Russell Silva
6/24/2018 – 6/30/2018
AMRUPT, Summer ‘18

Weekly Report #1 – RTL-SDR Hardware Configuration and Sampling Tests

Goals

- Successfully power and connect all five RTL-SDRs on the Coherent Receiver to the Raspberry Pi
- Perform sampling checks on all five RTL-SDRs individually and all at the same time

RTL-SDR Power Consumption and Connection to Raspberry Pi

Each RTL-SDR typically consumes 260 mA – 300 mA [2,3] when they are being used by a program (Linux Terminal, GNU Radio, GQRX, etc.). The Raspberry Pi support 1200 mA total for all usb peripheral devices [4]. Therefore, four RTL SDR connections may overwork the Raspberry Pi. Furthermore, five RTL SDRs need to be connected to the Raspberry Pi if data extraction is being performed on four RTL SDRs with one RTL SDR supervisor. The RTL SDR supervisor needs to be connected to the Raspberry Pi, so that RF switching commands can be sent to the coherent receiver from the same embedded device that is performing the data extraction (it would be more difficult and expensive to control RF switching from a different Raspberry Pi).

To combat power consumption and usb availability issues, a multiport usb hub was used to connect all RTL SDRs to the coherent receiver. This hub powers each RTL SDR, and multiplexes the usb transfers from all five RTL SDR devices into one signal output that can be connected to one usb port on the Raspberry Pi. The initial thought was that the multiplexing of multiple usb data transfers could possibly jeopardize the real time component of the data extraction for high sampling rates near the RTL-SDR stable sampling rate limit (2.56 MS/s); however, no samples were lost in the rtl_tests (discussed in continuous sampling check section). This is because each port on the usb hub supports data transfer of up to 5Gbps.

Continuous Sampling Check

If samples are lost during data extraction, it cannot be guaranteed that the RTL-SDR receivers remain synchronized [1]. Therefore, rtl_tests were performed to make sure that no samples were lost from the RTL SDRs when samples are being read at a high sampling rate of 2.3 MS/s for each RTL SDR.

The following code was entered into the Linux Terminal:

```
rtl_test -d0 -s 2300000 & rtl_test -d1 -s 2300000 & rtl_test -d2
-s 2300000 & rtl_test -d3 -s 2300000 & rtl_test -d4 -s 2300000
```

where -d specifies the receiver number (0: supervisor, 1-4: single channel receivers) and -s specifies the sampling rate.

By entering this code, an rtl_test is performed at each RTL SDR to see if any samples are lost with all receivers transmitting data at the same time. When the tests were first performed, error messages such as “PLL not locked,” “failed to allocate zero-copy buffers,” and initial sampling losses were reported. These errors occurred for the standard, most widely used RTL SDR driver library known as Osmocom. None of these error messages were displayed when the improved Keenerd driver library was used. The specific driver library installation can be found on the updated tutorial on GitHub.

For each individual rtl_test the following output is displayed for a successful test:

```
pi@raspberrypi:~ $ rtl_test -d0 -s 2300000

Found 5 device(s):
0:  Realtek, RTL2838UHIDIR, SN: 00000001
1:  Realtek, RTL2838UHIDIR, SN: 00000001
2:  Realtek, RTL2838UHIDIR, SN: 00000001
3:  Realtek, RTL2838UHIDIR, SN: 00000001
4:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
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 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0 49.6
Exact sample rate is: 2300000.022848 Hz
Sampling at 2300000 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...
```

If no samples are lost, no error messages will appear under “Reading samples in async mode…” for the entire duration of the test.

The parallel rtl_test, with all RTL SDRs being tested at the same time, has an output message with each line of an individual test (e.g. “Using device 0: Generic RTL2832U OEM) interleaved over each other. In this test, no error messages were seen (e.g. “Failed to allocate zero-copy buffer…”), and most of the tests did not record any sample losses. For a few tests, only an initial sample loss was recorded for a single receiver channel. Initial sample losses (a line “lost at least X bytes” occurs only once for an individual receiver) should not impact the synchronicity of receivers during data acquisition. This is because initial sample losses can be treated as negligible for a cross correlation using a much larger sample size than the sample loss (i.e. an initial sample loss of 10 bytes will be negligible in a cross correlation with 10000 bytes of data).

For the parallel rtl_test with all RTL SDRs to work successfully, the following line had to be entered as a root user within the Linux Terminal:

```
echo 0 > /sys/module/usbcore/parameters/usbfs_memory_mb
```

This command erases the maximum size limit of all buffers that libusb can allocate on the Kernel[1].

To login as a root user within the Linux Terminal, enter “sudo passwd root” within the terminal to set the root user password. Then, enter “su –“ to become the root user using the password from the previous step. After, a “#” instead of “$” should appear to the left of the command line. When this happens, enter the “echo…” command.

Resources and relevant Forum Posts

[1] https://coherent-receiver.com/getting-started
[2] https://www.rtl-sdr.com/forum/viewtopic.php?t=1587
[3] https://www.reddit.com/r/RTLSDR/comments/2e9u7v/power_consuption/
[4] https://www.raspberrypi.org/help/faqs/#powerReqs

Note: The work described above is representative of two days (Sunday – Tuesday: Proposal, Friday: Away). The Gannt Chart goals for next week - implementation of RF switching and cross correlation protocols to successfully synchronize data streams - are still applicable.