GNU RADIO

GNU Radio :: **GNU Radio** is a free software development toolkit that provides signal processing blocks to implement software-defined radios and signal-processing systems. It can be used with external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic, and commercial environments to support both wireless communications research and real-world radio systems.

The GNU Radio software provides the framework and tools to build and run software radio or just general signal-processing applications. The GNU Radio applications themselves are generally known as "Flow-graphs", which are a series of signal processing blocks connected together, thus describing a data flow.

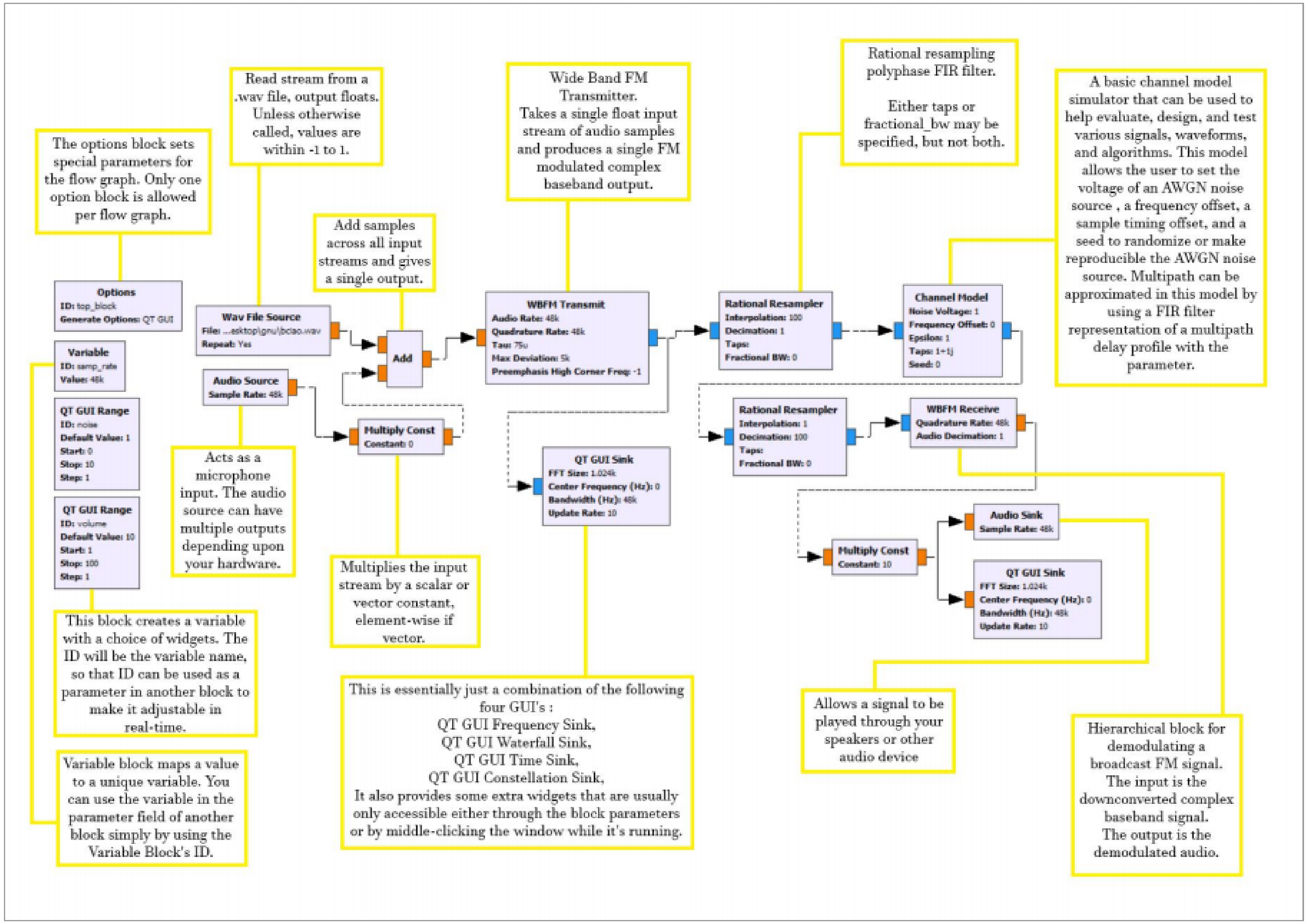
As with all software-defined radio systems, re-configurability is a key feature. Instead of using different radios designed for specific but disparate purposes, a single, general-purpose, radio can be used as the radio front-end, and the signal-processing software (here, GNU Radio), handles the processing specific to the radio application.

GNU Radio Companion :: The GNU Radio Companion is a graphical UI used to develop GNU Radio applications. This is the front-end to the GNU Radio libraries for signal processing. GRC was developed by Josh Blum during his studies at Johns Hopkins University (2006-2007), then distributed as free software for the *October 2009 Hack-fest*. Starting with the 3.2.0 release, GRC was officially bundled with the GNU Radio software distribution.

GRC is effectively a Python code-generation tool. When a flow-graph is "compiled" in GRC, it generates Python code that creates the desired GUI windows and widgets, and creates and connects the blocks in the flow-graph.

PLOTTING AND DISPLAY :: GNU Radio provides many common plotting and data visualization data sinks, including FFT displays, symbol constellation diagrams, and scope displays. These are commonly used both for debugging radio applications and as the user-interface to a final application.

A brief information about all blocks used :: (Note: I have used QT GUI to perform following assignment)



The options block sets special parameters for the flow graph. Only one option block is allowed per flow graph.

Options
ID: top_block
Generate Options: QT GUI

Read stream from a .wav file, output floats. Unless otherwise called, values are within -1 to 1.

Wav File Source
File: ...esktop\gnu\bciao.wav
Repeat: Yes

Add samples across all input streams and gives a single output.

Add

Wide Band FM Transmitter. Takes a single float input stream of audio samples and produces a single FM modulated complex baseband output.

WBFM Transmit
Audio Rate: 48k
Quadrature Rate: 48k
Tau: 75u
Max Deviation: 5k
Preemphasis High Corner Freq: -1

Rational resampling polyphase FIR filter.

Either taps or fractional_bw may be specified, but not both.

Rational Resampler
Interpolation: 100
Decimation: 1
Taps:
Fractional BW: 0

A basic channel model simulator that can be used to help evaluate, design, and test various signals, waveforms, and algorithms. This model allows the user to set the voltage of an AWGN noise source , a frequency offset, a sample timing offset, and a seed to randomize or make reproducible the AWGN noise source. Multipath can be approximated in this model by using a FIR filter representation of a multipath delay profile with the parameter.

Channel Model
Noise Voltage: 1
Frequency Offset: 0
Epsilon: 1
Taps: 1+1j
Seed: 0

Variable
ID: samp_rate
Value: 48k

QT GUI Range
ID: noise
Default Value: 1
Start: 0
Stop: 10
Step: 1

QT GUI Range
ID: volume
Default Value: 10
Start: 1
Stop: 100
Step: 1

Audio Source
Sample Rate: 48k

Multiply Const
Constant: 0

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 48k
Update Rate: 10

Rational Resampler
Interpolation: 1
Decimation: 100
Taps:
Fractional BW: 0

WBFM Receive
Quadrature Rate: 48k
Audio Decimation: 1

Multiply Const
Constant: 10

Audio Sink
Sample Rate: 48k

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 48k
Update Rate: 10

Acts as a microphone input. The audio source can have multiple outputs depending upon your hardware.

Multiplies the input stream by a scalar or vector constant, element-wise if vector.

This block creates a variable with a choice of widgets. The ID will be the variable name, so that ID can be used as a parameter in another block to make it adjustable in real-time.

Variable block maps a value to a unique variable. You can use the variable in the parameter field of another block simply by using the Variable Block's ID.

This is essentially just a combination of the following four GUI's :
QT GUI Frequency Sink,
QT GUI Waterfall Sink,
QT GUI Time Sink,
QT GUI Constellation Sink,
It also provides some extra widgets that are usually only accessible either through the block parameters or by middle-clicking the window while it's running.

Allows a signal to be played through your speakers or other audio device

Hierarchical block for demodulating a broadcast FM signal. The input is the downconverted complex baseband signal. The output is the demodulated audio.

Setup and Installation ::

- The link used to download the GNU Radio Companion - http://www.gcndevelopment.com/gnuradio/downloads.htm
- After successful downloading the above file, just unzip(extract) it and run it.
- After successful installation we just need to open the application(GNU Radio Companion) and we can start working on it.

About the Interface of the application ::

(Note: None of the given information is official, its just written in user friendly language by an basic level user)

- Upwards we get a toolbar containing many options, you can save your file, edit and process many different operations.
- Downwards we get a kind of **Compiler** which auto-saves our work and execute it in search of errors. It shows error messages when anything require some changes else it shows the output.
- On the right Side we get the **Module Box** from which we can drag and use the various function blocks for doing several things.
- On the Center of the window we get area to work(worksheet/work area).

General Information(GYAN) to work fluidly on the application ::

(Note: These are some information for beginner usage of GNU Radio Companion by an basic level user)

- To create a new file, open file menu -> click on New -> select GUI u want to use (or directly select New icon).
- To save file, open file menu -> click on Save -> Choose Location & save (or directly click on Save icon).
- To Run, from menu select Run -> execute (or directly click on execute icon).

Procedure used and some more information about the blocks used to perform Transmission of an FM wave ::

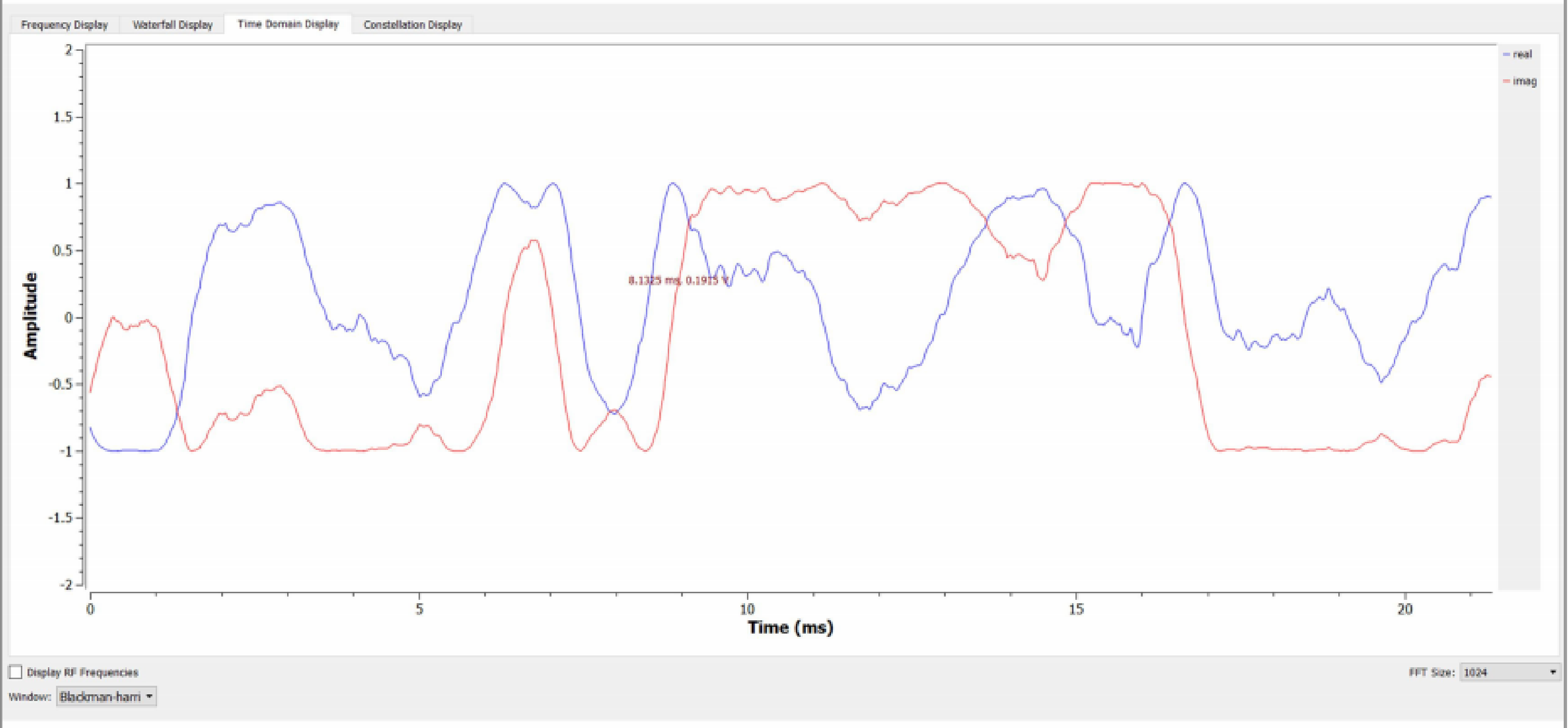
(Note: We have used an .wav(converted from an random mp3 song) sound file as an input to the FM Transmitter)

- Initializing Generate Option from **Options block** to QT GUI and setting Sample Rate i.e. Value in **Variable block** as 48k (which is the sample rate of the audio file used).
- Converted an .mp3 file (bciao.mp3) into .wav file(bciao.wav) and importing it into GNU Companion using **Wav file Source Block** (which generally converts audio sound file into float).
- Also initializing an **Audio Source block** with variable sample rate and attaching a **Multiply Const Block** with Constant value zero, so that the Audio Source cancels out, which results in stabilizing source audio.
- Inserting an **Add Block** which simply adds **Wav file Source Block** with **Audio Source Block** and **Multiply Const Block**.
- Now connecting the transmitter i.e. **WBFM transmit Block** with setting up the Max Deviation to 5k, Audio rate and Quadrature rate to variable sample rate. WBFM Transmit Block simply convert the signal into modulating Signal. For this purpose we can also use **NBFM Transmit Block** but the only different is that NBFM uses small deviation.
- After the above point we have displayed the output through **QT GUI Sink Block** (all outputs are attached at the last).
- Next, we have connected **Rational Resampler Block** which works as a FIR filter while demodulating the above wave(for first Rational Resampler Interpolation is set to 1 and Decimation is set to 100).
- For adjusting Noise in the above signal we have used **Channel Model Block** with Noise Voltage set to an variable(noise), which is set as ID of the **QT GUI Range Block**(it works as an slider to adjust the noise accordingly).
- Going further we have connected one more **Rational Resampler Block** (this time Interpolation is set to 100 and Decimation is set to 1).
- To **Demodulate** the signal we have used **WBFM Receive Block** in which Quadrature Rate is set same as sample rate and Audio Decimation is set to 1.
- We have set an Multiply Constant Block using parameter-variable(volume), so that we can adjust the volume of the final audio signal using **QT GUI Range Block** of ID (volume).
- For getting Output Audio we have used **Audio Sink Block** with same sample rate as variable block, and for displaying the final output signals we have used **QT GUI Sink Block** (all outputs are attached at the last).

OBSERVED OUTPUTS ::

➢ **Output observed for Modulated Signal in Time and Frequency Domain are follows ->**

MODULATED FM SIGNAL IN TIME DOMAIN

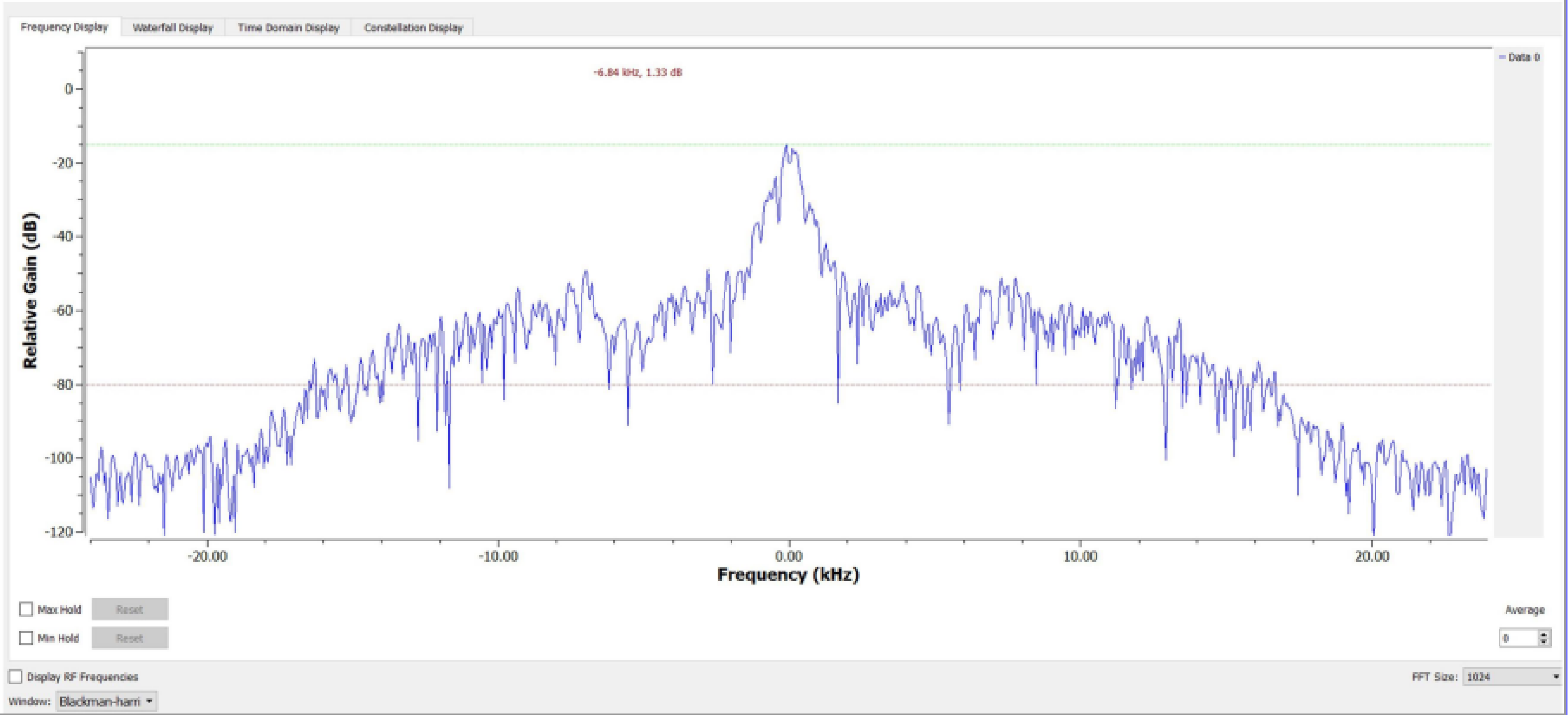


Display RF Frequencies

Window: Blackman-harri

FFT Size: 1024

MODULATED FM SIGNAL IN FREQUENCY DOMAIN



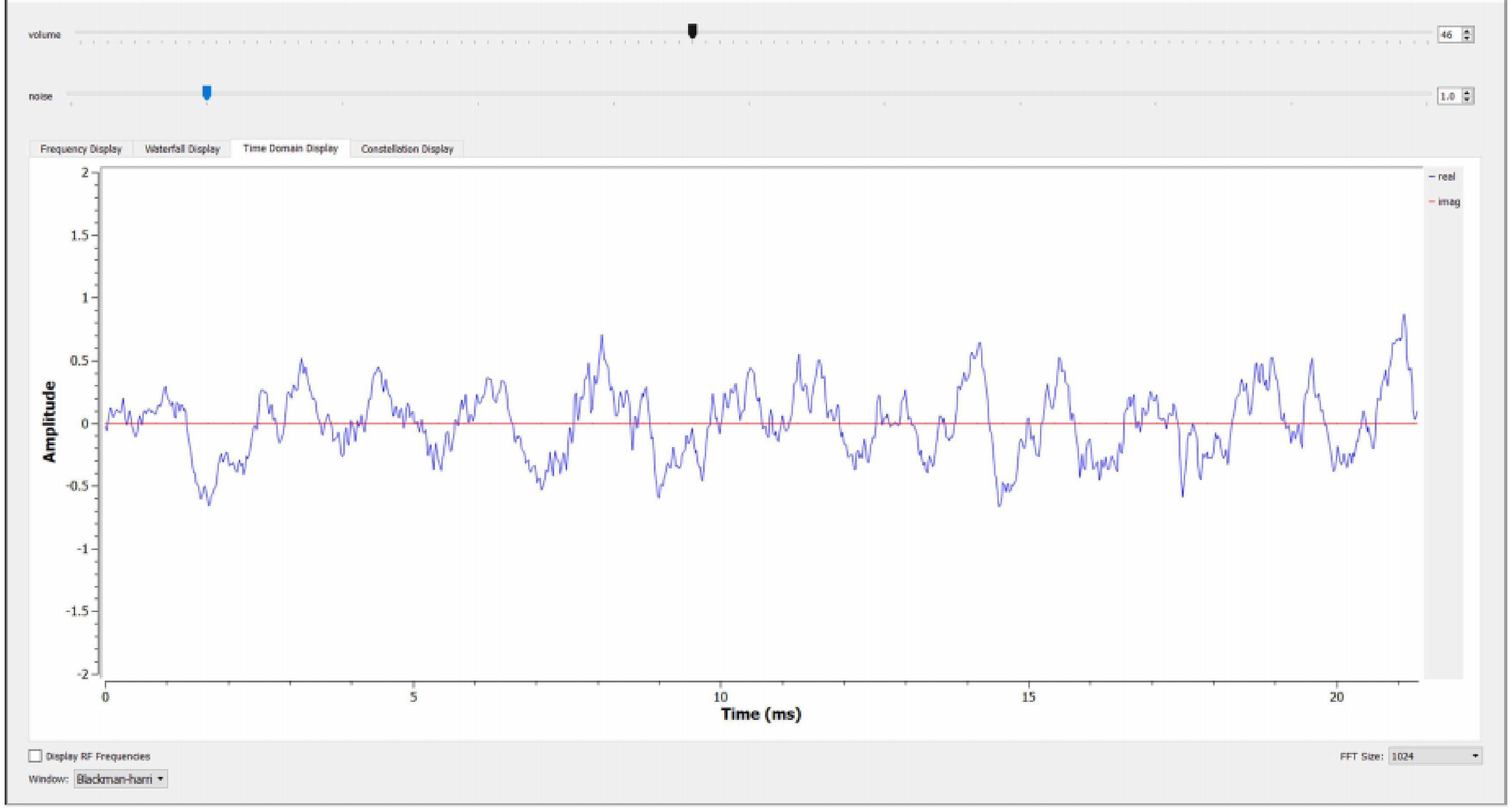
Average: 0

FFT Size: 1024

Window: Blackman-harris

- **Output observed for Demodulated Signal in Time and Frequency Domain are follows ->**

DEMODULATED FM SIGNAL IN TIME DOMAIN



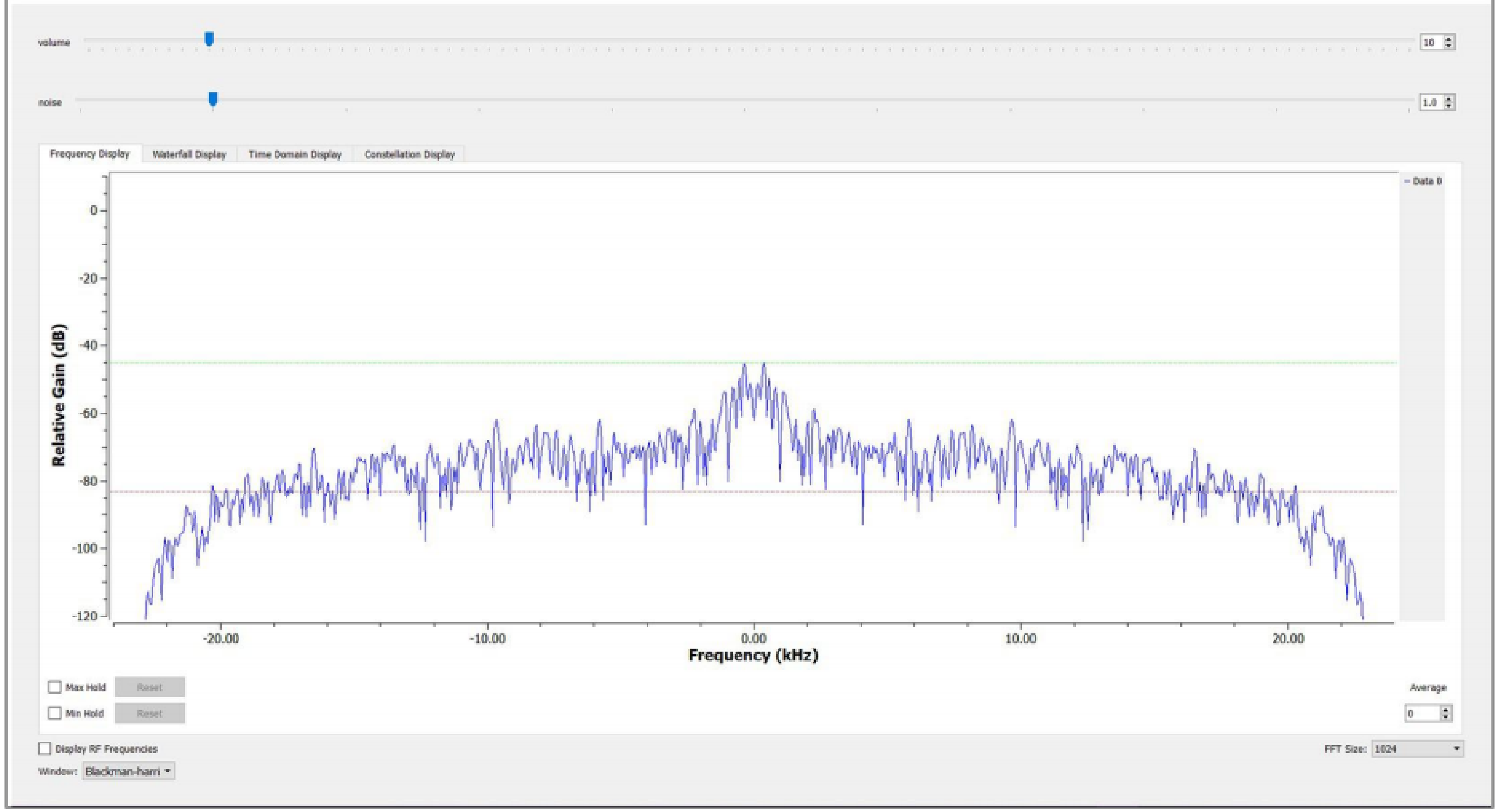
volume 46

noise 1.0

Window: Blackman-harri

FFT Size: 1024

DEMODULATED FM SIGNAL IN FREQUENCY DOMAIN



volume 10

noise 1.0

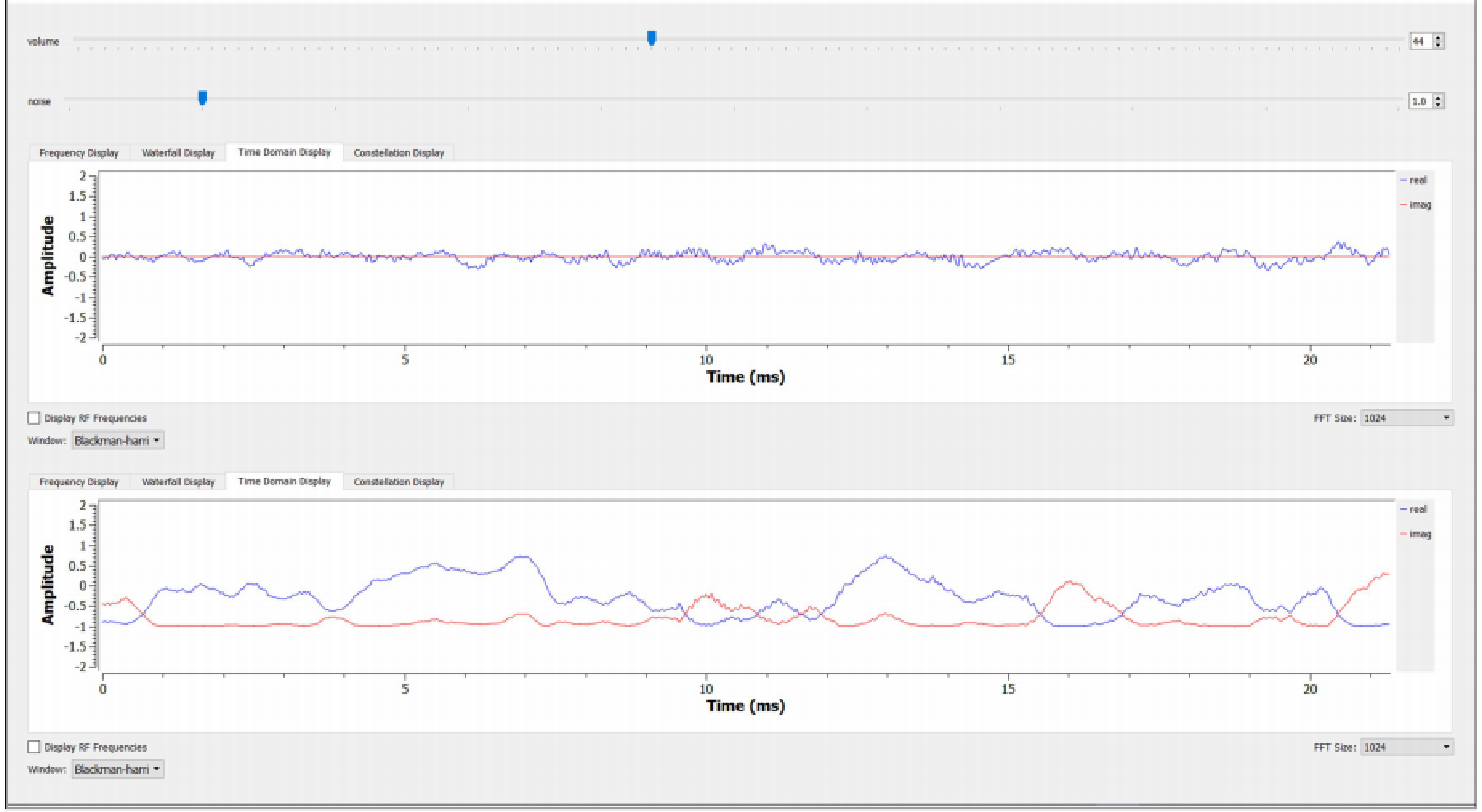
Average 0

Window: Blackman-harris

FFT Size: 1024

➢ **Final outputs observed for Modulated and Demodulated FM signal in Time and Frequency Domain are as follows ->**

FINAL OUTPUTS IN TIME DOMAIN



volume 44

noise 1.0

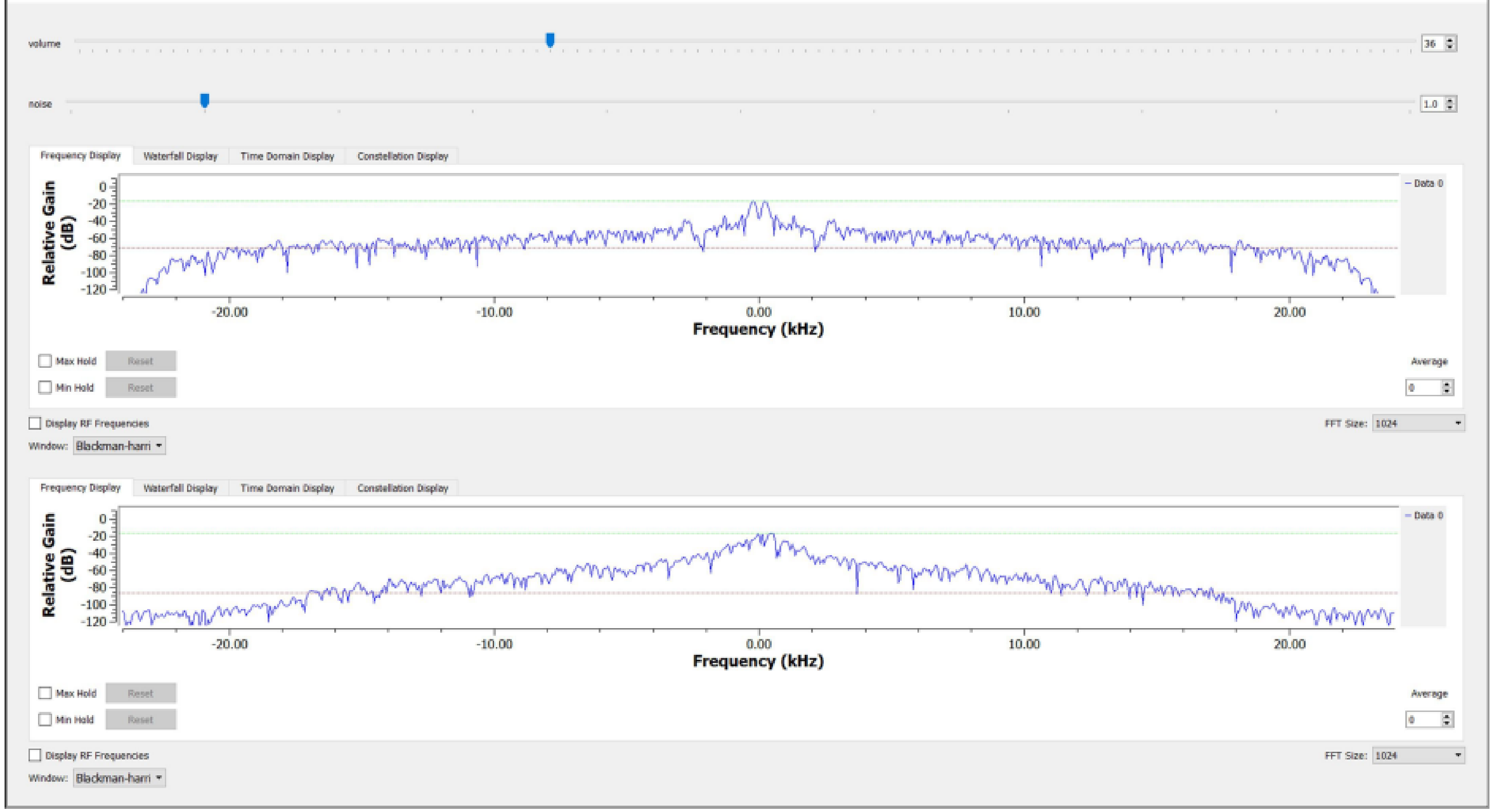
Window: Blackman-harri

FFT Size: 1024

Window: Blackman-harri

FFT Size: 1024

FINAL OUTPUTS IN FREQUENCY DOMAIN



volume 36

noise 1.0

Frequency Display | Waterfall Display | Time Domain Display | Constellation Display

Max Hold Reset

Min Hold Reset

Average 0

Display RF Frequencies

FFT Size: 1024

Window: Blackman-harris

Frequency Display | Waterfall Display | Time Domain Display | Constellation Display

Max Hold Reset

Min Hold Reset

Average 0

Display RF Frequencies

FFT Size: 1024

Window: Blackman-harris