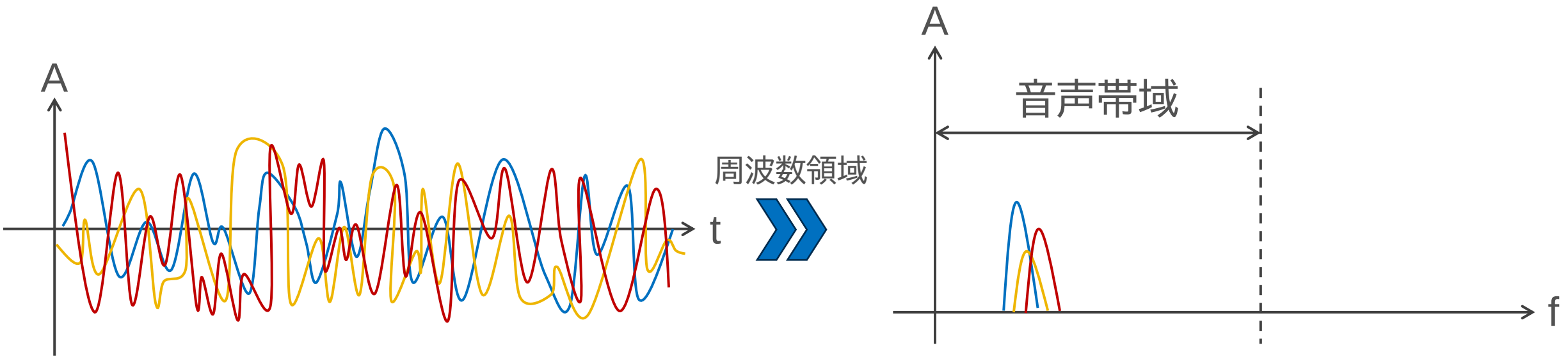
ソフトウェア無線機

電子磁気応用資料

内容

- 全4回程度を想定（演習の進み具合で変化）
  1. 変調について
  2. 無線機のハードウェア
  3. ソフトウェアによる無線機の構成
     - ✓ 直交変復調方式について
  4. ソフトウェア無線の実習(GNU Radio使用）
     1. AMの受信
     2. FMの受信

音声を伝送する

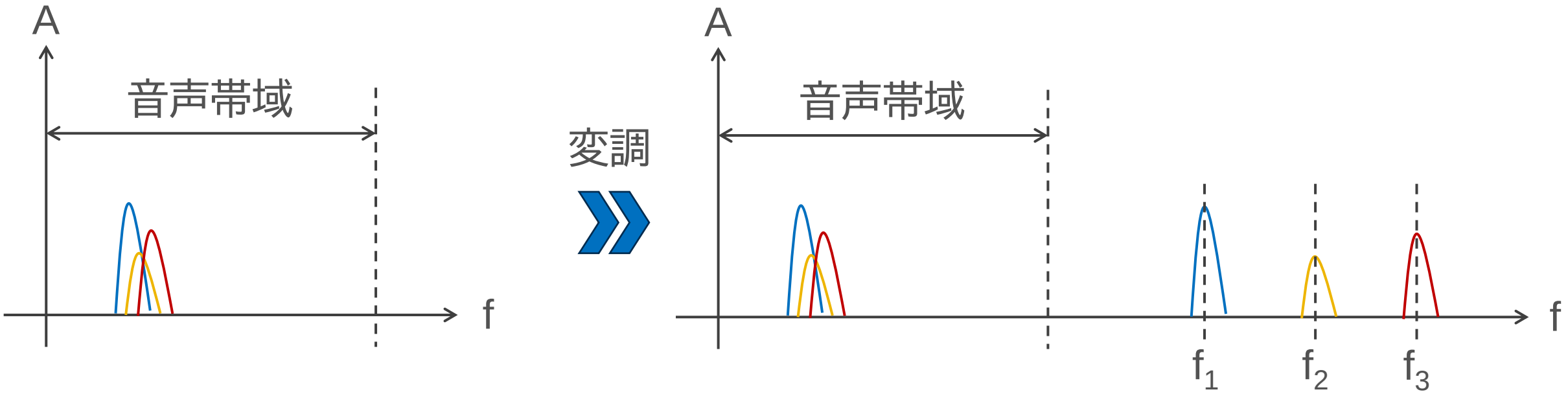


3つの音声を同時に混ぜて送っても、受け手側で分離することは難しい。

» 異なる周波数に情報を載せ直してから伝送する

→ **変調**という

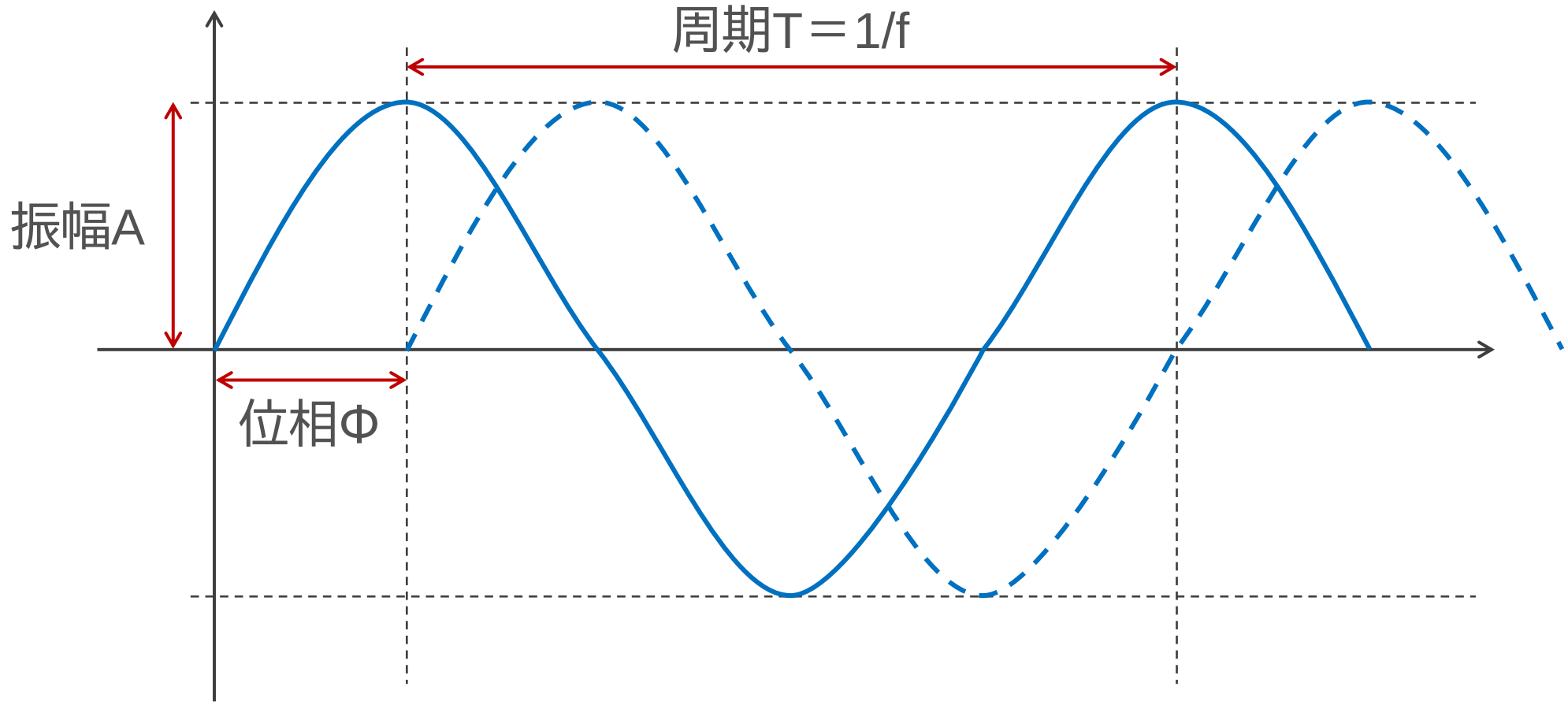
変調とは



そのままでは混ざってしまう信号を別の周波数帯域に分離して伝送、
受信側で元の信号に戻す方法

波を特徴づける3要素のいずれかに目的の信号を載せる事ができる。

波を特徴づける3要素



$$v(t) = A\sin(2\omega f t + \phi)$$

波の3要素を変調する

- 振幅を変調
  - AM: amplitude modulation
- 周波数を変調
  - FM: frequency modulation
- 位相を変調
  - PM: phase modulation

振幅変調

AMの式

変調信号

$$v_{AM} = (V_c + V_s \cos(2\pi f_s t))\sin(2\pi f_c t)$$

キャリア信号

キャリアの**振幅成分**に信号を乗せる

一般的には変調度$m = {}^{V_m}/_{V_c}$を用いて、

$$v_{AM} = V_c(1 + m\cos(2\pi f_s t))\sin(2\pi f_c t)$$

と書く。

振幅変調のシミュレーション

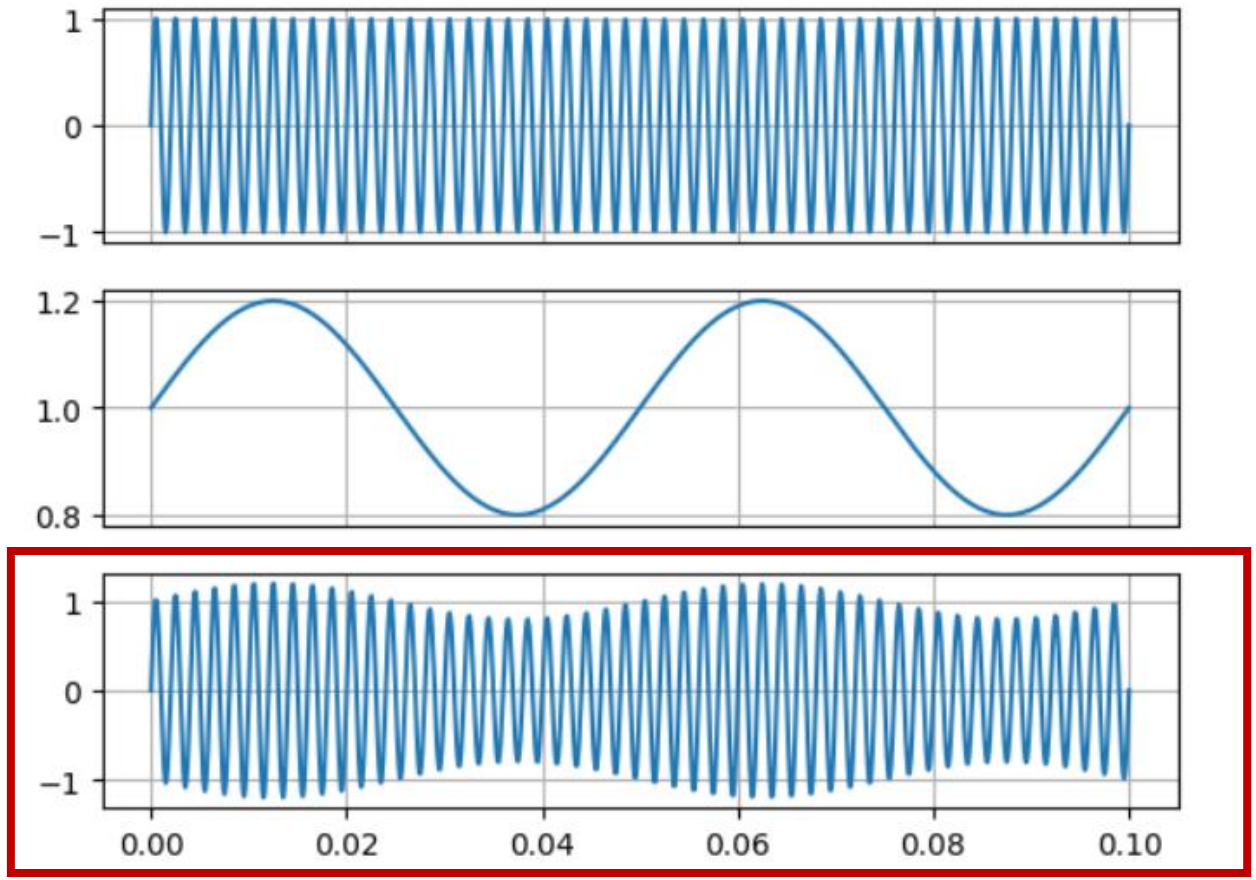
```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 0.1, 1000)
fc = 500
vc = np.sin(2*fc*np.pi*t)
m = 0.2
fs = 20
vs = 1 + m*np.sin(2*fs*np.pi*t)

v = vs*vc

fig = plt.figure()
ax = fig.add_subplot(311)
ax.grid()
ax.plot(t, vc)
ay = fig.add_subplot(312)
ay.grid()
ay.plot(t, vs)
az = fig.add_subplot(313)
az.grid()
az.plot(t, v)
axes = fig.get_axes()
for n in axes:
  n.tick_params(axis='x', which='both', bottom=False, labelbottom=False)
n.tick_params(axis='x', which='both', bottom=True, labelbottom=True)
```

0～0.1秒の時間軸を作る
500Hzのキャリア周波数
キャリアを生成
変調指数m
信号の周波数
信号の生成
キャリアと信号の積 (AM処理)

この辺はグラフ処理

Colabでのシミュレーション結果

```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 0.1, 1000)
fc = 500
vc = np.sin(2*fc*np.pi*t)
m = 0.2
fs = 20
vs = 1 + m*np.sin(2*fs*np.pi*t)

v = vs*vc

fig = plt.figure()
ax = fig.add_subplot(311)
ax.grid()
ax.plot(t, vc)
ay = fig.add_subplot(312)
ay.grid()
ay.plot(t, vs)
az = fig.add_subplot(313)
az.grid()
az.plot(t, v)
axes = fig.get_axes()
for n in axes:
  n.tick_params(axis='x', which='both', bottom=False, labelbottom=False)
n.tick_params(axis='x', which='both', bottom=True, labelbottom=True)
```



授業ではこのグラフだけ作成

この部分を`plt.plot(t, v)`に置き換える

位相変調

PMの式

信号

$$v_{PM} = V_c \sin(2\pi f_c t + \Delta\phi \sin(2\pi f_s t))$$

キャリアの**位相成分**に信号を乗せる

ΔΦはどの程度位相を偏移させるのかを表すパラメータで最大位相偏移という。
正弦波の値域は－1～＋1なので、－ΔΦ～＋ΔΦの範囲で位相が変化する。

位相変調のシミュレーション

```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 0.1, 1000)
fc = 200

delta_phi = 120/180*np.pi
fs = 20
vs = delta_phi * np.sin(2*fs*np.pi*t)

v = np.sin(2*fc*np.pi*t + vs)

fig = plt.figure()
ax = fig.add_subplot(211)
ax.grid()
ax.plot(t, vs)
ay = fig.add_subplot(212)
ay.grid()
ay.plot(t, v)

axes = fig.get_axes()
for n in axes:
  n.tick_params(axis='x', which='both', bottom=False, labelbottom=False)
n.tick_params(axis='x', which='both', bottom=True, labelbottom=True)
```

0～0.1秒の時間軸を作る

200Hzのキャリア周波数を設定

最大位相偏移ΔΦを指定

信号の周波数を設定

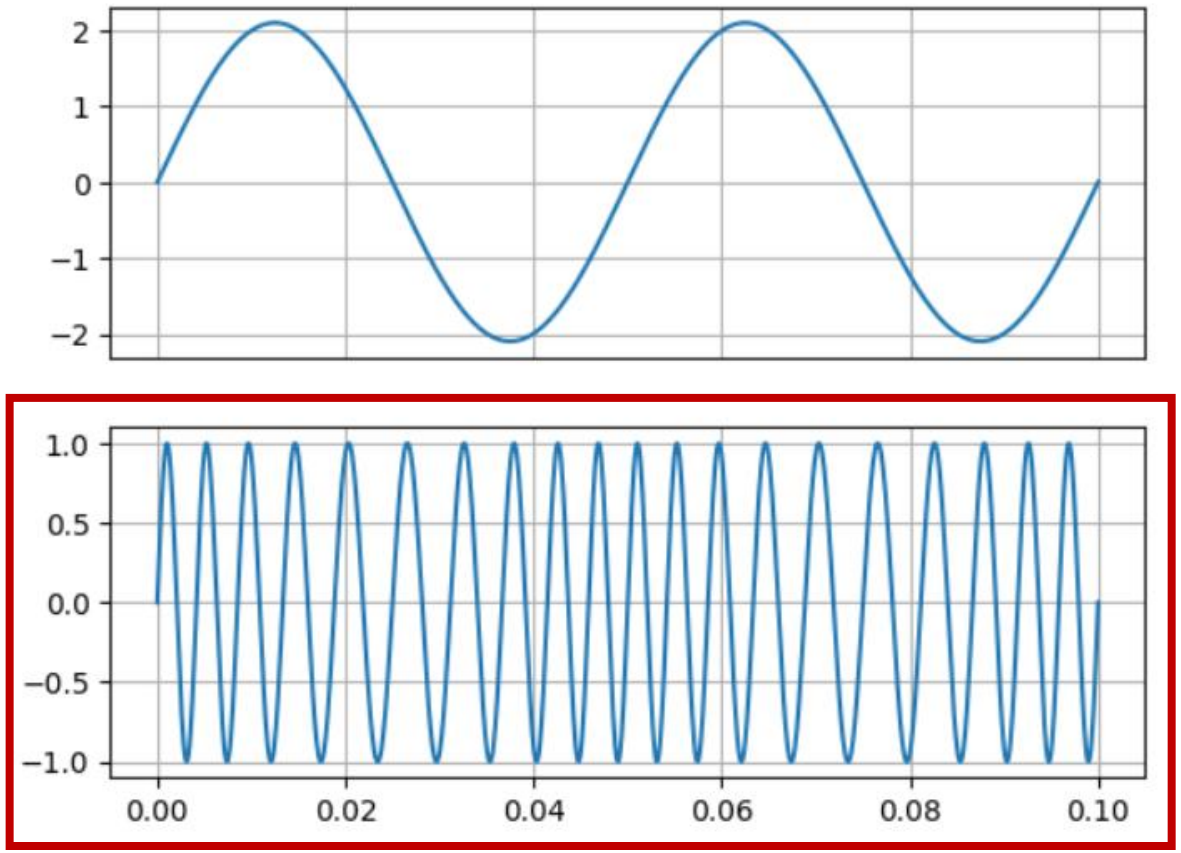
信号の生成

PM処理

この辺はグラフ処理

Colabでのシミュレーション

```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 0.1, 1000)
fc = 200

delta_phi = 120/180*np.pi
fs = 20
vs = delta_phi * np.sin(2*fs*np.pi*t)

v = np.sin(2*fc*np.pi*t + vs)

fig = plt.figure()
ax = fig.add_subplot(211)
ax.grid()
ax.plot(t, vs)
ay = fig.add_subplot(212)
ay.grid()
ay.plot(t, v)

axes = fig.get_axes()
for n in axes:
  n.tick_params(axis='x', which='both', bottom=False, labelbottom=False)
n.tick_params(axis='x', which='both', bottom=True, labelbottom=True)
```



授業ではこのグラフだけ作成

この部分を`plt.plot(t, v)`に置き換える

周波数変調

FMの式

信号

$$v_{FM} = V_c \sin(2\pi(f_c + \Delta\omega \sin(2\pi f_s t))\, t)$$

キャリアの**周波数成分**に信号を乗せる

» と、書きたいけれど、実際にはこの定義を実現する回路が無い。
（現実の回路は位相または位相の変化量を変える回路のみ）

FMの定義

信号

$$v_{FM} = V_c \sin(2\pi f_c t + \Delta\omega \int_0^t \sin(2\pi f_s t)\mathrm{dt})$$

PMとほとんど同じ式となる。

周波数変調と位相変調の関係1

周波数とはなにか?

位相の時間変化を示す値(位相の時間微分)

$\frac{d\Phi_s}{dt} = \boxed{\omega_s}$ » 位相を使って表す事のできる値

角周波数(2πで割ると周波数)

周波数変調における位相を考える

信号で周波数を変える » $\frac{d\Phi_s}{dt} = \boxed{s(t)}$ ← 位相の時間変化が信号s(t)になるようにする

したがって、ある時間の位相$\Phi_s(t)$は積分することで、

$$\Phi_s(t) = \int_0^t \sin(2\pi f_s t)\mathrm{dt})$$

となる。

周波数変調と位相変調の関係2

周波数変調の位相はキャリアの位相との合成になる。

$$\Phi(t) = 2\pi f_c t + \Delta\omega \int_0^t \sin(2\pi f_s t)\mathrm{dt})$$

FMの位相項

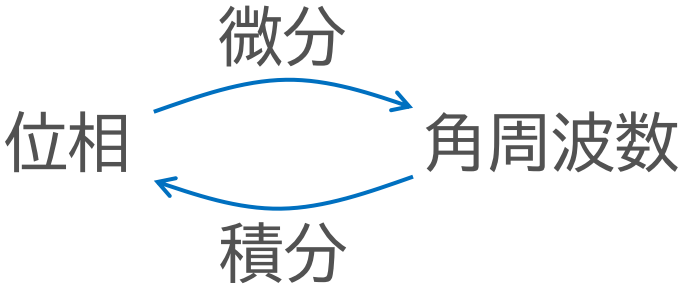
キャリアの位相項

したがってFMの式は、

$$v_{FM} = V_c \sin(2\pi f_c t + \Delta\omega \int_0^t \sin(2\pi f_s t)\mathrm{dt})$$

となる。

位相と周波数は次の関係がある。 »



周波数変調のシミュレーション

```
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(0, 0.1, 1000)
fc = 200

delta_omega = 0.1
fs = 20
vs = delta_omega * np.sin(2*fs*np.pi*t)
phi_s = np.cumsum(vs)

v = np.sin(2*fc*np.pi*t + phi_s)

fig = plt.figure()
ax = fig.add_subplot(211)
ax.grid()
ax.plot(t, phi_s)
ay = fig.add_subplot(212)
ay.grid()
ay.plot(t, v)

axes = fig.get_axes()
for n in axes:
  n.tick_params(axis='x', which='both', bottom=False,
labelbottom=False)
n.tick_params(axis='x', which='both', bottom=True,
labelbottom=True)
```

0～0.1秒の時間軸を作る

200Hzのキャリア周波数を設定

最大周波数偏移Δωを設定

信号の周波数を設定

信号の生成

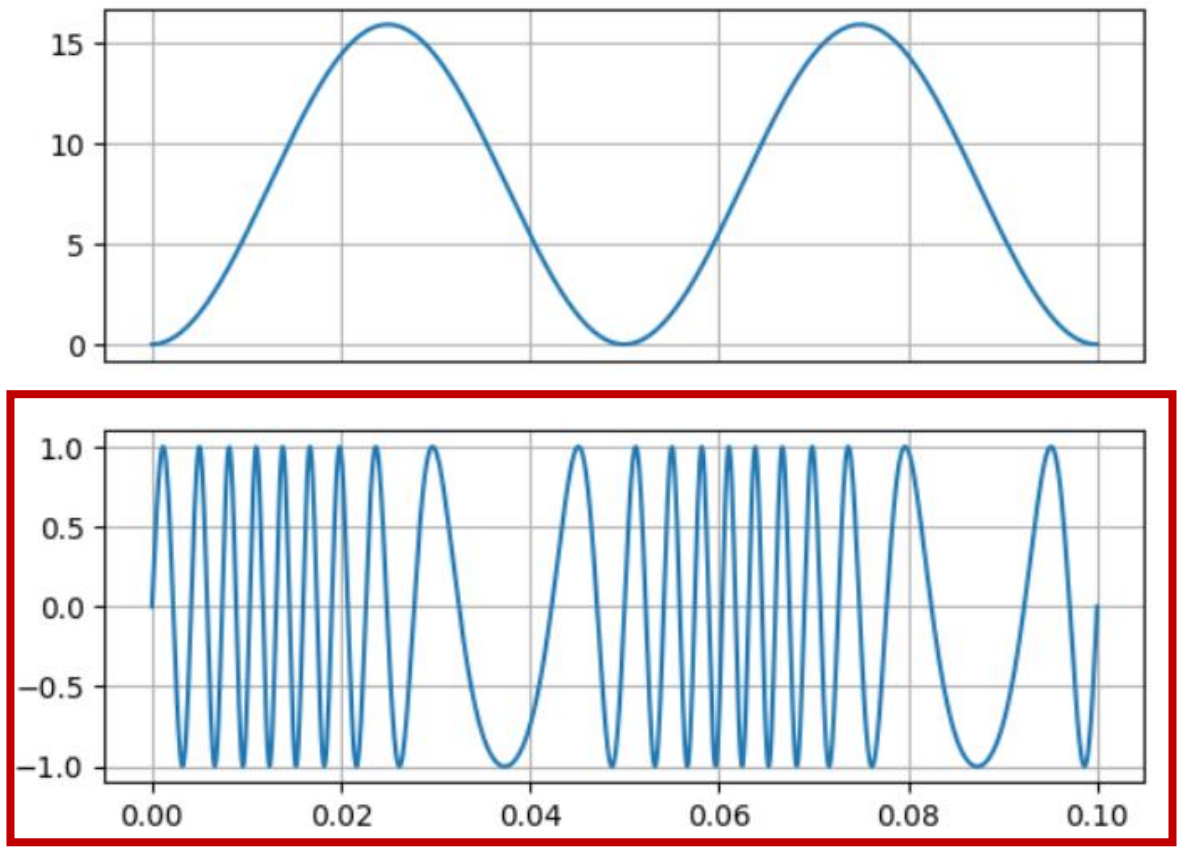
積分処理(離散信号では累積和(Σ)で代替

FM処理

この辺はグラフ処理

Colabでのシミュレーション

```
[18] import numpy as np
     import matplotlib.pyplot as plt

     t = np.linspace(0, 0.1, 1000)
     fc = 200

     delta_omega = 0.1
     fs = 20
     vs = delta_omega * np.sin(2*fs*np.pi*t)
     phi_s = np.cumsum(vs)

     v = np.sin(2*fc*np.pi*t + phi_s)

     fig = plt.figure()
     ax = fig.add_subplot(211)
     ax.grid()
     ax.plot(t, phi_s)
     ay = fig.add_subplot(212)
     ay.grid()
     ay.plot(t, v)

     axes = fig.get_axes()
     for n in axes:
       n.tick_params(axis='x', which='both', bottom=False, labelbottom=False)
     n.tick_params(axis='x', which='both', bottom=True, labelbottom=True)
```

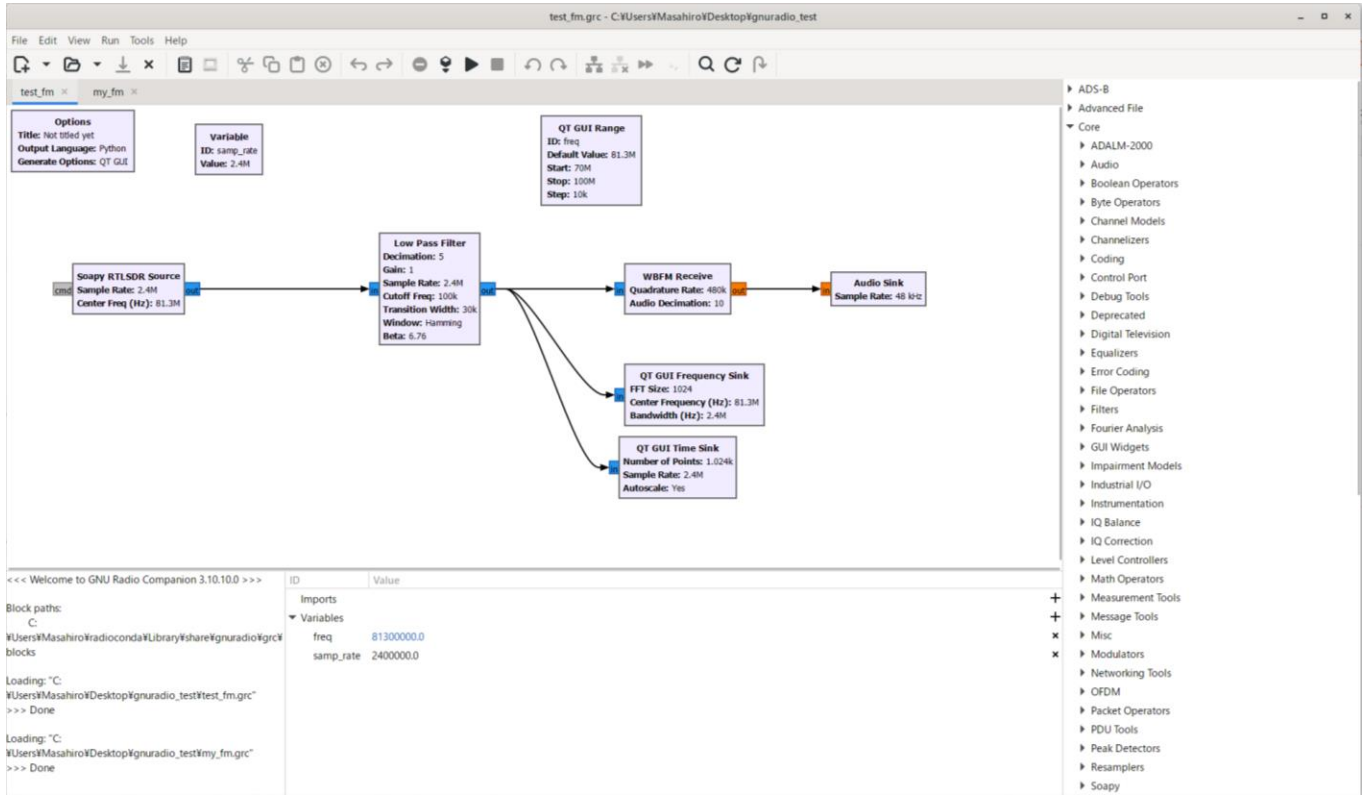


授業ではこのグラフだけ作成

この部分を`plt.plot(t,v)`に置き換える

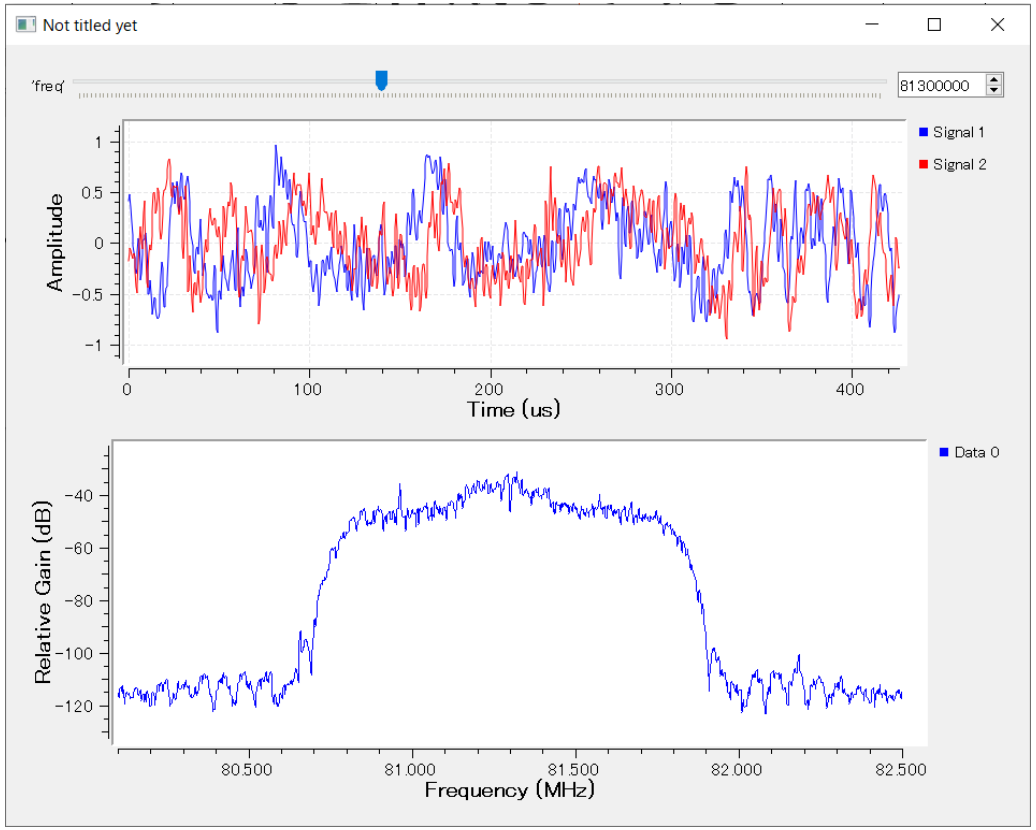
SDRの体験

ここではFMラジオを作成してみる



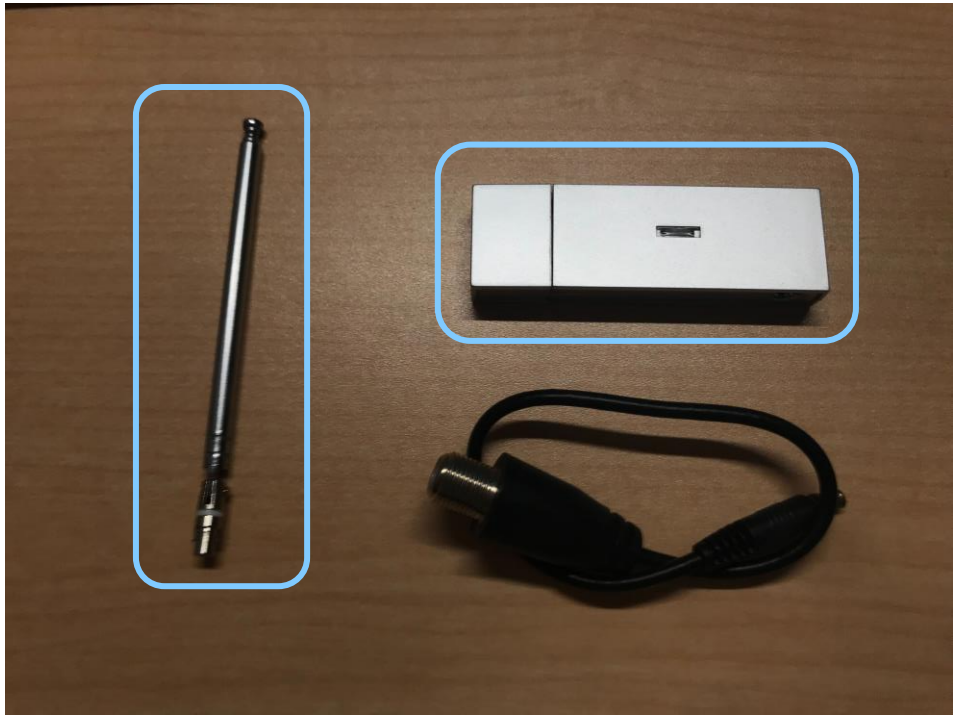
こんなブロック図で

こんな画面が出る



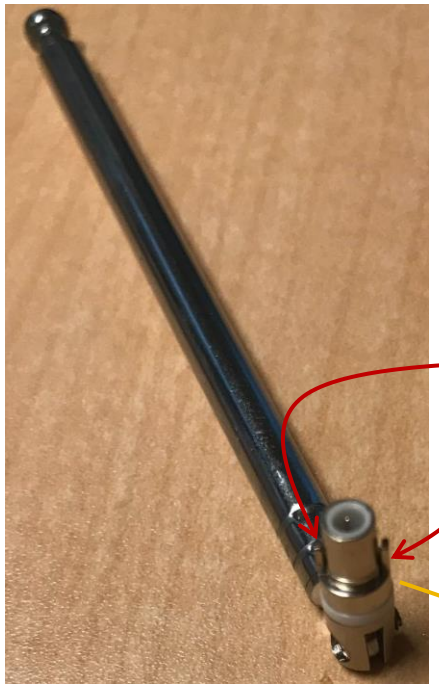
受信機の準備

内容物



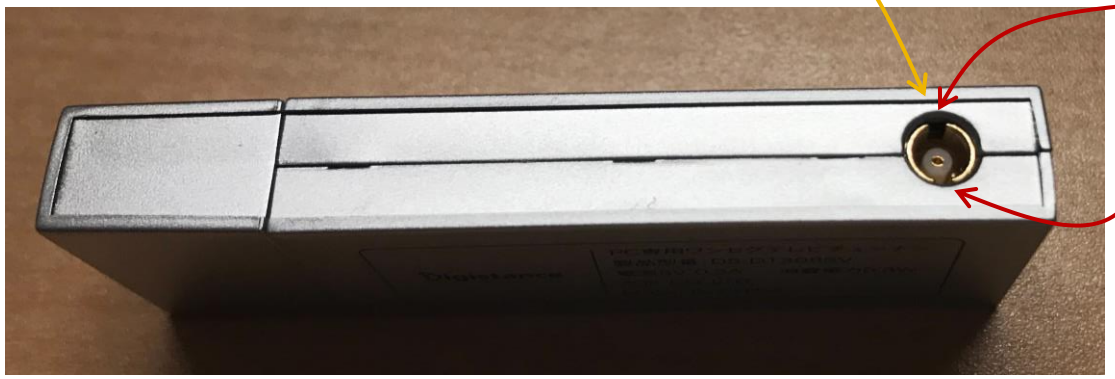
今回使うのはアンテナと銀色の本体

アンテナの接続



つめ

切り欠きとつめを
合わせて差し込む

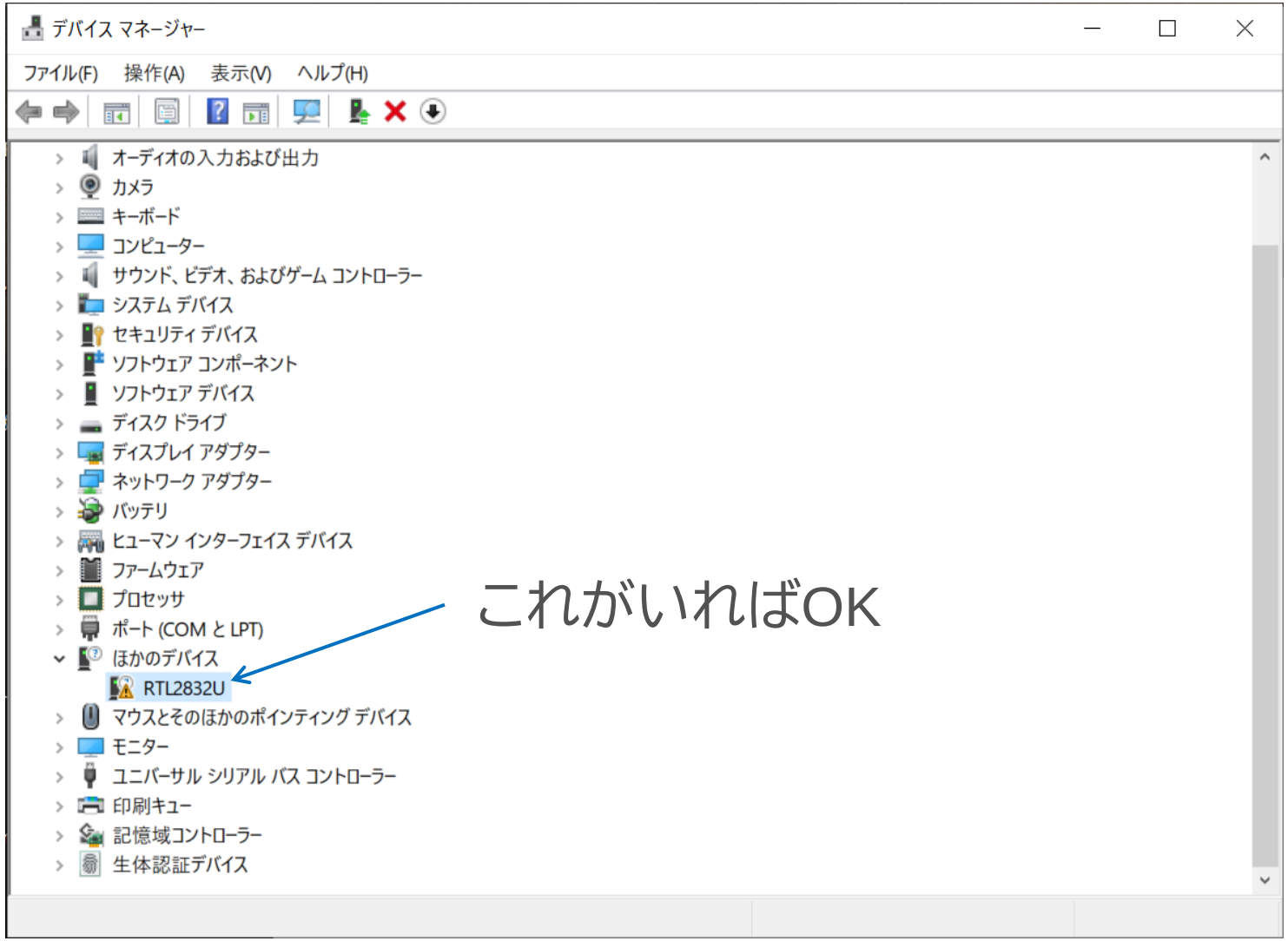


切り欠き

本体とアンテナの縁が同じに
なるくらいまで押し込む

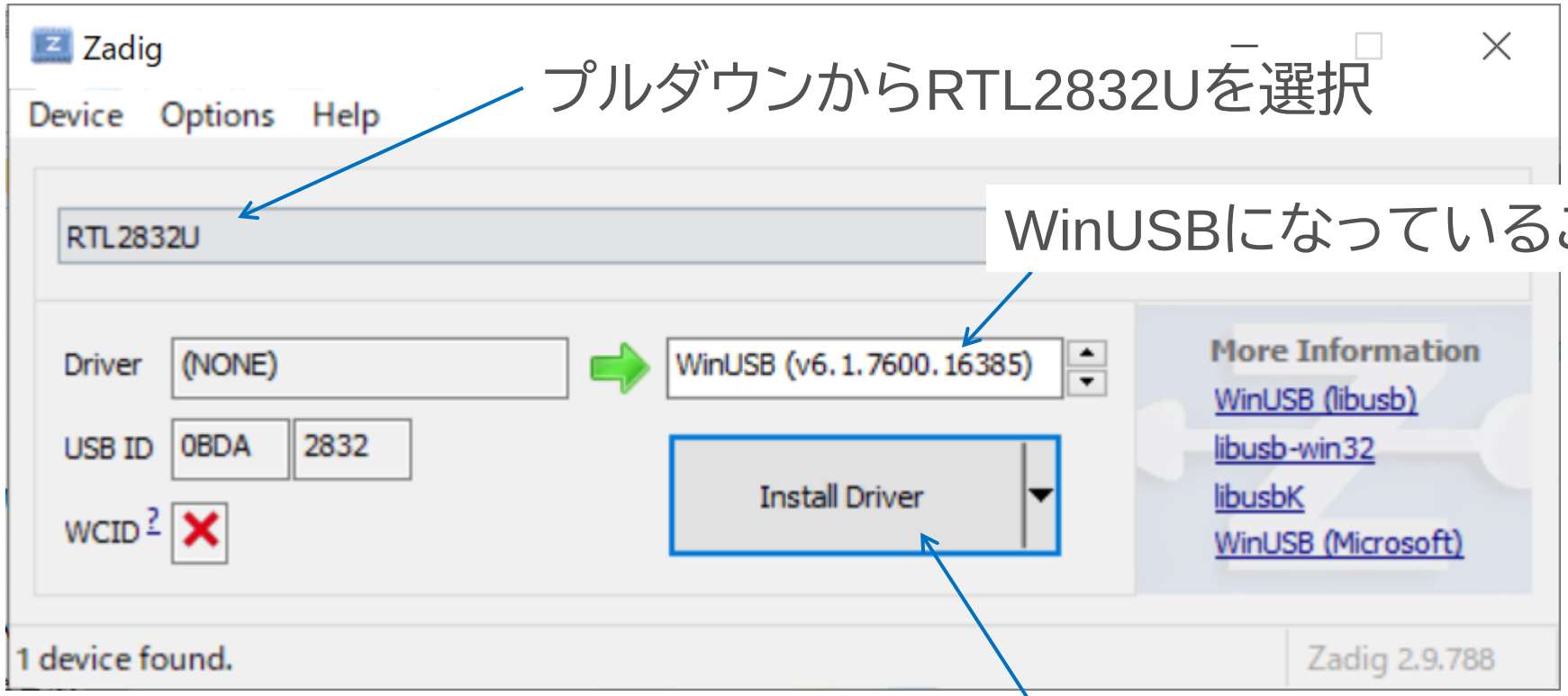
ドライバの入れ替え

デバイスが認識しているか確認



Zadigの起動

事前にダウンロードしたZadigを起動する。

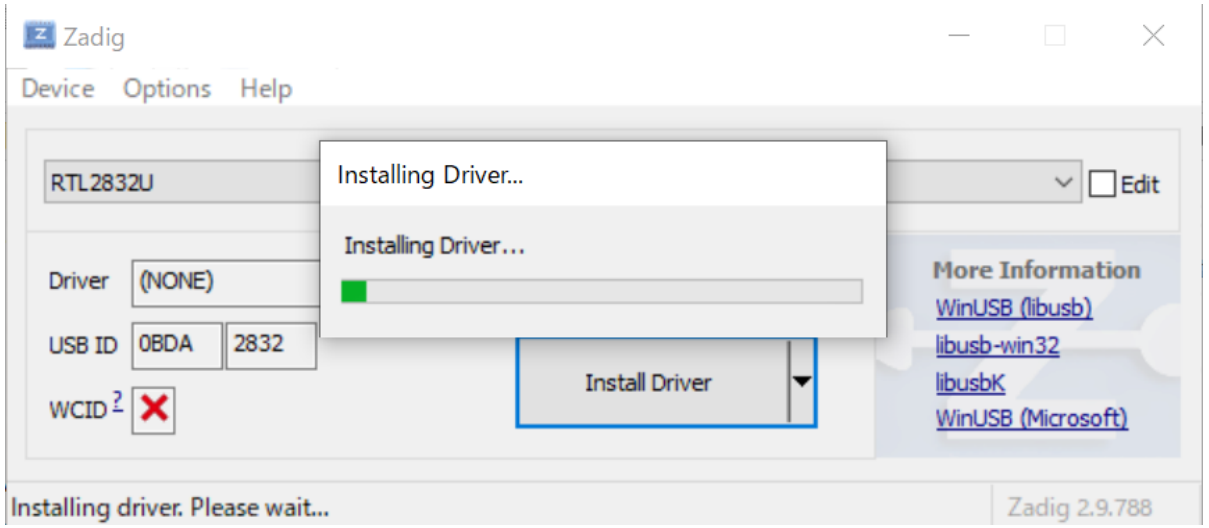


プルダウンからRTL2832Uを選択

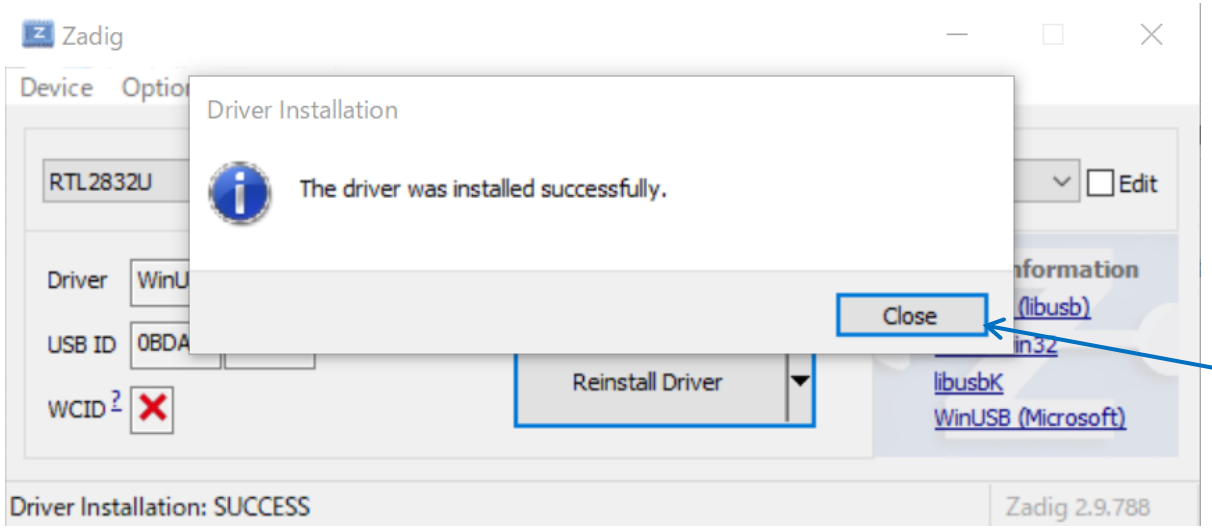
WinUSBになっていることを確認

対象とドライバを確認したら左クリック

インストール中



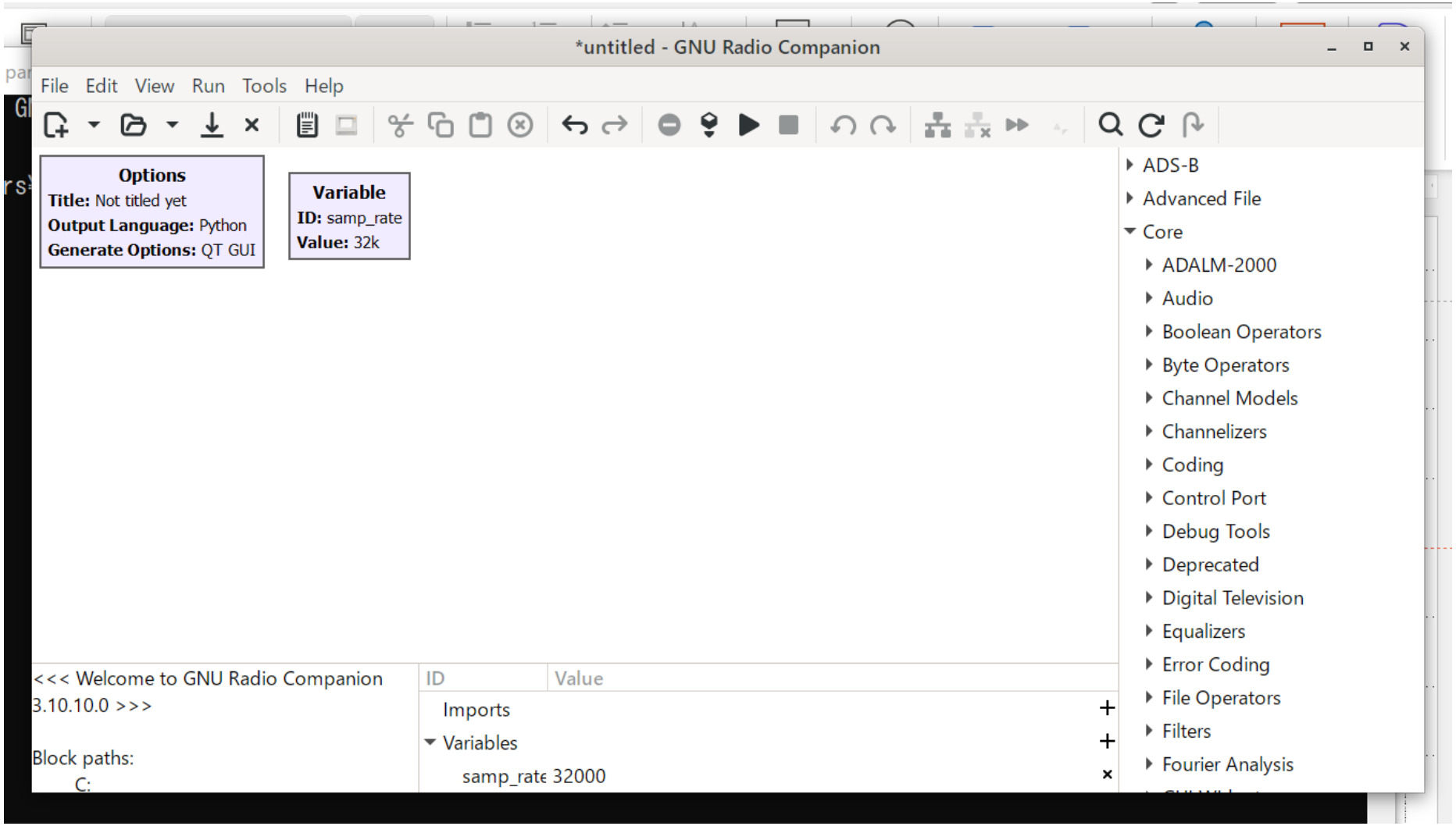
インストール完了



左クリックで閉じる

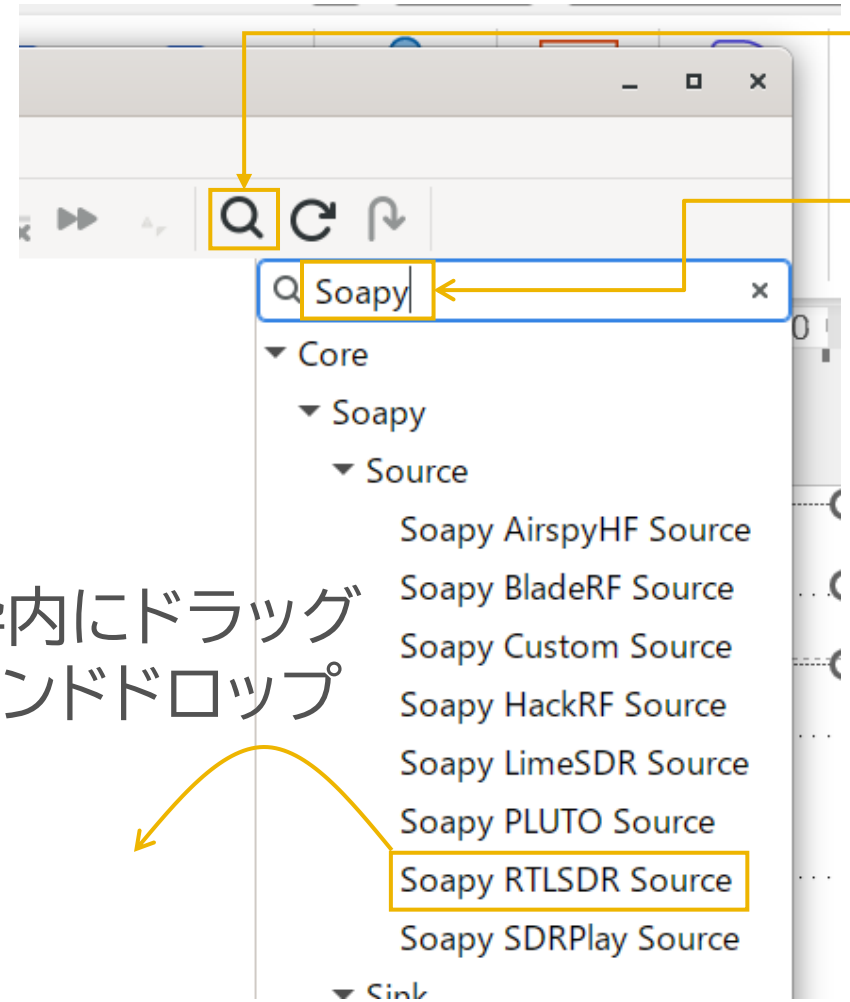
FMラジオの実装

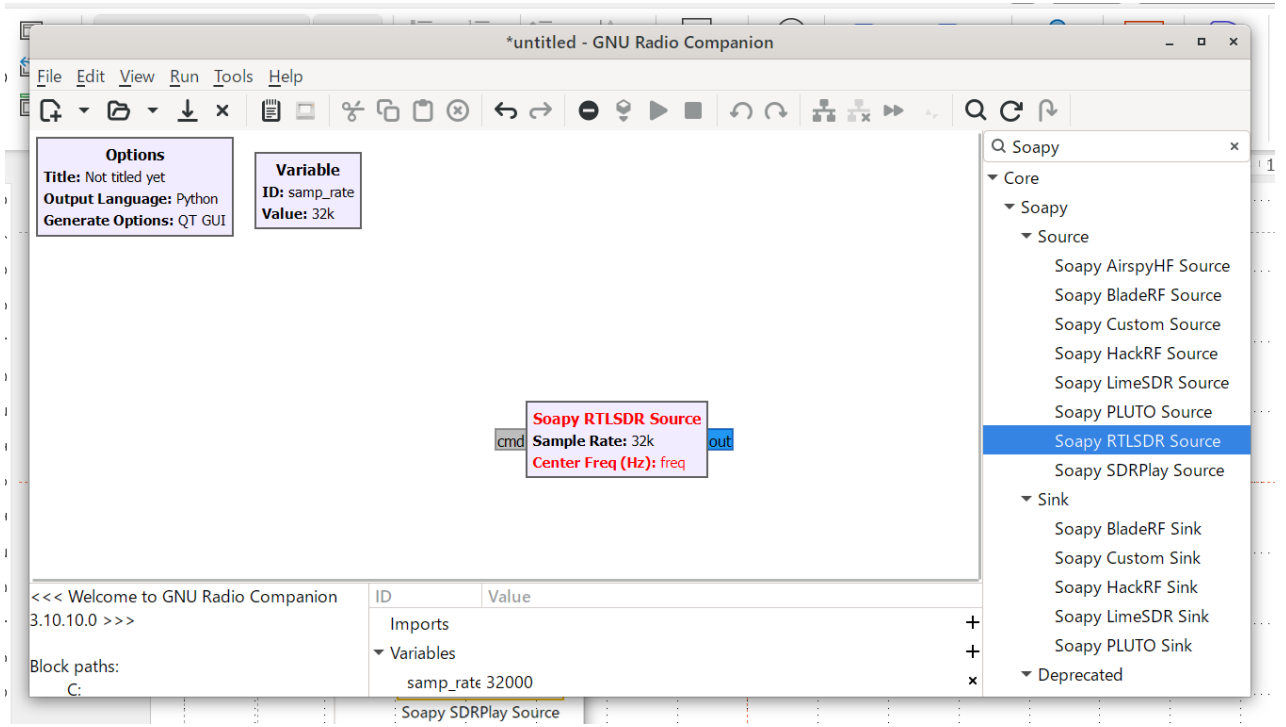
GNU Radioを起動



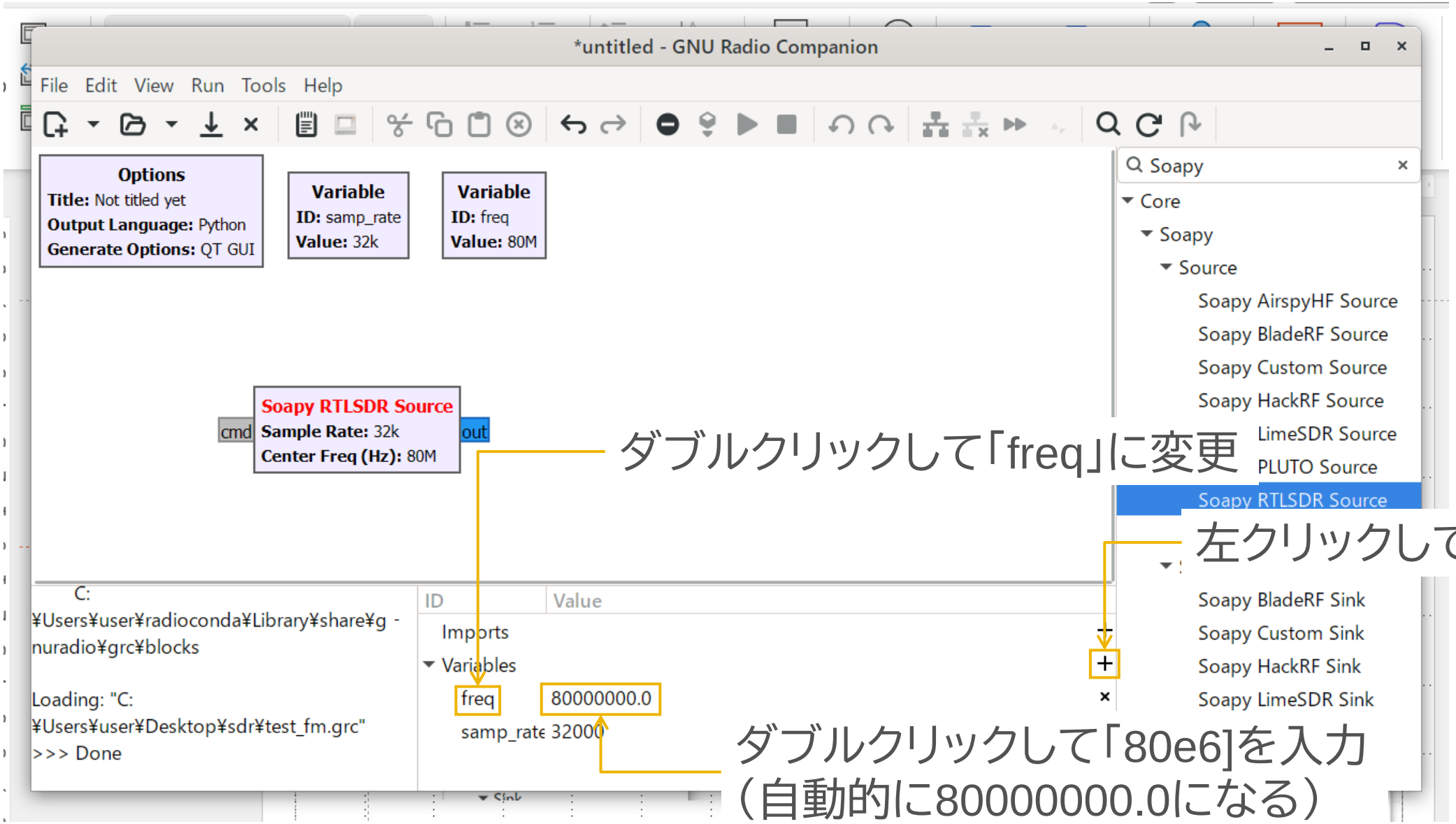
起動画面はこんな感じ

RTL SDRを追加

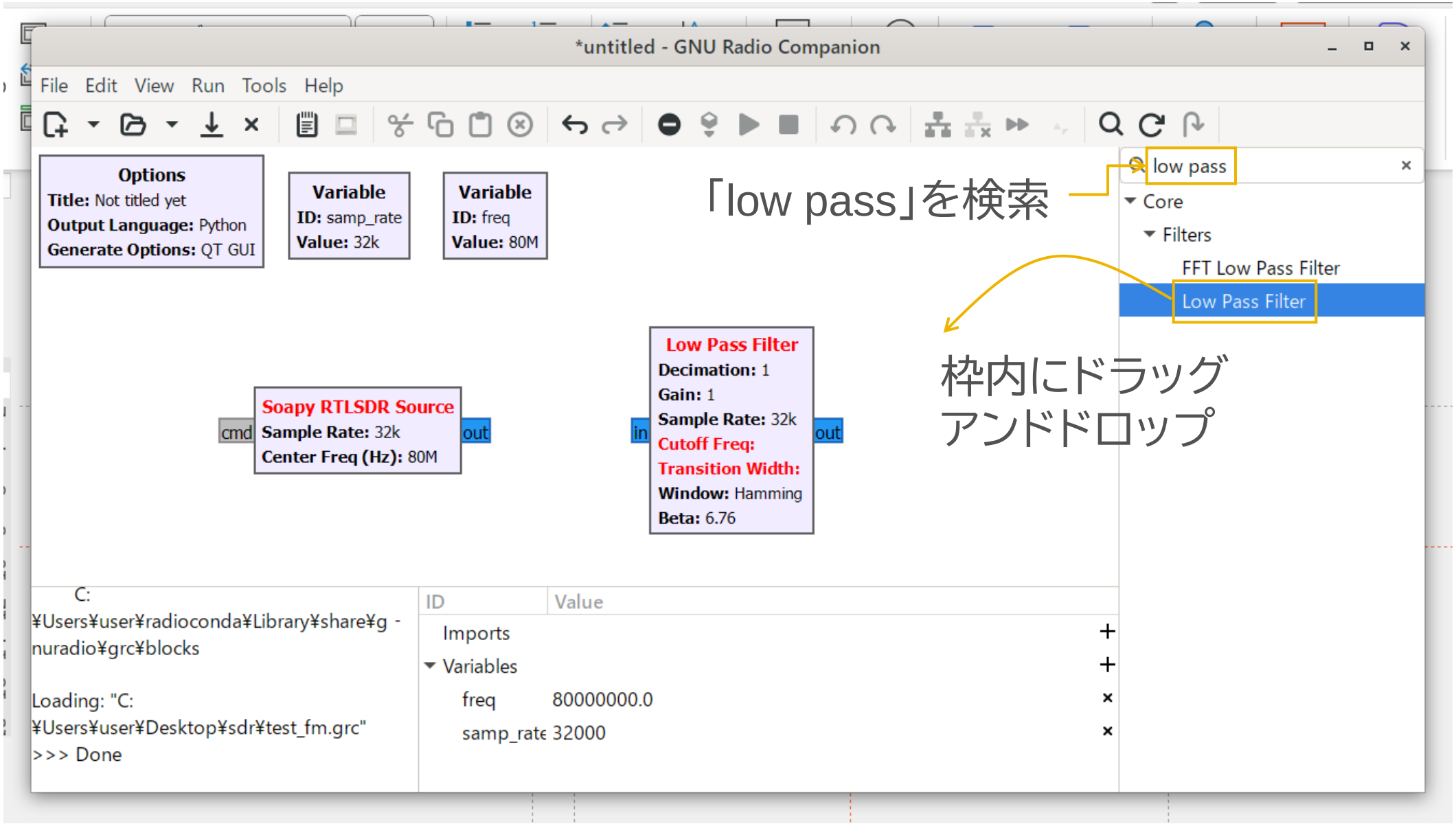

虫眼鏡(検索)を左クリック
Soapyを入力
Soapy
Core
Soapy
Source
Soapy AirspyHF Source
Soapy BladeRF Source
Soapy Custom Source
Soapy HackRF Source
Soapy LimeSDR Source
Soapy PLUTO Source
Soapy RTLSDR Source
Soapy SDRPlay Source
Sink
枠内にドラッグ
アンドドロップ


*untitled - GNU Radio Companion
File Edit View Run Tools Help
Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI
Variable
ID: samp_rate
Value: 32k
Soapy RTLSDR Source
cmd
Sample Rate: 32k
Center Freq (Hz): freq
out
<<< Welcome to GNU Radio Companion
3.10.10.0 >>>
Block paths:
C:
ID
Value
Imports
Variables
samp_rate 32000
Soapy SDRPlay Source
Soapy
Core
Soapy
Source
Soapy AirspyHF Source
Soapy BladeRF Source
Soapy Custom Source
Soapy HackRF Source
Soapy LimeSDR Source
Soapy PLUTO Source
Soapy RTLSDR Source
Soapy SDRPlay Source
Sink
Soapy BladeRF Sink
Soapy Custom Sink
Soapy HackRF Sink
Soapy LimeSDR Sink
Soapy PLUTO Sink
Deprecated

変数の追加

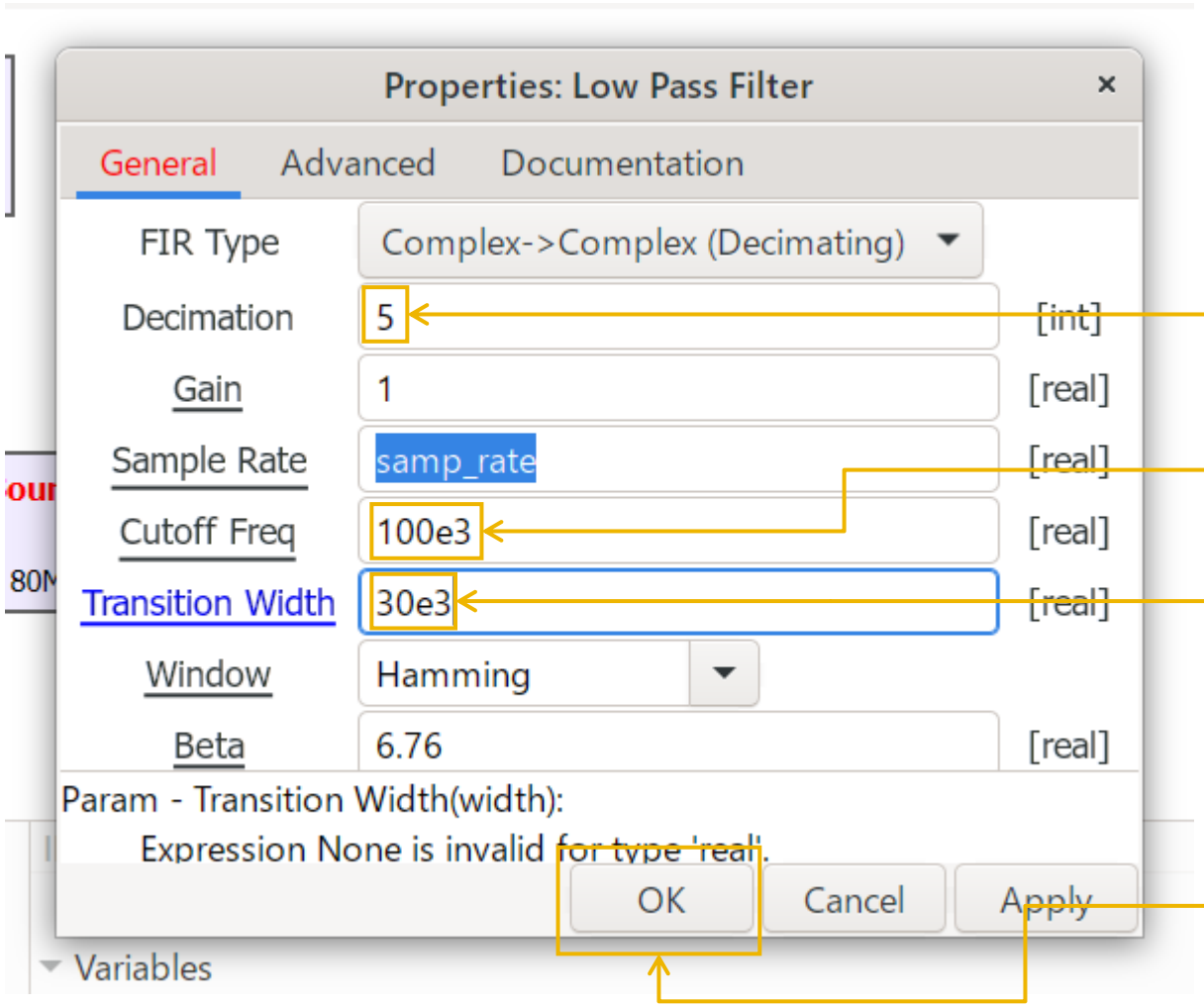

*untitled - GNU Radio Companion
File Edit View Run Tools Help
Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI
Variable
ID: samp_rate
Value: 32k
Variable
ID: freq
Value: 80M
Soapy RTLSDR Source
cmd
Sample Rate: 32k
Center Freq (Hz): 80M
out
Soapy
Core
Soapy
Source
Soapy AirspyHF Source
Soapy BladeRF Source
Soapy Custom Source
Soapy HackRF Source
LimeSDR Source
PLUTO Source
Soapy RTLSDR Source
Soapy BladeRF Sink
Soapy Custom Sink
Soapy HackRF Sink
Soapy LimeSDR Sink
C:
¥Users¥user¥radioconda¥Library¥share¥g -
nuradio¥grc¥blocks
Loading: "C:
¥Users¥user¥Desktop¥sdr¥test_fm.grc"
>>> Done
ID
Value
Imports
Variables
freq
80000000.0
samp_rate 32000
ダブルクリックして「freq」に変更
左クリックして変数を追加
ダブルクリックして「80e6]を入力
（自動的に80000000.0になる）

Low pass filterの追加



Low pass filter の設定

Low pass filterブロックをダブルクリックして設定画面を開く



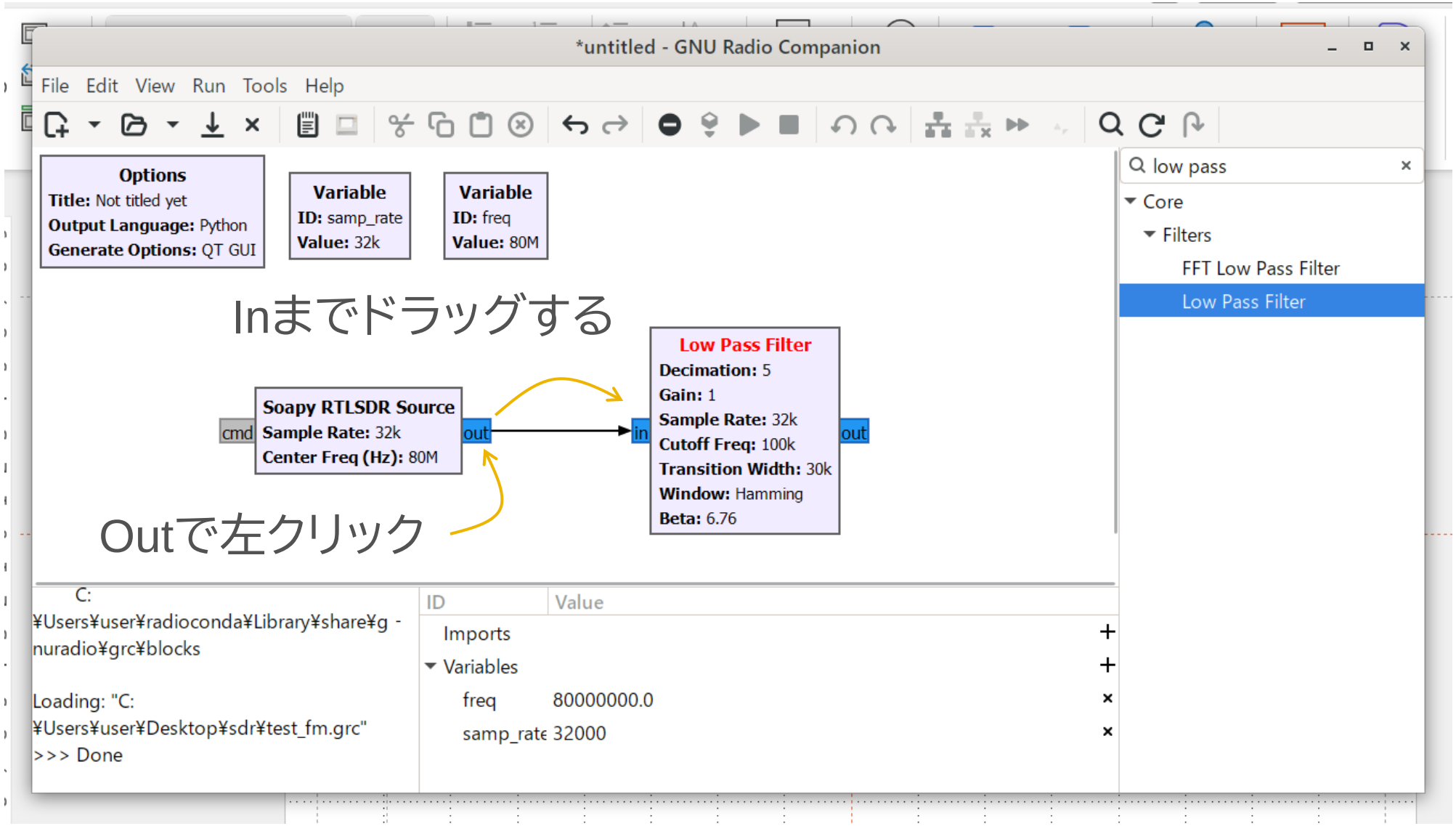
5を入力

100e3を入力

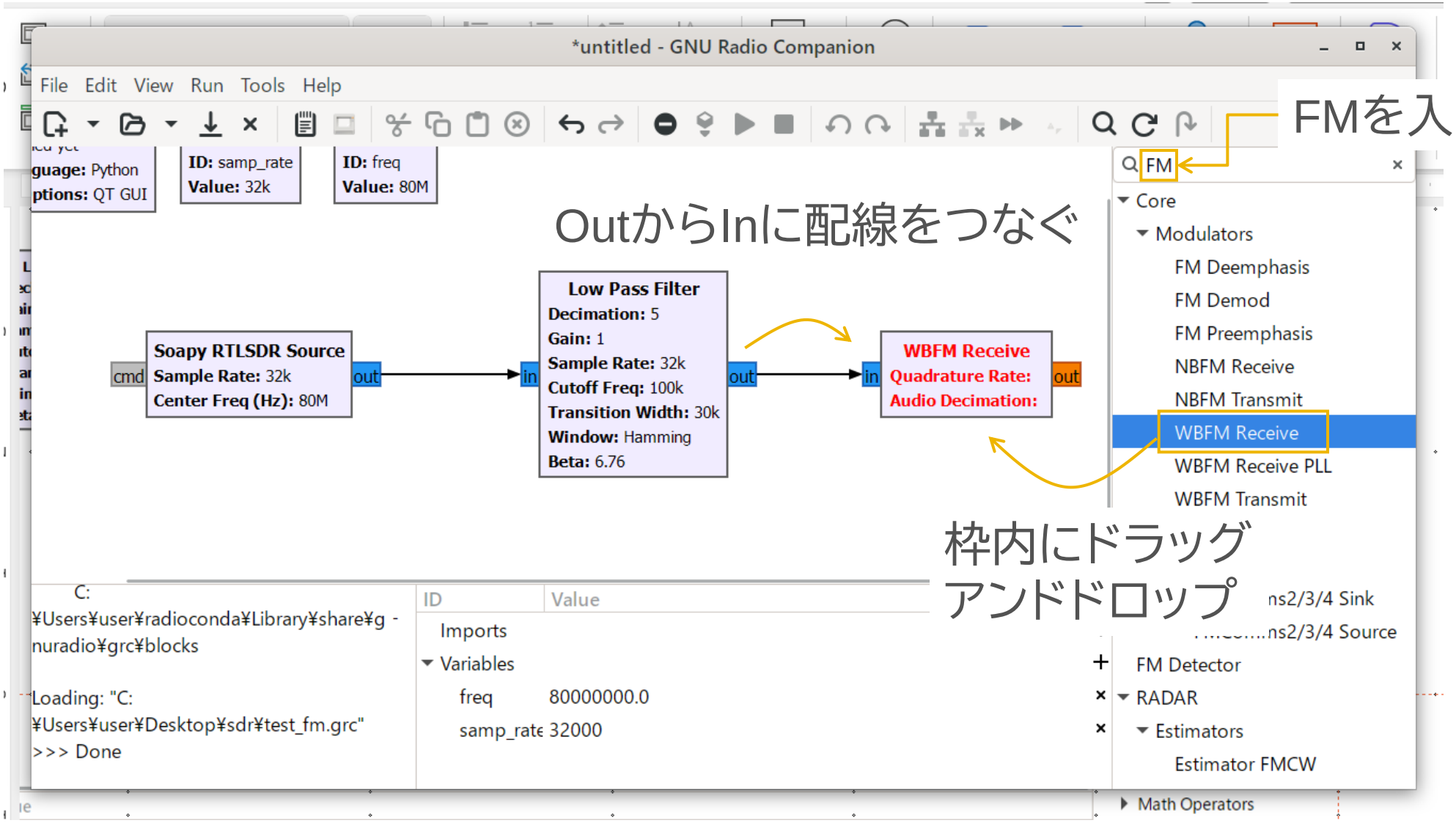
30e3を入力

入力が終わったらOKをクリック

SourceとLow pass filterの接続

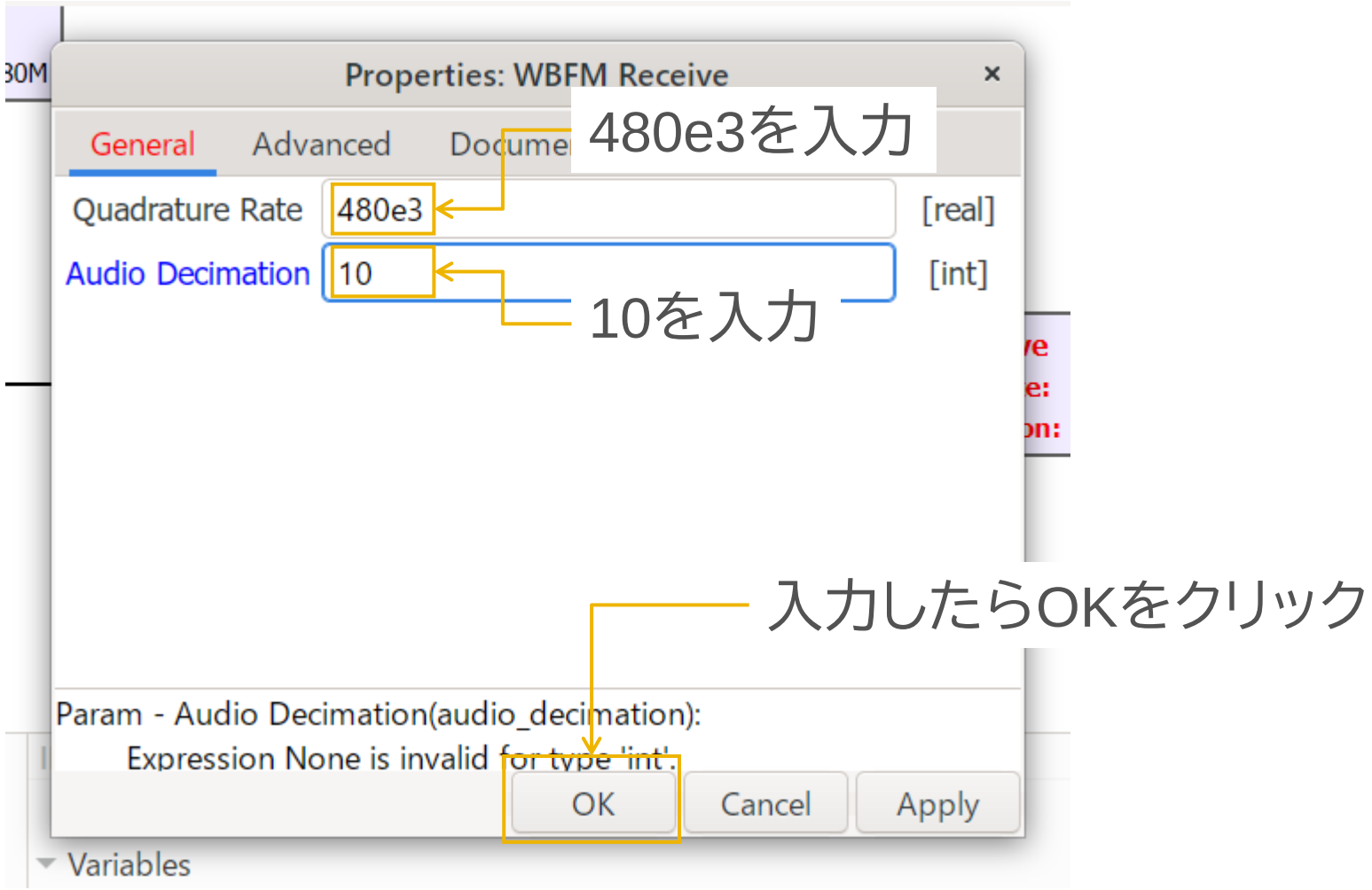


WBFM receiverブロックの配置

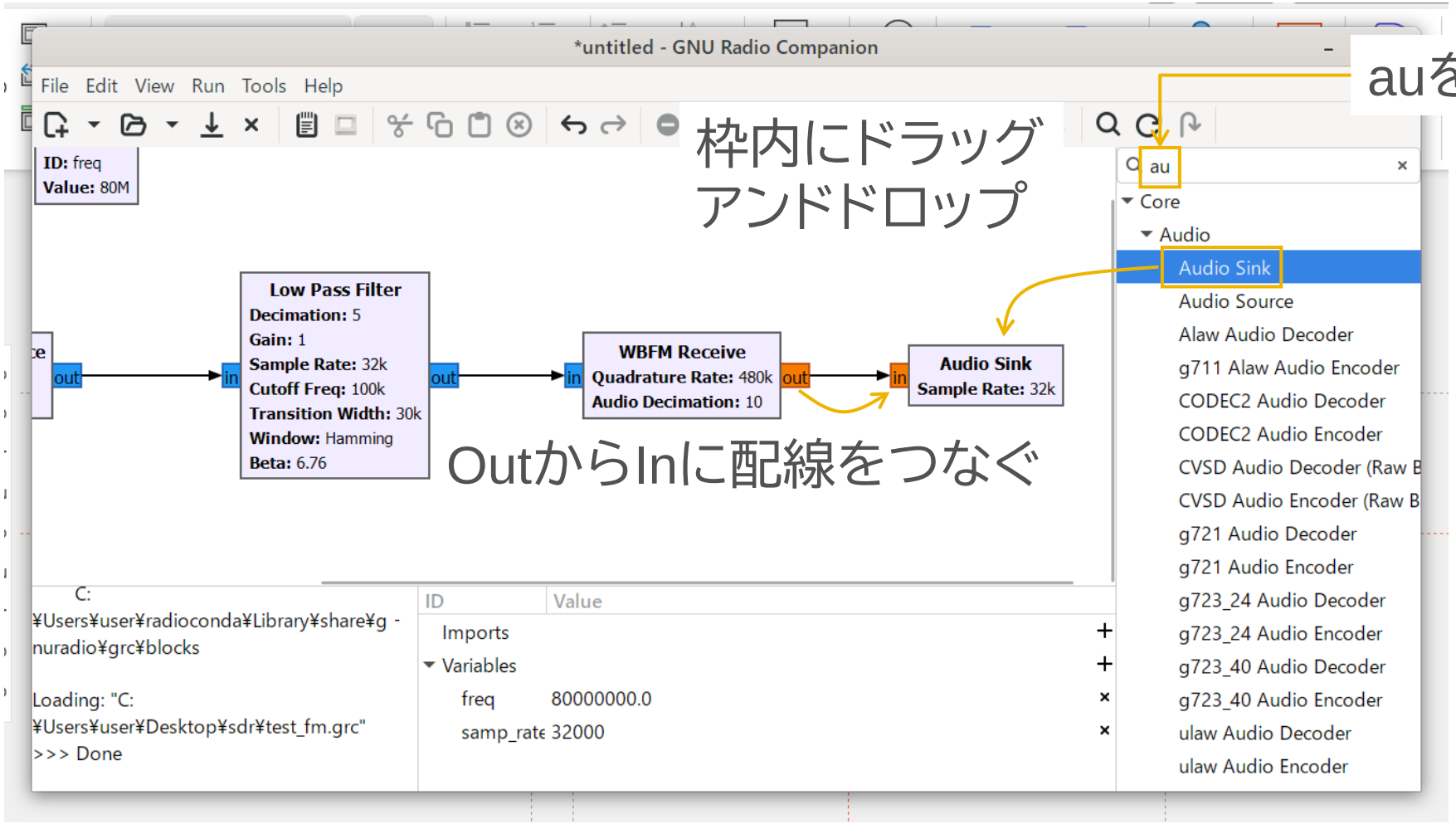


WBFM receiverブロックの設定

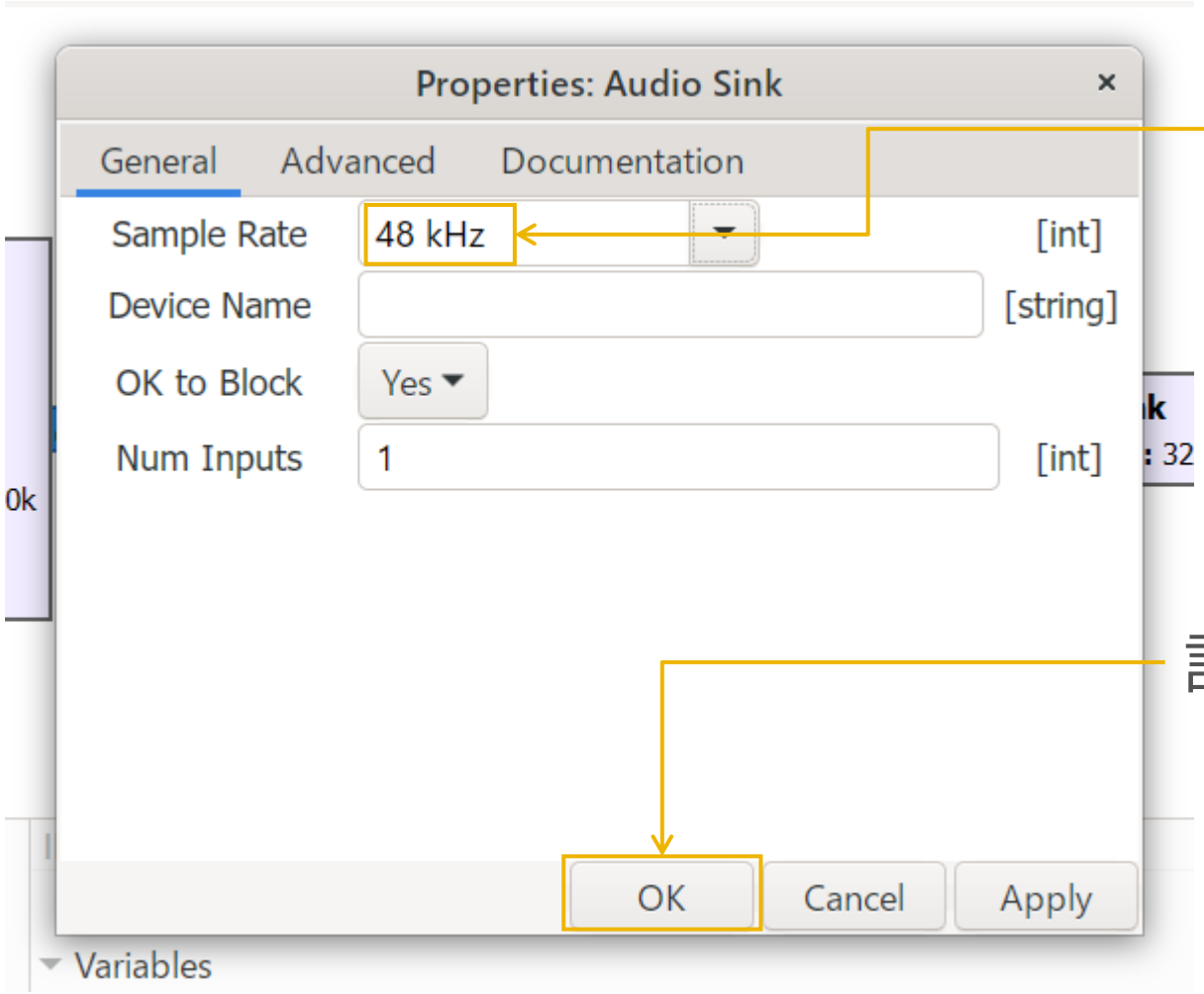
WFBM receiverをダブルクリックして設定を開く



Audio Sinkの接続



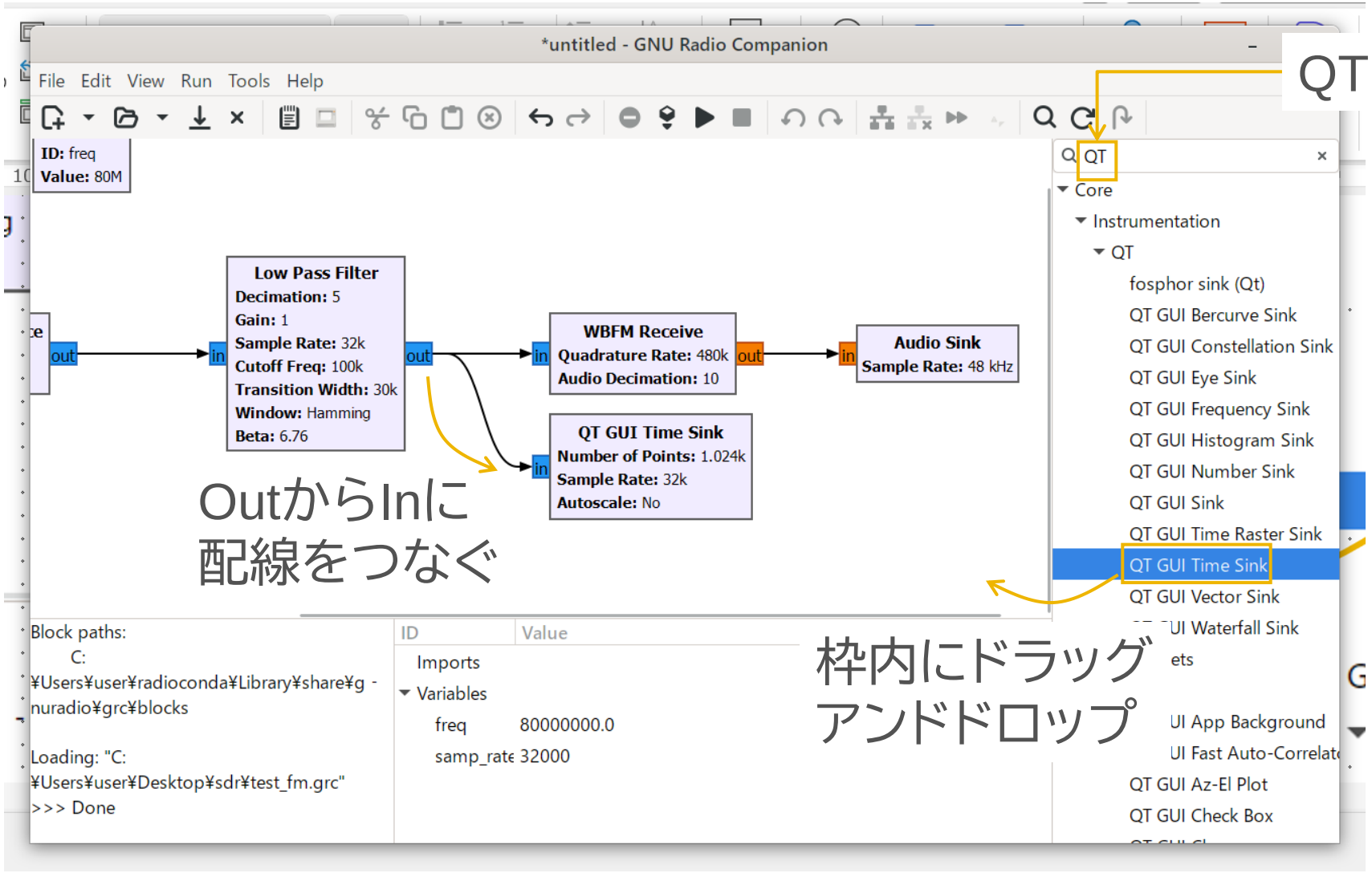
Audio Sinkの設定



プルダウンメニューから「48kHz」を選択

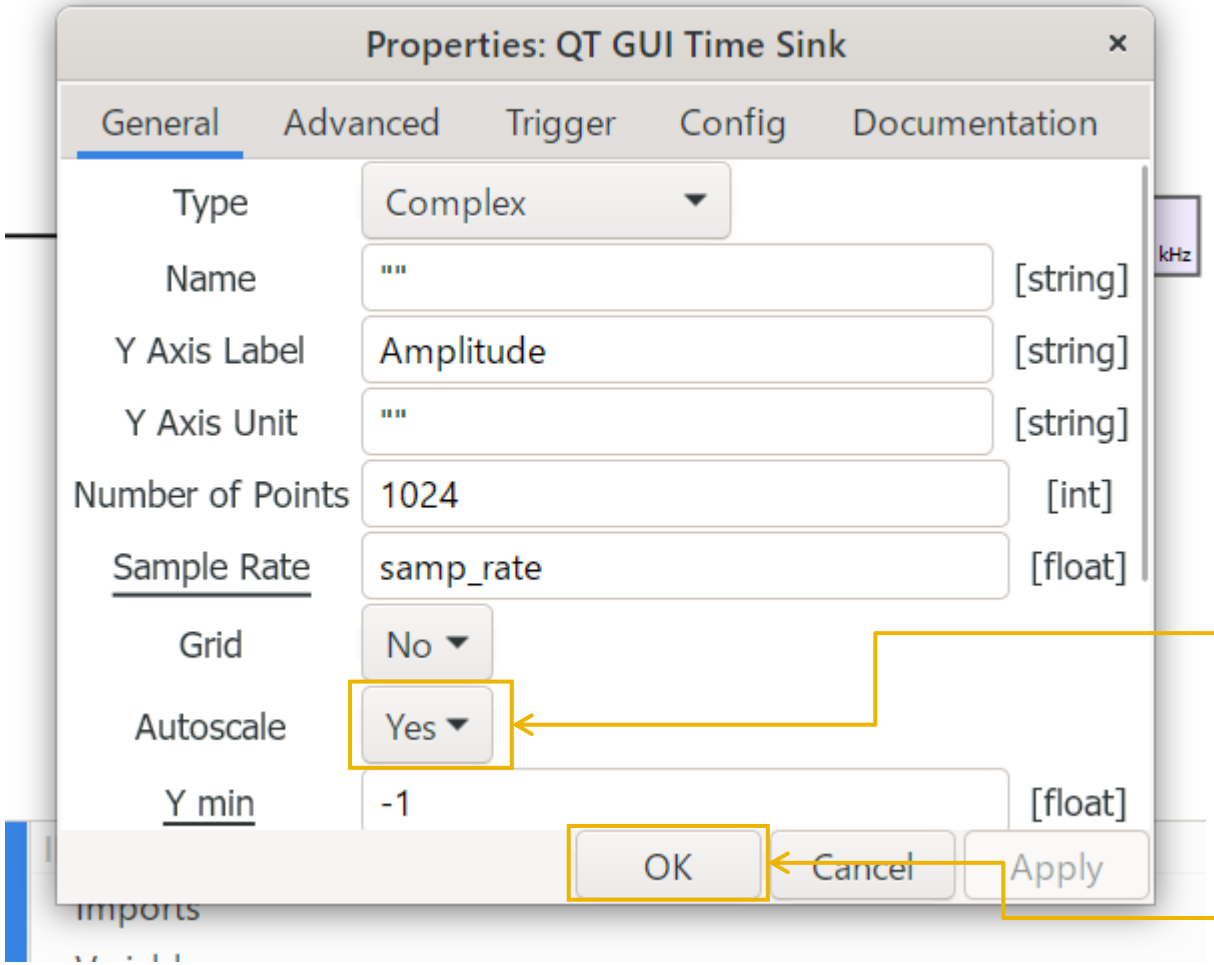
設定したらOKをクリック

QT GUI Time sinkの接続



QT GUI Time sinkの設定

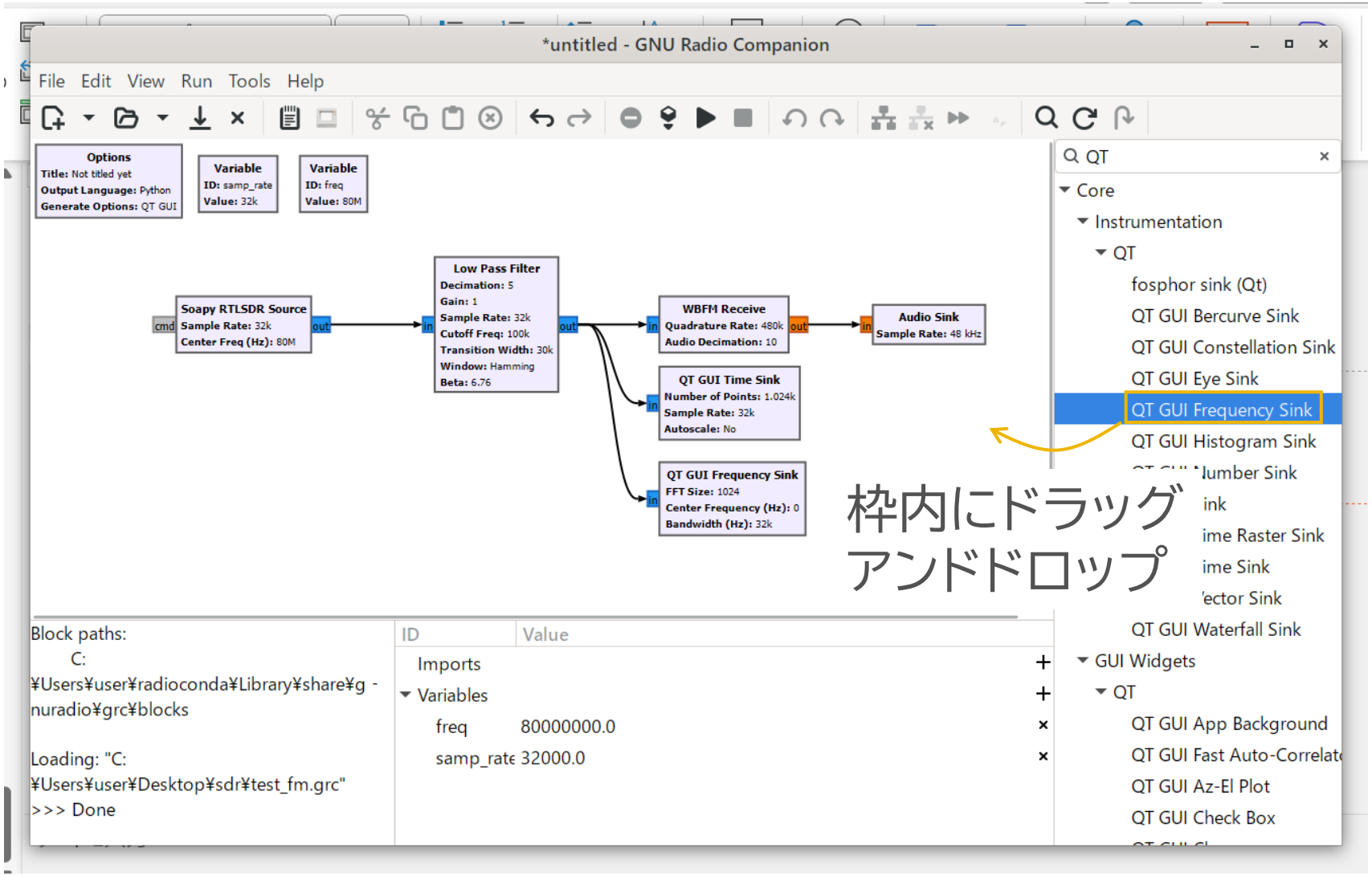
QT GUI Time sinkブロックをダブルクリックして設定を開く



AutoscaleをYesに変更する

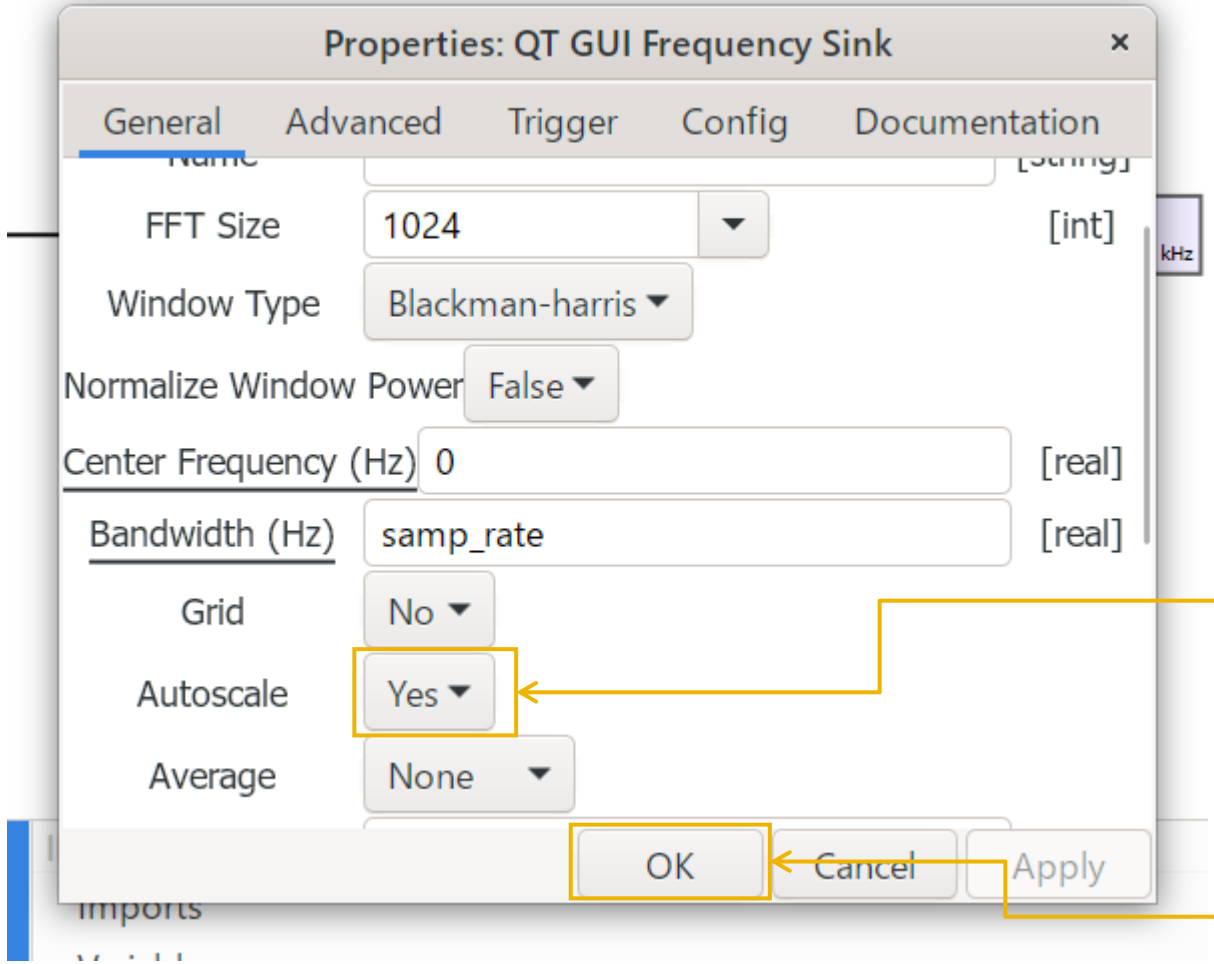
設定したらOKをクリック

QT GUI Frequency sinkの接続



QT GUI Frequency sinkの設定

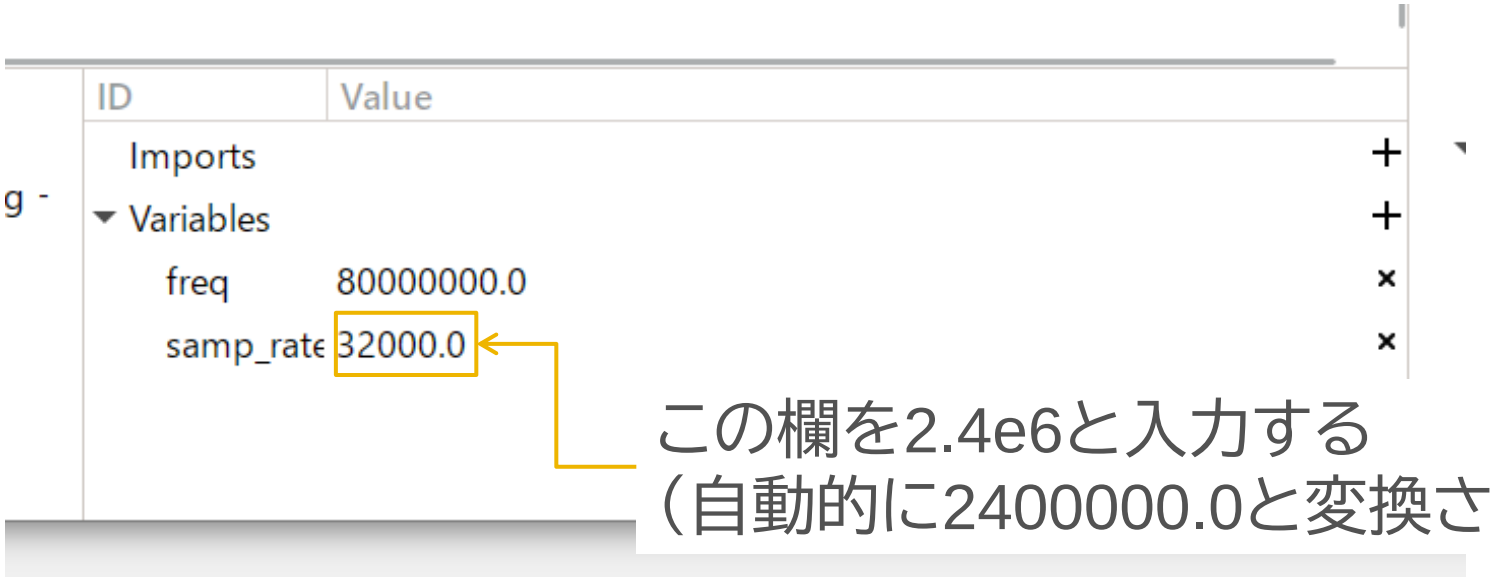
QT GUI Frequency sinkブロックをダブルクリックして設定を開く



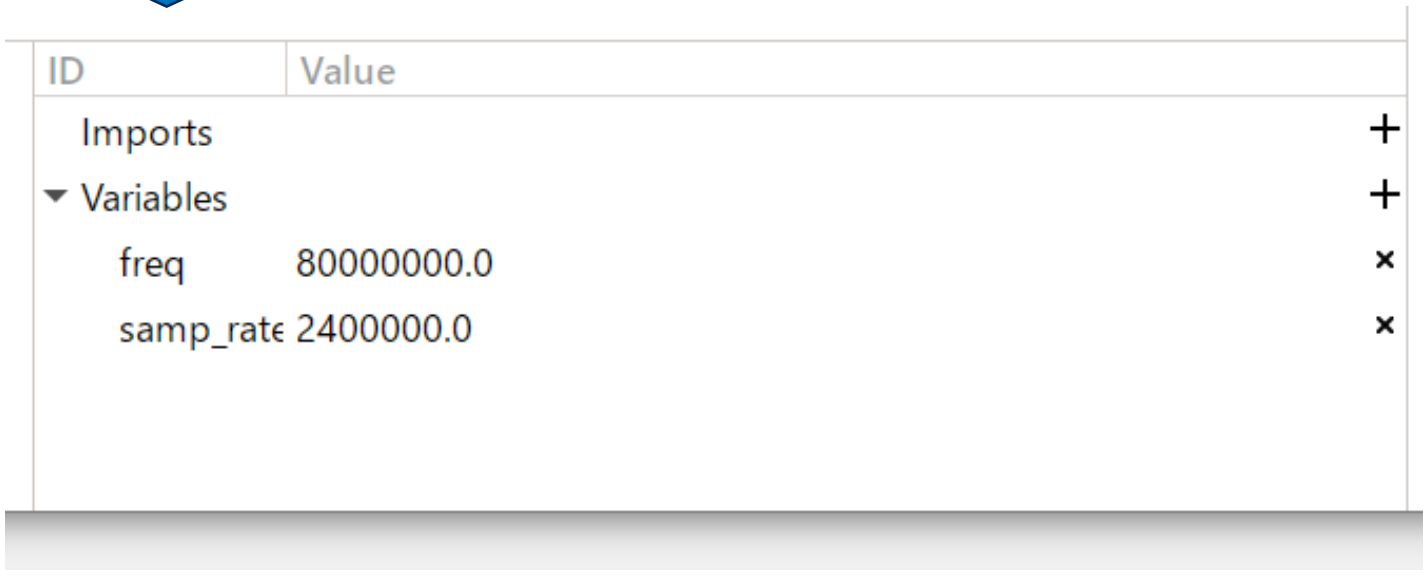
AutoscaleをYesに変更する

設定したらOKをクリック

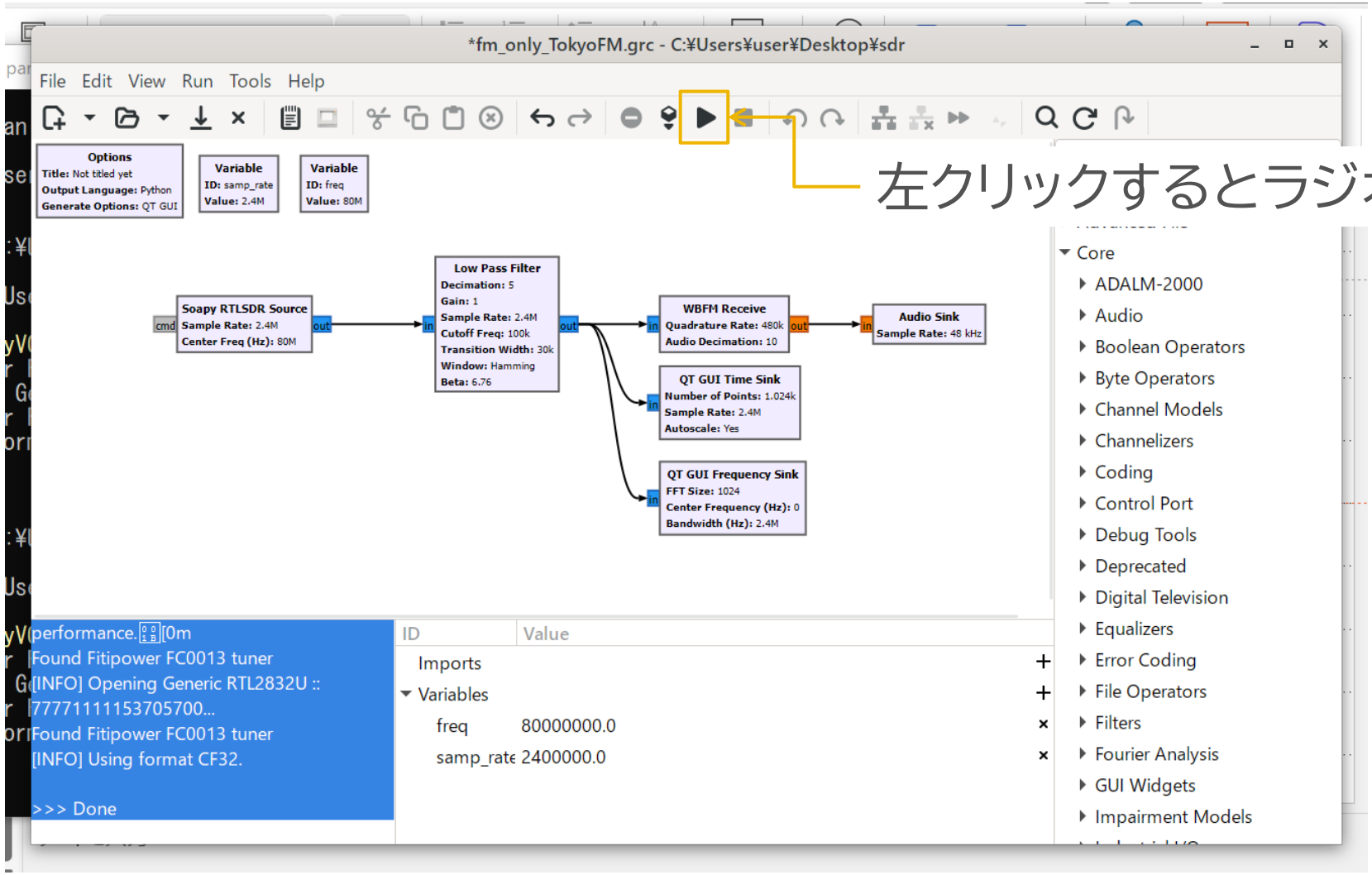
samp_rateの設定



この欄を2.4e6と入力する
（自動的に2400000.0と変換される

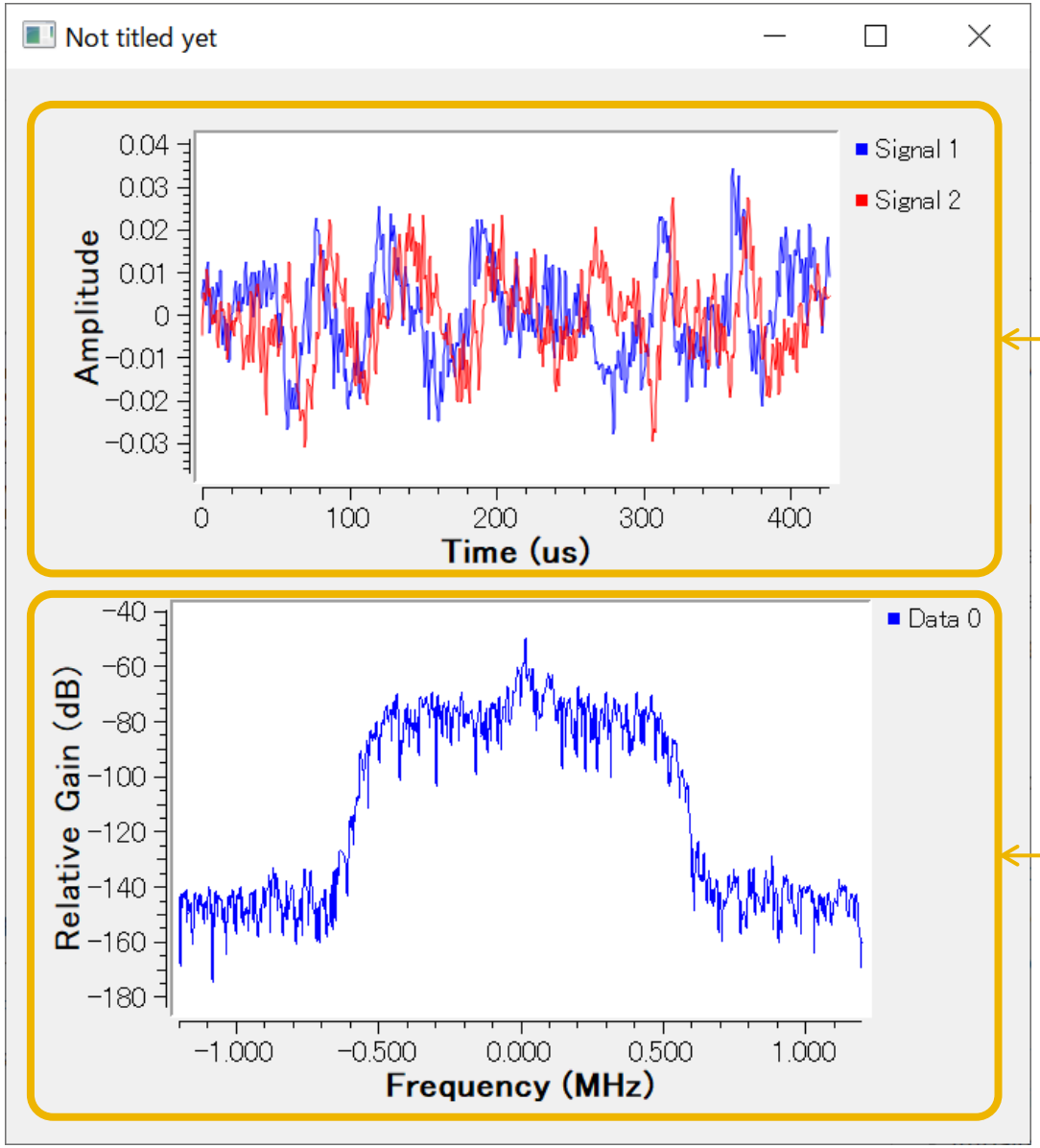


ラジオの起動



左クリックするとラジオが起動する

画面



IQの時系列信号

IFのスペクトラム

この状態でスピーカーからは
Tokyo FM (80.0MHz)が聞こえる…はず。

次回以降では…

- ブロック線図それぞれがどのような役割をしているのか？
- それぞれのブロックの中身は何をしているのか？
- 現代の無線機を構成する基本技術について
- WBFM receiverブロックを使わずにプリミティブなブロックのみで AM、FM受信機を作成