G400

for

Investigating Radio Frequency vulnerabilities in the Internet of Things

Prepared by
Student ID : 2097902
Vasilis Ieropoulos
School of Computer Science
IeropoulosV@Cardiff.ac.uk

Supervised By:
Eirini Anthi
Lecturer, Faculty of Science
School of Computer Science
AnthiES@cardiff.ac.uk

I hereby declare that this dissertation is all my own work, except as indicated in the text

October 6, 2021

Acknowledgements

I would like to thank you supervisor Eirini Anthi for allowing me to take on this project. I would also like to thank my Ham Radio colleges who have helped in exploring radio frequency and GNU radio further. Furthermore I would also like to thank Eleni for her help in proof reading my dissertation and any improvements I could make. Concluding I would also like to thank my parents for putting up with all my antennas on the roof and my experiments. We only do ham radio on days ending with "Y".

Abstract

The large adoption of IoT devices for both commercial and private use, the need to secure these devices has escalated drastically. Many of these types of sensors can give out confidential information which can compromise the security of the user as well as control other devices by exploiting gaps in both virtual and physical layers of their security, which could potentially physically affect the health and well being of its users. For example, a malicious threat actor might use a low level attack on the communication layer of a smart thermometer, manipulating packets as they are being received and transmitted by the device. The malicious threat actor could turn on the heating in an already hot building causing people to have an increased blood pressure which could lead to potential hospitalisation. Many manufacturers apply mostly software based resilience to attacks, for example mostly focusing on the networking layer of the device. This opens up the target device to a plethora of other vulnerabilities such as low level radio frequency attacks which can be almost undetectable without expensive setups and equipment. It should also be noted that most research being done on IoT devices is mostly focused on attacking the networking and protocol of the device usually emitting out the risks of radio frequency attacks. This paper explores different attacks on mainstream devices using software defined radio, to perform vulnerability assessment of the low level radio frequency layer with proposed solutions, which might help in securing the devices. A mixture of attacks where performed using software defined radios some of which produced different side effects which rendered the devices useless unless they where hard reset. For example when the weather station was tested for any radio frequency vulnerabilities, while performing a packet manipulation attack, even though it was not successful, the radio managed to de-synchronise the sensor with the receiver resulting in multiple readings being lost. Attacks performed on the devices range from simple jamming attacks, replay attacks, packet manipulation, signal and protocol reverse engineering.

Contents

List of Figures **iv**

List of Tables **1**

1 Introduction **2**

1.1 Internet of Things 4

1.1.1 Home-brew Internet oF Things 4

2 Related Work **5**

2.1 Securing ZigBee Communications Against Constant Jamming Attack Using Neural Network . . 5

2.2 Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning 5

2.3 A Lightweight Replay Attack Detection Framework for Battery Depended IoT Devices Designed for Healthcare 6

2.4 A Technical Review of Wireless security for the Internet of things: Software Defined Radio perspective 6

2.5 IoT-Based Smart Shopping Cart Using Radio Frequency Identification 7

2.6 Machine Learning-based RF Jamming Detection in Wireless Networks 8

2.7 Vulnerabilities in IoT Devices with Software-Defined Radio 8

3 Background Information **9**

3.1 Technical Background 9

3.1.1 Harmonics 10

3.1.2 Attenuation 11

3.1.3 Filtering 11

3.2 Examples of Attacks 12

3.2.1 Replay Attacks 12

3.2.2 Denial of Service Attacks 12

3.2.3 Sniffing 13

3.2.4 Man in the Middle 14

3.2.5 Radio Frequency Protection Techniques 15

3.2.6 Frequency Hopping Spread Spectrum (FHSS) . . . 15
3.2.7 Scrambling . . . 16
3.2.8 Split-Band inversion . . . 16
3.3 Equipment . . . 17
3.3.1 Hack RF . . . 17
3.3.2 RLT-SDR (NESDR) . . . 17
3.3.3 Temperature Compensated Crystal Oscillator (TCXO) . . . 18
3.3.4 CC2531 USB Evaluation Module Kit . . . 18
3.3.5 Gnu Radio . . . 19
3.4 Communication Protocols . . . 20
3.4.1 WiFi . . . 20
3.4.2 Zigbee . . . 20
3.4.3 Bluetooth . . . 20

4 Testing and Methodology **22**
4.1 Set up . . . 22
4.1.1 Pre-programming . . . 22
4.2 Methodology . . . 23
4.3 Belkin NestCam HD . . . 25
4.3.1 Testing . . . 25
4.3.2 Conclusions on Belkin NestCam HD . . . 27
4.4 NEST Smoke alarm . . . 28
4.4.1 Bluetooth Sniffing . . . 28
4.4.2 Jamming with IMCP . . . 29
4.4.3 Conclusion on NEST Smoke Alarm . . . 29
4.5 inFactory sensors . . . 30
4.5.1 Attack Scenario . . . 30
4.5.2 Data Acquisition . . . 30
4.5.3 Jamming . . . 31
4.5.4 Replay Attack . . . 31
4.5.5 Packet Manipulation . . . 32
4.5.6 Conclusion on Infactory Sensors . . . 33
4.6 SONOFF SNZB-02 . . . 34
4.6.1 Conclusion on Sonoff SNZB-02 . . . 36

5 Conclusion **37**
5.1 Future Work . . . 37
5.2 Project Appraisal . . . 38

5.3 Self Reflection 38

List of Figures

3.1 Representing dBi . . . 9
3.2 Coaxial and general RF cables . . . 11
3.3 Illustration of Replay Attack . . . 12
3.4 Mathematical Proof of RF waves using transverse propagation . . . 13
3.5 Illustration of Denial of Service Attack . . . 13
3.6 Illustration of sniffing . . . 14
3.7 Illustration of Man In the Middle . . . 14
3.8 Scrambled Signal . . . 16
3.9 Dual Inversion splitter . . . 16
3.10 HackRF . . . 17
3.11 NESDR . . . 18
3.12 TCXO . . . 18
3.13 CC2531 . . . 19
3.14 GNU Radio . . . 19
3.15 Bluetooth Packet . . . 21

4.1 CC 2531 Flashing . . . 23
4.2 Belkin NestCam HD . . . 25
4.3 Sparrow WiFi De-authentication . . . 25
4.4 HackRF 2.437 GHz . . . 26
4.5 HackRF 1.2185 GHz . . . 26
4.6 WiFi Sniffing . . . 27
4.7 NEST Smoke Alarm . . . 28
4.8 Using Bluetooth Receiver . . . 29
4.9 Using HackRF . . . 29
4.10 inFactory Sensor . . . 30
4.11 Infactory sensor data decoded . . . 31
4.12 Infactory Packet Analysis . . . 32
4.13 Weather station interval . . . 32

4.14 Weather station Packet Chunks 33
4.15 SONOFF SNZB-02 34
4.16 Sonoff SNZB-02 PCB antenna 35
4.17 Replay Attack 35
4.18 MQTT 36
4.19 SDR Angel 36

List of Tables

3.1 Harmonics 10

Chapter 1

Introduction

With the advancement in internet technology in the last 30 years, a new market of devices has emerged that combine the use of the internet and sensors to exchange information between them in a meaningful way. These devices are called Internet of Things. The mass marketing and adoption of these devices has made them easily available to many people who implement them in their home without proper infrastructure therefore many of them are vulnerable to low level attacks. There have been studies and testing done on IoT devices in regards to network penetration testing but there is a lack of research in the field of Radio Frequency. The increase in IoT device attacks has surged drastically in the past few years. Specifically in 2019 there was a record of 2.9 billion attacks collected by honey pots on IoT devices. That was a 300% increase in attacks in comparison to the previous year where there where only 819 million attacks[11].

One of the major attacks which have compromised many Zigbee based IoT devices is the Zigbee Worm attack which exploits hard-coded symmetric encryption keys in the Zigbee protocol that allow the worm to take control of the devices[43]. This then allows it to spread to other nearby Zigbee devices allowing for full control of the network. This can be done as far as 400 meters away which makes it even more difficult to track and trace. The worm can then spread from a single smart bulb to those nearby thanks to the use of these skeleton keys [43].

With 70% of IoT devices being vulnerable to attacks[31], investigating more and more devices that over saturate the market becomes increasingly important. We need to also analyse the threat of low level attacks that arise from the use of software defined radio or even analogue radio [31]. By analysing the threats of low level attacks will help in securing them especially when placed in sensitive areas such as homes and offices which have a high probability of containing sensitive data. Examples of sensitive data are, classified clientele information such as names, addresses and private and confidential projects which could be leaked through the means of video or audio. If such data where to be released would result in loss of reputation, loss of revenue and could potentially be used by competing companies to take advantage of the situation. The current illusion set by some manufacturers that their devices are "safe" allows malicious actors to pray upon unsuspecting users that set up their devices in an insecure manner.

As we will explore further down the project, many known brand devices are vulnerable to even the most low skilled of attacks which could potentially compromise small to large scale businesses that take advantage of these devices in their everyday use. Although there are security checks being implemented on the network side of the devices, there is a lot lacking on the radio frequency side. Hence the concept of radio frequency vulnerability assessment seems to be worthwhile investigating as there are not many off the shelf security implementations that can protect from these types of low level attacks.
Although there are methods to detect these types of attack, they require constant monitoring and resources, something many companies and especially individuals will not invest in[44]. This project aims to investigate these types of attacks and propose some solutions which could further help in the fight against malicious threat actors.

The following chapters are organised as follows :

- Related work: This section provides context on current implementations and related work to vulnerabilities in IoT devices as well as limitation of current technology. This also includes both commercially available hardware as well as home-brew equipment.
- Background Information: This section provides the background Information required for implementing and testing our devices with emphasis on the technologies used under the hood of the devices. We also explore the physical aspects of radio frequency and how that effects our testing.
- Testing and Methodology: This section covers the methodology used to test the devices along with the environment created to test them. Furthermore, it covers the results of the testing and some of the issues faced.
- Conclusion: This section provides an overall summary of the project as well as the author's thoughts about future work related to the project.

1.1 Internet of Things

IoT devices are usually small low powered devices that are connected to the internet to serve data to another system. They are usually made up of small sensors that feed data to interconnected hubs, switches and gateways which then provide a service to the end user[34]. Due to their large market and many manufacturers dipping their toes in the industry, many of these devices lack security, build quality and quality of service[34]. IoT devices are sensors of all types like microphones, cameras, thermometers, RFID transmitters etc. The data that is being collected by these devices can be sent to cloud services for analysis and execution of certain functionality for example turning on lights, ordering food or ink for your printer. With the advancement of the home-brew community and the availability of small and cheap circuit boards like Arduinos, many enthusiasts have built their own versions of IoT thus keeping all the processing and data private for their own use.
Typically IoT devices are connected wirelessly to their hub devices using a variety of technologies and protocols as we will explore in section 3.4. Usually IoT devices collect their data using some sort of sensor, this could be any thing from an temperature to a light sensor. Either way most IoT devices don't store their data on board but send it out directly to their respective connected device hubs.If they do have data storage it is usually very small and only store enough data so they can connect to their network. Typically these devices are designed to send out very small volumes of data at a time[34].The only processing a IoT device usually does is gather data and convert it to small meaningful chunks that can be sent out to the hub which can handle the processing.The devices are usually equipped with small light weight antennas usually etched on the PCB of the device to minimise footprint[36].

1.1.1 Home-brew Internet oF Things

With the decreasing cost of building and implementing your own Internet of Things many hobbyist have started to create their own devices with the use of low cost hardware. There are many devices to choose from such as the Arduino, Raspberry Pi, NodeMCU, ESP 8266 and many more. IoT development has endless possibilities in both hardware and software. We could classify these devices into two categories :

- Wearable devices such as : watches , rings, bracelets etc.
- Embedded devices such as : Arduino , ESP8266 , Raspberry PI etc.

The most complex part of any home-brew IoT device is developing the software needed for it to communicate with a central hub or an entity for more complex functions. This becomes more complex as we start to mix and match between devices since some devices are not 1:1 compatible and require some form of tinkering before they are connected together[49].
We could classify the services provided by these boards and systems into further categories :

- Development
- Attainment of Data
- Device Controller

Development is the analytical data gathered by the devices to produce an event-driven command such as a visualisation of the data or a help in the production of more applications[49].
Attainment of data contains the management and conversion of data at different stages of the IoT device for analysis by a more powerful device[49].
The Device Controller is composed of monitoring devices for security and firmware updates as well as the overall well being of the device[49].

Chapter 2

Related Work

In this section we shall explore related work which has been done in the past on radio frequency vulnerabilities in Iot devices.We shall also critically analyse the results and conclusions of the related works as well as the methodologies and testing environments used.

2.1 Securing ZigBee Communications Against Constant Jamming Attack Using Neural Network

The paper sets an interesting premise using low budget Zigbee receivers to conduct their research while using a HackRF to conduct their tests. The use of basic radio hardware is a good start but it is not enough to fully test the security of the protocol. The premise that these are a threat is true but it should be noted that the attacks have taken place at an extremely close proximity to the receivers and transmitters. Taking that into account, it would require the attacker to be very close to the device which would make it extremely obvious since, the only way this would be feasible would be to use a high-gain antenna paired with a substantial amplification. Furthermore, that sets the bar of these attacks to only be semi-successful at close range. In addition, the receivers used are bare bones with insufficient filtering or other advanced features and are not even enclosed in a case to at least reduce the chances of spurious emissions. The study assumes that this will work on more commercial device but it is not a guarantee that it will work on all devices as many off the shelf device are designed with extra filtering and components to meet FCC regulations. Taking this into account it should also be noted that the Zigbee protocol is designed to also change frequency (frequency hopping) uses channel 11(2.405 GHz) just for the initial connection phase. The paper explores the use of Neural Networks to create a jamming mitigation module which leverages its resources to decode Zigbee packets through malicious signal. This is a very interesting idea and the paper does a good job of explaining how it works, but it is important to note that the testing was done using laptops and not IoT devices. There needs to be a much more minimalistic approach to apply this method since most IoT devices are designed with very low end hardware. A solution to this would either be implement the neural network on a device like a raspberry pi which uses low power but is powerful enough to run a neural network or use a more powerful device like the Intel NCS[33]. We shall explore the use of low budget hardware and how its possible to conduct these attacks especially when using homebrew IoT devices. In essense we shall see the difficulties of attacking more comercially available devices using the zigbee network in section 4.6.

2.2 Jamming attack detection in a pair of RF communicating vehicles using unsupervised machine learning

The paper explores the possibility and the risk of attack as well as detecting attacks on Vehicular ad-hoc networks (VANET). The use of these RF networks is to create a network of vehicles which are able to com-

municate with each other using a common frequency which allows them autonomous vehicles to exchange information. In doing so they can predict driving patterns , traffic and hazards that have been detected by other vehicles along the road. The paper also describes the use of unsupervised machine learning to detect the presence of jamming attacks. [50] The main compromise which was made during the testing of the jamming attacks was that the jammer was using equal or less power than the cars transmitter, this was a maximum of 100mW. Reducing the power reduced the chance of the jammer being able to perform its intended purpose. Hence the scenario tested was an "idealistic" scenario where the jammer was able to function "most" of the time. In a real world scenario if this was to be performed at a large scale, the jammer could possibly emit a large amount of noise using more than 1W which would cause the cars to lose their connection. There is no mention in the paper of the use of any obstacles that could possibly reduce the transmission of car such as trees, buildings, etc [50]. This is a very crucial part in conducting a real world scenario of the attack, as in most cases cars wont be driving on open roads but in built up cities. Since propagation of the signal will greatly depend on line of sight as the frequency used by the cars are in the Microwave band (300mhz-30ghz) it would be more realistic to put some form of obstacles around the transmitters. Furthermore, VANET uses cell towers to send out information in an adhoc mode which might be in an issue in more rural areas where cell reception is weak[50]. Although we shall not be exploring the use of RF in vehicles, the concept of long range communication between two devices becomes prevalent throughout the project and displays the real challenge of interference that can be cause by different factors, such as other devices or the casing in which the devices are housed in. We shall a similar scenario in section 4.6 were a simple action can cause major disruption.

2.3 A Lightweight Replay Attack Detection Framework for Battery Depended IoT Devices Designed for Healthcare

The paper proposed a solution to replay attacks toward low end IoT devices. The proposed solution uses Universally unique identifier (UUID) to identify the device and uses a hash function to create a unique signature for each device. The signature is then hashed with a secret key. Expanding, with the use of timestamps it keeps a short log of events triggered while also adding a unique identifier to each log entry. The unique identifier is then used to identify the operation before and after execution. Furthermore, it theoretically uses a the battery depletion monitor to monitor battery power and append it to every new log that is created. This creates a theoretically low powered solution to monitoring any attacks. If a command is replayed back to the device that is exactly the same as the previous one it would be rejected by the micro controller as it it will be detected as a replay attack. The issue in using such a solution, it assumes that all IoT devices use similar hardware specifications. This is an issue since not all micro controllers work in the same way and this could create compatibility issues[14]. One major flaw of the solution is that it continuously creates a log of executing commands, which is not a good solution for a battery dependent device as it would constantly consume power and eventually run out of battery[14]. In reality, even if it detects a replay attack it would end up becoming a Denial oF Service attack due to this limiting factor. This is due to the fact that, the more processing power it needs to store the log file, as well as the storage needed, the more power it will consume thus draining more battery and effectively making turn itself off in much less time. This is a huge issue as it also posses a risk to health for a patient using such a device for monitoring their vitals. We shall explore the limitations of having a battery operated device in section 4.6 as well as the ease of jamming and disrupt services with very little effort in section 4.11.

2.4 A Technical Review of Wireless security for the Internet of things: Software Defined Radio perspective

The paper presents examples of attacks that can be done using software defined radio on IoT Devices. It explores some attacks that require planning for example replaying ACARS packets but also simplistic attacks like sniffing telemetry data using cheap SDR hardware. The paper does have a full list of attacks with a few examples as well as the devices capable of the attacks.The issue lies with the fact that many of the more sophisticated attacks require a lot of power,a lot of time to complete but also a lot of money to purchase the hardware. For example the attack on GSM, GNSS, and Bluetooth requires a USRP which is not only expensive

but also large in size compared to a HackRF one. A best case scenario would be to use it in a specific built vehicle like the Israeli Spy van in seized Cyprus[17]. A van like that is was equipped with over £9 million worth of equipment and was used to carry out a lot of attacks such as the alleged surveillance of high value assets from the Middle East[10]. What we can deduce from this is that a large amount of these attacks are usually state sponsored and usually only target high value targets [6]. That said we should not rule out the threat of these attacks but we should be aware of the fact that they are state sponsored and that they are not going to be easy to detect or targeted to the average person[17]. Expanding, the paper has made some claims that some of the attacks like the ones on NFC can be prevented by using a USRP. Although, this might be possible since it would make it difficult to anyone with a device such as a HackRF to correctly sniff the signal. It means that the cost of securing from an attack like this is quite high with the chances of it happening being quite low. Even if the attack happens "discretely" using the HackRF in a "portapack"[53] it should be noted that the attacker would need to get very close to the victim(even physically touching them) for this to happen and at that point the NFC signal would over power the signal generated by the USRP because of attenuation. Concluding as stated in the paper,Cyber-security challenges which arise from the use of software defined radio toward IoT devices are extremely complex and therefore are not something that we can expect to be able to solve in a short amount of time. Software Defined Radios are a great tool that should be used to secure IoT devices and should be combined with other security measures to make sure that the devices are secure from different kinds of attacks. In this project we shall explore the ease of performing simple attacks using "budget" hardware as well as the limitation they present such as lack of bandwidth to perform more sophisticated attacks.

2.5 IoT-Based Smart Shopping Cart Using Radio Frequency Identification

The aim of the paper is to create a low profile smart shopping cart using Radio Frequency identification to allow for seamless shopping in supermarkets without the need for long queues. The solution is made out of an Arunino Uno connected to a RFID reader which then relays the data via Bluetooth to a Smartphone and finally charges the customer once they leave the store. The main flaw of the the whole system is that it only focuses on the security of the customers information and not the products that the customer is buying. There is no mention of implementing any checks that all the items have been scanned or any checks that the device is actually connected properly to the system. With that being said, as demonstrated multiple times RFID tags can be easily blocked from being read by using very simple methods like a aluminium layer bag [51]. Furthermore, if a bad actor would like to pass through the till without raising an alarm they could use a simple jamming attack to block the tags from going off and being read by the security barriers. As we shall explore in section 4 we observe that in most cases this is simple and effective to perform. Many RF security barriers operate on 8.2MHz and rely on the possibility that a device passed through the gate will emit a 8.2MHz pulse that will trigger the alarm[4]. That being said the lack of security measures from the proposed solution lack any form of presence checking to actually authenticate if the product is actually there. Expanding a bad actor could easily pass through the gate without scanning the product, removing the tag or having a small rare earth magnet close to the tag to disable its transmission. A solution to try and mitigate this would be have the trolleys weighed when they pass through the till. Since the trolleys have roughly the same weight when unloaded, if the products that where scanned are in the trolley the trolley should of be "X" weight. Although this will not stop someone from placing small items in their backpack or pockets in a foil bag, it would severely limit the possibility of large quantities of items being stolen. This is relevant to our project as we have analysed the characteristics of low transmission powered devices and their effective range as well as the ease of preventing the transmission of these devices by using more powerful signals. We shall explore this further in section 4.

2.6 Machine Learning-based RF Jamming Detection in Wireless Networks

The paper aims to create a machine learning model for detecting RF jamming attacks in Wireless Networks using simulated jamming attack data. Using these simulations they were able to create an somewhat effective model for detecting these attacks.As stated in the paper, when the attacks are done at random the algorithm cannot perform well enough due to its poor resolution. Using simulated jamming attacks remove multiple crucial variables that can occur when an attack like this happens. For example, the simulated data will not take into account any harmonic or spurious emission that could take place when an attack like this happen which could lead to further devices being affect unwillingly. We shall explore the technicalities of harmonics and spurious emissions in section 3.1. Expanding, since the testing was done as a simulation it does not take into account the signal strength , filtering or distance between the jamming device and the victim. These results have only produced a best case scenario assuming the algorithm can receive the signal with 100% clarity which in many cases is not possible due to the distance between devices and the possibility of packet loss. Although some of the models look promising like the random neighbour algorithm, it is still weak against randomised and reactive jamming attacks. With the every expanding availability of low cost software defined radio (as we shall explore in section 3.3) the complexity and diversity of these attacks would results in the algorithms needing to be retrained or a much better alternative method should be developed. One alternative would be to implement reinforcement learning using Artificial Intelligence would could possibly combat the changes and development in jamming techniques. This of course would require more resources and time to train in the beginning but could possibly end up with much better results do to the nature of the algorithm. Our project demonstrated the problems of packet loss even at close ranges which would become a problem when using an algorithm to analyse the spectrum. Furthermore , our project explore the limits of using low cost devices to perform this kind of analysis as well as the limitations the carry. This is observed in section 4.

2.7 Vulnerabilities in IoT Devices with Software-Defined Radio

The paper sets the premise of exploring vulnerabilities in IoT devices using Software Defined radio. The paper uses a very simplistic Arduino board with an 433 MHz Transmitter and an identical one with a receiver instead. The code used in the paper is very simplistic and only sends out a few bits of data with no integrity checks between transmissions. Although the attacks were performed successfully, it was obvious from the beginning that this would be possible as there are no security countermeasures between the receiver and transmitter. Hence almost any attack would be possible. The signal was easily jammed using the HackRF but could also be jammed using a simple,cheap and affordable radio which could be bought for around £30 like the Baofeng UV5R.Furthermore, the method used to figure out the peak frequency of the transmission was over complicated as the same thing could be achieved by firing up either Universal Radio Hacker or GQRX and monitor 433 MHz with the HackRF. The message was decoded manually using Audacity by demodulated the signal received in raw form and imported into the program while observing the peaks for 1s and 0s. Too much effort was made by using very "primitive" methods instead of using more ready made programs which are tested and proven to work like the ones mentioned previously. Although GNU radio is a complex program the sketches used are very simplistic but do work. In essence, the overall approach of the project is extremely simplistic and lacks investigative research. The paper does mention mitigation strategies against RF attacks but it should noted that in most cases, more commercially available devices like the ones we will explore in section 4 already have some of these security measures implemented.Furthermore, frequency hopping is used in protocols such as Bluetooth and Zigbee which are already implemented with most IoT devices. The paper mentions the use of directional antennas for a more narrower beam to "prevent" sniffing and jamming. This could not be further from the truth. The use of narrower beams does not "prevent" sniffing or jamming, it will only make it harder to perform as the signal with travel in a much narrower path[45]. As demonstrated in our project, are many other factor that need to be taken into account when analysing and testing a device even if it is "simplistic" in nature. We shall explore this in further detail in section 4.5.5.

Chapter 3

Background Information

3.1 Technical Background

In this section we will explore some of the technical background regarding radio frequency that is relevant to this paper. We explore the low level and high level aspects of radio frequency as well as notation and abbreviations used throughout this paper. Most projects fail to explore if there is any impact on the devices if they are attacked on a harmonic frequency or send out any spurious emissions. The concepts shall be explored in sections 3.1.1 and 3.1.2 which will give us an in detail look on how they work and shall be applied in section 4.

Units of Measurement

Some of the most used units of measurement when dealing with Radio frequency is Decibel(dB),Decibel relative to Isotrope(dBi),Decibel per mili-Watt(dBm). Although they might look similar they are a measure of different things and in some cases are used to confuse end-users to give the illusion of much greater gain than actually indicated like the use of Megabits and Megabytes per second used by Internet Service Providers[39].

We will go into some detail about how each one is measured and what they mean as to fully grasp the concept of antenna gain.

dB

dB is the relative measure between two outputs the input and the output. It is not an absolute value as the measure taken to calculate the dB is being compared to a given scale. Loss of gain with the us dB is indicated by negative values.[20]

dBi

Although the main unit of measurement is Decibel the way we measure dBi is completely different to the rest of the methods. It is measured using an isotropic antenna which receives and transmits in a uniformly in all directions.An isotropic antenna has 0dB gain making it a great reference point for taking measurements[20]. When an antenna is rated using dBi it references the rated power it can transmit from a given point. More specifically we can represent this mathematically as:

$$\int \mathbf{10} \log() \, 10(\frac{l_x}{l_z})$$

Figure 3.1: Representing dBi

Where:

- G=Gain
- 10 log10 = the standard logarithm measure of relative power
- Ix = the intensity of the antenna in a single direction at a certain distance
- Iz = the electromagnetic intensity of an isotropic antenna at the same distance

dBm

dBm is used to indicate the actual power output of an antenna or device. For example 0dBm is equal to approximately 1mW hence 20dBm is equal to 100mW. This will become more relevant later on in section 3.4.1. This unit of measurement is especially useful when dealing with low powered devices such as IoT devices which use only miliwatts of power[9].

In conclusion, it is important to distinguish between these units of measurement as they can sometimes be used interchangeably but have completely different meanings. This becomes a huge factor when designing or buying antennas as the units can become misleading[9].

3.1.1 Harmonics

Every electronic device which utilises radio frequencies for communication and operation such as WiFi,Zigbee,LoRa etc. are prone to harmonic frequency emission[45]. This is one of the fundamental properties of electromagnetic radiation. Electronic devices are under heavy regulation from bodies like the FCC which are in charge of reducing and controlling the behaviour of these emissions[47].

Table 3.1: Harmonics

Harmonic	**No. of waves**	**No. of Nodes**	**No. of anti-Nodes**	**Length vs Wavelength Relationship**
1	1/2	2	1	Wavelength = (2/1) * L
2	1	3	2	Wavelength = (2/2) * L
3	3/2	4	3	Wavelength = (2/3) * L

When a wave oscillates it will produce harmonic waves with a reduction in bandwidth every time which results in an oscillation on a different frequency. In our case this could allow for attacks to be carried out by using even less "sophisticated" equipment by transmitting on lower frequencies[45].

Harmonics can cause unintended interference to other devices operating on the frequency which the harmonic will oscillate. As we get closer to the third harmonic and the wavelength increases,the noise floor of the electrical signal will also increase which in turn can cause "jamming" of legitimate signals and stop them from reaching their intended receivers. Furthermore, they can cause the capacitance level of power factor capacitors to fl actuate and inductance of the power supply to become unstable causing the device or neighbouring devices to behave irregularly[2].

Harmonic Spurious emissions are also a huge issue in electronic devices as they produce unwanted oscillations close to the fundamental frequency, as-well as harmonic frequencies which can cause interference . For example a device transmitting at exactly 433 MHz with a bandwidth of 25 kHz will also bleed onto other frequencies well over its bandwidth without proper filtering[2]. This is especially prominent in dual side mode modulations such as Frequency Modulation (FM), Amplitude Modulation (AM), Dual Side Band (DSB) ,Phase Modulation (PM). Protocols like Wifi use a combination of Binary Phase Shift keying (BPSK) and Quandrature Phase Shift Keying(QPSK). The protocol switches between modulation types to optimise transmission rate and error correction[2]. Many of these spurious emissions are generated by Low Noise Amplifier embedded in the devices which are used the amplify the low powered signal transmitted from the device. Using this type of amplification causes not only the desired transmission signal to be amplified but also any harmonics or "noise" it generates[47].

3.1.2 Attenuation

Attenuation is the degradation of signal strength during transmission, this can cause interference to devices close to the transmitting device which might cause it to loss data. Attenuation is represented using decibels(dB). Decibels are the logarithms amplification by a factor of 10 by an input signal and divided by the output[18] .

We can represent the attenuation in a coaxial cable by :

$$\text{Attenuation (dB / 100 m)} = 10 \times \log()\, 10 \left(\frac{P_1(W)}{P_2(W)} \right)$$

Figure 3.2: Coaxial and general RF cables

Where :

- dB are the decibels being transmitted from the device
- P2 is the power in watts being transmitted by the device .
- P2 is the power at the end of the cable

Attenuation decreases the efficiency of a transmitted electrical signal due the absorption or scattering of the photons. One of the main benefits of using 2.4ghz for protocols like WiFi and Bluetooth is the lack of interference due to weather conditions[41].

3.1.3 Filtering

For any device to be FCC compliant it should have adequate filtering to reduce the harmonic emission from the device . The harmonic produced by a device should be significantly less powerful than the fundamental frequency of operation . To limit the spread of spurious emissions and harmonics adequate filtering should be used such as a low pass and high pass filter to only allow emission on a specified frequency range[28].

One of the advantages on installing filtering is the increase in transmission quality as the packet drop rate decreases[28].

3.2 Examples of Attacks

In this section we shall explore the various types of attacks that could potentially be performed by a malicious threat actor. When perfoming any attack it is important to first identify what protocol the device is using by firstly sniffing radio frequency spectrum of any packets that could possibly be decoded.Then using data from the manufacturers website as well as FCC listings of the device (assuming they are commercially available) we can then determine the frequency or frequencies of operation of the device hence conclude on which type of attack would be most suitable.

3.2.1 Replay Attacks

Replay attacks are "simplistic" in nature as their end goal is to replay a transmitted signal to the targeted device. This could cause a device to respond to a message or command that was issued before, for example turning on and opening the garage door, turning off the light etc. These have become very common as they are easy to execute on low-end devices such as sensors and IoT devices especially without any form of protection[16]. Some mitigation strategies are to use rolling codes which are a type of encryption that can be used to prevent replay attacks. A rolling code works by using a hash function to create a unique code for each message. This code is then used to encrypt the message and then the hash function is used to create a unique code for each message. This is done to prevent replay attacks[16].

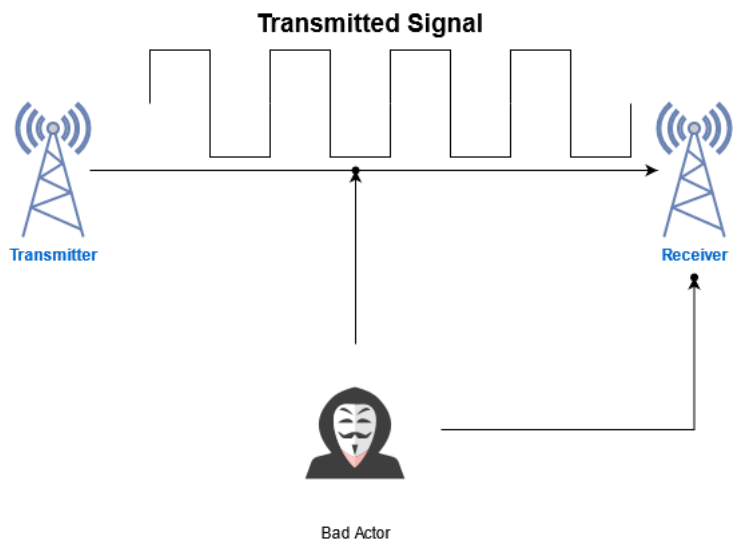

Figure 3.3: Illustration of Replay Attack

As demonstrated by the diagram, the attacker needs to be in between the transmitter and the receiver. Once the bad actor receives the transmission , they can then replay the same transmission back to the receiver[16]. Assuming no security measures have been put into place like rolling codes, there should be no problem for the receiver to decode and execute the signal.

3.2.2 Denial of Service Attacks

Denial of Service Attacks are attacks that cause a device to crash or become unresponsive. They are very common in the form of flooding IMCP packets to web server or devices to disconnect them from a network. It is one of the oldest and most common types of attacks. In the RF world it is known usually as jamming[42]. To produce a jamming signal, the jammer uses a signal generator to produce a continuous wave signal which is then sent to the target device. Usually a jammer will send out a stronger signal than the signal it is trying to jam, effectively making the target device nonoperational. In sound waves , we could cancel out the wave by transmitting an equal and opposite wave[42]. In doing this, we cancel out the wave and the signal is completely

cancelled hence the sum of the waves would be zero. This is not the same concept in the RF world as RF uses transverse waves which do not oscillate the same way as sound waves[19]. We can deduce this using the formula below figure 3.4 which shows that waves are perpendicular to each other [19].

$$\int \mathbf{A}dV \cdot \int \mathbf{B}dV \neq \int \mathbf{A} \cdot \mathbf{B}dV = 0$$

Figure 3.4: Mathematical Proof of RF waves using transverse propagation

RF waves are always perpendicular to the direction of oscillation, thus when we examine their behaviour away from the their source of generating we can conclude that they are always perpendicular. Therefore, it becomes very difficult to jam a signal in the same way as sound waves since there are many factors taking place which could cause the signal to pass through[19].

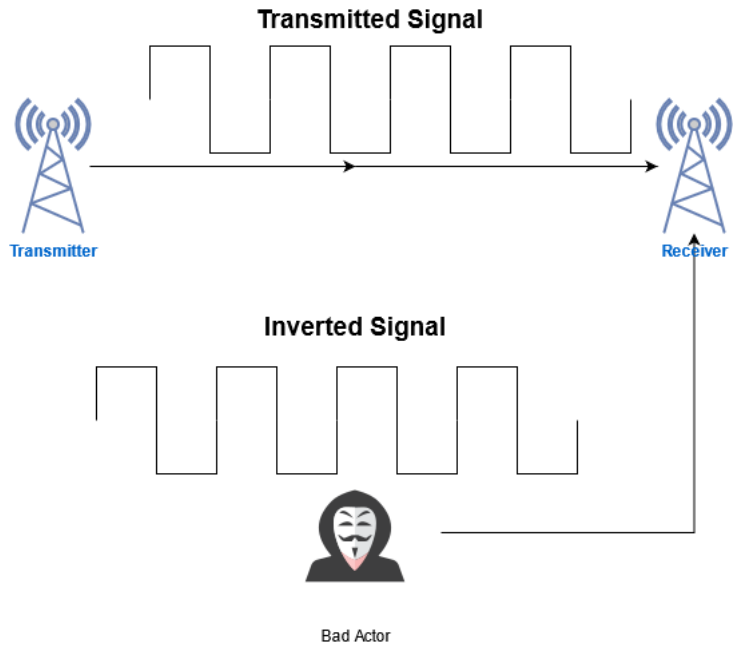

Figure 3.5: Illustration of Denial of Service Attack

Using the illustration above we can deduce that jamming a signal can be achieved by transmitting the same or more powerful signal to the receiver in order to make the less powerful signal inaudible to the receiver. A bad actor could achieve the same result with less power by getting closer to the receiver thus requiring less power to reach it and the phenomenon of attenuation will take place.It should be noted that this is one of the "easiest" attacks to perform but there have mitigation strategies put into place in more sophisticated devices, as we will explore in section 3.2.5.

3.2.3 Sniffing

Sniffing is one of the most common forms of RF attacks. Since there is no way of physically hiding an RF transmission, any transmission which does not use some form of encryption, hopping or scrambling, faces the threat of being monitored by an unauthorised third party. In many cases using a proprietary protocol can also make it difficult for inexperienced unauthorised third party to sniff and decode the signal. This is because the protocol is not well known thus there are no presents available to decode the signal. In this case we can use a software to analyse the signal and determine its modulation, sample rate,bandwidth etc. In our case we will be using a software called Universal radio hacker which is a free and open source software that can be used to analyse the signal. Expanding, once a malicious threat actor managed to sniff and decode a signal, they can then use this information to perform a variety of attacks for example packet manipulation,replay attacks etc.

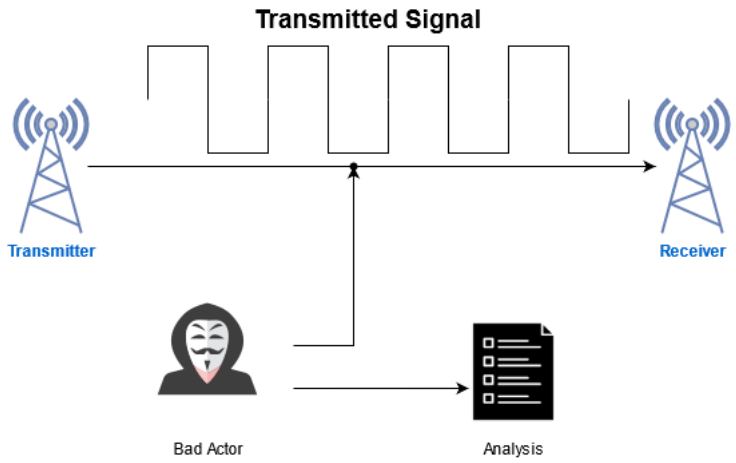

Figure 3.6: Illustration of sniffing

Using the illustration above, a malicious threat actor will monitor a signal sent out by a transmitter and try to analyse it using tools and if successful it could decode the signal. There are no well documented ways to stop sniffing from happening or them being detected but there are ways to prevent analysis and decoding using sophisticated techniques as we will explore in section 3.2.5.

3.2.4 Man in the Middle

A man in the middle attack is performed by a malicious third party by eavesdropping between two or more devices which are communicating with each other[37]. Usually the attacker will try and mask themselves as the receiving or transmitting device that the other device has established a connection to. This allows the malicious threat actor to sniff and analyse the victim's transmission without them knowing and also send back malformed packets which can be used to execute commands remotely[37]. These can be in the form of forged SSL certificates, authentication keys for establishing secure connections or rogue WiFi/zigbee/Bluetooth devices which try to take control of a victims device. One of the first instances of a man in the middle was the decoding of the German Military's radio communication done by the Royal British Intelligence during World War II. This was done using the Enigma machine which was used to decode the "encrypted" form of communication done by the German army[12].

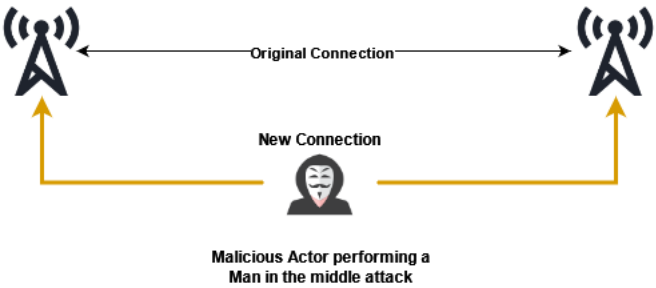

Figure 3.7: Illustration of Man In the Middle

Man in the Middle (MITM) attacks usually happen on the communication layer. One of the most common forms of MITM attacks are spoofing attacks where the attacker takes on the identity of an authentic device trying to lure in the victim and steal their data[37].

3.2.5 Radio Frequency Protection Techniques

In this section we shall explore radio frequency protection techniques which can be implemented to mitigate different types of radio frequency attacks. Many of these methods are important in establishing confidentiality, integrity and reliability between devices. It is important to understand these techniques, as we shall observe in section 4 they play a crucial role in trying to make it difficult to perform some attacks especially with our choice of hardware.

Encryption

Encryption is used in a wide range of applications even on radio frequency transmissions. One of the most used encryption methods is AES256 which is a symmetric encryption model[48].

AES (Advanced Encryption Standard) is a symmetric encryption algorithm that uses different key sizes of 128/192/256 in 128 blocks. Symmetric key encryption uses a private key to encrypt and decrypt data between two or more devices[38]. Since all devices need to have the same private key to decrypt the data, there is always a risk that the key could be filtrated to the malicious actors especially in large networks. Thus using asymmetric key encryption was created to solve this problem[48]. Instead of using the same key to encrypt and decrypt, we use a private key and a public key. The public key is shared with everyone, while the private key is used to maintain its function to encrypt confidential data and only allows the owner to access the data. This does come at a disadvantage as is comes with an performance overhead and requires more processing time and resources, which is something exceedingly sparse on low-end devices such as sensors and IoT devices. This is why even though AES is using a Symmetric Key Encryption standard, due to its low resource requirements it able to work on IoT devices.[48]

3.2.6 Frequency Hopping Spread Spectrum (FHSS)

One of the early adopters of frequency hopping spread spectrum are for military use. It works by constantly alternating between the carrier frequency and other pre-programmed frequencies. These changes are done by a code plugin which is known both the receiver and transmitter. FHSS is very useful in trying to avoid interference to other signals, and enabling Code-Division multiple access which allows communication of multiple signals through one entity and prevent eavesdropping[1].

By dividing the available band space into smaller bands or channels signals hop between them in a predetermined fashion. If there is interference at a specific frequency it will only affect the signal during a short interval.

There are three main advantages of using FHSS compared to using a fixed frequency:

1. Jamming is difficult to accomplish by a malicious third party if the pattern is unknown
2. The signals are difficult to intercept if the pattern is unknown
3. Since the signal keeps changing frequency it is resistant to narrow-band interference from other sources of transmission
4. FHSS allows multiple devices to transmit on the same frequency

The main consumer devices that use FHSS are devices that use the 2.4 Ghz and 5.8 Ghz bands. One of its main challenges is synchronising the transmitter and the receiver. One of the most common solution to this is to have the transmitter use all the allocated channels at a fixed time sequence allowing for the receiving device to detect the carrier signal by switching randomly between channels. Furthermore, once the signal is detected it checks if it's a valid signal sometimes using a checksum for checking the integrity of the packet and to confirm the identity of the device. The devices can also use fixed frequency hopping patterns using a set of tables, thus allowing the devices to synchronise using a specific pre allocated table[1].

3.2.7 Scrambling

The oldest method of obfuscating the content of RF transmissions is scrambling. This is done using a set frequency value to invert the frequency of the input signal which creates a "clone" of the signal on a different frequency range[54]. In other words, the signal is transposed to a different range and can only be decoded by a device using the same frequency offset and settings. Compared to other methods of protections, it is the most primitive as it can be easily broken and de-scrambled. Even more complex scrambling methods like split-band inversion are also prone to de-scrambling[46].

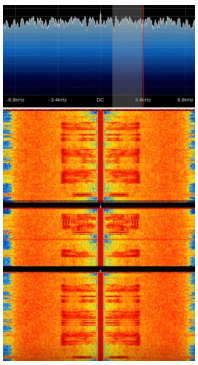

Figure 3.8: Scrambled Signal

For example, de-scrambling voice communication can be done by recording the signal we want to de-scramble using a program like "SDR#" and an SDR capable of listening on the specified frequency. After recording the IQ data we play it back using the same software and look for the inverse modulation. Depending on the modulation we can use either Upper or Lower Side band modulation to demodulate one of the two inverted modulations[54]. We can observe this phenomenon in figure 3.8.

3.2.8 Split-Band inversion

Split-Band inversion works by adding one more carrier frequency which divides the spectrum into two equal but opposite parts and then combines them both into one signal. The most difficult part is finding the two initial frequencies before the combination happens.

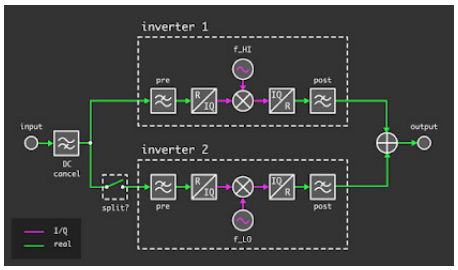

Figure 3.9: Dual Inversion splitter

Scrambling conclusion

There are automated programs that do this for us. One example is "deinvert" by Oona Raisanen which can automatically monitor a frequency using an RTL-SDR and do the whole processes without any fiddling around. This also works with Split-Band Inversion but at a lower success rate[56]. On its own scrambling is effectively weak and only stops the weakest of malicious actors from listening and sniffing data. It needs to be combined with other forms of security.

3.3 Equipment

In this section we shall explore the equipment used in the project. In our case it was necesary to use easy to obtain hardware which was relatively cheap but also versatile in performing the necessary attacks and alaysis techniques required to conduct our research. In our case we chose to perfom the majority of the attacks using the HackRF which we will go into its specifications in section 3.3.1. Furthermore, as we will explore ni section 3.3.2 we also used a cheaper receive only RTL-SDR (NESDR) because of its low cost and excellent price to performance. The use of a CC2531 module kit was also necessary as explained in section 3.13, to bypass some of the limitations of using proprietary software to control some devices.

3.3.1 Hack RF

The HackRF is a half-duplex SDR transceiver designed for RF investigation. It has an operating frequency of 1Mhz - 6Ghz with a 8-bit quadrature sample rate. The HackRF has a maximum transmission power of 30mW which makes it suitable for transmission at close ranges. Nonetheless, it is prone to emission of sporadic emissions which could compromise nearby equipment if used with amplification equipment. The HackRF is compatible with GNU-Radio allowing it be calibrated to work with multiple other equipment and protocols. One of the main disadvantages of the HackRF is its sub par receive performance . In our case we added a Temperature Compensated Crystal Oscillator (TCXO) to the HackRF to increase the receive performance to minimise frequency drift while receiving for long periods from time. We will explore the usefulness of a TCXO in the context of this research in the next section 3.12. The TCXO chip used in our HackRF is a Si5351 which is a high performance crystal oscillator which can operate at 0.1 PPM accuracy.

Figure 3.10: HackRF

3.3.2 RLT-SDR (NESDR)

The NESDR is a low cost SDR receiver capable of receiving from 25Mhz up to 1.7 Ghz. Its wide popularity has given rise to multiple software packages supporting it with a wide range of decoding protocols such as DMR,APRS,LORA,P25,FSK, etc[40]. The low cost of the NESDR makes it a good choice for testing and development purposes as is it has a much greater sensitivity than the HackRF when paired with a resonant antenna. In contrast to the HackRF the Smart SDR is only capable of receiving but has an overall better software compatibility. RTL-SDR based devices are based on two chips. The RTL2832U demodulator chip and the R820T2 tuner chip. It also features a 0.5 PPM(parts per million) low noise Temperature Compensated Crystal Oscillator (TCXO) chip capable of keeping the tuner in sync with the demodulator as it heats up.Since heat is a major factor in the signal stability of the tuner it is important to keep the device in a stable temperature range[40].

Figure 3.11: NESDR

3.3.3 Temperature Compensated Crystal Oscillator (TCXO)

Temperature Compensate Crystal Oscillators are used to correct the voltage of the tuner chip which results in variations in frequency over temperature differential. The voltage correction is applied to varactor diode located on the crystal circuit which varies the crystal frequency by a minuscule amount. This results in a much more stable frequency[32]. TCXO can reach a stability of 0.1 PPM but there are some issues being introduced. When a TCXO chip resides at a temperature extreme for long periods of time might a exhibit a frequency shift when it returns back to its normal ambient room temperature. This is a major issue for long term operation of the device as it might skew the results of the project. For example,when monitoring a device for long periods of time we can sniff packets being transmitted by the device we might also experience a shift in frequency causing loss of packets even when we are "monitoring" on the correct frequency of operation. This hysteresis error is temporary but it is important to take into account when monitoring a device for long periods of time [32].

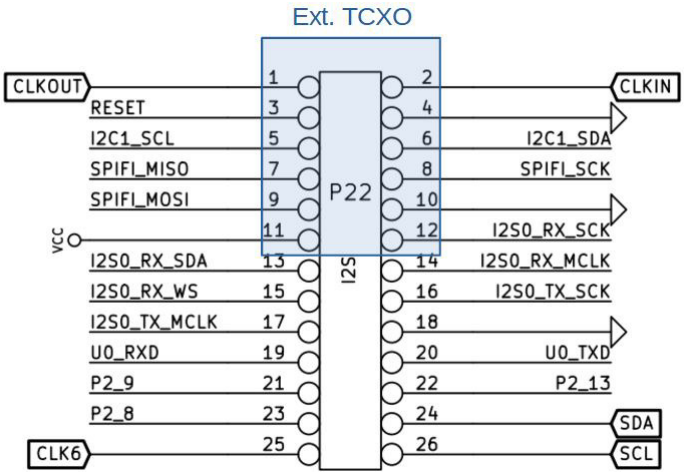

Figure 3.12: TCXO

3.3.4 CC2531 USB Evaluation Module Kit

The CC2531 chip provides an easy and convenient way to interface with the 802.15.4 Zigbee protocol, which we will explore in this paper.It shall allow us to decode and sync up Zigbee devices and provide us with verbose logging data that shall enable us to explore any vulnerabilities or shortcomings of the tested devices[15].The small PCB antenna allows us to test these devices within a small range without having to expose them to the unauthorised third parties. The small 8 pin headers on the board allow us to flash custom firm wire to the board depending on our needs. The on board CC2531 is one of the standard chips used for the Zigbee protocol along with the CC2530 which is a discrete level chip without the USB interface and the CC2538 which features a Arm Cortex-M3 which allows it handle much more simultaneous devices[15].

The low cost of the CC2531 allows home-brew enthusiasts to create there own IoT environment using free and open-source software, allowing them to break away from the chains of using proprietary services that might one day become obsolete. As we will explore in section 4.6, the use of the module kit was essential as downtime by the service provider prohibited testing from being carried out[15].

Figure 3.13: CC2531

3.3.5 Gnu Radio

Figure 3.14: GNU Radio

GNU RADIO is a open-source software development platform used to create virtual signal processing blocks which can be implemented with software defined radio such as the Hackrf. Using the platform allows low cost RF radio equipment to be "equipped" virtually with decoders,filters and processing blocks to encode and decode data in the RF spectrum. Using Gnu Radio allows us to make complex RF signal simulations without the need for expensive hardware thus reducing the cost of testing equipment. Many of the software that used through out this paper is using or is based on GNU radio like SDRangel[22], Sparrow WIFI[27], zigbee2MQTT [35] and rtl_433. Many of components of GNU radio are interchangeable and allow it to interface with multiple programs expanding its functionality dramatically. With a large community backing up GNU radio it has allowed it to be expanded and develop throughout the years.

3.4 Communication Protocols

In this section we will explore the different protocols we will interact with throughout this project,including details on how they work and their advantages and disadvantages. The communication protocols explored in this section are widely used in many popular IoT devices hence we need to understand how they work so we can analyse them for any vulnerabilities.

3.4.1 WiFi

WiFi has been commercially around since 1997, allowing devices to connect to a network wirelessly with speeds now reaching up to 2 Gbps. The WiFi protocol is a family of wireless network protocols that are based on the IEEE 802.11 standard and are mostly used for LAN (Local Area Network) communication[13]. It is one of the most used protocols for computer networking and is trademarked by the WiFi Alliance who control which devices are certified to use the protocol. WiFi uses the microwave bands to communicate specifically 2.4 GHz, 5 GHz and the up and coming 6 GHz. By using these high frequency they allow high data-rate transfers using line of sight[21].

Just like other wireless communication protocols, WiFi is set up to use "channels" to communicate between devices. These are split up in 20 Mhz chunks and for 2.4 GHz use up to 13 channels. It is also limited to a transmission power of 100 mW or 20 dBm[21].

3.4.2 Zigbee

The Zigbee protocol is a IoT specific protocol used for the transmission of data from sensors and automation control networks using the IEEE 802.15.4 standard. It uses very low data data rates of around 250Kpbs,which operated on the microwave frequency allocation of 868, 902-928 MHz and 2.4 GHz frequencies[8]. Usually higher frequencies are used to transfer data between devices at higher data rates but closer range. Zigbee devices have a maximum range of 100m (assuming propagation allows)[55].

Zigbee infrastructure

The structure of the Zigbee network consists of three main devices. The coordinator, Router and End Device. A Zigbee network cannot function without a coordinator, as is bridges the device with the network.The coordinator is in charge of storing and the information sent by the end device while the router will retransmit the data to the appropriate device[23].

3.4.3 Bluetooth

Bluetooth is a short range,low power mode for transferring data between two devices. It is usually used to sync between devices in a mesh network, send files between devices, stream music etc. It operates on the 2.4 GHz portion of the spectrum utilising 79 channels of 1 MHz bandwidth each[7]. Bluetooth network operates using a frequency hopping to minimising interference to other devices and try to mitigate any potential impact it might have to other devices. When two devices connect to each other using Bluetooth they initiate a handshake signal which usually requires authorisation of both parties with some sort of pop up notification or by entering a one time password. Bluetooth devices are organised a star topology meaning that there is one master server which sends data to multiple slave nodes[5]. For example, a master server could be a phone and the slaves are the wireless earphones or speakers. A Bluetooth packet is made up of three parts:

1. Access code
2. Packet Header
3. Payload

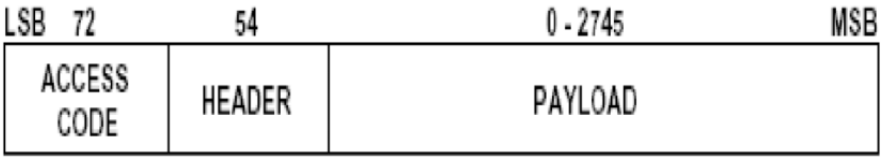

Figure 3.15: Bluetooth Packet

This allows the protocol to work either Synchronous mode for voice calls or Asynchronous Mode for data transfer[29].

Bluetooth Controller

The Bluetooth controller is in charge of implementing the physical layer (Layer 1) of Bluetooth which is made up of the radio, base band and link management layer. It establishes the initial communication with the destination device using the Host Controller Interface. The link management layer is in charge of creating that initial connection between two devices in turn use the base band layer to create an access control system through the radio link layer[29].

Host Controller Interface (HCI)

The HCI layer is a mediator between the controller and host. The host uses the host controller interface to establish a link between it and the Bluetooth controller[29].

Bluetooth Host

The Bluetooth Host houses the Bluetooth protocols logical stack components which allow the Bluetooth protocol to be used with different applications[29].

Chapter 4

Testing and Methodology

In this section we will explore how we set up and executed the testing, as well as the outcomes of each device tested. We shall also explore the reasoning behind each attack other outcomes that have been noted when performing specific attacks.

4.1 Set up

For conducting our experiments we placed the devices within close proximity to our software defined radios. To eliminate the chances of interference, we placed the devices in a large room with no other devices turned on in close proximity. Since our main receiving device, the HackRF, has poor receiving performance we did not have to worry about interference from devices from nearby rooms. We made sure to use an appropriately tuned 2.4GHz rubber duck antenna on our HackRF to allow it to transmit with almost zero loss. To further reduce the chances of interference from other devices we reduced the Low Noise Amplifier gain and Base Band Gain until we could only see signals coming from the tested device.
In the case of testing our Weather Station sensors we used an RTL-SDR (NESDR) which was connected to a Raspberry Pi using the "RTL_433" decoding software. This allowed us to filter out any unwanted signals and capture only the ones we needed. In addition to this we used multiple parameters with our decoding software to only decode the signal we wanted by referencing the "rtl_433" software manual.

4.1.1 Pre-programming

Even though we had all our devices ready we still needed to use the correct software and firmware to get them to work correctly . The cc2531 board had to be flashed with the sniffing firmware to allow us to use it with "Zigbee to MQTT" which would give us verbose logging of the testing process. By using this method we eliminate the need for third party servers and software which could go down at anytime . The problem with going through a third party service like "Hive" is that when the hive service goes down we are unable to properly configure and access the devices connected to a device like the "Hive Hub". This was the case when we initially tried testing the Zigbee protocol and the device would only start transmitting Zigbee packets once it was set up with an account from Hive. When we first tried setting up an account it would either not work or no confirmation email was sent out for us to actually activate it and use the account. To mitigate this we flashed a custom firmware on our cc2531 board by connecting to the device using a USB to serial connection through the 8 pin headers. The device used for this was a "Texas instruments CC debugger".

In figure 4.1 we can see how we connected up the device for flashing using the debugger. It only took us a few seconds to flash and could be reverted back at any time. After flashing the firmware we loaded up a Linux based (Parrot OS 4.11)[30] device to act as an MQTT server for connecting to our Zigbee devices via the board. Through the MQTT server terminal interface we are able to see a very verbose logging of the devices connected as well as giving us the option of specifying which channel to use for the initial communication. To avoid complicating things, we stuck with the default configuration on channel 11.

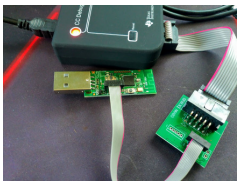

Figure 4.1: CC 2531 Flashing

4.2 Methodology

Before testing any of the devices we need to analyse their characteristics, specifications and how they operate. By using the labels listed on the device such as indications of certifications of specific protocols like WiFi , we can then start to assess the situation with the types of attacks we can perform. When investigating our devices we used the following questions to determine the approach we would take when attacking the device:

1. **What protocols does the device use?**
   By first identifying the protocol that were used by the device such as WiFi , Bluetooth , Zigbee or a custom protocol we researched any known vulnerabilities that may exist first before performing our own customised testing.

2. **What frequencies does the device operate on ?**
   Using either the FCC listing or our software defined radios we determined the frequencies of operation of the devices any search for any possible harmonic or spurious emission coming from the device. In doing so allowed us to plan our attack and analysis of the device.

3. **What other unique characteristics does the device possess?**
   Some of the devices such as the InFactory sensor tested in section 4.5 was using a custom protocol, which was a unique characteristic of the device. This meant that we needed to create a custom attack vector such that it would conform to the specifications of the device.

4. **Does the device contain sufficient filtering?**
   In many cases especially in smaller device were space is limited, sufficient filtering and protection from out side interference. This was the case with one of our devices were the design of the device made it so that it could easily be stopped from working by either miss placing or putting some in front of it. We explore this phenomenon is section 4.6.

5. **Is the device dependant on a third party?**
   Many IoT devices are dependant on third party applications which make them non operational without them or with some kind of custom firmware being installed on the device. This was the case with some of our devices which relied on third party services to operate which meant that we had to find a way around the issue by installing customised firmware on the receiving device.In doing so we were able to critically analyse the device in question in section 4.6.

By using the FCC listings for most of the devices we could take a deeper look, into the inner workings of the device and what components are being used that could possibly be exploited. As we will explore in section 4.15 it was crucial in the effectiveness of a low skill and low effort attack.
We made sure our tests were reproducible and repeated every attack at least three times to make sure it was

effective and not a fluke, but as it was observed in some cases the randomness of Frequency hipping spread spectrum functionality of some of the protocols prevented us from reproducing some of the results. This was expected to happen at some point and was recorded as such accordingly.
Furthermore, it should be noted that not every attack was possible on all devices, as such we needed to be flexible and creative in the types of attacks we chose to investigate further.

4.3 Belkin NestCam HD

The Belkin Nestcam HD is a small compact wireless camera which is used for internal surveillance. It uses a 720p camera for capture with a maximum transmission power of 67.5 mW (Which is well below the maximum transmission power allowed by the FCC of 100 mW). With this much power being available for transmission and the use of a small PCB antenna, it is only able to be received in a small area[24]. We can observe is small size in figure 4.2.

Figure 4.2: Belkin NestCam HD

The device is controlled using a proprietary app by Belkin, which is now being discontinued. This means that the device itself will become useless unless custom firmware is installed to allow it to function without the software. Even if you access the cameras web interface there is no way to receive the feed without using the software, since it uses authentication from Belkin services to access the feed.

4.3.1 Testing

By using Sparrow WiFi in conjunction with our HackRF we deduced that the Belking Netstcam was operating at 2.437 Ghz (Channel 6), we were also able to monitor any clients that connected to the device or at least initiated a connection.

Using Sparrow WiFi, we managed to use its inbuilt de-authentication function which uses both the HackRF and the computers WiFi adaptor. This produced valid results and disabled our devices from connecting to the camera directly. If any of the devices had been previously connected it would kick them off the network and disallow them from connecting back again.

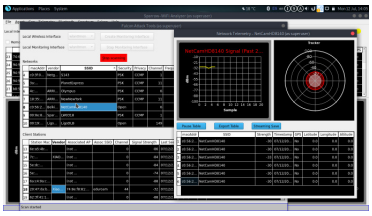

Figure 4.3: Sparrow WiFi De-authentication

In essence, it would make sense that using Sparrow WiFi would be the most successful, since we can cover the whole channel of operation of the device using the laptop's inbuilt WiFi adaptor. Because of its larger bandwidth compared to the HackRF, this is more plausible.

Furthermore, we also used the HackRF on its own. We started by first monitoring the frequency for IMCP Packets which we sent using our phone. Once we captured the packets we re-transmitted the packets using the HackRF on the same frequency. We used "hackrf_transfer" to cause a primitive jamming attack by monitoring the frequency 2.437 Ghz (-f) with a sample-rate of 2 Msps (Mega samples per second) and a receive sensitivity of 40 db and we finally saved it to a file called test.bin.

```

hackrf_transfer -r test.bin -f 2437000000 -s 20000000 -l 40


hackrf_transfer -t test.bin -f 2437000000 -s 20000000 -x 47
```

Attacking First Harmonic

Further testing has been done by sending IMCP packets on a harmonic frequency of 1.2185 GHz, which has a second harmonic of 2.437 GHz. Using the same method as above but just changing the frequency of transmission, we observed that there was, a small increase in latency although that was not very substantial. This could be due to many factors such as external interference making it an unreliable.

This resulted in spikes in latency from the phone to the camera but no packet loss. We can observe this from both figure 4.4 and figure 4.5. This could be due to the following reasons:

1. The HackRF does not transmit at a bandwidth large enough to cover the whole channel on its own. Even though the nest cam does transmit using only 20 MHz bandwidth , it is likely stronger than the signal being sent by the HackRF thus overpowering it.
2. The low power transmitted by the Hackrf is not enough to completely overshadow the transmission of the Nestcam.
3. WiFi switches between modulation types therefor if we are jamming using only one form of modulation, we might only temporally manage to cause some disruption.
4. Even when using Universal Radio Hacker we where unable to decode any packets primarily, because of the lack of bandwidth.

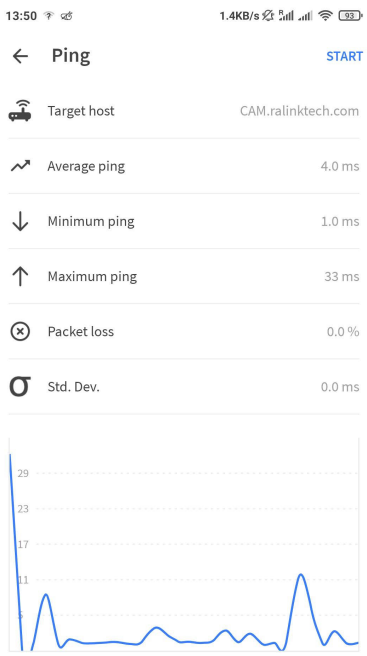

Figure 4.4: HackRF 2.437 GHz

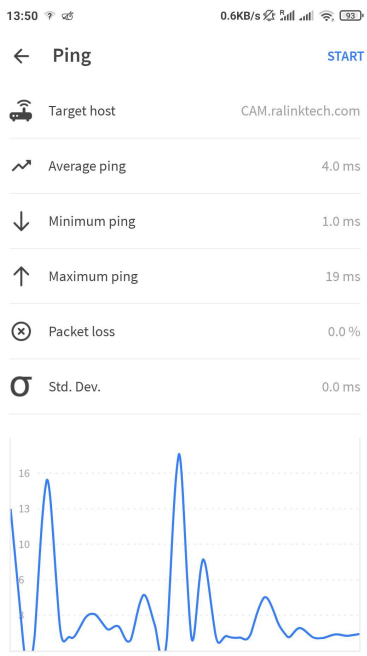

Figure 4.5: HackRF 1.2185 GHz

Sniffing

With refrence to figure 4.6 by using the laptop's inbuilt WiFi adaptor instead of the HackRF which has limited bandwidth, we enabled monitor mode on the adaptor and saw the packets being sent out by the camera. After inspecting the packets, there were no obvious ways we could capture and manipulate the packet or the ability to decode the basic authentication it has implemented. Instead, we saw multiple beacon packets being sent out and even sending out IMCP packets to the camera with the HackRF did not stop it from sending out beacons or being detected.

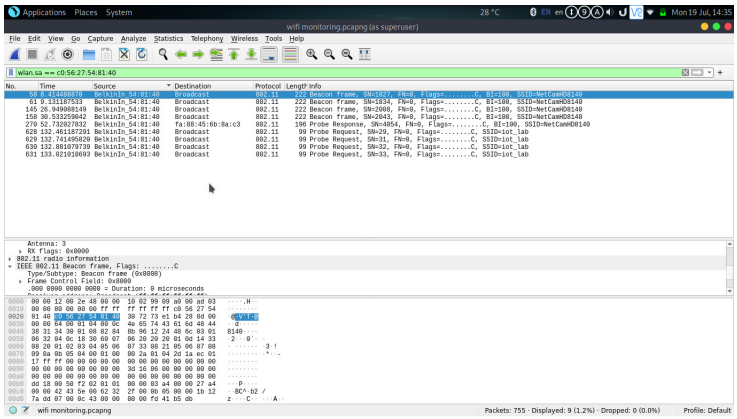

Figure 4.6: WiFi Sniffing

4.3.2 Conclusions on Belkin NestCam HD

What we can take away from this is that we could possibly cause disruption of service by using a higher gain antenna and much stronger amplification of the signal. This would potentially disrupt the signal enough to cause substantial packet loss. In doing so, we could possibly attack the device from a larger distance making it almost undetectable without specialist radio equipment. More specifically, some form of triangulation would be necessary to detect the direction of the signal.

4.4 NEST Smoke alarm

Figure 4.7: NEST Smoke Alarm

The NEST smoke alarm uses both Bluetooth and WiFi to connect to the internet and a nest hub. It uses Bluetooth 4.0 with a max transfer rate of 1mbps. It uses a modulation type of GFSK , operating between 2.402 and 2.480 Ghz. The channel spacing used by the device is 2 MHz which makes it ideal for the HackRF to sniff at least one channel[25]. Looking on the inside of the device we can see that it uses a meandering mono-pole antenna with a gain of 0.29dBi and a maximum transmit power of 0.995 mW. The internal coax connections are done using a Sucoflex 104 which is a low loss , high stability cable with shielding acceptable up to 26.5ghz [52].

De-authentication

By using sparrow WiFi we could see that the device was using channel 6 for WiFi. We where successful in de-authenticating the device using deauthetication packets from both the HackRF and the WiFi adaptor. By using only the HackRF we could see a small rise in ping. The same methodology as the previous testing was followed.

```

hackrf_transfer -r test.bin -f 2437000000 -s 20000000 -l 40


hackrf_transfer -t test.bin -f 2437000000 -s 20000000 -x 47
```

Furthermore, we have also sniffed the WiFi packets using our WiFi adaptor in monitor mode. By analysing the packets we determined normal operation and nothing out of the ordinary was present. The only interesting result was that the device was continuously sending out beaconing packets at very small intervals.

4.4.1 Bluetooth Sniffing

By using the HackRF we have also sniffed part of the Bluetooth spectrum to capture the connection between the mobile phone and the Nest Device . Because of the lack of bandwidth available on the HackRF we cannot sniff the whole Bluetooth spectrum hence we only focused on channel 37 which was the most commonly used one. This resulted in multiple malformed packets being received with the only "useful" packets being decoded being the broadcast packet by the NEST device.

As we can observe from figure 4.8 we can see all the packets being sent out from the NEST device were mostly beaconing packets so other devices could connect to it.We have captured them all successfully but comparatively to figure 4.9, we can deduce that the bandwidth that the HackRF can receive is not large

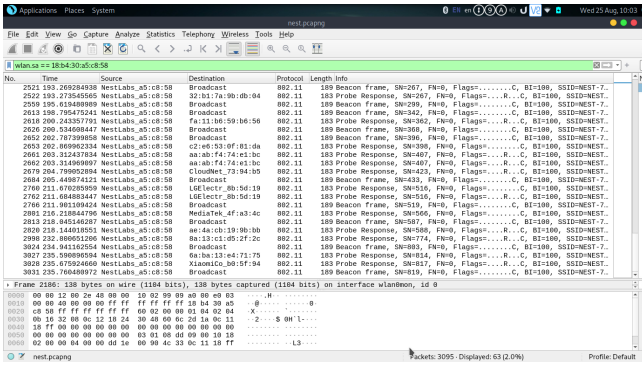

Figure 4.8: Using Bluetooth Receiver

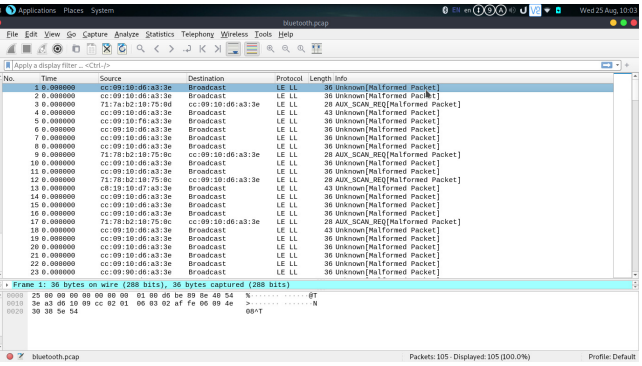

Figure 4.9: Using HackRF

enough to receive the whole packet making it useless for attacking Bluetooth devices. The lack of bandwidth means we only managed to receive the first part of the packet, which just states the name of the device, but we could not decode the whole message.

4.4.2 Jamming with IMCP

While sniffing the packets there was no way for us to receive the handshake signal, which could allow us to perform a replay attack. Even using the "hackrf_transfer" function to save the raw data and re-transmit them back did not enable us to see any effect on the device or the network it was connected to. Furthermore, by using a harmonic freq of 1.2185 Ghz we were able to slightly increase the latency between the device and other devices but not substantial enough to cause any issues.

4.4.3 Conclusion on NEST Smoke Alarm

The informative conclusion is that the results where interesting and paved the way for more investigation. The most successful of the attacks was de-authenticating the device, which was the easiest to perform using our current hardware. Expanding, we achieved increasing the latency of the device by transmitting IMCP packets using the HackRF on both the carrier frequency as well as the harmonic frequency. Although it was not a substantial increase, there is room for improvement using more high powered equipment to over-saturate the frequency band. As far as sniffing Bluetooth packets is concerned, the lack of bandwidth available on the HackRF did not allow us to fully capture the Bluetooth packets and made it difficult to conduct any form of Bluetooth related attacks.

4.5 inFactory sensors

inFactory sensors are widely used sensors which can be found in many weather stations. Many of these weather stations either have built in functionality to send data over the internet or use another third party device to do this for them. They usually operate on 433.920 Mhz using FSK modulation which send out packets to neighbouring monitoring devices which display the data. This data is sent using an unencrypted signal which can be decoded using the "rtl_433" software. Many IoT enthusiasts have used "rtl_433" to decode the data and perform analysis on the data. We can observe an image of the sensor in figure 4.10.

Figure 4.10: inFactory Sensor

4.5.1 Attack Scenario

Many of these weather stations send data to the internet which are used by community driven services. This data is then used to create graphs and send out notifications to users. The data is then stored on the internet and can be accessed by anyone. This is a potential threat, since if the service becomes overrun by malformed information which could render the information useless. One such service is the Citizen Weather Observer Program (COWP) which is a community driven service providing weather data using community driven weather stations. This information is made available and used by weather services and homeland security.The COWP is a great example of a service which is not only used by the community but also by the government. COWP lists on their website that they are used by over 800 different government and non-government organisations [3].

4.5.2 Data Acquisition

To acquire our data we used "rtl_433" to decode the data and perform analysis on the data. We were able to see that the data was being sent out using a frequency of 433.920 MHz. This frequency is used by the COWP weather stations to send out data. We used the following command to decode, store the signal and later perform analysis on the data using Universal Radio Hacker. The flags used in the command are as follows:

```

- -d: Device number
- -R: Device ID (at specified in "rtl_433" wiki)
- -a: Analyze mode 4
- -A: Pulse Analysis
- -S: Store all raw signals

rtl_433 -d 1 -R 91 -a 4 -A -S all
```

As we can deduce from figure 4.11 the data is being sent out in imperial units in Fahrenheit and then being converted to Celsius (if the user wants to) on the receiver. In this case, the temperature is 85.70F with a humidity of 61%. We can also see that the channel being used is listed, which in this case is channel 1. This information will prove helpful in trying to reverse engineer the packet using Universal Radio Hacker for packet manipulation as we will explore in section 4.5.5

```
time      : 2021-07-25 20:40:00
model     : inFactory-TH ID        : 87
Channel   : 1            Battery OK: 1            Temperature: 85.70 F     Humidity  : 61 %         Integrity : CRC

time      : 2021-07-25 20:40:01
model     : inFactory-TH ID        : 87
Channel   : 1            Battery OK: 1            Temperature: 85.70 F     Humidity  : 61 %         Integrity : CRC

time      : 2021-07-25 20:40:01
model     : inFactory-TH ID        : 87
Channel   : 1            Battery OK: 1            Temperature: 85.70 F     Humidity  : 61 %         Integrity : CRC

time      : 2021-07-25 20:40:01
model     : inFactory-TH ID        : 87
Channel   : 1            Battery OK: 1            Temperature: 85.70 F     Humidity  : 61 %         Integrity : CRC

time      : 2021-07-25 20:40:01
model     : inFactory-TH ID        : 87
Channel   : 1            Battery OK: 1            Temperature: 85.70 F     Humidity  : 61 %         Integrity : CRC
```

Figure 4.11: Infactory sensor data decoded

4.5.3 Jamming

Many of the sensors are housed in thin plastic casing with little or no shielding from the elements. Furthermore, the circuit board itself has minimal protection from external RF interference which makes it easy to jam and break the connection between the screen and the sensor. To achieve this we sent out a signal using a HackRF irrelevant of modulation mode as it would still be more powerful than the sensors internal amplifier. We were able to jam the connection between the HackRF and the sensor, and also able to take this a step further and use a handheld transceiver to transmit a carrier wave on 433.920 MHz with more than 1 watt. This could potentially jam multiple devices on the same frequency and at a much larger distance.

4.5.4 Replay Attack

The simplistic nature of the protocol used, as well as the lack of authentication and encryption, makes it quite easy to replicate the signal. Furthermore, the protocol is very simple and can be easily modified to send out a different message. This attack is very effective as it requires no knowledge of the protocol and can be easily replicated. By capturing the packet using Universal Radio Hacker we were able to analyse and deconstruct it in a readable format . This was achieved using six different samples of the packet to allow us to cross reference between them and also detect any change that might be present in the packet.

To perform the replay attack we placed the sensor in a very cold place giving us an unrealistic outside temperature so that we could easily detect if our replay attack worked. We placed the sensor in the freezer for an hour so that it would reach a temperature of -9c degrees and then using URH we captured the packet that was being sent. After allowing the sensor to warm up again and return to room temperature, we tried sending back the signal to receiver. Unfortunately this did not work and we tried the following methods to get it to work but without any success:

1. Removing the batteries from the sensor and sending the signal to the receiver without the sensor present
2. Resetting the receiver and then sending the signal
3. Sending the signal at 30 second intervals
4. Capturing the handshake signal sent out by the sensor , disabling the sensor and using that signal first, before sending out the data packet

Even though none of these methods worked in producing a replay attack, there was a side effect that became apparent. After performing the attack with the sensor off, when we turned it back on again it would not sync up with the receiver even though we used the "sync" button located on the back of the device. This led us to believe that we had managed to "desync" the internal clock of the receiver which prevented any "new" devices to connect to it. Even though we tried changing the channels of the sensor in case that was the problem, it would still not connect to the receiver. The only way to get the devices to sync up again would be to reset the receiver by removing the batteries from the device.

4.5.5 Packet Manipulation

Building on our replay attack and the decoding of the signal using URH , we can further try to modify the packets using signal generator packet. After changing values randomly in the data part of the packet, we were able to send out a modified packet although this was not decoded by the receiver. Upon further inspection there seemed to be an initial synchronisation preamble which was unique every time the devices would sync together.

While trying to reverse engineer the packet we deduced that the first 12 bits are used as a preamble to synchronise the local time with the remote time. The next 4 bits are used to encode the synchronisation and the remaining bits are used to encode the message being sent. With reference to figure 4.12 we can see this action using URH. By using multiple samples in our analysis we can confirm this was the case, as this was consistent throughout the samples with some minor anomalies which could be due interference from other sources.

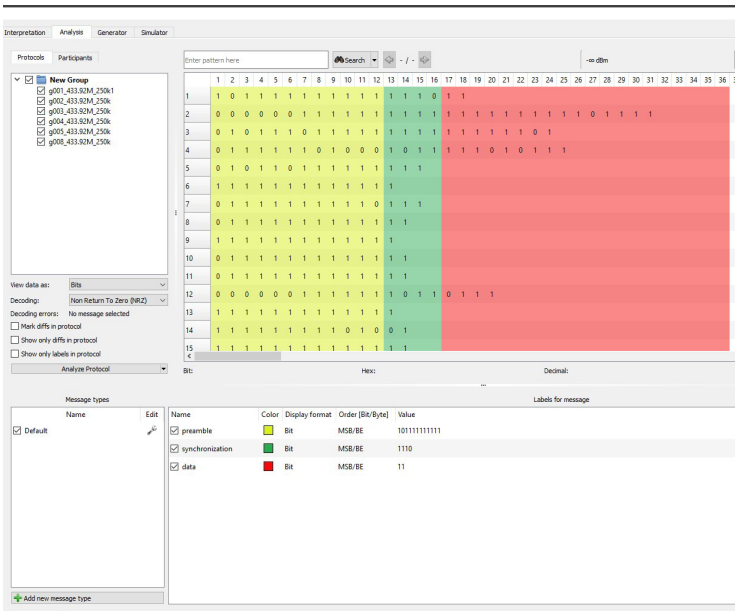

Figure 4.12: Infactory Packet Analysis

The packets are being sent in 67 second intervals which gives us a 67 second window to send out a message. We can observe this in figure 4.13 where URH has detected a window of 67 seconds between each packet. At

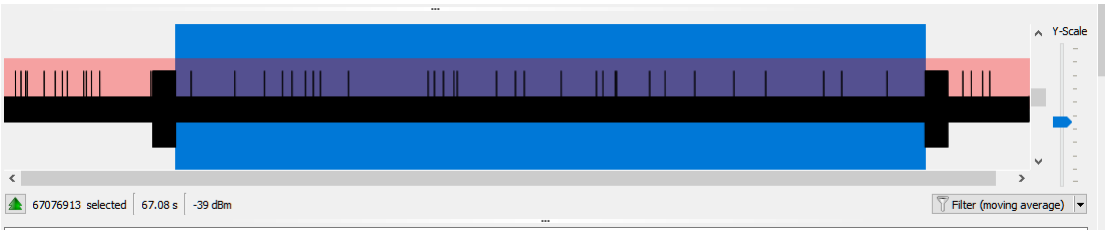

Figure 4.13: Weather station interval

first glance it seemed to be one large packet containing the data but as we zoomed in we could see 6 smaller chunks of data being sent out. Figure 4.14 illustrates that this became extremely obvious once we zoomed and we can further explore these chunks and deduce that the are of length 311.90 milliseconds each with a 31.5 millisecond gap between them.

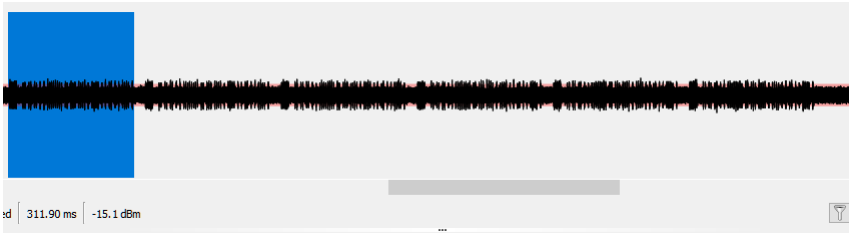

Figure 4.14: Weather station Packet Chunks

By taking into account figure 4.11 we can deduce that this is correct and not a fluke since "rtl_433" has actually decoded the same message 6 times in the same interval(due to the size of the terminal there is one off screen) .

The message is sent out using the same protocol as the sensor is using and is interpreted by the receiver. The message is then decoded and displayed on the screen. But disappointingly in our case this did not happen as we were unable to manipulate the packet correctly in order to get it to be decoded by the receiver.

4.5.6 Conclusion on Infactory Sensors

Overall our findings were positive as we still managed to analyse the packet and perform some of the attacks stated in this paper. While not all of them were successful, we still gained valuable information on the side effects of trying these attacks like de-syncing the two devices. This could potentially cause disturbance in a large area as this could "knock out" multiple sensor simultaneously rendering the readings useless until someone physically accessed the device to perform a hard reset. The possibility of sending a manipulated packet it still there, but further research is needed to correctly perform the attack.

4.6 SONOFF SNZB-02

The Sonoff SNZB-02 is a small Temperature / Humidity monitor that operates using the zigbee protocol. It is very small in size as we can deduce from figure 4.15. It uses a small internal PCB antenna with a maximum power of 3mW or 4.58 dBm. This was achieved by using a cc2530 chip which is a very small and low power radio chip[26]. The device uses a standard 2.4 Ghz RF frequency and uses GFSK modulation which is a very low bandwidth and low power modulation. The device itself is housed in a small plastic casing with minimal shielding from other forms of RF interference. This device is very small and can be used in a small space. The device itself will send out a beaconing packet alerting any receiving device of the current temperature and humidity of the area[26].

The device sends out data using metric units while also sending telemetry for monitoring the health of the device. The SNZB-02 sends out a packet containing:

1. Temperature in Celsius with a resolution of 0.01 degrees
2. Humidity in % an accuracy of 0.01
3. Battery Level from 0 to 100
4. Link Quality from 0 to 100
5. Voltage in millivolts

Figure 4.15: SONOFF SNZB-02

The device itself suffers from the poor design element of not having a way to turn off the device. This means that if the device is left on for too long it will continue to send out packets and consume power but also the positioning of the PCB antenna will affect the signal strength .

As we can observe, in figure 4.16 the PCB antenna is located next to the cc2530 chip on the very edge of the board. Although there is very little room for improvement of the board design we can still improve the design by moving the PCB antenna to a different side of the board or even adding an IPX connector to add a small wire antenna.This would allow us to reduce signal loss of the device but also increase its transmission range. This is important, as a lot of the devices are not able to receive signals from a distance greater than a few meters. This has the benefit of reducing the chances of a rogue device being able to intercept the signal.

Expanding on this dilemma, a very low skill attack of just sticking some aluminium tape on the side of the device could potentially block the signal from being received by the coordinator.

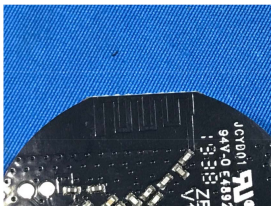

Figure 4.16: Sonoff SNZB-02 PCB antenna

Replay Attack

Using Universal Radio Hacker we were able to capture the signal of the device, yet due to the nature of the Zigbee protocol and its use of frequency hopping, we are unable to successfully replay the initialisation packet. This means we are unable to connect to the coordinator using our HackRF.
Furthermore, just like Bluetooth, the low bandwidth used by the Zigbee protocol means that we are unable to replay the entire packet. We were able to capture part of the packet but not large enough to decode or analyse it. Even though we took this into account, we managed to briefly replay the packet back to the device when it was listening channel 11. This was only momentary and only happened for a few minutes, after that we were unable to do the same thing again for some time.

With reference to figure 4.17 we can deduce the following:

1. The replay attack was successful since the temperature and battery values are the same through out.
2. The link quality is calculated by Zigbee2MQTT, not by the sensor, as we started moving the HackRF further and further away from the receiver to test the range.
3. The values for humidity seem drastically different, even if it was the actual sensor that was sending out the signal the difference is too big and within a few seconds we can deduce that it might be some malformed packets being sent out by the HackRF.

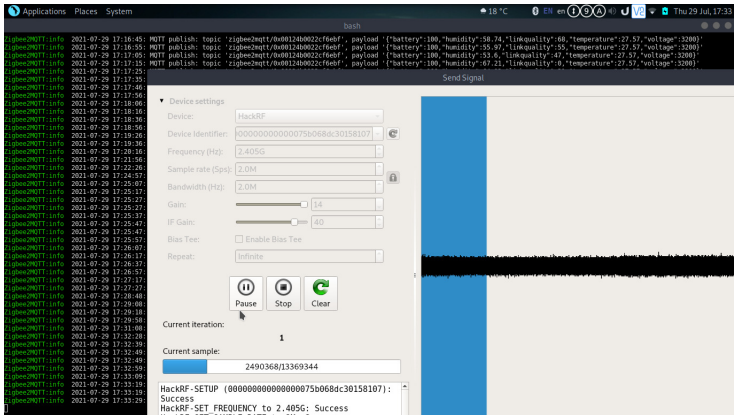

Figure 4.17: Replay Attack

Overall the replay attack was somewhat successful since we only managed to do this briefly. Despite that, it could have catastrophic effects if the user would have the sensor connected to a thermostat. One scenario would be that the thermostat could turn up the heat when it was already quite warm in the building which could cause heatstroke or worse especially to people suffering from high blood pressure or sensitive to heat.

Jamming

Like most devices, the device started sending a beaconing packet on channel 11 (2.405 GHz). Channel 11 is the channel most frequently used by devices in the 2.4 GHz band to initiate the synchronisation of the local time with the remote time. This channel is also used by the Zigbee protocol to send out the beaconing packets. But, since the Zigbee protocol is designed to use frequency hopping it becomes difficult to actual jam the channel. But after further investigation we manage to Jam the signal using SDRANGEL using its 802.15.4 module. By transmitting a random Zigbee packet we were able to stop the receiver from gathering telemetry from the device. It is worth mentioning that this only worked after synchronising the device with the coordinator. Furthermore, it should also be noted that the HackRF has a much greater transmit power than the Sonoff device meaning that we were just overlapping the signal. We used a bandwidth of 10 MHz just enough to jam the channel and the one next to it, allowing us to stop the device from hopping over to a "free" channel".

With reference to figure 4.18 we can see that we manged to stop the device from being received for three minutes between 1436 and 1439. Once the device hopped to an other channel it would then start being received again by our sniffer on the MQTT server.

With reference to figure 4.19 We have set up sent out a random Zigbee packet and pushed the HackRF to its limits by setting the transmit power to 61 and the LNA gain to 47 which gave us the best possible scenario in jamming both channels. In our case we tried jamming channel 11 and since we sent out a 10 Mhz packet we also managed to jam the channel right next to it which was channel 12.

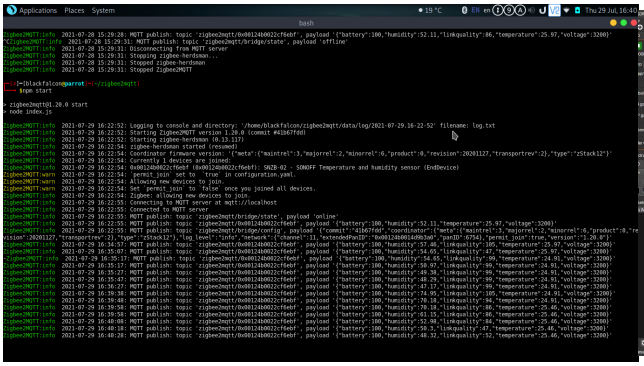

Figure 4.18: MQTT

Figure 4.19: SDR Angel

4.6.1 Conclusion on Sonoff SNZB-02

Upon exploring the situation from multiple perspectives, we have to take some things into account. Although we were able to successfully jam the signal, it was only possible if it was done during the synchronisation of the devices. Furthermore, even though replaying the packet was somewhat successful it was only momentary, with the only attack pattern being to first disrupt the signal and hope that it would cause more some miss-configuration, or pre-programmed functions like turning on the heating to be delayed.

Chapter 5

Conclusion

The project was overall successful in achieving its goal with some limitations. The main goal of the project was to investigate radio frequency vulnerabilities in the Internet oF Things. In this respect the project achieved what it was set out to do. All four devices tested proved to have some sort of vulnerability even if in some cases it was minor. The project has set a precedent for future work as its testing could be expanded on other devices of similar nature that could hopefully reveal even more vulnerabilities. The results from testing the Belkin NestCam were positive as we did manage to create some disruption during the overall testing. Even though we did not manage to replay the packets or disrupt the signal completely using our software defined radios, the increase in latency proves there is room for improvement using more powerful hardware. Expanding, when investigating the NEST smoke alarm, the use of both Bluetooth and WiFi gave us multiple points of attack. Although the limited bandwidth of the HackRF did not allow us to fully disconnect or perform a replay attack on the device, the use of the internal WiFi adaptor of the laptop in conjunction with the HackRF allowed us to performed a de-authentication attack disconnecting the device completely from the network. Furthermore, moving on to the inFactory sensors we managed to reverse engineer the packet, decode it and disrupt the transmission and operation of the sensor and receiver. In essence our testing was successfully even though we did not manage to manipulate and send back the packet. Concluding, when testing the Sonoff SNZB-02 there were some positive results. The disruption of the signal was successful both physically and using the HackRF. By using SDRAngel we were able to disrupt the signal of the sensor for some time until it switched channels. Although this was temporary this could possibly be detrimental if the device used for turning on and off the heating in a building.

5.1 Future Work

The project can be further expanded using more sophisticated equipment to investigate the devices much more thoroughly. There are many aspects which could be investigated further, for example deeper analysis of Bluetooth in the RF spectrum could be investigated by using higher bandwidth equipment like a USRP SDR or an Ubertooth. It should also be noted that as the field of IoT expands, more and more devices will flood the market which shall increase the possibility of more vulnerabilities being discovered. As proved by other studies we could leverage the use of machine learning or Artificial intelligence to create more sophisticated attack and defence techniques that would be difficult to produce manually.
Long term expansion of the project could be to introduce the use of Electromagnetic pulses to disrupt the use of the devices at a longer range. In addition, we could also investigate the use of magnets and the effect they could have on the devices, especially ones that use relays. It should be noted that many smart devices that use analogue forms of operation, such as relays, are vulnerable to manipulation using magnets, hence it should also be investigated if we can operate them using radio frequency.
With the adoption of wireless tire pressure sensors by major manufacturers, investigating the manipulation of those packets and the impact they could have on the operation of the vehicle could be a major stepping stone in securing "smart" cars.

5.2 Project Appraisal

The project had a set goal from the beginning and has met that goal. The research and testing time allocated to the project was consistent and allowed the project to run smoothly. Most of the work was done early on to compensate some time if things would go wrong. This has allowed more techniques and tools to be used for testing which resulted in higher quality results. To keep up with the overall scope of the project, the FCC reports of the devices were used to analyse the internal workings of the devices as well as the fine details the manufacturers don't disclose to the consumers (as they are too technical). Expanding, the use of a TCXO chip in the HackRF allowed us to reproduce our results and also make sure our readings were within margin of error. Since we were constantly using the device, heat build up would cause a frequency drift when receiving which might have caused inconsistencies in our results. As with every project it did not come with its short comings, some of which where due to the fact that software becomes outdated and software gets patched. Some example where :

1. GNU radio modules being depreciated in favour of newer ones meant that some software stopped working and we had to use alternatives instead. In our case we used SDR angel to over come the depreciation of the gnu radio to wire shark module which we would like to use for our Zigbee attacks.
2. Hive account creation services were down, thus we could not create an account to use our hive devices. One of its big limitation was that for it to actually emit a Zigbee packet (for us to analyse) we needed to create an account. To get around this we created our own Zigbee network using the cc2531 and the Sonoff sensor.

Despite some of the challenges faced during the project produced some positive results. Even the simplistic attacks and some of the "unwanted" consequences caused by the attacks have shown that there is plenty of potential for more sophisticated attack and defence methods to take place. Overall some light has been shed on the vulnerabilities of IoT devices and the danger they pose if not installed and operated correctly. This will hopefully help the battle against malicious threat actors in the world of IoT.

5.3 Self Reflection

The scope of the project was very ambitious as the probability of finding any new vulnerabilities is always low. Given that I had most of the background of radio frequency covered because of the my ham radio licence, exploring the use of software defined radio as a means of conducting vulnerability assessment was quite new to me. Thus, I organised my time accordingly sparing no time to researching the tools I would need to complete the task at hand. I spent most of the early days of the project researching for different papers that would prove useful for my project as well as talking to experts that either use software defined radio or more analogue means of conducting research. This helped me move on with the project faster as I spent less time debugging and more time testing my devices.
In essence the project achieved its goal even if I did not manage to find any new vulnerabilities because I managed to prove that there are still attacks that can be done to these devices even if some of them are "low-skill". Overall the project was very rewarding and helped in advancing my knowledge in the world of cyber-security and the world of radio frequency. I have gained valuable skills that will prove useful in upcoming projects that will hopefully benefit the community.

Bibliography

[1] In: *Principles of Spread-Spectrum Communication Systems* (Oct. 2005). DOI: 10.1007/b99535.

[2] Aug. 2021. URL: https://www.tutorialspoint.com/wi-fi/wifi_radio_modulation.htm.

[3] URL: http://wxqa.com/.

[4] *8.2mhz eas system*. Feb. 2020. URL: https://www.enguard-security.com/rf-antenna/eg-rf01-antenna.html.

[5] *A Beginner's Guide to Bluetooth Technology and How Does it Work*. May 2008. URL: https://techspirited.com/how-does-bluetooth-work.

[6] AFP. *Cyprus Arrests Three in 'Israeli Spy Van' Probe*. Dec. 2019. URL: https://www.securityweek.com/cyprus-arrests-three-israeli-spy-van-probe.

[7] Scientific American. *How does Bluetooth work?* Nov. 2007. URL: https://www.scientificamerican.com/article/experts-how-does-bluetooth-work/.

[8] Imran Amin and Atif Saeed. "5.10 Wireless Technologies in Energy Management". In: *Comprehensive Energy Systems*. Ed. by Ibrahim Dincer. Oxford: Elsevier, 2018, pp. 389–422. ISBN: 978-0-12-814925-6. DOI: https://doi.org/10.1016/B978-0-12-809597-3.00524-1. URL: https://www.sciencedirect.com/science/article/pii/B9780128095973005241.

[9] Wilson Amplifiers. *dB vs. dBi: What Gain Actually Means*. Aug. 2021. URL: https://www.wilsonamplifiers.com/blog/db-vs-dbi-what-gain-actually-means/.

[10] ASP. *Cyprus police investigate Israeli owner's 'spy' van*. Nov. 2019. URL: https://abcnews.go.com/International/wireStory/cyprus-police-investigate-israeli-owners-spy-van-67069061.

[11] *Attack Landscape H1 2019: IoT, SMB traffic abound - F-Secure Blog*. Sept. 2019. URL: https://blog.f-secure.com/attack-landscape-h1-2019-iot-smb-traffic-abound/.

[12] W. P. B. et al. "Kahn on Codes: Secrets of the New Cryptology". In: *Foreign Affairs* 62.4 (May 1984), p. 1009. DOI: 10.2307/20041931.

[13] Vangie Beal. *WiFi*. July 2021. URL: https://www.webopedia.com/definitions/wifi/.

[14] Gilles Callebaut et al. "The Art of Designing Remote IoT Devices—Technologies and Strategies for a Long Battery Life". In: *Sensors* 21.3 (Jan. 2021), p. 913. ISSN: 1424-8220. DOI: 10.3390/s21030913. URL: http://dx.doi.org/10.3390/s21030913.

[15] *CC2531EMK*. URL: https://www.ti.com/tool/CC2531EMK.

[16] Jian Chen et al. "Practicing a record-and-replay system on USRP". In: *Proceedings of the second workshop on Software radio implementation forum - SRIF 13* (Aug. 2013). DOI: 10.1145/2491246.2491257.

[17] Joseph COX. *NSO Group Closes Cyprus Office of Spy Firm*. Aug. 2020. URL: https://www.vice.com/en/article/ep48kp/nso-group-cyprus-circles-bulgaria-ss7.

[18] CAS DATALOGERS. "THE BASICS OF SIGNAL ATTENUATION". In: *CAS DATALOGERS* (). URL: dataloggerinc.COM.

[19] physicsforums Delta2. $Are_Electromagnetic_Waves_Always_Transverse$?. Apr. 2021. URL: https://www.physicsforums.com/insights/are-electromagnetic-waves-always-transverse/.

[20] draytek. *Home*. Mar. 2014. URL: https://www.draytek.co.uk/support/guides/difference-between-db-dbm-dbi.

[21] *Electronic Code of Federal Regulations*. Aug. 2021. URL: https://www.ecfr.gov/cgi-bin/retrieveECFR?gp=1&SID=aad0dee9d5dd560e44f29c7d58918f9d&h=L&mc=true&n=pt47.1.15&r=PART&ty=HTML.

[22] f4exb. *f4exb/sdrangel: SDR Rx/Tx software for Airspy, Airspy HF , BladeRF, HackRF, LimeSDR, PlutoSDR, RTL-SDR, SDRplay RSP1 and FunCube*. URL: https://github.com/f4exb/sdrangel.

[23] Shahin Farahani. "Chapter 2 - ZigBee/IEEE 802.15.4 Networking Examples". In: *ZigBee Wireless Networks and Transceivers*. Ed. by Shahin Farahani. Burlington: Newnes, 2008, pp. 25–32. ISBN: 978-0-7506-8393-7. DOI: https://doi.org/10.1016/B978-0-7506-8393-7.00002-9. URL: https://www.sciencedirect.com/science/article/pii/B9780750683937000029.

[24] fccid. *Belkin , . NetCam HD F7D7602V2 FCC ID K7SF7D7602V2*. Feb. 2014. URL: https://fccid.io/K7SF7D7602V2.

[25] fccid. *Nest Labs Inc Wireless Protect S30*. June 2014. URL: https://fccid.io/ZQAS30.

[26] fccid. *Shenzhen Sonoff Technologies Co., Ltd. Temperature and Humidity Sensor SNZB-02*. May 2020. URL: https://fccid.io/2APN5SNZB-02.

[27] ghostop14. *ghostop14/sparrow-wifi: Next-Gen GUI-based WiFi and Bluetooth Analyzer for Linux*. URL: https://github.com/ghostop14/sparrow-wifi.

[28] Deock-Ho Ha. "Improvements of Call-drop Rate of CDMA Systems by Removing Spurious Emissions from UHF TV Broadcast Band". In: *Journal of Korean Institute of Information Technology* 14.7 (Oct. 2016), p. 63. DOI: 10.14801/jkiit.2016.14.7.63.

[29] J.C. Haartsen. "The Bluetooth radio system". In: *IEEE Personal Communications* 7.1 (2000), pp. 28–36. DOI: 10.1109/98.824570.

[30] *How to sniff Zigbee traffic*. URL: https://www.zigbee2mqtt.io/how_tos/how_to_sniff_zigbee_traffic.html.

[31] HP. *HP Study Reveals 70 Percent of Internet of Things Devices ...* July 2014. URL: https://www.hp.com/us-en/hp-news/press-release.html?id=1744676.

[32] Xianhe Huang et al. "100-MHz Low-Phase-Noise Microprocessor Temperature-Compensated Crystal Oscillator". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62.7 (2015), pp. 636–640. DOI: 10.1109/TCSII.2015.2415652.

[33] Intel. *Intel® Movidius™ Neural Compute Stick*. Sept. 2018. URL: https://software.intel.com/content/www/us/en/develop/articles/intel-movidius-neural-compute-stick.html.

[34] Minhaj Ahmad Khan and Khaled Salah. "IoT security: Review, blockchain solutions, and open challenges". In: *Future Generation Computer Systems* 82 (2018), pp. 395–411. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2017.11.022. URL: https://www.sciencedirect.com/science/article/pii/S0167739X17315765.

[35] Koenkk. *Koenkk/zigbee2mqtt: Zigbee to MQTT bridge , get rid of your proprietary Zigbee bridges*. URL: https://github.com/Koenkk/zigbee2mqtt.

[36] In Lee and Kyoochun Lee. "The Internet of Things (IoT): Applications, investments, and challenges for enterprises". In: *Business Horizons* 58.4 (2015), pp. 431–440. ISSN: 0007-6813. DOI: https://doi.org/10.1016/j.bushor.2015.03.008. URL: https://www.sciencedirect.com/science/article/pii/S0007681315000373.

[37] Avijit Mallik. "Man-In-The-Middle-Attack: Understanding In Simple Words". In: *Cyberspace: Jurnal Pendidikan Teknologi Informasi* 2.2 (2019), p. 109. DOI: 10.22373/cj.v2i2.3453.

[38] NESDIS. *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*. Nov. 2001. URL: https://www.nesdis.noaa.gov/CRSRA/files/AES%20fips-197.pdf.

[39] Aviv Nevo, John L. Turner, and Jonathan W. Williams. "Usage-Based Pricing and Demand for Residential Broadband". In: *Econometrica* 84.2 (2016), pp. 411–443. DOI: https://doi.org/10.3982/ECTA11927. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3982/ECTA11927. URL: https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA11927.

[40] *Nooelec NESDR SMArt v4 SDR - Premium RTL-SDR w/ Aluminum Enclosure, 0.5PPM TCXO, SMA Input. RTL2832U R820T2-Based*. URL: https://www.nooelec.com/store/sdr/sdr-receivers/nesdr/nesdr-smart-sdr.html.

[41] Electronics Notes. *RF Attenuators: basics, types, symbols*. Aug. 2021. URL: https://www.electronics-notes.com/articles/radio/rf-attenuators/what-is-an-rf-attenuator-types.php.

[42] Freddy Odille et al. "Noise Cancellation Signal Processing Method and Computer System for Improved Real-Time Electrocardiogram Artifact Correction During MRI Data Acquisition". In: *IEEE Transactions on Biomedical Engineering* 54.4 (2007), pp. 630–640. DOI: 10.1109/TBME.2006.889174.

[43] Darren Pauli. *IoT worm can hack Philips Hue lightbulbs, spread across cities*. Nov. 2016. URL: https://www.theregister.com/2016/11/10/iot_worm_can_hack_philips_hue_lightbulbs_spread_across_cities/.

[44] Tutorials Point. Apr. 2014. URL: https://www.tutorialspoint.com/wireless_security/wireless_security_rf_monitoring_tools.htm.

[45] punchthrough punchthrough punchthrough. *Harmonics Part 1 - Introduction to Harmonics BLE*. Mar. 2020. URL: https://punchthrough.com/harmonics-part-1-introduction-to-harmonics-ble-2/.

[46] Oona Räisänen(Oona Räisänen). *Descrambling split-band voice inversion with deinvert*. Sept. 2017. URL: http://www.windytan.com/2017/09/descrambling-split-band-voice-inversion.html.

[47] *RF Wireless World*. Aug. 2021. URL: https://www.rfwireless-world.com/Terminology/spurious-vs-harmonics.html.

[48] Mohammad Zakir Hossain Sarker and Md. Shafiul Parvez. "A Cost Effective Symmetric Key Cryptographic Algorithm for Small Amount of Data". In: *2005 Pakistan Section Multitopic Conference* (May 2005). DOI: 10.1109/inmic.2005.334435.

[49] Kiran Jot Singh and Divneet Singh Kapoor. "Create Your Own Internet of Things: A survey of IoT platforms." In: *IEEE Consumer Electronics Magazine* 6.2 (2017), pp. 57–68. DOI: 10.1109/MCE.2016.2640718.

[50] *Solving Urban Infrastructure Problems Using Smart City Technologies*. Elsevier.

[51] WIRED Staff. *Jamming Tags Block RFID Scanners*. Mar. 2004. URL: https://www.wired.com/2004/03/jamming-tags-block-rfid-scanners/.

[52] *SUCOFLEX® 104 Microwave Cable Assemblies*. July 2019. URL: https://eu.mouser.com/new/huber-suhner/huber-suhner-sucoflex-104-microwave-cable-assemblies/.

[53] ShareBrained Technology. *PortaPack for HackRF One, Kit*. URL: https://store.sharebrained.com/products/portapack-for-hackrf-one-kit.

[54] VOCAL. *VOCAL Technologies*. Aug. 2021. URL: https://www.vocal.com/secure-communication/frequency-scrambler-frequency-inversion/.

[55] T. Watteyne. "4 - Lower-power wireless mesh networks for machine-to-machine communications using the IEEE802.15.4 standard". In: *Machine-to-machine (M2M) Communications*. Ed. by Carles Antón-Haro and Mischa Dohler. Oxford: Woodhead Publishing, 2015, pp. 63–77. ISBN: 978-1-78242-102-3. DOI: https://doi.org/10.1016/B978-1-78242-102-3.00004-6. URL: https://www.sciencedirect.com/science/article/pii/B9781782421023000046.

[56] Windytan. *windytan/deinvert: A voice inversion descrambler (and scrambler)*. URL: https://github.com/windytan/deinvert.