# CHAPTER 1

# INTRODUCTION

Recent technologies have paved the way to the objective evaluation of professional skills. The assessment of skills is mandatory and preferable in areas such as security, medicine and other professions where physical and mental skills are crucial.

There is remarkable trend in computer science and Internet of Things towards the development of systems. The purpose of such a system is to provide personalized feedback and actions to their users or to acquire the ability to respond in a more flexible way.

## 1.1MOTIVATON

A contribution towards the Internet of Things in developing smart, unbiased and convenient technologies. Rising interest in monitoring and assessing motorist's driving style, which defines driving skills and behind-wheel behavior. Observation of inconvenience and corruption in applying for RTO registered Driving License.

Driving style has a huge impact on energy consumption of vehicles. Improvements in the driving style can enable savings of up to 30% of total energy expenses. An eco-driving system can be developed to identify the factors that affect energy consumption.

## 1.2 OBJECTIVE

In today's contemporary world, recent innovations in technology have laid the foundation for comprehensive assessment of professional skills. However, such assessments are conducted by authorities of the respective fields who are intuitive and are able to review only a limited number of details and assessment styles. These abstract addresses the issue of unbiased assessment of various driving styles which are independent of circumstances. The proposed assessment of driving style is based on various indicators that are related to some parameters like vehicle's speed, jerk, acceleration, driving time and also engine rotational speed.

The Internet of Things concept is used for design of the present solution. The proposed system will be experimented on by a group of drivers with different driving styles. The overall nature of driving performance allows us to make a conclusion about the driving style. The monitoring is concluded as the continuous efforts to collect and analyze information about the values of indicative parameters of the vehicle.

# CHAPTER 2

# LITERATURE SURVEY

**[1]** The proposed objective assessment of driving style is based on eight indicators, which are associated with the vehicle's speed, acceleration, jerk, engine rotational speed and driving time. These indicators are used to estimate three driving style criteria: safety, economy and comfort. The presented solution is based on the embedded system designed according to the Internet of Things concept.

**[2]** Assessment of driving style based on two analysis: (i) the evaluation of self-assessment of the driving style, (ii) the prediction of aggressive driving style based on drivers' activity and environment parameters.

**[3]** This paper provides an accurate classification of a person's driving style and identification of driving maneuvers.

**[4]** This paper provides two approaches for driving behavior supervision based on the vehicle velocity signal which is acquired from GPS.

**[5]** This paper presents a method and system that allows providing a quantitative description of the driving style and illustrating it to the driver by means of real-time visual feedback.

**[6]** This algorithm helps to characterize the type of road on which the vehicle is moving, as well as the degree of aggressiveness of each driver.

**[7]** 6 out of 10 get driving license without test in India.

**[8]** 997 RTOs issue 1.15 crore driving licenses per year.

# CHAPTER 3

# HARDWARE DESCRIPTION

## 3.1 ACCELEROMETER

Acceleration is the measurement of the change in velocity, or speed divided by time. The application of accelerometers extends to multiple disciplines, both academic and consumer driven. Accelerometers can be used to measure vehicle acceleration, to measure vibration on cars, machines, buildings, process control systems and safety installations. They can also be used to measure seismic activity, inclination, machine vibration, dynamic distance and speed with or without the influence of gravity.

### 3.1.1 ADXL345 DIGITAL ACCELEROMETER

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g as shown in fig 3.1.1. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface. The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

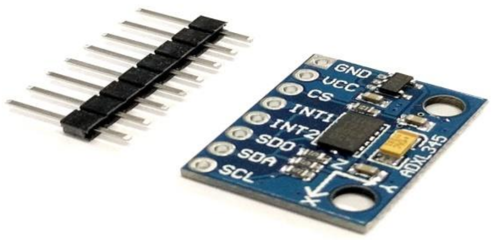


Figure 3.1.1: ADXL345

### 3.1.2 OPERATION OF ADXL345

The ADXL345 is a complete 3-axis acceleration measurement system with a selectable measurement range of ±2 g, ±4 g, ±8 g, or ±16 g. It measures both dynamic accelerations resulting from motion or shock and static acceleration, such as gravity, that allows the device to be used as a tilt sensor. The sensor is a polysilicon surface-micromachined structure built on top of a silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against forces due to applied acceleration. Deflection of the structure is measured using differential capacitors that consist of independent fixed plates and plates attached to the moving mass. Acceleration deflects the proof mass and unbalances the differential capacitor, resulting in a sensor output whose amplitude is proportional to acceleration. Phase-sensitive demodulation is used to determine the magnitude and polarity of the acceleration.

## 3.1.3 PIN CONFIGURATION AND FUNCTION DESCRIPTION

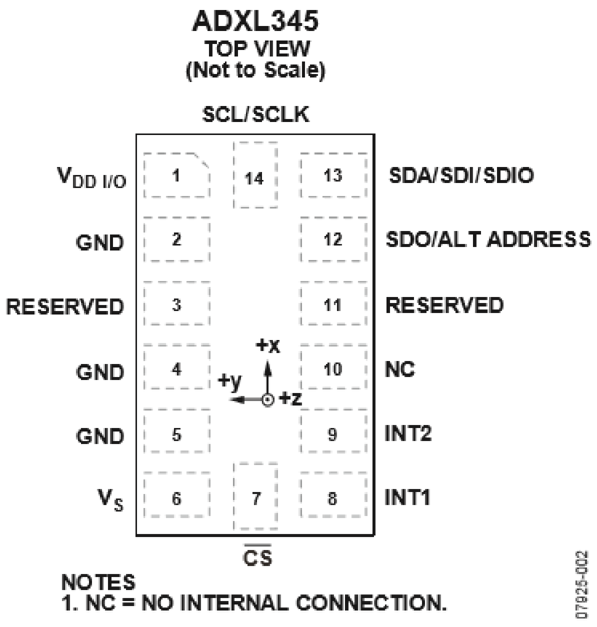


Figure 3.1.3: Pin configuration

Table 3.1: Function description

| Pin No. | Pin name | Description |
|---|---|---|
| 1. | VDD I/O | Digital interface supply voltage. |

| | | |
|---|---|---|
| 2. | GND | This pin must be connected to the ground. |
| 3. | RESERVED | This pin must be connected to Vs or left open. |
| 4. | GND | This pin must be connected to the ground. |
| 5. | GND | This pin must be connected to the ground. |
| 6. | Vs | Supply voltage. |
| 7. | CS | Chip selects. |
| 8. | INT1 | Interrupt 1 output. |
| 9. | INT2 | Interrupt 2 output. |
| 10. | NC | Not internally connected. |
| 11. | RESERVED | This pin must be connected to the ground or left open. |
| 12. | SDO/ALT ADDRESS | Serial data output (SPI 4-wire)/alternate I2C address select(I2C). |
| 13. | SDA/SDI/SDIO | Serial data(I2C)/serial data input (SPI 4-wire)/serial data input and output (SPI 3-wire). |
| 14. | SCL/SCLK | Serial communications clock. SCL is the clock for I2C, and SCLK is the clock for SPI. |

## 3.2 RASPBERRY PI ZERO W

Raspberry Pi Zero is an ultra-low cost and ultra small variant of the original Raspberry Pi. It's tiny, measuring just 65mm x 30mm (fig 3.2), and is perfectly designed for embedded applications, wearables, prototyping and any other Pi based tinkering. The Raspberry Pi Zero Wireless features on board Wireless Internet & Bluetooth for all the connectivity needs. The Raspberry Pi Zero features a BCM2835 chipset, over clocked to 1Ghz with 512MB RAM, and the same 1080p video output. It also features the same 40 pin GPIO layout as the Raspberry Pi 2/B+/A+. With it's small form factor and reduction in connectors, the Raspberry Pi Zero only uses ~ 140mA at 5V.

Raspberry Pi Zero has an unpopulated GPIO, unpopulated composite (RCA) header and an unpopulated reset header.

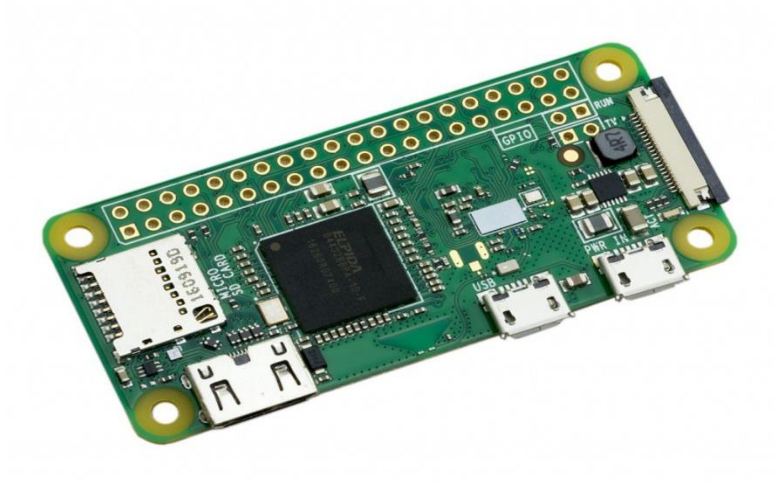


Figure 3.2: Raspberry pi zero W

### 3.2.1 HARDWARE OVERVIEW

- In built Wi-Fi and Bluetooth.
- Uses mini-HDMI connector
- To save space, the Zero has a USB On-the-Go (OTG) connection. The Pi Zero uses the same Broadcom IC that powered the original Raspberry Pi A and A+ models. This IC connects directly to the USB port allowing for OTG functionality, unlike the Pi B, B+, 2 and 3 models, which use an onboard USB hub to allow for multiple USB connections.
- Power is provided through a microUSB connector. The voltage supplied to the power USB should be in the range of 5-5.25V.
- Another familiar interface is the microSD card slot. Insert microSD cards that contain Raspberry Pi image file here.
- Zero W offers both 802.11n wireless LAN and Bluetooth 4.0 connectivity.

- The Raspberry Pi Zero V1.3+ and all Zero Ws have an onboard camera connector. This can be used to attach the Raspberry Pi Camera module.

## 3.3 BLUETOOTH MODULE

The Bluetooth module HC-05 is a MASTER/SLAVE module as shown in fig 3.3. By default, the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to your embedded project, etc.

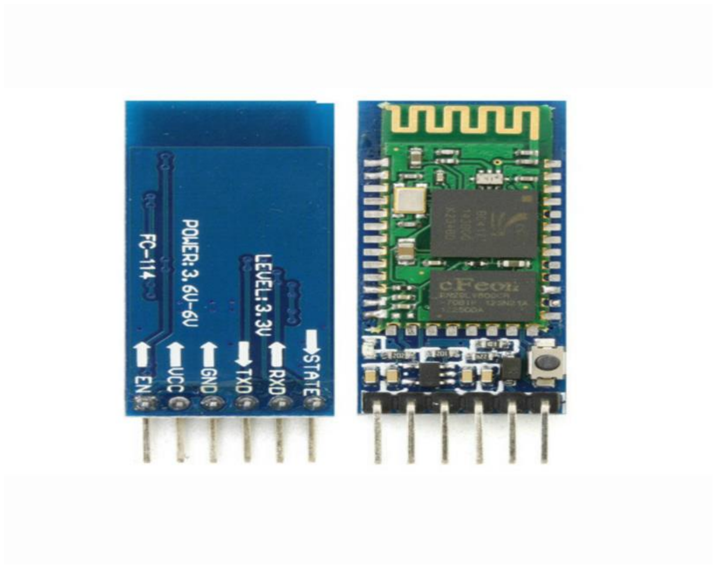


Figure 3.3: HC-05 Bluetooth module

**Hardware Features**

- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- 3.3 to 5 V I/O.

- PIO (Programmable Input/Output) control.
- UART interface with programmable baud rate.
- With integrated antenna.
- With edge connector.

**Software Features**

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1, Parity:No parity.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"1234" as default.

# CHAPTER 4
# SOFTWARE DESCRIPTION

## 4.1 RASPBIAN

Raspbian is the main and basic software for RPi devices, officially supported by the Raspberry Pi Foundation. It is an operating system, based on Debian and optimized for Raspberry Pi hardware. It comes with pre-installed pieces of software appropriate for most ARM users and developers.

Wheezy, Jessie and Stretch are the three most popular versions of the Raspbian software. The version being used in the project is Raspbian Stretch and the UI interface is shown in fig 4.1.

### 4.1.1 RASPBIAN STRETCH

The Raspberry Pi official website provides the free current latest version of the Raspbian software in image format. Once opened the website provides two archives: Raspbian and Raspbian Lite, which means the full package of Raspbian and limited one. The difference between the two is that the Lite version doesn't have GUI and X-server and is made to fulfill very specific tasks [9].

When compared with the previous versions or Raspbian, Stretch stands out with its internal technical developments and offers:

- Updated applications (like Sonic Pi or Chromium)
- More advanced approach to audio over Bluetooth
- Usernames handling has been improved. Automatic logging in with the name "pi" has been replaced with the password name of the current user
- Scratch 2 (programming language) improvement. It is possible to use the Sense HAT with Scratch 2

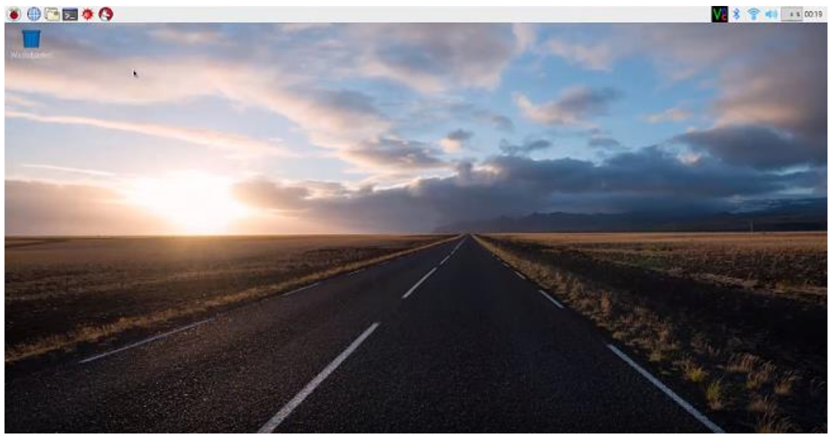
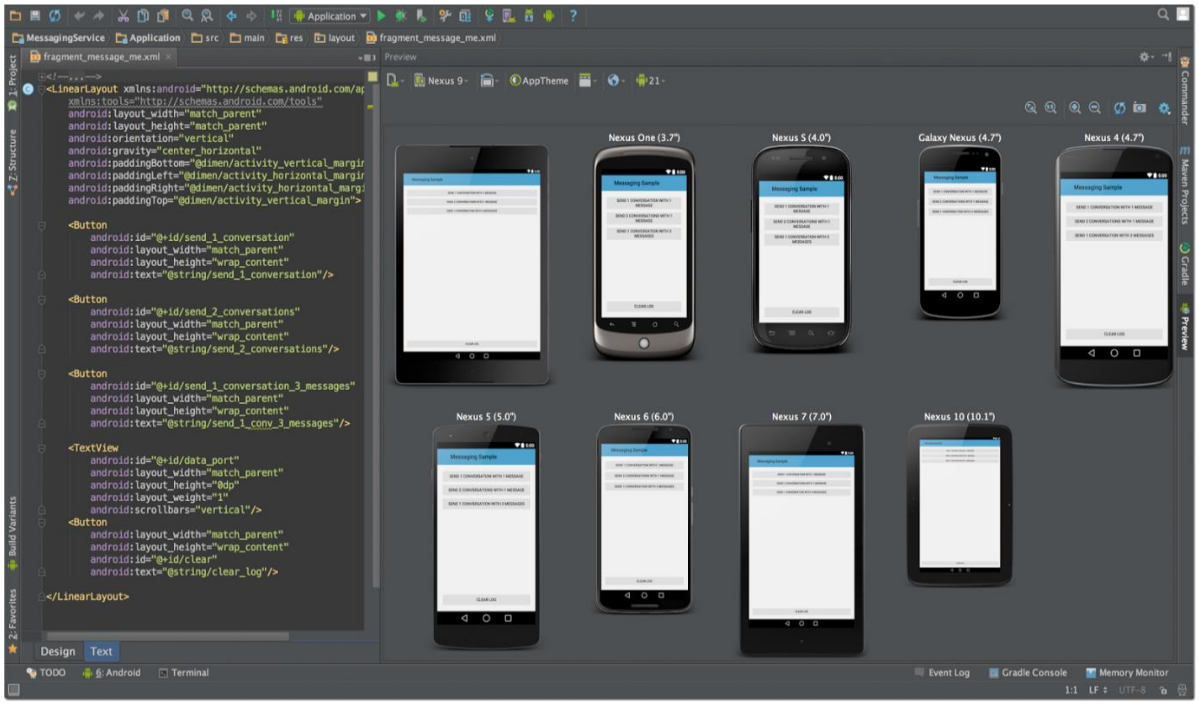


Figure 4.1: Raspbian Stretch UI

## 4.2 ANDROID STUDIO

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. The layout of an Android studio UI is shown in fig 4.2.



```
<!--...-->
<LinearLayout xmlns:android="http://schemas.android.com/ap
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center_horizontal"
    android:paddingBottom="@dimen/activity_vertical_margin
    android:paddingLeft="@dimen/activity_horizontal_margin
    android:paddingRight="@dimen/activity_horizontal_margi
    android:paddingTop="@dimen/activity_vertical_margin">

    <Button
        android:id="@+id/send_1_conversation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/send_1_conversation"/>

    <Button
        android:id="@+id/send_2_conversations"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/send_2_conversations"/>

    <Button
        android:id="@+id/send_1_conversation_3_messages"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/send_1_conv_3_messages"/>

    <TextView
        android:id="@+id/data_port"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:scrollbars="vertical"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/clear"
        android:text="@string/clear_log"/>

</LinearLayout>
```

Figure 4.2: Android Studio version 3.0 UI

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- Rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps

- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ e.g. Java, C++

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.

Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

The software was first announced at Google I/O in May 2013, and the first stable build was released in December 2014. Android Studio is available for Mac, Windows, and Linux desktop platforms. It replaced Eclipse Android Development Tools (ADT) as the primary IDE for Android application development. Android Studio and the Software Development Kit can be downloaded directly from Google.

- **Application**, an Android application can have one Application class instance, instantiated as soon as the applications starts (and the last to stop when shutting down).
- **Activity**, the visual representation of an application; there can be several. Activities are the basis for the user interface.
- **Context** connects to the Android system executing the application.

## 4.3 APP COMPONENTS

App components are the essential building blocks of an Android app. Each component is an entry point through which the system or a user can enter the app. Some components depend on others.

There are four different types of app components:

- Activities

- Services
- Broadcast receivers
- Content providers

Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed. The following sections describe the four types of app components.

### 4.3.1 ACTIVITIES

An activity is the entry point for interacting with the user. It represents a single screen with a user interface. For example, an email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email app, each one is independent of the others. As such, a different app can start any one of these activities if the email app allows it. For example, a camera app can start an activity in an email app that composes new mail to allow the user to share a picture. An activity facilitates the following key interactions between system and app:

- Keeping track of what the user currently cares about (what is on screen) to ensure that the system keeps running the process that is hosting the activity.
- Knowing that previously used processes contain things the user may return to (stopped activities), and thus more highly prioritize keeping those processes around.
- Helping the app handle having its process killed so the user can return to activities with their previous state restored.
- Providing a way for apps to implement user flows between each other, and for the system to coordinate these flows. (The most classic example here being share.)

An activity is implemented as a subclass of the Activity class.

### 4.3.2 SERVICES

A service is a general-purpose entry point for keeping an app running in the background for all kinds of reasons. It is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different app, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it to interact with it. There are two very distinct semantics services that tell the system about how to manage an app: Started services tell the system to keep them running until their work is completed. This could be to sync some data in the background or play music even after the user leaves the app. Syncing data in the background or playing music also represent two different types of started services that modify how the system handles them:

- Music playback is something the user is directly aware of, so the app tells the system this by saying it wants to be foreground with a notification to tell the user about it; in this case the system knows that it should try really hard to keep that service's process running, because the user will be unhappy if it goes away.
- A regular background service is not something the user is directly aware of as running, so the system has more freedom in managing its process. It may allow it to be killed (and then restarting the service sometime later) if it needs RAM for things that are of more immediate concern to the user.

A service is implemented as a subclass of Service.

### 4.3.3 BROADCAST RECEIVERS

A broadcast receiver is a component that enables the system to deliver events to the app outside of a regular user flow, allowing the app to respond to system-wide broadcast announcements. Because broadcast receivers are another well-defined entry into the app, the system can deliver broadcasts even to apps that aren't currently running. So, for example, an app can schedule an alarm to post a notification to tell the user about an upcoming event... and by delivering that alarm to a Broadcast Receiver of the app, there is no need for the app to remain running until the alarm goes off. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Apps can also initiate broadcasts—for example, to let other apps know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a gateway to other components and is intended to do a very minimal amount of work. For instance, it might schedule a JobService to perform some work based on the event with JobScheduler

A broadcast receiver is implemented as a subclass of, BroadcastReceiver and each broadcast is delivered as an Intent object.

### 4.3.4 CONTENT PROVIDERS

A content provider manages a shared set of app data that you can store in the file system, in an SQLite database, on the web, or on any other persistent storage location that your app can access. Through the content provider, other apps can query or modify the data if the content provider allows it. For example, the Android system provides a content provider that manages the user's contact information. As such, any app with the proper permissions can query the content provider, such as ContactsContract.Data, to read and write information about a particular person. It is tempting to think of a content provider as an abstraction on a database, because there is a lot of API and support built into them for that common case. However, they have a different core purpose from a system-design perspective. To the system, a content provider is an entry point into an app for publishing named data items, identified by a URI scheme. Thus, an app can decide how it wants to map the data it contains to a URI namespace, handing out those URIs to other entities which can in turn use them to access the data.

A content provider is implemented as a subclass of ContentProvider and must implement a standard set of APIs that enable other apps to perform transactions. A unique aspect of the Android system design is that any app can start another app’s component. For example, if the user wants to capture a photo with the device camera, there's probably another app that does that and your app can use it instead of developing an activity to capture a photo yourself. There is no need to incorporate or even link to the code from the camera app. Instead, we can simply start the activity in the camera app that captures a photo. When complete, the photo is even returned to your app so you can use it. To the user, it seems as if the camera is actually a part of your app.

When the system starts a component, it starts the process for that app if it's not already running and instantiates the classes needed for the component. For example, if your app starts the activity in the camera app that captures a photo, that activity runs in the process that belongs to the camera app, not in your app's process. Therefore, unlike apps on most other systems, Android apps don't have a single-entry point (there is no main () function).

## Creating a button:

A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.

Depending on whether we want a button with text, an icon, or both, we can create the button in your layout in three ways:

With text, using the Button class:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    ... />
```

With an icon, using the ImageButton class:

```
<ImageButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/button_icon"
    ... />
```

With text and an icon, using the Button class with the android:drawableLeft attribute:

```
<Button
    android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content"
android:text="@string/button_text"
android:drawableLeft="@drawable/button_icon"
... />
```

When the user clicks a button, the Button object receives an on-click event.To define the click event handler for a button, add the android:onClick attribute to the `<Button>` element in your XML layout. The value for this attribute must be the name of the method you want to call in response to a click event. The Activity hosting the layout must then implement the corresponding method.

### Styling a button:

The appearance of the button (background image and font) may vary from one device to another, because devices by different manufacturers often have different default styles for input controls.

To customize individual buttons with a different background, we need to specify the android:background attribute with a drawable or color resource. Alternatively, you can apply a style for the button, which works in a manner similar to HTML styles to define multiple style properties such as the background, font, size, and others.

# CHAPTER 5

# METHODOLOGY

## 5.1 ACCELEROMETER SECTION

Different driving maneuvers are found and differentiated by using each individual axis of the accelerometer. The entries in table 5.1 represent the significance of each axis with respect to deciding the driving style of the user.

Table 5.1: Significance of tri-axial measurements:

| Axis | Direction | Typical Driving style |
|---|---|---|
| X-axis | Front | Acceleration |
| Y-axis | Left/Right | Turning |
| Z-axis | Up/Down | Vibrations/Road Anomalies |

The accelerometer module is connected to the raspberry Pi module through the GPIO pins. The real -time values from the accelerometer are received into the raspberry and are transferred in serial format into the Bluetooth module for transfer. Fig 5.1(a), (b), (c) shows the graphical representation of the varying accelerometer positions along X, Y, Z with respect to both positive and negative axis.

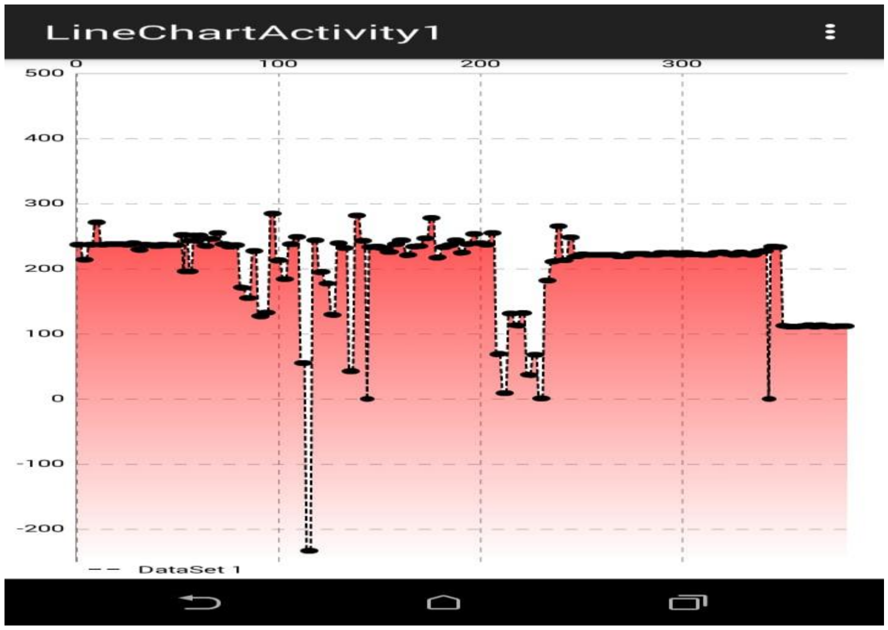


Figure 5.1(a): Graphical representation of X-axis accelerations

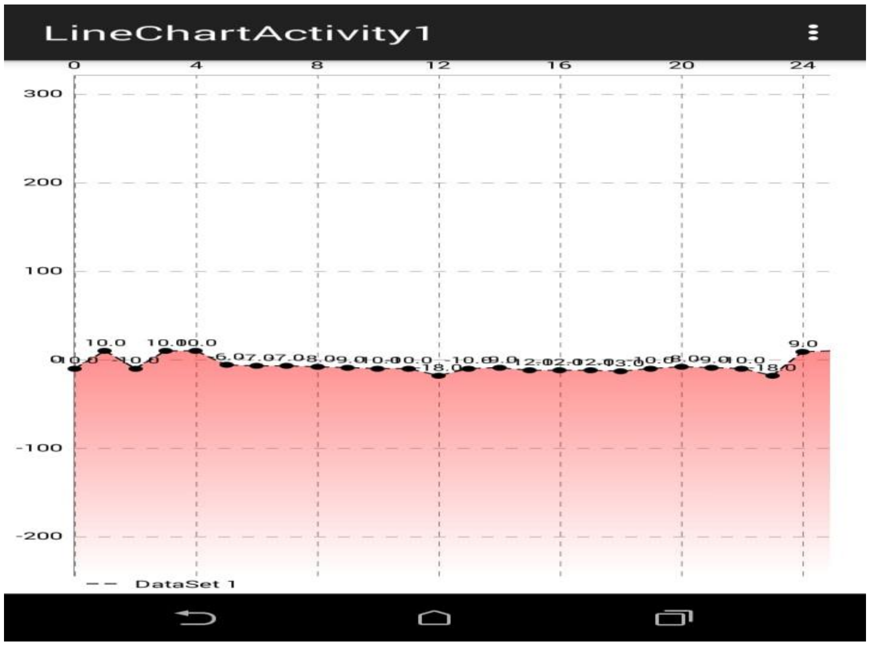


Figure 5.1(b): Graphical representation of Y-axis accelerations

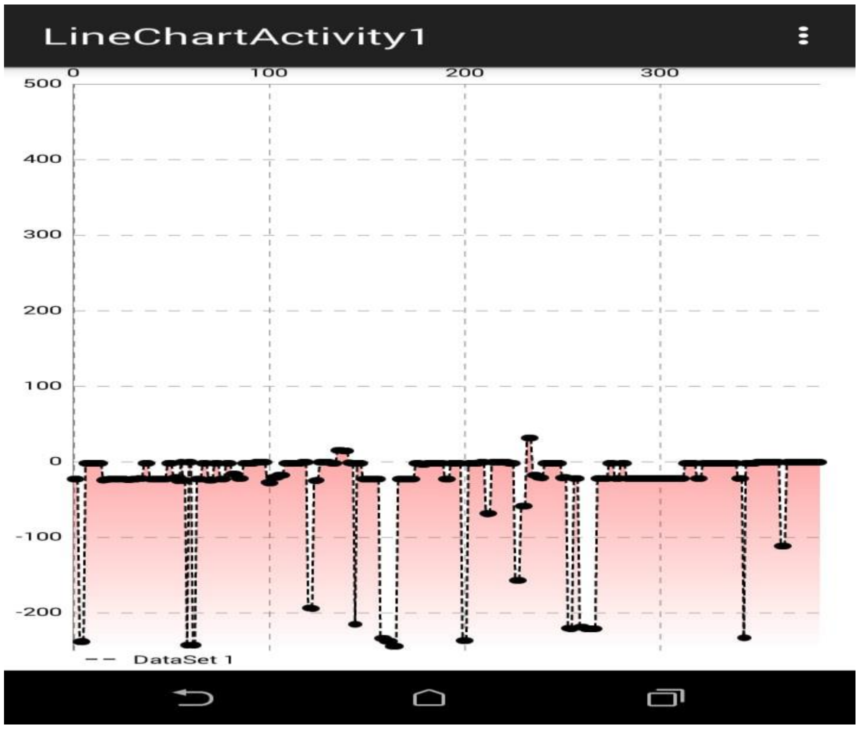


Figure 5.1(c) : Graphical representation of Z-axis accelerations

It is evident that if the input accelerations cross a certain level, then the driving style of the user is considered as non-ideal or rash. The decision for the same is made at the application level where a threshold for each direction of the individual accelerations is set and any accelerometer value crossing that threshold is considered accountable.

## 5.2 APPLICATION: 'DRIVER STYLE'

**Phase 1: Face Recognition**

For authentication purposes the application has been interfaced with a third-party face recognition system which initially takes the training image of the user as the input and stores it. When the 'Driver style' application is opened the front camera of the android phone is initiated to read the image of the user to decide if the user is valid for the application usage as shown in fig 5.2(a).

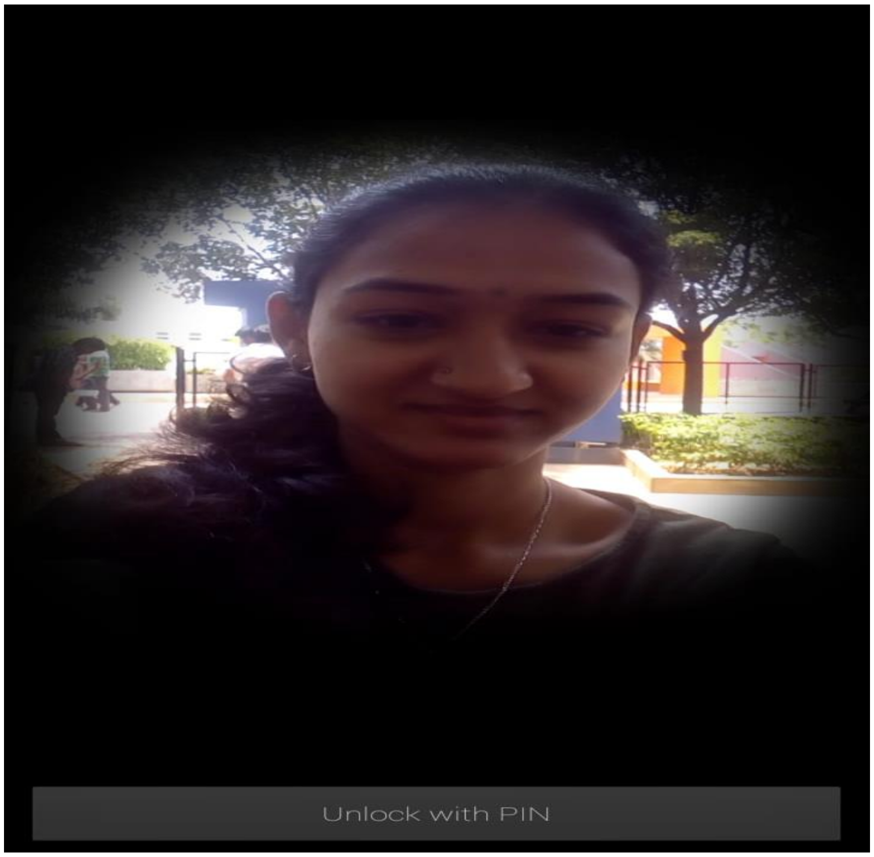


Figure 5.2(a): Face recognition

**Phase 2: Bluetooth initiation**

When the face features of the user have been recognized successfully the first UI window of the application as in fig 5.2(b) is observed. Before any further steps the android phone under use should be paired with the Bluetooth module which is named as 'HC-05' in this case. In the first UI interface window two buttons can be observed 'OPEN' and 'CLOSE'. On pressing the OPEN button communication with the Bluetooth module of the hardware is initiated and the application starts receiving real-time accelerometer values. Accordingly, the accelerometer values are recorded in the form of a plot, where each plot button represents the graphical reading of each of the accelerometer axis.

**Phase 3: Accelerometer threshold**

The next phase is that of an option to enter the threshold values of both positive and negative directions of all the three axes (X, Y, Z). If the accelerometer value exceeds the threshold that has been entered, then the respective count at the bottom of the window is updated.

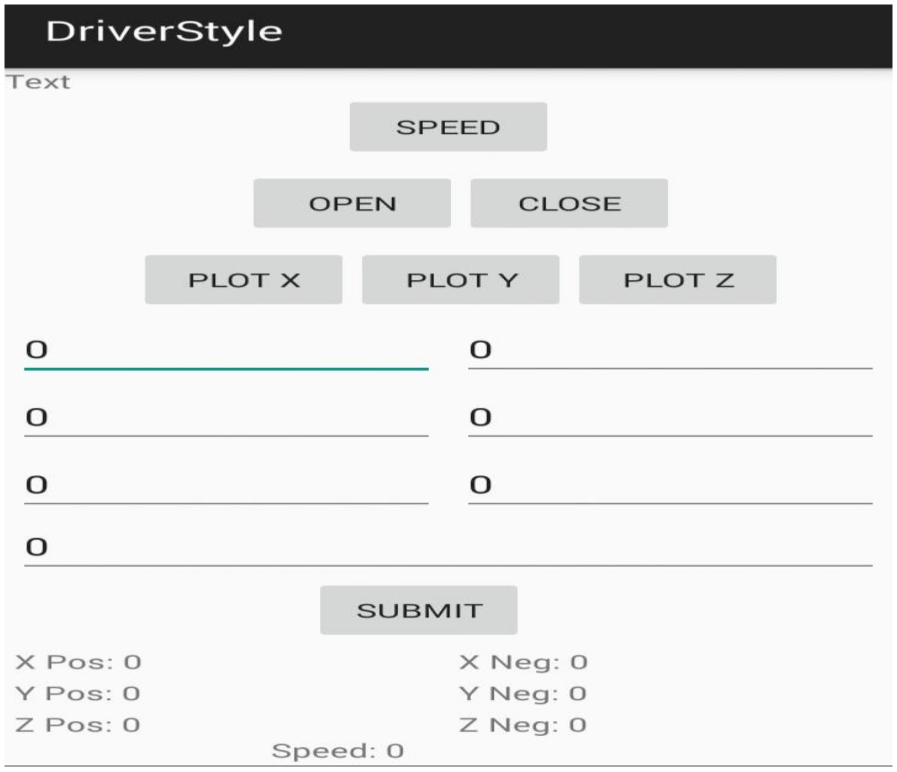


Figure 5.2(b): First UI window of the application

**<u>Phase 4: Speed calculation</u>**

A closer look into fig 5.2(b) shows a 'SPEED' button. This button is meant for the purpose of extracting the mobile phone's GPS location and calculating the speed at which the position of the phone is changing which is nothing but the speed of the vehicle in which the phone is placed.

$$\text{Speed} = \frac{\text{change in successive position of the android phone}}{\text{Time}}$$

Successive positions are obtained using the latitude and longitude co-ordinates of the GPS in the mobile. Also, experimentation and study show that the GPS location gives more accurate values if the mobile is placed at the centre of the interior in a car [10].

Like the acceleration threshold values a threshold value can be set for the speed parameter after crossing which the count for speed will increase indicating a behavior of over speeding.

On pressing the ‘SPEED’ button a blank screen will open and will display the speed at which the position of the car is changing. Once the screen is closed the count for over speeding is updated.

## 5.3 FLOW DIAGRAM

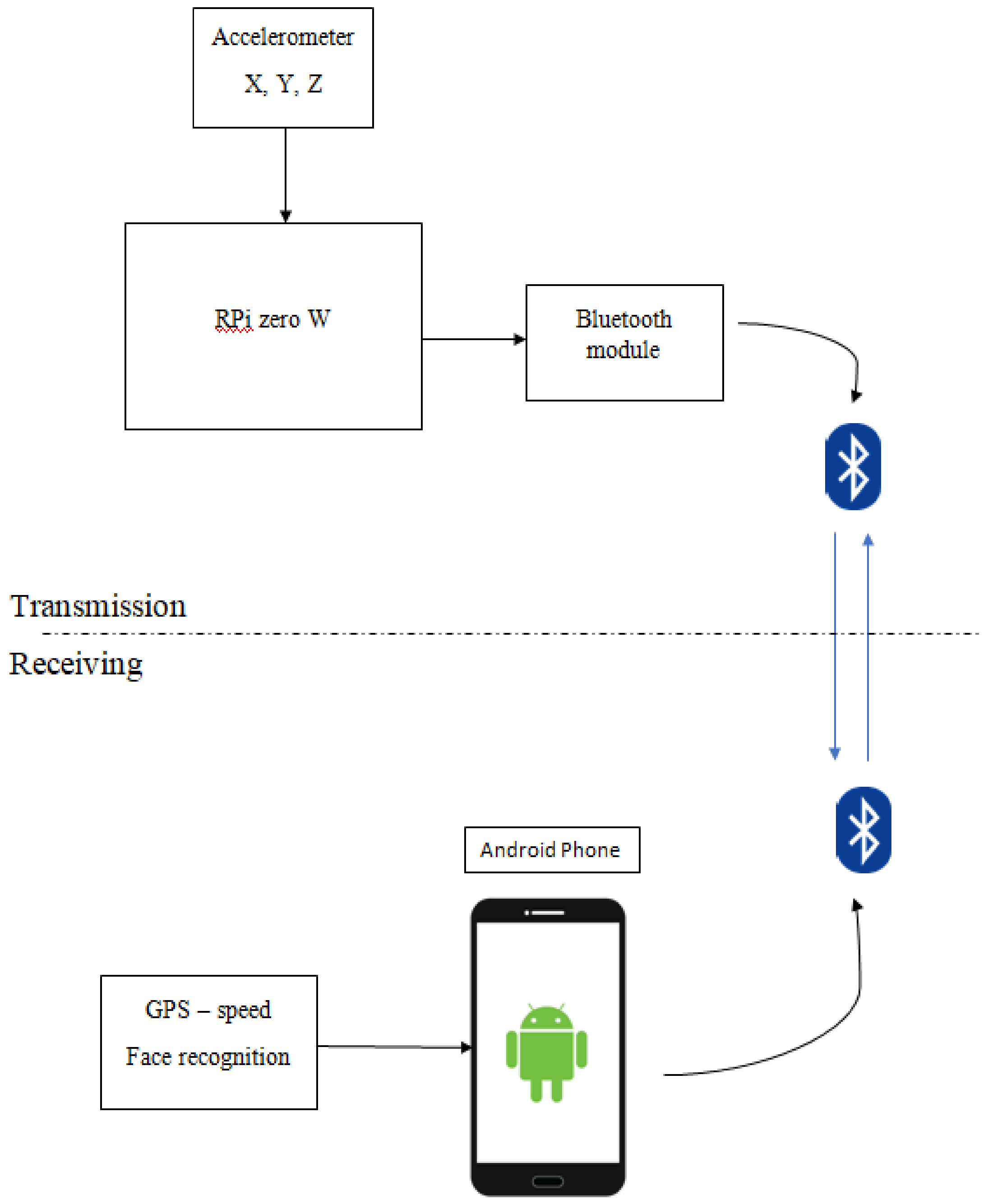


Figure 5.3: Flow diagram of operation

## 5.4 IMPLEMENTATION [USER INTERFACE]

Step 1: The Bluetooth of the android phone used for determining the driving style is paired with that of the hardware Bluetooth module. If not paired prior, on opening the application, the application will ask for Bluetooth connection access by displaying a prompt message.

Step 2: Before opening the main application, it is necessary for the user to set his face image for face recognition purposes in the third-party application that is used for authentication purposes. The input to the third-party application is a set of training images of the user which will be further used as an access grant when our application is opened.

Step 3: On opening our application, the first UI window that is seen is shown in fig 5.2(b).

Significance of each button in the UI

- SPEED: Opens a window where the speed of the mobile moving, which is further than the speed of the car, is displayed in real time. For the application to display speed this window needs to be open until the driving test is completed. On exiting the screen, the over speeding count is updated as per the threshold that was entered respectively.
- OPEN: Initiates Bluetooth connection for receiving accelerometer values from the RPi.
- CLOSE: Cuts down the Bluetooth communication indicating an end to the driving test
- PLOT X, PLOT Y, PLOT Z: Provides the graphical representation of the accelerometer axes values that are obtained through the Bluetooth and RPi interface

Below the buttons are the option for entering the threshold values for positive/ negative X, Y, Z axis and speed. The SUBMIT button takes these values as input and the application from here on will start considering the newly entered threshold values for considerations. As we can see in fig 5.2(b), the bottom most part of the interface gives out count values for exceeding readings of positive/ negative X, Y, Z axis and speed. These scores can be directly made use by the RTO [Regional Transport Office] for determining the driving style of the applicant and issue of Driving License accordingly.

Step 4: To erase the data of the previous user and to take new values from the beginning the flow to be followed is- Settings→ Apps→Driver Style→ Clear data

The same can be applied for feeding new face recognition training images for the third-party application that is being used by our application.

# APPLICATIONS

- **RTO administered driving license issue**

  A driving license is an official document that authorizes its holder to operate various types of motor vehicles on highways and other roads to which the public have access. This document was provided by the RTO (Regional transport office) based on a real-time driving style test. The system that we propose can act as an unbiased alternative for the existing system of driving style test.

- **Insurance and car rental companies**

  Supervised information about the driving style of a client can be accessed by insurance and car rental companies through this assessment in providing insurance and some special discounts for those driving economically and safely.

- **Registered driving schools**

  With an efficient analysis program recognized by RTO, driving schools can get information on trainees to decide if they are ready for examination or need more practice.

# 6. RESULTS

## 6.1 HARDWARE SECTION

The resulting prototype after setting up the device onto the front end of the car's interior can be seen in fig 6.1. The raspberry pi only needs a power supply through an USB cable. The set up is in such a way that as soon as the RPi is booted after plugging in the power supply it starts running the program in an infinite loop, hence tending to continuous Bluetooth serial communication.

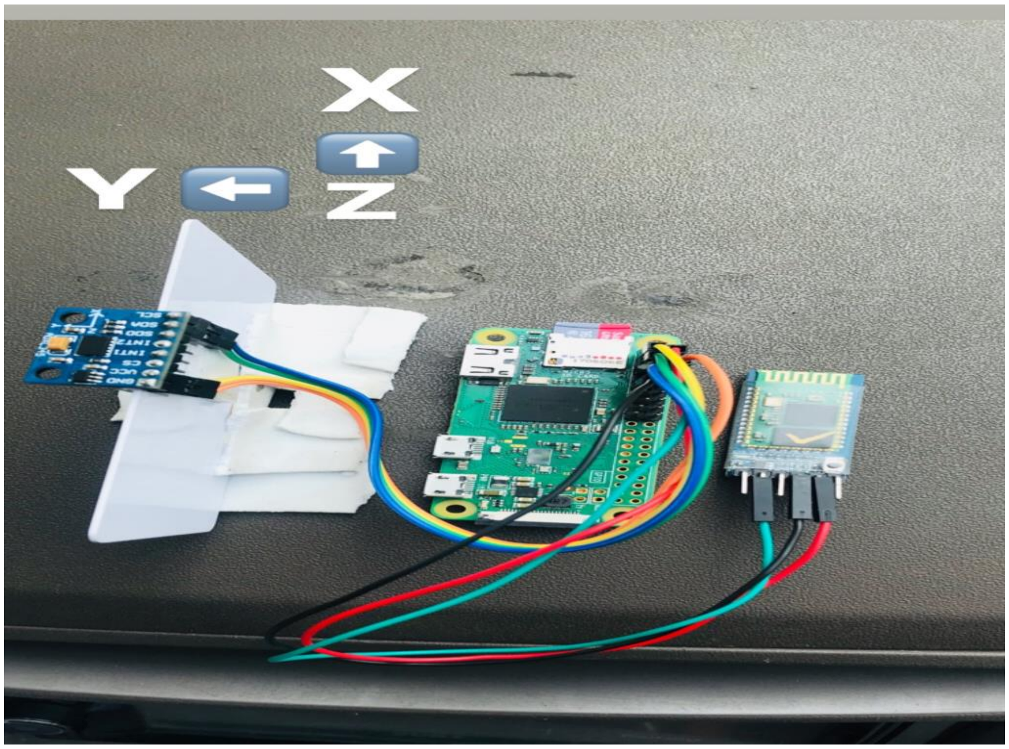


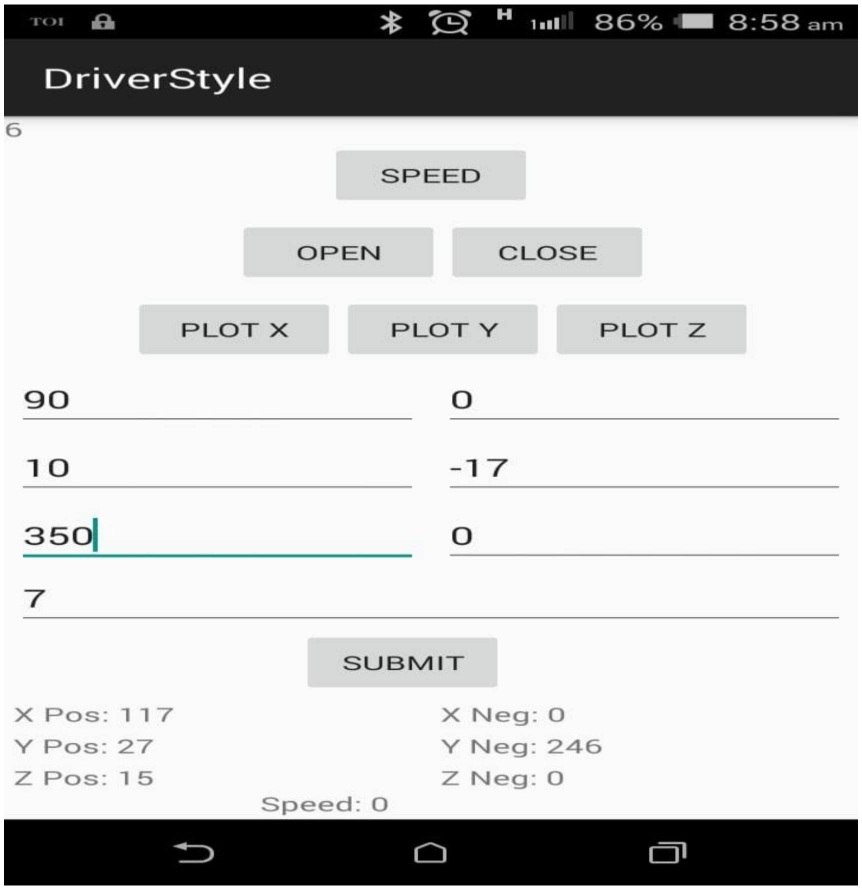
Figure 6.1: Hardware section

The prototype is an easy and convenient set up when compared to the previously existing model [1]. It is user friendly in terms of connections and maintenance, since the only hardware connection required is that of an USB cable as power supply for the RPi. The use of a Bluetooth module adds in more convenience because of the availability. The hardware is also affordable in terms of its cost, where it might cost up to a maximum of Rs.1500 including the cost of the RPi mother board for processing.

The prototype can be used at various locations and unlike conventional methods it does not need the requirement of a human personnel to examine the driving style from outside a car.

The fig 6.1 also shows the axis calibration in which the device needs to be pleased in order to follow the significance of changes in each accelerometer axis.

## 6.2 APPLICATION SECTION



90 0

10 -17

350 0

7

X Pos: 117 X Neg: 0

Y Pos: 27 Y Neg: 246

Z Pos: 15 Z Neg: 0

Speed: 0

Figure 6.2: Final look of application

The final look of the application UI is shown in fig 6.2 along with final graph plots for the X, Y, Z axis accelerations. The bottom most part of the UI shows the final count values as a result of the experiment.

<u>Significance of each score</u>

X pos: Sudden change in acceleration.

Y pos: Sudden cornering/change of lane towards left.

Y neg: Sudden cornering/change of lane towards right.

Z pos: Indicates a score for sudden change in the positive Z direction which is directed upwards and hence indicates abrupt driving in road humps and other road anomalies.

# CONCLUSION AND FUTURE SCOPE

The prototype of the system used for driving style assessment was validated experimentally in the real-world scenario. The system is a valid solution for determining unusual or rash driving conditions considering values from an accelerometer and GPS coordinates for speed calculation. It also makes use of a third-party face recognition application for authentication purposes. The application proves to be an efficient solution for user friendly assessment of users' driving style

The designed system could be supplemented with additional vision and noise sensors used to monitor external driving conditions such as weather and road conditions, along with using more parameters from the cars' on-board computer.

# REFERENCES

[1] Bartosz Jachimczyk , Damian Dziak, Jacek Czapla, Pawel Damps and Wlodek J. Kulesza. “IoT On-Board System for Driving Style Assessment”. Sensors 2018, 18(4), 1233-; Also available online: https://doi.org/10.3390/s18041233

[2] Mikhail Sysoev, Andrej Kos, Jože Guna and Matevž Pogacˇnik . “Estimation of the Driving Style Based on the Users’ Activity and Environment Influence”. Sensors 2017, 17(10), 2404 -; Also available online: https://doi.org/10.3390/s17102404

[3] P. Brombacher, J. Masino, M. Frey and F. Gauterin, "Driving event detection and driving style classification using artificial neural networks", *2017 IEEE International Conference on Industrial Technology (ICIT)*, Toronto, ON, 2017, pp. 997- 1002.doi:10.1109/ICIT.2017.7915497 Also available online:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7915497&isnumber=7912587

[4] O. Derbel and R. J. Landry, "Driving style assessment based on the GPS data and fuzzy inference systems," *2015 IEEE 12th International Multi-Conference on Systems, Signals & Devices (SSD15)*, Mahdia, 2015, pp.1-8.doi:10.1109/SSD.2015.7348214 Also available online:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7348214&isnumber=7348090

[5] A. Corti, C. Ongini, M. Tanelli and S. M. Savaresi, "Quantitative Driving Style Estimation for Energy-Oriented Applications in Road Vehicles," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, 2013, pp. 3710-3715.doi: 10.1109/SMC.2013.632

Also available online:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6722385&isnumber=6721750

[6] J. E. Meseguer, C. T. Calafate, J. C. Cano and P. Manzoni, "DrivingStyles: A smartphone application to assess driver behavior," *2013 IEEE Symposium on Computers and Communications (ISCC), Split*,2013, pp.000535-000540. doi:10.1109/ISCC.2013.6755001

Also available online:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6755001&isnumber=6754912

[7] “6 out of 10 get driving license without test in India” [Source-The Times of India]

[8] “997 RTOs issue 1.15 crore driving licenses per year” [Source-The Times of India]

[9] https://eltechs.com/raspbian-and-other-raspberry-pi-software/#stretch

[10] Fazeen, Mohamed &Gozick, Brandon &Dantu, Ram &Bhukhiya, Moiz& Gonzalez, Marta C... (2012). Safe Driving Using Mobile Phones. Intelligent Transportation Systems, IEEE Transactions on. 13. 1462-1468. 10.1109/TITS.2012.2187640.