CONTENTS

LIST OF ABBREVIATIONS

AM	Amplitude modulation
ASK	Amplitude shift keying
ERP	Effective radiated power
FM	Frequency modulation
FSK	Frequency shift keying
I/Q	In-phase and quadrature
ISM	Industrial, scientific and medical
ITU	International Telecommunications Union
NFC	Near field communication
OOK	On-off keying
PCB	Printed circuit board
PCM	Pulse code modulation
PIR	Passive infrared
PKES	Passive keyless entry and start system
PM	Phase modulation
PPM	Pulse position modulation
PSK	Phase shift keying
PWM	Pulse-width modulation
RFID	Radio frequency identification
RSSI	Received signal strength indicator
SDR	Software-defined radio
SNR	Signal to noise ratio
Sub-GHz	Sub-gigahertz
SWR	Standing wave ratio
TPMS	Tire pressure monitoring system
UHF	Ultra high frequency
VHF	Very high frequency
VNA	Vector network analyzer
VSWR	Voltage standing wave ratio

1 INTRODUCTION

Many common devices are communicating on sub-GHz frequencies using power efficient microcontrollers. These frequencies are less susceptible to interference and have better range than ones operating on gigahertz frequencies. Frequently used frequency bands are the industrial, scientific and medical (ISM) bands 433.9 and 915.0 MHz, which in many countries are allocated for license free use. (Microchip 2024.) Devices that use these frequencies include weather stations, locking systems, meters measuring utilities, such as power and water consumption, and tire pressure monitoring systems (Rtl_433 project 2024).

Sub-GHz is often forgotten in wireless cybersecurity as the devices do not directly work using the internet protocol. However, in some cases they can transmit data to systems that communicate it further using the internet protocol. Vulnerabilities of these radio communications are local to the reach of the radio transceiver on the device. Common concerns are bypassing locking and alarm systems that have led to some governments banning easy to use sub-GHz hacking devices from entering these countries (Riches 2024).

In this work, the goal is to find affordable ways of analyzing traffic from devices operating on sub-GHz frequencies to identify common vulnerabilities. For example, vulnerabilities such as different replay attacks, reading unencrypted traffic and signal jamming will be investigated. The results of this work will be replicable using equivalent software, hardware and methods.

2 RESEARCH DESIGN

The research will focus on finding ways to analyze traffic originating from devices operating on sub-GHz frequency bands to help identify attacks and vulnerabilities. However, the work will not offer solutions for mitigating these issues, instead, it focuses on finding methods for identifying issues and leaving any mitigating action to anyone using these methods.

One of the main requirements for the work is to use affordable equipment to do the analysis as specialized radio equipment can be expensive. Where viable, multiple options for hardware and software will be introduced depending on the use case.

Devices operating using sub-GHz frequencies are often forgotten in cybersecurity even though vulnerabilities exist. Sub-GHz hacking has been covered in the media because of stolen vehicles and some governments globally have been concerned about easy-to-use hacking devices (Riches 2024).

2.1 Research questions

The work will answer the following questions:

- What kind of devices are communicating on sub-GHz frequencies?
- What types of attacks and vulnerabilities sub-GHz devices are affected by?
- How to tell if a device is susceptible to these types of attacks and vulnerabilities?
- What equipment and software are needed for analyzing these devices?

Answering these questions will give a basic understanding of how to generally analyze devices operating on sub-GHz frequencies.

2.2 Research resources and analysis

Finding ways to analyze for vulnerabilities on devices that use radio frequencies from the sub-GHz bands is the goal for the work. This limits the subject to the sub-GHz bands. Case study as a research method is utilized as it can be used to investigate a specific part of a larger phenomenon. It is usually used for studying a specific subject from a group of subjects such as an organization, a group or a process. (Kallinen & Kinnunen 2021.) In this work, the case is the process of analyzing traffic on sub-GHz frequency bands to help identify attacks and vulnerabilities on devices.

Constructive approach is derived from the case study method focusing on real world problem solving by creating a new construct to solve a problem. One of the key points is experimentation that results in a new construct to challenge existing methods or replace them with a brand new one. Problem solving revolves around the analysis of what does and does not work. This method is used as a theoretical contribution through demonstrating already existing theories when creating the construct. The focus can be improving processes within an organization, for example. (Lukka 2001.) Since the goal is to find processes to analyze for vulnerabilities by experimenting with affordable hardware and software, this approach is helpful for what the work is trying to achieve.

In the constructive approach, knowledge about the topic must be gathered theoretically and practically to get a starting point (Lukka 2001). The research for this work begins by getting a basic understanding of radio communications as it is needed to understand how the signaling works and how to best capture data. Written media from different internet resources and books will be used to learn about radio communications. Researching documentation will be utilized to find information about software and hardware for identifying the best match for the use cases. Documentation will help understand how to operate the equipment and software. The hardware needs to be supported by the software, which means making sure that support is included in the software before acquiring the hardware. Another useful resource is videos demonstrating tools and hands-on skills on websites like YouTube. Hands-on skills that will be researched are operating the acquired hardware and software, and creating antennas optimized for specific use cases.

The hardware must be capable of performing consistently with enough coverage. Having equipment that yields consistent readings will be the key to capture reliable data. Depending on the hardware used, external antennas might be used instead of built in ones. Testing antenna configurations and building antennas optimized for specific frequencies will require some experimenting but the performance will be confirmed by logging values using measuring tools. Tuning the performance will enable reliable data capture.

Observation will be used as a research method to understand the type of data being transmitted and what device it originates from. This means observing the traffic on sub-GHz frequencies and learning the tools to capture traffic. For analyzing the captured data some basic reverse engineering will be used. This will require an understanding of different radio modulation techniques commonly used by manufacturers to encode data as radio signals. This comes back to using materials such as books and websites as resources to understand encoding methods. Additionally, decoding the data requires knowledge of the tools and will require experimentation since manufacturers use different protocols and they might not be well documented. Household devices will be used for testing as there are many devices using sub-GHz frequencies for communication. For example, devices like doorbells, motion detectors and locks can be researched.

After gathering enough material, the next step is to create a construct that solves the problem. If the construct cannot be created at this point, the project cannot continue. This does not necessarily mean that the work does not have any value but will instead be looked at from the viewpoint of why the process failed and what can be done to overcome this failure next time. If there is enough information to implement a solution, the work moves to the implementation phase. (Lukka 2001.) This phase will be creating and implementing the processes for identifying the researched vulnerabilities using the selected hardware. To make sure the process is functional, multiple devices will be analyzed to confirm that the process is repeatable. Another way is to repeat the same analysis using different software or hardware to see if the results are the same or similar.

Once the implementation and tests are done, it is time to reflect on the results. The first thing that will be looked at is whether the research questions were answered and requirements met. If successful, this moves the work towards the goal. Even if the tests end up in a failure, it is important to analyze what went wrong as this will be helpful for future projects and developments. (Lukka 2001.)

The thesis will be closed by reflecting on future developments that can be derived from the work. Interesting and worthwhile ideas for future projects will be brought up, including ones that cannot fit into the scope or timeframe.

2.3 Research ethics and legal considerations

When performing research and testing on radio and data transmissions, it is important to comply with the local laws. In the criminal code of Finland on information and communication crimes, chapter 38, it is prescribed that interfering with communications and information systems using, for example, a radio device, is illegal and punishable by law. Data breach, when done without a right, using a specialized technical device or technical method to a device that processes, stores and transmits data is illegal. (Rikoslaki 19.12.1889/39.) It is important that any testing will not disrupt communications accidentally as this is considered illegal. Traficom radio frequency regulation 4 AE/2024M must be followed to ensure transmitting is allowed without a license. (Liikenne- ja viestintäviraston radiotaajuusmääräys 4 AE/2024M). The frequencies allowing operation without a license will be investigated in more detail when researching how the sub-GHz bands are regulated in Finland.

In chapter 12 of the criminal code, possession of burglary tools is prescribed illegal in cases where it can be reasonably suspected that the tools are mainly used for accessing a closed space that is not under the person’s control to perform a criminal act (Rikoslaki 19.12.1889/39). This means that the possession of tools that can be used for bypassing smart locks, for example, is not illegal when used for testing your own devices or when permission from the owner has been received to perform testing.

Any device that is in production or in active use will not be manipulated without considering the risks first. There are always risks such as damage to the receiver or desynchronization of the devices when using unsupported transmitters. A mitigation plan must be in place when any production equipment is tested by

manipulating them to ensure recovery. The easiest way to avoid this is to test the equipment before it is in production or acquiring duplicates for testing purposes. This will not completely mitigate monetary risks since the equipment can break.

3 THEORETICAL FRAMEWORK

A starting point for the research will be built by gathering theory, which will be complemented by practical testing later. The goal is to acquire an understanding of sub-GHz frequencies, their regulation in Finland and how these radio communications generally work. In addition, background about sub-GHz cybersecurity and research will be investigated.

3.1 Sub-GHz

Sub-GHz devices often operate on industrial, scientific and medical (ISM) bands that have been agreed to in International Telecommunications Union (ITU) radio regulations. ITU is an agency specializing in information and communication technologies under the United Nations and the ITU Member states are bound to follow the regulations. ITU radiocommunication regulation 5.138 provisions the 433.05 to 434.79 MHz band with the center frequency 433.92 MHz in region 1 that includes Europe, Africa and Russia. Region 2, which includes The Americas, is covered by the ITU radiocommunication regulation 5.150 that provisions the 902 to 928 MHz band with the center frequency of 915 MHz for ISM. However, there are exceptions in some countries for these allocations. It is stated that any operation on ISM bands must be able to withstand harmful interference. (ITU 2015, 26.) However, the ITU radiocommunication regulation 15.13 states that administrations must take all necessary and practicable steps in ensuring that radiation from ISM applications is kept minimal (ITU 2017). In Finland, 433.05 to 434.79 MHz band is regulated by Traficom to have maximum effective radiated power (ERP) of exactly or below 25 milliwatts (mW). It is intended for non-specific short-range devices such as alarms, telemetry and equipment control among other use cases. The 915 to 919.40 MHz band must have a maximum ERP of exactly or below 25 mW and is intended for data network usage. It is smaller than the 915 MHz band in region 2 as some of the full band is used for other

applications. The 865 MHz to 868 MHz band is not considered ISM band, however, it has been allocated for short range radio transmitters and radio frequency identification (RFID) applications. Operating on these bands does not require a license in Finland. The 315 MHz and 345 MHz bands are reserved for military use in Finland. There are other sub-GHz bands allocated for different commercial uses in Finland, and they are listed in the radio frequency regulation if information about a specific one is required. (Liikenne- ja viestintäviraston radiotaajuusmääräys 4 AE/2024M.) Sub-GHz bands fall under the ultra high frequency (UHF) band that covers 300 MHz to 3 GHz (Britannica 2024a). Very high frequency (VHF) band is 30 MHz to 300 MHz that is typically used for commercial radio broadcasts and amateur radio, for example (Britannica 2024b).

Using Sub-GHz frequencies for devices has other benefits than license free operation on ISM bands. Lower frequencies have lower attenuation from obstacles and the angle of diffraction in which the radio waves bend from collision with obstacles is higher. This means lower frequency radio waves have easier time traveling further even if blocked by obstacles. The ISM bands have less interference compared to the popular 2.4 GHz band because it is used usually for transmitting short bursts of data. The 2.4 GHz band is more crowded due to wide use in WiFi and Bluetooth applications where data transfers are more substantial and frequent. (Farnell element14 2018.)

LoRaWAN is a specification for low powered wide area network applications operating on the license free bands. It is typically used for Internet of Things applications such as reading utilities wirelessly. (LoRa Alliance 2024.) This specification will not be covered since it requires a LoRaWAN network and devices to study and analyze.

3.2 Software-defined radio

Software-defined radio (SDR) is technology used in radio communications to enable flexibility and digital signal processing capabilities. Software is used to change how radio hardware functions, which offers many operation modes for a

single piece of radio hardware. SDRs use analog-to-digital converters (ADC) to change the analog radio signal into something a digital signal processing (DSP) software can use. (Youngblood 2002.) One implementation of SDR is the RTL-SDR USB dongle. It is an effort to support the use of Realtek RTL2832U based DVB-T dongles for raw transfer of in-phase and quadrature (I/Q) samples to a host PC for software to utilize (Osmocom 2024). This chip is only capable of receiving as it is originally designed as a DVB-T television signal demodulator that in addition supports FM, DAB and DAB+ commercial radio technologies (Realtek 2024). The RTL-SDR project utilizes the commercial radio support of the demodulator for signal receiving (Osmocom 2024).

Figure 1. RTL-SDR Blog branded software-defined radio dongle.

The RTL-SDR driver is developed by Osmocom while RTL-SDR Blog develops and distributes hardware, demonstrated in Figure 1, specifically optimized for the driver. There are different SDR implementations using other hardware and drivers that have better performance and come with more features, but RTL-SDR is one of the best when it comes to value for money. (RTL-SDR Blog 2024.) On top of the hardware and driver, a software for processing and decoding the traffic is needed. There is software for different types of use cases but the more popular ones currently for general use are Airspy SDR#, SDR++ and SDRangel.

3.3 Research and motivations

Multiple governments have been concerned about easily obtainable sub-GHz hacking tools that are possibly used for criminal activities. For example, a pocket-sized radio frequency hacking tool called Flipper Zero is capable of replaying signals sent by keyless entry systems, garage doors and sensors. (Mehrotra 2022.) This device was banned by the Government of Canada due to its alleged role in the stolen car crisis in the country. Flipper Devices has challenged this decision by saying that it does not have the capability to bypass modern security systems of vehicles manufactured after the 1990s. (Riches 2024.) The New Jersey Cybersecurity and Communications Integration Cell stated that most modern wireless devices are not vulnerable to these types of simple replay attacks that the Flipper Zero can produce even though many videos on TikTok and other social media platforms show otherwise. They suspect that most of these videos are misinformation and have been staged. (New Jersey Cybersecurity and Communications Integration Cell 2023.) In Brazil, the Flipper Zero was not certified by the national telecommunications regulator Anatel as they were concerned that it can be used for illegal activities. This means the devices shipped to Brazil are seized by the national post office. (Buddington & Alimonti 2023.)

There have not been many master's or bachelor's theses covering sub-GHz radiocommunications and cybersecurity. For example, in their bachelor's thesis Smart home systems' security and networking, Karvonen covers some sensors that use 433 megahertz technologies but mostly focuses on the user facing software and its internet traffic (Karvonen 2018).

In the 2/2013 issue of Skrolli, Räisänen demonstrates using RTL-SDR for decoding unencrypted signals from a wireless keyboard, Logitech iTouch, manufactured around the year 2000. The keyboard communicates on 27.140 MHz frequency and a radio dongle using the RTL2838U demodulator supports being tuned into the frequency using SDR software. Räisänen demonstrates

recording and saving the traffic from the keyboard into a WAV file for analysis. In the article they show that each press of the same key produces a similar bit string every time, which indicates that there is no encryption. This makes it possible to tie keypresses to specific keys. It means that even if the attacker does not have access to the same keyboard, they can use frequency analysis to compare the frequency of captured bit strings with the frequency of letters in words of a given language. Since the messages for specific key presses are similar, but not identical, the data can be averaged and made into a list of keys. Then a piece of software can be written to match them to the closest key. (Räisänen 2013, 12.)

3.4 Common vulnerabilities

Replay attack is when a signal for a code is captured by an attacker and replayed to perform an action using the same code (National Institute of Standards and Technology 2024). Rolling codes are often used to try to mitigate the risk of replay attacks by having a code that changes based on an algorithm that the receiving end knows, and the remote is synchronized to (Microchip 2001). If the remote is used outside of the range of the receiver there is a risk of desynchronization. This is why some simple implementations allow a range of codes to be used at a given time. It means that the receiver might accept the same code several times to reduce the risk of desynchronization. (Brain 2001.)

A version of the replay attack called jam and replay attack is used against vehicle locks and garage doors, for example. The attack happens by jamming the receiver while simultaneously the signal from the transmitter is captured. The code never reaches the receiver, and it does not know it has already been used. The attacker is then able to perform an action using the captured code that is still valid from the receiver’s point of view. Since in real world scenarios the user usually notices that the first press did not work and tries to press the button again, the attacker captures the second code while jamming the receiver again and then transmits the first code. At this point the attacker has stored a fresh code, and the user is happy that their button press worked. This type of attack can be performed using a full-duplex device as they can transmit the jamming

signal and capture the transmission at the same time. The software can be programmed to immediately transmit the first code when the jamming signal stops. The captured code only expires after a fresh code is transmitted to the receiver. (Greenberg 2015.)

Many modern cars have been installed with a passive keyless entry and start system (PKES) for operating the vehicle locking mechanism. The vehicle key needs to be in a certain operating proximity to the vehicle for the lock and ignition to operate. Relay attacks have been used to bypass this type of locking mechanism. Two attackers are needed for this type of exploit because they must relay messages between the key fob and vehicle. When there is an attempt to unlock the lock, the vehicle sends a challenge to the key fob and needs a key as response from the key fob. The first attacker relays the challenge to the second attacker who is near the key fob, and they then relay it to the key fob. The key fob then sends the response that is relayed by the second attacker to the other attacker who relays that to the vehicle. This allows the first attacker to open the car and start the ignition. (Jeong & So 2018.)

3.5 Flipper Zero

Flipper Zero is a popular tool used for hacking and penetration testing sub-GHz devices. It is known for its small size and ease of use. (Flipper Devices 2024a.)

Figure 2. Flipper Zero plugged in using a USB-C cable.

Flipper Zero, seen in Figure 2, has directional buttons and a separate back button for navigating its menus. An additional silicon cover can be attached to the device to protect the back and sides of the device. It is battery powered and charged using a USB-C charger. (Flipper Devices 2024a.)

It supports sub-GHz radio signals, 125 kilohertz (kHz) radio frequency identification (RFID), near field communication (NFC) at 13.56 MHz, Bluetooth and infrared. The firmware of the device is fully open-source and customizable. The module used for sub-GHz support is CC1101 transceiver that supports the frequency range of 300 to 928 MHz. The tool includes an antenna with a maximum range of 50 meters that is specifically tuned for operation at 300-348, 387-464 and 779-928 MHz bands. It supports amplitude modulation with bandwidth of 270 (AM270) and 650 (AM650) kHz. In addition, it supports frequency modulation with the bandwidth of 270 kHz and the deviation of 2.380371 (FM238) or 47.60742 (FM476) kHz. It is capable of reading, saving and emulating some protocols for remote controls as well as capturing any signal in RAW format. RAW captures are used for unknown and unsupported protocols with the caveat of having to set the correct modulation to correctly demodulate the signal. It has a frequency analyzer to help identify frequencies of received signals, which is useful when the frequency of a device is unknown. Another

helpful feature is frequency hopping that enables the user to hop through available frequencies in the sub-GHz read mode. (Flipper Devices 2024a.)

Managing Flipper Zero is done using a desktop application called qFlipper that is available for Windows, MacOS and Linux. For Linux there is an AppImage available. (Flipper Devices 2024c.) AppImages on Linux operating systems run without installing additional dependencies as they contain all of them within the image. If needed, the application can be built from the source code that is currently available on GitHub. One of the main uses for qFlipper is that it is the official way for updating the firmware of Flipper Zero. The device is currently receiving constant firmware updates. (Flipper Devices 2024c.) In addition, there are multiple custom firmware being actively developed with additional functionality through application and configuration changes. The most popular custom firmware often removes any regional limits for the transceiver allowing the user to transmit on restricted bands. The qFlipper application is used for managing files in the internal memory and SD card (Flipper Devices 2024c). This is useful for editing and analyzing captured signals. Using the qFlipper user interface it is possible to rename, download and delete files from the device (Flipper Devices 2024c). Downloading files such as saved captures of radio communications enable the user to view and edit these files on a PC using a text editor. It is utilized for modifying parameters or decoding RAW data into bit strings. The software makes it possible to control and view the screen of the device from the PC desktop (Flipper Devices 2024c). It is especially useful for taking screenshots and testing things after modifying files without having to touch the device.

Flipper Devices is hosting a web page called Flipper Lab where users can submit applications, they have written for Flipper Zero to add more functionality. Applications from the Flipper Lab are installed using a modern browser with the device plugged into a USB port of a PC. Approval for the browser to communicate with the device is required. After the device is connected, applications are installed using the install button on the page for the application. Additionally, the connection allows browsing files and accessing the CLI on the

web page. Tools for NFC, pulse plotting and pixel art are available on the web page. (Flipper Devices 2024d.)

3.6 Signal modulation

Modulation in radio technology refers to creating a modulated signal by changing the characteristics of a carrier signal based on an input signal. The transmitting side performs the modulation, and the signal is then demodulated in the receiving end. Typical analog modulations used for radio communications are amplitude, frequency and phase modulation. In amplitude modulation (AM) the amplitude of the carrier signal changes based on the input signal. The carrier signal frequency stays the same. In frequency modulation (FM) it is the opposite. The amplitude does not change, and the frequency of the carrier signal is changed based on the input signal. Phase (PM) modulation is like FM in that there is no need to change the amplitude, but instead the phase of the carrier signal is changed. (Faruque 2017, 1-4)

Common digital modulation types are amplitude, frequency and phase shift keying. In amplitude shift keying (ASK) a binary signal is presented as changes in amplitude of the carrier signal. A low amplitude represents a bit value of 0 while high amplitude represents a bit value of 1. (Faruque 2017, 5-7) A type of ASK modulation where the signal power of infinite represents the 0 state is called on-off keying (OOK). This method is most prone to interference due to the reliance of the signal being on or off. When performing the demodulation some signal can be present due to noise when instead the signal is ideally nonexistent. This can be worked around but its effectiveness depends on the method used. (McCune 2010, 104, 115.) In frequency shift keying (FSK) the carrier signal has two different frequencies that represent the bit values of 0 and 1 that are modulated based on the input binary signal. Phase shift keying (PSK) uses manipulation of the phase of the carrier signal to produce a signal that represents the binary digital signal. Phases of 0 and 180 degrees are used for the different binary values. Bit value of 0 is a carrier signal with a phase of 0 degrees while for the bit

value of 1 the phase of the carrier signal is changed by 180 degrees. (Faruque 2017, 5-7)

Pulse modulation is a technique where the amplitude, duration and position of a pulse within a signal is modulated to encode data (Cadence PCB solutions 2023). Pulse-width modulation (PWM) is a method to encode digital data using analog signal levels by supplying on and off pulses of varying width. Since the signal always remains digital, it is less prone to noise and does not need analog to digital conversion, which makes the circuit simpler. The term duty cycle is used to describe the percentage of time the signal stays on within a period. The length of a period is determined by the frequency. (Barr 2001, 103-104.) Pulse position modulation (PPM) is when a fixed-width pulse changes positions within a period to encode data. The positions relative to the reference point within the period are predetermined. This technique is used for its reliability due to resilience against noise and interference. (Cadence PCB solutions 2023.)

3.7 Antennas

Antennas are transducers; devices that convert energy from one form to another. In the case of antennas, guided waves are converted into radiated waves when transmitting and the opposite when receiving signals. (Cisco 2007.) Generally, antennas are reciprocal meaning they have the same characteristics when receiving and transmitting (Milligan 2005, 1-2). Typical antennas have a coaxial connector with a coaxial cable attached that is used to transmit energy between the transceiver and the antenna (Cisco 2007).

Radiation patterns are used to graphically present properties of an antenna as it radiates energy into space. Antennas radiate energy into every direction, so a three-dimensional pattern is often presented as principal plane patterns by taking two slices of the three-dimensional pattern at maximum value. These are usually known as the azimuth plane pattern and elevation plane pattern. Azimuth refers to the horizontal pattern (x-y) while the elevation is the vertical pattern (y-z, x-z). Most common way to visualize these patterns is using polar coordinates that help

visualize the radiation pattern when the antenna is mounted. Parts of the pattern that are the strongest in radiation, surrounded by weaker radiation, are known as lobes. Main lobe can often be thought of as the main area of radiation for the antenna but other lobes, side lobes and back lobe, can be important depending on the use case. (Cisco 2007.)

The isotropic radiator is used generally as a theoretical lossless radiator that radiates equal energy in all directions when defining gain. Gain is defined as the ratio of power gain in any given direction between the antenna and the reference antenna, generally being the isotropic radiator with the gain of 0 dB. The unit of dBi is used as the value to represent decibels relative to the isotropic radiator. Gain characterizes the radiated power in each direction, but the antenna itself does not create radiated power. Gain in dBi is presented using the following Formula 1. (Cisco 2007.)

$$G = 10\,Log_{10}(\frac{G_{num}}{G_{iso}}) \quad (1)$$

$$G_{iso} = 1$$

$$G = 10\,Log_{10}(\frac{G_{num}}{1}) = 10\,Log_{10}(G_{num})$$

where	G	gain	[dBi]
	G_{num}	gain as numeric value	[-]
	G_{iso}	gain of an isotropic radiator	[-]

Theoretical dipole is sometimes used as a reference and the values are presented as the unit of dBd that is decibels relative to a dipole (Cisco 2007).

Omnidirectional antennas, such as collinear and dipole antennas, have a circular pattern in an orthogonal plane while directional antennas have a directionally concentrated pattern. It is typical that a directional antenna has one main lobe or several smaller lobes. Yagi-Uda antenna is a common example of a directional antenna. (Cisco 2007.)

Standing wave is caused by an impedance mismatch when power is transmitted from a source to load, and power is reflected towards the transmitter. The voltage standing wave ratio (VSWR) is the ratio between maximum and minimum voltage of a standing wave pattern. It is a measurement of how much power is being transmitted to the load. VSWR of 1:1 means that all the power is transmitted, and no power is reflected. VSWR bandwidth is the range of frequencies that are within a specified VSWR on an antenna. (Cisco 2007.)

SWR of the antenna is ideally kept under 2:1 for UHF and VHF, otherwise with a coaxial cable the losses in power will be significant. If a coaxial RG-58 cable is used for VHF and UHF applications, it must be short enough that the cable attenuation is not affecting the performance of the transmission. This is because a higher SWR does not cause as much power loss if the power is not weakened when reflecting and forth within the cable before it escapes. (Ford 1994, 70-72.)

4 HARDWARE RESOURCES

Hardware that will be used for research includes Flipper Zero, RTL-SDR USB dongle, antennas and a personal computer (PC). Software used on the PC includes operating systems Fedora Linux 39 and Microsoft Windows 11 with RTL-SDR drivers installed. SDR software used will be SDR#, SDR++ and rtl_433.

RTL-SDR blog sells their RTL2832U dongles with optimizations for SDR use. Currently they offer versions 3 and 4 of the dongles but version 4 is not currently supported by all software, which is why version 3 will be used due to its better software support. The dongles are available from different online marketplaces, for example, Amazon and eBay. They are available as bundle costing around 80 euros, which includes the dongle, an antenna base with 60 cm RG174 coaxial cable with an SMA connector, 23 cm to 100 cm multipurpose dipole antennas, 5 cm to 13 cm multipurpose dipole antennas, a 3 m RG174 coaxial extension cable with SMA connectors, a flexible tripod mount and a suction cup mount. The

included antennas are best used for terrestrial and satellite reception. (RTL-SDR Blog 2024.) By itself, the dongle costs around 50 euros (Amazon 2024a).

For comparison another SDR dongle called LimeSDR 2.0 goes for around 700 euros on Amazon. The biggest difference with this dongle is that it can also transmit. (Amazon 2024b.) In addition, there is a device called HackRF One that is capable of transmitting. Its key differences are that it can transmit and is a larger device that supports standalone operation but can also be connected to a PC using a USB cable. It features headers for expanding its capabilities and costs around 400 euros on Amazon. (Amazon 2024c.)

The Flipper Zero currently costs 165 euros at the official store with some optional extras available at additional cost such as the silicone case at 15 euros and screen protectors at 7.50 euros. None of the additional modules or development boards are needed for typical sub-GHz usage. (Flipper Devices 2024a.) It is affordable when compared to SDRs that can transmit but comes with the caveat of not being as powerful and versatile. However, it compensates for being less powerful by being portable and easy to use.

NanoVNA is an affordable open-source vector network analyzer (VNA) specifically aimed at amateur radio hobbyists. It is battery powered and has a small footprint making it portable. It is used for measuring and calculating reflection loss, passing loss, complex impedance and SWR. (NanoVNA 2024.) It costs around 70 euros on Amazon and includes a calibration kit (Amazon 2024e).

5 SOFTWARE RESOURCES

The main PC used for testing will run Fedora Linux, which is an open-source operating system that uses the Linux kernel and is distributed for free. Its primary sponsor is Red Hat, who use the distribution as a basis for Red Hat Enterprise Linux development. (Fedora 2024a.) As of writing, the most current Fedora release version 39 has rtl-433 and rtl-sdr packages available in its software repositories (Fedora 2024b). Microsoft Windows 11, which is a commercial and

proprietary operating system, is used to test software that is not available on Linux operating systems.

Open-source software called rtl_433 will be used for decoding and logging sub-GHz traffic. It is a generic data receiver that works with RTL-SDR and is mainly designed for receiving the 315, 345, 433.92, 868 and 915 MHz frequency bands. It is possible to read data in JavaScript Object Notation (JSON) text format that can be piped or relayed into a database using scripts. (Rtl_433 project 2024.)

SDR++ is an open source SDR software that works on Windows, Linux and MacOS. It requires a graphics card with OpenGL 2.1 support to run in graphical mode but is more CPU rather than GPU intensive. Currently there are no official minimum requirements available. The software has ports for Android and ARM processor-based Raspberry Pi. For Raspberry Pi it requires at least Pi 3 but runs best from Pi 4 and up. It supports many SDR receivers such as RTL-SDR dongles, HackRF and Lime SDRs. The displays include a radio frequency spectrum and a waterfall with some adjustments available. Out of the box it includes modes for demodulation of narrow band frequency modulation (NFM), wideband frequency modulation (WFM), amplitude modulation, double sideband (DSB), upper sideband (USB), lower sideband (LSB) and continuous wave (CW). It is also capable of outputting a RAW signal. (Donkersley 2024.)

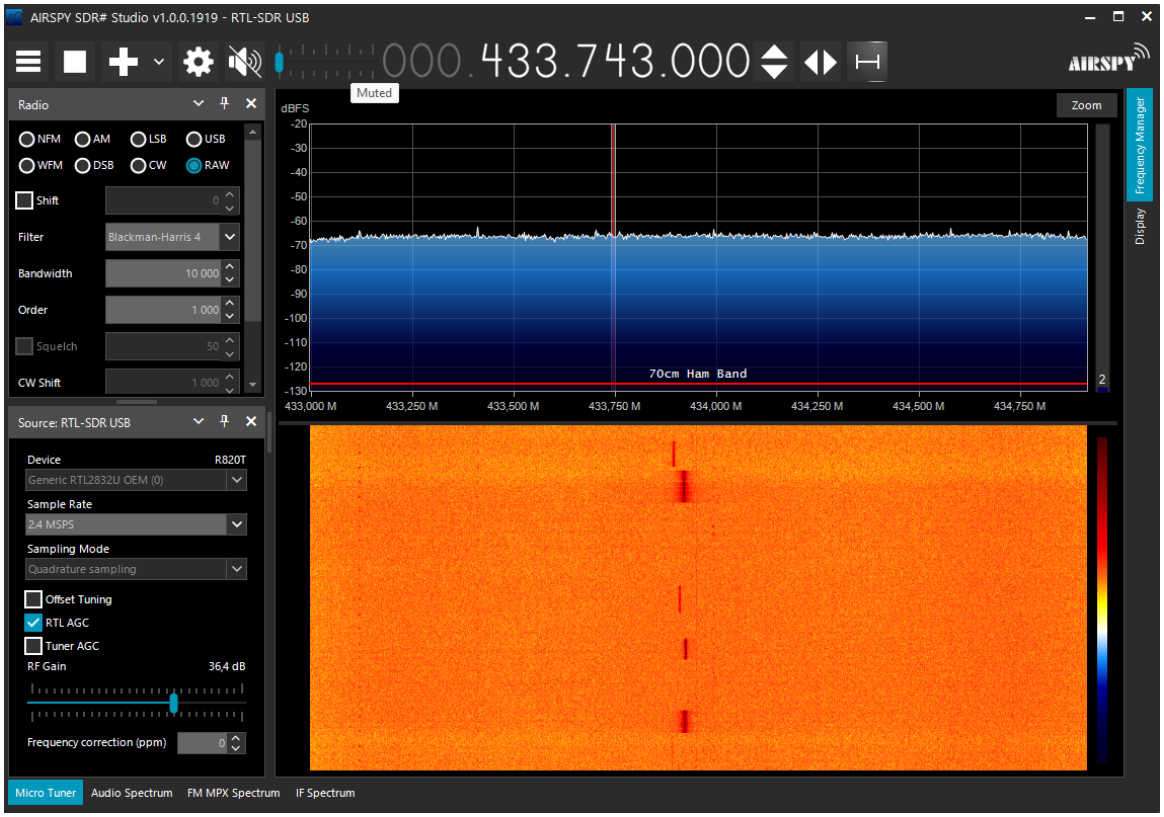


Figure 3. Transmission bursts on 433 MHz bandwidth seen on the waterfall graphic using Airspy SDR#.

SDR#, shown in Figure 3, is a very similar closed source software developed by Airspy available only on Windows operating systems. The biggest difference is that it has a larger library of community developed plugins for processing signals such as additional demodulation options and filters. (RTL-SDR Blog 2024.)

6 FLIPPER ZERO

Reading supported sub-GHz protocols and raw signals using Flipper Zero will be tested in practice. This includes operating the sub-GHz modes, understanding saved sub-GHz files and decoding raw signals into binary values.

6.1 Sub-GHz operation

To access sub-GHz features on the Flipper Zero, the Sub-GHz menu from the main menu was opened. The same features can be accessed from the Apps

menu as applications. This is where applications that have been installed are found in addition to the ones provided with the firmware. To start reading signals coming from a remote, for example, it is best to start with the frequency analyzer function to find out the frequency band the remote is transmitting on. This information is needed to set up the correct frequency band when reading incoming signals, otherwise the Flipper Zero will try to read signals from the wrong frequency band and will not capture any data or will capture unwanted data from another source. To test reading signals using this functionality, a battery-operated wireless doorbell was used for a simple implementation of sub-GHz communication between devices.

Figure 4. Generic wireless doorbell that operates at 433MHz and the Flipper Zero on top right.

Figure 4 shows the wireless doorbell and Flipper Zero used for this testing. The doorbell includes a battery powered wireless transmitter and receiver. The transmitter is triggered by pressing the button, which will make the receiver ring.

Figure 5. Inside view of the doorbell button.

Figure 5 is the inside view of the button where in the middle there is a component that is labeled surface acoustic wave (SAW) on the silkscreen of the PCB. This is a filter that lets specific frequencies pass through. It is done by generating an acoustic resonance from the electrical signal that causes some frequencies to be amplified and others to be suppressed. (ChipSun Technology 2021.) The component has HD R433M written on it that can indicate it is a 433 MHz part. To confirm if the frequency band the remote operates on is 433 MHz, the frequency analyzer was accessed. Flipper Zero was placed next to the doorbell remote and the button was pressed.

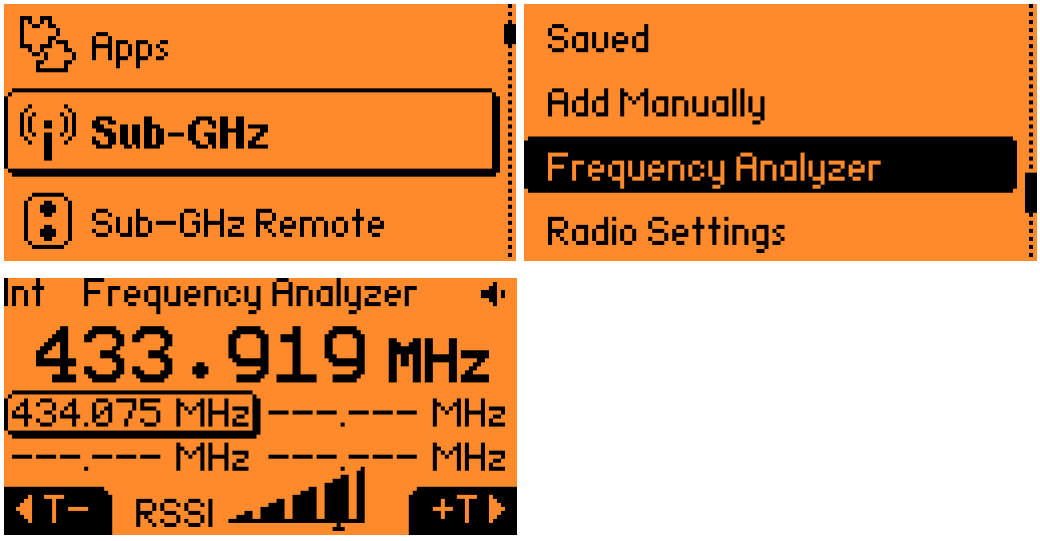


Figure 6. Accessing the frequency analyzer to detect frequency of a transmitter.

In Figure 6, it is shown that the frequency of 434.075 MHz was detected, and 433.919 MHz signal was also received from another source while the analyzer was active. The received signal strength indicator (RSSI) value, which is the strength of the signal received, was adjusted with the left and right buttons on the device to try to get rid of unwanted signals picked up from afar. If the remote is close to the Flipper Zero it is unlikely that a low RSSI value is needed unless it is sending a very weak signal. The back button was used to exit the frequency analyzer and get back to the Sub-GHz menu to next access the read functionality. Once in the read application the Flipper Zero immediately starts to scan on the configured frequency band. Whenever it receives a signal from a known protocol it will decode and record it on a list. The list can hold up to 55 decoded transmissions.

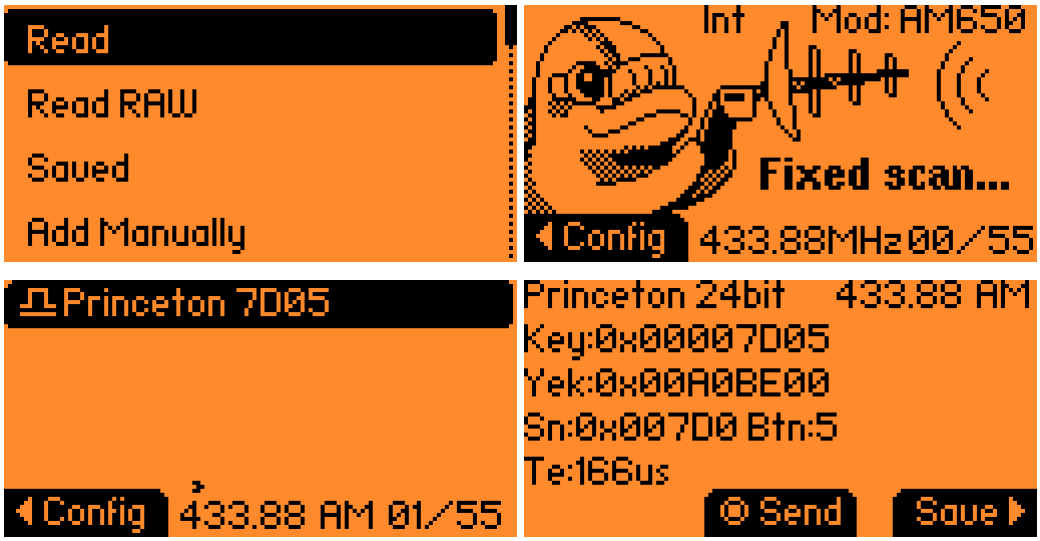


Figure 7. Accessing the Sub-GHz read functionality and reading a data transmission.

In Figure 7, the read functionality was accessed and scanning started on the 433.88 MHz band. The modulation used in this case was amplitude modulation with 650kHz bandwidth (AM650) indicated by the values on top right of the screen. To capture a key press from the wireless doorbell remote, the frequency analyzer revealed that the 433.88 MHz band is close enough with 650 kHz bandwidth. The modulation was unknown, but it can be configured from the config menu that is accessed using the left button. When the button on the remote was pressed Princeton 7D05 appeared on the list without having to adjust the configuration. More information about the captured signal was found using

the middle button on the device. The information indicates that the button is using the Princeton 24bit protocol on the 433.88 MHz band using amplitude modulation. Key is the actual data that was captured. The key that was received from the remote is 0x00007D05. It is presented as a hexadecimal value that is indicated by the preceding 0x. The key consists of 8 digits, which is less than the 24 bits indicated by the protocol since in hexadecimals each digit contains 4 bits. The key consists of the serial number 0x007D0, presumably to pair the receiver with the transmitter. The last hexadecimal value of 5 at the end is to indicate the button number. The serial number and button results in 0x007D05, which matches the 24-bit length of the protocol. Using the middle button on the Flipper Zero, the captured key was sent, and it made the doorbell ring just like when using the remote where the key was read. The right button is used for saving the key for later use with the possibility to rename it to something descriptive. Once saved it can be accessed through the Saved menu that lists all the saved files. Loading a saved key will allow emulating, renaming or removing it. Emulating the key will open the same view as the one on Figure 7 bottom right without the save option. All the information shown is sustained and the contents will stay the same as before saving. For analysis, the file had to be downloaded to a PC.

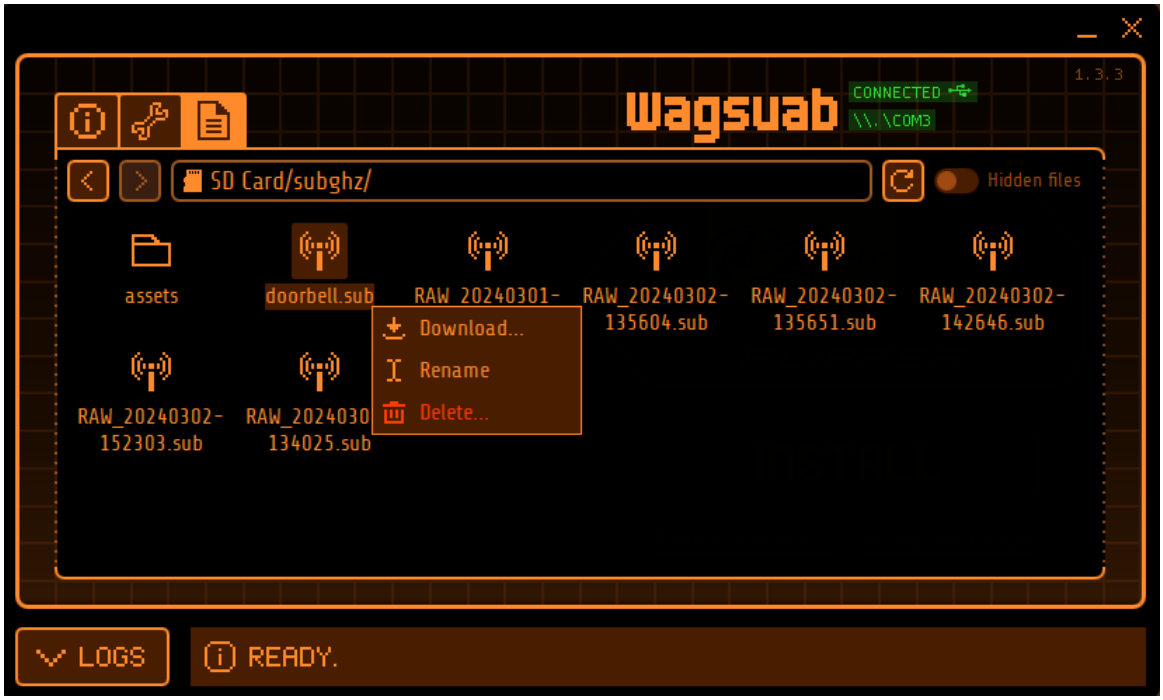


Figure 8. Accessing subghz directory on the Flipper Zero SD card using qFlipper application on a PC.

The file was accessed on a PC from the file browser of the qFlipper program as shown in Figure 8. By default, the Sub-GHz application will save files in the subghz directory on the SD card. Using the file browser the file can be renamed, deleted or downloaded from the device. Files saved from the Sub-GHz application will have the sub extension but can be opened as normal text files on Windows, Linux or MacOS. The contents of the saved doorbell.sub file was accessed:

```
Filetype: Flipper SubGhz Key File
Version: 1
Frequency: 433920000
Preset: FuriHalSubGhzPresetOok650Async
Protocol: Princeton
Bit: 24
Key: 00 00 00 00 00 00 7D 05
TE: 166
```

The saved file reveals the filetype and the preset that is used for the transceiver configuration. The default Flipper Zero firmware version 0.98.3 currently includes four types of built in presets. There are two types of On/Off keying (OOK) at 270kHz or 650kHz bandwidths and two types of 2 Frequency Shift Keying (2FSK) at 270kHz bandwidth with deviation of 2kHz or 47kHz. TE value is the quantization interval that is the length of a period. (Flipper Devices 2024b.) In this case FuriHalSubGhzPresetOok650Async is OOK with the 650kHz bandwidth.

As mentioned previously, the last hexadecimal value 5 is for identifying the button pressed. The file was modified and uploaded back onto the Flipper Zero. The last hexadecimal value was changed to 1, which shows up on the Flipper Zero as button 1:

```
Key: 00 00 00 00 00 00 7D 01
```

The file was uploaded by dragging and dropping it into the subghz directory on the qFlipper application. Next it was accessed on the device.

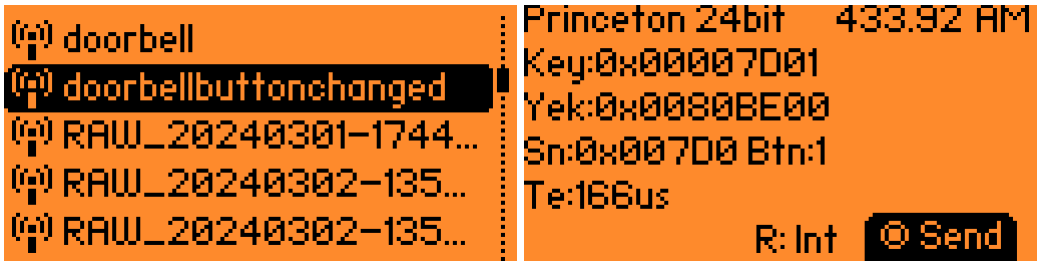


Figure 9. Accessing the modified file on the device after uploading.

In Figure 9, the modified and renamed file showed up in the Saved menu on the Flipper Zero Sub-GHz application where it was opened for emulation. The button value Btn changed to 1. Sending the modified button press made the doorbell ring. Testing through values from 0 through 6, the doorbell receiver triggered with any of these values. This showed that it does not enforce the button identification value.

6.2 Capturing and analyzing RAW signal

Signals can be captured using the Read RAW function as raw data when Flipper Zero is unable to decode the protocol. It can also be used for supported protocols when the raw signal needs to be analyzed, or a sequence of presses is required to be captured.

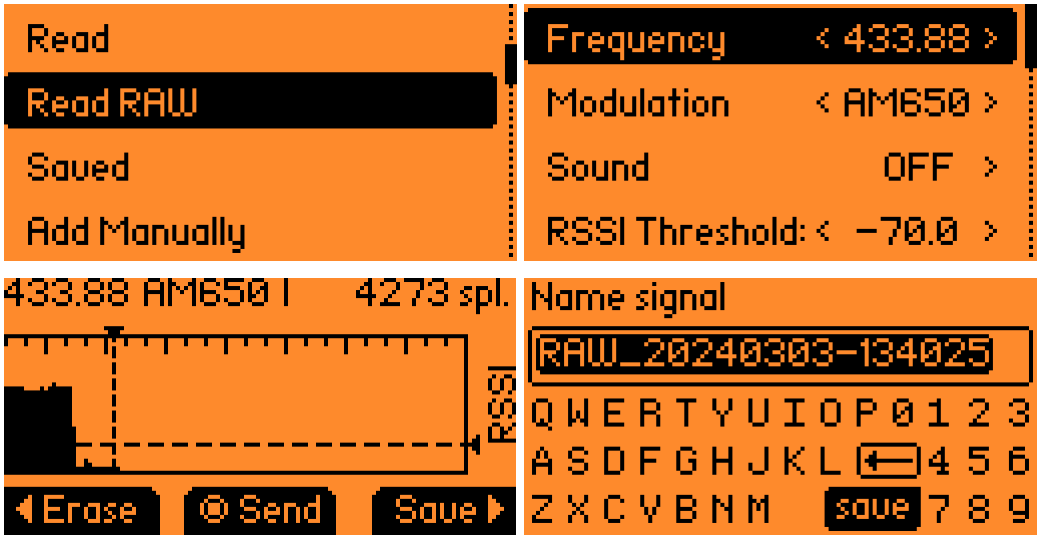


Figure 10. Read RAW function of the Sub-GHz Flipper Zero application.

This function is accessed from the Sub-GHz menu, shown on the top left of Figure 10. Once opened, an empty RSSI diagram and information about the mode on the top of the screen, such as the bandwidth and modulation, were visible. During a capture the diagram is filled based on the signal strength. This is shown in the bottom left of Figure 10. Settings for adjusting the frequency bandwidth and modulation were accessed using the left button before capturing any data. If data has been captured, it must be erased to access the settings. The menu is demonstrated in Figure 10 on the top right. The menu has settings for sound that will enable audible representation of the signal through the speaker. This is helpful when using the device without line of sight to the screen. RSSI threshold was set to make the application initiate capture only once a certain signal strength is reached. It should be set to at least above the average noise, if possible, to avoid having unnecessary data being captured. If the signal is further away or obstructed by walls, the threshold might have to be lowered or disabled. There is a small delay before the application stops recording after the signal falls under the threshold. Once a signal is captured, it can be resent using the middle button on the device and saved using the right arrow. The doorbell signal was captured again as a raw signal and saved into a file.

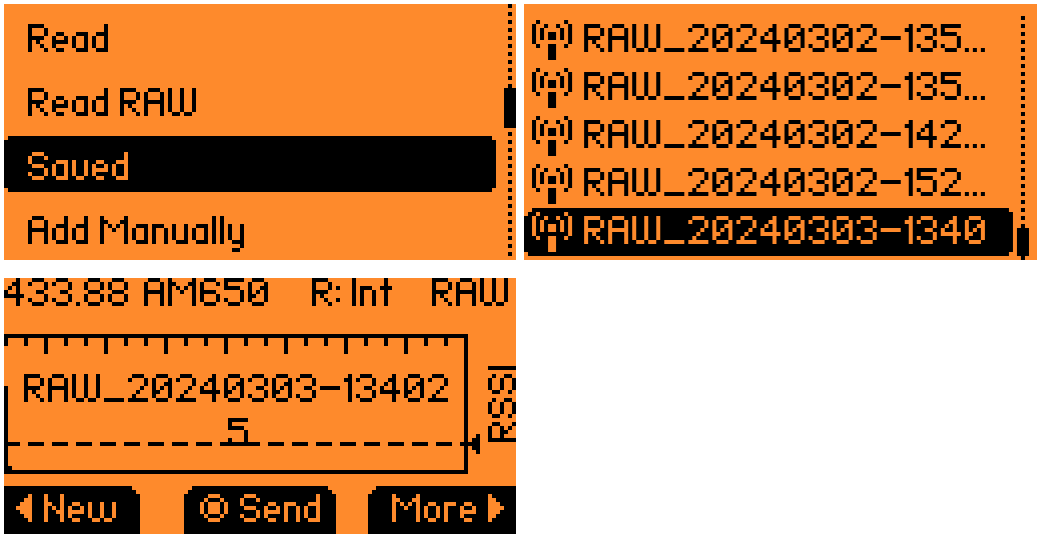


Figure 11. Accessing saved RAW files on Flipper Zero.

Figure 11 illustrates accessing the saved captures that show up in the same menu as ones captured using the Read function. If a signal is loaded from memory, it will not show the RSSI graph but instead shows the name of the file it was saved as. The saved RAW sub file was downloaded using the qFlipper application onto a PC for further analysis.

```
1  Filetype: Flipper SubGhz RAW File
2  Version: 1
3  Frequency: 433889000
4  Preset: FuriHalSubGhzPresetOok650Async
5  Protocol: RAW
6  RAW_Data: 2411 -12572 165 -3994 65 -100 331 -466 331 -66 399 -100 165 -66 235 -200 22737 -15588 67 -1958 197 -464 231 -166
7  RAW_Data: 135 -522 167 -488 167 -518 167 -488 167 -516 473 -206 487 -196 445 -238 451 -198 485 -204 145 -504 481 -202 145 -
8  RAW_Data: 455 -194 471 -206 457 -200 477 -204 143 -536 479 -202 145 -506 131 -534 157 -538 131 -540 129 -536 453 -198 163 -
9  RAW_Data: 145 -504 131 -566 93 -544 165 -518 129 -538 487 -200 133 -520 483 -196 125 -5248 131 -532 141 -534 133 -526 173 -
10 RAW_Data: 139 -520 485 -212 107 -5236 137 -526 165 -524 137 -522 167 -520 135 -520 165 -522 133 -522 165 -522 135 -520 477
11 RAW_Data: 133 -520 133 -522 167 -516 163 -504 155 -512 163 -504 505 -210 457 -196 471 -204 455 -228 443 -238 109 -538 477 -
12 RAW_Data: 471 -208 457 -198 471 -206 455 -228 443 -238 109 -536 481 -202 145 -510 131 -550 139 -530 131 -526 173 -524 445 -
13 RAW_Data: 481 -202 145 -510 131 -562 139 -528 131 -526 137 -524 479 -204 143 -536 481 -202 145 -5206 135 -526 161 -536 135
14 RAW_Data: 483 -210 141 -534 447 -244 107 -5232 139 -546 133 -514 175 -500 165 -518 139 -530 133 -526 173 -492 165 -524 137
15 RAW_Data: 131 -520 171 -492 165 -520 171 -490 165 -524 137 -524 167 -524 481 -194 467 -206 455 -196 471 -224 467 -210 107 -
16 RAW_Data: 133 -524 477 -202 469 -208 487 -194 461 -212 477 -198 131 -538 473 -210 137 -542 103 -546 131 -532 127 -536 137 -
17 RAW_Data: 139 -536 469 -206 139 -522 141 -534 131 -560 139 -528 131 -522 481 -198 125 -554 479 -194 125 -5250 131 -530 127
18 RAW_Data: 131 -510 473 -210 141 -532 475 -206 143 -5228 137 -524 131 -560 141 -502 163 -500 157 -538 135 -520 161 -506 155
19 RAW_Data: 161 -538 139 -528 133 -520 171 -524 129 -536 123 -522 163 -506 169 -524 473 -202 473 -210 447 -236 443 -204 471 -
20 RAW_Data: 135 -524 165 -522 479 -196 465 -208 459 -198 473 -204 455 -228 131 -536 473 -210 139 -510 127 -536 137 -556 131 -
21 RAW_Data: 441 -202 143 -518 483 -202 145 -516 165 -518 139 -530 133 -558 139 -526 447 -204 143 -538 483 -202 145 -5198 167
22 RAW_Data: 131 -526 137 -560 445 -204 143 -538 481 -202 145 -5204 135 -560 129 -508 169 -524 131 -548 133 -524 131 -542 129
23 RAW_Data: 137 -522 165 -524 135 -520 165 -526 133 -520 167 -488 167 -520 165 -522 133 -520 477 -206 459 -232 441 -206 493 -
24
```

Figure 12. RAW sub file open in a text editor.

In Figure 12, the contents of the file are shown open in a text editor. They are like those captured using the Read function with two key differences. The protocol value will always be RAW, and the file will contain raw data at the end of the file. The raw data is an array of the OOK pulse data in microseconds that always

starts with a positive number in this case. The reason the raw data is split into multiple rows is due to the limitation of 512 values per line.

It is possible to find the data sent by the doorbell remote from the captured raw data by looking for long gaps when the signal is off that are used as a divider between transmissions. The doorbell transmitter sends the key over and over when the button is being pressed. During a normal press the key is sent multiple times. Between these transmissions is a gap where the signal is off for a longer period. From the capture, approximately 5200 microsecond gaps were found at specific intervals. Counting the values between two of these gaps resulted in 24 unique values. This indicates that there are 24 bits of data that match the protocol. Further analysis revealed two additional types of gaps at around 500 and 200 microseconds. Since amplitude modulation is used it will be assumed that the short and long silence values present different states. Assuming long silence is 0 and short silence is 1, a binary sequence can be formed from the values. Modifying the file to have each pulse on a different row will help make reading the data easier. The binary values can then be added next to the pulses to decode the binary data:

```
1.      145 -5208 Long gap
2.      169 -490 0
3.      165 -504 0
4.      163 -536 0
5.      141 -528 0
6.      131 -536 0
7.      123 -558 0
8.      129 -510 0
9.      169 -524 0
10.     131 -544 0
11.     473 -208 1
12.     459 -198 1
13.     469 -204 1
14.     493 -166 1
15.     473 -192 1
16.     151 -548 0
```

```
17.     465 -202 1
18.     145 -532 0
19.     131 -516 0
20.     173 -502 0
21.     165 -518 0
22.     139 -528 0
23.     479 -202 1
24.     145 -508 0
25.     481 -202 1
26.     143 -5244 Long gap
```

Next the binary can be then converted into a hexadecimal value. The converted hexadecimal value 0x007D05 matches the one from the Read function data capture:

```
Binary 0000 0000 0111 1101 0000 0101
Hexadecimal 007D05
```

6.2.1 Using a spreadsheet and diagram for timing analysis

Demonstrated by Jamison in a YouTube video on sub-GHz remote signals, the timings are numerical data and can be moved into a spreadsheet for analysis (Jamison 2023a). Google Sheets was used to create a stacked column chart for visualizing the pulses. The values were moved into a spreadsheet in the original order they were captured in. With positive values in the A column and negative values in the B column, a stacked column chart was created. This was done by selecting both columns and inserting a stacked column chart.

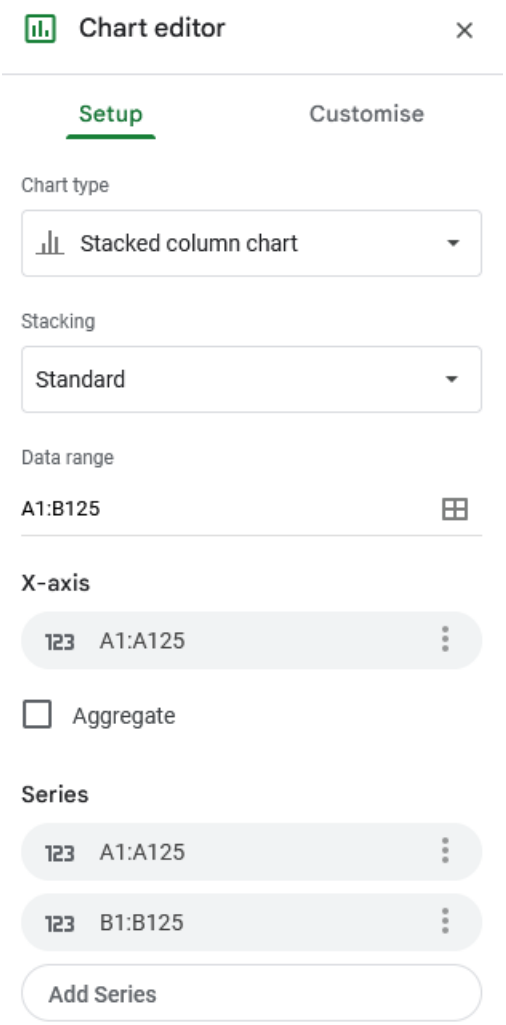
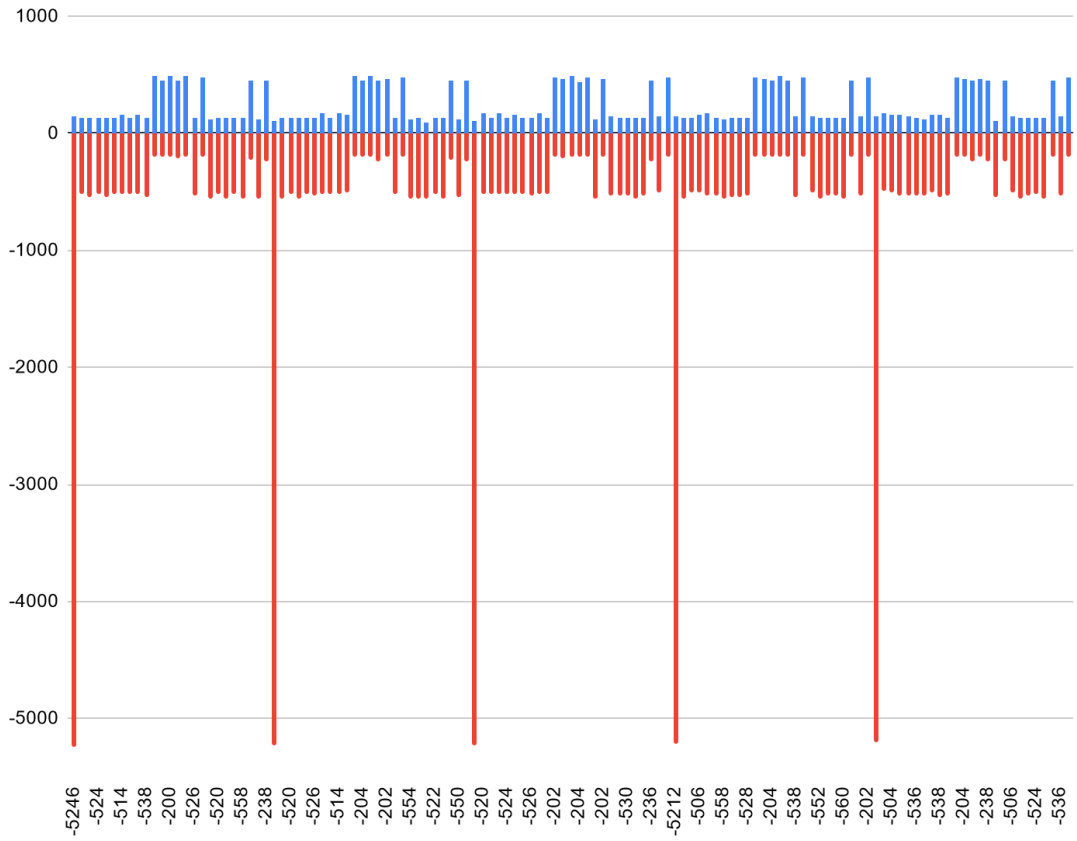


Figure 13. Stacked column chart settings in Google Sheets.

Figure 13 shows the settings for the stacked column chart. Both columns need to be added as series from the setup to stack the A and B columns.

Chart 1. Doorbell timings visualized using a stacked column chart in Google Sheets.



The resulting Chart 1 visualizes the doorbell timings from the previous example. It contains the same data being sent five times. The positive values are the pulses whereas the negative values are the silence. From the chart it is possible to analyze what type of modulation was used by comparing these timings. The chart clearly shows that the pulse width is shorter when the silence is longer and vice versa. This indicates that the modulation used is pulse width modulation. It shows over 5000 microsecond silence spikes with the same short pulse as when the silence is at around 500 microseconds. The spikes are the gaps between transmissions since the data is transmitted multiple times. This means it can be used as a delimiter for a single data transmission. This chart was used for decoding the binary values between the delimiters where the low positive spikes are binary values of 0 and high spikes are 1. The resulting binary string is the same as the earlier example.

6.2.2 Using a pulse plotter to analyze RAW files

The Flipper Lab website has a pulse plotter where it is possible to upload a RAW sub file for analysis. In addition to raw sub-GHz signals, it supports raw infrared and RFID signals.

Figure 14. Pulse plotter on the Flipper Lab webpage with the doorbell RAW sub file plotted.

As shown in Figure 14, it will present the OOK pulses visually on a timeline on the horizontal axis and on and off state of the signal on the vertical axis. The slicer settings are adjusted below the plot with the short value being the length of a short pulse, and the value being a long pulse. Sync is the length of a sync sequence, which in this case is the gap between transmissions. The short and long pulses are figured out from the pulses section where it lists the number of detected pulses of any specific length. For PWM they are normally the two most frequent values. In this case, the third frequent value is the sync length.

Figure 15. Close up of the plotted pulses demonstrating OOK PWM.

Figure 15 has the pulses with the length of the longest part of the period written in microseconds. The vertical dotted lines divide the timeline into periods. The tool could decode the bits in this case.

Bits: {5894} / 58 / DF 75 00 / 00 1F 41 40 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00
00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 0
00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 0
00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 0
00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 0
00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 0
00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 00 7D 05 / 0

Figure 16. Bits presented on the pulse plotter site using hexadecimals.

Figure 16 is the bottom part of the page where the bits are decoded into hexadecimal values. The values match the key 0x007D05 decoded in earlier examples.

7 SETTING UP RTL-SDR

To begin using the RTL-SDR dongle on Linux systems, the drivers need to be installed first. On Fedora Linux a package called rtl-sdr was installed using the dnf package manager that provides the needed driver for communication with the device. After installing the package, SDR software must be installed for operating

the SDR-RTL, such as, rtl_433 and SDR++. Installing rtl_433 will be looked at later in this work.

On Windows the setup is a bit different. For RTL-SDR Dongle V3, a generic driver had to be installed using a program called Zadig. After that, setting up SDR# software required downloading the rtl-sdr drivers as DLL files from the RTL-SDR-Blog GitHub repository and placing the rtlsdr.dll file to the root folder of the program for RTL-SDR support.

Out of the box the RTL-SDR dongle does not have any antenna. The dongle is officially sold in a bundle that includes two dipole antennas, but a separate antenna can be bought or built. The two dipole antennas provided in the bundle are not ideal for the 433.92 MHz frequency band so a better antenna will be built. The dongle has a standard SMA female connector, which means the cable from the antenna must be a male SMA connector or an adapter must be used.

7.1 Antenna design and build

A quarter wave ground plane antenna is a vertical antenna with ground plane radials. Materials used for building the antenna were enameled copper wire with 0.6 mm diameter and an SMA connector with four panel mount holes. The holes on the connector will be used to attach the ground plane radials. A soldering iron and solder are needed to attach the copper wires to the connector. Calculating element length of a quarter wave vertical antenna is done using Formula 2 (Cirera 2023).

$$\lambda = \frac{c}{f} \tag{2}$$

$$L = \frac{\lambda}{4} \cdot V_f \quad = \frac{(C \div f)}{4} \cdot V_f$$

where	λ	wavelength	[m]
	c	speed of light	[m/s]
	f	frequency	[Hz]

L	element length	[m]
V_f	velocity factor	[-]

Formula 2 was used to calculate the length of a quarter wave antenna element resonant at 433.92 MHz:

$$L = \frac{(299792458 \div 433920000)}{4} \cdot 0.95 \quad = 0.16408\ldots = 16.4\ cm$$

Formula 2 shows that the element length for the antenna is approximately 16.4 centimeters.

The antenna was modeled using an antenna modeling software for theoretical SWR and to visualize the radiation pattern. The program used was MMANA-GAL for Windows operating systems that has a free basic version available for download on their website. Once opened, the program defaulted to the geometry tab. The geometry tab in the software allows the modeling of the antenna elements.

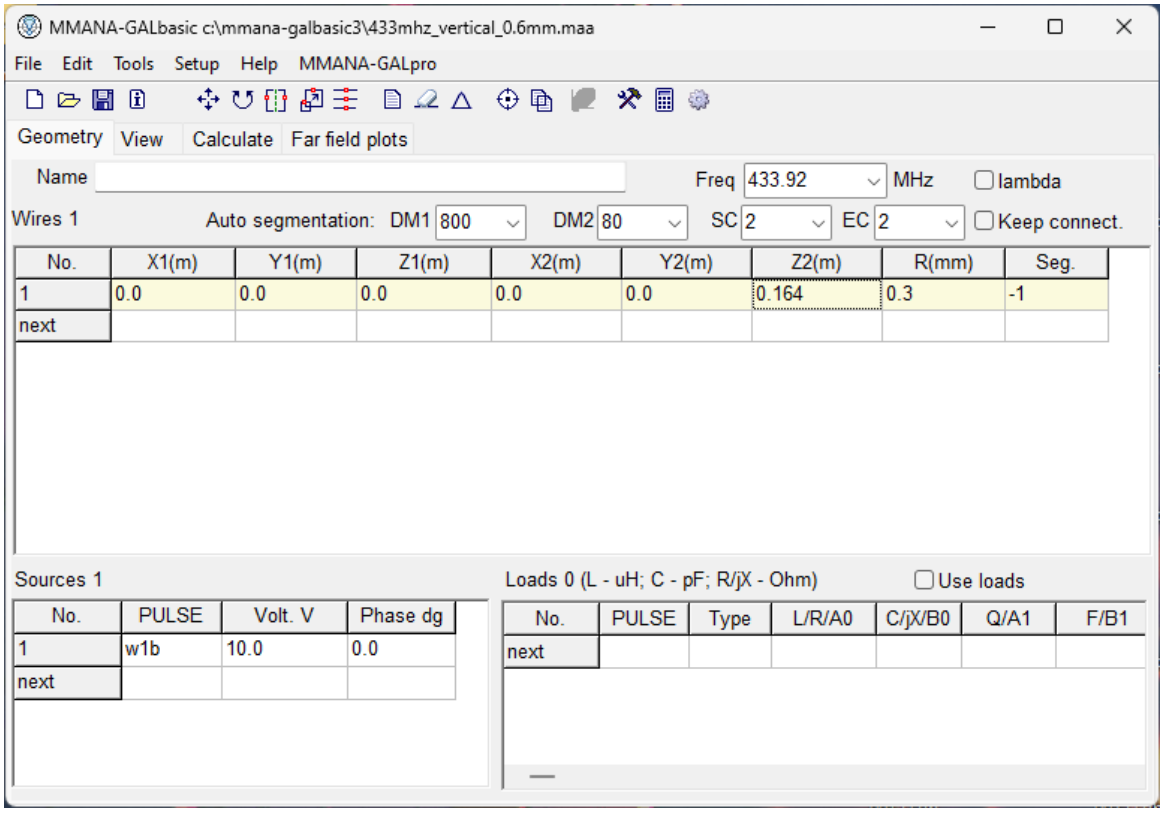


Figure 17. Antenna geometry tab filled in MMANA-GAL basic.

As demonstrated in Figure 17, values were inserted to the different fields to model the wires of an antenna. The frequency field on top is the frequency for the antenna, which in this case is 433.92 MHz. The quarter wave ground plane antenna has one radiating element. Next, the element was added as a wire in the Z axis. By filling the Z2 field with the length of the element in meters, which is 0.164 m from the formula, the wire was modeled in the software. X1, Y1 and Z1 are the starting points of the wire on each axis thus they were left at values of 0. Radius of the wire is set in millimeters using the R field. The diameter of the copper wire is 0.6 mm, which meant the value was set to 0.3 mm. The segmentation method field is automatically filled to the recommended default value of -1. Next the antenna model was viewed from the view tab.

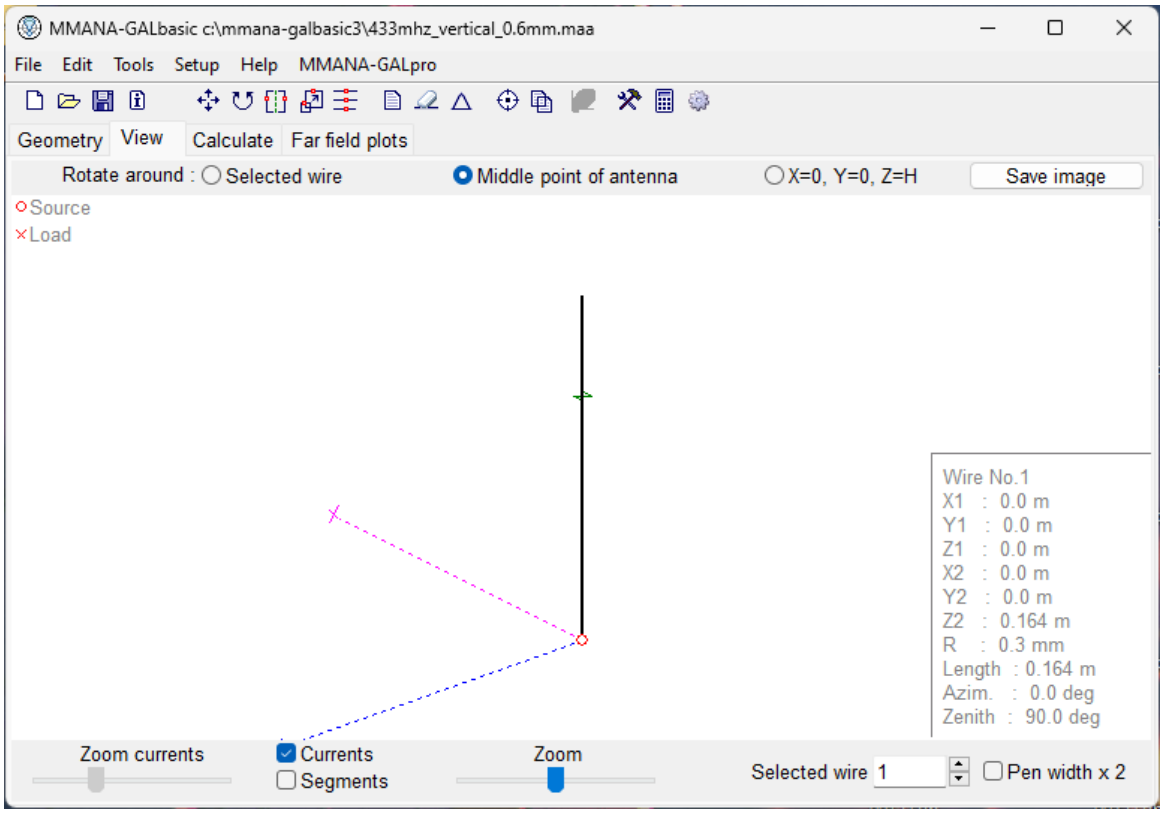


Figure 18. View tab showing the antenna element in space.

As demonstrated in Figure 18, the model is viewable in 3D space. The bottom right corner shows the position of the wire along with azimuth and zenith angles. The source of the signal is shown with a red circle that is the connector. This tab

is especially useful for an antenna with multiple elements as a visualization tool. The program can calculate the performance of the modeled antenna.

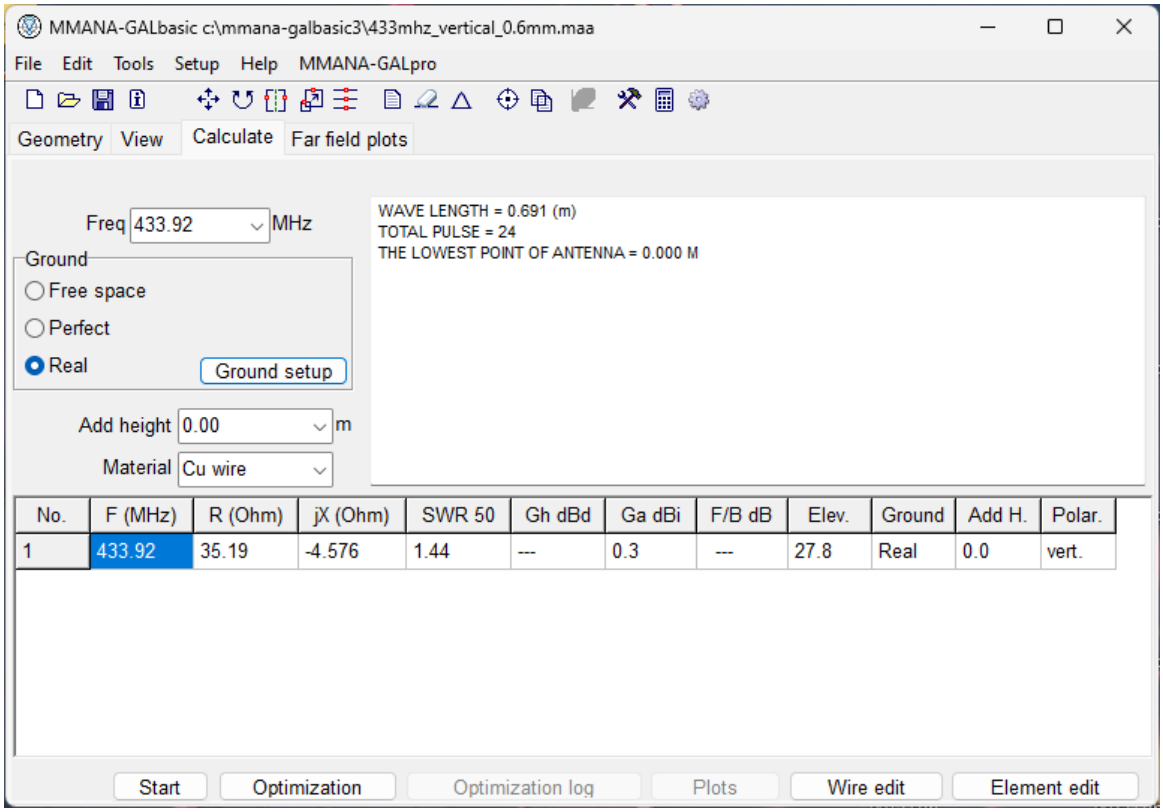


Figure 19. Calculate tab with 433.92 MHz antenna calculations.

The calculate tab, shown in Figure 19, was accessed to perform the calculation. It also creates the calculations to present the far field plots for the design. The material of the wire was set to copper wire for this design. The ground plane was then set up using the ground setup menu. The ground setting had to be changed to real for the ground setup to become accessible.

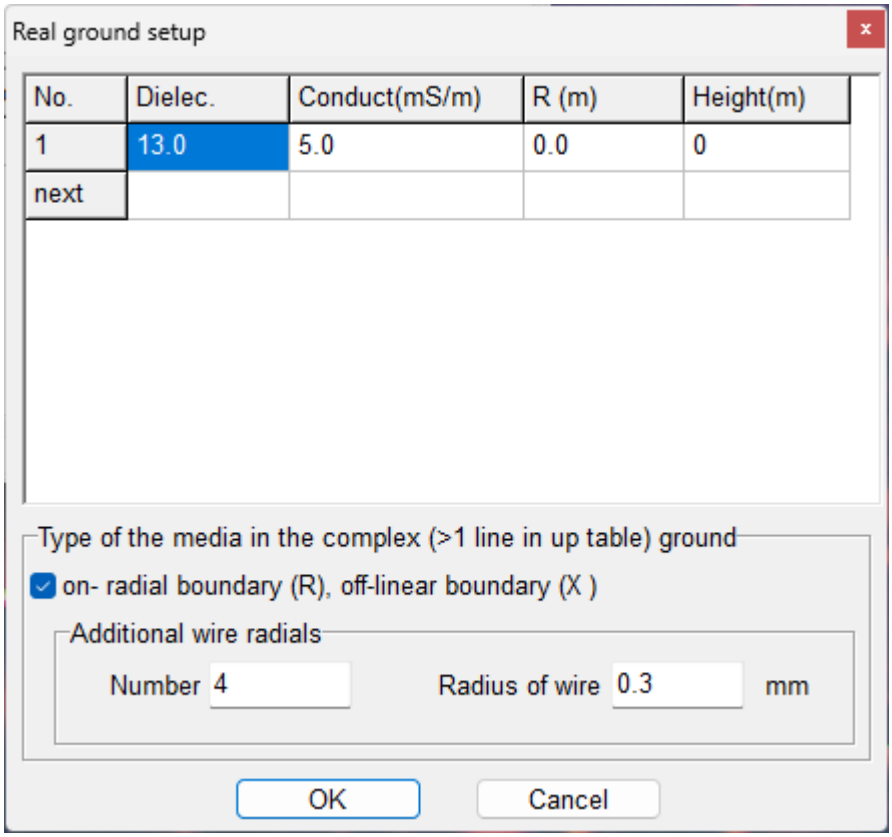


Figure 20. Ground setup popup.

The menu in Figure 20 popped up when the ground setup button was pressed. The number of radials and their radius was defined here. In this design four radials are made from the same wire. Next the start button was pressed in the calculate tab to run the calculations. As seen in Figure 19, the SWR for this design is 1.44 at 433.92 MHz. The peak gain in dBi was 0.3. Next the far field plots were accessed to see the radiation patterns.

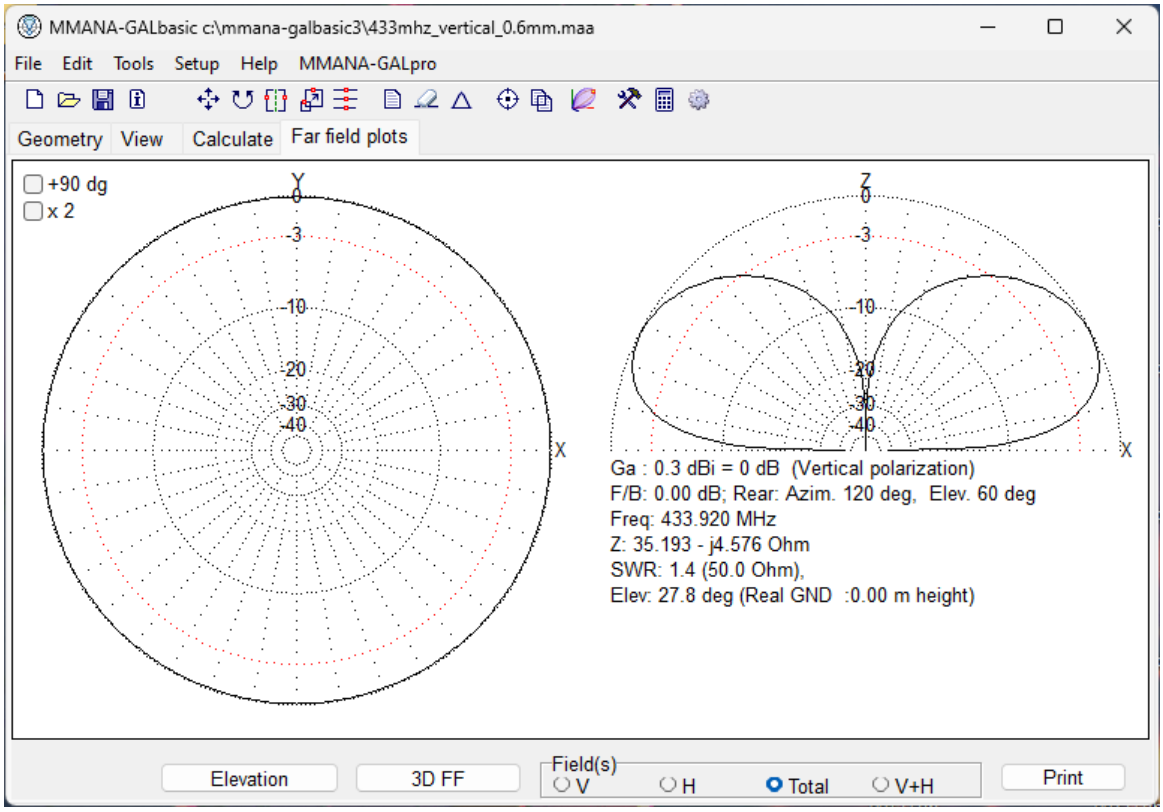


Figure 21. Far field plots tab showing two-dimensional graphs.

Since the calculations were run, the far field plots tab was populated as demonstrated in Figure 21. It shows separate two-dimensional far field plots for the xy axes and xz axes. Top left corner has a setting to toggle and rotate the plot 90 degrees to change the position of the horizontal axes. Additionally, a three-dimensional far field plot was viewable in the program.

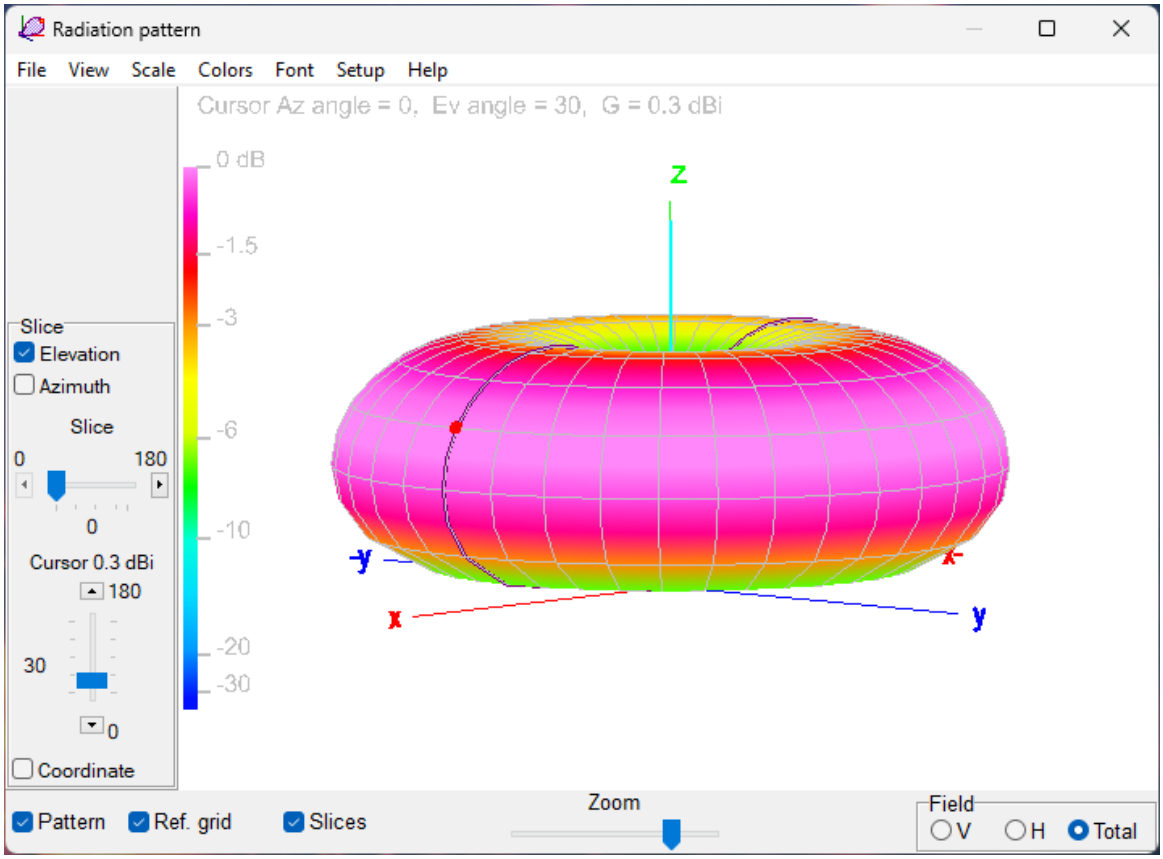


Figure 22. 3D representation of the radiation pattern.

Pressing the 3D FF button opened the three-dimensional far field plot window shown in Figure 22. Toggling slices on from the bottom enabled a cursor, shown as a red dot, that can be placed on the pattern for a dBi measurement at its position. Maximum gain of 0.3 dBi was at around 25-30 degrees elevation angle on this pattern. The azimuth angle does not matter since the vertical antenna radiates equally in the horizontal plane. The program can perform automatic optimization of the antenna design. From the calculation tab, automatic optimization was accessed by clicking the optimization button.

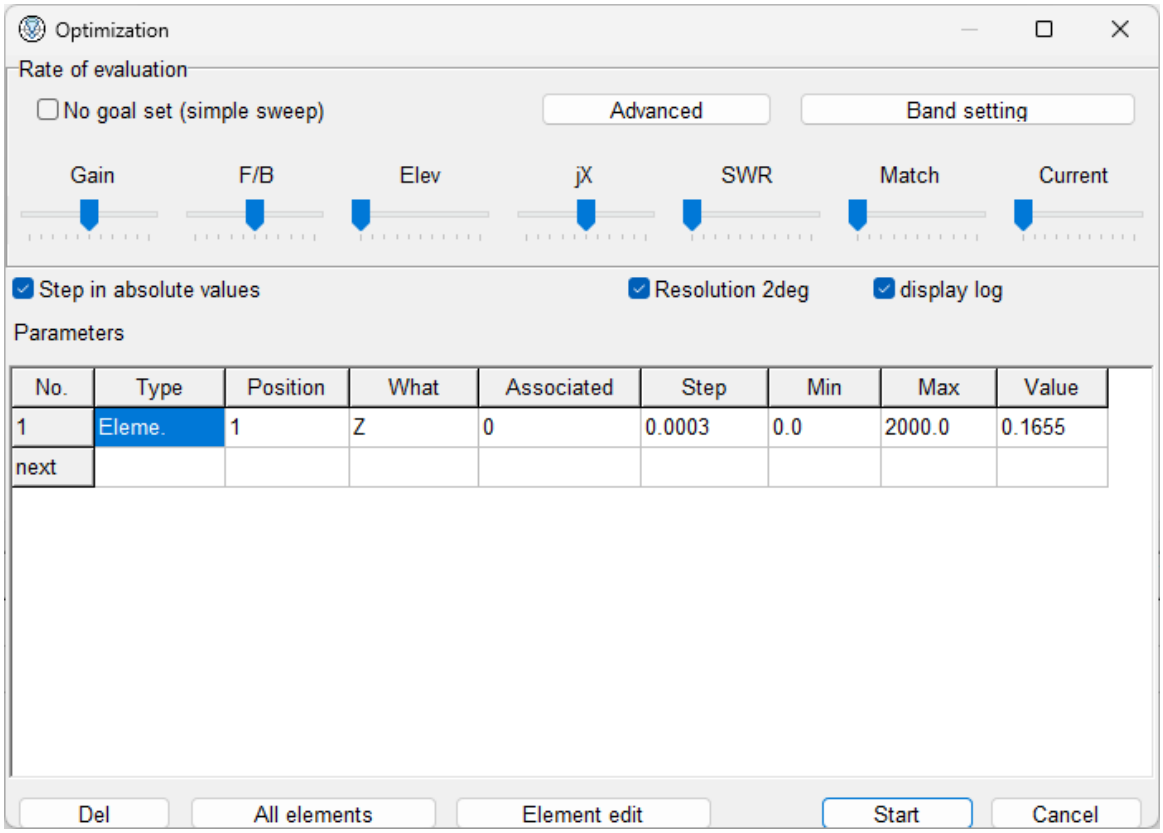


Figure 23. Optimization popup menu.

It opened a window, shown in Figure 23, where settings are adjusted. In this case, all elements button was clicked to select the only element and then the optimization was performed using the start button. The program ran the optimization and presented the values.

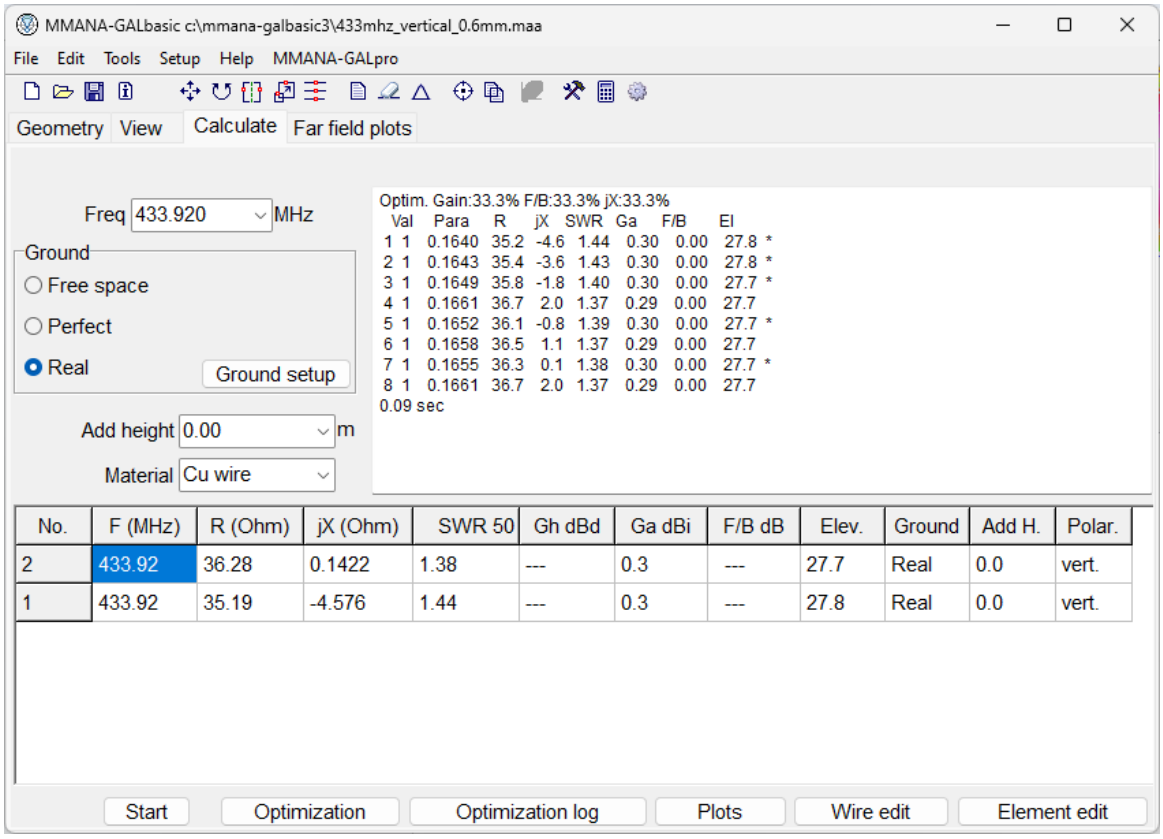


Figure 24. Results of the automatic optimization.

Figure 24 shows the result for the optimization on top of the table at the bottom. The log on the top right has the results for the different values that were tested. The tool determined that 0.1655 meters is the optimal element length, lowering the SWR from 1.44 to 1.38. The geometry tab was accessed next to see if the element length was updated.

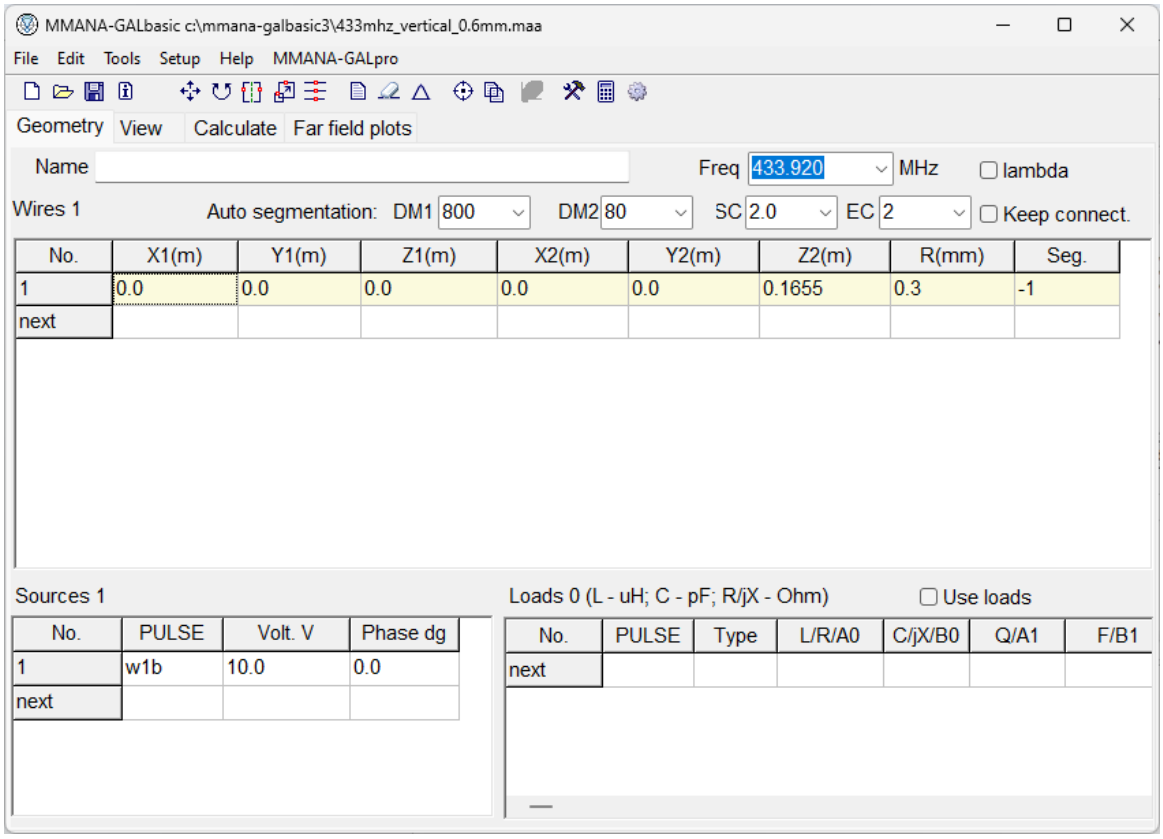


Figure 25. Adjusted Z2 value after the optimization.

The tool changed the Z2 element length automatically to the optimized value, which is shown on the geometry tab in Figure 25. Next the antenna was built using these values. The copper wire was cut pieces that had a couple of millimeters extra since the antenna performance will be measured and tuned later if needed. Removing the enamel coating from the end of the wire was required before soldering and was done using a small piece of folded sandpaper. Another option is using a knife, such as a craft knife, to scrape off the coating. The copper wire pieces were soldered onto the connector. The vertical element was attached to the side of the metal piece sticking out from the middle of the Teflon insulation. The four radials were attached to the mounting holes of the connector pointing downwards towards the connector side. The radials were adjusted to a 45 degrees angle after attaching them by bending the wires. The 45-degree angle matches the impedance to the 50-ohm wire, cable and connectors (Cirera 2023). The 0.6 mm diameter wire proved to be too flexible for the antenna to stand on the ground plane, which meant that the antenna needed to be attached to something.

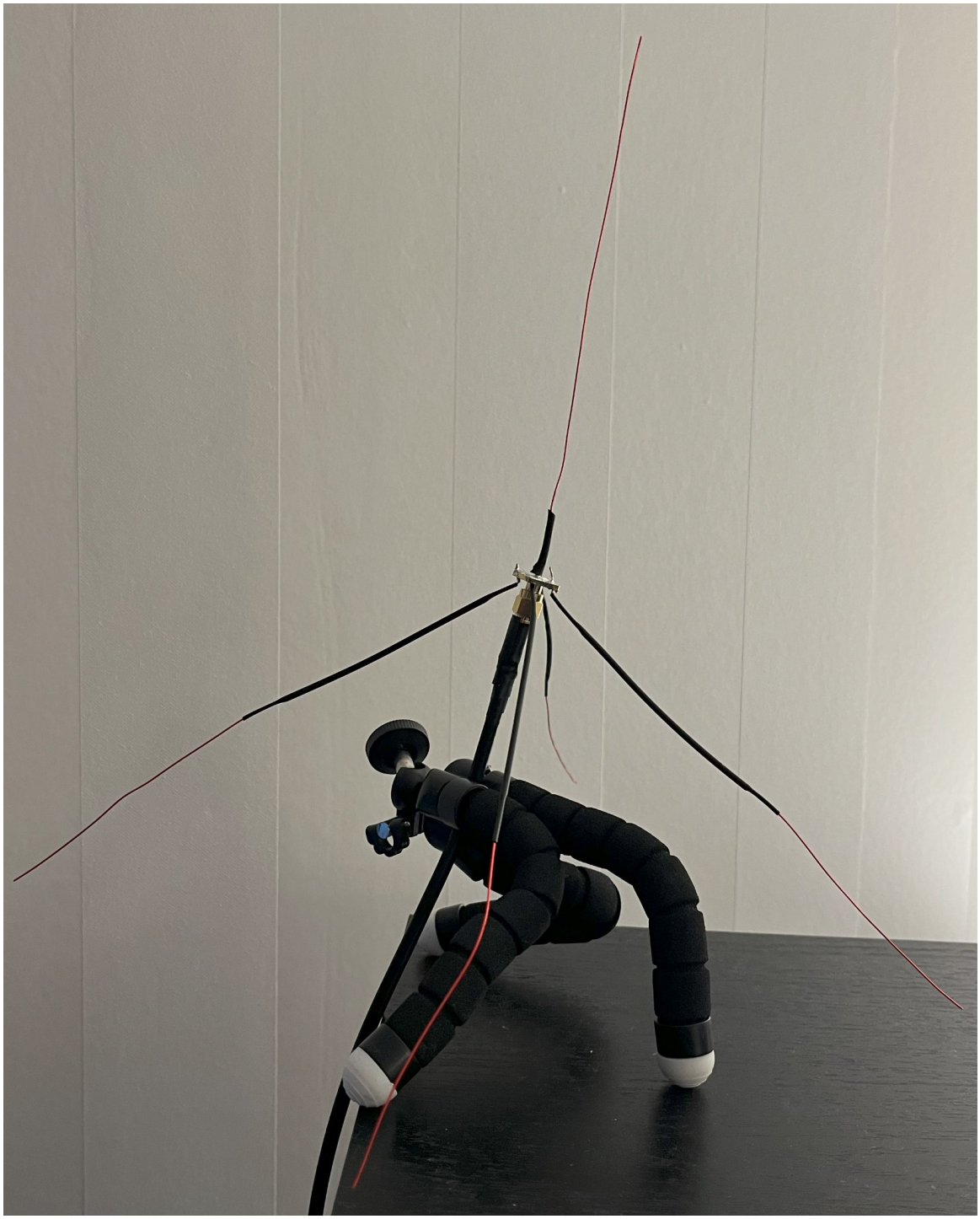


Figure 26. The 433.92 MHz quarter wave ground plane antenna is attached to a tripod.

Figure 26 demonstrates that a small tripod from the RTL-SDR bundle was used to pinch the coaxial cable between the legs to hold the antenna in place while suspending it so that the ground plane wires hold the 45-degree angle. 1.5 mm copper wire was tested later and held its shape better. It required a larger tip on

the soldering iron and more heat to attach the ground plane because there was more copper to sink the heat, and the solder did not melt.

7.2 Antenna testing and calibration

Calibration of the antenna was done using the NanoVNA vector network analyzer. The device first needed calibration and setup to accurately measure the antenna's SWR. To calibrate the NanoVNA three bits for open, short and load were needed. They are usually included with the NanoVNA kit but depending on the manufacturer it is possible that they need to be bought separately.

Figure 27. Calibration set for the Nano VNA includes, from left to right, the open, short and load bits.

Figure 27 shows the bits used for the calibration with the left one being the open, middle one shorts the connector, and the right one introduces a load. The calibration is performed from the calibration menu on NanoVNA. First the frequency range for the calibration of the device was set from 100 to 500 MHz. Next the calibration was accessed, and the bits were attached one by one.

Figure 28. NanoVNA calibration menu.

The buttons open, short and load, shown in Figure 28, were pressed based on the bit that was attached. Once the open, short and load were calibrated, the done button was used to save the calibration to memory for that frequency range. The display for measuring needed to be set up next. From the display menu the device was adjusted to show the phase and smith charts for the S11 connector, which is port 1, that was calibrated.

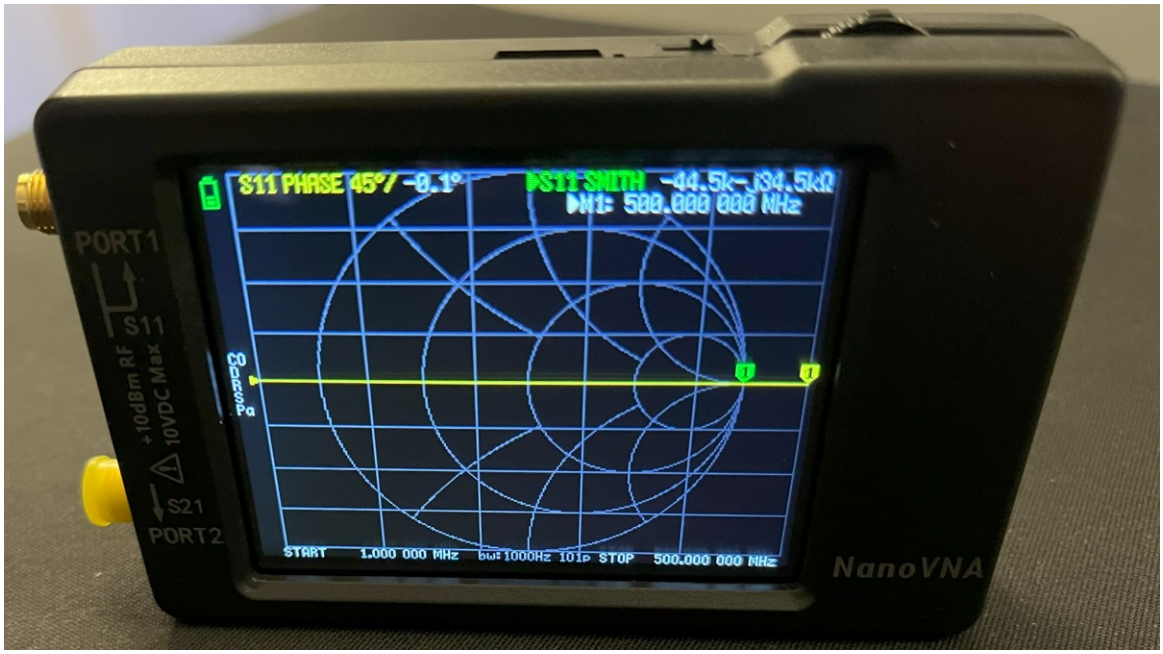


Figure 29. Phase and smith charts after calibration.

Figure 29 demonstrates the charts after calibration where the phase is 0 degrees with the yellow cursor showing at the right edge of the screen at 500MHz. The green cursor is on the smith chart and is located on the middle right edge of the circle at 0 degrees. The connector on the antenna was female and the NanoVNA has a female connector for measuring, which means a male-to-male adapter was required to attach the antenna.

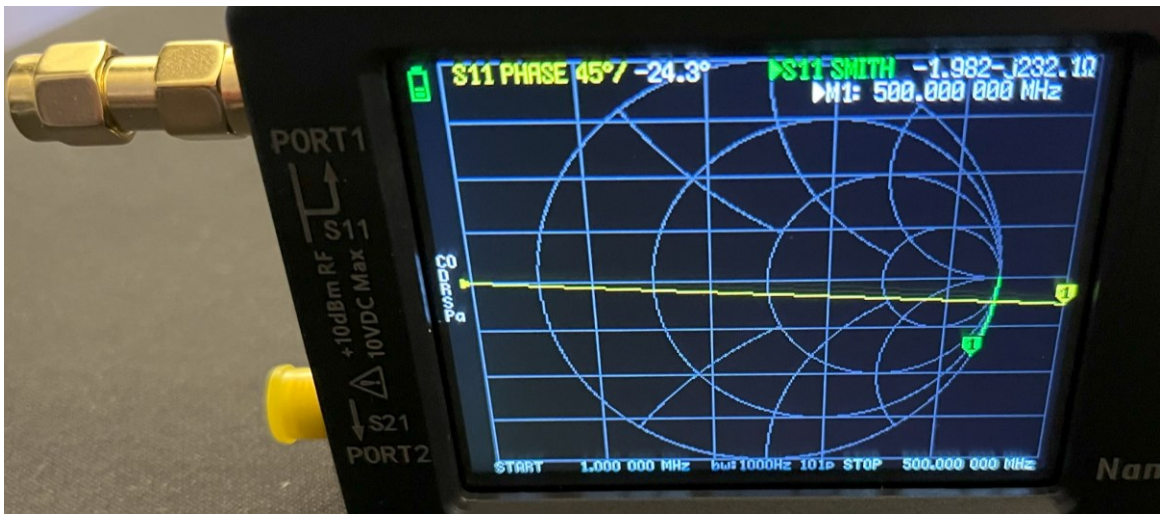


Figure 30. Phase and smith chart after attaching a male-to-male adapter.

Figure 30 visualizes what happened when a male-to-male adapter was introduced. The phase shifted to -24.3 degrees and the cursor for the smith chart moved from its original calibrated position. This means the antenna cannot be accurately measured since the starting point is wrong. This happens because it now takes more time to get to the end of the connector and the device expects that the antenna is attached straight to port 1. The most accurate way to handle this is a female calibration set attached to the adapter, but the device usually does not include them, and they can cost as much as the NanoVNA itself. To work around this, an electronic delay was set to adjust the calibration. From the NanoVNA menu, under Display and Scale, an option called e-delay for electronic delay was adjusted.

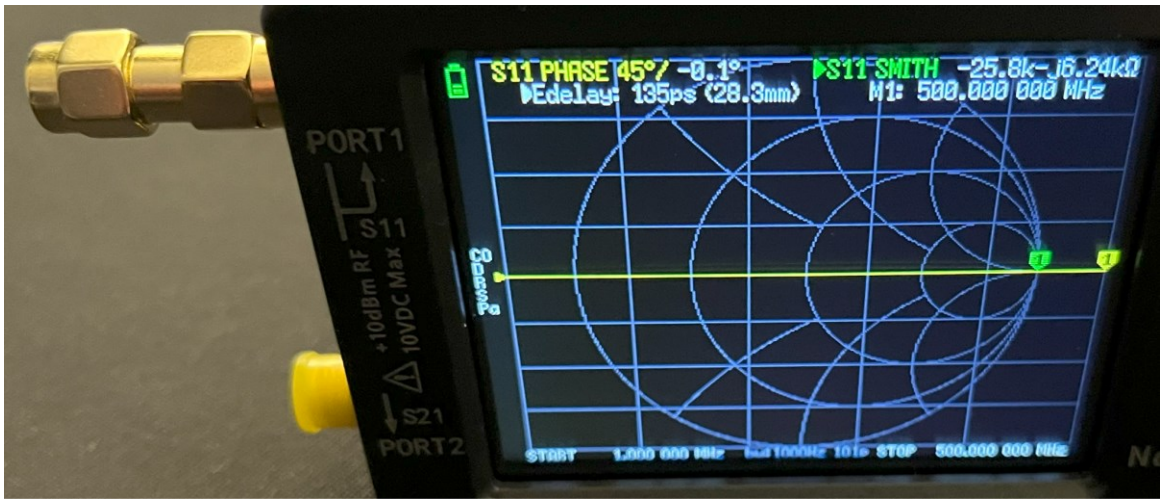


Figure 31. Phase and smith chart with male-to-male adapter connected and electronic delay adjusted to 135 picoseconds.

The value was first set at 100 picoseconds, which was not quite correct but after experimenting with 10 and 5 picosecond intervals it was found out that 135 picosecond delay will get the starting point just about back to where it was after calibration as demonstrated on Figure 31. The Edelay value on the top left of the screen shows the delay being used with measurements also in millimeters. The same method can be used when attaching cables.

For the antenna calibration, the standing wave ratio (SWR) was measured to know how much return loss the antenna had. Both SWR and return loss are

measures for the signal being reflected by the connector indicating the amount of signal getting transferred. The closer the SWR is to the ratio of 1:1 the better. On the NanoVNA, the SWR measurements are shown by enabling them through the display menu for S11.

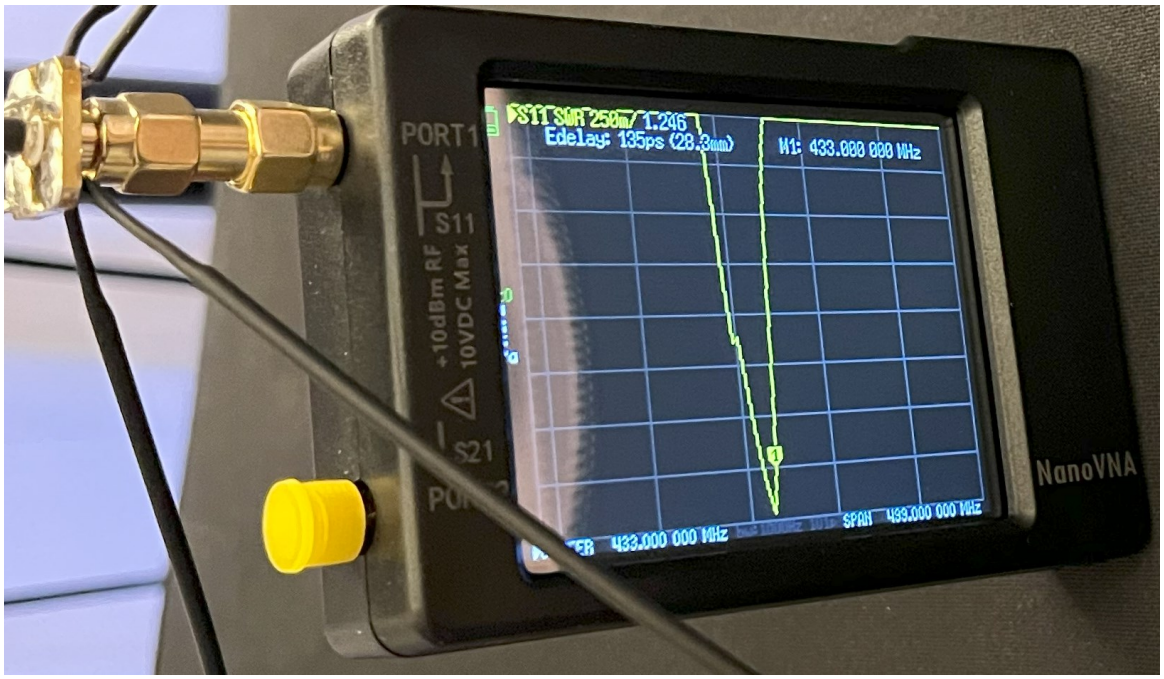


Figure 32. Standing wave ratio of the do-it-yourself antenna measured with cursor on the 433 MHz.

The cursor was centered at 433 MHz and Figure 32 has the measurements for the SWR on the left top corner with the value of 1.246. This performance is very good since the value should be between 1:1 and 1:2, which meant the antenna did not require any additional calibration.

7.3 Setting up rtl_433

To get the latest version of rtl_433, it was built from the source code. First Git was installed using the package manager DNF provided with Fedora. Git is a revision control software that was used to clone the latest version of the source code from rtl_433 project's GitHub page. After installation, the git clone command was run using the url to rtl_433 project GitHub repository:

```
$ cd ~
$ git clone https://github.com/merbanan/rtl_433.git
$ cd rtl_433
```

The cd ~ moves to the home directory of the user and the git command clones the source code from the repository to the local hard drive. Then cd was used to move to the directory with the cloned files. Next the program was built and installed:

```
$ cmake -DFORCE_COLORED_BUILD:BOOL=ON -GNinja -B build
$ cmake --build build -j 4
$ sudo cmake --build build --target install
```

The first cmake command was used to configure before building, the second cmake command built the code and the third one used sudo to install it globally to all users. If the installation is done to the local user's home directory, the sudo will be left out.

8 RTL_433 OPERATION

The rtl_433 program is accessed from the terminal emulator using the rtl_433 command. When run with default settings on a residential area at 433.92 MHz, rtl_433 will typically start decoding weather stations.

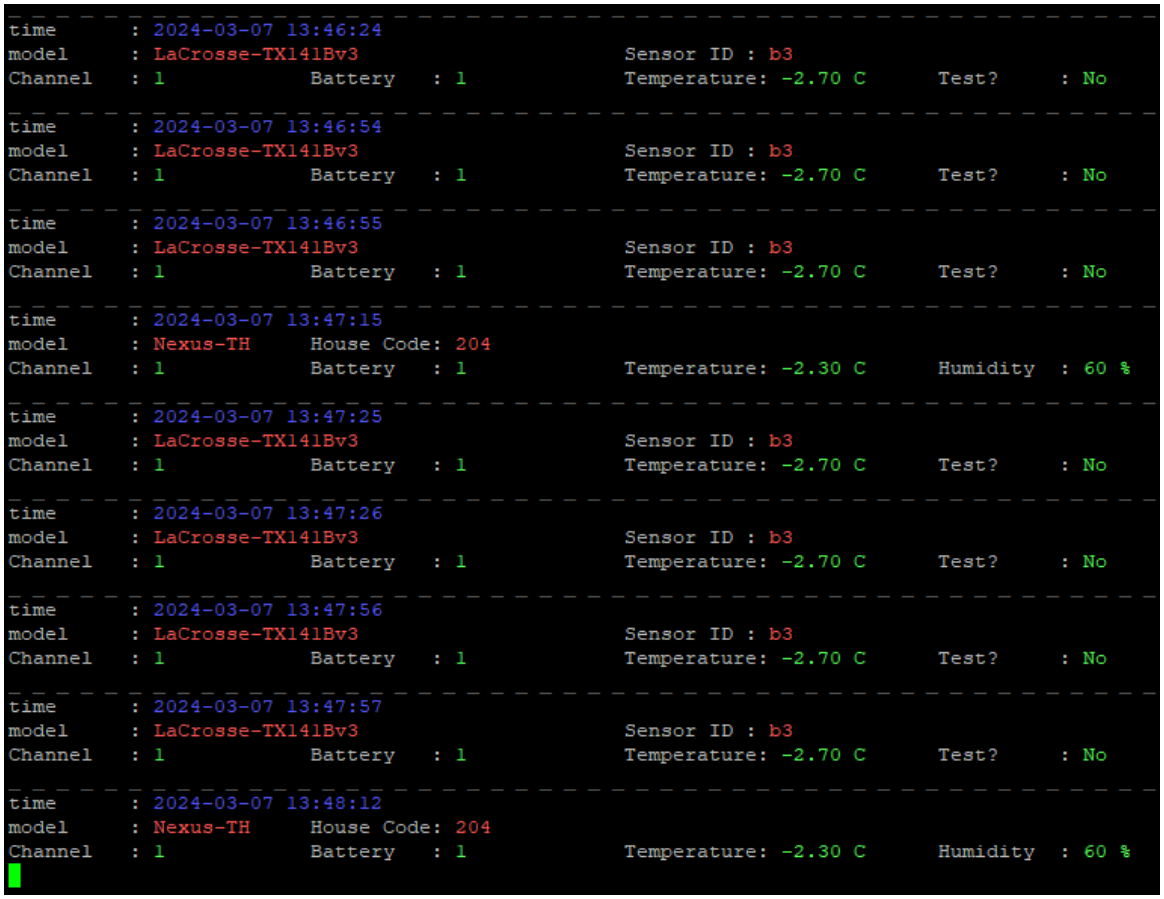


Figure 33. Typical decoded weather station data captured by rtl_433.

The normal decoder mode and weather station captures are shown in Figure 33. It shows the date and time when a signal was detected, and the model field describes the protocol that was used for decoding. This is followed by all the fields for the information that was decoded.

Analyzing the doorbell, which was used with the Flipper Zero, was done using rtl_433 in the pulse analyzer mode. It is accessed using -A flag when running the command. By default, it will still decode data, but it is possible to disable decoding using the flag -R 0. Analyzer mode was run, and the doorbell button was pressed.

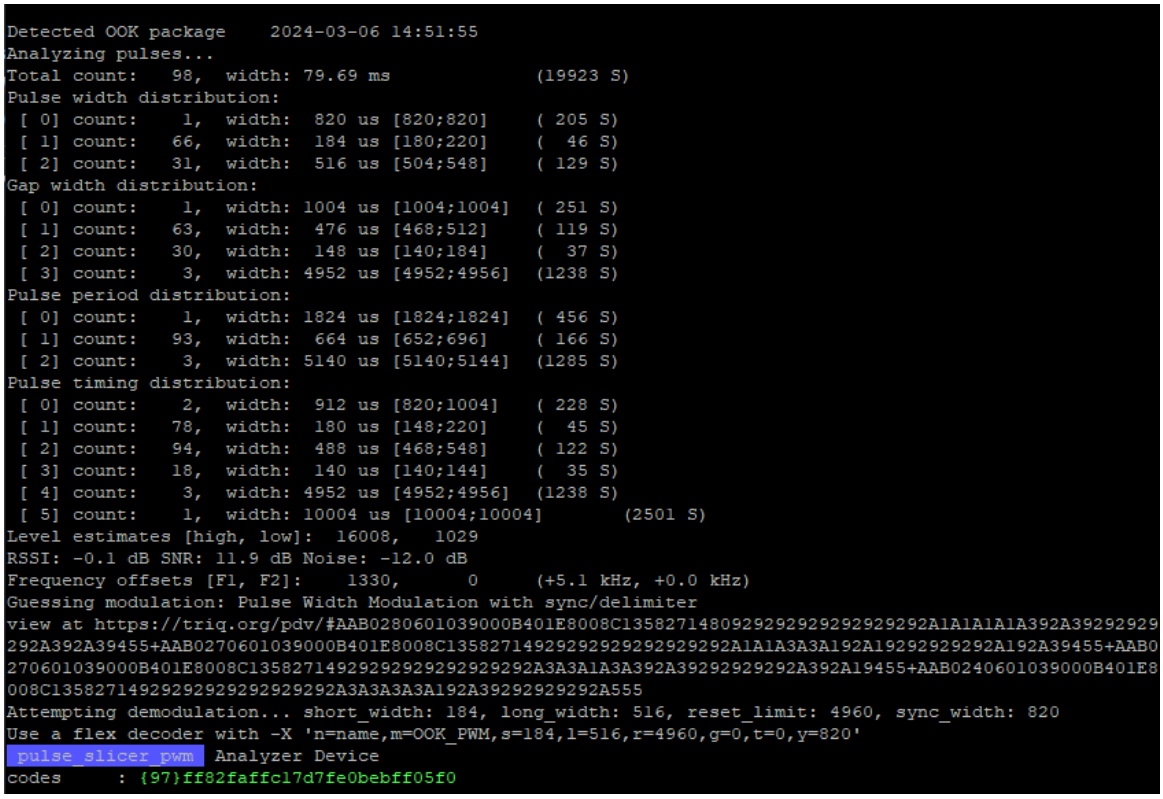


Figure 34. Doorbell signal captured using RTL-SDR and rtl_433.

The results are shown in Figure 34. It detected the data signal as an OOK package and analyzed the pulse and gap widths listing the counts received. Additionally, it analyzed pulse period and timing distributions. Signal quality is shown using RSSI, SNR and noise values in decibels. The program attempted demodulation using the short and long widths with the highest count and showed a code at the bottom. It did not match the results from Flipper Zero analysis.

Spotting a button press can be challenging because of frequent traffic and noise on the band. The button was pressed close to the antenna, and it was relatively easy to spot the received signal using the RSSI value. Since the button was pressed close to the antenna, the RSSI value was very close to 0dB at -0.1dB. It is possible to add a minimum detection level from -1.0 to -99.0 dB. For devices near the antenna, it makes sense to limit it to -1.0 as follows:

```
$ rtl_433 -Y minlevel=-1.0 -A -R 0
```

From the pulse and gap width distributions it was possible to get an idea what type of modulation was used. Figure 34 shows that 184 microsecond pulses were counted 66 times, and 516 microsecond pulses were counted 31 times. This indicates that the width of the pulse is changing. Same was noticed for the gaps with 63 gaps of 476 microsecond width and 30 gaps of 148 microsecond width detected. The pulse period was 664 microseconds, that is the width of the pulse and gap combined. With this information it can be determined that this is pulse width modulation, which was what the program was guessing as well. The delimiter is the 5140 microsecond pulse period consisting of the long 4952 microsecond silence and a short pulse.

Signals can be recorded as raw I/Q stream samples for analysis using the -S flag that needs an argument to specify if all, known or unknown data will be captured. For ease of use -S all was used. The command run was as follows:

```
$ rtl_433 -Y minlevel=-1.0 -A -R 0 -S all
```

The program created files with the following naming scheme when individual signals were recorded:

```
g001_433.92M_250k.cu8
g002_433.92M_250k.cu8
...
```

The first part is a number in sequence, the second part is the frequency band followed by the sample rate set in the configuration. The file extension used is cu8, which means complex 8-bit unsigned integer samples. These files are replayed using the -r flag and specifying the file name and directory:

```
$ rtl_433 -A -R 0 -r g001_433.92M_250k.cu8
```

The raw OOK pulse data can be either printed to the stdout using -w flag with the argument OOK:- or saved into a file by specifying the filename after the flag, for example, -w g007_433.92M_250k.ook. Printing the data from the doorbell capture revealed similar data to the Flipper Zero RAW captures:

```
$ rtl_433 -R 0 -w OOK:- ./g007_433.92M_250k.cu8
...
168 500
504 168
168 5204
164 508
164 504
168 504
164 504
168 504
...
```

The only difference is that the off pulses do not start with a minus. The same binary values, as from the Flipper Zero RAW data, were decoded from between the two long off gaps at around 5200:

```
Binary 0000 0000 0111 1101 0000 0101
Hexadecimal 007D05
```

An online I/Q spectrogram and pulse data analyzer is available online on triq.org website. The rtl_433 analyze feature automatically generates an URL to triq.org when it detects an OOK package to analyze it on the website.

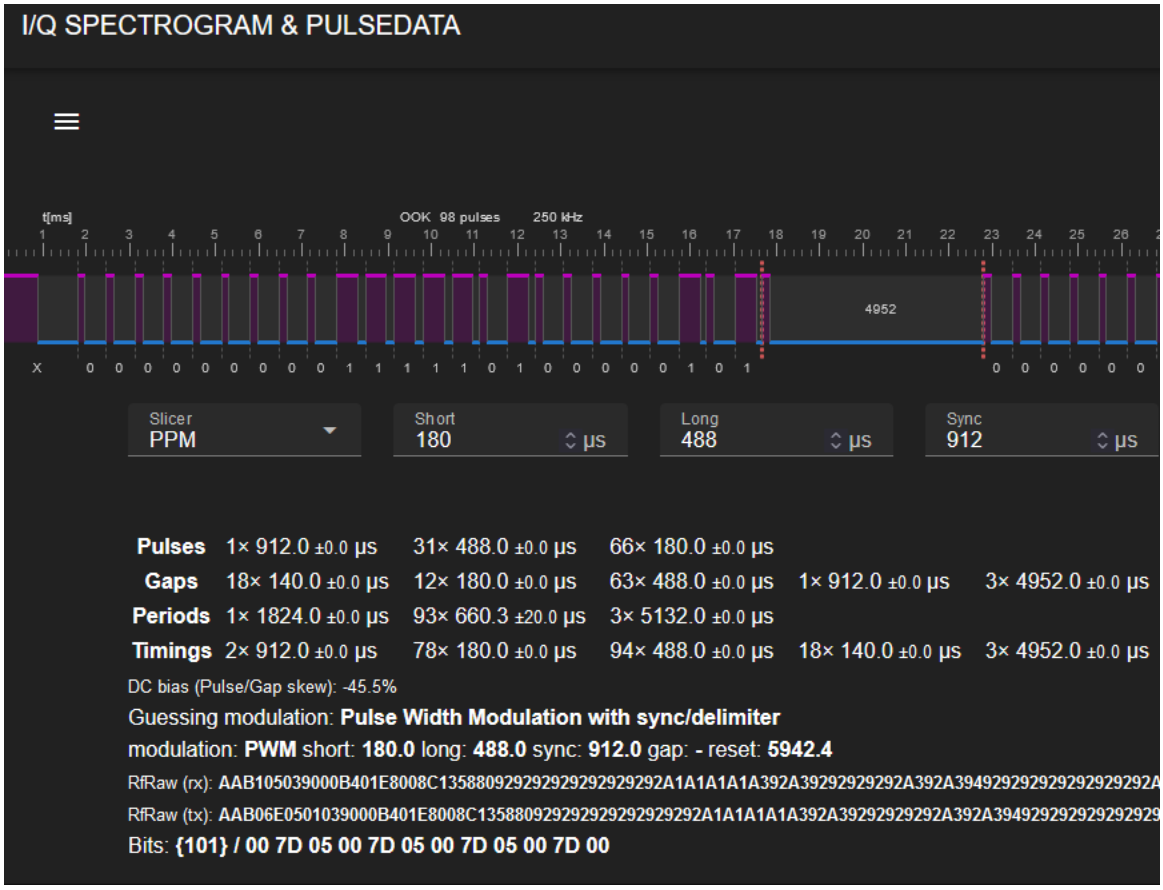


Figure 35. I/Q spectrogram and pulse data analyzer on triq.org.

Figure 35 shows the captured doorbell package as a visual representation of the OOK pulses, which matches the binary representation from before. It guessed the modulation as PWM and in the bits section at the bottom showed the values in hexadecimal.

9 ANALYZING RECORDED RAW I/Q STREAMS USING AUDACITY

The recorded files can be analyzed using an audio workstation. The recorded raw I/Q stream was imported into an audio workstation called Audacity, which is a free and open-source program.

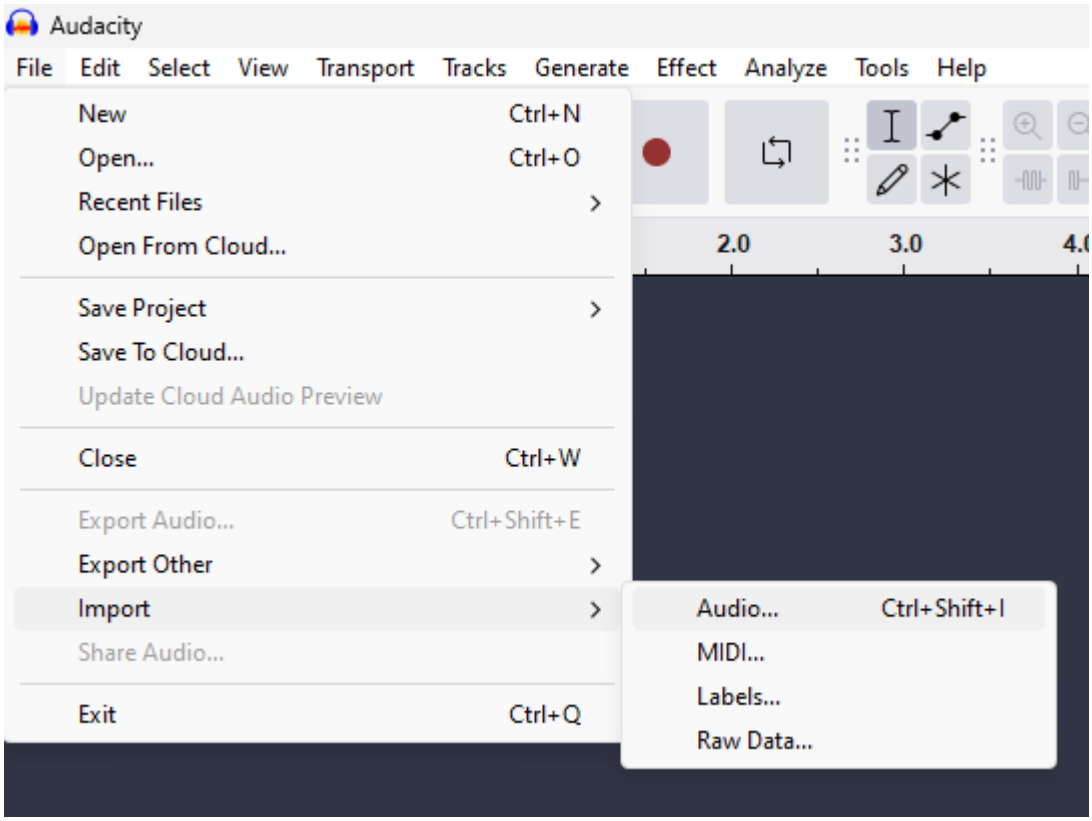


Figure 36. File import menu in Audacity.

Figure 36 shows the import file menu where raw data was selected to be imported. After selecting a file that was recorded using rtl_433, correct parameters needed to be selected from a popup.

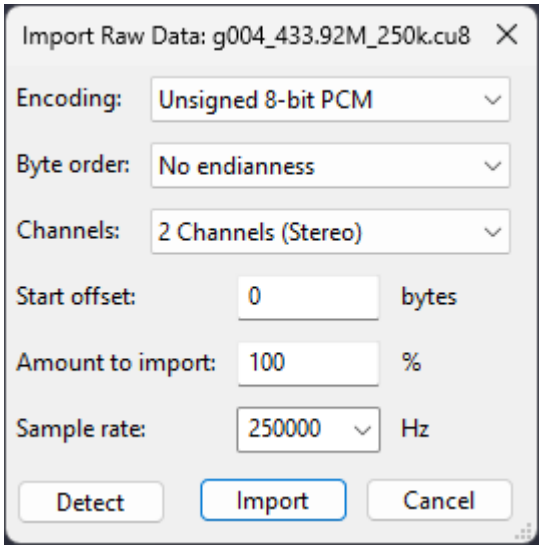


Figure 37. Settings for importing raw data into Audacity.

The popup in Figure 37 has the settings for the sample rate raw I/Q stream selected. Encoding of the data was set to unsigned 8-bit pulse code modulation (PCM) for the rtl_433 recordings and the byte order of no endianness was selected. This is because rtl_433 records the file as complex 8-bit unsigned integer samples. Sample rate had to be manually set to 250000 hz in this case but depends on the sample rate set on rtl_433.

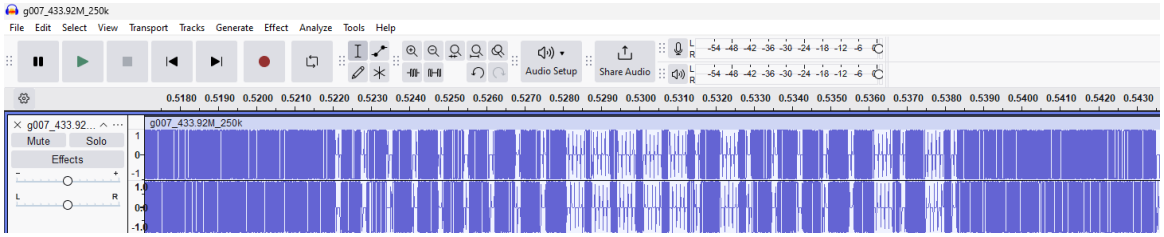


Figure 38. Waveform of a recorded raw I/Q stream in Audacity.

After importing, the waveform is visible for analysis as shown in Figure 38. The waveform needed to be zoomed to properly see the changes in patterns.

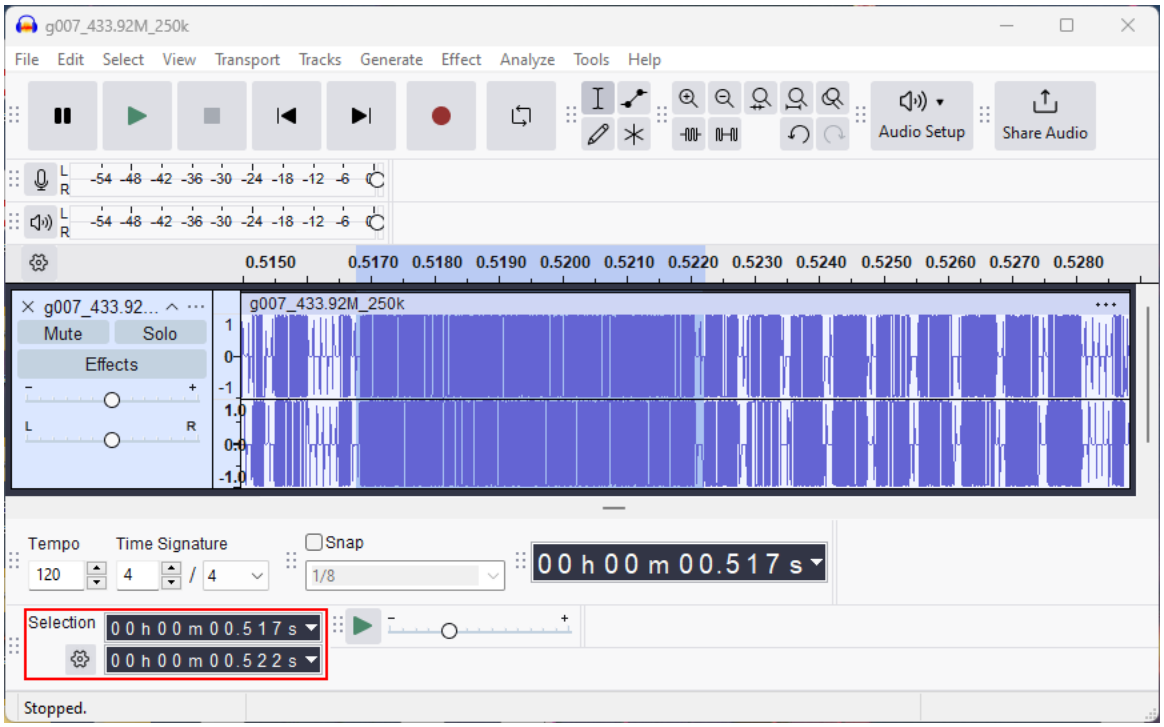


Figure 39. Highlighting the gap between transmissions from the waveform shows the start and end of the selection.

Figure 39 shows the waveform closer and reveals the 5 millisecond, or 5000 microsecond, reset part of the transmission with a short pulse and long off gap. Between two of these resets the actual data was found for decoding. Highlighted in red, the selection section shows the start and end time of the selected part of the waveform. Audacity can show the time in milliseconds but can require settings to be changed to enable this level of detail.

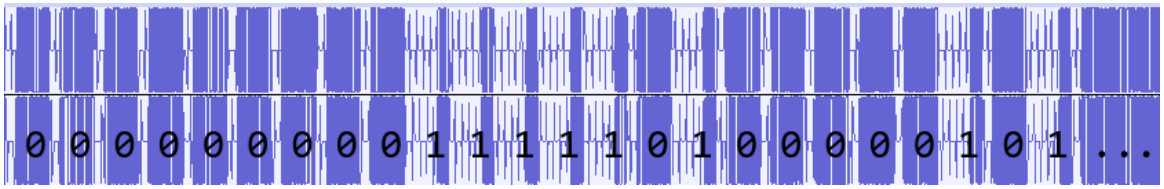


Figure 40. Waveform overlaid with binary values based on OOK and PWM pulses.

Figure 40 zooms into the data package part of the waveform where the binary data can be decoded from. When the signal is transmitted, a steady wave is visible in the waveform.

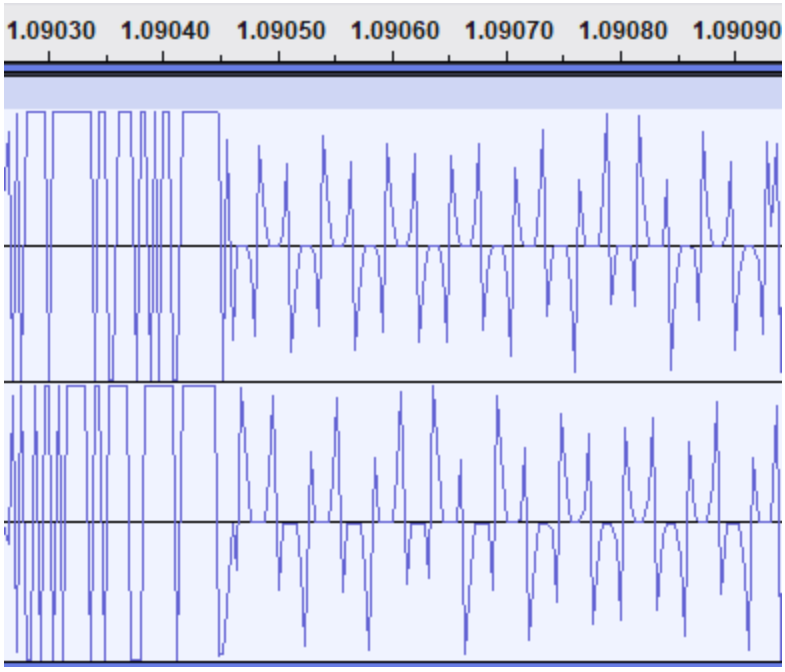


Figure 41. Waveform starts with noise transforming into steady waves.

Figure 41 zooms even closer to the waveform, where on the left, the noise changes into a steady wave when the pulse starts. When the signal was completely off, only noise was captured filling the gap. Manual gain control was

tested but it did not affect the noise levels significantly since the noise was strong. The gain was controlled using the -g <gain> argument.

From the previous examples it is known that short bursts are interpreted as binary values of 0 and long bursts with short off gaps are values of 1. As overlaid on Figure 40, the result was the same doorbell key bit string:

```
Binary 0000 0000 0111 1101 0000 0101
Hexadecimal 007D05
```

10 ROLLING CODES

Rolling code protocols have a key that changes between transmissions. The key is synchronized with a receiver that should not accept out of sequence keys.

Figure 42. KeeLoq protocol compatible remote (left) and the circuit board removed from inside the remote (right).

To demonstrate an implementation of a rolling code protocol, five button presses from both buttons were captured from a KeeLoq protocol compatible remote seen in Figure 42:

Top button

```
93 22 57 74 37 4F 00 08
51 5E 3E D2 37 4F 00 08
75 FC 9B 92 37 4F 00 08
52 0F 00 A8 37 4F 00 08
24 F2 F6 D9 37 4F 00 08
```

Bottom button

```
E3 40 C5 42 37 4F 00 02
6D 32 CD 7F 37 4F 00 02
96 46 04 0B 37 4F 00 02
3E 21 0F 73 37 4F 00 02
1F 4D 0A 2F 37 4F 00 02
```

The captured keys show repeating patterns. For both buttons, the last eight hexadecimal values stay the same. Those are the last 32 bits that are transmitted from the 64-bit KeeLoq protocol. Comparing the two buttons, the last 4 bits change depending on the button being pressed. It is assumed that this is the identification for the button:

Top button

```
93 22 57 74 37 4F 00 08
Hexadecimal 08
Bits 0000 1000
```

Bottom button

```
E3 40 C5 42 37 4F 00 02
Hexadecimal 02
Bits 0000 0010
```

Looking at captures from both buttons, the 28 bits before the last 4 bits remain unchanged. Ignoring the 12 bits that have the value of 0, the 16 bits before the last 16 bits retain the hexadecimal values of 37 4F. This value is likely used for identifying the remote on the receiver:

Top button

```
51 5E 3E D2 37 4F 00 08
75 FC 9B 92 37 4F 00 08
```

Bottom button

```
6D 32 CD 7F 37 4F 00 02
96 46 04 0B 37 4F 00 02
```

It is hard to say if the 12 bits with the value of 0 are used for anything or if they are just unused bits:

Top button

```
51 5E 3E D2 37 4F 00 08
75 FC 9B 92 37 4F 00 08
```

Bottom button

```
6D 32 CD 7F 37 4F 00 02
96 46 04 0B 37 4F 00 02
```

To prove how many codes exist, a key fob can be rigged for automatic presses and then all the codes are captured. This has been demonstrated by Jamison in a video where they wrote code for the Flipper Zero to automate capturing keys from a key fob using the GPIO pins on the device. The result was a text file with 65536 codes from the key fob they were using, which can be used to brute force the receiver to open a garage door. (Jamison 2023b.)

11 TESTING A GARAGE DOOR RECEIVER

Generic wireless receivers for garage door openers are available from various online marketplaces. For testing, a receiver with support for multiple different garage door opener brands was purchased for approximately 29 euros from Amazon. It claims to support multiple different brands using 433.92 MHz and 868.36 MHz fixed and rolling code implementations.

Figure 43. A generic wireless receiver for a garage door opener.

The receiver is shown on Figure 43, where two buttons are visible for synchronizing a key fob to two different channels labeled S1 and S2. Two LEDs, labeled LED1 and LED2, are there for indicating when a channel is activated. The device needs to be paired up with a key fob to operate it.

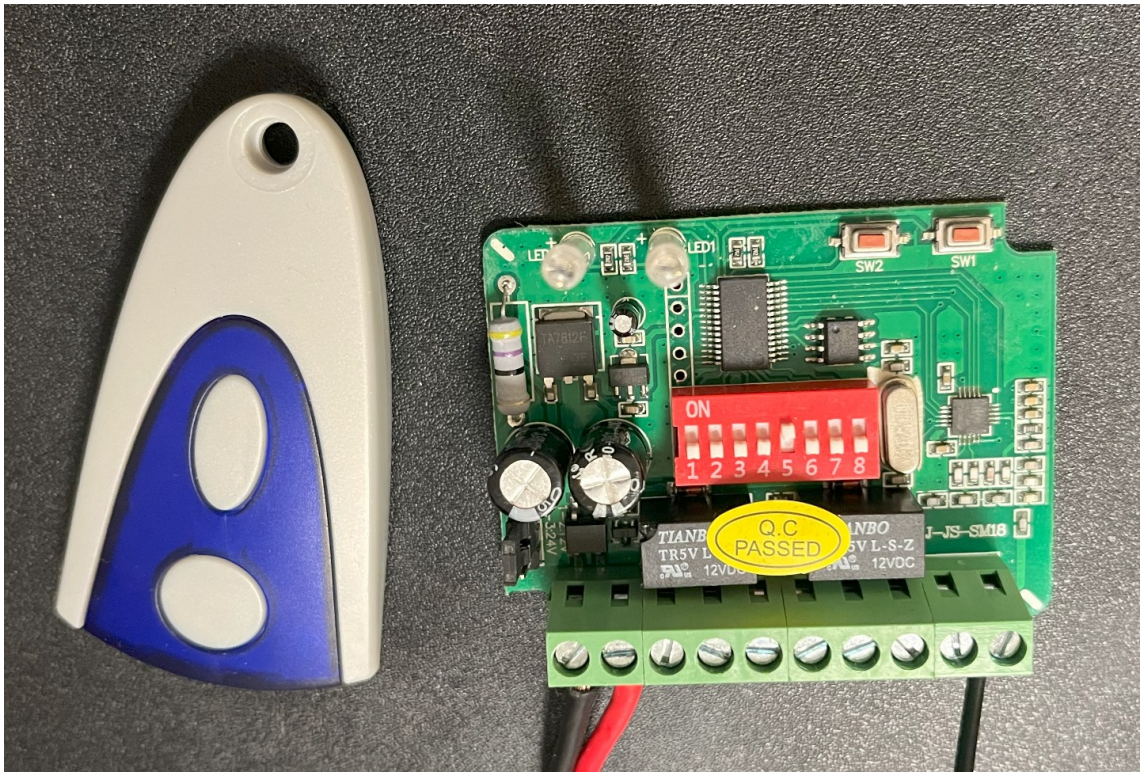


Figure 44. Novoferm receiver compatible garage door key fob and PCB of a wireless receiver.

Two unofficial clones of Novoferm brand key fobs were purchased from Amazon for approximately 23 euros total, one of them shown in Figure 44 on the left. They are going to be paired up with the receiver to operate it. The receiver was disassembled by removing a couple of screws from the bottom of the case. The PCB was then removed from the case. On the right side in Figure 44, dip switches on the PCB were used for setting up the supported manufacturer. For universal rolling code support, the number 5 switch was moved to on position while the other switches were left to off position. This was specified in the manual shipped with the product. The wireless receiver can be powered with 12 or 24 volts, which is selected using a jumper on the PCB.

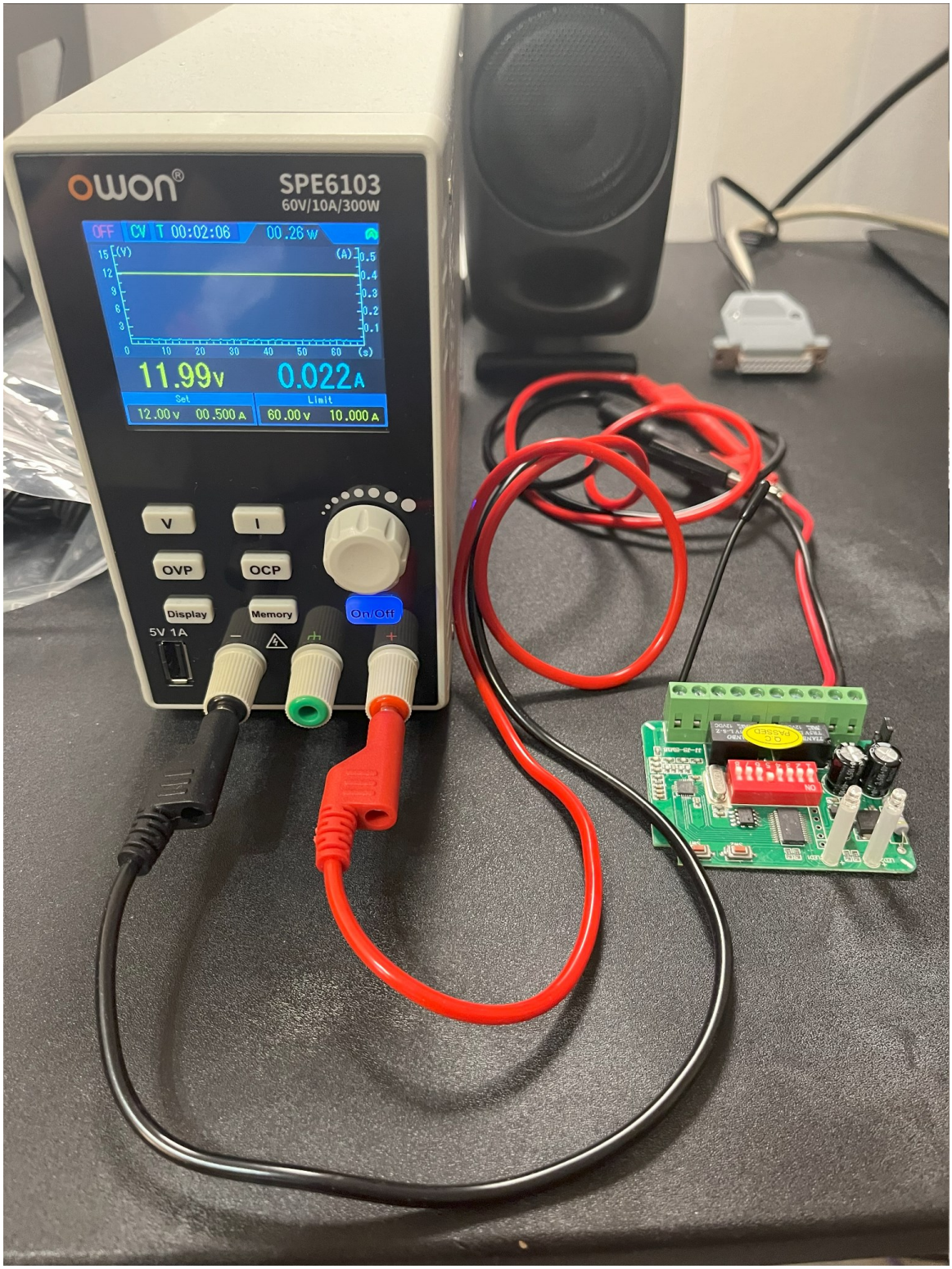


Figure 45. Wireless receiver connected to power using a power supply.

The jumper was left to the default 12V position, and the receiver was connected to a power supply by attaching wires to the terminals on the PCB labeled power,

as shown in Figure 45. The power supply was set for 12-volt output and the current was limited to 0.5 amperes, which is plenty more than what is needed.

The buttons for learning a key fob are labeled SW1 and SW2 on the PCB silkscreen. SW1 was pressed until the LED1 was lit, then the round button on the key fob was pressed. The LED flashes 4 times before going off to indicate it was synchronized. Same was repeated for button SW2 and oval shaped button on the key fob. After synchronization, the round button on the key fob was pressed and LED1 lit up and a click from a relay was audible. The device sends a signal through channel 1 terminals that are normally connected to the garage door opener motor. When the oval button on the key fob was pressed the same happened for LED2 and channel 2. The sub-GHz read function on the Flipper Zero was used to capture a key transmitted from the key fob.

Figure 46. Replaying rolling codes from the Flipper Zero on the 433.92 MHz band.

Figure 46 demonstrates that the Flipper Zero detected the key as a KeeLoq 64bit protocol on the 433.92 MHz band using amplitude modulation. The key was replayed by accessing it from the list of captured keys and pressing the round middle button. Sending the same key to the receiver worked and it was sent a total of 15 times, and it was accepted by the receiver every time. Normal rolling code implementations accept the code usually once or, to reduce the risk of desynchronization, a couple times. At this stage, it looks like the receiver is not enforcing any code expiry.

The captured keys were saved on the Flipper Zero's memory and downloaded onto a PC using the qFlipper application. Contents of a file were viewed and edited on a text editor:

```
Filetype: Flipper SubGhz Key File
Version: 1
Frequency: 433920000
Preset: FuriHalSubGhzPresetOok650Async
Protocol: KeeLoq
Bit: 64
Key: 55 98 F7 F9 37 4F 00 00
Manufacture: Unknown
```

The first 32 bits of the key were changing between the button presses of the key fob, so it was assumed that this is the rolling code. The next 28 bits stayed the same between the button presses and the last 4 bits changed when the other button on the key fob was pressed, which meant that the last 4 bits were used to determine the button. Files were then created using the following keys:

```
Key: FF FF FF FF 37 4F 00 08
```

```
Key: 00 00 00 00 37 4F 00 08
```

```
Key: 12 34 56 78 37 4F 00 08
```

The first 32 bits that were determined to be the rolling code were changed fully to ones in one file resulting in eight Fs in hexadecimal. In the second file they were filled with zeros, which is presented by eight 0s in hexadecimal. In the third file they were filled with a sequence from 1 to 8 in hexadecimal. With this, the maximum and minimum values for the rolling code will be tested. The additional value is to test if an out of sequence code will be accepted. The new files were transferred to Flipper Zero using the qFlipper application and sent from the menu for saved keys in the sub-GHz functions menu. The receiver accepted all three keys multiple times, just like the key that was captured.

Another file was created with the 08 hexadecimal value for the button identification changed to 02, which is the one for the oval button. It is to test if the same key works for the other button:

```
Key: 55 98 F7 F9 37 4F 00 02
```

When played, the modified key activated the LED2 and the relay. Additional files were created with different values, but they did not activate the receiver:

```
Key: 55 98 F7 F9 37 4F 00 00
```

```
Key: 55 98 F7 F9 37 4F 00 01
```

```
Key: 55 98 F7 F9 37 4F 00 06
```

To summarize, the receiver does not enforce the rolling code expiry and seems to accept any value for the key. It uses the static part of the key to identify the key fob it talks to. The last 4 bits are used to identify the button which was enforced.

12 INTERFERENCE AND JAMMING

Jamming communications between sub-GHz devices is achieved by sending random data on the same frequency. If the jamming signal is near and strong enough, the receiver is unable to distinguish actual transmissions. If the jamming signal is not strong enough, it will be considered noise since the actual transmission received will be much stronger. How effectively the receiver handles any noise depends on the implementation and the modulation used. For example, amplitude modulation implementations are more sensitive to noise over frequency modulation. Jamming for amplitude modulation happens when the receiver is receiving a strong signal at least most of the time and cannot distinguish the changes in amplitude. This might not always be a malicious actor trying to jam the communications but can be a malfunctioning device or other devices operating on the same frequency overlapping the data transfer.

Using Flipper Zero it is possible to make a custom sub file by editing a captured signal and changing the values to send random pulses. Using the 650 kHz bandwidth OOK preset for maximum bandwidth and the RAW protocol to start building a file:

```
Preset: FuriHalSubGhzPresetOok650Async
Protocol: RAW
```

Then adding long OOK bursts with short off timing to minimize the silence will make the receiver actively receive a signal. For example, 30000 microsecond burst and 250 microsecond silence repeated over as long as we need:

```
RAW_Data: 30000 -250 30000 -250 30000 -250 30000 -250 30000
-250 ...
```

Sending this near the receiver is enough to overpower real communication if it is stronger than the signal from the transmitter. Realistically Flipper Zero can only jam very near the receiver since the transceiver on it is specified for up to 50

meters in a perfect environment. The signal strength is dropped by distance and obstacles meaning that the signal strength at maximum distance is unlikely to overpower a receiver.

SDR software with a waterfall display is helpful for detecting when there is signal overlap as it visually shows the detected signal amplitude. The SDR# software on Windows 11 was used to view the transmitted signal.

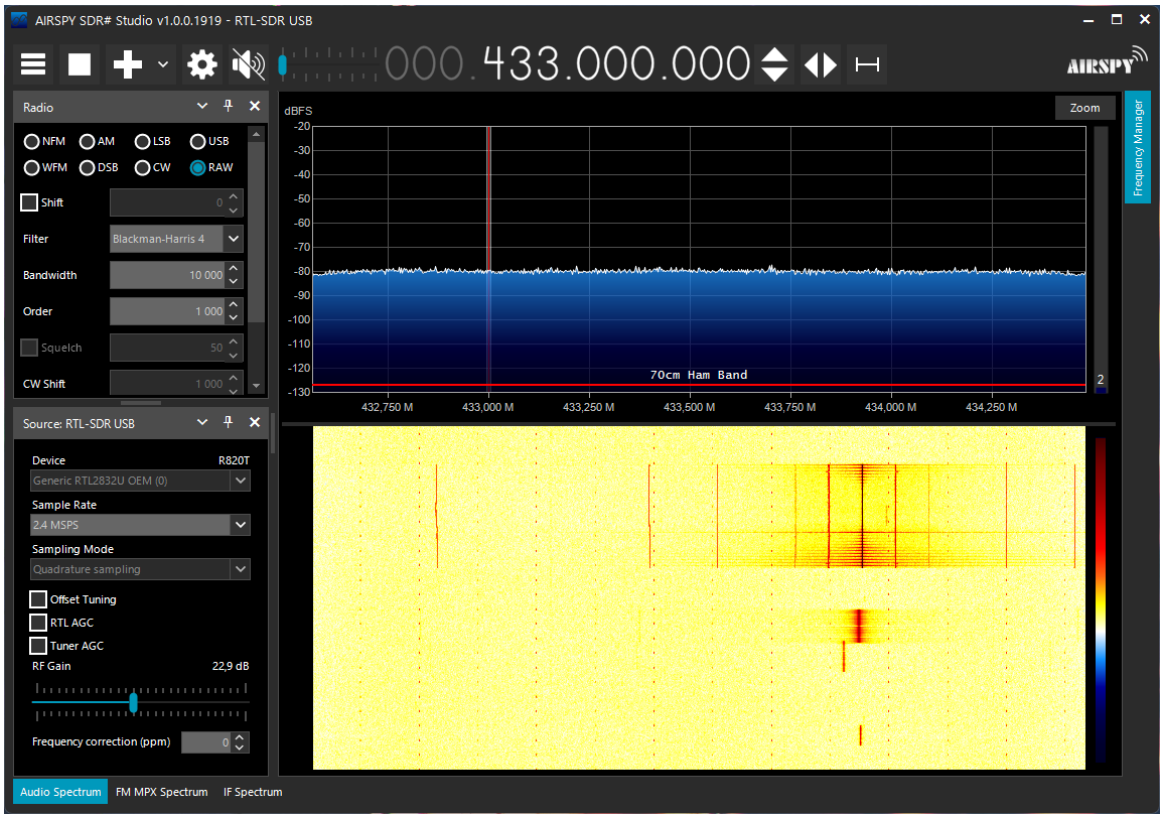


Figure 47. A small burst of data transmitted using the Flipper Zero was witnessed on the waterfall of SDR# software.

Figure 47 shows that the software has a waterfall on the bottom of the screen showing amplitude of received signals. The waterfall moves from top to bottom as indicated by the name. It means that the most recently detected signal strength is on the top. It shows values for frequencies within the bandwidth of the SDR device. The frequency is dialed using the numbers on the top bar in hertz. The top display is the radio frequency spectrum that shows signal amplitude in real time. The source section on the bottom left of the screen shows that the RTL-SDR USB is used. It has adjustments for gain and options to toggle on automatic

gain control (AGC) for adjusting the gain automatically based on noise and signal strength.

Figure 47 shows a transmission of the created RAW sub file on the SDR# waterfall. A solid line was seen on the top part of the waterfall. It indicates that there was a constant stream of transmission on around 433.92 MHz frequency. The line leaks onto other frequencies and causes some saturation. With the Flipper Zero placed approximately 3 meters from a doorbell receiver, it was enough to overpower the doorbell transmitter that was approximately 6 meters away from the receiver behind a wall. In the real world, if someone wants to cause harm by jamming from a distance, they will use a more powerful transmitter to get a better reach. Below the jamming signal, the software shows three normal data bursts from devices.

Another approach to jamming is spoofing values of an unencrypted transmission and constantly overwriting the actual values. This can be used for overwriting values of an HVAC system or industrial devices, for example. Testing for this method was not conducted because a suitable device was not available.

13 TRACKING AND SURVEILLANCE

Tracking unencrypted data transmissions will be investigated using documentation, source code and knowledge learned from the tests before. The key point will be that unencrypted transmissions can be captured, decoded and logged into a database with timestamps.

13.1 Tire Pressure Monitoring System

Tire pressure monitoring system (TPMS) can expose information when a car is in the vicinity of a receiver. Multiple common car manufacturers are covered by the rtl_433 software, which can decode the transmissions from these sensors. More vehicle manufacturers can be added but require reverse engineering the data being sent by sensors and building a decoder. The communications have

potential to be used for surveillance and tracking because the sensors have an ID that synchronizes them to a specific vehicle. Tracking the ID using an automated setup is not difficult as the data is easily logged and it is even possible to make a system that creates an entry when an ID for a specific vehicle is detected. The wiki page for the rtl_433 project has example scripts for data logging to a file or database (Rtl_433 project 2024). For example, the rtl_433 output can be set to output in JSON format that is then parsed and imported into a database using a script.

Since rtl_433 is an open-source program, the source code can be accessed from the project page. The values rtl_433 can decode from Ford TPMS were found in the source code of the program.

```
data_t *data = data_make(
     "model",            "",            DATA_STRING, "Ford",
     "type",             "",            DATA_STRING, "TPMS",
     "id",               "",            DATA_STRING, id_str,
     "pressure_PSI",     "Pressure",    DATA_FORMAT, "%.2f
PSI", DATA_DOUBLE, pressure_psi,
     "temperature_C",    "Temperature",  DATA_COND,
temperature_valid, DATA_FORMAT, "%.1f C",   DATA_DOUBLE,
(float)temperature_c,
     "moving",           "Moving",      DATA_INT, moving,
     "learn",            "Learn",       DATA_INT, learn,
     "code",             "",            DATA_STRING,
code_str,
     "unknown",          "",            DATA_STRING,
unknown_str,
     "unknown_3",        "",            DATA_STRING,
unknown_3_str,
```

```
     "mic",                "Integrity",   DATA_STRING,
"CHECKSUM",
     NULL);
```

Code 1. Values decoded by rtl_433 for Ford TPMS from the rtl_433 source code (Zuckschwerdt 2017).

Code 1 is a snippet from the source code where the fields for the values are declared (Zuckschwerdt 2017). The significant fields here are model, id, moving and temperature. The model field helps to tie the transmission to a specific brand of vehicle and the id value to a specific vehicle. These fields can be used to monitor when a specific vehicle has been in the vicinity of the receiver. Building upon this, the moving value will show if the vehicle was moving when the transmission happened. In addition, clever use of the temperature values helps identify when a vehicle is arriving as opposed to leaving since the tire temperatures will rise when it is moving.

13.2 Security systems

Sub-GHz is used in some security systems when communicating between the devices. It is favored due to its power efficiency and capability to penetrate walls better than devices operating above 1 GHz. Security system communications without encryption provide potential to monitor access within the system’s operating perimeter.

Security systems by Chuango Security Technology Corporation use 433.92 MHz or 315 MHz frequency band for communicating between the hub and different devices (Chuango Security Technology Corporation 2024a). They produce devices for do-it-yourself security systems and home automation applications (Chuango Security Technology Corporation 2024b). Devices manufactured by them can be decoded using rtl_433. The source code lists some devices as tested including touch alarm system, door sensors, wireless keypad, passive infrared sensor (PIR), remote control, smoke and water sensors. It also defines the commands it can decode. (Vestermark 2015.)

```
switch (cmd) {
case 0xF: cmd_str = "?"; break;
case 0xE: cmd_str = "?"; break;
case 0xD: cmd_str = "Low Battery"; break;
case 0xC: cmd_str = "Closing"; break;
case 0xB: cmd_str = "24H Zone"; break;
case 0xA: cmd_str = "Single Delay Zone"; break;
case 0x9: cmd_str = "?"; break;
case 0x8: cmd_str = "Arm"; break;
case 0x7: cmd_str = "Normal Zone"; break;
case 0x6: cmd_str = "Home Mode Zone"; break;
case 0x5: cmd_str = "On"; break;
case 0x4: cmd_str = "Home Mode"; break;
case 0x3: cmd_str = "Tamper"; break;
case 0x2: cmd_str = "Alarm"; break;
case 0x1: cmd_str = "Disarm"; break;
case 0x0: cmd_str = "Test"; break;
default: cmd_str = ""; break;
}
```

Code 2. Supported Chuango commands from rtl_433 source code (Vestermark 2015).

Code 2 is a snippet of where they are defined (Vestermark 2015). A manual for the Chungo LTE-400 WiFi and cellular smart home alarm system is available online. It was used to understand what the different commands do. The devices are set to be armed when the space being monitored is unoccupied and this is when the device can trigger alarms. When the system is disarmed, the alarms are deactivated. Home mode disarms any sensors set to the home zone. Devices set to the 24-hour zone will stay armed regardless of mode the system is in. By default, any fire, smoke and water sensors will not be disarmed. The delay zone is the same as the default arm zone, but the user has an option to set a delay for

the alarm to be triggered. (Chuango Security Technology Corporation 2021.) Table 1 below visualizes the armed zones in home mode:

Table 1. The zones that are armed when set in home mode (Chuango Security Technology Corporation 2021).

Home Mode	Arm Zone	Home Zone	24-Hour Zone	Delay Zone
Armed	x		x	(delayed)
Disarmed		x		

Some of the devices have a tamper switch on the back that will trigger when it is removed from the surface it is attached to, for example, the door sensors. The PIR motion detectors have a tamper switch, which triggers when the cover of the device is removed from the housing. (Chuango Security Technology Corporation 2021.)

For monitoring access, the arm, disarm and home mode commands are the most useful ones. Capturing the commands using rtl_433 with timestamps and building a dataset makes it easy to draw conclusions about habits of access to a perimeter. For example, if a vulnerable system like this was used by a business, a dataset would be built first before an unauthorized person attempted to access it. Based on the gathered data to know when the alarm is usually armed and monitoring when the user forgets to arm the alarm, they would be able to access the perimeter without triggering the security system.

There is a possibility that this type of system is vulnerable to jam and replay attacks depending on if any kind of expiration for the codes has been implemented. To make sure the system is not vulnerable, it requires testing with a device and software capable of performing the jam and replay attack. Since it is possible to deduce from the documentation that the device has a tamper switch, there is potential to use a jammer to block the tamper signal when the device is removed until the battery is unplugged. The manual did not specify how the device behaves when the battery runs out and only mentions that it will blink three times when it is low (Chuango Security Technology Corporation 2021).

Real world testing is required to confirm how it behaves, but this shows that documentation is a powerful reconnaissance tool.

13.3 Utilities

Itron has developed a protocol called encoder receiver transmitter (ERT) that uses automatic meter reading (AMR) systems for radio frequency communications to read and monitor different utilities such as electricity, gas and water consumption. The AMR system can be configured to be read using a fixed reader for a larger geographic area while utilizing repeaters to expand the area. Another option is to use mobile readers such as a vehicle or handheld readers to collect data when moved near the proximity of the devices. Its patent does not describe any encryption for obfuscating the data transmission, which means the values are readable if the packet formatting is understood. (Cornwall et al. 2007.)

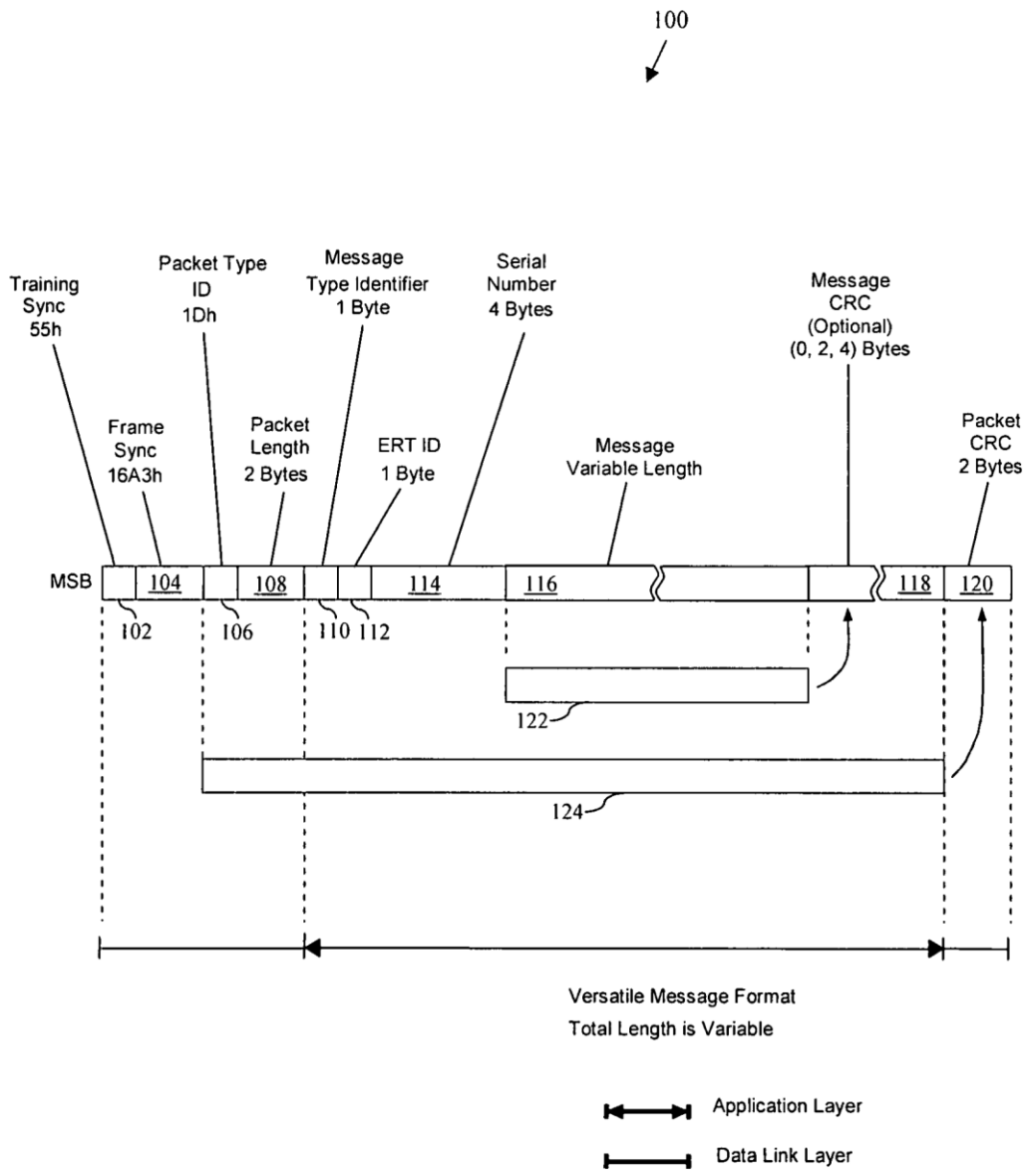


Figure 48. Packet structure for a generic ERT protocol implementation (Cornwall et al. 2007).

In Figure 48 from the patent, an implementation of a packet structure for the ERT protocol is demonstrated. It shows that 4 bytes are used for the serial number and a message of variable length is used. (Cornwall et al. 2007.) The serial number is used for identifying specific meters and if the values from the message are decoded, the usage can be monitored.

Project called rtlamr on GitHub has an implementation for using rtl-sdr to read ERT protocol packages transmitted from utility meters. (Rtlamr project 2024.) In

addition, an implementation on the rtl_433 project for decoding the protocol is included (Rtl_433 project 2024). These implementations demonstrate that it is possible to collect data from ERT transmissions. The protocol uses serial numbers as identifiers, which means the data can be mass collected and tied to specific customer identifiers. This data can then be used to analyze behavioral and usage patterns of the customer location, for example. It might be hard to tell where the transmissions are coming from, but signal strength can be used to make approximations. Triangulation or other radio direction finding methods are an option to pinpoint more specific locations.

14 RESULTS

In this section the results will be looked at from the point of view of the research questions. The gathered research material will be condensed to answer the questions.

14.1 The devices communicating on sub-GHz frequencies

Devices considered sub-GHz typically operate on the ISM frequency bands. They send data periodically or when triggered externally, and these data transmissions happen in small bursts. These devices usually have a low power consumption compared to more actively transmitting devices, which makes them suited for low power applications. Typical use cases include measurement systems, such as utility meters and TPMS, and security systems, such as key fobs, remotes and alarm systems.

14.2 The equipment and software for analyzing sub-GHz devices

An RTL-SDR dongle was effective for analyzing captured traffic and was affordable. A simple antenna was built with minimal tools including wire cutters and a soldering iron. Materials needed were wire, solder and an SMA connector. An SMA cable can be bought separately but one is provided in the RTL-SDR bundle to connect an antenna to the dongle. A simple antenna was easy to model using the free version of the MMANA-GAL antenna modeling software to

visualize radiation patterns. NanoVNA was used to measure the performance of the antenna. Off-the-shelf antennas are available and are a good option if tools and materials are not available.

Free software was available for use with the RTL-SDR driver and provided multiple tools to analyze captured signals. Helpful visualization tools and methods were found in addition to terminal tools.

The Flipper Zero was used in a similar fashion to analyze packets from RAW captures. The benefit of the device was the ability to send within legal power limits. It worked for test rolling code implementation of a receiver used for garage door openers. The ability to send packets made it possible to understand the limitations of the implementation and find vulnerabilities in it.

14.3 Identifying the types of vulnerabilities on sub-GHz devices

Common vulnerabilities for sub-GHz devices are replay attacks and jamming. If the codes sent by devices do not have sufficient protections in place, such as proper rolling code implementation and encryption, they are vulnerable to replay attacks. When the captured code was static, the device was always vulnerable to a replay attack. Identifying non-expiring rolling codes was achieved using the Flipper Zero’s replay feature utilizing captured and modified codes. Identifying that rolling code was used by a device was possible without a device capable of transmitting, but to confirm that the code expiry was enforced by the receiver, a transmitting device was required.

Devices are vulnerable to jamming when an attacker can send an overpowering signal to the receiver. Some implementations are less susceptible to jamming as they can handle more noise, for example, frequency shift keying as opposed to amplitude shift keying. Jam and replay attacks are less effective when time to live for the keys is implemented.

Different security and measurement systems can be used for surveillance when their data is not encrypted. It is possible to log captured data to a database using software, and track performed actions or measurements from devices. Understanding unknown protocols requires reverse engineering and knowledge of different modulation techniques. It was demonstrated that documentation, such as manuals, source code and patents, can be used to understand how a device operates.

15 DISCUSSION

The work succeeded in the main goal of introducing affordable techniques to analyze sub-GHz traffic. Theoretical and practical material to answer the research questions was sufficiently gathered, but since the range of devices is broad, gaps in technologies covered exist. The work focused on analyzing devices that use amplitude modulation because devices using frequency or phase modulation were not available for testing. This left some open questions about how the devices used for testing interact with these modulation techniques. If this was considered in the early phases, devices that use other modulation techniques would have been looked for more actively. Then again, it is hard to tell the type of modulation used before a device is acquired and analyzed. Based on theory and device support, they can be analyzed using the same or similar techniques if the technical differences are considered. Since the analysis begins with understanding the techniques used, it will be automatically considered.

Understanding of modulation techniques was built for decoding captured transmissions. It was crucial for the success of the packet analysis with accurate results. Based on theory, it was expected that unencrypted traffic would be available for reading and decoding, which was proven true using the demonstrated techniques.

The results of decoded captured key codes were confirmed using multiple methods and software using RTL-SDR and Flipper Zero. Identifying modulation techniques was supported by the theory of the typical methods used to modulate

data into radio frequency signals. The surveillance section of the work was based on the results from the research to decode data and brought an additional angle into the security considerations of unencrypted packets. It left questions for future research since it was based on earlier findings in addition to inspecting documentation and source code without practical testing.

Easy exploitability of the low-effort vulnerabilities raises concerns and shows that more research on sub-GHz cybersecurity should be conducted. As a consumer it is hard to tell whether a device is secure, which makes spreading awareness tough. The general assumption is that modern devices offer better protection, but as demonstrated, cheap variants of devices are sold that are fully exploitable. It would be valuable research to survey sub-GHz applications in infrastructure, industrial and office locations to get an idea how vulnerable they are. Easy exploitability can be especially common for legacy installations, and they should be researched.

16 DEVELOPMENT IDEAS

Multiple ideas for future research were thought up due to the extent of technologies using sub-GHz frequencies for communication. Some of these ideas are clear extensions for the work while others explore new techniques.

16.1 Mitigation

Mitigation was left out of this work and is an obvious area for future research. The methods described in this work can be used to find issues in real world implementations or new equipment that can be tested before installation. Testing production equipment should be done with the points brought up in the research design in mind. After issues are found, mitigation can be researched and a plan for implementing corrective actions can be built. It would be interesting to see what options vulnerable devices have for mitigation other than replacing them with equipment with better security. Another objective could be to see how in critical environments the effectiveness of jamming could be lessened.

16.2 Heat maps using GPS

Exploring signal locating has potential for discovering misbehaving transmitters. Heat maps can be used to identify and locate origins of radio frequency transmissions. Radio signal triangulation as an option can be explored. The use case would be, for example, locating particularly strong signals causing interference with other devices.

A generic GPS receiver dongle can be used to tie the location data in coordinates while logging signal strength using the rtl_433 software. A generic USB GPS navigation module called VK-162 is available. It is supported by the Linux kernel out of the box on Fedora Linux. They are available from Amazon for around 15 to 20 euros (Amazon 2024d). This project requires scripting to pipe data from rtl_433 to combine it with the coordinate entry. The project page has a wiki with examples of how to create integrations (Rtl_433 project 2024). A Python3 GPSD client can be used in the script to poll GPS data (Braam 2017). This data can be combined with something like the Jupyter gmaps plugin to create heat maps in Jupyter notebooks overlaying them on Google Maps views (Bugnion 2016).

16.3 Spoofing

Spoofing captured communications can be exploited to control and manipulate devices. Devices communicating without encrypting the packets are especially vulnerable as they can be captured, manipulated and resent. This requires reverse engineering to understand the modulation and protocol used but with experience and capturing enough data, patterns such as changing fields, can be noticed.

It is possible that some utility measurement systems are vulnerable to spoofing of usage values, but it requires researching specific implementations to confirm. There is a possibility for further research if systems without encryption are still being used in Finland as opposed to modern LoRaWAN implementations, for example.

16.4 Surveillance

Surveillance by logging unencrypted traffic can be achieved, as raised in the work. Building a system that logs traffic and researching how it can be used for surveillance would be another angle to sub-GHz cybersecurity. Making sure that this research is performed within the legal limits could prove to be a challenge but can possibly be overcome by getting permission to research traffic at a specific location and having correct limits to what data will be collected. It would require researching the privacy and data gathering laws.

17 CONCLUSIONS

Sub-GHz devices have typically low power consumption and utilize the ISM frequency bands that have been agreed globally by the ITU member states. The work was successful in showing that it is possible to analyze captured unencrypted traffic between these devices using affordable equipment. RTL-SDR handled capturing signals for analysis but lacked the ability to transmit. Researching some of the vulnerabilities required a device capable of transmitting, which added more cost. The Flipper Zero proved to be reasonably priced when its functionality was taken into consideration. The techniques demonstrated in this work proved that vulnerabilities can be found especially when the traffic is unencrypted. Further research raised concerns about possibilities of surveillance but requires more research on different implementations to understand the seriousness. Research ideas for continuation were introduced as there is a lot more to uncover when it comes to sub-GHz cybersecurity.