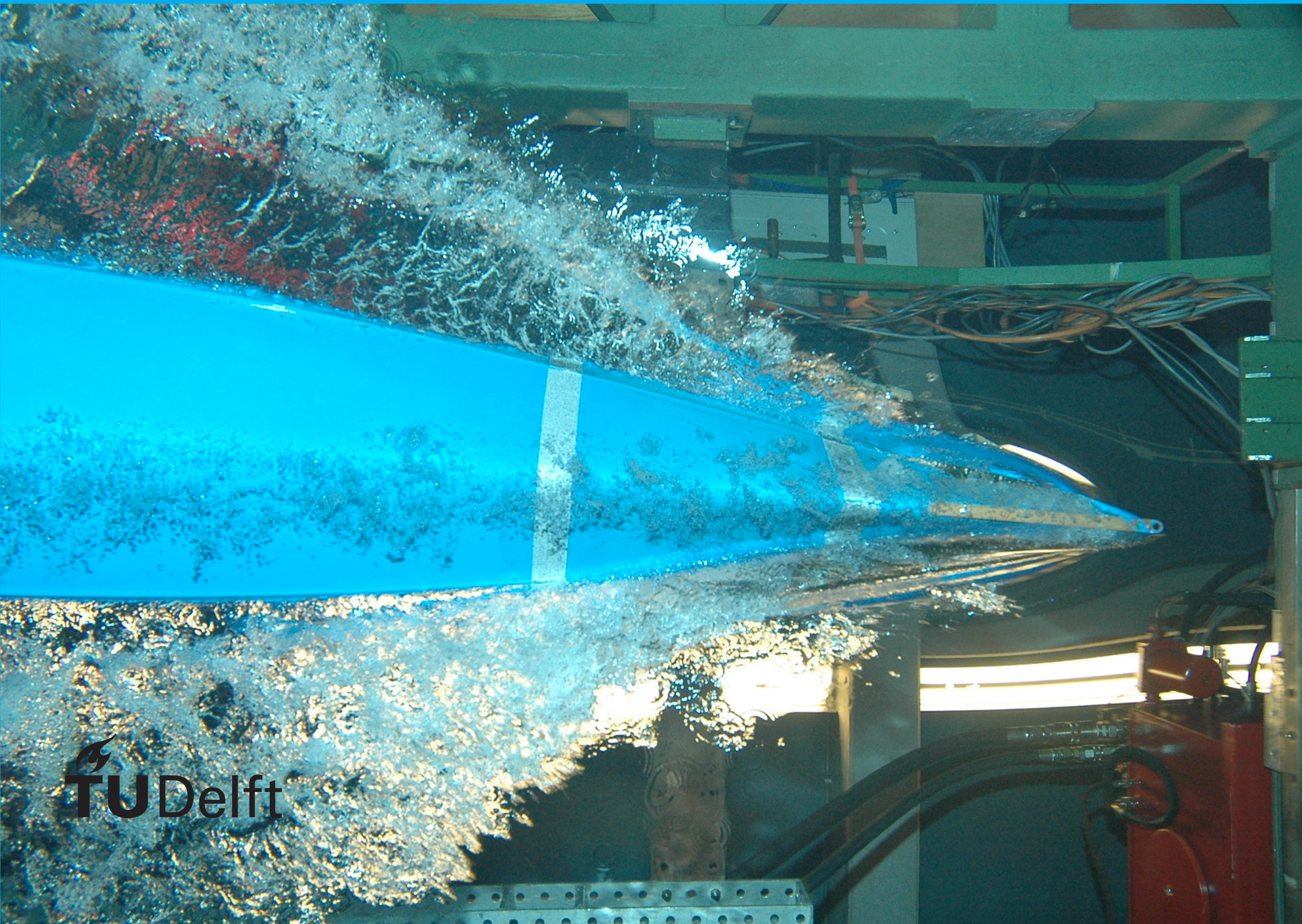
A FSK-decoder in GNU Radio

J.B Miog



A FSK-decoder in GNU Radio

by

J.B Miog

.

Student number: 4336313
Date: April 2016

TU Delft

Contents

1 Background

The transceiver module that is being used has been build by the author in order to fulfill the degree of bachelor of applied sciences in 2013. The transceiver module is designed at, and currently produced by, T-Minus engineering B.V. It contains two chips, an Atmega Arduino compatible micro controller and a transceiver chip, the Murata TRC105. The module can be connected to a computer via USB, after which a Arduino sketch can be uploaded.

At the moment, this transceiver is used by ESA during the CanSat competition, in which multiple high-school teams create mini satellites the size of a soda can. The mini satellites are launched in a rocket, up to a height of 1,5 kilometer. Then, the satellites are decoupled and should open a parachute to safely land. During descending, measurements are being done by the satellite. In order to simulate a space mission as good as possible, the data of the measurements is to be wireless transmitted during the 'mission'.

While the teams on the ground aim Yagi antenna's on their CanSat's to receive the signal, every once in a while, they don't succeed in receiving anything, while the CanSat is thought to be transmitting.

The goal of this project is to build a receiver which can listen to the channels of each team simultaneously. After receiving, the data has to be decoded. In order to receive in such a wide spectrum, an SDR will be used.

2 Transmitter side

Two transceivers will be used during this project, both are set to transmitting mode only. They are named 'Buenos aires' and 'Pijnacker' respectivily. The transmitter settings are show in the table below.

	Tx Freq	Deviation	Mode	Tx Data	baudrate	Tx power
Pijacker	432.9mHz	+/- 20kHz	FSK	'Pijnacker'	1200 symb/sec	-8dBm
Buenos Aires	433.62	+/- 20kHz	FSK	'Buenos Aires'	1200 symb/sec	-8dBm

The signal that will be transmitted will contain a preamble of 4 bytes (0xAA). This is followed by the transmitter identifiers, which are the same for both modules. The identifier is "TMCS" (0x54, 0x4D, 0x43, 0x53), followed by a node address byte (0xXX).

3

Receiver side

In order to receive both nodes simultaneously, an Ezcap DVB-T FM DAB receiver is used, which contains the RTL2832 chip. The SDR dongle is connected to the computer by USB2.0, and the I&Q data is imported using the GNU-Radio software.

3.1. Graphical view of the signal

First, figure 3.1 is obtained to measure the frequency offset. The left signal is produced by pijnacer, and should be at center frequency 432.990 mHz, but as we can see, it is actually received at 432.964 mHz. The difference is caused by the initial offsets of the transmitters, and the frequency offset of the VCO in the SDR.

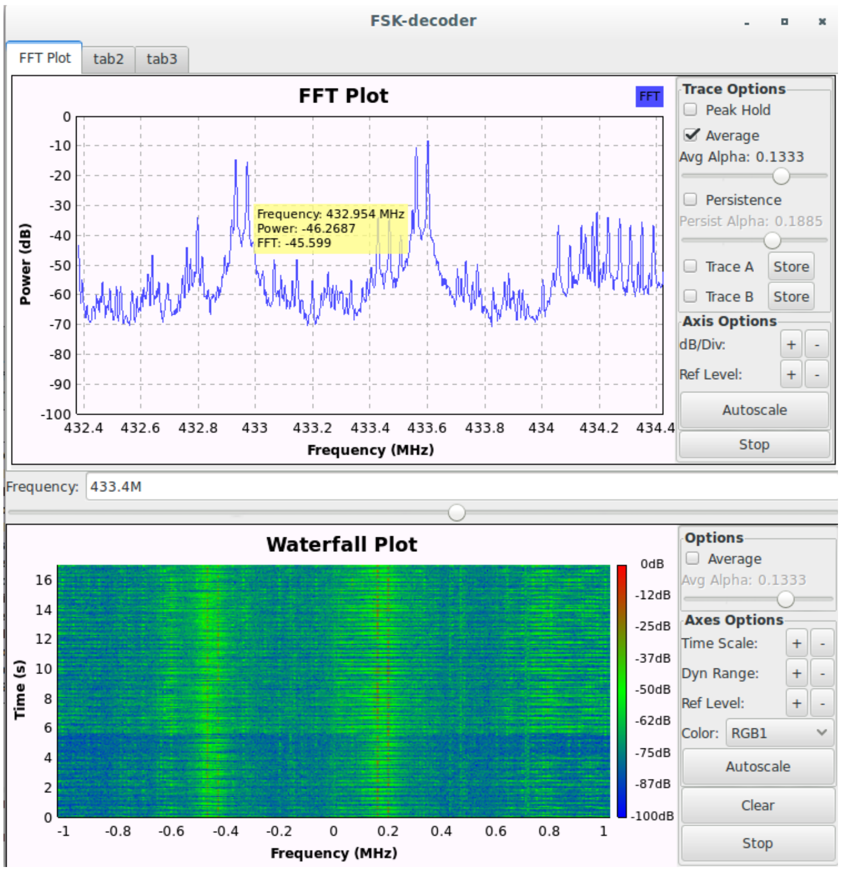
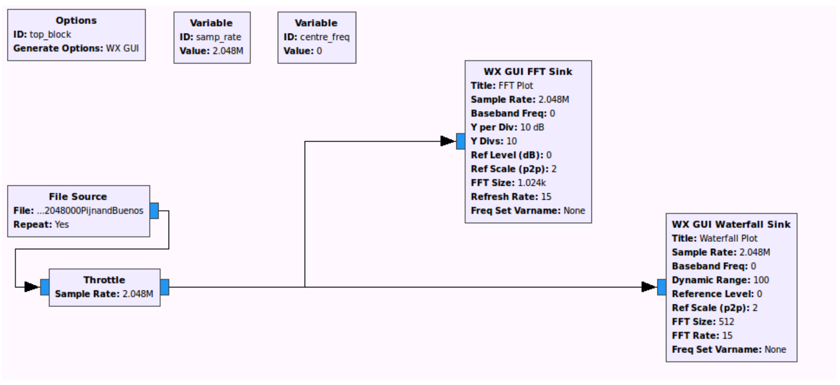


Figure 3.1: a gnu-radio capture of the frequency spectrum

In order to build the decoder efficiently, the FSK_capture_to_file.grc is build, which contains a simple data sink block. The sampling frequency is set to 2.048 MSPS, which results in $2.048 MSPS/1200 baud \approx 1706 samples/symbol$. This is useful to know in order to perform a correct clock recovery later on.

The input data is measured with a center frequency 433.4. We'll take a graphical look at the recorded signal of both transmitters using the GNU radio flowchart depicted in 3.2



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2.048M

Variable
ID: centre_freq
Value: 0

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2.048M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Freq Set Varname: None

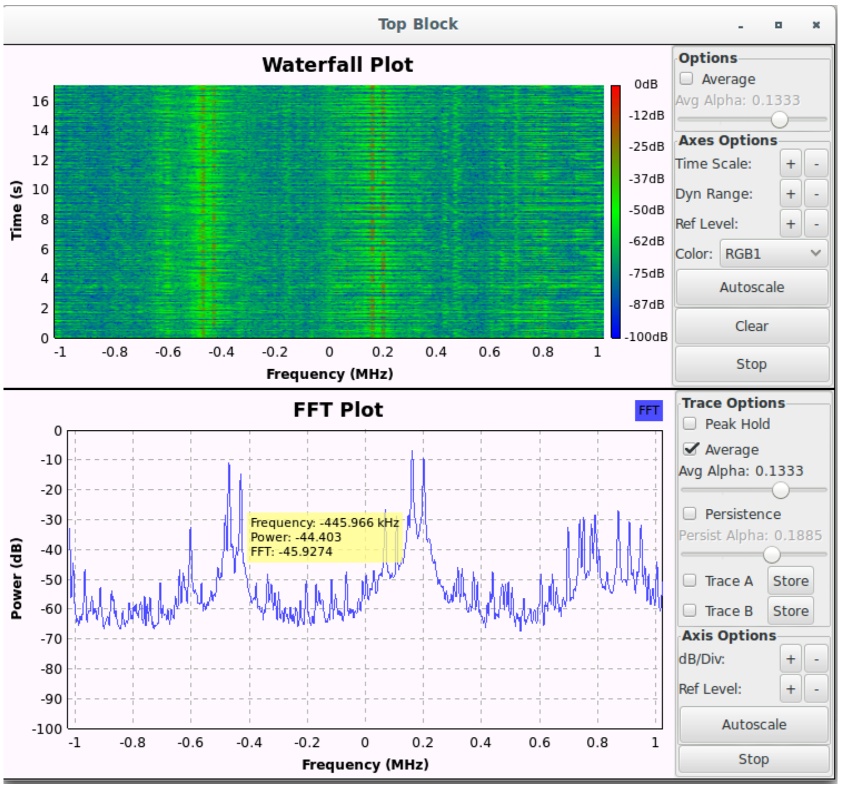
File Source
File: ...2048000PijnandBuenos
Repeat: Yes

Throttle
Sample Rate: 2.048M

WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 2.048M
Baseband Freq: 0
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 512
FFT Rate: 15
Freq Set Varname: None

Figure 3.2

We expect that Pijnacker transmitting frequency is now 433.4 - 432.950 = 450 kHz below the current center frequency. Which is roughly shown in 3.3.



Frequency: -445.966 kHz
Power: -44.403
FFT: -45.9274

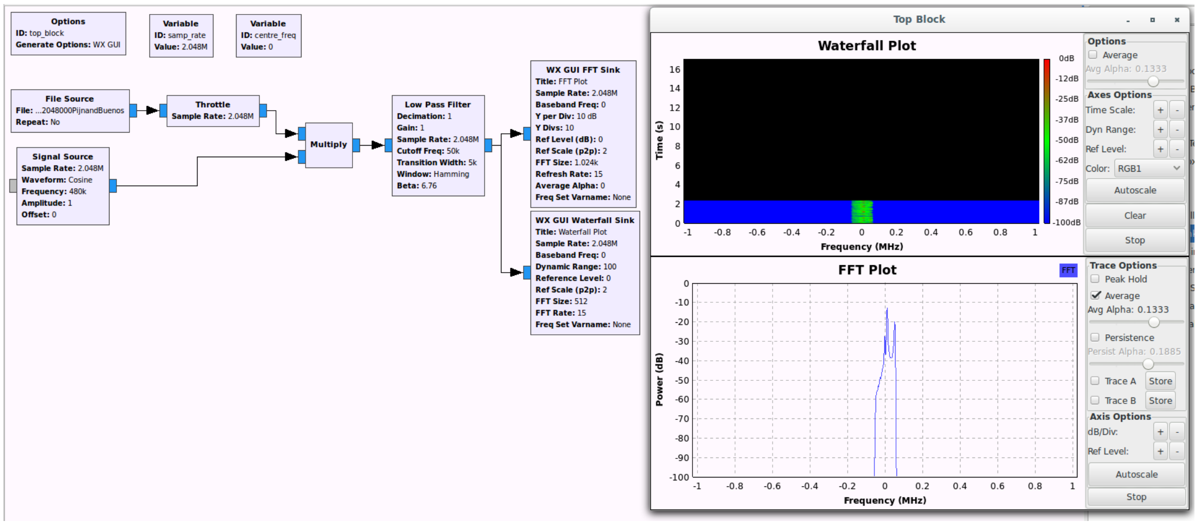
Average
Avg Alpha: 0.1333
Color: RGB1
✓ Average
Avg Alpha: 0.1333
Persist Alpha: 0.1885

Figure 3.3: replay of signal

Once we replay the saved data using the file source block, the frequency of Pijnacker is at -450kHz, which can be seen in figure 3.3.

In order to decode the signal, the frequency is increased using a multiplier and an addition signal source block, with a frequency of 470 kHz. This sets the '0' at + 30kHz, and the '1' at 50 kHz. Then, the signal is passed through a low pass filter. The flowchart and the results are depicted in **??**.

From this point, we could use the GFSK decode block already availabe in the GNU-radio toolbox. However, this block only supports binairy data output which is unfortunately not easy readable. To gain a better



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2.048M

Variable
ID: centre_freq
Value: 0

File Source
File: ...2048000PijnandBuenos
Repeat: No

Throttle
Sample Rate: 2.048M

Signal Source
Sample Rate: 2.048M
Waveform: Cosine
Frequency: 480k
Amplitude: 1
Offset: 0

Multiply

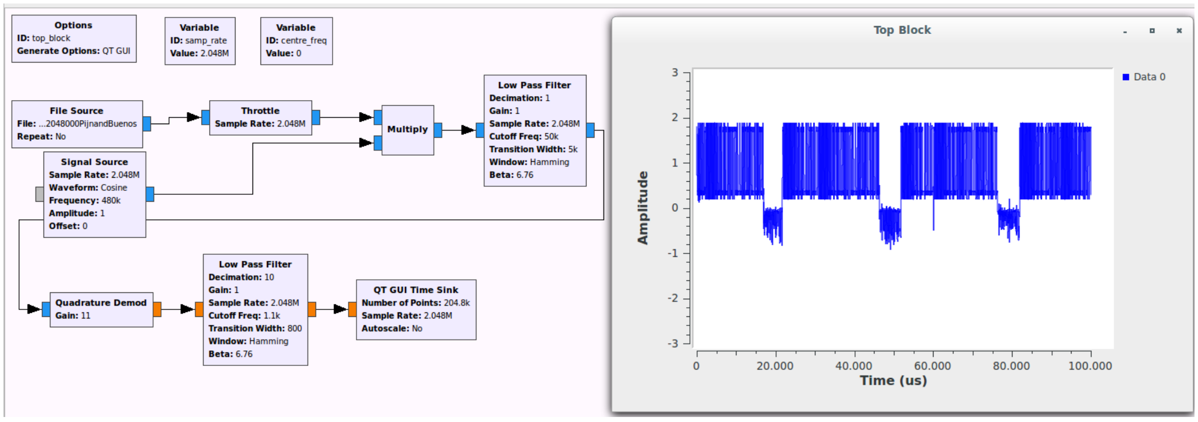
Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 50k
Transition Width: 5k
Window: Hamming
Beta: 6.76

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 2.048M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Average Alpha: 0
Freq Set Varname: None

WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 2.048M
Baseband Freq: 0
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 512
FFT Rate: 15
Freq Set Varname: None

understanding in the underlying principles, it is decided to use a quadratude demodulator, and look decode the signal in a binary AM fashion. After adding the quadrature demodulator, another low pas filter is used. with a cutt-off frequency of 1200 hertz. As the baudrate is 1200 symbols a second, and each symbol represents a binary '1' or a binary '0', we know that a signal representing a bit in AM can never be at a frequency higher than 1200 hertz. However, the filter is not infinitively steep, and therefor the cutoff frequency is set to 1100 hertz with a 800 hertz transition width.

The flowchart and the output up to this point are depicted in figure 3.4.



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2.048M

Variable
ID: centre_freq
Value: 0

File Source
File: ...2048000PijnandBuenos
Repeat: No

Throttle
Sample Rate: 2.048M

Signal Source
Sample Rate: 2.048M
Waveform: Cosine
Frequency: 480k
Amplitude: 1
Offset: 0

Multiply

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 50k
Transition Width: 5k
Window: Hamming
Beta: 6.76

Quadrature Demod
Gain: 11

Low Pass Filter
Decimation: 10
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 1.1k
Transition Width: 800
Window: Hamming
Beta: 6.76

QT GUI Time Sink
Number of Points: 204.8k
Sample Rate: 2.048M
Autoscale: No

Figure 3.4: Qaudrature demodulation with noise

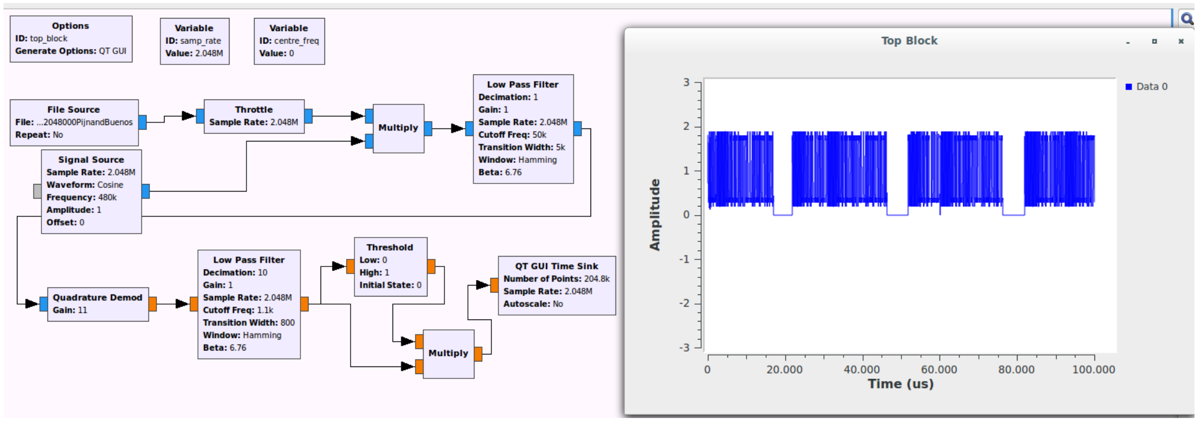
To get rid of the noise in between the data frames, a simple threshold detector is used. The output of the threshold detector is multiplied with the output of the second low pass filer. Again, the flowchart and the result are depicted in 3.5

In order to recover the clock signal from the preamble, a Muller and Mueller clock recovery block ($Clock_recovery_MM$) block is available in the toolbox. This block is not well documented, but the mailing list of gnu-radio provided the information necessary: [1]

Omega: this is the expected number of samples per symbol (SPS) which is 1706 Gain omega: Control loop value indicating how fast adjustment in the omega value can be made. Mu: Initial estimate of the phase of the sample. Gain mu: Control loop value indicating how fast adjustment in the mu value can be made. Omega Relative Limit: Percentage of maximal allowed omega deviation.

The output, after the clock recover block is show in 3.6. Each positive value corresponds to a '1' being detected, a negative value corresponds to a '0' being detected.

In order to clip the amplitude of this output to a '0' or a '1', the binary slicer is used. After the data is appropriately visible, we can add tags using the "correlate access code" tag block which we provide with the



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2.048M

Variable
ID: centre_freq
Value: 0

File Source
File: ...2048000PijnandBuenos
Repeat: No

Throttle
Sample Rate: 2.048M

Multiply

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 50k
Transition Width: 5k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 2.048M
Waveform: Cosine
Frequency: 480k
Amplitude: 1
Offset: 0

Quadrature Demod
Gain: 11

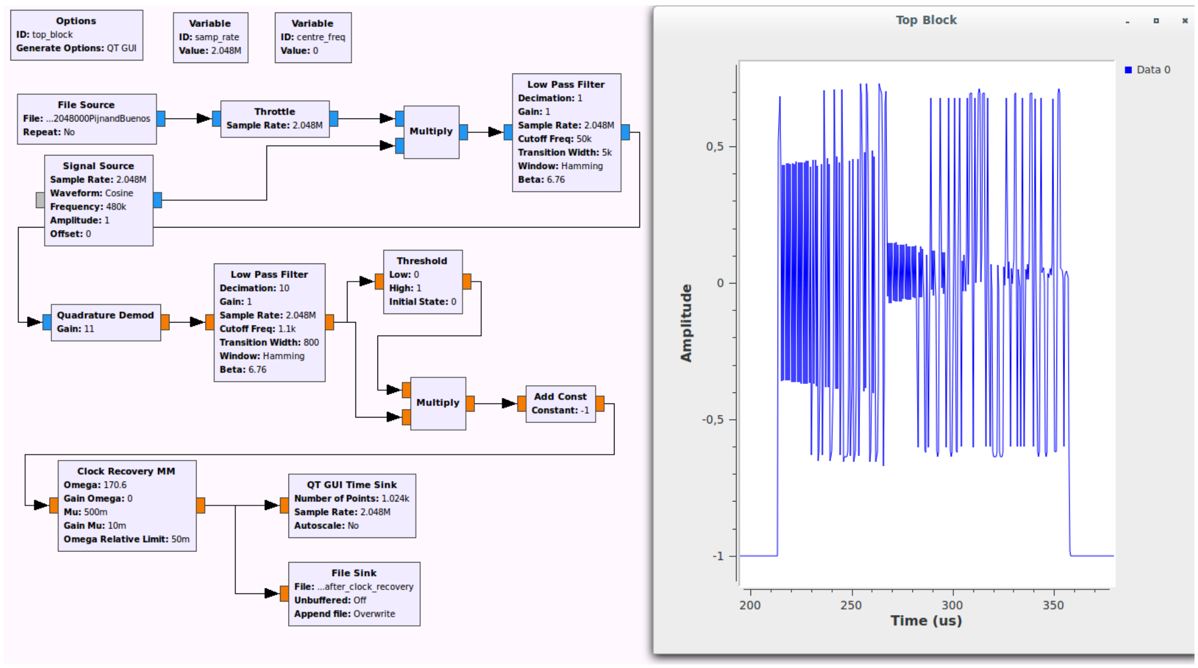
Low Pass Filter
Decimation: 10
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 1.1k
Transition Width: 800
Window: Hamming
Beta: 6.76

Threshold
Low: 0
High: 1
Initial State: 0

Multiply

QT GUI Time Sink
Number of Points: 204.8k
Sample Rate: 2.048M
Autoscale: No

Figure 3.5: inter packet noise removed



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2.048M

Variable
ID: centre_freq
Value: 0

File Source
File: ...2048000PijnandBuenos
Repeat: No

Throttle
Sample Rate: 2.048M

Multiply

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 50k
Transition Width: 5k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 2.048M
Waveform: Cosine
Frequency: 480k
Amplitude: 1
Offset: 0

Quadrature Demod
Gain: 11

Low Pass Filter
Decimation: 10
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 1.1k
Transition Width: 800
Window: Hamming
Beta: 6.76

Threshold
Low: 0
High: 1
Initial State: 0

Multiply

Add Const
Constant: -1

Clock Recovery MM
Omega: 170.6
Gain Omega: 0
Mu: 500m
Gain Mu: 10m
Omega Relative Limit: 50m

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 2.048M
Autoscale: No

File Sink
File: ...after_clock_recovery
Unbuffered: Off
Append file: Overwrite

Figure 3.6: data after clock recovery

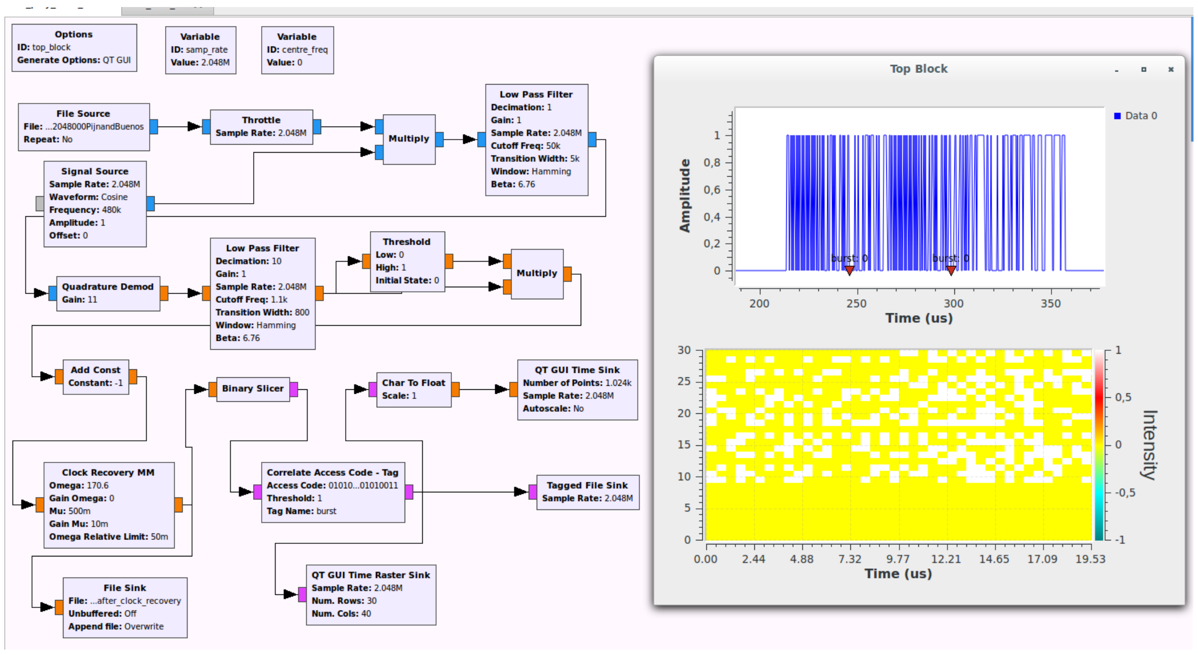
binairy representation of "TMCS". For convenience, the time raster sink block is added to display the data in a longer run. The output of this block is show in the bottom graph in 3.7. Yellow represents a '1' bits detected, and in white the detection of a '0' bit is visible. Unfortunately, the default background color is also yellow. Note that the completely yellow block at the end actually indicates data that is not yet received. The graph on the top shows the tags as small red triangles. These are placed upon the bit stream by the correlate access block.

At this point, also the data is stored in the tagged file sink. The documentation of this block is unclear where the obtained files are actually located, but does specify that each block after a tag detection creates a new file. As this is not eligible for our goal, the author has decided to use the file obtained in fig 3.6 and write a own script in python to detect the preamble, and further decode the message offline.

The python script "fsk_decode_to_char" also implements a binary slicer, appends the '0' and '1' to a string, and searches this string for the data, which it extracts, converts to ascii characters, and outputs as data.

The script skips the first two characters of the transmitted data. It is expected that this is due to the node identifier, as this is not jet taken into account.

[here i'll input the script, and the script output]



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2.048M

Variable
ID: centre_freq
Value: 0

File Source
File: ...2048000PijnandBuenos
Repeat: No

Throttle
Sample Rate: 2.048M

Multiply

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 50k
Transition Width: 5k
Window: Hamming
Beta: 6.76

Signal Source
Sample Rate: 2.048M
Waveform: Cosine
Frequency: 480k
Amplitude: 1
Offset: 0

Quadrature Demod
Gain: 11

Low Pass Filter
Decimation: 10
Gain: 1
Sample Rate: 2.048M
Cutoff Freq: 1.1k
Transition Width: 800
Window: Hamming
Beta: 6.76

Threshold
Low: 0
High: 1
Initial State: 0

Multiply

Add Const
Constant: -1

Binary Slicer

Char To Float
Scale: 1

QT GUI Time Sink
Number of Points: 1.024k
Sample Rate: 2.048M
Autoscale: No

Clock Recovery MM
Omega: 170.6
Gain Omega: 0
Mu: 500m
Gain Mu: 10m
Omega Relative Limit: 50m

Correlate Access Code - Tag
Access Code: 01010...01010011
Threshold: 1
Tag Name: burst

Tagged File Sink
Sample Rate: 2.048M

File Sink
File: ...after_clock_recovery
Unbuffered: Off
Append file: Overwrite

QT GUI Time Raster Sink
Sample Rate: 2.048M
Num. Rows: 30
Num. Cols: 40

Figure 3.7: data after clock recovery, with amplitude correction and preamble tagging

Bibliography

[1] Burst fsk receiver clock synchronization problems. URL `https://lists.gnu.org/archive/html/discuss-gnuradio/2015-12/msg00075.html`.