

## 1. PROJECT OVERVIEW

In this system using Raspberry pi, GSM network, accelerometer, GPS tracking system, Ultrasonic sensors a device developed for emergency assistance for the driver and accident monitoring. In the event of accident, this device will send short message indicating the position of vehicle to family member and a call to emergency medical service (EMS) with the accident location so as to provide assistance to the driver. It also provides the driver with distance of vehicles around it with help of a voice message .This device can also be used as a media center which can be used to as listen to music watch videos etc. The biggest advantage of this device is that it is cheap and affordable to all the people who drive a vehicle.

## 2. INTRODUCTION AND MOTIVATION

### 2.1. Theory behind the Project Concept

EASV is used for quick assistance to the driver in the case of accidents. Despite awareness campaign, accidents are still increasing due to driver's poor behaviors such as speed driving, drunk driving, riding with no helmet protection, riding without sufficient sleep etc. The numbers of death and disability are very high because of late assistance to people who were involved in the accident. Hence EASV is used to provide assistance for people who get into accidents.

EASV can also be used to avoid accidents. The ultrasonic distance measuring sensor will let driver to know about the nearby vehicles, so that driver will get alert and so can avoid the accident same also can also be applied for parking assistance. EASV uses Raspberry PI which can be further used with Raspbian OS to run large number of applications including Gmail, Facebook, online radio, You-tube, etc.

It can play almost any media formats, including playing from archives and iso files, it allows passengers in the car to control the media center (play media, create playlist, browse files).

## 2.2. Problem Definition

Designing a system which will alert driver about the distance of nearby vehicles and if at all accident happens, then provide information about the location to their family members.

Application includes Python on Raspbian to provide simple GUI and for simple programming, Ultrasonic sensor HC-SR04 to provide the distance of nearby vehicles, 3-axis accelerometer with high resolution (13-bit) measurement at up to  $\pm 16$  g, which also detects shock or impact and GSM module which is used to send message.

## 2.3. Need for Project

The number of accidents occurring in India and the resulting deaths or injuries have always been high. Despite awareness campaign, this problem is still increasing due to rider's poor behaviors such as speed driving, drunk driving, riding with no helmet protection, riding without sufficient sleep etc. The numbers of death and disability are very high because of late assistance to people who are involved in an accident. These cause huge social and economic burdens to people involved. Therefore, several research group and major vehicle manufacturers including Honda have developed safety devices to protect drivers from accidental injuries. However, good safety device for vehicles is difficult to implement and very expensive. Alternatively, intelligence schemes such as incident detection with tracking system have also recently been devised to notify the accident to related people so that quickest assistance can reach people who got the accident. Presently, tracking system is only installed in some high end vehicles because these systems are still too expensive for most drivers. Thus, accident detection has recently gained attention because these systems are expected to save life by helping the drivers to get medical treatment on time.

## 2.4. Challenges

---

There are many challenges associated with the accuracy, integration and usefulness of the system. For this system there are limitations on the equipment used because the aim was to make cheaper product. Making system work in a remote area becomes difficult due to unavailability of cellular phone network. The major challenge regarding this project is accuracy of sensor and working procedure because even in the case of fatal accidents there is always a chance of false alarm.

## 3. ANALYSIS AND DESIGN

### 3.1. Software Development

Software development is the development of a software product. The term "software development" may be used to refer to the activity of computer programming, which is the process of writing and maintaining the source code, but in a broader sense of the term it includes all that is involved between the conception of the desired software through to the final manifestation of the software, ideally in a planned and structured process. Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products.

Software can be developed for a variety of purposes, the three most common being to meet specific needs of a specific client/business, to meet a perceived need of some set of potential users, or for personal use (e.g. a scientist may write software to automate a mundane task). Embedded software development, that is, the development of embedded software such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product.

There are several different approaches to software development, much like the various views of political parties toward governing a country. Some take a more structured, engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece-by-piece. Most methodologies share some combination of the following stages of software development:

- Analyzing the problem
- Market research
- Gathering requirements for the proposed business solution
- Devising a plan or design for the software-based solution
- Implementation (coding) of the software
- Testing the software
- Deployment
- Maintenance and bug fixing

There are significant advantages and disadvantages to the various methodologies, and the best approach to solving a problem using software will often depend on the type of problem. If the

problem is well understood and a solution can be effectively planned out ahead of time, the more "waterfall" based approach may work the best. If, on the other hand, the problem is unique (at least to the development team) and the structure of the software solution cannot be easily envisioned, then a more "extreme" incremental approach may work best.

## 3.2. Flow of Project

### 3.2.1. Literature Survey

**Paper:** Wireless Black Box Using MEMS Accelerometer and GPS Tracking for Accidental Monitoring of Vehicles **Published in:** International Journal of Computer Science and Mobile Computing

**Date of Publication:** March 2014

**Authors:** Pallakila Sai Avinash,K.ThenKumari

**Summary:**

An innovative wireless black box using MEMS accelerometer and GPS tracking system has been developed for vehicle accidental monitoring. The system can detect accident from accelerometer signal using threshold algorithm, after accident is detected, short alarm message data (alarm message and position of accident) will be sent via GSM network. The safety and rescue are the primary concern in every part of fast moving world. There are many accidental event occur due to an unavoidable reasons. Though the occurrence of accident is quite unavoidable, this innovative project is challengingly undertaken to make the change in worst scenario by providing importance to alerting, monitoring and tracking the location of an event. Which would in turn provides efficient quick response for rescue process to be carried out without any latency.

### 3.2.2. Component Review

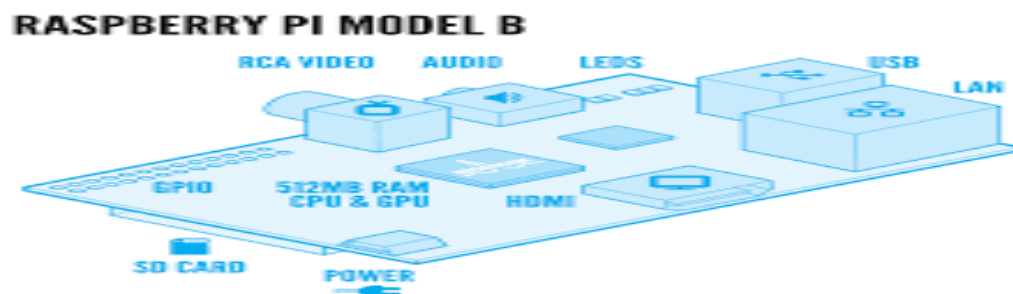
#### **Raspberry Pi**

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

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras.

The Raspberry Pi measures 85.60mm x 56mm x 21mm (or roughly 3.37" x 2.21" x 0.83"), with a little overlap for the SD card and connectors which project over the edges. It weighs 45g.

Fig 3.2.2.1 Raspberry PI model B



- **HOW POWERFUL IS IT?**

The GPU provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile encode and decode.

The GPU is capable of 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs of general purpose compute and features a bunch of texture filtering and DMA infrastructure.

This means that graphics capabilities are roughly equivalent to the original Xbox's level of performance. Overall real world performance is something like a 300MHz Pentium 2, only with much, much swankier graphics.

- **WHAT HARDWARE INTERFACES DOES IT HAVE?**

The Raspberry Pi has 26 dedicated GPIO pins, including a UART, an i2c bus, a SPI bus with two chip selects, i2s audio, 3v3, 5v, and ground.

The maximum number of GPIOs can theoretically be indefinitely expanded by making use of the i2c or SPI bus.

- **WHAT ARE THE POWER REQUIREMENTS?**

The device is powered by 5v micro USB. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you hook up to it. We have found that purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi for most applications, though you may want to get a 2.5A (2500mA) if you want to use all 4 USB ports on the Model B without using an external powered USB hub.

Typically, the model B uses between 700-1000mA depending on what peripherals are connected, the Model B+ has slightly more efficient power circuits as well as more available USB ports and can use between 600-2000mA, and the model A can use as little as 500mA with no peripherals attached. The maximum power the Raspberry Pi Model A and B can use is 1 Amp, so if you need to connect a USB device that will take the power requirements of the Raspberry Pi above 1 Amp then you must connect it to an externally powered USB hub. Alternatively, the maximum power the Model B+ can use is 2 Amps before needing to connect devices to an externally powered USB hub.

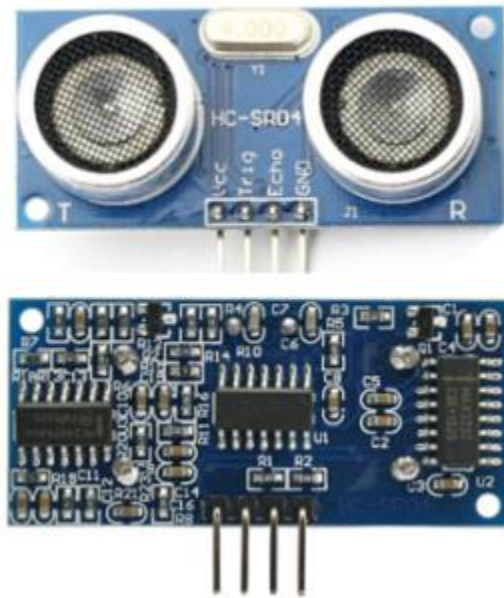
The power requirements of the Raspberry Pi increase as you make use of the various interfaces on the Raspberry Pi. The GPIO pins can draw 50mA safely (that is 50mA distributed across all the pins! An individual GPIO pin can only safely draw 16mA), the HDMI port uses 50mA, the camera module requires 250mA, and keyboards and mice can take as little as 100mA or over 1000mA! Check the power rating of the devices you plan to connect to the Pi and purchase a power supply accordingly. If you're not sure, buy a powered hub.

### **Ultrasonic sensor HC-SR04**



The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. From 2cm to 400 cm or 1" to 13 feet. Its operation is not affected by sunlight or black material like Sharp rangefinders are (although acoustically soft materials like cloth can be difficult to detect). It comes complete with ultrasonic transmitter and receiver module.

Fig 3.2.2.2 Ultrasonic sensor



- **Pins**
  - VCC: +5VDC
  - Trig : Trigger (INPUT)
  - Echo: Echo (OUTPUT)
  - GND: GND

### **ADXL345 - Triple-Axis Accelerometer**

The ADXL345 is a low-power, 3-axis MEMS accelerometer modules with both I2C and SPI interfaces. The Adafruit Breakout boards for these modules feature on-board 3.3v voltage regulation and level shifting which makes them simple to interface with 5v microcontrollers such as the Arduino.

The ADXL345 features 4 sensitivity ranges from +/- 2G to +/- 16G. And it supports output data rates ranging from 10Hz to 3200Hz.

Fig 3.2.2.3 ADXL345 Digital Accelerometer



### **How does it work**

The sensor consists of a micro-machined structure on a silicon wafer. The structure is suspended by polysilicon springs which allow it to deflect smoothly in any direction when subject to acceleration in the X, Y and/or Z axis. Deflection causes a change in capacitance between fixed plates and plates attached to the suspended structure. This change in capacitance on each axis is converted to an output voltage proportional to the acceleration on that axis.

## Adafruit Ultimate GPS

Fig 3.2.2.4 Adafruit Ultimate GPS



The Ultimate GPS Breakout V3 is a hard to find good, reliable GPS module for microcontroller use, which is why haven't carried one. Most are complicated to use, power hungry and require a completely clear view of the sky to get a decent fix. But when we tested the Adafruit Ultimate GPS, we knew it was worthy enough to be in the Maker Shed.

Not only is this module easy to use, it comes fully loaded. The Ultimate GPS breakout is based on the MTK3339 chipset which can track up to 22 satellites on 66 channels, has a high-sensitivity receiver and a built in -165 dB antenna. It's capable of 10 Hz updates, has a position accuracy of 1.8 meters, a velocity accuracy of .1 meters per second, and it only draws 20ma of current. It's also been successfully tested at over 88,000 feet. There's even options to use a battery to power the RTC for starts, a jack for an external antenna, a pulse per second output, and an output for adding an external LED to indicate a fix.

By far the most interesting feature of the Ultimate GPS Breakout though is it's built in data logger. The module includes an on-board microcontroller and enough FLASH memory to log the time, date, longitude, latitude and height every 15 seconds for up to 16 hours.

### 3.2.3. Feasibility Study

A Feasibility study is defined as an evaluation or analysis of the potential impact of a proposed project or program. A Feasibility study is conducted to assist decision-makers in determining whether or not to implement a particular project or program.

Feasibility study is divided into the following categories for a project:

- **Technical Feasibility**

Technical Feasibility is defined as whether reliable hardware and software as well as other technical resources capable of meeting the needs of a proposed system can be acquired or developed by an organization in the required time. Our application, EASV, will be developed using Raspberry Pi OS Raspbian and Python.

- **Economic Feasibility**

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. The procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. The economic feasibility of our project is nil.

- **Behavioral Feasibility**

The application will be easy to use for the users because of its simple GUI. No special training is required to use this application. Users need to have basic knowledge of using computer.

### 3.2.4. Cost Analysis

---

#### **Function Point Analysis**

Function Point Analysis (FPA) is an ISO recognized method to measure the functional size of an information system. The functional size reflects the amount of functionality that is relevant to and recognized by the user in the business. It is independent of the technology used to implement the system.

The process of performing Function Point Analysis is called a Function Point Count and it involves the **identification**, **classification** and **weighting** of each of these *transactions* and *data* components. The weightings are combined to give the functional size as an Unadjusted Function Point Count. An additional step, involves assessing the technical and quality features embedded in the software product and adjusting the functional size accordingly. The result is referred to as the Adjusted Function Point Count.

**EI - external inputs**, which are the components responsible for introducing changes in system's internal data.

**EO - external outputs**, which are the ways system's internal data can be presented, but beware - there are a few similarities with EQ components, though.

**EQ - external inquiries**, which are the methods for reading system's data without modifying it.

**EIF - external interface files**, which are responsible for exchanging data with other systems.

**ILF - internal logical files**, which are files that are being used by system itself.

**Domain Characteristic Table**

| MEASUREMENT PARAMETER         | COUNT<br>(value >= 0) | WEIGHTING FACTOR                 |                                  |                       |
|-------------------------------|-----------------------|----------------------------------|----------------------------------|-----------------------|
|                               |                       | Simple                           | Average                          | Complex               |
| Number of User Input          | 0                     | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Number of User Outputs        | 3                     | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| Number of User Inquiries      | 0                     | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| Number of Files               | 14                    | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| Number of External Interfaces | 4                     | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |

**Complexity Adjustment Table**

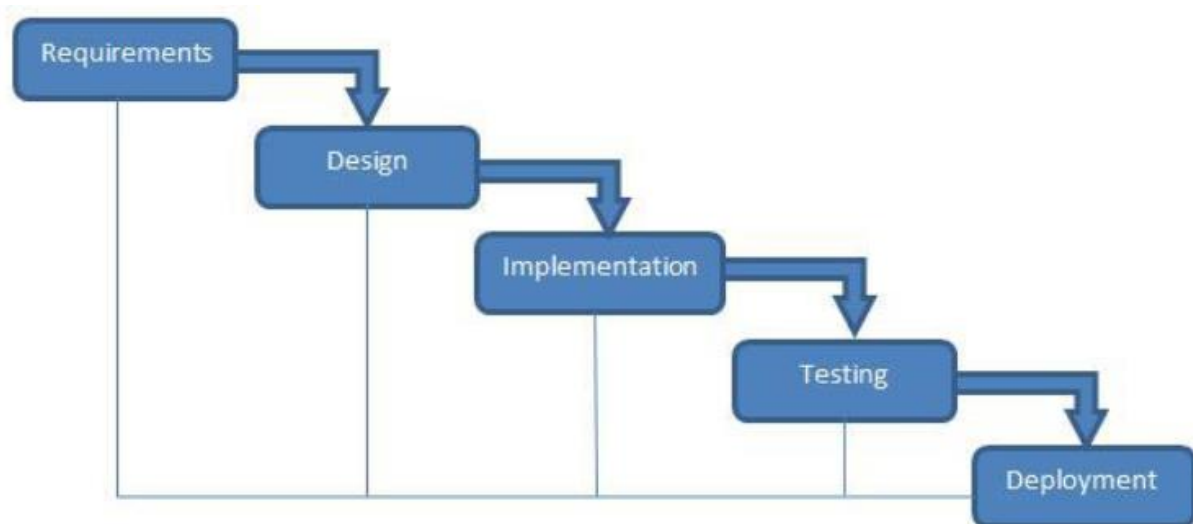
| ITEM | COMPLEXITY ADJUSTMENT QUESTIONS  | SCALE                            |                                  |                                  |                                  |                                  |                       |
|------|--|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|-----------------------|
|      |  | No Influence<br>0                | 1                                | 2                                | 3                                | 4                                | Essential<br>5        |
| 1    | Does the system require reliable backup and recovery?  | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| 2    | Are data communications required?  | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/> |
| 3    | Are there distributed processing functions?  | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 4    | Is performance critical?   | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 5    | Will the system run in an existing, heavily utilized operational environment?                              | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| 6    | Does the system require on-line data entry?  | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 7    | Does the on-line data entry require the input transaction to be built over multiple screens or operations? | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 8    | Are the master files updated on-line?  | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 9    | Are the inputs, outputs, files or inquiries complex?   | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 10   | Is the internal processing complex?  | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 11   | Is the code to be designed reusable?   | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| 12   | Are conversion and installation included in the design?  | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |
| 13   | Is the system designed for multiple installations in different organizations?                              | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/> |
| 14   | Is the application designed to facilitate change and ease of use by the user?                              | <input type="radio"/>            | <input type="radio"/>            | <input checked="" type="radio"/> | <input type="radio"/>            | <input type="radio"/>            | <input type="radio"/> |

**PROJECT FUNCTION POINTS: 164.7**

### 3.2.5. Process Model

In the given project the requirements will be fixed and there is no near chance of changing it. Waterfall approach was first Process Model to be introduced and followed widely in Software Engineering to ensure success of the project. In — The Waterfall approach, the whole process of software development is divided into separate process phases. The phases in Waterfall model are: Requirement Specifications phase, Software Design, Implementation and Testing & Maintenance. All these phases are cascaded to each other so that second phase is started as and when defined set of goals are achieved for first phase and it is signed off, so the name —Waterfall Model. All the methods and processes undertaken in Waterfall Model are more visible.

Fig 3.2.5 Waterfall model



**The stages of waterfall model are as follow:**

#### **Requirement Analysis & Definition**

All possible requirements of the system to be developed are captured in this phase. Requirements are set of functionalities and constraints that the end-user (who will be using the system) expects from the system. 18

The requirements are gathered from the end-user by consultation, these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is

also studied. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

### **System & Software Design**

Before a starting for actual coding, it is highly important to understand what we are going to create and what it should look like? The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

### **Implementation & Unit Testing**

On receiving system design documents, the work is divided in modules/units and actual coding is started. The system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality; this is referred to as Unit Testing.

### **Integration & System Testing**

As specified above, the system is first divided in units which are developed and tested for their functionalities. These units are integrated into a complete system during Integration phase and tested to check if all modules/units coordinate between each other and the system as a whole behaves as per the specifications. After successfully testing the software, it is delivered to the customer.

### **Operations & Maintenance**

This phase of — The Waterfall Model is virtually never ending phase.

Generally, problems with the system developed (which are not found during the development life cycle) come up after its practical use starts, so the issues related to the system are solved after deployment of the system. Not all the problems come in picture directly but they arise time to time and needs to be solved; hence this process is referred as Maintenance.



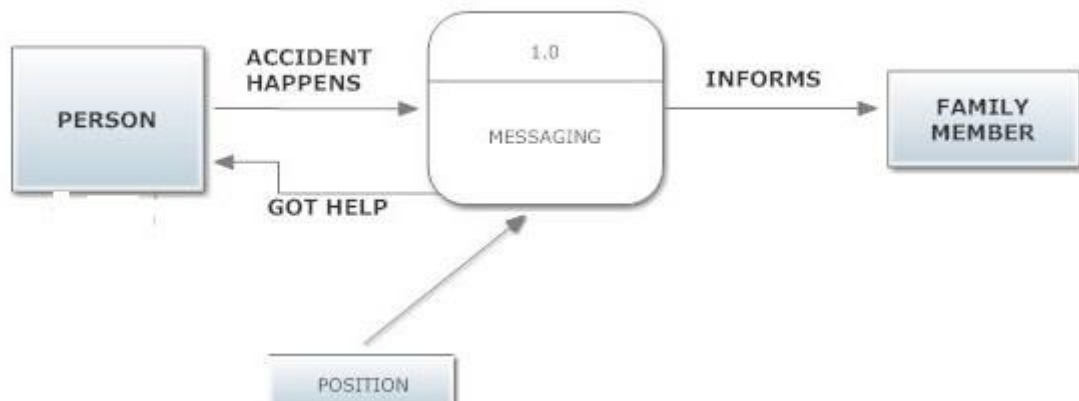
### 3.2.6. Data Flow Diagrams

Fig 3.2.6. Data flow diagrams

LEVEL 0 DFD

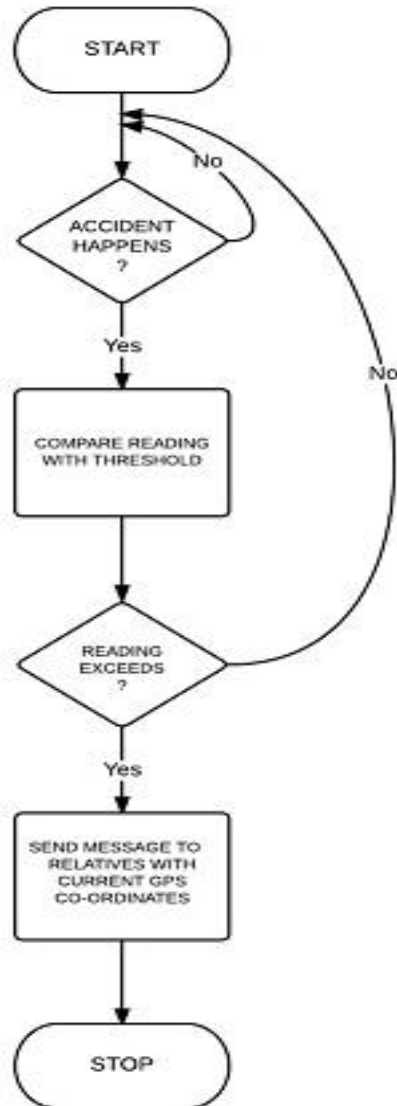


LEVEL 1 DFD



### 3.3. Flow chart of operation

Fig 3.3 Flow Chart



### 3.4. Technologies Used

#### 3.4.1. Hardware & Software Requirements

##### **Hardware**

Raspberry PI Model B

Micro SD card

Ultrasonic sensor HC-SR04

330 Ohm Resistor

470 Ohm Resistor

ADXL345 - Triple-Axis Accelerometer

Adafruit Ultimate GPS Breakout

Huawei e173 dongle

Prepaid SIM card

Personal Computer

Connection wires

Breadboard

##### **Software**

Raspbian OS

Python

OMX player

Tom's planner

Text editor

### 3.4.2. Introduction to Programming Tools

#### **Python**

Python is a great object-oriented, interpreted, and interactive programming language. It is often compared to Lisp, Tcl, Perl, Ruby, C#, Visual Basic, Visual Fox Pro, Scheme or Java7.

Python combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems. New built-in modules are easily written in C or C++ (or other languages, depending on the chosen implementation). Python is also usable as an extension language for applications written in other languages that need easy-to-use scripting or automation interfaces.

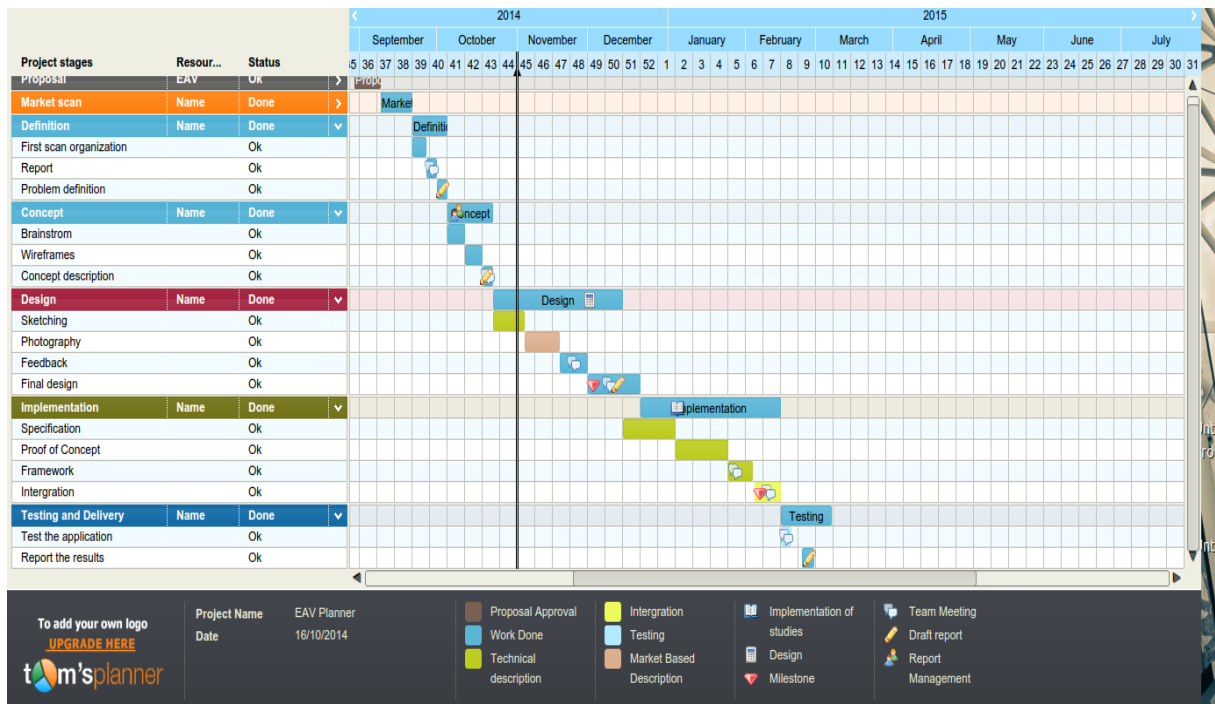
#### **Python GPIO**

Using Python on the Raspberry Pi opens up the opportunity to connect to the real world through the GPIO pins on raspberry Pi. This can be done with the RPi GPIO library. It is preinstalled on recent Raspbian images

## 4. PROJECT TIME & TASK DISTRIBUTION

### 4.1. Timeline Chart

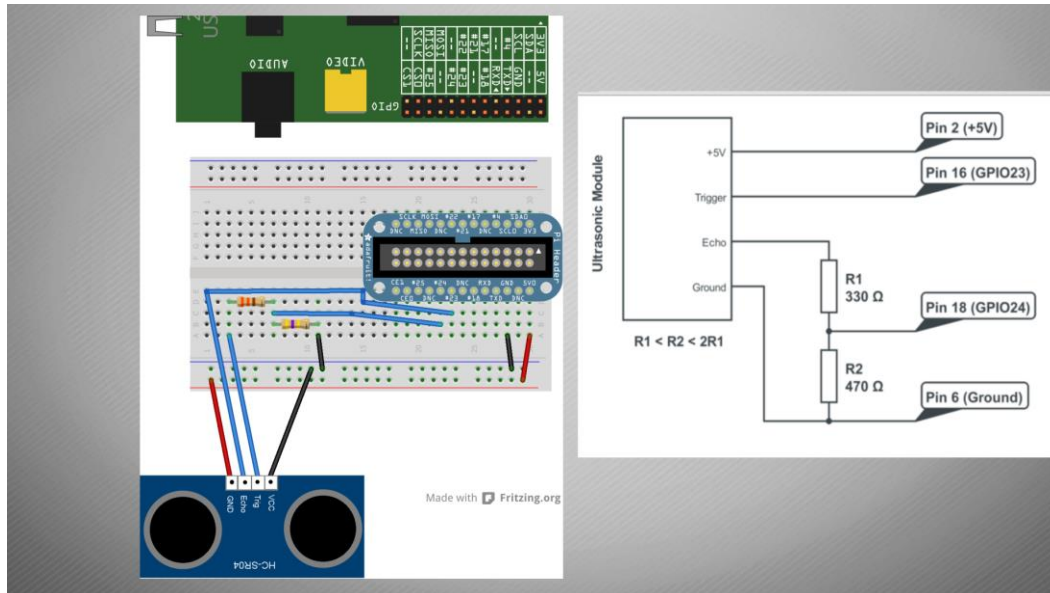
Fig 4.1 Timeline Chart



## 5. IMPLEMENTATION

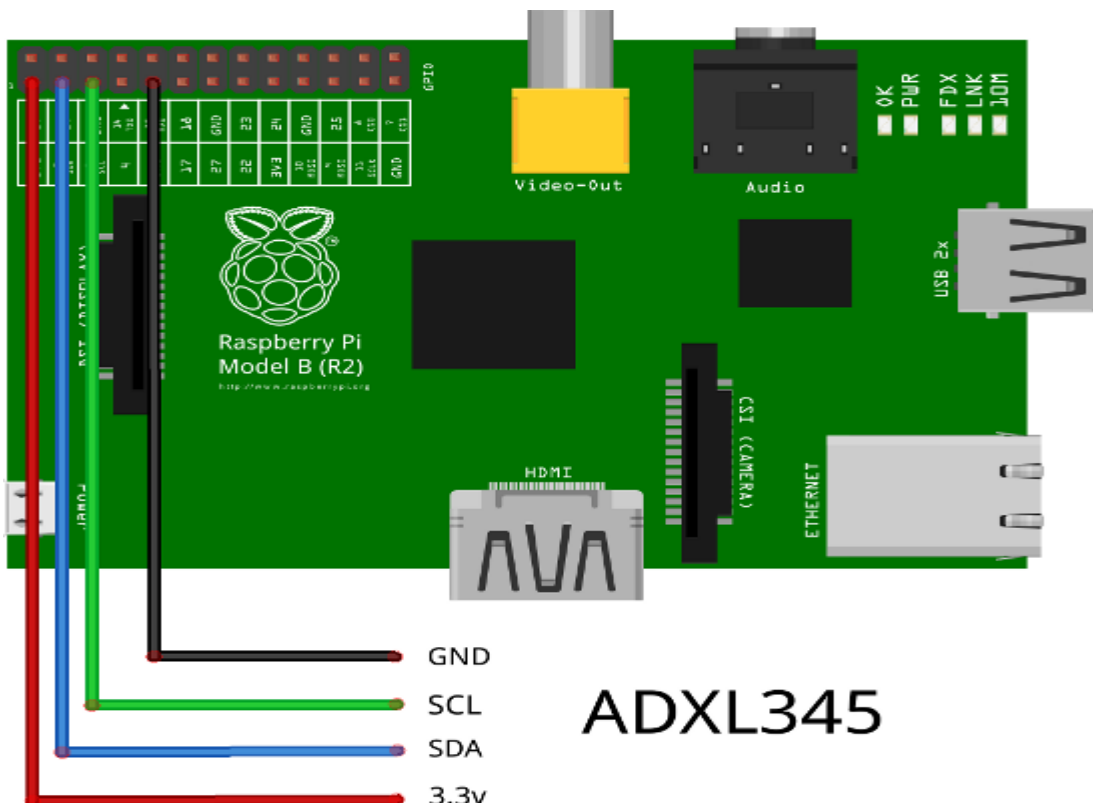
### Ultrasonic sensor connection

Fig 5.1 Ultrasonic sensor connection



### Accelerometer connection.

Fig 5.2 Accelerometer sensor connection



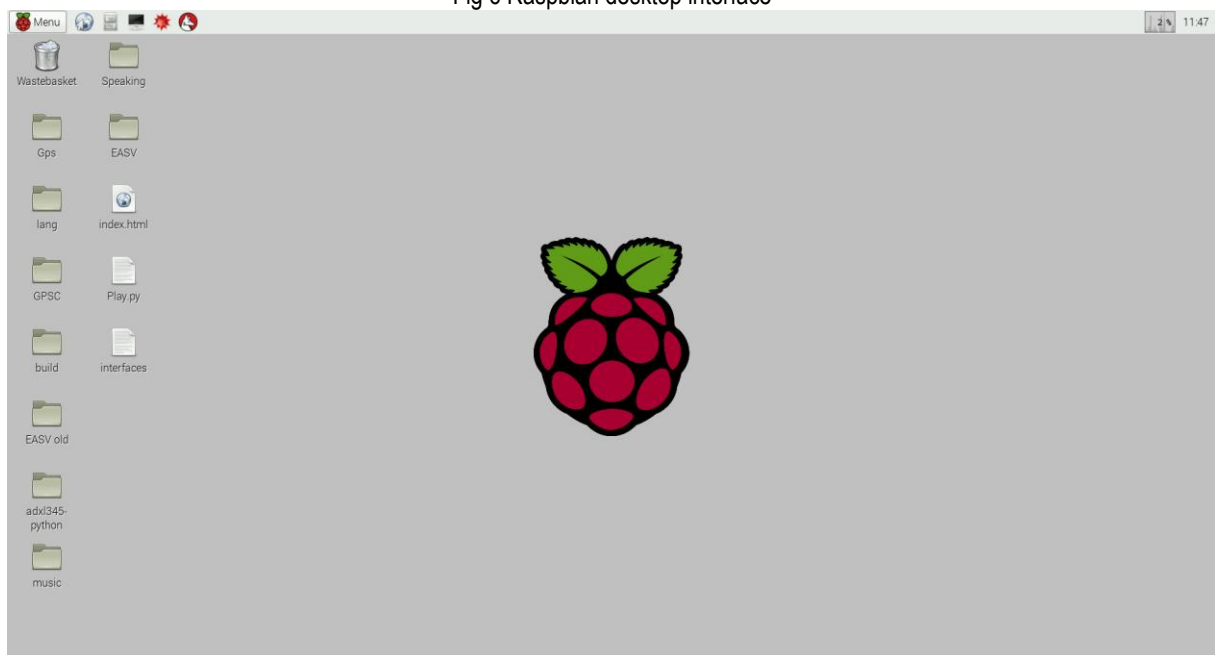
## 6. TEST CASES

The following are the test cases for the user interface and some expected output tallying:

### 6.1. Graphical User Interface

As system is automatic there is hardly any interaction from user to have dedicated user interface. Although if users opt to have touch screen display then they can use Raspbian Desktop interface.

Fig 6 Raspbian desktop interface



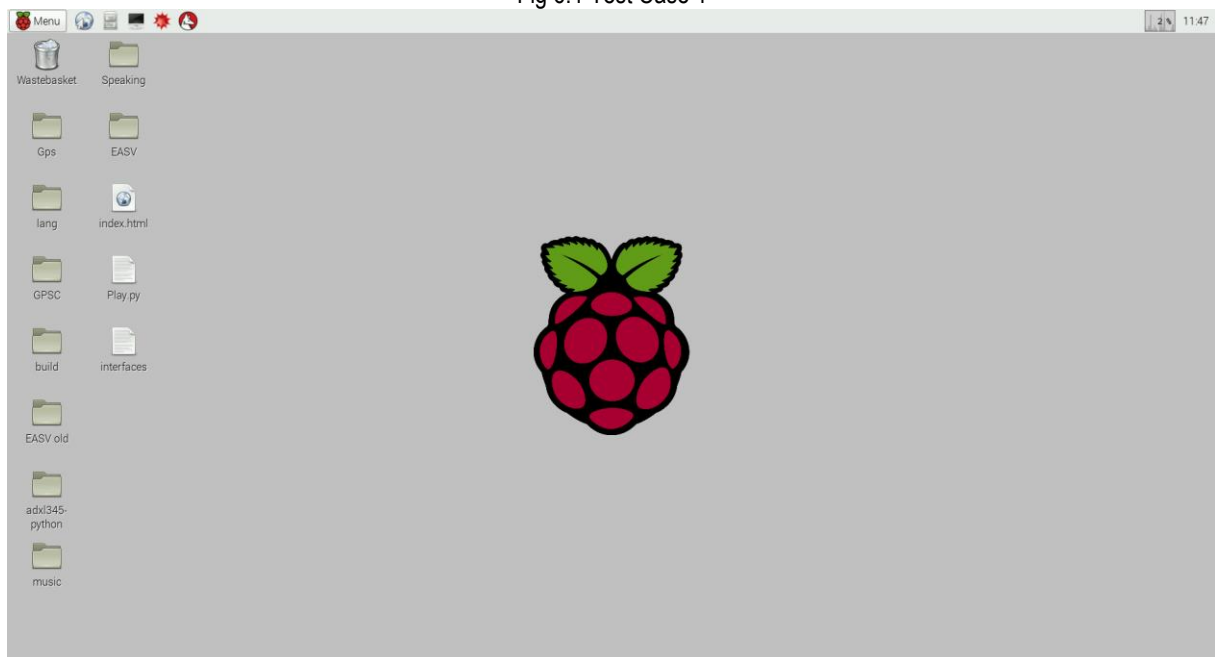
## 6.2. Test Cases

### Test Case #1

Purpose: To ensure that Raspbian OS booted properly.

| Input                        | Expected Output            | Actual Output              |
|------------------------------|----------------------------|----------------------------|
| Raspbian OS image on SD card | Raspbian desktop interface | Raspbian desktop interface |

Fig 6.1 Test Case 1



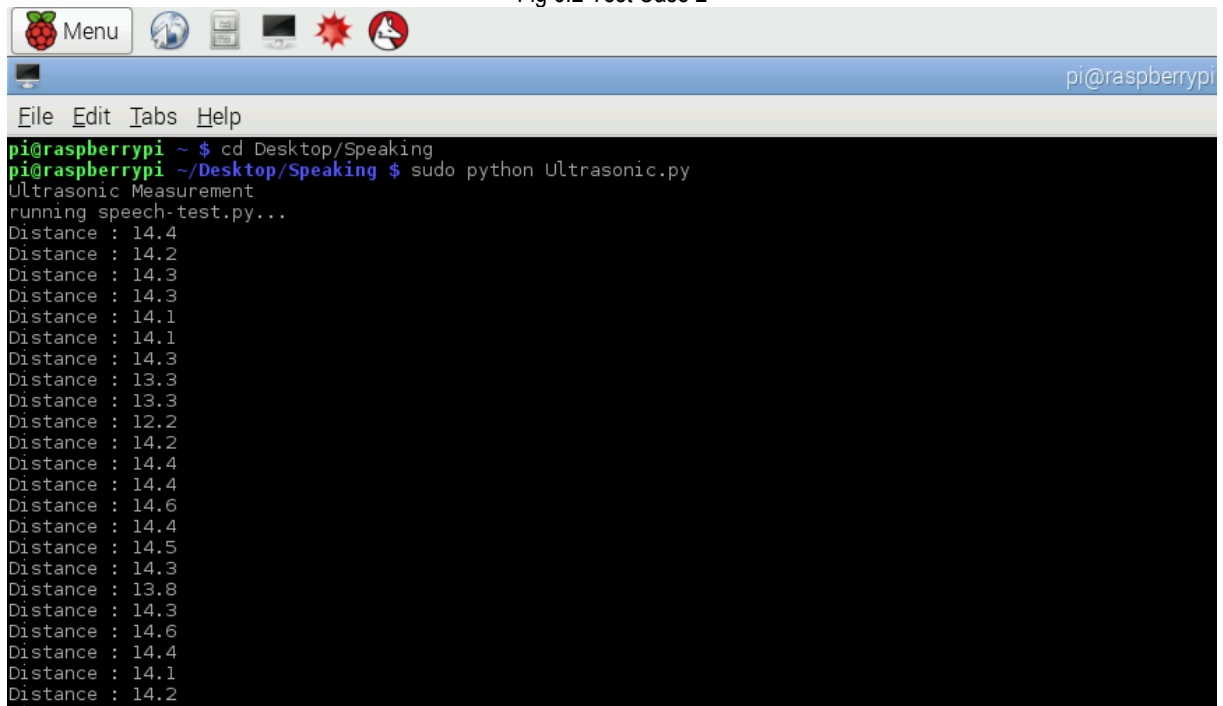


## Test Case #2

Purpose: To ensure Ultrasonic sensor shows distance properly.

| Input                                  | Expected Output                         | Actual Output                           |
|--|---|---|
| Obstacle in front of Ultrasonic sensor | Proper distance from sensor to obstacle | Proper distance from sensor to obstacle |

Fig 6.2 Test Case 2



The screenshot shows a terminal window on a Raspberry Pi. The window title is "pi@raspberrypi". The terminal output is as follows:

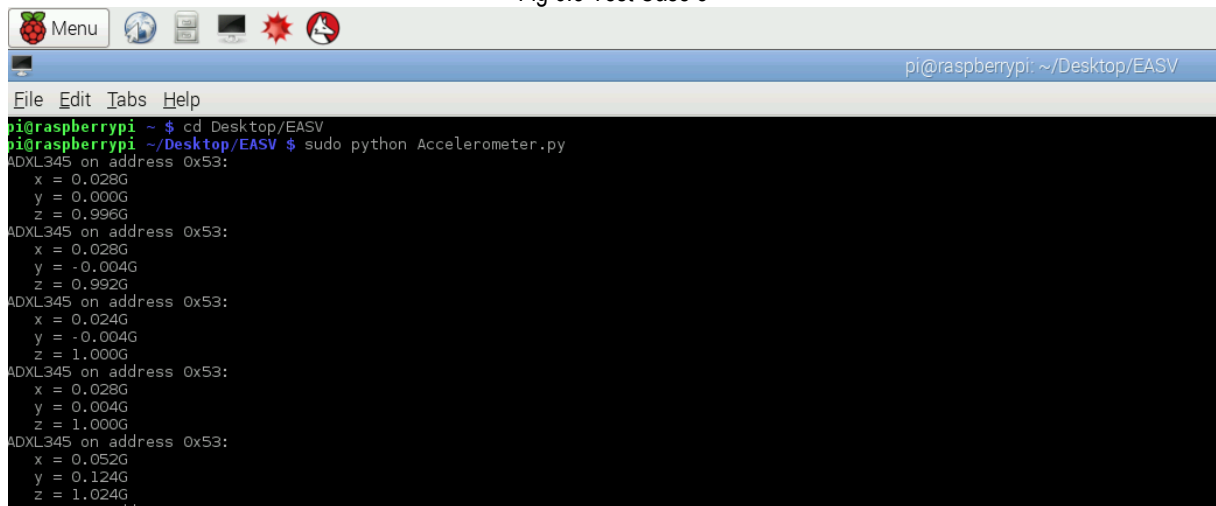
```
pi@raspberrypi ~ $ cd Desktop/Speaking
pi@raspberrypi ~/Desktop/Speaking $ sudo python Ultrasonic.py
Ultrasonic Measurement
running speech-test.py...
Distance : 14.4
Distance : 14.2
Distance : 14.3
Distance : 14.3
Distance : 14.1
Distance : 14.1
Distance : 14.3
Distance : 13.3
Distance : 13.3
Distance : 12.2
Distance : 14.2
Distance : 14.4
Distance : 14.4
Distance : 14.6
Distance : 14.4
Distance : 14.5
Distance : 14.3
Distance : 13.8
Distance : 14.3
Distance : 14.6
Distance : 14.4
Distance : 14.1
Distance : 14.2
```

### Test Case #3

Purpose: To ensure that Accelerometer readings are obtained properly.

| Input   | Expected Output                      | Actual Output                        |
|---|--------------------------------------|--------------------------------------|
| Accelerometer sensor input from Raspberry PI GPIO pins. | X,Y,Z axis readings of accelerometer | X,Y,Z axis readings of accelerometer |

Fig 6.3 Test Case 3



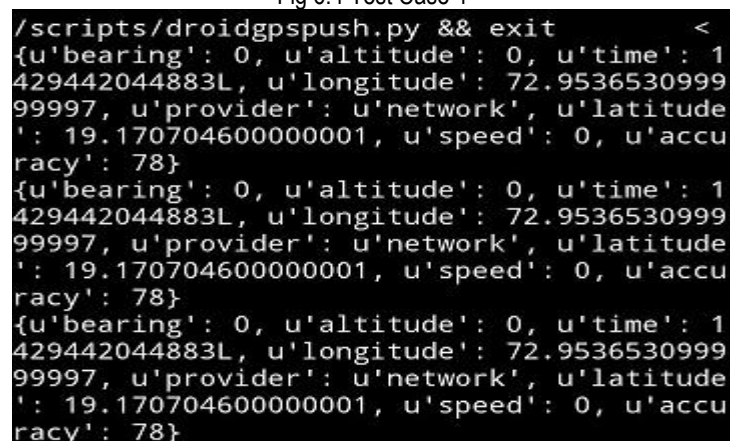
```
pi@raspberrypi ~ $ cd Desktop/EASV
pi@raspberrypi ~/Desktop/EASV $ sudo python Accelerometer.py
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.000G
  z = 0.996G
ADXL345 on address 0x53:
  x = 0.028G
  y = -0.004G
  z = 0.992G
ADXL345 on address 0x53:
  x = 0.024G
  y = -0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.052G
  y = 0.124G
  z = 1.024G
```

### Test Case #4

Purpose: To ensure that accurate GPS coordinates are received.

| Input                          | Expected Output                      | Actual Output                        |
|--------------------------------|--------------------------------------|--------------------------------------|
| GPS coordinate from GPS sensor | Accurate GPS coordinates of location | Accurate GPS coordinates of location |

Fig 6.4 Test Case 4



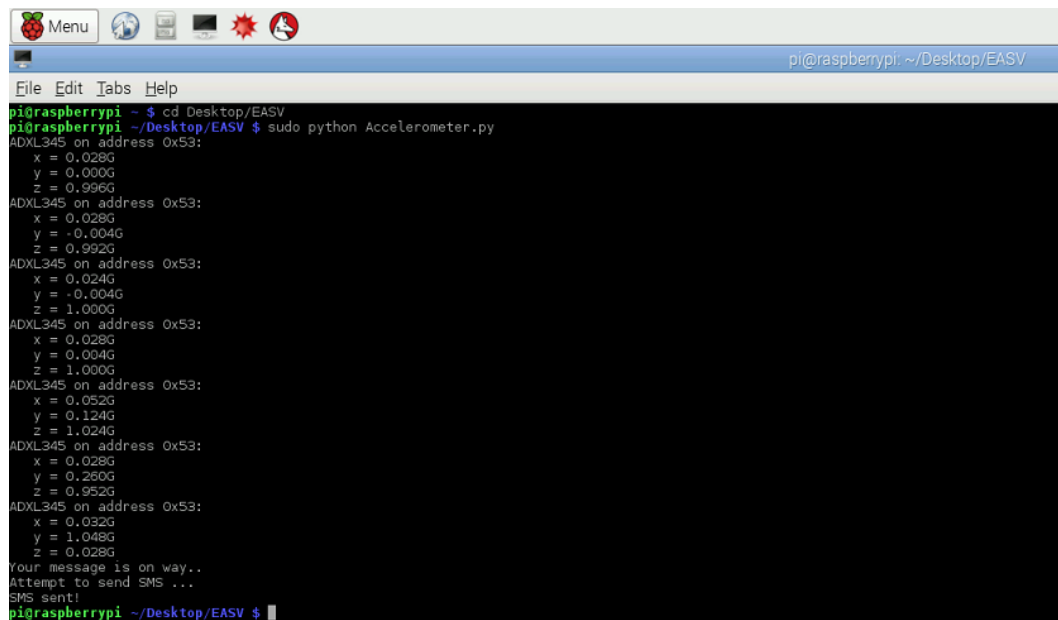
```
/scripts/droidgpspush.py && exit <
{u'bearing': 0, u'altitude': 0, u'time': 1429442044883L, u'longitude': 72.953653099999997, u'provider': u'network', u'latitude': 19.170704600000001, u'speed': 0, u'accuracy': 78}
{u'bearing': 0, u'altitude': 0, u'time': 1429442044883L, u'longitude': 72.953653099999997, u'provider': u'network', u'latitude': 19.170704600000001, u'speed': 0, u'accuracy': 78}
{u'bearing': 0, u'altitude': 0, u'time': 1429442044883L, u'longitude': 72.953653099999997, u'provider': u'network', u'latitude': 19.170704600000001, u'speed': 0, u'accuracy': 78}
```

## Test Case #5

Purpose: To ensure that SMS with proper GPS coordinates of accident location is sent on exceeding accident threshold limit.

| Input                                  | Expected Output   | Actual Output   |
|--|---|---|
| Accelerometer and GPS sensor readings. | SMS with accurate GPS coordinates send on relative's phone. | SMS with accurate GPS coordinates send on relative's phone. |

Fig 6.5 Test Case 5(1)

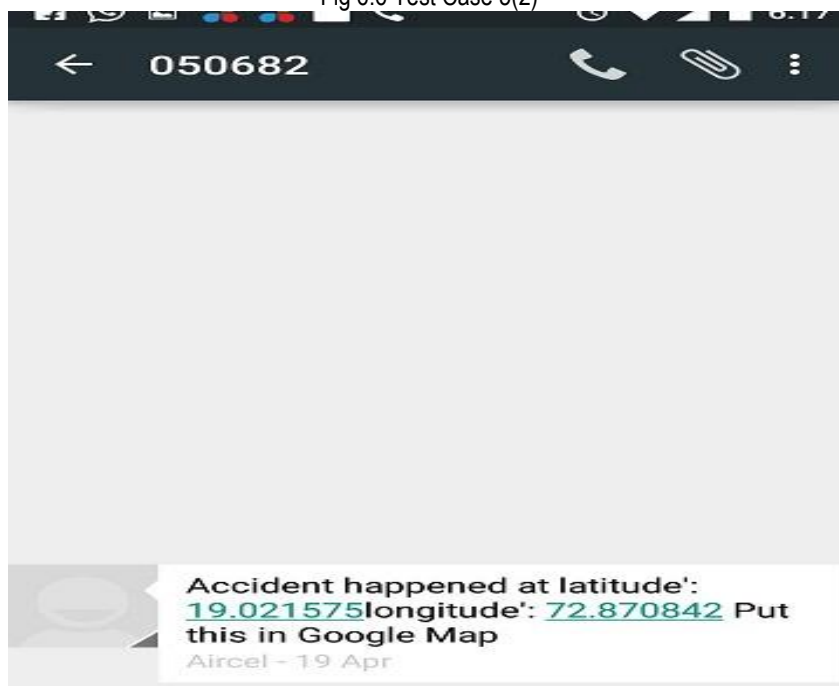


```

pi@raspberrypi ~ $ cd Desktop/EASV
pi@raspberrypi ~/Desktop/EASV $ sudo python Accelerometer.py
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.000G
  z = 0.996G
ADXL345 on address 0x53:
  x = 0.028G
  y = -0.004G
  z = 0.992G
ADXL345 on address 0x53:
  x = 0.024G
  y = -0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.052G
  y = 0.124G
  z = 1.024G
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.260G
  z = 0.952G
ADXL345 on address 0x53:
  x = 0.032G
  y = 1.048G
  z = 0.028G
Your message is on way..
Attempt to send SMS ...
SMS sent!
pi@raspberrypi ~/Desktop/EASV $

```

Fig 6.6 Test Case 5(2)



## 7. CONCLUSION AND FUTURE SCOPE

EASV is used for quick assistance to the driver in the case of accidents. Despite awareness campaign, accidents are still increasing due to driver's poor behaviours such as speed driving, drunk driving, riding with no helmet protection, riding without sufficient sleep etc. The numbers of death and disability are very high because of late assistance to people who were involved in the accident. Hence EASV is used to provide assistance for people who get into accidents.

EASV can also be used to avoid accidents. The ultrasonic distance measuring sensor will let driver to know about the nearby vehicles, so that driver will get alert and so can avoid the accident same also can also be applied for parking assistance. EASV uses Raspberry PI which can be further used to run large number of applications including Gmail, Facebook, online radio, You-tube, etc.

It can play almost any media formats, including playing from archives and iso files, it allows passengers in the car to control the media centre (play media, create playlist, and browse files).

Further features like theft detection can also be added using GPS and GSM modules.

## APPENDIX A: USER MANUEL

### **For indoor testing:**

- 1) Connect system to power supply.  
Power supply details: 5V and 800mA to 1000mA (micro USB charger)
- 2) Connect Raspberry Pi to display device using HDMI cable.
- 3) Connect android phone and Raspberry Pi in same network. Use Wi-Fi dongle for Raspberry Pi.
- 4) Run server\_socket.py in lxterminal of Raspberry Pi.
- 5) Run droidgppush.py on android phone in Qpython app.
- 6) Run sendsms.py in lxterminal of Raspberry Pi.
- 7) Now tilt accelerometer till it reaches threshold.
- 8) SMS will be sent on cell phone of given contact number.
- 9) For Ultrasonic sensor testing put obstacle in front of Ultrasonic sensor.
- 10) Run speaking.py in lxterminal. System will speak out distance when obstacle distance is less than 10cm.

### **For outdoor use:**

- 1) Connect system to power supply.  
Power supply details: 5V and 800mA to 1000mA (micro USB charger)
- 2) Mount the system in a car. Mount ultrasonic sensor in the front and back side of car.
- 3) System will start automatically when car is started by user.

## APPENDIX B: CLASSES AND EXTERNAL LIBRARIES

### 1. I2C Python Library - 3-Axis Digital Accelerometer (ADXL345)

The ADXL345 is a 3 axis accelerometer that can communicate to the Raspberry Pi with I2C. It offers several advanced features that can be easily incorporated into your project via this library. In general an accelerometer returns the gravitational effects on 3 sensors; for example, if the z axis is perpendicular to the earth, it should have a reading of approximately 1.

Some features of this library:

- Instance initializes with default values for common uses simplifying initial setup
- Settings can be tweaked for individual applications
- Easy setup for single tap, double tap, or free fall sensing
- Configure interrupts to trigger based on activity, taps or free fall

### 2. Rpi.GPIO

Using Python on the Raspberry Pi opens up the opportunity to connect to the real world through the Pi's GPIO pins. This can be done with the RPi GPIO library. It is preinstalled on recent Raspbian images, but if you have an older one you can install it with:

```
sudo apt-get install python-rpi.gpio  
OR  
sudo apt-get install python3-rpi.gpio
```

### 3. Urllib and urllib2

This module provides a high-level interface for fetching data across the World Wide Web. In particular, the `urlopen()` function is similar to the built-in function `open()`, but accepts Universal Resource Locators (URLs) instead of filenames. Some restrictions apply — it can only open URLs for reading, and no seek operations are available.

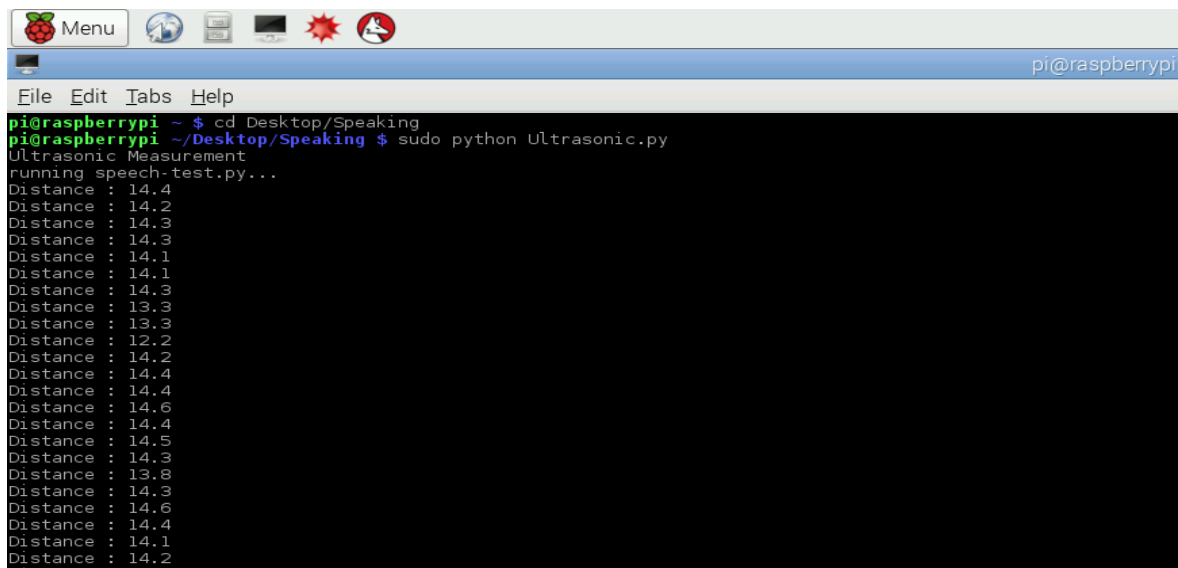
## APPENDIX C: INPUT AND OUTPUT FOR TEST CASES

### Test Case #1

Purpose: To ensure Ultrasonic sensor shows distance properly.

| Input                                  | Expected Output                         | Actual Output                           |
|--|---|---|
| Obstacle in front of Ultrasonic sensor | Proper distance from sensor to obstacle | Proper distance from sensor to obstacle |

Fig C1 Test Case 1



The screenshot shows a terminal window on a Raspberry Pi. The window title is "pi@raspberrypi". The terminal output is as follows:

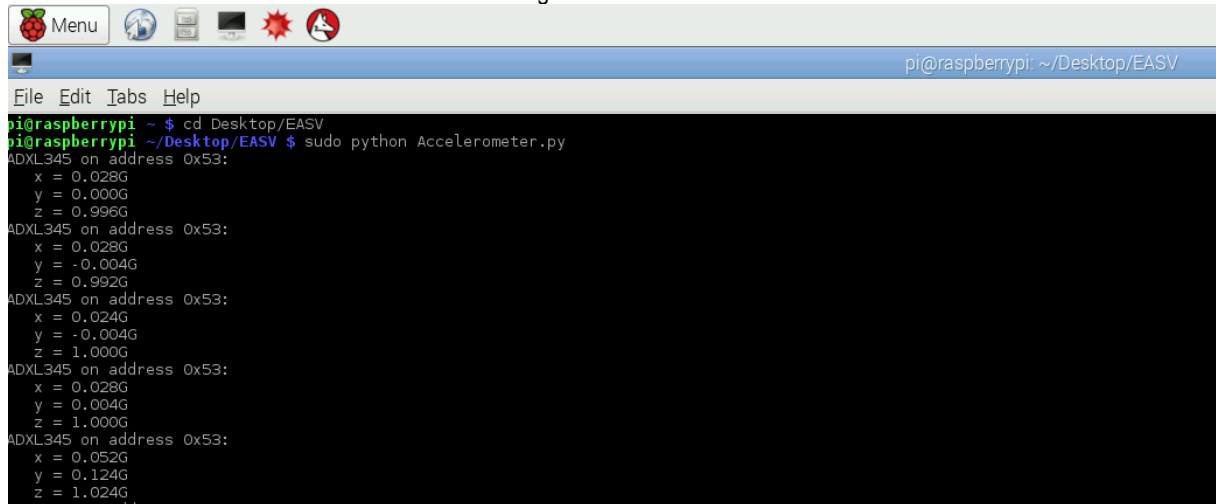
```
File Edit Tabs Help
pi@raspberrypi ~ $ cd Desktop/Speaking
pi@raspberrypi ~/Desktop/Speaking $ sudo python Ultrasonic.py
Ultrasonic Measurement
running speech-test.py...
Distance : 14.4
Distance : 14.2
Distance : 14.3
Distance : 14.3
Distance : 14.1
Distance : 14.1
Distance : 14.3
Distance : 13.3
Distance : 13.3
Distance : 12.2
Distance : 14.2
Distance : 14.4
Distance : 14.4
Distance : 14.6
Distance : 14.4
Distance : 14.5
Distance : 14.3
Distance : 13.8
Distance : 14.3
Distance : 14.6
Distance : 14.4
Distance : 14.1
Distance : 14.2
```

## Test Case #2

Purpose: To ensure that Accelerometer readings are obtained properly.

| Input   | Expected Output                      | Actual Output                        |
|---|--------------------------------------|--------------------------------------|
| Accelerometer sensor input from Raspberry PI GPIO pins. | X,Y,Z axis readings of accelerometer | X,Y,Z axis readings of accelerometer |

Fig C2 Test Case 2



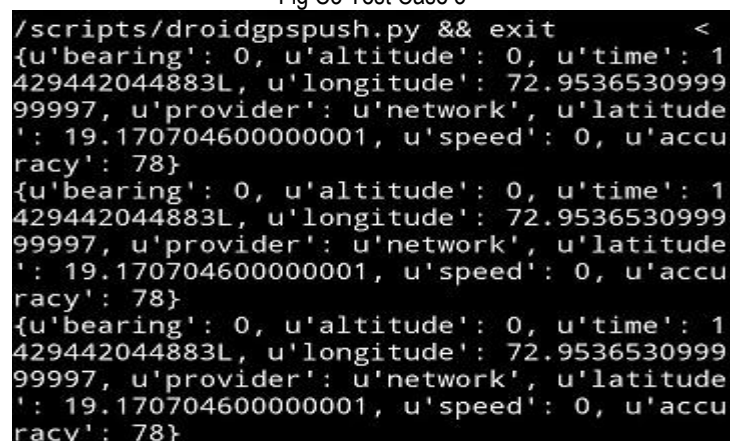
```
pi@raspberrypi ~ $ cd Desktop/EASV
pi@raspberrypi ~/Desktop/EASV $ sudo python Accelerometer.py
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.000G
  z = 0.996G
ADXL345 on address 0x53:
  x = 0.028G
  y = -0.004G
  z = 0.992G
ADXL345 on address 0x53:
  x = 0.024G
  y = -0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.052G
  y = 0.124G
  z = 1.024G
```

## Test Case #3

Purpose: To ensure that accurate GPS coordinates are received.

| Input                          | Expected Output                      | Actual Output                        |
|--------------------------------|--------------------------------------|--------------------------------------|
| GPS coordinate from GPS sensor | Accurate GPS coordinates of location | Accurate GPS coordinates of location |

Fig C3 Test Case 3



```
/scripts/droidgpspush.py && exit <
{'bearing': 0, 'altitude': 0, 'time': 1429442044883L, 'longitude': 72.953653099999997, 'provider': 'network', 'latitude': 19.170704600000001, 'speed': 0, 'accuracy': 78}
{'bearing': 0, 'altitude': 0, 'time': 1429442044883L, 'longitude': 72.953653099999997, 'provider': 'network', 'latitude': 19.170704600000001, 'speed': 0, 'accuracy': 78}
{'bearing': 0, 'altitude': 0, 'time': 1429442044883L, 'longitude': 72.953653099999997, 'provider': 'network', 'latitude': 19.170704600000001, 'speed': 0, 'accuracy': 78}
```

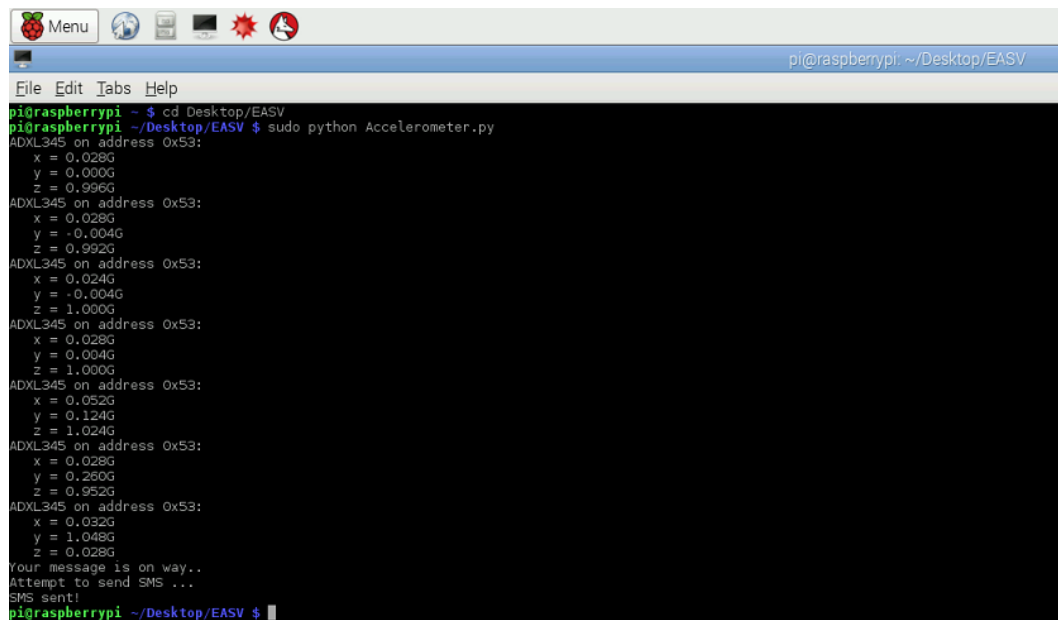


#### Test Case #4

Purpose: To ensure that SMS with proper GPS coordinates of accident location is sent on exceeding accident threshold limit.

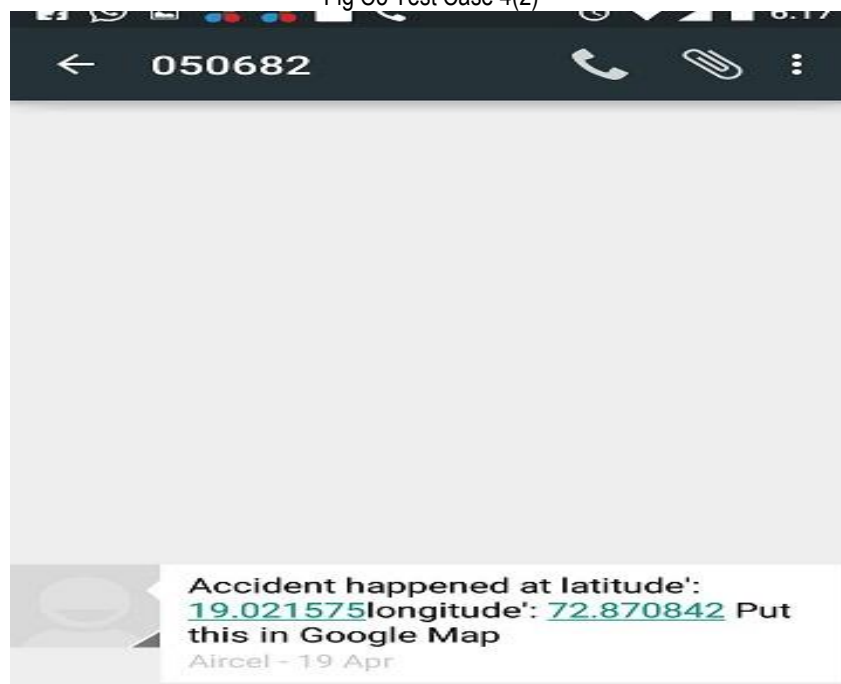
| Input  | Expected Output   | Actual Output   |
|--|---|---|
| Accelerometer tilted till threshold and GPS sensor readings. | SMS with accurate GPS coordinates send on relative's phone. | SMS with accurate GPS coordinates send on relative's phone. |

Fig C4 Test Case 4(1)



```
pi@raspberrypi ~ $ cd Desktop/EASV
pi@raspberrypi ~/Desktop/EASV $ sudo python Accelerometer.py
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.000G
  z = 0.996G
ADXL345 on address 0x53:
  x = 0.028G
  y = -0.004G
  z = 0.992G
ADXL345 on address 0x53:
  x = 0.024G
  y = -0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.004G
  z = 1.000G
ADXL345 on address 0x53:
  x = 0.052G
  y = 0.124G
  z = 1.024G
ADXL345 on address 0x53:
  x = 0.028G
  y = 0.260G
  z = 0.952G
ADXL345 on address 0x53:
  x = 0.032G
  y = 1.048G
  z = 0.028G
Your message is on way..
Attempt to send SMS ...
SMS sent!
pi@raspberrypi ~/Desktop/EASV $
```

Fig C5 Test Case 4(2)



## BIBLIOGRAPHY

- [1] Mattias Norrel "Sending sms from Raspberry pi" <http://www.mattiasnorell.com/blog/send-sms-from-a-raspberry-pi/>
- [2] Pallakila Sai Avinash,K.ThenKumari "Wireless Black Box Using MEMS Accelerometer and GPS Tracking for Accidental Monitoring of vehicles"
- [3] Bhaumik Bhat, "PiDroid GPS tracker"  
<http://www.instructables.com/id/PiDroidGPSTracker/?ALLSTEPS>
- [4] "Embedded Linux Wiki" [www.elinux.org](http://www.elinux.org)
- [5] "Setting Up your ADXL345 Accelerometer & Python for Raspberry Pi"  
<https://blog.adafruit.com/2014/07/04/adxl345-accelerometer-python-for-raspberry-pi/>
- [6] "Raspberry pi official forums" <http://www.raspberrypi.org/forums/>
- [7] Rui Santos "Complete Guide for Ultrasonic Sensor HC-SR04"  
<http://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/>