Universidade do Minho
Departamento de Informática

SENSORIZAÇÃO E AMBIENTE

MESTRADO EM ENGENHARIA INFORMÁTICA, 1º ANO - Perfil SI

Universidade do Minho
Departamento de Informática

Soft/Physical Sensors

Agenda

- Soft Sensors
    - Mobile
- Hands On

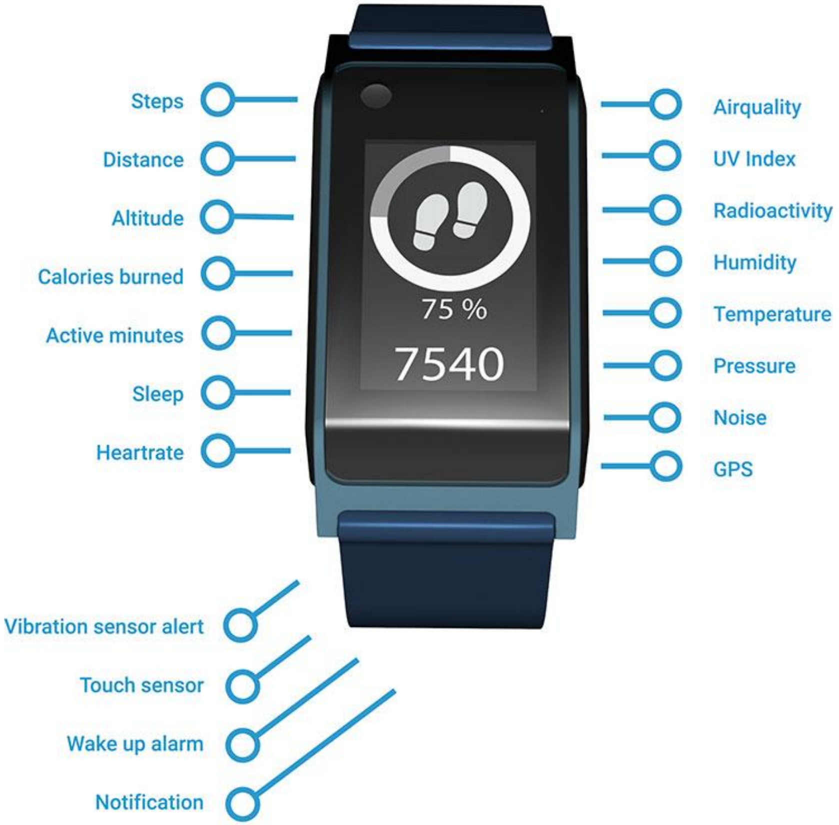
Universidade do Minho
Departamento de Informática

Soft Sensors

Mobile

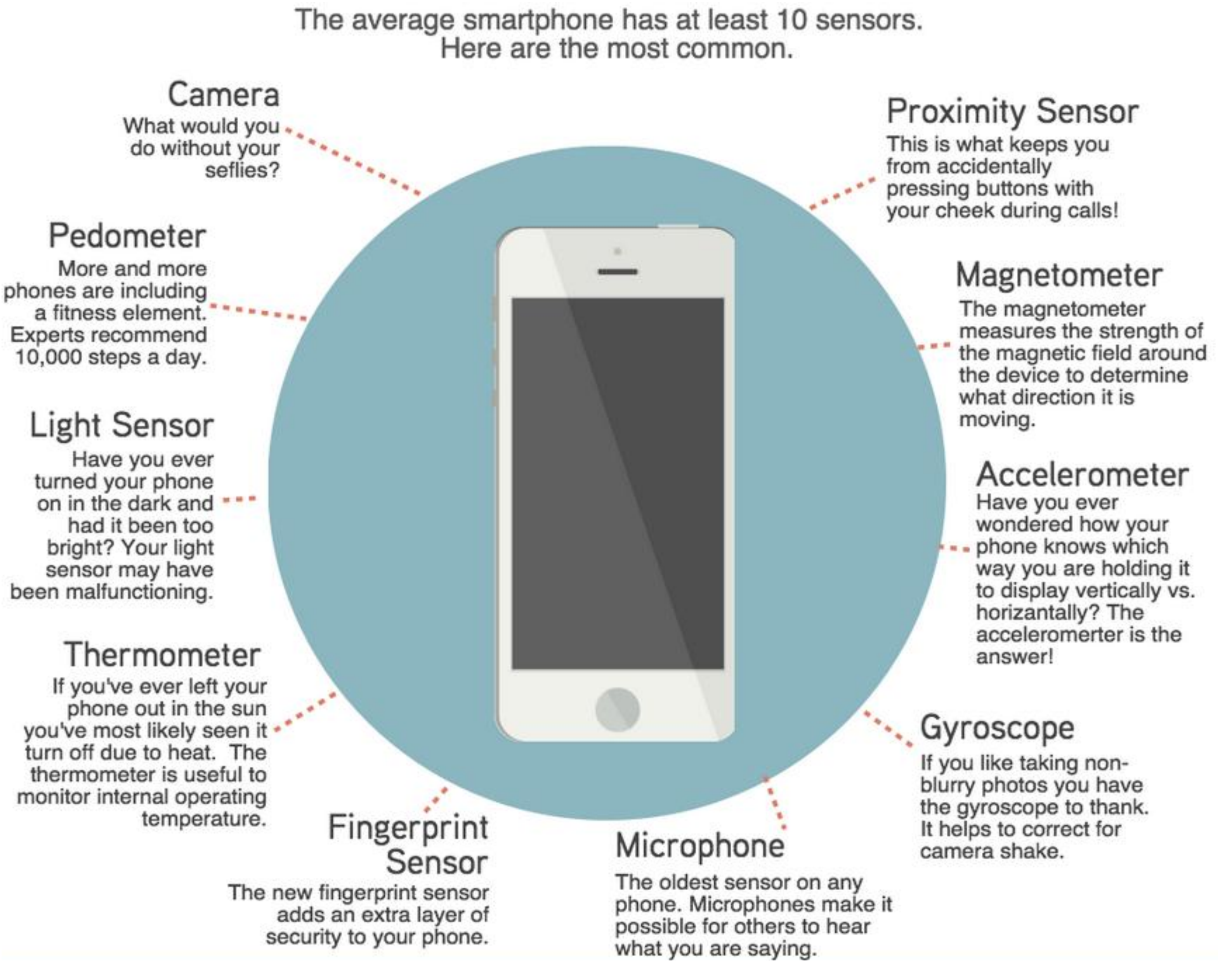
Mobile Sensors

Mobile Sensors



Mobile Sensors

Mobile Sensors



The average smartphone has at least 10 sensors.
Here are the most common.

Camera
What would you do without your seflies?

Pedometer
More and more phones are including a fitness element. Experts recommend 10,000 steps a day.

Light Sensor
Have you ever turned your phone on in the dark and had it been too bright? Your light sensor may have been malfunctioning.

Thermometer
If you've ever left your phone out in the sun you've most likely seen it turn off due to heat. The thermometer is useful to monitor internal operating temperature.

Fingerprint Sensor
The new fingerprint sensor adds an extra layer of security to your phone.

Microphone
The oldest sensor on any phone. Microphones make it possible for others to hear what you are saying.

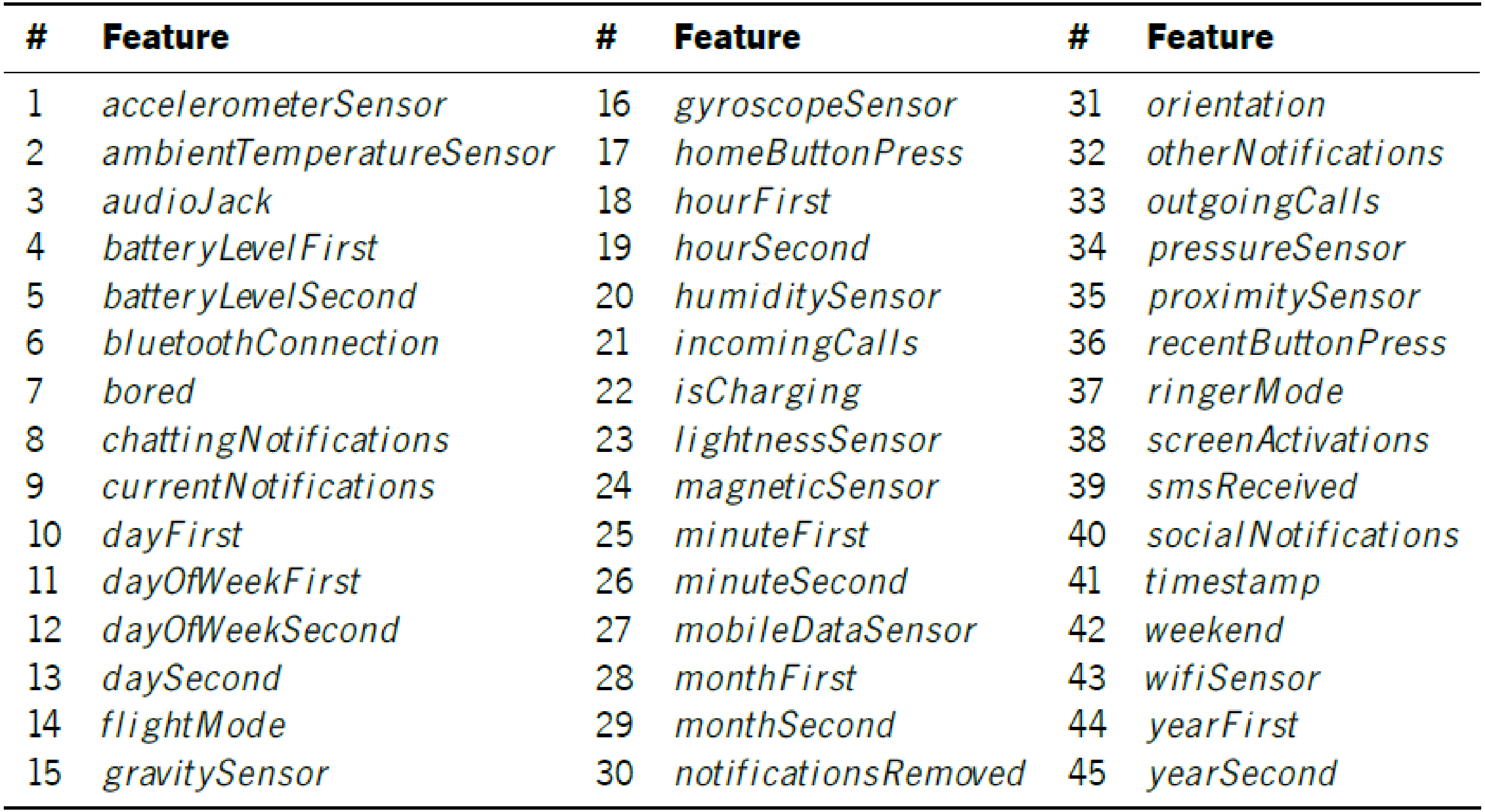
Proximity Sensor
This is what keeps you from accidentally pressing buttons with your cheek during calls!

Magnetometer
The magnetometer measures the strength of the magnetic field around the device to determine what direction it is moving.

Accelerometer
Have you ever wondered how your phone knows which way you are holding it to display vertically vs. horizantally? The acceleromerter is the answer!

Gyroscope
If you like taking non-blurry photos you have the gyroscope to thank. It helps to correct for camera shake.

Mobile Sensors



#	Feature	#	Feature	#	Feature
1	*accelerometerSensor*	16	*gyroscopeSensor*	31	*orientation*
2	*ambientTemperatureSensor*	17	*homeButtonPress*	32	*otherNotifications*
3	*audioJack*	18	*hourFirst*	33	*outgoingCalls*
4	*batteryLevelFirst*	19	*hourSecond*	34	*pressureSensor*
5	*batteryLevelSecond*	20	*humiditySensor*	35	*proximitySensor*
6	*bluetoothConnection*	21	*incomingCalls*	36	*recentButtonPress*
7	*bored*	22	*isCharging*	37	*ringerMode*
8	*chattingNotifications*	23	*lightnessSensor*	38	*screenActivations*
9	*currentNotifications*	24	*magneticSensor*	39	*smsReceived*
10	*dayFirst*	25	*minuteFirst*	40	*socialNotifications*
11	*dayOfWeekFirst*	26	*minuteSecond*	41	*timestamp*
12	*dayOfWeekSecond*	27	*mobileDataSensor*	42	*weekend*
13	*daySecond*	28	*monthFirst*	43	*wifiSensor*
14	*flightMode*	29	*monthSecond*	44	*yearFirst*
15	*gravitySensor*	30	*notificationsRemoved*	45	*yearSecond*

Mobile Sensors (Android)

- Use **Broadcast Receivers** to subscribe to specific events that may happen at any time (similarly to the publish-subscribe pattern):
  - Such events may be raised by the Android system or any other application;
  - When a broadcast is sent, the Android system routes broadcasts to all applications that have subscribed to it;
  - Broadcasts can be received through manifest-declared receivers or context-registered receivers.

- Examples of **Broadcast Receivers** include:
  - audio jack receiver;
  - boot receiver;
  - calls receiver;
  - keys receiver;
  - screen receiver;
  - and others.

https://developer.android.com/develop/background-work/background-tasks/broadcasts

Mobile Sensors (Android)

- Audio Jack Receiver example:

```
class AudioJackReceiver: BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent) {
        if (intent.action == Intent.ACTION_HEADSET_PLUG)
            FB.audioJack = intent.getIntExtra("state", -1).toFloat()
    }

    fun getFilter(): IntentFilter? {
        val filter = IntentFilter()
        filter.addAction(Intent.ACTION_HEADSET_PLUG)
        filter.priority = 1000
        return filter
    }
}
…
//in a service, activity, or other class you can then:
registerReceiver(audioJackReceiver, audioJackReceiver.getFilter())
unregisterReceiver(audioJackReceiver)
```

Mobile Sensors (Android)

- Use **Sensors Event Listeners** to receive notifications from the sensor manager when there is new sensor data.

- Examples of **Sensors Event Listeners** include:
    - Accelerometer Sensor Listener;
    - Ambient Temperature Sensor Listener;
    - Gravity Sensor Listener;
    - Gyroscope Sensor Listener;
    - Humidity Sensor Listener;
    - Light Sensor Listener;
    - Magnetic Sensor Listener;
    - Pressure Sensor Listener;
    - Proximity Sensor Listener;
    - and others.

https://developer.android.com/reference/android/hardware/SensorEventListener

Mobile Sensors (Android)

- Accelerometer Sensor Listener example:

```
class AccelerometerSensorListener: SensorEventListener {
    private lateinit var sensorManager: SensorManager
    override fun onSensorChanged(event: SensorEvent) {
        if (event.sensor.type == Sensor.TYPE_ACCELEROMETER)
            FB.accelerometerSensor = event.values[0]
            sensorManager.unregisterListener(this)
        }
    }
    fun setSensorManager(sensorMan: SensorManager) {
        sensorManager = sensorMan
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
}
…
//in a service, activity, or other class you can then:
var sensorManager: SensorManager = SensorUtils.getSensorManager(context)
var mAccelerometer = getSensor(sensorManager, Sensor.TYPE_ACCELEROMETER)
if (mAccelerometer != null){
       val accelerometerSensorListener = AccelerometerSensorListener()
       accelerometerSensorListener.setSensorManager(sensorManager)
       sensorManager.registerListener(accelerometerSensorListener, mAccelerometer, 0)
}
```

Android App for Sensors

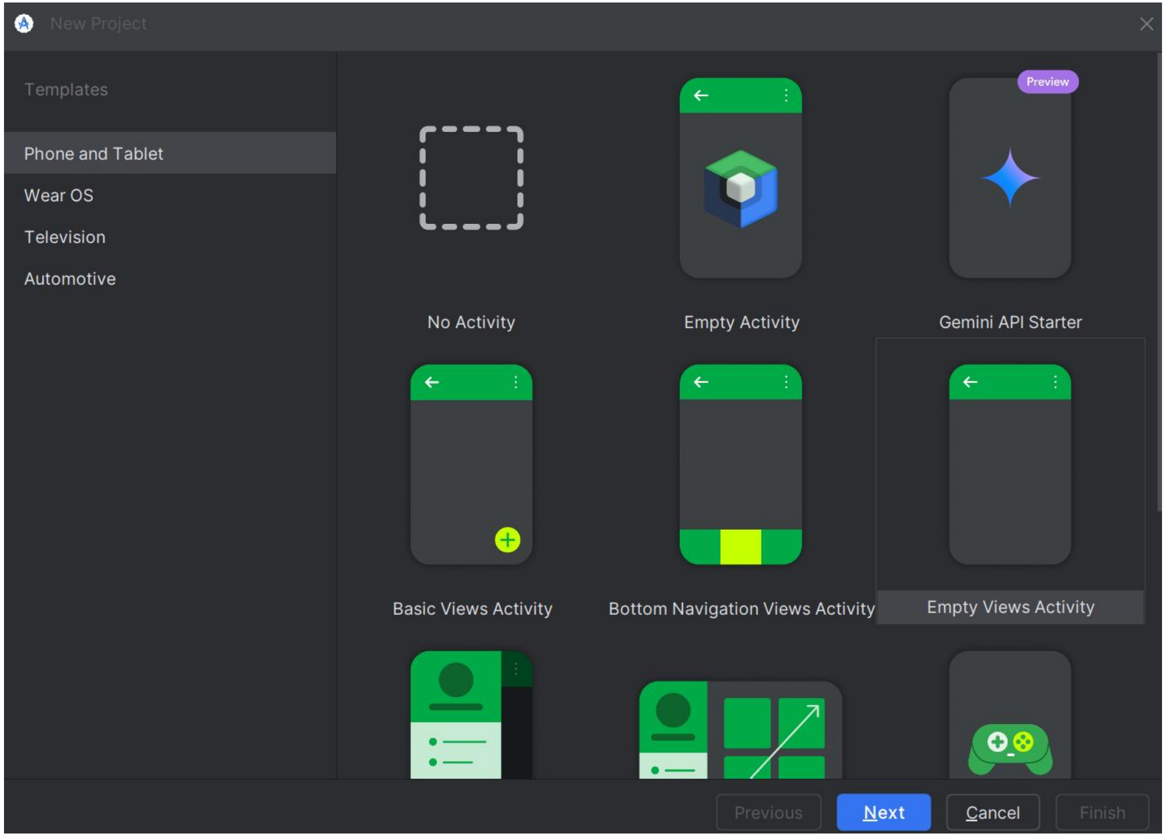
- Let’s create a mobile app and implement **Sensors Event Listeners** to see what is happening with smartphone’s sensors.

- Requirements:
  1. **Android Studio**;
  2. **USB debugging** enabled - if you have an Android smartphone, you can use it to develop and debug. You may need to install some drivers on your PC, and you will need to activate in your smartphone the developer options (usually going to Settings > System > About > and tap the build number 7 times). Then, go the developer options that now appear in the smartphone and enable the USB debugging option;
  3. You may also use the IDE’s emulator, but it may not work properly.

https://developer.android.com/codelabs/basic-android-kotlin-compose-first-app#0

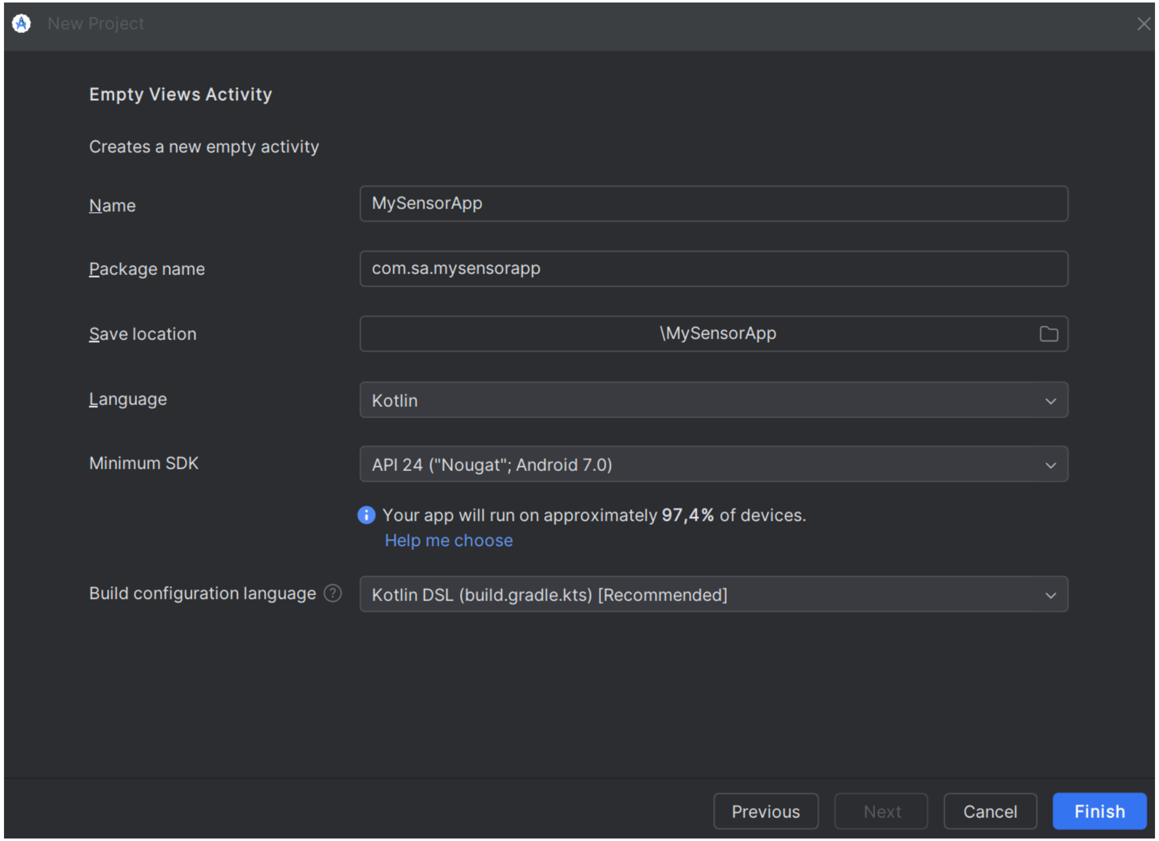
Android App for Sensors

An **activity** interacts with the user, so it **creates a window for you in which you can place your UI** with `setContentView(View)`



Android App for Sensors

Give a **name** to your app and select the programming language: **Kotlin** or **Java**. The next slides will be in Kotlin but you can use Java



Empty Views Activity

Creates a new empty activity

Name: MySensorApp

Package name: com.sa.mysensorapp

Save location: \MySensorApp

Language: Kotlin

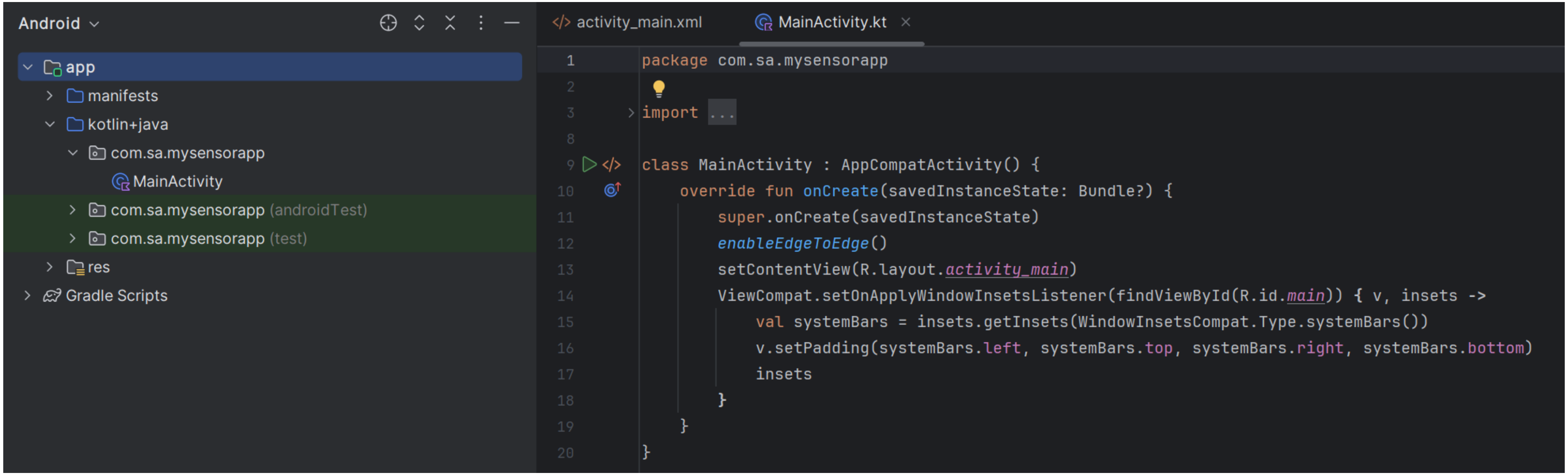
Minimum SDK: API 24 ("Nougat"; Android 7.0)

Your app will run on approximately **97,4%** of devices.

Help me choose

Build configuration language: Kotlin DSL (build.gradle.kts) [Recommended]

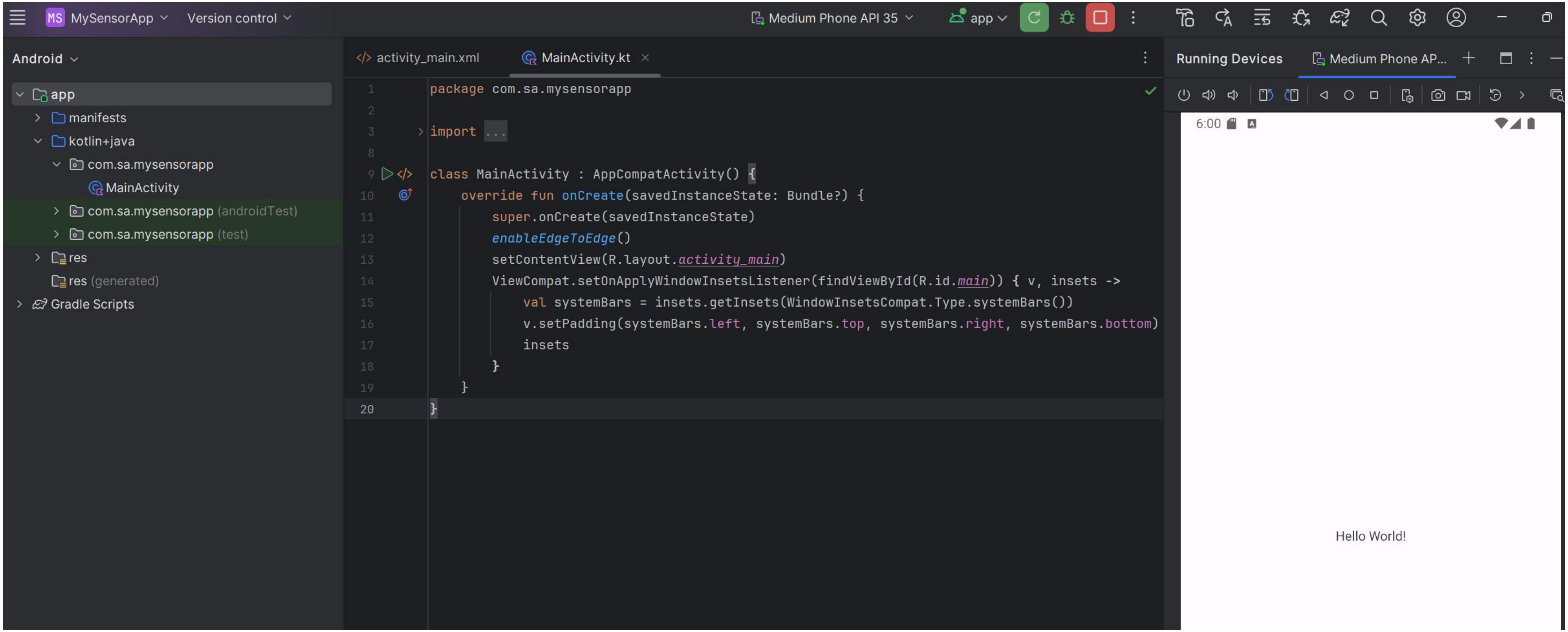
Android App for Sensors



```kotlin
package com.sa.mysensorapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

Android App for Sensors

Build your app using your smartphone or an emulator

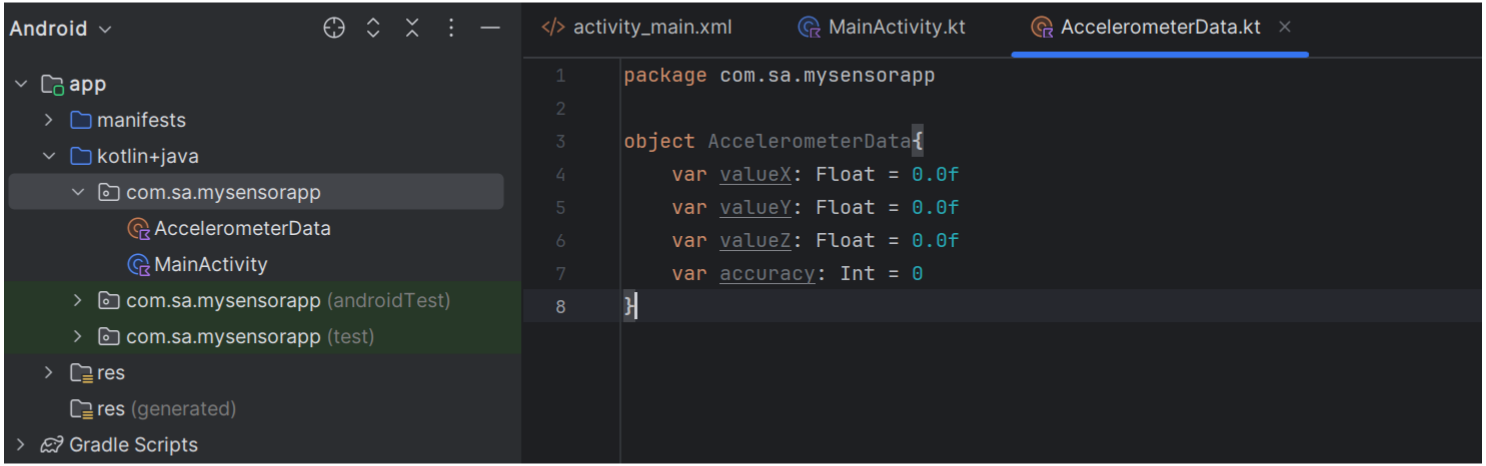


```kotlin
package com.sa.mysensorapp

import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
            val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
            insets
        }
    }
}
```

Hello World!

Android App for Sensors

Create the `AccelerometerData` **object** to hold the data

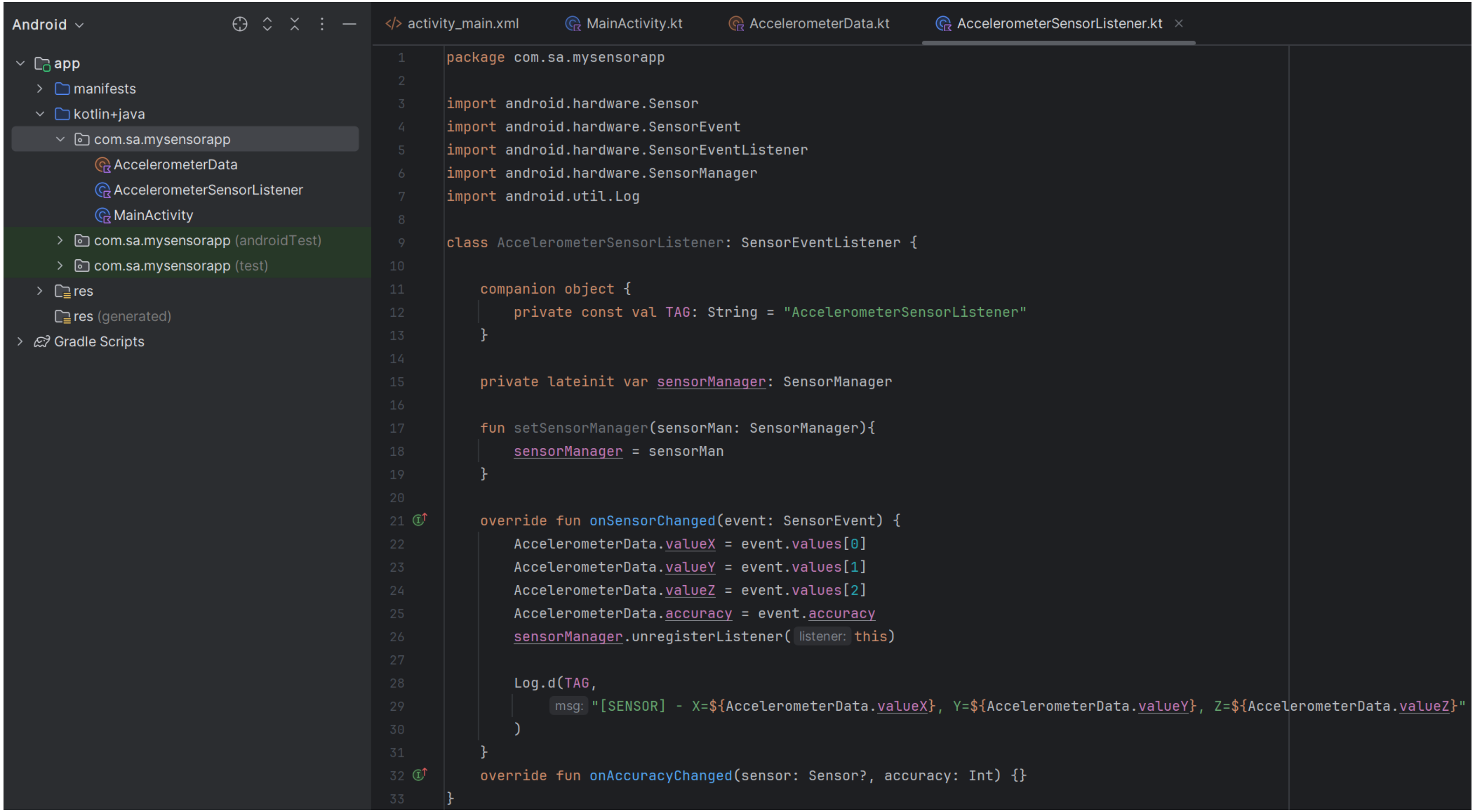


```kotlin
package com.sa.mysensorapp

object AccelerometerData{
    var valueX: Float = 0.0f
    var valueY: Float = 0.0f
    var valueZ: Float = 0.0f
    var accuracy: Int = 0
}
```

Note: this is not the practice. Although it worst, his not the proper way to communicate with an activity. For production apps, you should use the `ViewModel` class, which is designed to store and manage UI-related data in a lifecycle conscious way.

Android App for Sensors

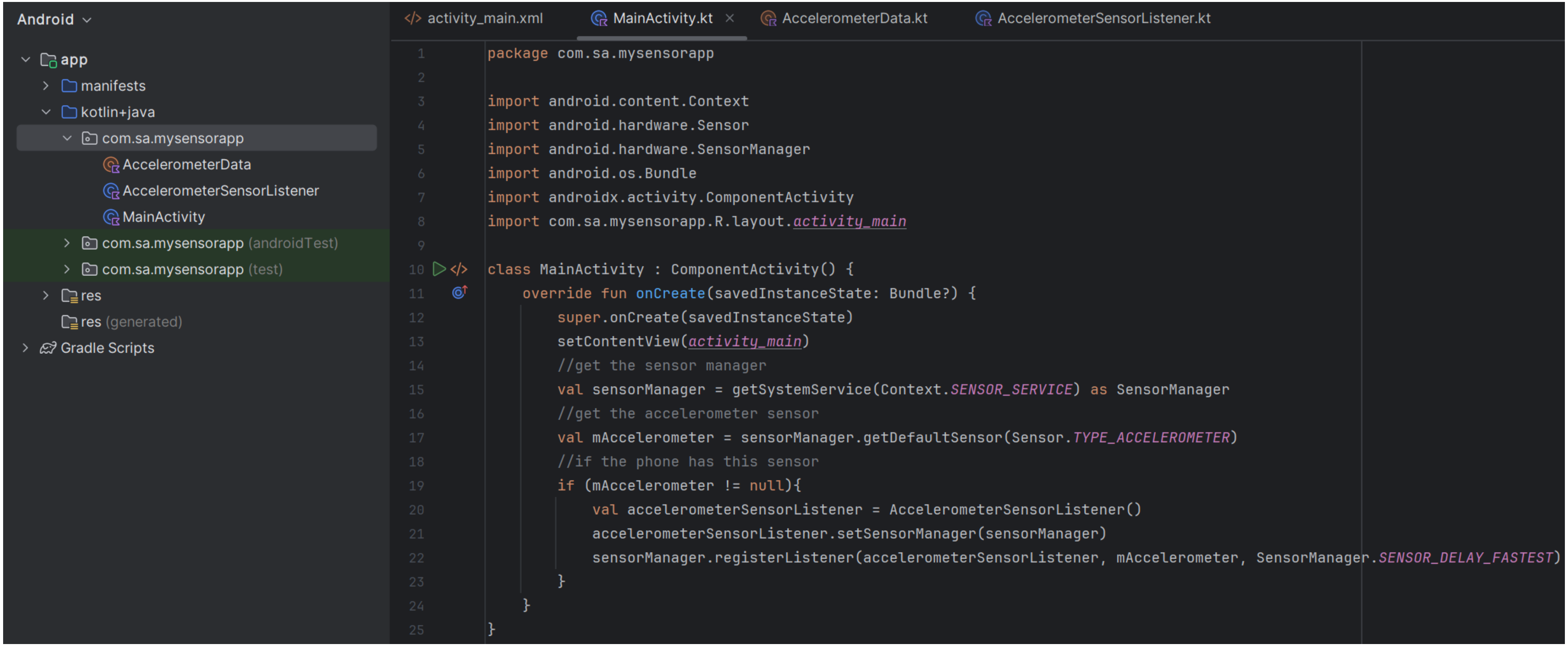
Create the `AccelerometerSensorListener` class, similar to what was shown previously



```kotlin
package com.sa.mysensorapp

import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.util.Log

class AccelerometerSensorListener: SensorEventListener {

    companion object {
        private const val TAG: String = "AccelerometerSensorListener"
    }

    private lateinit var sensorManager: SensorManager

    fun setSensorManager(sensorMan: SensorManager){
        sensorManager = sensorMan
    }

    override fun onSensorChanged(event: SensorEvent) {
        AccelerometerData.valueX = event.values[0]
        AccelerometerData.valueY = event.values[1]
        AccelerometerData.valueZ = event.values[2]
        AccelerometerData.accuracy = event.accuracy
        sensorManager.unregisterListener(this)

        Log.d(TAG,
            "[SENSOR] - X=${AccelerometerData.valueX}, Y=${AccelerometerData.valueY}, Z=${AccelerometerData.valueZ}"
        )
    }
    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
}
```

Android App for Sensors

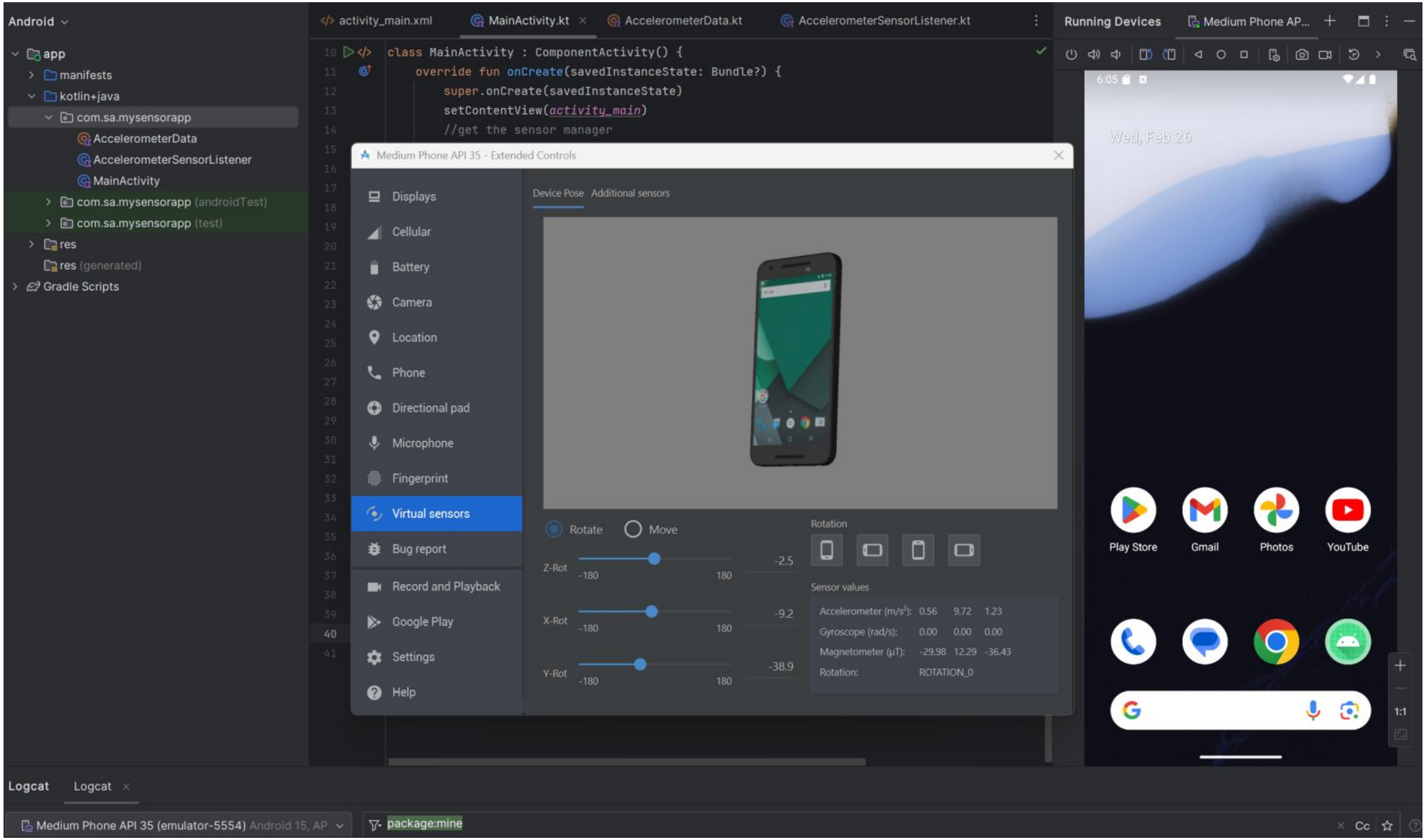
Call the **listener** in the `MainActivity` class



```kotlin
package com.sa.mysensorapp

import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorManager
import android.os.Bundle
import androidx.activity.ComponentActivity
import com.sa.mysensorapp.R.layout.activity_main

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(activity_main)
        //get the sensor manager
        val sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager
        //get the accelerometer sensor
        val mAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
        //if the phone has this sensor
        if (mAccelerometer != null){
            val accelerometerSensorListener = AccelerometerSensorListener()
            accelerometerSensorListener.setSensorManager(sensorManager)
            sensorManager.registerListener(accelerometerSensorListener, mAccelerometer, SensorManager.SENSOR_DELAY_FASTEST)
        }
    }
}
```

Android App for Sensors

Run the app and check the **log**



```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(activity_main)
        //get the sensor manager
```

● Rotate ○ Move

Z-Rot -180 180 -2.5

X-Rot -180 180 -9.2

Y-Rot -180 180 -38.9

```
Accelerometer (m/s²):  0.56    9.72   1.23
Gyroscope (rad/s):     0.00    0.00   0.00
Magnetometer (μT):   -29.98   12.29 -36.43
Rotation:             ROTATION_0
```

Medium Phone API 35 (emulator-5554) Android 15, AP

package:mine

Universidade do Minho
Departamento de Informática

Hands On

Hands On

Discover and implement:

- **For even student numbers**
    - Display this info in the app’s UI

- **For odd student numbers**
    - Dump this data into Firebase/Adafruit