Replay Attack On A Remote Controlled Relay

Max-Felix Müller

2020
September

Contents

List of Figures

1 Motivation

It is called a replay attack when the attacker captures the signal of a device and stores it. At a later point in time the attacker transmitts the same signal again. The target (receiver) can not distinguish between the signal coming from the original transmitter and the signal the attacker is sending, thus it is handeld as a "real" signal and the target responds normaly.

In this case I want to learn about SDR and handling receiving and sending data in a very basic way. I found a remote controlled relay working at 433,92MHz in the ISM band at a local hardware store. By capturing the signal from the remote and then transmitting it, I should be able to control the relay from my PC and not using the included remote control.

I plan on using the Pluto SDR for this project. The main reason being that it is able to transmit and receive at the same time. This is also known as Full Duplex. That is useful for this application since I want to check on what I am sending to know how bad I am distrubing other applications in the RF spectrum. Also I have been using the cheap RTL SDR for quite some time now and the HackRF is used in the tutorial so that is not really a challenge. Another point against both of those is that I have only one USB port at the laptop I'm using and I would need both to simultaneously transmit and see what I am transmitting.

2 The Target For The Attack

The target is a cheap remote controlled relay from a local hardware store. It uses the ISM band on 433,92MHz (which is printed on it's back). If the frequency of operation were unknown, the possiblities would range from scanning the whole spectrum to searching for the FCC-ID.

The FCC-ID is a unique number the FCC gives to a device that it has approved for operation. Searching for this number on their website will result in datasheets, test results and sometimes even pictures from the inside of the device.

In this case though the frequency is just printed on the relay. Using dip switches the device can be set to one of many possible channels. On each channel there is space for up to four relays. The remote control has the same dip switches and eight buttons, two for each relay, one to turn the relay on and one to turn it off.

Figure 1: The target device for the replay attack. It is a simple remote controlled relay I found at a local hardware store.

3 Capturing The Signal

I want to capture the signal for the relay turning on and about a second later turning off again.

3.1 Getting A First Impression

To get a first impression of the signal I created a very basic flowgraph with the pluto source and a fft sink. That way I can look at the spectrum in near real time.

In the flowgraph you can also see the file sink. To capture the signal the file sink is used, either in parallel to the fft sink, or alone. Using both sinks in parallel will allow to view the signal while it is being captured. However on older hardware it is better running only the file sink, since using too much processing time could result in sampled being dropped and as a consequence the captured signal being not as clean as it could be. When using the Pluto SDR source and sink blocks it is important to remember that both the sampling rate as well as

the frequency have to be set. On other blocks especially the sampling rate is often set by default.

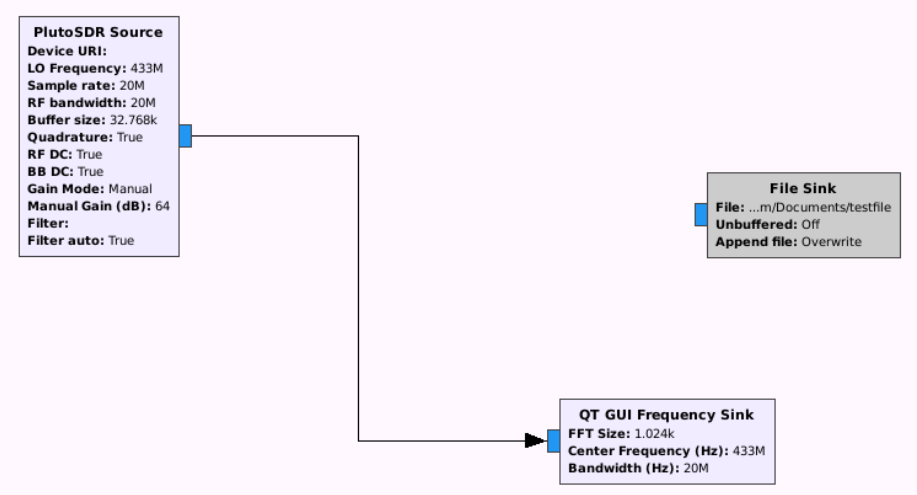

Figure 2: A simple flowgraph to look at the spectrum

To be honest, I already had a look at it before so I was horrified when I saw the signal the pluto SDR showed me. Either all signals I have seen so far were wrong, or the pluto captures the signal of the remote in a very weird way.

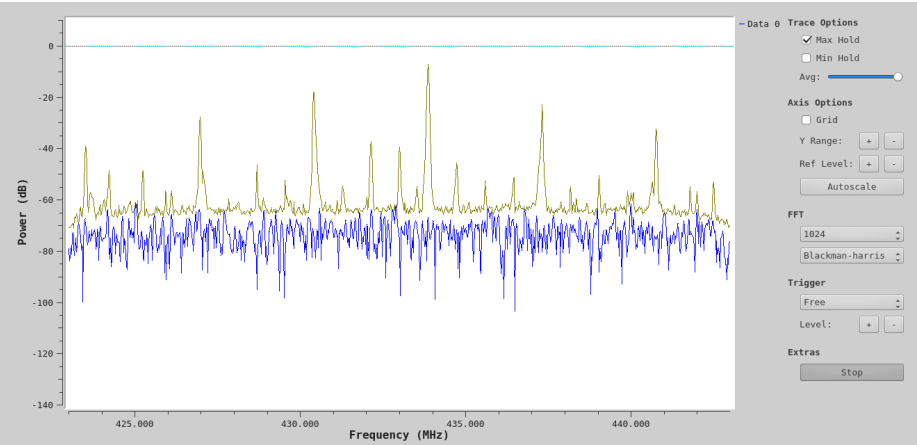

Figure 3: Looking at the spectrum of the remote control with the Pluto SDR. There should only be one peak visible at the working frequency of the remote. I know that, because I already had a look at it with two other SDR receivers.

My first guess was the sample rate being too high for the Pluto SDR but after a quick google search it turned out I was within specifications. When I was about to change the SDR I noticed a loose connection to the antenna which might result in reflections and thus a wrong signal, but that also turned out to be not the reason for the received signal.

So instead of fixing it I decided to change SDR to the HackRF One for the sake of capturing a clean signal. At least for now.

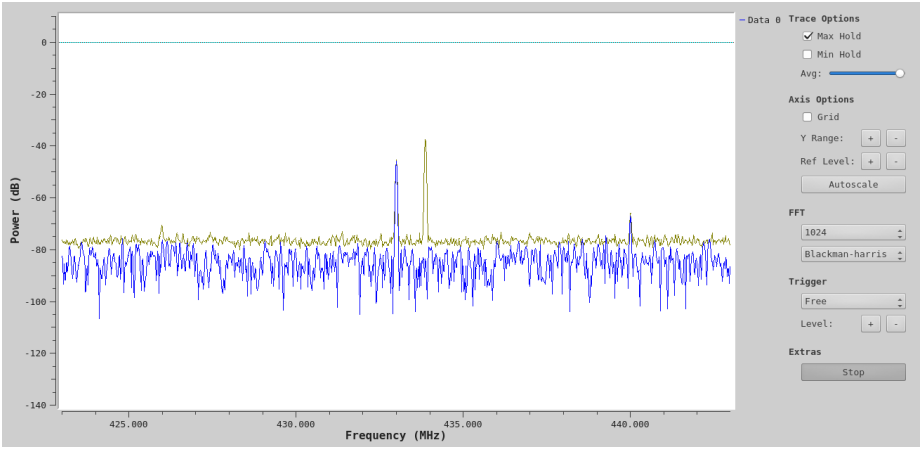

Figure 4: Looking at the specturm of the remote control with the HackRF One

I limited the sample rate to 10Msps and (so 10MHz bandwidth) and captured the on and off signal for relay A with all the dip switches being in the off position. The signal I captured was "on - on - off - off". I also captured just the on and just the off signal so I can have a closer look.

3.2 A Closer Look At The Pluto SDR

I had the urge to skip this, but that won't help me in the future. I have to figure out what it is that the Pluto SDR is doing that I get such a signal from it.

I started by bringing the Pluto SDR back to its default state. In this case that means: Standard antenna for receiving, standard antenna for transmitting (which I don't do here so it should change nothing), all values in Gnuradio back to default (which they should have been).

To my surprise, this already fixed the problem. That concludes that either the antenna for the receiver is bad or that the transmitter antenna influences the receiver. I say that because I used the same antenna on the HackRF where I got the expected signal.

However, as soon as I switched back to the antenna I used previously, the harmonics in the spectrum were visible once again. The next suspicion I had was the very high gain setting in the source block. I did not notice this earlier but when I saw it I was suspicious since the gain on the HackRF is lower to begin with. The gain was set to 64dB which is probably useful when looking at very weak signals, but since I am sitting right next to the receiver when transmitting I might overload it and thus harmonics are falsely measured. Before testing that I added a scope sink in the flowgraph, because I was curious wether or not I am able to see something in the time domain plot.

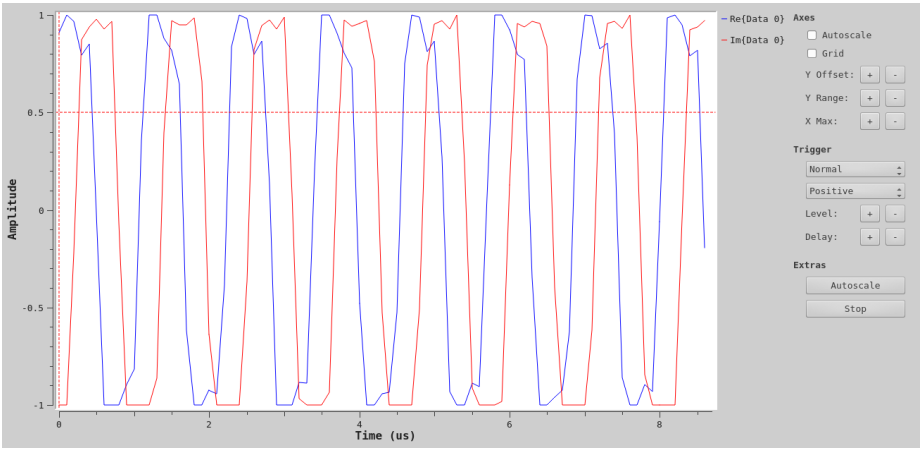

Figure 5: The time domain as captured by the Pluto SDR with too much gain

And indeed the signal is hitting the upper and lower range and does not look sinusoidal at all, the signal was clipping. The maker of the HackRF talks in his tutorial about reducing the gains of the HackRF source block as a starting point and increasing them if necessary. That would have been useful in this case too. First I went too far, reducing the gain to 16dB. The signal was barely above the noise level at this point. I kept increasing it, back to 50dB which seems to be a nice value for this setup. In the future I would recommend starting with a lower gain and increasing it if necessary.

To prove that this change did indeed result in a clean spectrum I switched back to the fft sink and I will just let this screenshot speak for itself.

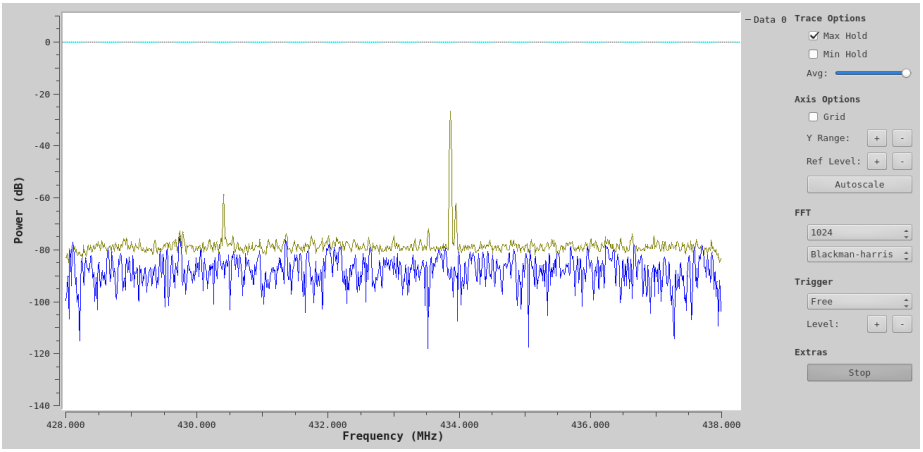

Figure 6: The time domain as captured by the Pluto SDR. This time the gain of the receiver has been reduced so the signal is no longer cut off.

4 Replaying The Signal

4.1 Preparation In Hard- And Software

To transmit I though let's start carefully. I added my "Random RF Attenuator" with about 27dB of attenuation at 433MHz between the Pluto SDR and the antenna.

On the software side I followed the tutorial from the HackRF maker, even though I was using the Pluto. The tutorial covers the basics very well and thus can easily be adapted to different SDRs. I started with a file source, a throttle block and a fft sink. Since I captured the signal with the HackRF the peak in the center of the spectrum was clearly visible. I could recapture the data, which I might have to do anyways, I'll talk about that later. Instead I chose to first boost the signal by multiplication with a constant and then filtering it with a rather sharp bandpass filter. I was confused at first when I saw the resulting spectrum, but then I realized (or remembered) that I had to switch the filter taps to complex in order to let only one side of the spectrum pass.

Another thing I found was the "DC block" block. It works even better to suppress the DC offset of the HackRF that results in the peak at the center frequency. The length of the blocker sets the number of taps. More taps means sharper (in bandwidth) and therefor better attenuation of DC signals.

4.2 Transmitting?

I tried to "simply" connect the output of the prepared flowgraph to a Pluto SDR sink. I adjusted the attenuation of the block, since that is what can be set on the sink. I adjusted it to 0dB since I used the external attenuator I recently built. With 27dB my external (and proven to work) attenuator dampens the signal more than the standard 10dB of attenuation the Pluto sink has by default. At the same time I added a Pluto SDR source that I connected to the fft sink to observe the output.

Let's just say it's not that simple. The signal (the filtered one) was only sometimes visible in the spectrum. I remembered the man in the tutorial talking about USB 2.0 specifications not being up to the full 20MHz bandwidth. I already sampled at "only" 10MHz in order to keep the bandwidth down. So I removed the receiving part to test if that would help, but the relay was not reacting.

Just like before I thought to myself: Maybe it's another weird thing the Pluto SDR does? But changing to the HackRF it did not work as well. After partially successfull attempts at reducing the sample rate with decimation I had enough and decided to remove blocks of the flowgraph. Indeed after removing the filter block, the signal was transmitted successfully and the realy did switch on, but not off.

After this many setbacks I had to take a break and continued a few weeks later.

5 Second attempt - with faster hardware

5.1 Hardware

I used my desktop PC and the Pluto SDR this time. I have chosen to go with the PC since it has much more processing power so the only limit on the data rate should be the USB cable (as long as I keep filtering and other signal processing within reasonable limits).

I had some troubles setting up the industrial io blocks for gnuradio which are necessary to use the Pluto SDR device. The trouble came from the fact that I am running Manjaro on my desktop which is an Arch based os, but the references from Analog Devices only work for Ubuntu based versions. After some searching on the internet I found some promising blog entries and AUR packages. It was strange, but the installation process for Arch based systems seems not very well documented in this case. For gnuradio and the osmocom blocks the setup was no problem.

Figure 7: The test system set up in my room. The transmitter antenna might be hard to see against that background. I used a big metal box as ground for the receiver antenna.

5.2 Testing the setup

Using this hardware I came up with a simple flowgraph to test the receiving and transmitting capabilities of the setup.

As before I reduced the gain on the receiveing side. Instead of the built attenuator I used the builtin attenuator of the Pluto SDRs transmitter for this

test. I set the attenuation to 30dB and reduced the gain (receiver) to 40dB. I also had the antennas for transmitter and receiver about a meter appart.

Due to the bandwidth limitation of the USB connection, the highets sampling rate for full duplex operation was 6 Msps for my setup. This leads to 12 Msps over the USB port which is actually very close to the advertised speed.

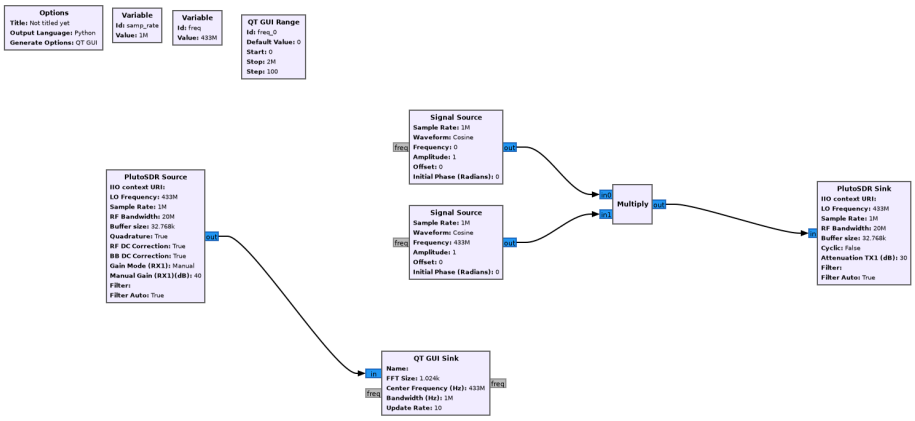

Figure 8: I tested the system with this flowgraph. It sends a signal of variable frequency and records its own transmission.

5.3 Capturing a new signal

I decided to capture the signal once again, this time using the Pluto SDR. There is not much to say about the process. I used a simple two block flowgraph with the source and a file sink. To limit the amount of data I set the sample rate to 1 Msps and the center freuqency to 434 MHz.

5.4 Inspecting The Signal

For a replay attack it is not stricktly necessary to know anything about the signal (except for the bandwidth maybe). For the learning experience that this is all about however, I'd like to find out a bit more like the encoding that's being used.

To view the signals different sinks are available. The fft sink shows the signal in the frequency domain, while the scope sink (or time sink) displays the signal in the time domain. More advanced sinks like the waterfall sink show a signal similar to the one from the fft sink, but also how the frequency distribution changes over time. There is also the option to have multiple sinks in one GUI of course.

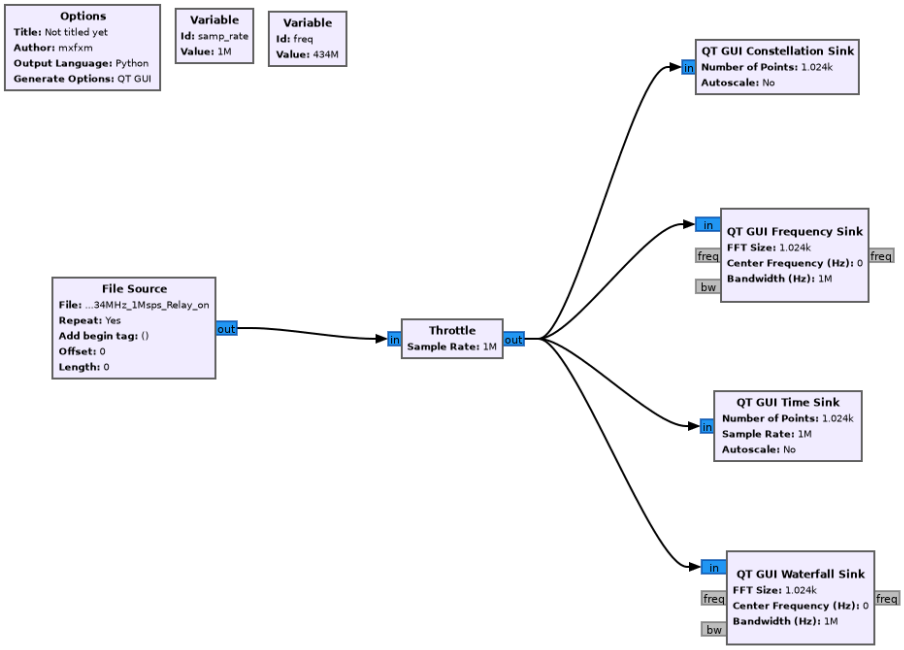

Figure 9: The flowgraph I used to inspect the signal. It uses different instrumentation sinks for QT. Different modulations show very distinct patterns in the different domains.

With the QT Gui there is also the "universal" sink. This allows to switch between different sink types during runtime. That way the signal can be inspected in the different domains very easily. The downside is that this sink can not be configured with triggers or other parameters, so I have chosen to go with the other sinks instead.

From the previous investigation we already know that the signal uses only a single frequency, so frequency modulation is not used. In the waterfall display a repeating pattern can be seen while the button on the remote control is being pressed. In the constellation plot it was visible that the amplitude of the signal remains constant during each transmission.

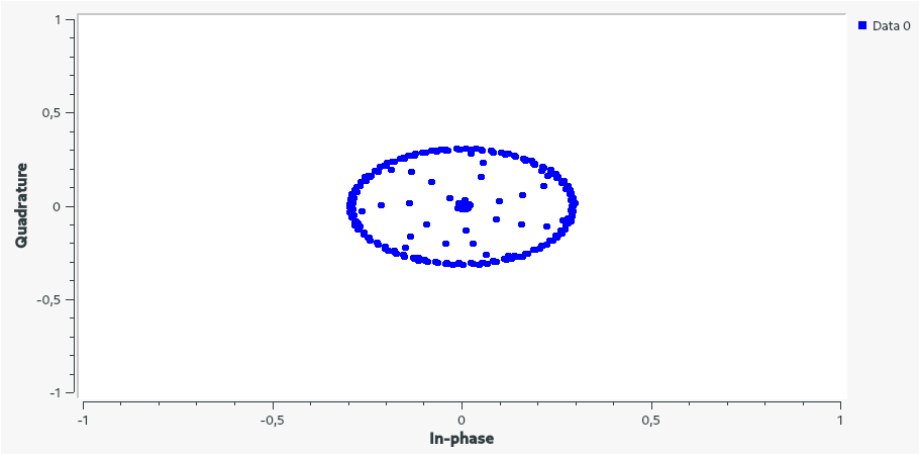

Figure 10: The constellation view of the signal. In this case it is a circle which shows at least that there is no amplitude modulation. The circle also has a dot in the middle, so there is a time where the signal is off.

The time domain plot was the most useful in this case. A pattern of bursts was visible. This kind of modulation is called on off keying, where the carrier waveform is switched on and off. The different bits (0 and 1) are represented as short or long bursts, similar to the morse code. One of the bits is representative of the state of the realy and turns it on or off. The other bits represent the channel A to D and the different possible combination of the DIP switches.

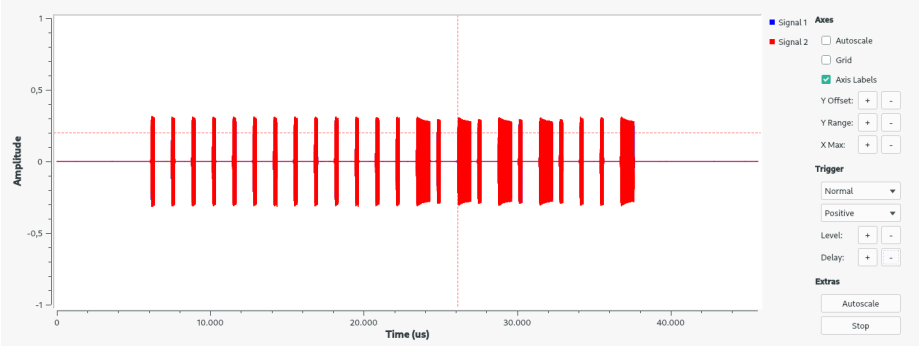

Figure 11: A view of the on off keyed signal in the time domain. The trigger was set up to capture one full frame. Different length of the signal burst can be seen.

I noticed one additional bit at the end of the message on some frames, but not on others. I suspect that it is a way of telling the receiver whether this was the last frame or if it will be repeated again. The frame is repeated with this additional bit while the button on the remote is pressed and once without after the button was released.

5.5 Transmitting!

I am finally transmitting the signal using the Pluto SDR. The simplest flowgraph to send a signal is to connect the file source directly to the SDR sink.

It is however possible, that the signal does not have enough dynamic range for the SDR to transmit it properly. This has nothing to do with the attenuation on the transmitter but only the effective number of bits the signal is using. By multiplying the signal with a constant value in the digital domain I am able to boost this.

Now it is all working as intened.

5.6 Transmitting cleaner

Although this would already be enough, the signal I am transmitting is not very clean (in the spektrum it is using). That means that I am not only transmitting the signal, but also all the noise I captured within the 1 MHz sampling rate I set up.

To counteract this a fist step to take is using a bandpass filter and only letting the frequency of the signal through. This is done with just one other block, a bandpass filter. By using complex taps I am able to pick only one side of the spektrum and not its mirror image at negative frequencies. Alternatively a lowpass can be used, if the signal is right at the center of the captured waveform.

This is still not the cleanest way to transmit. The reason being, that the noise in the frequency band of the signal is still transmitted even when the signal is off (and the noise on top of the signal itself).

5.7 Transmitting even cleaner

By measuring the amplitude of the signal I am able to detect when the on off keying is on or off. A threshold block can be used to convert from amplitude to zero and one. That signal then can be multiplied with the original signal.

That means that the signal, and therefor the noise in that band, is only being transmitted when the signal is on. There is no transmission of the noise in the off slots.

5.8 Cleanest transmission - Synthesized signal

The cleanest signal possible is to synthesize the signal myself. I used the same threshold procedure from above to check if the signal has to be on or off at any time. The result is then multiplied with a clean sine wave instead of the original signal. That way there in no noise when the signal is off and no noise when the signal is on as well.

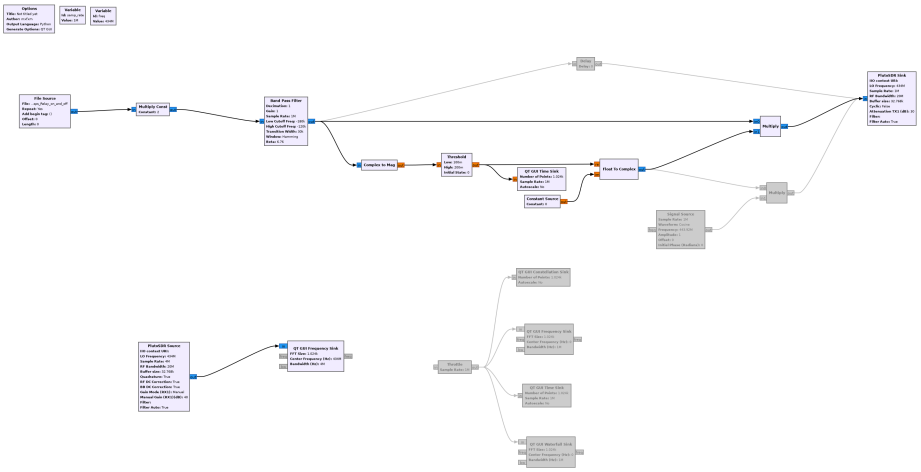

Figure 12: The full flowgraph used to transmit. By enabling and disabling the blocks towards the SDR sink, different methods of cleaning the signal can be compared.

This is the full flowgraph for all described transmitter setups. By switching the signal going into the SDR sink the method can be selected.

There is one thing left to say: Cutting of the signal with the rectangle from the threshold block is not clean in the spektrum. The sharp edges introduce harmonics. It would be better to multiply the signal with some kind of curve, like a gaussian bell, to counteract those sharp edges. Also another filter could be used after the multiplication with the rectangle. This method however is not present in the displayed flowgraph so the cleanest way of these would be just getting the signal after the bandpass.

6 A possible use case

A possible use case that is not illegal would be to capture the signals of all four channels A to D of the remote for all possible DIP switch configurations. That way a program could control the relays and I would not need the remote control anymore. The program could be controlled via home automation software such as node red. Depending on the command, the program would chose the respective stored signal and replay it.

I will not do this though, due to the relatively high cost of a SDR transmitter which would be permanently in use for this application.

7 What I Have Learned

When trying to capture the signal from the remote control I noticed a very ugly spectrum with the Pluto SDR which I did not get with the HackRF. The standard antennas delivered with the Pluto SDR did not show this behaviour, but the antenna I originally used was fine with the HackRF. It turned out to be the gain setting of the Pluto SDR being too high, in a way that with the "good" antenna it was overpowering the receiver and introducing harmonics into the signal which were visible in the spectrum.

Using complex filter taps the filter will only filter one side of the spectrum. In other words a filter with only real taps will have a mirrored effect on negative frequencies. I did not really learn this here, but I remembered it when I saw the specturm with a filter that uses real taps. I would have forgotton about that otherwise, so I count it as learned.

The old laptop I am using is limiting my SDR capabilities. At least when it comes to filters that use a considerable amount of math for the MAC operations (multiply accumulate). This is a basic operation for digital signal processing but it takes a lot of processing power. When I removed the filter the laptop was fast enough, but that is not a clean way to transmit signals. However, the USB connection between the SDR and the PC is another bottleneck. It is not possible to take advantage of the full bandwidth of my SDR devices due to this fact. The 20 Msps in both directions simultaneously are not supported.

I also learned about how to clean up the signal you are transmitting. The idea of the bandpass filter was explained in the tutorial, but seeing it myself makes another impression of how good it works and how useful it is (in the spektrum). I came up with the "threshold filter" idea by myself after I discovered that block. It works very nice, however when it is not layed out properly the sharp edge when the signal turns on might introduce additional harmonics to the otherwise clean signal. Workarounds are another filter after the multiplication or adding a window function like a gaussian curve to smoothen the edges from turning on and off.