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

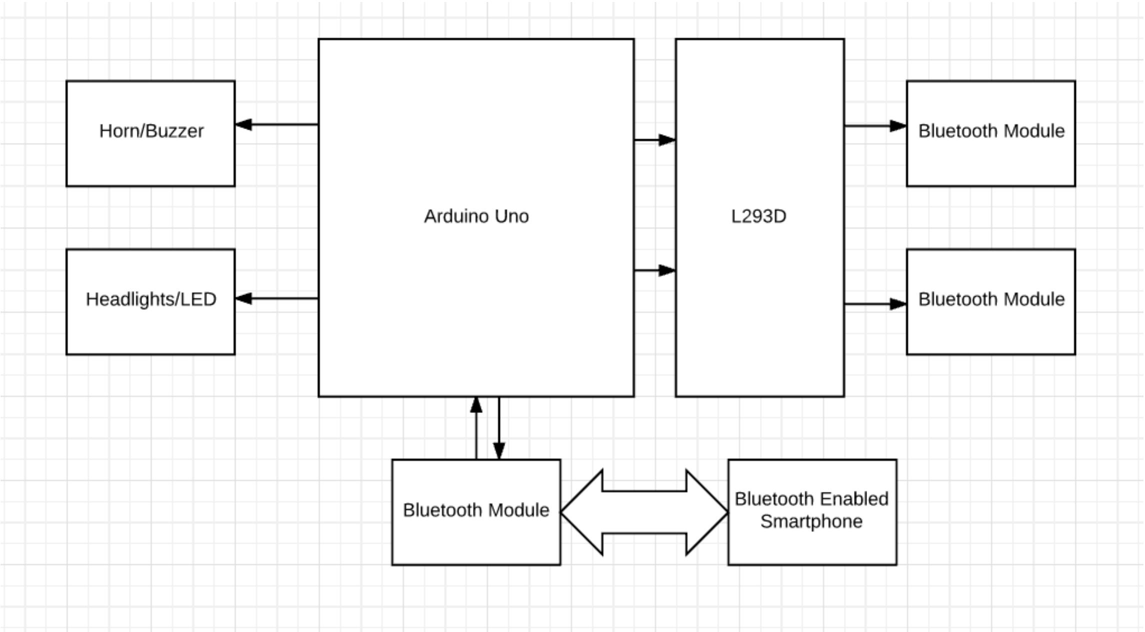

Figure 1: Overall System Block Diagram

Flow Charts

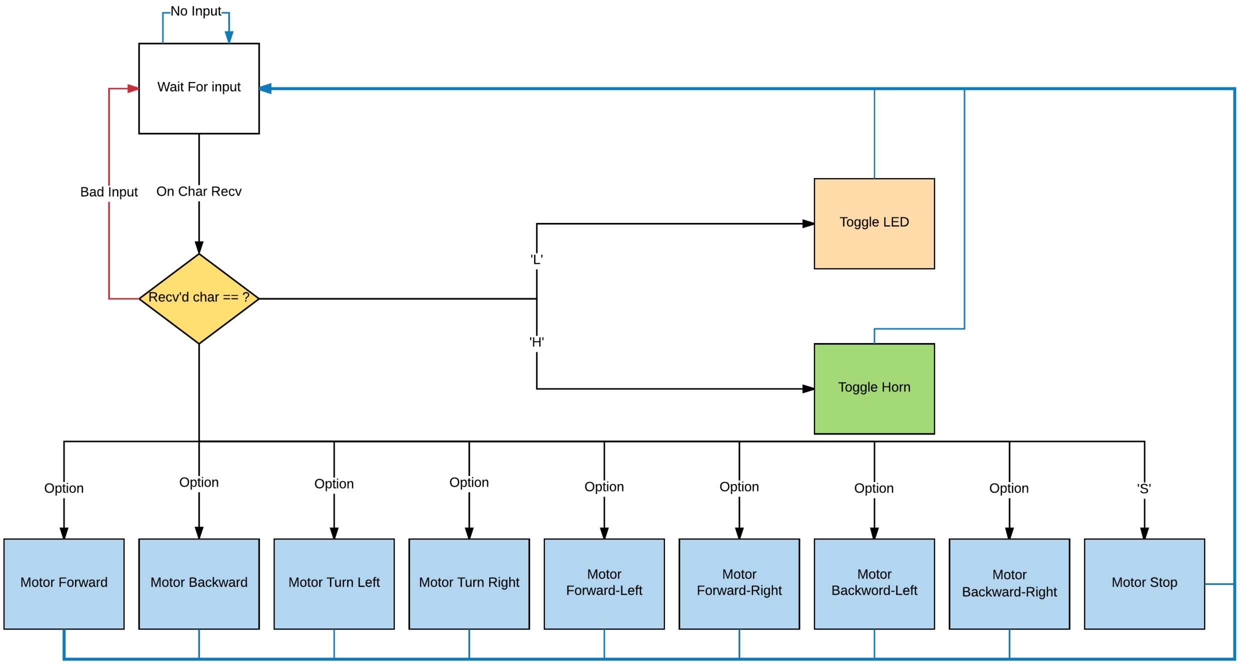

Figure 2: Overall System Block Diagram

Implementation Details

Schematic

Overall

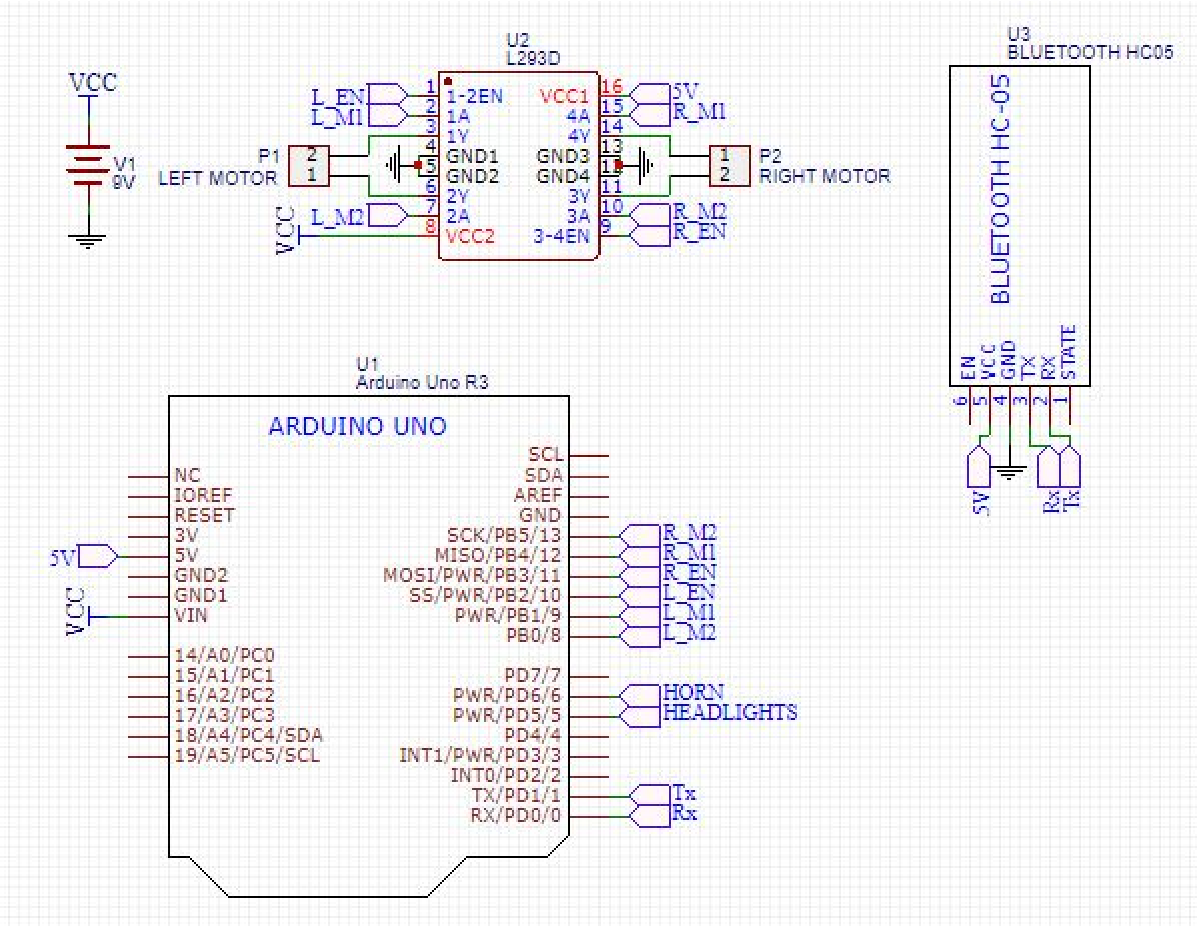

Figure 3: Overall System Schematic

Detailed Portions of Schematic

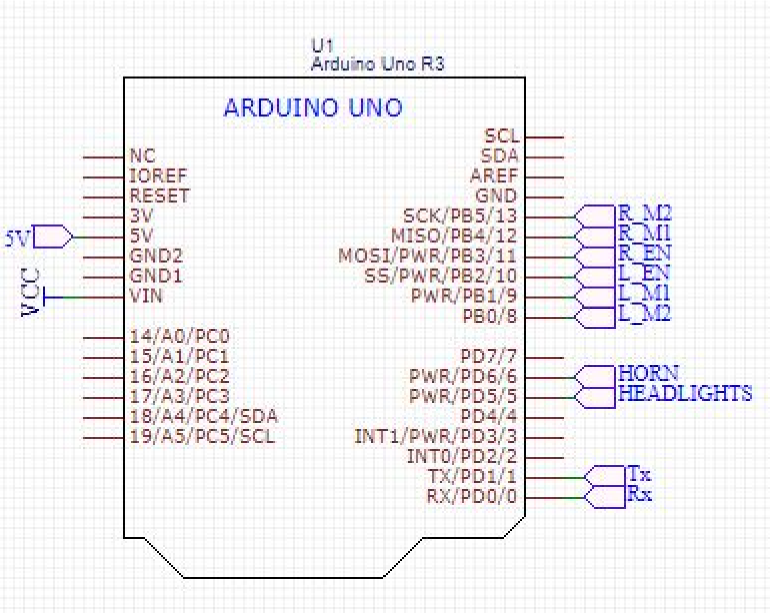

Figure 4: Close up of the Arduino Uno

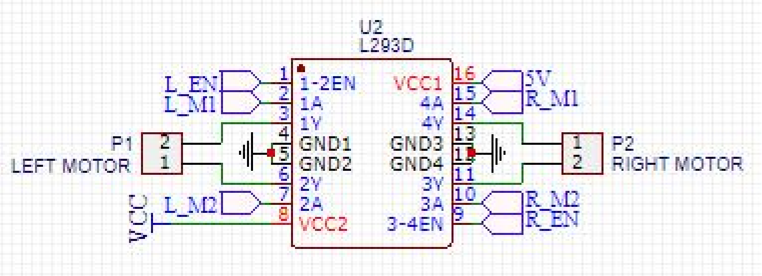

Figure 5: Close up of the Motor driver

Key Code Portion

Main Loop

```
void loop()
{
    //check for bt val
    if(Serial.available())
    {
        bt_char = Serial.read();
        Serial.println(bt_char);
        Serial.flush();
    }

    // Conduct action based on BT Value
    switch(bt_char)
    {
        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);
```

```
            digitalWrite(L_M2, HIGH);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, HIGH);
        break;

        case FR:
            // Fowards Right
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, LOW);
            digitalWrite(L_M2, HIGH);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, HIGH);
        break;

        case BL:
            // Backwards Left
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
            analogWrite(R_EN, BASE_SPEED + DIAG_OFFSET);
            digitalWrite(L_M1, HIGH);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, HIGH);
            digitalWrite(R_M2, LOW);
        break;

        case BR:
            // Backwards Right
            analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
            analogWrite(R_EN, BASE_SPEED);
            digitalWrite(L_M1, HIGH);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, HIGH);
            digitalWrite(R_M2, LOW);
        break;

        default:
            //stop
            //Serial.println("Stop");
            digitalWrite(L_M1, LOW);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, LOW);
        break;
    }
}
```

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);
   pinMode(L_M1, OUTPUT);
   pinMode(L_M2, OUTPUT);
   pinMode(R_M1, OUTPUT);
   pinMode(R_M2, OUTPUT);

   //setup BT
   Serial.begin(9600);

   //set initial motor dir to stop and set enables
   analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
   analogWrite(R_EN, BASE_SPEED);
   set_motor(STOP);
   toggle_headlights();
}

void loop()
{
   //check for bt val
   if(Serial.available())
   {
      bt_char = Serial.read();
      Serial.println(bt_char);
      Serial.flush();
   }

   // Conduct action based on BT Value
   switch(bt_char)
   {
      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
}

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);
                digitalWrite(L_M2, HIGH);
                digitalWrite(R_M1, LOW);
                digitalWrite(R_M2, HIGH);
            break;

            case FR:
                // Fowards Right
                analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
                analogWrite(R_EN, BASE_SPEED);
                digitalWrite(L_M1, LOW);
                digitalWrite(L_M2, HIGH);
                digitalWrite(R_M1, LOW);
                digitalWrite(R_M2, HIGH);
            break;

            case BL:
                // Backwards Left
                analogWrite(L_EN, BASE_SPEED + L_M_OFFSET);
                analogWrite(R_EN, BASE_SPEED + DIAG_OFFSET);
                digitalWrite(L_M1, HIGH);
                digitalWrite(L_M2, LOW);
                digitalWrite(R_M1, HIGH);
                digitalWrite(R_M2, LOW);
            break;

            case BR:
                // Backwards Right
                analogWrite(L_EN, BASE_SPEED + L_M_OFFSET + DIAG_OFFSET);
                analogWrite(R_EN, BASE_SPEED);
                digitalWrite(L_M1, HIGH);
                digitalWrite(L_M2, LOW);
                digitalWrite(R_M1, HIGH);
                digitalWrite(R_M2, LOW);
            break;
```

```

        default:
            //stop
            //Serial.println("Stop");
            digitalWrite(L_M1, LOW);
            digitalWrite(L_M2, LOW);
            digitalWrite(R_M1, LOW);
            digitalWrite(R_M2, LOW);
        break;
    }
}

void toggle_horn()
{
    //Serial.println("Horn");
    horn_stat = !(horn_stat);
    if(horn_stat)
    {
        //horn on
        analogWrite(HORN, 255/2);
    }
    else
    {
        //horn off
        digitalWrite(HORN, LOW);
    }
}

void toggle_headlights()
{
    //Serial.println("Headlights");
    light_stat = !(light_stat);
    if(light_stat)
    {
        //light on
        digitalWrite(HEADLIGHT, HIGH);
    }
    else
    {
        //light off
        digitalWrite(HEADLIGHT, LOW);
    }
}
```