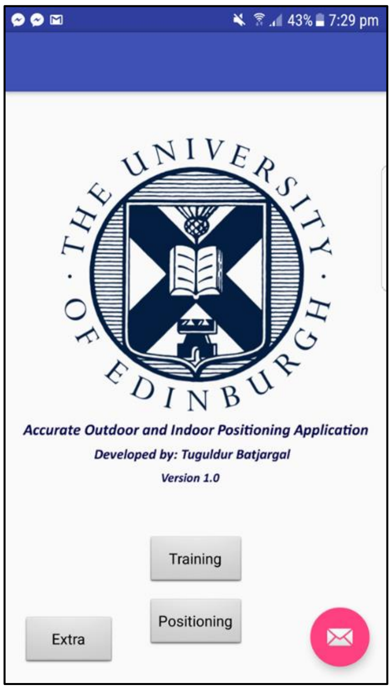
Embedded Mobile and Wireless System (E-Wireless) 5

Assignment 2: Accurate Outdoor and Indoor Positioning

1. Users Guide



THE UNIVERSITY OF EDINBURGH
Accurate Outdoor and Indoor Positioning Application
Developed by: Tuguldur Batjargal
Version 1.0
Training
Positioning
Extra

Figure 1.1: Main Activity Window

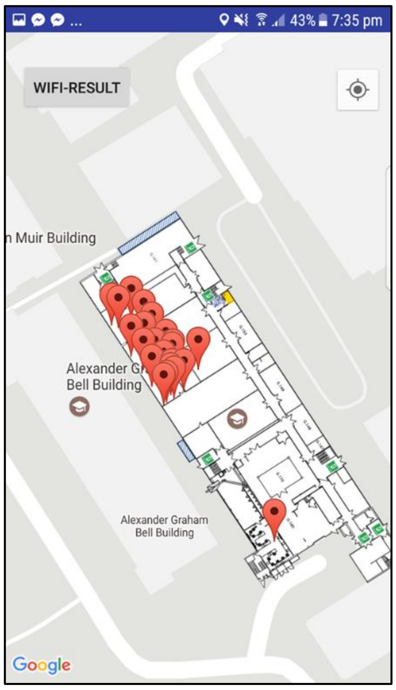
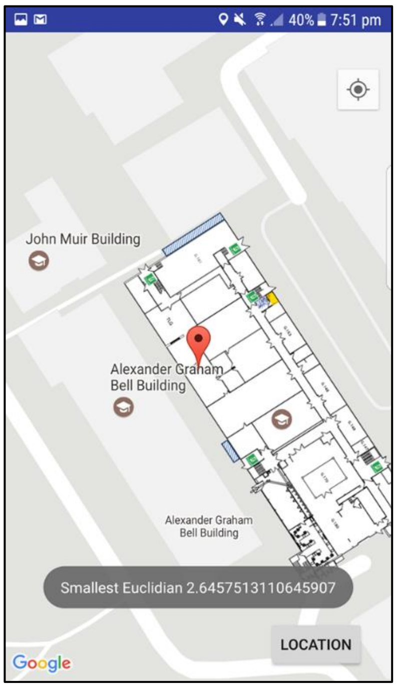


Figure 1.2: Training Activity Window



John Muir Building
Alexander Graham Bell Building
Alexander Graham Bell Building
Smallest Euclidian 2.6457513110645907
LOCATION
Google

Figure 1.3: Positioning Activity Window

1.1 Introduction

Android OS provides a location service application through Google Map. But indoor positioning is quite inaccurate [1]. This Positioning Application uses stored Wi-Fi received signal strength to provide accurate positioning in indoor scenario and GPS to provide an accurate location in an outdoor scenario. A number of other extra features are available in this application, such as battery management information, image capture, compass and linear acceleration information.

1.2 Operation of the Application

The application uses Wi-Fi signal and GPS to provide accurate location both indoor and outdoor scenario. Therefore, the user has to give permission to access location when installing the application on their android devices (with Android 6.0 or higher). The application may ask to activate GPS or Wi-Fi if it is turned off.

1.3 Quick Start-Up

Figure 1.1 shows the main UI of the application. It has a three button that guides the user to different functions of the application. For quick start-up of the application, take following steps for an indoor scenario:

1. Click on the Training button shown on figure 1.1

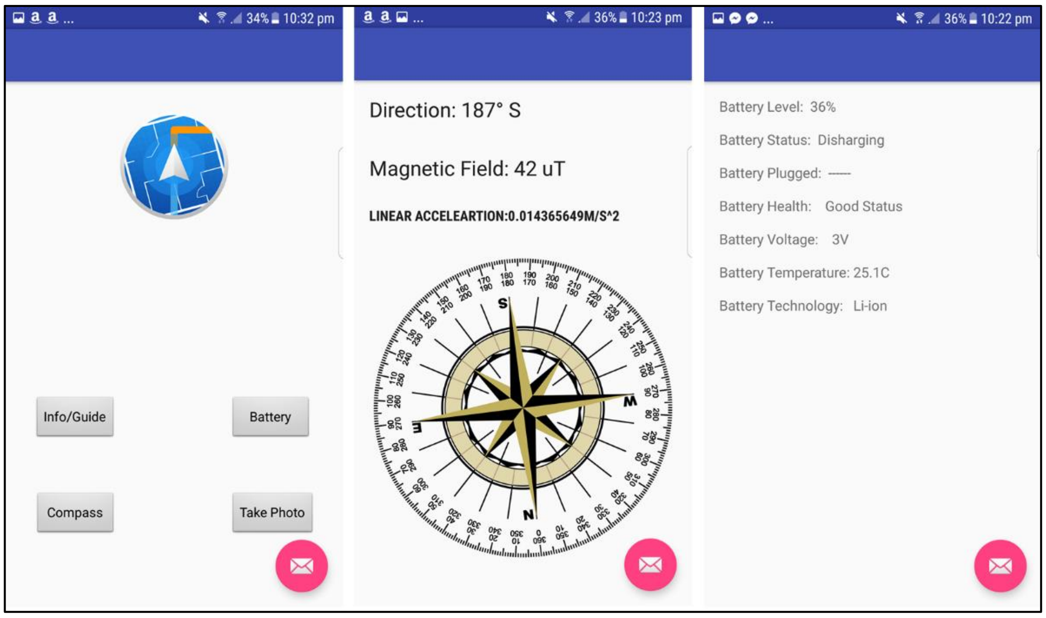
2. Update your current position on the map by clicking the icon (top right corner of the app). It will zoom into your approximate current location shown in Figure 1.2.
3. Check if there are any Wi-Fi signal readings in the database by clicking button WIFI-RESULT (top left corner of the app). This will show any past Wi-Fi scans in the database. Figure 1.2 shows past Wi-Fi signal readings in Fleeming Jenkin building.
4. If there are not significant Wi-Fi readings in the database around your location. Take three Wi-Fi scan reading from three different location by clicking on your current location on the map. The pop-up window Location scan finished will show a successful Wi-Fi reading stored in the database.
5. If there is enough Wi-Fi scan reading in the database (like shown in Figure 1.2), go to the Positioning activity by clicking on the Positioning button shown in Figure 1.1.
6. To show your indoor location, click on button LOCATION (bottom right corner of the app). An accurate indoor location will be shown in red marker shown in Figure 1.3. Also, the nearest Wi-Fi router will be shown in Euclidian distance shown in Figure 1.3.

For outdoor location:

1. Make sure the GPS is activated by accepting pop-up GPS activation requests from the app.
2. Click on the Positioning button. Click on the button LOCATION which will provide an accurate outdoor location from GPS.

1.4 Extra Features

Beside from indoor and outdoor positioning functionality, the application offers a number of extra features that might be useful for the user. The extra functions can be accessed through the button Extra (bottom left corner of the app).



Direction: 187° S

Magnetic Field: 42 uT

LINEAR ACCELEARTION:0.014365649M/S^2

Battery Level: 36%

Battery Status: Disharging

Battery Plugged: ----

Battery Health: Good Status

Battery Voltage: 3V

Battery Temperature: 25.1C

Battery Technology: Li-ion

Figure 1.4: Screenshots of Extra function activity window, Compass activity window and Battery activity window

Description of following functionalities:

- Info/Guide: Provides information about the app and general guideline of how to use the application for new users.
- Battery: Provides information of user's phone battery, such as battery level and battery health.
- Compass: Provides accurate compass with magnetic field reading and linear acceleration.
- Take Photo: Provides option to take a photo from the application with longitude and latitude tagged to the captured image.

2. Programmers Guide

This section will help programmers to understand the application interface and functionality. It will also list sensors, application data processing and algorithms used in the application.

2.1 Application Interface and Functionality

Figure 2.1 shows the block level diagram of the application which illustrates application interface. It also shows the communication in arrows between each activity and classes. Main Activity holds three-sub activities called Training, Positioning and Extra activity.

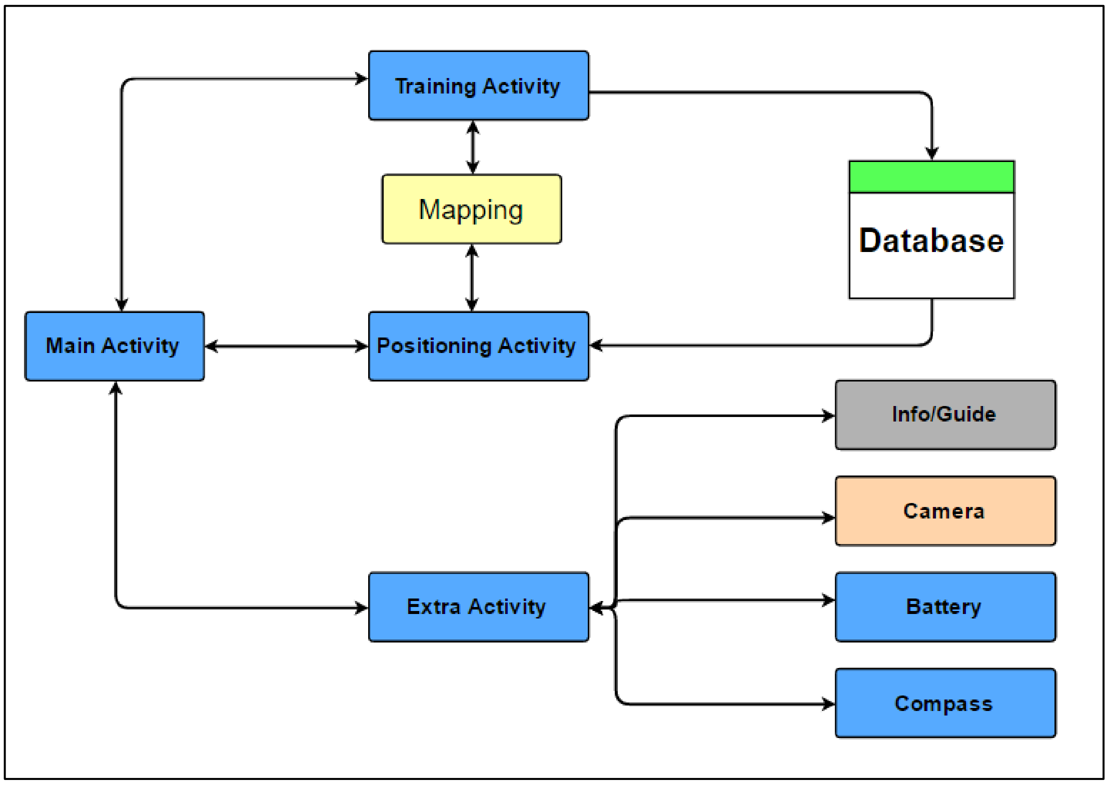


Figure 2.1: Block level diagram of the application

Colour coding are following:

- Activities: Blue
- Classes: Green (for database) and Yellow (for mapping)
- Others: Orange (for Camera API) and Grey (for pop-up text)

2.2 Sensors used

The application uses GPS, Wi-Fi and mobile network provider for the core location function. On top of that, accelerometer and magnetic field sensors provide data reading to the application. Mapping class uses Google map API for the map layout and basic mapping services. Database class uses SQLite to create tables for the Wi-Fi readings. Battery activity uses BatteryManager to provide relative values of battery status.

2.3 Sensor Data processing

Google Map:

The indoor and outdoor location handover is handled using:

```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
bestprovider = locationManager.getBestProvider(getcriteria(), true);
locationManager.requestLocationUpdates(bestprovider, 0, 0, locationListener);
```

Bestprovider uses getcriteria() which is a set of predefined conditions. Bestprovider allows the app to use GPS when it is outdoor and Wi-Fi scanning when it is indoor. LocationManager uses bestprovider and locationListener to efficiently handle the handover between GPS and Wi-Fi scanning.

Google Map marker and interface:

```
mMap.setOnMapClickListener(this);
mMap.setOnMarkerClickListener(this);
```

setOnMapClickListener provides information when the user touches on the Google Map. Using this information setOnMarkerClickListener places marker on the Google map. If the user accepts this marker, it records the longitude and latitude of the marker placed.

Adding Floorplan to Google Map:

```
fleemingjenkins = map.addGroundOverlay(new GroundOverlayOptions()
.image(BitmapDescriptorFactory.fromResource(R.drawable.fleemingjenkins)).bearin
g(57).position(new LatLng(55.922686, -3.172950), 94.2f).anchor(0.018f, 0.8186f)
.transparency(0f));
```

The addGroundOverlay adds image of the floorplan on top of Google Map. It requires image and location (longitude and latitude) of the image to be added on the map.

Wi-Fi signal reading:

```
class WifiReceiver extends BroadcastReceiver {
    public void onReceive(Context c, Intent intent){

        List<ScanResult> wifiScanlist = WiFimanager.getScanResults();
        WiFimanager.getScanResults();
        Wifis = new String[wifiScanlist.size()];

        String bssids = "";
        String levels = "";
        for (int i = 0; i < wifiScanlist.size(); i++){
            Wifis[i] = wifiScanlist.get(i).level+"  "+wifiScanlist.get(i).BSSID;
// scan list for signal level and BSSID }
```

BroadcastReceiver and onReceive receive the result for Wi-Fi scanning. Wi-Fi scan results are stored inside a List of ScanResult variable. The data of scan results are retrieved and stored in string array for easy transfer to a list of views. The wifiScanlist.get(i).level is the signal strength of Wi-Fi scan and the wifiScanlist.get(i).BSSID is the router manufacturers unique identity.

Database:

```
public DataBase(Context context) {super(context, DATABASE_NAME, null, 1);}
public void onCreate(SQLiteDatabase db) {
    db.execSQL("create table " + TABLE_NAME + " (BSSID TEXT PRIMARY KEY,
signalLvl TEXT, lat REAL, lng REAL)");
}
public boolean insertData(String ID, String signal, double lat, double lng)
public Cursor getAllData()
```

SQLite Database is an open source database which is local and builds in Android OS. The database is created using SQLite where a table is created using db.execSQL. A new data can be inserted into the database using insertData() and accessed from using Cursor getAllData(). The application database includes BSSID, signal strength, current longitude and latitude.

2.4 Algorithm used

According to a recent study [2], the most reliable way of indoor positioning is scanning Wi-Fi fingerprint which includes BSSID and signal strength. The popularity of this method is that it does not require extra signal transmitting or receiving equipment, but utilises the aid of inherent Android device infrastructure. The study suggests that the app should have an extensive list of Wi-Fi fingerprints in the database. When indoor positioning is requested from the application, the app should scan the current Wi-Fi fingerprints and compare with the database. According to a recent report on indoor positioning [3], the K-nearest neighbour's algorithm is the best-suited method. Following code implements the K-nearest neighbour:

```
        for (int k = 0; k < testCount; k++){
            for (int m = 0; m < signalsCount; m++) {
                if (testBssids[k].equals(trainBssids[m])) {
                    sum += Math.pow(testSignals[k] - trainSignals[k], 2);
                    count++;}} }
```

Wi-Fi signal strength is treated as the metric distance between the user and the router/transmitter. This code searches the database, created in the Training process, and selects the points which fit the best compared to the currently received power signals.

```
for (int k = 0; k < signalsCount; k++){
    trainSignals[k] = Integer.parseInt(splitSignals[k]);
}
```

To select the best fitting points, the smallest Euclidean distance of powers is used, using the powers as metric distances. The smallest Euclidean distance of power is calculated using:

```
                sum += Math.pow(testSignals[k] - trainSignals[k], 2);
```

References

[1] Schutzberg, A. (2017). *Ten Things You Need to Know About Indoor Positioning*. [online] Directions Magazine. Available at: http://www.directionsmag.com/entry/10-things-you-need-to-know-about-indoor-positioning/324602 [Accessed 9 Apr. 2017].

[2] Jia, M., Yang, Y. and Kuang, L. (2017). *An Indoor and Outdoor Seamless Positioning System Based on Android Platform - IEEE Xplore Document*. [online] Ieeexplore.ieee.org. Available at: http://ieeexplore.ieee.org/document/7847066/ [Accessed 9 Apr. 2017].

[3] DIAZ VELASCO, A. and MELLADO DELGADO, S. (2017). *Indoor Positioning using the Android Platform*. [online] BTH – BLEKINGE INSTITUTE OF TECHNOLOGY. Available at: https://www.divaportal.org/smash/get/diva2:829725/FULLTEXT01.pdf [Accessed 9 Apr. 2017].