I - Abstract

Over the past decades, robotics has become an increasingly popular method to engage students in the fields of Science, Technology, Engineering, and Math(STEM). However, due to the large learning barriers and massive costs, many students and institutions turn away from this opportunity. This project introduces a robot that is used to compete in the RoboCup Junior Competition and provides an opportunity for interested students to engage in robotics. Using this robot as a starter kit, students will be able to learn STEM concepts through the creation, testing, and implementation of their own Soccer Robot.

II - Introduction

In today's job market, the need for specialized workers in STEM have increased steadily. is estimated that by 2025 there will be over 3.5 million job vacancies in the STEM fields (Lazio and Ford Jr, 2019, para. 7). In hopes of increasing the number of graduates in the STEM disciplines, governments have implemented different programs to increase interest from a young age (Lazio and Ford, 2019). One of the largest and most effective forms of STEM education can be found with robotics, an activity that introduces STEM concepts to students through hands-on learning opportunities. (Pierce, 2019).

Throughout the world there are already many initiatives and organizations that provide these opportunities to young students, however, institutions that are interested in these programs often need to invest large sums of money to be able to afford educational kits and other learning materials. (Pierce 2019). In addition, many standard robotics kits do not allow flexibility to be

used in combination with other kits, reducing the possibility of innovation and effectively increasing the costs of obtaining fully functional robots (Susilo et al., 2016). In today's market, educational kits are either highly functional and expensive or affordable with restrictive functionality (Darrah et al., 2018). My project of designing an easy-to-use yet highly functional robotics kit will accelerate and promote STEM education for students around the world.

Current Work:

The LEGO MINDSTORMS kit allows beginners to visualize the different working components of the robot while enabling them to operate these sensors using "block" programming through the custom LEGO software (LEGO, 2019). While these features make sense for beginner students, more advanced students undertaking unique projects will be limited in the types of components they can use and will also be prohibited from viewing and modifying low-end code to add additional customizability. Additionally, as of today, there is no official communication protocol for educational robots, therefore each manufacturer creates their own hardware and electrical scheme, enabling communication solely with products of their own family (Susilo et al., 2016). This means that these kits are not modular and cannot be used interchangeably, limiting schools from using the same kit for multiple different courses and students from integrating unique sensors sold by different vendors (Susilo et al., 2016). Finally, the MINDSTORMS is also sold for USD $359.99 making it unattainable for most students (LEGO, 2019). Other less expensive kits such as the mBot by Makeblock also suffer from the same deficiencies but with the reduced price, the overall amount of functionality present is also reduced (Miller-Klugman et al., 2022).

Plan:

Printed Circuit Boards (PCB) have been monumental in electronic design in the last decades as they allow electronic components to be neatly arranged, connected, and isolated within a single copper plate (Malpan, 2021). PCBs are designed using Computer Aided Design(CAD) that simplifies complex circuits and introduces the possibility of industrial manufacturing and soldering using pick-and-place machines (PCB Unlimited, 2021). A more robust and reliable design paired with the possibility to add more electronics in a smaller space makes PCBs a great tool to be used in educational robot kits.

I have designed a robot that is based on a custom circular PCB that houses all of the sensors, the Teensy Low Cost(LC), motor drivers, battery holder, camera, and various different expansion ports. All electronic parts will be purchased from DigiKey as they are inexpensive when each discrete electronic component is purchased independently. The circular form factor of the kit coupled with the circular sensor arrangement will allow students to explore algorithms that include vector math and achieve parabolic movement of the robot. The previously mentioned expansion ports on the kit will consist of usable digital and analog pins as well as ports for UART, SPI, and I2C devices that will allow students to add many types of sensors and other dependent devices. Similarly, a custom chassis can be designed by students around the many mechanical holes scattered throughout the robot. Lastly the ability to program using any language or software the student prefers is made possible through the many open-source third-party add-ons available for the Teensy microcontroller (PJRC, 2023).

In recent years First Robotics Challenge(FRC) has showcased many teams that make use of “swerve drives” to enhance robot agility, however, swerve drives become impossible in smaller-scale robots (team1640, 2013). Based on this technology, my robot has utilized 3d

printed Omni wheels that will be placed at a 120° angle to recreate the same agile movements on a smaller scale.

III - Project Planning & Development Cycle

…

1. Identifying the objective and goal of a soccer robot.
    a. The Robot should be able to win the Robocup Junior Lightweight
    b. The robot should comply with all rules of the lightweight league
    c. The robot must stand apart from other competing robots
    d. The robot needs to be beginner friendly to allow students to use as a kit
    e. I used multiple CAD software such as OnShape and KiCad to create and design the robots. CAD helps create a more precise robot

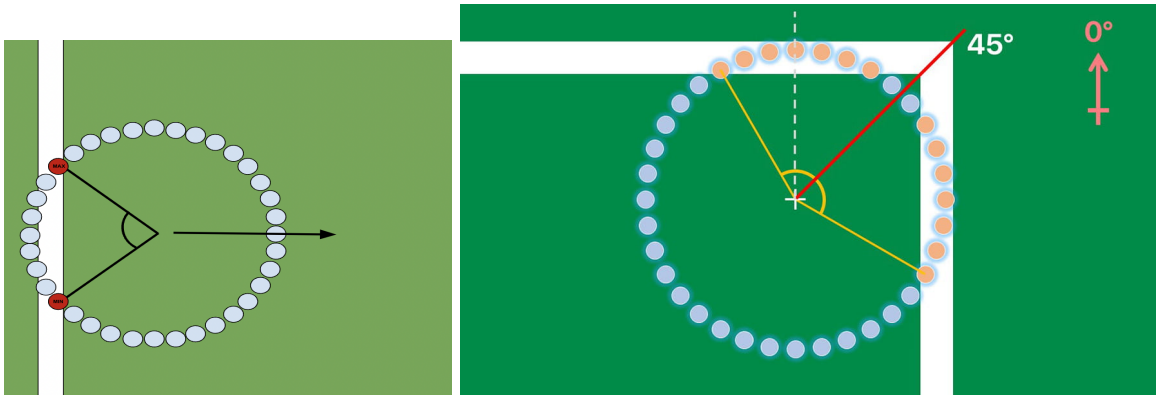
IV - Software/ algorithms

Line Control

One of the primary criteria that the robot has to meet is staying within the boundaries of the field, which are delineated by white lines. This task was mainly achieved with the help of the line PCB. The line PCB was placed on the very bottom of the robot to be able to detect the lines and keep the robot within the field. The photoresistors on the line PCB detect the reflected light from the LEDs depending on what color surface the robot is on, and thus, by setting threshold values, can be used to differentiate between the white lines and the green turf. The underside of the PCB has 24 LEDs and a photodiode in 15-degree increments, allowing it to cover all 360 degrees. The teensy receives data from each photodiode as a value from 0-1023. This number corresponds with the amount of light that the photoresistor can see. Since the only light source on the underside of the robot are the red LEDs on the line PCB the photodiode output data on how light is reflected off of the carpet. If the robot is over a white line more light is reflected.

I then assigned each sensor a degree value from 0 to 360. Using the preset threshold values, the robot is able to know which sensors are detecting white and which are detecting green. The next step was to convert each line sensor into a vector that has a magnitude corresponding to the amount of life reflected. When the robot is over a line, specific vectors are larger due to the increase in light reflectivity. The angle is then easy to find by performing a simple vector

addition to find the resulting vector. The angle of this vector can then by found using the inverse tangent function that returns an angle.



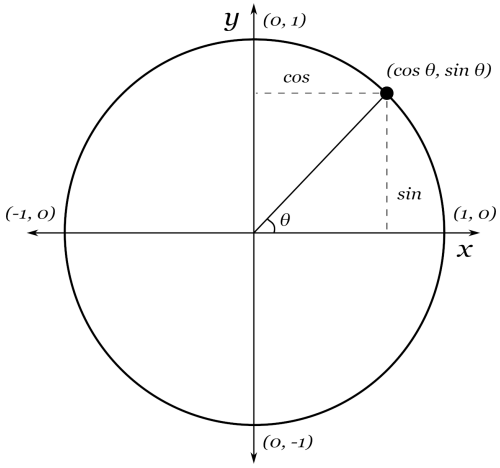
One problem I had with this was the bias toward a specific direction. If more than 2 sensors were activated at a time then the resulting vector would be favored to one side resulting in an inaccurate vector. To combat this I took use of a binary search algorithm that would test the dot product of every combination of activated sensors to find the two farthest sensors. The dot product of two vectors returns a number that corresponds to how far apart the vectors are. After finding the two farthest vectors vector addition is then conducted on these two to produce a more accurate result. The rest of the algorithm was simple. When a line was detected, tell the robot to move in the direction opposite of the line.

While this helped avoid the line for one-half of the robot it did not keep the robot within the field when the robot was pushed more than halfway across the line. When the robot is pushed more than halfway the angles that are returned are pointed toward the inside of the field. If the previous algorithm is performed, moving opposite this angle would cause the robot to outside the field. For this problem, I made an angle comparing functions. Every time the robot detected an angle I would set an initial angle variable. Every new line angle detected afterward would be compared to this initial angle. If the initial angle was greater than 90 degrees it meant the robot had crossed halfway over the line. Using this information it would output the accurate angle that the robot would need to move to stay within the field.

Finally to calculate the power or priority of the line avoidance I used the concept of chord length. I calculated chord length using the chord length formula. If the chord length was near one that means that the motors would have 100% priority to avoid the line. If the chord length was closer to 0 then the priority would decrease accordingly. This allowed the robot to pursue the ball while still being partially over the line, giving my robot a competitive advantage to other robots that would not be able to reach the ball.

Ball Sensing

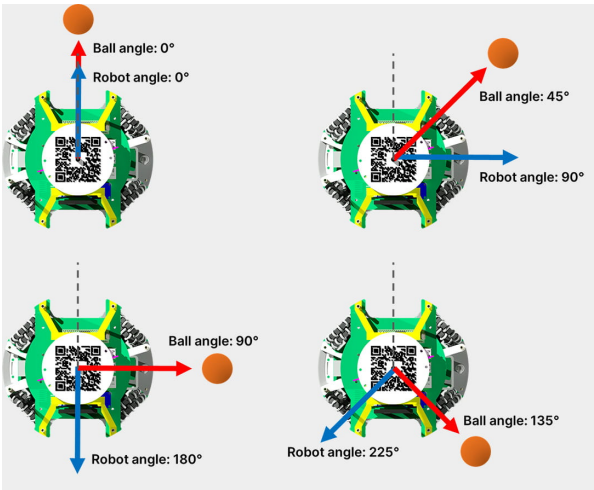
To calculate ball angle I once again used vector calculation. First, I read the sensor values of each ball sensor. Then I multiplied the read values with the sin(ø) and cos(ø). ø being the angle of the ball sensor. After doing this I had vectors of each ball sensor. The sensor that the ball was closer to would have a larger vector pointing in that direction. To get one vector pointing to the ball I simply added all of the vectors. To convert the x and y coordinates of the vectors to an angle I used the atan2() function which returns the inverse tangent, the function takes in x and y as parameters and outputs an angle in degrees.



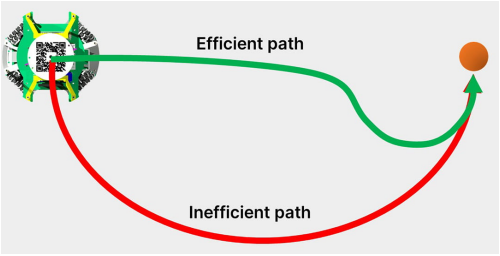
Orbit and Dampen Functions

Although I was able to capture the ball using the ball detection software, the orbit and dampen functions allow me to do this using the most efficient and optimal path, which may not always be a straight line.

So far all I have is the angle of the ball in reference to the robot. However, if I instruct the robot to move straight at the ball there is no guarantee that the ball will be accurately captured in the capture zone. This is mainly because of a lack of a way to keep the ball in our control, as the momentum of the robot would inadvertently bump the ball out of control. The orbit function which is a simple exponential function returns a value that should be added to the current angle of the ball. This value depends on the curve of the exponential graph and results in a very smooth path.



This path is then further optimized using the dampening function. The dampen function is once again another form of an exponential curve, this time however the function takes in a value corresponding to how far away the ball is in reference to the robot. The function returns a number from 0 to 1 that acts as a multiplier to the previously defined orbit value. This function reduces the amount that the robot's path alters from its original straight path. This is important as it prevents the robot from following an unnecessarily long path. These functions together create a path that is mostly straight other than a slight smooth curve as the robot approaches the ball to maintain control.



Rotation Correction:

the robot always faces forward at all times. This is done because taking paths that the orbit function returns is much more efficient than rotating the robot and then collecting the ball. However, ensuring that the robot always faces the front proves to be a challenge due to uneven weight distribution and different friction coefficients of the wheels. To solve this I added a compass sensor that returns the current orientation of the robot relative to the earth's magnetic field. Based on this data I created a simple correction algorithm that would slightly add power to certain motors depending on if the robot orientation was offset from its original trajectory.

Camera

Color detection with the openMV was relatively easy as I was given very great example codes in the application. All I had to do was set color thresholds which are able to detect specific colors in the image. After I detected the goal I compared the x and y coordinates of the goal to the

coordinates of the center of the image. Finally, I compared these coordinates and once again used the atan2() function to find an angle to the goal.

I was also able to find the distance the camera was relative to the goal. I found this by using focal length and comparing the size of the goal as the robot moves closer or farther away.
Theoretically, this works great however I found that this data was not very reliable. For this reason, I only used the goal distance for very limited uses.

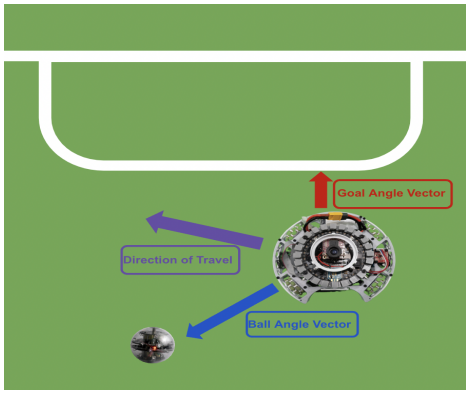
Xbee

I use two Xbee S2C radios configured using ZigBee protocol to communicate between the two robots locally. On paper this was meant to work perfectly, however, during testing, something always went wrong. I am able to read and write data between the robots but it is possible that the transmission speeds are too slow for the robot to receive instructions properly. The plan was to use this radio to communicate current roles between the robot. If one robot was closer to the ball than the other, then the robots should be able to switch roles between offense and defense. I also found that it was necessary to set "switching thresholds" as in a real match, if one of the robots would have to be taken out of the game, the remaining robot must switch and stay at offense at all times. So to do this I had to set thresholds on the line sensors to detect when a robot would be picked up and then send a hexadecimal bit to the other robot, forcing it to stay at offense.

Defense

Programming defense proved to be a great challenge as there were many new constraints with the new field design. the main algorithm surrounded vector addition once again. I calculated both the ball angle and the goal angle and used the two angles to place the robot between the two. While this seemed easy many problems arose.

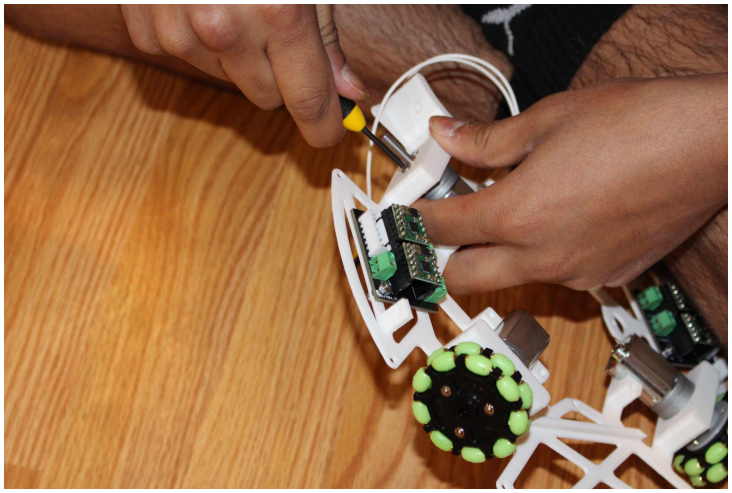
The most noticeable problem was that when playing the role of defense, at times, the robot would chase the ball as opposed to defending the goal. Essentially over time the robot would creep up to the middle of the field. This was because the robot does not have any sensors that give it depth perception. Therefore there was no way for the robot to know how far away from the goal it really is.
To combat this issue, I designed an algorithm calculated the time it spent away from the goal line, if the time exceeded a certain threshold then the robot would be forced to move back until it "saw" the line again.



V- Hardware - Mechanical and Electronic design and manufacturing

V-1 Mechanical Design and Manufacturing

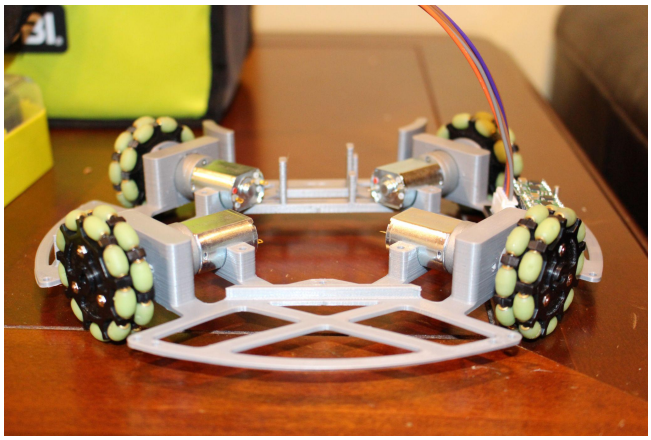
V-1.a Design



I designed a powerful lightweight robot using 4 DCX19 motors. I paired these motors with custom-made Omni wheels that attached to these motors in a very innovative way. Essentially the wheel is designed so the shaft adapter(the thing that is attached to the motor shaft enabling wheels to be mounted) is embedded within the wheel. This made it so that additional space was saved between the wheel and the motor. Additionally, I also added a kicker because I believe it will improve scoring speed and accuracy. To aim toward the goals I used an OpenMV cam and machine vision that provides angle using color detection of the goal. In order to detect the ball I designed a PCB that holds 24 Infrared receivers that can detect the pulsed signal of the RCJ ball. Finally, to keep the robot within the field I used 24 custom-designed line sensors that gave values

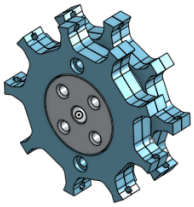
between 0 and 1023 when a line is detected. To control and take in all of this information I used a teensy 4.1 because of its extreme speeds and reliability. To design the custom pieces I need to hold these parts I used FDM 3-D printing using PLA material as it proved to be light and strong.

V-1.b Chassis



In order to meet the weight requirements while maintaining structural integrity and accuracy I designed and 3-D printed a chassis using the CAD software Onshape. Angling the motors at 80 degrees, I achieved a higher effective gear ratio when moving forward and a lower ratio when moving sideways, as opposed to what would be possible with a 90-degree orientation. By creating separate outer walls I were able to shield the wheels and prevent any possible catching. I created a compartment with a separate door to house the 3S 1300 mAh LiPo battery which powers the robot.

Custom Omniwheel



The factory-made Daisen Omniwheels used a mounting mechanism that was incompatible with our Maxon motors. To solve this problem I designed my own Omni wheels with a D-shaft compatible mount that is able to use the same rubber rollers from the Daisen Omni wheels. Finally to ensure secure attachment I used the Pololu-made aluminum adapter that uses a set screw that screws on against the d-shaft to ensure secure attachment with the motor shaft.

Solenoid Kicker

Additionally I used a solenoid that works as a kicker. I used this Solenoid as a kicker due to it having the power to reach the Kicker limit for the competition. When I first tested the solenoid kicker, I did it without any modification in order to test its initial accuracy so I could determine if I needed any modification done. Just as I predicted, the kicker was not accurate so I

designed a Kicker Plate which was clamped to the kicker with 2 screws in order to have better accuracy relative to the goal. A disadvantage this kicker had was its weight. Because of the weight of the Kicker, I had to compromise other parts of the robot or make parts of the robot lighter.

Ball Sensor Shield

To further limit inaccuracies in ball detection I needed to have something isolating the infrared sensors into one angle, which is why I have the Ball Sensor Shield. the Ball sensor Shield is a 3D-printed plastic cover that goes over the sensors that cover up the back and the sides of the ball sensors while leaving a 2mm gap in the front of the sensor to allow proper detection. It was challenging to design this because I did not know what the dimensions of the ball sensors would be after soldiering them so I had to search for 3d models online and use careful estimation.

Mirror design

In order to first determine the measurements of the mirror I needed to use Onshape which is the CAD software to calculate the mirror angle and mirror diameter using the camera's FOV.

The plan was to thermoform mirror foil using a mold with these calculated angles. This meant that I needed to create a mold that is extremely smooth and polished. I realized regular FDM 3D printers would not be able to achieve the level of accuracy I needed for the mold. Looking at other alternatives, I found SLA. I first used SLA 3d Printing and used high grit sandpaper to smoothen the mold out, I then molded the mirror sheets by using heat. Unfortunately, this did not work very well as there were many small bumps and curves that corrupted the reflection. For a second try, I found a once again used NylonX. I went through the same process and 3D printed a mold and created a mirror as aforementioned. This time the mirror seemed to be much more clear and more functional Finally, I used a clear acrylic tube to keep the mirror floating above the camera to view the field omnidirectionally.

CAD software

I fully designed the robots in Onshape. This CAD software has many advantages over other popular choices such as Fusion360, Solidworks, and AutoCAD. Since Onshape is browser-based, it is easy for anyone to view or make changes to the CAD files making it a much more convenient CAD Software. Compared to industry standard CAD software, Onshape is much easier to learn and design small robots with. Onshape has many parametric features that help make the CAD model more robust, it helped us make all the precise little details that helped us have a more exact robot which eventually helped with the building part of the production of the robot.

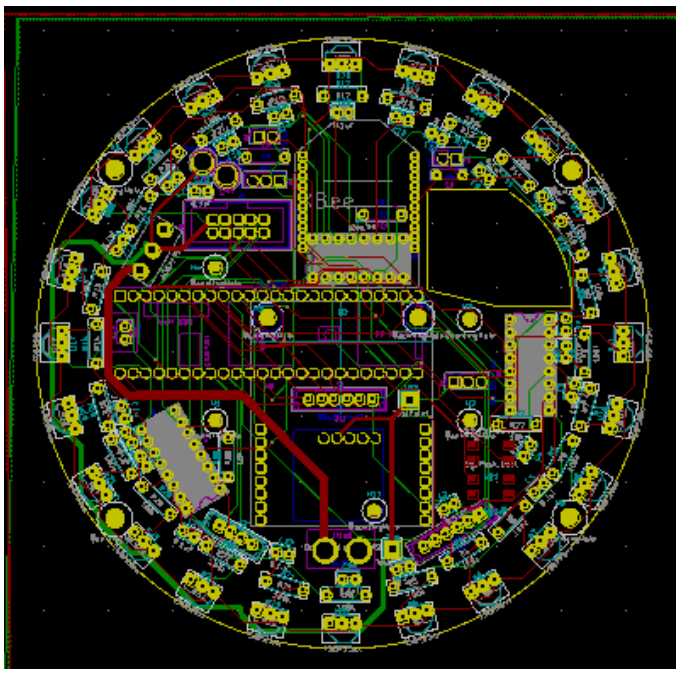
3D printing

One distinguishing feature of the robots is the extensive use of 3D printing. I used 3D printing technology which allowed us to make complex designs, and at the same time have lightweight parts and countless design iterations. FDM 3D printers are generally much cheaper than CNCs,

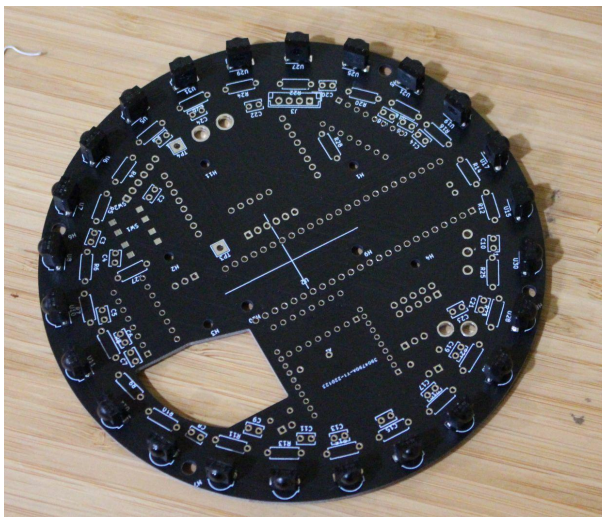
laser cutters, and other traditional fabrication machines, and make iteration easier and cheaper. I used a Prusa i3 MK3S with a custom enclosure and a heavily modified Creality CR-10.

Electronic Design

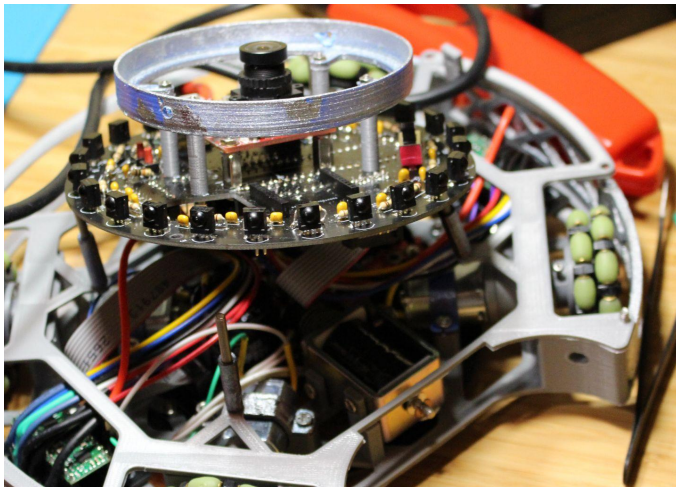
Main PCB



the main PCBis placed on top of the robot and holds the 24 ball sensors, OpenMv cam, Xbee, voltage regulator, multiple switches and buttons, ports for the light gate (led and photoresistor), and a voltmeter. The teensy 4.1 is placed on the underside of the main PCB to save space for size-constrained parts such as the camera and Bbee. While designing the board I were able to use the circular pattern function on kiCad to accurately place the pins for the ball sensors at 15-degree increments. Additionally, because of the large amounts of sensor data, I would receive I implemented the use of analog-to-digital converters that would take in 8 analog signals. This significantly reduced the number of pins that would need to be used on the teensy. One problem I had was converting the digital signal from the ball sensors to analog signals that the ADCs would be able to read. To solve this I used low-pass sensors that would output the average signal of each ball sensor. I used a 100k resistor and 0.1uf capacitor to create this low-pass filter. The ADC communicated with the teensy using the SPI communication protocol. This meant that each ADC would need a MISO, MOSI, CLK, and Chip select connection. It is also important to note that the miso MOSI and CLK pin are shared with all the ADC used. Another communication I used extensively was the UART protocol which was used

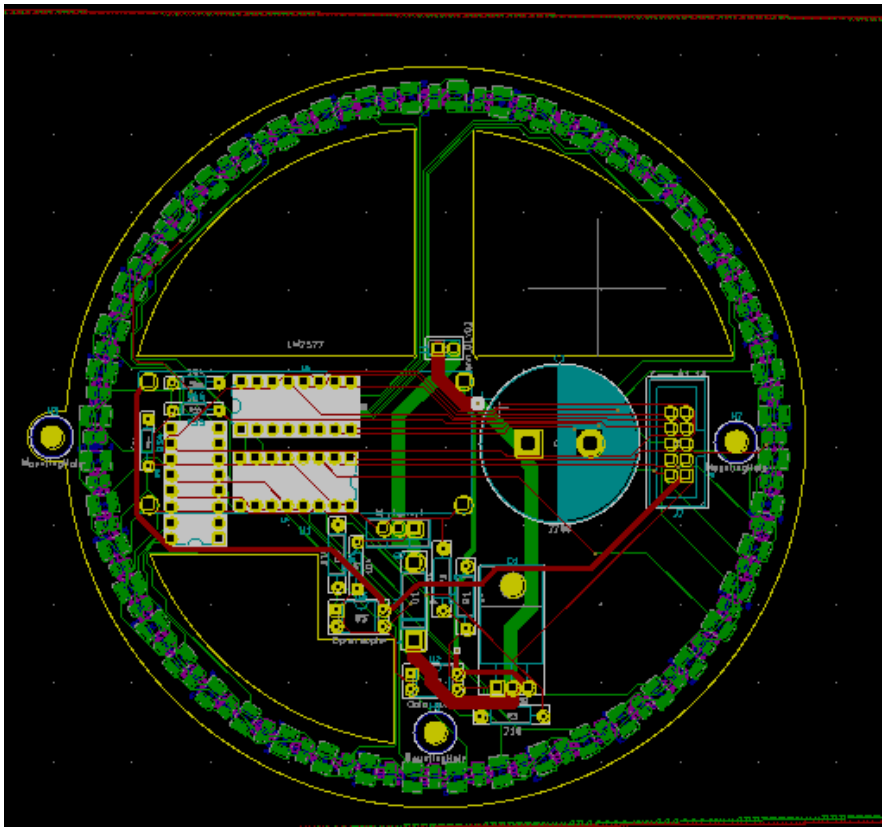


for the camera and Xbee. The OpenMV camera also had a microcontroller inbuilt which I was able to program to process and calculate the angle from the raw camera images.



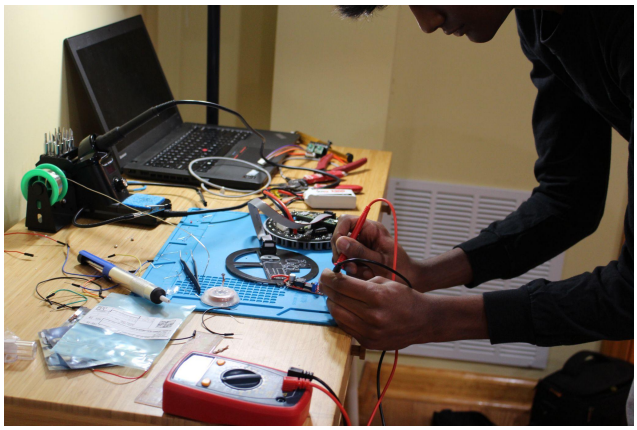
While designing the PCB I prioritized through-hole components as it would be easier to hand solder.

Line PCB



The line PCB was placed on the very bottom of the robot to be able to detect the lines and keep the robot within the field. The underside of the line PCB had 24 LEDs and 24 photoresistors to detect the reflected light. The photoresistors detect the reflected light from the LEDs depending on what color surface I am on. Once again using the circular pattern tool I was able to place both the LEDs and photoresistors at 15-degree increments. This made sure that in every scenario at least one LED and photoresistor would be on top of the line. Another problem I faced was the relatively small wheels which resulted in extremely small ground clearance. To solve this problem I used surface-mount LEDs and photoresistors that projected out a mere 1 mm. Although this made it hard to solder it still helped keep the line PCB from scraping the ground.

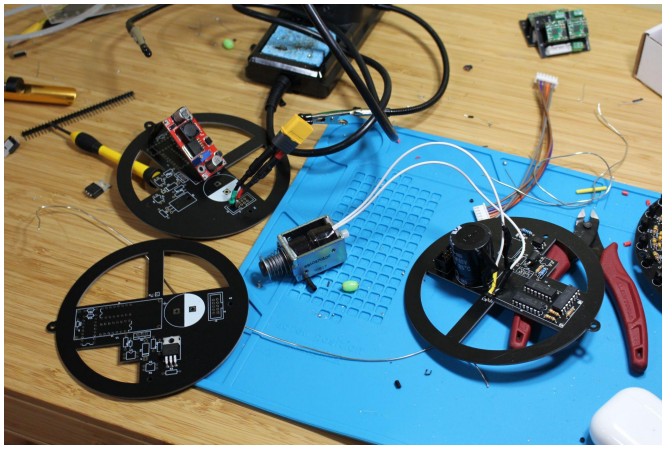
The other purpose for the line sensor was the inclusion of the kicker circuit. The kicker circuit was extremely hard to design and I encountered many failures and damaged many parts. the kicker, the CB1037 open-frame solenoid required an extremely high voltage at a high current for it to be able to kick significantly. To obtain these high voltages from the 12v battery I used a buck converter that converted voltages up to 60v. After creating this 60v current I used a very large 2200uf capacitor to store a very high current. High voltages as such raise another problem, which is the inability to control these voltages using the teensy. To solve this I used a series of MOSFETs and optocouplers that let us use the teensy signal to control the high-voltage circuit.



Testing the boost converter for the first time to see if it would output a 60v current

First the teensy signal gets sent to an optocoupler. When the optocoupler receives power from the teensy the other two pins of the optocoupler send an appropriate signal to the gate pin of an N-channel MOSFET. This MOSFET, depending on the gate signal, either closes or opens the source and drain pins. In the circuit, this N-Channel MOSFET connects the kicker solenoid to ground. This means that when the MOSFET is open the solenoid is not connected to ground, but the moment the teensy sends a signal the MOSFET would close, connecting the kicker to ground and causing the kicker to fire.

The last thing I included was a protection circuit for the kicker. As explained previously the kicker used many amps to be able to fire. As a safety measure I also included an exact replica of the above circuit but instead using a P-Channel MOSFET, This P-Channel MOSFET disconnected the kicker circuit from the rest of the robot every time the kicker fired. This ensured that the rest of the robot wouldn't lose power while the solenoid kicked.



Debugging electrical problems with the kicker circuit. Many failed attempts,

VI - Constraints and Future Interest

1. The absence of a dribbler. This year I could not implement a dribbler because of weight limitations. However the dribbler is massively useful as it can hold the ball within the robot capture zone even when moving backward

VII - Conclusion

I had a great experience in learning new things such as 3D CAD and Electrical design, selecting and purchasing mechanical and electrical parts, assembling them, programming, testing and making it work.

The robot performed extremely well overall. Not only did the robot win first place in the US, but it also won the Best Engineering Award and the Best Algorithms Award. On the world stage, I

learned that there are a lot more improvements that I can make to the robot and take it to a whole new level.

Over the summer this robot also worked really well as an educational kit. As the president of Princeton Soccer Robotics, I am honored to have the opportunity of sharing my knowledge with interested students. I continue to introduce beginners by allowing them to use this robot to learn about circuits and software algorithms. Unlike previous years, new students are instantly motivated when they are given a physical kit that they can see physically moving and responding to changes they make.