theta_{\text{ball}} + \min\left\{m_1 e^{p_1 \theta_{\text{ball}}}, 90\right\} \cdot \min\left\{m_2 e^{p_2 d}, 1\right\}$$

where θ_{ball} is the ball angle, m_1 and p_1 are constants that determine the orbit strength (larger values mean the robot circles more), m_2 and p_2 are constants that determine the dampening strength, and d is the estimated proximity.

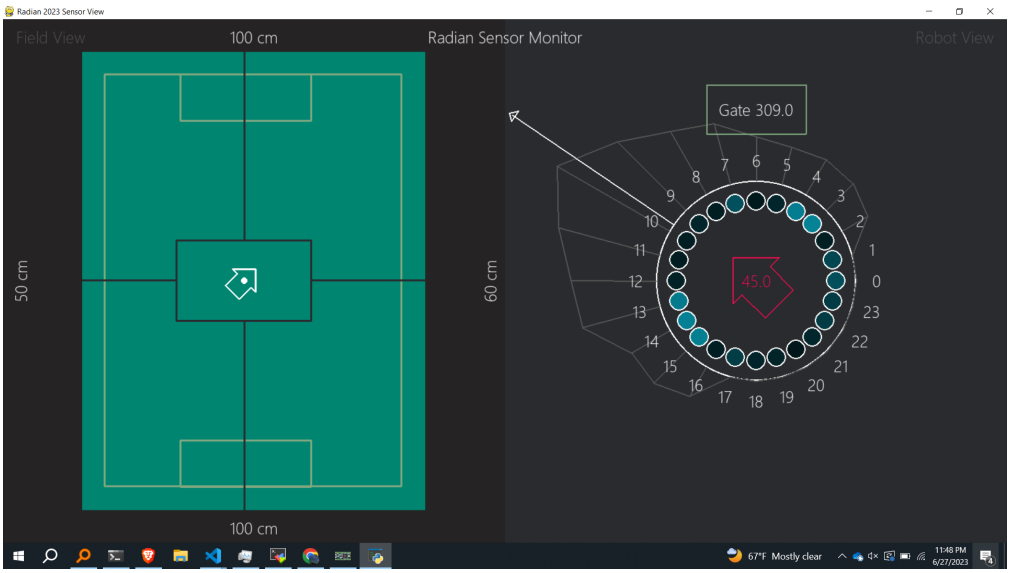

Figure 9: Sensor Visualizer

4.3.3 Sidestep and Goal Rotation

The ball will not always be in front of the goal; therefore, it is necessary that the robot can maneuver the ball towards the goal even when it is placed at the boundaries of the field. The typical solution to this is an omnidirectional camera tower that constantly determines the angle to the goal, but cameras suffer from visual interference and a tower/mirror assembly can weigh over a hundred grams. Instead, we opted for an ultrasound sensor assembly.

When the ball is in front and sufficiently close to the robot, the sidestep/goal rotation program activates. Since the ultrasound sensors can only function when the robot is facing directly forward, the robot first conducts a "sidestep" maneuver where it moves horizontally while maintaining its heading and calculates the angle to the goal with ultrasounds until the ball, goal, and robot are collinear. Then, it shuts off the ultrasounds, turns towards the goal, charges at the ball, and fires its kicker upon confirmation from the light gate that the ball has been captured.

4.3.4 Defense

The defense algorithm utilizes a PID controller to quickly and efficiently close the distance between the ball and itself while also minimizing overshoot. To maintain a proper position, the line straddling algorithm maintains position in front of the goal; ultrasound sensors complement this by ensuring that the robot does not accidentally run backwards into the goal or run to the sides of the field. A line corner detection algorithm ensures that the robot stays within the goalie-box area.

4.4 Debugging Tools

We designed a custom graphical sensor visualizer (Figure 9) to easily debug the 54 sensors on the robot, allowing us to detect broken sensors or determine good sensor thresholds. For example, the data shown in the image reveals issues with the line sensors incorrectly activating. The program is written in Python and communicates with the robot over a serial link.

5 Reliability

Our robot was designed from the ground up to be very modular, so the different components are swappable in the event of a failure of any individual module. For example, the solenoid voltage boost circuit is a separate PCB, which not only reduces the electromagnetic interference but also enables it to be more easily swapped out if necessary. Wide usage of snap connectors enable the robot to be more easily taken apart, and components are mounted with header pins wherever possible to facilitate ease of replacement. Even the ADCs are mounted on DIP sockets to allow for replacement if necessary.

The software is built on the principle of *graceful degradation*. It is capable of operating with over half of its infrared and line sensors malfunctioning; on-the-fly sensor failure detection enables it to reduce the impact of malfunctioning sensors.

6 Limitations and Further Directions

The main limiting factor we encountered is the weight limit of the robot. With only 1100 grams to work with, we were unable to implement many features, such as a camera tower for goal localization. The modular nature of the robot, while improving its reliability, come at a not insignificant cost in weight as the components are less well integrated together; we had to swap all the steel nuts and bolts for nylon ones in order to save weight. Our motors are also relatively weak, and so the robot's capabilities are limited by its top speed.

7 Acknowledgements

This project was only possible with the help of our generous sponsors. We thank Pololu for offering discounts on their 25D 12V high-power motors, which enable high torque at a low price point. We thank JLCPCB for sponsoring our PCBs, and Digikey and Mouser for offering discounts on electronic components. We also thank OpenMV for providing two free cameras, though they were not integrated into the final robot. We thank the former members of Team Orion for providing expertise to help resolve our electrical design issues. Finally, we thank our parents for supporting us and for putting up with our late-night work sessions.