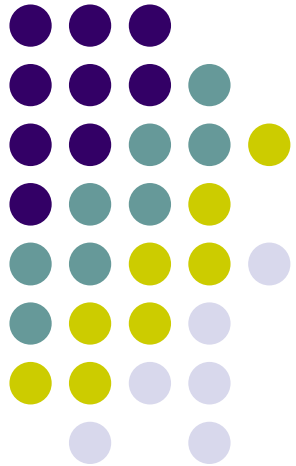
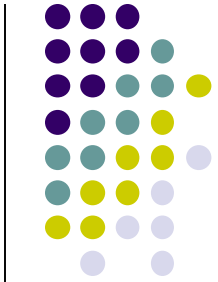
CS 528 Mobile and Ubiquitous Computing

Lecture 6a: Maps & Sensors

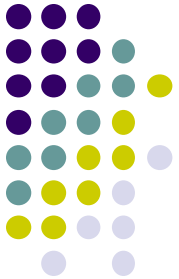
Emmanuel Agu



Administrivia



- Groups should submit 1-slide on their final project (due next class)

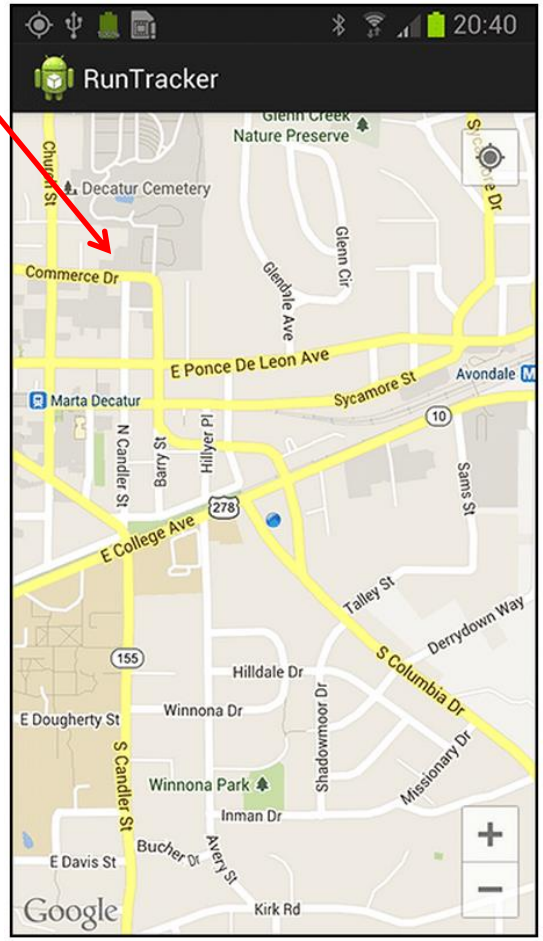


Using Maps

MapView and MapActivity

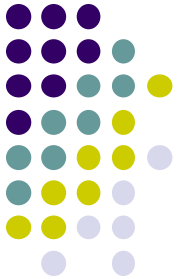
- **MapView:** UI widget that displays maps

- **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!!)
   - https://developer.android.com/studio/index.html
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

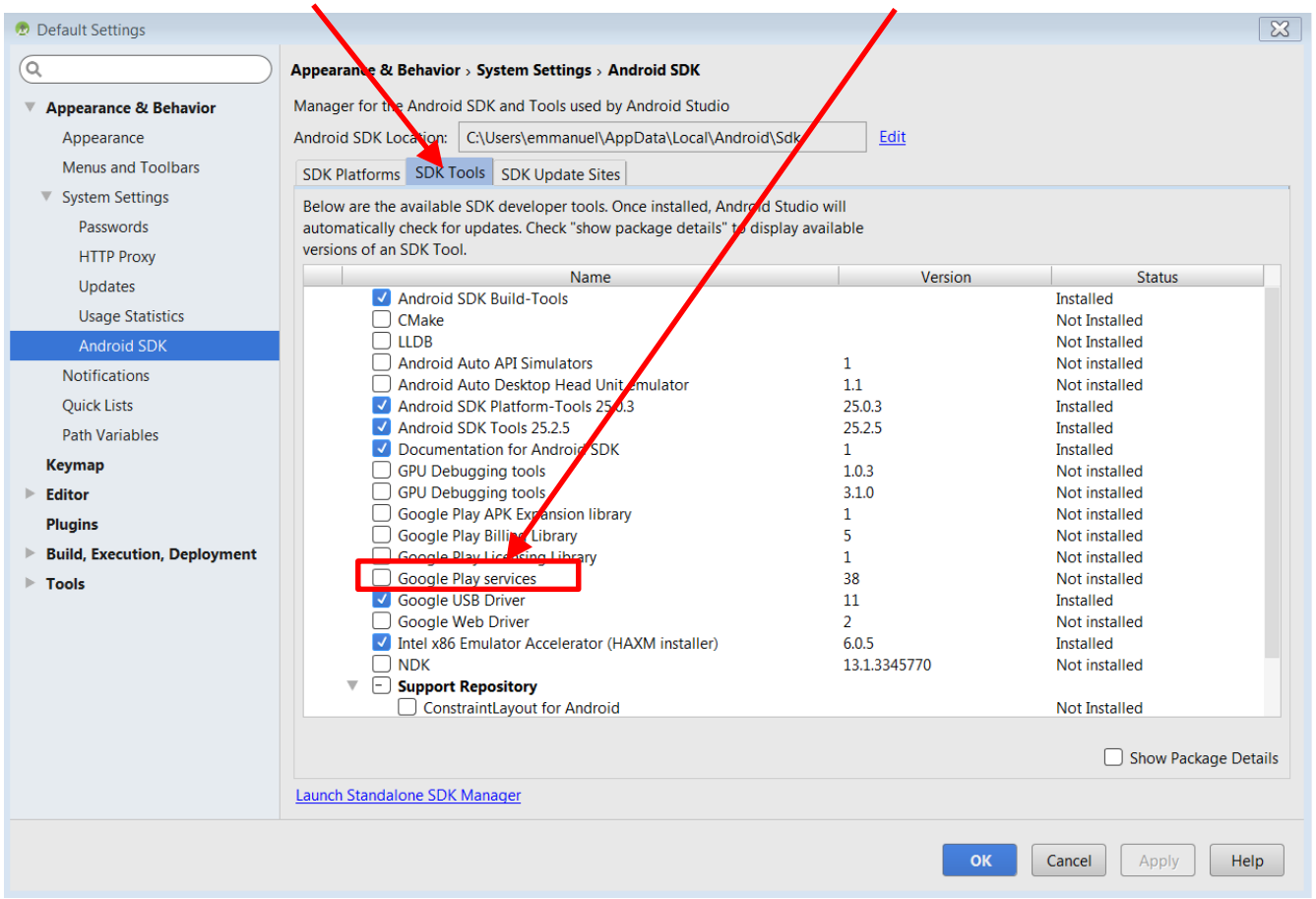
Step 2: Add Google Play Services to Android Studio

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

- 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

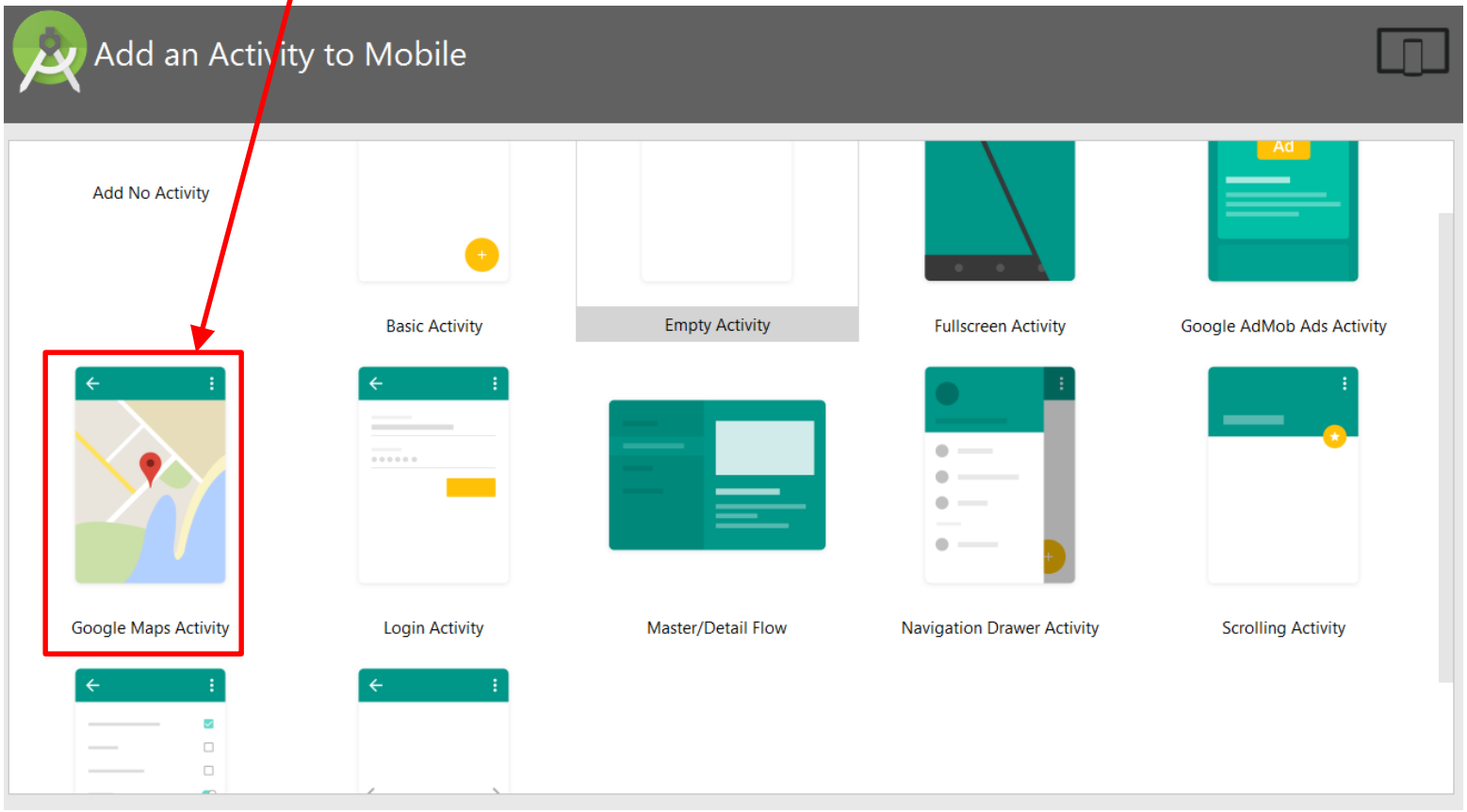
Check Google Play Services, then Ok



Step 3: Create new Android Studio Project

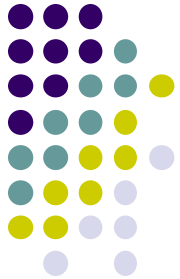
https://developers.google.com/maps/documentation/android-api/start

- Select “Google Maps Activity, click Finish



Step 4: Get Google Maps API key

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

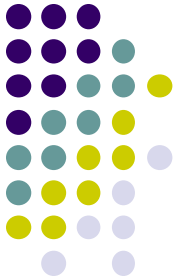


- To access Google Maps servers using Maps API, must add Maps API key to app
- Maps API key is free. E.g.

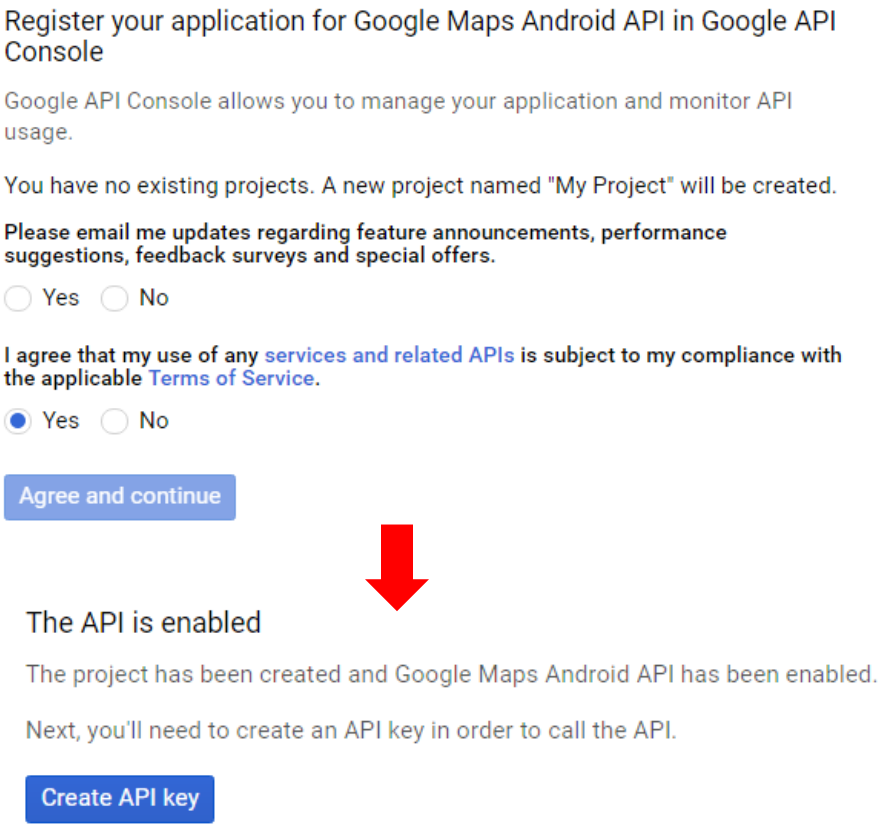
- Google uses API key to uniquely identify your app, track its resource usage, etc

Step 4a: Fast, Easy way to get Maps API Key

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

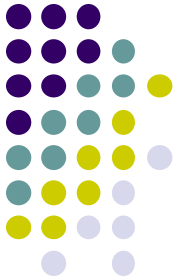


- Copy link provided in **google_maps_api.xml** of Maps template into browser
- Goes to Google API console, auto-fills form
- Creates API key

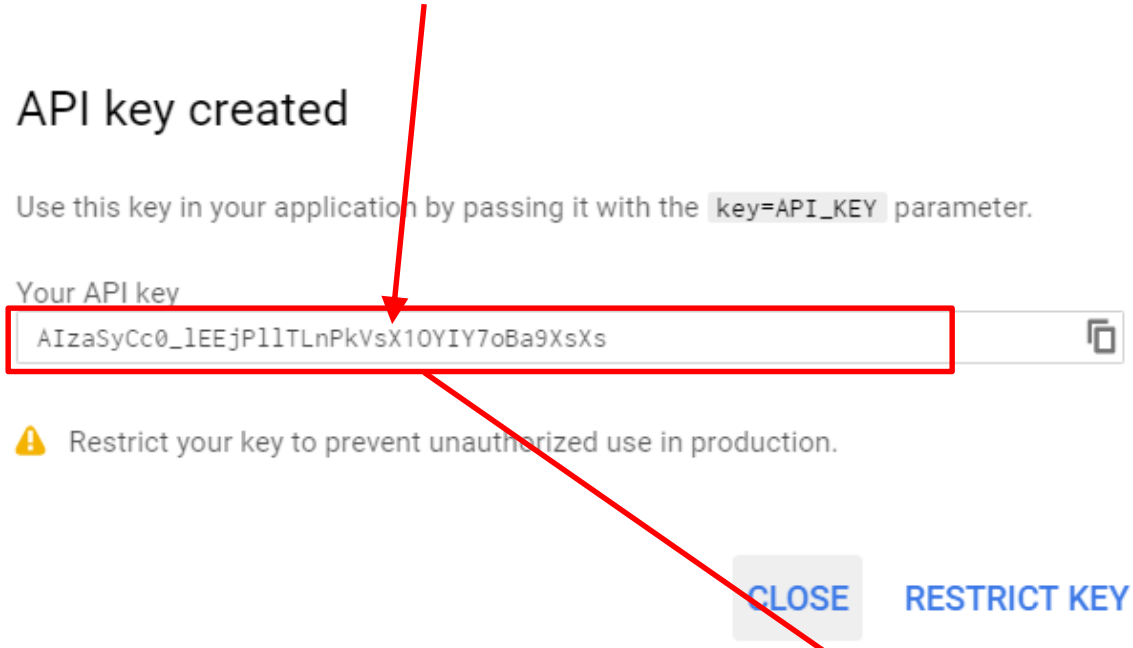


Step 4a: Fast, Easy way to get Maps API Key

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



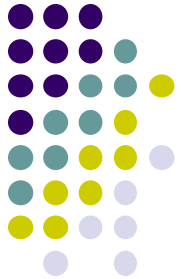
- If successful, Maps API key generated



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

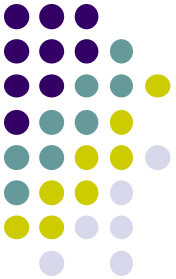
```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyCc0_lEEjPllTLnPkVsX1OYIY7oBa9XsXs</string>
```

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**

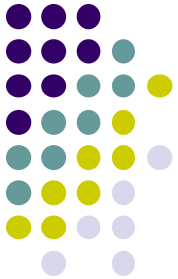
Step 5: Examine Code Generated buy Android Studio Maps Template



- XML file that defines layout is in **res/layout/activity_maps.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 buy Android Studio Maps Template

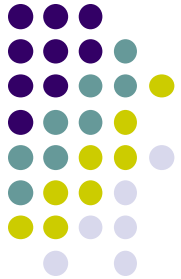


- Default Activity file is **MapActivity.java**

```
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 MapsActivity 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

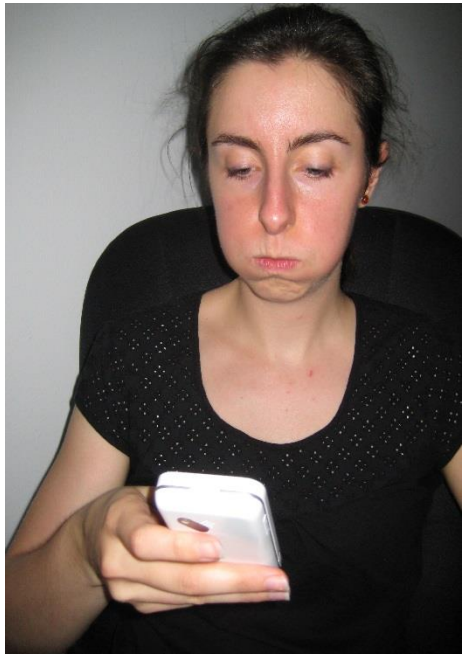
- **Step 6:** Connect to an Android device (smartphone)
- **Step 7:** Run the app
  - Should show map with a marker on Sydney Australia
- More code examples at:
  - https://github.com/googlemaps/android-samples



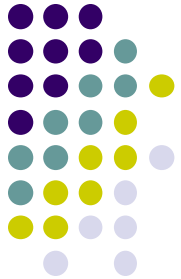
AsyncTask API

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



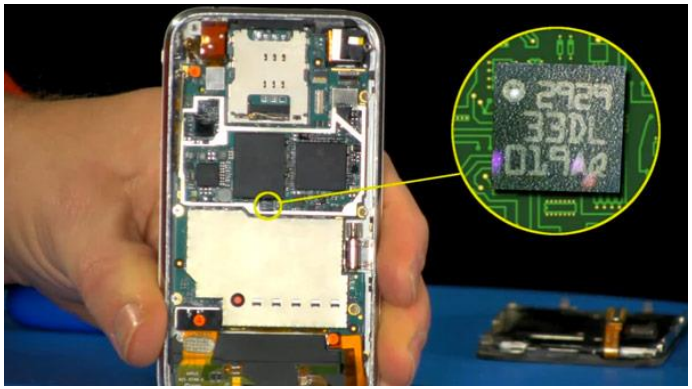
One thread
=
Frustrated user !

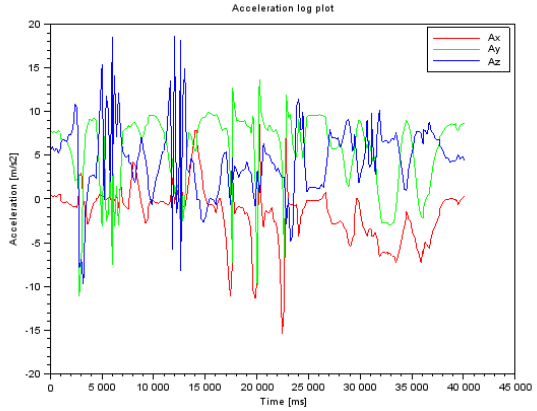


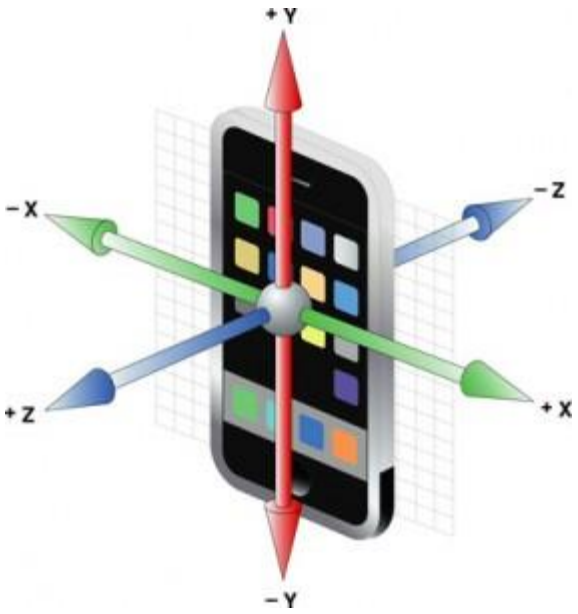
Android Sensors

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

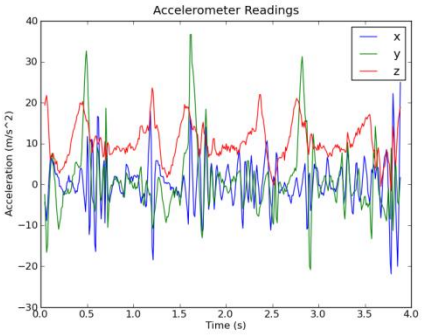




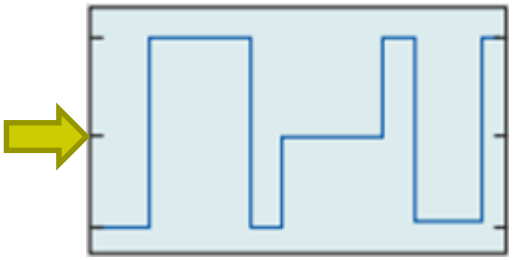


So What?

- 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

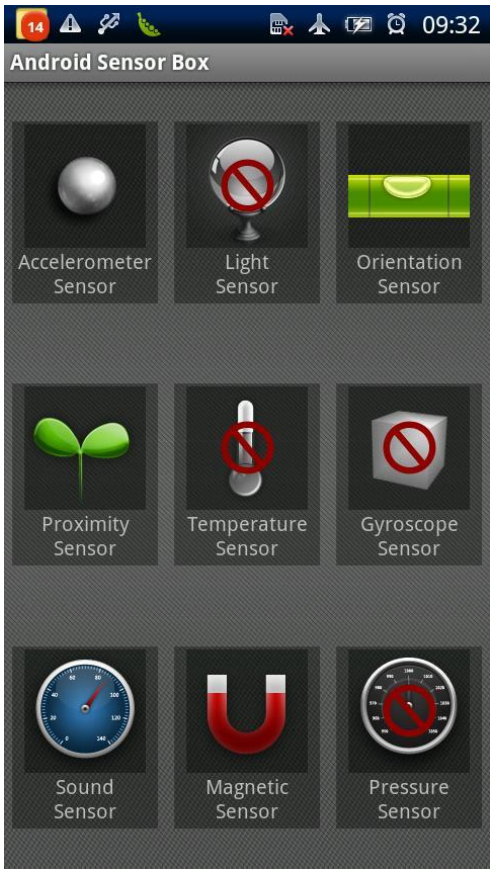
Walking
Running
Jumping
Step count
Calories burned
Falling

Android Sensors

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

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

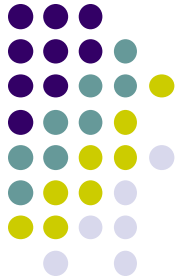
AndroSensor



Android Sensor Box

Android Sensor Framework

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

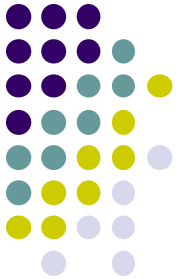


- Enables apps to:
  - Access sensors available on device and
  - Acquire raw sensor data

- 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

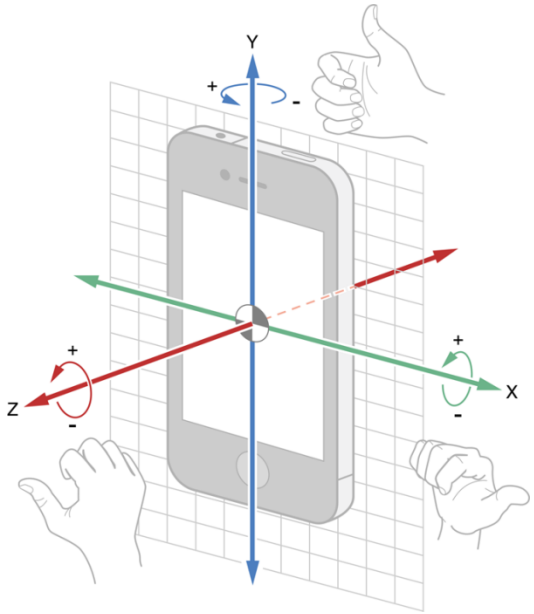
- **Hardware sensor:**
  - physical components built into phone,
  - **Example:** temperature

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

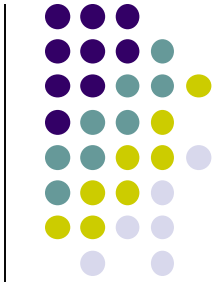
Sensor Types Supported by Android

- 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



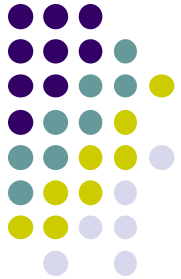
Sensor	HW/SW	Description	Use
TYPE_ACCELEROMETER	HW	Rate of change of velocity	Shake, Tilt
TYPE_AMBIENT_TEMPERATURE	HW	Room temperature	Monitor Room temp
TYPE_GRAVITY	SW/HW	Gravity along X,Y,Z axes	Shake, Tilt
TYPE_GYROSCOPE	HW	Rate of rotation	Spin, Turn
TYPE_LIGHT	HW	Illumination level	Control Brightness
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

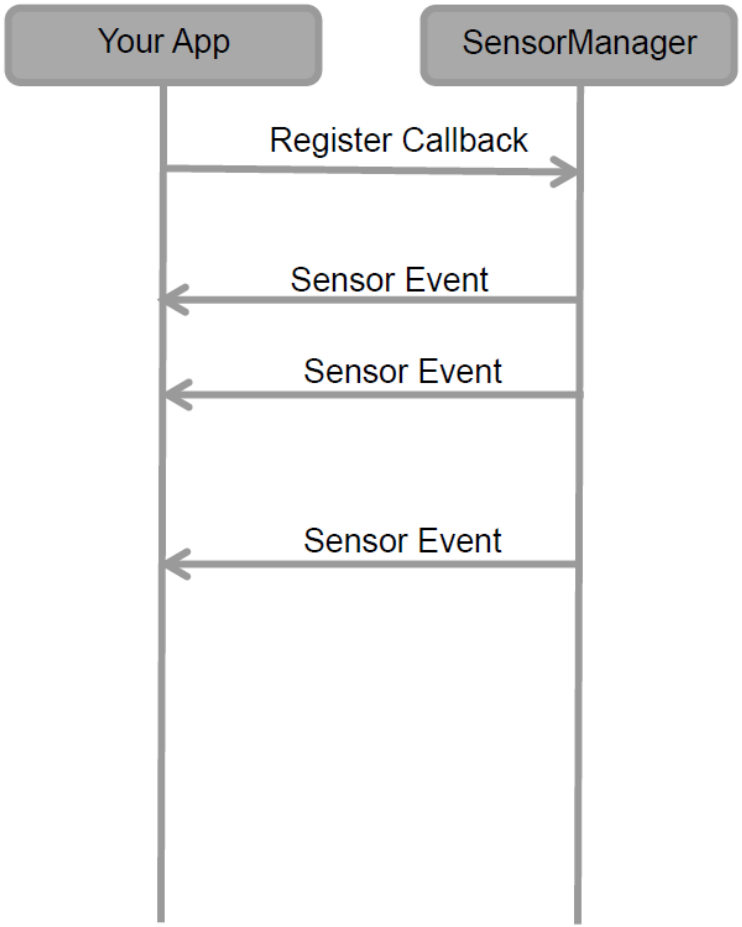


- Sensor framework is part of **android.hardware**
- Classes and interfaces include:
  - **SensorManager**
  - **Sensor**
  - **SensorEvent**
  - **SensorEventListener**
- These sensor-APIs used for:
  1. Identifying sensors and sensor capabilities
  2. Monitoring sensor events

Sensor Events and Callbacks

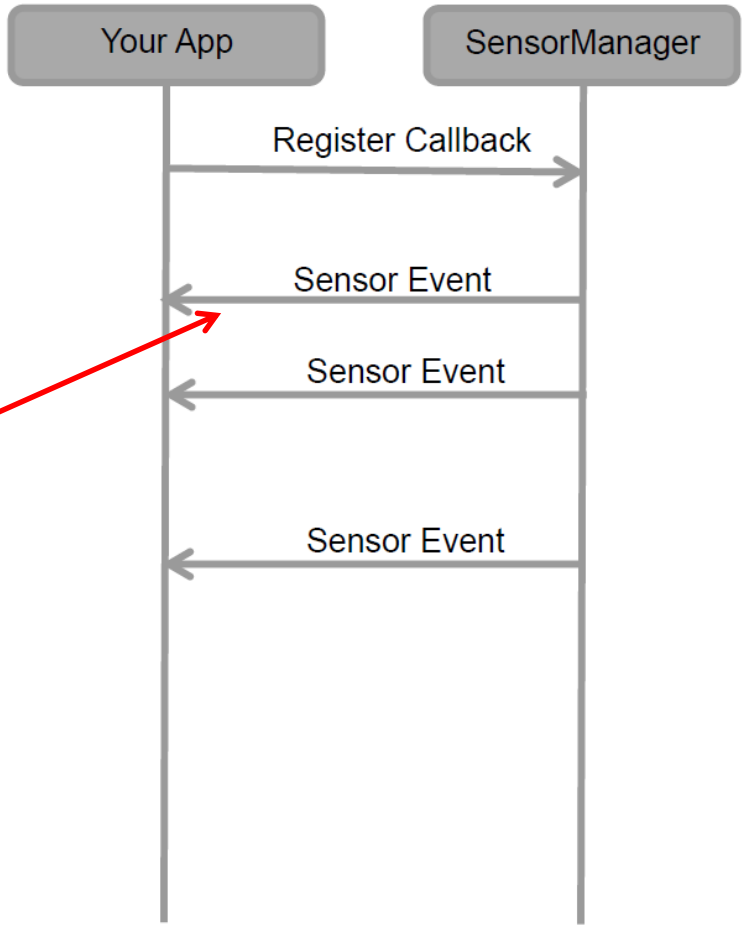
- Sensors send events to sensor manager asynchronously, when new data arrives

- 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

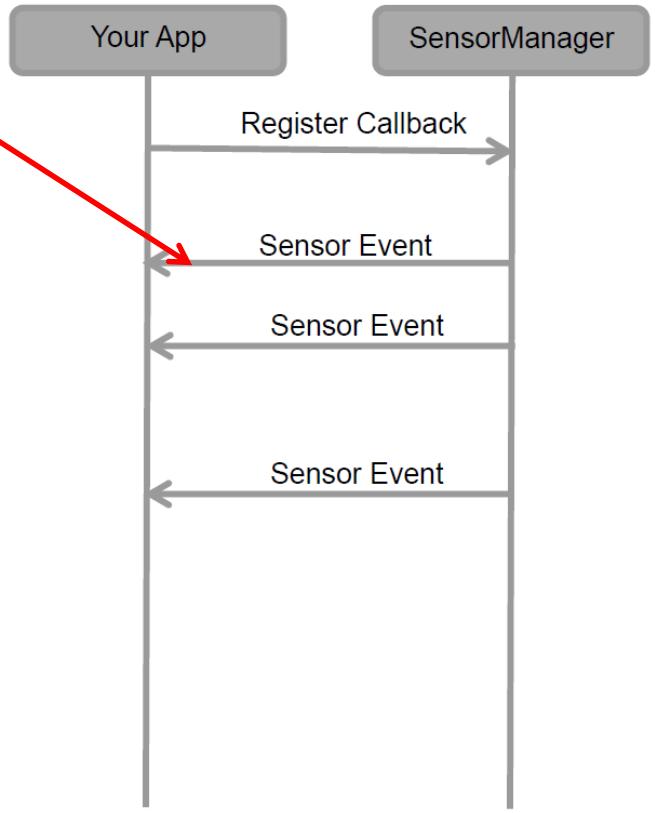


SensorEvent

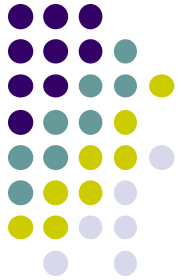
- Android system sensor event information as a **sensor event object**

- **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.	

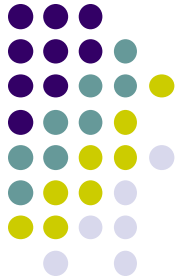


Sensor Values Depend on Sensor Type

Sensor Values Depend on Sensor Type

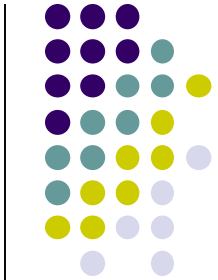
Sensor	Sensor event data	Description	Units of measure
TYPE_LINEAR_ACCELERATION	SensorEvent.values[0]	Acceleration force along the x axis (excluding gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (excluding gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (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( ))**

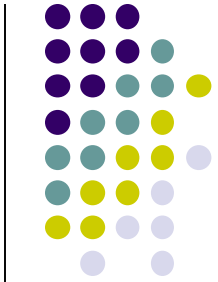
Sensor API Tasks



- **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

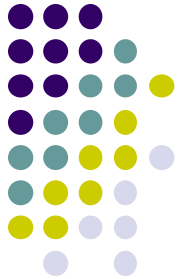
```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

- 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

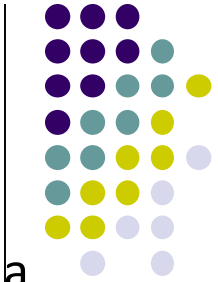


Checking if Phone has at least one of particular Sensor Type

- 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

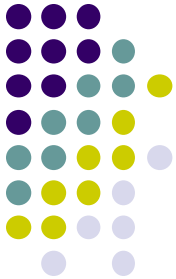
```
public class SensorActivity extends Activity implements SensorEventListener {
  private SensorManager mSensorManager;
  private Sensor mLight;

  @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

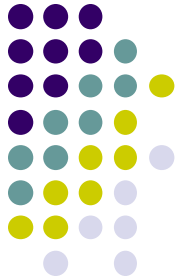
```
@Override
public final void onSensorChanged(SensorEvent event) {
  // The light sensor returns a single value.
  // Many sensors return 3 values, one for each axis.
  float lux = event.values[0];
  // Do something with this sensor value.
}

@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

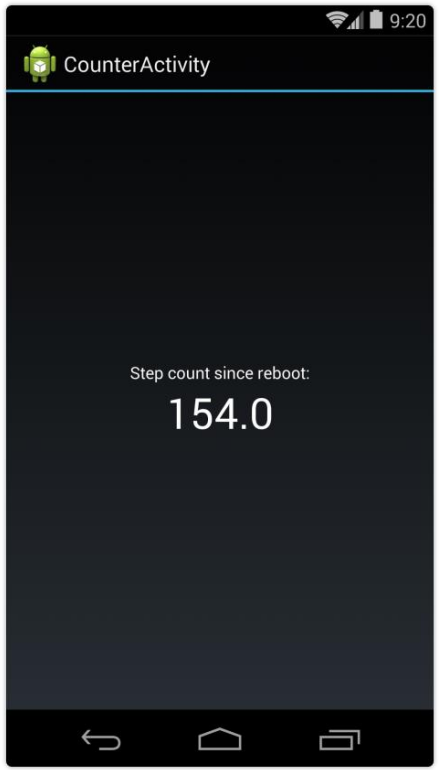
- Following code checks if device has at least one pressure sensor

```
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

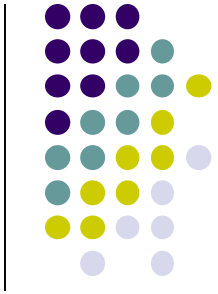
- **Goal:** Track user's steps, display it in TextView
- **Note:** Phone hardware must support step counting

```
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/

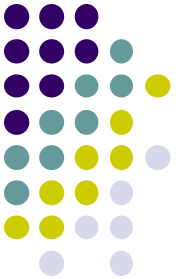
Example Step Counter App (Contd)



```
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/

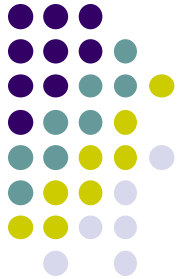
Example Step Counter App (Contd)



```
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/

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