DRIP Implementation

Andrei Gurtov & students
Linköping University, Sweden

DRIP IETF 110th Online Meeting
March 9, 2021

Starting Points

- OpenHIP (v2 alpha branch) - 4 students
  - https://bitbucket.org/openhip/
  - draft-moskowitz-hip-new-crypto-06
- OpenDroneID - 5 students
  - https://www.opendroneid.org/code/
- TDDE21 Advanced Project: Secure Distributed and Embedded Systems (6 ECTS) Sep-Dec 2020
  - https://www.ida.liu.se/~TDDE21/

New Requirements for HIPv2

- New cryptographic algorithm (EdDSA)
- New ORCHIDs - include additional info in Host Identity Tag (HIT), needed for hierarchical HITs
- Hierarchical HITs - embed information about the issuing authority inside HIT

New ORCHID

- Overlay Routable Cryptographic Hash Identifiers
- Endpoint identifiers at applications
- Before => ORCHID := Prefix | OGA ID | Encode_96(Hash)
  - Prefix = 2001:20::/28 (IANA)
  - ODA ID = 4 bit hash identifier
  - Encode_96(Hash) = Middle 96 bits of the hash output
- Now => ORCHID := Prefix | OGA ID | Info (n) | Hash (m)
  - New prefix for HHITs
  - Hash(m) = Hash function which outputs m bits, use cSHAKE
  - Encode_96(Hash) split into Info(n) and Hash(m), Info(n) used in HHITs as a tag

New Crypto

- EdDSA (Edwards-curve Digital Signature Algorithm) is a digital signature scheme that is based on elliptic-curve cryptography.
  - Designed to be a fast algorithm without sacrificing security
  - Less dependent on a good random number generator, compared to ECDSA

- The Keyak cipher is used as a lightweight alternative to AES, and also supports authentication of the encrypted data
  - Move to Xoodyak, follow NIST

- The KKDF key derivation function (based on KMAC) is a more efficient alternative to the HMAC-based HKDF

Current status – New crypto

All parts of the draft are working:

- Curve25519 and **Curve448** Diffie-Hellman key exchange
- EdDSA key generation (HI) and signatures **(based on both curves mentioned above)**
- **HIT generation with New ORCHIDs is done**
- KKDF as key derivation function instead of HKDF
- Keyak as HIP_CIPHER alternative (lightweight, authenticated cipher)
- **Base exchange with HIT suite negotiation now works fine**

Problem statement - DRIP

Broadcast prototype Drone ID over Bluetooth or WiFi as HIP Host identity tag

- DRIP IETF Working Group
- Host Identity Tag
- Raspberry Pi
- 20 Bytes
- Receiver in Android

State of the art

Remote ID RID

- No trust in broadcast messages

Host Identity:

- Host Identity Tag (HIT)
  - Hashed encoding of the Host Identifier
  - Encoded according to ORCHID generation method using a specific context ID
  - Algorithms used: SHA-1, SHA-256 and SHA-384
- Hierarchical Host Identity Tag (HHIT)
  - Adds two levels of hierarchical administration control
- Drone Remote Identification Protocol (DRIP)
  - Architecture document draft 2020-10-28

Current status

App:

- Scanning advertising IDs works as intended
- Database added to update positions of drones on a map
- When a drone is detected it will fetch the position and display it on the map

RPI:

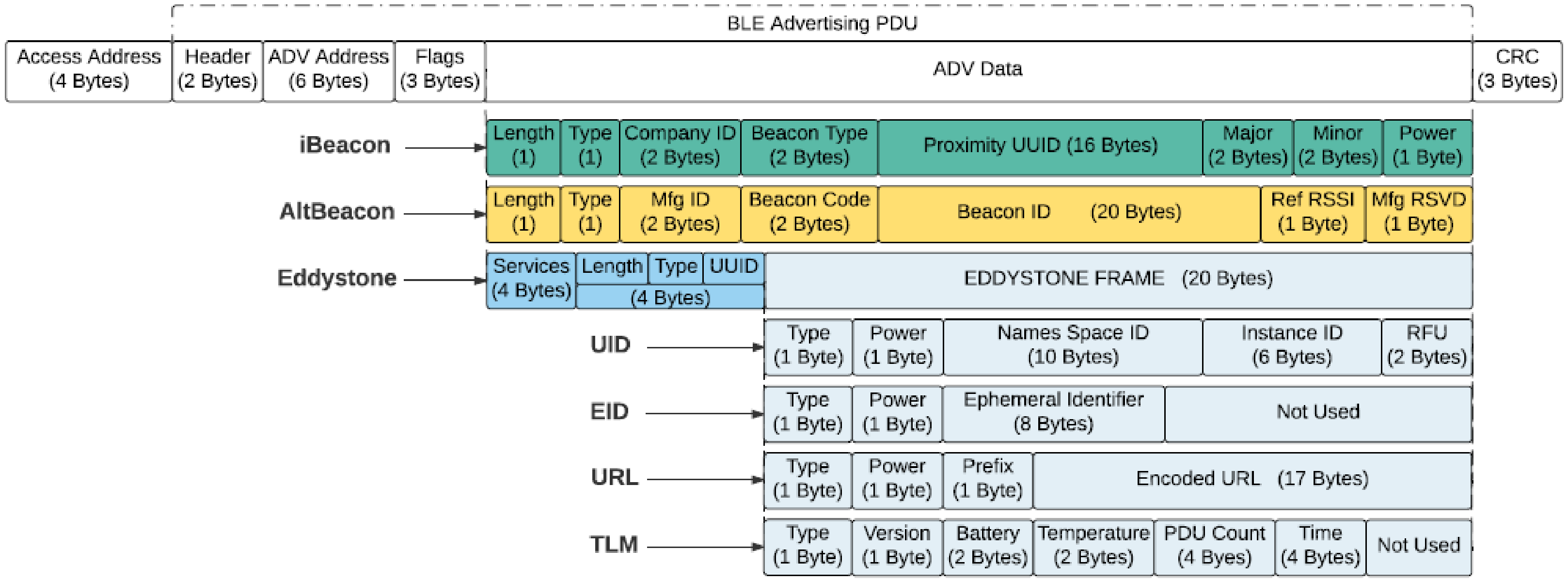
- A script that generates a HIT tag and broadcast it over Bluetooth

Raspberry Pi: Bluetooth advertising

- Two scripts
- One for configuring the RPI to the use the correct version of openssl and hitgen from openhip
- The other to actually broadcast the generated HIT

Broadcasting over Bluetooth

Multiple Beacon standards



BLE Advertising PDU

Access Address (4 Bytes) | Header (2 Bytes) | ADV Address (6 Bytes) | Flags (3 Bytes) | ADV Data | CRC (3 Bytes)

iBeacon → Length (1) | Type (1) | Company ID (2 Bytes) | Beacon Type (2 Bytes) | Proximity UUID (16 Bytes) | Major (2 Bytes) | Minor (2 Bytes) | Power (1 Byte)

AltBeacon → Length (1) | Type (1) | Mfg ID (2 Bytes) | Beacon Code (2 Bytes) | Beacon ID (20 Bytes) | Ref RSSI (1 Byte) | Mfg RSVD (1 Byte)

Eddystone → Services (4 Bytes) | Length | Type | UUID (4 Bytes) | EDDYSTONE FRAME (20 Bytes)

UID → Type (1 Byte) | Power (1 Byte) | Names Space ID (10 Bytes) | Instance ID (6 Bytes) | RFU (2 Bytes)

EID → Type (1 Byte) | Power (1 Byte) | Ephemeral Identifier (8 Bytes) | Not Used

URL → Type (1 Byte) | Power (1 Byte) | Prefix (1 Byte) | Encoded URL (17 Bytes)

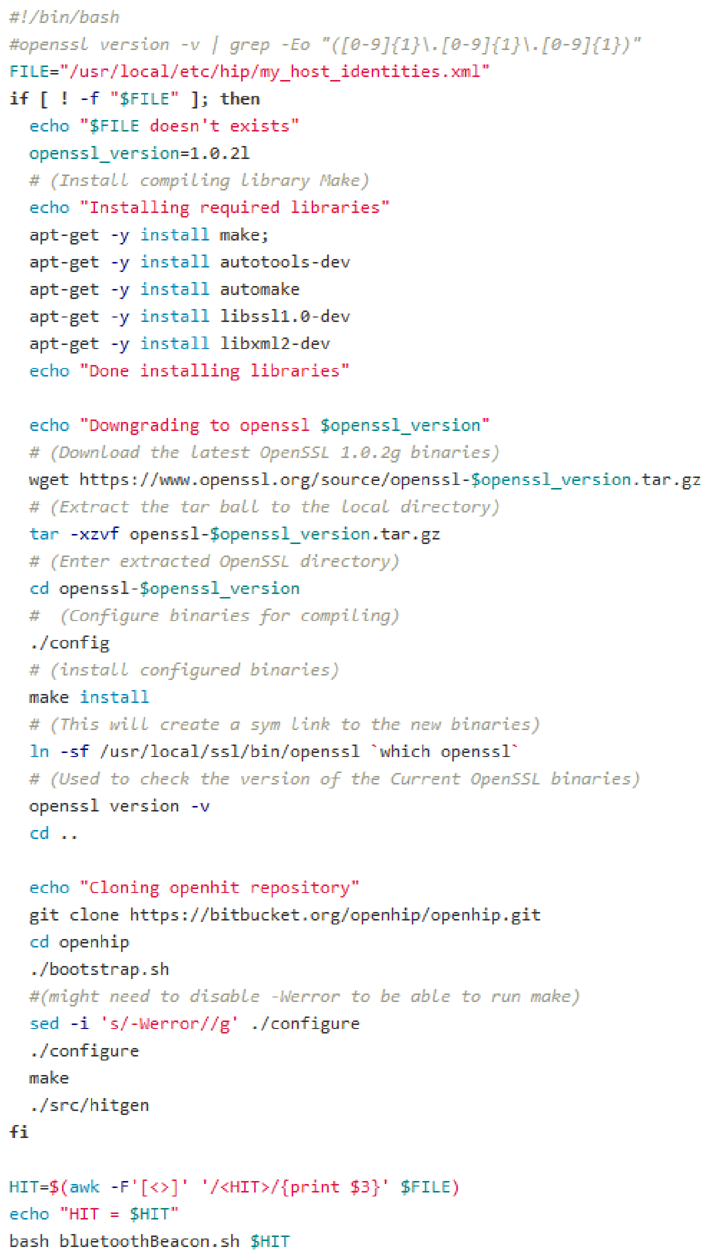
TLM → Type (1 Byte) | Version (1 Byte) | Battery (2 Bytes) | Temperature (2 Bytes) | PDU Count (4 Byes) | Time (4 Bytes) | Not Used

Figure 1. Hernández-Rojas DL, Fernández-Caramés TM, Fraga-Lamas P, Escudero CJ. Design and Practical Evaluation of a Family of Lightweight Protocols for Heterogeneous Sensing through BLE Beacons in IoT Telemetry Applications. *Sensors*. 2018; 18(1):57.

Broadcasting over Bluetooth

- Easy to switch between standards, they have very similar structure
- Easy to send out beacons using hcitool on the RPi, only requires a few lines of code
- Much easier to scan, there is already widespread support for scanning beacons on both Android and IOS devices
- Max range for BLE with Bluetooth 4 is around 91m
- Max range for BLE with Bluetooth 5 is around 548m
- Advertising extension enables 255 bytes payloads in Bluetooth 5

Scripts

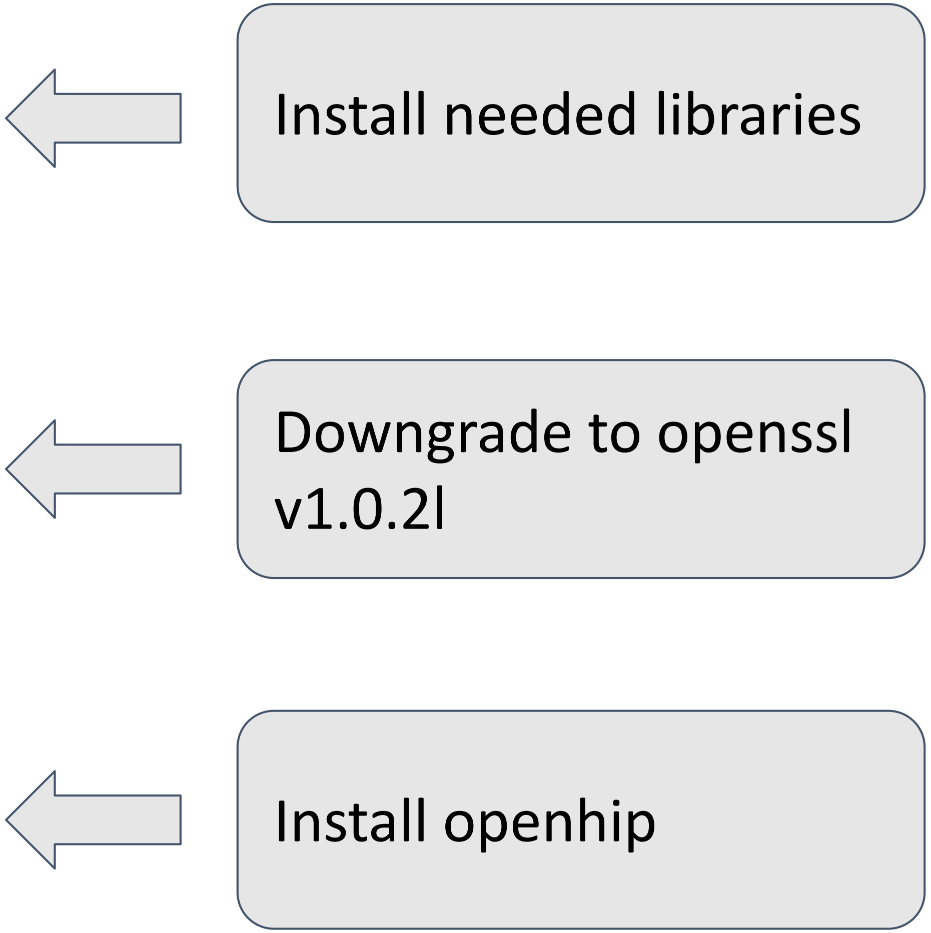


```
#!/bin/bash
#openssl version -v | grep -Eo "([0-9]{1}\.[0-9]{1}\.[0-9]{1})"
FILE="/usr/local/etc/hip/my_host_identities.xml"
if [ ! -f "$FILE" ]; then
  echo "$FILE doesn't exists"
  openssl_version=1.0.2l
  # (Install compiling library Make)
  echo "Installing required libraries"
  apt-get -y install make;
  apt-get -y install autotools-dev
  apt-get -y install automake
  apt-get -y install libssl1.0-dev
  apt-get -y install libxml2-dev
  echo "Done installing libraries"

  echo "Downgrading to openssl $openssl_version"
  # (Download the latest OpenSSL 1.0.2g binaries)
  wget https://www.openssl.org/source/openssl-$openssl_version.tar.gz
  # (Extract the tar ball to the local directory)
  tar -xzvf openssl-$openssl_version.tar.gz
  # (Enter extracted OpenSSL directory)
  cd openssl-$openssl_version
  #  (Configure binaries for compiling)
  ./config
  # (install configured binaries)
  make install
  # (This will create a sym link to the new binaries)
  ln -sf /usr/local/ssl/bin/openssl `which openssl`
  # (Used to check the version of the Current OpenSSL binaries)
  openssl version -v
  cd ..

  echo "Cloning openhit repository"
  git clone https://bitbucket.org/openhip/openhip.git
  cd openhip
  ./bootstrap.sh
  #(might need to disable -Werror to be able to run make)
  sed -i 's/-Werror//g' ./configure
  ./configure
  make
  ./src/hitgen
fi

HIT=$(awk -F'[<>]' '/<HIT>/{print $3}' $FILE)
echo "HIT = $HIT"
bash bluetoothBeacon.sh $HIT
```



Install needed libraries

Downgrade to openssl v1.0.2l

Install openhip

Web server

Used technologies

- REST-API
- MySQL
- PHPMyAdmin
- Docker

Fields

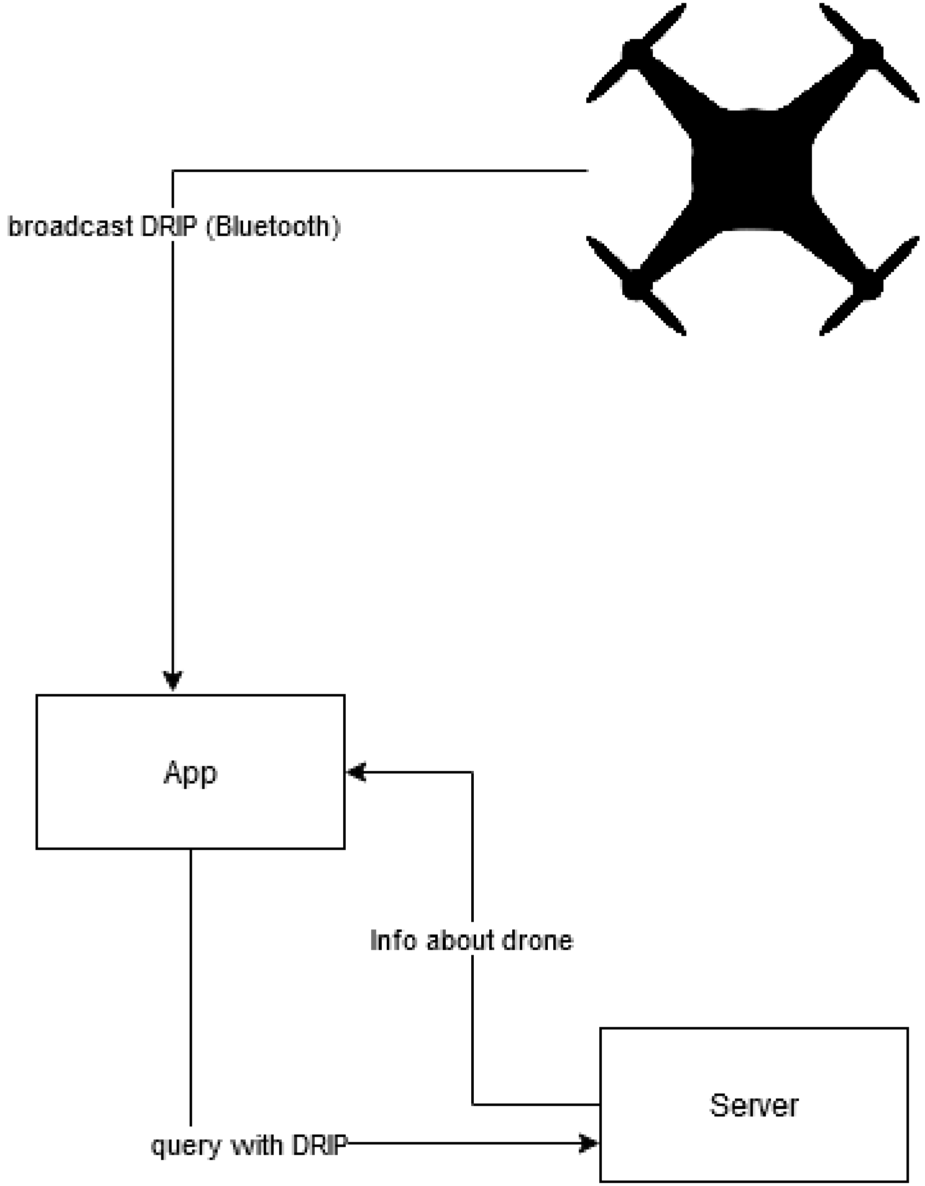
- ID
- DRIP ID
- Timestamp
- Model of aircraft
- Latitude
- Longitude
- Owner

Android application

- Bluetooth beacon parser
- Database lookup

Functions:

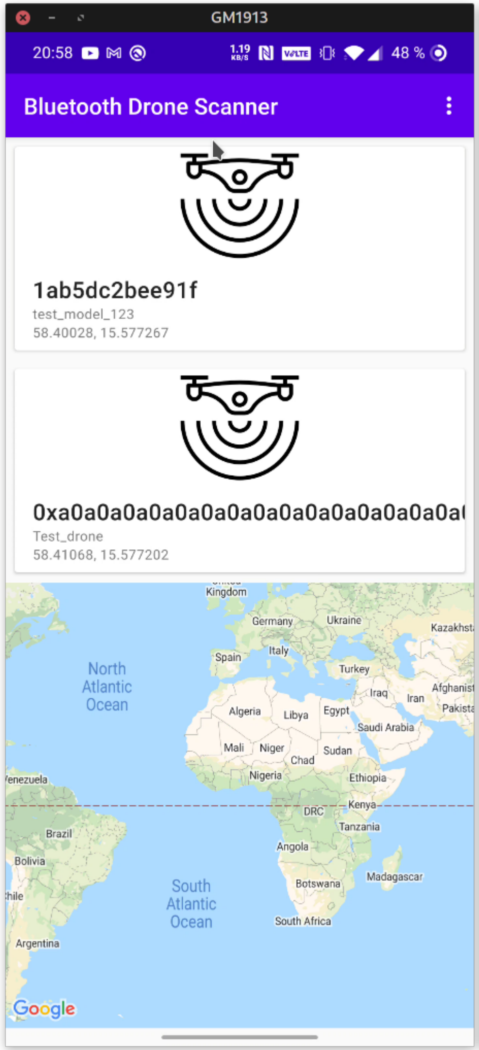
- Read Bluetooth LE beacons
- Draw found drone positions on map
- Draw drone flight path on map



Development

- Java and Gradle for build
- Altbeacon
  - Read BLE broadcasts
  - includes distance measuring
- OkHTTP
  - Send HTTP-requests
- Google Maps API
  - Draw maps and markers
  - Requires API-key from Google

Android (images)



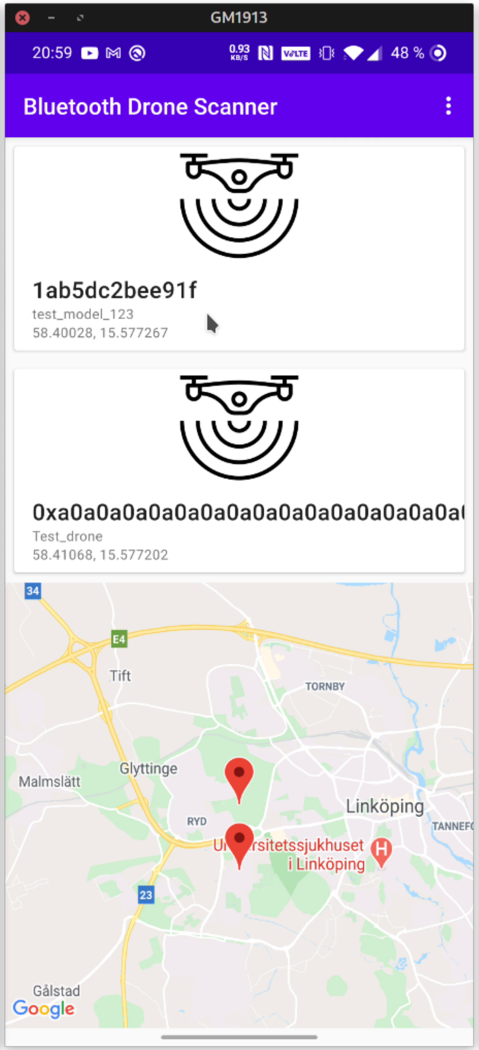
GM1913

20:58 1.19 48 %

Bluetooth Drone Scanner

1ab5dc2bee91f
test_model_123
58.40028, 15.577267

0xa0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a
Test_drone
58.41068, 15.577202



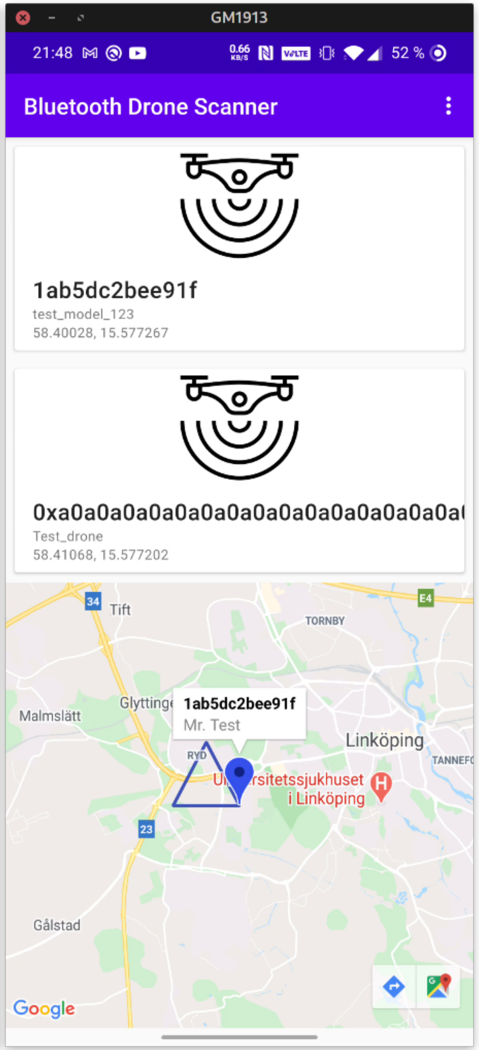
GM1913

20:59 0.93 48 %

Bluetooth Drone Scanner

1ab5dc2bee91f
test_model_123
58.40028, 15.577267

0xa0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a
Test_drone
58.41068, 15.577202



GM1913

21:48 0.66 52 %

Bluetooth Drone Scanner

1ab5dc2bee91f
test_model_123
58.40028, 15.577267

0xa0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a
Test_drone
58.41068, 15.577202

1ab5dc2bee91f
Mr. Test

Android Application (cont.)

Previous software

OpenDroneID

- Bluetooth
- Includes parser for Bluetooth messages
- Functionality for Maps, points and information about the drone inside the GUI

Attempt for wifi

- No low level access to broadcast messages without rooting phone
- Possible to connect to individual drones

Method to scan

- Lookup found BLE broadcasts with query to server
- Draw found drones on the map and place in list

Extending the Android Application

Possible new features

- Support for receiving DRIP using Wi-Fi
- Bluetooth 5.0
- Certificate when implemented in the sent broadcasts
- Visualize different types of information about drone based on receiver privilege

Conclusion

- A working prototype of broadcasting a HIT has been implemented.
- A lot of time was spent on research and finding ways around encountered problems
- Good starting point for future work to make full use of Bluetooth 5’s extended advertising.