EN.530.421 Mechatronics, Section 1

Final Project - Team Ice Sk8er Boi

5/16/2022

Report Submitted By:

Lisa Chizmadia, Nyeli Kratz, Daniel Okereke, Jerry Zhang

Overall Objectives and Design Requirements

The objective of this project was to create two robots which autonomously win a game of JHockey. JHockey is won by shooting the puck into the opponent's goal more times than the opponent over the span of a 4-minute game with one 90-second timeout for each team. The start of a game is signaled to each robot over Radio Frequency (RF) as well as the position of the puck throughout the game, as read by an overhead Pixy camera. Each goal is centered on the width of the arena and colored with the corresponding defending team color.

The robots can have a frame perimeter of no greater than 80cm and a height of no greater than 20cm. The robot cannot block more than half of the goal at any time. All robots will wear a jersey secured to the top of the robot indicating their team color.

Overall Strategy

The team decided to make one robot specifically for defense and another robot specifically for offense in order to avoid the two robots interfering with each other throughout the game. The team members split into defense and offense bot teams with one member primarily responsible for hardware and the other member primarily responsible for software on each team.

Offense Bot Strategy

The offense bot was designed to move to the puck as fast as possible, get possession of the puck, and then move towards the goal. At a certain distance from the goal, the robot would shoot the puck.

It was decided on a simple shooting mechanism of just one wheel on top of the puck spinning towards the robot to intake the puck and then spinning away from the robot to shoot the puck. A DC motor with an encoder was chosen so that information could be read about the speed of the wheel from the encoder. When the robot drives the wheel inwards but the speed of the shooting wheel is zero, this indicates that the robot has possession of the puck and thus changes the goal of the robot from obtaining the puck to driving towards the goal. See the shooting mechanism video linked in the appendix.

To track the puck, a Pixy camera was used to find the puck and drive the robot towards the puck. Once the robot had possession of the puck based on the information from the encoder, the wheels then drove the robot towards the goal rather than the puck.

An RF module was used to receive the game start signal. Due to the high number of pins required by the Pixy, the motor driver, and the RF communication, two Arduino Unos were required to control all of the sensors on the robot. One Arduino controlled all three wheels (both drive wheels and the shooting wheel) and the Pixy camera while the other Arduino controlled the RF and read the encoder from the shooting motor.

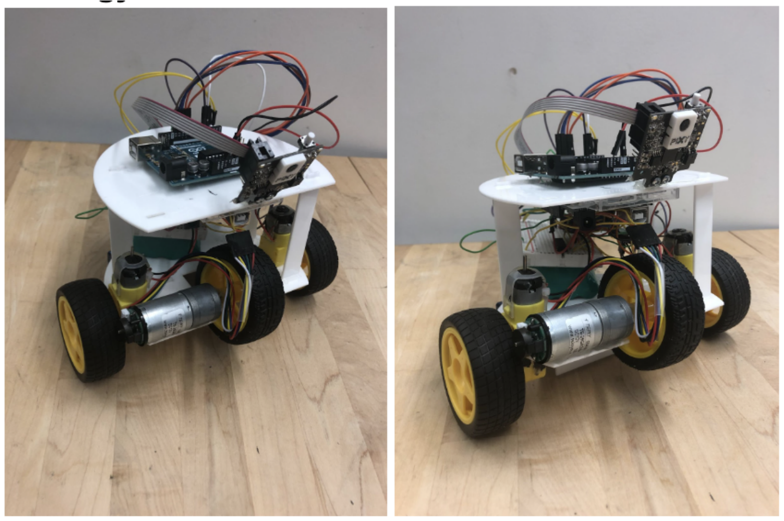


Figure 1. Final offense bot design shown from two angles

Offense Bot Mechanical Design

The first iteration of the offense bot was designed to maximize speed while also providing adequate mounting surfaces. Arduinos and breadboard were placed on the red vertical surface seen in Figure 2. Ultimately, this designed was abandoned based on center of mass issues causing drift and because the pointed shape of the chassis made it too easy for the robot to get trapped against walls.

In the second iteration, the body of the offense bot was designed to maximize speed, have a center of mass between the two drive wheels, and also avoid being pinned against walls by the opposing team. A small footprint helps maximize speed and the circular shape avoids being pinned by any corners. The arc cut off of the circular roof above the chassis ensures that the robot never covers more than half of the puck, in keeping with the game rules. The tab that extends towards the ground underneath the bottom chassis of the offense bot holds the puck between the drive wheels when the robot intakes the puck. Batteries and breadboards were mounted on the lower chassis platform while two arduinos and a Pixy camera were mounted on the top chassis platform. The motors were mounted vertically to allow us to easily adjust the height of the chassis to ensure that the shooting motor was exactly the correct height off the ground to allow the robot to intake and shoot the puck. From the CAD image, one of the rectangular standoffs was removed to allow space for the DC motor and encoder shooting wheel.

An adapter was designed and 3D printed so that the wheel would fit on the shaft of the DC motor with encoder, however the small size difference between the wheel shaft and the motor shaft meant that the team could not print an adequate adapter. Instead, we wrapped the output shaft of the DC motor with encoder in electrical tape to slightly increase its diameter which allowed us to slide on the wheel in a press fit. The entire offense bot chassis was laser cut from acrylic as this is a relatively strong and inexpensive material that the team had experience working with.

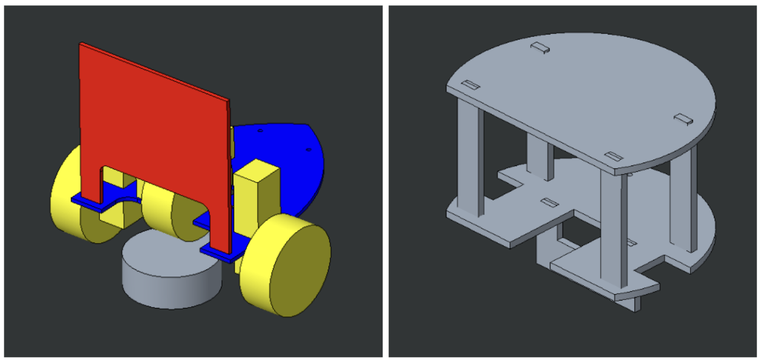


Figure 2. CAD models of offense bot iteration 1 (left) and iteration 2 (right)

Offense Bot Electrical Design

Due to the variety of motors and sensors used, two Arduino Unos were used to ensure we had adequate pins for all the components. Arduino One (SensorDuino) was connected to the RF transceiver and the shooting motor encoder and could read data. Arduino Two (MotorDuino) was connected to two H-bridges and was able to control the movement motors, shooting motor, and PIXY camera. Two H-bridges were needed since we needed to control the movement motors that drove the robot and let it turn left and right and we also needed to change the shooting motor from turning slowly in the intake direction and then suddenly to rapidly spinning in the opposite direction to shoot the puck towards the goal.

Offense Bot Software

The logic for the robot was split between two Arduino Unos, SernsorDuino and MotorDuino. Initially, the team planned to have the Arduinos communicate with each other via Serial communication, but switched to HIGH/LOW over digital pins since it was easier to implement and mitigated string corruption or other memory issues. Additionally, the signals which we needed to send between the two Arduinos were simple enough to be represented in this binary fashion.

<u>SensorDuino</u>

During testing, it was determined that the RF transceiver may not always accurately detect the GAME START signal, but could reliably receive coordinates. Thus, an if statement was used to check if any RF signal was received and it was not a question mark. If so, the motorDuino was instructed to move. Otherwise, the offense bot remained in this “stop” state.

Originally, it was planned to have the shooting motor constantly spinning slowly towards the robot such that it would stop rotating once it firmly held the puck. This Arduino read the output of the shooting motor’s encoder, so it could tell when it stopped rotating which indicated puck possession. Then, it would switch to a GO_FOR_GOAL state where we look use the Pixy to find the opponent’s goal. Unfortunately, the issues with Serial communication prevented the team from fully implementing this feature, so the robot simply had the shooting motor, controlled by motorDuino, constantly spinning outwards.

<u>MotorDuino</u>

It begins by checking if a stop signal is received on the digital pin from sensorDuino. Otherwise, it begins moving forwards and searching for the puck. The shooting rotates towards the robot by default such that it is always to acquire and hold the puck.

Utilizing color tracking code from the maze project, it was simple to implement this puck tracking, especially since the camera and motors were controlled by the same Arduino. If the puck was not seen, then it would alternate between going forward and making right turns until the puck became visible. Once the puck was acquired, the robot would search for the goal’s color. Once the goal was seen, then the shooting motor’s rotation would reverse direction thus shooting the puck.

After seeing the size of the arena and discovering that the Pixy could recognize the puck from a considerable distance, it was decided that it was not worth the effort of implementing tracking via RF coordinates after struggling to efficiently implement a method for determining which direction the robot was heading such that it could always turn to face the puck. However, the team was concerned about situations where another robot was blocking the puck from our line-of-sight which would prevent us from seeing it and potentially spin in circles. Therefore, we had the robot go forward in addition to turning such that it would eventually find the desired object.

Defense Bot Strategy

To create a goalie that would prevent the other team from scoring, we decided the robot only needed to move left and right in front of the goal rather than around the play area in a plane. The team also decided that it did not need to track the other team, but rather track only the puck. This meant that while the RF receiver was used to begin the game, that was the only use of the RF receiver in this particular robot. Furthermore, as part of the design of the robot, it was decided that there would be tank tracks that would push the puck away from the goal if an opponent pushed the puck into the goalie. This design can be seen better in Figure 3.

In order to achieve all of these goals set forth, the sensors that the robot needed were then picked out. The Pixy camera was used to track the puck and inform the robot which side of the goal to defend.

Since the robot only moved back and forth, the team wanted to create a design that the robot could see to the left and right. It was then decided that a servo motor would be utilized to assist the Pixy tracking. While the Pixy didn’t detect anything, the servo would pan in case the puck was out of frame to the left or right of the goal. When the puck was detected by the Pixy, the servo would stop panning and the robot would move in the direction that the servo was facing, also taking into account where the puck was with respect to the Pixy frame.

Defense Bot Mechanical Design

Because the robot only had to move in one direction, a single DC motor with two wheels on each end of the motor shaft was used. For balance, three fixed caster wheels were used and it was ensured that the motorized wheels stuck out slightly further than the caster wheels so that they would always have traction with the ground.

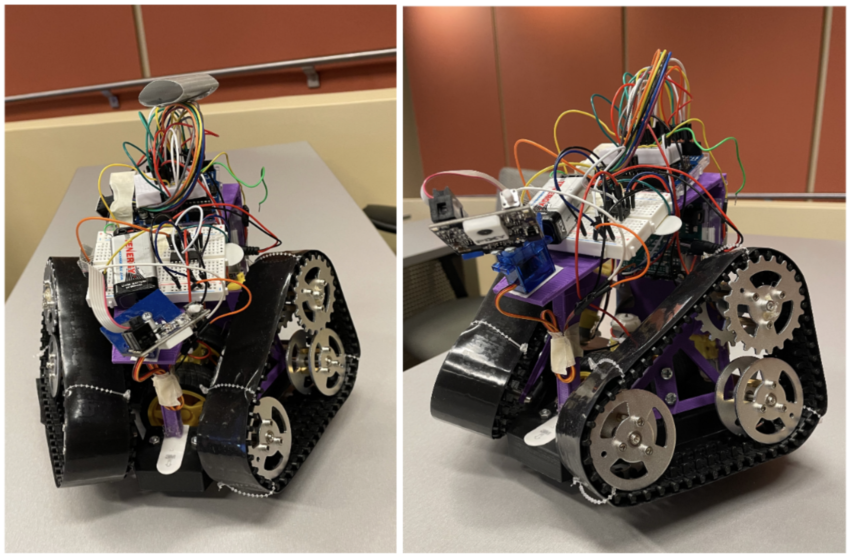


Figure 3. Final defense bot design from two angles

For the active defense mechanism, we scavenged tank tracks and wheels and designed and 3D printed a mount that could hold the wheels and let one wheel be powered by a DC motor. Since we need a tank mechanism for both sides, two tank mechanisms were made, with mirrored designs, and mounted to the sides of the robot's base. They were also angled 20 degrees up to enable the puck to be caught underneath. We found that the plastic tank treads had issues with grip onto the puck and covered it with silicone tape and zip ties, which greatly improved grip, as shown in the appendix video.

Since the motor and casters had to be mounted at specific heights and the tank mechanism needed to be mounted at specific angles, it was decided that the design should be 3D printed in PLA instead of cut from acrylic. The electronics were mounted on the sloped roof of the robot, with the panning Pixy camera at the front facing edge.

Defense Bot Electrical Design

There were three DC motors to control, but since the two DC motors powering the tank tracks did not need speed control and could spin at a constant velocity to push the puck away, we only needed one H-bridge to control the DC motor that moved the robot left and right. The other two DC motors were directly powered from the Vin pin of an Arduino. We connected the servo motor with the Pixy Camera directly to an Arduino and the RF module to another. The servo motor with its known position was helpful in knowing where the Pixy camera was facing when it sees the puck, so that the robot can know if it should move in the right or left direction. This change in movement was aided by the use of an H-bridge.

Defense Bot Software

The code for this robot was written using two Arduinos and two sources of power. This was mainly due to the higher power needed to power the tank drivers, as well as the fact that the RF receiver had many wires and could not be placed on the same Arduino as the motors and Pixy cam. Therefore, the first Arduino (Arduino1) had the tank motors and the RF receiver.

The second Arduino (Arduino2) consisted of the two wheels under the robot to move in front of the goal, the servo motor, and the Pixy cam.

The Arduino1 is what controls the start of the game for the defense bot. The code begins with the robot doing nothing until it receives the start signal. This was achieved by connecting the two Arduinos and outputting a "HIGH" signal to a digital pin on the second Arduino. Therefore, without the signal, everything is "LOW". Similar to the offense bot, the team was worried that it may miss te start signal. So, instead of looking for the start signal, it was coded such that the the cue to start was if RF signal was received and it was not a question mark. (The question marks indicated stop and were continuously played when the game was not occurring). Therefore, even if the start command was missed, when coordinates were sent , this would still trigger the robt to begin gameplay.

When the signal is read, Arduino1 then powers the tank motors which will spin continuously throughout the game.It also sends the signal to Arduino2 which has the motors for the wheels that move the robot left and right, the servo motor, and the PIXY cam.

Arduino2 is what controls the primary functions of the goalie, so once this happens, the first thing that occurs is the servo motor starts panning. The team decided to hace a panning serco to maximize the area that the Pixy cam would be able to see. The Pixy cam, mounted on the servo, begins searching for the puck in frame. This was achieved using a similar code that the team previously used to track colors in the labyrinth lab. When the puck is seen by the Pixy cam, the servo then stops panning and moves according to where the puck is with relation to the servo motor and the Pixy cam frame. An example of this is, say the angle of the servo motor is less than 90 degrees but it is picked up to the left side of the Pixy cam. This means that the puck is to the right of the goal but not all the way to the right. When this happens, the wheels move to the right, but the servo moves 10 degrees to the left to try to center the puck in the frame of the Pixy cam.

The code for the defense bot can be seen in the Appendix of this report.

Testing Results

The team designed and built two JHockey robots over the 6 weeks spent on this project. The offense bot was able to track and drive towards the puck, intake the puck, and make a shot on the goal. The defense bot was able to track the puck and block shots. However, the robots did not perform well in the final JHockey tournament. There were some technical difficulties in power management with the PIxy cameras and the other sensors and motors. We troubleshooted the camera and cables before switching to fresh batteries, which solved the issue. Furthermore, one member of the offense bot team, unfortunately, got COVID 5 days before the competition which made it very difficult to get the offense bot working in time for the competition. We had a faulty H-bridge that prevented proper intake and shooting, which we did not realize until after the competition. Once we swapped out the H-bridge, our shooting worked well and links to a video along with puck tracking and goal defense are in the appendix.

Further Improvement

If the team was to do this project again, a good goal would be to lock in hardware earlier and spend more time testing software on the completed system. It was a fun, yet difficult, experience trying to figure out where sensors should go, on a fluid design and on the flip side, creating a design where there was not a set space for the sensors. Because there were so many options on the design of the robot, and much more freedom, the team spent a lot of time trying to lock down what would work the best. The tank mechanism on the defense bot was especially difficult to iterate, as the tracks had to have the right tolerances and sometimes jammed. Ultimately, the team was happy with the simple but functional shooting mechanism and puck tracking mechanism.

References
None

Appendix

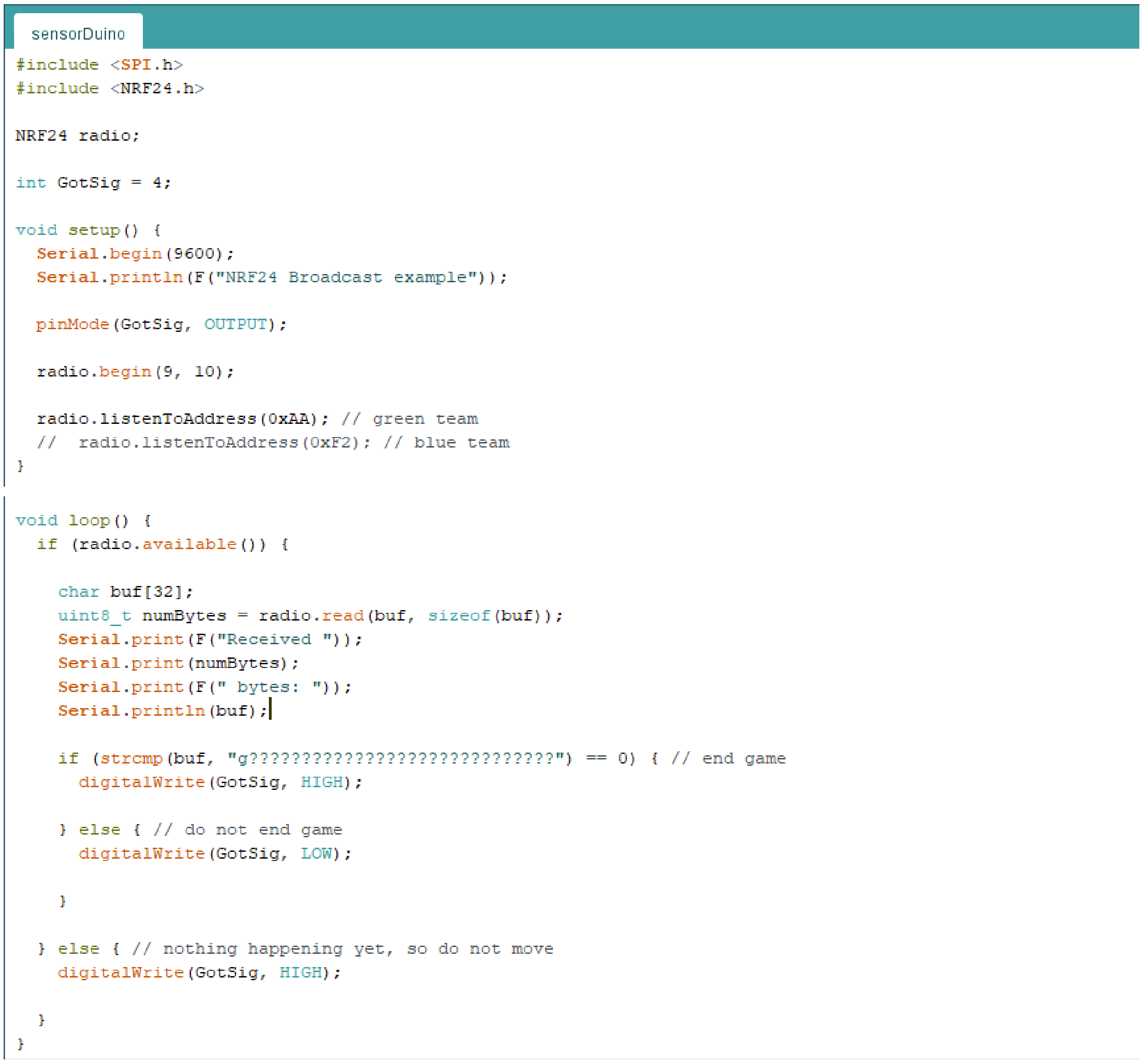
Materials List

Offense Bot Materials	Defense Bot Materials
- Acrylic - 2 DC motors - 1 DC motor with Encoder - 1 DRV8835 Motor Shield - 1 RF module - 1 Pixy Camera - 2 Arduino Unos - 1 LD239 H-bridge - 1 Bread board - Misc. jumper wires - 1 LiPo Battery	- Makerbot PLA - 3 DC motors - 1 Servo Motor - 3 Caster Wheels - 1 RF module - 1 Pixy Camera - 2 Arduino Unos - 1 LD239 H-bridge - 1 Bread board - Tank tracks and wheels - Silicone tape - Misc. jumper wires - 1 LiPo Battery - 1 9V Battery

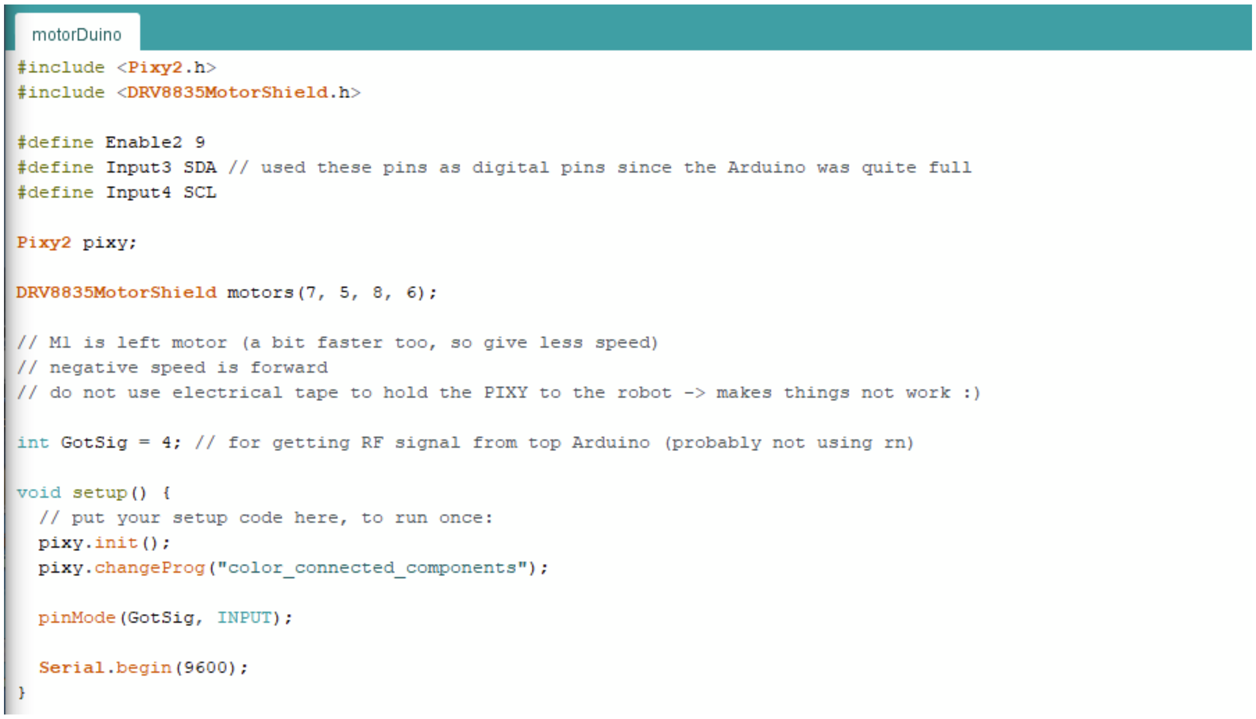
Budget Spreadsheet
Link to Google Sheet:
https://docs.google.com/spreadsheets/d/15vq8h1GKtUE2xzz9Y82bBGwlh5PBpRLTyGYesW-orlg/edit?usp=sharing

Offense Bot
Video for offense bot:
https://youtu.be/FXEZgtutSBY

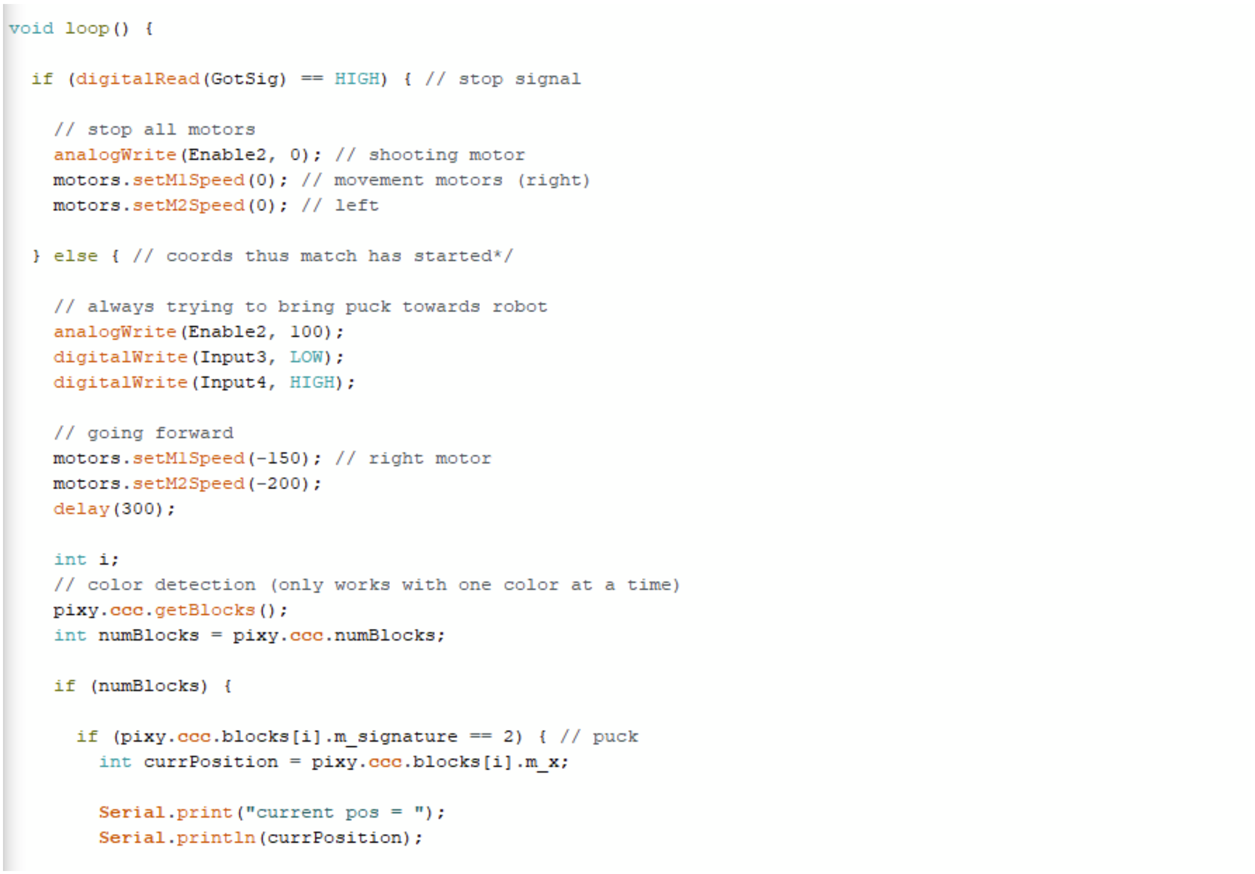
Code for offense bot:



```
#include <SPI.h>
#include <NRF24.h>

NRF24 radio;

int GotSig = 4;

void setup() {
  Serial.begin(9600);
  Serial.println(F("NRF24 Broadcast example"));

  pinMode(GotSig, OUTPUT);

  radio.begin(9, 10);

  radio.listenToAddress(0xAA); // green team
  //  radio.listenToAddress(0xF2); // blue team
}

void loop() {
  if (radio.available()) {

    char buf[32];
    uint8_t numBytes = radio.read(buf, sizeof(buf));
    Serial.print(F("Received "));
    Serial.print(numBytes);
    Serial.print(F(" bytes: "));
    Serial.println(buf);

    if (strcmp(buf, "g?????????????????????????????") == 0) { // end game
      digitalWrite(GotSig, HIGH);

    } else { // do not end game
      digitalWrite(GotSig, LOW);

    }

  } else { // nothing happening yet, so do not move
    digitalWrite(GotSig, HIGH);

  }
}
```



```
#include <Pixy2.h>
#include <DRV8835MotorShield.h>

#define Enable2 9
#define Input3 SDA // used these pins as digital pins since the Arduino was quite full
#define Input4 SCL

Pixy2 pixy;

DRV8835MotorShield motors(7, 5, 8, 6);

// M1 is left motor (a bit faster too, so give less speed)
// negative speed is forward
// do not use electrical tape to hold the PIXY to the robot -> makes things not work :)

int GotSig = 4; // for getting RF signal from top Arduino (probably not using rn)

void setup() {
  // put your setup code here, to run once:
  pixy.init();
  pixy.changeProg("color_connected_components");

  pinMode(GotSig, INPUT);

  Serial.begin(9600);
}
```



```
void loop() {

  if (digitalRead(GotSig) == HIGH) { // stop signal

    // stop all motors
    analogWrite(Enable2, 0); // shooting motor
    motors.setM1Speed(0); // movement motors (right)
    motors.setM2Speed(0); // left

  } else { // coords thus match has started*/

    // always trying to bring puck towards robot
    analogWrite(Enable2, 100);
    digitalWrite(Input3, LOW);
    digitalWrite(Input4, HIGH);

    // going forward
    motors.setM1Speed(-150); // right motor
    motors.setM2Speed(-200);
    delay(300);

    int i;
    // color detection (only works with one color at a time)
    pixy.ccc.getBlocks();
    int numBlocks = pixy.ccc.numBlocks;

    if (numBlocks) {

      if (pixy.ccc.blocks[i].m_signature == 2) { // puck
        int currPosition = pixy.ccc.blocks[i].m_x;

        Serial.print("current pos = ");
        Serial.println(currPosition);
        // y is 0 - 208
        // max camera width was 319 so we divided by 2 -> 160 for center
        // created a little center range so that we go straight without turning
        if (currPosition > 150) { // Left
          Serial.println("Left(200, 200)");

          //motors.setM1Speed(-120); // reversing left motor for left turn
          motors.setM1Speed(70);
          motors.setM2Speed(-200);

          delay(200);
        }
        else if (currPosition < 170) { // Right
          Serial.println("Right(200, 200)");

          motors.setM1Speed(-150);
          motors.setM2Speed(140);

          delay(200);
        }
      } else if (pixy.ccc.blocks[i].m_signature == 3) { // goal

        analogWrite(Enable2, 255); // motor at full speed
        digitalWrite(Input3, HIGH); // shoot puck
        digitalWrite(Input4, LOW);
        delay(1000);

      } else { // just turn to try and find the puck or goal
        motors.setM1Speed(-150);
        motors.setM2Speed(140);
        delay(350);

      }

    }

  }

}
```

Defense Bot

Video for defense bot:
https://youtube.com/shorts/QAvJhtn5Pa0?feature=share

Codes for defense bot:

1. *Code for Arduino containing the RF receiver and the tanks:*

```
#include <SPI.h>
#include <NRF24.h>
#define Enable1 6 // left tank
#define Enable2 3 // right tank
#define Input1 8
#define Input2 7
#define Input3 2
#define Input4 4
#define GotSig 5

NRF24 radio;

bool tx;

void setup()
{
  pinMode(Enable1, OUTPUT); //inputs 3 & 4
  pinMode(Enable2, OUTPUT); //inputs 1 & 2
  pinMode(Input1, OUTPUT);
  pinMode(Input2, OUTPUT);
  pinMode(Input3, OUTPUT);
  pinMode(Input4, OUTPUT);
  pinMode(GotSig, OUTPUT);

  Serial.begin(115200);
  Serial.println(F("NRF24 Broadcast example"));
  radio.begin(9, 10);

  // Pin 7 sets the mode (Sender or Receiver). Connect to GND on the sender
  pinMode(7, INPUT_PULLUP);
  tx = !digitalRead(7);

//listen to address of assigned side
  radio.listenToAddress(0xAA);
  //}

  Serial.print(F("TX mode: "));
  Serial.println(tx);
}
void loop() {

//when it is broadcasting something
  if (radio.available())
  {
    char buf[32];
    uint8_t numBytes = radio.read(buf, sizeof(buf));
    Serial.print(F("Received "));
    Serial.print(numBytes);
    Serial.print(F(" bytes: "));
    Serial.println(buf);
    //if the buf received equals the stop signal, then keep the pin
    //connected to the second arduino low
    if (strcmp(buf, "g?????????????????????????????") == 0) {
      digitalWrite(GotSig, LOW);
      analogWrite(Enable1, 0);
      analogWrite(Enable2, 0);

    }

    //if it is transmitting anything but the stop signal
    //i.e. start signal or coordinates in case it misses
    //start signal
    else if (buf != "g?????????????????????????????") {
      //lets other arduino know the signal has been recieved
      digitalWrite(GotSig, HIGH);
      //starts the tanks
      digitalWrite(Input1, LOW);
      digitalWrite(Input2, HIGH);
      digitalWrite(Input3, LOW);
      digitalWrite(Input4, HIGH);
      analogWrite(Enable1, 125);
      analogWrite(Enable2, 125);
    }
  }
}
```

2. *Code for Arduino containing the two motion wheels, the servo, and the Pixy:*

```
//FINAL CODE FOR DEFENSE BOT
//Mechatronics - final project
//Lisa Chizmadia, Jerry Zhang, Daniel Okereke, Nyeli Kratz

//libraries
#include <Servo.h>
#include <PIDLoop.h>
#include <Pixy2.h>
#include <Pixy2CCC.h>
//PING
#define pingPin 5
//Servo
#define servo 9
//Wheels
#define Enable1 6
#define Input1 8
#define Input2 7
#define GotSig 3

//for motors
void Left(int MotorSpeed) {
  digitalWrite(Input1, HIGH);
  digitalWrite(Input2, LOW);
  analogWrite(Enable1, MotorSpeed);
}
 void Right(int MotorSpeed) {
   digitalWrite(Input1, LOW);
   digitalWrite(Input2, HIGH);
   analogWrite(Enable1, MotorSpeed);
 }

 //for servo
 Servo myservo;
 int pos = 0;
 bool returningToZero;

 //for PING
 long oldcm = 0;
 float duration;
 long cm;
```

```
//for PIXY
Pixy2 pixy;
int currSig = 0;
int currIndex = 0;
int detectedIndex = 0;


void setup() {
  //Serial.begin(9600);
  //motors
  pinMode(Enable1, OUTPUT);
  pinMode(Input1, OUTPUT);
  pinMode(Input2, OUTPUT);
  pinMode(GotSig, INPUT);
  digitalWrite(GotSig, LOW);

  //ping pin
  pinMode(pingPin, INPUT);

  //servo
  myservo.attach(9);

  //pixy
  pixy.init();
  pixy.changeProg("color_connected_components");
}

void loop() {
  //only happens when the signal is sent from the
  //Arduino containing the RF is received
 while (digitalRead(GotSig) == 1) {

    //for PIXY
    int i;
    uint16_t blocks;
```

```
// grab blocks and focus on the most centered block
blocks = pixy.ccc.getBlocks();

//if it can't see the puck, it doesnt move

if (blocks) {
  //centering by tracking the puck with PIXY

  currIndex = pixy.ccc.blocks[i].m_index;
  int currPosition = pixy.ccc.blocks[i].m_x;
  //Serial.print("current pos = ");
  //Serial.print(currPosition);
  //Serial.print("    Current Index = ");
  //Serial.println(currIndex);

  // max camera width was 319 so we divided by 2
  //if servo is facing the left of the goal

  if (pos >= 120) { // Left
    myservo.write(pos - 10);
    Left(200);
    delay(100);
    Left(0);
    //Serial.println("Left");

    //Serial.println(pos);

    //trying to center the puck
    if (currPosition > 160) {
      pos = pos + 10;
      if (pos > 180) {
        pos = 179;
      }
      myservo.write(pos);
    }
    if (currPosition < 160) {
      pos = pos - 10;
      if (pos < 0) {
        pos = 1;
      }
      myservo.write(pos);
    }
  }
```

```
        //same as before but right of the goal
        if (pos <= 100) { // Right
          Right(200);
          delay(100);
          Right(0);
        //Serial.println("Right");
        //Serial.println(pos);

        //trying to center the puck
        if (currPosition > 160) {
          pos = pos + 10;
          if (pos > 180) {
            pos = 179;
          }
          myservo.write(pos);
        }
        if (currPosition < 160) {
          pos = pos - 10;
          if (pos < 0) {
            pos = 1;
          }
          myservo.write(pos);
        }
      }
    }
    else {
      //stop moving
      Right(0);
        //for servo to go back to panning from where it has stopped
        if (pos < 180 && returningToZero == false) {
          pos += 10;
          myservo.write(pos);
          delay(100);
          //Serial.print("Searching for object. Servo's angle: ");
          //Serial.println(pos);
          if (pos == 180 || pos > 180) {
            pos = 180;
            returningToZero = true;
          }
        }
        else if (returningToZero == true) {
          //Serial.println("else if");
          pos -= 10;
          //Serial.println("Searching for object. Servo's angle: ");
        //Serial.println(pos);
        myservo.write(pos);
        delay(100);
        if (pos == 0 || pos < 0) {
          pos = 0;
          returningToZero = false;
        }
      }
    }
  }
}
```