Creating a Custom Block for GnuRadio

GnuRadio, being open-source software, is highly configurable due to all of its source code being public. The block-coding style used in the software allows for custom blocks to be created, that perform functions not available in the given library of a fresh installation. There exists official documentation on how to do this[1].

What Do We Need?

What we need is a clean way of getting data into GnuRadio, byte by byte. This will largely be based on the existing "File Source" block, whose source code is available online[2]. The problem with this block, is that it isn't great at receiving "live" data, as in data that is asynchronously passed to it. For example, it would be nice to be able to type a message and pass it to GnuRadio without having to save it as a file first. The block is also not great at repeating messages. There is an option to "repeat" files, but what it does is just loop back to the beginning of a file without any pause, which becomes difficult to receive.

To get a foot in the door on creating custom blocks, a first step would be to just create a file source block that has a built-in sleep timer. This sleep timer should be configurable from the flowgraph perspective, like a parameter.

Previous Approach and Why it Did Not Work

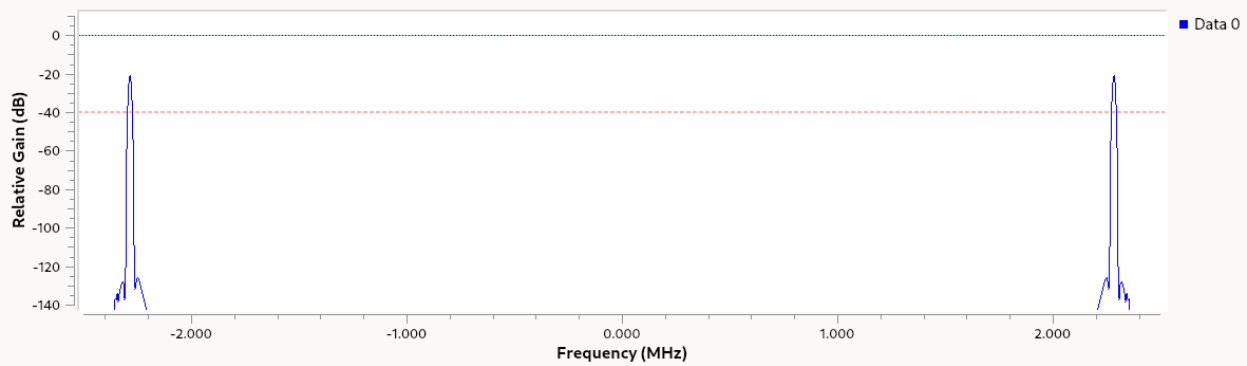
While this approach "worked" in the most technical sense, it was an extremely poor solution. It is a C program that passed a message to a 'FIFO' file byte by byte, and GnuRadio would read each byte from the FIFO file. This solution is documented in the ground station GitHub[3], along with a final report and presentation with further information.

It had many flaws. It is very inelegant due to the fact that the C program must first be run through the terminal, followed by running the associated GnuRadio flowgraph. It also has hard-coded synchronization messages meaning there is no option for configuration. Lastly, the timing is significantly off. When running, the messages seem to not be repeated with a steady period. The biggest issue, however, is that when not transmitting information, the C program feeds in a series of null characters, which is read by GnuRadio as '0', and then causes the SDR to transmit the carrier frequency despite no information being sent. This is a massive waste of power, and the main reason this task needs to be done.

---

[1] https://wiki.gnuradio.org/index.php?title=Creating_C%2B%2B_OOT_with_gr-modtool
[2] https://github.com/zimmerle/gnuradio/blob/master/gnuradio-core/src/lib/io/gr_file_source.cc
[3] https://github.com/JayTWilliams/GroundStation/tree/main/Documentation/Fall24

Below is what you will see on the frequency plot if just a long string of ‘0’ is being sent.



There is also reason to do this task so that more advanced code can be written to create state machines for better communication links, such as RDT (reliable data transmission).

Current Issues with Development

As linked previously, the GnuRadio documentation on how to create these blocks is very helpful. GnuRadio refers to these blocks as OOT or “Out-of-Tree”. These guides will help you install the necessary dependencies and guide you through the process of setting up environments to start working on a custom block.

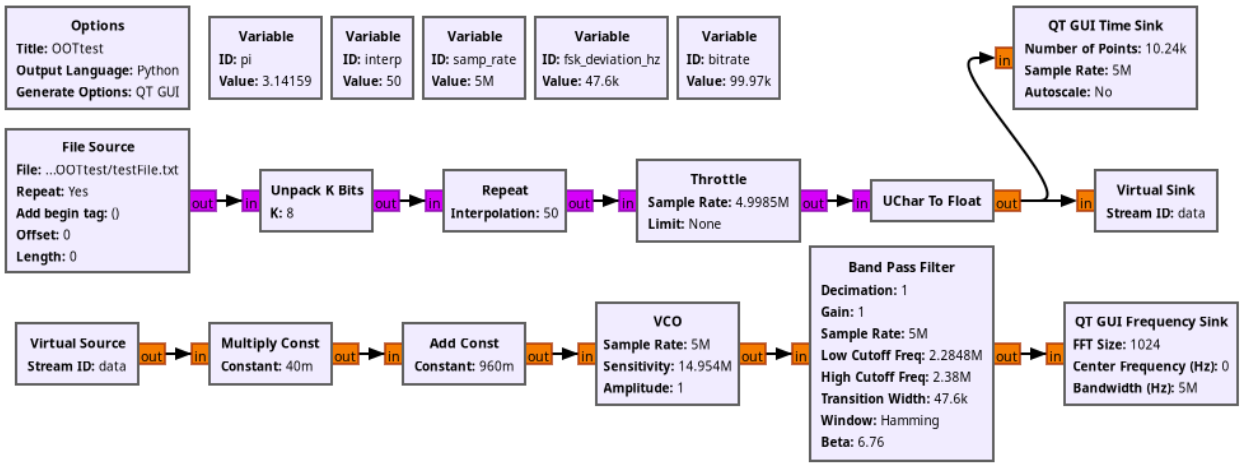
I have followed these guides a few times, but each time I run into the same error when running the command: `cmake ..` when compiling the block. The error screen is given below:

```
cmake: /builddir/build/BUILD/libuv-
v1.47.0/src/unix/process.c:972: uv_spawn: Assertion `!(options-
>flags & ~(UV_PROCESS_DETACHED | UV_PROCESS_SETGID |
UV_PROCESS_SETUID | UV_PROCESS_WINDOWS_HIDE |
UV_PROCESS_WINDOWS_HIDE_CONSOLE | UV_PROCESS_WINDOWS_HIDE_GUI |
UV_PROCESS_WINDOWS_VERBATIM_ARGUMENTS))' failed.

Aborted (core dumped)
```

This error message is the main roadblock I ran into when trying to create any custom block. Nothing seems to point towards any file I have written, so I cannot make any judgements as to what is going wrong. The main hunch that I have is that there is a software dependency issue.

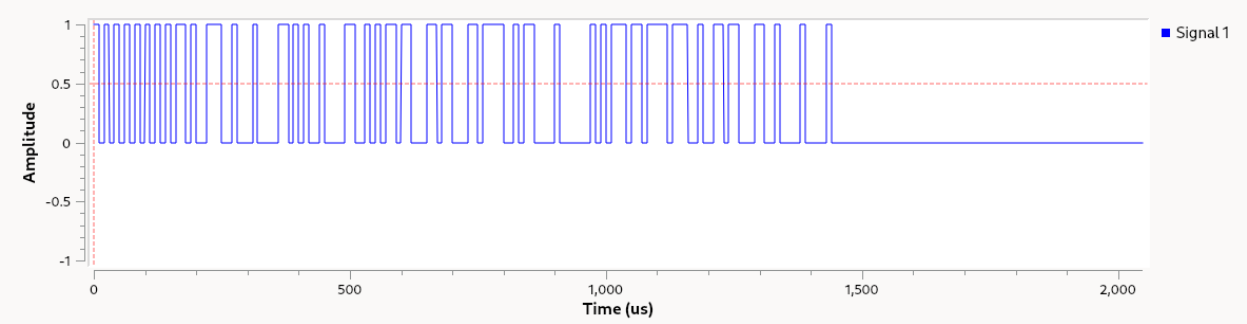
Your Task

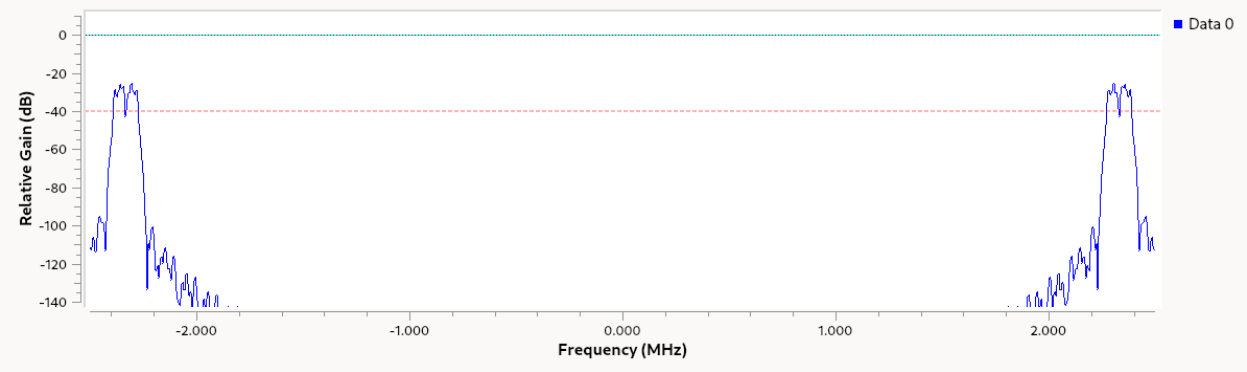


Above is the GnuRadio flowgraph you will be given to work with. It can be found under Documents -> gnuradio -> OOTtest. Nothing here is to be touched, except for the file source block. That block is a placeholder for the custom file source block you will be building.

To prove it works, use the test file found in the same directory, and demonstrate that the time domain plot of the bits loops every second (or whatever the sleep timer is set to). The frequency plot should also update at the same time. Both the time and frequency plots are set to update once it detects a new signal.

Below are some examples of what you should see in the time and frequency plots. Note that the time domain example contains different data than what you will be feeding, so if the bits look a little different, that is okay.





Notice how much ‘rougher’ the frequency plot looks when actual information is being processed. Also notice the two small peaks on top of each larger peak. These smaller peaks are the frequency being shifted for ‘0’ and ‘1’, which is the modulation scheme we are using (FSK).

Other Things to Consider

Please understand that the ‘solution’ I am presenting here may not be the best way to do this, let alone may not work at all even if everything is done correctly. This is really just an idea I had for what might fix the problem. If it doesn’t work, or you find a better solution through some additional research, I welcome anything you can give that solves the problem.