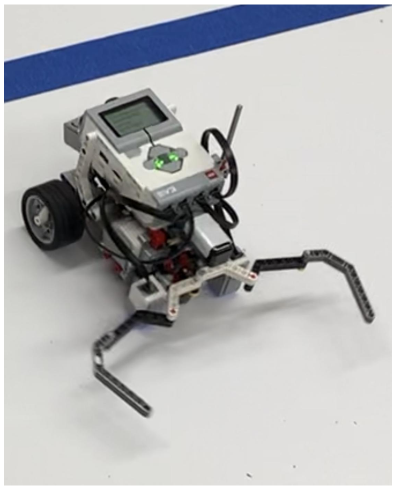
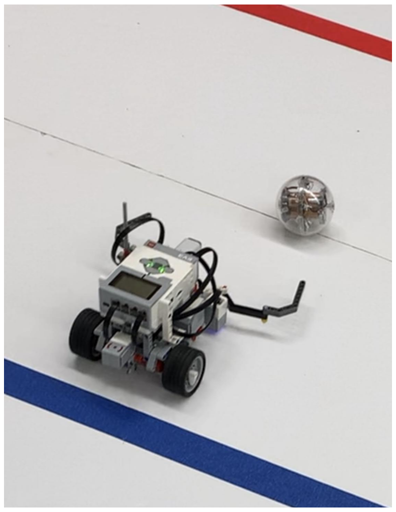
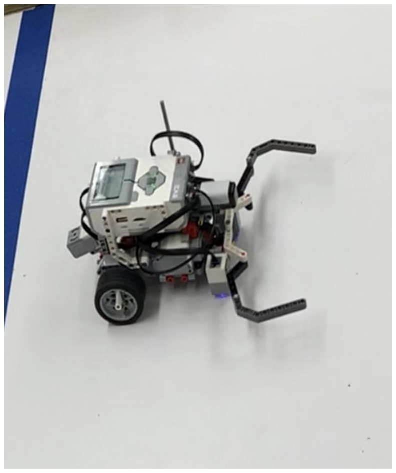
<u>Final Project Report:</u>
CSE-4360-001 Autonomous ROBOT Design
Simplified Soccer Project

Team 15:
Amanda Whisenand
Vivek Kumar Yadav
Gabriel De Sa

Robot Description

Our robot was built using the Lego Mindstorm EV3 kit. We went with a simple tricycle design which consists of a small base and two wheels. Each wheel has a low profile tire that is 56x28 mm and is connected to a large motor. Each large motor is connected to the main brick. The model balances itself with an omni directional ball in the rear. We went with these design choices because the unicycle type robots are relatively easy to control compared to other possible design choices.

In addition, the robot has two color sensors. One on the front right and one on the front left. These sensors are placed less than one inch above the ground and are secured in place with zero degrees of freedom. The light sensors are used to detect the defense lines and attack and defend accordingly. On the front top of the robot, we have placed an infrared sensor. This is used to detect the incoming infrared from the ball to align itself and shoot to score if it is in its shooting zone. Also, the robot has a gyro sensor that helps to keep track of the direction so that the robot knows which direction to shoot and defend accordingly.

Figures 1, 2, and 3. Views of robot from different angles

Control System

The robot plays a simplified version of soccer against another team using an IR ball and IR seeker sensors. The align() function starts the robot to keep moving forward until it senses the strong sensor strength from the ball equipped with a set of IR LEDs. The robot starts moving towards the ball after it detects the strength greater than 15 and changes direction accordingly to align itself to ball i.e., in zones[4,5,6]. When the sensor strength is very strong the robot uses a gyro sensor to rotate in the direction of the opponent's defense base and the ballFollow() function is called which keeps aligning and moving forward with the ball and later calls score() function which checks if the gyroSensor.angle() is between -5 and 5 and speeds up the robot to push the ball forward and score. ballFollow() and score() keeps executing until the front color sensors detect the opponent's defense line. After the robot scores and reaches the defense line of the opponent goToHome() is called, which makes the robot rotate in the opposite direction towards its defense zone with the help of gyro sensor and it moves forward looking for its own defense line. When it reaches home it rotates back to face towards the opponent's defense zone and goes into the defensive move.

In the defensive mode, the robot stays calm and continuously keeps looking for the incoming ball with the help of its IR sensor. This behavior is activated by defAlign() function which also tries to catch the incoming ball and then calls ballFollow() to follow along with the ball and eventually calls score() functions to score when the ball is in zone[4, 5, 6], the sensor strength is very high, and the robot is facing in the direction of the opponent's defense zone. All of these behaviors of the robot use the fundamental behavior functions like goToTraget(), resetAngles(), rotate() to move to a location and change directions respectively.

Program :

```python
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 UltrasonicSensor, GyroSensor)
from pybricks.iodevices import I2CDevice
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile
import time
import random

homeZone = Color.BLUE
goalZone = Color.RED

# Create InfraredSensor class
class InfraredSensor():
    def __init__(self, port):
        self.sensor = I2CDevice(port, 0x01)

    def get_zone(self):
        """Returns Zone that IR signal is observed in."""
        return int.from_bytes(self.sensor.read(0x42, length=1), "little")

    def get_strength(self):
        """Returns an array with the relative strength of IR in each 
zone."""
        retArray = []
        for i in range(5):
            strength = int.from_bytes(self.sensor.read(0x43+i, length=1), 
"little")
            retArray.append(strength)
        return retArray


# This program requires LEGO EV3 MicroPython v2.0 or higher.
# Click "Open user guide" on the EV3 extension tab for more information.
wheelCirc = (5.3975*3.14159)/100 # meters
robotCirc = (12.065*3.14159)/100 # meters
wallColor = 10
ev3 = EV3Brick()

def argmax(arr):
    return arr.index(max(arr))

def goToTarget(right, left, target, velocity = 360):
    resetAngles(right, left)
    desired_angle = (target/wheelCirc)*360
    right.run_angle(velocity, desired_angle, Stop.HOLD, False)
    left.run_angle(velocity, desired_angle)

def resetAngles(right, left, angle = 0):
    right.reset_angle(angle)
    left.reset_angle(angle)

def rotate(right, left, angle, gyroSensor, velocity = 150):
    while gyroSensor.angle() != angle:
        if(gyroSensor.angle() < angle):
            # if its less than 0 turn right
            right.run(-velocity)
            left.run(velocity)
```

```python
        else:
            # if its more than 0 turn left
            right.run(velocity)
            left.run(-velocity)
    else:
        right.stop()
        left.stop()
        return

def goToHome(colorSensor, gyroSensor, right, left):
    ev3.screen.print("Going Home")
    goToTarget(right, left, -.1)
    rotate(right, left, 180, gyroSensor)
    time.sleep(.25)
    while colorSensor[0].color() != homeZone:
        right.run(500)
        left.run(500)
    else:
        right.stop()
        left.stop()
        goToTarget(right, left, .3)
        rotate(right, left, 0, gyroSensor)
        gyroSensor.reset_angle(0)
        return

def score(gyroSensor, colorSensor, irSensor, right, left):
    ev3.screen.print("scoring")
    while(gyroSensor.angle() < -5 or gyroSensor.angle() > 5):
        if(gyroSensor.angle() < 0):
            # if its less than 0 turn right
            right.run(-160)
            left.run(160)
        else:
            # if its more than 0 turn left
            right.run(160)
            left.run(-160)
    else:
        right.stop()
        left.stop()
        while colorSensor[0].color() != goalZone and
colorSensor[1].color() != goalZone:
            right.run(1000)
            left.run(1000)
            if max(irSensor.get_strength()) < 10:
                right.stop()
                left.stop()
                return False
        else:
            right.stop()
            left.stop()
            goToHome(colorSensor, gyroSensor, right, left)
            return True


def align(irSensor, colorSensors, right, left):
    ev3.screen.print("Aligning")
    zones = [4, 5, 6]
    while max(irSensor.get_strength()) < 15:
        right.run(500)
        left.run(500)
        if colorSensors[0].color() == goalZone or colorSensors[1].color()
== goalZone:
            right.stop()
            left.stop()
            return False
        elif irSensor.get_zone() not in zones and
max(irSensor.get_strength()) > 2:
            if(argmax(irSensor.get_strength()) < 2):
                 # if its less than 5 its on its left
                while (argmax(irSensor.get_strength()) != 2):
                    right.run(500)
                    left.run(-500)
                else:
                    right.stop()
                    left.stop()
            else:
                # if its more than 5 its on the right
                while (argmax(irSensor.get_strength()) != 2):
                    right.run(-500)
                    left.run(500)
                else:
                    right.stop()
                    left.stop()
            continue
    else:
        return True

def defAlign(irSensor, colorSensors, right, left):
    ev3.screen.print("Defensive Aligning")
    zones = [4, 5, 6]
    while True:
        if irSensor.get_zone() not in zones and
max(irSensor.get_strength()) > 5:
            if(irSensor.get_zone() < 5) or
(argmax(irSensor.get_strength()) < 2):
                # if its less than 5 its on its left
                right.run(500)
                left.run(-500)
            elif (irSensor.get_zone() > 5) or
(argmax(irSensor.get_strength()) > 2):
                # if its more than 5 its on the right
                right.run(-500)
                left.run(500)
            elif (irSensor.get_zone() == 5) or
(argmax(irSensor.get_strength()) == 2):
                right.stop()
                left.stop()
                return True
        else:
            while max(irSensor.get_strength()) < 10 and
irSensor.get_zone() in zones:
                if colorSensors[0].color() == goalZone or
colorSensors[1].color() == goalZone:
                    right.stop()
                    left.stop()
                    return False
                else:
                    right.run(500)
                    left.run(500)
            else:
                if irSensor.get_zone() in zones:
                    right.stop()
                    left.stop()
                    return True
                else:
                    right.stop()
                    left.stop()
                    return False

def ballFollow(irSensor, colorSensor, gyroSensor, right, left):
    ev3.screen.print("Following Ball")
    strengthZones = [1, 2, 3]
    zones = [4, 5, 6]
    while (argmax(irSensor.get_strength()) in strengthZones) and
(irSensor.get_zone() in zones) and (max(irSensor.get_strength()) < 30 and
max(irSensor.get_strength()) > 5):
        if(colorSensor[0].color() == goalZone or colorSensor[1].color() ==
goalZone):
            goToHome(colorSensor, gyroSensor, right, left)
            return False
        right.run(500)
        left.run(500)
    else:
        if irSensor.get_zone() in zones:
            right.stop()
            left.stop()
            return True
        else:
            right.stop()
            left.stop()
            return False


# Create your objects here.
def main():
    rightMotor = Motor(Port.A, Direction.COUNTERCLOCKWISE)
    leftMotor = Motor(Port.D, Direction.COUNTERCLOCKWISE)
    colorSensor = ColorSensor(Port.S3)
    leftColorSensor = ColorSensor(Port.S4)
    gyroSensor = GyroSensor(Port.S2)
    irSensor = InfraredSensor(Port.S1)
    # # Write your program here.
    ev3.speaker.beep()
    gyroSensor.reset_angle(0)
    if align(irSensor, (colorSensor, leftColorSensor), rightMotor,
leftMotor):
        if ballFollow(irSensor, (colorSensor, leftColorSensor),
gyroSensor, rightMotor, leftMotor):
            score(gyroSensor, (colorSensor, leftColorSensor), irSensor,
rightMotor, leftMotor)
    else:
        goToHome((colorSensor, leftColorSensor), gyroSensor, rightMotor,
leftMotor)
    while True:
        if defAlign(irSensor, (colorSensor, leftColorSensor), rightMotor,
leftMotor):
            if ballFollow(irSensor, (colorSensor, leftColorSensor),
gyroSensor, rightMotor, leftMotor):
                score(gyroSensor, (colorSensor, leftColorSensor),
irSensor, rightMotor, leftMotor)


main()
```