Name: Shaan Yadav
NetID: ay140
Honor Code: *I have adhered to the Duke Community Standard in completing this assignment.*

1. Code for Navigation, Sensing, Transmitting, Receiving, Displaying and Analyzing/Calculating

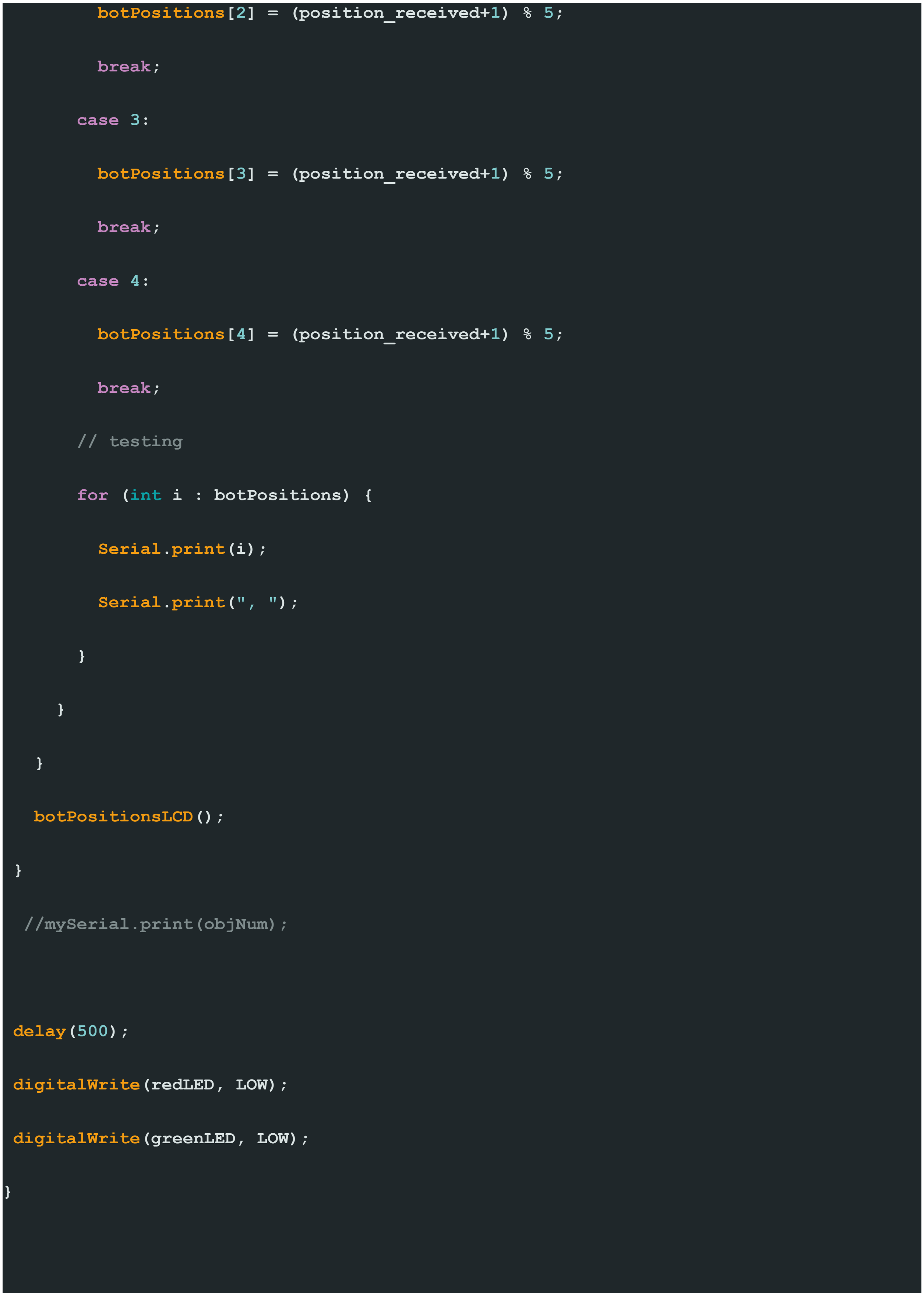
```
//Pins for QTI connections on board

#define leftQTI 51

#define middleQTI 53

#define rightQTI 52

#define TxPin 14



#include <Servo.h>  // Include servo library



//renaming serials

#define LocalBot Serial

#define XBee Serial2



char val = 0;  // variable to store the data from the serial port

int len = 12;



Servo servoLeft;  // Declare left servo signal

Servo servoRight;



#include <Wire.h>              // I2C library, required for MLX90614

#include <SparkFunMLX90614.h>  //Click here to get the library:
http://librarymanager/All#Qwiic_IR_Thermometer by SparkFun
```

```
#include <SoftwareSerial.h>

//LCD Screen
SoftwareSerial mySerial = SoftwareSerial(255, TxPin);

// Define pins for built-in RGB LED
#define redpin 45
#define greenpin 46
#define bluepin 44

#define redLED 2
#define greenLED 3
#define blueLED 4
#define yellowLED 5

int hashCount = 0;
int reds[5] = { 0, 0, 255, 255, 100 };
int greens[5] = { 255, 0, 0, 255, 255 };
int blues[5] = { 255, 255, 255, 0, 0 };

// end code
int botPositions[5] = { 0, 0, 0, 0, 0 };
int countFinalHashes=0;
int location = 0;
bool normal = true;
int runningNum = 0;
int objNum = 0;

//code for transmission and receiving data
const int squadron_shift = 97;
int myPosition = 0;

void setup() {
 LocalBot.begin(9600);  //start the serial monitor so we can view the output
 Serial1.begin(9600);   // connect to the serial port for the RFID reader
 XBee.begin(9600);      // initialize Xbee Tx/Rx

 servoLeft.attach(12);   // Attach left signal to P13
 servoRight.attach(11);  // Attach left signal to P12

 servoLeft.writeMicroseconds(1500);   // 1.5 ms stay still sig, pin 13
 servoRight.writeMicroseconds(1500);  // 1.5 ms stay still sig, pin 12

 pinMode(redpin, OUTPUT);
 pinMode(greenpin, OUTPUT);
 pinMode(bluepin, OUTPUT);

 // start with light off

 analogWrite(redpin, 255);

 analogWrite(greenpin, 255);

 analogWrite(bluepin, 255);



 pinMode(redLED, OUTPUT);    //transmit

 pinMode(greenLED, OUTPUT);  //receive

 pinMode(blueLED, OUTPUT);

 pinMode(yellowLED, OUTPUT);



 //LCD setup

 mySerial.begin(9600);

 delay(100);

 mySerial.write(12);  // clear

 delay(10);

 //mySerial.write(22); // no cursor no blink

 delay(10);

 //mySerial.write(17); // backlight

 delay(10);

}




void loop() {

 int lQTI = rcTime(leftQTI);
```

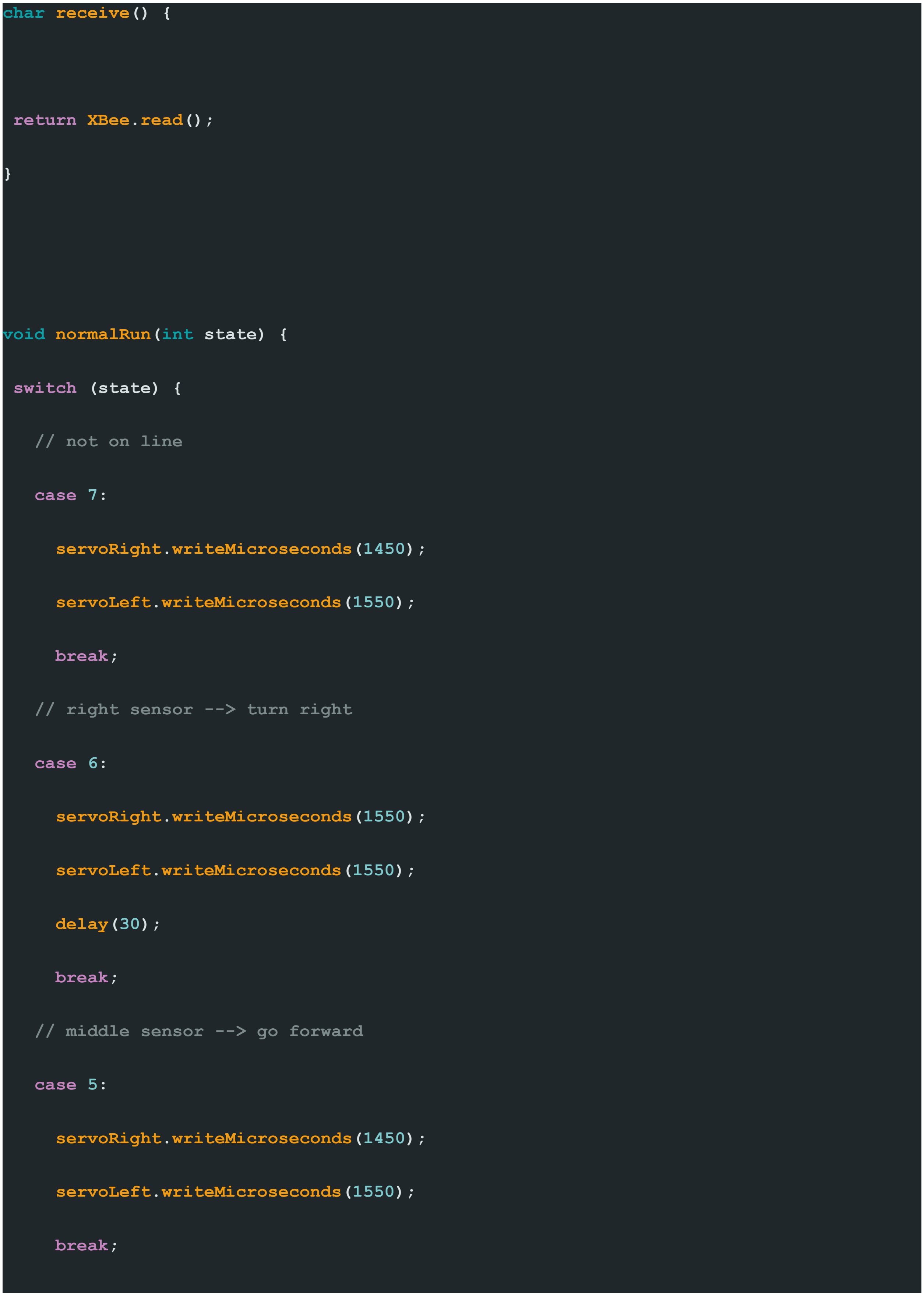
```
  int mQTI = rcTime(middleQTI);
  int rQTI = rcTime(rightQTI);

  int state = 4 * (lQTI < 200) + 2 * (mQTI < 200) + (rQTI < 150);
  // Serial.println(state);


  char incoming = XBee.read();
  if (incoming == '1') {
    runningNum = runningNum + 1;
  }

  if (normal) {
    normalRun(state);
    // Serial.print(111111);
    // location = 2;
    // ending();
  } else {
    ending();
  }

  delay(50);
}
```

```
//Defines funtion 'rcTime' to read value from QTI sensor

// From Ch. 6 Activity 2 of Robotics with the BOE Shield for Arduino

long rcTime(int pin) {

 pinMode(pin, OUTPUT);     // Sets pin as OUTPUT

 digitalWrite(pin, HIGH);  // Pin HIGH

 delay(1);                 // Waits for 1 millisecond

 pinMode(pin, INPUT);      // Sets pin as INPUT

 digitalWrite(pin, LOW);   // Pin LOW

 long time = micros();     // Tracks starting time

 while (digitalRead(pin))

;                       // Loops while voltage is high

 time = micros() - time;  // Calculate decay time

 return time;             // Return decay time

}




//code to use RFID Scanner

void rfidScan() {

 char rfidData[len + 1] = {};

 int get_more = 1;

 int timeoutInt = 0;

 int i = 0;



 while (get_more == 1 && timeoutInt < 200) {
   if (Serial1.available() > 0) {
     val = Serial1.read();

     // Handle unprintable characters
     switch (val) {
       case 0x2: break;                // start of transmission - do not save
       case 0x3: get_more = 0; break;  // end of transmission - done with code
       case 0xA: break;                // line feed - do not save
       case 0xD: break;                // carriage return - do not save
       default:
         rfidData[i] = val;
         i += 1;
         break;  // actual character
     }
   }
   timeoutInt += 1;
   delay(10); //DO NOT REMOVE - NEEDED FOR RFID TO WORK
 }
 LocalBot.println(rfidData);

 if (timeoutInt < 200) {
   char outgoing = rfidData[9];  // Read character
   if (outgoing == 'D') {
     myPosition = 74 + hashCount + 1;
```

```
     botPositions[2] = hashCount + 1;
     location = hashCount + 1;
   }
 }

 //show all bot positions

 botPositionsLCD();

}

void botPositionsLCD() {
 mySerial.write(12);

 for (int i : botPositions) {
   mySerial.print(i);
   mySerial.print(", ");
 }
}

void xbeeTransmit(char charToSend) {
 digitalWrite(redLED, HIGH);  //transmit
 LocalBot.print(charToSend);
 XBee.print(charToSend);  // Send to XBee
 botPositionsLCD();
 delay(100);
 digitalWrite(redLED,LOW);
}

//receive data

void recieveTransmissionAndLED() {

 if (XBee.available()) {
   //this is only code for transmission and receive with 1 other bot
   for (int i = 0; i < 4; i++) {
     digitalWrite(greenLED, HIGH); //
     char incoming = receive();
     int position_received = (int)incoming - squadron_shift;
     int botNumber = position_received / 5;
     switch (botNumber) {
       case 0:
         botPositions[0] = (position_received+1) % 5;
         break;
       case 1:
         botPositions[1] = (position_received+1) % 5;
         break;
       case 2:
```



```
         botPositions[2] = (position_received+1) % 5;

         break;

       case 3:

         botPositions[3] = (position_received+1) % 5;

         break;

       case 4:

         botPositions[4] = (position_received+1) % 5;

         break;

       // testing

       for (int i : botPositions) {

         Serial.print(i);

         Serial.print(", ");

       }

     }

   }

   botPositionsLCD();

 }

  //mySerial.print(objNum);



 delay(500);

 digitalWrite(redLED, LOW);

 digitalWrite(greenLED, LOW);

}
```



```
char receive() {

 return XBee.read();
}


void normalRun(int state) {
 switch (state) {
   // not on line
   case 7:
     servoRight.writeMicroseconds(1450);
     servoLeft.writeMicroseconds(1550);
     break;
   // right sensor --> turn right
   case 6:
     servoRight.writeMicroseconds(1550);
     servoLeft.writeMicroseconds(1550);
     delay(30);
     break;
   // middle sensor --> go forward
   case 5:
     servoRight.writeMicroseconds(1450);
     servoLeft.writeMicroseconds(1550);
     break;
```

```
  // middle + right sensor --> turn right, slight
  case 4:
    servoRight.writeMicroseconds(1550);
    servoLeft.writeMicroseconds(1550);
    delay(40);
    break;
  // left sensor --> turn left
  case 3:
    servoRight.writeMicroseconds(1450);
    servoLeft.writeMicroseconds(1450);
    delay(25);
    break;
  // left + middle sensor --> turn left, slight
  case 1:
    servoRight.writeMicroseconds(1450);
    servoLeft.writeMicroseconds(1450);
    delay(25);
    break;
  // at HASHMARK --> stop, forward
  case 0:

    //mySerial.print("h");

    servoRight.writeMicroseconds(1500);
    servoLeft.writeMicroseconds(1500);

    if (hashCount < 4) {
      rfidScan();

      delay(2000);

      // turn light off
      analogWrite(redpin, 255);
      analogWrite(greenpin, 255);
      analogWrite(bluepin, 255);

      servoRight.writeMicroseconds(1300);
      servoLeft.writeMicroseconds(1700);  // right 13 is forward, left 17 is forward

      delay(500);

      digitalWrite(redLED, LOW);
      digitalWrite(greenLED, LOW);
      digitalWrite(blueLED, LOW);
      digitalWrite(yellowLED, LOW);

      hashCount += 1;
```

```
     } else {

       //hashCount = 0;

       //recieveTransmissionAndLED();

       //xbeeTransmit(myPosition);


       mySerial.write(12);

       mySerial.print(botPositions[2]);


       Serial2.print(1);

       runningNum = runningNum + 1;


       // servoRight.writeMicroseconds(1300);

       // servoLeft.writeMicroseconds(1700);  // right 13 is forward, left 17 is forward


       // delay(500);


       while ((runningNum - objNum) != 4) {


         Serial.print(runningNum);

         Serial.print(location);

         delay(100);


         if (XBee.available()) {
           char incoming = XBee.read();
           if (incoming == '1') {
             runningNum = runningNum + 1;
           }
         }
         //Just creating lag

       }

       ending();
       normal = false;

     }

     break;
   // everything else
   default:
     break;
 }
}

 //VENTILATION SHAFT CODE

void ending(){
 //stops on fifth hash

 servoLeft.writeMicroseconds(1500);

 servoRight.writeMicroseconds(1500);

 //location is the location of your object, or the position you are in the order of bots going to the end

 //int delayNumber = (location*5000)-5000;

 //delay(delayNumber);//should change based on finalhash number to give bots time to move



 //move off of fifth hash to the final line

 moveToFinalLine();



 //when on final line, go to the correct hash

 int finalHash = 5-location;

 moveToFinalHash(finalHash);

 }



void moveToFinalLine(){

 int finalState = 0;



 //moves off of fifth hash, stops at the white portion

 // while(finalState==0||finalState==5){

   servoLeft.writeMicroseconds(1550);

   servoRight.writeMicroseconds(1450);

   delay(1750);
```

```
//   finalState = 7 - (4 * (rcTime(leftQTI) >= 200) + 2 * (rcTime(middleQTI) >= 200) + (rcTime(rightQTI) >= 200));
//   delay(50);
// }

//extraLED(255, 255, 255);
servoLeft.writeMicroseconds(1500);
servoRight.writeMicroseconds(1500);
delay(1000);

//turn right
servoLeft.writeMicroseconds(1550);
servoRight.writeMicroseconds(1550);
delay(600);
servoLeft.writeMicroseconds(1550);
servoRight.writeMicroseconds(1450);
delay(200);

//turns onto the final line
while(finalState!=5 && finalState != 1){
  servoLeft.writeMicroseconds(1550);
  servoRight.writeMicroseconds(1450);
```

```
   finalState = 7 - (4 * (rcTime(leftQTI) >= 200) + 2 * (rcTime(middleQTI) >= 200) + (rcTime(rightQTI) >= 200));

   delay(25);

 }

  //extraLED(0, 255, 0);

}



void moveToFinalHash(int finalHash){

 countFinalHashes = -1;

 while(countFinalHashes<finalHash){

   servoLeft.writeMicroseconds(1550);

   servoRight.writeMicroseconds(1450);

    int finalState = 7 - (4 * (rcTime(leftQTI) >= 200) + 2 * (rcTime(middleQTI) >= 200) + (rcTime(rightQTI) >= 200));

   moveFinal(finalState);

   delay(50);

 }



 servoLeft.writeMicroseconds(1500);

 servoRight.writeMicroseconds(1500);

 //extraLED(0, 0, 0);

  //transmit "done" char

 //transmitChar(-23);//transmit a ".", ascii is 46, must subtract -69 because of method

  delay(100000);
```

```
}

void moveFinal(int finalState){
 switch (finalState) {
   //hashmark
   //colors: red, yellow, green, blue, and purple
   case 0:
     delay(300);
     countFinalHashes=countFinalHashes+1;
     if (countFinalHashes == 0) {
       //xbeeTransmit(46); //send period on first ventilation hashmark
       XBee.print(1);
     }
     break;
   //turn left
   case 1:
     servoLeft.writeMicroseconds(1450);
     servoRight.writeMicroseconds(1450);
     break;
   case 3:
     servoLeft.writeMicroseconds(1450);
     servoRight.writeMicroseconds(1450);
     break;
```

```
//unlikely
// case 2:
//   servoLeft.writeMicroseconds(1500);
//   servoRight.writeMicroseconds(1500);
//   break;

//turn right
case 4:
  servoLeft.writeMicroseconds(1550);
  servoRight.writeMicroseconds(1550);
  break;

case 6:
  servoLeft.writeMicroseconds(1550);
  servoRight.writeMicroseconds(1550);
  break;

//go straight
case 5:
  servoLeft.writeMicroseconds(1550);
  servoRight.writeMicroseconds(1450);
  break;

//white only
```

```
   case 7:



     servoLeft.writeMicroseconds(1550);

     servoRight.writeMicroseconds(1450);

     break;

   }

 }
```

2. Code for Full System Integration

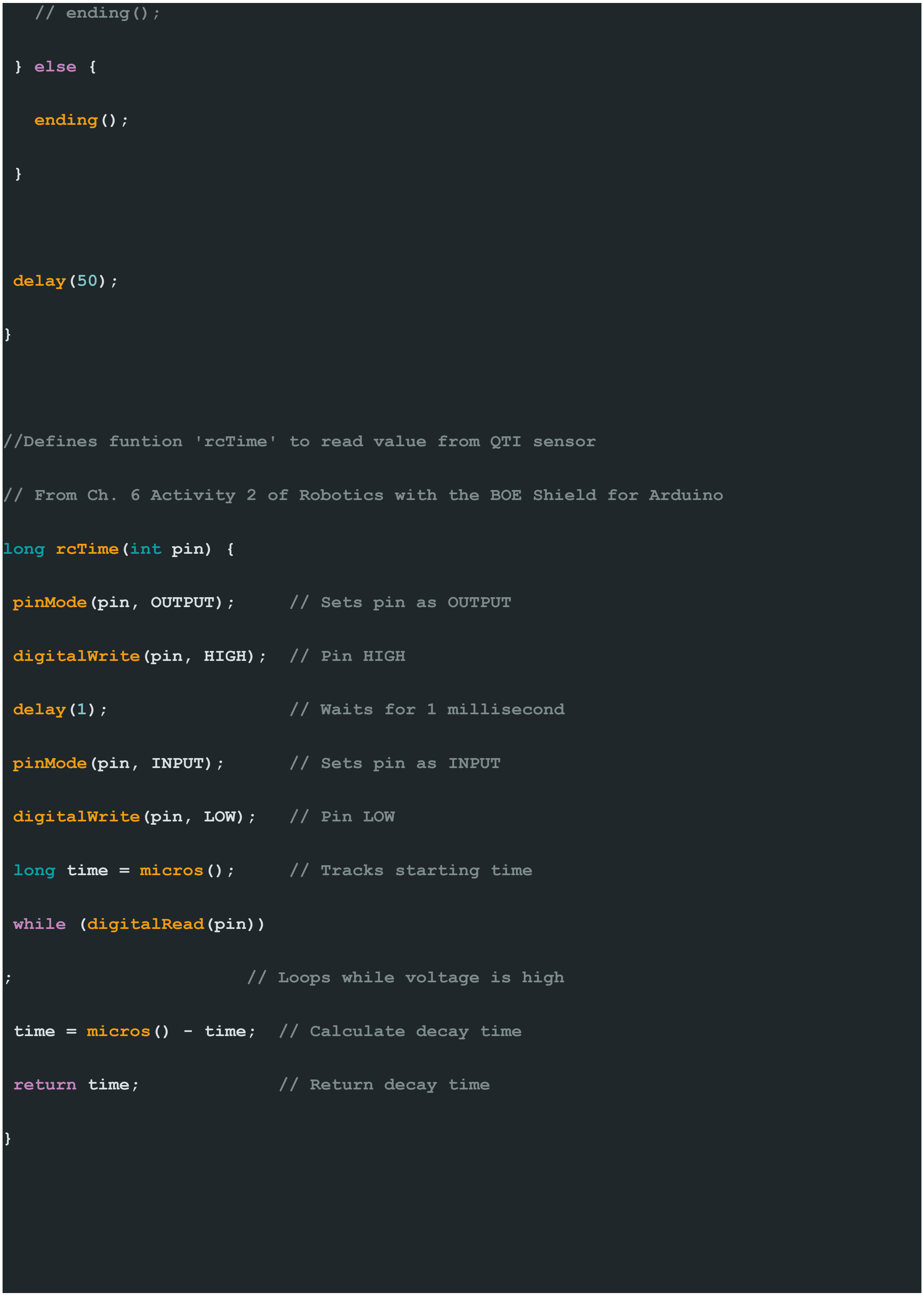
```
//Pins for QTI connections on board

#define leftQTI 51

#define middleQTI 53

#define rightQTI 52

#define TxPin 14



#include <Servo.h>  // Include servo library



//renaming serials

#define LocalBot Serial

#define XBee Serial2



char val = 0;  // variable to store the data from the serial port
```

```
int len = 12;


Servo servoLeft;  // Declare left servo signal

Servo servoRight;


#include <Wire.h>              // I2C library, required for MLX90614

#include <SparkFunMLX90614.h>  //Click here to get the library: http://librarymanager/All#Qwiic_IR_Thermometer by SparkFun


#include <SoftwareSerial.h>


//LCD Screen

SoftwareSerial mySerial = SoftwareSerial(255, TxPin);


// Define pins for built-in RGB LED

#define redpin 45

#define greenpin 46

#define bluepin 44


#define redLED 2

#define greenLED 3

#define blueLED 4

#define yellowLED 5
```

```
int hashCount = 0;
int reds[5] = { 0, 0, 255, 255, 100 };
int greens[5] = { 255, 0, 0, 255, 255 };
int blues[5] = { 255, 255, 255, 0, 0 };

// end code
int botPositions[5] = { 0, 0, 0, 0, 0 };
int countFinalHashes=0;
int location = 0;
bool normal = true;
int runningNum = 0;
int objNum = 0;

//code for transmission and receiving data
const int squadron_shift = 97;
int myPosition = 0;

void setup() {
 LocalBot.begin(9600);  //start the serial monitor so we can view the output
 Serial1.begin(9600);   // connect to the serial port for the RFID reader
 XBee.begin(9600);      // initialize Xbee Tx/Rx

 servoLeft.attach(12);   // Attach left signal to P13
 servoRight.attach(11);  // Attach left signal to P12
```

```
servoLeft.writeMicroseconds(1500);   // 1.5 ms stay still sig, pin 13
servoRight.writeMicroseconds(1500);  // 1.5 ms stay still sig, pin 12

pinMode(redpin, OUTPUT);
pinMode(greenpin, OUTPUT);
pinMode(bluepin, OUTPUT);

// start with light off
analogWrite(redpin, 255);
analogWrite(greenpin, 255);
analogWrite(bluepin, 255);

pinMode(redLED, OUTPUT);    //transmit
pinMode(greenLED, OUTPUT);  //receive
pinMode(blueLED, OUTPUT);
pinMode(yellowLED, OUTPUT);

//LCD setup
mySerial.begin(9600);
delay(100);
mySerial.write(12);  // clear
delay(10);
//mySerial.write(22); // no cursor no blink
```

```
 delay(10);

 //mySerial.write(17); // backlight

 delay(10);

}




void loop() {

 int lQTI = rcTime(leftQTI);

 int mQTI = rcTime(middleQTI);

 int rQTI = rcTime(rightQTI);


 int state = 4 * (lQTI < 200) + 2 * (mQTI < 200) + (rQTI < 150);

 // Serial.println(state);




 char incoming = XBee.read();

 if (incoming == '1') {

   runningNum = runningNum + 1;

 }


 if (normal) {

   normalRun(state);

   // Serial.print(111111);

   // location = 2;
```



```
   // ending();
 } else {
   ending();
 }


 delay(50);
}


//Defines funtion 'rcTime' to read value from QTI sensor
// From Ch. 6 Activity 2 of Robotics with the BOE Shield for Arduino
long rcTime(int pin) {
 pinMode(pin, OUTPUT);     // Sets pin as OUTPUT
 digitalWrite(pin, HIGH);  // Pin HIGH
 delay(1);                 // Waits for 1 millisecond
 pinMode(pin, INPUT);      // Sets pin as INPUT
 digitalWrite(pin, LOW);   // Pin LOW
 long time = micros();     // Tracks starting time
 while (digitalRead(pin))
;                      // Loops while voltage is high
 time = micros() - time;  // Calculate decay time
 return time;             // Return decay time
}
```

```
//code to use RFID Scanner
void rfidScan() {
 char rfidData[len + 1] = {};
 int get_more = 1;
 int timeoutInt = 0;
 int i = 0;

 while (get_more == 1 && timeoutInt < 200) {
   if (Serial1.available() > 0) {
     val = Serial1.read();

     // Handle unprintable characters
     switch (val) {
       case 0x2: break;                // start of transmission - do not save
       case 0x3: get_more = 0; break;  // end of transmission - done with code
       case 0xA: break;                // line feed - do not save
       case 0xD: break;                // carriage return - do not save
       default:
         rfidData[i] = val;
         i += 1;
         break;  // actual character
     }
   }
   timeoutInt += 1;
   delay(10); //DO NOT REMOVE - NEEDED FOR RFID TO WORK
 }
 LocalBot.println(rfidData);

 if (timeoutInt < 200) {
   char outgoing = rfidData[9];  // Read character
   if (outgoing == 'D') {
     myPosition = 74 + hashCount + 1;
     botPositions[2] = hashCount + 1;
     location = hashCount + 1;
   }
 }

 //show all bot positions

 botPositionsLCD();

}

void botPositionsLCD() {
 mySerial.write(12);

 for (int i : botPositions) {
   mySerial.print(i);
   mySerial.print(", ");
 }
}

void xbeeTransmit(char charToSend) {
 digitalWrite(redLED, HIGH);  //transmit
 LocalBot.print(charToSend);
 XBee.print(charToSend);  // Send to XBee
 botPositionsLCD();
 delay(100);
 digitalWrite(redLED,LOW);
}

//receive data

void recieveTransmissionAndLED() {

 if (XBee.available()) {
   //this is only code for transmission and receive with 1 other bot
   for (int i = 0; i < 4; i++) {
     digitalWrite(greenLED, HIGH); //
     char incoming = receive();
     int position_received = (int)incoming - squadron_shift;
     int botNumber = position_received / 5;
```

```
  switch (botNumber) {

    case 0:

      botPositions[0] = (position_received+1) % 5;

      break;

    case 1:

      botPositions[1] = (position_received+1) % 5;

      break;

    case 2:

      botPositions[2] = (position_received+1) % 5;

      break;

    case 3:

      botPositions[3] = (position_received+1) % 5;

      break;

    case 4:

      botPositions[4] = (position_received+1) % 5;

      break;

    // testing

    for (int i : botPositions) {

      Serial.print(i);

      Serial.print(", ");

    }

  }

}

botPositionsLCD();
```

```
 }
  //mySerial.print(objNum);

 delay(500);
 digitalWrite(redLED, LOW);
 digitalWrite(greenLED, LOW);
}

char receive() {

 return XBee.read();
}


void normalRun(int state) {
 switch (state) {
   // not on line
   case 7:
     servoRight.writeMicroseconds(1450);
     servoLeft.writeMicroseconds(1550);
     break;
   // right sensor --> turn right
   case 6:
     servoRight.writeMicroseconds(1550);
     servoLeft.writeMicroseconds(1550);
     delay(30);
     break;
   // middle sensor --> go forward
   case 5:
     servoRight.writeMicroseconds(1450);
     servoLeft.writeMicroseconds(1550);
     break;
   // middle + right sensor --> turn right, slight
   case 4:
     servoRight.writeMicroseconds(1550);
     servoLeft.writeMicroseconds(1550);
     delay(40);
     break;
   // left sensor --> turn left
   case 3:
     servoRight.writeMicroseconds(1450);
     servoLeft.writeMicroseconds(1450);
     delay(25);
     break;
   // left + middle sensor --> turn left, slight
   case 1:
     servoRight.writeMicroseconds(1450);
     servoLeft.writeMicroseconds(1450);
```

```
    delay(25);
    break;
  // at HASHMARK --> stop, forward
  case 0:

    //mySerial.print("h");

    servoRight.writeMicroseconds(1500);
    servoLeft.writeMicroseconds(1500);

    if (hashCount < 4) {
      rfidScan();

      delay(2000);

      // turn light off
      analogWrite(redpin, 255);
      analogWrite(greenpin, 255);
      analogWrite(bluepin, 255);

      servoRight.writeMicroseconds(1300);
      servoLeft.writeMicroseconds(1700);  // right 13 is forward, left 17 is forward

      delay(500);
      digitalWrite(redLED, LOW);
      digitalWrite(greenLED, LOW);
      digitalWrite(blueLED, LOW);
      digitalWrite(yellowLED, LOW);

      hashCount += 1;

    } else {
      //hashCount = 0;
      //recieveTransmissionAndLED();
      //xbeeTransmit(myPosition);

      mySerial.write(12);
      mySerial.print(botPositions[2]);

      Serial2.print(1);
      runningNum = runningNum + 1;

      // servoRight.writeMicroseconds(1300);
      // servoLeft.writeMicroseconds(1700);  // right 13 is forward, left 17 is
forward

      // delay(500);
      while ((runningNum - objNum) != 4) {

        Serial.print(runningNum);
        Serial.print(location);
        delay(100);

        if (XBee.available()) {
          char incoming = XBee.read();
          if (incoming == '1') {
            runningNum = runningNum + 1;
          }
        }
        //Just creating lag

      }

      ending();
      normal = false;

    }

    break;
  // everything else
```

```
    default:
      break;
  }
}


 //VENTILATION SHAFT CODE


void ending(){
 //stops on fifth hash
 servoLeft.writeMicroseconds(1500);
 servoRight.writeMicroseconds(1500);
 //location is the location of your object, or the position you are in the order of bots going to the end
 //int delayNumber = (location*5000)-5000;
 //delay(delayNumber);//should change based on finalhash number to give bots time to move


 //move off of fifth hash to the final line
 moveToFinalLine();


 //when on final line, go to the correct hash
 int finalHash = 5-location;
 moveToFinalHash(finalHash);
 }
```

```
void moveToFinalLine(){
 int finalState = 0;

 //moves off of fifth hash, stops at the white portion
 // while(finalState==0||finalState==5){
   servoLeft.writeMicroseconds(1550);
   servoRight.writeMicroseconds(1450);
   delay(1750);

 //   finalState = 7 - (4 * (rcTime(leftQTI) >= 200) + 2 * (rcTime(middleQTI) >= 200) + (rcTime(rightQTI) >= 200));
 //   delay(50);
 // }

 //extraLED(255, 255, 255);
 servoLeft.writeMicroseconds(1500);
 servoRight.writeMicroseconds(1500);
 delay(1000);

 //turn right
 servoLeft.writeMicroseconds(1550);
 servoRight.writeMicroseconds(1550);
 delay(600);
 servoLeft.writeMicroseconds(1550);
 servoRight.writeMicroseconds(1450);
 delay(200);


 //turns onto the final line
 while(finalState!=5 && finalState != 1){
   servoLeft.writeMicroseconds(1550);
   servoRight.writeMicroseconds(1450);


   finalState = 7 - (4 * (rcTime(leftQTI) >= 200) + 2 * (rcTime(middleQTI) >= 200) + (rcTime(rightQTI) >= 200));
   delay(25);
 }
  //extraLED(0, 255, 0);
}


void moveToFinalHash(int finalHash){
 countFinalHashes = -1;
 while(countFinalHashes<finalHash){
   servoLeft.writeMicroseconds(1550);
   servoRight.writeMicroseconds(1450);
    int finalState = 7 - (4 * (rcTime(leftQTI) >= 200) + 2 * (rcTime(middleQTI) >= 200) + (rcTime(rightQTI) >= 200));
   moveFinal(finalState);
   delay(50);
 }
 servoLeft.writeMicroseconds(1500);

 servoRight.writeMicroseconds(1500);

 //extraLED(0, 0, 0);

  //transmit "done" char

 //transmitChar(-23);//transmit a ".", ascii is 46, must subtract -69 because of
method

  delay(100000);

}



void moveFinal(int finalState){

 switch (finalState) {

   //hashmark

   //colors: red, yellow, green, blue, and purple

   case 0:

     delay(300);

     countFinalHashes=countFinalHashes+1;

     if (countFinalHashes == 0) {

       //xbeeTransmit(46); //send period on first ventilation hashmark

       XBee.print(1);

     }

     break;

   //turn left

   case 1:
     servoLeft.writeMicroseconds(1450);
     servoRight.writeMicroseconds(1450);
     break;
   case 3:
     servoLeft.writeMicroseconds(1450);
     servoRight.writeMicroseconds(1450);
     break;

   //unlikely
   // case 2:
   //   servoLeft.writeMicroseconds(1500);
   //   servoRight.writeMicroseconds(1500);
   //   break;

   //turn right
   case 4:
     servoLeft.writeMicroseconds(1550);
     servoRight.writeMicroseconds(1550);
     break;

   case 6:
     servoLeft.writeMicroseconds(1550);
     servoRight.writeMicroseconds(1550);
     break;
   //go straight
   case 5:
     servoLeft.writeMicroseconds(1550);
     servoRight.writeMicroseconds(1450);
     break;

   //white only
   case 7:

     servoLeft.writeMicroseconds(1550);
     servoRight.writeMicroseconds(1450);
     break;
   }
 }
```

3. Reflection Paragraph

This lab was not too difficult technically, but was very tedious. It involved a lot of sharing code and discussing with other groups which was a very important learning experience. We could no longer sperately develop code for our bots, but had to implement everything in a similar way - both to make the implementation run smoothly and so that we could communicate in a productive way with each other about bugs/errors/places for improvement. Overall a great learning experience.