

BLESP *lo.it*

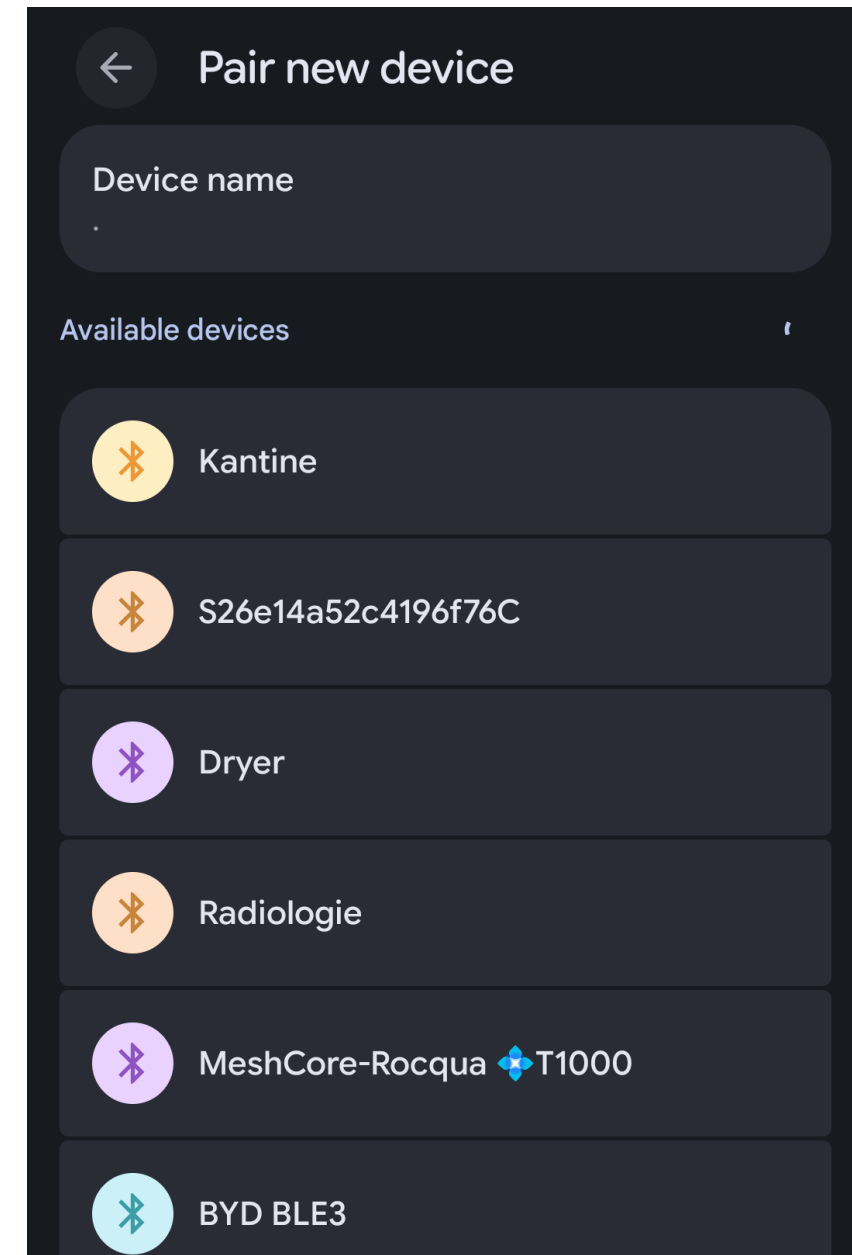


SŁAWOMIR
JASEK



Your Bluetooth experience?

Using a Bluetooth device?



Using a Bluetooth scanner app?

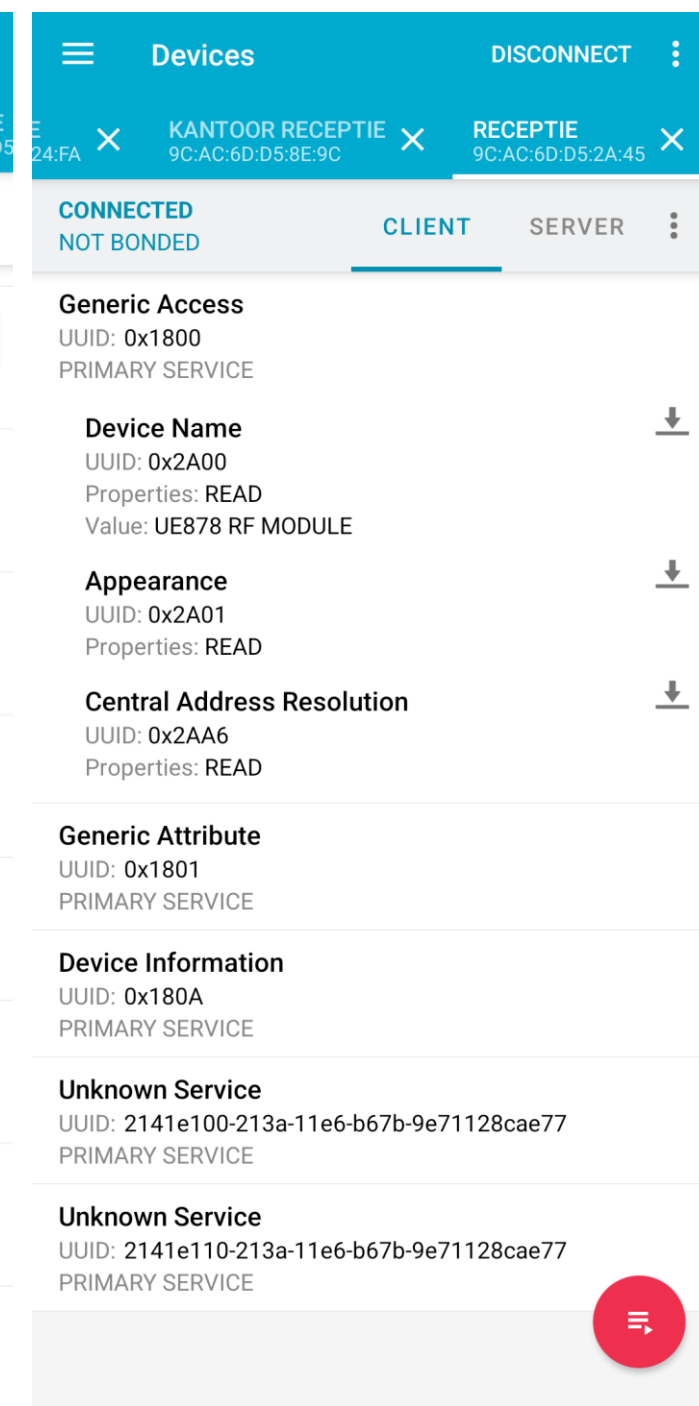
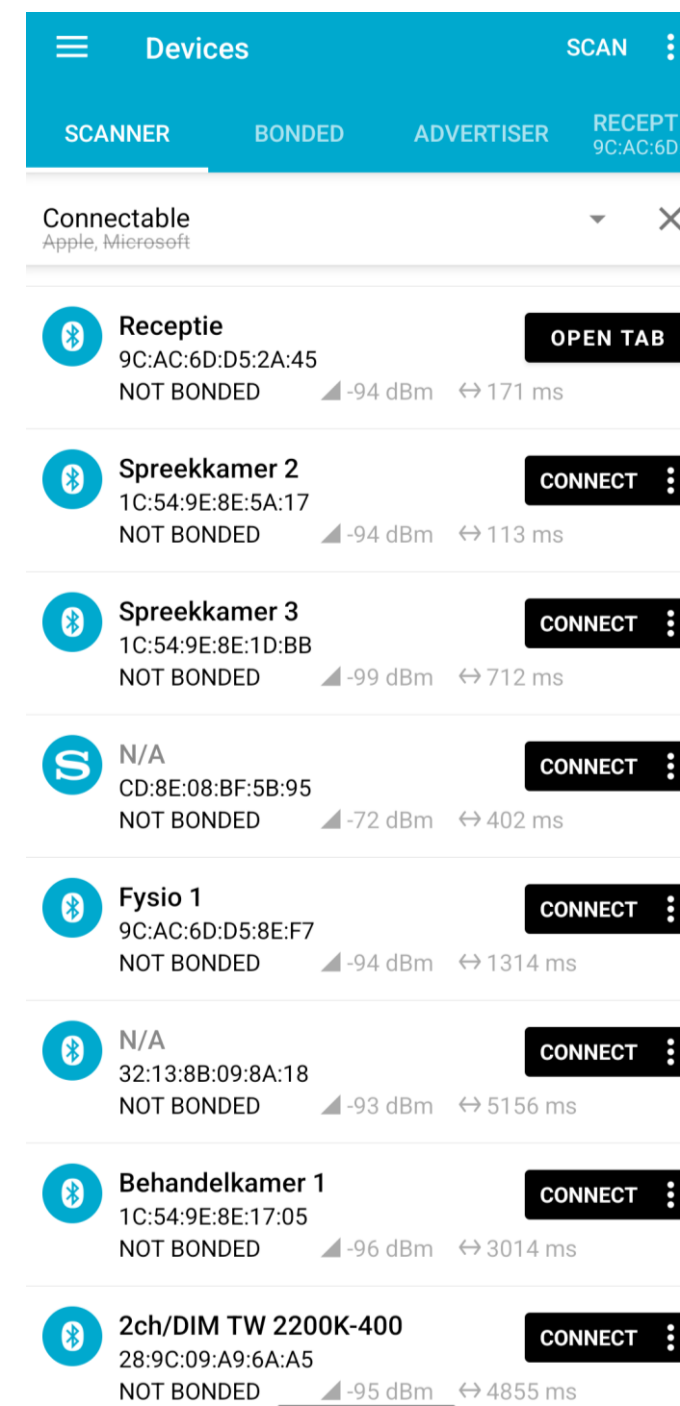
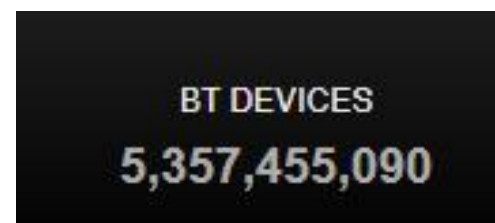
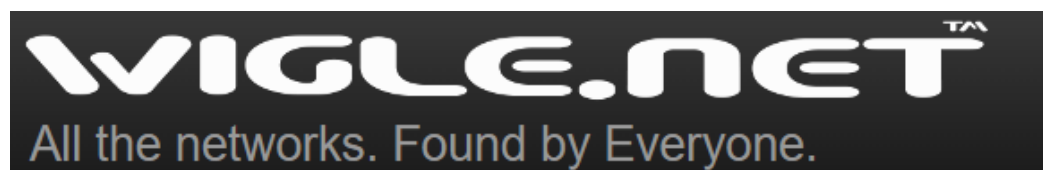
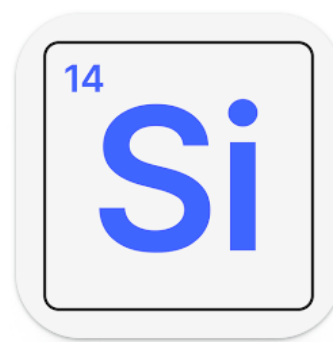
nRF Connect for Mobile

Nordic Semiconductor ASA



Simplicity Connect

Silicon Laboratories



Using a BLE sniffer, Wireshark?



Capturing from nRF Sniffer: /dev/ttyUSB0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-F>

No.	Time	Source	Destination	Protocol	Length	Info
282	79.066089	d1:7c:65:9a:9a:a5	Broadcast	LE LL	55	ADV_IND
283	79.267289	Blueradi_13:9f:95	Broadcast	LE LL	57	ADV_IND
284	79.468741	TexasIns_c3:a8:1e	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
285	80.072562	d1:7c:65:9a:9a:a5	Broadcast	LE LL	55	ADV_IND
286	80.273961	Blueradi_13:9f:95	Broadcast	LE LL	57	ADV_IND[Malformed Packet]
287	80.475318	TexasIns_c3:a8:1e	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
288	80.676430	f8:c7:7f:1e:2e:8b	Broadcast	LE LL	39	ADV_IND[Malformed Packet]
289	80.777508	Blueradi_13:5f:95	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
290	80.980162	TexasIns_c3:a8:1e	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
291	81.081553	d1:7c:65:9a:9a:a5	Broadcast	LE LL	55	ADV_IND

Frame 183: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0

Nordic BLE Sniffer

Bluetooth Low Energy Link Layer

Access Address: 0x8e89bed6

Packet Header: 0x1d40 (PDU Type: ADV_IND, ChSel: #1, TxAdd: Random)

Advertising Address: d1:7c:65:9a:9a:a5 (d1:7c:65:9a:9a:a5)

Advertising Data

Flags

Device Name (shortened): smartlockpicking01

```

0000 fc 06 30 01 4a 19 06 0a 01 27 38 00 00 93 02 00 ..0.J... '8....
0010 90 d6 be 89 8e 40 1d a5 9a 9a 65 7c d1 02 01 04 ....@... |e|....
0020 13 08 73 6d 61 72 74 6c 6f 63 6b 70 69 63 6b 69 ..smartl ockpicki
0030 6e 67 30 31 88 65 14 ng01.e.
  
```

Pentesting? Trying to re-create known attacks?

- Scientific research PoC usually not very “user friendly”
- Custom boards, ridiculous stacks, ...

BLERP: BLE Re-Pairing Attacks and Defenses



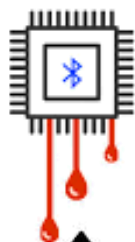
BLE Spoofing
Attack BLESA



BlueBorne™

BLE CHIP RCE Vulnerabilities

Bleeding Bit



WEYN↑OOOTH



KNOB Attack

Key Negotiation of Bluetooth Attack: Breaking Bluetooth Security

ANDY GREENBERG

LILY HAY NEWMAN

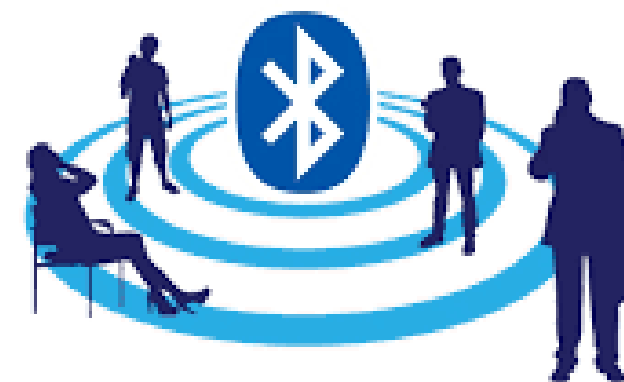
SECURITY

JAN 15, 2026 7:00 AM

Hundreds of Millions of Audio Devices Need a Patch to Prevent Wireless Hacking and Tracking

Flaws in how 17 models of headphones and speakers use Google’s one-tap Fast Pair Bluetooth protocol have left devices open to eavesdroppers and stalkers.

<https://www.wired.com/story/google-fast-pair-bluetooth-audio-accessories-vulnerability-patches/>



Critical Bluetooth BIAS Attack
Let Hackers Access Billions of Devices

SŁAWOMIR (SŁAWEK) JASEK

<suavomeer>

<suaveck>

<yaseck>

#HCE

#NFC

#KNX

#SDR

#BLE

#embedded

#RFID

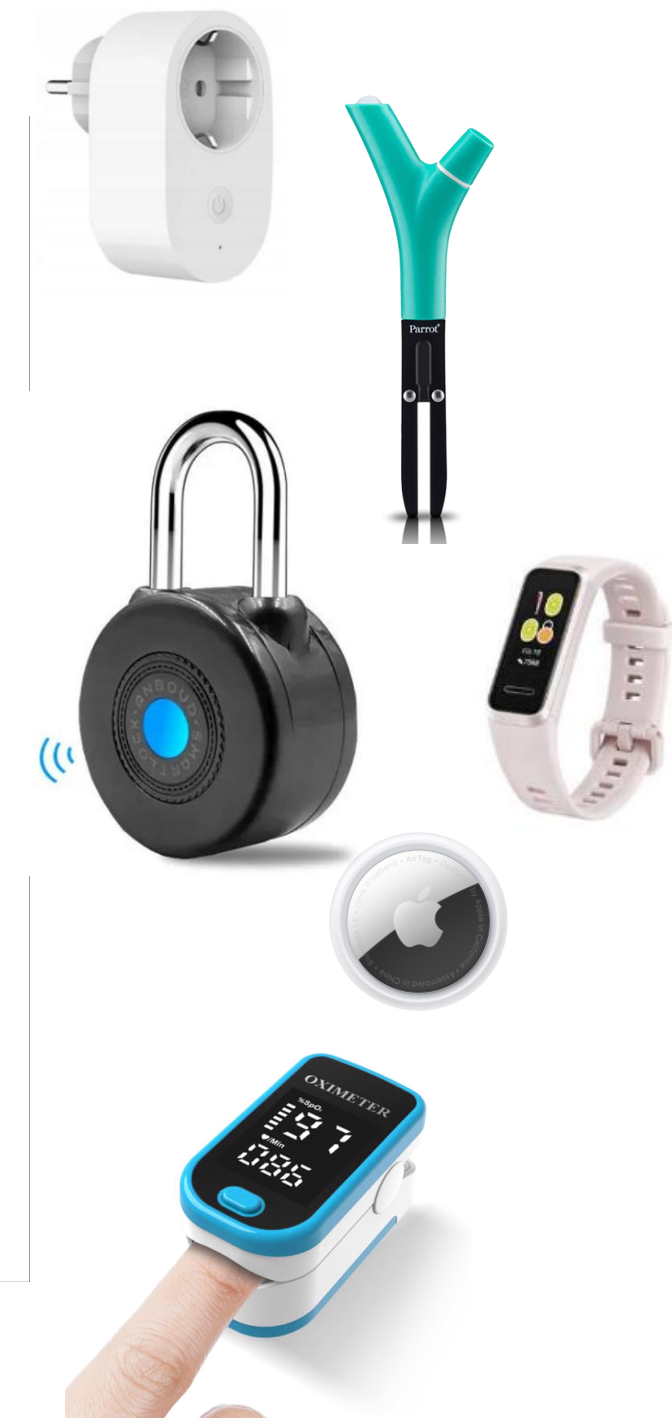
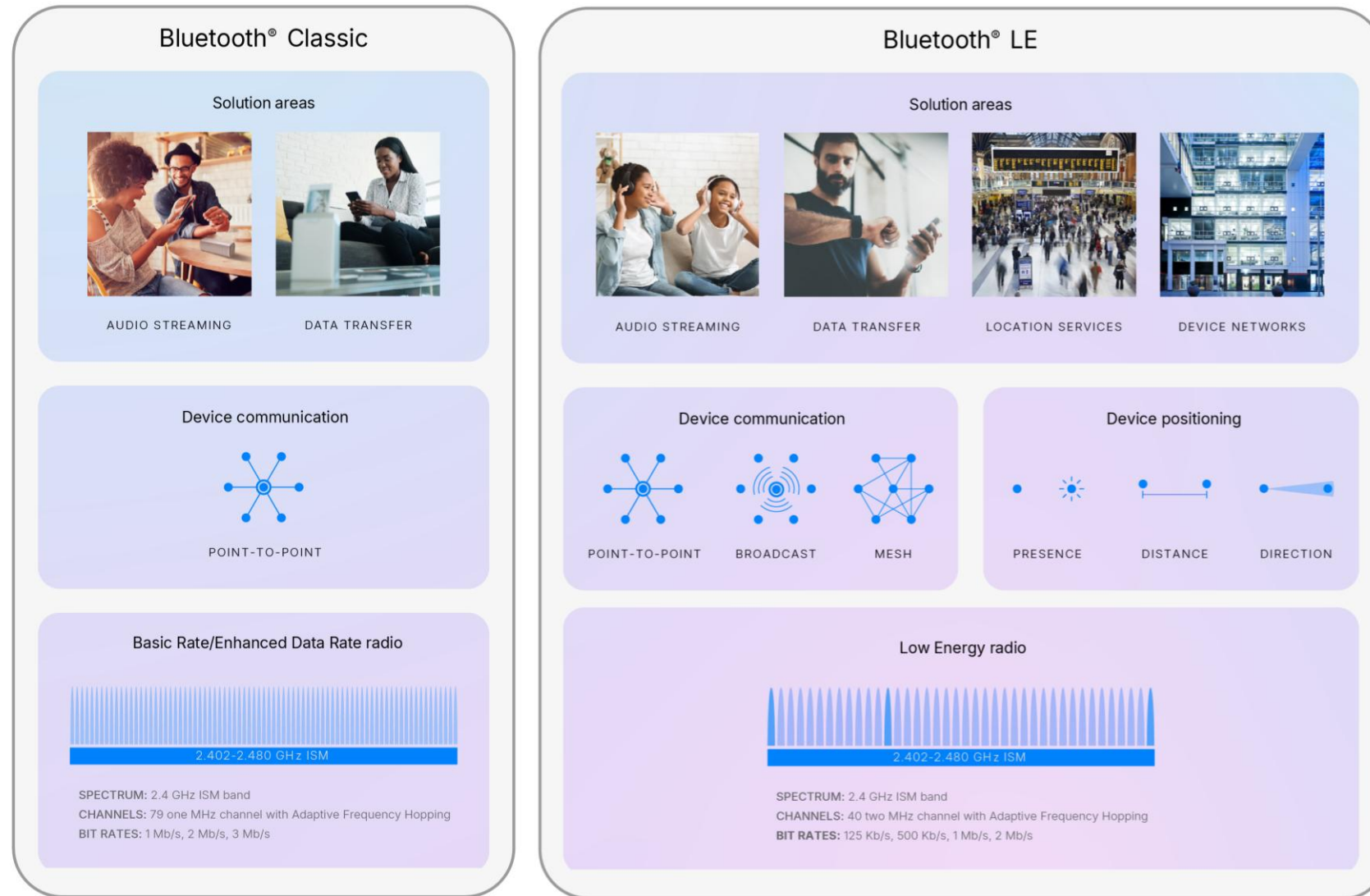
#IoT



DEEPSEC



Bluetooth "Classic" vs Low Energy



<https://www.bluetooth.com/learn-about-bluetooth/tech-overview/>

Moving towards the BLE

5.9 billion

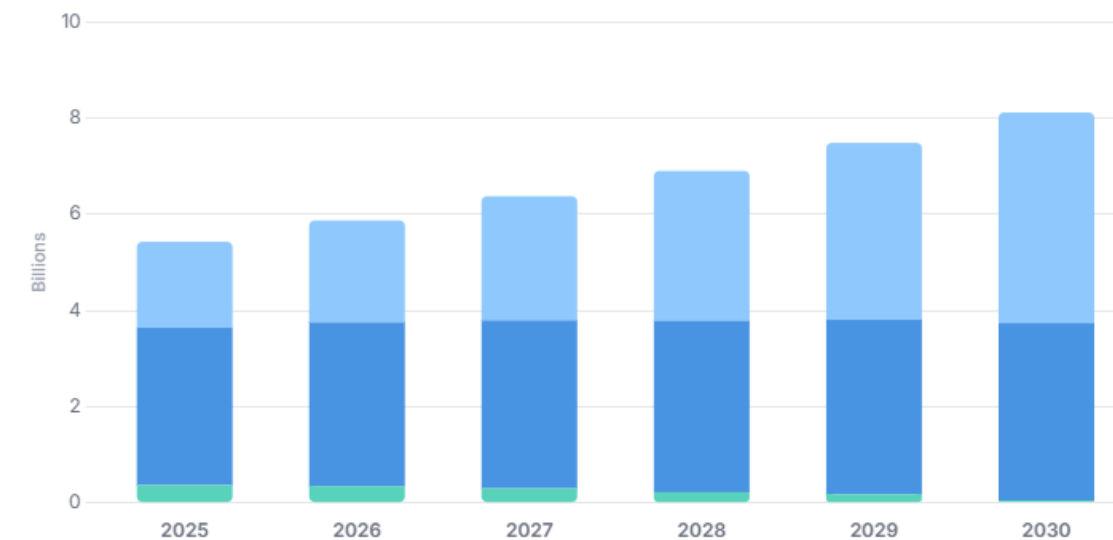
annual shipments of
Bluetooth® enabled devices
in 2026

Total shipments

By radio

Bluetooth® device shipments reflect the evolving mix of radio implementations across the ecosystem. Bluetooth dual-mode radios remain the dominant configuration, particularly in platform devices such as smartphones, tablets, PCs, and laptops.

Total annual Bluetooth® device shipments



Bluetooth® Classic (~33% CAGR) Bluetooth® Dual-Mode (2% CAGR)
Bluetooth® LE (20% CAGR)

Key data

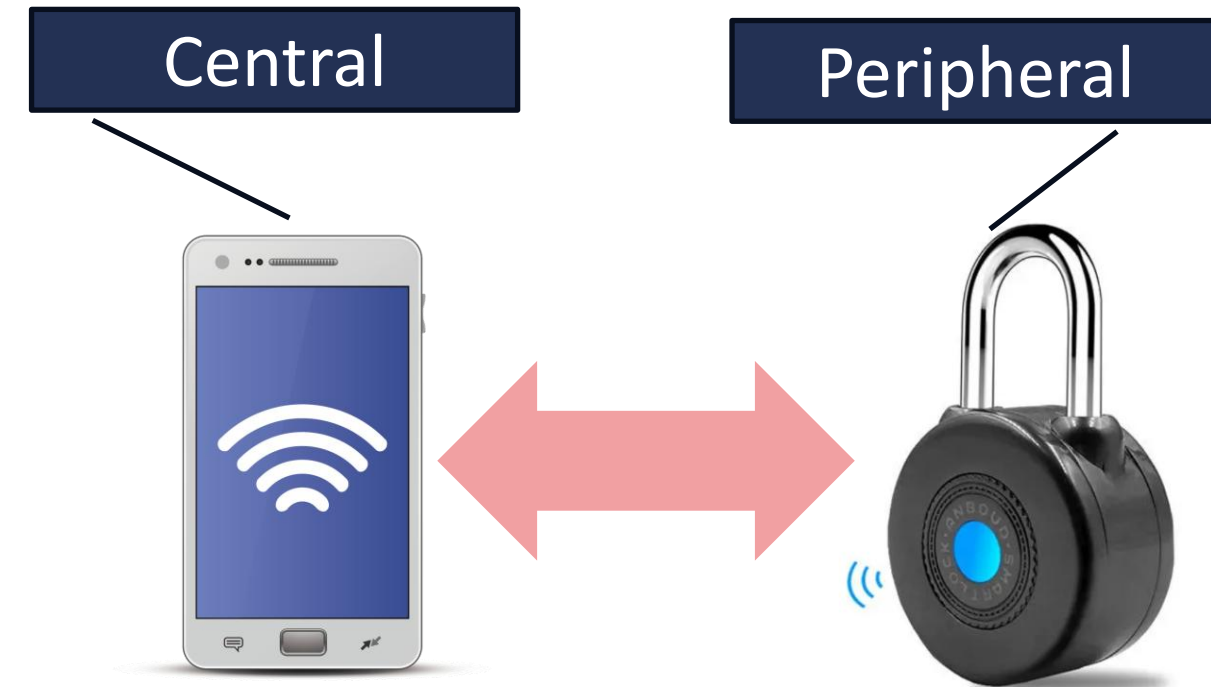
99%

of all Bluetooth® devices shipped in 2030 will include Bluetooth® LE

20%

CAGR for Bluetooth® LE devices (2025–2030)

BLE operation modes



Two-way communication
(e.g. app controls smart lock).
Starts with broadcast.



BLESPloit



Mobile app

- Rich UI, complex parsing/fingerprinting, ...
- Uses phone's BLE capabilities

Device library with dynamic scripts

ESP32 “companion”

- Connect via WiFi or USB-C
- Scan/connect remotely
- Get more low-level details on packets
- Simulate any device (optionally with display/buttons)
- Fully control the pairing parameters



BLESPloit mobile application



Android: <https://play.google.com/store/apps/details?id=it.blesplo.app>

iOS: <https://apps.apple.com/us/app/blesploit/id6787705285>

BLE requires location/nearby permissions, but no location data is saved nor shared!

Policies > Mobile Application Privacy Policy

Mobile Application Privacy Policy

Effective date: May 25, 2026

Version: 1.1

Summary: This app does not collect any data. BLE scan results are optionally saved by the user, locally on their own device only — nothing is transmitted to us or anyone else.

<https://blesplo.it/policies/app-privacy-policy/>

Permissions needed

You will be asked to grant the permissions. You can change them later in system settings.

These permissions are required for local Bluetooth scanning and related features here. The application DOES NOT use this for other purposes, and does not track your location. You can still use Remote mode and connect to an ESP32 without them.

Nearby devices — required for Bluetooth scanning, connections, and related features.

Location — required by Android so the system allows Bluetooth Low Energy scanning.

If you land on App info, tap Permissions next.

[Privacy Policy](#)

[Cancel](#)

[Continue](#)

Local scan

Actions from the phone

LOCAL

REMOTE

RELAY

Filter

Scan

-66

★



Triones-FFFF3859A91F
Happy Light: Triones
FF:FF:38:59:A9:1F



-87

★



Unknown
72:55:86:9B:E3:1C



-40

★



Rapidpower-dog-fire
Rapidpower robotic dog
9A:CE:45:BF:A2:99



-69

★



Flipper Oltot
White
80:E1:26:47:F7:83



-98

★



Unknown
18:18:EB:D8:8E:E4



-74

★



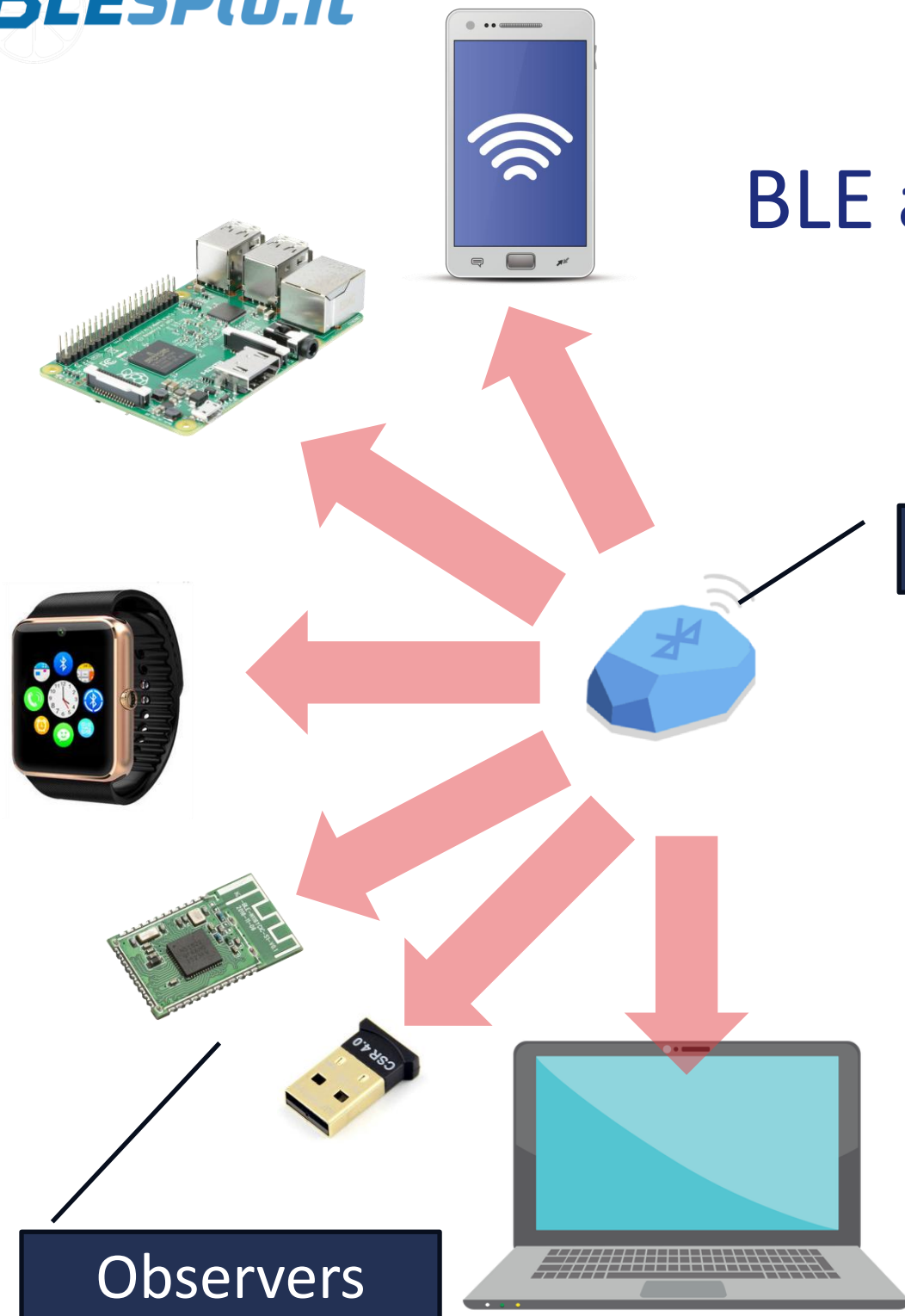
Sony
75:3E:46:AC:A0:19



Optional connection with ESP32

Scan results

BLE advertising



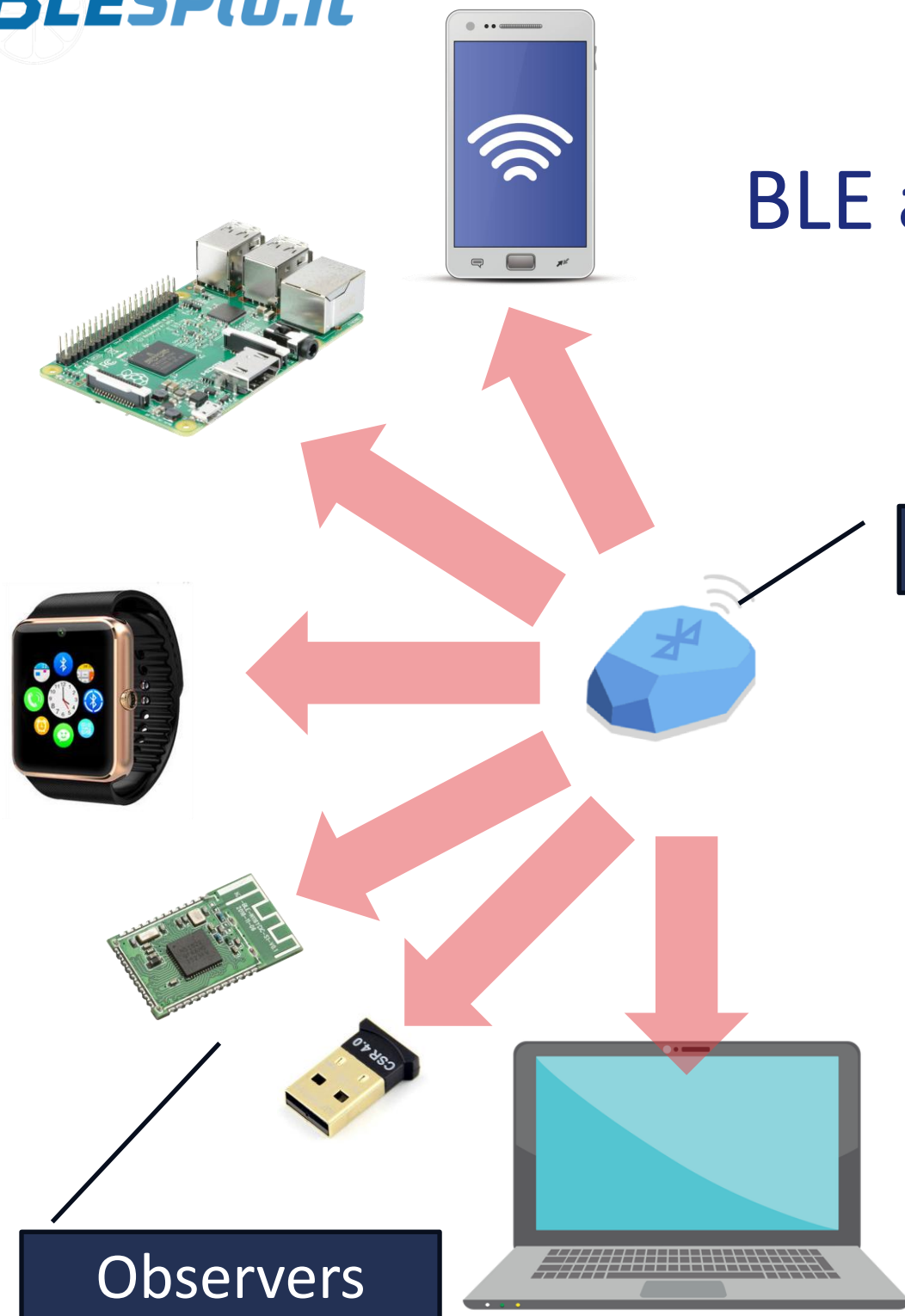
Tap for more info

SmartLockPicking app interface showing a list of discovered BLE devices. The interface includes a menu icon, tabs for LOCAL, REMOTE, and RELAY, a Filter dropdown, and a Scan button. The list of devices is as follows:

Signal Strength	Icon	Device Name	MAC Address	Actions
-56	Windows logo	Windows Desktop	3E:36:43:3F:96:55	
-59	Eddystone logo	Eddystone BMO	24:EC:4A:30:79:6A	
-54	Apple logo	Nearby Info	66:F3:77:E0:68:25	Copy, Link
-52	Tesla logo	S4c180378021e8570C	S4c180378021e8570C 70:04:1D:DA:73:4A	Copy, Link
-63	Lightbulb icon	Triones-FFFF3859A91F	Happy Light: Triones FF:FF:38:59:A9:1F	Copy, Link
-89	iBeacon logo	iBeacon	50:2E:11:69:3E:1E	
-80	Google logo	Unknown		

A line points from the "Tap for more info" text to the Apple logo in the "Nearby Info" entry.

BLE advertising



-39 **Nearby Info**
Idle (screen off/locked)
6C:6A:4D:27:9D:68

✓ [Capture \(this session\)](#) · no svc

Nearby · Adv ~279ms · 0s ago · Not bonded

Raw	Parsed	Decoded
Source: Apple Nearby Info (apple_nearby_info) · 30		
NearbyInfo authtag_4b: no protocol: NearbyInfo action: Idle (screen off/locked) authtag_pres: yes airdrop_rcv: no beaconformat: NearbyInfo airpods_conn: no auth_tag: 301091 action_code_hex: 0x03 primary_device: no		

Filter LOCAL REMOTE RELAY Scan

- 56** **Windows Desktop**
3E:36:43:3F:96:55
- 59** **Eddystone BMO**
24:EC:4A:30:79:6A
- 54** **Nearby Info**
66:F3:77:E0:68:25
- 52** **S4c180378021e8570C**
S4c180378021e8570C
70:04:1D:DA:73:4A
- 63** **Triones-FFFF3859A91F**
Happy Light: Triones
FF:FF:38:59:A9:1F
- 89** **iBeacon**
50:2E:11:69:3E:1E
- 80** **Unknown**

Advertising packets

Short packets in Len Type Value format.

Sample packet broadcasted by iPhone:

02011a020a0c0aff4c001005231c301091

-39

☆



Nearby Info

Idle (screen off/locked)

6C:6A:4D:27:9D:68

📄

🔗

▼

Capture (this session) · no svc

Nearby · Adv ~272ms · 0s ago · Not bonded

Raw

Parsed

Decoded

ADV data

02011a020a0c0aff4c001005231c301091

LEN	TYPE	VALUE
0x02 (2)	0x01	1A
0x02 (2)	0x0A	0C
0x0A (10)	0xFF	4C001005231C301091

Advertising packets

Short packets in Len Type Value format.

Sample packet broadcasted by iPhone:

02011a020a0c0aff4c001005231c301091

Length: 02 Type: 01 Value: 1a

-50

☆



Nearby Info

Idle (screen off/locked)

56:4E:65:92:CF:82

✓

Capture (this session) · no svc

Nearby · Adv ~276ms · 0s ago · Not bonded

Raw

Parsed

Decoded

ADV data

02011a020a0c0aff4c001005231cc740f9

LEN	TYPE	VALUE
0x02 (2)	0x01	1A
0x02 (2)	0x0A	0C
0x0A (10)	0xFF	4C001005231CC740F9

Advertising packets

Short packets in Len Type Value format.

Sample packet broadcasted by iPhone:

02011a020a0c0aff4c001005231c301091

Length: 02 Type: 0a Value: 0c

-47



Nearby Info

Idle (screen off/locked)

56:4E:65:92:CF:82






Capture (this session) · no svc

Nearby · Adv ~276ms · 0s ago · Not bonded

Raw

Parsed

Decoded

ADV data

02011a020a0c0aff4c001005231cc740f9

LEN	TYPE	VALUE
0x02 (2)	0x01	1A
0x02 (2)	0x0A	0C
0x0A (10)	0xFF	4C001005231CC740F9

Advertising packets

Short packets in Len Type Value format.

Sample packet broadcasted by iPhone:

02011a020a0c0aff4c001005231c301091

Length: 0a Type: ff Value: 4c0010...

-47

☆



Nearby Info

Idle (screen off/locked)

56:4E:65:92:CF:82

▼

Capture (this session) · no svc

Nearby · Adv ~284ms · 0s ago · Not bonded

Raw

Parsed

Decoded

ADV data

02011a020a0c0aff4c001005231cc740f9

LEN	TYPE	VALUE
0x02 (2)	0x01	1A
0x02 (2)	0x0A	0C
0x0A (10)	0xFF	4C001005231CC740F9

Type: Bluetooth specification

SPECIFICATIONS

Assigned Numbers

Documents	Description
Assigned Numbers Document (PDF)	The Assigned Numbers Document contains all assigned number values.
Assigned Numbers Repository (YAML)	The Assigned Numbers are available in YAML format in a publicly available repository.

<https://www.bluetooth.com/specifications/assigned-numbers/>

Advertising types

Defined in Bluetooth specification.

0x01 = Flags

0x0A = TX Power Level

...

0xFF = “Manufacturer specific”

2.3 Common Data Types

Referenced from the following:

- Supplement to the Bluetooth Core Specification Part A, Section 1 [22].
- Bluetooth Core Specification [Vol 3] Part C, Section 8 [4].
- Bluetooth Core Specification [Vol 3] Part C, Section 11 [4].
- Bluetooth Core Specification [Vol 6] Part B, Section 2.3.4.8 [4].

Last Modified: 2023-08-08

Filename: [assigned_numbers/core/ad_types.yaml](#)

Common Data Type	Name	Reference
0x01	Flags	Core Specification Supplement, Part A, Section 1.3
0x02	Incomplete List of 16-bit Service Class UUIDs	Core Specification Supplement, Part A, Section 1.1
0x03	Complete List of 16-bit Service Class UUIDs	Core Specification Supplement, Part A, Section 1.1
0x04	Incomplete List of 32-bit Service Class UUIDs	Core Specification Supplement, Part A, Section 1.1
0x05	Complete List of 32-bit Service Class UUIDs	Core Specification Supplement, Part A, Section 1.1
0x06	Incomplete List of 128-bit Service Class UUIDs	Core Specification Supplement, Part A, Section 1.1
0x07	Complete List of 128-bit Service Class UUIDs	Core Specification Supplement, Part A, Section 1.1
0x08	Shortened Local Name	Core Specification Supplement, Part A, Section 1.2
0x09	Complete Local Name	Core Specification Supplement, Part A, Section 1.2
0x0A	Tx Power Level	Core Specification Supplement, Part A, Section 1.5

Manufacturer specific data: Bluetooth specification

Supplement to Bluetooth Core Specification | CSS v9, Part A

page 13

Data Types Specification



1.4 MANUFACTURER SPECIFIC DATA

1.4.1 Description

The Manufacturer Specific data type is used for manufacturer specific data. The first two data octets shall contain a company identifier from [Assigned Numbers](#). The interpretation of any other octets within the data shall be defined by the manufacturer specified by the company identifier.

1.4.2 Format

Data Type	Description
«Manufacturer Specific Data»	Size: 2 or more octets The first 2 octets contain the Company Identifier Code followed by additional manufacturer specific data

Table 1.5: Manufacturer Specific data type

<https://www.bluetooth.com/specifications/specs/core-specification-supplement-9/>

Advertising packets

Short packets in Len Type Value format.

Sample packet broadcasted by iPhone:

02011a020a0c0aff4c001005231c301091

Length: 0a Type: ff Value: [Apple]1005...

-47

☆



Nearby Info

Idle (screen off/locked)

56:4E:65:92:CF:82

▼

Capture (this session) · no svc

Nearby · Adv ~284ms · 0s ago · Not bonded

Raw

Parsed

Decoded

ADV data

02011a020a0c0aff4c001005231cc740f9

LEN	TYPE	VALUE
0x02 (2)	0x01	1A
0x02 (2)	0x0A	0C
0x0A (10)	0xFF	4C001005231CC740F9

0xFF	Manufacturer Specific Data	Core Specification Supplement, Part A, Section 1.4
------	----------------------------	--

Parsing

-47



Nearby Info
Idle (screen off/locked)
56:4E:65:92:CF:82



✓ Capture (this session) · no svc

Nearby · Adv ~284ms · 0s ago · Not bonded

Raw

Parsed

Decoded

ADV data
02011a020a0c0aff4c001005231cc740f9

LEN	TYPE	VALUE
0x02 (2)	0x01	1A
0x02 (2)	0x0A	0C
0x0A (10)	0xFF	4C001005231CC740F9

-40



Nearby Info
Idle (screen off/locked)
56:4E:65:92:CF:82



✓ Capture (this session) · no svc

Nearby · Adv ~274ms · 0s ago · Not bonded

Raw

Parsed

Decoded

0x01 Flags
0x1A: LE General Discoverable Mode, Simultaneous LE and BR/EDR to Same Device Capable (Controller), Simultaneous LE and BR/EDR to Same Device Capable (Host)

0x0A TX Power Level
12 dBm

0xFF Manufacturer Specific Data
Company Code: 0x004C (Apple, Inc.)
Data: 0x1005231CC740F9

BLESPloit: packet decoding

After parsing, packet can be optionally decoded by matching dynamic scripts



-36

☆



Nearby Info
 Idle (screen off/locked)
 6C:6A:4D:27:9D:68





Capture (this session) · no svc

Nearby · Adv ~277ms · 0s ago · Not bonded

Raw

Parsed

Decoded

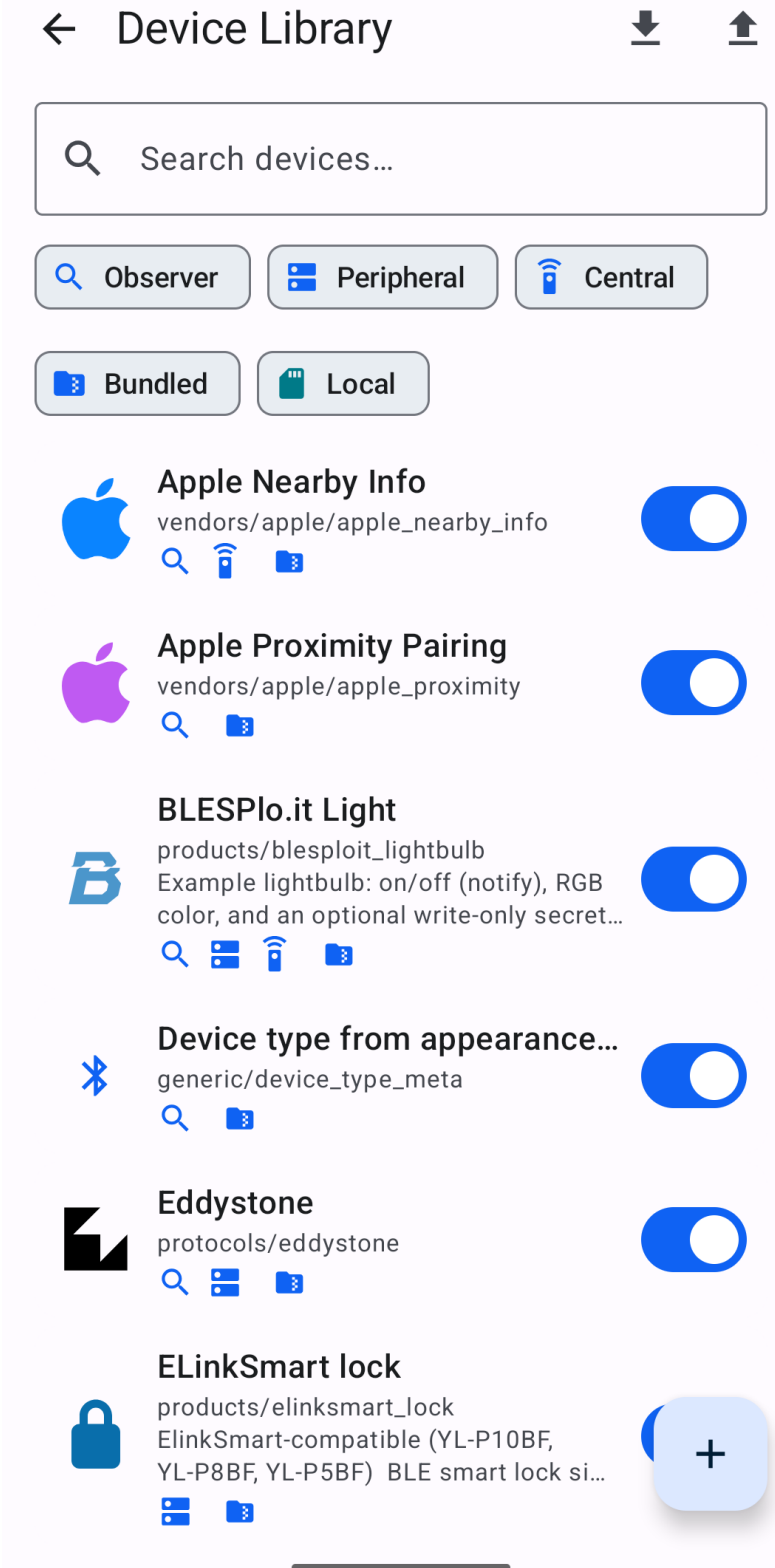

[Source: Apple Nearby Info \(apple_nearby_info\) · 30](#)

NearbyInfo
 authtag_4b: no
 protocol: NearbyInfo
 action: Idle (screen off/locked)
 authtag_pres: yes
 airdrop_recv: no
 beaconformat: NearbyInfo
 airpods_conn: no
 auth_tag: 301091
 action_code_hex: 0x03
 primary_device: no

Dynamic scripts

Scripts in “Device Library” can match specific devices based on packet content (name, vendor, payload, structure, ...).

Possible to enable/disable and edit, create own.



Apple dynamic scripts

< Device library

Search devices

Observer

Peripheral

Central

Bundled

Local

Apple Continuity (dispatcher)
vendors/apple/apple_dispatcher

Apple Find My
vendors/apple/apple_findmy

Apple Handoff
vendors/apple/apple_handoff

Apple Nearby Action
vendors/apple/apple_nearby_action

Apple Nearby Info
vendors/apple/apple_nearby_info

Apple Proximity Pairing
vendors/apple/apple_proximity

0xFF Manufacturer Specific Data

Company Code: 0x004C (Apple, Inc.)

Data: 0x1005231CC740F9

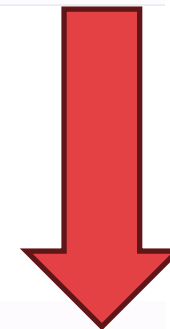
Priority: 20



Apple Continuity
(dispatcher)

vendors/apple/apple_dispatcher
v2 bundled

02 (iBeacon)
4c001005231c301091
12 (FindMy)



Priority: 30



Apple Nearby Info

vendors/apple/apple_nearby_info
v1 bundled

4c001005231c301091

Apple nearby decoding

-36



Nearby Info
Idle (screen off/locked)
6C:6A:4D:27:9D:68



☆

✓ Capture (this session) · no svc

Nearby · Adv ~277ms · 0s ago · Not bonded

Raw

Parsed

Decoded

^

[Source: Apple Nearby Info \(apple_nearby_info\) · 30](#)

NearbyInfo
authtag_4b: no
protocol: NearbyInfo
action: Idle (screen off/locked)
authtag_pres: yes
airdrop_recv: no
beaconformat: NearbyInfo
airpods_conn: no
auth_tag: 301091
action_code_hex: 0x03
primary_device: no



Nearby Info (+1)

Active user (screen on)

6C:6A:4D:27:9D:68



Nearby Info (+2)

Recent interaction

6C:6A:4D:27:9D:68

JSON

Simple definitions (pattern matching, ...)

Organize structure (manifest, roles: observer, peripheral, central).

Optional script entry.

Priority: lower runs first (can be chained).

```
{
  "name": "Apple Continuity (dispatcher)",
  "assets": {
    "icon": "assets/apple_gradient.svg",
    "icon_tint": "#8E8E93"
  },
  "roles": {
    "observer": {
      "scan_conditions": {
        "company_id": "004C"
      },
      "entry": "observer/adv_decode.lua",
      "priority": 20
    }
  },
  "manifest_version": 1,
  "version": 2
}
```

Dynamic scripts: LUA

Should work on any platform (mobile, embedded, native, ...)

Easy to script, short, simple to understand.

Don't know Lua? Current programming language is English anyway...

```
local ACTION_LABELS = {
    [0x01] = "Activity reporting disabled",
    [0x03] = "Idle (screen off/locked)",
    [0x05] = "Audio playing, screen locked",
    [0x07] = "Active user (screen on)",
    [0x09] = "Screen on, video playing",
    [0x0A] = "Watch on wrist, unlocked",
    [0x0B] = "Recent interaction",
    [0x0D] = "Driving",
    [0x0E] = "Phone/FaceTime call",
}

local function decode_nearby_info(data, offset,
len)
    local byte0 = data[offset + 1] or 0
    local byte1 = data[offset + 2] or 0

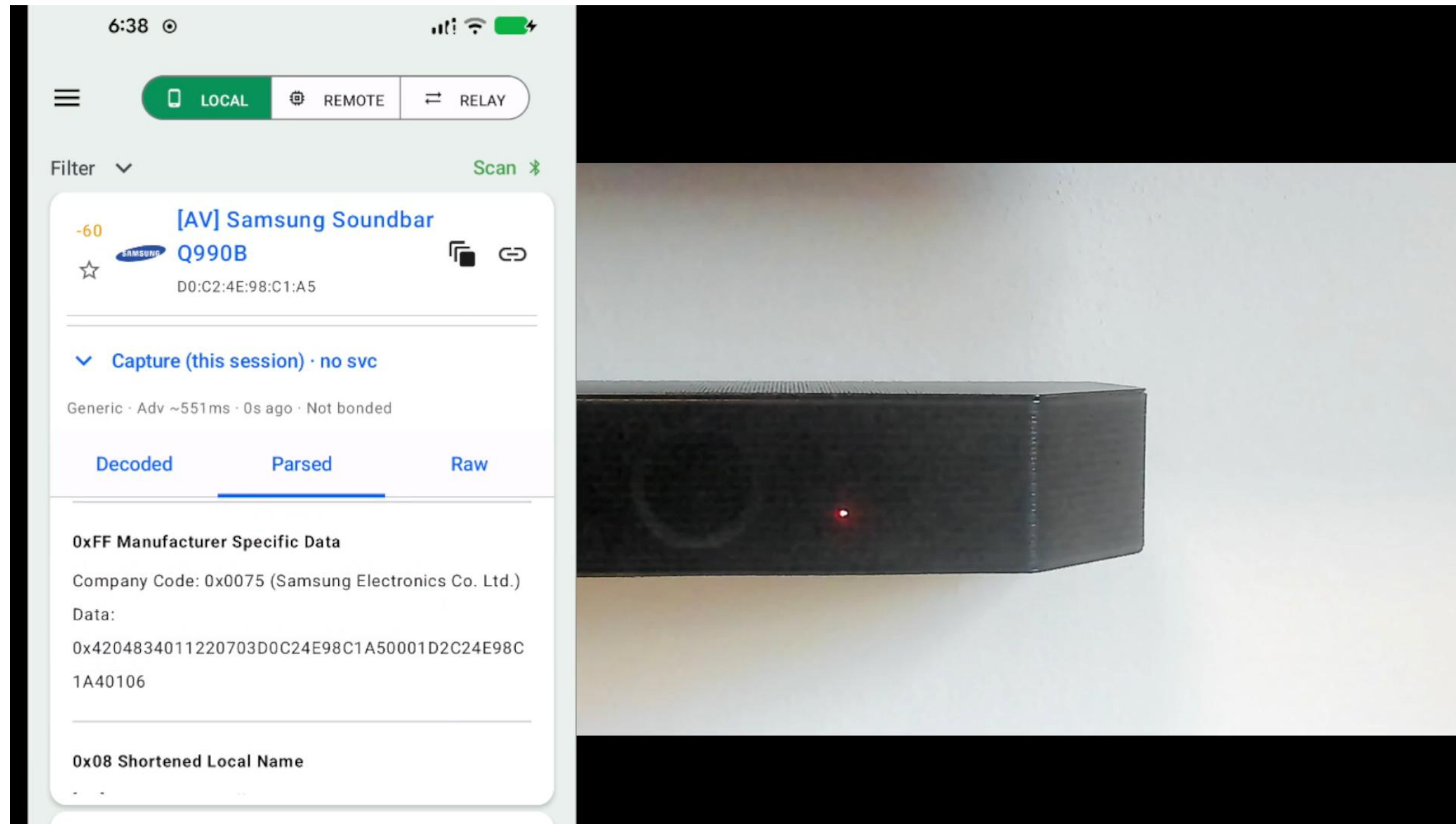
    local status_flags =
bits.band(bits.rshift(byte0, 4), 0x0F)
    local action_code = bits.band(byte0, 0x0F)

    local wifi_on = bits.band(byte1, 0x04) ~= 0
    local authtag_4b = bits.band(byte1, 0x02) ~= 0
    local authtag_pres = bits.band(byte1, 0x10) ~=
```



ADDING NEW DEVICE

Simple observation: Samsung soundbar



A small change in advertising payload ON vs OFF

-64

[AV] Samsung Soundbar Q990B (+1)

D0:C2:4E:98:C1:A5

▼

Capture (this session) · no svc

adv1

adv2

Generic · Adv ~401ms · 0s ago · Not bonded

Decoded

Parsed

Raw

0xFF Manufacturer Specific Data

Company Code: 0x0075 (Samsung Electronics Co. Ltd.)

Data:

0x4204834011220703D0C24E98C1A50001D2C24E98C1A40106

0x08 Shortened Local Name

0x4204834011

-64

[AV] Samsung Soundbar Q990B (+1)

D0:C2:4E:98:C1:A5

▼

Capture (this session) · no svc

adv1

adv2

Generic · Adv ~377ms · 0s ago · Not bonded

Decoded

Parsed

Raw

0xFF Manufacturer Specific Data

Company Code: 0x0075 (Samsung Electronics Co. Ltd.)

Data:

0x4204830111220703D0C24E98C1A50001D2C24E98C1A40106

0x08 Shortened Local Name

0x4204830111

Capture History

-94

[AV] Samsung Soundbar Q990B (+1)

D0:C2:4E:98:C1:A5

☆

SAMSUNG

View capture files

^ Capture (this session)

Advertising profiles saved: 2

Services file (ble.json): No

Connection logs: 0

adv1

adv2

Generic · Adv ~2468ms · 0s ago · Not bonded

Decoded

Parsed

Raw

0x01 Flags

0x18: Simultaneous LE and BR/EDR to Same Device Capable (Controller), Simultaneous LE and BR/EDR to Same Device Capable (Host)

Stop

Manufacturer Specific Data

Manufacturer: 0x0075 (Samsung Electronics Co. Ltd.)

Scan

Graph

Active connections

← [AV] Samsung Sound... End session

[AV] Samsung Soundbar Q990B
D0:C2:4E:98:C1:A5
Source: Local

Create Device Library entry

Files in this capture

adv.json
1.05 kB

meta.json
351 B

View raw

View raw

← adv.json

```
{
  "profiles": [
    {
      "id": "adv1",
      "enabled": true,
      "legacy_pdu": true,
      "connectable": true,
      "scannable": true,
      "channel_map": 7,
      "primary_phy": "1M",
      "secondary_phy": "1M",
      "adv_interval_ms": 250,
      "adv_interval_ms_observed": 2427,
      "addr_type": "random_specific",
      "bd_addr": "D0:C2:4E:98:C1:A5",
      "adv_data_hex_combined":
"0201181bff75004204830111220703d0c24e98c1a50001d2c24e98c1a401061c085b41565d2053616d73756e6720536f756e64626172205139393042"
    },
    {
      "id": "adv2",
      "enabled": false,
      "legacy_pdu": true,
      "connectable": true,
      "scannable": true,
      "channel_map": 7,
      "primary_phy": "1M",
      "secondary_phy": "1M",
      "adv_interval_ms": 250,
      "adv_interval_ms_observed": 2427,
      "addr_type": "random_specific",
      "bd_addr": "D0:C2:4E:98:C1:A5",
      "adv_data_hex_combined":
"0201181bff75004204834011220703d0c24e98c1a50001d2c24e98c1a401061c085b41565d2053616d73756e6720536f756e64626172205139393042"
    }
  ]
}
```

Device Library reference

Device folder layout

```
my-device/
  manifest.json      ← identity, role declarations, asset paths
  ble.json           ← GATT services, characteristics, descriptors, pairing
  vars.json          ← runtime variables (strings, numbers, bools)
  uuids.json         ← symbolic name → UUID map
  assets/
    icon.svg
    other_graphic.png
    graphics.json
  observer/
    adv_decode.lua   ← Lua script executed during mobile BLE scan
  central/
    dynamic.lua      ← Lua script executed on the mobile GATT client
    menu.json        ← GATT-client menu tree with Lua snippet hooks
  peripheral/
    adv.json         ← advertising profiles (raw hex, scan response)
    interface.json   ← peripheral display / UI simulation bindings
    dynamic.lua      ← Lua script executed on the ESP32 simulator
```

Reference

File / topic	
<code>manifest.json</code> — identity, roles, asset paths	Manifest
<code>ble.json</code> (GATT services and characteristics), <code>adv.json</code> (advertising data), <code>interface.json</code> (physical buttons)	Peripheral
<code>uuids.json</code> (proprietary service/characteristic definitions), <code>vars.json</code> (optional variables), <code>assets/graphics.json</code> (graphic elements)	optional json
<code>menu.json</code> — central menu tree	central

Lua scripting

Lua runs in three separate contexts. No globals are shared between them.

Context	Runs on	Reference
Peripheral	ESP32 firmware	ESP32 peripheral LUA
Central	Mobile (phone)	mobile central LUA
Observer	Mobile (phone)	mobile observer LUA

<https://blesplo.it/docs/device-library/>

SKILL.md for LLM

name	blesploit-device-library-entry
description	Creates BLESPloit device library entry (manifest.json, observer/central/peripheral Lua, uuids, assets). Use when adding a device, creating a library entry, writing a Lua adv decoder, or generating a manifest for a BLE device.

BLESPloit device library entry

Best results with the `device-library` (<https://github.com/blesploit/device-library>) repo open in the workspace.

Reference entry: `products/blesploit_lightbulb/` (all three roles, full asset layout).

Attach for full detail: `docs/device-manifest.schema.json` , `docs/lua-mobile.md` ,
`docs/lua-esp32.md` (peripheral runs on ESP32).

Full schema: <https://blesplo.it/docs/device-library>

Workflow

1. Classify entry: `products/<id>/` (specific product), `vendors/<id>/` (vendor-wide), `protocols/<id>/` (cross-vendor), `generic/<id>/` (fallback).
2. Derive `scan_conditions` from advertisement captures; tighten with AND keys before adding Lua.
3. Write `manifest.json` (`manifest_version: 1` , `version: 1` on new entries).
4. Add role scripts only for requested capabilities. Omit unused roles entirely.
5. Store reference captures in `sample_scans/` (not loaded at runtime).
6. Validate JSON against schema; match naming/style of nearest existing entry.

Entry folder id (e.g. `products/blesploit_lightbulb`) is the runtime device id. All manifest paths are relative to that folder.

<https://github.com/BLESPloit/device-library/blob/main/skills/blesploit-device-library-entry/SKILL.md>

Write decoder script

The `samsung_soundbar_q990b_adv.json` has two profiles: `adv1 (ON)` and `adv2(OFF)`. Write a simple observer script that will decode the soundbar state.

Creating the observer script and minimal device structure. ON/OFF is encoded in Samsung manufacturer data byte 3: `0x40` (ON) vs `0x01` (OFF), from the two captures.

Explored 2 files

It reads Samsung manufacturer data (company ID `0x0075` / key `"117"`) and expects this layout:

Bytes	Meaning
<code>42 04 83</code>	Fixed header
Next byte	Power state: <code>0x01</code> = on (adv1), <code>0x40</code> = off / standby (adv2)
Following 6 bytes	Embedded MAC (<code>D0:C2:4E:98:C1:A5</code>)

20 seconds later...

Decoded output includes:

- `state` — "on" or "off"
- `state_label` — "On" or "Off (standby)"
- `mac` — formatted BLE address from the payload
- `parsed_protocol` — e.g. `state=on mac=D0:C2:4E:98:C1:A5`
- Scan-row `display_info` — the state label

To hook it up in the app, add a `manifest.json` observer role pointing at `observer/adv_decode.lua`, with scan conditions such as:

```
"scan_conditions": {
  "device_name_contains": "Samsung Soundbar Q990B",
  "company_id": "0075",
  "manufacturer_data_prefix_hex": "420483"
}
```

You'll also want `assets/samsung.svg` (or another icon) under the product folder if you use `custom_icon`. I can add the manifest and icon next if you want.

```
-- Samsung Soundbar Q990B: Samsung manufacturer data (company 0x0075) payload
-- 42 04 83 <state> ...
-- adv1 (0N): state = 0x01
-- adv2 (0FF): state = 0x40 (standby)

local COMPANY_ID = "117" -- 0x0075 Samsung Electronics
local ENTRY_ID = "samsung_soundbar_q990b"
local HEADER = "420483"

local STATE_LABELS = {
  [0x01] = { state = "on", label = "On" },
  [0x40] = { state = "off", label = "Off (standby)" },
}
```

```
local state_byte = tonumber(data:sub(7, 8), 16)
if state_byte == nil then
  return entries
end

local known = STATE_LABELS[state_byte]
local state_str = known and known.state or string.format("0x%02X", state_byte)
local state_label = known and known.label or state_str
local mac = format_mac(data:sub(17, 28))
local parsed_protocol = "state=" .. state_str
if mac then
  parsed_protocol = parsed_protocol .. " mac=" .. mac
end
```

And now we can see the decoded state

Will it work with other soundbars?

Maybe with TVs?

Let's extend the dynamic observation with application reversing!

-64

★

SAMSUNG

[AV] Samsung Soundbar Q990B (+1)

D0:C2:4E:98:C1:A5

📄

🔗

✓ Capture (this session) · no svc

adv1

adv2

Generic · Adv ~401ms · 0s ago · Not bonded

Decoded

Parsed

Raw

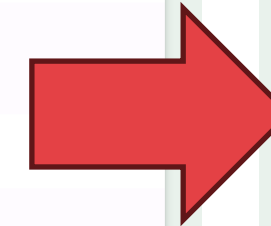
0xFF Manufacturer Specific Data

Company Code: 0x0075 (Samsung Electronics Co. Ltd.)

Data:

0x4204834011220703D0C24E98C1A50001D2C24E98C1A40106

0x08 Shortened Local Name



-98

★

SAMSUNG

[AV] Samsung Soundbar Q990B

On

D0:C2:4E:98:C1:A5

📄

🔗

✓ Capture (this session) · no svc

Audio · Adv ~3985ms · 2s ago · Not bonded

Decoded

Parsed

Raw

Source: Samsung Soundbar Q990B

^

(samsung_soundbar_q990b)

Samsung Soundbar Q990B

state: on

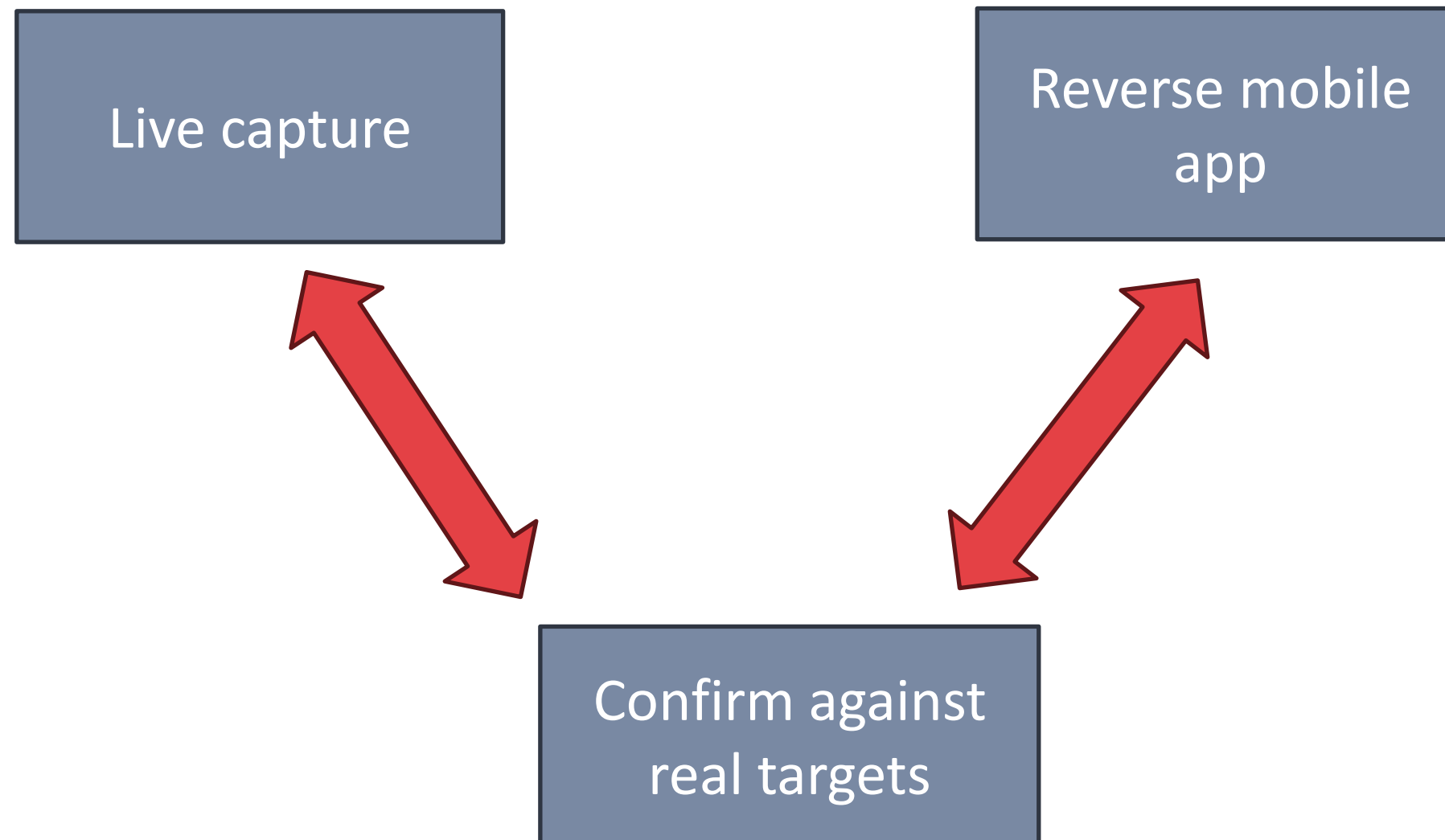
mac: D0:C2:4E:98:C1:A5

state_byte: 1

parsed_protocol: state=on mac=D0:C2:4E:98:C1:A5

state_label: On

Create new device



Mobile app reversing

Download apk (e.g. apkpure.com).

Older version also recommended (may decompile better).

Home > Apps > Lifestyle > SmartThings



SmartThings
 Samsung Electronics Co., Ltd.
 1.8.47.24

Download APK (231.1 MB)

Request APK on Telegram

How to install XAPK / APK file

9.3 ★
25 Reviews

Jun 28, 2026
Update date

0/60
Security

E
Everyone

Android 12.0+
Android OS

SmartThings 1.7.94.21

APK 97.5 MB Jan 14, 2023

Download

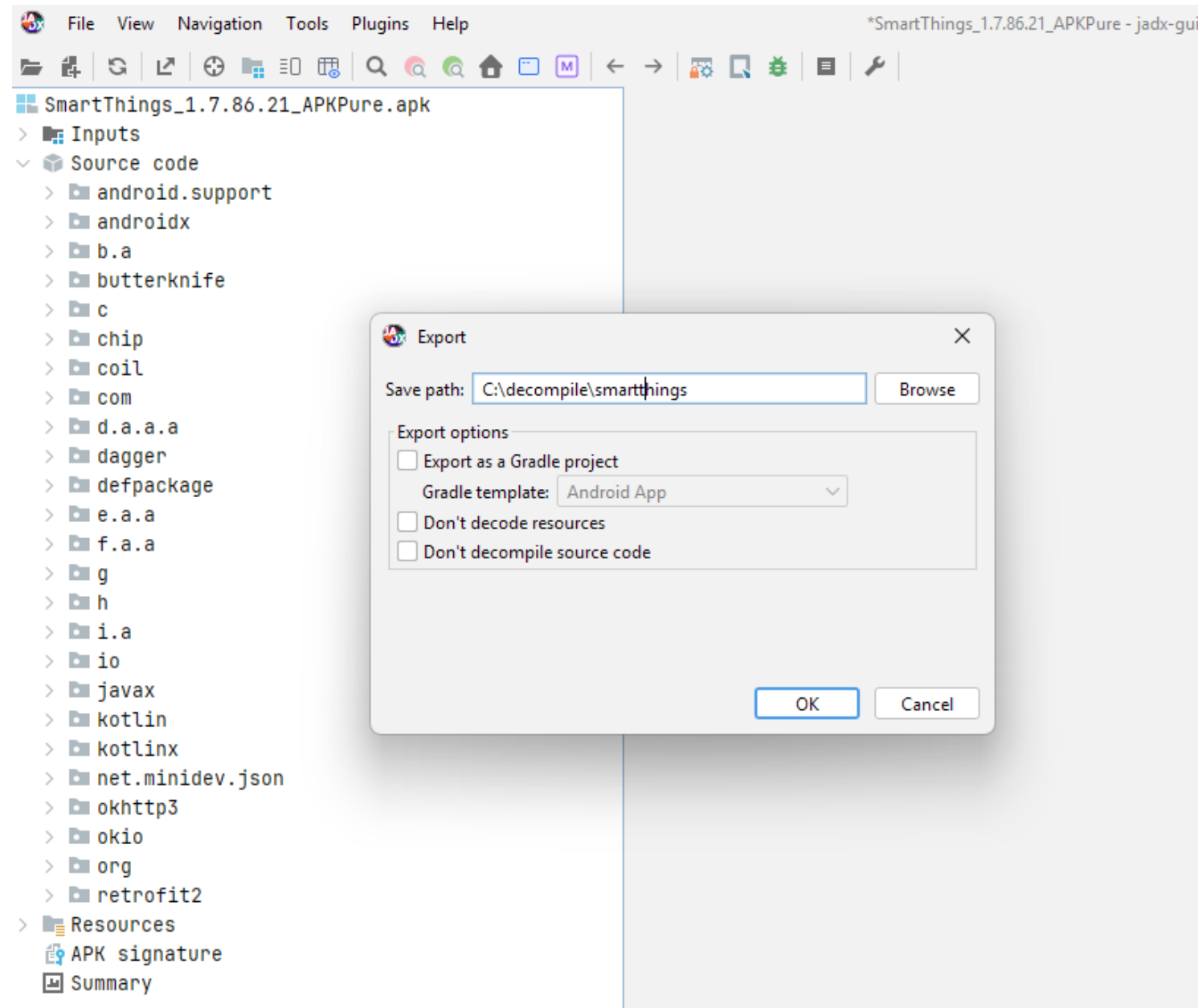
SmartThings 1.7.86.21

APK 90.8 MB Jul 1, 2022

Download

<https://apkpure.com/smartthings/com.samsung.android.oneconnect/>

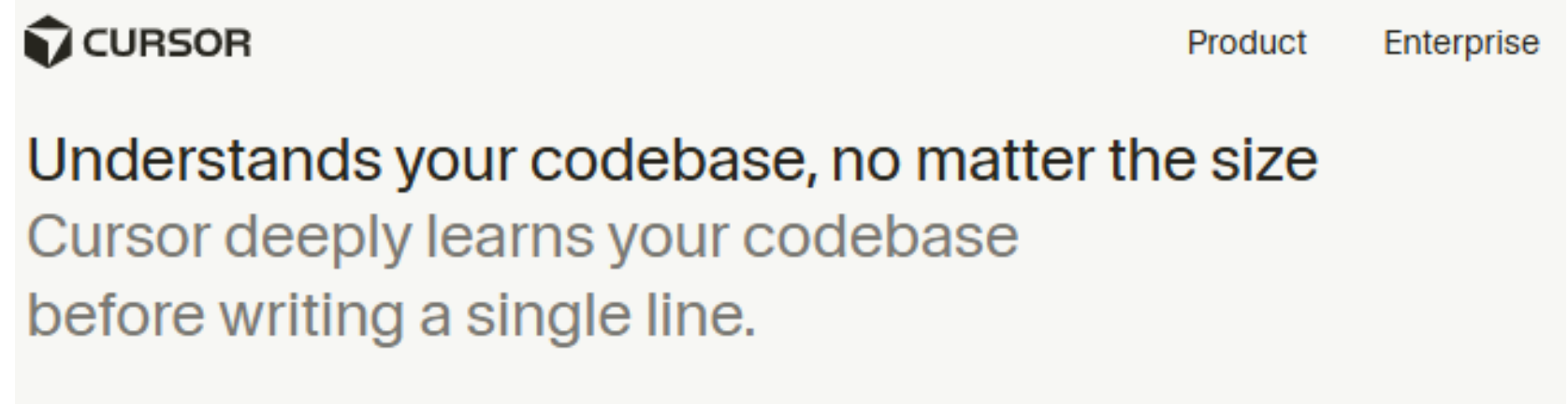
Decompile app, export (jadx-gui)



Agentic use

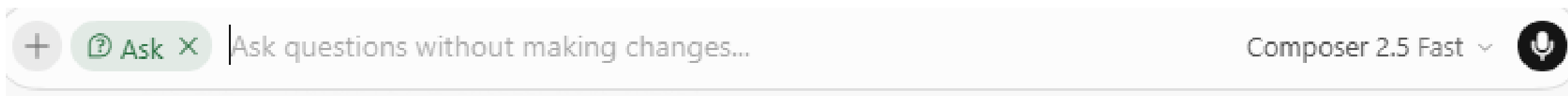
Grant access to the whole decompiled folder

- Android Studio
- Claude Code CLI
- Cursor IDE
- ...



<https://cursor.com/product>

Cursor: possible to switch models and mode (“Ask” for initial analysis)



Open the folder, ask agent (cursor: Ask mode)



Is it possible to figure out the Samsung devices BLE advertising format (@sample_scans) based on the decompiled application in @app_1.7.86.21 ?

+ Composer 2.5 Fast ▾



Worked for 1m 42s

Yes — it's feasible, and your two sources complement each other well. The sample scans give concrete payloads; the decompiled SmartThings app contains the parsers that interpret them. You won't get a single one-size-fits-all spec, but you can reverse-engineer the Samsung manufacturer-data layout for the device classes in `sample_scans/`.

Samsung decoder

AV / soundbars (type nibble 3) — fully documented in code

From `parseAvPacket` when `bArr[5] == 3`:

```
if (b4 == 3) {
    int i3 = (((bArr[20] & 240) >> 4) * 10) + 2000 + (bArr[20] & 15);
    byte b6 = bArr[21];          // avAvailableService
    byte b7 = bArr[22];          // ocfInfo / secDeviceIcon
    // P2P MAC @ [7:12]
    // BT MAC @ [13:18]
    String strG2 = this.a.g(bArr[19]); // mnId
    // setupId from [23:25]
```

TV (type nibble 1) — partially from samples + model classes

The TV-specific parser method `g.d()` failed decompilation, but samples + `DeviceBleTv` line up:

Offset	Example [TV] Samsung	Likely meaning
[5]	0x40 or 0x01	tvAvailableService or power/status
[6]	varies	extension / channel / flags
[7:12]	matches <code>bd_addr</code>	P2P / Wi-Fi Direct MAC
[13:18]	similar, last byte differs	Classic BT MAC
[19+]	EF 01..., FF 00...	OCF info, year, setup ID, registered DB

☰

LOCALREMOTERELAY

Filter ▼Scan ✂

-81

☆

SAMSUNG

[AV] Samsung Soundbar Q990B

On

D0:C2:4E:98:C1:A5

📄🔗

-88

☆

SAMSUNG

Samsung

Off

D4:9D:C0:A1:5A:B9

-92

☆

SAMSUNG

Samsung

Off

00:C3:F4:E4:0F:39

-85

☆

SAMSUNG

[TV] Samsung Q8 Series (65) (+1)

On

84:C0:EF:15:A4:21

📄🔗

-92

☆

SAMSUNG

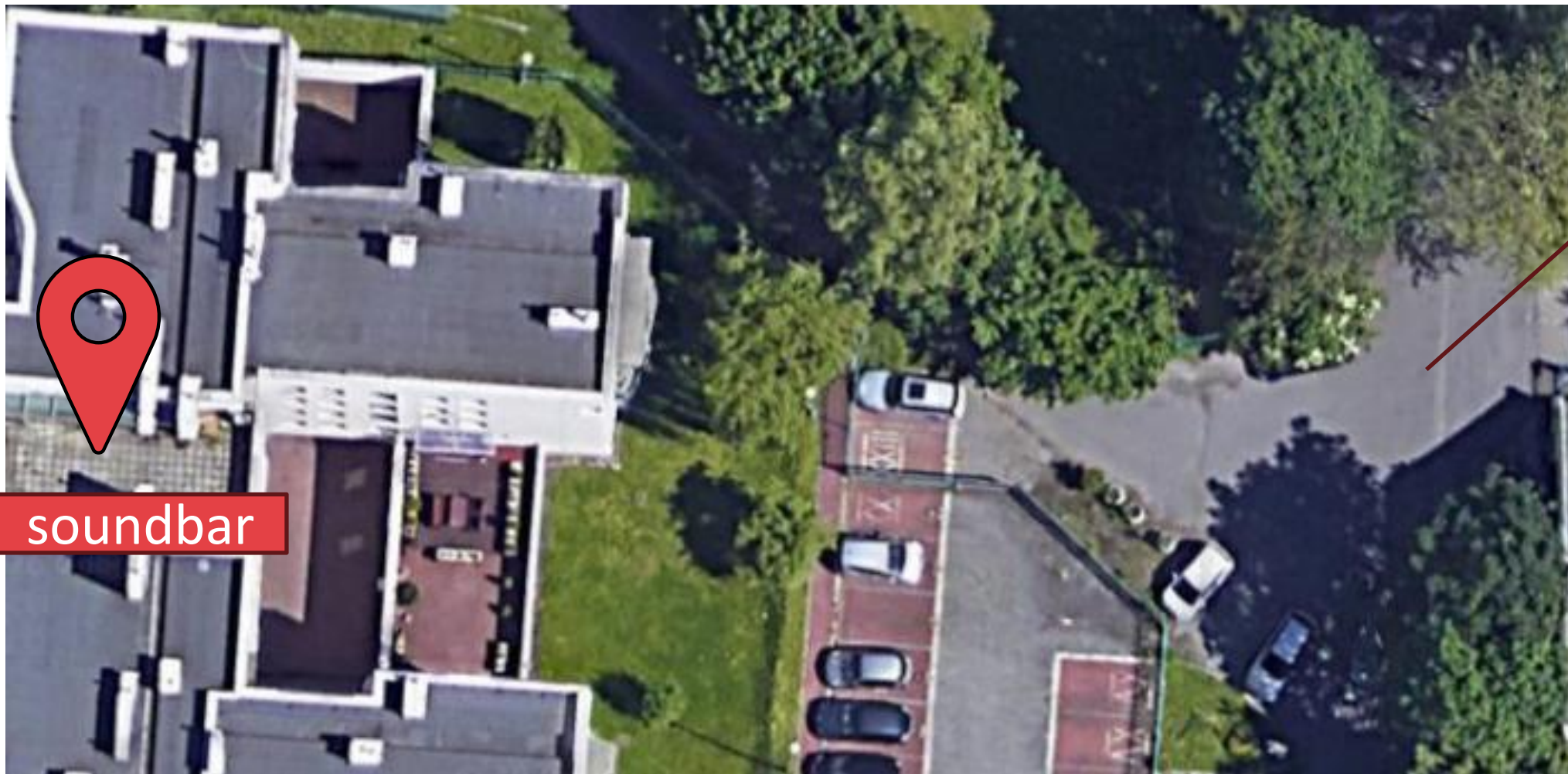
[TV] Samsung 7 Series (50)

On

D0:03:DF:F0:62:81

📄🔗

Is this a security issue?



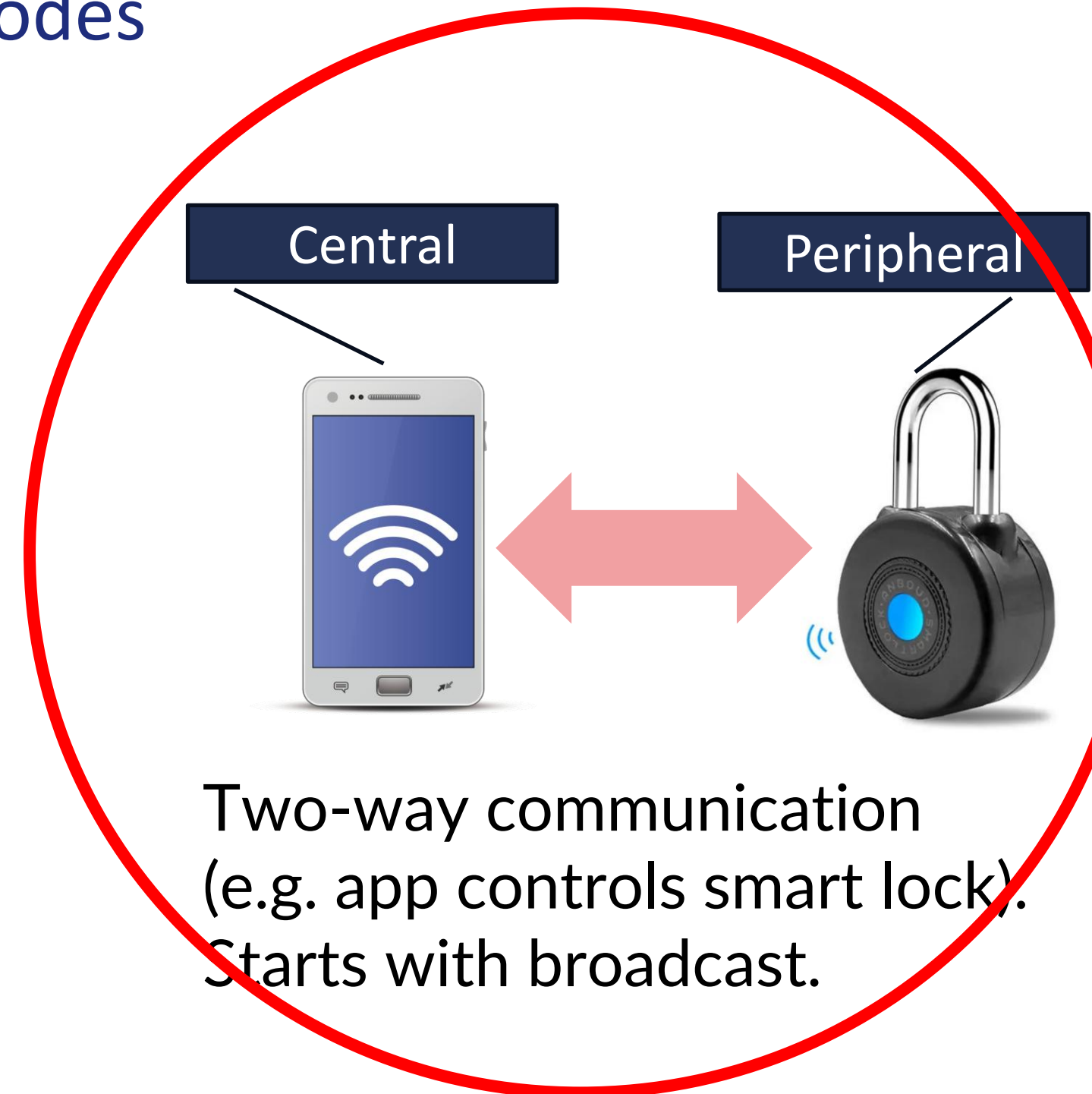
soundbar

signal visible in a
standard phone

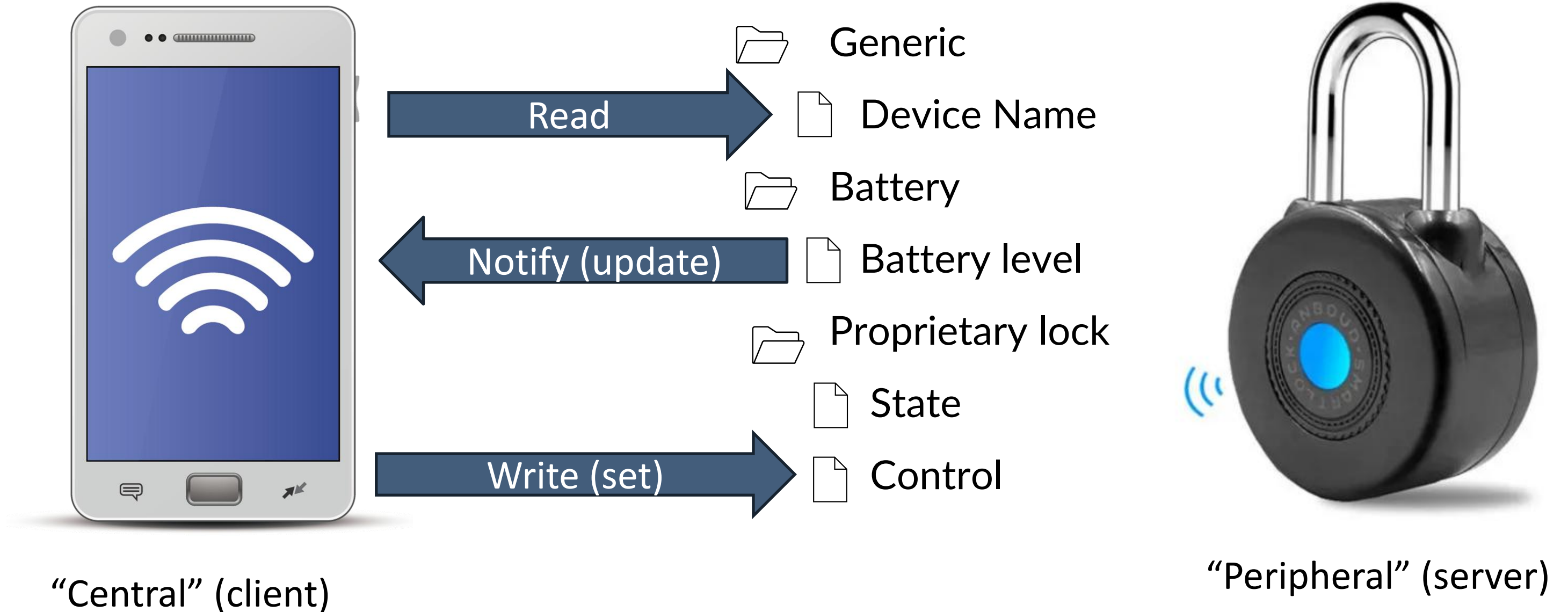
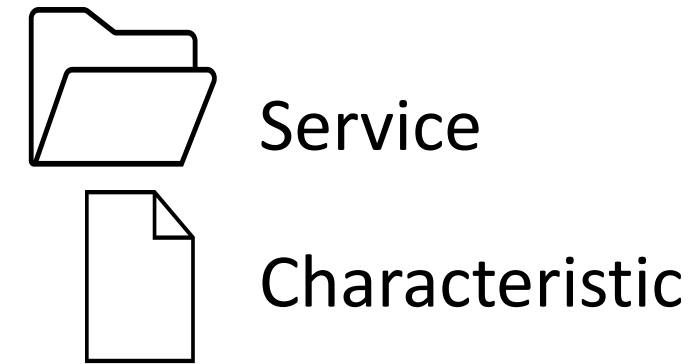


CONNECTIONS

BLE operation modes



BLE services and characteristics



UUIDs (“file names”)

Short, defined by Bluetooth spec, e.g.

0x1800 – Generic Access (mandatory)

0x2A00 – Device Name

0x180A – Device information (manufacturer name, model number...)

0x180F – Battery service

Long – for proprietary use

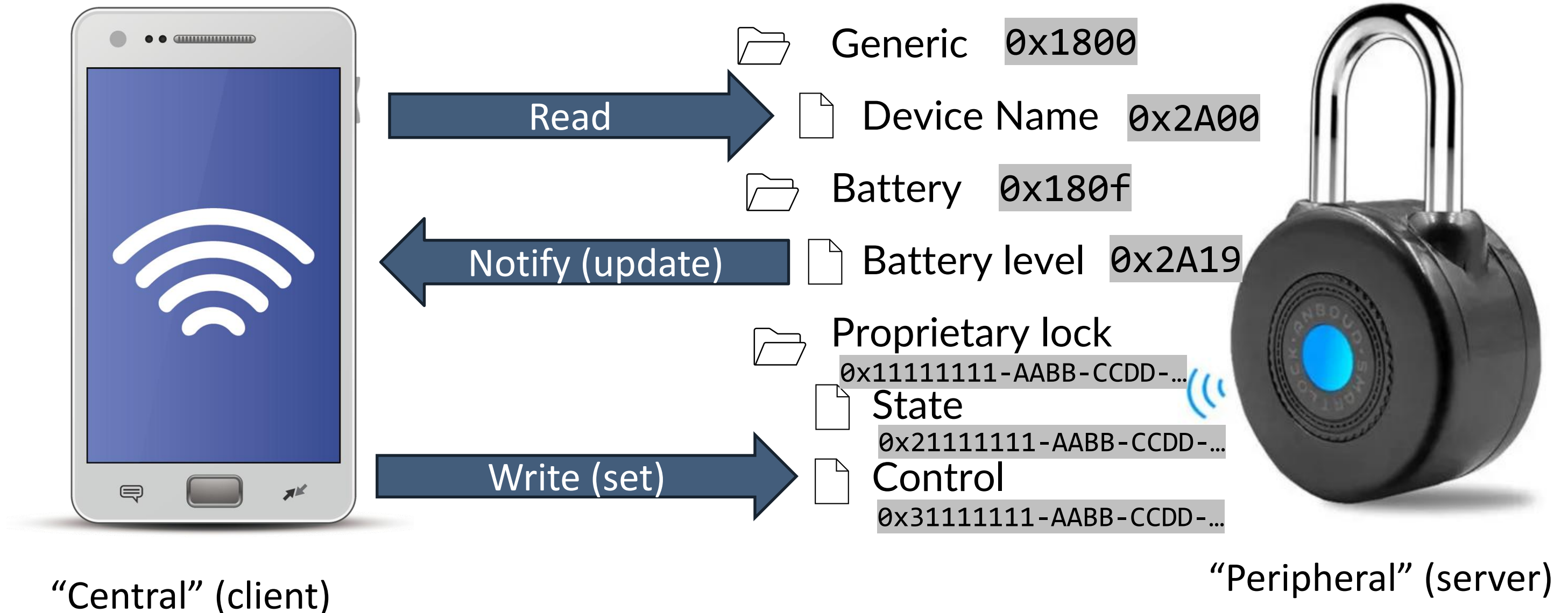
0x11111111-AABB-CCDD-EEEE-1234567890AB

- smart lock command

UUID	Service Name
0x1800	Generic Access service
0x1801	Generic Attribute service
0x1802	Immediate Alert service
0x1803	Link Loss service
0x1804	Tx Power service
0x1805	Current Time service
0x1806	Reference Time Update service
0x1807	Next DST Change service
0x1808	Glucose service
0x1809	Health Thermometer service
0x180A	Device Information service
0x180D	Heart Rate service
0x180E	Phone Alert Status service
0x180F	Battery service

BLE services and characteristics

 Service
 Characteristic



BLESPlo.it connect options

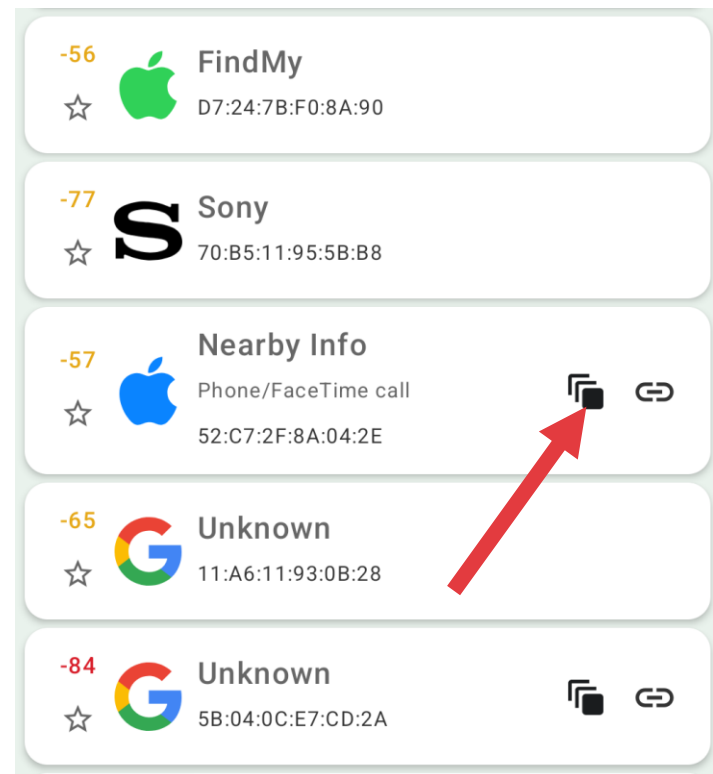
For connectable devices:



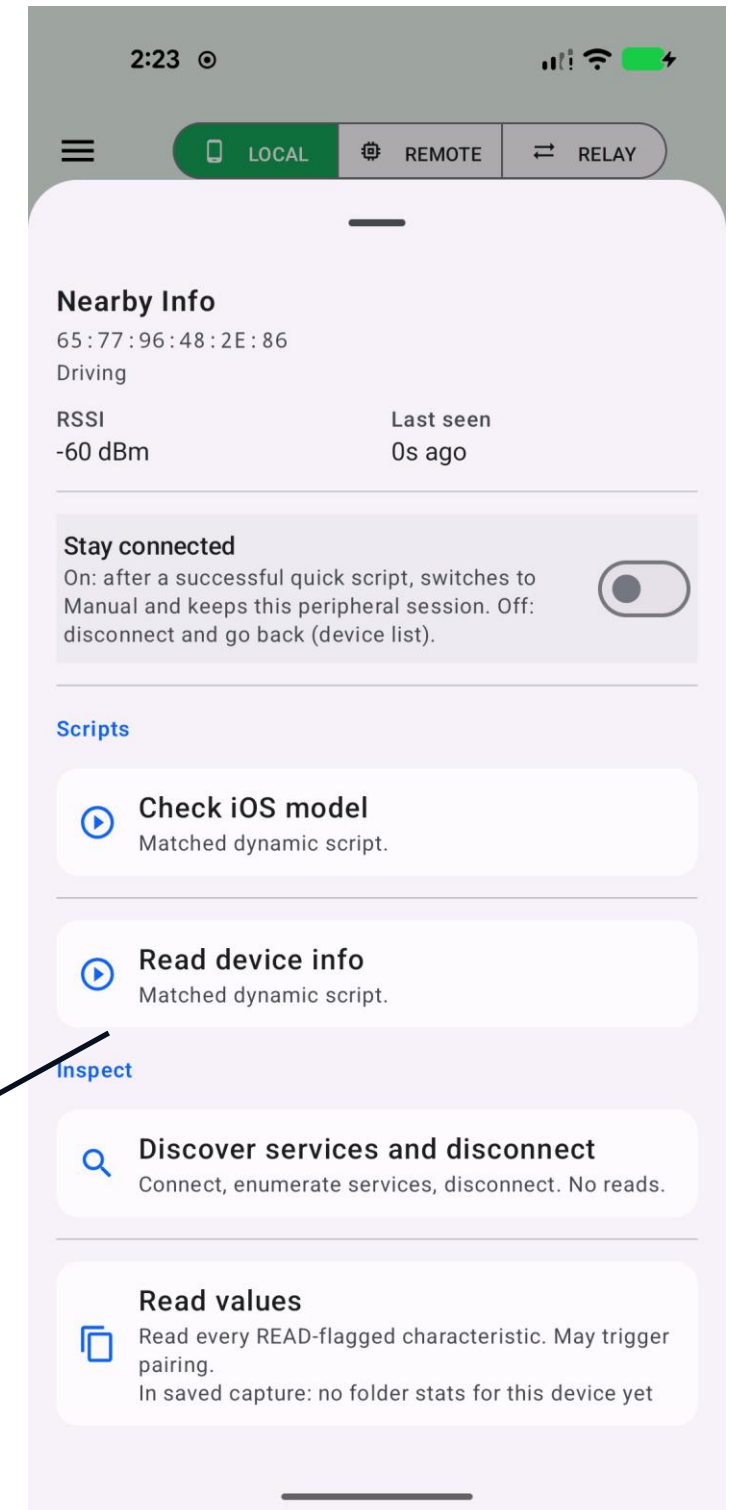
“quick action”: run a matching script automatically



connect manually (opens Services Browser)



Matching scripts



Phones

iOS: “check ios model”

-59

☆



iPhone SE (1st generation)

Driving

65:77:96:48:2E:86



Apple Watch SE (40mm, GPS, 2nd gen)

Activity reporting disabled

71:A5:4B:96:74:32



iPad (10th generation, Wi-Fi)

4E:0C:E6:B8:95:40

Android: read device info

-83

☆



Unknown

16:46:B9:15:B0:29



-83

☆



Galaxy A36 5G

04:20:68:04:26:2F

-79

☆



Kris z'n handi

2D:8D:80:FF:98:6F

<https://youtube.com/shorts/tiY5Onazs20>

Read device info: wearables, gadgets

Often serial numbers, hardware revisions, model details, ...



LE-Technics EAH-AZ40M2

CD:D8:7B:19:9D:1A

AIROHA-DIS-EXAMPLE · Airoha

RSSI
-92 dBm

Last seen
Scanning is off

► Read device info: Done

Enriched: model_number=AIROHA-DIS-EXAMPLE, system_id_hex=5.544332211887766E15, pnp_id_hex=021519eeee0100, device_name=LE-Technics EAH-AZ40M2, manufacturer_name=Airoha, serialNumberString=20180422, hardwareRevision=Version1.0, firmwareRevision=Version1.0, display_info=AIROHA-DIS-EXAMPLE · Airoha, enrichment_display_name=LE-Technics EAH-AZ40M2

Scripts

Read device info
Matched dynamic script.
user library

-99

☆

Mi Smart Band 6

C2:BD:EE:73:4F:E1

📄 🔗



📍 Fingerprint
Device info · Mi Smart Band 6

From GATT

▼ Reads & enrichment (overlay)

^ Device info

Generic BLE

system_id_hex	c2bdeefffe734fe1
pnp_id_hex	0157015b000101
device_name	Mi Smart Band 6
serialNumberString	32093/10484018
hardwareRevision	V0.82.19.3

-86

☆

Oclean_Y3L

OCLEANY3PB · Goodix

70:28:45:7F:59:CE

📄 🔗



📍 Fingerprint
Device info · OCLEANY3PB · Oclean_Y3L

From GATT

^ Reads & enrichment (overlay)

model_number	OCLEANY3PB
device_name	Oclean_Y3L
manufacturer_name	Goodix
firmwareRevision	1.0
hardwareRevision	HH
serialNumberString	1.0
system_id_hex	123456789abcdef0
pnp_id_hex	0000000000000000

Corona Tracer?

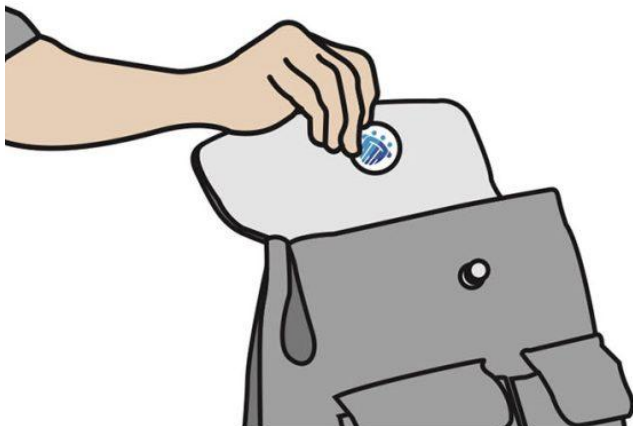
Model Number: Corona Tracer

Serial Number: 00909345

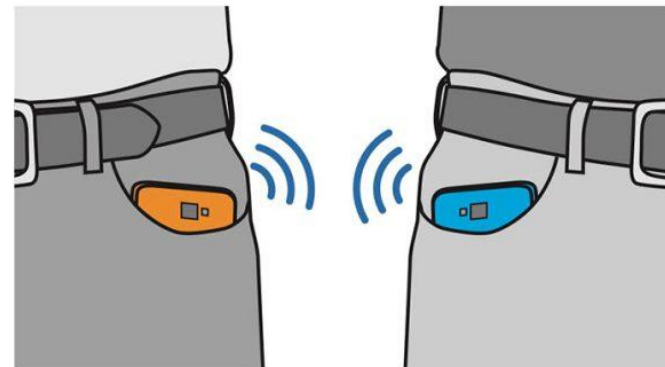
Firmware Revision: 0.4.14 (Jul 14 2021)

Software Revision: 2.2.4

1 Users carry around the TraceTogether Token with them. It will be a small device that they can slip into a bag or their pocket.



2 The token exchanges short-distance Bluetooth signals with other nearby tokens or handsets with the TraceTogether app. The interaction is logged in the devices.



SIZE COMPARISON

Width: 45mm
Height: 62mm
Thickness: 22mm

NOTE: Sizes are approximates.

ABOUT THE TOKEN

- Check that the green light is blinking**

Green light
If your token is working, the green light should blink once every minute. If you see a blinking red light, your token is not working.
- Data**
Stored information older than 25 days will be automatically erased.
- Don't cover the QR code on the token**

QR code
You will get a sticker to label your token, and you can write your name on it – just don't cover the QR code.
- Battery**
The battery lasts six months and is not rechargeable.

Wait, what about the pairing?

As opposed to Bluetooth Classic, link **pairing in BLE is optional**, and most devices don't perform it.

Sometimes selected characteristics are protected (trigger pairing on access). Usually no protection for device info (name, serial number, ...)

Often some proprietary application-layer protocol

- PIN / password protected
- Data encryption (shared key, ...)
- Plain packets of specific format

Proprietary protocol

Analyse mobile application
(static/dynamic)



Sniff/MITM
wireless

 **Bluetooth**
SMART

 **Bluetooth**
SMART



Dump/reverse
firmware
(static/dynamic)

Get the Android mobile application apk

- Extract from your own phone (root).
- Download from alternative store (apkpure.com, apk-dl.com, ...)
- Worth checking previous versions in case the current one is obfuscated


[Games](#)
[Apps](#)
[Articles](#)
[Products](#)
[AIPURE](#)

[Home](#) > [Apps](#) > [Tools](#) > **Rapidpower**



Rapidpower

Shantou Huahui Information Technology Co., Ltd.

1.2.4

[Download APK \(42.5 MB\)](#)

[How to install XAPK / APK file](#)

Apr 7, 2025

Update date



Security

E

Everyone

Android 7.0+

Android OS



Use APKPure App

Get Rapidpower old version APK for Android

[Download](#)

About Rapidpower

Rapidpower for controlling smart Bluetooth robotic dog toys

Supports voice command, control, content, feeding, programming and other functions.

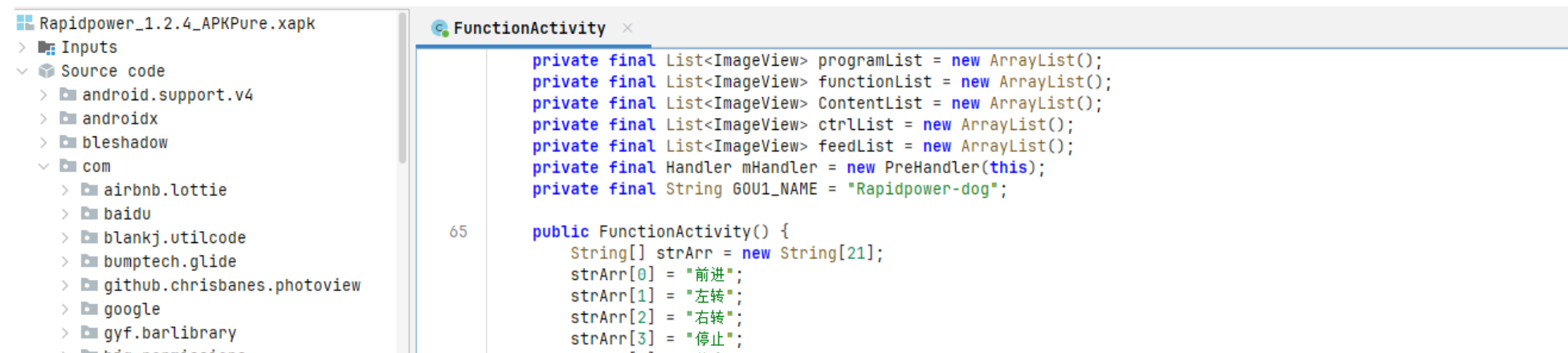
Additional APP Information

Latest Version  1.2.4	Uploaded by Attar Storeshop Bima
--	--

<https://apkpure.com/rapidpower/com.zhongrun.robotdog>

Decompile the Android mobile application

Jadx-gui, bytecode-viewer, ...



```

Rapidpower_1.2.4_APKPure.xapk
├── Inputs
├── Source code
│   ├── android.support.v4
│   ├── androidx
│   ├── bleshadow
│   └── com
│       ├── airbnb.lottie
│       ├── baidu
│       ├── blankj.utilcode
│       ├── bumptech.glide
│       ├── github.chrisbanes.photoview
│       ├── google
│       ├── gyf.barlibrary
│       └── ...
└── ...

```

```

FunctionActivity
private final List<ImageView> programList = new ArrayList();
private final List<ImageView> functionList = new ArrayList();
private final List<ImageView> ContentList = new ArrayList();
private final List<ImageView> ctrlList = new ArrayList();
private final List<ImageView> feedList = new ArrayList();
private final Handler mHandler = new PreHandler(this);
private final String GOU1_NAME = "Rapidpower-dog";

65 public FunctionActivity() {
    String[] strArr = new String[21];
    strArr[0] = "前进";
    strArr[1] = "左转";
    strArr[2] = "右转";
    strArr[3] = "停止";
}

```

```

this.cmdText = strArr;
this.cmdTextEn = new String[]{"move forward:forward", "turn left:left", "turn right:right", "stop", "swimm
this.cmdCode = new String[]{"F02A0001D5FFFE", "F02A0004D5FFFB", "F02A0007D5FFF8", "F02A000AD5FFF5", "F02A0
this.legsList = new ArrayList();
this.executeIndex = 0;

```



```

utils
view
├── BaseActivity
├── BaseStartActivity
└── FunctionActivity
    ├── AnonymousClass1
    ├── AnonymousClass2
    ├── PreHandler
    ├── EXECUTE_METHOD int
    ├── REQUEST_CODE_SPEECH_INPU
    └── ...

```

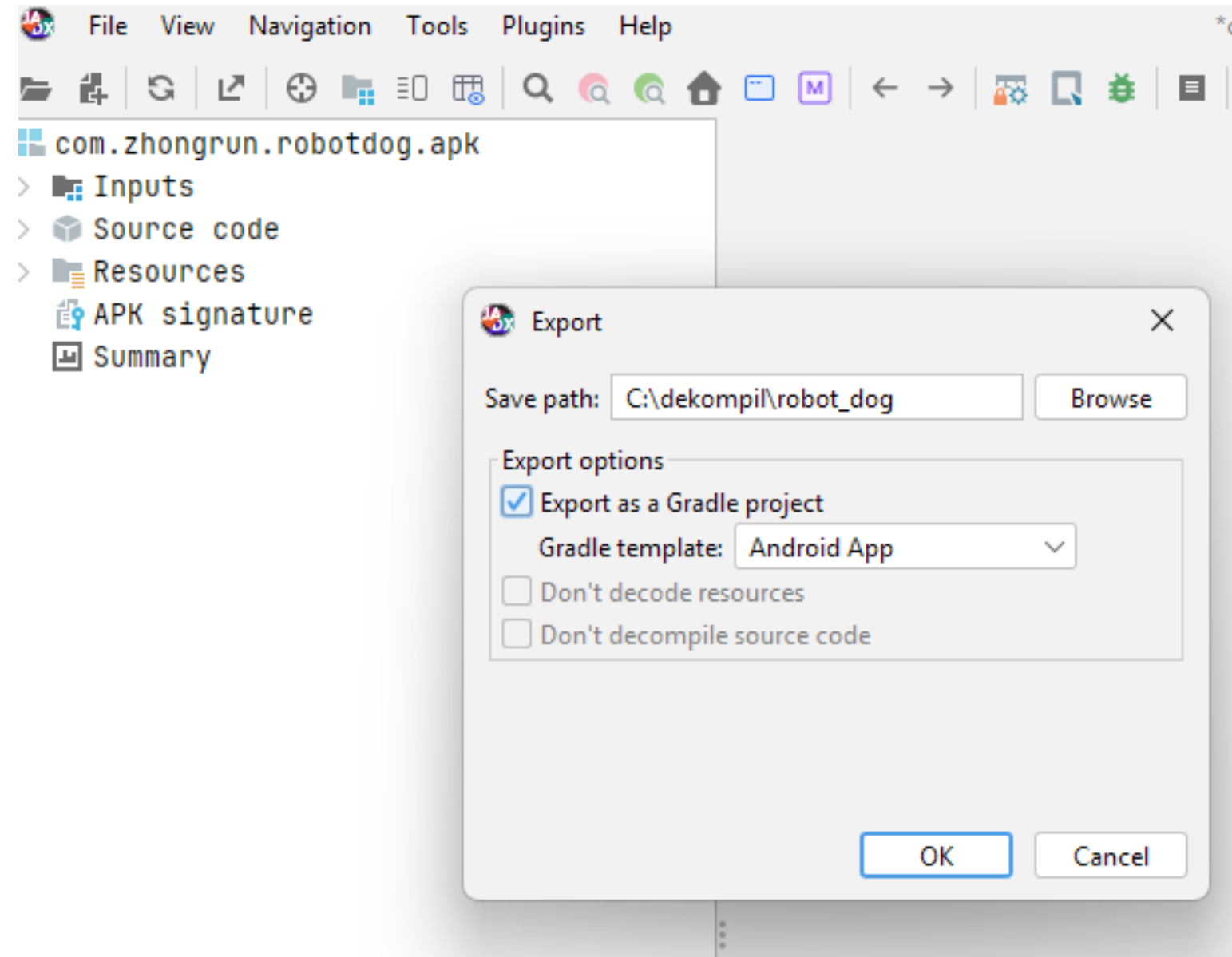
```

132 strArr[17] = "跳舞";
158 strArr[18] = "故事";
185 strArr[19] = "歌曲";
186 strArr[20] = "百科";
this.cmdText = strArr;
this.cmdTextEn = new String[]{"move forward:forward", "turn left:left", "turn right:right", "stop", "swimm
this.cmdCode = new String[]{"F02A0001D5FFFE", "F02A0004D5FFFB", "F02A0007D5FFF8", "F02A000AD5FFF5", "F02A0
this.legsList = new ArrayList();
this.executeIndex = 0;
}

```

Export

1. Decompile and export
2. Load the folder in your agent
3. Ask to explain how it works



Explain the communication protocol that the application is using to control the peripheral Bluetooth device

Worked

The app controls the robot dog over **Bluetooth Low Energy (BLE)** using a **custom binary command protocol** on top of **GATT characteristics**. There is no hardcoded service/characteristic UUID — the app discovers them at runtime.

Application-layer packet format

Commands are **hex strings** converted to bytes via `TypeUtils.hexToBytes()` before write.

Format A — fixed 7-byte preset commands

Most UI/voice commands use this layout:

```
[F0] [TYPE] [00] [PARAM] [(0xFF - TYPE)] [FF] [(0xFF - PARAM)]
```

Examples:

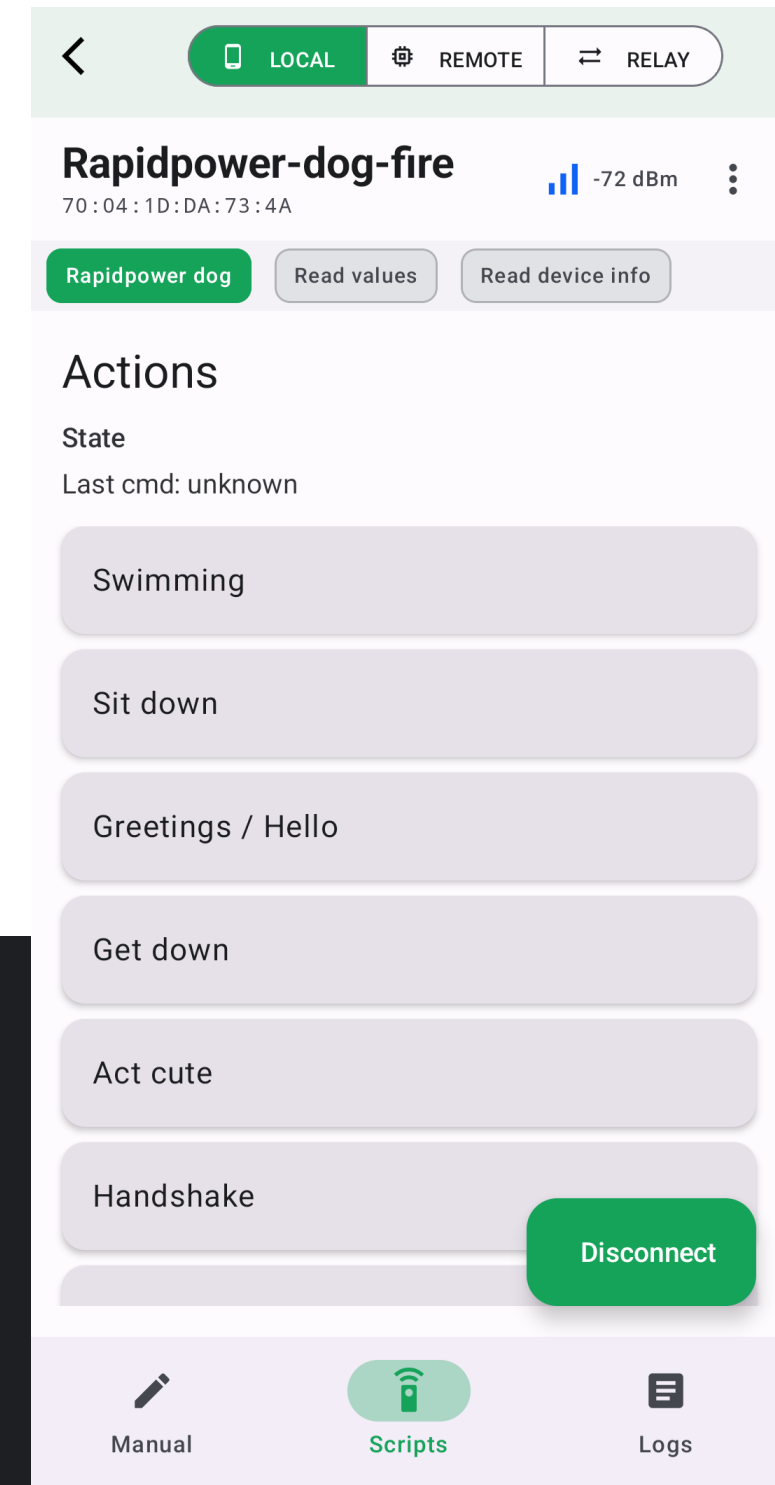
Hex	Meaning
F02A0001D5FFFE	Forward (type 0x2A, param 0x01)
F02A000AD5FFF5	Stop (param 0x0A)
F0300004CFFFFB	Feed/interaction (type 0x30, param 0x04)
F0120001EDFFFE	Story track 1 (type 0x12)

Ask agent to write BLESPloit script

Q: Write a dynamic menu script for the BLESPloit application that controls the dog

```
{
  "id": "movement",
  "title": "Movement",
  "items": [
    { "id": "send_cmd(F02A0001D5FFFE)", "label": "Move forward" },
    { "id": "send_cmd(F02A0004D5FFFB)", "label": "Turn left" },
    { "id": "send_cmd(F02A0007D5FFF8)", "label": "Turn right" },
    { "id": "send_cmd(F02A000AD5FFF5)", "label": "Stop" },
    { "id": "send_cmd(F02A002BD5FFD4)", "label": "Patrol" },
    { "id": "pop_menu", "label": "Back" }
  ]
},
```

```
function send_cmd(hex)
  if not hex or hex == "" then
    return
  end
  hex = hex:upper()
  local ok = ble_write(uuids.SVC_RAPIDPOWER, uuids.CHR_COMMAND, hex)
  if ok then
    set_title("CMD")
    set_state("cmd", hex)
  end
end
```



The screenshot shows the BLESPloit application interface. At the top, there's a navigation bar with a back arrow, a 'LOCAL' tab, and 'REMOTE' and 'RELAY' options. Below this, the device name 'Rapidpower-dog-fire' is displayed with a signal strength indicator at -72 dBm and the MAC address 70:04:1D:DA:73:4A. There are three buttons: 'Rapidpower dog' (highlighted in green), 'Read values', and 'Read device info'. The main section is titled 'Actions' and shows the 'State' as 'Last cmd: unknown'. A list of actions is displayed as buttons: 'Swimming', 'Sit down', 'Greetings / Hello', 'Get down', 'Act cute', 'Handshake', and a green 'Disconnect' button. At the bottom, there's a navigation bar with 'Manual', 'Scripts' (highlighted with a green icon), and 'Logs'.

Demo

LOCAL

REMOTE

RELAY

Filter

Scan

-72

S

Unknown

6A:91:82:63:04:11

-84

iBeacon

7C:F7:D0:4E:2E:55

-53

Windows Desktop

00:89:F3:96:F4:79

-90

Unknown

48:70:8C:B5:3E:D6

-67

Triones-FFFF3859A91F

Happy Light: Triones

FF:FF:38:59:A9:1F

-94

R07E9346680E1D62

90:EB:48:15:A9:D3

-73

REALOV_VIBE

Realov Vibe

:D2:69:E5:23:B1

-66

Unknown

Stop

Scan

Graph

Active connections



<https://youtu.be/g5mr9TURWvI>

That's fine, but why?

We could just use the official application – there was no auth / pairing...

- Not always obvious what application to use. BLESPlo.it = The One App To Rule Them All!
- Few lines of LUA vs heavy proprietary app
- We may want to send some invalid commands



Humanoid robots

Unitree G1

\$13,500.00 USD



Unitree H2

\$29,900.00 USD



Unitree H1 (Contact us for the real price)

\$90,000.00 USD



Robot architecture

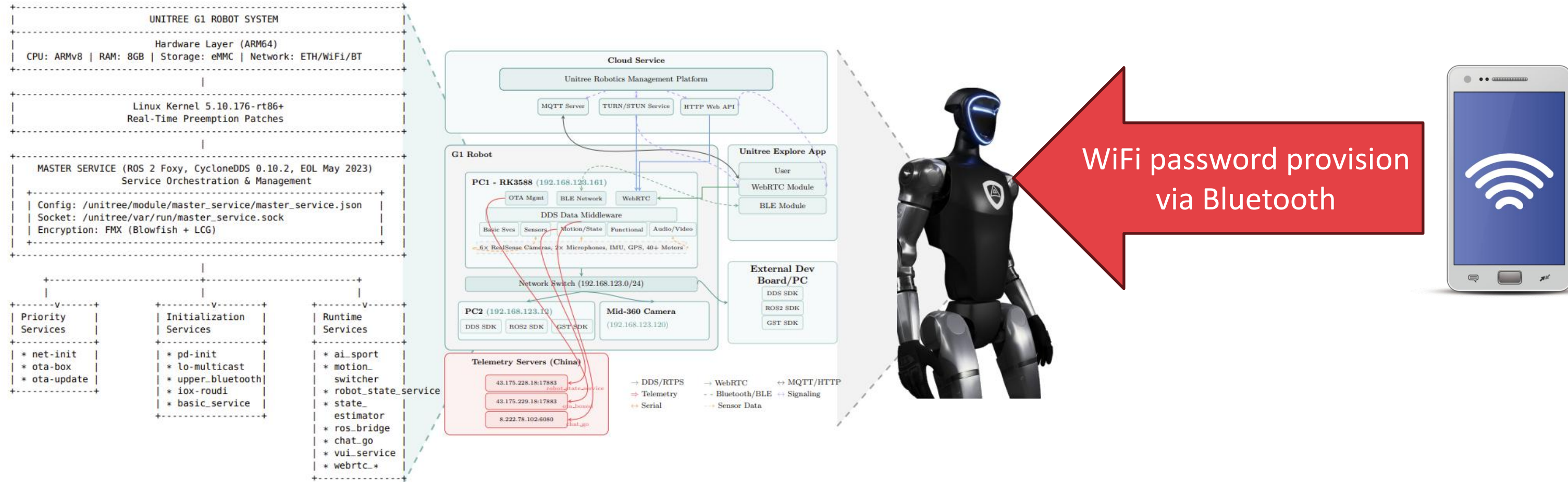


Figure 1: Humanoid Robot Architecture: Internal system structure showing hardware layer, Linux kernel, master service orchestration, and service hierarchy (left), and high-level ecosystem with communication paths showing authorized cloud services, telemetry servers, and internal components including obstacle avoidance, path planning, and speech recognition with DDS/ROS2 compatibility (right). **Critical finding:** Persistent telemetry connections to external servers transmit robot state and sensor data without explicit user consent.

BLE vulnerability in wifi provisioning

2.2 BLE Provisioning Vulnerability

The robot's Bluetooth Low Energy provisioning interface, accessible on models G1, H1, and R1 (also applicable to Go2 and B2 legged robots), contains a critical command injection vulnerability. The Wi-Fi configuration protocol accepts SSID and password fields without sanitization, allowing attackers to inject shell commands using the pattern `" ;$(command) ;#"`. All BLE traffic uses AES-CFB encryption with a static key (`df98b715d5c6ed2b25817b6f2554124a`) and IV (`2841ae97419c2973296a0d4bdfe19a4f`) hardcoded across the entire fleet, reducing effective entropy to zero. Exploitation requires only BLE proximity and knowledge of these universal credentials, enabling remote code execution with root privileges through the provisioning daemon. The vulnerability persists across firmware

```
$(reboot -f);#  
as wifi SSID or pass
```

Just use the application?

The “hack”:

1. Download app
2. Use default 1234 pin
3. Upload your own recording

The whole “security” was based on the fact that it was not obvious which app to use (and that there is an app).

Did they just change the pin?

An “unofficial” app could brute PIN combinations or bypass the authentication.



TECH NEWS

Crosswalks in Silicon Valley Hacked to Talk Like Musk and Zuckerberg

In one snippet, a crosswalk hacked to sound like Mark Zuckerberg told individuals there was nothing they could do to stop progress of AI.

BY THOMAS MAXWELL PUBLISHED APRIL 14, 2025, 11:40 AM ET

READING TIME 3 MINUTES



<https://gizmodo.com/crosswalks-in-silicon-valley-hacked-to-talk-like-musk-and-zuckerberg-2000589013>

The Polara protocol reverse

Explain the BLE communication protocol of this application with peripheral devices

Worked for 1m 37s

3. GATT services and characteristics

Defined in `INavPbs`:

Command/Response service — `5EE1B93C-3DF6-11E4-9D9F-164230D1DF67`

Characteristic UUID	Role
<code>AAAAAAAA-3DF6-11E4-9D9F-164230D1DF67</code>	TX: write commands (<code>writeCharacteristic</code> , 20 ms delay between writes)
<code>BBBBBBBB-3DF6-11E4-9D9F-164230D1DF67</code>	RX: command responses via notifications

MLDP service — `00035B03-58E6-07DD-021A-08123A000300` (Microchip MLDP)

Characteristic UUID	Role
<code>00035B03-58E6-07DD-021A-08123A000301</code>	Data — bulk TX/RX
<code>00035b03-58e6-07dd-021a-08123a0003ff</code>	Control

4. Application protocol: NBF (Network Button Frame)

All normal commands use NBF packets in `NBFPacket` / `DataPacket`.

Packet layout (minimum 3-byte header + payload)

Byte 0

Bits	Field
7–6	Framing: <code>00</code> single, <code>10</code> multi start, <code>11</code> multi end
5–0	Target LogicalId (0–63)

Byte 1

Bits	Field
7–6	C/R: <code>00</code> command, <code>01</code> response, <code>10</code> upload, <code>11</code> download
5	Command flush / frame-index flag
4–0	Session ID (5 bits)

Byte 2 — Command ID (responses use `command + 128`, see `Command.getRspByte()`)

Byte 3+ — Payload

Faucets at BlackHat in Mandalay Bay



Who would have expected faucets have Bluetooth...

CVE-2021-20107: unauthenticated BLE interface.

Based on the real device scan in [faucet_adsku02_a0149.zip](#), sample devices in [device-library](#) and [SKILL.md](#) write a new device library entry consisting of observer role to fingerprint such devices, and a quick action central script "dispense water": Opcode: Write Request (0x12)



Worked for 1m 31s 

Faucets in Mandalay Bay

-44

**FAUCET ADSKU02 A0149**

Sloan Optima · ADSKU02 · A0149



☆

Capture (this session) · 1/133

Smart home · Adv ~259ms · 0s ago · Not bonded

Raw

Parsed

Decoded

^

[Source: Sloan Optima faucet \(sloan_optima_faucet\)](#)

Sloan Optima

gen1_faucet_service: true
device_name: FAUCET ADSKU02 A0149
protocol: sloan_optima
generation: gen1
sku: ADSKU02
unit_id: A0149

Automatic

FAUCET ADSKU02 A0149

14:C1:9F:D5:9F:8E
Sloan Optima · ADSKU02 · A0149

RSSI
-35 dBm

Last seen
Scanning is off

Stay connected

On: after a successful quick script, switches to Manual and keeps this peripheral session. Off: disconnect and go back (device list).



^

Capture (this session) · 1/133

 Fingerprint
Sloan Optima · FAUCET ADSKU02 A0149



Scripts

 **Dispense water**
Matched dynamic script.

 **Read device info**
Matched dynamic script.

Inspect

 **Discover services and disconnect**
Connect, enumerate services, disconnect. No reads.



Read values

<

LOCAL

REMOTE

RELAY

FAUCET ADSKU02 A0149

14:C1:9F:D5:9F:8E

 -32 dBm 

Dispense water

Read values

Ri > 3 scripts

[Source: Sloan Optima faucet \(sloan_optima_faucet\)](#)

Dispense water

Run

sloan_optima: wrote 0x31 to Water Dispense (remote dispense)

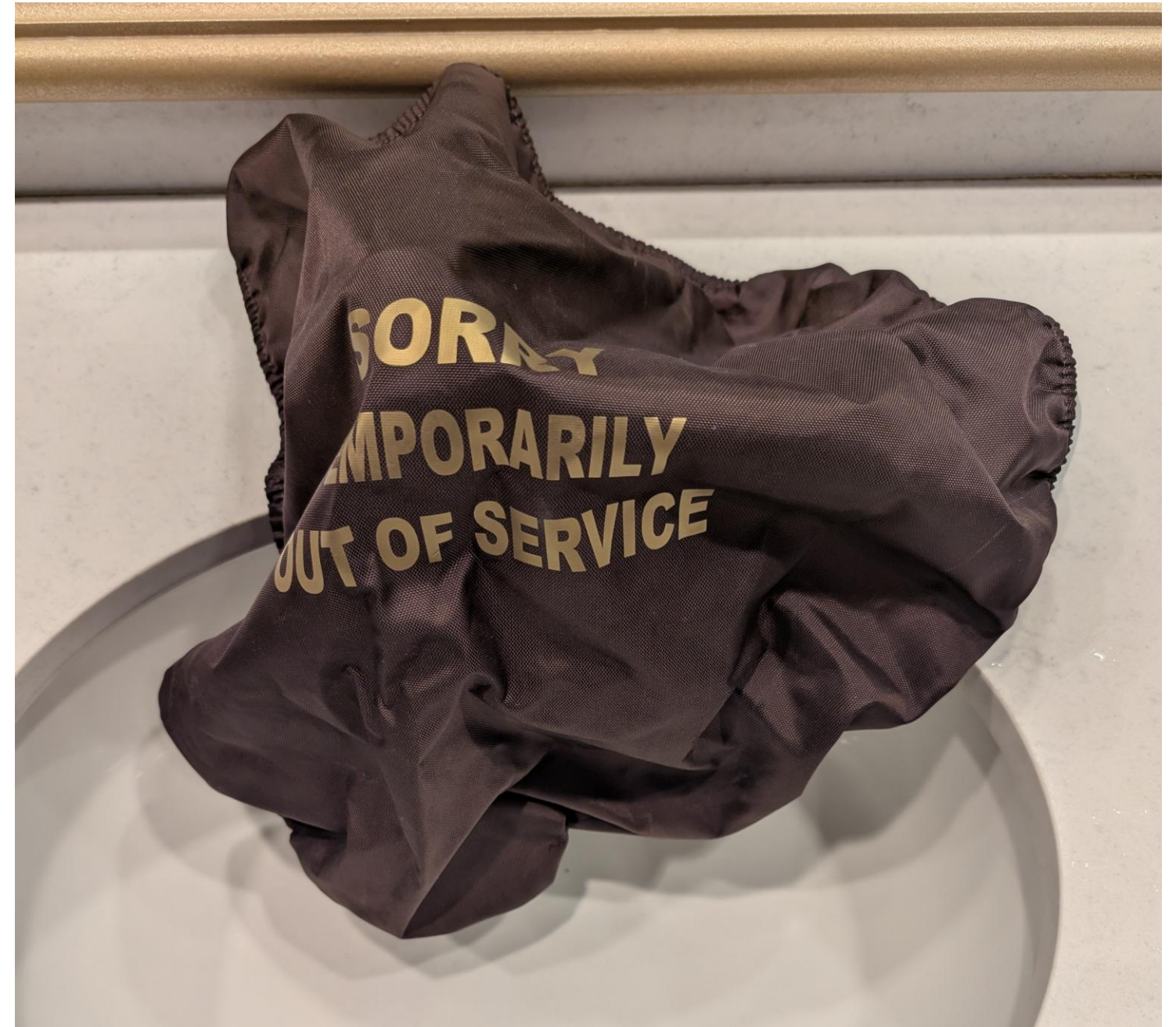
 Manual

 Scripts

 Logs

The broken faucet is just a coincidence...

I have absolutely nothing to do with this!



BTW Las Vegas: casinos also have Bluetooth



-86

☆

IGT Card Reader

Casino card reader

34:32:E6:21:44:DF

-91

☆

IGT Card Reader

Casino card reader

34:32:E6:21:40:90

✓

Capture (this session) · no svc

Generic · Not bonded

Raw

Parsed

Decoded

0x01 Flags

0x05: LE Limited Discoverable Mode, BR/EDR Not Supported

0x0A TX Power Level

-72 dBm

0x07 Complete List of 128-bit Service UUIDs

e079c6a0-aa8b-11e3-a903-0002a5d5c51b

0x02 Incomplete List of 16-bit Service UUIDs

0x1800 - GAP,

0x1801 - GATT

0x00 Complete Local Name

HPE aruba
networking

MAC: 389E4CC0D90C

S/N: [REDACTED]

BLE MAC: F01AA0D1A62D

-79	HPE Aruba AP (+2)	
☆	BluConsole · ch 222 20:4C:03:44:B6:8D	
-34	HPE Aruba AP (+1)	
☆	BluConsole · ch 233 20:4C:03:FB:0D:F1	
-73	HPE Aruba AP (+4)	
☆	BluConsole · ch 180 20:4C:03:44:F1:77	
-86	HPE Aruba AP (+2)	
☆	BluConsole · ch 28 20:4C:03:44:B5:8D	
✓ Capture (this session) · no svc		
Generic · Adv ~502ms · Not bonded		
iBeacon	HPE Aruba AP	HPE Aruba AP
Raw	Parsed	Decoded
^ Source: HPE Aruba Access Point (aruba)		
HPE Aruba AP		
record_subtype: 08		
bd_addr: 20:4C:03:44:B5:8D		
protocol: aruba_ap		

E-Rikshaw viral last month

Edition  IN

English

 Delhi
  38°C

THE TIMES OF INDIA

Subscribe

India

Gadgets Now

Urban Safety

Maharashtra

Delhi

Karnataka

Telangana

West Bengal

Tamil Nadu

Andhra Pradesh

News / India News / Explained: The App That Can Stop An E-Rickshaw — And W...

Trending
CBSE 10 second result
Mumbai rain
Gold rate today
Karnataka School Holiday
Trump tariff
<
>

Explained: The app that can stop an e-rickshaw — and why it has exposed a bigger cybersecurity risk

Yashasvi Vasistha / TIMESOFINDIA.COM / Updated: Jul 08, 2026, 12:56 IST

2 Comments
Share
Print
AA



Viral videos circulating on social media show individuals using smartphone apps to remotely disable moving e-rickshaws

One morning in Delhi, an e-rickshaw driver suddenly found his vehicle stranded in the middle of a busy road. There was no warning light, no smoke, no mechanical failure. The vehicle had simply stopped moving.

Assuming it was a technical fault, he pushed it to a nearby mechanic. What followed surprised him. The mechanic opened a mobile application, tapped a few buttons, and the rickshaw came back to life within minutes.

<https://timesofindia.indiatimes.com/india/explained-the-app-that-can-stop-an-e-rickshaw-and-why-it-has-exposed-a-bigger-cybersecurity-risk/articleshow/132236387.cms>

E-Rikshaw: supply chain problem

Chinese battery comes with a Bluetooth-enabled BMS managed by an app.

By default it's not protected, anyone in range can control (e.g. shut down) the battery.



<https://timesofindia.indiatimes.com/videos/news/bluetooth-security-flaw-in-some-e-rickshaw-batteries-allows-remote-shutdown-reports/video/132153678.cms>

The Indian government's solution

Ban the applications!

Congratulation, problem solved!

Centre Orders Removal Of 7 Apps Used To Stop E-Rickshaws In Delhi

MeitY Secretary S Krishnan said the government will take up this issue with the app stores to ensure potentially harmful applications don't come up in the public domain.

Reported by: [Ravish Ranjan Shukla](#) | Edited by: [Aastha Ahuja](#) | [India News](#) | Jul 03, 2026 19:20 IST ⓘ

Read Time: 3 mins

Add **NDTV** As A

Share



<https://www.ndtv.com/india-news/centre-orders-removal-of-chinese-apps-used-to-stop-e-rickshaws-in-delhi-bat-bms-epoch-li-ion-e-rickshaw-prank-11721450>

BAT-BMS reversing

Explain the BLE communication protocol of the application with peripheral devices (battery systems)

Worked for 2m 6s >

1. GATT Profile (BLE Layer)

Battery peripherals expose a vendor-specific service that acts as a **transparent serial channel**:

Role	UUID
Service	0000ff00-0000-1000-8000-00805f9b34fb
Notify (RX from BMS)	0000ff01-0000-1000-8000-00805f9b34fb
Write (TX to BMS)	0000ff02-0000-1000-8000-00805f9b34fb

```
public static final String uuid_char = "0000ff02-0000-1000-8000-00805f9b34fb";  
public static final String uuid_readChara = "0000ff01-0000-1000-8000-00805f9b34fb";  
public static final String uuid_service = "0000ff00-0000-1000-8000-00805f9b34fb";
```

- **App → BMS**: writes raw protocol bytes to `FF02`
- **BMS → App**: pushes responses via notifications on `FF01`

There is also a **legacy HM-10-style profile** (`FFE0` / `FFE1`) in `BluetoothClientSingleGrenergy`, but the active path uses `FF00` / `FF01` / `FF02`.

3. Application Protocol (Serial Framing)

Above BLE, the app speaks a **binary serial protocol** with three frame types.

Format A — Standard BMS frame (primary)

Used for almost all telemetry and control.

Field	Value
Start	0xDD
Read mode	0xA5
Write mode	0x5A
End	0x77

BAT-BMS protocol

Offset	Field	Scale
0–1	Total voltage	$\div 100 \rightarrow V$
2–3	Current	$\div 100 \rightarrow A$
4–5	Remaining capacity	$\div 100 \rightarrow Ah$
6–7	Nominal capacity	$\div 100 \rightarrow Ah$
8–9	Cycle count	integer
10–11	Production date	bit-packed Y-M-D
12–15	Balance state bitmask	per-cell
16–17	Protection state	14 fault bits
18	Firmware version	hex
19	RSOC %	
20	FET state	bit0=charge MOS, bit1=discharge MOS
21	Cell count	
22	NTC count	
23+	Temperatures	$(raw - 2731) / 10 \text{ }^{\circ}C$

Configuration & control

Cmd	Hex	Purpose
0	0x00	Open factory mode (56 78)
1	0x01	Close factory mode (00 00)
6	0x06	Password pair
7	0x07	Password change
9	0x09	Clear password
10	0x0A	Control subcommands (reboot, clear alarms, sleep, balance, etc.)
45	0x2D	Function config bitfield
46	0x2E	NTC config
170	0xAA	Protection event counters
171	0xAB	Charge history
246	0xF6	Internal resistance per cell
250	0xFA	Parameter block read/write by address
251	0xFB	MOS switch / lock control

By the way Indian government and Bluetooth

GitHub ordered to remove decentralized messaging app Bitchat in India

July 24, 2026 By [Alex Lekander](#) — [Leave a Comment](#)



the government of india does not like technologies like bitchat and wants it taken down



14C-MHA
Government of India

Notice No:11072601011432

Date: 23/07/2026 23:16:04

Notification No. : CG-DL-E-14032024-252989 Dated 12/03/2024

ORDER FOR REASONED INTIMATION TO INTERMEDIARY FOR REMOVAL OR DISABLING ACCESS TO UNLAWFUL INFORMATION AS PER S. 79(3)(b) OF THE IT ACT, 2000 r/w RULE 3(1)(d) OF THE IT RULE, 2021

Whereas, Section 79(3)(b) of the Information Technology Act, 2000 provides that the exemption from liability of intermediaries for third party information, data, or communication link made available or hosted by them shall not apply if upon receiving actual knowledge, or on being notified by the appropriate Government or its agency that any information, data or communication link residing in or connected to a computer resource controlled by the intermediary is being used to commit the unlawful act, the intermediary fails to expeditiously remove or disable access to that material on that resource without vitiating the evidence in any manner.

the following:

Sr. No.
1
2
3

Brief Su
Home A
establish
mobile n
communi
of comm
intercept
directly t
evade la
imposed
organize
communi

<https://x.com/jack/status/2080565084586135773>

<https://cyberinsider.com/github-ordered-to-remove-decentralized-messaging-app-bitchat-in-india/>

Also by the way: implementing BitChat in BLESPloit took 4m10s

Based on the source code [bitchat](#) sample implementations in device-library and [SKILL.md](#) write a BLESPloit device library entry that would work in observer role to decode the Bitchat devices

Worked for 2m 28s

Add also central role quick action script to read peer identity, and uuids.json

Worked for 1m 42s 

BitChat in BLESPloit

-56

☆

BIT

mainnet

5C:D6:35:D3:40:50

▼

Capture (this session) · no svc

Nearby · Adv ~287ms · 0s ago · Not bonded

Raw

Parsed

Decoded

^

Source: [bitchat \(bitchat\)](#)

bitchat

device_name:

protocol: bitchat

network: mainnet

characteristic_uuid:

a1b2c3d4-e5f6-4a5b-8c9d-0e1f2a3b4c5d

service_uuid: f47b5e2d-4a9e-4c5a-9b3f-8e1d2c3a4b5c

Automatic

bitchat

5C:D6:35:D3:40:50

mainnet

RSSI

-64 dBm

Last seen

0s ago

Stay connected

On: after a successful quick script, switches to Manual and keeps this peripheral session. Off: disconnect and go back (device list).

▼

Capture (this session) · no svc

Fingerprint

bitchat ·

▼

Scripts

▶

Read peer identity

Matched dynamic script.

▶

Read device info

Matched dynamic script.

Fingerprint

bitchat ·

▼

▶ Read peer identity: Done

Run log

Hide

```

=== bitchat peer identity ===
network           : mainnet
nickname          : anon3005
sender_id         : 4c0e3e37d803b590
peer_id (noise)   : 4c0e3e37d803b590
peer_id match     : true
noise pubkey      :
87a218a34d687666d6251d579dd37e211a03cdb902e
de92faecc71e75dab9342
signing pubkey    :
29eadd38aaa73616a44ae347c524e45f7f9d970bb9d
35f85ade40837630f110d
capabilities      : 0x0 (none)
neighbors         : 0
announce ts(ms)   : 1785222656029
signed            : true
Summary: anon3005 · 4c0e3e37d803b590
Disconnecting

```

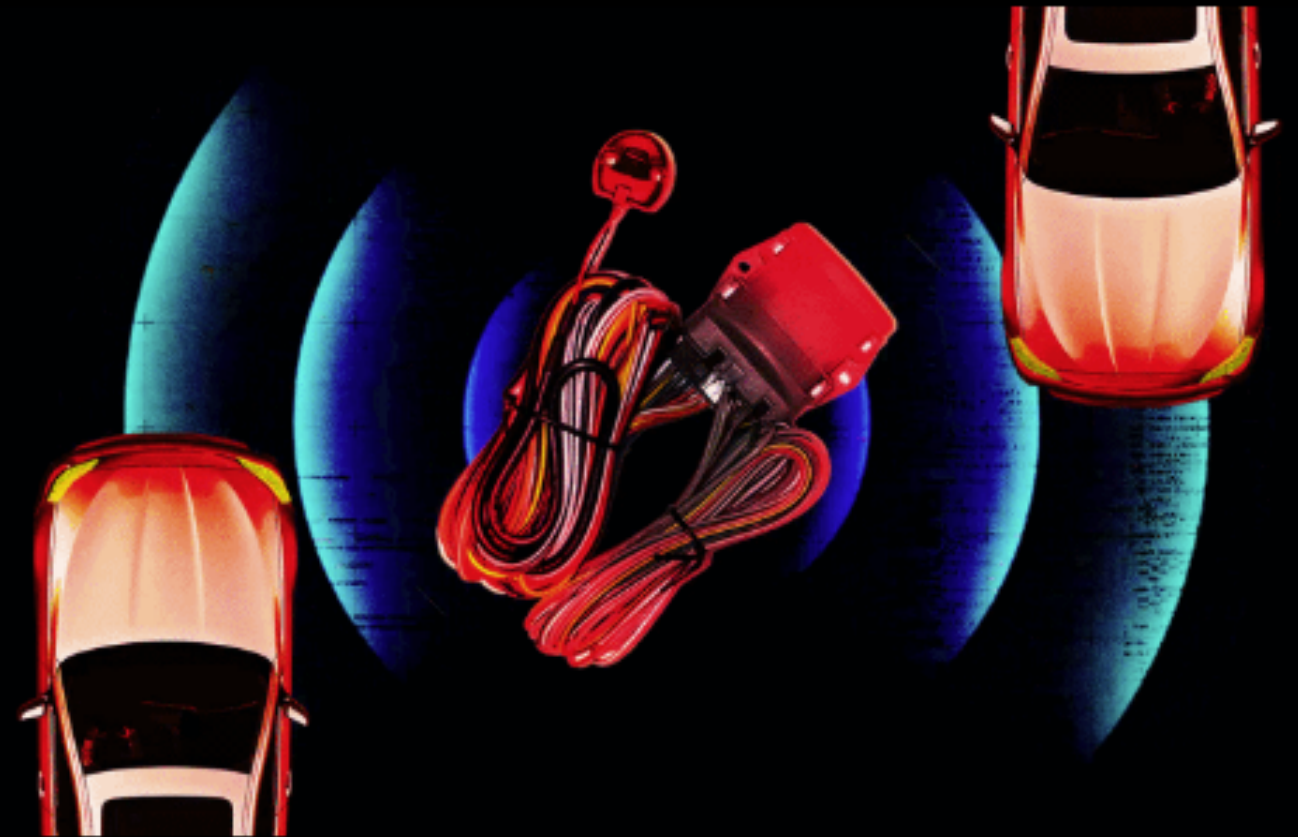
2026.07: KARR Security

ANDY GREENBERG

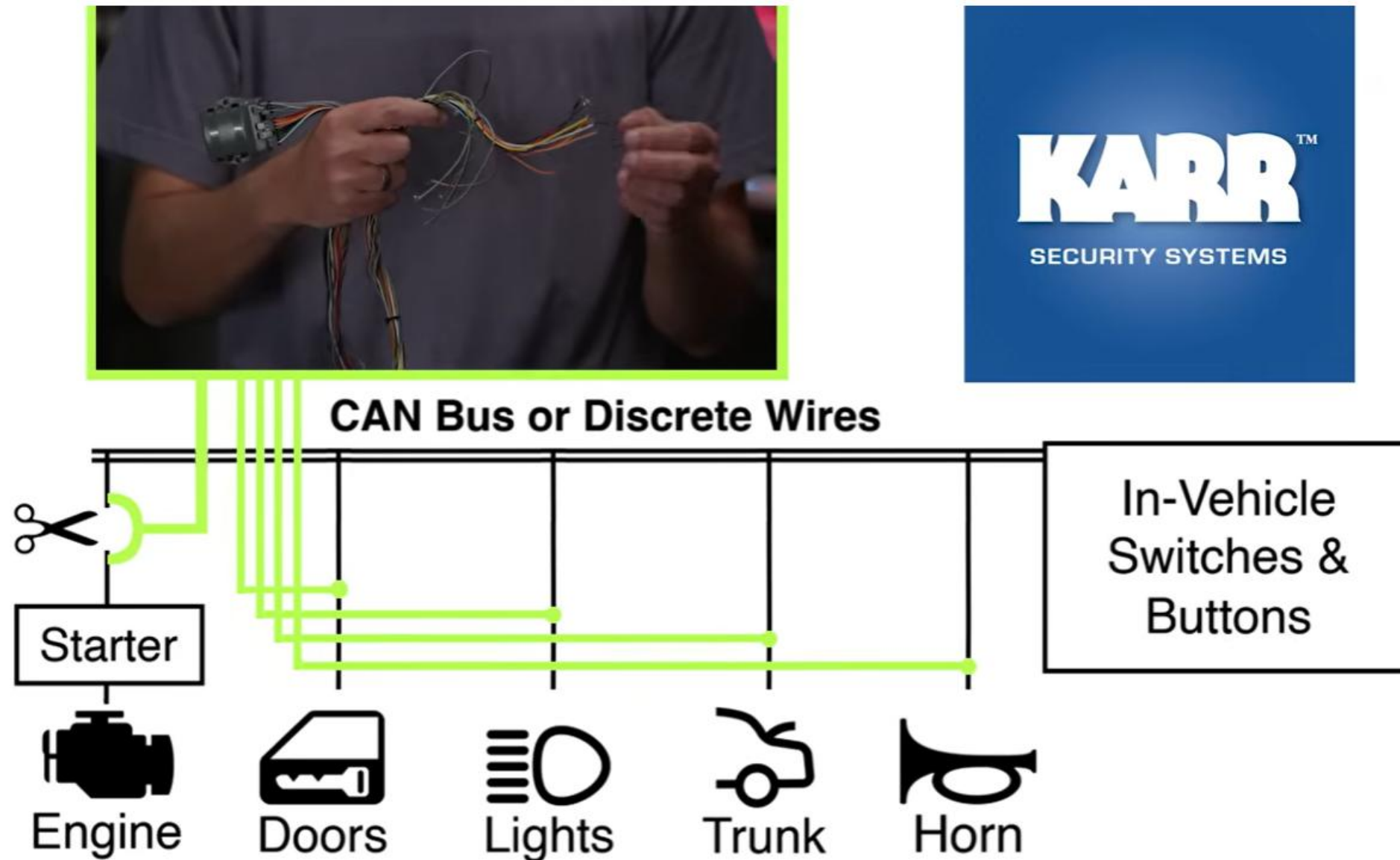
SECURITY JUL 21, 2026 6:00 AM

A Device Hidden in Cars Across the US Leaves Them Vulnerable to Hacking and Paralysis. Patch It Now

Dealerships installed alarms in millions of vehicles—and left them in even if the buyer didn't want them. Now researchers warn they can be hacked to unlock, track, and disable cars.



A single shared key controls all the vehicles (engine, doors, lights, horn)



<https://www.youtube.com/watch?v=FwA3CuJxbk4>

Protocol reverse

Based on the decompiled code explain the BLE communication protocol of the application with peripheral devices

Worked for 23m 26s

The app talks to Quantam/Gridtraq units over BLE GATT Transparent UART (Microchip/ISSC style), then layers a custom Quantam `QtProtocol` on top. Transport is in `SWDSCustomer` (`BTServices` / `TransparentData`); framing, auth, and opcodes are in `MGridtraqPL` (`QtProtocolParser`, `AuthService`).

Application framing (`QtProtocolParser`)

Three modes (`ProtocolMode`):

Mode	Sync	Layout
Original (0)	0xAA	[AA][len_hi][len_lo][cmd][payload...][checksum]
Counter (1)	0xBB	[BB][counter][len_hi][len_lo][cmd][payload...][checksum]
Encrypted (2)	0x55	[55][IV 16 bytes][AES ciphertext...]

- **Length** includes the command byte (`data.Length + 1`).
- **Checksum**: 8-bit two's complement so the mod-256 sum of checked bytes is 0; failure string is "CRC mismatch" (not CRC32).
- **Encryption**: AES-128-CBC, **Zeros** padding; key from vehicle `CustomKey` (Base64), often fetched via cloud (`GetUnitKey` / `GUK`) if missing. Inner plaintext is a counter-mode frame without the 0xBB sync.

Wait, what took you so long?

Worked for 23m 26s

The app wraps .NET into Android

How .NET runs on Android here

This is a .NET MAUI app (formerly Xamarin-style). On device:

1. Normal Android process starts (`MainApplication` / `MainActivity`).
2. `mono.MonoRuntimeProvider` boots the **Mono/.NET runtime** embedded in the APK.
3. That runtime loads your C# assemblies and runs them; Java only bridges Android APIs (BLE callbacks, UI host, etc.).

So it is still one Play Store / sideload Android app. .NET is a **runtime + libraries packaged with the APK**, not "install .NET separately" and not "download the app logic from a CDN at first launch."

Where the DLLs live in *this* package

It's an **Android App Bundle** style layout (`requiredSplitTypes` includes `base__abi`). The ABI split `config.arm64_v8a.apk` contains native libs such as:

Artifact	Role
<code>lib/arm64-v8a/libmonogen-2.0.so</code>	Mono GC/runtime
<code>lib/arm64-v8a/libxamarin-app.so</code>	App/runtime glue
<code>lib/arm64-v8a/libassembly-store.so</code>	Packed managed assemblies (blob)
<code>lib/arm64-v8a/libaot-MGridtraqPL.dll.so</code>	AOT image for protocol lib
<code>lib/arm64-v8a/libaot-SWDSCustomer.dll.so</code>	AOT image for app
<code>lib/arm64-v8a/libaot-Plugin.BLE.dll.so</code>	AOT for BLE plugin
plus many other <code>libaot-*.dll.so</code>	Framework / NuGet AOT

KARR+Security_0.49_APKPure.apk

```
> Inputs
> Source code
  > android
  > androidx
  > com
  > crc64087678da79fdfe22
  > crc640879981a986d658b
  > crc640d7c6d57b8a5f296
  > crc6414252951f3f66c67
  > crc641881f1fcda569922
  > crc642118fa8c490a1683
  > crc6423633d566e126249
  > crc64256bdbbc174c459bf
  > crc64345add925791a062
  > crc6434af9c19aa01b597
  > crc64396a3fe5f8138e3f
  > crc643ddf594df781e5ec
  > crc643f46942d9dd1fff9
  > crc644103bb497e895a1b
  > crc644c2e2be3552eb7c9
  > crc6450e07d0e82e86181
  > ...
```


So the agent developed new tools to parse it...

Why Python scripts were needed

1. Jadx doesn't decompile the protocol — searching `latest\sources` for `QtProtocol` / `AuthService` finds nothing useful (only unrelated Google/Sentry names).
2. Assemblies are .NET PE/IL, not Java DEX. Understanding them means reading metadata + method IL (via `dnfile`), not `.java` files.
3. There was no nice C# decompilation (no ready ILSpy/dnSpy output in the tree). The `tools\dump_*.py` / `disasm_*.py` scripts are a lightweight substitute: dump types, `ldstr`s, and IL for `Generate` / `Parse` / `OldHash` / `DeviceModel` / `TransparentData`.
4. Details like "checksum sums sync as 0, then write `0xAA`" only show up clearly in IL, not in string literals.

Those scripts target the extracted DLLs, e.g. `disasm_protocol.py` → `MGridtraqPL.dll`, `dump_swds_auth.py` → `SWDSCustomer.dll`, and write dumps under `tools\*.txt`.

	<code>adv_ctor_dump</code>	<code>txt</code>
	<code>adv_il_dump</code>	<code>txt</code>
	<code>api_proto_dump</code>	<code>txt</code>
	<code>auth_keys_dump</code>	<code>txt</code>
	<code>disasm_parse_auth</code>	<code>py</code>
	<code>disasm_protocol</code>	<code>py</code>
	<code>dump_adv_extras</code>	<code>py</code>
	<code>dump_adv_full</code>	<code>py</code>
	<code>dump_api21_scan</code>	<code>py</code>
	<code>dump_auth_adv</code>	<code>py</code>
	<code>dump_parse_scan</code>	<code>py</code>
	<code>dump_qt_protocol_detail</code>	<code>py</code>
	<code>dump_swds_auth</code>	<code>py</code>
	<code>dump_swds_auth2</code>	<code>py</code>
	<code>dump_transparent</code>	<code>py</code>
	<code>gen_frame_examples</code>	<code>py</code>
	<code>ldstr_hits</code>	<code>txt</code>
	<code>move_next_dump</code>	<code>txt</code>
	<code>protocol_dump</code>	<code>txt</code>
	<code>qt_detail</code>	<code>txt</code>
	<code>secret_refs</code>	<code>txt</code>
	<code>send_closures</code>	<code>txt</code>
	<code>swds_auth</code>	<code>txt</code>
	<code>swds_auth2</code>	<code>txt</code>

Now let's make a BLESPloit script for it

We'll start with simple "Observer" that would decode the advertisements, and a matching "Peripheral" for testing

Based on sample devices in [device-library](#) and [SKILL.md](#) create a device library entry consisting of observer role that decodes all the data in the advertisement, as well as sample peripheral that advertises it

Worked for 9m 46s

Created `device-library/vendors/quantam/` with observer + peripheral (no central).

Observer decode

From manufacturer AD (0xFF): VIN, unit type, mode, armed/snow/elite/debug/encrypted, VBat (x0.06), lat/lon, company id, raw MSD hex. Format branches on length 26 (extended) vs ≥5 (old).

Sample extended-old packet decodes as: `QT DEADBEEF · QT6_CAN · User · armed · snow · 12.60V · VIN 12345` (debug+encrypted).

Works on sim, confirmed also on real signals in Las Vegas

☰

LOCAL

REMOTE

RELAY

Filter 3

2 devices

Stop

-84

KARR

QT 8B10B3F4 · QT6_CAN · No_Sale · 13.32V · VIN 68976

Capture (this session) · no svc

Vehicle · Adv ~1426ms · 14s ago · Not bonded

Raw

Parsed

Decoded

KARR

protocol: KARR

vin: 68976

armed: false

mode_name: No_Sale

msd_hex: 7638397736de

unit_type_name: QT6_CAN

snow_mode: false

msd_len: 6

debug_mode: false

msd_name:

msd_present: true

mode: 3

latitude:

-94

KARR

QT D073D221 · QT6_CAN · Unknown · ... 44:B7:D0:73:D2:21

Capture (this session) · no svc

Vehicle · Adv ~1426ms · 23s ago · Not bonded

Raw

Parsed

Decoded

KARR

protocol: KARR

vin: 68976

armed: false

mode_name: No_Sale

msd_hex: 7638397736de

unit_type_name: QT6_CAN

snow_mode: false

msd_len: 6

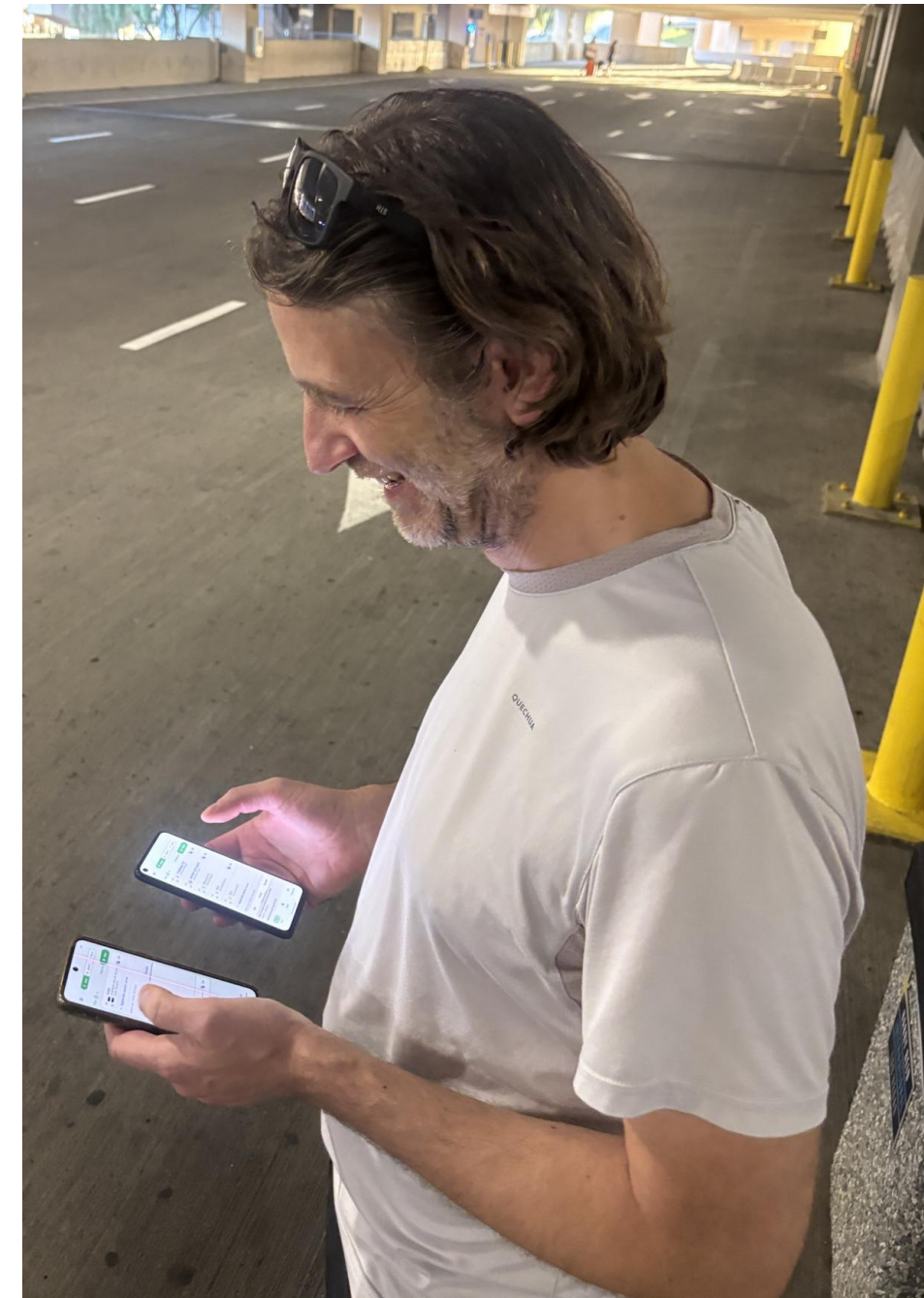
debug_mode: false

msd_name:

msd_present: true

mode: 3

latitude:



How about “central” script to control the car?

extend the peripheral to include GATT (ble.json), write also sample “central” menu script to lock/unlock etc

Worked for 7m 3s

Extended `vendors/quantam` with GATT + central control.

Flow

1. Connect → subscribe TDC
2. **Authenticate (sim)** → `ProvideAuth` (32×00; sim accepts any) → `AuthSuccess`
3. **Lock / Unlock** (also silent / secure) → `AckOk`, updates `vars.armed` and advertising armed flag

Real challenge-response hashing is not implemented (`accept_any_auth: true` in `vars.json`). Frames match the app layout, e.g. lock `aa00010bf4`, unlock `aa00010cf3`.

Turns out the keys are on the server...

What are the real hashing and encryption keys, are they in the application code, retrieved from the cloud, ... ?

Worked for 10m 24s >

BLE secrets are not hardcoded. Hash/AES material comes from the Quantam/Gridtraq cloud (or cached copies). The app only hardcodes HTTP API "app keys" used to call that backend.

Risk assessment according to vendor

“THE VULNERABILITY DESCRIBED IN THE RESEARCH IS
HIGHLY COMPLEX AND PRESENTS A LOW RISK TO CUSTOMERS
UNDER REAL-WORLD CONDITIONS.
NEVERTHELESS, WE RESPONDED PROMPTLY AND DEVELOPED A
FIRMWARE UPDATE TO ADDRESS THE ISSUE.”

ACRISURE PG

<https://www.youtube.com/watch?v=FwA3CuJxbk4>

...but LLM + easy to use app may shift the risk impact significantly.

Fast pair: let's make it “magically” easy



Fast Pair

Introduction

The Google Fast Pair Service (GFPS) utilizes Bluetooth Low Energy (BLE) to discover nearby Bluetooth devices without using significant phone battery, enabling “magical” scenarios based on device proximity.

Features ⇄

“magical” easy

GFPS is aimed at facilitating the pairing of Bluetooth and BLE devices, such as speakers, headphones, car kits, mice and keyboards, with as little user interaction required as possible. By implementing the following spec, Google will continue to release additional features that build upon it. This includes:

1. Displaying a half page notification when the device is in pairing mode to facilitate easy initial pairing. Additionally companion apps are easily marketed to users.

<https://developers.google.com/nearby/fast-pair/specifications/introduction>

Fast Pair



Spam (Google Fast Pair, Apple Quick Pair)

Broadcast advertisements simulating many pairing devices.

User unable to control the phone (popups).

Invalid packets crashed iPhone.

Spotted at Defcon 2023, then many implementations (Python, ESP32, Flipper, Android, ...)



<https://ca.finance.yahoo.com/news/hacking-device-flipper-zero-spam-184119946.html>

BTW: BLESPlo.it detecting flippers

LOCAL

REMOTE

RELAY

Filter ▼

Scan 🔍

-71

Flipper Oltot

White

80:E1:26:47:F7:83

Filter

Reset

Search

flip

×

RSSI

-130 ... 0 dBm

Favorites

Connectable

Bonded

Hide Nearby, beacon, tracker

Advertisement

✓ Any

Legacy

Extended

-70

Flipper Oltot

White

80:E1:26:47:F7:83

Capture (this session) · no svc

Generic · Adv ~1957ms · 1s ago · Not bonded

Decoded

Parsed

Raw

Oltot

0x02 Incomplete List of 16-bit Service Class UUIDs

0x3082 - Unknown Service UUID

0x0A TX Power Level

0 dBm

-84

Flipper Oltot

White

80:E1:26:47:F7:83

Capture (this session) · no svc

Generic · Adv ~1823ms · Not bonded

Decoded

Parsed

Raw

Source: Flipper Zero (flipper_zero)

Flipper Zero

shell_color: White

← Observer role(preview)

```
-- source:
https://github.com/k3yomi/Wall-of-Flippers/
local SHELL_BY_UUID = {
  ["3081"] = "Black",
  ["3082"] = "White",
  ["3083"] = "Transparent",
}

local function norm_uuid16(cell)
  if type(cell) == "string" then
    return cell:gsub("^0x", ""):upper()
  end
  return nil
end

local function detect_shell(input)
  local t = input.service_uuids_16
  if not t then return nil end
  local i = 1
  while true do
    local u = norm_uuid16(t[i])
    if not u then break end
    local shell = SHELL_BY_UUID[u]
    if shell then return shell end
    i = i + 1
  end
  return nil
end

function parse(input)
  local shell = detect_shell(input)
  if not shell then
```


Wigle: search flippers in BT data

wigle.net/search#btSearch

General Search

WiFi/Cell Detail

Bluetooth Search

Average Location - Address

Num: 141

Street: West Jackson Boulevard

City: Chicago

Region: IL

Country: US

Postal: 60604

Average Location - Coordinates

Lat: 47.25264

to: 47.25265

Lon: -87.256243

to: -87.256244

Search Radius Tolerance(+/- degrees): 0.010

Network Characteristics

Last Updated: 20010925174546

Minimum data quality⁰: 0

NETID/MAC: 80:E1:26

Network Name (exact match): foobar

Network Name (wildcards¹: % and _): foobar%

☐ Only Networks I Was the First to Discover

Minimum Manufacturer ID: 2504

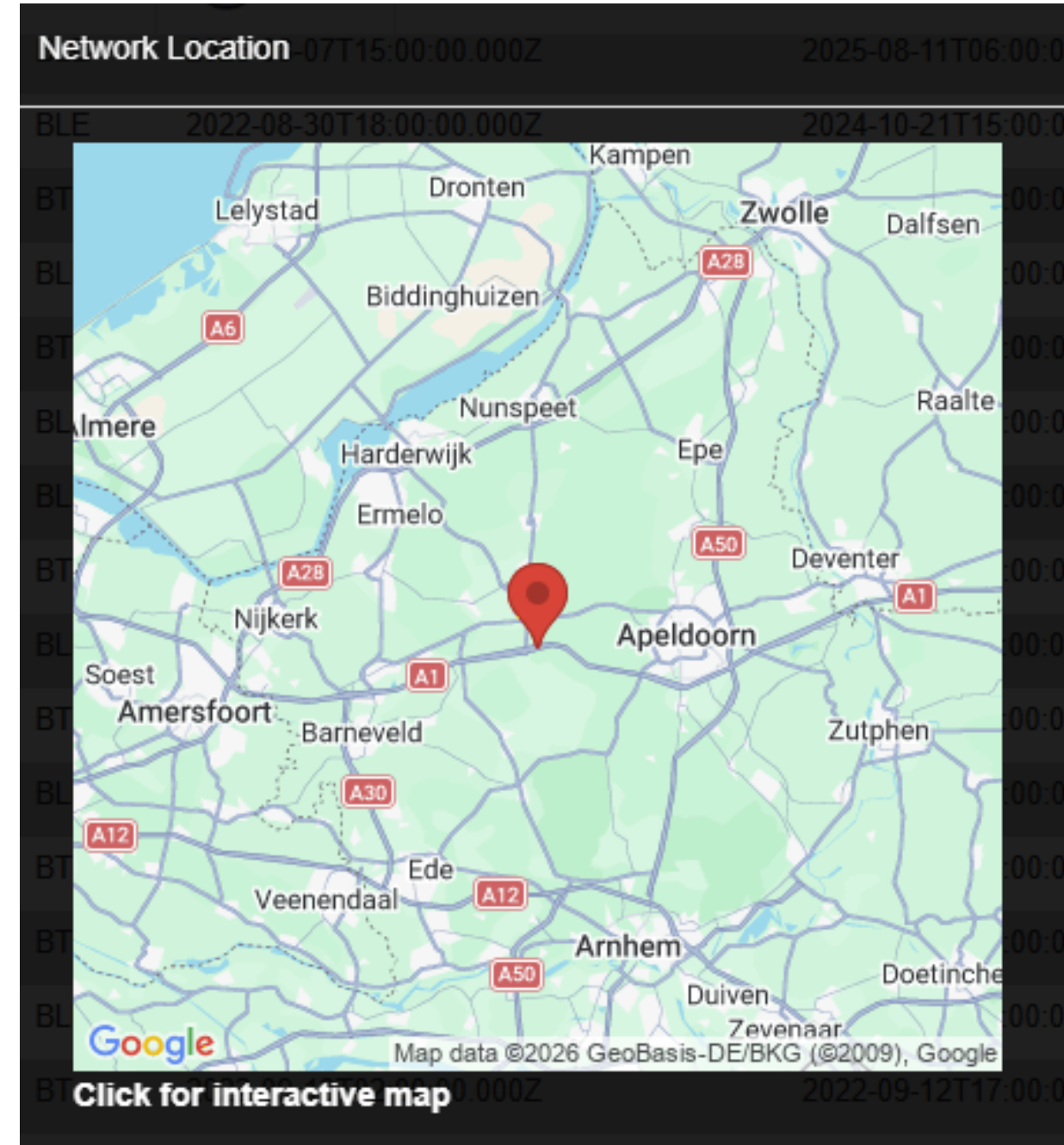
Maximum Manufacturer ID: 2505

Query

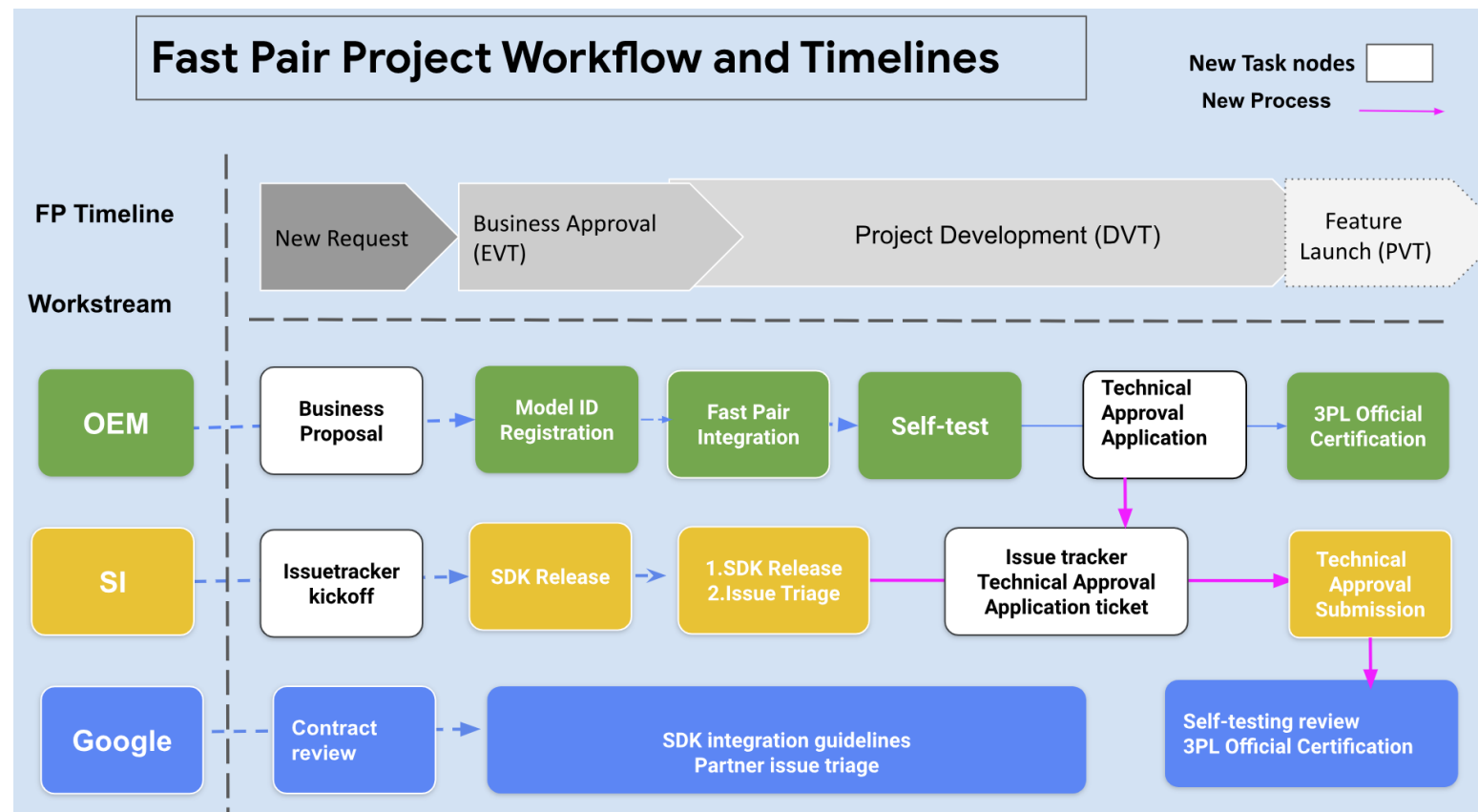
Reset

⁰ 0-7 Product of number of observers and observations.
¹ '%' means zero-or-more characters, '_' means a single character.

Map	Net ID	Name	Type	First Seen	Most Recently	Type	Est. Lat	Est. Long	QoS
map	80:e1:26:00:00:0f	Flipper Mygurn	BLE	2026-04-28T18:00:00.000Z	2026-05-15T23:00:00.000Z		43.75499344	-79.36147308	5
map	80:e1:26:00:00:28	Flipper Ranp0I0	BLE	2024-02-12T17:00:00.000Z	2024-09-12T06:00:00.000Z		59.99482346	30.29911232	0
map	80:e1:26:00:00:30	0	BLE	2023-12-01T09:00:00.000Z	2026-05-18T05:00:00.000Z		46.37207031	6.4782939	0
map	80:e1:26:00:00:31	1	BLE	2025-08-07T15:00:00.000Z	2025-08-11T06:00:00.000Z		36.13418198	-115.15818787	6
map	80:e1:26:00:00:41	A	BLE	2022-08-30T18:00:00.000Z	2024-10-21T15:00:00.000Z		36.00226974	-78.91191101	1
map	80:e1:26:00:00:43	zzRogue Ch0rnke	BT	2022-09-04T17:00:00.000Z	2022-12-31T18:00:00.000Z		39.98361969	-105.10372162	2
map	80:e1:26:00:00:44	zzRogue Dy3t0nix	BLE	2022-08-17T14:00:00.000Z	2022-08-18T03:00:00.000Z		40.94362259	-74.11135864	0
map	80:e1:26:00:00:45	zzRogue Elg3m	BT	2022-08-17T09:00:00.000Z	2022-09-10T16:00:00.000Z		40.63731766	-74.33651733	0
map	80:e1:26:00:00:46	Flipper Fhelve	BLE	2022-08-26T08:00:00.000Z	2022-10-18T19:00:00.000Z		-37.78333282	145.27186584	4
map	80:e1:26:00:00:48	zzRogue He1ckune	BLE	2003-01-09T13:00:00.000Z	2025-06-23T01:00:00.000Z		51.1443367	17.5138588	0
map	80:e1:26:00:00:48	zzRogue Hakol	BT	2003-01-09T13:00:00.000Z	2024-02-28T14:00:00.000Z		40.844841	-74.11954498	2
map	80:e1:26:00:00:49	I	BLE	2022-08-27T12:00:00.000Z	2025-11-22T06:00:00.000Z		52.25391769	0.030979	0
map	80:e1:26:00:00:49	zzRogue Irem4x	BT	2022-08-27T12:00:00.000Z	2022-11-20T14:00:00.000Z		28.5953331	-81.30431366	2
map	80:e1:26:00:00:4b	K	BLE	2022-08-24T13:00:00.000Z	2026-04-26T22:00:00.000Z		39.40671921	-76.6006012	6



The Fast Pair Adoption Process



<https://developers.google.com/nearby/fast-pair/landing-page>

<https://developers.google.com/nearby/fast-pair/system-integrator-roles-and-responsibilities>

Google Fast Pair: involved parties

 defines protocol and certification process

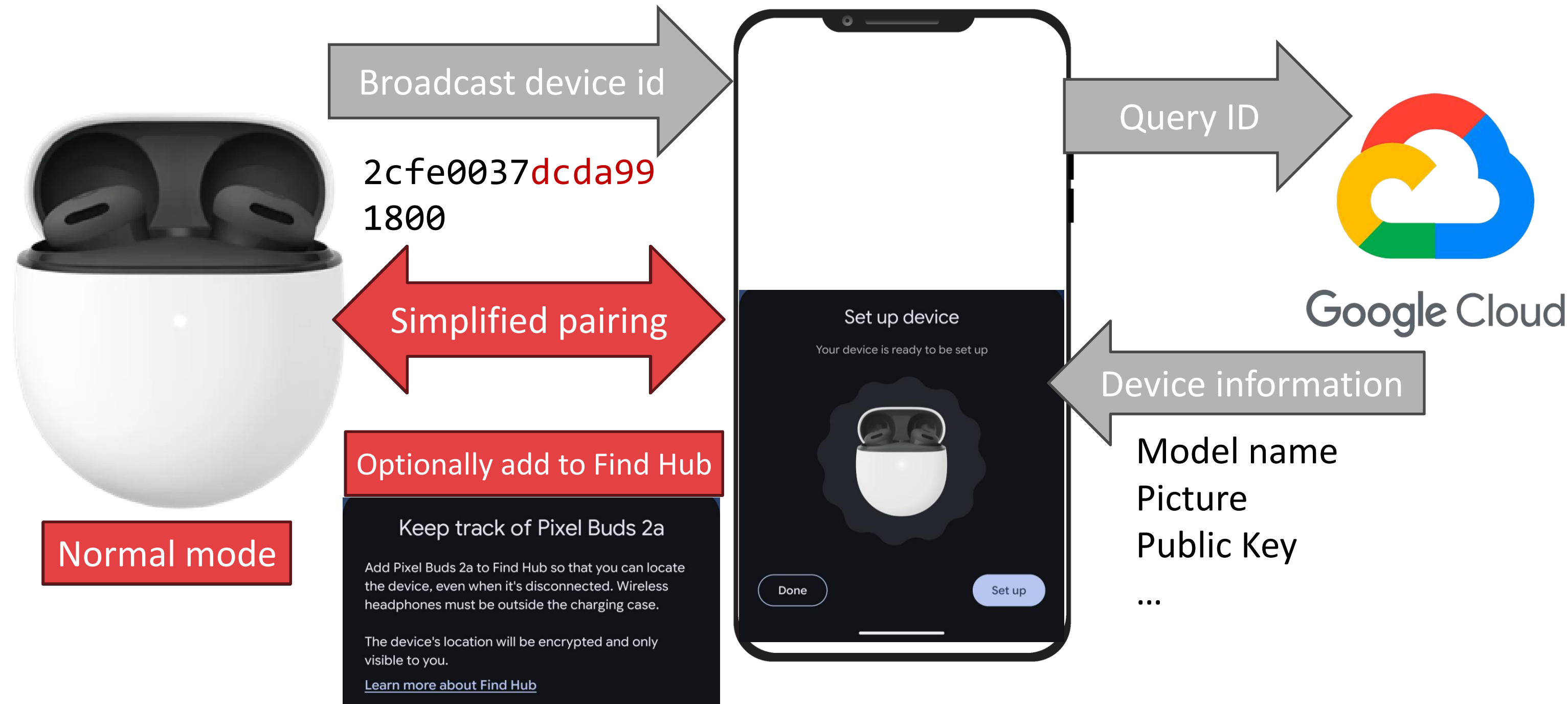
 applies for certification

  REALTEK
 partner (OEM)




3rd Party Lab Certification

Fast Pair



WhisperPair

Hijacking Bluetooth Accessories Using Google Fast Pair

Is my device vulnerable?

About

Read the paper

Cite

Watch the video



Bluetooth hijacking

Many Bluetooth accessories do not implement Google Fast Pair correctly, enabling an attacker to forcefully pair with a vulnerable accessory.

Learn more



Location tracking

If an accessory has never been paired with an Android device, an attacker may be able to track its location using Google's Find Hub Network.

Learn more



Cross-ecosystem impact

Since the Fast Pair functionality of an accessory cannot be disabled, users outside the Android ecosystem are also vulnerable to WhisperPair.

Learn more

<https://whisperpair.eu>

Google

SONY

BOSE

Jabra^{GN}

Marshall
logitech

JBL

SONOS

ONEPLUS

REDMI

hp

NOTHING

ANDY GREENBERG

LILY HAY NEWMAN

SECURITY

JAN 15, 2026 7:00 AM

Hundreds of Millions of Audio Devices Need a Patch to Prevent Wireless Hacking and Tracking

Flaws in how 17 models of headphones and speakers use Google's one-tap Fast Pair Bluetooth protocol have left devices open to eavesdroppers and stalkers.

<https://www.wired.com/story/google-fast-pair-bluetooth-audio-accessories-vulnerability-patches/>

WhisperPair in BLESPloit

LOCAL

REMOTE

RELAY

Filter

Scan

-43

Unknown

Fast Pair · L 70%

C3:66:29:13:69:DD

Capture (this session) · no svc

Fast pair · Adv ~405ms · 0s ago · Not bonded

Decoded

Parsed

Raw

Source: Google Fast Pair (fast_pair)

Google Fast Pair (account advertisement)

fp_battery_value_count_nibble: 1

salt_field_header_octet: 17

fp_service_data_hex: 00907c5c7d102224617ad811f21446

fp_battery_header_octet: 20

fp_battery_left_charging: false

-42

WH-CH720N

Sony WH-CH720N

70:04:1D:DA:7A:8A

Stop

Unknown

58:A0:BC:B1:94:78

Stop

Scan

Graph

Active connections

Automatic

LE-WH-CH720N

CA:40:62:43:E8:E2

Sony WH-CH720N

RSSI -54 dBm

Last seen Scanning is off

Stay connected

On: after a successful quick script, switches to Manual and keeps this peripheral session. Off: disconnect and go back (device list).

Capture (this session) · no svc

Fingerprint

Google Fast Pair (account advertisement)

Fast Pair model ID

Search local catalog for devices with a known anti-spoofing key

Scripts

Check WhisperPair vulnerability

Matched dynamic script.

Check Fast Pair model

Matched dynamic script.

Read device info

Matched dynamic script.

C3:66:29:13:69:DD · WhisperPair confirmed

C3:66:29:13:69:DD

Fast Pair · L 70%

Open image URL

RSSI -50 dBm

Last seen 0s ago

Stay connected

On: after a successful quick script, switches to Manual and keeps this peripheral session. Off: disconnect and go back (device list).

Capture (this session) · 2/38

Check WhisperPair vulnerability: Done

Enriched:

kb_notify_hex=ef08cf0fb54ba5374030b695d367f9dc,

kb_flags=0, kb_whisper_pair_confirmed=true,

gatt_mac=C3:66:29:13:69:DD,

kb_probe_verdict=whisper_pair_confirmed,

kb_write_ok=true, display_info=Fast Pair · Sony WH-CH720N · 0xF42FFC,

enrichment_display_name=C3:66:29:13:69:DD · WhisperPair confirmed

Run log

Hide

Connecting

LOCAL

REMOTE

RELAY

Filter

Scan

-52

Unknown

Fast Pair · L 70%

C3:66:29:13:69:DD

Nearby Info

7E:FD:3C:32:C5:3A

-51

WH-CH720N

Sony WH-CH720N

70:04:1D:DA:73:4A

-72

LIGHT BMO

state=off R=219 G=219 B=72

24:EC:4A:30:79:DA

-74

Unknown

38:D1:E6:87:65:FA

-89

Unknown

0F:FA:40:98:DC:9E

Stop

Scan

Graph

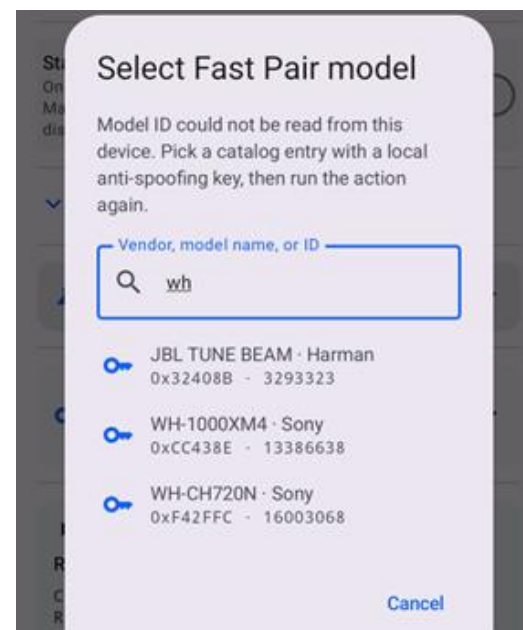
Active connections

WhisperPair check in BLESPloit

The check performs an encrypted write to the Fast Pair characteristic, based on devices' public key.

BLESPloit contains several keys from WhisperPair tool (<https://github.com/KULeuven-COSIC/WhisperPair/>)

If device does not return ID, select manually:



Fingerprint


Google Fast Pair (account advertisement) · Product image

Fast Pair model ID


0xF42FFC (16003068) · catalog search

► Check WhisperPair vulnerability: Done

Run log [Hide](#)

Connecting...
Running Check WhisperPair vulnerability...
Executing Lua...
Fast Pair KBP: write 80 octets
(enc+ephemeral pub), gatt=CA:40:62:43:E8:E2
Fast Pair KBP: notify (hex) =
b6c7b01546e95bb15efba544b4c337d6
RESULT: PASS – WhisperPair confirmed
(Provider returned KBP NOTIFY).
Disconnecting...

Scripts

Supply chain

Google

defines protocol and certification process

SONY

applies for certification

MEDIATEK
AIROHA

REALTEK

partner (OEM)

Qualcomm



3rd Party Lab Certification

Airoha



Component used in major brands headsets: Sony, Bose, Marshall, JBL, Jabra, ...

Unauthenticated access to internal memory via proprietary protocol

- Extract encryption keys, other data
- Hijack calls
- Remote control voice assistants
- Arbitrary code execution

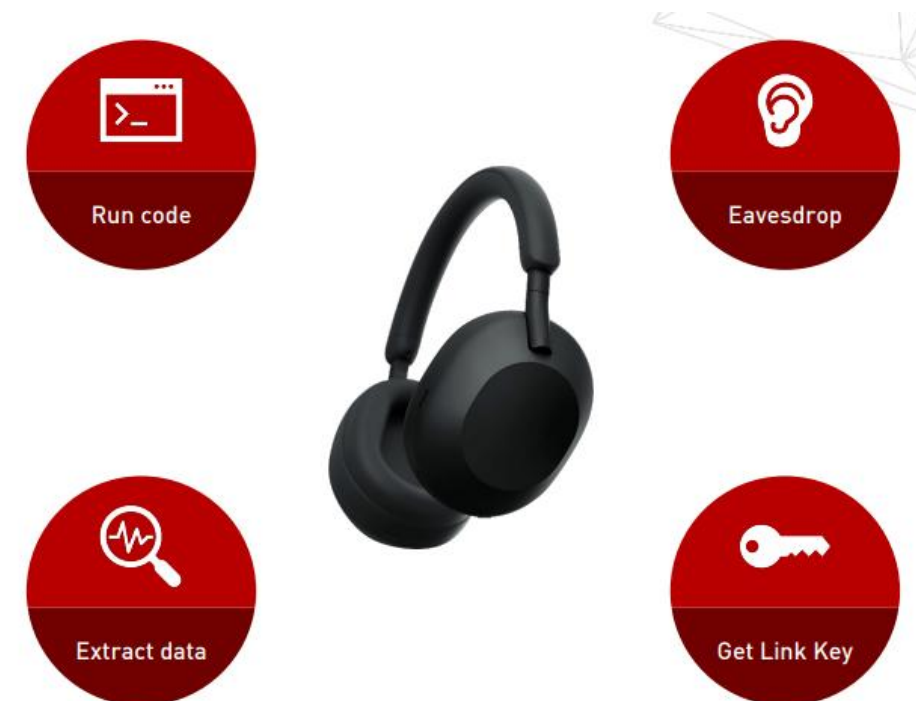
More info:

<https://insinuator.net/2025/12/bluetooth-headphone-jacking-full-disclosure-of-airoha-race-vulnerabilities/>

Zero-day: Bluetooth gap turns millions of headphones into listening stations

The Bluetooth chipset installed in popular models from major manufacturers is vulnerable. Hackers could use it to initiate calls and eavesdrop on devices.

<https://www.heise.de/en/news/Zero-day-Bluetooth-gap-turns-millions-of-headphones-into-listening-stations-10460704.html>



https://troopers.de/downloads/troopers25/TR25_Headphone_Jacking_FBNB8Y.pdf

Airoha BLESPlo.it: quick action or manual



2:51

C3:66:29:13:69:DD · Airoha build version read confirmed

C3:66:29:13:69:DD

Fast Pair · L 70%

RSSI
-46 dBm

Stay connected
On: after a successful quick pairing. Off: disconnect and go back (device list).

▼ Capture (this session) · 1/38

► Check Airoha vulnerability: build version (GATT): Done

Enriched: research_ref=CVE-2025-20700 CVE-2025-20700
race_notify_prefix_hex=055b7300081e006d785f65766b0000000004d54323832325f534792d455236395f6d647231345f6334327370000000, gatt_mac=C3:66:29:13:69:DD, race_build_vulnerability_confirmed=true, race_variant=sony, race_build_verdict=vendor_debug_confirmed, race_build_decoded=mt2822x_evk | MT2822_SDK_Sony-ER69_mdr14_c42sp_1 | 18:58:55 GMT +08:00, enrichment_display_name=C3:66:29:13:69:DD build version read confirmed

Run log [Hide](#)

Connecting...
Running Check Airoha vulnerability: build version (Sony GATT)...
Executing Lua...
RESULT: PASS — Airoha build-info NOTIFY confirmed (vendor debug responding to cmd 0x1E08, variant sony).
Build version (decoded ASCII): mt2822x_evk | MT2822_SDK_Sony-ER69_mdr14_c42sp_1 | 18:58:55 GMT +08:00

Annotations for
proprietary UUID



LOCAL REMOTE RELAY

LE-WH-CH720N
C3:66:29:13:69:DD -46 dBm

Fingerprint
Google Fast Pair (account advertisement)

▼ Capture (this session) · 1/38

RACE — Airoha vendor (Sony layout)
DC405470-A351-4A59-97D8-2E2E3B207FBB

RACE TX (Sony) — host → device
BFD869FA-A3F2-4C2F-BCFF-3EB1EC80CEAD

RACE RX (Sony) — device → host
2A6B6575-FAF6-418C-923F-CCD63A56D955

Client Characteristic Configuration
0x2902

Value: 01 00 · Notify

Device Information Service
0x180A

FE59BFA8-7FE3-4A05-9D94-99FADC69FAF8

Manual Scripts Logs

1. subscribe

2. write

Write

Hex Text

Get build version Read flash 256 B @ 0x0810

Hex: pairs or spaces (e.g. 01 00 ab).

05 5a 02 00 08 1e

Cancel Send

RACE RX (Sony) — device → host
2A6B6575-FAF6-418C-923F-CCD63A56D955

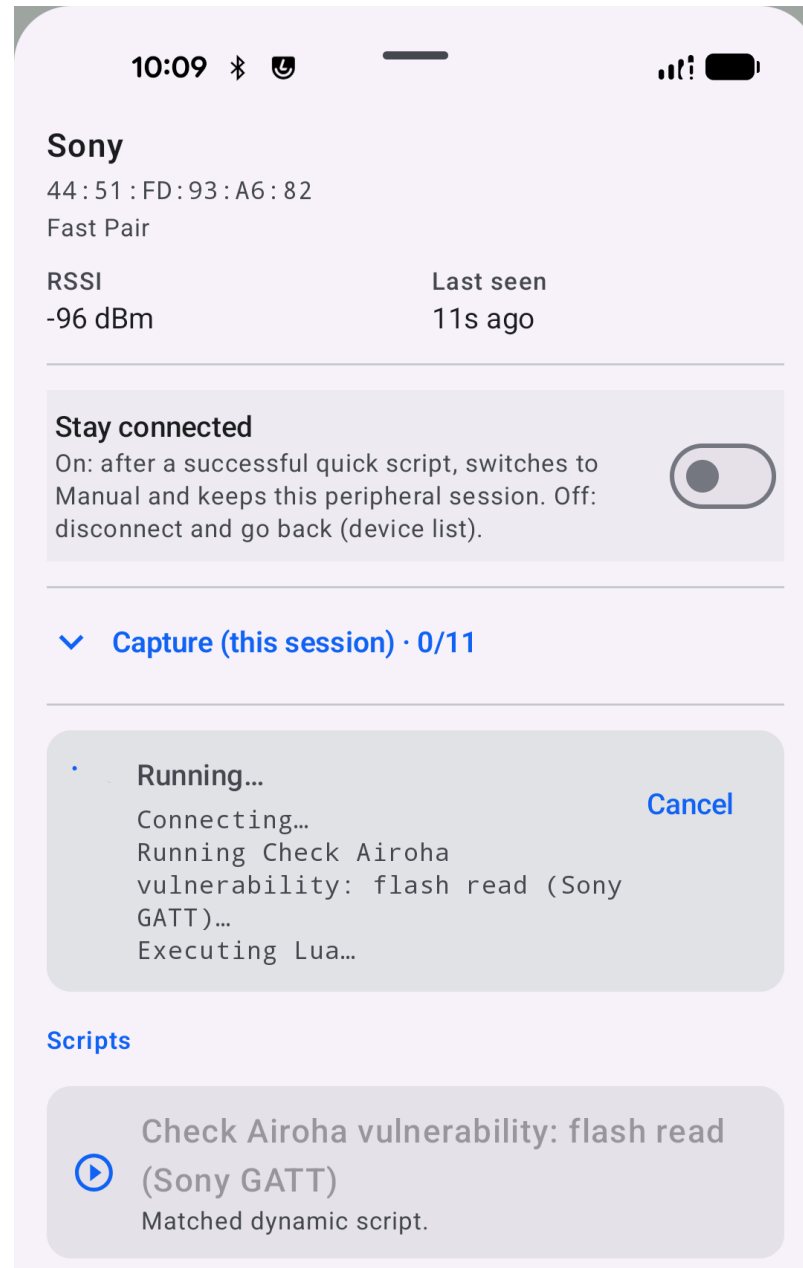
[smt2822x_evkMT2822_SDK_Sony-ER69_mdr14_c42sp_12024/09/18 18:58:55 GMT +08:00

Value: dr14_c42sp_12024/09/18 18:58:55 GMT +08:00

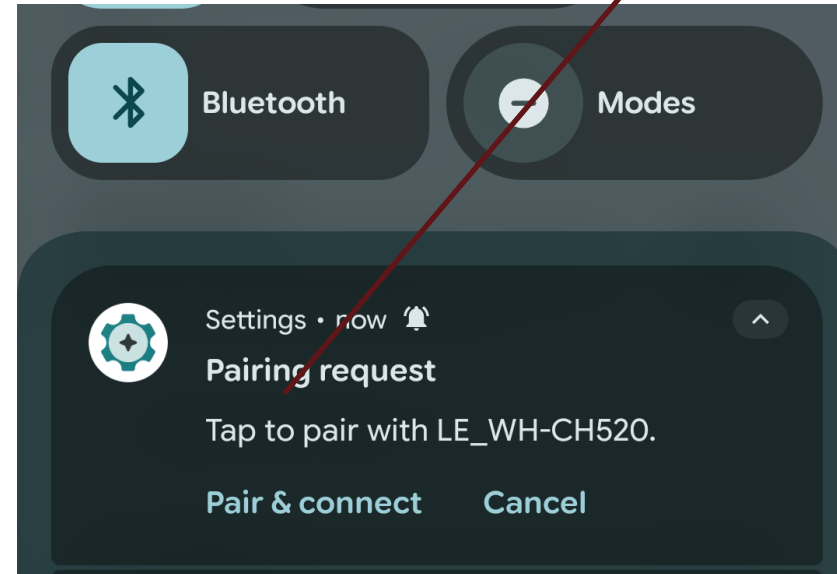
Client Characteristic Configuration
0x2902

Value: 01 00 · Notify

Patched Sony headset



Pairing enforced on attempt to access Airoha characteristic



► Check Airoha vulnerability: build version (Sony GATT): Done

Enriched: race_build_verdict=ble_subscribe_failed, race_build_vulnerability_confirmed=false

Run log [Hide](#)

```
Connecting...
Running Check Airoha vulnerability: build
version (Sony GATT)...
Executing Lua...
RESULT: FAIL – could not enable NOTIFY on
RACE RX (check Client Characteristic
Configuration / subscription). variant
sony.
Disconnecting...
```



EXTERNAL ESP32

BLESPlo.it: ESP32 “companion”

- Connect via WiFi or USB-C ethernet
- Works from app or standalone www
- Scan/connect remotely
- Get more low-level details on packets
- Simulate any device (optionally with display/buttons)
- Fully control the pairing parameters
- Relays and more coming next



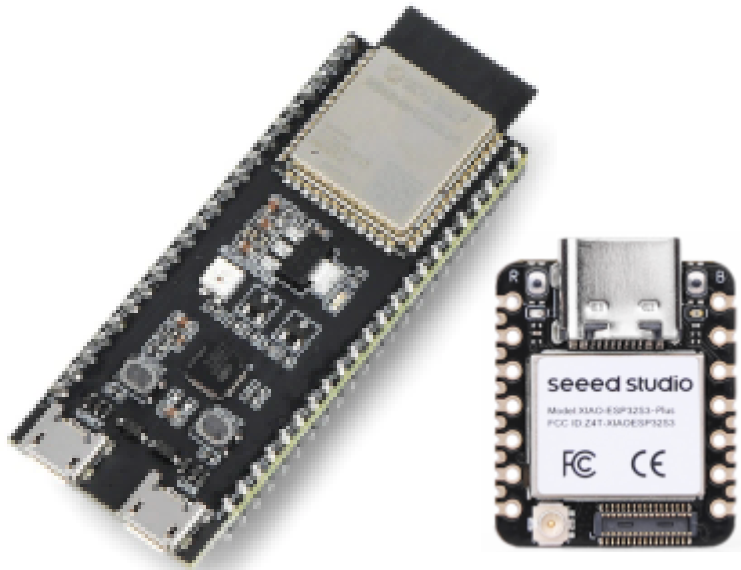
Currently supported hardware (more to come)

Any ESP32S3

No need for PSRAM

No display (simulation
status in web/mobile UI)

1M for devices



Lilygo T-QT-PRO

S3FN4R2 - 2M PSRAM

4M flash

(NOT S3FN8!)

1M for devices



M5Stack M5StickS3

8M PSRAM

8M flash

5M for devices



Flashing BLESPloit firmware

Firmware can be flashed using:

- Android application: connect via USB-C directly to your phone.
- Web flasher via browser (should work on any laptop)
- Manual command-line (e.g. esptool.py)

Image versions

- “merged” – overwrite the whole memory (reset settings, ...) – first flash
- “application” – just update the app and keep settings

More information:

<https://blesplo.it/docs/esp32/flashing/>

Android

☰

LOCALREMOTERELAY

● Not connected

Connect via WebSocket...

Connect via USB Ethernet (192.168.5.1)

Plug in the BLESPlloit ESP32 via USB. Wait a few seconds for the USB network interface to appear.

Debug

Connect USB serial console

Flash firmware

☰

LOCALREMOTERELAY

● Not connected

Flash firmware (USB)

1 Board2 Mode3 Confirm4 Flash

Select target board

M5Stack M5StickS3
ESP32-S3 · 8 MB PSRAM built-in

LilyGo T-QT Pro
ESP32-S3 · 2 MB PSRAM built-in

Bare ESP32S3 (must be S3!)

BackNext →

Scan

Connected

Sim

Console

Flash firmware (USB)

BoardModeConfirmFlash

Syncing...

Syncing... (attempt 1 of 8)

Allow BLESPlloit to access USB JTAG/serial debug unit?

CancelOK

Flash firmware (USB)

BoardModeConfirmFlash

Writing firmware...

24%

Writing... 437 / 1786 KB

Flash firmware (USB)

BoardModeConfirmFlash

✓

Firmware flashed successfully

Restart the device, access it via USB console or Wi-Fi (first time: setup may be needed).

Done

Connecting to ESP32

1. Wifi (2.4GHz only)

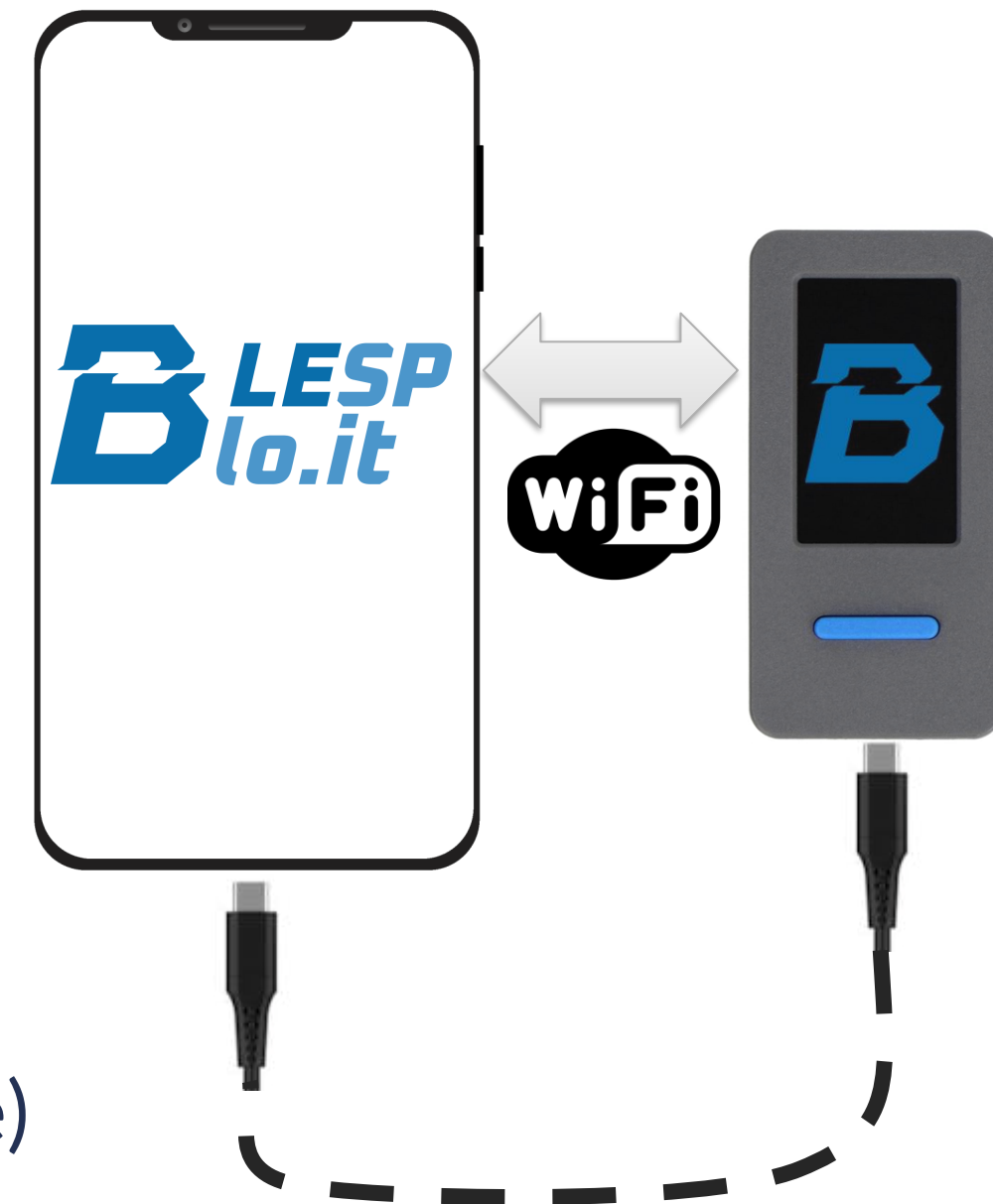
- After flashing it starts in AP mode, ssid: BLESPlo.it_AABB, pass: BLESPlo.it
- Possible to adjust (STA, other AP, ...)

2. USB ethernet

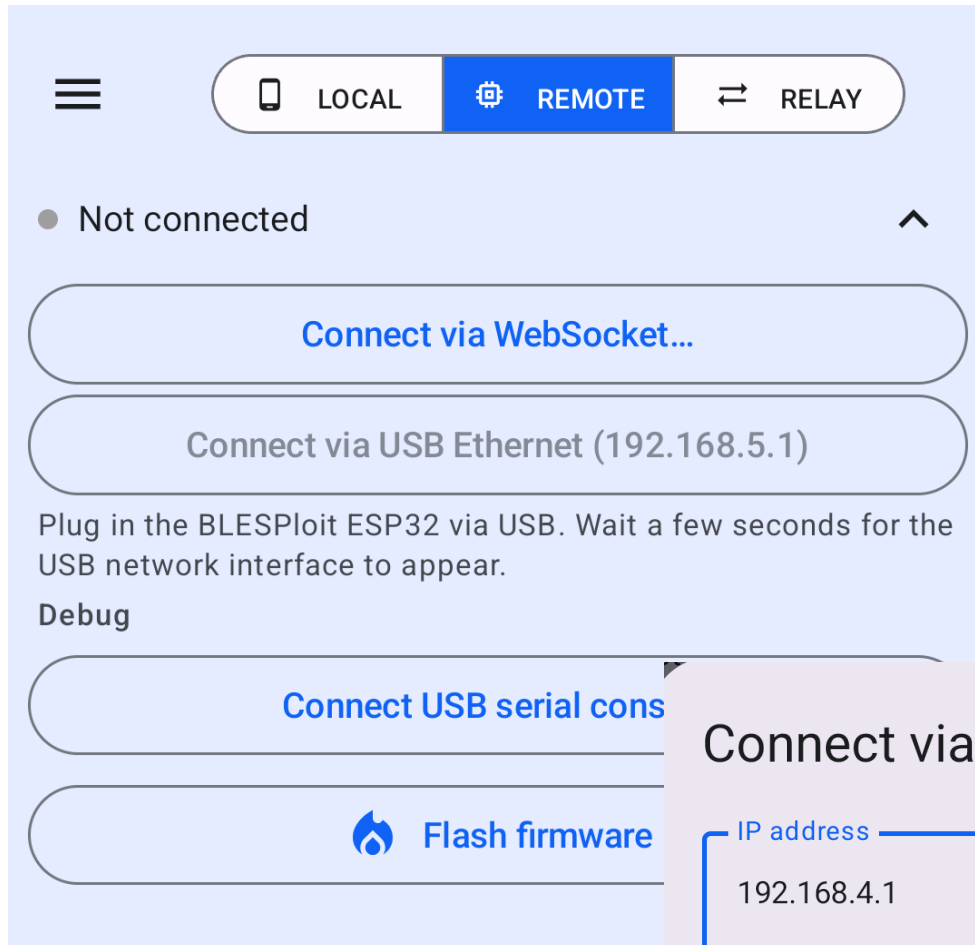
- Android, iPhone, laptop, ...

3. USB serial debug console (second USB device)

- From the Android phone directly or from laptop (115200 baud)



Wifi connection from the app



Connect via WebSocket

IP address

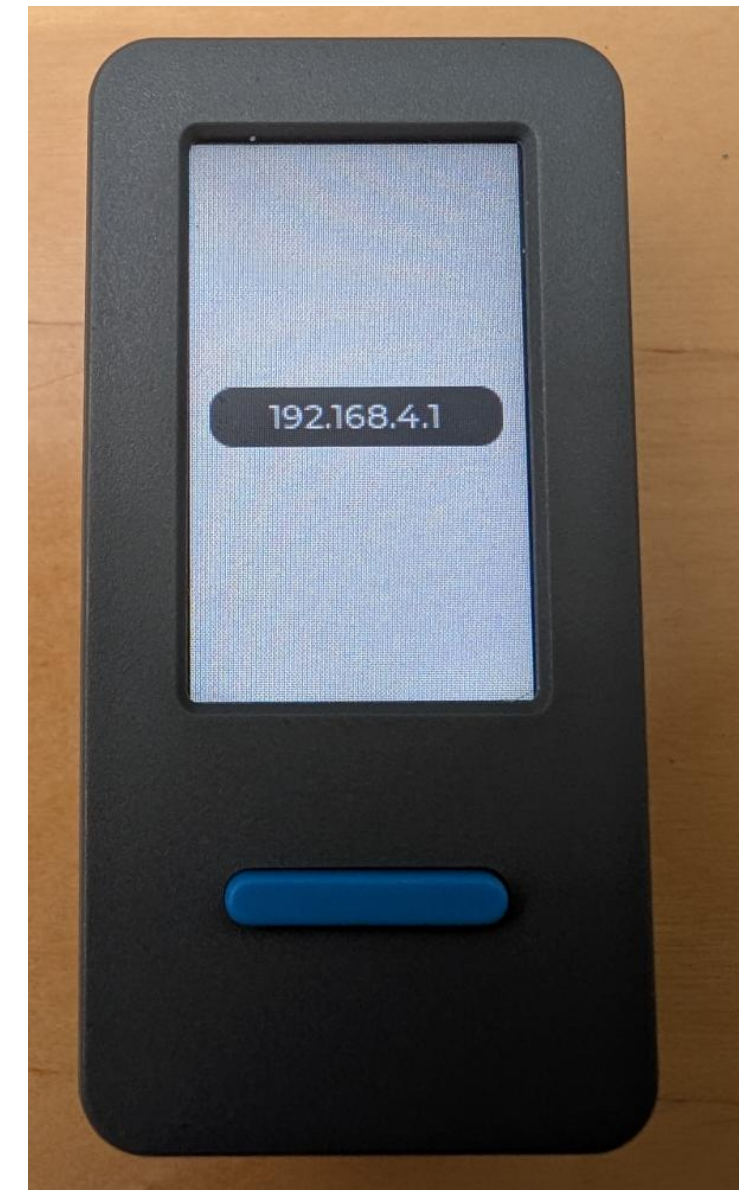
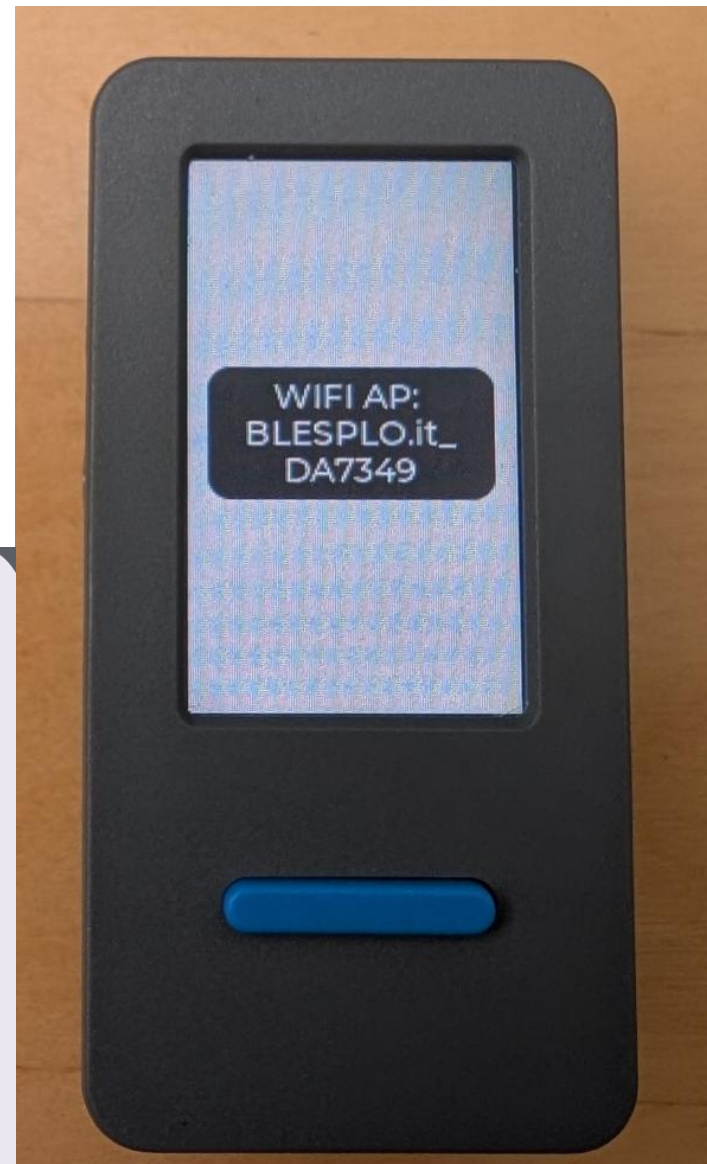
192.168.4.1

Port

80

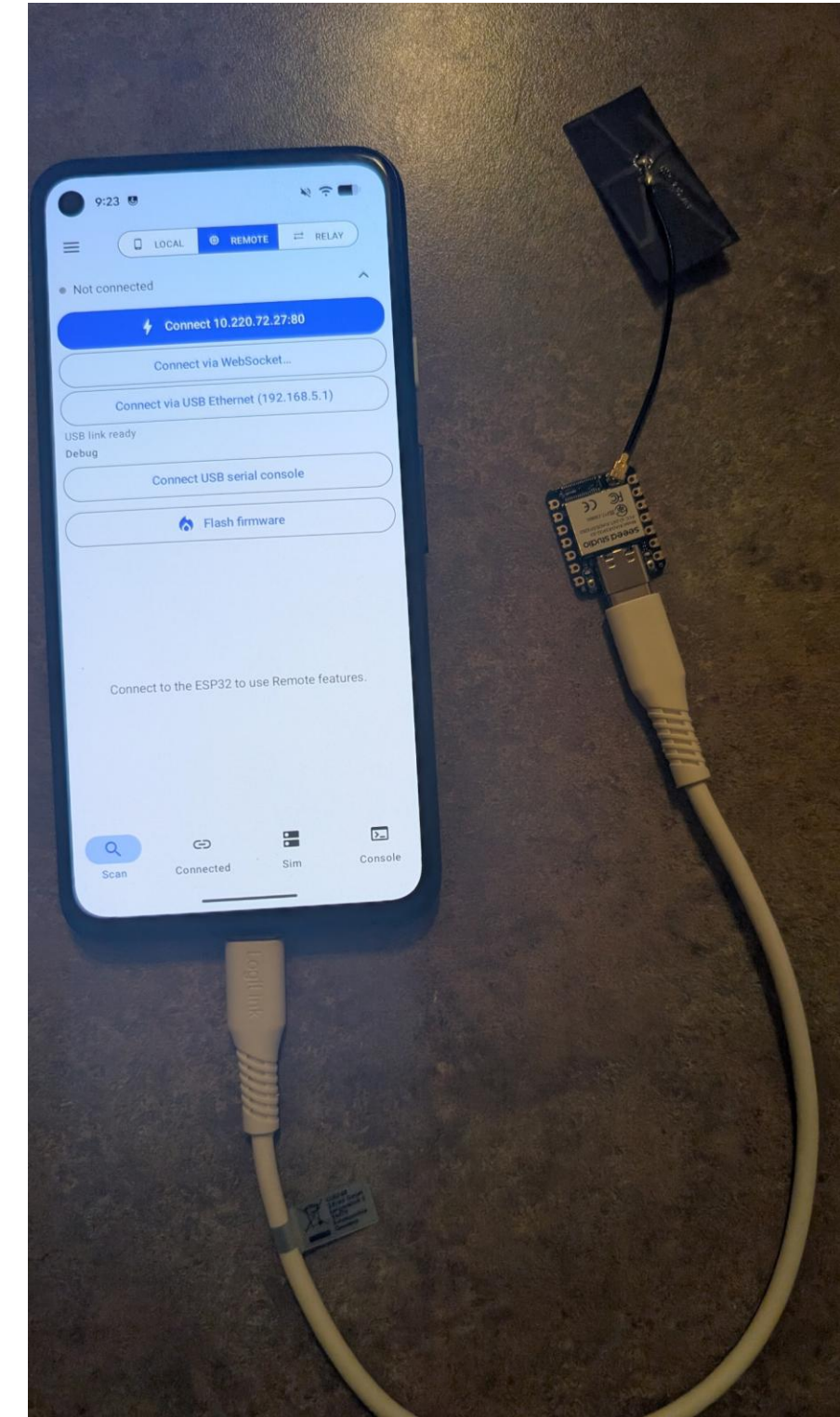
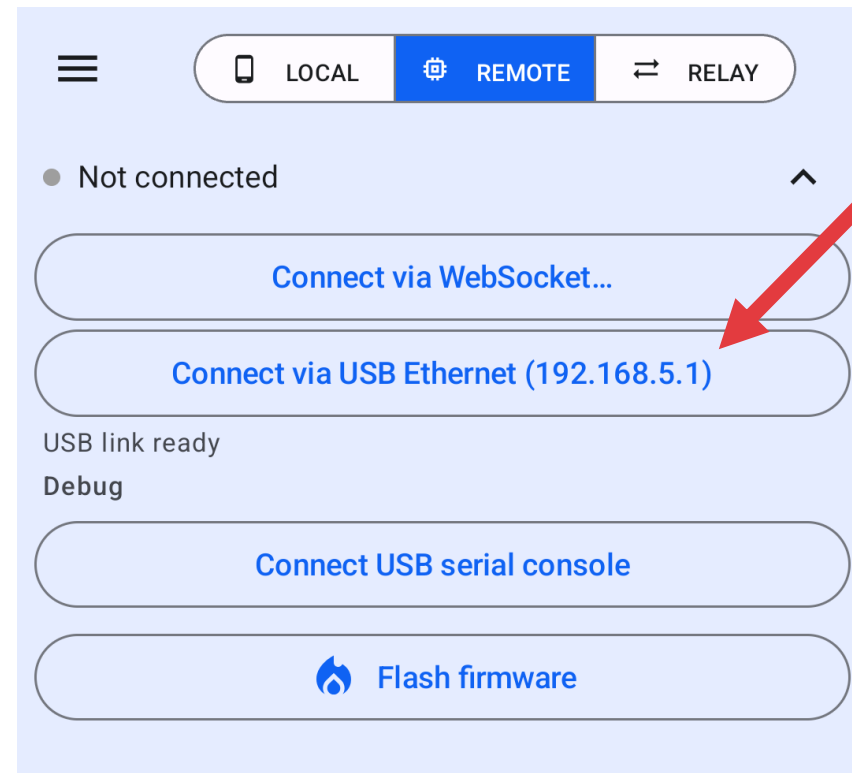
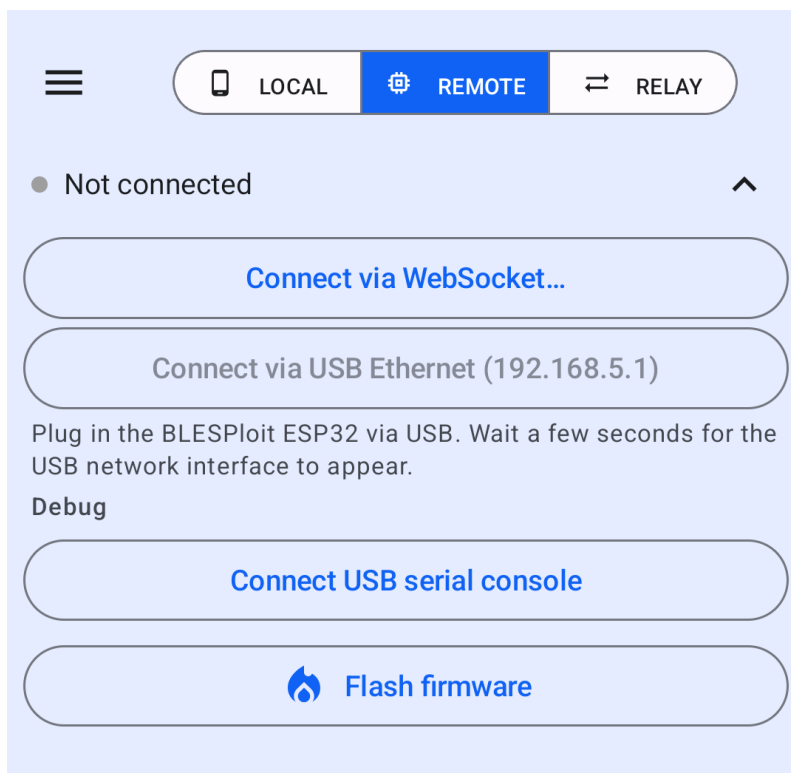
Cancel

Connect



USB Ethernet

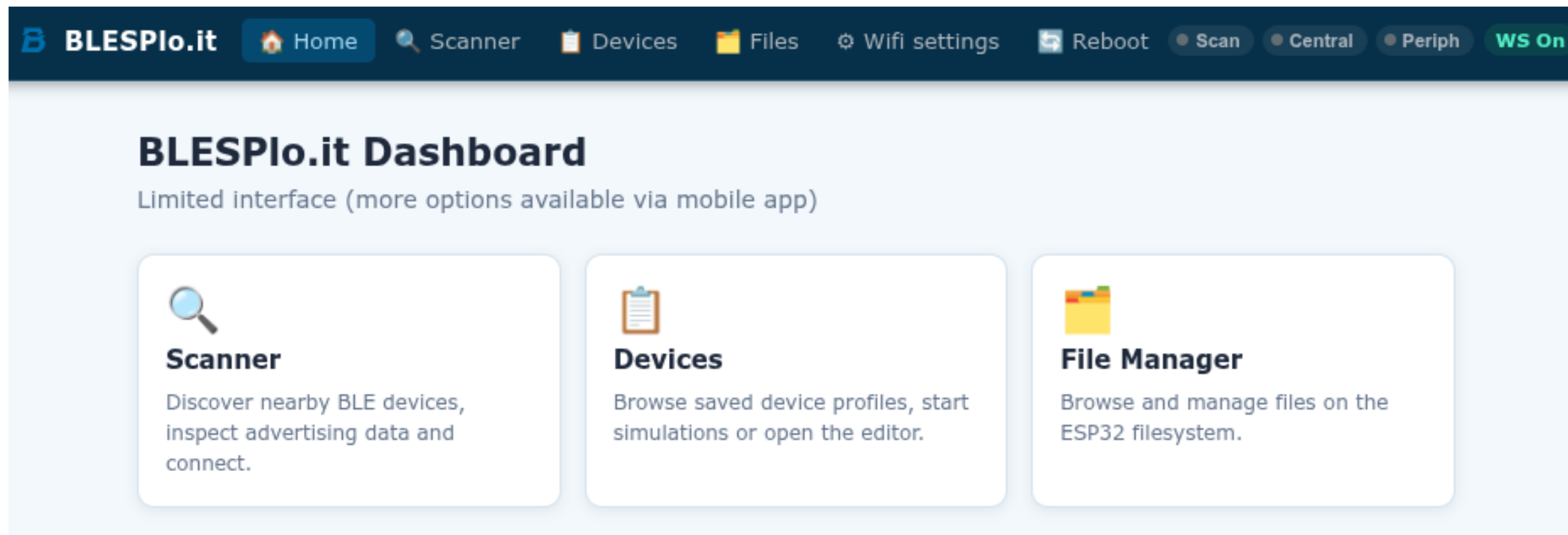
WiFi takes significant amount of power and memory. Preferably connect via USB-C Ethernet (directly to the phone)



Direct web interface

You can access ESP32 directly (web interface) – also from your laptop

- Wifi AP: <http://192.168.4.1> (or your IP in STA)
- Local USB: <http://192.168.5.1>



Web interface: possible to use from laptop (wifi or USB eth)



iPhone USB

USB-C model: connect directly

Lightning: adapter needed, look for “Lightning USB OTG Camera”

Nope



Careful, may overheat



Most stable



Super-powers for iPhone

iOS limits BLE access for applications

- No “raw” advertisements
- Some fields stripped (e.g. beacon, nearby, ...)
- Limited services after connecting

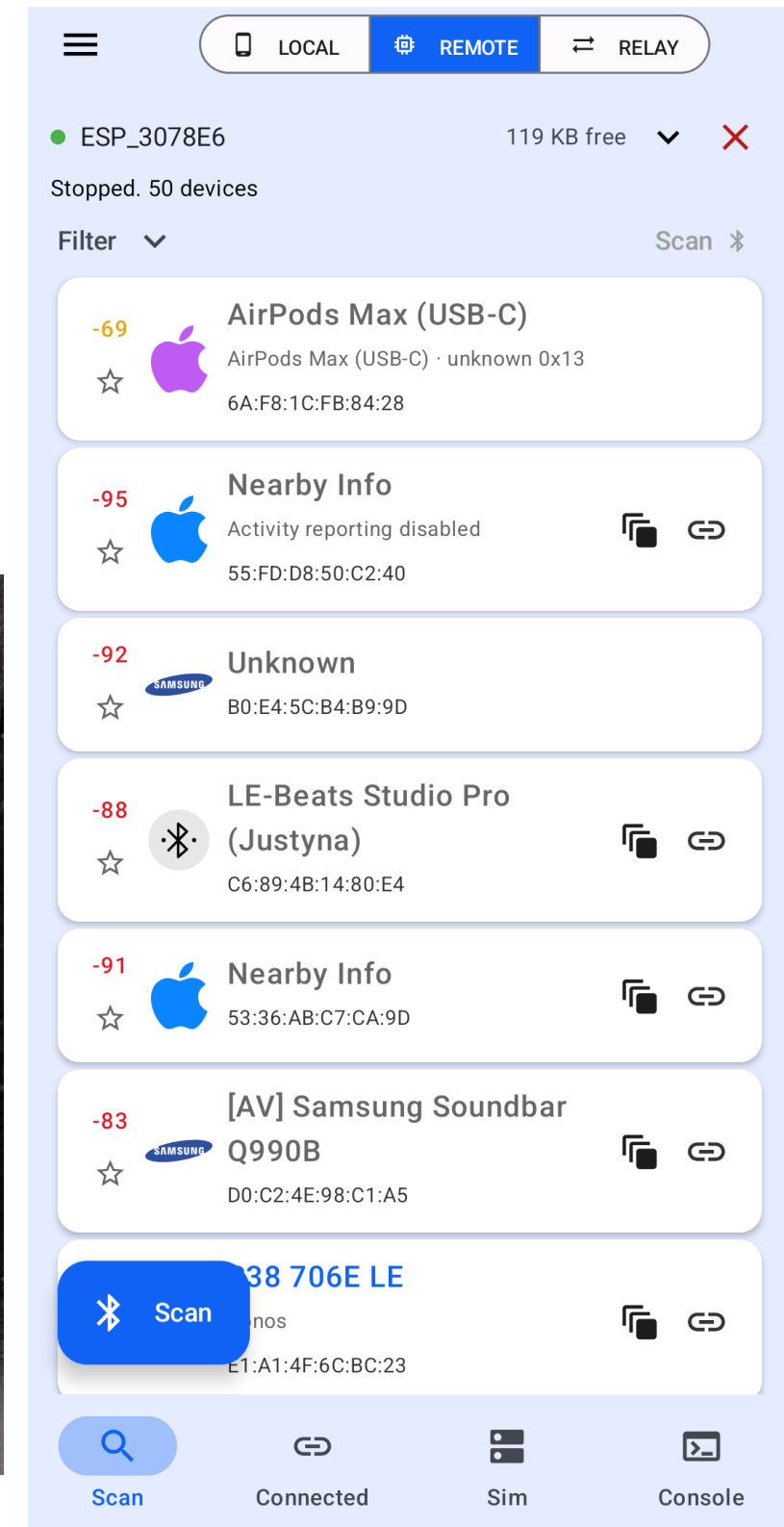
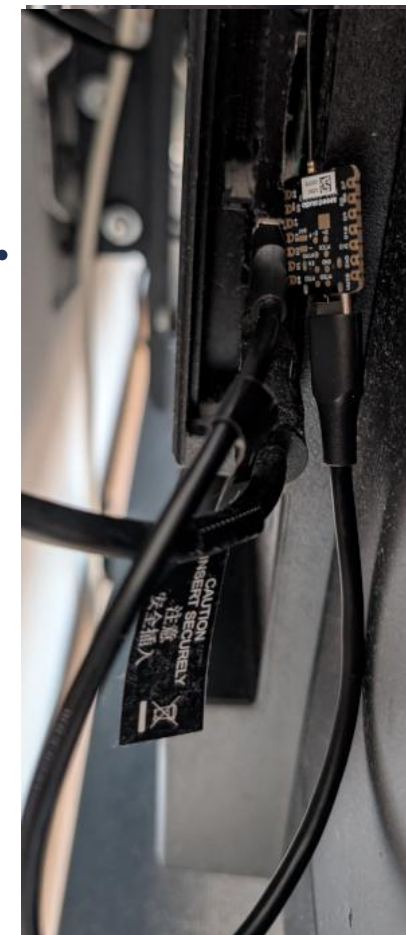
BLESploit ESP32 solves this problem



Remote Scan

Scan: works like local but using ESP32 Bluetooth adapter remotely (works also via Internet).

Remotely scan, connect, control devices (“Bluetooth proxy”). Full relay coming next.

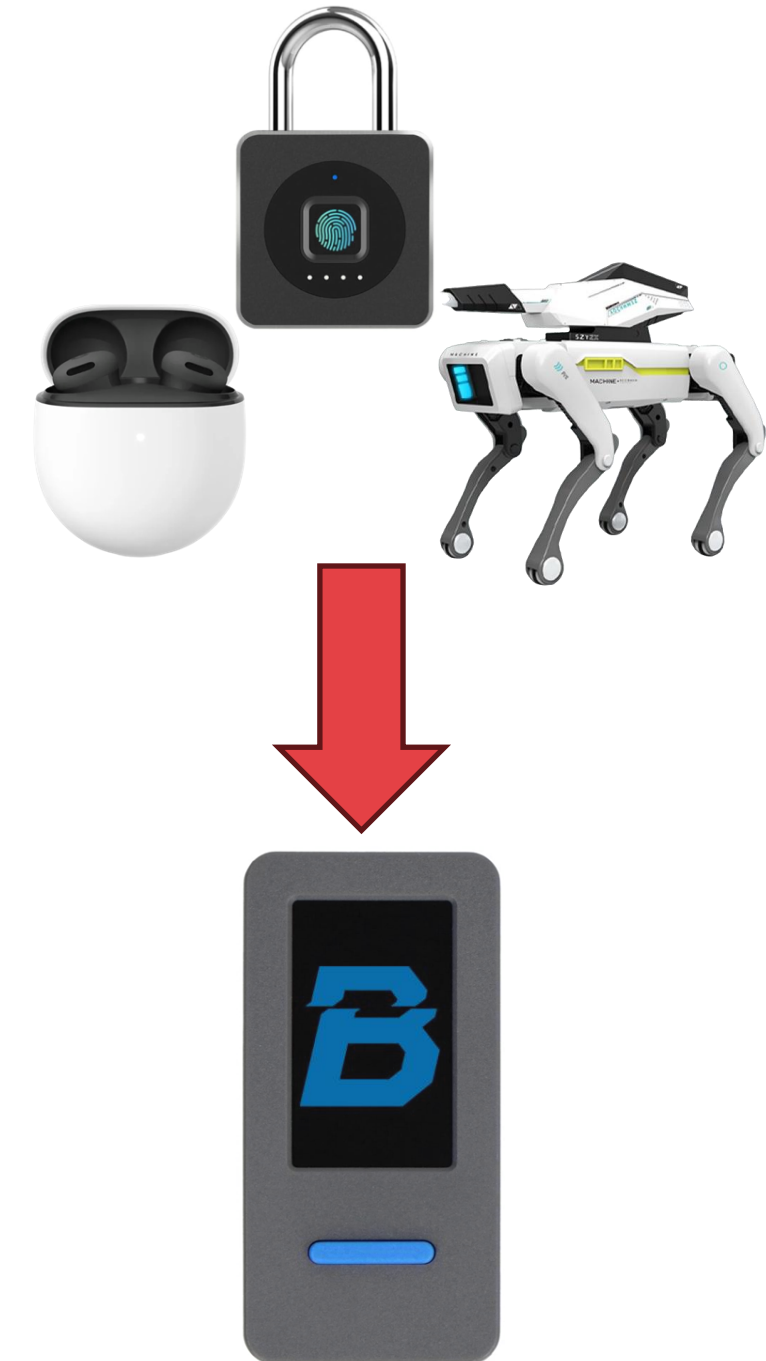


Remote Sim

Simulate any BLE device on ESP32

- Test application without the need to have real hardware
- Prototype, test tools in a stable environment
- Simply capture application traffic in various device states
- Remove link-layer pairing (app will not notice).
- Attack the mobile app by malicious device

Several example “peripheral” profiles in the Device Library



Create a new “clone” in < 60s

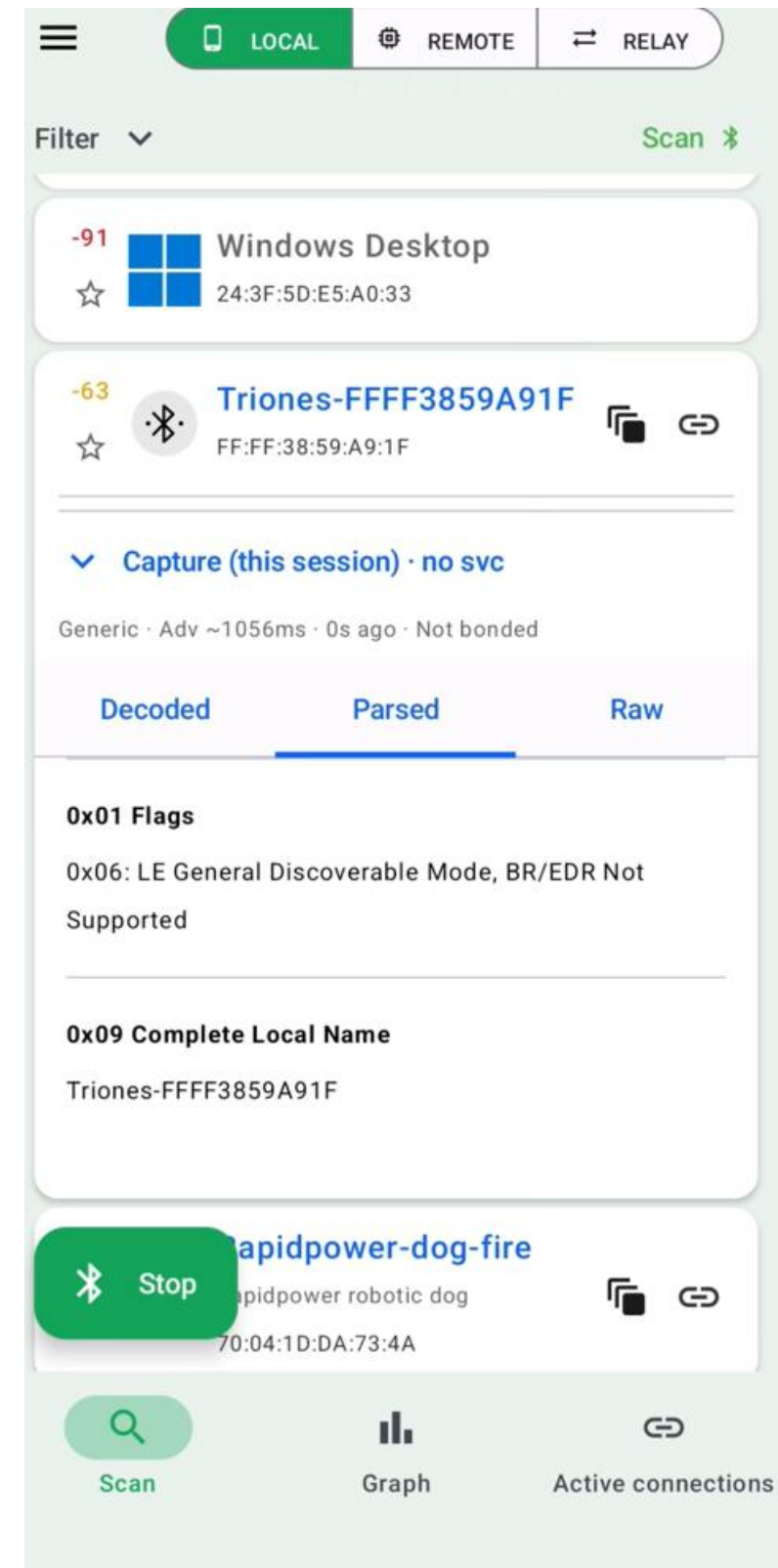
1. Scan the original device

- The advertisement is saved as soon as you tap on it
- The services+characteristics are saved after connection

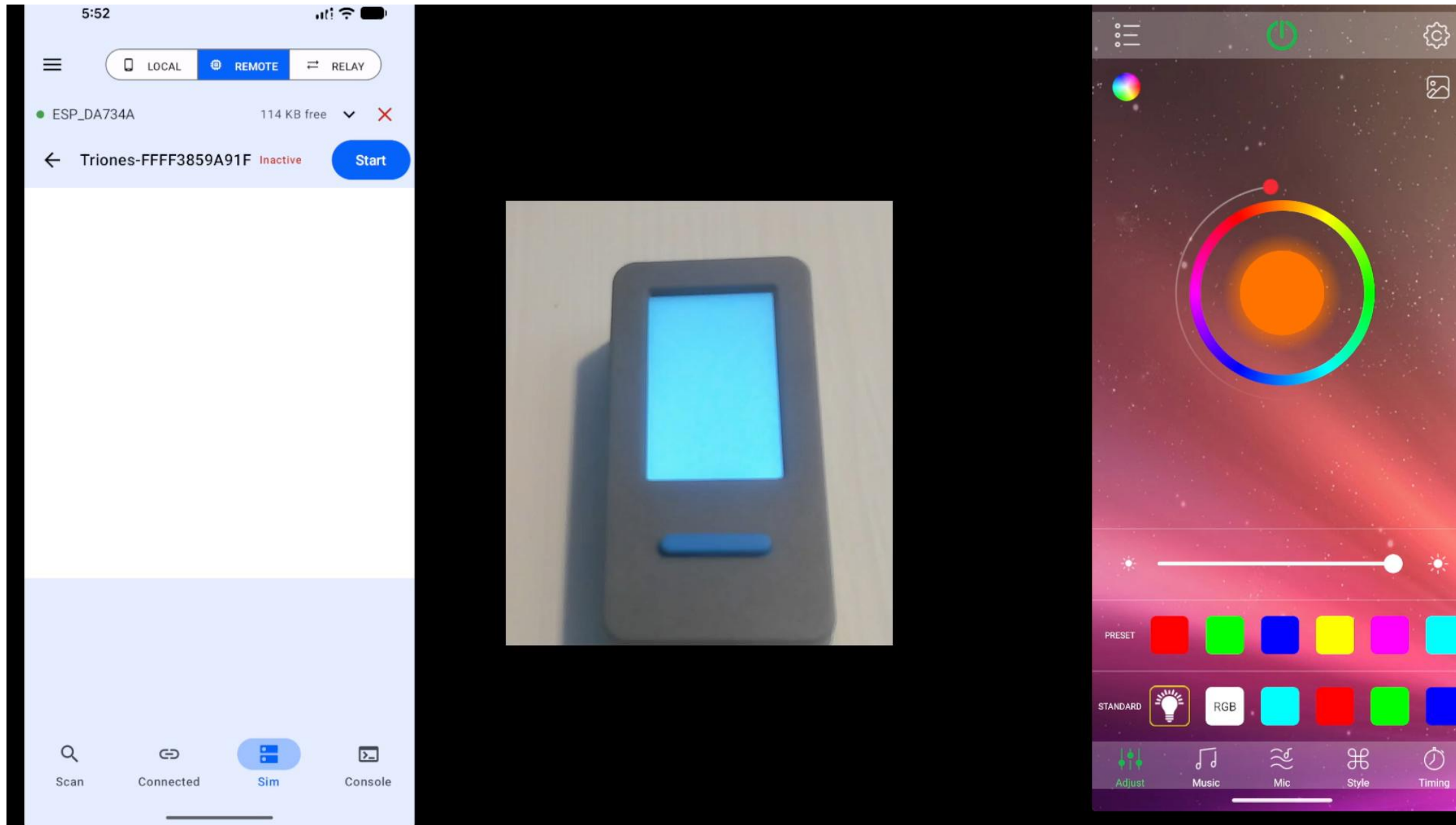
2. Select “Create device library entry”

3. Upload to ESP32

https://youtube.com/shorts/mMpj99VAb_4



The original application connects to the clone, grab the trace



Proprietary protocol

Analyse mobile application
(static/dynamic)



Capture the packets
on the phone

Sniff/MITM
wireless



Dump/reverse
firmware
(static/dynamic)

Proprietary protocol

Analyse mobile application
(static/dynamic)



Capture the packets
on the phone

Sniff/MITM
wireless



Simulate original
device to capture
the data from app



Dump/reverse
firmware
(static/dynamic)

Figure out the commands

ON/OFF: CC2333 / CC2433

Change color: 5618FF0000F0AA

Explain the BLE communication protocol between the mobile application and the devices.

3) Packet style

This is a **byte-oriented protocol** (usually 3–12 bytes, sometimes 18 or long timer payloads), no generic protobuf/JSON/etc.

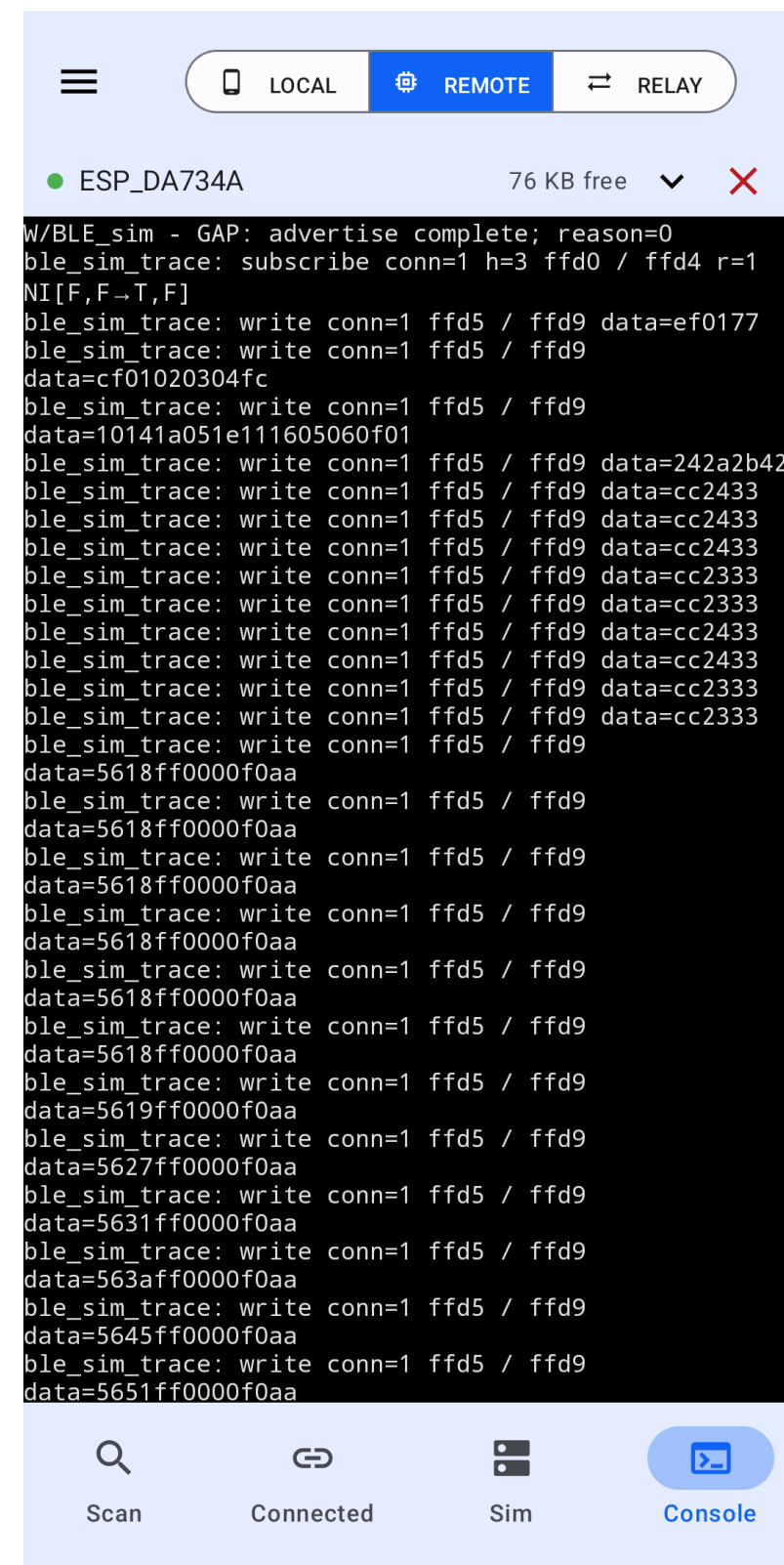
Common pattern:

- first byte = opcode/header,
- middle bytes = params,
- last byte often = fixed tail/check marker.

Examples:

- set RGB: 56 R G B W F0 AA
- power on/off: CC 23 33 / CC 24 33
- query state: EF 01 77
- password check: CF d1 d2 d3 d4 FC

(Those correspond to methods in setColor, openLight, getLightData, checkpwd.)



The screenshot shows the SMARTLOCKPICKING.COM mobile application interface. At the top, there are three tabs: LOCAL, REMOTE, and RELAY. The REMOTE tab is selected. Below the tabs, the device name 'ESP_DA734A' is displayed, along with '76 KB free' and a red 'X' icon. The main area shows a list of BLE communication packets. The packets are as follows:

```
W/BLE_sim - GAP: advertise complete; reason=0
ble_sim_trace: subscribe conn=1 h=3 ffd0 / ffd4 r=1
NI[F,F→T,F]
ble_sim_trace: write conn=1 ffd5 / ffd9 data=ef0177
ble_sim_trace: write conn=1 ffd5 / ffd9
data=cf01020304fc
ble_sim_trace: write conn=1 ffd5 / ffd9
data=10141a051e111605060f01
ble_sim_trace: write conn=1 ffd5 / ffd9 data=242a2b42
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2433
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2433
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2433
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2333
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2333
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2433
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2433
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2333
ble_sim_trace: write conn=1 ffd5 / ffd9 data=cc2333
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5618ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5618ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5618ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5618ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5618ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5619ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5627ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5631ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=563aff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5645ff0000f0aa
ble_sim_trace: write conn=1 ffd5 / ffd9
data=5651ff0000f0aa
```

At the bottom, there are four icons: Scan, Connected, Sim, and Console. The Console icon is highlighted with a blue background.

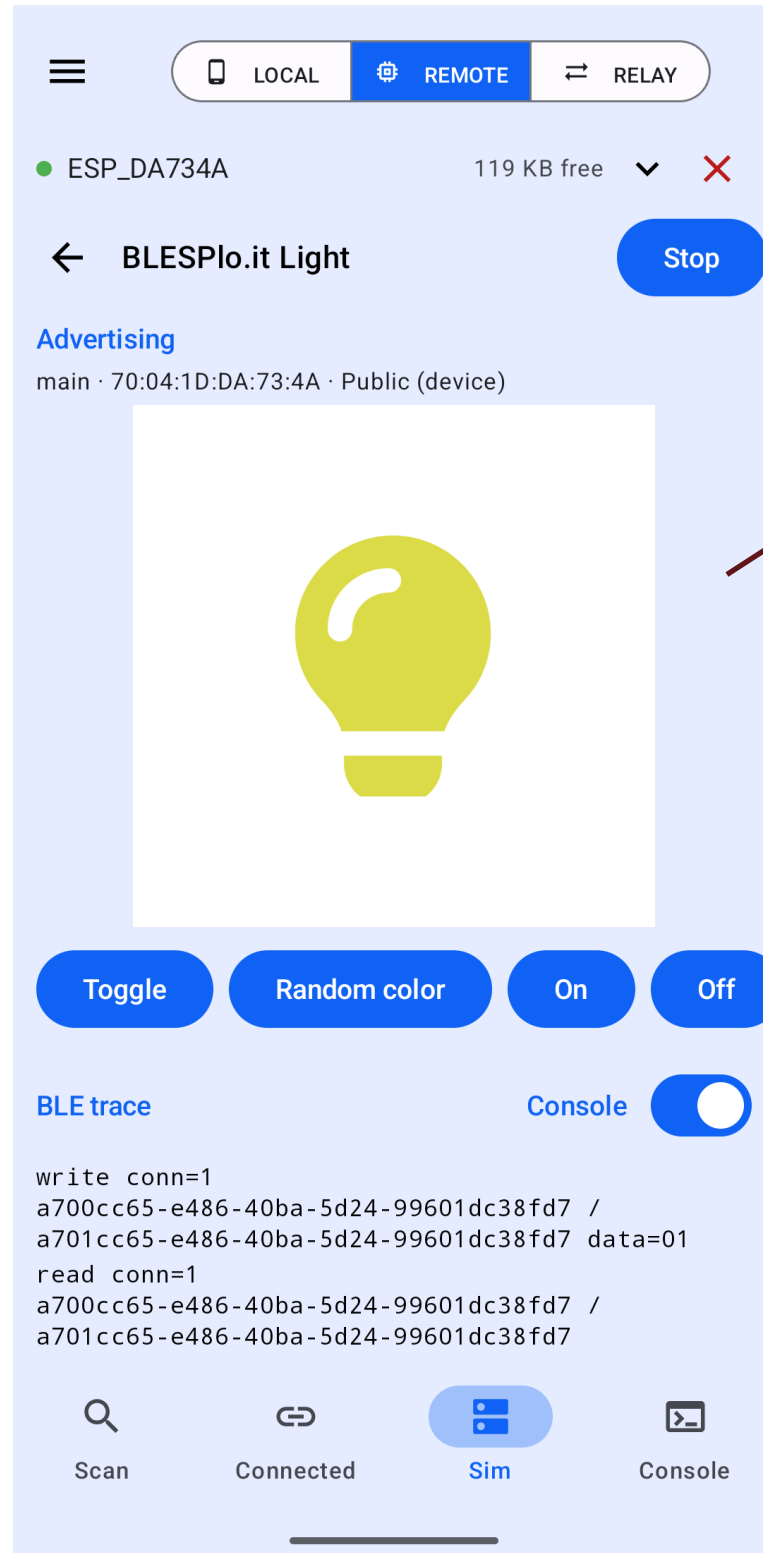
Cloning the BDADDR

BLE address can be Public or Random (Static, Resolvable, Non-Resolvable) – defined in the top 2 bits.

A small patch on NimBLE allows to set any arbitrary value as Random.

In majority of situations such cloned address is the same for the phone that connects to the device.

Extend to add graphics, buttons, logic, ...



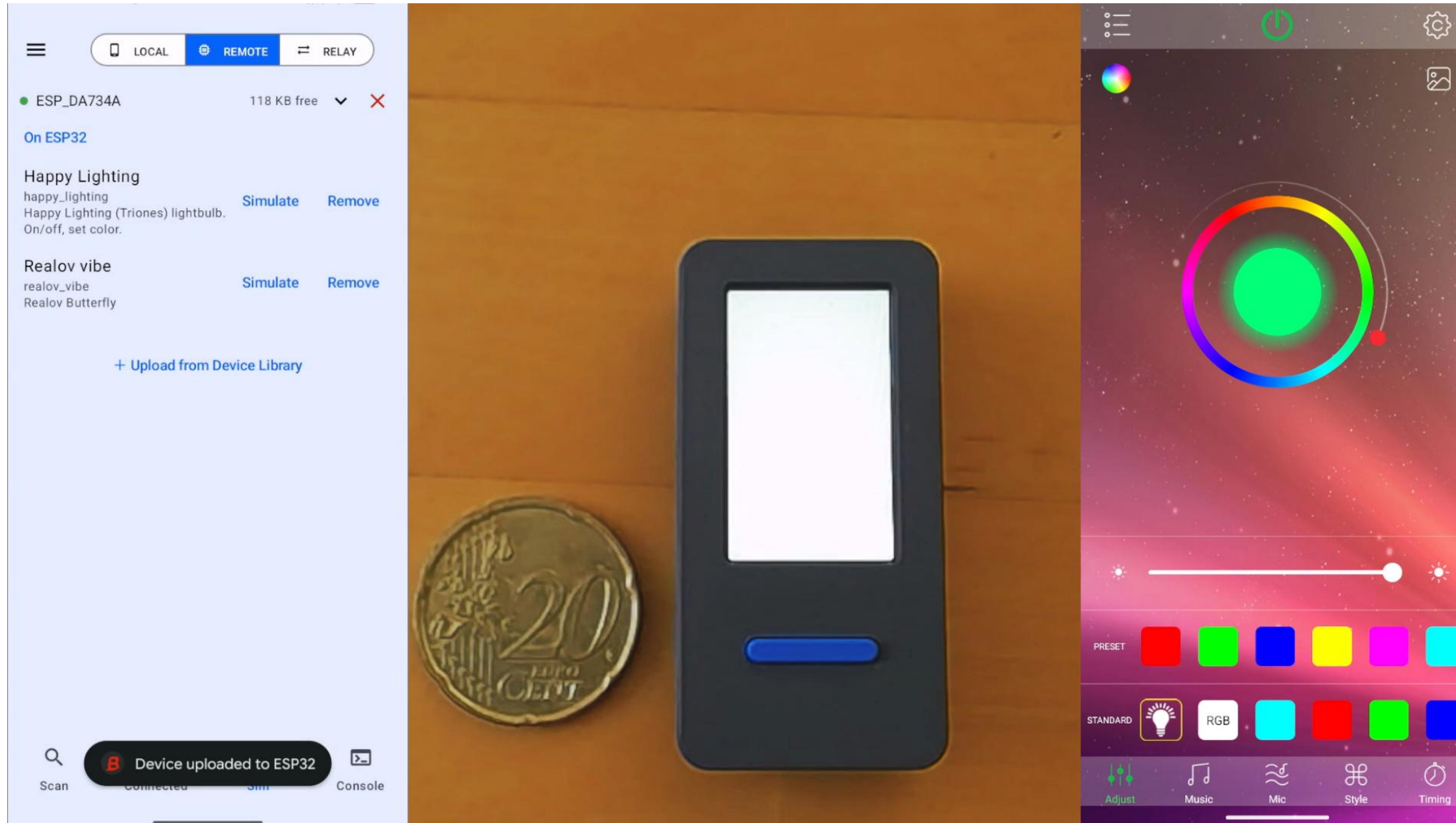
Simulation “canvas” –
updated by dynamic scripts

Optional “local” buttons –
simulate on-device interface
(e.g. light switch)

Trace of all the incoming
packets

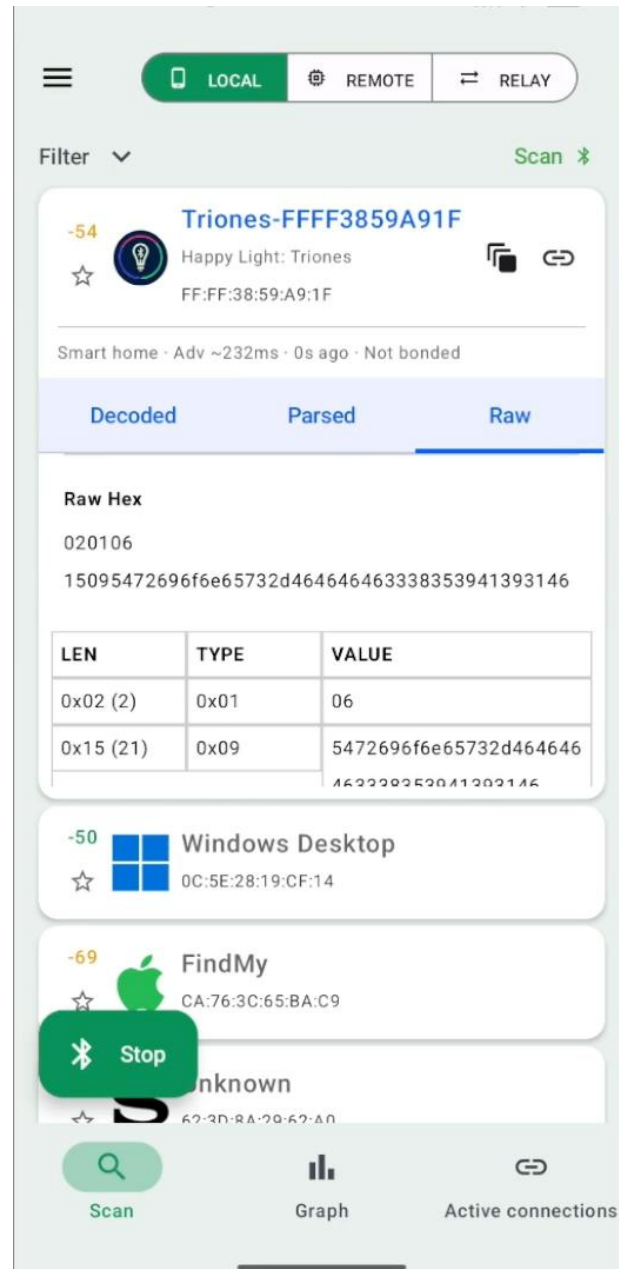


Demo



<https://www.youtube.com/watch?v=zUJHDkzPpjU>

Demo - scripts



Many sample devices available in the library



Rapidpower robotic dog
 products/rapidpower_robotic_dog
 Rapidpower










Realov vibe
 products/realov_vibe
 Realov Lydia I Smart Butterfly










Shell SuperCars
 products/shell_supercars
 Shell Racing toy car (2020 edition)










Sony WH-CH720N
 products/sony_wh_ch720n
 Sony Headphones, vulnerable to
 WhisperPair and Airoha









Tesla
 vendors/tesla
 iBeacon-format vehicle proximity token
 (18-char local name starting with S).









Happy Lighting


products/happy_lighting
 v1 bundled
 Slawomir Jasek

Happy Lighting smart light systems. On/off, set color.

Product / model URL
<https://happylightingapp.com/>

Companion apps

Direct download links


 confirmed APK
 v1.6.49
<https://apkpure.com/happylighting/com.xiaoyu.hlight/download/1.6.49>


 Google Play
 There are also several unofficial ones available.
<https://play.google.com/store/apps/details?id=com.xiaoyu.hlight>


 Apple App Store
<https://apps.apple.com/us/app/happy-lighting-life-with-smart/id1145694075>

Notes

Confirmed working with 'Triones' lightbulb. Other device names extracted from the app (unconfirmed). The device seems to have a PIN authentication, but it is not enforced nor visible anywhere the user interface. The a...

Show more



Realov vibe


products/realov_vibe
 v1 bundled
 Slawomir Jasek

Realov Lydia I Smart Butterfly

Product / model URL
<https://www.realov.com>

Companion apps


 Google Play
https://play.google.com/store/apps/details?id=cn.ble.realov_new


 Apple App Store
<https://apps.apple.com/us/app/realov/id893957103>



Rapidpower robotic dog


products/rapidpower_robotic_dog
 v1 bundled
 Slawomir Jasek

Rapidpower

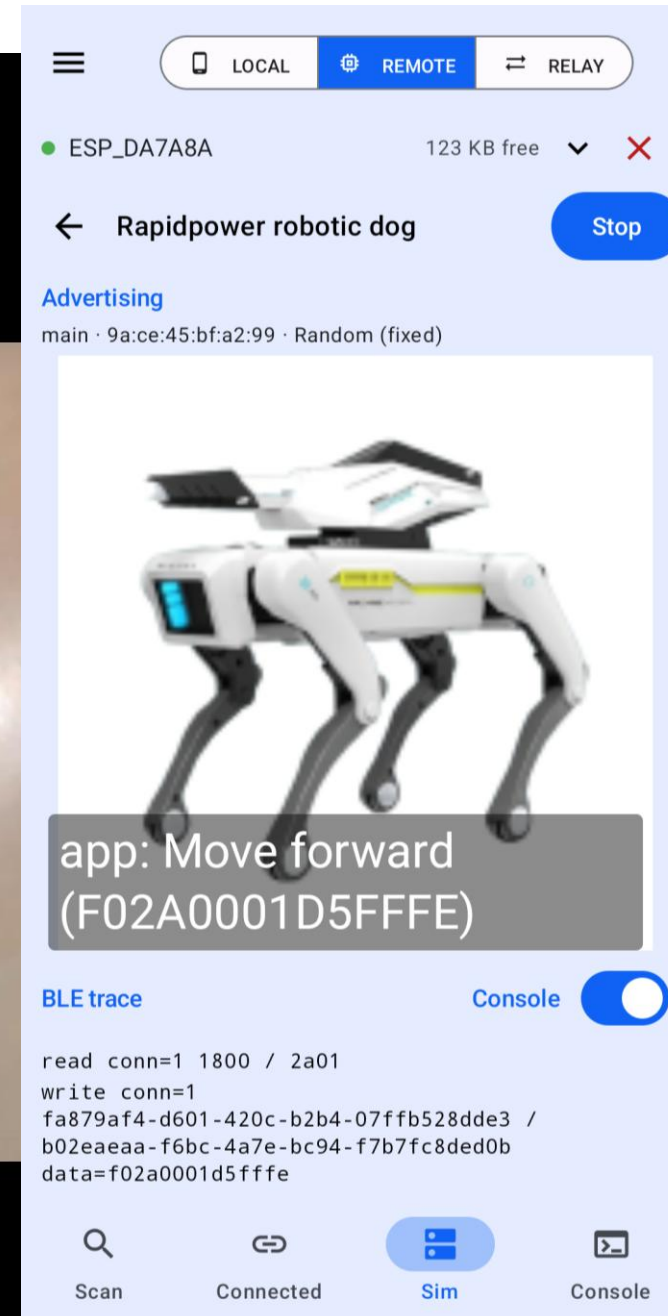
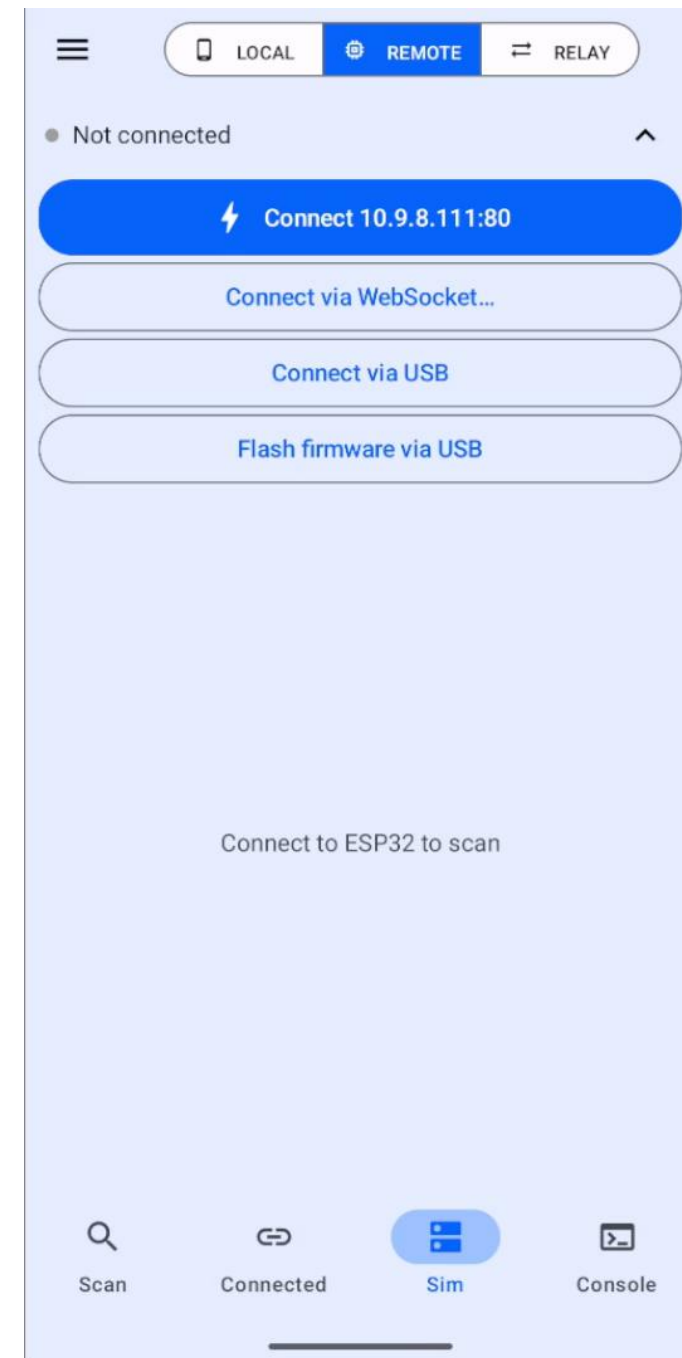
Product / model URL
<http://www.rapidpowertoys.com>

Companion apps

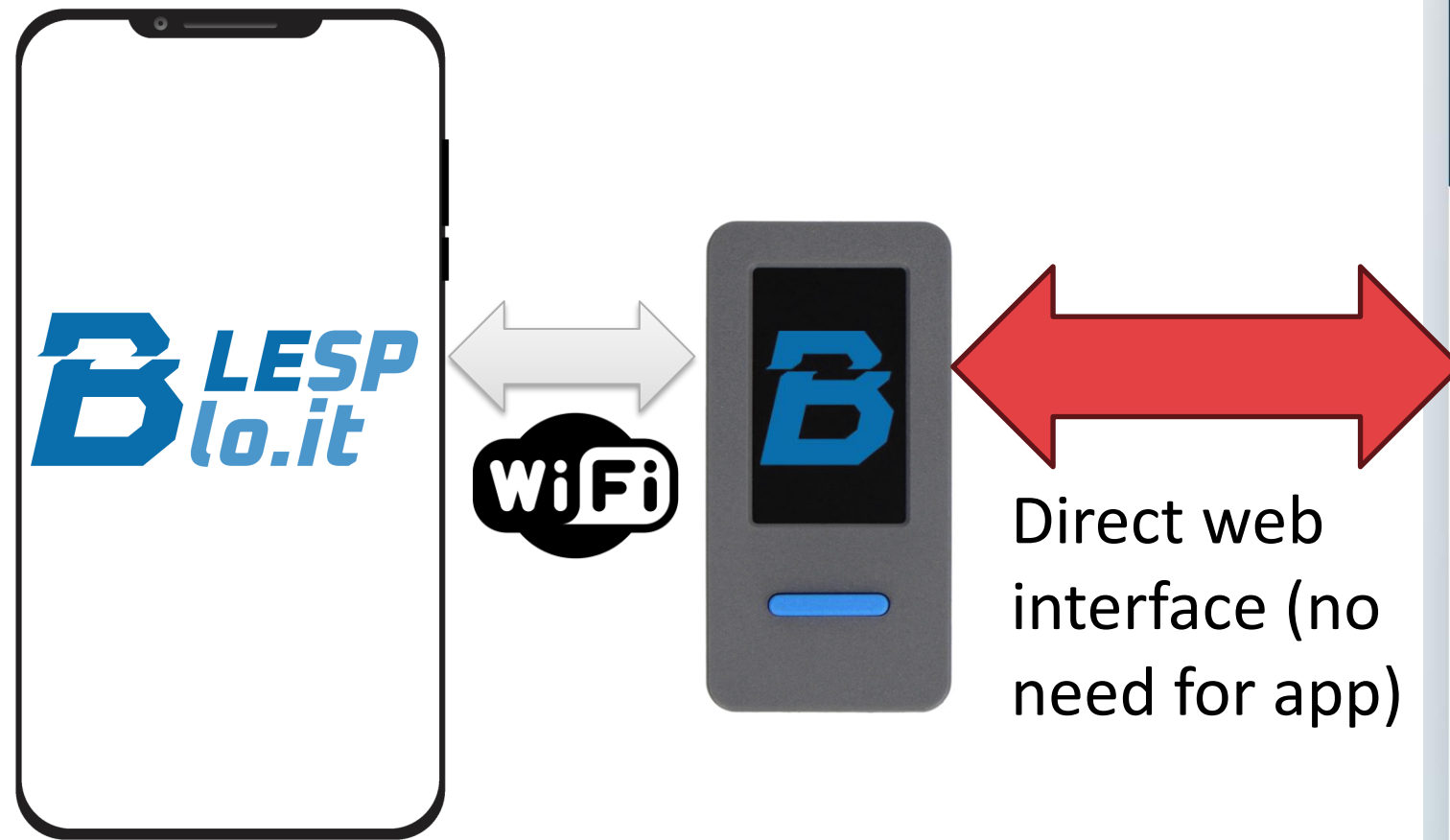

 Google Play
<https://play.google.com/store/apps/details?id=com.zhongrun.robotdog>


 Apple App Store
<https://apps.apple.com/us/app/rapidpower/id6479237013>

Demo: simulate the robotic dog



Direct web interface simulation status



BLESPlo.it Home Scanner Devices Files Wifi settings Reboot Scan Central Periph WS On

Rapidpower robotic dog
 Active
 Device ID: rapidpower_robotic_dog
 Rapidpower
 Author: Slawomir Jasek
 WS On

Device Display
 Start Simulation Stop Simulation

Advertising Status
 Instance 0 — main advertising
 BD Address: 70:04:1D:DA:73:4A | Type: Public (hardware)

Debug Log
☐ WebSocket messages ☒ BLE inbound trace
☒ ESP logs (Error, Warning)
☐ ESP logs (Info)

08:19:18 web [I] WebSocket connected
 08:19:18 wslog [I] filter loaded — enabled=true, info=false, tags=[]
 08:19:18 web [I] WS raw frame (non-JSON):
 08:19:18 web [I] Device info loaded: Rapidpower robotic dog
 08:19:19 web [I] WS hello from firmware (v?)
 08:19:19 sim [I] Simulating: rapidpower_robotic_dog
 08:19:20 canvas [I] PNG loaded:

Idle

Realov Vibe: official app



Realov

PauloCostacx66 Entertainment

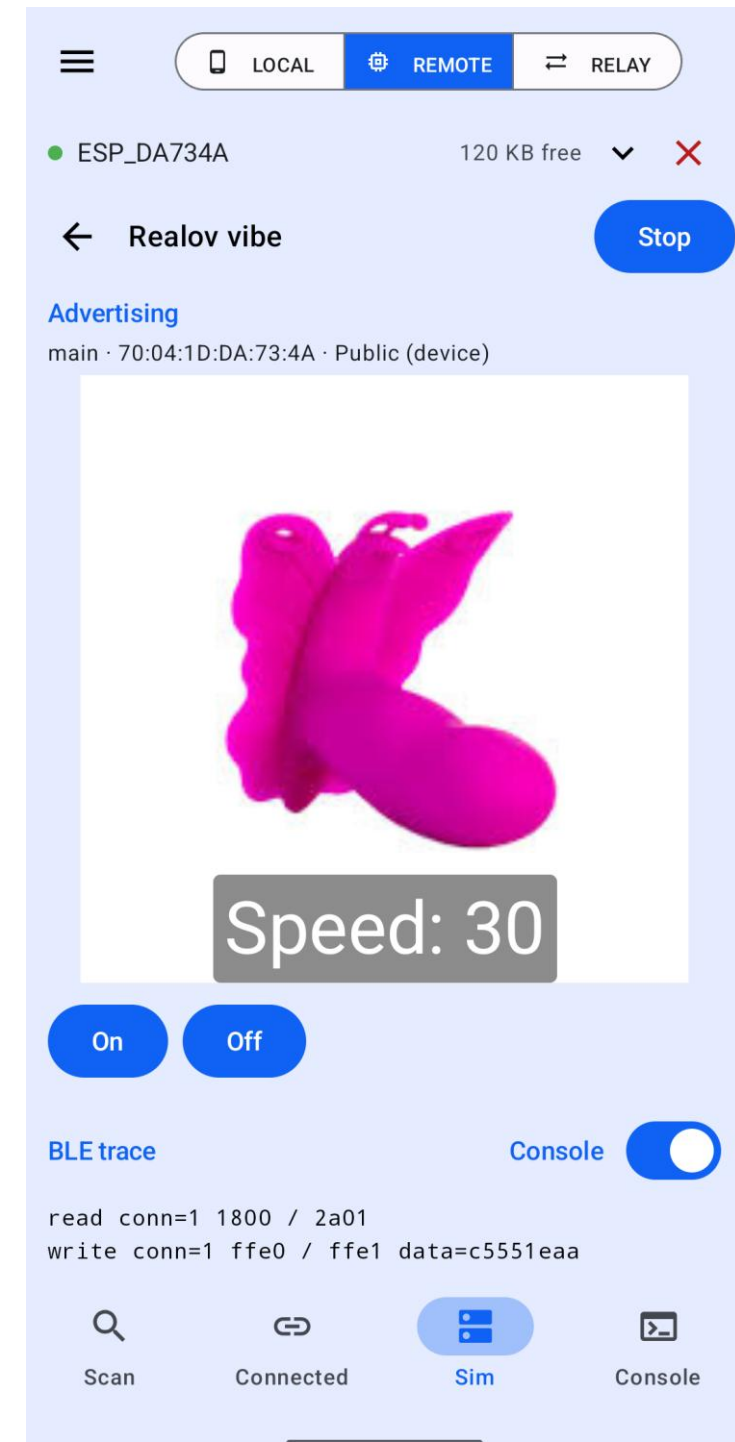
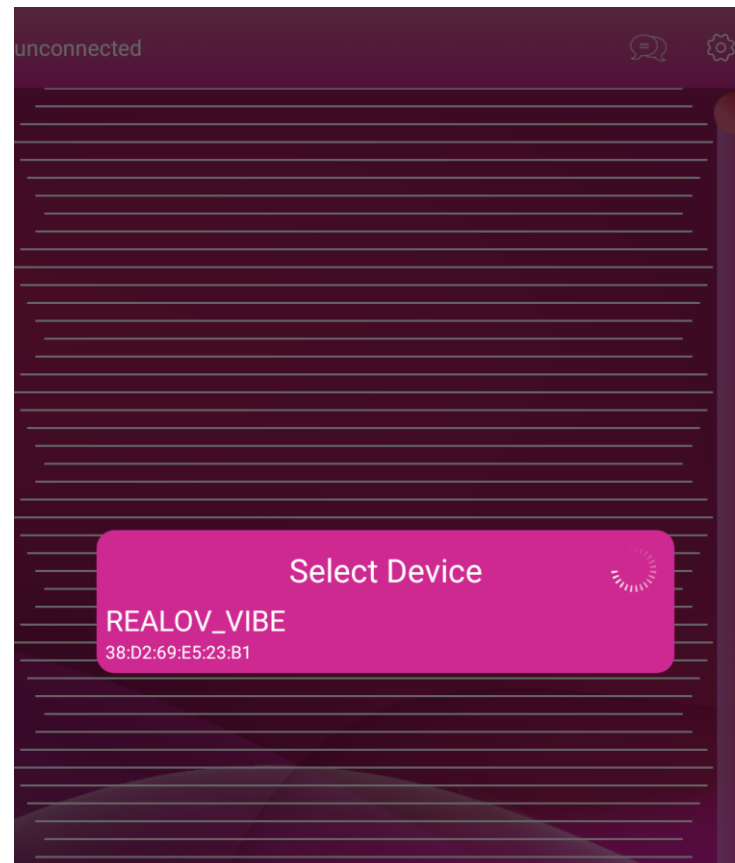
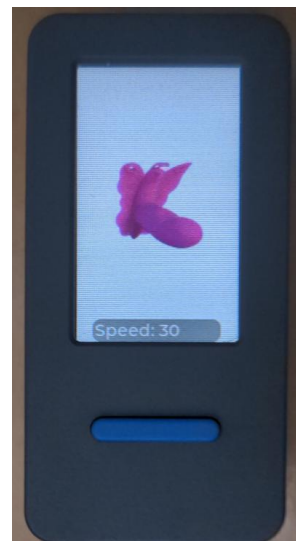
★★★★★ 103

PEGI 3

Add to Wishlist

Install

https://play.google.com/store/apps/details?id=cn.ble.realov_new

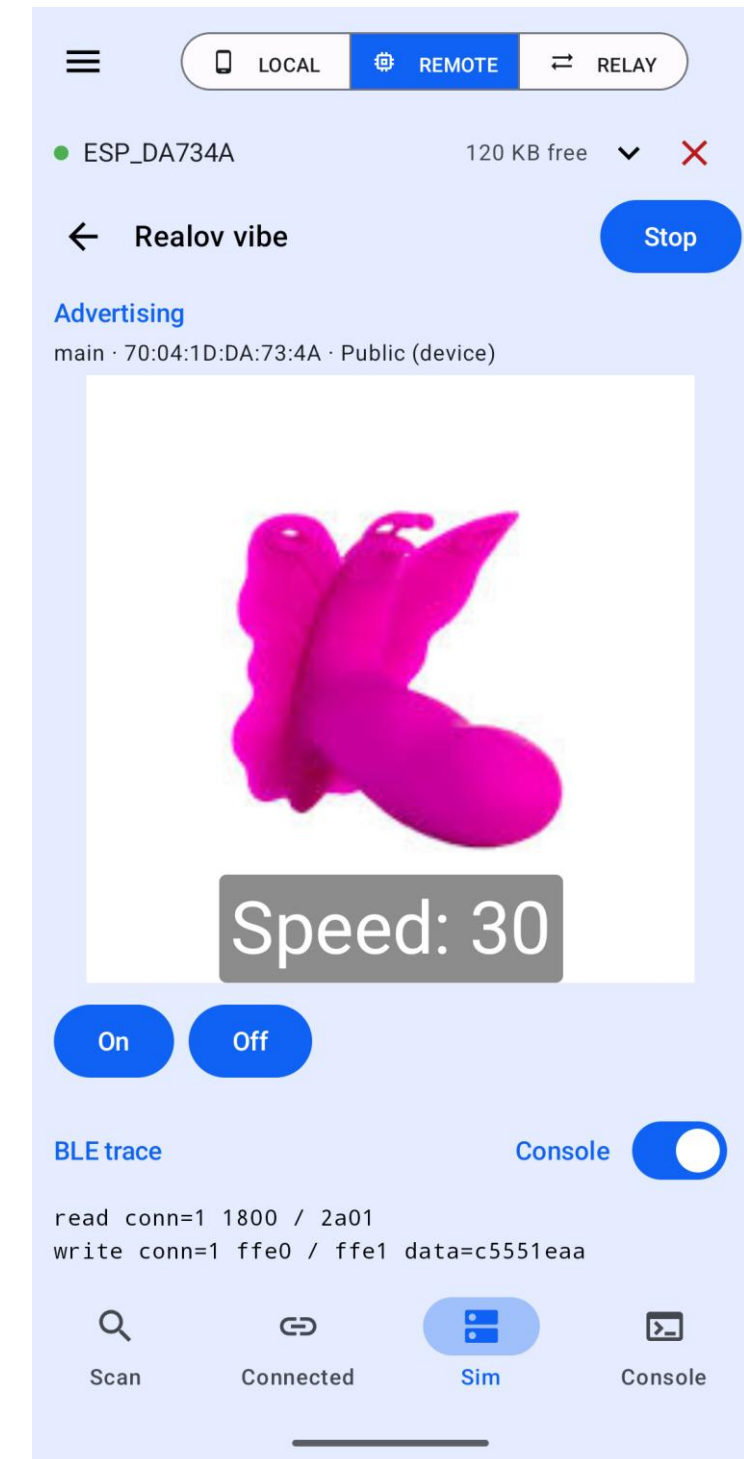
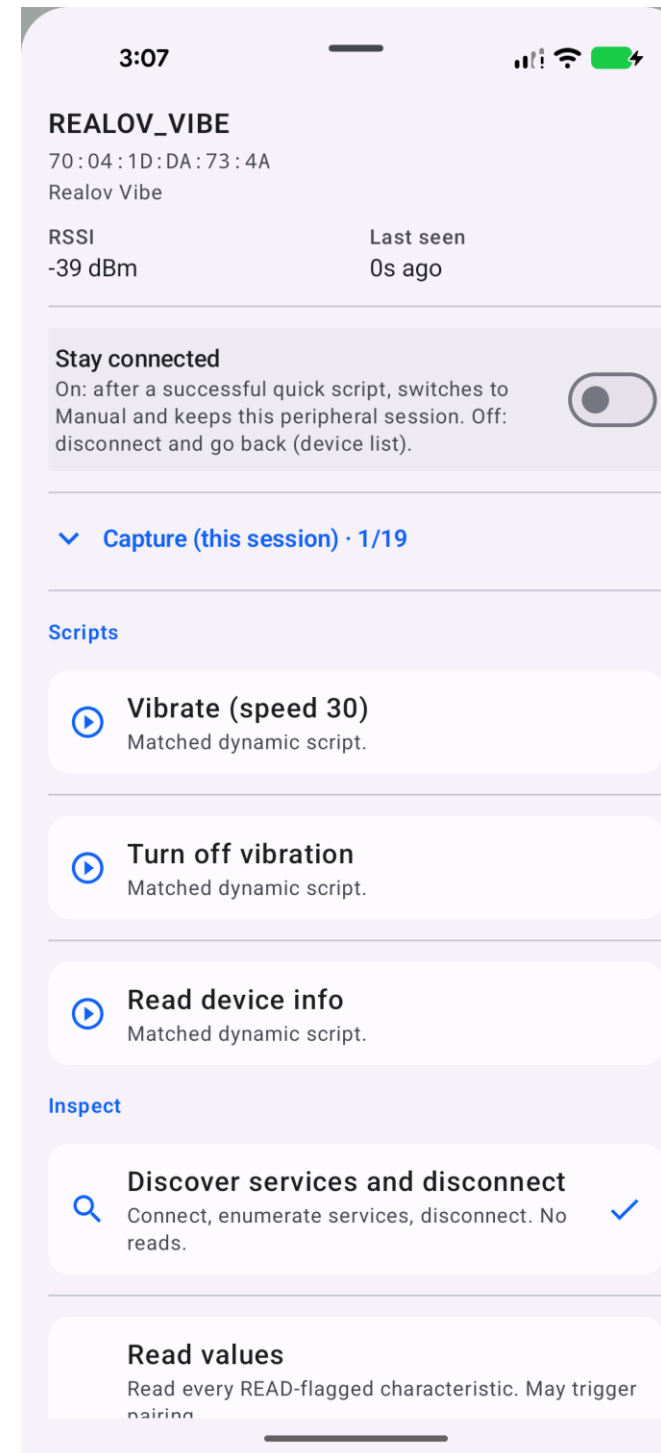
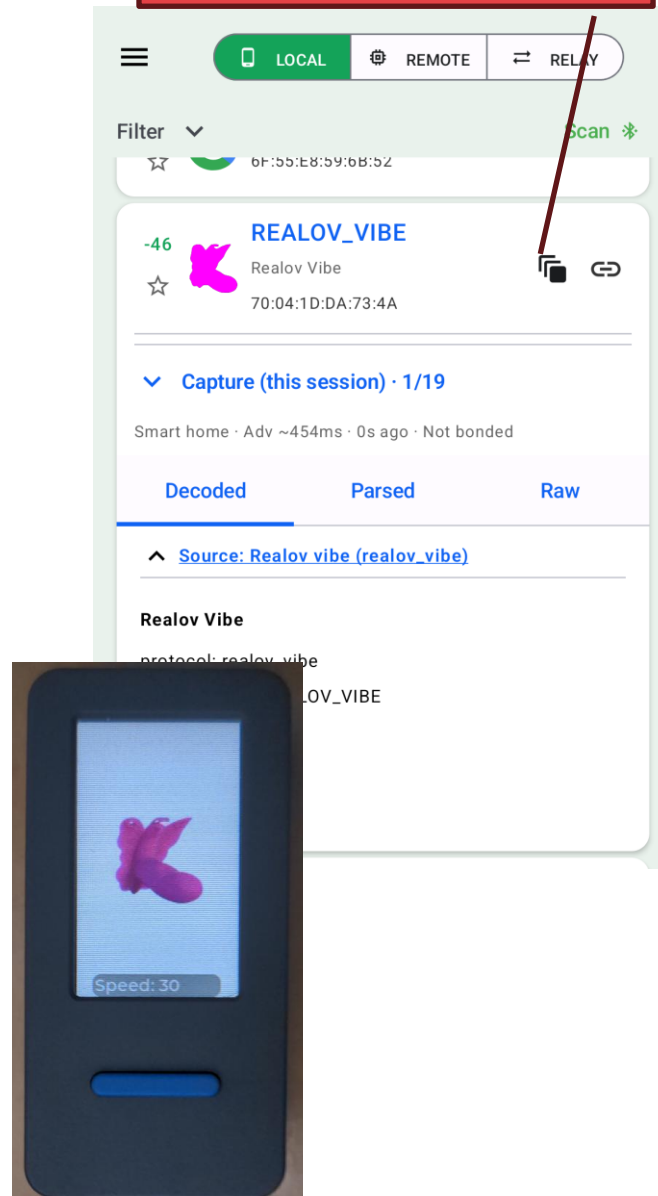


Use BLESPlo.it scripts

simulate



Quick actions



Ferrari SF-24



https://en.wikipedia.org/wiki/Ferrari_SF-24

Our simplified target...



Original application

Shell Racing Legends

Carbon12011

Shell Racing Legends (note: NOT “Shell Racing”)



<https://play.google.com/store/apps/details?id=com.TDF.ShellRacingLegends>

<https://apps.apple.com/pl/app/shell-racing-legends/id1578287269>

Another application - Carbon RC

<https://play.google.com/store/apps/details?id=com.bt.car>

Simulate, observe packets, try to replay, ...

Device library

Observer Peripheral Central

Bundled Local

Shell Racing Legends
products/shell_racing_legends
Shell Bluetooth RC cars (Shell Racing Legends app). Plain 8-byte drive write...

Shell SuperCars
products/shell_supercars
Shell Racing toy car (2020 edition)

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Value	Info
201	28.561...	controller	host	HCI_EVT	50		Rcvd LE Meta (LE Extended Advertising Report)
202	29.464...	localhost ()	13:05:aa:01:2...	ATT	24	01000100000000100	Sent Write Command, Handle: 0x000a (Unknown)
203	29.492...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
204	29.569...	localhost ()	13:05:aa:01:2...	ATT	24	01000000000000100	Sent Write Command, Handle: 0x000a (Unknown)
205	29.582...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
206	29.636...	localhost ()	13:05:aa:01:2...	ATT	24	01010000000000100	Sent Write Command, Handle: 0x000a (Unknown)
207	29.672...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
208	29.835...	localhost ()	13:05:aa:01:2...	ATT	24	01010000000000100	Sent Write Command, Handle: 0x000a (Unknown)
209	29.897...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
210	29.972...	controller	host	HCI_EVT	50		Rcvd LE Meta (LE Extended Advertising Report)
211	30.034...	localhost ()	13:05:aa:01:2...	ATT	24	01010000000000100	Sent Write Command, Handle: 0x000a (Unknown)
212	30.077...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
213	30.233...	localhost ()	13:05:aa:01:2...	ATT	24	01010000000000100	Sent Write Command, Handle: 0x000a (Unknown)
214	30.257...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
215	30.432...	localhost ()	13:05:aa:01:2...	ATT	24	01010000000000100	Sent Write Command, Handle: 0x000a (Unknown)
216	30.482...	controller	host	HCI_EVT	12		Rcvd Number of Completed Packets
217	30.502...	localhost ()	13:05:aa:01:2...	ATT	24	01000000000000100	Sent Write Command, Handle: 0x000a (Unknown)

> Frame 202: Packet, 24 bytes on wire (192 bits), 20 bytes captured
> Bluetooth
> Bluetooth HCI H4
> Bluetooth HCI ACL Packet
> Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol

> Opcode: Write Command (0x52)
Handle: 0x000a (Unknown)
Value: 01000100000000100

0000 02 13 00 0f 00 0b 00 04 00 52 0a 00 01 00 01 00
0010 00 00 01 00

“Perfect security” smart lock with great vulnerabilities

2023: LeHack'23: Vulnerability Analysis of a BLE Padlock (Orlaine Guetsa, Alexandre Goncalves, Morgan Yakleh)

<https://www.youtube.com/watch?v=PmoklaUQvXs>

<https://lehack.org/track/vulnerability-analysis-of-a-bluetooth-low-energy-padlock/>

2023: I submitted IDOR (get any lock PIN from server) to the vendor, no response so far.

2024: “Open, Sesame!” - Unlocking Bluetooth LE padlocks with polite requests. Alex Pettifer, Milosz Gaczowski

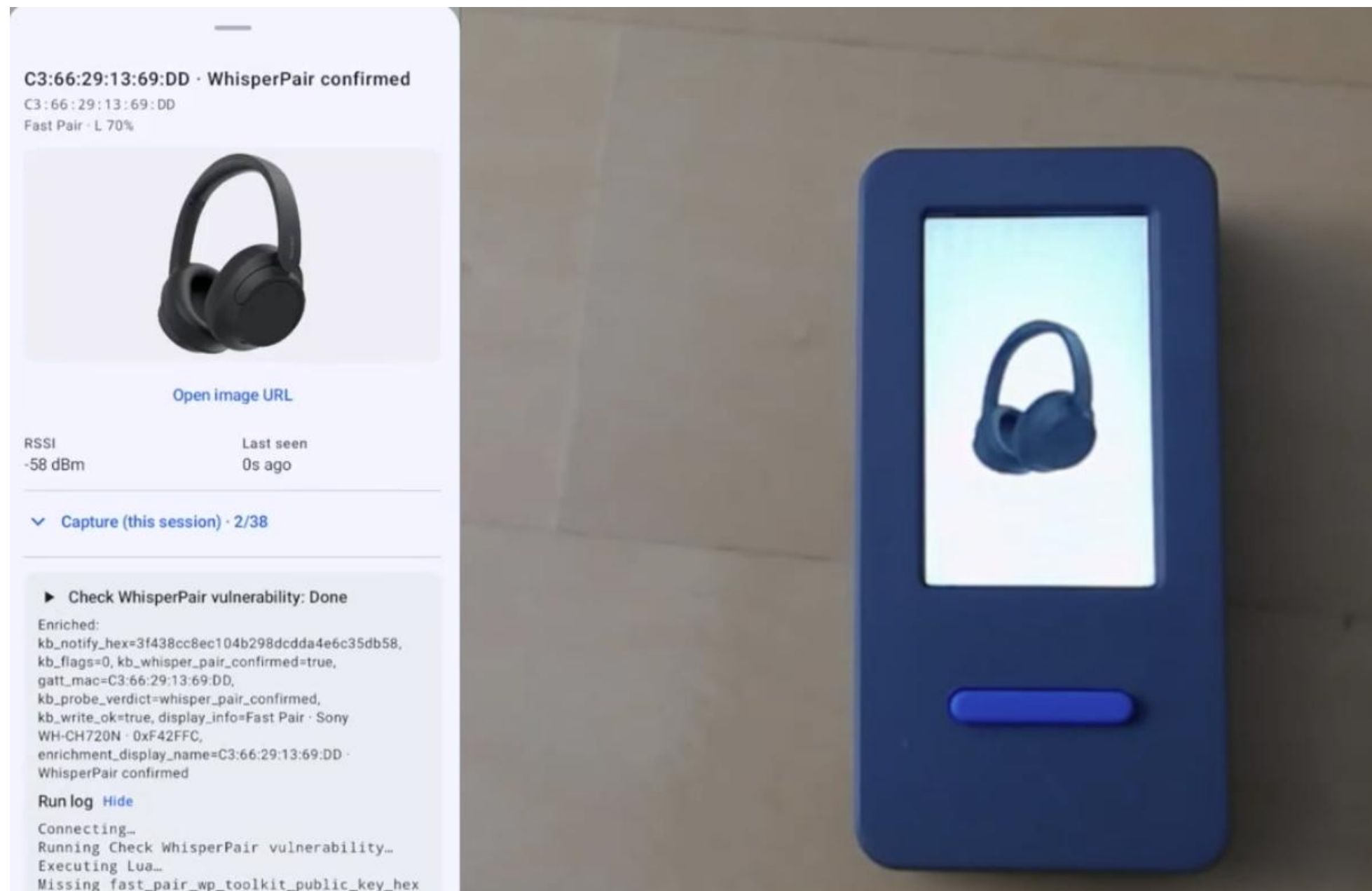
<https://labs.reversec.com/posts/2024/02/multiple-vulnerabilities-in-elinksmart-padlocks>

<https://www.youtube.com/watch?v=1JDqhzQCtAA>

2025: hextree.io free “Android Bluetooth reversing course” based on this lock <https://app.hextree.io/map/android>



Simulate headset to test WhisperPair, Airoha, ...



https://www.youtube.com/watch?v=m_757sqfw_Q

The One App To Rule Them All

Scan, fingerprint, decode, script, control proprietary protocols, create custom interfaces...



Scan/control remotely ("Bluetooth proxy")

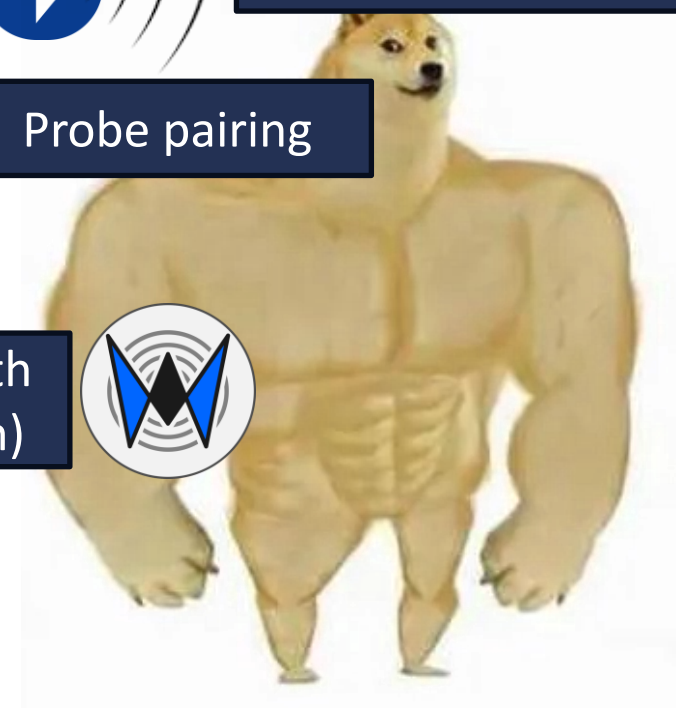
Clone / simulate any BLE device

Test vulnerabilities and attacks in a controlled environment

Probe pairing

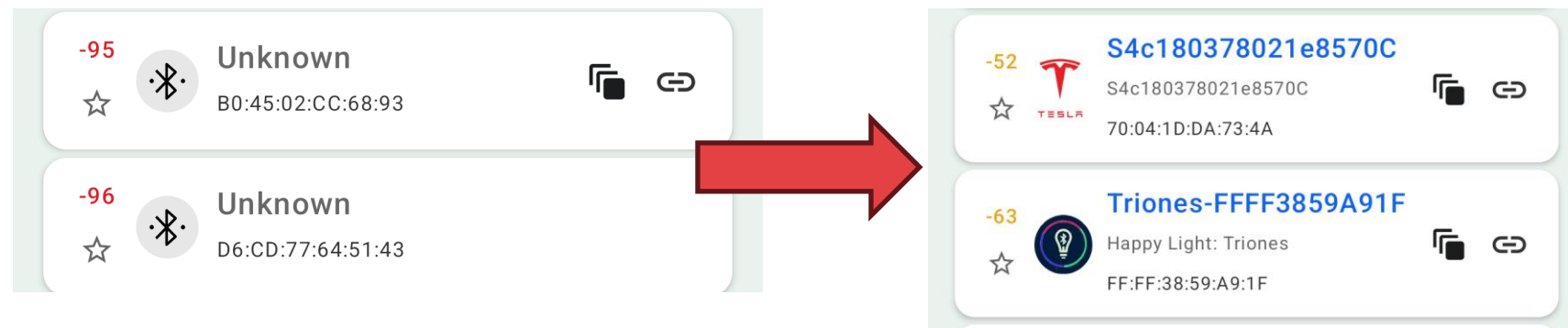
Direct web interface

Compatible with WHAD (python)



Now your turn!

Help unravel the “Unknown” and expand Device Library



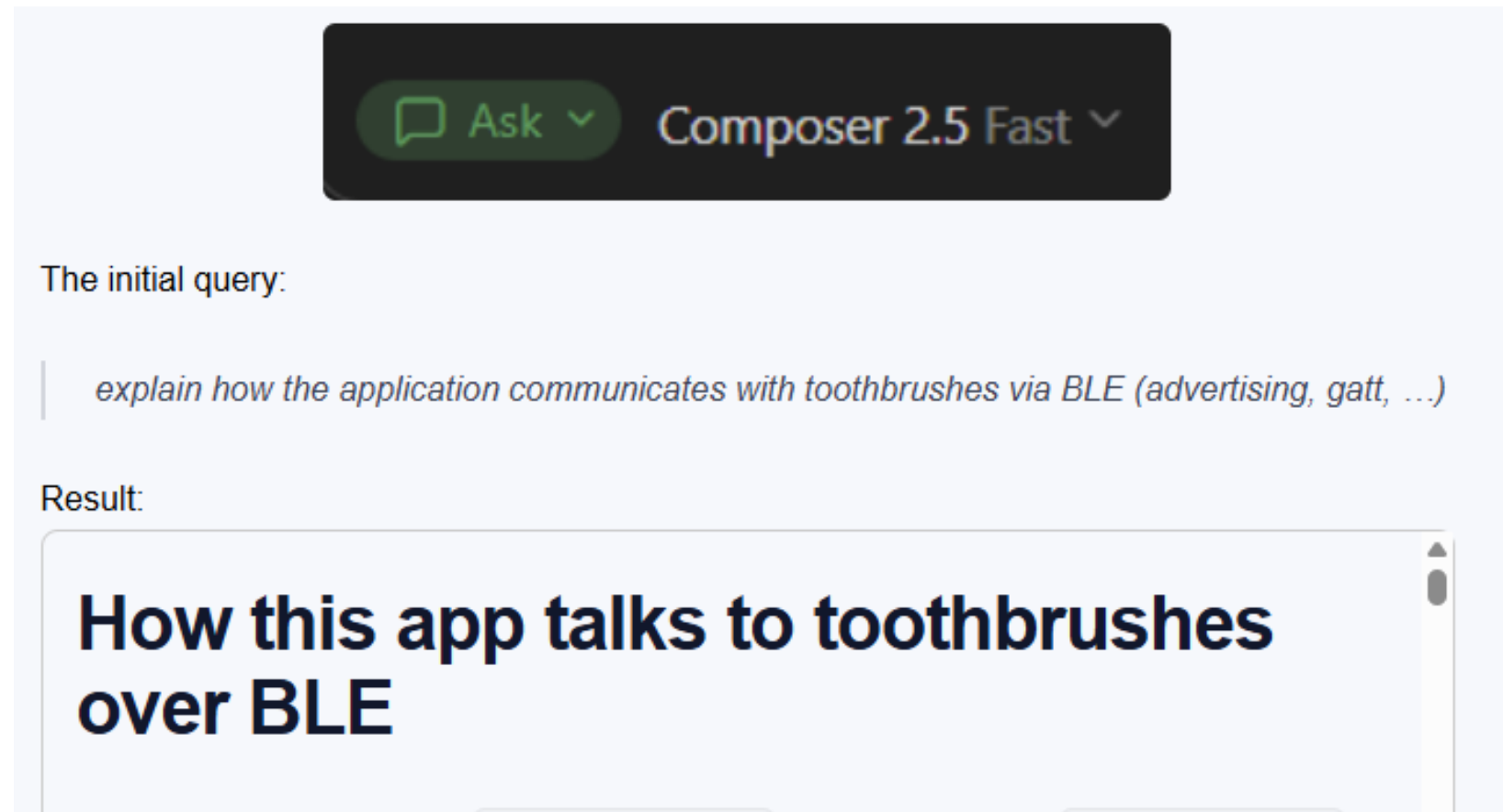
Even if you just scan the original device and let us take from there.



Sample process step-by-step based on a smart toothbrush

How to create a BLESPloit device

https://blesplo.it/docs/howto/create_blesploit_device/



Responsible Use

Test only devices you own or have been explicitly authorized.

Medical devices may cause real harm!

Policies > Responsible Use Policy

Responsible Use Policy

<https://blesplo.it/policies/responsible-use-policy/>

TECH NEWS

224 Injured After Glitchy Diabetes App Drains Insulin Pump Batteries

Users are being told to closely monitor their insulin pump battery levels before bed.

BY MATT NOVAK PUBLISHED MAY 9, 2024, 3:45 PM ET

READING TIME 2 MINUTES

What comes next?

-  Mobile app open-source
-  Bug fixes! Don't give me one star, send me the bug and I'll fix it!
-  Grow the Device Library. Help appreciated!
-  Remote relays.
-  Pairing probe/bypass.
-  Low level control over the ESP32 chip (NimBLE patch has limitations).
-  Integration with other fingerprinting sources.

Follow-up

Come to the Embedded Systems Village!

Trainings: BlackHat, next up India, EU, private,...

<https://blesplo.it> these slides, more docs,...



<https://github.com/BLESPloit>



<https://discord.gg/YaauZqG8d8>



<https://linkedin.com/in/slalomirjasek>



<https://bsky.app/profile/blesplo.it>

