Peidong Qi
04/24/2018
AMRUPT, Spring 2018

RTL SDR to Raspberry Pi Connection and Datalogging

Goals

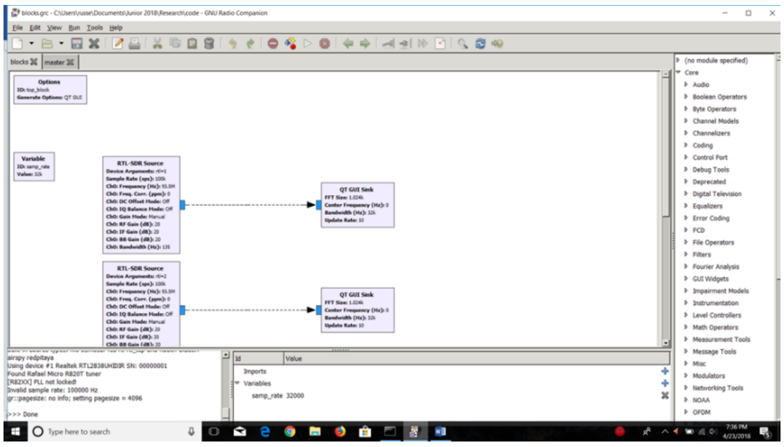
Run GRC on Raspberry PI and add custom blocks, build the flow graph for direction finding.   Install SDR card on Raspberry pi

General approach

At the beginning of this week, I have try to install SDR cards on Raspberry pi. Russell and I have successfully install SDR cards on Windows. We have run a program called "airspy" which use SDR card to receive radio signal. This proved our SDR card is working. Then I have tried to install SDR card on Raspberry PI.
There is a tutorial online, I successfully installed the SDR card on Raspberry PI by following the instruction.[9]. Here is result of installation:

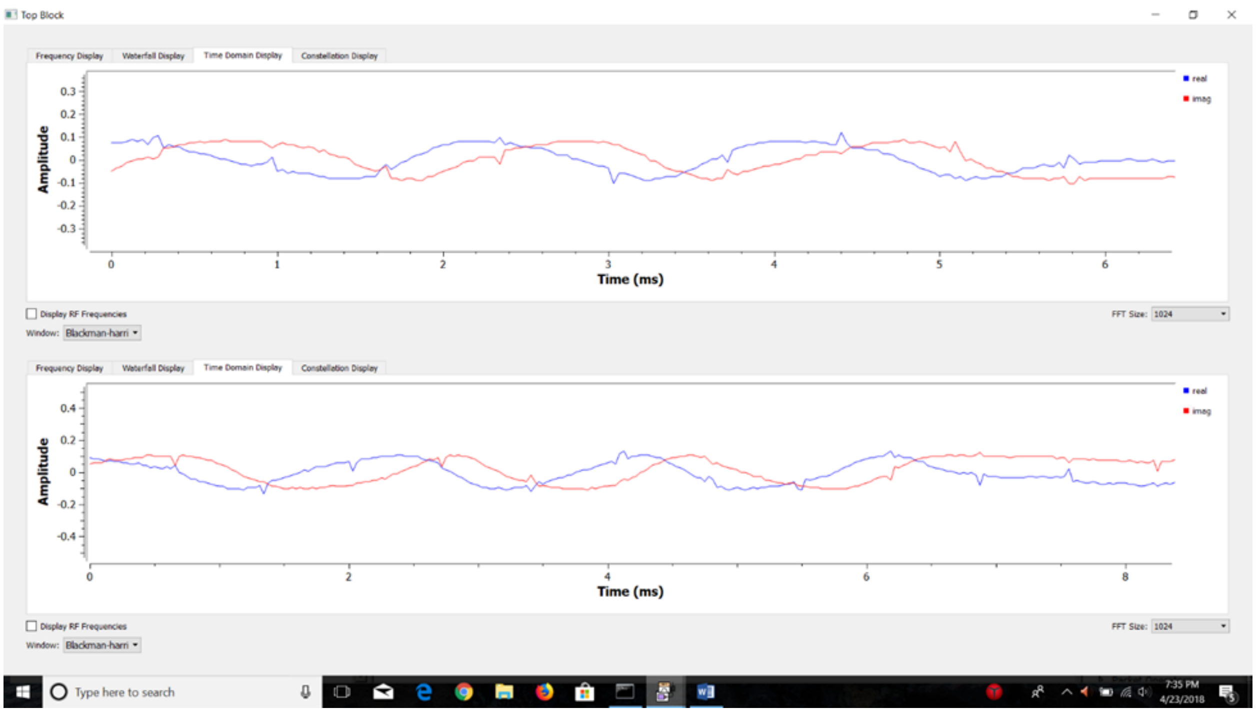
```
pi@raspberrypi ~ $ rtl_test
Found 1 device(s):
  0:  Generic, RTL2832U, SN: 77771111153705700

Using device 0: Generic RTL2832U
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7 20.7 22.9 25.4 28.0 29.7 32.8 33.8
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
```

After finish the installation, we run the rtl_test and able to reading samples.
Right now, we have both SDR card and GRC installed, we can build our own flowgraph.
The following flow graph is going to test two rtl sdr cards' signal.



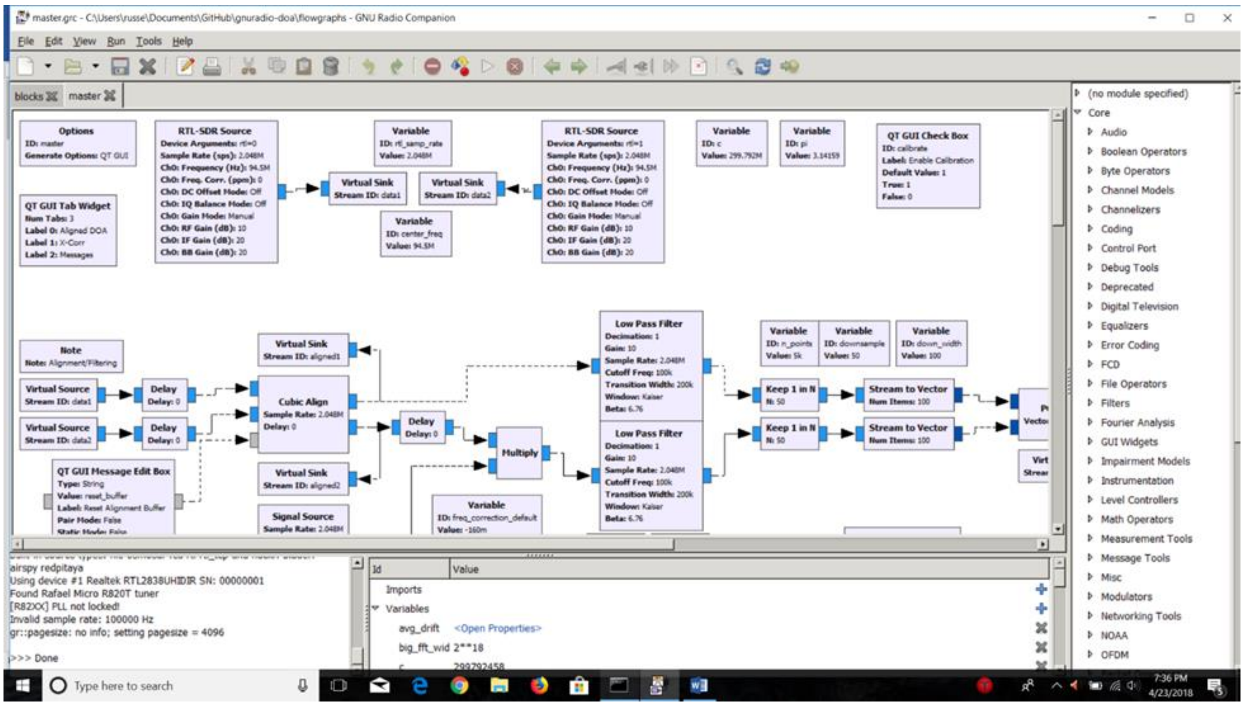
```
blocks.grc - C:\Users\russe\Documents\Junior 2018\Research\code - GNU Radio Companion

Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 32k

RTL-SDR Source
Device Arguments: rtl=1
Sample Rate (sps): 100k
Ch0: Frequency (Hz): 93.5M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 20
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20
Ch0: Bandwidth (Hz): 135

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 32k
Update Rate: 10

RTL-SDR Source
Device Arguments: rtl=2
Sample Rate (sps): 100k
Ch0: Frequency (Hz): 93.5M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 20
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 32k
Update Rate: 10

airspy redpitaya
Using device #1 Realtek RTL2838UHIDIR SN: 00000001
Found Rafael Micro R820T tuner
[R82XX] PLL not locked!
Invalid sample rate: 100000 Hz
gr::pagesize: no info; setting pagesize = 4096

>>> Done

Id        Value
Imports
Variables
  samp_rate  32000
```

When we run this flow graph with rtl sdr cards connected, we got the result shown below:



As we can see, we have received two sine wave with phase difference. That proved the rlt sdr cards working with GRC.

Then we implement the flow graphs from direction finding in coherence-receiver.



```
airspy redpitaya
Using device #1 Realtek RTL2838UHIDIR SN: 00000001
Found Rafael Micro R820T tuner
[R82XX] PLL not locked!
Invalid sample rate: 100000 Hz
gr::pagesize: no info; setting pagesize = 4096

>>> Done
```

For this flow graphs, we have a lot of missing blocks, we need to add custom blocks by ourselves. Russell was working on adding blocks on pc. I was working on adding blocks on raspberry pi. On Raspberry PI, we used “gr_modtool” to add customs block, the blocks are constructed by a header file and cpp file. In the code level, I have shown the three cpp codes for the custom blocks. The first block is pi2pi block, the function of this block is limit input signal from -2pi to 2pi. The second block is phase2doa block, the function of this block is convert phase to direction of arrival. The third blocks is called hold_ff block, the function of this block is hold the last value for which pass through this block.

Code level

Pi2pi block

```
int pi2pi_ff_impl::work(int noutput_items, gr_vector_const_void_star & input_items, gr_vector_void_star & output_items) {
    const float * in = (const float * ) input_items[0];
    float * out = (float * ) output_items[0]; // Do <+signal processing+>
    for (int i = 0; i < noutput_items; ++i) {
        out[i] = fmod( in [i] + 20 * PI, 2 * PI);
        if (out[i] > PI) out[i] -= 2 * PI;
    } // Tell runtime system how many output items we produced.
    return noutput_items;
}
```

Phase2doa block

```
int phase2doa_ff_impl::work(int noutput_items, gr_vector_const_void_star & input_items, gr_vector_void_star & output_items) {
    const float * in = (const float * ) input_items[0];
    float * out = (float * ) output_items[0];
    float arg;
    for (int i = 0; i < noutput_items; i++) { // DOA = arccos(phase/(2*pi*alpha)) - pi/2
        arg = in [i] / (2 * M_PI * d_alpha); // clip at -1 and 1 so we don't get NaNs
        if (arg > 1) {
            arg = 1;
        } else if (arg < -1) {
            arg = -1;
        }
        out[i] = acos(arg) - M_PI_2;
    } // Tell runtime system how many output items we produced.
    return noutput_items;
}
```

Hold_ff block

```
int hold_ff_impl::work(int noutput_items, gr_vector_const_void_star & input_items, gr_vector_void_star & output_items) {
    const float * in = (const float * ) input_items[0];
    float * out = (float * ) output_items[0];
    if (!d_hold) { //pass every sample through, save the last one into d_value
        memcpy(out, in , sizeof(float) * noutput_items);
        d_value = in [noutput_items - 1];
    } else { //hold = true, so just return the last given value
```

```
        std::fill(out, out + noutput_items, d_value);
    } // Tell runtime system how many output items we produced.
    return noutput_items;
}
```

Planned Course of Action

Try to run a sample project with GRC. Keep study the dirction-finding project from coherence-receiver website.

Resources and relevant Forum Posts

[1] “RTL-SDR Blog silver dongle first impressions, compared to NooElec blue dongle”https://medium.com/@rxseger/rtl-sdr-blog-silver-dongle-first-impressions-compared-to-nooelec-blue-dongle-4053729ab8c7
[2] “VIDEO TUTORIAL: INSTALLING GQRX AND RTL-SDR ON A RASPBERRY PI”https://www.rtl-sdr.com/video-tutorial-installing-gqrx-and-rtl-sdr-on-a-raspberry-pi/
[3] “DIGITAL RADIO WITH A RASPBERRY PI” http://www.michaelcarden.net/?p=48
[4] “Guided Tutorial GNU Radio in Python ”https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_Python
[5] “GRCon17 - Real-Time Direction Finding Using Two Antennas on an Android Phone - Sam Whiting” https://www.youtube.com/watch?v=jptYYiHth8U
[6]” Real-Time Direction Finding Using Two Antennas on an Android Phone - Sam Whiting” https://www.gnuradio.org/wp-content/uploads/2017/12/Todd-Moon-Gnuradio-DOA.pdf
[7]”gnuradio-doa” https://github.com/samwhiting/gnuradio-doa/tree/master/flowgraphs
[8]” GNURadioCompanion” https://wiki.gnuradio.org/index.php/GNURadioCompanion
[9] “Install RTL-SDR on Raspberry PI” https://gist.github.com/floehopper/99a0c8931f9d779b0998