A

PROJECT REPORT

ON

“DRIVING MONITORING AND ANALYSIS”

SUBMITTED TO THE

SHIVAJI UNIVERSITY KOLHAPUR

IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE

DEGREE OF

BACHELOR OF ENGINEERING (ELECTRONICS ENGINEERING)

SUBMITTED BY

Mr. Ranjit Patil

Mr. Aniruddha Shahapurkar

Mr. Megharaj Bhosale

Mr. Burhanuddin Bharmal

UNDER THE GUIDANCE OF

Prof. V. K. Desai

DEPARTMENT OF ELECTRONICS

K.I.T's COLLEGE OF ENGINEERING,

KOLHAPUR YEAR 2016-2017

DEPRTMENT OF
ELECTRONICS ENGINEERING

KOLHAPUR INSTITUTE OF TECHNOLOGY'S
COLLEGE OF ENGINEERING KOLHAPUR

CERTIFICATE

This is to be certify that following group members

1. Mr. Ranjit P. Patil
2. Mr. Aniruddha A. Shahapurkar
3. Mr. Megharaj S. Bhosale
4. Mr. Burhanuddin A. Bharmal

Of B.E. (Electronics Engineering) have successfully completed their project work entitled "DRIVING MONITORING AND ANALYSIS" towards the partial fulfillment of the award of Bachelor of Engineering (Electronics) degree as laid by the Shivaji University Kolhapur. During the academic year 2016-2017.

Prof. V. K. Desai
Guide

Dr. Y.M.Patil
H.O.D
(Department of Electronics)

EXTERNAL EXAMINER

Dr. V.V. Karjinni
Principal

Acknowledgement

It gives us immense pleasure to express our sincere gratitude for constant help, encouragement and suggestions to us for our project report entitled **"DRIVING MONITORING AND ANALYSIS"** Under the guidance of Prof. V.K. Desai. We are thankful to him for guiding us through various difficulties and making it look easier.

We would also like to extend our sincere gratitude to Principal Dr. V. V. Karjinni KIT's College of Engineering and Dr.Y.M.Patil H.O.D., Electronics Engineering for their whole support and guidance and their keen interest during the process of our project. Without the inspiration and encouragement, the completion of project would be a difficult task.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

OBD	On Board Diagnostic
GUI	Graphical User Interface

ABSTRACT

Nowadays, vehicles are most important part of human life. We can't even imagine life without vehicles. Cars are one of the most essential parts of our life. We know that, road accident is one of the serious problems we are facing. According to a survey around 382 people in India are killed every day due to road accidents. Ninety-seven per cent of road accidents in the state in 2013 was owing to rash driving, says the statistics brought out by the State Crime Records Bureau.

One of the major reasons of these accidents is rash driving. So, it is very important to monitor the driving skills of the driver. If we will be able to monitor whether the driver is driving the car correctly or not & also to generate a report based on the driving skills of the driver, we will be able to evaluate drivers & also can charge them penalties if rash driving is detected. Which in turn will reduce the accidents due to rash driving. Also, the rash driving of cars affects the car performance such as reduction in fuel economy, reduction in the performance of engine, wearing & burning of some mechanical parts in some cases, etc.

The main aim of this project is to implement a system which will monitor the driver performance & will generate a report depending upon the driving skills of the driver. Rash driving detection will be based upon certain parameters such as speeding the car more than rated speed, accelerating the car more than its respective gear limits, taking inappropriate turns, etc. Also, if car is driven efficiently within certain limits, it will also in turn increase the car performance along with reduction in number of accidents.

INDEX

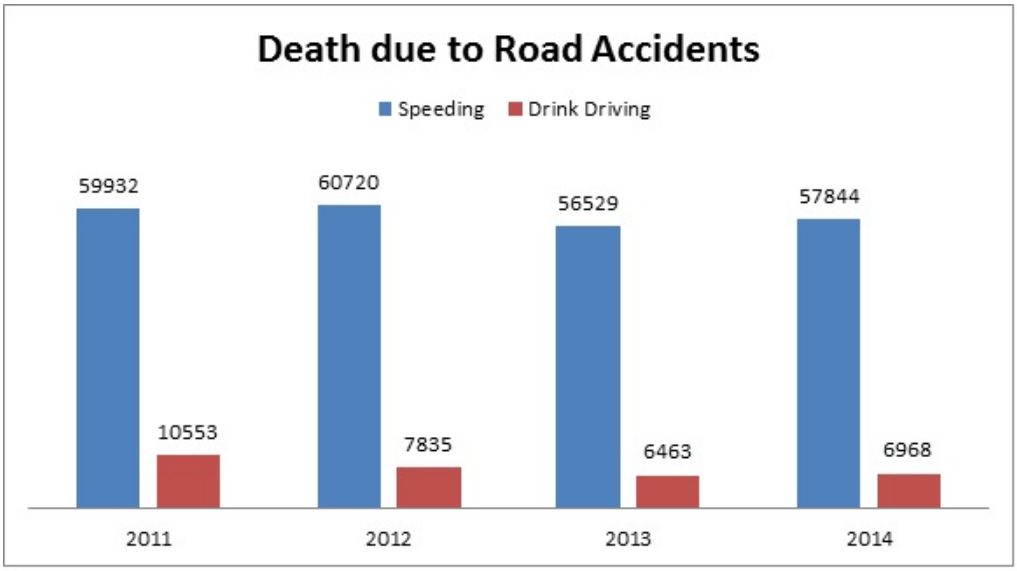
CHAPTER 1

INTRODUCTION

1.1 GENERAL

We know that, road accident is one of the serious problems we are facing One of the major reasons of these accidents is inefficient & rash driving. So, it is very important to monitor the driving skills of the driver. Another issue with rash and inefficient driving is life span of vehicles. Large amount of fuel consumption due to inefficient driving is again a big issue.

If we will be able to monitor whether the driver is driving the car correctly or not & also to generate a report based on the driving skills of the driver, we will be able to evaluate drivers & also can charge them penalties if rash driving is detected. Which in turn will reduce the accidents due to rash driving.



The main aim of this project is to implement a system which will monitor the driver performance & will generate a report depending upon the driving skills of the driver. Rash driving detection will be based upon certain parameters such as speeding the car more than rated speed, accelerating the car more than its respective gear limits, taking inappropriate turns, etc. Also, if car is driven efficiently within certain limits, it will also in turn increase the car performance along with reduction in number of accidents.

1.2 OBJECTIVES

- Continuously monitor & measure car parameters using OBD II tool.
- Use Rasp Berry Pi with OBD device to detect rash driving related parameters by writing efficient algorithm in it.
- Notify the driver immediately if there is rash driving so that he/she will drive the car efficiently
- Creating a report using the stored value.
- Creating a bar graph from the error count.

1.3 PRESENT THEORIES AND PRACTICE

Currently, there is a device to detect rash driving on highways and to alert the traffic authorities in case of any speed violation. In one of the system, to detect rash driving the police has to use a handheld radar gun and aim at the vehicle to record its speed. If the speed of the vehicle exceeds the speed limit, nearest police station is informed to stop the speeding vehicle. This is an ineffective process as after detecting one has to inform the same and a lot of time is wasted.

Another proposed system will check on rash driving by calculating the speed of a vehicle using the time taken to travel between the two set points at a fixed distance. A set point consists of a pair of sensors comprising of an IR transmitter and an IR receiver, each of which are installed on either side of the road. The speed limit is set by the police who use the system depending upon the traffic at the very location.

The time taken by the vehicle to travel from one set point to the other is calculated by control circuit. Based on that time it then calculates the speed and displays that on seven segment displays. Moreover, if the vehicle crosses the speed limit, a buzzer sounds alerting the police. The proposed system fulfills that requirement as it consists of two blocks: transmitter and receiver; both use a microcontroller of the 8051 family and a rectified-power supply. This project consists of an RF transceiver module operating at 2.4 GHz.

This concept can be extended in future by integrating a camera with the system which could capture the image of the number plate of the vehicle which will be sent to the traffic authorities.

1.4 THEORETICAL ANAYSIS

In our project, we are going to use OBD (On Board Diagnostics) technology. This technology gives us the status of various vehicle subsystems such as torque measurement, RPM , speed ,acceleration, etc. There is a Bluetooth device which we can use to obtain all the information collected by OBD system on the android phone.

The OBD tool will read the various car parameters listed above. Out of these we have to select the parameters related to rash driving. And we will create the report based on all these parameters. For collecting parameters and creating the report we have to write efficient algorithm.

Our major task in the project is writing algorithm which will give us the report of driving skills or driving performance of the driver. To obtain this report, we are going to use the information collected by the OBD system.

CHAPTER 2

LITERATURE SURVEY

2.1 **Drunken driving and rash driving prevention system**

As is needless to say; a majority of accidents, which occur, are due to drunk driving. As such, there is no effective mechanism to prevent this. Here we have designed an integrated system for the same purpose. Alcohol content in the driver's body is detected by means of an infrared breath analyzer placed at the steering wheel. An infrared cell directs infrared energy through the sample and any unabsorbed energy at the other side is detected. The higher the concentration of ethanol, the more infrared absorption occurs (in much the same way that a sunglass lens absorbs visible light, alcohol absorbs infrared light). Thus the alcohol level of the driver is continuously monitored and calibrated on a scale. When it exceeds a particular limit the fuel supply is cutoff. If the device is removed also, the fuel supply will be automatically cut off or an alarm is sounded depending upon the requirement.

2.2 **Symbian Based Rash Driving Detection System**

Rash driving is a major cause of traffic accidents throughout the world. In this paper, we propose use of mobile phones as the platform for rash driving detection system development, as they offer platforms for detection and post detection communication. We intend to design a system aimed at early detection and alert of dangerous vehicle driving patterns related to rash driving. The entire implementation requires only a mobile phone placed in vehicle and with accelerometer. A program installed on a mobile phone computes accelerations based on sensor readings, and compares them with typical rash driving patterns extracted from real driving tests. Once any confirmation of rash driving is detected, the mobile phone will automatically alert the driver first and if same driving persists then call 100. We implement the detection system on symbian phone and have it tested with different kinds of driving behaviors.

2.3 **Detection of Over Speeding Vehicles on Highways**

This paper presents a device to detect rash driving on highways and to alert the traffic authorities in case of any violation. In past, lot of devices to detect rash driving on highways has been made. Most of the approaches require human concentration and involve a lot of effort, which is difficult to implement. In this paper we intend to design a system aimed at early detection and alert of dangerous vehicle driving patterns related to rash driving. The entire implementation requires an IR transmitter, an IR receiver, a control circuit and a buzzer. The speed limit is set by the police who use the system depending upon the traffic at the very location. The time taken by the vehicle to travel from one set point to the other is calculated by control circuit and displays that on seven segment displays. Moreover, if the vehicle crosses the speed limit, a buzzer sounds alerting the police.

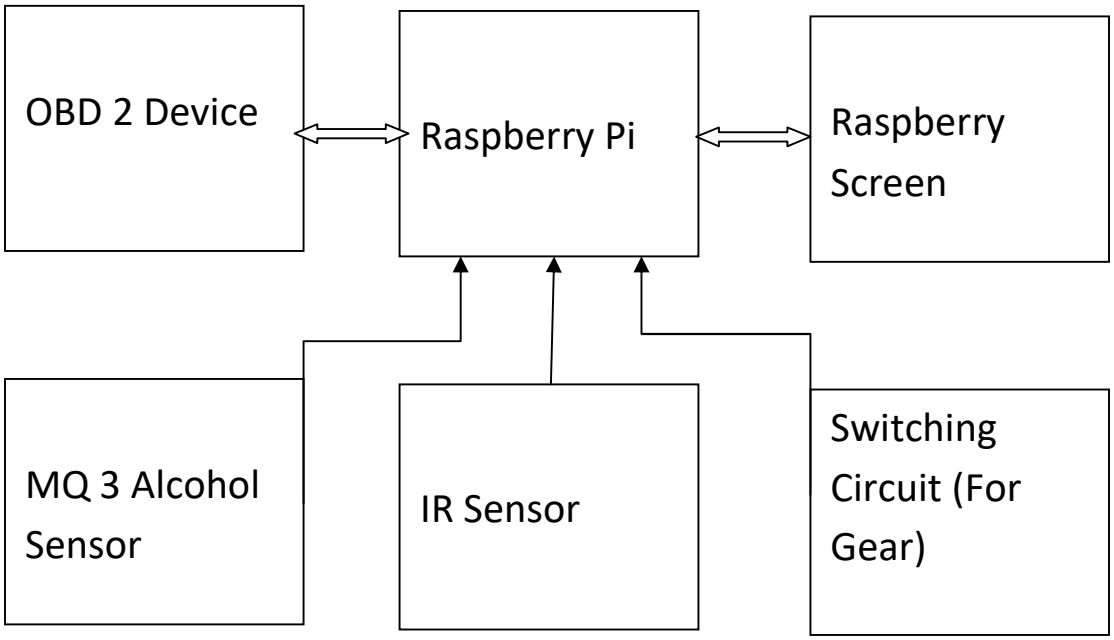
CHAPTER 3

PRESENT WORK

3.1 PROPOSED METHODOLOGY

1. Selecting appropriate OBD device.
2. Connecting OBD device to the car
3. Collecting information related to various car subsystem parameters from the OBD device
4. Getting the data on Raspberry Pi.
5. Writing efficient algorithm.
6. The algorithm should be such that it will collect the rash driving parameters.
7. Creating proper GUI.
8. It will generate a report highlighting the Rash driving information.
9. Transmitting the created report to the android phone of the car owner.

3.2 BLOCK DIAGRAM



DESCRIPTIION:

In our system, we are collecting various parameters continuously from car such as Speed, RPM, Coolant temperature, Alternator voltage through the OBD II device. The OBD II device is having Bluetooth connectivity. We are reading the above mentioned parameters from OBD II device on Raspberry Pi after establishing Bluetooth connection between the two devices.

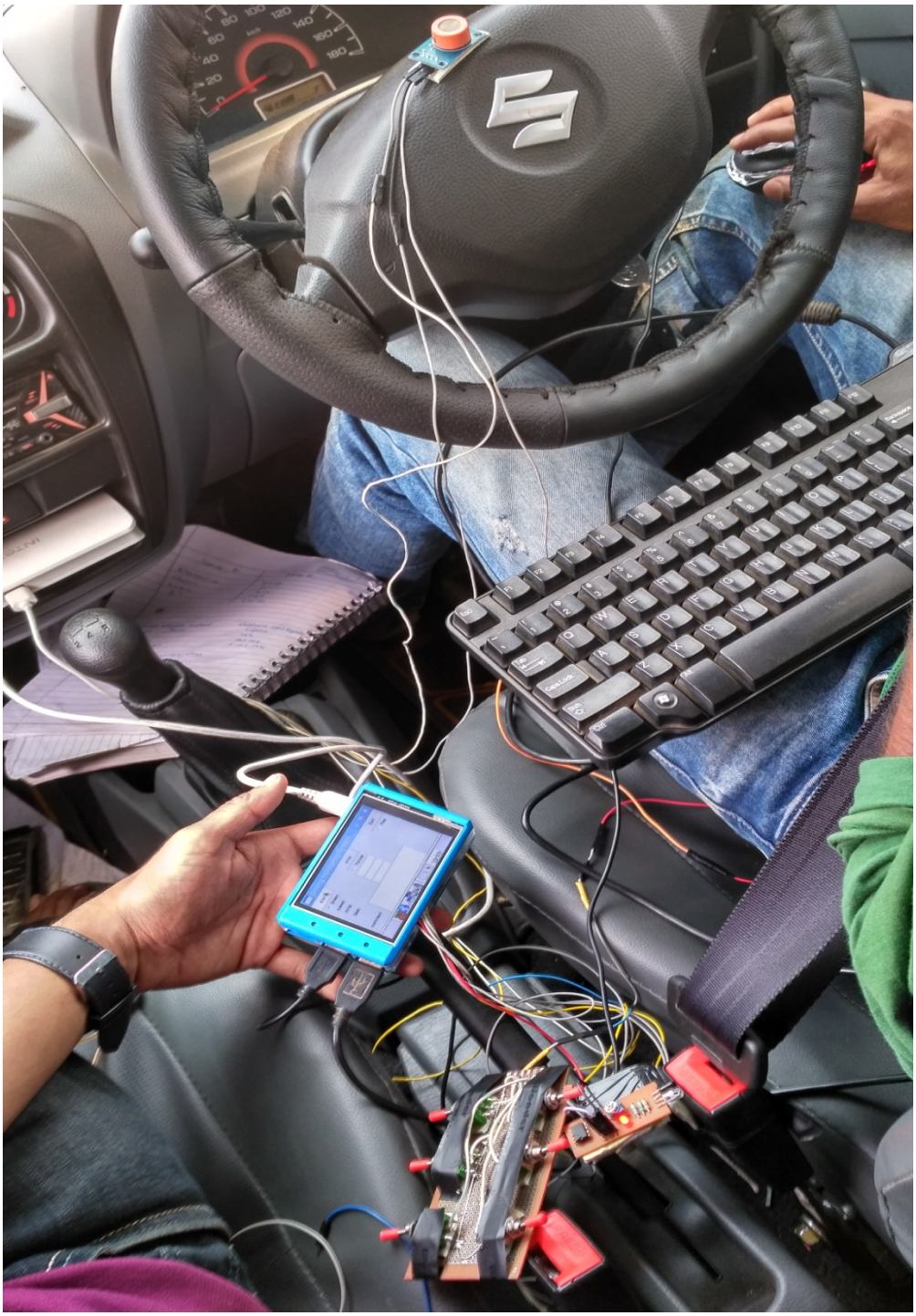
We are using IR sensor to detect whether the driver is applying the seat belt or not. If the seat belt is not applied, he will be informed to apply the seat belt on the Raspberry Pi screen.

We are using MQ3 Alcohol sensor to detect alcohol consumption by the driver.

We have designed a gear box module using switch based circuit through which we will be able to change the current gear as per the actual gear of the car.

A Raspberry Pi screen is used to display the GUI model & display various comments regarding driving performance.

3.3 COMPLETE HARDWARE SETUP



3.4 COMPONENTS DETAILS

3.4.1 Raspberry pi 3 model B

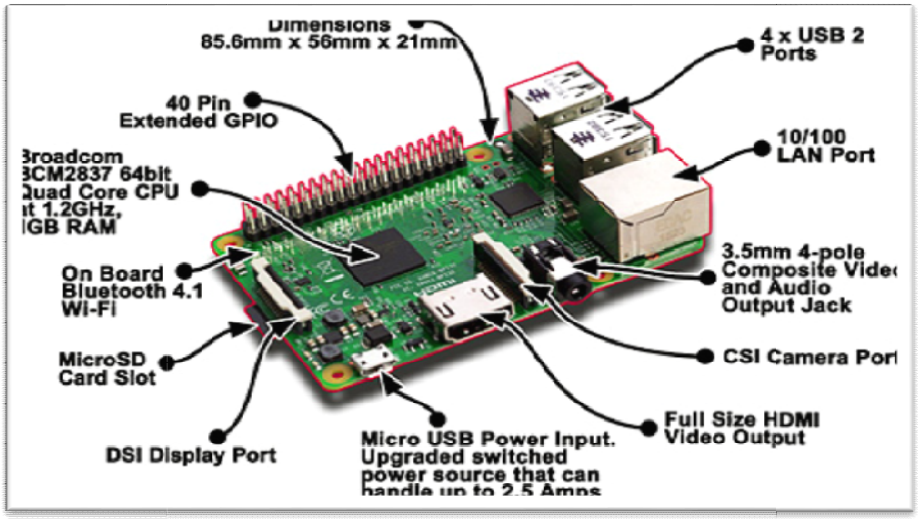


Fig. raspberry pi board layout

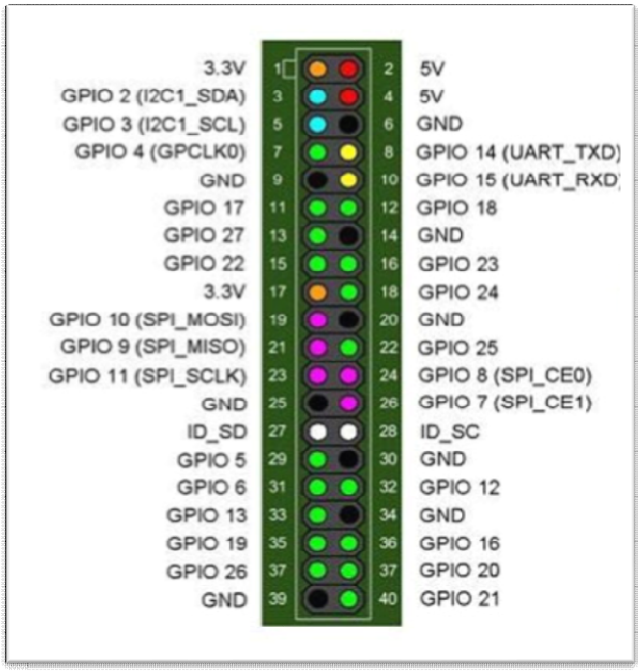


Fig. Pin Diagram

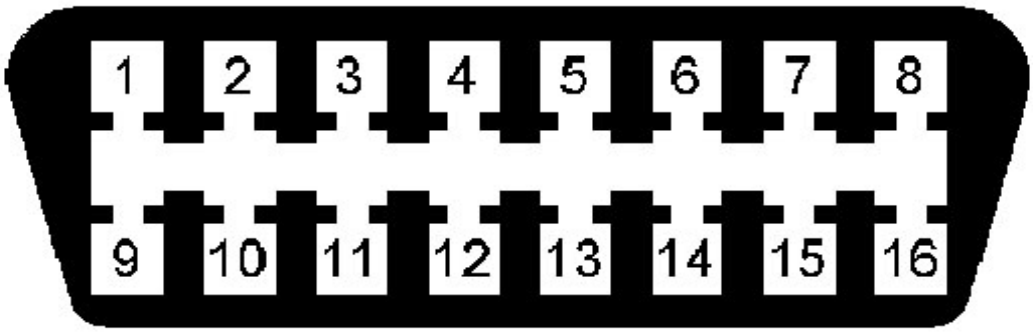
Features: -

Feature	Specification	Use
CPU	1.2GHz Quad-Core ARM Cortex-A53 Processor	
GPU	Broadcom BCM2387 chipset (SoC)	Graphics Processor
Memory	1 GB RAM at 900 MHz	Internal Memory
Video	HDMI, composite	Monitor Connection
Audio	HDMI, stereo analog	
USB	4 x USB2.0 (model B)	To connect various devices like keyboard, Mouse
Storage	SD card (8GB, Class 4)	Operating System and Data storage
Networking	10/100 Ethernet	Network / Internet

Feature	Specification	Use
Power	5V micro USB	Power supply for kit (2.5A)
Bluetooth	Bluetooth 4.1	Data transfer from various sensors
BLE	Bluetooth Low Energy	Data transfer from various sensors
CSI	Camera Serial Interface	Raspberry Pi camera interface
DSI	Display Serial Interface	LCD, Touch-screen connection

3.4.2 OBD II

Fig. OBD II



PIN	DESCRIPTION	PIN	DESCRIPTION
1	Vendor Option	9	Vendor Option
2	J1850 Bus +	10	j1850 BUS
3	Vendor Option	11	Vendor Option
4	Chassis Ground	12	Vendor Option
5	Signal Ground	13	Vendor Option
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Vendor Option	16	Battery Power

OBD-II Connector and Pinout

OBD-II Systems were mandated to help technicians diagnose and service the computerized engine management systems of modern vehicles.
OBD-II PIDs (On-board diagnostics Parameter IDs) are codes used to request data from a vehicle, used as a diagnostic tool.
Typically, an automotive technician will use PIDs with a scan tool connected to the vehicle's OBD-II connector.

- The technician enters the PID
- The scan tool sends it to the vehicle's controller–area network (CAN)-bus, VPW, PWM, ISO, KWP (After 2008, CAN only)
- A device on the bus recognizes the PID as one it is responsible for, and reports the value for that PID to the bus
- The scan tool reads the response, and displays it to the technician

There are 10 modes of operation for OBD-II standard SAE J1979.
For our application, we would require only 3 modes.
Mode 1 gives real time data, Mode 2 gives the freeze data And Mode 9 for vehicle identification number.
The parameters which we get from OBD and which are required for detection of rash and inefficient driving are

- Engine RPM.
- Vehicle speed
- Engine load.
- Throttle Position.
- Distance traveled with malfunction indicator lamp (MIL) on

OBD-II PIDs: -

OBD-II PIDs (On-board diagnostics Parameter IDs) are codes used to request data from a vehicle, used as a diagnostic tool.

SAE standard J/1939 defines many PIDs, but manufacturers also define many more PIDs specific to their vehicles. All light duty vehicles (i.e. less than 8,500 pounds) sold in North America since 1996, as well as medium duty vehicles (i.e. 8,500-14,000 pounds) beginning in 2005, and heavy duty vehicles (i.e. greater than 14,000 pounds) beginning in 2010, are required to support OBD-II diagnostics, using a standardized data link connector, and a subset of the SAE J/1979 defined PIDs (or SAE J/1939 as applicable for medium/heavy duty vehicles), primarily for state mandated emissions inspections.

Typically, an automotive technician will use PIDs with a scan tool connected to the vehicle's OBD-II connector.

The technician enters the PID

The scan tool sends it to the vehicle's controller–area network (CAN)-bus, VPW, PWM, ISO, KWP. (After 2008, CAN only)

A device on the bus recognizes the PID as one it is responsible for, and reports the value for that PID to the bus

The scan tool reads the response, and displays it to the technician.

Modes: -

There are 10 modes of operation described in the latest OBD-II standard SAE J1979. They are as follows:

Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non-CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN, only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

Standard PIDs: -

The table below shows the standard OBD-II PIDs as defined by SAE J1979. The expected response for each PID is given, along with information on how to translate the response into meaningful data. Again, not all vehicles will support all PIDs and there can be manufacturer-defined custom PIDs that are not defined in the OBD-II standard.

Note that modes 1 and 2 are basically identical, except that Mode 1 provides current information, whereas Mode 2 provides a snapshot of the same data taken at the point when the last diagnostic trouble code was set. The exceptions are PID 01, which is only available in Mode 1, and PID 02, which is only available in Mode 2. If Mode 2 PID 02 returns zero, then there is no snapshot and all other Mode 2 data is meaningless.

When using Bit-Encoded-Notation, quantities like C4 means bit 4 from data byte C. Each bit is numerated from 0 to 7, so 7 is the most significant bit and 0 is the least significant bit.

A							
A7	A6	A5	A4	A3	A2	A1	A0
B							
B7	B6	B5	B4	B3	B2	B1	B0

3.4.3 RPI LCD 3.2”

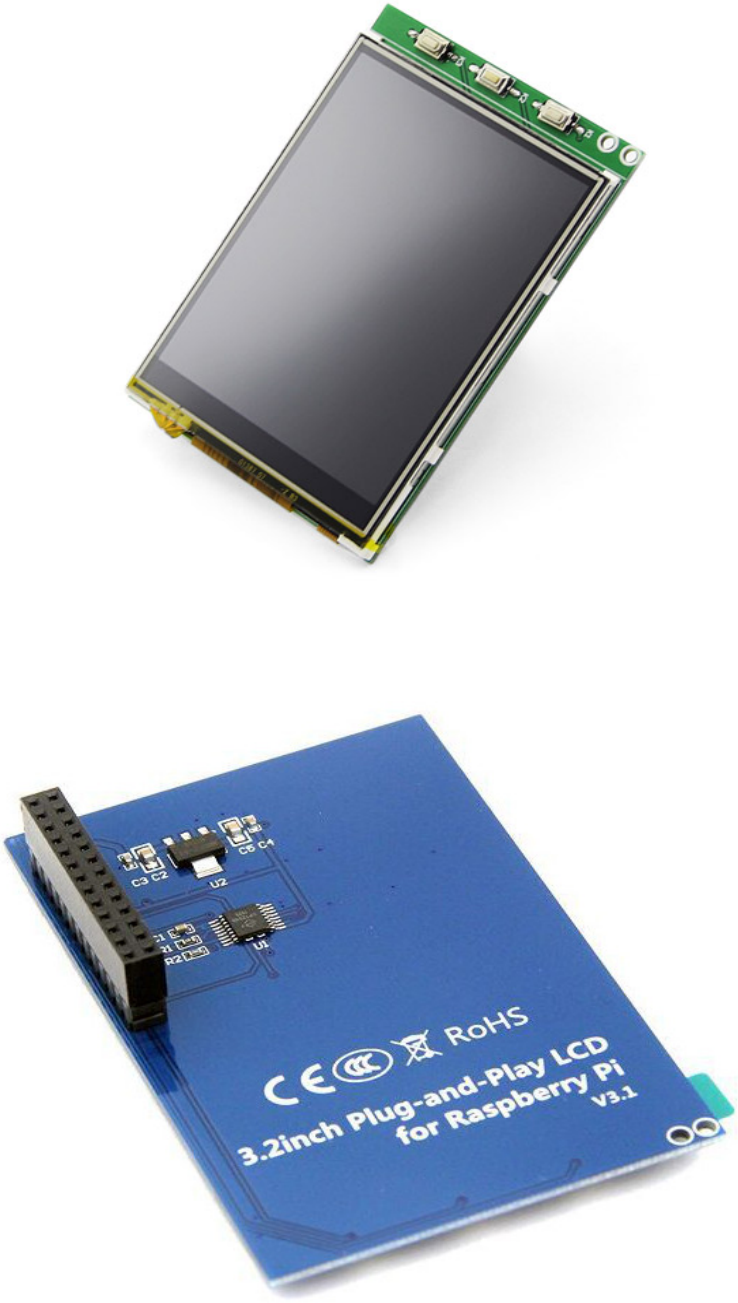


Fig. RPi LCD Screen

Features:

-Support access raspberry pie, easier to use
-Support Raspbian system that allows your system:
-Support for video playback (MP4 and other formats)
-Support for touch control camera (17 kinds of camera mode)
-Support soft keyboard (mouse and keyboard can also be controlled from the system)

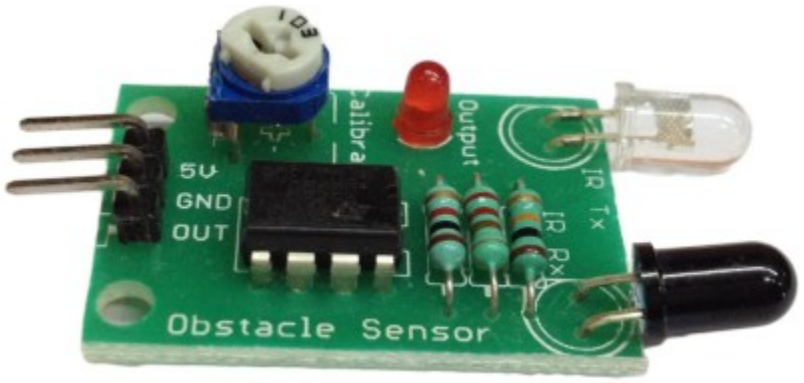
Specification:

Type TFT
SPI Interface
Touch panel control chip XPT2046
Levels Index 65536
LED backlight
Resolution of 320×240 (Pixel)
Size ratio 4:3

Interface definition		
Pin No.	Symbol	Description
1, 17	3.3V	Power supply + (3.3V power input)
2, 4	5V	Power supply + (5V power input)
3, 5, 7, 8, 10, 22	NC	NC
6, 9, 14, 20, 25	GND	Ground
11	TP_IRQ	Touch Panel interrupt, low level while the Touch Panel detects touching
12	KEY1	Button
13	RST	Reset
15	LCD_RS	LCD command control, command / data register select

16	KEY2	Button
18	KEY3	Button
19	LCD_SI/TP_SCK	LCD display / touch panel SPI data input
21	TP_SO	Touch panel SPI data input
23	LCD_SCK/TP_SCK	LCD display / touch panel SPI clock signal
24	LCD_CS	LCD chip select signals, low-level selection LCD
26	TP_CS	Touch panel chip select signal, choose low touch panel

3.4.4 IR SENSOR



Product Description: -

The sensor provides a digital output. The sensor outputs a logic one (+3.5 V) at the digital output when an object is placed in front of the sensor and logic zero (0 V), when there is no object in front of the sensor. An onboard LED is used to indicate the presence of an object. Operating Voltage 5 V Adjustable Range using preset (potentiometer on board).

Pin and Parameter Description: -

Dimensions- 3.8 x 2 x 0.5 cm AND Weight – 9 g

1. Operating Voltage -5V DC

2. Digital Output – logic one (+3.5V DC) logic zero (0V DC)

3. GROUND

USES: -

Obstacle Sensor
line follower ROBOTS
Visitor Counter Systems
Security System etc.

3.4.5 MQ 3 SENSOR

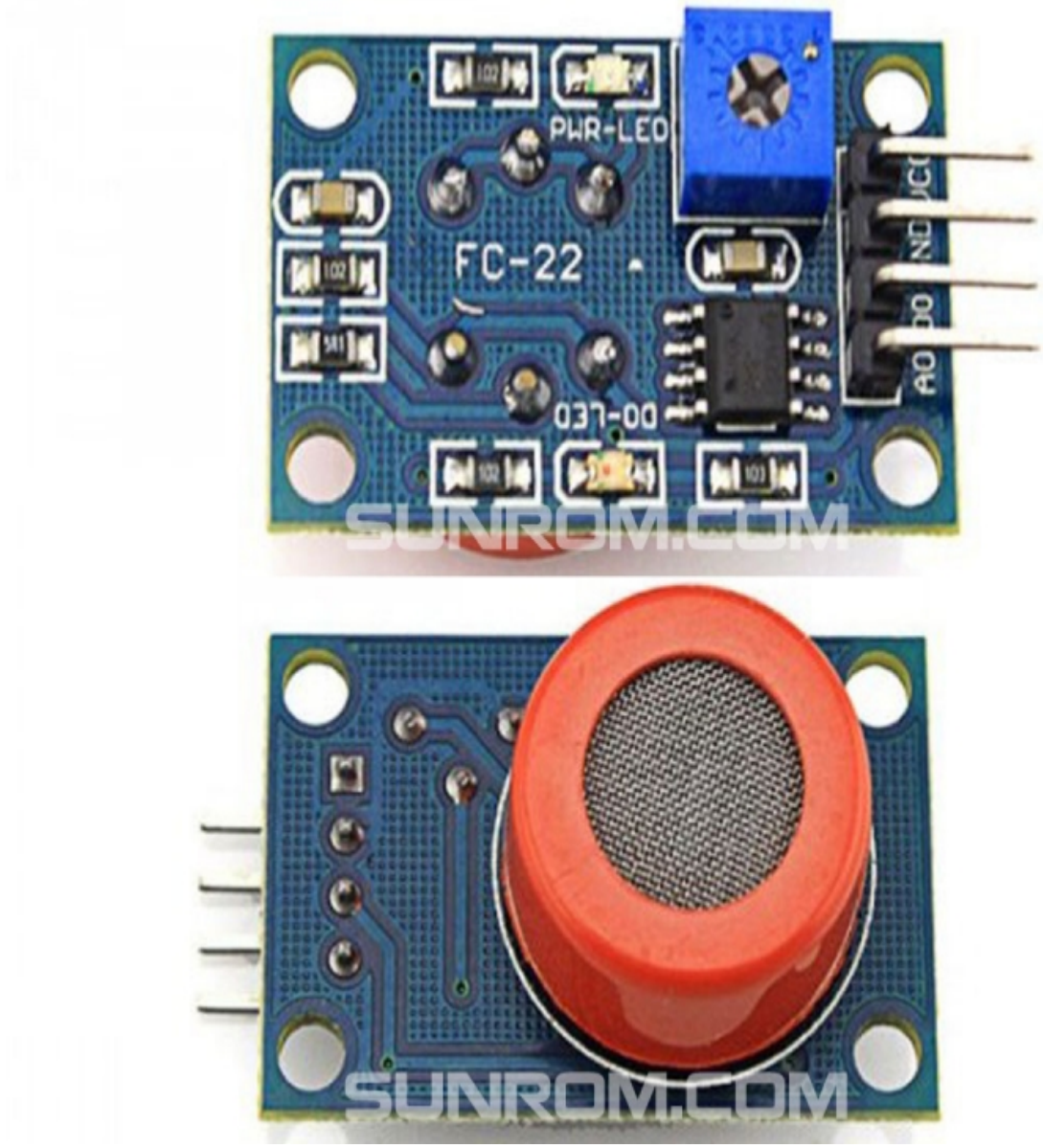


Fig. MQ 3 Sensor

Product Description: -

This module is made using Alcohol Gas Sensor MQ3. It is a low-cost semiconductor sensor which can detect the presence of alcohol gases at concentrations from 0.05 mg/L to 10 mg/L. The sensitive material used for this sensor is SnO2, whose conductivity is lower in clean air. Its conductivity increases as the concentration of alcohol gases increases. It has high sensitivity to alcohol and has a good resistance to disturbances due to smoke, vapor and gasoline. This module provides both digital and analog outputs. Threshold level for digital output can be easily adjusted using the preset on the board. MQ3 alcohol sensor module can be easily interfaced with Microcontrollers, Arduino Boards, Raspberry Pi etc.

Features: -

Low Cost
Fast Response
Stable and Long Life
Good Sensitivity to Alcohol Gas
Both Digital and Analog Outputs
On-board LED Indicator

Technical Data: -

Concentration: 0.05 mg/L ~ 10 mg/L Alcohol
Operating Voltage: 5V ±0.1
Current Consumption: 150mA
Operation Temperature: -10°C ~ 70°C

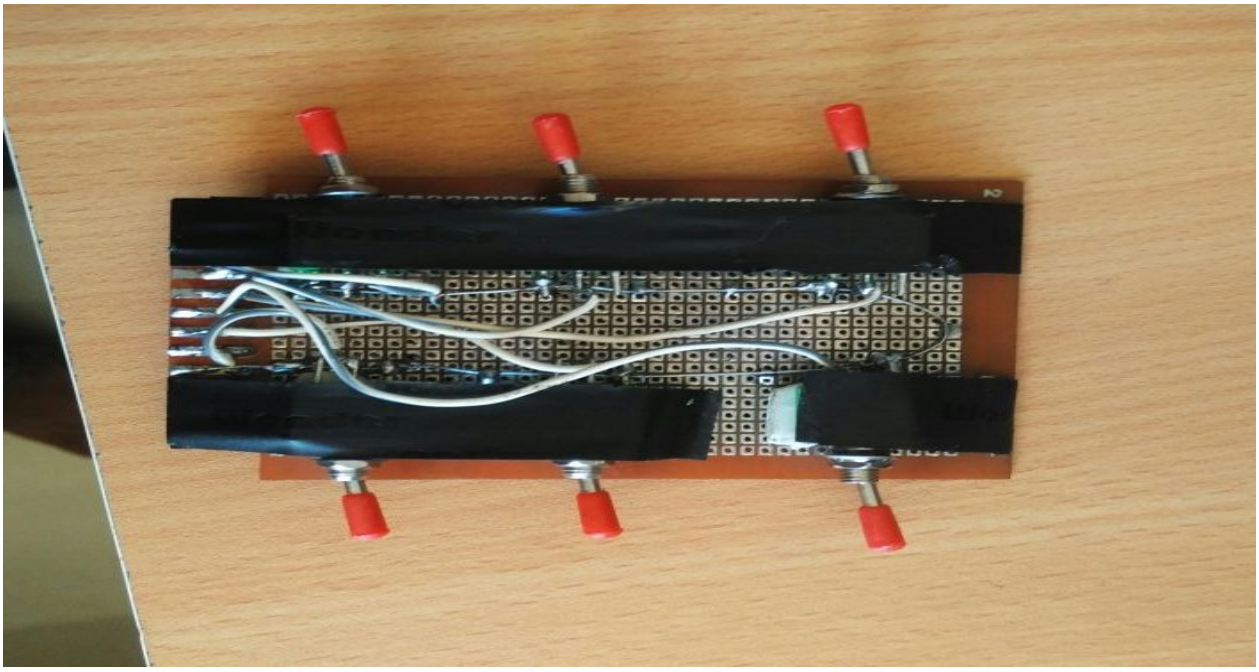
Pin Out: -

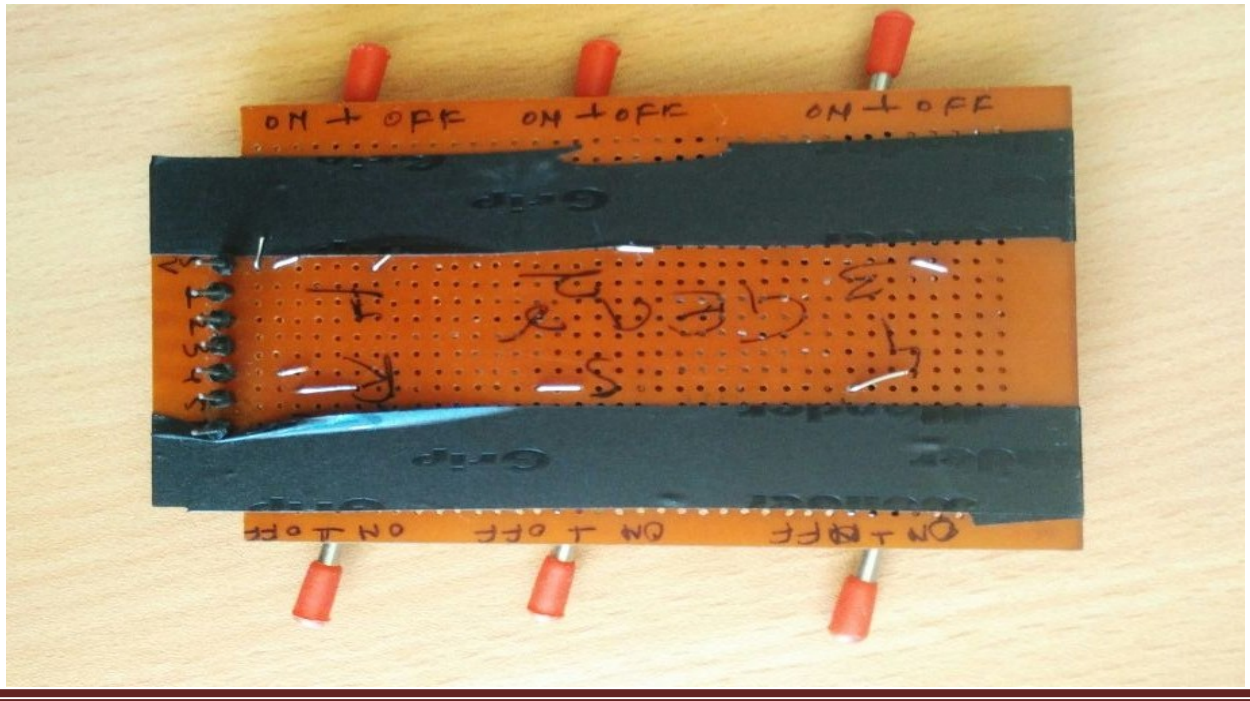
VCC – Input Power Supply
GND – Supply Ground
DO – Digital Output
AO – Analog Output

Applications: -

Vehicle Alcohol Detector
Portable Alcohol Detector

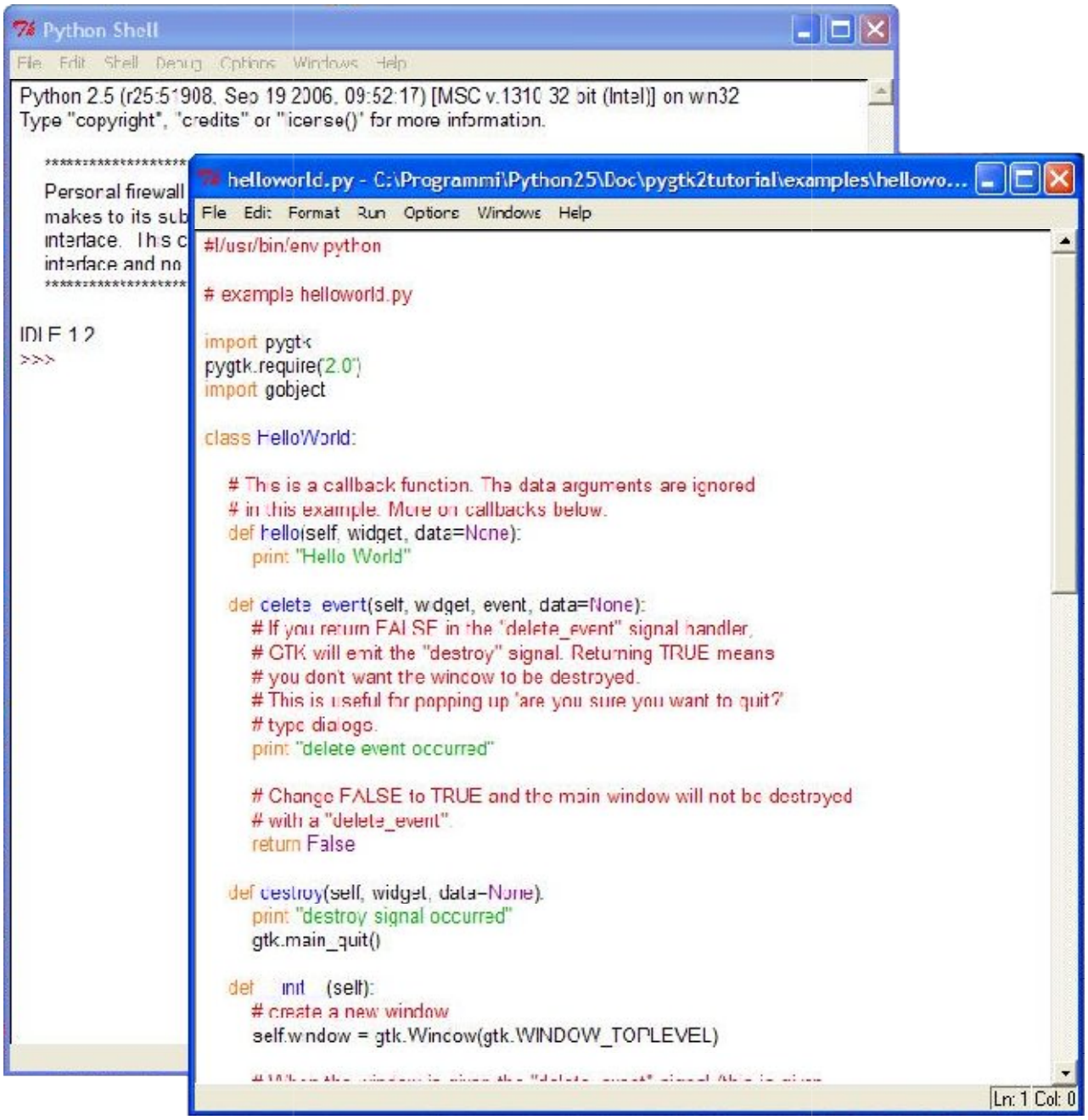
3.4.6 Switching circuit for gearing





3.5 SOFTWARE IMPLEMENTATION

3.5.1 TOOL USED



IDLE (Integrated Development Environment or Integrated Development and Learning Environment [3]) is an integrated development environment for Python, which has been bundled with the default implementation of the language since 1.5.2b1. It is packaged as an optional part of the Python packaging with many Linux distributions. It is completely written in Python and the Tkinter GUI toolkit (wrapper functions for Tcl/Tk).

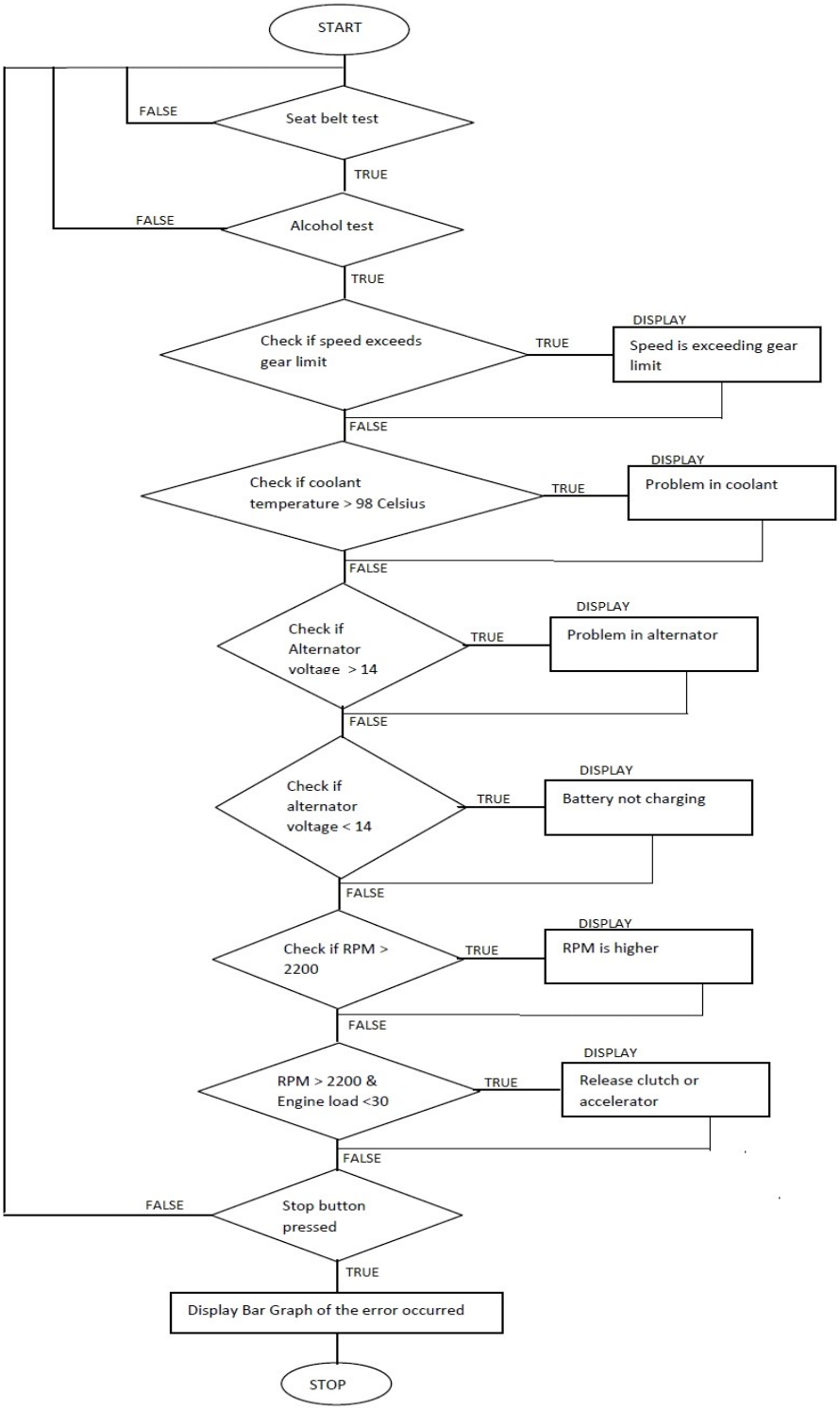
IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

According to the included README, its main features are:

Multi-window text editor with syntax highlighting, auto completion, smart indent and other.
Python shell with syntax highlighting.
Integrated debugger with stepping, persistent breakpoints, and call stack visibility.
IDLE has been criticized for various usability issues, including losing focus, lack of copying to clipboard feature, lack of line numbering options, and general user interface design; it has been called a "disposable" IDE, because users frequently move on to a more advanced IDE as they gain experience.

Author Guido van Rossum says IDLE stands for "Integrated development Environment". and since van Rossum named the language Python partly to honor British comedy group Monty Python, the name IDLE was probably also chosen partly to honor Eric Idle, one of Monty Python's founding members.

3.5.2 FLOW CHART



3.5.3 PROGRAM

```
import obd
from obd import OBDCommand
from obd.protocols import ECU
from obd.utils import bytes_to_int
from time import sleep
import xlwt
import subprocess
import datetime
import RPi.GPIO as GPIO
from Tkinter import *
import tkFont
import signal
import os

GPIO.setmode(GPIO.BOARD)
gear1=35
gear2=36
gear3=37
gear4=38
gear5=40
rev=33
rf=32
alco=31
ou=29
GPIO.setup(gear1,GPIO.IN)
GPIO.setup(gear2,GPIO.IN)
GPIO.setup(gear3,GPIO.IN)
GPIO.setup(gear4,GPIO.IN)
GPIO.setup(gear5,GPIO.IN)
GPIO.setup(rev,GPIO.IN)
GPIO.setup(rf,GPIO.IN)
GPIO.setup(alco,GPIO.IN)
GPIO.setup(ou,GPIO.OUT)



process = subprocess.call('sudo rfcomm connect 0 00:00:00:00:00:01 1&',shell = True)

c = obd.OBD()
```

```
def rpm(messages):
    d = messages[0].data
    v= bytes_to_int(d)/4.0
    return v

def speed(messages):
    d = messages[0].data
    v= bytes_to_int(d)
    return v

def engine_load(messages):
    d = messages[0].data
    v= bytes_to_int(d)/2.55
    return v

def engine_temp(messages):
    d = messages[0].data
    v= bytes_to_int(d)-40
    return v

def control_module_vtg(messages):
    d = messages[0].data
    v= bytes_to_int(d)/1000
    return v

def distance_w_mil(messages):
    d = messages[0].data
    v= bytes_to_int(d)
    return v

rp = OBDCommand("RPM","Engine RPM", b"010C",2,rpm,ECU.ENGINE,True)

sp = OBDCommand("SPEED","Engine SPEED", b"010d",1,speed,ECU.ENGINE,True)

tem = OBDCommand("COOLANT_TEMP","Engine COOLANT_TEMP",
b"0105",1,engine_temp,ECU.ENGINE,True)

el = OBDCommand("ENGINE_LOAD","Engine ENGINE_LOAD",
b"0104",1,engine_load,ECU.ENGINE,True)

alt_vtg = OBDCommand("CONTROL_MODULE_VOLTAGE","Engine
CONTROL_MODULE_VOLTAGE", b"0142",2,control_module_vtg,ECU.ENGINE,True)
```

```
dwm = OBDCommand("DISTANCE_W_MIL","Engine DISTANCE_W_MIL",
b"0121",2,distance_w_mil,ECU.ENGINE,True)
x=1



def handler(signum,frame):
    global sheet1
    global book
    global x
    global now
    if GPIO.input(rf)==0:
        print("connect your seatbelt")
        cb3.select()
        T.delete('1.0',END)
        T.insert(END,"apply seat belt")
        GPIO.output(ou, GPIO.LOW)
    else:
        if GPIO.input(alco)==0:
            print("Dont drive the car")
            cb3.select()
            T.delete('1.0',END)
            T.insert(END,"alcohol detected")
            GPIO.output(ou, GPIO.LOW)
        else:
            #print("alco not detected")
            GPIO.output(ou, GPIO.HIGH)
            cb3.deselect()
            cb2.deselect()
            cb1.select()
            global c
            global count1,count2,count3,count4,count5,count6
            T.delete('1.0',END)
            a=c.query(rp ,force = True)
            i=c.query(sp, force = True)
            aa=c.query(el, force = True)
            ab=c.query(tem, force = True)
            ac=c.query(alt_vtg, force = True)
            ad=c.query(dwm, force = True)
            sheet1.write(x,0,str(a))
            sheet1.write(x,1,str(i))
            sheet1.write(x,2,str(aa))
```

```
sheet1.write(x,3,str(ab))
sheet1.write(x,4,str(ac))
sheet1.write(x,5,str(ad))
#print 'rpm =',a.value
#print 'speed=',i.value
#print 'engine_load=',aa.value
#print 'temp=',ab.value
#print 'alt_vtg=',ac.value
#print 'dwm=',ad.value
T1.delete('1.0',END)
T1.insert(END,str(i))
T2.delete('1.0',END)
T2.insert(END,str(a))
#print x
if GPIO.input(gear1)==1:
   T3.delete('1.0',END)
   T3.insert(END,"1")
   sheet1.write(x,6,'Gear 1')
   if i.value > 15:
      T.insert(END,"Speed is exceeding change the gear\n")
      sheet1.write(x,12,"Speed is exceeding")
      count1=count1+1
      cb2.select()
elif GPIO.input(gear2)==1:
   T3.delete('1.0',END)
   T3.insert(END,"2")
   sheet1.write(x,6,'Gear 2')
   if i.value > 30:
      T.insert(END,"Speed is exceeding change the gear\n")
      sheet1.write(x,12,"Speed is exceeding")
      count1=count1+1
      cb2.select()
elif GPIO.input(gear3)==1:
   T3.delete('1.0',END)
   T3.insert(END,"3")
   sheet1.write(x,6,'Gear 3')
   if i.value > 40:
      T.insert(END,"Speed is exceeding change the gear\n")
      sheet1.write(x,12,"Speed is exceeding")
      count1=count1+1
      cb2.select()
elif GPIO.input(gear4)==1:
   T3.delete('1.0',END)
```

```
        T3.insert(END,"4")
        sheet1.write(x,6,'Gear 4')
        if i.value > 60:
            T.insert(END,"Speed is exceeding change the gear\n")
            sheet1.write(x,12,"Speed is exceeding")
            count1=count1+1
            cb2.select()
    elif GPIO.input(gear5)==1:
        T3.delete('1.0',END)
        T3.insert(END,"5")
        sheet1.write(x,6,'Gear 5')
        if i.value > 80:
            T.insert(END,"Speed is exceeding\n")
            sheet1.write(x,12,"Speed is exceeding")
            count1=count1+1
            cb2.select()
    elif GPIO.input(rev)==1:
        T3.delete('1.0',END)
        T3.insert(END,"R")
        sheet1.write(x,6,'Reverse Gear')
        if i.value > 30:
            T.insert(END,"Speed is exceeding\n")
            sheet1.write(x,12,"Speed is exceeding")
            count1=count1+1
            cb2.select()
    else:
        T3.delete('1.0',END)
        T3.insert(END,"N")
        sheet1.write(x,6,'Neutral')
    if a.value > 2200 :
    #print 'RPM IS HIGHER AND AFFECTING FUEL ECONOMY'
        T.insert(END,"RPM IS HIGHER AND AFFECTING FUEL ECONOMY\n")
        sheet1.write(x,11,'RPM IS HIGHER')
        count2=count2+1
        cb2.select()
    if aa.value < 30 :
        if a.value > 2200 :
        #print 'Release the clutch or accelerator pedal '
            T.insert(END,"Release the clutch or accelerator pedal\n")
            count3=count3+1
            sheet1.write(x,8,'release the cutch or acclerator')
            cb2.select()
    if ac.value > 14 :
```

```
            #print 'Alternator voltage is greater ,and it is affecting battery'
            T.insert(END,"Alternator voltage is greater ,and it is affecting battery\n")
            count4=count4+1
            cb3.select()
            sheet1.write(x,9,'Alternator voltage is greater')
        if ac.value < 14 :
            #print 'Alternator voltage is lower ,and it is not charging battery'
            T.insert(END,"Alternator voltage is lower ,and it is not charging battery\n")
            count5=count5+1
            sheet1.write(x,10,'Alternator voltage is lower')
            cb3.select()
        if ab.value > 98 :#& i.value == 0 :
            #print 'check the coolant and visit to service centre'
            T.insert(END,"Check the coolant and visit to service centre\n")
            count6=count6+1
            sheet1.write(x,7,'check the coolant')
            cb3.select()

        book.save("/home/pi/Desktop/program1/"+now+".xls")
        x=x+1



    signal.signal(signal.SIGALRM,handler)
    signal.setitimer(signal.ITIMER_REAL,0.001)



win = Tk()

myFont = tkFont.Font(family = 'Helvetica', size = 8, weight = 'bold')

def startprogram():
    global count1,count2,count3,count4,count5,count6,x
    global now
    global book
    global sheet1
    count1= 0
    count2= 0
    count3= 0
    count4= 0
    count5= 0
    count6= 0
```

```
    x=1
    now = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    book = xlwt.Workbook(encoding="utf-8")
    sheet1= book.add_sheet("sheet 1")
    sheet1.write(0,0,"RPM")
    sheet1.write(0,1,"SPEED")
    sheet1.write(0,2,"ENGINE LOAD")
    sheet1.write(0,3,"TEMP")
    sheet1.write(0,4,"ALTERNATOR VOLTAGE")
    sheet1.write(0,5,"MIL DIST")
    sheet1.write(0,6,"GEAR")
    sheet1.write(0,13,"gear error= ")
    sheet1.write(0,14,"RPM error= " )
    sheet1.write(0,15,"Clutch error= ")
    sheet1.write(0,16,"Alternator voltage grater= ")
    sheet1.write(0,17,"battery not charging= ")
    sheet1.write(0,18,"coolant temp error= ")
    if GPIO.input(rf)==1:
        if GPIO.input(alco)==1:
            signal.signal(signal.SIGALRM,handler)
            signal.setitimer(signal.ITIMER_REAL,0.01)
        else :
            T.delete('1.0',END)
            T.insert(END,'alcohol is detected\n')
            GPIO.output(ou, GPIO.LOW)
            cb3.select()
    else:
        T.delete('1.0',END)
        T.insert(END,'Apply the seat belt\n')
        GPIO.output(ou, GPIO.LOW)
        cb3.select()


def exitProgram():
        win.destroy()


def stopProgram():
    global count1,count2,count3,count4,count5,count6
    signal.setitimer(signal.ITIMER_REAL,0)
    print ('gear error=',count1)
    print ('RPM error=',count2)
    print ('clutch error=',count3)
```

```
print ('alternator voltage greater=',count4)
print ('battery not charging=',count5)
print ('coolant temp=',count6)
sheet1.write(1,13, str(count1))
sheet1.write(1,14, str(count2))
sheet1.write(1,15, str(count3))
sheet1.write(1,16, str(count4))
sheet1.write(1,17, str(count5))
sheet1.write(1,18, str(count6))
book.save("/home/pi/Desktop/program1/"+now+".xls")

root = Tk()
root.title("Bar Graph")
count4 = count4 + count5
z_width = 320
z_height = 180
z = Canvas(root, width=z_width, height=z_height)
z.pack()

if ((count1 > 400) | (count2 > 400) | (count3 > 400) | (count4 > 400) | (count6 > 400) ):
    q=6
    z.create_text(20 , 127, text="200")
    z.create_text(20 , 93, text="400")
    z.create_text(20 , 60, text="600")
    z.create_text(20 , 27, text="800")
    z.create_text(20 , 10, text="Count")
else:
    q=3
    z.create_text(20 , 127, text="100")
    z.create_text(20 , 93, text="200")
    z.create_text(20 , 60, text="300")
    z.create_text(20 , 27, text="400")
    z.create_text(20 , 10, text="Count")

speed1 = 160-(count1/q)
rpm1 = 160-(count2/q)
C_T1 = 160-(count6/q)
Alt_Vtg1 = 160-(count4/q)
Clutch1 = 160-(count3/q)


z.create_rectangle(60, speed1, 90, 160, fill="red")
z.create_text(75, 168, text="speed")
```

```
    z.create_rectangle(100, rpm1, 130, 160, fill="blue")
    z.create_text(115, 168, text="rpm")
    z.create_rectangle(140, C_T1, 170, 160, fill="green")
    z.create_text(155, 168, text="C_T")
    z.create_rectangle(180, Alt_Vtg1, 210, 160, fill="orange")
    z.create_text(195, 168, text="Alt_vtg")
    z.create_rectangle(220, Clutch1, 250, 160, fill="yellow")
    z.create_text(255, 168, text="clutch error")
    z.create_line(0, 160, 500, 160)
    z.create_line(40, 0, 40, 500)

    z.create_text(75, (speed1-10), text= count1)
    z.create_text(115, (rpm1-10), text= count2)
    z.create_text(155, (C_T1-10), text= count6)
    z.create_text(195, (Alt_Vtg1-10), text= count4)
    z.create_text(235, (Clutch1-10), text= count3)

    root.after(30000, lambda: root.destroy())
    root.mainloop()



win.title("Driving Monitoring System")
win.geometry('320x240')

exitButton  = Button(win, text = "Exit", font = myFont, command = exitProgram, height =1 , width
= 6)
exitButton.grid(row =1 , column = 2)

startButton  = Button(win, text = "start", font = myFont, command = startprogram, height =1 ,
width = 6)
startButton.grid(row =1 , column = 0)

stopButton  = Button(win, text = "stop", font = myFont, command = stopProgram, height =1 ,
width = 6)
stopButton.grid(row =1 , column = 1)

l=Label(win,text="comment",font=myFont)
l.grid(row = 1000, column = 0)
T= Text(win,height=5, width= 30,font=myFont)
T.grid(row = 1000 , column = 1)

l1=Label(win,text="speed",font=myFont)
```

```
l1.grid(row = 6,column = 0)
T1= Text(win,height=1, width= 10,font=myFont)
T1.grid(row =6,column = 1)

l2=Label(win,text="RPM",font=myFont)
l2.grid(row = 7,column = 0)
T2= Text(win,height=1, width= 10,font=myFont)
T2.grid(row =7,column = 1)

l3=Label(win,text="Gear",font=myFont)
l3.grid(row = 8,column = 0)
T3= Text(win,height=1, width= 10,font=myFont)
T3.grid(row =8,column = 1)

cb1 = Radiobutton(win,text="Green",value=1,font=myFont,width=6)
cb1.grid(row = 3, column=0)
cb2 = Radiobutton(win,text="Yellow",value=2,font=myFont,width=6)
cb2.grid(row = 3, column=1)
cb3 = Radiobutton(win,text="Red",value=3,font=myFont,width=6)
cb3.grid(row = 3, column=2)
cb1.select()


win.mainloop()
```

CHAPTER 4

RESULT DISCUSSION

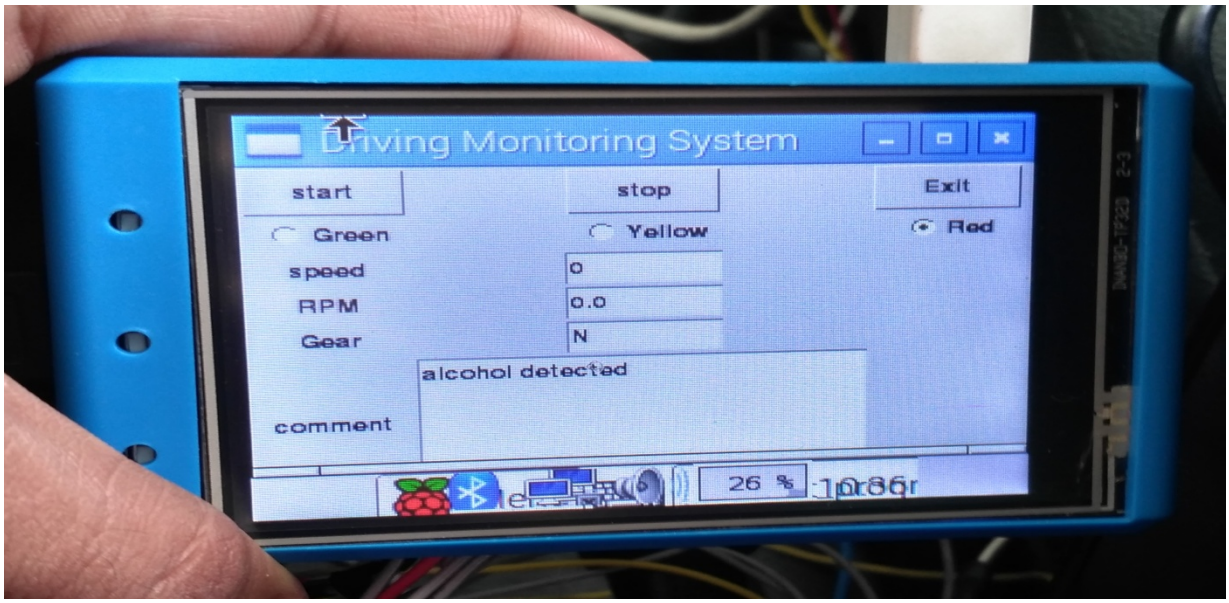
4.1 RESULT

4.1.1 Excel Sheet Snap

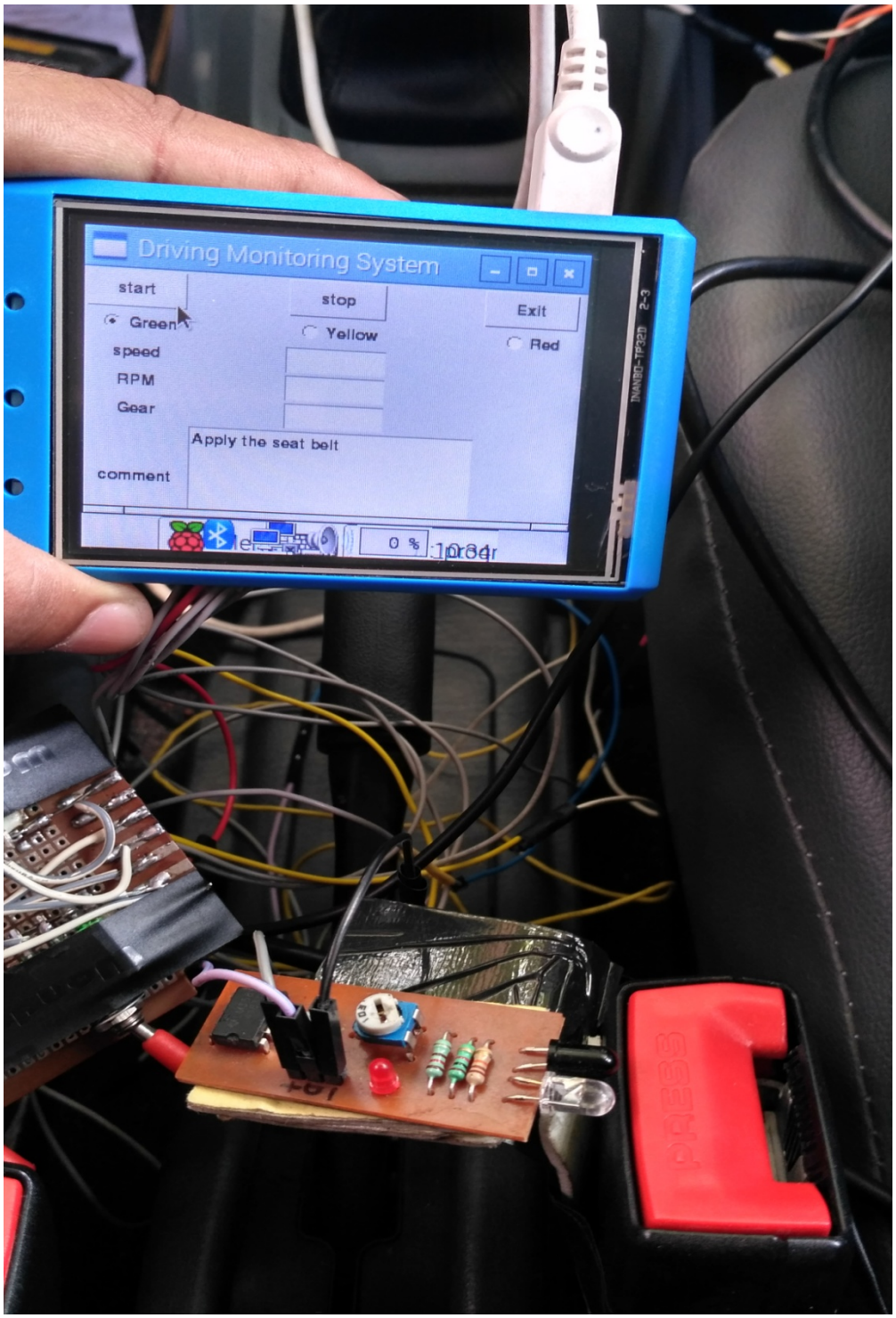
We are generating excel sheet which will be based on driver's performance throughout the journey. The excel sheet will be saved with the name having Date and time of that day. It will contain the complete details of the driver's performance including the comments.

RPM	SPEED	ENGINE LOAD	TEMP	ALTERNATOR VOLTAGE	MIL DIST	GEAR
964.75	0	28.6274509804	78	12	0	Gear 1
926.0	0	29.4117647059	77	13	0	Gear 1
889.5	0	30.5882352941	77	13	0	Gear 1
862.0	0	31.3725490196	77	13	0	Gear 1
847.0	0	32.5490196078	77	13	0	Gear 1
837.0	0	32.9411764706	77	13	0	Gear 1
838.25	0	33.3333333333	77	13	0	Gear 1
840.0	0	33.7254901961	77	13	0	Gear 1
838.75	0	34.1176470588	77	13	0	Gear 1
839.5	0	34.5098039216	77	13	0	Gear 1
838.25	0	34.9019607843	77	13	0	Gear 1
834.75	0	35.2941176471	77	13	0	Gear 1
831.25	0	35.2941176471	77	13	0	Gear 1
824.0	0	35.6862745098	77	13	0	Gear 1
832.25	0	36.0784313725	77	13	0	Gear 1
828.0	0	36.4705882353	77	13	0	Gear 1
852.25	0	37.6470588235	77	13	0	Gear 1
927.0	0	35.6862745098	77	13	0	Gear 1
948.5	0	33.3333333333	77	14	0	Gear 1
943.5	0	32.1568627451	77	13	0	Gear 1
917.75	0	32.1568627451	77	13	0	Gear 1
892.0	0	32.5490196078	77	13	0	Gear 1
896.5	0	32.9411764706	77	14	0	Reverse Gear
897.5	0	33.3333333333	77	14	0	Reverse Gear
955.5	0	38.8235294118	77	14	0	Reverse Gear
1196.25	0	38.8235294118	77	13	0	Reverse Gear
1455.75	0	31.3725490196	77	13	0	Reverse Gear
1623.5	0	27.4509803922	77	13	0	Reverse Gear
1729.5	0	23.5294117647	77	13	0	Reverse Gear
1761.0	0	21.568627451	77	13	0	Reverse Gear
1744.5	0	20.7843137255	77	13	0	Reverse Gear
1710.75	0	20.7843137255	78	13	0	Reverse Gear
1652.0	0	20.3921568627	78	14	0	Reverse Gear
1589.5	0	21.1764705882	78	14	0	Reverse Gear

4.1.2 Alcohol Detecting Test



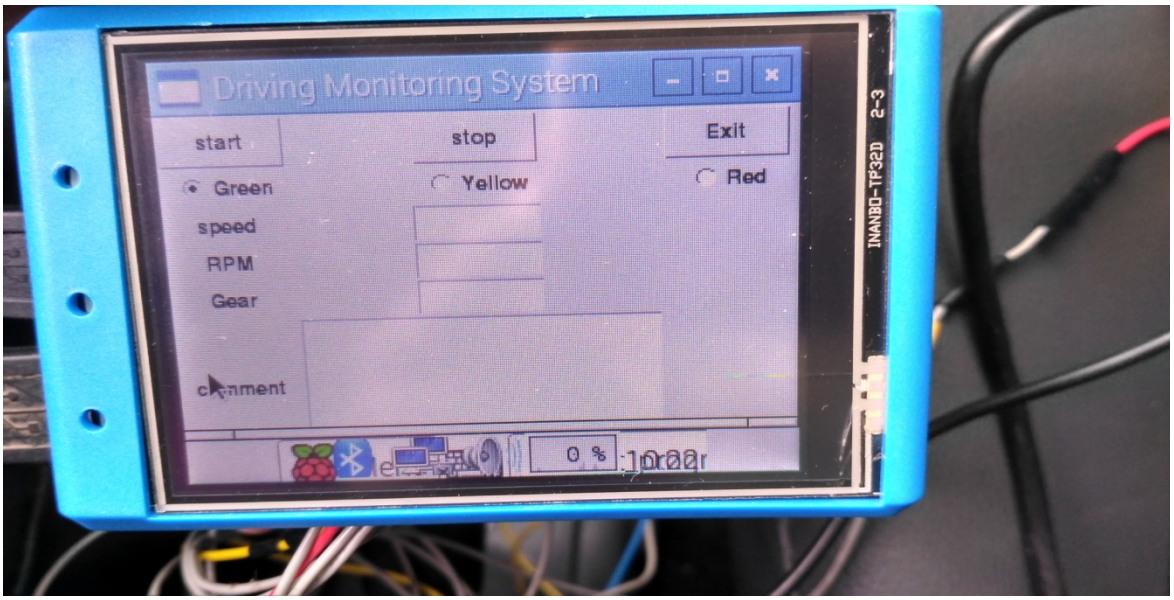
4.1.3 Seat Belt Test

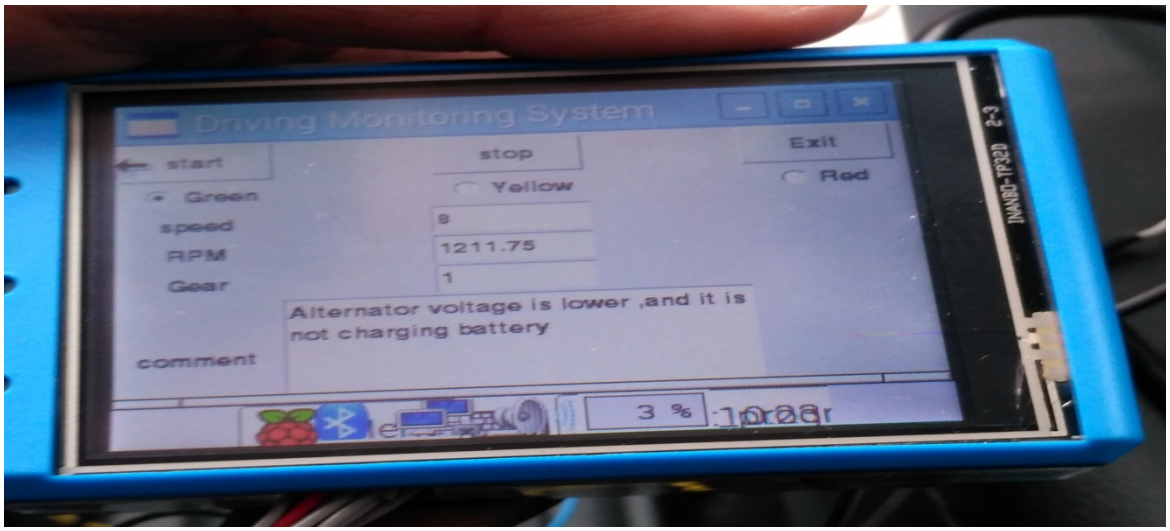


4.1.4 Continues Monitoring Snap

We have designed the GUI which consists of START, STOP & EXIT buttons on it. We are also continuously displaying Speed, RPM & Gear on our GUI.

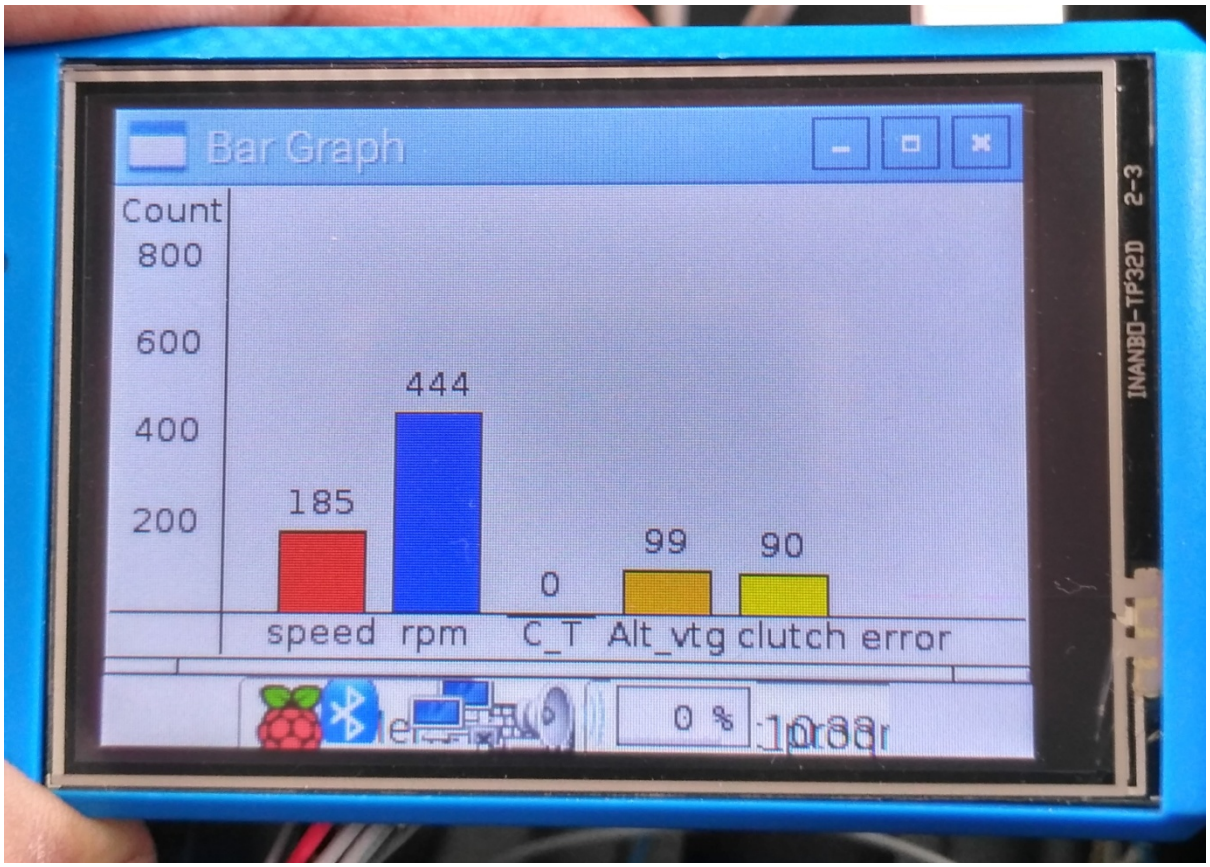
It is also having three indicators viz., Green, Yellow, and Red. Whenever driver is driving the car safely without crossing any threshold limits, Green indicator will be ON. If the driver will exceed speed limit of the gear, or when the engine load is less than 30 & RPM is greater than 2200, then Yellow indicator will be ON. If alcohol is sensed by alcohol sensor & seatbelt is not connected, then Red indicator will be ON.





4.1.5 Bar Graph

At the end of the journey, a bar graph will be displayed including error counts for Speed, RPM, Coolant temperature, Alternator voltage and Clutch error. It will indicate the number of times for which driver exceeds the respective limits. The graph will be displayed for the interval of 30 seconds.



CHAPTER 5

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION:

- Thus, we have designed a system which continuously monitors, measures and saves car parameters & notifies the driver immediately if there is inefficient driving so that he/she will drive the car efficiently & at the end a bar graph indicating mistakes made by the driver is displayed, which will be helpful to analyze the performance of the driver and increase the safety of the driver..

5.2 FUTURE SCOPE:

- Currently, we are using external gear circuit, later it can be replaced by Hall Effect sensor for detection of gear.
- The excel sheet which is generated as a report can be transmitted via server to the cab company.
- Steering angle and speed breaker can be detected using accelerometer.
- When alcohol is detected and seatbelt is not applied then the car's engine can be turned off.

References

1. Electronic instrumentation and measurement (English) 2nd edition (paperback, David a. Bell)
2. OBD reference: http://smogcheck.ca.gov/pdf/Appendix_J_5.22.14.pdf
   On-board diagnostics - Wikipedia
   https://en.wikipedia.org/wiki/On-board_diagnostics
3. TinyBASIC for Raspberry Pi - Raspberry Pi
   https://www.raspberrypi.org/blog/tinybasic-for-raspberry-pi/
4. Python – Programming Language
   pythonprogramminglanguage.com
5. https://www.tutorialspoint.com/python/python_gui_programming.html
6. http://python-obd.readthedocs.io/en/latest/
   python-OBD.html
7. http://python-obd.readthedocs.io/en/latest/Connections/
   OBD Connections - python-OBD.html
   http://python-obd.readthedocs.io/en/latest/Responses/
   Responses - python-OBD.html
9. http://python-obd.readthedocs.io/en/latest/Custom%20Commands/
   Custom Commands - python-OBD.html
10. MQ3 – datasheet
11. https://learn.sparkfun.com/tutorials/raspberry-gpio