Software Defined Radio and its Applications

Julian Janisch
FH OÖ Campus Hagenberg, Mobile Computing
Softwarepark 106/14
4232 Hagenberg im Mühlkreis, Austria
s1710237010@students.fh–hagenberg.**at**

Alexander Kemptner
FH OÖ Campus Hagenberg, Mobile Computing
Softwarepark 106/9
4232 Hagenberg im Mühlkreis, Austria
s1710237015@students.fh–hagenberg.**at**

***Abstract*—In the last few years, software defined radios (SDRs) have captured a large part of the radio transmitter and receiver market. Increasing CPU power makes it possible to replace expensive analog hardware with digital signal processing.**
This paper is an introduction to SDR. To present its capabilities, we performed experiments with inexpensive, off-the-shelf hardware: receiving air traffic transmissions, weather satellite images and performing replay attacks.

***Index Terms*—SDR, antenna, signal processing, ADS-B, NOAA, replay attack**

I. Introduction

A. What is SDR?

A software defined radio (SDR) system is a radio transmission system that heavily relies on software instead of analog hardware [1].
In case of a transmitter, it takes data as an input (source), performs a sequence of operations via software on this data, converts it to an analog signal (with the help of an DAC) and transmits it wirelessly through its radio frequency (RF) frontend. The receiver performs the same tasks in reverse order (with an ADC), finally outputting the same data on its sink [2, 5pp].

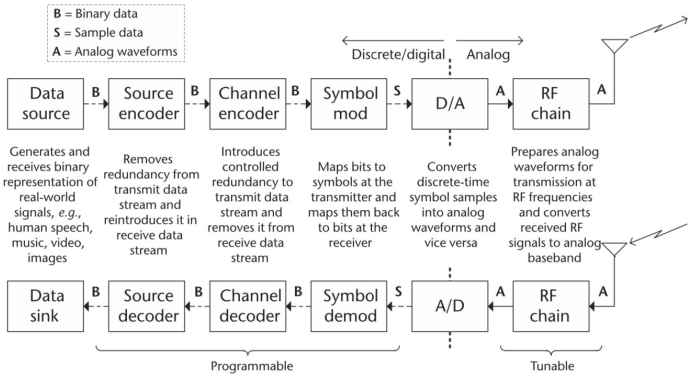


Fig. 1. The structure of an SDR. Most of the processing is done on the digital side, the D/A and A/D converters are at the very end/beginning of the chain.

In the diagram, the blocks marked as programmable can be implemented in software or programmable logic [2, 5p]. This makes it possible to build highly generic RF hardware that can be used for different tasks [2, 4p].
The digital processing needs to be done on an IC suited for the tasks. Commonly used parts include [1, 3p]:

- FPGAs and ASICs for sample rate conversions
- Digital signal processors (DSPs) for expensive computations in the demodulator
- Microcontroller units (MCUs) or CPUs for all other tasks

B. In this paper

In this paper, we will present applications using inexpensive SDR hardware.
First, we will show the hardware we used for our experiments. Afterwards, we will present software packages available to use with SDR receivers on the different platforms. Finally, we will discuss different applications using SDRs.

II. Hardware

A. SDR receivers/transmitters

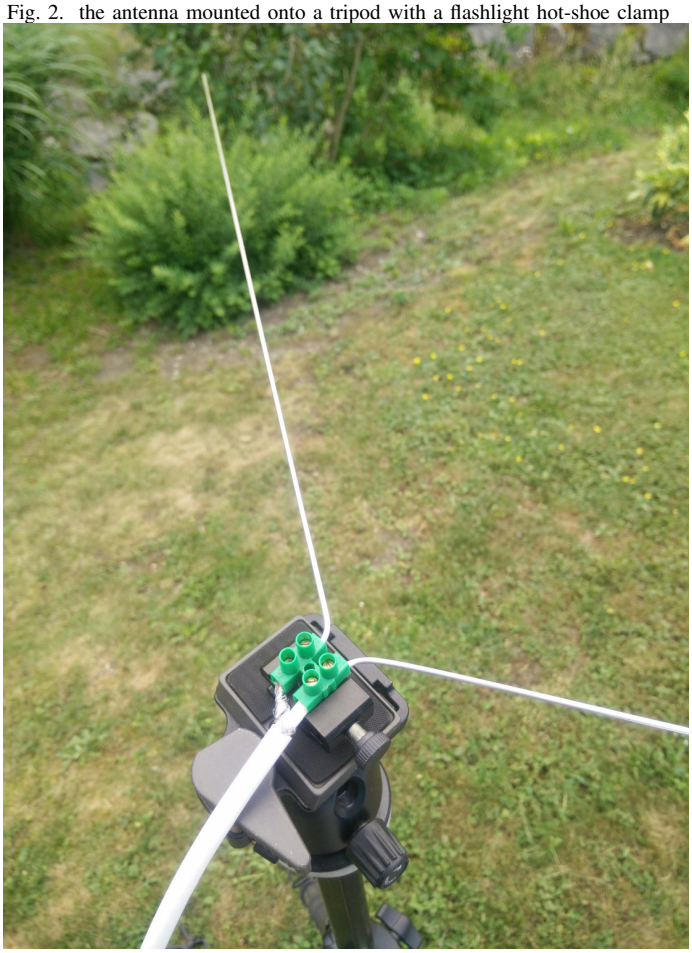
There is various hardware on the market for software defined radio receivers and transmitters (HackRF, SDRplay, RTL-SDR, Airspy, LimeSDR, FunCube, etc.). The software defined radios we used were:

- HackRF One - for data transmitting and receiving
- RTL-SDR - for data receiving
- SDRplay - for data receiving

1) HackRF One: Compared to the other Software Defined Radios of our list, the HackRF One with a price of around 300€ is the most expensive. But it has several advantages: It provides a very high operational bandwidth (from 30MHz to 6GHz) with an instantaneous bandwidth of 20MHz. (Instantaneos bandwidth = It can acquire 20MHz of RF spectrum around the chosen center frequency in real-time without a re-tuning of the oscillator) [3] The HackRf is not only able to receive radio signals, but can also to transmit them. Radio signals between 30MHz and 6GHz (e.g. Signals from remote controls of wireless sockets) can be recorded and afterwards sent again. This makes it possible to perform replay attacks (see section VI).

2) SDRplay: In order to receive radio signals we also ordered an SDRplay (Type RSP1). This receiver covers the RF spectrum from 1kHz to 2GHz. The SDRplay ships with a lot of software support (HDSDR, SDR Console, Cubic SDR and SDRuno), hence is very easy to setup. The instantaneous bandwidth is 10MHz and there are several preselection filters built in for better receiving-results. To provide more accuracy

Fig. 2. the antenna mounted onto a tripod with a flashlight hot-shoe clamp



the SDRplay uses a 14-Bit ADC (RTL-SDR uses only 8-Bit) [4].

3) RTL-SDR: This is the cheapest product on our list.(It did cost about 30€) The RTL-SDR is a USB-dongle which is capable of receiving radio signals between 500kHz and 1,75GHz. The usage of these dongles started years ago from mass produced DVB-T TV Tuners. Because of an easy accessible chipset (RTL2832U) those tuners could be converted into wideband software defined radios (trough custom software drivers). This discovery made the access to the radio spectrum much cheaper [5].

B. Antennas

1) ADS-B Antenna: For our experiments, we used a monopole antenna specifically designed for ADS-B on 1090 MHz. It was sourced on Amazon for € 10,56.

2) NOAA APT V-dipole: Some frequently used antenna designs for receiving NOAA satellites are the (cheaper) turnstile antenna and the (better suited) quadrifilar antenna [6, 45p], because the ideal polarisation is right-hand circular [6, 44p]. In our experiment, we used a different, easier design. Our antenna was a V-dipole with horizontal polarisation. This design achieves 20 dB of attenuation of the vertically polarised broadcasting stations located near the 137 MHz frequency of the POES satellites. It uses two 53.4 cm aluminium rod legs bent apart 120 degrees. They are fixed in a choc block terminal. On the other side, the shield of a coax cable is connected to one of the terminals, the inner conductor to the other [7]. We used standard TV coax cable, finished them with type F connectors and connected the antenna to our SDR receiver through an F-to-SMA connector. Our finished design is shown in figure 2.

III. Software

A. GNU Radio and the GNU Radio Companion

GNU Radio is a signal processing software and software development toolkit available for Linux, OS X and (with less support available) Windows [8] [9]. It is licenced under GPLv3 or later [8].

The desired signal processing steps are laid out as blocks that are connected to each other. End user applications are written using Python, while the performance-critical backend is written in C++.

The GNU radio companion is a GUI for GNU Radio which aids in building these blocks through drag-and-drop, comparable to Simulink [8].

B. GQRX

GQRX is a graphical frontend for GNU Radio offering an FFT plot and waterfall graph, hardware abstractions for various SDR devices to set frequency and other settings and multiple demodulators.

The received data can be digitally filtered and recorded to a file [11].

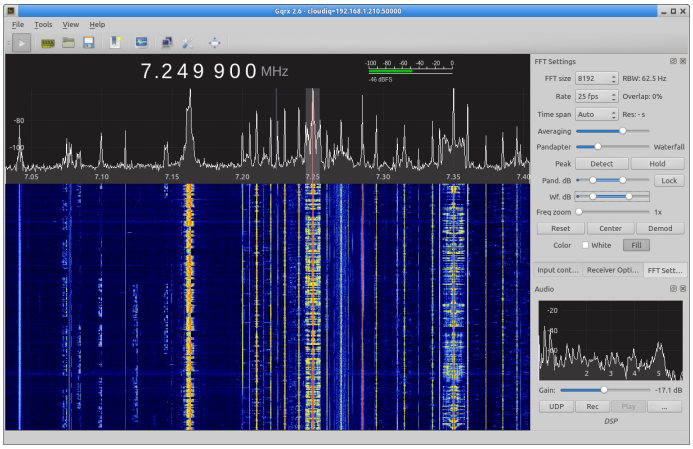


Fig. 4. GQRX main window [11]

C. SDR#

SDR# (SDR sharp) is developed at Airspy for their Airspy SDRs, but can also be used with other SDRs like the RTL-SDR.

The main software package is available for Windows only, while the ”Spy Server”, a program that allows streaming of

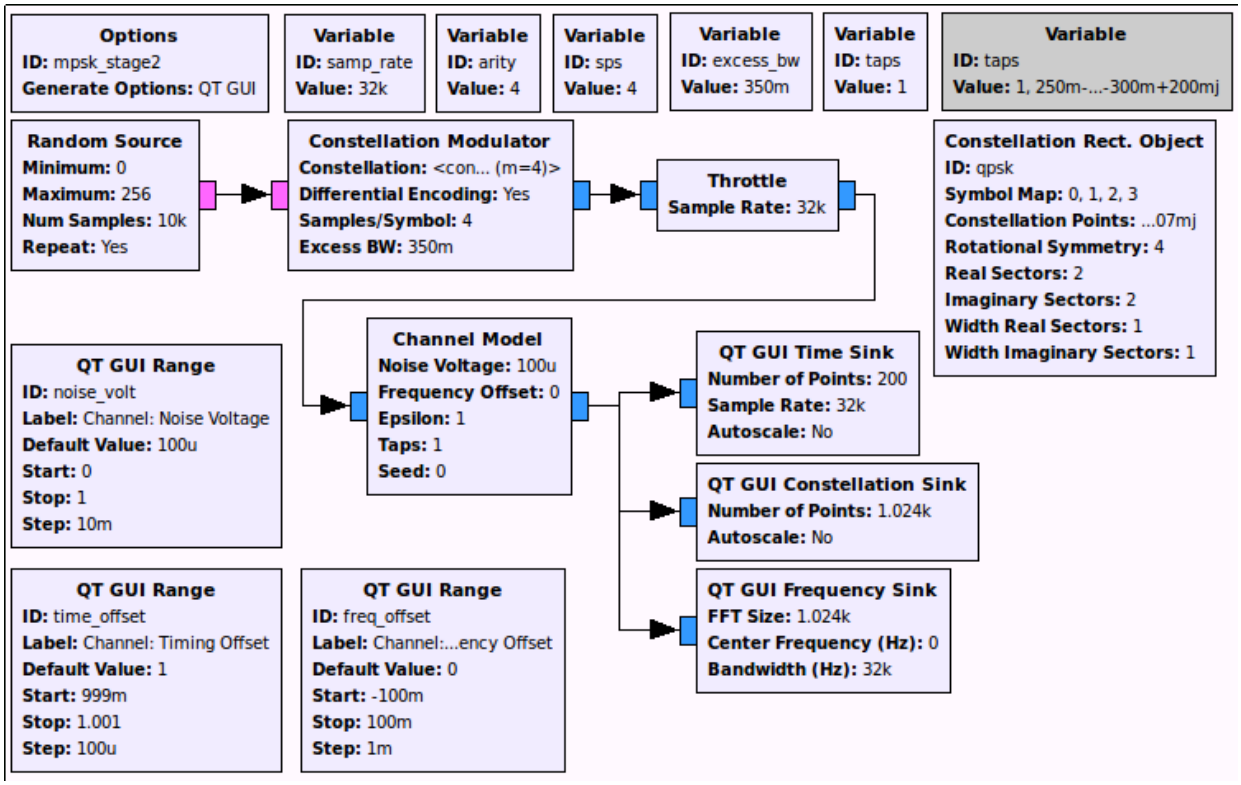


Fig. 3. An example flowgraph in GNU Radio Companion [10]

the signal to clients, is available for Linux on x64/x86 and ARM (Raspberry Pi, Orange Pi).
The functionality can be expanded with third-party plug-ins [12].

IV. ADS-B

Automatic Dependent Surveillance Broadcast is a system in air traffic control for transmitting flight data (location, ...) from an aircraft to air traffic control stations.
It should replace radar as the primary tracking mechanism because of its higher precision needed in modern, densely populated air space [13].
In this experiment, we want to capture and visualise the ADS-B messages sent by airplanes in our area.
Two standards with different frequencies are in use: Universal Access Transceiver (978 MHz) and 1090 MHz extended squitter (1090ES). The latter is more common in commercial aircraft, as it can be used with hardware already fitted for a previous communication standard (mode S) [13].
We used an HackRF SDR receiver for our experiment. Data was captured using the HackRF command line tool hackrf_transfer . The full command used was

```
hackrf_transfer -r - -f 1090000000 -s 2000000 -p 0 -a 0 -l 40 -g 62
```

1) -r: receive data
2) -: output to stdout
3) -f: frequency in Hz
4) -s: sample rate in samples per second
5) -p: antenna port power (disable)
6) -a: receiver and transmitter RF amplifier (disable)
7) -l: receiver end low noise amplifier gain in dB
8) -g: receiver end baseband amplifier gain in dB

The output of hackrf_transfer was then piped into SoX, an audio processing application [14]. This was necessary because the hackrf uses signed bytes, while the next application expects unsigned bytes. The full command was [15]:

```
sox --rate 2000000 --channels 1 --type sb - --type ub -
```

1) –rate: the sample rate
2) –channels: number of channels
3) –type: type of input (signed byte)
4) -: read from stdin
5) –type: type of output (unsigned byte)
6) -: write to stdout

SoX is licenced under the GPL and is available on sourceforge.net.
To decode and visualize the received data, we used dump1090. It can output the data on the console and on a website running on localhost. It is available on GitHub under a BSD licence [16]. The command used was:

```
dump1090 --ifile - --net --interactive
```

1) –ifile -: input file, read from stdin
2) –net: enable networking
3) –interactive: show interactive data on console

Fig. 6. The APT protocol [6]

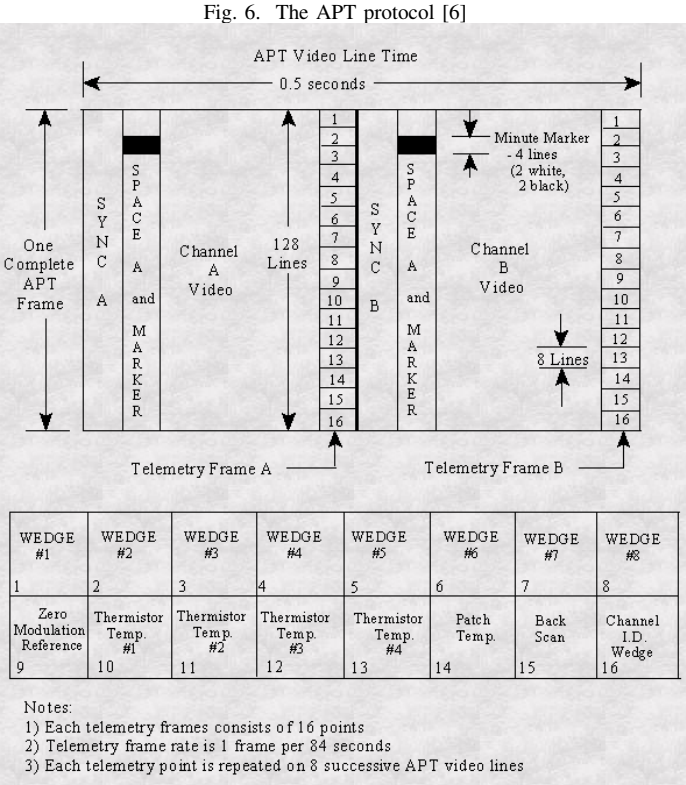


Fig. 5. Output of dump1090 on 23rd June, 20:13 in Hagenberg/Mühlkreis

dump1090 --interactive

Hex	Mode	Sqwk	Flight	Alt	Spd	Hdg	Lat	Long	RSSI	Msgs	Ti
0083C3	S	3373	CAI625	33975	450	282			-26.9	119	10
49D336	S			11100					-38.1	3	28
452075	S			20850					-35.7	3	22
4BA90C	S	7164		37000	477	113			-31.5	106	10
4BAA59	S			32975					-35.2	3	24
505CE6	S	0454	ENT73WZ	36000					-27.6	161	10
896196	S								-33.7	6	39
06A125	S	3417	QTR9KP	38350	458	115	48.819	16.904	-25.7	274	4
44038B	S			33000					-30.6	7	49
4CA56A	S	1000	RYR2VD	37000	427	108	49.084	15.725	-24.5	915	9
02A194	S	7107	TAR643	16050	387	208	47.749	15.957	-26.9	358	2
502CE7	S	5411	DLH5CJ	34000	472	276	47.957	15.936	-26.4	618	4
47878E	S	4561	NAX38FY	36000	440	358	46.435	15.970	-28.8	246	3
49D152	S	1000	OKPMP	24000	233	023	47.295	14.345	-28.3	237	11
3C0A5B	S			38000					-32.5	7	10
896457	S	4610	UAE79Y	39000	477	110	48.962	16.339	-26.4	394	10
440828	S	1000	AUA257	20250	387	315	47.533	14.726	-24.5	250	9
3C4B4B	S	6420		29725	412	100			-31.4	33	13
39CEAB	S	1000	TVF44JA	35550	446	095	47.068	15.131	-28.3	478	0
471F67	S	1335	WZZ8GL	37050	446	235	47.652	14.057	-28.1	378	10
4CA840	S	6041	ERN274	39000					-30.2	702	1
4B1A29	S	1000	EZS52XB	25650	417	076	47.565	15.020	-18.6	764	8
70605C	S	3420	KAC102	35000	471	118	48.393	15.680	-26.3	1093	0
3C6659	S	1000	DLH9V	20100	453	281	48.216	15.073	-26.1	884	2
42453A	S	0471	SVR5002	36000	438	061	48.049	16.169	-30.2	382	0
42438B	S	0273	PBD858	35000	442	047	47.196	17.699	-32.8	232	23
AE04F2	S	7654		26950	271	119			-29.2	55	9
4BAACC	S	3205	THY7ME	34000	407	298	46.869	16.033	-28.3	548	2
3C0A4C	S	2021	SXD13H	38000	426	358	46.646	15.958	-26.8	277	9
3C48D0	S	1000	EWG3G	20425	412	206	47.592	15.750	-27.0	930	0
73806E	S	6015	ELY351	29525	472	293	47.420	14.725	-22.9	995	8
4242F4	S	3247	AHY007	40000	469	287	48.201	15.431	-19.3	1899	0
300738	S	1000	DLH3RP	33850	471	279	47.893	14.847	-26.7	1400	0
4BA9AB	S	4474	THY39LT	35000	443	133	47.611	16.376	-32.3	866	2
3C66B0	S	3263	DLH1301	36025	461	286	48.218	14.506	-19.2	1966	0
44009D	S	4142	AUA924	12700	320	082			-26.4	828	0
4BB853	S	3427	PGT93L	36975	443	125	48.082	14.828	-18.8	2220	0
44B8E9	S	6162	FYG58W	40000	436	291	49.614	14.838	-29.2	305	10
4BAAD5	S	7652	THY1LR	39000	440	133	48.260	15.444	-27.1	1567	0
4CA242	S	5157	RYR9PM	38000	464	337	49.145	15.660	-26.5	807	9
342108	S	1455		38050					-30.4	24	12
503DC2	S	6430	SXS147	37000	445	134	48.594	14.813	-14.5	1413	7
4CA6D3	S	5155	RYR773	36000	444	304	48.934	15.034	-29.4	142	9
44006C	S	1000	AUA374	23200	414	120	48.658	15.077	-23.8	1212	0
49D093	S	7653	EWG62E	22075	390	094	48.553	14.777	-24.7	1372	2

Alternatively, the flightaware fork of dump1090 has built-in support for RTL-SDR dongles [17]. It can be started with

dump1090 $--interactive$

Both sources, RTL-SDR and HackRF produce an output similar to the one shown in figure 5. In interactive mode, one row corresponds to one aircraft. Rows are updated continuously and deleted if no messages from this aircraft were received in the last 60 seconds. Decoded fields are, among others, flight number, squawk code, altitude, coordinates, speed, heading, number of messages received and signal strength [16]. One can see that just after the one-minute-mark, some aircraft are approaching 2,000 messages.

V. NOAA APT

The *National Oceanic and Atmospheric Administration* is an organisation of the US government producing, among other things, weather forecasts [18]. For this purpose, a suborganisation called the *Satellite and Information Service* currently operates 18 satellites [19]. Among them, there are 5 geostationary GOES satellites and four NOAA sun-synchronous polar-orbiting satellites (POES) [20]. We will focus on the latter in this application example.

Automatic Picture Transmission is an analog system for transmitting images from satellites to ground stations. In case of (POES) NOAA satellites, two images of different wavelengths are transmitted simultaneously.

The system uses frequency modulation with a carrier frequency of around 137 MHz [21]. The downsampled 8-bit signal from the sensor is modulated onto a 2.4 kHz band via amplitude modulation. This is in turn frequency modulated onto the carrier frequency of the satellite [6].

As shown in figure 6, data is transmitted line-by-line, one line every 0.5 seconds, each 2800px long (including metadata) [22, 2p]. Lines start with a synchronisation block (832 Hz for IR, 1040 Hz for visible to distinguish the two) for the decoding software, followed by minute and space markers (the sensor briefly scans empty space turned away from earth, which shows up as pure black or white, and inserts an opposingly colored bar every minute). This is followed by the image data. At the end of each frame is the telemetry data, containing calibration data [6, 41pp].

In this experiment, we attempt to receive these APT images with the mentioned inexpensive, partially self-built equipment. We performed two experiments, receiving two NOAA POES satellites, NOAA 15 and NOAA 19, at different locations.

Information about upcoming passes was sourced from heavens-above.com. The first pass was NOAA 19 on 21th June 2019, 16:24 local time. The antenna was set up in a valley, and the maximum elevation of the satellite was 39 degrees, which led to a loss of signal at the beginning and end of transmission. The second attempt was on an observation deck on a mountain at 750m above sea level. This reception shows more information at the extremes of the image, also due to the satellite passing directly overhead (88 degrees [23]).

On the hardware side, we used the home-built V-dipole

antenna mentioned in II-B2, the SDR receiver used was an SDRplay (RSP1A), described in II-A2.
For receiving the signal, we used GQRX (see III-B), tuned to the respective satellite frequency. Modulation was set for frequency modulation (FM). We used varying levels of gain, settling at 33.0 dB IFGR and 4.0 dB RFGR (IF = intermediate frequency/digital side, RF = radio frequency/analog side [24], GR = gain reduction from maximum [25]). Audio gain was set to -15.0 dB. The immediate bandwidth was set to approximately 60 kHz. A screenshot can be seen in figure 7. The signal was recorded to a .wav-file, as shown by the pressed button in the lower right corner of figure 7.

Afterwards, the image was decoded using the *noaa-apt image decoder*, which is available for Windows, Linux and MacOS. A popular alternative and precedessor, *WXtoIMG*, is no longer actively maintained and is only available from unofficial websites [26].
As mentioned, the second recording was able to capture more information in upper/lower part of the image due to the higher vantage point and more favourable altitude of the pass. The results can be compared in figure 8 and 9. The right part of figure 8 (near-visible light, as the image was flipped 180 degrees due to the passing direction of the satellite) shows the Black Sea and the Crimean Peninsula in the south, and the Baltic Sea in the north. Both can be distinguished from land by their distinct darker shade of gray [6, 42p]. Figure 9 shows Great Britain, Denmark and the Netherlands, France and Sweden, Norway, Finland and Estonia. Italy and parts of Libya and Tunisia can also be seen. The position of the station is nearly exactly in the horizontal center, according to the pass data [23], just below a cloud formation as seen on-site.
The left parts show the infrared spectrum of the same area [6].

VI. Replay Attack

To perform a replay attack, some radio signals are recorded at first (e.g.: unlocking a car via remote key) and transmitted at some later point in time. If the transmitted radio signal to get some desired behaviour (e.g. switching on a remote socket) never changes, an attacker could record it with a software defined radio and transmit it afterwards at any time he/she wants to get that behaviour again. Because of this, newer cars use authentication protocols with changing codes for unlocking [27].
To perform this attack, whe had to know the approximate frequency range of the radio signal and to use hardware that is capable of sending and receiving radio signals within that range (e.g. HackRF One).

A. Example remote socket

To try out the capabilities of the HackRF, we did our own replay attack on the remote control of a wireless switch unit. Most of these radio signal devices have their sending frequency written on them. This was also the case with our remote control and switch unit. On the back it said 433,92MHz (see Fig. 10).

Fig. 10. Remote Control and switch unit frequency.

1) Inspecting the signal: Based on the given frequency, we set up our RTL-SDR with the gqrx-software to inspect the sent data. When pressing the on-/off-button of the remote control we could clearly see the changes in the graphic. (see Fig. 11)

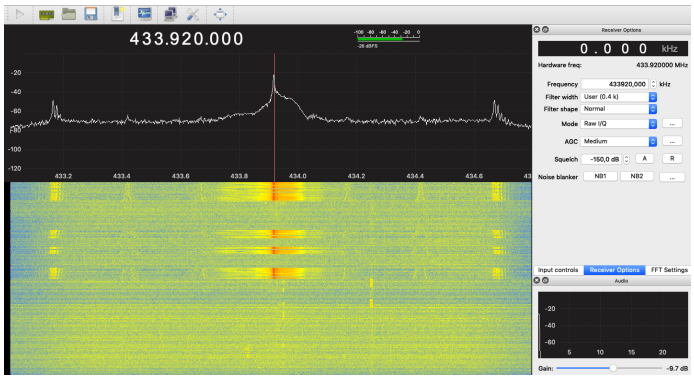


Fig. 11. Received remote control signals using gqrx.

The signal seems to have its peak at exactly 433,92MHz. There are also some side lobes noticeable (at 433,19MHz and 434,65MHz).

2) Recording the data: Knowing the center frequency, we tried to record the data with our HackRF One. Therefore we installed the HackRF software [28] using homebrew.

```
brew install hackrf
```

To record and transmit the data with the HackRF we used the *hackrf_transfer* function. Our command was the following:

Fig. 7. Screenshot of GQRX while receiving NOAA 15 on 21th June, 19:16

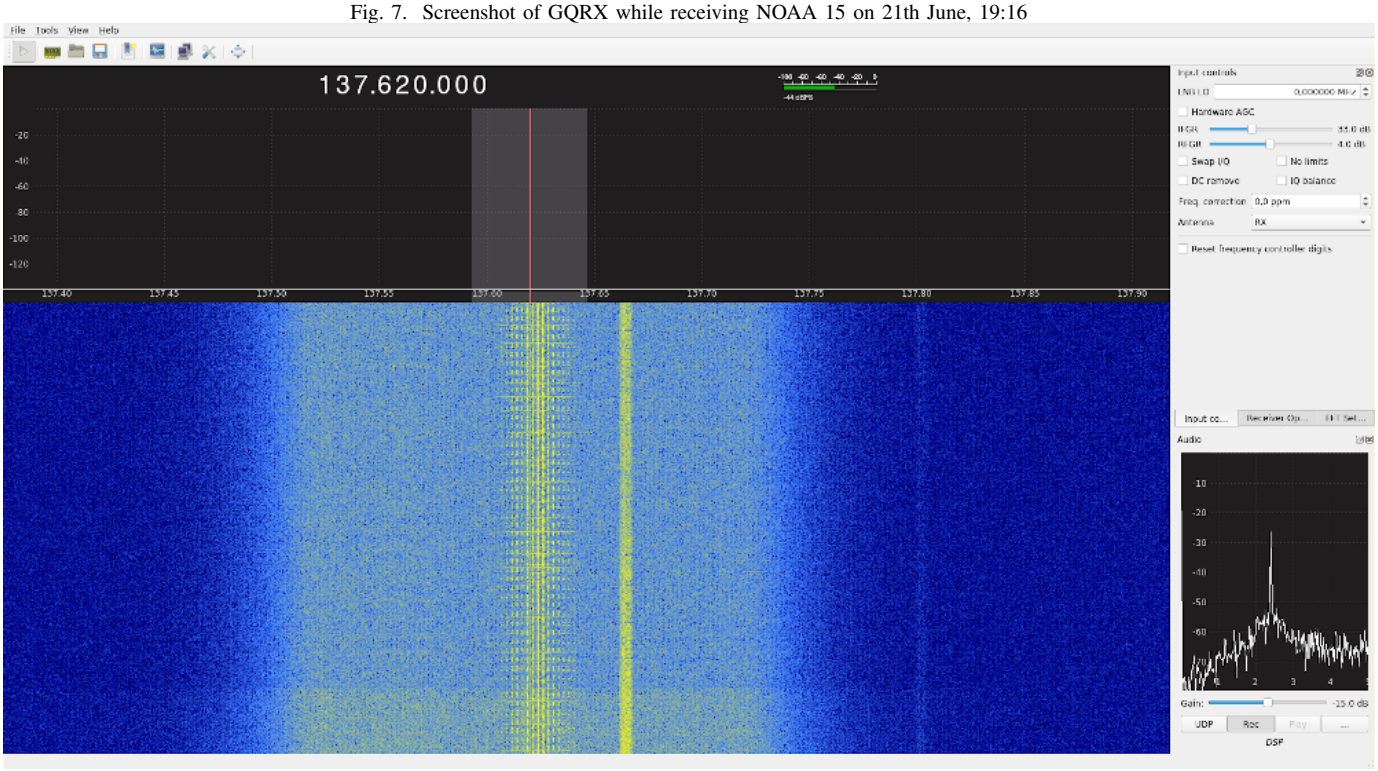


Fig. 8. NOAA 19 on 21th June, 16:24

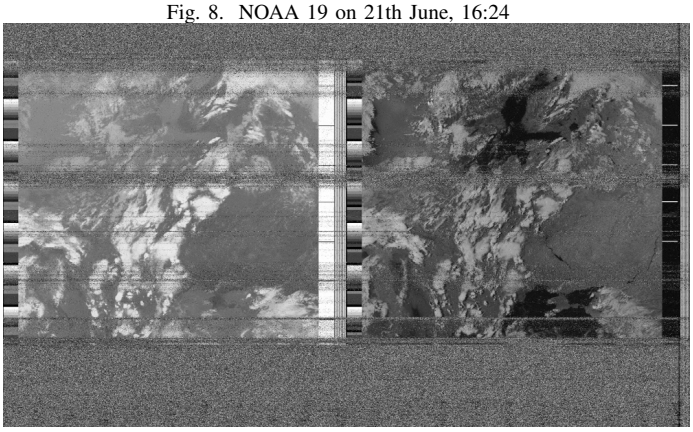
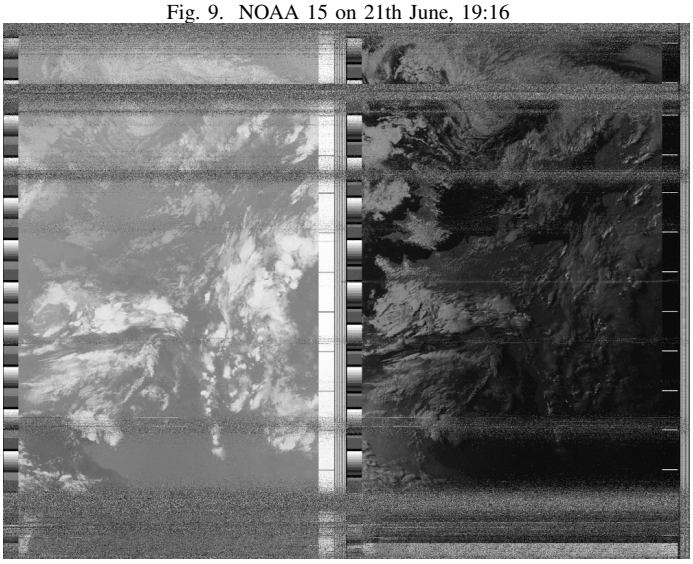


Fig. 9. NOAA 15 on 21th June, 19:16



```
hackrf_transfer -r output/received.raw -f 433920000
    -l 20 -g 20 -b 2000000
```

1) -r: receive
2) -f: frequency (in Hz)
3) -l: gain (IF) in dB
4) -g: gain (baseband) in dB
5) -b: baseband filter bandwidth in Hz

3) Transmitting the data: To transmit the edited file we again used the *hackrf_transfer* function, but now in the transmitting mode. Our transmitting command was the following:

```
hackrf_transfer -t output/edited.raw -f 433920000 -x
    40 -b 2000000
```

1) -t: transmit
2) -f: frequency (in Hz)
3) -x: TX gain in dB
4) -b: baseband filter bandwidth in Hz

Simultaneously we watched the sent data with our RTL-SDR. Through inspecting we detected that our chosen gain was too low to be detectable from the switch unit (see Fig. 12), so we

put it up to 40dB (see *-x* in command above). That gave the signal enough power to be detected by the switch unit and to recreate the recorded behaviour. Although the replayed signal looks different than the recorded one in qgrx (see Fig. 13), it still works. So our replay attack was already successful.

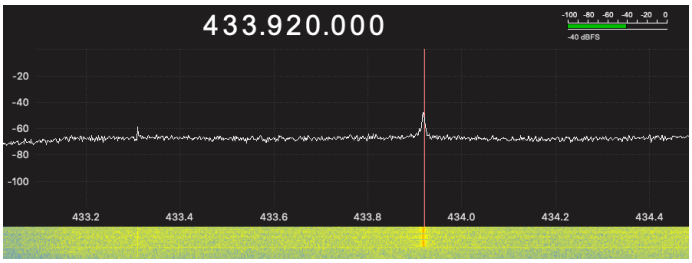


Fig. 12. The transmitted HackRF signal captured with a RTL-SDR using the gqrx software and 10dB TX gain.

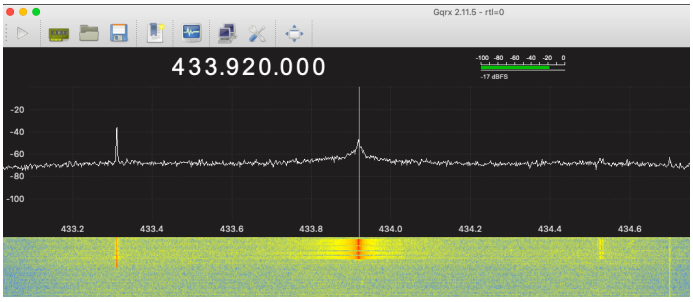


Fig. 13. The transmitted HackRF signal captured with a RTL-SDR using the gqrx software and 40dB TX gain.

4) Cutting in Audacity: The recorded data was very big (ca. 25MB per second of recording time), so we analysed the recorded files with the audio-editing software audacity. We imported the files into the software through *file - import - rawdata* and set the import settings to the following:

1) Encoding: Unsigned 16-Bit PCM
2) Byte order: Big Endian
3) Channels: 1 Channel (Mono)
4) Offset: 0 Bytes
5) Amount: 100
6) Sample rate: 10MHz (default of hackrf_transfer)

After doing that we could detect, that when switching on or off there is the same signal sent periodically (see Fig. 14 and 15).

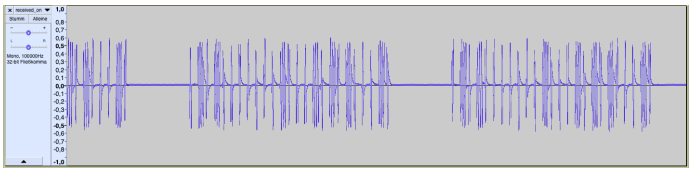


Fig. 14. The recorded data to switch the wireless switch unit "On", analyzed in audacity with two and a half occurrences of the periodic signal.

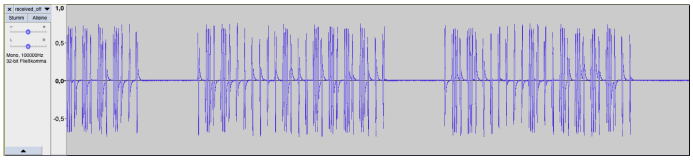


Fig. 15. The recorded data to switch the wireless switch unit "Off", analyzed in audacity with two and a half occurrences of the periodic signal.

So we could cut away most of these repetitive signals. In our case we needed at least three repetitions of the signal to switch the unit. After the cutting we exported the data again as raw file (*File - export - audio export - raw (header-less) with the unsigned 16-Bit PCM encoding*) and sent it with the HackRF. In doing that we could reduce the filesize from 70MB to 2MB.

VII. Conclusion and Further Work

There are a lot of applications for software defined radio. The experiments we did were just some of the plenty things that are possible with inexpensive receivers and transmitters. The community around SDR is very big, hence there are a lot of projects on different use cases to find on the internet [5]. Some of them are:

- Tracking boat positions with AIS decoding [29]
- Listening to the International Space Station (ISS) [30]
- Watching analog broadcast TV [31]
- Sniffing GSM signals [32]
- Monitor meteor burst communications [33]

References

[1] A. Heuberger and E. Gamm, *Drahtlose Netzwerke*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017, pp. 135–141. [Online]. Available: https://doi.org/10.1007/978-3-662-53234-8_4

[2] A. Wyglinski, R. Getz, T. Collins, and D. Pu, *Software-Defined Radio for Engineers*, ser. Artech House mobile communications series. Artech House, 2018. [Online]. Available: https://books.google.at/books?id=cKR5DwAAQBAJ

[3] A. Mishra and D. Johnson, *White Space Communication: Advances, Developments and Engineering Challenges*, ser. Signals and Communication Technology. Springer International Publishing, 2014. [Online]. Available: https://books.google.at/books?id=G0vPBAAAQBAJ

[4] SDRplay, "SDRplay RSP1A Datasheet," Jun. 2019, [accessed 18. Jun. 2019]. [Online]. Available: https://www.sdrplay.com/docs/RSP1Adatasheetv1.9.pdf

[5] RTL-SDR, "About rtl-sdr," Jun. 2019, [accessed 18. Jun. 2019]. [Online]. Available: https://www.rtl-sdr.com/about-rtl-sdr/

[6] National Oceanic and Atmospheric Administration, *User's Guide for Building and Operating Environmental Satellite Receiving Stations*, Feb. 2009. [Online]. Available: https://noaasis.noaa.gov/NOAASIS/pubs/Users_Guide-Building_Receive_Stations_March_2009.pdf

[7] Adam, "DIY 137 MHz APT Weather satellite antenna – Or, do we need a circular polarization?" Dec. 2015. [Online]. Available: https://lna4all.blogspot.com/2017/02/diy-137-mhz-wx-sat-v-dipole-antenna.html

[8] GNU, "FAQ - GNU Radio," Mar. 2019, [accessed 17. Jun. 2019]. [Online]. Available: https://wiki.gnuradio.org/index.php/FAQ

[9] "What is GNU Radio? - GNU Radio," Mar 2019, [accessed 17. Jun. 2019]. [Online]. Available: https://wiki.gnuradio.org/index.php/What_is_GNU_Radio%3F

[10] "File:Stage2 grc.png - GNU Radio," Mar 2019, [accessed 17. Jun. 2019]. [Online]. Available: https://wiki.gnuradio.org/index.php?title=File:Stage2_grc.png&oldid=267

[11] GQRX, "Welcome to GQRX," Jun. 2019, [accessed 17. Jun. 2019]. [Online]. Available: http://gqrx.dk/

[12] Airspy, "Sdr software download," Jun. 2019, [accessed 17. Jun. 2019]. [Online]. Available: https://airspy.com/download

[13] M. Strohmeier, M. Schafer, V. Lenders, and I. Martinovic, "Realities and challenges of nextgen air traffic management: the case of ADS-B," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 111–118, 2014.

[14] S. contributors, "SoX - Sound eXchange — HomePage," Feb. 2015, [accessed 17. Jun. 2019]. [Online]. Available: http://sox.sourceforge.net/

[15] R. X. Seger, "Flight tracking with ADS-B using dump1090-mutability and HackRF One on OS X," Jun. 2016, [accessed 17. Jun. 2019]. [Online]. Available: https://link.medium.com/wsTz35CQOX

[16] S. Sanfilippo, "Dump1090 is a simple Mode S decoder for RTLSDR devices," Jan. 2017, [accessed 18. Jun. 2019]. [Online]. Available: https://github.com/antirez/dump1090

[17] FlightAware, "Dump1090 is a simple Mode S decoder for RTLSDR devices ," May 2019, [accessed 23. Jun. 2019]. [Online]. Available: https://github.com/flightaware/dump1090

[18] National Oceanic and Atmospheric Administration, "About our agency," Jun. 2019, [accessed 22. Jun. 2019]. [Online]. Available: https://www.noaa.gov/our-mission-and-vision

[19] Satellite and I. Service, "Currently Flying," Jun. 2019, [accessed 22. Jun. 2019]. [Online]. Available: https://www.nesdis.noaa.gov/content/currently-flying

[20] National Oceanic and Atmospheric Administration, "NOAA weather satellites," Dec. 2017, [accessed 18. Jun. 2019]. [Online]. Available: https://www.nws.noaa.gov/om/marine/wxsat.htm

[21] ——, "POES Operational Status," Mar. 2019, [accessed 22. Jun. 2019]. [Online]. Available: https://www.ospo.noaa.gov/Operations/POES/status.html

[22] V. Dascal, P. Dolea, O. Cristea, and T. Palade, "Advanced VHF ground station for NOAA weather satellite APT image reception," *Acta Technica Napocensis*, vol. 53, no. 3, p. 1, 2012. [Online]. Available: http://users.utcluj.ro/~ATN/papers/ATN_3_2012_1.pdf

[23] C. Peat, "NOAA 15 - Pass Details," Jun. 2019, [accessed 22. Jun. 2019]. [Online]. Available: https://heavens-above.com/passdetails.aspx?lat=47.9518&lng=14.4547&loc=Laussa&alt=0&tz=CET&satid=25338&mjd=58655.7238210938&type=A

[24] Analog Devices, *RF/IF CIRCUITS*, Analog Devices, Jan. 2007. [Online]. Available: https://www.analog.com/media/en/training-seminars/design-handbooks/Basic-Linear-Design/Chapter4.pdf

[25] T. Lemiech, "Channel frequencies," Jan. 2018, [accessed 22. Jun. 2019]. [Online]. Available: https://github.com/szpajder/RTLSDR-Airband/issues/75#issuecomment-360882803

[26] M. Bernardi, "noaa-apt 1.1.0 Image decoder for APT signals from NOAA satellites," Jun. 2019, [accessed 22. Jun. 2019]. [Online]. Available: https://noaa-apt.mbernardi.com.ar/index.html

[27] R. Verdult and F. D. Garcia, "Cryptanalysis of the megamos crypto automotive immobilizer," *USENIX; login*, vol. 40, no. 6, pp. 17–22, 2015.

[28] M. Ossmann, "Software for HackRF, a low cost, open source Software Defined Radio platform." Jun. 2019, [accessed 19. Jun. 2019]. [Online]. Available: https://github.com/mossmann/hackrf

[29] RTL-SDR, "Cheap ais ship tracking," Jun. 2019, [accessed 24. Jun. 2019]. [Online]. Available: https://www.rtl-sdr.com/rtl-sdr-tutorial-cheap-ais-ship-tracking/

[30] ——, "Listening to an astronaut transmitting from the international space station," Jun. 2019, [accessed 23. Jun. 2019]. [Online]. Available: https://www.rtl-sdr.com/listening-to-an-astronaut-transmitting-from-the-international-space-station/

[31] ——, "Analogue tv with rtl-sdr," Jun. 2019, [accessed 24. Jun. 2019]. [Online]. Available: https://www.rtl-sdr.com/analogue-tv-with-rtl-sdr/

[32] ——, "Analyzing gsm with airprobe/gr-gsm and wireshark," Jun. 2019, [accessed 23. Jun. 2019]. [Online]. Available: https://www.rtl-sdr.com/rtl-sdr-tutorial-analyzing-gsm-with-airprobe-and-wireshark/

[33] ——, "Meteor reflection observations with rtl-sdr," Jun. 2019, [accessed 23. Jun. 2019]. [Online]. Available: https://www.rtl-sdr.com/meteor-reflection-observations-with-rtl-sdr/