Final Project: Bluetooth Controlled Car

Matthew Friedman 861151348
Souradeep Bhattacharya 861105938
EE128 Section: 021

December 15, 2017

Project Description

Summary

Our objective was to design and build a remote controlled car that could be controlled with a smartphone or any other Bluetooth enabled master device.

Requirements/Goals

The car must be able to:

- The car must be able to move in 4 standard directions
  - Forward
  - Backward
  - Left
  - Right
- The car must abe able to move in the 4 combination directions
  - Forward-Left
  - Forward-Right
  - Backward-Left
  - Backward-Right
- The car must be able to turn on and off headlights
- The car must be able to turn on and off a horn
- The car must be able to drive as straight as possible with out the use of external sensors.
- The car's control circuitry must be made as simply as possible with no extraneous parts.

System Design

Block Diagram

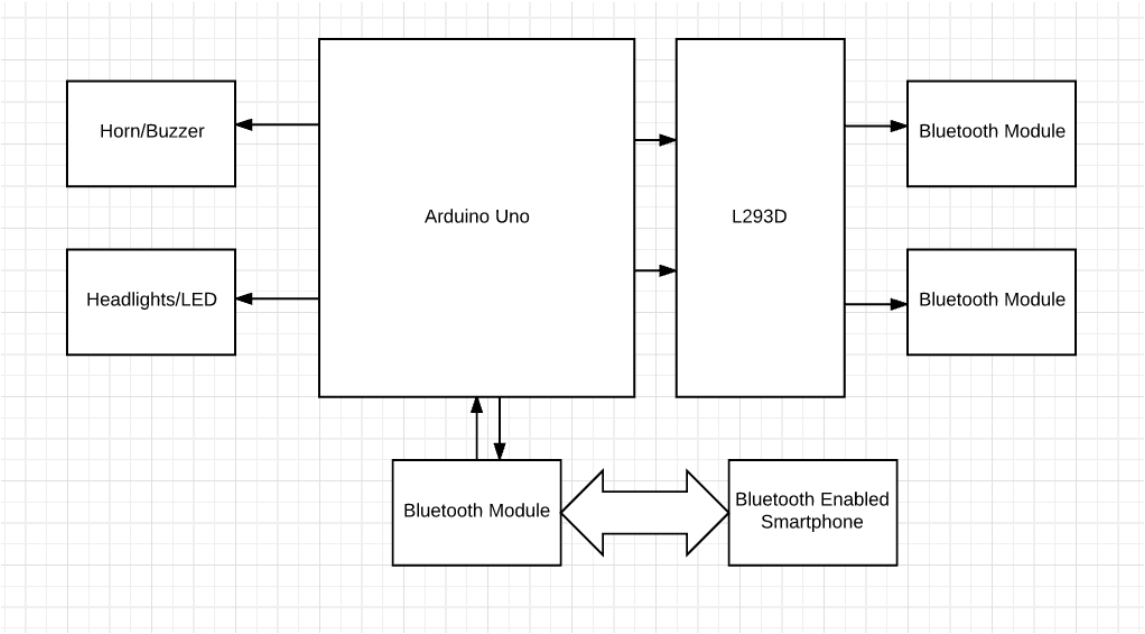

Figure 1: Overall System Block Diagram

Flow Charts

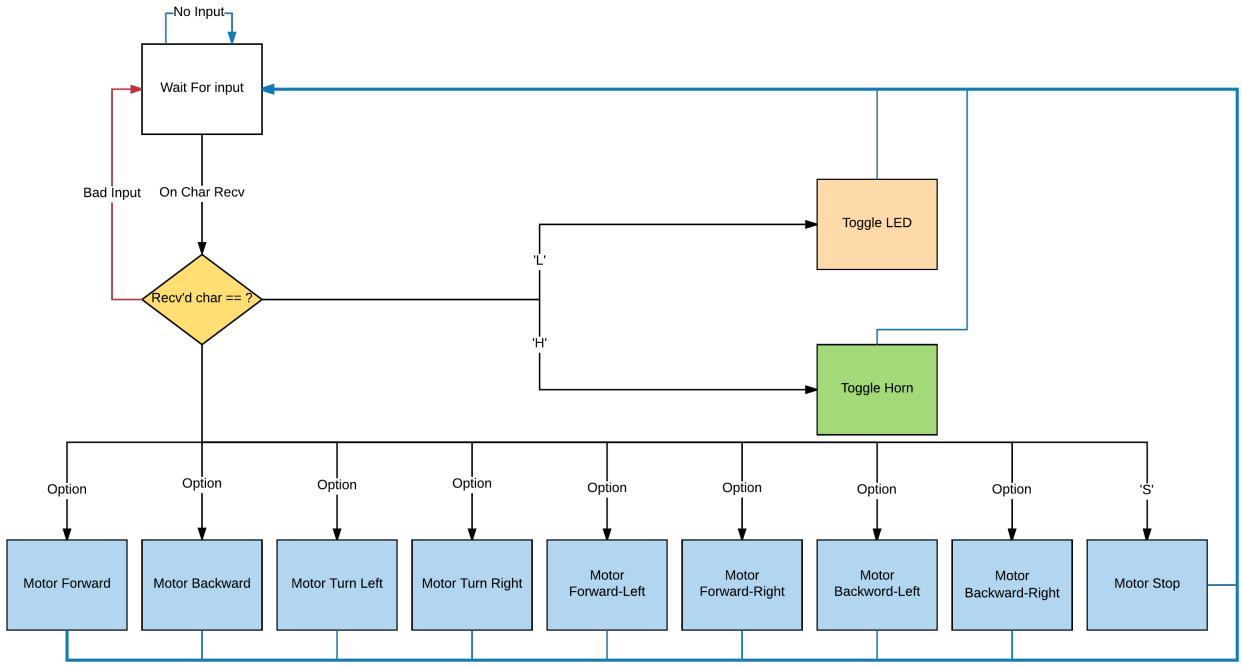

Figure 2: Overall System Block Diagram

Implementation Details

Schematic

Overall

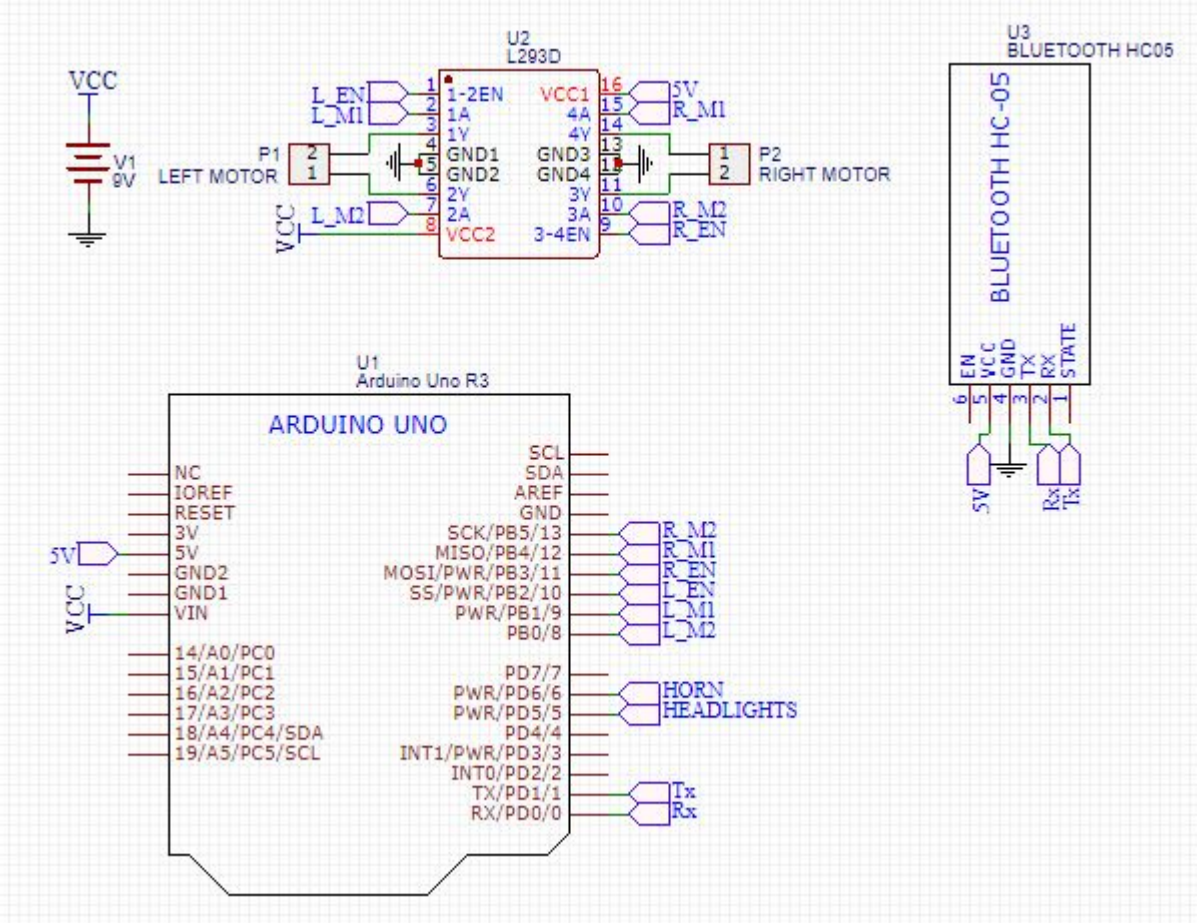

Figure 3: Overall System Schematic

Detailed Portions of Schematic

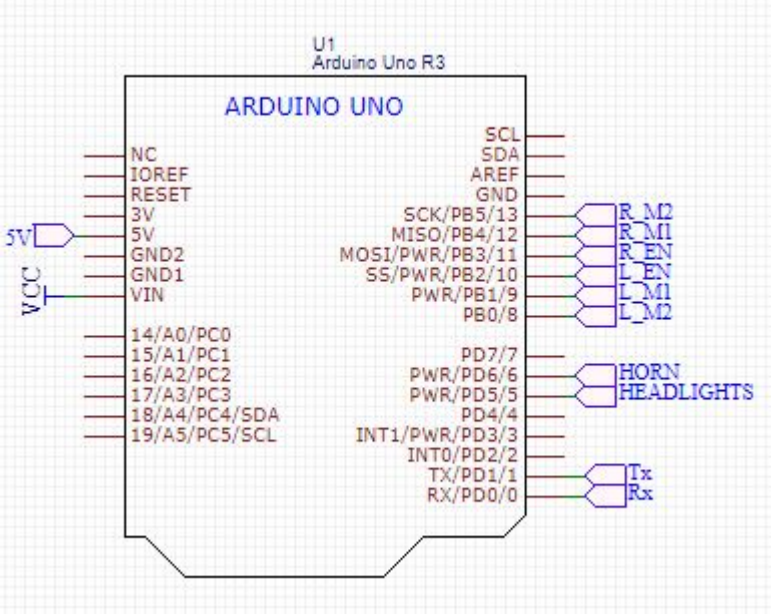

Figure 4: Close up of the Arduino Uno

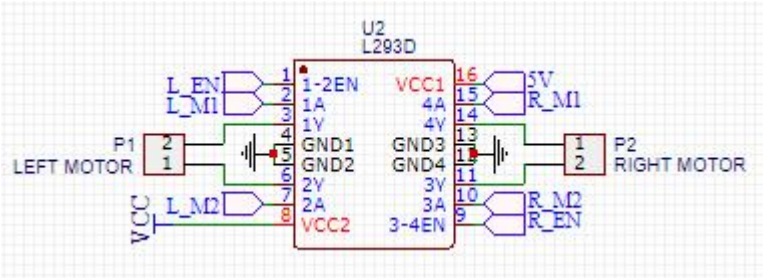

Figure 5: Close up of the Motor driver

Key Code Portion

Main Loop

```
1   void loop()
2   {
3       //check for bt val
4       if(Serial.available())
5       {
6           bt_char = Serial.read();
7           Serial.println(bt_char);
8           Serial.flush();
9       }
10
11      // Conduct action based on BT Value
12      switch(bt_char)
13      {
14          case LIGHT_CHAR:
```

```
            toggle_headlights();
        break;

        case HORN_CHAR:
            toggle_horn();
        break;

        case FOR_CHAR:
            set_motor(FOR);
        break;

        case BACK_CHAR:
            set_motor(BACK);
        break;

        case CCW_CHAR:
            set_motor(CCW);
        break;

        case CW_CHAR:
            set_motor(CW);
        break;

        case STOP_CHAR:
            set_motor(STOP);
        break;

        case FW_LEFT:
            set_motor(FL);
        break;

        case FW_RIGHT:
            set_motor(FR);
        break;

        case BACK_LEFT:
            set_motor(BL);
        break;

        case BACK_RIGHT:
            set_motor(BR);
        break;

        default:
            set_motor(STOP);
        break;
    }
    delay(5); //200 ticks per second
}
```

Motor control

```
void set_motor(int dir)
{
    switch(dir)
    {
        case STOP:
```

```
            //stop
            //Serial.println("Stop");
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, LOW);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, LOW);
        break;

        case BACK:
            //backward
            //Serial.println("Back");
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, HIGH);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, HIGH);
            digitalWrite(R_M2, LOW);
        break;

        case FOR:
            //forward
            //Serial.println("Forward");
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, LOW);
            digitalWrite(L_M2, HIGH);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, HIGH);
        break;

        case CW:
            //rotate CW
            //Serial.println("CCW");
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, HIGH);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, HIGH);
        break;

        case CCW:
            //rotate CCW
            //Serial.println("CW");
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, LOW);
            digitalWrite(L_M2, HIGH);
            digitalWrite(R_M1, HIGH);
            digitalWrite(R_M2, LOW);
        break;

        case FL:
            // Forwards Left
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED + DIAG_OFFSET);
            digitalWrite(L_M1, LOW);
```

```
65              digitalWrite(L_M2, HIGH);
66              digitalWrite(R_M1, LOW);
67              digitalWrite(R_M2, HIGH);
68          break;
69
70          case FR:
71              // Fowards Right
72              analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
73              analogWrite(R_EN, BASE_SPEED);
74              digitalWrite(L_M1, LOW);
75              digitalWrite(L_M2, HIGH);
76              digitalWrite(R_M1, LOW);
77              digitalWrite(R_M2, HIGH);
78          break;
79
80          case BL:
81              // Backwards Left
82              analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
83              analogWrite(R_EN, BASE_SPEED + DIAG_OFFSET);
84              digitalWrite(L_M1, HIGH);
85              digitalWrite(L_M2, LOW);
86              digitalWrite(R_M1, HIGH);
87              digitalWrite(R_M2, LOW);
88          break;
89
90          case BR:
91              // Backwards Right
92              analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
93              analogWrite(R_EN, BASE_SPEED);
94              digitalWrite(L_M1, HIGH);
95              digitalWrite(L_M2, LOW);
96              digitalWrite(R_M1, HIGH);
97              digitalWrite(R_M2, LOW);
98          break;
99
100         default:
101             //stop
102             //Serial.println("Stop");
103             digitalWrite(L_M1, LOW);
104             digitalWrite(L_M2, LOW);
105             digitalWrite(R_M1, LOW);
106             digitalWrite(R_M2, LOW);
107         break;
108     }
109 }
```

Testing/Evaluation

We tested the car by driving it around in lab mostly. We were able to use the lines in between tiles to see how straight the car was going. We then adjusted the value and tried again. We also drove the car around in lab to test the maximum range of the Bluetooth.

Discussions

Challenges

One of the challenges was find a Bluetooth App that would work right out of the box on our phones. We found an app that simply transmits a character on a press or release of a button.

Limitations

One of our motors was weaker then the other and resulted in the car wanting to turn to the left. In order to correct for this we had a small offset value for that motor that we added whenever we did an analogue write. This required us to calibrate the value whenever changes were made to the car, like tightening the wheel.

Possible Improvements

Adding an external sensor, like a pair of rotary encoders, may result in a much better ability for the car to drive straight. We could have fed back the sensor value to the MCU and used a PID control system to adjust the car to keep it going straight. We could have also written a simple Bluetooth app using MIT App Inventor to also send the offset value for the motors and have the user calibrate the system.

Roles and Responsibilities

Matthew Designed Car and built car. Wrote primary code, primary calibration, assisted on report.

Gogol Wrote code to allow for combination movement, some calibration in lab, wrote report and documentation.

Conclusion

In this project we created a Bluetooth Remote Controlled Car. We didn't have any major difficulties implementing this project. We did have to adjust for the fact that one of the motors was stronger than the other, but that was easily adjusted by controlling the PWM for the enable.

Appendix

Code

```
//define pins
#define L_EN 10
#define R_EN 11
#define L_M1 9
#define L_M2 8
#define R_M1 12
#define R_M2 13
#define HORN 6
#define HEADLIGHT 5

//directions
#define FOR 1
#define BACK 2
#define CCW 3
#define CW 4
#define STOP 5
#define FL 6
#define FR 7
#define BL 8
#define BR 9

//BT
char bt_char = '0';
#define FOR_CHAR 'F'
#define BACK_CHAR 'B'
#define CCW_CHAR 'R'
#define CW_CHAR 'L'
#define STOP_CHAR 'S'
#define LIGHT_CHAR 'W'
#define HORN_CHAR 'V'
#define FW_LEFT 'G'
#define FW_RIGHT 'I'
#define BACK_LEFT 'H'
#define BACK_RIGHT 'J'

//globals
bool horn_stat = false;
bool light_stat = false;
int L_M_OFFSET = 25;
int BASE_SPEED = 150;
int DIAG_OFFSET = 80;

//function prototypes
void set_motor(int dir);
void toggle_horn();
void toggle_headlights();

void setup()
{
    //pin modes
    pinMode(HORN, OUTPUT);
    pinMode(HEADLIGHT, OUTPUT);
    pinMode(L_EN, OUTPUT);
    pinMode(R_EN, OUTPUT);
```

```
55          pinMode(L_M1, OUTPUT);
56          pinMode(L_M2, OUTPUT);
57          pinMode(R_M1, OUTPUT);
58          pinMode(R_M2, OUTPUT);
59
60          //setup BT
61          Serial.begin(9600);
62
63          //set initial motor dir to stop and set enables
64          analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
65          analogWrite(R_EN, BASE_SPEED);
66          set_motor(STOP);
67          toggle_headlights();
68      }
69
70      void loop()
71      {
72          //check for bt val
73          if(Serial.available())
74          {
75              bt_char = Serial.read();
76              Serial.println(bt_char);
77              Serial.flush();
78          }
79
80          // Conduct action based on BT Value
81          switch(bt_char)
82          {
83              case LIGHT_CHAR:
84                  toggle_headlights();
85              break;
86
87              case HORN_CHAR:
88                  toggle_horn();
89              break;
90
91              case FOR_CHAR:
92                  set_motor(FOR);
93              break;
94
95              case BACK_CHAR:
96                  set_motor(BACK);
97              break;
98
99              case CCW_CHAR:
100                 set_motor(CCW);
101             break;
102
103             case CW_CHAR:
104                 set_motor(CW);
105             break;
106
107             case STOP_CHAR:
108                 set_motor(STOP);
109             break;
110
111             case FW_LEFT:
112                 set_motor(FL);
113             break;
```

```
114
115             case FW_RIGHT:
116                 set_motor(FR);
117             break;
118
119             case BACK_LEFT:
120                 set_motor(BL);
121             break;
122
123             case BACK_RIGHT:
124                 set_motor(BR);
125             break;
126
127             default:
128                 set_motor(STOP);
129             break;
130         }
131         delay(5); //200 ticks per second
132     }
133
134     void set_motor(int dir)
135     {
136         switch(dir)
137         {
138             case STOP:
139                 //stop
140                 //Serial.println("Stop");
141                 analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
142                 analogWrite(R_EN, BASE_SPEED);
143                 digitalWrite(L_M1, LOW);
144                 digitalWrite(L_M2, LOW);
145                 digitalWrite(R_M1, LOW);
146                 digitalWrite(R_M2, LOW);
147             break;
148
149             case BACK:
150                 //backward
151                 //Serial.println("Back");
152                 analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
153                 analogWrite(R_EN, BASE_SPEED);
154                 digitalWrite(L_M1, HIGH);
155                 digitalWrite(L_M2, LOW);
156                 digitalWrite(R_M1, HIGH);
157                 digitalWrite(R_M2, LOW);
158             break;
159
160             case FOR:
161                 //forward
162                 //Serial.println("Forward");
163                 analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
164                 analogWrite(R_EN, BASE_SPEED);
165                 digitalWrite(L_M1, LOW);
166                 digitalWrite(L_M2, HIGH);
167                 digitalWrite(R_M1, LOW);
168                 digitalWrite(R_M2, HIGH);
169             break;
170
171             case CW:
172                 //rotate CW
```

```
173             //Serial.println("CCW");
174             analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
175             analogWrite(R_EN, BASE_SPEED);
176             digitalWrite(L_M1, HIGH);
177             digitalWrite(L_M2, LOW);
178             digitalWrite(R_M1, LOW);
179             digitalWrite(R_M2, HIGH);
180         break;
181
182         case CCW:
183             //rotate CCW
184             //Serial.println("CW");
185             analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
186             analogWrite(R_EN, BASE_SPEED);
187             digitalWrite(L_M1, LOW);
188             digitalWrite(L_M2, HIGH);
189             digitalWrite(R_M1, HIGH);
190             digitalWrite(R_M2, LOW);
191         break;
192
193         case FL:
194             // Forwards Left
195             analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
196             analogWrite(R_EN, BASE_SPEED + DIAG_OFFSET);
197             digitalWrite(L_M1, LOW);
198             digitalWrite(L_M2, HIGH);
199             digitalWrite(R_M1, LOW);
200             digitalWrite(R_M2, HIGH);
201         break;
202
203         case FR:
204             // Fowards Right
205             analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
206             analogWrite(R_EN, BASE_SPEED);
207             digitalWrite(L_M1, LOW);
208             digitalWrite(L_M2, HIGH);
209             digitalWrite(R_M1, LOW);
210             digitalWrite(R_M2, HIGH);
211         break;
212
213         case BL:
214             // Backwards Left
215             analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
216             analogWrite(R_EN, BASE_SPEED + DIAG_OFFSET);
217             digitalWrite(L_M1, HIGH);
218             digitalWrite(L_M2, LOW);
219             digitalWrite(R_M1, HIGH);
220             digitalWrite(R_M2, LOW);
221         break;
222
223         case BR:
224             // Backwards Right
225             analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
226             analogWrite(R_EN, BASE_SPEED);
227             digitalWrite(L_M1, HIGH);
228             digitalWrite(L_M2, LOW);
229             digitalWrite(R_M1, HIGH);
230             digitalWrite(R_M2, LOW);
231         break;
```

```
232
233             default:
234                 //stop
235                 //Serial.println("Stop");
236                 digitalWrite(L_M1, LOW);
237                 digitalWrite(L_M2, LOW);
238                 digitalWrite(R_M1, LOW);
239                 digitalWrite(R_M2, LOW);
240             break;
241         }
242     }
243
244     void toggle_horn()
245     {
246         //Serial.println("Horn");
247         horn_stat = !(horn_stat);
248         if(horn_stat)
249         {
250             //horn on
251             analogWrite(HORN, 255/2);
252         }
253         else
254         {
255             //horn off
256             digitalWrite(HORN, LOW);
257         }
258     }
259
260     void toggle_headlights()
261     {
262         //Serial.println("Headlights");
263         light_stat = !(light_stat);
264         if(light_stat)
265         {
266             //light on
267             digitalWrite(HEADLIGHT, HIGH);
268         }
269         else
270         {
271             //light off
272             digitalWrite(HEADLIGHT, LOW);
273         }
274     }
```