Car driver alert system

**Group - 14**

Shruti Hindocha - 1641016

Manav Chotalia - 1641036

Dhruvi Gajjar - 1641038

Kishan Drangadharia - 1641048



**Summary**

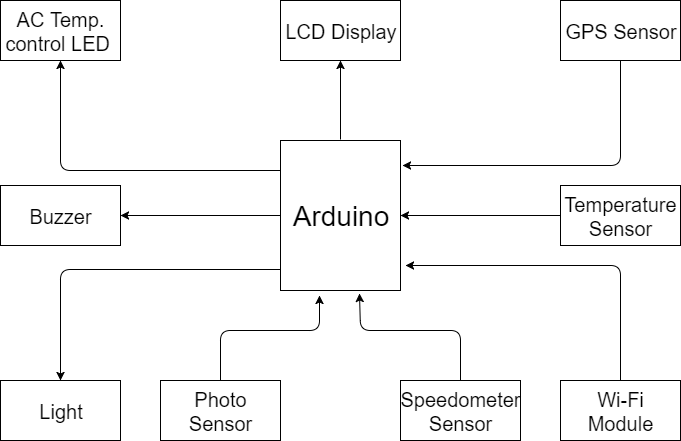
As there are so many road accidents, we would like to propose the System to minimise the road accidents.

To achieve this we would first need to analyse the factors causing accident. After thorough analysis we discovered that the most prominent reasons are over speeding and distraction. We will design our module to rectify this issue.

To avoid over speeding we would first find their location using GPS module and based on previous records of speed in the cloud storage we would give warning to user if their speed, which is to be calculated by speedometer, is more than speed limit or is hazardous. For other distractions like light intensity, temperature inconvenience, alerts will be made accordingly.

Light intensity is checked by photo sensor and temperature regarding alerts are handled by temperature level sensor.

**Block Diagram**



**Components**

* DC Motors
* Hall effect sensor (for speedometer)
* Magnet
* GPS Module(NEO-6M)
* Ethernet Shield
* Temperature Sensor
* Photo Sensor
* LCD Display (16x2)
* LEDs
* Buzzer
* Arduino
* Jumper Wires
* Resistors

**Selection criteria of components**

**LM35:**  
LM35 has good balance between price and accuracy and also the operating voltage of LM35 is 5v which is very feasible when working with Arduino.

**Ethernet-shield:**  
Main purpose of using this module is to receive data of speed of corresponding area from web-page. We are using this module as no other module allow us to receive data from web page on Arduino. Or we can us Arduino with in-built Ethernet-shield,

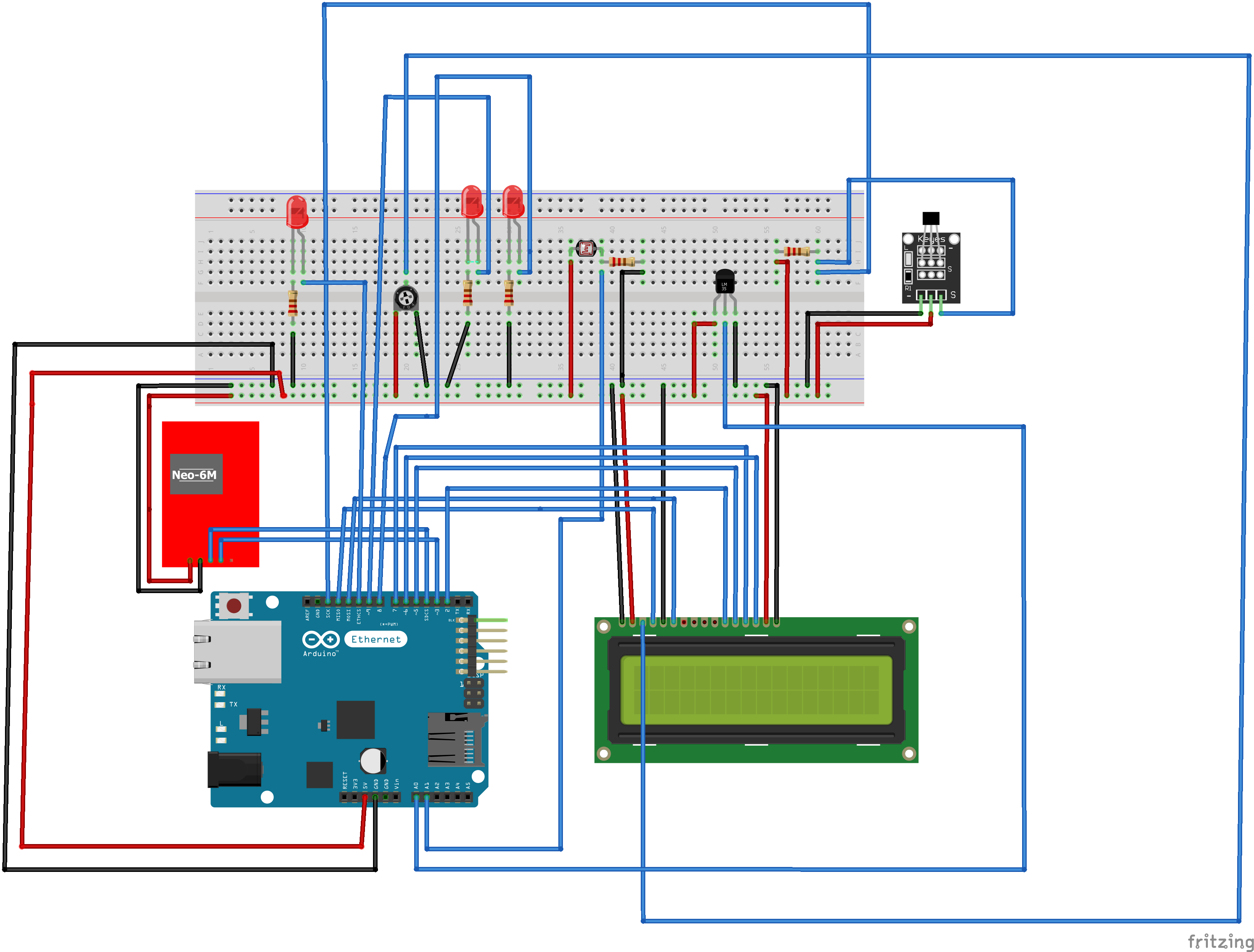
**Photo-Resistor:**  
Photo-resistor is cheapest option available for detecting intensity of light and also its operating voltage is feasible with that of Arduino.

**Hall-effect sensor:**  
Hall-effect sensor will be used to calculate speed of the vehicle . We choose this option as its result doesn't get affected by change in surrounding like lighting condition.

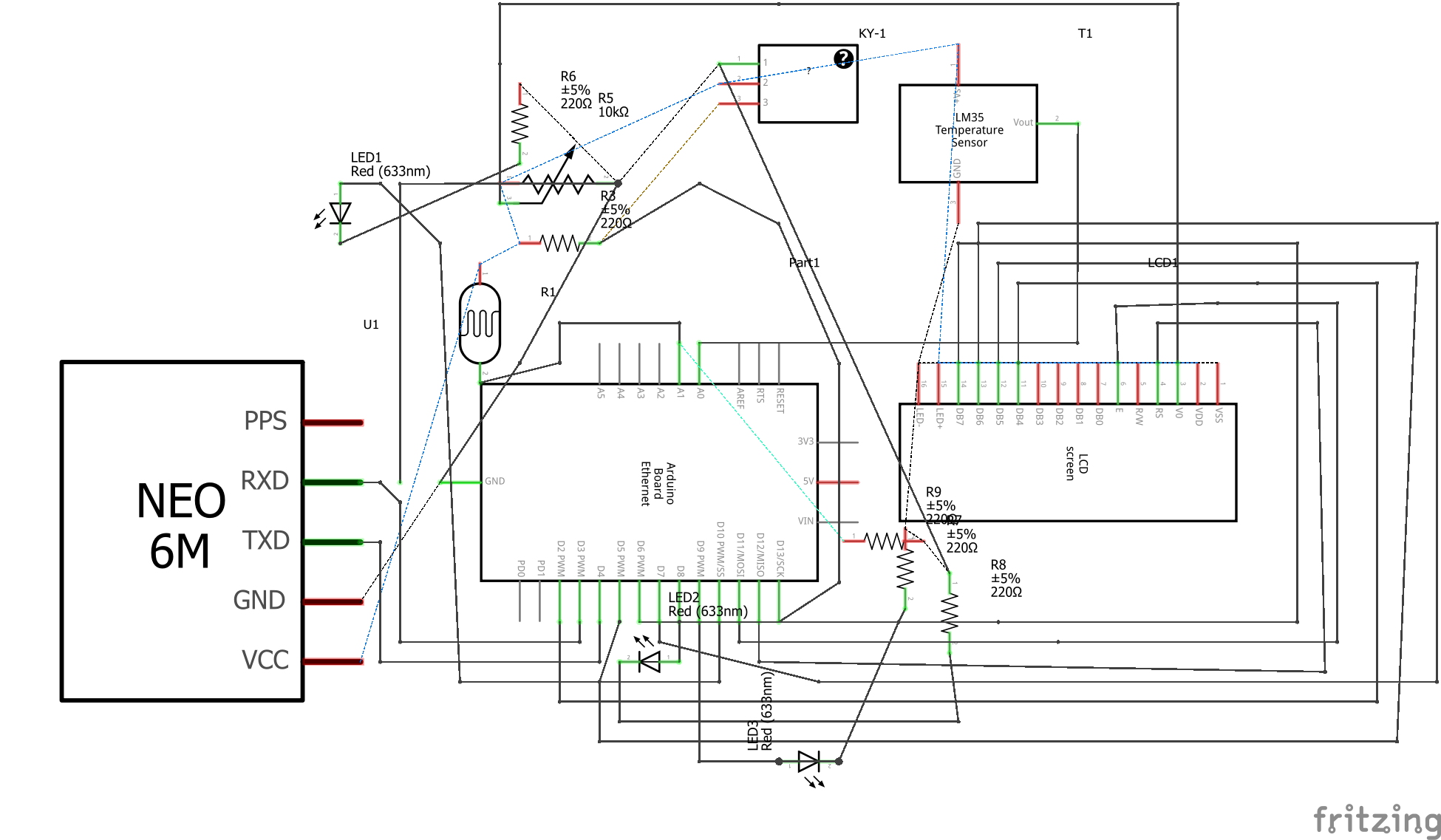
**GPS Module NEO-6M-0-001**:  
We have changed our sensor SKM53 to NEO-6M-0-001 because of availability of sensor and price of sensor. Also NEO-6M-0-001is more popular than SKM53. NEO-6M-0-001 will be used to get precise location of vehicle. As GPS works globally and NEO-6M-0-001 is very cheap considering this category it naturally becomes the most feasible option for positioning.

**LED**:  
They will be used to indicate the inference derived from data collected from above mentioned sensors

**Circuit Diagram**



**Schematic Diagram**

****

**Flow chart**

Yes

Turn on LED A and Buzzer

No

If Speed limit of that area > Speedometer’s speed

Input from GPS module and Ethernet shield and hall effect speedometer

A

C

B

A

D

Start

B

Input from Photo register

Display current Temp., current Speed, Top Speed, Temp, Needed

Input from Temperature sensor, Ethernet, GPS Sensor, Hall Effect Speedometer

C

Yes

No

LED B(Head light) turn on

If Light intensity of environment is < visible intensity

D

LED C blink slowly

LED C Turn On

LED C turn off

Surrounding Temp. is = Temp needed

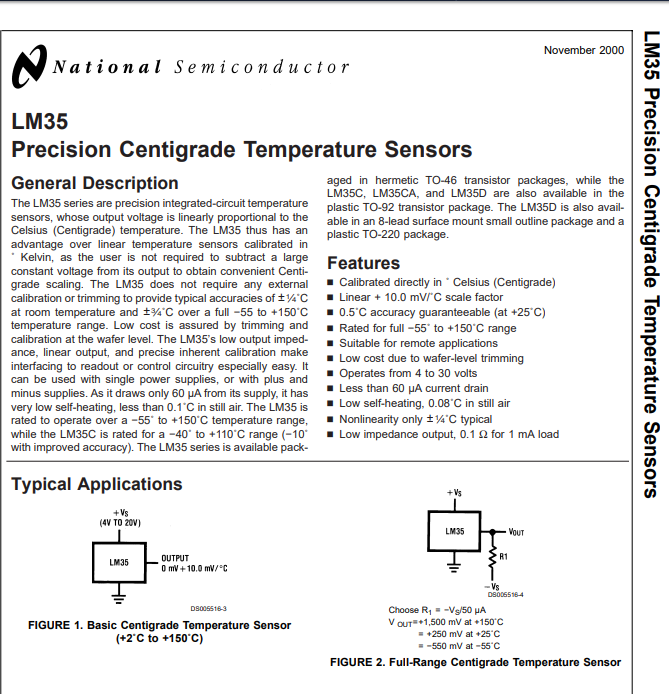
Surrounding Temp. is < Temp needed

Surrounding Temp. is > Temp needed

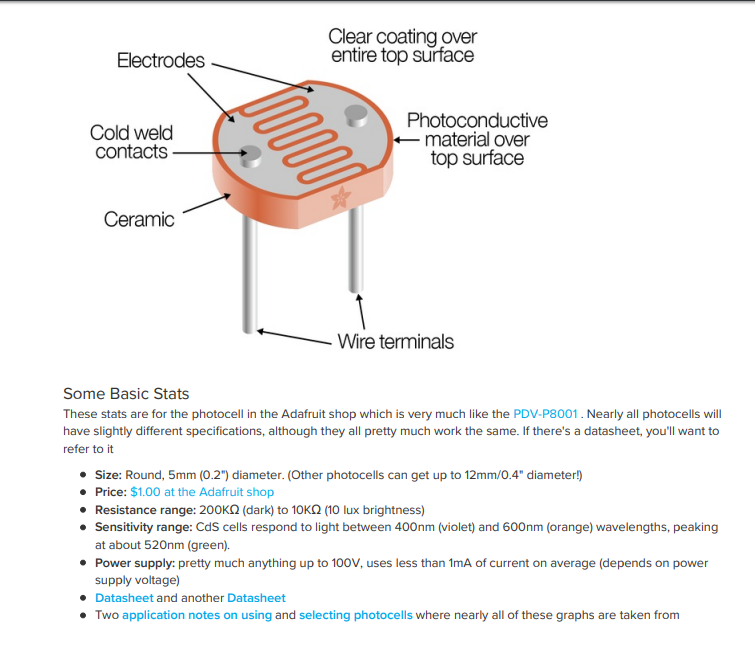
Input from Temperature sensor

**Datasheets**

**LM35:**

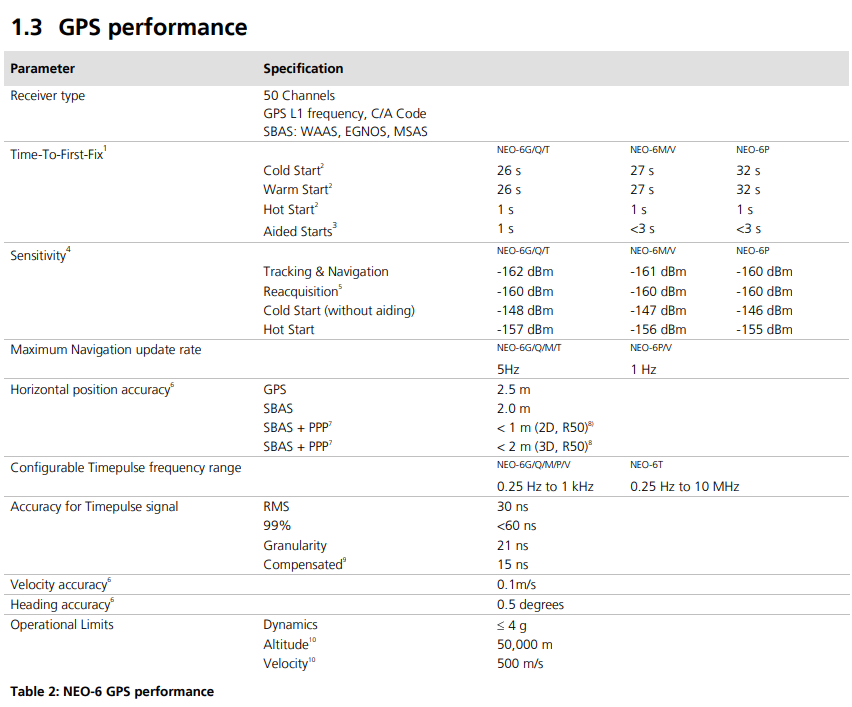
****

**Photo Register :**

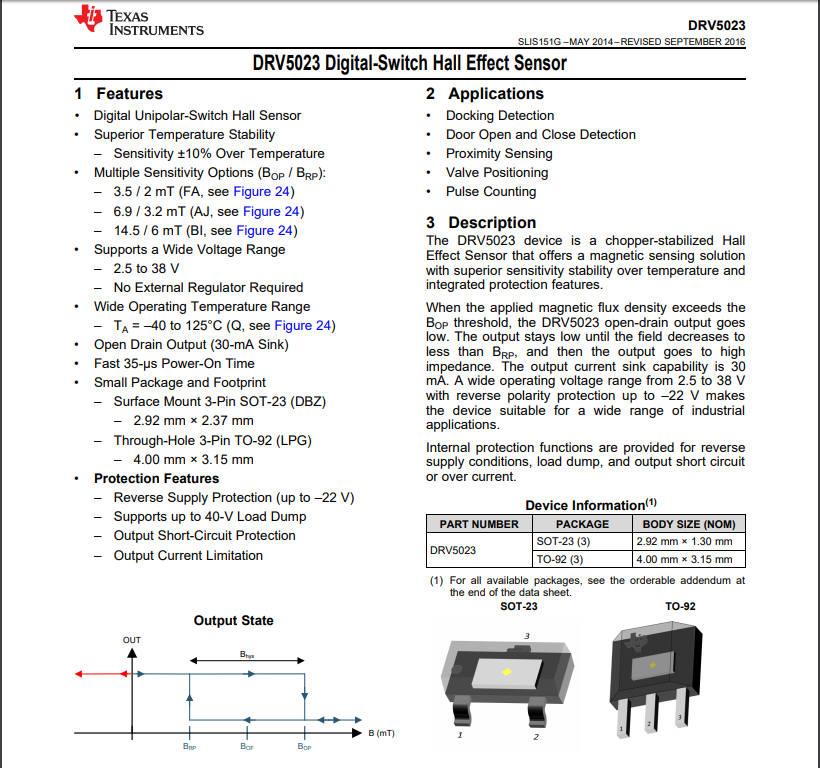
****

**GPS Module (NEO-6M):**

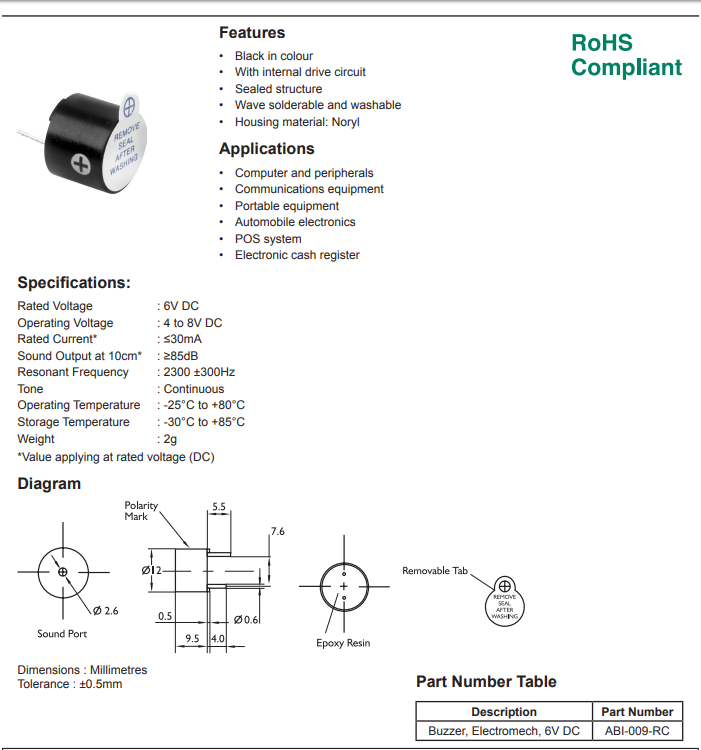
The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints. The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of under 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.

****

**Hall Effect Sensor :**

****

**Buzzer:**

****

**Operating Principles**

**LM35**

LM35 works on Proportional To Absolute Temperature (PTAT) principle.

PTAT dictates that voltage across the PN junction present in LM35 varies with current density and temperature fluctuation. Surrounding circuits is made in order to make operating current density constant, so that fluctuation of the temperature can be linearized to change in voltage, so that temperature can then be measured in terms of voltage. Using this we would calculate the temperature of cabin of car and thus turn on/off the ac as per convenience pre-set by the user.

**Code**

float tempC;

int tempPin = 0;

void setup(){

Serial.begin(9600);

}

void loop(){

tempC = analogRead(tempPin);

tempC = (5.0 \* tempC \* 100.0)/1024.0);

Serial.println((byte)tempC);

delay(1000);

}

**Photo Register**

A photo resistor is made of a high resistance semiconductor. Photo resistor can have a resistance as high as several mega-ohms (MΩ), and as low as a few hundred ohms. Photo resistor have light sensitive material whose resistance increase with decreasing light intensity. Thus light intensity is inversely proportional to resistance of a photo resistor. Using this we would calculate the intensity of light and determine whether user should turn on\off their headlight

**Code**

int LDR = 0;

int LDRValue = 0;

int light\_sensitivity = 25;

void setup(){

Serial.begin(9600);

pinMode(13, OUTPUT);

}

void loop() {

LDRValue = analogRead(LDR);

Serial.println(LDRValue);

delay(50);

if (LDRValue < light\_sensitivity) {

digitalWrite(13, HIGH);

}

else{

digitalWrite(13, LOW);

}

}

**GPS Module (NEO-6M-0-001)**

Neo-6m works as GPS device. It is capable of calculating distance between geostatic satellite and itself. It would calculate the distance between itself and three other satellite, which would be enough to determine the longitude and latitude of the place from which device is operated. Using this we would calculate our longitude and latitude. Which would then be passed to web server in order to get the speed limit set by transportation authority of that area. If the speed calculated by Hall Effect sensor is greater than this speed then driver is notified to decrease their speed.

**Code**

#include <TinyGPS.h>

#include <LiquidCrystal.h>

#include <SoftwareSerial.h>

#include <TinyGPS.h>

float lat = 0,lon = 0;

SoftwareSerial gpsSerial(3,4);//rx,tx

LiquidCrystal lcd(A0,A1,A2,A3,A4,A5);

TinyGPS gps;

void setup(){

Serial.begin(9600);

Serial.println("The GPS Received Signal:");

gpsSerial.begin(9600);

lcd.begin(16,2);

}

void loop(){

while(gpsSerial.available()){

if(gps.encode(gpsSerial.read())){

gps.f\_get\_position(&lat,&lon);

lcd.clear();

lcd.setCursor(1,0);

Serial.println("GPS Signal");

lcd.setCursor(1,0);

lcd.setCursor(5,0);

lcd.setCursor(0,1);

lcd.setCursor(5,1);

String latitude = String(lat,6);

String longitude = String(lon,6);

Serial.println("latitude"+latitude+";"+"longitude"+longitude);

delay(1000);

}

}

}

**Hall Effect sensor**

In a Hall Effect sensor a thin strip of metal has a current applied along it, in the presence of a magnetic field the electrons are deflected towards one edge of the metal strip, producing a voltage gradient across the short-side of the strip thus allowing us to note presence of the magnetic field and its direction. Thus variation in voltage, caused due to presence of magnetic field can be noted. Using this principle we can place a magnet on end of tire and can measure time between detection of magnet which would allow us to calculate the speed of the vehicle.

int potPin = A0;

int motorPin = 9;

int potValue = 0;

int motorValue = 0;

void setup() {

Serial.begin(9600);

}

void loop() {

potValue = analogRead(potPin);

motorValue = map(potValue, 0, 1023, 0, 255);

analogWrite(motorPin, motorValue);

Serial.print("potentiometer = " );

Serial.print(potValue);

Serial.print("\t motor = ");

Serial.println(motorValue);

delay(2);

}

**Buzzer**

The vibrating disk in a magnetic buzzer is attracted to the pole by the magnetic field. When an oscillating signal is moved through the coil, it produces a fluctuating magnetic field which vibrates the disk at a frequency equal to that of the drive signal thus emitting sound.

**Code**

int buzzer = 3;

void setup() {

}

void loop() {

tone(buzzer, 1000, 500);

}

**16x2 LCD Display**

**Code**

#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(14, 15, 16, 17, 18, 19);

void setup() {

// set up the LCD's number of columns and rows:

lcd.begin(16, 2);

// Print a message to the LCD.

lcd.print("hello, world!");

}

void loop() {

// set the cursor to column 0, line 1

// (note: line 1 is the second row, since counting begins with 0):

lcd.setCursor(0, 1);

// print the number of seconds since reset:

delay(500);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("Hello!");

lcd.setCursor(10, 1);

lcd.print("Peeps");

delay(1000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("ICT");

lcd.setCursor(8, 1);

lcd.print("2020");

delay(1000);

}

**Final Code**

#include <SPI.h>  
#include <Ethernet.h>  
#include <TinyGPS.h>  
#include <SoftwareSerial.h>  
#include<LiquidCrystal.h>  
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};  
IPAddress ip(10,20,12,177);  
EthernetClient client;  
char server[] = "10.20.32.239";  
size\_t maxSize = 512;  
char response[512];  
SoftwareSerial gpsSerial(3,4);//rx,tx  
TinyGPS gps;  
float tempC, LDRValue, latitude, longitude;  
int tempPin, Headpin, hallEffect, vehicleStatus, speedLimit, count, vehicleSpeed;  
LiquidCrystal lcd(12, 11, 6, 7, 5, 2);  
int LDR,analogInputTemp;  
void setup() {  
    Serial.begin(9600);  
    Serial.println("--- Start ---");  
    delay(1000);  
    Ethernet.begin(mac, ip);   
    Serial.print("My IP address: ");  
    Serial.println(Ethernet.localIP());  
    gpsSerial.begin(9600);  
    lcd.begin(16,2);  
  
}  
void loop() {  
    LDRValue = analogRead(LDR);    
    tempC = (5.0 \*analogRead(analogInputTemp)\* 100.0)/1024.0;  
    lcd.print("Temperature:" + String(tempC));  
    if(tempC > 25)  
        digitalWrite(tempPin,HIGH);  
    else  
        digitalWrite(tempPin,LOW);  
    if(LDRValue < 80)  
        digitalWrite(Headpin,HIGH);  
    else  
        digitalWrite(tempPin,LOW);  
    if(gpsSerial.available() && gps.encode(gpsSerial.read())){   
        gps.f\_get\_position(&latitude,&longitude);  
        speedLimit = httpRequest(latitude,longitude);  
    }  
    else  
        Serial.println("GPS is not working");  
    if(vehicleStatus == 1 && hallEffect == 1) {  
        count = 0;  
        while(hallEffect == 1) {  
              
        }  
        while(hallEffect == 0) {  
            count++;  
        }  
    }  
    vehicleSpeed = 3.142\*count\*0.5;  
    if(vehicleSpeed > speedLimit){  
        digitalWrite(tempPin,HIGH);  
        lcd.print("over SPEED!! ");  
    }  
    else{  
      lcd.print("SPEED:"+vehicleSpeed );  
        digitalWrite(tempPin,LOW);  
    }  
      
}  
  
int httpRequest(float latitude, float longitude) {  
    client.stop();  
    if (client.connect(server, 80)) {  
        String s = "GET /index.php?lat=";  
        s.concat(String(latitude));  
        s.concat("&lon=");  
        s.concat(longitude);  
        client.println(s);  
        client.println("Host: 10.20.32.239");  
        delay(1000);  
        size\_t len = client.readBytes(response, maxSize);  
        response[len] = 0;  
        client.println("Connection: close");  
        client.println();  
        //Serial.println(response);  
        int Limit = atoi(response);  
        return(Limit);  
    }  
    else  
        Serial.println("connection failed");  
    delay(1000);  
}