# SDR Assignment number 2 (Walkie Talkie)

**Author: Serafim Ciobanu**

**Date: 31.12.2024**

**Course: Emergent Security Techniques**

**Disclaimer: I will try to explain most of the things that have been done in this lab, the way I understand it, and it might not be the best explanation from various sides.**

Description: We recently captured the IQ SDR file that is linked to below using a HackRF receiver, and we suspect that a secret code can be found within.  Can you find this secret code?  The following information about the transmission has already been gathered.

* center frequency of the capture:445 MHz
* sampling frequency of the capture:10 MHz
* Rumors are that the subject uses an**Alecto PMR446** device

Here is the flow graph that works in my case of the decoding.

A diagram of a diagram of a person

Description automatically generated with medium confidence

This graph allows me to change to some different sample rates from the original one, but overall, it allows me to hear the specific frequency where the code is being transmitted.

Now is the hard part regarding the block explanation

A close-up of a white box

Description automatically generated

First blocks are **Variable,** and **Options**. **Variable** allows me to set variables that can be then used further in the graph, while **Options** represents the various settings for the environment, like the name of the python executable, and then the name and description for the files.

A screenshot of a computer

Description automatically generated  
The next bunch of blocks are **File source, Throttle, Signal Source** and **QT GUI Range**.

**File Source** is meant togive the input from the capture further along the graph. It does not have any specific parameters, except I found out that unlike the FM radio, here we already have enough signal/output to actually hear the needed compartment, or we have to barely tune anything.

**Signal source** creates additional sound that allows us to tune to the right moment when the code is spoken.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In my case the **Signal Source** and **QT GUI Range** is linked, so that I can have more information on the frequency.

**Throttle** limits how much actual “data” is sent to be processed by my CPU.

After all of this, this values are all added together into one sound wave via **Multiply** block.

The initial values for Time and Frequency Sink visualization can tell us a lot about what is going on.

A graph on a white background

Description automatically generated

A screenshot of a computer

Description automatically generated

This values tell me that I am almost right, by default, and as far as I can understand the main information gets transmitted in the 446 MHz wave, which means I would need to lower or up myself on the line.

Next block is **Low Pass Filter** which is needed to get rid of all unnecessary noise out of the capture. In this case the hardest part was to get the values right, because it would also not want to compile sometimes.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

The decimation was set to 100000 because the initial value that we are sending in is 10MHz, and we need to leviate it, and afterwards have the frequencies cut off (this values can be changed but I left them as is because the sound is pretty clear).

The gain does not really help in this case because it was meant to up the sound level a bit.

A blue lines on a white background

Description automatically generated

Another part of the manipulation of the sounds is Demodulation with the **FM Demod** block, which is meant to yet again spread our signal further away and allow us to work on a wider range of motion.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Channel rate was set to 100000 as a matter of the low pass decimation (10000000 / 100 = 100000)

Decimation was set to 10 in order to lower the value even further to 10000, while Audio Pass and Stop was meant to let or stop certain frequencies from being processed. Deviation and the other values were left default.

Last important processing blocks are **Rational Resampler** and **Audio Sink**.

A diagram of a computer flow

Description automatically generated with medium confidence

In this case **Rational Resampler** is meant to manipulate the number of the frequency that comes in to be reproduced by the hardware.

Since I want to get to 48kHz sound, I multiplied the 10000 after demodulation by 48 to get to 480000 and then decimated it by 10 to get to 48000 = ~48kHz for the output.

A screenshot of a computer

Description automatically generated

In the end this is how the settings look to get the secret code transmit in my case:

A screenshot of a computer

Description automatically generated

The Initial time sink tells me that the lower it was, the better the probability to hear the code.

**The code is 17935.**That seems to be the end of all possible adjustments I wanted to make for this assignment.