Spring 5-1-2023

Radio Fingerprinting of UAVs: An Exploration of Zigbee Data Collection

Arielle Monson
University of Nebraska-Lincoln

Radio Fingerprinting of UAVs: An Exploration of Zigbee Data Collection

An Undergraduate Honors Thesis
Submitted in Partial Fulfillment of
University Honors Program Requirements
University of Nebraska-Lincoln

by
Arielle Monson, BS
Computer Science
School of Computing

May 2023

Faculty Advisors:
Brittany Duncan, Ph.D.
Nirnimesh Ghose, Ph.D.

ABSTRACT

Radio fingerprinting creates a way to uniquely identify devices based on slight variations in the signal they transmit. These variations occur due to fluctuations in the manufacturing processes of wireless hardware. Prior work in this area has focused on IoT devices to determine the best way to translate this practice into a network security context. Here, we extend this prior work to utilize radio fingerprinting to identify signals from UAV devices. We used a UAV device with a Zigbee antenna to collect data, which supports the possibility of identifying these devices through radio fingerprinting. This paper should serve as a starting point for future researchers to replicate the data collection procedure in order to test Zigbee UAV data against current radio fingerprinting protocols. The references section contains a link to a GitHub repository with code and sample data for replicating the process.

Key Words: UAV, Zigbee, Radio Fingerprinting, Networks

TABLE OF CONTENTS

1. INTRODUCTION

Background

Radio fingerprinting is a practice that utilizes process variations during the manufacturing of wireless hardware in order to distinguish devices' radio signals from each other. These variations include but are not limited to: I/Q imbalance, phase noise, frequency or sampling offset [3], and harmonic distortions [4]. Fingerprinting operates at the physical layer and provides an alternate authentication method for devices when the more traditional cryptographic methods are unavailable [1] due to compatibility concerns, limited resources, or compromised credentials. Creating models for this authentication method, however, is a challenging task. One must consider the robustness and accuracy of their data collection schemes and how much time is required to train the system to identify these devices. When it comes to creating models that handle these items, Cekic, Gopalakrishnan, and Madhow [2] ask how to make a system only learn the desired information rather than what it deems to be easiest. We extend this work to discuss how to handle the accuracy and robustness of radio fingerprinting systems.

With additional research efforts, radio fingerprinting could provide a more reliable method of authenticating devices. The difficulty of mimicking a specific hardware imperfection makes duplication of signals more complicated for adversaries to spoof data or for one device to pretend to be another. Current research on radio fingerprinting is primarily being done on IoT devices using IEEE 802.11 (WiFi) standards [8]; however, researchers could include standards such as IEEE 802.15.4 (Zigbee) [9] and devices such as UAVs by expanding these efforts.

Current Practices

Researchers are currently exploring a couple of different methods of radio fingerprinting. The first of these methods includes using deep learning algorithms to train a system to match signals based on hardware differences [1][4]. Ongoing work seeks to optimize this system's accuracy and find how often they must retrain models to identify machines based on their signals. Other researchers are also using deep neural networks (DNN) to attempt radio fingerprinting of IoT devices. According to Cekic, Gopalakrishnan, and Madhow [2], an advantage of using DNNs is their lack of need for explicit training models to identify signals. Al-

Shawabka et al. [3] determined that convolutional neural networks (CNN) were sufficiently accurate for data collection due to their ability to create multidimensional mappings and compatibility across various devices and technologies.

While research on radio fingerprinting shows positive results, researchers are exploring many options to improve the accuracy and robustness of radio fingerprinting for IoT devices. Some are finding that they have to retrain deep learning models daily in order to keep up with the changes in signals [1], while others are finding that the channels used for transmitting signals are causing increased inaccuracy in the system [3]. To improve upon these findings, researchers are introducing methods of equalizing data [3], averaging data over input values rather than different types of machines [2], and exploring models for radio fingerprinting which do not require deep learning.

Al-Shawabka et al. [3] express the need for a standard, diverse dataset to be used by researchers. They collected bursts of data over ten days from twenty different devices placed in various environments. They describe that each burst contains about 30 seconds of transmission, and each occurs approximately one minute after the previous one. Having an openly available data set to use as a standard, according to Al-Shawabka et al. [3], should help to improve the accuracy with which research on radio fingerprinting is moving forward. Their work focused on utilizing the IEEE 802.11 standard for data collection. We extend this work to provide a data collection strategy for devices using the IEEE 802.15.4 standard.

Challenges

Challenges in this research area are primarily related to the ability to connect to a device and accurately collect data from it in a timely fashion. As Al-Shawabka et al. [3] noted, even the channel used for sending data could cause inaccuracies in identification models as the equalized signal can make the desired imperfections harder to distinguish. The training of deep learning models has also proved to be challenging for radio fingerprinting due to the need for retraining on a regular basis [1]. Cekic, Gopalakrishnan, and Madhow [2] propose the challenge of modeling the collected material to receive the desired data from a device.

Regarding Zigbee and UAV devices, the challenge of device connection is significant. Zigbee devices do not use IP addresses, so one must use a different method to find the signal

being sent. UAV devices are meant to be mobile devices, so timely, accurate data collection and a fast enough connection to find the signal are significant factors that researchers must consider.

Equipment Requirements

In order to replicate or extend work on this project, one must first meet specific software and hardware requirements. First [7], a computer running Ubuntu 20.04.4 or better is necessary to download GNU radio 3.11 and the projects containing the Out of Tree blocks required for the Zigbee protocol. These projects include the gr-foo and gr-ieee802-15-4 libraries linked in the Wime installation instructions [6]. The machine also needs an updated version of Python; the laptop used in this work was running Python 3.8.10. Wireshark must also be downloaded to observe the data collected. A USRP with a VERT2450 antenna and a Zigbee UAV device are also required. XCTU and QGroundControl (Figure 2.4 and Figure 3.2) must also be downloaded to verify the connection to the Zigbee UAV. The Zigbee device used in this project was only the brain and the controller chip for ease of transport and use for data collection (Figure 1.1).

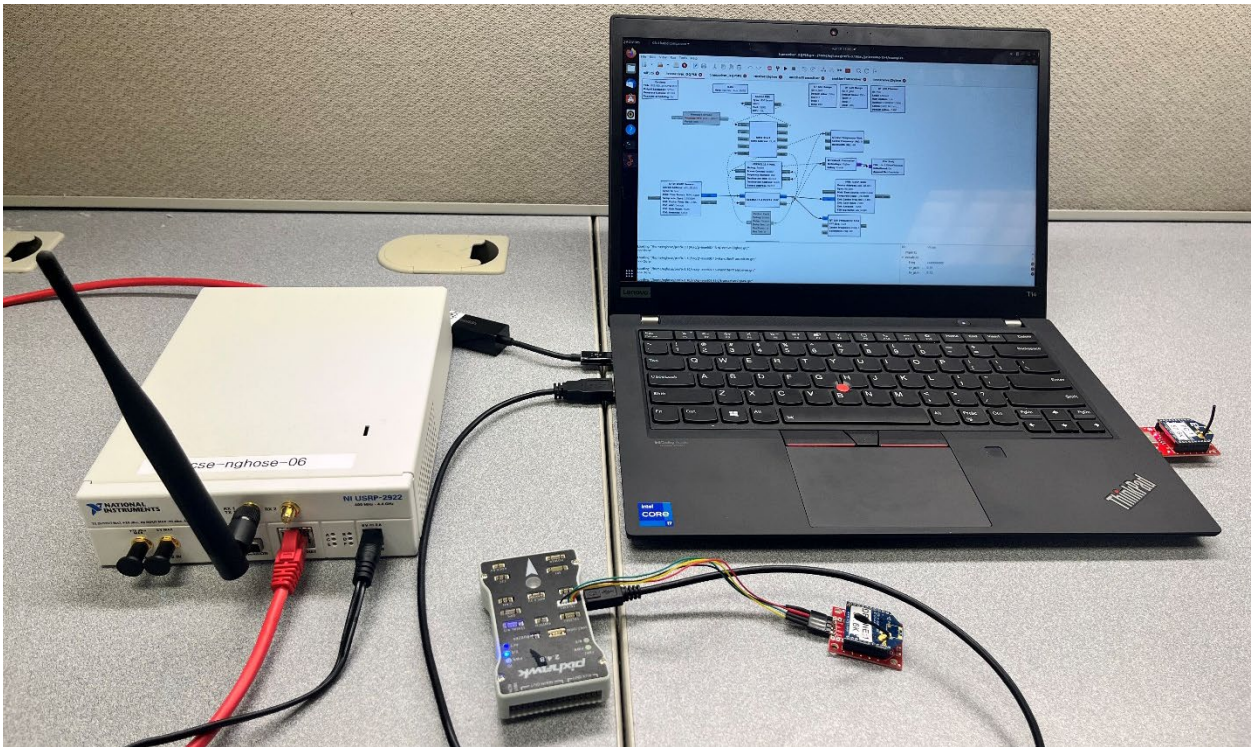

Figure 1.1 Final Equipment Setup

2. METHODOLOGY

Wime Zigbee Transceiver

Core to this project was an inspection of the Zigbee transceiver flowgraph provided by Wime [5]. This flowgraph was designed to use a single USRP device to send and receive test messages and export data into the Wireshark application. We made edits to this flowgraph (Figure 2.1) to see additional information with a GUI Frequency Sink block and to test how sending more than one message affected the output by adding additional Message Strobe blocks, expanding the values in the RIME Stack block, and adding additional Socket PDU blocks.

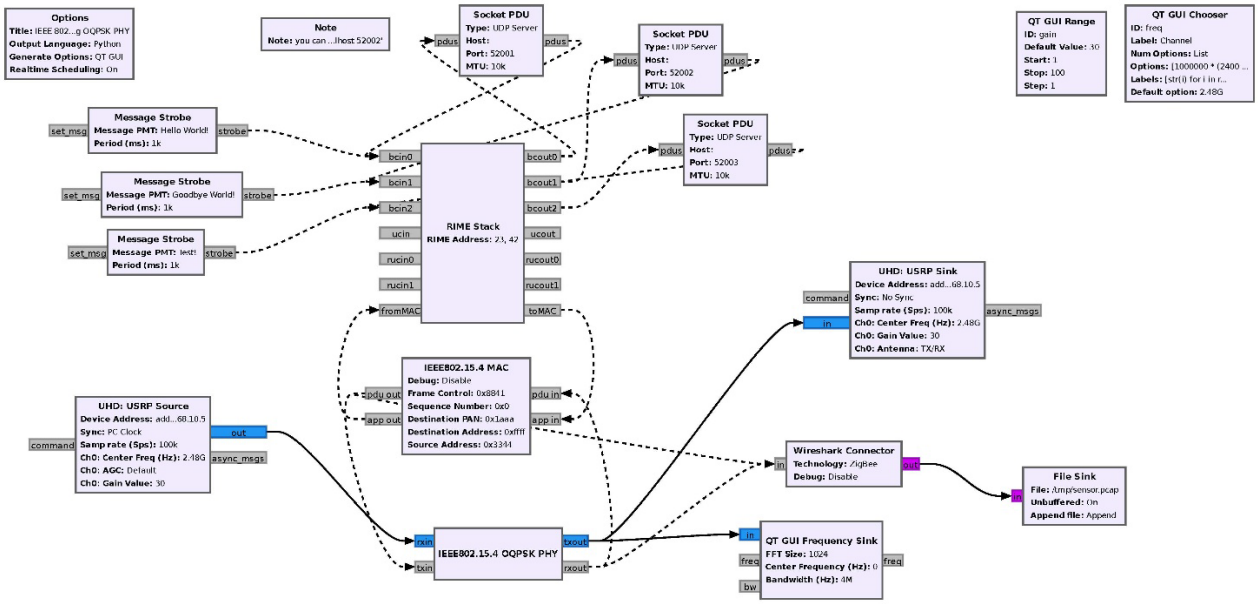

Figure 2.1 Edited Wime Transceiver

This flowgraph includes multiple custom out-of-tree blocks that require additional downloads to use. The RIME stack block provides a network layer to connect to transmitter ports. It determines which channels to broadcast to and provides addresses for the Socket PDU blocks to connect to when sending messages. This block also provides a loopback connection to the MAC block. The IEEE802.15.4 MAC block controls frame rate and determines which sources and destinations are allowed to connect to the flowgraph. Finally, the IEEE802.15.4 O-QPSK PHY block provides the physical layer to connect our source to our sink blocks. This

block must be obtained by first running the appropriate flowgraph provided with the gr-ieee802-15-4 package download.

There are two options to run this flowgraph in GNU Radio. The first is the standard way of clicking the play button at the top of the application and allowing it to run that way. However, when using Wireshark, you cannot see the data presented in real-time with this method, as it will not open automatically. A better option is to edit the transceiver.sh file provided in the gr-ieee802-15-4 download apps folder to run with the correct file path. Then you will run "./transceiver.sh" in the terminal to run the flowgraph, open Wireshark, and view the data that the USRP has pulled into the output file in real-time.

Transmitters and Receivers

To learn more about how the out-of-tree blocks were working in Wime's project, we split the functionality between individual transmitters and receivers. Because the Wime flowgraph's out-of-tree blocks combined these functionalities, it was difficult to determine where the split between the receiver and transmitter was supposed to be. Ultimately, I came to the flowgraphs shown in Figure 2.2 and Figure 2.3.

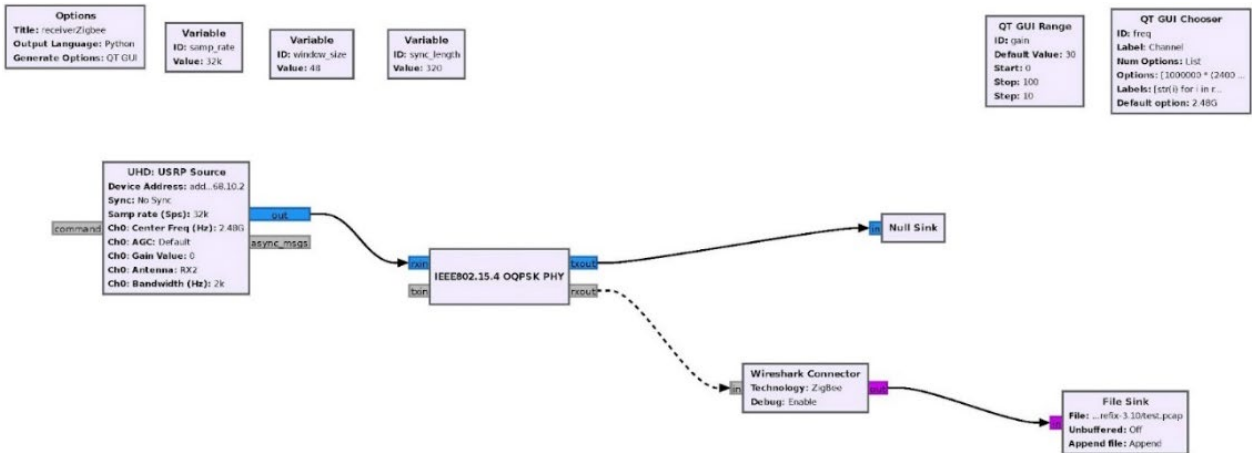

Figure 2.2 Receiver

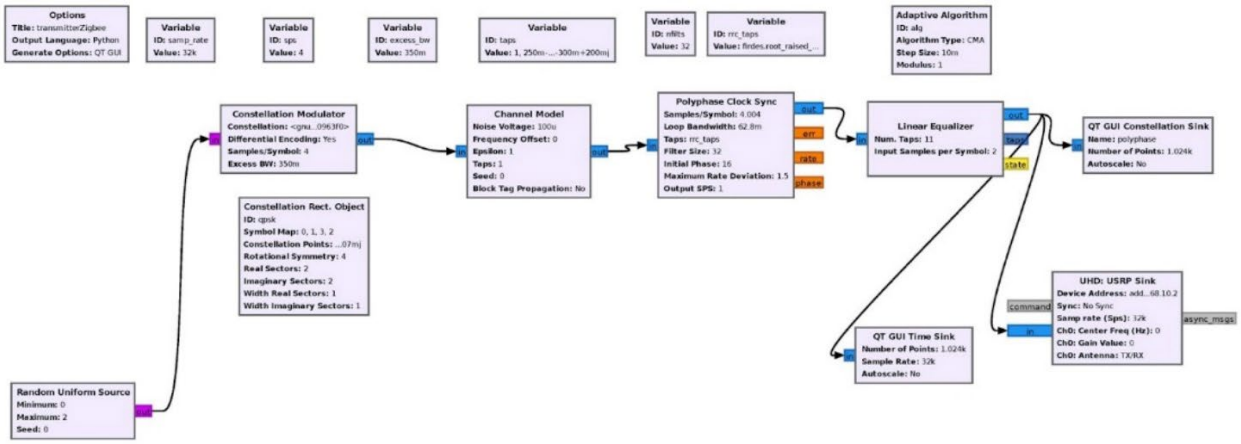

Figure 2.3 Transmitter

Each flowgraph appeared to work individually with an examination of the frequency graphs, but it was unclear if they were working together to send and receive messages.

Zigbee UAVs

As this was the first time the GNU Radio download used here had connected to a Zigbee device, it did not have the same configuration set up beforehand as the USRPs. XCTU was used to scan the laptop ports for the UAV device connections and return information about the device that was necessary for its connection to GNU radio.

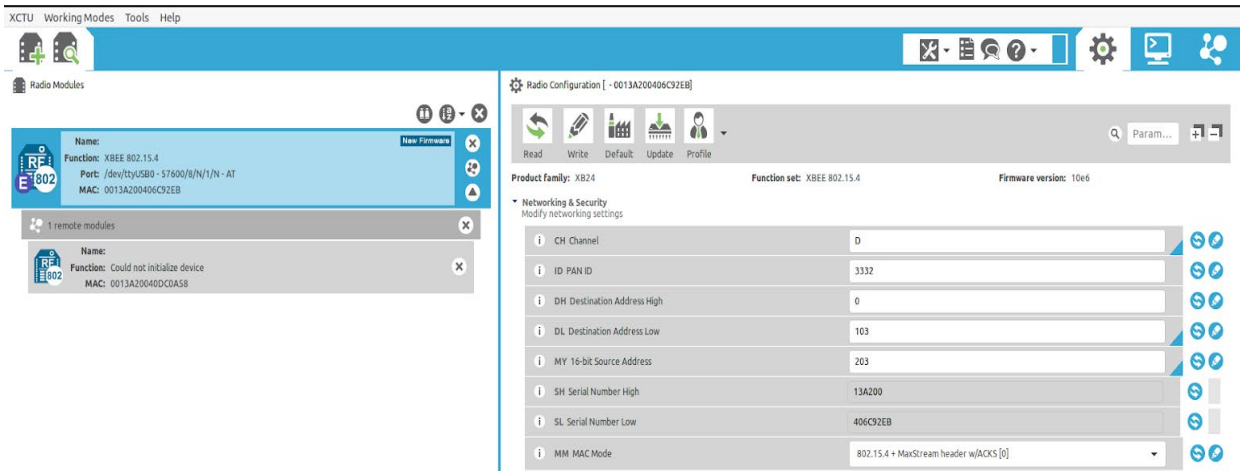

Figure 2.4 XCTU Output

Unlike the USRP devices, Zigbee devices do not connect with IP Addresses but with MAC addresses. GNU Radio, however, does not support this type of connection. Although Zigbee devices do have serial numbers, and GNU Radio supports a serial connection for USRP devices, the Zigbee UAV would not connect to the laptop in order to form a connection in this manner. We decided to use a packet sniffing procedure to collect data from the Zigbee UAV device.

The parts of the Zigbee UAV device connect and begin communicating with each other automatically once plugged in, so we prompted the USRP device to run the flowgraph (Figure 2.6), pull in the packet information sent by the Zigbee UAV, and display it in Wireshark.

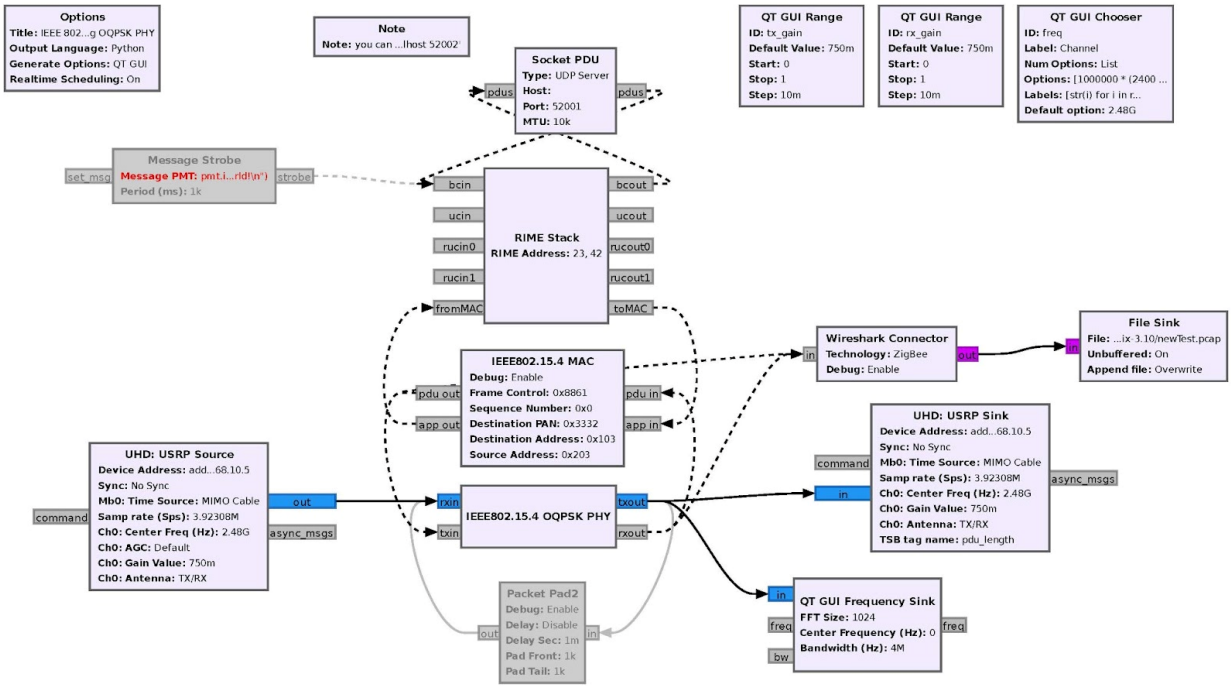

Figure 2.6 Packet Sniffing Flowgraph

This flowgraph disabled the transmitting block (Message Strobe) and the Packet Pad2 block from the Wime version to create a packet-sniffing receiver. The IEEE802.15.4 MAC block takes information about the UAV's source address, destination address, and destination PAN as inputs, and the source and sink blocks take the address of the USRP device as input, per usual. When you run the flowgraph, information is sent to Wireshark every time the USRP detects a packet with the appropriate source and destination information.

3. RESULTS

Collecting data [7] from a Zigbee UAV device is possible using packet sniffing. Figure 3.1 shows how the Wireshark software presents this data.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0x0103	0x0203	IEEE 802.15.4	53	Data, Dst: 0x0203, Src: 0x0103
2	0.000503			IEEE 802.15.4	5	Ack
3	0.096725	0x0103	0x0203	IEEE 802.15.4	48	Data, Dst: 0x0203, Src: 0x0103
4	0.097264			IEEE 802.15.4	5	Ack
5	0.135389	0x0103	0x0203	IEEE 802.15.4	106	Data, Dst: 0x0203, Src: 0x0103
6	0.135911			IEEE 802.15.4	5	Ack
7	0.258639	0x0103	0x0203	IEEE 802.15.4	77	Data, Dst: 0x0203, Src: 0x0103
8	0.259149			IEEE 802.15.4	5	Ack
9	0.276947	0x0103	0x0203	IEEE 802.15.4	106	Data, Dst: 0x0203, Src: 0x0103
10	0.277491			IEEE 802.15.4	5	Ack
11	0.319436	0x0103	0x0203	IEEE 802.15.4	34	Data, Dst: 0x0203, Src: 0x0103
12	0.319950			IEEE 802.15.4	5	Ack
13	0.375874	0x0103	0x0203	IEEE 802.15.4	53	Data, Dst: 0x0203, Src: 0x0103
14	0.376423			IEEE 802.15.4	5	Ack
15	0.500405	0x0103	0x0203	IEEE 802.15.4	53	Data, Dst: 0x0203, Src: 0x0103
16	0.500900			IEEE 802.15.4	5	Ack
17	0.625839	0x0103	0x0203	IEEE 802.15.4	53	Data, Dst: 0x0203, Src: 0x0103
18	0.626299			IEEE 802.15.4	5	Ack

Figure 3.1.1 Wireshark Data Scroll

```
▾ Frame 3: 48 bytes on wire (384 bits), 48 bytes captured (384 bits)
    Encapsulation type: IEEE 802.15.4 Wireless PAN (104)
    Arrival Time: Mar  3, 2023 16:18:56.148188000 CST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1677881936.148188000 seconds
    [Time delta from previous captured frame: 0.096222000 seconds]
    [Time delta from previous displayed frame: 0.096222000 seconds]
    [Time since reference or first frame: 0.096725000 seconds]
    Frame Number: 3
    Frame Length: 48 bytes (384 bits)
    Capture Length: 48 bytes (384 bits)
    [Frame is marked: False]
    [Frame is ignored: False]
    [Protocols in frame: wpan:data]
▾ IEEE 802.15.4 Data, Dst: 0x0203, Src: 0x0103
  ▸ Frame Control Field: 0x8861, Frame Type: Data, Acknowledge Request, PAN ID Compression, Destination Addressing Mode: Short/16-bit, Frame Version: IEEE Std 802.15.4-2003, Source Addressing Mode: Shor…
    Sequence Number: 133
    Destination PAN: 0x3332
    Destination: 0x0203
    Source: 0x0103
    FCS: 0xb012 (Correct)
    [Ack In: 4]
▾ Data (37 bytes)
    Data: b600fd170000020301fd000004454b463220494d55312066…
    [Length: 37]
```

Figure 3.1.2 Wireshark Data Information

```
0000  61 88 85 32 33 03 02 03  01 b6 00 fd 17 00 00 02   a··23··· ········
0010  03 01 fd 00 00 04 45 4b  46 32 20 49 4d 55 31 20   ·····EK F2 IMU1
0020  66 6f 72 63 65 64 20 72  65 73 65 74 c2 dc 12 b0   forced r eset····
```

Figure 3.1.3 Wireshark Received Message

In using the USRP to capture packets and send them to Wireshark, we are not only able to see messages that the parts of the device are sending to each other but also acknowledgments

of those messages. This data can also be verified by examining the UAV's QGroundControl software, as shown in Figure 3.2.

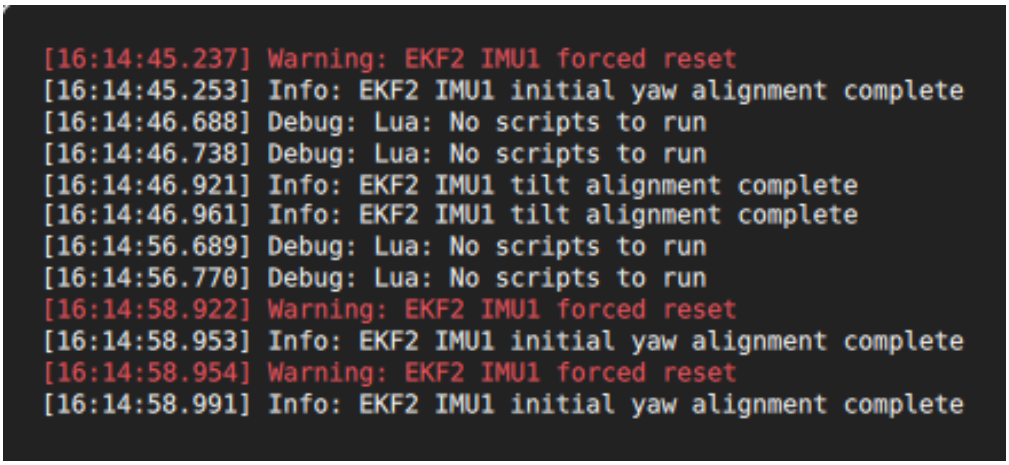

Figure 3.2 QGroundControl Messages

The calibration messages in Wireshark are the same as those in QGroundControl, verifying that the USRP is pulling the data as expected.

Figure 3.3 shows one area that may require further examination. These unknown messages and malformed packets show that bleed is occurring in the channel, which could lead to inconsistencies if additional research uses this data.

No.	Time	Source	Destination	Protocol	Length	Info
166	7.000085	0x0103	0x0203	IEEE 802.15.4	53	Data, Dst: 0x0203, Src: 0x0103
167	7.000597			IEEE 802.15.4	5	Ack
168	7.125154	0x0103	0x0203	LwMesh	53	Lightweight Mesh, Nwk_Dst: 0x00…
169	7.125673			IEEE 802.15.4	5	Ack
170	7.255467	0x0103	0x0203	LwMesh	77	Encrypted data (55 byte(s)) NO …
171	7.255975			IEEE 802.15.4	5	Ack
172	7.273613	0x0103	0x0203	LwMesh	106	Encrypted data (84 byte(s)) NO …
173	7.274156			IEEE 802.15.4	5	Ack
174	7.318969	0x0000	0x09fd	ZigBee	34	Unknown Command
175	7.319341			IEEE 802.15.4	5	Ack
176	7.375252	0x0000	0x1cfd	ZigBee	53	Route Request, Dst: 0xa300, Src…
177	7.375758			IEEE 802.15.4	5	Ack
178	7.378684	0x0000	0x1cfd	ZigBee	53	Route Request, Dst: 0xa300, Src…
179	7.379210			IEEE 802.15.4	5	Ack
180	7.499797	0x0103	0x0203	LwMesh	53	Encrypted data (31 byte(s)) NO …
181	7.500261			IEEE 802.15.4	5	Ack
182	7.536408	0x0103	0x0203	LwMesh	38	Encrypted data (16 byte(s)) NO …
183	7.536922			IEEE 802.15.4	5	Ack
184	7.625002	0x0000	0x1cfd	ZigBee	53	Unknown Command
185	7.625512			IEEE 802.15.4	5	Ack
186	7.749680	0x0000	0x1cfd	ZigBee	53	Many-to-One Route Request, Dst:…
187	7.750183			IEEE 802.15.4	5	Ack
188	7.757119	0x0103	0x0203	LwMesh	53	Encrypted data (29 byte(s)) NO …
189	7.757629			IEEE 802.15.4	5	Ack
190	7.823614	0x0203	0x0103	IEEE 802.15.4	34	Data, Dst: 0x0103, Src: 0x0203
191	7.824082			IEEE 802.15.4	5	Ack

Figure 3.3 Wireshark Malformed Data

It is left for future research, however, that converting the collected data from Wireshark's hex output to binary makes it viable for use in radio fingerprinting equations.

4. CONCLUSION

Radio fingerprinting provides researchers with a more accurate and robust method for identifying devices based solely on differences in their radio signals. Because these differences are difficult to mimic or spoof on a different device, those using radio fingerprinting can be certain that they are identifying the correct device. Most research on this topic has been conducted on IoT devices using standard WiFi protocols. As the research expands, smart devices running Zigbee protocols and UAVs are being added to the equation.

Zigbee devices not using IP addresses poses a challenge when collecting data, but using a USRP device to packet sniff via a GNU Radio flowgraph makes it possible. We can collect this data in Wireshark and then translate it into binary for use in deep learning and machine-learning algorithms currently being used for IoT devices.

Knowing that it is possible to collect this data allows for numerous follow-up questions for future researchers to examine. The first and likely most important of these questions is whether the data is compatible with the current radio fingerprinting formulas. If it is incompatible, what needs to change to make it compatible? Based on the data we collected in Wireshark, there is also the question of determining the validity of the collected data. Even when giving GNU Radio a specific source, destination, and channel to sniff from, it still pulled extraneous and malformed data. Specifically for UAV devices in relation to radio fingerprinting, there are questions of range issues or the ability to collect a sufficient amount of data quickly enough to identify the device if it is moving at a great speed. UAVs also tend to travel in swarms, leading to a question of how to distinguish one from another in that setting.

Research into radio fingerprinting is still in such early stages that the answers may take some time, especially with the increased complexity of adding UAVs. Being able to collect data, however, is a significant first step into being able to test the extent that this research can strive for in years to come.

REFERENCES

[1] H. Li, K. Gupta, C. Wang, N. Ghose, and B. Wang, "RadioNet: Robust Deep-Learning Based Radio Fingerprinting," 2022 IEEE Conference on Communications and Network Security (CNS), Austin, TX, USA, 2022, pp. 190-198, doi: 10.1109/CNS56114.2022.9947255.

[2] M. Cekic, S. Gopalakrishnan and U. Madhow, "Wireless Fingerprinting via Deep Learning: The Impact of Confounding Factors," 2021 55th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2021, pp. 677-684, doi: 10.1109/IEEECONF53345.2021.9723393.

[3] A. Al-Shawabka et al., "Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Toronto, ON, Canada, 2020, pp. 646-655, doi: 10.1109/INFOCOM41043.2020.9155259.

[4] F. Restuccia, et al., "DeepRadioID: Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms," Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing, Catania, Italy, 2019, pp. 51-60, doi: 10.1145/3323679.3326503.

[5] "ZigBee TRX | Wime Project." [Online]. Available: https://www.wime-project.net/tutorials/zigbee/.

[6] "Installation | Wime Project." [Online]. Available: https://www.wime-project.net/installation/.

[7] "amonson316/Radio-Fingerprinting-of-UAVs-An-Exploration-of-Zigbee-Data-Collection." [Online]. Available: https://github.com/amonson316/Radio-Fingerprinting-of-UAVs-An-Exploration-of-Zigbee-Data-Collection.

[8] "IEEE SA – IEEE 802.11-2020." [Online]. Available: https://standards.ieee.org/ieee/802.11/7028/.

[9] "IEEE SA – IEEE 802.15.4-2020." [Online]. Available: https://standards.ieee.org/ieee/802.11/7028/.