Contents

Raspberry PI Operating System configuration 1
Code dependencies, libraries and tools 1
Downloading and building rtl_power_fftw 3
Special configuration for Raspberry PI 3 only 3
GNUPLOT 3
Setup and configuration of the Python programs 5
Sharing the output files with the community 6
A diagram of programs interactions 7

Raspberry PI Operating System configuration

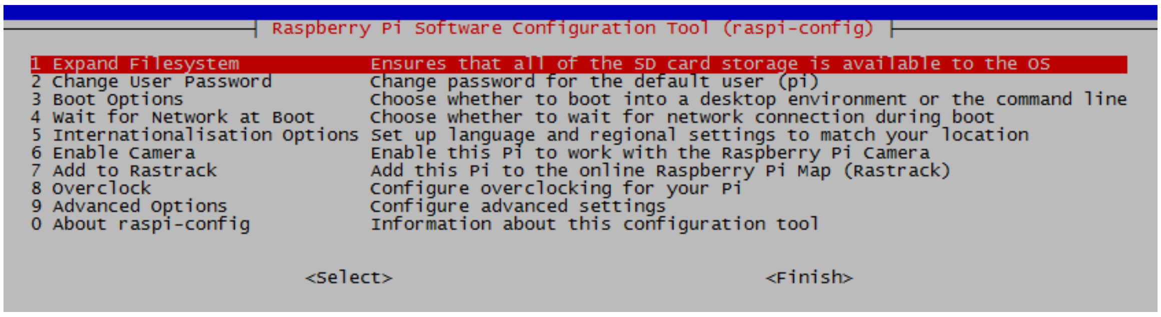
Please follow the instructions below in the sequence they are presented.
Some of these commands will ask you confirmation: reply yes when requested.

If you already have the latest Raspbian Jessie OS ready on a suitable microSD you can skip to the next block "*Code dependencies, libraries and tools*".

1. Download the OS image file from here: https://www.raspberrypi.org/downloads/
2. unzip and write to microSD using win32diskimager following the Raspberry Pi official instructions
3. insert the microsd in the R-PI slot and switch on / boot the R-PI
4. connect using Putty or similar terminal client using the default credentials (user: **pi** password: **raspberry**)

Extend the micro SD partition to full size using the raspi-config tool menu:

```
sudo raspi-config
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem              Ensures that all of the SD card storage is available to the OS
2 Change User Password           Change password for the default user (pi)
3 Boot Options                   Choose whether to boot into a desktop environment or the command line
4 Wait for Network at Boot       Choose whether to wait for network connection during boot
5 Internationalisation Options   Set up language and regional settings to match your location
6 Enable Camera                  Enable this Pi to work with the Raspberry Pi Camera
7 Add to Rastrack                Add this Pi to the online Raspberry Pi Map (Rastrack)
8 Overclock                      Configure overclocking for your Pi
9 Advanced Options               Configure advanced settings
0 About raspi-config             Information about this configuration tool

                    <Select>                         <Finish>
```

Also set the language / keyboard / time zone to your preferences using the same menu.
Then select exit and reboot using this command:

```
sudo reboot
```

Code dependencies, libraries and tools

We will compile the radio software from keenerd sources, so we first prepare the development environment with some tools/libraries. Enter the following:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install cmake
sudo apt-get install libusb-1.0-0-dev
git clone https://github.com/keenerd/rtl-sdr.git
```

in this way you prepared the development environment and downloaded the open source sw tools for the RTL-SDR radio that will be used to run the scans.
Build the software using the following commands:

```
cd rtl-sdr
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

at this point the sw has been built and installed.

Change directory so that we can create a file to avoid issues with the device drivers:

```
cd /etc/modprobe.d
```

create this file:

```
sudo nano no-rtl.conf
```

and now enter these 3 lines:

```
blacklist dvb_usb_rtl28xxu
blacklist rtl2832
blacklist rtl2830
```

then to save hit ctrl-o <enter> and then ctrl-x to exit

If your rtl-sdr dongle was already connected to the R-Pi usb port when you booted up, you need to remove the driver currently loaded:

```
sudo rmmod dvb_usb_rtl28xxu rtl2832
```

Go back to the main folder:

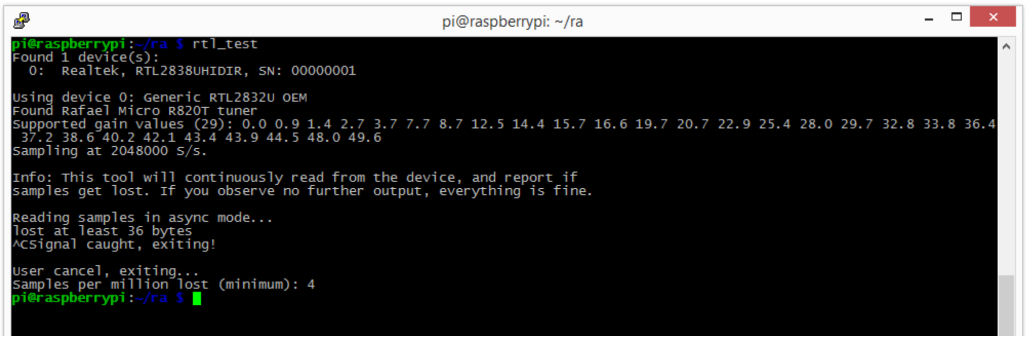
```
cd /home/pi
```

now connect the dongle to one of the R-PI usb ports and enter this command to run a first test:

```
rtl_test
```

this will test the possible tuning range and packet loss if any. It will automatically find the dongle device and start the test. If you get errors please check the previous steps.

Example output:



```
pi@raspberrypi:~/ra $ rtl_test
Found 1 device(s):
  0:  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
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...
lost at least 36 bytes
^CSignal caught, exiting!

User cancel, exiting...
Samples per million lost (minimum): 4
pi@raspberrypi:~/ra $
```

Downloading and building rtl_power_fftw

Please note that at the moment only my branch contains the crop percentage parameter. Other changes have already been merged in the main sources by the authors of the origin version. I will be working with them for some time on a new way to manage data with post-processing programs. In the mean time, I will keep separate this branch.
You can download and build my branch of rtl_power_fftw with these commands:

```
git clone https://github.com/mariocannistra/rtl-power-fftw.git
```

Only if it is the first time that you build this program you also have to enter:

```
sudo apt-get install libfftw3-dev libtclap-dev
```

... to setup your environment.
Then you go to the program folder and enter the following commands to compile it:

```
cd rtl-power-fftw
mkdir build
cd build
cmake ..
make
```

Special configuration for Raspberry PI 3 only

Ever since we want to avoid producing additional RFI, we should disable the R-PI-3 WiFi and bluetooth. We will connect exclusively using an ethernet cable with ferrites possibly at both ends.
In order to disable both WiFi and Bluetooth, enter the following command:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

then add the following lines:

```
#wifi
blacklist brcmfmac
blacklist brcmutil
#bt
blacklist btbcm
blacklist hci_uart
```

and then to save hit ctrl-o <enter> and then ctrl-x to exit
Reboot your R-PI entering:

```
sudo reboot
```

At this point you will be able to connect only using the cabled network that is enabled by default.

GNUPLOT

We will use the latest version of gnuplot to produce our charts and we will build it from sources. Here we prepare the R-PI with the required library to build it with options to produce png and jpeg output:

```
sudo apt-get install libgd2-dev
```

We will also use a python based astronomy package named SkyField that has some other code dependencies. It will take some time to compile numpy, be patient (on Raspberry PI 3 it's already available within the OS image):

```
sudo apt-get install python-dev
sudo pip install numpy
```

these will take some time to complete and display several warnings during compile (on R-PI-3 3'50" for numexpr) :

```
sudo pip install numexpr
sudo pip install pytz
```

then install skyfield, pyephem (on R-PI-3 1'27") and the AWS Python modules using the following commands.
Please note that my python programs have been tested with current Skyfield version 0.6.1 .
I will update these instructions when I will test my programs with newer Skyfield releases.

```
sudo pip install skyfield==0.6.1
sudo pip install pyephem
sudo pip install boto
```

We are now ready to get the gnuplot sources and compile it as well (on R-Pi-3 1'28" for configuration and 6'25" for make):

```
wget http://sourceforge.net/projects/gnuplot/files/gnuplot/5.0.1/gnuplot-
5.0.1.tar.gz

tar -xvzf gnuplot-5.0.1.tar.gz

cd gnuplot-5.0.1

./configure

Make

sudo make install
```

go back to the home folder to install the MQTT module for python and the datetime tool to force clock sync with NTP servers (more on this few lines later).

```
cd /home/pi
sudo pip install paho-mqtt
sudo apt-get install ntpdate
```

The Raspbian image comes with NTP daemon already installed with default configuration. This means that you will get the date/time updated within few minutes from boot. Without going too much in detail, let's say that you can configure your preferred servers so that it can reach them with the shortest possible network delay. I have the Italian official atomic clock source at 40 Km so I will configure its 2 NTP servers on the R-PI. You can search the web to find out your closest one. Why I'm doing this ? Having the data properly time stamped is important when you have to compare your data with that produced by others and having it precise enough is a good start.
Enter the following:

```
sudo nano /etc/ntp.conf
```

scroll to the first "server" statement and add one or more lines like these:

```
server ntp1.inrim.it
server ntp2.inrim.it
```

We now do a similar config for ntpdate:

```
sudo nano /etc/default/ntpdate
```

add these 2 servers :

```
ntp1.inrim.it ntp2.inrim.it
```

Now install one more package used by my python programs (on R-PI-3 this takes 3'28") :

```
sudo apt-get install python-matplotlib
```

Setup and configuration of the Python programs

Make sure you are positioned in your home folder:

```
cd /home/pi
```

then retrieve my custom scripts and python programs:

```
git clone https://github.com/mariocannistra/radio-astronomy-fftw.git
cd radio-astronomy-fftw
```

Configure your geographical position and scan parameters editing radioConfig.py:

```
sudo nano radioConfig.py
```

you will see three groups of settings: radio scan, position of observatory, AWS config:

```
# scan parameters configured here
# are used by all the scripts

freqCenter = 21000000   # if you want to scan from 17 to 25 MHz you enter here 21000000
as center frequency and 8000000 below as bandwidth
freqBandwidth = 8000000
upconvFreqHz = 125000000    # this is your upconverter offset frequency in Hz

# maximum value successfully tested on the R-PI-3 : 2800000
rtlSampleRateHz = 2000000

# specify one of the two values and set the other to 0
totalFFTbins = 4000    # this is the total number of bins available on a scan (will be
divided by number of necessary hops)
binSizeHz = 0 # this is the FFT bin size in Hz, lower values will give you more detail
in spectrograms, but larger files to process on the R-PI (gnuplot is memory hungry)

# Specify the percentage of bins to discarded on each scan.
# It will actually be half on each side of a single scan.
cropPercentage = 30

# specify one of the two values and set the other to 0
integrationIntervalSec = 0.5   # see rtl_power_fftw documentation, this is the
integration time in seconds. can be < 1 sec
integrationScans = 0    # see rtl_power_fftw documentation, this is the number of scans
that will be averaged for integration purposes. Allows to integrate for less than 1
second.

gain = 30   # maximum gain is about 49, can be too much in certain positions with
strong interfering sources
linearPower = False    # flag to calculate linear power values instead of logarithmic
tunerOffsetPPM = 0 # see rtl_power documentation, i can use 0 on my TCXO based dongle,
cab be anything from 0 to 60 or more depending on the dongle (use kalibrate to find it
out)
dataGatheringDurationMin = 30   # duration of single scan in minutes. 30 minutes is the
suggested duration for better chart and limit of 1 GB on memory available to gnuplot
sessionDurationMin = 600    # overall session duration in minutes 10 hours = 10 * 60 =
600
```

```
scanTarget = "Jupiter"  # this string will be used to create a subfolder under which
all sessions spectrograms will be stored (one subfolder per each date)

# the following observer position is used by Jupiter-Io radio emission
# prediction utility to calculate Jupiter apparent position / visibility
# for your site:
stationID = "myObservatoryNameOrAcronym"
stationTimezone = "Europe/Rome"
stationLat = "00.00000 X"  # enter your station latitude here with this string format
"00.00000 X"  where X is either N or S (North or Sud)
stationLon = "00.00000 X"  # enter your station longitude here with this string format
"00.00000 X"  where X is either E or W (East or West)
stationElev = 0   # enter your station elevation in meters (above sea level)
# here you can specify:
plotWaterfall = True
uploadToS3 = False     # if you want to upload your scans
sendIoTmsg = False  # if you want to send notifications. Please, get in touch with me
to get the certificates in order to send mqtt notifications
# do not change the following lines:
awshost = "A101P9EGJE1N9Z.iot.eu-west-1.amazonaws.com"
awsport = 8883
clientId = "mcawsthings-test2"
thingName = "mcawsthings-test2"
caPath = "./aws-iot-rootCA.crt"
certPath = "./cert.pem"
keyPath = "./privkey.pem"
```

Now download the JPL ephemerides files using the following command. It will download 2 files for a total of 1 GB to your R-PI (relax, it will do this only the first time)

```
python dl-ephemeris.py
```

As an alternative, you can download them from http://naif.jpl.nasa.gov/ and then copy them to your R-PI via sftp or WinSCP. **These are the direct urls for the 2 files:**

```
http://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/satellites/jup310.bsp
http://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/a_old_versions/de421.bsp
```

In this case download them to the same /home/pi/radio-astronomy-fftw folder and then run in any case the following program to test that the SkyField based programs can work properly. The program will find the files already there and will just output some test information:

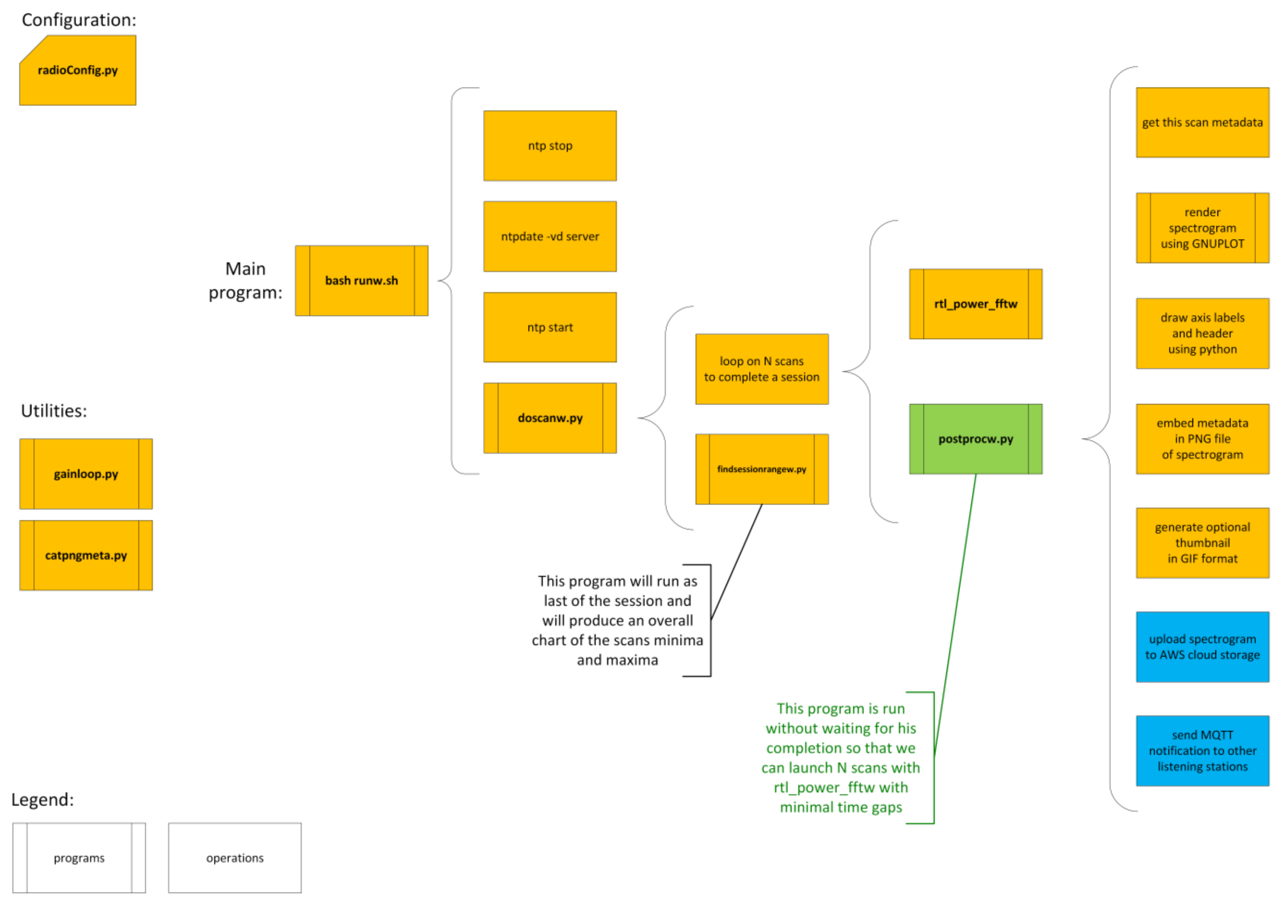
```
python dl-ephemeris.py

UTC:  2016-01-06 11:04:49.445000+00:00
Local: 2016-01-06 12:04:49.445000+01:00

Jupiter position from my lat-long:
(<Angle 11h 32m 46.41s>, <Angle +04deg 12' 21.6">, <Distance 5.31815 au>)

Io position around Jupiter:
(<Angle 09h 37m 28.75s>, <Angle +14deg 47' 36.9">, <Distance 0.00281808 au>)
<skyfield.jpllib.Body object at 0x75940610>

Jovicentric Declination of Earth:
(<Angle 23h 32m 46.88s>, <Angle -04deg 12' 19.6">, <Distance 5.31871 au>)
```

Sharing the output files with the community

In order to send notifications using MQTT, you need 3 AWS IoT certificate files to be in the same folder of my python programs. If you wish to participate in the cloud based shared experience just get in touch with me and I will send you those 3 files. If you don't want to send notifications just edit **radioConfig.py** to leave **sendIoTmsg = False**

A diagram of programs interactions

Please find below a diagram showing the relationships between the various programs:


Configuration:
radioConfig.py
Main program:
bash runw.sh
ntp stop
ntpdate -vd server
ntp start
doscanw.py
loop on N scans to complete a session
findsessionrangew.py
rtl_power_fftw
postprocw.py
get this scan metadata
render spectrogram using GNUPLOT
draw axis labels and header using python
embed metadata in PNG file of spectrogram
generate optional thumbnail in GIF format
upload spectrogram to AWS cloud storage
send MQTT notification to other listening stations
This program will run as last of the session and will produce an overall chart of the scans minima and maxima
This program is run without waiting for his completion so that we can launch N scans with rtl_power_fftw with minimal time gaps
Utilities:
gainloop.py
catpngmeta.py
Legend:
programs
operations