Software Engineering Large Practical: Maps and location services

Stephen Gilmore
School of Informatics

October 3, 2017

Contents

Android software development

Android software development

- Android software development is supported by well-documented software APIs.
- It is also supported by many good tutorials with Android code snippets and example projects showing how APIs are used.
- **In this practical, you are encouraged to make use of example code which you find available in tutorials and Android code samples, and to include libraries as needed.**
- This is *re-use*, which is a good thing, not *plagiarism*, which is a bad thing.
- Please cite the sources which you used in developing your app.

The nature of software development

- The practice of software development has changed markedly over the last five to ten years.
- The existence of sites such as stackoverflow.com has made accessing relevant experience in application development much easier.
- The existence of open-source repositories such as GitHub has made accessing working example software projects much easier.
- The fact that both of these archives are searchable by Google makes them more accessible again.
- For Android specifically, the existence of github.com/googlesamples/ provides many high-quality examples of Android projects.

Android software development in practice

1. Investigate relevant Android concepts using tutorials and documentation from developer.android.com/training/
2. Investigate code samples which provide examples of these concepts in use. Download and try these.
3. Identify relevant libraries and services to import into your project. Install these.
4. Add code to your project based on the concepts learned and example code seen, modifying as necessary.

Using Google maps

Adding Google maps to your app

- Google Maps are provided as a remote service which you access via the Google Maps API.
- Access to some Maps APIs are charged, but the Google Maps Android API currently offers up to 25,000 API requests per day for free. The apps which we create on this course will make many fewer requests than this.
- API keys allow access to the Google servers. They associate requests with a particular app to enforce request limits.
- When you add a new Maps activity to your app an XML document is also created to store your Google Maps API key. This XML document is res/values/google_maps_api.xml.

Resource file res/values/google_maps_api.xml

```
<resources>
    <!--
        TODO: Before you run your application, you need a Google Maps
            API key. To get one, follow this link, follow the directions and
            press "Create" at the end:

        https://console.developers.google.com/flows/enableapi?apiid=maps_android_b
        ...
        Once you have your key (it starts with "AIza"), replace the
            "google_maps_key" string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve"
        translatable="false">YOUR_KEY_HERE</string>
</resources>
```

Generated onCreate method

```
public class MapsActivity extends FragmentActivity implements
      OnMapReadyCallback {
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment
        SupportMapFragment mapFragment =
                (SupportMapFragment) getSupportFragmentManager()
                  .findFragmentById(R.id.map);
        // Get notified when the map is ready to be used. Long-running
              activities are performed asynchronously in order to keep the user
              interface responsive
        mapFragment.getMapAsync(this);
    }
    ...
}
```

Generated onMapReady callback

```
public class MapsActivity extends FragmentActivity implements
      OnMapReadyCallback {

    private GoogleMap mMap;
    ...
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney, Australia, and move the camera.
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new
            MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
        // Also available: newLatLngZoom(sydney, 15)
    }
}
```

Making the user's location visible

```
@Override
public void onMapReady(GoogleMap googleMap) {
    ...
    // Also available: newLatLngZoom(sydney, 15)

    try {
        // Visualise current position with a small blue circle
        mMap.setMyLocationEnabled(true);
    } catch (SecurityException se) {
        System.out.println("Security exception thrown [onMapReady]");
    }
    // Add ''My location'' button to the user interface
    mMap.getUiSettings().setMyLocationButtonEnabled(true);
}
```

Making the user's location visible

```
@Override
public void onMapReady(GoogleMap googleMap) {
    ...
    // Also available: newLatLngZoom(sydney, 15)

    try {
        // Visualise current position with a small blue circle
        mMap.setMyLocationEnabled(true);
    } catch (SecurityException se) {
        System.out.println("Security exception thrown [onMapReady]");
    }
    // Add ''My location'' button to the user interface
    mMap.getUiSettings().setMyLocationButtonEnabled(true);
}
```

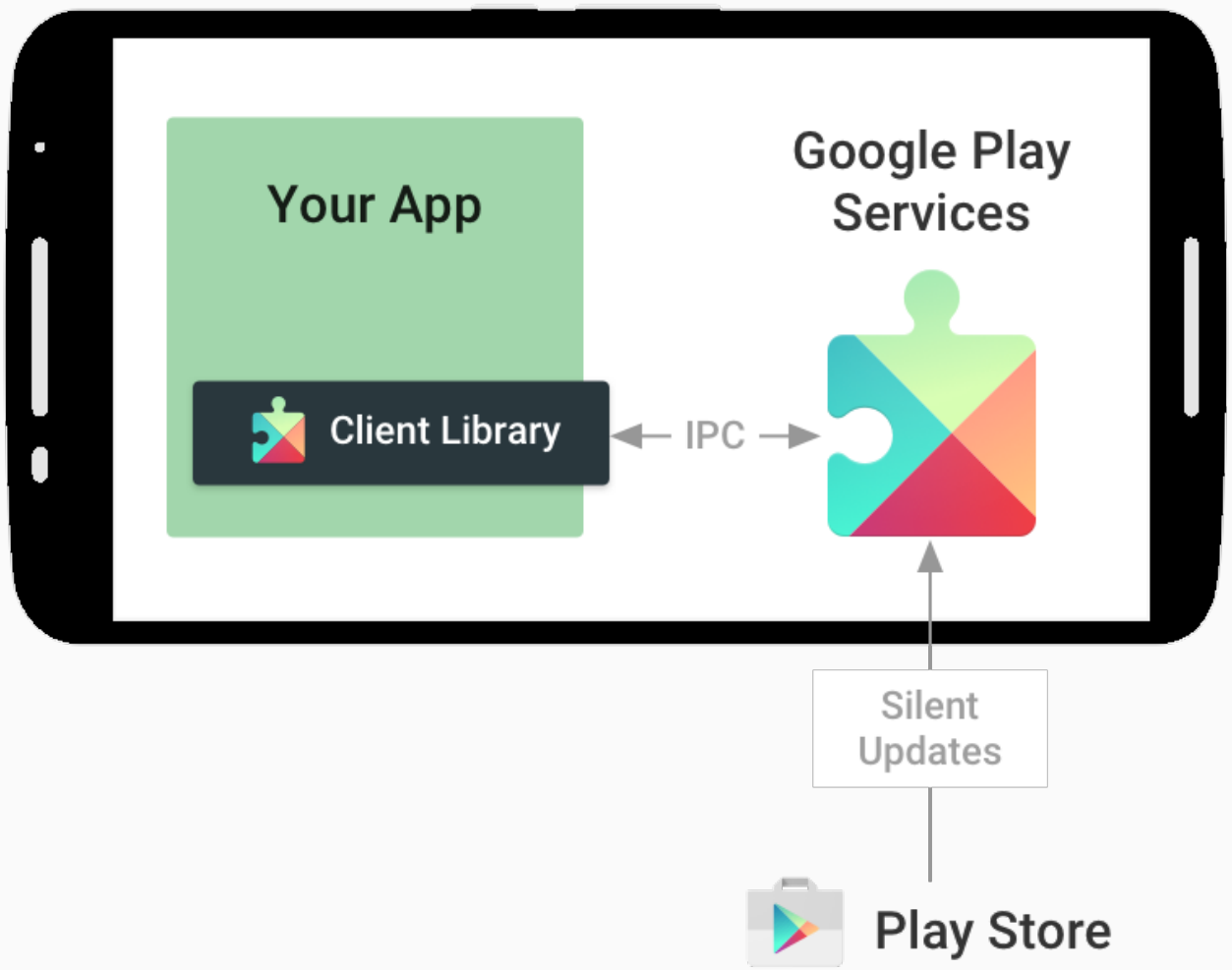
"My location" button — 

Using location services

Using location services

- Location-awareness is a core feature of apps and services for mobile devices. Services of all kinds can be enhanced with location-awareness (e.g. a search app providing the option to "find restaurants *near me*").
- The Google Play Services location APIs in the package `com.google.android.gms.location` are the preferred way of adding location awareness to your app.
- Google Play Services have a distinguished status within Android apps because they can be updated directly from Google Play (Google's "app store") and are invoked by inter-process communication from a client library in your app.

Google Play Services



From https://developers.google.com/android/guides/overview

build.gradle (Module: app)

To add location services to your app you need to add this dependency to your Gradle build file.

```
...
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:26.+'
    compile 'com.google.android.gms:play-services-maps:11.0.4'
    compile 'com.google.android.gms:play-services-location:11.0.4'
    testCompile 'junit:junit:4.12'
}
```

Getting permission to access locations

- Apps that use location services must request permission to access the user's location using `ACCESS_COARSE_LOCATION` or `ACCESS_FINE_LOCATION`.
- For our purposes, `ACCESS_FINE_LOCATION` is the right choice.
- Permission is requested with the `uses-permission` element in your app manifest (`AndroidManifest.xml`).

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="uk.ac.ed.inf.simplemapsactivity" >

  <uses-permission
      android:name="android.permission.ACCESS_FINE_LOCATION" />
  <application
  ...
  </application>
</manifest>
```

Retrieving the current location

- Using the Google Play services location APIs, your app can request the last known location of the user's device.
- Google Play services are part of Google Mobile Services (GMS).
- We will make use of
  - com.google.android.gms.common.ConnectionResult
  - com.google.android.gms.common.api.GoogleApiClient
  - com.google.android.gms.location.LocationListener
  - com.google.android.gms.location.LocationRequest
  - com.google.android.gms.location.LocationServices

(Not) Using the connectionless API

- **Note:** some of the code examples which follow use deprecated methods, which could be considered bad style, but the new methods (using the connectionless API) appear not to work with the Android emulator.
- We would rather have code which works than not, so we will use some deprecated methods.

Class structure

```
public class MapsActivity
        extends FragmentActivity
        implements OnMapReadyCallback,
            GoogleApiClient.ConnectionCallbacks,
            GoogleApiClient.OnConnectionFailedListener,
            LocationListener{

    private GoogleMap mMap;
    private GoogleApiClient mGoogleApiClient;
    private final int PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION
        =1;
    private boolean mLocationPermissionGranted = false;
    private Location mLastLocation;
    private static final String TAG = "MapsActivity";
    ...
}
```

Adding to `onCreate()`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Long-running activities are performed asynchronously in order to
        keep the user interface responsive
    mapFragment.getMapAsync(this);

    // Create an instance of GoogleAPIClient.
    if (mGoogleApiClient == null) {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API)
            .build();
    }
}
```

Activity `onStart()` and `onStop()`

```
@Override
protected void onStart() {
    super.onStart();
    mGoogleApiClient.connect();
}

@Override
protected void onStop() {
    super.onStop();
    if (mGoogleApiClient.isConnected()) {
        mGoogleApiClient.disconnect();
    }
}
```

Creating a location request

```
protected void createLocationRequest() {
    // Set the parameters for the location request
    LocationRequest mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(5000); // preferably every 5 seconds
    mLocationRequest.setFastestInterval(1000); // at most every second
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    // Can we access the user's current location?
    int permissionCheck = ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION);
    if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(
                mGoogleApiClient, mLocationRequest, this);
    }
}
```

Get the last known location of a device

```
@Override
public void onConnected(Bundle connectionHint) {
    try { createLocationRequest(); }
    catch (java.lang.IllegalStateException ise) {
        System.out.println("IllegalStateException thrown [onConnected]");
    }
    // Can we access the user's current location?
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION) ==
              PackageManager.PERMISSION_GRANTED) {
        mLastLocation =
              LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
    } else {
        ActivityCompat.requestPermissions(this,
         new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
               PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
    }
}
```

Issues in getting the last known location of a device

- A call to FusedLocationApi.getLastLocation may return null.
- This happens if the GoogleApiClient passed as a parameter is not connected.
- It is always necessary to check that the Location object returned is not null.

Being informed that the user's location has changed

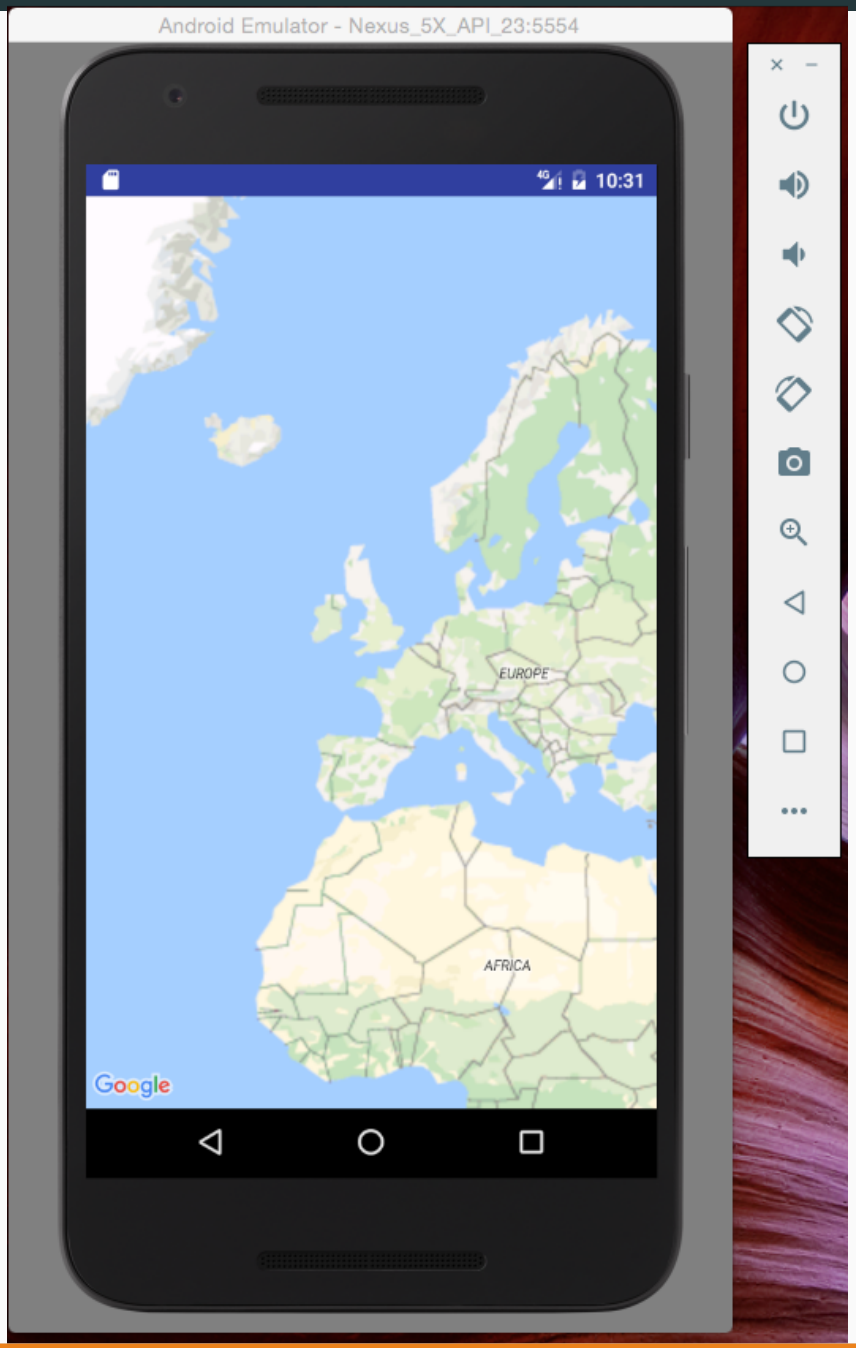
```
@Override
public void onLocationChanged(Location current) {
    System.out.println(
            " [onLocationChanged] Lat/long now (" +
                String.valueOf(current.getLatitude()) + "," +
                String.valueOf(current.getLongitude()) + ")"
    );
    // Do something with current location
    ...
}
```

Connection suspended or connection failed

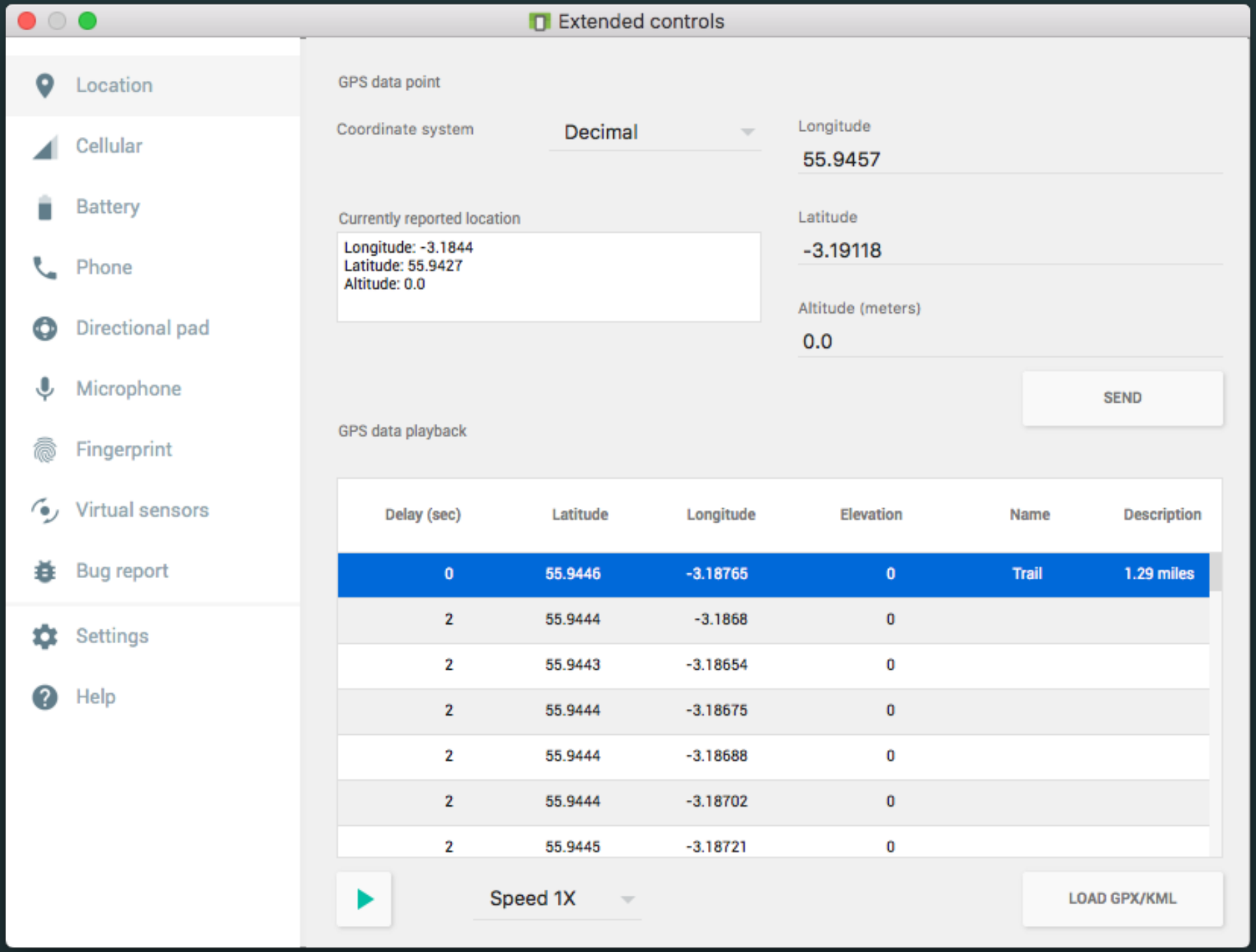
```
@Override
public void onConnectionSuspended(int flag) {
    System.out.println(" >>>> onConnectionSuspended");
}

@Override
public void onConnectionFailed(ConnectionResult result) {
    // An unresolvable error has occurred and a connection to Google APIs
    // could not be established. Display an error message, or handle
    // the failure silently
    System.out.println(" >>>> onConnectionFailed");
}
```

Testing location-based apps

Testing location-based apps with the Android emulator



Click on '...' to access extended controls


Extended controls
Location
Cellular
Battery
Phone
Directional pad
Microphone
Fingerprint
Virtual sensors
Bug report
Settings
Help
GPS data point
Coordinate system
Decimal
Longitude
55.9457
Currently reported location
Longitude: -3.1844
Latitude: 55.9427
Altitude: 0.0
Latitude
-3.19118
Altitude (meters)
0.0
SEND
GPS data playback
Delay (sec)
Latitude
Longitude
Elevation
Name
Description
0
55.9446
-3.18765
0
Trail
1.29 miles
2
55.9444
-3.1868
0
2
55.9443
-3.18654
0
2
55.9444
-3.18675
0
2
55.9444
-3.18688
0
2
55.9444
-3.18702
0
2
55.9445
-3.18721
0
Speed 1X
LOAD GPX/KML

Latitude: 55.9427
Altitude: 0.0

Altitude (meters)

0.0

SEND

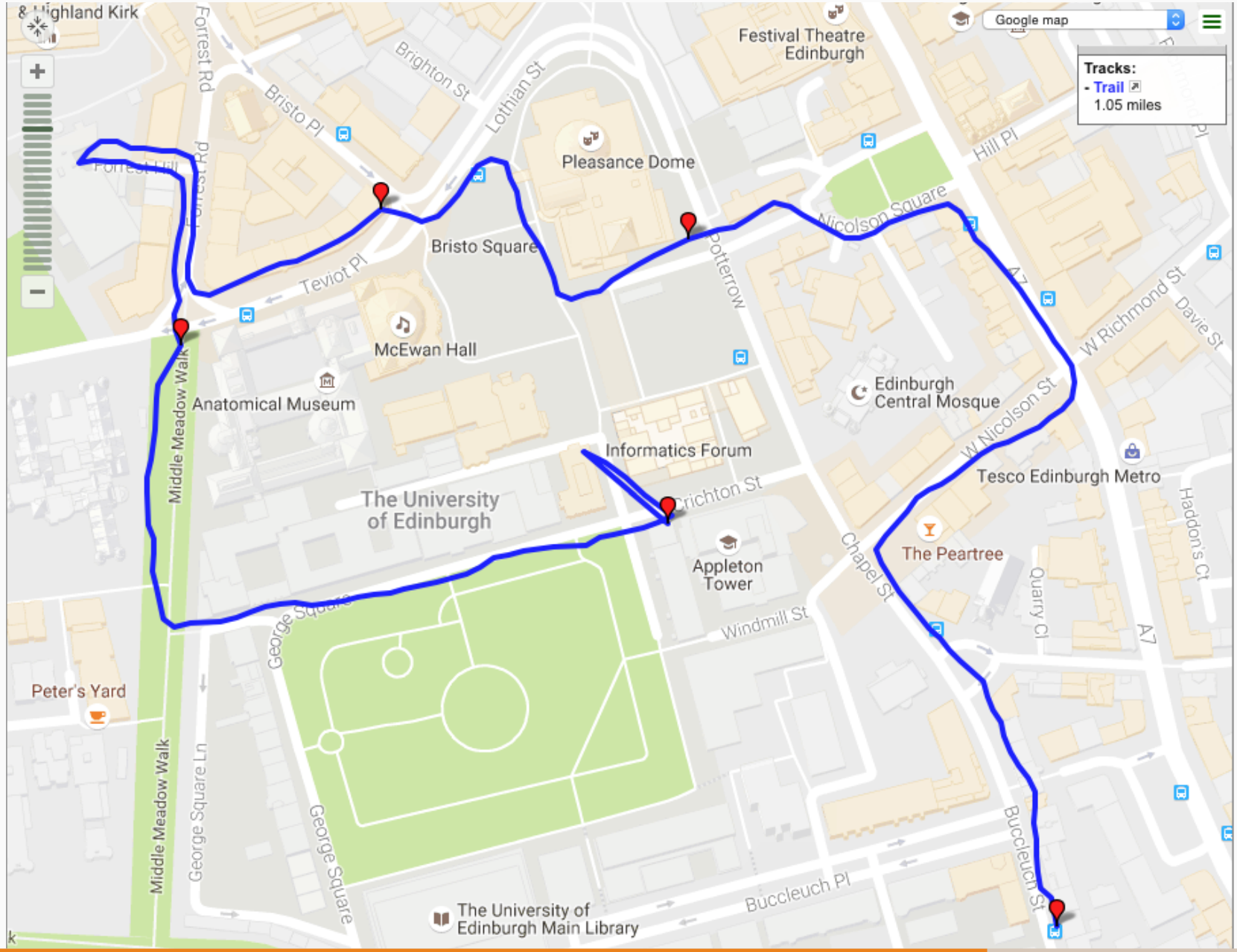
GPS data playback

Delay (sec)	Latitude	Longitude	Elevation	Name	Description
0	55.9446	-3.18765	0	Trail	1.29 miles
2	55.9444	-3.1868	0		
2	55.9443	-3.18654	0		
2	55.9444	-3.18675	0		
2	55.9444	-3.18688	0		
2	55.9444	-3.18702	0		
2	55.9445	-3.18721	0		

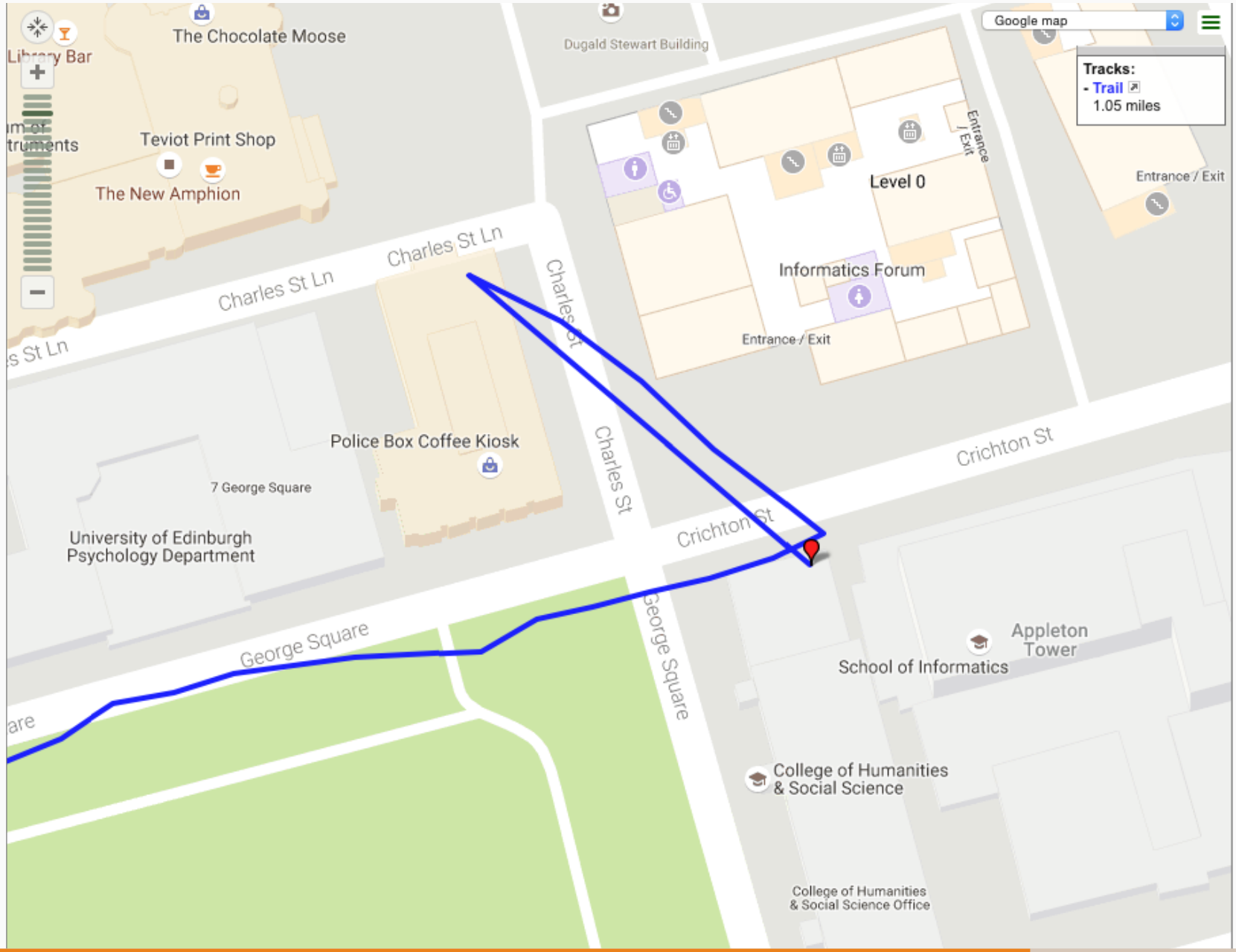
Speed 1X

LOAD GPX/KML

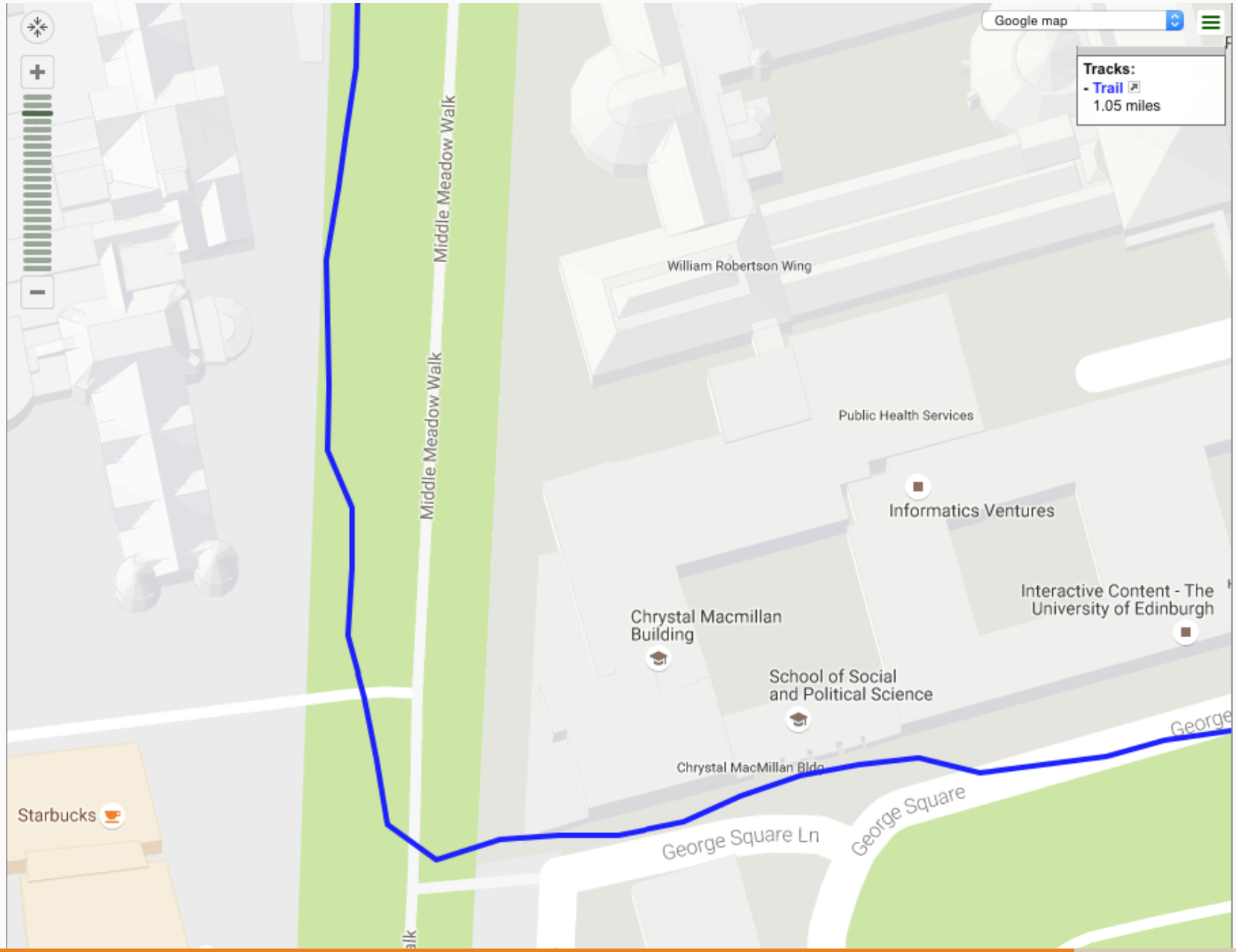
Trace data for testing (using `gpsvisualizer.com`)



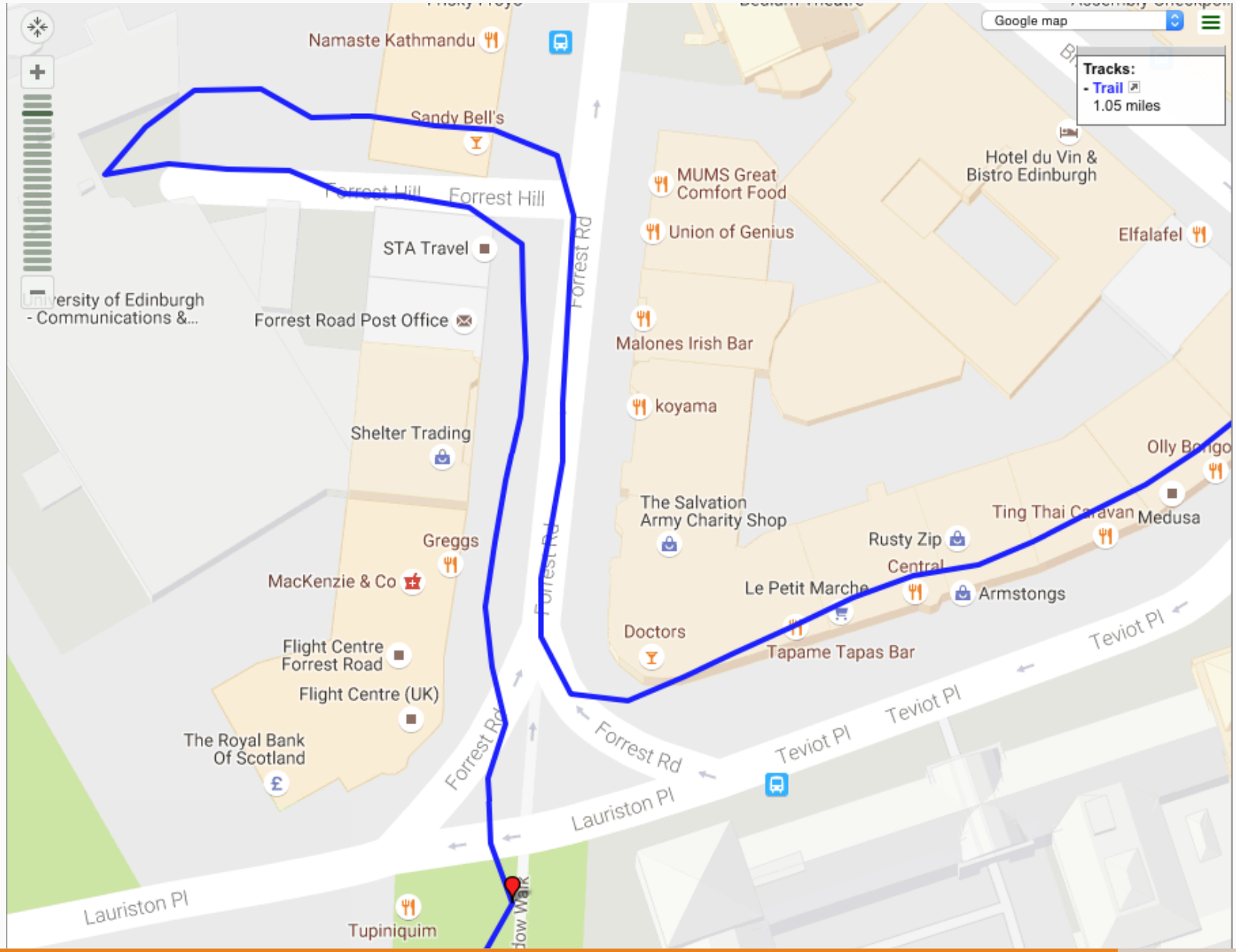
Initial GPS jitter



Some measurement errors seen

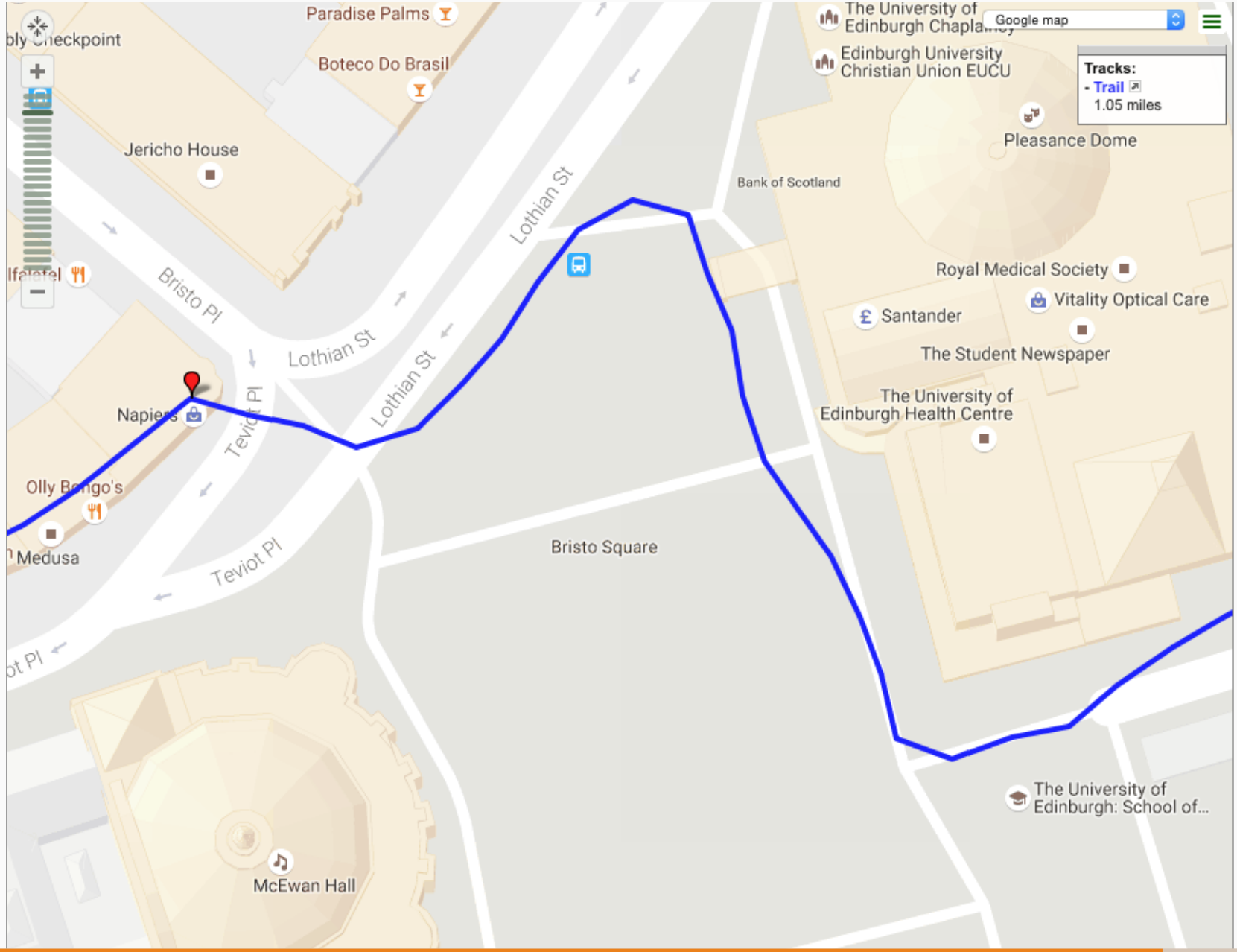


GPS measurement errors increase near tall buildings

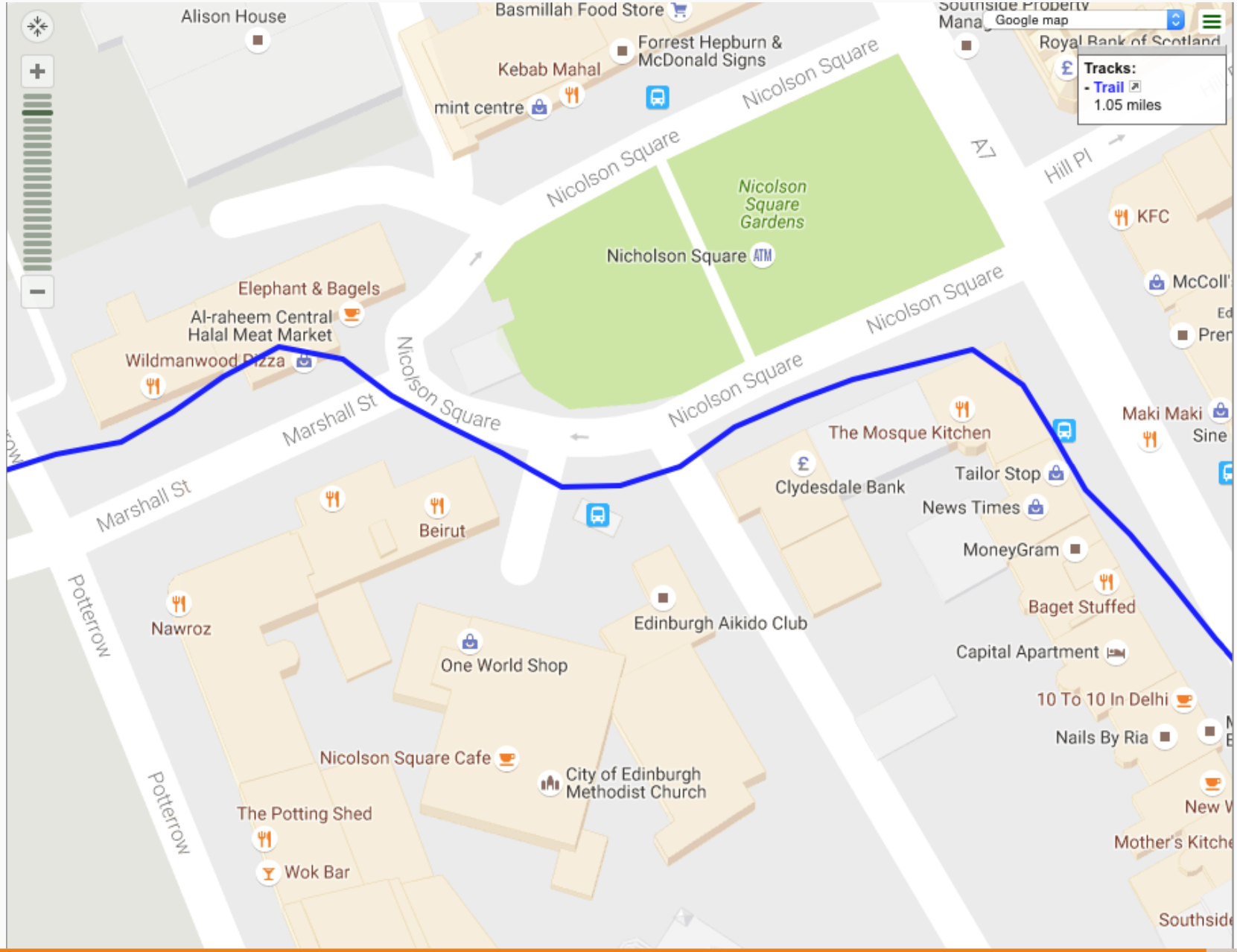


Trail continues

Shot 2016-10-05 at 10.40.14.png

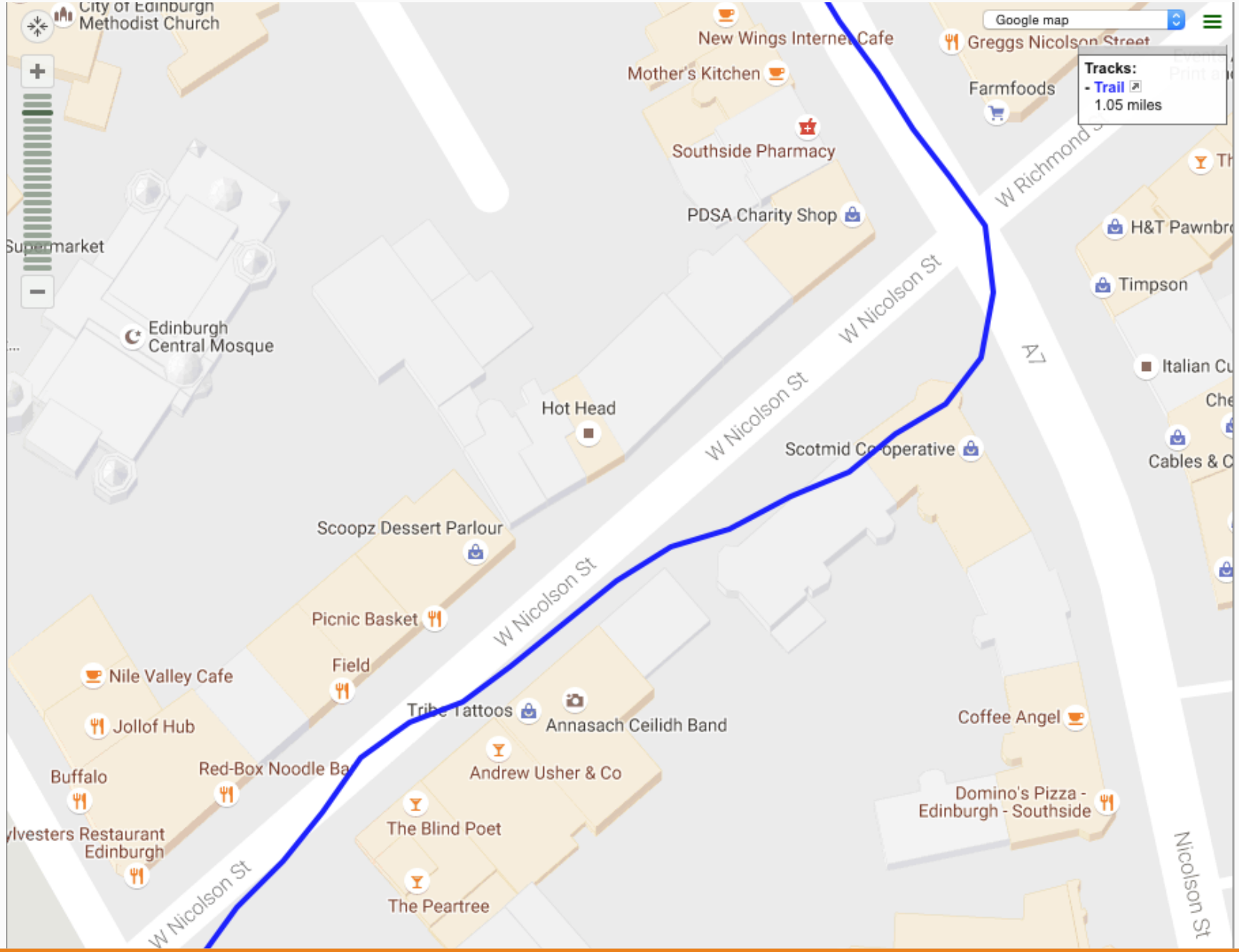


Trail continues



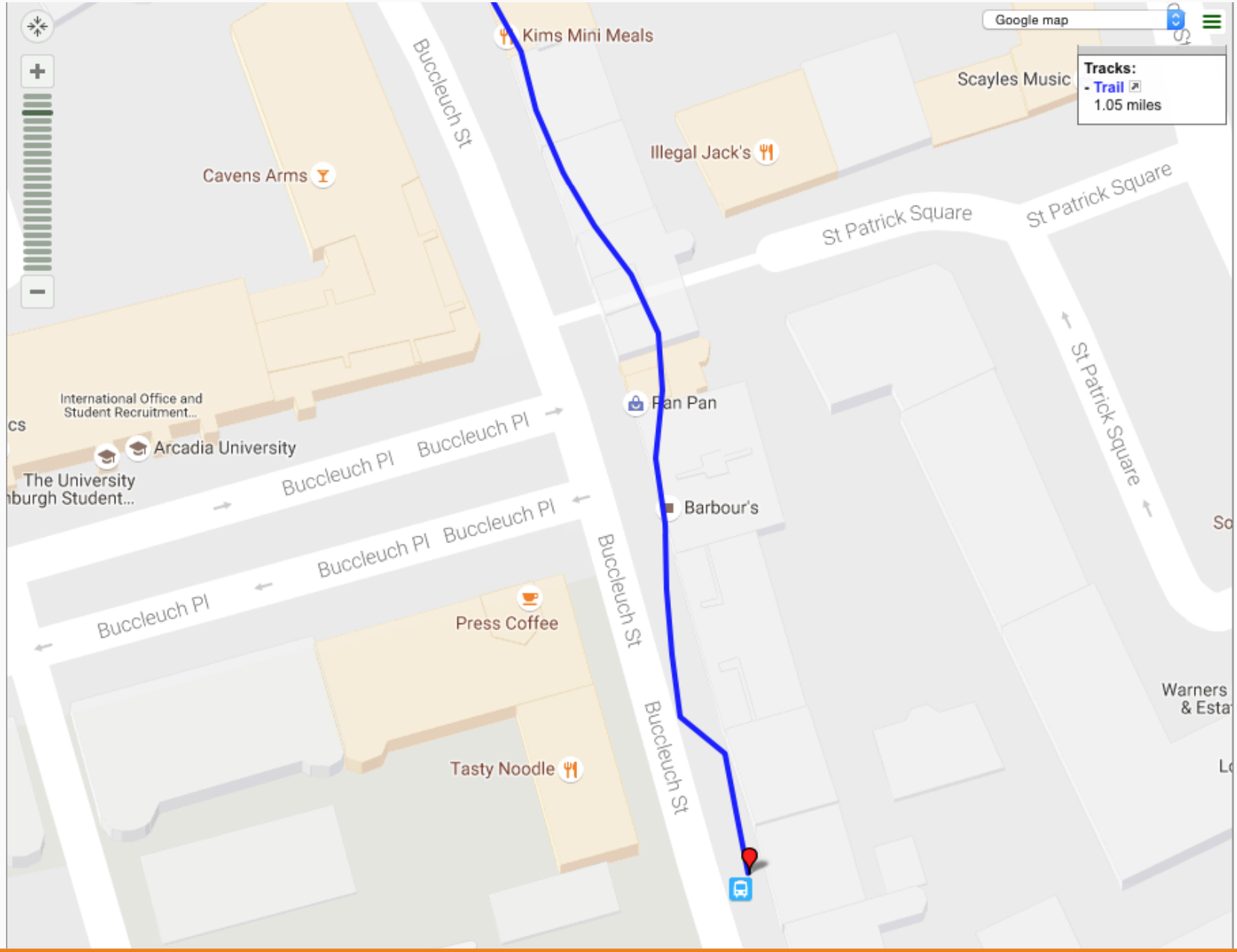
Trail continues

Shot 2016-10-05 at 10.40.49.png



Trail ends

Shot 2016-10-05 at 10.41.08.png



Concluding remarks

- This GPS trace was obtained under near-ideal weather conditions (clear sky, no cloud cover) and still contains a number of measurement errors.
- There is nothing that we can do to fix these errors, we simply take the position as reported as being the location of the user.
- The Android emulator allows us to load and replay GPS data stored in KML format. This is a useful feature for testing.