CS 528 Mobile and Ubiquitous Computing

Lecture 6b: Ubicomp: Sensors, step counting, HAR

Emmanuel Agu

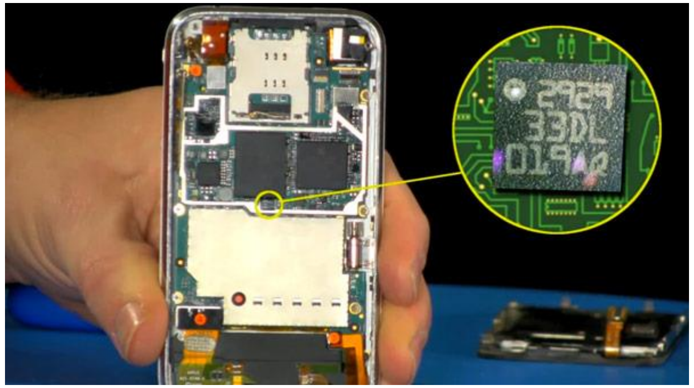
Administrivia

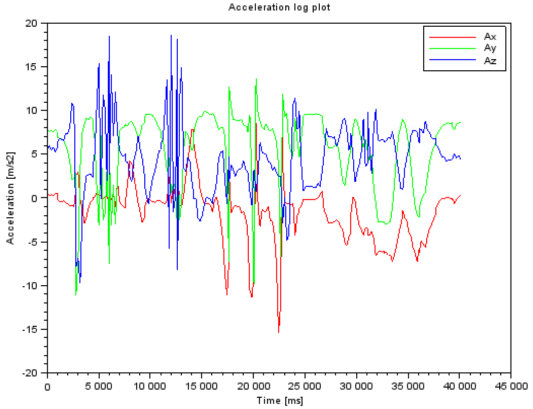
- Groups should submit 1-slide on their final project (due next class)
- Quiz
  - Covers lectures 5-6
  - All code in those lectures handed out
  - Papers and handouts
- Project 3 posted
  - I’ve covered everything you need to do it EXCEPT Activity Recognition (Next week)

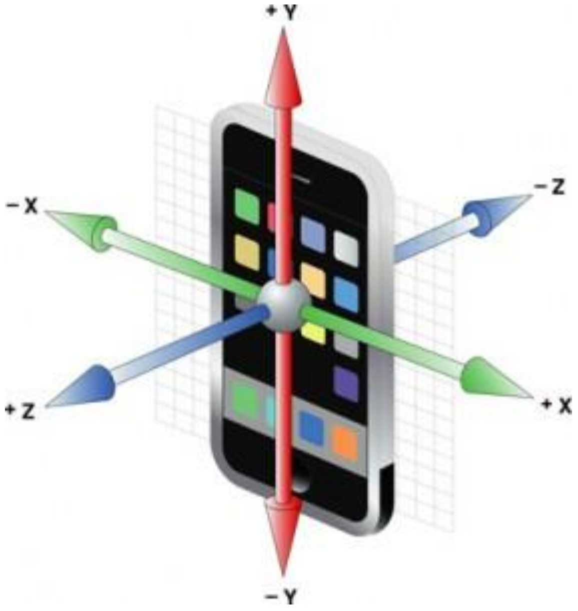
Android Sensors

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

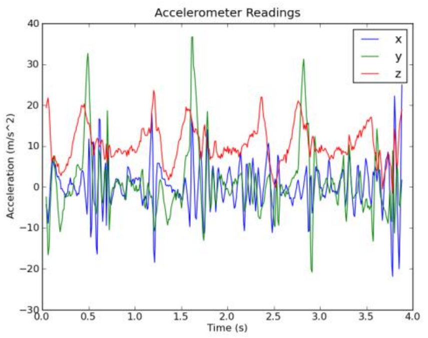




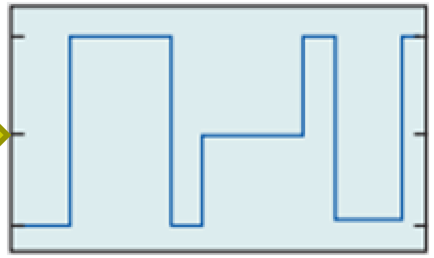


So What?

- 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

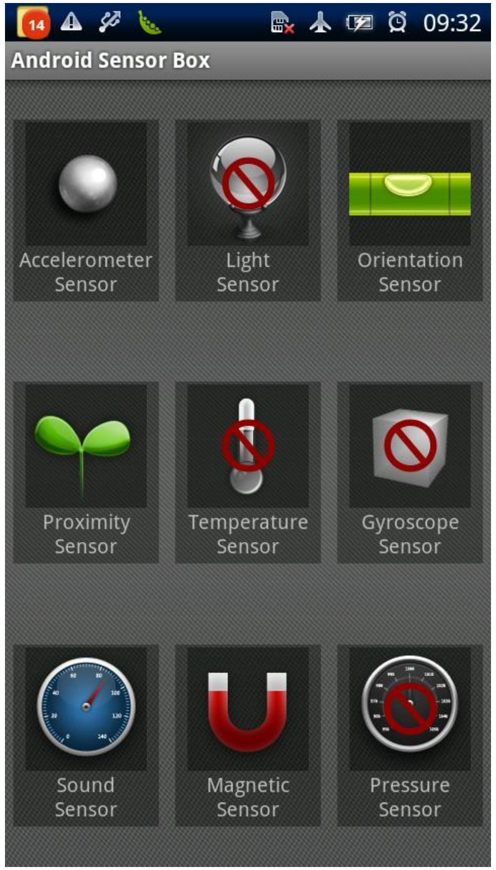
Walking
Running
Jumping
Step count
Calories burned
Falling

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

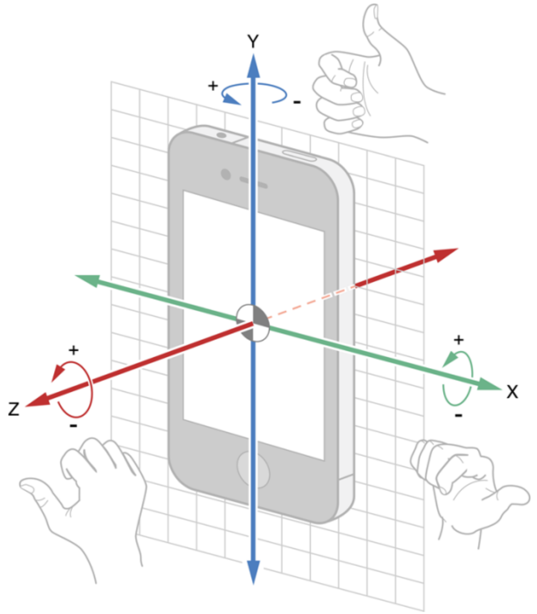
- **Hardware sensor:**
  - physical components built into phone,
  - **Example:** temperature

- **Software sensor (or virtual sensor):**
  - Not physical device
  - Derives their data from one or more hardware sensors (a formula)
  - **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 step counting available through Google Play Services (more later)

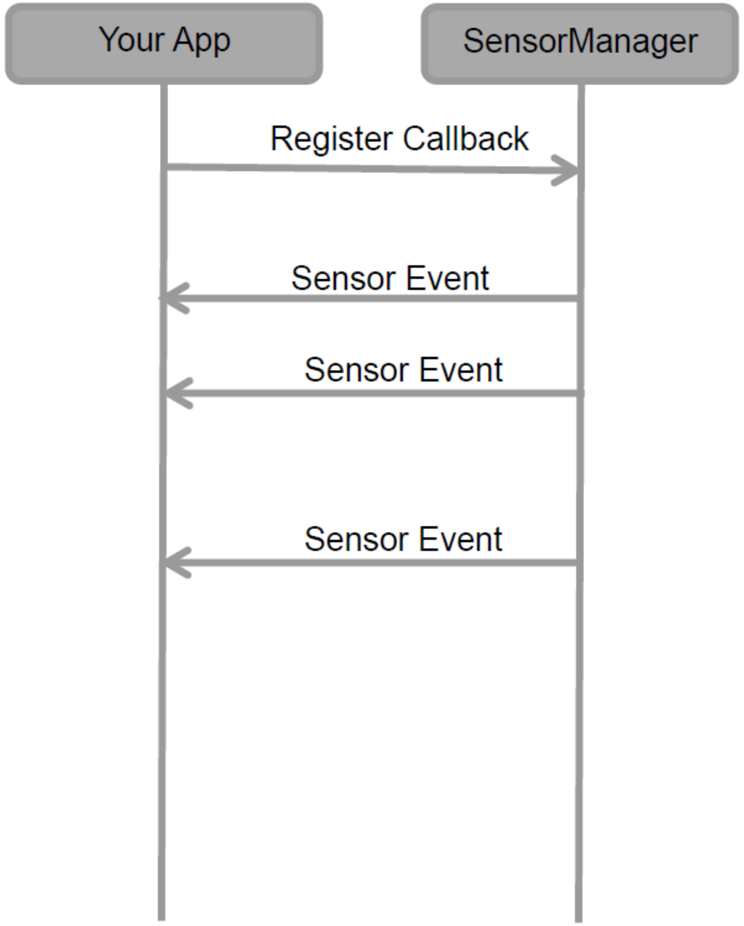
Sensor Programming

- 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)

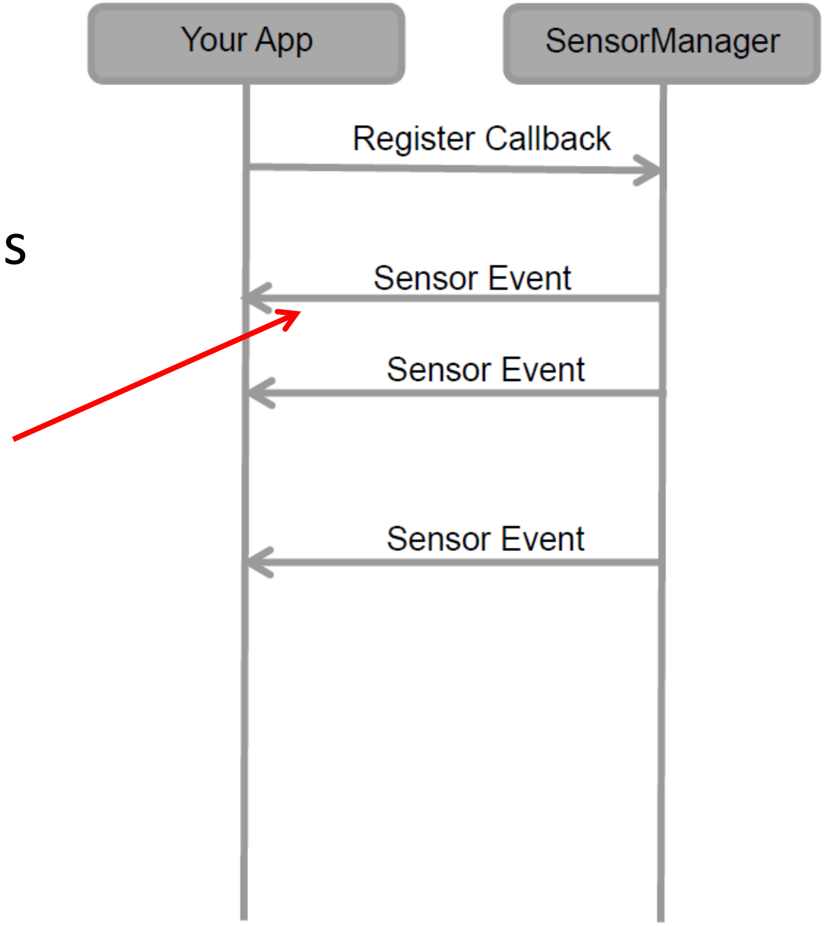


Sensor

- A class that can be used to create instance of a specific sensor
  - E.g instance of accelerometer

- Has methods used to determine a sensor's capabilities

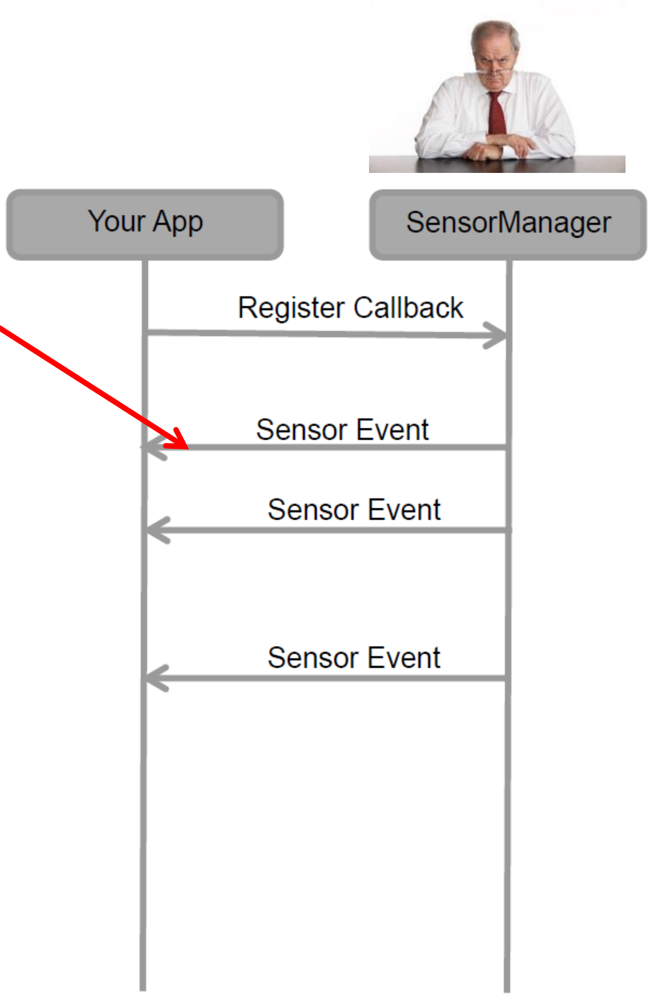
- Included in sensor event object



SensorEvent

- Android system sends 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
  - If multiple sensors of 1 type, choose 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
    - E.g. change in phone's rotational velocity triggers gyroscope sensor event

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

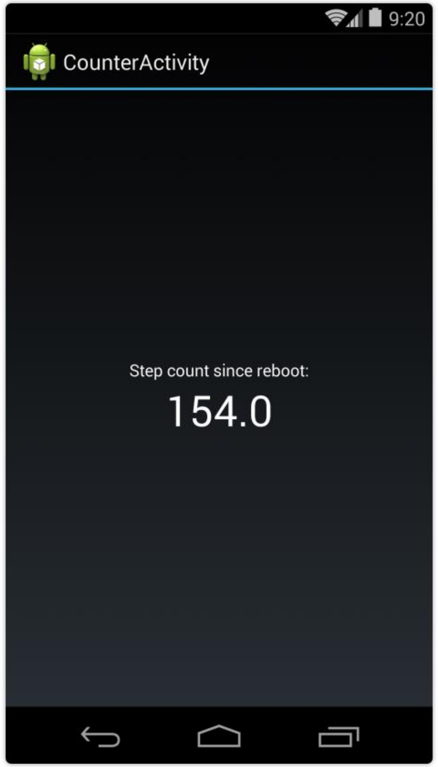
- Following code checks if device has at least one pressure sensor

```
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

```java
package com.starboardland.pedometer;

import android.app.Activity;
import android.content.Context;
import android.hardware.*;
import android.os.Bundle;
import android.widget.TextView;
import android.widget.Toast;

public class CounterActivity extends Activity implements SensorEventListener {

    private SensorManager sensorManager;
    private TextView count;
    boolean activityRunning;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        count = (TextView) findViewById(R.id.count);

        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    }
```



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

Example Step Counter App (Contd)

```java
    @Override
    protected void onResume() {
        super.onResume();
        activityRunning = true;
        Sensor countSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
        if (countSensor != null) {
            sensorManager.registerListener(this, countSensor, SensorManager.SENSOR_DELAY_UI);
        } else {
            Toast.makeText(this, "Count sensor not available!", Toast.LENGTH_LONG).show();
        }

    }

    @Override
    protected void onPause() {
        super.onPause();
        activityRunning = false;
        // if you unregister the last listener, the hardware will stop detecting step events
//          sensorManager.unregisterListener(this);
    }
```

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

Example Step Counter App (Contd)

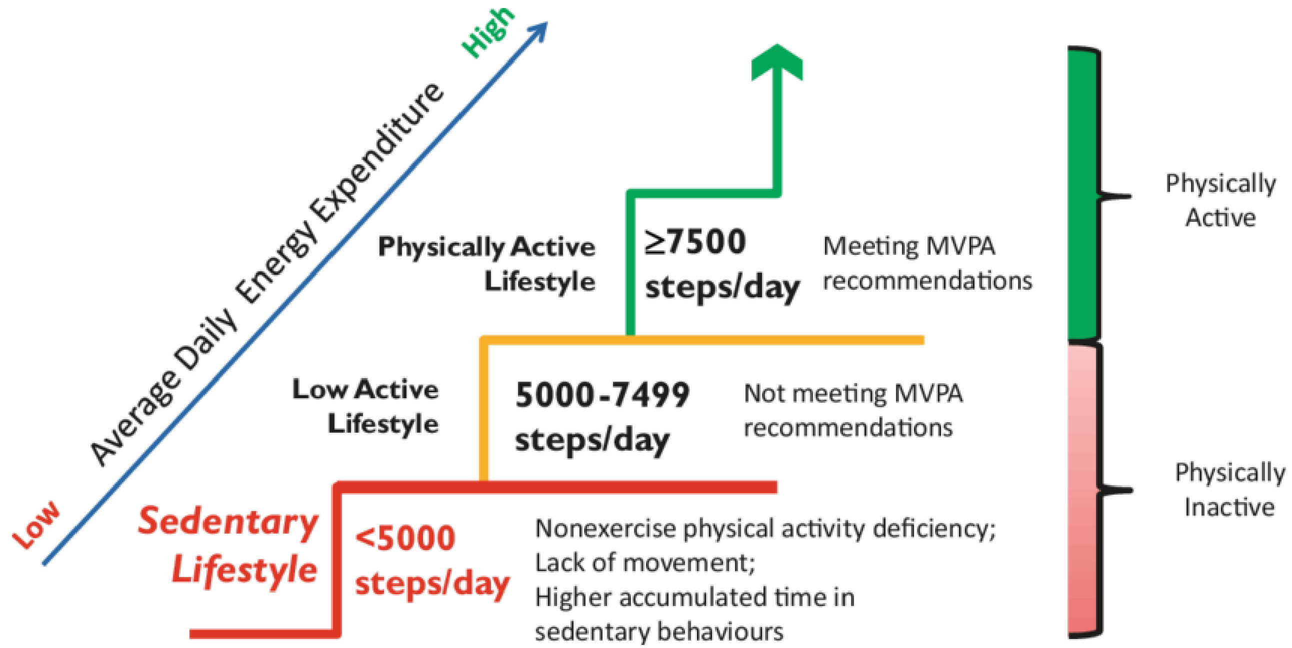
```java
    @Override
    public void onSensorChanged(SensorEvent event) {
        if (activityRunning) {
            count.setText(String.valueOf(event.values[0]));
        }

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }
}
```

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

Step Counting
(How Step Counting Works)

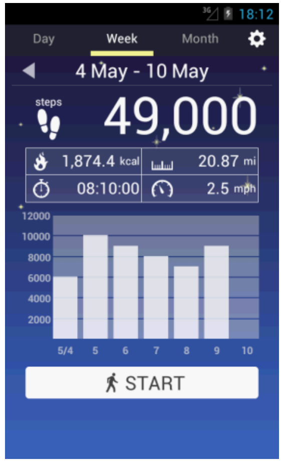
Sedentary Lifestyle

- 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

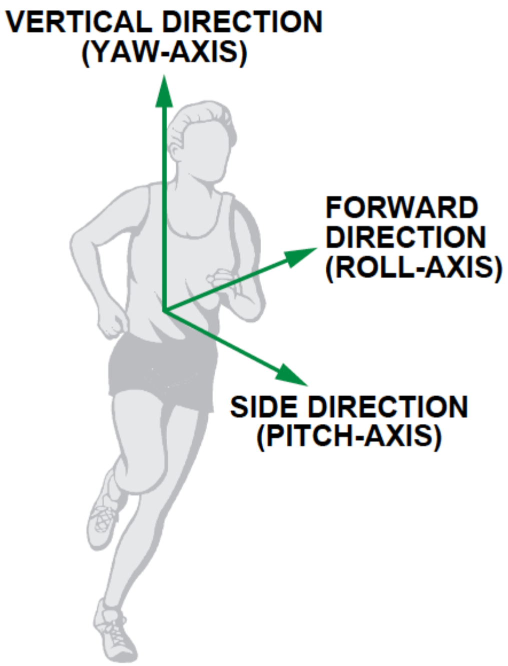
- Everyone is crazy about step count these days
- Pedometer apps, pedometers, fitness trackers, etc
- Tracking makes user aware of activity levels, motivates them to exercise more

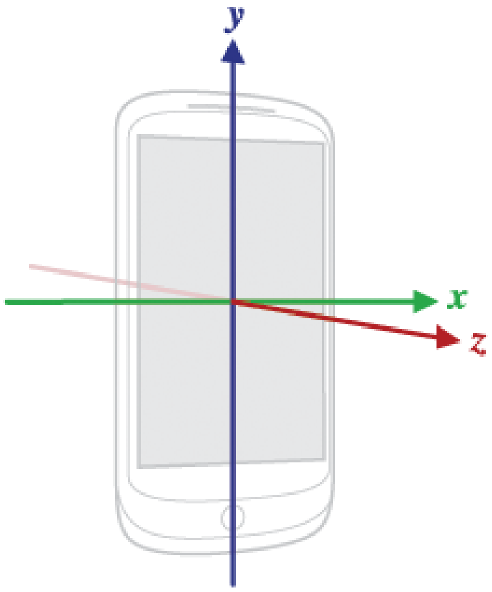


How does a Pedometer Detect/Count Steps

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

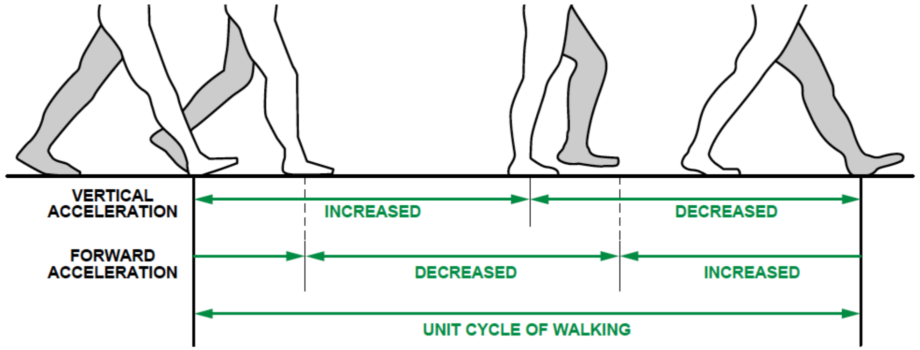


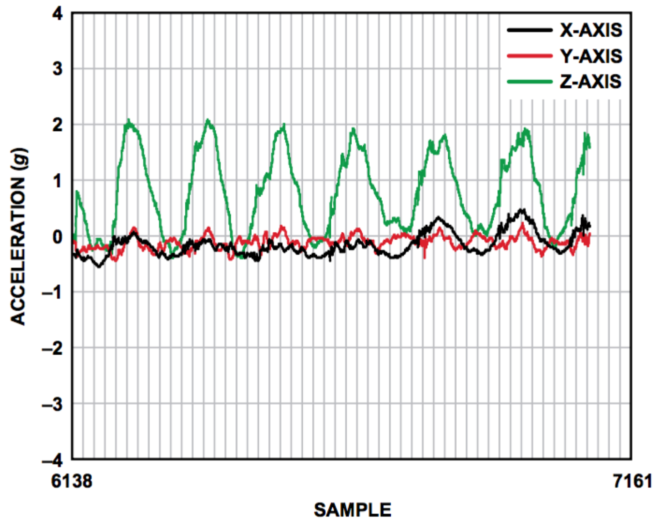


The Nature of Walking

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

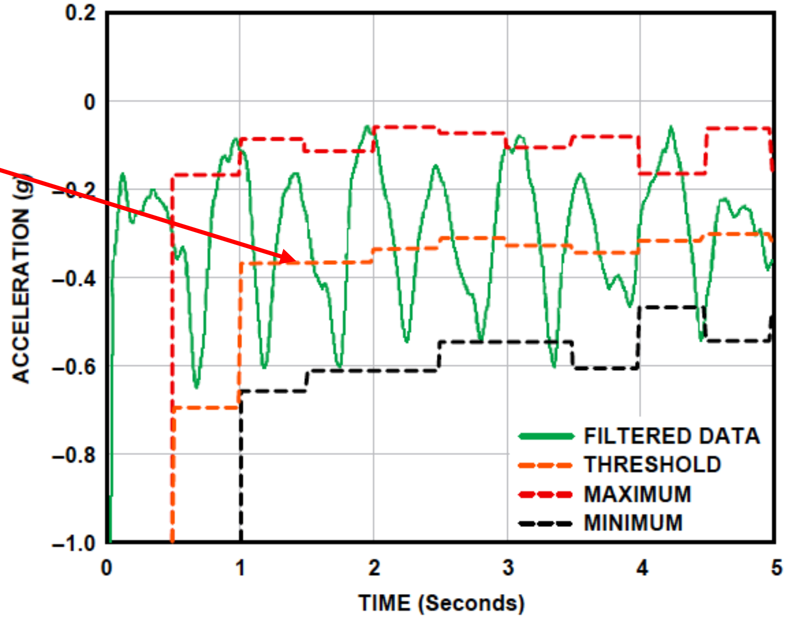




Step Detection Algorithm

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

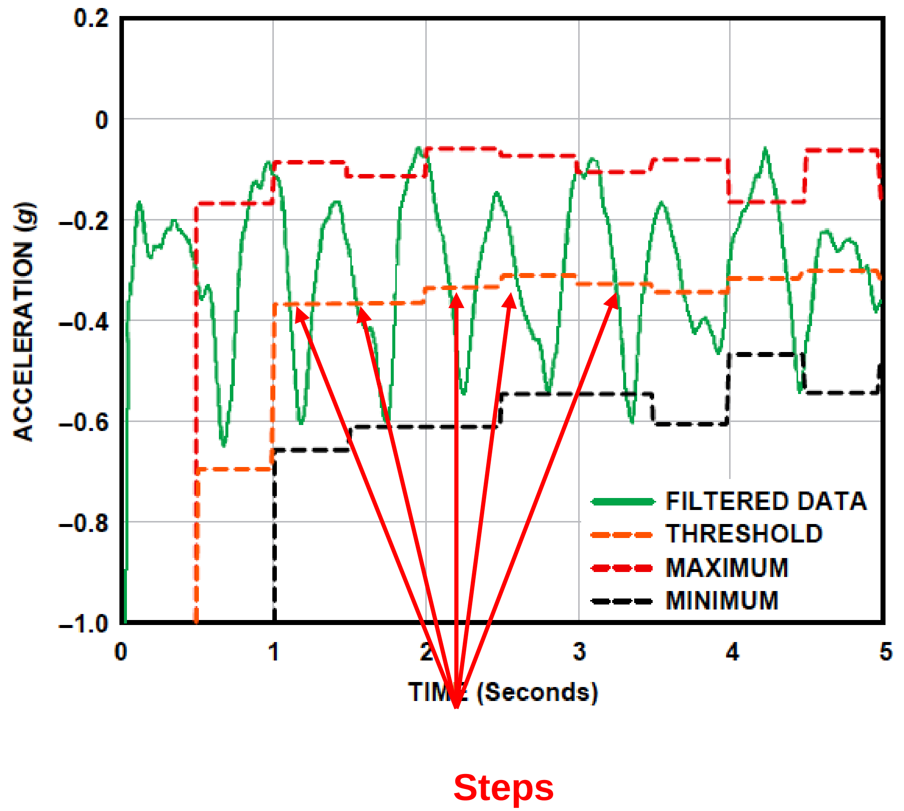
- **Step 1: smoothing**
  - Signal looks choppy
  - Smooth by replacing each sample with average of current, prior and next sample (Window of 3)
- **Step 2: Dynamic Threshold Detection**
  - Focus on accelerometer axis with largest peak
  - Would like a threshold such that each crossing is a step
  - 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

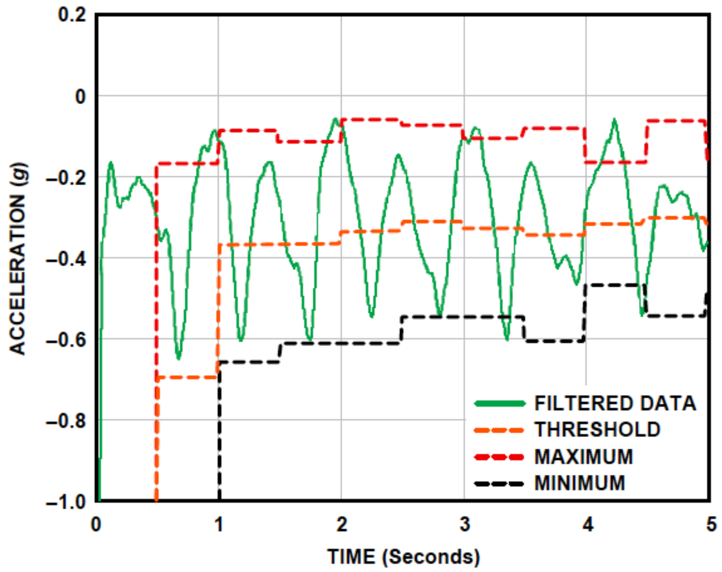
- A step is
  - indicated by crossings of dynamic threshold
  - Defined as negative slope (sample_new < sample_old) when smoothed waveform crosses dynamic threshold



Step Detection Algorithms

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

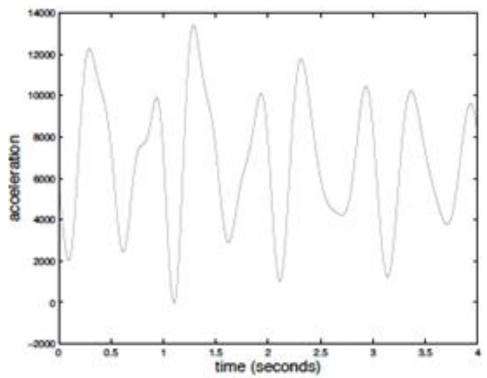
- **Problem:** vibrations (e.g. mowing lawn, plane taking off) could be counted as a step
- **Optimization:** Fix by exploiting periodicity of walking/running
- Assume people can:
  - **Run:** 5 steps per second => 0.2 seconds per step
  - **Walk:** 1 step every 2 seconds => 2 seconds per step
  - So,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



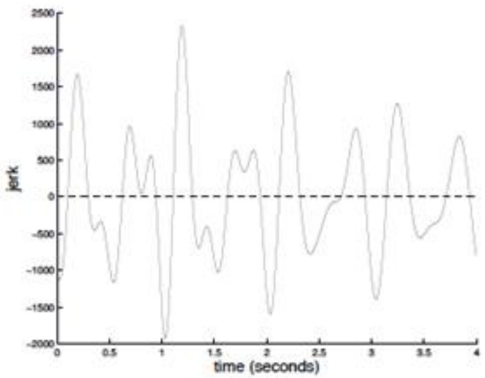
Step Detection Algorithms

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)

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

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