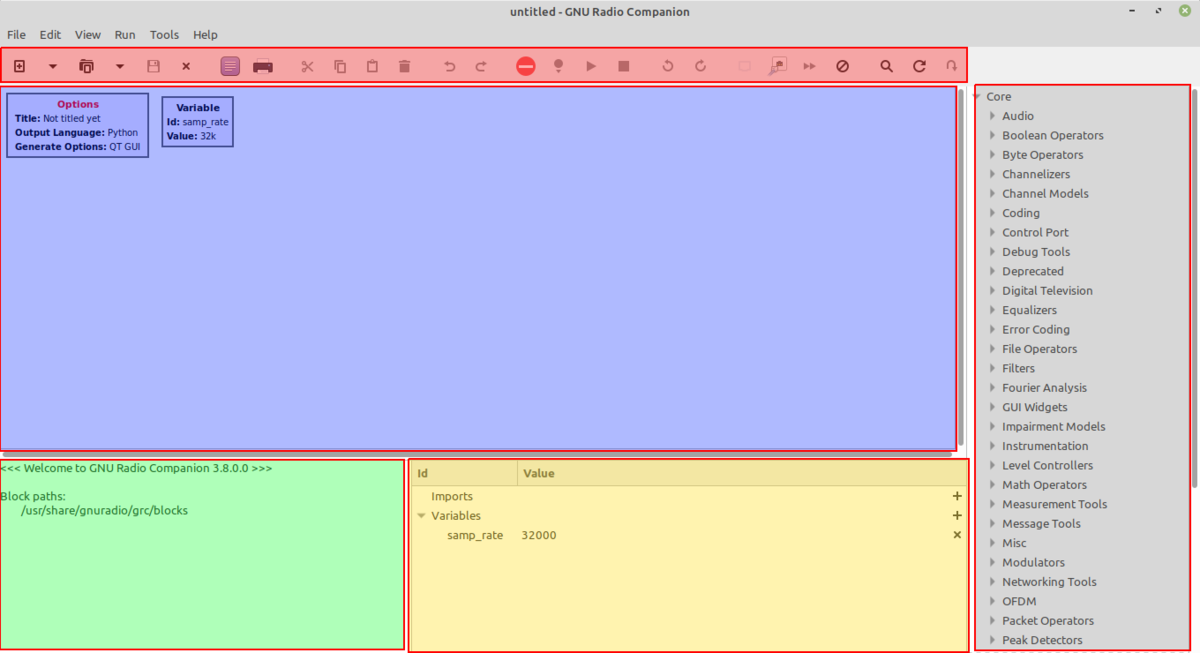
GUI Research and Documentation

SECTION I – GNU Radio



* Works as a simple drag-and-drop GUI interface
* Deleting of items will require the use of the ‘delete’ key as opposed to the ‘backspace’ key, or right-clicking and cutting the item
* Double-clicking of any block will bring up a context menu.

There is a wonderfully constructed tutorial for the basics of the workspace located here: <https://kaitlyn.guru/projects/gnu-radio-introduction/>

Continuing with the basics would undermine this existing resource. However, there are elements to the placement of GUI elements within a window that need recognition.

* All GUI elements (sinks, widgets, etc.) have a “GUI Hint” section at the bottom of the “General” tab in their respective context menus.
* This allows for a row-column system of placement within a single view window.
* The order of this arrangement is [row, col, row-span, col-span]

SECTION II – PyScript



Official Website: <https://pyscript.net/>

* Framework for integrating Python code into a browser

Core Components, taken from the PyScript website:

* **Python in the browser:** Enable drop-in content, external file hosting, and application hosting without the reliance on server-side configuration
* **Python ecosystem:** Run many popular packages of Python and the scientific stack (such as numpy, pandas, scikit-learn, and more)
* **Python with JavaScript:** Bi-directional communication between Python and Javascript objects and namespaces
* **Environment management:** Allow users to define what packages and files to include for the page code to run
* **Visual application development:** Use readily available curated UI components, such as buttons, containers, text boxes, and more
* **Flexible framework:** A flexible framework that can be leveraged to create and share new pluggable and extensible components directly in Python

NOTE:

The pyadi-iio library from Analog Devices is responsible for connectivity to Analog Devices’ Pluto SDR using Python code. While conducting research for PyScript, it was found that there was a consistent error when importing the library. Despite the pyadi-iio library being listed as part of PyScript’s library catalogue, an attribute error would occur where the program would attempt to “lookup unknown symbol ‘iio\_get\_backends\_count’” in a library for whihc pyadi-iio is built from: libiio.

The first assumption was that the libiio library was not integrated into PyScript, but this assumption begged the question as to why the pyadi-iio would be listed but not a library it was built from. Looking into the libiio GitHub, I found that the library, most notably the area of code referenced in the error, was written in C#.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Is this compatible with necessary libraries for conducting signal processing? | Does it support toggles, buttons, and other means of user interaction? | Can it update graphs continuously? | Can it simultaneously display multiple graphs and GUI elements? | Does it run at a reasonably responsive rate? |
| N | - | - | - | - |

Wiki: <https://wiki.analog.com/resources/tools-software/linux-software/pyadi-iio>

GitHub: <https://github.com/analogdevicesinc/pyadi-iio>

SECTION III – PyGame



Official Website: <https://www.pygame.org/news>

PyGame is a Python development toolkit that allows for the creation of games through the three main basis of interactive development:

* GUI Window – PyGame allows for the creation of a window that is customizable, allowing the display of graphics.
* Gameplay Loop – PyGame contains modules and tools to create a game loop for any type of game. In our group’s context, this gameplay loop is what can allow our graphs to be constantly updated.
* Event Handling – PyGame’s ability to monitor specific events is what enables buttons, dials, and even key bindings to interact with the GUI.

**III.A – PyGame-Chart**

PyGame-Chart is a complimentary package to PyGame that allows for MatPlotLib-style graph generation and manipulation.

<https://github.com/enessafak/pygame-chart>

Assessment of PyGame’s viability is variable. A test with the library PyGame-Chart was first conducted as methods of integrating standard graphing libraries into PyGame’s window system seemed reliant on manipulating the graph’s rendering.

Below is a general summary of the results when testing Jon Kraft’s monopulse tracker algorithm into PyGame using the complimentary PyGame-Chart library:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Is this compatible with necessary libraries for conducting signal processing? | Does it support toggles, buttons, and other means of user interaction? | Can it update graphs continuously? | Can it simultaneously display multiple graphs and GUI elements? | Does it run at a reasonably responsive rate? |
| Y | Y | Y | Y | N |

I found that PyGame itself was suitable for our project and offered much in terms of its ability to work with necessary libraries, but the overall performance was quite slow. This greatly holds this option back as a GUI platform alternative, but more can be investigated regarding PyGame specifically.

**III.B – MatPlotLib and the Rendering Method**

Sources online show it is possible to install MatPlotLib’s graphing methods into PyGame, though it greatly deals with MatPlotLib’s backend through AGG. In this way, one must utilize a buffer system to optimize the way in which PyGame tracks changes. Through this experiment, however I did get to work, it is not too different from the speed of MatPlotLib, which isn’t to say much fast at all.

<https://www.pygame.org/wiki/MatplotlibPygame>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Is this compatible with necessary libraries for conducting signal processing? | Does it support toggles, buttons, and other means of user interaction? | Can it update graphs continuously? | Can it simultaneously display multiple graphs and GUI elements? | Does it run at a reasonably responsive rate? |
| Y | Y | Y | Y | N |