

Bachelors of Technology Project Report

(on)

Vehicle Monitoring System Using the Internet of Things

By

Arpan Mangal (1310110076)

Bikki Manoj (1310110104)

Mayank Jain (1310110206)

Supervisor

Dr. Debopam Acharya

debopam.acharya@snu.edu.in

Submitted in the partial fulfillment of requirements for

B.Tech in Computer Science and Engineering



SHIV NADAR UNIVERSITY

Department of Computer Science and Engineering,

School of Engineering , Shiv Nadar University,

Gautam Buddha Nagar, U.P., India, 201314

<http://www.snu.edu.in>

Approval Sheet

This report titled “Vehicle Monitoring System Using the Internet of Things“ by Arpan Mangal, Bikki Manoj and Mayank Jain is approved for the degree of B. Tech in Computer Science and Engineering.

Project Advisor

Name Dr. Debopam Acharya

Signature _____

Date _____

Declaration Sheet

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of the Student _____Signature_____

Name of the Student _____Signature_____

Name of the Student _____Signature_____

Date _____

Abstract

In this modern and fast moving world, it has become a basic necessity to be constantly aware of youngster`s safety. When youngsters are just eligible to drive, it is important to keep monitoring their driving to where they are going and at what speed. Therefore, parents might put constraints to where they should go and at what speed by messaging them periodically.

In this work, have developed a vehicle event monitoring system using IoT Technologies. Along with tracking of various events related to driving, we have also developed an emergency/panic button and installed fire sensor inside the car which would send an emergency message to the parent`s mobile number and relevant emergency response service agencies during abnormal situation.

1. Content

2. Executive Summary and Related Works.....	9
2.1. Executive Summary.....	13
2.2. Related Works.....	14
3. Project Management.....	16
3.1. The team.....	16
3.1.1. Group Member 1.....	16
3.1.2. Group Member 2.....	16
3.1.3. Group Member 3.....	16
3.1.4. Project Mentor.....	17
3.2. Methodology.....	17
3.2.1. Software Development Process.....	17
3.2.2. Schedule.....	17
3.2.3. Work Breakdown.....	18
3.3. Budget.....	18
3.4. Milestones.....	19
4. Scope of the Project.....	21
5. Project Description.....	23
5.1. Architecture.....	24
5.2. Sensing System.....	25
5.3. Server.....	25
5.4. Mobile Application.....	26
6. Technical Requirements.....	27
6.1. Performance Requirements.....	28
6.1.1. Sensing Requirements.....	28
6.1.2. Mobility Requirements.....	28
6.2. Hardware Requirements.....	28
6.2.1. Raspberry-Pi Requirements.....	29
6.2.2. Sensing Requirements.....	29
6.2.2.1. Fire/Flame Sensor.....	30

6.2.2.2. GPS Module.....	31
6.2.2.3. Push Button.....	33
6.2.2.4. Cooling Fan.....	34
6.2.3. Communication Requirements.....	35
6.2.4. Other Electronic Requirements.....	35
6.3. Tools and Software Requirements.....	35
6.3.1. Python.....	36
6.3.2. Firebase.....	36
6.3.3. Twilio.....	36
6.3.4. MySQL.....	36
6.3.5. Android Studio.....	36
6.3.6. Microsoft Visio 2016.....	37
6.3.7. Fritzing.....	37
6.4. Deliverables.....	37
6.4.1. Basic Setup.....	37
6.4.2. Software Deployed in Raspberry Pi.....	37
6.4.3. Documentation.....	37
6.4.4. Full Working Prototype.....	38
6.4.5. Mobile Application.....	38
6.4.6. Website.....	38
6.4.7. Project Poster.....	38
6.4.8. Complete Code in CD.....	38
7. Domain Analysis.....	39
7.1. Microcontroller Board.....	40
7.1.1. About the Boards.....	40
7.1.2. Raspberry Pi.....	40
7.1.3. Choice of on-board computer.....	40
7.2. Data Communication.....	41
7.2.1. About Data Communication.....	41
7.2.2. Wi-Fi.....	42
7.2.3. Bluetooth.....	42

7.2.4. Mobile Data.....	42
7.2.5. Choice of Communication Mode.....	42
7.3. Sensors and other components.....	43
7.3.1. About the Flame Sensor.....	43
7.3.2. GPS Module.....	43
7.4. Server.....	43
7.5. Android Studio to develop Android App.....	44
8. Project Setup.....	45
8.1. Overview.....	46
8.2. Hardware Setup.....	46
8.2.1. Setting up the Raspberry Pi.....	46
8.2.2. Interfacing the sensors.....	50
8.2.3. Setting up the GPS module.....	52
8.2.4. Panic Button Feature.....	53
8.2.5. Fire Detection Feature.....	54
8.3. Android Application Setup.....	55
8.3.1. Android Studio.....	55
8.3.1.1. Setup.....	56
8.3.1.2. Front End.....	56
8.3.1.2.1. Geo-Fencing.....	57
8.3.1.2.2. Over Speed limit.....	61
8.3.1.3. Back End.....	61
9. Risk Analysis.....	63
10. Conclusion.....	65
11. References.....	67

List of Figures and Tables

Table 3.1 Work Division among Group Members.....	18
Table 3.2 Budget for the Project.....	18
Table 3.3 Timeline for the Project.....	19
Image 5.1 Architecture of VMS.....	24
Image 6.1 Raspberry-Pi 3 Model B.....	29
Image 6.2 On-board LM393 voltage comparator chip and infrared sensing probe.....	30
Image 6.3 U-blox NEO-6M GPS Module.....	31
Image 6.4 Push Button.....	33
Image 6.5 5V Mini cooling Fan.....	34
Image 7.1 Pin diagram of the hardware.....	41
Image 8.1 Configuring the Raspberry Pi 3 B.....	47
Image 8.2 Creation of table 'DATA' in the MySQL database in Raspberry Pi.....	48
Image 8.3 Installation of Twilio in the Raspberry Pi 3 B.....	48
Image 8.4 Fire message alert sent from Raspberry Pi to parent mobile device.....	49
Image 8.5 Push Button message alert sent from Raspberry Pi to parent mobile device.....	49
Image 8.6 The GPIO pin diagram for raspberry pi 3B.....	51
Image 8.7 Push Button pressed attached on the Raspberry Pi and Message alert send on Parent's mobile number.....	53
Image 8.8 URL in the message alert shows the current location of the Raspberry Pi on Google Maps.....	53
Image 8.9 Fire sensor tested attached on the Raspberry Pi and Message alert send on Parent's mobile number.....	54
Image 8.10 URL in the message alert shows the current location of the Raspberry Pi on Google Maps.....	54
Image 8.11 Hardware of IoT system setup which contains Raspberry Pi 3 B, GPS module, Push Button, Fire sensor, Cooling Fans, LEDs and Power Bank.....	55
Image 8.12 First view of the Android Application which has name of the app.....	56
Image 8.13 Starting location depicted by Green marker.....	57

Image 8.14 a) Blue polyline trail.....	58
Image 8.14 b) Geo-fence created with trail.....	58
Image 8.15 a) Orange marker for creating geo-fence and menu option on RHS for creating or clearing geo-fence.....	59
Image 8.15 b) Setting geo-fence radius in meters.	59
Image 8.15 c) Geo-fence created w.r.t Orange marker.....	59
Image 8.16 Geo-fence cleared when Clear Geo-fence is clicked on the menu.....	59
Image 8.17 Alert Notification when entering geo-fence.	60
Image 8.18 Alert Notification when exiting geo-fence.....	60
Image 8.19 a) Over-Speed limits view.....	61
Image 8.19 b) Setting speed limit in kmph.....	61
Image 8.19 c) Speed limit set with Alert message.....	61
Image 8.20 Firebase database.....	62

Acknowledgement

We would begin by expressing our gratitude towards our mentor Dr. Debopam Acharya for constant support and inputs. He continually guided us through various phases of this project since its inception. We would further like to thanks our fellow friends for their valuable inputs from time to time. We wish to express our sincere thanks to Anurag Barthwal for taking out time from his busy PhD schedule and guiding us on how to calibrate the sensors. We are indebted to the whole Computer Science department of Shiv Nadar University for providing us with valuable constructive feedback.

We would also take this opportunity to acknowledge our parents for their encouragement, support and attention. We would also like to thank each other for continually motivating the team and working in tandem the course of the project.

Sign: Arpan Mangal

Sign: Bikki Manoj

Sign: Mayank Jain

Acronyms and Abbreviations

Various abbreviations notations and nomenclature used in the report are listed below

- APP: Application
- DB: Database
- ERT`s: Emergency Response Teams
- GPIO: General purpose Input/output
- HTTP: Hypertext Transfer Protocol
- IDE: Integrated Development Environment
- I/P: Input
- IOT: Internet of Things
- JSON: JavaScript Object Notation
- KMPH: Kilometers Per Hour
- Lat: Latitude
- Long: Longitude
- O/P: Output
- OS: Operating System
- Raspberry Pi: Raspberry Pi 3 Model B
- RHS: Right Hand Side
- SQL: Structured Query Language
- UI: User Interface
- URL: Uniform Resource Locator
- VMS: Vehicle Monitoring System
- WiFi: Wireless Fidelity
- WLAN: Wireless Local Area Network
- W.R.T: With Respect to

Chapter 2
Executive Summary
And
Related Works

2. Executive Summary & Related Works

2.1 Executive Summary

Before you hand over the car keys to your teenager, know the facts. Car accidents are the leading cause of teenage deaths. Drivers aged between 16 to 21 are nearly 9 times more likely to have an accident than middle-aged drivers. According to the Road Accident Report for 2014 prepared by the road transport and highways ministry of India, 75,000 youngsters were killed in road crashes in 2014.

The latest estimate by the World Health Organization also shows that globally, world road traffic injuries are the number one cause of death among young people aged 15-29 years and annually, about 3.4 lacks youngsters in this age group die in accidents. Teenagers often underestimate or are unable to recognize hazardous driving conditions. One of the major causes of road deaths is over-speeding.

Therefore, our project's primarily purpose is teenager's safety while driving. She/he should be able to immediately alert the parents and concerned ERT's by pressing the button when in danger. In case of emergency like fire automatic alert should be sent to relevant ERT's. Moreover, parents can also ensure their child's safety by putting a restriction on the speed limit and monitor their location with features like geo-fencing.

2.2 Related Works

Historically, vehicle tracking has been accomplished by installing a box into the vehicle, either self-powered with a battery or wired into the vehicle's power system. Several types of vehicle tracking devices exist. Typically they are classified as "passive" and "active". "Passive" devices store GPS location, speed, heading and sometimes a trigger event such as key on/off, door open/closed that transfer data via wireless download. "Active" devices also collect the same information but usually transmit the data in near-real-time via cellular or satellite networks to a computer or data center for evaluation.

Urban public transit authorities and fleet operators for fleet management functions such as fleet tracking, routing, dispatching, on-board information and security are an increasingly common user of vehicle tracking systems, particularly in large cities. An in-vehicle monitoring system (IVMS) describes any electronic device that is used to track, assess and report on vehicle activity which in turn reflects on the driver's behavior. Its objectives are to reduce incidents on the road and deliver efficiency in road transport operations. Increasingly vehicle manufacturers are fitting new vehicles with this technology as standard and in some countries, it is becoming mandatory.

Some systems that are already in the market are Road Vehicle Monitoring System Based on Intelligent Visual Internet of Things which provides intelligent video analysis, such as the cases of jumping the red light or illegally turning around, so as to improve the efficiency of traffic management. Car parking management and monitoring system (CPMMS) that make the car drivers to park conveniently and quickly as well as monitor the parking space to make the management more standard and orderly.

Chapter 3

Project Management

3. Project Management

This section discusses the team members and their responsibilities, methodology adopted and the various milestones for the duration of the project.

3.1 The Team

The team for this project comprised of three working members and one mentor. All the members add lot to the team in their own right. The development work is equally divided among all the three working members. A greater part of the project was done together, with the required guidance from the mentor.

3.1.1. Group Member 1

Arpan Mangal is a fourth year Computer Science and Engineering student. He was born and raised in Dadri. His technical abilities and knack of picking up technical stuff quickly is essential to the team. His jovial and gregarious attitude brings much needed positive energy during the course of such a demanding project.

3.1.2. Group Member 2

Bikki Manoj is a fourth year Computer Science and Engineering student. He was born in south India from Andhra Pradesh state. He is positive person about every aspect in life. His previous experience with IOT and python coding helped the project to take big leap ahead. His self-made, ambitious and work alcoholic nature helped to achieve the project to its best in limited time.

3.1.3. Group Member 3

Mayank Jain is a fourth year Computer Science and Engineering student. He was born in Patiala, but has been staying in New Delhi from past few years. His ability to grasp new things quickly and good time-management skills have been a big plus for the team. His optimistic and realistic approach towards his work has played an important factor during the project.

3.1.4. Project Mentor

Dr. Debopam Acharya heads the Department of Computer Science and Engineering at Shiv Nadar University. His Research interests are in various domain of Mobile Computing. He is the best person to mentor a project based on an IoT setup. He is a true asset to the team as a focused guide and a motivator. His knowledge adds to the finer learning during the duration of project building.

3.2 Methodology

3.2.1. Software Development Process

We used Agile methodology as the appropriate software development process for the project. The sprints comprised of bi-weekly scrums. These bi-weekly scrums were in coordination with the bi-weekly team meetings. As a result of this, the code development, testing and review occurred twice every week. The members stuck to the process to the best of their ability with certain exceptions. Since the development, testing and review happened simultaneously, the team benefitted in the longer run with many roadblocks being identified at early stages. The work done was incrementally, which allowed the members to review their progress and determine their path for the remainder of the project.

3.2.2. Schedule

The project schedule was well structured. The meeting being of two types: Team meetings and Mentor meetings. The team had scheduled bi-weekly Team meetings and weekly Mentor meetings. Team meetings involved the team members coming together and discussing about the current state of the project and its short-term goals. The team meeting generally consisted of topics related to the immediate requirements of the project. Mentor meetings involved all the team members meeting with the mentor during a designated time at least once every week. Mentor meetings were used to evaluate the steps needed to be taken in the upcoming week and their impact in the longer run. The latter meetings were also useful for the mentor to keep a track over the progress of the students working under him.

3.2.3. Work Breakdown

The Table 3.1 below shows the work breakdown among three group members. Although the work was equally divided and completed as a team, the below table only aims to serve as an indicator to highlight their major contribution respectively.

Name	Architecture	Sensing System	Server	Mobile Application	Testing	Documentation	Hours/Week
Arpan	✓	✓	✓		✓	✓	20
Bikki	✓	✓	✓		✓	✓	20
Mayank	✓			✓	✓	✓	20

Table 3.1 Work Division among Group Members

3.3. Budget

The budget for the project was determined by finding the best price available for each of the components needed for the project to work. These components were chosen depending on the cost and the level of necessity. Some degree of risk analysis was done before buying any of the products or components, some of which has been discussed in the later chapter. Once the components for the project were chosen, the team worked hard to get these components for the best available price and within least time possible. Table 3.2 shows the major component requirements for the project and their cost to the team.

S.No.	PRODUCT	QUANTITY	COST	TOTAL
1.	Raspberry Pi – 3B	1	5000	5000
2.	Flame Sensor(DFR0076)	1	200	200
3.	GPS Module	1	2000	2000
4.	Push button	1	30	30
5.	Power Bank 10000maH	1	1200	1200
6.	Jumper Wire set	1	300	300
7.	3G Dongle	1	1400	1400
8.	Development Machine	1	-	-
				10,130/-

Table 3.2. Budget for the Project

3.4. Milestones

The team used a list of specific tasks throughout the semester as milestones to judge completed progress and future goals. The milestones completed in the semester are laid out in this section in the form of a timeline.

TASKS /Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Explore Topics															
Requirement & Feasibility Study															
Project Architecture															
Mid-Term write-up															
Mid-Term PPT															
Coding															
Implementation															
Testing															
End-Term write-up															

Table 3.3 Timeline for the Project

Listed below were the key tasks that were completed during the stretch of the semester. While the first half milestones revolved around doing necessary background research, setting up hardware and software, the second half milestones were the ones which completed the loop. Greater part of code development was done in the second half with supported documentation.

- Setup Raspberry Pi with Raspbian Jessie OS with desired configuration
- Basic hardware setup
- Feasibility Study
- Necessary background research
- Calibrate the sensors to sense flame e.t.c.
- Mobile App integration
- Sending data from Raspberry Pi to Android App
- Completion of IoT setup with Wi-Fi connectivity through internet dongle
- Refined UI for the Android App
- Testing for fully functional system
- Report Documentation.

Chapter 4

Scope of the Project

4. Scope of the Project:

Project deliverables: Project will provide features like geo-fencing and over-speed limit constraint, integrated in an Android application, panic/emergency button implementation and fire sensor installed on Raspberry pi.

Use: As mentioned earlier, our work can be used by parents to monitor their teenager's driving behavior and location. Our work can also be helpful for car rental companies like Volar Cars, Zoom. They can use features like Geo-Fence and Over-speed limit constraint while they rent out their cars to customers to monitor their location and driving behavior.

Boundary: In vehicle monitoring, we are restricting ourselves to monitor location, speed.

Challenges: Using wheel speed sensor or vehicle speed sensor is not feasible due to budget limitation, therefore we are using GPS module to measure speed. At this point of time monitoring the health of driver is beyond the scope of the project.

Assumptions: We believe ERT's would respond to the message as soon as possible when received. We also assume that the geographical location and speed measured by the GPS module is in acceptance range. Raspberry pi is always connected to Internet.

Future Scope: Future work would include driver health monitoring like monitoring heart-beat, pulse, and sleeping and alcohol detection. To have smooth and safe driving experience during damp, wet, icy conditions we can deploy road weather and surface condition sensors which would monitor the atmospheric conditions near roadways, as well as actual road-surface conditions.

Chapter 5

Project Description

&

Architecture

5. Project Description

In order to get a focused approach for development of the project, a basic scope for the project was decided between the mentor and the working members. This was essential to get the minimum expectations from the project.

5.1 Architecture

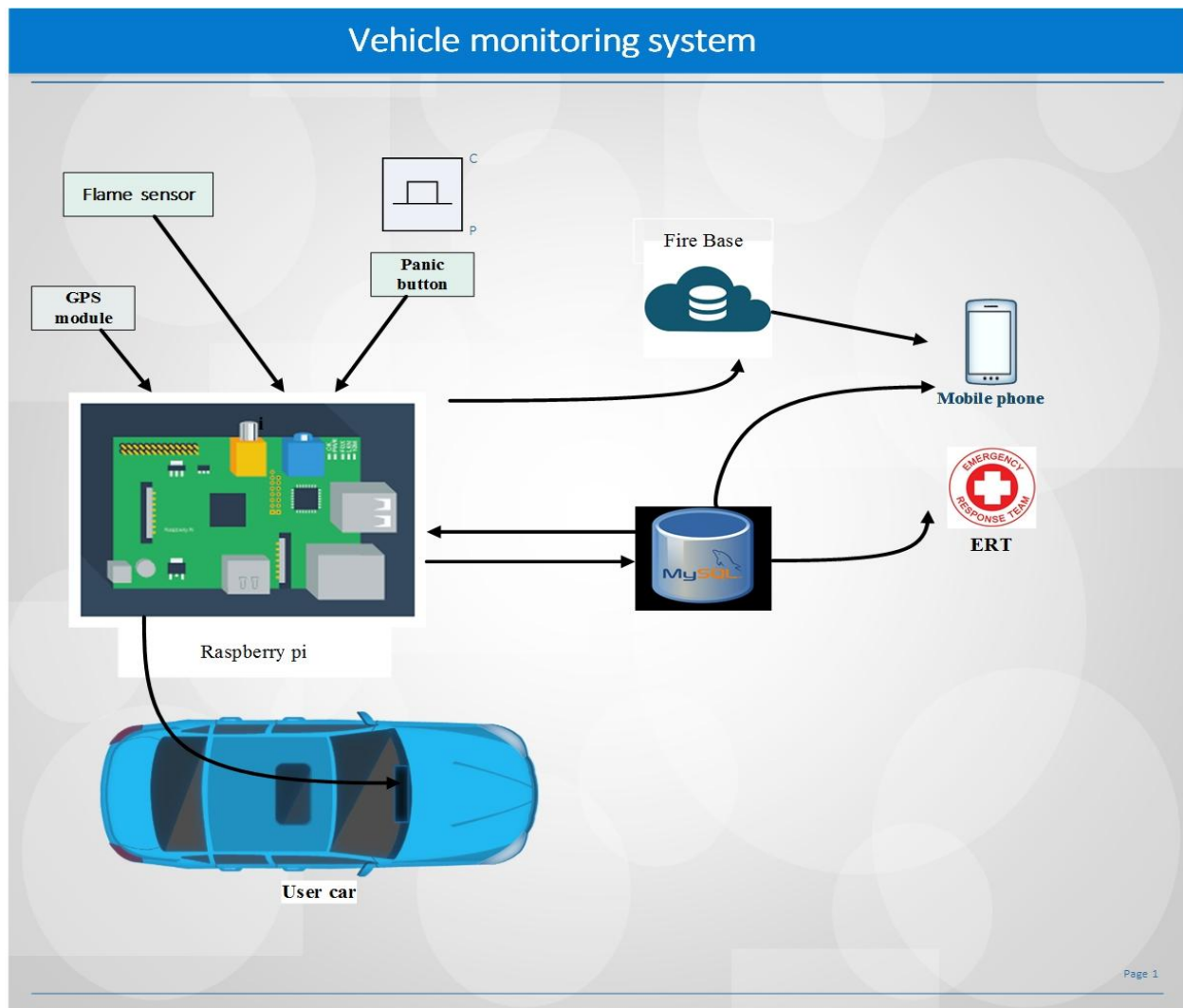


Fig 5.1 Architecture of VMS

The design of our project can be divided into three major components:

5.2 Sensing System

In this part, the main purpose is to collect data from vehicle using the respective sensors and modules. These sensors and modules will be connected to Raspberry pi which is powered through power bank and works on Raspbian Jessie Linux operating system. Since we are focusing on less power consumption and more performance device so raspberry pi is the suitable microcontroller for our implementation. The detailed list of hardware needed is as follows:

a) Flame Sensor (DFR0076): The flame sensor can be used to detect fire or other wavelength at 760 nm ~ 1100 nm light. Detection range of this sensor is 20 cm-100cm.

b) GPS module (EM-406a): Device which gives the geographical location in terms of latitude, longitude and speed of the vehicle from the GPS satellite. Its accuracy in position is 10 meters. It can operate at temperatures from -40 C to 85 C.

c) Push button: Panic/Emergency button, output is high when button is pressed.

5.3 Server

All the data collected from the sensors will be saved and updated regularly in the MySQL database in the Raspberry pi locally. In case of emergency like push button pressed or fire detected, the message alert will be send on parent's mobile number and corresponding ERT with the current location of Raspberry pi. We have used Twilio as a third party web service to provide the SMS service. To avail this service, user need to have a Twilio account. There is a 60 days free trial available for Twilio which we have used in our project. If we wish to continue use this service we need to have membership for it.

All real-time data like location, speed collected from the Raspberry Pi is pushed to the Google Firebase server via Internet over Wi-Fi. Data is periodically every 30 seconds collected and analyzed, with corresponding changed values are sent to the mobile application.

5.4 Mobile Application

After all the necessary analytics are performed on the server, alert message and relevant information is sent to an Android application present in the mobile device of the parent`s.

The mobile device includes functionalities like:

5.4.1 Panic Button and Fire Detection

Alert message with geographical location on Google Maps when the panic button is pressed or fire is detected.

5.4.2 Android Application

Geo-fencing: Monitoring the location of vehicle and creating a virtual geographic boundary, enabling application to trigger a response when a vehicle leaves or enters the particular area.

Over-speed limit: Setting a permissible speed limit on the vehicle, enabling application to trigger a response when the vehicle crosses the particular speed limit.

Chapter 6

Technical Requirements

6. Technical Requirements

6.1 Performance Requirements

The Project's performance requirements determine how well it will function. The project functionality is designed taking into account the desired usability by having required performance measures as well as avoiding being overly expensive.

6.1.1. Sensing Requirements

Mapping to the project's requirements, one of the essential needs of the project is to sense the flames of fire in the car. Other than that we deployed GPS module to give us latitude and longitude of the current location of the car. GPS module also determines the speed of the vehicle. These sensors selected were based on the basis that sensors needed to be powerful enough to sense within acceptable error range. Also they required to be compatible with Raspberry Pi setup without being overly expensive. Also the project required the current supply needed by the sensors as they would be powered by the mobile Raspberry Pi 3B itself.

6.1.2. Mobility Requirements

The project required mobility inbuilt as one of the major traits. All the necessary equipments were purchased keeping the need of mobility in mind. Since this is intended to be placed at the car's dashboard, it requires to be small enough to be placed there without much difficulty. Since, it is a mobile setup; it is also powered by 10400maH power bank. The setup also supports a Wi-Fi module in order to let the sensor values to be uploaded on the server. One can use Wi-Fi dongle in case there's no internet connectivity.

6.2 Hardware Requirements

The Project's system needs to be robust enough to work for prolonged hours. The various part of the system needs to be able to interact with other parts of the required technology. Each sensor must adhere to certain minimum specifications mentioned. A high level diagram of the IoT

setup, which displays all the majors I/Ps and O/Ps and also highlight how they will interact with each other, is shown in figure below.

6.2.1. Raspberry-Pi Requirements

The Raspberry-Pi is a single on-board computer, which is quite popular among the youth. It has one of the cheapest prices for the single-board computer with adequate memory and processing power. In addition to its low price, Raspberry-Pi has the largest support forum of any single-board computer on the market due to its open-source coding and wide fan base in the do-it-yourself market. This contributes to a great deal of support to troubleshooting and coding. It is 1.2 GHz 64-bit quad-core ARMv8 CPU, 802.11n Wireless LAN, Bluetooth 4.1, 1 GB RAM, 4 USB ports, 40 GPIO pins, Full HDMI port, Ethernet port.



IMAGE 6.1 Raspberry-Pi 3 Model B

Image taken from <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

6.2.2. Sensing Requirements

The various sensors like flame sensor, with the necessary information, used in the setup are listed below.

6.2.2.1. Fire/Flame Sensor

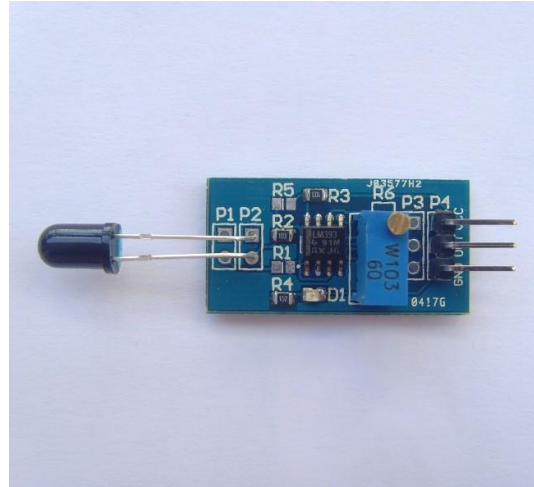


Image 6.2 On-board LM393 voltage comparator chip and infrared sensing probe

Image taken from <http://www.banggood.com/LM393-760nm-1100nm-IR-Infrared-Flame-Sensor-Module-Board-For-Arduino-p-918444.html>

General Description:

Flame sensor is the most sensitive to ordinary light that is why its reaction is generally used as flame alarm purposes. This module can detect flame or wavelength in 760 nm to 1100 nm range of light source. Small plate output interface can and single-chip can be directly connected to the microcomputer IO port. The sensor and flame should keep a certain distance to avoid high temperature damage to the sensor. The shortest test distance is 80 cm, if the flame is bigger, test it with farther distance. The detection angle is 60 degrees so the flame spectrum is especially sensitive. The detection angle is 60 degrees so the flame spectrum is especially sensitive.

Specifications:

- On-board LM393 voltage comparator chip and infrared sensing probe.
- Support 5V/3.3V voltage input.
- On-board signal output indication, output effective signal is high level, and the same time the indicator light up, output signal can directly connect with microcontroller IO.
- Signal detection sensitivity can be adjusted.

- Reserved a line voltage compare circuit (P3 is leaded out).
- PCB size: 30(mm) x15(mm).

Pin Configuration:

1. VCC
2. Output
3. Ground

6.2.2.2. GPS Module

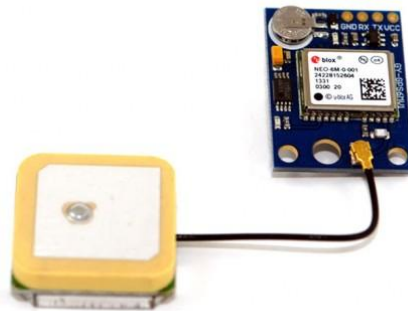


Image 6.3 U-blox NEO-6M GPS Module

Image taken from <https://developer.mbed.org/users/edodm85/notebook/gps-u-blox-neo-6m/>

Overview

The NEO-6 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 6 positioning engine. These flexible and cost effective receivers offer numerous connectivity options in a miniature 16 x 12.2 x 2.4 mm package. Their compact architecture and power and memory options make NEO-6 modules ideal for battery operated mobile devices with very strict cost and space constraints.

The 50-channel u-blox 6 positioning engine boasts a Time-To-First-Fix (TTFF) of less than 1 second. The dedicated acquisition engine, with 2 million correlators, is capable of massive parallel time/frequency space searches, enabling it to find satellites instantly. Innovative design and technology suppresses jamming sources and mitigates multipath effects, giving NEO-6 GPS receivers excellent navigation performance even in the most challenging environments.

GPS NEO 6M Features:

- Standalone GPS receiver
- U-blox NEO-6M GPS module
- Under 1 second time-to-first-fix for hot and aided starts
- Super Sense ® Indoor GPS: -162 dBm tracking sensitivity
- Anti-jamming technology
- Support SBAS (WAAS, EGNOS, MSAS, GAGAN)
- u-blox 6 50 channel positioning engine with over 2 million effective correlators
- Time pulse
- 5Hz position update rate
- Operating temperature range: -40 TO 85°C
- UART TTL socket
- EEPROM to store settings
- Rechargeable battery for Backup
- Build in 18X18mm GPS antenna
- RoHS compliant

Specifications:

Dimension: 22mmX30mm

Height: 13mm

Hole dia.: 3mm

Weight: 12g

6.2.2.3. Push Button

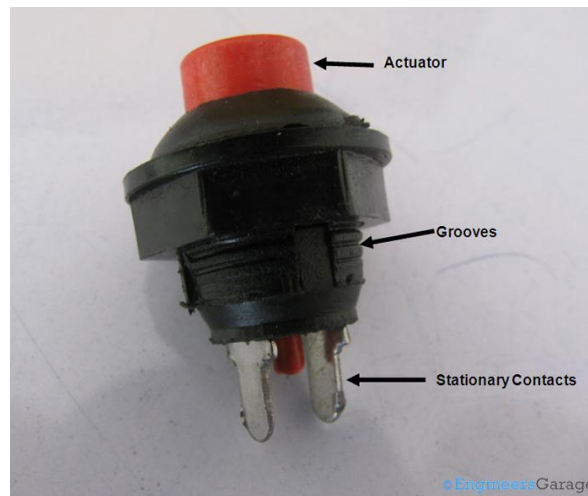


Image 6.4 Push Button
Image taken from <https://en.wikipedia.org/wiki/Push-button>

Type: Momentary

- Ergonomic design
- Easy-to-use
- Event annotation

A common need when recording bio-signals is the annotation of meaningful events that occur during the recording session, synchronously with the bio-signal data. This ergonomic handheld manual button was specifically designed for that purpose.

6.2.2.4. Cooling Fan



Image 6.5 5V Mini cooling Fan

Image taken from <https://thepihut.com/products/adafruit-miniature-5v-cooling-fan-for-raspberry-pi-and-other-computers>

General Description:

This 5V Mini Cooling Fan prevents our hard-working Pi from overheating. It's also great for use with any small computer or FPGA or motor driver or anything that needs cooling. We just need to plug the fan directly into your Raspberry Pi's 5V+GND GPIO power pins for instant cooling. We can also place it on top of our Raspberry-Pi to give it extra air-flow.

It is compatible with any and all Raspberry Pi computers but only the Pi 3 runs hot enough to benefit from any cooling, when doing intense computations. We need a fan for our Pi, as it will automatically adjust its speed to avoid overheating.

Specifications:

- Operating voltage: 5V
- Current: 0.2 A
- Brushless DC fan
- Fan dimensions: 30mm x 30mm x 8mm
- Wire length: 3.25" / 80mm
- Fan weight: 6.2g / 0.22oz

6.2.3. Communication Requirements

In order to upload data on the Firebase server, the setup can make use of the Wi-Fi as well as Ethernet as Raspberry Pi 3 B comes inbuilt with the Ethernet port and Wi-Fi module. Apart from Wi-Fi connectivity, the setup also enables the user to use a 3G dongle when not near the Wi-Fi source for ease of use. This will let the user collect data almost anywhere where he/she likes to go in daily life.

6.2.4. Other Electronic Requirements

Other electronic/hardware requirements include a portable power bank. This is the easiest way to provide our setup the power for its functioning. A couple of development machine/ laptop would also be required for the coding and testing the system. We would also require a vehicle for testing. Also a Smartphone running on android would complete the major hardware requirement for the project.

6.3 Tools and Software Requirements

The scope of the project requires working across multiple platforms. Building an IoT system would involve interacting between different platforms. The development of this project would require the developers to have a basic understanding of IoT as a concept. The most pivotal part of the whole setup will be the various raspberry pi(s) which serve as the main source of data collection. Below is the list of technologies which are being used to bring the project into life.

6.3.1. Python

This language is used for reading the GPIO pins of the Raspberry Pi. It is used to deliver the meaningful data to the terminal by interpreting the data from the sensors. Python was chosen as it was fast compared to the Java SE and more readily available Raspberry Pi resources for it.

6.3.2. Firebase

We have used Google Firebase to upload real time data. Firebase is a mobile and web application platform with tools and infrastructure designed to help developers build high-quality apps. Firebase is made up of complementary features that developers can mix-and-match to fit their needs. We have used Firebase as real time database that allows us to store and synchronize data.

6.3.3. Twilio

We have used Twilio to send and receive text messages. Twilio is a cloud communications platform as a service (PaaS) company based in San Francisco, California. Twilio allows software developers to programmatically make and receive phone calls and send and receive text messages using its web service APIs. Twilio's web services are accessed over HTTP and are billed based on usage.

6.3.4. MySQL

We have used MySQL open-source relational database management system (RDBMS). We have install MySQL in Raspberry pi to store sensor data locally. When Raspberry pi senses the data, we update the data in the MySQL database as well as update it on Firebase server.

6.3.5. Android Studio

The application for the android Smartphone will be developed using Android Studio. Android Studio is the official IDE available for the platform. It makes android development a little easier with an interactive environment.

6.3.6. Microsoft Visio 2016

Microsoft Visio is a diagramming and vector graphics application and is part of the Microsoft Office family. It is used for many things that utilize layouts, diagrams and architecture.

6.3.7. Fritzing

Fritzing is an open source initiative to develop amateur or hobby CAD software for the design of electronics hardware, to support designers and artists ready to move from experimenting with a prototype to building a more permanent circuit. We have used Fritzing to make pin diagram of the setup.

6.4 Deliverables

Since this project serves as the Final Semester Project for the completion of Bachelors of Technology degree from Shiv Nadar University, there are number of hardware and software deliverables that are due to faculty panel and project advisor for grading.

6.4.1. Basic Setup

A basic hardware setup complete with all the sensors and LEDs able to write the data into MySQL database and upload it on Firebase server.

6.4.2. Software Deployed in the Raspberry Pi 3

A fully functional Raspberry Pi 3 with the latest Raspbian OS installed. All the necessary configurations should be complete and tested.

6.4.3. Documentation

A project report stating in detail about the project. The project report contains all the necessary details required for setting up the project hardware. It will be detailed script about the project development.

6.4.4. Full Working Prototype

A fully functional setup for the IoT system with the necessary hardware and software support.

6.4.5. Mobile Application

An Android application for the user to monitor the Raspberry pi attached on the vehicle. User will be able to track the location of the car. User will receive message alert in case of emergency. User will be able to set the speed limit and complete boundary. User will be able to see the complete trail the vehicle has covered.

6.4.6. Website

Complete information about the IoT system is uploaded on the website.

Link: <https://bm5995.wixsite.com/vms-project>

6.4.7. Project Poster

A final poster to the panelist shall also be submitted summarizing our project details.

6.4.8. Complete Code in CD

A completed and in-depth code files which have been used in the entire project.

Chapter 7

Domain Analysis

7. Domain Analysis

The system to be developed in the project had various sub-parts needed to be acquired or used. The team researched and did domain analysis on the available technologies that needed to be acquired or used.

7.1 Microcontroller Board

7.1.1. About the Boards

An on-board computer was necessary in order to run the sensors and push these values to the server. Apart from sending these values to the server, the system must also files to run various sensors. Since the project required a single board computer with GPIO pins to interact with the sensors, most popular single board computers in the market were researched.

7.1.2. Raspberry Pi

The Raspberry Pi has one of the biggest communities online for projects involving sensors and home automation. It is the most popular single-board computer on the market. For its high memory capacity and huge processing power, it is relatively low priced. It's wonderful support forums and huge fans base makes it one of the most versatile computers.

7.1.3. Choice of On-Board Computer

The Raspberry Pi 3 B was chosen for use in the project. The decision was made for its cheap price and the processing power and networking capabilities it offers. Also, the raspberry pi was readily available, hence the team started to work on the project immediately.

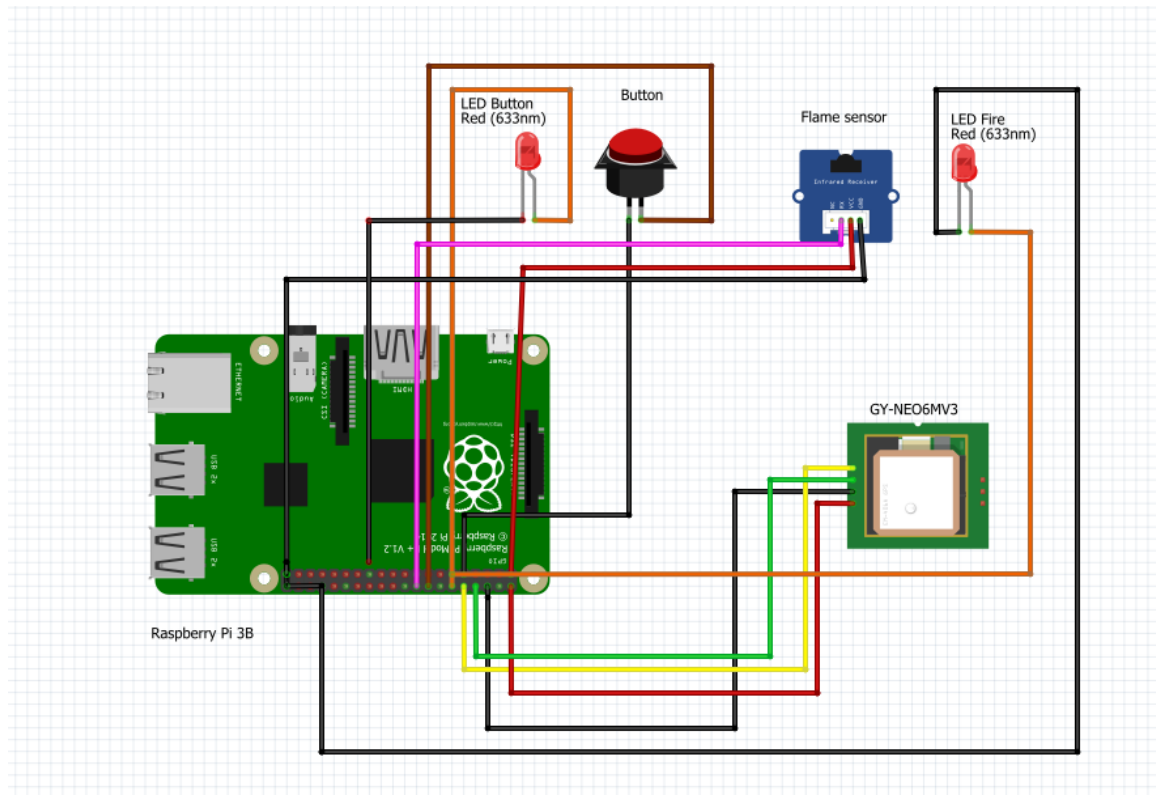


Image 7.1 Pin diagram of the hardware

7.2 Data Communication

7.2.1. About Data Communication

The system developed in the project would be equipped with fire sensor and push button and would be providing meaningful values to the server. In order to establish this remote communication, a pre-existing wireless technology must be chosen. In order to choose the perfect technology needed for the project, the team decided to do research on the available technologies. The team initially decided that the technology used must be wireless as the system developed would be used at a varying distance from the server.

7.2.2. Wi-Fi

‘Wireless Fidelity’ was the first option that occurred in the team’s mind. It is one of the most used technologies. Transmitting data through Wi-Fi is fast, reliable and secure. Wi-Fi offers almost unlimited range as once the system is connected to a router, it can both transmit data to and receive data from almost any place on earth.

7.2.3. Bluetooth

The next immediate wireless technology that the team thought of was Bluetooth. We planned to send data directly to Android Smartphone instead of server and do all the analysis on Raspberry Pi itself. Bluetooth is both fast and easy to setup. The drawback with this technology though is that it does not have the range a Wi-Fi router has and does not have pre-existing intranet setup for its immediate implementation.

7.2.4. Mobile Data

Mobile Data through CDMA/GSM accessed by dongle was considered by the team in order to provide the user the ease of uploading data when not near Wi-Fi source. This would let the data submitted be real time instead of collecting and then uploading the data later.

7.2.5. Choice of Communication Mode

With the above research done in the respective fields, the team chose Wi-Fi and 3G dongle as the appropriate means of remote communication between the system and the server. The campus has a pre-existing setup of WLAN by the name “Student” accessible to the team. This was chosen as the initial Wi-Fi on which the setup would be run. Later the team might shift the project to a safe and private WLAN. The team also tested a 3G dongle to connect to the internet successfully.

7.3 Sensors and Other Components

7.3.1. About the Flame Sensor

Sensing fire inside the vehicle was one of the project's concern. In order for the system to be developed in the project to run smoothly and efficiently, the technology used in sensing must be wisely chosen. The team eventually had to decide between whether to choose an all in one sensor which would sense both the flames and smoke, or use separate sensors for both. Then team decided not to put smoke sensor, because it will unnecessary produce alert if someone is smoking inside vehicle with choice. Therefore, team finally decided to have only flame sensor on board.

7.3.2. GPS Module

Getting the current location of the vehicle was one of the projects prime concerns. One of the ways to get the location is from driver's mobile device. But it requires driver mobile phone to be nearby Raspberry Pi always. Hence, the team decided to buy separate GPS module to track the location of the vehicle. The GPS NEO 6M used in the project was compatible with the Raspberry Pi. We were able to install it with ease. The team was also satisfied with the results given by GPS module. Latitude, Longitude and Speed given by the GPS module were in the acceptable range.

7.4 Server

The team decided to use Google Firebase as a server. This choice was made because Google Firebase is open source and easy to use. The project doesn't require heavy analytics at the server side, therefore it was not worth to make separate machine as a server. Secondly, Google Firebase server is available 24 x 7 to use. The delay incurred in the communication was also acceptable.

7.5 Android Studio to develop Android App

The team decided to use Android Studio (version 2.2.3) as a software tool to develop the parental control android application used in the project. We have used Android studio 2.2.3, as it is open source and free to use, we have other option to use like Eclipse IDE to make android application but working with Eclipse can be difficult at times, while debugging and designing layouts, Eclipse sometimes get stuck and we have to restart Eclipse from time to time. More over there is a vast community of android studio resources available on internet. One of the numerous advantages of using Android Studio is we can use customized android Emulator. Hence android studio is perfect for rapid application development.

Chapter 8

Project Setup

8. Project Setup

8.1 Overview

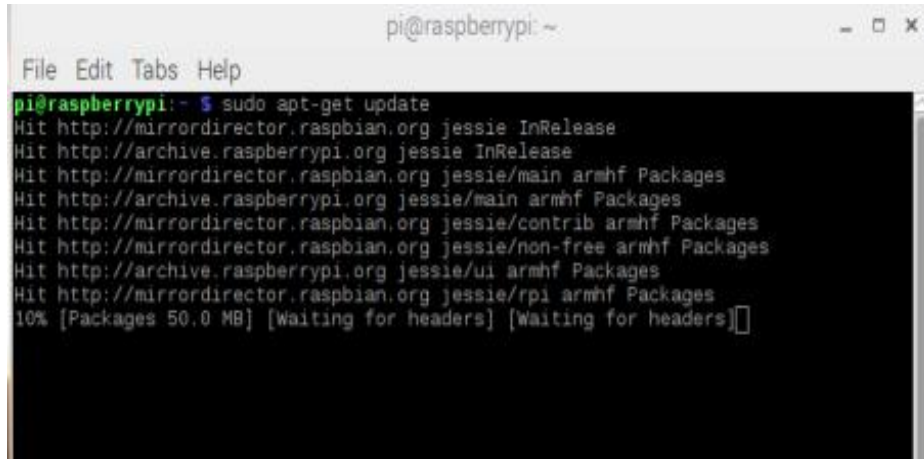
The chapter discusses the project setup in great detail. It consists of two major sections, namely Hardware Setup and Android Application Setup. The Hardware Setup section discusses how all the aforementioned components are made use of in the project. It also depicts the pin diagrams and explains the significance of each of the connections made in the hardware setup. It depicts how message alert will generate in case of emergency like fire or push button pressed.

The Android Application Setup section discusses the software architecture of the system. It highlights how data is transferred from the Raspberry Pi to the Firebase Server and then from server to the android app. It highlights the main features of the app i.e. Over-Speed limit constraint and Geo-Fencing. It depicts how the app will generate message alerts when vehicle crosses the boundary or speed limit.

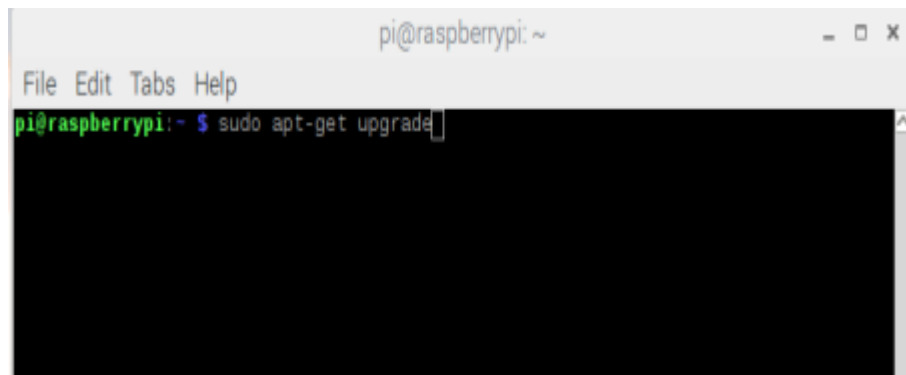
8.2 Hardware Setup

8.2.1. Setting up the Raspberry Pi

- I. **Operating System:** Before reading the sensors, the Raspberry Pi 3B has to be configured. A system OS needs to be installed on the microSD card that gets plugged into the Pi. The system OS used for this project's development was the Raspbian Jessie 2017 OS, which needs to be timely updated. Users can directly flash the microSD card with the OS image file or use a software called NOOBS which directly downloads and installs the selected OS on booting up the raspberry pi for the first time. Make sure to update and upgrade the system on installation by using “`sudo apt-get update`” & “`sudo apt-get upgrade`” commands on the terminal. This will require internet connectivity, so make sure to use an Ethernet cable.



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo apt-get update
Hit http://mirrordirector.raspbian.org jessie InRelease
Hit http://archive.raspberrypi.org jessie InRelease
Hit http://mirrordirector.raspbian.org jessie/main armhf Packages
Hit http://archive.raspberrypi.org jessie/main armhf Packages
Hit http://mirrordirector.raspbian.org jessie/contrib armhf Packages
Hit http://mirrordirector.raspbian.org jessie/non-free armhf Packages
Hit http://archive.raspberrypi.org jessie/ui armhf Packages
Hit http://mirrordirector.raspbian.org jessie/rpi armhf Packages
10% [Packages 50.0 MB] [Waiting for headers] [Waiting for headers]
```



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~ $ sudo apt-get upgrade
```

Image 8.1 Configuring the Raspberry Pi 3 B

- II. **Python Libraries:** python IDE by default available in Raspbian Jessie. We were required to install the required libraries to use its functionalities. For connecting python to firebase we installed the fire base module “ sudo pip install python -firebase ” & “ sudo pip install requests==1.1.0”. For connecting python to MYSQL we installed the required modules by using “sudo python setup.py install”. For connecting python to Twilio we installed the required modules using “sudo pip install twilio”.
- III. **MySQL:** To save the sensor data locally in the raspberry pi we planned to make Mysql data base in raspberry pi itself. We created a database called VMS in it. It contains the table DATA which has fields like BUTTON, FIRE, LATITUDE, LONGITUDE, SPEED. The following are commands to do create the database.

- Create database VMS ;
- Use VMS;
- Create table DATA (BUTTON int , FIRE int ,LATITUDE float , LONGITUDE float , SPEED float);

```

pi@raspberrypi: ~
File Edit Tabs Help
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use VMS;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> describe DATA;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| BUTTON     | int(11)       | NO   |     | NULL    |       |
| FIRE       | int(11)       | NO   |     | NULL    |       |
| LATITUDE   | varchar(25)   | YES  |     | NULL    |       |
| LONGITUDE  | varchar(25)   | YES  |     | NULL    |       |
| SPEED      | varchar(25)   | YES  |     | NULL    |       |
| S_NO       | int(11)       | NO   | PRI | 0       |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>

```

Image 8.2 Creation of table 'DATA' in the MySQL database in Raspberry Pi

IV. Twilio: To send an alert messages in case of emergency like fire, push button pressed we took help of third party i.e. twilio. With the help of twilio we can send messages directly from raspberry pi to user's mobile phone. For this users need to have an account on twilio. Initially for 2 months we can go for free trail as well.

```

pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ pip install twilio

```

Image 8.3 Installation of Twilio in the Raspberry Pi 3 B

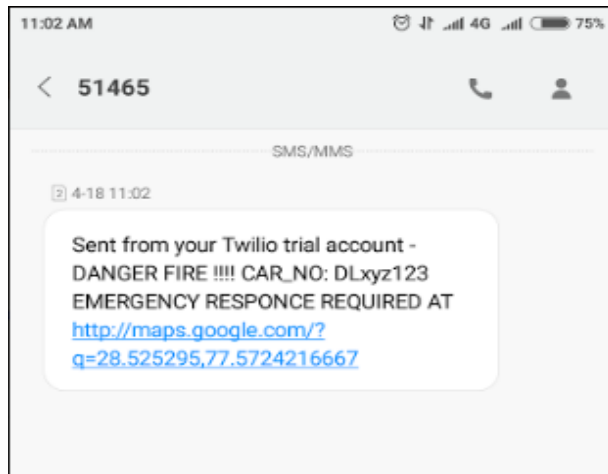


Image 8.4 Fire message alert sent from Raspberry Pi to parent mobile device

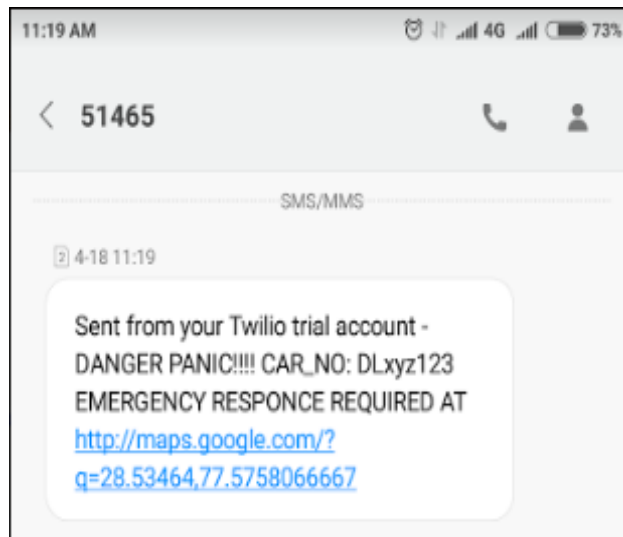


Image 8.5 Push Button message alert sent from Raspberry Pi to parent mobile device

8.2.2. Interfacing the Sensors

The hardware setup makes use of flame sensor, push button which gives output in digital and the GPS module give data string using UART pins present in raspberry pi. All the sensors require their connections to be made to the raspberry pi.

- I.** The 5V GPIO pins of the raspberry pi are used to supply VCC to the sensors. Raspberry pi 3 has 2 5V sources, at pin 2 and pin 4. Refer to the image 8.1 for your reference.
- II.** The GND pins of the raspberry pi are used to ground the sensors. Raspberry pi 3 has several ground pins which can be used as per user's convenience.
- III.** The third connection required is the one used by the sensor to transmit data to the Raspberry pi 3B. Since the GPIO pins of raspberry pi 3B reads only digitally, the data pin of the flame sensor can be connected to any of the GPIO pins directly. Only thing to be taken care of here is that the pin number used in the sensor libraries to read the data should mention the logical GPIO pin number and not the physical pin. The logical pin numbers are mentioned in the rectangular boxes adjacent to the pins in the image 8.1.

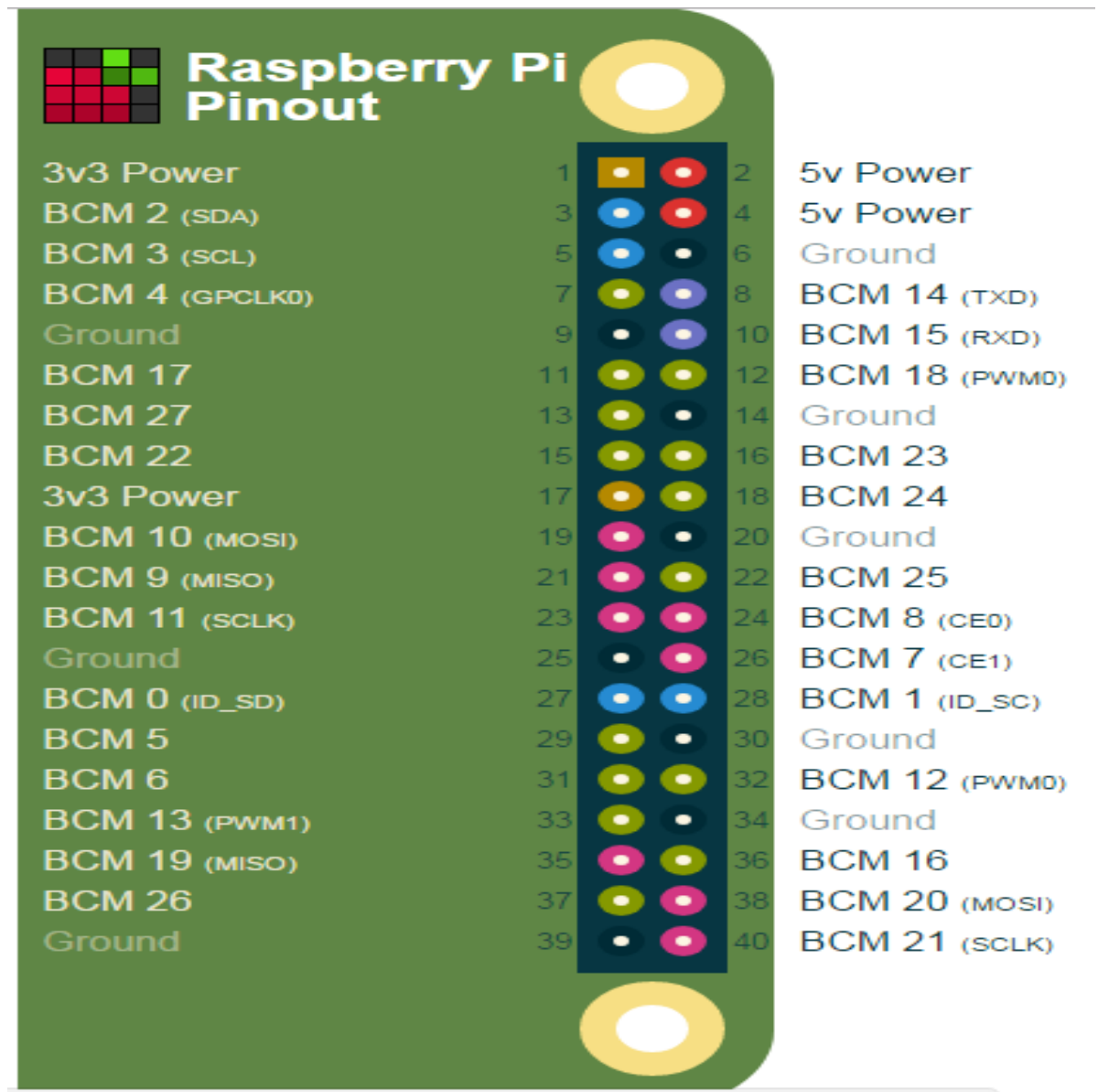


Image 8.6 The GPIO pin diagram for raspberry pi 3B.

Image taken from <https://pinout.xyz/>

8.2.3. Setting up the GPS Module

To make the data location aware the GPS module was needed. The GPS NEO 6M module fits into our Internet Of Things setup pretty conveniently. It can be connected to the Raspberry pi via TXD and RXD pin in Raspberry pi.

Installing GPS daemon

- `sudo apt-get install gpsd gpsd-clients cmake subversion build-essential espeak freeglut3-dev imagemagick libdbus-1-dev libdbus-glib-1-dev libdevil-dev libfontconfig1-dev libfreetype6-dev libfribidi-dev libgarmin-dev libglc-dev libgps-dev libgtk2.0-dev libimlib2-dev libpq-dev libqt4-dev libqtwebkit-dev librsvg2-bin libsdl-image1.2-dev libspeechd-dev libxml2-dev ttf-liberation`
- `sudo nano /boot/config.txt` add `core_freq=250 ,enable_uart=1`
- `sudo nano /boot/cmdline.txt` replace with this `dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait`

Run

- `sudo systemctl stop serial-getty@ttyS0.service`
- `sudo systemctl disable serial-getty@ttyS0.service`
- `sudo systemctl stop gpsd.socket`
- `sudo systemctl disable gpsd.socket`

Reboot & Execute the daemon reset

- `sudo killall gpsd`
- `sudo gpsd /dev/ttyS0 -F /var/run/gpsd.sock`

Test GPS NMEA data with

- `cat /dev/ttyS0` or
- `gpsmon /dev/ttyS0` or with `sudo cgps -s`

8.2.4. Panic Button Feature

As a safety feature we have installed a push button to the Raspberry pi. In case of emergency, it helps user to send message alerts to parent's mobile device and corresponding ERTs. This feature is kept thinking that even if user is little bit conscious in case of accident, he/she would be able immediately alert the parents and ERT. Since Raspberry pi is kept in front of driver on vehicle dashboard, user will be able to press the button straight away in case of danger.

The message alert sent on parent's mobile device and ERT will contain car number and current vehicle location in form of URL which will take us to the Google Maps showing the vehicle current location.

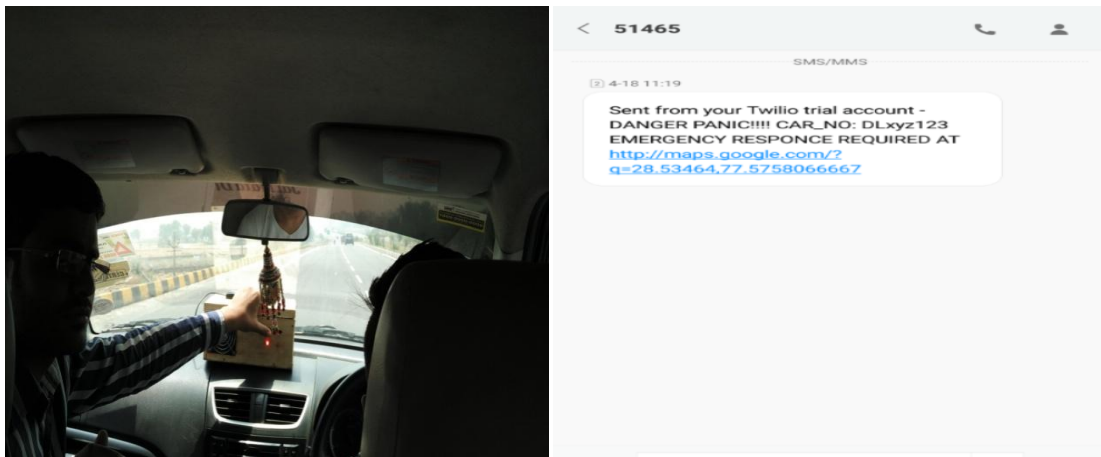


Image 8.7 Push Button pressed attached on the Raspberry Pi and Message alert send on Parent's mobile number

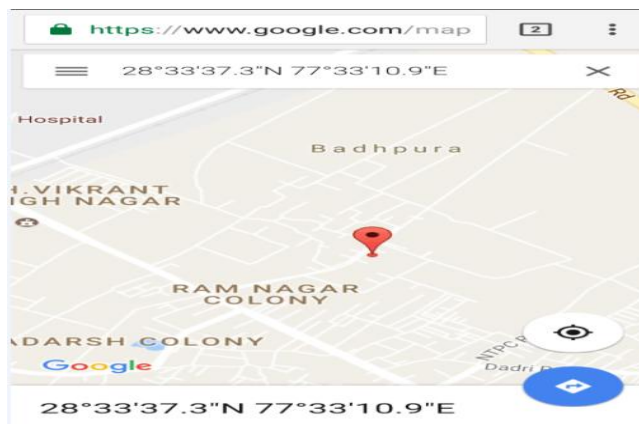


Image 8.8 URL in the message alert shows the current location of the Raspberry Pi on Google Maps.

8.2.5. Fire Detection Feature

As a safety feature we have installed a flame sensor to the Raspberry pi. It will detect the flames near Raspberry Pi. Sometimes driver is unable to detect fire in early stages, that's why we kept this safety feature. When there is a fire around the Raspberry Pi, LED attached on Raspberry Pi will start blinking alerting the driver to move out of the vehicle.

Also, the message alert will be sent on parent's mobile device and ERT will contain car number and current vehicle location in form of URL which will take us to the Google Maps showing the vehicle current location.

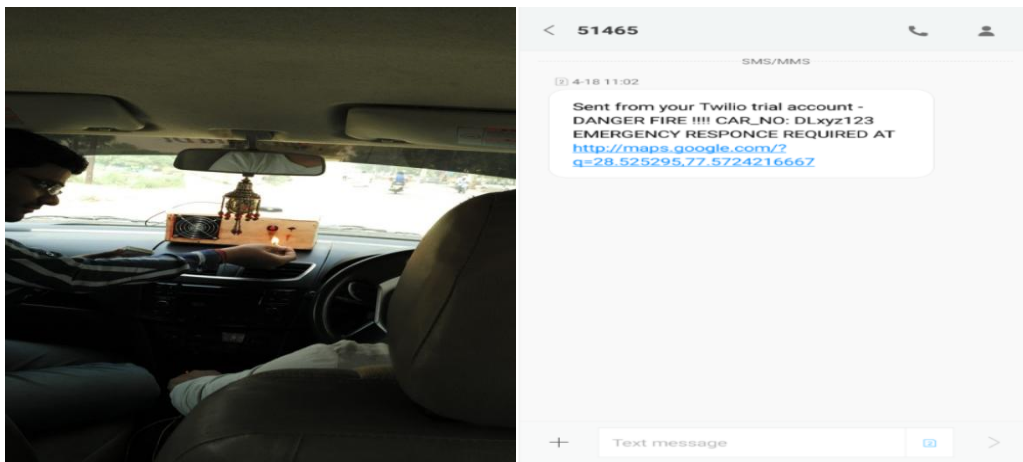


Image 8.9 Fire sensor tested attached on the Raspberry Pi and Message alert send on Parent's mobile number

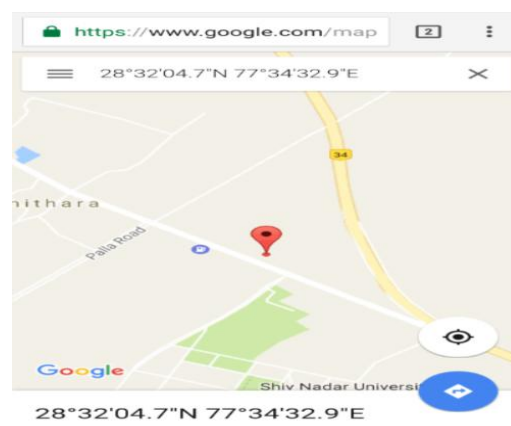


Image 8.10 URL in the message alert shows the current location of the Raspberry Pi on Google Maps.

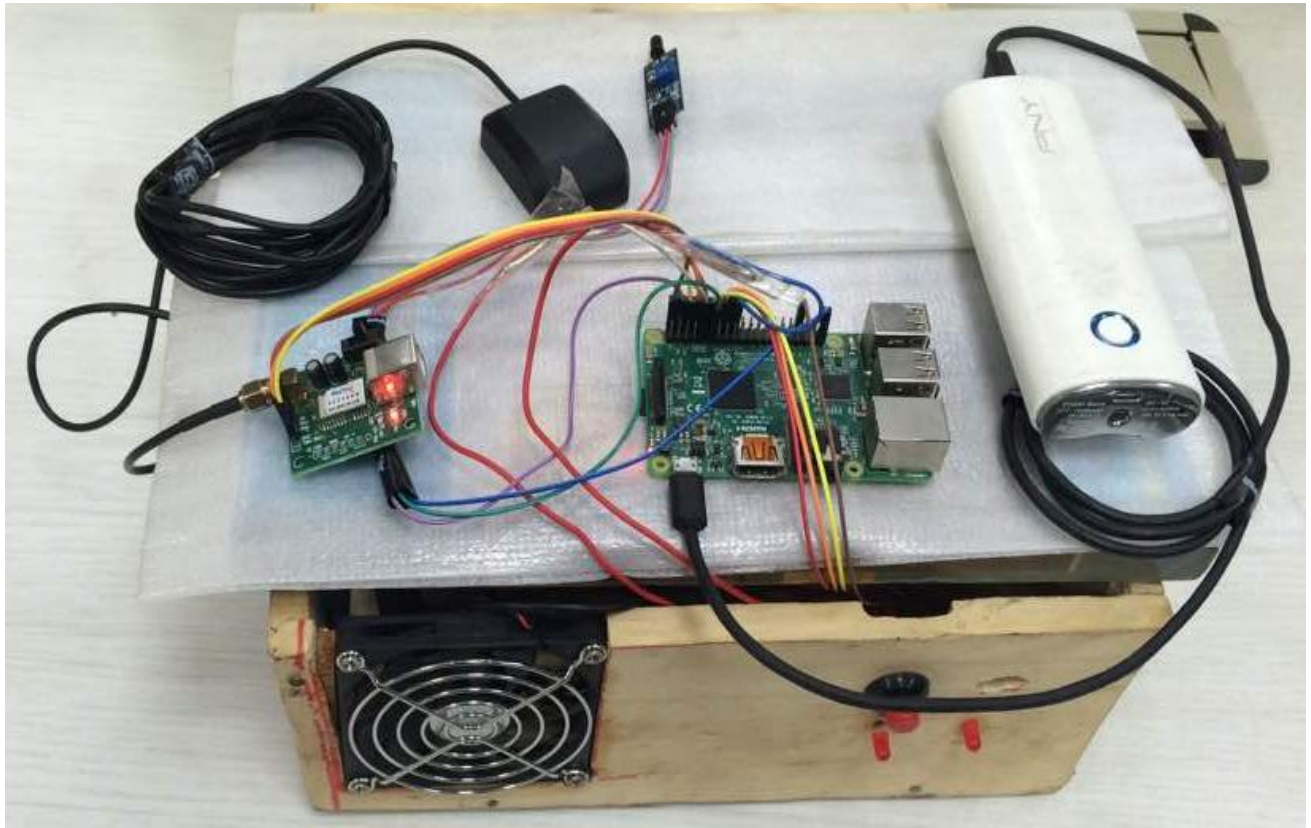


Image 8.11 Hardware of IoT system setup which contains Raspberry Pi 3 B, GPS module, Push Button, Fire sensor, Cooling Fans, LEDs and Power Bank

8.3 Android Application Setup

8.3.1. Android Studio

We have used Android studio 2.2.3, as it is open source and free to use, we have other option to use like Eclipse IDE to make android application but working with Eclipse can be difficult at times, while debugging and designing layouts, Eclipse sometimes get stuck and we have to restart Eclipse from time to time. More over there is a vast community of android studio resources available on internet. One of the numerous advantages of using Android Studio is we can use customized android Emulator. Hence android studio is perfect for rapid application development.

8.3.1.1. Setup

The installation of android studio requires Android Sdk and gradle to be installed on the machine before use. For this we have installed Sdk android 7.1.1 (Nougat), API level- 25. The gradle used for building android studio is gradle:2.2.3. We have used minSdkVersion 19, as it will run on approximately 73.9% devices that are active on the Google Play store. We are using targetSdkVersion 22 so that run time permissions are not asked when running application. Our app also requires Google play-services installed before using application, for this we have used latest Google play-services:10.2.1. The reason to use Google play-services is to access the Google maps. We also require firebase to be configured on our android studio. To configure firebase we have to follow the following steps: go to Tools -> Firebase -> Realtime Database -> Connect your app to Firebase -> Add the Realtime Database to your app -> Write to and Read from respectively from your database.

8.3.1.2. Front end

The Parental Control android application has two features Geo-fencing and Over-speed limit constraint which can be found in the navigation bar of our application.



Image 8.12 First view of the Android Application which has name of the app and Navigation bar

8.3.1.2.1. Geo-fencing

The first view is the Geo-fencing feature which has the map view. At the top of the page there are two text fields in a single row showing the 'Lat' and 'Long' of the current Raspberry pi location. Below the text fields we have an input field and 'FIND' button where user can enter the location of the geo-fence marker which will take us to the corresponding location when the 'FIND' button is pressed. The rest of the page is filled by Google maps where the 'green' marker shows the initial starting location of the Raspberry pi in the vehicle. After every 30 seconds location of the Raspberry pi in the vehicle is updated on Google maps which are depicted by a 'red' marker. A Blue Polyline path is created depicting a trail between the last two markers.

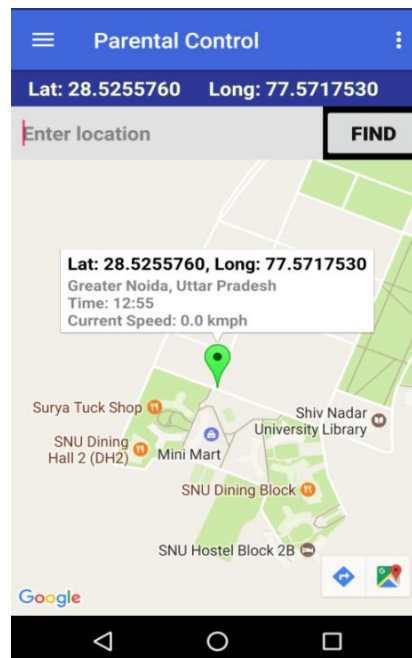


Image 8.13 Starting location depicted by Green marker

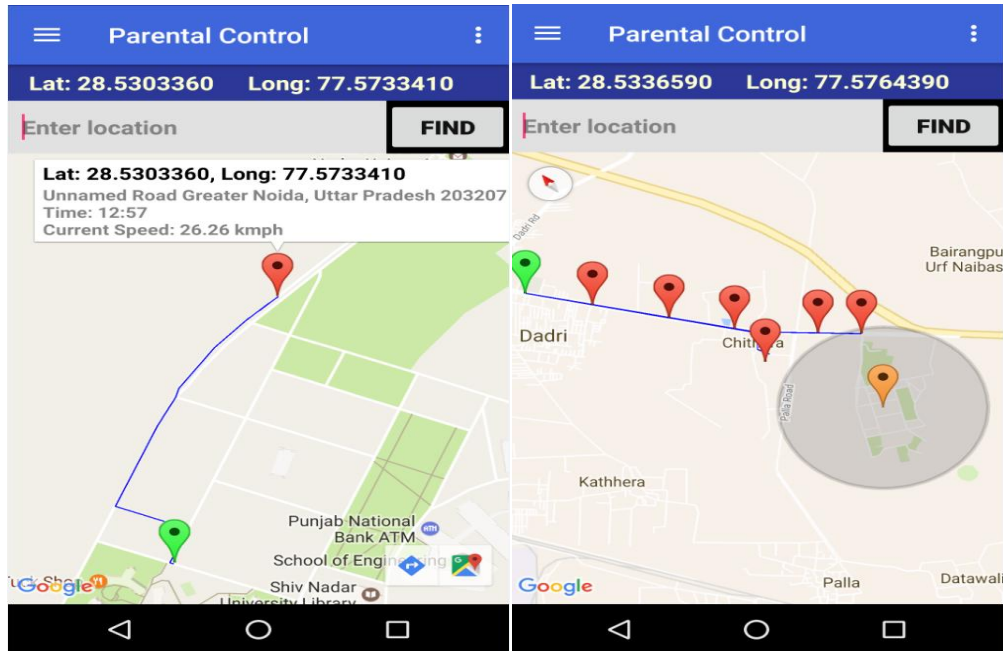


Image 8.14 a) Blue polyline trail, Image 8.14 b) Geo-fence created with trail.

When clicked on map, 'orange' marker is created which is the marker for creating geo-fence. On clicking any marker some details will be displayed in a rectangular box having Lat and Long, address of the marker, current time and current speed of the vehicle.

For creating a geo-fence some steps needs to be followed: click on the map (orange marker created) -> click on RHS menu option -> click on Create Geofence -> enter radius in meters -> click OK (geo-fence will be created).

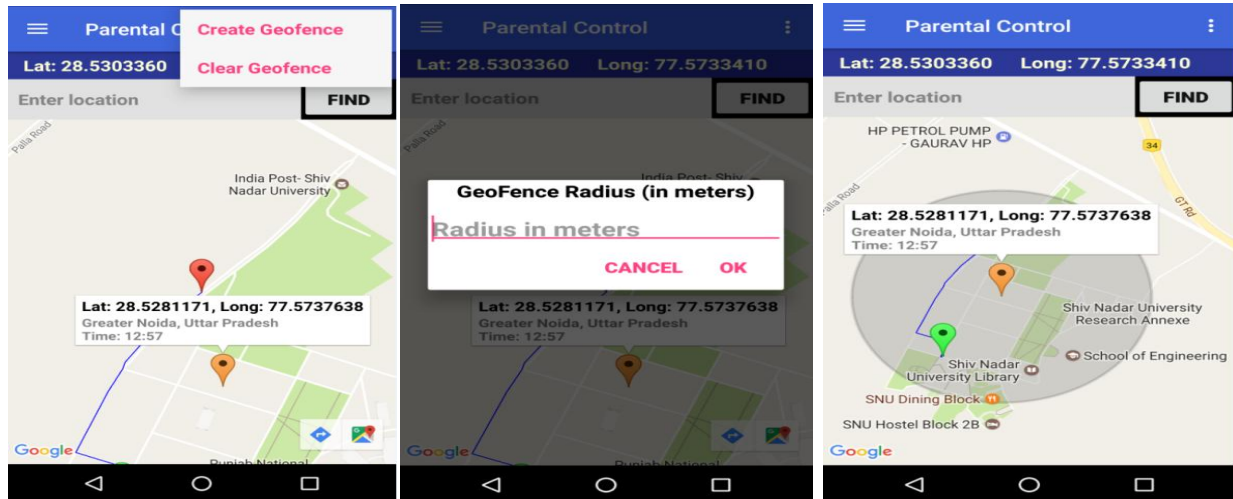


Image 8.15 a) Orange marker for creating geo-fence and menu option on RHS for creating or clearing geo-fence, Image 8.15 b) Setting geo-fence radius in meters, Image 8.15 c) Geo-fence created w.r.t Orange marker

For clearing already created geo-fence: click on RHS menu option -> click on Clear Geofence (geo-fence will be cleared).

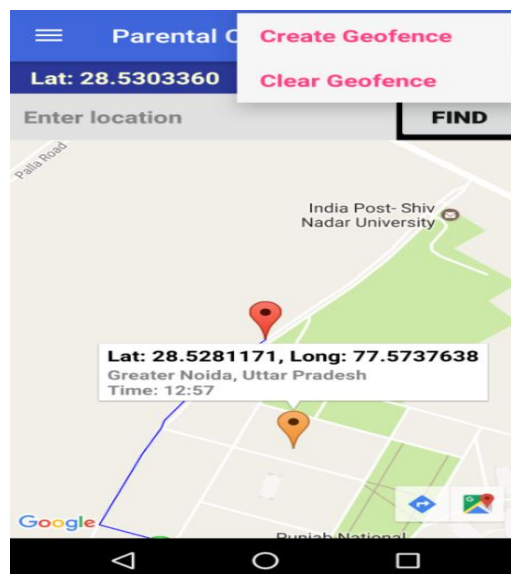


Image 8.16 Geo-fence cleared when Clear Geo-fence is clicked on the menu.

When a vehicle enters an already created geo-fence a notification would be triggered with an alert message on the notification bar of the parent`s mobile device.

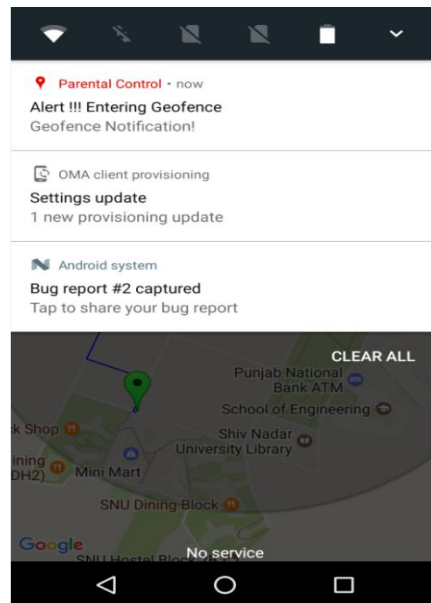


Image 8.17 Alert Notification when entering geo-fence.

When a vehicle exits an already created geo-fence a notification would be triggered with an alert message on the notification bar of the parent`s mobile device.

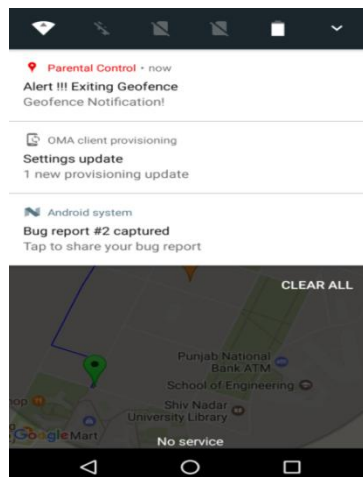


Image 8.18 Alert Notification when exiting geo-fence

8.3.1.2.2. Over-Speed Limit

The second view has an image of speed limit at the top. Below the image there is an input field and submit button, where user can enter the speed limit in kmph and that speed would be set as speed limit when submit button is pressed. At the bottom, current vehicle speed is shown in kmph rounded to 2 decimal places.

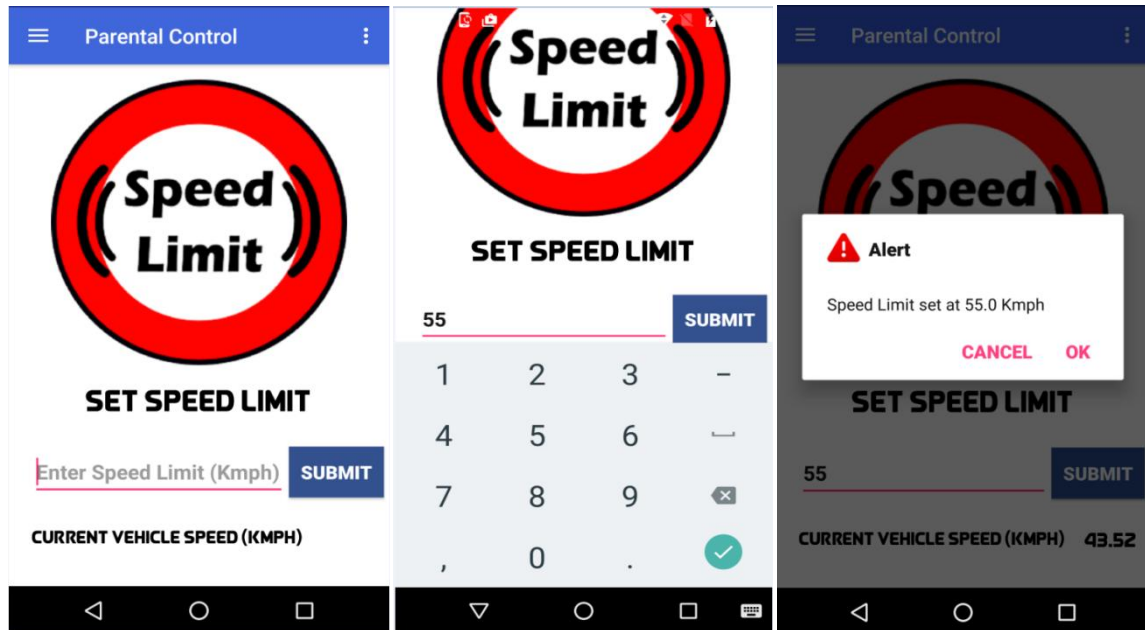


Image 8.19 a) Over-Speed limit view.

Image 8.19 b) Setting speed limit in kmph.

Image 8.19 c) Speed limit set with Alert message.

8.3.1.3. Back end

Our parental control application uses Google Firebase as its database technology. Google Firebase is an open source cross-platform API where the real time data updates are stored quickly on the cloud. One of the biggest advantages of Firebase is that it stores data in JSON format and automatically reports any crashes to the user. For this we just need to have an account on Firebase logged in from Gmail.

The real time data is pushed to Firebase from a python script running on the Raspberry pi placed in the vehicle. We are updating data on Firebase periodically every 30 seconds and whenever there is data change, android application automatically fetches from the corresponding value from Firebase. On Firebase database we have two branches: location and speed. Location has two children: Latitude and Longitude which is in the form of degree decimal (DD) and is in double precision value. Speed is taken in kmph and is up to 2 decimal precision.

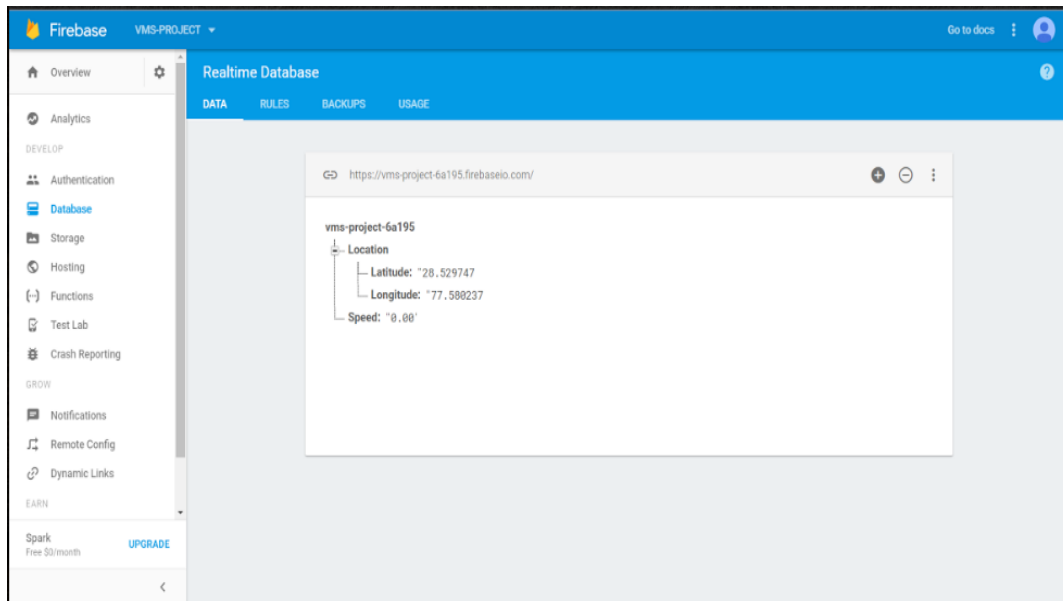


Image 8.20 Firebase database, <https://vms-project-6a195.firebaseio.com/>

Chapter 9

Risk Analysis

9. Risk Analysis

The team, upon designing the system, examined various risks that the team might face in the future. Hence team developed solutions to counter the same.

9.1 Fire Sensor

The project involves development of the system using fire sensor that would be reading data for prolonged period of time. Hence it was essential that the team tested the sensor for long duration. The sensor worked perfectly with no heating issues since we have installed cooling fans already, and the accuracy is also unaffected.

9.2 GPS Module

Continuous usage of GPS module leads to Raspberry Pi getting extremely hot. Hence we installed two mini-cooling fans near the Raspberry Pi to keep it cool. We have tested by continuously running the setup for 4 hours. It is completely safe to run the setup for long duration with same efficiency.

9.3 Wireless Communication

Since the system to be developed in the project would involve a raspberry send data a Google Firebase server, it was essential to test the wireless communication between them. The data was sent using Wi-Fi. We are able to send data without much delay. And from firebase to android app also there is a smooth communication.

Chapter 10

Conclusion

10. Conclusion

Upon extensive research and careful analysis, the project team has been able to design a fully functional IoT system. The team has been successful in implementing features like panic button and fire detection. The team is able to send message alerts with the current location of vehicle. The team is able to monitor the location of the vehicle.

The team has also been successful in making an android application which has features like Geo-fencing and over-speed limit constraint. User can create a virtual circular boundary, which triggers alert when vehicle enter/exit the created boundary. In addition to this, user can see the complete trail of path taken by the vehicle on Google Maps. User can also monitor the speed of the vehicle in real time which is updated every minute. Moreover, user can also set the speed limit for the vehicle that it doesn't want to cross which would trigger an alert message if the vehicle crosses that limit.

Chapter 11

References

11. References

1. Motivation - <https://www.edgarsnyder.com/car-accident/who-was-injured/teen/teen-driving-statistics.html>
2. Road Accident Statistics in India. (2016). Available at: <http://timesofindia.indiatimes.com/india/75000-youths-killed-in-road-crashes-last-year/articleshow/48780863.cms>
3. Mr. Basavaraju S R. (2015). Automatic Smart Parking System using Internet of Things (IOT). Available at: <http://www.ijsrp.org/research-paper-1215/ijsrp-p4898.pdf>
4. Scope template: <http://www.projectmanagementdocs.com/project-documents/project-scope-statement.html#axzz4YroGkxQd>
5. Vehicle tracking system: https://en.wikipedia.org/wiki/Vehicle_tracking_system
6. Related Works: IN-VEHICLE MONITORING SYSTEMS IMPROVE DRIVING SKILLS. Available at: <http://www.shell.com/business-customers/shell-fuel-cards/health-security-safety-and-the-environment/in-vehicle-monitoring-systems-can-help-everyone-to-improve-their-driving-skills.html>
7. Using UART instead of USB: <https://learn.adafruit.com/adafruit-ultimate-gps-on-the-raspberry-pi/using-uart-instead-of-usb>
8. Messaging software: Twilio. Available at: <https://www.twilio.com/>
9. Cloud Platform: Firebase. Available at: <https://firebase.google.com/>
10. Geofencing: <https://developer.android.com/training/location/geofencing.html>
11. How to work with Geofences on android. Available at: <https://code.tutsplus.com/tutorials/how-to-work-with-geofences-on-android--cms-26639>
12. Read and write data on Android: <https://firebase.google.com/docs/database/android/read-and-write>