A PROJECT REPORT

ON

SMART STEERING FOR IMPROVING DRIVER'S SAFETY

Submitted By

Vikash Kumar Thakur

Vidya Sagar Jha

Under the guidance of

Dr. S. Pravinth Raja

ABSTRACT

All over the world, most of the road accidents are occurred due to drunk and drive, rash-driving, over-speeding, poor health condition of driver and drowsiness. The main concept of this product is to prevent road accidents. So, this product consists of some sensors like alcohol detection sensor, speed sensor, pulse sensor, temperature sensor, GPS sensor to detect the driver physical condition and using some algorithms like Viola-Jones algorithm and some formulas and frameworks like Open-CV, Dlib to detect the driver's mental condition. The alcohol sensor will detect if driver is drunken or not. The speed sensor will detect the speed of vehicle. The pulse and temperature sensor will detect the Heart- beat rate and body temperature of driver respectively. Based, on the different readings taken by sensors the system can take decision in the favour of driver. Also, this product detects driver drowsiness based on visual input (driver's face and head). It combines off-the-shelf software components for face detection, human skin color detection, and eye state (open vs closed). In this process, the vehicle and driver condition (physical and mental) can be monitored using an Android application i.e. how much the driver have drunken, speed of the car, location of the vehicle, driver's health and driver's drowsiness.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1. General

Each year, 1.35 million people are killed on roadways around the world. Every day, almost 3,700 people are killed globally in road traffic crashes involving cars, buses, motorcycles, bicycles, trucks, or pedestrians. More than half of those killed are pedestrians, motorcyclists, and cyclists. The most common cause of accidents is drowsiness, drunk driving, breaking the speed limit, Driver health issue and rash driving. A 2014 AAA Traffic Safety Foundation study found that 37 percent of driver's report having fallen asleep behind the wheel at some point in their lives. An estimated 21 percent of fatal crashes, 13 percent of crashes resulting in severe injury and 6 percent of all crashes, involve a drowsy driver. Apr 6, 2020 - 2019 drunk driving statistics 20.1% of responders aged 35 to 44 know someone who has been killed in a drunk-driving accident. 26.6% of female respondents aged 18 to 24 admitted to driving while buzzed. 37.4% of college-age respondents believed they were fit to drive after 3 or 4 drinks. As many as 97,588 people died in 2018 due to accidents caused by over speeding. If drowsiness, drunk driving, over speeding, Drivers health condition and rash driving status can be accurately detected, incidents can be prevented by countermeasures, such as the arousing of driver and deactivation of cruise control. To detect the drowsiness, speed of the vehicle, driver's health, alcohol consumed by the driver, and to measure rash driving status the model is installed with some sensors and a movable camera. The sensors will detect the physical condition of the driver and the camera module will take the live recording of the driver's face part to detect the drowsiness. Also, we developed an android application to monitor the driver status. Here the data received by sensors and camera module will be the input for the microcontrollers and based on the input the microcontroller will give the proper decision either to alert the driver or not. Also, whatever the data the sensors and camera module will take the live data will go to the firebase and from that firebase to the android application. Here the user can see the live status of the driver using the application. Also, the user can send the message to the driver either by text message or by calling the driver. To find the drowsiness we are considering the two readings. The first one is driver's eye

closing time and second one is the yawning rate of the driver.

1.2. Problem Domain

REQUIREMENTS	DESCRIPTION	HARDWARE/SOFTWARE
DLIB	General purpose cross platform	SOFTWARE
CV2	General purpose cross platform	SOFTWARE
CAMERA	Real time video streaming	HARDWARE
ANDROID-STUDIO	Android application	SOFTWARE
ARDUINO	For sensors reading	HARDWARE
RASPBERRY PI	For drowsiness	HARDWARE
STEERING WHEEL	Real steering wheel to connect every hardware's.	HARDWARE

Table 1.2

1.2.1 Drowsiness

Sleep cycle is divided into nonrapid-eye-movement (NREM) sleep and rapid-eye-movement (REM) sleep, and the NREM sleep is further divided into stages 1-4. Drowsiness is stage 1 of NREM sleep – the first stage of sleep. A number of efforts have been reported in the literature on the developing of drowsiness detection systems for drivers. These drowsiness detection methods can be categorized into two major approaches but in our project, we are going with first approach. The two major approaches are:

1.2.1.1 Imaging processing techniques:

This approach analyses the images captured by cameras to detect physical changes of drivers, such as eyelid movement, eye gaze, yawn, and head nodding. For example, the PERCLOS system developed by W. W. Wierwile et. al. used camera and imaging processing techniques

to measure the percentage of eyelid closure over the pupil over time [8-10]. The three-in-one vehicle operator sensor developed by Northrop Grumman Co. also used the similar techniques. Although this vision-based method is not intrusive and will not cause annoyance to drivers, the drowsiness detection is not so accurate, which is severely affected by the environmental backgrounds, driving conditions, and driver activities (such as turning around, talking, and picking up beverage). In addition, this approach requires the camera to focus on a relatively small area (around the driver's eyes). It thus requires relative precise camera focus adjustment for every driver.

1.2.1.2 Physiological signal detection techniques:

This approach is to measure the physiological changes of drivers from bio signals, such as the electroencephalogram (EEG), electrooculography (EOG), and electrocardiogram (ECG or EKG). Since the sleep rhythm is strongly correlated with brain and heart activities, these physiological bio signals can give accurate drowsiness/sleepiness detection. However, all the researches up to date in this approach need electrode contacts on drivers' head, face, or chest. Wiring is another problem for this approach. The electrode contacts and wires will annoy the drivers, and are difficult to be implemented in real applications.

1.2.2 Over Speeding

The proposed system comprises of speed sensor LM393.

The **LM393** IC is a low-power, single supply, low-offset voltage, double, differential comparators. Generally, a common comparator IC is a tiny voltmeter by included switches. It is used to calculate the voltages at two dissimilar terminals and contrasts the dissimilarity in voltage quantity. When the vehicle reaches the speed limit and start to cross the limit then buzzer will inform the driver to stop the vehicle otherwise vehicle will stop in some duration.

1.2.3 Drunk and Drive

The alcohol sensor is technically referred to as a MQ3 sensor which detects ethanol in the air. When a drunk person breathes near the alcohol sensor it detects the ethanol in his breathe and provides an output based on alcohol concentration. If there is more alcohol concentration then the vehicle will not start.

1.2.4 Health Issue

Driver health is the most important factor to prevent road accident. If the driver having any health issue it can cause accident. In our proposed method we are using two sensors to monitor driver health. The sensors are:

1.2.4.1 Temperature Sensor

This sensor will be embedded in the steering wheel. DHT-11 will be used to sense the driver body temperature and if the body temperature is more than normal then the buzzer will warn the driver for medication otherwise vehicle will stop in some moment.

1.2.4.2 Pulse Sensor

This sensor will also be embedded in the steering wheel. XD58-C will be used to sense the drivers pulse rate and if the pulse rate is more than normal then the buzzer will warn the driver for medication otherwise vehicle will stop after some time duration.

1.2.5 Rash Driving

In the proposed system we are not using any sensor to detect the rash driving. We are using an algorithm to detect the rash driving.
The proposed system checks rash driving by calculating the speed of a vehicle using the time taken to travel certain distance.

1.3 Related Studies

This study in computer vision for monitoring driver vigilance uses active infrared illumination and a single camera placed on the car dashboard. It has employed this technique because the goal is to monitor a driver on real conditions.

Facial landmarks refer to characteristic points on a face like the corners of the mouth, the corners of the eyes or the tip of the nose. Detection of facial landmarks in images of faces is an important step in most face image interpretation tasks. Most existing facial landmark

detectors simultaneously model local appearance around the landmarks and their geometrical configuration. The local appearance is described either by generative models or by discriminatively trained detectors. The geometrical structure of the landmarks is usually modeled by a Point Distribution Model, describing the landmark positions of a face in canonical pose, and by a subsequent transformation from the canonical pose into 2D image coordinates.

The other studies are in the field of sensors which will detect the different conditions of drivers as well as vehicle. For example, the LM393 which is used for sensing the vehicle speed, when the vehicle crosses the speed limit the current speed of the vehicle will be stored in the android application and will give a voice message to alert the driver. Anyone who is monitoring the vehicle can call the driver and give him/her to slow the car.

Chapter – 2

REQUIREMENT ANALYSIS

2.1. HARDWARE REQUIREMENTS

1. **Arduino UNO SMD**: It is open source microcontroller. The board is equipped with sets of digital and Analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board features 14 Digital pins and 6 Analog pins.
2. **HDMI Cable**: HDMI cable is used to connect. That is used to view the GUI of Raspberry Pi.
3. **Adapter**: Adapter with Input 100-240VAC 50-60Hz and Output 12V- 2Amp is connected to GSM module.
4. **GSM module**: GSM module is used for text messaging. It is used to read the status of the driver's blinking eyes.
5. **Android Mobile**: In this project, android mobile is used for real time monitoring of drowsiness.
6. **Desktop/Laptop/Tablet**: It is used for coding the program.
7. **IR[Infra-Red] Transmitter & Receiver LED**: It is fitted across the sides the specs. It is used to read the eye blink status.
8. **Buzzer**: Buzzer will be blown at driver side when eye lid is closed for more than 1 second.
9. **Speed sensor:** It is used for measure the vehicle speed.
10. **Alcohol sensor:** It is used to measure alcohol consumption by the driver before driving.
11. **Pulse sensor:** It is used to measure the heart beat rate of driver.
12. **Temperature sensor:** This sensor will be used to measure the driver present body temperature.
13. **Steering wheel:** It is used to combine all sensors in one body i.e. steering wheel itself.
14. **Wi-Fi Module:** It is used to connect the Arduino with internet.
15. **Camera module:** Camera module is used for live video streaming of driver.
16. **Servo Motor:** We are using motor to show ignition of the vehicle.

17. **LED bulb:** We are using LED to show ignition of the vehicle.
18. **LCD 16 pin display:** It is used to display output to the driver.

2.2 SOFTWARE REQUIREMENTS

1. **Android programming software:** Android application is developed through android programming software to check the status of driver's drowsiness condition in real time.
2. **Cross Compiler Arduino 1.5.5v:** It is an open source computer hardware and software platform for building digital devices and interactive objects that can sense and control the physical world around them.
3. **Compiler android studio:** It is the official IDE for android development and includes everything you need to build android app.
4. **Python 3:** It is used to write code.
5. **NumPy (python framework):** used to create matrix for image processing.
6. **Pandas (python framework):** Python framework for data analysis.
7. **OpenCV (python framework): OpenCV** (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images.
8. **DLIB (python framework):** Framework used for face detection.

CHAPTER- 3

LITERATURE REVIEW

3.1 RASH DRIVING

Various researchers have tried to monitor driver behaviors using both dedicated sensors deployed inside car, roadside and smartphone inbuilt sensors. In paper [1], P. Singh et al. developed an android based application. This application collects data from accelerometers, GPS and also record sounds with the help of microphone, and then data is combined and analyzed to detect rash driving patterns. The various patterns such as speed breaker, lane-change left/right, left/right turn, sudden breaking and sudden acceleration were analyzed.

3.2. DROWSINESS

In 2008, Hong Su et. al. [15] described 'A Partial Least Squares Regression-Based Fusion Model for Predicting the Trend in Drowsiness'. They proposed a new technique of modelling driver drowsiness with multiple eyelid movement features based on an information fusion technique—partial least squares regression (PLSR), with which to cope with the problem of strong collinear relations among eyelid movement features and, thus, predicting the tendency of the drowsiness. The predictive precision and robustness of the model thus established are validated, which show that it provides a novel way of fusing multi-features together for enhancing our capability of detecting and predicting the state of drowsiness.

3.3 DRIVER'S HEALTH MONITORING

A team of researchers at Washington state university developed smart steering- wheel which employed Sensio Foil sensing technology. This steering wheel intended to detect grip lessens and hand movements. The team claimed that this technology overcomes many limitations currently available in video-based driver drowsiness detection. Ford's European research and innovation center initial worked on a car seat which could detect heart attacks by embedded sensors, later, they have stopped to continue their research.

For the automated driver-assistance system, the Detroit carmaker testing eye-tracking features. Also, GM proposed eye gaze monitoring technology to it super cruise cars. Furthermore, German automakers like Audi and Volkswagen expected to launch eye-tracking systems in quicker time. Most recently, IPPOCRATE [20] projected a steering wheel monitoring system. This system could monitor body temperature, cardio activity, blood pressure, pulse rate, and eye movement. Similarly, BMW and Technishche Universities Muenchen (TUM) developed a smart steering wheel that monitors driver's vital signs Toyota also developed a smart steering wheel to monitor the

driver's ECG.

CHAPTER -4

EXISTING SYSTEM

4.1 DROWSINESS DETECTION

By using a non-intrusive machine vision-based concepts, drowsiness of the driver detected system is developed. Many existing systems require a camera which is installed in front of driver. It points straight towards the face of the driver and monitors the driver's eyes in order to identify the drowsiness. For large vehicle such as heavy trucks and buses this arrangement is not pertinent. Bus has a large front glass window to have a broad view for safe driving. If we place a camera on the window of front glass0, the camera blocks the frontal view of driver so it is not practical. If the camera is placed on the frame which is just about the window, then the camera is unable to detain the anterior view of the face of the driver correctly. The open CV detector detects only 40% of face of driver in normal driving position in video recording of 10 minutes. In the oblique view, the Open CV eye detector (CV-ED) frequently fails to trace the pair of eyes. If the eyes are closed for five successive frames the system concludes that the driver is declining slumbering and issues a warning signal. Hence existing system is not applicable for large vehicles. In order to conquer the problem of existing system, new detection system is developed in this project work.

4.2 RASH DRIVING

The existing system checks an over speeding vehicle or rash driving by calculating the speed of the passing vehicle using the time taken to travel between two check points (at a fixed distance) installed on either side of the road at a fixed distance. Our proposed system is doing it using speed sensor installed in vehicle.

4.3 ALCOHOL DETECTION

In the existing system, alcohol detectors are not proposed in any of the vehicles, hence there is a chance for anyone to drink and drive. Traffic police uses alcohol detectors to avoid drunk and drive system. Our system having inbuilt steering alcohol sensor. Before driving the driver have to blow on the steering and if the alcohol consumption is zero then only the ignition of the vehicle will start.

4.4 HEALTH MONITORING

There is no existing system for live checking of driver health. Our proposed system consists of two health checking sensors that is pulse sensor and temperature sensor.

4.5 COMPARISON TABLE

Sl. No.	Criteria	Existing System	Our System
1.	Drowsiness	Fixed Camera	Movable Camera
2.	Drowsiness	10 min recording	Continuous Recording
3.	Drowsiness measure	Reading Blink Rate	Eye Closing Time
4.	Drowsiness measure	No Yawn Detection	Yawn Detection
5.	Rash Driving	Speed/Time	Speed Sensor
6.	Alcohol Detection	No Attached Mechanism	Attached Sensor
7.	Health Monitoring	No Mechanism in Vehicle	Live Health Monitoring
8.	Location Tracker	Old Hardware of Bigger Size	Latest GPS Sensor of Small Size
9.	Android Application	No Such Application Present	Application Can Monitor Everything
10.	Steering	Normal Steering	Smart Steering

Table 4.5

CHAPTER -5

PROPOSED WORK

Our work is divided into few parts like drowsiness detection, alcohol consumption detection, over-speeding detection etc. We are covering all working methodologies part by part.

5.1 DROWSINESS

Feeling abnormally sleepy or tired during the day is commonly known as drowsiness. Drowsiness may lead to additional symptoms, such as forgetfulness or falling asleep at inappropriate times. Drowsy driving is the dangerous combination of driving and sleepiness or fatigue. This usually happens when a driver has not slept enough, but it can also happen because of untreated sleep disorders, medications, drinking alcohol, or shift work. The National Highway Traffic Safety Administration estimates that drowsy driving was responsible for 72,000 crashes, 44,000 injuries, and 800 deaths in 2013. However, these numbers are underestimated, and up to 6,000 fatal crashes each year may be caused by drowsy drivers.

A part of our project is for detecting driver drowsiness. The camera model what we are using in our proposed system will take the live eye blinking rate of the driver and if the blinking rate is more than normal then due to some algorithms we are using and some python library like OpenCV and a DLIB.

To build our blink detector, we'll be computing a metric called the eye aspect ratio (EAR), introduced by Soukup ova and Cech in their 2016 paper, Real-Time Eye Blink Detection Using Facial Landmarks.

Unlike traditional image processing methods for computing blinks which typically involve some combination of:

1. Eye localization.

2. Thresholding to find the whites of the eyes.

3. Determining if the "white" region of the eyes disappears for a period of time (indicating a blink).

The eye aspect ratio is instead a much more elegant solution that involves a very simple calculation based on the ratio of distances between facial landmarks of the eyes. This method for eye blink detection is fast, efficient, and easy to implement. In terms of blink detection, we are only interested in two sets of facial structures -the eyes. Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region:

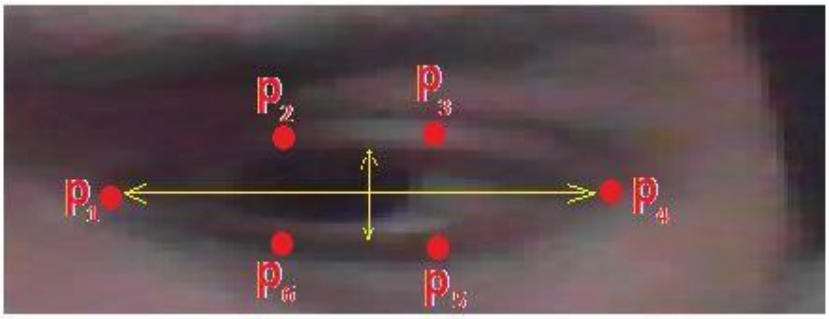


Figure 5.1

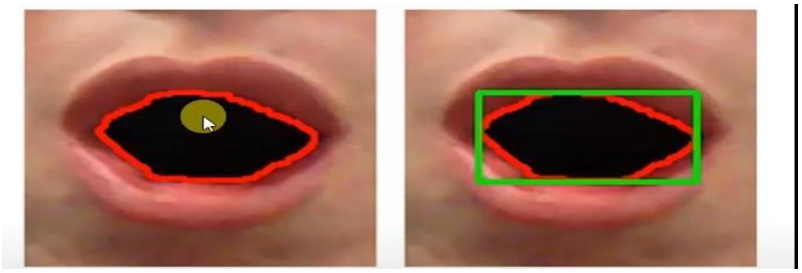
The 6 facial landmarks associated with the eye.

Based on this image, we should take away on key point: There is a relation between the width and the height of these coordinates. Based on the work by Soukup ova and Cech in their 2016 paper, Real-Time Eye Blink Detection using Facial Landmarks, we can then derive an equation that reflects this relation called the eye aspect ratio (EAR):

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Equation 5.1

Where p1, …, p6 are 2D facial landmark locations.



Detecting Yawn using facial landmarks.

5.2 CAMERA MODULE

Figure 5.2
CAMERA MODULE

The camera board attaches to the Raspberry Pi via a 15-way ribbon cable. There are only two connections to make: the ribbon cable needs to be attached to the camera PCB, and to the Raspberry Pi itself. You need to get the cable the right way round, or the camera will not work. On the camera PCB, the blue backing on the cable should face away from the PCB, and on the Raspberry Pi it should face towards the Ethernet connection (or where the Ethernet connector would be if you're using a model A).
The camera we are using in our module will take the live eye blinking rate of driver and if the eye blinking rate is not normal then our algorithm work here.
When determining if a blink is taking place in a video stream, we need to calculate the eye aspect ratio. If the eye aspect ratio falls below a certain threshold and then rises above the threshold, then we'll register a "blink" the EYE_AR_THRESH is this threshold value. We default it to a value of 0.3 as this is what has worked best for my applications, but you may need to tune it for your own application.

5.3 ALCOHOL SENSOR

Figure 5.3
Alcohol Sensor

Our proposed work consists of various units that make up the system: one of them is the alcohol detection unit. The working of this unit is based on the alcohol sensor and its output. The alcohol sensor is technically referred to as a MQ3 sensor which detects ethanol in the air. When a drunk person breathes near the alcohol sensor it detects the ethanol in his

breathe and provides an output based on alcohol concentration. If there is more alcohol concentration more LED's would lit.
If the driver is drunk then the sensor will read the real alcohol consumption then the car ignition will not start and the data will go to the database which is connected to our android application. Anyone who is having steering Id and password can monitor the driver's data as well as location.

5.4 SPEED SENSOR

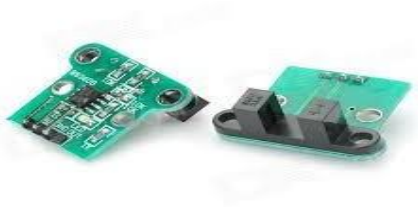


Figure 5.4
Speed Sensor

Our proposed work consists of various units that make up the system: one of them is the speed detection unit. We are doing it by using speed sensor. A speed sensor is a type of tachometer that is used to measure the speed of a rotating object like a motor. In order to measure speed of a motor using Arduino, I have used an LM393 Speed sensor with Arduino.

5.4.1 Circuit diagram (speed sensor)

First, connect the VCC and GND of the LM393 Sensor to +5V and GND of Arduino. The OUT or SIG pin of the sensor is connected to Pin 11 of Arduino. Coming to the LCD, its RS and E pins are connected to Pins 7 and 6 of Arduino. The data pins D4 – D7 are connected to Pins 5, 4, 3 and 2 of Arduino.

Here also when the driver over speeding the vehicle then the current speed data will get read by speed sensor and get saved in the database and the person who is monitoring can send a message to the driver to slow down the car. If nobody is monitoring then there will be inbuilt buzzer in the vehicle for warning. The buzzer will give warning to the driver to slow down the vehicle.

5.5 PULSE SENSOR

Figure 5.5.1

Pulse Sensor is a well-designed plug-and-play heart-rate sensor for Arduino. The working of this sensor can be done by connecting it from the fingertip or human ear to Arduino board. So that heart rate can be easily calculated. The pulse sensor includes a 24 inches color code cable, ear clip, Velcro Dots-2, transparent stickers-3, etc.

- A color code cable is connected to header connectors. So, this sensor is easily connected to an Arduino into the project without soldering.
- An ear clip size is the same as a heart rate sensor and it can be connected using hot glue at the backside of the sensor to wear on the earlobe.
- Two Velcro dots are completely sized toward the sensor at the hook side. These are extremely useful while making a Velcro strap to cover approximately a fingertip. This is used to cover the Sensor around the finger.
- Transparent strikers are protection layers used to protect the sensor from sweaty earlobes and fingers. This sensor includes three holes in the region of the external edge so that one can easily connect anything to it.

5.5.1 Pin configuration

The heartbeat sensor includes three pins which discussed below.

Figure 5.5.2
Pulse-sensor-pin-configuration

- Pin-1 (GND): Black Color Wire – It is connected to the GND terminal of the system.
- Pin-2 (VCC): Red Color Wire – It is connected to the supply voltage (+5V otherwise +3.3V) of the system.
- Pin-3 (Signal): Purple Color Wire – It is connected to the pulsating o/p signal.

Here this pulse sensor will read the heart beat rate of the driver who is driving the vehicle and if the heart rate is more than normal then the buzzer will alert the driver to take medication. Here this sensor is taking input and sending it to database and matching it to the normal heart beat rate value.

5.6 TEMPERATURE SENSOR

Figure 5.6
Temperature Sensor

DHT11 is a low-cost digital sensor for sensing temperature and humidity. This sensor can be easily interfaced with any micro-controller such as Arduino, Raspberry Pi etc., to measure humidity and temperature instantaneously.

DHT11 humidity and temperature sensor is available as a sensor and as a module. The difference between this sensor and module is the pull-up resistor and a power-on LED. DHT11 is a relative humidity sensor. To measure the surrounding air this sensor uses a thermistor and a capacitive humidity sensor. DHT11 sensor consists of a capacitive humidity sensing element and a thermistor for sensing temperature. The humidity sensing capacitor has two electrodes with a moisture holding substrate as a dielectric between them. Change in the capacitance value occurs with the change in humidity levels. The IC measure, process this changed resistance values and change them into digital form.

For measuring temperature this sensor uses a Negative Temperature coefficient thermistor, which causes a decrease in its resistance value with increase in temperature. To get larger resistance value even for the smallest change in temperature, this sensor is usually made up of semiconductor ceramics or polymers. The temperature range of DHT11 is from 0 to 50 degree Celsius with a 2-degree accuracy. Humidity range of this sensor is from 20 to 80% with 5% accuracy. The sampling rate of this sensor is 1Hz.i.e. it gives one reading for every second. DHT11 is small in size with operating voltage from 3 to 5 volts. The maximum current used while measuring is 2.5mA.

5.6.1 Pin Configuration

DHT11 sensor has four pins- VCC, GND, Data Pin and a not connected pin. A pull-up resistor of 5k to 10k ohms is provided for communication between sensor and micro-controller.

In the proposed model the sensor is taking the live driver body temperature to check the driver is physically good or not. If the reading taken by sensor is more than 98.6°F (37°C) then the driver will get the warning in the form of voice message to take proper medication and then drive. Also, who is monitoring the driver can alert the driver in the form of text message.

5.7 RASH DRIVING

In the proposed system we are not using any sensor to detect the rash driving. We are using an algorithm to detect the rash driving. The proposed system checks rash driving by calculating the speed of a vehicle using the time taken to travel certain distance.

5.8 ARDUINO UNO

Figure 5.8
Arduino Uno

Arduino is a single-board microcontroller meant to make the application more accessible which are interactive objects and its surroundings. The hardware features with an open-source hardware board designed around an 8-bit Atmel AVR microcontroller or a 32-bit Atmel ARM. Current models consists a USB interface, 6 analog input pins and 14 digital I/O pins that allows the user to attach various extension boards.

The Arduino Uno board is a microcontroller based on the ATmega328. It has 14 digital input/output pins in which 6 can be used as PWM outputs, a 16 MHz ceramic resonator, an ICSP header, a USB connection, 6 analog inputs, a power jack and a reset button. This contains all the required support needed for microcontroller. In order to get started, they are simply connected to a computer with a USB cable or with a AC-to-DC adapter or battery. Arduino Uno Board varies from all other boards and they will not use the FTDI USB-to-serial driver chip in them. It is featured by the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

In our proposed system we are using Arduino uno to connect all our sensors. And by taking input from sensors and based on that inputs our system is performing operations for driver safety.

5.9 RASPBERRY PI

figure 5.9
Raspberry PI

An SD card inserted into the slot on the board acts as the hard drive for the Raspberry Pi. It is powered by USB and the video output can be hooked up to a traditional RCA TV set, a more modern monitor, or even a TV using the HDMI port. This gives you all of the basic abilities of a normal computer. It also has an extremely low power consumption of about 3 watts. To put this power consumption in perspective, you could run over 30 Raspberry Pi's in place of a standard light bulb.

We are using raspberry pi for detect the drowsiness of the driver. For doing this we are using camera model. The camera board attaches to the Raspberry Pi via a 15-way ribbon cable. There are only two connections to make: the ribbon cable needs to be attached to the camera PCB, and to the Raspberry Pi itself. You need to get the cable the right way round, or the camera will not work. On the camera PCB, the blue backing on the cable should face away from the PCB, and on the Raspberry Pi it should face towards the Ethernet connection (or where the Ethernet connector would be if you're using a model A).
Next, you need to power up the Raspberry Pi and log in to Raspbian. The camera module options were not available in early versions of Raspbian, so to start with, you need to make sure that Raspbian is completely up to date. In a terminal window, type:

$ sudo apt-get update
$ sudo apt-get upgrade

Now, camera module is taking the live eye blinking rate and comparing it with the minimum eye blinking rate of normal person (i.e. pre-set value). If it is not normal then the driver will get alert message that "not to sleep".

5.10 LCD MONITOR

The display I'm using is a 16×2 LCD display. It's called a 16×2 LCD because the part 16×2 means that the LCD has 2 lines, and can display 16 characters per line. Therefore, a 16×2 LCD screen can display up to 32 characters at once. It is possible to display more than 32 characters with scrolling though.

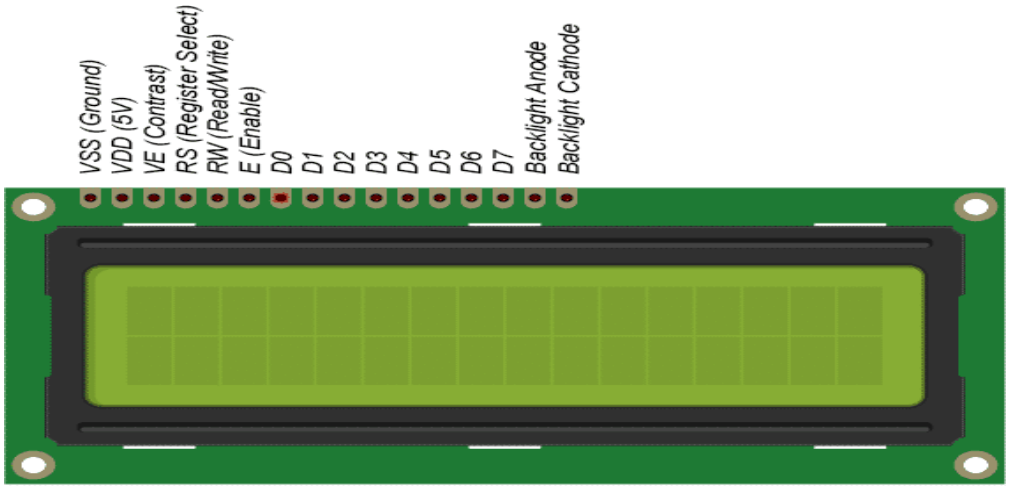


Figure 5.10
LCD Monitor

5.10.1 Pin configuration

To wire LCD screen to board, connect following pins.

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

We are using this Liquid-crystal-display to show the output to the driver.

5.11 ANDROID APPLICATION

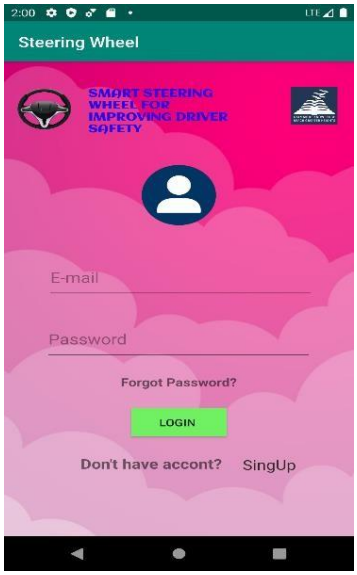


Figure 5.11

Android Application

An Android app is a software application running on the Android platform. Because the Android platform is built for mobile devices, a typical Android app is designed for a smartphone or a tablet PC running on the Android OS. The name of our android application is smart steering.

In our project we are also developed an android application through which user can monitor the vehicle as well as the driver. Our application is connected with firebase for database and firebase is getting input from Arduino uno and raspberry Pi. And our application is showing the output based on the data stored.

5.12 FIREBASE

Firebase is a mobile and web app development platform that provides developers with a plethora of tools and services to help them develop high-quality apps, grow their user base, and earn more profit.

Realtime syncing makes it easy for your users to access their data from any device, be it web or mobile. Realtime Database also helps your users collaborate with one another.

Another amazing benefit of Realtime Database is that it ships with mobile and web SDKs, allowing you to build your apps without the need for servers.

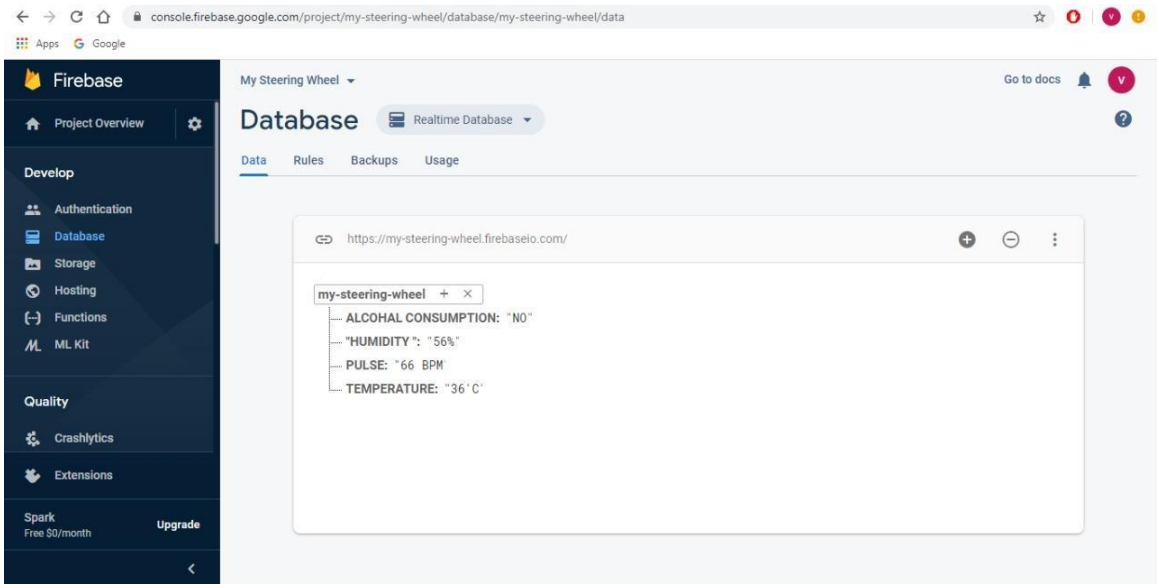


Figure 5.12

Firebase

5.13 GPS MODULE

Figure 5.13

GPS Module

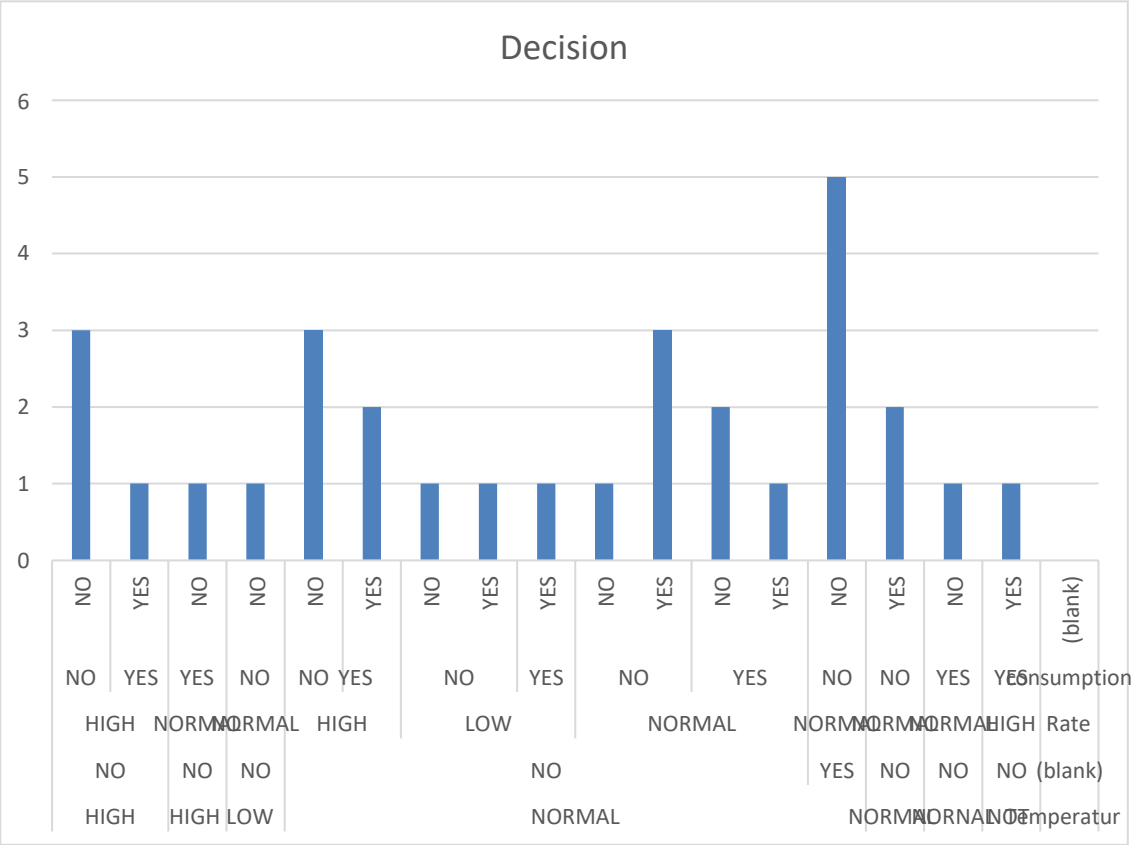
The heart of the module is a NEO-6M GPS chip from u-blox. It can track up to 22 satellites on 50 channels and achieves the industry's highest level of sensitivity i.e. -161 dB tracking, while consuming only 45mA supply current. The u-blox 6 positioning engine also boasts a Time-To-First-Fix (TTFF) of under 1 second. One of the best features the chip provides is Power Save Mode(PSM). It allows a reduction in system power consumption by selectively switching parts of the receiver ON and OFF. This dramatically reduces power consumption of the module to just 11mA making it suitable for power sensitive applications like GPS wristwatch. The necessary data pins of NEO-6M GPS chip are broken out to a "0.1″ pitch headers. This includes pins required for communication with a microcontroller over UART.

5.14 DATA FOR DECISION MAKING

Sl no	Name	Drowsiness	Alcohol consumption	Body Temperature	Pulse Rate	Decision
1	Vidya Sagar Jha	NO	NO	NORMAL	NORMAL	YES
2	Vikash Kumar	NO	NO	NORMAL	NORMAL	YES
3	Hanny Singh	YES	YES	NORMAL	HIGH	NO

4	Vicky	NO	NO	LOW	NORMAL	NO
5	Sarvesh	NO	NO	HIGH	HIGH	NO
6	Sunny Jha	YES	YES	NOT	HIGH	NO
7	Sumit Sahu	NO	YES	NORNAL	NORMAL	NO
8	Maha Prabhu	YES	YES	NORMAL	HIGH	NO
9	Abhay Jha	NO	YES	NORMAL	NORMAL	NO
10	Harsh	YES	YES	NORMAL	NORMAL	NO
11	Yashwant	NO	NO	HIGH	HIGH	NO
12	Aditya Madhav	NO	NO	NORMAL	LOW	NO
13	Vivek Sharma	NO	YES	NORMAL	NORMAL	NO
14	Ayush Tibriwal	YES	NO	NORMAL	NORMAL	NO
15	Shraddha Prasad	NO	NO	HIGH	HIGH	NO
16	Shashank	NO	NO	NORMAL	NORMAL	NO
17	Rupesh	YES	NO	NORMAL	NORMAL	NO
18	Raghavendra	NO	YES	HIGH	NORMAL	NO
19	Rohit Sala	YES	YES	NORMAL	LOW	NO
20	Ritis	YES	NO	NORMAL	NORMAL	NO
21	Ritesh	YES	NO	NORMAL	NORMAL	NO
22	Shivam	NO	NO	NORMAL	HIGH	NO
23	Arun	NO	NO	NORMAL	NORMAL	YES
24	Pushkar	YES	YES	HIGH	HIGH	NO
25	Sonu	YES	NO	NORMAL	NORMAL	NO
26	Raju	NO	NO	NORMAL	HIGH	NO
27	Shanjana	YES	NO	NORMAL	LOW	NO
28	Amit Mishra	NO	NO	NORMAL	HIGH	NO
29	Salman Khan	NO	NO	NORMAL	NORMAL	YES
30	Dr Pravinth Raja	NO	NO	NORMAL	NORMAL	YES

Table 5.14


Decision
6
5
4
3
2
1
0
NO
YES
NO
NO
NO
YES
NO
YES
YES
NO
YES
NO
YES
NO
YES
NO
YES
(blank)
NO
YES
YES
NO
NO
YES
NO
YES
NO
YES
NO
NO
YES
YES
Consumption
HIGH
NORMAL
NORMAL
HIGH
LOW
NORMAL
NORMAL
NORMAL
NORMAL
HIGH
Rate
NO
NO
NO
NO
YES
NO
NO
NO
(blank)
HIGH
HIGH
LOW
NORMAL
NORMAL
NORMAL
NORMAL
Temperatur

Graph 5.14

CHAPTER – 6

SYSTEM DESIGN

6.1 USE CASE DIAGRAM

This diagram shows the user's interaction with system. This diagram also shows the different types of users. Here In the figure (6.1) driver and owner is the user and they are interacting with Android Application and IOT System.

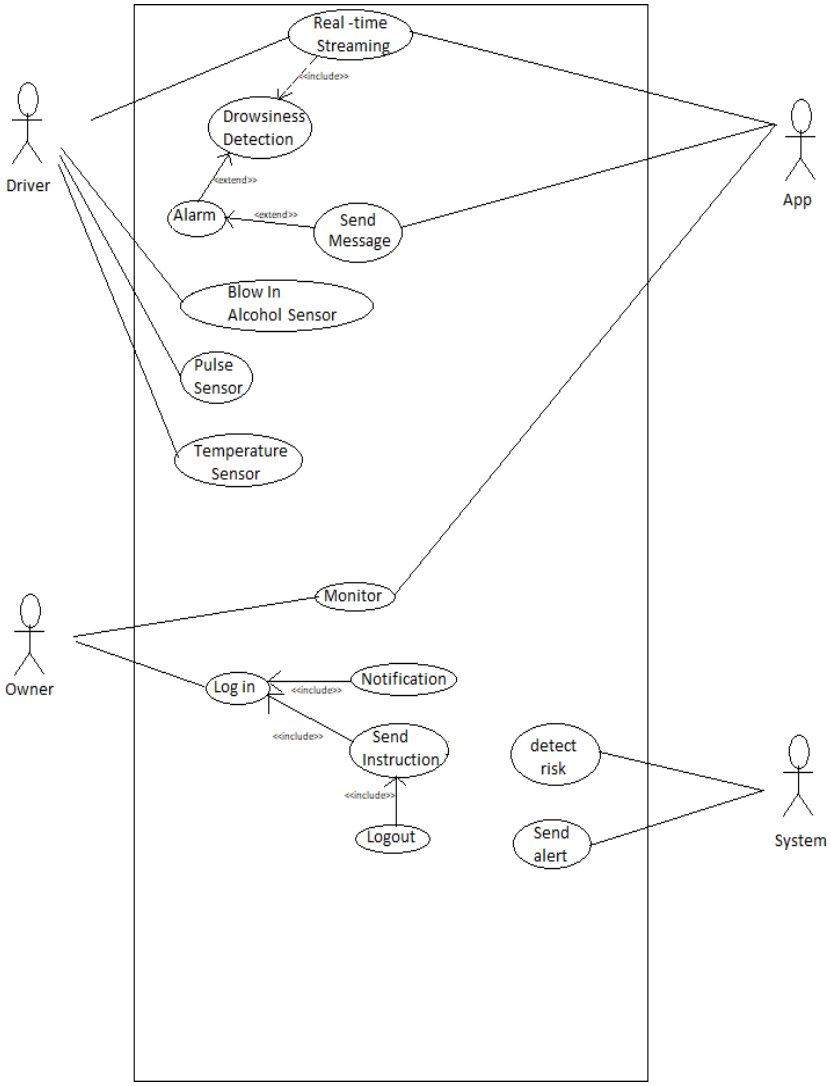


Figure 6.1
Use Case Diagram

6.2 SEQUENCE DIAGRAM

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged

between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

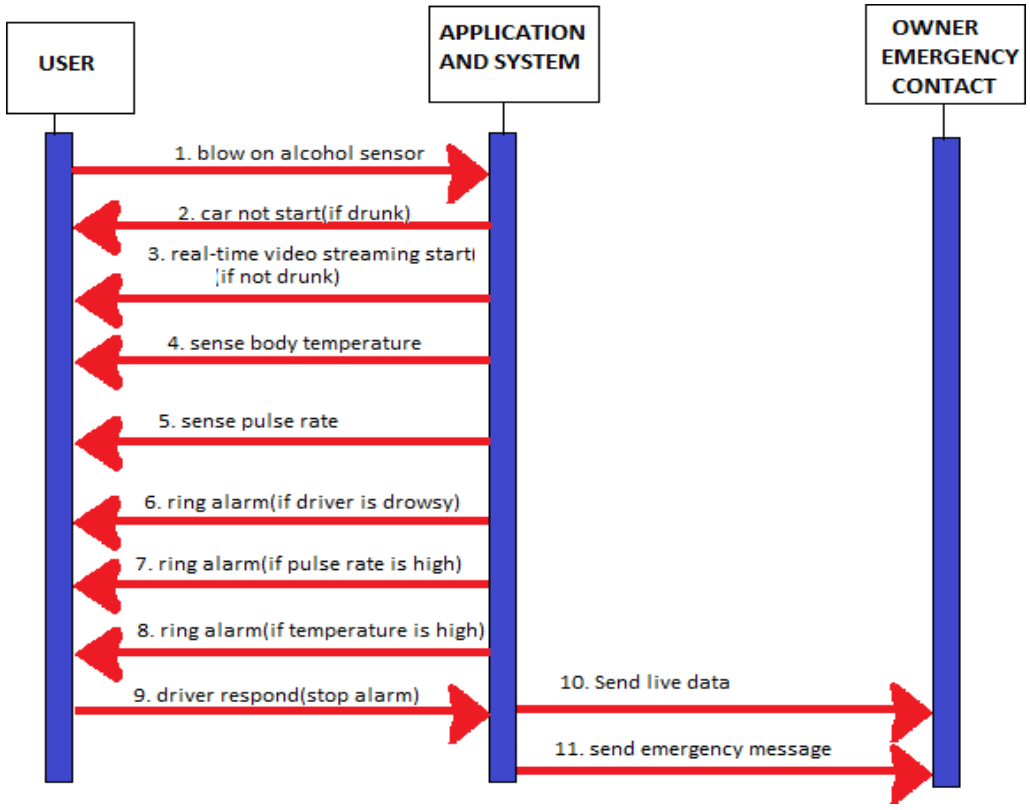


Figure 6.2
Sequence Diagram

6.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- *ellipses* represent *actions*;
- *diamonds* represent *decisions*;
- *bars* represent the start (*split*) or end (*join*) of concurrent activities;
- a *black circle* represents the start (*initial node*) of the workflow;
- an *encircled black circle* represents the end (*final node*).

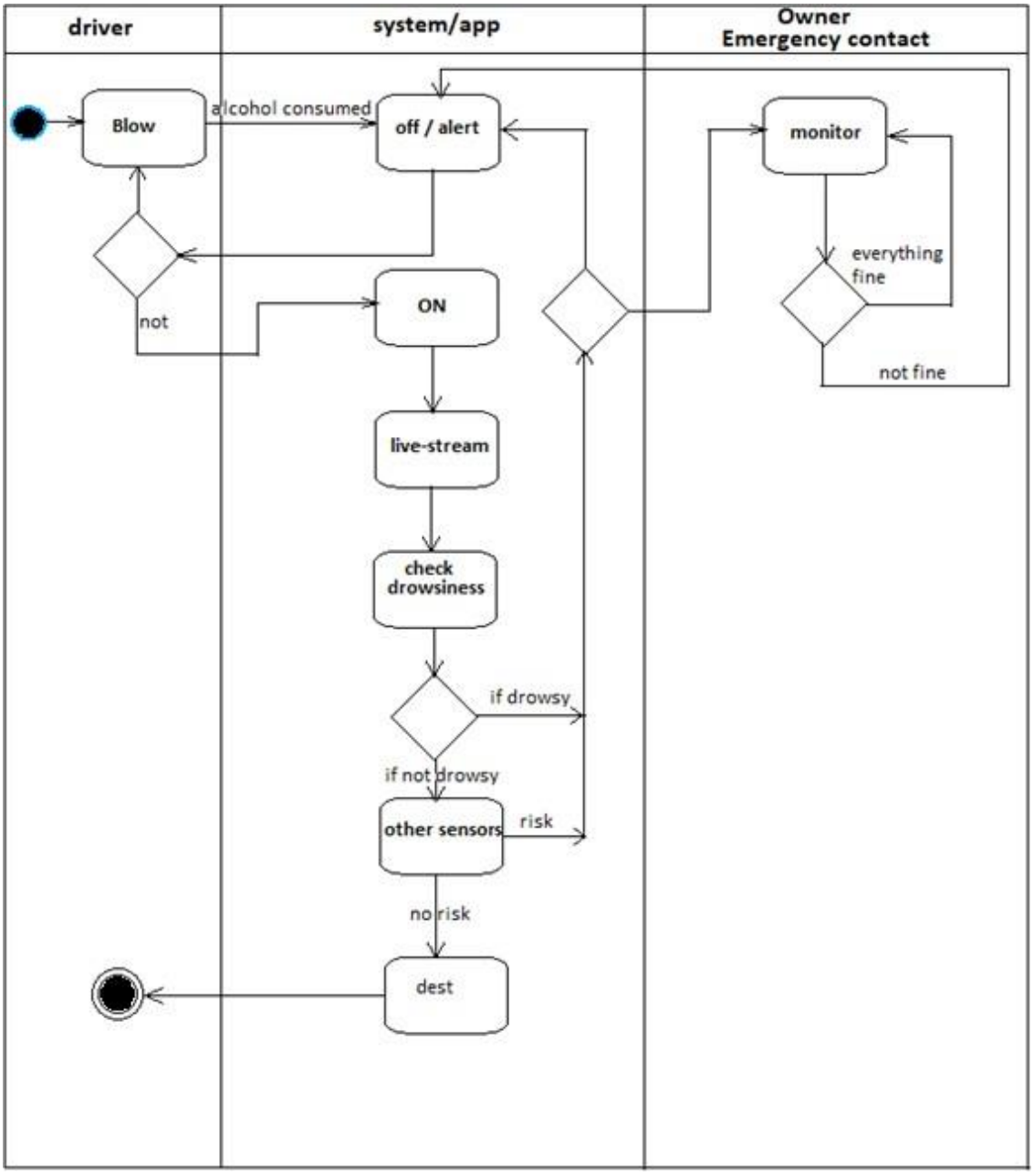


Figure 6.3
Activity Diagram

CHAPTER -7

IMPLIMENTATIONS

In this section the report is covering the various phases the project vision and plans. In this section we are covering the Architecture, overall duration of the project, infrastructure, training implementation, training data etc.

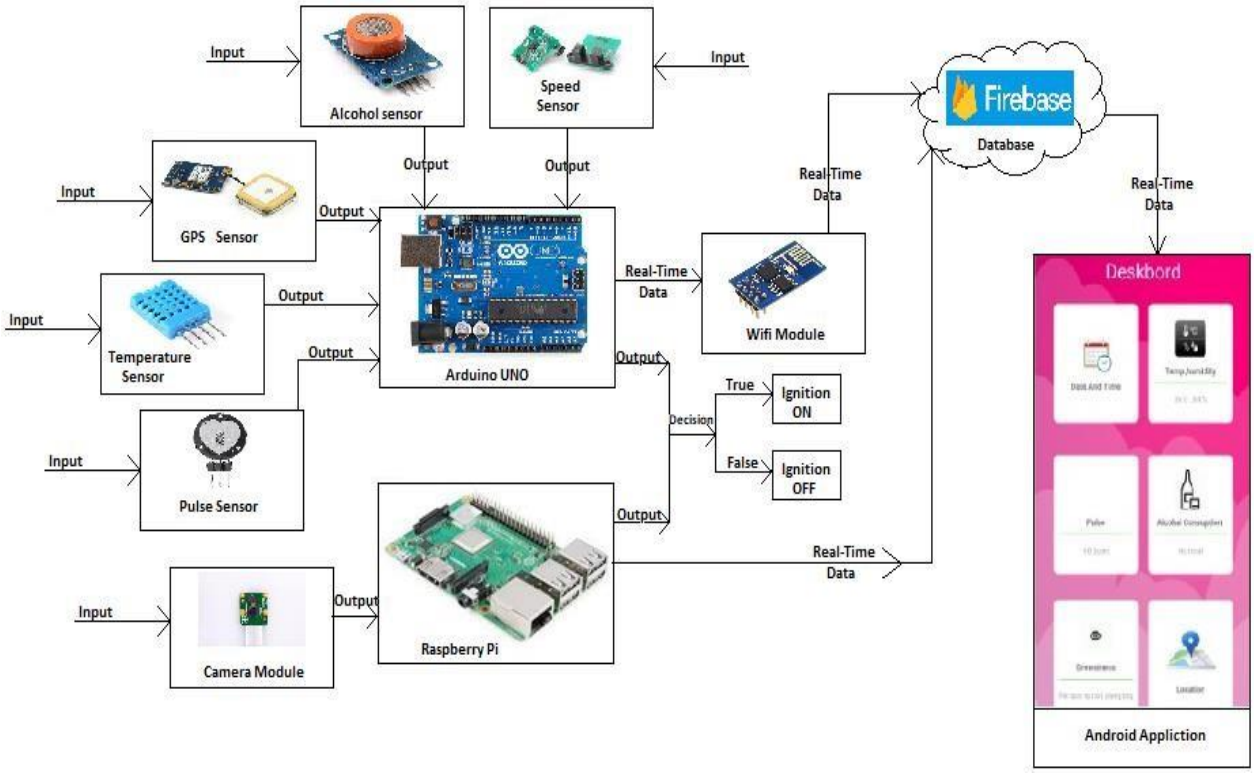


Figure 7
Architecture Diagram

7.1 TECHNICAL ENVIRONMENT

The project is a Python, IOT and Java based app using many libraries like Dlib, OpenCV, SoftwareSerial, TinyGPS etc.

7.2 DROWSINESS SYSTEM ARCHITECTURE

Our driver drowsiness detection system consists of four main stages.

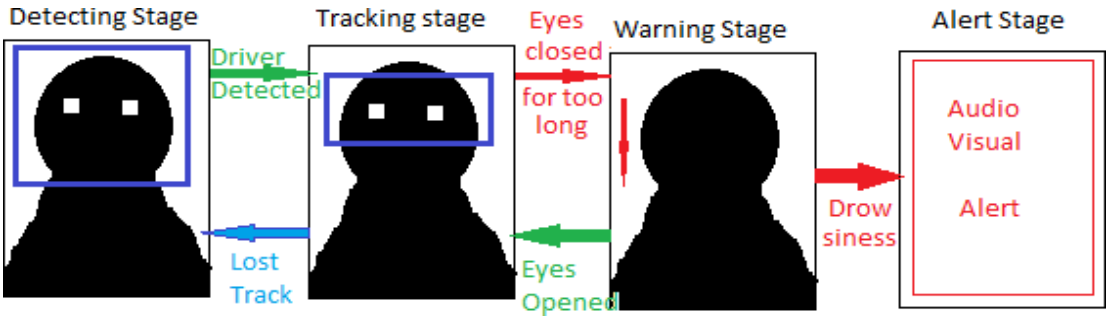


Figure 7.2.1
drowsiness system architecture

1. **Detection Stage**: This is the initialization stage of the system. Every time the system is started it needs to be set up and optimized for current user and conditions. The key step in this stage is successful head detection. If the driver's head is correctly located, we can proceed to extract the features necessary for setting up the system. Setup steps include:
(I) extracting driver's skin color and using that information to create custom skin color model and
(II) collecting a set of open/closed eyes samples, along with driver's normal head position. To help achieve these goals, user interaction might be required. The driver might be asked to sit comfortably in its normal driving position so that system can determine upper and lower thresholds needed for detecting potential nodding. The driver might also be asked to hold their eyes closed and then open for a matter of few seconds each time. This is enough to get the system started. Over time, the system will expand the dataset of obtained images and will become more error resistant and overall, more robust.

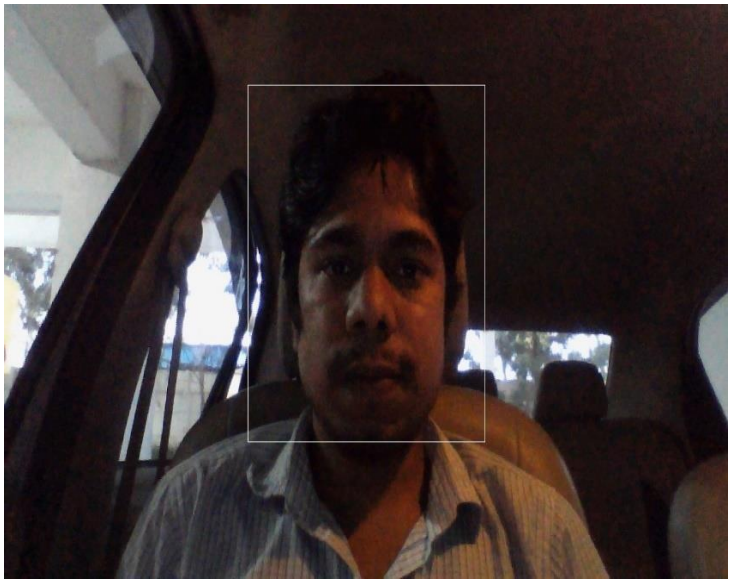


Figure 7.2.2
Detection Stage

2. **Tracking Stage:** Once the driver's head and eyes are properly located and all the necessary features are extracted, the system enters the regular tracking (monitoring) stage. A key step in this stage is the continuous monitoring of the driver's eyes within a dynamically allocated tracking area. More specifically, in order to save some processing time, the system will determine the size of the tracking area based on the previous history of eye movements. For example, if the eyes were moving horizontally to the left for a number of frames it is to be expected that that trend with continue in the following frame also. So, it is logical to expand the tracking area towards the expected direction of the eyes and shrink the area in other three directions. During this stage, the system must also determine the state of the eyes. All these tasks must be carried out in Realtime; depending on the processor's abilities and current load, it might be necessary to occasionally skip a few frames, without sacrificing algorithmic accuracy.

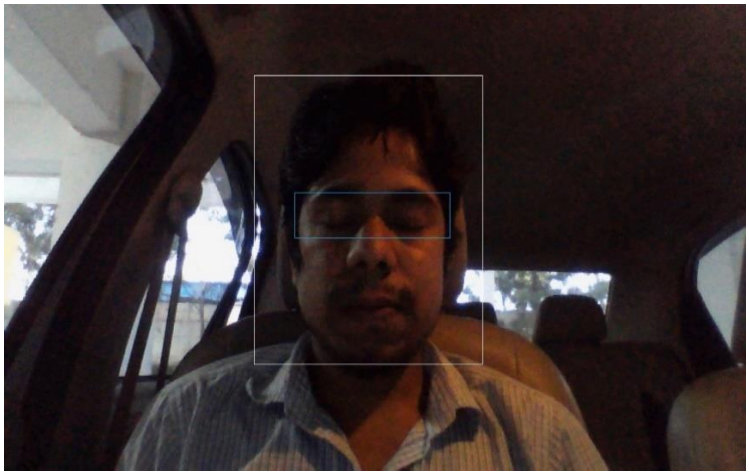


Figure 7.2.3
Tracking Stage

3. **Warning Stage**: If the driver keeps his eyes closed for prolonged period of time or starts to nod, alertness has to be raised. The key step within this stage is close monitoring of driver's eyes. The system must determine whether the eyes are still closed, and what is the eyes' position relative to previously established thresholds. We cannot afford to skip frames in this stage. In practice, tracking of eyes is performed much in the same way as in the tracking stage with the addition of the following processes: calculation of velocity and trajectory of the eyes and threshold monitoring. These additional computations are required to improve the system's ability to determine whether the driver is drowsy or not.

4. **Alert Stage:** Once it has been determined that the driver appears to be in an abnormal driving state, the system has to be proactive and alert the driver of potential dangers that can arise. Combination of audio/visual alerts are used to attract the driver's attention and rase their alertness level. Alerting has to be implemented in such a way as not to cause the opposite effect of intended and startle the driver into causing an accident.

7.3 Sensors Connectivity Architecture

In this section we have covered architecture of different sensors and their connectivity with the proposed system.

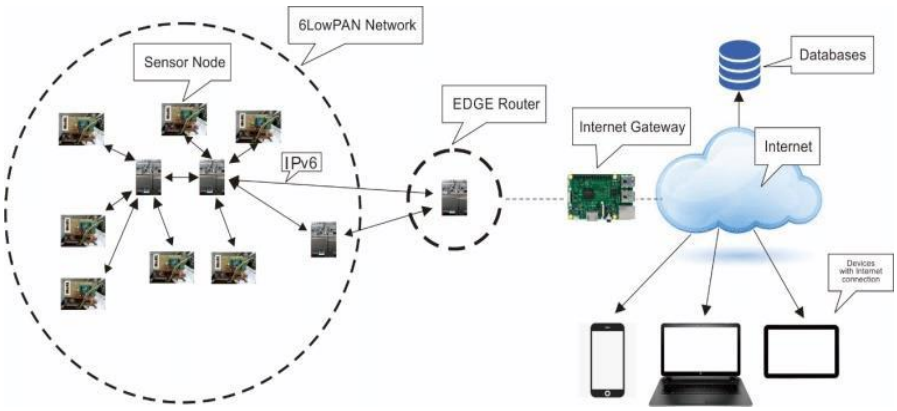


Figure 7.3
Sensors Connectivity Architecture

7.3.1 Alcohol Sensor architecture

If the driver is drunk, the alcohol sensor fixed to the microcontroller will identify the alcohol content and its output will be triggered. This output of the sensor will be sent as an input to the microcontroller. Thus, the microcontroller continuously sends real time sensed alcohol detected values to the WIFI application where somebody (family or owner or police) is using at the scene, now android application analyses the streaming sensor values and display the level of alcohol detected.

If the alcohol level is above the normal range then it displays the detected level along with corresponding violated due amount and also sends that information to the somebody (family

or owner or police) data base and in turn violation due amount will be placed message to the driver.

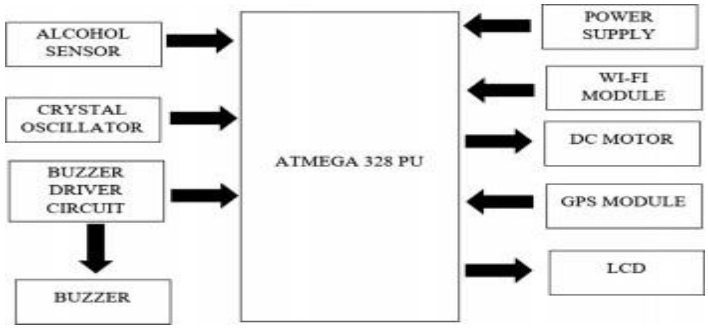


Figure 7.3.1
Alcohol Sensor architecture

7.3.2 Temperature Sensor architecture

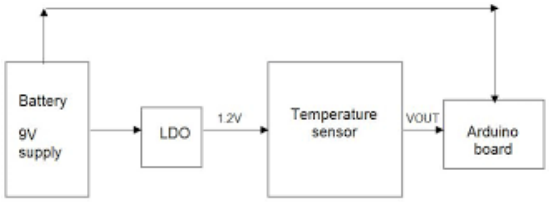


Figure 7.3.2(1)
Temperature Sensor architecture

Figure 7.3.2(2)
Real-time image

7.3.3 Pulse Sensor Architecture

Connect Pulse Sensor to Arduino Uno Board as following:

- + to +5V.

- - to GND.
- S to A0 of Arduino.

2. Connect LCD to Arduino Uno Board as following:

- VSS to +5V of Arduino.
- VDD to GND
- RS to D12 of Arduino.
- RW to GND
- E to D9 of Arduino.
- D4 to D7 of Arduino.
- D5 to D6 of Arduino.
- D6 to D5 of Arduino.
- D7 to D4 of Arduino.
- A/VSS to +5V
- K/VDD to GND

3. Connect 10K Potentiometer to LCD as following.

- GND to GND.
- Data to VEE.
- VCC to +5V.

4. Connect LED to Arduino as following:

- LED1 (RED, blinkPin) to D13 of Arduino.
- LED2 (GREEN, fadeRate) to D8 of Arduino.

5. Connect WIFI module as following.

- GND to GND
- VCC to +3.3V
- RXD pin to D10 of Arduino.

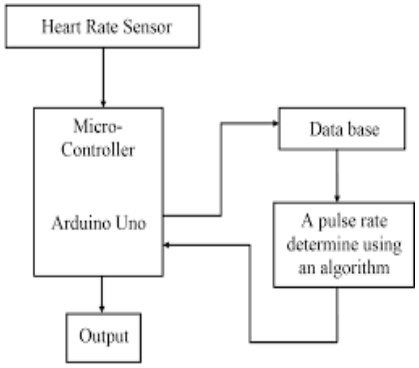


Figure 7.3.3

7.4 Android Application Architecture

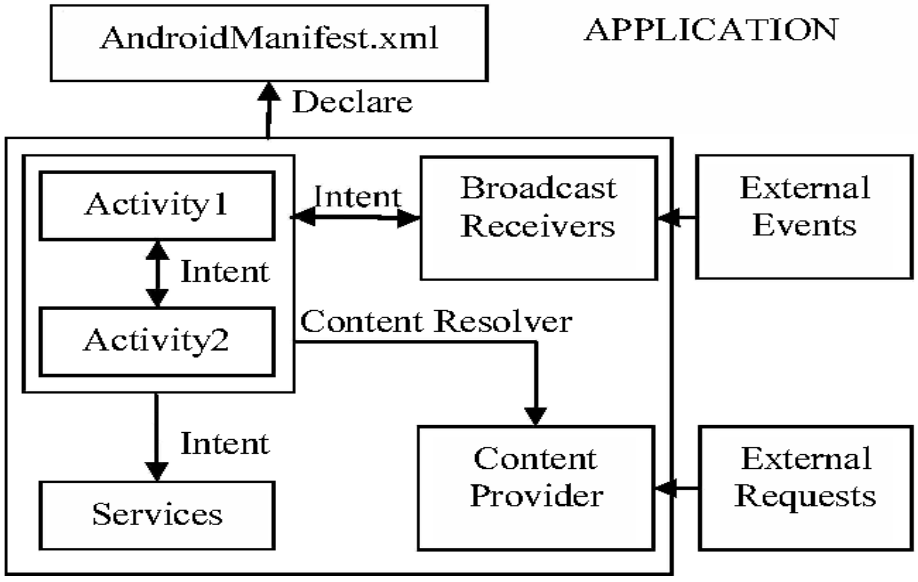


Figure 7.4.1
Android Application Architecture

7.5 DURATION OF THE PROJECT

As the project is divided into two parts the first one is drowsiness part and other one sensor part and android part. So, there is different durations of every individual part.

Sl no	PART OF PROJECT	DURATION
1	Sensors and modules buying and study about them.	2 weeks
2	Code for sensors and for drowsiness	4 weeks
3	Researches and study old references	2 weeks
4	Android application and connectivity with sensors	2 weeks
5	Training data set for drowsiness	3 weeks
6	Connecting everything with firebase	1 week
7	Overall implementation	1 week

Table 7.5

7.6 Real-Time Images

Figure 7.6.1

Real-Time Image

Figure 7.6.2

Real-Time Image

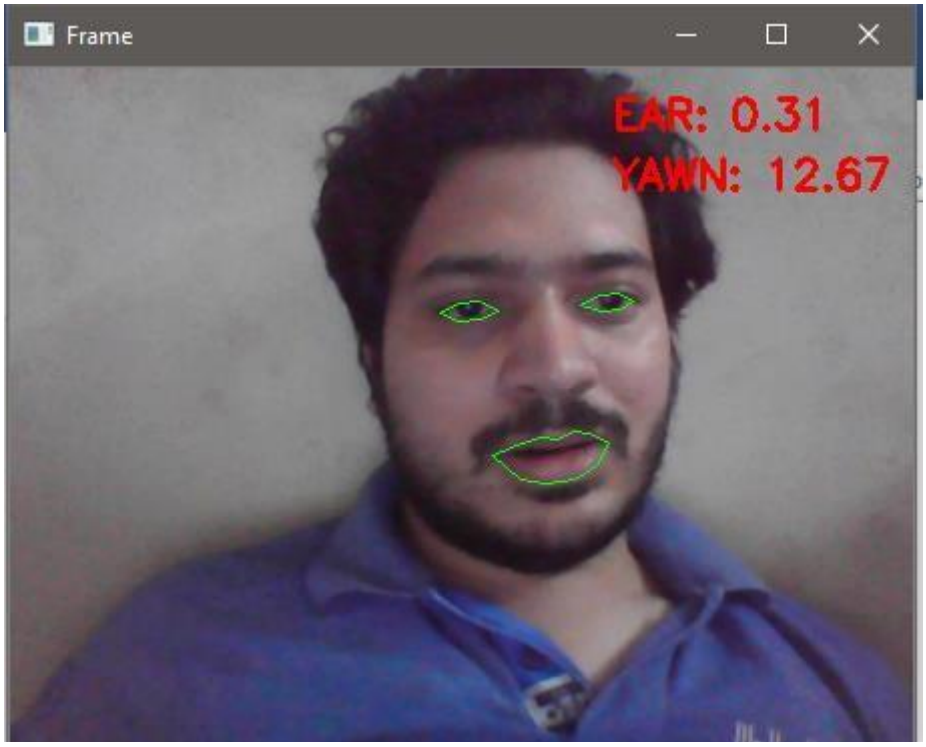


Figure 7.6.3

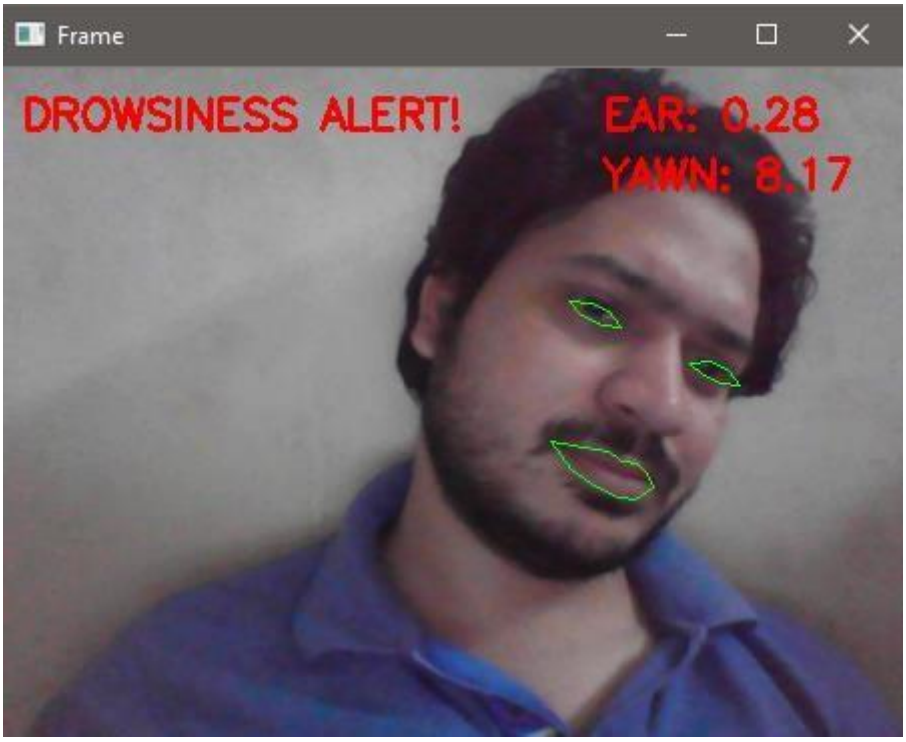


Figure 7.6.4

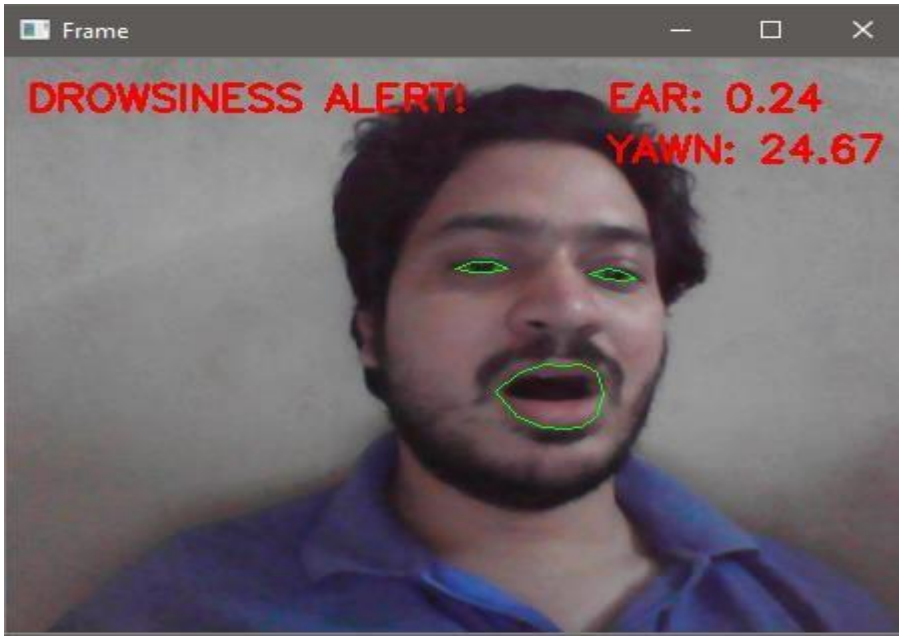


Figure 7.6.5

Figure 7.6.6

CHAPTER -8

TESTING

8.1 INTRODUCTION

This product is divided in four main parts and these parts have their own design fashions. These parts are sensors, drowsiness detection, cloud and mobile application. The project aims to develop a smart steering wheel for improving driver's safety. The project can be used to monitor a person sitting on driver seat.

8.1.1 Features to be Tested

Software in Arduino Board is going to be tested deployed with different combinations of sensors. Bandwidth of WIFI module and serial port of Arduino will be in the features to be tested.

Drowsiness will be tested using camera module and the trained machine using some algorithm.

Database in cloud will be tested for different cases. Data collection software will be tested connected to the database. Analysis tests are going to be held with differentiating conditions.

Phone application tests will be held for notifications and availability.

Protocols between components of system are subjects to the test. System-wide integration can be included in features to be tested as well.

8.1.2 Features not to be Tested

Device dependency tests for hardware architectures other than already stated ones will be omitted. Arduino board is able to employ more than three sensors but no additional sensors will be tried. Rationale behind that is lack of available hardware components and undecided design.

Third-party open source libraries usually conduct their own tests, so tests for those libraries will only take place in integration and unit tests of software implemented.

User stress tests for server will not exceed a certain limit because, for a brand new product, hardware requirements for scaling are subjects of successive test documents. Security tests will also be postponed for this version of document.

For each level, tests will be held using simulated inputs flow originated from lower levels since real world inputs are not sufficiently required.

8.1.3 Item Pass/Fail Criteria

There are several test categories which specify fail criteria for different tests:

- Integrity Checks: If integrity of values are not preserved during relay operations, test is failed.
- Connectivity Tests: If packages or connection is lost, test is failed.
- Analysis Accuracy Checks: If analysis accuracy is lower than 70%, test is failed.
- Integration Tests: For integration tests, predecessor component shall be available for test status to be passed, otherwise it is failed.
- Performance Tests: There is no specification of delays that is not negligible in this version of document.
- Unit Tests: Unit crash or permanent failure means that test is failed.

8.1.4 Suspension Criteria and Resumption Requirements

During integration tests, if one lower level fails, tests associated with the lower level are invalidated. In such a case, tests for both levels shall be repeated. For system integration tests, any type of component crash requires this test to be suspended. Unit tests for crashed item shall be conducted again.

8.2. TEST MANAGEMENT

Best way to test the whole system is creating subsystem tests. This approach is superior to complete system testing because subcomponents are pretty big already. Their interactions and communications with each other should be well defined and work steadily. Finally, the complete system integration and testing must be performed. Thus, the workload of testing schema is as follows:

8.2.1. Testing a sub component within itself

This process ensures that the component is working without any interaction by any other sub components. To be able to conduct this test procedure for the wearable subsystem, in addition to cores of the hardware such as micro•controller, pulse sensor, temperature sensor, camera module, we need a serial connection to a workstation. This will allow us to be able analyses & debug the embedded code much more efficiently. On the other hand, remaining subsystems do not depend on specialized hardware/software parts. Nevertheless, their unit tests are written by corresponding programming languages (Python / C) and deployed individually on them.

8.2.2. Testing the communication protocols and interactions between adjacent sub-system components

Purpose of this part is to stabilize inter subsystem communications. Techniques are mostly composed of erroneous situation generation. As defects reveals the real-life situations more realistically. Sending an invalid JSON object from fixed component to cloud server might be an example. In fact, when exactly an exception occurs and how subsystems react those are the heart of this test activity.

8.2.3. Integration of the complete system and testing

This is the final test scenario and describes how will the complete system work when it is deployed in real life. Thus, conducting stress tests for the system is necessary obviously. To make stress testing effective, several duplicate simulators will be designed and executed. Those simulators mimic realistic behaviors of the subsystems. As an example, wearable component simulators generate data for the fixed component.

8.3. SYSTEM TEST LEVELS

This part describes the different test levels based on wearable component level, fixed component level, cloud component level etc.

8.3.1. Wearable Component Level

8.3.1.1. Wearable Test 1

Test Case Identifier	WEARABLE•TEST•01
Objective	Correctness of temperature sensor measurements
Scenario	Heating and cooling the temperature sensor via external inputs
Input	Holding the analog temperature sensor for a while and blowing the cold air to the sensor afterwards.
Outcome	Temperature output of the micro•controller smoothly rises first and falls later on.
Requirements	Avoidance of extraordinary noise interruption

Table 8.3.1.1

8.3.1.2. Wearable Test 2

Test Case Identifier	WEARABLE•TEST•02
Objective	Correctness of pulse sensor measurements
Scenario	Breathing heavily when equipped with pulse sensor on the finger
Input	Increasing rate of the person's heart rate
Outcome	Heart beat per minute data output of the micro•controller rises

Requirements	Equipping the pulse sensor properly so that no extra environmental light disturbs the led of the sensor

Table 8.3.1.2

8.3.1.3. Wearable Test 3

Test Case Identifier	WEARABLE•TEST•03
Objective	Correctness of alcohol measurements
Scenario	Blowing without alcohol, spray ethanol, spray alcohol.
Input	If consumed red light blink otherwise green light will blink
Outcome	Rapid changes for the accelerometer output of the micro•controller
Requirements	Equipping the alcohol sensor properly so that no extra air disturbs the testing part of the sensor

Table 8.3.1.3

8.3.1.4 Wearable Test 4

Test Case Identifier	WEARABLE•TEST•05
Objective	WIFI connection establishment
Scenario	Bringing the peer device closer and send it away later on
Input	Increasing the power level of the WIFI signal, then decreasing it.
Outcome	First, micro•controller sets up a WIFI connection with the peer and sends measured data to it. Whenever the peer is far away from the micro•controller, connectivity drops. The whole scenario is looped again when the peer is getting closer.
Requirements	Necessary configurations on WIFI prior to connection

Table 8.3.1.4

8.3.1.5. Wearable Test 5

Test Case Identifier	WEARABLE•TEST•05
Objective	Camera module establishment
Scenario	Bringing the person closer and send it away later on

Input	Bringing the person face near the camera.
Outcome	If the person closes the eye for long time the buzzer will alert for the drowsiness.
Requirements	Necessary configurations of camera with raspberry pi.

Table 8.3.1.5

8.3.2 Fixed Component Level

8.3.2.1. Fixed Component Test 1

Test Case Identifier	FIXED•TEST•02
Objective	Getting Sensor Data
Scenario	Sensor data is obtained from wearable device.
Input	Data queue of a sensor periodically enqueued with new sensor data
Outcome	Periodically obtaining the same data from queue concurrently
Requirements	A simulator which generates sensor data or wearable device input is required

Table 8.3.2.1

8.3.2.2. Fixed Component Test 2

Test Case Identifier	FIXED•TEST•02
Objective	Regular Sender Periodic Send Operation
Scenario	Given a period by configuration file, regular sender thread works every N seconds to wipe out the output sensors, pack the sensor data and send them to cloud.
Input	Regular sensor data.
Outcome	Threads work concurrently and without starvation
Requirements	FIXED•TEST• 01 should be successful

Table 8.3.2.2

8.3.2.3. Fixed Component Test 3

Test Case Identifier	FIXED•TEST•03
Objective	Urgent Sender Immediately Send Operation

Scenario	In case of emergency identified by analyser, urgent send er thread sends the emergent situation data immediately to the server.
Input	Urgent sensor data.
Outcome	If network delays are ignored, there is no delay when sending data to cloud
Requirements	FIXED•TEST• 01 should be successful

Table 8.3.2.3

8.3.3. Cloud Component Level

8.3.3.1. Cloud Component Test 1

Test Case Identifier	CLOUD•TEST•01
Objective	To test the register component of the system
Scenario	User sends register info via mobile application
Input	name, surname, address, phone email, password, device id
Outcome	A new user is created in the system.
Requirements	The mobile application should be started

Table 8.3.3.1

8.3.3.2. Cloud Component Test 2

Test Case Identifier	CLOUD•TEST•02
Objective	To test the login component of the system with valid data
Scenario	User sends login info via mobile application
Input	username, password
Outcome	The user is logged in to the application.
Requirements	The mobile application should be started and user must be registered.

Table 8.3.3.2

8.3.3.3. Cloud Component Test 3

Test Case Identifier	CLOUD•TEST•03
Objective	To test the logout functionality.
Scenario	User sends logout info via mobile application
Input	None

Outcome	The session should be invalidated.
Requirements	The mobile application should be started and user must be logged in.

Table 8.3.3.3

8.3.4 Mobile Application Component Level

8.3.4.1. Mobile Test 1

Test Case Identifier	MOBILE•TEST•01
Objective	Testing the Emergency Notification Systems' Registration procedure
Scenario	When a user installs the mobile application and runs it, he or she will be required to enter an emergency contact number
Input	Providing the emergency contact number
Outcome	Emergency contact number will be registered to cloud automatically
Requirements	•

Table 8.3.4.1

8.3.4.2. Mobile Test 2

Test Case Identifier	MOBILE•TEST•02
Objective	Testing the Emergency Notification
Scenario	An event, such as 'dangerously high temperature', occurred that needs a notification to be sent to emergency contact. Then for such cases, our server sends messages to emergency contacts.
Input	•
Outcome	Emergency contact will be texted.
Requirements	•

Table 8.3.4.2

8.3.5. System Integration Level

8.3.5.1. System Test 1

Test Case Identifier	SYSTEM•TEST•01
Objective	Testing the whole data flow from sensor and camera to cloud
Scenario	Sensors on hardware are working and sampling data with an arbitrary rate.
Input	Sensor's and camera Input
Outcome	Data is transmitted to the cloud

Requirements	User has sensor hardware, fixed device working. Cloud is running.

Table 8.3.5.1

8.4. DIFFERENT TEST CASES

Here We are taking some reading and according to that the system take decision

8.4.1 drowsiness case

Mean Squared Error. The Mean Squared Error (**MSE**) or Mean Squared Deviation (MSD) of an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss.

Performance of the model in predicting drowsiness level with the testing dataset: mean square error (MSE), standard deviation (STD), according to whether dataset is used with (1) or without driving time (0), participant information, and source of recorded information. The * symbol indicates the worst performance and the # symbol the best performance. The best and worst performance are also highlighted in bold.

Driving Time	Participant information	Dataset	Source	MSE	STD	|Error |95%	% Error <5
0	0	Testing	All	33.64	7.63	9.29	0.79
0	0	Testing	Behavioral	23.61	3.15	8.12	0.86
0	0	Testing	Car	**60.09***	6.19	13.12	0.73
0	0	Testing	Physiological	43.77	6.24	11.47	0.74
0	1	Testing	All	28.26	2.82	8.79	0.82
0	1	Testing	Behavioral	22.83	4.03	7.98	0.89
0	1	Testing	Car	50.22	8.84	12.11	0.73
0	1	Testing	Physiological	41.82	4.11	11.83	0.74
1	0	Testing	All	10.64	3.39	4.26	0.97
1	0	Testing	Behavioral	5.46	1.50	2.92	0.99
1	0	Testing	Car	31.14	10.73	6.25	0.93
1	0	Testing	Physiological	15.97	1.70	7.01	0.89
1	1	Testing	All	7.69	2.17	3.12	0.98
1	1	Testing	Behavioral	**4.18#**	1.17	1.98	0.99
1	1	Testing	Car	4.67	1.33	2.43	0.99

Driving Time	Participant information	Dataset	Source	MSE	STD	|Error |95%	% Error <5
1	1	Testing	Physiological	5.51	1.84	2.62	0.98

Table 8.4.1

8.4.2 Alcohol sensor reading

Table 8.4.2(1): Alcohol sensor reading

Normal Sensor Readings	**16**	**20**	**24**	**26**	**27**	**30**	**35**	**40**
Experimental Readings	17	23	26	28	30	34	37	41

Sensitivity level characteristics

VOLATGE (V)	PPM (PART PER MILLION)	PERCENTAGE (%)
0	0	0
0.5	100	10
1	200	20
1.5	300	30
2	400	40
2.5	500	50
3	600	60
3.5	700	70
4	800	80
4.5	900	90
5	1000	100

Table 8.4.2(2)

Level of drunkenness

LEVEL OF DRUNKNESS			
Voltage Output	200 - 300ppm 1 – 1.5V 20 – 30%	300 – 400ppm 1.5 – 2V 30 – 40%	400 – 500ppm 2– 2.5V 40 – 50%
LCD Display	Intoxicated	Slightly Drunk	Drunkenness
Alarm	Off	Off	On
Ignition SYS	On	On	Off
Indicator	Led Green On Led Red Off	Led Green On Led Red Off	Led Red On Led Green Off

Table 8.4.2(3)

CHAPTER -9

CONCLUSION

In this report, we have described the process of designing and implementing a driver drowsiness detection system, driver health monitoring and vehicle speed monitoring and control by combining some off-the-shelf algorithms with some of the novel approaches in a clever manner. The system is dynamic, it can update and modify its components like current driver's eye model and skin model throughout its life cycle. Constantly upgrading baseline models used can increase the overall resilience towards errors. Moreover, the system is user specific. All the feature models created are solely based on the current users features instead of using generalized parameters. Such an approach simplifies the system while providing solid performance. Each of the algorithms is performing solidly by itself. But they do have their limitations. To increase the reliability and accuracy of the system, both baseline detection/tracking algorithm and eye-state classification algorithm are complimented with simple, but efficient, custom algorithms.

APPENDICES

Source Codes

Drowsiness

```
#python drowniness_yawn.py --webcam webcam_index

from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import os

def alarm(msg):
global alarm_status
global alarm_status2
global saying

while alarm_status:
print('call')
s = 'espeak "'+msg+'"'
os.system(s)

if alarm_status2:
print('call')
saying = True
s = 'espeak "' + msg + '"'
os.system(s)
saying = False

def eye_aspect_ratio(eye):
A = dist.euclidean(eye[1], eye[5])
B = dist.euclidean(eye[2], eye[4])

C = dist.euclidean(eye[0], eye[3])
```

```
ear = (A + B) / (2.0 * C)

return ear

def final_ear(shape):
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]

leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)

ear = (leftEAR + rightEAR) / 2.0
return (ear, leftEye, rightEye)

def lip_distance(shape):
top_lip = shape[50:53]
top_lip = np.concatenate((top_lip, shape[61:64]))

low_lip = shape[56:59]
low_lip = np.concatenate((low_lip, shape[65:68]))

top_mean = np.mean(top_lip, axis=0)
low_mean = np.mean(low_lip, axis=0)

distance = abs(top_mean[1] - low_mean[1])
return distance


ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0,
help="index of webcam on system")
args = vars(ap.parse_args())

EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 30
YAWN_THRESH = 20
alarm_status = False
alarm_status2 = False
saying = False
```

```
COUNTER = 0

print("-> Loading the predictor and detector...")
#detector = dlib.get_frontal_face_detector()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")    #Faster but less
accurate
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')


print("-> Starting Video Stream")
vs = VideoStream(src=args["webcam"]).start()
#vs= VideoStream(usePiCamera=True).start()       //For Raspberry Pi
time.sleep(1.0)

while True:

frame = vs.read()
frame = imutils.resize(frame, width=450)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

#rects = detector(gray, 0)
rects = detector.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5, minSize=(30, 30),
flags=cv2.CASCADE_SCALE_IMAGE)

#for rect in rects:
for (x, y, w, h) in rects:
rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))

shape = predictor(gray, rect)
shape = face_utils.shape_to_np(shape)

eye = final_ear(shape)
ear = eye[0]
leftEye = eye [1]
rightEye = eye[2]

distance = lip_distance(shape)

leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
```

```
cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

lip = shape[48:60]
cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

if ear < EYE_AR_THRESH:
COUNTER += 1

if COUNTER >= EYE_AR_CONSEC_FRAMES:
if alarm_status == False:
alarm_status = True
t = Thread(target=alarm, args=('wake up sir',))
t.deamon = True
t.start()

cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

else:
COUNTER = 0
alarm_status = False

if (distance > YAWN_THRESH):
cv2.putText(frame, "Yawn Alert", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
if alarm_status2 == False and saying == False:
alarm_status2 = True
t = Thread(target=alarm, args=('take some fresh air sir',))
t.deamon = True
t.start()
else:
alarm_status2 = False

cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)


cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
```

```
if key == ord("q"):
break

cv2.destroyAllWindows()
vs.stop()
```

Arduino

```
#include <dht.h>
#include <Wire.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>
#include <TinyGPS.h>
SoftwareSerial mySerial(10, 9);
TinyGPS gps;
void gpsdump(TinyGPS &gps);
void printFloat(double f, int digits = 2);

LiquidCrystal lcd(12, 11, 7, 6, 5, 4);
//pulse
 double alpha=0.75;
   int period=20;
   double refresh=0.0;
//alcohal
int ledPin = 13;
int sensorPin = A1;
int value;

//dht
dht DHT;
#define DHT11_PIN 8


void setup()
{



   pinMode(ledPin,OUTPUT);
lcd.begin(16,2);
lcd.print("Welcome");
```

```
  delay(5000);
   Serial.begin(9600);
  // set the data rate for the SoftwareSerial port
  mySerial.begin(9600);
  delay(1000);
  Serial.println("uBlox Neo 6M");
  Serial.print("Testing TinyGPS library v. "); Serial.println(TinyGPS::library_version());
  Serial.println("by vikash,vidya");
  Serial.println();
  Serial.print("Sizeof(gpsobject) = ");
  Serial.println(sizeof(TinyGPS));
  Serial.println();
}

void loop()
{
delay(1000);
int chk = DHT.read11(DHT11_PIN);
switch (chk)

lcd.clear();

lcd.setCursor(0,0);
lcd.print("Temp = ");
lcd.print(DHT.temperature, 0);
lcd.print("'C ");
lcd.print(DHT.temperature * 1.8 + 32, 0);
lcd.print("F");


lcd.setCursor(0,1);
lcd.print("Humidity = ");
lcd.print(DHT.humidity, 0);
lcd.print("%");
delay(6000);
lcd.clear();
if(DHT.temperature * 1.8 + 32>99.2)
   {
    digitalWrite(ledPin,HIGH);
   }
  else
  {
```

```
    digitalWrite(ledPin,LOW);

  }
static double oldValue=0;
   static double oldrefresh=0;

   int beat=analogRead(A0);

   double value=alpha*oldValue+(0-alpha)*beat;
   refresh=value-oldValue;



   lcd.setCursor(0,0);
   lcd.print(" Heart Monitor ");
   lcd.setCursor(0,1);
   lcd.print("  ");
   lcd.print("bpm");
   lcd.setCursor(0,1);
   lcd.print(beat/10);
   oldValue=value;
   oldrefresh=refresh;
   delay(8000);

lcd.clear();

int Value = analogRead(sensorPin);
value = analogRead(A1);
lcd.print("Alcohol Lev.:");
lcd.print(value-50);
Serial.print(value);
if (value-50 > 500)

{
    digitalWrite(ledPin,HIGH);
     lcd.setCursor(0, 2);
     lcd.print("Alert... !!!");

  }
  else {
    digitalWrite(ledPin,LOW);
    lcd.setCursor(0, 2);
```

```
    lcd.print(".....Normal.... ");

  }

delay(5000);
lcd.clear();

 bool newdata = false;
  unsigned long start = millis();
  // Every 5 seconds we print an update
  while (millis() - start < 5000)
  {
    if (mySerial.available())

    {
      char c = mySerial.read();
      //Serial.print(c); // uncomment to see raw GPS data
      if (gps.encode(c))
      {
        newdata = true;
        break; // uncomment to print new data immediately!
      }
    }
  }

  if (newdata)
  {
    Serial.println("Acquired Data");
    Serial.println("------------ ");
    gpsdump(gps);
    Serial.println("------------ ");
    Serial.println();
  }

}

void gpsdump(TinyGPS &gps)
{
  long lat, lon;
  float flat, flon;
  unsigned long age, date, time, chars;
  int year;
```

```
  byte month, day, hour, minute, second, hundredths;
  unsigned short sentences, failed;

  gps.get_position(&lat, &lon, &age);
  Serial.print("Lat/Long(10^-5 deg): "); Serial.print(lat); Serial.print(", "); Serial.print(lon);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");

  // On Arduino, GPS characters may be lost during lengthy Serial.print()
  // On Teensy, Serial prints to USB, which has large output buffering and
  // runs very fast, so it's not necessary to worry about missing 4800
  // baud GPS characters.

  gps.f_get_position(&flat, &flon, &age);
 Serial.print("Lat/Long(float): ");
 printFloat(flat, 5);
Serial.print(", ");
printFloat(flon, 5);
Serial.print(" Fix age: ");
Serial.print(age);
Serial.println("ms.");

  gps.get_datetime(&date, &time, &age);
Serial.print("Date(ddmmyy): ");
Serial.print(date);
Serial.print(" Time(hhmmsscc): ");
   Serial.print(time);
  Serial.print(" Fix age: ");
Serial.print(age);
Serial.println("ms.");

  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths,
&age);
  Serial.print("Date: ");
Serial.print(static_cast<int>(month));
Serial.print("/");
    Serial.print(static_cast<int>(day));
Serial.print("/");
Serial.print(year);
  Serial.print(" Time: ");
Serial.print(static_cast<int>(hour+8));
 Serial.print(":");
//Serial.print("UTC +08:00 Malaysia");
```

```
    Serial.print(static_cast<int>(minute));
Serial.print(":");
Serial.print(static_cast<int>(second));
    Serial.print(".");
Serial.print(static_cast<int>(hundredths));
Serial.print(" UTC +08:00 Malaysia");
Serial.print(" Fix age: ");
Serial.print(age);
 Serial.println("ms.");

  Serial.print("Alt(cm): "); Serial.print(gps.altitude()); Serial.print(" Course(10^-2 deg): ");
    Serial.print(gps.course()); Serial.print(" Speed(10^-2 knots): ");
Serial.println(gps.speed());
  Serial.print("Alt(float): "); printFloat(gps.f_altitude()); Serial.print(" Course(float): ");
    printFloat(gps.f_course()); Serial.println();
  Serial.print("Speed(knots): "); printFloat(gps.f_speed_knots()); Serial.print(" (mph): ");
    printFloat(gps.f_speed_mph());
  Serial.print(" (mps): "); printFloat(gps.f_speed_mps()); Serial.print(" (kmph): ");
    printFloat(gps.f_speed_kmph()); Serial.println();

  gps.stats(&chars, &sentences, &failed);
  Serial.print("Stats: characters: "); Serial.print(chars); Serial.print(" sentences: ");
    Serial.print(sentences); Serial.print(" failed checksum: "); Serial.println(failed);
}

void printFloat(double number, int digits)
{
  // Handle negative numbers
  if (number < 0.0)
  {
     Serial.print('-');
     number = -number;
  }

  // Round correctly so that print(1.999, 2) prints as "2.00"
  double rounding = 0.5;
  for (uint8_t i=0; i<digits; ++i)
    rounding /= 10.0;

  number += rounding;

  // Extract the integer part of the number and print it
```

```
unsigned long int_part = (unsigned long)number;
double remainder = number - (double)int_part;
Serial.print(int_part);

// Print the decimal point, but only if there are digits beyond
if (digits > 0)
  Serial.print(".");

// Extract digits from the remainder one at a time
while (digits-- > 0)
{
  remainder *= 10.0;
  int toPrint = int(remainder);
  Serial.print(toPrint);
  remainder -= toPrint;
}
```

REFERENCES

[1] A. Malla, P. Davidson, P. Bones, R. Green and R. Jones, "Automated Video-based Measurement of Eye Closure for Detecting Behavioral Microsleep", in 32nd Annual International Conference of the IEEE, Buenos Aires, Argentina, 2010.

[2] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", in Proceedings of the IEEE Computer Society Conference on
Computer Vision and Pattern Recognition, 2001.

[3] OpenCV. Open Source Computer Vision Library Reference Manual, 2001.

[4] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", IEEE Transactions on Systems,Man and Cybernatics, pp. 62-66, 1979.

[5] I. T. S. D. a. A. Group, I. T. Forum, "IRTAD road safety annual report 2015", Organisation for Economic Co-operation and Develop, 2015.

[6] J. Lavanya, R. E. Raj, "A Mobile Based Novice Detection of Driver's Fatigue Level and Accident Reporting Solution", Power Electronics and Renewable Energy Systems Proceedings of ICPERES 2014, vol. 326, pp. 883-892, 2015.

[7] L. Hanwei eletronics co, MQ-3 Gas Sensor Datasheet, 2016.

[8] M. Piccardi, "Background subtraction techniques: a review", International Conference on Systems Man and Cybernetics 2004 IEEE, vol. 4, pp. 3099-3104, 10-13 October 2004.

[9] D. Sawicki, Traffic Radar Handbook: A comprehensive Guide to Speed Measuring Systems, Author House, 2002.

[10] Joel L. Wilder, Aleksandar Milenkovic, Emil Jovanov, "Smart Wireless Vehicle Detection System", The 40th Southeastern Symposium on System Theory, pp. 159-163, March 2008.

[11] World Health Organization, Global Status Report on Road Safety 2013: Supporting a Decade of Action: Summary. World Health Organization, 2013. [Online]. Available: http://books.google.com/books?id=qzK2nQEACAAJ

[12] T. Akertedt, P. Fredlung, M. Gillberg, and B. Jansson, “A prospective study of fatal occupational accidents relationship to sleeping difficulties and occupational factors,” Journal of Sleep Research, vol. 11, no. 1, pp. 69–71, 2002. [Online]. Available: http://dx.doi.org/10.1046/j.13652869.2002.00287.x

[13] S. H. Fairclough and R. Graham, “Impairment of driving performance caused by sleep deprivation or alcohol: A comparative study,” Human Factors: The Journal of the Human Factors and Ergonomics Society, vol. 41, no. 1, pp. 118–128, 1999. [14] R. Feng, G. Zhang, and B. Cheng, “An on-board system for detecting driver drowsiness based on multi-sensor data fusion using dempstershafer theory,” in Networking, Sensing and Control, 2009. ICNSC ’09. International Conference on Networking, 2009, pp. 897–902.

[15] E. Vural, “Video based detection of driver fatigue,” Ph.D. dissertation, Sabanci University, 2009.

[16] P. Viola and M. Jones, “Robust real-time object detection,” in International Journal of Computer Vision, 2001.

[17] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: http://dx.doi.org/10.1007/BF00994018
[18]. W. R. Miles, Alcohol and human efficiency experiments with moderate quantities and dilute solutions of ethyl alcohol on human subjects, Carnegie Institution of Washington, Mar. 1924.
[19]. H. Moskowitz and S. Sharma, "A behavioral mechanism of alcohol-related accidents", *1st Ann. Conf. National Institutes of Alcohol Abuse and Alcoholism*, 1971-June-26.
[20]. Y. P. Tsividis, "Accurate analysis of temperature effects in Ic-VBE characteristics with application to bandgap reference sources", *IEEE J. Solid-State Circuits*, vol. 15, pp. 1076-1084, Dec. 1980.

ACCOMPLISHMENTS

Name of Inventor	Title	Patent Number
Mr. Vikash Kumar Thakur (Student) Mr. Vidya Sagar Jha (Student) Dr. S. Pravinth Raja (faculty)	Smart Steering Wheel for Improving Driver's Safety	202041007193

Vikash Kumar Thakur, Vidya Sagar Jha, Dr. S. Pravinth Raja "Smart Steering Wheel For Improving Driver's Safety" Presented in International Conference on Big Data and Business Analytics conducted by Department of CSE, M. Kumarasamy College of Engineering on 4th April 2020 in association with Indian Institute of Management Bangalore (IIMBx).