

AirSpy Server on the Raspberry Pi 3

Whitham D. Reve

1. Introduction

The diminutive and inexpensive Raspberry Pi computer platform may be used as a server for the AirSpy receiver. The server allows remote access and control of the receiver by a client PC running SDR# (figure 1). The hardware and software aspects of the server installation are covered here. The server application software is called *spyserver* or *AirSpy Server* in this document and is available as a free download from the receiver manufacturer. It supports multiple simultaneous client connections over a local area network or the Internet. The advantage of the server is its ability to reduce the bandwidth requirements of the access connection.

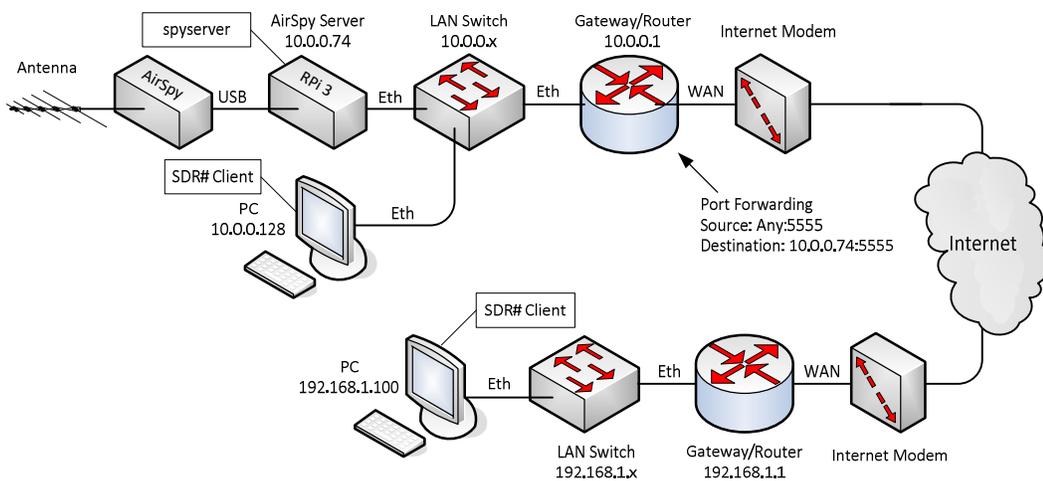


Figure 1 ~ Block diagram of the AirSpy Server and Clients. The IP addresses 10.0.0.x and 192.168.1.x are non-routable addresses used in the respective local area networks. The only software required besides the operating systems are the AirSpy Server, associated libraries and SDR# application.

The AirSpy (figure 2) is a popular and relatively low cost (169 USD) software defined radio receiver that has been used in radio astronomy applications since its introduction in 2014. The AirSpy's native frequency range is 24 to 1800 MHz with up to 10 MHz wide spectrum view. For lower frequencies the manufacturer has an up-converter called SpyVerter (49 USD) that extends the range down to 1 kHz and the AirSpy HF+ (199 USD) that has a range of 9 kHz to 31 MHz and 60 to 260 MHz. The SpyVerter is accommodated in the server by frequency offset and activation of the receiver's built-in bias-tee for powering. Separate setups are needed for the original AirSpy and AirSpy HF+.



Figure 2 ~ AirSpy receiver (left) and SpyVerter (right). The two modules may be connected by an SMA-M to SMA-M coupler (middle) or coaxial cable. The SpyVerter is powered by the receiver through internal bias-tees. Dimensions of each module are 53 x 40 x 25 mm not including the SMA-F RF connectors.

The AirSpy receiver uses a USB port for power and control. Before the server application software became available, the receiver needed to be close to a desktop or laptop PC running the control software such as SDR#. This is not an optimum operational setup because of possible radio frequency interference and long antenna cable lengths. Another problem is that remote LAN access to the AirSpy is impeded or impossible because of the high

network bandwidth requirements. Using the Raspberry Pi (RPI) as a server allows the receiver and RPI to be placed closer to the antenna or farther from sources of RFI yet still easily controlled. The RPI server powers the receiver and performs initial signal processing to greatly reduce the network connection bandwidth. The RPI itself requires power and network connections so there still is some cabling involved (figure 3). These potentially are sources of RFI but presumably are more manageable than a long RF connection to an antenna.

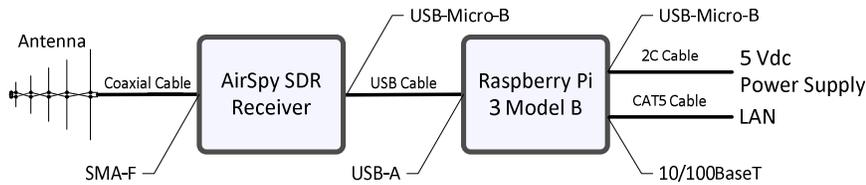


Figure 3 ~ AirSpy receiver and Raspberry Pi cabling. Because the micro-USB-B power connector on the RPI is so small, the wire size in the 2-conductor power cable is necessarily small.

Receivers compatible with spyserver: AirSpy R0 (original AirSpy receiver), AirSpy R2 (2nd generation), AirSpy HF+, AirSpy Mini and RTL-SDR. The AirSpy Server also supports heterodyne converters such as the SpyVerter up-converter. For information on the AirSpy receivers see {[AirSpy](#)}. For this article, I used only the AirSpy R0 and R2 with the server. Note: On 5 March 2018 AirSpy announced that the tuner integrated circuit (IC) used in the AirSpy R2 and Mini was discontinued and these receivers no longer will be manufactured. They also indicated that a replacement may be developed using a different tuner IC. Presumably, any new receiver product will be compatible with the spyserver.

Spyserver installations: When spyserver is installed as discussed in the following sections, it is setup for only one basic receiver type. For example, the spyserver can be installed for the R0 and R2. If the RPI hardware is to be used later with the HF+, the drivers for the HF+ must be installed on the RPI memory card. I do not know if the drivers for the R0/R2 and HF+ coexist. In the worst case, if both receiver types are to be used, two separate RPI installations may be needed. It is a simple matter to install the spyserver software on a memory card to be used for the R0 and R2 and on another memory card to be used for the HF+. If desired, two RPis, one with spyserver for the R0 or R2 and another for the HF+, can be operated simultaneously on a local area network. Each spyserver will have its own network address.

2. Hardware Installation

Raspberry Pi: I successfully installed and operated the spyserver on the Raspberry Pi 3 model B (RPI3), Raspberry Pi 3 model B+ (RPI3+) and Raspberry Pi 2 model B (RPI2), but I suspect the RPI2 does not support as many simultaneous connections as the RPI3 or RPI3+. I did not attempt to install spyserver on the other Raspberry Pi platforms (Zero, A, B and B+) and I did not run as many tests on the RPI2 and RPI3+ as on the RPI3. The RPI3+ was newly released in mid-March 2018 and I only had time to verify operation and did not do extensive testing. From here on, I will refer only to the RPI3.

No other hardware components are needed except those described in this section. The RPI3 is setup and operated in a “headless” mode in which no display, keyboard or mouse is connected to the RPI3. All installation activity is through a PC on the same LAN as the RPI3 using a command line interface (*console*) such as PuTTY or Tera Term. In this application the RPI3 is dedicated to the single purpose of running the server.

Memory card: The operating system and applications software require < 2 GB. I used both 4 GB class 4 and 8 GB class 10 micro-SD memory cards in the RPi3. There is no requirement for data storage on the spyserver.

Power supply: The dc input current to the RPi3 varies with its processing load, which depends on the spyserver digital signal processing settings and number of active connections. Higher spectral bandwidths result in higher processor loads. The RPi3 input current also is increased by loads powered through its USB ports, which includes the AirSpy receiver. It is very important to use the RPi3 with an adequate power supply (5.0 to 5.5 Vdc, 2.0 to 2.5 A) and a high quality power cable. The RPi3 has an onboard 2.5 A resettable PTC fuse and polarity guard, both of which introduce some voltage drop. Excessive voltage drop in the power cable or a weak power supply is manifested as failure to boot or intermittent and unreliable operation (lockup). Generally, inadequate input voltage may be observed on the red Power LED on the RPi3 PCB. If the red LED flashes or goes out completely during the boot process or any other time, the input voltage drop is too high. See section 7 for current and power measurements of the RPi3 while running the AirSpy Server software with active connections.

A short power cable will minimize voltage drop, and a power supply with 5.5 Vdc output will provide some input margin. However, in a remote installation, the power cable necessarily will be relatively long. Cables for the tiny USB-micro-B power connector on the RPi3 have small conductors and could cause voltage drop problems. The largest conductors I have seen are 22 AWG in 3 m long cables from {Volutz}. That length probably is too short for most remote setups. One solution is to splice a short cable with a USB-micro-B connector on one end to a larger gauge cable run to the system power supply (figure 4). The voltage drop in the power cable should be ≤ 0.25 V at a current of 2.0 A. This is equivalent to ≤ 0.125 ohms cable loop resistance or about 5 m of 16 AWG (1.5 mm), 2-conductor cable. To minimize noise, the power cable (both conductors) should be wound several turns through ferrite beads (such as clamshell beads), one at each end of the cable.

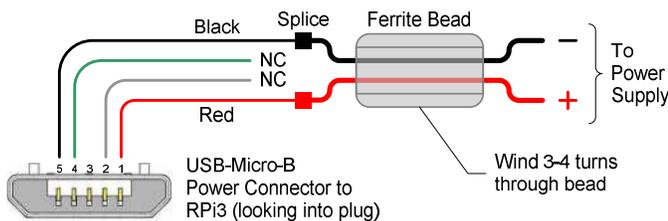


Figure 4a ~ RPi3 power cable consists of larger power supply conductors spliced to the red and black conductors in the USB-micro-B power adapter cable. A ferrite bead is placed at each end of the cable. The data conductors on pins 2 and 4 of the USB connector are not used and should be tied or cut back.



Figure 4b ~ USB-Micro-B power adapter cable. The cable is cut from a regular 300 mm long USB adapter cable, which in this case has a right-angle USB-micro-B connector (upper-right). The yellow/black PowerPole connectors (upper-left) are a splice point for connection to a larger power supply cable. In my station power systems, all connectors are color-coded and yellow/black indicates 5 V.

Cooling: The RPi3 processor runs hot when the AirSpy Server has active connections unless it is externally cooled. If the RPi3 processor gets too hot, it automatically reduces its clock. I do not know if this guarantees against device failure or reduced life caused by overheating but the reduced clock likely will adversely affect performance. In any case, I recommend using passive and active cooling. For passive cooling, small heatsinks are

available for installation on the CPU system on a chip (SoC), USB/Ethernet controller and RAM integrated circuits (IC). These usually are stick-on types held with what should be thermally conductive adhesive. The available heatsinks are copper or aluminum, but copper is more thermally conductive and provides better cooling than aluminum.

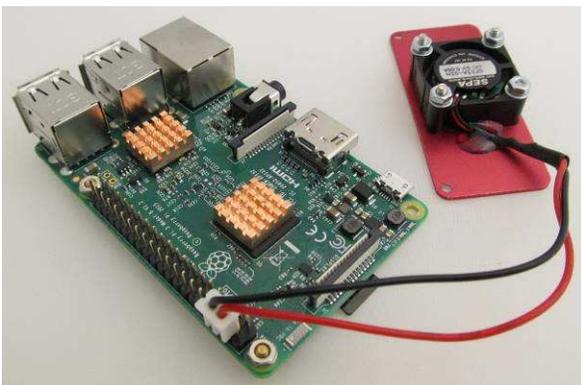
There are several active cooling methods available including small fans that clamp directly to the RPi3 PCB and enclosures that have a preinstalled fan. The fans usually are rated 5 Vdc and can be powered directly from the RPi3 GPIO connector pins 4 and 6 (5 V and GND, respectively). Note: It may be tempting to connect the fan to the RPi3 internal 3.3 V bus (GPIO pins 1 and 6) to reduce its speed and therefore noise; however, the 3.3 V bus is not designed for a fan load, and fan noise is hardly noticeable.

I built two systems, one with fan only and another with fan and heatsinks. I acquired both enclosures through eBay (figure 5). AliExpress is another source. Using an enclosure without a fan is not recommended because of heat build-up. See section 7 for comparative temperature measurements using these cooling methods.



Figure 5 ~ RPi cooling using passive and active methods.

Upper: Two examples of extruded aluminum enclosures for the Air Spyservers shown with an Airspy R2. The left enclosure has a 25 mm fan mounted on the top part of the clamshell enclosure, but the RPi2 PCB inside does not have heatsinks. The right enclosure is slightly larger and has a 25 mm fan mounted on one end panel. The fans blow warmed air out through slots in the enclosures. I found the right enclosure to be quite fiddly and the fan end panel interfered with the RPi3 memory card (the enclosure should be at least 2 mm longer). I ended up removing the two lower fan screws so the end panel could close completely as shown here. I later replaced the 10 mm thick fan with one that is only 7 mm thick, which allows shorter mounting screws with sufficient clearance.



Lower: RPi3 PCB with copper pin-finned heatsinks shown on the CPU (near center of PCB) and USB/Ethernet controller integrated circuits (next to USB connectors at top-left of image). Not shown is the RAM IC on the bottom, which uses a flat plate heatsink. This set of heatsinks costs about 1 USD including shipping. Also shown is the 5 V fan and end panel associated with the enclosure; the fan is connected to the GPIO connector pins 4 and 6.

Network and USB connectivity: The RPi3 has one 10/100 Mb s⁻¹ Ethernet port and four USB 2.0 ports. The USB and Ethernet interfaces are handled by a single integrated circuit. The Ethernet port on the RPi3 connects to the SoC through the internal USB 2.0 bus. The RPi3+ uses a GigE (1000BaseT) Ethernet connector and interface but

still is limited by the internal USB 2.0 bus speeds. However, there supposedly is some gain in Ethernet interface performance with speeds approaching 300 Mb s^{-1} . The RPi3 and RPi3+ also have built-in WiFi capability (2.4 GHz, 802.11n) but I did not attempt to use it. WiFi is unlikely to work very well except when compression is used on the streaming data (described in section 6). If the RPi3 is installed in a metallic enclosure, the built-in WiFi will not work at all.

The AirSpy receiver has one USB 2.0 port. Therefore, the digital rate between the AirSpy receiver and RPi3 is at most 480 Mb s^{-1} and the rate between the RPi3 and local area network is at most 100 Mb s^{-1} (approximately 300 Mb s^{-1} for the RPi3+). These are interface speeds. Payload speeds are considerably lower, 1/10 to at best 1/2 the interface speeds. For trouble-free operation, it is imperative to use high-quality cables for connectivity including cables for the USB, Ethernet and antenna connections.

3. Software Installation

The AirSpy Server software installation involves downloading and installing the operating system and several application and library files. Follow the steps below in order. Note: If the spyserver is setup for the AirSpy R0 and R2 it will not work with the AirSpy HF+; similarly, if it is setup for the AirSpy HF+ it will not work with the R0 or R2. Therefore, the following steps diverge slightly depending on the receiver to be used.

It will take longer to read through this document than to actually install the software, which requires around 10 to 15 minutes. Spyserver v2.0.1629 was the version available at the time this document was written. The software can be considered in development and is not yet perfect. Section 6 includes some operating notes.

Operating system: Install the operating system as described in *Raspberry Pi Basic Setup* {[RPiBasic](#)}. My installation is based on the versions of Raspbian Stretch Lite {[Stretch](#)} that was available in January and March 2018. Note: If Raspbian Stretch Lite is superseded by a later version, the server software may not work unless it, too, has been updated to be compatible. Presumably, compatibility information will be provided in the spyserver release notes provided by the manufacturer.

The default User *pi* should be used when logging into the RPi3 and provisioning the spyserver. The default Password is *raspberry* and it must be changed if the spyserver is to be connected to the internet.

The host name should be changed to something more easily recognized on the LAN (for example, *spyserver*). Procedures for the password and host name changes are described in the basic setup document referenced above. For best trouble-free operation, the spyserver should be setup with a fixed IP address. Although it may be configured for a static IP address as described in {[Static](#)}, LAN management is much easier if the IP address is reserved in the LAN router DHCP server, in which case the router will always assign the same IP address when a DHCP request comes from the RPi3.

Make note of the RPi3 IP address from the basic setup or find it using NetScan, Advanced IP Scanner or AngryIP (figure 6). The default host name is *raspberrypi* and it will appear as such in the IP address scan unless it has been changed as mentioned above.

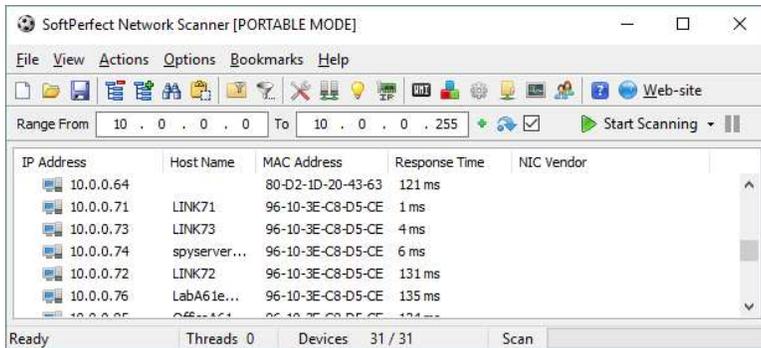


Figure 6 ~ Screenshot of NetScan showing the Raspberry Pi spyserver at IP address 10.0.0.74. This image was acquired after the host name of the RPi3 was changed to *spyserver*.

Upgrade the operating system to the latest available components. Using PuTTY or other terminal emulator, login to the RPi3 and at the prompt enter:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

Spyserver application software: The application software includes the spyserver and associated libraries. Many command line entries below are easily handled by copying from this document one line at a time and pasting into the PuTTY prompt. See [\[RPiBasic\]](#) for copy/paste procedures. However, if a command returns an error it may be because a character is not coded properly by Word when written to the PDF. In this case, simply hand-type the command.

Download and install the latest version of the required libraries:

```
sudo apt-get install build-essential cmake libusb-1.0-0-dev pkg-config
```

Note: If the above command returns an error about cmake not being installed or available, run the following command and then try again:

```
sudo apt-get update
```

At this point, the procedures diverge slightly for the AirSpy R0 and R2 and the AirSpy HF+. Follow steps 1. and 2. only for the AirSpy R0 and R2 and follow steps 3. and 4. only for the AirSpy HF+.

AirSpy R0 and R2:

1. Download and compile the airspyone driver software:

```
wget https://github.com/airspy/airspyone_host/archive/master.zip
unzip master.zip
cd airspyone_host-master
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

2. Clean out temporary files while in the `airspyone_host-master/build` directory:

```
rm -rf *
```

AirSpy HF+:

3. Download and compile the airspyhf driver software:

```
wget https://github.com/airspy/airspyhf/archive/master.zip
unzip master.zip
cd airspyhf-master
mkdir build
cd build
cmake ../ -DINSTALL_UDEV_RULES=ON
make
sudo make install
sudo ldconfig
```

4. Clean out temporary files while in the airspyhf-master/build directory:

```
rm -rf *
```

Follow the remaining steps for both the AirSpy R0 and R2 and the AirSpy HF+.

Prepare a directory for the spyserver application:

```
mkdir ~/spyserver
```

Download and extract the airspy server software package to the new directory:

```
wget https://airspy.com/downloads/spyserver-arm32.tgz
tar xvf spyserver-arm32.tgz -C ~/spyserver
```

Note: The second command above may not work properly when copied/pasted from the PDF file because of improper ASCII coding of the dash character – in –C. Simply hand-type the command being sure to use upper-case C.

Double-check that two files are installed in the spyserver directory:

```
cd ~/spyserver
ls
```

Two files should be shown: **spyserver** and `spyserver.config`

While in the spyserver directory, change permissions on the spyserver file to make it executable:

```
sudo chmod +x spyserver
```

Install the GNU Compiler Collection, GCC-5, and reboot

```
sudo apt-get install -y gcc-5
sudo reboot
```

Note: If GCC-5 is not available, try GCC-6 (replace gcc-5 with gcc-6 in the first command above)

4. Configuring and Running AirSpy Server

This section describes a basic setup that can be used to verify proper software installation. Additional operational details are given in sections 5 and 6. The AirSpy receiver does not need to be connected to the RPi3 server until later in this section but there is no harm in connecting it now.

Configure the airspy server: Connect to the RPi using PuTTY or other terminal emulator. After logging in, change to the spyserver directory and edit the configuration file:

```
cd ~/spyserver
sudo nano spyserver.config
```

Scroll down to the `bind_host` parameter and change the IP address shown there to the RPi3 IP address determined in section 3 above (figure 7). If this is not done correctly, when SDR# attempts to connect, it will display an error window that indicates spyserver refused the connection (figure 8). If the IP address of the RPi3 is changed for any reason, such as replacing the RPi3 hardware, the `bind_host` parameter must be updated with the new IP address.

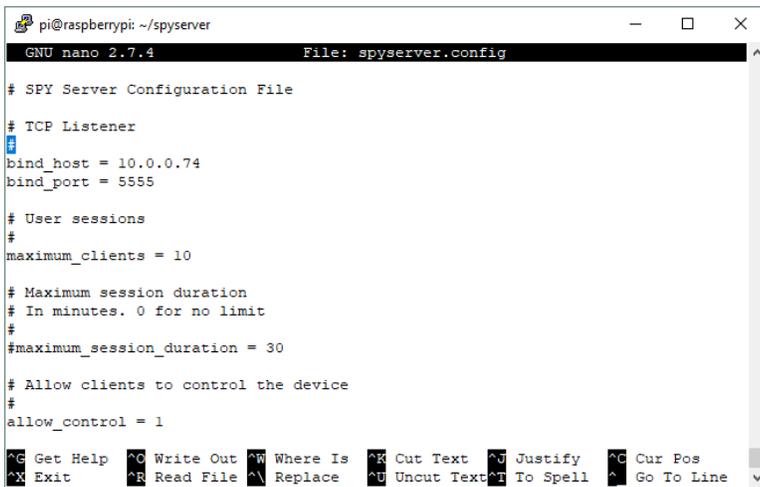


Figure 7 ~ Screenshot of the AirSpy Server configuration file showing the IP address assigned to the `bind_host` parameter. In this example, the address is 10.0.0.74.

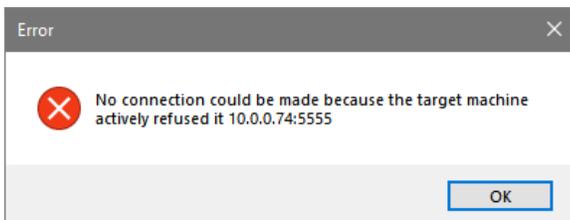


Figure 8 ~ SDR# error window indicating that the `bind_host` parameter does not match the RPi3 spyserver IP address.

The default configuration uses the AirSpyOne device (AirSpy R0, R2 or Mini). If a different receiver is to be used, the configuration must be changed. Scroll down to the `device_type` parameter and change to the desired receiver type, selecting from the list indicated.

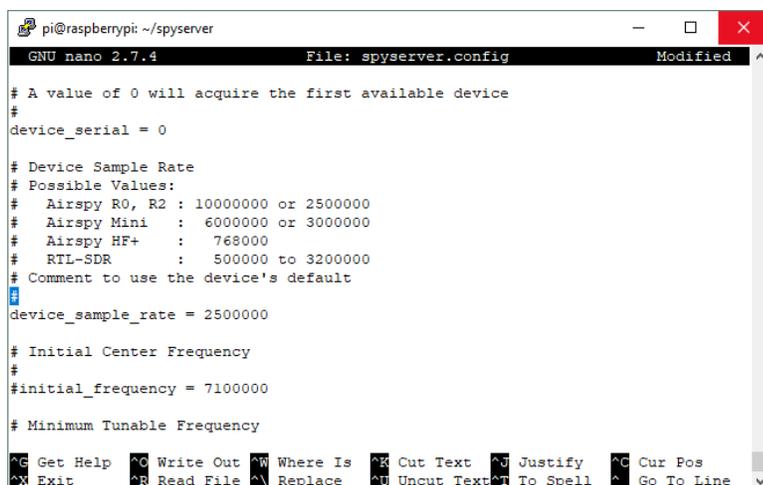
```
# Device Type
# Possible Values:
#   AirspyOne (R0, R2, Mini)
#   AirspyHF+
#   RTL-SDR
```

```
#
device_type = AirspyOne [change to desired receiver type]
```

Scroll down to the line `#device_sample_rate` and delete the hash character `#` to uncomment the line and then change as necessary for the receiver `device_type` set above.

```
# Device Sample Rate
# Possible Values:
#   Airspy R0, R2 : 10000000 or 2500000
#   Airspy Mini  : 6000000 or 3000000
#   Airspy HF+   : 768000
#   RTL-SDR      : 500000 to 3200000
# Comment to use the device's default
#
device_sample_rate = 2500000 [change to the rate required by the device]
```

For the AirSpy R0 and R2, change to `2 500 000` without spaces between the numbers (spaces are shown here for clarity) (figure 9). This sets the sample rate to $2.5 \text{ Msample s}^{-1}$, which allows a displayed spectrum bandwidth of 2 MHz. This value can be increased later to `10 000 000` (without spaces) or $10 \text{ Msample s}^{-1}$ for a displayed bandwidth of 8 MHz but the lower value should be used for initial setup and testing.



```
pi@raspberrypi: ~/spyserver
GNU nano 2.7.4 File: spyserver.config Modified
# A value of 0 will acquire the first available device
#
device_serial = 0

# Device Sample Rate
# Possible Values:
#   Airspy R0, R2 : 10000000 or 2500000
#   Airspy Mini  : 6000000 or 3000000
#   Airspy HF+   : 768000
#   RTL-SDR      : 500000 to 3200000
# Comment to use the device's default
#
device_sample_rate = 2500000

# Initial Center Frequency
#
#initial_frequency = 7100000

# Minimum Tunable Frequency

^G Get Help ^C Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^N Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figure 9 ~ Another screenshot of PuTTY showing the AirSpy Server configuration file in which the `device_sample_rate` parameter has been uncommented (hash mark `#` deleted) and changed to 2500000 samples per second.

When finished with all the changes to the configuration file, save the changes and exit the editor (CTRL-X, Y, Enter).

Physical connections: At this point, the software installation is ready for testing. Connect the receiver to the RPi3 USB port using a high-quality USB-A to USB-micro-B cable, preferably the one supplied with the receiver. Any one of the four RPi3 USB ports may be used.

Run the server: The `spyserver` must be run from the `spyserver` directory. Enter

```
cd ~/spyserver
./spyserver spyserver.config
```

Note: The `spyserver.config` option is the default and may be eliminated from the second command line above (that is, simply enter `./spyserver`).

The PuTTY command line interface should indicate that the spyserver is listening on the IP address configured above and port 5555 (figure 10). The spyserver version is also shown.



Figure 10 ~ Screenshot of PuTTY showing at the bottom the command to run the AirSpy Server using the spyserver.config configuration file. It has successfully connected to the AirSpy receiver and is listening for client connections on port 5555.

SDR# client: Install the latest version of SDR# on the Windows PC to be used as a client {[AirSpyPkg](#)}. Be sure to note and comply with the PC and operating system requirements for SDR# (for example, it does not run on Windows XP). See section 6 for an alternative to SDR# that also is compatible with the spyserver, but initial testing should use SDR#. If a previous version of SDR# already exists on the PC, upgrade to the latest. With The AirSpy Server running, open SDR# on the client PC (figure 11). In the *Sources* dropdown list (upper-left corner of SDR#) select Spy Server (if necessary, first click the Gear icon). Using the format `sdr://x.x.x.x:5555`, enter the IP address and port of the spyserver in the field below the source selection. The default port number is 5555 as defined by the `bind_port` parameter in the configuration file. After entering, press the Connect (C) button to the right of the address:port field. Refer back to the PuTTY interface to see the connection (figure 12). The Connect button is a toggle that can be pressed again to disconnect SDR# at any time.

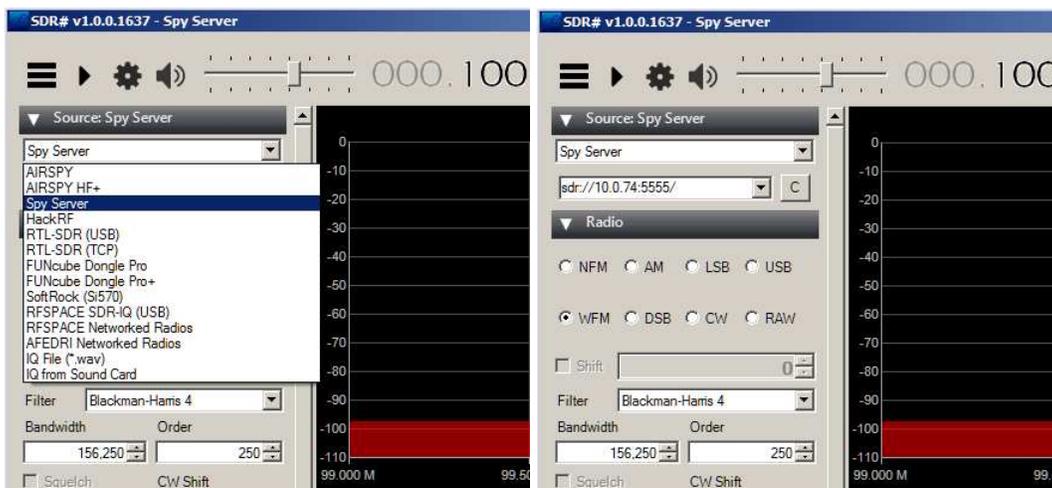


Figure 11 ~ Screenshots of SDR# Source selection dropdown list (left) and IP address and port number (right) using SDR# v1.0.0.1637. SDR# is not yet connected in the right screenshot. When connected, the IP address and port field is grayed out.

As shown in the right image, SDR# is set to demodulate wideband FM (WFM) but connection bandwidth can be greatly reduced by selecting RAW (no demodulation) instead.

```
pi@raspberrypi: ~/spyserver
part of the FQDN) in the /etc/hosts file.
pi@raspberrypi:~/spyserver $ sudo nano spyserver.config
pi@raspberrypi:~/spyserver $ ./spyserver spyserver.config
SPY Server v2.0.1629 - http://airspy.com
Reading the configuration file: spyserver.config
Listening for connections on 10.0.0.74:5555
Accepted client 10.0.0.128:62425 running SDR# v1.0.0.1592 on Microsoft Windows N
T 6.2.9200.0
Device was sleeping. Wake up!
Acquired the device
Changed display pixels for FFT stream => 1000
Changed dB offset for FFT stream => 0
Changed dB range for FFT stream => 127
Changed digital gain for IQ stream => Automatic
Changed format for FFT stream => 1
Changed format for IQ stream => 2
Changed frequency for FFT stream => 100000000 Hz
Changed frequency for IQ stream => 100000000 Hz
Changed decimation for IQ stream => 1/8
Changed format for IQ stream => 2
Changed decimation for IQ stream => 1/16
Changed format for IQ stream => 2
Changed streaming state => 1
```

Figure 12 ~ Screenshot of PuTTY showing a successful client connection to the AirSpy Server. The connected client IP address is 10.0.0.128. Various parameters are exchanged by the SDR# client and spyserver, and these are shown after the connection.

Now click the Play (triangle) icon in upper-left corner of SDR#. At this point the AirSpy receiver is under control of the SDR# client and activity should be indicated in the spectrum window to the right. The Play icon will change to a Stop (square) icon (figure 13). If the AirSpy Server is not running, SDR# will produce an error window (figure 14). Note that the SDR# connection can be established by simply pressing the Play button without first pressing the Connect button. However, pressing the Stop button does not disconnect the SDR# client. Disconnection is only through the Connect button toggle.

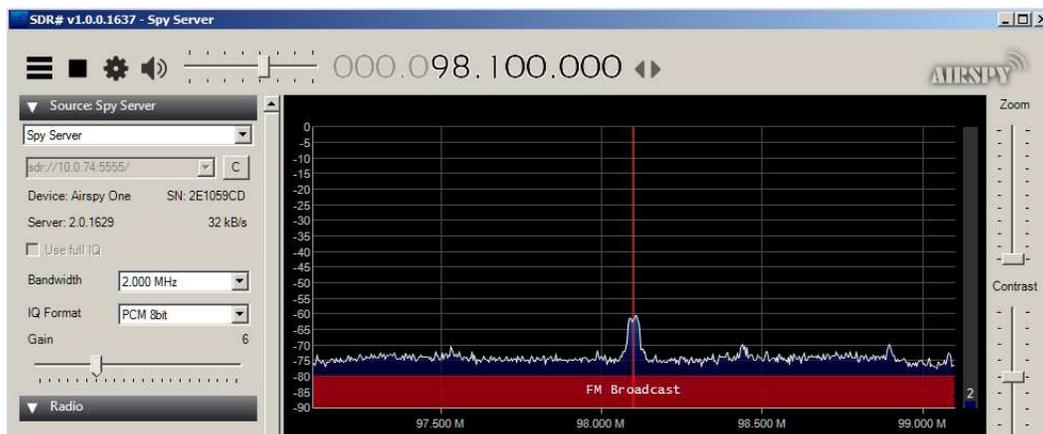


Figure 13 ~ Screenshot of SDR# showing a successful connection by an SDR# client to the AirSpy Server. The displayed bandwidth is 2 MHz and the receiver is tuned to a local FM broadcast station at 98.1 MHz. The Play icon has changed to a Stop icon and the spectrum display window is active. Note also that the server IP address and port fields are grayed out. In this image the Gain control slider is active, indicating it is the first client connection and has control.

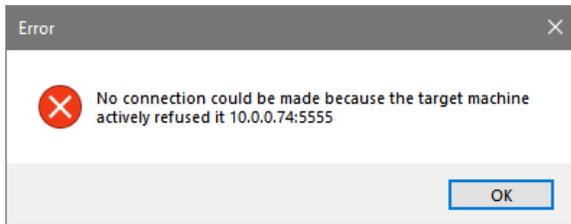


Figure 14 ~ An error window is produced by SDR# if the connection attempt fails either because the server is not running, it is not accessible because of a LAN problem or the wrong IP address is entered in SDR#.

Testing: After installation is completed and when first running and connecting to the server, it is recommended to tune the receiver to the FM broadcast band (88 to 108 MHz) to confirm that the receiver and antenna connections are working properly. Even if the receiver and antenna are to be used in other frequency bands, reception of FM broadcast stations should be possible unless an FM band reject filter is used. For the test images shown here I connected a small telescoping whip antenna directly to the AirSpy RF input jack.

Start and stop the spyserver: For the basic setup described above, first change to the spyserver directory and start spyserver by entering:

```
cd ~/spyserver
./spyserver spyserver.config
```

To stop spyserver, enter at the PuTTY command line session used to run the application:

```
CTRL-c
```

A method is described in the next section that automatically opens the spyserver when the RPi3 starts or is rebooted.

Updates: Over time, the spyserver software will be updated. The update process is simple but it is important to backup the spyserver configuration file. Updating will overwrite the existing spyserver.config file and any user changes will be lost. To update, copy the configuration file to backup:

```
cd ~/spyserver
cp spyserver.config spyserver.back
```

Download the updated spyserver software and extract it:

```
wget https://airspy.com/downloads/spyserver-arm32.tgz
tar xvf spyserver-arm32.tgz -C ~/spyserver
```

Note: Be careful of the dash character in -C if copy/pasting the second command above.

Now, replace the default configuration file with the backup

```
cp spyserver.back spyserver.config
```

5. Automatic Operation as a Service

The basic setup described above requires the AirSpy Server program to be manually opened when the RPi3 is cold started or rebooted. The spyserver can be optionally setup to start automatically. Skip this section if you require only manual operation.

There are a number of ways to setup the software so that the spyserver is automatically opened when the RPi3 is cold started or rebooted. I use the *systemd* daemon manager in Linux to setup the AirSpy Server as a service, which can be enabled to open automatically. Do the following steps if automatic operation is desired:

Use the nano editor to open a new *service unit* configuration file called *spyserver.service*:

```
sudo nano /etc/systemd/system/spyserver.service
```

The editor will open an empty file. Add the following parameters to the file (figure 15):

```
[Unit]
Description=AirSpy Spyserver Service
After=multi-user.target

[Service]
Type=idle
ExecStart=/home/pi/spyserver/spyserver /home/pi/spyserver/spyserver.config
Restart=always

[Install]
WantedBy=multi-user.target
```

Save the changes and exit the editor (CTRL-X, Y, Enter).



Figure 15 ~ Screenshot of the nano editor showing the complete contents of the service Unit file. The *Description* parameter provides a label that is referenced whenever the service is queried or reported. The *Type* parameter requires that the *ExecStart* parameter only be run after all other processes have loaded. The *Restart* parameter is not shown in this screenshot.

Setup the permissions on the service file (readable and writable by all users):

```
sudo chmod 777 /etc/systemd/system/spyserver.service
```

Load the new service file, enable the *spyserver.service* and reboot

```
sudo systemctl daemon-reload
sudo systemctl enable spyserver.service
sudo reboot
```

When the RPi3 reboots, the spyserver will start automatically. The status of the *spyserver.service* can be checked with (figure 16):

```
sudo service spyserver status
```

```

pi@spyspserver:~$ sudo service spyspserver status
● spyspserver.service - AirSpy SpyServer Service
   Loaded: loaded (/etc/systemd/system/spyspserver.service; enabled; vendor preset
   Active: active (running) since Tue 2018-02-06 21:03:34 UTC; 4min 5s ago
   Main PID: 520 (spyspserver)
   CGroup: /system.slice/spyspserver.service
           └─520 /home/pi/spyspserver/spyspserver /home/pi/spyspserver/spyspserver.conf
Feb 06 21:03:34 spyspserver systemd[1]: Started AirSpy SpyServer Service.
lines 1-8/8 (END)

```

Figure 16 ~ Status of the spyspserver.service when running indicates that the service is enabled and Active. It also shows when the service was started, in this case at 21:03:34 on Feb 6.

The spyspserver.service can be stopped, started, and restarted with the following commands (the `stop` command does not survive a reboot):

```

sudo service spyspserver stop
sudo service spyspserver start
sudo service spyspserver restart

```

When the command executes properly, it will return the command line prompt after a moment; otherwise, it will return an error message. The `restart` command is used after changes are made to the spyspserver configuration file. Alternately, before making changes, `stop` the service and then after changes are completed, `start` the service.

The spyspserver.service can be disabled by entering:

```

sudo systemctl disable spyspserver.service

```

To re-enable the spyspserver.service enter:

```

sudo systemctl enable spyspserver.service

```

Both the `disable` and `enable` commands survive a reboot.

After the spyspserver.service has been disabled, the spyspserver can be manually started and stopped as described in the basic setup.

6. Operating Notes

This section applies to spyspserver v2.0.1629 and SDR# v1.0.0.1655. Operating information and some defects are discussed below.

Inconsistent SDR# operation: Some parameters in SDR# when used with spyspserver, for example, *Resolution* in the FFT Display tab and others, may be changed when SDR# is first opened but before spyspserver is connected. Connecting the spyspserver causes these parameters to be grayed out, but they stay grayed out after spyspserver has been disconnected. Closing SDR# and then opening it usually will make the parameter available again. Other parameters do not seem to perform a function when used with spyspserver so some experimentation will be necessary.

USB interference: When the spyspserver is in operation with the RPi2 and RPi3, the spectrum display shows pulsing spikes with a spacing of about 450 kHz without regard to center frequency setting. These spikes rise and fall with a period of about 1 s, seem to come and go and sometimes rise 15 dB above the noise floor to a level near -60 dBFS (dB with respect to Full Scale of the AirSpy analog-digital converter). The pulsing spikes are most

apparent at lower gain settings where the noise floor is lower. I tried many mitigation measures (different power supplies, different cables, ferrite beads, and more) with no success. I found that simply stopping and then starting spyserver changed the pulse levels or eliminated them altogether. Generally, the pulsing spectrum can be reduced by setting the receiver gain above 10; see next item. I noted that this problem does not appear when using the RPi3+, possibly because it has slightly better performance.

Gain setting: The gain setting in SDR# operates differently when used with spyserver than when the AirSpy receiver is connected directly to a PC. When using spyserver, the LNA, mixer and IF gains are linked to a single control in SDR#. Some experimentation will be necessary for best performance. The gain setting used when SDR# first connects to the spyserver is specified by the `initial_gain` parameter in the `spyserver.config` file.

Open the `spyserver.config` file using the nano editor:

```
cd ~/spyserver
sudo nano spyserver.config
```

Now, scroll down to the `initial_gain` parameter and change to 10.

```
# Initial Gain
#
initial_gain = 10
```

After saving the changes (CTRL X, Y, Enter), restart the spyserver.

```
sudo service spyserver restart
```

Operational limits: The spyserver configuration file may be used to specify several operational limits including bandwidth, frequency limits and initial frequency, number of simultaneous connections and session duration. The configuration file includes brief comments for each parameter and experimentation is suggested. Only the first SDR# connection controls the center frequency, span and gain of the AirSpy receiver. Subsequent client connections can tune within the frequency range set by the first client but cannot change any other operational parameter.

Performance: Connection performance can be adjusted with SDR#. When the *Use Full IQ* option in the Source tab of SDR# is checked, the server sends full rate I/Q data from the receiver to the client; however, this requires a lot of network bandwidth (table 1) and probably is not practical over the internet or even over many LANs. Uncheck the *Use Full IQ* option and then use the *IQ Format* option to adjust the number of bits per sample. This effectively increases the data compression and lowers the data rate. During operation the data rate is shown in real-time next to the *Use Full IQ* checkbox and it can be used to compare different *IQ Format* settings. For internet connectivity the *IQ Format* should be set to *PCM 8bit* but there will be some loss in dynamic range.

Performance and data rate also can be adjusted by editing the spyserver configuration file. The `spyserver.config` parameters that may be changed by the user to adjust the data rate are device sample rate (`device_sample_rate`), FFT bin bits (`fft_bin_bits`), FFT frames per second (`fft_fps`) and force 8-bit mode (`force_8bit`). Generally, a reduction in sample rate, FFT bin bits or FFT frame rate will reduce not only the network load but also the span, fidelity or resolution of the spectrum display. The force 8-bit mode in the configuration file overrides the *IQ Format* setting in SDR#. Another parameter in the configuration file that can be adjusted to limit the network bandwidth usage is the number of simultaneous clients or user sessions

(`maximum_clients`). The default is 10 and lower values will limit the number of clients and, thus, limit the network usage.

Table 1 ~ Bit rates displayed in the SDR# client while tuned to a VHF FM broadcast station and 2 MHz bandwidth.

Demodulation	SDR# Setting	Rate
WFM	Full IQ	5.2 Mb s ⁻¹
WFM	Float 32 bit	1.3 Mb s ⁻¹
WFM	PCM 24 bit	961 kb s ⁻¹
WFM	PCM 16 bit	660 kb s ⁻¹
WFM	PCM 8 bit	331 kb s ⁻¹
Raw	PCM 8 bit	32 kb s ⁻¹

Choppy audio can occur on older PCs and slow internet connections. To improve performance in these situations, adjust the FFT frame rate parameter `fft_fps` in the configuration file. The default is 15, and lower values will improve performance at the expense of a more pixelated waterfall display in the SDR# client. Also, increase the buffer parameters `buffer_size_ms` and `buffer_count`. The default settings are 50 and 10, respectively. The best settings will depend on the available network bandwidth between the client and server and other factors. Experimentation is recommended.

Remote access: Users who access the spyserver via the Internet will use the public IP address of the gateway where the spyserver is installed. The entry in the field below the *Sources* drop-down list will use the same format as a local connection; that is, `sdr://xxx.xxx.xxx.xxx:5555`, where in this case `xxx.xxx.xxx.xxx` is the public IP address. The public IP address can be determined from the local router/gateway. See also next item. AirSpy maintains a list of remote spyservers at [{Spyserver}](#).

IP port: The default port number for the spyserver is 5555 but any port number may be used. The port number is changed in the spyserver configuration file (parameter `bind_port`), and it also must be changed in the SDR# client setup. If the AirSpy Server is to be accessed through the internet, the same port must be setup for forwarding in the local internet router (figure 17). If port forwarding is used, the IP address of the AirSpy Server RPi3 should be set to static or reserved in the router as previously described.

Application name	External Port	Internal Port	Protocol	Device IP#	Enabled	
Radio-Sky Spectrograph	3627	3627	Both	10.0.0.29	True	Edit/ Delete
Reeve Cloud-IQ	50000	50000	Both	10.0.0.69	True	Edit/ Delete
RPI-Shake	16022	16022	Both	10.0.0.121	True	Edit/ Delete
RemServe-1	1001	22	TCP	10.0.0.201	True	Edit/ Delete
RemServ-2	1002	1002	TCP	10.0.0.134	True	Edit/ Delete
Spyserver	5555	5555	Both	10.0.0.74	True	Edit/ Delete

Figure 17 ~ Port forwarding setup window in the LinkSys WRT1900AC router/gateway. The bottom entry is for the spyserver on port 5555 and it is directed to 10.0.0.74, the IP address of the RPi3 spyserver. This allows any external client to connect through the router to the spyserver on that specific port. Other routers have an equivalent setup screen.

SpyVerter: The spyserver installs with a default configuration file that is setup for a standalone AirSpy receiver. If the SpyVerter up-converter (or another heterodyne converter) is to be used with the AirSpy R0 or R2 receivers, it is necessary to adjust a few configuration parameters. The configuration may be saved as a uniquely named

file and then specified as a command line parameter when the server is run. This feature enables many different configurations for both the standalone and converted setups.

If the SpyVerter is used, edit the spyserver.configuration file:

```
cd ~/spyserver
sudo nano spyserver.config
```

Adjust the parameter values shown below for the SpyVerter (different values may be needed for other converters). As for the bias-tee parameter, according to AirSpy the bias-tee can supply up to 250 mA at 4.5 Vdc but 50 mA is a more practical limit (the SpyVerter R2 requires only 10 mA). The hash mark # indicates a commented line; delete the mark if necessary.

```
# Initial Center Frequency,  $1000 \leq F_{init} \leq 600000000$  Hz
initial_frequency = 7100000
#
# Minimum Tunable Frequency,  $1000 \leq F_{min}$  Hz
# Comment if using the device default
minimum_frequency = 1000
#
# Maximum Tunable Frequency,  $F_{max} \leq 600000000$  Hz
# Comment if using the device default
maximum_frequency = 35000000
#
# Converter Offset, SpyVerter Foffset = -120000000 Hz positive image
# Comment if not using a converter
converter_offset = -120000000
#
# Bias-Tee, 1=enable 0=disable
# Enable for AirspyOne only - Used for LNAs and SpyVerter
enable_bias_tee = 1
#
```

After making the changes in nano, the configuration can be saved as a new file. For example, to save the file as spyverter1.config, enter the following:

```
CTRL-X
Save modified buffer? (Answering "No" will DISCARD changes.) Y, Enter
File Name to Write [DOS Format]: spyverter1.config Enter
Save file under DIFFERENT NAME? Y, Enter
Enter
```

To run spyserver with the new configuration file spyverter1.config, enter

```
cd ~/spyserver
./spyserver spyverter1.config
```

Timekeeping: By default, the RPi operating system uses Coordinated Universal Time (UTC), and it should be left that way. The Stretch operating system, unlike earlier versions, does not include the full Network Time Protocol (NTP) for timekeeping but uses a scaled-down version called Simple NTP (SNTP). This is a client-only version of NTP that is implemented through the timesync daemon, timesyncd. Timesyncd synchronizes only when the RPi boots so does not have the precision of NTP. The RPi3 does not have a built-in battery backed real-time clock. Nevertheless, timesyncd is suitable for the AirSpy Server application, which apparently does not timestamp data passed to a client. However, the client itself should use NTP as explained below.

Even though precision timekeeping may not be needed in the server, some improvements still can be made. By default timesyncd uses a compiled-in list of NTP servers. Users in the USA may override the defaults and use time servers from the US pools. Users in other countries can select time servers specific to their country or region. To use US server pools, open the timesyncd.conf configuration file with the nano editor:

```
sudo nano /etc/systemd/timesyncd.conf
```

Now, uncomment (delete the hash mark #) the parameter `NTP=` and add US pool time servers to it as follow (each pool is separated by a space):

```
NTP=0.us.pool.ntp.org 1.us.pool.ntp.org 2.us.pool.ntp.org 3.us.pool.ntp.org
```

Local time servers should be used if available; their IP addresses may be entered on the same line (separated by a space). Save the changes to the configuration file and exit the editor (CTRL-X, Y, Enter). Reboot the RPi3

```
sudo reboot
```

To confirm the timekeeping, wait several minutes and then enter:

```
timedatectl status
```

The results of this query should show the current time in UTC and the following:

```
Time zone: Etc/UTC (UTC, +0000)
Network time on: yes
NTP synchronized: yes
```

The proper functioning of the server pools can be checked by looking at the syslog. Enter:

```
cat /var/log/syslog | grep systemd-timesyncd
```

If NTP is to be used instead of timesyncd on the spyserver, timesyncd must be turned off and then NTP can be installed:

```
sudo timedatectl set-ntp false
sudo apt-get install ntp
```

The time server pools used in NTP should be changed to country-specific pools; see [{RPIBasic}](#) for additional information.

The SDR# client program may be used for data logging, and the PC on which it runs should use NTP for timekeeping. Installing NTP on a Windows PC is described at [{NTP}](#).

Alternative to SDR#: Users may want to try the SDR-Radio Console software [{SDR-Radio}](#) with the spyserver. It is compatible with both the spyserver and a directly connected AirSpy receiver (and other SDR receivers). However, it is suggested that users first concentrate on SDR# and once satisfied that everything is working then try SDR-Radio Console.

AirSpy forum: AirSpy operates a forum at [{Forum}](#) that, in principle, can be used to discuss their products and to report problems. However, many users find it unfriendly. The AirSpy poster often is rude to forum members and insists that unless software bugs are reported in a “more technical way” and “like an engineer’s description of the problem”, there will be no consideration given. For problems not involving software bugs, other forum members are very helpful.

7. Power and Temperature measurements

Load current measurements: Measurements of a standalone RPi3 indicate it can momentarily draw over 0.7 A during a cold boot (figure 18). The start-up current does not last more than several seconds. However, if the power supply is weak or the power cable has high enough resistance, the startup current can result in enough voltage drop to disrupt the boot process. Normal running current of the RPi3 is around 300 to 350 mA with no USB port load and 5.0 Vdc input.

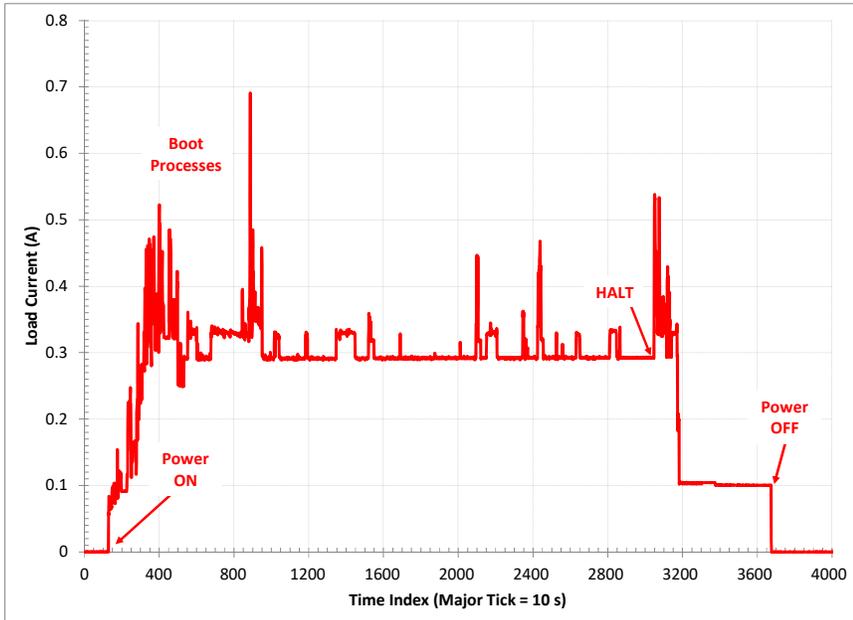


Figure 18 ~ Standalone Raspberry Pi 3 load current during cold boot, running and halt processes. No applications were running and no devices were plugged into the RPi3 USB ports during these measurements. A Keithley 2110 benchtop DMM, associated K-I Tool software and a shop-built current monitoring fixture were used for the measurements. The sampling interval was 25 ms. It is possible that a shorter sampling interval would detect larger current spikes.

The AirSpy receiver is powered through the RPi3 USB port, adding about 300 mA to the RPi3 load current. Also, when running the AirSpy Server software, the RPi3 load current increases another 200 mA or more due to the added processing load when data is actively being served to a client or clients. If a fan is used and is powered from the RPi3 5 V bus, the load will increase even more. In my test system, the fan current rating is 90 mA but actually added about 50 mA to the dc load (probably indicating the fan is starved or under-loaded).

The load current measurements were made on an operational system consisting of the AirSpy receiver and RPi3 with and without active client connections while tuned to a local FM broadcast station (table 2).

Table 2 ~ RPi3 AirSpy Server load current measurements with 5.0 Vdc input. Measurements include up to three active connections. All measurements were with 2 MHz bandwidth. Idle: No client connection; Opr: Active client connections.

Active Connections	Idle (A)	Opr (A)	Power (W)	Cooling
1	0.385	0.82	4.1	None (open air)
2	0.385	0.90	4.5	None (open air)
3	0.385	0.95	4.8	None (open air)
0	0.385	0.80	4.0	Heatsinks (open air)
1	0.432	0.90	4.5	Enclosure fan & heatsinks
2	0.432	0.96	4.8	Enclosure fan & heatsinks
3	0.432	1.02	5.1	Enclosure fan & heatsinks

Temperature measurements: I made comparative measurements of the SoC internal temperature with and without the heatsinks and with and without the PCB installed in a fan cooled enclosure (table 3). The temperature measurements were made using the following sequence:

- ⚙ Run AirSpy Server but no connection. Measure temperature 15 minutes later;
- ⚙ Connect first client and Play. Measure temperature 0.5 to 1 hour later;
- ⚙ Connect second client and Play. Measure temperature 0.5 to 1 hour later;
- ⚙ Connect third client and Play. Measure temperature 0.5 to 1 hour later.

While the measured temperatures were well within the allowable range for the components, they were measured with a relatively low ambient temperature and with a maximum of only three active connections. As a first approximation, for a given number of connections, every 1 °C increase in ambient temperature will raise the operating temperature 1 °C. Additional active connections likely will increase the SoC temperatures.

Table 3 ~ RPi3 AirSpy Server SoC temperature measurements with and without a fan and heatsinks. Ambient temperature +19 °C. The elapsed time between Start and End was 0.5 to 1 h to allow the temperatures to stabilize. All measurements were with 2 MHz bandwidth.

Cooling method	Start (°C)	End (°C)	Rise (°C)	Active connections
None (open air)	40.8	62.3	21.5	1
None (open air)	62.3	67.1	4.8	2
None (open air)	67.1	72.0	4.9	3
Heatsinks (open air)	31.7	60.7	29.0	1
Enclosure fan & heatsinks	32.2	44.0	11.8	1
Enclosure fan & heatsinks	44.0	47.2	3.2	2
Enclosure fan & heatsinks	47.2	49.4	2.2	3

The SoC internal temperature was measured by entering the following string at the PuTTY command line prompt:

```
vcgencmd measure_temp && date
```

The first part of the command displays the SoC internal temperature and the second part provides a timestamp.

The CPU load can be monitored with the *top* or *htop* process viewer application, which is similar to the Windows Task Manager. For example, to view the Raspberry Pi processes with *htop* enter the command (figure 19):

```
htop
```

For the above temperature and CPU process measurements, the AirSpy Server was run from one PuTTY session and the measurements taken from another simultaneous PuTTY session.

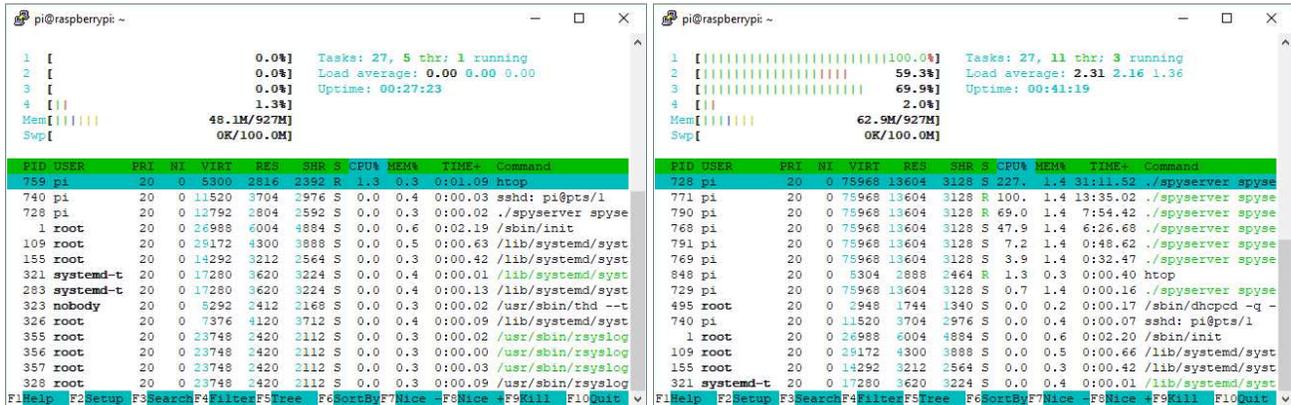


Figure 19 ~ Screenshots of the htop application. Left: AirSpy Server running but no connection. Note the negligible loads on the four CPU cores listed in top-left half of the window. Right: AirSpy Server running and one active client connection. Note that core 1 load = 100%, core 2 load = 59%, core 3 load = 70% and core 4 load = 2%. The individual processes are listed below the core load report but only a portion of the full list is shown here.

8. References and Weblinks

- {AirSpy} <https://airspy.com/>
- {AirSpyPkg} <https://airspy.com/?ddownload=3130>
- {Forum} <https://airspy.groups.io/g/main>
- {Noise} http://www.reeve.com/Documents/Noise/Reeve_Noise_5_NFMeas.pdf
- {NTP} http://www.reeve.com/Documents/Articles%20Papers/Reeve_NTP-MeinMon_Install.pdf
- {RPIBasic} http://www.reeve.com/Documents/Articles%20Papers/Reeve_RPi_BasicSetup.pdf
- {SDR-Radio} <http://www.sdr-radio.com/>
- {Spyserver} <https://airspy.com/spy-servers/>
- {Static} http://www.reeve.com/Documents/Articles%20Papers/Reeve_RPi_StaticIP.pdf
- {Stretch} <https://www.raspberrypi.org/downloads/raspbian/>
- {Volutz} <http://volutz.com/>

9. Preprogrammed Memory Card Available

A preprogrammed memory card, provisioned as described above for automatic operation, is available for 15 USD, which includes postage to United States destinations and its territories. For other destinations, please inquire. Email: orderinfo@reeve.com. Be sure to include a meaningful subject line.

Document information

Author: Whitham D. Reeve

Copyright: © 2018 W. Reeve

Revisions: 0.0 (Draft started 27 Jan 2018)
0.1 (Added temperature and load current data, verified procedures, 29 Jan 2018)
0.2 (Formatted figures, 30 Jan 2018)
0.3 (Cleanup, 03 Feb 2018)
0.4 (Added auto-start service, 06 Feb 2018)
0.5 (Rearranged introduction, 08 Feb 2018)
0.6 (Expanded timekeeping explanation, 09 Feb 2018)
0.7 (Added RPi3 power cable, 12 Feb 2018)
0.8 (Minor cleanup, 21 Feb 2018)
0.9 (Minor cleanup, 25 Feb 2018)
1.0 (Added R2 and Mini discontinued, 06 Mar 2018)
1.1 (Added RPi2, 08 Mar 2018)
1.2 (Added change device type, 21 Mar 2018)
1.3 (Added HF+ procedures, 26 Mar 2018)
1.4 (Cleanup for distribution, added RPi3+, 01 Apr 2018)
1.5 (Added restart to service unit file, 03 Apr 2018)
1.6 (Added Volutz link, 11 Sep 2018)



Word count: 7958

File size: 4772864