CS 4518 Mobile and Ubiquitous Computing
Lecture 7: Location-Aware Computing

Emmanuel Agu
Adminstrivia

- Project 3 mailed out tomorrow, due next Thursday
- Graded papers for projects 0 and 1 now on InstructAssist
- Quiz in class next Monday, February 5 (first 15 mins)
  - Lectures 6, 7 + any code referenced
  - Project 1, 2 code
- Groups should submit 1-slide on their final project (due 11.59PM on Monday, February 15)
Reminder: Final Project

- 1-slide from group next Monday (2/5):
  - 2/35 of final project grade

- Slide should cover 3 aspects
  1. **Problem you intend to work on**
     - Solve WPI/societal problem (e.g. walking safe at night)
     - Use at least location, 1 sensor or camera
     - If games, must gamify solution to real world problem
  
  2. **Why this problem is important**
     - E.g. 37% of WPI students feel unsafe walking home

  3. **Summary of envisioned mobile app (?) solution**
     1. E.g. Mobile app automatically texts users friends when they get home at night

- Can bounce ideas of me (email, or in person)
- Can change idea any time
Final Project: Difficulty Score

- **Project execution:** 80%
- **Project difficulty score:** 20%
- **Mobile Components and Android UI (4 points each)**
  - Every 5 Android screens (A maximum of 8 points can be earned for the UI)
  - Playback audio/video
  - Maps, location sensing
  - Camera: simply taking pictures
- **Ubiquitous Computing Components & Android UI (6 points each)**
  - Activity Recognition, sensor programming, step counting
  - GeoFencing, Mobile Vision API: e.g. Face/barcode detection/tracking
- **Machine/Deep Learning (10 points each)**
  - Machine/deep learning (i.e. run study to gather data or use existing dataset to classify/detect something)
Location-Aware Computing

**Definition:** Location-aware applications generate outputs/behaviors that depend on a user’s location

**Examples:**
- Map of user’s “current location”
- Print to “closest” printer
- Apps that find user’s friends “closeby”
- Reviews of “closeby” restaurants

**Apps above require first determining user’s location**
Determining User Location on Smartphones
Location Tracking on Smartphones

- **Outdoors**: Uses GPS (More accurate)
- **Indoors**: WiFi or cell tower signals (Location fingerprinting, less accurate)
Global Positioning System (GPS)

- 27 satellites orbiting earth
- **20,000 km above earth** (Medium earth orbit)
- 6 orbital planes with 4 satellites each
- 4 satellites visible from any spot on earth
- Location of any location on earth specified as <longitude,latitude>
- E.g. Worcester MA has **Latitude**: 42.2625, **Longitude**: -71.8027778
GPS User Segment

- **Triangulation:** GPS receiver calculates user’s position by comparing time delay of signals to multiple satellites at known positions.

- Accuracy within 5 - 10 meters (16-32 feet)

http://adamswalk.com/gpx-2/
Determining User Location

- GPS reasonably accurate but
  - Requires line-of-sight between satellite and car receiver
  - Only works OUTDOORS (signals don’t penetrate buildings)
  - Lag/delay in acquiring satellites (~270 msec) or re-acquiring if lost
  - Drains battery power

- **Alternative:** Use Wi-Fi location sensing indoors
**WiFi Location Fingerprinting**

- **Key insight:** At each (X,Y) location, WiFi APs observed + their signal strengths, is unique

- **WiFi Location fingerprinting:** Infer device’s location based on combination of Wi-Fi access points seen + Signal Strengths

<table>
<thead>
<tr>
<th>OBSERVED AP SIGNAL STRENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP1</td>
</tr>
<tr>
<td>(X,Y)</td>
</tr>
</tbody>
</table>

### Diagram

- **Location (X,Y)**
- **AP1**
- **AP2**
- **AP3**
## Location Estimation using Wi-Fi Fingerprinting

### PRE-RECORDED TUPLES

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>SIGNAL STRENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>80</td>
<td>145</td>
</tr>
<tr>
<td>40</td>
<td>145</td>
</tr>
<tr>
<td>220</td>
<td>355</td>
</tr>
<tr>
<td>260</td>
<td>355</td>
</tr>
<tr>
<td>350</td>
<td>210</td>
</tr>
<tr>
<td>380</td>
<td>145</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AP1</th>
<th>AP2</th>
<th>AP3</th>
<th>AP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>28</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>36</td>
<td>20</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>-</td>
<td>25</td>
<td>36</td>
<td>44</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>39</td>
<td>42</td>
</tr>
<tr>
<td>16</td>
<td>-</td>
<td>28</td>
<td>36</td>
</tr>
<tr>
<td>22</td>
<td>12</td>
<td>-</td>
<td>44</td>
</tr>
</tbody>
</table>

### OBSERVED SIGNAL STRENGTH

<table>
<thead>
<tr>
<th>AP1</th>
<th>AP2</th>
<th>AP3</th>
<th>AP4</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>36</td>
<td>45</td>
<td></td>
</tr>
</tbody>
</table>

Google builds and stores this database (APs + Signal Strength) at each X,Y location.

- **Inference Algorithms**
  - Min. Threshold
  - Euclidean Dist.
  - Joint Probability
  - Bayesian Filters
How to Build table of APs observed at (X,Y) Locations?

- Devices (e.g. smartphone) with GPS and WiFi turned on simultaneously build table
- Send data to third party repositories (e.g. Wigle.net) or Google
- Also called war driving
- Can record cell tower signal strength instead of AP

Google gathers Location, AP seen Data if you consent

GPS satellites
Location consent
Allow Google's location service to collect anonymous location data. Some data may be stored on your device. Collection may occur even when no apps are running.

Google gathers Location, AP seen Data if you consent

<table>
<thead>
<tr>
<th>LOCATION</th>
<th>SIGNAL STRENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>80</td>
<td>145</td>
</tr>
<tr>
<td>40</td>
<td>145</td>
</tr>
<tr>
<td>220</td>
<td>355</td>
</tr>
<tr>
<td>260</td>
<td>355</td>
</tr>
</tbody>
</table>

WiFi card gathers APs seen + Signal Strengths
Location Sensing in Android Apps
Google Location APIs

https://developer.android.com/guide/topics/locationstrategies.html

- Android now has 2 location APIs (older vs newer)
- Newer location API is now part of Google Play Services
- Older Android framework location APIs (android.location)
  - Used by most books, online sources. We will use that

- LocationManager:
  - Android module receives location updates from GPS, WiFi, etc
  - App registers/requests location updates from LocationManager

```
Your app

requestLocationUpdates( LocationListener )

onStatusChanged

onProviderEnabled

onProviderDisabled

Location information

Android LocationManager
```

GPS WiFi Cell
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
        // Called when a new location is found by the network location provider.
        makeUseOfNewLocation(location);
    }
    public void onStatusChanged(String provider, int status, Bundle extras) {};
    public void onProviderEnabled(String provider) {};
    public void onProviderDisabled(String provider) {};
};

// Register the listener with the Location Manager to receive location updates
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, locationListener);
Requesting User Permissions

- Need smartphone owner’s permission to use their GPS

```
<manifest ...
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    ...
    <!-- Needed only if your app targets Android 5.0 (API level 21) or higher. -->
    <uses-feature android:name="android.hardware.location.gps" />
    ...
</manifest>
```

- **ACCESS_FINE_LOCATION**: GPS
- **ACCESS_COARSE_LOCATION**: WiFi or cell towers
Getting Cached Copy of Location (Fast)

- Getting current location may take a while
- Can choose to use location cached (possibly stale) from Location Manager

```java
String locationProvider = LocationManager.NETWORK_PROVIDER;
// Or use LocationManager.GPS_PROVIDER

Location lastKnownLocation = locationManager.getLastKnownLocation(locationProvider);
```
Stopping Listening for Location Updates

- Location updates consume battery power
- Stop listening for location updates whenever you no longer need

```java
// Remove the listener you previously added
locationManager.removeUpdates(locationListener);
```
Distance Travelled Updates using Services
Example from Head First Android
Example: Odometer (Distance Travelled) updates as a Services
(Ref: Head First Android 2\textsuperscript{nd} edition pgs 789 - 800)

- **Services**: long running background processes, no UI

- May want background service (a module in our app) to continuously retrieve location updates from LocationManager, forward updates to our Activity

- Ref: Head First Android pg 789
  - Example of using a Service
  - Nice Example app using Odometer Service
  - Tracks distance travelled
  - Gets, displays distance travelled every 10 secs
Example: Odometer (Distance Travelled) updates as a Services
(Ref: Head First Android pg 789)

- Example odometer app that tracks distance travelled
- getMiles(), displays distance travelled every 10 seconds

Study this example!!!
Location Representation
Semantic Location

- GPS represents location as \(<\text{longitude}, \text{latitude}>\)
- **Semantic location** is better for reasoning about locations
- E.g. Street address (140 Park Avenue, Worcester, MA) or (building, floor, room)
- **Android supports:**
  - **Geocoding:** Convert addresses into longitude/latitude coordinates
  - **Reverse geocoding:** convert longitude/latitude coordinates into human readable address

**Android Geocoding API:** access to **geocoding** and **reverse geocoding** services using HTTP requests
Google Places API Overview

- Access information, **high-quality photos** of a place
- Users can also add place information to the database
  - E.g. business owners can add their business as a place in Places database
  - Other apps can then retrieve info after moderation

**On-device caching:** Can cache places data locally on device to avoid roundtrip delays on future requests
Google Places

- **Place**: physical space that has a name (e.g. local businesses, points of interest, geographic locations)
  - E.g Logan airport, place type is **airport**

- **API**: Provides Contextual information about places near device.

- **E.g**: name of place, address, geographical location, place ID, phone number, place type, website URL, etc.

- Compliments geographic-based services offered by Android location services
Sample Place Types

accounting  hospital  city_hall  physiotherapist
airport    insurance_agency    clothing_store    place_of_worship (deprecated)
amusement_park    jewelry_store    convenience_store    plumber
aquarium    laundry    courthouse    police
art_gallery    lawyer    dentist    post_office
atm    library    department_store    real_estate_agency
bakery    liquor_store    doctor    restaurant
bank    local-government_office    electrician    roofing_contractor
bar    locksmith    electronics_store    rv_park
beauty_salon    lodging    embassy    school
bicycle_store    meal_delivery    establishment (deprecated)    shoe_store
tooker    meal_takeaway    finance (deprecated)    shopping_mall
book_store    mosque    fire_station    spa
bowling_alley    movie_rental    florist    stadium
bus_station    movie_theater    food (deprecated)    storage
cafe    moving_company    funeral_home    street
campground    museum    furniture_store    store
car_dealer    night_club    gas_station    subway_station
car_rental    painter    general_contractor (deprecated)    synagogue
car_repair    park    grocery_or_supermarket    taxi_stand
car_wash    painting    gym    train_station

Google Places API Overview

- **Use Place picker UI**: allows users to select place from “possible place” on a map

- **Get current place**: place where device is last known to be located
  - Returns **list** of likely places + likelihood device is in that place
Google Places API Overview

- **Autocomplete**: queries the location database as users type, suggests nearby places matching letters typed in.
Learning Google Places API

● Official Google Places website is “decent”, up to date:
  ● https://developers.google.com/places/

● Two great references:
  a) Getting started with Google Places API
     https://developers.google.com/places/android-api/start

  b) Tutorial by Paul Trebilco-Ruiz may be more readable:
     ● http://code.tutsplus.com/articles/google-play-services-using-the-places-api--cms-23715
Other Useful Google Maps/Location APIs
**GeoFencing**

https://developer.android.com/training/location/geofencing.html

- **Geofence**: Sends alerts when user is within a certain radius to a location of interest

- Can be configured to send to app:
  - **ENTER** event when user enters circle
  - **EXIT** event when user exits circle

- Can also specify a duration or **Dwell** user must be in circle before triggering event
GeoFencing
https://developer.android.com/training/location/geofencing.html

- **Great reference:**
  - How to work with GeoFences on Android by Tin Megali
    https://code.tutsplus.com/tutorials/how-to-work-with-geofences-on-android--cms-26639
Other Maps/Useful Location APIs

- **Maps Directions API:** calculates directions between locations (walking, driving) as well as public transport directions

- **Distance Matrix API:** Calculate travel time and distance for multiple destinations

- **Elevation API:** Query locations on earth for elevation information, calculate elevation changes along routes
Other Useful Maps/Location APIs

- **Roads API:**
  - snaps set of GPS coordinates to road user was likely travelling on (best fit)
  - Returns posted speed limits for any road segment (premium plan)

- **Time Zone API:** request time zone for location on earth
Using Maps
MapView and MapActivity

- **MapView**: UI widget that displays maps
- **MapActivity**: java class (extends Activity), handles map-related lifecycle and management for displaying maps.
7 Steps for using Google Maps Android API
https://developers.google.com/maps/documentation/android-api/start

1. Install Android SDK (Done!!)
2. Add Google Play services to Android Studio
3. Create a Google Maps project
4. Obtain Google Maps API key
5. Hello Map! Take a look at the code
6. Connect an Android device
7. Build and run your app
Step 2: Add Google Play Services to Android Studio

- Google Maps API v2 is part of Google Play Services SDK
- Use Android Studio SDK manager to download Google Play services

Open SDK Manager
Click on SDK Tools
Check Google Play Services, then Ok
Step 3: Create new Android Studio Project

https://developers.google.com/maps/documentation/android-api/start

- Select “Google Maps Activity, click Finish
Step 4: Get Google Maps API key
https://developers.google.com/maps/documentation/android-api/start

- To access Google Maps servers using Maps API, must add Maps API key to app
- Maps API key is free. E.g.

```
AIzaSyCc0_lEEjPllTLnPkVsX10YIY7oBa9XsXs
```

- Google uses API key to uniquely identify your app, track its resource usage, etc
Step 4a: Fast, Easy way to get Maps API Key

https://developers.google.com/maps/documentation/android-api/start

- Copy link provided in `google_maps_api.xml` of Maps template into browser
- Goes to Google API console, auto-fills form
- Creates API key

Register your application for Google Maps Android API in Google API Console

Google API Console allows you to manage your application and monitor API usage.

You have no existing projects. A new project named 'My Project' will be created.

Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.

☐ Yes  ☐ No

I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.

☐ Yes  ☐ No

Agree and continue

The API is enabled

The project has been created and Google Maps Android API has been enabled.

Next, you'll need to create an API key in order to call the API.

Create API key
Step 4a: Fast, Easy way to get Maps API Key
https://developers.google.com/maps/documentation/android-api/start

- If successful, Maps API key generated

API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key

AIzaSyCc0_lEEjPllTLnPkVsX10YIY7oBa9XsXs

⚠ Restrict your key to prevent unauthorized use in production.

- Copy key, put it in `<string>` element in `google_maps_api.xml` file

```xml
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyCc0_lEEjPllTLnPkVsX10YIY7oBa9XsXs</string>
```
Step 4b: Longer (older) way to API key

- If easy way doesn’t work, older way to obtain a Maps API key
- Follow steps at:
  - See: https://developers.google.com/maps/documentation/android-api/signup
Step 5: Examine Code Generated by Android Studio Maps Template

- XML file that defines layout is in `res/layout/activity_maps.xml`

```xml
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/map"
  tools:context="MapsActivity"
  android:name="com.google.android.gms.maps.SupportMapFragment" />
```
Step 5: Examine Code Generated by Android Studio Maps Template

- Default Activity file is `MapActivity.java`

```java
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;

public class MapActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @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));
    }
}
```
Steps 6, 7

- **Step 6:** Connect to an Android device (smartphone)

- **Step 7:** Run the app
  - Should show map with a marker on Sydney Australia

- More code examples at:
  - [https://github.com/googlemaps/android-samples](https://github.com/googlemaps/android-samples)
Location-Aware Apps from Past Offerings
Location-Aware Ideas from Previous Offerings

- **Ground rules:**
  - Apps must use mobile, location or sensors
  - Try to solve problems of benefit to WPI community

- More than half of apps used location.

- **Give me some space:** Bianchi, Chow, Martinez ’16
  - Find available study spaces on campus during exam week
  - Set up geoFences at study locations, count users in/out
Location-Aware Ideas from Previous Offerings

- **HomeSafe**: Nickerson, Feeley, Faust ’16
  - Safety app
  - Automatically sends message to users’ subscribers when they get home safely

- Project from grad class:
  - **Mansoor et al**: WPI automatic parking tracking/finder
Some Interesting Location-Aware Apps
MileIQ

- **The Problem:** Mileage tracking is useful but a burden.
  - IRS deductions on taxes
  - Some companies reimburse employees for mileage,
- Passively, automatically tracks business mileage, IRS compliant
- Swipe right after drive to indicate it was a business trip
- Project idea? Implement some of this functionality

- How Android modules? For what?
- What stats to decide if this is tackling important problem?
Trigger

- Use geofences, NFC, bluetooth, WiFi connections, etc to set auto-behaviors
  - Battery low -> turn off bluetooth + auto sync
  - Silence phone every morning when you get to work
  - Turn off mobile data when you connect to your home WiFi
  - Silence phone and set alarm once I get into bed
  - Use geofence for automatic foursquare checkin
  - Launch maps when you connect to your car’s bluetooth network

- Project idea? Implement subset of these features

- What triggers would be useful for a WPI student?
AsyncTask API
AsyncTask API

- For compute intensive tasks, remote or tasks that take a long time, doing it in main activity blocks
- **AsyncTask**: spawn separate thread to offload such task, free up main Activity
Playing Audio and Video in Android
**MediaPlayer**

http://developer.android.com/guide/topics/media/mediaplayer.html

- Classes used to play sound and video in Android
  - **MediaPlayer**: Plays sound and video
  - **AudioManager**: plays only audio

- MediaPlayer can fetch, decode and play audio or video from:
  - Audio/video files stored in app’s resource folders (e.g. `res/raw/` folder)
  - External URLs (over the Internet)

- Any Android app can use MediaPlayer APIs to integrate video/audio playback functionality
MediaPlayer supports:

- **Streaming network protocols**: RTSP, HTTP streaming
- **Media Formats**:
  - Audio (MP3, AAC, MIDI, etc),
  - Image (JPEG, GIF, PNG, BMP, etc)
  - Video (MPEG-4, H.263, H.264, H.265 AVC, etc)

4 major functions of a Media Player

- **User interface**, user interaction
- Handle **Transmission errors**: retransmissions, interleaving
- **Decompress** audio
- **Eliminate jitter**: Playback buffer (Pre-download 10-15 secs of music)
Using Media Player:
http://developer.android.com/guide/topics/media/mediaplayer.html
Step 1: Request Permission in AndroidManifest or Place video/audio files in res/raw

- If streaming video/audio over Internet (network-based content), request network access permission in AndroidManifest.xml:

```xml
<uses-permission android:name="android.permission.INTERNET" />
```

- If playing back local file stored on user’s smartphone, put video/audio files in res/raw folder
Using MediaPlayer

Step 2: Create MediaPlayer Object, Start Player

- To play audio file saved in app’s `res/raw/` directory

```java
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

- **Note:** Audio file opened by create (e.g. `sound_file_1.mpg`) must be encoded in one of supported media formats
Using MediaPlayer

Step 2: Create MediaPlayer Object, Start Player

- To play audio from remote URL via HTTP streaming over the Internet

```java
String url = "http://......."; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```
Releasing the MediaPlayer

- MediaPlayer can consume valuable system resources
- When done, call `release()` to free up system resources
- In `onStop()` or `onDestroy()` methods, call

```java
mediaPlayer.release();
mediaPlayer = null;
```

- **MediaPlayer in a Service**: Can play media (e.g. music) in background while app is not running
  - Start MediaPlayer as service
Playing Audio File using MediaPlayer
Example from Android Nerd Ranch 1\textsuperscript{st} edition
MediaPlayer Example to Playback Audio
from Android Nerd Ranch (1st edition) Ch. 13

- **HelloMoon app** that uses **MediaPlayer** to play audio file

Buttons to start/stop audio
References

- John Corpuz, 10 Best Location Aware Apps
- Liane Cassavoy, 21 Awesome GPS and Location-Aware Apps for Android,
- Head First Android
- Android Nerd Ranch, 2nd edition
- Busy Coder’s guide to Android version 6.3
- CS 65/165 slides, Dartmouth College, Spring 2014
- CS 371M slides, U of Texas Austin, Spring 2014