Everything is better with(out) Bluetooth

This presentation is available at
https://docs.google.com/presentation/d/1lSReYeUx0t_KwUnmn1NX47ozq8uY5GoZ4xzz0lwlJ3s/

About us

Falko

- Full stack tinkerer
- Cargo Bike Pimper
- Chief (technical) Product Manager at Sensorberg[Android/iOS dev]
- about.me/falkorichter @volkersfreunde

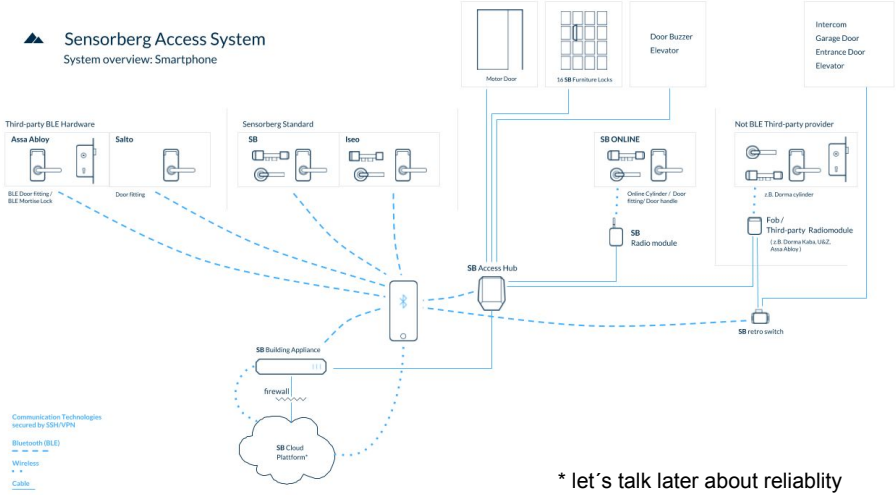
Ronaldo

- Android Dev
- Raspberry Pi enthusiast
- Expert Diaper changer
- github.com/budius | @ronaldopace

Project Overview

- Sensorberg
  - Smart Spaces — where people and buildings interact
  - Getting into a space is the first interaction
  - Of course with your phone!
  - Bluetooth LE really is the only common denominator for a reliable connection
    - offline capable, fast, reliable*

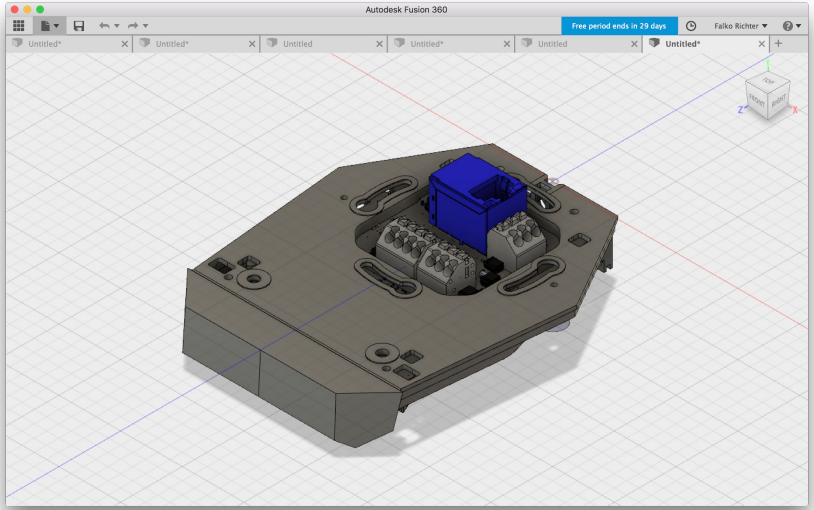
- Interactions currently:
  - book a meeting room
  - open a locker
  - open the entrance
  - open a garage

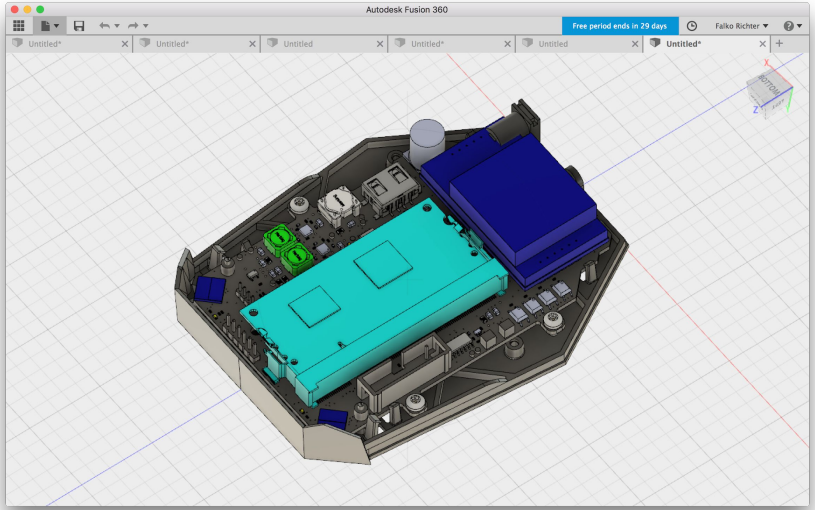


* let´s talk later about reliablity

Project Overview

- Device needed which has BLE, internet & can open door hardware
  - Pi compute Module Platform
  - Custom board 100% made in Berlin
  - All based on open source software & hardware





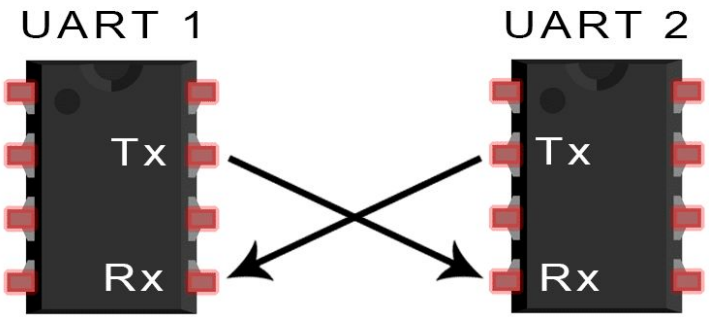
BLE - what you should know

- GATT (The Generic Attributes (GATT) define a hierarchical data structure that is exposed to connected Bluetooth Low Energy (LE) devices)
  - Service
    - Group of Characteristics
  - Characteristic
    - Write/ Read /Subscribe
    - has a UUID
- Descriptions availble on bluetooth.com
  - https://www.bluetooth.com/specifications/gatt
  - you can also make up your own!

Project Overview - universal bluetooth communication

UART/Serial Port Emulation over BLE

- open source / standard by nordic
- https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/uart-service
  - find (by Service)
  - connect
  - discover
  - subscribe to Rx
  - **write to Tx** **Request**
  - **wait for answer on Rx** **Response**
  - disconnect



https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.sdk5.v14.0.0%2Fble_sdk_app_nus_eval.html

Challenge

Open a door/locker/garage as fast as possible

~ 0.5 million bluetooth connects (both platforms) add up to a lot of (potentially wasted) lifetime

Displaying Access Requests **1 - 25** of **422029** in total

How to Bluetooth-LE Scan

```
private val bleScanner = BluetoothAdapter.getDefaultAdapter().bluetoothLeScanner

override fun onStart() {
        super.onStart()
        bleScanner.startScan(callback) // optional scan filters and scan settings
}

override fun onStop() {
        bleScanner.stopScan(callback)
        super.onStop()
}

private val callback = object : ScanCallback() {
        override fun onScanResult(callbackType: Int, result: ScanResult) {
                Log.d("MyApp", "onScanResult: $result. rssi: ${result.rssi} dB")
        }
}
```

How to Bluetooth-LE GATT communication

```
if (isRightDevice(result)) result.device.connectGatt(context, false, gattCallback)

private val gattCallback = object : BluetoothGattCallback() {
        override fun onConnectionStateChange(gatt: BluetoothGatt, status: Int, newState: Int) {
                if (isConnected(status, newState)) gatt.discoverServices()
                else gatt.close()
        }

        override fun onServicesDiscovered(gatt: BluetoothGatt, status: Int) {
                val gattCharacteristic = gatt.getService(serviceUuid).getCharacteristic(characteristicWrite)
                gattCharacteristic.value = "open door"
                gatt.writeCharacteristic(gattCharacteristic)
        }

        override fun onCharacteristicWrite(gatt: BluetoothGatt, characteristic: BluetoothGattCharacteristic, status: Int) {
                val gattCharacteristic = gatt.getService(serviceUuid).getCharacteristic(characteristicRead)
                gatt.readCharacteristic(gattCharacteristic)
        }

        override fun onCharacteristicRead(gatt: BluetoothGatt, characteristic: BluetoothGattCharacteristic, status: Int) {
                if(characteristic.value == "door opened") {  /* success !!! */ }
                gatt.disconnect()
        }
}
```

(OUT)

Everything is better with Bluetooth

Everything is better with(out) Bluetooth … scanning

- BluetoothAdapter.getDefaultAdapter().startLeScan(scanCallbackV18)
- Some devices doesn’t support certain scan settings parameters
- Some devices doesn’t support scan filters
  - Some devices say they support scan filters, but result on callback never called
- Workaround:
  - Use Nordic Scanner compat https://github.com/NordicSemiconductor/Android-Scanner-Compat-Library
  - // disable hardware filtering
    .setUseHardwareFilteringIfSupported(**false**)

Everything is better with(out) Bluetooth … scan callback

- override fun **onScanResult**(**callbackType:** Int, **result:** ScanResult) {
- On some devices is called for every scan
- on others, only on the first time a BLE device is scanned
- Workaround is to **startScan**/**stopScan** every second
- Starting on Android Nougat Developer Preview 4

  > We've changed the BLE Scanning behavior starting in DP4. We'll prevent applications from starting and stopping scans more than 5 times in 30 seconds. For long running scans, we'll convert them into opportunistic scans.

  - Sources:
    - https://web.archive.org/web/20160820074825/https://developer.android.com/preview/support.html#dp4
    - https://android-review.googlesource.com/c/platform/packages/apps/Bluetooth/+/215844/15/src/com/android/bluetooth/gatt/AppScanStats.java#144

Everything is better with(out) Bluetooth … scan callback

- override fun **onScanFailed**(**errorCode:** Int) {
  // SCAN_FAILED_APPLICATION_REGISTRATION_FAILED
  // SCAN_FAILED_INTERNAL_ERROR
  }
- Workaround:

Everything is better with(out) Bluetooth … scan callback

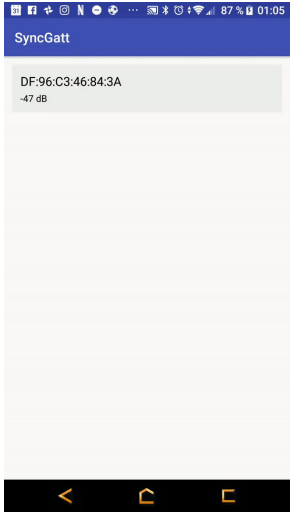
- ```
  private val scanCallback = object : ScanCallback() {
  	override fun onScanResult(callbackType: Int, result: ScanResult) {
  		scanResults[result.device.address] = result

  private fun openClosest() {
  	val scanList = scanResults.values.filter { it.rssi >= threshold }.sortedBy { it.rssi }
  	open(scanList[0])
  }
  ```
- rssi variance
  - time dependent weighted moving average https://github.com/sensorberg-dev/motionless-average
- on device learning of threshold variance
  - start with a permissive default
  - adjust with a moving average of the rssi during successful connection
  - offset a few decibels for actual filtering
  ```

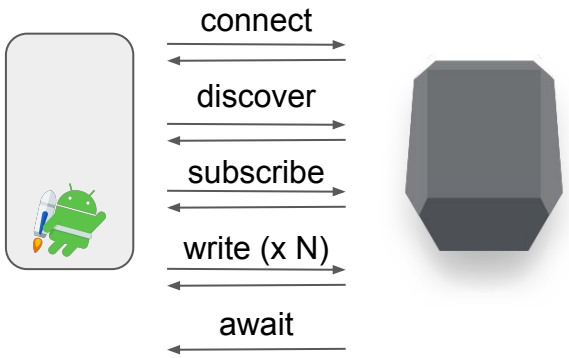
Everything is better with(out) Bluetooth … GATT communication

- `result.device.connectGatt(...)` only works if called from UI-thread
  - on some devices
- override fun **onConnectionStateChange**(**gatt:** BluetoothGatt, **status:** Int, **newState:** Int) {
  if (newState != BluetoothGatt.STATE_CONNECTED) // connect again
- override fun **onCharacteristicWrite**(**gatt:** BluetoothGatt, **characteristic:** BluetoothGattCharacteristic, **status:** Int) {
  override fun **onCharacteristicRead**(**gatt:** BluetoothGatt, **characteristic:** BluetoothGattCharacteristic, **status:** Int) {
  - **if**(**status** != GATT_SUCCESS) // retry
  - **if** (status == **133**)
    - Source code:
      #define GATT_ERROR 0x0085

Everything is better with(out) Bluetooth … GATT communication



- Asynchronous callback based API




```
Thread {
  SynchronousGatt(scanResult.device).apply {
    connectGatt(this@MainActivity, false, 10000)
    discoverServices(3000)

    val notify = bluetoothGatt.getService(serviceUuid)
        .getCharacteristic(characteristicRead).getDescriptor(descriptorNotify)
    notify.value = enableNotify
    writeDescriptor(notify, 3000)

    val writeCharacteristic = bluetoothGatt
        .getService(serviceUuid).getCharacteristic(characteristicWrite)
    writeCharacteristic.setValue("open door")
    writeCharacteristic(writeCharacteristic, 3000)

    val changed = awaitCharacteristicChange(5000)
    if (changed.characteristic.getStringValue(0) == "door opened") sucess()
    disconnect(1000)
  } }.start()
```

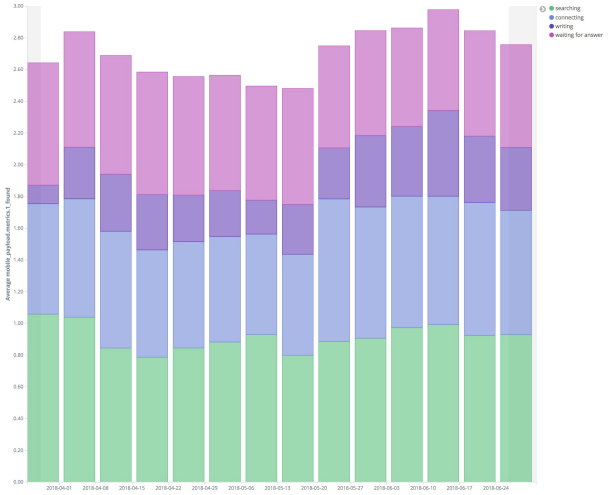
Everything is better with(out) Bluetooth … after much hacking

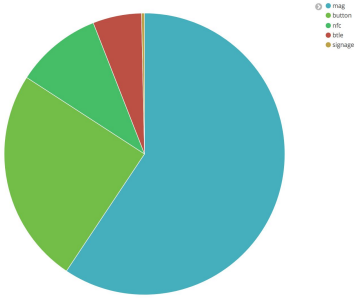
- Scanning:
  - NordicScannerCompat
  - ScannerStartStop
  - ScannerNougat
  - Averaging algorithms
  - Did you try turn off and on again?
- Communicating:
  - Synchronous API
  - Retry on random fail
  - Retry on random disconnection
  - Did you try turn off and on again?

Data

~⅓ time is wasted trying to find the right device

Google is the best manufacturer (74% of access requests under 2.5s)





Device	Last mobile_payload.device.manufacturer.keyword	Time	Affected users	door openings	Percentile rank 2.50 of "Percent Below"	Percentile rank 5.00 of "Percent Below"
HUAWEI	HUAWEI	3.46	77	9,557	62.155%	81.535%
blackberry	blackberry	3.37	2	209	54.723%	81.224%
motorola	motorola	3.19	28	3,202	46.439%	87.674%
Fairphone	Fairphone	3.15	5	831	49.922%	86.467%
Xiaomi	Xiaomi	3.08	12	1,946	61.618%	88.5%
OnePlus	OnePlus	2.92	56	13,940	57.753%	89.395%
bq	bq	2.84	3	344	59.43%	90.244%
HTC	HTC	2.83	11	1,398	62.694%	90.316%
LGE	LGE	2.79	30	5,732	63.593%	90.806%
ZTE	ZTE	2.78	4	26	56.206%	87.186%
LENOVO	LENOVO	2.76	5	1,113	56.657%	92.071%
samsung	samsung	2.60	183	32,622	68.518%	91.909%
Sony	Sony	2.55	31	5,144	68.821%	91.426%
Google	Google	2.29	46	11,288	73.454%	95.331%

manufacturers ordered by average opening time
at least 2 users

Data

```
@POST("/")
@Headers("Content-Type: application/json")
Call<String> pushStatistics(@Body Stats stats);
```

- Firebase is not enough
  - not realtime
  - no filtering
  - aggregations are to limited
- Elasticsearch to the rescue
  - account: 2 minutes https://www.elastic.co/cloud/elasticsearch-service/signup
  - post JSON to your [...].aws.cloud.es.io:9243/{index}/mobile_statistics
  - simple rules for data integrity
    - never change a type (~~String becomes Number, String becomes Object~~)
    - lower case all strings / remove spaces
    - add UUIDs/hashes so you can count devices / users
    - one index per deployment (production / staging)
  - leave out personal data
  - delete indexes (buckets) regularly

Data

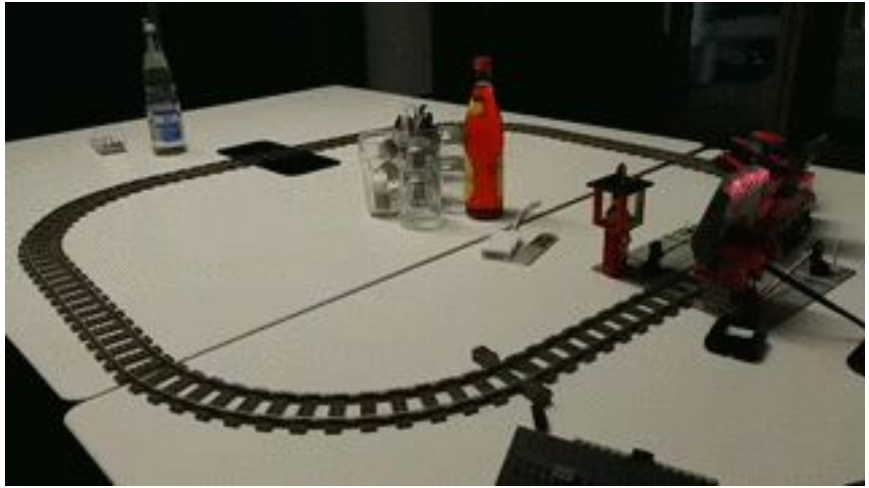
- hack some visualizations in Kibana
- combine visualizations to powerful interactive dashboards
- metrics to find fragmented devices:
  - BuildConfig.VERSION_CODE
  - BuildConfig.VERSION_NAME
  - BuildConfig.APPLICATION_ID
  - Build.VERSION.SDK_INT
  - Build.MODEL
  - Build.MANUFACTURER

… how to avoid bluetooth

- Hybrid detection and communication
  - Detection
    - Bluetooth (“This BTLE device is so close, I should connect to it”)
    - NFC (nfc://connect/to/DF:96:C3:46:84:3A)
    - Optical?!
    - (magnetic) tap detection
  - communication (try all, winner takes it all)
    - Bluetooth
    - HTTPS via wifi/cellular
    - NFC (in the next generation of the hardware)
    - synchronize with a request ID in backend or on access hub

Testing

- LEGO train #ftw
  - get the (not sold anymore) powered tracks (no battery)
  - use two power modules for reliability - it will make the train go reliably slow!
    - Schwerlastzug 60098 is great
- You need to physically test it!
  - real devices
  - actual Bluetooth hardware
  - use your statistics to verify the performance
  - Long term testing
- Parameters
  - speed
  - distance to device
  - additional sensor to detect phone position
  - power the train from the access hub => full control



Thx & Sensorberg Open Source:

- https://github.com/sensorberg-dev/permission-bitte
- https://github.com/sensorberg-dev/motionless-average
- https://github.com/sensorberg-dev/EasyIPC
- https://github.com/sensorberg-dev/gradle-scripts
- https://github.com/sensorberg-dev/android-sdk
- soon: https://github.com/sensorberg-dev/synchronous-gatt

We will of course make an SDK from this as well
https://www.youtube.com/watch?v=rLuBxsLot5U

Another Bluetooth related GDG talk by Falko
https://www.youtube.com/watch?v=OSJ8gIPnvDw