NEQ-6 : Next generation

Arjan te Marvelde, Version 01, initial version Dec 2024,

This article describes the things I have done to upgrade my SkyWatcher stock NEQ-6 mounting. The ultimate goal is to:

- Replace the motor driver board with an OnStep implementation (with Teensy 4.0)
- Integrate this with the Raspberry-Pi based INDI server (as in my previous Remote Control)
- Install Rowan's belt drive modification

Where the belt drive upgrade can be treated independently, the electronics parts need some design work. The used modules are given: *Raspberry-Pi 4*, *Teensy 4.0*, *TMC2130 drivers*, *GPS module*, *Power conditioning*. However, the connecting module between these modules is a new PCB, which is a part of this project.

The old stack still contains the original NEQ-6 driver board. This has largely determined the way it has been put together, but since a new PCB will be created for the OnStep electronics, this can be extended to host the other modules as well. Also, it can be made to click directly on the Raspberry-Pi 40 pin extension connector.

<<IMG>>

There is no room to orient the RPi in such a way that the connectors are flush with the plate surface, so the same approach is taken as in the current setup: connectors accessible from the side. There is however more dimensional freedom, since the original SkyWatcher driver board is ditched.

<<IMG>>

Contents

1 NEQ6 with OnStep and INDI inside

Optimum observation time on higher latitudes is usually during the winter and early spring. These months can also be quite cold, and hence observation and astrophotography are less than comfortable. So, it is time for a remote-control facility, enabling the operation of mount and astrophotography from inside the warm house.

I have done remote control inside the NEQ6 mount before (see above), but this new version integrates it with a replacement OnStep-based motor driver. Previous versions have been based on dual Raspberry Pi2B, a single Raspberry Pi3B, but now the Pi4 and Pi5 have emerged allowing a more performant solution. The Pi5 has not yet been tried, but would give about twice the performance as the Pi4 in this implementation.

The document describes step-by-step how I made the hardware, driver and server software configuration. The description is chopped up in parts which may also be used independently.

1.1 Build summary

1.1.1 PCB stack

- RPi4-4GB with a fast 64GB SD card, running Raspberry Pi OS (Bookworm), providing a WiFi hotspot access point as well as an Ethernet interface to connect to an upstream network.
- The RPi4 hosts EKOS/KStars, INDI-server and all INDI drivers, PHD2, Astrometry plate solver and more. The Raspberry Pi OS is very lightweight, and a rudimentary desktop, which is made available remotely via RealVNC server and client.
- The RPi4 is integrated with a purpose made PCB containing OnStep motor driver, GPS receiver and Power conditioning. The stack is built into the NEQ6 RA housing.

1.1.2 Devices

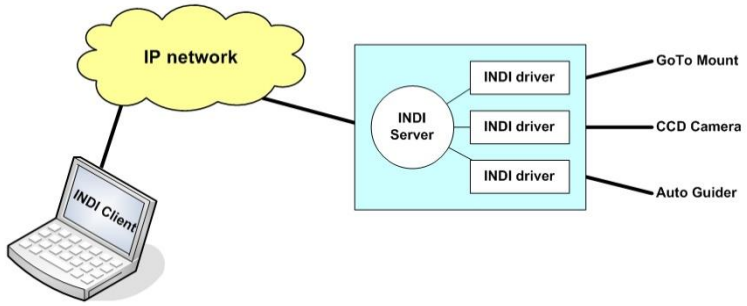
- A Canon 450D DSLR is connected through USB and a dedicated power adapter,
- The NEQ6 mount is driven by OnStep and connected internally through a serial interface,
- The GPS is connected internally and made available through gpsd and chrony for time and location,
- A home made joystick is connected through USB,
- A ZWO ASI385MC autoguider / planetary camera also connected through USB.

1.1.3 User interface

- A Laptop, which has a wireless (or wired) connection to the INDI box.
- The laptop hosts a VNC client (RealVNC) enabling remote access to the RPi desktop. This laptop could be replaced with any other device hosting a VNC client.
- The RPi server also supports FTP and SSH access.

1.2 Design choices

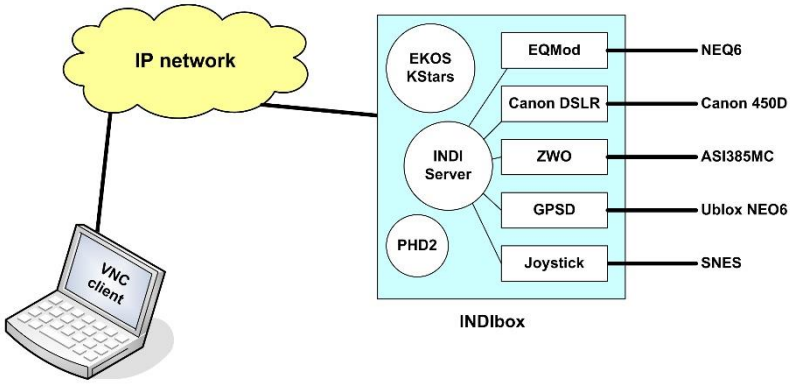
One option for implementing remote-control is to route a lot of cables from the observation location into the house. A much more elegant way is to run all device-interfacing on a local embedded computer and to have a remote workstation for controlling the observatory setup. This is exactly what the INDI framework offers: a server that provides a standardized method to access and control the variety of attached devices, such as goto, camera and auto guider. The INDI server connects to an INDI compatible client on a workstation, through any IP network, such as the home LAN.



Distributed Architecture

When the workstation PC is Windows based, the choice of clients boils down to ***Cartes du Ciel*** for goto control and the photo capturing software ***CCD-Ciel***. The main alternative ***KStars / EKOS*** is running on Linux.

A more robust solution is to use a remote *virtual desktop* and run the INDI client on the remote computer as well. The loose interface allows the network to fail while the remote telescope control continues to function. However, allocating everything to the RPi computer implies that significantly more processing power is needed. The RPi3 and certainly the RPi4 appear to be powerful enough to run everything in one processor. The user interface can be a virtual desktop, running on the workstation, or even on a tablet or a smart-phone.



VNC based architecture

1.3 Peripherals

Apart from the electronics stack and functional software some supporting hardware is needed to complete the system.

1.3.1 Enclosure

The current implementation integrates the lot inside the NEQ6 mount, replacing the access panel and legacy NEQ6 driver board. The reticule LED and RA/DEC motor connectors can be plugged into the driver board. Power input is 12V and an auxiliary power output are on the enclosure opposite to the Pi4 LAN and USB interfaces.

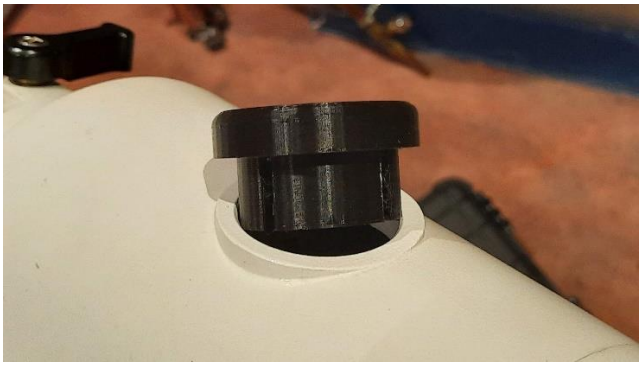
<<IMG>>

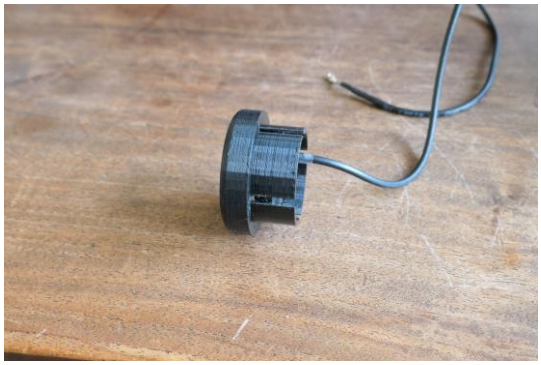
The PCB stack is enclosed in a 3D printed box, which fits into the hole of the original service panel. The NEQ6 housing appears to have plenty space to allow for this.

<<IMG>>

All plastic parts are 3D printed in PETG and PLA.

1.3.2 GPS antenna

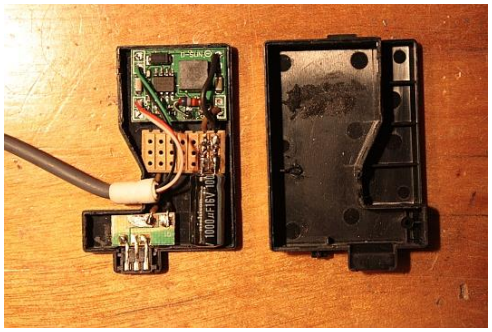




The GPS antenna module has been built into a plug that replaces the cap for the polar alignment scope. You can still take this antenna plug out allowing polar alignment of the mount.

1.3.3 DSLR Power:

For powering the Canon 450D directly instead of a battery (which will run empty), I found a cheap plastic adapter on the web. This is nothing more than a battery shaped enclosure that just contains a pair of elco's, which should be powered from an external net adapter.



Since I wanted to connect this directly to the 12V outlet of the NEQ6-Next unit, I pried the adapter open and used the empty space to put in a small buck converter. The voltage setting potentiometer did not work, and was replaced with a suitable fixed resistor, to yield a 7.6V output voltage. I re-used one of the elco's and added a 100nF capacitor for further filtering purposes.

1.3.4 Game controller / Joystick:

For corrections and scanning the sky a gamepad can be used as a joystick. This could be a Super Nintendo SNES controller with a USB interface:

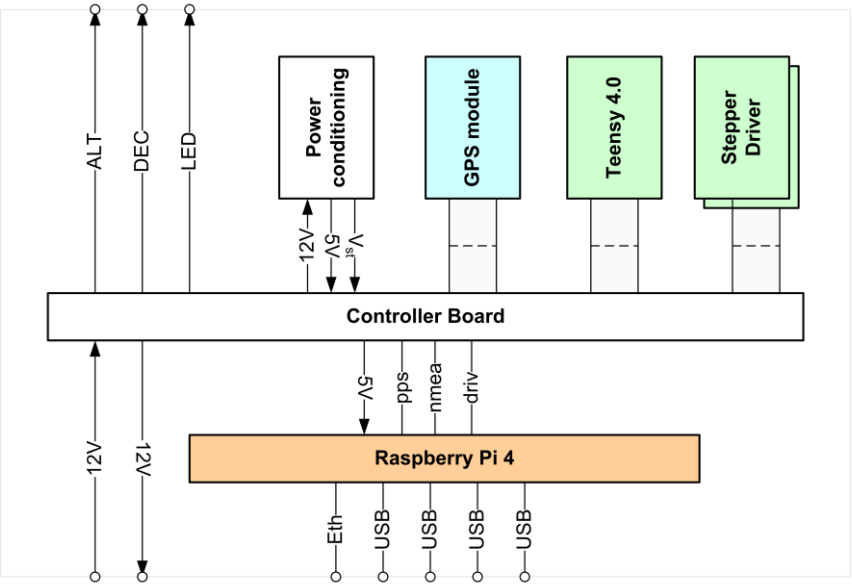
The version I had broke down, so I decided to build a more robust one based on an Arduino Micro and a handful of buttons. You can use any other, as long as it is USB interface and recognized by the Rpi OS.

2 Controller board

This section discusses the motor controller board that is plugged onto the RPi4 module.

2.1 Electronics

The block schematic diagram of the controller/connection board circuit:



The board will have 12V input from battery or PSU and an output 12V to supply e.g. a camera. There are two stepper drivers, one for each axis, connecting to the existing NEQ6 RA and DEC stepper motors. The stepper drivers can be A4988, TMC2130 or TMC5160 modules, and possibly others as well (see OnStep website). The board also hosts a NEO-8 GPS receiver, which provides NMEA-0183 and 1PPS interfaces for locations and precise timing.

The Teensy 4.0 hosts all OnStep software, that drives the steppers. It also interfaces to the RPi4 through a LVTTL serial interface UART, via its utility connector. The GPS module interfaces with the RPi4 via another serial interface. The 1PPS signal from the GPS module is connected to the RPi4 via a GPIO pin.

The RPi4 hosts an Ubuntu server that does all the heavy lifting, controlling the OnStep software running on the Teensy 4.0.

Power consumption:

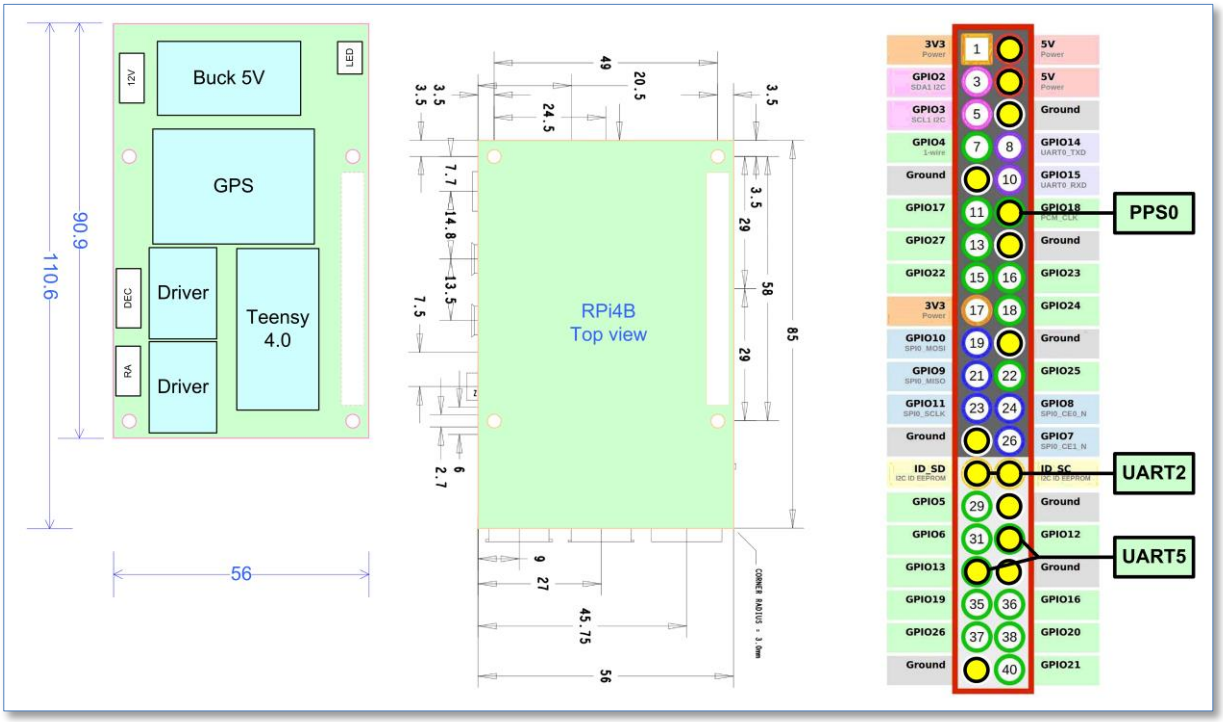
- The *Raspberry Pi 4B* requires about 3A. This is divided over the processor itself and to support the powering of connected USB devices, worst case adding up to 1.2A.
- The *Teensy 4.0* module requires 5V, 250mA. This includes the 3V3 outputs.
- The *Stepper Driver* logic require 3V3, max 20mA, supplied by the Teensy.
- The stepper motor voltage is switched by the drivers, and supplied by a separate boost converter.
- The *GPS NEO8* module requires 3V3, max 100mA, supplied by the Teensy.

There is an on-board buck converter that provides the 5V/5A required by both the Raspberry Pi 4B and the Teensy 4.0. The 3V3 for the other modules is taken from the Teensy 4.0 output, since the required current is fairly low. The stepper voltage is provided by a boost converter that delivers up to 36V/2A.

External interfaces are for:

- Stepper motors (RA and DEC)
- LED for Polar alignment reticule.
- Power input and output (12V)
- USB from RPi4 (2x USB2, 2x USB3)
- Ethernet from RPi4B

Layout of component on the controller board, compared to the RPi4B it will be plugged on to:



As said, the controller board plugs into the 40-pin utility connector of the Raspberry Pi 4B, as shown in the above image. The 5V buck converter is located next to the GPS module, while the 33V boost converter is on the backside of the PCB. The sandwich is fit into the NEQ6 stepper housing, upside-down, so that the RPi4 is on top and the business connectors point into the housing.

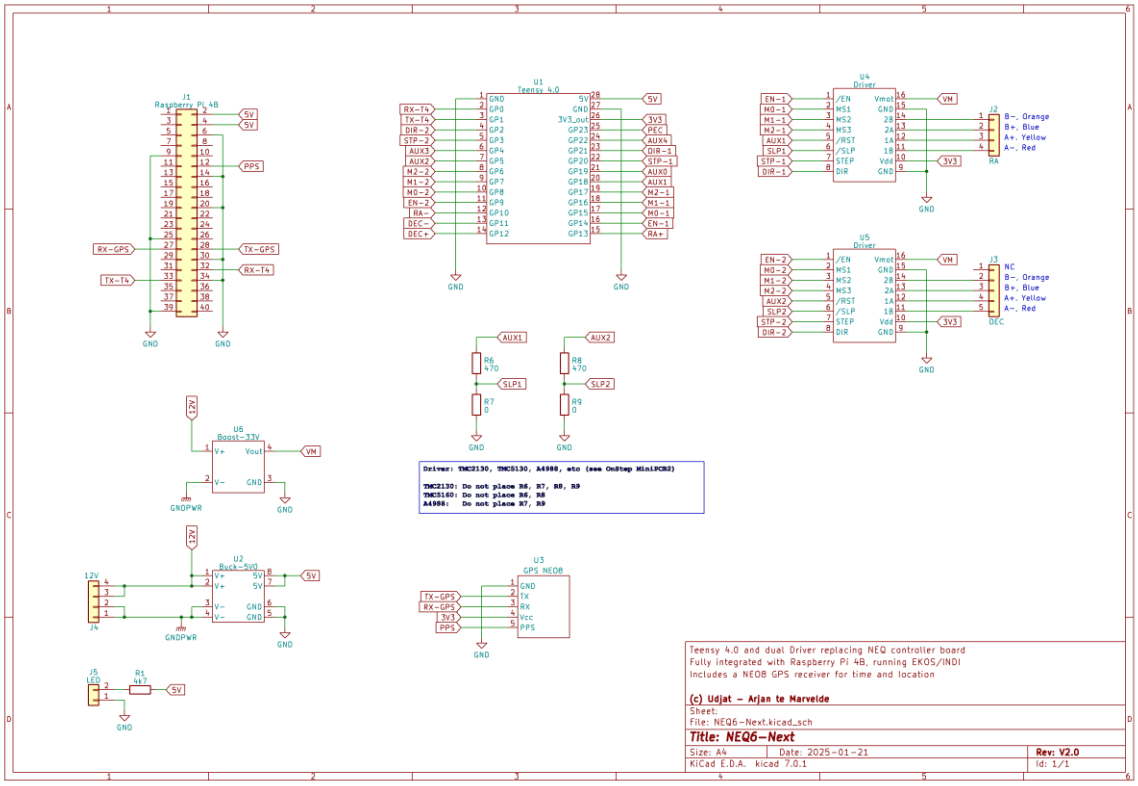
The controller board has connectors for the Dec and RA steppers, the 12V I/O interface and the reticule LED supply. The connectors are compatible with those used on the original controller board. Optionally the ST4 interface can also be used, but in my mount I will rely on PHD2.

The controller+RPi sandwich fits into a 3D printed encosure that replaces the NEQ6 front plate.

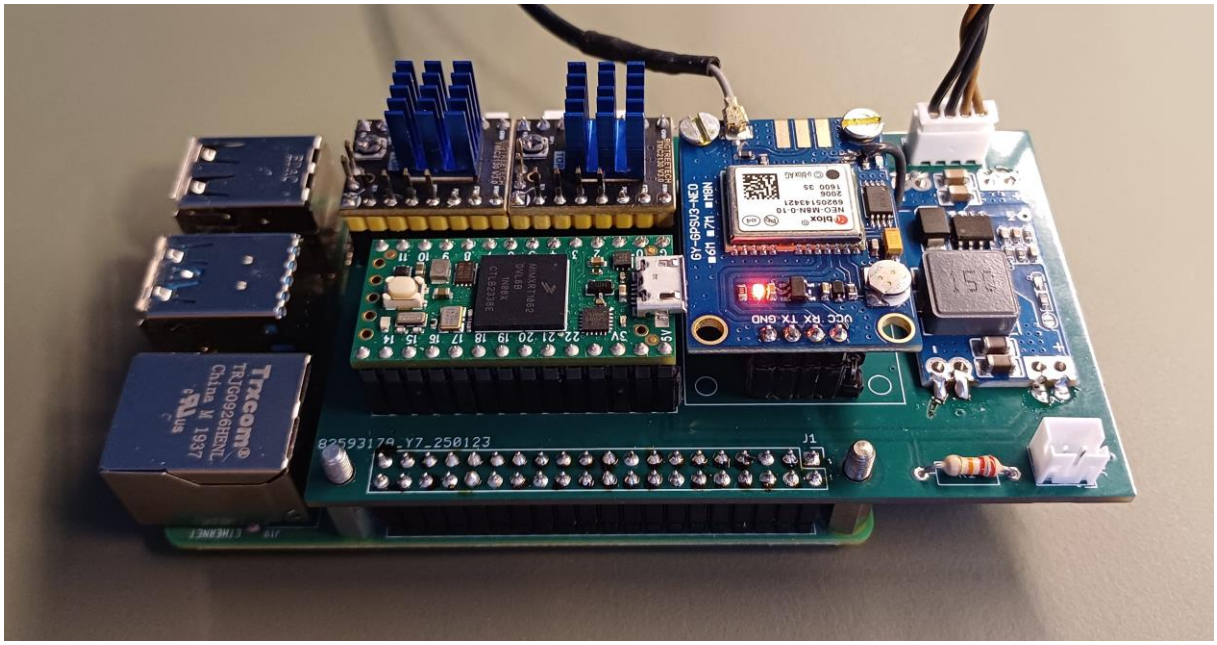
The utility connector pin usage:

- 3V3 out: pin 1 → to GPS and level shifters
- Vcc in: pin 2, 4 → from DC-DC converter
- GND: various (all) → common ground
- GP18: pin 12 (GPIO 18) → to GPS PPS
- Tx2: pin 27 (GPIO 0) → to GPS RxD
- Rx2: pin 28 (GPIO 1) → to GPS TxD
- Tx5: pin 32 (GPIO 12) → to NEQ6 RxD
- Rx5: pin 33 (GPIO 13) → to NEQ6 TxD

The electronics schematic diagram:



As can be seen, it is really very simple: below the arrangement of the modules is shown, installed on the PCB which plugs onto the RPi4.



2.1.1 Bill Of Materials

The controller board is mostly built up from modules. Below there is a list of these modules and place where I found them. Probably similar variants can be ordered directly from other sources.

Shop link: https://www.tinytronics.nl/index.php

Module	SKU (item number, use in search)
Boost converter 12V – 33V	000074
Buck converter 12V – 5V	005590
Bigtreetech Stepper driver TMC2130 V3.0	005543
Teensy 4.0	002468
GPS NEO-8	004332

For the GPS module I have tapped the PPS signal from NEO pin 3, by scratching the silklayer from the track and soldering on a wire:

2.2 OnStep installation

The OnStep software needs to be configured, built and loaded onto the Teensy module. Follow the instructions on the OnStep wiki to get this done (https://onstep.groups.io/g/main/wiki/32779).

Summarizing:

- Install the latest Arduino environment (https://www.arduino.cc/en/software)
- From the Arduino IDE install the Teensy support package
- The Neq6-Next board is pretty basic, so no additional libraries needed
- Get the OnStep code zip and unpack in a project folder (https://github.com/hjd1964/OnStep)
- Backup and adapt Config.h in the project folder, to match the actual HW
- Build the target SW
- Load the Teensy

2.3 Configuration

The settings that need to be set in **Config.h** are mostly according to MiniPCB2, which has served as input for NEQ6-Next schematic. Note that Axis 1 is connected to RA and Axis 2 to DEC stepper. The steppers themselves do 200 steps per revolution and are originally driven with 64 microsteps. The total gear ratio is 705:1 of which the worm ratio is 180 and the drive gearing ratio is 47:36(or 32) and 36(or 32):12. Best to use the spreadsheet to calculate paramaters.

Parameter	Value	Comment
PINMAP	MiniPCB2	
ST4_INTERFACE	OFF	
SLEW_RATE_BASE_DESIRED	2.0	
AXIS1_STEPS_PER_DEGREE	25066.667	200 steps, 64usteps, GR1=47/12, GR2=180
AXIS1_STEPS_PER_WORMROT	0	PEC is off, otherwise 50133
AXIS1_DRIVER_MODEL	TMC2130	V3.0
AXIS1_DRIVER_MICROSTEPS	64	
AXIS1_DRIVER_REVERSE	OFF	Set to ON when direction is wrong
AXIS2_STEPS_PER_DEGREE	25066.667	
AXIS2_STEPS_PER_WORMROT	0	PEC is off, otherwise 50133
AXIS2_DRIVER_MODEL	TMC2130	V3.0
AXIS2_DRIVER_MICROSTEPS	64	
AXIS2_DRIVER_REVERSE	OFF	Set to ON when direction is wrong

3 Raspberry Pi 4B installation

Before this I have used Ubuntu as platform, but while some things have made installation a lot easier, other things just cost too much time to get them working. So this guide uses the Raspberry Pi OS instead, which currently is based on Debian Bookworm. It is lightweight and has VNC support already built-in. Setting up everything is actually very easy, a lot has been pre-installed already so it is just a matter of configuration.

Note: Not in scope of this guide, but another possibility is to install Ubuntu Server instead of Desktop. The lightweight MATE desktop can be added to the server afterwards, instead of replacing the default Gnome desktop manager.

The following steps will be followed:

1. Install Rasberry Pi OS on the RPi, including remote desktop and HW overlays
2. Tweak the networking,
3. Install the GPS and Time drivers, as well as an FTP server
4. Install applications: EKOS/KStars/INDI and drivers, Astrometry, PHD2, etc

Attributes in yellow background, used in text or code examples should be customized to your own situation.

3.1 Install OS

3.1.1 SD-card preparation

The easiest way to prepare the SDXC card: use Raspbery Pi Imager to download and format/write your card in one go. Select the Pi4, Raspberry Pi OS (64 bit) and the micro SDXC card plugged into your PC. I use a very fast 64GB one, to speed up local image handling.

In Raspbery Pi Imager select the option to edit the OS settings before writing it to the SD card, enabling to already initialize some basic settings like user name and password.

3.1.2 Configuration

Switch off Bluetooth. In the network configuration (↑↓ icon upper right) setup a hotspot and change the LAN address to be static. This is a bit easier for connecting to the Pi with FTP, SSH or VNC viewer.

After reboot, a notification about updates is given; best to install those now. Otherwise further down the process updates can always be installed by:

```
sudo apt update
sudo apt upgrade
```

Then open the Raspberry configuration tool:

```
sudo raspi-config
```

- Under Interface Options: enable the SSH server.
- Under Advanced Options: expand the Filesystem and select X11 instead of Wayfire as display manager.

Exit the tool with reboot, then open the Raspberry configuration tool once more.

- Under Interface Options: enable the VNC server.
- Under Display Options: change the VNC resolution to your liking (for headless operation)

Reboot again.

Now RealVNC connect is running on the Pi and an icon is visible in the taskbar top right. RealVNC viewer can now be tried to connect from a workstation. If that works the HDMI display (and keyboard/mouse) can be disconnected and then reboot.

Now you can do further installation by using SSH (with PuTTY) or via remote desktop with RealVNC viewer.

3.1.3 Overlays

The specific hardware attached to the 40 pin utility connector requires some special OS setting to enable access to the interfaces. A couple of overlays need to be loaded at startup, to configure the different pin mappings enabling UART2 (GPS), UART5 (OnStep controller) and the GPIO18 (PPS). To load these overlays during boot, the file **`/boot/firmware/config.txt`** needs to be edited.

At the end of this file add the lines:

```
dtoverlay=pps-gpio,gpiopin=18
dtoverlay=uart2
dtoverlay=uart5
```

After a reboot the utility connector pinning is handled correctly by OS devices and accessible under **`ttyAMA2`**, **`ttyAMA5`** and **`PPS0`** respectively.

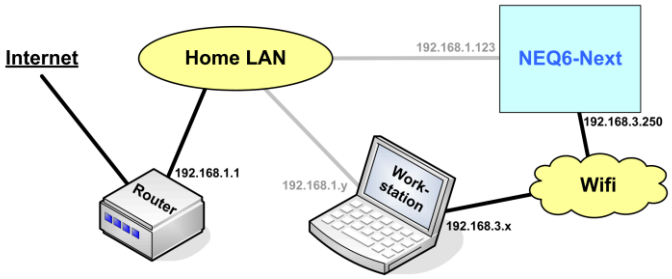
You can check whether the connected GPS is actually working by trying:

```
sudo cat /dev/ttyAMA2
```

The stream of NMEA strings will then be roll over the screen.

3.1.4 Networking

The intended network topology allows for stand-alone operation in the field, with just the workstation and the NEQ6-Next system. But it can also be hooked up to the wired home LAN network, where the workstation might also be connected. For this reason, the WiFi interface will be set up as an *access point* while upstream traffic is routed to the Ethernet interface.



Intended network layout

The NEQ6-Next Ethernet interface address is static, matching the home LAN. In case the Ethernet is not connected, the INDI box operates in stand-alone mode and there will be no internet access. The Workstation, running the remote virtual desktop, normally connects through the WiFi AP, but when the INDI box Ethernet is connected to a host LAN, it can also reach the internet through the INDI box router. Alternatively, the Ethernet interface can be used to connect the virtual desktop device via the home LAN.

The Raspberry Pi OS makes network configuration easy, in fact this has already been put in place when the OS was installed and configured.

Hotspot address range

One thing that needs to be settled is a defined IP address for the NEQ6-Next hotspot, which makes VNC or SSH connection from the workstation a bit easier. Since there is a fixed IP address on the LAN side, this can be used as SSH/VNC server address too.

Hotspot autostart

Another thing is that the hotspot doesn't start automatically. To accomplish this (as per Raspberry support page) we need to use the network manager cli:

```
nmcli connection
```

This shows the interfaces along with their UUID, copy the UUID associated with the hotspot interface. Then issue a new command:

```
nmcli connection show <UUID>
```

In the output there will be lines like:

```
connection.autoconnect:                 no
connection.autoconnect-priority:        0
```

These need to be changed into 'yes' and '100' respectively:

```
sudo nmcli connection modify <UUID> connection.autoconnect yes
sudo nmcli connection modify <UUID> connection.autoconnect-priority 100
```

Reboot and try.

For debugging NetworkManager issues try:

```
journalctl -b | grep NetworkManager
```

Note: After much frustration I found that the power save option of the WiFi connection is default enabled, understandable as subscriber but not as an access point. Test this setting with:

```
iw wlan0 get power_save
```

If it is enabled you can switch it of with:

```
nmcli c modify <connection name>  802-11-wireless.powersave 2
```

3.2 OS extensions

A few extensions need to be installed and tested:

- GPS handling with gpsd
- Time server with chrony
- FTP server for offloading images

3.2.1 Location and Time servers

The telescope location will be derived from a GNSS source, a NEO 8 receiver. This receiver is conencted to **/dev/ttyAMA2** and will be handled by **gpsd**.

For the precise time also the **/dev/pps0** devise for PPS pulses is used. The PPS signal can be tested with **ppstest**. First install **pps-tools**:

```
sudo apt install -y pps-tools
```

Then test by issuing:

```
sudo ppstest /dev/pps0
```

This produces a message each second, with incrementing seconds counter.

To install **gpsd** execute:

```
sudo apt install -y gpsd
```

Next edit the file **/etc/default/gpsd** to conform with:

```
START_DAEMON="true"
USBAUTO="false"
DEVICES="/dev/ttyAMA2 -s 9600 /dev/pps0"
GPSD_OPTIONS="-n"
```

Now reboot and check the working with **cgps**. There should be activity, and at some point a location fix is made.

We will use **chrony** as time server since it is better than NTP. Itcan be installed with:

```
sudo apt install -y chrony
```

Now chrony needs to be configured. Open **/etc/chrony/chrony.conf** for editing.

Add the following lines under the pool line:

```
#pool 2.debian.pool.ntp.org iburst
refclock SHM 0 refid NMEA offset 0.000 precision 1e-3 poll 3 noselect
refclock PPS /dev/pps0 refid PPS lock NMEA poll 3
```

The pool line itself can be commented out. Also let chrony act as time server by allowing access. Add the following at the end of **chrony.conf**:

```
allow
```

You can be more restrictive by adding e,g, the local LAN range, like 192.168.0.0/24, but in stand alone or home LAN mode this is not really needed.

Save the file and restart the **chrony** service:

```
sudo systemctl restart chrony
```

3.2.2 FTP server

Best is Very Secure FTP daemon, so to install this:

```
sudo apt install vsftpd
```

In the configuration file /etc/vsftpd.conf set (uncomment) the following:

```
anonymous_enable=NO
local_enable=YES
write_enable=YES
local_umask=022
```

Restart the service and make sure it will be started automatically after a reboot:

```
sudo service vsftpd restart
sudo systemctl enable vsftpd
```

Now perform another reboot and test the FTP access from a PC with for example Filezilla.

3.3 Applications

Now the basic platform is running, the business applications can be installed. Instructions can be found on the different installation web pages, below the steps are described for:

- INDI-Kstars
- PHD2
- Astrometry
- SkyChart (Cartes de Ciel)
- CCDCiel
- Other

3.3.1 INDI – Kstars – PHD2 – Astrometry

Since this is not Ubuntu, the regular way of installing INDI EKOS and Kstars from PPA does not work.

Instead I followed the guide on this page: https://gitea.nouspiro.space/nou/astro-soft-build

I cloned the git repo in the **<user>** home directory, so it all ends up in **~/astro-soft-build**. Then **cd** to that directory and install the dependencies and build the stable branch. This takes a long time since everything needs to be built from sources.

PHD2 is also included in the above installation, but needs to be separately passed to the build script:

```
./install-dependencies.sh
./build-soft-stable.sh phd2
```

To install updates you need to do a git pull and then repeat above steps.

```
git pull origin
```

Some configuration hints:

Use the Ubuntu **gpsd** service as direct source of location and time for EKOS. In EKOS, select the **gpsd** device under *Aux* → *Others*.

The mount to be used is *SkyWatcher* → *EQMod*. To get the mount working, the right port has to be chosen to connect. The device **ttyAMA5** is connected to HW **UART5**, and available on pins 32 and 33 (GPIO12 and 13). This serial port device needs to be selected in the *EQMod Mount / Connection* tab in the Kstars control panel.

The Astrometry plate solver can be used to accurately align the telescope in a very easy way. In above installation it is already included, so no need to separately install.

3.3.2 SkyChart – CCDCiel – indistarter

Find the website tyo get the instructions for installation:

https://www.ap-i.net/skychart/en/documentation/installation_on_linux_ubuntu

Follow the commands on that page, which also work for the Raspberry Pi OS:

```
bash <(wget -qO-
       https://raw.githubusercontent.com/pchev/skychart/master/setup_skychart
       _deb.sh)
sudo apt update
sudo apt install -y skychart
sudo apt install -y ccdciel
sudo apt install -y indistarter
```

3.3.3 Other settings

For the game controller a driver is already available in the Raspberry Pi OS installation, for testing the controller it can be loaded as an INDI device in Kstars.

For specifc device testing the following package can be used:

```
sudo apt -y install jstest-gtk
```

This test package shows the numbers for the different buttons, to which KStars will refer.

4 Belt drive upgrade

Future update.

5 Some References

The INDI tutorials ("Painless remote control with Ekos/INDI")

http://indilib.org/support/tutorials.html

INDI Forum, (search for Pi 4 or Pi 5)

https://www.indilib.org/forum/index.html

INDI installation for Debian

https://gitea.nouspiro.space/nou/astro-soft-build

Raspberry Pi OS for the RPi4 (64 bit):

https://www.raspberrypi.com/software

Terminal emulator for SSH, PuTTY:

http://www.putty.org/

Onstep parameters spreadsheet:

https://baheyeldin.com/sites/baheyeldin.com/files/OnStep-Calculations.xls