Faculty of Information Technology

IS 1900 – Business Project

Anti-sleep alarm with alcohol and flame detection

Final Report

Group No: 08

No	Index No	Name
1	**205080K**	**Raguraj S.**
2	205092A	Sagini N.
3	205048V	Kaneshan T.
4	205039U	Ishvini A.

Supervisor Name: Mr. B.H. Sudantha
Dean/Senior Lecturer,
Faculty of Information Technology,
University of Moratuwa

Examiner Name: Mr. S. M. U. Premasiri
Lecturer,
Department of Information Technology,
University of Moratuwa

Date of Submission: 08/06/2022

Table of Contents

Tables

Table of Figures

1. Introduction

All over the world the ever-increasing numbers of traffic accidents are occurred due to the driver's drowsiness and alcohol usage, and fire accidents. Sleep related accidents tend to be more severe, possibly because of the higher speeds involved and driver is unable to take any avoiding action, or even brake, prior to the collision. And also, a considerable part of the traffic accident has some correlation with drunk driving.

2. Literature Survey

A video-based strategy and Artificial intelligence in expensive new model cars[1][2][3] are used to identify driver's sleepiness. There is also an anti sleep alarm with vibrator in an online market but it is irritating,ear infection and skin abrasion for drivers. Our system is designed to identify, monitor, and study the head position and eye movement of drivers in order to validate the drowsiness of the driver. Eye blink sensors can help to reduce accidents caused by drowsy drivers. The driver must wear the head band while they are driving, and their eyelids must be closed for a few seconds to detect tiredness. If the driver nods off, the LCD will show the warning.

When an accident occurs, it is critical to provide medical assistance as quickly as possible, so that the message will be reached to the authorized person. This is how the driver and the authorized person may both be warned if the driver becomes drowsy and alcoholic. The ideal answer for this problem is to send the geographical coordinates of the accident location. This difficulty may be solved with the use of GPS and GSM modules. The alcohol detection system is set up in such a way that if any of the drivers in the car are inebriated, the system sends a message to the authorized person.If fire is detected,an alarm will be rung and location will be sent to authorized person.

For a long time, researches have investigated driver drowsiness. There are only a few commercial goods on the market right now.For example, Lumeway's Eye Alert system (2015), uses infrared camera/sensors to track the rate and duration of a driver's eye closure. It raises an alarm when the driver begins to behave in an unsafe manner.

Toyota initially launched the Driver Attention Monitor car safety technology in 2006 for its Lexus newest model cars (ex:2006-2011 Lexus GS 450h). Infrared sensors are used to

track the driver's attention. The driver monitoring system features a CCD camera mounted on the steering column that uses infrared LED detectors to track the driver's face. If a risky scenario is detected when the driver is not paying attention to the road ahead, the system will alert the driver with flashing lights and warning sounds. If no action is taken, the car will automatically apply the brakes.

The InSight technology (InSight, 2015) from SMI was designed to identify driver drowsiness and inattention by monitoring the driver's face with cameras. DADSTM (Driver Alertness Detection SystemTM) (DADS, 2015) is a cloud-based service that monitors a driver's alertness in real-time to reduce the risk of drowsiness and fatigue-related road accidents. A smartphone and a certified Bluetooth camera are required to use the system. The camera gathers information from a driver's face, which is then analyzed by algorithms to track their level of attention while driving. If a risk threshold is exceeded, the motorist will receive a phone alarm. According to the creators, the device can alert the driver up to two hours before it reaches a critical level.

Table 1 Cars and name of their drowsiness detection system

Audi	Rest Recommendation System
BMW	Active Driving Assistant
Bosch	Driver Drowsiness Detection
Ford	Driver Alert in 2011
Hyundai	Driver Attention Monitor in 2017
Nissan	Driver Attention Alert in 2014
Volkswagen	Fatigue Detection System

In present situation more Electric cars like Tesla are using Artificial Intelligence related technologies to identify driver drowsiness in their vehicles. As a conclusion, more efficient actions should take place to achieve the main goal of preventing and reducing accidents.

We can improve our project by using more advance microcontroller and technologies to give precise results to driver at affordable price.

3. Aim and Objectives

3.1 Aim

The aim of our project is to reduce the accident that caused by drowsiness, drunk driving, and fire.

3.2 Objectives

- To check whether, driver is sleeping or not.
- To check whether, driver has drunk or not.
- To reduce fire-based accidents.
- To inform to his well-wishers about unstable situations.
- To identify the current location of vehicle in the unstable situations.

4. Analysis and Design

4.1 Block Diagram

Our system will use a flame sensor, pressure sensor, eye blinking sensor and gyroscope module to check on fire, speeding, driver's eye movement and head movement respectively.

If the driver's sleepiness and fire detection conditions are met, it will display a warning message and alert him by blinking hazard lights, Control the sound, and ringing the buzzer. If the driver receives an alarm within 5 seconds, he must turn off the buzzer. If not, our system will use the GPS module to identify the location and send it to a predetermined phone number via the GSM/GPRS module. If the driver consumes alcohol which is sensed by a gas sensor, the location will be sent to the predefined phone number.

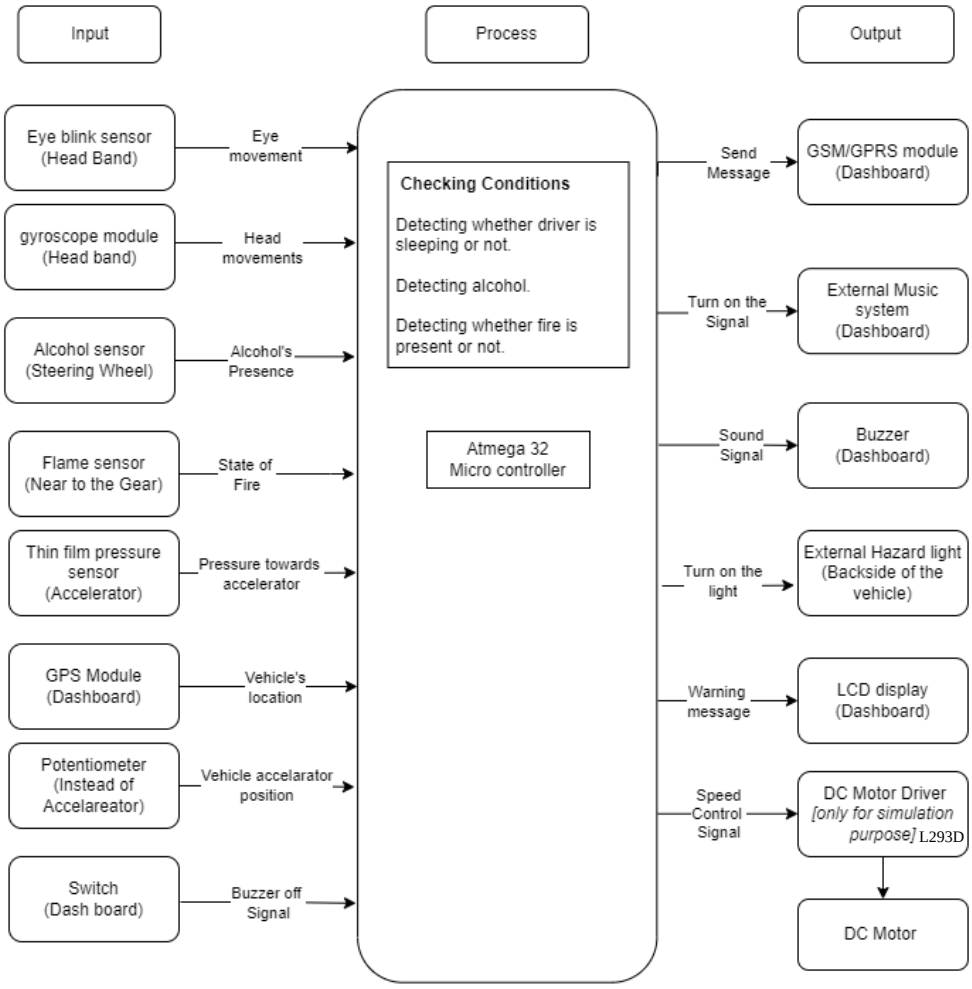

Figure 1 Input Output Diagram

4.2 3D Design

Figure 2 Location of Gyroscope Module

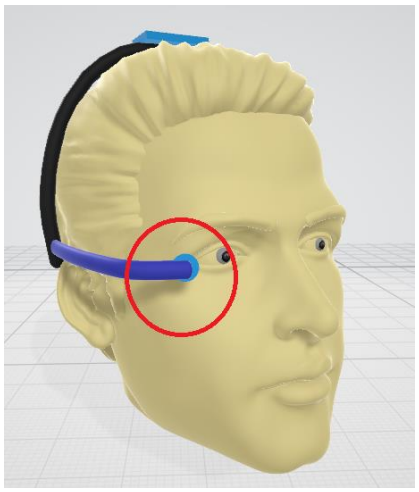

Figure 3 Location of Eyeblinking Sensor

Figure 4 Location of MQ3 Gas Sensor

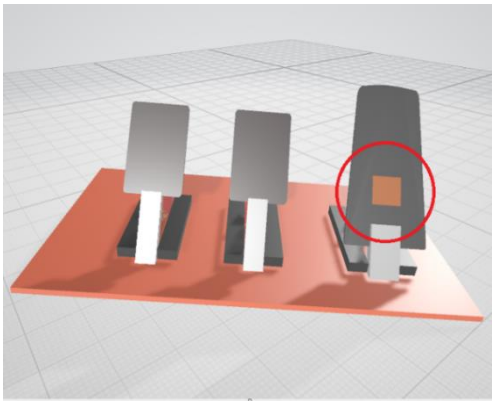

Figure 5 Location of Pressure Sensor

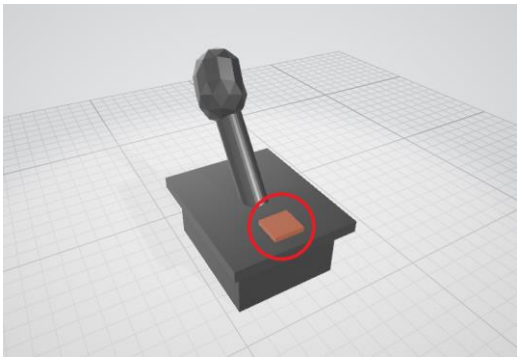

Figure 6 Location of Flame Sensor

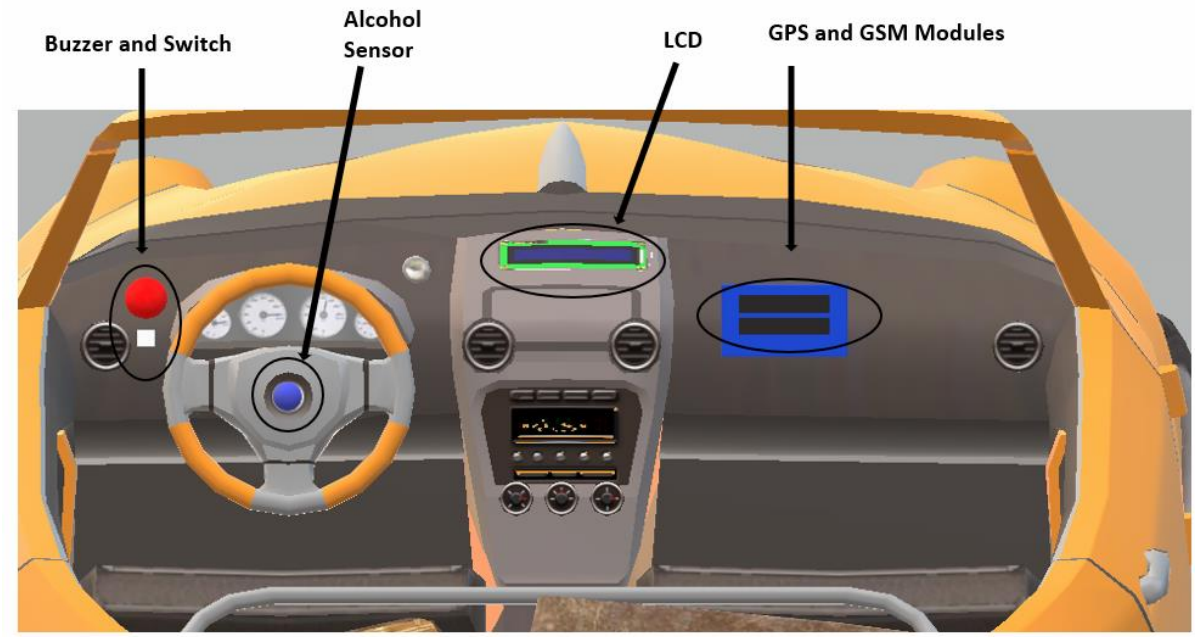

Figure 7 Components on Dashboard

4.3 Schematic Diagram

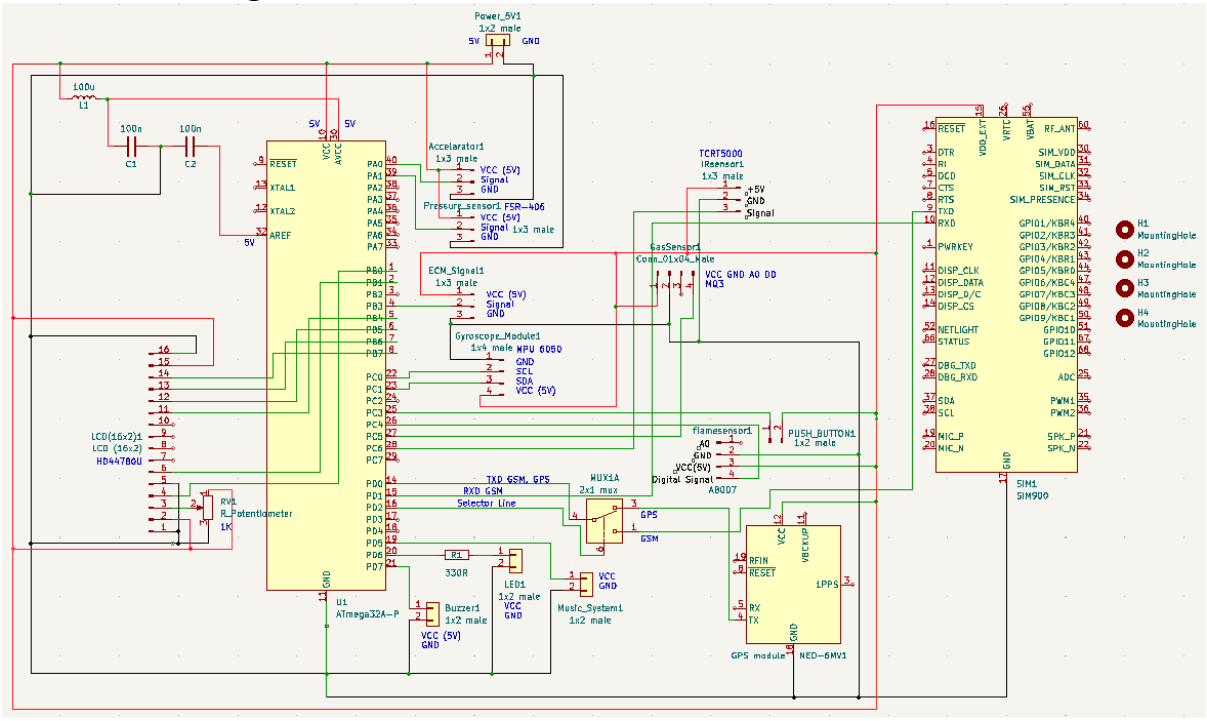

Figure 8 Complete Schematic Diagram - KiCad

4.4 Final Pin Diagram - Proteus

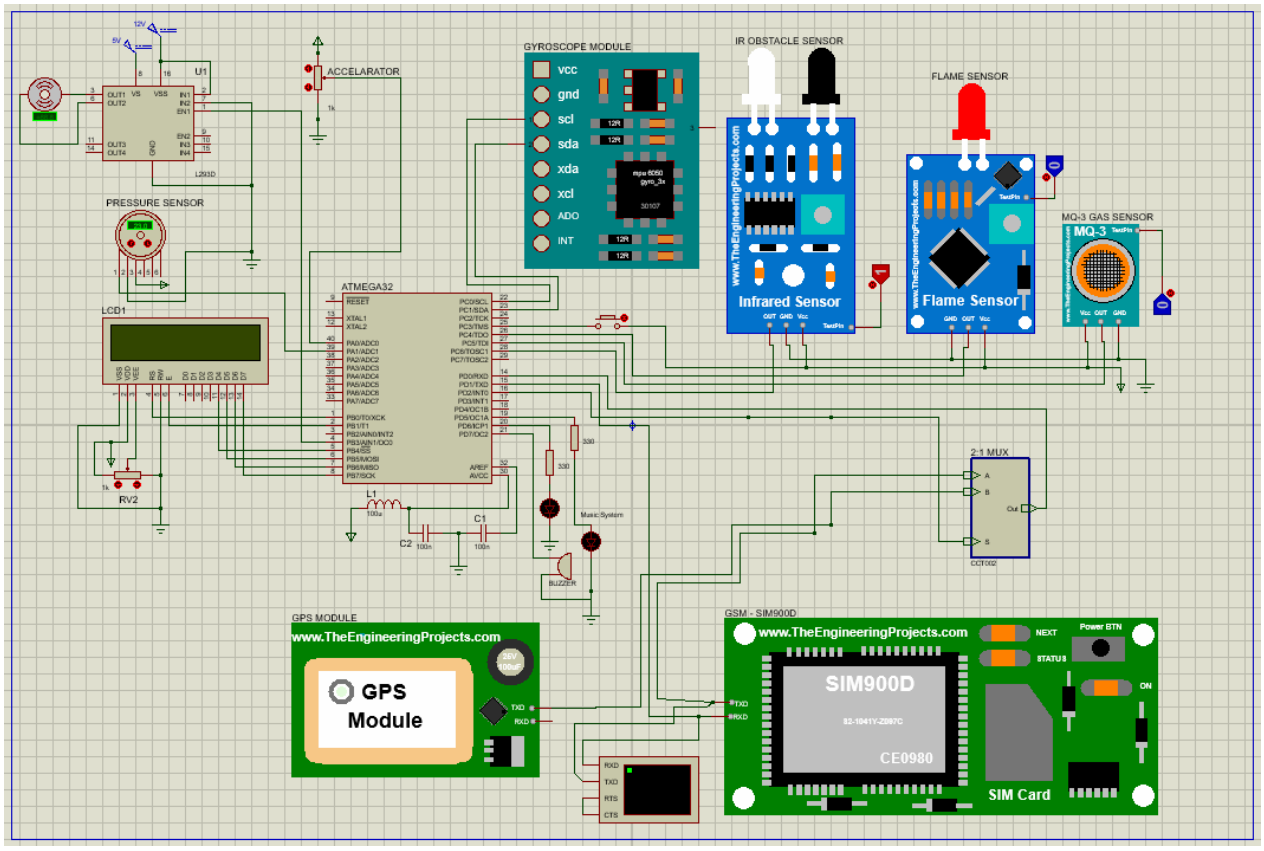

Figure 9 Final Pin Diagram - Proteus

4.5 PCB Design

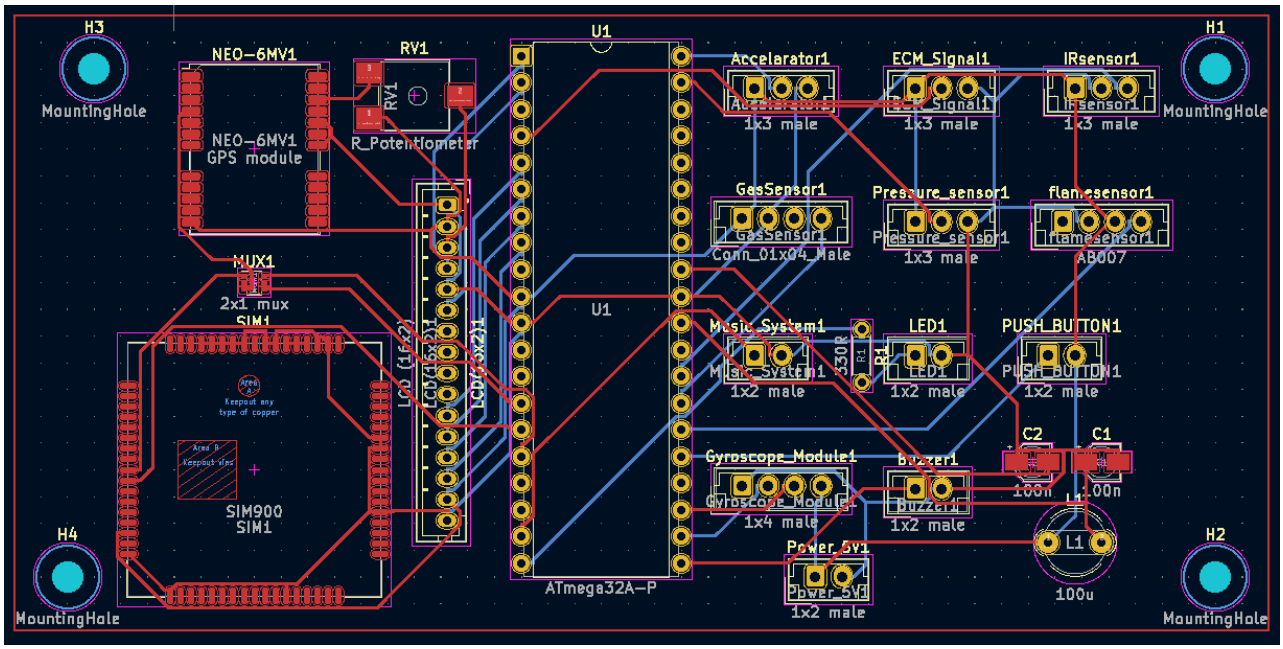

Figure 10 Final PCB Design

4.6 PCB 3D View

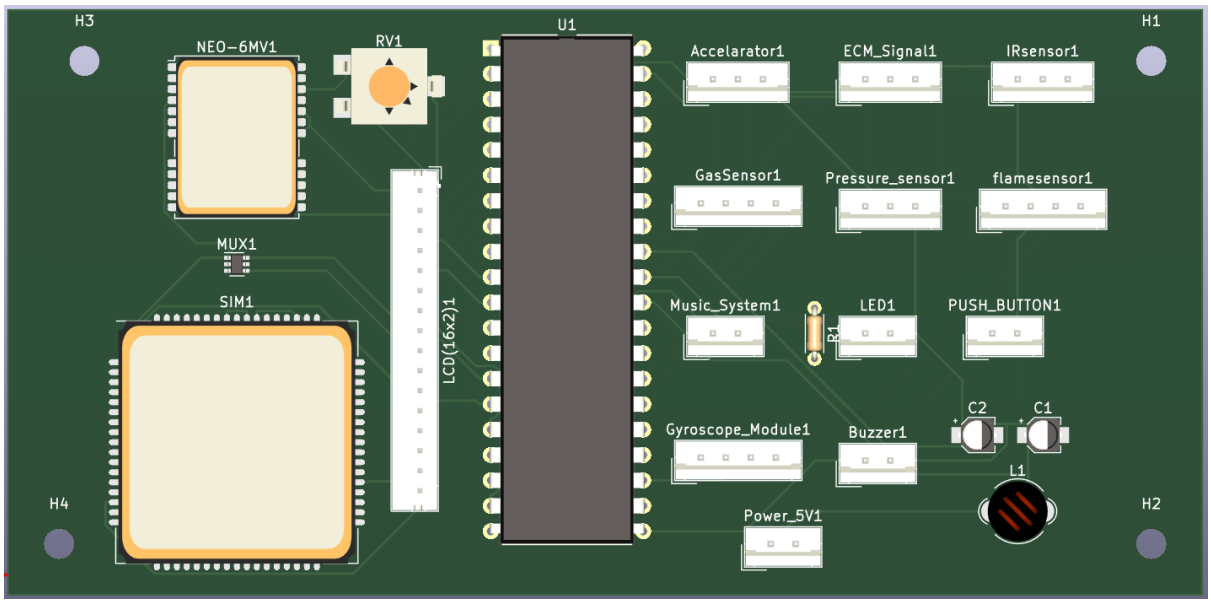

Figure 11 Final PCB 3D View

4.7 Power Supply

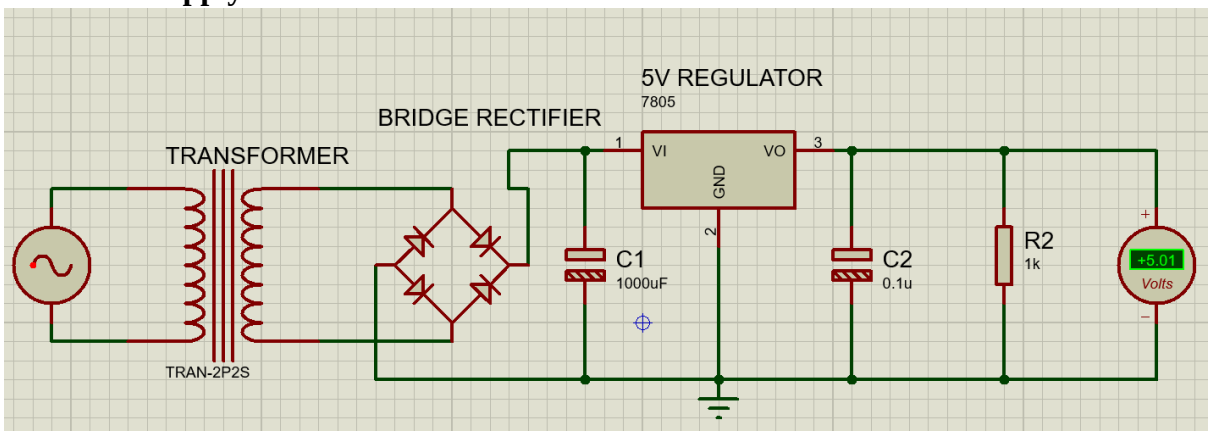

Figure 12 5V Power Supply

5. Testing and Implementation

Simulation Results

The below figures show the results of the following situations. First, our system checks whether there is a flame detected or not. If the flame is not detected, our system will check the pressure sensor for identifying whether the vehicle is moving or not. If the vehicle is moving it will check whether the driver is drunk or not. If the driver is not drunk it will check whether the driver is sleeping or not by using a gyroscope and IR sensor. Since we don't have gyroscope module in proteus here we are using a logic toggle instead of a gyroscope module.

Test 1: When flame is detected

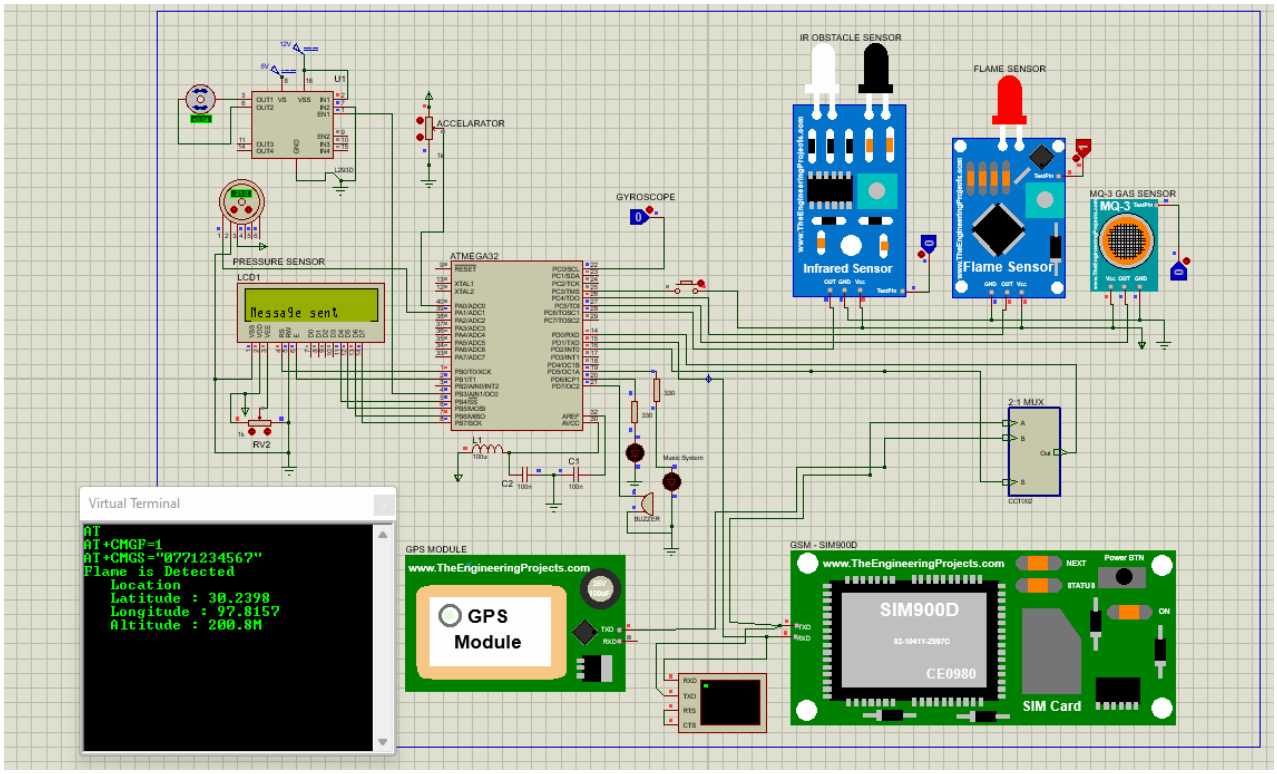

Figure 13 Result when flame is detected

Test 2: When Alcohol is detected

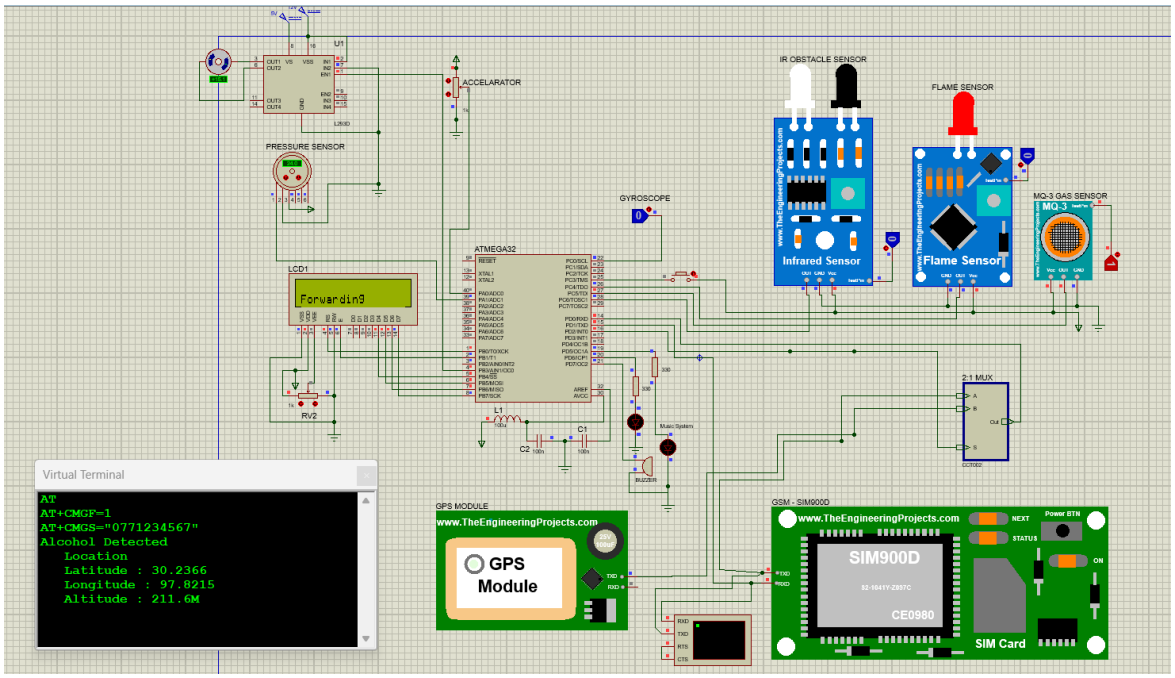

Figure 14 Result when alcohol is detected

Test 3: When vehicle is not moving

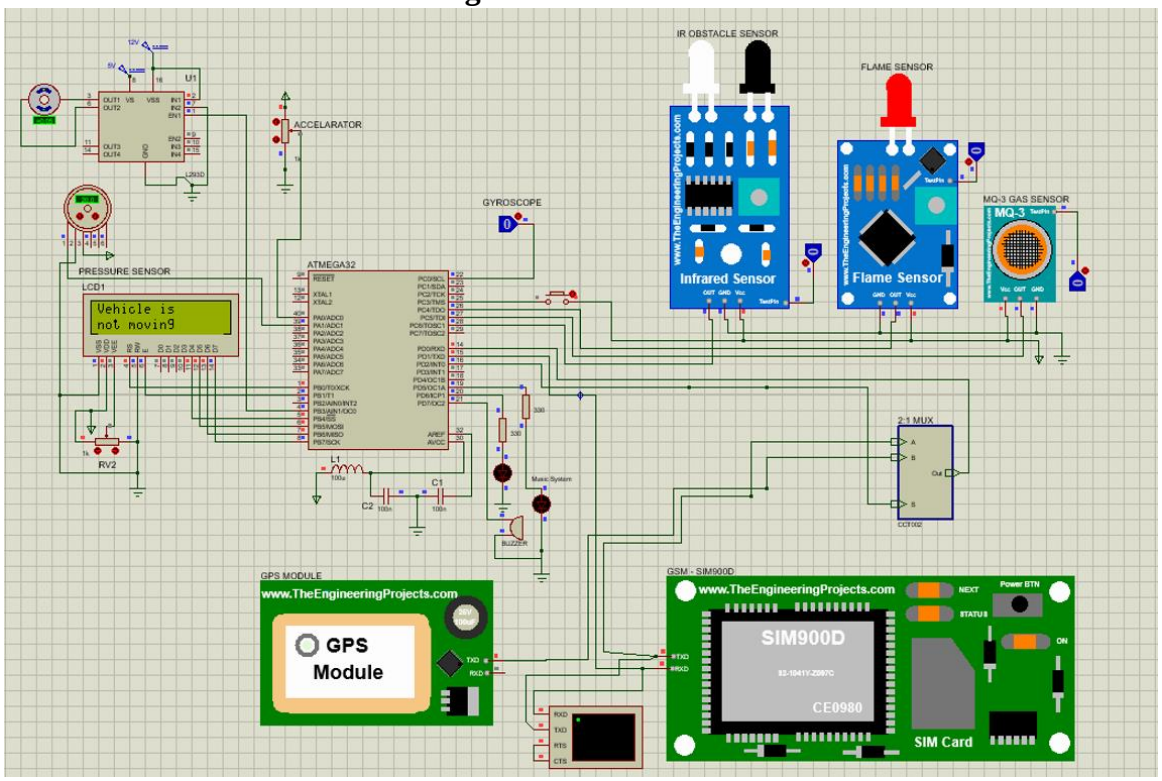

Figure 15 Result when vehicle is not moving

Test 4: When Driver is Sleeping

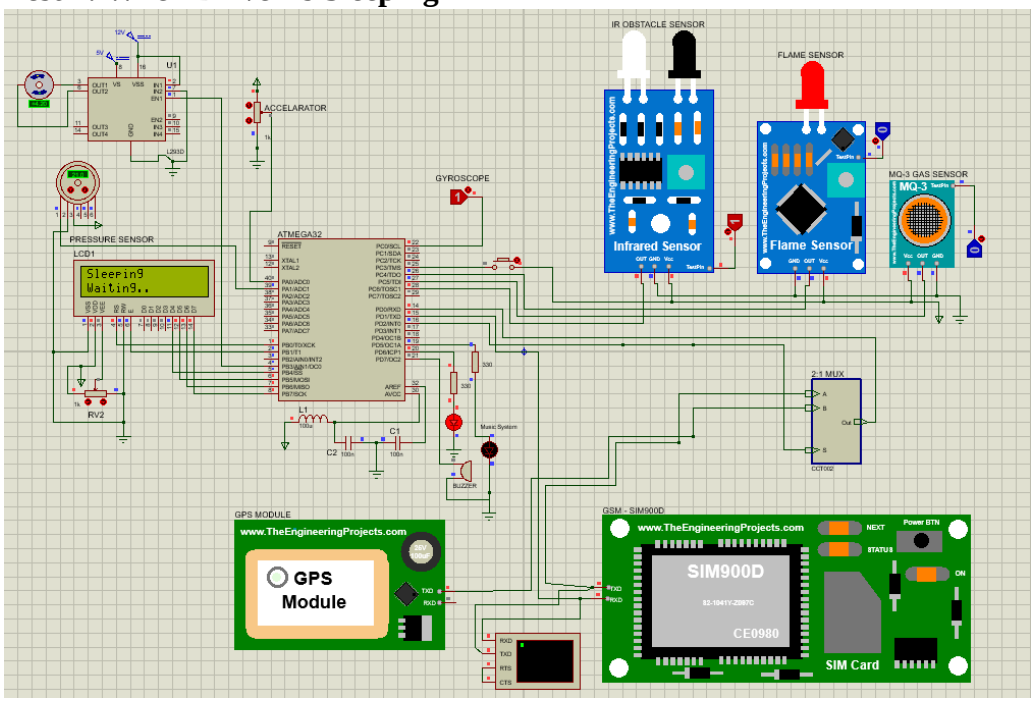

Figure 16 Result when driver is sleeping

Test 5: When Sending Message to the Predefined Phone number

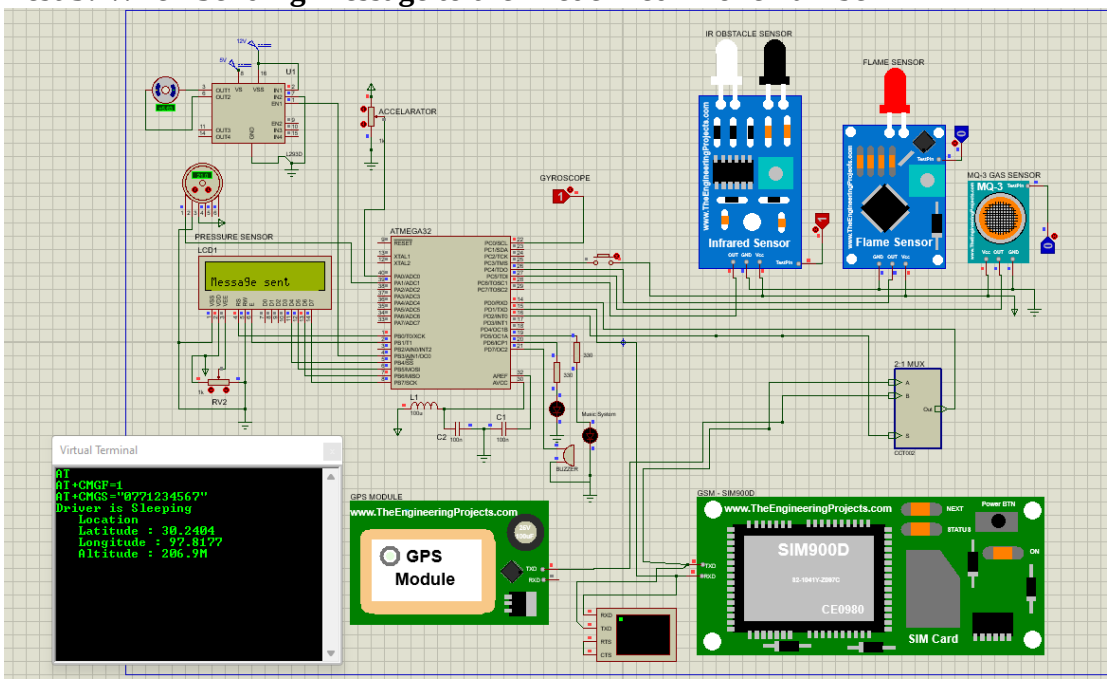

Figure 17 Sending messages after find drowsiness in the driver

Total Code

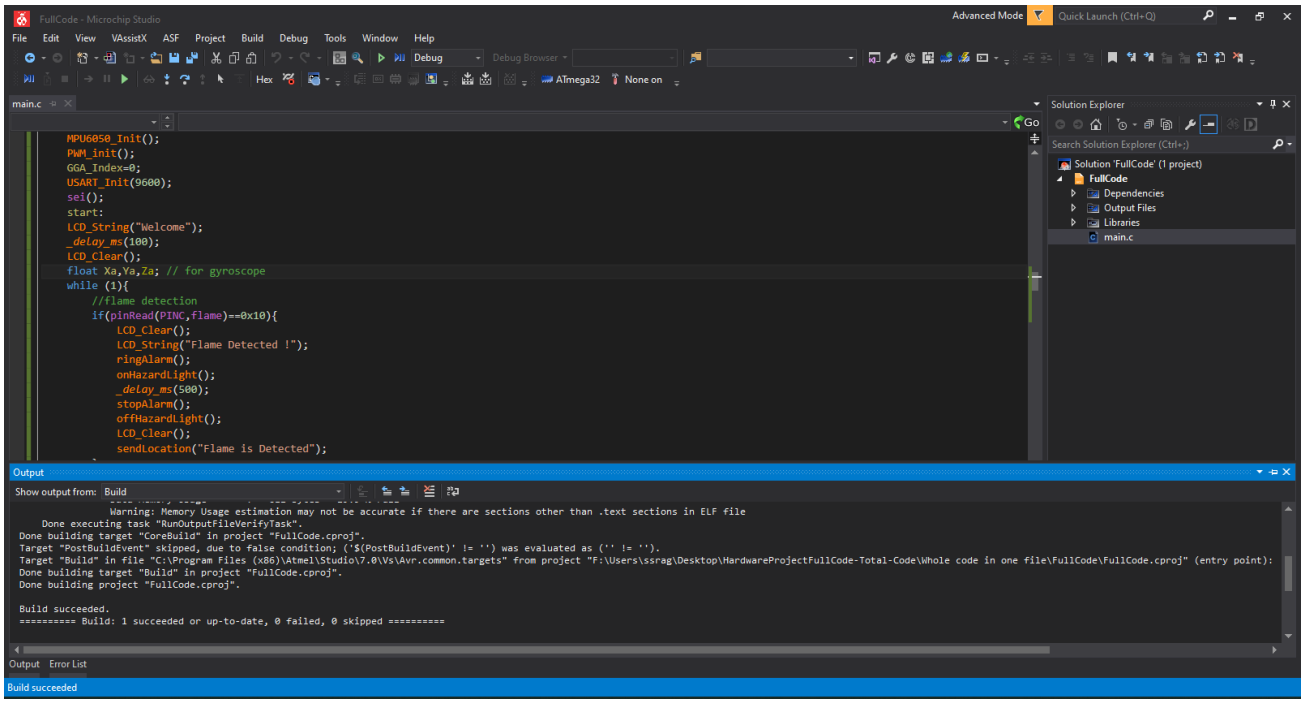

Figure 18 Final Total Code Screenshot

6. Total Estimated Cost

Table 2 Total Estimated Cost

Hardware specifications	Cost
Atmega32-L Microcontroller	540.00
MQ-3 Alcohol Ethanol Gas Sensor	395.00
Eye blinking sensor	150.00
Potentiometer	10.00
DC Motor	400.00
Flame sensor	200.00
Thin-film pressure sensor	725.00
Gyroscope module	660.00
Buzzer	80.00
Radio system/ MP3 player	1600.00
GPS module	1600.00
GSM/GPRS Module	2220.00
LCD display	550.00
Hazard light(LED)	10.00
Total cost	9140.00

7. Further work

Since, Gyroscope Module MPU 6050 and Pressure Sensor FSR 406 are not available in the proteus we could not do simulation for our final code. But we did simulation version of code and proteus design using alternative methods.

8. Appendix A

References

[1] *"BMW model upgrade measures taking effect from the summer of 2013," Bmwgroup.com. [Online]. Available: https://www.press.bmwgroup.com/global/article/detail/T0141144EN/bmw-model-upgrade-measures-taking-effect-from-the-summer-of-2013. [Accessed: 30-Nov-2021].*

[2] *"Audi MediaCenter," Audi-mediacenter.com. [Online]. Available: https://www.audi-mediacenter.com/en. [Accessed: 30-Nov-2021].*

[3] *"2016 Nissan Maxima '4-Door Sports Car' makes global debut at New York International Auto Show," Nissannews.com, 02-Apr-2015. [Online]. Available: https://usa.nissannews.com/en-US/releases/2016-nissan-maxima-4-door-sports-car-makes-global-debut-at-new-york-international-auto-show. [Accessed: 30-Nov-2021].*

[4] *P. by:NeilNie, "Hacking the accelerator — controlling the speed using an Arduino," Neilnie.com, 17-Jan-2018. [Online]. Available: https://neilnie.com/2018/01/17/modifying-the-acceleration-system-control-the-speed-of-the-golf-cart-using-an-arduino/. [Accessed: 30-Nov-2021].*

[5] *"Guide to troubleshooting photocell sensors," Sciencing.com, 30-Mar-2011. [Online]. Available: https://sciencing.com/guide-troubleshooting-photocell-sensors-8142366.html. [Accessed: 30-Nov-2021].*

[6] *M. Abinaya, K. Adithya, B. Kalaivani, P. M. Lavanya, K. Kumaran, and M. Sowmiya, "Drowsiness Detection using Ir sensor," European Journal of Molecular & Clinical Medicine, vol. 7, no. 8, pp. 2106–2114, 2020.*

[7] *Last Minute Engineers, "Send Receive SMS & Call with SIM900 GSM Shield & Arduino," Lastminuteengineers.com, 15-Nov-2018. .*

[8] *S. Z. Nasir, "GPS library for Proteus," Theengineeringprojects.com, 22-Dec-2015. [Online]. Available: https://www.theengineeringprojects.com/2015/12/gps-library-proteus.html. [Accessed: 30-Nov-2021].*

[9] *M. Al-Yassary, K. Billiaert, G. S. Antonarakis, and S. Kiliaridis, "Evaluation of head posture using an inertial measurement unit," Sci. Rep., vol. 11, no. 1, p. 19911, 2021.*

[10] *M. (Max), "The ADC of the AVR," Wordpress.com, 20-Jun-2011. [Online]. Available: https://maxembedded.wordpress.com/2011/06/20/the-adc-of-the-avr/comment-page-1/. [Accessed: 30-Nov-2021].*

[11] *jojo, "DC motor speed control using PWM in AVR Atmega32," Circuitstoday.com, 24-Mar-2017. [Online]. Available: https://www.circuitstoday.com/dc-motor-speed-control-using-pwm-avr. [Accessed: 30-Nov-2021].*

[12] *M. Tresanchez, T. Pallejà, and J. Palacín, "Optical mouse sensor for eye blink detection and pupil tracking: Application in a low-cost eye-controlled pointing device," J. Sens., vol. 2019, pp. 1–19, 2019.*

[13] *Audi-mediaservices.com*. [Online]. Available: https://www.audi-mediaservices.com/publish/ms/content/en/public/hintergrundberichte/2012/03/05/a_statement_about/driver_assistance.html. [Accessed: 21-Jan-2022].

[14] "BMW model upgrade measures taking effect from the summer of 2013," *BMW Group PressClub*. [Online]. Available: https://www.press.bmwgroup.com/global/article/detail/T0141144EN/bmw-model-upgrade-measures-taking-effect-from-the-summer-of-2013. [Accessed: 21-Jan-2022].

[15] "Ford's wake-up call for Europe's sleepy drivers," *Archive.org*. [Online]. Available: https://web.archive.org/web/20110513232258/http://media.ford.com/article_print.cfm?article_id=34562. [Accessed: 21-Jan-2022].

[16] "Driver Attention Monitor," *Honda.com*. [Online]. Available: https://owners.honda.com/vehicles/information/2017/CR-V/features/Driver-Attention-Monitor/1/driver-attention-monitor-pdf. [Accessed: 21-Jan-2022].

[17] "2016 Nissan Maxima '4-Door Sports Car' makes global debut at New York International Auto Show," *Official U.S. Newsroom*, 02-Apr-2015. [Online]. Available: https://usa.nissannews.com/en-US/releases/2016-nissan-maxima-4-door-sports-car-makes-global-debut-at-new-york-international-auto-show. [Accessed: 21-Jan-2022].

9. Appendix B

Individual Contribution

Name of Student : Sagini N. 205092A

I am responsible for Alcohol sensor, GSM/GPRS module. First, I did code for alcohol sensor and successfully simulated it. We use MQ3 alcohol sensor to detect whether the driver is drunk or not. Then I did code for GSM/GPRS module. I selected SIM900A model to send message to responsible person.

My further works are, I did full Schematic diagram and PCB Design using KiCad for MQ3 and GSM/GPRS module. I design power supply, proteus schematic diagram for final simulation. Then I combined all individual's separate files into total coding using Atmel Studio and also I helped to write reports and prepare presentation

Responsible Part: MQ3 alcohol sensor

Technique and Specification

- To detect whether driver is drunk or not, we are using MQ3 alcohol sensor. If alcohol is detected by MQ3 sensor, the alarm will ring and the message "Driver is drunk" and the location also will send to the responsible person.
- Operating voltage: 5V DC
- Current consumption: 150mA
- Detecting concentration 0.05 -10mg/L
- Operating Temperature: -10°C ~ 70°C
- Pins: VCC, Analog Output, Digital Output, Ground

Figure 19 MQ3 Gas Sensor

Responsible Part: SIM900A GSM/GPRS module

Technique and Specification

- To send message, we are using SIM900A GSM/GPRS module.
- Operating voltage: 3.4V to 4.5V DC
- Operating temperature: -30ºC to +80ºC
- Weight: 3.4g
- Operating Current: up to 2A
- Pins: VCC, TXD, RXD, Ground

Figure 20 SIM 900 GSM Module

Schematic Diagram – Whole Component

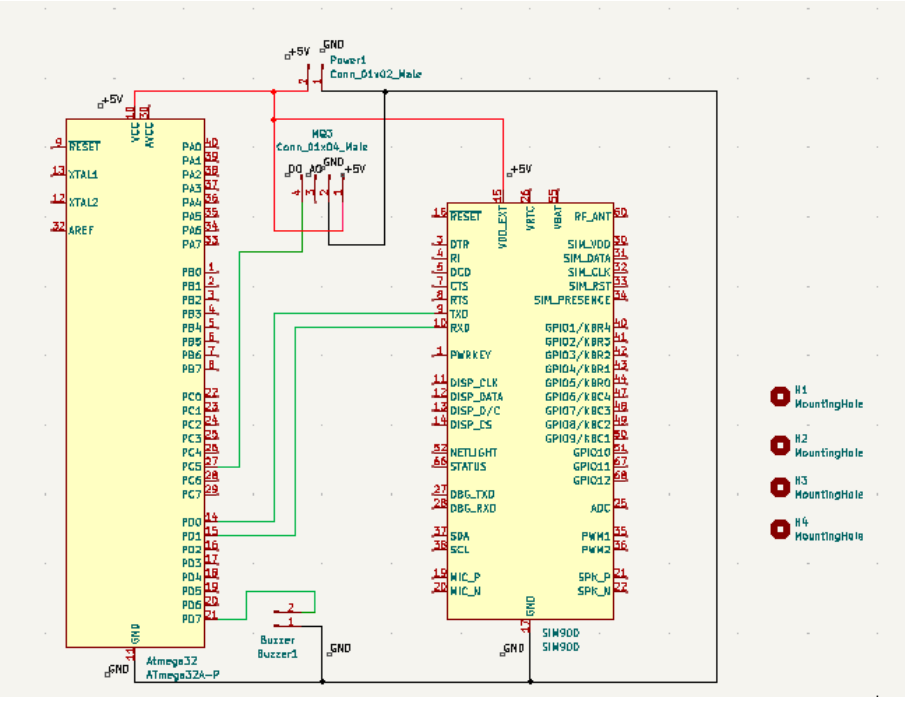

Figure 21 Whole Schematic Diagram 205092A

PCB Design – Whole Component

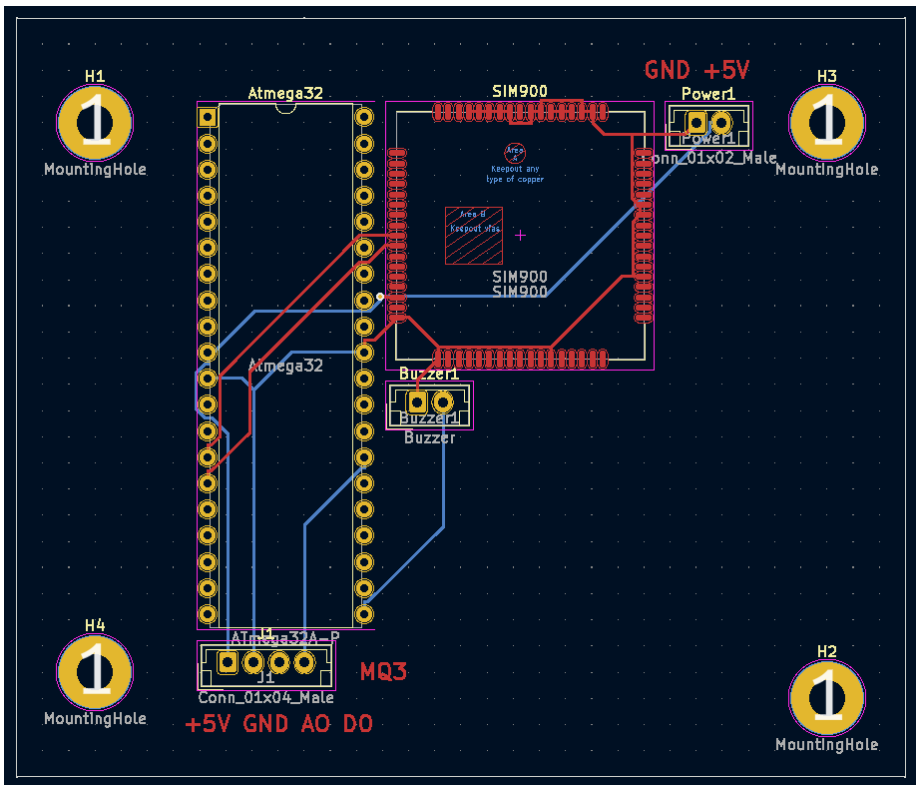

Figure 22 Whole PCB Design 205092A

Code – Whole Component

Main.c

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "usart.h"
char longtitude[15]="97.8177";
char latitude [15]="12.9732";
void sendMessage(char* msg,char* longtitude,char* latitude);

int main(void)
{
	DDRD |= 0x80;
	USART_init(9600); /* initialize USART with 9600 bps*/
	_delay_ms(100);
	while (1)
	{
		if((PINC&0x20)==0x20){		/*If alcohol is detected*/
			PORTD |= 0x80;
			_delay_ms(800);
			sendMessage("Driver is Drunk",longtitude,latitude);
			break;
		}else{
			PORTD &= 0x80;
			_delay_ms(100);
		}
	}
}

void sendMessage(char* msg,char* longtitude,char* latitude)
{
	unsigned char cmd_1[4]="AT";
	unsigned char cmd_2[10]="AT+CMGF=1";
	unsigned char cmd_3[10]="AT+CMGS=";
	char* cmd_4 = msg;
	strcat(cmd_4,"\rLocation\r   Longtitude : ");
	strcat(cmd_4,longtitude);
	strcat(cmd_4,"\r   Latitute : ");
	strcat(cmd_4,latitude);
	unsigned char num[11] = "0771234567";


	for (int i=0;cmd_1[i]!='\0';i++) /*checking communication*/
	{
		USART_txc(cmd_1[i]);
		_delay_ms(100);
	}
	USART_txc('\r'); /*carriage return ---> begining from the first line without going to next line*/
	_delay_ms(200);

	for (int i=0;cmd_2[i]!='\0';i++) /* set the operating mode to SMS text mode*/
	{
		USART_txc(cmd_2[i]);
		_delay_ms(100);
	}
	USART_txc('\r');
	_delay_ms(200);
```

```
		for (int i=0;cmd_3[i]!='\0';i++) /* send SMS in text mode*/
		{
			USART_txc(cmd_3[i]);
			_delay_ms(100);
		}
		UDR="";
		_delay_ms(100);
		for (int i=0;num[i]!='\0';i++) /* SMS to be sent */
		{
			USART_txc(num[i]);
			_delay_ms(100);
		}
		UDR="";
		_delay_ms(100);
		UDR='\r';
		_delay_ms(200);

		for (int i=0;cmd_4[i]!='\0';i++) /* message */
		{
			USART_txc(cmd_4[i]);
			_delay_ms(100);
		}
		_delay_ms(3000);
}
```

Usart.h

```
//UBBR USART Baud Rate Register
#define BAUD_PRESCALE (((F_CPU/(USART_BAUDRATE*16UL)))-1)
void USART_init(long USART_BAUDRATE);
unsigned char USART_rxc();
void USART_txc(char ch);
void USART_send(char *str);

void USART_init(long USART_BAUDRATE)
{
		UCSRB|=(1<<RXEN)|(1<<TXEN)|(1<<RXCIE);
		UCSRC|=(1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);
		UBRRL=BAUD_PRESCALE;

		UBRRH=(BAUD_PRESCALE>>8);
}
unsigned char  USART_rxc()
{
		while(!(UCSRA&(1<<RXC)));	/* while data received is null */
		return (UDR);					/* read data from buffer */
}
void USART_txc(char ch)
{
		while (!(UCSRA&(1<<UDRE)));
		UDR=ch; /* return the given command */
}
void USART_send(char *str)
{
		unsigned char j=0;
		while (str[j]!='\0')			/* Send string till null */
		{
			USART_txc(str[j]);			// send to TX bit by bit
			j++;
		}
}
```

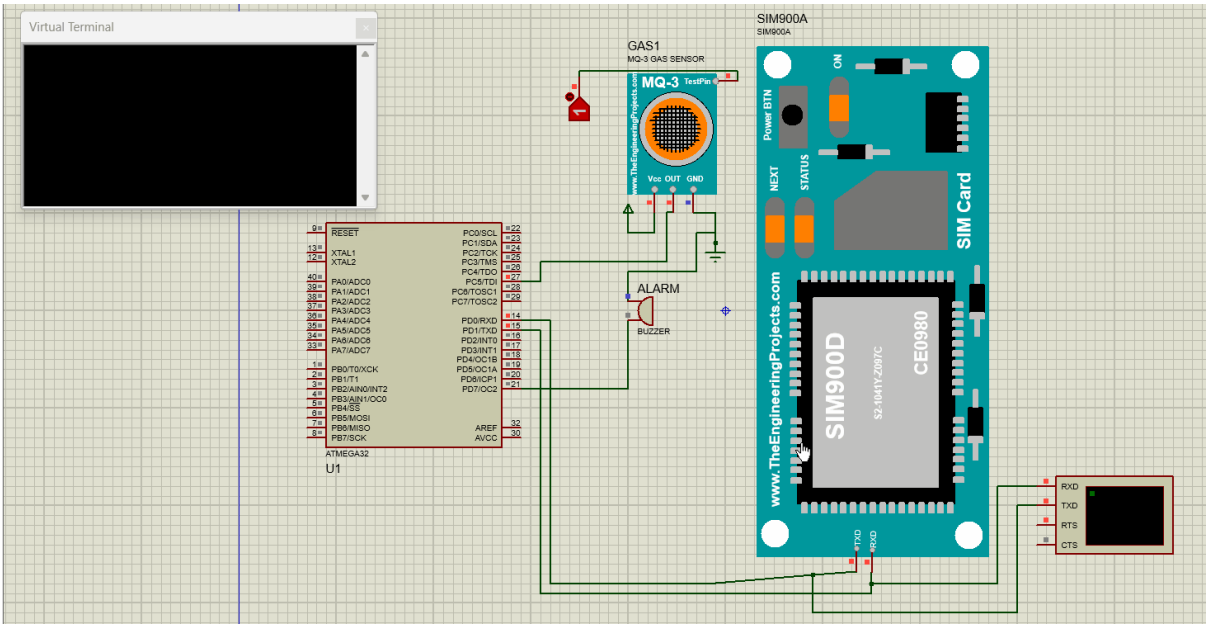

Figure 23 Whole Proteus Simulation 205092A

Name of Student : Raguraj S. 205080K

I am responsible for LED, Buzzer, Switch, Gyroscope Module, DC Motor and Potentiometer. First, I did code for LED, Buzzer and successfully simulated it. We use LED and Buzzer to alert driver and other passengers. Then I did code for Gyroscope module. I selected MPU6050 model for my purpose. We use gyroscope module to get X, Y, Z head position of the driver. By using head position, we can assume whether driver is sleeping or not. I can't run and simulate it. Because Gyroscope module is not available in proteus. So, I used hard coded values to simulate the project using one trigger button. Then I completed coding part and simulation for DC Motor and Potentiometer. I selected L293D as DC motor driver. In our real project, DC Motor, Potentiometer will not available. We use those components here to simulate our whole project. Because main objective of project is based on those two components. We use switch to stop alarm when driver wakeup. Driver should press switch with in 5 seconds. Otherwise, speed of the vehicle will be reduced.

My further works are, I did full Schematic diagram and PCB Design using KiCad. I also did proteus schematic diagram for final simulation. Then I combined all individual's separate files into total coding using Atmel Studio and also I helped to write reports and prepare presentation.

Responsible Part: LED

Technique and Specification

- Here we are using it to alert other driver about the situation by blinking the external hazard light. It will be in the back outside of the car.
- Voltage Range (RED): 1.65 – 2.00V
- Current: 20mA
- Reverse Voltage: 5V
- Number of Pins: 2 (Anode and Cathode)
- Operating Temperature Range: −55° to +100°C

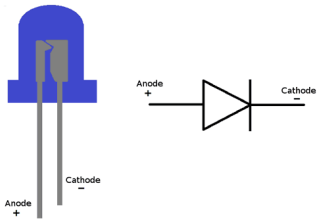

Figure 24 LED

Responsible Part: Buzzer

Technique and Specification

- We are using buzzer to alert the driver
- Voltage Range: 4V to 8V DC
- Rated Voltage: 6V DC
- Rated Current: ≤ 30 mA
- Number of Pins: 2 (Positive and Negative)
- Output: High Pitch Beep Sound - Continuous
- Purpose: To add sound alert to system

Figure 25 Buzzer

Responsible Part: Potentiometer

Technique and Specification

- Vehicle accelerator is also working under the principle of the potentiometer so, We are using potentiometer to simulate the vehicle accelerator.
- Standard Resistance: 0 to 1000 Ohms (1K Ohms)
- Maximum Operating Voltage: 200V
- Operating Temperature: -10° C to +75° C
- Pins: Ground, VCC, Analog signal

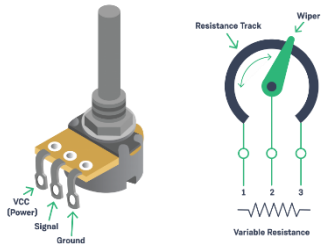

Figure 26 Potentiometer

Responsible Part: DC Motor

Technique and Specification

- We are using DC Motor to simulate vehicle's wheel
- Input Voltage: 5V
- No Load Speed: 12623 RPM
- No Load Current: 0.06A
- Lifetime: 17 Hours
- Number of Pins: 2 (Positive and Negative Terminals)

Figure 27 DC Motor

Responsible Part: Gyroscope Module

Technique and Specification

- We are using Gyroscope module to get head position of the driver
- 3-aixs accelerometer and 3-axis gyroscope values combined
- Power Supply: 3-5V
- Communication: I2C protocol
- Built-in 16-bit ADC provides high accuracy
- Pins: 8 pins

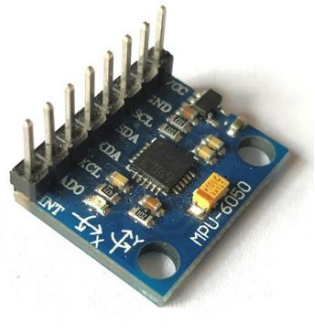

Figure 28 MPU 6050 Gyroscope Module

Responsible Part: Push Button

Technique and Specification

- We are using push button to allow driver to stop the alarm when sleepiness is identified in driver
- Mode of Operation: Tactile feedback
- Power Rating: MAX 50mA 24V DC
- Operating Temperature Range: -20 to +70 ℃

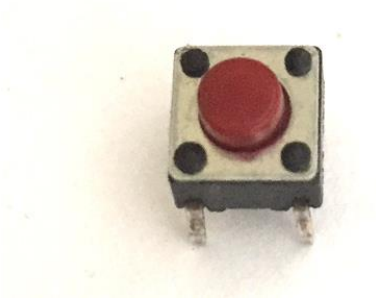

Figure 29 Push Button

Schematic Diagram – Whole Component

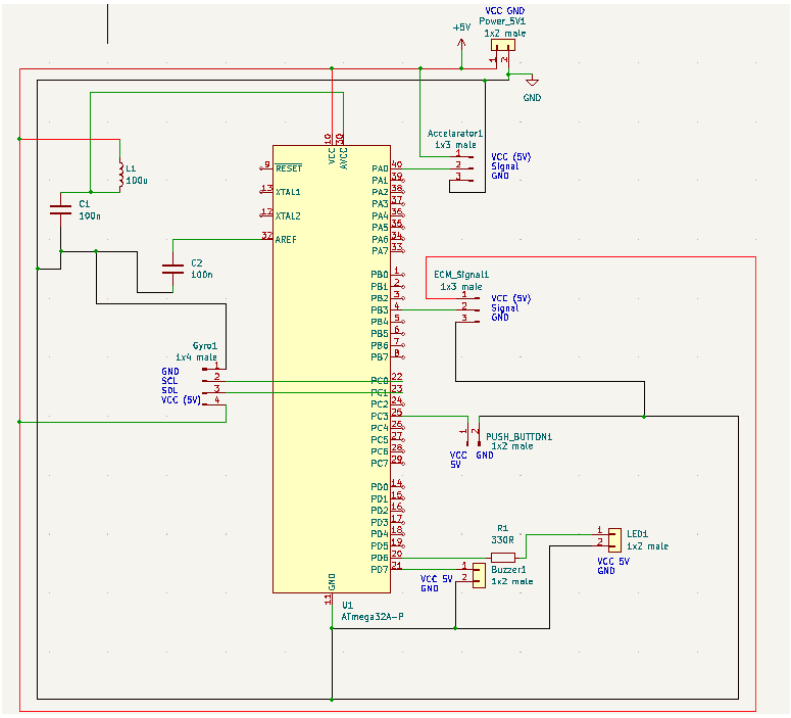

Figure 30 Whole Schematic Diagram 205080K

PCB Design– Whole Component

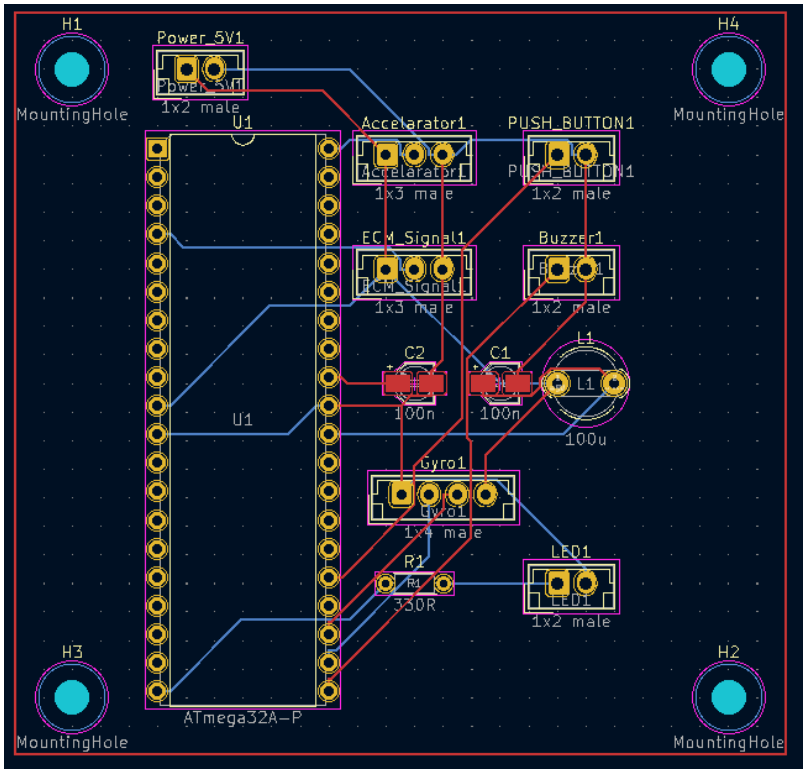

Figure 31 Whole PCB Design 205080K

Code – Whole Component

Main.c

```
#define F_CPU 8000000UL
#define portHigh(reg,pinNumber) reg =  reg | (1<<pinNumber)
#define portLow(reg,pinNumber) reg = reg & (~(1<<pinNumber))
#include <avr/io.h>
#include <util/delay.h>
#include <inttypes.h>
#include <stdlib.h>
#include <stdio.h>
#include "MPU6050_res_define.h"
#include "I2C_Master_H_file.h"
#include "mpu6050.h"
void ADC_Init();
int ADC_Read(char channel);
void PWM_init();
int main(){
		resetCode:
		PWM_init();ADC_Init();I2C_Init(); MPU6050_Init();
		DDRD=0xC0;//set pd6,pd7 as output
		while(1){
				int val=ADC_Read(0);
				float speed=(val/1024.0)*255.0;
				OCR0=(int)speed;
				float X,Y,Z;
				Read_RawValue();
				X = (Acc_x/16384.0)*9.8066;
				Y = (Acc_y/16384.0)*9.8066;
				Z = (Acc_z/16384.0)*9.8066;
				if(isDriverSleepingGyro(X,Y,Z)){
						portHigh(PORTD,6);
						portHigh(PORTD,7);
						//wait for 2sec
						for(int i=1;i<=25;i++){
								if((PINC&0x80)==0x80){
										// if driver stop the alarm between 2sec
										//reset the system
										portLow(PORTD,6);
										portLow(PORTD,7);
										goto resetCode;
								}
								_delay_ms(80);
						}
						OCR0=0; //setting PWM value to 0 in order to slow down the wheel
						while(1){}
				}
		}
}

void PWM_init(){
		/*set fast PWM mode with non-inverted output*/
		TCCR0 = (1<<WGM00) | (1<<WGM01) | (1<<COM01) | (1<<CS00);
		DDRB|=(1<<PB3);  /*set OC0 pin as output*/
}

void ADC_Init(){
		DDRA=0x00;				/* Make ADC port as input */
		ADCSRA = 0x87;				/* Enable ADC, fr/128  */
		ADMUX = 0x40;				/* Vref: Avcc, ADC channel: 0 */
}

int ADC_Read(char channel){
		int Ain,AinLow;
		ADMUX=ADMUX|(channel & 0x0f);		/* Set input channel to read */
```

```
        ADCSRA |= (1<<ADSC);                    /* Start conversion */
        while((ADCSRA&(1<<ADIF))==0); /* Monitor end of conversion interrupt */
        _delay_us(10);
        AinLow = (int)ADCL;             /* Read lower byte*/
        Ain = (int)ADCH*256;            /* Read higher 2 bits and Multiply with weight */
        Ain = Ain + AinLow;
        return(Ain);                    /* Return digital value*/
}
```

Mpu6050.h

```
float Acc_x,Acc_y,Acc_z,Temperature,Gyro_x,Gyro_y,Gyro_z;
void MPU6050_Init()
                /* Gyro initialization function */
{
        _delay_ms(150);
                        /* Power up time >100ms */
        I2C_Start_Wait(0xD0);
        /* Start with device write address */
        I2C_Write(SMPLRT_DIV);
                /* Write to sample rate register */
        I2C_Write(0x07);
                /* 1KHz sample rate */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(PWR_MGMT_1);
                /* Write to power management register */
        I2C_Write(0x01);
                /* X axis gyroscope reference frequency */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(CONFIG);
                /* Write to Configuration register */
        I2C_Write(0x00);
                /* Fs = 8KHz */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(GYRO_CONFIG);
                /* Write to Gyro configuration register */
        I2C_Write(0x18);
                /* Full scale range +/- 2000 degree/C */
        I2C_Stop();

        I2C_Start_Wait(0xD0);
        I2C_Write(INT_ENABLE);
                /* Write to interrupt enable register */
        I2C_Write(0x01);
        I2C_Stop();
}

void MPU_Start_Loc()
{
        I2C_Start_Wait(0xD0);
        /* I2C start with device write address */
        I2C_Write(ACCEL_XOUT_H);
        /* Write start location address from where to read */
        I2C_Repeated_Start(0xD1);
        /* I2C start with device read address */
}

void Read_RawValue()
{
```

```
		MPU_Start_Loc();
			/* Read Gyro values */
		Acc_x = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
		Acc_y = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
		Acc_z = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
		Temperature = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
		Gyro_x = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
		Gyro_y = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Ack());
		Gyro_z = (((int)I2C_Read_Ack()<<8) | (int)I2C_Read_Nack());
		I2C_Stop();
}

int isDriverSleepingGyro(float Xa,float Ya,float Za){
		if((Za>-2.0 && Za<2.0) && (Xa>-3.0 && Xa<3.0) && (Ya>=8.0 && Ya<=9.0)){
			return 0;//driver is not sleeping
		}else{
			return 1;//driver is sleeping
		}
}
```

I2C_Master_H_file.h

```
#ifndef I2C_MASTER_H_FILE_H_
#define I2C_MASTER_H_FILE_H_
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <math.h>
#define SCL_CLK 100000L
#define BITRATE(TWSR)		((F_CPU/SCL_CLK)-16)/(2*pow(4,(TWSR&((1<<TWPS0)|(1<<TWPS1)))))

void I2C_Init();
uint8_t  I2C_Start(char);
uint8_t  I2C_Repeated_Start(char);
void I2C_Stop();
void I2C_Start_Wait(char);
uint8_t  I2C_Write(char);
char I2C_Read_Ack();
char I2C_Read_Nack();
#endif
```

I2C_Master_C_file.h

```
#include "I2C_Master_H_file.h"
void I2C_Init()
{
		TWBR = BITRATE(TWSR = 0x00);
}


uint8_t I2C_Start(char slave_write_address)
{
		uint8_t status;
		TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
		while (!(TWCR & (1<<TWINT)));
		status = TWSR & 0xF8;
		if (status != 0x08)
		return 0;
		TWDR = slave_write_address;
		TWCR = (1<<TWEN)|(1<<TWINT);
		while (!(TWCR & (1<<TWINT)));
		status = TWSR & 0xF8;
		if (status == 0x18)
		return 1;
		if (status == 0x20)
```

```
		return 2;
		else
		return 3;
}

uint8_t I2C_Repeated_Start(char slave_read_address)
{
		uint8_t status;
		TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
		while (!(TWCR & (1<<TWINT)));
		status = TWSR & 0xF8;
		if (status != 0x10)
		return 0;
		TWDR = slave_read_address;
		TWCR = (1<<TWEN)|(1<<TWINT);
		while (!(TWCR & (1<<TWINT)));
		status = TWSR & 0xF8;
		if (status == 0x40)
		return 1;
		if (status == 0x20)
		return 2;
		else
		return 3;
}

void I2C_Stop()
{
		TWCR=(1<<TWSTO)|(1<<TWINT)|(1<<TWEN);
		while(TWCR & (1<<TWSTO));
}

void I2C_Start_Wait(char slave_write_address)
{
		uint8_t status;
		while (1)
		{
			TWCR = (1<<TWSTA)|(1<<TWEN)|(1<<TWINT);
			while (!(TWCR & (1<<TWINT)));
			status = TWSR & 0xF8;
			if (status != 0x08)
			continue;
			TWDR = slave_write_address;
			TWCR = (1<<TWEN)|(1<<TWINT);
			while (!(TWCR & (1<<TWINT)));
			status = TWSR & 0xF8;
			if (status != 0x18 )
			{
				I2C_Stop();
				continue;
			}
			break;
		}
}

uint8_t I2C_Write(char data)
{
		uint8_t status;
		TWDR = data;
		TWCR = (1<<TWEN)|(1<<TWINT);
		while (!(TWCR & (1<<TWINT)));
		status = TWSR & 0xF8;
		if (status == 0x28)
		return 0;
		if (status == 0x30)
		return 1;
		else
		return 2;
```

```
}

char I2C_Read_Ack()
{
		TWCR=(1<<TWEN)|(1<<TWINT)|(1<<TWEA);
		while (!(TWCR & (1<<TWINT)));
		return TWDR;
}

char I2C_Read_Nack()
{
		TWCR=(1<<TWEN)|(1<<TWINT);
		while (!(TWCR & (1<<TWINT)));
		return TWDR;
}
```

MPU6050_res_define.h

```
#ifndef MPU6050_RES_DEFINE_H_
#define MPU6050_RES_DEFINE_H_
#include <avr/io.h>
#define XG_OFFS_TC 0x00
#define YG_OFFS_TC 0x01
#define ZG_OFFS_TC 0x02
#define X_FINE_GAIN 0x03
#define Y_FINE_GAIN 0x04
#define Z_FINE_GAIN 0x05
#define XA_OFFS_H 0x06
#define XA_OFFS_L_TC 0x07
#define YA_OFFS_H 0x08
#define YA_OFFS_L_TC 0x09
#define ZA_OFFS_H 0x0A
#define ZA_OFFS_L_TC 0x0B
#define XG_OFFS_USRH 0x13
#define XG_OFFS_USRL 0x14
#define YG_OFFS_USRH 0x15
#define YG_OFFS_USRL 0x16
#define ZG_OFFS_USRH 0x17
#define ZG_OFFS_USRL 0x18
#define SMPLRT_DIV 0x19
#define CONFIG 0x1A
#define GYRO_CONFIG 0x1B
#define ACCEL_CONFIG 0x1C
#define FF_THR 0x1D
#define FF_DUR 0x1E
#define MOT_THR 0x1F
#define MOT_DUR 0x20
#define ZRMOT_THR 0x21
#define ZRMOT_DUR 0x22
#define FIFO_EN 0x23
#define I2C_MST_CTRL 0x24
#define I2C_SLV0_ADDR 0x25
#define I2C_SLV0_REG 0x26
#define I2C_SLV0_CTRL 0x27
#define I2C_SLV1_ADDR 0x28
#define I2C_SLV1_REG 0x29
#define I2C_SLV1_CTRL 0x2A
#define I2C_SLV2_ADDR 0x2B
#define I2C_SLV2_REG 0x2C
#define I2C_SLV2_CTRL 0x2D
#define I2C_SLV3_ADDR 0x2E
#define I2C_SLV3_REG 0x2F
#define I2C_SLV3_CTRL 0x30
#define I2C_SLV4_ADDR 0x31
#define I2C_SLV4_REG 0x32
#define I2C_SLV4_DO 0x33
```

```
#define I2C_SLV4_CTRL 0x34
#define I2C_SLV4_DI 0x35
#define I2C_MST_STATUS 0x36
#define INT_PIN_CFG 0x37
#define INT_ENABLE 0x38
#define DMP_INT_STATUS 0x39
#define INT_STATUS 0x3A
#define ACCEL_XOUT_H 0x3B
#define ACCEL_XOUT_L 0x3C
#define ACCEL_YOUT_H 0x3D
#define ACCEL_YOUT_L 0x3E
#define ACCEL_ZOUT_H 0x3F
#define ACCEL_ZOUT_L 0x40
#define TEMP_OUT_H 0x41
#define TEMP_OUT_L 0x42
#define GYRO_XOUT_H 0x43
#define GYRO_XOUT_L 0x44
#define GYRO_YOUT_H 0x45
#define GYRO_YOUT_L 0x46
#define GYRO_ZOUT_H 0x47
#define GYRO_ZOUT_L 0x48
#define EXT_SENS_DATA_00 0x49
#define EXT_SENS_DATA_01 0x4A
#define EXT_SENS_DATA_02 0x4B
#define EXT_SENS_DATA_03 0x4C
#define EXT_SENS_DATA_04 0x4D
#define EXT_SENS_DATA_05 0x4E
#define EXT_SENS_DATA_06 0x4F
#define EXT_SENS_DATA_07 0x50
#define EXT_SENS_DATA_08 0x51
#define EXT_SENS_DATA_09 0x52
#define EXT_SENS_DATA_10 0x53
#define EXT_SENS_DATA_11 0x54
#define EXT_SENS_DATA_12 0x55
#define EXT_SENS_DATA_13 0x56
#define EXT_SENS_DATA_14 0x57
#define EXT_SENS_DATA_15 0x58
#define EXT_SENS_DATA_16 0x59
#define EXT_SENS_DATA_17 0x5A
#define EXT_SENS_DATA_18 0x5B
#define EXT_SENS_DATA_19 0x5C
#define EXT_SENS_DATA_20 0x5D
#define EXT_SENS_DATA_21 0x5E
#define EXT_SENS_DATA_22 0x5F
#define EXT_SENS_DATA_23 0x60
#define MOT_DETECT_STATUS 0x61
#define I2C_SLV0_DO 0x63
#define I2C_SLV1_DO 0x64
#define I2C_SLV2_DO 0x65
#define I2C_SLV3_DO 0x66
#define I2C_MST_DELAY_CTRL 0x67
#define SIGNAL_PATH_RESET 0x68
#define MOT_DETECT_CTRL 0x69
#define USER_CTRL 0x6A
#define PWR_MGMT_1 0x6B
#define PWR_MGMT_2 0x6C
#define BANK_SEL 0x6D
#define MEM_START_ADDR 0x6E
#define MEM_R_W 0x6F
#define DMP_CFG_1 0x70
#define DMP_CFG_2 0x71
#define FIFO_COUNTH 0x72
#define FIFO_COUNTL 0x73
#define FIFO_R_W 0x74
#define WHO_AM_I 0x75

#endif /* MPU6050_RES_DEFINE_H_ */
```

Note: This is the real code we used in the full code. As we don't have any Gyroscope module for proteus, we could not simulate and get results. In the individual recording, I explained simulation version of this code. Here I attached screenshot of proteus simulation.

-Thankyou

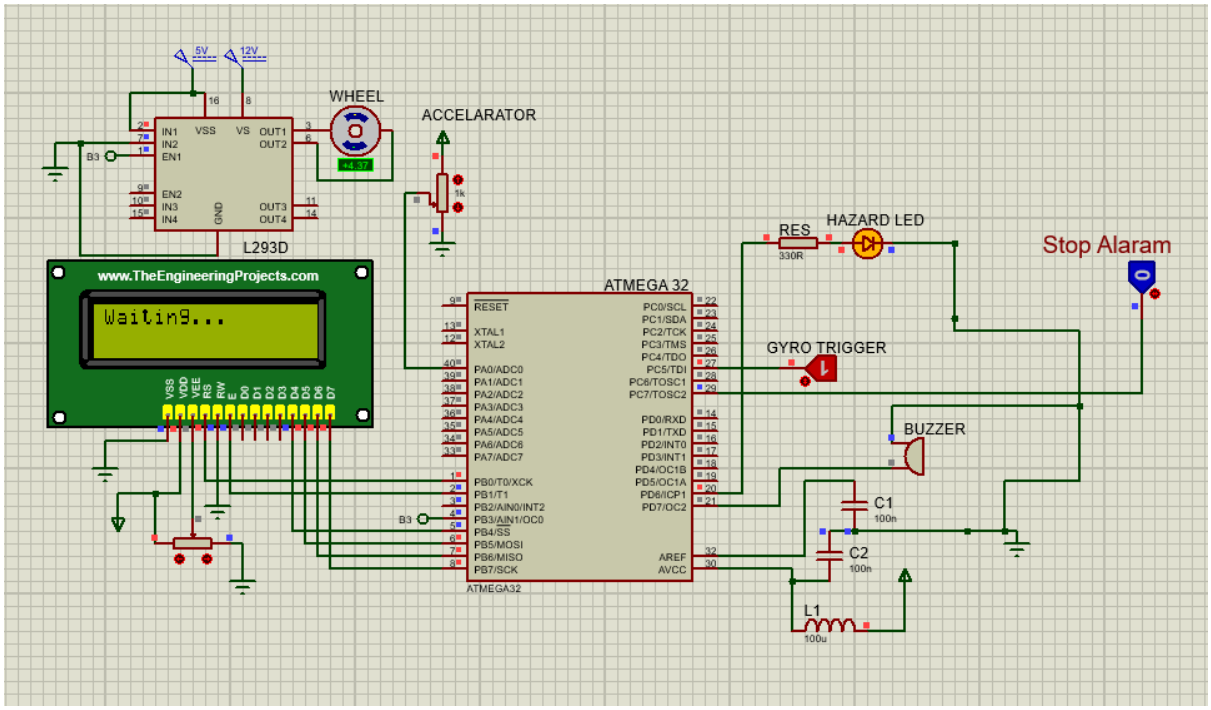

Figure 32 Whole Proteus Simulation 205080K

Name of Student : Ishvini A. 205039U

I am responsible for GPS Module and flame sensor. First, I did code for flame sensor and successfully simulated it. We use flame sensor to detect the flame inside the car. Then I did code for GPS module. I selected NEO 6MV2 GPS model for my purpose. We use GPS module to get location of the driver.

My further works are, I did 3D design for our project. Then I did full Schematic diagram and PCB Design using KiCad. I also did proteus schematic diagram for final simulation. I helped to write reports and prepare presentation.

Responsible Part: Flame sensor

Technique and Specification

- To detect whether flame is detected or not, we are using AB007 flame sensor. If flame is detected by AB007 sensor, the Hazard light will blink, alarm will ring and the message “Flame is detected” and the location also will send to the responsible person.
- Operating Voltage: 3.3V – 5V
- Operating Current: 15 mA
- Output type: Digital and Analog output
- Detection angle: 0 – 60 degrees
- Pins: VCC, Analog Output, Digital Output, Ground

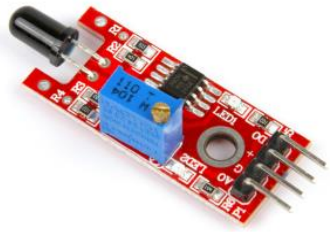

Figure 33 AB007 Flame Sensor

Responsible Part: GPS module

Technique and Specification

- To Find the location, we are using NEO 6MV2 GPS module.
- Supply voltage: 3.6V
- Operating current: 45mA
- Operating temperature range: -40ºC TO 85°C
- Pins: VCC, TXD, RXD, Ground

Figure 34 NEO-6MV2 GPS Module

Schematic Diagram – Whole Component

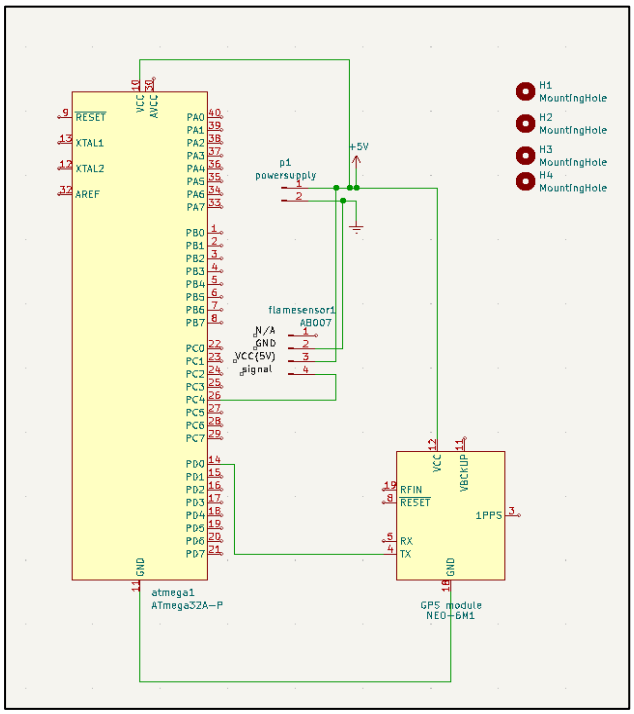

Figure 35 Whole Schematic Diagram 205039U

PCB Design – Whole Component

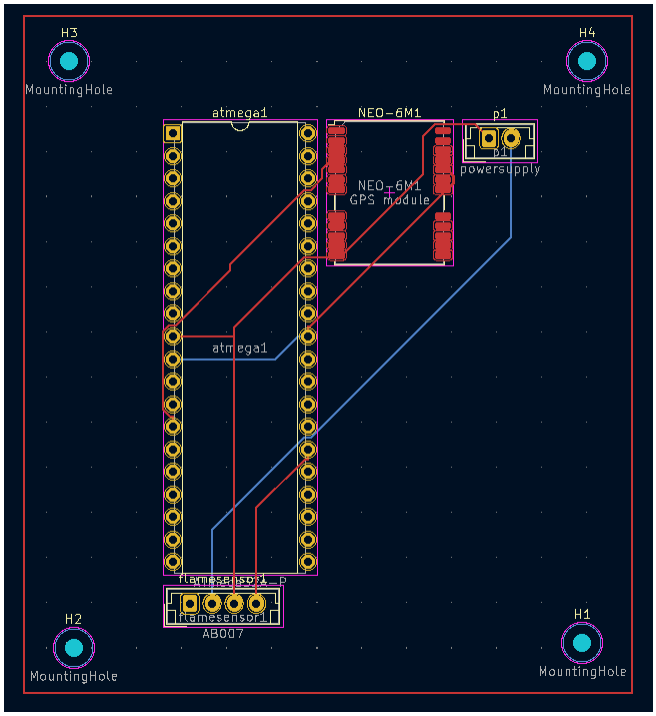

Figure 36 Whole PCB Design 205039U

Code – Whole Component

Main.c

```
#define F_CPU 8000000UL
#define SREG   _SFR_IO8(0x3f)
#include <avr/io.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "LCD.h"
#include "USART_Interrupt.h"
#include "Track.h"

int main() {

		DDRC|=0x20;
		GGA_Index=0;
		LCD_Init();
		USART_Init(9600);
		sei();



		/*		Distance=Speed*Time
		speed is the radio signal equal to speed of light(3*10^8)
		Time required for a signal to travel from the satellite to the receiver.
		subtracting the sent time from the received time, we can determine the travel time.
		*/

		while (1)
		{
					LCD_String("Flame:");
					_delay_ms(200);
					if((PINC&0x10)==0x10){
							PORTC=0x20;
							LCD_String_xy(1,3,"Detected");
							_delay_ms(100);
							LCD_Clear();
							_delay_ms(1000);
							LCD_String_xy(2,0,"Lat: ");
		//Display as"Lat"
							get_latitude(GGA_Pointers[0]);		// Extract Latitude- convert raw
latitude value into degree format and pass that value as string
							LCD_String(degrees_buffer);		/* display latitude in degree */


							LCD_String_xy(1,0,"Long: ");
							get_longitude(GGA_Pointers[2]);		/* Extract Longitude */
							LCD_String(degrees_buffer);		/* display longitude in degree */
							//memset(degrees_buffer,0,degrees_buffer_size);//memset is an in-build
function used to fill a block of memory with a particular value
							_delay_ms(1000);
							LCD_Clear();


							LCD_String_xy(2,0,"Alt: ");
		//Display as"Alt"
							get_altitude(GGA_Pointers[7]);		/* Extract Altitude in meters*/
							LCD_String(Altitude_Buffer);				//display the altitude
value
							_delay_ms(1000);
							LCD_Clear();
```

```
                                        }else{
                                        PORTC=0x00;
                                        LCD_String_xy(2,3,"Not Detected");
                                        _delay_ms(1000);
                                        LCD_Clear();
                                }
        }
        return 0;
}
```

Track.h

```
void convert_to_degrees(char *);

#define Buffer_Size 150
#define degrees_buffer_size 20

char Latitude_Buffer[15],Longitude_Buffer[15],Altitude_Buffer[8];//define the string size
char degrees_buffer[degrees_buffer_size];               /* save latitude or longitude in degree - degrees_buffer[20]*/
char GGA_Buffer[Buffer_Size];                   /* save GGA string - GGA_Buffer[150] */
uint8_t GGA_Pointers[20];                       /* to store instances of ',' */
char GGA_CODE[3];

volatile uint16_t GGA_Index, CommaCounter; // unsigned 16-bit integer - integers between 0 and 65,535

bool        IsItGGAString           = false,
flag1                               = false,
flag2                               = false;



void get_latitude(uint16_t lat_pointer){  //unsigned 16-bit integer - integers between 0 and 65,535
        cli(); //Command Line Interface - CLIs accept as input commands that are entered by keyboard
        uint8_t lat_index; //unsigned 8-bit integer- integer has a range of 0 to 255
        uint8_t index = lat_pointer+1;//unsigned 8-bit integer- integer has a range of 0 to 255
        lat_index=0;

        /* parse Latitude in GGA string stored in buffer */
        for(;GGA_Buffer[index]!=',';index++){
                Latitude_Buffer[lat_index]= GGA_Buffer[index];
                lat_index++;
        }

        Latitude_Buffer[lat_index++] = GGA_Buffer[index++];
        Latitude_Buffer[lat_index]= GGA_Buffer[index];          /* get direction */
        convert_to_degrees(Latitude_Buffer);   // convert raw latitude into degree format and pass that value as string
        sei(); //It is a macro that executes an assembler instruction to enable interrupts.
}

void get_longitude(uint16_t long_pointer){
        cli(); //Command Line Interface - CLIs accept as input commands that are entered by keyboard
        uint8_t long_index;  //unsigned 8-bit integer- integer has a range of 0 to 255
        uint8_t index = long_pointer+1;
        long_index=0;

        /* parse Longitude in GGA string stored in buffer */
        for( ; GGA_Buffer[index]!=','; index++){
                Longitude_Buffer[long_index]= GGA_Buffer[index];
                long_index++;
        }

        Longitude_Buffer[long_index++] = GGA_Buffer[index++];
        Longitude_Buffer[long_index]   = GGA_Buffer[index]; /* get direction */
        convert_to_degrees(Longitude_Buffer); // convert raw longitude into degree format and pass that value as string
```

```
		sei(); //It is a macro that executes an assembler instruction to enable interrupts.
}

void get_altitude(uint16_t alt_pointer){  //unsigned 16-bit integer - integers between 0 and 65,535
		cli(); //Command Line Interface - CLIs accept as input commands that are entered by keyboard
		uint8_t alt_index;  //unsigned 8-bit integer- integer has a range of 0 to 255
		uint8_t index = alt_pointer+1;
		alt_index=0;
		/* parse Altitude in GGA string stored in buffer */
		for( ; GGA_Buffer[index]!=','; index++){
				Altitude_Buffer[alt_index]= GGA_Buffer[index];
				alt_index++;
		}

		Altitude_Buffer[alt_index]   = GGA_Buffer[index+1];
		sei();//It is a macro that executes an assembler instruction to enable interrupts.
}


void convert_to_degrees(char *raw){

		double value;
		float decimal_value,temp;

		int32_t degrees;

		float position;
		value = atof(raw); /* convert string into float for conversion */

		/* convert raw latitude/longitude into degree format */
		decimal_value = (value/100);
		degrees = (int)(decimal_value);
		temp = (decimal_value - (int)decimal_value)/0.6;
		position = (float)degrees + temp;

		dtostrf(position, 6, 4, degrees_buffer); /* dtostrf is a function that convert float value into string.Here the
position is a float value and it convert as a string degree buffer variable*/
}



ISR (USART_RXC_vect)
{
		uint8_t oldsrg = SREG;  //unsigned 8-bit integer- integer has a range of 0 to 255
		cli();                                                          //Command Line Interface - CLIs accept as
input commands that are entered by keyboard
		char received_char = UDR;

		if(received_char =='$'){                                /* check for '$' */
				GGA_Index = 0;
				CommaCounter = 0;
				IsItGGAString = false;
		}
		else if(IsItGGAString == true){                         /* if true save GGA info. into buffer */
				if(received_char == ',' ) GGA_Pointers[CommaCounter++] = GGA_Index;     /* store instances of
',' in buffer */
				GGA_Buffer[GGA_Index++] = received_char;
		}
		else if(GGA_CODE[0] == 'G' && GGA_CODE[1] == 'G' && GGA_CODE[2] == 'A'){    /* check for GGA
string */
				IsItGGAString = true;
				GGA_CODE[0] = 0; GGA_CODE[1] = 0; GGA_CODE[2] = 0;
		}
		else{
				GGA_CODE[0] = GGA_CODE[1];  GGA_CODE[1] = GGA_CODE[2]; GGA_CODE[2] =
received_char;
		}
```

```
        SREG = oldsrg;
}
```

Usart_Interrupt.h

```
#define USART_INTERRUPT_H_
#define F_CPU 8000000UL                                         /* Define CPU clock
Frequency e.g. here its 8MHz */
#include <avr/io.h>                                             /* Include AVR std.
library file */
#include <avr/interrupt.h>
#define BAUD_PRESCALE (((F_CPU / (BAUDRATE * 16UL))) - 1)       /* Define prescale value */

void USART_Init(unsigned long BAUDRATE)                         /* USART initialize function */
{
        UCSRB |= (1 << RXEN) | (1 << TXEN) | (1 << RXCIE);                      /* Enable
USART transmitter and receiver */
        UCSRC |= (1 << URSEL)| (1 << UCSZ0) | (1 << UCSZ1);     /* Write USCRC for 8 bit data and 1 stop bit */
        UBRRL = BAUD_PRESCALE;
        /* Load UBRRL with lower 8 bit of prescale value */
        UBRRH = (BAUD_PRESCALE >> 8);                                    /* Load
UBRRH with upper 8 bit of prescale value */
}

char USART_RxChar()
        /* Data receiving function */
{
        while (!(UCSRA & (1 << RXC)));                          /* Wait until new data
receive */
        return(UDR);
                /* Get and return received data */
}

void USART_TxChar(char data)                                    /* Data transmitting
function */
{
        UDR = data;
                        /* Write data to be transmitting in UDR */
        while (!(UCSRA & (1<<UDRE)));                           /* Wait until data
transmit and buffer get empty */
}

void USART_SendString(char *str)                                /* Send string of USART data
function */
{
        int i=0;
        while (str[i]!=0)
        {
                USART_TxChar(str[i]);                                    /* Send
each char of string till the NULL */
                i++;
        }
}
```

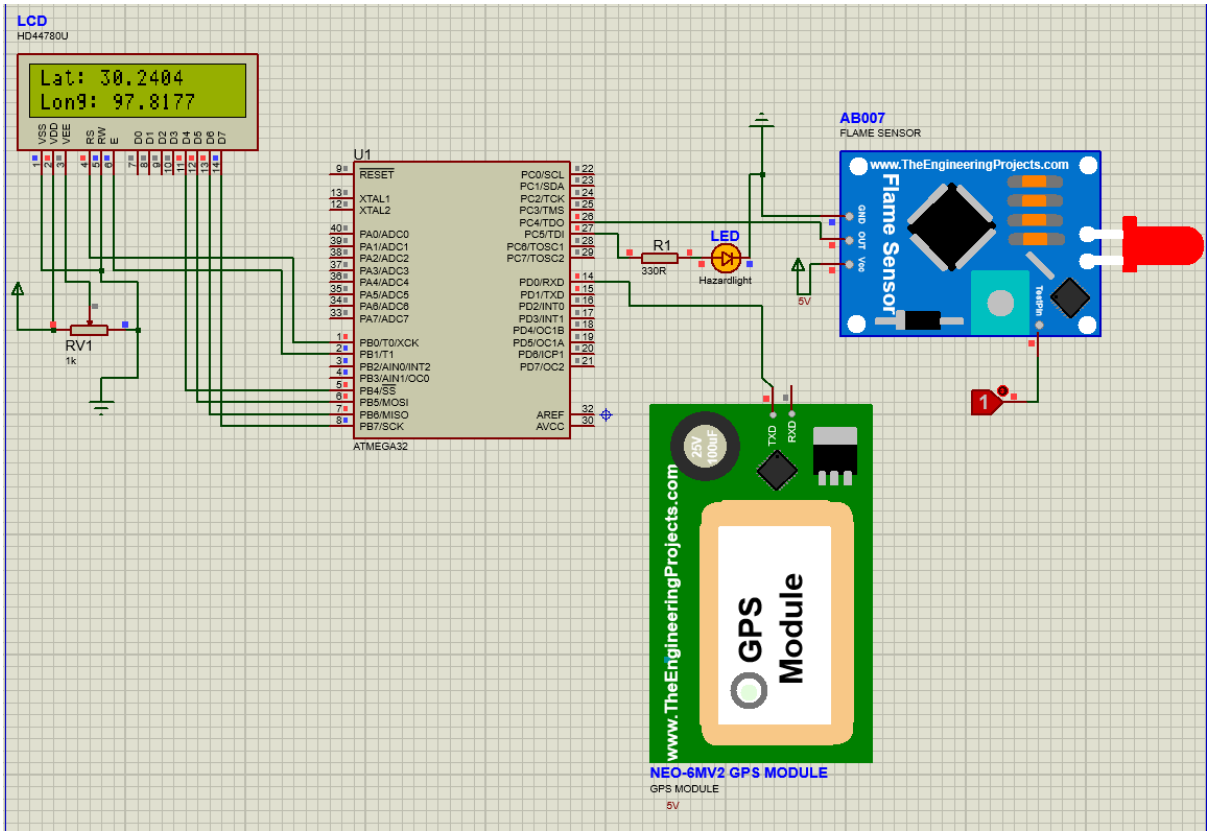

Figure 37 Whole Proteus Simulation 205039U

Name of Student : Kaneshan T. 205048V

I am responsible for the FSR406 thin-film pressure sensor, Eye blink sensor – TCRT5000, LCD(16x2) Display and radio system. I place a thin-film pressure sensor on the acceleration pedal to ensure the force is given by the driver when driving. If the driver gives force to the pedal the resistance of the sensor will be decreased. By this, we can validate the vehicle movement. I did coding for this sensor by using the ADC concept. When simulating in the proteus to check whether my code is correct or not, unfortunately, I couldn't find the library for the FSR406 module. Then I use the MPX4115 pressure sensor for simulation purposes. I use the TCRT5000 eye blink sensor on the headband before the eyes to check whether the eyes are closed or not. If the driver closes his eyes for more than 2 seconds continuously we assume that driver is sleeping. Then the radio system will play to wake up the driver. I used the external library for the eyeblink sensor as there are no inbuilt libraries in the proteus. And I use an LCD display to give a warning to the driver via a visible display. When our system detects the fire presence or alcohol intensity or driver's sleepiness it will show a warning message to the driver on the LCD display.

For the coding part, I used Atmel studio. And for the simulation, I used proteus software. I used KiCad for the schematic and PCB design for my individual components. I refer to the details of my individual works by using datasheets, websites and YouTube channels. Furthermore, I did the 3D design of our system. To do that I searched about the features of the software and spent a lot of time to get practised by using YouTube tutorials and websites.

Responsible Part: Thin-film pressure sensor - FSR-406

Technique and Specification

- To detect whether the vehicle is moving or not, we are using FSR-406 thin-film pressure sensor. We place this thin film pressure sensor on the acceleration pedal. If the driver give force to the acceleration pedal, thin film sensor pressure will be increased then we can validate the vehicle movement.
- Force range : 0.1N -10N
- Size :43.69 x 43.69mm
- Active Area: 38.1mm x 38.1mm
- Thickness range -0.2 - 1.25 mm
- Nominal thickness: 0.54 mm
- Operating temperature -30 - +70 ºC
- Stand-Off Resistance : >10M ohms
- Pins : VCC, GND, Analog Out

Figure 38 FSR 406 Pressure Sensor

Responsible Part: Eye blink sensor – TCRT5000

Technique and Specification

- To check the driver's drowsiness, we are using TCRT 5000 eyeblink sensor.
- We place this sensor on the headband before the eyes to check whether the eyes are closed or not. From this we can identify whether driver is sleeping or not.
- reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light.
- Reverse voltage - 5V
- Forward current - 60mA
- Operating temperature - 25 to +85 °C
- Peak operating distance - 2.5 mm
- Operating wavelength - 950nm
- Pins – VCC, GND, Digital Output, Analog Output

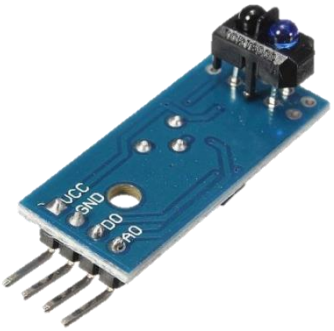

Figure 39 TCRT5000 Eye Blink Sensor

Responsible Part: Lcd Display (16x2) HD44780U

Technique and Specification

- To show the warning message to the driver, we are using lcd display.
- Operating Voltage is 4.7V to 5.3V
- Current consumption is 1mA
- Consists of two rows and each row can print 16 characters.
- Alphanumeric LCD display module that can display alphabets and numbers.
- Each character is built by a 5×8 pixel box.

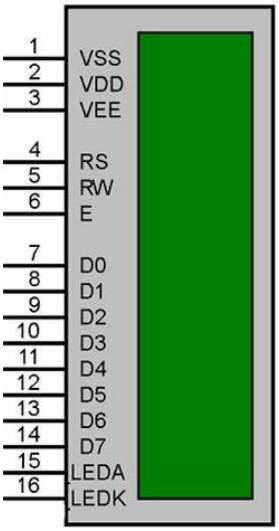

Figure 40 HD44780U LCD Display 16x2

Shematic Diagram – Whole Component

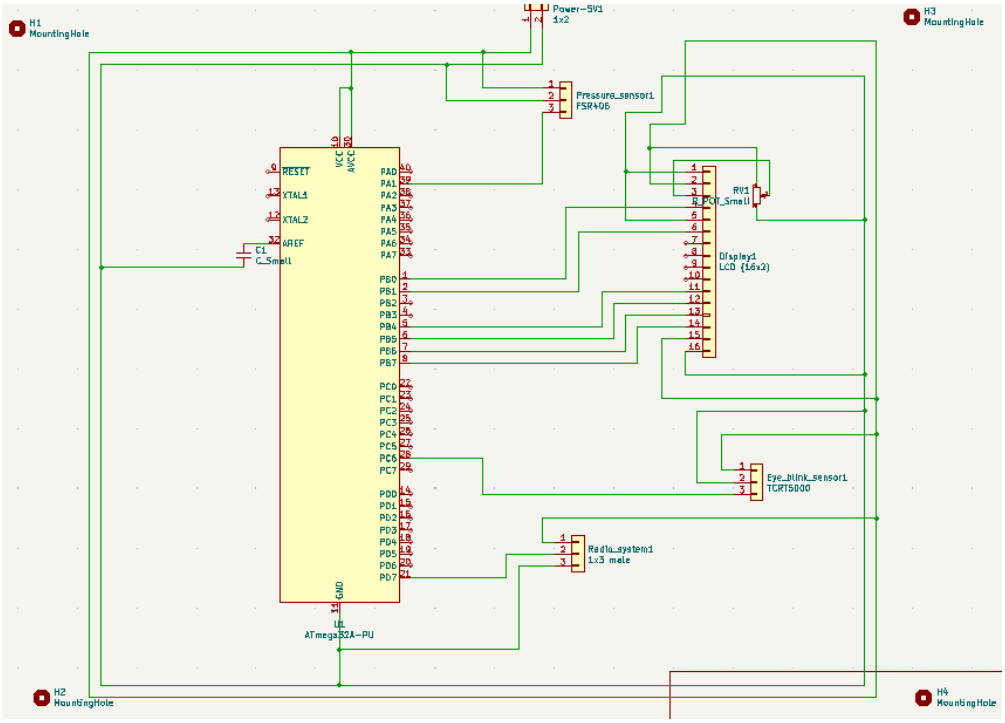

Figure 41 Whole Schematic Diagram 205048V

PCB Design – Whole Component

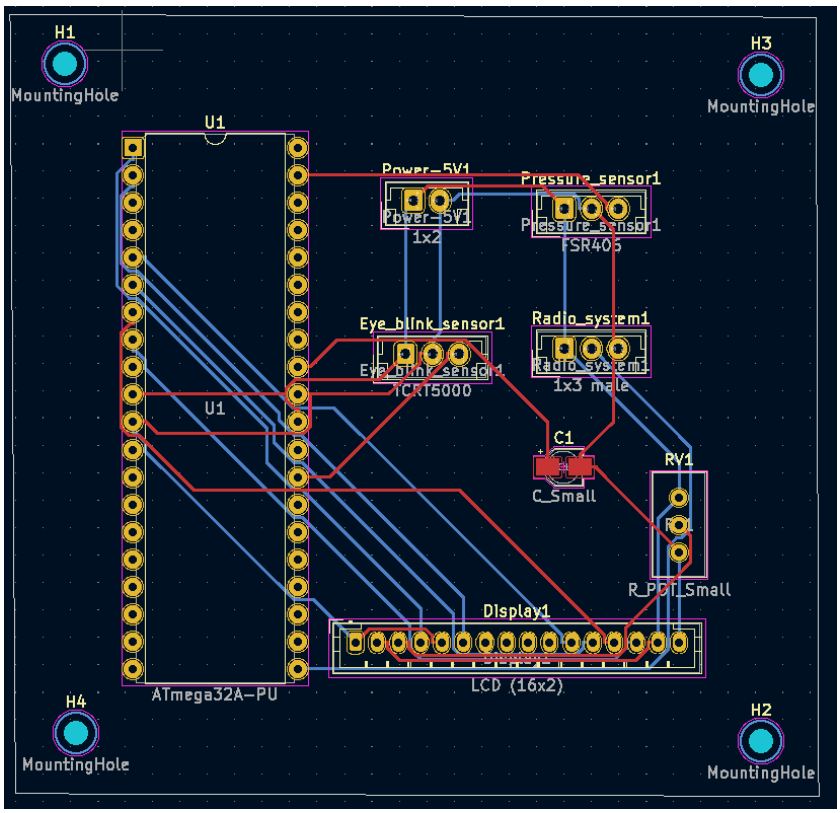

Figure 42 Whole PCB Design 205048V

Code – Whole Component

Main.c

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>
#include "LCD.h"

void ADC_Init()
{
	DDRA=0x00;
	ADCSRA = 0x87;
	ADMUX = 0xC0;
}

int ADC_Read(char channel)
{
	int Ain,AinLow;

	ADMUX=ADMUX|(channel & 0x0f);
	ADCSRA |= (1<<ADSC);
	while((ADCSRA&(1<<ADIF))==0);
	_delay_us(10);
	AinLow = (int)ADCL;
	Ain = (int)ADCH*256;
	Ain = Ain + AinLow;
	return(Ain);
}

int main()
{
	int value;
	ADC_Init();
	LCD_Init();
	LCD_String("vehicle:");
	while (1)
	{
		LCD_Command(0xc4);
		value = ADC_Read(1);
		if (value > 107)// when pressure above 15
		{LCD_String("moving....");
			_delay_ms(200);
		LCD_Clear();
		LCD_String("EYE STATUS :");
		LCD_Command(0xc0);
		if((PINC & 0X40)== 0X40)
		{
			LCD_String("Eyes closed");
			_delay_ms(2000);
			if((PINC & 0X40)== 0X40)
			{	LCD_Clear();
				LCD_String("SLEEPING !!!!");
				LCD_Command(0xc0);
				LCD_String("WAKE UP  !!!!");
				while(1){
					PORTD = 0x80;
					_delay_ms(100);
					PORTD = 0x00;
					_delay_ms(100);
				}
			}
		}
		else
		{
```

```
                                        LCD_String("Eyes opened");
                                        _delay_ms(200);
                                        LCD_Clear();
                                }
                                }
                                else
                                {LCD_String("not moving");
                                _delay_ms(150);}
                }
                return 0;
}
```

LCD.h

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

#define LCD_Dir DDRB
#define LCD_Port PORTB
#define RS PB0
#define EN PB1
void LCD_Command( unsigned char cmnd );
void LCD_Char( unsigned char data );
void LCD_Init (void)     ;
void LCD_String (char *str)             ;
void LCD_String_xy (char row, char pos, char *str);
void LCD_Clear();

void LCD_Command( unsigned char cmnd )
{
                LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0);
                LCD_Port &= ~ (1<<RS);
                LCD_Port |= (1<<EN);
                _delay_us(1);
                LCD_Port &= ~ (1<<EN);

                _delay_us(200);

                LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4);
                LCD_Port |= (1<<EN);
                _delay_us(1);
                LCD_Port &= ~ (1<<EN);
                _delay_ms(2);
}


void LCD_Char( unsigned char data )
{
                LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0);
                LCD_Port |= (1<<RS);
                LCD_Port|= (1<<EN);
                _delay_us(1);
                LCD_Port &= ~ (1<<EN);

                _delay_us(200);

                LCD_Port = (LCD_Port & 0x0F) | (data << 4);
                LCD_Port |= (1<<EN);
                _delay_us(1);
                LCD_Port &= ~ (1<<EN);
                _delay_ms(2);
}

void LCD_Init (void)
{
```

```
		LCD_Dir = 0xFF;
		_delay_ms(20);
		LCD_Command(0x33);
		LCD_Command(0x32);
		LCD_Command(0x28);
		LCD_Command(0x0c);
		LCD_Command(0x06);
		LCD_Command(0x01);
		_delay_ms(2);
		LCD_Command (0x80);
}


void LCD_String (char *str)
{
		int i;
		for(i=0;str[i]!=0;i++)
		{
			LCD_Char (str[i]);
		}
}

void LCD_String_xy (char row, char pos, char *str)
{
		if (row == 0 && pos<16)
		LCD_Command((pos & 0x0F)|0x80);
		else if (row == 1 && pos<16)
		LCD_Command((pos & 0x0F)|0xC0);
		LCD_String(str);
}

void LCD_Clear()
{
		LCD_Command (0x01);
		_delay_ms(2);
		LCD_Command (0x80);
}
```

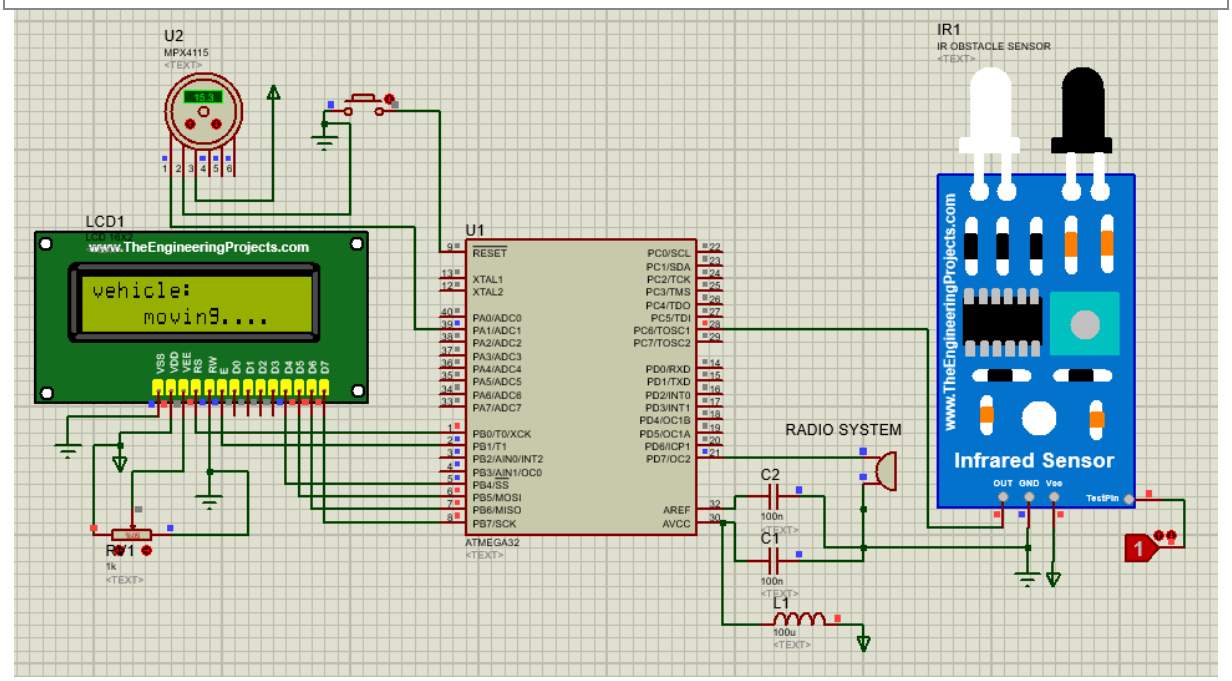

Figure 43 Whole Proteus Simulation 205048V