Rahul Kumar Bhadani *rahulbhadani@email.arizona.edu*
Matt Watson *mattcwatson@email.arizona.edu*
Jue Zhang *ficco@email.arizona.edu*

Design Report

Team CATV

ECE 573 Spring 2016

April 10, 2016

Department of Electrical and Computer Engineering,
The University of Arizona

Executive Summary

The goal of this project is to develop an android application that will act as an Operator Control Unit for a test autonomous vehicle, CATVehicle. It will display necessary information published by the CATvehicle such as position and velocity of the vehicle, its GPS location, and any other sensors/actuators in a user friendly manner and additionally perform health monitoring functions. It will also inform the operator of any discrepancy in the data such as GPS location so that operator may take necessary precautions to avoid any accident.

Table of Contents

List of Figures

List of Tables

1

Project Overview

The CATVehicle is an autonomous vehicle that can be completely controlled via software. Because of the risks associated with autonomous vehicles, it is necessary to have multiple levels of system monitoring and error checking, one of which is a human user. In order to make it easier for a user to monitor the vehicle status, this project aims to build an android application which will perform many of these functions on a smartphone. Having this capability in a phone application will make it easy for virtually any user to easily access and monitor data regarding vehicle status.

The application will record and analyze data coming from the CATVehicle and provide appropriate warnings and errors messages as required. For example, when GPS data is stale and inaccurate, the application will notify the user that GPS data is being drifted or not matching with the GPS data being obtained by the smartphones sensor.

The application will also provide an easy way to access these data files and share them via an email client for further analysis if necessary.

2

Requirements

The Application will act as an interface to display the crucial information of the autonomous vehicle CATVehicle regarding its current state and other variables present governing the state of the vehicle.

2.1 Software and Hardware Requirements

- An android OS equipped smartphone with internet connectivity either in the form of cellular data or wifi.
- An Autonomous vehicle capable of acting as a ROS Publisher to publish messages such as headway, headway angle, positions, velocity, acceleration, heading, array of distance measurements, gas/fuel availability from actuators/sensors present in the autonomous vehicle.

2.2 Application/Functional Requirements

2.2.1 Overview

The Application will act as an interface to display the crucial information of the autonomous vehicle CATVehicle regarding its current state and other variables present governing the state of the vehicle.

2.2.2 'B' requirements

1. The Application should be able to establish a connection to a ROS publisher. Difficulty level: 1.
2. The Application should be able to receive the ROS message being published by the CATVehicles ROS publisher. Difficulty level: 2. Dependency: B1.

3. The Application should be able to correctly display a graphical representation of velocity, brake and accelerator data being received from ROS master. Difficulty level: 3. Dependency: B2.
4. The Application will display an error message if the brake and the accelerator are actuated at the same time. Difficulty level: 3. Dependency: B2.
5. The Application should be able to record and save the messages received during a session and should be able to email this data via the email client when prompted. Difficulty level: 1. Dependency: B2.

2.2.3 'A' requirements

1. The Application should be able to trigger a warning notification when it detects that the vehicle is exceeding the speed limit. Difficulty level: 3. Dependency: B requirements.
2. The Application should display an error message if the GPS data changes beyond the margin of error while the velocity is zero. Difficulty level: 3. Dependency: B requirements.
3. The Application should be able to notify user if the LIDAR detects obstruction in front of the car. Difficulty level: 4. Dependency: B requirements.
4. The Application will compare phone GPS readings with ROS GPS readings to be within a specified preference of each other and will provide an alert if this is not the case. Difficulty level: 3. Dependency: B requirements.

3

Application Analysis

3.1 Overview

This section discusses UI design, their connection with various use cases discussed in section 4.3 and state models explaining how it will interact with the user.

3.2 User Interface Design

When the app is first opened, an Entry Screen will ask for the URI of the ROS Master before proceeding. Also you can open settings menu on clicking settings icon and set various parameters.

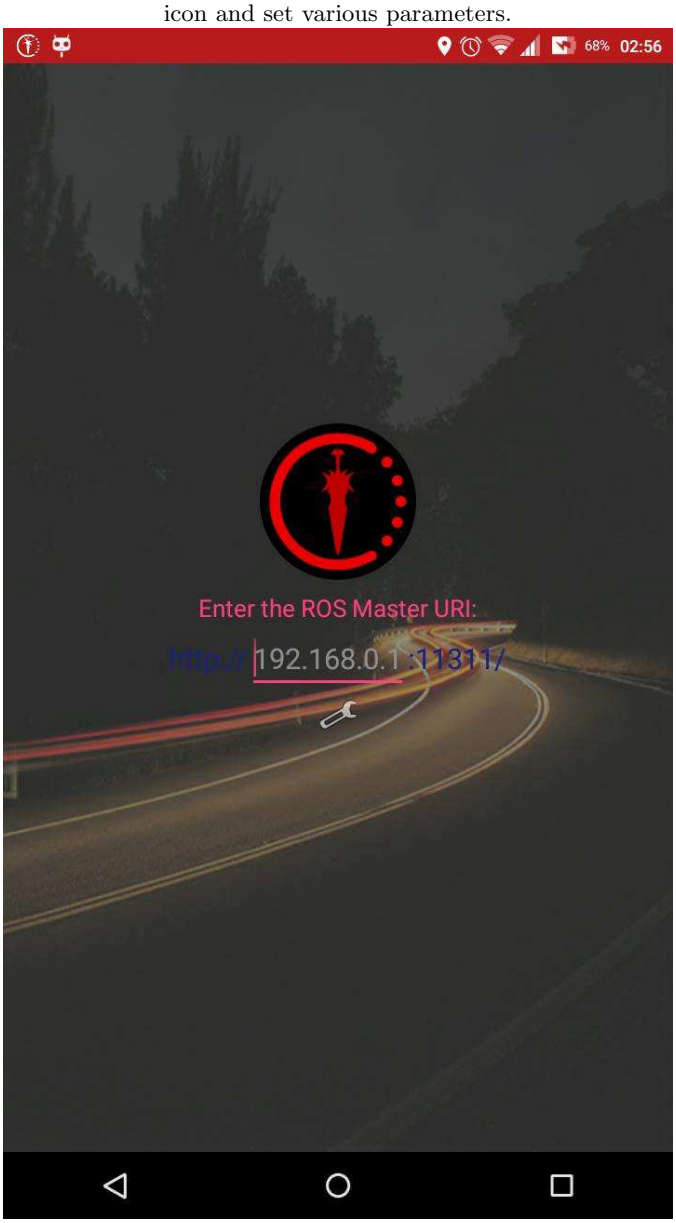

Fig. 3.1: Screen1: ROSConnectActivity - Enter the ROS Master URI

If the connection cannot be established, an error message will appear in the form of snackbar.

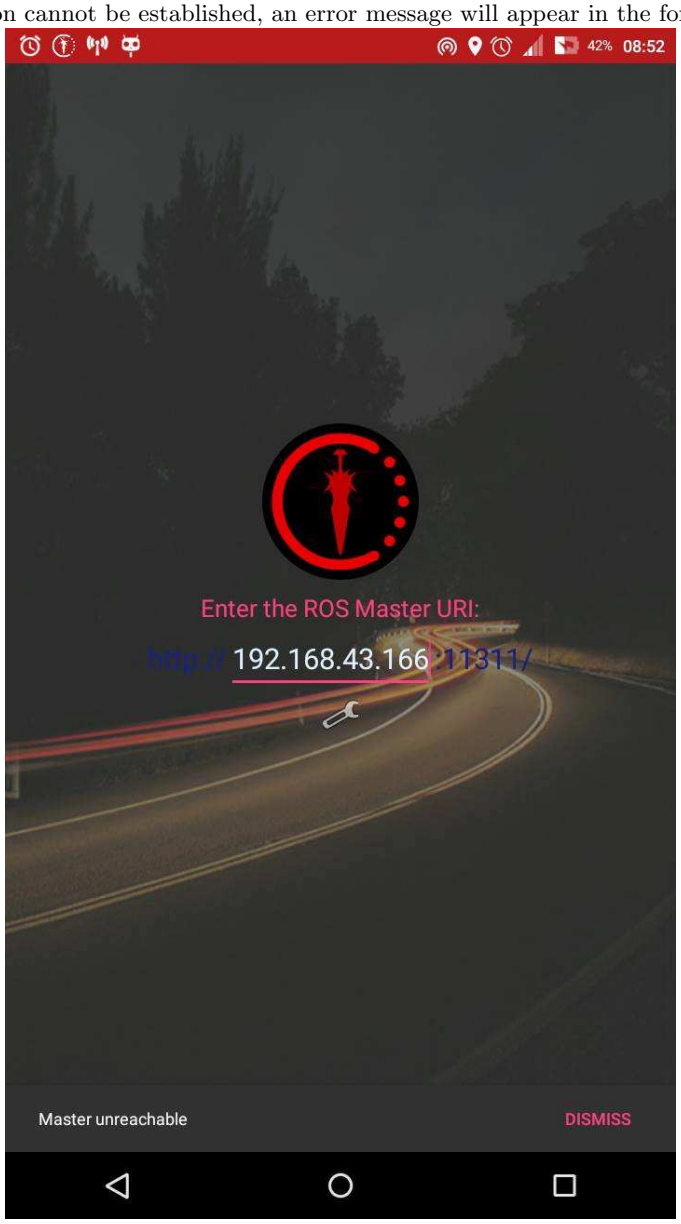

Fig. 3.2: Screen1: ROSConnectActivity - Connection Failed. Try Again

If the connection can be established but no ROS message is being published, this No Message being received dialogue box will be displayed.

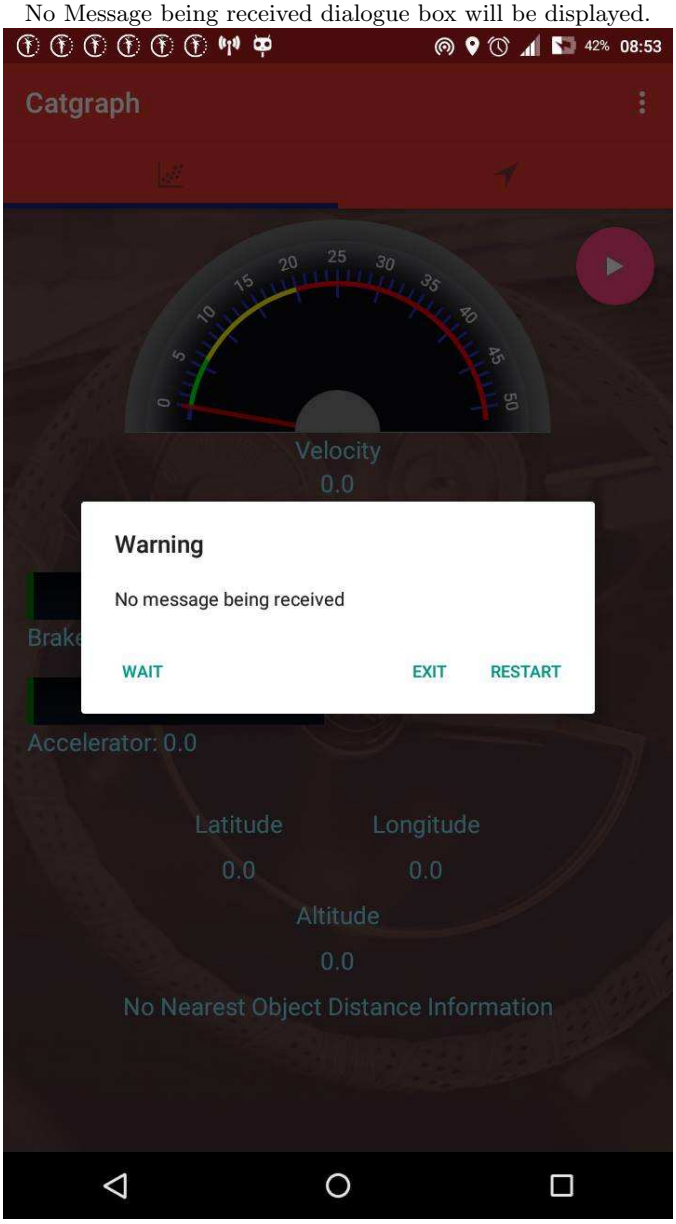

Fig. 3.3: Screen2: MainActivity - No ROS Message Received

If the connection to the ROS Master is lost, the Connection Lost dialogue box will appear.

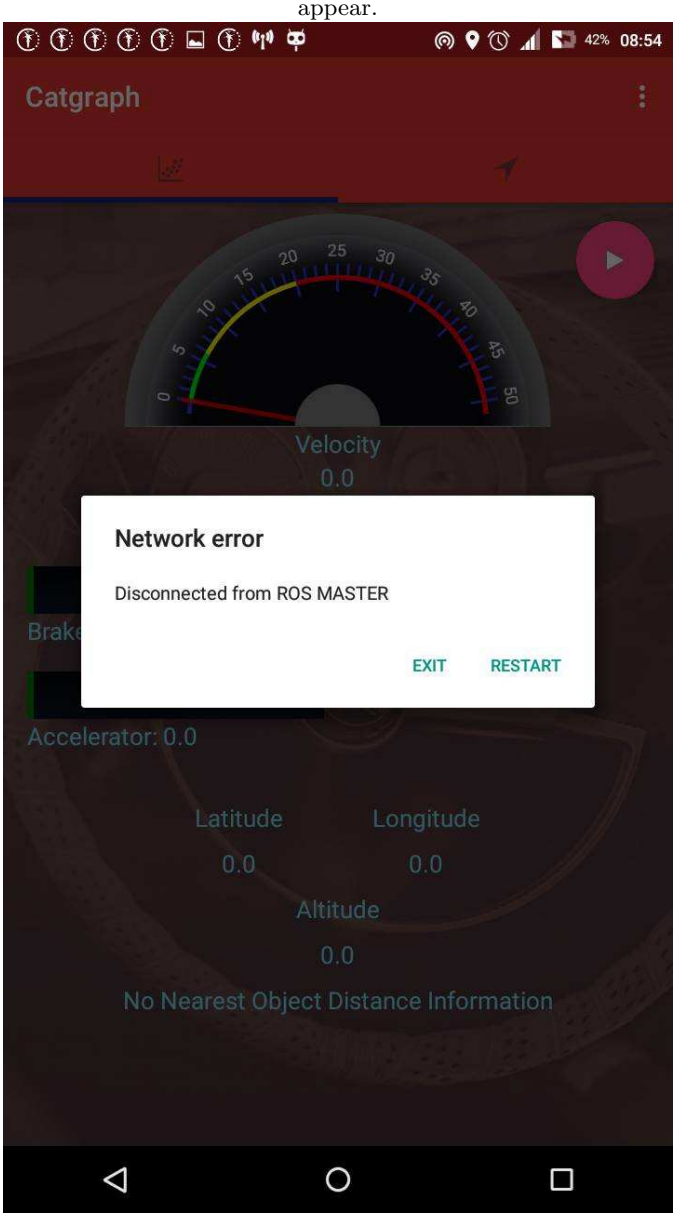

Fig. 3.4: Screen2: MainActivity - Connection to ROS Master Lost

This is the Main Activity Screen that will be displayed upon successful receipt of a ROS message.

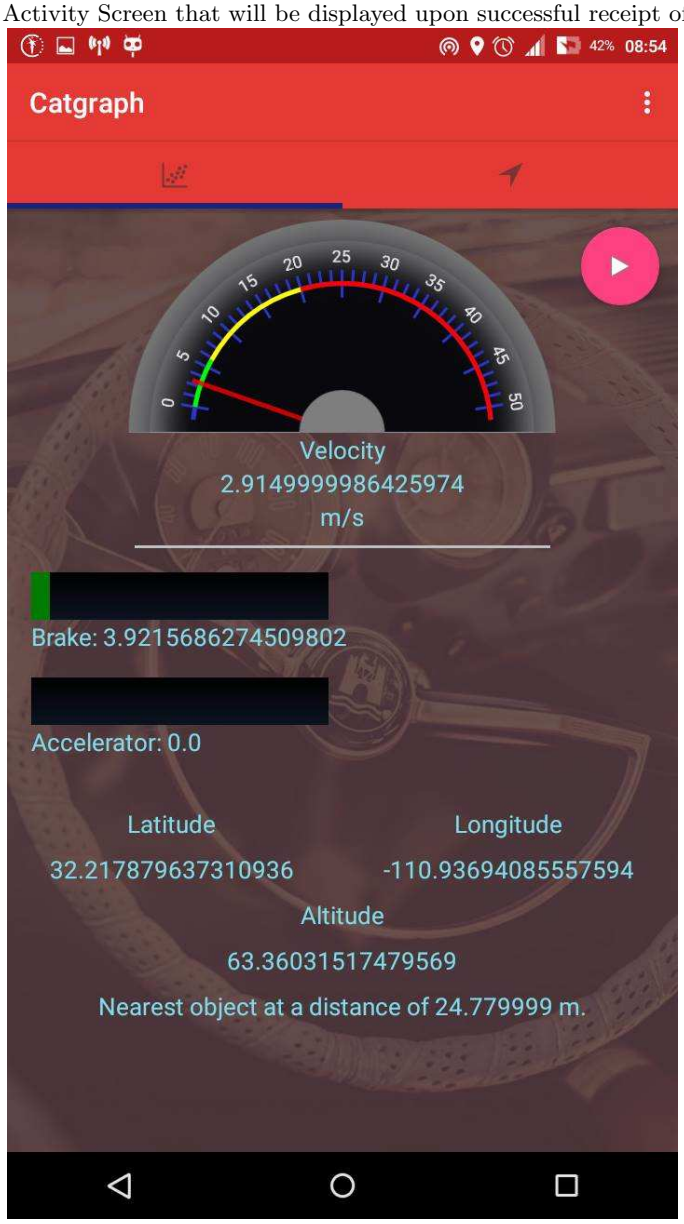

Fig. 3.5: Screen2: MainActivity - Home Tab - Normal Operation. Floating action button to stop monitoring vehicle and pause monitoring vehicle.

Speed Warning Dialogue Box will appear if the app senses that the speed of the vehicle has exceeded the speed limit.

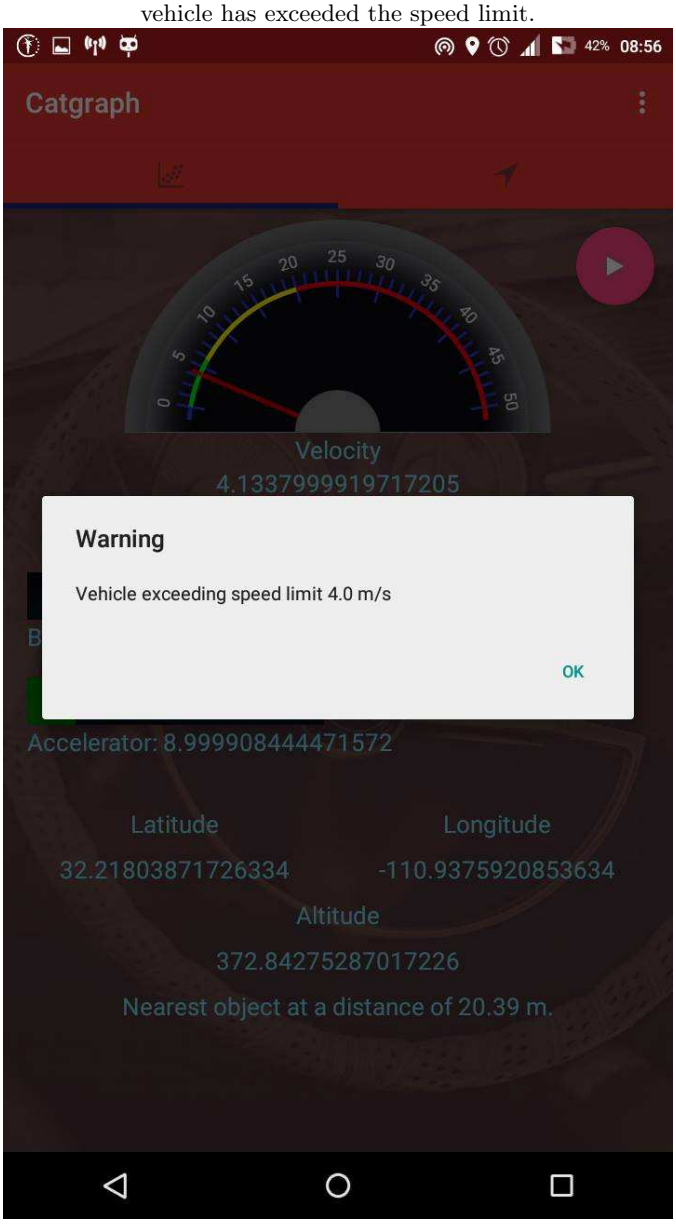

Fig. 3.6: Screen2: MainActivity - Home Tab - Speed limit Warning with alarm.

Obstacle Detection Dialogue Box will appear if an obstacle is detected closer than a given threshold.

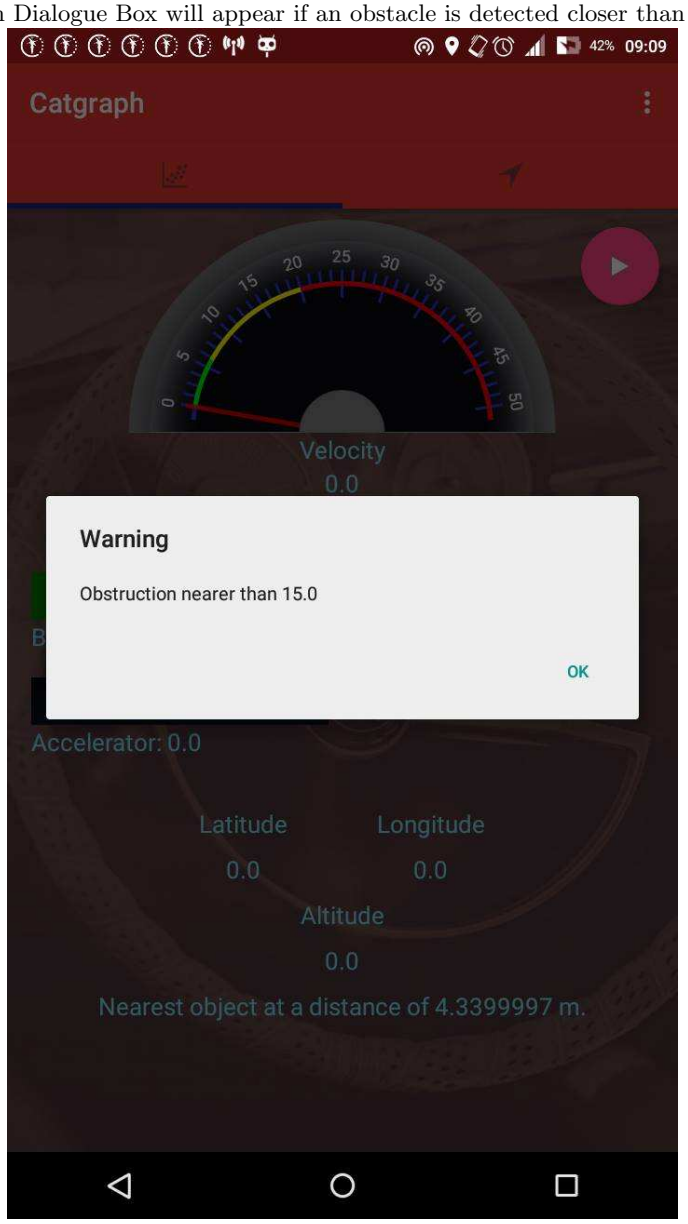

Fig. 3.7: Screen2: MainActivity - Home Tab - Obstacle detection warning with alarm.

GPS drift error message Box will appear in the form of snackbar if the GPS data from the catvehicle drift with respect to the GPS data determined by the GPS sensor of smartphone.

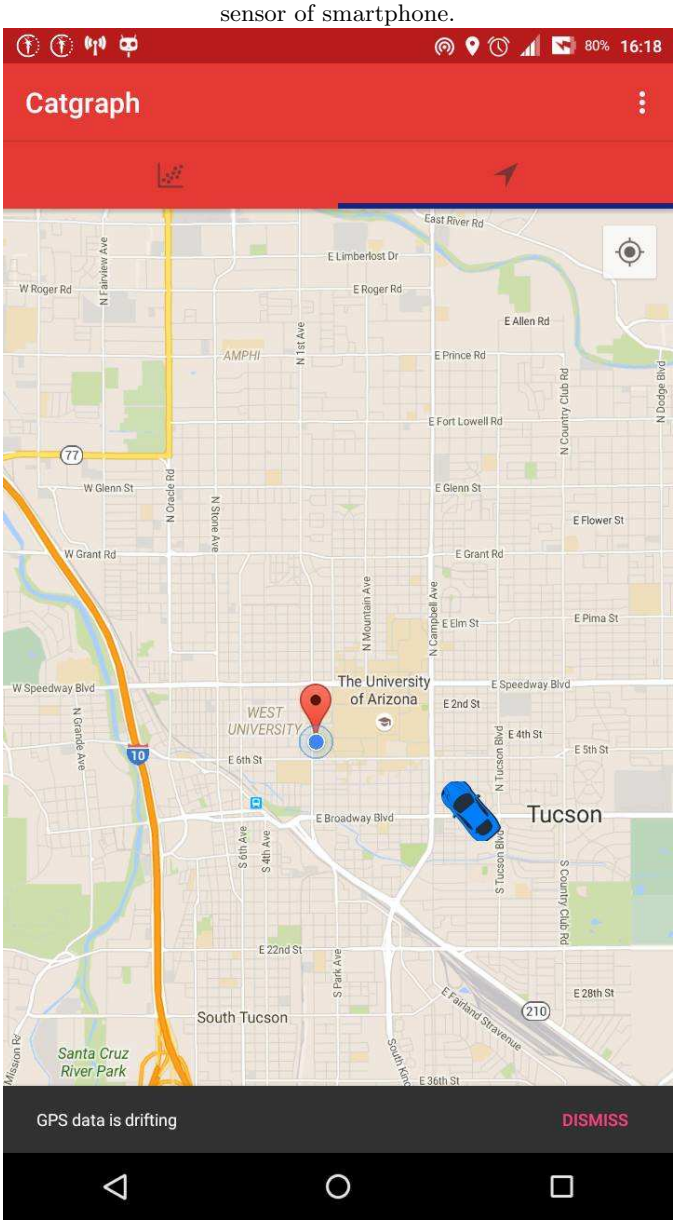

Fig. 3.8: Screen2: MainActivity - Home Tab - GPS drift warning with alarm.

Overflow Menu for additional functionality in the app.

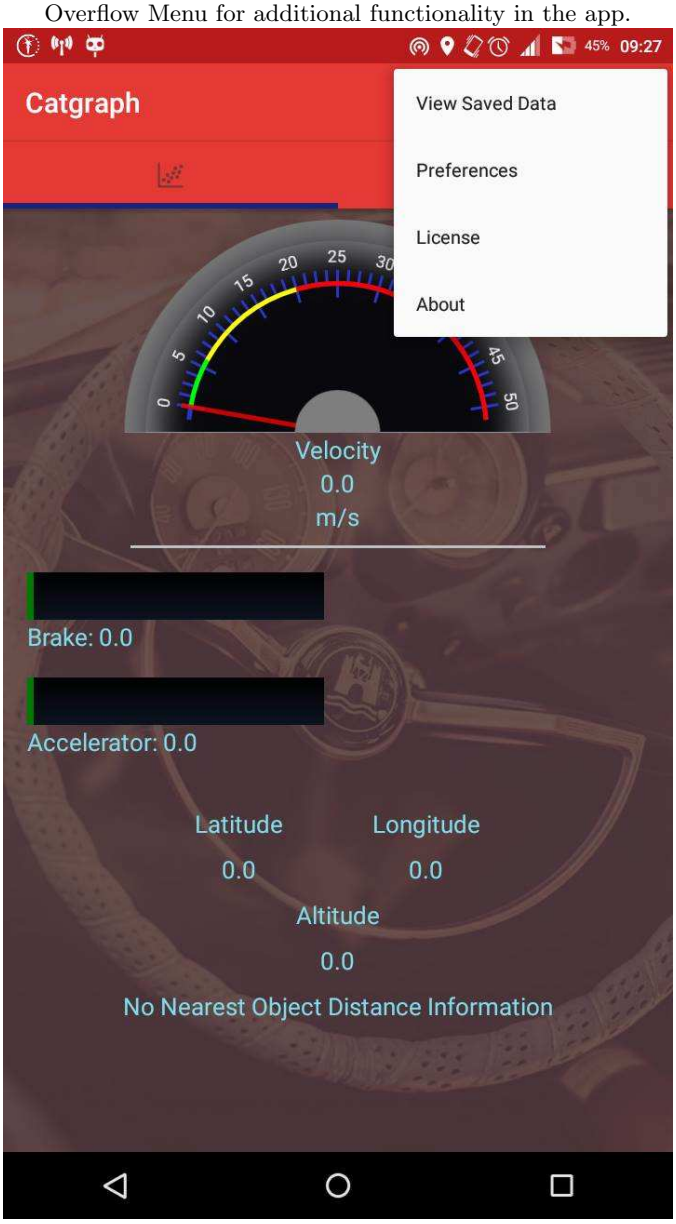

Fig. 3.9: Screen2: MainActivity - Home Tab - Displaying overflow menu.

Map Screen tab, which may be accessed by swiping from other tabs. A pointer for both the smartphone GPS and car GPS will be displayed.

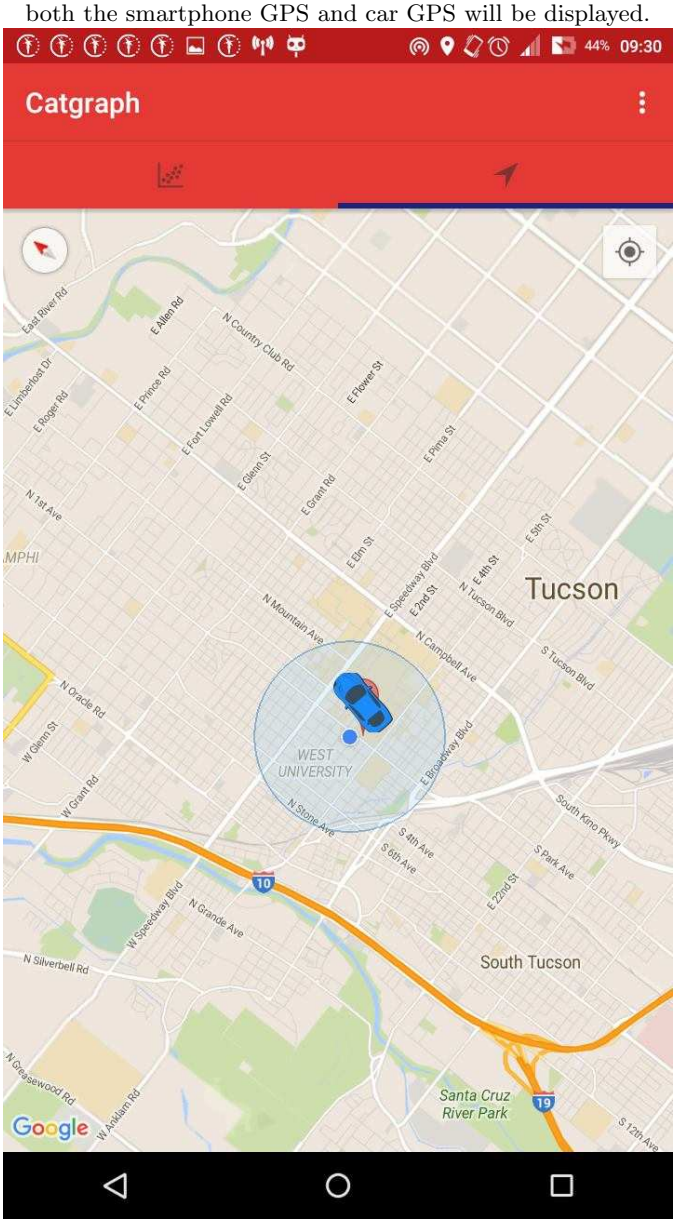

Fig. 3.10: Screen3: MainActivity - Map Tab - Showing two pointers, one from smartphone and another the location of vehicle.

List of file can be accessed by clicking view saved data option from overflow menu.

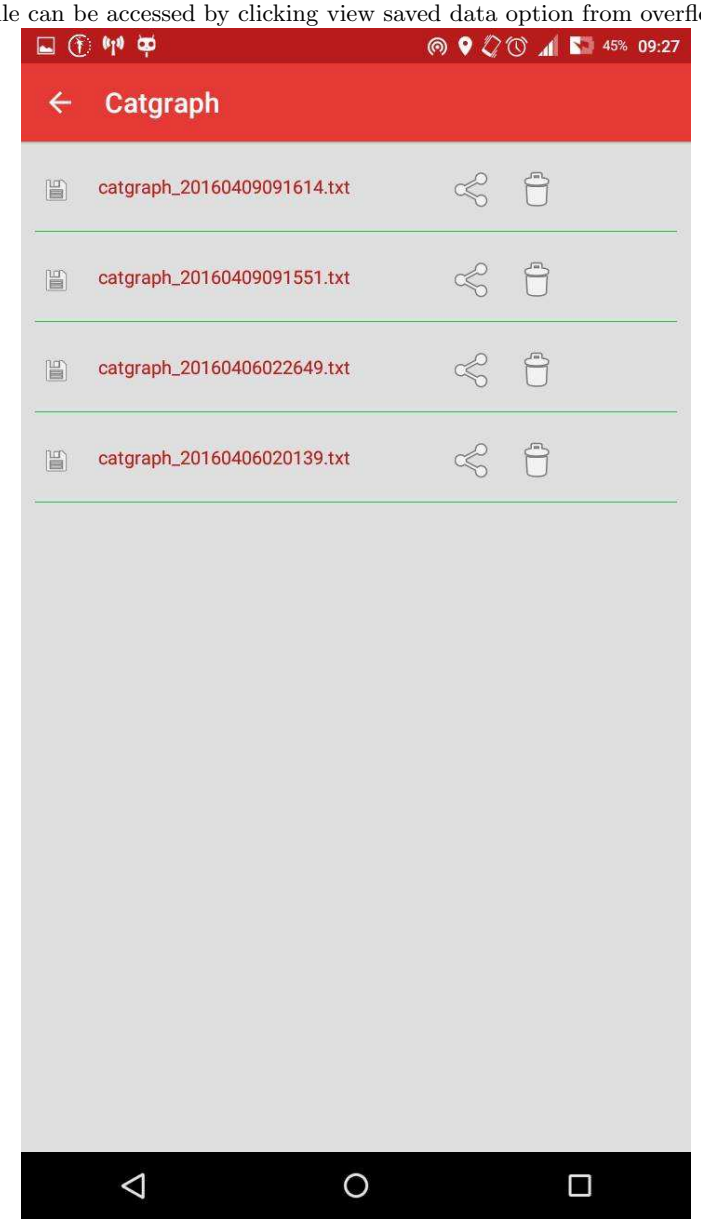

Fig. 3.11: Screen5: Activity showing list of Files - Displaying the file that has been saved with recorded data.

Preference/Setting can be accessed by clicking preferences option from overflow menu.

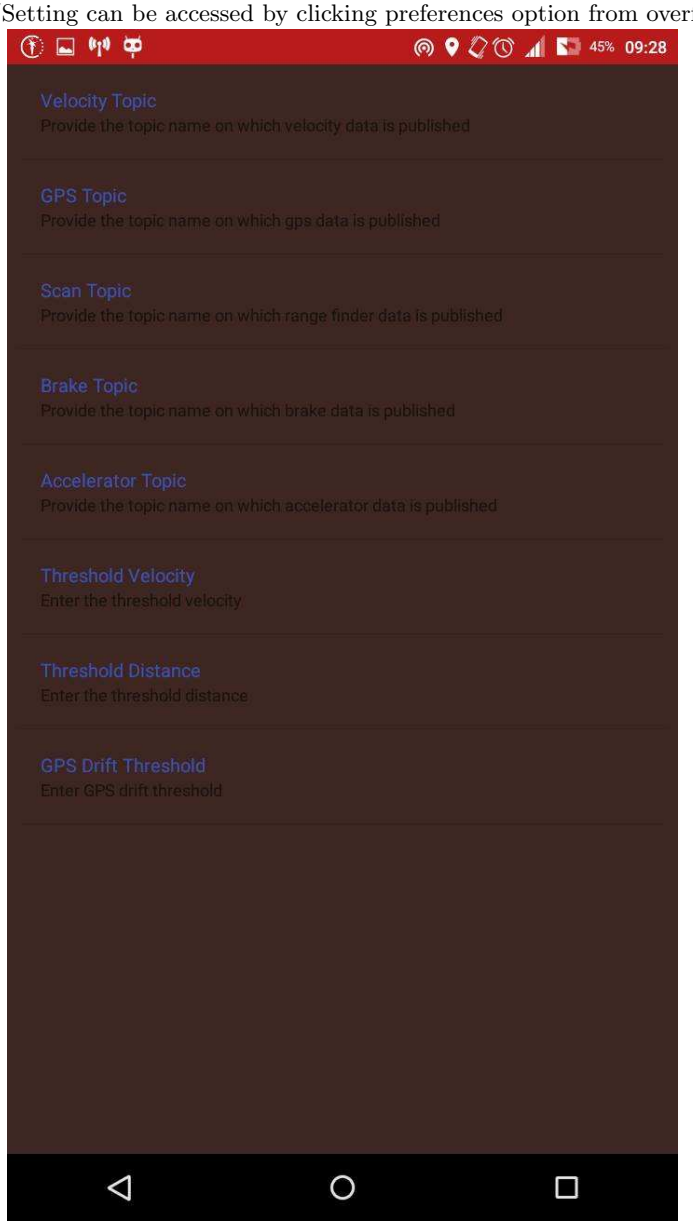

Fig. 3.12: Screen5: PreferenceActivity - Allows user to set threshold values and list of topics.

3.3 State Model

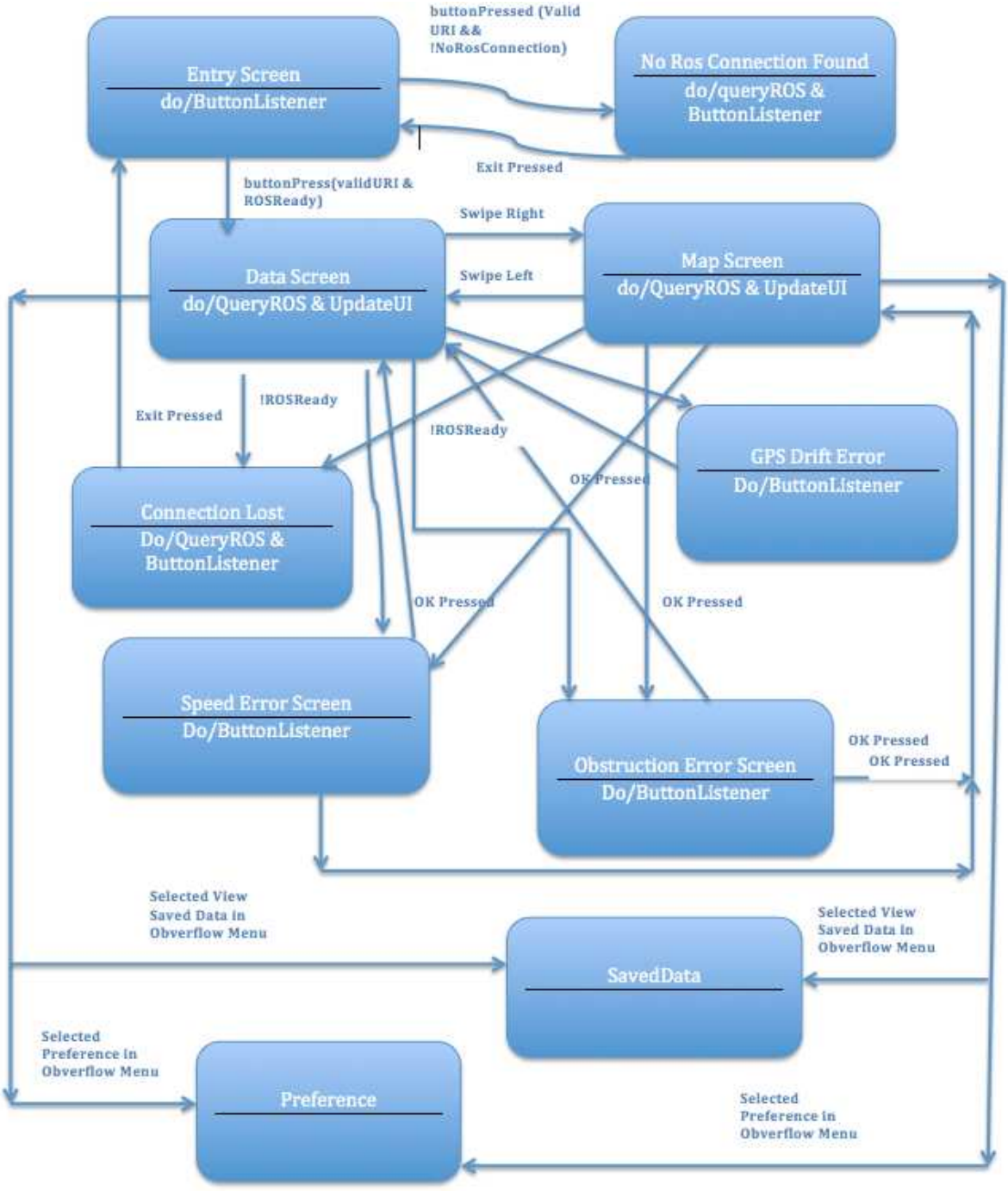

Fig. 3.13: State Model depicting the flow of applications based on user interaction

4

Domain Analysis

4.1 Overview

This portion of the document talks about the domain knowledge of the application and difference use cases that may be possible. We will also talk about how we are utilising this information to present provide various notifications to the user.

4.2 ROS Messages chosen to work with this application

As the primary purpose of this android application is to use ROS messages being published by the CATVehicle, the application will act as ROS subscriber. It is being design to handle following ROS topics:

1. /DistanceEstimator/dist: Estimate of the nearest object (in meters).
2. /DistanceEstimator/angle: Angle of above nearest object (in radians), angle increases in counterclockwise direction.
3. catVehicle/brake: Magnitude of Force and Torque being applied to the catVehicle in metric unit.
4. /catVehicle/steering: Magnitude of Force and Torque being applied by steering in metric unit.
5. /catVehicle/vel: Linear and angular velocity of the catVehicle.
6. /diagnostics: This topic is used to send diagnostic information about the state of the robot. Each diagnostic message has various level among warning, error, stale and okay for particular test component which is also specified in the message.
7. /gps_fix: This topic is of type sensor_msgs/NavSatFix. It is Navigation satellite fix for any global navigation satellitle system specified using WGS84 reference ellipsoid.
8. /scan: This topic is of type sensor_msgs/LaserScan. This contains scan data of neary by object catVehicle detects from its LIDAR sensors.

4.3 Use cases

Following use cases can be possible for this applications:

4.3.1 'B' requirements:

1. ROS Master URI present: connect to the ROS Master and move to the new activity.
2. ROS Master URI absent: flag error notification and ask user to re-enter the ROS Master URI.
3. Connection Lost: If the connection is suddenly lost, then pause all the activity and prompt user to re-enter the ROS Master URI once again.
4. Connection Successful, no ROS message is being received: Notify user of this situation and ask user if he wants to re-connect. There should be an option to reconnect with ROS Master URI or kill the connection and re-enter the ROS Master URI.
5. Connection Successful: Plot the odometer like graphics to display velocity and acceleration and level graph to display applied brakes and steering.
6. Display all the data such as longitude, latitude, velocity etc in numeric format.
7. ROS messages being received are inconsistent: For example brake value and accelerator value both are positive or greater than zero simultaneously. In this case, appropriate message must be displayed to the user in the form of Snackbar.
8. Major discrepancy between GPS data received from catVehicle and that of phone is detected: Notify user of the situation.

4.3.2 'A' requirements

1. Velocity of the Car is above the threshold: Notify the user of this situation.
2. Integrate Google Map to display the location of the catVehicle on map, simultaneously display the location of smartphone device on Google Map with different pointer.
3. Detect an object nearer than threshold distance: Notify user of the situation.

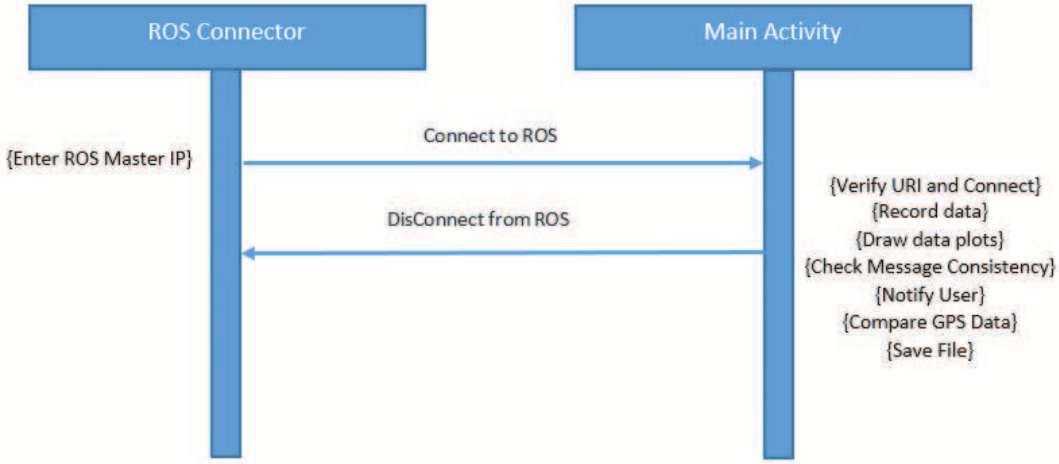

Fig. 4.1: Interaction Model showing Sequnce diagram of the application.

Note that it seems simple because our proposed design involves no more than two activities for achieving all the required task.

5

Important Algorithms

5.1 Overview

This section will discuss various approach that needs to be taken in order to implement the desired goal and various use cases discussed in the Chapter 4.

5.2 Connection checking and Spooling

Application will be pinging at a regular interval of time to check for the presence of connection.

5.3 Calibrating Speed and Acceleration for display

V_{max} and a_{max} will correspond to 180^0 for visualization and they will be calibrated in 180 units.

5.4 Laser scan data

Let the detection angle be θ.

Anglular resolution be α.

Then, $angle_{min} = -\theta/2$

$angle_{max} = \theta/2$

$angle_{increment} = \alpha$ which is angular resolution between two beams.

If the laser scan has frequency of β scans per second then in $1/\beta$ second, 1 scan will give $g = (\theta/\alpha) + 1$ beams.

Time increment can be calculated as $t_increment = (1/\beta)/g$ We can use this formulae to check for the consistency of the laser scan message. If the size of range array is not equal to the number of beams 1 scan gives then it should be assumed that there is something wrong with the message.

If r and θ are range and angle from one scan laserscan, obstacle is at $(rcos\theta, rsin\theta)$ in laser frame.

5.4.1 Algorithm

Algorithm to plot the 2-D map

```
foreach timestamp t
begin
    i = 0
    foreach data in rangeArray r
    begin
            if(range ϵ [range_min, range_max])
                    plot(r[i]*cos(i*α),r[i]*sin(i*α))
            end if
    end foreach
    invalidatePlot
end foreach
```

5.5 GPS data

GPS data is read from the topic /gps_fix and is of the form sensor_msgs/NavSatFix.

position_covariance provides an insight into the variability of the GPS data being received. High covariance means GPS data is bad and it is a candidate for the rejection. But how to interpret the position_covariance matrix is determined by position_covariance_type.

uint8 COVARIANCE_TYPE_UNKNOWN = 0

This is to be used when the GPS data does not provide any quality estimation or we do not know to model the reported precision in SI units (m^2). Our algorithms will assume large possible errors with this case and as a safety application is supposed to outright reject this.

uint8 COVARIANCE_TYPE_APPROXIMATED = 1

This use case occurs when GPS sensor is not very sophisticated for high accuracy. One safe way to use the data would be to approximate the covariance as a diagonal matrix with the accuracy squared for each axis application wants to report.

uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN = 2

When GPS sensor is high end, it outputs variances or standard deviation for every measurement. For example, when such kind of sensor gives outputs longitude deviation, latitude deviation and altitude deviation, we simply have to square these values and plug them into the diagonal of the covariance matrix, leaving the other elements zero.

uint8 COVARIANCE_TYPE_KNOWN = 3

This is used by very sophisticated GPS sensors and data are extremely accurate. They give 3x3 full covariance matrix as 9 numbers and can be trusted completely.

Note that covariance matrix are stored as one dimensional array in row major form.

In order to check for the accuracy of GPS data, we are going to calculate correlation coefficient of GPS data on the fly.

Correlation coefficient calculation for latitude:

A be the latitude data set from catVehicle messages. a be the random variable of data set A.

Let B be the latitude data set from smartphone GPS sensor. b be the random variable of the data set B.

Then the correlation coefficient can be calculated as :

$$\mathbf{r} = \frac{\mathbf{n}(\sum \mathbf{ab}) - (\sum \mathbf{a})(\sum \mathbf{b})}{\sqrt{[\mathbf{n}\sum \mathbf{a^2} - (\sum \mathbf{a})^2][\mathbf{n}\sum \mathbf{b^2} - (\sum \mathbf{b})^2]}}$$

If correlation coefficient is below threshold, then user should be notified about the bad GPS data.

6

Class Design

6.1 Overview

This section talks about the class design and its relation with User Interface discussed in the chapter 3.

6.2 Class Diagrams

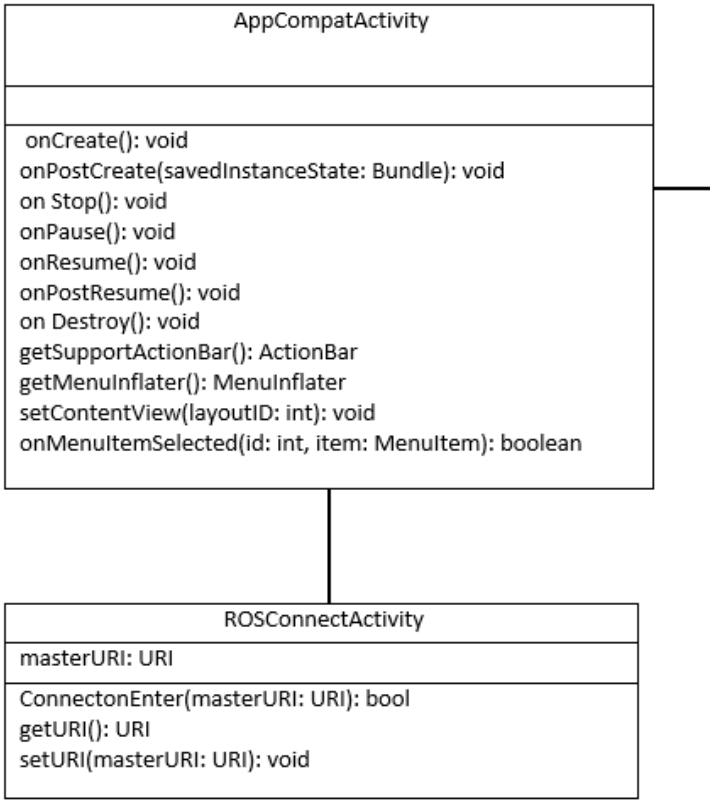

Fig. 6.1: ROSConnnectActivity Class Diagram

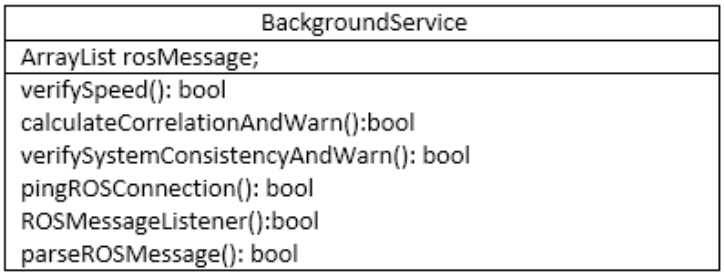

Fig. 6.2: BackGroundService Class Diagram

This class will be instantiate in the background. Its lifecycle will be application's life cycle and it will check for ROS connections, receive and parse ROS Messages and will act as bridge between various activities.

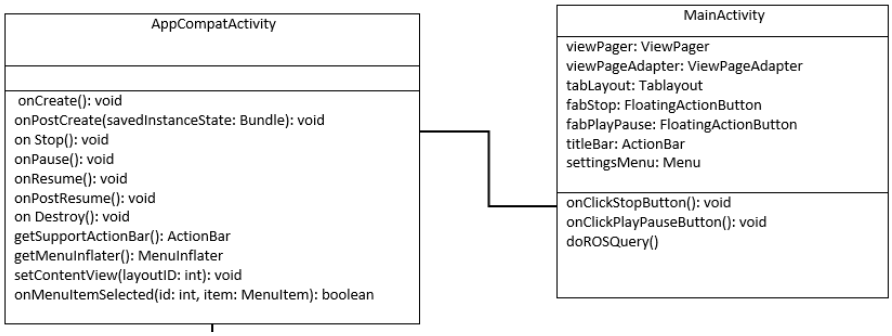

Fig. 6.3: MainActivity Class Diagram

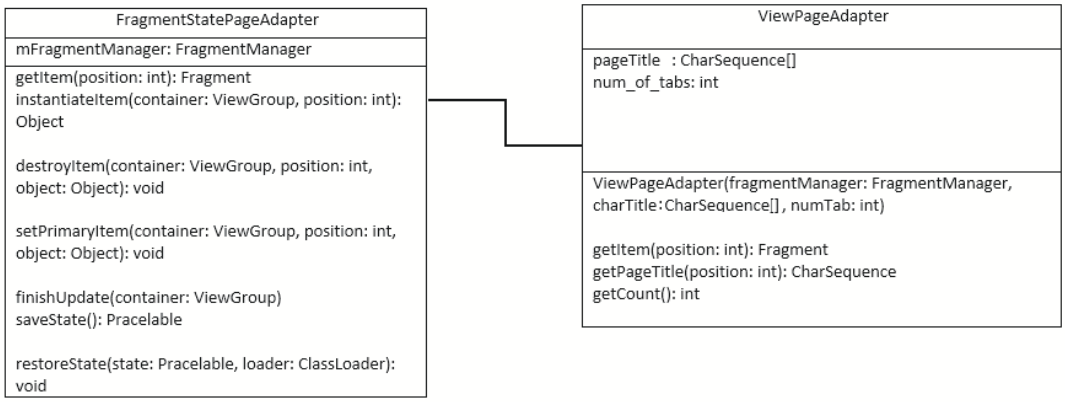

Fig. 6.4: ViewPageAdapter Class Diagram

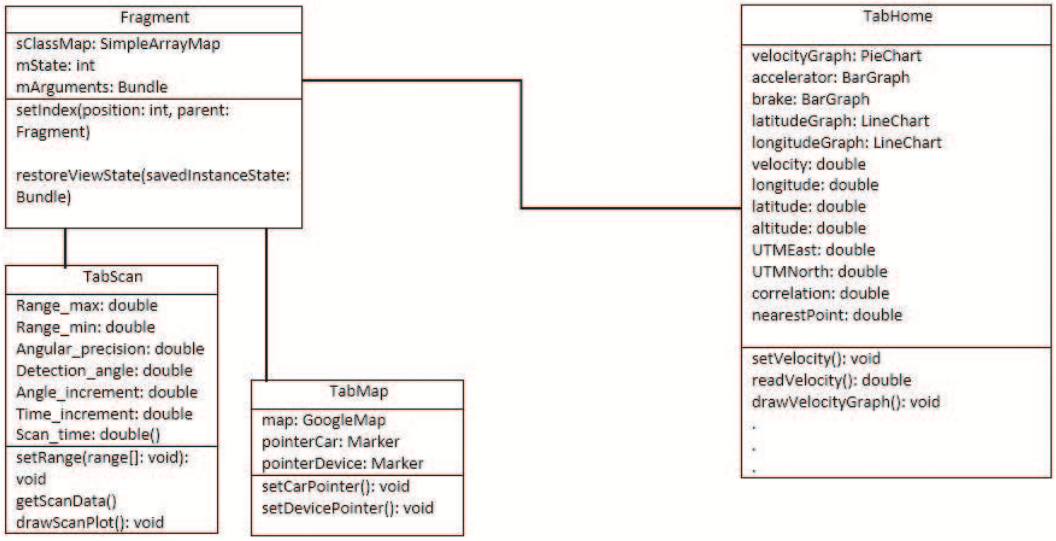

Fig. 6.5: TabHome, TabMap and TabScann Class Diagrams

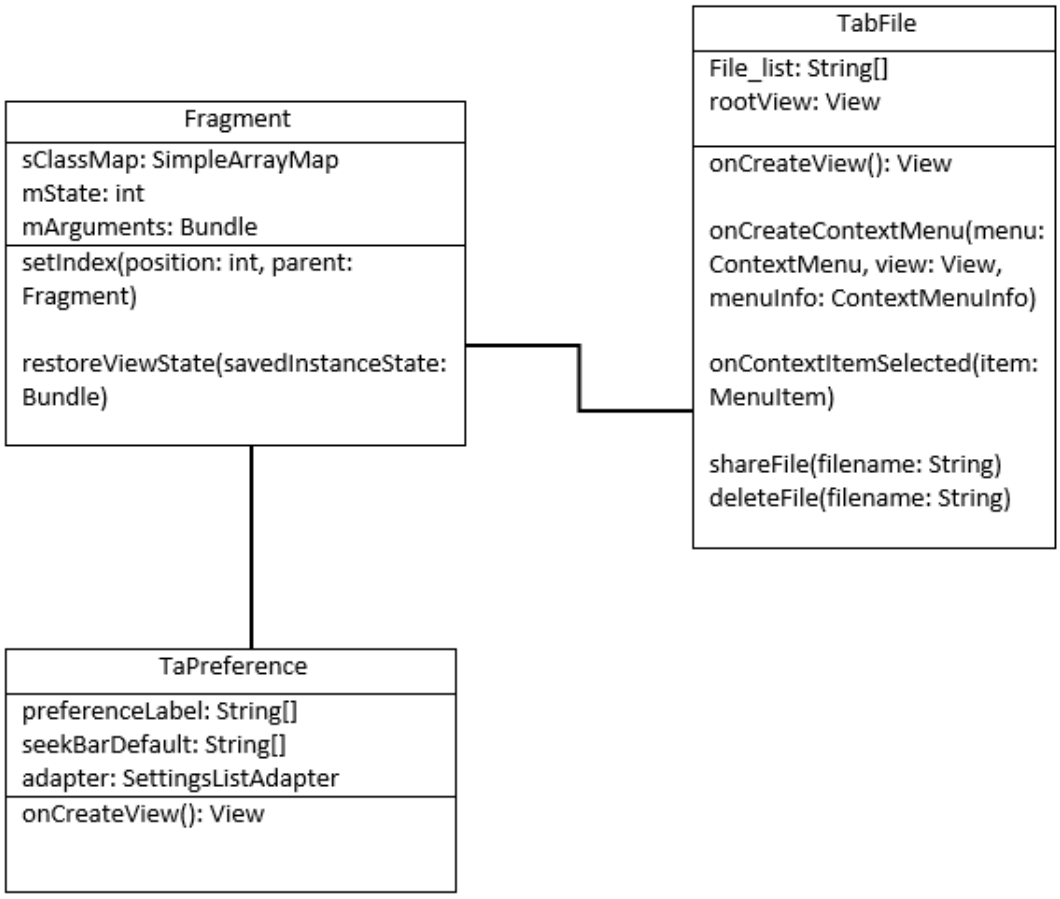

Fig. 6.6: TabPreference and TabFile Class Diagrams

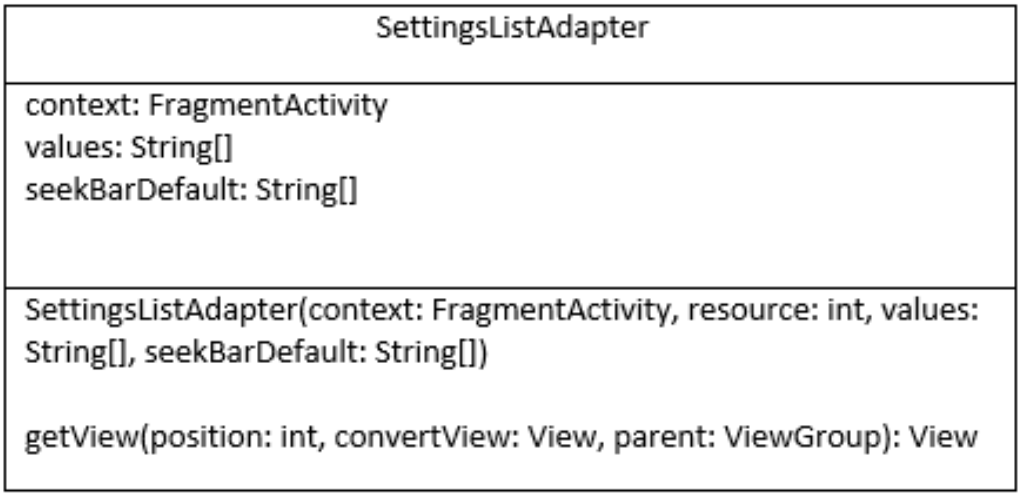

Fig. 6.7: SettingsListAdapter Class Diagrams

ROSConnectActivity will implement Screen1, MainActivity with the help of other classes will implement Screen 2-5.

Note that the classes thought of in this report are not sufficient and we may need some other classes as we discover the requirements. It is also possible that we have to change the class definition to suit our needs.

7

Testing Strategy

7.1 Overview

This chapter explains the various test strategy that will be employed to test the application catGraph. It has been planned they will be tested on different devices that runs on android.

7.2 Off-site test

As a part of testing strategy, we are initially going to use ROS-Gazebo environment to create virtual environment for catVehicle. There after we will use readily available rosbag files available from previous experiments of catvehicle. Rosbag play command will simulate the actual experiment and ROS will publish all of the messages that were recorded in the real time. Catgraph application will subscribe the topics being published and act upon them.

7.3 Fake data test

We will create some fake data with extreme values and some inscosistent messages to test the application for different possible scenarios.

7.4 On-Site test

There is also a plan to test the application after beta release using actual catVehicle, probably in UA tech park of CatTran lot.

7.5 ROS Behaviour and Error handling Test cases

1. Incorrect Master URI/NO ROS Master present: In this case if HomeActivity doesn't open, test passes.
2. Correct Master URI: HomeActivity should open upon pressing enter.
3. ROS Master running but not publishing any data: If no alarm activates, fail.
4. ROS Master killed while recording the data: Application should move to ROSConnect Activity after saving the data.
5. No /scan data being published: Boundary scan graph should not change. If change, test fails.
6. No /gps data being published: If use is not notified with alarm. Test Fail.
7. GPS of smartphone is not active: If user is not notified of this situation, test fail.
8. catVehicle is not moving: If user is not notified of this situation test fails.
9. catVehicle detects obstruction: If user is not notified of this situation test fails.
10. When No ROS Connection Found dialog box is active, press Exit. If application doesn't move to ROS Connect Activity, test fails.
11. While No ROS Connection Found dialog box is active, make the previously illegetimate ROS address legitimate and try again. If the application doesn't move to Main Activity screen, test fails.
12. While in Home Activity Screen and among Map tab, Boundary Scan Detect tab, File tab, or Settings tab, make ROS connection become unavailable. If Connection Lost dialogue doesn't appear, test fails
13. While Connection Lost dialogue box is active, make ROS connection available once again. If you don't see Home Activity active, test fails.
14. While Connection Lost dialogue box is active, send exit signal. If application doesn't move to RosConnectActivity, test fails.
15. Send a GPS signal that is errant while in each of the five Home Activity tabs. If the GPS Error dialogue is not shown, test fails.
16. While GPS Error dialogue box is active, press Okay signal. If HomeActivity is not resumed, test fails.
17. Send a speed signal that exceeds the speed limit while in each of the five main tabs. If the Speed Error dialogue doesn't show up, test fails.
18. While Speed Error dialogue box is active, press OK. If HomeActivity is not resumed, test fails.
19. Send an obstruction signal while in each of the five main tabs. If the Obstruction Error dialogue box is not shown, test fails.
20. While Obstruction Error dialog box is active, press OK. If HomeActivity is not resumed, test fails.

7.6 UI Behavior tests

1. Swipe right and swipe left to each of [Home tab, Map Tab, Boundary Scan Tab, File Tab, Settings Tab] and verify that the application navigates to correct tab.
2. Check whether upon pressing file item list in file tab, context menu appears.

8

Integration with Platform

This application requires use of smartphone GPS sensors as well as an autonomous vehicle equipped with LIDAR and GPS sensors.

GPS sensors on smartphones are required to calculate the correlation with GPS data received from catVehicle.

LIDAR is required to detect obstruction around the catVehicle.

This application is not suitable for systems that do not possess these sensors.

ROS version: indigo

Android version: Ice Cream sandwich or higher.

Installation requires minimum of 200MB of disk space in android devices.

9

Task allocation and breakdown

9.1 B requirement task allocation

1. The Application should be able to establish a connection to the ROS publisher of the CATVehicle. (Rahul)
2. The Application should be able to receive the ROS message being published by the CATVehicles ROS publisher. (Rahul)
3. The Application should be able to display a graphical representation of the timeseries messages being received. (Jue)
4. The Application should be able to perform error analysis of the messages for system consistency. (Jue)
5. The Application should be able to save the message in specified format and should be able to email via favorite email client when prompted. (Matt)

9.2 A requirement task allocation

1. The Application should be able to trigger a warning notification when it detects that the vehicle is exceeding the speed limit. (Jue)
2. The Application should be able to notify user if the LIDAR detects obstruction in front of the car. (Matt)
3. The Application will compare GPS output of the car with phone GPS readings and be able to alert the user of discrepancies. (Matt)
4. The Application will allow the User to enter an expected update frequency from the vehicle and if this frequency is not met the User will be alerted. (Rahul)
5. The Application should send a decelerate command to the car if above situation occurs. (Rahul)

10

Time line for completion

Fig. 10.1: Time line of the project representing task allocation of each member.