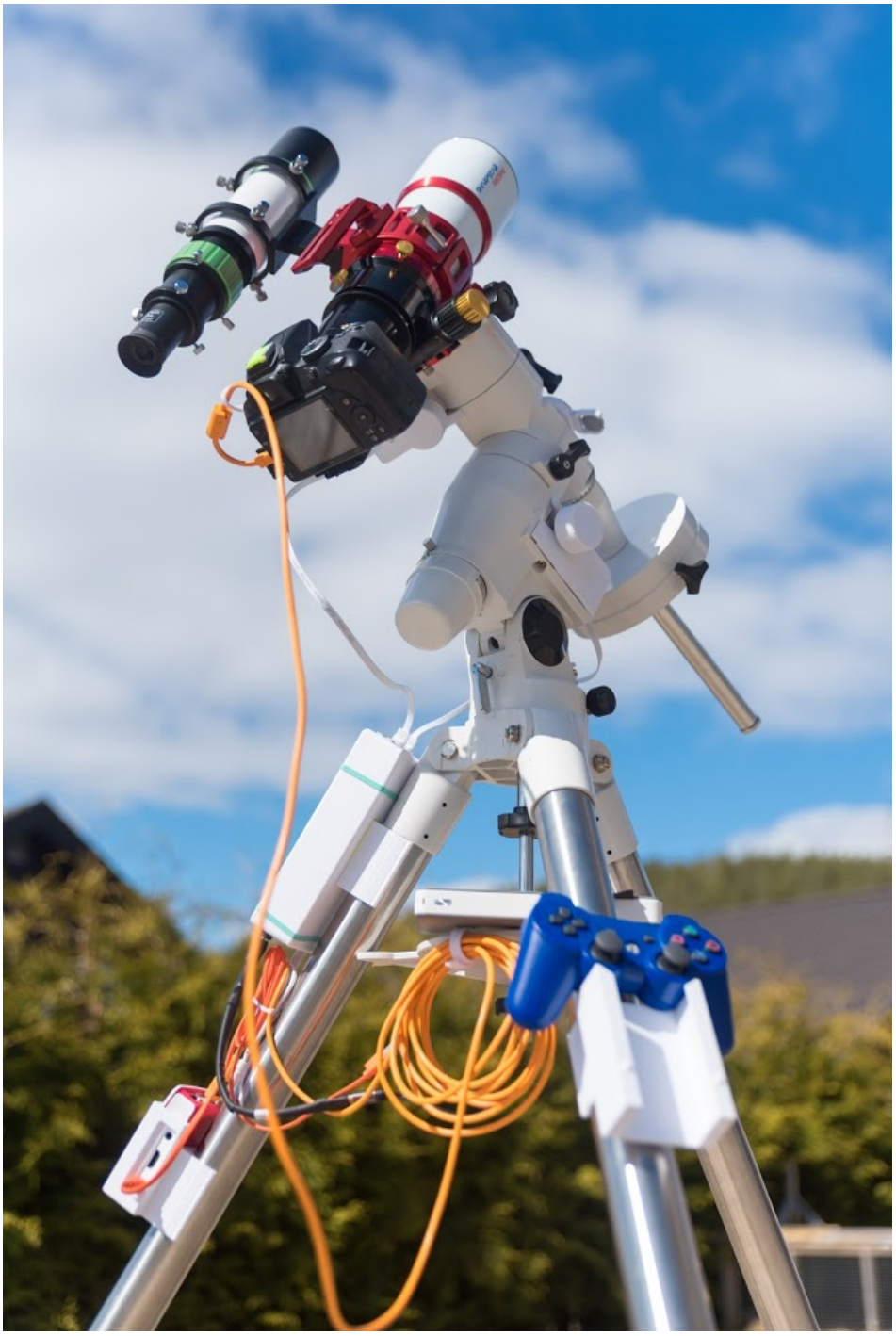
OnStep EQ5 GoTo

BUILD INSTRUCTIONS



By Oddvar S. Næss

Revisions

Version	Reason for revision	Date	By
A	First issue	11 May 2021	OSN

Information

This instruction contains information not essential for building a OnStep GoTo System. For readers that want to jump right into the building instruction please go directly to the instructions page.

The following building instructions are mainly based on my experience when modifying my old Sky-Watcher EQ5 mount, but can also be adapted to many other mounts. Main steps are general, but items and configuration might be mount/controller specific. Please refer to the OnStep Wiki for other controllers and mounts.

Content

Background

I've always had an interest in astronomy as well as photography. Right up until the coronavirus hit us I was using some of my spare time doing wide angle Milky Way photography, going to remote locations with minimal light pollution to maximise the results. However, as the society was gradually shut down during the increased corona restrictions applied by the Norwegian gouvernments I started to investigate how I could pursue this hobby from my home. I was also in need of some in-house projects to burn off some of the free time I had these days. Soon after I was searching for used second hand telescope mounts, preferably the Sky-Watcher EQ5. When the mount was found and bought the intention was to use this mount with my original DSLR and lens, but I got carried away and bought a used SharpStar 61EDPH (Mk I) and EvoGuide 50ED as well. This should be a good beginner starting point for some wide field astrophotography.

The mount came with the original RA Motor Drive system. When testing the new setup from our porch this seemed to work quite well. However, slewing took forever and finding targets that I could not easily see through the finderscope was next to impossible (nebulas etc.). I thought I could live with this, but that was just until I realised how long it took to make all the photos needed for some decent stacking results later on in the process. I had quite a few freezing nights outside (this was during winter time in Norway) before I realized I had to look for alternatives, else I would probably give up this new hobby of mine.

Based on my experience so far I had a few ideas of what I wanted to achieve now:

- A system I could remote control from the warm inside of my house or car
- It had to be able to run on battery power for some hours
- It should function as a stand-alone system (no internet required)
- Ideally modular and portable

A few evenings of googling made me bump into OnStep and a few months later I could finally confirm I had a fully working GoTo mount and celestial tracker that gave me very satisfactory results. Not only that, but connecting the OnStep controller to a Raspberry Pi running with Astroberry made for an impressive system overall.

Looking back at all the time I invested in finding solutions, information, how-to's and such I thought I would try to help others in the same situation starting from scratch with little or no knowledge of electronics nor astronomy. During this project I've made notes along the way and did partial documentation of the process and the parts used for my specific configurations. Based on these I decided to make this build instruction in the hopes it might be of help for others finding themselves in the situation that I did.

NOTE: Please keep in mind that this is done with my best intentions, but remember that I am also just an amateur - most of the things involved related to this OnStep controller are like magic to me so I would strongly recommend that you do your own research and read the Wiki thoroughly.

Building instructions

OnStep Controller

Bill of materials

Item:	Qty.:	Price ($)	URL
WEMOS D1 R32 ESP32 WiFi BT Development Board	1	7.00	LINK
Arduino CNC V3 Shield	1	3.50	LINK
LV8729 Drivers	3	18.00	LINK
NEMA17 Stepper Motor 0.9deg 1.68A 40mm*	2	34.00	LINK
GT2 12t 6.35mm bore 6mm belt pulley (DEC shaft)	1	4.00	LINK
GT2 16t 6.35mm bore 6mm belt pulley (RA shaft)	1	3.00	LINK
GT2 48t 5mm bore 6mm belt pulley (Stepper shafts)	2	9.00	LINK
GT2 6mm 150 Closed Loop Timing Belt (RA Axis)	1	4.00	LINK
GT2 6mm 140 Closed Loop Timing Belt (DEC Axis)	1	3.00	LINK
RJ45 breakout boards (Pack of 6pcs, but only 5 needed)	6	6.00	LINK
RJ11 / RJ12 / 6P6C Breakout Board	1	12.00	LINK
NodeMCU ESP8266**	1	3.00	LINK
DS3231 Real Time Clock***	1	1.50	LINK
Set of Dupont 20cm Jumper Wire (FF/MM/FM)	1	8.00	LINK
Set of 5.5x2.1mm DC Power Plugs (F/M)	1	5.50	LINK
Total cost		**121.50**	

* Cheap stepper motor, but not the most efficient one. Based on some forum posts there seems to be versions of this motor rated for 0.9A only with similar performance. This should probably be considered if you're aiming for maximum energy efficiency (ie. maximum battery life etc.)
** I used a NodeMCU ESP8266 DevBoard I already had from an earlier project (and it's been dead stable), but the Wiki recommends the Wemos D1 Mini version.
*** I'm not really sure if this RTC is adding any real benefit or improvements to this system, but since these cost next to nothing I figured it was best to include it just to be on the safe side.

Software

Arduino IDE and configuration files

1. Follow the OnStep Wiki instructions carefully for setting up Arduino IDE correctly for flashing both the Wemos R32 and the ESP8266 (Link and Link). Pay special attention to Library versions!
2. Download OnStep V4.x Master File and extract to default Arduino sketch folder (Direct Download Link)
3. Download and populate the Configuration Calculation sheet with correct values for your mount, stepper motors and gears (Direct Download Link):
    a. 400 steps motors
    b. 32 microsteps
    c. Axis 1 Gear Ratio 1 with my setup = 3 (48/16)
    d. Axis 1 Gear Ratio 2 for EQ5 = 144 (EQ5 internal worm gear)
    e. Axis 2 Gear Ratio 1 with my setup = 4 (48/12)
    f. Axis 2 Gear Ratio 2 for EQ5 = 144 (EQ5 internal worm gear)
    g. Desired Slewing Speed = 3
4. Then we need to use the Web Configurator for making a new config.h (Link):
    a. 4.x Masterfile selected
    b. Equatorial mount selected
    c. Desired Slewing Speed = 3
    d. Select 400step 0.9deg motor for both axis
    e. Select 32 micro steps
    f. Axis 1 Gear Ratio 1 with my setup = 3 (48/16)
    g. Axis 1 Gear Ratio 2 for EQ5 = 144 (EQ5 internal worm gear)
    h. Axis 2 Gear Ratio 1 with my setup = 4 (48/12)
    i. Axis 2 Gear Ratio 2 for EQ5 = 144 (EQ5 internal worm gear)
    j. DS3231 I2C RTC enabled
    k. ST4 Port for guiding enabled
    l. All else is left at default or “NO”
    m. Generate the “Config.h” file and replace the Config.h file in the folder from step 2.

Flashing instructions

5. Flashing the Wemos R32 with Arduino IDE:
    a. Connect the Wemos R32 to the computer and start Arduino IDE
    b. Open the onstep.ino file in the folder from step 2.
    c. Identify correct board type (ESP32 Dev Module) and com port
    d. Select “Upload” to start the flashing process
    e. When flashing is done disconnect the Wemos R32
6. Flashing the ESP8266 Wifi Module with Arduino IDE:
    a. Connect the ESP8266 to the computer and start Arduino IDE
    b. Open the wifi.ino file in the folder from step 2.
    c. Identify correct board type (Generic ESP8266 Module) and com port
    d. Select “Upload” to start the flashing process
    e. When flashing is done disconnect the ESP8266
7. That's it for the software part. Now we need to connect everything...

Electrical assembly

Connection diagram

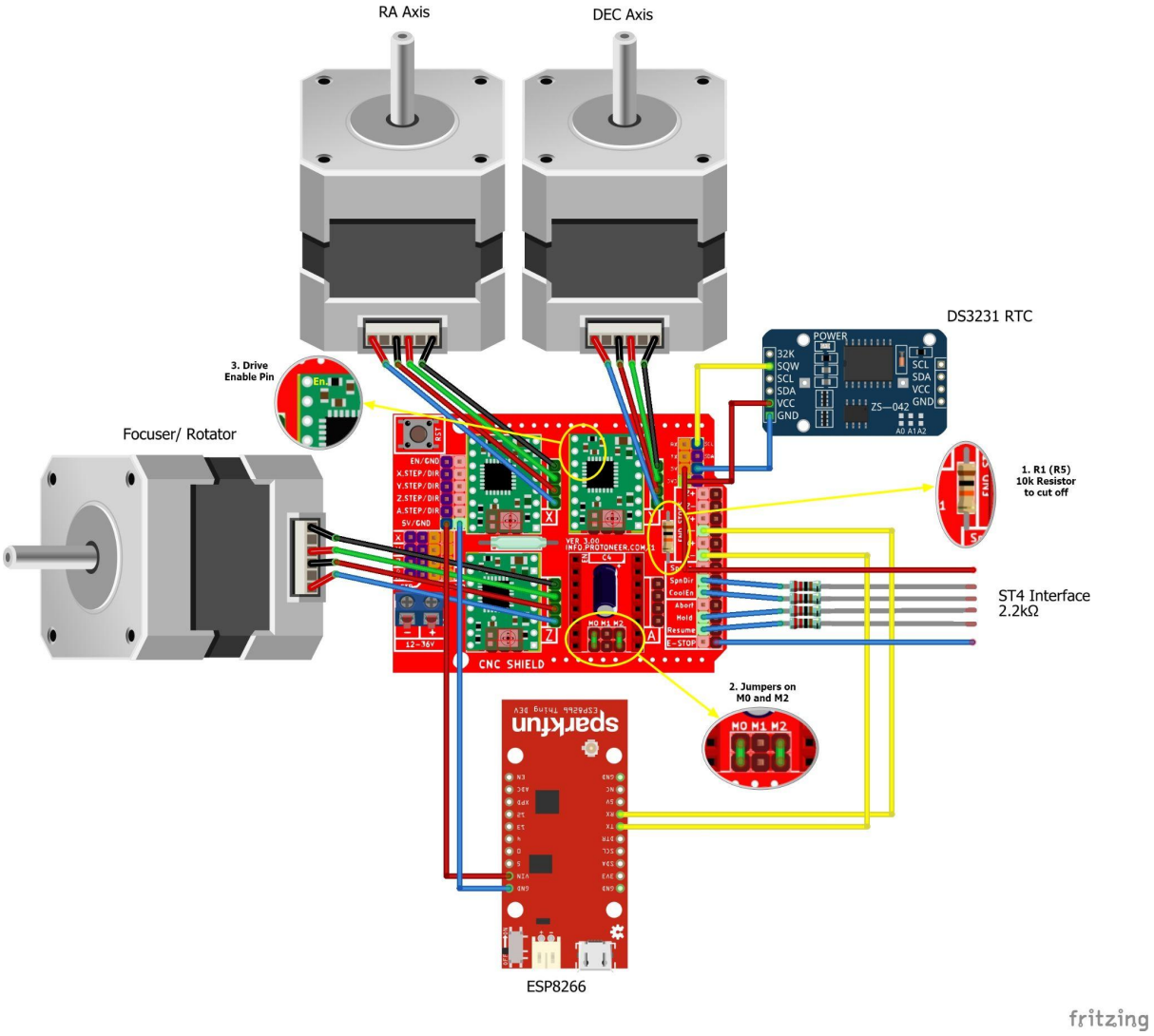


Figure 1 - Connection diagram

NOTE: Wiki instructions for the electrical connections are excellent so make sure to read it before following the instruction on the next pages (Link).

Assembly instructions

1. We'll begin by cutting off the R1 (R5) 10kΩ resistor on the Arduino CNC V3 Shield to remove the pullup to 5Vdc as the Wemos R32 is based on 3.3Vdc:

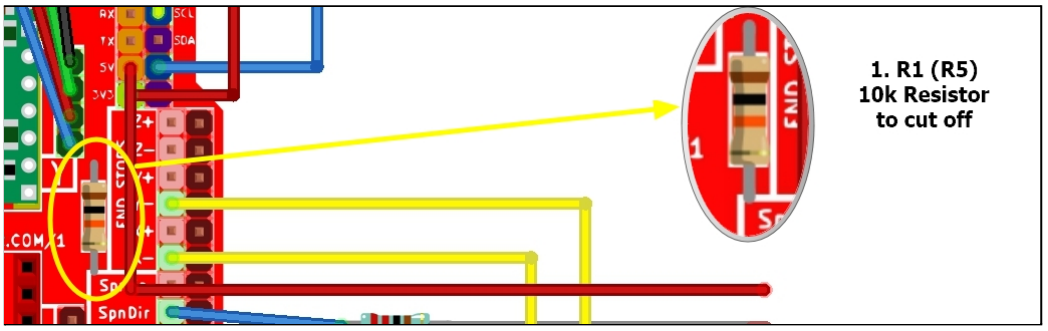


Figure 2 - R1 10k Resistor

2. The LV8729 drivers can be configured for 32 microsteps by inserting 2 jumpers on the CNC shield in slot M0 and M2 (also M1 and M3 on some versions). This must be done for each driver slot that will be used (leave the middle jumer slot empty - Link):

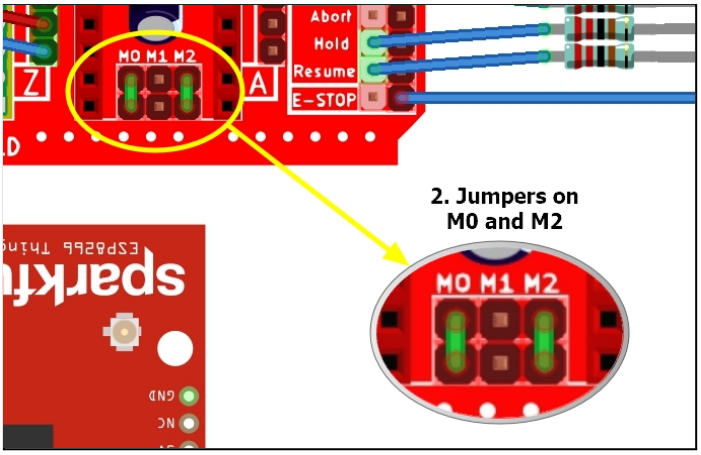


Figure 3 - Driver Slots Jumper Settings (M0, M1, M2)

3. Now insert all the LV8729 drivers into their sockets. Make sure that the “En - Enable” pin is located in the upper left corner and that the Vref adjustment screw is pointing down or towards you when you have the shield oriented in such a way that all the text is in its normal readable orientation.

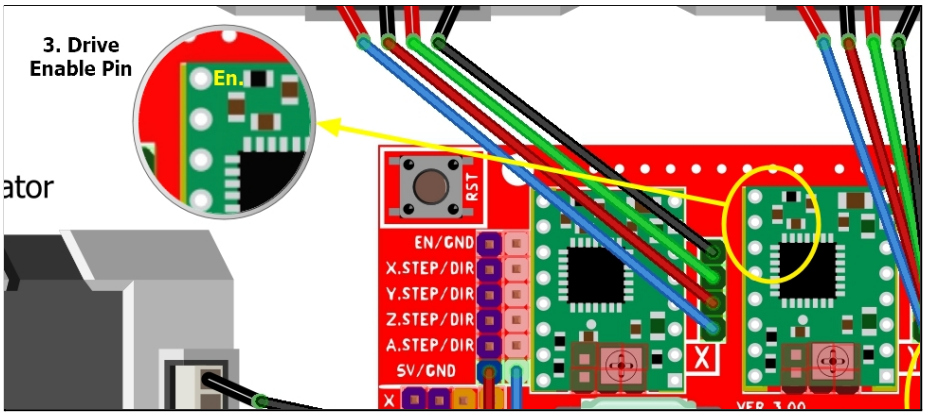


Figure 4 - LV8729 Driver orientation

4. Connect everything, except the motors, together according to the “*Figure 1 - Connection diagram*”.

5. Before we connect the motors we have to make sure to adjust the drivers correctly with regards to the motor ratings and the correct type of drivers. This must be done with the motors disconnected (Never connect or disconnect a stepper motor if the system is powered up!). Formula to find the current starting point for Vref is:

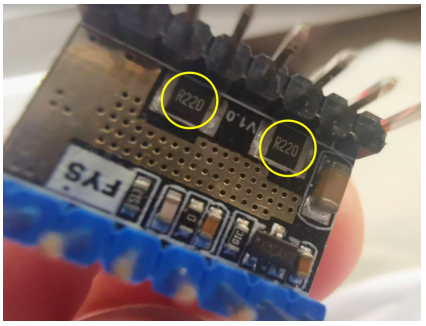
   Motor Max Amp * 1.41 * 0.4 = Starting Amp for Vref” (Link).

   In my case this would be: 1.68A * 1.41 * 0.4 = 0.95A

Note: The LV8729 has a current limit of 1.8A. Since the motor is rated for 1.68A this will be our limiting factor and we’re comfortably within the operational constraints of the driver. The general recommendation seems to be not to load the motors with more than 70%.

---

6. Now that we have the starting Amp for Vref we need to identify the actual Vref. However, the LV8729 seems to come in two flavours; some with 2x 100Ω resistors, and some with 2x R220Ω resistors. You find what version you have by checking the resistors on the underside of the drivers. Mine had 2x R220Ω.

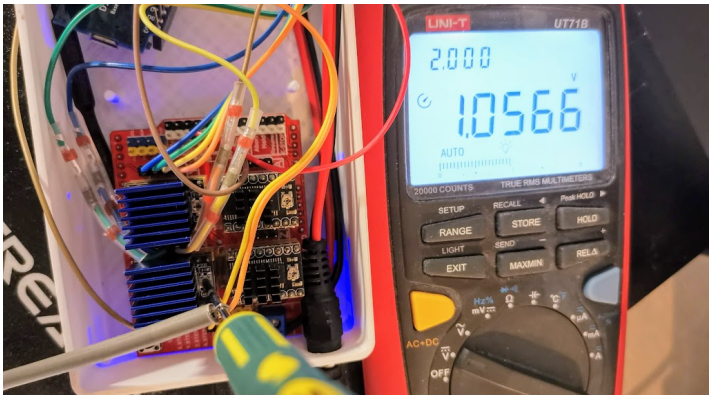


Now, by using the following formulas you can identify correct Vref (Link):

2x R100Ω: Desired motor Amps * 0.5 = Vref
2x R220Ω: Desired motor Amps * 1.1 = Vref

In my case this would be: 0.95A * 1.1 = 1.05V

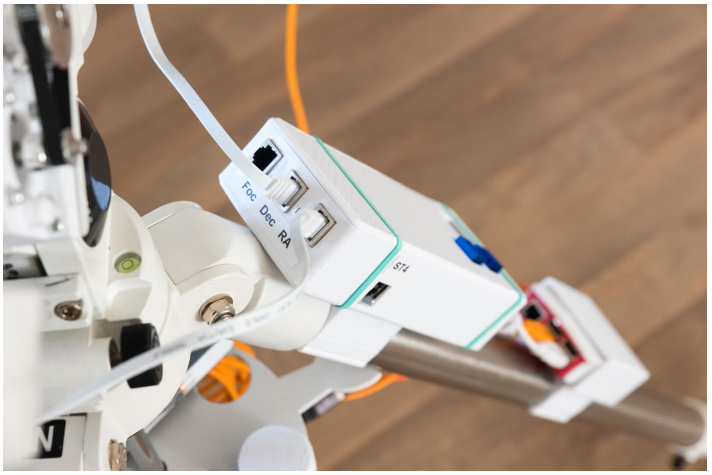
7. Adjusting the Vref potentiometer on the LV8729 driver is quite easy if you're able to connect a probe from the multimeter to one of the Gnd-points (ground) on the Arduino CNC Shield and the other probe to the skrewdriver you're actually adjusting the potentiometer with. That way you'll have a live view of the adjustments and you can easily hit your target.



Note: If you notice the motors are struggling when moving/slewing when you start to test the complete build with a fully loaded mount you can increase in 0.1A increments until you have stable movements. I ended up with Vref=1.1V (equals 1Amp and 60% of max motor rating).

---

8. Now we're ready to connect all remaining parts. This is mostly a straight forward job, and most parts can be directly connected with jumper wires, except maybe the stepper motors. Some motors are delivered with pre-made plugs/connectors on both ends, but others are not. However, in either case we probably need to extend these cables so we have some room to move the mount in all directions (this is especially important for the DEC Axis) relative to where we mount the OnStep Controller on our rigs. I used RJ45 Breakout Boards for this. This way I could simply connect or disconnect the motors with standard network patching cables.



9. Now you can test the system with all motors connected (not necessarily to the mount) and everything powered up. Verify full functionality by any of the possibilities (Link) and observe how the motors are reacting. If any of the motors are going in the wrong direction this can easily be rectified in the software; In Arduino IDE open the onstep.ino file and in the Config.h file locate the "#define AXISn_DRIVER_REVERSE" line (n is the affected axis 1,2,3 or 4) and set this to "ON". Save and re-flash the software to the Wemos R32. This axis should now move in the opposite direction.

Mechanically assembly

3D Printed parts

- Nema17 Brackets: https://www.thingiverse.com/thing:4853748 (Partial remix of the excellent https://www.thingiverse.com/thing:4682671 adjusted for the RJ45 ports)
- OnStep Case w/Bracket for EQ5: https://www.thingiverse.com/thing:4853768

Additional items complimenting my setup, but not related to OnStep directly:

- Raspberry Pi 4 with Astroberry as main server connected to the OnStep Controller (Link)
- Cheap Chinese PS3 Game Controller for manual slewing through Astroberry (Link)
- 3D printed Standalone Polar Scope LED illumination by a CR2032 battery (Link)
- 3D Printed Raspberry PI Case: Slim version of https://www.thingiverse.com/thing:3723561
- Raspberry PI w/EQ5 Bracket: https://www.thingiverse.com/thing:4853773
- 3D Printed Power Bank Bracket for EQ5: https://www.thingiverse.com/thing:4853776
- Everything is powered by a 20.000mAh PD 12V capable power bank with special 12V PD Trigger cables for the OnStep (Link and Link).

Telescope and camera:

- Main Telescope: SharpStar 61EDPH (Mk1) w/flattener
- Finder scope: Sky-Watcher EvoGuide 50ED (soon to be guide scope)
- Camera: Nikon D3300 and Nikon D750

Thank you!

Thanks to the creator and all the contributors of OnStep I now have a low budget, wireless, highly accurate tracker and GoTo system. I'm now able to polar align the mount, do a one star alignment and then go inside and do everything else from my cosy warm sofa - no more hours of freezing in below zero celsius here in Norway for these night time photos.