The entire installation guide for getting RTL SDR Drivers and GNU Radio working on a Raspbian OS.

Commands with the command prefix “sudo apt-get …” installs a widely used library on the Linux filesystem (which Raspbian uses). These commands can be done on a Linux OS (Raspbian included) installed to any machine (PC or Raspberry Pi). The machine used will determine the file system space (a memory space greater than 16 Megabytes is adequate for a correct installation of all the necessary libraries/modules in this tutorial, and a 16/32 Megabyte SD card is often used among Raspberry Pi users).

The purpose of this installation is to get the operating system to recognize RTL SDRs, acquire data samples from the RTL SDRs with minimal sampling losses, and download GNU Radio for DSP visualization purposes. The end goal of these steps will give the user the complete software framework for getting a functional AoA estimator running on a Raspberry Pi or PC (to the level we have achieved).

Important Note: This tutorial uses the terminology “generic cmake instructions” to refer to this set of instructions to install github libraries into Ubuntu:

$ git clone “*github library here”*

$ cd “*installed library name”*

$ mkdir build

$ cd build

$ cmake ..

$ make

$ make test

$ sudo make install

$ sudo ldconfig

1. Make Sure to update the operating system before the following installations:

pi@raspberrypi ~ $ sudo apt-get update  
pi@raspberrypi ~ $ sudo apt-get upgrade

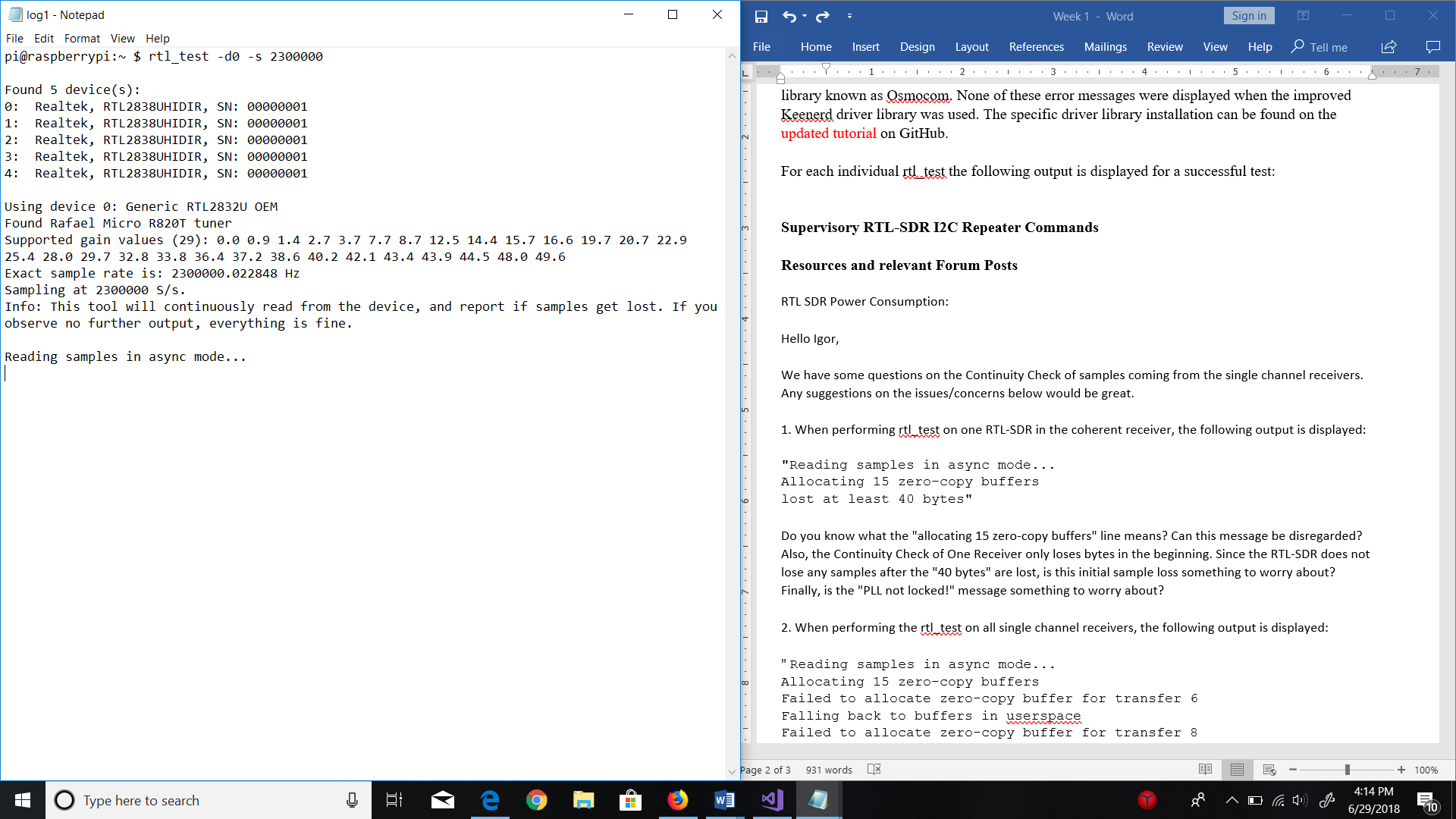
2. Install GNU Radio by using the command sudo apt-get install gnuradio-dev

3. Next, install the RTL SDR drivers, this part of the tutorial largely derives from [this page](https://gist.github.com/floehopper/99a0c8931f9d779b0998). It is essential for the Keenerd driver library to be installed, and not the standard Osmocom one for RTL SDRs within a coherent receiver to work properly.

pi@raspberrypi ~ $ cat <<EOF >no-rtl.conf  
blacklist dvb\_usb\_rtl28xxu  
blacklist rtl2832  
blacklist rtl2830  
EOF  
pi@raspberrypi ~ $ sudo mv no-rtl.conf /etc/modprobe.d/  
  
pi@raspberrypi ~ $ sudo apt-get install git-core  
pi@raspberrypi ~ $ sudo apt-get install git  
pi@raspberrypi ~ $ sudo apt-get install cmake  
pi@raspberrypi ~ $ sudo apt-get install libusb-1.0-0-dev  
pi@raspberrypi ~ $ sudo apt-get install build-essential  
  
pi@raspberrypi ~ $ git clone https://github.com/keenerd/rtl-sdr  
pi@raspberrypi ~ $ cd rtl-sdr/  
pi@raspberrypi ~/rtl-sdr $ mkdir build  
pi@raspberrypi ~/rtl-sdr $ cd build  
pi@raspberrypi ~/rtl-sdr/build $ cmake ../ -DINSTALL\_UDEV\_RULES=ON  
pi@raspberrypi ~/rtl-sdr/build $ make  
pi@raspberrypi ~/rtl-sdr/build $ sudo make install  
pi@raspberrypi ~/rtl-sdr/build $ sudo ldconfig  
pi@raspberrypi ~/rtl-sdr/build $ cd ~  
pi@raspberrypi ~ $ sudo cp ./rtl-sdr/rtl-sdr.rules /etc/udev/rules.d/  
pi@raspberrypi ~ $ sudo reboot

After sending the above commands, you should be able to perform a simple test which generates a list of RTL SDRs recognized by the operating system. The test also checks the possible tuning range of each RTL SDR and the maximum sampling rate possible on the device (computer or Raspberry Pi) being use. To perform an rtl\_test on an individual RTL SDR with a specified sampling rate enter “rtl\_test -dX -s Y” where X is an RTL SDR id and Y is a sampling rate.

A successful test on an individual RTL SDR produces the following output:

  
To perform a parallel test on all connected RTL SDRs, enter something like this:

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

This test should have an output message with each line of an individual test message (e.g. “Using device 0: Generic RTL2832U OEM) interleaved over each other. In this test, no error messages should be displayed (e.g. “Failed to allocate zero-copy buffer…”), and if there are sample losses, they should only be initial sample losses (a line “lost at least X bytes” occurs only once for an individual receiver). Initial sample losses 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).

4. Install the bias\_tee software from <https://github.com/rtlsdrblog/rtl_biast> (this will be used to turn the bias\_tee on/off when performing RF Switching (not yet completed)) using the generic cmake instructions.