TDDE21 - DRIP Project Report

Suleman Khan, Markus Loborg, Robin Lidekrans, Lukas Rajala

December 2021

1 Introduction

This report will present the work done in the course TDDE21 Advanced Project: Secure Distributed and Embedded Systems at Linköping University during the fall of 2021. The goal of the project is to implement the Drone Remote ID Protocol (hereafter referred to as DRIP). The group working on the project consists of four students, and the work is a continuation of the work done by another group in the course the previous year.

The proposed Drone Remote Identification Protocol (DRIP) is an extension of the ASTM F3411-19 standard for Drone Remote Identification (DRI). As a result, a new set of requirements for all parties participating in a UAS network and their communication are being established.

Unmanned Aircraft (UA), operators, observers, and registries are examples of these entities, although they are not restricted to them. The goal of the new protocol is for observers to be able to detect any UAs in their airspace and execute lookups for information on those UAs.

Data supplied by any entity should be verifiable and trustworthy to the observer as well. In order to protect sensitive information, such as Personally Identifiable Information (PII), authorized individuals must be granted access to such data.

2 Background

This section goes over previous work and fields related to DRIP.

2.1 Open Drone ID

Open Drone ID is a project to provide beacon capabilities to drones so that they can be identified. It describes formats for messages and how they can be broadcasted over Bluetooth and WiFi. The Open Drone ID project resulted in the ASTM Remote ID specification, which is an official standard describing these messages. These messages include information about the drone, such as location, speed, ID, and such. It also includes messages for authenticating the identity of the drone, but it does not specify how these should be authenticated. Thus, the

DRIP protocol was created to create a trustworthy way to authenticate drones. DRIP messages live in the authentication messages of the ASTM standard.

2.2 DRIP

DRIP is an acronym for Drone Remote Identification Protocol which is a protocol with the goal to provide a trustworthy and secure way for remote identification of unmanned aircraft systems. The idea is similar to license plates on cars. Remote identification allows any user, both civilian and government workers, to look up and identify unmanned aircraft systems. For the identity of a UAS, DRIP uses a unique identifier, similar to a license plate, and broadcasts it using technologies like Bluetooth and WiFi. Any user with a device capable of receiving the broadcast can then remotely receive the identifier and use it to look up the identity of the owner and information of the drone. The DRIP protocol is specified in several documents written by the Internet Engineering Task Force [1]. The relevant documents for this project are draft-ietf-drip-rid-09 [2] and draft-ietf-drip-auth-01 [3].

Draft-ietf-drip-rid-09 [2], The document focuses on describing the use of a technology called Hierarchical Host Identity Tags (HHITs) to be used as a trustable identifier used for remote ID of UAS. HHITs are a hierarchical HIT which means that it's possible to assign an organization responsible for authentication of the HIT as well as enabling using DNS queries. The authors come up with the term DRIP Entity Tag(DET), which refers to HHITs within the context of UAS. HITs are unique IPv6 identifiers that are cryptographically secure and will not produce two identical HITs. Adding the hierarchy to the HIT and adding a HHIT registration process will provide global HHIT uniqueness. HHITs use Hierarchy ID(HID) to organize HITs into domains. HIDs provide two authorizations for the hierarchy, Registered Assigning Authority(RAA) and Hierarchical HIT Domain Authority(HDA). The RAA could be an organization that manages a registry of HDAs. At the same time, an HDA is any sort of third party that takes on the responsibility to provide services for enabling HIP.

draft-ietf-drip-auth-01 [3], describes how to add trust into the Broadcast RID. The document uses ASTM Authentication Message, which defines and standardizes new Authentication formats. This was needed to be able to provide trustworthy Broadcasting of the RID. See the ASTM F3411-19 standard for further information.

2.3 Bluetooth Advertising

Bluetooth Low Energy (BLE) was released with Bluetooth 4. BLE has an advertising feature that allows devices to broadcast packets up to 32 bytes. Bluetooth 5 further improved this with new encoding modes, one of them being called coded PHY, which enabled longer range and an increase in package size to 255 bytes.

Bluetooth is defined in the over 3000 pages long Bluetooth Core Specification [4]. The Bluetooth stack in the official Linux kernel is called BlueZ [5], which can

be controlled using the D-Bus API [6]. To make it easier to manage Bluetooth devices on Linux, tools like hcitool, bluetoothctl, and btmgmt allow the user to control BlueZ using the command line.

BlueZ communicates with Bluetooth devices using the Bluetooth Host Controller Interface (HCI). The hcitool allows you to send HCI commands, which can be found in the Bluetooth Core Specification, directly to the controller. For example, the following command is used to configure Bluetooth Low Energy to advertise every 100ms on all available channels:

hcitool -i hci0 cmd 0x08 0x0006 a0 00 a0 00 03 00 00 00 00 00 00 00 00 07 02

0x08 0x0006 is the HCI_Set_Advertising_Parameters command, and the following hexadecimal values are the command parameters. Tools like the bluetoothctl and btmgmt make it easier to manage Bluetooth devices by adding abstractions to the HCI. Instead of specifying which HCI command to run, these tools use a CLI interface with options such as "enable BLE advertising" or "advertise every 100ms".

2.4 WiFi Beacon

WiFi Beacon is a part of the 802.11 [7] protocol and is what allows devices to find WiFi networks by broadcasting data to devices within range. Access points or routers usually use it to allow devices to detect them, but it can also be used to broadcast a website link or arbitrary data. WiFi NaN WiFi NaN stands for WiFi Neighbourhood Area Network and is sometimes referred to as WiFi Aware though technically NaN is the actual standard, and Aware is when a device is certified to support it. WiFi [8] protocol uses modified 802.11 frames to allow direct communication between end devices without the need for an access point. There are two important main functions in WiFi NaN. The publish function and the subscribe function. The publish function allows a device to broadcast a "NaN Service." A service can be anything from a single line of text to identification data. The subscribe function allows devices to listen for a particular "NaN Service." If two devices are within range of each other and one is subscribed to a service that the other is publishing, they will connect and start sending any data needed to fulfill the service.

When a device starts using NaN, it listens for any "NaN cluster," which is a group of NaN devices that periodically broadcasts that their cluster exists. If it finds a cluster, it will join it. Otherwise, it will create its own cluster.

After joining a cluster, it will listen for any service it is subscribed to as well as publish its services. These messages will only be accepted by other devices in the cluster. Any other device not in the cluster will ignore them. There are also some synchronization messages that are sent within the cluster to keep all devices synchronized since the protocol has windows where different actions are taken. For example, there is a discovery window which is the time period where all devices are actively listening for new services.

2.5 Project Goals

The following goals were defined for the project:

- Update the project in accordance with the differences introduced in draft-ietf-drip-rid-09.
- Implement authentication as specified in draft-ietf-drip-auth-01.
- Test the project with the provided BT 5 dongle.
- Test battery powered Raspberry PI with GPS on a Phantom drone.

  In addition to these goals, two additional goal were specified
- Improve the documentation and readability of the project.
- Attempt to implement WiFi Broadcasting.

3 Method

The group started out by working together to understand the scope of the project by reading relevant documents and code from the previous year. After understanding the goal of the project, the work was divided between the group members. The identified areas of work were Bluetooth, the transmitter application and the receiver application. The working method for each of the areas can be seen below.

3.1 Bluetooth:

- Read and understand the Bluetooth implementation of previous year
- Read about Bluetooth 5 and how it works.
- Examine the provided Bluetooth dongle (see hardware)
- Attempt Bluetooth 5 Low Energy advertising
- Integrate Bluetooth 5 in the transmitter application

3.2 Transmitter application:

- Gain an understanding of previous code.
- Refactor to improve usability.
- Improve modularity to allow for multiple broadcasting technologies.
- Improve documentation of the code.

3.3 Receiver application:

- Understanding last year's receiver application
- Examining Open Drone ID receiver application
- Implementing DRIP authentication in the receiver application
- Adding new features

3.4 WiFi:

- Understand the standards
- Implement WiFi NaN in Python
- Implement WiFi Beacon script
- Integrate WiFi broadcasting in the transmitter application

4 Hardware

Raspberry Pi 4 running the Raspberry Pi foundation's Raspbian image was used for running the transmitter application. The built-in chips were used for Bluetooth 4 and WiFi beacon/NaN. For Bluetooth 5, the nRF52840 USB dongle from Nordic Semiconductor was used. For Bluetooth 5 development with the dongle, laptops running Xubuntu 20.04, Manjaro 21.1.4, and Windows 10 were used.

The receiver applications broadcasting code was developed with and tested on a plethora of devices. Specifics can be found in the README.md file here. As [9] for the extensions that were made to the applications, they were done using a Samsung Galaxy A52 5G. The code was written in Android Studios Arctic Fox on a computer running Windows 10. In addition to the A52, the code was also tested on a Samsung Galaxy A71.

5 Result

This section covers the result of the project.

5.1 Bluetooth

The first challenge was understanding the Bluetooth 5 protocol, what Bluetooth Low Energy is, and how applications are developed. The next challenge was to figure out how to use the Linux Bluetooth stack with the provided nRF52840 dongle. Nordic Semiconductor provides an application called nRF Connect [10] that can be used for controlling some of the Bluetooth capabilities of the chip, but advertising was not one of them. The group then attempted to use the dongle

with the Bluetooth tools provided by BlueZ, but the dongle was not recognized as a Bluetooth controller. Upon further research, a sample program [11] that allows the dongle to be recognized by BlueZ was found. The toolchain had to be set up using the instructions here [12] to compile programs for the dongle. Following the instructions for compiling and transferring programs to the dongle found here [13]. This allowed the Linux kernel to recognize the dongle as a Bluetooth device, and we could control it through the Linux Bluetooth stack, bluez [5]. However, neither bluetoothctl nor btmgmt could use the Bluetooth 5 extended advertising features. Attempting to set the PHY to LE coded resulted in error messages, and it was not possible to use the extended package size. Using hcitool with the command "hcitool -i -hci1 cmd 08 31 03 04 04" allowed us to set the preferred PHY to LE coded, but it was not possible to enable advertisement with this set.

A Linux transmitter example exists on the Open Drone ID GitHub repository here [14]. Running this program with the dongle did not work either. Using the program btmon we could monitor the calls being made to Bluez. There we could see that the Bluetooth 5 extended advertising HCI commands returned unknown errors. In addition, calling the "Read Advertising Features" command through bluetoothctl and reading the return value through btmon reports LE coded as unavailable. Therefore, it appears like the Bluetooth 5 features of the dongle is either not available to or not detected by, Bluez.

The next was to write native code for the chip. Attempts to compile and run examples on code using Bluetooth 5 advertisements, such as this one [15], were made. However, setting up the development environment for development beyond compiling simple example code proved non-trivial, and the group was unsuccessful in getting any example code to run. Attempts to write programs using Nordic Semiconductor's library to communicate with the chip[https://github.com/NordicSemiconductor/pc-ble-driver-py] were also made. However, despite following all available documentation for compiling and programming the necessary softdevice program to the dongle, it was not possible to program the dongle with the necessary software.

Having tried running the dongle as a USB device, writing native code for the dongle, as well as communicating directly with the softdevice on the dongle without any success, the group decided to use the remaining time for other parts of the project. However, tests done by the manufacturer suggest that with a proper configuration, a range of 1300 meters is achievable [16].

5.2 Beacon Script

A lot of the old code contained hard-to-read Bluetooth commands or encoded messages in hexadecimal strings, making it difficult to read or modify the code. Therefore, the group decided to clean up the code, modularize it, and make it easier to add new features.

pyOdid is a set of classes that behaves a lot like the Open Drone ID core library to create and encode messages. It provides classes and constants that allow you to generate messages that adhere to the ASTM standard easily. It

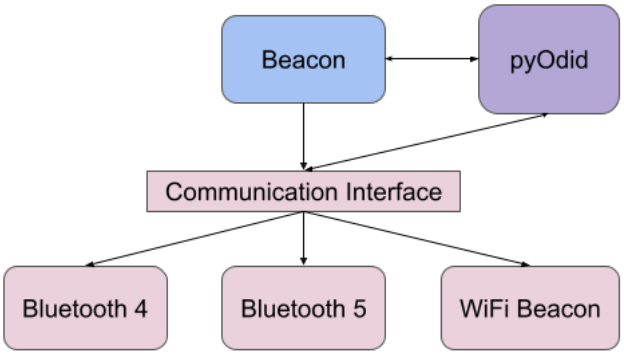

Figure 1: Beacon Script

also provides a way to encode these messages for broadcasting.

The communication interface is a simple layer that makes it easy to add additional broadcasting mediums and enable or disable interfaces not supported on the device. It is an interface to which the Beacon script can send pyOdid message objects to broadcast them over the enabled communication channels. This decouples the internal structure of the messages from the beacon script and the implementation details of the various communication protocols. Therefore, the beacon script is mainly concerned with the content of the messages and the timing of the broadcasts.

6 Authentication

The authentication works in multiple steps. First, a public and private key is generated using the Edwards-curve Digital Signature Algorithm. The drone is then registered using the public key and receives a HHIT from the registry. While the drone is used, it periodically broadcasts its id and authentication data. The authentication data is generated on the drone and is signed using the private key.

Any observer can verify the drone's identity with an application that can receive the messages broadcasted by the drone. Upon receiving the messages, it will fetch the public key from the registry that the drone registered with. The application will get the needed address for the registry from the ID (HHIT). This can either be done using an online server or a local database in situations where no internet connection is available. The authentication data is then verified using the drone's public key.

Edwards-curve Digital Signature Algorithm working:

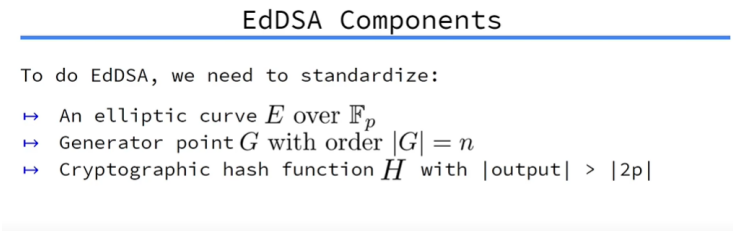

Figure 2: EdDSA Components

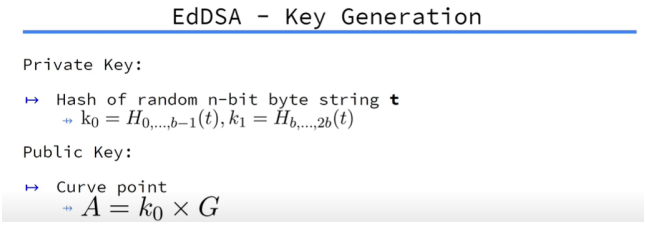

Figure 3: EdDSA Key Generation

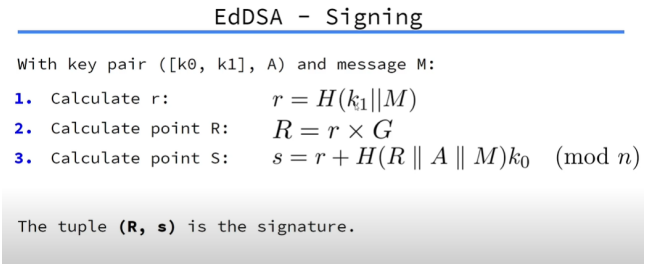

Figure 4: EdDSA Signing

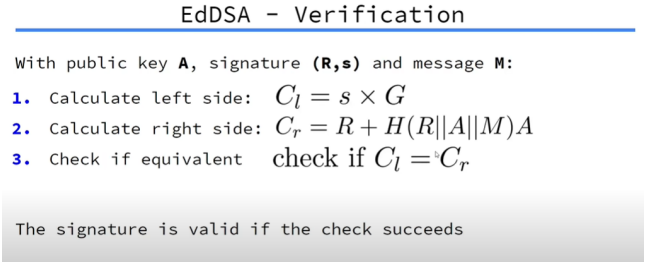

Figure 5: EdDSA Verification

7 Application:

For this project, a mobile application for android phones that can receive ASTM broadcasts and verify DRIP authentication data was developed. The application

is based on the open-source project Open Drone ID. The choice to base the application on this project was made due to the fact that it already had support for detecting beacons using Bluetooth 4 and 5, as well as WiFi NaN. This meant that the group only had to add support for DRIP authentication. This was done in several steps. First, the DRIP was added as a possible type to detect. After that, authentication using a static public key was added, meaning that it only worked with a specific drone. This was then expanded to work with any drone by fetching the public key from a specific server. The communication with the server is done over HTTPS to make it secure. Lastly, the possibility to manually enter the public key to a local database was added.

When the application starts, it listens for broadcasts on any of the supported broadcasting methods. If it finds one, it will parse it and display it in a list. If one clicks on the item, it will be focused on a map to where the drone is. If one clicks the info button, information about the drone, such as location, potential speed, acceleration, etc., is shown.

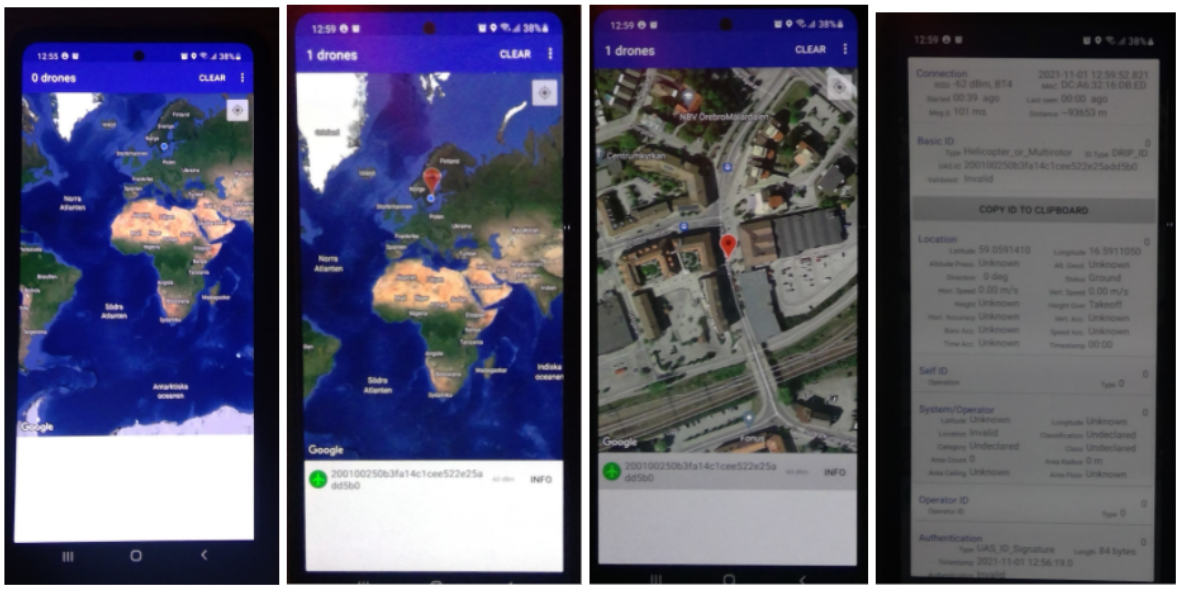

Figure 6: Application Demo

Information about the validity of the authentication is shown. The authentication can be valid, invalid, or unknown. The unknown state happens if the application is unable to connect to the server or when manual authentication fails. The reason it will still claim unknown when the manual authentication fails is to account for the fact that the user could have input the public key incorrectly. The application will report the authentication invalid if the public key taken from the server fails to verify the drone. If the application succeeds in verifying

the authentication, then no matter if the key came from manual entry or the server, it will report the authentication as valid.

7.1 WiFi Broadcasting

As part of implementing the DRIP protocol, we need to implement the various broadcasting technologies mentioned both in DRIP and in the ASTM standard. WiFi NaN For NaN, the group started developing a library for Python since there was no open-source implementation of NaN in any language available, and the only implementation the group could find was in Android. Halfway through the development, the group was made aware that the DRIP standard was moving away from NaN due to it working poorly with UAVs traversing an area, and development was therefore halted. This resulted in a half-complete implementation of NaN being created.

7.2 WiFi Beacon

For the beacon, a script was created for broadcasting a DRIP beacon with whatever data is given to the script. However, during development, it became clear that WiFi beacons were not compatible with the hardware available in the group. The script uses a library called scapy [17] to allow the creation and transmission of messages at the data link layer. It starts by creating a normal 802.11 beacon frame. It then modifies the relevant fields to make it a DRIP beacon. Data is then added, and the script starts to transmit the beacon.

The problem came in the fact that while scapy allows the code to create and transmit on the data link layer, the actual WiFi chip on the hardware does not support transmitting messages directly to the data link layer. So it will look like the script is running and transmitting, but in reality, nothing leaves the device. It is sent to the wifi-chip, which promptly discards it without notice or warning. This would also affect WiFi NaN if that code ever got to the transmitting stage as well.

8 Discussion

8.1 Update to draft-ietf-drip-rid-09 and draft-ietf-drip-auth-01

While improving the code from last year's project the group made sure to read the drafts to make sure that it followed the rid-09 and auth-01 drafts so that it is doing properly. Similarly when extending the Open Drone ID app the group based the work on previous year in combination with the documents from IETF workgroup. Since the Open Drone ID already supported receiving WiFi beacons and WiFi NaN, and it is included in the ASTM and DRIP standard, it was decided to try and get the broadcasting part to work as well.

8.2 Test BT 5 dongle

All attempts at getting Bluetooth 5 to run were unsuccessful. We did however test the Bluetooth 5 dongle, and thus feel like we succeeded with this goal. The Bluetooth standard is very hard to grasp, while at the same time there is not a whole lot of documentation available on the internet describing how to use any of the tools meant to make Bluetooth development easier. This meant that a lot of time spent on trying various things and trying to get them to work could have been avoided. Once we finally grasped the concepts and tools, it was quite easy to see that the dongle was not working for our purposes. Unfortunately we came to this conclusion at a late stage in the project, and since we had no backup device prepared and since the dongle does not have many of the features one would expect when doing embedded development, it was decided to stop working on Bluetooth 5 and focus on finishing the other parts of the project.

8.3 Test battery powered RP with GPS on a Phantom drone

Due to the aforementioned Bluetooth 5 issues, we did not carry out this step. As mentioned in the result chapter, range tests using the nRF52840 chip indicates a range over 1.3km, suggesting that using the chip with an actual drone could allow for DRIP authentication over quite long ranges.

8.4 Improve the documentation and readability of the project

The work and attempts of previous year being somewhat lacking in documentation, the Bluetooth 5 standard being very hard to read, the DRIP specifications being very abstract and hard to grasp, as well as the lack of both documentation and examples on how to do Bluetooth development on Linux meant that a lot of time in this project was spent in the dark trying to get things to work or understand how cryptic one-liners work. As a result of this, one of the priorities of this project was to document the solutions we tried and make sure that the produced code was easy to follow. The transmitter application now uses a library with named fields for configuring the content of the broadcasted messages, and the hexadecimal commands used for controlling the Bluetooth 4 controller have been bundled together in a communication library to make the code easier to read and understand.

8.5 WiFi

As mentioned earlier there was an issue with the WiFi chip not supporting transmission directly on the data link layer. This can be quite easily fixed by simply getting a WiFi adapter that supports this and is compatible with the OS used. Then you simply need to tell the script to use the adapter instead of the built in chip.

Support for data link transmission is called monitor mode. Therefore, you need to find a WiFi adapter that uses a WiFi chip that supports monitor mode. Note however that this is not something that is written among specifications on the WiFi adapters. So to find a good adapter might take a lot of searching. One way is to search for the actual WiFi chips that support it and then find what adapters use those chips.

9 Future Work

Here we present the areas we think that future groups should focus on.

9.1 Bluetooth

As mentioned in the report, the nRF52840 dongle does not appear to work as a Bluetooth 5 dongle with the BlueZ Bluetooth stack. Therefore if this dongle is to be used one would most likely have to write native code for the chip. The dongle is not suited for this purpose since it does not have any debugging features, and it might therefore be suitable to use the nRF52840 development kit. Another solution would be to use a Bluetooth 5 dongle with Extended Advertising support that is confirmed to work with the BlueZ Bluetooth stack. Once you have a working Bluetooth 5 controller, it should prove trivial to integrate it in the transmission application.

9.2 WiFi

Get a WiFi adapter that supports monitor mode. Then ensure the beacon script actually does what it is supposed to do. After that one could attempt to complete the NaN if one wanted to, or potentially find an open source version that can be used.

9.3 Application

Ensuring the application is up to date with any new standard. Possibly try and create an application for iphones. However note that as of now iphones do not have built in support for WiFi NaN.

9.4 DRIP

The drafts that the group have been working on have been replaced with newer versions and the group hasn't done a thorough check to see what needs to be replaced but that could be a good start for anyone who wishes to get started with our project.

10 Conclusion

It has been very interesting to work on cutting-edge technology and being among the first to implement something. While we wish to have produced something more tangible, a lot of time was spent trying to get things like Bluetooth 5 and WiFi NaN to work. The lack of good documentation and examples held us back at times, which is why we felt it was important to spend extra time documenting our findings. We hope this report can be of use to anyone wishing to continue working on this project, and help them avoid falling into some of the traps we fell into.

10.1 Appendix: Project overview

A gitlab group containing the repositories used in the project can be found at https://gitlab.liu.se/tdde21-drip-2021 (feel free to contact any of the group members if you need access). The available repositories are as follows:

- DRIP Android Observer - The observer application.
- DRIP Backend - The backend used for authentication.
- DRIP Beacon - The transmitter application.
- DRIP Iroha 1.1.3 - A custom version of Iroha used for storing the location of the drone using an Iroha blockchain (not part of this project).
- Python Nan - Standalone library implementation of NaN for Python (partially completed).
- Installation guidelines are available in the README-files of the repositories.

References

[1] IETF, "Drone remote id protocol (drip)," Available at https://datatracker.ietf.org/wg/drip/about/ (2021/12/03).

[2] A. W. A. G. Robert Moskowitz, Stuart Card, "Drip entity tag (det) for unmanned aircraft system remote identification (uas rid)," Available at https://datatracker.ietf.org/doc/draft-ietf-drip-rid/09/ (2021/09/08).

[3] R. M. Adam Wiethuechter, Stuart Card, "Drip authentication formats," Available at https://datatracker.ietf.org/doc/draft-ietf-drip-rid/09/ (2021/06/18).

[4] Bluetooth, "Drip authentication formats," Available at https://www.bluetooth.com/specifications/specs/core-specification/ (2021/11/18).

[5] BlueZ, "Release of BlueZ 5.62," http://www.bluez.org/, 2021, [Online; accessed 11-September-2021].

[6] ——, "Bluez git," https://git.kernel.org/pub/scm/bluetooth/bluez.git/about/, 2021.

[7] D. Stanley, "Ieee 802.11 tm wireless local area networks," *The Institute of Electrical and Electronics Engineers, Inc.(IEEE).*

[8] Developers, "Wifi Aware Overview," https://developer.android.com/guide/topics/connectivity/wifi-aware, 2021, [Online; accessed 07-November-2021].

[9] A. Gurtov, "Drip drone observer," https://gitlab.liu.se/tdde21-drip-2021/drip-android-observer/-/tree/master, 2021.

[10] Nordic, "nRF Connect for Desktop," https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop, 2021, [Online; accessed 17-November-2021].

[11] ——, "nRF Connect for Desktop," https://developer.nordicsemi.com/nRF˙Connect˙SDK/doc/latest/zephyr and-debugging, 2021, [Online; accessed 18-November-2021].

[12] ——, "nRF Connect for Desktop," https://infocenter.nordicsemi.com/index.jsp, 2021, [Online; accessed 18-November-2021].

[13] Bluetooth, "Bluetooth: HCI USB," https://developer.nordicsemi.com/nRF˙Connect˙SDK/doc/latest/zephy 2021, [Online; accessed 02-December-2021].

[14] friissoren, "transmitter-linux," https://github.com/opendroneid/transmitter-linux, 2021.

[15] M. Afaneh, "Bluetooth 5 Advertisements: Everything you need to know," https://www.novelbits.io/bluetooth-5-advertisements/, 2021, [Online; accessed 27-November-2021].

[16] jornbh, "Nordicsemiconductor," https://github.com/NordicSemiconductor/pc-ble-driver-py, 2021.

[17] Scapy, "Scapy Project," https://scapy.net/", year = 2021, note =.