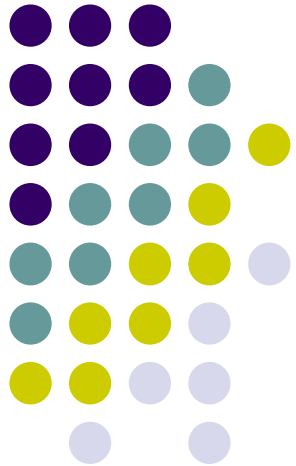
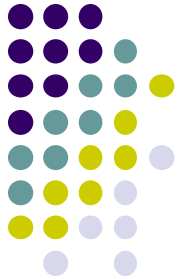
CS 4518 Mobile and Ubiquitous Computing

Lecture 8: Sensors, Step Counting & Activity Recognition

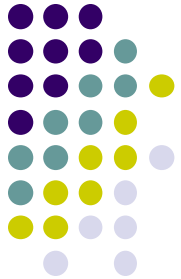
Emmanuel Agu



Administrivia



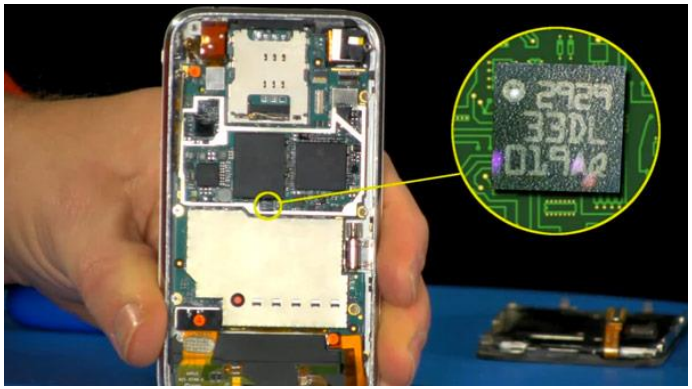
- Project 3 mailed out Saturday night, due Friday 11.59PM
- Groups should submit 1-slide on their final project (due 11.59PM tonight)
- Thursday: further discussion of final project proposal, presentation (on Monday)

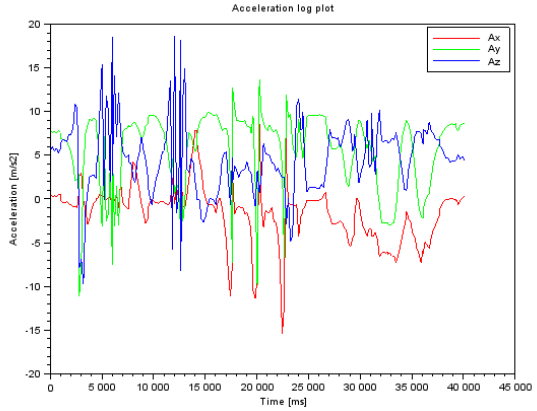


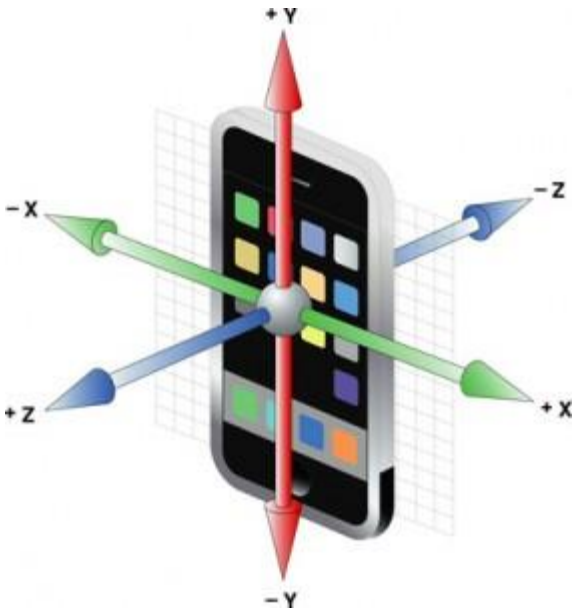
Android Sensors

What is a Sensor?

- Converts physical quantity (e.g. light, acceleration, magnetic field) into a signal
- **Example:** accelerometer converts acceleration along X,Y,Z axes into signal

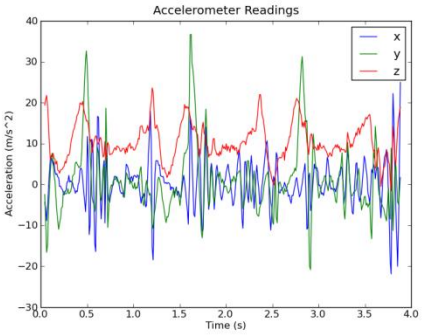




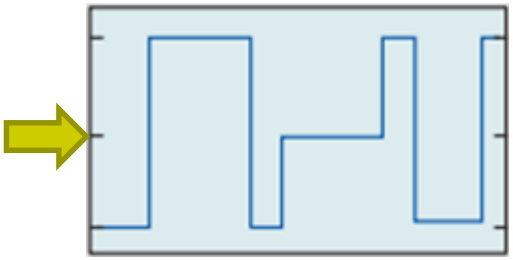


So What?

- Raw sensor data can be processed into useful info
- **Example:** Raw accelerometer data can be processed/classified to infer user's activity (e.g. walking running, etc)
- Voice samples can be processed/classified to infer whether speaker is nervous or not



Raw accelerometer readings



Machine learning Feature extraction and classification

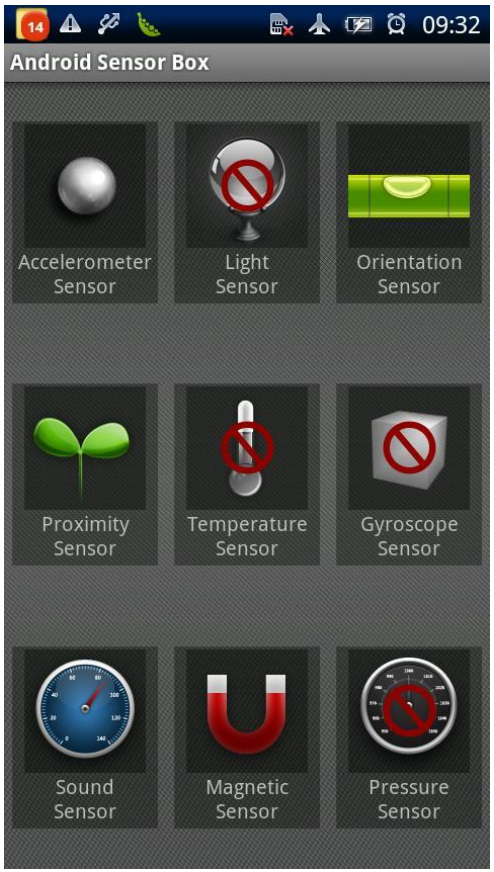
Walking
Running
Jumping
Step count
Calories burned
Falling

Android Sensors

- Microphone (sound)
- Camera
- Temperature
- Location (GPS, A-GPS)
- Accelerometer
- Gyroscope (orientation)
- Proximity
- Pressure
- Light

- **Different phones do not have all sensor types!!**

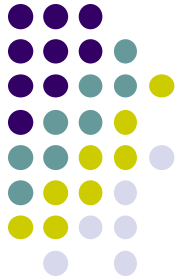
AndroSensor



Android Sensor Box

Android Sensor Framework

http://developer.android.com/guide/topics/sensors/sensors_overview.html

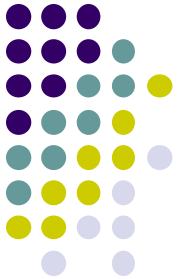


- Enables apps to:
  - Access sensors available on device and
  - Acquire raw sensor data

- Specifically, using the Android Sensor Framework, you can:
  - Determine **which sensors** are available on phone
  - Determine **capabilities of sensors** (e.g. max. range, manufacturer, power requirements, resolution)
  - **Register and unregister** sensor event listeners
  - **Acquire raw sensor data** and define data rate

Android Sensor Framework

http://developer.android.com/guide/topics/sensors/sensors_overview.html



- Android sensors can be either hardware or software

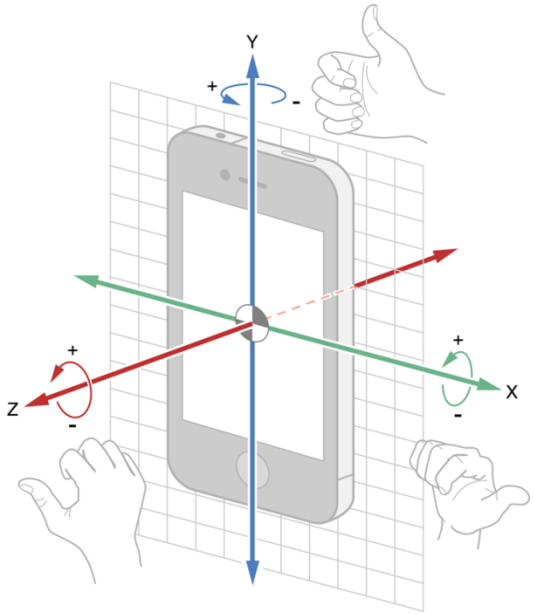
- **Hardware sensor:**
  - physical components built into phone,
  - **Example:** temperature

- **Software sensor (or virtual sensor):**
  - Not physical device
  - Derives their data from one or more hardware sensors
  - **Example:** gravity sensor

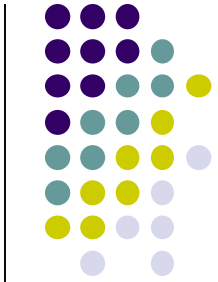
Sensor Types Supported by Android

- TYPE_PROXIMITY
  - Measures an **object's proximity to device's screen**
  - **Common uses:** determine if handset is held to ear

- TYPE_GYROSCOPE
  - Measures device's **rate of rotation** around X,Y,Z axes in rad/s
  - **Common uses:** rotation detection (spin, turn, etc)



Types of Sensors



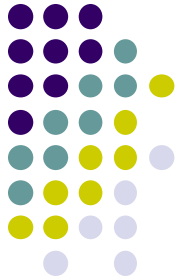
Sensor	HW/SW	Description	Use
TYPE_ACCELEROMETER	HW	Rate of change of velocity	Shake, Tilt
TYPE_AMBIENT_TEMPERATURE	HW	Room temperature	Monitor Room temp
TYPE_GRAVITY	SW/HW	Gravity along X,Y,Z axes	Shake, Tilt
TYPE_GYROSCOPE	HW	Rate of rotation	Spin, Turn
TYPE_LIGHT	HW	Illumination level	Control Brightness
TYPE_LINEAR_ACCELERATION	SW/HW	Acceleration along X,Y,Z – g	Accel. Along an axis
TYPE_MAGNETIC_FIELD	HW	Magnetic field	Create Compass
TYPE_ORIENTATION	SW	Rotation about X,Y,Z axes	Device position
TYPE_PRESSURE	HW	Air pressure	Air pressure
TYPE_PROXIMITY	HW	Any object close to device?	Phone close to face?
TYPE_RELATIVE_HUMIDITY	HW	% of max possible humidity	Dew point
TYPE_ROTATION_VECTOR	SW/HW	Device's rotation vector	Device's orientation
TYPE_TEMPERATURE	HW	Phone's temperature	Monitor temp

2 New Hardware Sensor introduced in Android 4.4

- TYPE_STEP_DETECTOR
  - Triggers sensor event each time user takes a step (**single step**)
  - Delivered event has value of 1.0 + timestamp of step

- TYPE_STEP_COUNTER
  - Also triggers a sensor event each time user takes a step
  - Delivers total ***accumulated number of steps since this sensor was first registered by an app***,
  - Tries to eliminate false positives
- **Common uses:** step counting, pedometer apps
- Requires hardware support, available in Nexus 5
- Alternatively available through Google Play Services (more later)

Sensor Programming

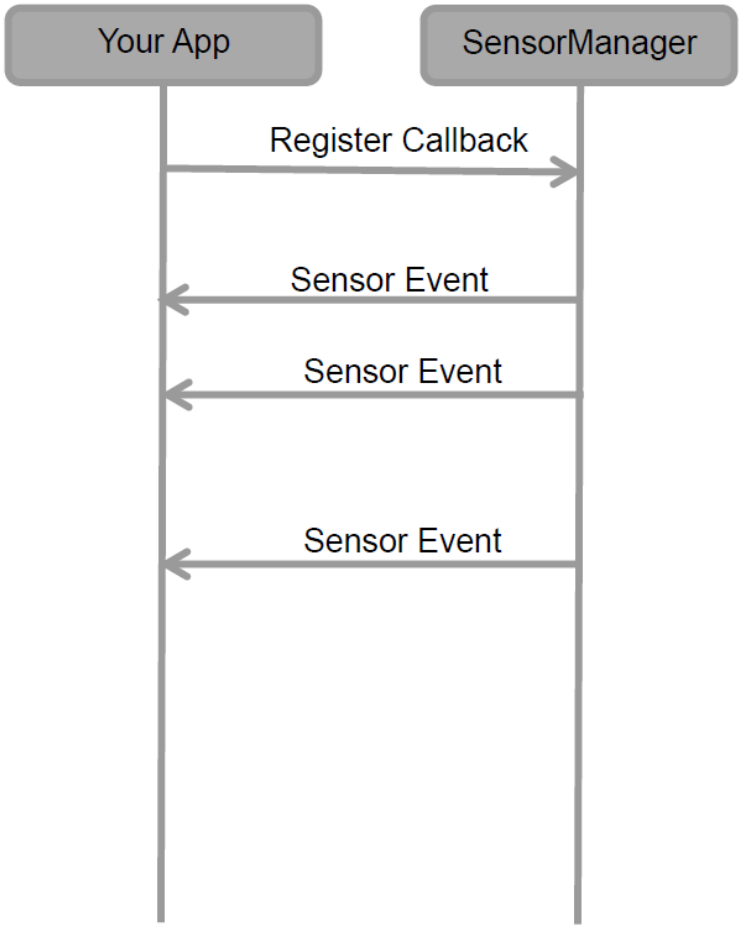


- Sensor framework is part of **android.hardware**
- Classes and interfaces include:
  - **SensorManager**
  - **Sensor**
  - **SensorEvent**
  - **SensorEventListener**
- These sensor-APIs used for:
  1. Identifying sensors and sensor capabilities
  2. Monitoring sensor events

Sensor Events and Callbacks

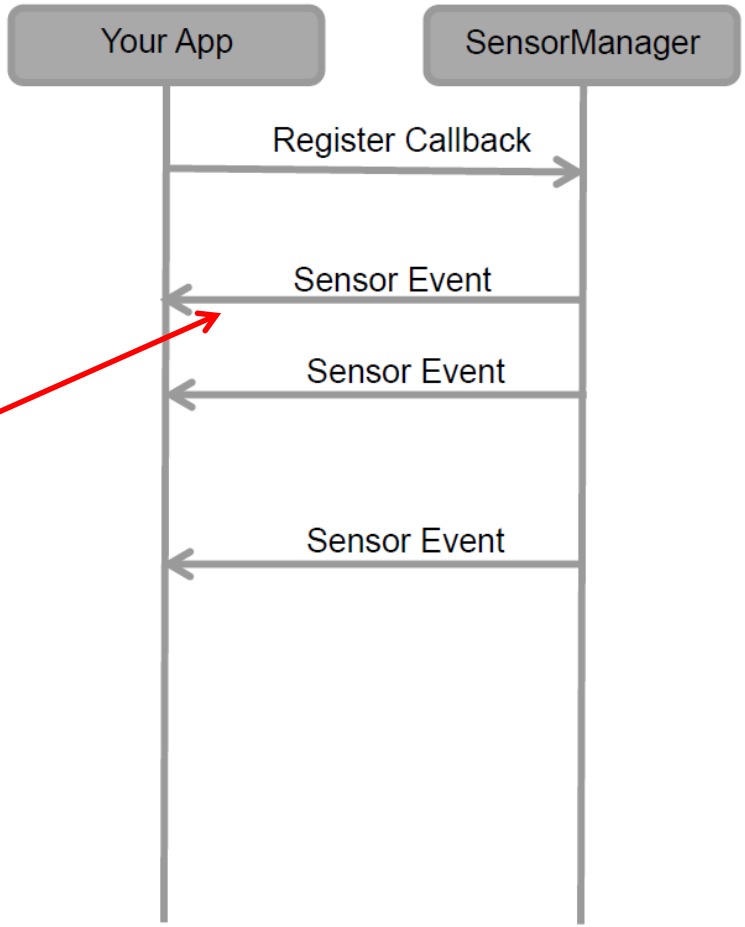
- Sensors send events to sensor manager asynchronously, when new data arrives

- General approach:
  - App registers callbacks
  - **SensorManager** notifies app of sensor event whenever new data arrives (or accuracy changes)



Sensor

- A class that can be used to create instance of a specific sensor
- Has methods used to determine a sensor's capabilities
- Included in sensor event object

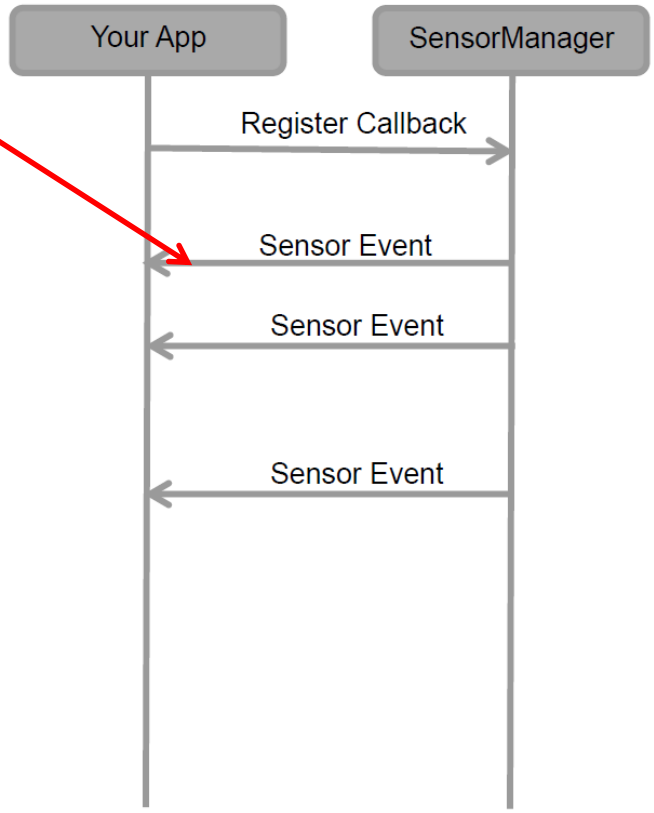


SensorEvent

- Android system sensor event information as a **sensor event object**

- **Sensor event object** includes:
  - ***Sensor:*** Type of sensor that generated the event
  - ***Values:*** Raw sensor data
  - ***Accuracy:*** Accuracy of the data
  - ***Timestamp:*** Event timestamp

Sensor value depends on sensor type



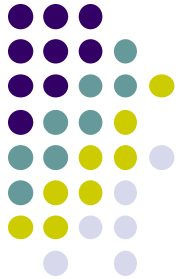
Sensor	Sensor event data	Description	Units of measure
TYPE_ACCELEROMETER	SensorEvent.values[0]	Acceleration force along the x axis (including gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (including gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (including gravity).	
TYPE_GRAVITY	SensorEvent.values[0]	Force of gravity along the x axis.	m/s^2
	SensorEvent.values[1]	Force of gravity along the y axis.	
	SensorEvent.values[2]	Force of gravity along the z axis.	
TYPE_GYROSCOPE	SensorEvent.values[0]	Rate of rotation around the x axis.	rad/s
	SensorEvent.values[1]	Rate of rotation around the y axis.	
	SensorEvent.values[2]	Rate of rotation around the z axis.	
TYPE_GYROSCOPE_UNCALIBRATED	SensorEvent.values[0]	Rate of rotation (without drift compensation) around the x axis.	rad/s
	SensorEvent.values[1]	Rate of rotation (without drift compensation) around the y axis.	
	SensorEvent.values[2]	Rate of rotation (without drift compensation) around the z axis.	
	SensorEvent.values[3]	Estimated drift around the x axis.	
	SensorEvent.values[4]	Estimated drift around the y axis.	
	SensorEvent.values[5]	Estimated drift around the z axis.	

Sensor Values Depend on Sensor Type

Sensor Values Depend on Sensor Type

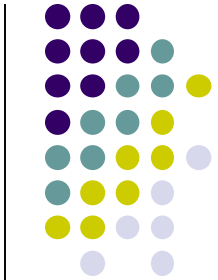
Sensor	Sensor event data	Description	Units of measure
TYPE_LINEAR_ACCELERATION	SensorEvent.values[0]	Acceleration force along the x axis (excluding gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (excluding gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (excluding gravity).	
TYPE_ROTATION_VECTOR	SensorEvent.values[0]	Rotation vector component along the x axis (x * sin(θ/2)).	Unitless
	SensorEvent.values[1]	Rotation vector component along the y axis (y * sin(θ/2)).	
	SensorEvent.values[2]	Rotation vector component along the z axis (z * sin(θ/2)).	
	SensorEvent.values[3]	Scalar component of the rotation vector ((cos(θ/2)).[1]	
TYPE_SIGNIFICANT_MOTION	N/A	N/A	N/A
TYPE_STEP_COUNTER	SensorEvent.values[0]	Number of steps taken by the user since the last reboot while the sensor was activated.	Steps
TYPE_STEP_DETECTOR	N/A	N/A	N/A

SensorEventListener



- Interface used to create 2 callbacks that receive notifications (sensor events) when:
  - Sensor values change **(onSensorChange( ))** or
  - When sensor accuracy changes **(onAccuracyChanged( ))**

Sensor API Tasks



- **Sensor API Task 1: Identifying sensors and their capabilities**
- Why identify sensor and their capabilities at runtime?
  - Disable app features using sensors not present, or
  - Choose sensor implementation with best performance

- **Sensor API Task 2: Monitor sensor events**
- Why monitor sensor events?
  - To acquire raw sensor data
  - Sensor event occurs every time sensor detects change in parameters it is measuring

Sensor Availability

- Different sensors are available on different **Android versions**

Sensor	Android 4.0 (API Level 14)	Android 2.3 (API Level 9)	Android 2.2 (API Level 8)	Android 1.5 (API Level 3)
TYPE_ACCELEROMETER	**Yes**	**Yes**	**Yes**	**Yes**
TYPE_AMBIENT_TEMPERATURE	**Yes**	n/a	n/a	n/a
TYPE_GRAVITY	**Yes**	**Yes**	n/a	n/a
TYPE_GYROSCOPE	**Yes**	**Yes**	n/a[1]	n/a[1]
TYPE_LIGHT	**Yes**	**Yes**	**Yes**	**Yes**
TYPE_LINEAR_ACCELERATION	**Yes**	**Yes**	n/a	n/a
TYPE_MAGNETIC_FIELD	**Yes**	**Yes**	**Yes**	**Yes**
TYPE_ORIENTATION	**Yes**[2]	**Yes**[2]	**Yes**[2]	**Yes**
TYPE_PRESSURE	**Yes**	**Yes**	n/a[1]	n/a[1]
TYPE_PROXIMITY	**Yes**	**Yes**	**Yes**	**Yes**
TYPE_RELATIVE_HUMIDITY	**Yes**	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	**Yes**	**Yes**	n/a	n/a
TYPE_TEMPERATURE	**Yes**[2]	**Yes**	**Yes**	**Yes**

Identifying Sensors and Sensor Capabilities

- First create instance of **SensorManager** by calling **getSystemService( )** and passing in SENSOR_SERVICE argument

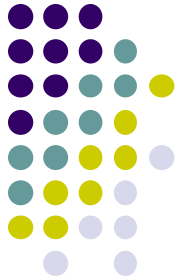
```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- Then list sensors available on device by calling **getSensorList( )**

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_ALL);
```

- To list particular type, use **TYPE_GYROSCOPE, TYPE_GRAVITY**, etc

http://developer.android.com/guide/topics/sensors/sensors_overview.html

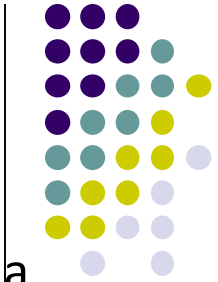


Checking if Phone has at least one of particular Sensor Type

- Device may have multiple sensors of a particular type.
  - E.g. multiple magnetometers
- If multiple sensors of a given type exist, one of them must be designated “the default sensor” of that type
- To determine if specific sensor type exists use **getDefaultSensor( )**
- **Example:** To check whether device has at least one magnetometer

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
  // Success! There's a magnetometer.
  }
else {
  // Failure! No magnetometer.
  }
```

Example: Monitoring Light Sensor Data



- **Goal:** Monitor light sensor data using **onSensorChanged( )**, display it in a **TextView** defined in main.xml

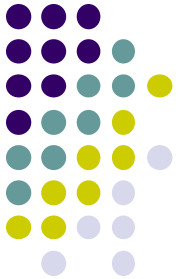
```
public class SensorActivity extends Activity implements SensorEventListener {
  private SensorManager mSensorManager;
  private Sensor mLight;

  @Override
  public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
  }

  @Override
  public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Do something here if sensor accuracy changes.
  }
```

Create instance of Sensor manager

Get default Light sensor

Called by Android system when accuracy of sensor being monitored changes

Example: Monitoring Light Sensor Data (Contd)



Called by Android system to report new sensor value
Provides SensorEvent object containing new sensor data

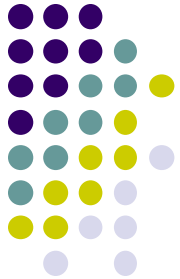
```
@Override
public final void onSensorChanged(SensorEvent event) {
  // The light sensor returns a single value.
  // Many sensors return 3 values, one for each axis.
  float lux = event.values[0];
  // Do something with this sensor value.
}

@Override
protected void onResume() {
  super.onResume();
  mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_DELAY_NORMAL);
}

@Override
protected void onPause() {
  super.onPause();
  mSensorManager.unregisterListener(this);
}
}
```

Get new light sensor value

Register sensor when app becomes visible

Unregister sensor if app is no longer visible to reduce battery drain

Handling Different Sensor Configurations



- Different phones have different sensors built in
- **E.g.** Motorola Xoom has pressure sensor, Samsung Nexus S doesn't
- If app uses a specific sensor, how to ensure this sensor exists on target device?
- Two options
  - **Option 1:** Detect device sensors at runtime, enable/disable app features as appropriate
  - **Option 2:** Use AndroidManifest.xml entries to ensure that only devices possessing required sensor can see app on Google Play
    - **E.g.** following manifest entry in AndroidManifest ensures that only devices with accelerometers will see this app on Google Play

```
<uses-feature android:name="android.hardware.sensor.accelerometer"
              android:required="true" />
```

Option 1: Detecting Sensors at Runtime

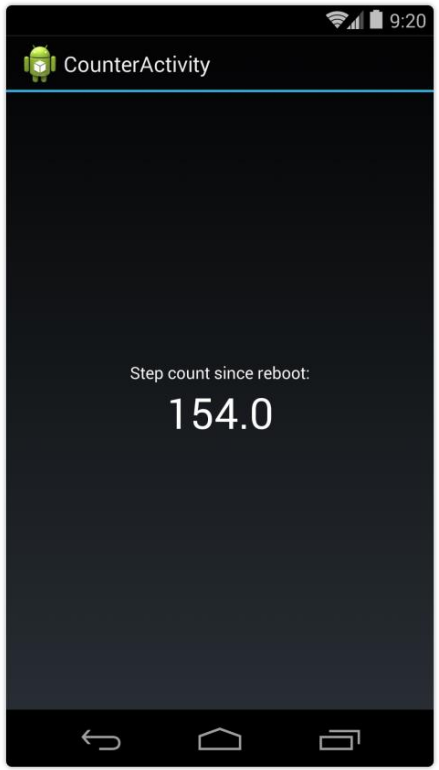
- Following code checks if device has at least one pressure sensor

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE) != null){
// Success! There's a pressure sensor.
}
else {
// Failure! No pressure sensor.
}
```

Example Step Counter App

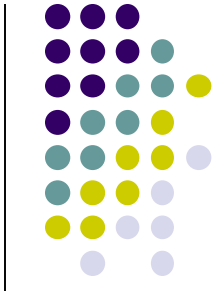
- **Goal:** Track user's steps, display it in TextView
- **Note:** Phone hardware must support step counting

```
1  package com.starboardland.pedometer;
2
3  import android.app.Activity;
4  import android.content.Context;
5  import android.hardware.*;
6  import android.os.Bundle;
7  import android.widget.TextView;
8  import android.widget.Toast;
9
10 public class CounterActivity extends Activity implements SensorEventListener {
11
12     private SensorManager sensorManager;
13     private TextView count;
14     boolean activityRunning;
15
16     @Override
17     public void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.main);
20         count = (TextView) findViewById(R.id.count);
21
22         sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
23     }
```



https://theelfismike.wordpress.com/2013/11/10/android-4-4-kitkat-step-detector-code/

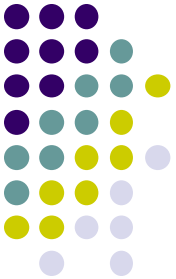
Example Step Counter App (Contd)



```
25      @Override
26      protected void onResume() {
27          super.onResume();
28          activityRunning = true;
29          Sensor countSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
30          if (countSensor != null) {
31              sensorManager.registerListener(this, countSensor, SensorManager.SENSOR_DELAY_UI);
32          } else {
33              Toast.makeText(this, "Count sensor not available!", Toast.LENGTH_LONG).show();
34          }
35
36      }
37
38      @Override
39      protected void onPause() {
40          super.onPause();
41          activityRunning = false;
42          // if you unregister the last listener, the hardware will stop detecting step events
43  //          sensorManager.unregisterListener(this);
44      }
```

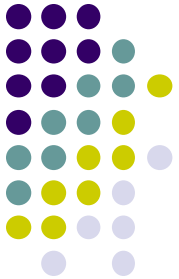
https://theelfismike.wordpress.com/2013/11/10/android-4-4-kitkat-step-detector-code/

Example Step Counter App (Contd)



```
46      @Override
47      public void onSensorChanged(SensorEvent event) {
48          if (activityRunning) {
49              count.setText(String.valueOf(event.values[0]));
50          }
51
52      }
53
54      @Override
55      public void onAccuracyChanged(Sensor sensor, int accuracy) {
56      }
57  }
```

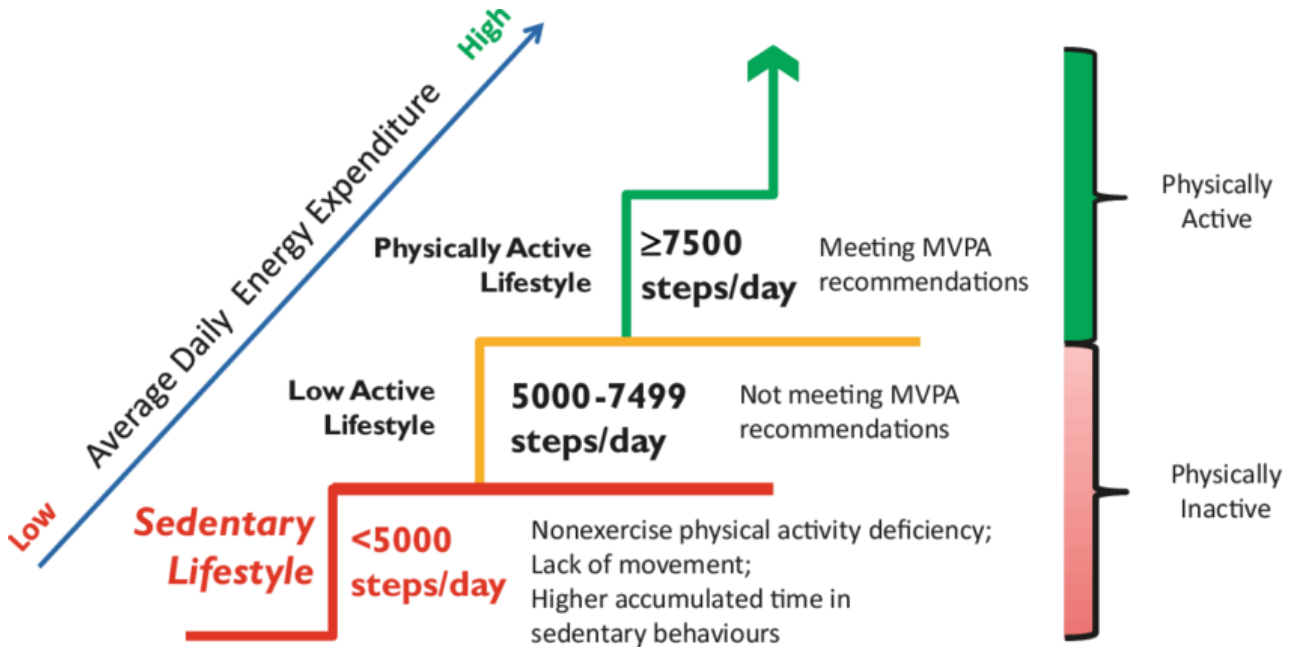
https://theelfismike.wordpress.com/2013/11/10/android-4-4-kitkat-step-detector-code/



Step Counting
(How Step Counting Works)

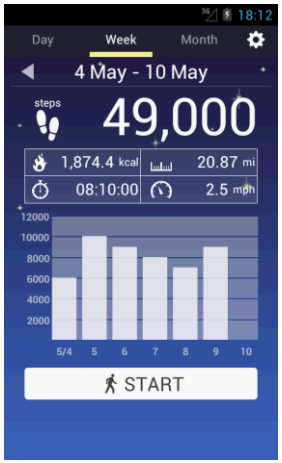
Sedentary Lifestyle

- Sedentary lifestyle
  - increases risk of diabetes, heart disease, dying earlier, etc
  - Kills more than smoking!!
- Categorization of sedentary lifestyle based on step count by paper:
  - "Catrine Tudor-Locke, Cora L. Craig, John P. Thyfault, and John C. Spence, A step-defined sedentary lifestyle index: < 5000 steps/day", Appl. Physiol. Nutr. Metab. 38: 100–114 (2013)



Step Count Mania

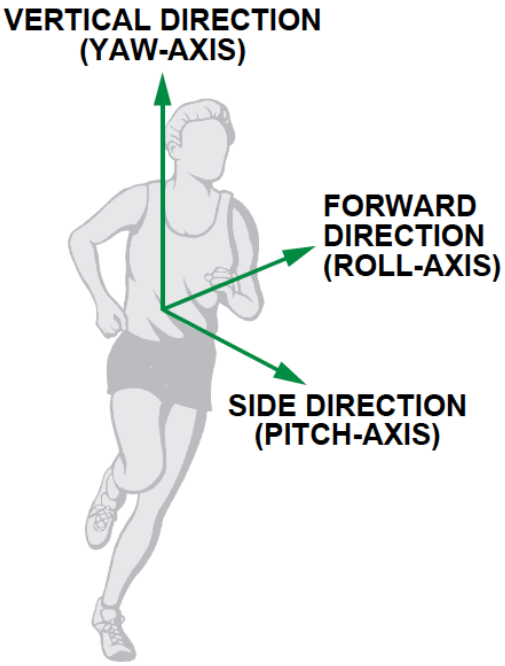
- Everyone is crazy about step count these days
- Pedometer apps, pedometers, fitness trackers, etc
- Tracking makes user aware of activity levels, motivates them to exercise more

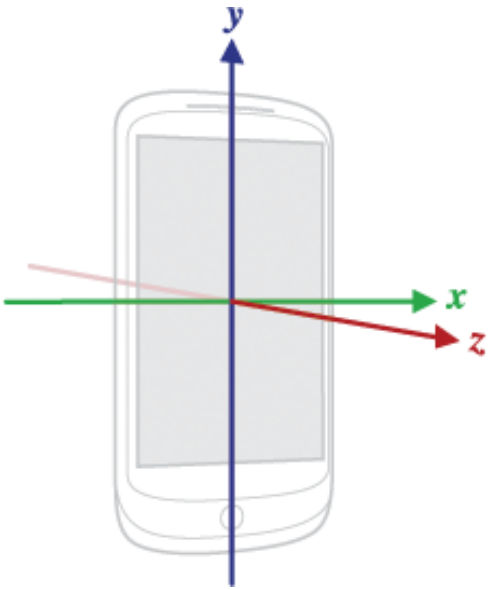


How does a Pedometer Detect/Count Steps

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- As example of processing Accelerometer data
- Walking or running results in motion along the 3 body axes (forward, vertical, side)
- Smartphone has similar axes
  - Alignment depends on phone orientation

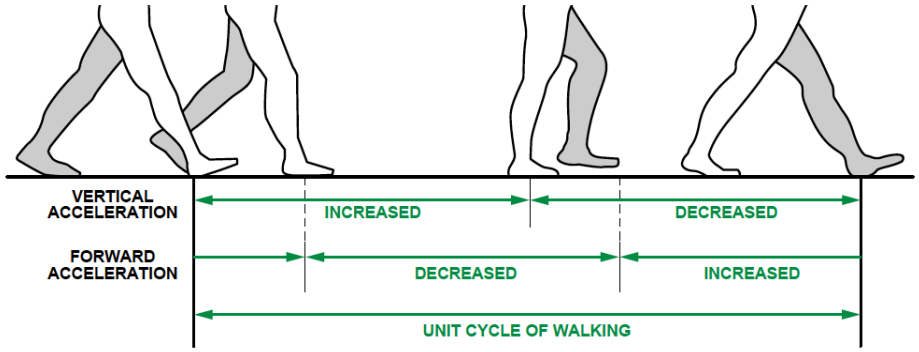


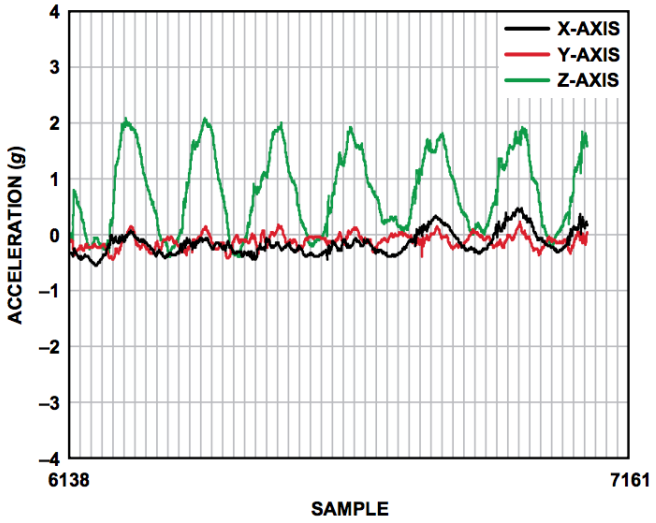


The Nature of Walking

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- Vertical and forward acceleration increases/decreases during different phases of walking
- Walking causes a large periodic spike in one of the accelerometer axes
- Which axes (x, y or z) and magnitude depends on phone orientation

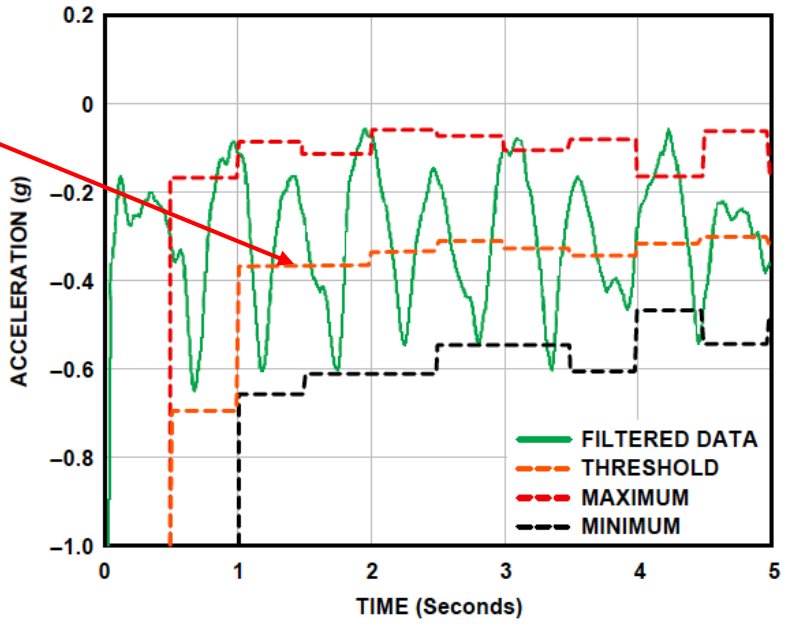




Step Detection Algorithm

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

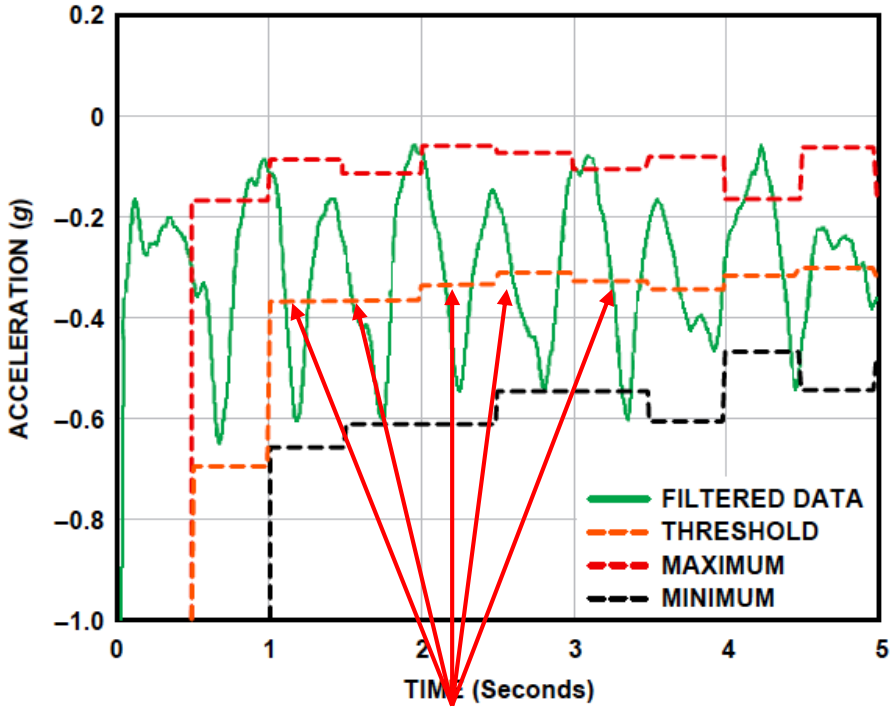
- **Step 1: smoothing**
  - Signal looks choppy
  - Smooth by replacing each sample with average of current, prior and next sample (Window of 3)
- **Step 2: Dynamic Threshold Detection**
  - Focus on accelerometer axis with largest peak
  - Would like a threshold such that each crossing is a step
  - But cannot assume fixed threshold (magnitude depends on phone orientation)
  - Track min, max values observed every 50 samples
  - Compute ***dynamic threshold: (Max + Min)/2***



Step Detection Algorithm

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- A step is
  - indicated by crossings of dynamic threshold
  - Defined as negative slope (sample_new < sample_old) when smoothed waveform crosses dynamic threshold

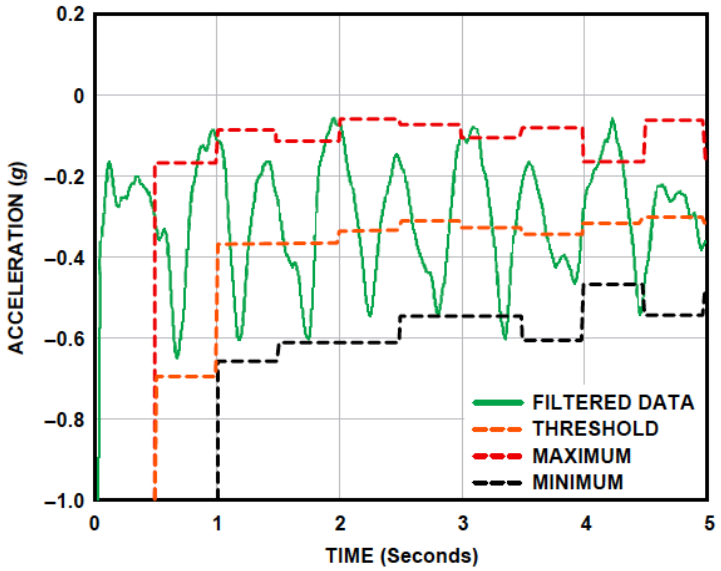


Steps

Step Detection Algorithms

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

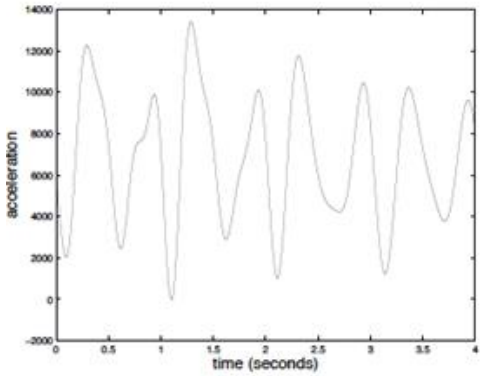
- **Problem:** vibrations (e.g. mowing lawn, plane taking off) could be counted as a step
- **Optimization:** Fix by exploiting periodicity of walking/running
- Assume people can:
  - **Run:** 5 steps per second => 0.2 seconds per step
  - **Walk:** 1 step every 2 seconds => 2 seconds per step
  - So, eliminate "negative crossings" that occur outside period [0.2 – 2 seconds] (e.g. vibrations)



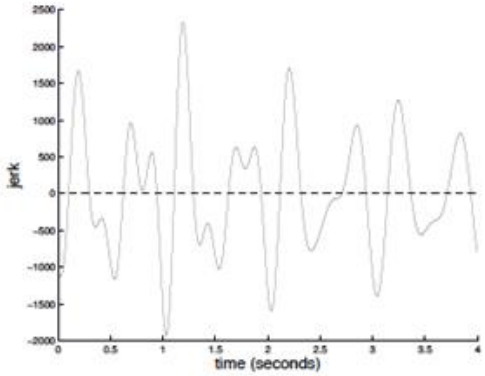
Step Detection Algorithms

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- Previous step detection algorithm is simple.
- Can use more sophisticated signal processing algorithms for smoothing
- Frequency domain processing (E.g. Fourier transform + low-pass filter)



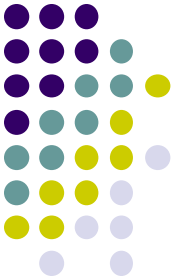
(c) Output of the low-pass filter.



(d) Derivative of the low-pass filter.

Estimate Distance Traveled

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter



- Calculate distance covered based on number of steps taken

Distance = number of steps × distance per step (1)

- Distance per step (stride) depends on user's height (taller people, longer strides)
- Using person's height, can estimate their stride, then number of steps taken per 2 seconds

Steps per 2 s	Stride (m/s)
0~2	Height/5
2~3	Height/4
3~4	Height/3
4~5	Height/2
5~6	Height/1.2
6~8	Height
>=8	1.2 × Height

Estimating Calories Burned

Ref: Deepak Ganesan, Ch 2 Designing a Pedometer and Calorie Counter

- To estimate speed, remember that speed = distance/time. Thus,

Speed (in m/s) = (no. steps per 2 s × stride (in meters))/2s (2)

- Can also convert to calorie expenditure, which depends on many factors E.g
  - Body weight, workout intensity, fitness level, etc
- Rough relationship given in table

Running Speed (km/h)	Calories Expended (C/kg/h)
8	10
12	15
16	20
20	25

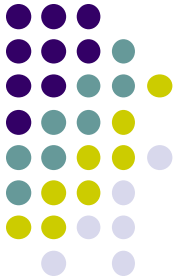
x / y = 1.25

- Expressed as an equation

Calories (C/kg/h) = 1.25 × running speed (km/h) (3)

- First convert from speed in km/h to m/s

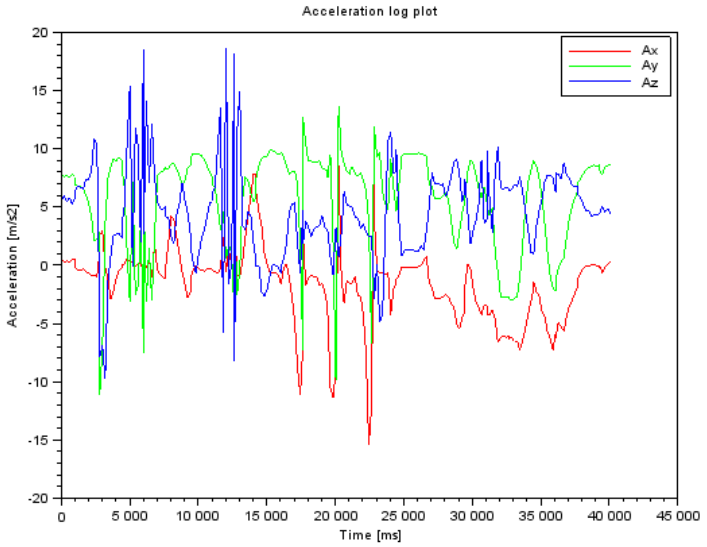
Calories (C/kg/h) = 1.25 × speed (m/s) × 3600/1000 = 4.5 × speed (m/s) (4)



Introduction to Activity Recognition

Activity Recognition

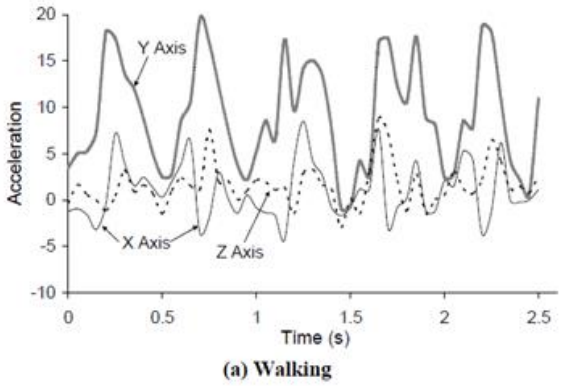
- **Goal:** Want our app to detect what activity the user is doing?
- **Classification task:** which of these 6 activities is user doing?
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing



- Typically, use machine learning classifers to classify user's accelerometer signals

Activity Recognition Overview

Gather Accelerometer data



(a) Walking

Machine Learning Classifier

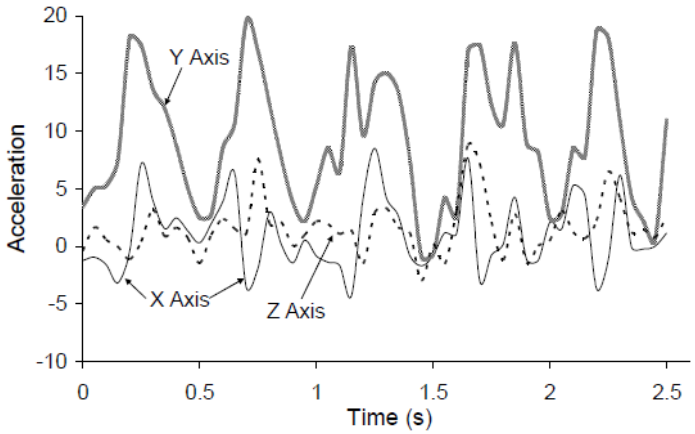
Classify Accelerometer data

Walking

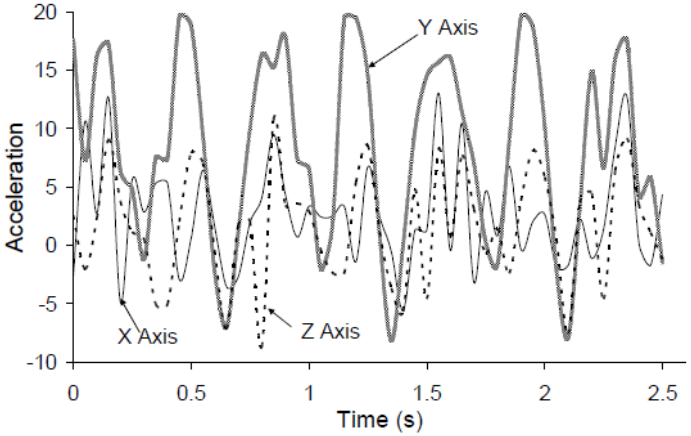
Running

Climbing Stairs

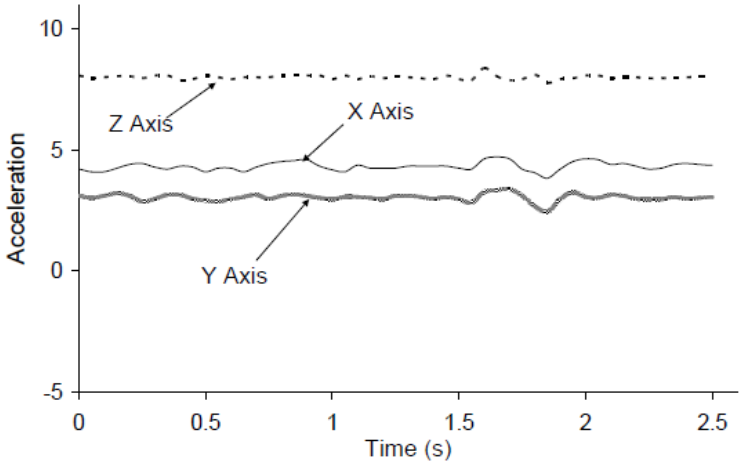
Example Accelerometer Data for Activities



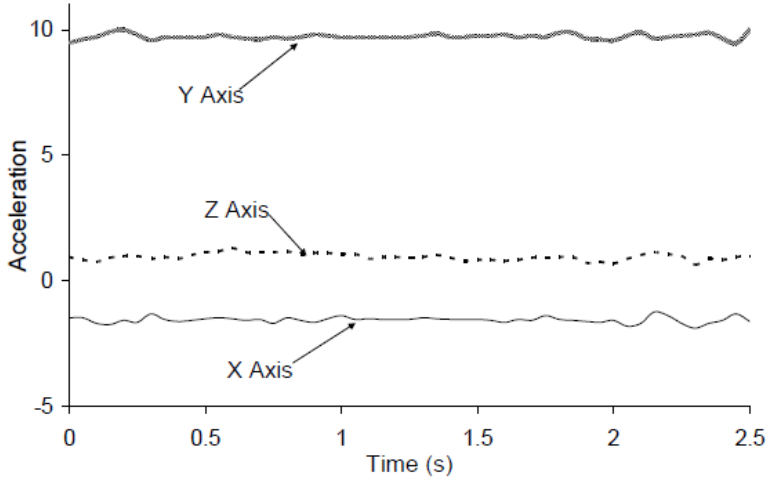
(a) Walking



(b) Jogging

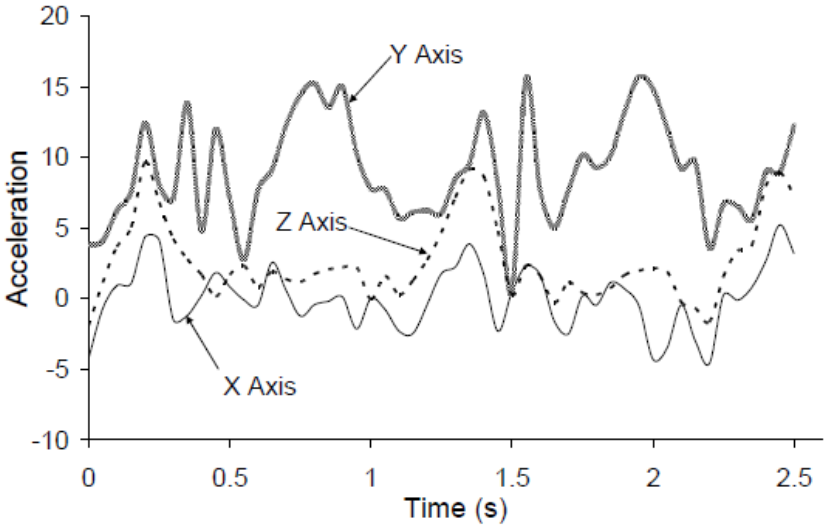


(e) Sitting

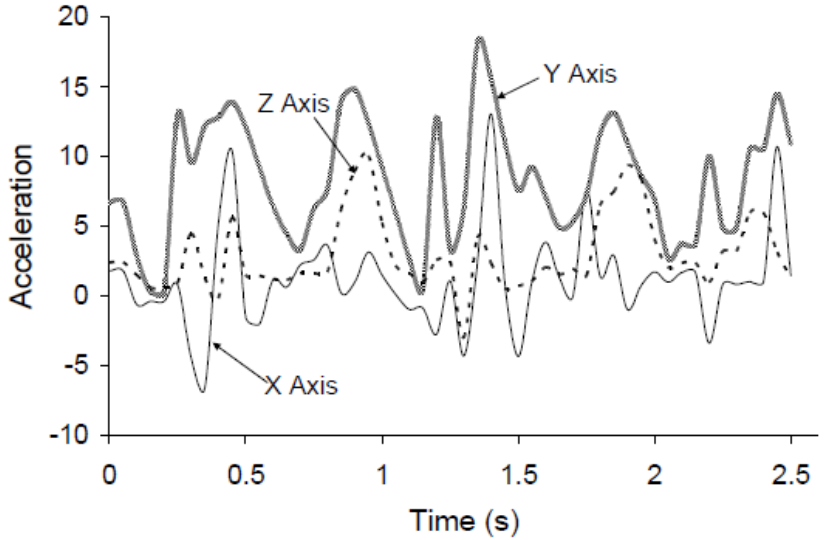


(f) Standing

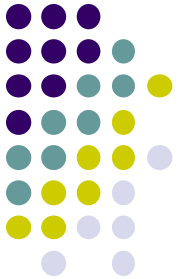
Example Accelerometer Data for Activities



(c) Ascending Stairs



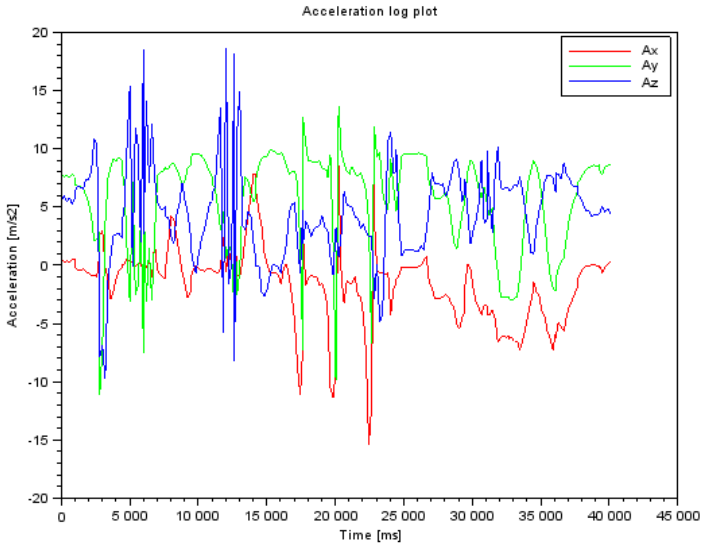
(d) Descending Stairs



Applications of Activity Recognition

Recall: Activity Recognition

- **Goal:** Want our app to detect what activity the user is doing?
- **Classification task:** which of these 6 activities is user doing?
  - Walking,
  - Jogging,
  - Ascending stairs,
  - Descending stairs,
  - Sitting,
  - Standing

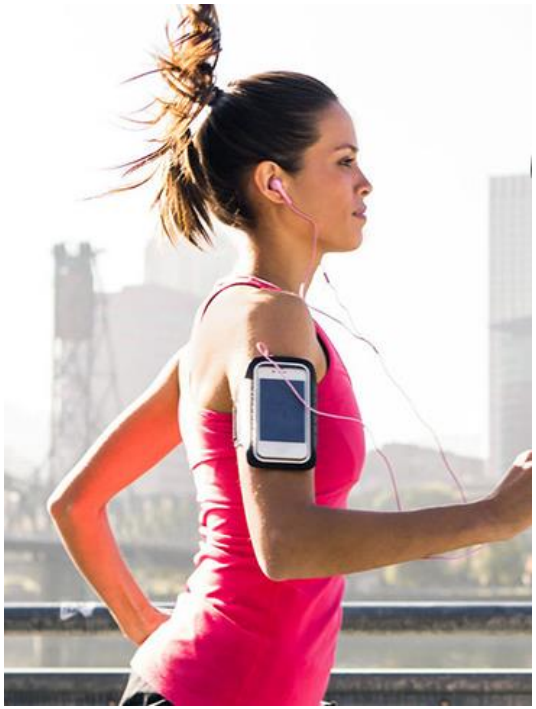


- Typically, use machine learning classifers to classify user's accelerometer signals

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Fitness Tracking:**
  - **Initially:**
    - Physical activity type,
    - Distance travelled,
    - Calories burned
  - **Newer features:**
    - Stairs climbed,
    - Physical activity (duration + intensity)
    - Activity type logging + context e.g. Ran 0.54 miles/hr faster during morning runs
    - Sleep tracking
    - Activity history



Note: AR refers to algorithm
But could run on a range of devices (smartphones, wearables, e.g. fitbit)

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Health monitoring:** How **well** is patient performing activity?
- Make clinical monitoring pervasive, continuous, real world!!
  - Gather context information (e.g. what makes condition worse/better?)
  - E.g. timed up and go test
- Show patient contexts that worsen condition => Change behavior
  - E.g. walking in narror hallways worsens gait freeze

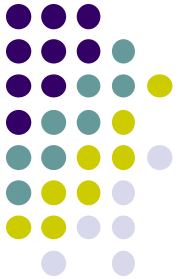
Parkinsons disease
Gait freezing

Question: What data would you need to build PD gait classifier? From what types of subjects?

COPD, Walk tests in the wild

Applications of Activity Recognition

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

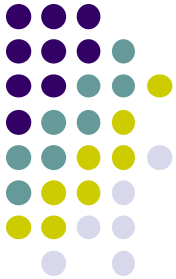


- **Fall:** Leading cause of death for seniors
- **Fall detection:** Smartphone/watch, wearable detects senior who has fallen, alert family
  - Text message, email, call relative

Fall detection + prediction

Applications of Activity Recognition (AR)

Ref: Lockhart *et al,* Applications of Mobile Activity recognition



- **Context-Aware Behavior:**
  - In-meeting? => Phone switches to silent mode
  - Exercising? => Play song from playlist, use larger font sizes for text
  - Arrived at work? => download email

- Study found that messages delivered when transitioning between activities better received

- **Adaptive Systems to Improve User Experience:**
  - Walking, running, riding bike? => Turn off Bluetooth, WiFi (save power)
  - Can increase battery life up to 5x

Applications of AR

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Smart home:**
  - Determine what activities people in the home are doing,
    - **Why?** infer illness, wellness, patterns, intrusion (security), etc
    - E.g. TV automatically turns on at about when you usually lie on the couch

Applications of AR: 3rd Party Apps

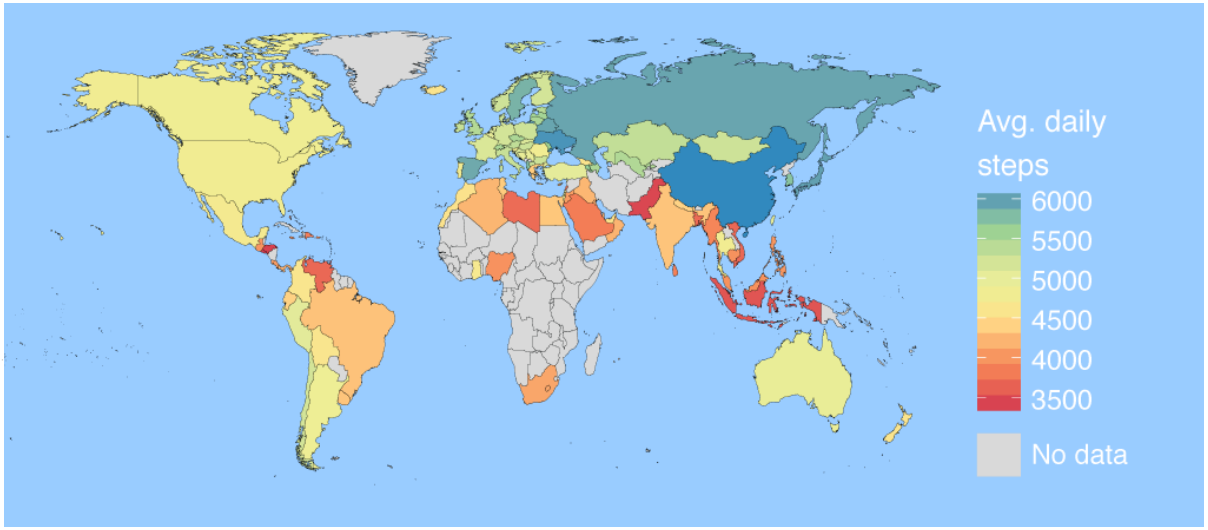
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Targeted Advertising:**
  - AR helps deliver more relevant ads
  - E.g user runs a lot => Get exercise clothing ads
  - Goes to pizza places often + sits there => Get pizza ads

Applications of AR: 3rd Party Apps

Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Research Platforms for Data Collection:**
  - E.g. public health officials want to know how much time various people (e.g. students) spend sleeping, walking, exercising, etc
  - Mobile AR: inexpensive, automated data collection
  - E.g. Stanford Inequality project: Analyzed physical activity of 700k users in 111 countries using smartphone AR data
  - http://activityinequality.stanford.edu/



Applications of AR: 3rd Party Apps

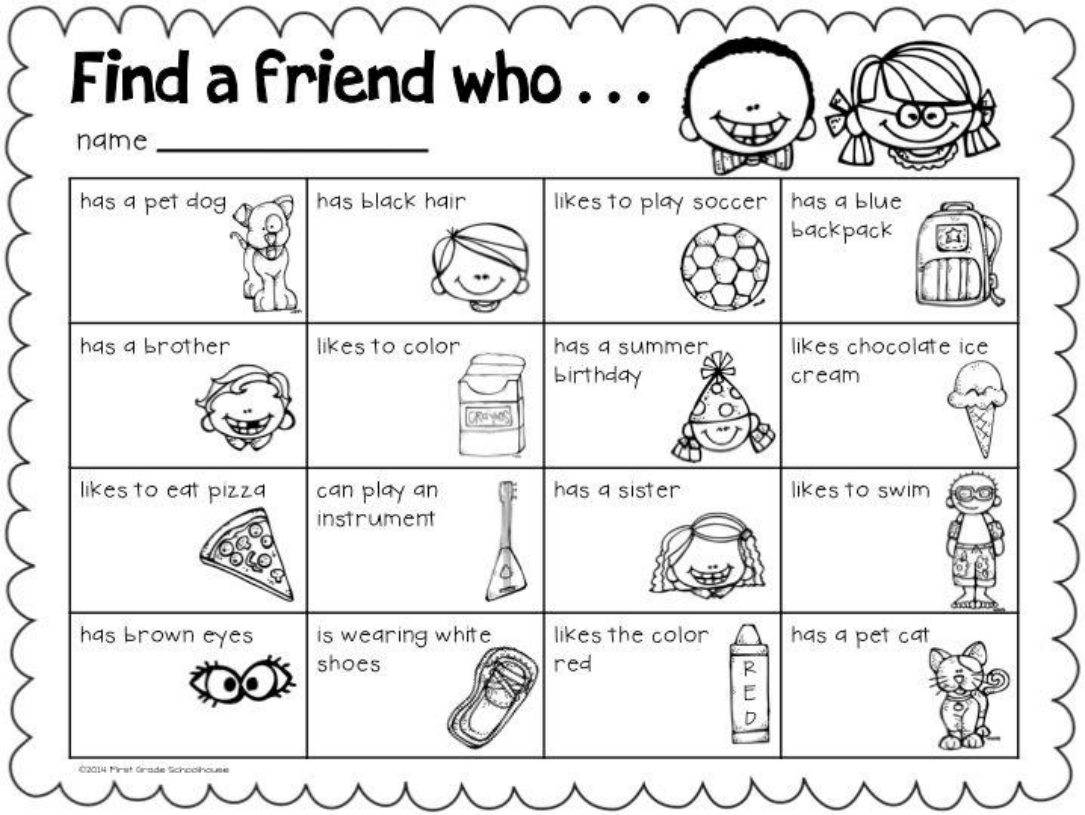
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Track, manage staff on-demand:**
  - E.g. at hospital, determine “availability of nurses”, assign them to new jobs/patients/surgeries/cases

Applications of AR: Social Networking

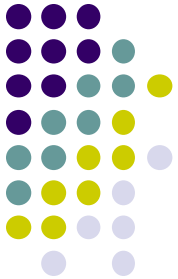
Ref: Lockhart *et al,* Applications of Mobile Activity recognition

- **Activity-Based Social Networking:**
  - Automatically connect users who do same activities + live close together



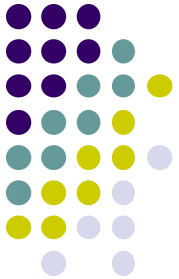
Applications of AR: Social Networking

Ref: Lockhart *et al,* Applications of Mobile Activity recognition



- **Activity-Based Place Tagging:**
  - Automatically “popular” places where users perform same activity
  - E.g. Park street is popular for runners (activity-based maps)

- **Automatic Status updates:**
  - E.g. Bob is sleeping
  - Tracy is jogging along Broadway with track team
  - Privacy/security concerns => Different Levels of details for different friends



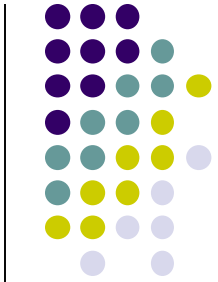
Activity Recognition Using Google API

Activity Recognition

- Activity Recognition? Detect what user is doing?
  - Part of user's context
- Examples: sitting, running, driving, walking
- Why? App can adapt it's behavior based on user behavior
- **E.g.** If user is driving, don't send notifications

https://www.youtube.com/watch?v=S8sugXgUVEI

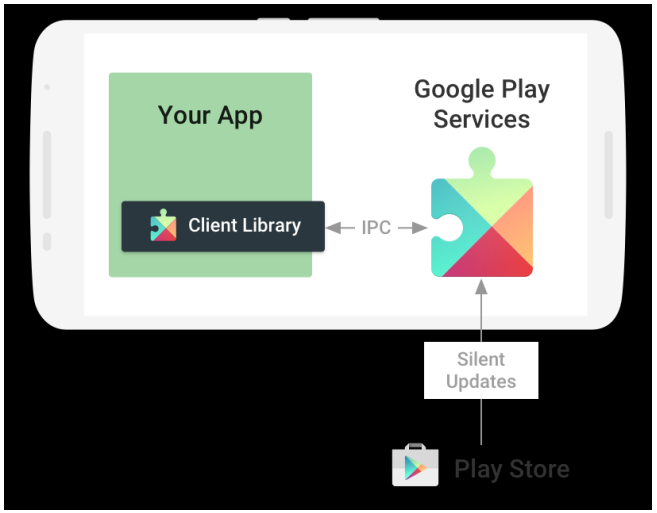
Google Activity Recognition API

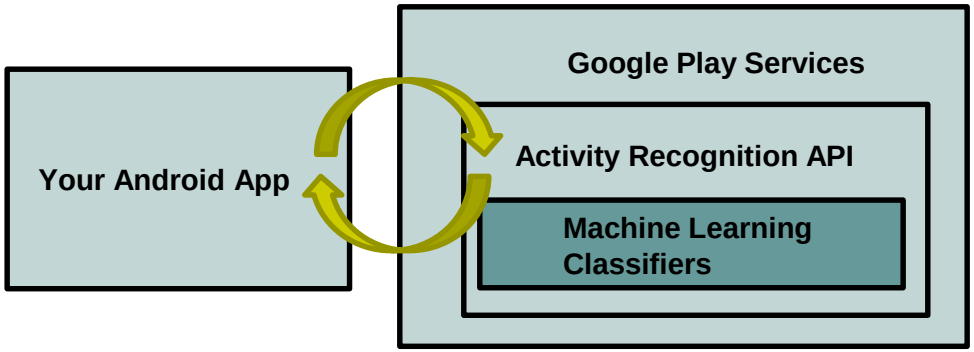


- API to detect smartphone user's current activity
- Programmable, can be used by your Android app
- Currently detects 8 states:
  - In vehicle
  - On Bicycle
  - On Foot
  - Running
  - Walking
  - Still
  - Tilting
  - Unknown

Google Activity Recognition API

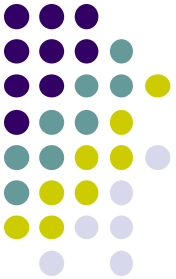
- Deployed as part of Google Play Services





Activity Recognition Using AR API

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



- Example code for this tutorial on gitHub:

  https://github.com/tutsplus/Android-ActivityRecognition

- Google Activity Recognition can:
  - Recognize user's current activity (Running, walking, in a vehicle or still)

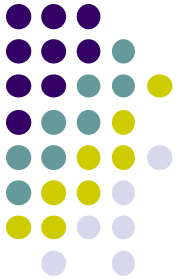
- Project Setup:
  - Create Android Studio project with blank Activity (minimum SDK 14)
  - In **build.gradle** file, define latest Google Play services (now 11.8) as dependency

```
compile 'com.google.android.gms:play-services:8.4.0'
```

Now currently Version 11.8.0

Activity Recognition Using AR API

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials



- Create new class **ActivityRecognizedService** which extends **IntentService**
- **IntentService:** type of service, asynchronously handles work off main thread
- Throughout user's day, **Activity Recognition API** sends user's activity to this IntentService in the background
- Need to program this Intent to handle incoming user activity

```
01  public class ActivityRecognizedService extends IntentService {
02
03      public ActivityRecognizedService() {
04          super("ActivityRecognizedService");
05      }
06
07      public ActivityRecognizedService(String name) {
08          super(name);
09      }
10
11      @Override
12      protected void onHandleIntent(Intent intent) {
13      }
14  }
```

Called by Android OS to deliver User's activity

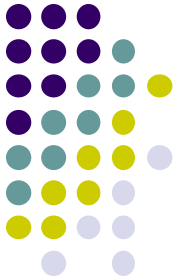
Activity Recognition Using AR API

Ref: How to Recognize User Activity with Activity Recognition by Paul Trebilcox-Ruiz on Tutsplus.com tutorials

- Modify **AndroidManifest.xml** to
  - Declare **ActivityRecognizedService**
  - Add com.google.android.gms.permission.ACTIVITY_RECOGNITION permission

```
01<?xml version="1.0" encoding="utf-8"?>
02<manifest xmlns:android="http://schemas.android.com/apk/res/android"
03    package="com.tutsplus.activityrecognition">
04
05    <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
06
07    <application
08        android:icon="@mipmap/ic_launcher"
09        android:label="@string/app_name"
10        android:theme="@style/AppTheme">
11        <activity android:name=".MainActivity">
12            <intent-filter>
13                <action android:name="android.intent.action.MAIN" />
14
15                <category android:name="android.intent.category.LAUNCHER" />
16            </intent-filter>
17        </activity>
18
19        <service android:name=".ActivityRecognizedService" />
20    </application>
21
22</manifest>
```

Requesting Activity Recognition



- In **MainActivity.java**, To connect to Google Play Services:
  - Provide **GoogleApiClient** variable type + implement callbacks

```
01 public class MainActivity extends AppCompatActivity implements GoogleApiClient.ConnectionCallbacks,
02 GoogleApiClient.OnConnectionFailedListener {
03
04     public GoogleApiClient mApiClient;
05
06     @Override
07     protected void onCreate(Bundle savedInstanceState) {
08         super.onCreate(savedInstanceState);
09         setContentView(R.layout.activity_main);
10     }
11
12     @Override
13     public void onConnected(@Nullable Bundle bundle) {
14
15     }
16
17     @Override
18     public void onConnectionSuspended(int i) {
19
20     }
21
22     @Override
23     public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
24
25     }
}
```

Handle to Google Activity Recognition client

Normal AR call if everything working well

Called if sensor (accelerometer) connection fails

Called if Google Play connection fails

Requesting Activity Recognition

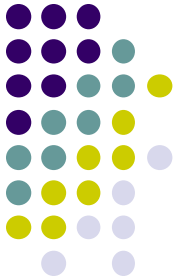
- In onCreate, initialize client and connect to Google Play Services

```
01  @Override
02  protected void onCreate(Bundle savedInstanceState) {
03      super.onCreate(savedInstanceState);
04      setContentView(R.layout.activity_main);
05
06      mApiClient = new GoogleApiClient.Builder(this)
07              .addApi(ActivityRecognition.API)
08              .addConnectionCallbacks(this)
09              .addOnConnectionFailedListener(this)
10              .build();
11
12      mApiClient.connect();
13  }
```

Request ActivityRecognition.API

Associate listeners with our instance of GoogleApiClient

Handling Activity Recognition



- Simply log each detected activity and display how confident Google Play services is that user is performing this activity

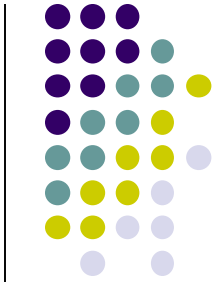
```
private void handleDetectedActivities(List<DetectedActivity> probableActivities) {
    for( DetectedActivity activity : probableActivities ) {
        switch( activity.getType() ) {
            case DetectedActivity.IN_VEHICLE: {
                Log.e( "ActivityRecogition", "In Vehicle: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.ON_BICYCLE: {
                Log.e( "ActivityRecogition", "On Bicycle: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.ON_FOOT: {
                Log.e( "ActivityRecogition", "On Foot: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.RUNNING: {
                Log.e( "ActivityRecogition", "Running: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.STILL: {
                Log.e( "ActivityRecogition", "Still: " + activity.getConfidence() );
                break;
            }
            case DetectedActivity.TILTING: {
                Log.e( "ActivityRecogition", "Tilting: " + activity.getConfidence() );
                break;
            }
```

Switch statement on activity type

Sample output

```
1  E/ActivityRecogition: On Foot: 92
2  E/ActivityRecogition: Running: 87
3  E/ActivityRecogition: On Bicycle: 8
4  E/ActivityRecogition: Walking: 5
```

Handling Activity Recognition



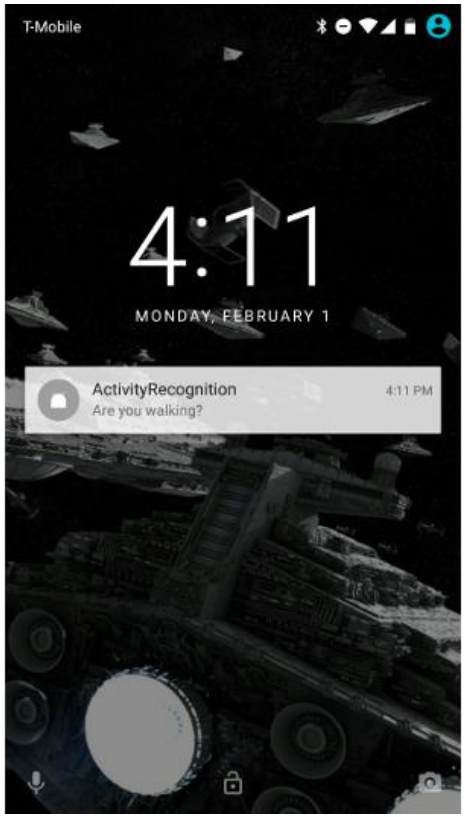
- If confidence is > 75, activity detection is probably accurate
- If user is walking, ask “Are you walking?”

```
            case DetectedActivity.WALKING: {
                Log.e( "ActivityRecogition", "Walking: " + activity.getConfidence() );
                if( activity.getConfidence() >= 75 ) {
                    NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
                    builder.setContentText( "Are you walking?" );
                    builder.setSmallIcon( R.mipmap.ic_launcher );
                    builder.setContentTitle( getString( R.string.app_name ) );
                    NotificationManagerCompat.from(this).notify(0, builder.build());
                }
                break;
            }
            case DetectedActivity.UNKNOWN: {
                Log.e( "ActivityRecogition", "Unknown: " + activity.getConfidence() );
                break;
            }
        }
    }
}
```

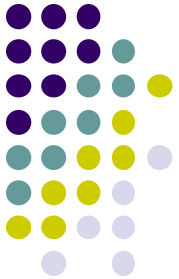
Sample Output of Program

- Sample displayed on development console

```
1  E/ActivityRecogition: On Foot: 92
2  E/ActivityRecogition: Running: 87
3  E/ActivityRecogition: On Bicycle: 8
4  E/ActivityRecogition: Walking: 5
```



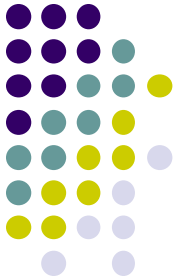
- Full code at: https://github.com/tutsplus/Android-ActivityRecognition



Android Awareness API

Awareness API

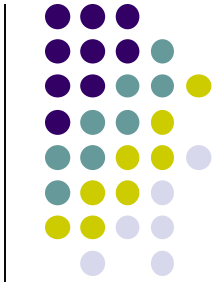
https://developers.google.com/awareness/overview



- Single Android API for context awareness released in 2016
- Combines some APIs already covered (Place, Activity, Location)

Context type	Example
Time	Current local time
Location	Latitude and longitude
Place	Place, including place type
Activity	Detected user activity (walking, running, biking)
Beacons	Nearby beacons matching the specified namespace
Headphones	Are headphones plugged in?
Weather	Current weather conditions

Awareness API



- **Snapshot API:**
  - Return cached values (Nearby Places, weather, Activity, etc)
  - System caches values
  - Optimized for battery and power consumption

- **Fences API:**
  - Used to set conditions to trigger events
  - E.g. if(user enters a geoFence & Activity = running) notify my app

- Good tutorials for Awareness API:
  - Google Play Services: Awareness API by Paul Trebilcox-Ruiz

    https://code.tutsplus.com/tutorials/google-play-services-awareness-api--cms-25858
  - Exploring the Awareness API by Joe Birch

    https://medium.com/exploring-android/exploring-the-new-google-awareness-api-bf45f8060bba

References

- Android Sensors Overview, http://developer.android.com/guide/topics/sensors/sensors_overview.html
- 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