A Forensic Tool to Acquire Radio Signals Using Software Defined Radio

M. A. Hannan Bin Azhar(✉) and German Abadia

School of Engineering, Technology and Design, Canterbury Christ Church University, Canterbury, UK
{hannan.azhar,g.abadia173}@canterbury.ac.uk

Abstract. The adoption of radio technologies and wireless devices in our society has been increasing with the time. A wide range of devices use radio communications for sending and receiving data. The increasing number of attack vectors used in the radio field, and wireless technology's use in recent terrorist incidents, make spectrum forensics essential to gathering intelligence, especially while the crime is still unfolding, and the attackers remain at large. When most of the wireless acquisition tools on the market work either on Wi-Fi or Bluetooth protocols, using software defined radio technology or SDR can allow us to capture signals regardless of the protocol or modulation. This paper describes the development of a forensically valid extension to the HackRF toolset which includes a SDR module capable of logging details of files for penetration testing. The tools and methods presented in this paper provide the specification and experimental validation of the SDR technology for forensic investigation of potentially vulnerable wireless devices. The two case studies reported here use radio controls to simulate intruder attacks and walkie-talkies to simulate intelligence gathering during a terrorist attack.

Keywords: SDR · HackRF · Network forensics · Spectrum forensics · Live-forensics · Cybersecurity

1 Introduction

Wireless data communications have evolved rapidly over the last few decades and have become widely used worldwide. The growth of wireless communication devices has led to an increase in misuse of these devices and their security vulnerabilities. It has been reported that everyday wireless devices, such as a garage door with a fixed code, can be brute forced in seconds using a cheap transceiver chip [1]. In a relay attack against Passive Keyless Entry (PKE), attackers can steal high-priced cars using devices built for just 22 dollars [2]. The exploitation of Software Defined Radio (SDR) has been reported as signal disruption, spoofing against the GPS protocol, and record and replay attacks [3]. Even as attackers' techniques continue to evolve, law enforcement and forensic examiners lack the tools and programs necessary to acquire radio spectrum. There has been a great deal of research and guidelines developed regarding the forensic capture and analysis of wireless

J. Garcia-Alfaro et al. (Eds.): SecureComm 2021, LNICST 398, pp. 499–512, 2021.
https://doi.org/10.1007/978-3-030-90019-9_25

traffic in wireless local area networks. Several researchers have addressed technical and legal issues related to wireless forensics [4, 5]. Other researchers have developed forensic models for this type of wireless data acquisition [6]. Applications such as Kismet allow GPS logging in addition to performing passive network monitoring and logging, thus provide forensic examiners and law enforcement agencies the capability of intercepting traffic [7]. The increased attack vectors in the radio field [3] and the lack of security in Internet of Things (IoT) devices have forced the law enforcement and forensics community to find low-cost, flexible, and easy-to-operate spectrum forensics tools.

Software Defined Radio is a low-cost and flexible way to acquire radio signals. With SDR technology, investigators can record and save live samples without knowing the modulation or protocol being used. With the same tool and hardware, an investigator can record the signal from a key-fob, a GPS spoofing attack, a jamming attack, a walkie-talkie transmission, and any other radio transmission within the range and capabilities of SDR. It is possible to use Open-Source programs to monitor, demodulate, and save spectral data, such as the Linux based program GQRX [8]. Universal Radio Hacker (URH) [9] is another open-source program that records the radio spectrum from a variety of SDR devices and saves the data in multiple formats. Despite the fact that different SDR programs have the ability to record radio signals, they all lack forensic capabilities and logging features, making them insufficient for providing actionable evidence.

The remainder of the paper will be organised as follows: Sect. 2 will introduce legal and forensic procedural soundness of spectrum forensics, equipment used in the experiment will be detailed in Sect. 3, Sect. 4 will discuss the developed program, Sect. 5 will report the results in two simulated case studies and finally, Sect. 6 will conclude the paper and mention possible future work.

2 Spectrum Forensics

A sub-discipline of network forensics, wireless forensics involves employing methodologies and tools to intercept and analyse wireless traffic that can be presented as valid digital evidence. Since the data is volatile, special consideration should be given to the ACPO Principal [10], since conducting live forensics requires access to the original evidence. It is imperative that anyone taking part in such activities be competent and aware of the potential consequences. Following are the forensic minimum requirements for a wireless forensic tool to potentially produce a legal result:

- Minimal interaction - the forensic tool should have a minimal interaction with the potential sources of evidence, and if such an interaction is required, its effects must be understood and justified. Passive scanning is more suitable for forensic purposes, since no interaction is needed to collect evidence. This feature is linked with the forensic principle of the Locard's Exchange [11], relating also to Pollitt's work on Digital Forensics Models [12].
- Accuracy and repeatability - implying that the system will produce the same results when used in the same environment. It is also important to know the technical limitations of the procedure. This feature is linked to the scientific principle of ensuring repeatability in experimentation.

- The operation of the tool must comply with all legal requirements. The Wireless Telegraphy Act 2006 [13] and the Communications Act 2003 [14] govern radio use in the UK. UK's Office of Communications (OFCOM) is the government-approved body for spectrum management and regulation. It specifies the services that can be listened to without a license in its guidance on the use of radio scanners [15]. These services are licensed broadcasting stations, amateur and citizens band radio transmissions, weather, and navigation transmissions. The UK Frequency Allocation Table is available on the OFCON page [16]. Spectrum acquisition activities must be accomplished after receiving proper authorisation unless the radio band being utilised is an amateur radio band or other devices operated at short ranges and low power and exempted by regulations made by Ofcom [17]. The Investigatory Powers Act 2016 defines lawful authority for carrying out an interception, as well as the necessary warrant for a targeted or bulk interception [14]. These legislation acts must be followed by investigators in order to ensure that the data interception is legal.

3 Equipment for Experiment

In order to perform forensic acquisition of radio signals, the following hardware devices were used for the experiments:

- HackRF One: SDR half-duplex transceiver capable of receiving and transmitting from 1 MHz to 6 GHz with a maximum bandwidth of 20 MHz and an 8-bit resolution.
- RTL-SDR: Nooelec NESDR SMART v4 capable of receiving from 25 MHz to 1750 MHz, with an internal TCXO and an aluminium enclosure to reduce noise.
- Antenna: ANT500, is a 50 Ω telescopic antenna that covers a frequency range from 75 MHz to 1 GHz. Also, included is a Fixed frequency 433 MHz (ISM) antenna mast from Nooelec [18].
- Aluminium enclosure: Aluminium enclosure for the HackRF to prevent unwanted electromagnetic interferences [19].
- TCXO [20]: A 0.5 PPM external Temperature Compensated Crystal Oscillator for the HackRF One to improve noise performance and avoid frequency drifting.
- USB cable: A shielded USB to MicroUSB cable from HackRF was used along with three ferrite beads.
- GPS receiver: VFAN UG-353 GPS receiver was used to capture the GPS location and UTC timestamps.
- Faraday box: RF shielded environment to avoid legal issues and ensure that the transmitted radio signals are contained and do not interfere with any devices operating in the same frequency band.

4 Radio Signal Acquisition Tool

The tool aims to provide a forensically sound application for acquiring radio signals, allowing a uniform method of configuring different SDR devices. Python 3.8 was used to develop the tool. Interfacing with the SDR devices is done through the open-source library SoapySDR [21]. The library is used to instantiate, configure, and stream the samples of different SDR devices.

4.1 Generated Files

The following files are generated by the application:

- Report file: A detailed report file of the acquisition containing the following
  - Local and UTC time of start of the capture.
  - Session log filename.
  - GPS log filename, when GPS logging is enabled.
  - GPS coordinates, and GPS UTC timestamps of start when a receiver is available.
  - Hardware information of the device such as name, label, serial number or version.
  - The configuration used to capture the samples: Frequency, sample rate, bandwidth, gain of elements and stream format.
  - MD5 and SHA256 hashes of generated samples files. The filenames include the file index, the device name and serial identifier, a time stamp, and with and appendix corresponding to its stream format.
  - Total number of saved samples.
  - Total time of capture.
  - Local and UTC time of the end of the capture.
- Sample files: The captured signal has two components, I (In phase) and Q (Quadrature). I/Q samples are the real and imaginary parts of the digital complex baseband signal. The filenames of the captured files indicate the number of the file, the device id, the date and time, and the stream format used to store the samples. Due to the large amount of data generated when the samples are produced, the files are split and it prevents hashing one big file at the end of the capture.
- GPS file: Timestamped GPS log file indicating the UTC time, latitude, longitude and altitude when available.
- Log file: A timestamp log file contains the date, time, and information level of all logs generated by the GPS, the SoapySDR, and the application itself. The file structure of the capture includes first the log file of the capture, with logs from the GPS connection or the SoapySDR library. The capture report is located in the timestamped directory. Additionally, the directory contains the GPS log file. Inside the output directory are the captured I/Q samples files, and the appendix of each file indicates the stream format of the samples.

4.2 Libraries Used in the Tool

The program makes use of the following libraries:

- soapySDR: is used for interfacing with the SDR devices, instantiating, configuring and streaming the samples from the devices.
- gpsd: is used to get GPS coordinates and UTC timestamps when available.
- logging: is used to log the activity to an external file.
- datetime: is used to get date and time.
- time: is used for the sleep command.

- threading: for the use of threads when hashing the samples files or saving the IQ samples.
- hashlib: is a library for creating MD5 and SHA256 hashes.
- numpy: is a scientific computing library used for special data type arrays of complex numbers for the samples buffers.
- sys: is used to send errors to stderr.
- os: is used to interact with directory structure.
- readline: input line editing, tab completion and command history.
- signal: is used for detecting SIGINT signal and prevent from closing the application.

4.3 Program Console

The program provides a console where the devices can be configured. The different available commands will be described below:

- list: provides a list of the connected devices showing the device identifier and the driver being used.
- info: provides a detailed list of information about the device such as the name, the serial number, label or version. And also, information such as the frequency range, bandwidth rate, possible sample rates, stream formats or configuration specific to the device.
- status: indicates the status of the connected devices and the tuned frequency. If the device is recording samples, it indicates the elapsed time, the number of saved samples, and the total size of the recorded samples.
- configure [id]: is used to enter the configuration menu of a specific device to configure its attributes.
- show [id]: is used to show the current state of the different possible configurable attributes from a device.
- set [setting] [value]: command is used to set the different attributes of the devices.
- start [id]: command starts the radio acquisition once the necessary attributes are set.
- stop [id]: command to stop the radio acquisition of a specific device.

5 Experimental Scenarios and Findings

Two different scenarios were used for acquiring radio signals using the developed tool. Depending on the scenario, radio signals operated at different frequencies and modulation methods. Details of the scenarios and findings are given below:

5.1 Test Case 1 – Walkie Talkie in a Terror Attack

This scenario was based on a simulated case study in which members of a terrorist group were identified during an operation. Due to this, the terrorist group has been confined inside a warehouse with workers of the warehouse being held hostage. Intelligence reports that they are using PMR walkie-talkies to communicate with an external agent in the area. The PMR band covers 460.0–446.2 MHz. Therefore, no further information would be required to capture the signal.

To avoid legal issues in this test scenario, an equivalent FM transmission from a local station was used instead of the PMR446 band. Since the chosen station uses the same modulation to transmit the data, the data capture and analysis would be the same. In this case, the signal was captured from FM radio SER+MADRID, which transmits at 104.30 MHz.

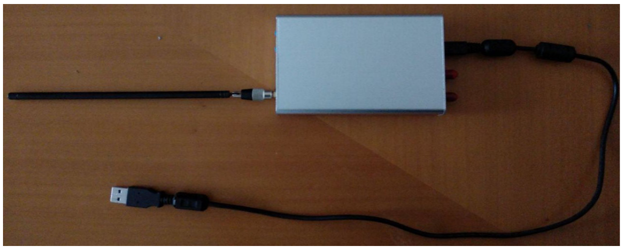


Fig. 1. SDR capture hardware setup.

The hardware setup for the experiment is shown in Fig. 1. The hardware used were HackRF One, Antenna ANT500, GPS receiver, Aluminium enclosure, TCXO and a USB 2.0 shielded cable with ferrite beads. Initially, GQRX [8] was used to locate and identify the signal, providing details of the modulation used and the radio activity. Then, using the developed program, the radio signal was captured in a forensically sound manner.

5.2 Findings in Test Case 1 – Walkie Talkie in a Terror Attack

Figure 2 shows the radio spectrum analyser, showing the tuned FM SER+MADRID radio signal at 104.30 MHz, surrounded by different radio station signals.

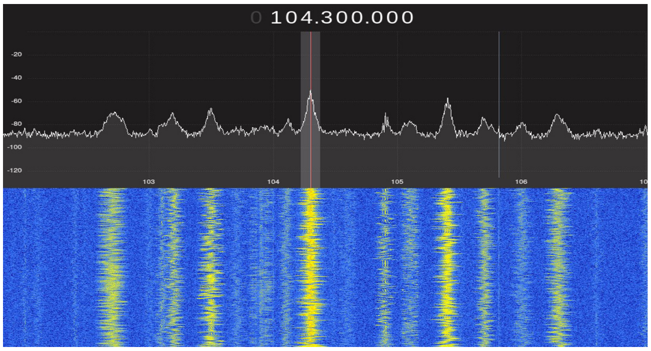
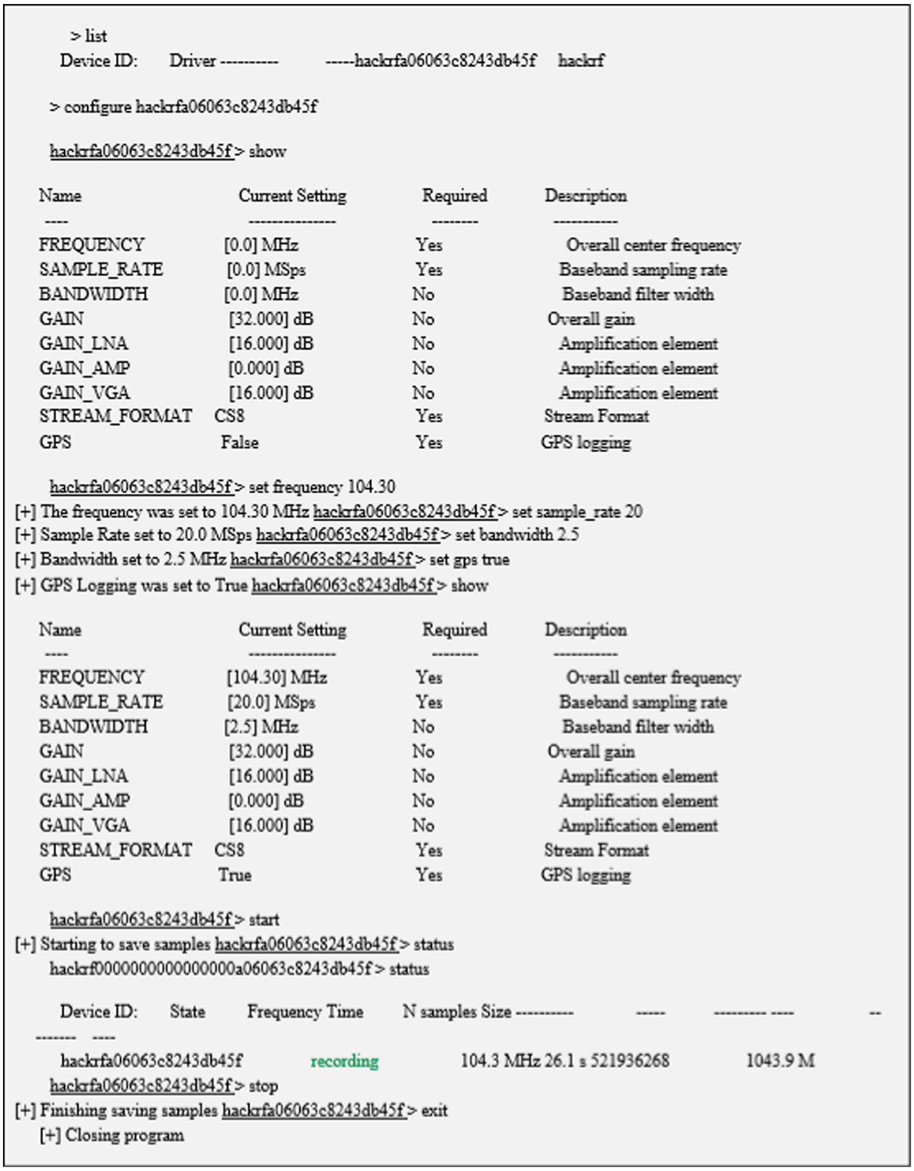


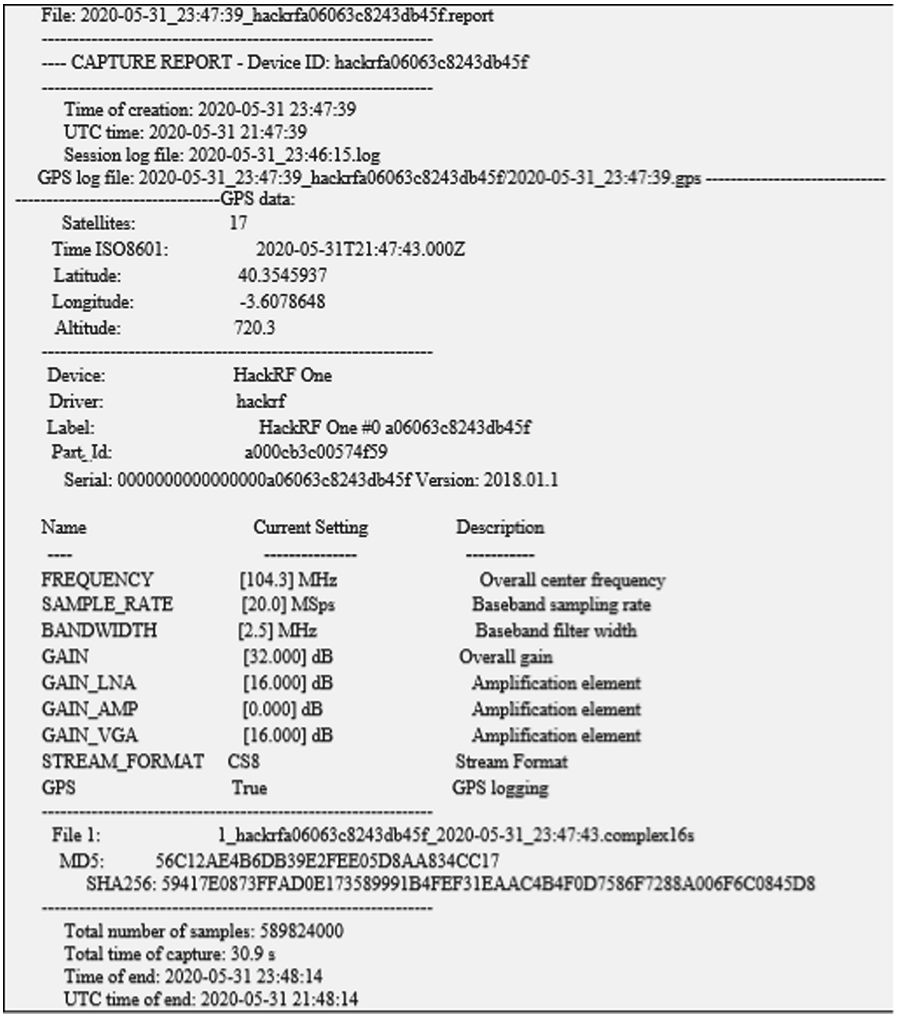
Fig. 2. 104.30 MHz captured by GQRX.

Once the signal was identified, the developed program was used to capture the signal. The sample rate was set to 20 MSPS and the frequency was set to the local FM frequency of 104.30 MHz. To ensure that the surrounding radio transmissions were captured, the bandwidth was set to 2.5 MHz. GPS logging was enabled to log position data to be able to estimate the transmitter location based on the output power. After setting the configuration for the capture (Fig. 3), the start command was executed in the tool to start the acquisition of the samples.



```
      > list
     Device ID:     Driver ----------        -----hackrfa06063c8243db45f    hackrf

    > configure hackrfa06063c8243db45f

    hackrfa06063c8243db45f > show

   Name                      Current Setting          Required        Description
    ----                       ---------------          --------         -----------
   FREQUENCY              [0.0] MHz                 Yes               Overall center frequency
   SAMPLE_RATE            [0.0] MSps                Yes              Baseband sampling rate
   BANDWIDTH              [0.0] MHz                 No               Baseband filter width
   GAIN                    [32.000] dB                No             Overall gain
   GAIN_LNA                [16.000] dB                No               Amplification element
   GAIN_AMP                [0.000] dB                 No               Amplification element
   GAIN_VGA                [16.000] dB                No               Amplification element
   STREAM_FORMAT    CS8                             Yes             Stream Format
   GPS                        False                         Yes             GPS logging

    hackrfa06063c8243db45f > set frequency 104.30
[+] The frequency was set to 104.30 MHz hackrfa06063c8243db45f > set sample_rate 20
[+] Sample Rate set to 20.0 MSps hackrfa06063c8243db45f > set bandwidth 2.5
[+] Bandwidth set to 2.5 MHz hackrfa06063c8243db45f > set gps true
[+] GPS Logging was set to True hackrfa06063c8243db45f > show

   Name                      Current Setting          Required        Description
    ----                       ---------------          --------         -----------
   FREQUENCY              [104.30] MHz              Yes               Overall center frequency
   SAMPLE_RATE            [20.0] MSps               Yes              Baseband sampling rate
   BANDWIDTH              [2.5] MHz                 No               Baseband filter width
   GAIN                    [32.000] dB                No             Overall gain
   GAIN_LNA                [16.000] dB                No               Amplification element
   GAIN_AMP                [0.000] dB                 No               Amplification element
   GAIN_VGA                [16.000] dB                No               Amplification element
   STREAM_FORMAT    CS8                             Yes             Stream Format
   GPS                        True                          Yes             GPS logging

    hackrfa06063c8243db45f > start
[+] Starting to save samples hackrfa06063c8243db45f > status
    hackrf0000000000000000a06063c8243db45f > status

     Device ID:     State       Frequency Time      N samples Size ----------        -----        --------- ----              --
  -------   ----
     hackrfa06063c8243db45f            recording             104.3 MHz 26.1 s 521936268             1043.9 M
    hackrfa06063c8243db45f > stop
[+] Finishing saving samples hackrfa06063c8243db45f > exit
   [+] Closing program
```

Fig. 3. Configuration and capture of signal.



```
File: 2020-05-31_23:47:39_hackrfa06063c8243db45f.report
--------------------------------------------------------------
---- CAPTURE REPORT - Device ID: hackrfa06063c8243db45f
--------------------------------------------------------------
   Time of creation: 2020-05-31 23:47:39
   UTC time: 2020-05-31 21:47:39
   Session log file: 2020-05-31_23:46:15.log
GPS log file: 2020-05-31_23:47:39_hackrfa06063c8243db45f/2020-05-31_23:47:39.gps ----------------------------
---------------------------------GPS data:
   Satellites:              17
  Time ISO8601:                2020-05-31T21:47:43.000Z
  Latitude:                 40.3545937
  Longitude:                -3.6078648
  Altitude:                720.3
--------------------------------------------------------------
  Device:                  HackRF One
  Driver:                  hackrf
  Label:                       HackRF One #0 a06063c8243db45f
   Part_Id:                  a000cb3c00574f59
   Serial: 000000000000000a06063c8243db45f Version: 2018.01.1

Name                    Current Setting              Description
----                    --------------               ----------
FREQUENCY               [104.3] MHz                     Overall center frequency
SAMPLE_RATE             [20.0] MSps                    Baseband sampling rate
BANDWIDTH               [2.5] MHz                      Baseband filter width
GAIN                     [32.000] dB                 Overall gain
GAIN_LNA                 [16.000] dB                   Amplification element
GAIN_AMP                 [0.000] dB                    Amplification element
GAIN_VGA                 [16.000] dB                   Amplification element
STREAM_FORMAT       CS8                          Stream Format
GPS                      True                        GPS logging
--------------------------------------------------------------
  File 1:                 1_hackrfa06063c8243db45f_2020-05-31_23:47:43.complex16s
   MD5:        56C12AE4B6DB39E2FEE05D8AA834CC17
      SHA256: 59417E0873FFAD0E173589991B4FEF31EAAC4B4F0D7586F7288A006F6C0845D8
--------------------------------------------------------------
   Total number of samples: 589824000
   Total time of capture: 30.9 s
   Time of end: 2020-05-31 23:48:14
   UTC time of end: 2020-05-31 21:48:14
```

Fig. 4. Report file contents of the capture.

Figure 4 shows the report file generated, indicating timestamps, the session and GPS log files, the GPS data at the start of the capture, hardware information of the device, its configuration for the capture, followed by the IQ samples file with MD5 and SHA256 hashes, and timestamps at the end of the capture. The log file of the capture was generated by the tool indicating different information related to the capture such as settings applied, GPS connection information, stream format, and time of start and finish of the capture. The GPS log file contained the UTC time received by the GPS, the latitude, longitude, and altitude information.

Universal Radio Hacker [9] was used to analyse the complex IQ file in the format CS8 (complex 8-bit signed integer samples). The IQ file was converted into frequency domain data in this experiment. The captured signal was the FM modulated transmission from

SER+MADRID, surrounded by signals from other local stations. Open-Source tools like Gnu-Radio [22], GQRX [8] or SDR# [23] can be used to process the signal and demodulate it.

5.3 Test Case 2 – Intruder Attack of Garage Door

Reference [1] demonstrated how easy it was to brute-force any fixed-code garage door. The codes sent by the key-fob to the garage are usually modulated using ASK(OOK) modulation. An ASK(OOK) modulated 10-bit garage signal code is shown in Fig. 5.

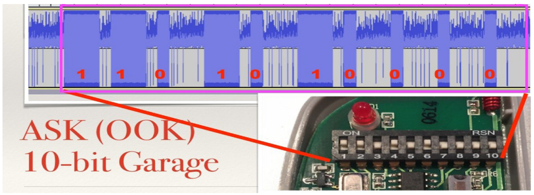


Fig. 5. Garage 10-bit code signal [1].

Reference [1] explained how shift registers decode received codes by sequentially loading each bit into the register, and then pushing the previous bits out as the new bits arrive. The De Brujin Sequence [24] interlaces multiple codes into one sequence to reduce the number of codes needed for transmission. Using the De Brujin Sequence and removing the wait periods between sending each code, for every 8 to 12-bit garage code the brute-force time is 8.2 s. This experiment simulates a real attack by generating and transmitting a De Brujin Sequence of 10 bits in the 433.92 MHz band and capturing the transmission.

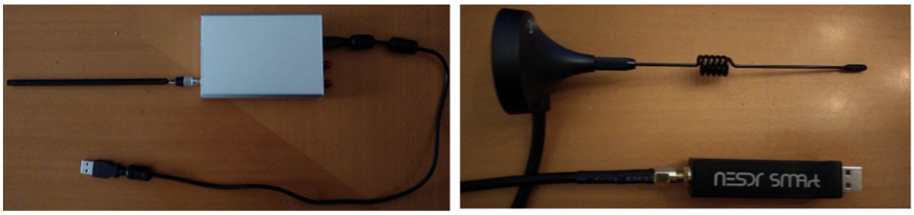


Fig. 6. SDR transmitter setup (left) and SDR capture setup (right).

To perform the experiment, two SDRs were used, one for transmitting the signal and another one for capturing. The hardware used for this experiment were: The HackRF One transmitter to transmit the generated De Brujin Sequence, the RTL-SDR to capture the signals transmitted from the HackRF using the developed program and a fixed frequency antenna for 433 attached to the RTL-SDR using an antenna base. Figure 6 shows the hardware setup for transmitting the data, as well as the RTL-SDR device and a 433 MHz antenna for capturing the data.

For the garage attack experiment to be reproduced legally, it was conducted within an RF shielded environment, using a very low power SDR without the airborne ability [17]. Thus, the transmitted radio signal would be contained within a very short range and not interfere with any other devices operating on the same frequency band. As a result of the RF shielded environment, GPS reception was not possible and therefore, the GPS receiver was not used. The experiment was performed by first generating a 10-bit Brujin Sequence with the ‘rfpwnon’ tool [25]. ASK (Amplitude Shift Key) modulation is needed to transmit this bit sequence. This was accomplished using the tool HackRF_OOK [26]. This program enables the modulation and transmission of OOK modulated data using the HackRF.

```
hackrf_ook -s 0 -b 1700 -0 1284 -1 416 -f 433920000 -r 1 -m 0001...
```

Fig. 7. Command to modulate and transmit OOK data.

Figure 7 shows the command used to modulate and transmit the bit sequence. The ‘−s’ option sets the preamble duration to 0, ‘−b’ option sets the overall bit duration in microseconds, and ‘−0’ and ‘−1’ indicate the gap widths for the 0 and 1 bits, respectively. The ‘−f’ sets the frequency to 433.92 MHz. The ‘−r’ indicates the number of repeated transmissions, and ‘-m’ sets the bits sequence to transmit.

5.4 Findings in Test Case 2 – Intruder Attack of Garage Door

The procedure for acquiring the signal is similar to the previous experiment. Initially, the GQRX program was used to identify the signal. The transmitted sequence was captured with the RTL-SDR using the developed software tool, setting the frequency to 433.92 MHz, and using a sample rate of 2.048 MSPS. The device was unstable and dropped samples with a higher sample rate. Due to the RF shielded environment, GPS reception was not possible. The ASK radio signal transmitted by the HackRF is shown in Fig. 8. Figure 9 shows the configuration of the tool and the signal capture. The generated report of the capture is shown in Fig. 10. Figure 11 shows the content of the captured log file.

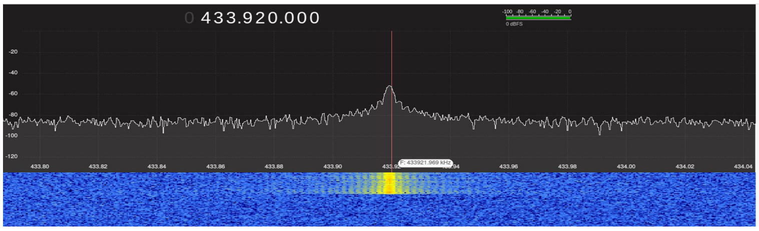
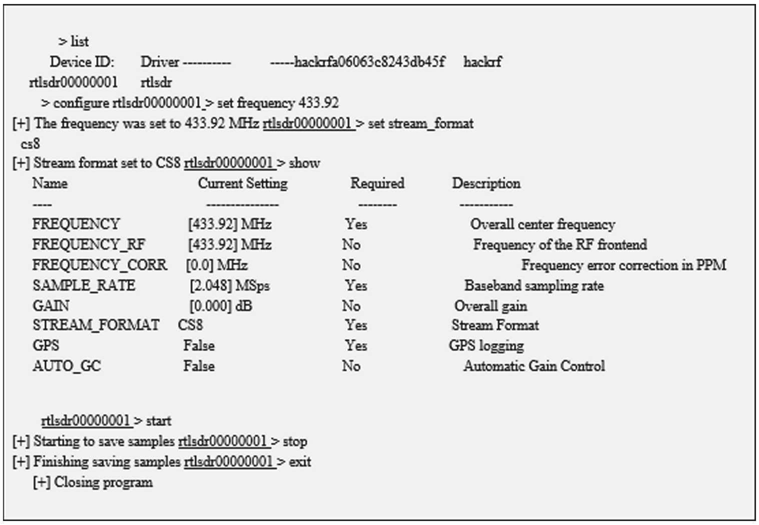
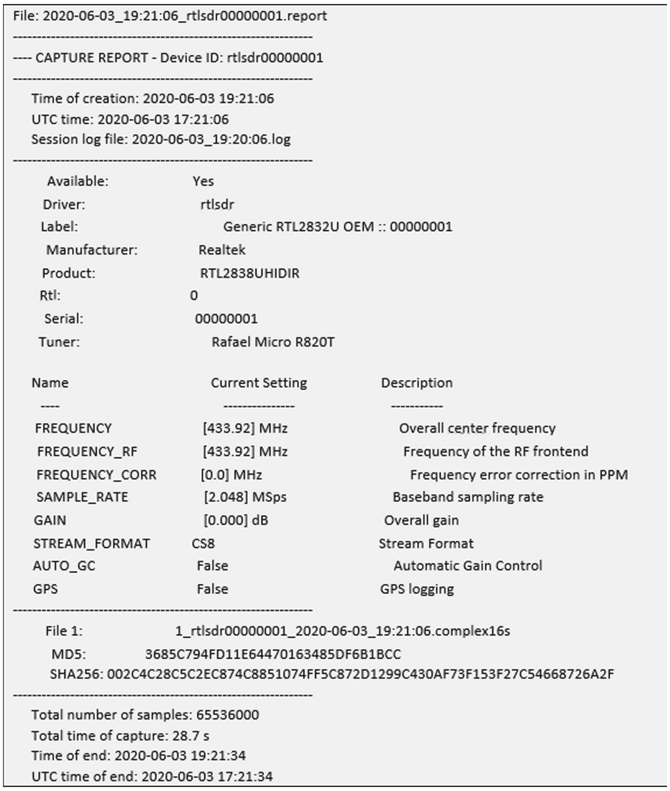


Fig. 8. Signal transmitted detected by GQRX.



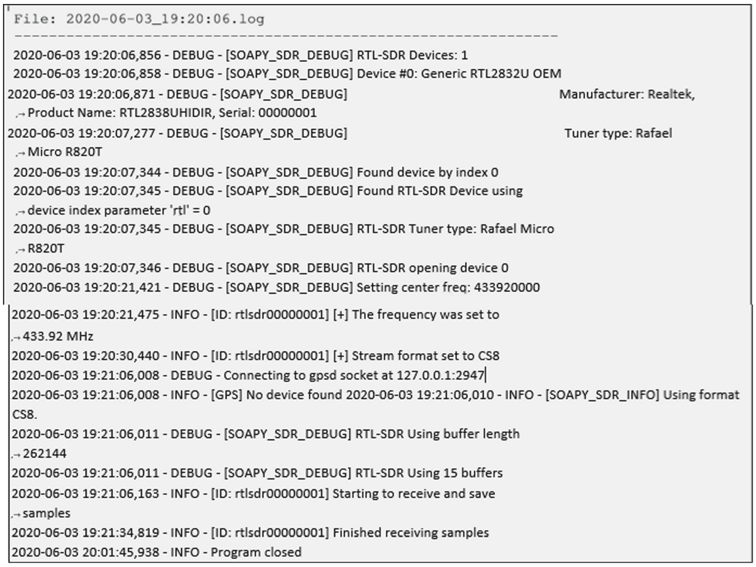
```
   > list
  Device ID:    Driver ---------      -----hackrfa06063c8243db45f    hackrf
rtlsdr00000001    rtlsdr
   > configure rtlsdr00000001 > set frequency 433.92
[+] The frequency was set to 433.92 MHz rtlsdr00000001 > set stream_format
 cs8
[+] Stream format set to CS8 rtlsdr00000001 > show
  Name                 Current Setting        Required       Description
  ----                 ---------------        --------       -----------
  FREQUENCY            [433.92] MHz           Yes            Overall center frequency
  FREQUENCY_RF         [433.92] MHz           No             Frequency of the RF frontend
  FREQUENCY_CORR       [0.0] MHz              No             Frequency error correction in PPM
  SAMPLE_RATE          [2.048] MSps           Yes            Baseband sampling rate
  GAIN                 [0.000] dB             No             Overall gain
  STREAM_FORMAT        CS8                    Yes            Stream Format
  GPS                  False                  Yes            GPS logging
  AUTO_GC              False                  No             Automatic Gain Control


   rtlsdr00000001 > start
[+] Starting to save samples rtlsdr00000001 > stop
[+] Finishing saving samples rtlsdr00000001 > exit
  [+] Closing program
```

Fig. 9. Configuration and capture of the signal.



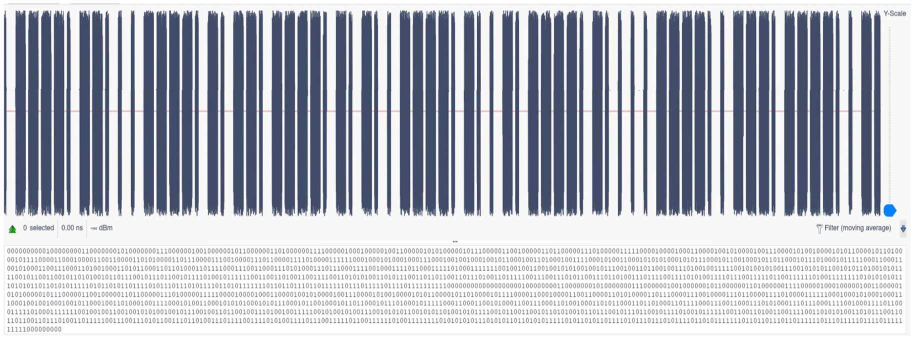
```
File: 2020-06-03_19:21:06_rtlsdr00000001.report
---------------------------------------------------------------
---- CAPTURE REPORT - Device ID: rtlsdr00000001
---------------------------------------------------------------
   Time of creation: 2020-06-03 19:21:06
   UTC time: 2020-06-03 17:21:06
   Session log file: 2020-06-03_19:20:06.log
---------------------------------------------------------------
      Available:          Yes
      Driver:             rtlsdr
      Label:              Generic RTL2832U OEM :: 00000001
      Manufacturer:       Realtek
      Product:            RTL2838UHIDIR
      Rtl:                0
      Serial:             00000001
      Tuner:              Rafael Micro R820T

   Name                  Current Setting        Description
    ----                 ---------------        -----------
    FREQUENCY            [433.92] MHz           Overall center frequency
    FREQUENCY_RF         [433.92] MHz           Frequency of the RF frontend
    FREQUENCY_CORR       [0.0] MHz              Frequency error correction in PPM
    SAMPLE_RATE          [2.048] MSps           Baseband sampling rate
    GAIN                 [0.000] dB             Overall gain
    STREAM_FORMAT        CS8                    Stream Format
    AUTO_GC              False                  Automatic Gain Control
    GPS                  False                  GPS logging
---------------------------------------------------------------
      File 1:          1_rtlsdr00000001_2020-06-03_19:21:06.complex16s
       MD5:         3685C794FD11E64470163485DF6B1BCC
       SHA256: 002C4C28C5C2EC874C8851074FF5C872D1299C430AF73F153F27C54668726A2F
---------------------------------------------------------------
   Total number of samples: 65536000
   Total time of capture: 28.7 s
   Time of end: 2020-06-03 19:21:34
   UTC time of end: 2020-06-03 17:21:34
```

Fig. 10. Capture report.



```
File: 2020-06-03_19:20:06.log
----------------------------------------------------------------
2020-06-03 19:20:06,856 - DEBUG - [SOAPY_SDR_DEBUG] RTL-SDR Devices: 1
2020-06-03 19:20:06,858 - DEBUG - [SOAPY_SDR_DEBUG] Device #0: Generic RTL2832U OEM
2020-06-03 19:20:06,871 - DEBUG - [SOAPY_SDR_DEBUG]                    Manufacturer: Realtek,
↪Product Name: RTL2838UHIDIR, Serial: 00000001
2020-06-03 19:20:07,277 - DEBUG - [SOAPY_SDR_DEBUG]                    Tuner type: Rafael
↪Micro R820T
2020-06-03 19:20:07,344 - DEBUG - [SOAPY_SDR_DEBUG] Found device by index 0
2020-06-03 19:20:07,345 - DEBUG - [SOAPY_SDR_DEBUG] Found RTL-SDR Device using
↪device index parameter 'rtl' = 0
2020-06-03 19:20:07,345 - DEBUG - [SOAPY_SDR_DEBUG] RTL-SDR Tuner type: Rafael Micro
↪R820T
2020-06-03 19:20:07,346 - DEBUG - [SOAPY_SDR_DEBUG] RTL-SDR opening device 0
2020-06-03 19:20:21,421 - DEBUG - [SOAPY_SDR_DEBUG] Setting center freq: 433920000
2020-06-03 19:20:21,475 - INFO - [ID: rtlsdr00000001] [+] The frequency was set to
↪433.92 MHz
2020-06-03 19:20:30,440 - INFO - [ID: rtlsdr00000001] [+] Stream format set to CS8
2020-06-03 19:21:06,008 - DEBUG - Connecting to gpsd socket at 127.0.0.1:2947
2020-06-03 19:21:06,008 - INFO - [GPS] No device found 2020-06-03 19:21:06,010 - INFO - [SOAPY_SDR_INFO] Using format
CS8.
2020-06-03 19:21:06,011 - DEBUG - [SOAPY_SDR_DEBUG] RTL-SDR Using buffer length
↪262144
2020-06-03 19:21:06,011 - DEBUG - [SOAPY_SDR_DEBUG] RTL-SDR Using 15 buffers
2020-06-03 19:21:06,163 - INFO - [ID: rtlsdr00000001] Starting to receive and save
↪samples
2020-06-03 19:21:34,819 - INFO - [ID: rtlsdr00000001] Finished receiving samples
2020-06-03 20:01:45,938 - INFO - Program closed
```

Fig. 11. Content of the log file.



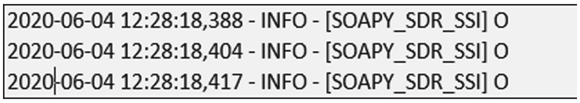
```
000000000010000000001100000000101000000001110000000100100000001011000000011010000000111100000100010000010011000001010100000101110000011001000001101100000111010000011111000010000100011000010010100001001110000101001000010101100001011010 0
0010111100001100010000110011000011010100001101110000111001000011101100001111010000111111000100010100010001110001001001000100101100010011010001001111000101001100010101010001010111000101100100010110110001011101000101111100011 00011
0010100011001110001101001000110101100011011010001101111000111001100011101010001110111000111100100011110110001111101000111111100100100110010010101001001011100100110110010011101001001111100101001010011100101010110010101101 00101011
1100101100110010110101001011011100101110110010111101001011111100110011010011001111001101010100110101110011011011001101110100110111110011100111010110011101101001110111100111101010011110111001111101100111111010011111111 01010101011
10101011011010101111101011010110111101011101110101111011010111111101101101110110111111011101111101111011111111110000000000000000000100000000011000000001010000000111000000100100000010110000001101000000111100000100010000010011000001
0101000001011100000110010000011011000001110100000111110000100001000110000100101000010011100001010010000101011000010110100001011110000110001000011001100001101010000110111000011100100001110110000111101000011111100010001010001000 11
1000100100100010010110001001101000100111100010100110001010101000101011100010110010001011011000101110100010111110001100011001010001100111000110100100011010110001101101000110111100011100110001110101000111011100011110010001111011 00
0111110100011111110010010011001001010100100101110010011011001001110100100111110010100111001010101100101011010010101111001011001100101101010010110111001011101100101111010010111111001100110100110011110011010101001101011100110
1101100110111010011011111001110011101011001110110100111011110011110101001111011100111110110011111101001111111101010101110101011011010101111101011010110111101011101110101111011010111111101101101110110111111011101111101111011111
11111000000000
```

Fig. 12. Matching binary sequence.

The open source program Universal Radio Hacker [9] was used to analyse and decode the signal of the captured IQ samples file. Figure 10 generated by the program shows the analogue representation of the OOK pulses of the transmission, representing the start of the binary sequence presented previously. The captured file was analysed and decoded and the decoded data from URH matched the binary sequence transmitted (Fig. 12). As a result of the forensic capture of the signal as shown in this scenario, information such as signal logging, GPS location, UTC time, etc. can be compiled to prove an event. Also, the signal can be reproduced with the original device in the future and compared with the forensic capture to prove that it is the same signal.

6 Conclusion

This paper reports the development of a forensic tool for acquiring radio spectrum in a forensically sound way with the necessary logging capabilities for examinations by investigators. Using the tool, two experiments were conducted involving the acquisition of radio signals. The main limitations of the acquisition are in the SDR receiver used, but there are also limitations in the software. During sampling at the highest sample rate supported by the HackRF (20 MHz) and using a complex 32-bit float stream format (CF32), an overflow was reported and logged to the log file (Fig. 13). The error did not occur when using the 8-bit integer complex sample (CS8) native format used by RTL-SDR and HackRF, but it would be a problem in high end SDR devices with higher bit resolution.



```
2020-06-04 12:28:18,388 - INFO - [SOAPY_SDR_SSI] O
2020-06-04 12:28:18,404 - INFO - [SOAPY_SDR_SSI] O
2020-06-04 12:28:18,417 - INFO - [SOAPY_SDR_SSI] O
```

Fig. 13. Log messages of stream status indicating overflow and lost samples.

The streaming implementation should be done using a low-level language rather than Python in order to increase the speed of the samples and avoid stream overflows. Currently, GPS logging consists of a file containing different information separated by colons. GPX is a standard format that can easily be processed and converted to other formats, so future implementations should record GPS data in this format. A visual display of the spectrum should be added to the tool in the future to allow live monitoring of radio activity for a selected band by investigators. While capturing samples, file compression techniques should also be examined to reduce file sizes.

References

1. Kamkar, S.: Drive It Like You Hacked It (DEF CON 23) (2015). https://samy.pl/defcon2015/. Accessed 10 June 2021
2. Keyless Entry System Attacks (2017). https://conference.hitb.org/hitbsecconf2017ams/sessions/chasing-cars-keyless-entry-system-attacks/. Accessed 10 June 2021
3. Ballantyne, S.N.T.: Wireless Communication Security: Software Defined Radio-Based Threat Assessment, MSC by Research Thesis, Coventry University (2016)
4. Achi, H., Hellany, A., Nagrial, M.: Digital forensics of wireless systems and devices technical and legal challenges. In: 6th International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET), pp. 43–46 (2009). https://doi.org/10.1109/HONET.2009.5423057
5. Turnbull, B., Osborne, G., Simon, M.: Legal and technical implications of collecting wireless data as an evidence source. In: Sorell, M. (ed.) e-Forensics 2009. LNICSSITE, vol. 8, pp. 36–41. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02312-5_5
6. Ngobeni, S., Venter, H., Burke, I.: A forensic readiness model for wireless networks. In: Chow, K.-P., Shenoi, S. (eds.) DigitalForensics 2010. IAICT, vol. 337, pp. 107–117. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15506-2_8

7. Kismet (2020). https://www.kismetwireless.net/. Accessed 10 June 2021
8. Csete, A.: Gqrx SDR – Open Source Software Defined Radio by Alexandru Csete OZ9AEC (2021). https://gqrx.dk. Accessed 10 June 2021
9. Universal Radio Hacker (2021). https://github.com/jopohl/urh. Accessed 10 June 2021
10. ACPO Good Practice Guide for Digital Evidence (2012). https://www.digital-detective.net/digital-forensics-documents/ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf. Accessed 10 June 2021
11. Kirk, P.L.: Crime investigation: physical evidence and the police laboratory. Science **118**(3061), 256–257 (1953). https://doi.org/10.1126/science.118.3061.256
12. Pollitt, M.: An ad hoc review of digital forensic models. In: Second International Workshop on Systematic Approaches to Digital Forensic Engineering, IEEE (2007). https://doi.org/10.1109/SADFE.2007.3
13. Wireless Telegraphy Act 2006. https://www.legislation.gov.uk/ukpga/2006/36/section/48. Accessed 10 June 2021
14. Communications Act (2003). https://www.legislation.gov.uk/ukpga/2003/21/contents. Accessed 10 June 2021
15. Office of Communications (OFCOM). Guidance on Receive-Only Radio Scanners. https://www.ofcom.org.uk/spectrum/interference-enforcement/radio-interception. Accessed 10 June 2021
16. Ofcom UK Frequency Allocation Table (UKFAT) (2017). https://www.ofcom.org.uk/__data/assets/pdf_file/0016/103309/uk-fat-2017.pdf. Accessed 10 June 2021
17. IR 2030 – UK Interface Requirements 2030 by Ofcom. https://www.ofcom.org.uk/__data/assets/pdf_file/0028/84970/ir-2030.pdf. Accessed 10 June 2021
18. Nooelec NESDR SMArt v4 & 3 Antennas (2021). https://www.nooelec.com/store/nesdr-smart.html. Accessed 10 June 2021
19. Nooelec - Extruded Aluminum Enclosure for HackRF One (2021). https://www.nooelec.com/store/sdr/sdr-addons/hackrf-enclosure.html. Accessed 10 June 2021
20. TCXO Nooelec for HackRF (2021). https://www.passion-radio.com/sdr-accessory/tiny-tcxo-719.html. Accessed 10 June 2021
21. Python Library, Pothosware/SoapySDR (2020). https://github.com/pothosware/SoapySDR. Accessed 10 June 2021
22. GNU Radio - The Free & Open Source Radio Ecosystem · GNU Radio (2021). https://www.gnuradio.org/. Accessed 10 June 2021
23. SDR# - airspy.com (2021). https://airspy.com/download/. Accessed 10 June 2021
24. De Bruijn Sequence (2021). https://en.wikipedia.org/wiki/De_Bruijn_sequence. Accessed 10 June 2021
25. Harding, C.: Exploitagency/Github-Rfpwnon (2016). https://github.com/exploitagency/github-rfpwnon. Accessed 10 June 2021
26. Bodor, D.: 0xDRRB/hackrf_ook (2019). https://github.com/0xDRRB/hackrf_ook. Accessed 10 June 2021