Smart Helmet for
Accident Detection
&
Reporting
“SHADR”

By

Muneeb Naveed

Talha Usman

Saad Bin Khalid

M. Suleman Mazhar

Submitted to the Faculty of Department of Electrical(telecommunications) Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad

in partial fulfillment for the requirements of B.E Degree in

Electrical(telecommunications) Engineering

JUNE 2019

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

"Smart Helmet for Accident Detection and Reporting"

is carried out by

Muneeb Naveed, Talha Usman, Saad Bin Khalid, M. Suleman Mazhar

under my supervision and that in my judgement, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical Engineering from National University of Sciences and Technology (NUST), Islamabad.

Approved By:

Signature: ________________________________

Supervisor: **Asst. Prof. Mir Yasir**

MCS, Rawalpindi

ABSTRACT

Our project is based on creating an accident detection and reporting platform. An alarming number of deaths are caused by delayed arrival of emergency services at the location of accident. The project was aimed with designing a solution that can mitigate the delay in response time from user end and introduce a smart solution in that regard.

A prototype was designed for motorcyclists with an accelerometer and gyroscope sensor; to get raw values regarding orientation and sudden changes in speed. These values were fed into a Raspberry Pi 0W, microcontroller that determines whether an accident has occurred or not. In case of an accident, microcontroller communicates with android application on the phone via Bluetooth where user information is stored. The smartphone is used for intimating emergency services and contacts. Machine Learning has been implemented to reduce false positives.

This project aims at designing a scalable and economical accident detection platform with the ability to add additional features as per user requirement.

DECLARATION OF ORIGINALITY

We hereby declare that no content of work presented in this report has been submitted in support of another award of qualification or degree either in this institution or anywhere else.

“It is better to be unhappy and know the worst, than to be happy in a fool’s paradise”

-

Fyodor Dostoyevsky

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Taala is the sole guidance in all domains.

Our parents, colleagues and most of all supervisor, Dr. Mir Yasir, this could not have been possible without your input.

The group members, despite all adversities worked steadfastly.

Table of Contents

Section - I

Section - II

Section - III

List of Figures

List of Abbreviations

SPDT	–	Sing Pole, Double Throw
GSM	–	Global System for Mobile Communication
GPS	–	Global Positioning System
MEMS	–	Microelectromechanical System
ADC	–	Analog to Digital Converter
DMP	–	Digital Motion Processor
GPIO	–	General Purpose Input Output
BLE	–	Bluetooth Low Energy
UI	–	User Interface
ML	–	Machine Learning
PBS	–	Pakistan Bureau of Statistics
ERT	–	Emergency Response Teams
QRF	–	Quick Reaction Force
WHO	–	World Health Organization
GHO	–	Global Health Observatory

CHAPTER - 1

Introduction

The team has designed a smart helmet for motor cyclists which has two main features:

- Accident Detection
- Accident Reporting (To emergency contacts with the live accident location)

This project can be divided into three main parts. The first unit consists of the sensor circuitry. We have used a combination of accelerometer (for measuring change in velocity) and a gyroscope (for measuring change in orientation) covering 6-axis to get readings for accident detection. The second unit contains a Raspberry Pi microcontroller which will be used as the main processing unit, taking raw values from the sensors through a wired medium, scaling them and finally evaluating whether an accident has occurred or not. The third platform is an android application which will be communicating with the Raspberry Pi via Bluetooth. In case of an accident, the microcontroller will intimate the application and an SMS will be generated and sent to the Emergency contacts. Smart Helmet application provides additional features like Navigation and live Speed Check. In addition, several features can be integrated inside the helmet to assist the motor cyclist like voice and sound assistance, brake and indicator lights, rear view camera etc. Keeping in mind, that the price of the helmet will increase relatively if we add more feature, we developed the prototype with just the basic accident detection and reporting feature.

1.1 Problem Statement

A 2014 World Health Organization report indicates that 20 people out of every 100,000 die in a road accidents per year [1-4]. More than 50% accidents involve motor bikes and 75% of the registered vehicles are motor cycles according to Pakistan Bureau of Statistics [3]. Therefore, inadequate safety and untimely first aid of those involved in accidents is evident from research.

As most injuries in motorcycle accidents result in neural and head injuries as well as blood loss, therefore it is of primary importance that emergency response teams arrive at the incident as soon as possible. These injuries can increase chances of life-threatening consequences for those involved with each passing moment. Survival rate may increase by prompt medical attention at the scene of accident.

1.2 Solution

The goal of the project is to introduce a smart helmet with sensors for accident detection and a solid reporting platform. This helmet will aid the rider as well as emergency response teams and his emergency contacts in case of an accident by notifying about the location of incident as well as the riders health information through SMS. Unavailability of a similar product in Pakistani market is also a motivation as motor cycles are the primary mode of transportation for a large percentage of the population in Pakistan. There are many types of disabilities. Advanced technology has been used in helping people overcome such impairments. However, developing such an advanced technology is expensive, making their selling price higher as we add more features.

1.3 Scope and limitations

This project uses modern day tech to design a helmet with the following scope:

- Using Raspberry-Pi and sensors (IC-MPU 5060) to improve a traditional motor cycle helmet.
- The Smart Helmet is more than just safety equipment with introduction of smart features.
- Remote monitoring of the motor cyclist's whereabouts and other details like live speed.
- User friendly android application.
- Wireless (via Bluetooth) integration of android application and Raspberry Pi improves interfacing.

1.5 Objectives

We have the following objectives for the design of our project:

- Our focus is on developing an accident detection and reporting platform.
- Additional features that can be added in android application are:
    - Navigation.
    - Speed check.
    - Audio functionality.
    - Application where client data can be stored and analyzed safely.

Also, this project aims to provide a '*cost-effective*' smart helmet for motorcycle riders keeping in mind the financial constraints of a common biker owner in Pakistan. This will not only introduce a smart helmet accessible to the public but also equipped with an array of sensors to support the emergency services. It would introduce smart features like Bluetooth audio connectivity for navigation purposes as well as a mobile application for a customized user experience.

1.6 Organization

This Thesis is divided into three main sections.

The first section contains the Introduction and Literature Review, chapters 1 and 2. The chapter on introduction includes the Problem Statement and the proposed Solution. Literary review will give you a brief idea on the research we conducted and the hardware equipment that was used for developing the project.

The second section contains the 3rd and 4th chapters. The third chapter focuses on the communication between the sensors and Raspberry Pi and "Accident Detection Mechanisms". The fourth chapter encompasses the "Reporting Mechanism" using an Android Application.

The third and the last section includes our conclusions and future works with the limitations faced, composed of the 5th chapter.

After that are references and works cited throughout the thesis and appendices.

CHAPTER - 2

Literature Review

This section provides an overview of the research carried out by various scholars, companies and engineering students which helped us in getting a sense of direction as well as aided us in formulating a strategy to achieve our desired goals.

2.1 Scholarly Articles

Before choosing the equipment to be used, we consulted several IEEE Articles related to our project. These research papers gave us rough idea of the design approach that was most viable in terms of cost and efficiency. The base design model we opted was based on the following diagram:

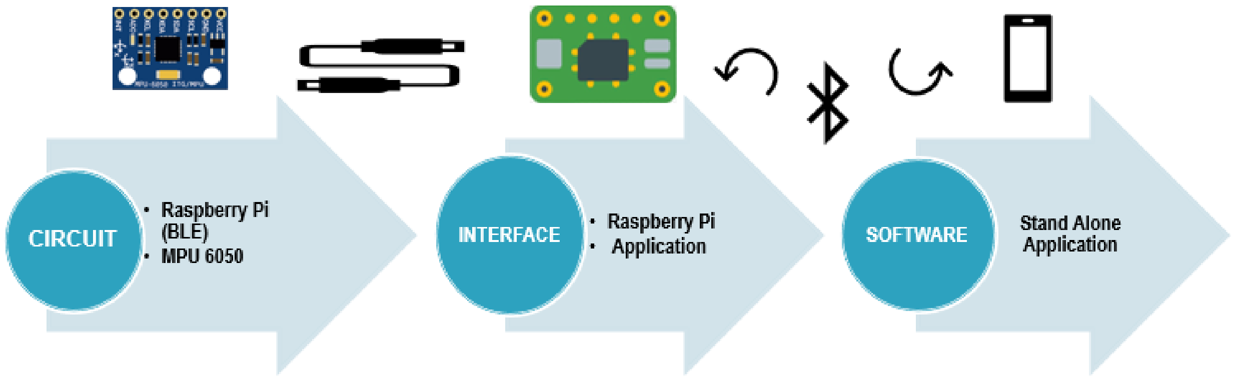


Figure 1 project architecture

In the light of this model, information from IEEE research papers will be explained.

1. Circuit

Accident detection is the fundamental part of our project. Choosing the appropriate sensor to detect the accident and microcontroller that can compute the raw values efficiently was considered in this section. There have been various methods used by researchers from India and Europe like:

- **Brushless Direct Current (BLDC) Fan** – It is a speed sensor that constantly measures speed against wind resistance [5]. This seemed somewhat irrelevant to our design parameters of keeping the project as minimal as possible because we introduced speed check with the help of android application.

Figure 2 BLDC fan [5]

- **Force Sensor Resistor (FSR) –** A flexible resistor that changes resistance directly proportional to the applied force [5]. This sensor is used to assure whether the motorcyclist is wearing the helmet or not. The issue we had in this case was to install FSR in an appropriate position inside the helmet without compromising integrity of helmet and calibrating it properly.

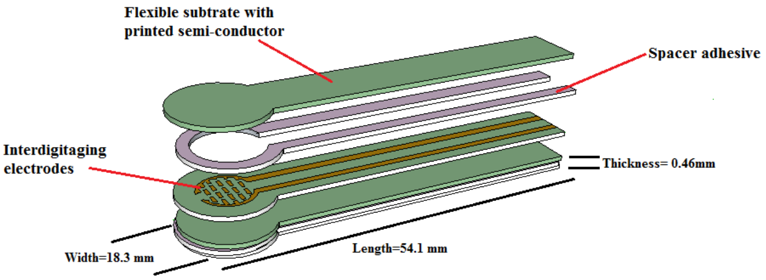

Flexible subtrate with printed semi-conductor
Spacer adhesive
Interdigitaging electrodes
Thickness= 0.46mm
Width=18.3 mm
Length=54.1 mm

Figure 3 force sensor [5]

- **Timer & Comparator IC –** These were used with physical sensors for accident detection and aided in serially interpreting values from them [5]. This triggered the actual accident detection flag. We deduced that we could remove them with the use of a smart microcontroller like Raspberry Pi or Arduino.

- **SPDT Switch –** It is a normal 3-way switch that has been used to assure the rider clips on the helmet or not [5]. This was important to implement as clipping in the helmet i.e. important but since market research revealed that such a switch with specific dimensions to be attached with motorbike helmet were not available. So, software checks were introduced.

- **Peripheral Interface Controller (PIC) 16F84A –** It is a Reduced Instruction Set Computer (RISC) based microcontroller [5] that is used for interfacing between detection sensors and reporting transmitters. This requires coding in assembly that is very specific and cannot be connected with multiple sensors. Therefore, we decided to choose a smart microcontroller with inbuilt wireless transmission capability.

- **Impact Sensor –** It is used in the automobile industry to detect impacts [6] like in airbags, but market research revealed that such sensors are not available in Pakistani market and importing variants of it from China was not feasible as they were vendor specific, not running on open source protocols and could be ordered only in bulk.

- **Accelerometer ADXL355 –** It is a 3-axis sensor [6]. We decided that using this sensor for accident detection would be most feasible as we can use raw values obtained from test drives and use trial and error method to develop an accurate accident detection model. We opted to use MPU6050 in the end, as software support for it was readily available for Raspberry Pi and it had an inbuilt gyroscope sensor as well, 6-axis sensor in total with which we had more room for improvement.

2. Interface

In this section we thoroughly researched about the approaches to communicate between sensors, microcontroller and android application.

- **RF Transmitter/Receiver –** This has been introduced for communication between the microcontroller and accident reporting mechanism [5]. This approach seemed redundant as already microcontrollers like Raspberry Pi are available with inbuilt protocols for Bluetooth communication. Therefore, the need for dedicated component only for this purpose was removed using, one in all microcontroller.

- **GSM-GPS Module –** This sensor was used by researchers for accurate location coordinates in case of an accident [6] and then intimation to emergency services as well as contacts. Introducing it into our project was not feasible as we could utilize an android application on smartphone for both of the earlier mentioned reasons and keep the cost constraints as low as possible removing redundancy.
- **Bluetooth Module –** This could be used in case of an independent microcontroller with no means of communicating with the android application [6], so we can keep the costs as low as possible. However, we opted to go with a microcontroller with inbuilt Bluetooth and WIFI capability because programming an analog microcontroller and coding Bluetooth protocol at our level was too complex.

3. Software

This section covers the aspects of how to approach the intimation of an accident, after it has been detected.

- **Android Application –** This is the appropriate software platform we used. An android application with all the user information and that intimates as well in case of an accident [6], removing the need for a dedicated GPS/GSM module. Additionally, we introduced speed checks via this app. An application utilized for the advantage of the masses already using smartphones and designed as simple as possible for most of the users.

- **Cloud Based Platform -** It was based on communicating directly with a cloud server that will process live data from sensors and intimate on your behalf [7-9]. While using this approach, we discovered that it was expensive and required funding as well as training to develop such a platform. However, this option could be explored more in the future for making a product for corporate/government clientele.

2.2 Market Research

A survey of the market revealed that some companies have been working rigorously on developing smart helmets to make a rider's experience both safe and enjoyable. Once the stuff of science fiction, smart helmets are now a very real product bringing built in accident detection capabilities, speakers and microphone allowing the rider to make and receive calls, audio multitasking, rear facing cameras, GPS directions and a heads-up display. Skully Technologies (USA) and Sena (South Korea) being the most prominent of all, have developed several helmets with

Figure 4 Skully Helmet [10]

accident detection capabilities as well as various added features to assist and aid the motorbike driver.

2.3 Components Used

After a month of research which included understanding concepts used by several individuals (students and professionals) in IEEE research papers for developing accidents detection and reporting platforms, online research and consulting with our professors, we decided to use the following equipment for designing our own accident detection and reporting platform:

- Raspberry Pi Zero W microcontroller
- IC MPU – 6050 (6-axis accelerometer and gyroscope)

2.4 Raspberry Pi Zero W

The Pi Zero is a full computer packed onto a single board which is popularly known as a single board computer. The introduction of Pi zero allowed embeding an entire computer onto a tiny integrated circuit. We used the latest model of "*Pi Zero*" product line, the Raspberry Pi Zero - Wireless. It provides features like onboard Wi-Fi module, with built-in Wi-fi and Bluetooth functionality. At the heart of the Pi zero W is a 1GHz BCM-2835 Single-Core Processor, the same as the B plus and A plus, with 512 Mega Byte RAM [11]. No matter how you want to use your Pi zero W, you will need a Micro-SD card with an OS and a high-quality 5V power supply to keep the SBC running.

Figure 5 Raspberry Pi Zero W [11]

The following equipment is necessary to use a Raspberry Pi:

- Micro-SD Card (for OS and memory)
- Monitor
- Keyboard
- Mouse
- USB and HDMI hub

2.4.1 Hardware Overview of Raspberry Pi Zero W

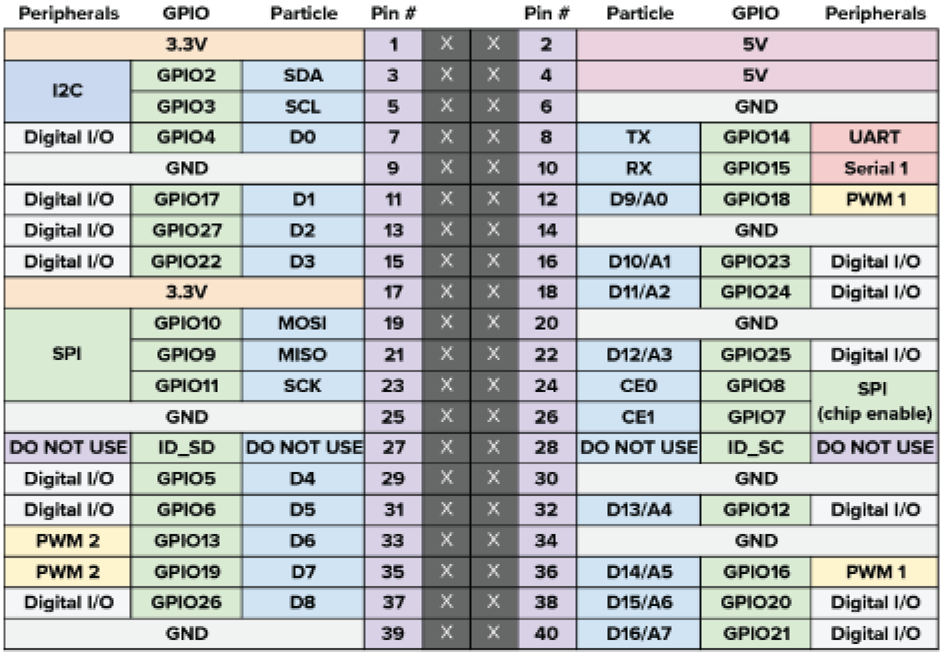
Pi Zero W has the following features:

- Mini – HDMI Port
- USB on-the-go Connection
- Micro-SD Card Slot
- 20 General Purpose Input Output Ports

2.4.2 Operating System of Raspberry Pi

The Raspberry Pi Foundation has developed Raspbian, a “Debian-based” Linux [12] distribution for download, as well as third-party “ubuntu”, Microsoft Windows 10 IOT core, RISC Operating System and specialized Media Centered Distributions. It promotes Python and Scratch as the primary programming languages alongside the support of many other languages. The default firm-ware is “Closed Source”, whereas an unofficial “Open Source” is also available.

2.4.3 Pin Configuration



Peripherals	GPIO	Particle	Pin #			Pin #	Particle	GPIO	Peripherals
3.3V			1	X	X	2	5V		
I2C	GPIO2	SDA	3	X	X	4	5V		
	GPIO3	SCL	5	X	X	6	GND		
Digital I/O	GPIO4	D0	7	X	X	8	TX	GPIO14	UART
GND			9	X	X	10	RX	GPIO15	Serial 1
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0	GPIO18	PWM 1
Digital I/O	GPIO27	D2	13	X	X	14	GND		
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1	GPIO23	Digital I/O
3.3V			17	X	X	18	D11/A2	GPIO24	Digital I/O
SPI	GPIO10	MOSI	19	X	X	20	GND		
	GPIO9	MISO	21	X	X	22	D12/A3	GPIO25	Digital I/O
	GPIO11	SCK	23	X	X	24	CE0	GPIO8	SPI (chip enable)
GND			25	X	X	26	CE1	GPIO7	
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE	ID_SC	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30	GND		
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4	GPIO12	Digital I/O
PWM 2	GPIO13	D6	33	X	X	34	GND		
PWM 2	GPIO19	D7	35	X	X	36	D14/A5	GPIO16	PWM 1
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6	GPIO20	Digital I/O
GND			39	X	X	40	D16/A7	GPIO21	Digital I/O

Figure 6 Pin Configuration Pi Zero W [12]

2.5 IC MPU – 6050 Six-Axis (Accelerometer + Gyroscope)

The MPU – 6050 is the world's first motion tracking devices designed for the low power, low cost and high-performance [13]. It is frequently used in Smartphones, Tablets and wearables.

The MPU-6050 incorporates Motion Fusion and Run-Time calibration firmware that enables manufacturers to eliminate the costly and complex selection and system level integration of devices in Motion-Enabled products delivering optimal performance for consumers.

The MPU-6050 consists of a 3-axis Gyroscope and Accelerometer on the same silicon die, together with an on board DMP, which processes complex 6-axis Motion Fusion Algorithms. The device can access external magneto-meters or other sensors through an Inter Integrated Circuit bus, allowing the devices to gather a full set of sensor data without intervention from the system processor. The devices are offered in a 4 mm x 4 mm x 0.9 mm QFN package.

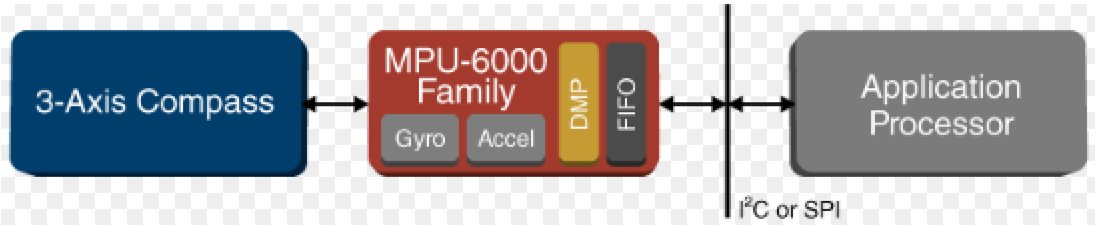


Figure 7 MPU 6050 architecture [13]

2.5.1 MPU – 6050 Features

- Micro Electro Mechanical System combines 3-axis Accelerometer and 3-axis Gyroscope raw values.
- The IC is supplied power at around 3-5 Volts.
- IC can communicate using the Inter Integrated Circuit bus protocol.
- It has a built in 16 bits Analogue to Digital Convector which provides high accuracy.
- A built in DMP helps the IC to provide a High Computational Power.
- Interface with other IIC devices like Magnetometer is also possible.
- The IC provides a configurable IIC – Address.

2.5.2 MPU – 6050 Working

A. The MPU – 6050 consists of a three-axis gyroscope with Micro-Electro-Mechanical System (MEMS) technology [13]. It is used for the detection of rotational velocity along the x, y, z axes as shown in the figure:

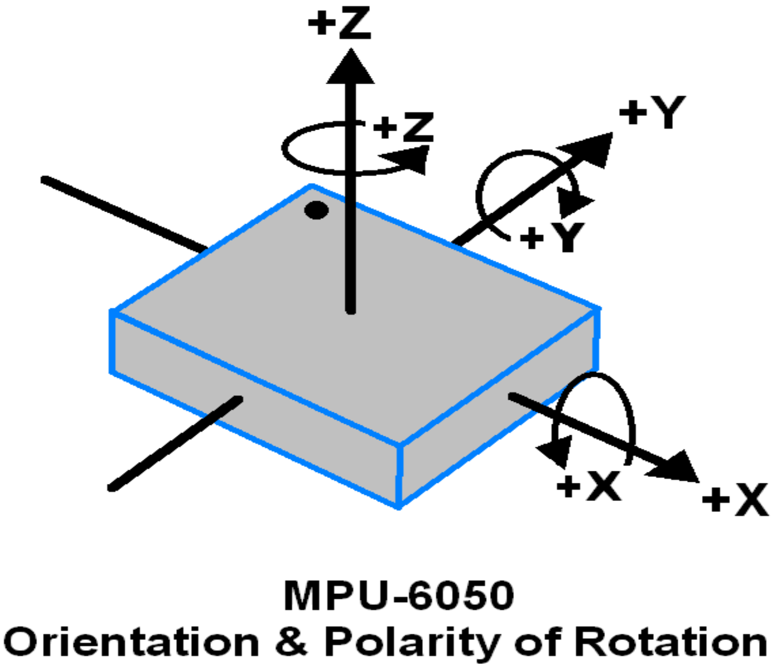


Figure 8 Gyroscope Axis [13]

- Whenever the Gyros are rotated about any of the axes, the "Coriolis Effect" causes a vibration which is detected using Micro Electro Mechanical System inside the MPU-6050.
- This resulting signal is then amplified, demodulated, and filtered to produce a voltage that is proportional to the Angular-Rate.
- This voltage is digitalized using a sixteen bit Analogue to Digital Convertor to sample each axis.
- The full-scale range of output are 250, 500, 1000 and 2000.

- It measures the angular velocity along each axis in degree per second unit.

B. The MPU – 6050 consists of a three-axis accelerometer with Micro-Electro-Mechanical system (MEMs) Technology. It measures the angle of tilt or inclination along the x, y and z axes as shown in the figure:

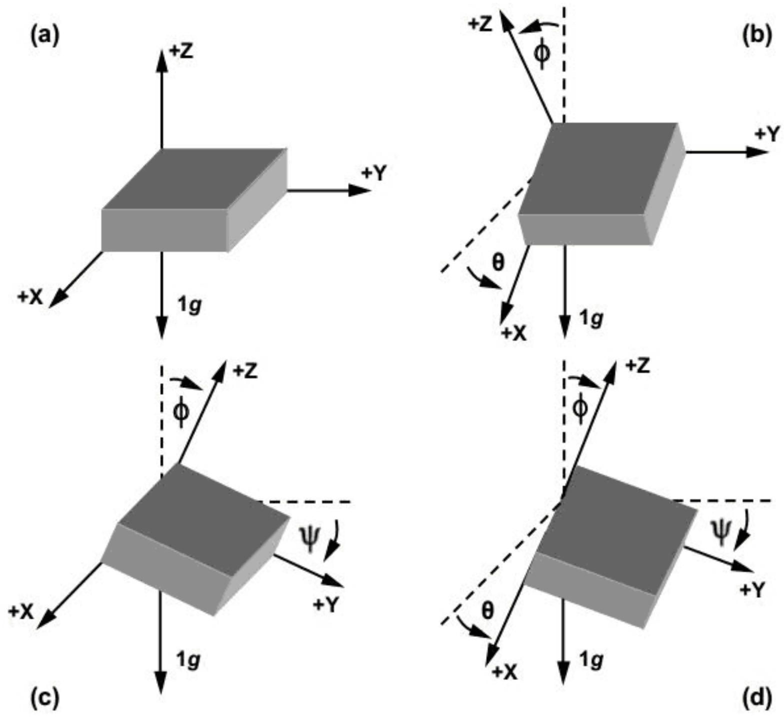


Figure 9 Accelerometer Axis [13]

- Acceleration along the axes deflects the movable mass.
- This displacement of moving plate (mass) unbalances the differential capacitor which results in sensor output. Output amplitude is proportional to acceleration.
- 16-bit ADC is used to get digitized output.
- The full-scale range of acceleration are +/- 2g, +/- 4g, +/- 8g, +/- 16g [13].

- It is measured in *g* (gravity force) unit.
- When device placed on flat surface it will measure 0g on X and Y axis and +1g on Z axis.

C. The embedded Digital-Motion-Processor [13] (DMP) computes Motion Processing Algorithms. It takes data from the Gyroscope, Accelerometer and additional 3rd party sensor such as a magneto-meter and processes the data. It provides motion data like Roll, Pitch, Yaw Angles, Landscape and Portrait sense etc. It minimizes the processes of host in computing Motion Data. The resulting data can be read from DMP registers.

D. An On-Chip Temperature sensor [13] output is given in a digital format using a built-in analogue to digital convertor. This reading can be read from sensor's Data register.

2.5.3 Pin Configuration

The MPU 6050 has 8 external port. Two Prots VCC and GND are used for power supply. Another port ADO is used by the built-in analogue to digital convertor. Rest of the pins along with their configurations are shown in the image below:

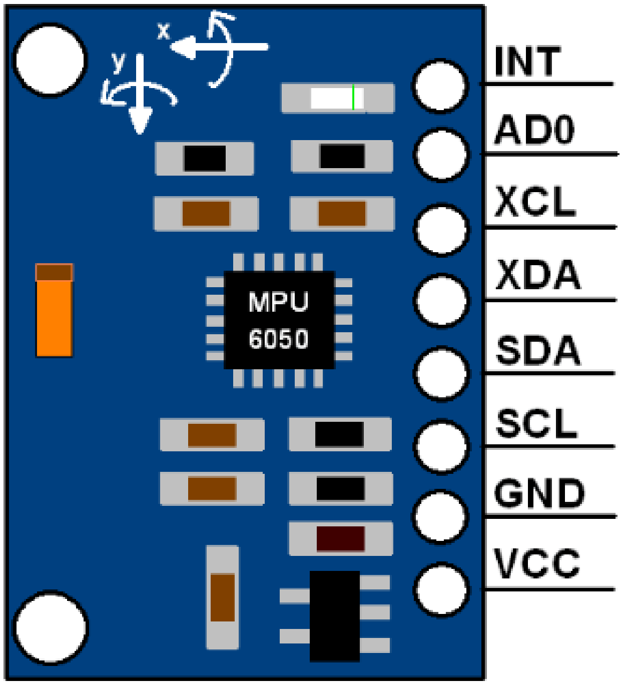


Figure 10 Pin configuration MPU 6050 [13]

CHAPTER - 3

Interfacing & Accident Detection

This section describes how the MPU – 6050 sensors are communicating with the Raspberry Pi and finally when value from the sensors arrive, how an accident is detected.

3.1 Communication Between MPU – 6050 with Raspberry Pi

For communication between the Pi and MPU – 6050 we used the I2C and SPI serial interfaces. Numerous Peripherals can be added to a micro-controller chip over the Inter-Integrated-Circuit bus and Serial Peripheral Interfaces; incorporating various sensors and few kinds of display. The former interfaces are not enabled by default rather need some additional configurations before you can fully utilize them.

- Inter Integrated Circuit is a bus that permits information trade among Micro-controllers and Peripherals with least wiring.
- Serial Peripheral Interfaces is a cousin of Inter Integrated Circuit bus with comparative applications.

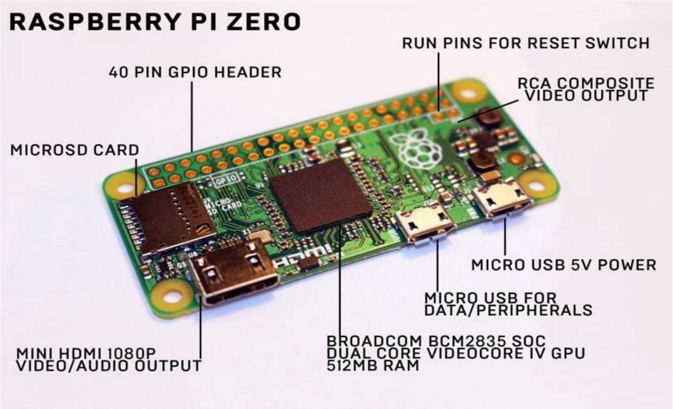

RASPBERRY PI ZERO
RUN PINS FOR RESET SWITCH
40 PIN GPIO HEADER
RCA COMPOSITE VIDEO OUTPUT
MICROSD CARD
MICRO USB 5V POWER
MICRO USB FOR DATA/PERIPHERALS
MINI HDMI 1080P VIDEO/AUDIO OUTPUT
BROADCOM BCM2835 SOC DUAL CORE VIDEOCORE IV GPU 512MB RAM

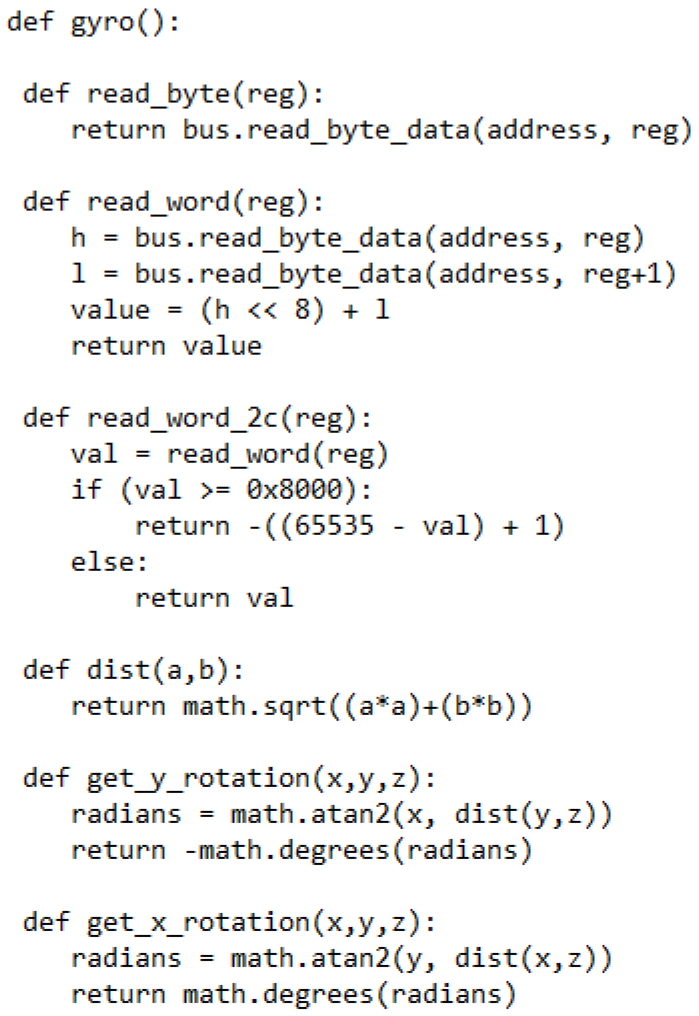
Figure 11 Raspberry Pi Zero [14]

The Pi Zero W has three kinds of Serial Interface on the "General Purpose Input Output" header. The Universal Asynchronous Receiver Transmitter serial port enables you to establish a remote login session from a Sequential Terminal Application, for example, PuTTY. The other two sequential interfaces are the "Serial Peripheral Interface" and "Inter Integrated Circuit bus". Serial Peripheral Interface on the Pi takes into account up to two joined gadgets, while Inter Integrated Circuit transport conceivably considers numerous gadgets, until their addresses start conflicting with each other.

3.2 Accident Detection Mechanism (Using Threshold Value)

After the communication has been established between the Pi Zero W and MPU – 6050 using the above-mentioned libraries, raw values from the sensor (MPU – 6050) can now be read by the micro-controller and these same values after some working are used for accident detection.

1. Firstly, we import WHTML, SPIDEV, time and SMBUS libraries in the code for the purpose of reading the raw values sent from the sensor to the Raspberry Pi using a wired medium.
2. The math library allows us to use built in mathematical functions in python for molding the required values as per their use.



```
def gyro():

 def read_byte(reg):
    return bus.read_byte_data(address, reg)

 def read_word(reg):
    h = bus.read_byte_data(address, reg)
    l = bus.read_byte_data(address, reg+1)
    value = (h << 8) + l
    return value

 def read_word_2c(reg):
    val = read_word(reg)
    if (val >= 0x8000):
        return -((65535 - val) + 1)
    else:
        return val

 def dist(a,b):
    return math.sqrt((a*a)+(b*b))

 def get_y_rotation(x,y,z):
    radians = math.atan2(x, dist(y,z))
    return -math.degrees(radians)

 def get_x_rotation(x,y,z):
    radians = math.atan2(y, dist(x,z))
    return math.degrees(radians)
```

Figure 12 Screenshot Appendix A

For Example:

- math. atan2 function returns the Arctangent in Radians
- math. sqrt is used to give us the square root

3. Finally, after getting raw values and applying mathematical functions, we get six final values. Three of these are accelerometer_xout, accelerometer_yout and accelerometer_zout give us the three-axis reading of acceleration or deceleration of

the MPU – 6050 sensor IC whereas the gyroscope_xout, gyroscope_yout, gyroscope_zout provide us with the three-axis values of angular rate. Once the values have been received by the micro-controller, these are scaled respectively so they can be used efficiently.

4. Once we have the scaled values, we are just left with one task i.e analyzing, manipulating and using these values for accident detection. We have inserted checks at specific points (to be discussed in the next point) in our code to detect the accident. These checks are based on certain threshold values. If the value is greater than a certain threshold, the script reports an accident or sends a 1 flag otherwise it continues its normal operation and the detection flag remains 0. These threshold values were set after careful experimentation and testing. The helmet was tested by a rider in these conditions:

    - Driving over a smooth road
    - Driving over a bumpy road and speed breakers
    - Instant braking and acceleration

    Additionally, the helmet was dropped, and the sensors values were logged for that particular instant that helped us to calculate a generic accident detection threshold. We threw the helmet several times at different speeds and after careful analysis decided the appropriate triggering values.

5. For detecting an accident, we go through the following steps:

- Firstly, define a value “acc_total” which will be the absolute sum of all three accelerometer values that are accelerometer_xout, accelerometer_yout and accelerometer_zout.

```
acc_total = abs(accelerometer_xout) + abs(accelerometer_yout) +
abs(accelerometer_zout)
```

- Then we use “two” *if* conditions for the purpose of accident detection that are:

1. Value of “acc_total” is less than a threshold in our case 40,000.

```
if (acc_total > 40000):
ADV.Accvalue1()
print "ACCIDENT DETECTED"
break
```

2. Using boolean “AND” and “OR” operators between gyroscope and accelerometer value to detect an accident

```
if ((accelerometer_xout_scaled < 0) and (accelerometer_yout_scaled
< 0)) and ((gyroscope_xout < 0) or (gyroscope_yout < 0) or
(gyroscope_zout < 0)):
ADV.Accvalue1()
print "ACCIDENT DETECTED"
break
```

3.3 Use of ML for Accident Detection (Random Forest Classifier)

- Accident detection needs to be as accurate as possible to remove false positives. Our algorithm uses various modes to detect an accident and report it, but it is still susceptible to false positives. Manufacturing a public safety project, special attention needs to be given to this short-fall.
- Our detection platform works based on values computed from accelerometer and gyroscope sensors. These values are updated every second so there is a huge amount of data that's coming through. We decided on implementing '*random forest classifier*' to reduce false positives.
- It was chosen because of its ability to make a '*decision tree'* based on a huge volume of data acquired by the sensors which, according to our research reveals is the appropriate algorithm for this. The output, true or false case is determined through a decision tree because of training dataset we provided, and it used accelerometer & gyroscope (3 axis), total and orientation values.

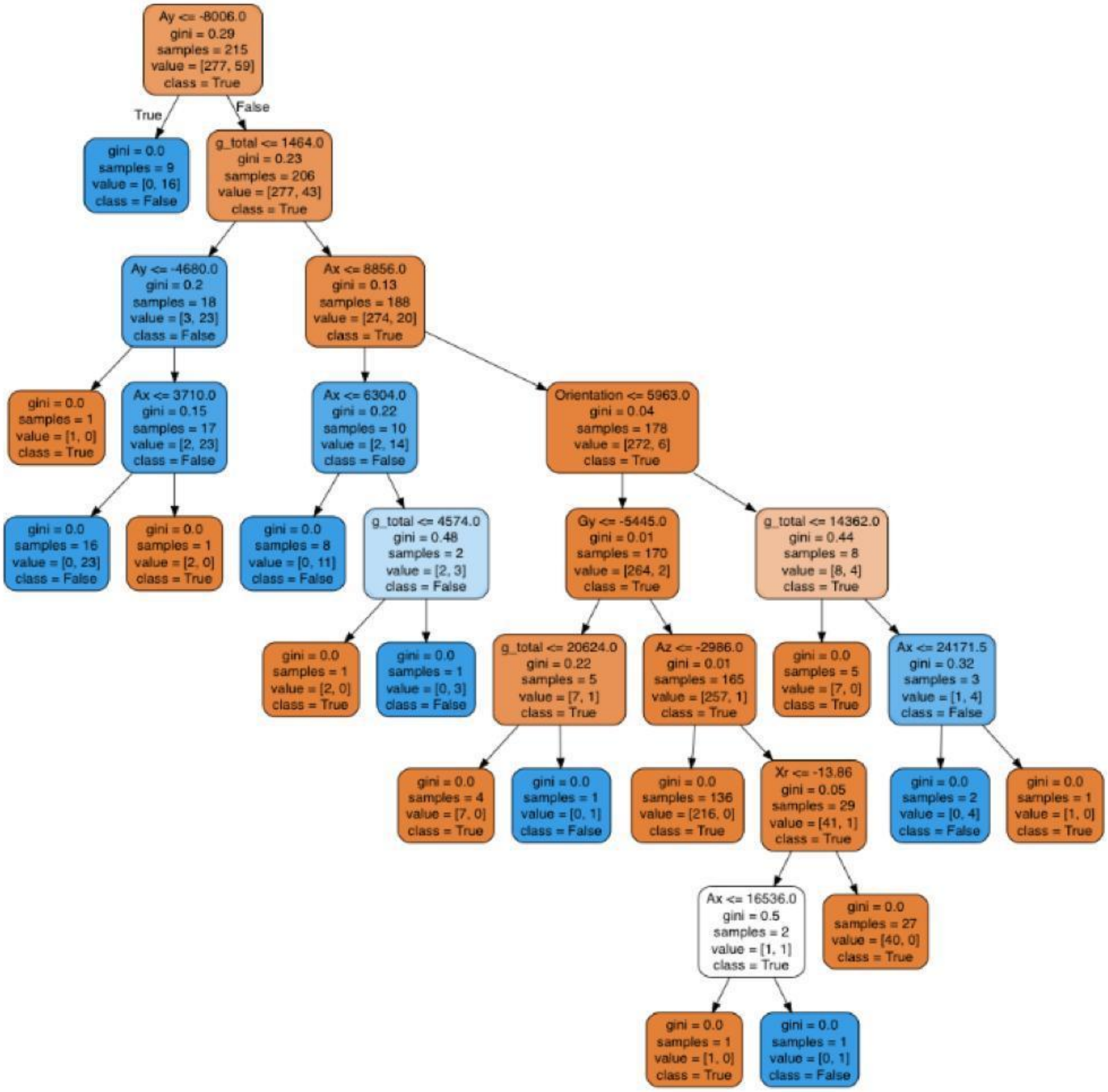


Figure 13 Decision tree Obtained after Training

- With the help of this decision tree, the raw values are classified into different pre-defined scenarios (in our case an accident).

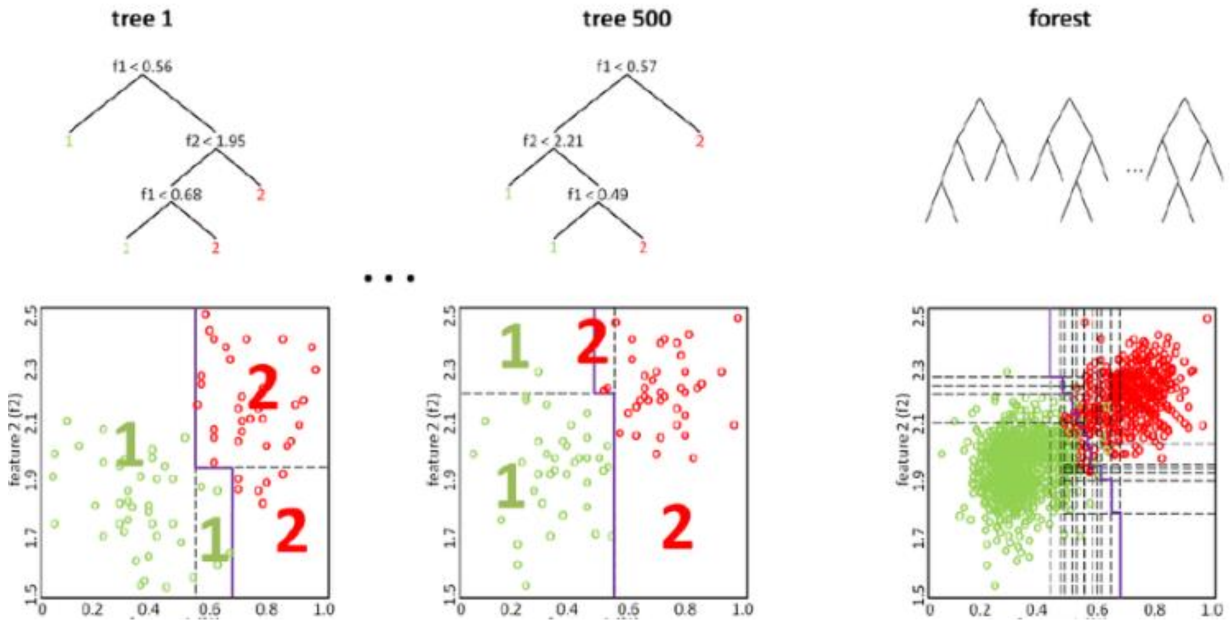


Figure 14 Random Forest Ensemble of test data

- These scenarios are plotted-on graphs and histogram.
- These graphs, when analyzed statistically can reveal patterns.
- The decision tree's conclusion approaches practicality with greater volume of data provided to the classifier.

CHAPTER - 4

Application Development & Interfacing with Raspberry Pi

4.1 Interfacing Options

1) **Personal Hotspot:**

The mobile phone can act as hotspot to provide connectivity to the app with Raspberry Pi. The communication between the app and Raspberry Pi will follow the common TCP/IP architecture and allow for simple client-server communication. The Raspberry Pi will be allotted an IP address from the mobile's DHCP server and then it can communicate over the network just like a LAN.

In our case, we have set-up an apache server on the Raspberry Pi which serves HTML pages. These HTML pages contain the sensor values and keeps the flag updated whether an accident is detected or not. The android application is developed to access this local web server hosted on the Raspberry Pi and extract the sensor values and the accident detection flag. If the server reports this flag as 1, then an accident has occurred otherwise it returns a 0 and normal operation continues.

2) **BTNAP:**

One of the shortcomings of personal hotspot is that it is not very energy efficient and consumes a lot of battery. An alternate energy efficient solution is to keep the underlying TCP/IP architecture same but use an energy efficient communication medium such as Bluetooth or Bluetooth Network Access Point in this case. This extends the phones battery life and at the same time functions as a quick-connected communication protocol.

In this Scenario, the Raspberry Pi becomes the Bluetooth Network Access Point and hosts the DHCP server to allocate IP addresses to the clients.

The phone connects to it as a client, after getting an IP address it retrieves the data such as sensor values and flags for detection from the hosted server.

3) Bluetooth-Socket:

The Bluetooth-Socket provides a more flexible communication framework as compared to BTNAP or Personal Hotspot because after being paired or the idle stage it generates sockets to communicate to services on a need basis. It is not always hosting or connecting which saves a lot of battery. The Bluetooth-Socket follows the same principle as client-server architecture. One node becomes the server and the other node becomes client and the roles can be interchanged. This type of functionality was missing from the prior mentioned interfacing options. This allows us to make a more optimized application to support it.

In this scenario the Bluetooth MAC addresses are used instead of the IP addresses and no network flow is required as in the TCP/IP architecture.

This mode also allows the mobile phone to use its own Data network simultaneously with the Bluetooth setup. This opens other possibilities to integrate additional features in the application that require access to the internet such as remote tracking services. Different API’s (Application Programming Interface) are available, some are paid while others are open-source and can be integrated with application to provide additional functionality. One such API is the Google Maps API that can be integrated with the application to provide navigation services via audio to the bike rider. These types of services usually charge a small amount yearly and are an economical option to provide additional features.

4.2 Android Application:

Android applications are developed in the Java language using the Android Software Development Kit (Android studio in our case). Once they are developed, Android applications can be uploaded on Google Play Store or can be transferred through APKS.

Android powers millions of smart phones around the globe. It is the largest installed operating system on any smart phone and it is growing exponentially.

The following figure shows the **Android Architecture**

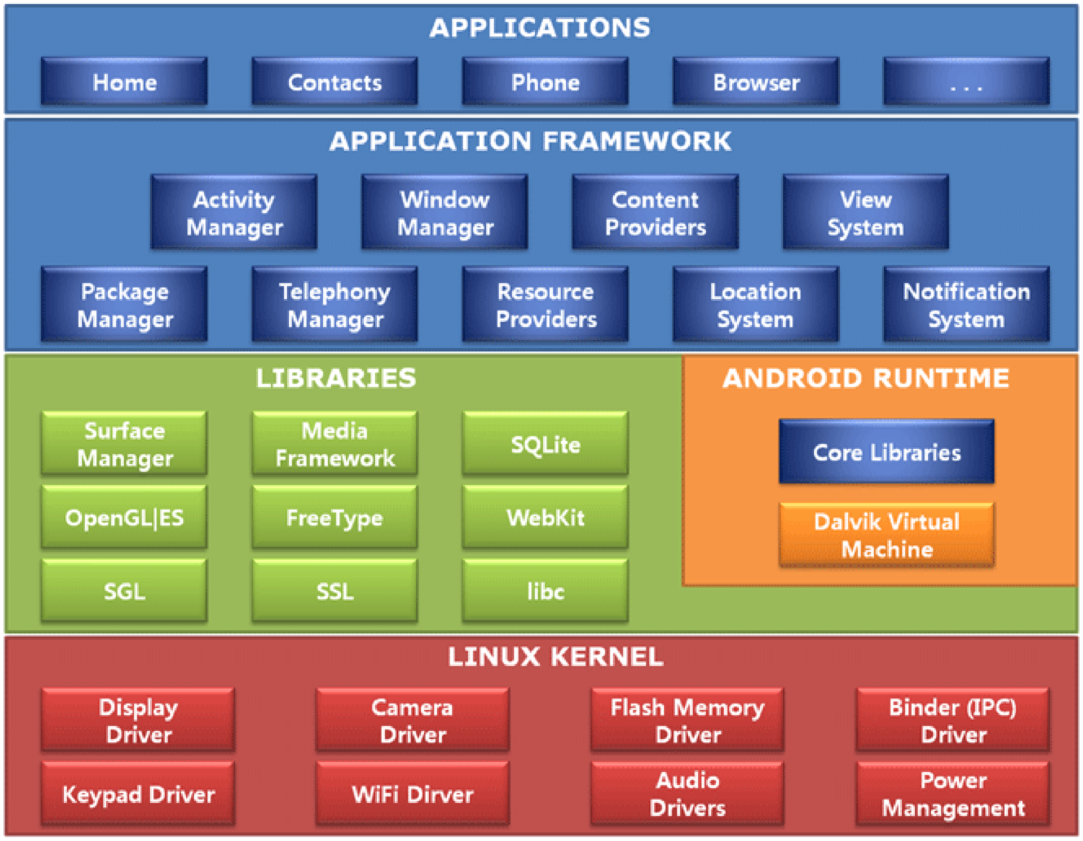


Figure 15 Android Architecture [15]

4.2.1 Linux Kernel:

The first or the bottom layer is Linux kernel. This provides a level of abstraction between the device hardware and software further controlling all the essential hardware drivers. The kernel handles all the things such as networking and controlling the drivers.

4.2.2 Libraries:

On top of Linux kernel, there is a combination of open-source libraries that help automate the OS's tasks and processes. These include libraries like SQLite, SSL, audio and video networking libraries, etc.

4.2.3 Android Core Libraries:

These are Java based libraries offered to the developer for designing customized applications. These include but are not limited to:

- **android.App** : It provides accessibility to the application model
- **android.webkit** : It is related to web browsing in applications
- **android.Content** : It provides content access.
- **android.database** : SQLite management classes.
- **android.OS** : It provides accessibility to standard OS services
- **android.text** : Render and manipulate display.
- **android.view** : Application UI.
- **android.widget** : Buttons, list views, labels, layout managers etc.
- **android.opengl** : Graphics rendering API.

4.2.4 Android Runtime:

This is present on the second layer of the architecture. This layer contains a major framework called **Dalvik Virtual Machine.** This is a JVM. It is specifically designed for Android. It uses Linux core features like multi-threading and memory-management. Every Android application runs in its own instance and this is provided by the Dalvik VM.

4.2.5 Application-Framework:

The Application-Framework is the third layer and provides many higher level services to apps. Developers make use of these services in their apps.

The Android framework provides the following features:

- **Activity Manager**: It controls the activity stack and apps lifecycle.
- **View System:** Create application UI (User Interface).
- **Notifications Manager:** Display notifications and alerts to the user.
- **Resource Manager:** Provides access to resources such as color, strings, UI layouts and settings.

4.2.6 Applications:

All the Android applications are present at the fourth layer. They use all the available resources from layers beneath and customizes them in a way to work together. Examples of such applications are Calculator, Browser, Games, Contacts etc.

4.3 SHADR APP

4.3.1 Permissions/Manifest File

The core libraries needed for the functioning of the SHADR application are included in the manifest file as shown in Appendix A. The XML file contains the necessary permissions required by the application for its proper functioning. This file must be presented to the operating system prior to running the application.

The main entries in our manifest file are shown below

< android.permission.ACCESS_COARSE_LOCATION/>

< android.permission.ACCESS_FINE_LOCATION/>

< android.permission.INTERNET />

< android.permission.SEND_SMS />

These four entries show that the SHADR app can and will perform functions related to SMS, Internet and Location.

4.3.2 Activities

Unlike other programming languages, the android system invokes functions by calling out activities rather than calling them through the main() method. This is a very useful feature and allows activities to be arranged in a specific order to be executed.

The SHADR application comprises of 2 main activities namely:

1) **Login/Data Collection Activity**

   This activity includes the necessary Java code for storing values and its cross checking with a server. The Login information can be hard-coded into the app as well to allow custom usernames and passwords. As each app will be specific to one individual, hard-coding makes it relatively easy to address each user.

   This Activity also stores the medical information of the user which is then passed onto the next activity for further processing. This medical information can include anything from allergies to blood group and is easily customizable by the user. Every User must do this before starting the ride.

   After entering the medical information, the app will ask for emergency contacts so that it can transfer all the medical records to that number incase an accident is occurred.

2) **<u>Main Activity:</u>**

   This is the core activity of the application and it handles the accident reporting framework. It is further sub divided into 3 activities:

   1) **Location Tracking:** This activity monitors for Location Change activity through the GPS and returns new coordinates at every instance the location is changed. These function returns LAT/LONG coordinates.

2) **Sensor Value Acquisition:** A Java library known as Beautiful Soup is used to interpret the values coming from the sensors in the Raspberry Pi by refreshing the web server and getting new values at every instant. The Web server displays the values along with the detection flag whereas the application uses the parsing technique to accumulate the values.

3) **Accident Reporting Through SMS:** The Beautiful Soup listener is constantly listening while the rider is riding the bike for the accident detection flag to turn 1. This means that an accident has occurred and it needs to report the following 3 things to the emergency contacts provided:

   a) Location Coordinates

   b) Speed (calculated by taking coordinates relative to time)

   c) Blood Group

   d) Name, other medical information (Strings)

   As soon as the Flag turns 1, the application initiates an SMS sending API to transfer the above given things to the emergency contacts provided initially.

These are all the major activities and their roles in the SHADR Application.

CHAPTER - 5

<u>Conclusion</u>

5.1 Objectives Achieved

The project provides the capability to detect and report accidents by modifying a motor cycle helmet. This is done using values from accelerometer and gyroscope sensor that are fed into a microcontroller which intimates the android application via Bluetooth. It was achieved with design parameters as follows:

a. Economical

b. Efficient

c. Scalability

The android application stores user emergency details and informs emergency contacts in case of an accident with location and health details eliminating the use of a separate GSM and GPS module.

5.2 Applications

SHADR provides an accident detection and reporting platform not only applicable to motor cycle riders specifically but also implementable across a variety of motor vehicles. The scalability of this project has diverse prospects for the future. This makes a project applicable to a multitude of clientele.

5.2.1 General Public

- A cost-effective solution was the fundamental design parameter that was followed throughout the design of this project.
- According to Pakistan Bureau of Statistics (PBS), 75% [17] of the Pakistani population owns motor cycles and through market research it was revealed, majority of this target market hails from lower middle class to low class. Therefore, this project compromises of minimal hardware components and is based on the ideology of improving software to provide a better user experience.
- To further facilitate such people, a smart helmet solution can be introduced via government and city traffic police incentives to make it more affordable. This would make it more affordable and a viable smart solution for the masses.
- Much care was taken about making mass deployment possible by keeping the android application as simple as possible requiring no training as only user health details and emergency contacts need to be added.

5.2.2 Corporate Clientele

- Ride hailing services like Uber, Careem, Foodpanda, Cheetay & Doodhwala can benefit from this product. Careem, Cheetay & Doodhwala have been contacted and showed interest in the product. A fully functional client-server architecture with remote monitoring not only aids them but also provides a safety perspective with the accident detection and reporting platform as no similar cost-effective solution is available in the market.
- The project falls within United Nation’s Sustainable Development Goal (SDG) – *Public Health & Well Being.* This aligns our project with global corporate and environment goals of human wellbeing that makes it an attractive solution to implement. This enables

introducing our product to a wide spectrum of corporate clientele that have responsible ethical values.

- The interesting aspect of this project is that it is not a piece of hardware only specific to motorcyclists but a platform of accident detection and reporting that can be introduced in the vast automobile industry. The sensors can be installed in any vehicle to detect accidents. Therefore, we can sell our algorithm and methodology via patents to automobile industry, so they can develop a product specific to their needs.

5.2.3 Government Departments

- City Traffic Police motorcyclists and 1122 Emergency Response Teams can also be equipped with this product. It not only ensures safety but also monitoring capability to provide better service to the public.
- Dolphin Police Force from Lahore was going to introduce a smart helmet costing Rs. 38,000 (pkr), this project failed as it was deemed financially as well as practically unviable. Our solution can be modified with added features as per their demand and can be a domestically designed solution, that is substantially cheaper.
- Similarly, military can benefit by using this product for QRF teams as the use of motorcycles is increasing with the advent urban warfare. Added features like wireless communication where mobile networks are not available and head cams can be added as required.

5.3 Future Work

Certain aspects of the project can be further polished to make it more user friendly as we have so far worked on a prototype. If time is given and work is done on it, a marketable product can be brought to consumers and clients.

5.3.1 Communication

- Bluetooth Personal Area Network is currently being used for communication between Raspberry Pi 0W and android application. This does not give room for adding a lot of features on top of the HTML parsed value being used for intimation of accident status on smart phone.
- Therefore, Bluetooth Low Energy (BLE) enabled sock interface or a WiFi hotspot can be introduced. This provides room to implement features like personal assistant used in navigation or wireless audio functionality.
- It is very beneficial as it can be fully implemented on software and requires little to no hardware addition. Therefore, costs do not increase per unit; rather one-time development cost to fully implement this communication mode.

Figure 16 Navigation System [16]

- This would enable more energy-efficient and a beneficial to the end user as it would enable the developer to add more features via software development.
- Battery conservation can be made possible by using any of the above-mentioned communication modes. This improves usage time in between charges as well as the over-all user experience.

5.3.2 Data Gathering and Analysis for ML

Recognizing and omitting false positive accident reports was one of the major hurdles in our way. The Random Forest Classifier mentioned above, is according to our research so far, the most suitable way to detect and thus reduce these false positive reports. Radom Forest Classifier requires extensive data sets and hours or testing and trials to give practical results.

We gathered some data for two basic scenarios, the true one being that an accident has occurred and the false case for no accident. We were successful in gathering data for the later i.e. the false case by driving a motor bike at random speeds and places for hours (on smooth and bumpy surfaces). The data was collected at an interval of one half of a second. The data for the true case was collected by throwing the helmet on different surfaces and by dropping the helmet of a motorbike at different speeds.

So, our research directed us towards a conclusion where a combination of both threshold values and decision using a Machine Learning algorithm with proper data collection and analysis can definitely help us improve our accident detection and reporting platform.

5.3.3 Additional Features

- Features like headcam, rear indicators, wireless headphones, sensors for further enhancing accident detection capability can be added.
- So, the motivation by improving the helmet by the addition of such features would be to cater for specific clientele like people who can afford an expensive, feature intensive headphone and government/corporate clients that would want specific features to be introduced in their variant of smart helmet.

- This will enable our product to target a broad clientele not only limited to the base model but manufacturing variants specific to needs of the consumer.

5.3.4 Product Development

- So far, we have worked on a prototype. This uses Raspberry Pi which is an expensive microcontroller and whose processing power is not fully utilized. Therefore, it is neither feasible nor smart to use it on a large scale.
- The prototype also costs a lot because market products like battery pack, helmet and sensor were used. If this project is to be sold as a marketable product, product design tailored to a customized helmet with room for sensors and other equipment that needs to be purchased in bulk. This will reduce product development costs by approximately 50%.
- To achieve that cost reduction of 50%, a custom microcontroller needs to be used with the sole purpose of interfacing with MPU-6050 and for communicating status with android application. As Raspberry Pi costs Rs. 3,000, this custom microcontroller will drastically reduce manufacturing costs. But ample research needs to be put in to make it possible.
- Similarly, as a part of market research, helmet manufacturers and dealers were contacted from Karachi like *Unique* & *Uneed* Helmets. It revealed that helmet manufacturing in Pakistan is not viable as product quality is not up to international safety standards at all. It was also revealed that only viable option is to directly contact Chinese helmet manufacturers to design a helmet according to our design specifications and assemble it with the required sensors. This removes the need for having an independent Pakistan based plant as it is not financially viable against a Chinese alternative. Outsourcing is the future!

Appendix A

MPU – 6050 (Accelerometer and Gyroscope) - Python Script

```
#!/usr/bin/python



import whtml as ADV

import spidev

import time

import smbus

import math# Register

power_mgmt_1 = 0x6b

power_mgmt_2 = 0x6c



def IC():                                  #Define Variables

  delay = 7

  ldr_channel = 0

  x = 1

  spi = spidev.SpiDev()       # Create SPI

  spi.open(0, 0)



def gyro():

  def read_byte(reg):
```

```
  return bus.read_byte_data(address, reg)


def read_word(reg):

  h = bus.read_byte_data(address, reg)

  l = bus.read_byte_data(address, reg + 1)

value = (h << 8) + l

return value


def read_word_2c(reg):

  val = read_word(reg)

if (val >= 0x8000):

  return -((65535 - val) + 1)

else :

  return val


def dist(a, b):

  return math.sqrt((a * a) + (b * b))


def get_x_rotation(x, y, z):

  radians = math.atan2(y, dist(x, z))

  return math.degrees(radians)
```

```
bus = smbus.SMBus(1)            # bus = smbus.SMBus(0) fuer Revision 1

address = 0x68# via i2cdetect


# Activate to be able to address the module
bus.write_byte_data(address, power_mgmt_1, 0)

while True:

  ADV.Accvalue1()

  print "Gyroscope"

  print "--------"


gyroscope_zout = read_word_2c(0x43)

gyroscope_xout = read_word_2c(0x45)

gyroscope_yout = read_word_2c(0x47)


print " accelerometer_xout: ", ("%5d" % gyroscope_yout), " scaled: ", (gyroscope_yout / 131)

print " accelerometer_yout: ", ("%5d" % gyroscope_xout), " scaled: ", (gyroscope_xout / 131)

print " accelerometer_zout: ", ("%5d" % gyroscope_zout), " scaled: ", (gyroscope_zout / 131)


print

print "Accelerometer sensor"
print "---------------------"


gyroscope_yout = read_word_2c(0x3b)

gyroscope_xout = read_word_2c(0x3d)

gyroscope_zout = read_word_2c(0x3f)


accelerometer_xout_scaled = gyroscope_xout / 16384.0


print "accelerometer_xout: ", ("%6d" % accelerometer_xout), " scaled: ",
accelerometer_xout_scaled

print "accelerometer_yout: ", ("%6d" % accelerometer_yout), " scaled: ",
accelerometer_yout_scaled

print "accelerometer_zout: ", ("%6d" % accelerometer_zout), " scaled: ",
accelerometer_zout_scaled


print "accelerometer_total: ", acc_total


print "X Rotation: ", get_x_rotation(accelerometer_xout_scaled, accelerometer_yout_scaled,
accelerometer_zout_scaled)

print "Y Rotation: ", get_y_rotation(accelerometer_xout_scaled, accelerometer_yout_scaled,
accelerometer_zout_scaled)

time.sleep(0.1)
```

```
if ((accelerometer_xout_scaled > 0) and(accelerometer_yout_scaled < -74))
and((gyroscope_xout < 1500) or(gyroscope_yout >0) or(gyroscope_zout > 0)):

  ADV.Accvalue0()

print "ACCIDENT DETECTED"

break

IC()

ADV.Accvalue1()

gyro()
```

Update Valon Apache

```
def Accvalue1():

  f = open('/var/www/html/index.html','w')

  message = """<html>

  <head></head>

  <title> Accident Detection:1(True)</</title>

  <body><p>1></body>

  </html>"""


  f.write(message)

  f.close()
```

```
 message = """<html>

 <head></head>

 <title> Accident Detection:0(False)</title>

 <body><p>0></body>

 </html>"""


 f.write(message)

f.delete(message)

 f.close()
```

Appendix B

Login activity App

```
package testing.gps_location;



import android.animation.Animator;

import android.content.Intent;

import android.animation.AnimatorListenerAdapter;

import android.annotation.TargetApi;

import android.content.pm.PackageManager;
import android.support.annotation.NonNull;

import android.support.design.widget.Snackbar;

import android.support.v7.app.AppCompatActivity;

import android.app.LoaderManager.LoaderCallbacks;

import android.content.ContentResolver;

import android.content.CursorLoader;

import android.content.Loader;

import android.database.Cursor;

import android.net.Uri;

import android.os.AsyncTask;

import android.os.Build.VERSION;

import android.os.Build;

import android.os.Bundle;

import android.provider.ContactsContract;

import android.text.TextUtils;

import android.view.KeyEvent;

import android.view.View;

import android.view.View.OnClickListener;

import android.view.inputmethod.EditorInfo;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;
import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import java.util.ArrayList;

import java.util.List;


import static android.Manifest.permission.READ_CONTACTS;


/**

 * A login screen that offers login via email/password.

 */

public class LoginActivity extends AppCompatActivity implements LoaderCallbacks < Cursor >
{


 /**

  * Id to identity READ_CONTACTS permission request.

  */

 private static final int REQUEST_READ_CONTACTS = 0;




 /**
  * A dummy authentication store containing known user names and passwords.
 * TODO: remove after connecting to a real authentication system.
 */
 private static final String[] DUMMY_CREDENTIALS = new String[]
 {
  "foo@example.com:hello",
  "bar@example.com:world"
   };
 /**
 * Keep track of the login task to ensure we can cancel it if requested.
 */
 private UserLoginTask mAuthTask = null;

 // UI references.
 private AutoCompleteTextView mEmailView;
 private EditText mPasswordView;
 private View mProgressView;
 private View mLoginFormView;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_login);

// Set up the login form.

mEmailView = (AutoCompleteTextView) findViewById(R.id.email);

populateAutoComplete();


mPasswordView = (EditText) findViewById(R.id.password);

 mPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener()

 {

     return true;

   }

     return false;

  }

 });


 Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);

 mEmailSignInButton.setOnClickListener(new OnClickListener() {

 @Override

 public void onClick(View view) {

 String message = mEmailView.getText().toString();

 String Number = mPasswordView.getText().toString();
 Intent i = new Intent(LoginActivity.this, MainActivity.class);

 i.putExtra("message", message);

 i.putExtra("Number", Number);

 startActivity(i);

 attemptLogin();

  }

 });


mLoginFormView = findViewById(R.id.login_form);

mProgressView = findViewById(R.id.login_progress);

}


private void populateAutoComplete() {

 if (!mayRequestContacts()) {

   return;

  }


 if (VERSION.SDK_INT >= 14) {

 // Use ContactsContract.Profile (API 14+)

 getLoaderManager().initLoader(0, null, this);

  }
  else if (VERSION.SDK_INT >= 8) {

  // Use AccountManager (API 8+)

  new SetupEmailAutoCompleteTask().execute(null, null);

   }

   }



 private boolean mayRequestContacts() {

  if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {

     return true;

  }

  if (checkSelfPermission(READ_CONTACTS) ==
PackageManager.PERMISSION_GRANTED) {

     return true;

  }

  if (shouldShowRequestPermissionRationale(READ_CONTACTS)) {

    @TargetApi(Build.VERSION_CODES.M)

    public void onClick(View v) {

    requestPermissions(new String[]

   {

      READ_CONTACTS

      }, REQUEST_READ_CONTACTS);
      }
     });
        } else
  {
   requestPermissions(new String[] {
   READ_CONTACTS
   }, REQUEST_READ_CONTACTS);
   }
       return false;
   }
 /**
 * Callback received when a permissions request has been completed.
 */
 @Override
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
 if (requestCode == REQUEST_READ_CONTACTS) {
 if (grantResults.length == 1 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
 populateAutoComplete();
   }
   }
   }
 /**
 * Attempts to sign in or register the account specified by the login form.
 * If there are form errors (invalid email, missing fields, etc.), the
 * errors are presented and no actual login attempt is made.
 */
 private void attemptLogin() {
  if (mAuthTask != null) {
      return;
  }
  // Reset errors.
  mEmailView.setError(null);
  mPasswordView.setError(null);

  // Store values at the time of the login attempt.
  String email = mEmailView.getText().toString();
  String password = mPasswordView.getText().toString();

  boolean cancel = false;
  View focusView = null;
```

```
// Check for a valid password, if the user entered one.
if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
mPasswordView.setError(getString(R.string.error_invalid_password));
focusView = mPasswordView;
cancel = true;
 }

// Check for a valid email address.
if (TextUtils.isEmpty(email))
}
 else if (!isEmailValid(email)) {
 mEmailView.setError(getString(R.string.error_invalid_email));
 focusView = mEmailView;
 cancel = true;
}

if (cancel) {
 // There was an error; don't attempt login and focus the first
 // form field with an error.
 focusView.requestFocus();
```

```
  }
Else
 {
   // Show a progress spinner, and kick off a background task to
   // perform the user login attempt.
   showProgress(true);
   mAuthTask = new UserLoginTask(email, password);
   mAuthTask.execute((Void) null);
  }
  }

 private boolean isEmailValid(String email) {
  //TODO: Replace this with your own logic
  return email.contains("@");
 }

 private boolean isPasswordValid(String password) {
 //TODO: Replace this with your own logic
     return password.length() > 4;
 }
  /**
  * Shows the progress UI and hides the login form.
  */
 @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
 private void showProgress(final boolean show) {
 // On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow
 // for very easy animations. If available, use these APIs to fade-in
 // the progress spinner.
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2)
 {
  int shortAnimTime = getResources().getInteger(android.R.integer.config_shortAnimTime);


   mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
   mLoginFormView.animate().setDuration(shortAnimTime).alpha
(
   show ? 0 : 1).setListener(new AnimatorListenerAdapter()
{
   @Override
    public void onAnimationEnd(Animator animation)
{
    mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
    }
   });


   public void onAnimationEnd(Animator animation) {
   mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
    }
   });
    }
else
 {
   // The ViewPropertyAnimator APIs are not available, so simply show
   // and hide the relevant UI components.
   mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
   mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
   }
   }
 @Override
 public Loader < Cursor > onCreateLoader(int i, Bundle bundle)
 {
  return new CursorLoader(this,
 // Retrieve data rows for the device user's 'profile' contact.
 Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,
ContactsContract.Contacts.Data.CONTENT_DIRECTORY), ProfileQuery.PROJECTION,


// Select only email addresses.

ContactsContract.Contacts.Data.MIMETYPE +

" = ?", new String[]

{

ContactsContract.CommonDataKinds.Email

.CONTENT_ITEM_TYPE

  },


// Show primary email addresses first. Note that there won't be

// a primary email address if the user hasn't specified one.

ContactsContract.Contacts.Data.IS_PRIMARY + " DESC");

 }


@Override

public void onLoadFinished(Loader < Cursor > cursorLoader, Cursor cursor) {

List < String > emails = new ArrayList < > ();

cursor.moveToFirst();

while (!cursor.isAfterLast())

{
 emails.add(cursor.getString(ProfileQuery.ADDRESS));
 cursor.moveToNext();
 }
addEmailsToAutoComplete(emails);
}
@Override
public void onLoaderReset(Loader < Cursor > cursorLoader) {
}


 mEmailView.setAdapter(adapter);
 }
private interface ProfileQuery {
String[] PROJECTION = {
ContactsContract.CommonDataKinds.Email.ADDRESS,
ContactsContract.CommonDataKinds.Email.IS_PRIMARY,
 };

 int ADDRESS = 0;
 int IS_PRIMARY = 1;
 }
 /**
 * Use an AsyncTask to fetch the user's email addresses on a background thread, and update
 * the email text field with results on the main UI thread.
 */
 class SetupEmailAutoCompleteTask extends AsyncTask < Void,
 Void,
 List < String >>
{
  @Override
  protected List < String > doInBackground(Void...voids) {
  ArrayList < String > emailAddressCollection = new ArrayList < > ();

   // Get all emails from the user's contacts and copy them to a list.
   ContentResolver cr = getContentResolver();
   Cursor emailCur = cr.query(ContactsContract.CommonDataKinds.Email.CONTENT_URI,
null,
   null, null, null);
   while (emailCur.moveToNext())
 {
   String email = emailCur.getString(emailCur.getColumnIndex(ContactsContract
   .CommonDataKinds.Email.DATA));
   emailAddressCollection.add(email);
  }
  emailCur.close();

  return emailAddressCollection;
 }

 @Override
 protected void onPostExecute(List < String > emailAddressCollection) {
  addEmailsToAutoComplete(emailAddressCollection);
 }
 }
/**
 * Represents an asynchronous login/registration task used to authenticate
 * the user.
 */
public class UserLoginTask extends AsyncTask < Void,
Void,
Boolean >
 UserLoginTask(String email, String password) {
 mEmail = email;
 mPassword = password;

 }

 @Override

 protected Boolean doInBackground(Void...params)

{

 // TODO: attempt authentication against a network service.

 try

{

 // Simulate network access.

 Thread.sleep(2000);

  }

 catch (InterruptedException e)

 {

       return false;

  }



  for (String credential: DUMMY_CREDENTIALS) {

   String[] pieces = credential.split(":");

   if (pieces[0].equals(mEmail)) {

   // Account exists, return true if the password matches.

       return pieces[1].equals(mPassword);
```

```
     }

     }


    // TODO: register the new account here.

          return true;

    }

    if (success)

  {

    finish();

    }

  else

   {

    mPasswordView.setError(getString(R.string.error_incorrect_password));

    mPasswordView.requestFocus();

    }

    }

   @Override

   protected void onCancelled() {

   mAuthTask = null;

   showProgress(false);

   }  }  }
```

Main Activity (speed, long, lat., location change)

```
package testing.gps_location;


import android.Manifest;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Build;

import android.os.Bundle;

import android.provider.Settings;

import android.support.annotation.NonNull;

import android.support.annotation.Nullable;

import android.support.v4.app.ActivityCompat;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;

import java.io.IOException;

import java.util.jar.Attributes;

import android.telephony.SmsManager;

public class MainActivity extends AppCompatActivity {


 private Button b;

 private TextView t;

 private LocationManager locationManager;

 private LocationListener listener;

 private Button b2;

 private TextView t2;

 double curTime = 0;

 double oldLat = 0.0;

 double oldLon = 0.0;


 @Override

 protected void onCreate(@Nullable Bundle savedInstanceState)

{

 super.onCreate(savedInstanceState);

  setContentView(R.layout.activity_main);
  Bundle bundle = getIntent().getExtras();
  final String message = bundle.getString("message");
  final String Number = bundle.getString("Number");
  t = (TextView) findViewById(R.id.textView);
  b2.setOnClickListener(new View.OnClickListener() {
 @Override
  public void onClick(View view)
{
  getWebsite();
  }
  });

  locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

  listener = new LocationListener() {
  @Override
  public void onLocationChanged(Location location) {
  t.setText("\n " + "Lat:" + location.getLongitude() + " " + " Long:" + location.getLatitude());
  double loc1 = location.getLatitude();
  double loc2 = location.getLongitude();
```

```
  final String Loc1 = Double.toString(loc2);
  final String Loc2 = Double.toString(loc1);
    final double x = getspeed(location);
    new Thread(new Runnable() {
    @Override
     public void run() {
     final StringBuilder builder = new StringBuilder();
     try {
     Document doc = Jsoup.connect("http://192.168.20.99").get();
     String title = doc.title();
     String a = "1";

     String text = doc.text();
     builder.append(text);
     builder.append("\n" + "\n" + "Speed: " + (x * 3500) + "Km/h" + "\n\n" + message);
     if (title.equals(a))
{
    String Loc = Loc1 + "," + Loc2;
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(Number, null, "ALERT!!!\n*************\n" + message + "
Was in an Accident at\n" + "Speed:" + ((x * 3500)) + "km/h\nLocation: " +  Loc, null, null);
```

```
    builder.append(" at\n");

    builder.append(Loc + "\n\nSENDING SMS");

      }

      }

  catch (IOException e)

{

  builder.append("Error:");

     }

 runOnUiThread(new Runnable() {

@Override

 public void run()

{

  t2.setText(builder.toString());


     }

    });

     }

    })

.start();

     }

 @Override
```

```
  public void onStatusChanged(String s, int i, Bundle bundle) {

     }

  @Override

  public void onProviderEnabled(String s) {

    }

  @Override

  public void onProviderDisabled(String s) {



  Intent i = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);

  startActivity(i);

   }

   };

 configure_button();

   }


private void getWebsite() {

new Thread(new Runnable() {

@Override

public void run() {

final StringBuilder builder = new StringBuilder();

try
```

```
   {
 Document doc = Jsoup.connect("http://192.168.20.99").get();
 String title = doc.title();
 builder.append(title);
      }
 catch (IOException e) {
 builder.append("Error:");
     }
 runOnUiThread(new Runnable() {


    }
    });
    }
    })
.start();
    }
private double getspeed(Location location)
 {
 double newTime = System.currentTime();
 double newLat = location.getLatitude();
 double newLon = location.getLongitude();
```

```
  if (location.hasSpeed())

 {

   double speed = location.getSpeed();

   Toast.makeText(getApplication(), "SPEED : " + speed * 3500+ "Km/h",
Toast.LENGTH_SHORT).show();

   return speed;

     }

else

 {

   double distance = calculationBydistance(newLat, newLon, oldLat, oldLon);

   double timeDifferent = curTime - newTime ;

   double speed = distance + timeDifferent/ timeDifferent;

   curTime = newTime;

   oldLat = newLat;

   oldLon = newLon;

Toast.makeText(getApplication(), "SPEED 2 : " + String.valueOf(speed * 3500) + "km/h",
Toast.LENGTH_SHORT).show();

   return speed;

  }

  }

 public double calculationBydistance(double lat1, double lon1, double lat2, double lon2) {

 double radius = 6717000;
 double dLat = Math.toRadians(lat2 - lat1);

 double dLon = Math.toRadians(lon2 - lon1);

 double a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +

  Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) *

  Math.sin(dLon / 2) * Math.sin(dLon / 2);

  double c = 2 * Math.asin(Math.sqrt(a));

  return radius * c;

   }

 @Override

 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {

 switch (requestCode) {

 case 10:

 configure_button();

 break;

 default:

 break;

    }

    }

void configure_button() {

 // first check for permissions
 if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

  }

   return;

   }

 // this code won't execute IF permissions are not allowed, because in the line above there is
return statement.

 b.setOnClickListener(new View.OnClickListener() {

@Override

 public void onClick(View view) {

 //noinspection MissingPermission

 locationManager.requestLocationUpdates("gps", 5000, 0, listener);

   }

  });

  }

  }
```

Appendix C

MainifestsXml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="testing.gps_location">

 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
 <uses-permission android:name="android.permission.INTERNET" />
 <uses-permission android:name="android.permission.SEND_SMS" />

 <!-- To auto-complete the email text field in the login form with the user's emails -->
 <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.READ_PROFILE" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />

    <application
     android:allowBackup="true"
     android:icon="@mipmap/ic_launcher"
     android:label="@string/app_name"
     android:supportsRtl="true"
```

```
android:theme="@style/AppTheme">

<activity

android:name=".LoginActivity"

android:label="@string/title_activity_login">

<intent-filter>

<action android:name="android.intent.action.MAIN" />



<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name=".MainActivity">

</activity>

</application>



</manifest>
```

Appendix D

Random Forest Classifier

```
# In[10]:


from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(labels)


# apply encoding to labels
labels = le.transform(labels)


# In[11]:


df.sample(5)


# In[12]:


df_selected = df.drop(['TIME', 'Ax','Ay','Az','Gx','Gy','Gz',"STATE"], axis=1)


# In[13]:
```

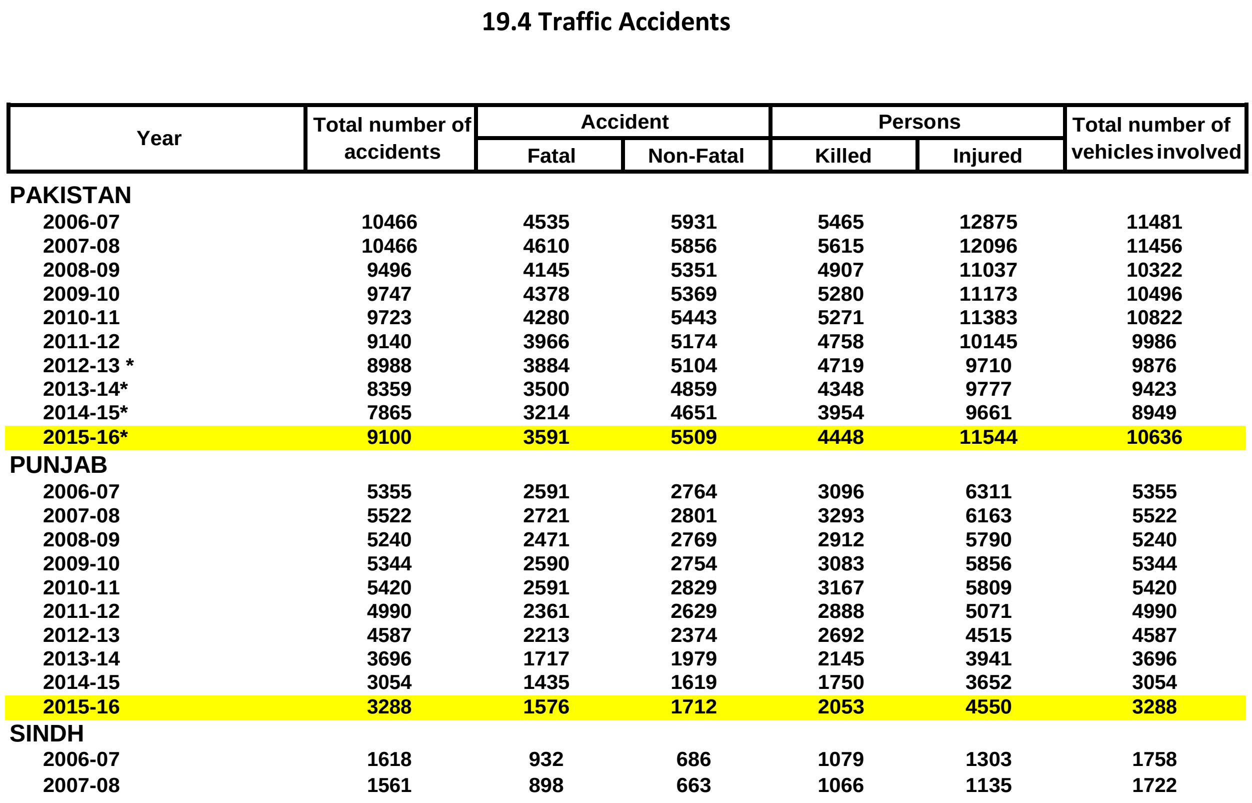
```
df_features = df_selected.to_dict(orient='records')
df.sample(5)


# In[14]:


from sklearn.model_selection import train_test_split
features_train, features_test, labels_train, labels_test = train_test_split(
  features, labels,
   test_size=0.20, random_state=42)
print(labels_train,)


# In[37]:


# import
from sklearn.ensemble import RandomForestClassifier


# initialize
clf = RandomForestClassifier()


# train the classifier using the training data
clf.fit(features_train, labels_train)
```

```
# In[38]:


# compute accuracy using test data

acc_test = clf.score(features_test, labels_test)

print ("Test Accuracy:", acc_test)

# Test Accuracy: 0.98


# In[39]:


acc_train = clf.score(features_train, labels_train)

print ("Train Accuracy:", acc_train)

# Train Accuracy: 0.98


# In[42]:


flower = [[-4,78]]

class_code = clf.predict(flower) # [1]


# In[43]:
```

```
print (clf.predict(flower))



# In[44]:



decoded_class = le.inverse_transform(class_code)

print (decoded_class)



# In[ ]:
```

Appendix E

Accident Statistics [17]



19.4 Traffic Accidents

Year	Total number of accidents	Accident		Persons		Total number of vehicles involved
		Fatal	Non-Fatal	Killed	Injured	
PAKISTAN						
2006-07	10466	4535	5931	5465	12875	11481
2007-08	10466	4610	5856	5615	12096	11456
2008-09	9496	4145	5351	4907	11037	10322
2009-10	9747	4378	5369	5280	11173	10496
2010-11	9723	4280	5443	5271	11383	10822
2011-12	9140	3966	5174	4758	10145	9986
2012-13 *	8988	3884	5104	4719	9710	9876
2013-14*	8359	3500	4859	4348	9777	9423
2014-15*	7865	3214	4651	3954	9661	8949
2015-16*	9100	3591	5509	4448	11544	10636
PUNJAB						
2006-07	5355	2591	2764	3096	6311	5355
2007-08	5522	2721	2801	3293	6163	5522
2008-09	5240	2471	2769	2912	5790	5240
2009-10	5344	2590	2754	3083	5856	5344
2010-11	5420	2591	2829	3167	5809	5420
2011-12	4990	2361	2629	2888	5071	4990
2012-13	4587	2213	2374	2692	4515	4587
2013-14	3696	1717	1979	2145	3941	3696
2014-15	3054	1435	1619	1750	3652	3054
2015-16	3288	1576	1712	2053	4550	3288
SINDH						
2006-07	1618	932	686	1079	1303	1758
2007-08	1561	898	663	1066	1135	1722

2008-09	1433	824	609	961	1160	1562
2009-10	1465	883	582	1031	1261	1580
2010-11	1270	758	512	927	1071	1541
2011-12	1054	681	373	756	681	1121
2012-13	935	582	353	696	637	960
2013-14	945	613	332	791	893	1103
2014-15	881	583	298	771	863	1029
2015-16	924	634	290	749	754	1144
KHYBER PAKHTUNKHWA						
2006-07	2942	779	2163	1006	4421	3756
2007-08	2893	755	2138	942	3884	3634
2008-09	2392	644	1748	786	3340	2975
2009-10	2559	712	1847	921	3560	3128
2010-11	2722	773	1949	986	4153	3479
2011-12	2772	785	1987	953	3913	3501
2012-13	2968	846	2122	1059	4016	3736
2013-14	3120	877	2243	1033	4257	3934
2014-15	3399	942	2457	1137	4524	4260
2015-16	4287	1083	3204	1299	5527	5490
BALOCHISTAN						
2006-07	551	233	318	284	840	612
2007-08	490	236	254	314	914	278
2008-09	431	206	225	248	747	545
2009-10	379	193	186	245	496	444
2010-11	311	158	153	191	350	382
2011-12	324	139	185	161	480	374
2012-13	297	136	161	163	362	381
2013-14	342	173	169	247	480	434
2014-15	315	147	168	178	440	389
2015-16	357	178	179	207	504	470
ISLAMABAD						
2012-13	201	107	94	109	180	212
2013-14	256	120	136	132	206	256
2014-15	216	107	109	118	182	217
2015-16	244	120	124	140	209	244

*- Islamabad Included.

Source: Provincial Police Department (Crime Branch)& Islamabad Police

Registered Motorbikes [17]

Year	Motor Cars jeeps & station wagons	Motor Cabs/taxis	Buses/Mini-Buses	Trucks	Motor Cycles 2 wheels	Motor Cycles/Motor Rickshaws	Others 3 Wheels	Total
PAKISTAN								
2004	1298353	90460	166136	179727	2609442	138153	848688	5330959
2005	1318488	91893	168713	182516	2649910	101058	861851	5374429
2006	1372191	105373	175589	189950	2757842	136394	896014	5633353
2007	1440801	103397	184368	199447	2895734	143215	940851	5907813
2008	1549854	104431	187367	202574	3039815	156068	961646	6201755
2009	1657860	106463	195163	210944	3215583	167910	1005441	6559364
2010	1726347	122882	198790	216119	4305121	201827	1081916	7853002
2011(P)	1881560	124651	202476	225075	5781953	266390	1178890	9660995
2012	2094289	143859	215374	240888	7500182	323189	1270788	11788569
2013	2281083	145234	220347	247197	9064547	377997	1334372	13670777
2014	2400690	145418	223624	251339	10341326	429319	1376369	15168085
PUNJAB								

2004	580189	20937	99554	62930	1743839	56370	563593	3127412
2005	589187	21292	101098	63907	1770883	27246	572333	3145946
2006	613185	23694	105218	66510	1843012	59577	595020	3306216
2007	643844	23931	110479	69835	1935163	62556	624795	3470603
2008	692576	24171	112276	70930	2031449	68170	638605	3638177
2009	740840	24641	116948	73861	2148912	73343	667688	3846233
2010	807162	24818	118366	76214	3078239	98340	708058	4911197
2011(P)	905240	26511	121306	79656	4153262	124181	767769	6177925
2012	1055975	45519	132270	87153	5342692	153263	825608	7642480
2013	1180078	46894	135691	89632	6608422	183645	868465	9112827
2014	1264574	47077	138579	91538	7724642	220807	907638	10394855
SINDH								
2004	536972	46997	25706	29398	719882	54525	160944	1574424
2005	545300	47726	26105	29854	731046	46131	163440	1589602
2006	567510	56186	27169	31070	760822	48009	169919	1660685
2007	595886	53718	28527	32624	798863	50410	178422	1738450
2008	640988	54255	28991	33135	838612	54934	182365	1833280
2009	685657	55311	30198	34504	887102	59102	190671	1942545
2010	686534	71539	31796	35520	1029454	67085	223848	2145776
2011(P)	738767	71556	32275	38107	1351197	101986	235488	2569376
2012	793849	71723	33454	43304	1800006	127468	259048	3128852
2013	848561	71723	33931	44717	2010246	146074	270650	3425902
2014	883635	71723	34317	46823	2169548	160233	273436	3639715
KHYBER PAKHTOON KHWA								
2004	147810	21911	34496	50275	93632	23430	79583	451137
2005	150102	22251	35031	51055	95084	23794	80818	458135
2006	156216	24797	36459	53135	98957	24763	84021	478348
2007	164027	25045	38282	55791	103904	26001	88226	501276
2008	176442	25295	38905	56666	109074	28335	90176	524893
2009	188738	25788	40523	59007	115381	30485	94282	554204
2010	188770	25790	40539	59078	121555	31337	95377	562446
2011(P)	193023	25817	40662	61161	188727	34992	103216	647598
2012	199488	25850	41227	63200	248443	36457	112062	726727
2013	206983	25850	42154	65034	327964	42272	120708	830965
2014	207020	25851	42157	65164	329221	42273	120746	832432
BALOCHISTAN								
2004	33382	615	6380	37124	52089	3828	44568	177986
2005	33899	624	6479	37700	52897	3887	45260	180746
2006	35280	696	6743	39235	55051	4045	47054	188104
2007	37044	703	7080	41197	57804	4248	49408	197484
2008	39848	710	7195	41843	60680	4629	50500	205405
2009	42625	723	7494	43572	64188	4980	52800	216382
2010	43881	735	8089	45307	75873	5065	54633	233583
2011(P)	44530	767	8233	46151	88767	5231	72417	266096
2012	44977	767	8423	47231	109041	6001	74070	290510
2013	45461	767	8571	47814	117915	6006	74549	301083
2014	45461	767	8571	47814	117915	6006	74549	301083

a= Islamabad, N. Areas and AJK included.

P= Provisional

Source: Provincial Excise & Taxation Departments, Punjab, Sindh, NWFP, Balochistan, AJK and Northern Areas.

References & Works Cited

[1] Global Health Observatory (GHO) data, 2013,

Available: https://www.who.int/gho/en/

[Accessed May 3, 2018]

[2] Sharief, Samiullah. “Road Accidents in Pakistan Reach Alarming High: Who's Responsible?” PakWheels Blog, 16 Sept. 2016,

Available: www.pakwheels.com/blog/alarming-increase-of-road-accidents-in-pakistan/.

[Accessed May 5, 2018]

[3] “Eight Reasons Motorcyclists in Pakistan Are More Prone to Road Accidents.” The Express Tribune, The Express Tribune, 26 Dec. 2017,

Available: www.tribune.com.pk/story/1584612/4-eight-reasons-motorcyclists-in-pakistan-are-more-prone-to-road-accidents/.

[Accessed May 5, 2018]

[4] Dogar, Arshad, and Sidrah Roghay. “Disturbing Data on Motorcycle Accidents.” TNS - The News on Sunday, 21 Dec. 2016,

Available: tns.thenews.com.pk/disturbing-data-motorcycle-accidents-in-lahore-and-karachi/.

[Accessed May 5, 2018]

[5] Mohid Khairul Afiq Rohd Rasli, Nina Korlina Madzi, Juliana Johari, 2013, *“Smart Helmet with sensors for accident Prevention”*

[Accessed June 12, 2018]

[6] Sayan Tapadar, Arnab Kumar Saha, Dr. Himadri Nath Saha, 2018, "*Accident and Alcohol Detection in Bluetooth enabled Smart Helmets for Motorbikes"*

[Accessed June 12, 2018]

[7] Sreenithy Chandran, Sneha Chandrasekar, Edina Lizbeth, 2016, "*Konnect: An IOT based smart helmet for Accident detection and Notification"*

[Accessed June 15, 2018]

[8] Elie Nasr, Elie Ffoury, David Khoury, 2016, *"An IOT Approach to Vehicle Accident Detection, Reporting and Navigation"*

[Accessed June 15, 2018]

[9] Akriti Singhakl, Sarishma, Ravi Tomar, 2016, *"Intelligent Accident Management System using IOT and Cloud Computing"*

[Accessed June 15, 2018]

[10] Skully Helmet, (2018), [image].

Available: https://www.skully.com/shop

[Accessed July 5, 2018]

[11] Raspberry Pi Zero W, (2018), [image].

Available:https://www.raspberrypi.org/products/raspberry-pi-zero-w/

[Accessed July 5, 2018]

[12] AdaFruit Learning Systems, Lady Ada, *"Introducing the Raspberry Pi Zero"*, datasheet, (2018).

Available: https://cdn-learn.adafruit.com/downloads/pdf/introducing-the-raspberry-pi-zero.pdf

[Accessed July 5, 2018]

[13] InvenSense Inc. [US], "*MPU-6000 and MPU-6050 Product Specification Revision 3.4*", datasheet, (2018).

Available: https://store.invensense.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf

[Accessed Aug. 20, 2018]

[14] Raspberry Pi Zero, (2019), [image]

Available: https://www.wired.co.uk/article/raspberry-pi-zero-starter-guide

[Accessed May 2, 2019]

[15] Android Architecture, (2019), [image].

Available: https://www.dev2qa.com/android-architecture-components-introduction/

[Accessed May 2, 2019]

[16] Navigation System, (2019), [image].

Available: https://www.vectorstock.com/royalty-free-vector/navigation-map-with-red-pin-vector-1847055

[Accessed May 2, 2019]

[17] Pakistan Bureau of Statistics, *"Pakistan Statistical Year Book 2016",* Appendix E.

Available:http://www.pbs.gov.pk/sites/default/files//PAKISTAN%20STATISTICAL%20YEAR%20BOOK%2C%202016.pdf

[Accessed May 1, 2018]