T126

Rádio Definido por Software

Aula #02
Instrumentalização

Python vs C++

Python

- Interpretado;
- Fácil de aprender;
- Fácil acesso à bibliotecas;
- Facilidade de escrita de código;
- Bom para prototipagem rápida devido ao pequeno tamanho do código;
- Código com fácil manutenção, orientado à objetos e simples de usar;
- Lento;
- Numba.

C++

- Compilado;
- Grande curva de aprendizado;
- Menos bibliotecas compartilhadas que o Python;
- Sintaxe complexa quando comparado ao Python;
- Prototipagem rápida é difícil devido ao grande tamanho do código;
- Código menos limpo e gerenciável em comparação ao Python;
- Alta velocidade de execução;
- SIMD (SSE, AVX, AVX-512), VOLK.

Referência: YouTube - Python vs C++ (Advantages/Disadvantages) + Execution Speed Test

Python vs C++

Python

- Numba: traduz funções do Python reescritas em código de máquina otimizado, para melhorar o desempenho (velocidade) do código quando comparado à linguagens como C e FORTRAN.

C++

- SIMD (*Single Instruction Multiple Data*): é uma classe de instruções para processamento paralelo de instruções.
    - Explora o paralelismo no processamento dos dados: vários cálculos são executados ao mesmo tempo em uma única instrução do processador.

- VOLK: é uma biblioteca que fornece ao usuário uma abstração das instruções SIMD.

Numpy

- É a uma das principais bibliotecas em Python;
- Contém uma coleção de técnicas e ferramentas utilizadas para resolver problemas de modelos matemáticos computacionais;
    - *Array* (arranjo): estrutura de dados multidimensional de alto desempenho utilizada em cálculos de vetores e matrizes;
    - Grande quantidade de funções matemáticas de alto nível operando nesses vetores e matrizes.

Tipos de bloco do GNU Radio

- sync (1:1)
- source
- sink
- decimation (N:1)
- interpolation (1:M)
- general (N:M)
- tagged_stream
- hier
- noblock

Fontes: GNU Radio Wiki - Types of Blocks,
GNU Radio Wiki - Tagged Stream Blocks,
Stack Overflow - How can I create a Hierarchical Block with GNU Radio?,
GNU Radio Wiki - Tutorial: Write a Python function (noblock)

Tipos de bloco do GNU Radio

- **sync (1:1)**: apresenta um número qualquer de entradas igual ao número de saídas.
    - A relação entre o número de dados de entrada e o número de dados de saída é de 1 para 1.

- **source**: ocorre quando um bloco de sincronização possui zero entradas.

- **sink**: ocorre quando um bloco de sincronização possui zero saídas.

- **decimation (N:1)**: o número de dados de entrada é um múltiplo fixo do número de dados de saída.

Tipos de bloco do GNU Radio

- **interpolation (1:M)**: o número de dados de saída é um múltiplo fixo do número de dados de entrada.

- **general (N:M)**: a relação entre o número de dados de entrada e o número de dados de saída é de N para M.
    - Todos os outros blocos são apenas simplificações do bloco geral.
    - Deve-se optar por usar o bloco geral quando os demais blocos não forem adequados.

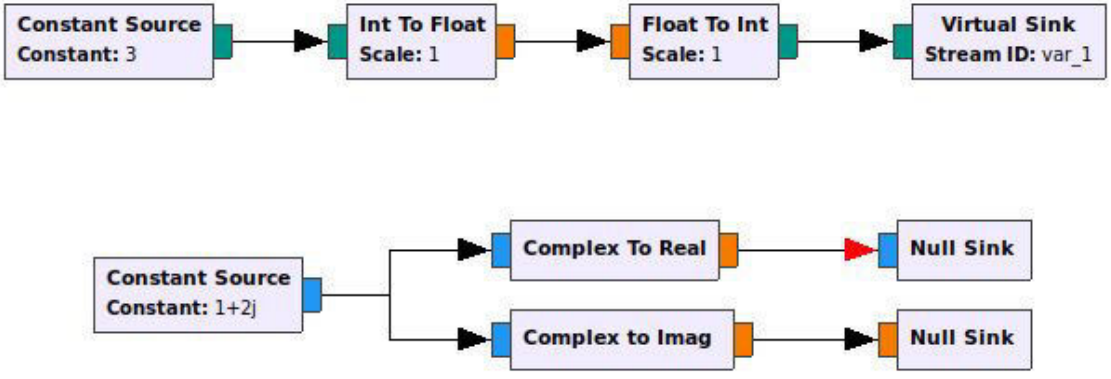
Tipos de bloco do GNU Radio

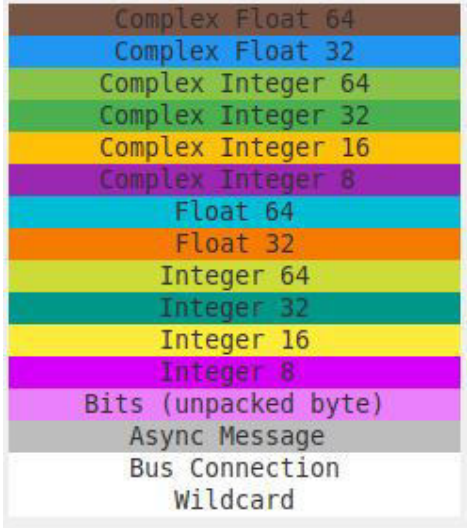
- **tagged_stream**: esse tipo de bloco permite adicionar ou receber uma *tag* em uma ou mais amostras do sinal, para uso dos próximos blocos do fluxograma.

- **hier**: podem ser criados a partir de blocos internos.
    - Podem ser instanciados dentro de outros gráficos de fluxo do GNU Radio.

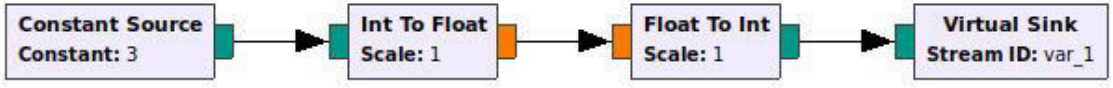
- **noblock**: consiste em um bloco que não segue a estrutura de um bloco convencional do GNU Radio e onde pode ser definida uma função qualquer que será utilizada pelo sistema quando for solicitada.

Estrutura de bloco do GNU Radio

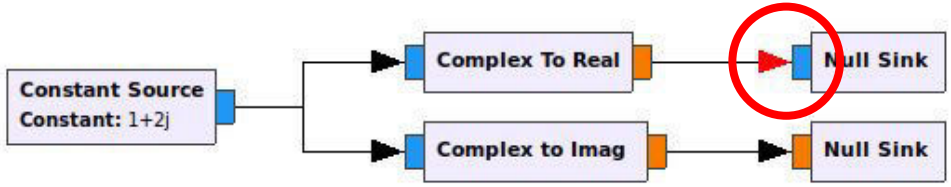


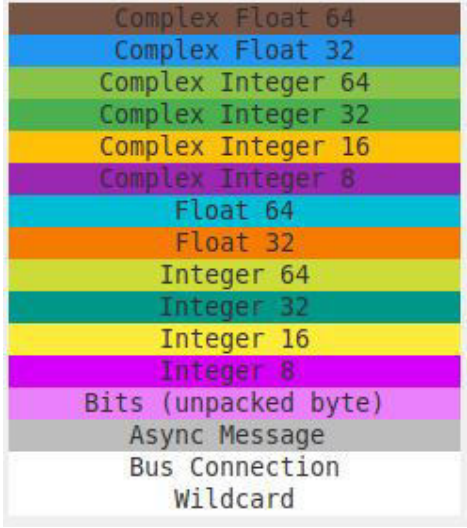


Estrutura de bloco do GNU Radio



Erro no tipo de dado





Estrutura de bloco do GNU Radio

```
from gnuradio import gr
import numpy as np


class mapper(gr.sync_block):
    """
    Comments for mapper block
    """
    def __init__(self, modulation):
        gr.sync_block.__init__(self,
                               name="mapper",
                               in_sig=[numpy.uint32],
                               out_sig=[numpy.complex64])
        if modulation == 0:
            self.modulation == 'bpsk'
        elif modulation == 1:
            self.modulation == 'qpsk'

        self.constellation = {
            'bpsk': np.array([-1+0j, 1+0j]), # 0, 1
            'qpsk': np.array([1+1j, 1-1j, -1-1j, -1+1j])} # 0, 1, 2, 3

    def work(self, input_items, output_items):
        in0 = input_items[0]
        out = output_items[0]
        out[:] = self.constellation[self.modulation].take(in0)
        return len(output_items[0])
```

Estrutura de bloco do GNU Radio

```
from gnuradio import gr
import numpy as np


class mapper(gr.sync_block):
    """
    Comments for mapper block
    """
    def __init__(self, modulation):
        gr.sync_block.__init__(self,
                               name="mapper",
                               in_sig=[numpy.uint32],
                               out_sig=[numpy.complex64])
        if modulation == 0:
            self.modulation == 'bpsk'
        elif modulation == 1:
            self.modulation == 'qpsk'

        self.constellation = {
            'bpsk': np.array([-1+0j, 1+0j]), # 0, 1
            'qpsk': np.array([1+1j, 1-1j, -1-1j, -1+1j])} # 0, 1, 2, 3

    def work(self, input_items, output_items):
        in0 = input_items[0]
        out = output_items[0]
        out[:] = self.constellation[self.modulation].take(in0)
        return len(output_items[0])
```

Construtor da classe

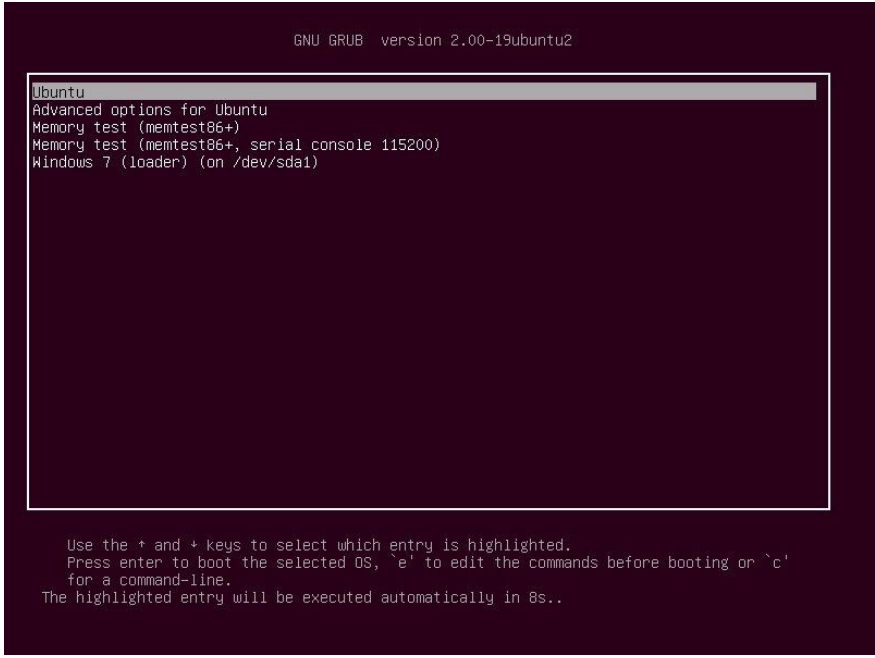
Estrutura de bloco do GNU Radio

```
from gnuradio import gr
import numpy as np


class mapper(gr.sync_block):
    """
    Comments for mapper block
    """
    def __init__(self, modulation):
        gr.sync_block.__init__(self,
                               name="mapper",
                               in_sig=[numpy.uint32],
                               out_sig=[numpy.complex64])
        if modulation == 0:
            self.modulation == 'bpsk'
        elif modulation == 1:
            self.modulation == 'qpsk'

        self.constellation = {
            'bpsk': np.array([-1+0j, 1+0j]), # 0, 1
            'qpsk': np.array([1+1j, 1-1j, -1-1j, -1+1j])} # 0, 1, 2, 3

    def work(self, input_items, output_items):
        in0 = input_items[0]
        out = output_items[0]
        out[:] = self.constellation[self.modulation].take(in0)
        return len(output_items[0])
```

Função *work*

Preparação de ambiente

- Instalar Ubuntu 18.04 e Windows em *Dual Boot* (link);
- Download de *releases* do Ubuntu (link);
- Programa para criar uma pen drive bootável (link);
- Opção sem instalação: GNU Radio Live SDR Environment (link 1, link 2).



Preparação de ambiente

```
#!/bin/bash

echo "Script for T126 SDR's softwares and drivers"

# Update system
sudo apt update
sudo apt upgrade

# Install GNU Radio Companion
sudo apt install gnuradio

# Install dependencies for ADALM-PLUTO drivers
sudo apt  install libxml2 libxml2-dev bison flex cmake git
libaio-dev libboost-all-dev swig

# Install libiio (ADALM-PLUTO driver)
git clone https://github.com/analogdevicesinc/libiio.git
cd libiio
cmake .
make
sudo make install
cd ..
```

```
# Install libad9361-iio
git clone https://github.com/analogdevicesinc/libad9361-iio.git
cd libad9361-iio
cmake .
make
sudo make install
cd ..


# Install gr-iio (Pluto SDR blocks for GNU Radio)
git clone https://github.com/analogdevicesinc/gr-iio.git
cd gr-iio
cmake -DCMAKE_INSTALL_PREFIX=/usr .
make
sudo make install
cd ..
sudo ldconfig
```

Instalação de máquina virtual Linux

- Como opção de acesso aos computadores do laboratório, podemos instalar uma máquina virtual Linux no computador pessoal, com sistema operacional Windows;
- O desempenho da máquina virtual é pior que a instalação em *dual boot* do Linux, mas não impedirá a realização das atividades da disciplina;
- Os arquivos para a instalação do gerenciador da máquina virtual (VirtualBox) e a imagem da máquina virtual estão disponibilizados em uma pasta no Google Drive.

Instalação de máquina virtual Linux

- Instalar o programa Oracle VM VirtualBox do arquivo `VirtualBox-6.1.4-136177-Win.exe` (durante a instalação, utilizar todas as opções padrão).
- Essa é a tela inicial do VirtualBox:

Instalação de máquina virtual Linux

Clicar em Importar

- Vamos importar uma imagem do Ubuntu já com o ambiente do GNU Radio;
- Selecione o arquivo `Ubuntu_T126.ova` e clique em Próximo (N);
- Na próxima tela, clicar novamente em Importar.

Instalação de máquina virtual Linux

- Para iniciar a máquina virtual, basta selecioná-la e clicar em Iniciar (T). O usuário para fazer *login* no Ubuntu é `aluno` e a senha é `alunoinatel`.

Instalação de máquina virtual Linux

- O módulo gr-t126, que será utilizado na disciplina, já foi adicionado à máquina virtual.
- O módulo gr-t126 deve ser adicionado à sua conta no GitHub:

```
git config --global user.name ‘<seu_nome>’
git config --global user.email ‘<seu_email>’

cd ~/gr-t126
git init
git add --all
git commit -m "Initial Commit"
git remote add origin <endereço_do_repositório_no_GitHub>
git push -u origin master
```

Atividade extraclasse

- Criação de ambiente de estudos em computador pessoal;
- Instalação do Ubuntu 18.04;
- Instalação do GNU Radio;
- Instalação de drivers para a Pluto SDR (opcional).