What can your smartphone do?

- Monitor your heart rate
- Pulse oximetry (Photoplethysmograph)
What can your smartphone do?

- Determine your altitude using pressure sensors
- Helps locate which floor you are on – indoor navigation
- Track outdoor adventures
What can your smartphone do?

- Help break into your phone
- Motion data captured from sensors could help predict your PIN/unlock codes
What can your smartphone do?

- Sniff out suspicious substances
- Cell-All by Department of Homeland Security (DHS)
What can your smartphone do?

• More sensors
  • Sensors to detect perspiration – mood conditions
  • Temperature and humidity sensors
  • Radiation – dosimeter
  • Air quality
  • Alcohol detector/breath analysis
  • Glucometer
What can your smartphone do?

- Detect aliens!
This talk

• Sensors on Android devices
• Location services in Android
• Region monitoring application
• Challenges
• Android vs. iOS
• Accessing motion sensors
Introduction
Sensor

- A device that can capture a physical quantity and convert it into a signal that can be measured
- Using sensors gives you knowledge of the external conditions and helps you respond appropriately
- Sensors give your app the power to provide amazing functionality
Sensors in smartphones

• Microphone
• Camera
• Touch screen
• Accelerometer
• Ambient temperature sensor
• Gyroscope
• Ambient light sensor
• Magnetometer
• Barometer
• Proximity sensor
• GPS
Sensors in Android

- **Motion sensors**
  - Measure acceleration and gravitational forces along 3 axes
  - Accelerometer, gyroscope

- **Environmental sensors**
  - Measure environmental parameters like temperature, pressure, illumination and humidity
  - Thermometer, barometer, photometer, etc.

- **Position sensors**
  - Measure the physical position of the device
  - Location, orientation sensors and magnetometer
Location services
Location services

• Android provides a variety of location services
• Get device location
  – Latitude, longitude
  – Altitude
• Geocoding
  – Mapping of (latitude, longitude) to street address
  – Reverse geocoding
• Proximity alerts
  – Notifications when the device comes within a given proximity of a location – GeoFencing
Location providers

- Android offers three providers to determine the location, depending on the requirements:
  - Network provider
    - Uses WiFi and mobile network
  - GPS provider
    - Uses on-board GPS
  - Passive provider
    - Uses location updates requested by other applications or services
Network provider

• WiFi access points
  – Identify access points in range (using MAC addresses) and estimate the relative location based on signal strengths
  – Requires WiFi be ON (need not be connected to any access point)

• Mobile network
  – Uses relative signal strengths from the current base station and surrounding base stations to ‘triangulate’ the device location

• Requires Google location service
  – To get the best known location of an access point/base station, if available
  – Network signature data is collected through dedicated instruments and crowd sourced data
  – You need internet access!
GPS provider

• Global Positioning System (GPS) – a satellite-based navigation system
  – Developed in 1973 and made fully operational in 1994 by the Department of Defense
  – About 32 satellites currently in 6 orbital planes

• Other similar systems
  – GLONASS (Russia)
  – Galileo (European Union)
  – Compass navigation (China)
  – Indian regional navigation (India)
GPS provider

- Compute your location based on the receptions from all visible GPS satellites
- Require at least 4 satellites to compute location
  - Latitude
  - Longitude
  - Altitude
  - Time
- Usually about 9 satellites are visible from any point on earth under clear sky
GPS provider

• Some problems
  – Signal interference e.g. atmospheric conditions, surrounding conditions like buildings, hills
  – Line of sight usually required – problematic indoors, in garages
  – Power consumption while locking to a satellite signal

• Assisted GPS (A-GPS)
  – Assistive GPS information received through mobile network
## Provider comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>GPS</th>
<th>Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to First Fix (TTFF)</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Power consumption</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Accuracy</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Altitude support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Bearing support</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Speed information</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Using location services
Using location services

• LocationManager
  - A service to provide access to location

• LocationListener
  - Interface for location services

• Location
  - Location data object

• LocationProvider
  - Location source
Using location services

• Implement LocationListener
  – onLocationChanged()
  – onProviderDisabled()
  – onProviderEnabled()
  – onStatusChanged()
• Register LocationListener with LocationManager
• Process Location object in onLocationChanged()
• Unregister LocationListener when done
private LocationManager locationManager;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.current_location);

    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

    // other initializations
}

Location permissions

Coarse location
- Network provider

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

Fine location
- Network provider
- GPS provider
- Passive provider

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
Implement LocationListener

```java
LocationListener locationListener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        // use the location
    }

    @Override
    public void onProviderEnabled(String provider) {
        //
    }

    @Override
    public void onProviderDisabled(String provider) {
        //
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        //
    }
};
```
Register LocationListener

• Decide the provider to request your location updates

locationManager.requestLocationUpdates(provider, 0, 0, locationListener);

• Pick your provider judiciously based on
  – Power requirements
  – Accuracy requirements
  – Other criteria

• Make sure the provider is enabled; prompt an alert to the user if not
Use Criteria

Criteria sensingCriteria = new Criteria();
sensingCriteria.setPowerRequirement(Criteria.POWER_LOW);
sensingCriteria.setHorizontalAccuracy(Criteria.ACCURACY_MEDIUM);

- Power requirement
- Horizontal accuracy
- Vertical accuracy
- Bearing accuracy
- Speed accuracy
Use Criteria

- Gets the best (permitted) provider satisfying the criteria
- If several providers meet the criteria, the one with the best accuracy is returned
- If no provider meets the criteria, the criteria are loosened in the following order:
  - Power requirement
  - Accuracy
  - Bearing
  - Speed
  - Altitude
- Be careful and smart in providing the criteria:
  - Too loose a criteria will always get you GPS
  - Too tight a criteria results in higher overhead

public String getBestProvider (Criteria criteria, boolean enabledOnly)
public List<String> getAllProviders ()

- Returns all known location providers

public List<String> getProviders (boolean enabledOnly)

- Returns all location providers that are enabled (if the argument is true)

public List<String> getProviders (Criteria criteria, boolean enabledOnly)

- Returns all location providers that satisfy the given criteria

public String getBestProvider (Criteria criteria, boolean enabledOnly)

- Returns the location provider that best meets the criteria
public void requestLocationUpdates (String provider,
    long minTime,
    float minDistance,
    LocationListener listener)

• provider – the name of the location provider
• minTime – minimum interval between location updates (in ms)
• minDistance – minimum distance between location updates (in m)
Best practices

✓ Filter old locations

locationRequestTime > location.getTime() + MAX_TIME_INTO_THE_PAST

✓ Filter locations with poor accuracy

location.getAccuracy() > MAX_ACCURACY

✓ Get a fast fix with last known location

locationManager.getLastKnownLocation(locationProvider);

✓ Unregister the listener when you are done

locationManager.removeUpdates(locationListener);
Geocoding

• Mapping a (lat, lng) to an address or vice-versa (reverse geocoding)

```java
public List<Address> getFromLocation(
    double latitude,
    double longitude,
    int maxResults)
```

- Returns a list of addresses known to describe the area surrounding the location
- E.g. (37.615223, -122.389979) returns an address for SFO

```java
public List<Address> getFromLocationName (String locationName, int maxResults)
```

- Returns a list of addresses known to describe the named location
- E.g. SFO might return an address with location (37.615223, -122.389979)
Region monitoring

Enter/Exit UCSC
Region monitoring

How it works (1 of 2)

Check-in to your campus
Automatically!

You just entered campus!

Realtime dashboard
Proximity alerts

- Adds a proximity alert to a circular region with the given center and radius; monitor only for requested time
- Uses both network and GPS in the background to monitor; requires the permission ACCESS_FINE_LOCATION

```java
public void addProximityAlert (
    double latitude,
    double longitude,
    float radius,
    long expiration,
    PendingIntent intent)
```
Proximity alerts

• If the device passes through the given area briefly, it is possible that no Intent will be fired
• An Intent could be fired if the device passes very close to the given area but does not actually enter it – a false positive
• No support for custom region shapes
• No fine-grained control over the monitoring mechanism
Custom region monitoring

- Android makes it possible to build your own custom region monitoring solutions!
- Use a background service that periodically requests location updates
- Use adaptive sampling depending on the current location information
  - If the device is far from a region, request an update after a long period (e.g. 30 min.)
  - If the device is close to a region, keep tracking more often (e.g. once a min.)
  - Other parameters to adapt the sampling rate e.g. speed, battery, time of day
Challenges

• Background service gets killed and does not start again

```java
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    // start your service
    return START_STICKY;
}
```

- START_STICKY is used for services that are explicitly started and stopped as needed
- If the service is killed, Android will try to re-create the service in due time
Challenges

• Background service dies often
  − When running low on memory, Android can kill the background service

```java
public final void startForeground(int id, Notification notification)
```

➢ Starts the service in foreground mode i.e. high priority
➢ The device will show an ongoing notification of your service
Challenges

- Battery consumption is high
  - CPU usage could be high
  - For sparsely running services, prefer AlarmManager to wake up once in a while

```java
AlarmManager - set (int type, long triggerAtMillis, PendingIntent operation)
```

- Fires an intent at the scheduled time
- To ensure that the device does not sleep while your intent is being processed, you can hold the CPU wake lock e.g. `type = RTC_WAKEUP`

```java
public void setInexactRepeating (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation)
```

- Schedules a repeating alarm for inexact trigger requirements
- More power-efficient than exact alarms
Challenges

• Battery consumption is high
  - Check how often GPS is requesting for a location
  - Check how long GPS takes to get a location fix
    - Most of the energy used by GPS is spent in acquiring a signal lock
    - Detect if the device is unable to acquire a lock (say, within 60 sec.) and resign the request
  - Getting a location from WiFi is cheaper and faster than using GPS
    - Check if WiFi is enabled
Challenges

• False positives
  ➢ The reported location can claim good accuracy and still be quite off
  ➢ Trigger a fence-break only after confirming the event using e.g.
    – Location fix from a higher accuracy provider
    – More location updates
Challenges

• Location providers not available
  - Make sure you check for location providers before requesting a location
  - If unavailable, inform the user
  - Have a fallback option e.g. use GPS if Network is unable to get you a location
## Android vs. iOS

<table>
<thead>
<tr>
<th>Feature</th>
<th>Android</th>
<th>iOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control location provider</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Control location accuracy</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Selectively disable location providers</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Region monitoring support</td>
<td>Not too strong</td>
<td>Works fairly well out-of-the-box</td>
</tr>
<tr>
<td>Fine-grained region control</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Uniformity of sensor performance</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Background service functionality</td>
<td>Yes</td>
<td>Yes (restricted)</td>
</tr>
<tr>
<td>Process scheduling</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
• How about a custom region monitoring solution in iOS?
  – Start a background service polling location
    o iOS times out a background service in 10 mins. from the last activity → need to poll location at least once every 10 mins.
    o The CPU is awake all the while the background service is active
  – No ‘alarm’ service to schedule future calls
  – Heavy battery usage – ~50% drain per day
Other sensors

Use SensorManager to access other sensors

```java
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

Get default sensor for a particular signal type

```java
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
    // Success! There's a magnetometer.
}
else {
    // Failure! No magnetometer.
}
```
Implement SensorEventListener to access the events

```java
sensorEventListener = new SensorEventListener() {
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // sensor accuracy changed
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        switch (event.sensor.getType()) {
            case Sensor.TYPE_ACCELEROMETER: // dosomething break;
            case Sensor.TYPE_GYROSCOPE: // dosomething break;
            case Sensor.TYPE_ROTATION_VECTOR: // dosomething break;
        }
    }
};
```
Motion sensors

- **Accelerometer**
  - `Sensor.TYPE_ACCELEROMETER`: Acceleration values in x, y, z directions in m/s²
  - `Sensor.TYPE_GRAVITY`: Acceleration due to gravity
  - `Sensor.TYPE_LINEAR_ACCELERATION`: Acceleration without the influence of gravity

- **Gyroscope**
  - `Sensor.TYPE_ROTATION_VECTOR`: Measures the device's orientation
  - `Sensor.TYPE_GYROSCOPE`: Measures the rate of rotation around x, y, z axes in rad/s
Motion sensors

- Example – monitoring user activity
  - Walking, climbing stairs, biking, etc. involve periodic motions
  - Careful analysis of accelerometer signal coupled with gyroscope data can discriminate between different modes

![Graph showing frequency vs. power spectral density](image)
Summary

- Android offers a variety of sensing capabilities
- The sensing API is simple and offers complete control to the user
- Accessing sensors requires battery consumption – be judicious in requesting sensor access and the frequency of access
- Although the OS handles the hardware interface for you, follow good practices in registering and unregistering the listeners
- The future devices will be sensor-rich opening amazing avenues that were impossible until recently
• Smartphone sensors reveal security secrets - http://www.bbc.co.uk/news/technology-21203035
• Motion sensors could unlock smartphones, say researchers - http://www.computerweekly.com/news/2240176819/Motion-sensors-could-unlock-smartphones-say-researchers
• The Sensors Are Coming! - http://bits.blogs.nytimes.com/2011/05/19/the-sensors-are-coming/
• Smartphones become pocket labs - http://www.antonymayfield.com/2012/08/14/smartphones-become-pocket-labs/
• Professional Android Sensor Programming – Greg Milette, Adam Stround
Thank you

contact@1618labs.com