# Distributed Systems – Assignment 1

Vincent Becker

vincent.becker@inf.ethz.ch

# The Exercise

- Objectives
  - Get familiar with Android programming
    - Emulator, debugging, deployment
  - Learn to use UI elements and to design an Activity
  - Learn how to connect Activities and Services using Intents
  - Learn how to use the Sensor API
  - Tackling problems with developing a real app

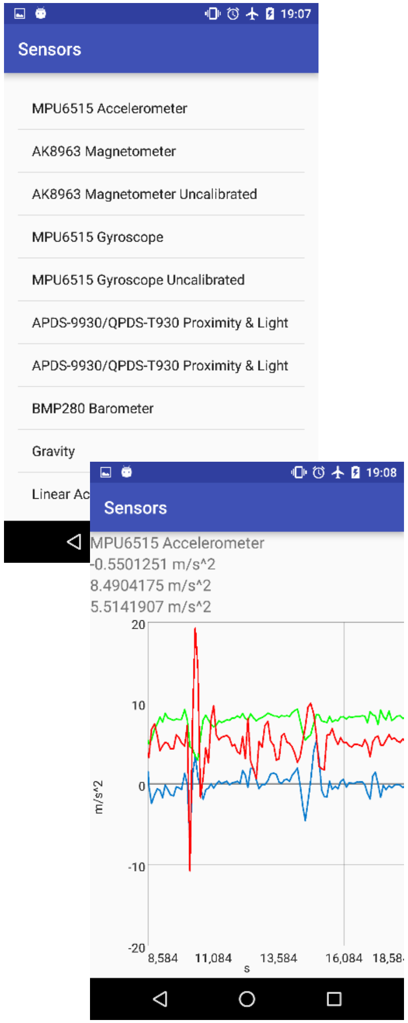
- Dates:
  - Exercise begins: Now
  - Exercise is due: 11:59 p.m., 12th October 2017

# The Exercise

- **Task 1:** Sensors
  - Create an application to access all available sensors
  - Use selected actuators
- **Task 2:** AntiTheft Alarm
  - Create an application to *“secure”* the device against theft
- **Task 3:** Bluetooth Low Energy
  - Create an application to sense temperature and humidity

# Task 1: Sensors and Actuators

- **Objectives:**
  - Learn how to create an Android project
  - Familiarize yourself with UI Elements
  - Understand the concept of Activities and Intents
  - Learn to use the sensor API
  - Show data in a graph
  - Learn how to execute automated tests
- **To do:**
  - Write an app that displays all available sensors in a `ListView`
  - Show sensor readings in a graph in a second activity
  - You MUST implement the following interfaces:
    - SensorTypes
    - GraphContainer



19:07
Sensors

MPU6515 Accelerometer
AK8963 Magnetometer
AK8963 Magnetometer Uncalibrated
MPU6515 Gyroscope
MPU6515 Gyroscope Uncalibrated
APDS-9930/QPDS-T930 Proximity & Light
APDS-9930/QPDS-T930 Proximity & Light
BMP280 Barometer
Gravity
Linear Ac

19:08
Sensors

MPU6515 Accelerometer
-0.5501251 m/s^2
8.4904175 m/s^2
5.5141907 m/s^2

# Hints

- Project names: VS_*nethz*_Sensors (leader's nethz ID)
- Do not forget to add all components (Activities, Services) to the application in the manifest file
- Do not forget the permissions in the manifest file
- Listing all the sensors:

```
private SensorManager sensorMgr;
private List <Sensor> sensors;
```

```
sensorMgr = (SensorManager) getSystemService(SENSOR_SERVICE);
sensors = sensorMgr.getSensorList(Sensor.TYPE_ALL);
```

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

# SensorTypes

```
private int getNumberValues(int sensorType);
private String getUnitString(int sensorType);
```

- Different values for different sensors. E.g.:
- LIGHT:

```
SensorTypeImpl.getNumberValues(Sensor.TYPE_LIGHT); -> 1
SensorTypeImpl.getUnitString(Sensor.TYPE_LIGHT); -> “lx”
```

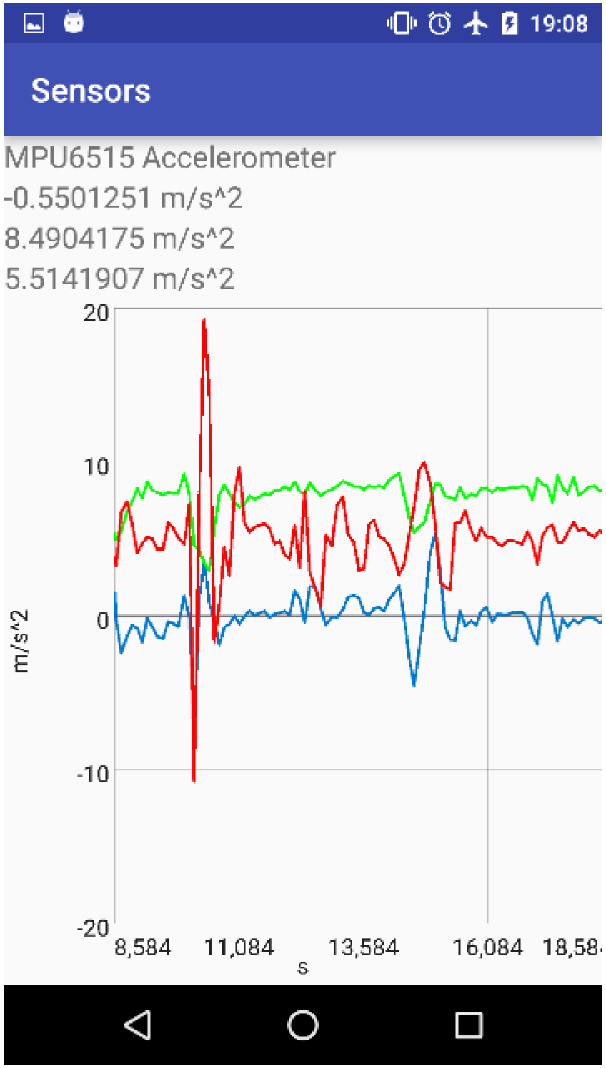
- ACCELERATION:

```
SensorTypeImpl.getNumberValues(Sensor.TYPE_ACCELERATION); -> 3
SensorTypeImpl.getUnitString(Sensor.TYPE_ACCELERATION); -> “m/s^2”
```

# Graph

- Use the Graph View library: http://www.android-graphview.org/
- Easy import in Gradle script

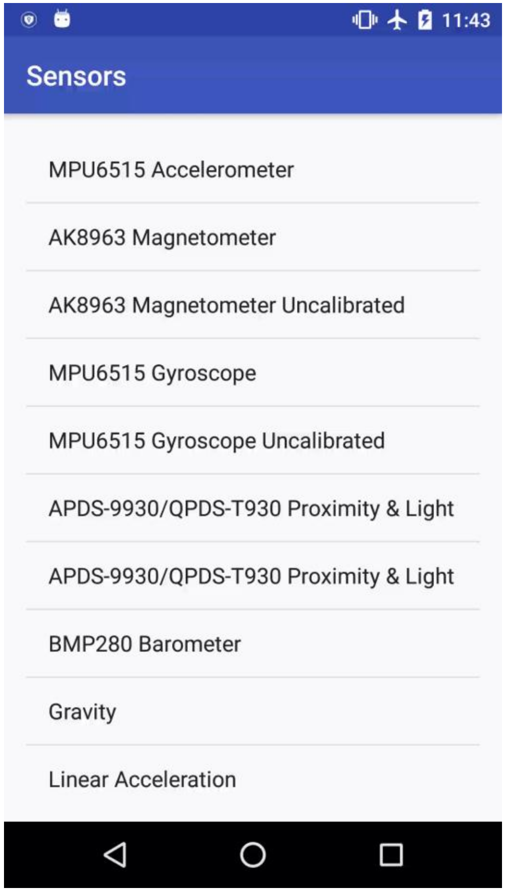
```
compile 'com.jjoe64:graphview:4.2.1'
```



Sensors

MPU6515 Accelerometer
-0.5501251 m/s^2
8.4904175 m/s^2
5.5141907 m/s^2

# Graph



Sensors

MPU6515 Accelerometer

AK8963 Magnetometer

AK8963 Magnetometer Uncalibrated

MPU6515 Gyroscope

MPU6515 Gyroscope Uncalibrated

APDS-9930/QPDS-T930 Proximity & Light

APDS-9930/QPDS-T930 Proximity & Light

BMP280 Barometer

Gravity

Linear Acceleration

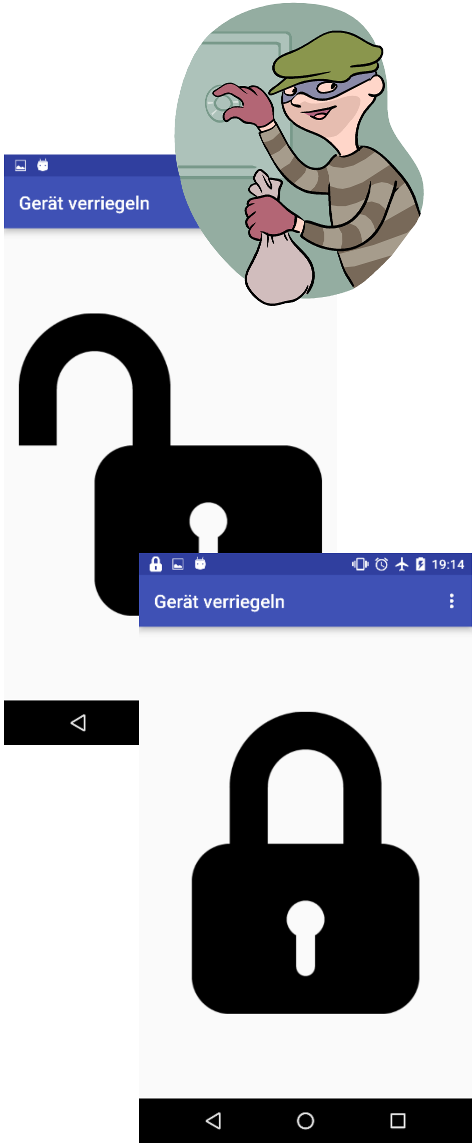
# Hints

- When starting `SensorActivity`, the Intent should carry the information which sensor to display in detail

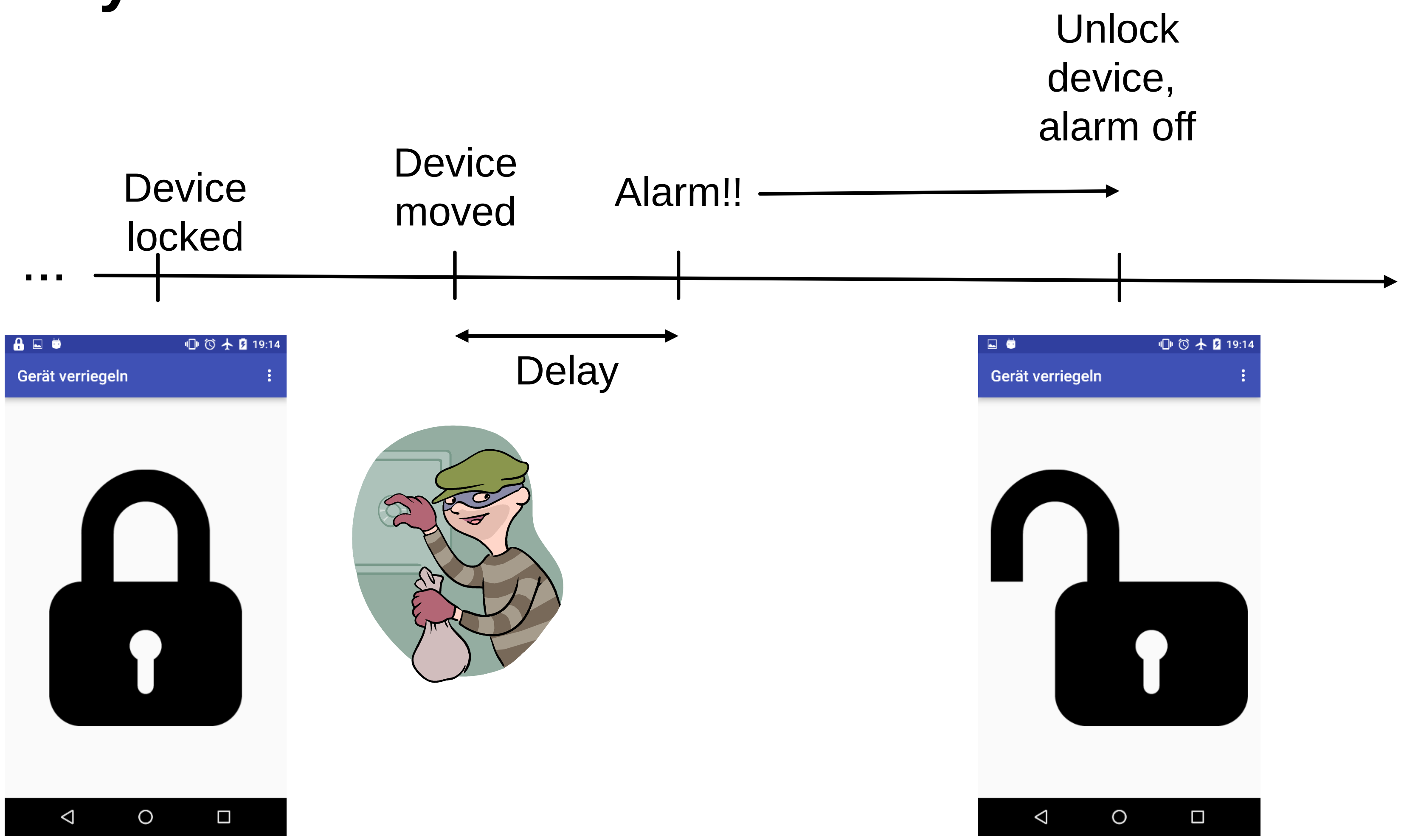
- `SensorActivity` should implement the `SensorEventListener` interface and continuously present the sensor’s value(s)

# Task 2: AntiTheft Alarm

- **Objectives:**
  - Transfer the knowledge of Task 1 into a real app
    - Understand problems stemming from a framework under development
  - Think about how to make use of the sensors
  - Learn how to use background services
- **To do:**
  - Write an app to *“secure”* the device against theft
    - Sound an alarm when the device is moved without authorisation
  - You MUST implement:
    - AbstractMovementDetector



# Delay



...

Device locked

Device moved

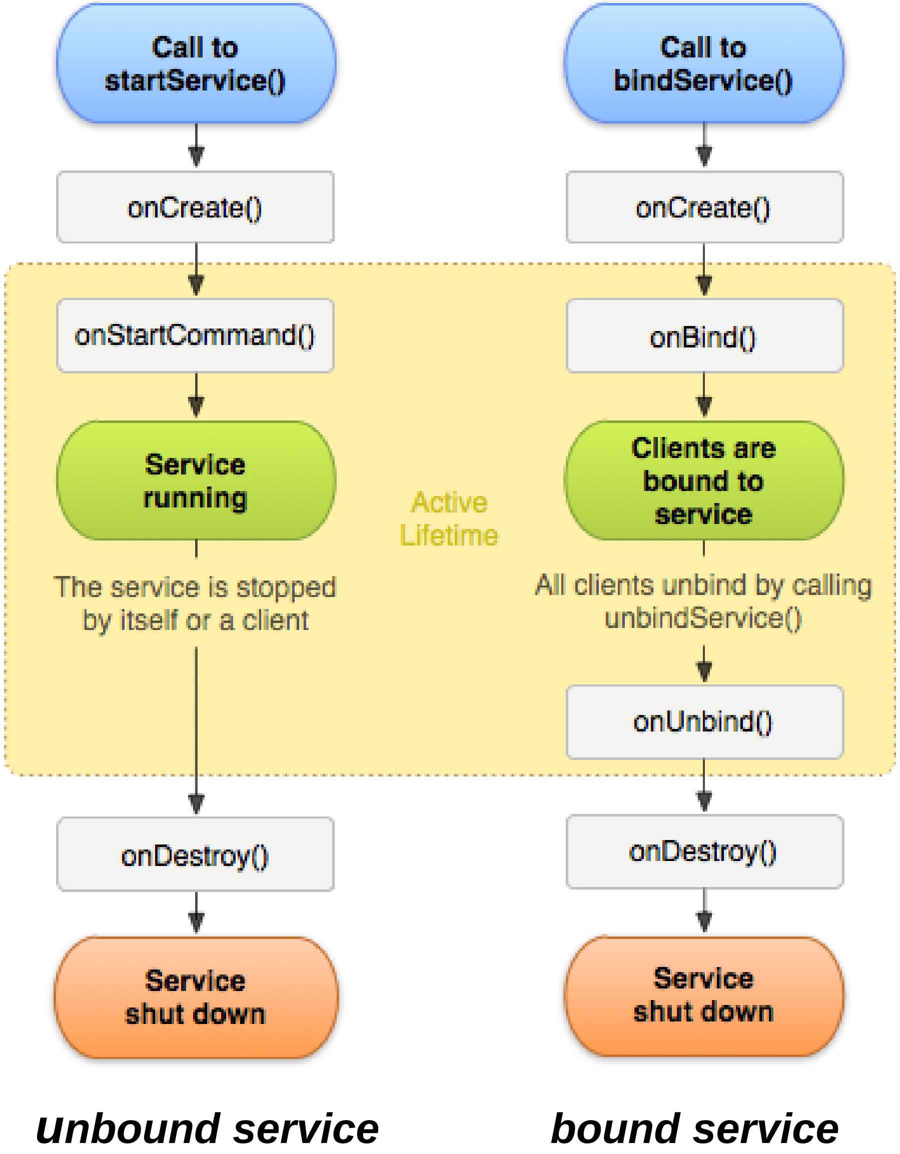
Alarm!!

Unlock device, alarm off

Delay

# Hints

- Study service lifecycle
- A **Started** Service (unbound service) is good for us



http://developer.android.com/guide/components/services.html

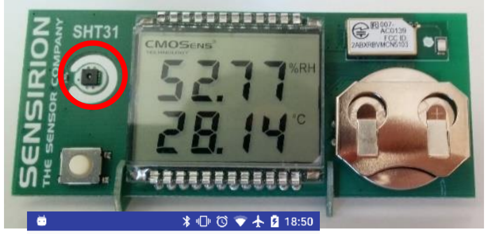
# Hints

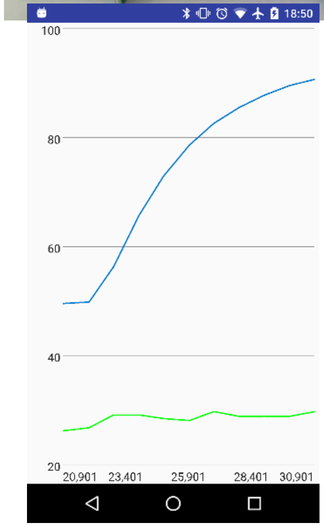
- We do not need to deal with inter-process communication
- See examples on the Android website
- Do not forget to declare the Service in the manifest file!

# Task 3: Bluetooth Low Energy

https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

- **Objective:**
  - Exploit the capabilities of smart gadgets and bluetooth low energy
  - Familiarize with the new generation of ‘wearable computers’
- **To do:**
  - Use Android’s Bluetooth Low Energy (BLE) framework
  - Connect to an SHT31 Smart Gadget to sense temperature and humidity
  - Display the current sensor measurements to the user





# Hints

- You have been given a Sensirion sensor
- Update the software version on the phone to at least 5.0
- Check first if bluetooth is enabled on the device
- Request to enable bluetooth without leaving the app
- Limit the scan time, do not drain the battery

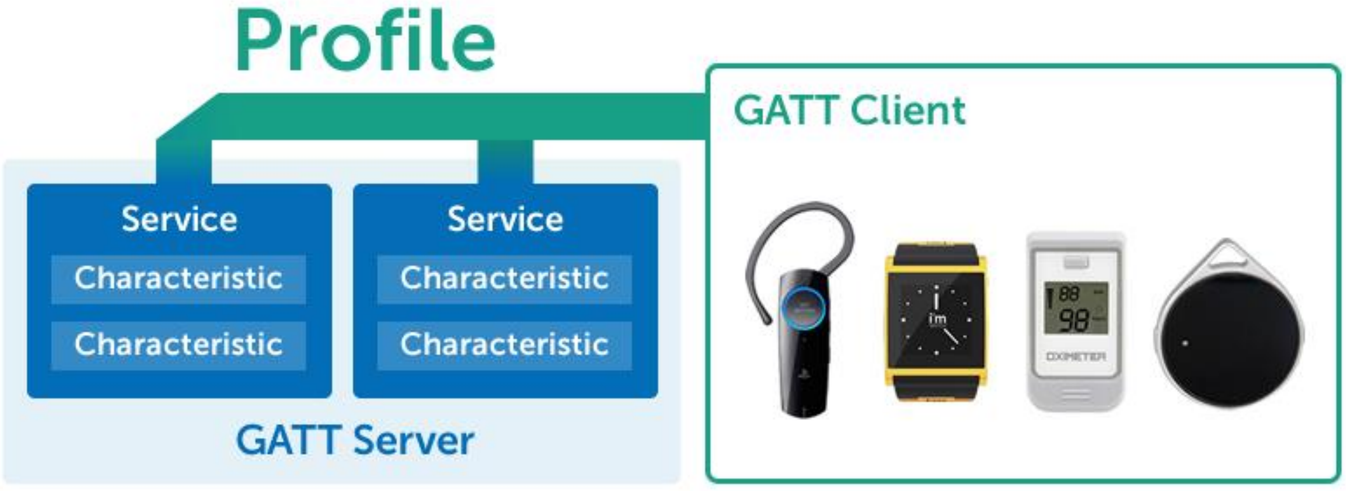
# Hints II

- From Android 6.0 ( ≥ API level 23):
  - Permissions have to be stated in the manifest
  - Dangerous permissions have to be requested at runtime (once, or until the user revokes the permission again)
  - E.g. ACCESS_FINE_LOCATION
  - You also have to enable the location service

  - More information:
    - https://developer.android.com/training/permissions/index.html

# Hints

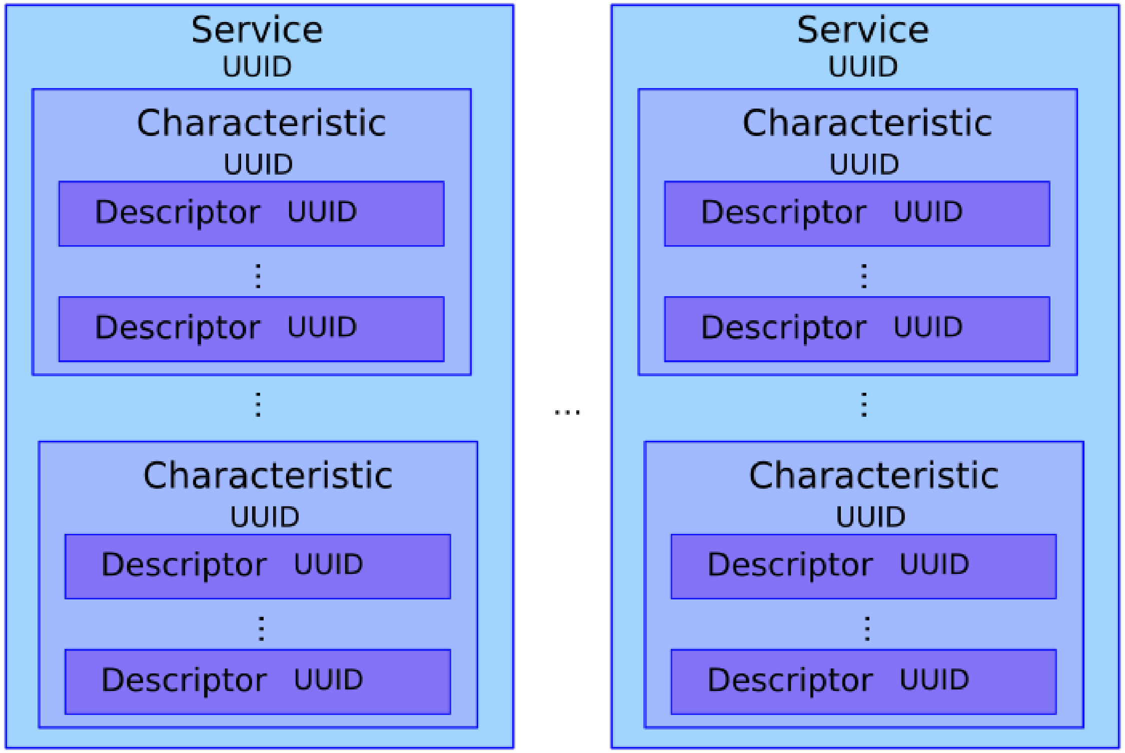
- The Android phone acts as a GATT client
- The SHT31 smart gadget acts as a GATT server
- We are interested in two services, humidity and temperature



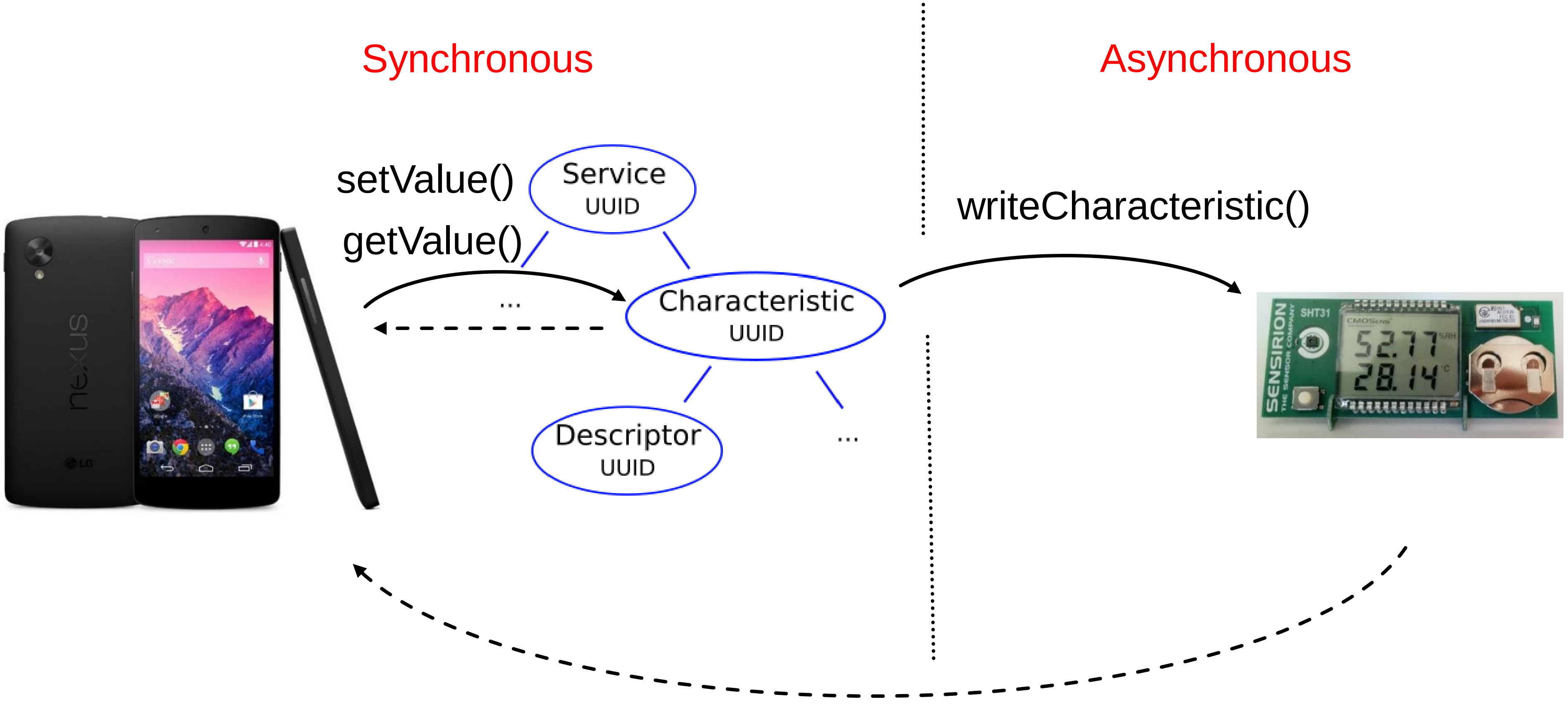
http://blog.lemberg.co.uk/

# Hints

- A UUID is a unique identifier for a service or a characteristic
- Characteristic: Attribute containing a single value
- Descriptor: describes its characteristic's value, e.g. indication about the unit or configuration



# BLE communication



setValue()

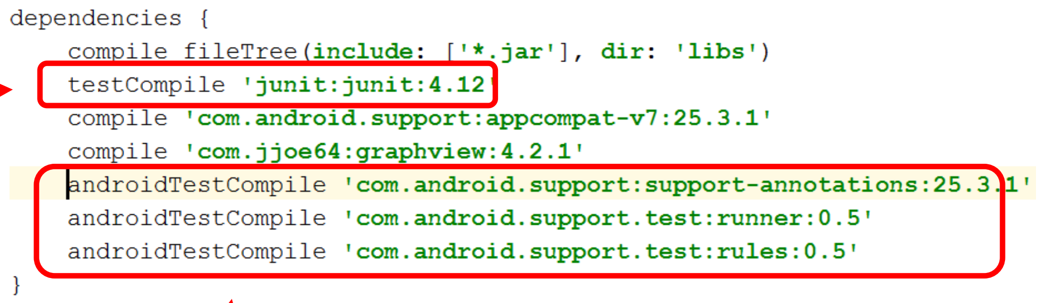
getValue()

writeCharacteristic()

onCharacteristicWrite()
onCharacteristicRead()
onCharacteristicChanged()

# Testing

- JUnit Tests
  - Test single ‘units‘
  - Test algorithm / logic
  - Run on local JVM
  - No access to functional Android framework APIs



```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.jjoe64:graphview:4.2.1'
    androidTestCompile 'com.android.support:support-annotations:25.3.1'
    androidTestCompile 'com.android.support.test:runner:0.5'
    androidTestCompile 'com.android.support.test:rules:0.5'
}
```

- Activity Test Rules
  - Run test on physical devices and emulators
  - Uses Android framework APIs
  - E.g. access to Context

# Mini-Test

- Required for each project
- LaTeX or Word
- English
- 6 questions (essay, true/false, code snippets, …etc)
- General Android questions + Assignment-related questions
- Submit as **answers.pdf** in PDF format

# Deliverables

- https://www.vs.inf.ethz.ch/edu/vs/submissions/
- Use your *nethz* logins
- File names: **answers.pdf** and **code.zip**
- New uploads will overwrite the old ones
- Check uploaded files

LEADER:

- Create group
- Add members
- Upload files

MEMBERS:

- Sign the submission

# Remarks

- Point reduction if the project does not compile!
- Point reduction if the project does not use the code skeleton provided to you!

- Please remove the build folders before submitting your solution (on project level and app level)

# Hardware

- You can keep the Sensirion sensors until the end of the lecture
- Then you have to return them (the smartphones too of course)!

# Team Work & Communication

- If you have a problem

1. Think about it
2. Read the Android guides and documentation
3. Ask your team mates!
4. Write an email
   - How to write an email:
     - State a clear description of the problem and what you have tried to solve it
     - Put all your team members in **cc**!! This ensures that you are “asking“ them at least via email

# Have fun!

Questions?