Android Sensing Project

Project Option 1: Bluetooth device survey

Luke Slemon 16421694(student submitting the report)

Caolan Gilroy

line 2-name of organization, acronyms acceptable

line 3-City, Country

line 4-e-mail address if desired

*Abstract*  
Android Applications are created for a wide range of scenarios such as locating the nearest ATM, streaming music, and ordering food. The application outlined in this report was created as a Bluetooth Device surveyor, which scans for Bluetooth devices at different locations. The purpose of this is to determine how many Bluetooth devices are active at any one time in a single area, and in particular what kind of device it is, i.e. Laptop, Phone, Heartbeat sensor, headphones, etc. Using the phones GPS and Bluetooth sensor, latitude, longitude, and Bluetooth device information was collected and stored to Google’s own open source Database Firebase.

Keywords—Android;GPS;Bluetooth;Firebase

# Introduction

This report details the development of two Android Applications, both of which utilize GPS location. The primary aim of this task was to teach students how Android development was accomplished and how to use opensource tools such as Android Studio and Firebase to create simple yet useful applications.

Android is a linux based operating system designed specifically for smart phones[1]. It is the counter part to Apple’s IOS in the smartphone game. Android, like linux, is open source and can be modified or remodeled to make more specialized OS similar to Cyanagen.

Android apps are typically designed using either Java or a newer language Kotlin using the dedicated IDE, Android Studio. Android applications are composed of 2 primary components, Activities and Services. Activities are the primary entry point for the user to the application, they handle all UI and presenting data to the User[2]. Services are similar to Activities without a UI. They are typically bound to an Activity and run in the background while the Activity is alive. Once the Activity is closed, the service will terminate.

If application state is important, then a layer of persistence is required to ensure the application state remains unchanged after the app is closed. Android developers are given the choice between using either the Android Room database or Google’s Firebase. The room database is a local SQLite database cached directly on the host device[3], while Firebase is a remote database holding JSON tree records.

# Workshop application : GPS location Upload and Retrieval

## Before working on the Bluetooth Surveyor application, a short workshop was completed where a simple GPS application that stored and retrieved GPS data was developed. The aim of this workshop was to teach students to use Android Studio for creating apps, how to use the devices sensors in an application and how to store the recorded data to a Firebase database and retrieve it when it’s needed.

## Application Description

The primary function of this application was to utilize the phone’s GPS sensor to determine the location of the phone every time the phone moves more than 5 meters or 10 seconds. Once the a location update is fired, the new location data is stored to the firebase database by utilizing a locationData object to abstract the data being stored to the Data base.

The user also has the option to force a location update at the push of a button allowing them to record their location data at will. Drop pins/markers are placed at the stored co-ordinates, where the user can tap on the marker and are presented with the co-ordinates.

## Application Code and Behaviour

### Structure

### The application is structured using a single Activity with a Map UI and a single button. The single Activity (MapsActivity) handles the location updates, the permission controls, and the database read/write operations.

### When the application is opened, the onCreate method is called which is used to inflate the view using the XML file, assign a button handler to the single button, check permissions and finally initiate location updates.

Once the UI has been loaded, the onMapReady method for the Map UI is called and a new data listener for the database is called either when new data is entered to the Database or when the listener is initially created.

### Permissions

Before the application can access the phone’s location, the user must grant the Coarse Access Location, and Fine Access Location permissions. When requesting the User’s permission, a dialog will be presented which asks them to grant their permission in order to allow the app to record their location.

### GPS Location Handling

After permissions have been granted, before location updates can be initiated, the app determines if the Location service or network service is enabled. If neither of them are, the user is prompted with a dialog requesting they enable their location.

The location of the user will be determined either by using the GPS provider which utilizes three satelites to triangulate the phone’s position, or by using the network provider which performs a lookup based on nearby cell towers and WiFi access points[4].

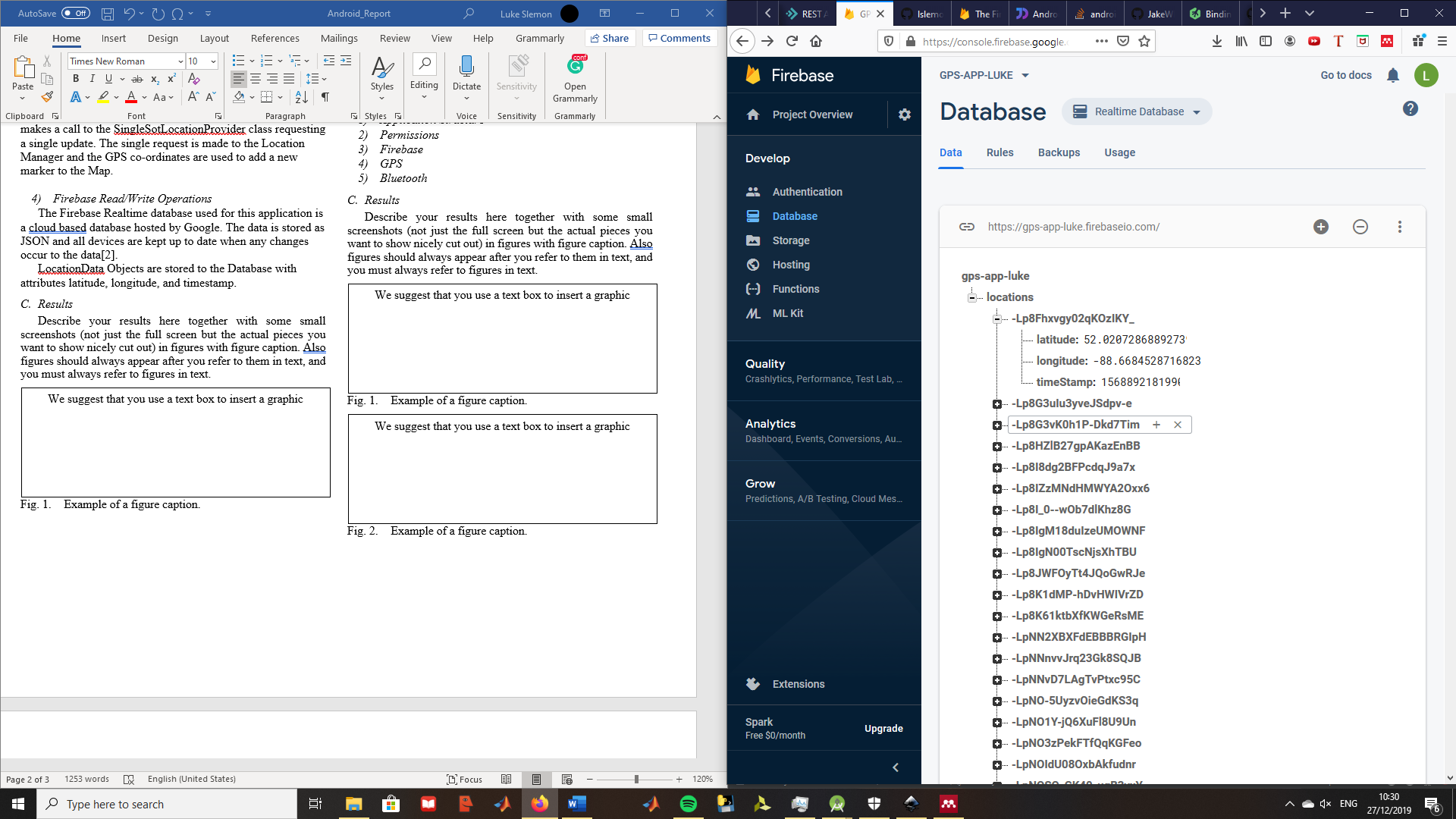
The application requests location updates every 27 minutes or every 100 meters. When one of these conditions is met, the onLocationChanged method of the LocationListener will be called. When this method is called, the Location object holding the Latitude and Longitude co-ordinates can be accessed and stored to the database.

The alternative option for requesting the current location of the user is by pressing the only button in the application. Once pressed, the forceGetLocation method is called which makes a call to the SingleSotLocationProvider class requesting a single update. The single request is made to the Location Manager and the GPS co-ordinates are used to add a new marker to the Map.

### Firebase Read/Write Operations

The Firebase Realtime database used for this application is a cloud based database hosted by Google. The data is stored as JSON and all devices are kept up to date when any changes occur to the data[5].

LocationData Objects are stored to the Database with attributes latitude, longitude, and timestamp. The JSON tree structure detailed in Figure 1 represents how the data is structure in the database, where locations are a child to the head of the database and each entry is a child to locations.

**

**Fig. 1. Example JSON tree for storing data in Firebase**

When a location update is received, the GPS co-ordinates of the new location are wrapped in a Location Data Object before pushing to the database. To determine where the data will be stored, push to the database to create a new node with a unique key[6]. Using a reference to the specified child, in this scenario it is locations, the new key is used to update the values at the new node.

Reading data from the database is handled using asynchronous listeners which are triggered once for the initial state of the database, and triggered again with any subsequent database changes[6]. When the listener is triggered, a data snapshot, which is a picture of the state of data at a particular key in the database. The snapshot of the data can then be cast to the LocationData object to make it easier for the code to access it.

## Results

The Application successfully recorded my location as seen in Figure 1, and each location is matched with a timestamp to keep a record of when a user was detected. The following Figures show the UI of the Maps Activity correctly placing markers for every location saved to the Database.

Figure 3 shows that the onClick listener for the markers work sufficiently by opening the Marker’s title, the time and date the entry was saved. Finally, figure 4 shows the onClick listener for the Force Get Location button works because the Loading Location Alert dialog is presented to the user while the SingleShotProvider is awaiting the location update.

A screenshot of a cell phone

Description automatically generated

**Fig. 2. Maps Activity UI with markers for every saved location**

A close up of a map

Description automatically generated

**Fig. 3. Maps Activity UI with marker title presented after marker clicked**

A screenshot of a computer

Description automatically generated

**Fig4. Maps Activity UI with alert Dialog presented after force get location pressed.**

# Project Application: Bluetooth Surveyor

The Bluetooth Surveyor Application was developed as a tool to determine how many devices leave their Bluetooth active and what kind of device was detected. This could be a useful tool for cybersecurity experts ensuring their system has no physical entry points for any hackers. Hackers can potentially gain remote access via a Bluetooth Keyboard. A cybersecurity expert can potentially use this tool to then locate any pitfalls and see how easily devices be located from a hacker’s position (i.e. public toilet next door).

## Application Description

The primary function of this application is to use the Bluetooth sensor on the host device to scan for nearby Bluetooth devices, while keeping a note of the location of the phone when it discovered each device. Using the on-board GPS sensor, and the devices Location Manager the phone can request location updates every 100 meters or 30 minutes. With every location update, a brief scan for Bluetooth devices will save a collection of devices associated with that location. Each discovered device will be stored to a remote Firebase database instance.

The recorded devices will be presented to the user in a list with information regarding their Bluetooth Address, Device name, and class of device be it a computer, headphones, a smartphone, keyboard, or heart monitor. On the Map each marker will represent a stored location, and when any of the markers are pressed, they will tell the user how many devices were discovered at this location.

## Application Code and Behaviour

### Application Structure

The application is structured using three activities and a single Service for handling all background, long running operations. When the application is opened, the user is presented with the opening Activity which has a logo and two buttons, one of which will open the Device List Activity and the other will open the Maps Activity. This activity just works to help separate the two primary Activities from each other, and also to demonstrate the use of navigation within in Android applications.

Add image of opening activity here

The Device List Activity presents a list of all discovered devices to the User, with information regarding class of device, the device name, and the Bluetooth Address. When any of the records in this list are clicked, the Maps Activity will be opened and the marker with the corresponding location of the device will be highlighted.

Add image of device list activity here

The Maps Activity presents a Google Maps style map to the user with a collection of markers positioned at the previously stored locations. When any of the markers are selected, they present the number of devices detected at this location as well as the time and date the entry was saved.

Add image of map sactivity here

Finally, the Service will be utilized for handling all background operations such as location updates and Bluetooth scanning. When a location update is fired, then a brief Bluetooth scan will begin for 1 minute. In this minute, the scanned Bluetooth devices will be saved to a collection, and once the timer completes, the devices and the location will be saved to the Firebase.

### Permissions

Before the application can access the phone’s location, the user must grant the Coarse Access Location, and Fine Access Location permissions. When requesting the User’s permission, a dialog will be presented which asks them to grant their permission in order to allow the app to record their location.

Finally, before the app can access the devices Bluetooth sensor it must request the Bluetooth Permissions.

### GPS

Similarly to the previous app outlined in this report, the

### Firebase

### Bluetooth

## Results

Describe your results here together with some small screenshots (not just the full screen but the actual pieces you want to show nicely cut out) in figures with figure caption. Also figures should always appear after you refer to them in text, and you must always refer to figures in text.

Fig. 1. Example of a figure caption.

We suggest that you use a text box to insert a graphic

Fig. 2. Example of a figure caption.

We suggest that you use a text box to insert a graphic

##### Conclusions

Conclude on all that you have done and described and the outcomes of the work. Describe any difficulty that you experienced or new knowledge you acquired. This may be a slight repetition of the conclusions also shown in the abstract. You should include a few references in your work, this can take the form of web addresses, however, a web address must be accompanied by a title like a regular reference and also by a date you accessed it, as shown in [1].

##### References

[1] “What is Android? Everything you need to know about Google’s OS.” [Online]. Available: https://www.androidauthority.com/what-is-android-328076/. [Accessed: 29-Dec-2019].

[2] “Android Activity and its Lifecycle.” [Online]. Available: https://blog.mindorks.com/android-activity-lifecycle. [Accessed: 29-Dec-2019].

[3] “Room Persistence Library  |  Android Developers.” [Online]. Available: https://developer.android.com/topic/libraries/architecture/room. [Accessed: 29-Dec-2019].

[4] “Android Location Providers (gps, network, passive) | developerlife.com.” [Online]. Available: https://developerlife.com/2010/10/20/gps/. [Accessed: 26-Dec-2019].

[5] “Firebase Realtime Database.” [Online]. Available: https://firebase.google.com/docs/database. [Accessed: 27-Dec-2019].

[6] “Saving Data  |  Firebase Realtime Database.” [Online]. Available: https://firebase.google.com/docs/database/admin/save-data. [Accessed: 27-Dec-2019].