# nanoVNA-H4 software support under Ubuntu

## Introduction

This is a work in progress.

## External control

### *NanoVNA Saver*

NanoVNA Saver[1] is an auxiliary program which supports:

- saving Touchstone files
- segmented sweeps, to permit increasing the number of frequency points
- an arbitrary number of associated calibration sets
- displaying and analysing resulting data

### *Installing NanoVNA Saver under Ubuntu 18.04*

*This procedure will be different under Ubuntu 20.04.*

This procedure is adapted from that suggested by the repository[2]. *In essence, success is due to the use of a Python Virtual Environment to isolate NanoVNA Saver & its associated Python requirements from anything else on the host machine.*

*Install Python 3.7*

This is required to run NanoVNA Saver, but must be installed in such a way as to retain Python 3.6 as the default Python 3 version.

- Install Python 3.7 and pip

```
sudo apt install python3.7 python3-pip
```

- Install Python 3.7 development files

```
sudo apt install python3.7-dev
```

It is now necessary to ensure that the default version of Python 3 remains 3.6, using the **update-alternatives** mechanism[3].

[1] See https://github.com/mihtjel/nanovna-saver.

[2] Noted at https://github.com/mihtjel/nanovna-saver#ubuntu-1804--1904.

[3] This process is inspired by, and adapted from, https://unix.stackexchange.com/questions/410579/change-the-python3-default-version-in-ubuntu, top answer.

```
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.7 1
```

```
sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.6 2
```

```
sudo update-alternatives  --set python3 /usr/bin/python3.6
```

*Install NanoVNA Saver itself*

*This process is distilled from a document*[4] *referenced on the support group Wiki*[5]. **NanoVNA Saver** is installed into, and run from, a **Python Virtual Environment** (**PVE**)[6].

- Install the **Python 3.7 virtual environment**:

```
sudo apt install python3.7-venv
```

- Install **Git**

```
sudo apt install git
```

- Make a **directory** to hold Python virtual environments:

```
cd
```

```
mkdir ~/.venv
```

- Create a **new PVE** for NanoVNA Saver, but **without pip**:

```
cd ~/.venv
```

```
python3.7 -m venv nvna-s --without-pip
```

- Activate the newly created **PVE**:

```
source ~/.venv/nvna-s/bin/activate
```

The **PVE** is now, effectively, an isolated instance of Python in to which NanoVNA Saver may be installed, fuss-free.

- Note that the command prompt is now prefixed by the **PVE**'s name:

```
(nvna-s) user@host:~/.venv$
```

- **Install** the latest version[7] of **pip** and associated support (**setuptools** & **wheel**)

```
curl https://bootstrap.pypa.io/get-pip.py | python
```

[4] Document found at https://groups.io/g/nanovna-users/files/NanoVNA PC Software/NanoVNA-Saver/nvna-s-pve-rev-c.pdf.

[5] Wiki entry at https://groups.io/g/nanovna-users/wiki/12336#Python.

[6] Outlined at https://docs.python.org/3.7/tutorial/venv.html.

[7] The version of **pip** associated with Python 3.7 is not the most up-to-date.

- At the end of this process there should be two lines similar to:

```
Installing collected packages: pip, setuptools, wheel
```

```
Successfully installed pip-20.1.1 setuptools-46.4.0 wheel-0.34.2
```

- Clone the source code from GitHub in to the PVE:

```
git clone https://github.com/mihtjel/nanovna-saver
```

- Install the dependencies and build **NanoVNA Saver**[8]:

```
cd ~/.venv/nvna-s/nanovna-saver
```

```
python3.7 -m pip install .
```

- Now run **NanoVNA Saver** (**nanovna-saver.py**) and test

```
python3.7 ./nanovna-saver.py
```

- Check that the program runs, and behaves roughly as expected.
- Quit the PVE, and return to the home directory

```
deactivate
```

```
cd
```

*Make a script*

In your favourite editor, make a script[9] **~/bin/nanovnaSaver.sh**:

```
#!/bin/bash
# nanovnaSaver.sh
# This script runs NanoVNA Saver from a PVE
# Adapted from a similar script by Nick, G3VNC
# RAG, 2020-05-22
cd
# Enter the PVE
source ~/.venv/nvna-s/bin/activate
# Run our program
python3.7 ./.venv/nvna-s/nanovna-saver/nanovna-saver.py
# Program finished, leave the PVE cleanly
```

[8] The dependencies are brought in automatically.

[9] This procedure assumes that **~/bin** already exists, and that **~/bin** is in included in the **PATH** environmental variable.

```
deactivate
```

Make the script executable:

```
chmod +x ~/bin/nanovnaSaver.sh
```

Finally, add a **menu entry**. There is an icon file in the source[10].

### *Installing NanoVNASaver under Ubuntu 20.04*

For Ubuntu 20.04, the default Python version is 3.8. Using the PVE still seems like a very smart move, though[11].

## Firmware

### *Setting firmware update mode*

Setting the nanoVNA-H4 in to DFU mode requires the following procedure[12]:

- Power off NanoVNA-H4
- Press down the jog switch
- Power on
- Release jog switch

At this point, the nanoVNA-H4 is in DFU mode, with a **blank screen**[13].

*For further investigation!*

## Octave interfacing

There is an example script available[14], which demonstrates how a **nanoVNA** may be controlled from **Octave**. Although the example is written with Windows in mind, it may readily be adapted to **Linux**. There is a syntax mistake which must be corrected[15], and then the example script permits measuring VSWR over a self-defined set of radio bands. The key component is the **Instrument Control** package at **Octave Forge**[16].

The virtue of the example is to demonstrate controlling the **nanoVNA** from **Octave**, in principle for any application.

*Not yet tested!*

[10] Icon file at https://github.com/mihtjel/nanovna-saver/blob/master/icon_48x48.png.

[11] Not yet investigated in practice.

[12] Noted at https://groups.io/g/nanovna-users/message/10140.

[13] As noted at https://groups.io/g/nanovna-users/message/10149.

[14] From GitHub, at https://github.com/PromptusCTO/K1YBE.

[15] Correction noted at https://groups.io/g/nanovna-users/message/13860.

[16] More informaation at https://octave.sourceforge.io/instrument-control/.

## Appendix A: GitHub nanoVNA Saver install process

*The following process is likely to fail due to version dependency issues, relating to PyQt5.*

This process is elaborated from the GitHub description[17]. **Python 3.7**, or later, must already be installed[18].

- Firstly install a specific version of PyQt5[19]:

```
sudo python3.7 -m pip install pyqt5==5.14
```

- Make a suitable directory to hold the code directory[20], and **cd** to it
- Clone the repository

```
git clone https://github.com/mihtjel/nanovna-saver
```

- **cd** to the source directory

```
cd nanovna-saver
```

- perform a **pip** installation

```
python3.7 -m pip install .
```

Test the installation:

```
python3.7 nanovna-saver.py
```

**RAG 2020-06-03**

[17] Install process given at https://github.com/mihtjel/nanovna-saver#installation-and-use-with-pip-1.

[18] This process is understood to be effective with Fedora and other distributions. Debian and derivatives display issues.

[19] As noted at https://www.reddit.com/r/learnpython/comments/fjm1cp/stuck_trying_to_run_nanovnasaver_via_pyqt5_on/. [Note the description of the full code is not quite right.]

[20] Such as **~/code**, but may be any directory to which the user has write access.