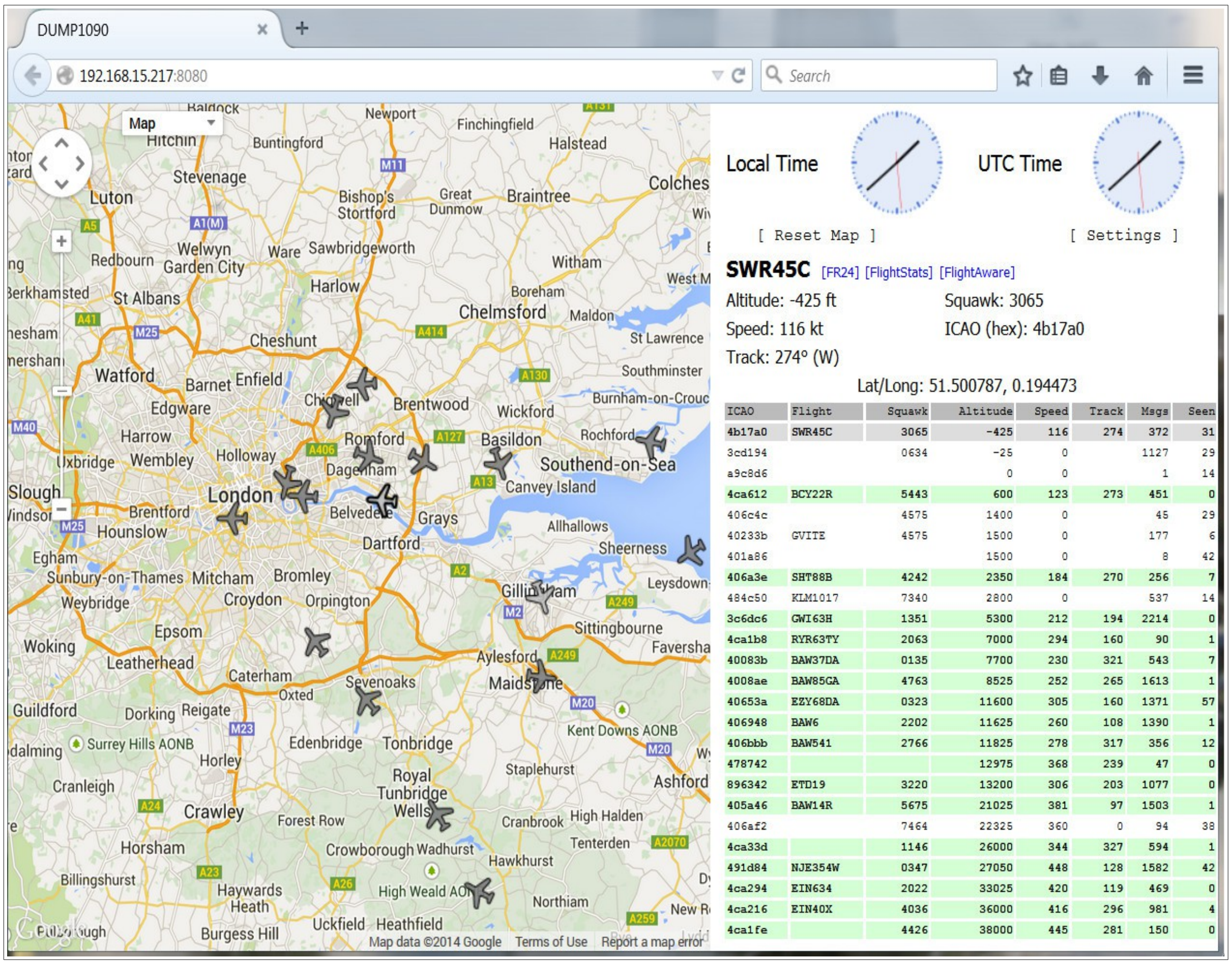
Raspberry PI 'How-To' Series

Using Software Defined Radio to Track Airplanes

KEEDOX DVB-T USB TV RTL-SDR



SWR45C

Altitude: -425 ft
Squawk: 3065
Speed: 116 kt
ICAO (hex): 4b17a0
Track: 274° (W)
Lat/Long: 51.500787, 0.194473

ICAO	Flight	Squawk	Altitude	Speed	Track	Msgs	Seen
4b17a0	SWR45C	3065	-425	116	274	372	31
3cd194		0634	-25	0		1127	29
a9c8d6			0	0		1	14
4ca612	BCY22R	5443	600	123	273	451	0
406c4c		4575	1400	0		45	29
40233b	GVITE	4575	1500	0		177	6
401a86			1500	0		8	42
406a3e	SHT88B	4242	2350	184	270	256	7
484c50	KLM1017	7340	2800	0		537	14
3c6dc6	GWI63H	1351	5300	212	194	2214	0
4ca1b8	RYR63TY	2063	7000	294	160	90	1
40083b	BAW37DA	0135	7700	230	321	543	7
4008ae	BAW85GA	4763	8525	252	265	1613	1
40653a	EZY68DA	0323	11600	305	160	1371	57
406948	BAW6	2202	11625	260	108	1390	1
406bbb	BAW541	2766	11825	278	317	356	12
478742			12975	368	239	47	0
896342	ETD19	3220	13200	306	203	1077	0
405a46	BAW14R	5675	21025	381	97	1503	1
406af2		7464	22325	360	0	94	38
4ca33d		1146	26000	344	327	594	1
491d84	NJE354W	0347	27050	448	128	1582	42
4ca294	EIN634	2022	33025	420	119	469	0
4ca216	EIN40X	4036	36000	416	296	981	4
4ca1fe		4426	38000	445	281	150	0

Written by: Sopwith
Revision 1.0
December 29, 2014
sopwith@ismellsmoke.net

Introduction

Since I was a little boy I have had a fascination with airplanes. I am an active RC airplane modeler and I have a radio scanner that I use to listen to air traffic control conversations. Imagine my surprise when I came across a very interesting article in the December 2014 issue of LINUX Magazine titled, "*Plane Spotting*."

Written by Charly Kuhnast, the one-page article describes how to use a USB DVB-T device to capture airplane traffic and plot it on a Google map. What a cool idea. Ol' Sopwith decided to purchase a DVB-T and see if it would work on the Raspberry Pi.

A few Google searches and I determined the use of the DVB-T device is quite popular among Linux users. And, of course, somebody has already figured out how to get the plane spotter app working on a Pi. The most helpful article to me was written by David Taylor titled, "ADS-B dump1090."

This Raspberry Pi "*How-To*" provides detailed instructions on how to get the plane spotter application up and running on a Pi. If you are in a hurry to get going, use David's article to determine how to install the required software and configure the system.

If you are a new Raspberry Pi user and prefer clear instructions on how to get things working – this article is for you.

Airplane Tracking

Almost all commercial airplanes around the globe transmit information about themselves on radio frequency 1090 MHz. Known as the *Automatic Dependent Surveillance-Broadcast System* (ADS-B), an airplanes heading, speed, altitude, GPS position, flight number, and squawk ID are repeatedly broadcast. This means anyone with the right radio receiver can receive these signals from airplanes within reach of the receivers antenna.

With the amazing advancement of software defined radio (SDR) in recent years combined with the dirt cheap availability of USB SDR devices, tracking airplanes and plotting them on a map is a relatively simple process. The trick is to get the right Linux drivers installed and working.

Required hardware

The hardware requirements to get you tracking airplanes are minimal. You will need three devices:

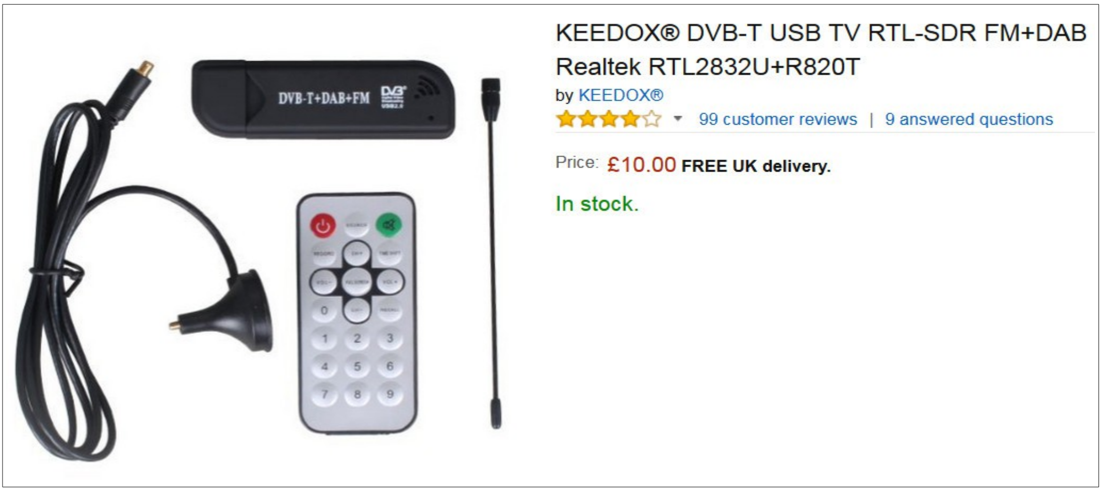
1. A Raspberry Pi that can communicate with the Internet
2. A powered USB hub
3. A USB DVB-T device.

Any version of Raspberry Pi will work. For this "*How-To,*" I used a Rev-1 Pi with a usb wireless network adapter. I installed *Raspian* and updated it. I also overclocked the Pi to "High" and enabled SSH so I can manage it remotely.

I purchased a 7-port powered USB hub from Staples a while ago that has served me and my Pi's well for over two years. You *must* use a powered USB hub for this project. The DVB-T device will not get enough power from the Pi if you try to use one of the Pi's USB ports to power it.

I ordered a DVB-T USB device from Amazon in the UK since I live in London. It is hard to believe this device was only £10 delivered to my front door. The device includes a mini infrared remote-control and an external antenna. This device is highly capable. It can receive FM radio, high-def terrestrial television signals as well as the 1009 MHz. signal we will receive from airplanes overhead.

Be sure to order the correct device for the area you live in. This particular device will not work in the US if you are planning on watching terrestrial TV channels.



KEEDOX® DVB-T USB TV RTL-SDR FM+DAB Realtek RTL2832U+R820T
by KEEDOX®
99 customer reviews | 9 answered questions
Price: £10.00 FREE UK delivery.
In stock.

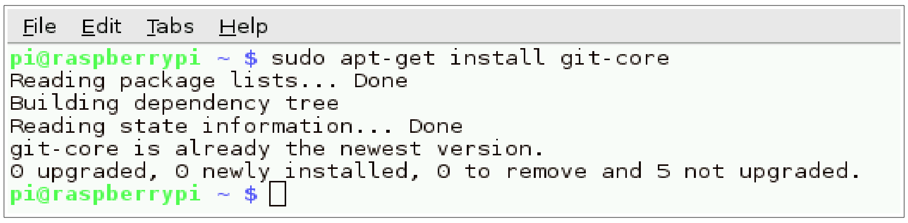
When my DVB-T device was delivered, I am ashamed to say that I installed the included software on Mrs. Sopwith's Windows 7 laptop to see if the device provided good high definition TV service. The software setup requires a “scan” to determine all the available channels. To my amazement, the device found 88 television and FM radio stations. Wow. The terrestrial high definition streaming video was fantastic. I was so impressed I ordered two more devices. The Mrs. surely enjoys her new high-def TV that was once just a Windows laptop.

Step-by-Step

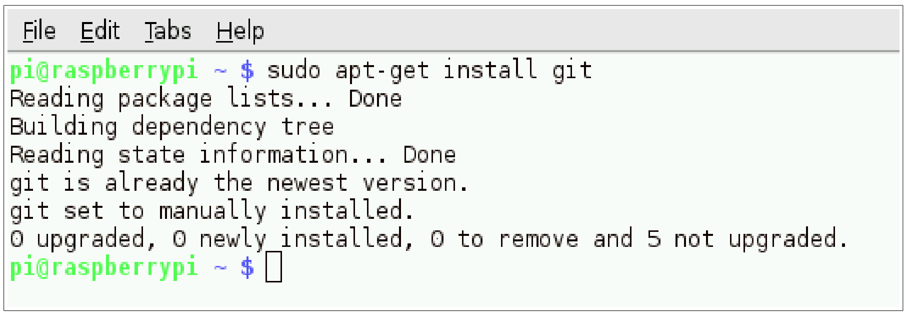
In this section I will walk you through all the steps of getting the airplane tracking system up and running on your Pi. These are pretty much the exact same steps describe in David Taylor's blog.

1) **Be sure your Pi has all the latest patches**
   Using a terminal window enter the following commands: $sud apt-get update
   $sudo apt-get upgrade

2) **Install git**



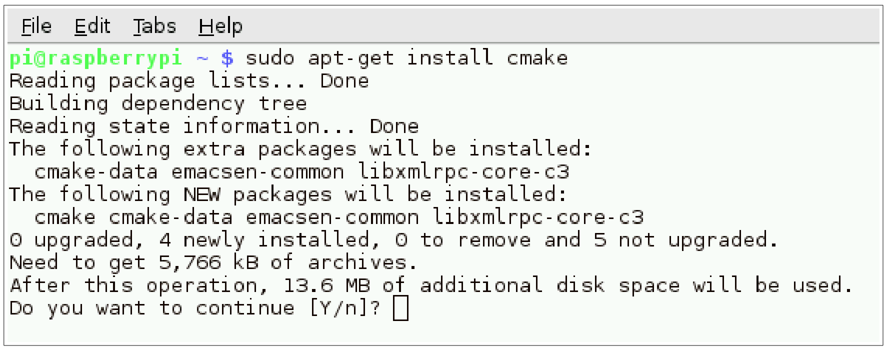
```
pi@raspberrypi ~ $ sudo apt-get install git-core
Reading package lists... Done
Building dependency tree
Reading state information... Done
git-core is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
pi@raspberrypi ~ $
```



```
pi@raspberrypi ~ $ sudo apt-get install git
Reading package lists... Done
Building dependency tree
Reading state information... Done
git is already the newest version.
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
pi@raspberrypi ~ $
```

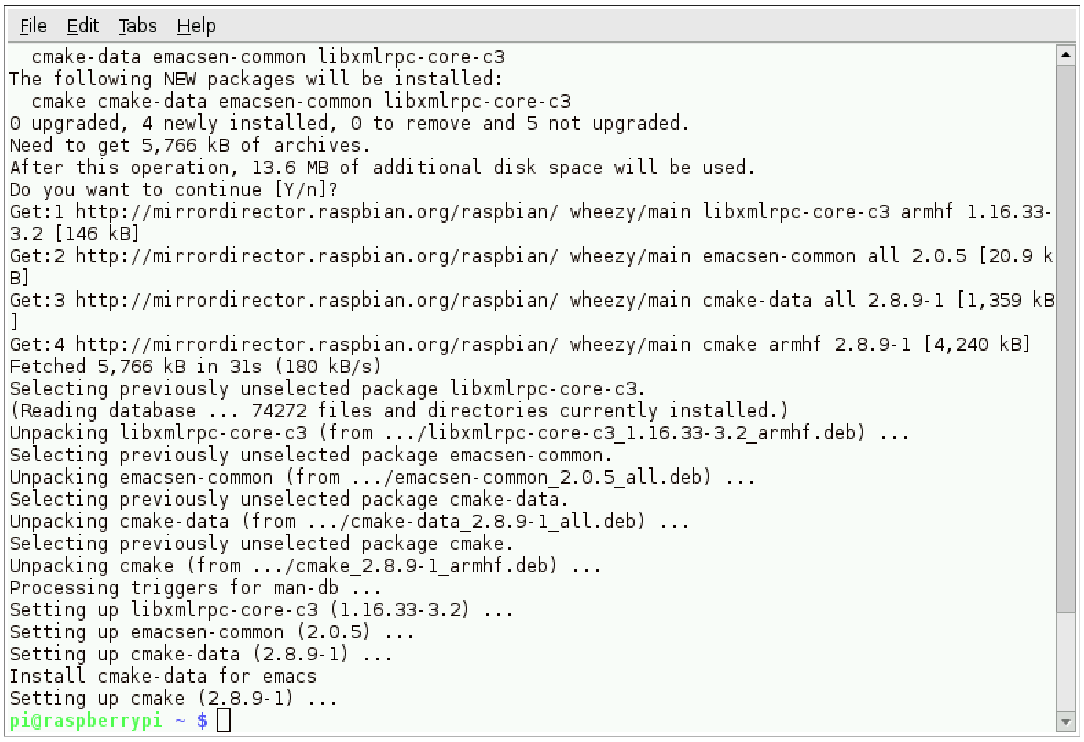
You can see in the above screen shots that my Pi already has git installed.

3) Install cmake



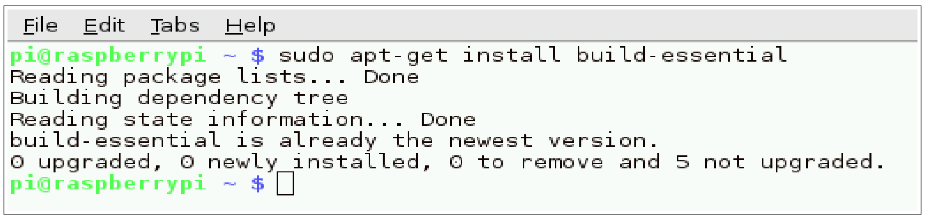
```
pi@raspberrypi ~ $ sudo apt-get install cmake
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  cmake-data emacsen-common libxmlrpc-core-c3
The following NEW packages will be installed:
  cmake cmake-data emacsen-common libxmlrpc-core-c3
0 upgraded, 4 newly installed, 0 to remove and 5 not upgraded.
Need to get 5,766 kB of archives.
After this operation, 13.6 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

Answer 'Y' to the prompt and cmake will be installed.



```
  cmake-data emacsen-common libxmlrpc-core-c3
The following NEW packages will be installed:
  cmake cmake-data emacsen-common libxmlrpc-core-c3
0 upgraded, 4 newly installed, 0 to remove and 5 not upgraded.
Need to get 5,766 kB of archives.
After this operation, 13.6 MB of additional disk space will be used.
Do you want to continue [Y/n]?
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libxmlrpc-core-c3 armhf 1.16.33-
3.2 [146 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main emacsen-common all 2.0.5 [20.9 k
B]
Get:3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main cmake-data all 2.8.9-1 [1,359 kB
]
Get:4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main cmake armhf 2.8.9-1 [4,240 kB]
Fetched 5,766 kB in 31s (180 kB/s)
Selecting previously unselected package libxmlrpc-core-c3.
(Reading database ... 74272 files and directories currently installed.)
Unpacking libxmlrpc-core-c3 (from .../libxmlrpc-core-c3_1.16.33-3.2_armhf.deb) ...
Selecting previously unselected package emacsen-common.
Unpacking emacsen-common (from .../emacsen-common_2.0.5_all.deb) ...
Selecting previously unselected package cmake-data.
Unpacking cmake-data (from .../cmake-data_2.8.9-1_all.deb) ...
Selecting previously unselected package cmake.
Unpacking cmake (from .../cmake_2.8.9-1_armhf.deb) ...
Processing triggers for man-db ...
Setting up libxmlrpc-core-c3 (1.16.33-3.2) ...
Setting up emacsen-common (2.0.5) ...
Setting up cmake-data (2.8.9-1) ...
Install cmake-data for emacs
Setting up cmake (2.8.9-1) ...
pi@raspberrypi ~ $
```

4) Install build-essential

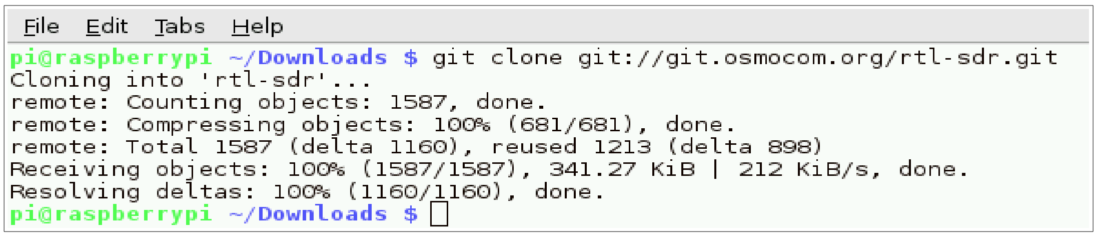


```
pi@raspberrypi ~ $ sudo apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
pi@raspberrypi ~ $
```

Build-essential is already installed on my Pi.

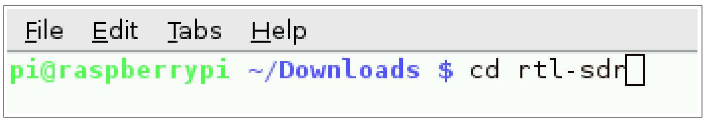
5) Install *rtl-sdr* from the git repository

When you clone a git repository, git will create a folder in your current working directory. I created a directory named 'Downloads' and moved to that directory prior to the *rtl-sdr* download.



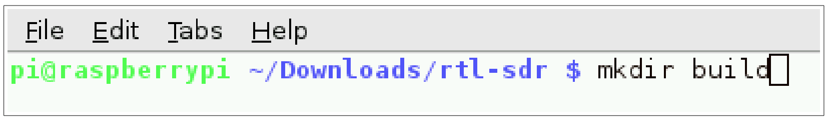
```
pi@raspberrypi ~/Downloads $ git clone git://git.osmocom.org/rtl-sdr.git
Cloning into 'rtl-sdr'...
remote: Counting objects: 1587, done.
remote: Compressing objects: 100% (681/681), done.
remote: Total 1587 (delta 1160), reused 1213 (delta 898)
Receiving objects: 100% (1587/1587), 341.27 KiB | 212 KiB/s, done.
Resolving deltas: 100% (1160/1160), done.
pi@raspberrypi ~/Downloads $ 
```

6) Change your current working directory to *rtl-sdr*



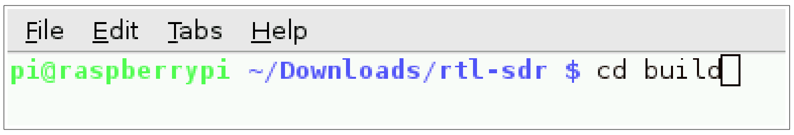
```
pi@raspberrypi ~/Downloads $ cd rtl-sdr
```

7) Make a directory named '*build*.'



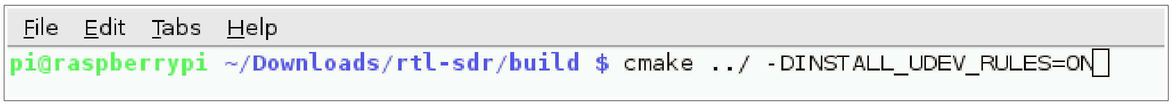
```
pi@raspberrypi ~/Downloads/rtl-sdr $ mkdir build
```

8) Change directories to *build*

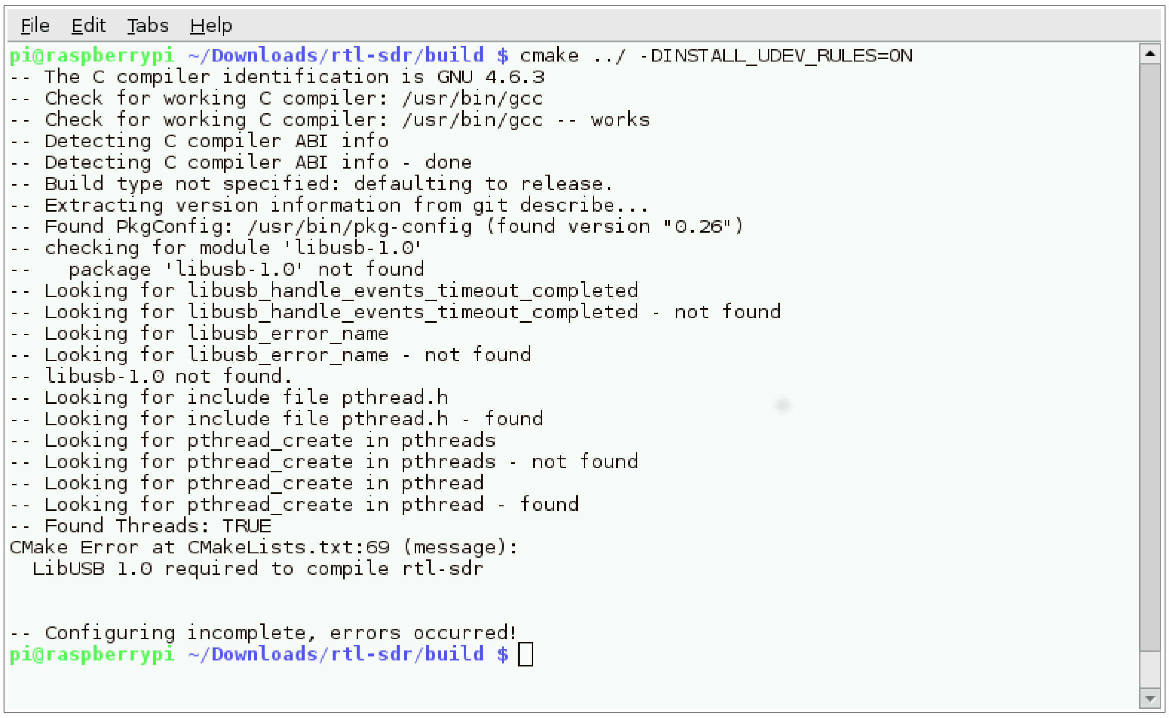


```
pi@raspberrypi ~/Downloads/rtl-sdr $ cd build
```

9) Create the build environment for *rtl-sdr* using cmake



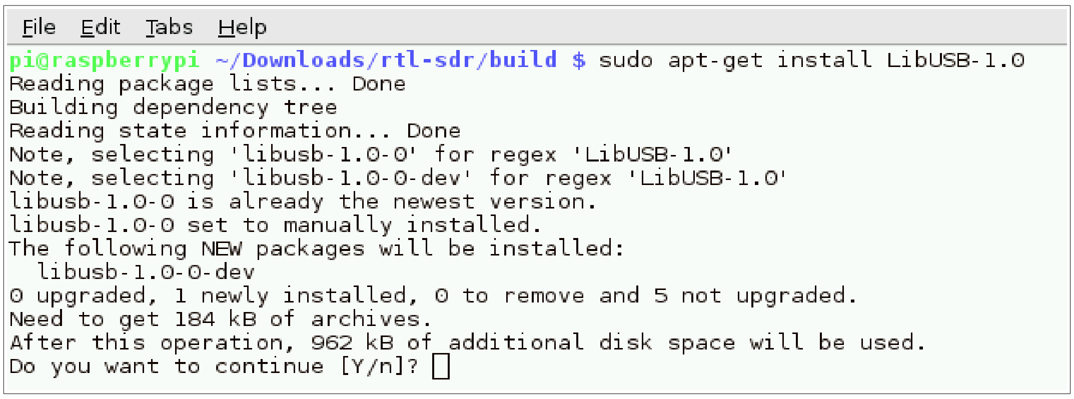
```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ cmake ../ -DINSTALL_UDEV_RULES=ON
```



```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ cmake ../ -DINSTALL_UDEV_RULES=ON
-- The C compiler identification is GNU 4.6.3
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Build type not specified: defaulting to release.
-- Extracting version information from git describe...
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.26")
-- checking for module 'libusb-1.0'
--   package 'libusb-1.0' not found
-- Looking for libusb_handle_events_timeout_completed
-- Looking for libusb_handle_events_timeout_completed - not found
-- Looking for libusb_error_name
-- Looking for libusb_error_name - not found
-- libusb-1.0 not found.
-- Looking for include file pthread.h
-- Looking for include file pthread.h - found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
CMake Error at CMakeLists.txt:69 (message):
  LibUSB 1.0 required to compile rtl-sdr


-- Configuring incomplete, errors occurred!
pi@raspberrypi ~/Downloads/rtl-sdr/build $ 
```

Notice that I got a compile error. The error message is telling me that LibUSB 1.0 is required to compile *rtl-sdr*. New users of the Pi are often frustrated by compiler and install errors. Most error listings will tell you exactly what the problem is as in the case here.

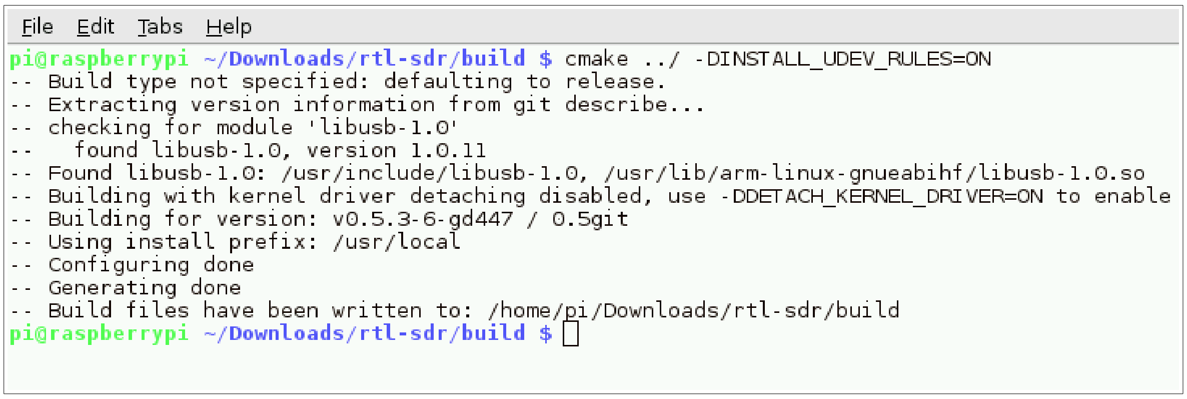
10) Install *LibUSB 1.0*



```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ sudo apt-get install LibUSB-1.0
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'libusb-1.0-0' for regex 'LibUSB-1.0'
Note, selecting 'libusb-1.0-0-dev' for regex 'LibUSB-1.0'
libusb-1.0-0 is already the newest version.
libusb-1.0-0 set to manually installed.
The following NEW packages will be installed:
  libusb-1.0-0-dev
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 184 kB of archives.
After this operation, 962 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

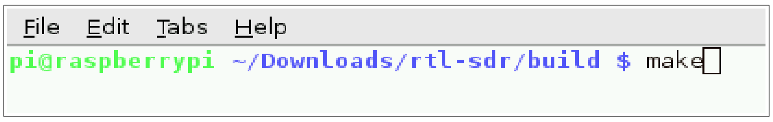
Answer 'Y' to the prompt and *LibUSB* will be installed.

11) Create the *rtl-sdr* build environment again using cmake.
You can see in the figure below the creation of the build environment for *rtl-sdr* completed successfully. We can now compile *rtl-sdr*.

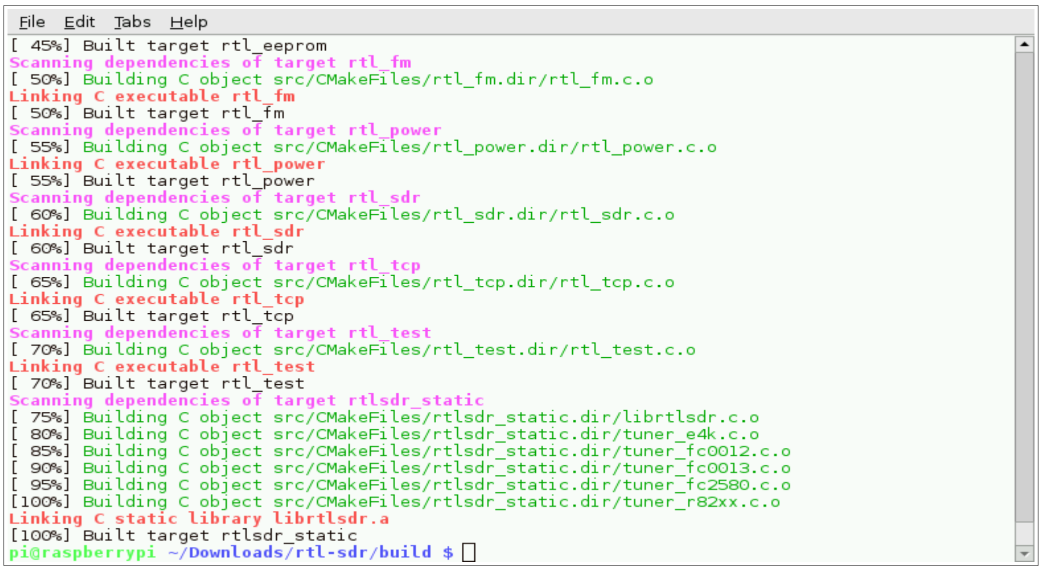


```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ cmake ../ -DINSTALL_UDEV_RULES=ON
-- Build type not specified: defaulting to release.
-- Extracting version information from git describe...
-- checking for module 'libusb-1.0'
--   found libusb-1.0, version 1.0.11
-- Found libusb-1.0: /usr/include/libusb-1.0, /usr/lib/arm-linux-gnueabihf/libusb-1.0.so
-- Building with kernel driver detaching disabled, use -DDETACH_KERNEL_DRIVER=ON to enable
-- Building for version: v0.5.3-6-gd447 / 0.5git
-- Using install prefix: /usr/local
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/Downloads/rtl-sdr/build
pi@raspberrypi ~/Downloads/rtl-sdr/build $
```

12) Compile *rtl-sdr*



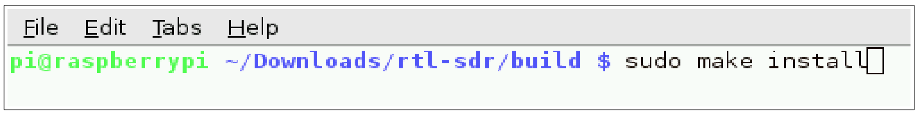
```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ make
```



```
[ 45%] Built target rtl_eeprom
Scanning dependencies of target rtl_fm
[ 50%] Building C object src/CMakeFiles/rtl_fm.dir/rtl_fm.c.o
Linking C executable rtl_fm
[ 50%] Built target rtl_fm
Scanning dependencies of target rtl_power
[ 55%] Building C object src/CMakeFiles/rtl_power.dir/rtl_power.c.o
Linking C executable rtl_power
[ 55%] Built target rtl_power
Scanning dependencies of target rtl_sdr
[ 60%] Building C object src/CMakeFiles/rtl_sdr.dir/rtl_sdr.c.o
Linking C executable rtl_sdr
[ 60%] Built target rtl_sdr
Scanning dependencies of target rtl_tcp
[ 65%] Building C object src/CMakeFiles/rtl_tcp.dir/rtl_tcp.c.o
Linking C executable rtl_tcp
[ 65%] Built target rtl_tcp
Scanning dependencies of target rtl_test
[ 70%] Building C object src/CMakeFiles/rtl_test.dir/rtl_test.c.o
Linking C executable rtl_test
[ 70%] Built target rtl_test
Scanning dependencies of target rtlsdr_static
[ 75%] Building C object src/CMakeFiles/rtlsdr_static.dir/librtlsdr.c.o
[ 80%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_e4k.c.o
[ 85%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_fc0012.c.o
[ 90%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_fc0013.c.o
[ 95%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_fc2580.c.o
[100%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_r82xx.c.o
Linking C static library librtlsdr.a
[100%] Built target rtlsdr_static
pi@raspberrypi ~/Downloads/rtl-sdr/build $
```

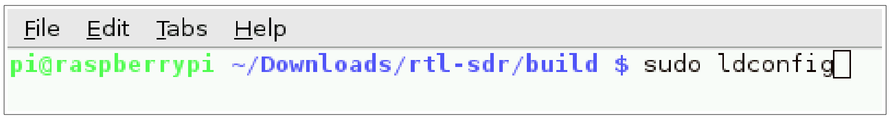
You can see in all the neon-color glory that *rtl-sdr* compiled successfully.

13) Install *rtl-sdr*



```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ sudo make install
```

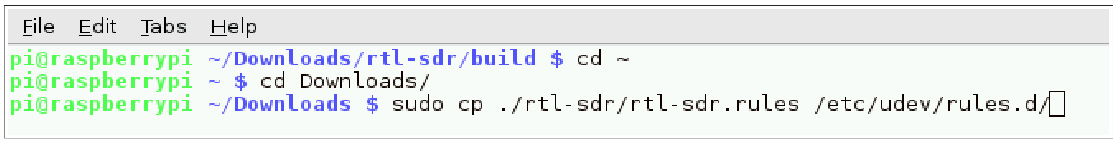
14) Configure linker run-time binaries



```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ sudo ldconfig
```

15) Install udev rules

Change directories so you are just above the rtl-sdr directory. In my case, this is the 'Downloads' directory. Next, copy the rtl-sdr.rules file to the /etc/udev/rules.d directory.



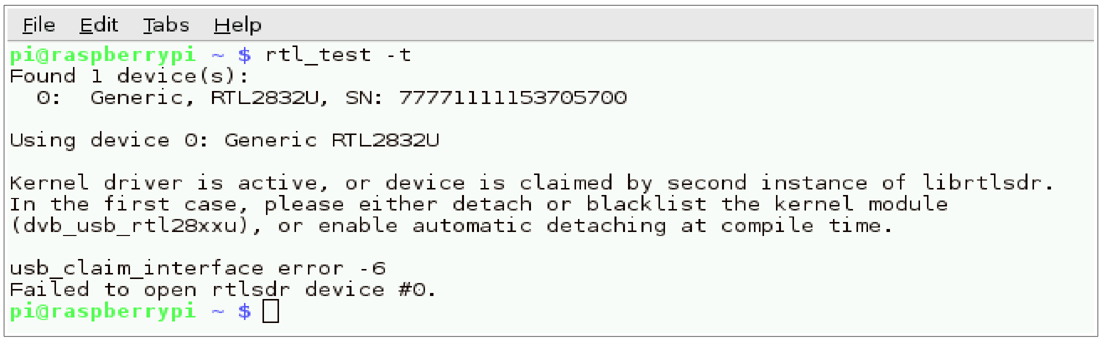
```
pi@raspberrypi ~/Downloads/rtl-sdr/build $ cd ~
pi@raspberrypi ~ $ cd Downloads/
pi@raspberrypi ~/Downloads $ sudo cp ./rtl-sdr/rtl-sdr.rules /etc/udev/rules.d/
```

16) Reboot your Pi: $ sudo reboot

17) Plug your DVB-T device into your powered USB hub

18) Test the *rtl-sdr* drivers

Enter the command shown in the below screen shot.



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

Using device 0: Generic RTL2832U

Kernel driver is active, or device is claimed by second instance of librtlsdr.
In the first case, please either detach or blacklist the kernel module
(dvb_usb_rtl28xxu), or enable automatic detaching at compile time.

usb_claim_interface error -6
Failed to open rtlsdr device #0.
pi@raspberrypi ~ $ 
```

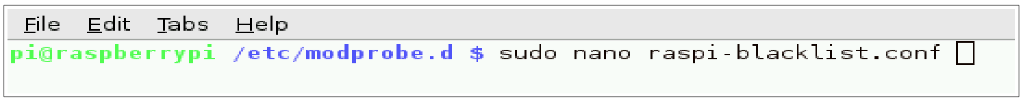
Here we have a good-news - bad-news scenario. My DVB-T device was found. You can even see the serial number of the device. So, we know the device is working. The bad news is that the kernel could not access the device. As always in the Linux world – do not panic. Simple read what the error message is telling you.

The kernel message says the device is already in use by another driver (dvb_usb_rtl28xxu). It is also advising that we detach (unload) the offending kernel module or blacklist it so it is not automatically loaded.

What happened here is that the kernel automatically plug-n-played the correct drivers for the DVB-T device when it was plugged in. This included the necessary drivers to enable the watching of high def TV stations. Since this is not our intended use of the device, we need to advise the kernel not to load this driver.

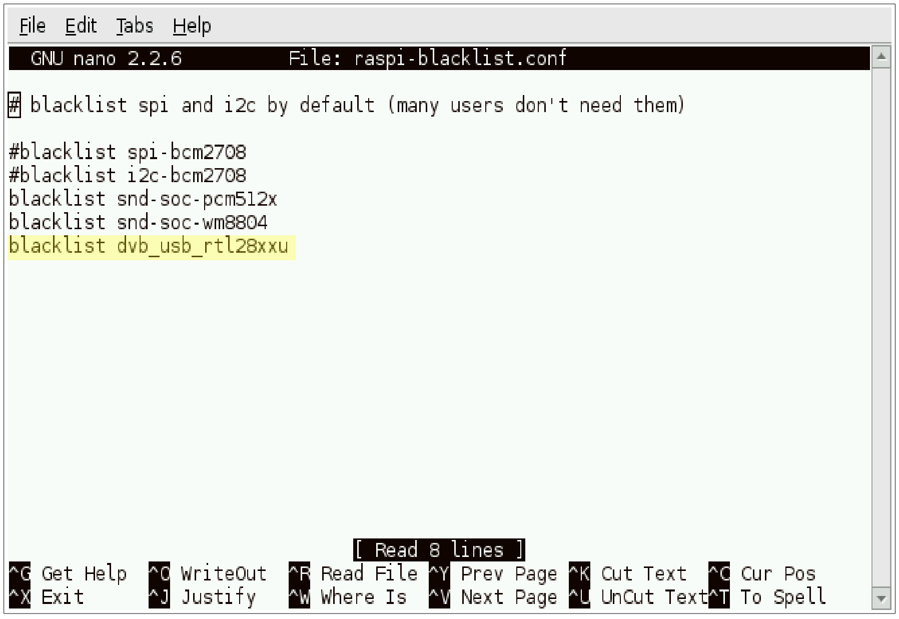
19) Blacklist the dvb_usb_rtl28xxu driver

Change directories to /etc/modprobe.d. Now, edit the raspi-blacklist.conf file using nano. (See command below).



```
pi@raspberrypi /etc/modprobe.d $ sudo nano raspi-blacklist.conf 
```

Add the line highlighted in yellow in the below screen shot.



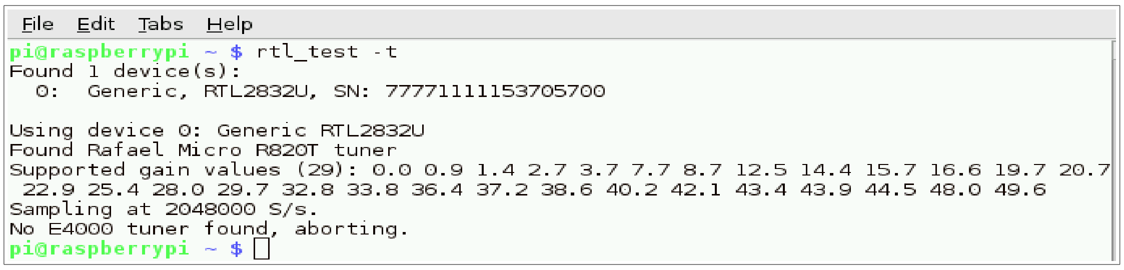
```
  GNU nano 2.2.6            File: raspi-blacklist.conf

# blacklist spi and i2c by default (many users don't need them)

#blacklist spi-bcm2708
#blacklist i2c-bcm2708
blacklist snd-soc-pcm512x
blacklist snd-soc-wm8804
blacklist dvb_usb_rtl28xxu








                              [ Read 8 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Press <Ctrl>O then <Enter> to save your changes to the file. Press <Ctrl> X <Enter> to exit nano. The line you just added tells the Linux kernel not to load the dvb_usb_rtl28xxu device driver when it finds your DVB-T device.

Reboot you Pi: *$ sudo reboot*.

20) Test the *rtl-sdr* drivers

Once your Pi reboots, run the rtl_test using the below command.

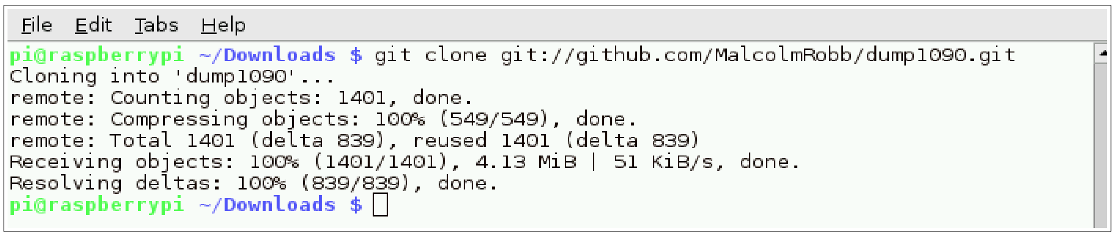


```
pi@raspberrypi ~ $ rtl_test -t
Found 1 device(s):
  0:  Generic, RTL2832U, SN: 77771111153705700

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 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.
No E4000 tuner found, aborting.
pi@raspberrypi ~ $ 
```

You can see in the above screen shot. my *rtl-sdr* driver found my device and was able to sample its gain values. You can safely ignore the message about the E4000 tuner. At this point we have the *rtl-sdr* driver working correctly. Now it is time to install the dump1090 application.

21) Install *dump1090*

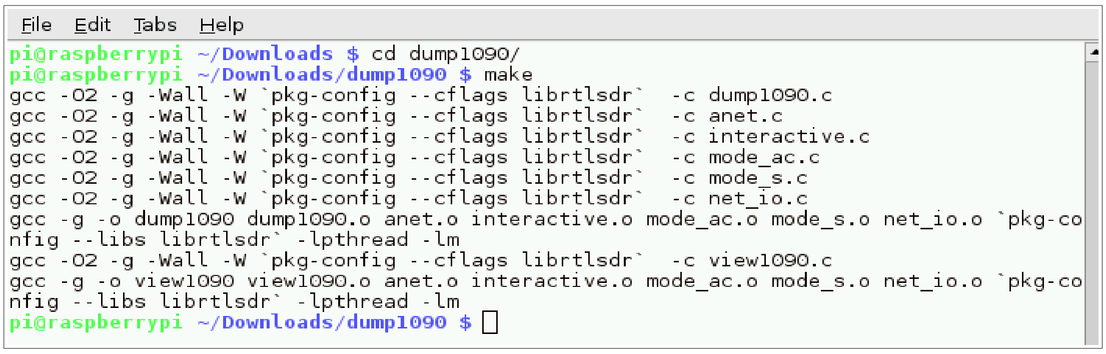
Change directories to your home directory. In my case I moved to my *Downloads* directory. Download the *dump1090* from github.



```
pi@raspberrypi ~/Downloads $ git clone git://github.com/MalcolmRobb/dump1090.git
Cloning into 'dump1090'...
remote: Counting objects: 1401, done.
remote: Compressing objects: 100% (549/549), done.
remote: Total 1401 (delta 839), reused 1401 (delta 839)
Receiving objects: 100% (1401/1401), 4.13 MiB | 51 KiB/s, done.
Resolving deltas: 100% (839/839), done.
pi@raspberrypi ~/Downloads $ 
```

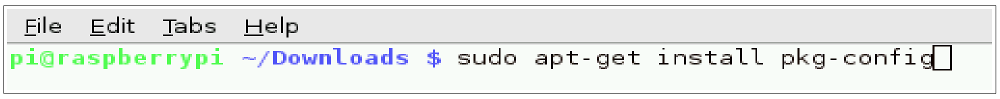
22) Make *dump1090*

Change directories to where github placed the *dump1090* directory and type make as shown in in the below screen shot.



```
pi@raspberrypi ~/Downloads $ cd dump1090/
pi@raspberrypi ~/Downloads/dump1090 $ make
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c dump1090.c
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c anet.c
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c interactive.c
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c mode_ac.c
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c mode_s.c
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c net_io.c
gcc -g -o dump1090 dump1090.o anet.o interactive.o mode_ac.o mode_s.o net_io.o `pkg-co
nfig --libs librtlsdr` -lpthread -lm
gcc -O2 -g -Wall -W `pkg-config --cflags librtlsdr`  -c view1090.c
gcc -g -o view1090 view1090.o anet.o interactive.o mode_ac.o mode_s.o net_io.o `pkg-co
nfig --libs librtlsdr` -lpthread -lm
pi@raspberrypi ~/Downloads/dump1090 $ 
```

You can see above that my compile completed successfully. If you get an error complaining that *pkg-config* is not installed, you can correct the problem by installing it.



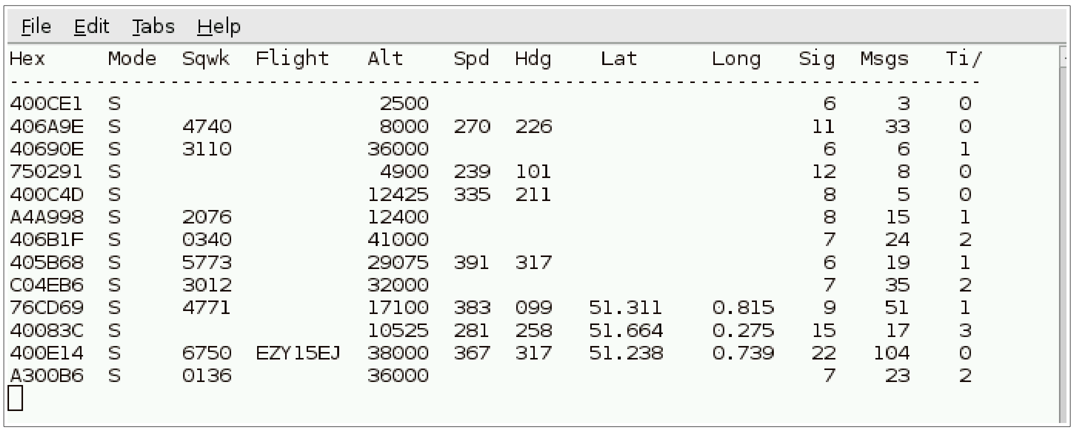
```
pi@raspberrypi ~/Downloads $ sudo apt-get install pkg-config
```

Once *pkg-config* is in place, recompile the *dump1090* application.

Finally, we have everything we need to track airplanes. Let's test the *dump1090* application.

23) Test *dump1090*

To run a simple test to ensure the *dump1090* application works, enter the below command from the *dump1090* folder:
$ ./dump1090 --interactive
If everything is configured correctly you should see some airplane data appear.



```
Hex     Mode  Sqwk  Flight   Alt    Spd  Hdg    Lat      Long   Sig  Msgs   Ti/
---------------------------------------------------------------------------------
400CE1  S                    2500                                 6     3    0
406A9E  S     4740           8000   270  226                     11    33    0
40690E  S     3110          36000                                 6     6    1
750291  S                    4900   239  101                     12     8    0
400C4D  S                   12425   335  211                      8     5    0
A4A998  S     2076          12400                                 8    15    1
406B1F  S     0340          41000                                 7    24    2
405B68  S     5773          29075   391  317                      6    19    1
C04EB6  S     3012          32000                                 7    35    2
76CD69  S     4771          17100   383  099   51.311    0.815    9    51    1
40083C  S                   10525   281  258   51.664    0.275   15    17    3
400E14  S     6750  EZY15EJ 38000   367  317   51.238    0.739   22   104    0
A300B6  S     0136          36000                                 7    23    2
```

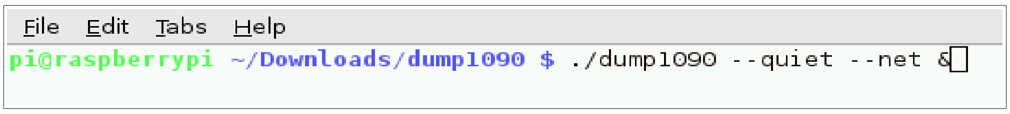
Wow – how cool is that?

If you get an error message – do not panic. Simply follow the trail of the messages you see from the app. Troubleshooting is part of the fun of Linux. Go back through the previous steps and make sure you did not miss a step.

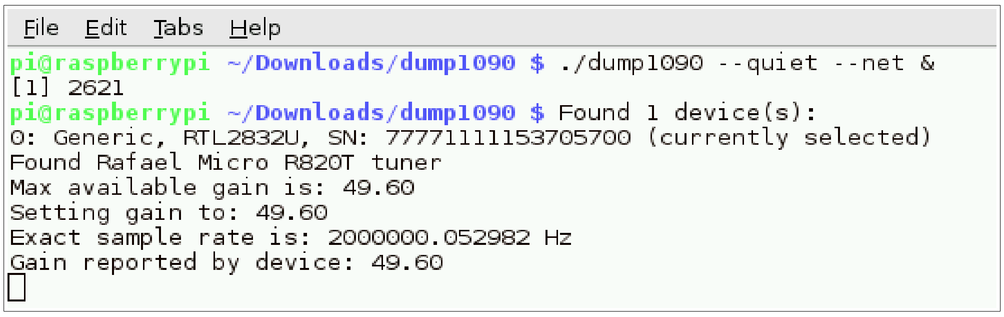
To exit the test press <Ctrl> C

24) Start the *dump1090* web service

The real fun of this project is watching airplanes from a web browser. To do this, simply enter the command shown in the below screen shot. The & at the end of the command tells Linux to run the application in the background.



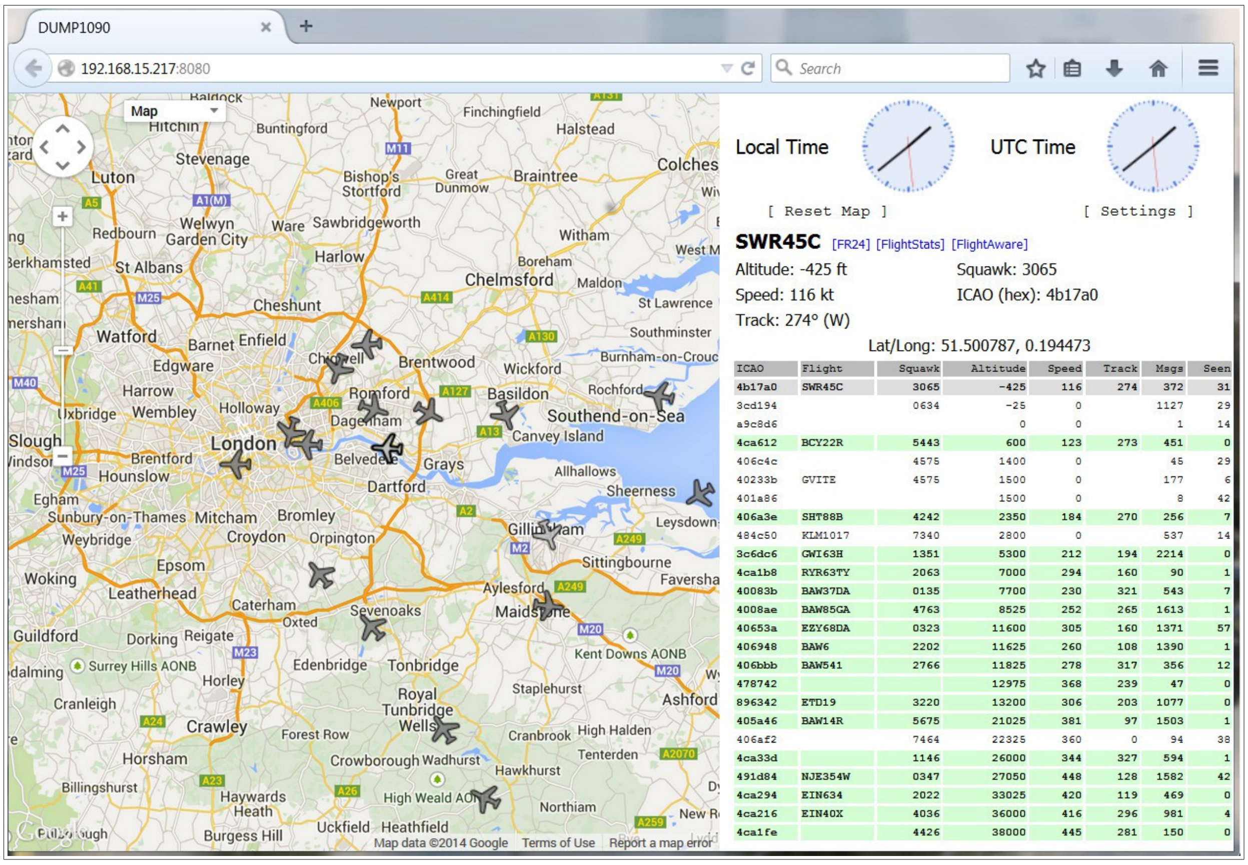
```
pi@raspberrypi ~/Downloads/dump1090 $ ./dump1090 --quiet --net &
```



```
pi@raspberrypi ~/Downloads/dump1090 $ ./dump1090 --quiet --net &
[1] 2621
pi@raspberrypi ~/Downloads/dump1090 $ Found 1 device(s):
0: Generic, RTL2832U, SN: 77771111153705700 (currently selected)
Found Rafael Micro R820T tuner
Max available gain is: 49.60
Setting gain to: 49.60
Exact sample rate is: 2000000.052982 Hz
Gain reported by device: 49.60
```

Open up a browser on your Pi or on another computer on your network. The Pi is a great web server for this app but you may have trouble getting the *Epiphany* web browser to behave properly. In my case I fired up Firefox on my Mrs. Sopwiths' laptop and pointed it to the IP address of my Pi. (http://192.168.15.217:8080). Do not forget to include the colon followed by

8080 since this is the port the web server is listening on.



SWR45C [FR24] [FlightStats] [FlightAware]

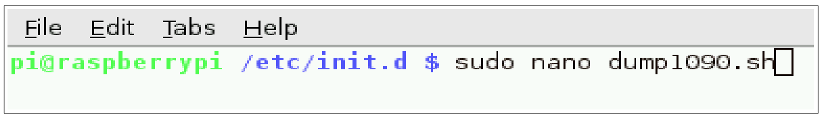
Altitude: -425 ft Squawk: 3065
Speed: 116 kt ICAO (hex): 4b17a0
Track: 274° (W)

Lat/Long: 51.500787, 0.194473

ICAO	Flight	Squawk	Altitude	Speed	Track	Msgs	Seen
4b17a0	SWR45C	3065	-425	116	274	372	31
3cd194		0634	-25	0		1127	29
a9c8d6			0	0		1	14
4ca612	BCY22R	5443	600	123	273	451	0
406c4c		4575	1400	0		45	29
40233b	GVITE	4575	1500	0		177	6
401a86			1500	0		8	42
406a3e	SHT88B	4242	2350	184	270	256	7
484c50	KLM1017	7340	2800	0		537	14
3c6dc6	GWI63H	1351	5300	212	194	2214	0
4ca1b8	RYR63TY	2063	7000	294	160	90	1
40083b	BAW37DA	0135	7700	230	321	543	7
4008ae	BAW85GA	4763	8525	252	265	1613	1
40653a	EZY68DA	0323	11600	305	160	1371	57
406948	BAW6	2202	11625	260	108	1390	1
406bbb	BAW541	2766	11825	278	317	356	12
478742			12975	368	239	47	0
896342	ETD19	3220	13200	306	203	1077	0
405a46	BAW14R	5675	21025	381	97	1503	1
406af2		7464	22325	360	0	94	38
4ca33d		1146	26000	344	327	594	1
491d84	NJE354W	0347	27050	448	128	1582	42
4ca294	EIN634	2022	33025	420	119	469	0
4ca216	EIN40X	4036	36000	416	296	981	4
4ca1fe		4426	38000	445	281	150	0

25) Automatically start *dump1090* at bootup (Optional)

If you want the *dump1090* web server to start as a service on your Pi at start up you will need to install a small start up script. Change directories to /etc/init.d and enter the command shown in the below screen shot.



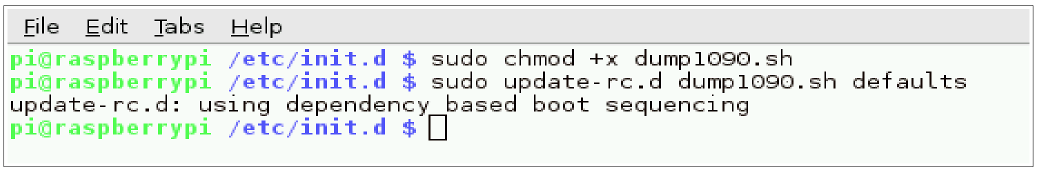
```
pi@raspberrypi /etc/init.d $ sudo nano dump1090.sh
```

Copy and paste the script on the next page into *nano*. Be sure to change the PROG_PATH line in the script to point the folder you installed *dump1090* (Line .15).

Save the file (<Ctrl>O) and exit nano <Ctrl>X.

```
#!/bin/bash
### BEGIN INIT INFO
#
# Provides:           dump1090
# Required-Start:     $remote_fs
# Required-Stop:      $remote_fs
# Default-Start:      2 3 4 5
# Default-Stop:              0 1 6
# Short-Description:  dump1090 initscript

#
### END INIT INFO
## Fill in name of program here.
PROG="dump1090"
PROG_PATH="/home/pi/Downloads/dump1090"
PROG_ARGS="--interactive --net --net-ro-size 500  --net-ro-rate 5"
PIDFILE="/var/run/dump1090.pid"

start() {
      if [ -e $PIDFILE ]; then
          ## Program is running, exit with error.
          echo "Error! $PROG is currently running!" 1>&2
          exit 1
      else
          ## Change from /dev/null to something like /var/log/$PROG if you want to save output.
          cd $PROG_PATH
          ./$PROG $PROG_ARGS 2>&1 >/dev/null &
          echo "$PROG started"
          touch $PIDFILE
      fi
}

stop() {
      if [ -e $PIDFILE ]; then
          ## Program is running, so stop it
         echo "$PROG is running"
         killall $PROG
         rm -f $PIDFILE
         echo "$PROG stopped"
      else
          ## Program is not running, exit with error.
          echo "Error! $PROG not started!" 1>&2
          exit 1
      fi
}

## Check to see if we are running as root first.
## Found at http://www.cyberciti.biz/tips/shell-root-user-check-script.html
if [ "$(id -u)" != "0" ]; then
      echo "This script must be run as root" 1>&2
      exit 1
fi

case "$1" in
      start)
          start
          exit 0
      ;;
      stop)
          stop
          exit 0
      ;;
      reload|restart|force-reload)
          stop
          start
          exit 0
      ;;
      **)
          echo "Usage: $0 {start|stop|reload}" 1>&2
          exit 1
      ;;
esac
exit 0
```

When you are back at the command prompt, enter the command shown in the screen shot. below. The first command makes the script file to executable. The second command updates the rc script to include *dump1090* at start up.



```
pi@raspberrypi /etc/init.d $ sudo chmod +x dump1090.sh
pi@raspberrypi /etc/init.d $ sudo update-rc.d dump1090.sh defaults
update-rc.d: using dependency based boot sequencing
pi@raspberrypi /etc/init.d $
```

Summary

At this point you should have a working plane tracker running on your Pi. Congratulations!

This project has been a great deal of fun. Who would have thought for £10 I could purchase a USB device that can capture airplane radio signals and map them on a Google map.

I live in a flat in East London not too far from the London City airport. I have learned so much about London flights using my flight tracker. My flat is very close to the approach pattern of inbound planes to Heathrow. It is such a joy to be able to predict when a plane will be passing overhead and then determine so much about the flight.

I hope you enjoy this project as well.

Oh – and one more thing. This project would be a great way to get kids working with a Raspberry Pi. Buy a kid a DVB-T and have some fun.

Sopwith

NOTE: This document was created entirely on a Raspberry PI using LibreOffice *Writer*. The screen shots were captured with *Scrot.* The screen-shots were edited with the *Gimp*. Windows not required.