

ものづくりやろう!

## 第五回 GNURadio による受信機の構成

JH3RGD 葭谷安正

---

### ■はじめに

前回、前々回と GNURadio の解説を行ってきました。今回は GNURadio でバンドパスフィルタ(BPF: Band Pass Filter)を実現しました。今回は GNURadio と広帯域受信機ドングルを使用して「ラジオの製作」、ではなく「受信機ソフトの構成」に挑戦します。前回の BPF の知識が今回の受信機ソフトの構成でも一部登場します。

すでに広帯域受信機用の受信ソフトとして HSDR や CubicSDR などこの「FB News」上でも過去に多くのソフトが紹介されています。ですからわざわざ GNURadio を使ってソフトを作る必要はないかもしれません。しかし、アマチュア無線家の中にもメーカー製の無線機や受信機を持っているけれども、新たに電子部品を買ったり手持ちの部品を使って、無線機やラジオを組み立てるのが楽しいという人が少なからず居られますね。私も掠る(かする)程度ですがその部類に入のですが、近頃は老眼が急速に進んでいます。このため、近距離から半田付けしたい箇所が見える距離まで離すと、半田ごてを握って両手を思いっきり伸ばしても半田付け箇所が届かないという情けない状態になっています。仕方なく半田ごてを握る回数を減らし、その分ソフトウェアで遊ばざるを得ないという状況です。

今回の受信機ソフトの構成では、FM ラジオ、AM ラジオ、CW 受信機、SSB 受信機のフローグラフ、動作画面の図の羅列になってしまいそうです。これらのフローグラフは数年前から GNURadio のホームページや他の書籍を参考にしてそのまま GNURadio に打ち込んだり、一部を変更して作成してきたものです。これらのフローグラフを実行すると、音量調整ノブやチューニングノブのレイアウトがラジオ毎に異なっていたり、チューニングにノブを使用していたりスライダーを使用していたり等、統一性が無いと感じられるかもしれません。これは作成時期が年単位で異なっていることが大きな原因です。今回この原稿を作成するためパソコンからかつて作成したラジオのソフトを引っ張りだしてきました。ブロックの組み方はスマートではないかもしれませんが、一応動くことを確認してあります。もう少しパラメータを調整した方が良いものや別のブロックに置き換えてコンピュータへの負荷を減らした方が良い部分もありますがご容赦のほどを。興味のある方はフロー図に手を入れていただいて、より良いものにしていただければと思います。

※この記事で説明しているフローグラフはこの HP のここからダウンロードできます(圧縮ファイル)。実際に動かしていただき、皆さんの用途にあわせて変更して使用していただければ幸いです。なおダウンロードにあたっては FB News のご厚意でこのサイトからダウンロードしていただけますが、ダウンロードしていただいたソフトに関しての質問やご意見については、決して FB News に質問等されないようにお願いします。

## ■使用物品

ソフトの構成には

- (1) 広帯域受信機
- (2) GNURadio がインストールされたパソコンを使用します。

(1)の「広帯域受信機( dongle )」ですが、今回使用したものは RTL-SDR です。筆者が以前購入した RTL-SDR は HF のダイレクトサンプリングモード対応しているものです。RTL-SDR Dongle は、本来デジタル TV 信号を PC でデコードするために設計されたものですが、これを受信機として利用するわけです。ネット上にもたくさんの情報があふれていますので細部の動作説明は省略します。

dongle の表に

「RTL-SDR.COM QUICKSTART SETUP V3 RTL2832U R820T2 …」

との記載があります。HF 帯の受信を考えておられる方は必ず「ダイレクトサンプリングモード」に対応している製品を使ってください。古い dongle の場合、ダイレクトサンプリングモードに対応しておらず、HF 帯の受信ができませんので気を付けてください。

(2)の「GNURadio がインストールされたパソコン」ですが、Windows 10 に GNURadio がインストールされているものを想定しています。また、RTL-SDR を使用するためには RTL-SDR のドライバがインストールされていなければ GNURadio が認識してくれません。ドライバは `zadig` を使用してインストールします。また、記載していませんが、受信機ですから dongle にアンテナを接続する必要があります。アンテナは対応する周波数帯のアンテナを接続してください。私は FM アンテナの代わりに 1 メートル程度の銅線で受信を試みましたが、筆者の部屋では全く受信できませんでした。プログラムが悪いのではと疑いましたが、あと 2 メートルほど銅線を追加すると FM 放送が入ってきました。それでは各種受信機のフローグラフを見ていきます。

## ■FM 受信機

### □FM 受信機フローグラフ

広帯域受信機ドングルを用いた GNURadio のサンプルとして FM 受信機がよく例示されています。この図 1 もそれらサンプルの一例です。フロー図は図 1 のように接続します。

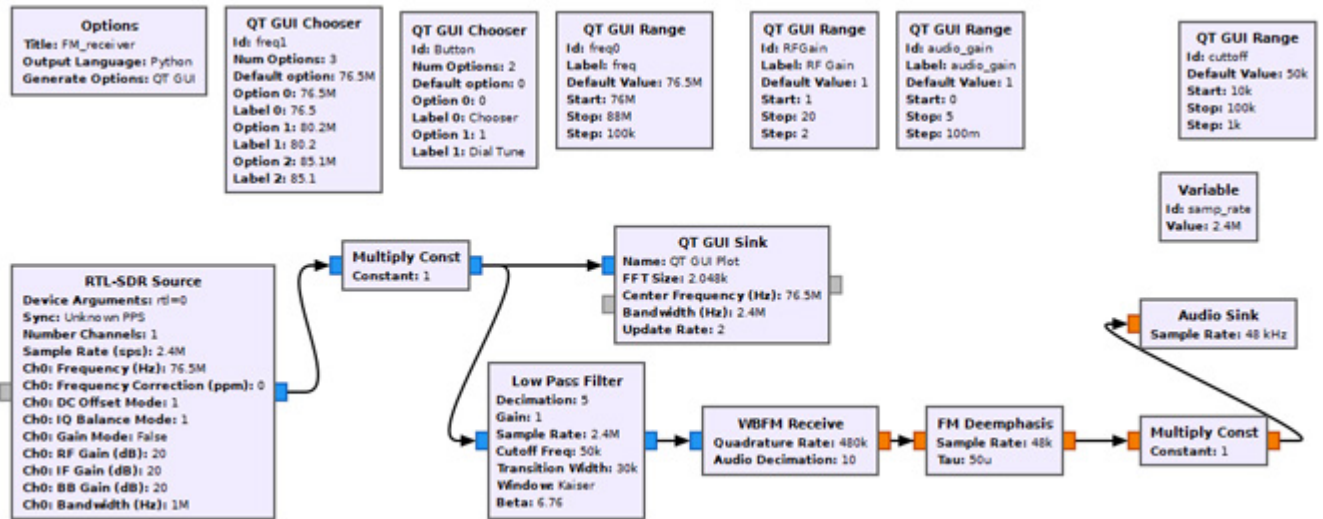


図 1 FM 受信機フローグラフ

RTL-SDR のドングルをパソコンの USB 端子に挿入します。その後 GNURadio を起動すると、GNURadio は RTL-SDR を自動認識しますからソフト側での設定は不要です。プログラム動作時の画面を図 2 に示します。

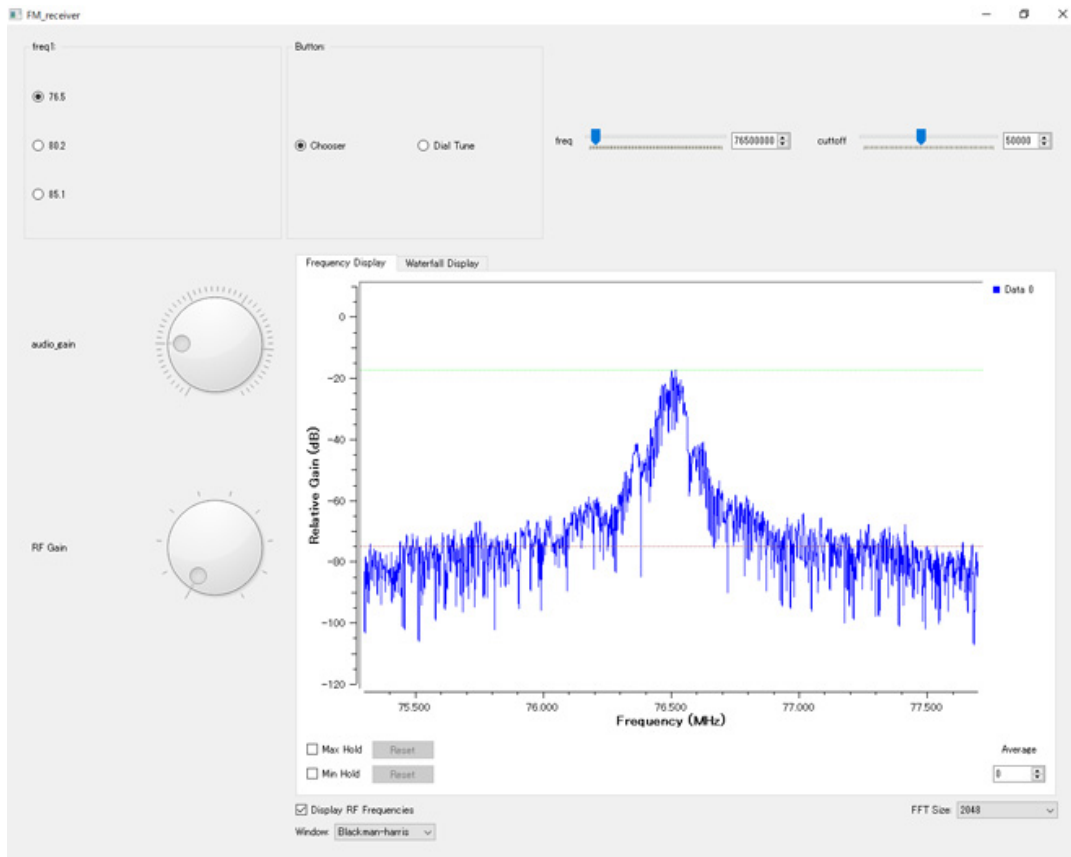


図 2 FM 受信機動作画面

## □動作概要

動作概要を図 1 の各ブロックに沿って簡単に説明します。

- 「RTL-SDR Source」は RTL-SDR ドングルの受信周波数やサンプリングレート他を設定します。受信周波数と記載しましたがイメージとしてはこの周波数を中心として一定帯域の信号が端子から出力されるという意味です。サンプリング周波数はこのドングルでは 2.4MHz で使用することが多いですが、サンプリング周波数 2.4MHz の時、「Frequency(Hz)」端子の周波数を中心に、(+)方向に 1.2MHz、(-)方向に-1.2MHz の帯域の信号が抽出されます。ですから、この端子から出る信号は「Frequency(Hz)」端子に設定した信号が 0Hz に変換されて出力されると考えることができます。受信したい周波数は「RTL-SDR Source」ブロック内の「CHO:Frequency(Hz):」に入れます。この例ではこの欄に数式「(1-Button)\*freq1+Button\*freq0」が入っていますが、「Chooser」の選択によって Button に 0 か 1 が入るので結果として freq1 の値か freq0 の値が選択されることとなります。この例では、信号の周波数は「QT GUI Chooser」-「Id:freq1」または「QT GUI Range」-「Id:freq0」の値を設定します。「QT GUI Chooser」-「Id:freq1」または「QT GUI Range」-「Id:freq0」のいずれの周波数を選択するのかは、図 2 実行画面の最上段にあるラジオボタン「QT GUI Chooser」の「Chooser」、「Dial Tune」のいずれの Button にチェックが入るので選択がきまります。「Chooser」にチェックが入ると「Id:freq1」の固定の周波数から選択でき、「Dial Tune」にチェックが入ると「Id:freq」の Slide バーで周波数が選択できるようになります。

- 「Multiply Const」は定数を掛け算するブロックです。アナログ回路の増幅に相当する機能をデジタル信号処理で実現する手段として使っています。何倍に増幅するのかは、このボックスをダブルクリックして「Constant」欄に設定します。「Constant」欄には RFGain と書かれていますが、これは「QT GUI Range」というブロックの Id と同じ名前ですので、このブロック内の値が使用されます。「QT GUI Range」の「Default Value」に実数(「Type:float」)の「1」がはいっていますので、この変数の初期値は 1 になります。

- 「Multiply Const」から「QT GUI Sink」と「Low Pass Filter」に結線がありますので、「1」倍された信号が両方にながれていきます。

- 「QT GUI Sink」は「RTL-SDR」の出力信号を周波数表示するためのスコープです。

- 「Low Pass Filter」では指定された周波数(「Cutoff Freq」)以下の周波数の信号を通過します。

図 1 の値では、「CHO:Frequency(Hz):」には初期値として 76.5MHz、「Low Pass Filter」の「Cutoff Freq」には「50kHz」がはいっていますので、「Low Pass Filter」出力端には 76.550MHz 以下の信号が出力されます。

- 「WBFM Receiver」は FM 変調信号を復調するためのブロックです。出力は復調されたオーディオ信号です。

- 「FM Deemphasis」は FM 復調で高域信号補正回路です。

## □ 「Sample rate」、「decimation」について

図 1 のブロックの中で、ブロックを経るにしたがって「Sample rate」や「decimation」が変わっていくところがあります。この「Sample rate」と「decimation」の関係はどのようになっているのでしょうか。「Sample rate」は、1 秒間に何回のデータを取り込むのかを示す数値です。「2.4M」と記載がありますので 1 秒間に 2,400,000 回のデータを取り込んでいます。「decimation」は、間引きを意味します。図 1 の設定では、1 秒間にデータを 2,400,000 回取り込みますが、現実に必要なデータ数はこのうちの一部分です。必要なエリアから必要な情報を取り込むことができれば残りのデータは不要ですので間引いておきます。この間引き率が数字で記載されます。図 1 のブロック中の「decimation」欄の値は 5 ですから 2,400,000 回のサンプリングデータのうち 5 分の 1 の 480,000 回分のデータを残して後は捨ててしまうといったイメージです。

## □ GUI レイアウト方法について

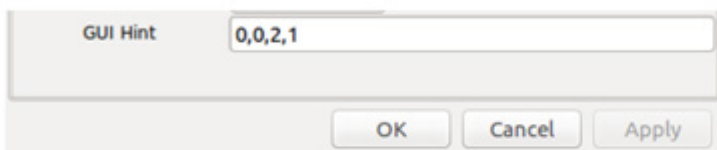
GNURadio を使用してフローグラフを作成するとき、様々な部品をレイアウト指定を行わないと部品がどこに配置されるのかわかりません。もし「図 2 FM 受信機動作画面」のように部品配置を行う場合は各ブロック内の GUI Hint 欄に次の要領で配置情報を入力します。

部品の配置には GUI 部品の「GUI Hint」欄に 4 つの数値をコンマで区切って指定します。例えば、「QT GUI Sink」の「GUI Hint」欄には「1,1,3,3」と記載されています。数値の意味は、左から、「行、列、行スパン、列スパン」を表しています。

GNURadio のホームページには「GUI Hint」の記載要領が次のように記載されています。部品をレイアウトする場合、行、列、スパンについて次のような図 3 が記載されています。

### GUI Hint

All of the QT GUI widgets and plots have a parameter called GUI Hint. This is used to arrange GUIs in the window, as well as assign them to tabs in a QT GUI Tab Widget.



The format is:

(row, column, row span, column span)

For example,

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)

Waveform Selector (0,0,2,1)	Offset Slider (0,1,1,1)
	Frequency Slider (1,1,1,1)
Time Display (2,0,1,1)	Frequency Display (2,1,1,1)

図 3 GUI Hint の記載例（「[https://wiki.gnuradio.org/index.php/GUI\\_Hint](https://wiki.gnuradio.org/index.php/GUI_Hint)」より）

行、列は 0 から始まる番号であり、スパンは各ツールが配置されるエリアのサイズを指定しています。数値の指定方法に従って GUI Hint を設定してみましたが、なかなか思ったイメージ通りにはなりません。



## FM 受信機の構成(2)

FM 受信機をもっとシンプルに構成する例が [https://wiki.gnuradio.org/index.php/FM\\_Demod](https://wiki.gnuradio.org/index.php/FM_Demod) に掲載されています。図 2 で使用しているブロックと若干異なりますが非常にシンプルです。この解説で使用したドングルと上の URL に記載されているフローグラフでは使用しているドングル、動作サンプルレートが異なりますのでそのままでは動きませんでした。動作するように変更したフローが図 4 です。

「RT-SDR Source」ブロックから出力された信号をすぐに 10 分の 1 に間引いています。その後「FM Demod」ブロックで FM 信号を復調しています。復調後さらに Decimation で 5 分の 1 に間引いて Sample rate を「Audio Sink」の Sample rate に合わせて渡します。「Audio Sink」の先につながっているハードウェアはサウンドボードですので、D/A 変換されて音声出力となっています。RT-SDR が面倒な信号処理を行うとともにブロックでも高度な信号処理機能を持っているとは言え、ブロックを何個か並べるだけで放送波が受信、復調されて音声が出てきてしまうのですから驚きです。

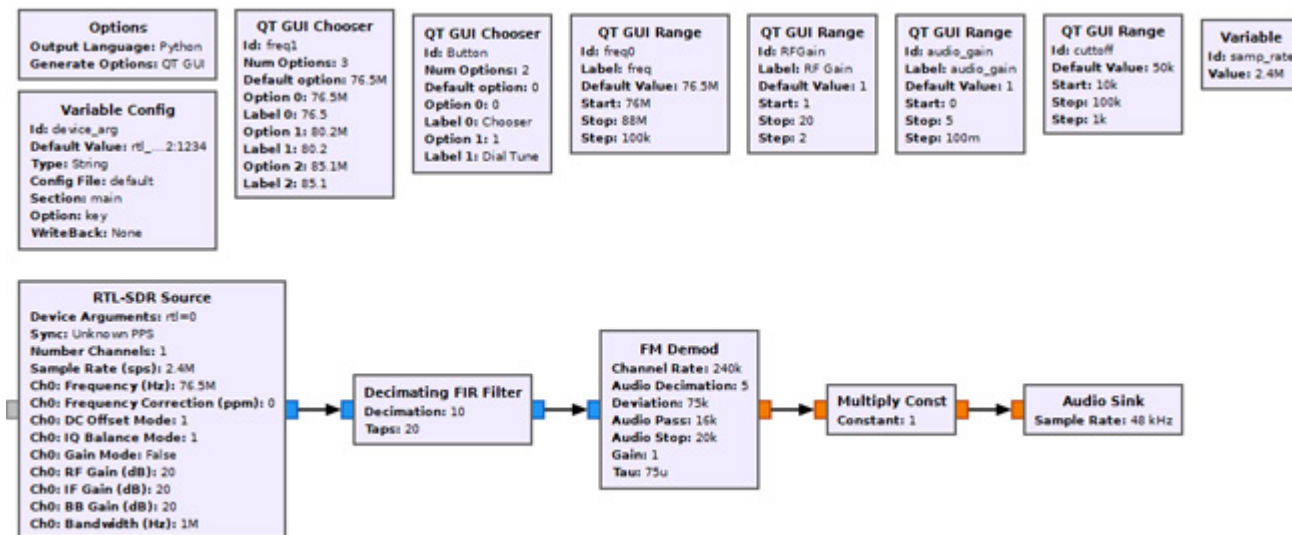


図 4 FM 受信機フロー図(2)

## ■7MHz CW 受信機

### ロダイレクトサンプリングモードについて

CW 受信機を含め、以後で記載する回路はすべて HF 帯を例にあげています。RTL-SDR ドングルを HF 帯で使用する場合、ダイレクトサンプリングモードに設定する必要があります。機種によってダイレクトサンプリングモードにする周波数帯が異なるかもしれませんが、私の持っているドングルの一つには、「100kHz~30MHz」がダイレクトサンプリングモードと記載があります。

設定の具体的方法は、「RTL-SDR Source」ブロックの「Device Arguments」欄に必ず「direct\_samp=2」と入れます。複数のドングルを接続する場合はすでにこの欄に「rtl=0」や「rtl=1」と入れてどのドングルかを指定する必要がありますので、その場合は「rtl=0, direct\_samp = 2」という具合に区切って記載してください。「direct\_samp」を記載しないと HF 帯を受信できませんのでくれぐれもお忘れなく。

## 7MHz CW レシーバフロー図と周波数シフト

図5にCW受信機のフローグラフを記載しましたが、この中の「Frequency Xlating FIR Filter」ブロックについて説明します。

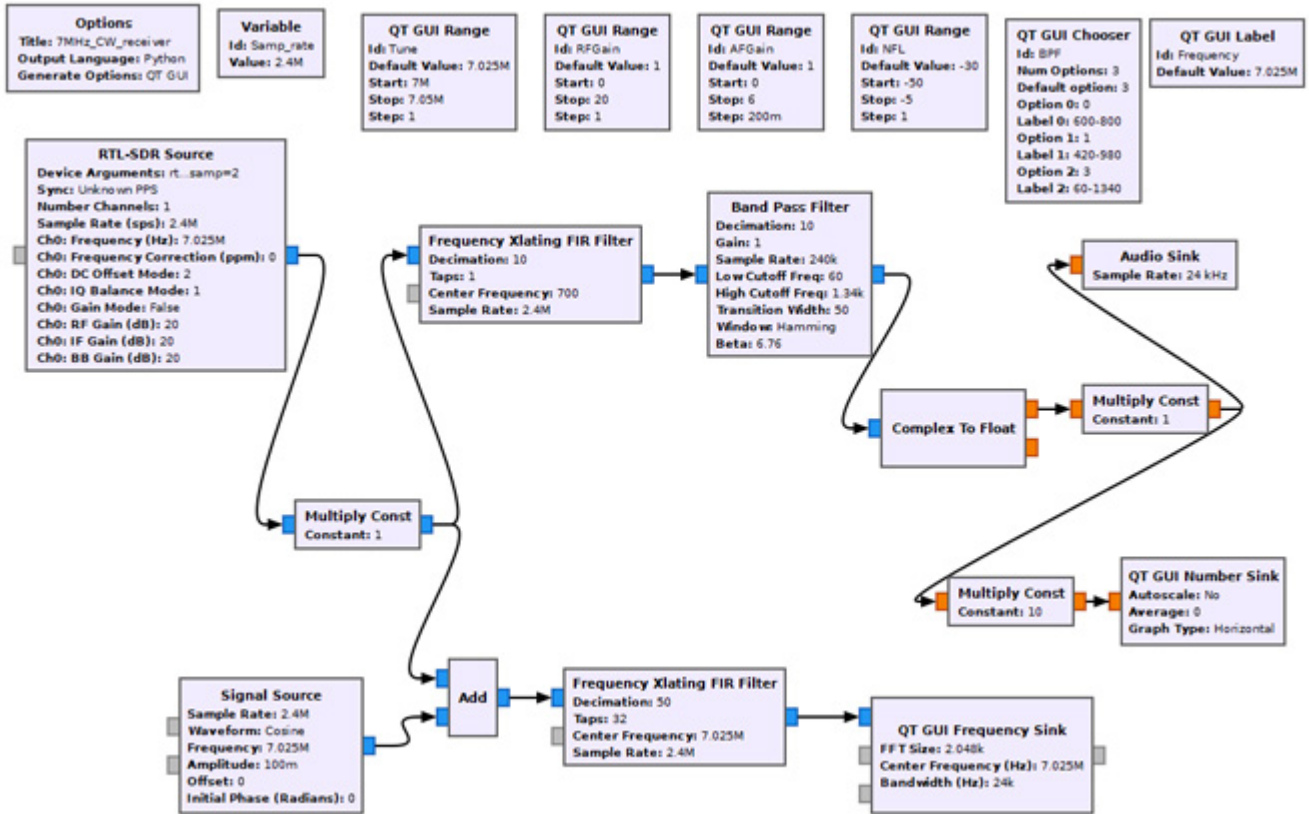


図5 7MHz CW レシーバフロー図

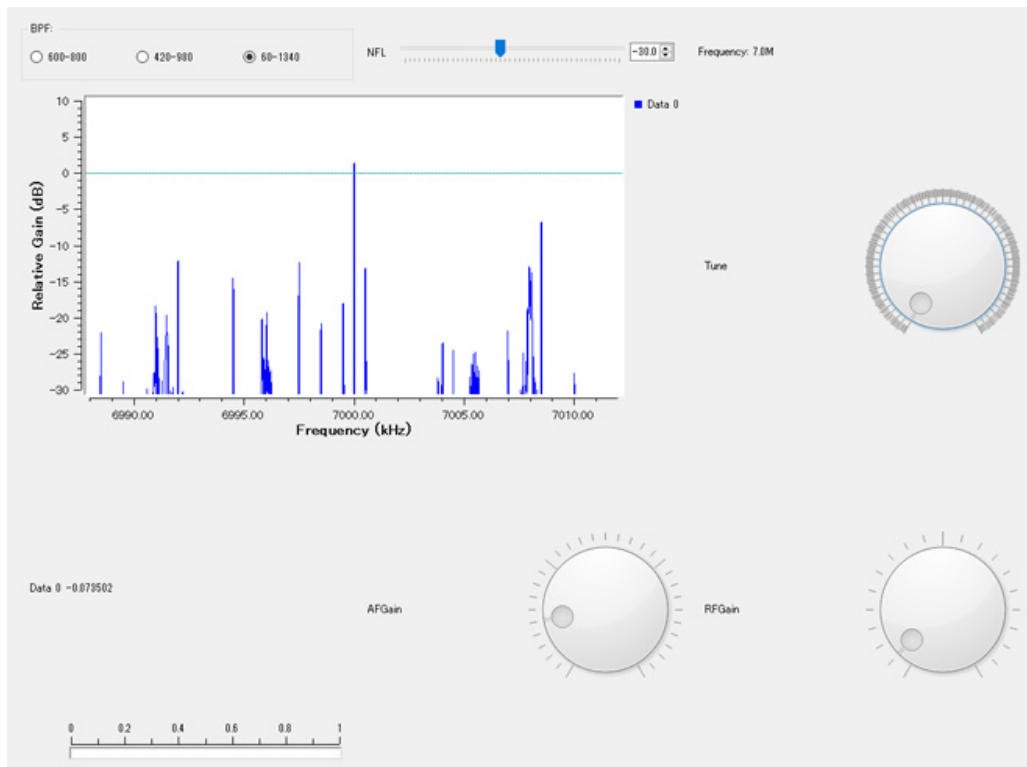


図6 7MHz CW レシーバ実行画面

「Frequency Xlating FIR Filter」ブロックはつぎのような処理をおこないます。

- (1) 入力信号の周波数を変換します(ずらします)。どれだけずらすのかというと「Center Frequency」欄に記載された周波数分オフセットをかけます。
- (2) 周波数をずらせた信号に Decimation を実行し、出力します。

周波数変換ですが、図 5 の「Frequency Xlating FIR Filter」ブロックの前後主要ブロックを抜粋して説明します(図 7)。「RTL-SDR Source」ブロックの出力は図 7 では 7.025MHz を中心にサンプルレートの半分の帯域の信号成分が含まれています。FFT 分析すると 7.025MHz を中心周波数として分布しているわけです。 dongle の内部で周波数変換が施され、dongle 出力からは 7.025MHz 成分が 0MHz に変換されて出ているというイメージです。dongle の出力信号は 0Hz 成分が 7.025MHz なんだと認識している必要があります。「Frequency Xlating FIR Filter」は受け取った信号を 700Hz シフトして出力しますので、結果的に「7.025MHz の周波数成分が 700Hz にシフトされる(ほかの周波数成分も 700Hz シフトされる)」、また同じことですが、「(7.025MHz-700Hz)の信号成分が中心周波数 0Hz に変換されるように周波数シフトされる」、「「Frequency Xlating FIR Filter」出力の 700Hz の信号はもともと 7.025MHz の信号成分」ということになります。ですから、繰り返しになりますが、だれかが 7.025MHz でモジュール通信をおこなっているとき、dongle をこの周波数に合わせると、GNU Radio のフローグラフのパラメータが図 7 のように設定されていれば、「Frequency Xlating FIR Filter」出力から 700Hz の信号が出力され、それはもともと 7.025MHz の信号なわけです。

つぎに「Decimation」欄がありますが、すでに説明したようにこの結果は帯域を減少させます。Decimation が 10 と設定されると、データを 10 分の 1 に間引くわけですが周波数軸上で 0Hz を中心に分布する信号成分の 0Hz から全データの 10 分の 1 の信号成分が残され(出力され)、それ以外が間引かれます(捨てられます)。帯域が狭くなりますが、処理しなければならぬデータ数も 10 分の 1 になるため、不要な信号を除去することができ、コンピュータの処理も軽くなります。

「Taps」については、FIR フィルタというデジタルフィルタのタップ係数を設定する欄です。ここには、Sample\_rate, decimation transition などの名前で定義された数値を使って、関数 `firdes.low_pass(1,Samp_rate,Samp_rate/(2*decimation), transition)` と入力するとタップ係数を計算するようです(入出力値が float か complex かで関数が少し異なります)。何種類か数値を入れて試しましたが、当方が設定を間違っているのか原因かわかりませんが、ソフトがフリーズしたり音飛びが発生して思ったとおりに動きませんでした。このため他の文献に記載のある 10 や 20 という数値を入れてみました。一応動いていますが、気持ちの悪い方は調べてお使いください。



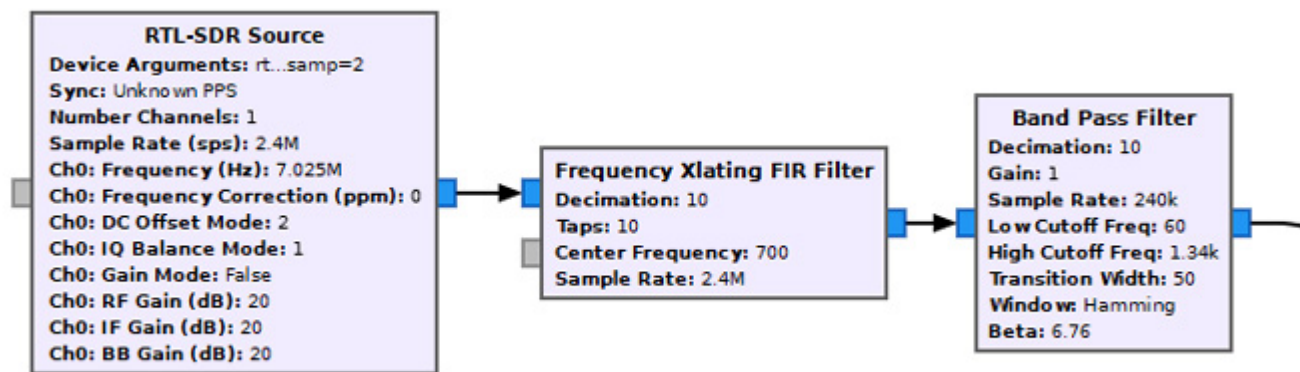


図7 「Frequency Xlating FIR Filter」動作説明用フローグラフ

Decimation のイメージは GNURadio のホームページにこの概念のイメージ図があります。図8が、周波数成分を見た場合の「frequency translation(周波数変換)」、「Decimation(間引き)」のイメージ図です。図8最上部の図が受信信号の周波数分析結果だと仮定します。図8の上から2番目の図は1番目の図の信号を左方向(周波数が低くなる方向)にずらしたものです。ずらしたというよりも回転させたというイメージであることがこの図でわかります。また、上から4番目の図がDecimationのイメージをあらわしています。注目する信号成分を中心部に移動させるために周波数シフトを行い、中心から外遠い部分を取り去ってしまうことで全体の処理しなければならない信号数を減少させているわけです。

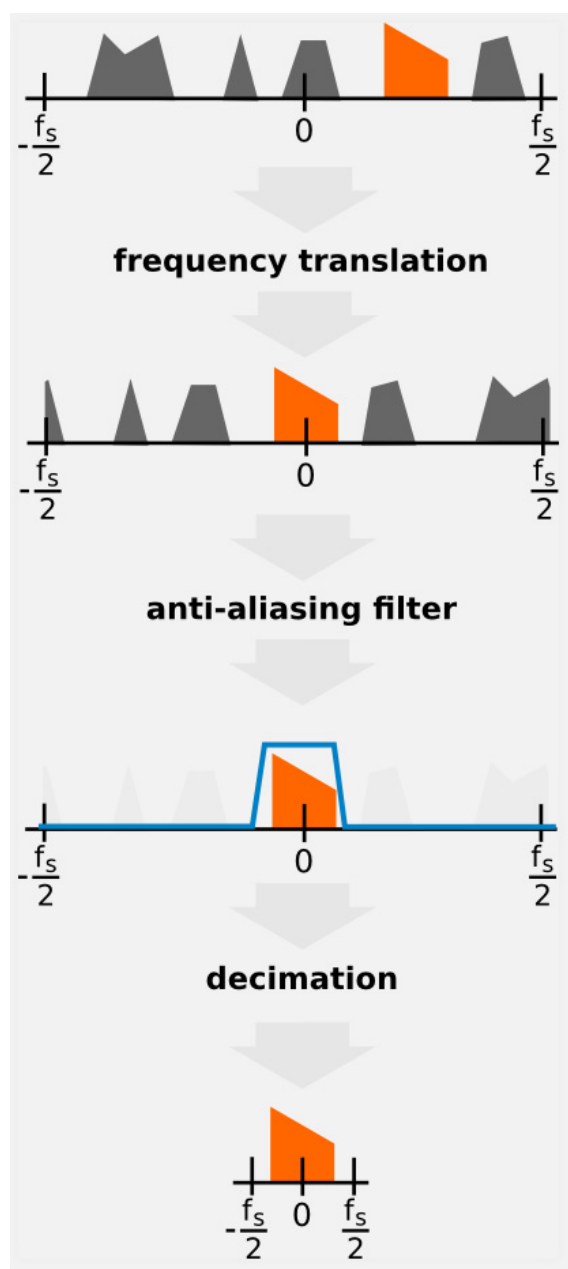


図8 周波数移動(Frequency translation)、間引き(decimation)のイメージ図

「Frequency Xlating FIR Filter」ブロック前後での周波数やサンプリングレート、データ数の関係についてはつぎのようになります。

- (1) 入力信号の中心周波数が「Center Frequency」欄に入力された数値の周波数に変換される。
- (2) Decimation が設定されている場合はデータの数が増加する。

これを Sample Rate で見ると

(ブロック出力の Sample Rate)

$$= (\text{ブロックへの入力 Sample Rate 数}) \div (\text{Decimation 欄の設定値})$$

フローグラフを作成するときにブロックを通過するに従い Sample rate が変わっていくことに注意してください。

なお、「RTL-SDR Source」の Sample rate は 2.4MHz ですが、最終ブロックの「Audio Sink」の Sample rate は 24kHz(製品によりますが、16kHz や 32kHz、48kHz も選べます)です。いくつかの処理ブロックを経由しますが、最終的に 2.4MHz を 24kHz まで落とさないと音が出せませんので途中 Decimation して Sample rate を低減していかなければなりません。この数値が一致しないとデータあふれや不足が起きて、スピーカーからポコポコという音が出たり、GNURadio のターミナルエリアに○がたくさん並んだりします。そのような症状がでたときには Sample rate の不一致を疑ってください。

## ■AM 受信機

### □AM 受信機構成例(1)

FM 受信機の構成とほぼ同じです。フロー図と動作画面をしめします。

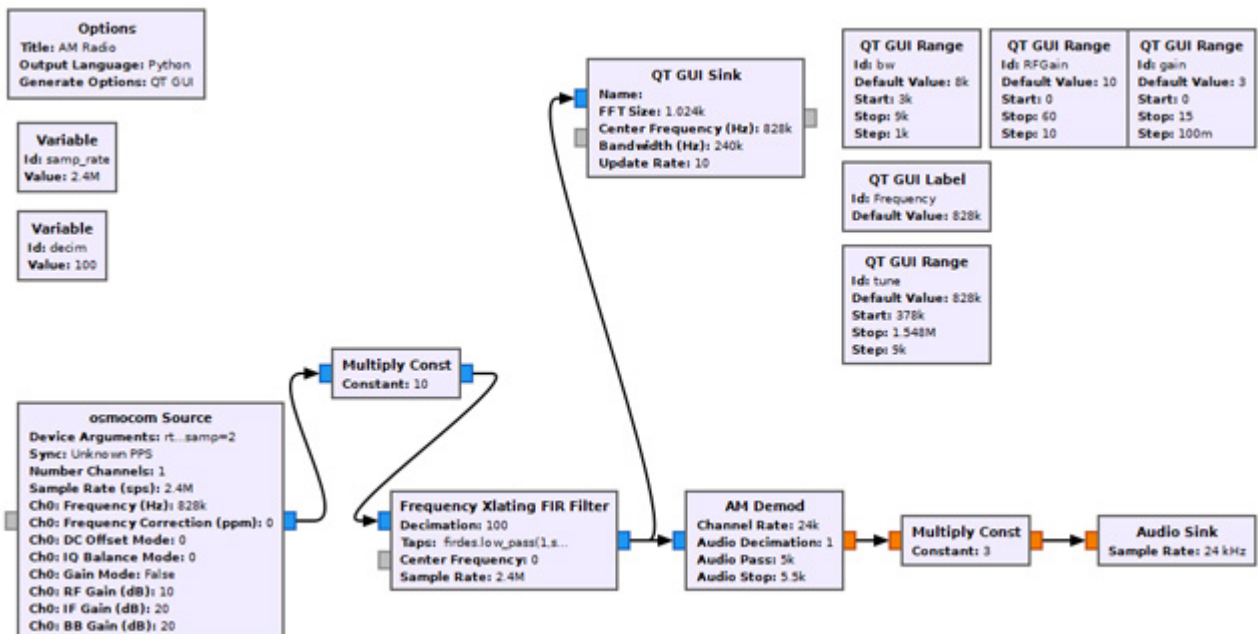


図 9 AM 受信機フロー図

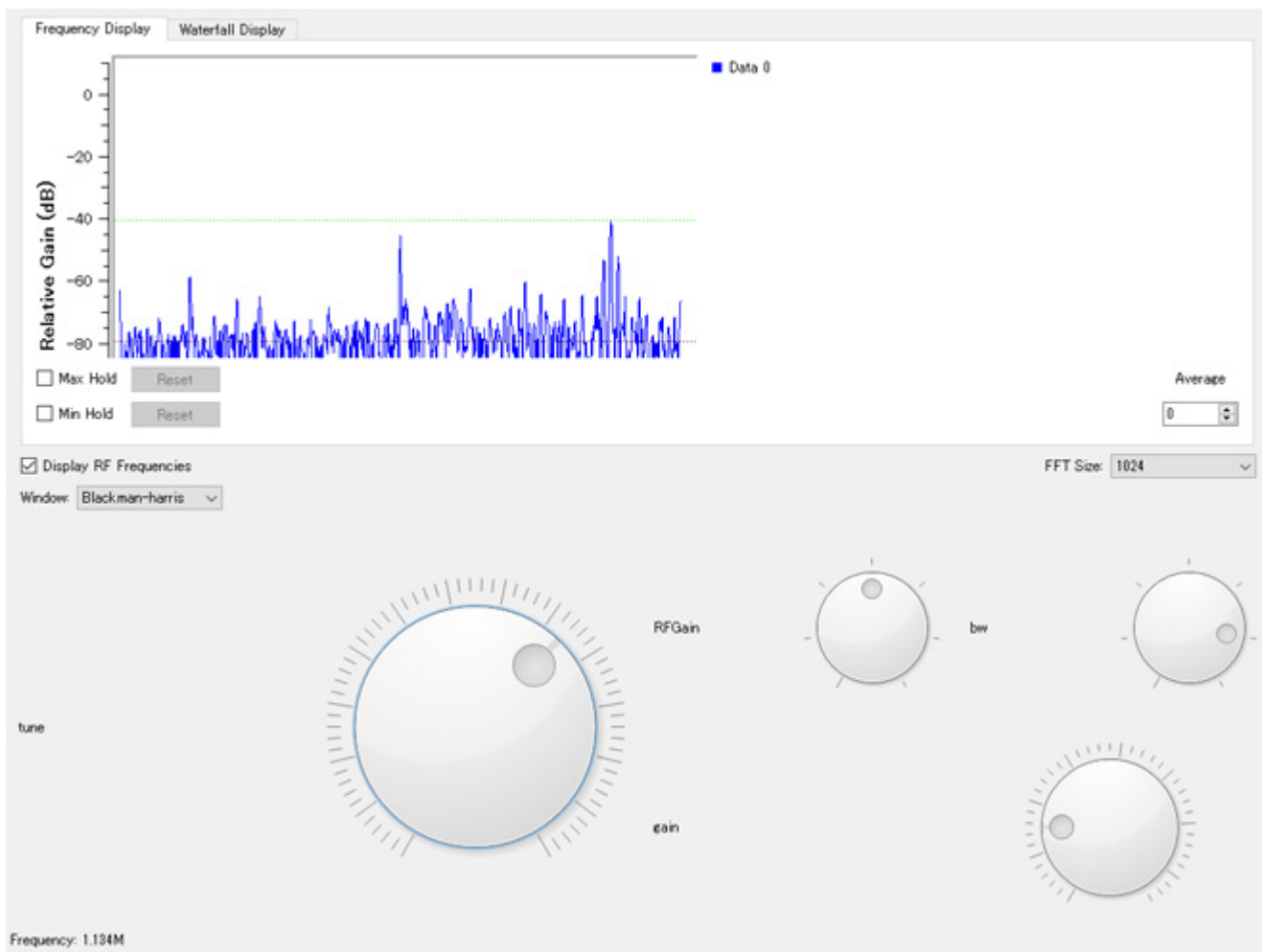


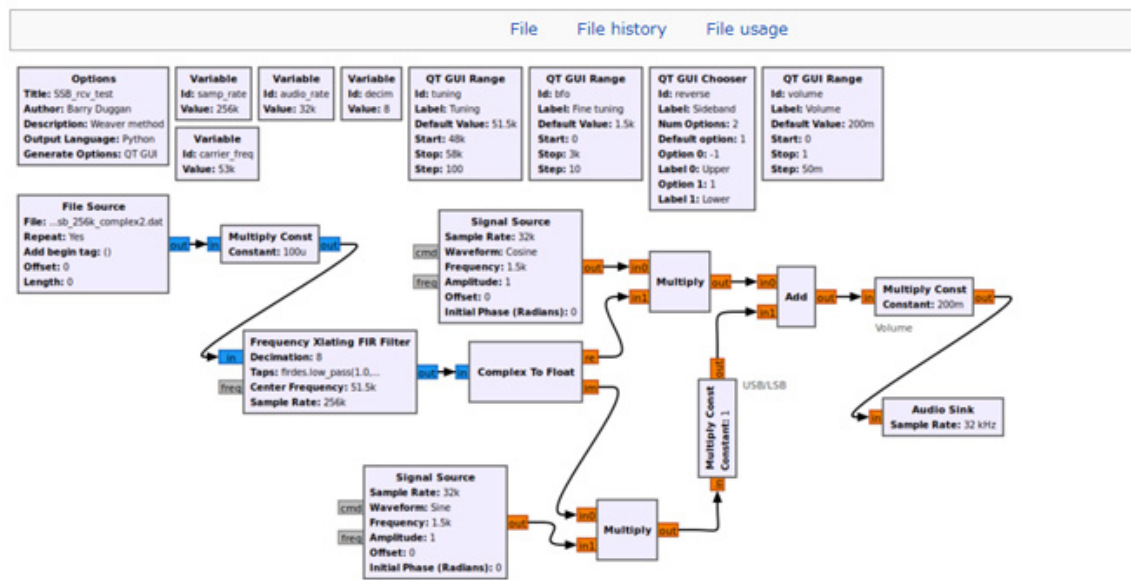
図 10 AM 受信機動作画面

なお、参考文献(1)には図9の構成とは別の構成方法が例示されています。詳しい説明は省略しますが、その理由は DC オフセットの問題があり、復調処理ができなくなってしまうとのことです。図9のフローグラフはとりあえず動いていますのでアマチュア無線家としては OK かもしれませんが、問題があるとのことですので注意してください。

## ■7MHz SSB レシーバ

GNURadio の HP にはたくさんの情報が記載されていますが、その中に SSB 受信機の構成例を見ることができます。それらの構成例を参考にして作成したのが下図のフローグラフです。動作事例とともに掲載しておきます。

# File:SSB test fg.png



Size of this preview: 800 × 420 pixels. Other resolutions: 320 × 168 pixels | 1,076 × 565 pixels.

Original file (1,076 × 565 pixels, file size: 110 KB, MIME type: image/png)

Single Sideband test flowgraph

図 11 SSB 受信機フローグラフ

([https://wiki.gnuradio.org/index.php/Simulation\\_example:\\_Single\\_Sideband\\_transceiver](https://wiki.gnuradio.org/index.php/Simulation_example:_Single_Sideband_transceiver) より)

図 11 のフローグラフは他の回路と比較してブロックの数が多く、面倒な処理をしているようにみえますが、SDR 技術を素直にフローにした構成方法です。もっと簡単な方法として、高橋知宏氏が記載されている構成例(1)はシンプルでわかりやすいものでしたので、高橋氏の構成例で作成してみました(図 12、図 13)。

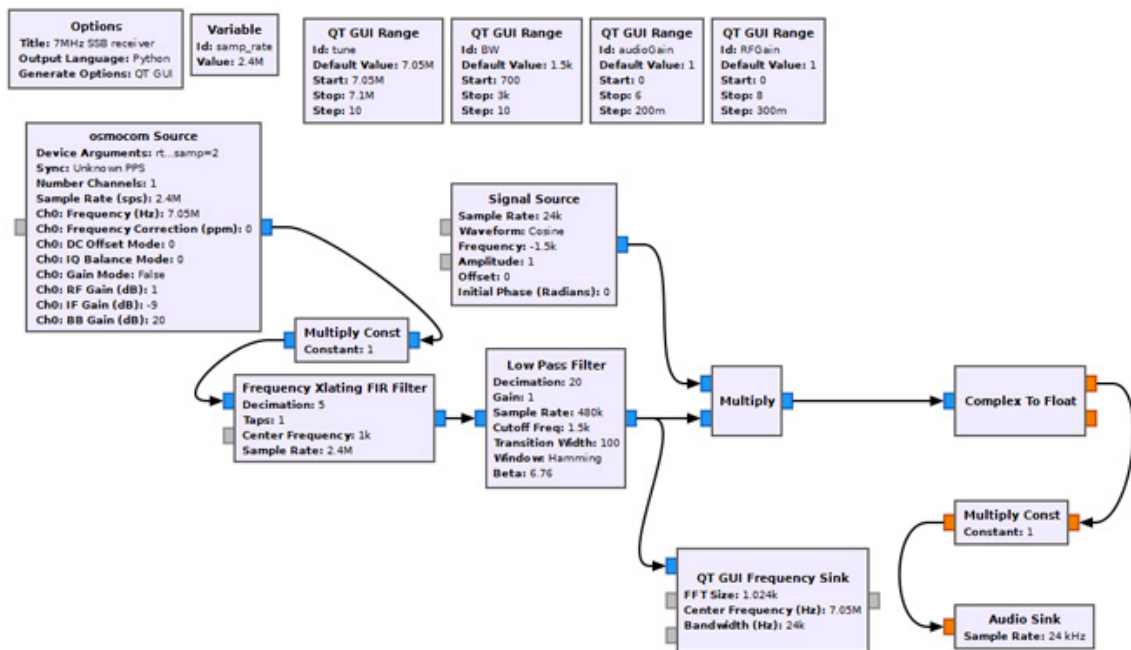


図 12 7MHz SSB レシーバフロー図

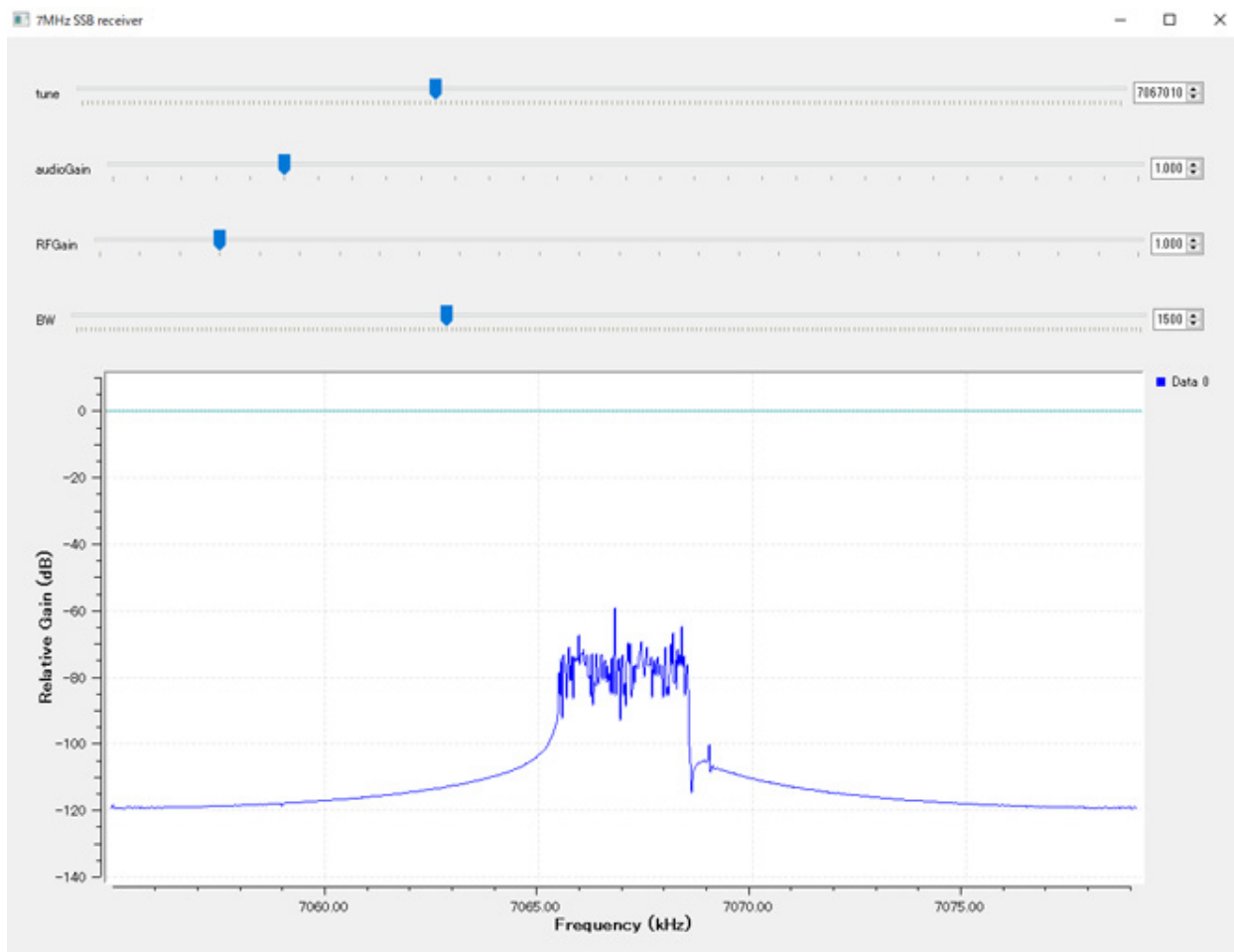


図 13 7MHz SSB レシーバ動作画面

図 13 の動作画面はローパスフィルタの出力段の信号を「QT GUI Frequency Sink」で観測した図です。この出力の周波数は周波数変換されて本来の受信周波数とはことなるのですが、元の周波数との対応をつけて観測しています。7MHz までは LSB です。14MHz 以上の周波数信号の復調方法の変更のためにパラメータを変更する必要があるとおもいます。

GNURadio での受信機構成例いかがでしょうか。この記事では GNURadio の機能のごく一部しか説明できませんでしたが、まだまだ多くの機能を有しているようです。半田ごてで実際の回路を作成するのも楽しいですが、ソフトウェアをさわるのも別の楽しみがあります(これはソフトじゃないでしょうという声も聞こえてきそうですが…)

ハード、ソフトを問わず、「ものづくりやろう!」

## ■参考文献

今回も参考文献として下記出版物,HP を参照させていただきました。

- (1) 高橋 知宏:「GRC で広がる SDR の世界」,RF ワールド No44,pp7~pp91, CQ 出版社, 2018.
- (2) rapidnack, GNURadio 電子工作 Vol.1 AM 送信・受信編 Kindle 版
- (3) rapidnack, GNURadio 電子工作 Vol.2 Embedded Python Block 編 Kindle 版