

Raspberry Pi- Pi-Nut Network

Installing Network UPS Tools (NUT) version 2.8.3 from source on a Raspberry Pi 4 running Raspberry Pi OS (based on Debian) requires compiling the software from the source code, as a pre-built Debian package for this specific version may not be available. Below are the detailed steps to accomplish this, tailored for a Raspberry Pi 4. These instructions assume you're using Raspberry Pi OS 64-bit (based on Debian Bullseye or later), but they should also work for 32-bit with minor adjustments.

Prerequisites

1. **Raspberry Pi OS**: Ensure your Raspberry Pi 4 is running an up-to-date version of Raspberry Pi OS. Update the system before starting:

```
sudo apt update && sudo apt upgrade -y
```

2. **Dependencies**: You'll need tools and libraries to compile NUT and support its functionality (e.g., USB, if your UPS connects via USB).

3. **Internet Connection**: Required to download the source code and dependencies.

4. **UPS Connection**: If using a USB-connected UPS, ensure it's plugged into the Raspberry Pi before configuring NUT.

Step-by-Step Installation Guide

1. Install Required Dependencies

NUT requires several development tools and libraries to build from source. Install them using:

```
sudo apt install -y build-essential autoconf automake libtool pkg-config \
libusb-dev libneon27-dev libsnmp-dev python3-dev python3-pip git
```

- **build-essential**: Provides compilers (gcc, g++) and make.
- **autoconf, automake, libtool**: Needed to generate the configure script.
- **pkg-config**: Helps locate libraries during compilation.
- **libusb-dev**: Required for USB UPS support (common for modern UPS devices).
- **libneon27-dev, libsnmp-dev**: Optional, for network-based UPS protocols like SNMP.
- **python3-dev, python3-pip**: Optional, for Python-based NUT tools.
- **git**: Used to clone the NUT repository if needed.

If you need additional features (e.g., CGI for web interfaces), you can install ``apache2`` and ``nut-cgi`` dependencies later, as shown in some guides.[](<https://technotim.live/posts/NUT-server-guide/>)

2. Download NUT 2.8.3 Source Code

The NUT 2.8.3 source code is available from the official NUT website or GitHub. Download the stable tarball for version 2.8.3:

```
wget https://networkupstools.org/source/2.8/nut-2.8.3.tar.gz
```

Verify the integrity of the downloaded file using the SHA-256 checksum (optional but recommended):

```
sha256sum nut-2.8.3.tar.gz
```

Compare the output with the SHA-256 sum provided on the NUT website (<https://networkupstools.org/source/2.8/>).[\]\(https://networkupstools.org/historic/v2.8.3/download.html\)](https://networkupstools.org/historic/v2.8.3/download.html)

Extract the tarball:

```
tar -xzf nut-2.8.3.tar.gz
cd nut-2.8.3
```

Alternatively, if you prefer to clone the Git repository and check out the specific 2.8.3 tag:

```
```bash
git clone https://github.com/networkupstools/nut.git
cd nut
git checkout v2.8.3
./autogen.sh
```
```

Note: The Git method requires `autoconf`, `automake`, and `libtool` to generate the configure script, as these are not included in the repository. The tarball already includes a pre-built configure script, making it simpler for most users.[\]\(https://networkupstools.org/historic/v2.8.3/download.html\)](https://networkupstools.org/historic/v2.8.3/download.html)

3. Configure the Build

Before compiling, configure the source tree for your system. Create a system user and group for NUT (e.g., `ups` and `nut`) to run the services securely:

```
sudo groupadd nut
sudo useradd -r -g nut -s /bin/false ups
```

-r : Creates a system user.
-g nut: Assigns the user to the `nut` group.
-s /bin/false: Prevents the user from logging in.

Now, configure the build with the appropriate user and group:

```
```bash
```

```
./configure --with-user=ups --with-group=nut --with-usb
...
```

- `--with-user=ups` and `--with-group=nut`: Specify the user and group for NUT services.  
- `--with-usb`: Enables USB support (common for UPS devices). Add other options like `--with-snmp` or `--with-cgi` if needed (see `./configure --help` for options).[](<https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>)

If you encounter issues with missing dependencies, install them using `apt` (e.g., `sudo apt install libusb-1.0-0-dev` for USB support).[](<https://github.com/networkupstools/nut/issues/2431>)

#### #### 4. Build and Install NUT

Compile the source code:

```
```bash  
make  
...
```

This builds the NUT client, server, and selected drivers. If you encounter errors, check for missing dependencies or consult the NUT user manual.[](<https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>)

Install the compiled programs:

```
```bash  
sudo make install
...
```

This installs NUT to the default location (`/usr/local/ups`). Sample configuration files are installed with `.sample` extensions to avoid overwriting existing configurations.[](<https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>)

Optionally, use `make install-as-root` to handle additional setup tasks like setting permissions and creating directories (see below).[](<https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>)

#### #### 5. Set Up the State Path

Create a directory for NUT to store UPS status data and set appropriate permissions:

```
```bash  
sudo mkdir -p /var/state/ups  
sudo chmod 0770 /var/state/ups  
sudo chown root:nut /var/state/ups  
...
```

This ensures the `ups` user and `nut` group have access to the state path.[](<https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>)

6. Configure USB Permissions (if using a USB UPS)

If your UPS connects via USB, set permissions for the USB device to allow the `ups` user to access it. USB devices are dynamically created, so use `udev` rules to manage permissions. Create a `udev` rule:

```
```bash
```

```
sudo nano /etc/udev/rules.d/99-nut-ups.rules
```
```

Add the following (adjust `idVendor` and `idProduct` based on your UPS, which you can find using `lsusb` or `nut-scanner -U`):

```
```bash
SUBSYSTEM=="usb", ATTR{idVendor}=="051d", ATTR{idProduct}=="0002", MODE="0660",
GROUP="nut"
```
```

Save and reload `udev` rules:

```
```bash
sudo udevadm control --reload-rules
sudo udevadm trigger
```
```

Run `nut-scanner` to identify your UPS:

```
```bash
sudo nut-scanner -U
```
```

This should output details like `vendorid`, `productid`, and `driver` (e.g., `usbhid-ups`). Use these in the next step.[](<https://technotim.live/posts/NUT-server-guide/>)[](<https://www.jeffgeerling.com/blog/2025/nut-on-my-pi-so-my-servers-dont-die>)

7. Configure NUT

NUT configuration files are installed in `/usr/local/ups/etc/`. Copy the sample files to their active versions:

```
```bash
sudo cp /usr/local/ups/etc/ups.conf.sample /usr/local/ups/etc/ups.conf
sudo cp /usr/local/ups/etc/upsd.conf.sample /usr/local/ups/etc/upsd.conf
sudo cp /usr/local/ups/etc/upsmon.conf.sample /usr/local/ups/etc/upsmon.conf
sudo cp /usr/local/ups/etc/upsd.users.sample /usr/local/ups/etc/upsd.users
sudo cp /usr/local/ups/etc/nut.conf.sample /usr/local/ups/etc/nut.conf
```
```

Edit `ups.conf` to define your UPS:

```
```bash
sudo nano /usr/local/ups/etc/ups.conf
```
```

Example configuration (adjust based on `nut-scanner` output):

```
```bash
[myups]
 driver = usbhid-ups
 port = auto
 desc = "My UPS"
 vendorid = 051d
 productid = 0002
```
```

Edit `upsd.conf` to allow network access:

```
```bash
sudo nano /usr/local/ups/etc/upsd.conf
```
```

Change:

```
```bash
LISTEN 127.0.0.1 3493
```
```

to:

```
```bash
LISTEN 0.0.0.0 3493
```
```

Edit `upsd.users` to define admin and monitor users:

```
```bash
sudo nano /usr/local/ups/etc/upsd.users
```
```

Example:

```
```bash
[admin]
 password = myadminpassword
 actions = set
 actions = fsd
 instcmds = all
 upsmon primary
```
```

[observer]

```
    password = myobserverpassword
    upsmon secondary
```
```

Edit `upsmon.conf` to monitor the UPS:

```
```bash
sudo nano /usr/local/ups/etc/upsmon.conf
```
```

Example:

```
```bash
MONITOR myups@localhost 1 admin myadminpassword primary
FINALDELAY 180
```
```

Set the NUT mode in `nut.conf`:

```
```bash
sudo nano /usr/local/ups/etc/nut.conf
```
```

Set:

```
```bash
MODE=netserver
```
```

### #### 8. Start and Enable NUT Services

Start the NUT driver and server:

```
```bash
sudo /usr/local/ups/sbin/upsdrvctl start
sudo /usr/local/ups/sbin/upsd
sudo /usr/local/ups/bin/upsmon
```
```

Enable services to start on boot:

```
```bash
sudo systemctl enable nut-server
sudo systemctl enable nut-monitor
sudo systemctl start nut-server
sudo systemctl start nut-monitor
```
```

If `systemctl` commands fail, it may be due to missing `systemd` service files. Create them manually or use `make install-as-root` to generate them. <https://www.jeffgeerling.com/blog/2025/nut-on-my-pi-so-my-servers-dont-die> <https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>

### #### 9. Verify Installation

Check if NUT is working:

```
```bash
upsc myups@localhost
```
```

This should display UPS status details (e.g., `battery.charge``, `device.model``). If it fails, check logs in `/var/log/syslog` or `/var/log/messages` for errors. <https://www.jeffgeerling.com/blog/2025/nut-on-my-pi-so-my-servers-dont-die>

### #### 10. Troubleshooting

- **Command not found** (e.g., `upsdrvctl``): Ensure `/usr/local/ups/bin`` and `/usr/local/ups/sbin`` are in your `PATH`:

```
```bash
export PATH=$PATH:/usr/local/ups/bin:/usr/local/ups/sbin
```
```

Add this to `~/.bashrc`` for persistence. This issue was reported in similar installations. <https://forums.raspberrypi.com/viewtopic.php?t=334804>

- **USB issues**: Verify `libusb-1.0-0-dev`` is installed and the `udev`` rules are correct. <https://github.com/networkupstools/nut/issues/2431>

- **Service not starting**: Check permissions on `/var/state/ups`` and configuration files. Ensure the `ups`` user has access.

- **Driver issues**: If `nut-scanner`` doesn't detect your UPS, consult the NUT hardware compatibility list (<https://networkupstools.org/docs/user-manual.chunked/ar01s05.html>) or try a different driver (e.g., `nutdrv_qx``). <https://www.jeffgeerling.com/blog/2025/nut-on-my-pi-so-my-servers-dont-die>

### #### 11. Optional: Install NUT CGI for Web Interface

To monitor via a web interface:

```
```bash
sudo apt install apache2 nut-cgi
sudo a2enmod cgi
sudo systemctl restart apache2
```
```

Configure `/usr/local/ups/etc/hosts.conf`:

```
```bash
sudo nano /usr/local/ups/etc/hosts.conf
```
```

Example:

```
```bash
MONITOR myups@localhost "My UPS"
```
```

Access the web interface at `http://<your-pi-ip>/cgi-bin/nut/upsstats.cgi`. [.\[\]\(https://technotim.live/posts/NUT-server-guide/\)](https://technotim.live/posts/NUT-server-guide/)

---

### ### Notes

- **Why 2.8.3?**: This version includes specific fixes and features not in earlier releases (e.g., improved driver support). Check the release notes for details ([https://networkupstools.org/docs/release-notes.chunked/NUT\\_Release\\_Notes.html#\\_release\\_notes\\_for\\_nut\\_2\\_8\\_3](https://networkupstools.org/docs/release-notes.chunked/NUT_Release_Notes.html#_release_notes_for_nut_2_8_3)). [.\[\]\(https://github.com/networkupstools/nut/releases\)](https://github.com/networkupstools/nut/releases)
- **Security**: Set strong passwords in `upsd.users` and limit network access in `upsd.conf` if not needed.
- **Upgrading**: If upgrading from NUT 2.7.x, remove the old version first (`sudo apt remove nut nut-client nut-server`) to avoid conflicts. [.\[\]\(https://forums.raspberrypi.com/viewtopic.php?t=334804\)](https://forums.raspberrypi.com/viewtopic.php?t=334804)
- **Documentation**: Refer to the NUT user manual (<https://networkupstools.org/docs/user-manual.chunked/>) for advanced configuration. [.\[\]\(https://networkupstools.org/docs/user-manual.pdf\)](https://networkupstools.org/docs/user-manual.pdf)

If you encounter specific errors, let me know the error messages, and I can provide targeted assistance!

---

Revision #1

Created 25 August 2025 14:02:04 by Steve Ling

Updated 30 December 2025 21:24:01 by Steve Ling