1 Location API and Google MAP

1.1 Get the current position

The current position can be obtained through methods: GPS/A-GPS, Cell triangulation, Wifi positions, and IP locations, which offers different accuracy.

1.2 Android Location API

Most Android devices today have a device which allows determining where the device is. This is based on a GPS (Global Positioning System) device. Android provides location API in the package "android.location" which allows determining the current position. You can register a listener to the location manager and receive periodic updates about the current location. In addition it is possible to receive the information if the device enters an area given by a longitude, latitude and radius (proximity alert).

Important Classes:

1. The class "LocationManager" provides access to the location service. This allows to access location provider to register for a location update or setting a proximity alert.
2. The class "LocationListener" can be registered with the "LocationManager" and will receive periodic updates about the location.
3. The class "LocationProvider" is the superclass of the different location providers which deliver the information about the current location.
    a. Network : Wifi-based or IP-based
    b. GPS: GPS or A-GPS

1.3 Google Maps

Google provides also a library in the package "com.google.android.maps" for using Google Maps in Android. Google Maps support is not part of the standard Open Source Platform Android and you require an additional key to use them.

1. The class "MapActivity" is an activity which you normally need to extend. "MapActivity" provides the basic functionality to get a "MapView" which can be used to display the map and takes care of the network communication required for the map.
2. The class "MapController" can be used to interact with the "MapView", e.g. by moving it. A "Geopoint" is a position described via latitude and longitude and the class "Overlay" can be used to drawn on the map, for example position markers.

1.4 Android Emulator

In case you want to use Google Maps or location APIs in your emulator make sure you have created a new device which supports the Google API's. This should be the case if you use the emulator created for the previous lab. During emulator creation select the target Google API's in the version of your SDK.

1.4.1 Set the location on the emulator

1.4.1.1 DDMS to set the location:

- After running your virtual device (you can do it from the virtual device manager tools -> android -> AVD manager)
- Go to tools -> android -> android device monitor
- Select the tab “Emulator Control”
- Location control, select manual tab
- Insert Longitude and Latitude and press send

1.4.1.2 Terminal to set the location

You can use legacy Linux terminal or the one embedded in android studio

- Get the port number of your device (e.g. 5554, usually shown in the top bar of the emulator)
- telnet localhost port_number
- geo fix <longitude> <latitude> (e.g. geo fix 7.04 14.8)

2 Activating the GPS Module [15’]

Important NOTE: To display the location you may need to run Google Maps on the emulator, this will activate the GPS provider.

3 Creating the API key to be able to use google APIs

Since we are going to deal with Google APIs during this lab, it is better to create the APIs key before starting, you will use the debug key and not the release.

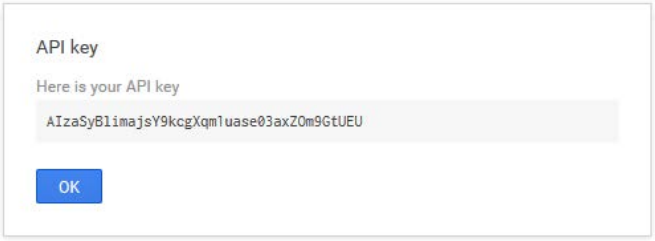
1. Go to https://console.developers.google.com/home/projects (authenticate with your google account if necessary and select Create Project, give to it the name you prefer
2. Once you created the project you should be prompted with the project dashboard, in this screen select “Enable and manage APIs”
3. In the Overview you can click on Enabled APIs, Google Maps API should be not enabled
4. Go back to the tab Google APIs and click on Google Maps Android API.
5. In the following screen select “enable API”

Now we need to create the key

1. Go to credentials
2. Add Credentials -> API key -> Android key
3. Run on your terminal keytool -list -keystore ~/.android/debug.keystore and copy the fingerprint it is prompted, the **password** required to access the file is simply “android”
4. Then go back to the Console and click on Add Package Name and finger print
5. You put as name the package name we are going to use and as fingerprint the one copied before

- Package name : fr.eurecom.android.locationservices
- FINGERPRINT COPIED BEFORE

6. Create

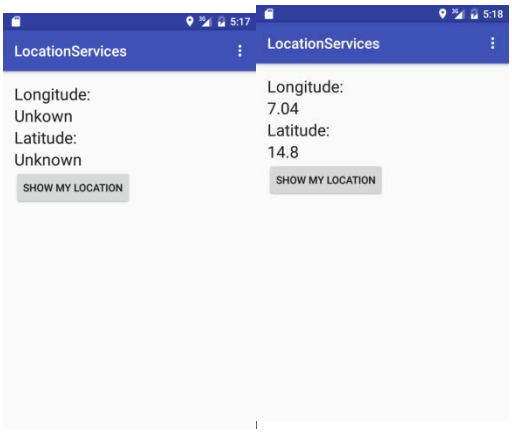


Copy the key in a file in which you can access later and press ok.

Now you have a valid API key to access google maps api.

4 Show Location [1h]

4.1 Create Project



Application name : LocationServices
Package name : fr.eurecom.android.locationservices
Activity Type: Blank
Activity Name: ShowLocation
Min SDK Version : 15
Use the SDK target to Google APIs 23 SDK.
The activity Showlocation **implements LocationListener**.

4.2 Layout and Permissions

Add the following to your main.xml

1. Add 4 “TextView” with “android:id = “@id/TextView01”, “@id/TextView02”, ...)
   a. Set the android to “Longitude, unknown, Latitude, unknown”, respectively.
2. Add a button with “android:id = “@id/Button01”, and android:onClick=”showLocation”
3. Add a “ScrollView”
   a. Add a “TextView” inside the ScrollView and set the android:id= “@id/output”
4. Add the following permission to your manifest.xml
   a. INTERNET, ACCESS_FINE_LOCATION , ACCESS_COARSE_LOCATION

Later, Change the RelativeLayout to the “ LinearLayout” with orientation Horizontal

5.

4.3 Code the Activity

4.3.1 Define class vars

```
protected LocationManager locationManager = null;
private String provider;
Location location;
TextView latitudeField;
TextView longitudeField;
public static final int MY_PERMISSIONS_LOCATION = 0;
```

4.3.2 Check if GPS is enabled at onStart, if not, ask user to enable it

```
@Override protected void onStart(){
	super.onStart();
	locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
	boolean gpsEnabled =
locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
	if(!gpsEnabled){
		Log.i("GPS", "not enabled");
			// Build an alert dialog here that requests the user
			// to enable location services when he clicks over "ok"
			enableLocationSettings();
	 }else{
		Log.i("GPS","enabled");
		}

private  void enableLocationSettings(){
	Intent settingsIntent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
	startActivity(settingsIntent);
	    }
```

4.3.3 Run and check if the alert is working correctly

To check if your alert is properly working simply disable the location service on the emulator you can do this from the settings or simply swiping down from the top of the screen toward the bottom and then disabling the location icon. Once the location is off, the app should ask the user if he is willing to enable it and start correctly the settings intent.

4.3.4 Start the location manager at onCreate

We now start the location manager. When accessing to contacts, camera, and location, it is important to request permissions and ask for the user authorization. Here, we also need to ask the user if it allows to access its own position before using the “getLastKnownLocation. Add the following to the “onCreate” lifecycle method.

```
Criteria criteria = new Criteria();
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
provider = locationManager.getBestProvider(criteria, false);

if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ContextCompat.checkSelfPermission(this.getApplicationContext(),
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED){
       Log.i("Permission: ", "To be checked");
       ActivityCompat.requestPermissions(this,
       new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
       Manifest.permission.ACCESS_COARSE_LOCATION},
       MY_PERMISSIONS_LOCATION);
return;
} else
       Log.i("Permission: ", "GRANTED");

latitudeField = (TextView) findViewById(R.id.TextView02);
longitudeField = (TextView) findViewById(R.id.TextView04);
location = locationManager.getLastKnownLocation(provider);
if (locationManager== null)
locationManager =
(LocationManager) getSystemService(Context.LOCATION_SERVICE);

provider = locationManager.getBestProvider(criteria, false);
location = locationManager.getLastKnownLocation(provider);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, this);
```

We now require to check the result of the "permission request". For this purpose, we simply need to override `onRequestPermissionsResult`, the variable `MY_PERMISSIONS_LOCATION` is used in order to distinguish the different permissions we might have asked to the user, e.g. location from access to the contact list, access to SMS service, camera etc.

```
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String permissions[], int[] grantResults) {
      switch (requestCode) {
            case MY_PERMISSIONS_LOCATION: {
                  // If request is cancelled, the result arrays are empty.
                  if (grantResults.length > 0
                        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                        Log.i("Access:","Now permissions are granted");
                        // permission was granted, yay!
                  } else {
                        Log.i("Access:"," permissions are denied");
                  //disable the functionality that depends on this permission.
                  }
            break;
            }
      // other 'case' lines to check for other permissions this app might request
      }
}
```

4.3.5 Remove update onPause

```
@Override
protected void onPause() {
    super.onPause();
    locationManager.removeUpdates(this);
 }
```

4.3.6 Show the user location

This activity will query the location manager and display the queried values in the activity.

```
public void showLocation(View view){
       Log.i("showLocation", "Entered");
       switch (view.getId()){
              case R.id.button01:
                 updateLocationView();
       }
}
public void updateLocationView(){
       if (location != null){
              double lat = location.getLatitude();
              double lng = location.getLongitude();
              latitudeField.setText(String.valueOf(lat));
              longitudeField.setText(String.valueOf(lng));
       } else{
             Log.i("showLocation","NULL");
         }
    }
```

4.3.7 Implement the listener

Now you need to implement the listener so that when the user changes location, its position is updated.

```
@Override
public void onLocationChanged(Location location) {
       Log.i("Location","LOCATION CHANGED!!!");
       updateLocationView();
    }
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {}
@Override
public void onProviderEnabled(String provider) {
       Toast.makeText(this, "Enabled new provider " + provider,
       Toast.LENGTH_SHORT).show();
}
@Override
public void onProviderDisabled(String provider) {
       Toast.makeText(this, "Disabled provider " + provider,
       Toast.LENGTH_SHORT).show();
}
```

4.3.8 Run the application

Run the application and through geo fix command or DDMS try to change the user positon.

- Is the position changing? If not, what is the problem? Fix it.

4.4 Add Google Map View

This lab uses Google Maps API, you should in this phase use the key created before in the introduction section. Add the following information to your AndroidManifest.xml file just before closing the application tag (i.e., before `</application>`)

```
<meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="THE_PREVIOUSLY_CREATED_KEY"/>
```

This key will be used to access correctly to the API while the application is running.

4.4.1 Compiling google Libraries

We need to compile additional libraries, namely Google Services libraries, to use the map object. In the Grandle file add the highlighted line to the dependencies:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.google.android.gms:play-services:8.1.0'
}
```

After adding the line you need to sync the Grandle file to the project, this will basically allow to use all those library that before were not available.

4.4.2 Adding the map fragment

The **MapFragment** class extends the Fragment class. Remember that a fragment has its own lifecycle, and thus the MapFragement. Add the following element to the layout file.

```
<fragment
        android:name="com.google.android.gms.maps.MapFragment"
        android:id="@+id/map"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

This fragment would be not recognized without importing the play services in the Grandle file.

4.4.3 Implementing the OnMapReadyCallback

Add the following global variables to the ShowLocation activity.

```
private GoogleMap googleMap;
static final LatLng NICE= new LatLng(43.7031,7.02661);
static final LatLng EURECOM = new LatLng(43.614376,7.070450);
```

Now we have all the elements to use a Google maps inside our application, we can start by implement the MapReady Callback in order to be able to handle it. Declare that the ShowLocation activity implements also OnMapReadyCallback and, as suggested add the method implementation:

At the end of the onCreate add the following lines

```
MapFragment mapFragment = (MapFragment) getFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
```

4.4.4 Fill/add onMapReady

We know fill the onMapReady method in order to center the map on Eurecom

```
@Override
public void onMapReady(GoogleMap Map) {

googleMap = Map;
        CameraPosition cameraPosition = new CameraPosition.Builder()
                .target(EURECOM)
                .zoom(17)
                .bearing(90)
                .tilt(30)
                .build();

        googleMap.addMarker(new MarkerOptions()

                .position(NICE)
                .title("Nice")
                .snippet("Enjoy French Riviera"));
        googleMap.addMarker(new MarkerOptions()

                .position(EURECOM)
                .title("EURECOM")
                .snippet("ENJOY STUDY!"));
}
googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

4.4.5 Run the application
4.4.6 Questions

- Explain what is a camera in google maps?
- Update automatically the position both in terms of longitude and latitude labels and in the map
- Now that the position is automatically updated the button show my location is useless: change it and use it to change the type of map from normal to hybrid
- Detect when a user is clicking on the map and show it through a toast

5 Proximity Alert

Proximity Alert Mobile Application allowing users to set and receive alerts based on location proximity, similar to setting and receiving traditional time-based event reminders [5-7].

Clicking on the map the user has the possibility to add proximity alert in specific positions.

Import in your project the class ProximityIntentReceiver.java

Add in the ShowLocation activity file the following global variables:

```
PendingIntent pendingIntent;
public SharedPreferences sharedPreferences;
private static final String PROX_ALERT_INTENT =
"fr.eurecom.locationservices.android.lbs.ProximityAlert";
private static final String POINT_LATITUDE_KEY = "POINT_LATITUDE_KEY";
private static final String POINT_LONGITUDE_KEY = "POINT_LONGITUDE_KEY";
private int count = 0
```

In onCreate() initialize sharedPreferences:

```
sharedPreferences = getSharedPreferences("location",0);
```

Through the last question of the previous section you should have added correctly the onMapClickListener and implemented the onMapClick method, in this method you need to insert the code which will add the proximity alert.

```
Intent intent = new Intent(PROX_ALERT_INTENT);
                PendingIntent proximityIntent =
PendingIntent.getBroadcast(getApplicationContext(), 0, intent, 0);

                if (ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
                        ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
{
                    Log.i("Permission: ", "To be checked");


                    return;
                } else {
                    Log.i("Permission: ", "GRANTED");
                }
                saveCoordinatesInPreferences((float) point.latitude,
                        (float) point.longitude);


                locationManager.addProximityAlert(point.latitude, point.longitude,
20, -1, proximityIntent);
                IntentFilter filter = new IntentFilter(PROX_ALERT_INTENT);
                registerReceiver(new ProximityIntentReceiver(), filter);
                Log.i("Registred", "proximity");


                Toast.makeText(getBaseContext(), "Added a proximity Alert",
Toast.LENGTH_LONG).show();
                ++count;
```

You need also to add the method saveCoordinatesInPreferences(…)

```
private void saveCoordinatesInPreferences(float latitude, float longitude) {

        SharedPreferences prefs =
this.getSharedPreferences(getClass().getSimpleName(),

                Context.MODE_PRIVATE);

        SharedPreferences.Editor prefsEditor = prefs.edit();

        prefsEditor.putFloat(POINT_LATITUDE_KEY, latitude);

        prefsEditor.putFloat(POINT_LONGITUDE_KEY, longitude);

        prefsEditor.commit();

    }
```

5.1.1 Run the code

Test the proximity by changing your simulator position

5.1.2 Questions

- What is a broadcast receiver? And an Filter?
- Add a marker where the proximity alert are added
- You have the possibility to add several Proximity Alerts, but, if n are added, you will notice that you will receive n notifications, can you solve this problem?

Upload the application to the course directory with the following name, "location_services_name1_name2.zip"

6 Reference

[1] https://developers.google.com/maps/documentation/android/
[2] http://developer.android.com/guide/topics/fundamentals.html
[3] http://www.vogella.de/google.html
[4] http://developer.android.com/resources/index.html
[5] http://code.google.com/p/android-for-gods/w/list
[6] http://blog.brianbuikema.com/2010/07/part-1-developing-proximity-alerts-for-mobile-applications-using-the-android-platform/
[7] http://marakana.com/training/android/ and http://marakana.com/forums/android/examples/
[8] https://github.com/gauntface/Android-Proximity-Alerts-Example
[9] http://code.google.com/p/demo-android-proximity-alert/source/checkout