R. V. COLLEGE OF ENGINEERING, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

An IoT Enabled Real-Time Communication and Location Tracking System for Vehicular Emergency

Minor Project Report

Submitted by,

Group Members	**USN**
Rahul R Jois	1RV15EC116
Saurav Bandlapalli	1RV15EC129
Srinivas H	1RV15EC152

Under the guidance of

Sri. S. Praveen
Assistant Professor
R V College of Engineering

In partial fulfilment for the award of degree
of
Bachelor of Engineering
in
Electronics and Communication Engineering
2018-19

R. V. COLLEGE OF ENGINEERING, BENGALURU-59

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the minor project work titled ***'An IoT Enabled Real-Time Communication and Location Tracking System for Vehicular Emergency'*** is carried out by **Rahul R Jois (1RV15EC116), Saurav Bandlapalli (1RV15EC129) and Srinivas H (1RV15EC152)** who are bonafide students of R.V College of Engineering, Bengaluru, in partial fulfilment for the award of degree of **Bachelor of Engineering in Electronics and Communication** of the Visvesvaraya Technological University, Belagavi during the year 2018-2019. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the minor project report deposited in the departmental library. The minor project report has been approved as it satisfies the academic requirements in respect of minor project work prescribed by the institution for the said degree.

Signature of Guide
(Sri. S. Praveen)

Signature of Head of the Department
(Dr. K S Geetha)

External Viva

Name of Examiners　　　　Signature with date

1
2

R.V. COLLEGE OF ENGINEERING, BENGALURU - 560059

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DECLARATION

We, Rahul R Jois, Saurav Bandlapalli, Srinivas H students of seventh semester B.E., Department of Electronics and Communication Engineering, R V College of Engineering, Bengaluru, hereby declare that the minor project titled **'An IoT Enabled Real-Time Communication and Location Tracking System for Vehicular Emergency'** has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering** in **Electronics and Communication Engineering** during the year 2018-19**.**

Further we declare that the content of the dissertation has not been submitted previously by anybody for the award of any degree or diploma to any other university.

Place: Bengaluru

Date:

Name Signature of the student

1. Rahul R Jois

2.Saurav Bandlapalli

3.Srinivas H

ACKNOWLEDGEMENT

We are indebted to our guide**, S Praveen,** Assistant Professor, **Dept of ECE, RVCE** for his wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis.

We also express our gratitude to our panel members **Dr. H. V. Ravish Aradhya,** Professor and **Dr. Abhay. A. Deshpande,** Associate Professor, Department of Electronics and Communication Engineering for their valuable comments and suggestions.

Our sincere thanks to **Dr. K S Geetha**, Professor and Head, Department of Electronics and Communication Engineering, RVCE for her support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for his appreciation towards this project work.

We thank all the **teaching staff and technical staff** of Electronics and Communication Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our **family** members who provided all the backup support throughout the project work.

ABSTRACT

Internet of Things (IoT) technology is helping mankind to achieve the goal of a smart city. Devices in smart city are connected to a single network 24x7 and constantly communicate with each other. Most of the deaths caused by accidents are due to delayed arrival of rescue teams to the accident location. So, if the accident information can be sent to the respective authorities immediately after it has occurred at least some of the lives could be saved. Present day technologies like the Ford Sync and Onstar by GM have come up solutions to deal with the issue. However, there is a serious limitation that exists in these systems. On the event of an emergency these systems contact their respective call centres which may or may not be close by. Instead it would be convenient to contact the nearest centre rather than contacting the respective call centres for help. So, the proposed system aims to overcome the drawbacks of all the existing systems.

The project is divided into three parts: Transmitter, Central hub and Receiver. The transmitter with its sensors detects the emergency situation and along with manual entry, the user sends the message to the central server. The server contacts the nearest rescue centre and the Rescue centre will assess the situation and send help accordingly. The module used for accident detection consists of accelerometer, gyroscope, GPS module and USB Webcam embedded to a microcontroller (Raspberry Pi). The emergency data is stored in Firebase and the Central Hub is hosted on AWS. The emergency history can be checked using the Android application.

Upon realizing the module and simulating an accident, along with manual triggers, the vehicular data was uploaded to the firebase. This data was then processed by the server and the SMS was received by the phone number registered to the nearest rescue centre. The emergency data was also displayed on the android application.

Emergency communication and location tracking system aimed at minimizing response time is the outcome of the project. The designed system will be made to manually or automatically trigger based on the emergency situation and the details will be sent to the nearest rescue centre. The project has been extended beyond emergencies and can also be used to alert the officials about the potholes on roads and can also be used to contact for help in case of mechanical failures. The future work of the project work includes improvement of app functionalities and integrating the entire module to a chip.

ACRONYMS

IoT - Internet of Things

RPi - Raspberry Pi

SoC - System on Chip

GPIO - General Purpose Input Output

SBC - Single Board Computer

NOOBS - New Out Of Box Software

CSI - Camera Serial Interface

DSI - Display Serial Interface

USB - Universal Serial Bus

GPS - Global Positioning System

DoF - Degree of Freedom

SMS - Short Messaging Service

API - Application Program Interface

SDK - Software Development Kit

JSON - Java Script Object Notation

AWS - Amazon Web Service

SSD - Solid State Device

GUI - Graphical User Interface

NoC - Network on Chip

NMEA - National Marine Electronics Association

WAAS - Wide Area Augmentation System

TABLE OF CONTENTS

Page No

Abstract v

List of Tables ix

List of Figures x

Chapter 1

1. Introduction 2
   1.1 Literature Survey 2
   1.2 Motivation 3
   1.3 Objectives of the project 3
   1.4 Methodology 4
   1.5 Block Diagram 5

Chapter 2

2. Hardware
   2.1 Hardware Components 8
   2.2 Hardware Specification 15
   2.3 Block Diagram 18

Chapter 3

3. Software
   3.1 Software Services 20
   3.2 Algorithm 22
   3.3 Design Flow 23
   3.4 Vehicle and Rescue Centre Database 25
   3.5 Code Snippets 27

Chapter 4

4. Results

4.1 Module Results 32

4.2 Server Results 35

4.3 Rescue Centre Results 36

Chapter 5

5. Conclusion and Future scope

5.1 Conclusion 38

5.2 Future Scope 39

References 41

List of Tables **Page No**

1. Emergency type and respective contact authorities 6

2. Budget Estimation 17

3. Comparison with other models 35

List of Figures **Page No**

1.1. Methodology 4

2.1. Raspberry Pi3 pin diagram 11

2.2. Interfacing Gyro with RPi 13

2.3. GPS message 14

2.4. Block diagram 18

3.1. Distance between two points on sphere 22

3.2. Control node 24

3.3. Situation node 25

3.4. Vehicle database 26

3.5. Rescue Centre database 26

3.6. Calculation of roll, pitch and yaw 27

3.7. Firebase authentication 27

3.8. Extraction of latitude and longitude 28

3.9. Haversine formula to find nearest centre 28

3.10. Twilio authentication 29

3.11. Sending SMS 29

3.12. Viewing image 30

4.1. Accident detection 32

4.2. Image capturing and location tracking 32

4.3. Uploading data 33

4.4. Acknowledgement Message 33

4.5. Nearest rescue centre 34

4.6. SMS received 34

4.7. Emergency message displayed on app 35

4.8. Final prototype 35

4.9. Prototype with webcam and external GPS antenna 36

4.10. Inside the prototype 36

CHAPTER-1
INTRODUCTION

Chapter 1
INTRODUCTION

Introduction

Internet of Things (IoT) technology is helping us to achieve the goal of a smart city. Devices in smart city are connected to a single network 24x7 and constantly communicate with each other. In this project, we use the concept of a smart city to provide a lifesaving system for a smart vehicle in any kind of emergency situation which normally occur. These days, most of the cars are equipped with sensors, embedded hardware, mechanical devices, software etc. to pre-detect a collision in order to avoid them and are very much reliable to the car drivers. However, there is a serious limitation that exists in these systems. As stated above, these systems can be used to avoid crashes. However, if a crash does occur, these systems have no provisions for further measures like calling for help, or provisions in case of other emergencies. If a mechanical failure occurs, or the driver falls seriously ill behind the wheels, these systems can't help us. A study says that in India 141,526 people were killed on road in 2014 by different types of road accidents [1]. Most of them were killed due to delayed arrival of rescue teams to the accident location. So, if the accident information can be sent to the respective authorities immediately after it has occurred at least some of the lives could be saved.

1.1 Literature Survey

Internet of Things and Smart City are emerging research topics these days in Internet oriented technologies and are grabbing the attention of researchers. The exponential growth of this field is taking us rapidly towards a smart planet, well-equipped with smart objects everywhere.

In [2] authors, J. Maleki, E. Foroutan, and M. Rajabi have proposed a GPS based location tracking system able to collect location information and send it through SMS. The main problem of this system is that it is not a fully automated system. The user has to start the system manually. So, this system isn't feasible for our requirements.

In [3] authors, C. Thompson, J. White, B. Dougherty, A. Albright, and D. C. Schmidt have introduced a system that can detect an accident by a smart phone's sensors, e.g. accelerometer sensor etc. and the phone uses its 3G connection to transmit accident information. But the system is not integrated into the vehicle and also not fully automated and sometime needs third party to send complete emergency information along with photos.

Ford also provides similar types of facility by their Ford Sync [4] app on their cars. When a user needs emergency assistance the app shows the emergency phone numbers from his smart phone on the screen so that the user can contact them immediately. The system can also contact the emergency 911 number for emergency situations. This app again is completely dependent on the user’s smartphone.

Authors of [5], Y.-K.Ki and D.-Y. Lee introduced ARRS that can automatically detect an accident and report it. They have used image processing approach to detect a vehicular crash from CC Camera videos. The main problem with this system is that the accidents can’t be detected in absence of a camera.

In [6] authors, SeokJu Lee, GirmaTewolde and Jaerock Kwon proposed a system which made good use of a popular technology that combines a Smartphone application with a microcontroller. The designed in-vehicle device worked using Global Positioning System (GPS) and Global system for mobile communication / General Packet Radio Service (GSM/GPRS) technology that is one of the most common ways for vehicle tracking. The device was embedded inside a vehicle whose position is to be determined and tracked in real-time. A microcontroller was used to control the GPS and GSM/GPRS modules.

In [7] the authors worked on a wireless black box using MEMS accelerometer and GPS tracking system is developed for accidental monitoring. The system consisted of cooperative components of an accelerometer, microcontroller unit, GPS device and GSM module. In the event of accident, the wireless device sent mobile phone short massage indicating the position of vehicle by GPS system to family member, emergency medical service (EMS) and nearest hospital. The threshold algorithm and speed of motorcycle were used to determine fall or accident in real-time.

In [10] the authors developed Automatic Smart Accident Detection (ASAD) an auto-detection unit system that immediately notifies an Emergency Contact through a text message when an instant change in acceleration, rotation and an impact force in an end of the vehicle is detected by the system, detailing the location and time of the accident. The system involved the use of fuzzy logic as a decision support built into the smartphone application that analyses the incoming data from the sensors and made decisions based on a set of rules.

In [13] the authors presented a novel method for automatic traffic accident detection, based on Smoothed Particles Hydrodynamics (SPH) wherein a motion flow field is obtained from the video through dense optical flow extraction. Then a thermal diffusion process (TDP) is exploited to turn the motion flow field into a coherent motion field. Approximating the moving particles to individuals, their interaction forces, represented as endothermic reactions, were computed using the enthalpy measure, thus obtaining the potential particles of interest.

In [16] the authors proposed an improved nonparametric regression (INPR) algorithm for forecasting traffic flows and its application in automatic detection of traffic incidents. The INPRA was constructed based on the searching method of nearest neighbours for a traffic-state vector and its main advantage lied in forecasting through possible trends of traffic flows, instead of just current traffic states, as commonly used in previous forecasting algorithms.

1.2 Motivation

Most of the accident detection/prevention systems are dependent on the user’s smartphone and aren’t automated. Many of these systems are specific to their vehicle’s company and will contact only the company’s call centre and not the nearest helpline. Also, there is no provision to contact the nearest police station or an hospital directly which unnecessarily causes delay in rescue mission. Also, there is no existing system which allows the user to contact mechanic centres in case the vehicle breaks down in remote areas. So that’s leaves us without a universal system which detects an accident or an emergency and helps us in contacting for help.

1.3 Objectives

- The project focuses on designing lifesaving system for a vehicle in any kind of emergency situation occurring on road.
- Project presents an IoT enabled approach that can provide emergency communication and location tracking services in a remote car that meets an unfortunate accident or any other emergency.

- System is designed for usage of both automatic and manual configuration.
- The proposed system will be able to send emergency information which includes emergency type, location, initial photo, and car's information, including emergency contact details to nearby rescue centre.

1.4 Methodology

- The system is divided in three major parts, an on-board embedded device (situation node), emergency control terminal room and rescue centre terminal.
- To store the necessary information of car and rescue centre a well-structured database is designed in firebase.
- The system can be triggered automatically and manually. To detect an accident, sensors such as gyroscope and accelerometer are being used.
- Additional information such as image and location are detected using camera and GPS Module.
- Fig 1 gives a brief idea about the methodology.

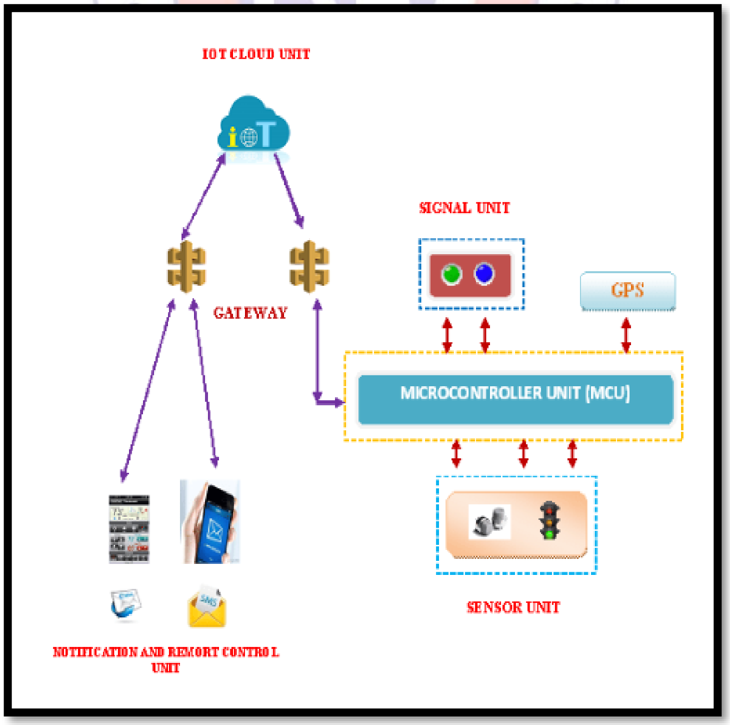
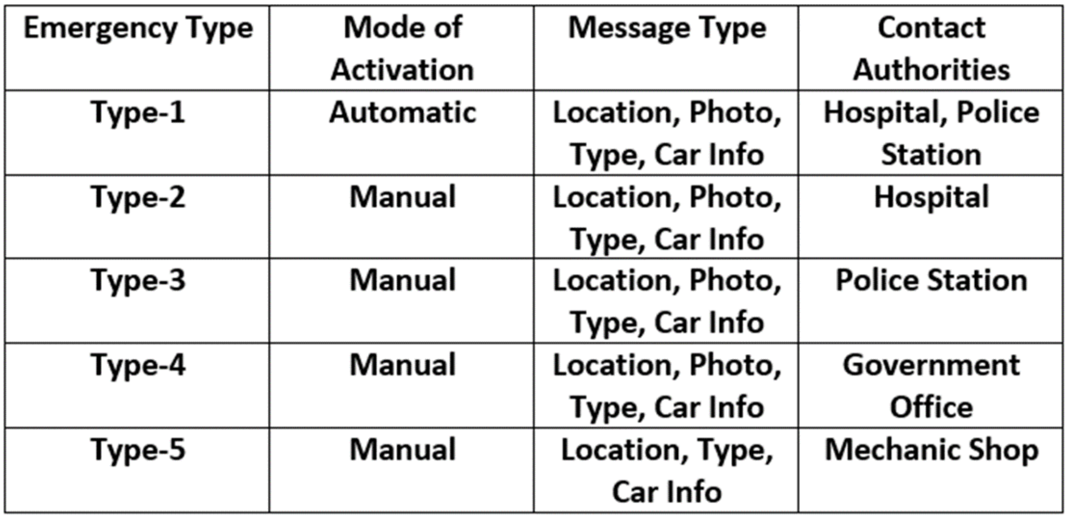

IOT CLOUD UNIT
iOT
SIGNAL UNIT
GPS
GATEWAY
MICROCONTROLLER UNIT (MCU)
SENSOR UNIT
NOTIFICATION AND REMORT CONTROL UNIT

Fig1.1: Methodology

Table1: Emergency types and respective contact authorities



Emergency Type	Mode of Activation	Message Type	Contact Authorities
Type-1	Automatic	Location, Photo, Type, Car Info	Hospital, Police Station
Type-2	Manual	Location, Photo, Type, Car Info	Hospital
Type-3	Manual	Location, Photo, Type, Car Info	Police Station
Type-4	Manual	Location, Photo, Type, Car Info	Government Office
Type-5	Manual	Location, Type, Car Info	Mechanic Shop

Table1 shows various types of emergencies included in the proposed system. The table shows different modes to activate and data considered for each mode. Each type is provided with their respective rescue centre and the message type along with the authorities who will be contacted for each type of emergency.

Brief explanation of each type of emergency is as follows:

- **Type-1(Accident):** This is the most important emergency type for a vehicle on road. When a vehicle crashes or meets any accident, the system sends the emergency message to the nearest hospital and police station. Also, the preinstalled cameras activate to help the rescue teams to understand the real scenario of the situation from the base station and act accordingly.
- **Type-2 (Medical):** Sometime it happens that a passenger or the driver of a car suddenly becomes sick and is unable to go to the hospital or find any hospital nearby. In that case, they can start the system manually and define the emergency type to medical issue. For this type of emergency, the control room sends the message to the nearest hospital as emergency medical situation and the hospital acts accordingly.
- **Type-3 (Criminal):** If a car meets some criminal issue, they can also contact the control room for help. For this case the nearest police station is informed.
- **Type-4 (Civil):** If there is any natural calamity, and the road is blocked by some barrier, the nearest government civil service office and police station are informed.

- **Type-5 (Mechanical):** If a vehicle meets some mechanical problems, nearest car workshop is informed.

1.5 Budget Estimate

Table 2: Budget estimation

Component	**Estimated Cost (₹)**
Raspberry Pi 3 Model B	3,066.5
9DoF AltIMU-10 Gyro/Accelerometer	2656.32
Adafruit Ultimate GPS Module	3689
SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable	376.42
External GPS Antenna	1273.22
5V Power Adapter 3A with Micro USB Cable	258.42
16GB Class 10 MicroSD Memory Card	470.82
CR1220 3V Lithium Battery	175
Raspberry Pi Case	334
Total Cost	12297.52

CHAPTER-2
HARDWARE

Chapter 2
HARDWARE

In this chapter, the different hardware components which are required in order to establish the methodology will be briefly explained. The specifications of all components used will also be discussed along with the block diagram of the module. Also, the interfacing of all these components with the microcontroller will also be explained.

2.1 Hardware Components

The hardware components for the module are chosen based on their specifications. The components used to build the module are as follows:

2.1.1 Raspberry Pi 3 Model B

Raspberry Pi is an ARM based credit card sized SBC created by Raspberry Pi Foundation. Raspberry Pi runs Debian based GNU/Linux operating system Raspbian and ports of many other OSes exist for this SBC.

The Raspberry Pi 3 has similar size as Raspberry Pi 2 with slight change in component placement to allow addition of Wi-Fi / Bluetooth SoC & Chip antenna in Pi 3. A ceramic chip antenna is used by Wi-Fi and Bluetooth 4.1 SoC BCM43438. The chip antenna moves the indicator LEDs that were present in Pi 2 to the lower side of PCB.

As the Raspberry Pi 3 supports HD video, it can even create a media centre for user. The Raspberry Pi 3 Model B is the first Raspberry Pi to be open-source from the get-go, expect it to be the de facto embedded Linux board in all the forums.

The quad-core Raspberry Pi 3 is both faster and more capable than its predecessor, the Raspberry Pi 2. The Pi 3's CPU-the board's main processor-has roughly 50-60 percent better performance in 32-bit mode than that of the Pi 2, and is 10x faster than the original single-core Raspberry Pi (based on a multi-threaded CPU benchmark in SysBench). Compared to the original Pi, real-world applications will see a performance increase of between 2.5x--for single-threaded applications--and more than 20x--when video playback is accelerated by the chip's NEON engine.

Unlike its predecessor, the new board is capable of playing 1080p MP4 video at 60 frames per second (with a bit rate of about 5400Kbps), boosting the Pi's media centre credentials. That's not to say, however, that all video will playback this smoothly, with performance dependent on the source video, the player used and bitrate.

The Pi 3 also supports wireless internet out of the box, with built-in Wi-Fi and Bluetooth. The latest board can also boot directly from a USB-attached hard drive or pen drive, as well as supporting booting from a network-attached file system, which is useful for remotely updating a Pi and for sharing an operating system image between multiple machines.

Setting up Raspberry Pi:

To setup Raspberry Pi, the following peripherals are required:

1. Power supply — A 5V micro-USB power supply.

2. USB keyboard and USB mouse or a Bluetooth keyboard and mouse.

3. MicroSD card — the microSD card must have at least 8 GB of storage and should preferably be a class 10 SD card.

4. MicroSD USB card reader — to connect the microSD card to your PC or Mac in order to download software onto it. Adafruit carries one that is perfect for Raspberry Pi.

5. A monitor or TV that supports HDMI or composite video.

6. An HDMI cable or composite video cable, depending on the requirements.

Once the above-mentioned components are ready, the following steps have to be followed to setup the RPi.

Step 1: Reformat your microSD card: The first step to getting started with Raspberry Pi is to reformat the microSD card that will be use to download the operating system. Even brand-new SD cards will have some extraneous files on them. Reformatting it will remove all files and completely clear the card.

Step 2: Download the RPi OS onto the microSD card. Once it is downloaded, the file can be extracted using any image writing software.

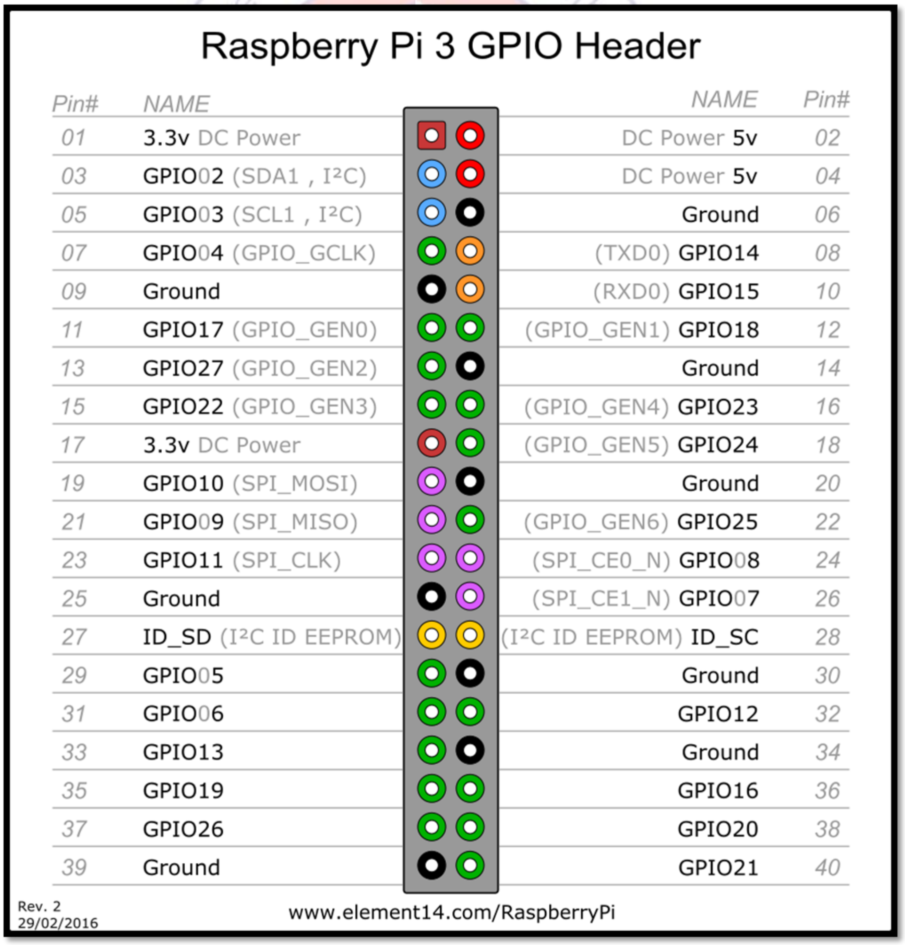
Step 3: Insert the microSD card into the card slot on the underside of the Raspberry Pi.

Step 4: Connect the RPi to the TV monitor using HDMI cable.

Step 5: Configure Raspberry Pi

When Raspbian begins to load a bunch of lines of code will appear. This will continue until the boot process has completed. Then, the Raspbian Home screen will appear. You will need to configure your Raspberry Pi system in order to add your location, date, and time.

Figure 2 shows the pin configuration of the RPi. There are 40 GPIO's available which includes 8 grounds, 3 DC power pins and 9 free GPIO pins.



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

Fig2.1: Raspberry Pi3 pin diagram

2.1.2 AltIMU-10 v4 Gyro, Accelerometer, Compass, and Altimeter

An accelerometer is a compact device designed to measure non-gravitational acceleration. When the object it's integrated into goes from a standstill to any velocity, the accelerometer is designed to respond to the vibrations associated with such movement. It uses microscopic crystals that go under stress when vibrations occur, and from that stress a voltage is generated to create a reading on any acceleration. Accelerometers are important components to devices that track fitness and other measurements in the quantified self-movement.

A gyroscope is a device that uses Earth's gravity to help determine orientation. Its design consists of a freely-rotating disk called a rotor, mounted onto a spinning axis in the centre of a larger and more stable wheel. As the axis turns, the rotor remains stationary to indicate the central gravitational pull, and thus which way is "down."

The main difference between the two sensors is simple: one can sense rotation, whereas the other cannot. In a way, the accelerometer can gauge the orientation of a stationary item with relation to Earth's surface. When accelerating in a particular direction, the accelerometer is unable to distinguish between that and the acceleration provided through Earth's gravitational pull. If you were to consider this handicap when used in an aircraft, the accelerometer quickly loses much of its appeal.

The gyroscope maintains its level of effectiveness by being able to measure the rate of rotation around a particular axis. When gauging the rate of rotation around the roll axis of an aircraft, it identifies an actual value until the object stabilizes out. Using the key principles of angular momentum, the gyroscope helps indicate orientation. In comparison, the accelerometer measures linear acceleration based on vibration.

The typical two-axis accelerometer gives users a direction of gravity in an aircraft, smartphone, car or other device. In comparison, a gyroscope is intended to determine an angular position based on the principle of rigidity of space. The applications of each device vary quite drastically despite their similar purpose. A gyroscope, for example, is used in navigation on unmanned aerial vehicles, compasses and large boats, ultimately assisting with stability in navigation. Accelerometers are equally widespread in use and can be found in engineering, machinery, hardware monitoring, building and structural monitoring, navigation, transport and even consumer electronics.

Interfacing the gyro (L3GD2) to Raspberry Pi is done by installing the I2C-tools. The config files in RPi should be edited by adding two lines – *i2c-dev* and *i2c-bcm2708*.

The default slave address of the gyroscope is 1101011b and that of the accelerometer is 0011101b. The readings can be obtained by fetching data from the above-mentioned slave addresses. Fig 2.2 shows how to interface the accelerometer and gyroscope with RPi. The VDD of the IC is connected to 3.3V pin of Raspberry Pi and SDA, SCL are both connected to I^{2}C SDA and SCL pins of RPi (Pins 3 and 5 respectively).

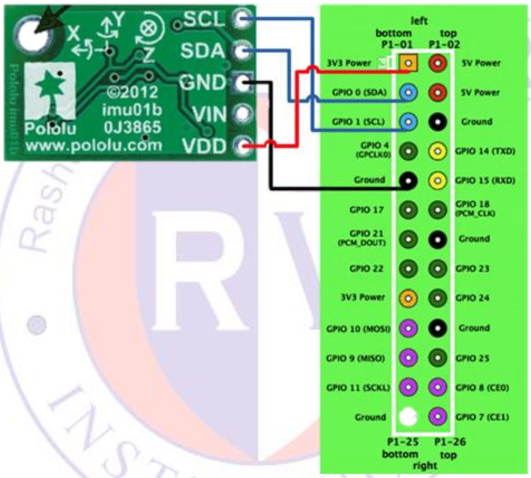


Fig2.2: Interfacing gyro and accelerometer with RPi

2.1.3 Adafruit Ultimate GPS Module:

GPS receivers use a constellation of satellites and ground stations to compute position and time almost anywhere on earth. At any given time, there are at least 24 active satellites orbiting over 12,000 miles above earth. The positions of the satellites are constructed in a way that the sky above your location will always contain at most 12 satellites. The primary purpose of the 12 visible satellites is to transmit information back to earth over radio frequency (ranging from 1.1 to 1.5 GHz). With this information and some math, a ground-based receiver or GPS module can calculate its position and time.

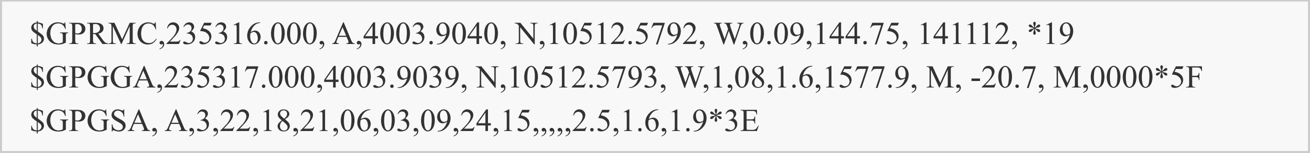
The data sent down to earth from each satellite contains a few different pieces of information that allows your GPS receiver to accurately calculate its position and time. An important piece of equipment on each GPS satellite is an extremely accurate atomic clock. The time on the atomic clock is sent down to earth along with the satellite's orbital position and arrival times at different points in the sky. In other words, the GPS module receives a timestamp from each of the visible satellites, along with data on where in the sky each one is located (among other pieces of data). From this information, the GPS receiver now knows the distance to each satellite in view. If the GPS receiver's antenna can see at least 4 satellites, it can accurately calculate its position and time**.** This is also called a lock or a fix.

The Adafruit Ultimate breakout is built around the MTK3339 chipset, a high-quality GPS module that can track up to 22 satellites on 66 channels, has an excellent high-sensitivity receiver (-165 dBm tracking), and a built-in antenna. It can do up to 10 location updates a second for high speed, high sensitivity logging or tracking. Power usage is very low, only 20 mA during navigation. The LED blinks at about 1Hz while it's searching for satellites and blinks once every 15 seconds when a fix is found to conserve power.

Two features that really stand out about version 3 MTK3339-based module is the external antenna functionality and the built-in data-logging capability. The module has a standard ceramic patch antenna that gives it -165 dBm sensitivity, but when a bigger antenna is required, any 3V active GPS antenna via the uFL connector can be snapped. The module will automatically detect the active antenna and switch over.

GPS data is displayed in different message formats over a serial interface. There are standard and non-standard (proprietary) message formats. Nearly all GPS receivers' output NMEA data. The NMEA standard is formatted in lines of data called sentences. Each sentence contains various bits of data organized in comma delimited format (i.e. data separated by commas).

The NMEA data will be obtained as shown in Fig 2.3. From this GPGGA data has to be extracted to obtain the co-ordinates.



```
$GPRMC,235316.000, A,4003.9040, N,10512.5792, W,0.09,144.75, 141112, *19
$GPGGA,235317.000,4003.9039, N,10512.5793, W,1,08,1.6,1577.9, M, -20.7, M,0000*5F
$GPGSA, A,3,22,18,21,06,03,09,24,15,,,,,2.5,1.6,1.9*3E
```

Fig2.3: GPS Message

2.1.4 Webcam

A webcam is a video camera that feeds or streams its image in real time to or through a computer to a computer network. When captured by the computer, the video stream may be saved, viewed or sent on to other networks travelling through systems such as the internet, and e-mailed as an attachment.

The webcam should be connected to the USB port of the Raspberry Pi. *fswebcam* should be installed to use the camera. This can be done using *pip install fswebcam* command on RPi. The resolution required can be set through software.

2.2 Hardware Specification:

1. Raspberry Pi 3 Model B:

   SoC: Broadcom BCM2837

   CPU: 4× ARM Cortex-A53, 1.2GHz

   GPU: Broadcom Video Core IV

   RAM: 1GB LPDDR2 (900 MHz)

   Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless

   Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy

   Storage: microSD

   GPIO: 40-pin header, populated

   Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

2. 9DoF AltIMU-10 Gyro/Accelerometer:

   Interface: I²C

   Minimum operating voltage: 2.5 V

   Maximum operating voltage: 5.5 V

   Axes: pitch (x), roll (y), and yaw (z)

   Measurement range: ±245, ±500, or ±2000°/s (gyro)
   ±2, ±4, ±6, ±8, or ±16 g (accelerometer)
   ±2, ±4, ±8, or ±12 gauss (magnetometer)

   Supply Current: 6 mA

3. Adafruit Ultimate GPS Module:

   Satellites: 22 tracking, 66 searching

   Patch Antenna Size:15mm x 15mm x 4mm

   Update rate: 1 to 10 Hz

   Position Accuracy:< 3 meters (all GPS technology has about 3m accuracy)

   Velocity Accuracy: 0.1 meters/s

   Warm/cold start: 34 seconds

   Acquisition sensitivity: -145 dBm

   Tracking sensitivity: -165 dBm

   Maximum Velocity: 515m/s

   Vin range: 3.0-5.5VDC

   MTK3339 Operating current: 25mA tracking, 20 mA current draw during navigation

   Output: NMEA 0183, 9600 baud default, 3V logic level out, 5V-safe input

   DGPS/WAAS/EGNOS supported

   FCC E911 compliance and AGPS support (Offline mode : EPO valid up to 14 days)

   Up to 210 PRN channels

4. External GPS Antenna:

   Supply current: 10 mA

   Additional gain provided: 28 dB

   Length: 5 m

5. MicroSD Card:

   Class 10

   Size: 16 GB

6. Web Camera:

   Still Image Resolution: 12.0 MP

   Video Resolution: 2.0 MP

7. 5V Power Adapter 3A with Micro USB Cable:

Input Voltage	: 110V to 220V
Output Voltage	: 5V DC
Max Output Current:	3A

2.3 Block Diagram:

- The main aim of hardware is to detect emergency situation this is done using sensors such as gyroscope and accelerometer.
- On detection of emergency situation, the system obtains the victim location using GPS breakout board and antenna connected to it.
- The system also obtains images in case needed and are used at rescue centre for drawing conclusion of condition. This is done using webcam.
- Finally, the system creates a message and sends the required data to database centre for further process.
- Fig 2.4 shows the block diagram of the hardware used in the module.

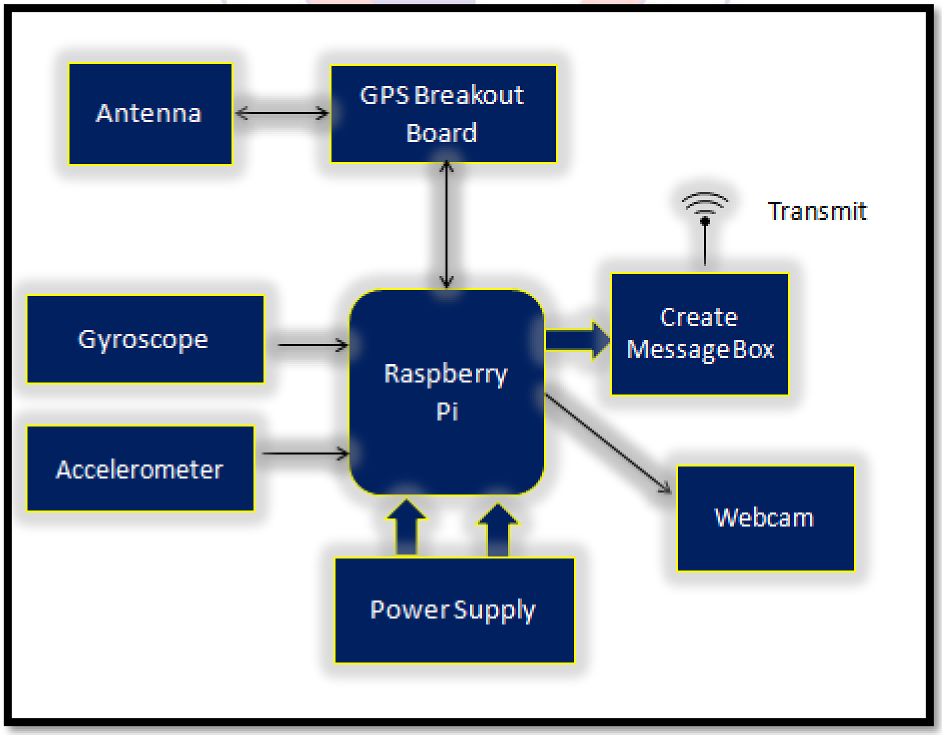


Fig2.4: Block diagram

CHAPTER-3
SOFTWARE

Chapter 3
SOFTWARE

This chapter introduces to different software services, flowcharts and algorithms which are required in order to establish the methodology. The software services are required to store the emergency data sent from RPi, a server to process the emergency data and also an android application to store the emergency history.

3.1 Software Services

3.1.1 Firebase

Firebase provides a real-time 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. Data is stored in firebase as a large JSON document. It is the same case as it is done in most of the NoSQL database systems like MongoDB, Cassandra, Couch DB etc. The data is stored as a large object which can hold key value pairs where value can be a string, number or another object. With just a single API, the Firebase database provides your app with both the current value of the data and any updates to that data. Real time syncing makes it easy for your users to access their data from any device, be it web or mobile. Real-time Database also helps your users collaborate with one another. Another amazing benefit of Real time Database is that it ships with mobile and web SDKs, allowing you to build your apps without the need for servers. When your users go offline, the Real time Database SDKs use local cache on the device to serve and store changes. When the device comes online, the local data is automatically synchronized. The Real time Database can also integrate with Firebase Authentication to provide a simple and intuitive authentication process.

3.1.2 Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability and a syntax that allows programmers to express concepts in fewer lines of code, notably using whitespace. It is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. Python combines remarkable power with very clear syntax. It has interfaces to many systems calls and libraries, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Also, Python is portable. It runs on many Unix variants, on the Mac OS, and on Windows 2000 OS and later. There are many versions of this python software available for different platforms. In this project Python Version 2.7.15 on Windows 10 OS is being used.

3.1.3 Amazon Web Services (AWS)

Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a virtual cluster of computers, available all the time, through the Internet. AWS's version of virtual computers emulate most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser. The browser acts as a window into the virtual computer, letting subscribers log-in, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for themselves and their customers.

3.1.4 MIT App Inventor

App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It allows newcomers to computer programming to create software applications for the Android operating system (OS). It uses a graphical interface, very similar to Scratch and the StarLogo TNG user interface, which allows users to drag-and-drop visual objects to create an application that can run on Android devices. In creating App Inventor, Google drew upon significant prior research in educational computing, as well as work done within Google on online development environments. App

Inventor and the projects on which it is based are informed by constructionist learning theories, which emphasizes that programming can be a vehicle for engaging powerful ideas through active learning. As such, it is part of an ongoing movement in computers and education that began with the work of Seymour Papert and the MIT Logo Group in the 1960s and has also manifested itself with Mitchel Resnick's work on Lego Mindstorms and StarLogo. MIT App Inventor is also supported with the Firebase Database extension. This allows people to store data on Google's firebase.

3.2 Algorithms

This section discusses about the different algorithms that are used to achieve the methodology.

3.2.1 Haversine Formula

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, which relates the sides and angles of spherical triangles.

The first table of haversines in English was published by James Andrew in 1805, but Florian Cajori credits an earlier use by José de Mendoza y Ríos in 1801. The term haversine was coined in 1835 by James Inman. Fig 3.1 shows the graphical representation of the implementation of haversine formula to calculate the distance between any two places on earth.

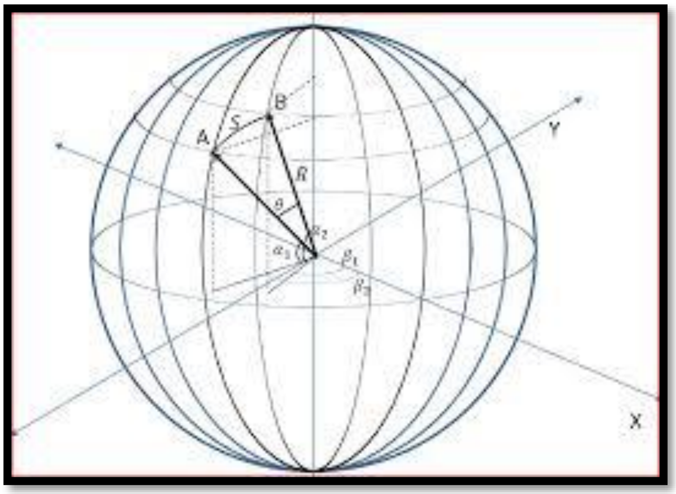


Fig3.1: Distance between two points on sphere

$$dlong = long1 - long \tag{3.1}$$

$$dlat = lat1 - lat \tag{3.2}$$

$$a = ((\sin\left(\frac{dlat}{2}\right))^2 + \cos lat \times coslat1 \times \left(\sin\left(\frac{dlong}{2}\right)\right)^2 \tag{3.3}$$

$$c = 2 \times arctan2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{3.4}$$

$$d = R \times c \tag{3.5}$$

Where,

lat = latitude of emergency location

lat1 = latitude of rescue centre 1

long = longitude of emergency location

long1 = longitude of rescue centre 1

R = Radius of earth (6371 kms)

d = Distance between two locations

3.3 Design Flow

The design flow deals with the way in which the software services are implemented. The system is classified into control node and situation node. Situation node involves the activities which take place after the occurrence of emergency till uploading the message to database, whereas control node involves the activities which take place right from when the emergency data arrives in the database, till notifying the nearest centre.

3.1.2. Situation node

- The sensors will be continuously be checking for the occurrence of the accident and will trigger automatically upon detecting the accident.
- The user will also be provided with manual triggering options for different emergencies like mechanical failure, theft, natural calamities.
- Upon occurrence of an accident, the microcontroller will capture the image, obtain the co-ordinates and will upload the data to Firebase. The data will further be handled by Control Node

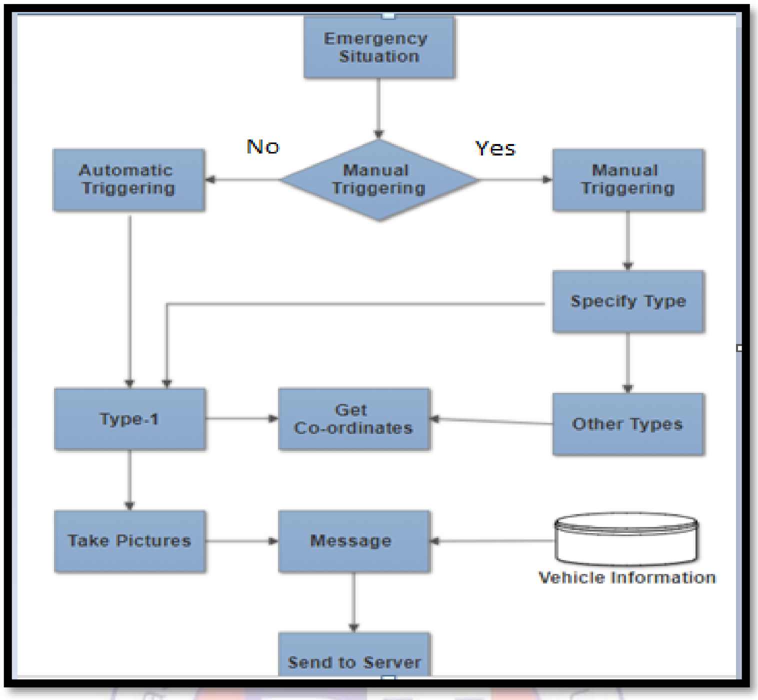


Fig3.2: Situation Node

3.1.2. Control Node:

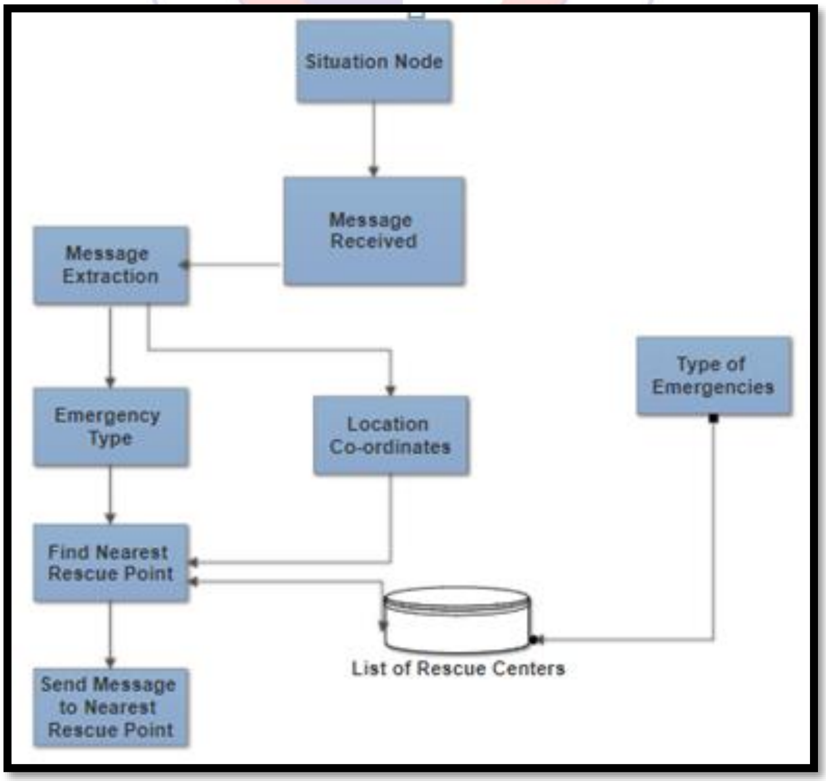
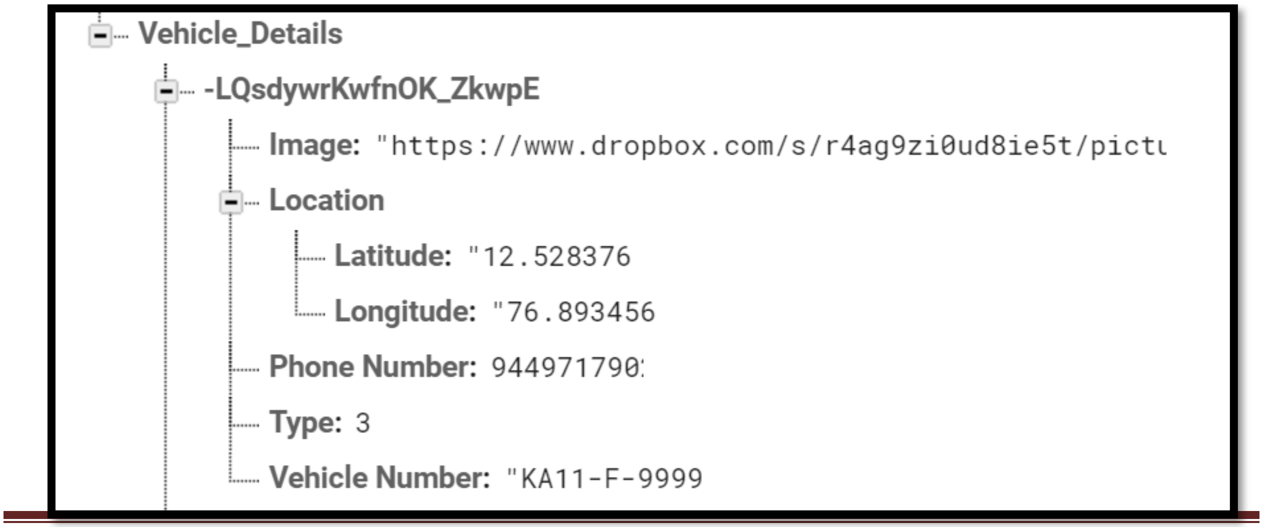


Fig3.3: Control Node

- The server extracts the location and emergency type once the message is received.
- From the list of rescue centre stored in firebase along with the help of location details of user and emergency type the code finds the nearest centre and sends the message to the respective centre.
- The system notifies the rescue centre about the situation and the rescue centre takes the respective actions that are required.

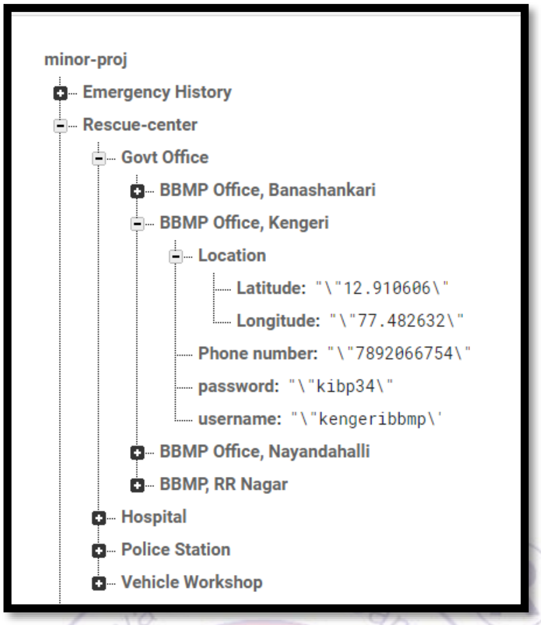
3.4 Vehicle and Rescue Centre Database

- Firebase helps in keeping real-time data, thus the databases are created and set up to keep track of emergency data sent from the device.
- The first database is the Rescue Centre database used to store all rescue centres and with other details such as location and phone number. Fig 3.5 shows the rescue centre database.
- The second database is the vehicle database used for storing the message in case of emergency situations. It contains location, phone number, type and the vehicle number. Fig 3.4 shows the Vehicle database which will contain the emergency message sent by the user. The emergency information includes the location of the emergency, the type, vehicle number, phone number and finally the image of the emergency situation. The server will be continuously checking for data to get added into vehicle database and will immediately find the nearest rescue centre using set of equations from (3.1) to (3.5).



```
Vehicle_Details
    -LQsdywrKwfnOK_ZkwpE
        Image: "https://www.dropbox.com/s/r4ag9zi0ud8ie5t/pictu
        Location
            Latitude: "12.528376
            Longitude: "76.893456
        Phone Number: 944971790:
        Type: 3
        Vehicle Number: "KA11-F-9999
```

Fig3.4: Vehicle database



```
minor-proj
  Emergency History
  Rescue-center
    Govt Office
      BBMP Office, Banashankari
      BBMP Office, Kengeri
        Location
          Latitude: "\"12.910606\"
          Longitude: "\"77.482632\"
        Phone number: "\"7892066754\"
        password: "\"kibp34\"
        username: "\"kengeribbmp\'
      BBMP Office, Nayandahalli
      BBMP, RR Nagar
    Hospital
    Police Station
    Vehicle Workshop
```

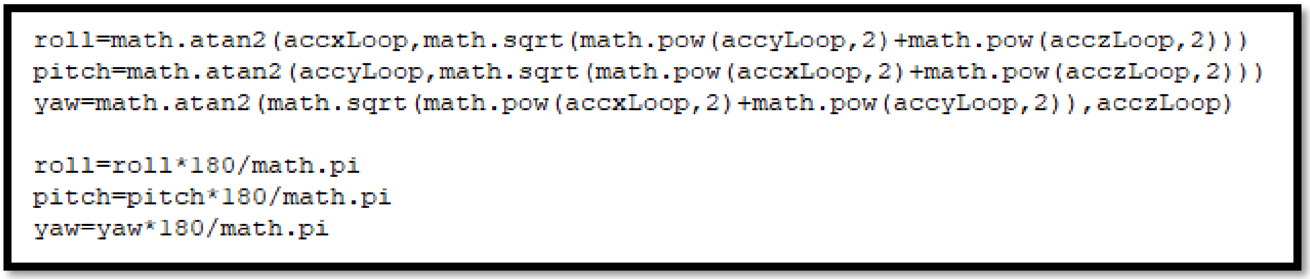
Fig3.5: Rescue Centre Database

3.5 Code Snippets

In this section, the significant sections of the code are discussed. This includes the code used in RPi, server as well as in the mobile application. The RPi code and the server code is written on Python and the application is developed using MIT App Inventor.

3.5.1 RPi Code

The RPi should continuously monitor the sensors used to detect the emergency. In this module, the accelerometer and gyroscope are used to detect an accident. Fig 3.6 shows the calculation of roll, pitch and yaw using the accelerometer readings. Similarly, the gyroscope readings are monitored at all time along with the manual triggering buttons for other types of emergencies. Therefore, at the same time RPi has to listen to multiple sensors at the same time.

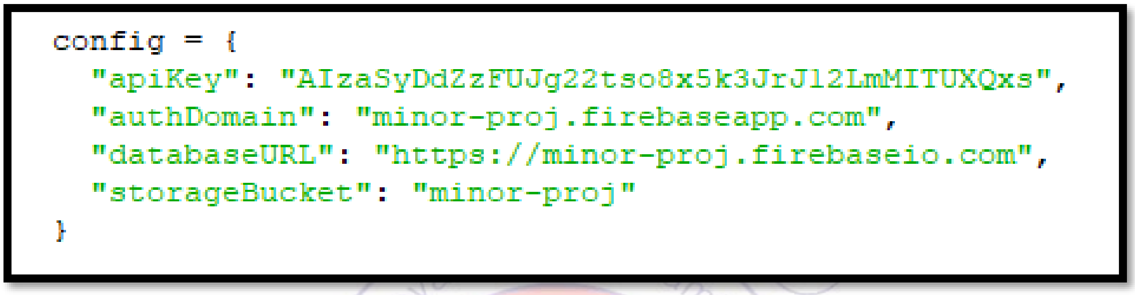


```
roll=math.atan2(accxLoop,math.sqrt(math.pow(accyLoop,2)+math.pow(acczLoop,2)))
pitch=math.atan2(accyLoop,math.sqrt(math.pow(accxLoop,2)+math.pow(acczLoop,2)))
yaw=math.atan2(math.sqrt(math.pow(accxLoop,2)+math.pow(accyLoop,2)),acczLoop)

roll=roll*180/math.pi
pitch=pitch*180/math.pi
yaw=yaw*180/math.pi
```

Fig3.6: Calculation of roll, pitch and yaw

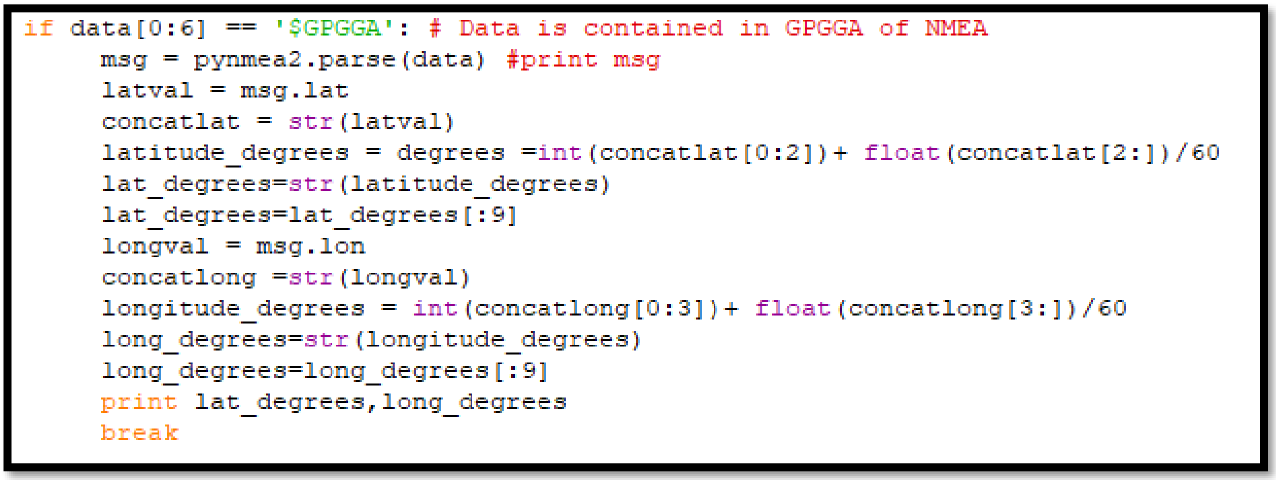
Once an emergency is detected, the RPi triggers the webcam to take the image and gets the co-ordinates of the location from the GPS module. Once the co-ordinates are obtained, the image is uploaded to DropBox and a sharable link is generated. Then RPi uploads the data to the firebase. To connect RPi to firebase, configuration should be done. Fig 3.7 shows the authorization code which includes the *apiKey* and *databseURL*. The configuration is essential in order to get connected to the firebase.



```
config = {
  "apiKey": "AIzaSyDdZzFUJg22tso8x5k3JrJl2LmMITUXQxs",
  "authDomain": "minor-proj.firebaseapp.com",
  "databaseURL": "https://minor-proj.firebaseio.com",
  "storageBucket": "minor-proj"
}
```

Fig3.7: Firebase Authentication

The module is connected to the GPS module via UART. The *serial0* of RPi is used and is initiated using the command "*/dev/serial0*" at a baud rate of 9600. Once the fix is obtained, GPS will get the co-ordinates of the location. The output obtained will be contained in the GPGGA string of the NMEA data. From this string, the latitude and longitude should be parsed and then converted to useful format. Fig 3.8 shows the extraction of latitude and longitude using some arithmetic operations.

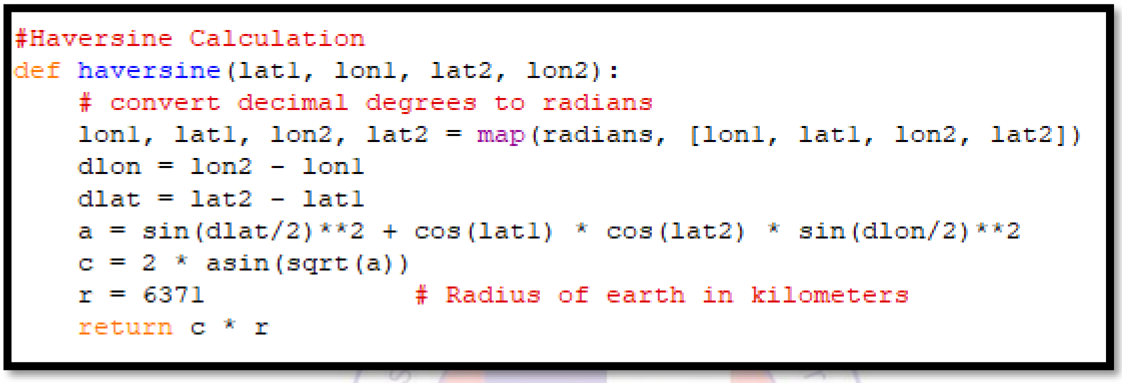


```
if data[0:6] == '$GPGGA': # Data is contained in GPGGA of NMEA
     msg = pynmea2.parse(data) #print msg
     latval = msg.lat
     concatlat = str(latval)
     latitude_degrees = degrees =int(concatlat[0:2])+ float(concatlat[2:])/60
     lat_degrees=str(latitude_degrees)
     lat_degrees=lat_degrees[:9]
     longval = msg.lon
     concatlong =str(longval)
     longitude_degrees = int(concatlong[0:3])+ float(concatlong[3:])/60
     long_degrees=str(longitude_degrees)
     long_degrees=long_degrees[:9]
     print lat_degrees,long_degrees
     break
```

Fig3.8: Extraction of latitude and longitude

3.5.2 Server Code

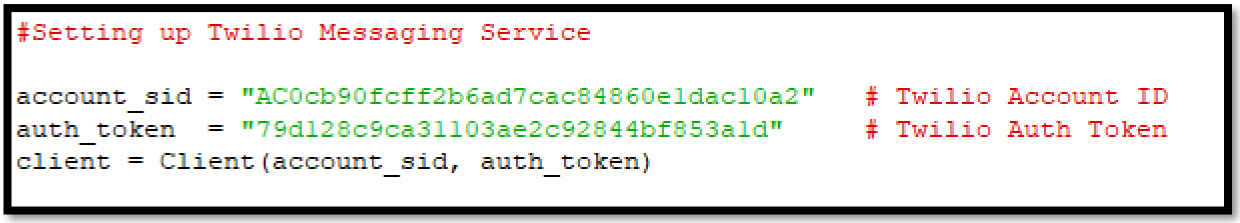
The server code will be hosted on the AWS and will be running 24x7. Whenever there's an emergency data sent from RPi, the data needs to be processed. When a new emergency data arrives, the code extracts the details of the emergency including the type of the emergency. Based on the type of emergency, the nearest rescue centre is found using Haversine formula. Fig 3.9 shows implementation of the Haversine formula.



```
#Haversine Calculation
def haversine(lat1, lon1, lat2, lon2):
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371              # Radius of earth in kilometers
    return c * r
```

Fig3.9: Haversine Formula to find nearest rescue centre

Once the nearest rescue centre is found, the rescue centre has to notified about the emergency. An SMS will be sent to the registered mobile number of the rescue centre. Twilio messaging service is used to send the SMS. First, a Twilio account has to be set up. Upon setting up the account, an account ID and authorization token will be generated. This has to be used in the code to setup the messaging services. Fig 3.10 shows the Authentication part.

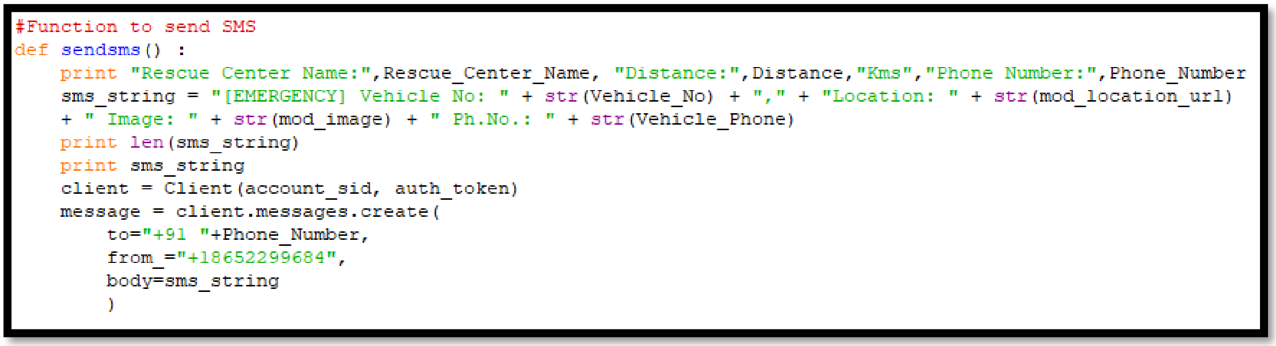


```
#Setting up Twilio Messaging Service

account_sid = "AC0cb90fcff2b6ad7cac84860e1dac10a2"   # Twilio Account ID
auth_token  = "79d128c9ca31103ae2c92844bf853a1d"     # Twilio Auth Token
client = Client(account_sid, auth_token)
```

Fig3.10: Twilio Authentication

Fig 3.11 shows the message sender function. In this function, the phone number purchased from Twilio will be used to send the SMS. The receiver phone number is obtained from the emergency message sent by RPi. The message body will involve all the necessary details like Vehicle Number, Emergency Image link, the location and the phone number.



```
#Function to send SMS
def sendsms() :
    print "Rescue Center Name:",Rescue_Center_Name, "Distance:",Distance,"Kms","Phone Number:",Phone_Number
    sms_string = "[EMERGENCY] Vehicle No: " + str(Vehicle_No) + "," + "Location: " + str(mod_location_url)
    + " Image: " + str(mod_image) + " Ph.No.: " + str(Vehicle_Phone)
    print len(sms_string)
    print sms_string
    client = Client(account_sid, auth_token)
    message = client.messages.create(
        to="+91 "+Phone_Number,
        from_="+18652299684",
        body=sms_string
        )
```

Fig3.11: Sending SMS

3.5.3 Rescue Centre Code

Whenever data is fetched from the firebase, the tags are compared before the value is obtained. Based on whether the tag is the required one the values are obtained and put in the particular label where it needs to be stored. So, the rescue centre personnel can view their previously received emergency requests on the Android Application. The Emergency Image can be viewed as well. Fig 3.12 shows how the Image URL obtained is converted to a button which upon clicking will redirect to the storage location of the image.

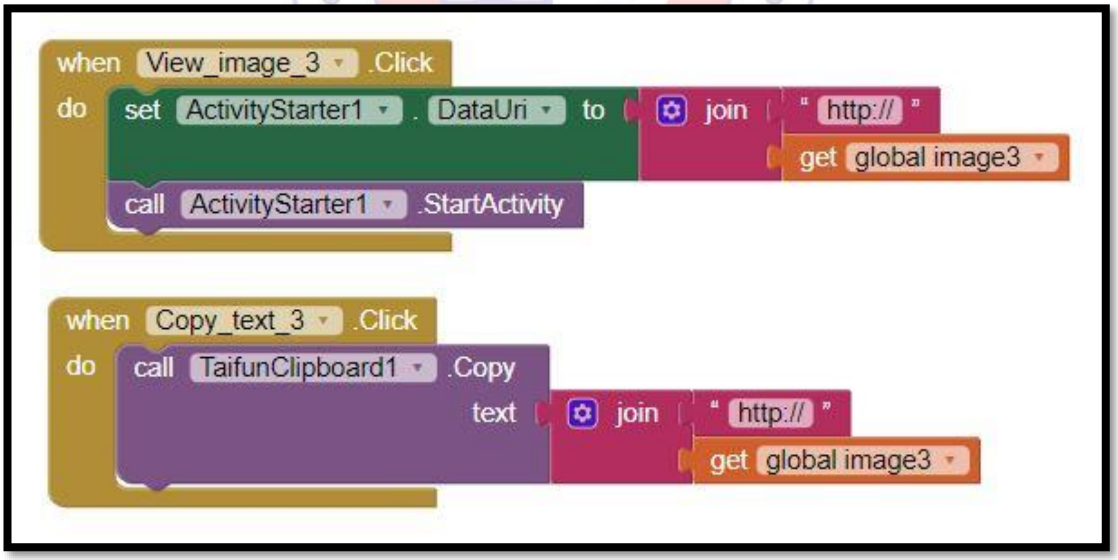

when View_image_3 .Click
do set ActivityStarter1 . DataUri to join " http:// "
get global image3
call ActivityStarter1 .StartActivity
when Copy_text_3 .Click
do call TaifunClipboard1 .Copy
text join " http:// "
get global image3

Fig3.12: Viewing Image

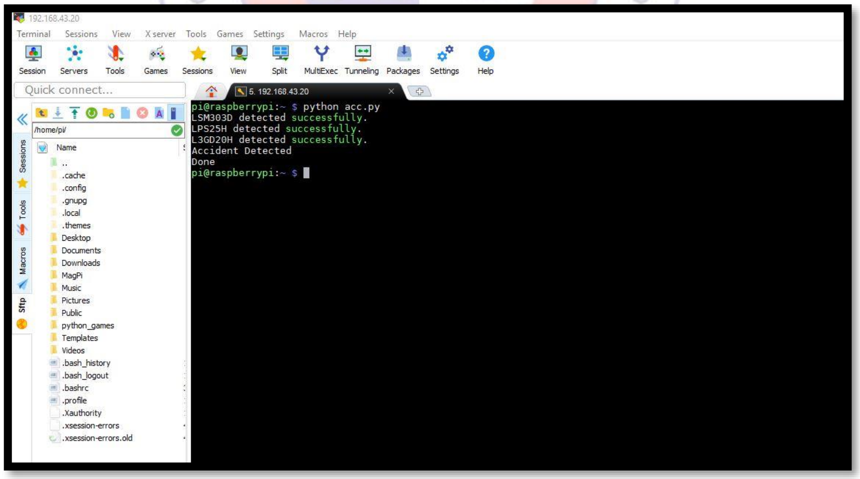
CHAPTER-4
RESULTS

Chapter 3
RESULTS

In this chapter, the implementation of the objectives mentioned in section 1.3 is discussed. The results are divided into different sections. In the first section, the situation node results will be discussed which involves detecting accident, capturing the image and uploading the same to Dropbox and finally the emergency message to firebase. The second section will talk about the server-side results. These include creating different databases, finding the nearest rescue centre and also sending SMS to the rescue centre. The mobile application result will be discussed in the third section. Finally, the last section, the prototype results will be discussed. This involves the design of the final prototype.

4.1 Situation Node Results

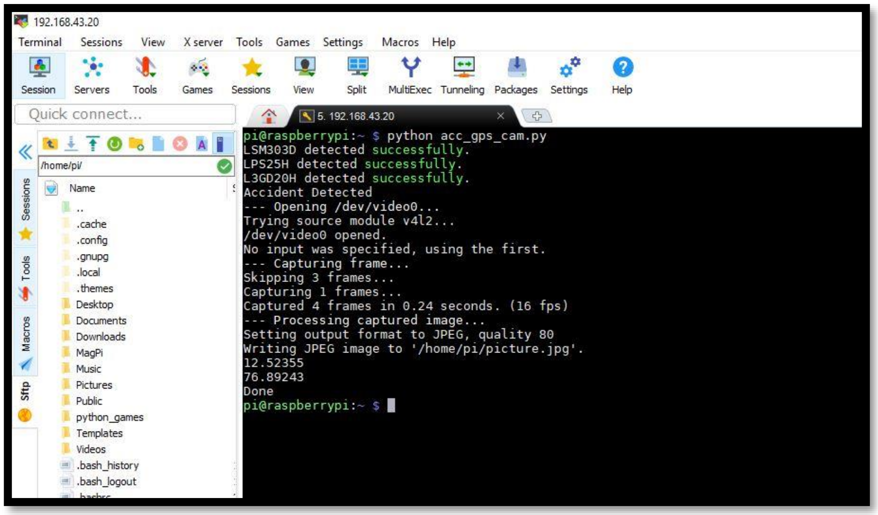
Fig 4.1 shows accelerometer and gyroscope being detected. Also, once an accident occurs, the module is able to detect it, as seen by the display message on the console. The module combines the accelerometer and gyroscope readings to detect an accident. Internally, the raw readings from accelerometer has to converted into roll, pitch and yaw as discussed in section 3.5.1.



```
pi@raspberrypi:~ $ python acc.py
LSM303D detected successfully.
LPS25H detected successfully.
L3GD20H detected successfully.
Accident Detected
Done
pi@raspberrypi:~ $
```

Fig4.1: Accident detection

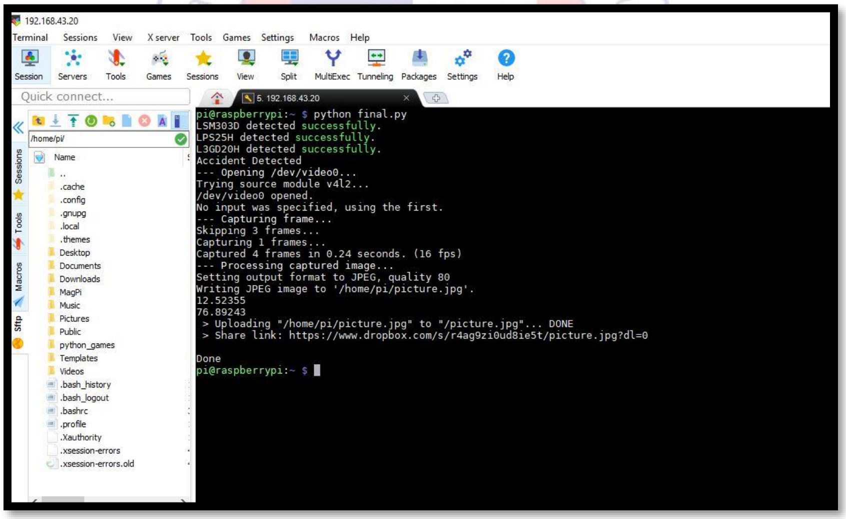
Once the accident is detected, the module has to trigger the webcam to capture the image and also fetch the co-ordinates from the GPS module. Fig 4.2 shows the USB camera capturing the image and also the fetching of latitude and longitude using *serial0* port of Raspberry Pi.



```
pi@raspberrypi:~ $ python acc_gps_cam.py
LSM303D detected successfully.
LPS25H detected successfully.
L3GD20H detected successfully.
Accident Detected
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Skipping 3 frames...
Capturing 1 frames...
Captured 4 frames in 0.24 seconds. (16 fps)
--- Processing captured image...
Setting output format to JPEG, quality 80
Writing JPEG image to '/home/pi/picture.jpg'.
12.52355
76.89243
Done
pi@raspberrypi:~ $
```

Fig4.2: Image capturing and location tracking

Once the image is captured and the co-ordinates are obtained, the module has to upload the data to Firebase. Before that, image will be uploaded to Dropbox and a public URL will be generated. Fig 4.3 shows the generation of sharable link and uploading the data along with the URL to Firebase.



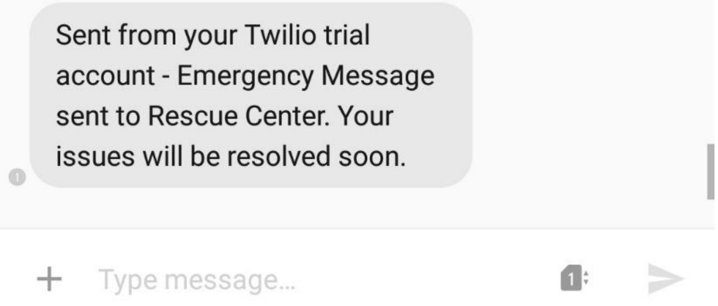
```
pi@raspberrypi:~ $ python final.py
LSM303D detected successfully.
LPS25H detected successfully.
L3GD20H detected successfully.
Accident Detected
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
--- Capturing frame...
Skipping 3 frames...
Capturing 1 frames...
Captured 4 frames in 0.24 seconds. (16 fps)
--- Processing captured image...
Setting output format to JPEG, quality 80
Writing JPEG image to '/home/pi/picture.jpg'.
12.52355
76.89243
 > Uploading "/home/pi/picture.jpg" to "/picture.jpg"... DONE
 > Share link: https://www.dropbox.com/s/r4ag9zi0ud8ie5t/picture.jpg?dl=0

Done
pi@raspberrypi:~ $
```

Fig4.3: Uploading data

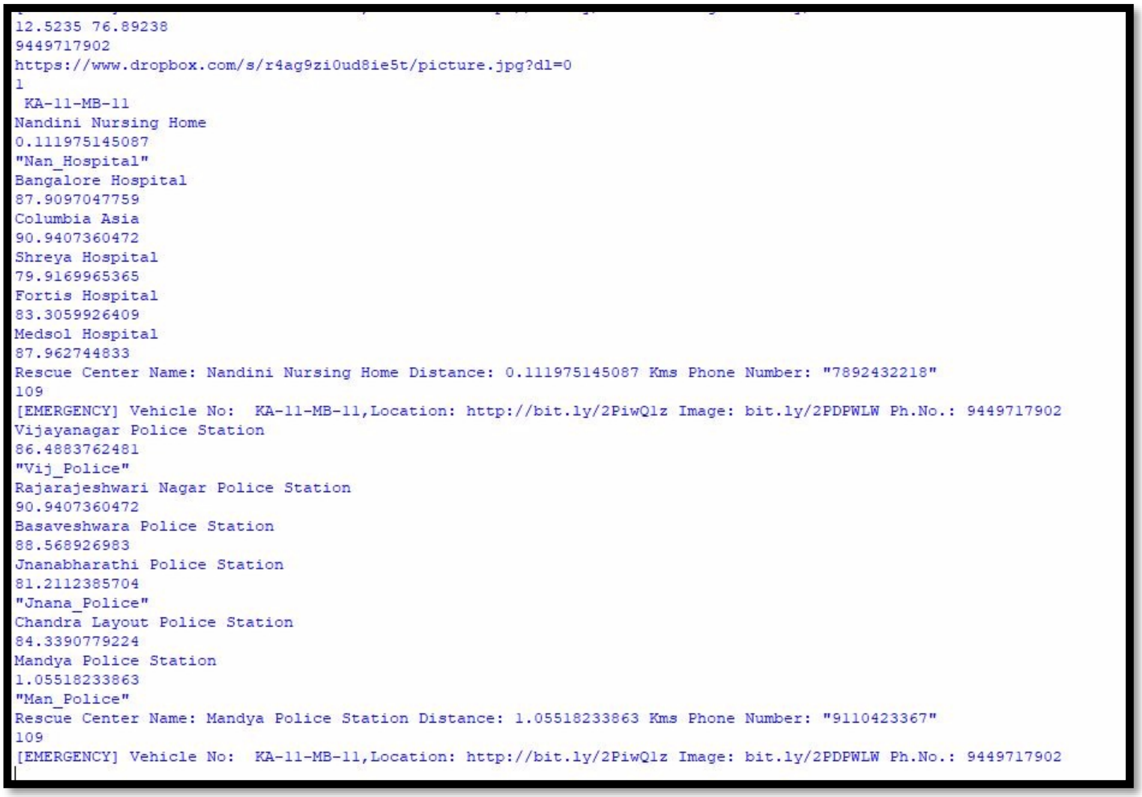
4.2 Server results

Once the data is uploaded to the Firebase, the server detects the emergency. Once it detects an emergency, it will fetch the details of the emergency. Based on the type of the emergency, it will find the nearest rescue centre of that particular type of emergency. Fig 4.5 shows the server finding the nearest hospital and the nearest police station for a type 1 emergency using the Haversine formula. Also, once it finds the nearest centre, it will send an SMS to that centre. Also, a confirmation message will be sent to the mobile number of the person who sent the emergency information. Fig 4.4 shows the acknowledgment message received by the user.



Sent from your Twilio trial account - Emergency Message sent to Rescue Center. Your issues will be resolved soon.

Fig4.4: Acknowledgement Message

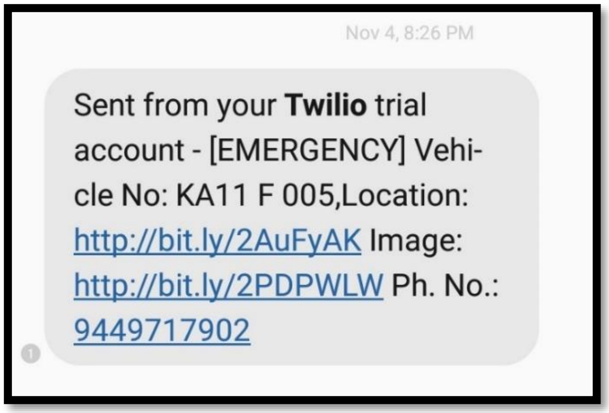


```
12.5235 76.89238
9449717902
https://www.dropbox.com/s/r4ag9zi0ud8ie5t/picture.jpg?dl=0
1
 KA-11-MB-11
Nandini Nursing Home
0.111975145087
"Nan_Hospital"
Bangalore Hospital
87.9097047759
Columbia Asia
90.9407360472
Shreya Hospital
79.9169965365
Fortis Hospital
83.3059926409
Medsol Hospital
87.962744833
Rescue Center Name: Nandini Nursing Home Distance: 0.111975145087 Kms Phone Number: "7892432218"
109
[EMERGENCY] Vehicle No:  KA-11-MB-11,Location: http://bit.ly/2PiwQlz Image: bit.ly/2PDPWLW Ph.No.: 9449717902
Vijayanagar Police Station
86.4883762481
"Vij_Police"
Rajarajeshwari Nagar Police Station
90.9407360472
Basaveshwara Police Station
88.568926983
Jnanabharathi Police Station
81.2112385704
"Jnana_Police"
Chandra Layout Police Station
84.3390779224
Mandya Police Station
1.05518233863
"Man_Police"
Rescue Center Name: Mandya Police Station Distance: 1.05518233863 Kms Phone Number: "9110423367"
109
[EMERGENCY] Vehicle No:  KA-11-MB-11,Location: http://bit.ly/2PiwQlz Image: bit.ly/2PDPWLW Ph.No.: 9449717902
```

Fig4.5: Nearest Rescue Centre

4.3 Rescue Centre Results

Fig 4.6 shows the SMS received at the rescue centre. The message will contain Vehicle number, location of the incident, image and the contact number.

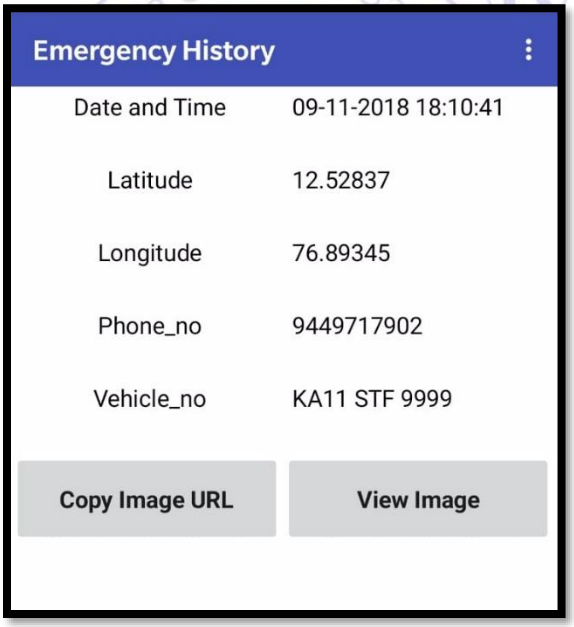


Nov 4, 8:26 PM

Sent from your **Twilio** trial account - [EMERGENCY] Vehicle No: KA11 F 005,Location: http://bit.ly/2AuFyAK Image: http://bit.ly/2PDPWLW Ph. No.: 9449717902

Fig4.6: SMS received

Fig 4.7 shows the Emergency information which was sent as an SMS being displayed on the rescue centre mobile application. This will be useful to the rescue centre to maintain a database. The image can be viewed by clicking the View Image button.



Emergency History

Date and Time	09-11-2018 18:10:41
Latitude	12.52837
Longitude	76.89345
Phone_no	9449717902
Vehicle_no	KA11 STF 9999

Fig4.7: Emergency message displayed on the app

4.3 Module Results

Fig 4.8 shows the final prototype developed. The four manual trigger buttons along with LED's to detect the trigger can be seen. Fig 4.9 shows the USB webcam and external GPS antenna being connected to the module. The external GPS antenna is connected to GPS module via SMA to uFL connector. Fig 4.10 shows the internal housing of the module. The sensors along with the Pi are rigidly mounted inside the module.

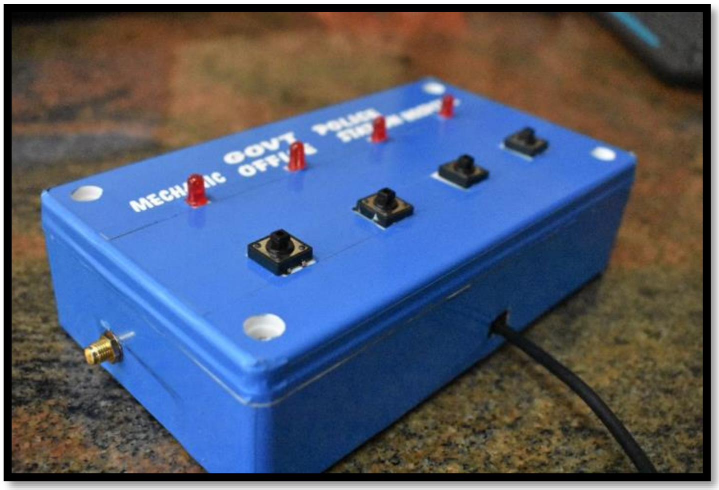


Fig4.8: Final Prototype

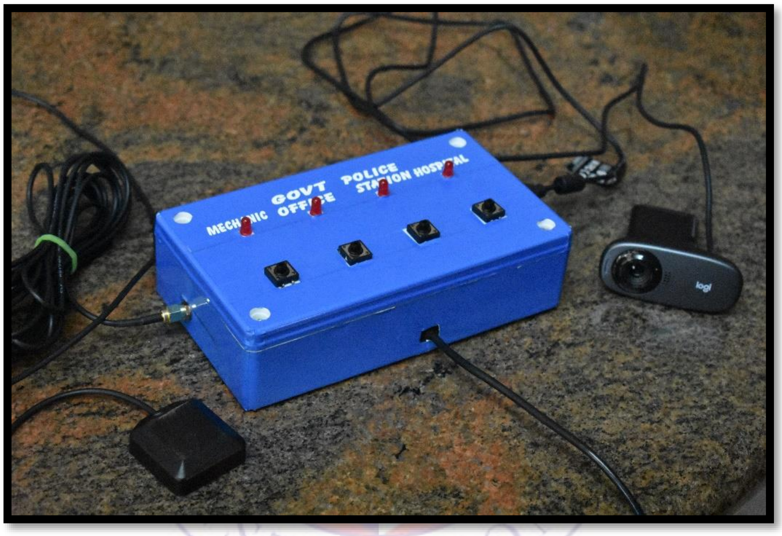


Fig4.9: Prototype with Webcam and External GPS Antenna

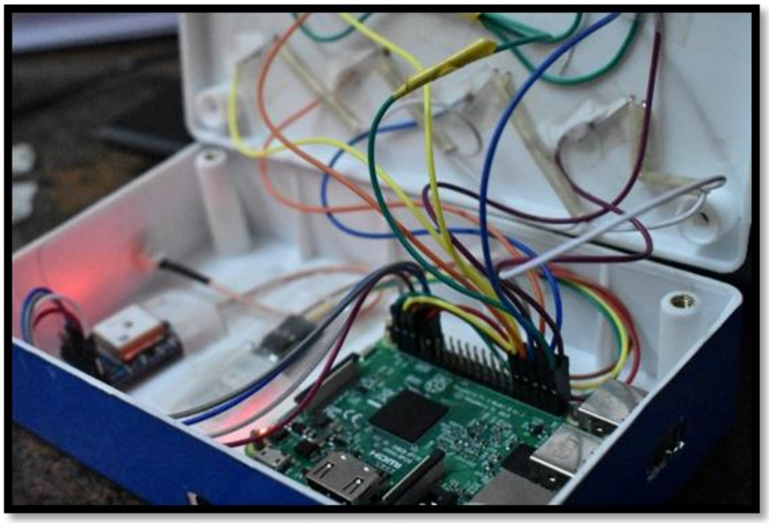


Fig4.10: Inside the Prototype

CHAPTER-5
Conclusion and Future Scope

Chapter 5
CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

Proposed emergency communication system is able to send automatic message to the control room with all relevant information of an emergency. The nearest one is calculated from the database and an automatic message is sent to them. The initial image taken by the car camera is also shown in the interface that can help the authorities to understand the situation.

Compared to other systems, it is an automated system that can automatically calculate the nearest rescue centre and send emergency message. Apart from accidents, this system provides other means of emergency options to the driver for other common emergencies that can take place on road.

This prototype is mainly designed for smart cities and IoT enabled vehicles. However, this system may also be used with existing infrastructure in any cities. This proposed system is only able to send emergency information from a vehicle to nearby rescue centres, but it can’t help to avoid any emergency issues. Also, the system is dependent on several mechanical and electrical devices in a car to detect accident or other emergencies.

Table 3: Comparison with other models

Emergency System	Automatic/ Manual	On board or not	Type of emergencies	Contact authorities
Proposed system	Fully automated	Integrated to vehicle	5	Directly to nearest centre
Onstar by GM	Partially automated	Integrated to vehicle	5	GM Customer Care
Ford Sync	Partially automated	Synced with phone	1	Ford Customer Care
ARRS	Fully automated	Fixed on traffic areas	1	Respective authorities

5.2 Future Scope

As convenient as the system maybe, it does however a few flaws which can be have rectified in future developments.

- The system can be improved to provide a better GUI to rescue centre by providing the option of either accepting or rejecting as a response to the user so that the next nearest centre can be calculated.
- The rescue centre app can be further improved upon by developing in android studio and possibly released as an iOS app in the future with new features for user safety.
- The proposed system has certain drawbacks due to its dependency on mechanical and electrical devices in a car to detect accident or other emergencies, thus in future more accurate devices providing fewer errors can be considered.
- The functionalities of the model can be designed on to a chip (System-on-Chip/ Network-on-Chip) system featuring these services in future.
- Display modules like LCD/LED can be embedded with the system to enhance the GUI.

References

[1] *ctp.gov.in*,2018, [Online], Available: http://www.ctp.gov.in/PressNote/2016/SupremeCourtGuidelinesonRoadSafety.pdf.

[2] J. Maleki, E. Foroutan, and M. Rajabi, "Intelligent Alarm System for Road Collision", Journal *of Earth Science and Engineering,* vol. 1, no. 3, Ju2011.

[3] C. Thompson, J. White, B. Dougherty, A. Albright, and D. C. Schmidt, "Using Smartphones and Wireless Mobile Sensor Networks to Detect Car Accidents and Provide Situational Awareness to Emergency Responders," *Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Conf*, June 2010.

[4] Y. Ki and D. Lee, "A Traffic Accident Recording and Reporting Model at Intersections", *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 188-194, 2007.

[5] S. Koley and P. Ghosal, "Addressing Hardware Security Challenges in Internet of Things: Recent Trends and Possible Solutions", *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015.

[6] S. Lee, G. Tewolde and J. Kwon, "Design and implementation of vehicle tracking system using GPS/GSM/GPRS technology and smartphone application", *2014 IEEE World Forum on Internet of Things (WF-IoT)*, 2014.

[7] N. Watthanawisuth, T. Lomas and A. Tuantranont, "Wireless black box using MEMS accelerometer and GPS tracking for accidental monitoring of vehicles", *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*, 2012.

[8] P. Verma and J.S Bhatia, “Design and Development of GPS-GSM based tracking system with Google map-based monitoring”, *International Journal of Computer Science, Engineering and Applications,* vol.3, no. 3, pp. 33-40, May 2013.

[9] B. Kodavati, V.K.Raju, S.S. Rao, A.V. Prabu, T.A. Rao and Dr.. Y. V. Narayana,” GPS and GSM Based Vehicle Location and Tracking System”, *International Journal of Engineering Research and Applications*, vol.1, no. 3, pp.616- 625, Mar. 2013.

[10] A. Ali and M. Eid, "An automated system for Accident Detection", *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, 2015.

[11] A. Baramy, P. Singh, A. Jadhav, K. Javir, and S. Tarleka, “Accident detection & Alerting system,” *International Journal of Technical Research and Applications,* e-ISSN: 2320-8163, Mar. 2016.

[12] B. Fernandes, V. Gomes, and J. Ferreira, “Mobile application for automatic accident detection and multimodal alert,” *Traffic Accident Detection Through a Hydrodynamic Lens 2015 IEEE 81st Vehicular Technology Conference (VTC Spring),* May 2015.

[13] H. Ullah, M. Ullah, H. Afridi, N. Conci, and F.G. De Natale, “Traffic accident detection through a hydrodynamic lens”, *IEEE International Conference on Image Processing (ICIP)*, Sep. 2015.

[14] *Owner.ford.com*, 2018. [Online]. Available: https://owner.ford.com/how-tos/sync-technology/sync/settings/911-assist-overview.html.

[I5] Ch. Ramya Keerthi, G. Shanmukh, Dr. R. Sivaram, "Various Accident Detection Technologies and Recovery Systems with Victim Analysis "*International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE)*, May 2014

[16] S. M. Tang and H. J. Gao, "Traffic-incident detection-algorithm based on nonparametric regression," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, 2015, pp. 38-42.

[17] Li Chuan-zhi, Hu Ru-fu and H. Ye, "Method of freeway incident detection using wireless positioning", *2008 IEEE International Conference on Automation and Logistics*, 2008.

[18] S. Prabakar. K. Porkumaran, U. Samson and J. Guna Sundari, “An enhanced accident detection and victim status indicating system: Prototype", *Conference on India Conference (INDlCON),* ppno: 978-1-4673-2270-6, 2014.

[19] J. Zaldivar, C. Calafate, J. Cano and P. Manzoni, "Providing accident detection in vehicular networks through OBD-II devices and Android-based smartphones", *2011 IEEE 36th Conference on Local Computer Networks*, 2011.

[20] F. Basheer, J. Alias, C. Favas, V. Navas, N. Farhan and C. Raghu, "Design of accident detection and alert system for motor cycles", *2013 IEEE Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS)*, 2013.

[21] S. K. Jose, X. A. Mary, N. Mathew, "ARM 7 Based Accident Alert and Vehicle Tracking System" *International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075*, vol 2, no. 4, pp. 93-96, March 2014.

[22] M. Fogue, P. Garrido, F. Martinez, J. Cano, C. Calafate and P. Manzoni, "A System for Automatic Notification and Severity Estimation of Automotive Accidents", *IEEE Transactions on Mobile Computing*, vol. 13, no. 5, pp. 948-963, 2014.

[23] M. Syedul Amin, J. Jalil and M. Reaz, "Accident detection and reporting system using GPS, GPRS and GSM technology", *2012 International Conference on Informatics, Electronics & Vision (ICIEV)*, 2012.

[24] D. Acharya, V. Kumar, N. Garvin, A. Greca and G. Gaddis, "A sun SPOT based automatic vehicular accident notification system", *2008 International Conference on Technology and Applications in Biomedicine*, 2008.

[25] A. Puscasiu, A. Fanca and H. Valean, "Tracking and localization system using Android mobile phones", *2016 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 2016.

[26] F. Aloul, I. Zualkernan, R. Abu-Salma, H. Al-Ali, M. Al-Merri, “iBump: Smartphone Application to Detect Car Accidents”, *Proc. of IAICT 2014*, pp. 52-56, Aug. 2014.

[27] M. Bhokare, S. Kaulkar, A. Khinvasara, A. Agrawal, Y. K. Sharma,”An Algorithmic Approach for Detecting Car Accidents using Smartphone”, *International Journal of Research in Advent Technology*, vol.2, No.4, April 2014, pp. 151-154, E-ISSN: 2321-9637

[28] A. Meena, S. Iyer, M. Nimje, S. Joglekar, S. Jagtap and M. Rahman, "Automatic Accident Detection and reporting framework for two wheelers", *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, 2014.