次世代アマチュア衛星受信のスタンダード「SDR+GNU Radio」環境構築・使いこなし指南

今村謙之(Noritsuna Imamura)
JI1SZP
noritsuna@siprop.org

×SIProp

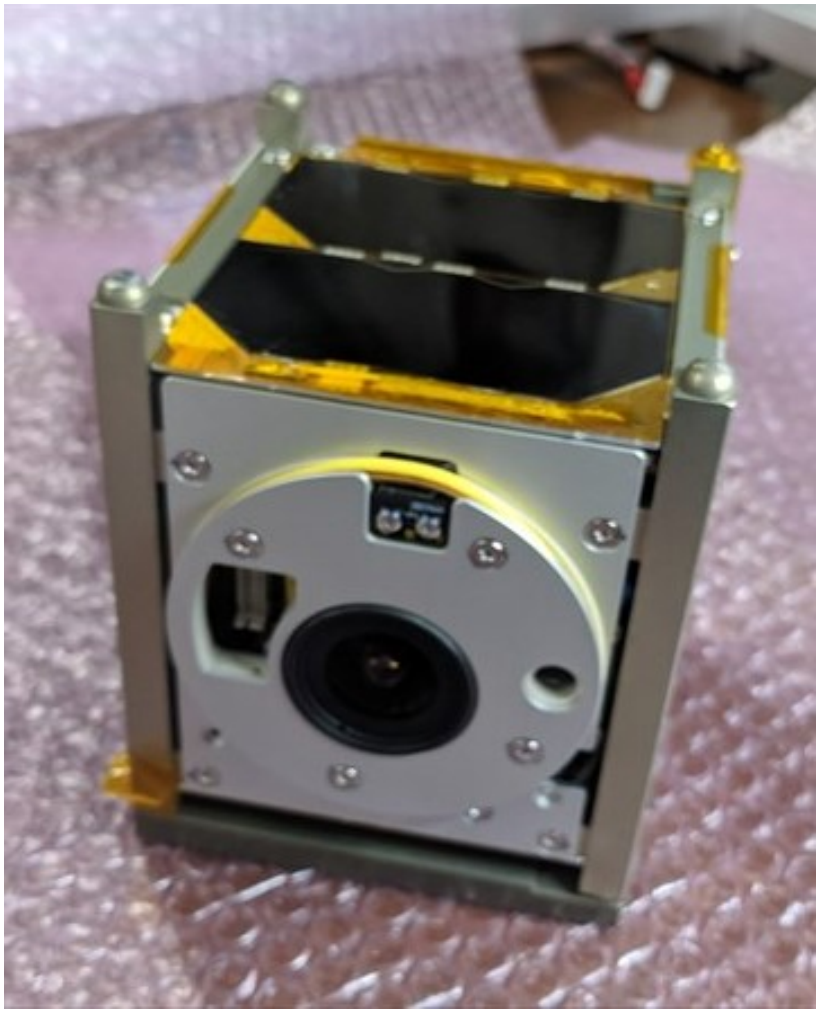
自己紹介

- 個人情報
  - 今村謙之
    - いまむらのりつな
  - JI1SZP
    - 平28.4.22開局
  - 群馬県館林市
- 衛星通信シャック
  - 無線機
    - IC-9100 + UX-9100
    - LimeSDR
  - アンテナ
    - RC5B-3 + ERC5A
    - 1.6m パラボラ

超小型衛星プロジェクト：Stars-AO

SIProp

- 静岡大学：能見研究室
  - 衛星概要
    - 1U
    - 天体撮影
    - 高速通信
  - 打ち上げ（済）
    - 2018/10/29
    - HII-Aロケット
    - 相乗り
    - 600km軌道



http://stars-ao.info/

超小型衛星プロジェクト：rsp-00

SIProp

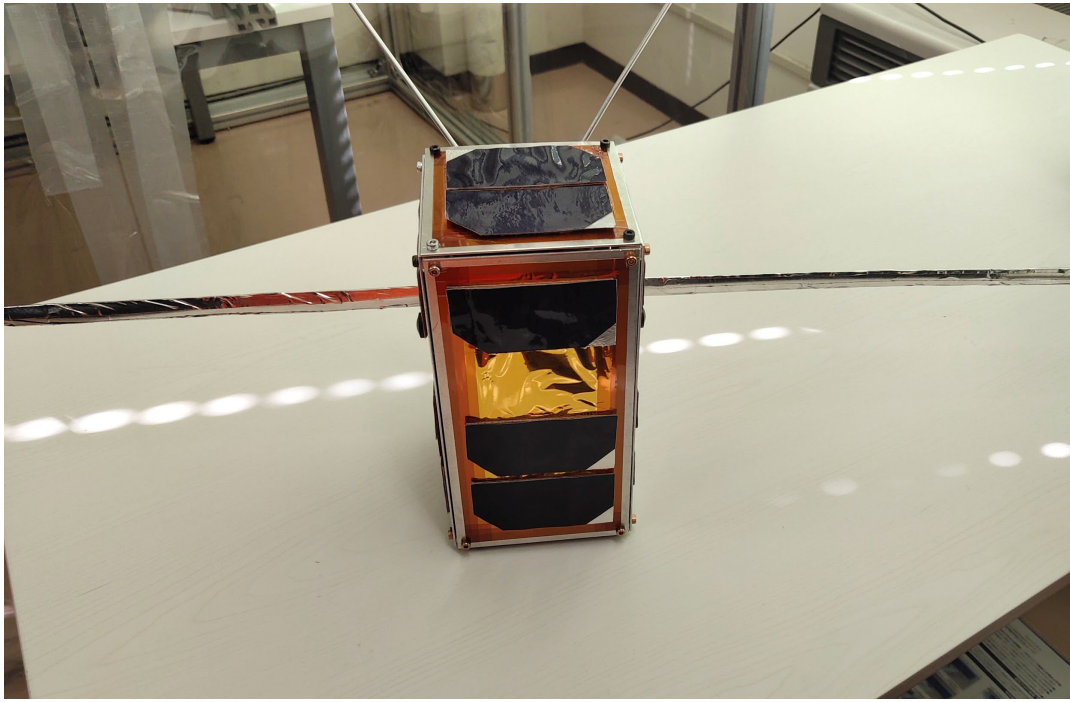
- サラリーマン有志による民間コミュニティー
  - 衛星概要
    - 1U
    - 自撮り
    - デジトーカー
    - QPSK無線機
      - JAMSAT製
  - 打ち上げ（済）
    - 2018/10/6
    - ISS放出

http://www.rymansat.com/

超小型衛星プロジェクト：KOSEN-1

SIProp

- 高専連合による初の超小型衛星プロジェクト
  - 衛星概要
    - 2U
    - 木星電波受信
      - 20MHz帯
    - ラズパイC&DH
  - 打ち上げ（予定）
    - 2021年度
    - 革新2号ロケット

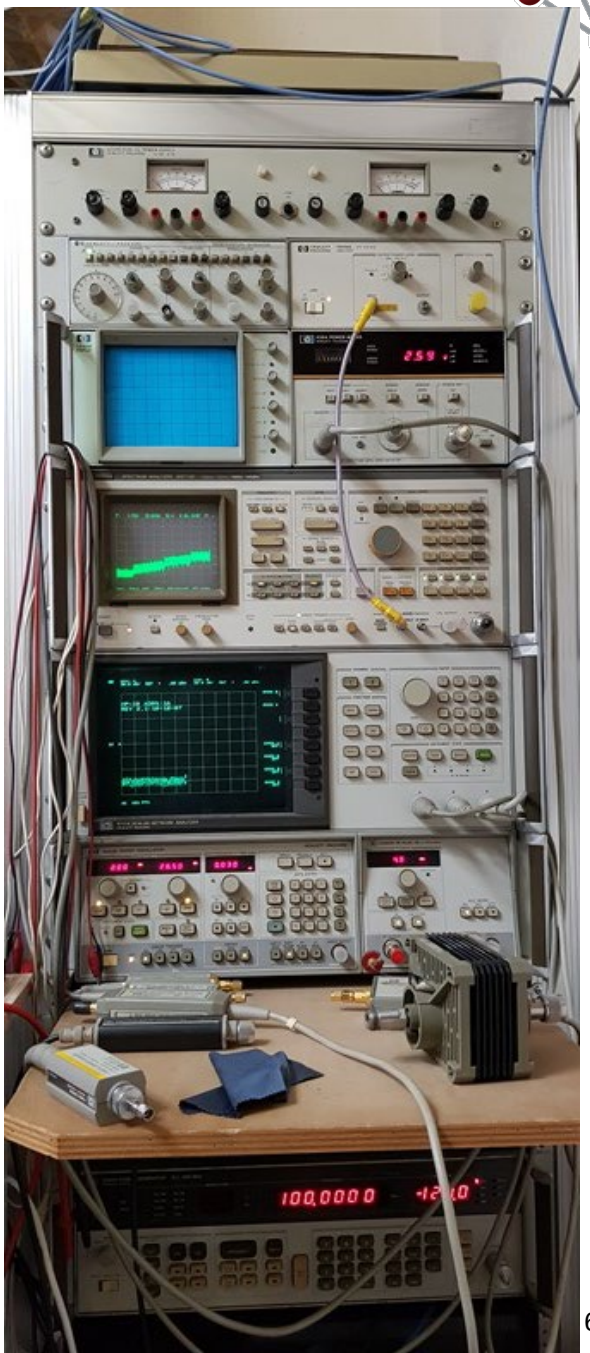


2Uキューブサットによる超高精度姿勢制御・超小型LinuxマイコンボードによるOBC・木星電波アンテナ展開技術の実証	今井 一雅 （高知工業高等専門学校）	市販の超小型Linuxマイコンボードを衛星に利用する提案であり、小型衛星開発の活性化につながると期待。独自のアンテナ展開機構は技術的にも面白い。また、宇宙に高専という新しいプレイヤーが参画することによる裾野拡大と教育・人材育成にも期待。

- http://www.jaxa.jp/press/2018/12/20181212_kakushin_j.html

外部公開計測環境

- HP 6253A Dual DC power supply
- HP 3312A Function Generator
- HP 11975A 2G-8G Amplifier for Spectrum Analyzer Local frequency
- HP 1340A X-Y monitor for HP 8970A noise figure measurement display
- HP 7470A X-Y Plotter for HP 8757A measurement result printing
- HP 436A Microwave Power Meter
- HP 8487A Power Sensor 50MHz-50GHz APC2.4
- Anritsu MS710F modified version as same as MS710E(External mixer ready to 140GHz)
- ANRITSU MH680B Tracking Generator
- HP 8757A Scalar Network Analyzer
- HP 8350B Microwave Seep Generator 2GHz-26.5GHz
- HP 11664E AC Detector 10MHz-26.5GHz
- HP 11664A AC Detector 10MHz-18.0GHz
- HP 85025B AC/DC Detector 10MHz-26.5GHz
- HP 8510C Vector Network Analyzer
- HP 8514A S-Parameter Test Set 500MHz-18GHz
- JDSU JD724C Vector Network Analyzer 5MHz-4GHz
- HP 8970A Automatic Noise Figure Meter
- HP 346C Noise Source 10MHz-26.5GHz
- HP 8586A Signal Generator 100KHz-990MHz
- HP 5351B Frequency Counter 10MHz-26.5GHz



本講演の構成

SIProp

- 2日構成
  - 1日目
    - 講演形式
      - SDRとGNU Radioの概要
      - 最後に明日のハンズオンのための案内アリ
  - 2日目
    - ハンズオン形式
      - SDR+GNU Radioのセットアップと使い方の講習会
        - 全講演が終わりクロージングした後、時間の限り行う
        - 見学のみも歓迎

SIProp

目次:1日目

- なぜ、必要か？
  - ディジタル時代に
  - 無線機の高速化
- 必要な設備解説
  - メイン機器一覧
    - チェックポイント
    - おすすめ機器
  - SDR概要
    - SDRソフトウェア
    - 衛星追尾ソフトウェア
  - GNU Radio概要
- ソフトウェアセットアップ
  - Linux
    - USBメモリへ
    - HDDへ
  - GNU Radio
    - Linux

目次:2日目:ハンズオン

SIProp

- 「アナログの知識のみで使いこなすディジタルソフトウェア無線機」
- 内容
  - NEXUSのデータ受信器(GMSKデータ受信器)を作成する
    - 目的:純粋にSDRのみを使ったディジタル無線
  - AFSK1200送受信器を作成する
    - 目的:Audio入出力を利用して、既存のアマチュア無線機と連携させる
  - GR-satellitesをビルドして、GNURadioに登録する
    - 目的:ネットに一杯存在しているGNURadioのレシピを自力で使えるようにする
      - GR-satellites: https://github.com/daniestevez/gr-satellites

SIProp

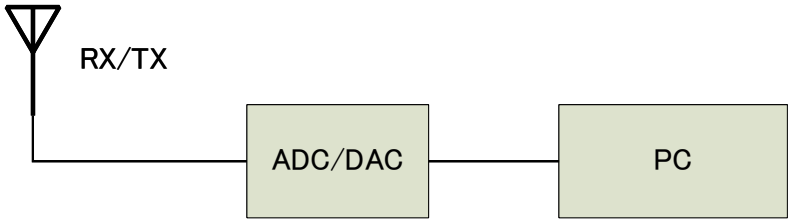
SIProp

- SDRとGNU Radioの概要

SDRとはなにか？

- Software Defined Radio
  ＝ソフトウェアで定義された無線機

- 一番簡略化した構造
  - 構成物
    - アンテナ
    - ADC/DAC
    - 処理器(DSP=Digital Signal Processor)
      - ソフトウェアで処理を定義できる信号処理に特化したプロセッサ
        - PCで代用可能！

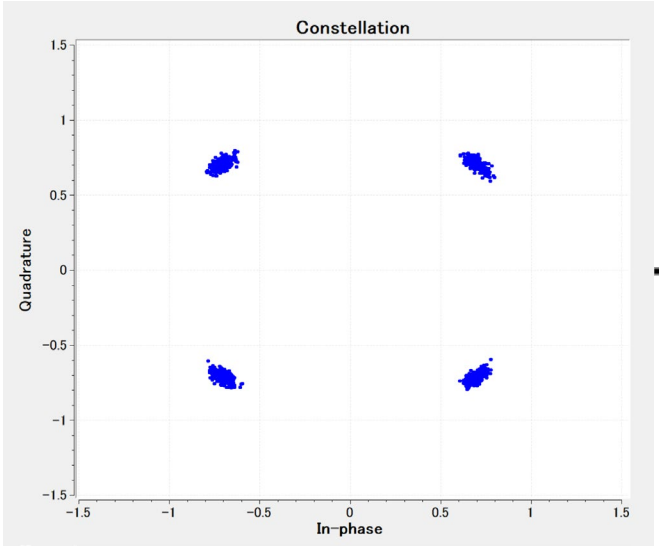


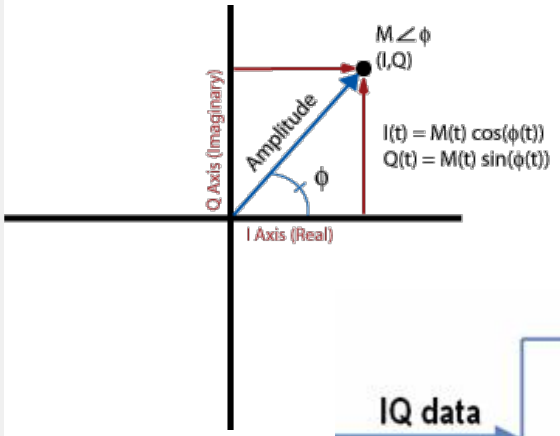
  - 回路で処理しているものをDSPで処理しているだけ！
    - 電波をDSPで処理できる形にする変換器≒SDR

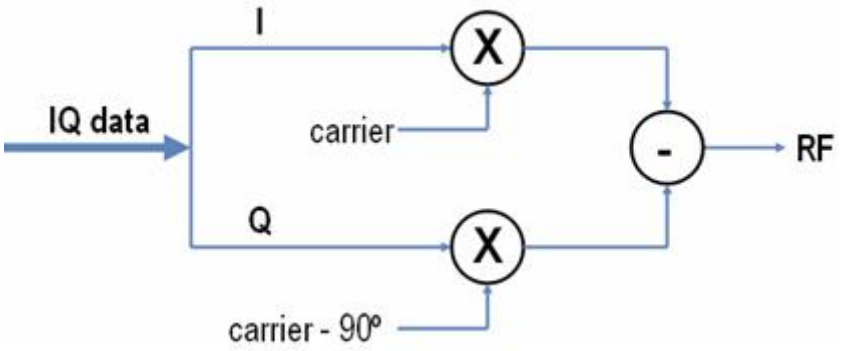
IQ(変調)だけ、理解する

SIProp

- ディジタル通信＝データ通信
  - それを実現しているのがIQ変調
    - 実は、GNURadioを使う上では使わない・・・







$$A_c \cos(2\pi f_c t + \phi)$$

Amplitude　Frequency　Phase

Angle

(Frequency = Rate of Change of Angle)

出典：http://www.ni.com/tutorial/4805/ja/

なぜ、SDRを利用するのか？

SIProp

- 無線はデジタルデータの時代へ
  - 商用無線
    - 地デジ
    - LTEや5GのVoIP化
  - 地上のアマチュア無線
    - JT65
  - アマチュア衛星
    - GMSK9600以上のデータ通信の高速化
- 市販の（アマチュア）無線機では受信できない
  - こともないが、PCなどと連携が必要

しかし・・・

- SDRの弱点
  - 無線機としての出来が悪い
    - 感度が低い
  - 免許状の申請が大変
    - 申請前例がないSDRが大半
    - SDRの利点を殺して、特定電波で申請する必要がある
  - 最強の組み合わせは実は・・・
    - 市販のアマチュア無線設備＋（SDR＋）GNURadio
      - GNURadioの出力をAudioやIFとして無線機に食わせる
        - この使い方は、演習の「3」でやります

- SDRの基本と必要機材

市販のSDR一覧

SIProp

Comparison Table

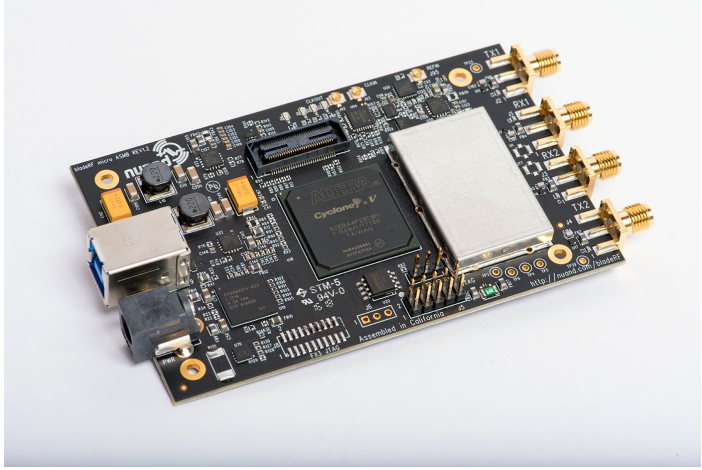
	HackRF One	Ettus B200	Ettus B210	BladeRF x40	RTL-SDR	LimeSDR	LimeSDR Mini
Frequency Range	1 MHz - 6 GHz	70 MHz - 6 GHz	70 MHz - 6 GHz	300 MHz - 3.8 GHz	22 MHz - 2.2 GHz	100 kHz - 3.8 GHz	10 MHz - 3.5 GHz
RF Bandwidth	20 MHz	61.44 MHz	61.44 MHz	40 MHz	3.2 MHz	61.44 MHz	30.72 MHz
Sample Depth	8 bit	12 bit	12 bit	12 bit	8 bit	12 bit	12 bit
Sample Rate	20 MSPS	61.44 MSPS	61.44 MSPS	40 MSPS	3.2 MSPS	61.44 MSPS	30.72MSPS
TX Channels	1	1	2	1	0	2	1
RX Channels	1	1	2	1	1	2	1
Duplex	Half	Full	Full	Full	N/A	Full	Full
Interface	USB 2.0	USB 3.0	USB 3.0	USB 3.0	USB 2.0	USB 3.0	USB 3.0
Programmable Logic Gates	64 macrocell CPLD	75k	100k	40k (115k avail)	N/A	40k	16K
Chipset	MAX5864, MAX2837, RFFC5072	AD9364	AD9361	LMS6002M	RTL2832U	LMS7002M	LMS7002M
Open Source	Full	Schematic, Firmware	Schematic, Firmware	Schematic, Firmware	No	Full	Full
Oscillator Precision	+/- 20 ppm	+/- 2 ppm	+/- 2 ppm	+/- 1 ppm	?	+/-1 ppm initial, +/-4 ppm stable	+/- 1 ppm initial, +/- 4 ppm stable
Transmit Power	-10 dBm+ (15 dBm @ 2.4 GHz)	10 dBm+	10 dBm+	6 dBm	N/A	max 10 dBm (depending on freq.)	max 10 dBm (depending on freq.)
Price	$299	$686	$1,119	$420 ($650)	~$10	$299	$159

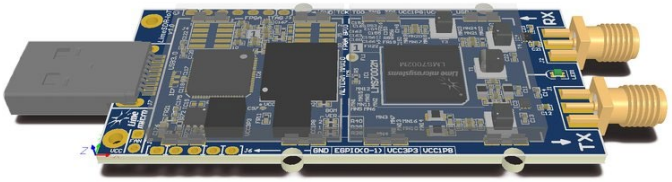
- LimeSDRのサイトより転載

- Freq Range
  - 短波非対応が多い
  - 6GHzがMax
- Sample Depth
  - 12bit欲しい
- 価格
  - 1万台～
- 対応ソフト
  - GNURadio
  - MATLAB

おすすめ機器

- 対応周波数
  - 5GHz以上
    - BladeRF2
- 対応ソフトウェア
  - MATLAB対応
    - PlutoSDR
- サイズと価格
  - ドングルサイズ
    - LimeSDR mini



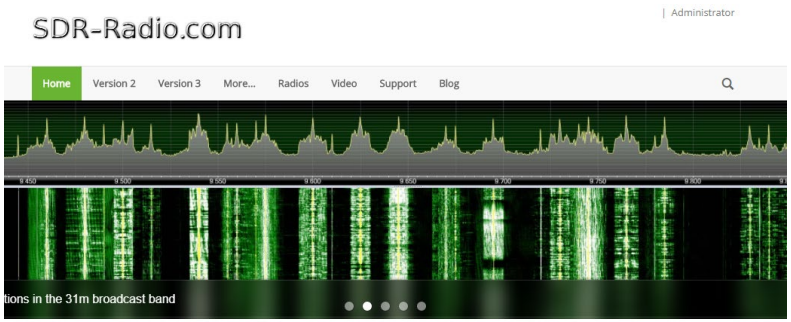


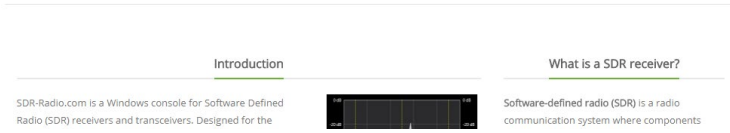
- SDRを使う
  - アナログ受信

SDR用のアナログ受信ソフトウェア

SIProp

- SDR-Radio V3
  - http://sdr-radio.com/
- 対応環境
  - Windows 7 or higher (32bit & 64bit)

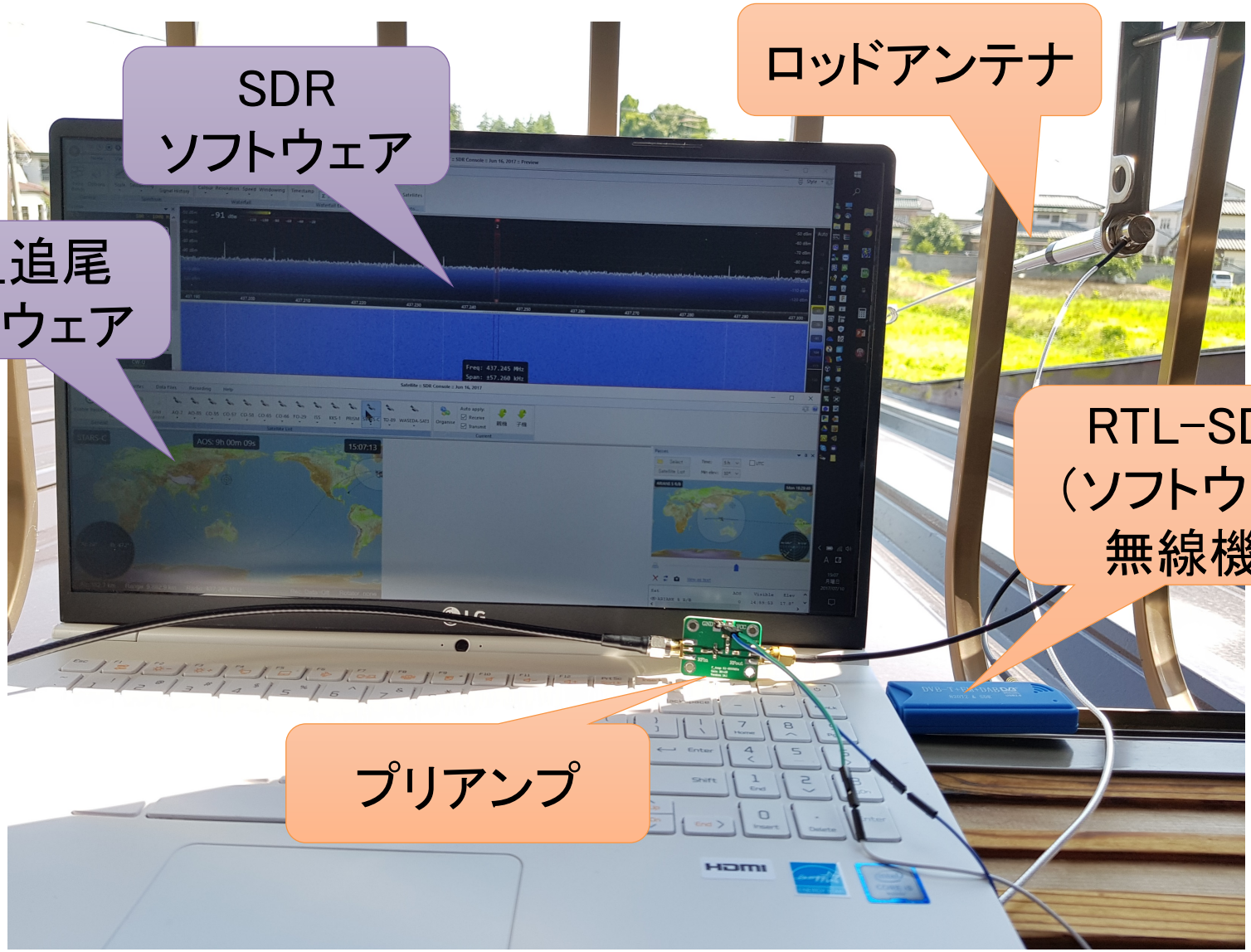




  - 5000円で誰でも作れる新世代衛星地上局
    - https://www.slideshare.net/noritsuna/5000-77691748

SDRを地上局として使う

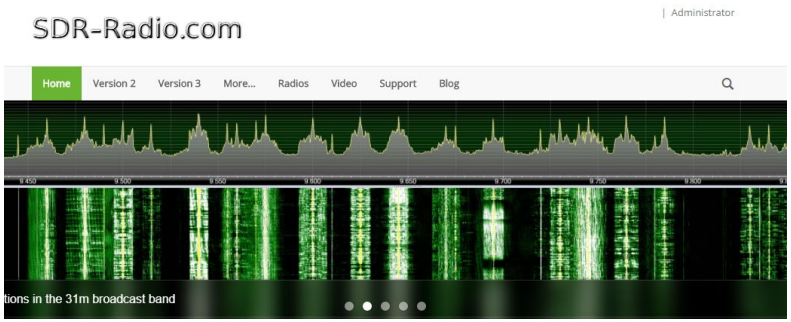
SIProp

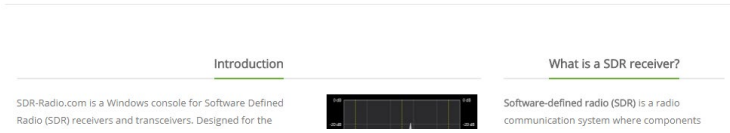


SDR用の衛星追尾ソフトウェア

SIProp

- SDR-Radio V3
  - http://sdr-radio.com/
- 対応環境
  - Windows 7 or higher (32bit & 64bit)



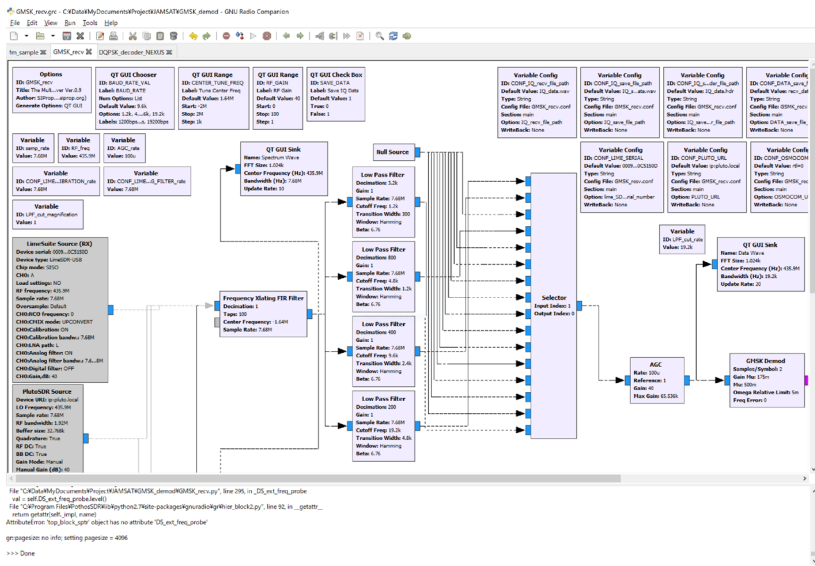


  - 5000円で誰でも作れる新世代衛星地上局
    - https://www.slideshare.net/noritsuna/5000-77691748

- SDRを使う
  - ディジタル送信
  - ディジタル受信
  - アナログ送信
  - アナログ受信

SDR用のディジタル送受信ソフトウェア

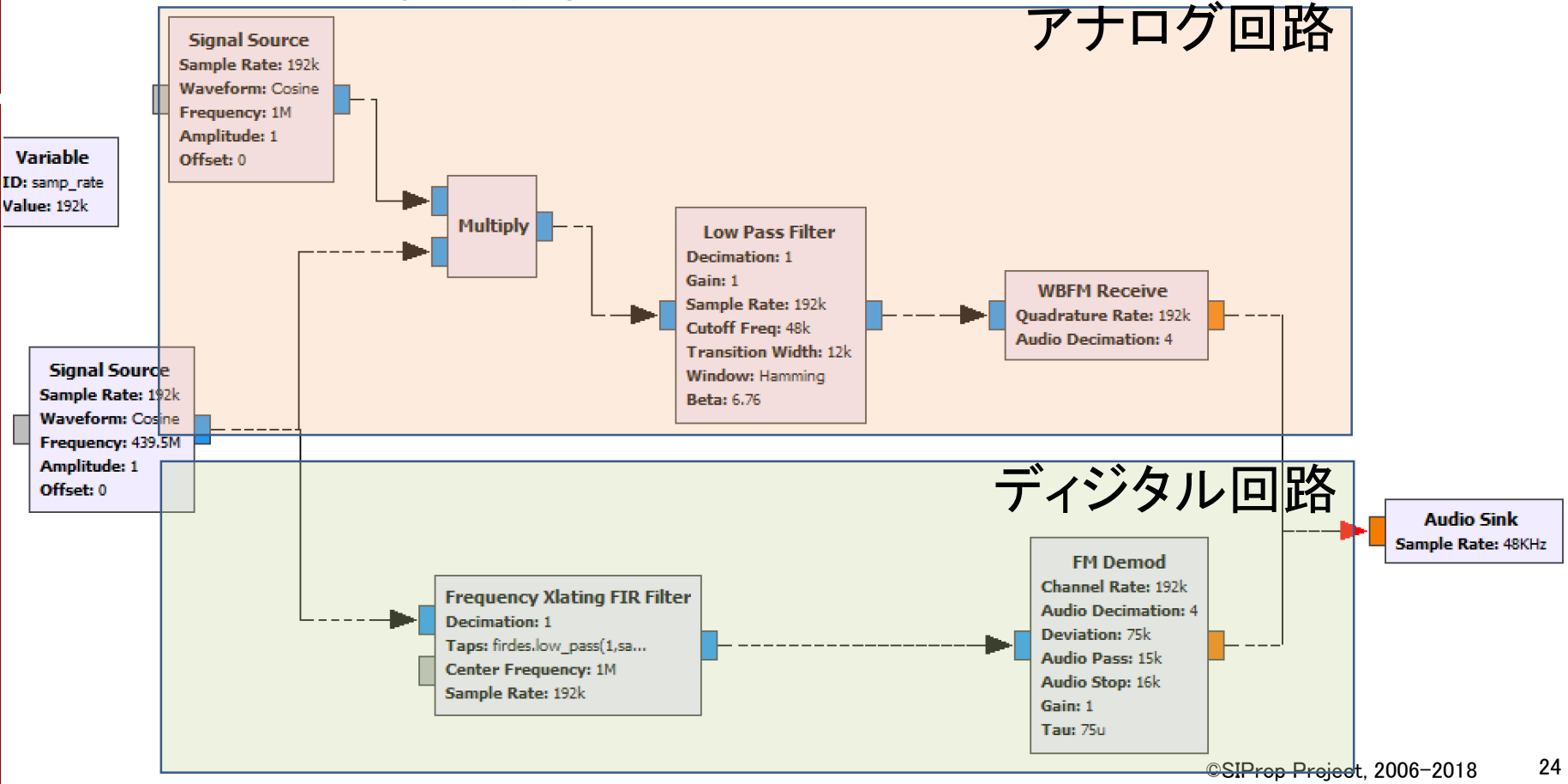
- GNURadio
  - オープンソースのソフトウェア信号処理器
    - フロー・ダイヤグラムによる実装が可能
      - ディジタル処理を行うためのソフトウェア
        - 同様の機能を持つ商用のソフトウェア信号処理器
          - MATLABのSimulink
          - LabVIEWのVI
  - 動作環境
    - Windows and Linux
      - https://www.gnuradio.org/
  - GRC(GNURadio-companion)
    - フローダイヤグラムエディタ
      - ブロック
        - 信号処理を行う単位
      - フロー
        - 信号の伝達ライン



ディジタルは難しい？

SIProp

- 必要な知識
  - アナログ無線
  - 無線用語の英語



GNURadioを使う利点

SIProp

- 大体のやりたいこと(変調回路)がインターネット上に存在している
  - JT65(中身はない？)
    - https://github.com/TheWylieStCoyote/gr-JT65
  - QPSK
    - https://wiki.gnuradio.org/index.php/Guided_Tutorial_PSK_Demodulation
  - DATV
    - https://github.com/csete/gr-datvexpress
  - 拾ってきて、ちょっといじるだけいろいろできる！

SIProp

SIProp

- チラ見せGNURadio演習
  - 明日の演習をちょっとだけお見せします

コンセプト

SIProp

- ディジタルの知識不要
  - IQの知識さえ、不要！
- プログラミングの知識不要
  - 必要な部分のみ、きちんと解説します
    - intなどのプリミティブ型とは？
    - Pythonとは何か？
    - git, configure, makeとは何か？
- Linux/Ubuntuの知識不要
  - 必要なコマンドなどは、きちんと解説します
    - Linuxコマンドの基本
- 最終習得知識
  - インターネットにあるGNURadioレシピを改造して、使いこなせるようにする！

進め方

- 演習ごとに、未完成のGRCファイルを用意
  - こちらの指示に従って、完成させていくという形
    - ブロックを追加
    - パラメータの設定
    - 結線

デモンストレーションタイム

SIProp

- 本日のデモンストレーション
  - SDRを使ったデモンストレーション
    - AFSK(APRS)受信デモ
    - SDRソフトウェアとの連携
  - Linuxのセットアップ実演
    - Windows領域に空き領域を作る
    - ちょっとミスすると起動しないデモ

SIProp

次世代アマチュア衛星受信のスタンダード「SDR+GNU Radio」演習

今村謙之(Noritsuna Imamura)
JI1SZP
noritsuna@siprop.org

×SIProp

はじめに

- 基本的なものこれを読もう！
  - RFワールド No.44
    - 「GRCで広がるSDRの世界」
  - 本演習はこの本でさえ躓いてしまった人を救います！

SIProp

SIProp

- GNURadio演習

コンセプト

SIProp

- ディジタルの知識不要
  - IQの知識さえ、不要！
- プログラミングの知識不要
  - 必要な部分のみ、きちんと解説します
    - intなどのプリミティブ型とは？
    - Pythonとは何か？
    - git, configure, makeとは何か？
- Linux/Ubuntuの知識不要
  - 必要なコマンドなどは、きちんと解説します
    - Linuxコマンドの基本
- 最終習得知識
  - インターネットにあるGNURadioレシピを改造して、使いこなせるようにする！

- SDRの基本と必要機材

必要機材

- パソコン
  - CPU: 第四世代以降Core 3-9i、2コア以上(4コア推奨)
    - 最新のパソコンを推奨
- USBメモリ or USB-HDD or USB-SSD
  - USB3.0以上推奨
  - 32GB以上(大容量のUSB-HDD推奨)
- SDR(送信までしたい場合)
  - Lime SDR mini
    - https://www.crowdsupply.com/lime-micro/limesdr-mini
  - Pluto SDR
    - https://www.analog.com/jp/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html

Linuxインストール方法（検討）

SIProp

- 今あるWindowsの領域を縮小させて、そこにLinuxをインストールして、DualBootとする
  - 一般的なLinuxのインストール方法
    - 問題点：Linuxを消すことはできなく、デフォルトだとLinuxが起動するモードとなる
      - OSに関する知識があれば、変更や削除可能
- USBメモリにLinuxをインストールする
  - USBメモリを引き抜けば、Linuxの痕跡をすべて消せる
    - 問題点：セットアップ時にミスをするとWindowsが起動しなくなる
      - 修復にはOSに関する知識が必要
- Live USBメモリのLinuxを利用する
  - USBメモリを引き抜けば、Linuxの痕跡をすべて消せる
    - 問題点：データの保存領域が4GBしか作れない
      - そのため、本格的な利用は不可能

インストール環境

SIProp

- OS
  - Ubuntu 18.04 LTS 日本語Remix
    - LTS版Ubuntuの現時点（2019年）での最新版
      - https://www.ubuntulinux.jp/japanese
- インストール方法
  - UNetBootin
    - USBメモリにLive Linuxをセットアップするためのツール
      - https://unetbootin.github.io/
    - これを使って、USBメモリやUSB-HDDにLinuxをセットアップする

Linuxのセットアップ

USBメモリ用ddイメージ入手先

- https://www.noritsuna.jp/download/grc_ubuntu.zip

Linuxとは？

SIProp

- Linux
  - Unixを参考にして作られたOS（POSIX非準拠）
- Linuxの種類
  - ディストリビューション
    - 各種コマンドやアプリケーションをパッケージ化したもの
      - Linuxとは本来Kernel(OSのコア)のみを指す
  - Ubuntu 18.04 LTS 日本語Remix
    - Ubuntu
      - Debian系ディストリビューションで一番使われている
    - 18.04
      - 前半がリリース年、後半はリリース月
    - LTS(Long Term Support)
      - 10年間のサポートをする長期利用可能なリリース
        - 偶数年の4月にリリースされる
        - これ以外は9か月

Linuxのセットアップ：PCの確認

SIProp

- ブートシステム（ファームウェア）
  - UEFI
    - Windows8(2012年)で採用されたためそれ以降のPCは個のブートシステムとなっている
    - 問題点
      - セキュリティーが強化されているため、いろいろとオプションを変えないとUSBメモリブートできない可能性がある
      - システムによって設定法が大きく違う
  - BIOS
    - UEFIの前のブートシステム
    - 問題点
      - USBメモリブートに対応していない可能性がある

UTC(協定世界時)に変更する

SIProp

- Ubuntuは、UTCで時刻管理をしているため、UbuntuからWindows(JSTでの管理)に戻ってくると9時間(JST分)時間がずれてしまう
  - WindowsをUTCに変更する
    - ロケール管理により、表示時間は自動的にUTC+9時間（JST分）されるため、使い勝手に影響はない

```
reg add
"HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥
Control¥TimeZoneInformation" /v RealTimeIsUniversal /d
1 /t REG_DWORD /f
```

詳しくは・・・

- これを読もう！
  - RFワールド No.44
    - 「GRCで広がるSDRの世界」

- GNURadioのセットアップ

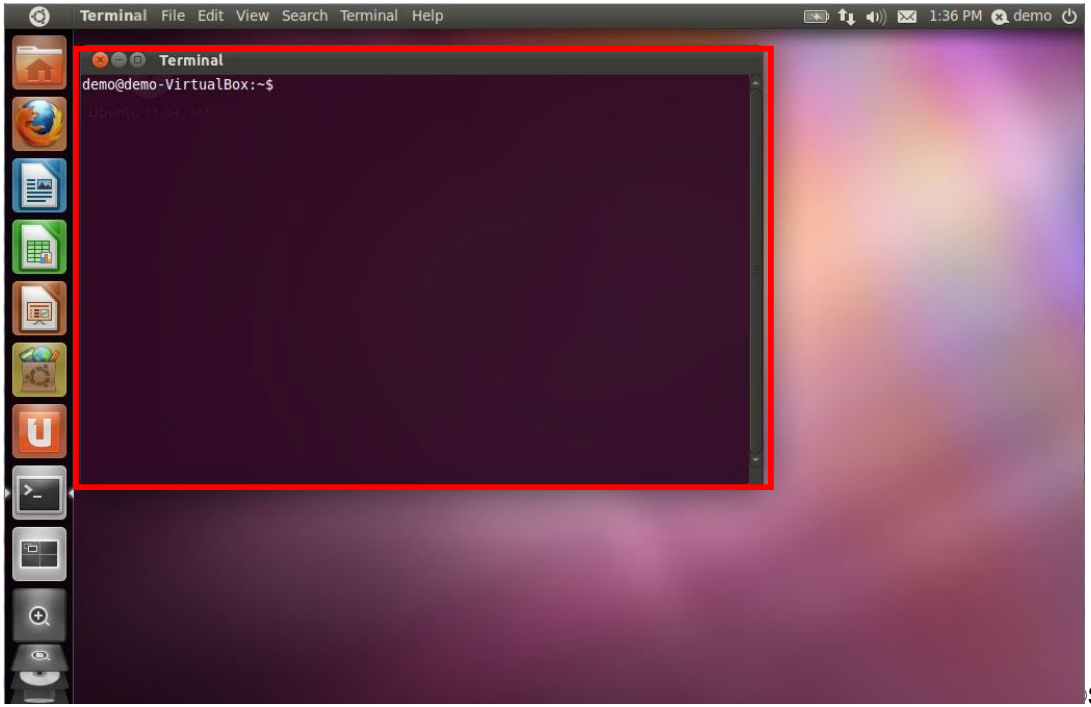
Linuxの操作方法：GUI

SIProp

- Windowsとほぼ同じ
  - アプリの起動方法
    - メニューからアイコンをクリックして起動する
  - アプリの操作方法
    - Windowsが開くので、マウスとキーボードで操作する

Linuxの操作方法：CUI

- コマンドプロンプト
  - 起動方法
    - 「端末(Terminal)」というアプリ
  - 操作方法
    - 「Linuxコマンド」を入力することで操作を行う



CUIにおける重要な概念

SIProp

- ディレクトリ
  - ルートディレクトリ
    - /
      - Windowsの「C:¥」に相当するもの
  - ホームディレクトリ
    - /home/[ユーザID]
      - Windowsの「PC」に相当するもの
      - ファイルなどはここに保存すること
- パーミッション(permission)
  - ユーザ権限
    - LinuxはUnix系OSであるため権限がきちんと決められている
      - コマンド実行時にpermission系のエラーが出た場合
        - sudo コマンドを付けて実行すること

Linuxコマンド（使うものだけ抜粋）

SIProp

SIProp

- 基本コマンド
  - コピー
    - cp [コピー元] [コピー先]
      - cp -ar * ../docs/
        - 再帰的に全て処理
  - ディレクトリ変更
    - cd [移動先]
      - cd ../
        - 一つ上に戻る
  - ディレクトリ作成
    - mkdir [ディレクトリ名]
  - ディレクトリ表示
    - ls
      - ls -la
        - 詳細表示
- 拡張コマンド
  - ソフトインストール
    - apt install [ソフト名]
  - スーパーユーザ実行
    - sudo [コマンド]
- 開発コマンド
  - ダウンロード
    - git clone [git URL]
  - コンフィグ
    - ./configure
  - ビルド
    - make
  - インストール
    - sudo make install

```
user@pc:~ $ sudo raspi-config
```

GNURadioのインストール

SIProp

- インストールするもの
  - GNURadio本体
    - デフォルトのものはバージョンが古い
      - 最新版を使いたい場合は、RFワールド No.44を参照
  - PlutoSDR用ドライバ
  - RTL-SDR用ドライバ
  - 開発環境

```
user@pc:~ $ sudo apt update
user@pc:~ $ sudo apt upgrade
user@pc:~ $ sudo apt install gnuradio
user@pc:~ $ sudo apt install gr-iio
user@pc:~ $ sudo apt install rtl-sdr gr-osmosdr
user@pc:~ $ sudo apt install git cmake build-essential
```

SIProp

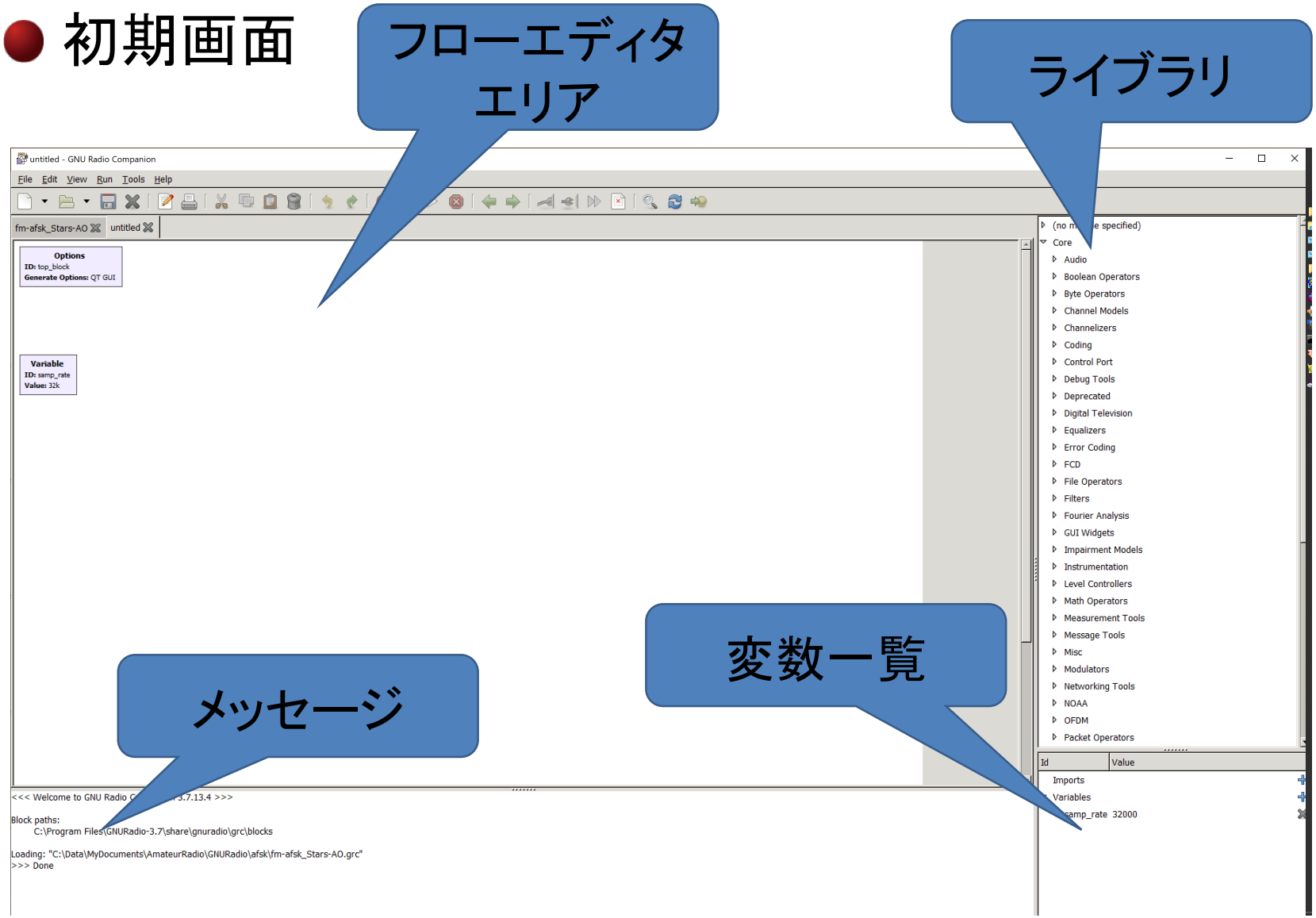
SIProp

- GNURadioの基本操作をマスターする

GNURadioの起動と画面の説明

SIProp

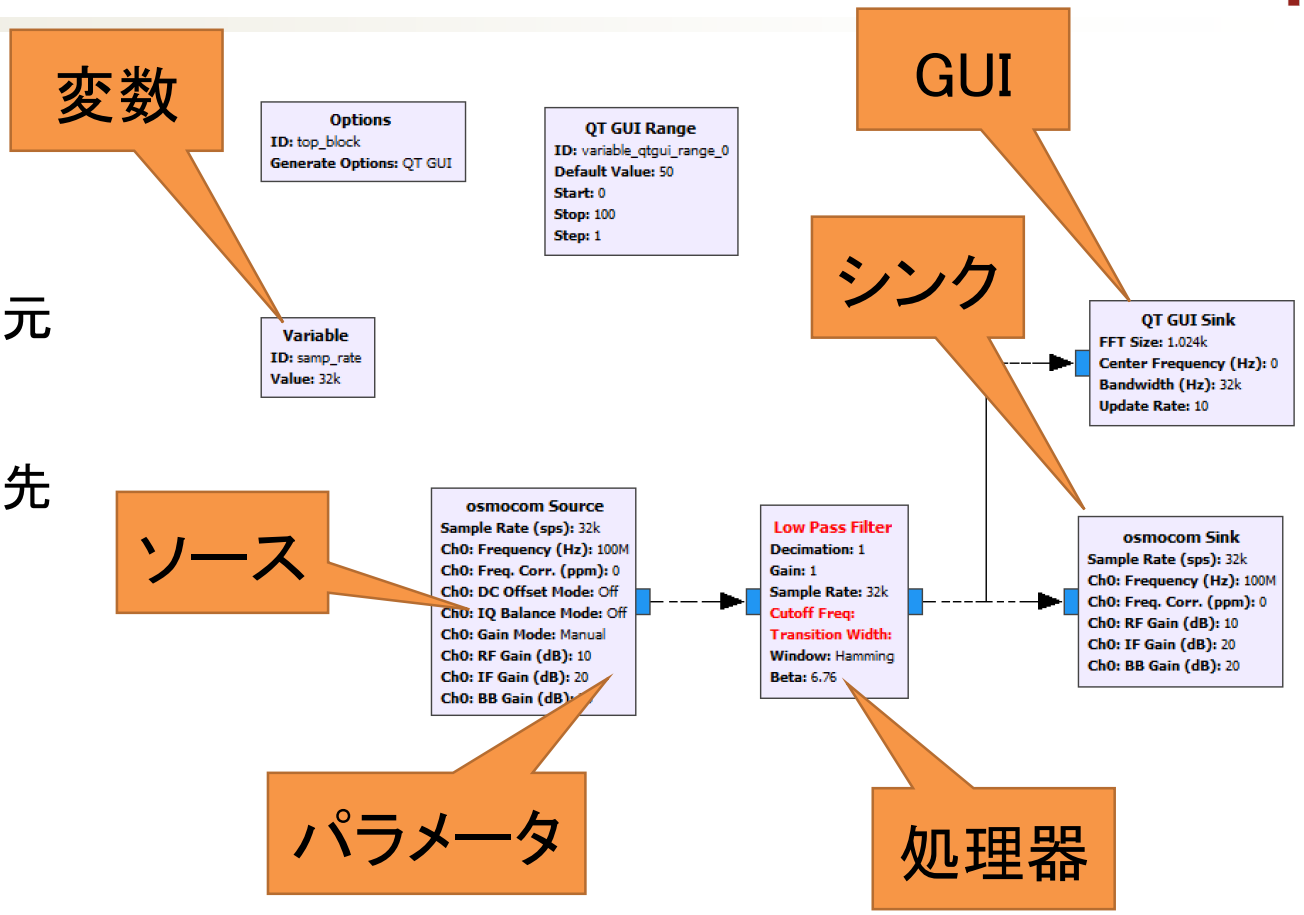
- 初期画面



ブロックの基本の解説

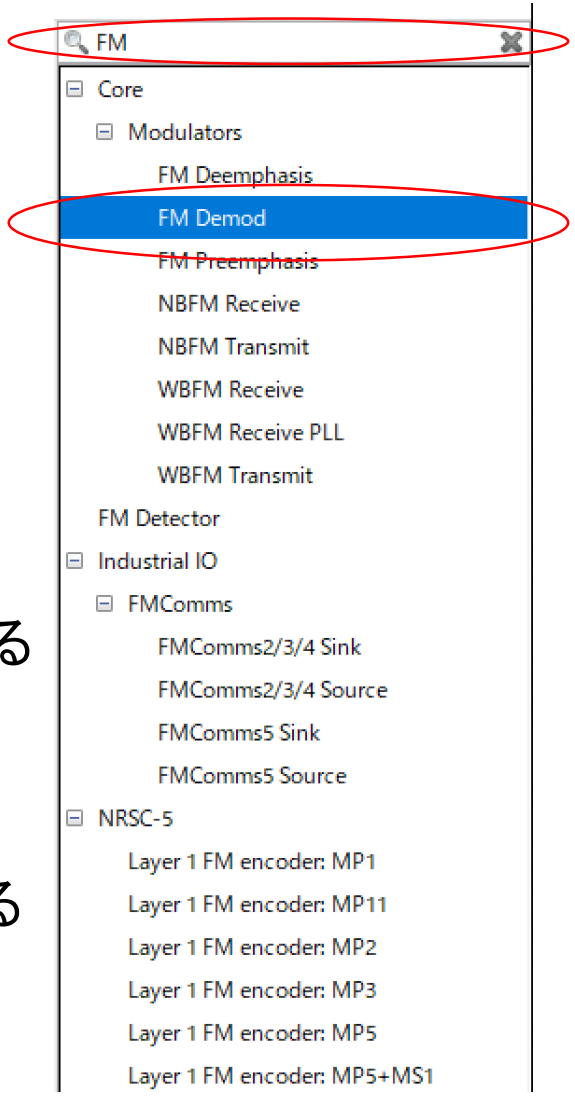
SIProp

- 分類
  - ソース
    - 入力元
  - シンク
    - 出力先
  - 処理器
  - 変数
  - GUI
- パラメータ
  - ブロックの処理の内容を決定する



ブロックの操作 1/2

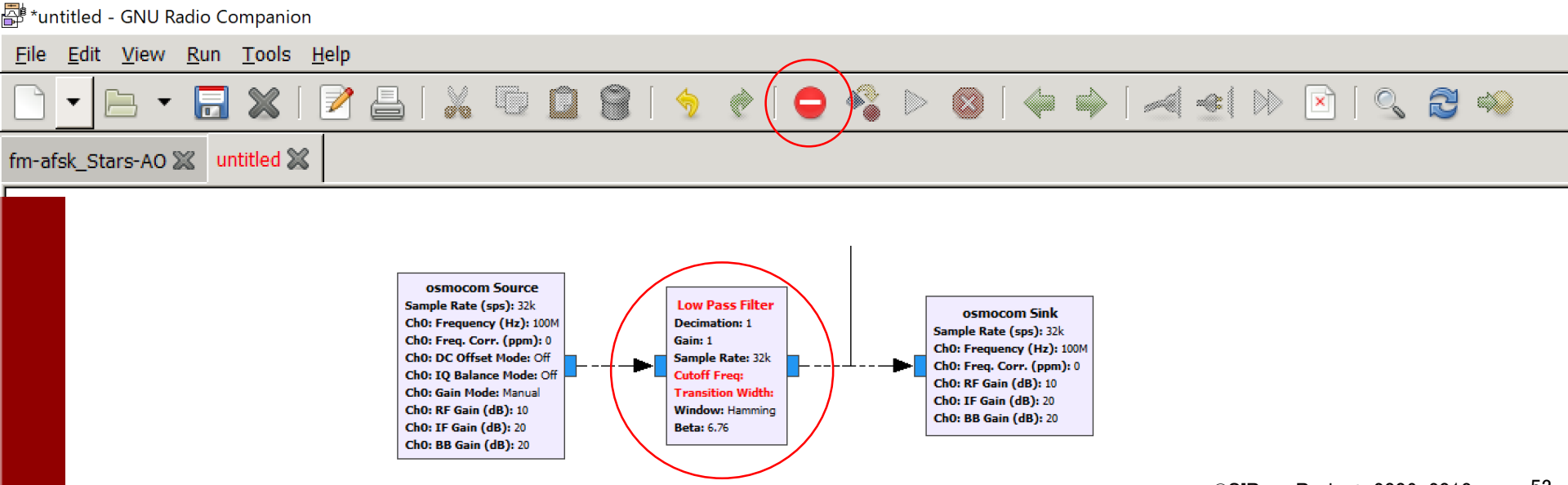
- 検索
  1. Ctrl + Fキーを押す
  2. ライブラリウィンドに検索エリアが出る
  3. 利用したいブロック名を入れる
  4. 候補が表示される
- 追加
  1. 利用したいブロックをダブルクリックする
  2. フローエディタエリアにブロックが出現する
- 選択・変更
  1. マウスでクリックすると選択できる
  2. ダブルクリックするとオプション変更できる
- 移動
  1. マウスでドラッグ＆ドロップで移動可能



ブロックの操作 2/2

SIProp

- エラーの修正
  1. 赤字部分がある場合、エラーがあるということである
  2. ブロックやフローをクリックして、選択する
  3. メニューの「エラー表示」アイコン（一時停止マーク）をクリックする
  4. エラーメッセージウィンドウに表示されている内容に合わせて修正する

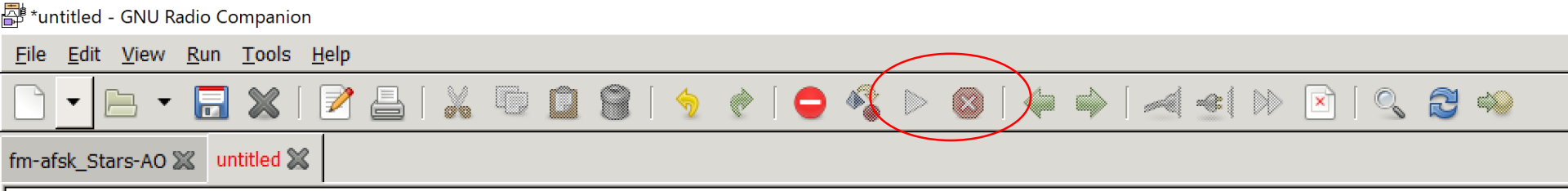


実行する

SIProp

- 実行
  1. 「Run」アイコン（▷マーク）をクリックする
  2. 自動で「generate」されて、実行される

- 停止
  1. 「Stop」アイコン（×マーク）をクリックする
  2. 終了する



SIProp

保存する

SIProp

- 保存
  1. 「Save」アイコン（フロッピーディスクマーク）をクリックする
  2. 保存先のディレクトリとファイル名を付けて、OKする
- 読み込み
  1. 「Open」アイコン（フォルダマーク）をクリックする
  2. 読み込みしたいファイルのディレクトリとファイル名を選択する
- 新規作成
  1. 「New」アイコン（白紙マーク）をクリックする

GNURadioの演習

演習用ファイル入手先

- https://www.noritsuna.jp/download/grc_study.zip

演習1：波形の生成と出力

SIProp

- 100Hzと150Hzのコサイン波を生成して、波形表示を行う

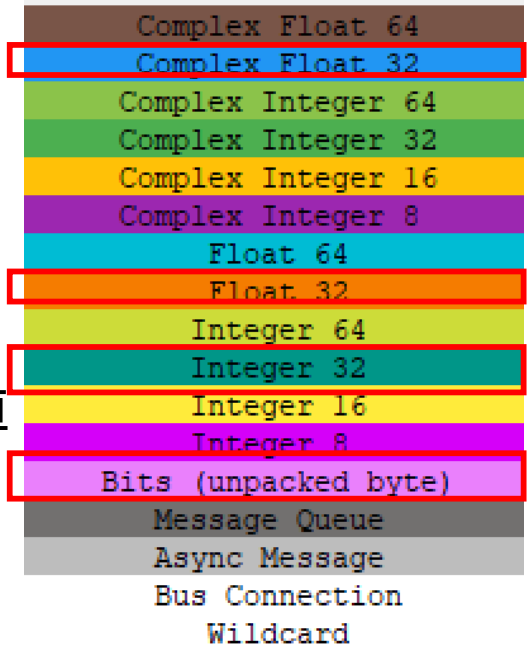
- 目的
  - 一番スタンダードな使い方をマスターする

  - 内容
    - ブロック
    - 接続
      - 型の理解
    - 生成
    - 実行
  - 演習用GRC

接続の方法と種類

SIProp

- 基本型（必要なもののみ抜粋）
  - Complex(複素数＝IQ)型
    - 波形のこと
  - プログラム型
    - 波形を信号処理した人間が理解できる数値
      - Float（小数点）
      - Integer（整数）
        - 32bit(約-21億～約21億)
  - Bit(データ)型
    - 波形を信号処理し、摘出されたデータ
      - パケット
- 配列（型の集合）
  - スカラー（1個）
  - ベクター(2個以上)



- 接続
  - n対1
  - 1対n

演習1-1：波形の生成と出力

SIProp

- コサイン波を生成して、波形表示を行う
  - 基本レシピを作る
    1. 「Signal Source」ブロックを配置する
       1. Frequencyを「100」に変更する
    2. 「Qt Time Sink」ブロックを配置する
    3. 接続する

SIProp

演習1-1：波形の生成と出力

SIProp

- コサイン波を生成して、波形表示を行う
  - 基本レシピを作る
    1. 「Signal Source」ブロックを配置する
       1. Frequencyを「100」に変更する
    2. 「Qt Time Sink」ブロックを配置する
    3. 接続する

SIProp

演習1-2：波形の生成と出力

SIProp

- コサイン波を生成して、波形表示を行う
  - 波形を一つだけにする
    1. 「Signal Source」ブロックの出力を変更する
       1. Complex -> Float

SIProp

演習1-3：波形の生成と出力

SIProp

- コサイン波を生成して、波形表示を行う
  - 違う波形も表示する
    1. 「Signal Source」ブロックを追加する
       1. Frequencyを「150」にする
    2. 「Qt Time Sink」ブロックの「Input of Number」を「2」にする
    3. 接続する

SIProp

演習2：SDRを接続し、各種表示を行う

- SDRからの出力を、FFT、ウォーターフォール表示させる

- 目的
  - SDRの使い方をマスターする

  - 内容
    - 周波数設定
    - サンプリングレート
    - ゲイン
    - 各種GUIツール

演習2-1：SDRを接続し、各種表示を行う

- SDRからの出力を、FFT、ウォーターフォール表示させる
  - SDR Sourceの出力を表示する
    1. 「Pluto SDR Source」ブロックを追加する
       1. Device URLを「ip:pluto.local」とする
       2. LO Frequencyを「438.2e6」とする
          1. e6=10^6(10の6乗)
       3. Sampling Rateを「samp_rate」とする
       4. RF Bandwidthを「samp_rate」とする
    2. 「Variable」ブロックのID：「samp_rate」を変更する
       1. Valueを「2084000」とする
    3. 「Qt GUI Sink」ブロックを追加する
       1. Center Frequencyを「438.2e6」とする
       2. Bandwidthを「samp_rate」とする
    4. 接続する
       1. LO Frequencyを「int(438.2e6)」とする

演習2-2：SDRを接続し、各種表示を行う

- SDRからの出力を、FFT、ウォーターフォール表示させる
  - 先程のファイルをそのまま使います
    - SDR Sinkに入力したものを表示する
      1. 「Pluto SDR Sink」ブロックを追加する
         1. Device URLを「ip:pluto.local」とする
         2. LO Frequencyを「int(438.2e6)」とする
            1. e6=10^6(10の6乗)
         3. Sampling Rateを「samp_rate」とする
         4. RF Bandwidthを「samp_rate」とする
      2. 「Signal Source」ブロックを追加する
      3. 接続する
      4. TXとRXを直結するか、ダミーローダーを接続する

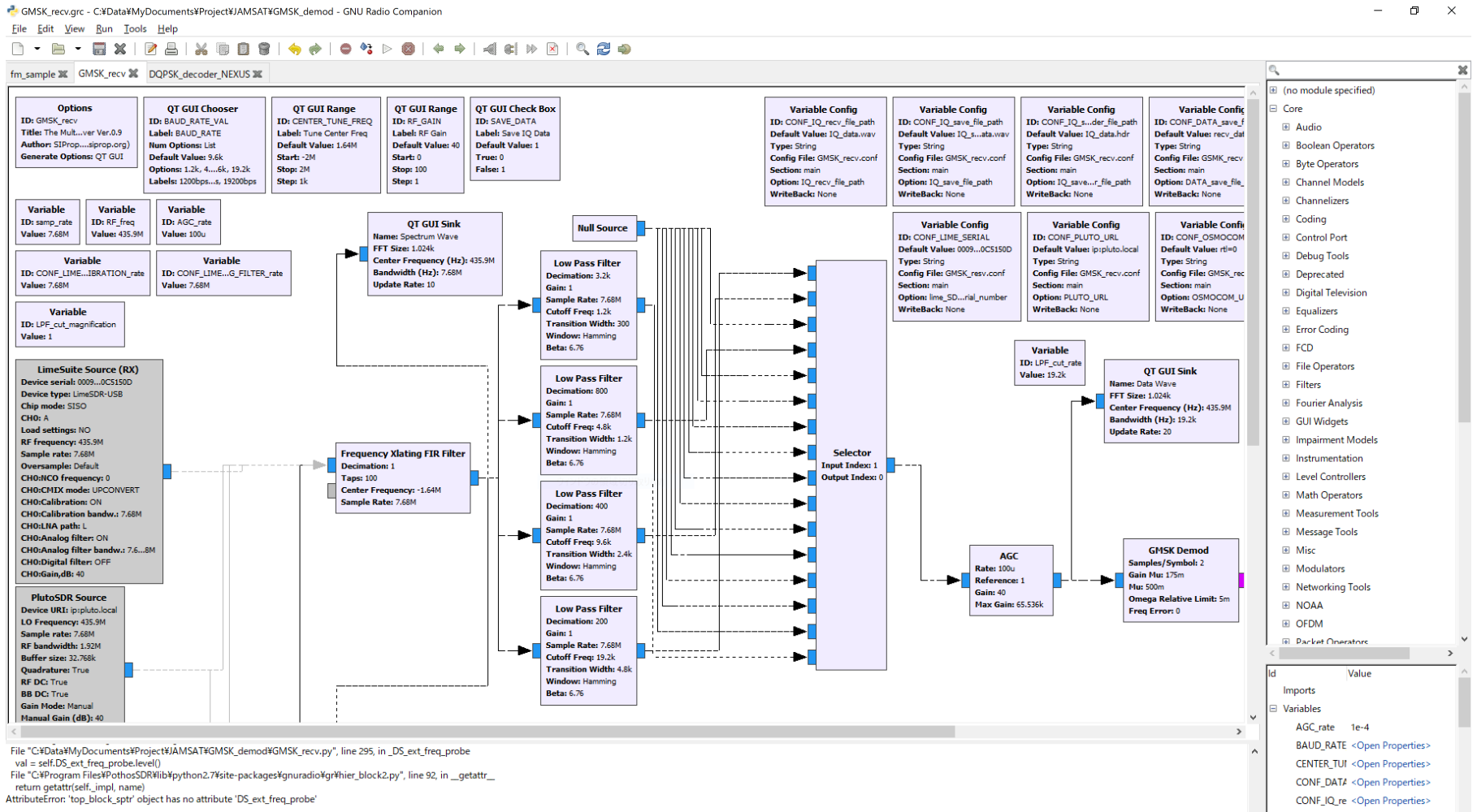
！！！注意：電波が出来ます。ダミーローダの接続必須！！！

- FO-99(NEXUS)のデータ受信器(GMSKデータ受信器)を作成する

FO-99(NEXUS)用のGMSK受信GRC

SIProp

- 完全版は非常に複雑
  - 簡易版を作りながら、GMSK(データ)通信の仕方を学ぶ



演習3:リアルタイム変更する

- 中心周波数を変更しながら、表示を行う
- 目的
  - 周波数変更など、実際にGNURadioを使いこなすうえで必要な機能をマスターする
  - 内容
    - スライドバー
      - https://www.amazon.co.jp/dp/B0032Y0OH0/
      - https://github.com/nanosyzygy/ShuttlePRO
    - 変数

演習3-1:リアルタイム変更する

SIProp

- 中心周波数を変更しながら、表示を行う
  - 「固定値の変数化」を行う
    1. 「Variable」ブロックを追加する
       1. IDを「center_freq」とする
       2. Valueを「center_freq_val」とする
    2. 「Qt GUI Range」ブロックを追加する
       1. IDを「center_freq_val」とする
       2. Default Valueを「438.2e6」とする
       3. Start=437e6, Stop=439e6, Step=200をする
          - 最小周波数、最大周波数、ステップ周波数
    3. 「Pluto SDR Source」ブロックを追加する
       1. LO Frequencyを「int(center_freq)」にする
       2. 他は「演習2」と同じにする
    4. 「Qt GUI」ブロックを追加する
       1. Center Frequencyを「438.2e6」
       2. Bandwidthは「samp_rate」とする

おまけ

- ジョグダイヤルを使えるようにする
  - ターゲット製品
    - ShuttleXpress 3D
      - https://www.amazon.co.jp/dp/B0032Y0OH0/
    - インストール方法
      - https://github.com/nanosyzygy/ShuttlePRO

  - 使い方
    1. Ubuntuを起動する
    2. USBに本製品を挿す
    3. デスクトップ上の「ジョグダイヤル」アイコンをダブルクリックする
    4. GNURadioの「Range」にフォーカスを合わせる
    5. ジョグを回す

実は・・・

SIProp

- ここまでは、GNURadioに無線機として、最低限あるべき機能を付加した
  - 基本的な機能
    - 電波の送受信
    - 各種情報の表示
    - 中心周波数変更
- これ以降は、無線機としての完成度向上やSDRならではの機能を付加していく
  - 無線機の性能を上げる
    - LPFやAGCなど
  - 無線機の機能を増やす
    - IQデータを保存する
    - GMSKなどの変調方式
    - データ処理（TNC）

演習4：波形を保存して、再生する

SIProp

- SDR受信波をファイルに保存して、その保存したサイン波を「まったく同じに」読み込む

- 目的
  - GNURadioにおけるタイミング処理の概念を理解する
    - FPGAなどにおけるクロック処理の概念となる
      - 1クロック≒一定時間を理解する
  - 内容
    - 保存
    - 読み込み
      - 処理時間の概念の把握
  - 既存の無線機の違い
    - 保存して、何度でも再生し、実験できる

演習4-1：波形を保存して、再生する

- SDR受信波をファイルに保存して、その保存したサイン波を「まったく同じに」読み込む
  - 「演習3」のファイルをそのまま使います
    - IQデータをファイルに保存する
      1. 「File Sink」ブロックを追加する
         1. Fileに「iq_data.wav」を入力する
      2. 「Pluto SDR Source」ブロックと接続する

演習4-2：波形を保存して、再生する

- SDR受信波をファイルに保存して、その保存したサイン波を「まったく同じに」読み込む
  - 新規作成をします
    - ファイルに保存したIQデータを再生する
      1. 「File Source」ブロックを追加する
         1. Fileに「iq_data.wav」を入力する
         2. Repeatを「No」にする
      2. 「Qt GUI Sink」ブロックを追加する
         1. Center Frequencyを「438.2e6」とする
            1. 保存時の中心周波数と同じ値にする
      3. 接続する

演習4-3：波形を保存して、再生する

SIProp

- SDR受信波をファイルに保存して、その保存したサイン波を「まったく同じに」読み込む
  - 先程のファイルをそのまま使います
    - 処理時間を実時間に同期させる
      1. 「Throttle」ブロックを追加する
         1. Sampling Rateに「2084000」を入力する
            1. 保存時のSampling Rateと同じ値にする
      2. 「File Source」ブロックの間に挿入する
      3. 「Qt GUI Sink」ブロックを変更する
         1. Bandwidthに「2084000」を入力する
            1. 任意の値でよいが、初期値が32000と低すぎるため、変更したほうが見栄えが良い。

演習5:LPFをマスターする

SIProp

- LPFをいれて、波形を表示する

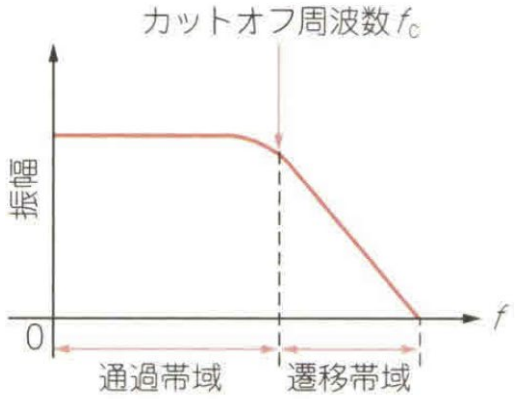
- 目的
  - アナログの処理器が付いていることを理解する
    - アナログ処理器ではあるが、内部的にはディジタル処理している
      - Tapで処理内容を変えられる
  - 内容
    - LPFの設定内容の解説
  - 既存の無線機の違い
    - 各種フィルターを固定値ではなく、自分で設計できる

演習5-1：LPFをマスターする

- LPFをいれて、波形を表示する
  - 「演習3」のファイルをそのまま使います
    - GNURadioにおけるLPFを理解する
      1. 「Low Pass Filter」ブロックを追加する
         1. Cutoff Freqに「640000」を入力する
         2. Transition Widthに「120000」を入力する
      2. 「Pluto SDR Source」ブロックの間に挿入する

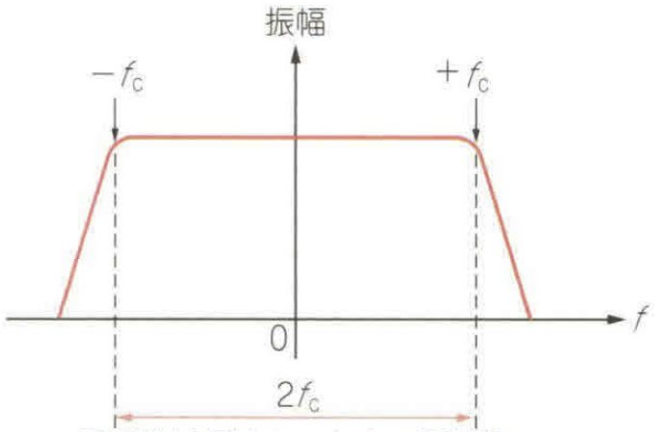
演習5:LPFをマスターする

- よくあるLPFにみえるけど・・・
  - アナログLPFとディジタルLPFの違い
    - 実数と複素数の違い！！！
    - ここだけ、抑えればアナログもディジタルも同じ！！！



ローパス・フィルタといえばこのような特性が思い浮かぶ

(a) アナログ信号処理による
ローパス・フィルタ特性



通過帯域幅はカットオフ周波数の2倍となる

複素信号に対してLPFはこのような正負対称の特性を持つ

(b) ディジタル信号処理による
ローパス・フィルタ特性

出典:RFワールドNo.44

演習6：GMSK Demodをマスターする

- GMSKを実際に受信して、データ化する
  - ただし、ファイルから読み込む形式
- 目的
  - コンスタレーション表示やアイパターン表示のさせ方を理解する
  - 内容
    - GMSKの可視化
  - 既存の無線機の違い
    - アイパターンやコンスタレーションも表示可能
      - ウォーターフォールだけじゃない！

演習6-1：GMSK Demodをマスターする

- GMSK9600を実際に受信して、データ化する
  - 先程のファイルをそのまま使います
    - GMSK復調し、タイム表示やコンスタレーション表示する
      1. 「GMSK Demod」ブロックを追加する
         1. 「Low Pass Filter」ブロックの後に接続する
      2. 「Low Pass Filter」ブロックを変更する
         1. Decimationに「samp_rate/9600/2」を入力する
         2. Cutoff Freqに「9600/2」を入力する
         3. Transition Widthに「600」を入力する
      3. 「Qt GUI Sink」ブロックを変更する
         1. Bandwidthに「9600*2」を入力する
      4. 「File Sink」ブロックを追加する
         1. Fileに「GMSK_demod.dat」を入力する
         2. Input Typeを「Byte」に変更する
      5. 保存したデータの表示コマンド
         1. hexdump -C GMSK_demod.dat

- AFSK1200送受信器を作成する

演習7：APRSパケット受信器を作成する

SIProp

- APRSパケットを受信してみる

- 目的
  - Audio入出力を利用して、既存のアマチュア無線機と連携させる

  - 内容
    - サンプリングレート
      - ≒インピーダンスマッチング
    - ハードウェア連携
  - 既存の無線機の違い
    - bit(データ)操作により、TNCを作成できる

演習7-1：APRSパケット受信器を作成する

- APRSパケットを受信してみる
  - 「study_7_start.grc」ファイルを読み込む
    1. 「Audio Sink」ブロックを追加する
    2. 「Multiply Const」ブロックを追加する
       1. IO Typeを「Float」に変更する
       2. Constantを「0.1」に変更する
          1. これがボリュームとなる
       3. 「Audio Sink」と「AX.25 FSK Mod」の間に接続する
    3. 「Audio Source」ブロックを追加する
       1. 「FSK Demodulator」と接続する
       2. 「Qt GUI Time Sink」と接続する
  - AFを利用し、既存無線機と連携する
    - 「Audio Sink」がスピーカー出力
    - 「Audio Source」がマイク入力である

- 実際に動かすには下記のライブラリが必要です
  - https://github.com/tkuester/gr-bruninga と https://github.com/dl1ksv/gr-ax25

- GR-satellitesをビルドして、GNURadioに登録する

演習8：GR-satellitesをビルドする

SIProp

- GR-satellitesをビルドして、GNURadioに登録する

- 目的
  - ネットに一杯存在しているGNURadioのレシピを自力で使えるようにする
    - ネットに存在しているものは、「ソースコード」の状態で存在してるため、コンパイル（ビルド）する必要がある
      - GR-satellites
        - https://github.com/daniestevez/gr-satellites
  - 内容
    - ライブラリの追加の方法
  - 既存の無線機の違い
    - 他人が作った機能を追加できる

コンパイル(ビルド)とは？

SIProp

- ソースコード(プログラム)を実行できる形に変換すること
  - その作業に必要なコマンド(ツール)
    - git clone [プログラムのURL]
      - ソースコードをダウンロードするためのコマンド
    - ./configure
      - コンパイルに必要な設定を自動で行うコマンド
    - cmake ../
      - 上記のconfigureと同じ。プログラムにより、こちらを使う場合もある
    - make
      - コンパイル(ビルド)を実際に行うコマンド
    - sudo make install
      - プログラムをインストールするコマンド

演習8-1： GR-satellitesをビルドする

SIProp

- ソースコードをダウンロードして、ビルドし、インストールする
  - 赤字部分がレシピ（アプリ）特有部分であとは同じである

```
user@pc:~ $ git clone https://github.com/daniestevez/libfec
user@pc:~ $ cd libfec
user@pc:~ $ ./configure
user@pc:~ $ make
user@pc:~ $ sudo make install
user@pc:~ $ cd ../

user@pc:~ $ git clone https://github.com/daniestevez/gr-satellites
user@pc:~ $ cd gr-satellites
user@pc:~ $ mkdir build
user@pc:~ $ cd build
user@pc:~ $ cmake ../
user@pc:~ $ make
user@pc:~ $ sudo make install
user@pc:~ $ cd ../../
```

今なら読みこなせる！

- 忘れてしまったら、こちらも読み返してください
  - RFワールド No.44
    - 「GRCで広がるSDRの世界」