GNU Radio

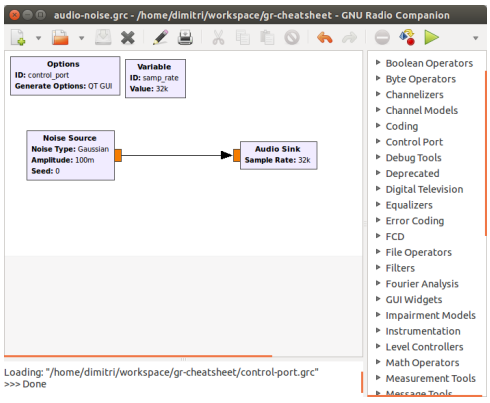
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

Installation

GNU Radio installation based upon PyBOMBS on Ubuntu OS

```
# Install pybombs via python-pip
sudo pip install PyBOMBS

# Add recipe lists from git repositories
pybombs recipes add gr-recipes git+https://github.com/
    ↪ gnuradio/gr-recipes.git
pybombs recipes add gr-etcetera git+https://github.com/
    ↪ gnuradio/gr-etcetera.git

# Set installation folder to '~/Desktop/pybombs'
pybombs prefix init ~/Desktop/pybombs -a myprefix

# Enable documentation
pybombs config builddocs=ON

# Run gnuradio installation with verbose output
pybombs -vv install gnuradio

# Outputs list of Gnuradio enabled components
# Check that the following components are enabled:
#
# ##############################################################
# # Gnuradio enabled components
# ##############################################################
#   * python-support
#   * testing-support
#   * volk
#   * doxygen
#   * gnuradio-runtime
#   * gr-ctrlport
#   * * thrift
#   * gr-blocks
#   * gnuradio-companion
#   * gr-fec
#   * gr-fft
#   * gr-filter
#   * gr-analog
#   * gr-digital
#   * gr-audio
#   * * alsa
#   * gr-channels
#   * gr-qtgui
#   * gr-uhd
#   * gr-utils
#   * gr-wavelet
#   * gr-zeromq

# Publish install variables as environment variables
source ~/Desktop/pybombs/setup_env.sh

# Apply also after re-booting
echo 'source ~/Desktop/pybombs/setup_env.sh' >> ~/.
    ↪ profile
echo 'source ~/Desktop/pybombs/setup_env.sh' >> ~/.bashrc

# Run GNU Radio Companion
gnuradio-companion
```

Getting Started



Run GNU Radio Companion (IDE):

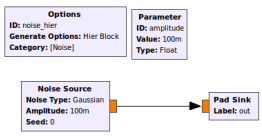
Toolbar to run flowgraphs + library search

Workspace current flowgraph

Library adding signal processing blocks

Terminal lists debug information

Create Hierarchical Block



Sub-flowgraphs can be re-used with hierarchical blocks

Generate Options set to `Hier Block`

Pad Source adds input port

Pad Sink adds output port

Parameter adds variable

Generated flowgraphs are exported to `~/.grc_gnuradio/` and will be available in GRC library after reloading

Create Python Block

New signal processing blocks can be added with **Python Block**

```
import numpy
from gnuradio import gr

class vector_sum_vff(gr.sync_block):
    def __init__(self, vlen):
        self.vlen = vlen
        gr.sync_block.__init__(self,
            name='vector_sum_vff',
            # Input signature: Float vector values
            in_sig=[(numpy.float32, vlen)],
            # Output signature: Float value
            out_sig=[(numpy.float32, 1)])

    def work(self,input_items,output_items):
        in0 = input_items[0]
        out = output_items[0]
        out[:] = numpy.sum(in0[0:1], axis=1)
        return 1
```

Signal processing lock for summation of an input vector:

Block type `gr.sync_block` for synchronized input and output item rates

In-/output signature `[(np.float32, 1)]` for 32-bit float items

Function work Signal processing goes here

Post-Processing

Matlab/octave post-processing of output file

```
% Open recorded cfile
f = fopen ('filename.cfile', 'rb');

% Activate recorded data type
%type = 'int'; % For int values
%type = 'char'; % For char values
%type = 'short'; % For cshort values
type = 'float'; % For float/complex values

% Read
v = fread (f, Inf, type);

% Activate for complex data type:
%v = v(1:2:end)+v(2:2:end)*j;

% Close cfile
fclose (f);

% Plot values
plot(v)
```

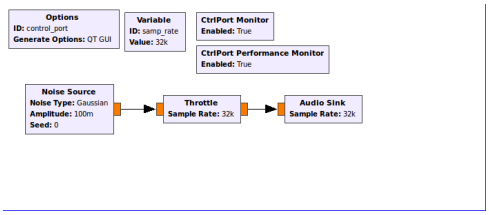
Performance Monitoring

OS requirements:

```
sudo pip install networkx

sudo apt-get install python-pygraphviz
```

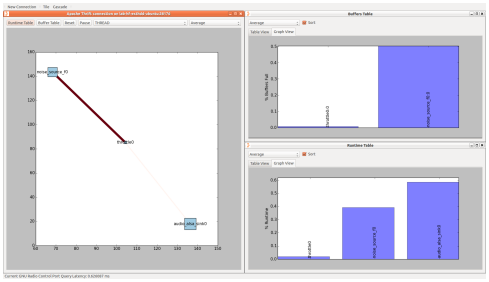
Change in file `./gnuradio/config.conf`:

```
...

[controlport]
edges_list = True
on = True

[perfcounters]
export = True
on = True

...
```



CtrlPort Monitor lists rates, memory, etc

CtrlPort Performance Monitor shows processing graph



Processing graph visualizes

Block size Processing time

Edge color/width Output buffer fullness