Computer Science Large Practical: Maps and location services

Stephen Gilmore
School of Informatics

November 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.

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

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
    ...
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_maps)
        // Obtain the SupportMapFragment and get notified when the map
             is ready to be used.
        val mapFragment = supportFragmentManager
                .findFragmentById(R.id.map) as SupportMapFragment
        // 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

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback {

    private lateinit var mMap: GoogleMap
    ...
    override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap

        // Add a marker in Sydney and move the camera
        val sydney = LatLng(−34.0, 151.0)
        mMap.addMarker(MarkerOptions().position(sydney).title("Marker in
            Sydney"))
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
        // Also available: newLatLngZoom(sydney, 15)
    }
}
```

A "private lateinit var" in Kotlin behaves like a field in Java

Making the user's location visible

```
override fun onMapReady(googleMap: GoogleMap) {
    ...
    // Also available: newLatLngZoom(sydney, 15)

    try {
        // Visualise current position with a small blue circle
        mMap.isMyLocationEnabled = true
    } catch (se : SecurityException) {
        println("Security exception thrown [onMapReady]")
    }

    // Add "My location" button to the user interface
    mMap.uiSettings.isMyLocationButtonEnabled = true
}
```

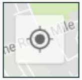
Kotlin has exceptions and try .. catch .. finally

Making the user's location visible

```
override fun onMapReady(googleMap: GoogleMap) {
    ...
    // Also available: newLatLngZoom(sydney, 15)

    try {
        // Visualise current position with a small blue circle
        mMap.isMyLocationEnabled = true
    } catch (se : SecurityException) {
        println("Security exception thrown [onMapReady]")
    }

    // Add "My location" button to the user interface
    mMap.uiSettings.isMyLocationButtonEnabled = true
}
```

Kotlin has exceptions and try .. catch .. finally

"My location" button —

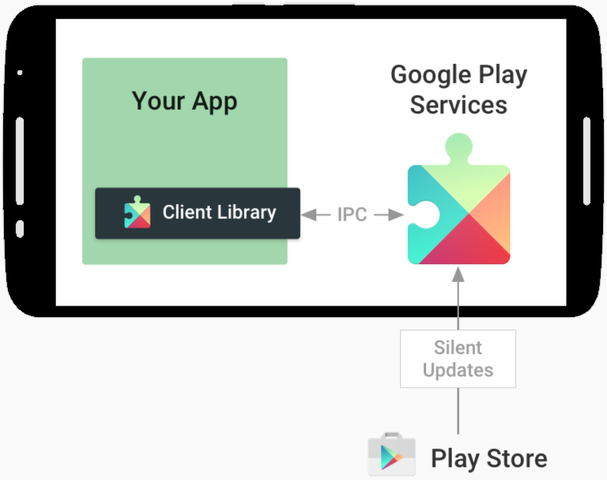


Using location services

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

```
class MapsActivity : AppCompatActivity(), OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationListener {

    private lateinit var mMap: GoogleMap
    private lateinit var mGoogleApiClient: GoogleApiClient
    val permissionsRequestAccessFineLocation = 1
    var mLocationPermissionGranted = false
    // getLastLocation can return null, so we need the type "Location?"
    private var mLastLocation : Location? = null
    val tag = "MapsActivity"
    ...
}
```

Location? means either a Location or `null`

Adding to `onCreate()`

```
override fun onCreate(savedInstanceState: Bundle?) {
    ....
    // 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)

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

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

```
override fun onStart() {
    super.onStart()
    mGoogleApiClient.connect()
}

override fun onStop() {
    super.onStop()
    if (mGoogleApiClient.isConnected) {
        mGoogleApiClient.disconnect()
    }
}
```

Creating a location request

```
fun createLocationRequest() {
    // Set the parameters for the location request
    val mLocationRequest = LocationRequest()
    mLocationRequest.interval = 5000 // preferably every 5 seconds
    mLocationRequest.fastestInterval = 1000 // at most every second
    mLocationRequest.priority =
         LocationRequest.PRIORITY_HIGH_ACCURACY

    // Can we access the user's current location?
    val 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 (1/2)

```
override fun onConnected(connectionHint : Bundle?) {
    try {
        createLocationRequest()
    } catch (ise : IllegalStateException) {
        println("[$tag] [onConnected] IllegalStateException thrown")
    }

    // Can we access the user's current location?
    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION) ==
            PackageManager.PERMISSION_GRANTED) {

            val api = LocationServices.FusedLocationApi
            mLastLocation = api.getLastLocation(mGoogleApiClient)
```

Get the last known location of a device (2/2)

```
            // Caution: getLastLocation can return null
            if (mLastLocation == null) {
                println("[$tag] Warning: mLastLocation is null")
            }
    } else {
        ActivityCompat.requestPermissions(this,
            arrayOf(android.Manifest.permission.ACCESS_FINE_LOCATION),
            permissionsRequestAccessFineLocation)
    }
}
```

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 fun onLocationChanged(current : Location?) {
    if (current == null) {
        println("[$tag] [onLocationChanged] Location unknown")
    } else {
        println("""[$tag] [onLocationChanged] Lat/long now
            (${current.latitude},
             ${current.longitude})"""
        )
        // Do something with current location
        ...
    }
}
```

Multi-line strings begin with triple quotes in Kotlin

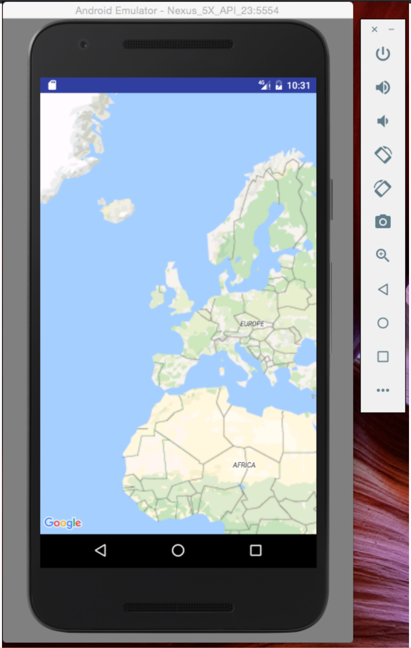
The ${. . . } syntax embeds expressions in strings

Connection suspended or connection failed

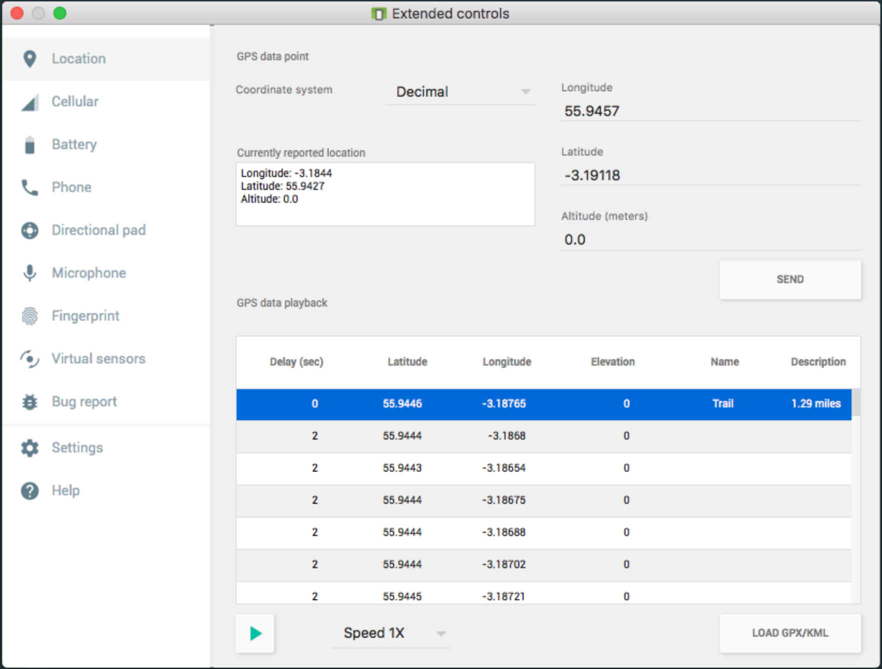
```
override fun onConnectionSuspended(flag : Int) {
    println(" >>>> onConnectionSuspended")
}

override fun onConnectionFailed(result : ConnectionResult) {
    // An unresolvable error has occurred and a connection to Google APIs
    // could not be established. Display an error message, or handle
    // the failure silently
    println(" >>>> onConnectionFailed")
}
```

Testing location-based apps

Testing location-based apps with the Android emulator



Click on '...' to access extended controls



GPS data point

Coordinate system: Decimal

Longitude: 55.9457

Latitude: -3.19118

Altitude (meters): 0.0

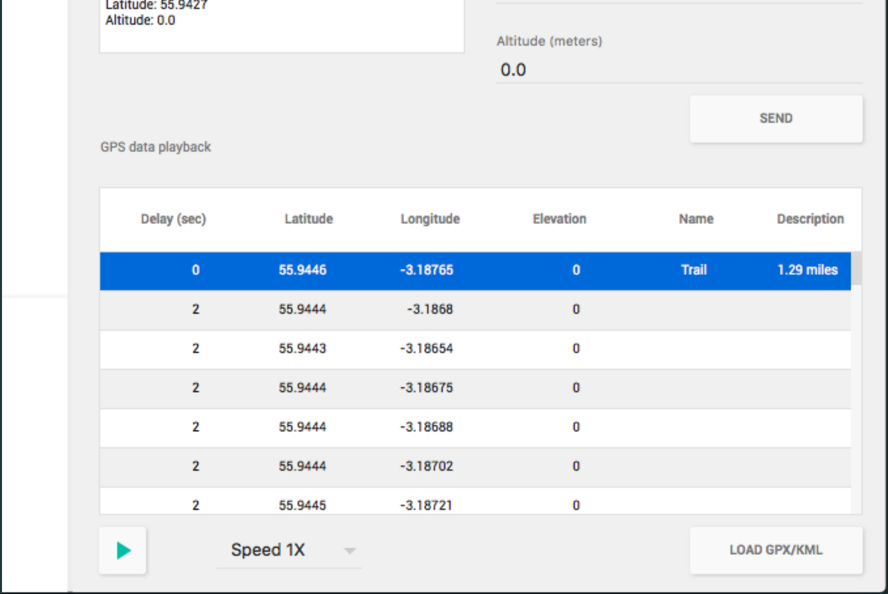
Currently reported location

Longitude: -3.1844
Latitude: 55.9427
Altitude: 0.0

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


Latitude: 55.9427
Altitude: 0.0

Altitude (meters)
0.0

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		

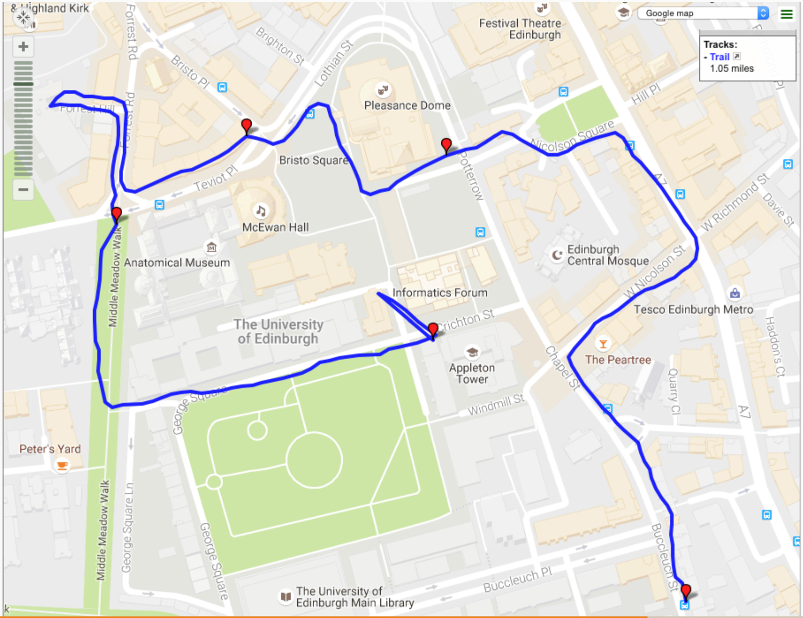
Speed 1X

Errors in GPS readings are to be expected.

- GPS-based location, spatial presence and situation play an increasing role in systems but satellite-based triangulation requires very precise measurement of extremely fast signals.
- Environmental factors impact on the accuracy of this triangulation. Signal bounce off tall buildings can interrupt the GPS signal. Heavy cloud cover, humidity and atmospheric pressure have an impact on measurements.
- Errors in GPS readings are to be expected.

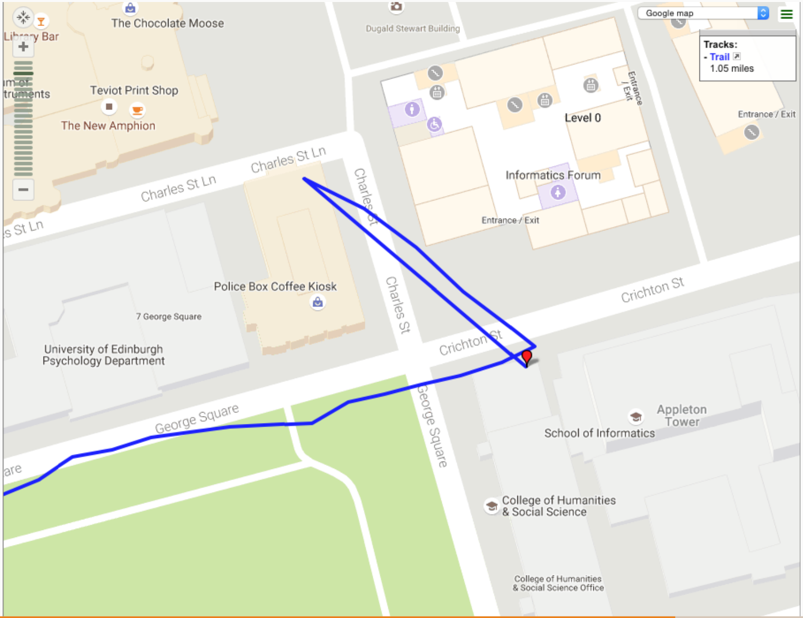
Trace data for testing (using `gpsvisualizer.com`)

Shot 2016-10-05 at 10.38.25.png



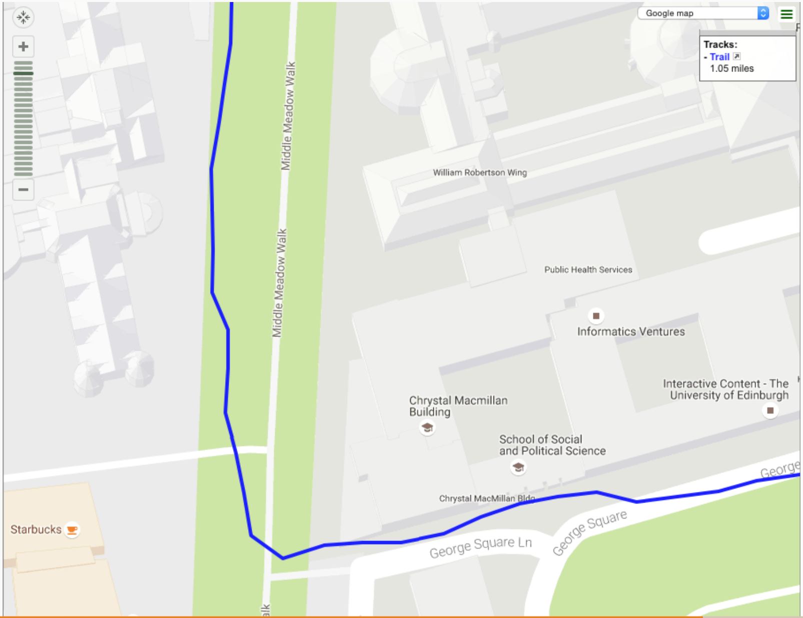
Initial GPS jitter

Shot 2016-10-05 at 10.38.50.png



Some measurement errors seen

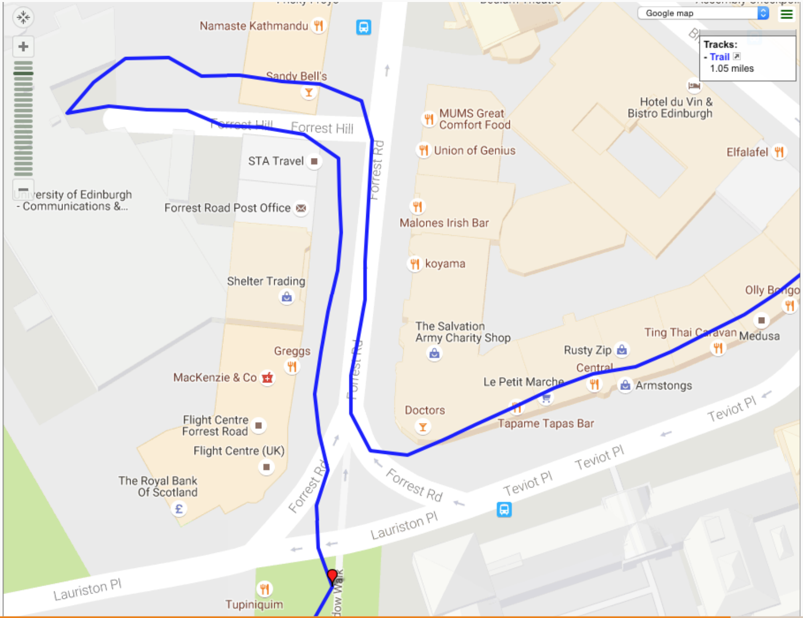
Shot 2016-10-05 at 10.39.10.png



Tracks:
- Trail
1.05 miles

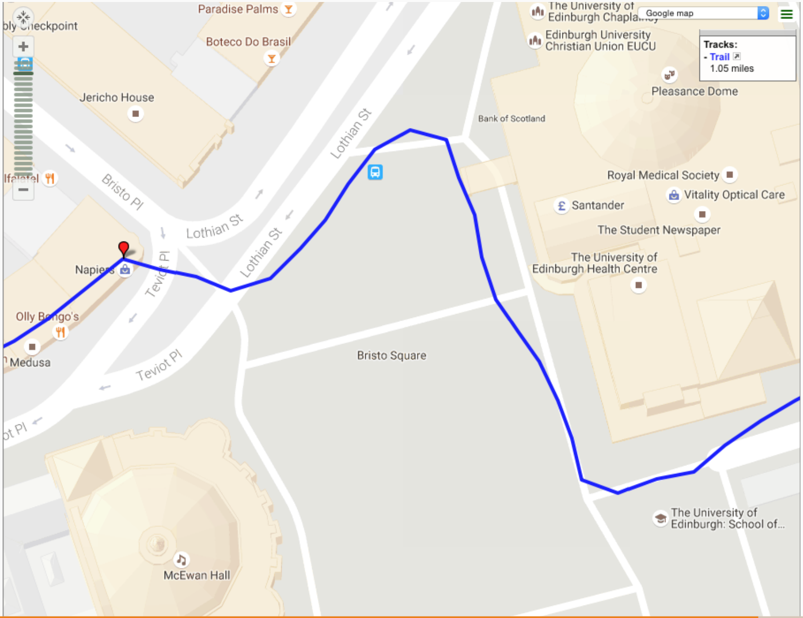
GPS measurement errors increase near tall buildings

Shot 2016-10-05 at 10.39.56.png



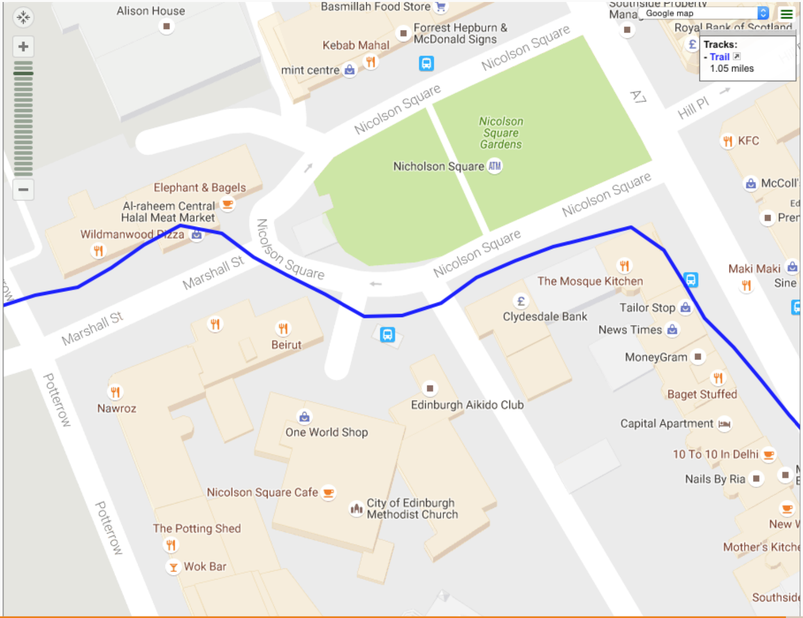
Trail continues

Shot 2016-10-05 at 10.40.14.png



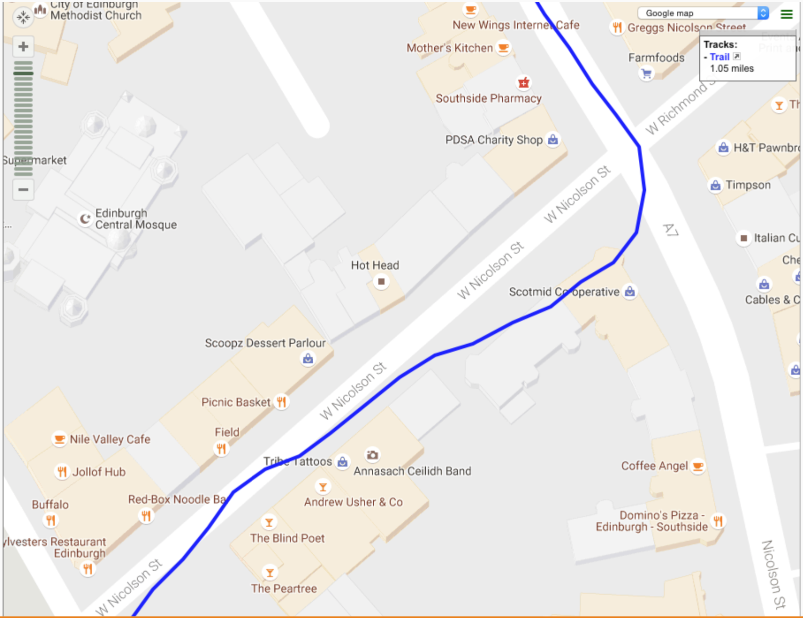
Trail continues

Shot 2016-10-05 at 10.40.32.png



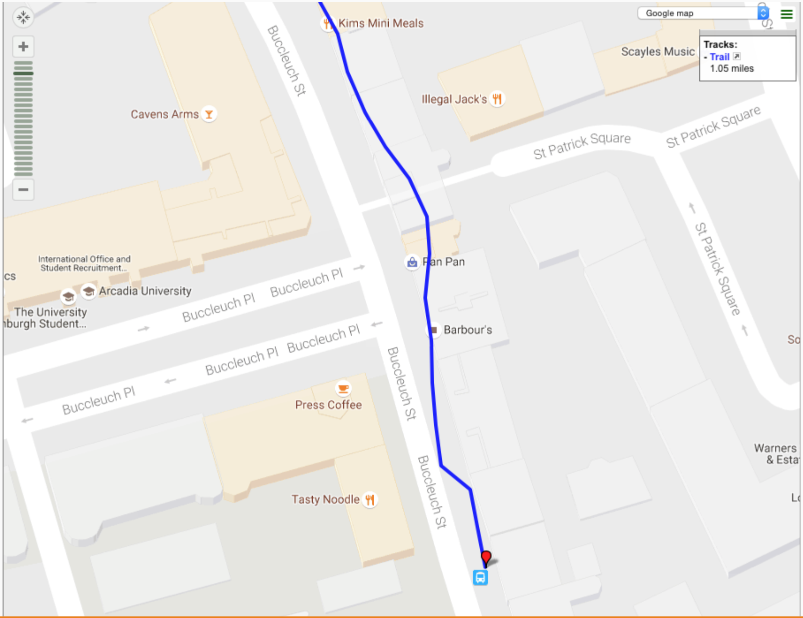
Trail continues

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.

Links

- Android developer tutorials
  - https://developer.android.com/training/
- Android developer tutorial on location services
  - https://developer.android.com/training/location/