Programming in Android

Session: 13

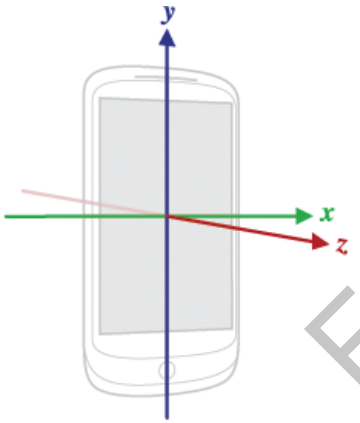
Sensors

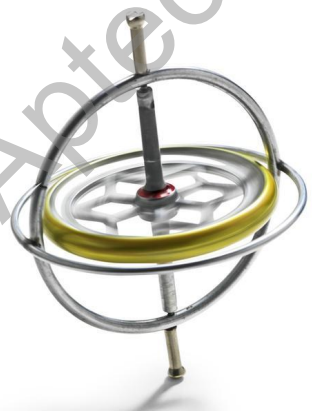
Objectives

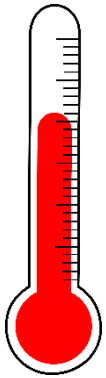
- Explain Various types of Sensors
- Use Motion Sensors
- Use Position Sensors
- Use Status Sensors
- Explain Context aware services based on sensor information

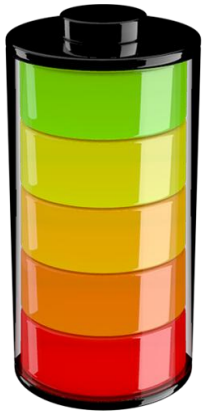
Introduction

- Sensors are hardware devices that record and measure physical data
- Sensors in Android play a key role in the core operating system as well as the platform as a whole
- Sensors in Android can be broadly classified into following three types:
  - Motion Sensors
  - Environmental Sensors
  - Position Sensors



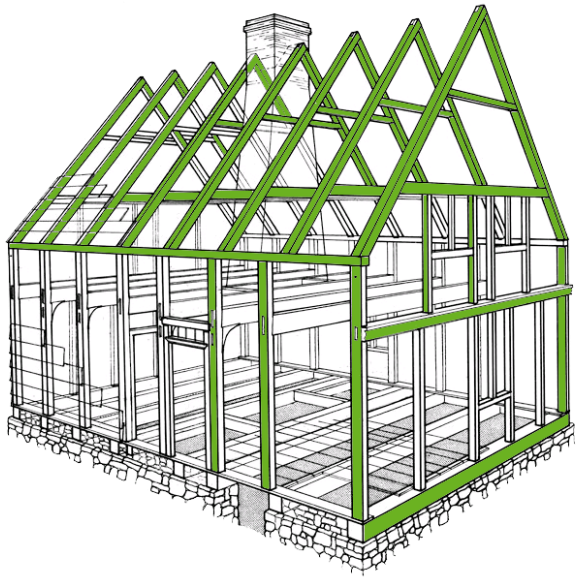






Sensor Framework

- Sensor Framework is responsible for translating the input received from the hardware devices to readings usable by the applications
- Raw data received from the sensors can also be accessed using the Sensor Framework
- Certain sensors can be implemented by software emulation using data received from other sensors



Sensor API 1-2

- **SensorManager**
  - This class is the central component of the sensor framework
  - It is responsible for retrieving sensor instances and setting event listeners for the sensor
  - It is also responsible for identifying the sensor support on the device

- **Sensor**
  - This class is used to reference a specific sensor on the device
  - An instance of a sensor class can be retrieved by using the getDefaultSensor(int type) or the getSensorList(int type) method of SensorManager

Sensor API 2-2

- **SensorEvent**
  - This class stores the event information whenever there is a change in status of the sensor

- **SensorEventListener**
  - This class provides an interface to implement event listeners for the sensors
  - The methods onSensorChanged() and onAccuracyChanged() need to be implemented

Working with Sensor Data

- Obtain a reference to the SensorManager instance
- Ensure that the sensor type is supported by the device
- Get a reference to the specific type of the sensor
- Create an event listener for the sensor(s) by implementing the SensorEventListener interface
- Implement the onSensorChanged() method to handle changes in sensor data
- Implement the onAccuracyChanged() method to handle accuracy changes
- Register the listener with the sensor manager for the type of the sensor
- Implement the onPause() method of the activity and unregister the listener
- Implement the onResume() method of the activity to re-register the listener. This saves CPU cycles that are wasted unnecessarily in processing the sensor changes

Working with Sensor Data Example 1-2

- Following Code Snippet demonstrates reading Sensor Data:

```
public class MainActivity extends Activity {

    SensorManager sm;

    private SensorEventListener listener = new SensorEventListener() {

        @Override
        public void onSensorChanged(SensorEvent event) { …              }
        }

     @Override
      public void onAccuracyChanged(Sensor sensor, int accuracy) {
            // Code here
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
}
```

Working with Sensor Data Example 2-2

- It is strongly recommended to unregister the listener when the activity is paused as shown in the following Code Snippet:

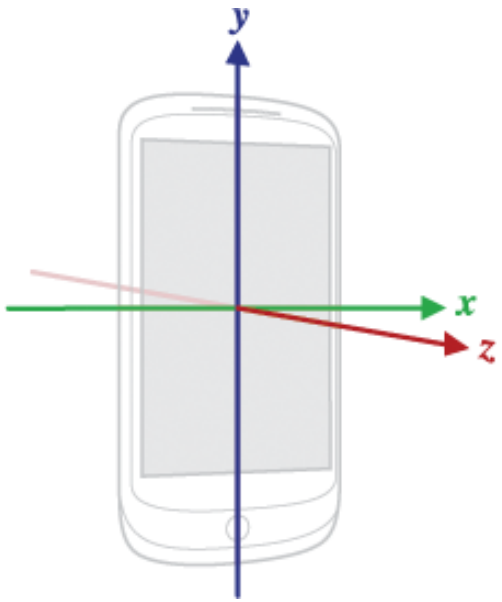
```
…
    @Override
    protected void onResume() {
        super.onResume();

sensorManager.registerListener(listener,sensorManager.getDefaultSensor(Sensor.
TYPE_PROXIMITY),SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(listener);
    }
…
```

Reading Sensor Event Data

- The onSensorChanged(SensorEvent event) and onAccuracyChanged(SensorEvent event) methods are passed an event object reference
- The event float matrix holds the sensor reading values. The values represent the following data:
  - For most sensors, the values of the indices 0, 1, and 2 measure the values along the X, Y, and Z axis respectively
  - Uncalibrated sensors have three additional fields in the indices 3, 4, and 5 which measure the drift values along the X, Y, and Z axis respectively
  - Environment sensors have only one value in the index 0, which measures the reading of the sensor

Motion Sensors

- Motion Sensors help monitor the movement of the device
- Some motion sensors are hardware only and the remaining may be implemented in software
- Motion sensors are primarily used in games and gimmicky multimedia applications



Motion Sensor Types

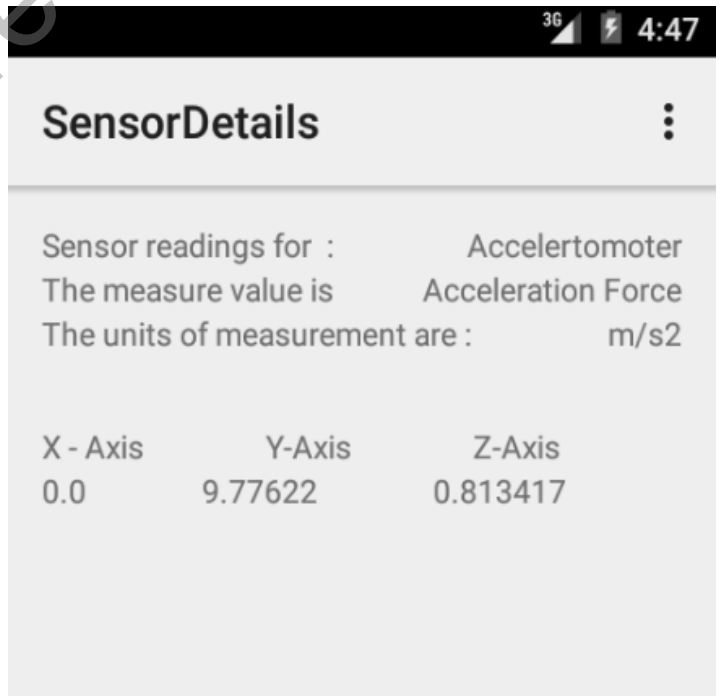
- Following table lists the types of motion sensors supported by Android along with their implementation option and utility:

Sensor Type	Implementation	Description
TYPE_ACCELEROMETER	Hardware	Accelerometer measures the acceleration or movement of the device in a direction
TYPE_GRAVITY	Hardware or Software	Gravity Sensor measure the gravitation force experienced by the device
TYPE_GYROSCOPE	Hardware	Gyroscope measures the rotational force of the device in a direction
TYPE_GYROSCOPE_UNCALIBRATED	Software	The software sensor provides raw uncalibrated data from the Gyroscope
TYPE_LINEAR_ACCELERATION	Hardware or Software	Linear Acceleration scope provides the acceleration force experienced by the device without the Gravitation force
TYPE_ROTATION_VECTOR	Hardware or Software	Provides the angle and inclination of the device
TYPE_STEP_COUNTER	Hardware or Software	Calculates the number of steps measured by the step sensor since the time of activation

Motion Sensor Example Application

- Using the code for Sensor Framework, an application for demonstrating Motion Sensors is created as shown in the following figure:

- Select a sensor and move the device in any direction to view the data as shown in the following figure:



Detecting Shake Motion

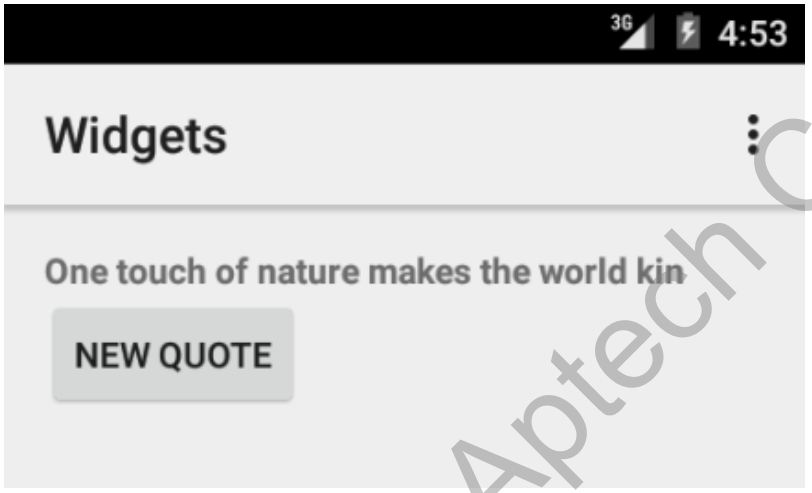
- A shake or jerk motion has been staple to mobile phone even before Android came into existence
- A shake motion is used to signify a ’Next’ gesture
- **Reading Sensor Input**
  - The main sensor that is responsible for detecting a shake motion is the accelerometer
  - The Linear Acceleration sensor is the ideal sensor for this purpose, but not all devices have this sensor
  - Hence, the accelerometer will be used as it is guaranteed to be supported on all devices
  - A shake motion in terms of sensor reading is characterized by a sudden increase in the acceleration experienced by the device in a short period of time

Steps to Detect Shake Motion

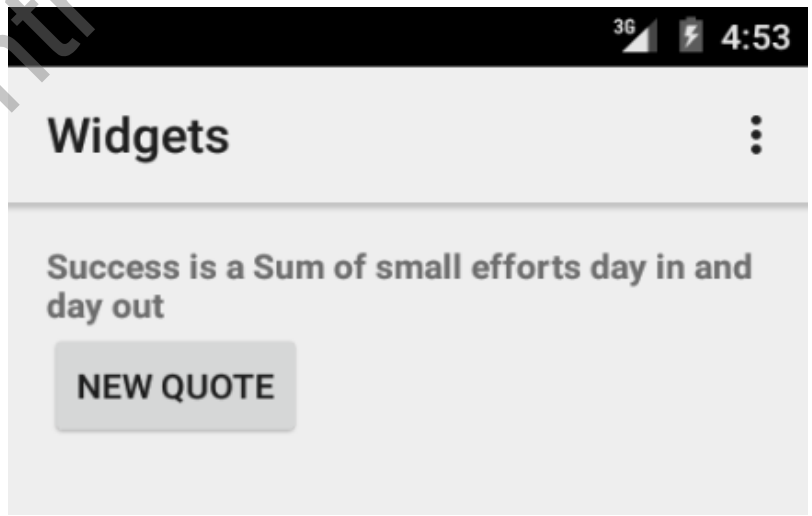
- Create a SensorEventListener object for receiving changes in the accelerometer readings
- Save the initial time of measuring the acceleration
- Calculate the Root Mean Square (RMS) values of the initial acceleration of the device, that is, Square root of $X^2+Y^2+Z^2$
- Store the initial acceleration value
- Whenever new acceleration values are experienced, check the time difference from the time of the initial measurement and time of changed values
- If the time difference is greater than a threshold (in this case, 300 milliseconds), then calculate the new RMS acceleration value
- Calculate the change in acceleration using the RMS value
- If the change in acceleration is greater than a threshold (in this case, 6), execute the code to handle a shake event. This implies a rapid change in acceleration in a short period time

Shake Motion Example Application

- Using the logic, an application for demonstrating Shake Motion is created as shown in the following figure:



- By shaking the device, a new quote is displayed as shown in the following figure:



Shake Motion Example Application Logic

- The application logic is as follows:

- Developer creates an Activity which displays a random Quote to the user
- The Quote is generated using the QuoteGenerator class which contains a list of Quotes of which one is returned randomly
- The app detects a shake motion using the change in acceleration using the accelerometer
- If a shake motion is detected, a new Quote is retrieved and displayed to the user
- A new quote is also displayed by clicking the New Quote button

Position Sensors

- Position Sensors help monitor the position and orientation of the device with respect to its surroundings
- The Geomagnetic Sensor and the Proximity Sensor are implemented in almost all Android Devices

Position Sensor Types

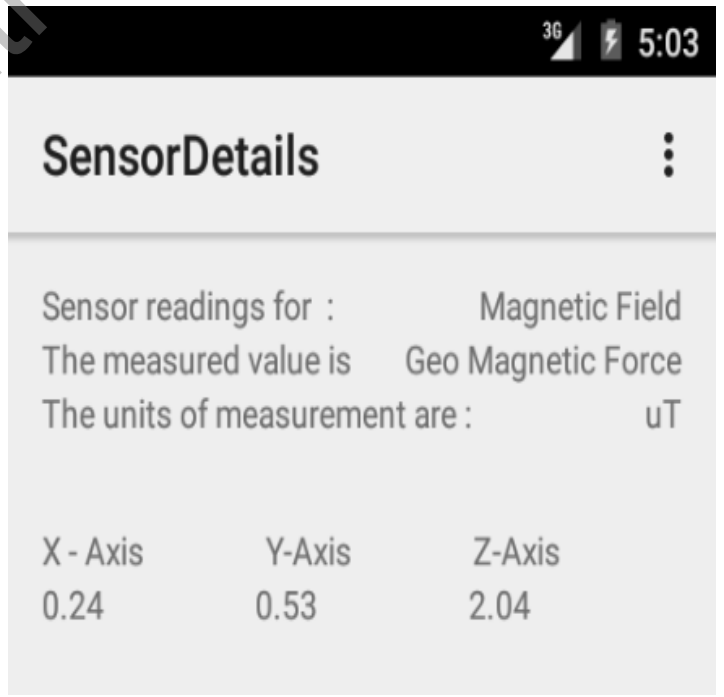
- Following table lists the types of position sensors supported by Android along with their implementation option and utility:

Sensor Type	Implementation	Description
TYPE_GAME_ROTATION_VECTOR	Software	A specialized rotation vector that is useful for developing games. It is the rotational vector value excluding the geomagnetic readings. This sensor is used to implement 'Steering Wheel' or 'Tilt' functionality in games
TYPE_GEOMAGNETIC_ROTATION_VECTOR	Hardware or Software	It provides a rotation angle and inclination information of the device similar to the rotation vector. The key difference being that it uses the Geomagnetic Sensor instead of the Gyroscope
TYPE_MAGNETIC_FIELD	Hardware or Software	Geomagnetic Sensor helps monitor changes in the Earth's magnetic field. The readings are reported in terms of field strength along the x, y, and z axis
TYPE_MAGNETIC_FIELD_UNCALIBRATED	Software	Provides Raw uncalibrated data from the Geomagnetic sensor
TYPE_PROXIMITY	Hardware or Software	The proximity sensor measures the distance of the device from a nearby object or the device user. This information can be in the form of exact distance measured in centimetres (cm) or just a binary value indicating far or near

Position Sensor Example Application

- Using the code explained for Sensor Framework, an application for demonstrating Position Sensors is created as shown in the following figure:

- Select a sensor and move the device in any direction to view the data as shown in the following figure:



Turning OFF the Device Screen 1-2

- There is no standard procedure in Android for Turning OFF the Screen
- Some of the workaround to achieve this behavior is explained as follows:
- **Device Policy Administrator**
  - This process involves adding the application as a Device Administrator and then, using the device policy manager to lock the screen of the device essentially switching OFF the screen
  - This is similar to emulating a Power button press
- **Power Manager Sleep Mode**
  - The Power Manger's goToSleep() method can be used to put the device in sleep mode

Turning OFF the Device Screen 2-2

- **Using a Wake Lock**
  - Another implementation using the power manager involves getting hold of a wake lock momentarily and then releasing it, to put the device to sleep
- **Setting the Brightness to Extremely Low**
  - Most Android devices turn off the screen entirely if the brightness value of the screen is set to an impossibly low value
- **Reducing the Screen Timeout Value**
  - The timeout value for the display to power OFF when there is no interaction can be set to a value low enough that the display powers OFF immediately

Turning OFF the Screen Example 1-2

- Following Code Snippet demonstrates various methods for turning OFF the device screen:

```
 DevicePolicyManager dpm =
(DevicePolicyManager)getSystemService(Context.DEVICE_POLICY_SERVICE);
        dpm.lockNow();

        //Alternate Method 1
        /*
        PowerManager manager = (PowerManager)
getSystemService(Context.POWER_SERVICE);
        manager.goToSleep(1000);
         */

        //Alternate Method 2
        /*
        PowerManager.WakeLock wl =
manager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "Your Tag");
        wl.acquire();
        wl.release();
         */
```

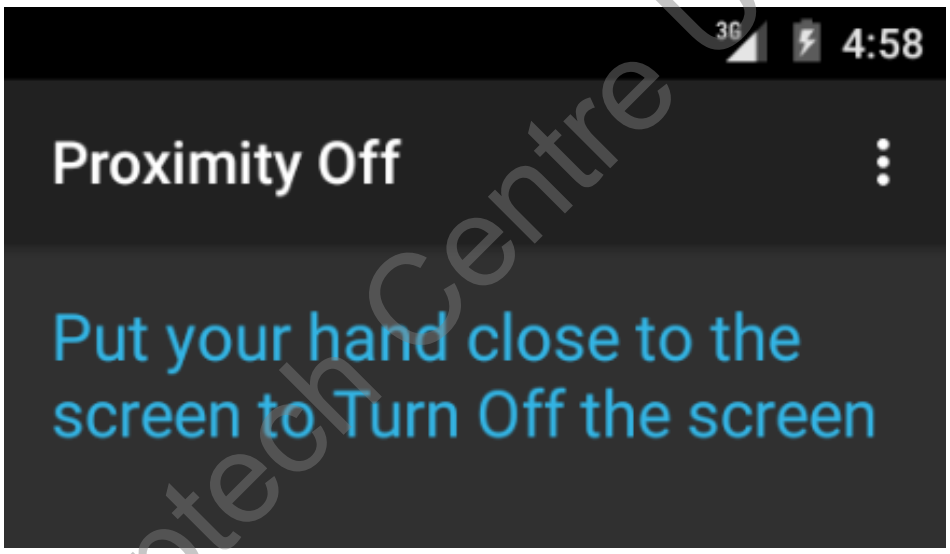
Turning OFF the Screen Example 2-2

- Following Code Snippet demonstrates various methods for turning OFF the device screen:

```
        //Alternate Method 3
        /*
        WindowManager.LayoutParams params = getWindow().getAttributes();
        params.flags |= LayoutParams.FLAG_KEEP_SCREEN_ON;
        params.screenBrightness = 0;
        getWindow().setAttributes(params);
         */

        //Alternate Method
        /*
        Settings.System.putInt(getContentResolver(),
Settings.System.SCREEN_OFF_TIMEOUT, 10);
         */
    }
```

Proximity OFF Example

- Using the explained code, an application for automatically turning OFF the screen on proximity is created as shown in the following figure:



- The screen will be turned OFF if a hand is placed near the screen or if the device is held close to the face in an in-call position

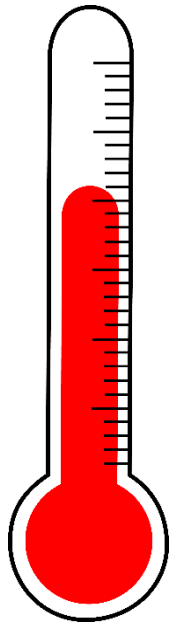
Proximity Example Application Logic

- The application logic is as follows:

- Developer creates an Activity which displays the usage information to the user
- A listener is registered for the Proximity Sensor
- Whenever there is change in the proximity values, the values are checked to be lesser than a threshold
- If the values are found to be lesser then the screen is turned off
- This is the same functionality provided by the Android's "Phone" application and other VoIP application such as skype

Environment Sensors

- Environment sensors help monitor the properties of the surroundings of the device
- All environment sensors are completely optional to implement by the device vendor
- All the Environment sensors are hardware based only



Environment Sensor Types

- Following table lists the types of Environment sensors supported by Android along with their implementation option and utility:

Sensor Type	Implementation	Description
TYPE_AMBIENT_TEMPERATURE	Hardware	Temperature sensor measures the room temperature that the device is placed in
TYPE_LIGHT	Hardware	Light sensor measures the ambient light falling on the device. This information is used in devices to adjust the brightness of the screen automatically
TYPE_PRESSURE	Hardware	Pressure sensor measures the air pressure around the device
TYPE_RELATIVE_HUMIDITY	Hardware	Relative Humidity sensor measures the relative humidity in the atmosphere

Calculating Additional Information

- **Absolute Humidity**
  - Absolute humidity is the total mass of water vapour present in a given volume of air
  - Absolute humidity is measured in grams/meter^3
  - It can be calculated using the following equation:

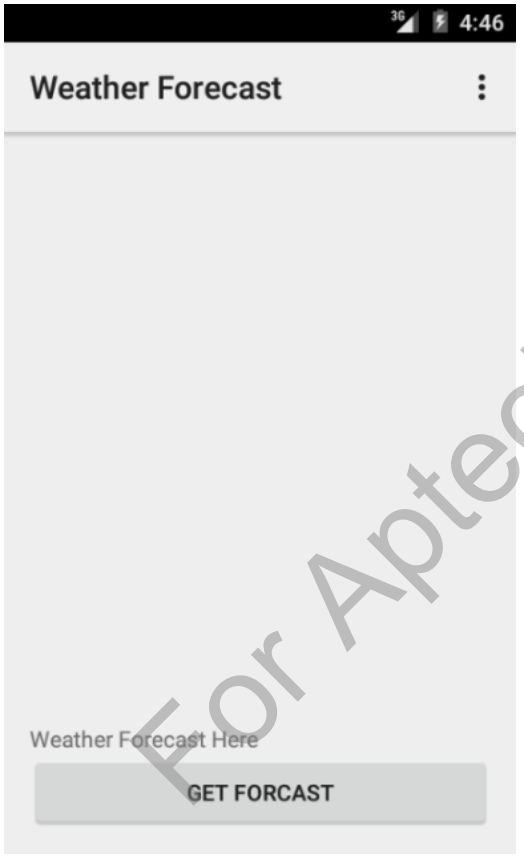
$$d_v(t,RH) = \frac{216.7 \cdot (RH/100\%) \cdot A \cdot \exp(m \cdot t/(T_n+t)}{273.15 + t}$$

- **Dew Point**
  - The dew point is the temperature at which the water vapour in a sample of air at constant barometric pressure condenses into liquid water at the same rate at which it evaporates
  - It can be calculated using the following equation:

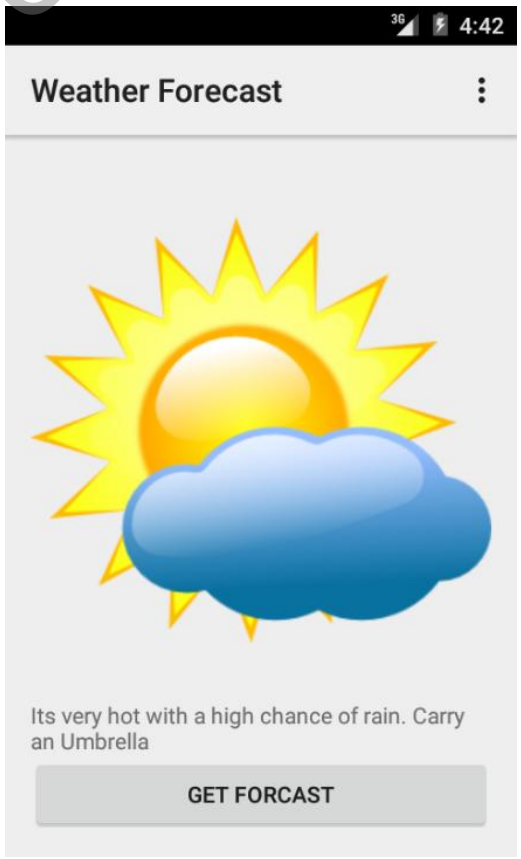
$$t_d(t,RH) = \frac{T_n \cdot \ln(RH/100\%) + m \cdot t/(T_n+t)}{m - [\ln(RH/100\%) + m \cdot t/(T_n+t)]}$$

Environment Sensor Example Application

- Using the code for Sensor Framework and the logic, an application for Weather Forecasting is created as shown in the following figure:



- By clicking GET FORECAST, a weather forecast is generated as shown in the following figure:



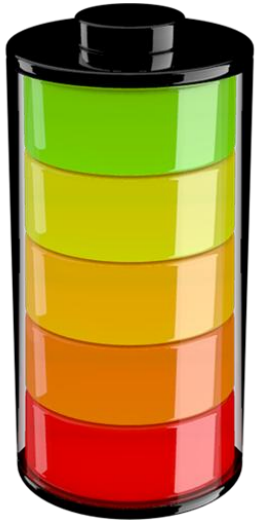
Weather Forecast Application Logic

- The application logic is as follows:

- Developer creates an Activity which displays an image summing up the weather condition and a tag line
- One the Get Forecast button is clicked the generateForecast() method is called
- The method retrieves Environment information such as temperature, humidity and also calculates further information based on the values
- Based on the inputs, a weather prediction is made and the appropriate Icon and tag lines are displayed to the user

Other Status Related Sensors

- Besides the three major types of sensors, there are additional sensors in Android devices which can be accessed by other means or optional and specific to certain device types.
- Some of them are:
  - Battery Sensor
  - Heart Rate Sensor



Battery Sensor

- Battery status can be retrieved using a broadcast receiver with an intent filter for the intent Intent.ACTION_BATTERY_CHANGED
- The receiver will receive an update whenever there is a change in the status of the battery (that is, charging, discharging, and so on.)
- The receiver will also receive and update when there is a change in the battery level

Heart Rate Sensor

- The Heart Rate sensor is a heart rate monitoring sensor that is unique to the Android Wear devices
- The application using this sensor needs to be targeted specifically for a wear device
- The requirements for this are small screens with circular layouts and limited resolution
- The information reported by the sensor is a single event value at the index 0
- The data represents the heart rate in beats per minute
- Heart Rate Monitor Sensor API is newly introduced and subject to change

Context Sensing and Location Based Services 1-2

- Sensors contain a lot of information which can be further processed to generate even more information
- Location also plays a key role in identifying the context
- Marketing is a prime area of use for location based services, followed by news and infotainment
- In order to implement context sensing, the developer can either write a specialized sensing framework from scratch that fits their needs

Context Sensing and Location Based Services 2-2

- An alternative would be to use existing context sensing services such as Intel's Context Sensing SDK
- The SDK framework automatically detects the available sensors and generates context data over time
- Google's own applications such as Google Now and products such as Google Ads implement context sensing to generate suggestions
- Before implementing Context Sensing, the developer needs to take the privacy concerns and the performance hit into account

Summary

- Sensors are devices that record physical data
- Android supports primarily three types of sensors
- Motion Sensors monitor the movement of the device
- Position Sensors monitor the position and orientation of the device
- Environment Sensors monitor the surroundings of the device
- Other miscellaneous or vendor specific sensors are also supported by Android
- Sensors can be used for context sensing in applications