# MCEN 4115-5115: Mechatronics and Robotics I


## Team Hodor:

Faizan Mir

Vinodharani Murugadoss

Santhosh Lakshmanan

Ramakrishnan Balasubramanian

Phillip Wu

# 1. Introduction

The competition held for Mechatronics and Robotics this semester drew upon aspects of rocket league and soccer. The competition was to fabricate and design a robot that can find the yellow ball located on the field and hit the ball into the opposing team's goal, earning a point. The robot was to compete completely autonomously against one other robot per match.

This semester's competition was quite difficult as there were many things that the robots needed to account for and many different ways to compete. To start, the robot had to know where it was located in relation to the ball. This could be done using radio communication as well as some sort of CV, Computer Vision. This is hard enough as is, but then the robot still needs to take into account the location of the goal and opponent as well. After doing all of this, the robot still has to make decisions based on all of these factors and determine the best course of action. This report will go into our team's design and approach to this challenge, highlighting some of the critical decisions we made along our path.

# 2. Mechanical Design

## 2.1 Chassis Design

The main design of our robot featured three layers made out of MDF and separated by 3.5 inch standoffs. MDF (Medium Density Fiber) was chosen for layers as it has a smooth surface, is easy to manufacture, is heat resistant, and most importantly, is rigid and structurally sound. The MDF sheets were designed to be laser cut for precision and accuracy. Single-layer parts were laser cut and a few complex parts were 3D printed in PLA because of its ease of manufacturability. The differences between each layer and what components they each house is shown here:

1. The 1st bottom layer consisted of the wheel assembly, and its associated components like motor, power sources (battery, power bank).
2. Next upcoming layer is the heart of the robot, having the Raspberry Pi, assembled printed circuit board, an Arduino board, motor driver, and a transistor for the hitter setup.
3. The final top layer has the director of the entire robot, the Pixy cam, which guides the entire robot, the ball holder (for team identification), and a switch for turning the robot setup. A handle is placed on the bottom of this layer for lifting the entire setup
4. The hitter mechanism, which was a powerful setup, with torsional springs for creating more tension when the ball nears the robot.

The robot was designed in such a way that no component exceeded the 3 layers' dimensions. The design was also tested for strength and rigidity. Mecanum wheels were chosen because they allow much higher maneuverability than other designs that we were considering. Maneuverability is important because there are going to be many different ways we are going to want to position ourselves in order to hit the ball. We decided to use high-speed motors (250rpm) because we felt that moving quicker would allow us to take complete control over the ball throughout the game.

**Here are some photos of our robot and the basic dimensions governing our design:**

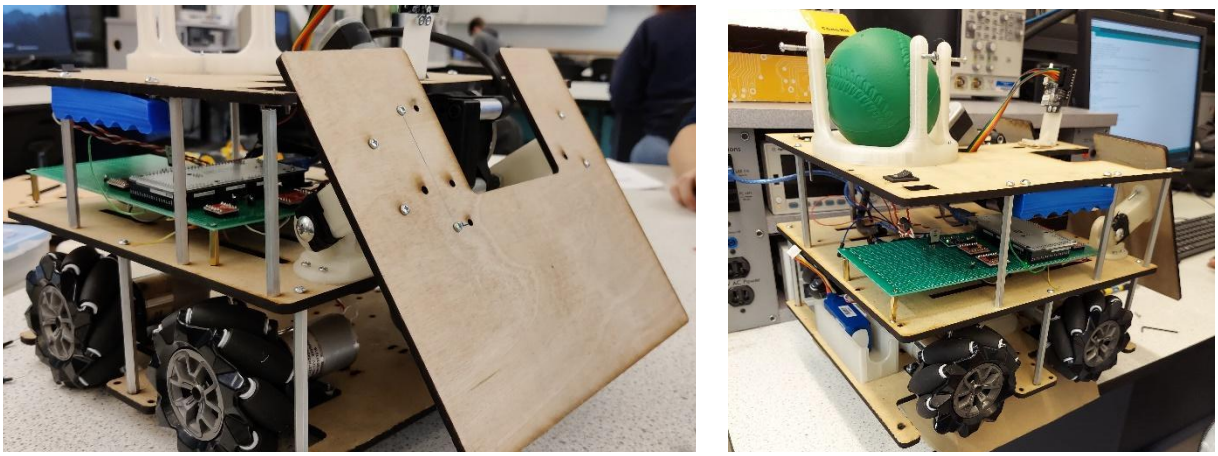


**Figure1**: Isometric views of the Robot

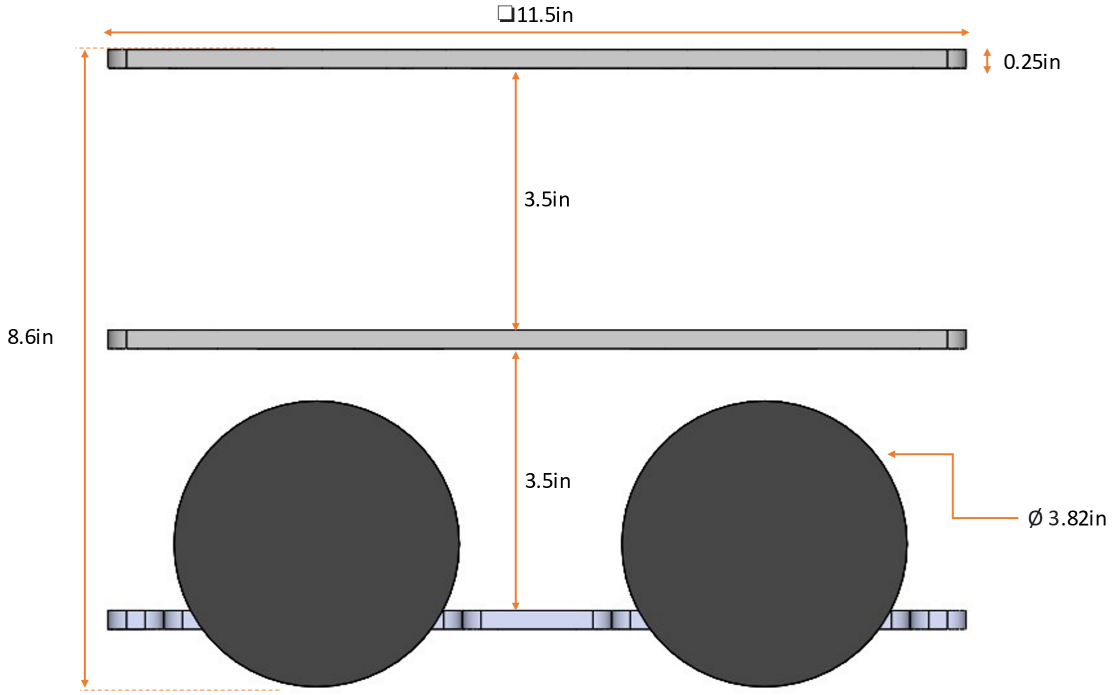



**Figure2**: Side view of the Robot

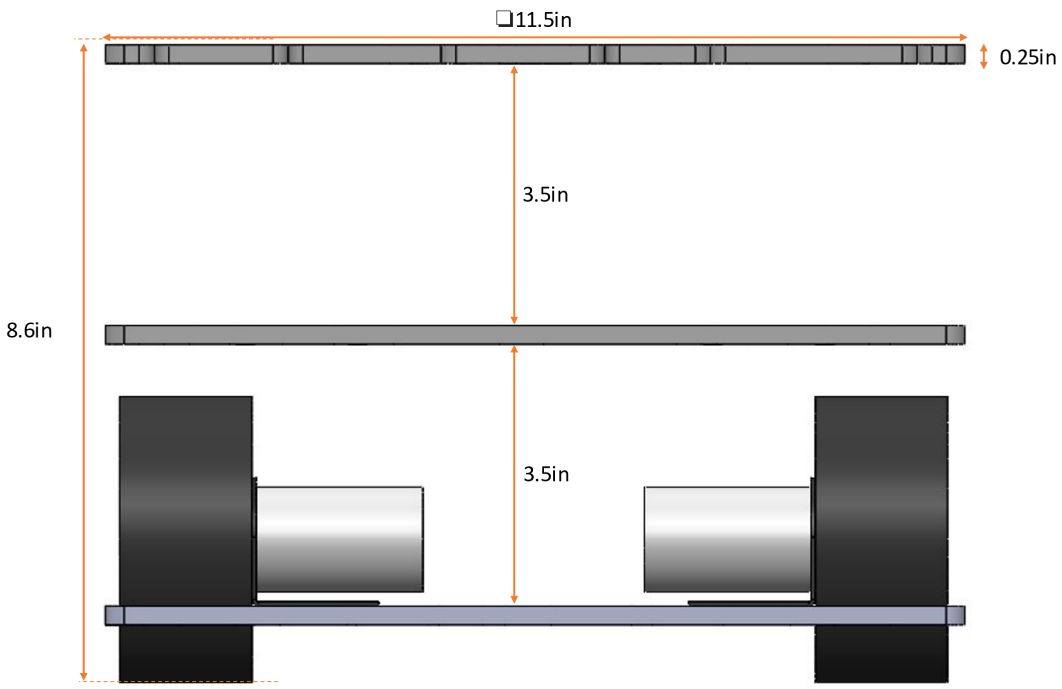



**Figure3**: Front view of the Robot

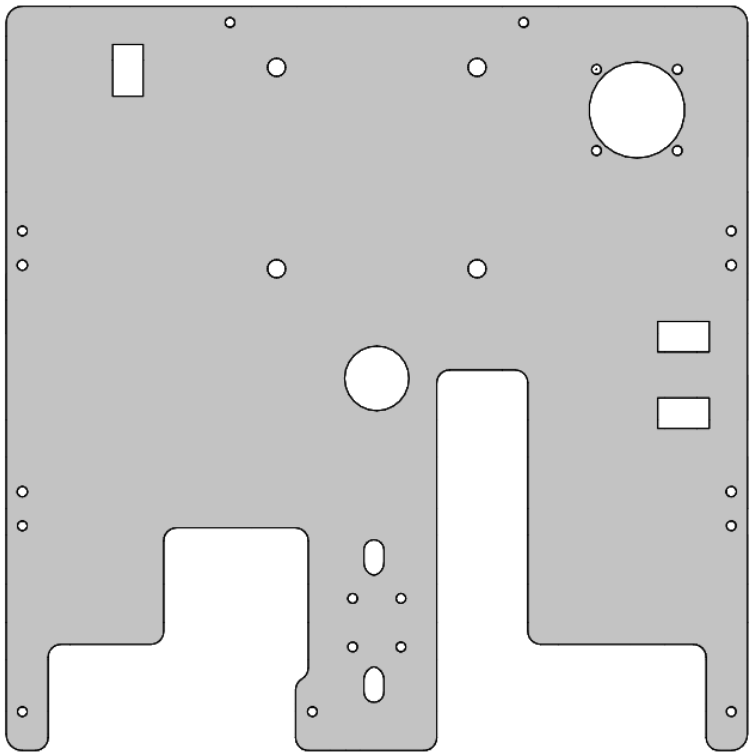


**Figure 4**: Top Layer of the Robot

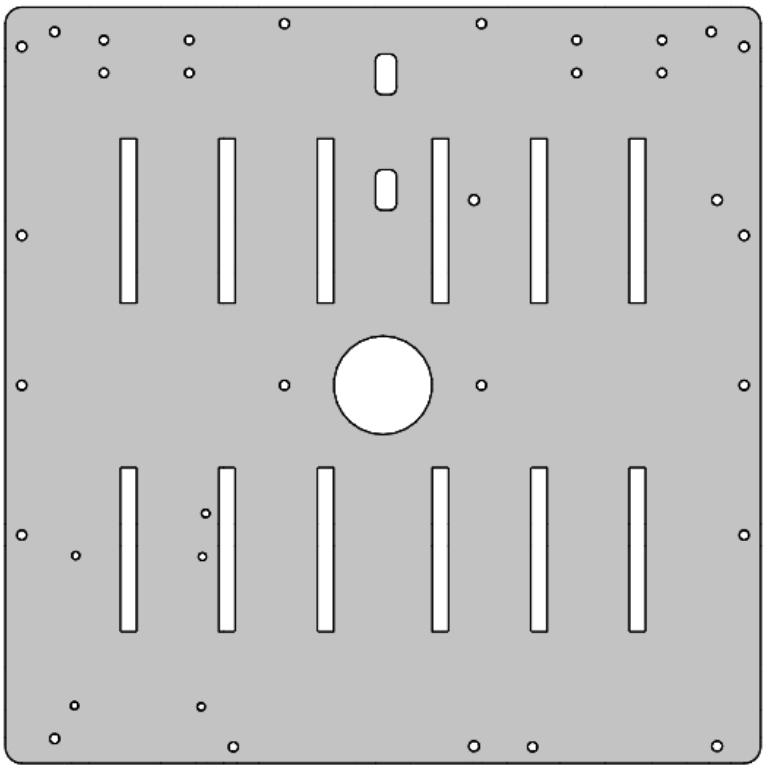


**Figure 5**: Middle Layer of the Robot

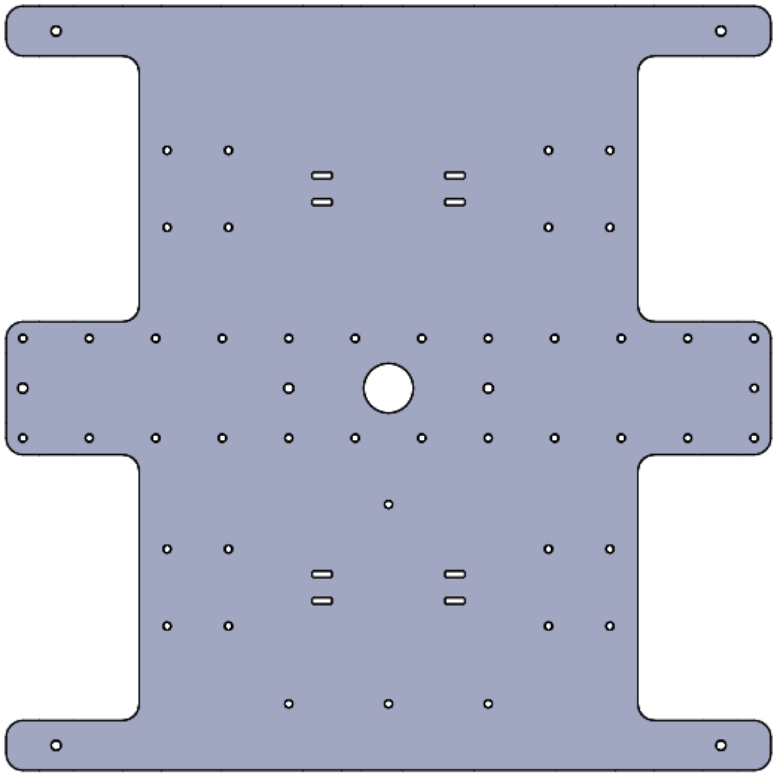


**Figure 6**: Bottom Layer of the Robot

## 2.1 Hitter Design

The hitter mechanism has been implemented to hit the ball quickly and accurately into the goal. Since the ball is quite bulky and hard to move, we decided that the hitter should provide as high an impulse to the ball as possible. This would cause it to move quicker towards a particular direction than with less impulse. Due to the cost constraints and complexity of some of our parts, we decided to 3d print most of the parts for the hitter. We chose to use PLA since it has high durability and strength while also being relatively inexpensive.

Our hitter went through 2 major designs throughout our project:

**Iteration 1: Reciprocating hitter**

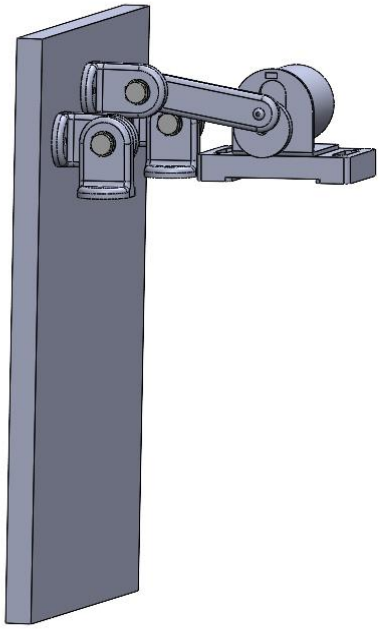


figure 7:- hitter with reciprocating mechanism

- The first version of hitter is simplistic; it is a reciprocating 4 bar mechanism with a linkage connected from the motor to the hitter and the hinges fixed on the top layer of the robot.
- the hitter turns 45 deg when the motor crank rotates a full rotation
- The link lengths and positioning of hinges are provided in such a way that the above mentioned motion is accomplished.
- The motor used here is Jameco 12vdc motor that has a 110 RPM and torque of 1300 g-cm

**Advantage**

- the hitter has can be controlled to a high degree of accuracy using control system because of the continuous control of the hitter by the motor
- The system does not suffer high levels of stress because of no flexible or elastic component within the assembly.

**Disadvantage**

The mechanism does not provide the required impulse because of a low performing motor. A bigger motor is needed in order to provide an impulse that we are happy with.

**Iteration 2: Spring loaded cam mechanism**

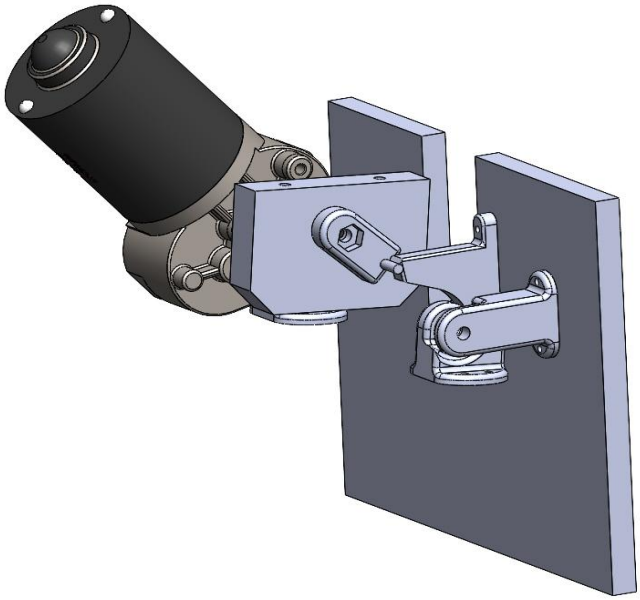


figure 8 :- spring loaded mechanism

- The second version adds more complexity to design but has more potential than the first one
- This mechanism is spring loaded using the power of the wiper motor.
- the mechanism contains a torsion spring that has been provided a seating the hinge mechanism itself so that when the motor is actuated the spring get compressed and thus loading the mechanism
- there is a special type of cam which is inspired from the geneva mechanism that will help in loading and unloading the mechanism

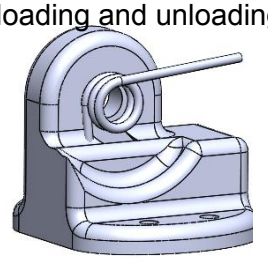


figure 9: Hinge with springs

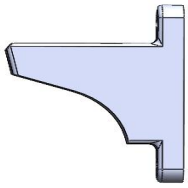


figure10: Cam

- The mechanism is always present in the vertical loaded position
- When the spring load is released, the mechanism goes to a 45 deg position and in that motion it will hit the ball with an impulse produced by the spring and the wooden hitter

**Advantage**

- The impulse provided by the loaded spring is enough to make the ball move at a walking speed.
- This method does not require a powerful motor to provide an impulse force.

**Disadvantage**

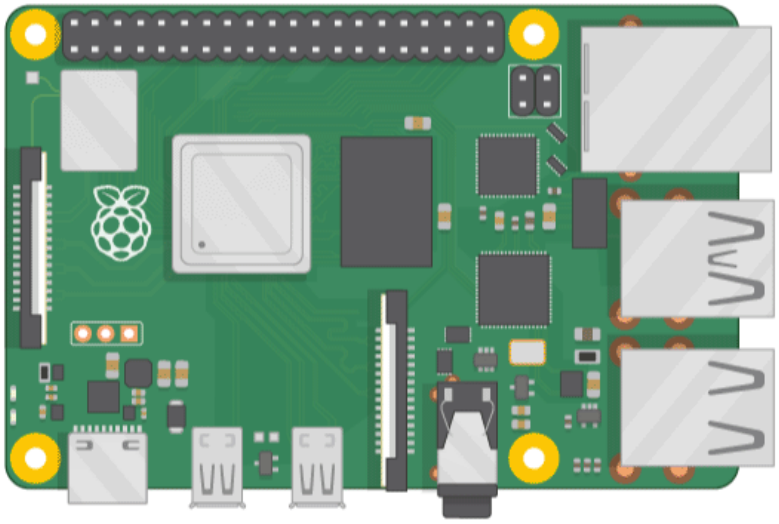
- The spring tension makes the component prone to stress and needs to have materials that will survive such stress.

# 3. Electrical design

Components Used:

## 3.1 Microcontrollers

Raspberry pi 3:



Raspberry pi 3 is a Single-board computer with wireless LAN and Bluetooth connectivity. It has the following functionality,

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 100 Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

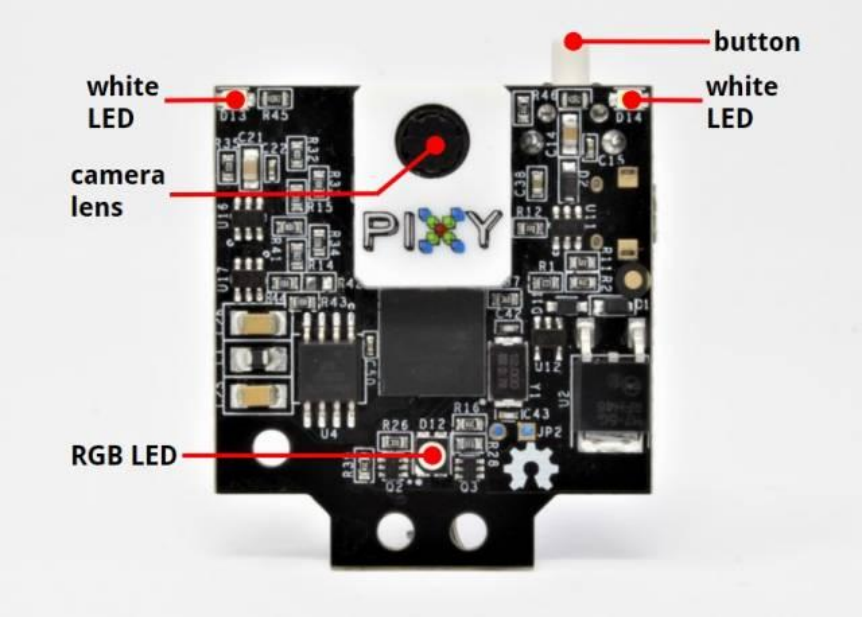
In this project we use the PI to send serial data from one arduino to another. We prefer using the pi because of its high computational speeds and power. This makes the system have faster response time and be able to “think “ faster than other microprocessors.

## 3.2 Arduino mega:

The **Arduino Mega 2560** is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 Analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

We have used Arduino to get data from all the sensors and send the data it receives to the pi. The controller then receives the decisions made by pi and activates what it needs to in order to do what the pi told it to do.

## 3.3 Pixy Camera:



Pixy is a fast vision sensor you can quickly “teach” to find objects, and it connects directly to Arduino and other controllers. Pixy uses a hue-based colour filtering algorithm to detect objects.  It’s capable of tracking hundreds of objects simultaneously and only provides the data you are interested in.

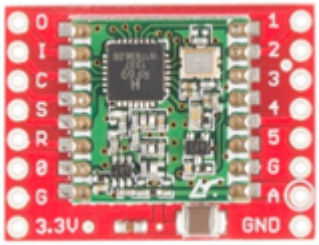
We have connected the pixy camera using ICSP to the Arduino. Frist, we have trained the pixy to identify the yellow ball (the ball of the game), so that every time the pixy sees the ball, it captures the frame and the pixy ccc library gives us the width, height, x and y axis of the all as data. We use that data of pixy to move our robot towards the ball.

## 3.4 Motor Driver:


Motor drivers act as an interface between the motors and the control circuits. Motors require a high amount of current whereas the controller circuit works on low current signals. The function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor.

We are using these motor drivers to drive the motor of the mecanum wheels. We connect the PWM signals of the motor driver to the Arduino PWM signals and give the speed constraints using that. We have used 2 motor drivers for the 4 wheels, which acts as a bridge between the wheels and Arduino mega.

## 3.5 Radio Receiver– RMF69

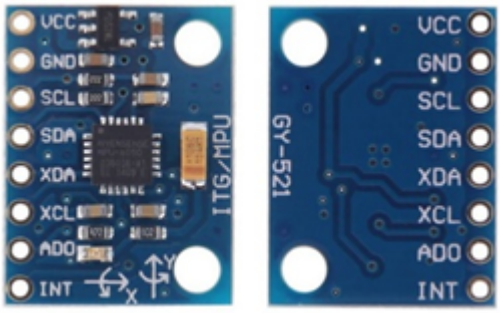


The RFM69HCW is a transceiver module capable of operation over a wide frequency range, including the 315,433,868 and 915MHz frequency bands. All major RF communication parameters are programmable and most of them can be dynamically set.

The RFM69 operates in 3.3 Volts and is connected to Arduino using BI-direction voltage convertor. The antenna is connected to the A pin of the transceiver and it receives the data from sky pixy which was 915MHz in our case.

After receiving data from RMF69 we have used it to obtain the coordinates of our position, opponent position and the ball position from sky pixy connected. These positions help us identify whether we are having to go on defence or offence.

## 3.6 Gyroscope:



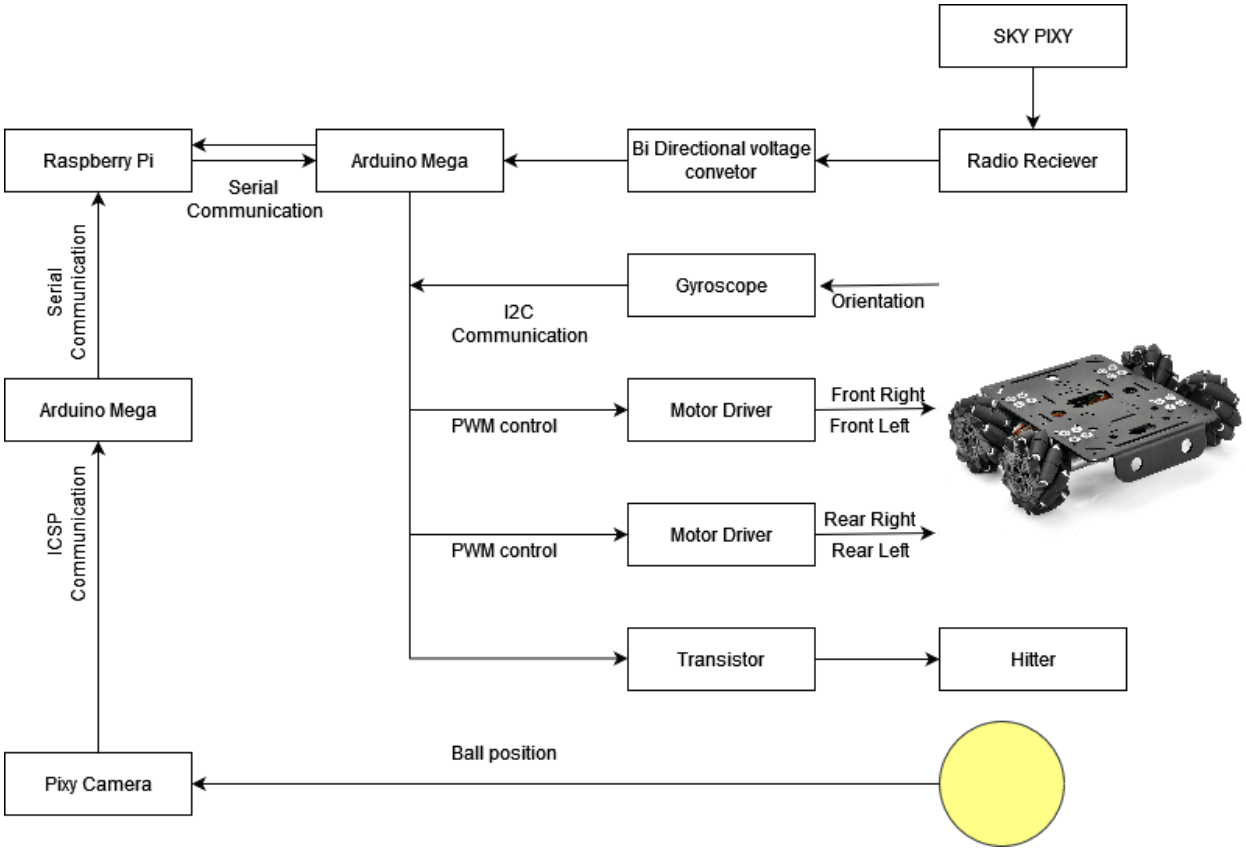

The GY521 contains both a 3-Axis Gyroscope and a 3-Axis accelerometer allowing measurements of both independently, but all based around the same axes, thus eliminating the problems of cross-axis errors when using separate devices.

GY521 is connected to the Arduino mega using the SDA, SCL that is the I2C connection from Arduino to the gyroscope. In our project we have used GY521 to calculate the yaw value, thus calculating it gives us the position of our vehicle. Calculating the position gives us heads on which direction we are headed, and helps us in orienting ourselves towards our home goal for defence and to score goals during offence.

## 3.7 Motor: Hitter

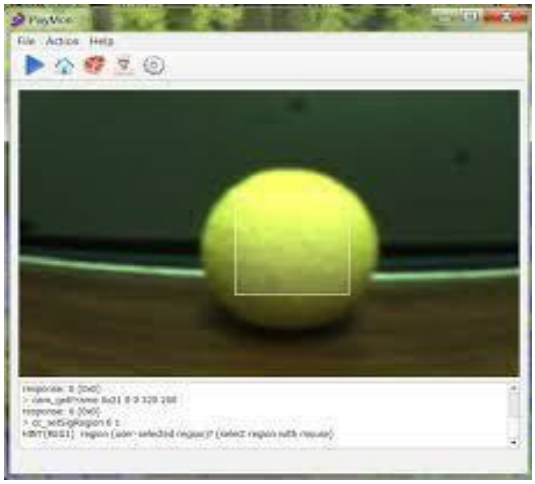
For the hitter we have used a wiper motor coupled with IRF540 MOSFET. The motor tends to use more power than an Arduino input can handle directly, so we have used an IRF540 transistor in order to power the motor. The IRF540 is an N-channel MOSFET used for fast switching and amplification purposes.

# 4. Circuit Design:



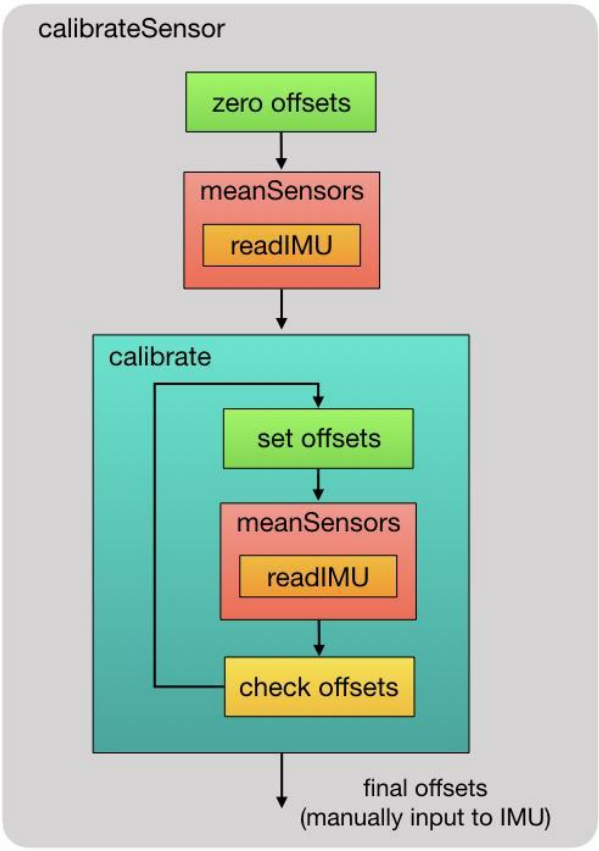
# 5. Algorithm

### 5.1 Ball Detection



Our Pixy2 camera was paired up with an arduino mega for ball detection. The color connected components program was used for ball detection from the Pixy2. The data that the pixy processes sends the location of the ball as x and y coordinates to the arduino. The Pixy2 camera is easy to set up but in order to get better results the camera settings were configured to detect the object in low light and the detection settings were refined for better detection of the yellow ball.

### 5.2 Robot Orientation

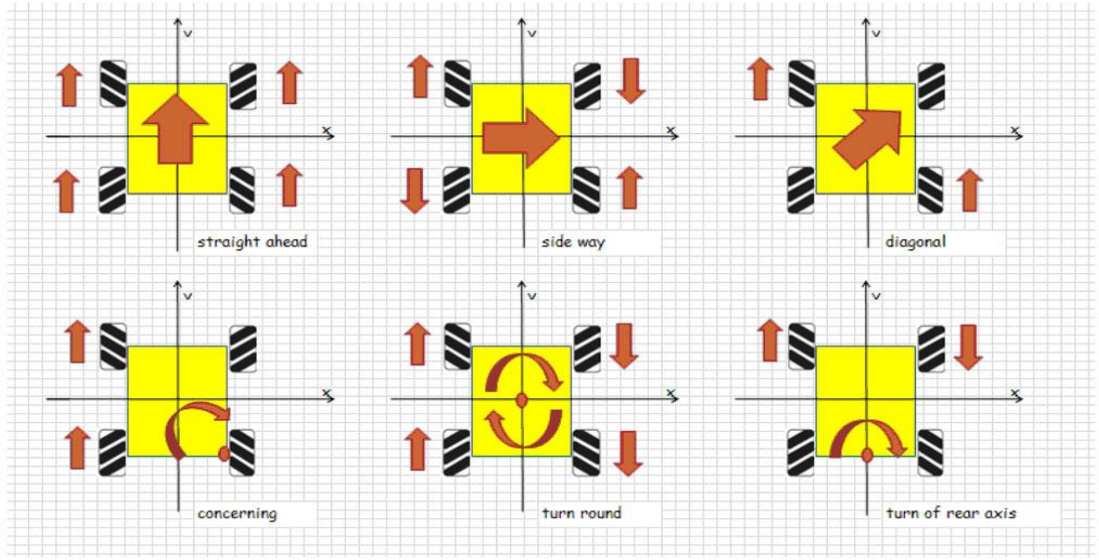



An inertial measurement unit IMU 6050 was employed to determine the robot's orientation. This was necessary for our setup as we wanted to identify the opponent goal which was not possible to specify beforehand as the team color would be randomly allotted in the competition.
At the starting all the accelerometer and gyro offsets are set to zero and then we take 100 readings from the IMU and calculate the mean value for each offset. These values are then entered into the IMU to become the new offsets. The calibrate routine will continue to take the mean IMU readings until the calibration is within a certain tolerance.

The Quaternion values are passed into the *dmpGetEuler()* function to transform them to *Euler* angles. After calculating the Euler angles, we can calculate the yaw by simply integrating the change in yaw value and passing them through the filter to remove any unaccounted yaw changes

### 5.3 Robot Motion

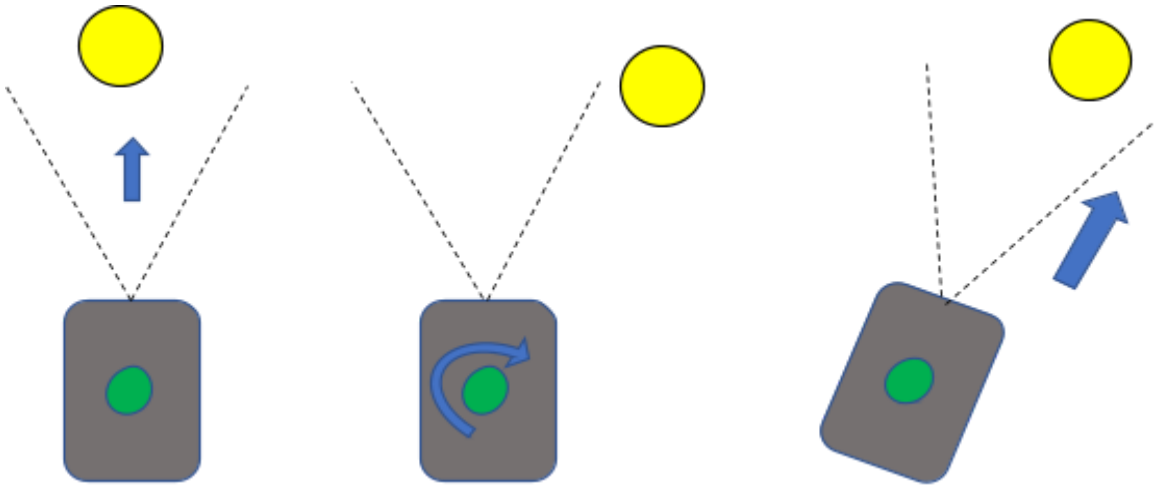
Arduino was used to control the four motors attached to the wheels. The robot used mecanum wheels and these were programmed in arduino to move forward, backward, rotate clockwise and counterclockwise and move in diagonal directions. To achieve these motions each wheel was programmed individually as per the below figure.




### 5.4 Ball Tracking

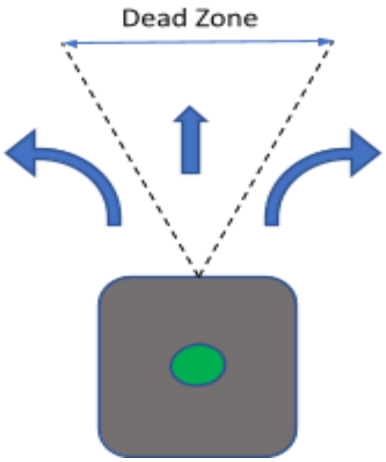
An arduino mega recieved ball signature, x, y coordinate and these values were normalized using 'mapfloat()' function to help with computation before sending to raspberry pi. Due to the low processing speed of the arduino all the computation was done using raspberry pi.

When the pixy camera detected a yellow signature, the arduino would send that data to pi for processing and would rotate clockwise if it couldn't see the ball.



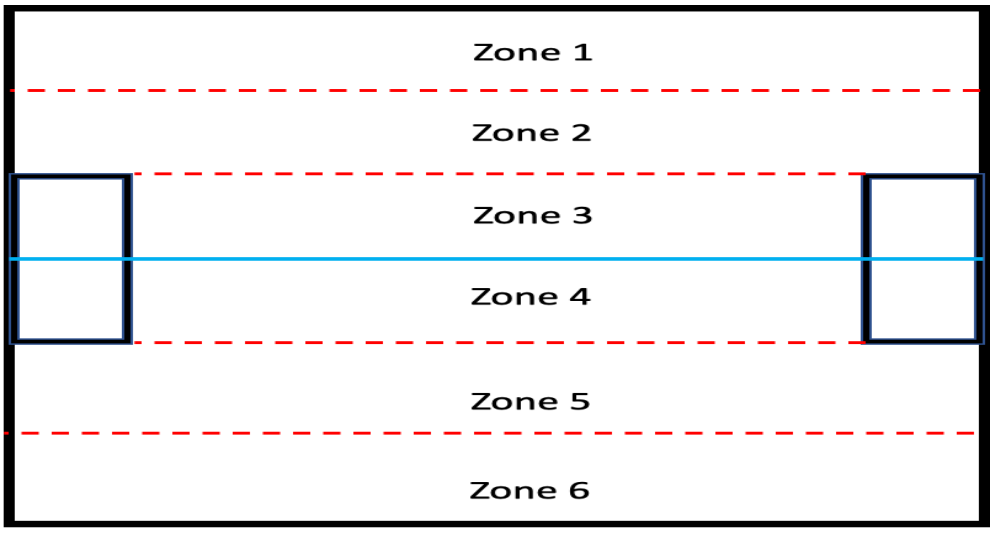
a) Ball signature received from Pixy. b) Robot turning clockwise as ball signature not received from pixy. c) Ball Signature received from pixy and sent to Pi

Raspberry pi was used as the main microcontroller to perform all the computation for the robot due to its higher processing speed. We defined a dead zone for the x-coordinate of the ball between the range of -0.20 and 0.20. This helped the robot identify the location of the ball in its frame. The robot would make a right turn if the value was greater than 0.20 and left turn if it was less than -0.20. This helped the robot to navigate in the direction of the ball and always keep the ball in front of the robot.




### 5.5 Scoring Algorithm

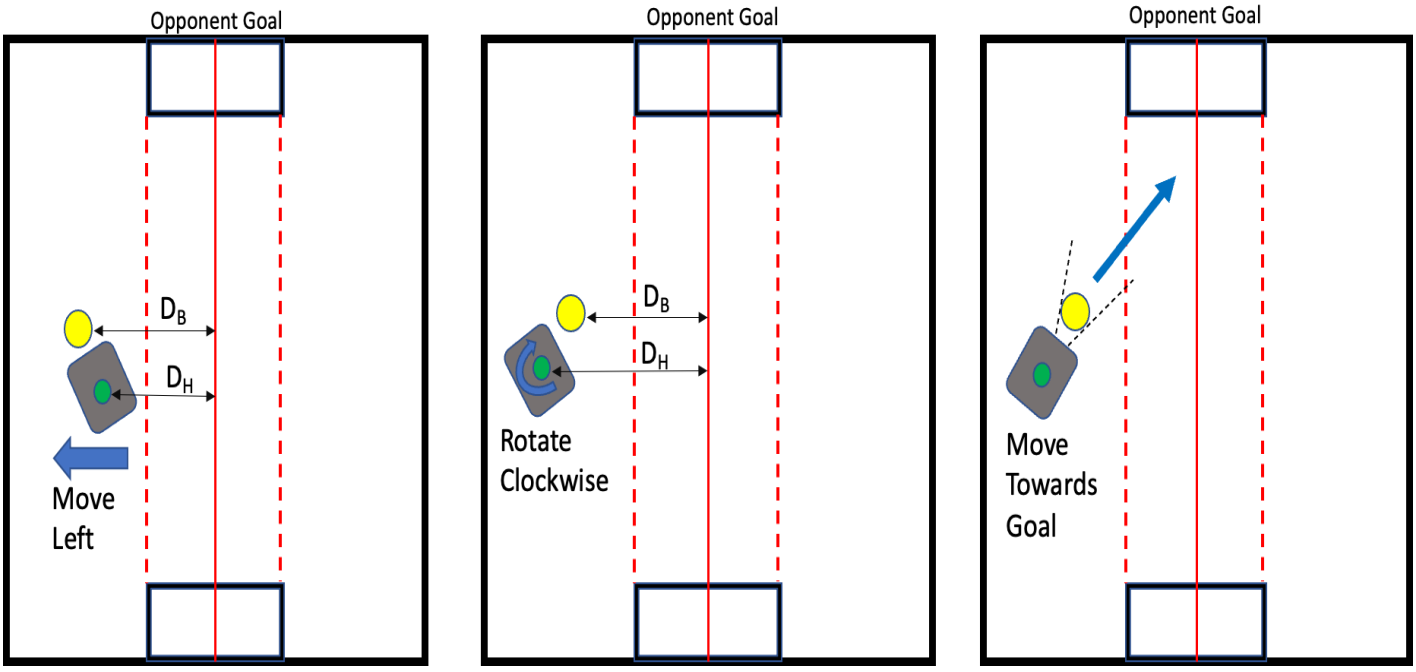
The playground was divided into 6 different zones and different functions were executed when the ball and the robot were in these regions. The main purpose of doing this was to make sure the robot aligns itself towards the opponent goal after reaching near the ball. The robot would calculate its distance from the center line (shown in blue color in the figure below) and distance of the ball from the center and then position itself in various zones to execute commands specific to that region.




The robot receives it location received from the radio module and compares the relative distance of robot and the ball from the centre line and then executes a specific set of commands depending its location in the zone as discussed below

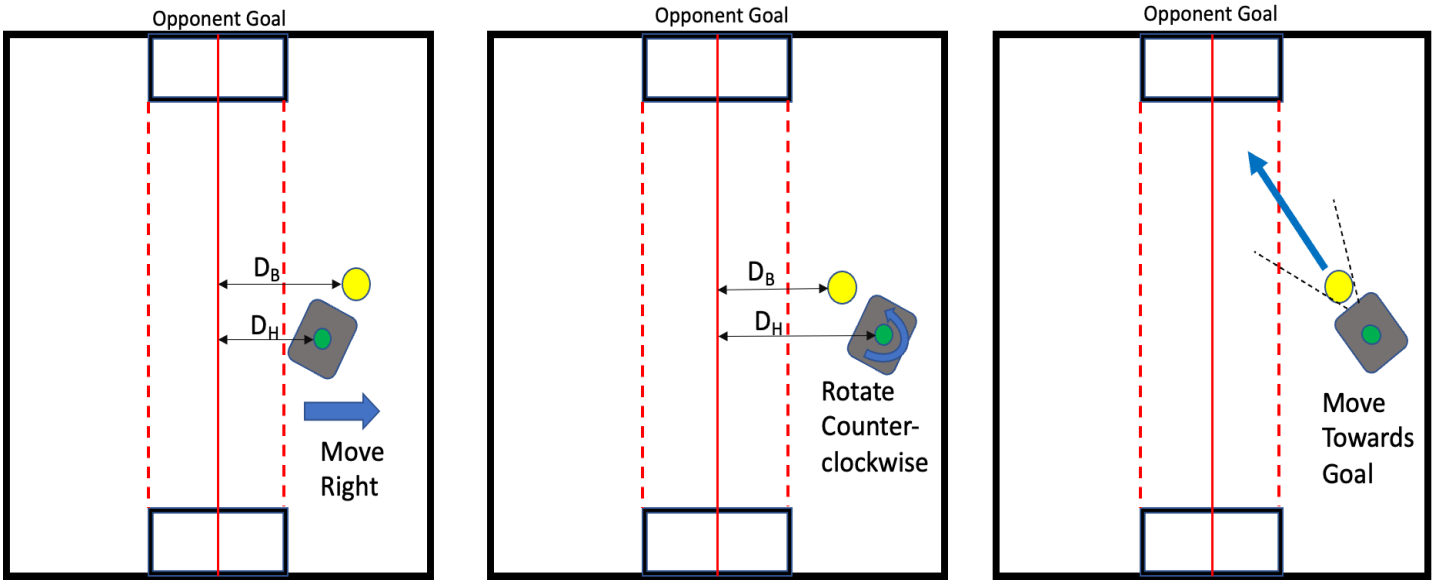
When the robot is in zone 3 and zone 4 the robot would align itself to face the opponent goal using the yaw angle obtained from the IMU module and then move towards the opponent goal with the ball. If the robot is facing our goal then the robot would move diagonally right or left depending on which zone it is in, and then rotate clockwise to face the ball again. This diagonal motion would help align the robot towards the opponent's goal.

If the robot is in Zone 5 and Zone 6, then the robot would first compare its distance and ball's distance to the center. If the distance of ball ($D_B$) is closer to center than the robot's distance ($D_H$) then the robot would directly move towards the goal maintaining a specific yaw angle which was tuned in the playground. For the case when the ball is farther away from the center as compared to the robot, the robot would move left and then rotate clockwise. This was done in order to align the robot towards the goal.



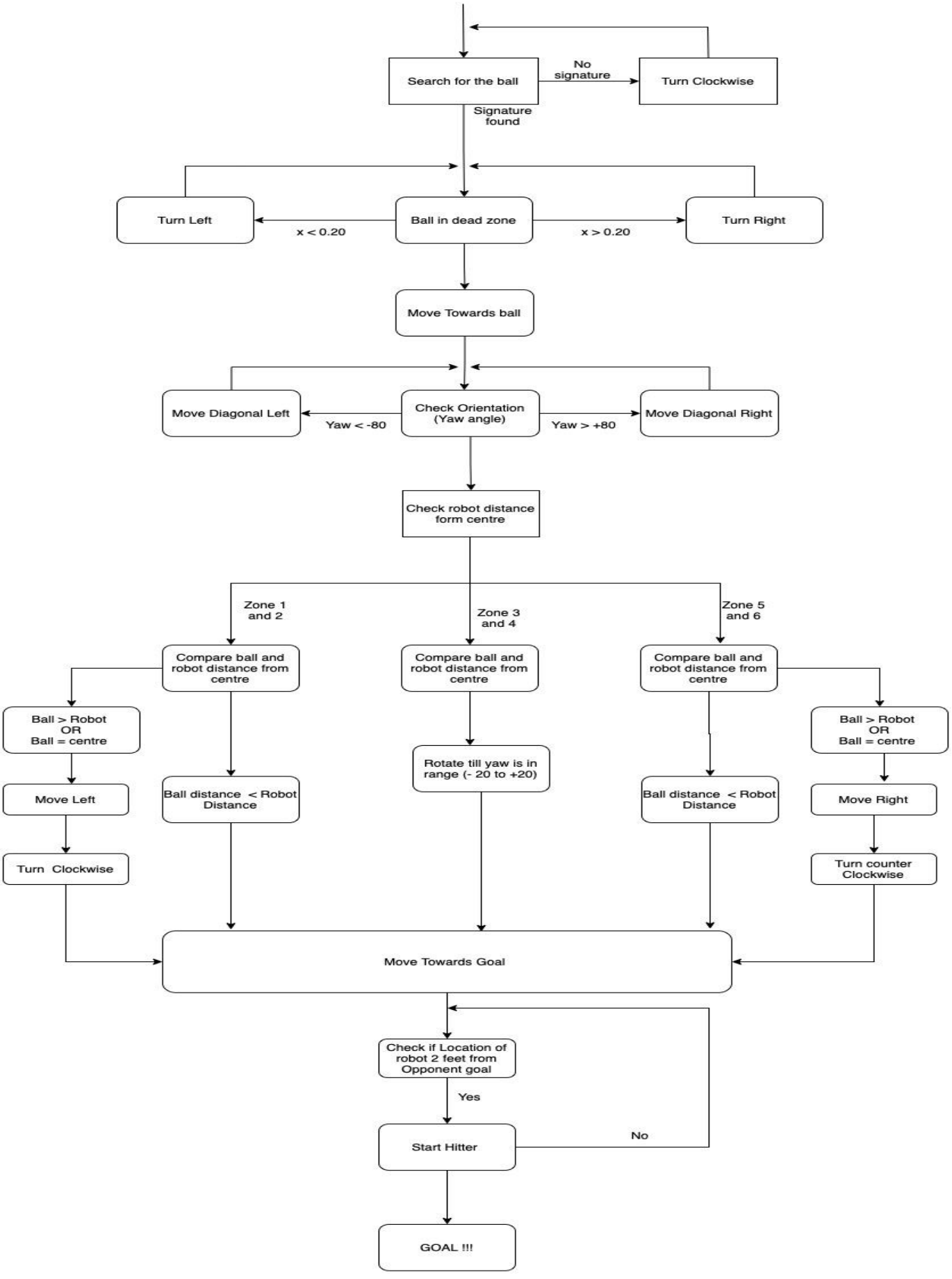

a) If the Robot distance ($D_H$) is greater than ball distance ($D_B$), the robot would move left.
b) The robot moves clockwise until a specific yaw angle to align itself towards the goal.
c) The robot aligned itself towards the goal and would move forward with the ball.

If the robot was in Zone 1 and Zone 2, then the robot would first compare its distance and ball's distance to the center. If the distance of ball ($D_B$) is closer to center than the robot's distance ($D_H$) then the robot would directly move towards the goal maintaining a specific yaw angle which was tuned in the playground. For the case when the ball is farther away from the center as compared to the robot, the robot would move right and then rotate in counter-clockwise direction. This was done in order to align the robot towards the goal.




a) If the Robot distance ($D_H$) is greater than ball distance ($D_B$), the robot would move right.
b) The robot moves in counter-clockwise direction till a specific yaw angle to align itself towards the goal.
c) The robot aligned itself towards the goal and would move forward with the ball.

The robot is programmed to move forward till it reaches the opponent goal, however when the opponent goal is 2 feet away from the robot it would stop and use the hitter by powering the hitter motor. The hitter uses a spring loaded mechanism to shoot the ball into the goal.


Search for the ball
No signature
Turn Clockwise
Signature found
Turn Left
x < 0.20
Ball in dead zone
x > 0.20
Turn Right
Move Towards ball
Move Diagonal Left
Yaw < -80
Check Orientation (Yaw angle)
Yaw > +80
Move Diagonal Right
Check robot distance form centre
Zone 1 and 2
Zone 3 and 4
Zone 5 and 6
Compare ball and robot distance from centre
Compare ball and robot distance from centre
Compare ball and robot distance from centre
Ball > Robot OR Ball = centre
Move Left
Turn Clockwise
Ball distance < Robot Distance
Rotate till yaw is in range (- 20 to +20)
Ball distance < Robot Distance
Ball > Robot OR Ball = centre
Move Right
Turn counter Clockwise
Move Towards Goal
Check if Location of robot 2 feet from Opponent goal
Yes
No
Start Hitter
GOAL !!!

## 6. Real-time problem

The robot was detecting multiple yellow signatures during the competition which confused the robot and made it move towards the wrong direction. This happened because the pixy camera was programmed to detect yellow balls in a very low light setting and the playground had very bright lights, which resulted in even mildly yellow or even orange colors to be detected as yellow. Even though we used raspberry pi for computation, the ball and robot data was sent to the pi by arduino which became bottle neck in the system and dictated the speed of the robot motion. This huge difference in the processing speeds at which the directions were sent to the arduino and executed by the arduino caused delay in the robot motion. This problem was corrected by adding a delay while moving the robot clockwise, so that it gets time to receive directions from raspberry pi and implement them.

## 7. Budget

| Sl. No. | Purchased Source | Purchased Product | Cost (In Total) |
|---|---|---|---|
| 1 | AMAZON | MOTORS, MOTOR MOUNTS & WHEELS | $101.78 |
| 2 | DIGIKEY | STANDOFFS (3.5" HEIGHT) | $27.99 |
| 3 | HOME DEPOT | MDF SHEETS | $16.00 |
| 4 | HOME DEPOT | SCREWS | $11.00 |
| 5 | HOME DEPOT | WIRES | $5.00 |
| 6 | MCGUCKINS | SCREWS & NUTS | $10.63 |
| 7 | MCGUCKINS | SPRINGS | $4.00 |
| 8 | MCGUCKINS | SCREWS | $3.72 |
| 9 | Amazon | Prototyping board | $2.00 |
| 10 | Amazon | Switches | $1.00 |
| 11 | Amazon | Ribbon cable | $3.00 |
| | | Total | $186.12 |

## 8. Lessons Learned

In the end, we were able to reach second place in the competition and even knocked the 1st place team out of the winners bracket for a true final match. We are very happy with our result as we did not expect to compete as well as we did.

The biggest thing that I would recommend would be to stay with a breadboard. While the benefits of a protoboard or a PCB are definitely tangible, it is an extremely time consuming and laborious process to solder every wire into place. If you are sure you know what your electronics will look like 2-3 weeks before the competition, the time commitment could be worth it. But if you

haven’t finalized your electronics a week before your competition, stick with the breadboard and noodle soup.

## 10. Appendix

Raspberry Pi Code.

https://drive.google.com/file/d/1vd1F6ApXFalGjS5A6IoL_zjtJuZJBjG4/view?usp=sharing

Arduino_1 Code:

https://drive.google.com/file/d/1zKG8KihaJriNoUgMnKsTIb68RGVNzh-O/view?usp=sharing

Arduino_2 Code:

https://drive.google.com/file/d/1dXVNaP2NyNqdkwjK6kajakZQLNAZJ2U6/view?usp=sharing