PRIDE 2024: RoboCup Soccer Lightweight

Kai Sugrue (kaisugrue2@gmail.com)
Anand Parab (bjxobjxo@gmail.com)
Stephen Wan (stephenmwan@gmail.com)
Brandon Liu (brandon.zekun.liu@gmail.com)
Mentor: Mrs. Crandall (mariannacrandall@hvrsd.org)

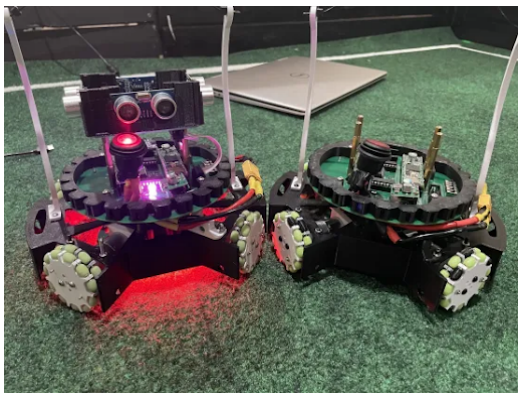
Abstract: This paper details the mechanical, electrical, and software components of an autonomous soccer robot created by team PRIDE for an international soccer robotics competition. The robots are designed to play soccer strategically by detecting the ball, remaining within the field boundaries, and accurately moving the ball into the goal.

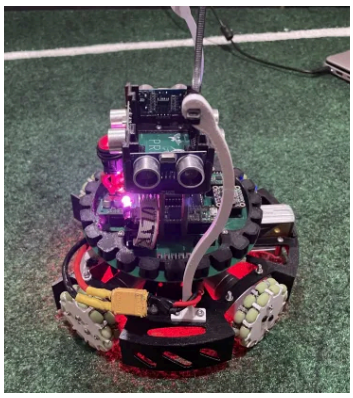
1 Introduction

Team PRIDE consists of five high school students competing in the Lightweight division of Robocup Junior Soccer as a STEM senior capstone project at Hopewell Valley Central High School, NJ. We are Kai Sugrue(Captain, Electrical), Anand Parab(Mechanical), Stephen Wan(Programming), and Brandon Liu(Programming).

We started this project to learn more advanced robotics that our school's engineering curriculum didn't cover. We wanted to gain real-world experience applying code/electronic/design theory to something practical.

Our team was fully student-led, forcing us to self-study everything. With a combination of past robotics experience through our FRC(FIRST Robotics Competition) team, watching YouTube videos, and surfing through papers/posters of past teams, we were able to learn enough to create this robot. Additionally, we're members of PSR(Princeton Soccer Robotics) where we receive guidance from past U.S. RoboCup teams.





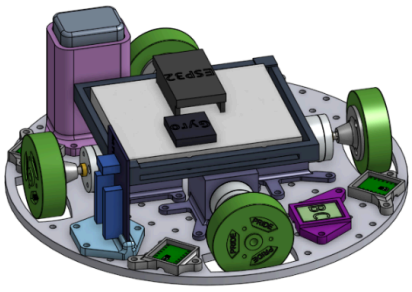
Parts List/Specifications:

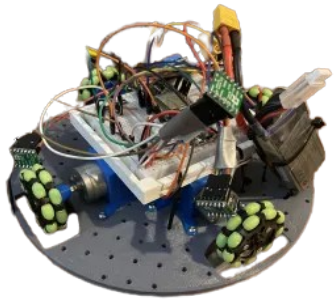
Name	Part	Quantity	Cost	Total Cost
PCB - Line/Ultrasonic/Line	JLCPCB	1	$50.00	$50.00
Lipo Battery	11.1V 1300mah	1	$16.99	$16.99
12V -> 5V Voltage Regulator	D30V30F5	1	$6.48	$6.48
Power Switch	GRB112B802BR1	1	$4.25	$4.25
Motors 12V 20.4:1	Pololu #: 3203	4	$14.48	$57.90
Motor Drivers	DRV8874	4	$4.58	$18.32
Microprocessor	Teensy 4.1	1	$29.95	$29.95
Gyroscope	BNO085	1	$24.95	$24.95
Line sensors	SFH-5701	24	$0.88	$21.07
IR sensors	TSSP4038	24	$0.60	$14.35
I/O expander SPI	MCP3008	6	$3.12	$18.72
Bluetooth Module	HC-05	1	$8.00	$8.00
Daisen Omniwheels	DOW-46P	4	14.94	$59.76
Wheel Hubs	Pololu #: 1997	2	$8.49	$16.98
PETG Filament	1kg spool	1	$15.00	$15.00
Hardware	Standoffs, M3 screws, etc.		$30.00	$30.00
Extra electronic components	JSTs, IDCs, wires, etc.		$20.00	$20.00
TOTAL				**$412.71**

Final Weight: 1.2kg

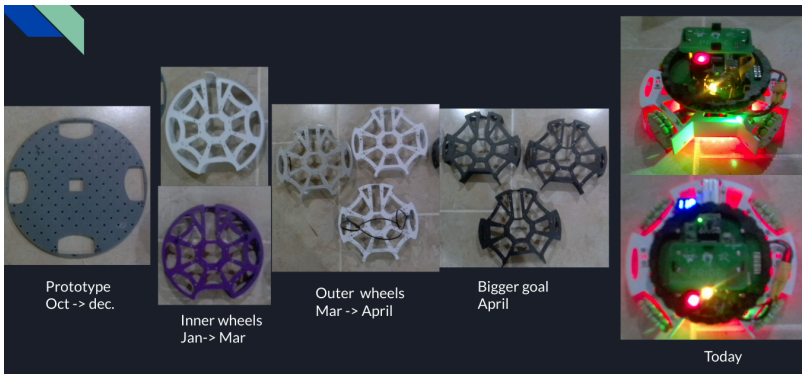
2 Mechanical

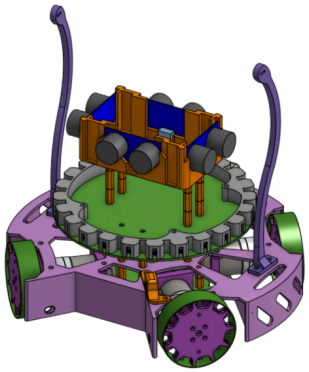
2.1 Robot iterations





- **RC driving - Prototype #1: Photos/videos, Building/Design Process**





Subsequent Prototypes: Building/Design Process **Final iteration**

Over 10 months, we've engaged in a continuous process of refining our robot's design. Since this was our first year, we strategically divided our design into small measurable goals. Initially, our focus centered on developing a rudimentary prototype capable of basic driving. Subsequent iterations progressively integrated IR, line, and ultrasonic sensors into our ever-evolving design.

2.2 Ultrasonic Holder

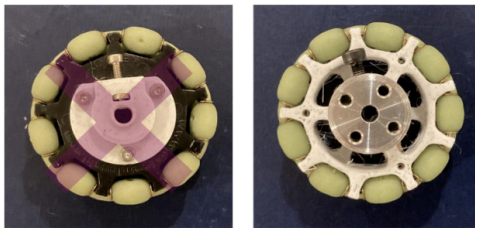
During testing, we encountered an issue where the ultrasonic sensors exhibited erratic movement, jerking back and forth in response to sudden changes in the robot's acceleration. Upon investigation, we identified the root cause as a subpar mechanical connection stemming from the electrical connector (female header). To address this issue, we designed and fabricated a 3D-printed housing specifically tailored to securely hold the ultrasonic sensors in place, thus stabilizing their movement and enhancing ultrasonic readings.

2.3 Structural Stability/Modularity

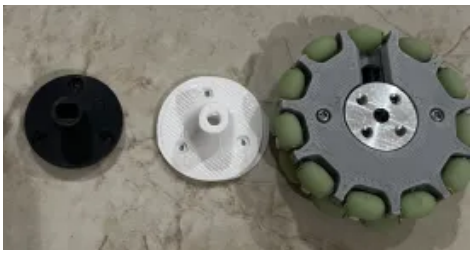
Our design prioritized both structural integrity and modularity. By crafting a PETG chassis, we ensured flexibility against robot collisions; PLA is stiffer but lacks the flexibility to take hits. Standoffs were strategically placed to seamlessly connect all components, including PCBs, fostering a modular assembly approach. Alongside this, we focused on easily accessible screws to ensure easy tightening. Moreover, our system was designed to accommodate the effortless attachment or detachment of modules like ultrasonics and line PCBs, allowing for quick adjustments.

2.4 Omni wheels

We opted for COTS Omniwheels to expedite our process, relying on their proven reliability. However, we encountered a problem: our motors differed from the standard, necessitating a custom adapter. Fortunately, we drew insight from Team Radian/Elite, who had previously tackled a similar challenge. During our journey, we learned that PLA-printed hubs were prone to excessive wear, prompting a shift to Pololu aluminum hubs featuring a convenient set screw mechanism for secure attachment to our motors.



Team Radian’s design(old vs new)



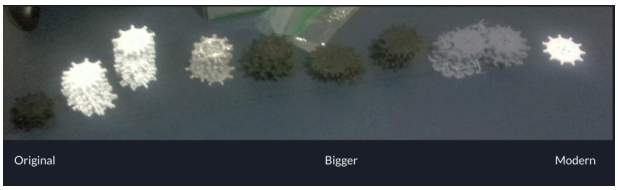
Our Design(old vs new)

Following extensive testing, we determined that our line sensors required greater elevation from the ground to function effectively. Unable to adjust the sensors' height, we opted to increase the size of the wheels to raise the robot accordingly. This adjustment necessitated two components: larger O-rings and beads. However, procuring these parts from the original manufacturer proved both costly and challenging due to customization requirements. To overcome this obstacle, we innovated by crafting our own custom parts. Using a jumbo paperclip, we formed a coil around a

3D-printed cylinder, shaping it into a circle with pliers. Additionally, for the custom beads, we utilized TPU to replicate the rubbery and flexible texture of the COTS beads.

Custom TPU wheels

Paperclip O-Ring



Wheel Iterations

2.5 Handle

We aimed for our handle to strike a balance between rigidity, ensuring it wouldn't obstruct sensors, and flexibility, preventing excessive pressure on the robot. Our approach involved a combination of a 3D-printed handle and a zip-tied top. This setup effectively distributed the load evenly over a longer duration, minimizing the force exerted on the handle.

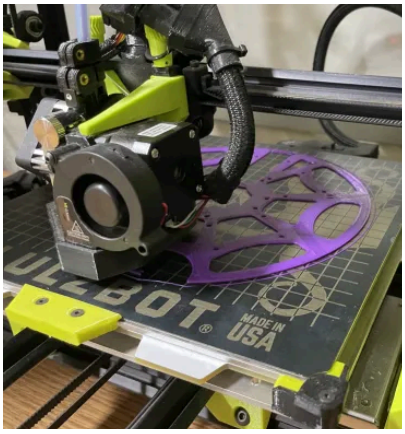
2.6 Motors choice

When deliberating on motor options, we found ourselves torn between Maxon motors and Pololu's 25D motors. While Maxon motors offered greater strength (being brushless) and extensive customization capabilities, the combined cost of a Maxon motor and its corresponding motor driver was approximately 7.5 times higher than that of a Pololu setup ($150 vs $20). In consideration of cost-effectiveness, we ultimately chose the Pololu option to keep expenses in check.

2.7 Fabricating hardware/software

For our CAD needs, we chose OnShape for its user-friendly interface and collaborative features, making it easy to learn and share designs among team members. For 3D prints, we printed Chassis parts at our Bambu Lab, and PLA parts on the Lulzbot printer.

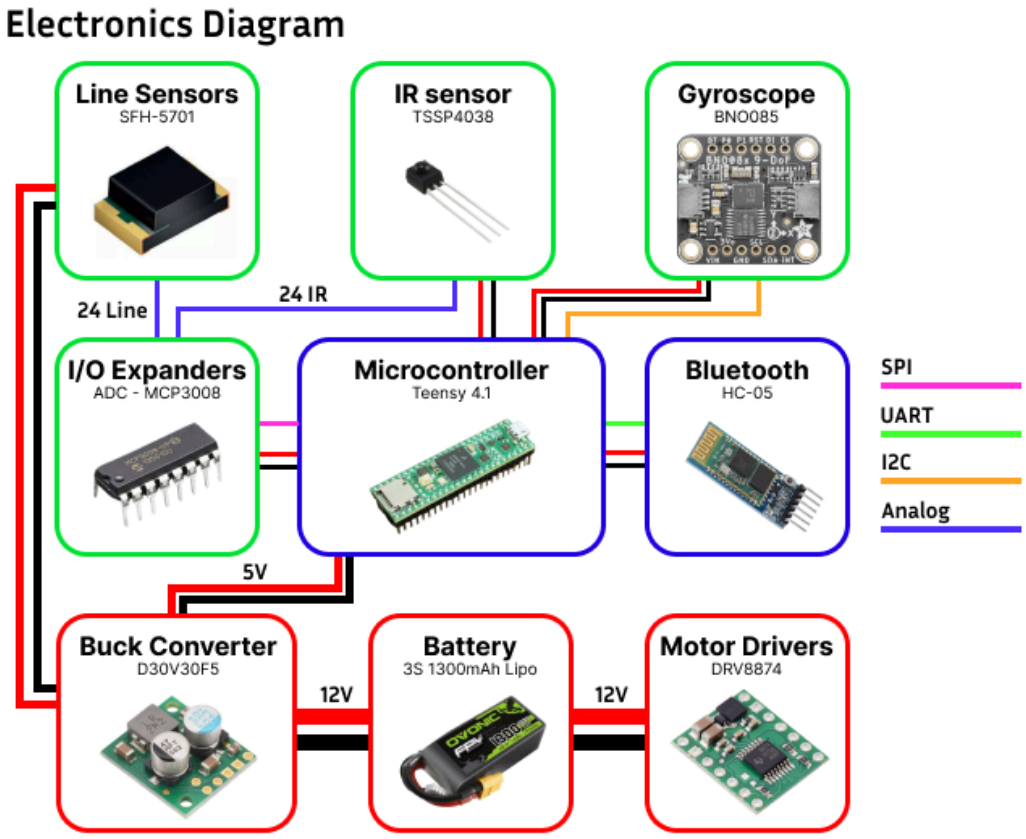
Bambu Lab X1-Carbon



LulzBot SideKick 747

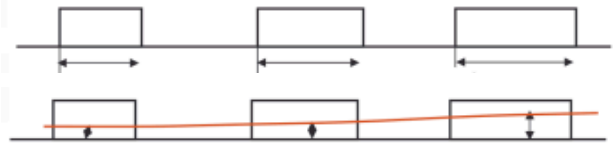
3 Electrical

Since this was our first time learning Electronics, most electronics are from the standard parts everyone uses in PSR—namely, the Teensy 4.1, TSSP4038, BNO085, HC-05, SFH-5701, and MCP3008. Our only real difference in electronics was going for an *aesthetically pleasing* electronics setup utilizing multiple PCBs with IDC and JST connections between PCBs and parts.



3.1 IR sensors

Rather than staying on the entire time, the ball sends out pulses of IR light. This was consistent with our testing as we got *virtually* floating point values despite the ball being directly in front of the sensors. To counter this, we used a low pass filter to smoothen out rapid fluctuations of the ball pulses. The ADC allowed each sensor to be read as magnitudes rather than a binary(0,1).



Black - Ball pulses

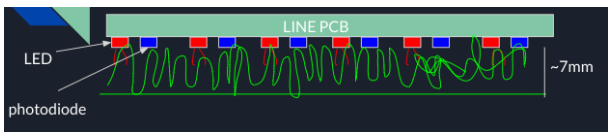
Red - Low pass filter

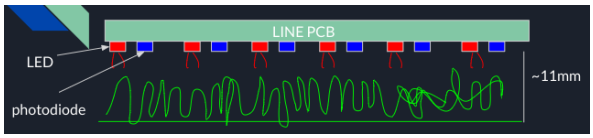
3.2 Line sensors

The line sensors we used were photodiodes, specifically the SFH-5701. Pairing this up with a red LED directly next to it creates a setup where the reflected light of the red LED bounces off

the carpet into the photodiode. Since red light is a conjugate color to the green carpet, the sensor values change drastically going from the green carpet to the white line.

One frustrating problem with this sensor(which we discovered the hard way) is that the top of these photodiodes needs at least 1.1cm of height above the ground, otherwise, the bristles on the carpet will come into contact with the photodiodes and give unexpectedly high values. To fix this, we made our wheels bigger.





3.3 Ultrasonics

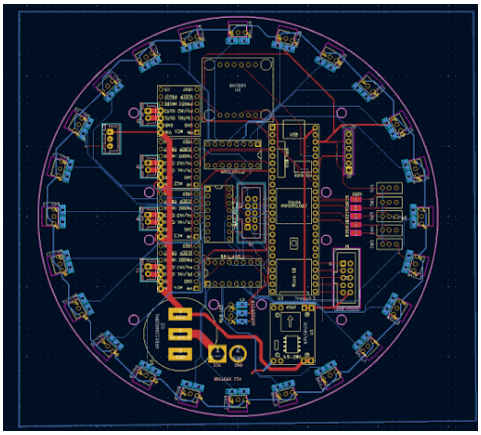
When originally conceptualizing this robot, we wanted to stay simple since this was our first year. Instead of opting for a camera that would be around a $200+ investment and weigh around 100 grams, we opted to have ultrasonic sensors which could assist for goal localization and help protect our defense and offense robots.

3.4 I/O expanders

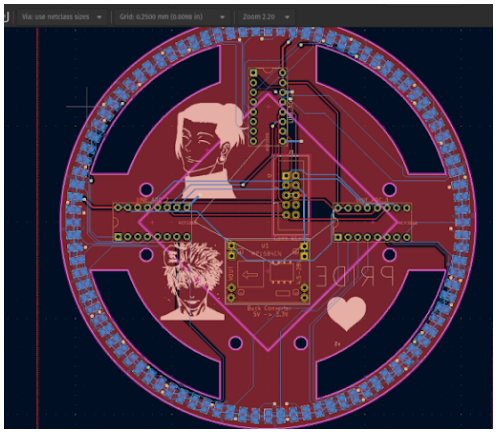
Since the Teensy 4.1 realistically has only 42 pins available, placing all 4 ultrasonics, 24 lines, and 24 IR sensors is impossible. Therefore, we expanded our GPIO pins using 6 MCP3008s, each capable of handling 8 sensors converted via the ADC. These MCP3008s communicate over the SPI bus, requiring only 9 pins since MOSI, MISO, and CLK can be shared. With so many spare pins, we allocated each ultrasonic a Trigger and Echo pin, totaling 8 pins. Fortunately, transitioning to SPI Bus communication did not affect the speed at which we gathered sensor data.

3.5 PCBs:

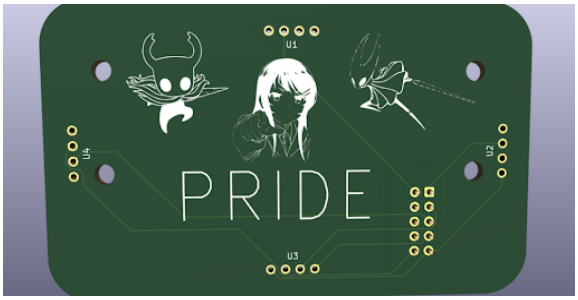
To incorporate so many sensors in precise locations and create an *aesthetically pleasing* robot, we made custom PCBs. The schematics and layout of our PCBs were created using KiCAD. Additionally, we incorporated as many SMD components(resistors, capacitors, LEDs, and photodiodes) to make our robot as compact as possible.



Main PCB



Line PCB



Ultrasonic PCB

4 Programming

4.1 IR detection

Our robot uses 24 IR sensors which allow IR vision in 15-degree increments. Additionally, we use a low-pass filter to convert the IR sensor readings from digital to analog. With analog, we can determine a lot more; as the balls get closer, the magnitude of readings increases; as the balls move angularly closer to an individual sensor, their magnitudes increase. [Video]

To find the ball vector, we take the vector sum of every sensor to find the general direction of the ball. Since the sensors are not perfect, taking the vector sum doesn't yield a perfect angle; rather, we needed to employ some type of “weighting” system for each sensor. We tried both Team Orion’s weighting system(5 highest sensors) and Team Radian’s weighting system(degrading multiplier) and had better results with the latter.

4.2 Line detection/Calibration method

Every line sensor is unique and thus has different tolerances. Additionally, each field is slightly different. EX: The lighting, what kind of white are the lines, etc. Therefore simply assigning predetermined thresholds would not be effective. Thus we created a two-part calibration method to determine the optimal thresholds for each line sensor on any field.
For initial calibration, we place the robot on the green part of the field to record maximum and minimum readings for each sensor, then calculate "green" and "in air" threshold values. To set the white line threshold, we place the robot over a white line, record maximum sensor values, and calculate the threshold as the average of the maximum green and white values.

4.3 Line bouncing/Line treading

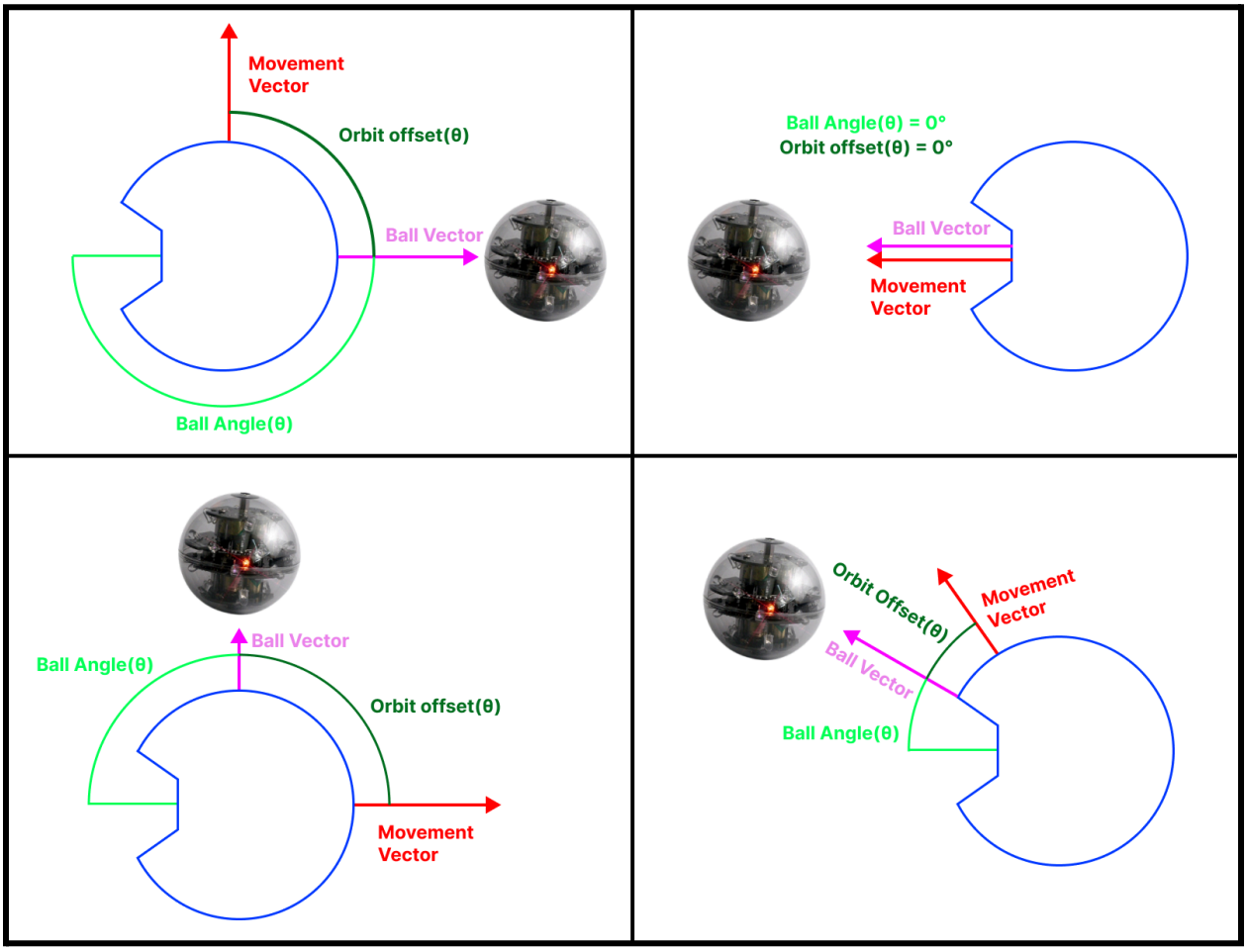
To navigate along the white line, our robot looks at which sensors are sensing the line. It then finds which sensors are most opposite to each other to find an accurate line vector. For example, if one sensor is to the right and one is to the left, it’ll calculate an angle facing perfectly in the middle of these two sensors. This angle is normal or perpendicular to the white line.

The robot then moves in that perpendicular direction whilst following the ball vector. If any white lines come about we simply bounce back and forth, never going over the line. We called this *line bouncing*.

Additionally, we experimented with *line treading*, where instead of bouncing off the line, the robot moves with the line. We take the original angle perpendicular to the line, and add or subtract 90 degrees to the normal angle. This gives us two possible angles to travel at. The robot picks the direction that's closer to where it wants to go and uses that to stay on the line.

4.4 Orbit

Since our robot doesn't have a dribbler, and thus can't rotate the ball with itself, the robot must get behind the ball to score it. To solve this problem, we implemented Orbit which calculates the optimal path to get behind the ball utilizing the ball angle. This uses an exponential function that creates an offset based on the ball angle. The smaller the angle, the smaller the offset; the bigger the angle, the bigger the offset. This offset is added to the current ball angle vector to create a movement vector.



One problem with this algorithm is that the robot follows the same arc path regardless of the ball's distance. To solve this, we use the ball vector's magnitude to adjust the arc path size. This ensures the robot navigates efficiently around the ball, saving precious scoring time.

4.5 Defined Roles - Offense and Defense

In recent years, the meta has oscillated between double offense and having distinct offense and defense robots. While the principle "the best defense is a good offense" might apply generally, in RoboCup, a double offense strategy can cause issues like constant collisions and disrupted orbit algorithms. Combining a defense and an offense robot offers the best of both worlds: a dedicated defense that effectively blocks engagements and an offense robot optimized for scoring.

Our offense robot's sole objective is to score goals, utilizing algorithms for orbit and sidestep maneuvers.

Our defense robot's sole objective is to traverse horizontally along the front line of our penalty box, positioning itself in front of the ball to prevent any shots from entering the goal.

4.6 Defense Timeout / Communication

In the new rule 1.5.5, any robot must approach and touch the ball when placed on the nearest neutral spot before a lack of progress is called; additionally, on its own side, the robot must move the ball to the opponent’s side. In other words, a permanent defense robot won't suffice as it’ll be penalized for not attacking the ball when it's on our side of the field. However, this rule ONLY applies if a lack of progress is imminent, which only happens in one situation: our offense robot is incapacitated.

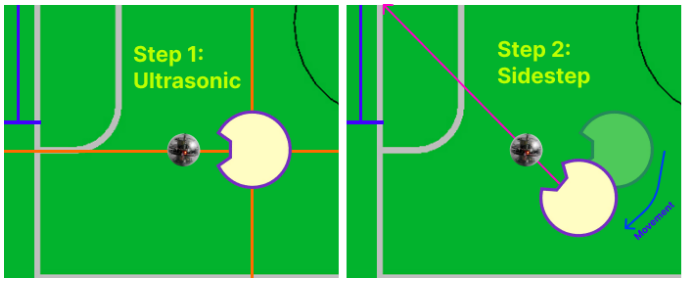
To detect if our offense robot is incapacitated, we use a dead man's switch with Bluetooth communication. For our Bluetooth algorithm, both robots constantly broadcast their roles to each other every second. If a robot stops receiving messages for 5 seconds, we determine the other robot is incapacitated.

If the offense robot is incapacitated, a timeout algorithm runs on the defense robot. Using the ball vector’s magnitude, we check if the ball is on our side. Using the ball angle, we check if the ball is moving. After 3 seconds of inactivity, the defense robot switches to offense to move the ball to the other side.

4.7 Goal Localization / Sidestep

With only an orbit maneuver, our robot simply pushes forward, which isn't effective when the ball is on the left or right. In a vacuum, this might not be too troublesome, but since the ball is placed on the left and right neutral spots when a lack of progress is called, likely, we wouldn’t likely be able to score from those positions.

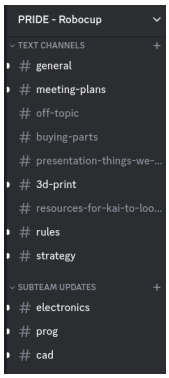
To address this, we added a sidestep maneuver. This maneuver uses our left, right, and front ultrasonic sensors to calculate the angle toward the goal. The robot then rotates and moves to align itself, the ball, and the goal in a straight line. Once aligned, the robot drives forward, successfully executing an angled shot. This algorithm executes when the ball is near the robot and has multiple fail safes to ensure sidestep doesn’t occur randomly

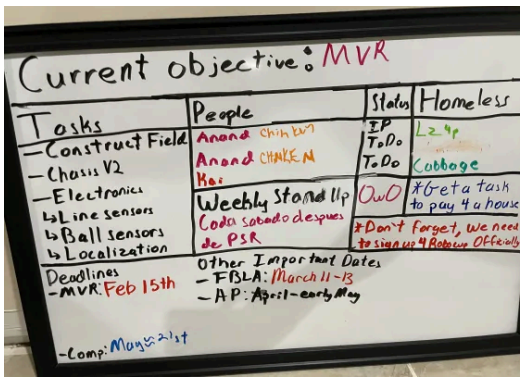


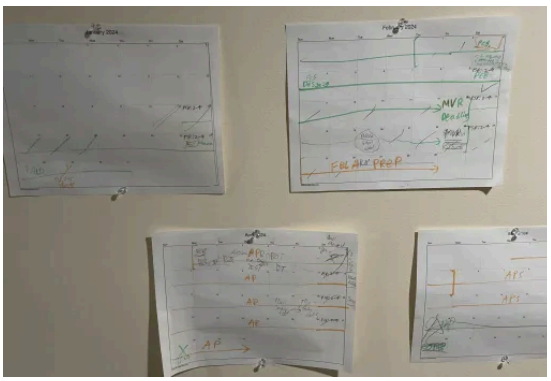
5 Coordination/Project Management

By itself, this project requires an intense amount of coordination by all teams(Mechanical, electronics, software). By frequently communicating with each other via in-person and on Discord regarding our robot's current state, we were always on the same page. Because of our innovation together, we integrated our robot subsystems so well.

Additionally, we had MANY planning documents to track our progress and set deadlines. With year-long calendars, we appropriately allocated our time. Regarding project management, we used TODO lists to demonstrate the current set of tasks being worked on by each subteam to complete an overarching objective. We also kept a *master* notebook, which we updated every week with a new list of subteam tasks and logged the progress made.







To see our step-by-step documentation along the way, we've made visual progress reports that are full of details, updates, and planning we made monthly.
https://drive.google.com/drive/folders/16GlOqHJ0BmVQAS2962yDETPzjIO8-D74?usp=sharing

References

- USA Team Orion
- USA Team Radian
- USA Team Elite
- JPN Team OI_DENGIKEN

Link to reference technical papers/posters -
https://drive.google.com/drive/folders/1mZHFgV5lfAHl1d91mmplxHj-TGebvryU?usp=sharing