COLOR ROVER

Juan C Garcia-Garcia, Elijah Toussaint

CENG 4900
Capstone Project
5/3/2019

Table of Contents

1. Overview
2. Requirements
   2.1. Rover Chassis
   2.2. The Microcontroller
   2.3. Choosing Color Sensors
   2.4. Assembling the Rover
   2.5. Programming the Rover to Detect Color
   2.6. Connecting the Motors
   2.7. Circuit Board Design
   2.8. Following a Colored Path
3. Materials
   3.1. Circular Rover
   3.2. Rectangular Rover
4. Results & Conclusion
   4.1. Color Tracking Accuracy
   4.2. What can be Changed
5. Future Features
   5.1. Adding Displays
   5.2. Implementing Different Environment Sensors
6. References
   Appendix A
   Appendix B

1. Overview

As technology advances people rely on machines to handle difficult task. One of the most well-known difficult tasks that humanity has endured since the beginning of time, is exploring the unknown. However, sending humans into outer space can be dangerous. Therefore, the goal of the Color Rover project is to design a robot autonomously identify colored paths and react to them based on instructions sent by the user.

2. Requirements

2.1 Rover Chassis

The first challenge was to find a frame that would be able to fit the required parts of the rover. From the multiple chassis' that were found, it came down to two kinds, one that had a round almost circular shape and the other which was more rectangular but still had a rounded front side. The almost circular chassis was chosen because the rover was planned to be compact. This frame allowed for a small body, which allowed for a small turning radius, but had limited space for the chosen microcontroller. The longer chassis, on the other hand, was chosen because of the amount of space it provided. This extra space allowed for a larger battery bank to be used and stored onto the frame without having to force components to be too tight.

2.2 The Microcontroller

Most rovers that were found, used the official Arduino boards, or Arduino clones. While these boards could have also been used, the only ones able to be possibly used were the Arduino Mega, Due, or Leonardo, memory and processor speed limits were the main reasons to look for other microcontroller boards. While there are many different types of boards available, and even custom made Arduino boards that have larger memory, the chosen board was the Teensy 3.6 from PJRC. Compared to the most popular Arduino boards, the Teensy 3.6 offered more functionality. Some of the advantages to using the Teensy 3.6 were: all the analog pins can also be used as digital pins, extra pins underneath the body, much higher processing speed/available memory. After the rover was mostly finished, it was discovered that the Arduino Due would have also been a suitable microcontroller board but due to budget constraints, the rover continued with the Teensy 3.6.

2.3 Choosing the Color Sensor

Only three kinds of color sensors were found that were able to work with the constraints of the rover: TCS3472, TCS3200 and the TCS230. While having the same functionality, they were all slightly different from each other. The TCS3472 used I2C communication and had a full library that could be used to program it easily. The other two sensors were the most similar because of the chip they used and how they were designed. The TCS230 was essentially the same as the TCS3200 except that it had no cover on the color sensing chip, which would have helped to isolate the sensor from external light sources. With this difference, the TCS3200 had more accurate readings and a faster response time. The two best sensors, TCS3472/TCS3200, were chosen because of their accuracy. The TCS3472 was used to detect the edge of the colored path while the TCS3200 was used to detect the actual color of the path.

2.4 Assembling the Rover

Two different designs were built to compare any differences in the rover’s response. The circular chassis used the TCS3472/TCS3200 sensors for path detection, while the more rectangular chassis used two TCS3200 sensors. The circular chassis had almost perfect mounting areas for the three sensors while for the rectangular chassis, two sections had to be drilled off in order to mount the two sensors.

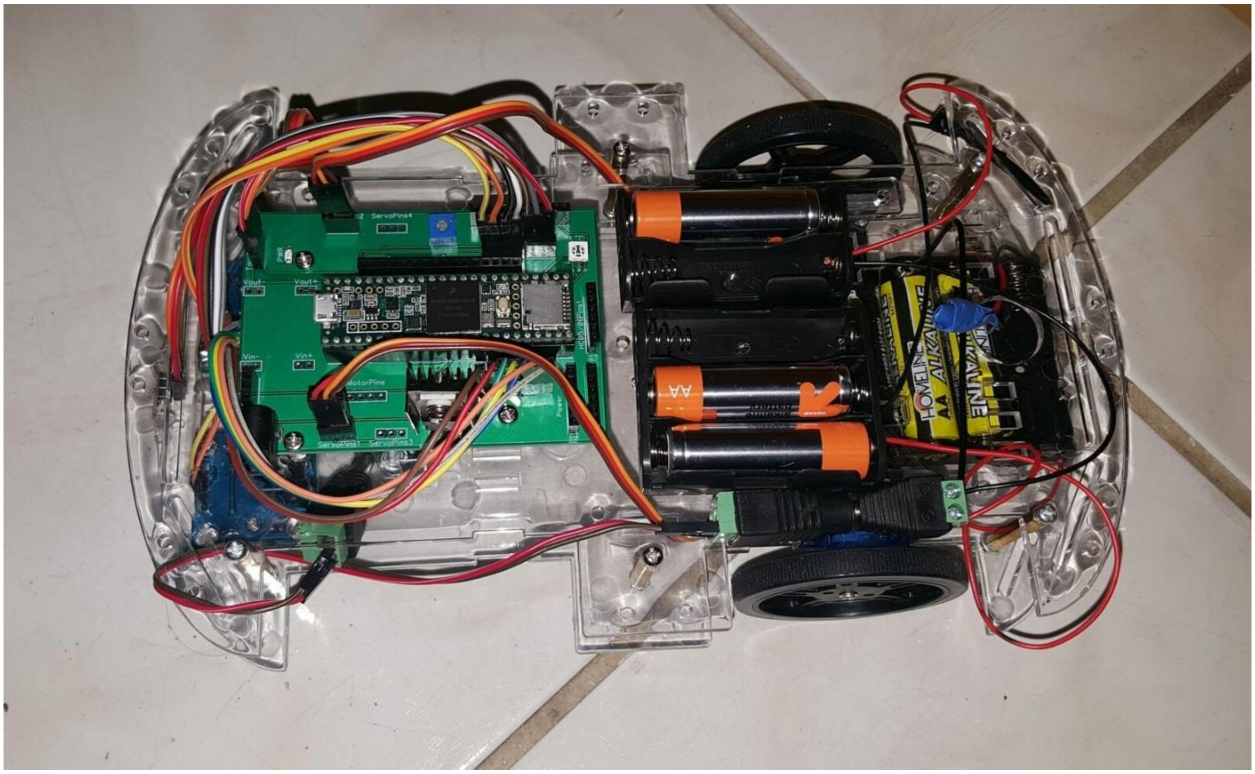


Figure 2.1: Rectangular chassis with TCS3200 sensors on the bottom

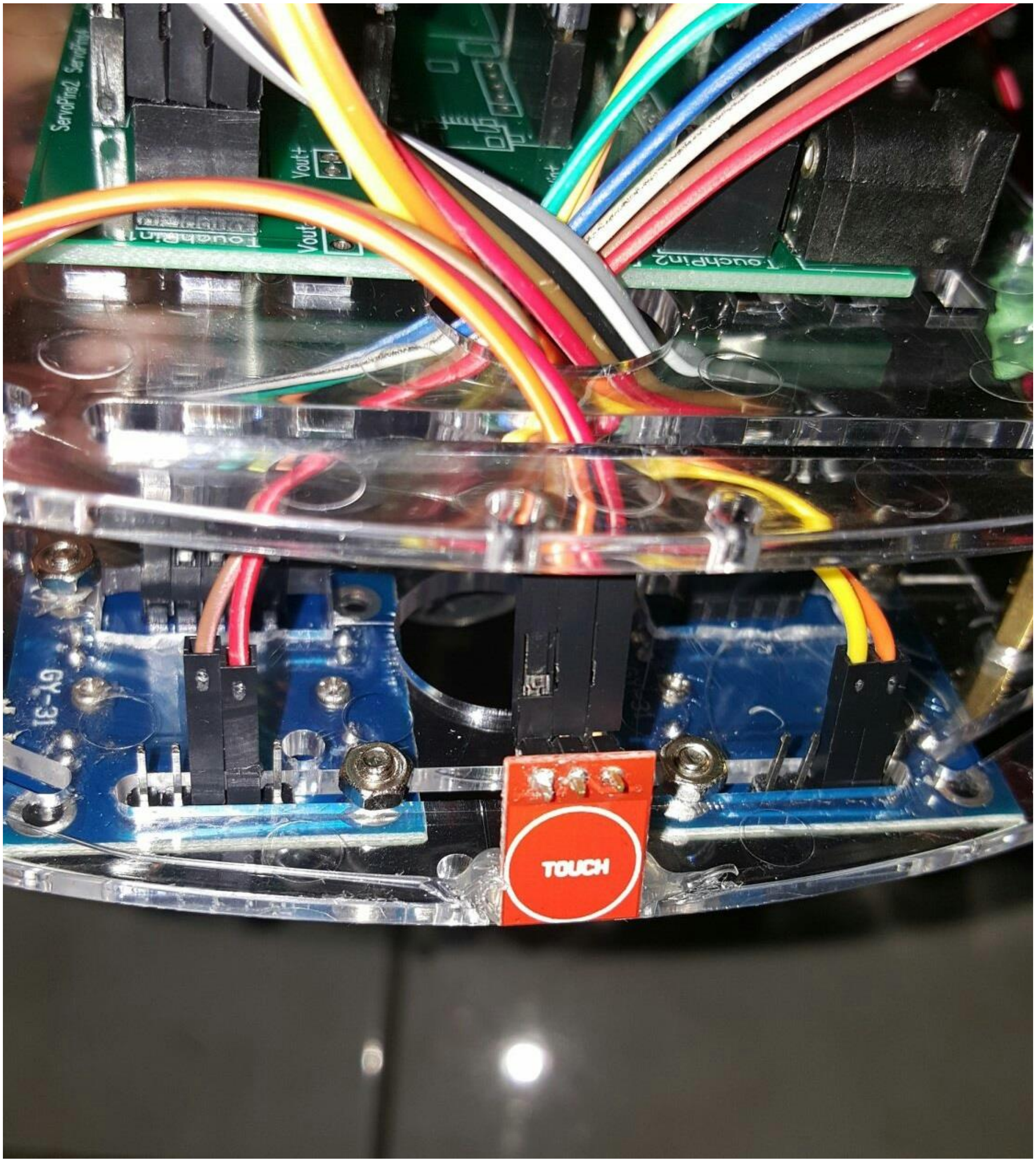


Figure 2.2: Drilled off sections behind the red/brown wires and orange/yellow wires to mount the sensors

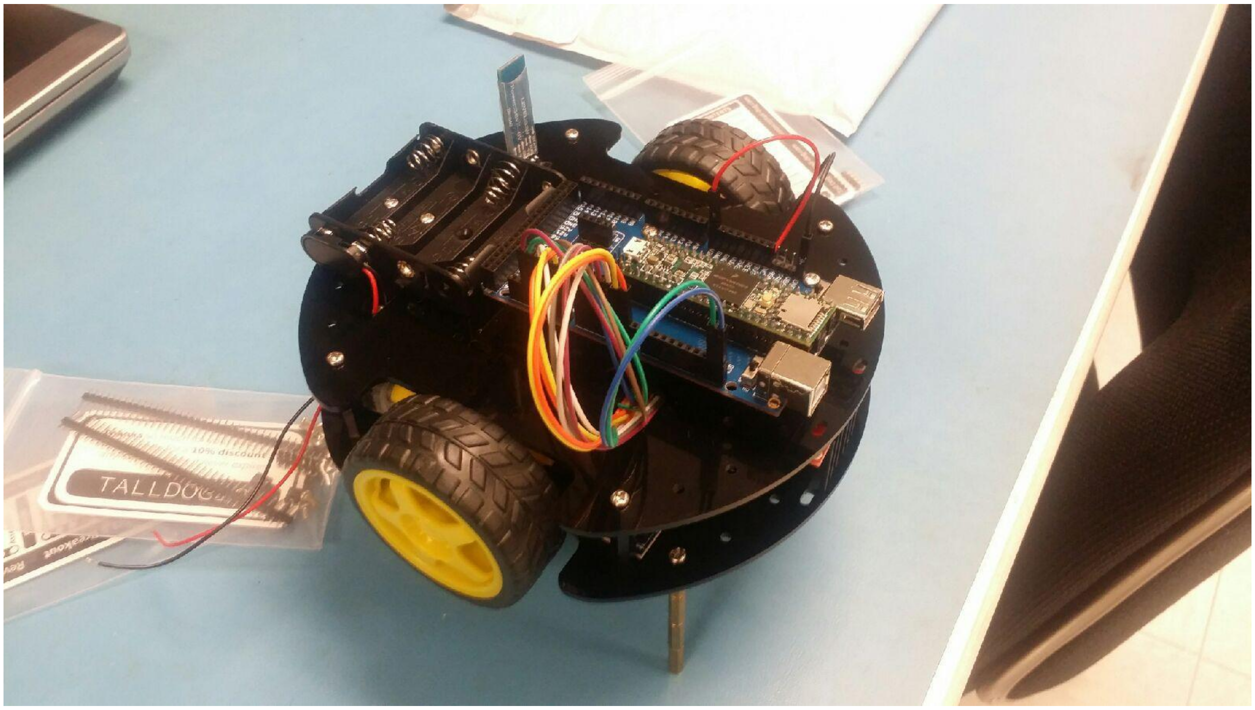
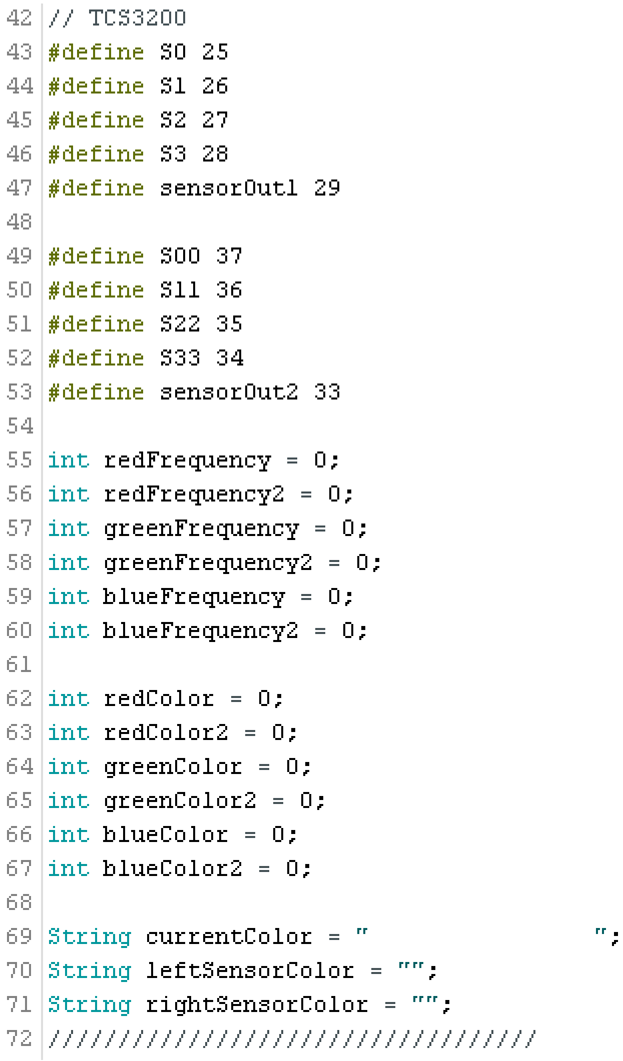


Figure 2.3: Circular chassis

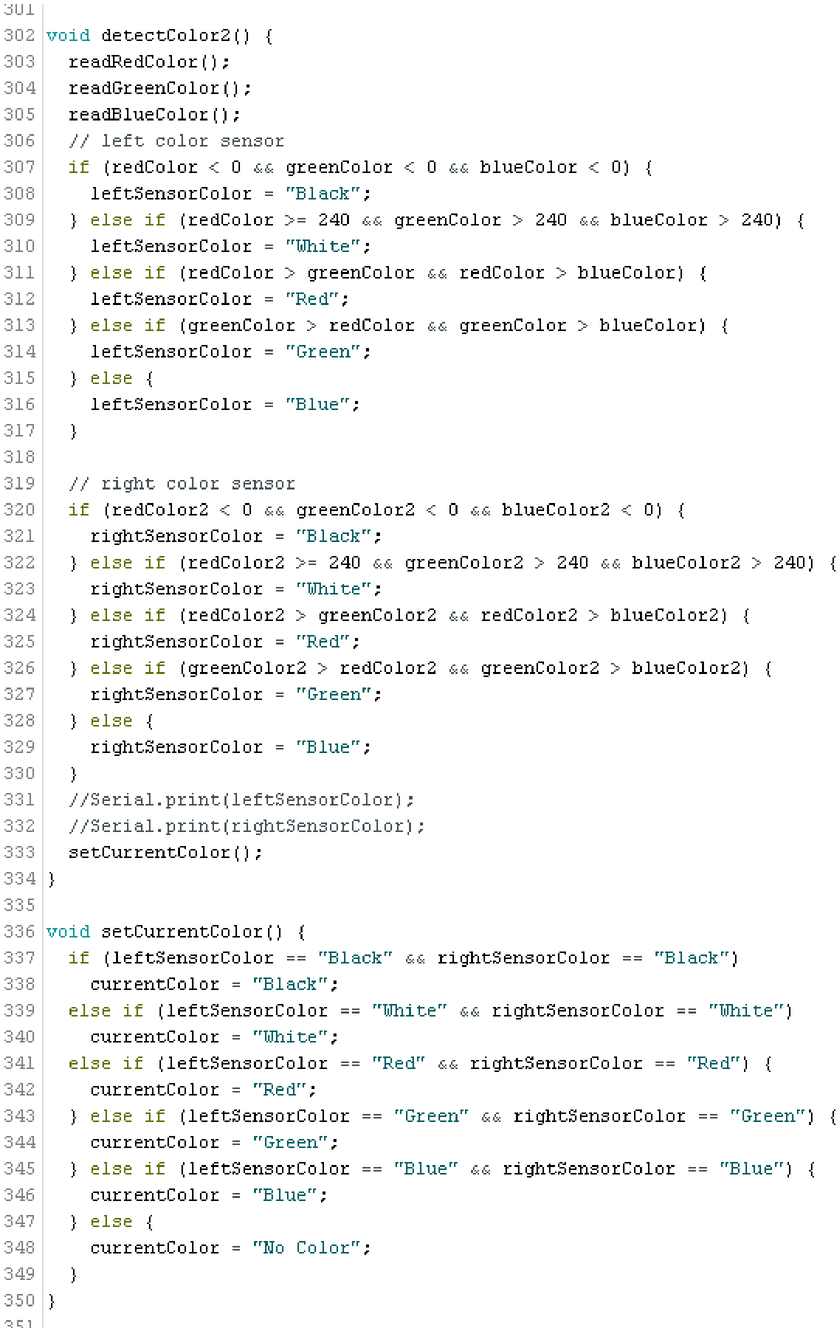
2.5 Programming the Rover to Detect Color

Even though the rover performed the same task, follow a colored path, they were programmed differently due to their sensors. The circular rover used the sensors premade library to detect the colors. The rectangular rover had to programmed almost from scratch because the sensors, since they were simpler to use and did not use I2C communication, were able to be interfaced with easily.



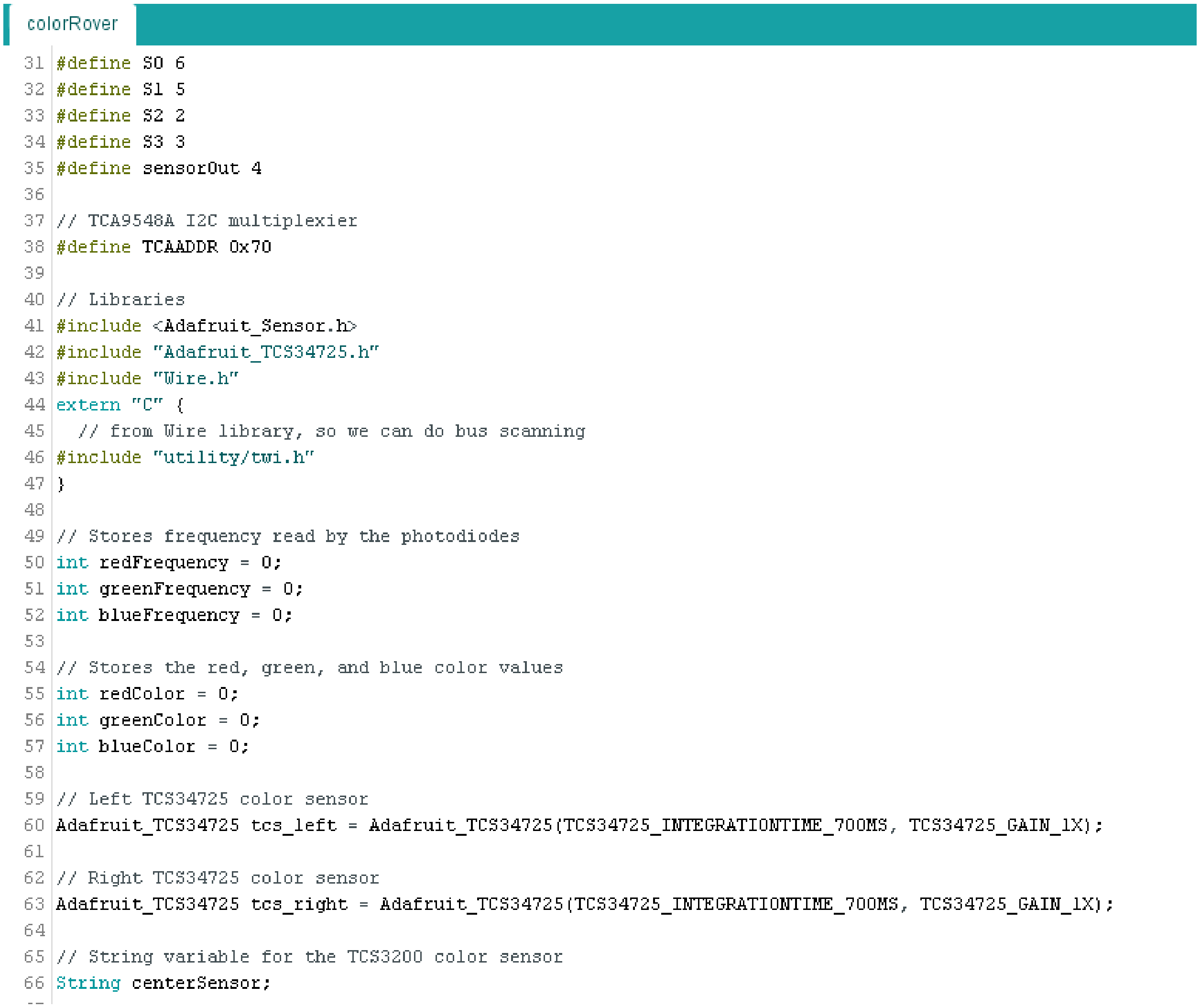
```
// TCS3200
#define S0 25
#define S1 26
#define S2 27
#define S3 28
#define sensorOut1 29

#define S00 37
#define S11 36
#define S22 35
#define S33 34
#define sensorOut2 33

int redFrequency = 0;
int redFrequency2 = 0;
int greenFrequency = 0;
int greenFrequency2 = 0;
int blueFrequency = 0;
int blueFrequency2 = 0;

int redColor = 0;
int redColor2 = 0;
int greenColor = 0;
int greenColor2 = 0;
int blueColor = 0;
int blueColor2 = 0;

String currentColor = "                ";
String leftSensorColor = "";
String rightSensorColor = "";
///////////////////////////////////
```

Figure 2.5.1: Variables used in the rectangular chassis to detect color



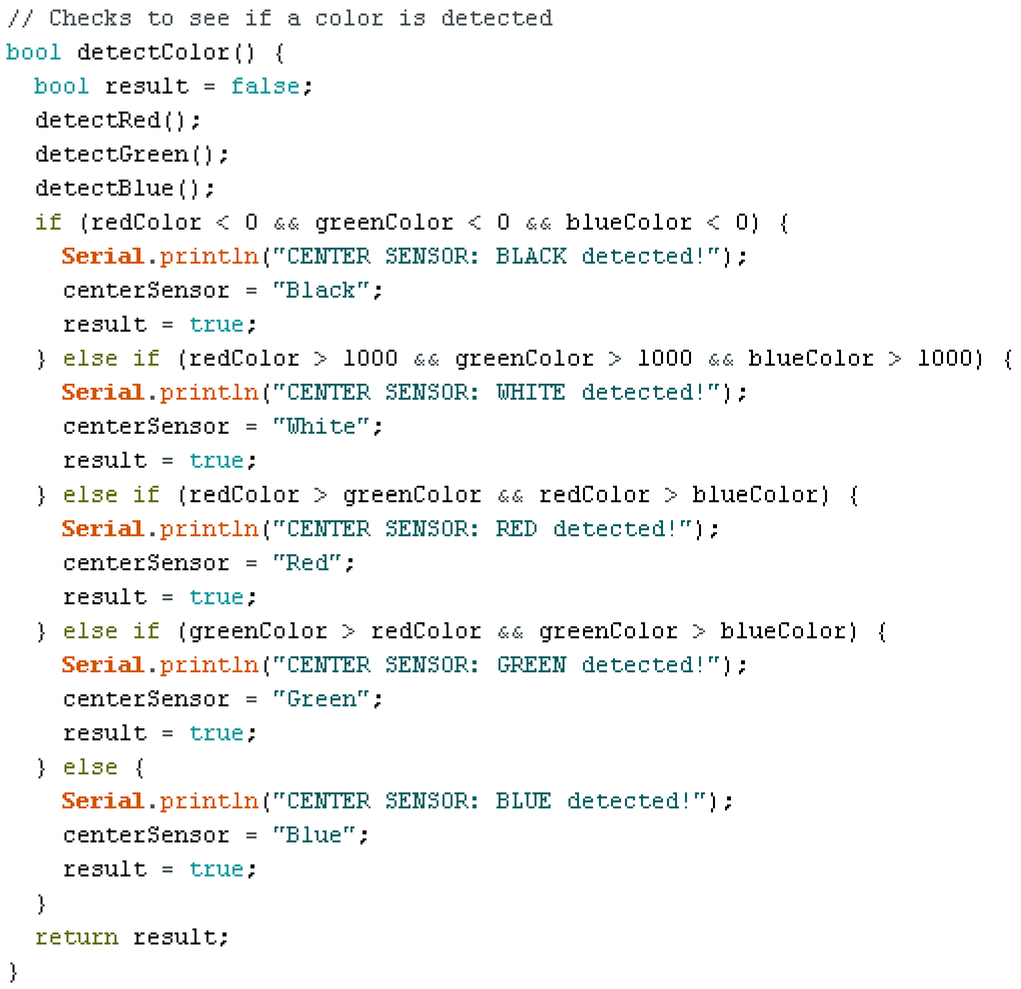
```
void detectColor2() {
  readRedColor();
  readGreenColor();
  readBlueColor();
  // left color sensor
  if (redColor < 0 && greenColor < 0 && blueColor < 0) {
    leftSensorColor = "Black";
  } else if (redColor >= 240 && greenColor > 240 && blueColor > 240) {
    leftSensorColor = "White";
  } else if (redColor > greenColor && redColor > blueColor) {
    leftSensorColor = "Red";
  } else if (greenColor > redColor && greenColor > blueColor) {
    leftSensorColor = "Green";
  } else {
    leftSensorColor = "Blue";
  }

  // right color sensor
  if (redColor2 < 0 && greenColor2 < 0 && blueColor2 < 0) {
    rightSensorColor = "Black";
  } else if (redColor2 >= 240 && greenColor2 > 240 && blueColor2 > 240) {
    rightSensorColor = "White";
  } else if (redColor2 > greenColor2 && redColor2 > blueColor2) {
    rightSensorColor = "Red";
  } else if (greenColor2 > redColor2 && greenColor2 > blueColor2) {
    rightSensorColor = "Green";
  } else {
    rightSensorColor = "Blue";
  }
  //Serial.print(leftSensorColor);
  //Serial.print(rightSensorColor);
  setCurrentColor();
}

void setCurrentColor() {
  if (leftSensorColor == "Black" && rightSensorColor == "Black")
    currentColor = "Black";
  else if (leftSensorColor == "White" && rightSensorColor == "White")
    currentColor = "White";
  else if (leftSensorColor == "Red" && rightSensorColor == "Red") {
    currentColor = "Red";
  } else if (leftSensorColor == "Green" && rightSensorColor == "Green") {
    currentColor = "Green";
  } else if (leftSensorColor == "Blue" && rightSensorColor == "Blue") {
    currentColor = "Blue";
  } else {
    currentColor = "No Color";
  }
}
```

Figure 2.5.2: Method used to detect color and set the current color of both sensors



```
#define S0 6
#define S1 5
#define S2 2
#define S3 3
#define sensorOut 4

// TCA9548A I2C multiplexier
#define TCAADDR 0x70

// Libraries
#include <Adafruit_Sensor.h>
#include "Adafruit_TCS34725.h"
#include "Wire.h"
extern "C" {
  // from Wire library, so we can do bus scanning
#include "utility/twi.h"
}

// Stores frequency read by the photodiodes
int redFrequency = 0;
int greenFrequency = 0;
int blueFrequency = 0;

// Stores the red, green, and blue color values
int redColor = 0;
int greenColor = 0;
int blueColor = 0;

// Left TCS34725 color sensor
Adafruit_TCS34725 tcs_left = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS, TCS34725_GAIN_1X);

// Right TCS34725 color sensor
Adafruit_TCS34725 tcs_right = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS, TCS34725_GAIN_1X);

// String variable for the TCS3200 color sensor
String centerSensor;
```

Figure 2.5.3: Variables used in the circular chassis



```
// Checks to see if a color is detected
bool detectColor() {
  bool result = false;
  detectRed();
  detectGreen();
  detectBlue();
  if (redColor < 0 && greenColor < 0 && blueColor < 0) {
    Serial.println("CENTER SENSOR: BLACK detected!");
    centerSensor = "Black";
    result = true;
  } else if (redColor > 1000 && greenColor > 1000 && blueColor > 1000) {
    Serial.println("CENTER SENSOR: WHITE detected!");
    centerSensor = "White";
    result = true;
  } else if (redColor > greenColor && redColor > blueColor) {
    Serial.println("CENTER SENSOR: RED detected!");
    centerSensor = "Red";
    result = true;
  } else if (greenColor > redColor && greenColor > blueColor) {
    Serial.println("CENTER SENSOR: GREEN detected!");
    centerSensor = "Green";
    result = true;
  } else {
    Serial.println("CENTER SENSOR: BLUE detected!");
    centerSensor = "Blue";
    result = true;
  }
  return result;
}
```

Figure 2.5.4: Detecting the color with the center sensor of the circular chassis

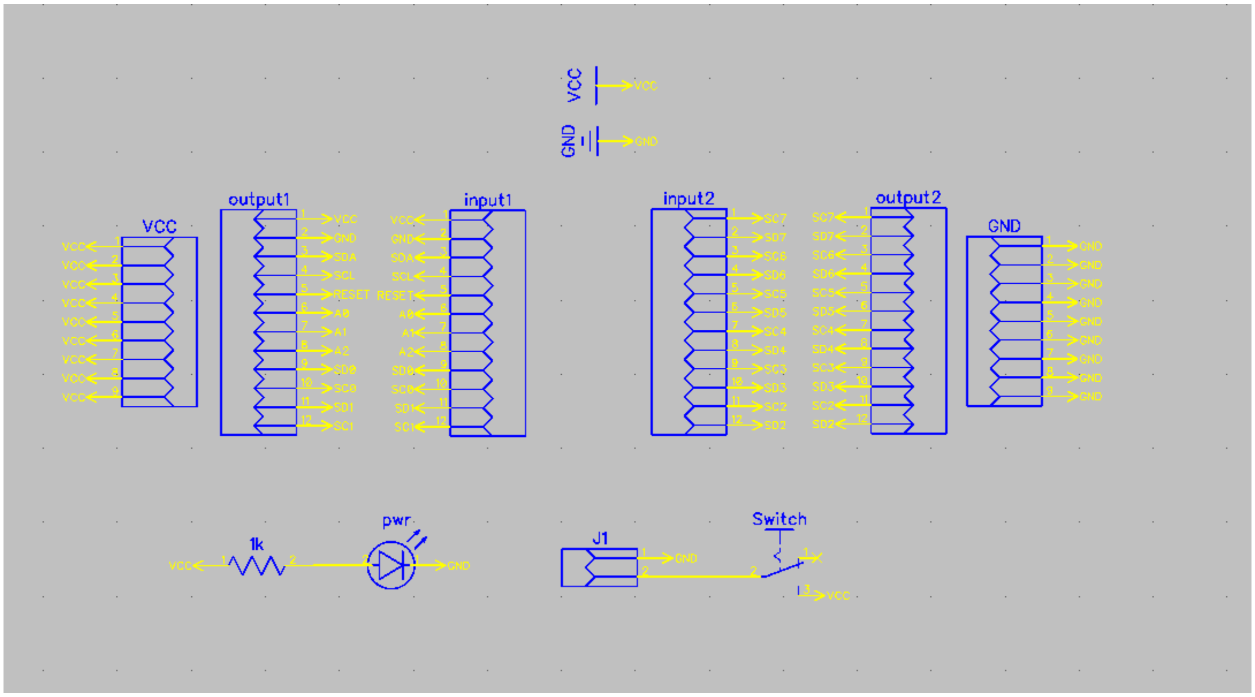
2.6 Connecting the Motors

The differing chassis gave different positions for the motors to be mounted. In the circular chassis, as seen in figure 2.3, the motors are very close to the center of the body. This allows for a very small turning radius and the robot would be able to turn without drastically changing the reading of the sensors. For the rectangular chassis, the motors are far from the center of the body, as seen in figure 2.1. This distance creates a huge turning radius that the circular chassis does not have. What this does to the reading of the sensors depends on how the motors are told to move. If the motors are made to keep moving forward as it turns, one sensor will almost always be off when the rover encounters a curve, or both sensors will almost always see the same color so when encountering a curve, the robot will go straight instead of actually turning when it should.

Another difference that was created between the two rovers was the types of motors used. In the circular chassis, DC motors were used, because they came with the chassis kit. The rectangular chassis, continuous servo motors were used. This choice of motors caused the PCB on the rectangular chassis to be designed to handle more voltage because the servos required a minimum of 4.8V to run. The DC motors, on the other hand, required only 3V to run which allowed the circular chassis' PCB to be designed to use less batteries.

2.7 Circuit Board Design

The circular chassis created a size constraint in the size of the PCB. This led to a smaller design, compared to the board on the rectangular chassis, and having to mount the designed PCB in a different space compared to the microcontroller PCB.



VCC
VCC
GND
GND
VCC
VCC VCC VCC VCC VCC VCC VCC VCC VCC
output1
VCC GND SDA SCL RESET A0 A1 A2 SD0 SC0 SD1 SC1
input1
VCC GND SDA SCL RESET A0 A1 A2 SD0 SC0 SD1 SC1
input2
SC7 SD7 SC6 SD6 SC5 SD5 SC4 SD4 SC3 SD3 SC2 SD2
output2
SC7 SD7 SC6 SD6 SC5 SD5 SC4 SD4 SC3 SD3 SC2 SD2
GND
GND GND GND GND GND GND GND GND GND
1 2 3 4 5 6 7 8 9 10 11 12
VCC
1k
pwr
GND
J1
GND
Switch
VCC

Figure 2.6.1: Circular chassis schematic

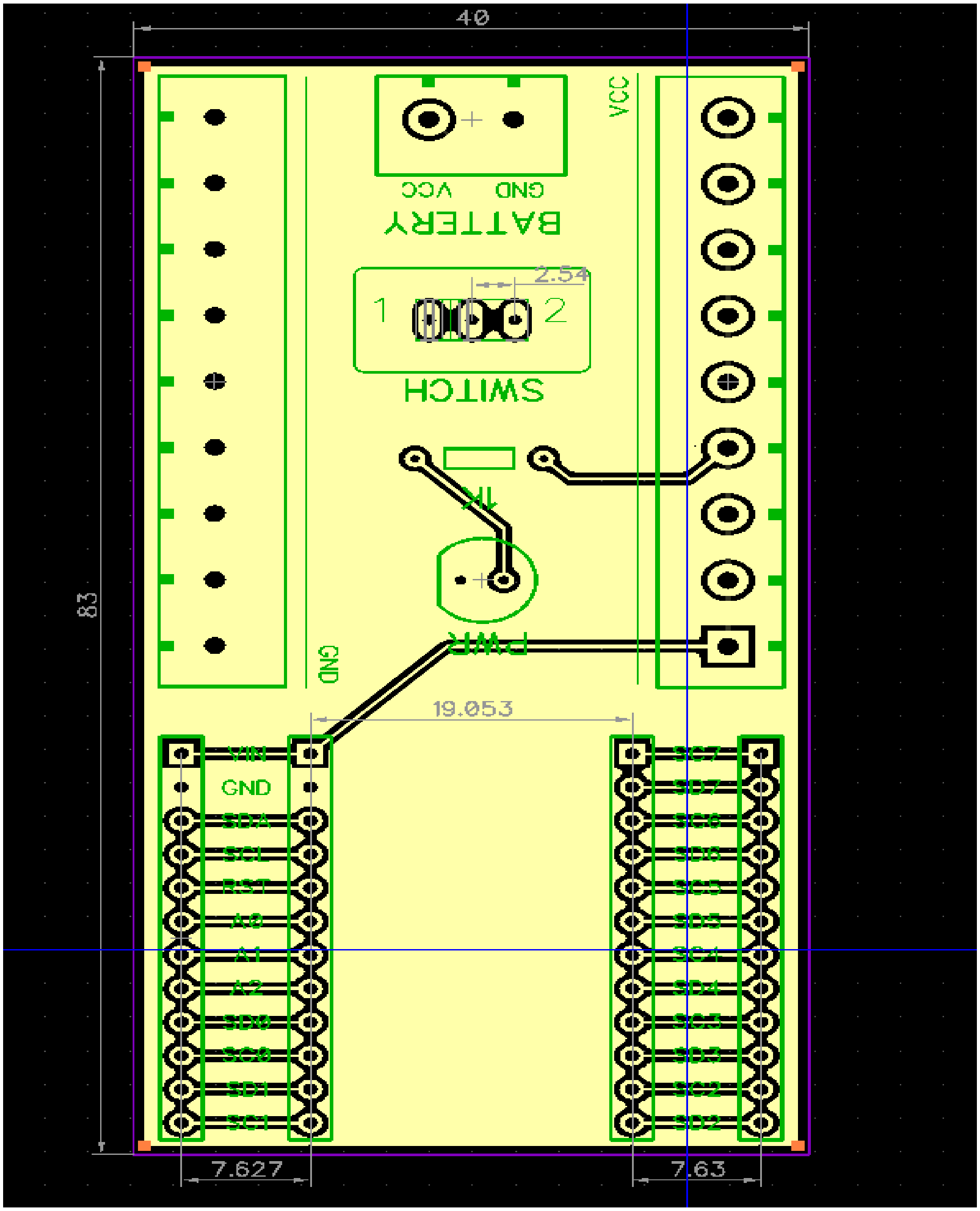


Figure 2.6.2: Circular chassis PCB

In comparison, the rectangular chassis allowed for a much bigger PCB to be designed. This extra space also allowed for extra features to be added onto the PCB.

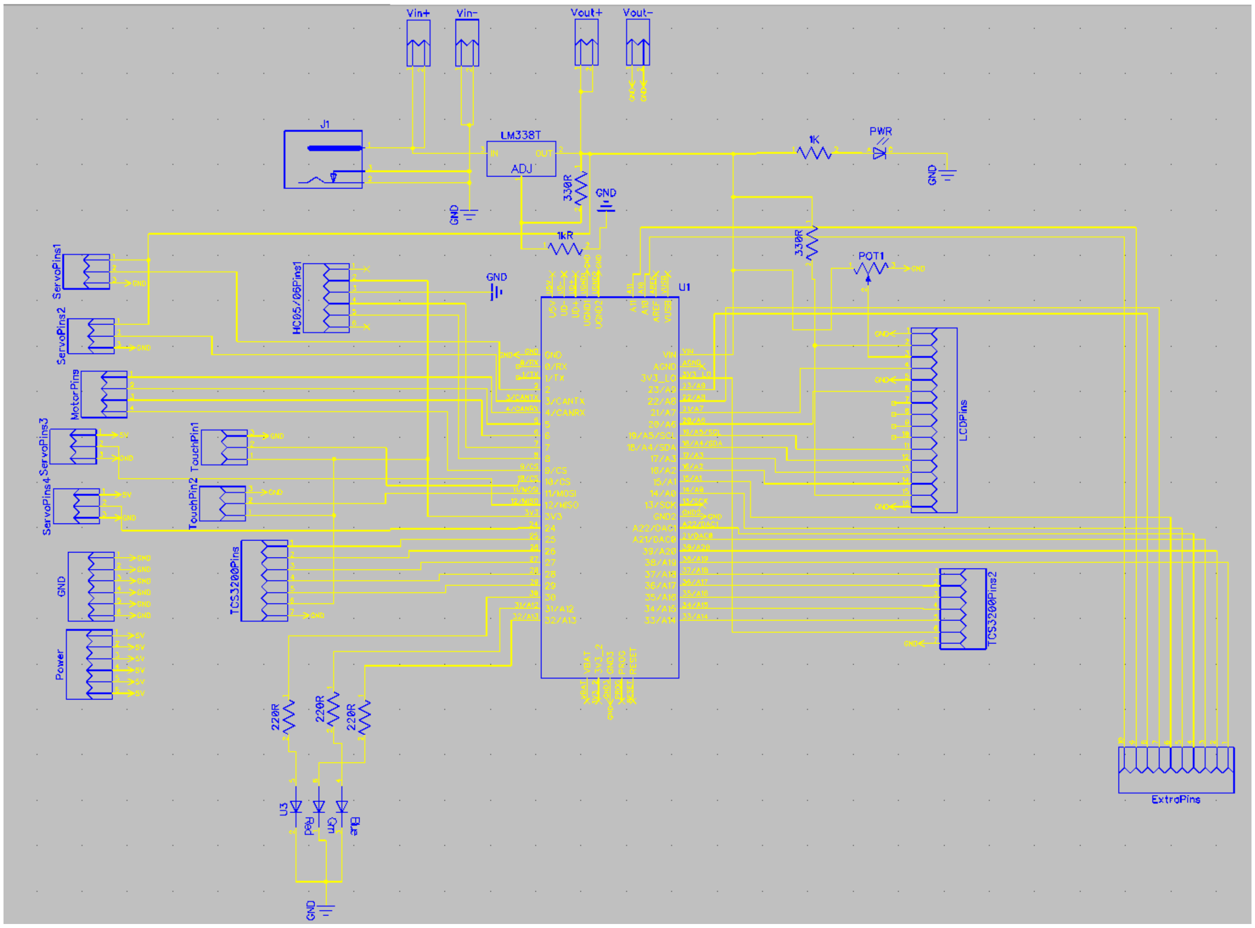


Figure 2.6.3: Rectangular chassis schematic

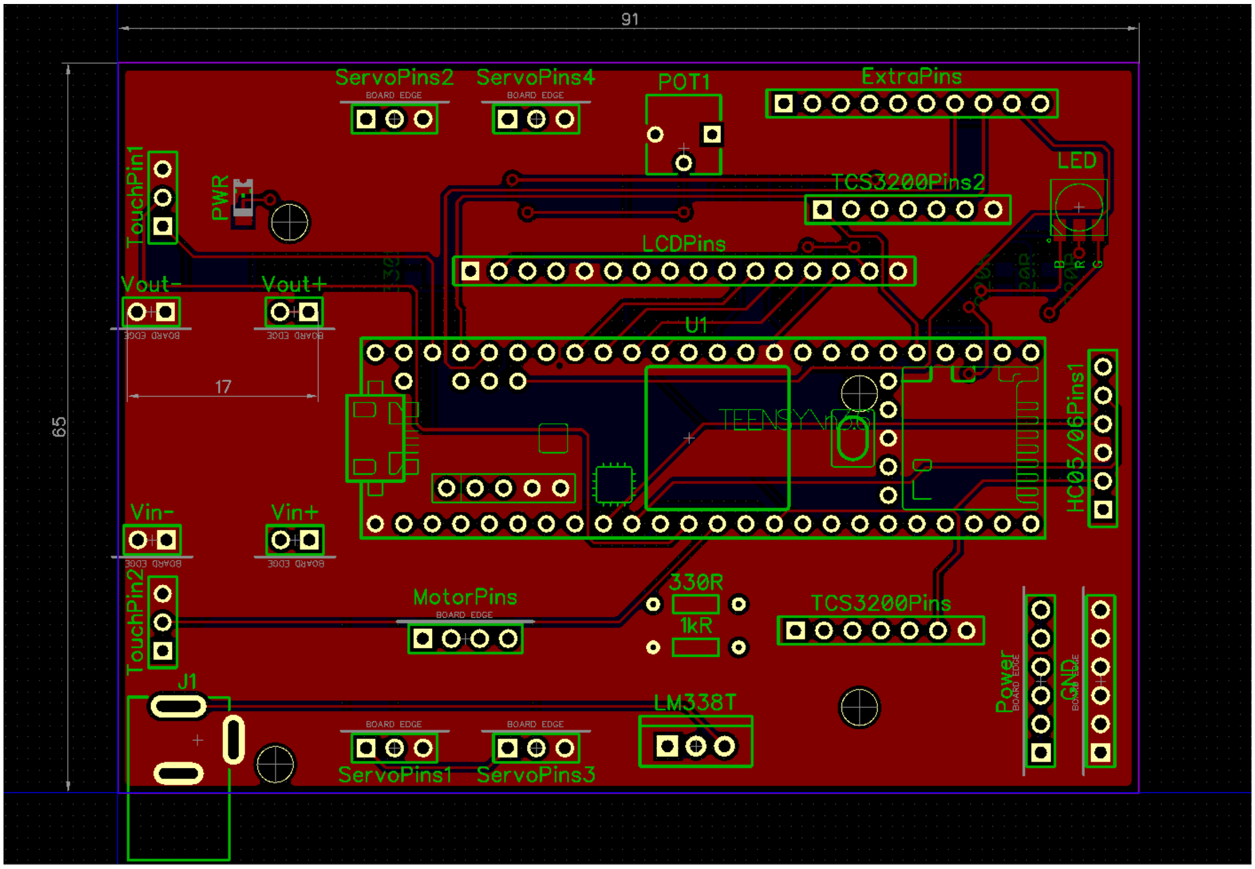
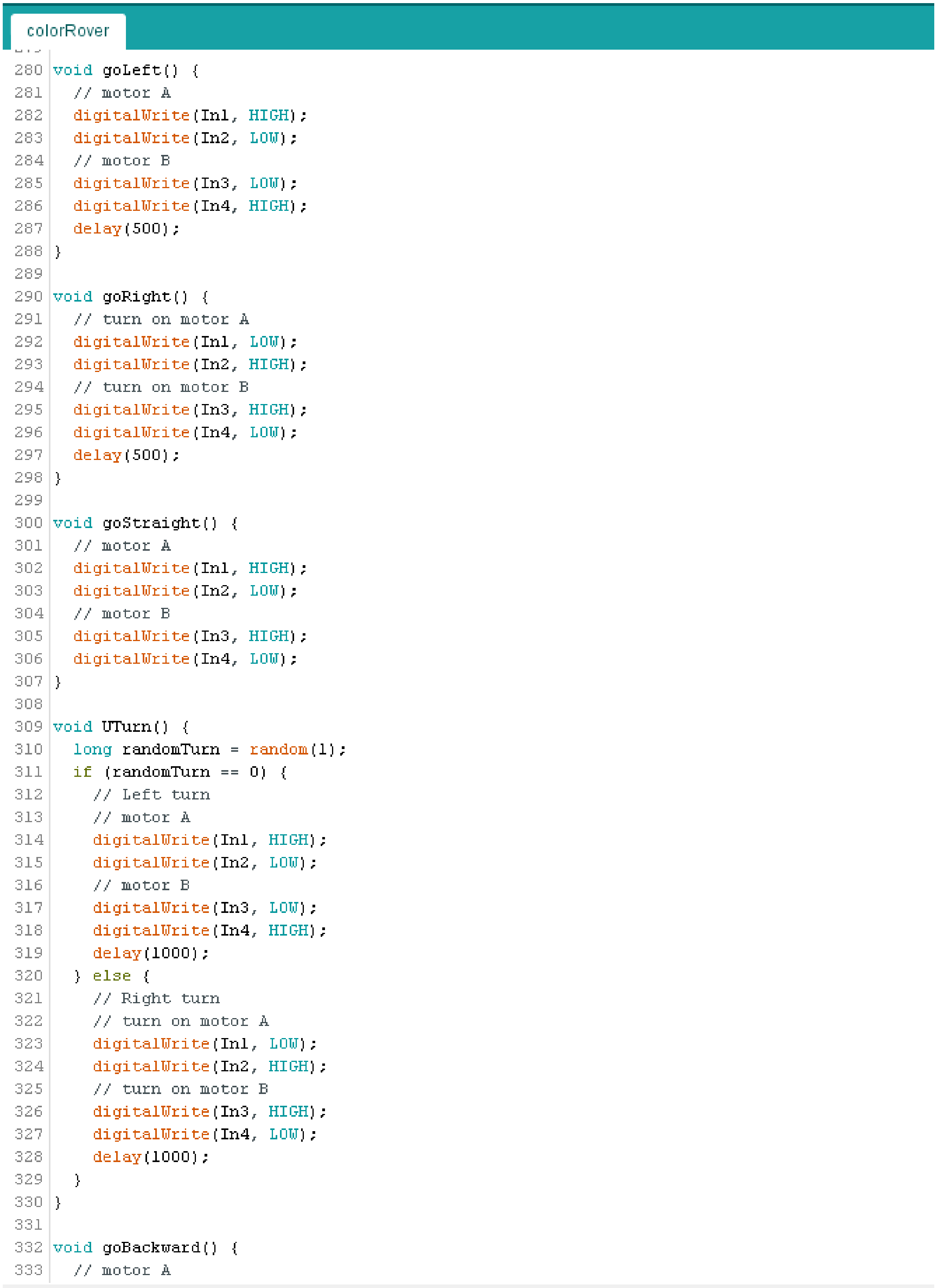


Figure 2.6.4: Rectangular chassis PCB

As can be seen from the size measurements in figures 2.6.4 and 2.6.2, the PCBs are not similar in size. The PCB for the circular rover was designed to be small in width but not in length due to the screw terminal connectors being used to have secure connections. In the PCB for the rectangular chassis, the size is very different compared to the circular chassis' PCB. This size difference was due to the fact that extra features were added, such as a connection for a 16x2 LCD display, extra power rails, six extra motors, two extra servo motor pins and four extra DC motor pins, and a connection to the unused pins on the Teensy 3.6. Both designs achieved the same task even though the larger PCB added extra features.

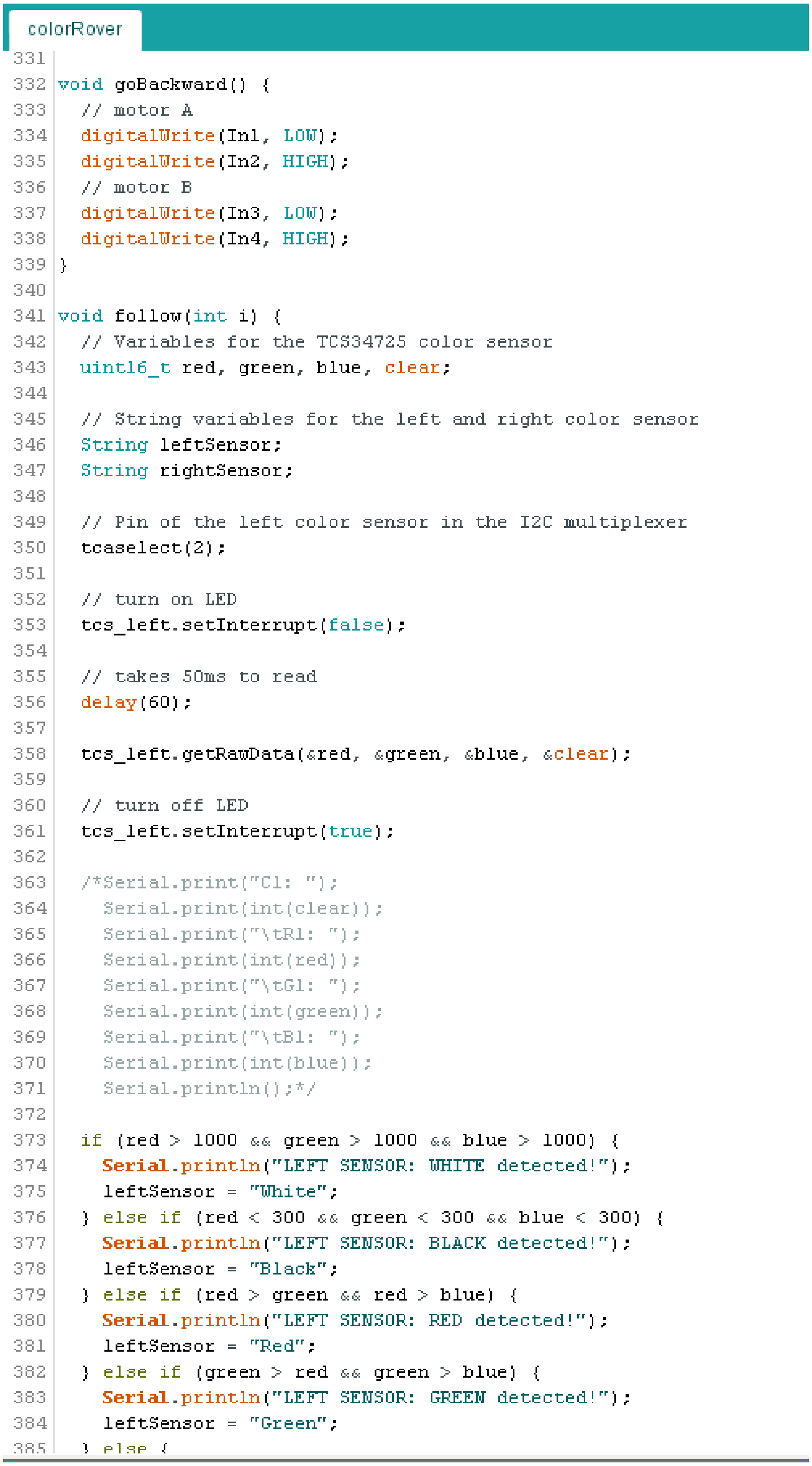
2.8 Following a Colored Path

This is where the robots became one again, in that even though they were built differently, they performed the same task. In the case of the circular rover, the rover was coded around the library of the TCS3472 sensor. This library allowed for calibration of the sensors to be performed before testing was done. This allowed for greater accuracy when searching for the edge of the path. The center sensor was also used in order to be able to detect the color of the path. While the edge sensors were the main sensors being used when the "follow" command was given, the center sensor acted as a way to test if the rover is detecting a color.



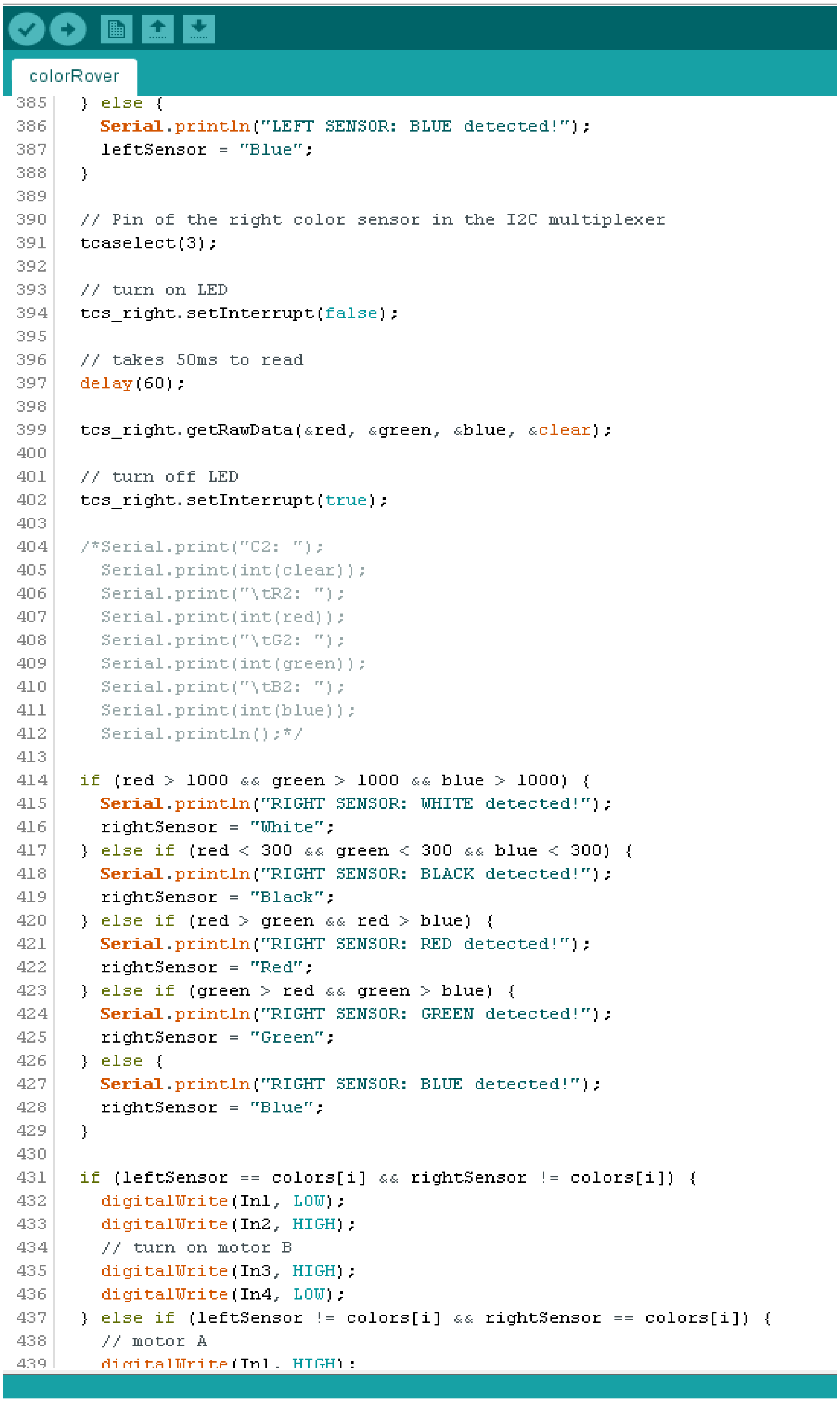
```
void goLeft() {
  // motor A
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  // motor B
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);
  delay(500);
}

void goRight() {
  // turn on motor A
  digitalWrite(In1, LOW);
  digitalWrite(In2, HIGH);
  // turn on motor B
  digitalWrite(In3, HIGH);
  digitalWrite(In4, LOW);
  delay(500);
}

void goStraight() {
  // motor A
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  // motor B
  digitalWrite(In3, HIGH);
  digitalWrite(In4, LOW);
}

void UTurn() {
  long randomTurn = random(1);
  if (randomTurn == 0) {
    // Left turn
    // motor A
    digitalWrite(In1, HIGH);
    digitalWrite(In2, LOW);
    // motor B
    digitalWrite(In3, LOW);
    digitalWrite(In4, HIGH);
    delay(1000);
  } else {
    // Right turn
    // turn on motor A
    digitalWrite(In1, LOW);
    digitalWrite(In2, HIGH);
    // turn on motor B
    digitalWrite(In3, HIGH);
    digitalWrite(In4, LOW);
    delay(1000);
  }
}

void goBackward() {
  // motor A
```

Figure 2.7.1: Code to turn rover in different directions and follow colored path



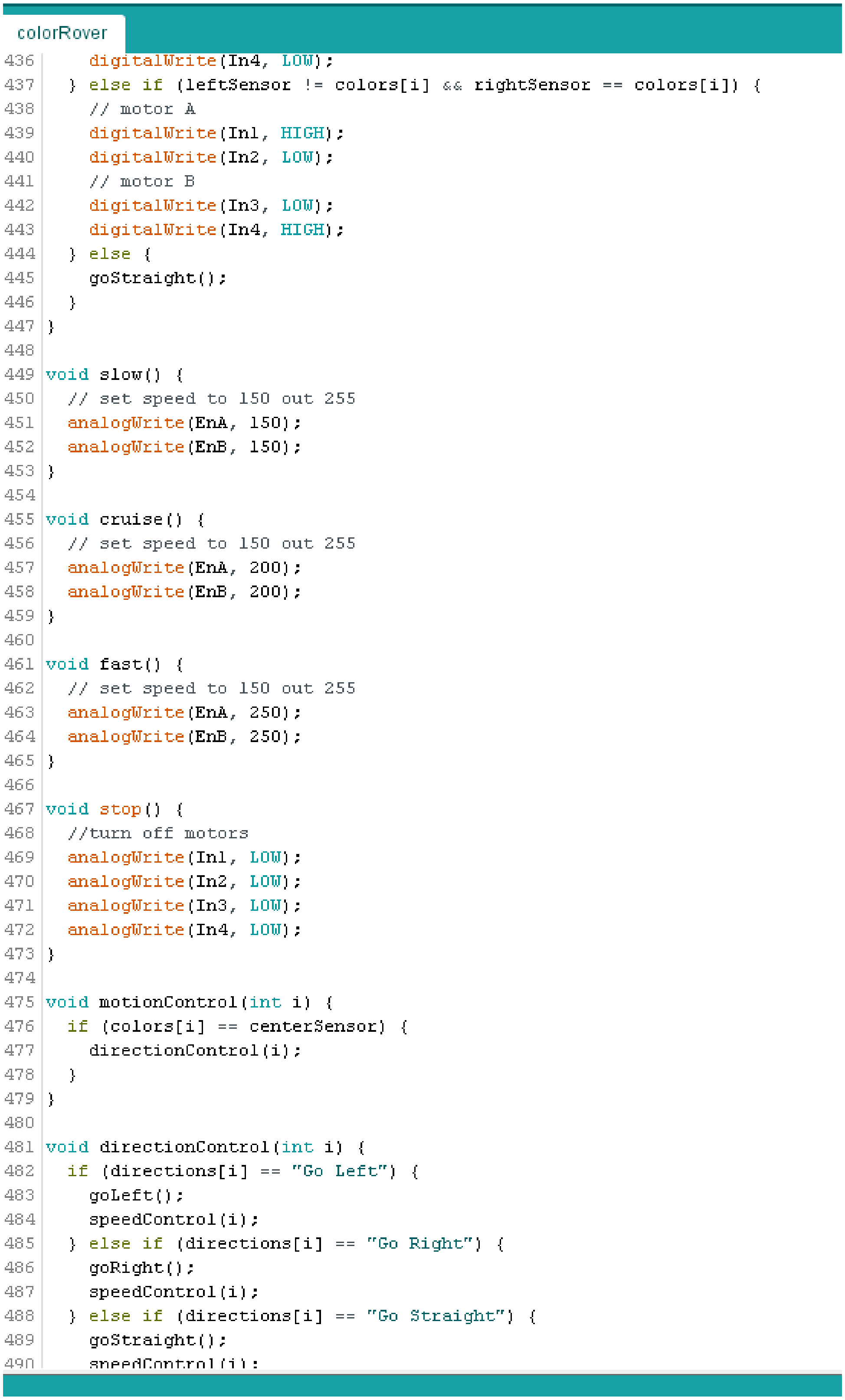
```

void goBackward() {
  // motor A
  digitalWrite(In1, LOW);
  digitalWrite(In2, HIGH);
  // motor B
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);
}

void follow(int i) {
  // Variables for the TCS34725 color sensor
  uint16_t red, green, blue, clear;

  // String variables for the left and right color sensor
  String leftSensor;
  String rightSensor;

  // Pin of the left color sensor in the I2C multiplexer
  tcaselect(2);

  // turn on LED
  tcs_left.setInterrupt(false);

  // takes 50ms to read
  delay(60);

  tcs_left.getRawData(&red, &green, &blue, &clear);

  // turn off LED
  tcs_left.setInterrupt(true);

  /*Serial.print("C1: ");
    Serial.print(int(clear));
    Serial.print("\tR1: ");
    Serial.print(int(red));
    Serial.print("\tG1: ");
    Serial.print(int(green));
    Serial.print("\tB1: ");
    Serial.print(int(blue));
    Serial.println();*/

  if (red > 1000 && green > 1000 && blue > 1000) {
    Serial.println("LEFT SENSOR: WHITE detected!");
    leftSensor = "White";
  } else if (red < 300 && green < 300 && blue < 300) {
    Serial.println("LEFT SENSOR: BLACK detected!");
    leftSensor = "Black";
  } else if (red > green && red > blue) {
    Serial.println("LEFT SENSOR: RED detected!");
    leftSensor = "Red";
  } else if (green > red && green > blue) {
    Serial.println("LEFT SENSOR: GREEN detected!");
    leftSensor = "Green";
  } else {
```

Figure 2.7.2: Code to turn rover in different directions and follow colored path



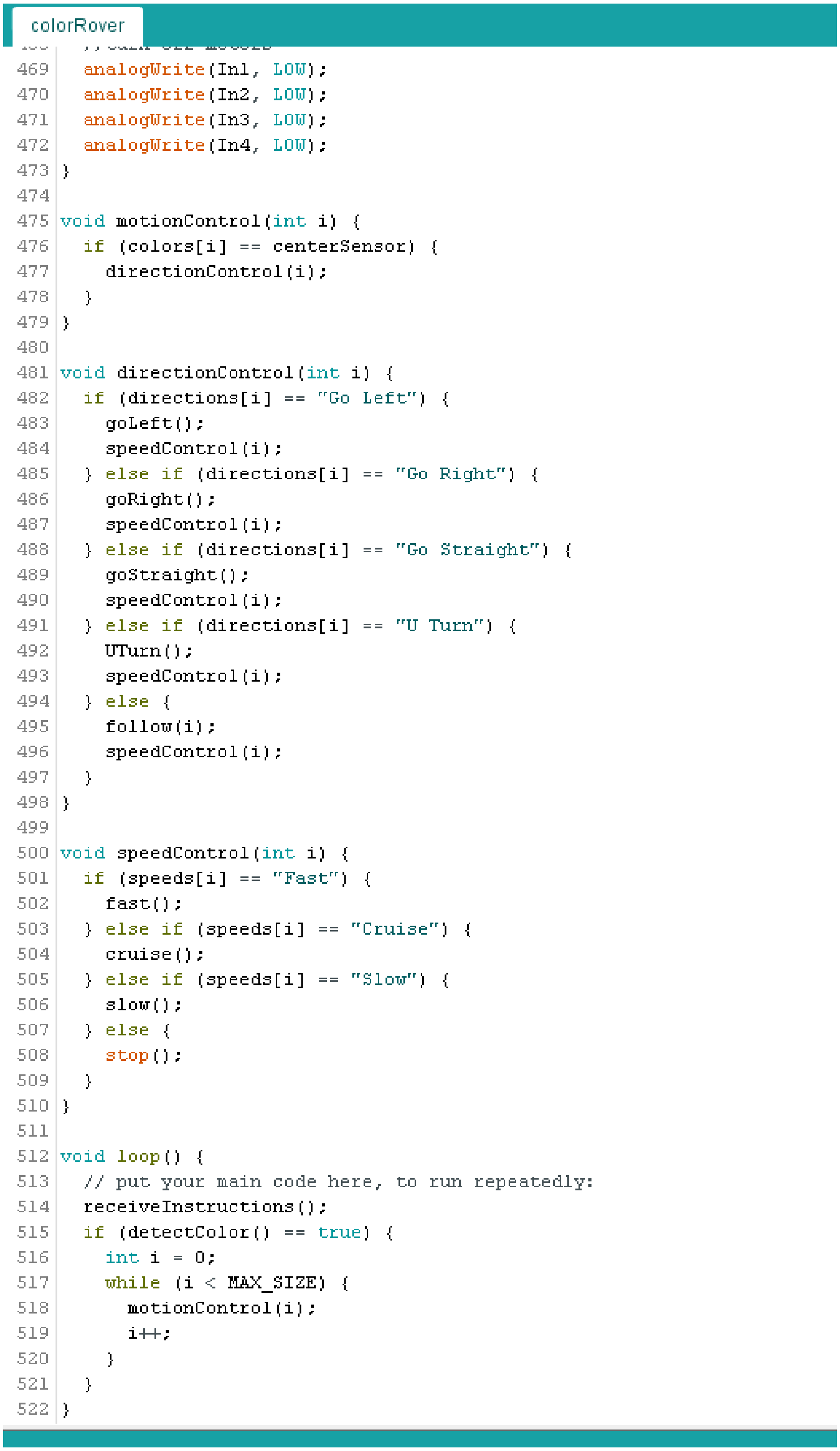
```
  } else {
    Serial.println("LEFT SENSOR: BLUE detected!");
    leftSensor = "Blue";
  }

  // Pin of the right color sensor in the I2C multiplexer
  tcaselect(3);

  // turn on LED
  tcs_right.setInterrupt(false);

  // takes 50ms to read
  delay(60);

  tcs_right.getRawData(&red, &green, &blue, &clear);

  // turn off LED
  tcs_right.setInterrupt(true);

  /*Serial.print("C2: ");
    Serial.print(int(clear));
    Serial.print("\tR2: ");
    Serial.print(int(red));
    Serial.print("\tG2: ");
    Serial.print(int(green));
    Serial.print("\tB2: ");
    Serial.print(int(blue));
    Serial.println();*/

  if (red > 1000 && green > 1000 && blue > 1000) {
    Serial.println("RIGHT SENSOR: WHITE detected!");
    rightSensor = "White";
  } else if (red < 300 && green < 300 && blue < 300) {
    Serial.println("RIGHT SENSOR: BLACK detected!");
    rightSensor = "Black";
  } else if (red > green && red > blue) {
    Serial.println("RIGHT SENSOR: RED detected!");
    rightSensor = "Red";
  } else if (green > red && green > blue) {
    Serial.println("RIGHT SENSOR: GREEN detected!");
    rightSensor = "Green";
  } else {
    Serial.println("RIGHT SENSOR: BLUE detected!");
    rightSensor = "Blue";
  }

  if (leftSensor == colors[i] && rightSensor != colors[i]) {
    digitalWrite(In1, LOW);
    digitalWrite(In2, HIGH);
    // turn on motor B
    digitalWrite(In3, HIGH);
    digitalWrite(In4, LOW);
  } else if (leftSensor != colors[i] && rightSensor == colors[i]) {
    // motor A
    digitalWrite(In1, HIGH);
```

Figure 2.7.3: Code to turn rover in different directions and follow colored path



```
    digitalWrite(In4, LOW);
  } else if (leftSensor != colors[i] && rightSensor == colors[i]) {
    // motor A
    digitalWrite(In1, HIGH);
    digitalWrite(In2, LOW);
    // motor B
    digitalWrite(In3, LOW);
    digitalWrite(In4, HIGH);
  } else {
    goStraight();
  }
}

void slow() {
  // set speed to 150 out 255
  analogWrite(EnA, 150);
  analogWrite(EnB, 150);
}

void cruise() {
  // set speed to 150 out 255
  analogWrite(EnA, 200);
  analogWrite(EnB, 200);
}

void fast() {
  // set speed to 150 out 255
  analogWrite(EnA, 250);
  analogWrite(EnB, 250);
}

void stop() {
  //turn off motors
  analogWrite(In1, LOW);
  analogWrite(In2, LOW);
  analogWrite(In3, LOW);
  analogWrite(In4, LOW);
}

void motionControl(int i) {
  if (colors[i] == centerSensor) {
    directionControl(i);
  }
}

void directionControl(int i) {
  if (directions[i] == "Go Left") {
    goLeft();
    speedControl(i);
  } else if (directions[i] == "Go Right") {
    goRight();
    speedControl(i);
  } else if (directions[i] == "Go Straight") {
    goStraight();
    speedControl(i);
```

Figure 2.7.4: Code to turn rover in different directions and follow colored path

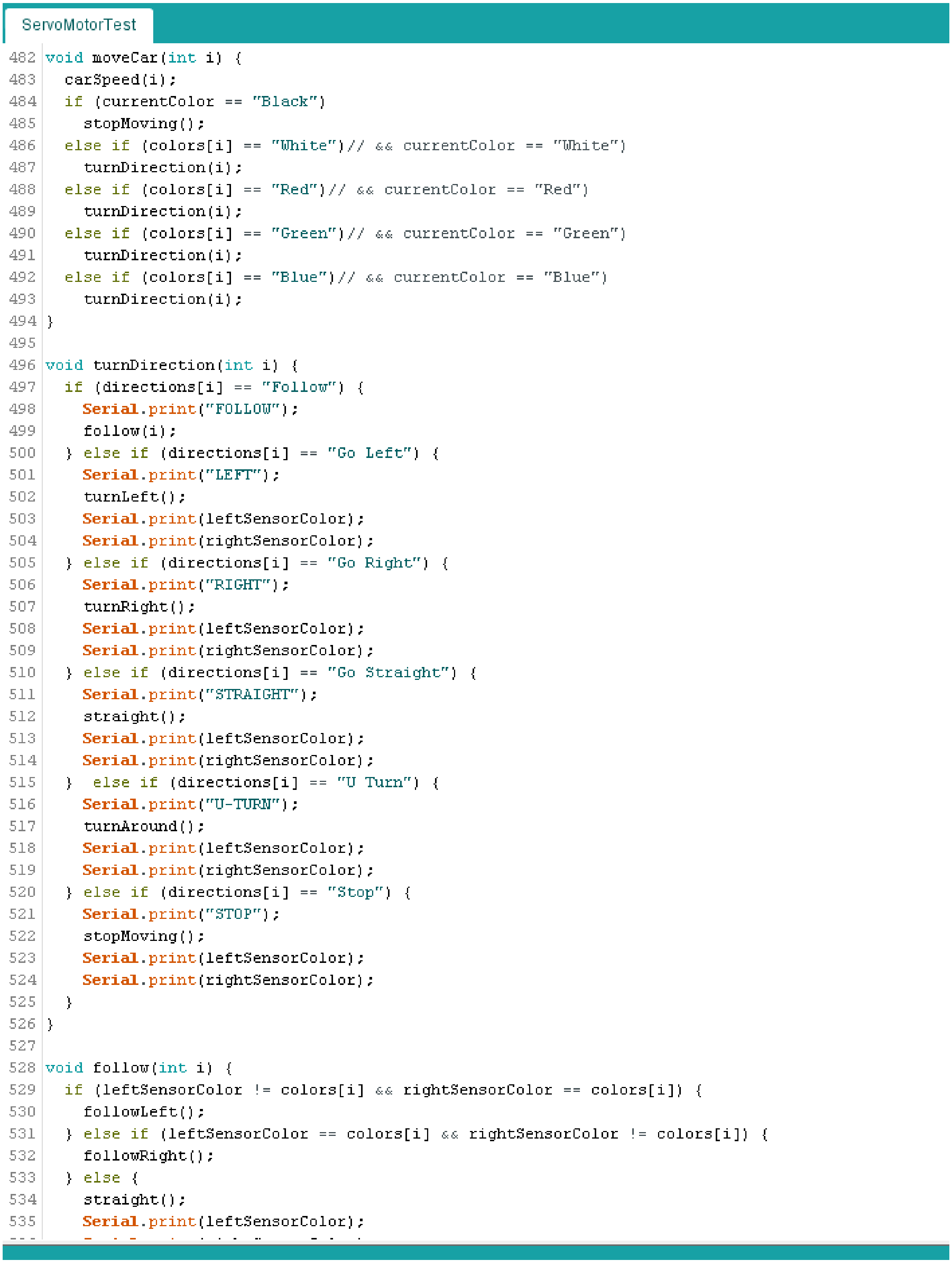


```
  // turn off motors
  analogWrite(In1, LOW);
  analogWrite(In2, LOW);
  analogWrite(In3, LOW);
  analogWrite(In4, LOW);
}

void motionControl(int i) {
  if (colors[i] == centerSensor) {
    directionControl(i);
  }
}

void directionControl(int i) {
  if (directions[i] == "Go Left") {
    goLeft();
    speedControl(i);
  } else if (directions[i] == "Go Right") {
    goRight();
    speedControl(i);
  } else if (directions[i] == "Go Straight") {
    goStraight();
    speedControl(i);
  } else if (directions[i] == "U Turn") {
    UTurn();
    speedControl(i);
  } else {
    follow(i);
    speedControl(i);
  }
}

void speedControl(int i) {
  if (speeds[i] == "Fast") {
    fast();
  } else if (speeds[i] == "Cruise") {
    cruise();
  } else if (speeds[i] == "Slow") {
    slow();
  } else {
    stop();
  }
}

void loop() {
  // put your main code here, to run repeatedly:
  receiveInstructions();
  if (detectColor() == true) {
    int i = 0;
    while (i < MAX_SIZE) {
      motionControl(i);
      i++;
    }
  }
}
```

Figure 2.7.5: Code to turn rover in different directions and follow colored path

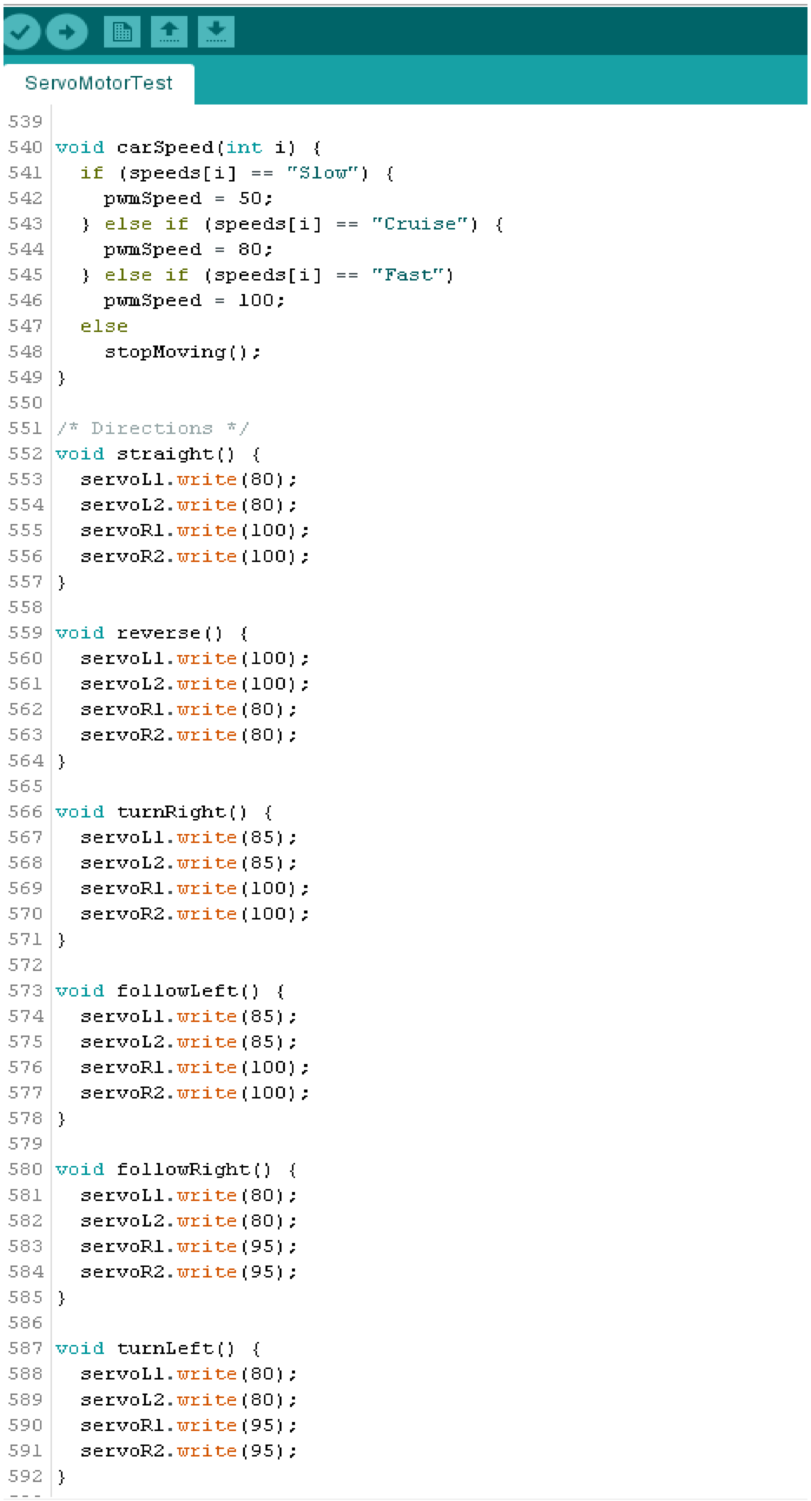
As can be seen from figures 2.7.2 and 2.7.3, the library is used to tell the Teensy when the sensor detects the path edge. The motors are then moved accordingly.

For the rectangular rover, the functions used were more simplistic because of the lack of a library. The functions of both rovers perform the same, only one is more accurate in its readings.



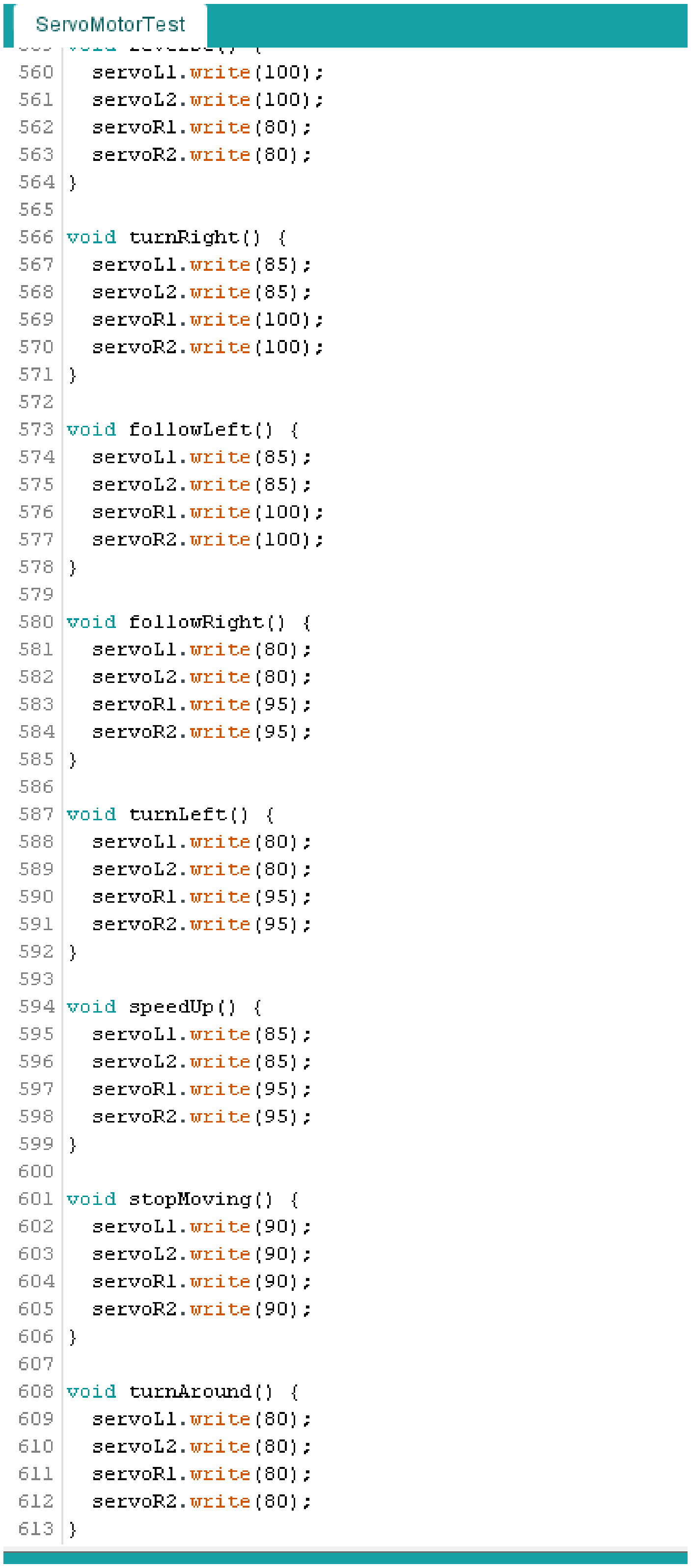
```
void moveCar(int i) {
  carSpeed(i);
  if (currentColor == "Black")
    stopMoving();
  else if (colors[i] == "White")// && currentColor == "White")
    turnDirection(i);
  else if (colors[i] == "Red")// && currentColor == "Red")
    turnDirection(i);
  else if (colors[i] == "Green")// && currentColor == "Green")
    turnDirection(i);
  else if (colors[i] == "Blue")// && currentColor == "Blue")
    turnDirection(i);
}

void turnDirection(int i) {
  if (directions[i] == "Follow") {
    Serial.print("FOLLOW");
    follow(i);
  } else if (directions[i] == "Go Left") {
    Serial.print("LEFT");
    turnLeft();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  } else if (directions[i] == "Go Right") {
    Serial.print("RIGHT");
    turnRight();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  } else if (directions[i] == "Go Straight") {
    Serial.print("STRAIGHT");
    straight();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  }  else if (directions[i] == "U Turn") {
    Serial.print("U-TURN");
    turnAround();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  } else if (directions[i] == "Stop") {
    Serial.print("STOP");
    stopMoving();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  }
}

void follow(int i) {
  if (leftSensorColor != colors[i] && rightSensorColor == colors[i]) {
    followLeft();
  } else if (leftSensorColor == colors[i] && rightSensorColor != colors[i]) {
    followRight();
  } else {
    straight();
    Serial.print(leftSensorColor);
```

Figure 2.7.5: Code to move rover depending on instructions sent



```
void carSpeed(int i) {
  if (speeds[i] == "Slow") {
    pwmSpeed = 50;
  } else if (speeds[i] == "Cruise") {
    pwmSpeed = 80;
  } else if (speeds[i] == "Fast")
    pwmSpeed = 100;
  else
    stopMoving();
}

/* Directions */
void straight() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(100);
  servoR2.write(100);
}

void reverse() {
  servoL1.write(100);
  servoL2.write(100);
  servoR1.write(80);
  servoR2.write(80);
}

void turnRight() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(100);
  servoR2.write(100);
}

void followLeft() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(100);
  servoR2.write(100);
}

void followRight() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(95);
  servoR2.write(95);
}

void turnLeft() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(95);
  servoR2.write(95);
}
```

Figure 2.7.5: Code to move rover depending on instructions sent



```
  servoL1.write(100);
  servoL2.write(100);
  servoR1.write(80);
  servoR2.write(80);
}

void turnRight() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(100);
  servoR2.write(100);
}

void followLeft() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(100);
  servoR2.write(100);
}

void followRight() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(95);
  servoR2.write(95);
}

void turnLeft() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(95);
  servoR2.write(95);
}

void speedUp() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(95);
  servoR2.write(95);
}

void stopMoving() {
  servoL1.write(90);
  servoL2.write(90);
  servoR1.write(90);
  servoR2.write(90);
}

void turnAround() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(80);
  servoR2.write(80);
}
```

Figure 2.7.5: Code to move rover depending on instructions sent

3. Materials

3.1 Circular Rover

- 1. Teensy 3.6
- Teensy 3.5 / 3.6 Breakout Due Revision A
- Bluetooth Slave Module (HC-06)
- Slide Switch
- BONATECH Arduino 2 Wheels Smart Car Chassis
- 2 DC Electric Motor 3-6V Dual Shaft Geared TT Magnetic Gearbox Engine
- Plastic Toy Car Tire Wheel (Outside: Φ67mm/2.6" Width: 27mm/1.06")
- Motor Drive Controller Board Module Dual H Bridge DC Stepper (L298N)
- Color Recognition Sensor Detector Module (TCS3200)
- 2 Adafruit color sensors (TCS34725)
- Adafruit I2C Multiplexer (TSA9548A)
- Step-Down Linear Voltage Regulator Module (5V Out, 6V to 12V In AMS1117-5.0 5.0V)
- 4 AA Alkaline batteries (1.5V each)
- 4 AA Battery Holder with 9V I Type Snap Connector Plastic Housing (LAMPVPATH) (6V)
- Capacitive Touch Switch Button Self-Lock Module (TTP223)

3.2 Rectangular Rover

- Teensy 3.6
- Bluetooth Master Module (HC-05)
- EMOZNY Arduino 4 Wheel Smart Car Chassis
- 2 FS90R Servo Motors
- 2 Color Recognition Sensor Detector Module (TCS3200)
- LM338 Adjustable Voltage Regulator
- 10 AA Alkaline batteries (1.5V each)
- 1 - 4 AA Battery Holder
- 1 – 1AA Battery Holder
- 1 - 2AA Battery Holder
- 1 - 3 AA Battery Holder
- Capacitive Touch Switch Button Self-Lock Module (TTP223)
- 1 RGB SMD LED CHANZON 5050
- 1 DC Jack
- 330Ω Resistor
- 1KΩ Resistor

4. Results & Conclusion

4.1 Color Tracking Accuracy

What was found was that the circular rover had better color tracking than the rectangular rover. This is mainly due to the extra sensor that the circular rover has. Most line tracking robots, that were found, had more than two IR sensors to detect the line it was following. The more sensors that were added, the more accurate the reading was and the smoother the robot would turn. The circular rover, along with its more advanced sensors, was able to track its path much more smoothly and quickly than the rectangular rover.

The rectangular rover, while it was able to track its path, had repeated errors when turning. Due to its huge frame, its turning radius was also huge so whenever it detected a color on both sensors, and started to move straight ahead, both sensors would move off the edge of the path. The floor would also have its own color so both sensors picked up this color and caused the rover to continue moving straight. While it worked, the turning radius was what caused most of the errors in the rover.

4.2 What can be Changed

The rectangular chassis could be switched out to an even smaller chassis or changed to the circular chassis. This would allow for more accurate readings when following its path. The only issue with that is that the motors and also the voltage regulator would have to be changed because the DC motors would not need a high voltage to run, they would spin too quickly, and the regulator could be replaced with another that does not have a relatively high voltage dropout.

5. Future Features

5.1 Adding Displays

One feature that can be added, or was not implemented correctly, are displays. In the rectangular chassis, the PCB contains an SMD LED that shows what color has been detected by the two sensors. This detection could be made easier to see if a display was added. As seen in figure 2.6.4, there is a 16x1 female header that says LCD Pins. This was an attempt at adding in a display to show what color has been detected. While it works, the LCD connections were not designed correctly. The LCD tells what color has been detected but once the backlight is turned on, it becomes difficult to see. This display could be added in a future version of the rover. Another display that could also be added is an LED matrix. It would require more code to run, but it could display the letters “R”, “G”, or “B” so that the user could see what color has been detected.

5.2 Implementing Different Environment Sensors

As can be seen from Appendix B, the rectangular chassis’ code has a space for a motion detector. This was in the original plan of the rover, to detect motion when it is not moving, but it was pushed as an extra feature because the motion detector did not match with the rover’s functionality. Due to the size of the chassis, the rover would be able to use extra sensors to be able to detect motion in its surroundings. Its only extra sensor is a touch sensor in the front of the rover, but that is only to move in a reverse direction when an object, with capacitive properties, comes into range of the touch pad.

6. References

Fagerness, T. (2015, November 10). How to Build a Robot - Line Follower. Retrieved from https://www.allaboutcircuits.com/projects/how-to-build-a-robot-line-follower/

LM317 / LM338 / LM350 Voltage Regulator Calculator and Circuits. (n.d.). Retrieved from https://diyaudioprojects.com/Technical/Voltage-Regulator/

Mybotic. (2017, October 23). Tutorial for TTP223 Touch Sensor Module (Capacitive). Retrieved from https://www.instructables.com/id/Tutorial-for-TTP223-Touch-Sensor-Module-Capacitive/

Random Nerd Tutorials. (2019, January 6). Arduino Color Sensor TCS230 TCS3200. Retrieved from https://randomnerdtutorials.com/arduino-color-sensor-tcs230-tcs3200/

ServoWrite. (n.d.). Retrieved from https://www.arduino.cc/en/Reference/ServoWrite

Teensy and Teensy++ Pinouts, for C language and Arduino Software. (n.d.). Retrieved from https://www.pjrc.com/teensy/pinout.html

Teensyduino: Using the UART (real serial) with Teensy on the Arduino IDE. (n.d.). Retrieved from https://www.pjrc.com/teensy/td_uart.html

Using the HC-05 Bluetooth Module. (n.d.). Retrieved from https://www.arduino-board.com/arduino/bluetooth

Appendix A: Circular Rover Code

```
/* Teensy RX/TX Pins */
#define HWSERIAL Serial1
// Constant variables
// Capacity to store instructions
#define MAX_SIZE 10
// Variables for receiving instructions
String data;
char array[MAX_SIZE * 10];
String strings[MAX_SIZE];
char *ptr = NULL;
String input;
int count = 1;
int i = 0;

// Arrays that store the instruction variables
String colors[MAX_SIZE];
String directions[MAX_SIZE];
String speeds[MAX_SIZE];

// DC Motor variables
#define In1 24
#define In2 25
#define In3 26
#define In4 27
#define EnB 29
#define EnA 30

// TCS3200 pins wiring to Arduino
#define S0 6
#define S1 5
#define S2 2
#define S3 3
#define sensorOut 4
```

```
// TCA9548A I2C multiplexier
#define TCAADDR 0x70

// Libraries
#include <Adafruit_Sensor.h>
#include "Adafruit_TCS34725.h"
#include "Wire.h"
extern "C" {
  // from Wire library, so we can do bus scanning
#include "utility/twi.h"
}

// Stores frequency read by the photodiodes
int redFrequency = 0;
int greenFrequency = 0;
int blueFrequency = 0;

// Stores the red, green, and blue color values
int redColor = 0;
int greenColor = 0;
int blueColor = 0;

// Left TCS34725 color sensor
Adafruit_TCS34725 tcs_left = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS, TCS34725_GAIN_1X);

// Right TCS34725 color sensor
Adafruit_TCS34725 tcs_right = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS, TCS34725_GAIN_1X);

// String variable for the TCS3200 color sensor
String centerSensor;

void setup() {
  // put your setup code here, to run once:
  //Sets RX/TX baud for reading from HC05
  HWSERIAL.begin(9600, SERIAL_8N1);
```

```
//Sets the baud for serial data transmission
//Serial.begin(9600);

// Outputs for DC motor controls
pinMode(EnA, OUTPUT);
pinMode(EnB, OUTPUT);
pinMode(In1, OUTPUT);
pinMode(In2, OUTPUT);
pinMode(In3, OUTPUT);
pinMode(In4, OUTPUT);

// Outputs for TCS3200 color sensor
pinMode(S0, OUTPUT);
pinMode(S1, OUTPUT);
pinMode(S2, OUTPUT);
pinMode(S3, OUTPUT);

// Setting the sensorOut as an input
pinMode(sensorOut, INPUT);

// Setting frequency scaling to 20%
digitalWrite(S0, HIGH);
digitalWrite(S1, LOW);

while (!HWSERIAL);
delay(1000);

Wire.begin();

Serial.begin(9600);
Serial.println("\nTCAScanner ready!");

for (uint8_t t = 0; t < 8; t++) {
  tcaselect(t);
  Serial.print("TCA Port #"); Serial.println(t);
```

```
    for (uint8_t addr = 0; addr <= 127; addr++) {
      if (addr == TCAADDR) continue;

      uint8_t data;
      if (! twi_writeTo(addr, &data, 0, 1, 1)) {
        Serial.print("Found I2C 0x");  Serial.println(addr, HEX);
      }
    }
  }
  Serial.println("\ndone");

  // Left color sensor I2C channel test
  tcaselect(2);
  if (tcs_left.begin()) {
    Serial.println("Left Color Sensor Found!");
  } else {
    Serial.println("No Left TCS34725 found ... check your connections");
    while (1);
  }

  // Right color sensor I2C channel test
  tcaselect(3);
  if (tcs_right.begin()) {
    Serial.println("Right Color Sensor Found!");
  } else {
    Serial.println("No Right TCS34725 found ... check your connections");
    while (1);
  }
}

// TCA9548A I2C channel test
void tcaselect(uint16_t i)
{
  if (i > 7) return;
  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

// receive and store instructions from bluetooth device
void receiveInstructions() {
  while (HWSERIAL.available() > 0) {
    data = HWSERIAL.readStringUntil(';');
    data.toCharArray(array, MAX_SIZE * 10);
    // takes a list of delimiters
    ptr = strtok(array, ":");
    int j = 0;
    while (ptr != NULL) {
      strings[j] = ptr;
      //Serial.println(strings[j]);
      if (strings[j] == strings[0]) {
        Serial.print("Color: ");
        Serial.print(strings[j]);
        colors[i] = strings[j];
        Serial.print(" --> colors[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(colors[i]);
      } else if (strings[j] == strings[1]) {
        Serial.print("Direction: ");
        Serial.print(strings[j]);
        directions[i] = strings[j];
        Serial.print(" --> directions[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(directions[i]);
      } else if (strings[j] == strings[2]) {
        Serial.print("Speed: ");
        Serial.print(strings[j]);
        speeds[i] = strings[j];
        Serial.print(" --> speeds[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(speeds[i]);
      }
      else {
        break;
      }
      j++;
      // takes a list of delimiters
      ptr = strtok(NULL, ":");
    }
    i++;
  }
}

//
void detectRed() {
  // Setting RED (R) filtered photodiodes to be read
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);

  // Reading the output frequency
  redFrequency = pulseIn(sensorOut, LOW);
  // Remaping the value of the RED (R) frequency from 0 to 255
  // You must replace with your own values. Here's an example:
  // redColor = map(redFrequency, 70, 120, 255,0);
  redColor = map(redFrequency, 35, 142, 255, 0);

  // Printing the RED (R) value
  Serial.print("R = ");
  Serial.print(redColor);
  delay(60);
```

```
}

void detectGreen() {
  // Setting GREEN (G) filtered photodiodes to be read
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);

  // Reading the output frequency
  greenFrequency = pulseIn(sensorOut, LOW);
  // Remaping the value of the GREEN (G) frequency from 0 to 255
  // You must replace with your own values. Here's an example:
  // greenColor = map(greenFrequency, 100, 199, 255, 0);
  greenColor = map(greenFrequency, 65, 257, 255, 0);

  // Printing the GREEN (G) value
  Serial.print(" G = ");
  Serial.print(greenColor);
  delay(60);
}

void detectBlue() {
  // Setting BLUE (B) filtered photodiodes to be read
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);

  // Reading the output frequency
  blueFrequency = pulseIn(sensorOut, LOW);
  // Remaping the value of the BLUE (B) frequency from 0 to 255
  // You must replace with your own values. Here's an example:
  // blueColor = map(blueFrequency, 38, 84, 255, 0);
  blueColor = map(blueFrequency, 43, 206, 255, 0);

  // Printing the BLUE (B) value
  Serial.print(" B = ");
  Serial.println(blueColor);
  delay(60);
}

// Checks to see if a color is detected
bool detectColor() {
  bool result = false;
  detectRed();
  detectGreen();
  detectBlue();
  if (redColor < 0 && greenColor < 0 && blueColor < 0) {
    Serial.println("CENTER SENSOR: BLACK detected!");
    centerSensor = "Black";
    result = true;
  } else if (redColor > 1000 && greenColor > 1000 && blueColor > 1000) {
    Serial.println("CENTER SENSOR: WHITE detected!");
    centerSensor = "White";
    result = true;
  } else if (redColor > greenColor && redColor > blueColor) {
    Serial.println("CENTER SENSOR: RED detected!");
    centerSensor = "Red";
    result = true;
  } else if (greenColor > redColor && greenColor > blueColor) {
    Serial.println("CENTER SENSOR: GREEN detected!");
    centerSensor = "Green";
    result = true;
  } else {
    Serial.println("CENTER SENSOR: BLUE detected!");
    centerSensor = "Blue";
    result = true;
  }
  return result;
}

void goLeft() {
  // motor A
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  // motor B
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);
  delay(500);
}

void goRight() {
  // turn on motor A
  digitalWrite(In1, LOW);
  digitalWrite(In2, HIGH);
  // turn on motor B
  digitalWrite(In3, HIGH);
  digitalWrite(In4, LOW);
  delay(500);
}

void goStraight() {
  // motor A
  digitalWrite(In1, HIGH);
  digitalWrite(In2, LOW);
  // motor B
  digitalWrite(In3, HIGH);
  digitalWrite(In4, LOW);
}

void UTurn() {
  long randomTurn = random(1);
  if (randomTurn == 0) {
    // Left turn
    // motor A
    digitalWrite(In1, HIGH);
    digitalWrite(In2, LOW);
    // motor B
    digitalWrite(In3, LOW);
    digitalWrite(In4, HIGH);
    delay(1000);
  } else {
    // Right turn
    // turn on motor A
    digitalWrite(In1, LOW);
    digitalWrite(In2, HIGH);
    // turn on motor B
    digitalWrite(In3, HIGH);
    digitalWrite(In4, LOW);
    delay(1000);
  }
}

void goBackward() {
  // motor A
  digitalWrite(In1, LOW);
  digitalWrite(In2, HIGH);
  // motor B
  digitalWrite(In3, LOW);
  digitalWrite(In4, HIGH);
}

void follow(int i) {
  // Variables for the TCS34725 color sensor
  uint16_t red, green, blue, clear;

  // String variables for the left and right color sensor
  String leftSensor;
  String rightSensor;

  // Pin of the left color sensor in the I2C multiplexer
  tcaselect(2);
```

```
// turn on LED
tcs_left.setInterrupt(false);

// takes 50ms to read
delay(60);

tcs_left.getRawData(&red, &green, &blue, &clear);

// turn off LED
tcs_left.setInterrupt(true);

/*Serial.print("C1: ");
  Serial.print(int(clear));
  Serial.print("\tR1: ");
  Serial.print(int(red));
  Serial.print("\tG1: ");
  Serial.print(int(green));
  Serial.print("\tB1: ");
  Serial.print(int(blue));
  Serial.println();*/

if (red > 1000 && green > 1000 && blue > 1000) {
  Serial.println("LEFT SENSOR: WHITE detected!");
  leftSensor = "White";
} else if (red < 300 && green < 300 && blue < 300) {
  Serial.println("LEFT SENSOR: BLACK detected!");
  leftSensor = "Black";
} else if (red > green && red > blue) {
  Serial.println("LEFT SENSOR: RED detected!");
  leftSensor = "Red";
} else if (green > red && green > blue) {
  Serial.println("LEFT SENSOR: GREEN detected!");
  leftSensor = "Green";
} else {
  Serial.println("LEFT SENSOR: BLUE detected!");
  leftSensor = "Blue";
}

// Pin of the right color sensor in the I2C multiplexer
tcaselect(3);

// turn on LED
tcs_right.setInterrupt(false);

// takes 50ms to read
delay(60);

tcs_right.getRawData(&red, &green, &blue, &clear);

// turn off LED
tcs_right.setInterrupt(true);

/*Serial.print("C2: ");
  Serial.print(int(clear));
  Serial.print("\tR2: ");
  Serial.print(int(red));
  Serial.print("\tG2: ");
  Serial.print(int(green));
  Serial.print("\tB2: ");
  Serial.print(int(blue));
  Serial.println();*/

if (red > 1000 && green > 1000 && blue > 1000) {
  Serial.println("RIGHT SENSOR: WHITE detected!");
  rightSensor = "White";
} else if (red < 300 && green < 300 && blue < 300) {
  Serial.println("RIGHT SENSOR: BLACK detected!");
  rightSensor = "Black";
} else if (red > green && red > blue) {
  Serial.println("RIGHT SENSOR: RED detected!");
```

```
    rightSensor = "Red";
  } else if (green > red && green > blue) {
    Serial.println("RIGHT SENSOR: GREEN detected!");
    rightSensor = "Green";
  } else {
    Serial.println("RIGHT SENSOR: BLUE detected!");
    rightSensor = "Blue";
  }

  if (leftSensor == colors[i] && rightSensor != colors[i]) {
    digitalWrite(In1, LOW);
    digitalWrite(In2, HIGH);
    // turn on motor B
    digitalWrite(In3, HIGH);
    digitalWrite(In4, LOW);
  } else if (leftSensor != colors[i] && rightSensor == colors[i]) {
    // motor A
    digitalWrite(In1, HIGH);
    digitalWrite(In2, LOW);
    // motor B
    digitalWrite(In3, LOW);
    digitalWrite(In4, HIGH);
  } else {
    goStraight();
  }
}

void slow() {
  // set speed to 150 out 255
  analogWrite(EnA, 150);
  analogWrite(EnB, 150);
}

void cruise() {
  // set speed to 150 out 255
  analogWrite(EnA, 200);
  analogWrite(EnB, 200);
}

void fast() {
  // set speed to 150 out 255
  analogWrite(EnA, 250);
  analogWrite(EnB, 250);
}

void stop() {
  //turn off motors
  analogWrite(In1, LOW);
  analogWrite(In2, LOW);
  analogWrite(In3, LOW);
  analogWrite(In4, LOW);
}

void motionControl(int i) {
  if (colors[i] == centerSensor) {
    directionControl(i);
  }
}

void directionControl(int i) {
  if (directions[i] == "Go Left") {
    goLeft();
    speedControl(i);
  } else if (directions[i] == "Go Right") {
    goRight();
    speedControl(i);
  } else if (directions[i] == "Go Straight") {
    goStraight();
    speedControl(i);
  } else if (directions[i] == "U Turn") {
    UTurn();
    speedControl(i);
  } else {
    follow(i);
    speedControl(i);
  }
}

void speedControl(int i) {
  if (speeds[i] == "Fast") {
    fast();
  } else if (speeds[i] == "Cruise") {
    cruise();
  } else if (speeds[i] == "Slow") {
    slow();
  } else {
    stop();
  }
}
void loop() {
  // put your main code here, to run repeatedly:
  receiveInstructions();
  if (detectColor() == true) {
    int i = 0;
    while (i < MAX_SIZE) {
      motionControl(i);
      i++;
    }
  }
}
```

Appendix B: Rectangular Rover Code

```
// Pins Used (Temp/Perm)  : 2,3,4,5,6,7,8,9,10,11,12,
//                  : 13,16,17,18,19,20,21,
//                  : 24,25,26,27,28,
//                  : 29,30,31
```

```
#include <Servo.h>
#include <LiquidCrystal.h>

Servo servoL1;
Servo servoR1;
Servo servoL2;
Servo servoR2;
LiquidCrystal lcd(21, 20, 19, 18, 17, 16);

const int ledPin = 13;
//int laser = 23;

/* RGB LED */
int redPin = 30;
int greenPin = 32;
int bluePin = 31;

/* Teensy RX/TX Pins */
#define HWSERIAL Serial3

/* BT Vars */
#define MAX_SIZE 15
String array[MAX_SIZE];
String input;
String data;            //Variable for storing received data
char *token = NULL;
String strings[MAX_SIZE];
char chars[MAX_SIZE * 90];
String colors[MAX_SIZE];
String directions[MAX_SIZE];
String speeds[MAX_SIZE];
int count = 1;
int i = 0;
///////////////////////////////////
```

```
/* Color Sensor Var */
// TCS3200
#define S0 25
#define S1 26
#define S2 27
#define S3 28
#define sensorOut1 29

#define S00 37
#define S11 36
#define S22 35
#define S33 34
#define sensorOut2 33

int redFrequency = 0;
int redFrequency2 = 0;
int greenFrequency = 0;
int greenFrequency2 = 0;
int blueFrequency = 0;
int blueFrequency2 = 0;

int redColor = 0;
int redColor2 = 0;
int greenColor = 0;
int greenColor2 = 0;
int blueColor = 0;
int blueColor2 = 0;

String currentColor = "               ";
String leftSensorColor = "";
String rightSensorColor = "";
///////////////////////////////////

/* DC Motor Var */
```

```
// Uncomment if no servo motors
// int EN1 = 2;
// int EN2 = 3;
// int EN3 = 12;
// int EN4 = 24;
// int IN1 = 4;
// int IN2 = 5;
// int IN3 = 6;
// int IN4 = 9;

const int buttonPin = 10;
const int buttonPin1 = 11;

int buttonState = 0;
int buttonState1 = 0;

int pwmSpeed = 0;
//int reversePWM = 100;
/////////////////////////////////

/* Line Edge Sensor */
// Back up Line Edge Detector
// #define leftSide A9
// #define rightSide A8
// int leftEdge = 0;
// int rightEdge = 0;
/////////////////////////////////

/* Extra Features */
//Motion Sensor//
int motionState = LOW;
int motionBehind = 31;
int sensorOutput = 0;
bool motionChecker = false;
//////////////////////////////////
```

```
void setup() {
  HWSERIAL.begin(9600, SERIAL_8N1); //Sets RX/TX baud for reading from HC05
  Serial.begin(9600);   //Sets the baud for serial data transmission
  pinMode(ledPin, OUTPUT);  //Sets digital pin 13 as output pin

  pinMode(S0, OUTPUT);
  pinMode(S1, OUTPUT);
  pinMode(S2, OUTPUT);
  pinMode(S3, OUTPUT);
  pinMode(S00, OUTPUT);
  pinMode(S11, OUTPUT);
  pinMode(S22, OUTPUT);
  pinMode(S33, OUTPUT);
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);
  digitalWrite(S00, HIGH);
  digitalWrite(S11, LOW);
  pinMode(sensorOut1, INPUT);
  pinMode(sensorOut2, INPUT);

  // Uncomment if using dc motors
  // pinMode(EN1, OUTPUT);
  // pinMode(EN2, OUTPUT);
  // pinMode(IN1, OUTPUT);
  // pinMode(IN2, OUTPUT);
  // pinMode(IN3, OUTPUT);
  // pinMode(IN4, OUTPUT);
  pinMode(buttonPin, INPUT);
  pinMode(buttonPin1, INPUT);
  // pinMode(buttonPin2, INPUT);
  // pinMode(buttonPin3, INPUT);
  pinMode(motionBehind, INPUT);
  pinMode(leftSide, INPUT);
  pinMode(rightSide, INPUT);
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  //pinMode(laser, OUTPUT);

  servoL1.attach(EN1);
  servoR1.attach(EN2);
  servoL2.attach(EN3);
  servoR2.attach(EN4);

  lcd.begin(16, 2);
}

void loop() {
  digitalWrite(laser, HIGH);
  int i;
  checkButton();
  if (buttonState == HIGH) {
    reverse();
    delay(1000);
  } else {
    //stopMoving();
    readAndStoreInstructions();
    detectColor2();
    lightUpLed();
    displayColor();
    for (i = 0; i < MAX_SIZE; i++) {
      if (colors[i] == "")
        break;
      moveCar(i);
    }
  }
}

/* Reading from Bluetooth */
```

```
void readAndStoreInstructions() {
  while (HWSERIAL.available() > 0) {
    analogWrite(13, 255);
    data = HWSERIAL.readStringUntil(';');
    Serial.print("Instructions ");
    Serial.print(count);
    Serial.print(": ");
    Serial.println(data);
    array[i] = data;
    Serial.print("Array position[");
    Serial.print(i);
    Serial.print("]: ");
    Serial.println(array[i]);
    data.toCharArray(chars, MAX_SIZE * 100);
    token = strtok(chars, ":");
    int j = 0;
    while (token != NULL) {
      strings[j] = token;
      if (strings[j] == strings[0]) {
        Serial.print("Color: ");
        Serial.print(strings[j]);
        colors[i] = strings[j];
        Serial.print(" --> colors[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(colors[i]);
      } else if (strings[j] == strings[1]) {
        Serial.print("Direction: ");
        Serial.print(strings[j]);
        directions[i] = strings[j];
        Serial.print(" --> directions[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(directions[i]);
      } else if (strings[j] == strings[2]) {
        Serial.print("Speed: ");
        Serial.print(strings[j]);
        speeds[i] = strings[j];
        Serial.print(" --> speeds[");
        Serial.print(i);
        Serial.print("]: ");
        Serial.println(speeds[i]);
      }
      else {
        break;
      }
      token = strtok(NULL, ":");  // takes a list of delimiters
      j++;
    }
    count++;
    i++;

    analogWrite(13, 0);
    //delay(250);
  }
}

/* Color Sensor */
void readRedColor() {
  // Setting RED (R) filtered photodiodes to be read
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  digitalWrite(S22, LOW);
  digitalWrite(S33, LOW);

  redFrequency = pulseIn(sensorOut1, LOW);
  redFrequency2 = pulseIn(sensorOut2, LOW);
  redColor = map(redFrequency, 91, 194, 255, 0);
  redColor2 = map(redFrequency2, 80, 197, 255, 0);
  //redColor = map(redFrequency, 132, 194, 255, 0);
  //redColor2 = map(redFrequency2, 83, 164, 255, 0);

  Serial.print("R = ");
  Serial.print(redColor);
  Serial.print(" R = ");
  Serial.print(redColor2);
}

void readGreenColor() {
  // Setting GREEN (G) filtered photodiodes to be read
  digitalWrite(S2, HIGH);
  digitalWrite(S3, HIGH);
  digitalWrite(S22, HIGH);
  digitalWrite(S33, HIGH);

  greenFrequency = pulseIn(sensorOut1, LOW);
  greenFrequency2 = pulseIn(sensorOut2, LOW);
  greenColor = map(greenFrequency, 166, 335, 255, 0);
  greenColor2 = map(greenFrequency2, 143, 316, 255, 0);
  //greenColor = map(greenFrequency, 177, 335, 255, 0);
  //greenColor2 = map(greenFrequency2, 154, 316, 255, 0);

  // Printing the GREEN (G) value
  Serial.print(" G = ");
  Serial.print(greenColor);
  Serial.print(" G = ");
  Serial.print(greenColor2);
}

void readBlueColor() {
  digitalWrite(S2, LOW);
  digitalWrite(S3, HIGH);
  digitalWrite(S22, LOW);
  digitalWrite(S33, HIGH);
  blueFrequency = pulseIn(sensorOut1, LOW);
  blueFrequency2 = pulseIn(sensorOut2, LOW);
  blueColor = map(blueFrequency, 122, 250, 255, 0);
  blueColor2 = map(blueFrequency2, 105, 220, 255, 0);
  //blueColor = map(blueFrequency, 129, 240, 255, 0);
  //blueColor2 = map(blueFrequency2, 114 , 233, 255, 0);

  // Printing the BLUE (B) value
  Serial.print(" B = ");
  Serial.println(blueColor);
  Serial.print(" B = ");
  Serial.println(blueColor2);
}

void detectColor2() {
  readRedColor();
  readGreenColor();
  readBlueColor();
  // left color sensor
  if (redColor < 0 && greenColor < 0 && blueColor < 0) {
    leftSensorColor = "Black";
  } else if (redColor >= 240 && greenColor > 240 && blueColor > 240) {
    leftSensorColor = "White";
  } else if (redColor > greenColor && redColor > blueColor) {
    leftSensorColor = "Red";
  } else if (greenColor > redColor && greenColor > blueColor) {
    leftSensorColor = "Green";
  } else {
    leftSensorColor = "Blue";
  }

  // right color sensor
  if (redColor2 < 0 && greenColor2 < 0 && blueColor2 < 0) {
    rightSensorColor = "Black";
  } else if (redColor2 >= 240 && greenColor2 > 240 && blueColor2 > 240) {
    rightSensorColor = "White";
  } else if (redColor2 > greenColor2 && redColor2 > blueColor2) {
    rightSensorColor = "Red";
  } else if (greenColor2 > redColor2 && greenColor2 > blueColor2) {
    rightSensorColor = "Green";
  } else {
    rightSensorColor = "Blue";
  }
  //Serial.print(leftSensorColor);
  //Serial.print(rightSensorColor);
  setCurrentColor();
}

void setCurrentColor() {
  if (leftSensorColor == "Black" && rightSensorColor == "Black")
    currentColor = "Black";
  else if (leftSensorColor == "White" && rightSensorColor == "White")
    currentColor = "White";
  else if (leftSensorColor == "Red" && rightSensorColor == "Red") {
    currentColor = "Red";
  } else if (leftSensorColor == "Green" && rightSensorColor == "Green") {
    currentColor = "Green";
  } else if (leftSensorColor == "Blue" && rightSensorColor == "Blue") {
    currentColor = "Blue";
  } else {
    currentColor = "No Color";
  }
}

/* void detectColor() {
  readRedColor();
  readGreenColor();
  readBlueColor();
  if (redColor < 0 && greenColor < 0 && blueColor < 0) {
    leftSensorColor = "Black";
    if (redColor2 < 0 && greenColor2 < 0 && blueColor2 < 0) {
      rightSensorColor = "Black";
      currentColor = "Black";
      Serial.print(currentColor);
      Serial.println(" detected!");
    }
  } else if (redColor >= 240 && greenColor > 240 && blueColor > 240) {
    leftSensorColor = "White";
    if (redColor2 >= 240 && greenColor2 > 240 && blueColor2 > 240) {
      rightSensorColor = "White";
      currentColor = "White";
      Serial.print(currentColor);
      Serial.println(" detected!");
    }
  } else if (redColor > greenColor && redColor > blueColor) {
    leftSensorColor = "Red";
    if (redColor2 > greenColor2 && redColor2 > blueColor2) {
      rightSensorColor = "Red";
      currentColor = "Red";
      Serial.print(currentColor);
      Serial.println(" detected!");
    }
  } else if (greenColor > redColor && greenColor > blueColor) {
    leftSensorColor = "Green";
    if (greenColor2 > redColor2 && greenColor2 > blueColor2) {
      rightSensorColor = "Green";
      currentColor = "Green";
      Serial.print(currentColor);
      Serial.println(" detected!");
    }
  } else if (blueColor > redColor && blueColor > greenColor) {
    leftSensorColor = "Blue";
    if (blueColor2 > redColor2 && blueColor2 > greenColor2) {
      rightSensorColor = "Blue";
      currentColor = "Blue";
      Serial.print(currentColor);
      Serial.println(" detected!");
    }
  }
  else {
    currentColor = " ";
    Serial.print("No Color");
    Serial.println(" detected!");
  }
  }*/

/* Line Sensor - Exclude */
// void checkForLineEdge() {
//  leftEdge = analogRead(leftSide);
//  rightEdge = analogRead(rightSide);
//}

/* Motion Sensor - Exclude */
// void checkMotion() {
//  sensorOutput = digitalRead(motionBehind);
//  if (sensorOutput == HIGH) {
//    motionState = HIGH;
//    digitalWrite(ledPin, HIGH);
//    Serial.println("Motion detected");
// } else {
//    motionState = LOW;
//    digitalWrite(ledPin, LOW);
//    Serial.println("No motion detected");
//  }
//  delay(10);
//}

/* LCD -Possibly Exclude */
void displayColor() {
  lcd.setCursor(0, 0);
  lcd.print("Detected color:");
  lcd.setCursor(0, 1);
  lcd.print(currentColor);
}
////////////////////////////////////////////

void lightUpLed() {
  //Serial.println(currentColor);
  if (currentColor == "Black") {
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);
  } else if (currentColor == "White") {
    digitalWrite(redPin, HIGH);
    digitalWrite(greenPin, HIGH);
    digitalWrite(bluePin, HIGH);
  } else if (currentColor == "Red") {
    digitalWrite(redPin, HIGH);
  }  else if (currentColor == "Green") {
    digitalWrite(greenPin, HIGH);
  } else if (currentColor == "Blue") {
    digitalWrite(bluePin, HIGH);
  } else {
    digitalWrite(redPin, LOW);
    digitalWrite(greenPin, LOW);
    digitalWrite(bluePin, LOW);
  }
}
////////////////////////////////////////////

/* Controlling the Motors*/
void checkButton() {
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
  // buttonState1 = digitalRead(buttonPin1);
  // Serial.println(buttonState1);
  // buttonState2 = digitalRead(buttonPin2);
  // Serial.println(buttonState2);
  // buttonState3 = digitalRead(buttonPin3);
  // Serial.println(buttonState3);
  // Serial.println(" ");

  if (buttonState == HIGH)// && buttonState1 == HIGH)
    digitalWrite(ledPin, HIGH);
  //else if (buttonState1 == HIGH)
  //  digitalWrite(ledPin, HIGH);
  // else if (buttonState2 == HIGH)
  //   digitalWrite(ledPin, HIGH);
  // else if (buttonState3 == HIGH)
  //  digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}

void moveCar(int i) {
  carSpeed(i);
  if (currentColor == "Black")
    stopMoving();
  else if (colors[i] == "White")// && currentColor == "White")
    turnDirection(i);
  else if (colors[i] == "Red")// && currentColor == "Red")
    turnDirection(i);
  else if (colors[i] == "Green")// && currentColor == "Green")
    turnDirection(i);
  else if (colors[i] == "Blue")// && currentColor == "Blue")
    turnDirection(i);
}

void turnDirection(int i) {
  if (directions[i] == "Follow") {
    Serial.print("FOLLOW");
    follow(i);
  } else if (directions[i] == "Go Left") {
    Serial.print("LEFT");
    turnLeft();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  } else if (directions[i] == "Go Right") {
    Serial.print("RIGHT");
    turnRight();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  } else if (directions[i] == "Go Straight") {
    Serial.print("STRAIGHT");
    straight();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  }  else if (directions[i] == "U Turn") {
    Serial.print("U-TURN");
    turnAround();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  } else if (directions[i] == "Stop") {
    Serial.print("STOP");
    stopMoving();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  }
}

void follow(int i) {
  if (leftSensorColor != colors[i] && rightSensorColor == colors[i]) {
    followLeft();
  } else if (leftSensorColor == colors[i] && rightSensorColor != colors[i]) {
    followRight();
  } else {
    straight();
    Serial.print(leftSensorColor);
    Serial.print(rightSensorColor);
  }
}

void carSpeed(int i) {
  if (speeds[i] == "Slow") {
    pwmSpeed = 50;
  } else if (speeds[i] == "Cruise") {
    pwmSpeed = 80;
  } else if (speeds[i] == "Fast")
    pwmSpeed = 100;
  else
    stopMoving();
}

/* Directions */
void straight() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(100);
  servoR2.write(100);
}

void reverse() {
  servoL1.write(100);
  servoL2.write(100);
  servoR1.write(80);
  servoR2.write(80);
}

void turnRight() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(100);
  servoR2.write(100);
}

void followLeft() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(100);
  servoR2.write(100);
}

void followRight() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(95);
  servoR2.write(95);
}

void turnLeft() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(95);
  servoR2.write(95);
}

void speedUp() {
  servoL1.write(85);
  servoL2.write(85);
  servoR1.write(95);
  servoR2.write(95);
}

void stopMoving() {
  servoL1.write(90);
  servoL2.write(90);
  servoR1.write(90);
  servoR2.write(90);
}

void turnAround() {
  servoL1.write(80);
  servoL2.write(80);
  servoR1.write(80);
  servoR2.write(80);
}
```