

Raspberry Pi Geophysics Monitoring Station

setting up the Raspberry Pi

Ian Robinson

December 10, 2023

revision 5.0

ian@schoolphysicsprojects.org

Abstract

This describes setting up a Raspberry Pi to act as a geophysics monitoring station. The monitoring program, written in python, may be configured to work with a range of sensors; to date infrasound, temperature, magnetic field and a micro-barometer have been used [3]. The monitor typically starts at boot saving data hourly in *.mseed*[1] format for later analysis. Hourly data plots are generated which may be automatically uploaded to a server via ftp. The Pi may be left running for months unattended, uploading plots to the net with data files being transferred off via ftp when required.

Versions of the monitoring program specific to flux-gate magnetometers, infrasound and microbarometers may be downloaded from GitHub [2].

1 Overview

Steps required to get a working system are:

- Copying a suitable Linux Operating system onto an s.d card
- Booting the Pi whilst connected to a LAN
- Configuring the Operating System
- Installing Required Software
- Configuring a Python Virtual Enviroment
- Copying over the monitoring software
- Configuring crontab
- Configuring auto-uploading of plots

2 Installing the O.S.

tested on a Raspberry Pi 4b running 64 bit Debian Bookworm

Currently the default Pi operating system is Raspian, based on Debian 11 'Bookworm', 64 bit. This needs to be downloaded and written to an sd card which will serve as the Pi's non-volatile memory.

I recommend a *headless* system i.e. a Pi without a monitor or keyboard. The system is accessed and configured over the network. Once installed the Pi may be left running for months without physical intervention.

A running system need only boot into a command line interface reducing required system resources. However we do need a desktop environment installed for the matplotlib graph-plotting routine [6] though we do not need to actually boot into the desktop.

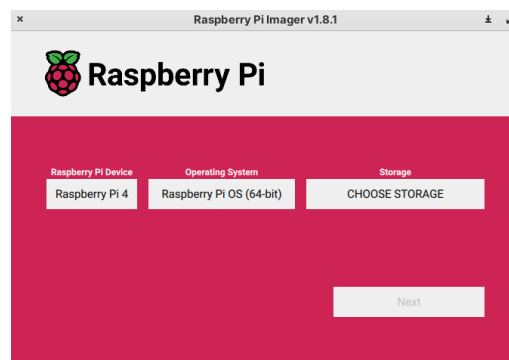


Figure 1: R.Pi Imager

The Raspberry PI *Imager* [4] app may be used to download an OS image, transfer to an SD card and perform some basic configuration before inserting the card into the R.Pi. Simple to use, Imager is available for Linux, Windows and Macs [fig-1]. The latest version will prompt you to preconfigure the OS when it is written to the flash card [fig-2]. We can specify a network hostname for the pi, such as *GeoPhysStation*. More importantly we will create a default user *geophysics* and enable network login via *ssh*.

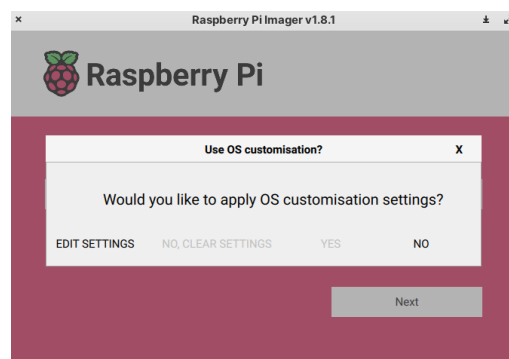


Figure 2: R.Pi Imager - configuration prompt

Enabling *ssh* allows us to connect to the Pi over a local network[fig-3].

After writing the OS image to the flash card insert it into a Pi, connected by ethernet or wi-fi (tricky) to a local network. Power up the Pi and use an *ssh* program on a desktop computer to log into the Pi using the username and password you specified in the Imager configuration step.

OS Customisation

General Services Options

☒ Set hostname: GeoPhysStation .local

☒ Set username and password

Username: geophysics

Password:

☐ Configure wireless LAN

SSID:

Password:

☒ Show password ☐ Hidden SSID

Wireless LAN country: GB

☒ Set locale settings

Time zone: Europe/London

Keyboard layout: gb

SAVE

Figure 3: R.Pi Imager - sample configuration

3 Configuring the Operating System

Hopefully we have now inserted the sd card and booted the Pi to which we have connected via a terminal *ssh* program.

When you are logged in run;

```
sudo raspi-config
```

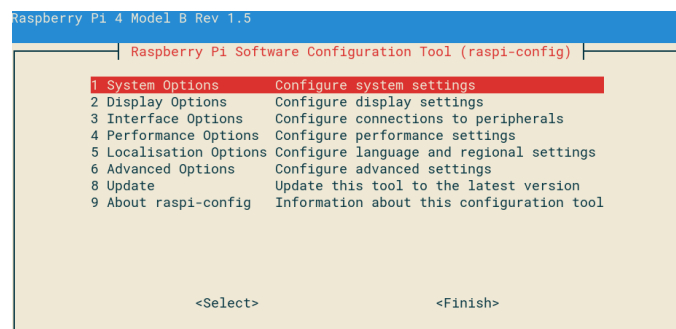


Figure 4: raspi-config

Select;

1. System options
- B2 Console Autologin Text console, automatically logged in as 'geophysics' user.

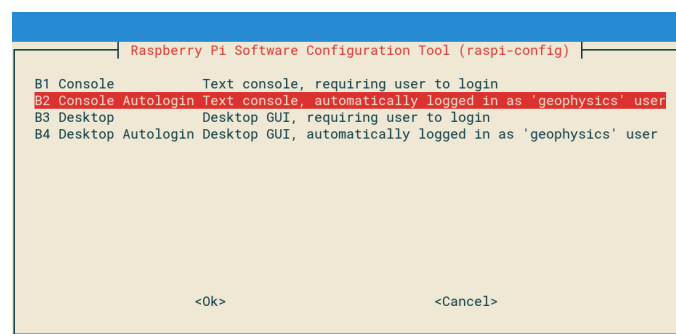


Figure 5: select boot to command line

If you are setting up an i²c sensor – such as micro-barometer or infrasound monitor select [fig-6]

3. Interface options
- I5 I2C Enable/disable automatic loading of I2C kernel module

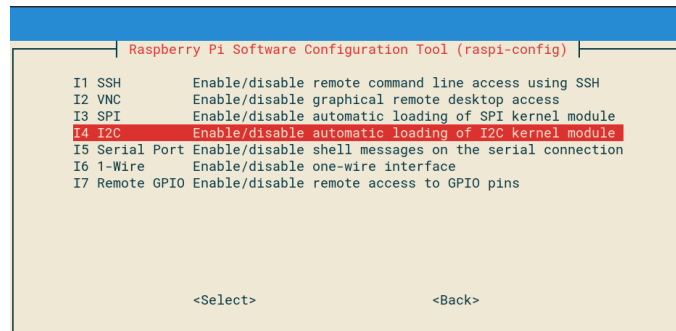


Figure 6: enabling i²c interface

Finally we need to expand the filesystem created by Imager to fill the entire sd card

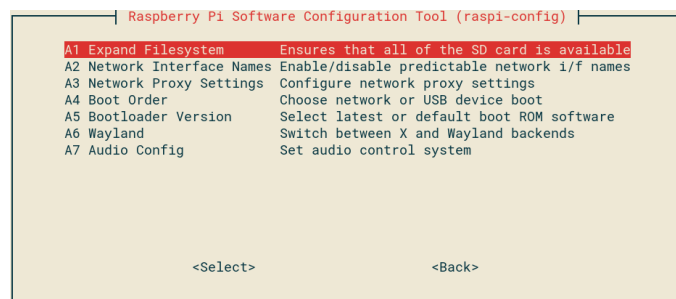


Figure 7: expanding file system

6. Advanced Options

A1 Expand Filesystem

Exit raspi-config and reboot with

```
sudo reboot
```

Wait a minute then login again

```
sudo apt install ftp
sudo apt install ntpdate
sudo timedatectl set-ntp True
sudo apt install libatlas-base-dev
```

4 Installing ObsPy

ObsPy [5] is a suite of geophysics routines written in python, this is used by the monitoring program to save data in .mseed format as well as some mathematical routines. Unfortunately ObsPy is not packaged for Raspian.

The latest OS discourages us from installing non-officially package software system wide. Rather than override this security feature we will create a local version of python for user 'geophysics' and install ObsPy into this virtual environment.

4.1 Creating A Virtual Environment

First install python3 venv (this may already be installed)

```
sudo apt install python3-venv -y
```

Now we need to create a folder to hold our python3 virtual environment.

```
python3 -m venv ~/my_venv
```

running the command

```
ls
```

shows us the folders in our home folder, including the new *my_venv* folder.

```
Bookshelf  Documents  Music      Pictures   Templates
Desktop    Downloads  my_venv    Public     Videos
```

We now active our virtual environment with

```
source ~/my_venv/bin/activate
```

The command prompt will now be prefaced with the name of our new virtual environment

```
(my_venv) geophysics@GeoPhysStation:~ $
```

With the virtual environment active can now install obspy into our virtual environment with

```
pip install obspy
```

This will probably take a while so have a cuppa/dram.

When it has finished we will also run

```
pip install smbus  
pip install smbus2
```

(smbus is used by the microbarometer, smbus2 by the aurora monitor)

If you are setting up the microbarometer

```
pip install icp10125
```

4.2 Virtual Environment?

Python3 will be already installed on our system as a default component of Raspbian. For security reasons we are discouraged from installing packages such as obspy system-wide, i.e. integrated into the system's python3.

We could override this but, what happens if the system's python version is upgraded? Obspy may lag behind the latest python and now may not run properly. We risk entering *dependancy hell*.

The solution is simple. We perform another local python install. Into this virtual-environment [7] we install obspy. Using the *pip* installer ensures that the local versions of obspy and python match .. *dependancy heaven*.

When we issued the command

```
~/my_venv/bin/activate
```

we were telling our terminal to always run the local version of python rather than the system wide one.

A problem arises in that the activation ends when we logout or reboot. The simplest way around this is, rather than call the python interpreter with

```
python3 myprogram.py
```

which gives us the system interpreter, we explicitly call the local version with

```
~/my_venv/bin/python3 myprogram.py
```


5 Installing the Monitoring Code

From a terminal window create a folder for the monitoring program. It will automatically create subfolders for data and plots on first run.

```
mkdir station
```

Note: If you change the name of this folder you will have to edit approx line 5 of *globalvariablesModule.py* as well as the file paths in *contab* (next section).

using ftp copy over the monitoring code to the station folder. There are 3 files

- geostationModules.py
- globalvariablesModule.py
- the sensor specific program;
— genericMonitor.py, microbarometer.py, auroraMonitor.py, or InfraSoundMonitor.py

If you intend having the plots automatically upload to a website or server then copy over the ftp uploader script to the station folder.

- uploadHourly.sh

You will need to edit this with the ftp address and password for your remote server. It may be necessary to set the file's permissions to *execute*.

5.1 uploadHourly.sh

```
HOST= '*** ftp _ site _ address *** '
USER= 'login_id '
PASSWD= 'loginpassword '

ftp -p -n -v $HOST << EOT
ascii
user $USER $PASSWD
prompt

cd schoolphysicsprojects/projects/testbed/PlotsA
ls -la
lcd ~/station/Plots
mput *.svg
bye
EOT
rm ~/station/Plots/*
```

- ftp ... initiates an ftp connection to a remote site
- cd ... changes to the remote directory holding uploaded plots
- lcd ... changes to the local directory, on the PI, holding the most recent plots in svg format
- mput ... copies all .svg files to the remote server
- EOT ... ends ftp session

- `rm ...` deletes the local copies of the uploaded images, saving space on the pi and reducing future uploads

6 Configuring crontab

To schedule our monitor to start at boot and to run scripts to upload graphs to a remote server we will use crontab, a linux application for scheduling task. From a terminal window run

```
crontab -e
```

Add the following lines to the bottom of the crontab file.

```
#  
# m h dom mon dow    command  
  
02 * * * * ~/station/uploadHourly.sh 2>> ~/station/errorsHourly.txt  
  
@reboot sleep 30 && ~/my_venv/bin/python3 ~/station/microbarometer.py 2>  
~/station/stationErrors.txt &
```

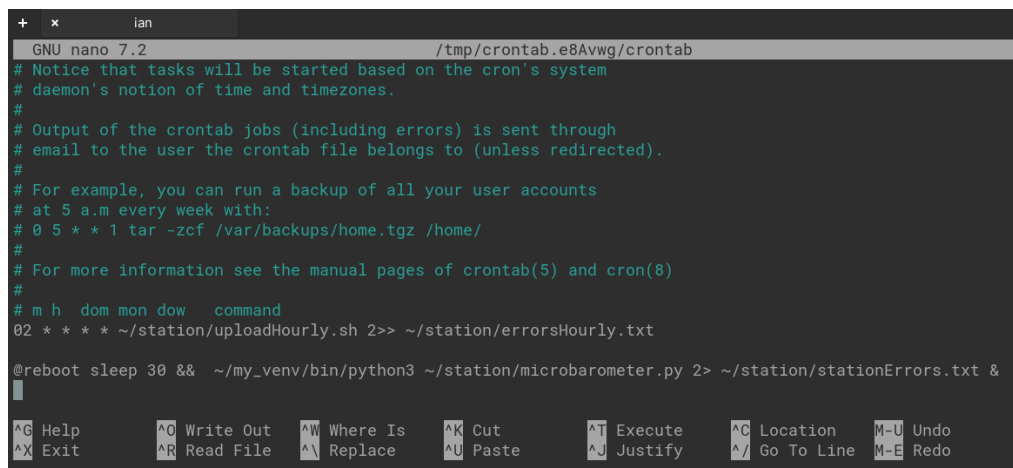
note: each command occupies a single line. Replace microbarometer.py with the name of the specific monitoring program.

CTRL-X saves

To explain:- looking at each command.

At 02 minutes past each hour uploadHourly.sh is run with any errors written to errorsHourly.txt

At reboot, after a wait of 30 seconds, python3 in our virtual environment executes microbarometer.py with any errors written to stationErrors.txt.

A screenshot of a terminal window with a dark background. The title bar shows a window icon, a close icon, and the name 'ian'. The terminal content shows the GNU nano 7.2 editor editing a file at /tmp/crontab.e8Avwg/crontab. The file contains several lines of comments and two cron jobs. The first job is '02 * * * * ~/station/uploadHourly.sh 2>> ~/station/errorsHourly.txt' and the second is '@reboot sleep 30 && ~/my_venv/bin/python3 ~/station/microbarometer.py 2> ~/station/stationErrors.txt &'. At the bottom, there is a status bar with various keyboard shortcuts like ^G Help, ^O Write Out, ^W Where Is, ^K Cut, ^T Execute, ^C Location, M-U Undo, ^X Exit, ^R Read File, ^\ Replace, ^U Paste, ^J Justify, ^_ Go To Line, and M-E Redo.

```
+ x ian  
GNU nano 7.2 /tmp/crontab.e8Avwg/crontab  
# Notice that tasks will be started based on the cron's system  
# daemon's notion of time and timezones.  
#  
# Output of the crontab jobs (including errors) is sent through  
# email to the user the crontab file belongs to (unless redirected).  
#  
# For example, you can run a backup of all your user accounts  
# at 5 a.m every week with:  
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/  
#  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow    command  
02 * * * * ~/station/uploadHourly.sh 2>> ~/station/errorsHourly.txt  
  
@reboot sleep 30 && ~/my_venv/bin/python3 ~/station/microbarometer.py 2> ~/station/stationErrors.txt &  
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo  
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo
```

Figure 8: crontab

7 Problems?

I have tested these instructions on both a pi-4b and pi-5. However I doubt they are without uncertainty. If you have any problems please email me - ian@sschoolphysicsprojects.org and I will be happy to try and help.

References

- [1] miniSEED, <http://docs.fdsn.org/projects/miniseed3/en/latest/>
- [2] GitHub Code Versions, <https://github.com/starfishprime101>
- [3] Example Systems, schoolphysicsprojects.org/index.html
- [4] R.Pi Imager, <https://www.raspberrypi.com/software/>
- [5] ObsPy, https://www.geophysik.uni-muenchen.de/~megies/www_obsrise/
- [6] Matplotlib, <https://matplotlib.org/>
- [7] Python3 Virtual Environments, <https://www.techcoil.com/blog/how-to-use-python-3-virtual-environments-to-run-python-3-applications-on-your-raspberry-pi/>