Sensing the World Around Us with Android Smartphones

Veljko Pejovic (veljko.pejovic@fri.uni-lj.si)
University of Ljubljana, Slovenia

Lancaster University, June 2025

Cyber-Physical World

Computing devices are integrated with our physical world

- Smartphones (more than 6 bn of them in the world!) are very personal devices, carried by their users at all times
- Wearable computing devices are in a constant physical contact with their users
- Approximately 30 billion Internet-of-Things (IoT) devices embedded in our everyday environment

Context-Aware Computing

Computers can infer a lot about the context in which they are used

- Devices are equipped with an array of sensors
- Ubiquitous connectivity enables transport of large amounts of data from user devices to the clouds
- Computing capabilities of edge devices enable complex computation, such as machine learning inference

Smartphone as a Sensor

Affordances

Accelerometer

Magnetometer

GPS

Light

Camera

Barometer

Gyroscope

Proximity

Microphone

WiFi

GSM

NFC

Bluetooth (BT)

Touch screen

Thermometer

Humidity sensor

Wireless gesture radar

Smartphone as a Sensor

Challenges

- Phone sensing requires a significant engineering effort:
  - Frequent sampling with what was supposed to be an occasionally used feature
  - Accuracy problems
  - Battery lifetime
  - Processing overhead
- Android is trying to lower the sensing overhead:
  - E.g. Google Play Services for location updates

Sensing in Android

Sensors in Android

Sensors (Android OS's def.) ≠ Sensors (our def.)

- Sensor (android.hardware.*):
  - Motion sensors (accelerometers, gyroscopes, etc.)
  - Environmental sensors (barometers, thermometers, etc.)
  - Position sensors (orientation sensors and magnetometers)
- Wireless
  - Bluetooth android.bluetooth.*
  - Wi-Fi android.net.wifi.*
  - NFC android.nfc.*

Sensors in Android

Sensors (Android OS's def.) ≠ Sensors (our def.)

- Google Play Services:
  - Location – no direct access to GPS
  - Physical Activity – an already embedded classifier, no need to query acceleration, location, etc.

Sensors Framework

Accessing certain hardware sensors

- SensorManager class
  - A system service for sensing management
    - e.g. infer (at runtime) which sensors are available
- Sensor class
  - Sensor types as constants e.g. Sensor.TYPE_MAGNETIC_FIELD
  - Data reporting: 1) streaming or 2) on change
  - getResolution(), getMaximumRange(), getPower()

```
private val sensorManager =
        getSystemService(Context.SENSOR_SERVICE)
        as SensorManager

val deviceSensors: List<Sensor> =
        sensorManager
        .getSensorList(Sensor.TYPE_ALL)
```

Sensors Framework

Publish-subscribe sensing

- SensorEvent class
  - Events containing new sensed values, accuracy timestamp, and sensor type information
- SensorEventListener interface
  - Implement and override:
    - onSensorChanged(event : SensorEvent)
    - onAccuracyChanged(sensor: Sensor, accuracy: Int)

Sensors Framework

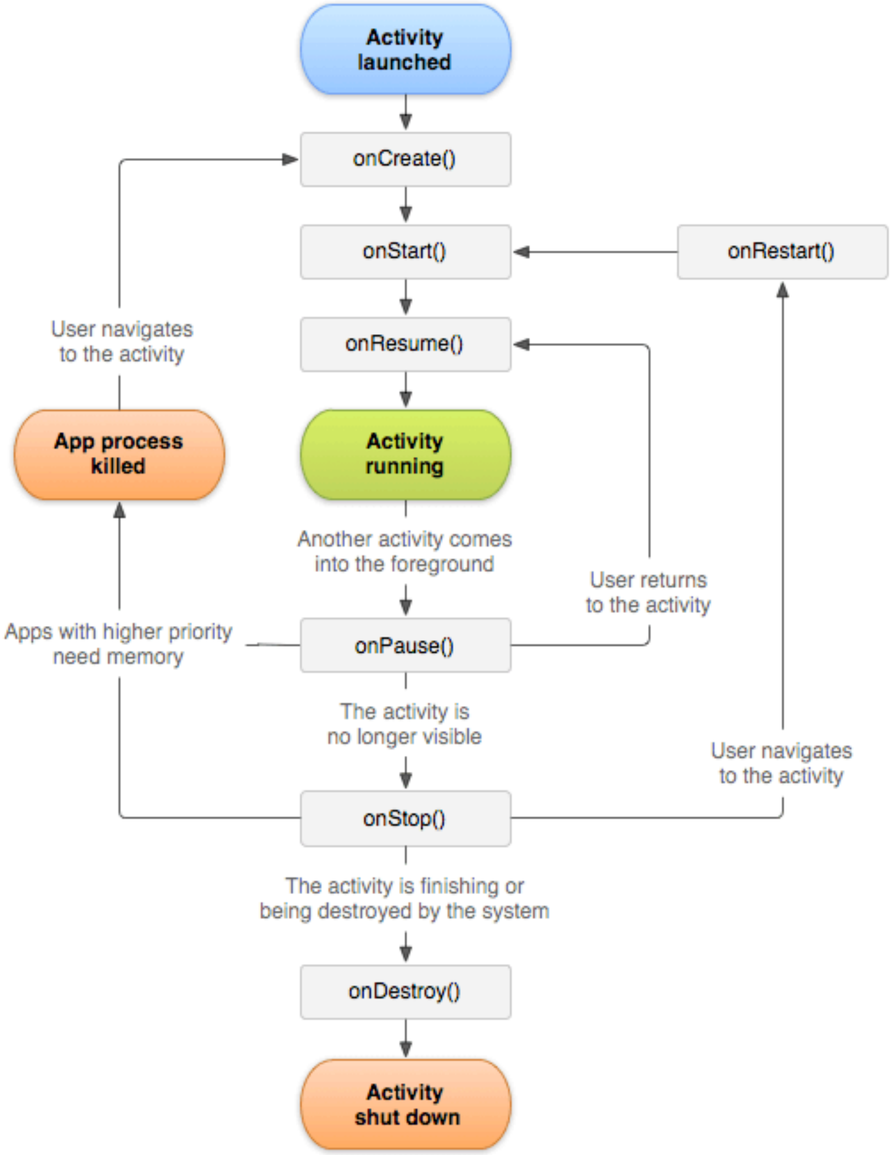
Publish-subscribe sensing

- SensorEventListener interface
  - Register the listener

```
sensorManager.registerListener(this, mSensor,
SensorManager.SENSOR_DELAY_NORMAL)
```

  - Unregister when done, leaving the Activity/Service

```
sensorManager.unregisterListener(this)
```



Sensors Framework

Summary

- android.hardware.* is how you sense accelerometer, magnetometer, barometer, temperature sensor, light, proximity, etc.
- Before using it, verify that a sensor is present
- Unregister the listener when leaving
- Don’t do heavy work in onSensorChanged
- Choose sensor reporting delay wisely
    - Use the highest delay that still works for your app’s purpose

Google Play Services

Purpose

- A background service providing access to a range of Google's services:
  - Maps
  - Google sign in
  - Google Drive
  - Location
  - Activity recognition
- Centralised handling of sensing requests reduces energy usage
  - One GPS result may be served to all apps requesting location within a short period

Google Play Services
Location sensing

- FusedLocationProviderClient class
  - Access to location determined via different means (GPS, network signal triangulation, etc.)
  - getLastLocation()
  - requestLocationUpdates()
  - Define request priority: from “high accuracy” to “no power”, and request interval
  - Get callback when a new location info is ready
  - Don’t forget to unregister the request!

Google Play Services
Location sensing

- GeofencingClient class
  - Define geofence region and transition types
  - Supply Intent to be fired when the conditions are met

Google Play Services

Activity sensing

- A built-in classifier of physical activity (walking, cycling, still, in vehicle, running)
- ActivityRecognitionClient class
  - Subscribe to activity recognition transitions or updates
  - Supply Intent to be called when the results are ready
  - Don’t forget to unregister the request!

Sensing in Android

Permissions and ethics

- Applications are sandboxed and only the basic functionalities are available by default
- Apps requests permissions to access:
  - User data (e.g. contacts)
  - Some cost-sensitive APIs (e.g. send SMS)
  - Sensors (e.g. camera, microphone, etc.)
- Permissions are declared in AndroidManifest.xml
  - Predefined Strings from Manifest.permission
  - <user-permission> in Manifest

```
<uses-permission android:name=
“android.permission.ACCESS_FINE_LOCATION”/>
```

Sensing in Android

Permissions and ethics

- Normal permissions - the OS automatically grants these at install time
  - FLASHLIGHT, VIBRATE, BLUETOOTH, etc.
- Dangerous permissions - explicitly ask the user to grant access
  - READ_CONTACTS, SEND_SMS, ACCESS_COARSE_LOCATION, etc.
- Special permissions - given through system settings
  - Very sensitive, e.g. MANAGE_EXTERNAL_STORAGE

Sensing in Android

Permissions and ethics - best practices

- Do not request permissions unless you really need them
  - E.g. do you need to write data to an external storage (file) or can you keep the information in DataStore?
- Show immediate benefit of granting a permission
  - E.g. your ToDo list app requires location info – show the user how she can make location-based reminders
- Use Intents to call other apps in case you don’t need to handle the functionality within your app:
  - E.g. call a camera app, rather than requesting the camera permission for your app

Practical lab: Geofencing