VEHICLE SERVICE MANAGEMENT AND LIVE MONITORING SYSTEM

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

by

XXXX (17BECXXXX)

Under the Guidance of

DR. YYYYY

SCHOOL OF ELECTRONICS ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

JANUARY 2021

CERTIFICATE

This is to certify that the Capstone Project work titled "**VEHICLE SERVICE MANAGEMENT AND LIVE MONITORING SYSTEM**" that is being submitted by **XXXX (17BECXXXX)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. YYYY
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR **DEAN**
B. Tech. ECE School Of Electronics Engineering

ACKNOWLEDGEMENTS

ABSTRACT

In this paper an efficient system is presented which can automatically manage complete servicing process and at the same time monitor the changes and operations done on the automobile. Regular Automobile Service Management is required to keep a check on various parts of the automobile to ensure proper functioning and efficient working of the automobile. Automobile service Management and monitoring system constantly checks the state of the automobile after frequent intervals. It warns through android application about any improper values received and suggests that servicing is now required. When the automobile is at any of the service centers, previous details of automobile service can be shared with garage on authentication by the customer using the same android application. Sensor interfaced with raspberry Pi constantly monitors different subsystems of the automobile. It also provides customers with live video of garage when their automobile is in service centers**.** Regular servicing of automobile can make the automobile run smoothly and also fuel utilization is done efficiently. Live Monitoring will ensure that the customer is not cheated and only those parts which actually require servicing or repairing are looked upon by the garage people.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1
INTRODUCTION

Automobiles have become an important part in daily life. Figure 1 shows the increase in the sales of automobiles over years. Majority of the people use automobile for their day to day movements. But in many cases automobile servicing is neglected. Regular servicing of automobile can make the automobile run smoothly and also fuel utilization is done efficiently. Problem is also there due to the fact that when automobiles are taken for servicing or maintenance, it is assumed that repairs will be done in a prudent manner, but it is observed that each year a number of automobile accidents take place due to the negligence of repair service station and automobile dealer's negligence.

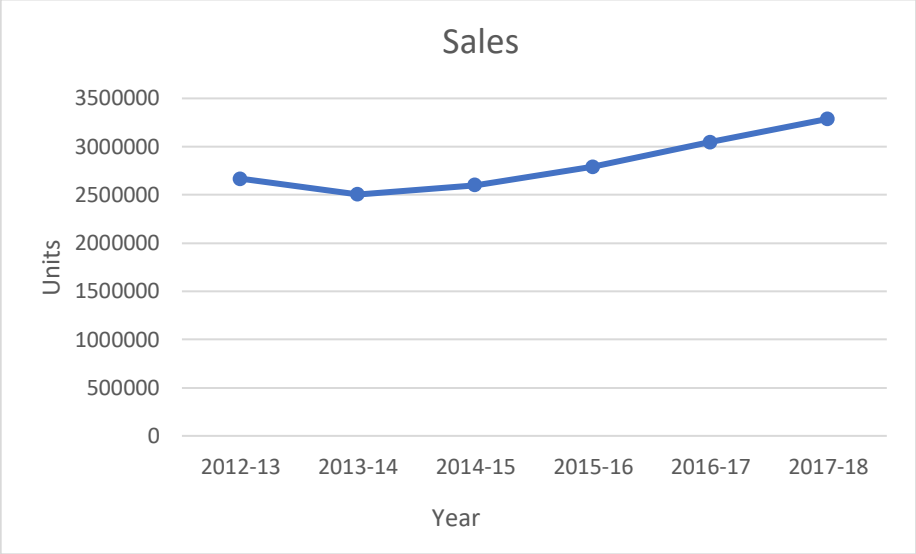

Figure 1 Automobile Sales

The issue of trust is paramount when automobile is given for servicing. There are many cases where customers are being cheated in the name of servicing. Removal of original parts and replacement with old ones is a major problem. It is also difficult for customers to find out if what

they are paying for is being done by the service providers. Service centers take the advantage of this condition and charge more than the actual bill.

A common practice of 'periodic automobile maintenance' [1] is followed in which the automobile is supposed to undergo periodic service and maintenance. When the automobile needs to get serviced, is commonly chosen by either a predefined time span or distance covered by the automobile. By and large, it is encouraged to get the automobile serviced every half year or after travelling 10000 kilometers.

But the issue with 'periodic automobile maintenance' is that no one is certain which part truly needs to be serviced or replaced which sometimes tends to parts in good condition being serviced/replaced, where our system will tell specific component which needs repair or servicing.

1.1 Objectives

The following are the objectives of this project:

- To design an efficient system which can automatically manage complete servicing process and at the same time monitor the changes and operations done on the car.
- Car service Management and monitoring system constantly checks the state of the car after frequent intervals. It warns through android application about any improper values received and suggests that servicing is now required.
- When the car is at any of the service centres, previous details of car service can be shared with garage on authentication by the customer using the same android application.
- Sensor interfaced with raspberry Pi constantly monitors different subsystems of the car.
- To provide customers with live video of garage where their car is in service centers.

1.2 Background and Literature Survey

A similar project on the ground of IOT (Internet of Things) was carried out by Rohit Dhall, Enterprise Architect, HCL Technologies, Noida, and Vijender Solanki, Research Scholar, Anna University, Chennai, India. This is our base paper; the title of our base paper is "An IoT Based Predictive Connected Car Maintenance Approach". It discussed a case, which can be implemented by the automobile industry, using technological advancements in the areas of IoT and Analytics. It used the concept of connected cars. 'Connected Car' is a terminology, often associated with cars and other passenger vehicles, which are capable of internet connectivity and sharing of various kinds of data with backend applications. The data being shared can be about the location and speed of the car, status of various parts/lubricants of the car, and if the car needs urgent service or not. Once data are transmitted to the backend services, various workflows can be created to take necessary actions, e.g. scheduling a service with the car service provider, or if large numbers of care are in the same location, then the traffic management system can take necessary action. 'Connected cars' can also communicate with each other, and can send alerts to each other in certain scenarios like possible crash etc.

We also referred to a paper titled "Automobile Service Center Management System" by Prof. Shilpa Chavan. This paper talks about mobile application for 'Automobile Service Center Management System'. This application is an android app which can be run on any android mobile phones. The app will enable any car user to search and communicate with any car service center in the vicinity. The user can find the service center, get its location and check and select any of the services provided by the respective service center. The user can send request for pick and drop, appointment for servicing, test drive as well as accessories purchase to the dealer. The dealer processes these requests and gives a response back to the user through push messages. This app also enables the user to set alarms for next servicing date, payment of insurance installment, etc.

Developing an application was also a learning process for us, we referred to a paper titled "MIT App Inventor-Enabling personal mobile computing" by Shaileen Crawford Pokress and Jose Juan Dominguez Veiga. This paper discusses that MIT App Inventor1 is a drag-and-drop visual programming tool for designing and building fully functional mobile apps for Android. App

Inventor promotes a new era of personal mobile computing in which people are empowered to design, create, and use personally meaningful mobile technology solutions for their daily lives, in endlessly unique situations. App Inventor's intuitive programming metaphor and incremental development capabilities allow the developer to focus on the logic for programming an app rather than the syntax of the coding language, fostering digital literacy for all.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 gives the cost involved in the implementation of the project.
- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report.
- Chapter 6 consists of codes.
- Chapter 7 gives references.

CHAPTER 2

VEHICLE SERVICE MANAGEMENT AND LIVE MONITORING SYSTEM

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

The following block diagram (figure 2) shows the system architecture of this project.

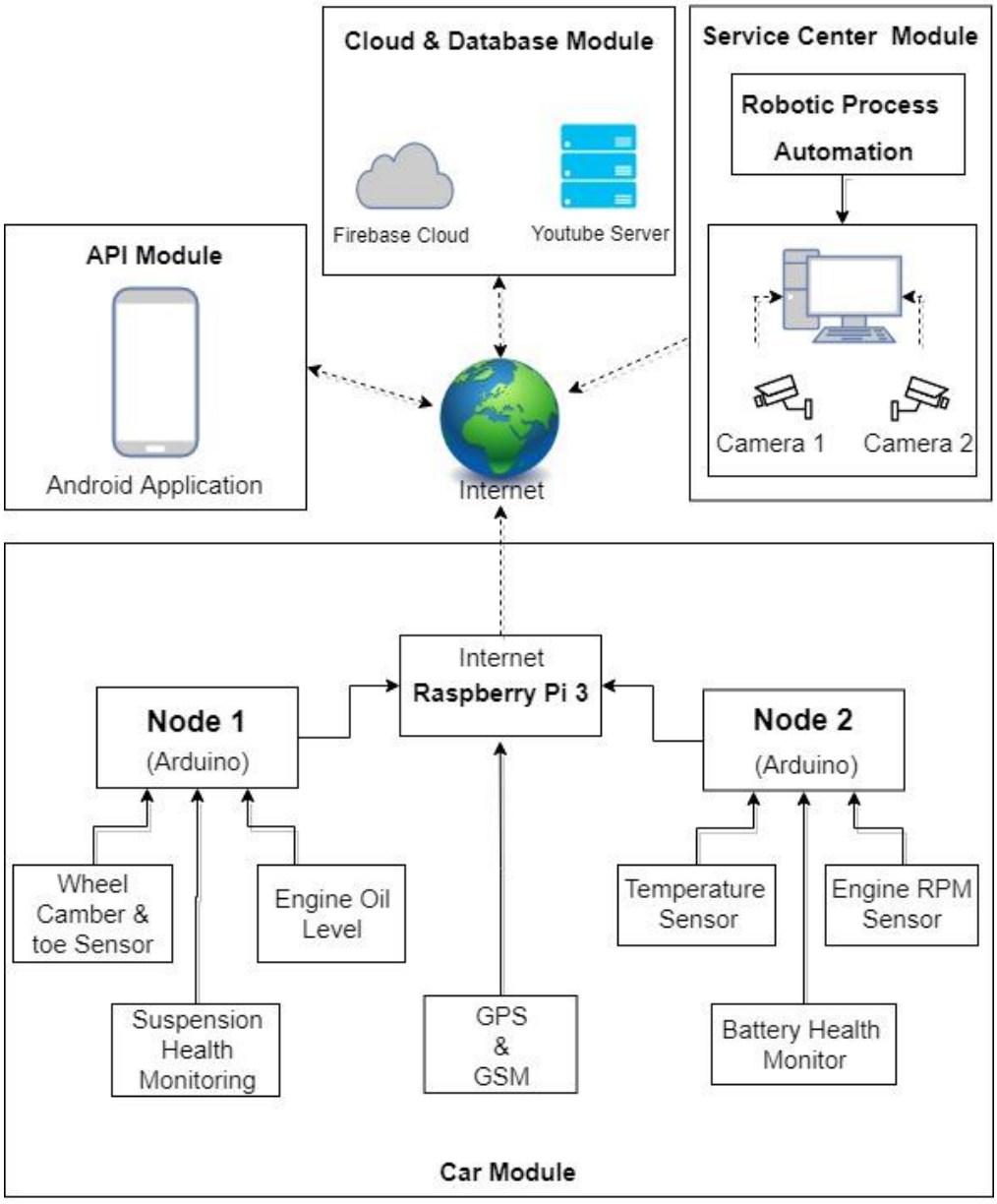

Figure 2 System Block Diagram

2.2 Working Methodology

The system has two sections, hardware and software. Hardware consists of Raspberry pi 3 which is connected to several Arduino which act as nodes. Arduino is connected to several sensors which constantly monitors different subsystems of the automobile.

Engine RPM is monitored using hall sensor. Alignment of wheels (Camber and toe) is monitored using Accelerometer and gyro sensor. Position of automobile is constantly monitored using GPS. Location of automobile can be checked anytime by customers using the application. Dedicated temperature sensors are used for monitoring temperature of engine. An oil level sensor is used for monitoring level of oil in engine. Brake failure can be very dangerous, a dedicated pressure sensor in brake lines is used to detect the pressure of fluid. Battery health is monitored using battery level indicator circuit. User is warned through application. If the automobile is taken out of garage due to any reason, customers will be notified through application. For this GPS system is used.

2.3 Standards

Various standards used in this project are:

- **SSL secured connection using firebase**

  The Secure Sockets Layer (SSL)—now technically known as Transport Layer Security (TLS)—is a common building block for encrypted communications between clients and servers. In a typical SSL usage scenario, a server is configured with a certificate containing a public key as well as a matching private key. As part of the handshake between an SSL client and server, the server proves it has the private key by signing its certificate with public-key cryptography.

- **Secure Shell (SSH)**

  It is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line login and remote command execution, but any network service can be secured with SSH. SSH provides a secure channel over an unsecured network in a client–server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two

major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. We are using it for PuTTy as SSH Client here in our project for connection with Raspberry Pi.

- **GSM**

  GSM is a standard for mobile phones. The ubiquity of the GSM standard makes international roaming very common with "roaming agreements" between operators. GSM differs significantly from its predecessors, in that both signalling, and speech channels are digital, which means that it is seen as a second generation (2G) mobile phone system. GSM is an open standard which is developed by the 3GPP. GSM has retained backward compatibility with original GSM phones. At the same time, the GSM standard continues to develop, and packet data capabilities were added in the Release '97 version of the standard with GPRS. Higher-speed data transmission has been introduced by providing a new modulation scheme with EDGE.

2.4 System Details

This section describes the software and hardware details of the system:

2.4.1 Software Details

YouTube, android app, firebase and robotic process automation are used.

i) Android Application

The android application is built on the platform called **MIT App Inventor [3].** App Inventor for Android is an open-source web application. It allows newcomers to create apps for the Android operating system. It uses a graphical interface, which allows users to drag-and-drop visual objects to create an app for android phones.

Developing Mobile Application

- Make an account on http://ai2.appinventor.mit.edu/(MIT App Inventor)

- In App Inventor Designer, design the App's User Interface by arranging both on- and off-screen components by adding different screens as needed on the application.

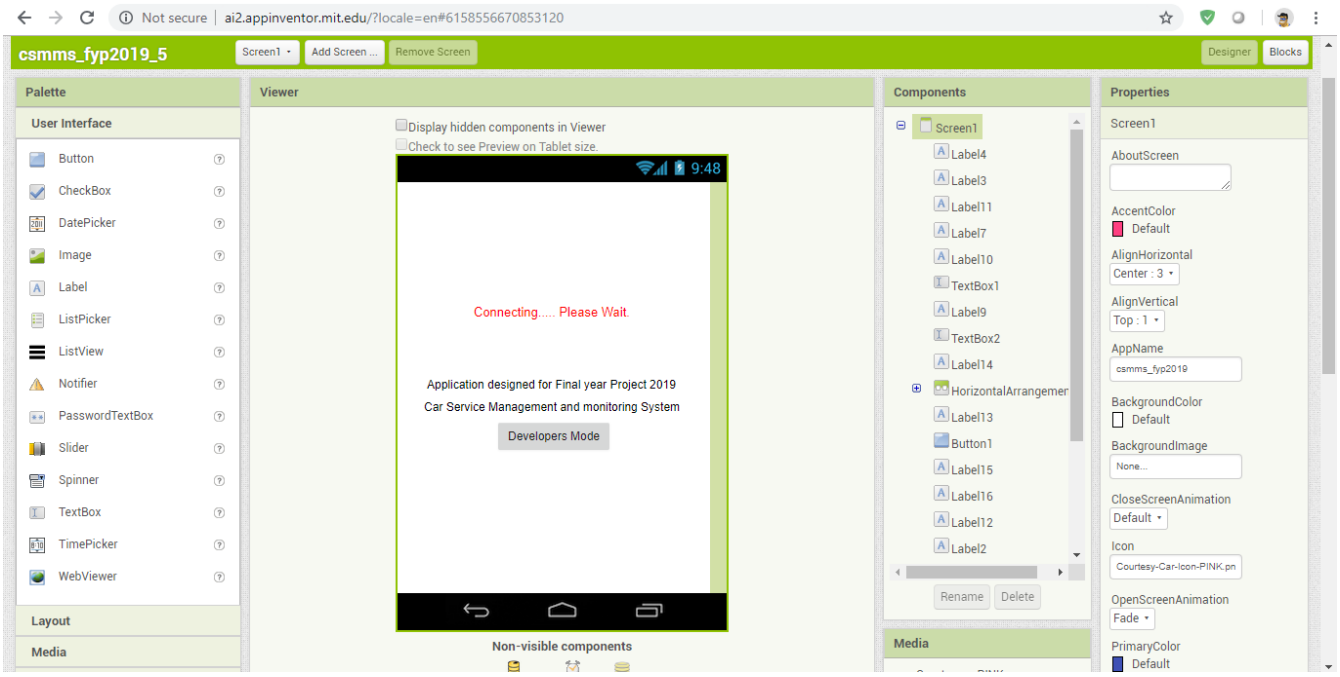

Figure 3 Screen 1

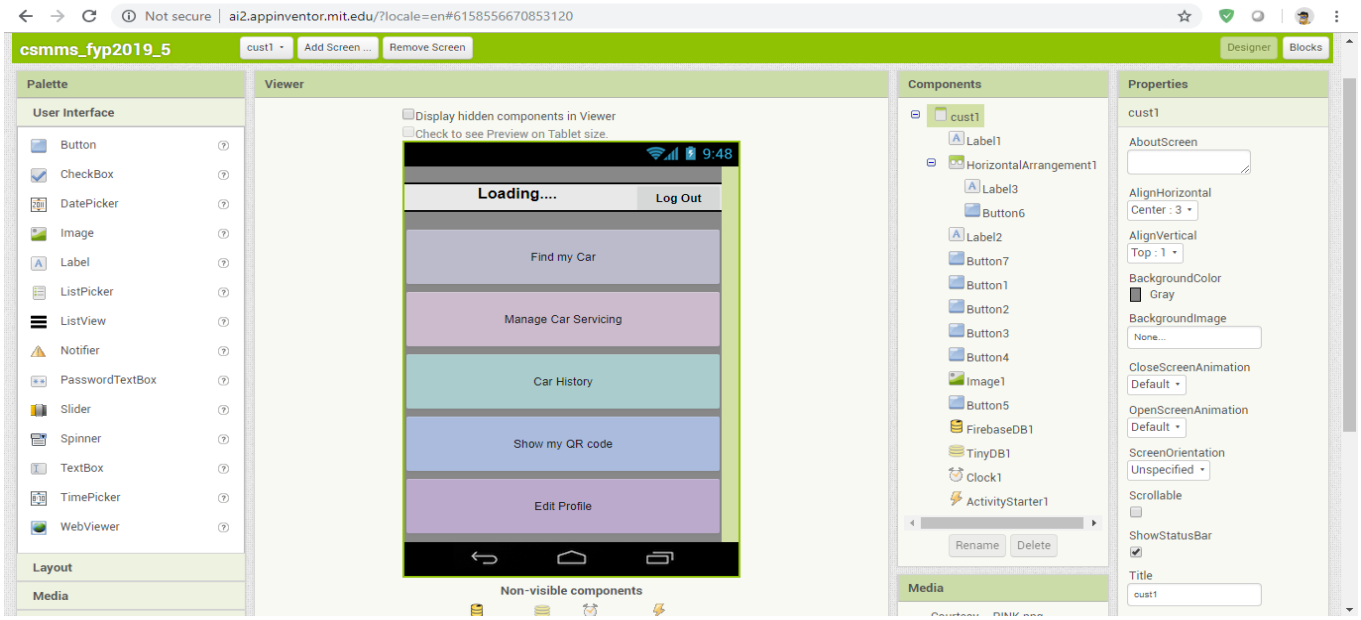

Figure 4 Screen 2

- Likewise add screens according to the features of the application.
- The App Inventor Blocks Editor, where you assemble program blocks that specify how the components should behave.

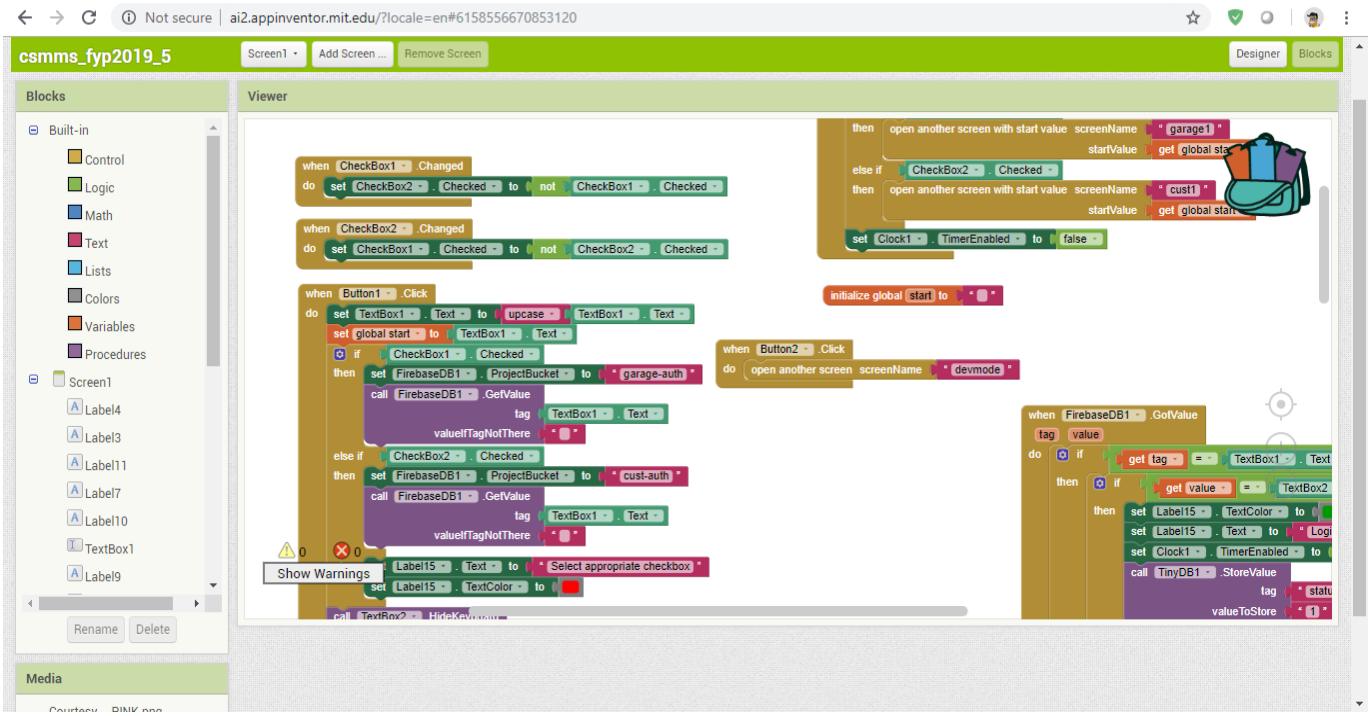

Figure 5 Block Editor for Screen 1

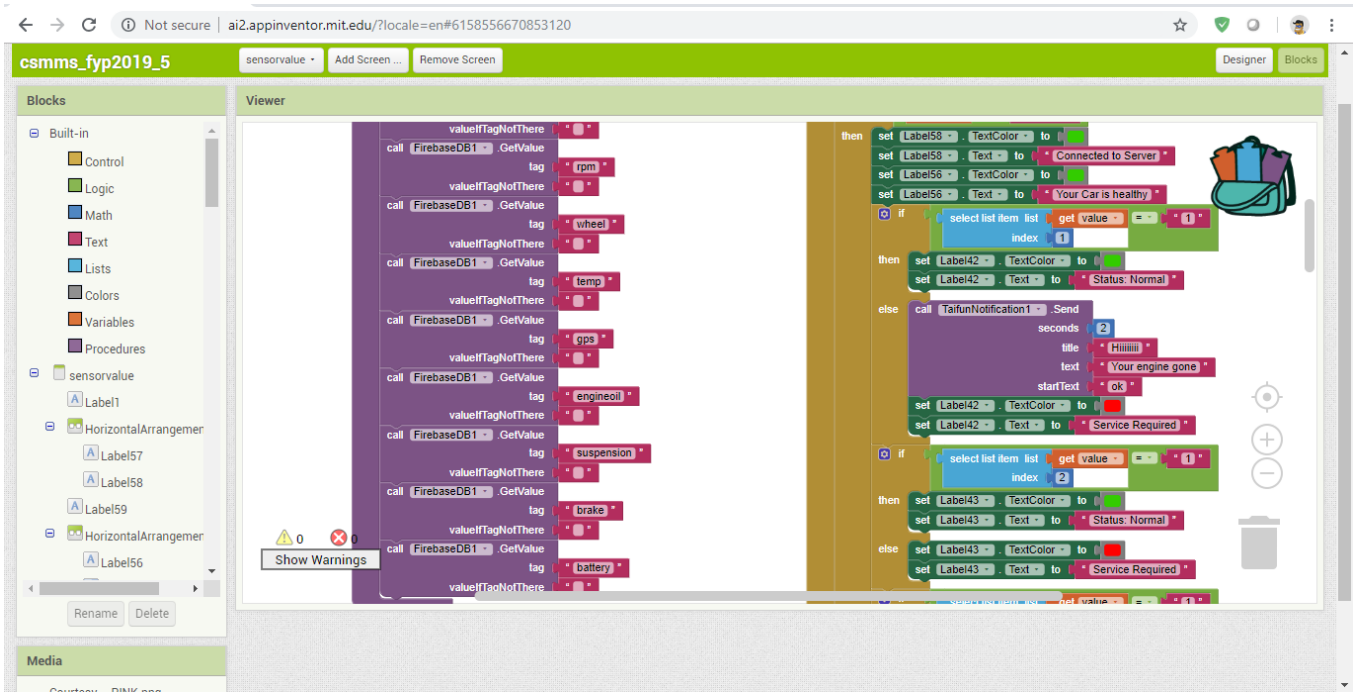

Figure 6 Block Editor for Screen 2

As shown in figure 7 is home page of the application, to log in one should have proper ID and password otherwise user will not able to log in. There are two options here garage and customer.

Garage owner who provides the services to the automobiles should select garage and if one is an automobile owner, then customer should be selected.

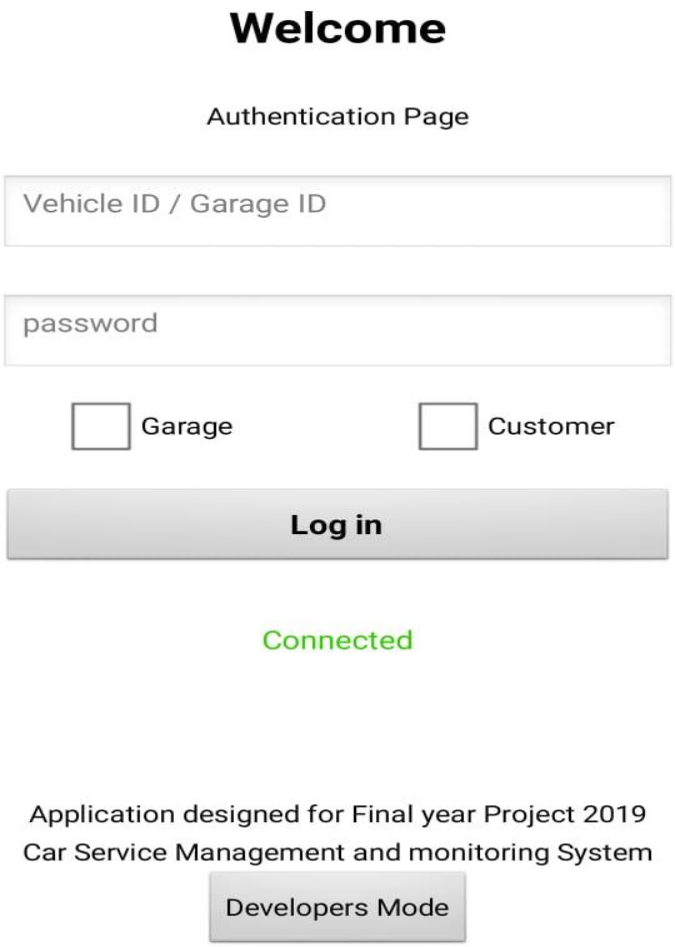

Figure 7 Home Page

Figure 8 Main Page

After successfully logging in using proper ID and password customer will be taken to the main page as shown in Figure 8. The values from the sensors will be displayed in this section, like if one clicks on ***Find my Automobile,*** it will take to the maps section where customer will get the shortest path to the automobile.

The next section ***Manage Automobile Service*** where the values from the sensor will be displayed as shown in the Figure 9.a and 9.b

As seen all the values from the sensor are being displayed in this section and if the values differ from the reference values then the “service required” in red text will be displayed under the part which requires the service. For example, here in the figure 9(a) battery voltage differs from the standard value so, service is required.

Figure 9 (a)Manage Automobile Service

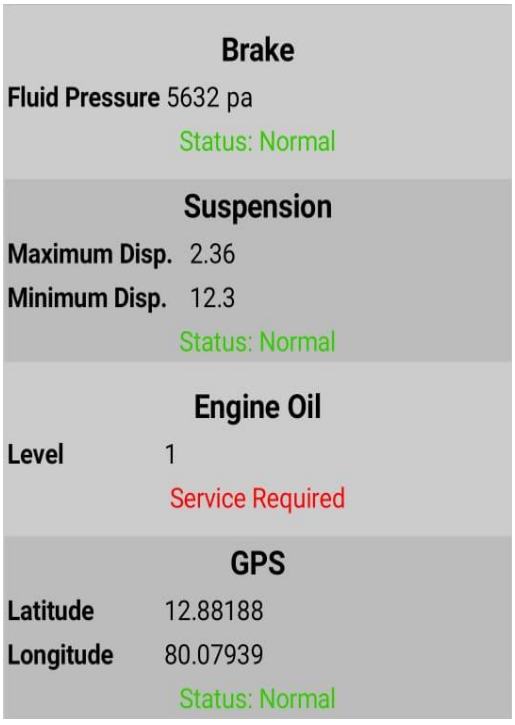

Figure 9(b) Manage Automobile Service

The information of automobile and owner can be changed i.e. if customer changes their vehicle; the vehicle name needs to be updated or if the customer sells the vehicle then owner

name needs to be updated. For this purpose, Edit Profile section as shown in figure 10 is added to the application

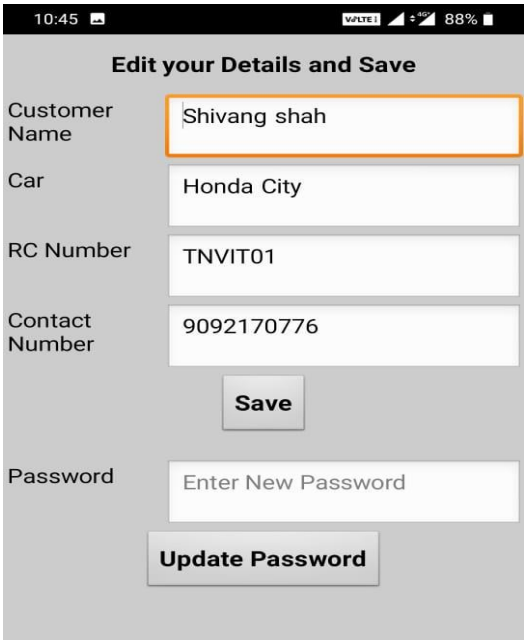

Figure 10 Edit Profile

ii) Firebase

Firebase is an innovation that permits to make web applications with no server-side programming so that development turns out to be quicker and simple [4]. It can be used in many ways: verifying users, storing information, and executing access rules. Applications utilizing this can control and use information.

Firebase also provides a Realtime database and backend as a service. The service provides application developers an API that allows application data to be synchronized across clients and stored on Firebase's cloud. The database is also accessible through a REST API and bindings for several JavaScript frameworks such as AngularJS, React, Ember.js and Backbone.js. The REST API uses the Server-Sent Events protocol, which is an API for creating HTTP connections for receiving push notifications from a server. Developers using the Realtime database can secure their data by using the company's server-side-enforced security rules. Cloud Firestore which is Firebase's next generation of the Realtime Database was released for beta use.

Connecting Application with Firebase

- After adding the screens, we will add buttons and create an authentication key.
- After appropriate authentication it is linked to the Google Firebase.

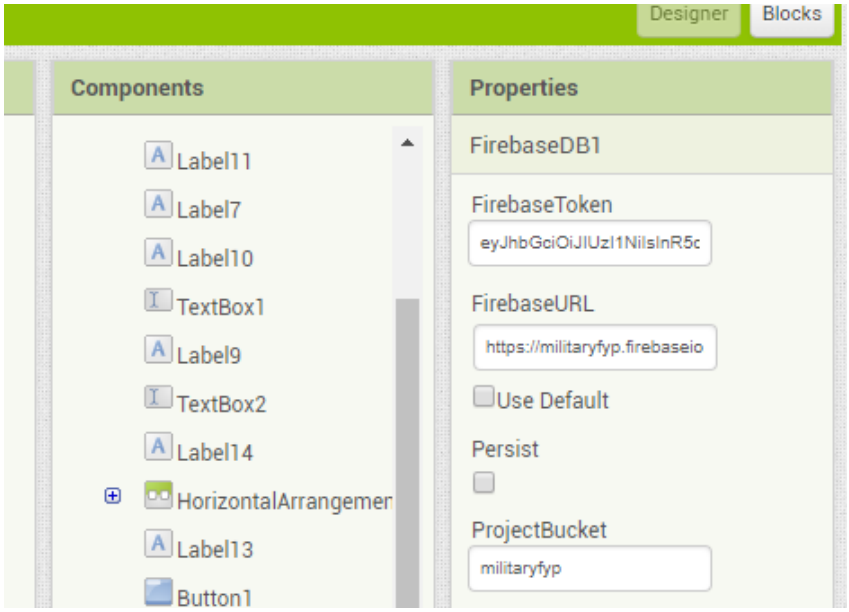

Figure 11 Firebase Authentication Key

Now on google Firebase, make an account.

- It is also linked with the app with the help of API Key.
- A database is created for authentication purpose.

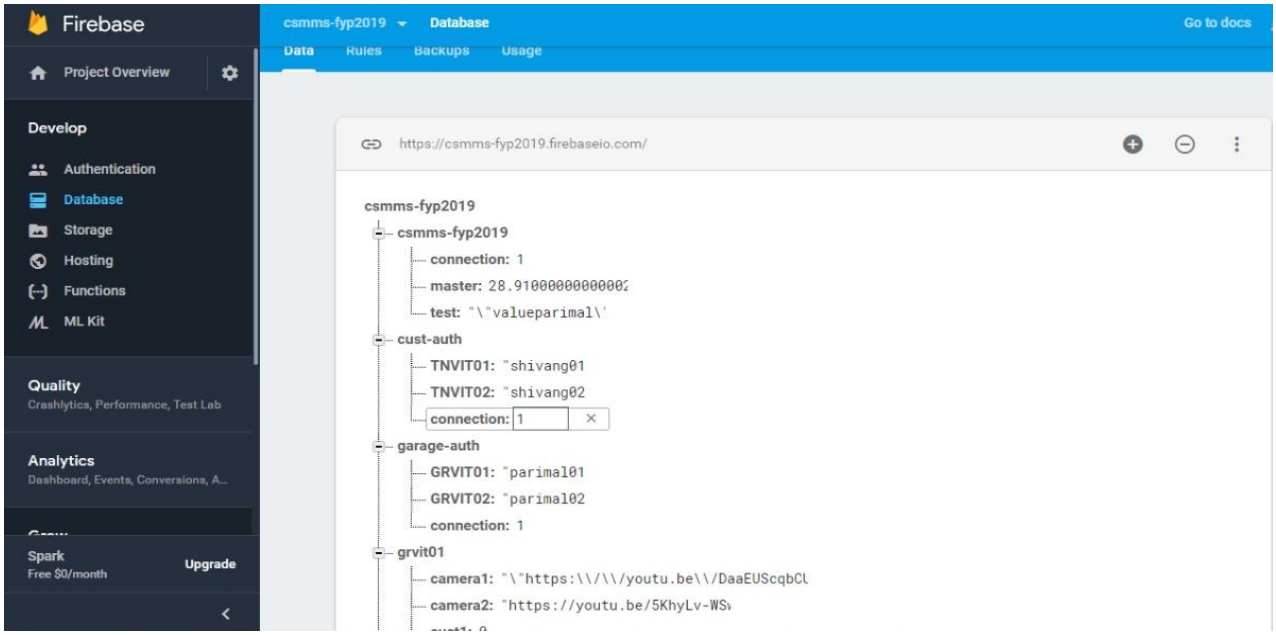

Figure 12 Firebase Console

iii) Live Video Transmission (YouTube and RPA)

Robotic Process Automation is the innovation that enables anybody today to design computer software, or a "robot" to imitate and incorporate the actions of a human interacting within digital systems. RPA robots utilize the user interface to capture data and manipulate applications just like humans do. UiPath Studio features a rich collection of pre-built activities, integrates with several programming languages, and promotes ease-of-use, scalability, and efficiency. There are many Software that provides RPA. UiPath is one such software. YouTube live is used for live streaming. When automobile is in garage, the link of current live video of garage will automatically be sent to the respective customer with the help of RPA deployed on the systems of garage. Garage owner will have the YouTube live video link which will be private so that only those having the link will be able to see the video. Now if one is logged in as garage owner then app will take to the page as shown in figure 13.

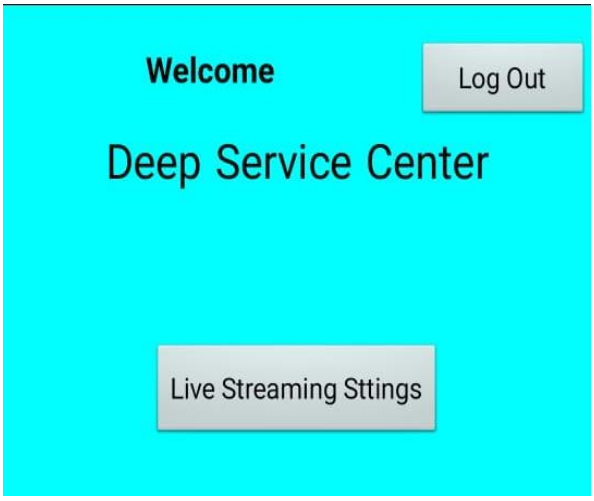

Figure 13 Garage Owner Main Page

Now to update the YouTube link so that automobile owner can see it, click on the ***live streaming setting*** and page will open as shown in figure 14.

Also, the link can be updated manually if required. Paste the YouTube live video link accordingly and click on update link the automobile owner will get notification and the link will

appear on its main page using which automobile owner can watch live video on YouTube official application

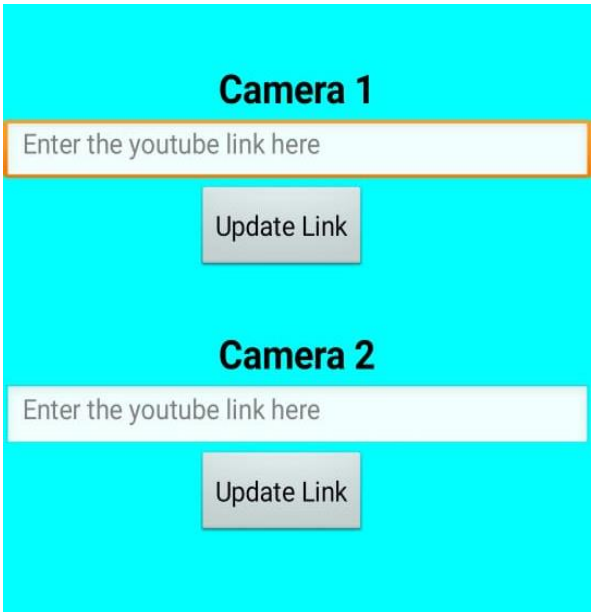

Figure 14 Live Streaming Setting

After garage owner updates the live video link in its login monitoring car servicing live button as shown in the figure 15 will pop up automatically in customer login.

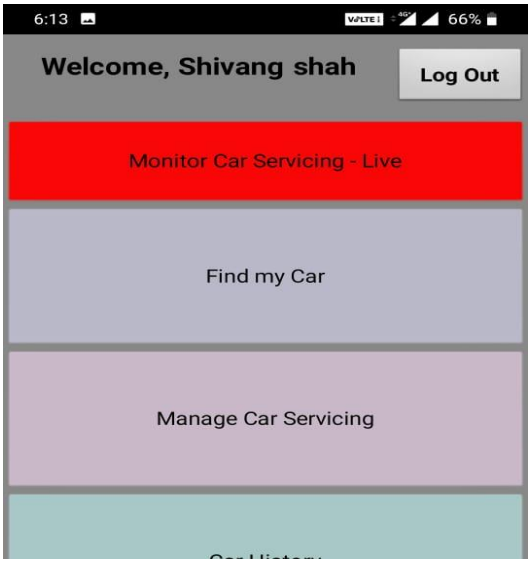

Figure 15 Live Monitoring

When the button will be clicked it will take customer to a page which has the link as shown in the figure 16, when clicked on any of the camera 1 or camera 2 buttons YouTube will open automatically, and live feed of the garage can be seen.

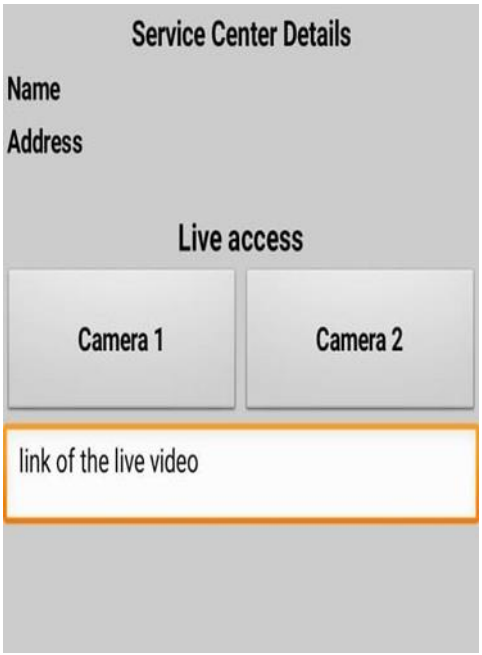

Figure 16 Link for Live Video

iv) Integration

Firebase [5] is used for communication between Application and server and server and Automobile. As firebase is fairly fast, customer gets notification easily as soon as there is any breach. Whenever any customer requests to see the live video with the help of application, Role of **UiPath** is appreciable. UiPath monitors if there is request on Firebase [6]. If request is found UiPath copies the link of the live video of the respective camera of respective garage into the account of user. This happens in 1-2 second and the user is redirected to the YouTube application in their phone and the video is accessed. Keeping in mind the security of garage and customers, the link for accessing camera is changed in every 8 hours. So, only authorized customers can have the access to camera. **Python** is really a popular choice when it comes for **IOT**. Python is processing all the data and sending it to the required tag in the firebase. Embedded C is used for programming atmega328p. All sensors are interfaced with this microcontroller. There are 2 nodes

(microcontroller) which interfaces various sensors. If any node has to send critical data, it sends a signal through a dedicated channel to raspberry pi and raspberry pi listens to the values and sends to firebase and a notification is received with values for the customer. Also, all the sensors data is recorded and updated on firebase at regular interval of 5 mins.

2.4.2 Hardware Details

As shown in Figure 1 we have various hardware components being used in this system. The details of each component is as follows

i) Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer as shown in Figure 17 that plugs into a computer monitor or TV and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word-processing, and playing games.

Figure 17 Raspberry Pi

The device uses the ARM processor which does nearly all of the hard work in order to run the Raspberry Pi.

The main aim of Raspberry Pi is to collect the data from various nodes i.e. Arduino and upload them to the firebase database.

Installation of Raspbian OS

1. Format the SD card- Anything that's stored on the SD card will be overwritten during formatting. So if the SD card on which you want to install Raspbian currently has any files on it, e.g. from an older version of Raspbian, you may wish to back these files up first to not lose them permanently.
2. New Out of Box Software (NOOBS)- Using the NOOBS software is the easiest way to install Raspbian on your SD card.
    - Download NOOBS- Visit the Raspberry Pi downloads page.
    - Extract NOOBS from the zip archive
    - Next, you will need to extract the files from the NOOBS zip archive you downloaded from the Raspberry Pi website.
    - Double-click on it to extract the files, and keep the resulting Explorer/Finder window open.
    - Select all the files in the NOOBS folder and drag them into the SD card window to copy them to the card.

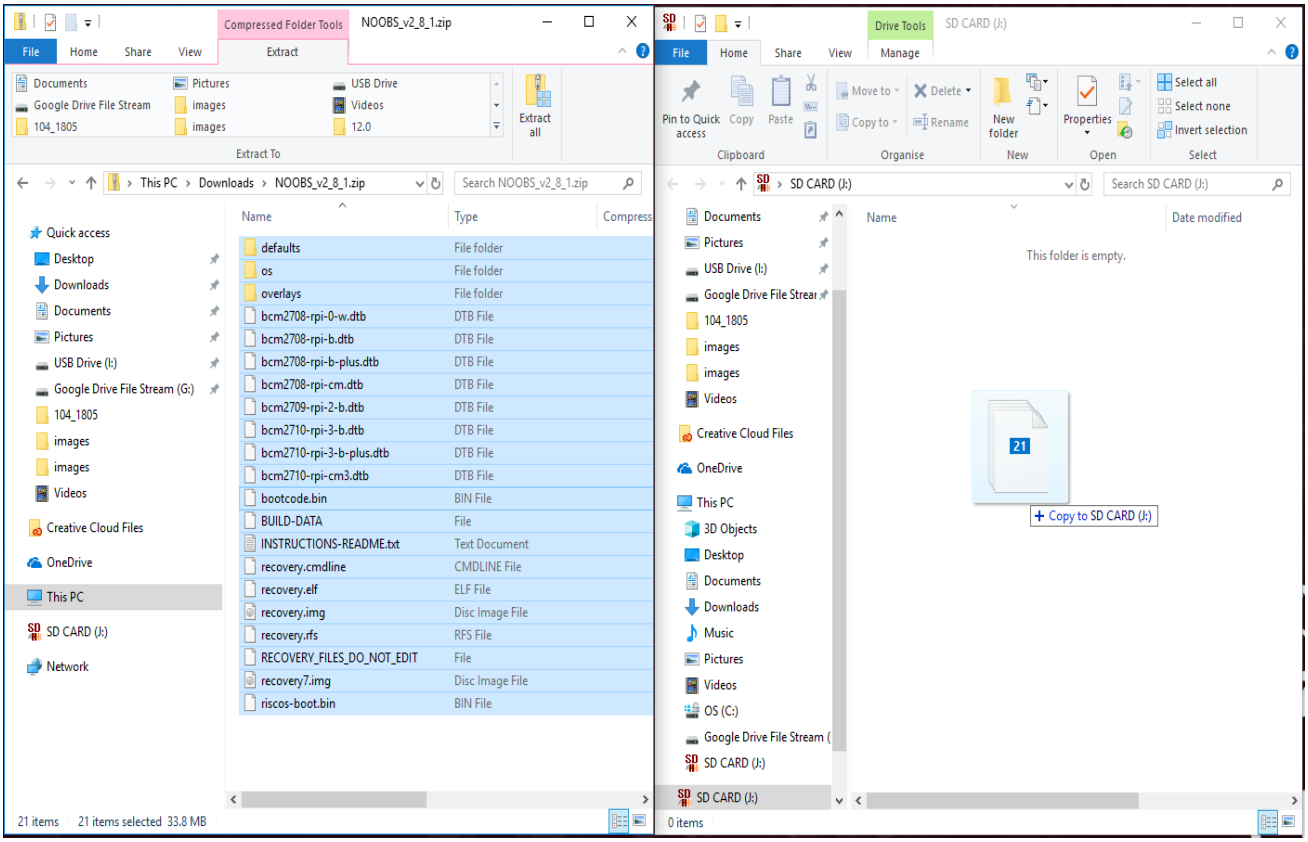

Figure 18 Copying to Sim Card

3. Connect your Raspberry Pi

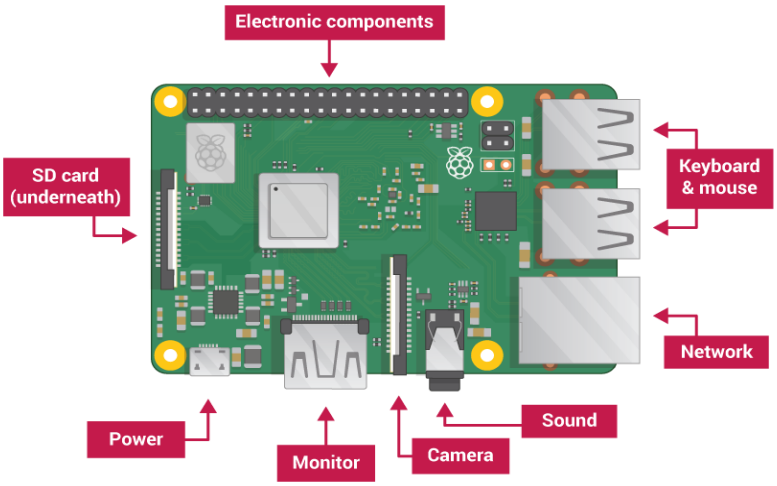

Figure 19 Raspberry Pi

- Insert the SD card with Raspbian (via NOOBS) into the micro SD card slot
- Connect the mouse and the keyboard to a USB port on the Raspberry Pi
- Use a cable to connect a screen to the Pi's HDMI port — use an adapter if necessary.
- Startup Raspberry Pi
- Connect a micro USB power supply it to you Pi's power port.

The red LED light up on the Raspberry Pi, indicates that the Pi is connected to power. As it boots up raspberries will appear in the top left-hand of the screen.

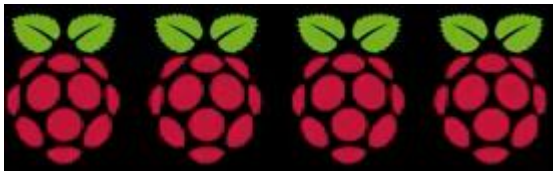

Figure 20 Powering Raspberry Pi

After a few seconds the Raspbian Desktop will appear.

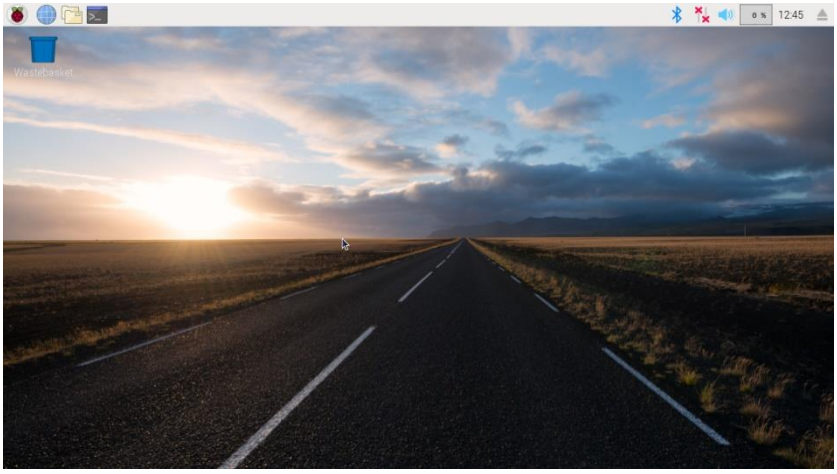

Figure 21 Raspbian Desktop

4. Finish the setup

When you start your Raspberry Pi for the first time, the Welcome to Raspberry Pi application will pop up and guide you through the initial setup.

Figure 22 Raspberry Pi Welcome Screen

- Click Next to start the setup.
- Click Done or Reboot to finish the setup.

Sometimes it is not convenient to work directly on the Raspberry Pi as sometimes we need to work on it from another device by remote control .VNC is a graphical desktop sharing system that allows us to remotely control the desktop interface of one computer (running VNC Server) from another computer or mobile device (running VNC Viewer). VNC Viewer transmits the keyboard and either mouse or touch events to VNC Server and receives updates to the screen in return.

VNC viewer and VNC server

VNC is a graphical desktop sharing system that allows one to remotely control the desktop interface of one computer (running VNC Server) from another computer or mobile device (running VNC Viewer). VNC Viewer transmits the keyboard and either mouse or touch events to VNC Server and receives updates to the screen in return.

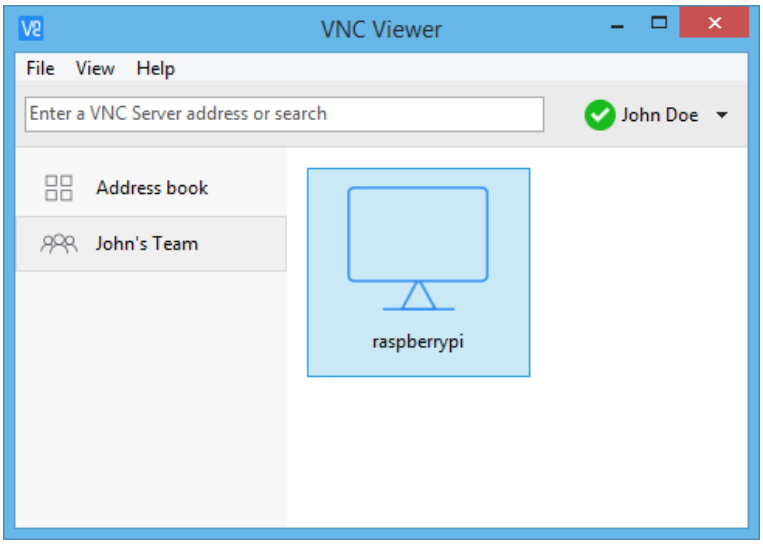

Figure 23 VNC Viewer

VNC Connect from Real-VNC is included with Raspbian. It consists of both VNC Server, which allows us to control your Raspberry Pi remotely, and VNC Viewer, which allows us to control desktop computers remotely from our Raspberry Pi.

1. Enabling VNC Server

On the Raspberry Pi, run the following commands to make sure the latest version of VNC is installed. Follow the following steps:

In the Terminal enter these commands

- sudo apt-get update
- sudo apt-get install realvnc-vnc-server realvnc-vnc-viewer

Now enable VNC Server. You can do this graphically or at the command line.

- Enabling VNC Server graphically
  - On your Raspberry Pi, boot into the graphical desktop.
  - Select **Menu > Preferences > Raspberry Pi Configuration > Interfaces**.
  - Ensure **VNC** is **Enabled**.
- Enabling VNC Server at the command line
  - You can enable VNC Server at the command line
  - sudo raspi-config
  - Now, enable VNC Server by doing the following:
    - Navigate to **Interfacing Options**.
    - Scroll down and select **VNC > Yes**.

2. Connecting to your Raspberry Pi with VNC Viewer

There are two ways to connect to your Raspberry Pi.

- Establishing a direct connection-Direct connection is quick and simple providing you're joined to the same private local network as your Raspberry Pi. For example, this might be a wired or wireless network at home, at school, or in the office).
  - On your Raspberry Pi (using a terminal window or via SSH) use these instructions or run ifconfig to discover your private IP address.
  - On the device you'll use to take control, download VNC Viewer. For best results, use the compatible app from RealVNC.
  - Enter your Raspberry Pi's private IP address into VNC Viewer:

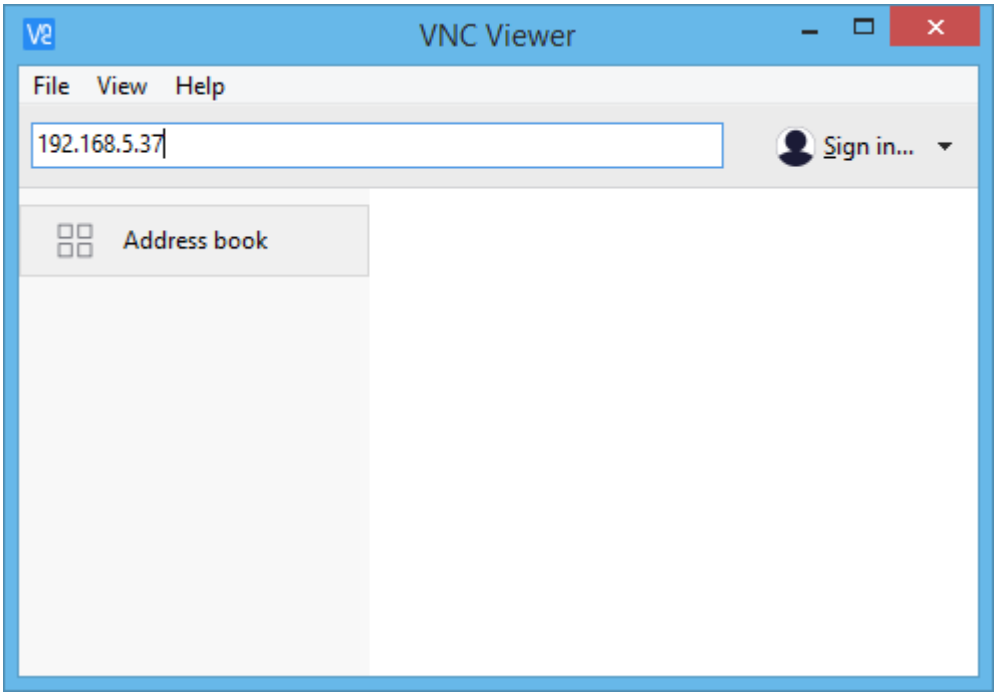

Figure 24 VNC Viewer Address

- Establishing a cloud connection- Cloud connections are convenient and encrypted end-to-end. They are highly recommended for connecting to your Raspberry Pi over the internet.
  - On your Raspberry Pi, sign in to VNC Server using your new RealVNC account credentials:

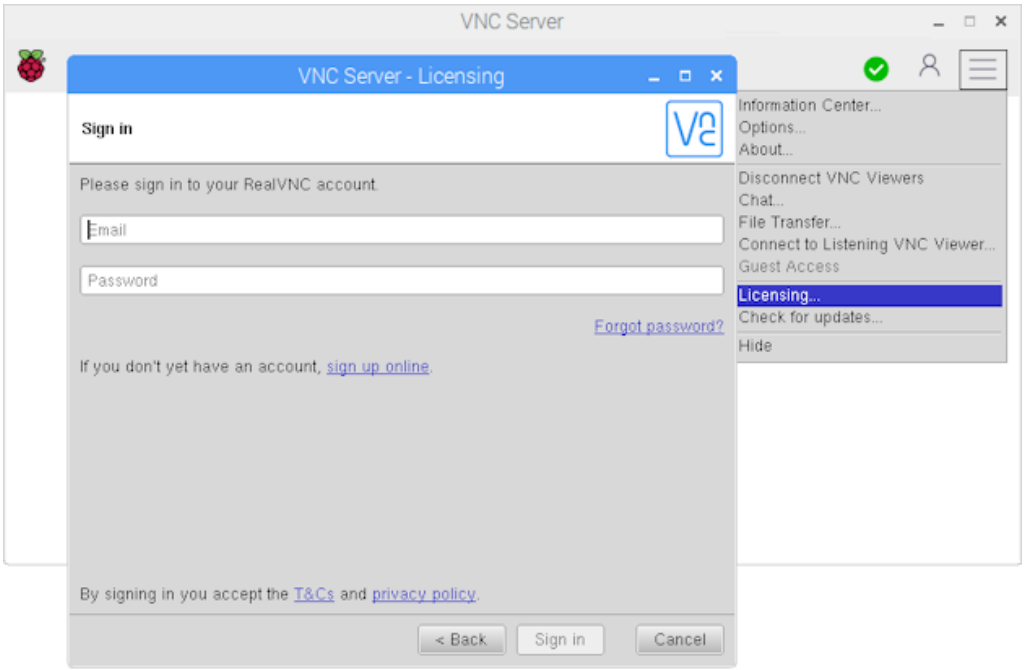

Figure 25 VNC Server

- On the device you'll use to take control, download VNC Viewer.
- Sign in to VNC Viewer using the same RealVNC account credentials, and then connect to your Raspberry Pi or set up a new connection:

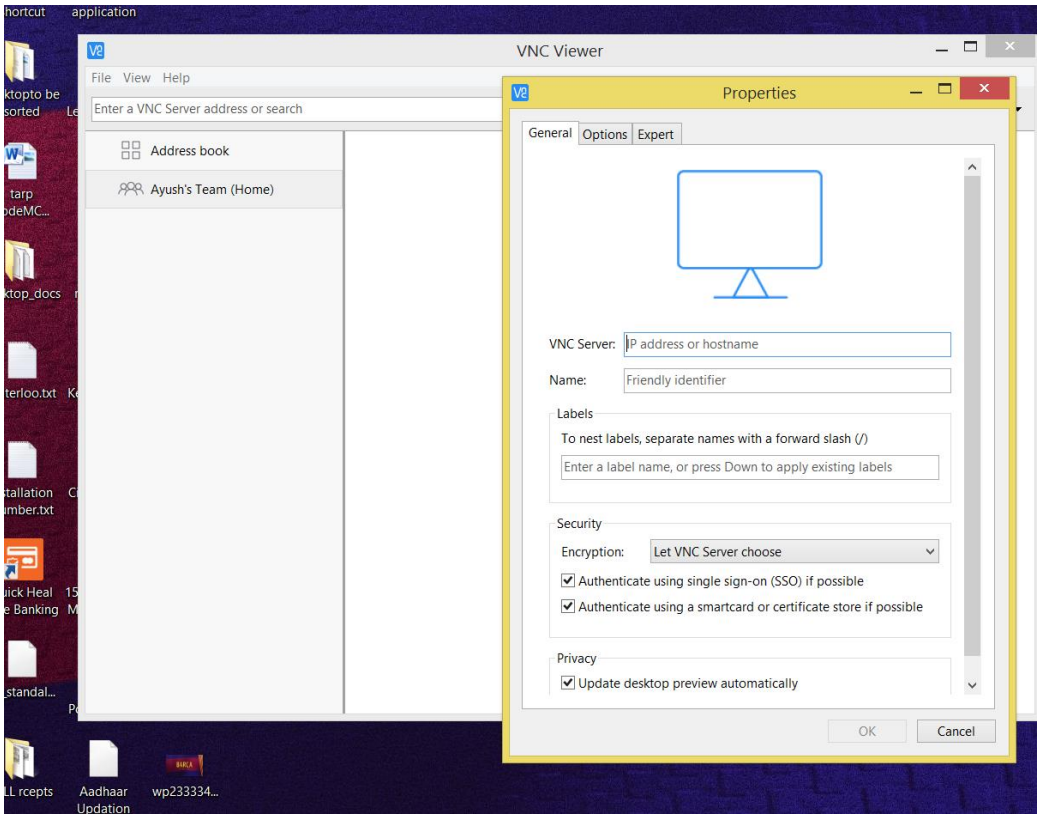

Figure 26 VNC Properties Window

- NOTE: IP address here is Internet IP address which is different from Ethernet IP (LAN) address as both Ethernet port and built in WIFI modules have different MAC addresses

ii) Arduino

Arduino is an open source microcontroller as shown in Figure 27 which can be easily programmed, erased and reprogrammed at any instant of time. Introduced in 2005 the Arduino platform was designed to provide an inexpensive and easy way for hobbyists, students and professionals to create devices that interact with their environment using sensors and actuators. Based on simple microcontroller boards, it is an open source computing platform that is used for constructing and programming electronic devices.

Figure 27 Arduino

The main purpose of the component is to collect the values, data from the various sensors and send it to the Raspberry Pi

iii) Hall Effect Module

A Hall effect module is a device as shown in Figure 28 that is used to measure the magnitude of a magnetic field. Its output voltage is directly proportional to the magnetic field strength through it. Hall effect module are used for proximity sensing, positioning, speed detection, and current sensing applications In order to guarantee Hall Effect module optimal behavior, high sensitivity, low offset, and low temperature drift are performance aspects that need to be achieved.

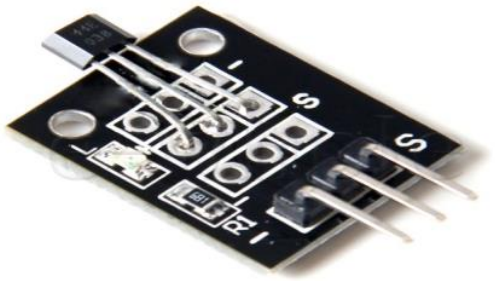

Figure 28 Hall Effect Module

The main purpose of this component is to monitor the RPM (rotation per minute) of the crankshaft connecting piston rods to pistons heads. And send those values to the respective nodes i.e. Arduino.

iv) Linear Potentiometer

A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a variable resistor or rheostat.

Figure 29 Linear Potentiometer

The measuring instrument called a potentiometer is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.

Here linear potentiometer depicts the suspension of the car. It behaves similar to the suspension and gives the values accordingly and sends those values to the node.

v) SIM 808

SIM808 module is a complete Quad-Band GSM/GPRS module which combines GPS technology for satellite navigation.

Figure 30 SIM 808

The compact design which integrated GPRS and GPS in a SMT package will significantly save both time and costs for customers to develop GPS enabled applications. Featuring an industry-standard interface and GPS function, it allows variable assets to be tracked seamlessly at any location and anytime with signal coverage. This GPS module will continuously track the location of the car and will sends the longitude and latitude values to the node i.e. Arduino.

vi) 3 - Axis Accelometer

The 3-axis accelerometer is based on the principle of capacitive sensing. The fig.1 shows basic principle of accelerometer sensor. The sensor is made of spring loaded, micro machined structure, mounted on silicon base. Force on the structure changes the position of seismic mass attached on the spring. This deflection is measured using fixed plate capacitor sensors. The change in acceleration unbalances capacitor plate distance, observed by modulation/demodulation circuits and thus, resulted in output proportional to acceleration. The sensing can be static (gravity) or dynamic (forced acceleration).

Figure 31 3- Axis Accelerometer

vii) Temperature Sensor

We will be using two different temperature sensors. First one being LM35 as shown in Figure 8 will be used to measure the temperature of the engine and IR temperature sensor as shown in Figure 9 will be used calculate temperature of the break rotar.

a) LM35

LM 35 is a precision temperature sensor whose output is linearly proportional to Celsius Temperature. The LM35 is rated to operate from -55° Centigrade to 150° Centigrade with a linear scale factor of +10mv/° C.

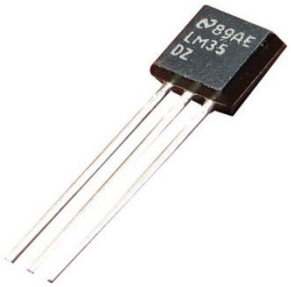

Figure 32 LM35

viii) Voltage Detector

A voltage detector is a device as shown in Figure 33 that determines the presence/absence of an electrical charge in an object. It can be a simple, pen-shaped piece of testing hardware that indicates the presence of electricity or an advanced tool that detects precise voltage levels in electrical systems.

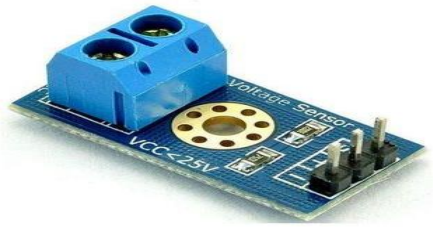

Figure 33 Voltage Detection Sensor

Here, this component monitors the health of the battery by checking the voltage of the battery and passes the values to the Arduino.

CHAPTER 3

COST ANALYSIS

3.1 List of components and their cost

The costs of the various components used in this project are given below in Table 3.1.

Table 3.1 List of components and their costs

COMPONENT	COST
Raspberry Pi	₹ 3000
Arduino (2 nos.)	₹ 1000
Hall Effect	₹ 120
Linear Potentiometer	₹ 350
SIM808 (GSM/GPS)	₹ 2200
Temperature Sensor	₹ 65
3-Axis Accelerometer	₹ 250
Voltage Detector	₹ 150
Miscellaneous	₹ 500
TOTAL	₹ 7635

CHAPTER 4

RESULTS AND DISCUSSIONS

a. Sensor Readings

The end nodes were able to transmit the values collected from the sensors by the system depicted in figure 34 to the Raspberry Pi, which is shown in Figure 36 and 37.

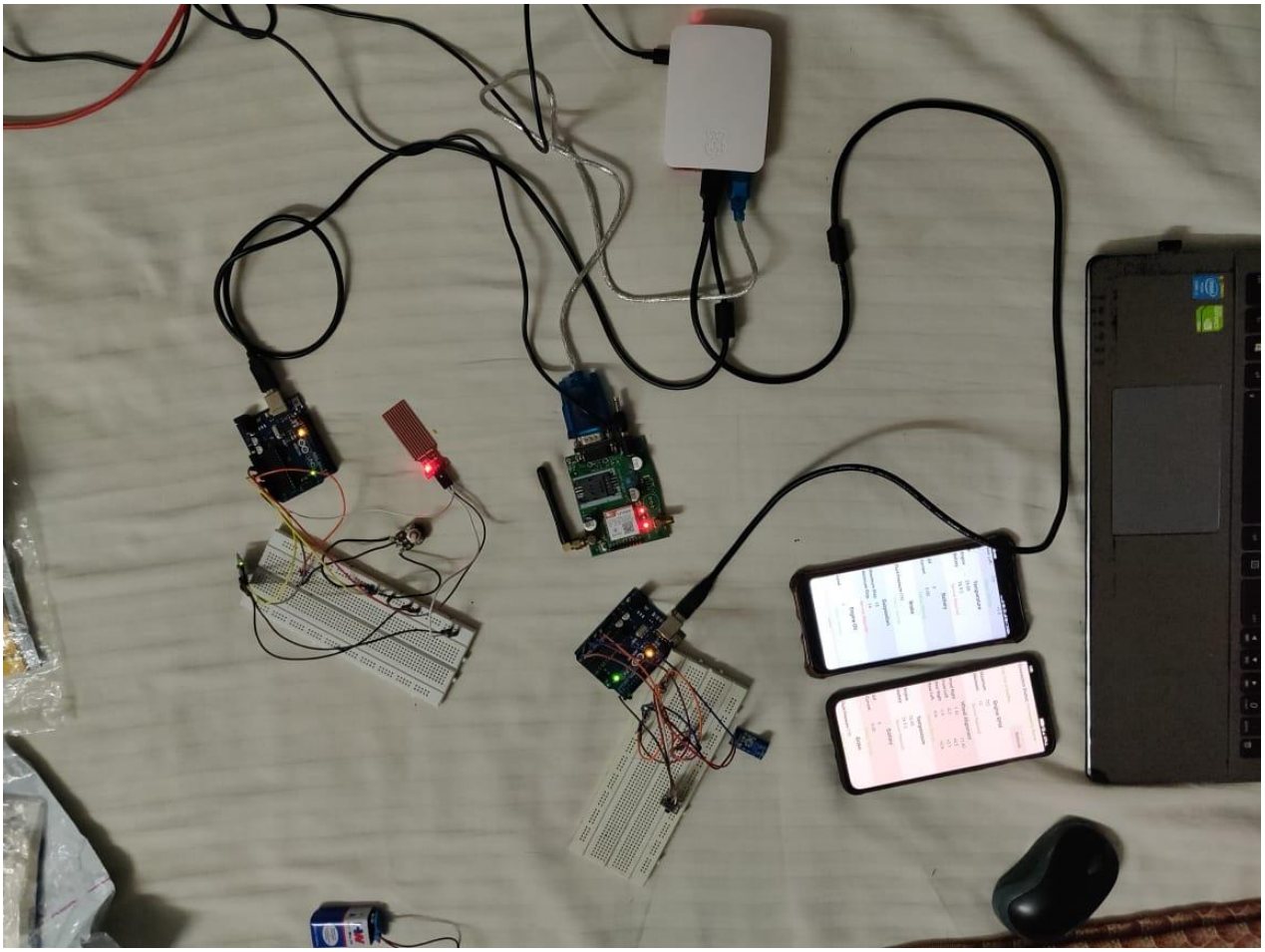

Figure 34 Hardware Setup

b. Integration Hardware and Software

The application consists of a home page to log in, as shown in Figure 7, which further leads us to the main page, as shown in Figure 8. The values were simultaneously uploaded on the application via firebase as shown in figure 34 by the Pi, as shown in Figure 36 & 37. Along with displaying the values on the application, the Pi was also able to take some decisions based on the values. For example: when the battery voltage drops, alert message is being sent and also the battery column in the application shows service required in red colored fonts as shown in figure 9(a) &9(b).

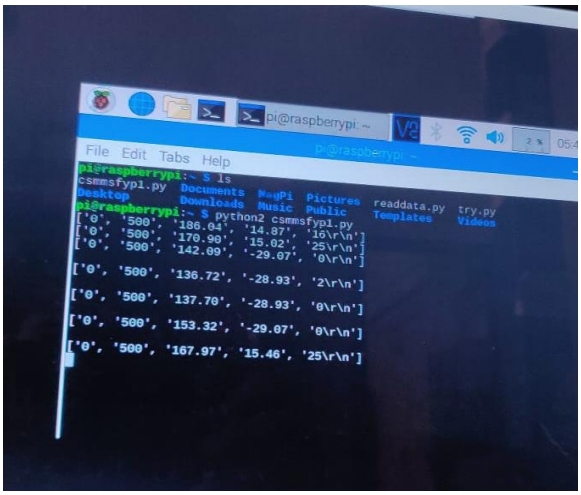

Figure 36 Pi Receiving data

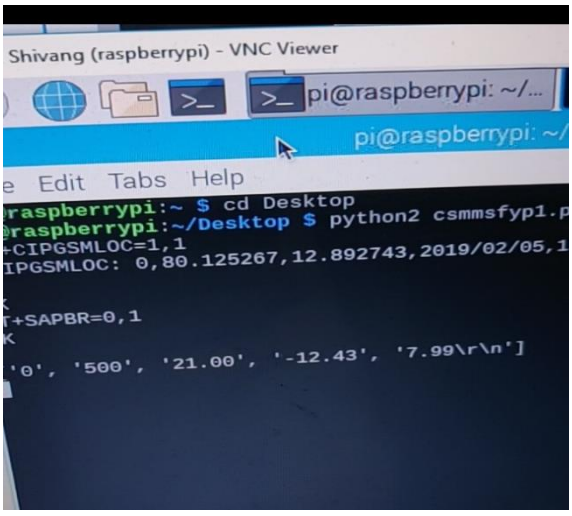

Figure 37 Pi Receiving data

CHAPTER 5

CONCLUSION AND FUTURE WORK

Car being an important part of our daily life needs to be regularly serviced for efficient working. Automation with IoT makes the whole experience of car servicing smart and fast. Above proposed system not only manages real-time health of our car but also provides necessary data and predictions to help us determine the time for next service and approximate cost. Though this system adds to the servicing cost, but it prevents service centers from charging more and makes customer aware about all the modulations done on car. All in all, this system saves time and money of customer. Technologies like IoT and RPA has fundamentally altered the way we live and work. It has made our life easier. This system increases the efficiency of our car and also reduces customer's effort at the same time.

Lot can be done in this area. There is a large scope which could be ventured, and new designs or system could be made to improve the conditions and efficiency of the vehicles and by using AI we can figure out if in near future any of the component might need attention.

CHAPTER 6

APPENDIX

Raspberry Pi Code

```
import serial
import time
from firebase import firebase
ser1 = serial.Serial('/dev/ttyACM0', 9600)
ser2 = serial.Serial('/dev/ttyACM1', 9600)
port = serial.Serial("/dev/ttyUSB0", baudrate=9600, timeout=1)
firebase = firebase.FirebaseApplication('https://csmms-fyp2019.firebaseio.com/', None)
data1=[]
data2=[]
dataGPS=[]

#Set threshold values
TminRPM=10
TmaxRPM=300
TengineTemp=30
TbatteryCurrent=2.0
TbatteryVoltage=5
required=0
normal=1
GPSlat="12.881880"
GPSlng="80.079387"
TpotValue=200
TpotValue2=800
TfuelValue=100
TbrakeSensor=0
Tcamber=0
Ttoe=0
def sendSMS():
    # Transmitting AT Commands to the Modem
    # '\r\n' indicates the Enter key
    port.write("AT+CMGF=1\r\n")
    time.sleep(1)
    port.write("AT+CSMP=17,167,0,16\r\n")
    time.sleep(1)
    rcv = port.read(50)
    port.write('AT+CMGS="7904027417"\r\n')
    port.write("Dear TNVIT01\nSome system of your car in not functioning properly. Please open
application for details.")
    time.sleep(1)
    port.write(chr(26))
    time.sleep(1)
```

```
    port.write("\n")
    time.sleep(1)
    port.flush()
    rcv = port.read(100)
    print (rcv)
    time.sleep(1)

def getloc():
    port.write("AT+SAPBR=3,1,\"Contype\",\"GPRS\"\r")
    time.sleep(1)
    port.write("AT+SAPBR=1,1\r")
    time.sleep(1)
    rcv = port.read(100)
    port.write("AT+CIPGSMLOC=1,1\r")

    time.sleep(5)
    port.write("AT+SAPBR=0,1\r")
    time.sleep(1)

    rcv = port.read(100)
    dataGPS=rcv.split(',')
    GPSlat=dataGPS[3]
    GPSlng=dataGPS[4]
    print (rcv)
    time.sleep(1)



#getloc()
while 1:

    ser1.flushInput()
    #while not(ser1.in_waiting >0):
    #    x=0

    a=ser1.readline()
    #print(a)
    data2=a.split('\r\n')
    data1=data2[0].split(',')
    leng=len(data1)
    print("Node1:")
    print(leng)
    if(leng==5):
        minRPM=data1[0]
        maxRPM=data1[1]
        engineTemp=data1[2]
        batteryCurrent=data1[3]
        batteryVoltage=data1[4]
        #getloc()
```

```
    print(data1)
    result=firebase.put('tnvit01/rpm','0',minRPM)
    #time.sleep(1)
    result=firebase.put('tnvit01/rpm','1',maxRPM)
    #time.sleep(1)
    result=firebase.put('tnvit01/temp','0',engineTemp)
    #time.sleep(1)
    result=firebase.put('tnvit01/battery','1',batteryCurrent)
    #time.sleep(1)
    result=firebase.put('tnvit01/battery','0',batteryVoltage)
    #time.sleep(1)
    result=firebase.put('tnvit01/gps','0',GPSlat)
    #time.sleep(1)
    result=firebase.put('tnvit01/gps','1',GPSlng)
    #time.sleep(1)
    print(data1)
    if(minRPM<TminRPM):
        result=firebase.put('tnvit01/sensor','0',required)
        #sendSMS()
    else:
        result=firebase.put('tnvit01/sensor','0',normal)

    if(maxRPM>TmaxRPM):
        result=firebase.put('tnvit01/sensor','0',required)
        #sendSMS()
    else:
        result=firebase.put('tnvit01/sensor','0',normal)

    if(engineTemp>TengineTemp):
        result=firebase.put('tnvit01/sensor','2',required)
        #sendSMS()
    else:
        result=firebase.put('tnvit01/sensor','2',normal)

    if(batteryCurrent<TbatteryCurrent):
        result=firebase.put('tnvit01/sensor','3',required)
        #sendSMS()
    else:
        result=firebase.put('tnvit01/sensor','3',normal)

    if(batteryVoltage<TbatteryVoltage):
        result=firebase.put('tnvit01/sensor','3',required)
        #sendSMS()
    else:
        result=firebase.put('tnvit01/sensor','3',normal)


ser2.flushInput()
#while not(ser2.in_waiting >0):
```

```
 #   x=0

a=ser2.readline()
#print(a)
data2=a.split('\r\n')
data1=data2[0].split(',')
leng=len(data1)
print("Node2:")
print(leng)
if(leng==6):
   potValue1=data1[0]
   potValue2=data1[1]
   fuelValue=data1[2]
   brakeSensor=data1[3]
   camber=data1[4]
   toe=data1[5]
   #getloc()
   print(data1)
   result=firebase.put('tnvit01/suspension','0',potValue1)
   #time.sleep(1)
   result=firebase.put('tnvit01/suspension','1',potValue2)
   #time.sleep(1)
   result=firebase.put('tnvit01/engineoil','0',fuelValue)
   #time.sleep(1)
   result=firebase.put('tnvit01/brake','0',brakeSensor)
   #time.sleep(1)
   result=firebase.put('tnvit01/wheel','1',camber)
   #time.sleep(1)
   result=firebase.put('tnvit01/wheel','0',toe)
   #time.sleep(1)
   print(data1)
   if(potValue1<TpotValue):
      result=firebase.put('tnvit01/sensor','6',required)
      #sendSMS()
   else:
      result=firebase.put('tnvit01/sensor','6',normal)

   if(fuelValue>TfuelValue):
      result=firebase.put('tnvit01/sensor','0',required)
      #sendSMS()
   else:
      result=firebase.put('tnvit01/sensor','0',normal)

   if(brakeSensor>TbrakeSensor):
      result=firebase.put('tnvit01/sensor','5',required)
      #sendSMS()
   else:
      result=firebase.put('tnvit01/sensor','5',normal)
```

```
if(camber<Tcamber):
   result=firebase.put('tnvit01/sensor','2',required)
   #sendSMS()
else:
   result=firebase.put('tnvit01/sensor','2',normal)

if(toe<Ttoe):
   result=firebase.put('tnvit01/sensor','2',required)
   #sendSMS()
else:
   result=firebase.put('tnvit01/sensor','2',normal)
```

Arduino Code (Node 1)

```
unsigned long timeold;
unsigned long timeold2;
unsigned int rev;
unsigned int wheel;
unsigned int minRPM=500;
unsigned int maxRPM=0;
int val;
int tempPin = A2;  //LM35

const int currentPin = A1;  //Current sensor
const int voltagePin = A0;  //Voltage sensor
int sensitivity = 66;
int adcValue= 0;
int offsetVoltage = 2500;
int offset =20;
int adcVoltage = 0;
float currentValue = 0;
void setup()
{
```

```
  Serial.begin(9600);
  attachInterrupt(0, wheel_rpm, RISING);
  rev = 0;
  timeold = 0;
}
void loop()
{
  if (rev >= 1)
  {
    detachInterrupt(0);
    wheel = 30 * 1000 / (millis() - timeold) * rev;
    timeold = millis();
    rev = 0;
    val = analogRead(tempPin);
    //float mv = ( val/1024.0)*5000;
    //float cel = (mv/10);
    /*Serial.print("Time=");
    Serial.print(timeold);
    Serial.print("    ;");
    Serial.print("rpm=");
    Serial.println(wheel,DEC);*/
    if(wheel<minRPM && wheel>10)
    {
      minRPM=wheel;
    }
    if(wheel>maxRPM && wheel<1000)
    {
      maxRPM=wheel;
    }
    attachInterrupt(0, wheel_rpm, RISING);
  }
  if(millis()-timeold2>2000)
  {

    timeold2=millis();
    //send Data
        val = analogRead(tempPin);
    float mv = ( val/1024.0)*5000;
    float cel = (mv/10);
    adcValue = analogRead(currentPin);
    int volt = analogRead(voltagePin);
    int voltage = map(volt,0,1023, 0, 2500) + offset;
    voltage /=100;
    adcVoltage = (adcValue / 1024.0) * 5000;
    //currentValue = ((adcVoltage - offsetVoltage) / sensitivity);
    currentValue = voltage*0.04;
    Serial.print(maxRPM);
    Serial.print(",");
    Serial.print(minRPM);
```

```
    Serial.print(",");
    Serial.print(cel);
    Serial.print(",");
    Serial.print(currentValue);
    Serial.print(",");
    Serial.println(voltage);
  }
}

void wheel_rpm()
{
  rev++;
}
```

Arduino Code (Node 2)

```
#include <MPU6050_tockn.h>
#include <Wire.h>

MPU6050 mpu6050(Wire);
int linearPot = A0;
int fuelSensor = A1;
int brakeSensor = A2;
int minpot=1024;
int maxpot=0;
void setup()
{
  Serial.begin(9600);

  Wire.begin();
  mpu6050.begin();
```

```
  mpu6050.calcGyroOffsets(true);

}

void loop()
{
 mpu6050.update();
  //Serial.print("angleX : ");
  float camber=mpu6050.getAngleX();
  //Serial.print("\tangleY : ");
  float toe=mpu6050.getAngleY();
 int potValue = analogRead(linearPot);
 if(potValue<minpot)
 {
  minpot=potValue;
 }
 if(potValue>maxpot)
 {
  maxpot=potValue;
 }
 int fuelValue2 = analogRead(fuelSensor);
int fuelValue=fuelValue2*100/700;
  int brakeStatus = analogRead(brakeSensor);
  // Output
  Serial.print(minpot);
   Serial.print(",");
    Serial.print(maxpot);
   Serial.print(",");
  Serial.print(fuelValue);
   Serial.print(",");
```

```
    Serial.print(brakeSensor);
   Serial.print(",");
  Serial.print(camber);
  Serial.print(",");
  Serial.print(toe);

  Serial.println();

  delay(1000);
}
```

REFERENCES

[1] Jr-Jen Huang, Yi-Yu Chu, and Yen-Jen Chen, "The System Design and Implementation of Vehicle Management", Journal of Advances in Computer Networks, Vol. 1, No. 1, March 2013

[2] Rohit Dhall, Vijender Solanki, "An IoT Based Predictive Connected Car Maintenance Approach", International Journal of Interactive Multimedia and Artificial Intelligence, Vol. 4, Nº3

[3] MIT App Inventor http://ai2.appinventor.mit.edu/

[4] K.N. Manoj Kumar, Kailasa Akhi, Sai Kumar Gunti, M. Sai Prathap Reddy, "Implementing smart home using firebase", International Journal of Research in Engineering and Applied Sciences (IJREAS), Vol. 6 Issue 10, October - 2016, pp. 193~198

[5] Navdeep Singh, "Study of Google Firebase API for Android", International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization) Vol. 4, Issue 9, September 2016

[6] Sonam Khedkar, Swapnil Thube, "Real Time Databases for Applications", International Research Journal of Engineering and Technology (IRJET) Real Time Databases for Applications, Volume: 04 Issue: 06 | June -2017

[7] Harshada Chaudhari, "Raspberry Pi Technology: A Review" International Journal of Innovative and Emerging Research in Engineering Volume 2, Issue 3, 2015

[8] Cheah Wai Zhao, Jayanand Jegatheesan, Son Chee Loon, "Exploring IOT Application Using Raspberry Pi", International Journal of Computer Networks and Applications Volume 2, Issue 1, January - February (2015)

[9] How to set up Raspberrypi https://www.raspberrypi.org/help/videos/

[10] Connecting an Arduino to Raspberry Pi for the best of both worlds https://conoroneill.net/2013/06/05/connecting-an-arduino-to-raspberry-pi-for-the-best-of-both-worlds/

[11] Maria-Alexandra Paun, Jean-Michel Sallese and Maher Kayal, "Hall Effect Sensors Design, Integration and Behavior Analysis" Journal of Sensor and Actuator Networks ISSN 2224-2708, Published: 8 February 2013

[12] Paun, M.A.; Sallese, J.M.; Kayal, M. "Temperature Considerations on Hall Effect Sensors Current-Related Sensitivity Behaviour", In *Proceedings of the 19th IEEE International* Conference *on Electronics, Circuits, and Systems (ICECS)*, Seville, Spain, 9–12 December 2012

[13] Electronics Basics – How a Potentiometer Works https://randomnerdtutorials.com/electronics-basics-how-a-potentiometer-works/

[14] J. Parthasarathy, “POSITIONING AND NAVIGATION SYSTEM USING GPS”, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, Volume XXXVI, Part 6, Tokyo Japan 2006

[15] Phongsak Keeratiwintakorn, “Real-Time Tracking Management System Using GPS, GPRS and Google Earth”, Conference Paper, June 2008

[16] Anas Siddiqui, Usman Saleem, Abdul Ur Rehman, Sohaib, Shiraz Latif, “GPS and GSM Based Advanced Vehicle Monitoring and Information System”, First International Conference on Modern Communication & Computing Technologies (MCCT'14)

[17] Accelerometer Basics: https://learn.sparkfun.com/tutorials/accelerometer-basics/all

[18] Infrared Temperature Measurement Theory and Application Author and Presenter: John Merchant, Sales Manager, Mikron Instrument Company Inc.

[19] Paun, M.A.; Sallese, J.M.; Kayal, M., “Temperature Influence Investigation on Hall Effect Sensors Performance Using a Lumped Circuit Model”. In *Proceedings of the 11th IEEE Sensors Conference*, Taipei, Taiwan, 28–31 October 2012

[20] Working Principle of Temperature Sensor and Its Application https://www.efxkits.us/lm35-temperature-sensor-circuit-working/

[21] What is Voltage Detector Used For https://www.wonkeedonkeetools.co.uk/voltage-detectors-and-testers/what-is-a-voltage-detector-used-for/

BIODATA

Photo

Name : xxxxx
Mobile Number : 123456789
E-mail : xx.xxxx@vitap.ac.in
Permanaent Address :

NOTE: Its **MANDATORY** for a student to attach all the PPT's, Sample Materials, Specification Sheets, Programming Codes and a 5-10 minutes demo Video of the Project Digitally In CD . Stick the Compact Disk (CD) in the final page of the Thesis after binding it.