DRIP Project Report - TDDE21

Abdullah Bin Zubair Hampus Rosenquist
Mirna Ghazzawi Younus Salman Yinan Wang

Examiner: Andrei Gurtov, *Supervisor:* Suleman Khan

December 16, 2022

Contents

1 Introduction

This report presents the work done in the course Advanced Project: Secure Distributed and Embedded Systems (TDDE21) at Linköping University during the fall of 2022. The project's goal was to further develop Linköpings University's open-source contributions to the Drone Remote ID Protocol (DRIP). The project was performed by five students at Linköping University and is a continuation of previous years students' work.

1.1 Project goals

This year's project goals were:

1. Update to draft-ietf-drip-rid-32.
2. Update to draft-ietf-drip-auth-17.
3. Implement draft-moskowitz-drip-secure-nrid-c2.
4. Test on Next Unit of Computing (NUC).
5. Check Bluetooth 5 status.
6. Check WiFi beacon status.
7. Test interoperability with other DRIP implementations.
8. Replace Google Maps with Open Maps in the Android app.

2 Background

This section presents the background of the different parts of the project.

2.1 DRIP

Drone Remote Identification Protocol (DRIP) is an architecture that provides secure and trustworthy remote identification for unmanned aircraft systems (UAS). The operation of the whole system is similar to the transportation system: there are two licenses, one for the car that indicates this car can run on the road legally, and one driver's license shows that this driver has the ability to drive a car. Similarly, there are several objects in the DRIP system: the registry, unmanned aircraft (UA), the operator of UA, and the observer. Both UA and operator need their respective certificates, which are preserved by the registry. In DRIP, a UA broadcasts its unique identifier through WiFi or Bluetooth, an observer can look for the information from the registry by using the identifier, and the registry will return the information about the UA as well as its operator. An identifier contains such normative requirements: length, registry ID, entity ID, uniqueness, non-spoofability and unlinkability [3]. This DRIP project also

refers to several documents published by the Internet Engineering Task Force (IETF), the specific version of the documents are: draft-ietf-drip-auth-17 (DRIP Entity Tag Authentication Formats and Protocols for Broadcast Remote ID) [13] and draft-ietf-drip-rid-32 (DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID)) [8].

The document draft-ietf-drip-auth-17 describes the way to add trust into the Broadcast Remote ID (RID) specification, which has been introduced in the DRIP Architecture [2]. This document defines the types of messages and associated formats that are sent within the Authentication Message.

The document draft draft-ietf-drip-rid-32 descirbes how the Hierarchical Host Identity Tags (HHIT) are generated and what information they should contain. HHIT is structured as follows: IPV6 prefix (max 28 bits), Hierarchy ID (HID) (28 bits), HHIT Suite ID (HHSI) (8 bits) and ORCHID hash (64 bits) [8].

The hardware used in the project is A raspberry pi 4 which represents the drone. A GPS module to get the location information and a pi Juice, which is a battery for the raspberry pi. Last an android app to receive the signals and show the information about the drone. The phone represents the observer.

2.2 Bluetooth advertising

Bluetooth is a technology for short-range wireless communication. Bluetooth advertising is one of its features that allows devices to broadcast information to any other devices in the local area. Furthermore, Bluetooth 4 and later offers a Low Energy mode suitable for battery-powered devices. In Bluetooth 5, new encoding modes were added. One of them was called coded PHY, which enabled communication at longer ranges, up to one kilometer, and larger packets, up to 255 bytes [1].

The Bluetooth stack is implemented in the official Linux kernel and is called BlueZ [11]. To interact with BlueZ, Bluetooth Host Controller Interface (HCI) is used. A useful interface for interacting with Bluetooth controllers on Linux is hcitool, which allows the user to issue HCI commands. What commands can be sent and how is specified in the Bluetooth Core Specification [1].

2.3 Google Maps for Android

Google offers a Software Development Kit for implementing Google Maps in Android applications [6]. To use this, a developer account must be created and personal API keys need to be retrieved. This API key is connected to the owner of the developer account and keys must be specified when compiling the Android application. This makes the distribution of the application outside of the Google Play Store inconvenient. Also, reliance on Google Maps may pose other challenges for an open-source application further down the line, such as costs.

An alternative tool is Osmdroid [10]. It is open source and does not require any API keys.

2.4 WiFi

Wireless Fidelity (WiFi) is a family of wireless network protocols, based on the IEEE 802.11 standard [7]. WiFi is used for local networking areas of devices and network accesses. It allows devices to communicate with each other and exchange data through radio waves with specific frequency signals instead of using wires.

WiFi beacon is one of the protocols that one can use for wireless communication. It allows devices to exchange data by broadcasting to each other within a specific range. Devices can be detected with the help of routers or access points. WiFi beacon contains all the information about the network and the beacon signals are transmitted periodically between 20ms to 65535ms [4], they serve to announce the presence of a wireless LAN and to synchronize the members of the service set.

For the WiFi beacon to work the hardware needs to have monitor mode configured. Monitor mode is a feature that some hardware has that allows the computer with a wireless network interface controller to monitor all traffic received on the wireless channel. And besides that, it allows the computer to perform packet sniffing, therefore the WiFi beacon script will be able to broadcast messages through WiFi.

3 Method

This section presents the methods used in the different parts of the project.

3.1 Pre-development

- Read and understand different documents to understand more about what DRIP is.
- Read what the previous years' students have already accomplished.
- Understand what the goals for this year are and discuss them with the supervisor and the examiner.
- Identify the different components of the project and divide the work into two subgroups.

3.2 DRIP drafts

- Attend the online hackathon with the others who are working on DRIP from the US. Discuss with them what needs to be done to update the code to meet the goal version of the drafts.
- Read and compare the current implemented version of each draft with the goal version of each draft.
- Identify the differences between the drafts' versions and search in code where it needs to be implemented.
- Implement the differences to meet the goal version of the drafts and perform tests.

3.3 Reciever Android application development

- Check the code of the application and try to get a general understanding of how it works and the different components of it.
- Identify where in the code is needed to be changed.
- Search and read about an alternative for Google maps.
- Implement the alternative maps by searching for similar alternatives to the functionality that Google maps had.
- Test the application and make sure it performs what it is supposed to do without any issues.
- Make it ready for being published on the Google play store.

3.4 Bluetooth sender development

- Check what can be the issue with the Bluetooth 5 dongle and solve it.
- Read and understand the code that is already implemented.
- Read about Bluetooth 5 and the extended advertising feature.
- Identify where in the code is needed to be changed to make Bluetooth 5 and extended advertising work.
- Implement the new commands and test them with the Android app.

3.5 WiFi sender development

- Understand the standards and the concept of monitor mode.
- Understand the wifi beacon code written by the group from last year.
- Implement wifi beacon broadcasting into transmitter application.
- Find a suitable wifi adapter with a chip that supports monitor mode.

4 Results

This section presents the results of the work done this year for the different parts of the project.

4.1 DRIP drafts

This part illustrates the conclusions about different drafts we have made throughout the whole project.
After the hackathon which the team attended halftime through the project, a good understanding of what is needed to update the drafts was gained.

4.1.1 draft-ietf-drip-rid-32

The transmitter has been updated according to the draft (rid-32) [8]. The format of the HHIT has been updated, see Figure 1. A new Prefix and Hash have been introduced in the drone_identities.xml file. The implementation of the verification of the hash has been updated so those correct parameters are passed to the cShake128 function according to the rid-32 draft.

Also, checks on the receiver side have been done to make sure it is following the rid-32.

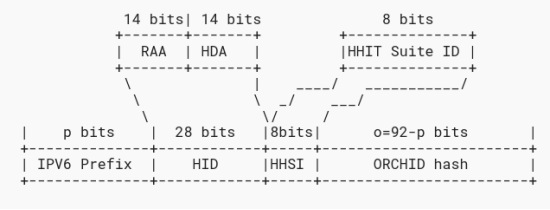


Figure 1: A screenshot of the HHIT from rid-32.

4.1.2 draft-ietf-drip-auth-17

The authentication draft (auth-17) is about the format of authentication messages being broadcast from the drone. There are four main types of authentication broadcast messages. Each type contains specific relative information, which makes it possible to authenticate the drone and increase trust in the broadcasted information. The functionality of Drip-Link and Drip-Wrapper was updated. New functionality for the Drip-Manifests was implemented. For Drip-Link, the proposed structure for endorsement broadcast was introduced as mentioned in draft-ietf-drip-auth-17 Appendix B [13].

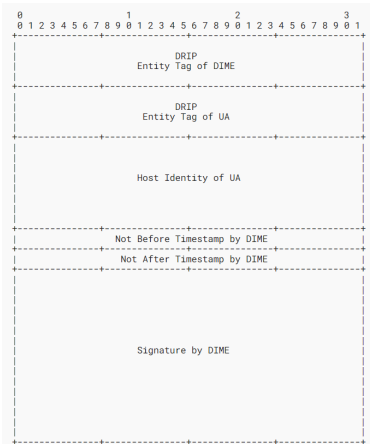


Figure 2: A screenshot of the Broadcast Endorsement: DIME, UA

Drip-Wrapper was updated as well to work with the new update, such as the introduction of Drip-Manifest, moreover, Drip-Wrapper structure was updated according to the draft-ietf-drip-auth-17.

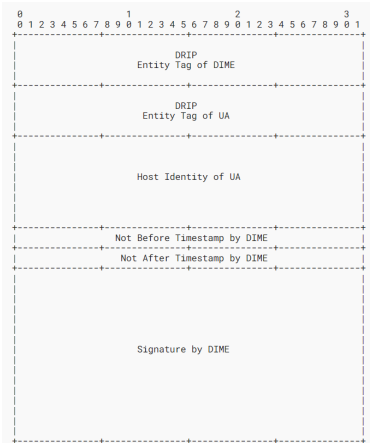


Figure 3: A screenshot of the Drip Wrapper over Legacy Transport

Drip-Manifest is implemented to broadcast the hashes of the location messages to increase trust and security. It was also implemented according to draft-ietf-drip-auth-17. The number of hashes in a single manifest, which we are sending in manifests for now, is discussed in Discussion 5.1.1.

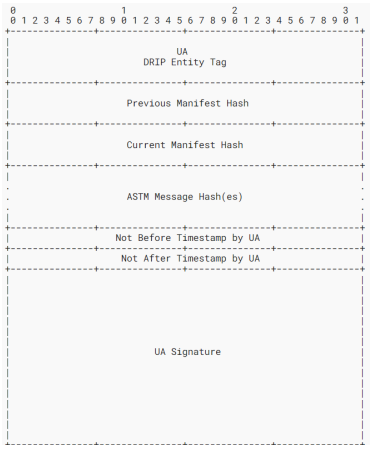


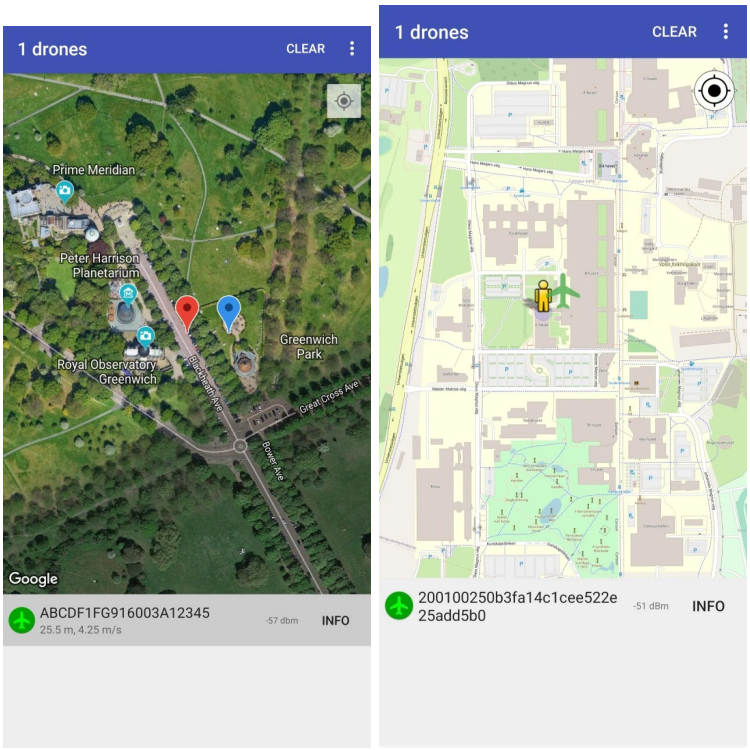
Figure 4: A screenshot of the Drip Manifest

We updated the session-id/flight-id for the drone to be set as the same as the Drone Host ID. We will discuss this in section 5.1.1. In the repository, we have "beacon_1.py" file in which all of the above functionality is implemented. As mentioned before, we did not have enough time to implement the functionality on the application on the android side. So when running the beacon_1.py, the android app will not receive anything due to the message format we implement being different from the old version and the app cannot decode that. Therefore we kept another file named "beacon.py", which is the original file that came from the last 2021 group, and we only added wifi function options to it.

4.2 Android application

An alternative has been successfully found, *osmdroid*, which is an abbreviation for OpenStreetMap and Android. It is an open-source tool that replaces Android's MapView class [10].
osmdroid has successfully replaced the already existing implementation of Google maps in the receiver application. Similar functionalities of Google maps have been found and replaced in the application. Meanwhile, the appearance is different from Google maps; as shown in figure 5 the osmdroid has a simpler view and lacks a satellite view. The current location icon is also different, as well as other icons but the other ones are not osmdroid own icons.
In figure 6 the view of the drone information in the application can be seen. No changes have been made from the previous year's work.

The Android application has been successfully updated to the latest Android SDK version and deprecated methods have been replaced. Most works was related to the update concerning runtime permissions for Bluetooth and location access. The application was also successfully published to the Google Play store.



(a) Google Maps. (b) Osmdroid.

Figure 5: Screenshots of the receiver Android application before and after the switch to Osmdroid.

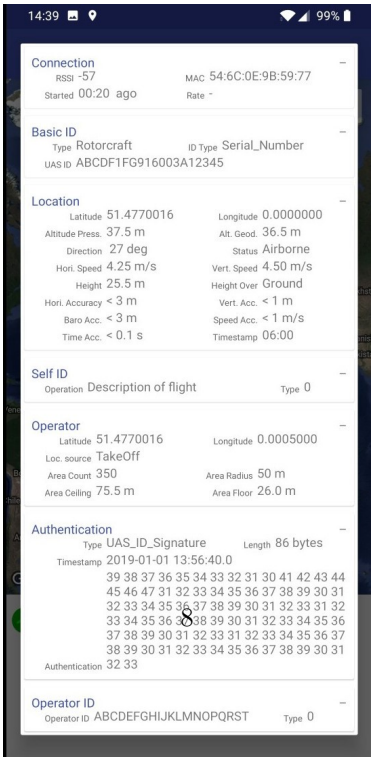


Figure 6: A screenshot of the information screen in the receiver Android app.

4.3 Bluetooth

A new appropriate Bluetooth USB dongle with support for Bluetooth 5 and long-range extended advertising support was acquired, called ASUS USB-BT500. Connecting the dongle to the computer running an up-to-date Linux distribution with a recent kernel, appropriate drivers were present and it worked without problems.

The beacon transmitter was successfully updated with the new HCI commands to make use of the extended version of Bluetooth advertising, and its long-range capabilities. All HCI commands, including the old ones, were documented thoroughly to make further development and debugging easier. The beacon transmitter can now make use of both Bluetooth 4 advertising and Bluetooth 5 extended advertising, by simply changing a flag: "-b 5".

4.4 WiFi

To be able to send messages through WiFi, Scapy is being used [12], which is a python-based interactive packet manipulation library that enables the user to send, sniff and dissect and forge network packets. First, an interface must be defined, in most cases, it should be *wlan0*. Its purpose is to allow Scapy functions to use wireless channels. After that, a frame object needs to be created, which will contain the actual message and additional information about the sender, for instance, the MAC address and more. Inside the frame, a dot11 object should be included, which is a class defined in the scapy library that contains the information. To send the message *sendp(...)* function is used which takes the interface and frame as parameters.

To allow the WiFi beacon script to work with our transmitter code, it must be integrated, so a new flag is added for running the transmitter that activates the WiFi beacon script. If it is activated, then the transmitter will use WiFi instead of Bluetooth as a transmitting medium. The flag is "–wifi"

A search for a suitable WiFi adapter is made. Before finding the adapter, a suitable WiFi chip must be researched first, which allows monitor mode. The chipset *Realtek RTL8812AU* is compatible with monitor mode, therefore any WiFi adapter which contains this chipset is configured with monitor mode. The best candidate according to our research is the adapter *Alfa AWUS1900*, which is an excellent choice for enabling the transmitter beacon script to send messages through WiFi. [5] is another choice of WiFi adapter that uses the same chipset.

4.5 NUC testing

Our supervisor, Andrei Gurtov, tested running the transmitter on a NUC device and could successfully run it for several days without any hiccups.

5 Discussion

This section discusses the results of the work done this year for the different parts of the project.

5.1 DRIP drafts

This subsection will separately discuss the drafts.

5.1.1 draft-ietf-drip-auth-17

In the previous work of the last group, they tried to implement the draft-ietf-drip-auth-01, which is the very early version of the authentication draft, although they did not finish the whole draft. We have discussed with Adam Wiethuechter(USA Team) and implemented the draft with his suggestions:

- The DRIP-Link should send a Broadcast Endorsement according to Figure 1, that contains the contents in such order: DRIP Entity Tag of DIME (DRIP Identity Management Entity), DRIP Entity Tag of drone, HI (host identity) of the drone, VNB timestamp, VNA timestamp, and the Signature generated by the registry.
- The DRIP-Wrapper contains a Location message. Drip-Wrapper structure is according to Figure 2. It contains UA DRIP Entity Tag, ASTM Message(s), Not Before Timestamp by UA, Not After Timestamp by UA and UA Signature. However, for now, we just place a single Location message in a wrapper.
- The DRIP-Manifest should be broadcasted according to Figure 3. It contains UA Drip Entity Tag, Previous Manifest Hash, Current Manifest Hash, ASTM Message Hash(es), Not Before Timestamp by UA, Not After Timestamp by UA and UA Signature. As discussed with USA Team, there can be a maximum of 11 hashes (8 bytes each), for now, we are placing one hash of one message. Moreover, the other team in the U.S. does not have Manifest fully interoperating.
- The Session ID is generated by a DIME during a UA DET, which is actually the DIME HI over UA DET/HI, according to section 5.1 of draft [13]. It is being used as a Session ID, registration. The session ID is used to make a difference between each flight, which is a recommendation, not a rule to follow. It is expected that the Session ID will be dynamically registered to a registry, which requires an Internet connection. In our code, we use "flight_id" as the session ID.

5.1.2 draft-ietf-drip-rid-32

After the changes between the different versions of the draft had been identified, it was still not easy to identify where to make changes; since the code did not

always match the old draft it was supposed to follow.

We successfully updated every change we identified in the transmitter, however, it is unfortunately not possible to verify with a guarantee at this time.

5.2 Android application

The goal for the Android application for this year has been achieved. While successfully implementing osmdroid it has been challenging with finding the same functionality as in Google maps. Some functionality was not there in osmdroid so an alternative way of implementing had to be searched for.

While implementing the new maps some issues from previous years' implementation had been encountered; e.g. the toggling between clicking on the current location and the drone location. The first time when starting the application, it works then it was not moving the screen to where the drone is. Since the entire map activity had to be re-implemented, a better understanding was gained and such bugs could be found and fixed.

Some issues were faced when trying to publish the application on Google Play store. Permission problems and the version of Android had to match. So location and Bluetooth permissions had to handle different kinds of Android versions to work and avoid the crash of the application. Another issue was when creating the aab file; it had to be signed with a key which caused a problem when trying to publish the application on the store.

5.3 Bluetooth

In the beginning, we had a lot of problems getting the new Bluetooth dongle up and running. First, it turned out that the Raspbian OS on our Raspberry Pi 4 missed a necessary kernel module. After updating to the most recent Rapsbian OS, the kernel module was present, but it was instead shipped with a too-old version of Glibc. Since upgrading this package manually usually causes a lot of new problems, a new OS had to be installed yet again. After installing Ubuntu 20.04, both the kernel and Glibc supported the dongle.

Implementing the extended advertising in the beacon transmitter required an understanding of the structure of HCI commands, but good information was found in the Bluetooth Core Specification [1]. A useful way we found to debug the HCI commands was to open up a terminal in the background with a Bluetooth monitor: "*sudo btmon*", which displayed the Bluetooth controller's status, responses and errors from any command currently being sent.

5.4 WiFi

The first half of the working time is spent researching different concepts surrounding the WiFi part. A literature reading about monitor mode and the required library for manipulating messages and sending them through WiFi.

The main challenge was to find a suitable chipset that allows monitor mode because the default chipset in the Raspberry Pi 4 does not allow monitor mode. Therefore, the messages are built correctly with the right size and content by using the Scapy library, but these messages are stuck inside the Raspberry Pi. By using an external WiFi adapter, the messages will be sent to other devices.

As mentioned in the result section, an external WiFi adapter was found, but the problem is that it is too late to order the adapter, therefore we assume that the WiFi script works if the adapter is being used.

6 Conclusion

This section presents the conclusions of the project, including challenges and a proposal for future work.

6.1 Challenges

In this section, the project's general challenges are presented. More specific challenges to each part of the project are discussed in the Discussion chapter above.

- Understanding the project and its parts took a lot of time in the beginning. This is because the information needed to understand the project was found in different sources.
- Difficulty to understand the drafts from the first time; we had to read them several times.
- It was not clear how much the previous group has implemented the drafts. So, it took us more time to understand what they have done and from where we need to start the implementation.
- We got stuck on some issues which slowed down the progress of the project in the first half of it.
- Information about specific parts was limited e.g. Bluetooth specific problems.
- The configuration phase was harder than we thought. A lot of libraries are missing and outdated if we try to follow the instructions for configuration in the readme file.

- Code challenges (No comments, uncompleted stuff that was mentioned as completed). Actually, the last group tried to implement the 01 version of the authentication draft but is not completed. Moreover, they did not mention what they have implemented and which part is left
- In the code, the names of the functions were not appropriately used. For example, in drafts, SAM types were not used according to the drip drafts.

6.2 Future work

This part presents future work that can be done by students in the coming years.

6.2.1 draft-ietf-drip-auth-17

Before going for the updated version of draft-ietf-drip-auth, the next group should know the work which is needed to complete for draft-ietf-drip-auth-17. We can break down the future work into two parts such as: sender/beacon.py and Android side. For the sender side, Drip-Link and Drip-Wrapper are completed on the beacon side according to the draft-ietf-drip-auth-17. However, for the Drip-Manifests we followed the structure of draft-ietf-drip-auth-17 for sending Manifest, but we are placing one hash in a manifest for now, as discussed with the USA team. So, We placed a single hash of 8 bytes in each manifest while the next group can increase these hashes to 11 where each should be of 8 bytes.

For the Android side, the next group needs to investigate the needed requirements according to the changes being done on sender/beacon.py side. All these changes are saved in a separate file named beacon_1.py file. After completing the updates on the android side, the next team just needs to run the beacon_1.py or wifi_beacon.py (wih suiteable wifi) file code instead of beacon.py.

6.2.2 draft-moskowitz-drip-secure-nrid-c2

Due to time restraints, no progress was made for this draft. This draft needs to be implemented.

6.2.3 WiFi

For future work, the specified WiFi adapter or a similar one that has the same chipset is needed to be ordered. A Testing phase should be done to make sure that the script and working and integrated correctly.

There is an alternative to finding a WiFi adapter, which is using Nexmon [9]. It is a firmware for patching framework that allows monitor mode inside the raspberry pi, so signals are sent by using the default WiFi chipset. This can be tested in the future but at your own risk as stated in their readme file, because it might damage the hardware.

References

[1] Bluetooth. *Bluetooth Core Specification v5.3*. 2021. URL: https://www.bluetooth.com/specifications/specs/core-specification-5-3/ (visited on 11/23/2022).

[2] Stuart W. Card et al. *Drone Remote Identification Protocol (DRIP) Architecture*. Internet-Draft draft-ietf-drip-arch-29. Work in Progress. Internet Engineering Task Force, Aug. 2022. 30 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-drip-arch/29/.

[3] Stuart W. Card et al. *Drone Remote Identification Protocol (DRIP) Requirements and Terminology*. RFC 9153. Feb. 2022. DOI: 10.17487/RFC9153. URL: https://www.rfc-editor.org/info/rfc9153.

[4] ddwart. *Advanced Wireless Settings*. URL: https://wiki.dd-wrt.com/wiki/index.php/Advanced_wireless_settings#Beacon_Interval (visited on 11/27/2022).

[5] ebay. *Realtek RTL8812AU USB Dual-band 2.4Ghz 5Ghz Wireless Dongle Adapter 1200mbps*. URL: https://www.ebay.com/itm/303442334248 (visited on 12/16/2022).

[6] Google. *Maps SDK for Android overview*. URL: https://developers.google.com/maps/documentation/android-sdk/overview (visited on 11/23/2022).

[7] "IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), pp. 1–3534. DOI: 10.1109/IEEESTD.2016.7786995.

[8] Robert Moskowitz et al. *DRIP Entity Tag (DET) for Unmanned Aircraft System Remote ID (UAS RID)*. Internet-Draft draft-ietf-drip-rid-32. Work in Progress. Internet Engineering Task Force. 35 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-drip-rid/32/.

[9] nexmon. *nexmon*. URL: https://github.com/seemoo-lab/nexmon (visited on 12/05/2022).

[10] osmdroid. *osmdroid*. URL: https://osmdroid.github.io/osmdroid/ (visited on 11/23/2022).

[11] Bluez Project. *Official Linux Bluetooth protocol stack*. 2022. URL: http://www.bluez.org/ (visited on 11/23/2022).

[12] scapy. *scapy*. URL: https://scapy.net (visited on 11/30/2022).

[13] Adam Wiethuechter, Stuart W. Card, and Robert Moskowitz. *DRIP Entity Tag Authentication Formats & Protocols for Broadcast Remote ID*. Internet-Draft draft-ietf-drip-auth-17. Work in Progress. Internet Engineering Task Force. 47 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-drip-auth/17/.