Lab 4:

Autonomous Car Parking

Matthew Spencer

10/11/2020

CONTENTS

Introduction 2

Materials 2

Theory 2

Procedure 3

Results 6

Conclusion 7

References 7

Appendix 8

INTRODUCTION

This lab sets out to configure and program a CAN Bus circuit to autonomously park an RC Car which has three Arduinos installed, each representing one transmitter, one receiver, and one motor shield node. The receiving Arduino sends movement commands to the RC Car chassis which is powered by the external motor shield, and thus controls the direction of the 4 DC motors attached to each of the 4 wheels on the chassis. The transmitter sends proprietary 1-byte messages controlling the direction of the receiving chassis to move forwards, backwards, and park. All movements are dependent on the distance read from the Ultrasonic Sensor attached to the transmitting Arduino facing the front of the chassis. [1]

MATERIALS

- Arduino Mega 2560 (3)
- CAN Bus Shield (2)
- Jumper Wires (2)
- Arduino IDE
- USB A-Male to B-Male Cord (3)
- RC Car Chassis
- Servo Motor
- Ultrasonic Sensor (HC-SR04)
- Arduino Motor Shield

THEORY

The transmitting (Tx) Arduino is used to send proprietary messages to the receiving (Rx) Arduino over a CAN Bus connection to control the receiver's RC Car chassis through the onboard Motor Shield. The transmitter reads distance data from an on-board ultrasonic sensor while backing up to "find" a parking spot once it determines there is enough space. It then sends 1-byte messages to the receiving Arduino node to either keep moving backwards, or park once a space is found. The receiving Arduino node reads the movement messages, and then powers the third Arduino's motor shield to move the RC chassis in the desired direction, or to stop if parked.

PROCEDURE

Three Arduinos are plugged into the computer by using a USB A to B Male cord, each are to be programmed using the Arduino IDE software environment. One is set up to be a transmitter over a SPI interface, sending data to one receiving Arduino via a CAN Bus connection created by installing the CAN Bus shield Arduino attachment to each Arduino node. This is the same process as discussed in the Lab 1 manual. [2] The third Arduino receives output voltages supplied by the receiving node's output pins, and converts them into the desired movement for the RC chassis's DC motors using an installed motor shield attachment.

The transmitting Arduino sends pre-written CAN messages to the receiving Arduino that tells the RC Car chassis which direction to go after reading information supplied by an attached ultrasonic sensor. The sensor is put onto a separate servo motor attached to the back, allowing it to turn from looking left for a parking spot to backwards.

The sensor turns at the start to initialize the first distance `D1` before beginning to move the chassis backwards. It then reads distances into `D2` while moving, looking for an opening to park when `D2 > D1`. Once the condition is met, the transmitter tells the chassis to stop moving backwards and park into the opening. This is done to simulate autonomous car parking.

To move, the receiving Arduino signals to the motor shield attachment on the third Arduino to turn the DC Motors attached to each of its tires by powering the corresponding pre-programmed pins on the motor shield. The libraries and Arduino code for the receiving Arduino, ultrasonic sensor, and motor shield is provided by the online Canvas lab page. [3]

Once the receiving Arduino receives the transmitted message to park, it then stops the RC chassis and turns it 90 degrees facing the perceived opening by the ultrasonic sensor. The chassis is then told to move backwards into the parking spot and stop indefinitely using the `delay()` [4] function. This completes the autonomous parking simulation.

All Arduino code for each of the three Arduinos can be viewed in the Appendix of this lab document, and the actual circuit configuration of the RC chassis can be viewed in Figure 1 and Figure 2.

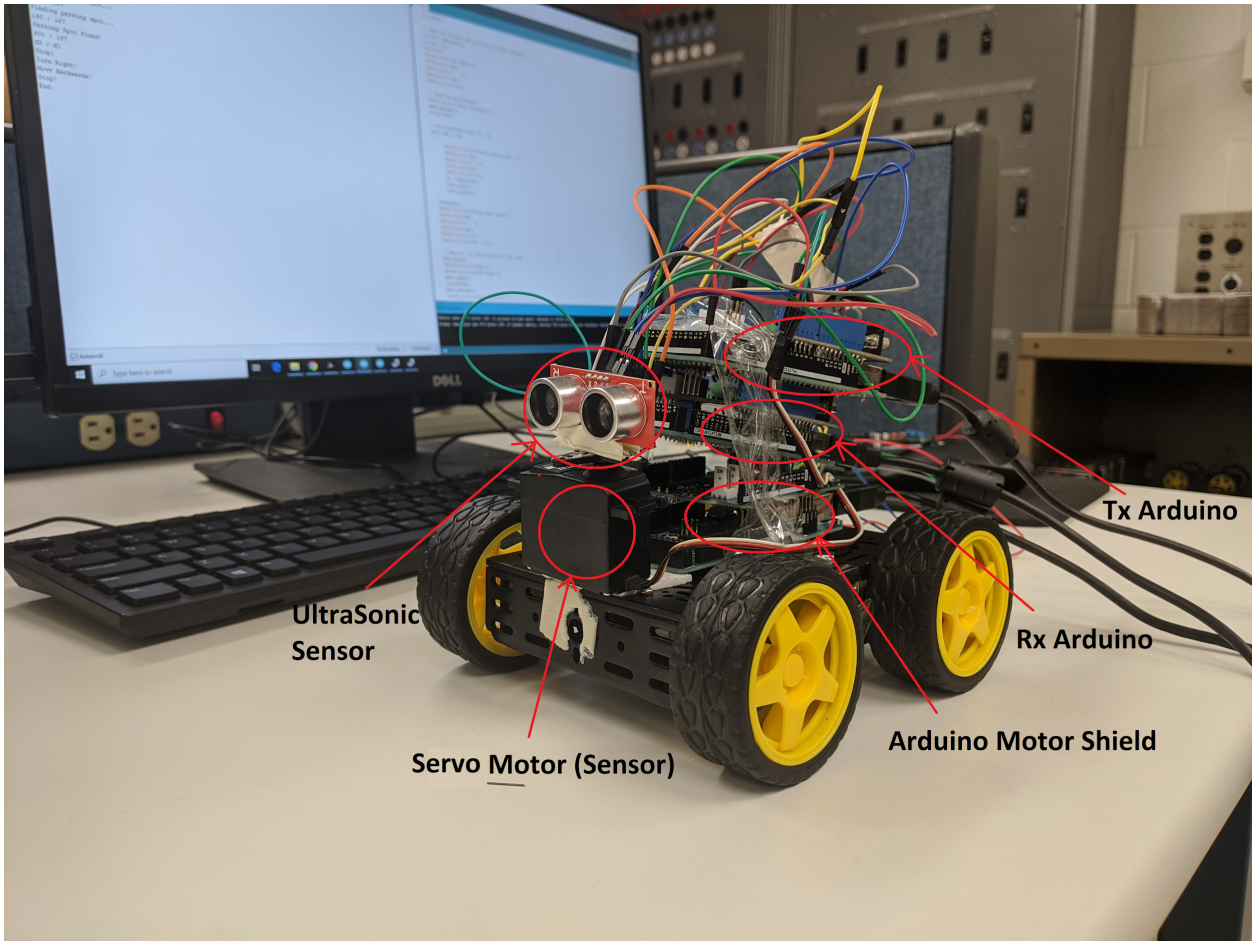


Figure 1: Lab 4 RC Chassis Circuit Configuration (Back)

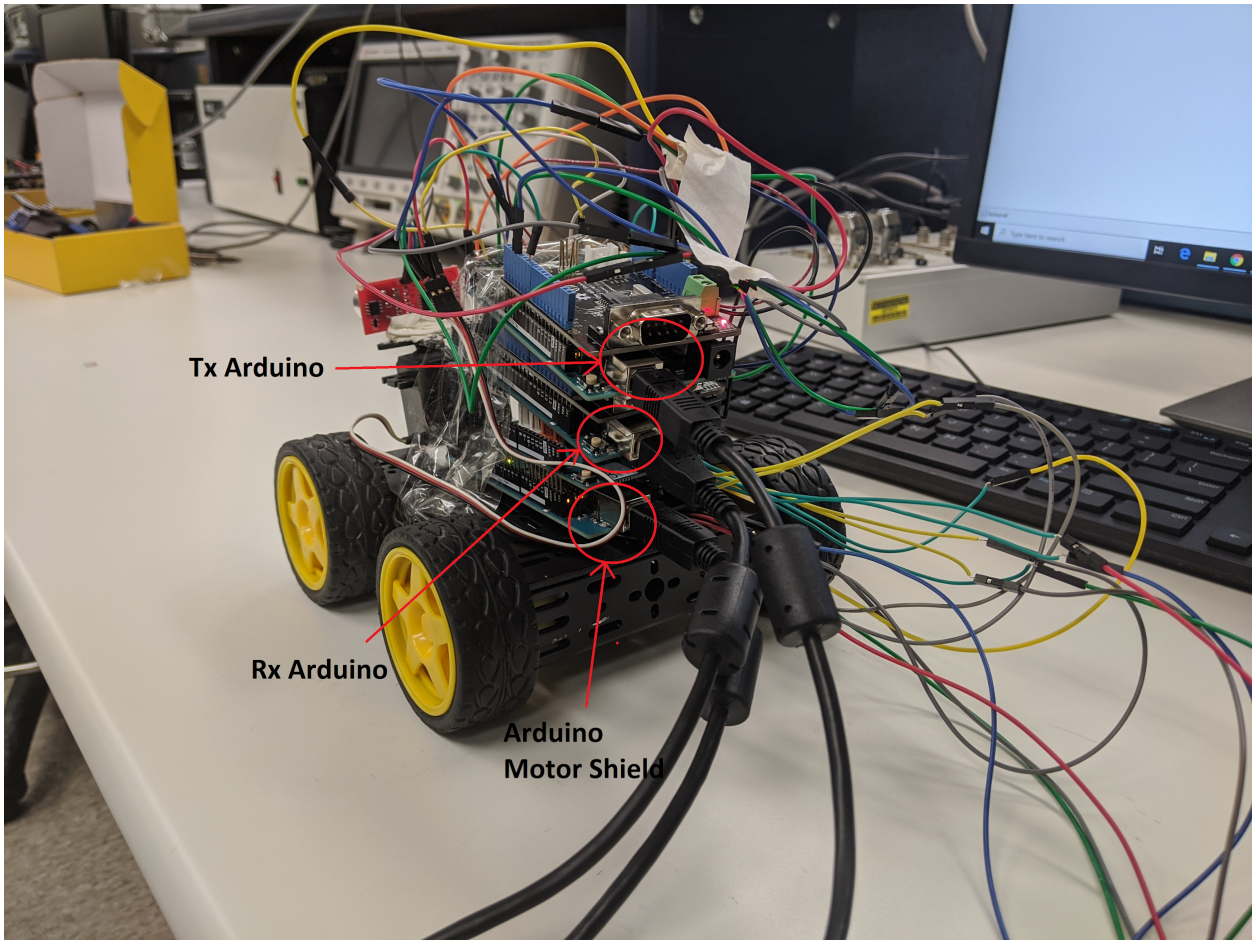


Figure 1: Lab 4 RC Chassis Circuit Configuration (Front)

RESULTS

Figure 3 shows the Tx Arduino Serial Monitor output while the RC chassis autonomously parks.

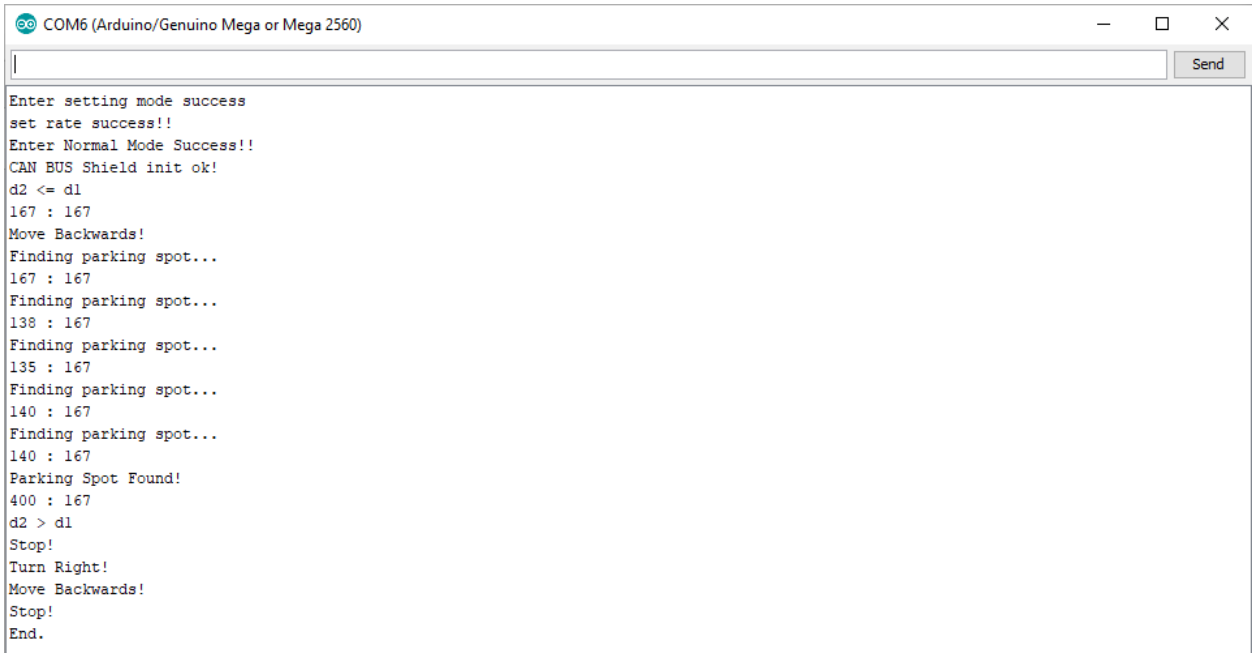


Figure 3: Transmitting Arduino Serial Monitor

The Ultrasonic Sensor successfully read in the first distance, initializing both `D1` and `D2` variables to be 167cm as seen after the statement `d2 <= d1` in Figure 3. The chassis was then told to move backwards to begin looking for a parking spot.

The chassis was told to continue moving backwards repeatedly until `d2 > d1.` The Tx Arduino program "`Send.io`" checked the ultrasonic sensor during movement and displayed the results in Figure 3 in the following format; `d2 : d1`. If `d2` was not greater than `d1` the transmitter successfully told the chassis to continue moving backwards looking for a spot, printing "`Finding parking spot...`" to the terminal.

Once `d2` was greater than `d1`, shown in Figure 3 by having `d2 = 400` and `d1 = 167`, the chassis was told to stop, turn right into the parking spot, move backwards to park, stop again, and then end the program.

CONCLUSION

The serial monitor results in Figure 3 show that the lab experiment to autonomously park was successful. The Tx Arduino was successfully able to transmit CAN Bus messages to the Rx Arduino using real-time data gathered by the ultrasonic sensor to find a parking spot. The Rx Arduino was then successfully able to control the Arduino node installed with the Motor Shield to autonomously park the RC chassis in the found parking spot triggered by the Tx Arduino.

All Arduino code used in this lab can be found in the Appendix at the end of this document.

REFERENCES

1. A. Oliveira, "Lab 4 - Autonomous Car Parking.", *Michigan Technological University*, Houghton, Michigan, United States, 2019.

2. A. Oliveira, "Lab 1 - Introduction to CAN with Arduino Environment", *Michigan Technological University*, Houghton, Michigan, United States, 2019.

3. A. Oliveira, "Lab 4", *Michigan Technological University*, Canvas, 2020, Available: https://mtu.instructure.com/courses/1326161/assignments/8425505, [Accessed Oct 12, 2020].

4. "delay()", Arduino, 2020, [Online], Available: https://www.arduino.cc/reference/en/language/functions/time/delay/, [Accessed Oct 12, 2020].

APPENDIX

```
/*****************************************************************************************
 * Class  : EE5365
 * Proj   : Lab 4
 * Name   : Send.ino
 * Author : Matthew Spencer
 * Desc   : Power an RC car using 2 arduinos over a CAN network and 1 Arduino with
 *          and attached driver. Programs it with an ultrasonic sensor to "look"
 *          for a parking spot.
 * Notes  : Needs Motor controller script and receiver script to work provided
 *          with the lab.
 *****************************************************************************************/
// Include files
#include <Servo.h>

// Initialize pins
#define ECHOPIN 2  // Pin to receive echo pulse
#define TRIGPIN 3  // Pin to send trigger pulse

// Initialize ultrasonic sensor
Servo servo_test;
int reg  = 0;
int val1 = 0;

// demo: CAN-BUS Shield, send data
#include <mcp_can.h>
#include <SPI.h>

// Initializations
int dist1 ;
MCP_CAN CAN(9);
unsigned char stmp0[1] = {0}; // Reverse
unsigned char stmp1[1] = {1}; // forward
unsigned char stmp2[1] = {2}; // left
unsigned char stmp3[1] = {3}; // right
unsigned char stmp4[1] = {4}; // STOP

// Function that reads and returns distance from the ultrasonic sensor
int read_dist()
{
    // Set the trigger pin to low for 2uS
    digitalWrite(TRIGPIN, LOW);
    delayMicroseconds(2);

    // Send a 10uS high to trigger ranging
    digitalWrite(TRIGPIN, HIGH);
    delayMicroseconds(10);

    // Send pin low again
    digitalWrite(TRIGPIN, LOW);

    // Read in times pulse
    int distance = pulseIn(ECHOPIN, HIGH);
    distance= distance/10;
    delay(100);
    return distance;
}

```

Appendix 1: Send.io Part 1

```
58  // Fucntions that move the RC car
59  void move_forward()
60  {
61      CAN.sendMsgBuf(100, 0, 1, stmp1);
62  }
63
64  void move_reverse()
65  {
66      CAN.sendMsgBuf(100, 0, 1, stmp0);
67  }
68
69  void move_left()
70  {
71      CAN.sendMsgBuf(100, 0, 1, stmp2);
72  }
73
74  void move_right()
75  {
76      CAN.sendMsgBuf(100, 0, 1, stmp3);
77  }
78
79  void move_stop()
80  {
81      CAN.sendMsgBuf(100, 0, 1, stmp4);
82  }
83
84  // Functions that move the ultrasonic sensor
85  void turn_left()
86  {
87      servo_test.write(130);
88  }
89
90  void turn_right()
91  {
92      servo_test.write(35);
93  }
94
95  void turn_back()
96  {
97      servo_test.write(110);
98  }
99
100
```

Appendix 2: Send.io Part 2

```
// Set up the Arduino hardware
void setup()
{
    Serial.begin(115200);
    servo_test.attach(6);
    pinMode(ECHOPIN, INPUT);
    pinMode(TRIGPIN, OUTPUT);

    START_INIT:
    // init can bus : baudrate = 500k
    if(CAN_OK == CAN.begin(CAN_500KBPS))
    {
        Serial.println("CAN BUS Shield init ok!");
    }
    else
    {
        Serial.println("CAN BUS Shield init fail");
        Serial.println("Init CAN BUS Shield again");
        delay(100);
        goto START_INIT;
    }
}

// Run the RC car
void loop()
{
    // Turn the sensor to the right to look for a "parking spot"
    turn_back();
    delay(250);
    turn_right();
    delay(250);

    // Read the distance and initialize distance variables
      // parking spot" found when when d2 > d1
    int d1 = read_dist();
    int d2 = d1;
    delay(10);
    Serial.print("d2 <= d1\n");
    Serial.print(d2);
    Serial.print(" : ");
    Serial.print(d1);
    Serial.print("\n");

    // Start moving backwards
    Serial.println("Move Backwards!");
    move_reverse();
    delay(1000);

```

Appendix 3: Send.io Part 3

```
148
149     // Move backwards until d2 > d1 "parking spot" is found
150     while (d2 <= d1)
151     {
152         Serial.println("Finding parking spot...");
153         Serial.print(d2);
154         Serial.print(" : ");
155         Serial.print(d1);
156         Serial.print("\n");
157         d2 = read_dist();
158         turn_right();
159         move_reverse();
160     }
161
162     // Turn the sensor back pointing forward and stop, parking spot found
163     turn_back();
164     Serial.println("Parking Spot Found!");
165     Serial.print(d2);
166     Serial.print(" : ");
167     Serial.print(d1);
168     Serial.print("\n");
169     Serial.println("d2 > d1");
170     Serial.println("Stop!");
171     move_stop();
172
173     // Turn the vehicle into the parking spot and back into it
174     Serial.println("Turn Right!");
175     move_right();
176     delay(270);
177     move_reverse();
178     Serial.println("Move Backwards!");
179     delay(1000);
180     Serial.println("Stop!");
181     move_stop();
182
183     // End the program and stop the car
184     Serial.println("End.");
185     delay(10000000);
186 }
187 /*****************************************************************************************
188   EOF
189 *****************************************************************************************/
```

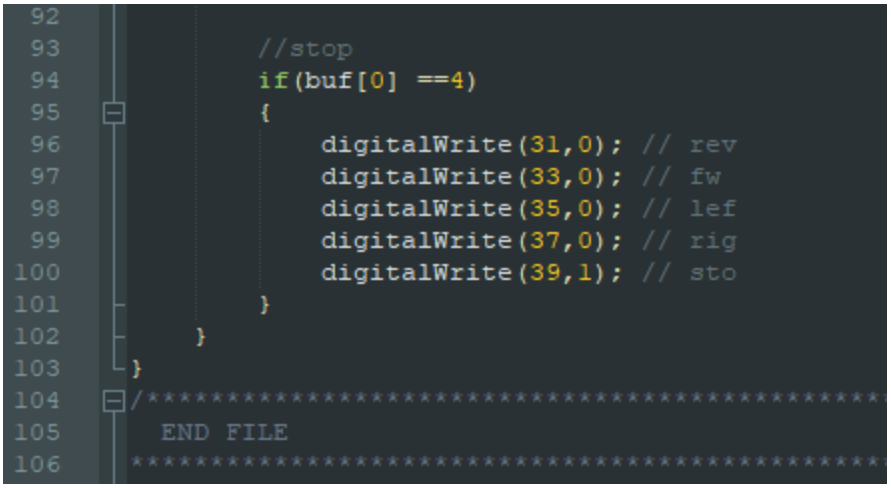
Appendix 4: Send.io Part 4

```
#include <mcp_can.h>
#include <mcp_can_dfs.h>

// demo: CAN-BUS Shield, receive data with check mode
// send data coming to fast, such as less than 10ms, you can use this way
// loovee, 2014-6-13

#include <SPI.h>
#include "mcp_can.h"

// Initialize variables
unsigned char Flag_Recv = 0;
unsigned char len = 0;
unsigned char buf[8];

// Set CS to pin 9
MCP_CAN CAN(9);

void setup()
{
    Serial.begin(115200);
    pinMode(31,OUTPUT);
    pinMode(33,OUTPUT);
    pinMode(35,OUTPUT);
    pinMode(37,OUTPUT);
    pinMode(39,OUTPUT);

START_INIT:
    // init can bus : baudrate = 500k
    if(CAN_OK == CAN.begin(CAN_500KBPS))
    {
        Serial.println("CAN BUS Shield init ok!");
    }
    else
    {
        Serial.println("CAN BUS Shield init fail");
        Serial.println("Init CAN BUS Shield again");
        delay(100);
        goto START_INIT;
    }
}

```

Appendix 5: Receive.io Part 1

```
// determine which direction to move
void loop()
{
    // check if data coming
    if(CAN_MSGAVAIL == CAN.checkReceive())
    {
        // read data,  len: data length, buf: data buf
        CAN.readMsgBuf(&len, buf);
        Serial.print(buf[0]);Serial.print("\t");

        //reverse
        if(buf[0] == 0)
        {
            digitalWrite(31,1); //rev
            digitalWrite(33,0); //fw
            digitalWrite(35,0); //lef
            digitalWrite(37,0); //rig
            digitalWrite(39,0); //sto
        }

        //forward
        if(buf[0] ==1)
        {
            digitalWrite(31,0); // rev
            digitalWrite(33,1); // fw
            digitalWrite(35,0); // lef
            digitalWrite(37,0); // rig
            digitalWrite(39,0); // sto
        }

        //left
        if(buf[0] ==2)
        {
            digitalWrite(31,0); // rev
            digitalWrite(33,0); // fw
            digitalWrite(35,1); // lef
            digitalWrite(37,0); // rig
            digitalWrite(39,0); // sto
        }

        //right
        if(buf[0] ==3)
        {
            digitalWrite(31,0); // rev
            digitalWrite(33,0); // fw
            digitalWrite(35,0); // lef
            digitalWrite(37,1); // rig
            digitalWrite(39,0); // sto
        }

```

Appendix 6: Receive.io Part 2

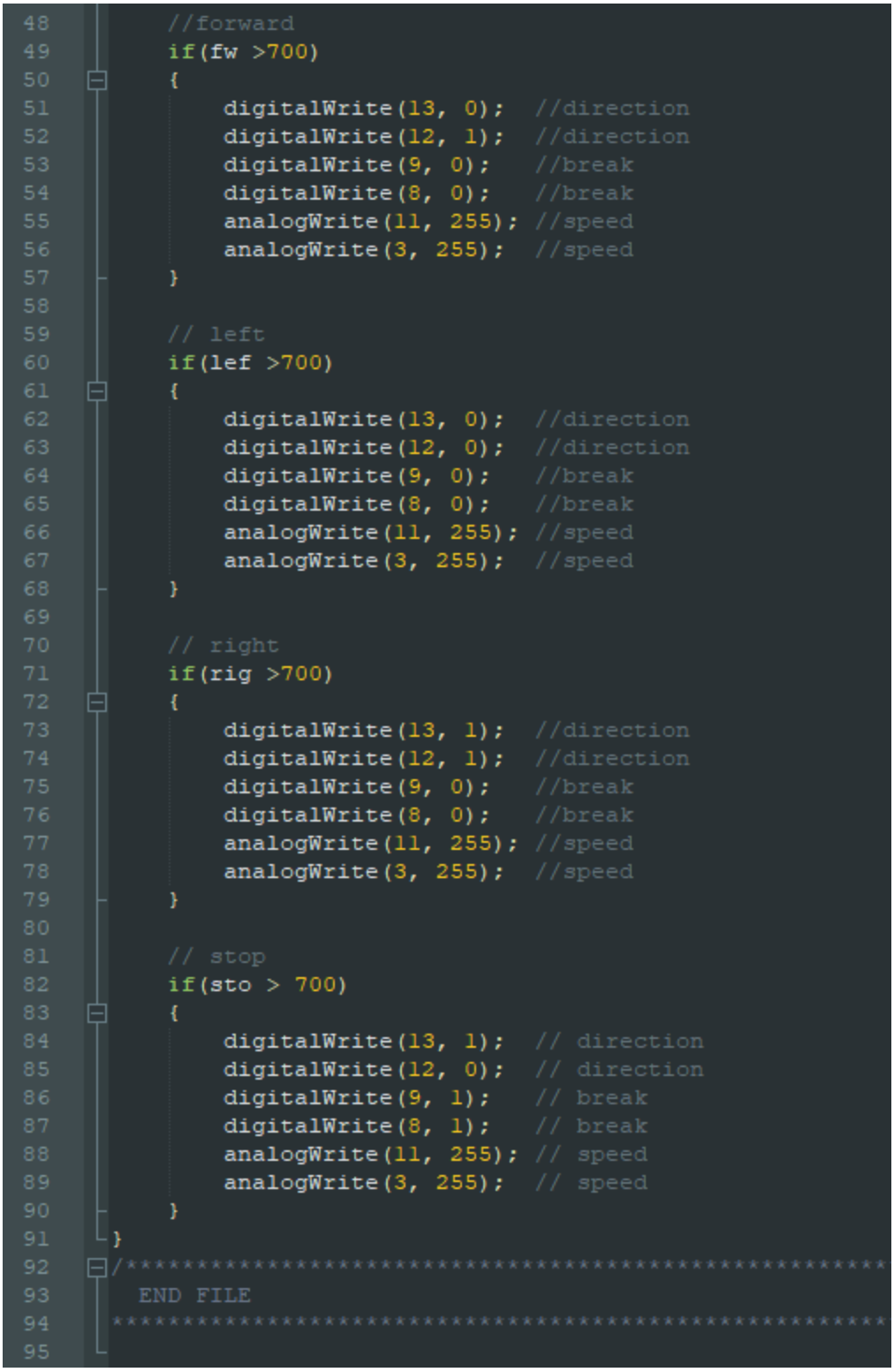


```
92
93          //stop
94          if(buf[0] ==4)
95          {
96              digitalWrite(31,0); // rev
97              digitalWrite(33,0); // fw
98              digitalWrite(35,0); // lef
99              digitalWrite(37,0); // rig
100             digitalWrite(39,1); // sto
101         }
102     }
103 }
104 /*****************************************************
105   END FILE
106 *****************************************************
```

Appendix 7: Receive.io Part 3

```
void setup()
{
    // put your setup code here, to run once:
    pinMode(13, OUTPUT);  // direction
    pinMode(12, OUTPUT);  // direction
    pinMode(8, OUTPUT);   // break
    pinMode(9, OUTPUT);   // break
    pinMode(11, OUTPUT);  // speed
    pinMode(3, OUTPUT);   // speed
    Serial.begin(9600);
}

void loop()
{
    // Read and print values
    int rev = analogRead(A15);
    int  fw = analogRead(A14);
    int lef = analogRead(A13);
    int rig = analogRead(A12);
    int sto = analogRead(A11);
    Serial.print("rev: ");
    Serial.print(rev);
    Serial.print("\n");
    Serial.print("fwd: " );
    Serial.print(fw);
    Serial.print("\n");
    Serial.print("lef: " );
    Serial.print(lef);
    Serial.print("\n");
    Serial.print("rig: " );
    Serial.print(rig);
    Serial.print("\n");
    Serial.print("stp: " );
    Serial.print(sto);
    Serial.print("\n");

    // reverse
    if(rev > 700)
    {
        digitalWrite(13, 1);  //direction
        digitalWrite(12, 0);  //direction
        digitalWrite(9, 0);   //break
        digitalWrite(8, 0);   //break
        analogWrite(11, 255); //speed
        analogWrite(3, 255);  //speed
    }

```

Appendix 8: MotorShield-1.io Part 1



```
48      //forward
49      if(fw >700)
50      {
51          digitalWrite(13, 0);  //direction
52          digitalWrite(12, 1);  //direction
53          digitalWrite(9, 0);   //break
54          digitalWrite(8, 0);   //break
55          analogWrite(11, 255); //speed
56          analogWrite(3, 255);  //speed
57      }
58
59      // left
60      if(lef >700)
61      {
62          digitalWrite(13, 0);  //direction
63          digitalWrite(12, 0);  //direction
64          digitalWrite(9, 0);   //break
65          digitalWrite(8, 0);   //break
66          analogWrite(11, 255); //speed
67          analogWrite(3, 255);  //speed
68      }
69
70      // right
71      if(rig >700)
72      {
73          digitalWrite(13, 1);  //direction
74          digitalWrite(12, 1);  //direction
75          digitalWrite(9, 0);   //break
76          digitalWrite(8, 0);   //break
77          analogWrite(11, 255); //speed
78          analogWrite(3, 255);  //speed
79      }
80
81      // stop
82      if(sto > 700)
83      {
84          digitalWrite(13, 1);  // direction
85          digitalWrite(12, 0);  // direction
86          digitalWrite(9, 1);   // break
87          digitalWrite(8, 1);   // break
88          analogWrite(11, 255); // speed
89          analogWrite(3, 255);  // speed
90      }
91  }
92  /*************************************************************
93    END FILE
94  *************************************************************
95
```

Appendix 9: Motor-Shield-1.io Part 2