The TSC Telescope controller: Usage and Assembly

Wolfgang Birkfellner
with contributions and support from
S. Elste, S. Bruns, M. Sproul, U. Baron and R. Schulz
Home repository:
`https://github.com/selste/TwoStepperControl`
Homepage: `tscatm.wordpress.com`
Contact: `wbirkfellner@gmail.com`

February 3, 2020

Contents

Part I

Introduction

1 TSC – an open telescope controller for large telescopes

Figure 1: The telescope for which TSC was developed. The overall weight of the optical tube assembly is in the range of 80 kg, and all mobile parts weight approximately 300 kg. The mount is driven by two NEMA23 stepper motors with 2.8A maximum coil current

TSC (TwinStepperControl) was developed out of an desire to develop a freely available advanced telescope controller powerful enough to drive large telescopes with stepper motors. Wolfgangs own telescope is a Houghton of 12.5" aperture and 1300 mm focal length. It is mounted on an equatorial fork mount. The overall weight of the optical tube assembly is approximately 80 kg due to the

large front mounted corrector doublet, and the total weight of the mobile parts is estimated to be 300 kg. Many standard telescope controllers do not supply sufficient power to the drives to move an instrument of this size. In general, common driver components like the widespread Polulu DRV8825 are overwhelmed by stepper drives beyond the familiar NEMA 17 convention according to the common opinion in the development community of do-it-yourself 3D printers and CNC mills. In Section 7.1, this will be discussed to some further extent. However, a number of excellent do-it-yourself projects exist for smaller telescopes, and you are encouraged to check these to some further extent[1].

TSC is an open hard- and software project. It was initially based on two industrial grade stepper controllers (Phidget 1067 B bipolar stepper controller, www.phidgets.com with a maximum coil current of 4A and 30V maximum supply; the resolution here was fixed with 1/16 microsteps). In TSC, the driver boards are controlled via USB 2. The controllers are connected to a Raspberry Pi 3 B, B+ or 4 running Raspian Buster[2]. Fig. 2 shows the TSC controller in a improvised housing, together with one of the two wireless handboxes available.

As 1/16 microstepping is a little bit rough for commercial mounts with rather small gear ratios in the range of 1:900, a custom set of drivers was developed; these are based on the Pololu AMIS 30543 boards. Here, a maximum coil current of 3A is feasible, and the minimum resolution for microstepping is 1/128. After initial tests, it was decided to use these boards with a Teensy 4.0 microcontroller running at 400 MHz. Fig. 3 shows an earlier version of these two boards, which are communicating over a specialized protocol with the Raspberry. In the future, adoption to other stepper controllers over the conventional STP/DIR interface is possible.

The programming environment used for development of TSC is Qt[3] and C++. The functionality of TSC includes

- support for german equatorial and equatorial fork mounts. TSC provides compensation of the Earths motion and GoTo functionality. Lunar and Solar speed as well as motion inversion for the southern hemisphere are supported. A meridian flip for GoTo functionality can be enabled for German equatorials.
- support for internal catalogs and user-editable catalogs including synchronization of the mount to a given star. Currently, custom catalogs supplied with TSC include the Messier, NGC and IC catalogues, the Abell catalog of rich clusters of galaxies, the Arp atlas of peculiar galaxies, Barnards list of dark nebulae, a list of visible bright stars for both hemispheres, the Caldwell catalogue, the Herschel 400 and 2500 catalogues, the Hickson compact groups, Lynds list of bright and dark nebulae, the Shakbazian and Sharpless lists as well as the Yale catalogue of bright starts with SAO names.

[1] Check out `www.stellarjourney.com`, `rduinoscope.byethost24.com`, and `www.freego2.de`.
[2] `https://www.raspberrypi.org/`
[3] `www.qt.io/`

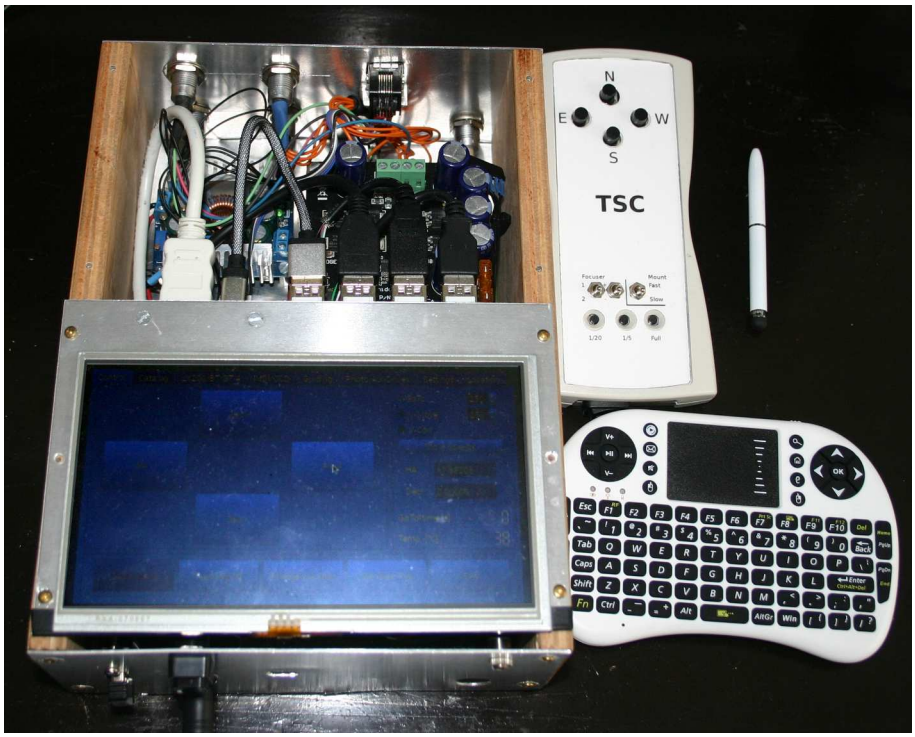


Figure 2: TSC in a very early version with a 7" touchscreen display, two high power stepper drivers, a secondary motor driver board, an internal 12V-5V block converter, and the wireless handbox. A miniature keyboard as shown is needed for configuration but is not required in routine use. Connectors include four motor connectors (two for right ascension and declination, two for additional focusers), a standard ST4 interface, a connector for an external temperature sensor, a USB outlet for connecting a guiding camera, a micro USB connector of an internal USB/Serial converter for serial LX200, and a 2.5 mm jack for connection a DSLR.

- support for telescope control and GoTo via the LX200 protocol and common planetarium programs such as Cartes du Ciel, Stellarium, SkySafari (on Android and iOS) and KStars. Connection is provided via autonomous WLAN (which is established if no other access point is available), or Ethernet and USB. Currently, TSC operates with permanent coil currents of 1.8A at 12V with my telescope, resulting in a GoTo speed of 100× sidereal speed. The top speed is dependent on the drives and gears used. On smaller drives, top speeds of 1000× sidereal speed were realized. As it is possible to operate the stepper drivers with voltages up to 30V, a further speed increase is possible. LX200 is also supported in the ASCOM 6.3 environment, therefore all programs utilizing ASCOM 6.3 such as SkyTechX can be used.
- support for a custom wireless WLAN handbox. The handbox allows for basic motion and control of the additional focuser motors.
- support for ST4, a standard protocol for autoguiding in astrophotography.
- an internal Autoguider connecting to standard guiding cameras via INDI[4].
- release control of time series for common digital single lens reflecting cameras including dithering of subsequent releases.
- a temperature compensated battery buffered clock.
- free configuration of mount parameters without any need for manual configuration files.
- control of two additional stepper motors for focuser drives.
- remote access to the controller via VNC from smart phones, tablets or computers.
- temperature measurement using an external sensor.
- full support of networking functionality over WLAN and Ethernet by using the interfaces of the Raspberry Pi.

2 Requirements and cost

In order to build TSC, one needs some basic understanding of computers, electronics, soldering and simple machining. Being familiar with the Arduino IDE and programming microcontrollers with this tool is an advantage.

The overall cost, probably not including all small materials like screws, and wires is approximately 135 € for the Raspberry Pi 4 and the additional hardware (the so-called HAT) – this also includes a 16 GB SD card with a pre-installed image of Raspian where TSC, all prerequisites and development tools are already installed. The stepper drivers are the single most expensive components with approximately 85 € for both. Denot that the choice of the Raspberry depends on the intended use. TSC works on Raspberry Pi 3 B+ models as well, but the

[4] `www.indilib.org`

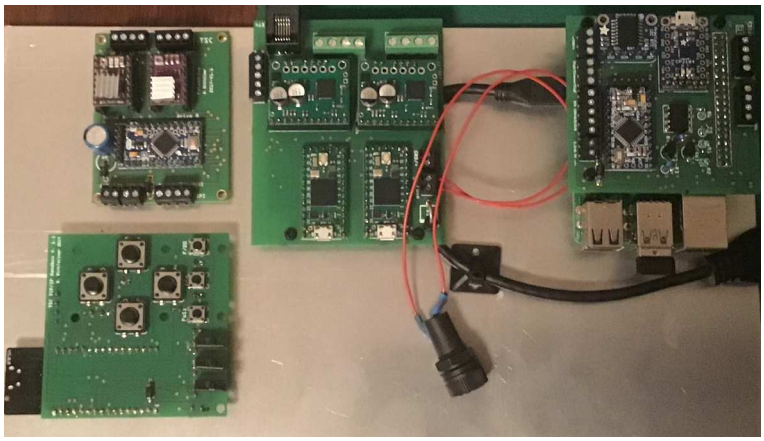


Figure 3: Some of the PCBs that make up TSC, including a 4 GB Raspberry Pi 4 mounted below the TSC-HAT, which forms the central component of the controller (right hand side). The custom driver board with two Teensy 4.0 microcontrollers (to be connected via USB to the Raspberry) and two Pololu AMIS boards is found in the top center. On the left hand side, the boards for the (optional) focuser driver and the (optional) wireless handbox can be seen. Not shown are the (optional) touchscreen for the Raspberry, the OLED dmounted on the wireless handbox, and a Step-Down converter for supplying the Raspberry with 5V and the motors with up to 30 V.

current range of Raspberry Pi 4 boards is preferrable. The RAM used depends on the intended use. The cheaper model with 1 GB RAM is sufficient for simple operation, but as the inclusion of plate solving using an internal copy of the astrometry.net engine is planned, the 4 GB RAM model is to be preferred.

A HDMI touchscreen with 800 × 480 resolution costs between 35 and 80 €. It is not compulsory but recommended. The wireless handbox accounts for approximately 35 €. If you want to use the focuser board, too, this is another 25 € in hardware. A detailed bill of materials is given in the home repository of TSC on `https://github.com/selste/TwoStepperControl` (see also Sect. 8). What is not included is the PCBs, which are also available in the repository. Here it makes sense to join forces as a small production run of 10 to 20 pieces can bring the price for the single board considerably.

PCBs and an image of the SD-card running the program can be requested from Wolfgang (wbirkfellner@gmail.com).

Part II

Functionality and System Description

3 A tour of TSC

TSC is still under development, but the functionality is basically available and while a few glitches are still to be resolved before β-release, TSC operates for almost 2 years by now. This section introduces the main components:

3.1 The main screen

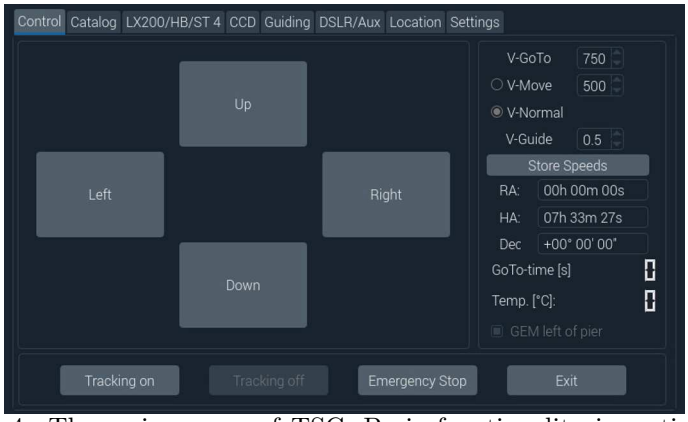


Figure 4: The main screen of TSC. Basic functionality is motion in four directions, adjustment of guiding, correction and GoTo speeds, current position as right, ascension, hour angle and declination, and buttons for starting and stopping tracking, stopping all drive motion immediately, and for terminating the program. In GoTo-mode during a slew, the estimated time of arrival is also displayed. If a meridian flip is enabled, the user can choose the side-of-pier in an initial phase.

Fig. 4 shows the main screen of TSC. It features virtual latching handbox switches for motion in the main axes. The basic speed is computed as the sidereal speed, which can also be switched to lunar and solar rates in a different dialogue. For faster motion, a radiobutton allows for switching to a faster speed (**V-Move**), which can be freely adjusted in multiples of the basic speed (**V-Normal**). The maximum speed is **V-GoTo**, which can be adjusted but is available only in GoTo mode, not via handbox operation. In autoguiding, a relative rate of 0.5 - 1 times the sidereal speed can be chosen in **V-Guide**. The pushbutton **Store Speeds** writes these settings to the preferences file and the controller. **Store Speeds** *needs to be clicked for all changes to take effect.*

The buttons on the lower part of the dialog allow for manually starting and stopping compensation motion in right ascension, emergency stopping and for exiting TSC. If meridian flips are enabled, the user can also choose whether the instrument is left or right-of-pier in an initial startup phase with the **GEM left of pier** checkbox.

On the left hand side, one finds also displays for the actual position given as right ascension (**RA**), hour angle (**HA**), and declination (**Decl**). If a slew is started in GoTo mode, either from TSC itself or via LX200 from an external program, most of the GUI functionality is blocked, and the remaining time until the end of the slew is displayed in seconds (**GoTo-time** [s]).

3.2 The catalog screen

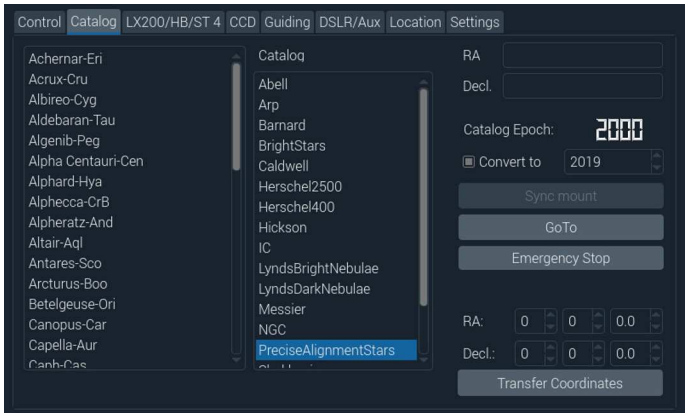


Figure 5: The screen for choosing objects, locations, and for one-star alignment and GoTo. It features an interface to manually editable catalog files. These positions, which are automatically converted to the current date, can be used for syncing the mount and for GoTo. It is also possible to enter an arbitrary location in right ascension and declination.

Fig. 5 shows the next tab, which is the **Catalog** screen. In a dedicated folder 'Catalogs' to be placed within the working directory of TSC, a set of .CSV-style files with the ending `.tsc` can be found. Information on how to edit these files and how to make your own catalog files is found in Sect. 6.2. The coordinates of the chosen object are given on the right hand side of the dialogue.

The catalog epoch is also stored in the .tsc-catalog files. The current year is read from the realtime clock of TSC, and if the checkbox **Convert to** is checked, the current position is computed. Upon pressing the **Sync mount button**, the mount is synced and tracking immediately begins. *If you are using an external program for controlling the mount via LX200, computation of the proper coordinate for the given year might be carried out there. In such a case, the* **Convert to** *button should be disabled.*

Two more pushbuttons **GoTo** and **Emergency Stop** exist; these trigger a GoTo motion to a chosen coordinate or cause a total stop of all drives. Denote

that these buttons also stay active when the remainder of the GUI is disabled, for instance during a slew to a given object. And finally, one can also manually enter right ascension and declination, which can be conveyed to TSC via the **Transfer Coordinates** button.

A few words should be told about the internal mechanics of the GoTo process; TSC uses kinematic parameters to control the drives. That is, acceleration and final speeds are given rather than stepping rates. In GoTo-mode, TSC does a few special things:

- It estimates the duration of the slew and corrects the distance in right ascension by that value. While this travel is carried out, a precise timer starts and measures the time that was really needed. This is corrected after the slew if a difference greater than a few milliseconds is encountered. Therefore it might happen that, after a slew, short action of the right ascension drive at a slower rate might follow. This is normal.
- During GoTo, most of TSCs functionality is disabled aside from the emergency stop. Denote that TSC is a multitasking program - in order to avoid multiple commands during GoTo action, this is necessary.
- The remaining time until the end of the slew is given in seconds on the main screen (Fig. 4).
- If the meridian flip is enabled in the **Settings** dialogue, TSC carries out flips for both the southern and northern meridian in order to avoid collisions with the pier during GoTo manueuvres.

3.3 The LX200/HB/ST4 dialogue

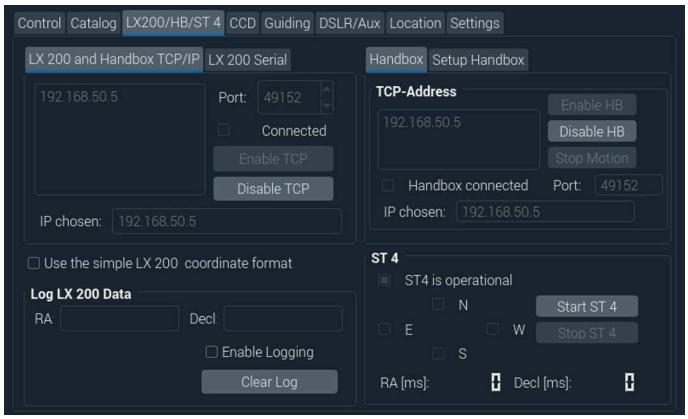


Figure 6: The configuration dialogue of TSC; LX200 via USB or WLAN and Ethernet, the connection to the custom TCP/IP handbox and for ST4 is configured here. Denote that in order to use ST4, the **Start ST4** button has to be pressed.

TSC provides several standard interfaces commonly used in amateur astronomy. These are:

- LX200 for syncing and positioning the mount via planetarium programs. These commands can be either be issued via USB as TSC features an internal USB-to-RS232 converter, via Ethernet and via WLAN. Denote that the Raspberry opens an WLAN access point **TSCHotspot** with the password **TSCRaspi** of its own if no router is within reach, therefore this functionality is also available in remote places without WLAN coverage. Sect. 9.3 gives more details on this functionality.
- TCP/IP for connecting a custom wireless handbox described in more detail in Sect. 12.
- ST4, a protocol for sending short correction commands from a dedicated controller such as the Shoestring GPUSB controller[5] or the Lacerta MGEN autoguider[6].

LX200, introduced by the Meade Corporation, is a standard protocol for controlling telescopes. The most important commands of LX200 are implemented in TSC. However, the term 'standard' is probably too big a word for the implementations available. Programs that work are Cartes du Ciel/SkyChart, Stellarium, KStars, various programs supporting ASCOM and SkySafari on Android and iOS tablets. This is discussed to some further extent in Sect. 5.3. However, for TCP/IP (that is Ethernet or WLAN) operation of LX200, Fig. 6 shows the most important dialogue in the left upper corner. TSC scans for available TCP/IP ports and displays these in the dialogue. Choose one of the addresses (`192.168.50.5` is to be used if TSC operates in independent hotspot mode spawning its own network access point) and click **Enable TCP**; using the chosen address and the preconfigured port (usually `49152`), one can now connect an external program to TSC. Details on this process are found in Sect. 5 on using planetarium programs. This setting is stored an can be accessed directly in the next startup.

If one is interested, it is possible to view the incoming and outgoing LX200 commands by checking the **Enable logging** checkbox. LX200 features also a second 'short' coordinate format. If necessary, this can be enabled in the **Use the simple LX200 coordinate format** checkbox. The coordinates received are always displayed in the **RA** and **Decl** fields.

For using LX200 in the standard serial mode, connect a micro-USB cable to the USB-to-RS232 converter mounted on the HAT of TSC (more on this converter in Sect. 10). A second tab behind the **LX200 TCP/IP** tab named **LX200 Serial** is shown. After connecting your computer to the USB-converter, a COM port should become visible in your planetarium program. Aside from that, you have to click the **Activate LX200 via Serial Interface** button shown in Fig. 7. After pushing this button, LX200 is available via USB. This setup can also be stored for future use without this configuration step.

Support for a wireless handbox is provided with TSC. It utilizes a small OLED display using the internal WLAN hotspot of TSC. It is configured using the

[5] `www.store.shoestringastronomy.com/products.htm.`
[6] `http://teleskop-austria.at/MGEN_Lacerta-Autoguider-Standalone--Remote`

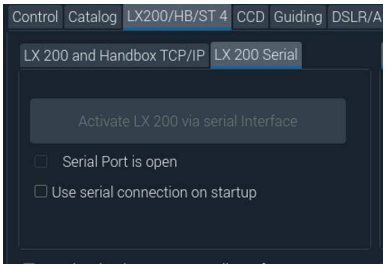


Figure 7: The dialogue to start LX200 interaction via the USB-to-RS232 converter of TSC.

TCP/IP Handbox widgets.

If you are running TSC in WLAN-standalone mode, the TCP/IP toolbox automatically connects to the WLAN-hotspot *TSCHotspot* generated by TSC. Select the IP-address `192.168.50.5`, and click the **Enable WLAN HBox** button. The handbox should connect (and this is also indicated by the checkbox **Handbox connected**). **Disable WLAN HBox** disconnects the handbox, and **Stop HBox-motion** is an emergency shutdown if motion was triggered by the handbox but connection was lost, for instance due to a fading battery. Configuration of th handbox is also planned here but not yet available.

The most simple interface is the ST4 interface. ST4 commands are received by a dedicated Arduino Mini Pro microcontroller on the HAT (see also Sect. 10.6) and conveyed to TSC using SPI-channel 0 of the Raspberry Pi. If the microcontroller is configured properly, the checkbox **ST4 is operational** is checked and ST4 can be started. Denote that most of the GUI of TSC is disabled while TSC is operating in ST4-mode. The lenght of the single correction pulses in milliseconds is dispalyed in the **RA Corr. [ms]** and **Decl Corr. [ms]** fields. St 4 needs to be started manually using the **Start ST4** button, and also has to be stopped to resume normal function.

3.4 INDI server configuration

TSC features it own autoguider functionality, which is admittedly still subject to further testing. A result of three 600 second exposures with 1300 mm focal length is shown in Fig. 15. However, TSC uses INDI to operate various guiding cameras. The cameras that are proven to work are the ZWO ASI 120 MM[7], the older QHY 5, the ToUpTek - family of guiding cameras and the standard video interface of Linux, V4L2. With V4L2, one can connect standard framegrabbers or webcams and test the program functionality. However, this needs to be tested individually as some devices were found to be instable when using V4L2.

Here, a supported camera can be chosen via the radiobuttons in the left top part of the dialogue. An INDI-server is started and runs on the localhost (thus the TCP/IP address `127.0.0.1`) using its standard port `7624`. Once the INDI-server is started, its messages are given in the log window in the lower part of the

[7] astronomy-imaging-camera.com/products/usb-2-0/asi120mm/

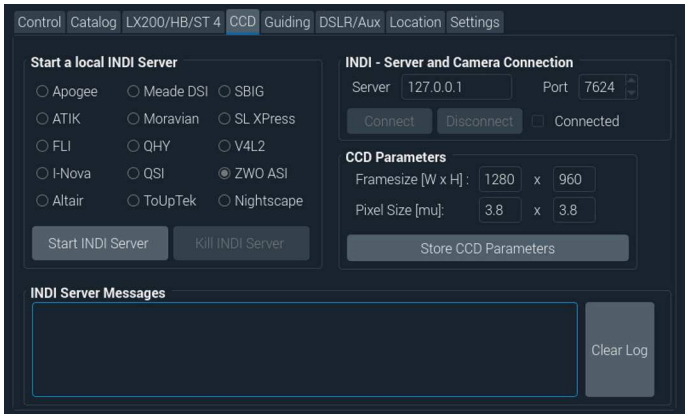


Figure 8: Dialogue for starting an INDI server and for connecting a guidecamera.

dialogue. By pressing the **Connect** button, a camera is now activated. After disconnecting the camera by pushing the **Disconnect** button, the INDI-server can be stopped by pressing the **Kill INDI Server** and a new camera can be selected. A connected camera is necessary to start exposure in the following **Guiding** dialogue (Sect. 3.5).

If your camera driver does not supply the pixel dimensions of your camera automatically, these can be entered in the **CCD Parameters** section; do not forget to store these by pressing the **Store CCD Parameters** button.

So far, only the ZWO, QHY, ToUpTek and V4L2 drivers were successfully tested as these are the only devices available to the authors. Be aware that V4L2 is a genuine interface for video devices under Linux. It is not given that every device works with this driver.

3.4.1 Acessing your guiding camera via WLAN

One of the strengths of TSC is its networking capabilities, which are mainly governed by the use of the Raspberry Pi. If you want to use an external guider and you have a guiding software that can connect via INDI, you can utilize the INDIserver started by TSC make a connection to the guider camera without having to plug the camera into your second computer. In other words – the USB cable of the camera goes into TSC, and if you are using the ST4 interface of the camera, this cable should also go into TSC. Start the INDIserver as described above, and configure INDI in such a fashion that it connects to to this server. Fig. 9 illustrates this on KStars. The remote computer runs a local INDIserver for LX200 commands and connects to the INDIserver of TSC to acquire camera data. After connecting, it is possible to control TSC via LX200 and camera images can be acquired via WLAN, rendering a wired connection to the camera unnecessary. Fig. 10 shows the INDI control panel after setting up this connection.

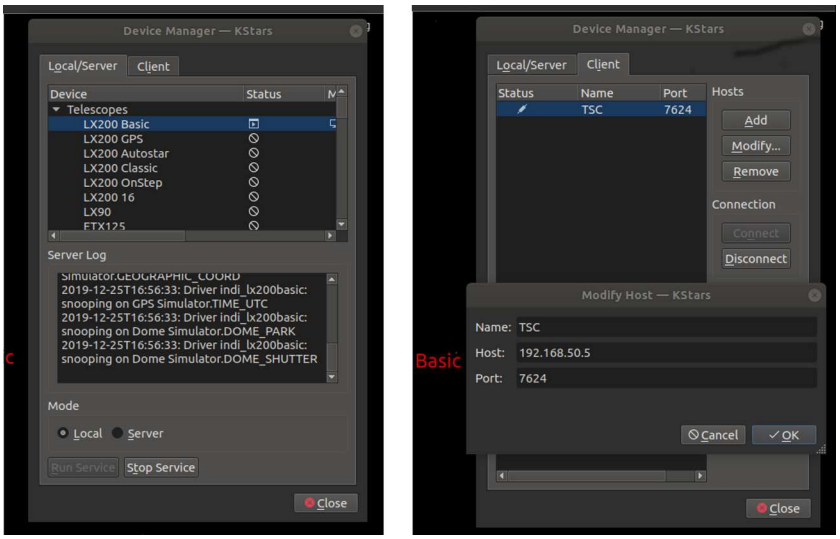


Figure 9: Configuration of INDI for using a local LX200 server and a remote INDIserver on the TSC Raspberry. The example here shows KStars. First, a local LX200 Basic INDI server is started on the remote computer (left hand side). Second, the network address of TSC and the port where the INDIserver running on the Raspberry Pi of TSC is entered (right hand side). After connecting to this server, it is both possible to control TSC from Kstars and to retrieve camera data from the Raspberry running TSC.

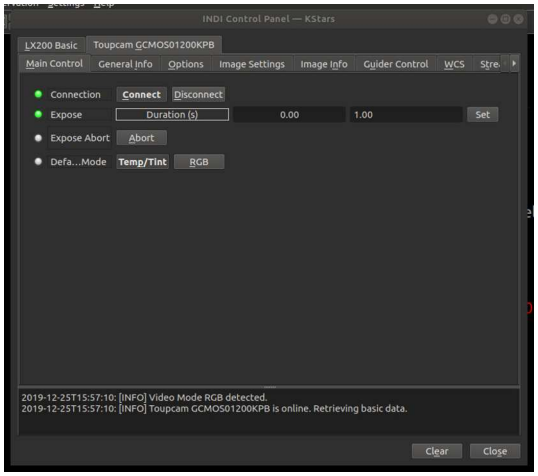


Figure 10: INDI control panel after connecting to the local LX200 server and the remote camera server of TSC. Images can be acessed on the remote computer just as if the camera would be connected via USB.

3.5 Guiding camera operation

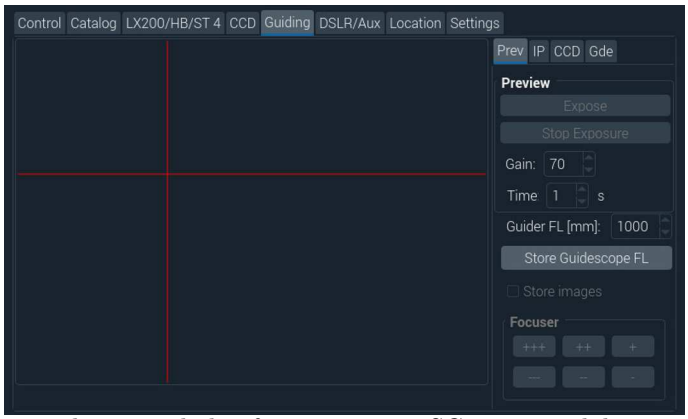


Figure 11: The main dialog for operating TSC as a standalone autoguider. If a camera is connected via INDI, exposure can be started; camera gain is adjusted if this feature is available for your camera, and the exposure time is also set. By clicking into the image, a crosshair can be positioned on the guide star. In addition, the focal length of the guidescope can be entered and stored. If the guiding images are to be stored as JPG-images, the checkbox **Store images** is to be activated – be aware that this can produce a lot of data. The buttons in the **Focuser** group allow for basic operation of an optional focuser drive connected to the guidescope.

Fig. 11 shows the main catalogue for selecting a guide star. If INDI is activated and a camera is connected, one can start exposure by pressing **Expose** pushbutton. **Stop Exposure** stops the exposure. If camera gain is adjustable, this can be controlled in the **Gain** spinbox. Exposure time is controlled by the **Time** spinbox in seconds. For autoguider operation, it is necessary to know about the focal length of the guidescope. This can be entered in the **Guide FL [mm]** spinbox and stored by pressing the **Store Guidescope FL** button. For debugging purposes, the single images can be stored as JPG image files; these are stored with the name `GuideCameraImagexxx.jpg` in the working directory. Denote that this option can produce humongous amounts of data, therefore it should be used with care.

The group **Focuser** provides a simple interface to the optional focuser board of TSC. Travel for the focuser can be adjusted in the **Photo/AuxDrives** dialog; +++ or — provide full travel, ++ or – does $\frac{1}{5}$ of that travel, and + or - does $\frac{1}{20}$ of the pre-defined travel. This functionality is also available from the wireless handbox.

The most important user interface element ist the imaging window itself; clicking on a bright star positions a red crosshair and pre-selects a guiding star. Further guiding steps are carried out in three following tabs called

- **IP** for basic image processing of the guide star using OpenCV functionality,

- **CCD** for autoguider calibration and
- **Gde** for actual autoguiding.

3.5.1 IP – guidestar selection and basic image processing

Figure 12: The **IP** sub-dialog for image processing of a guide star selected in the **Guiding** main screen. The guide star is pre-selected by clicking on the camera image. A final confirmation is done by pressing the **Select Star** pushbutton. Brightness threshold **THold**, image contrast **Cntr** and brightness **Brt** can be adjusted, and a 3×3 median filter can be activated. The size of the search window (45×45, 90×90 or 180×180 pixels can be adjusted by choosing the radiobutton in the **FOV** buttongroup. The basic size of the search window is 90×90 pixels. Once all parameters are set, this is to be confirmed by pressing the **Confirm** button.

Fig. 12 shows the **IP** subdialog of the **Guiding**-tab. The guidestar pre-selected by clicking into the main window of the camera image is confirmed here by pressing the **Select Star** button. The star image is shown in a search window, whose basic size is 90×90 pixels. This can be reduced to 45×45 pixels by pressing the ***0.5** radiobutton, or enlarged to 180×180 pixels with the ***2** radiobutton in the **FOV** buttongroup.

Basic image processing includes selecting a brightness threshold that segments the selected star chosen with the **Select Star** button – this is the **THold** slider.

Basic adjustment for contrast (**Cntr**) and brightness (**Brt**) of the image can also be defined. If the settings are chosen in such a way that pixels are saturated, this is indicated with the **Saturated Pixels** checkbox. A 3×3 median filter removing salt-and-pepper noise (that is hot or cold pixels on the guiding camera) and a 5x5 lowpass filter can be activated by checking the **3x3 Median Filter** and **5x5 Low pass** checkboxes. Denote that guiding takes place with sub-pixel accuracy. The actual location of the guide star is computed as the weighted centroid of the segmented star image, where variations in brightness are also taken into account. Once all parameters are set, the procedure is concluded by pressing the **Confirm** button.

3.5.2 CCD – autoguider calibration

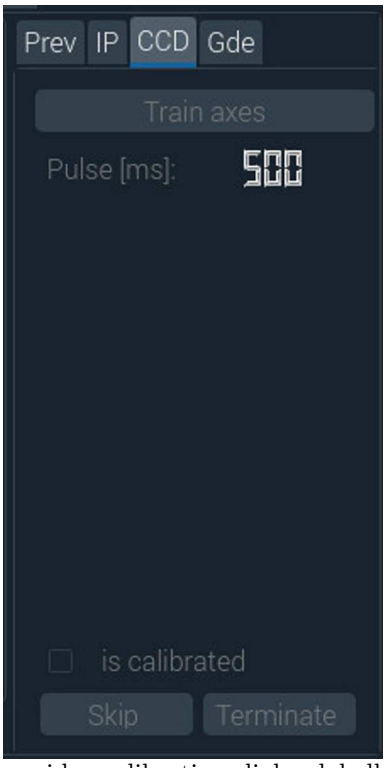


Figure 13: The autoguider calibration dialog labelled **CCD**. The **Train the axes** button triggers calibration of the mount, which runs automatically and is logged in the text field. This step can be skipped for testing purposes, and the procedure can be terminated prematurely.

Fig. 13 shows the **CCD** calibration dialogue. The calibration itself is automatic, using the parameters for pixelsize of the guiding camera and for the guide scopes focal length. It is triggered by the **Train axes** button. The progress is displayed in the text field. When calibration is finished, this is also displayed in the text field. For testing purposes, this step can also be skipped by pressing the **Skip Calibration**; a few standard calibration parameters are used in this case.

The calibration procedure takes place by doing four consecutive motions in right ascension and declination, which results in a travel of $\frac{''}{\text{ms}}$ and a relative angle of camera axes and mount axes. Backlash in declination is also determined. The **Terminate** button stops the calibration procedure at the next possible moment (this might take a few a seconds).

3.5.3 Gde – guiding using TSC

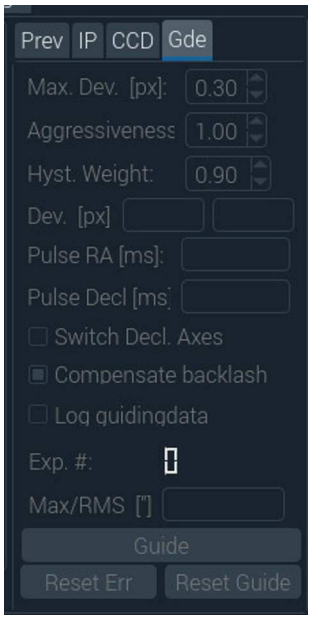


Figure 14: The dialog that starts the actual guiding procedure. A maximum permissible error in guide camera pixels can be set. By pressing the **Guide** button, autoguiding is started. Current guiding errors, correction move durations and the maximum error during guiding are displayed. If the declination axis operates in the wrong direction due to mirroring of the guiding camera image (due to a zenith mirror, for instance), this motion can be mirrored using the **Switch Decl. Axes** checkbox. Backlash compensation for declination can be deactivated using the **Compensate backlash** radiobutton. Furthermore, every inversion of the declination axis is indicated. A verbose log of the guiding process can be stored when the **Log guidingdata** checkbox is activated.

Fig. 14 shows the actual dialog for guiding. Maximum acceptable error can be adjusted in the spinboxes **Max. Dev. RA [px]** and **Max. Dev. Decl [px]** for guidecamera pixels. The aggressiveness – that is the percentage of corrective motion effectively carried out – can be adjusted and is set to 90 % by default. The actual deciation is displayed in the **Dev. RA [px]** and **Dev. Decl. [px]** text fields. The duration of the correction pulse in milliseconds is also displayed.

If the declination direction of motion is mirrored, for instance due to the used of a zenith prism, this can be compensated for by checking the **Switch Decl. Axes** checkbox. Compensation for declination backlash can be activated by

pressing the **Compensate backlash** checkbox, which is active by default. If an inversion of the declination axis during guiding takes place, this is indicated in the non-clickable checkbox **Declination was inverted**. A verbose guiding log can be stored if the **Log guidingdata** checkbox is activated. A detailed description of this log file is given in Sect. 6.3.

The maximum guiding and RMS error in arcseconds (") are also displayed in the field **Max/RMS ["]**. This error can be reset using the **Reset Error** button. Guiding is started by pressing the **Guide** button, which is also used for stopping the process once guiding is in progress. Denote that most of the GUI is deactivated when guiding.

Figure 15: Autoguiding using TSC shows a photographic first attempt on the Leo Triplet. The guidescope was a 4" f/10 refractor, the guiding camera was a ZWO ASI 120MM. 40 300" exposures with ASA 800 using a Canon EOS 1100D and a 13" f/4 Houghton were taken. Image processing with PixInsight.

3.5.4 The math of guiding

In general, TSC does a few things to optimize guiding:

- The image contrast and brightness of the camera is optimized. *Pixel saturation*, that is pixels that are a msximum brightness of the data range, are to be avoided for reasons explained in the following sections.
- A 3×3 median filter eliminates hot and dead pixels. Median filters are non-linear filters; eight pixels surrounding each pixel and the pixel itself are sorted in ascending fashion in dependence of their brightness. The central one – the fifth pixel in that sorted list – determines the brightness of the central pixel in the kernel. Outliers (that is hot or cold pixels) are

eliminated by this method.

- A 5×5 lowpass filter blurs the image slightly.
- The weighted centroid – that is the center of mass of non-zero pixels weighted by their brightness – within the search window is computed. This is assumed to be the position of the guide star.

Deviations form the guide star positions are measured and corrected; the correction has three main parameters.

- **Max. Dev. [px]:** The maximum deviation from the guide star position where no correction takes place.
- **Aggressiveness:** A factor increase or decrease the amount of correction. Ideally, this factor should be 1. If you see that the guide star is lagging behind after correction, you may increase this factor, then TSC does a little bit longer a correction thatn actually computed. An aggressiveness factor below 1 makes the correction moves shorter than actually computed.
- **Hyst. Weight:** The hysteresis of the correction motion. If set to 1, then only the last correction move is carried out as computed. If set to a value below 1, then the move is actually calculated as a weighted average of the last three moves following the formula

$$w_{i-2} = w_{i-1} = \frac{1-h}{2} \tag{1}$$

where w_{i-2} is the weight for the second to last correction move, w_{i-1} is the weight given to the last correction move, and h is the hysteresis weight. When applying this, the last two correction moves have an influence on the last move. The act as damping factors to avoid too sudden a movement, for instance due to air turbulence.

3.6 DSLR operation and focuser motor setup

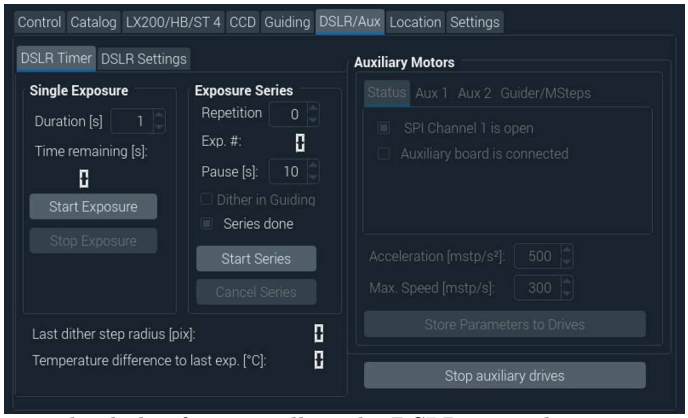


Figure 16: The dialog for controlling the DSLR in single exposures and exposure series, and for setting up the optional focuser drives.

The TSC hardware has a standard 2.5 mm jack connector for controlling the exposure of a digital single lens reflecting (DSLR) camera. This connector is isolated from the TSC HAT using a ILD 74 optocoupler and was tested with various Canon EOS cameras. Other DSLRs may be feasible but were not tested yet. In the left **DSLR Timer** group, one can set the duration of a single exposure in seconds with the **Duration** spinbox. A single exposure can be triggered with the **Start Exposure** button. The **Stop Exposure** button terminates exposure prematurely. The remaining exposure time in seconds is displayed.

In the right hand part of the **DSLR Timer** group, a series of exposures can be programmed. The duration of the exposures is given in the **Duration** spinbox, and the **Repetitions** spinbox gives the numer of single exposures. The topical number of exposures is displayed in the **Exp. : #** display. A pause of 5 – 30 seconds between single exposures can be set in the **Pause [s]** spinbox. **Start Series** triggers the series, and **Stop Series** terminates it prematurely. The checkbox **Dither in guiding** triggers a random dithering step during exposure; the settings for dithering are defined in the **Settings** dialog (see also Sect. 3.7). The radius of the last ditering step is displayed in the **Last dither step radius [pix]** display. If a temperature sensor is attached. The difference in °C from the beginning of the last exposure is displayed in the **Temperature difference to last exp.** [°C] display. Once a series is done, the **Series done** checkbox is set.

The tab **DSLR Settings** allows to set basic parameters of the DSLR such as pixel size, folcal length of the main telescope, and a minimum and maximum range for dithering in between exposures.

The group **Auxiliary Drives** is only activated if the optional focus motor board is connected to TSC (see also Sect. 13). If the SPI communication channel to the board is open and the board is present (that is, the microcontroller of the auxiliary board responds), the checkboxes **SPI Channel 1 is open** and **Auxiliary board is connected** are checked and the whole group is enabled for configuration of the focuser board.

Figure 17:

The focuser board can control two small stepper motors. In the **Aux 1** and **Aux**

Figure 18: One of the two tabs in the **Auxiliary drives** group that allows for configuring the additional steppers for the focuser board. It is possible to give a name to the drive, and to define the basic travel in the **Steps** field. The full travel, $\frac{1}{5}$ or $\frac{1}{20}$ can be tested in both directions. As long as the drive is active, this is indicated by the **Drive active** checkbox.

2 tabs it is possible to define a name and a pre-defined travel in microsteps for these two steppers using the **Steps** spinboxes. Only one drive can be activated at a time, which is indicated by the **Drive active** checkboxes. The drives can be tested by the $+$, $\frac{1}{5}+$ and $\frac{1}{20}+$ or the respective **-** buttons.

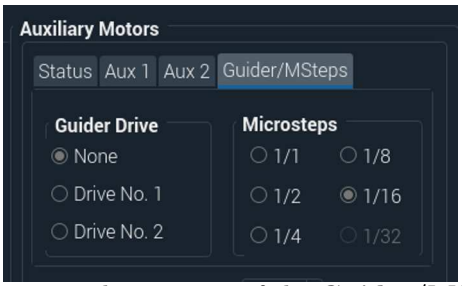


Figure 19: The contents of the **Guider/MSteps** tab.

In the **Guider/MSteps** tab, the **Guider Driver** radiobuttons allow to define which motor is attached to the focuser drive of the guidescope. As the focuser steppers are controlled via a small Arduino Mini Pro microcontroller and cost-effective Polulu A4988 or DRV8825 boards, it is possible to set the microstepping ratio either to a minimum of $\frac{1}{16}$ or $\frac{1}{32}$ using the radiobuttons for the microstepping ratio.

Settings that are applied to both drives like the acceleration in $\frac{\text{microsteps}}{\text{second}^2}$ and the maximum speed in $\frac{\text{micosteps}}{\text{second}}$ can be adjusted in the lower part of the **Auxiliary Drives** group wich can be seen in Fig. 16. An emergency stop for the steppers is also implemented, the **Stop auxiliary drives** button.

3.7 The Settings dialog

Fig. 20 shows the dialog for general settings of TSC. Of greatest importance is the group **RA Gears** and **Declination Gears**. Here, the gear ratio for steps

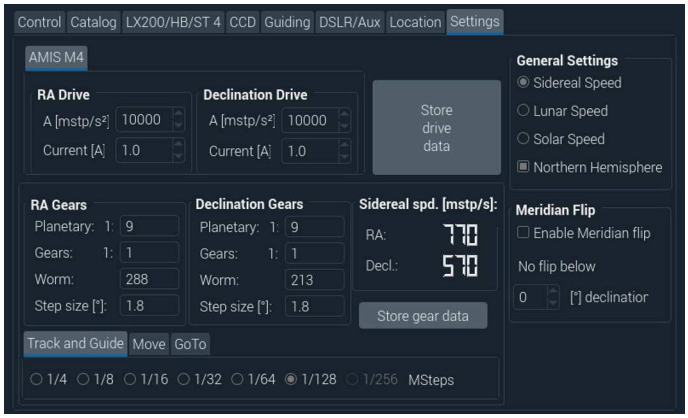


Figure 20: The dialog for setting various properties of TSC. The most important part here is the **RA Gears** and **Declination Gears** group. Here, the planetary gear ratio and the worm wheel size for the main axes are defined. If an additional gear (like a pulley) is added between the motor or the planetary gear, this ratio is also to be defined in the **Gears** field. Maximum acceleration and coil current are also set, together with the tracking rate (sidereal, lunar or solar) and inversion of the tracking direction for the southern hemisphere.

versus the Earths rotation is set. First, the ratio of an optional planetary gear is set in the text field **Planetary 1:**; if you do not have a planetary gear, set this value to 1. An intermediate gear between the worm and the effective axle of the stepper is set in the **Gears 1:** field. This can also be a pulley – if your ratio is scaling up, e. g. by driving a small pulley wheel with a bigger pulley wheel, this value is smaller than 1. If no intermediate gear is used, the ratio should be set to 1. The number of teeth on your worm wheels is set in the **Worm** field. The number of steps per full rotation is set in the **Step Size** [°] field. For a common stepper with 200 steps per rotation (=360°), this is 1.8°. Denote that changing the parameters is not feasible when the drive is running, for instance during tracking. Storage of parameters is triggered when the **Store gear data** button is pushed – *no changes take effect before the* **Store gear data** *button is clicked.* SOmething that is also set here is the microstepping ratios used for different modes of operation. In **Track and Guide**, utmost smoothness is desired – it is a good idea to set a fine resolution microstepping ratio, I use **1/128** here. For gross motion – **Move** – with the handbox, higher speeds as set in the **Control** tab are desired. I use **1/16** microsteps here. And finally, for **GoTo**, I use **1/4** microstepping to achieve maximum speed. Remember that the speeds are set in the **Control** tab. *Again,* **Store gear data** *needs to be clicked for these changes to take effect.*

The kinematic parameters for the stepper motor, that is its standard speed and its acceleration, are partially set automatically. The speed is the equivalent to sidereal speed for the given gear ratios and stepper motors; it is fixed. The acceleration gives the ramp for starting up the motors. A good value for me is 5000 $\frac{\text{microsteps}}{\text{second}^2}$. The coil current can be adjusted in the software for smooth

operation – this value depends on your motor and on your mount. Start with a small value (where the stepper probably will not turn) and increase it to a value where smooth operation is observed. Try not to exceed approximately 70% of the coil current your drive is rated for. Smooth operation means that the optimum torque needed for your scope is applied. Too low a current will lead to no or irregular motion of the drives – the drive actually loose steps. Too high a current leads to overhaeting the drives in holding position and to rough and noisy operation of the drives. Optimum current is given if your scope moves smoothly and quiet, and the drives do not get too hot. These data are also stored when pushing the **Store drive data** button.

The **General Settings** group allows for switching between sideral, lunar and solar speed. These checkboxes take effect immediately. When issueing a GoTo-command, TSC switches back to sideral speed. For operation on the southern hemisphere (where RA direction is reversed), the checkbox **Northern Hemisphere** should be unchecked.

The **Meridian flip** section handles the behaviour of the mount when a meridian flip becomes necessary – this is an intrinsic property of German equatorial mounts (GEMs). It is needed to avoid collision of the rear part of the telescope with the pier, especially at higher declinations. If one has a fork mount, a yoke mount, a GEM with an angled pier or a setup where the scope cannot collide with the pier due to is contruction, you do not need to check the **Enable Meridian flip** checkbox. It is also possible to suppress the meridian flip below certain declinations with the **No flip below** spinbox. Denote that when enabling the meridian flip, the mount needs to know initially on which side of the pier it is. This is done with the **GEM left of pier** checkbox in the **Control** tab at startup. After the first meridian flip, this cannot be changed anymore.

3.7.1 Math, logic and terminology of the meridian flip

- Enable the meridian flip only if you are using a GEM.
- Some external drivers like the *Advanced LX 200 ASCOM driver* from `https://pixelstelescopes.wordpress.com/advanced-lx200/` can also do a meridian flip. If you want to use this feature of the ASCOM driver, then you have to disable the meridan flip in TSC. If you want to use the meridian flip from TSC, you have to tell dhis driver that you are using a fork mount.
- In TSC, a meridian flip is only triggered in GoTo mode; with the handbox or during guiding, no meridian flip is carried out, but it can be done manually with the handbox.
- There is areas (namely the northern circumpolar region) that are not accessible with a GEM under some circumstances.

So what is a meridian flip? Basically, when receiving a GoTo command, TSC computes the shortest path from the topical positions and starts the drivers to carry out that motion. In cases where the rear end of the telescope might collide with the pier/tripod of the GEM, this mechanism is disabled:

- The right ascension drive makes a move of $\Delta_{RA} = -(180 - \delta_{RA})$ or $\Delta_{RA} = 180 + \delta_{RA}$ in dependence whether the flip goes from the left side of the pier to the right side or vice versa – also denoted as $\mathrm{MF_{CCW}}$ and $\mathrm{MF_{CW}}$ for clockwise and counterclockwise motion in Table 1. Δ_{RA} is the total travel in degrees, and δ_{RA} is the shortest travel path from the actual to the desired position.
- The declination drive makes a move over the north pole and carries out a travel of $(90 - 2 * x_{\mathrm{Decl}} - \delta_{\mathrm{Decl}}$. x_{Decl} is the topical declination value, and δ_{Decl} is the shortest path in declination to the desired position.
- When crossing the pole, the direction of the declination drive switches.

Denote that when doing a pole cross manually, the direction buttons keep their function as long as they are pressed. Upon release, they change direction. In other words – you can use the **Up** *button in the* **Control** *tab to move over the pole, and after passing the pole, it keeps up its direction and goes actually down to the north horizion. If you stop this motion, the* **Up** *button will steer your scope again back to the north pole.*

When am I left or right of pier? This terminology refers to the position the scope is looking at. If you look to the south horizon, you stand behind your scope and the scope is on the east side of the pier it is *left of pier*. If you look to the north horizon and the scope is on the west side, *it is still on the left side of the pier*. So left and right of pier always refer to the position of the scope and where it is looking at. Most people think of the meridian flip as a phenomenon that refers wehn looking south and going fro east to west of vice versa. But a flip also has to be carried out when crossing the northern meridian, hence the terminology.

When is a meridian flip carried out? Fig.21 illustrates the four cases where a flip becomes necessary. In general, the sky can be ddivided into four quadrants, whose position is fixed and defined by the hour angle (HA). So HA $\in (0...90]^{\mathrm{o}}$ defines quadrant I, HA $\in (90...180]^{\mathrm{o}}$ defines quadrant II and so on. If the scoep is *left of pier* and looking, for instance, at an object in qudarnt IV and a slew to quadrant I is issued, no meridian flip is triggered as this is considered safe. *Denote that the meridian flip is only carried out when a GoTo is issued.* If the scope is left of pier, looking at an object in quadrant I and a slew to quadrant IV is issued, a meridian flip is carried out. In Fig. 21, a green zone is also marked – here, no flip is carried out if one stays below a declination of -20^{o}. This value is set in the **No flip below** spinbox. *Take care that the* **GEM left of pier** *checkbox is set correctly at startup as it can no longer be changed after the first meridian flip.*

3.8 The Location dialog

This is probably the most simple of all setup dialogs. Here, the current date, time, Julian day and sidereal time are displayed as read from the hardware clock mounted to the TSC HAT. Denote that a Raspberry Pi does not have a clock, therefore these readings might be erratic if the HAT is not mounted and no

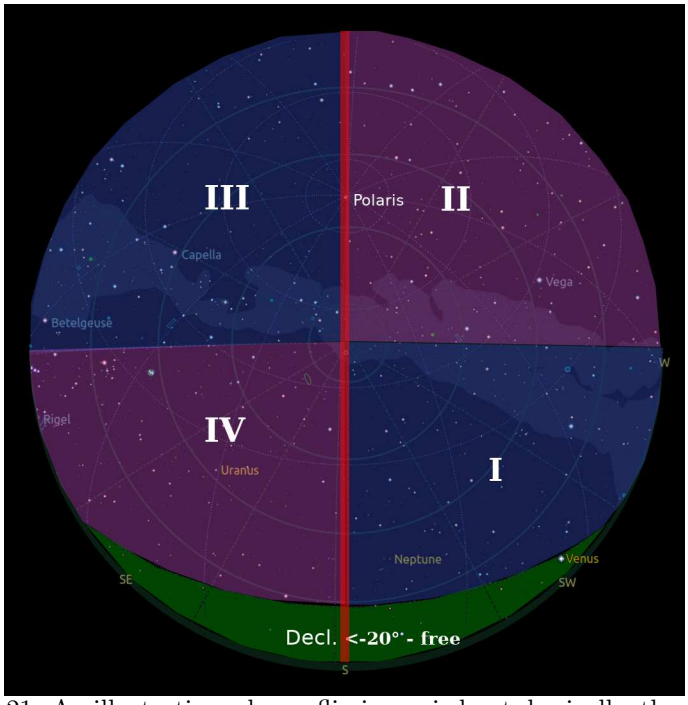


Figure 21: An illustration when a flip is carried out; basically, the sky is divided in four quadrants, labelled I to IV. The green zone indicates when a merdian flip is supressed by the value **No flip below** spinbox in the **Settings** tab. In this case, no merdian flip is carried out if the scope stays in a safe zone below -20 degrees declination. The red line denotes the meridian.

	Scope left of pier			
	TQ I	TQ II	TQ III	TQ IV
OQ I	x	x	MF_{CW}	MF_{CW}
OQ II	x	x	MF_{CW}	MF_{CW}
OQ III	MF_{CCW}	MF_{CCW}	MF_{CCW}	MF_{CCW}
OQ IV	MF_{CW}	MF_{CW}	MF_{CW}	MF_{CW}
	Scope right of pier			
OQ I	MF_{CCW}	MF_{CCW}	MF_{CCW}	MF_{CCW}
OQ II	MF_{CCW}	MF_{CCW}	MF_{CCW}	MF_{CCW}
OQ III	MF_{CW}	MF_{CW}	x	x
OQ IV	MF_{CCW}	MF_{CCW}	x	x

Table 1: Meridian flips carried out clockwise (MF_{CW}) and counterclockwise (MF_{CCW}) in dependence of the origin quadrant (OQ) and target quadrant (TQ) of the slewing motion. If a field is filled with 'x', no meridian flip is carried out.

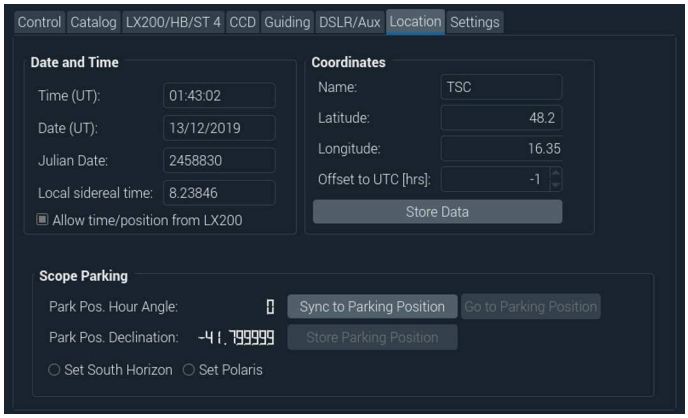


Figure 22: The location dialog. Current readings from TSCs internal hardware clock are displayed, and the observation site coordinates are defined. It is possible to set the position and time using LX200, for instance by SkySafari. If the mount is synchronized, it is also possible to define a parking position and to make the telescope return to that parking position.

network access is available.

In addition, one can store the location name, its latitude and longitude, and the offset to universal time. These data are necessary for correct computation of the sidereal time; pushing the **Store Data** button stores these data to the preferences file.

If the mount was synced, it is also possible to define a parking position. The current position is taken as the parking position when pressing **Store Parking Position**; when pressing **Go to Parking Position**, the scope returns to this position and stops tracking. Upon power up, it is possible to sync the mount again to the parking position using the **Sync to Parking Position** button. TSC computes the topical right ascension from the clock and the the stored position and if you scope was not moved in between, the mount should be synced again.

The **Allow time/position from LX200** checkbox allows external programs to set the location and internal clock if checked. This is particularily usful in combination with SkySafari, which has access to the GPS-receiver in a smartphone or tablet.

4 Remote access via VNC

As TSC spawns its own WLAN in absence of an accessible SSID via the independent access point `TSCHotspot` – more on this in Sect. 9.3 – it is also possible to access its functionality via a smart phone, tablet or personal computer. This is achieved by the VNC-Server running on Raspian. VNC is a widespread software for remote computer access. In order to access TSC, one needs to install a

VNC client from `https://www.realvnc.com/`, where a basic client for a variety of operating systems for private use can be downloaded for free.

When the `TSCHotspot` is visible for your device and the VNC client is installed via the VNC homepage, Google Play or a similar service, the following steps have to be taken:

- Select `TSCHotspot` as your WLAN access point (see also Sect. 5 and Fig. 24).
- Start the VNC client.
- Add a connection to a computer named `pi`.
- Select `192.168.50.5` if that is the IP-address of your Raspberry Pi (it should).
- Username is `pi`, password is `TSCRaspi`.

After this, one can manipulate the GUI of TSC in a similar manner to direct manipulation on the touchscreen; Fig. 23 shows this on a ASUS ZenPad running Android.

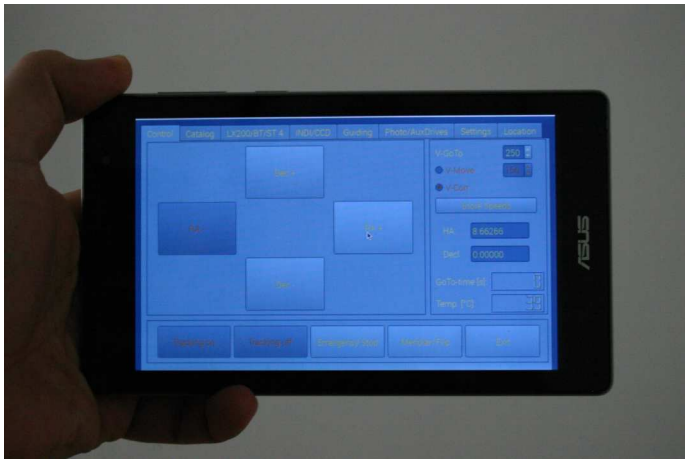


Figure 23: Remote control of TSC using VNC and an Android-based tablet.

5 Connecting to external planetarium programs

The internal catalogs are quite extensive and can be extended easily; but planetarium programs like SkySafari Pro[8], Sky Chart/Cartes du Ciel[9] or KStars[10] running on tablets, cell phones or laptops are a very nice add-on for exploring the night sky. The communication protocol supported by TSC is the well-known LX200 protocol developed by Meade[11], which was introduced more than 25

[8] `https://skysafariastronomy.com/`
[9] `https://www.ap-i.net/skychart/en/start`
[10] `https://edu.kde.org/kstars/`
[11] `https://www.meade.com/support/LX200CommandSet.pdf`

years ago. Originally, it relied on serial communication whereas most computers nowadays do no longer feature a COM-Port; the TSC HAT therefore features a USB/Serial converter, the Adafruit USBFriend (see also Fig. 42) based on a CP 2104 chip. It works fine with MacOS, Windows and Linux and usually shows up as `/dev/ttyUSB0` or a COM-port without further ado. Therefore, LX200 is supported via USB when connecting a MicroUSB cable to a PC. This connection has to be activated in the **LX200/HB/ST4** tab of TSC; details are found in Sect. 3.3 and Fig. 7. The usual communication parameters for the serial board are 9600 baud, 8 data bits, no parity, 1 stop bit.

LX200 sends command in text form, commands and data exchanged usually have the structure `:...#`. Once a LX200 connection is established, one can check the checkbox **Enable logging** in the **LX200/HB/ST** tab of TSC and watch the stream of data.

In addition, TSC also supports LX200 operation via WLAN – remember that TSC can open its own access point, and external devices such as laptops and tablets can connect to it. The default name for this autonomous WLAN access point is `TSCHotspot`. Fig 24 shows the looks of accessing TSC.

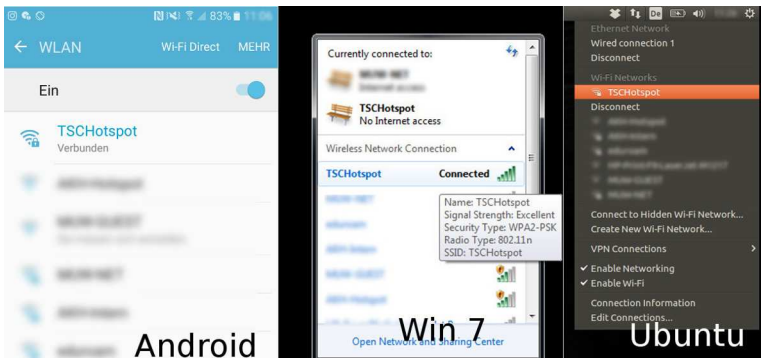


Figure 24: Various views of the autonomous `TSCHotspot` WLAN access point spawn by TSC.

Once your external device is connected to the `TSCHotspot`, it is possible to activate LX200 communication via WLAN. TSC offers the available IP-addresses in the sub-tab **LX 200 TCP/IP** of the **LX200/HB/ST4** tab. The WLAN address provided by TSC usually is `192.168.50.5`, whereas other addresses are provided by the Ethernet adapter. Click the WLAN address and press the **Enable TCP** button. TSC now waits for incoming requests.

5.1 Sky Safari Pro using WLAN

The Pro-Version of SkySafari (available for approximately 15 €) allows for connecting to TSC via WLAN; the communication protocol is *Meade LX200 Classic*, the IP address has to be the one provided by TSC, and the Port is `49152` by default. Fig. 25 shows a few screenshots. Basic operations include aignment, GoTo and handbox-style motion control with different speeds using the

up/down and left/right buttons of sky safari. The slider on the bottom allows for speed control. Controlling your telescope with a tablet is really cool, and SkySafari is an excellent program – the money for the Pro-version is definitely well spent.

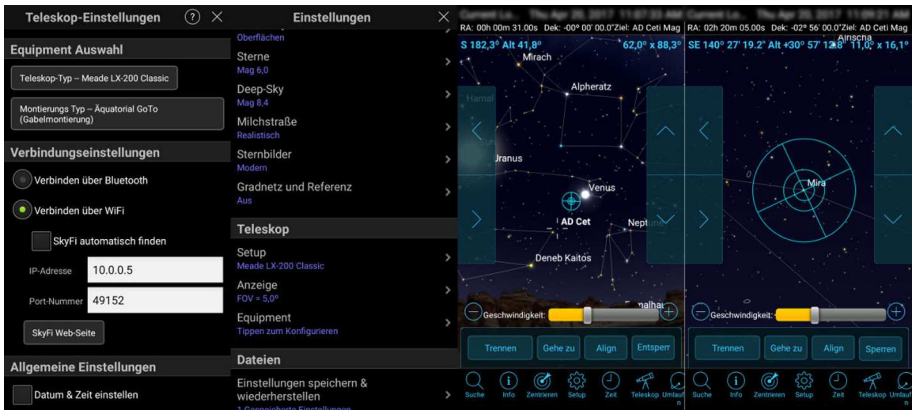


Figure 25: Connecting SkySafari Pro via WLAN to TSC allows for controlling the tescope. The actual position of the telescope is also updated on SkySafari.

5.2 SkyChart/Cartes du Ciel, KStars and Stellarium

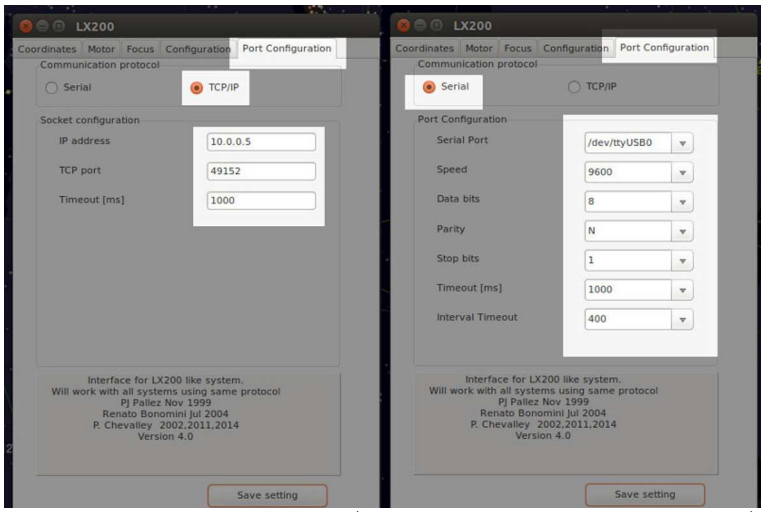


Figure 26: Dialogues in SkyChart/Cartes duCiel for setting up TCP/IP or serial communication for LX200 control.

A few excellent freely available programs also exist; my favourites are SkyChart and KStars. SkyChart/Cartes du Ciel (or CdC) can connect both via WLAN and USB and Fig. 26 shows the dialogues for configuring those interfaces; the communication protocol is *LX 200* and the *Display Precision* has to be set to **high** (see also Fig. 27). After this, CdC controls your telescope, either

via a virtual handbox or via the *Slew* command (see also Fig. 28). CdC was tested with Windows 7, MacOS and Linux and works perfectly fine with these parameters

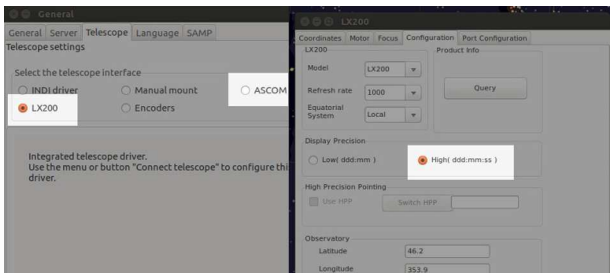


Figure 27: The further configuration steps necessary for connecting TSC to SkyChart – take care that the *Display Precision* is set to **high**.

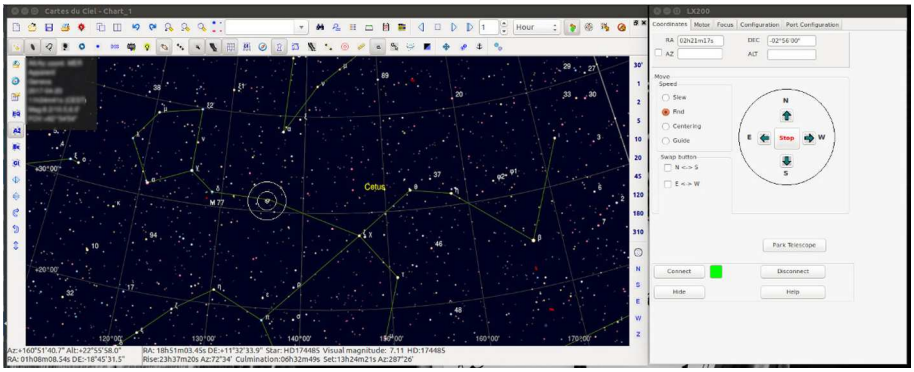


Figure 28: Connection of TSC and CdC. The telescope can be controlled via a virtual handbox or the *Slew* command of CdC; the actual position of the telescope is displayed on the screen with a crosshair.

Another excellent, freely available Program is KStars. Setting KStars up to work with TSC via the serial connection is easy. The protocol is *Meade LX 200 Classic*, and the serial port usually is `/dev/ttyUSB0` under Linux. Fig. 29 shows two screenshots of this process. After connection, KStars allows for control of telescope position using TSC, just like the other programs. Fig. 30 shows this.

Stellarium is also a beautiful program. Denote that older versions of Stellarium do not support syncing the telescope, but once this is done in TSC, Stellarium works fine via the serial communication protocol as shown in Fig. 31. It also supports INDI via WLAN but the *LX200 Basic* server has to be started manually apparently. I do not see a possibility to dispatch Sync or Slew commands with Stellarium.

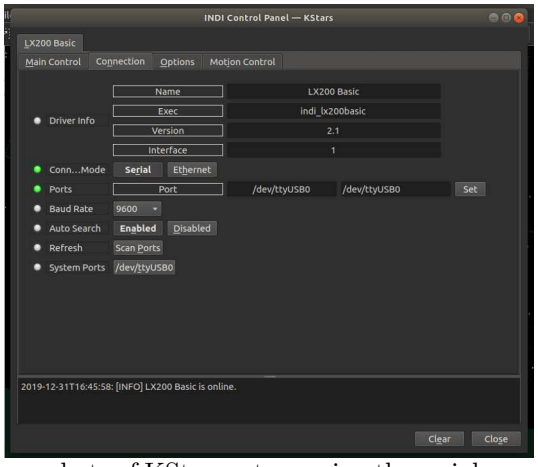


Figure 29: Screenshots of KStars set up using the serial connection under Linux.

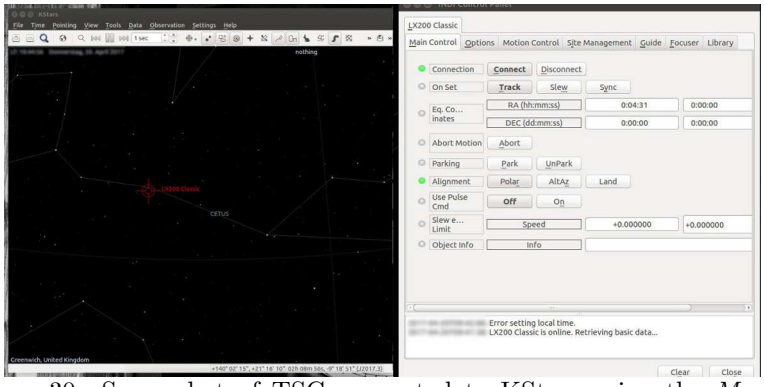


Figure 30: Screenshot of TSC connected to KStars using the *Meade LX200 Classic* protocol.

Figure 31: Configuration and display of telescope positions using serial communications in Stellarium.

5.3 ASCOM and Windows

The *Advanced LX200* driver[12] was tested with the following planetarium program supporting ASCOM 6.3 under Windows 7 Professional:

- SkyTechX[13]
- Computer-Aided Astronomy[14]
- CdC/Skychart
- HNSKY[15]

With all of these programs, the Advanced LX200 interface appeared to work fine over a USB connection; however, it is to be emphasized that the driver must not be started directly in the ASCOM interface but through the POTH. Fig. 32 shows the procedure;

- select the *POTH Hub* interface,
- click on *Properties*.
- Next click *Choose Scope*.
- In the following dialogue, select *Advanced LX200 Telescope* and click *Properties*.
- There, you can choose **Generic LX200** and the adequate COM-port, which should show the address of the USB/RS232 converter of the HAT.

It has to be denoted that operation via ASCOM is considerably slower compared to direct access to the serial port like in the case of CdC.

5.4 What about MacOS?

CdC also works via WLAN or USB with TSC, for other programs there is no experience.

6 Internal data formats

6.1 The preferences file

All relevant information entered in the GUI is stored in a dedicated preferences file stored in the home-directory of TSC. The name of this file to be found in the working directory is `TSC_Preferences.tsp`. It is read at program startup. It is readable as all information is stored as text, but it should not be edited. If the file is deleted, it is automatically generated anew by TSC. At the time of this writing, it contains the following elements:

- The gear ratio for planetary connected to RA-stepper.

[12] `https://pixelstelescopes.wordpress.com/advanced-lx200/`
[13] `http://skytechx.eu/`
[14] `http://www.astrosurf.com/c2a/english/`
[15] `http://www.hnsky.org/software.htm`

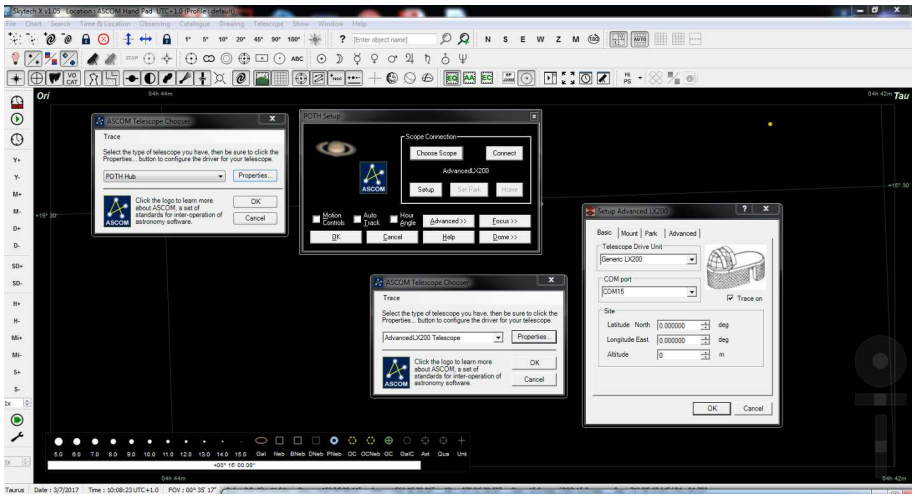


Figure 32: Setting up ASCOM 6.3 to work with the Advanced LX200 interface. The **Advanced LX200 Telescope** driver has to be selected via the **POTH** interface of ASCOM.

- The gear ratio of non planetary/non worm gear in RA.
- The number of teeth of the RA-worm wheel.
- The size of a full step for RA-Stepper in degrees.
- The gear ratio for planetary connected to declination-stepper.
- The gear ratio of non planetary/non worm gear in declination.
- The number of teeth of the declination worm wheel.
- The size of the full step for declination-stepper in degrees.
- The number of microsteps your driver does for different motion speeds.
- Acceleration in $\frac{\text{microsteps}}{\text{s}^2}$ for the RA-drive.
- Acceleration in $\frac{\text{microsteps}}{\text{s}^2}$ for declination drive.
- Maximum coil current in Ampere for the RA-drive.
- Maximum coil current in Ampere for the declination drive.
- Pixelsize in x-direction for the guiding camera. This value can either be manually edited in the GUI or it can be read directly via the INDI-driver.
- Pixelsize in y-direction for the guiding camera.
- Chip width in x-direction for the guiding camera. This value can either be manually edited in the GUI or it can be read directly via the INDI-driver.
- Chip width in y-direction for guiding camera.
- Focal length in millimeters of the guiding scope.

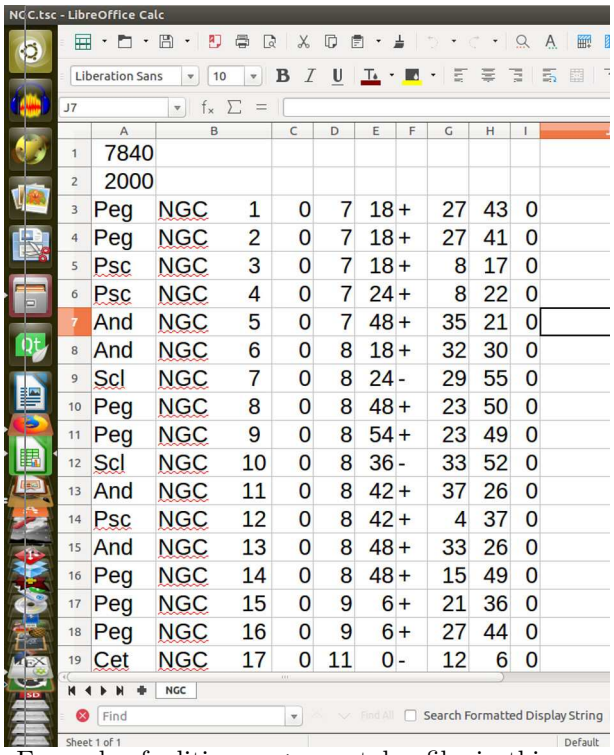
- Latitude of the observation site.
- Longitude of the observation site.
- UTC Offset of the observation site.
- Name of the observation site.
- Name of the first auxiliary/focus drive stepper.
- Name of the second auxiliary/focus drive stepper.
- Standard number of microsteps for first auxiliary/focus drive stepper.
- Standard number of microsteps for second auxiliary/focus drive stepper.
- Acceleration in $\frac{\text{microsteps}}{\text{s}^2}$ for both auxiliary/focus drive steppers.
- Speed in $\frac{\text{microsteps}}{\text{s}}$ for both auxiliary/focus drive steppers.
- Number of microsteps per full step for both auxiliary/focus drive steppers..
- The ordinary number (0, 1, or 2) of the focus driver connected to the guider scope focuser.
- Pixel diagonal size of the DSLR in μm.
- Focal length of the main telescope in mm.
- Minimum range in pixels of the DSLR for dithering of DSLR exposures.
- Maximum range for dithering of DSLR exposures.
- Speed for GoTo
- Speed for fast motion of the mount with the handbox.
- Hour angle of the parking position.
- Declination of the parking position.
- The default IP address for LX200 access via planetarium programs.
- The default IP address for the wireless handbox.
- A flag that indicates whether a meridian flip is to be carried out.
- The maximum declination for not accrying aout a meridian flip.
- A flag that indicates whether it is allowed to set location and time via LX200.

6.2 Catalog files

In the working directory of TSC, catalogs are stored in a dedicated folder `Catalogs`. It is stored separately on the github-repository. Within this directory, all catalog files ending with `.tsc` are treated as catalog files and are read by TSC during startup. Additional catalog files can easily be created with a common spreadsheet program such as Excel or Libre Office. The format is `CSV`, and the structure of entries is as follows:

- **Line 1** holds the number of objects in the catalog file; for the Messier catalog, this is for instance `110`.
- **Line 2** holds the epoch of the catalog file; most catalog files provided in the `Catalogs` directory are given for epoch 2000. Denote that TSC can convert these coordinates to the current epoch.
- The **next lines** hold the catalog entries which are given in the sequence *Constellation, Name, RA hour, RA minute, RA second, Declination sign +/-, Declination degrees, Declination arcminutes, Declination arcseconds.* If a constellation is not given, the first entry stays empty.

Fig. 33 shows a screenshot of what this looks like when editing the NGC catalog in Libre Office. One can add individual catalogs by following this convention and by putting them into the **Catalogs** folder in the working directory.



	A	B	C	D	E	F	G	H	I
1	7840								
2	2000								
3	Peg	NGC	1	0	7	18	+	27	43	0
4	Peg	NGC	2	0	7	18	+	27	41	0
5	Psc	NGC	3	0	7	18	+	8	17	0
6	Psc	NGC	4	0	7	24	+	8	22	0
7	And	NGC	5	0	7	48	+	35	21	0
8	And	NGC	6	0	8	18	+	32	30	0
9	Scl	NGC	7	0	8	24	-	29	55	0
10	Peg	NGC	8	0	8	48	+	23	50	0
11	Peg	NGC	9	0	8	54	+	23	49	0
12	Scl	NGC	10	0	8	36	-	33	52	0
13	And	NGC	11	0	8	42	+	37	26	0
14	Psc	NGC	12	0	8	42	+	4	37	0
15	And	NGC	13	0	8	48	+	33	26	0
16	Peg	NGC	14	0	8	48	+	15	49	0
17	Peg	NGC	15	0	9	6	+	21	36	0
18	Peg	NGC	16	0	9	6	+	27	44	0
19	Cet	NGC	17	0	11	0	-	12	6	0

Figure 33: Example of editing a `.tsc` catalog file, in this case the `NGC.tsc` file with *Libre Office* under Ubuntu Linux 16.04. The catalog files can be edited with any spreadsheet program as they are basically `.CSV` files. The first line holds the number of entries, the second line gives the epoch.

6.3 The guiding log

Due to space constraints, the autoguider does not offer a graphical display of guiding errors; however, maximum and RMS error are recorded. However,

the dialogue shown in Fig. 14 allows to check whether guiding data are to be recorded. These are stored in a local file `GuidingLog.tsl`, which can also be analyzed using a spreadsheet as it is stored in the .CSV-style. The file starts with a few calibration parameters (travel time per guidecam-pixel in ms for right ascension and declination, the rotation matrix for transforming camera directions into mount directions, the travel time for backlash compensation in ms and the position of the selected guiding star in camera coordinates). What follows is a record of the measured position of the guiding star (*"Measured centroid"*), the transformed measured position (*"Transformed position"*), the duration and direction of right ascension and declination correction and direction, and an indicator if declination direction was inverted and whether backlash compensation was activated.

6.4 The communication protocol for the focuser motors

The stepperboards for TSC communicate via SPI between the Raspberry and the microcontrollers running the steppers with the AccelStepper library[16]. The commands are transmitted as characters to the microcontroller, following the convention: *xyzzzzzz...* where x is a character denoting the action to be taken, y is a number between 0 and 4, where therse are literals, and *zzzzzz...* is a string of bytes that is converted to a long integer.

An example follows: "a1200" sets the acceleration (denoted by 'a') of the second drive ('1' instead of '0') to 200 microsteps per second.

Commands that apply to all drives (such as the number of microsteps, which can only be set for both drivers) are issued without the second byte. In general, parameters have to be set, then the drive is enabled, and then the drive can be started. Once a motion is finished, the drive is being automatically disabled.

The command set:

- To enable or disable drives: **exy** where x is the number of the drive ('0' or '1' for the focus-driver board) and y is a boolean - 1 means enable the drive (that is, power the coils up), and 0 means "disable".
- Setting the acceleration: **axyyyyy** where x is the number of the drive and yyyy is a long integer for the acceleration in $\frac{\text{microsteps}}{s^2}$.
- Setting microsteps: Microsteps can only be set for groups of two drives, the command is **m xxx**. xxx is either 001, 002, 004, 008, 016, 032, 064 or 128. This is the nominator of the ratio of microstepping. As an example, `m 008` sets both drivers to 1/8 microstepping.
- Setting final velocity: **vxyyyy** where x is the drive as usual and yyyy is a long integer for $\frac{\text{microsteps}}{s}$, which is the final velocity.
- Setting the number of steps: **sxyyy**. Again, x is the driver and yyy is the number of microsteps

[16] `http://www.airspayce.com/mikem/arduino/AccelStepper/`

- Checking whether a board is connected: **t**. This command responds with 'D', 'A', '0', '1' or 'B'. 'D' is short for the DRV 8825 and both drives are inactive. 'A' indicates an A4988 board with both drives inactive. If both are running, one gets a 'B', otherwise the numeral for the active drive is sent. If 'D' or 'A' is returned when 't' was sent, a board is connected.
- Stop a drive: **xy** where y is the address of the drive ('0' or '1').
- Start a drive: **oy** where y is again the address the drive. Steps, acceleration, microstepping and velocity have to be defined, and the drives have to be enabled.

6.5 The `StartTSC` script in `/home/pi`

This script

- switches to the build-directory of TSC and
- starts TSC.

It is called by the Desktop-Icon to start TSC. If you want to change you build-directory, edit this script as well. If you want TSC to start upon login, add a call to this script in
`/home/pi/.config/lxsession/LXDE/autostart`.

6.6 The hidden INDI-PID file

Another file that is generated in the working directory is the file `.INDIPID.tsl`, which is invisible. As TSC can start an INDI server of its own, it is necessary to remember the process ID iof that server if the program crashes. TSC read this file during startup and may kill a residual INDI process before starting a new one.

7 Future developments

7.1 Different driver boards and optimization for mobile use

The driver boards now operate in a two-stage approach; the AMIS 30543 is controlled via the Teensy 4.0 microcontroller, which receives its commands via USB from the Raspberry. As the AMIS utilizes the standard STP/DIR interface, other controllers are feasible in the near future.

7.2 High resolution encoders and pointing models

I initially wanted to include encoders from the very beginning, but standard rotary encoders offer too little resolution to monitor anything but the number of microsteps carried out by the stepper – which should not be an issue as long as the steppers and the current are properly chosen – and high-resolution encoders are really expensive. But in theory, there is sufficient computing power

to implement high precision encoders, pointing models and also plate solving. The implementation of a pointing model is certainly a task for the near future. Current experiments with the Raspberry 4 with 4 GB have shown that the Astrometry.net solver, for instance, can be run locally without routing the external servers.

Another possible feature is periodic error correction. With an camera interface provided by INDI, that is not a big problem. To me, the question remains whether it is worth the effort when autoguiding is already available.

Part III

Building the TSC controller.

8 Getting all the components

All software components of the TSC-controller are found in the repository `https://github.com/selste/TwoStepperControl`. This includes

- The Qt-project and C++ sourcecode for TSC.
- The electronic layouts in Fritzing format[17] for all hardware components; PCBs can, for instance, be ordered via the Fritzing software directly.
- C-programs for the Arduino IDE for the additional microcontrollers used by TSC.
- The Catalog files.
- A detailed Bill of Materials including pricing and part numbers.
- This documentation.

What is missing is an image of Raspian running TSC with all additional software requirements installed. This image is at least 8 GB large and can be requested by `wbirkfellner@gmail.com`. At least a 16 GB SD-card is necessary to run TSC.

9 Getting started

9.1 Installing a Raspian image running TSC

In order to give TSC a try, it is best to start with the basic configuration; for this purpose, one needs

- at least a Raspberry Pi 3 Model B with 1 GB of RAM. This device is for instance available from www.exp-tech.de for less than 40 €. A power supply for the Raspberry is also needed; this can be a standard microUSB 5V supply with sufficient current[18].
- a fast 16 GB SD card, which costs approximately 10 €.
- an image of Raspian Buster holding the development tools, the configuration and the TSC program from the authors.

The image of the operating system including TSC is copied on the SD card; under Linux, this is done by the `dd` command. Just insert your SD-card in a computer, **unmount it**, open a terminal, change to the directory where the SD-card image named `TSC.img` is located and type

[17] www.fritzing.org
[18] www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

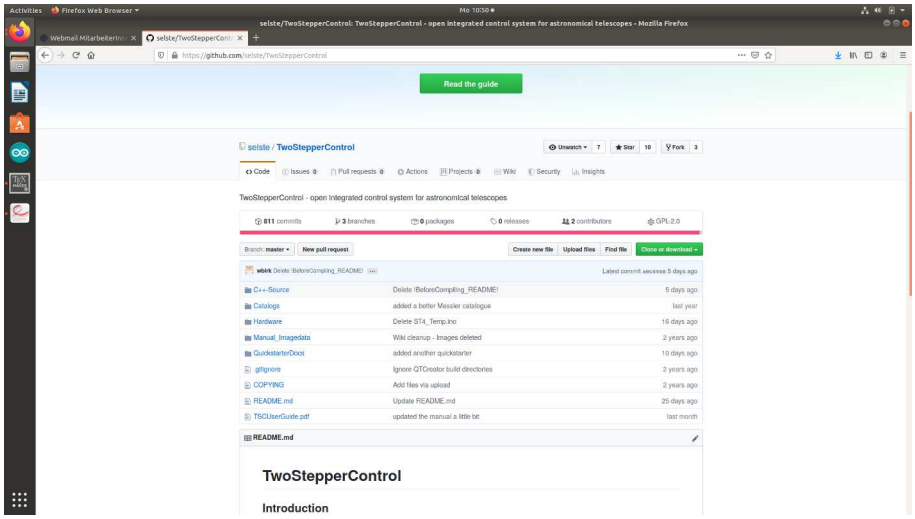


Figure 34: Screenshot of the home site for TSC on github. In the directory `C++-Source`, the sourcecode for TSC is found. The project can be directly compiled using Qt-Creator, installed on the Raspian image – take care to read the `!!!111!!!READMEBeforeCompiling.pdf` document for detailed instructions. `Catalogs` holds a number of observation lists which can be manually edited using common spreadsheet programs such as Microsoft Excel or OpenOffice. `Hardware` holds the PCB layouts and the sketches for Arduino - style microcontrollers as well as a list of materials. `Manual_Imagedata` contains pictures used in the online documentation. `QuickstarterDocs` is a series of very short in troductions to specific functionalites of TSC.

`sudo dd if=TSC.img of=/dev/mmcblk0 BS=8M status=progress`
to copy the image to your SD card. Denote that the device for the SD-card might have a different name than `/dev/mmcblk0` in dependence of your installation.

If you want to change the size of the partitions on your SD- card, you may consider using a utility like *gparted* in Linux – by using this, up to 128 GB SD cards were successfully used.

Insert the SD card to your Raspberry Pi and power it up. Now connect a HDMI-capable monitor to the Raspberry Pi and see the screen of the Raspberry. Now, one can explore the possibilities of TSC; however, two more additional configuration steps can be carried out, which are presented in the following two subsections.

9.2 Changing the screen resolution of the Raspberry

Figure 35: The HDMI touchscreen display available from Adafruit.com, which can be used in connection with TSC. Photo taken from www.adafruit.com.

By default, the software is designed for use with a 800 × 480 pixels. Any HDMI display can, however, be attached. The two HDMI touchscreen displays that were tested are the HDMI 5" 800x480 Display Backpack from www.adafruit.com or the corresponding 7" display from adafruit. Denote that the Raspberry Pi display **does not work** as the I^2C port of the Raspberry Pi is occupied by the hardware clock. Fig. 35 shows the display from the Adafruit-website. However, if you want to change the resolution of the Raspberry Pi, you have to open a terminal (by choosing it for the menu or typing `Ctrl-alt-T`) on the Raspberry. Type

```
pi@TSC:  $ sudo nano /boot/config.txt
```

What you see now is a file that configures the Raspberry (see also Fig. 36). Scroll to the lines reading

```
# uncomment to force a specific HDMI mode (here we are forcing 800x480!)
hdmi_group=2
```

`hdmi_mode=1`
`hdmi_mode=87`
`hdmi_cvt=800 480 60 6 0 0 0`
and put a comment sign (`#`) in front of the four lines starting with `hdmi...`. The command `Ctrl O` saves the file and `Ctrl X` terminates the `nano` editor. Now type `sudo reboot` in the terminal. The Raspberry reboots and now runs with its native resolution of 1920 × 1200 pixels. *Denote that on the Raspberry 4, the highest resolution of 2560 times 1440 pixels can jam the WLAN-signal.*

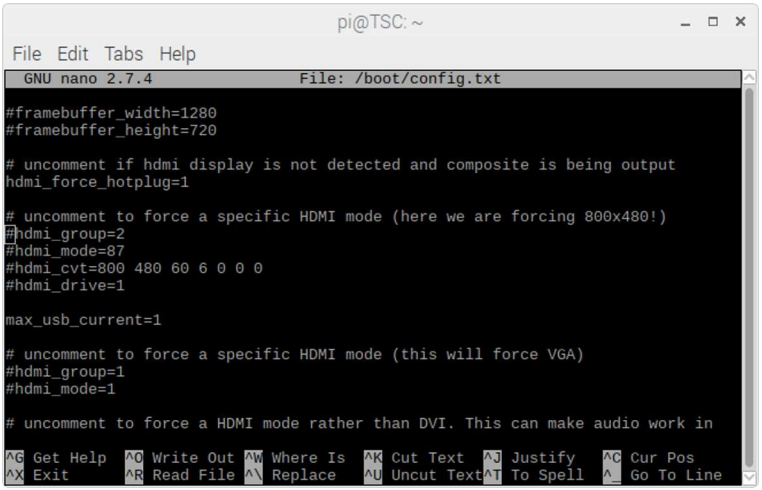


Figure 36: Configuration of the screen resolution of the Raspberry using VNC.

9.3 Setting up autonomous WLAN

TSC is set up following the recommendations from `https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md`. If no known access point is available, it opens up its own WLAN with the aforementioned access point`TSCHotspot`. *The password for accessing this hotspot is* **TSCRaspi**. The hotspot mode is indicated by two opposing arrows in the menu bar – see also Fig. 37.

Figure 37: If the autonomous WLAN with the access point `TSCHotspot` is opened, this is indicated by two opposing arrows.

9.3.1 Adding your WLAN

If you have a network already up and running, the easiest way to connect TSC to this network is to plug a Ethernet-cable into the Raspberry; by doing so, all traffic is routed also via the Ethernet connection and access to the outside

world including a timeserver is possible. Alternatively, one can add a simple WLAN repeater, adjust this device to access the **TSCHotspot**, and connect the repeater via a short WLAN cable to your router. Wolfgang realized remote access in his observatory by combining such a repeater and a devolo dLAN router.

9.4 Modifications to Raspian Buster

It is not recommended to setup your own Raspian image; however, it is noteworthy to list the modifications:

- The SPI, I^2C and Serial interfaces were activated in the configuration menu of the Pi.
- Instructions for setting up the hotspot were taken from `https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md`
- The latest INDI-version 1.8.1 was installed[19].
- Qt 5.7 and the OpenCV developer version were installed from Raspian repositories.
- The Arduino IDE V. 1.8.9 was installed from `www.arduino.cc`.
- udev rules for the Teensy family of microcontrollers were installed from `www.pjrc.com`.
- Fritzing was installed from the repositories.
- Large packages such as Wolfram Alpha were removed.

9.5 Compiling new versions of TSC

The whole project is in flow, and therefore it makes sense to update TSC once in a while; for this purpose, one has to got to the github-repository and click the **Clone or Download** button. This gives you a complete copy of the whole repository.

In a next step, the directory `TwoStepperControl-master` is being extracted by clicking on the zip-Archive. Fig. 38 illustrates this.

A new version can be compiled by

- extracting the `rc.tar` folder in `C++-Source/qdarkstyle`.
- clicking `TwoStepperControl.pro` in the folder `C++-Source`. This opens the QT-Designer IDE.
- setting the Build mode to "Release" in the lower left corner of the QTCreator and

[19] http://www.indilib.org/download/raspberry-pi.html

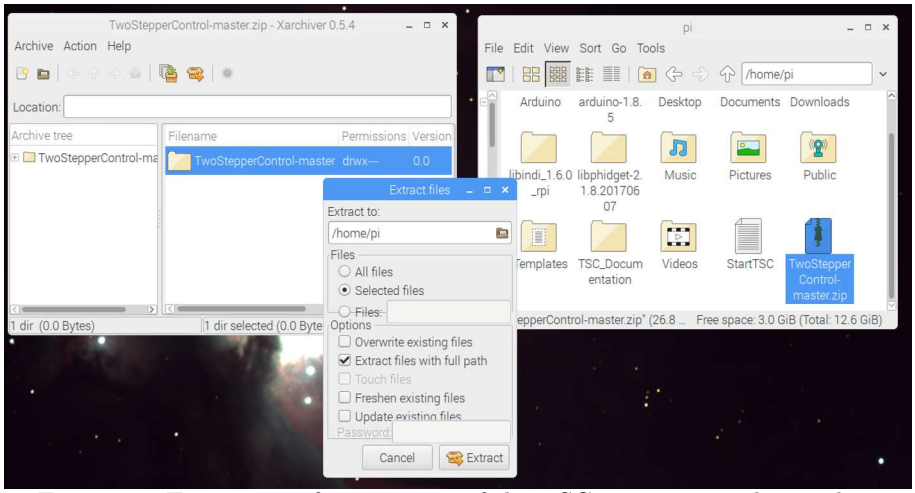


Figure 38: Extraction of a new copy of the TSC-repository. The result is a new `TwoStepperControl-master` archive with up to date materials.

- clicking the menu point
  `Build` → `Build Project "TwoStepperControl"`
  which generates a new copy of TSC in the directory
  `build-TwoStepperControl-Desktop-Release`.

Be aware that your preferences are now gone if they were not saved (they are stored in the build-directory) and that you need to copy the `Catalogs` folder again into the working directory. Details on compilation can be found in the file `!!!111!!!READMEBeforeCompiling.pdf` document in the source folder.

9.6 Functionality so far

You can now run first initial checks of TSC and connect your guiding camera to see whether it connects properly; however, no drivers are connected. Also, some of the additional hardware of TSC is missing – this includes the clock, the USB/Serial converter for LX200 operation and a few other goodies. In the next section, we will discuss the assembly of the HAT and the connection of the stepper drivers to the power supply.

10 Basic setup: HAT assembly and power supply

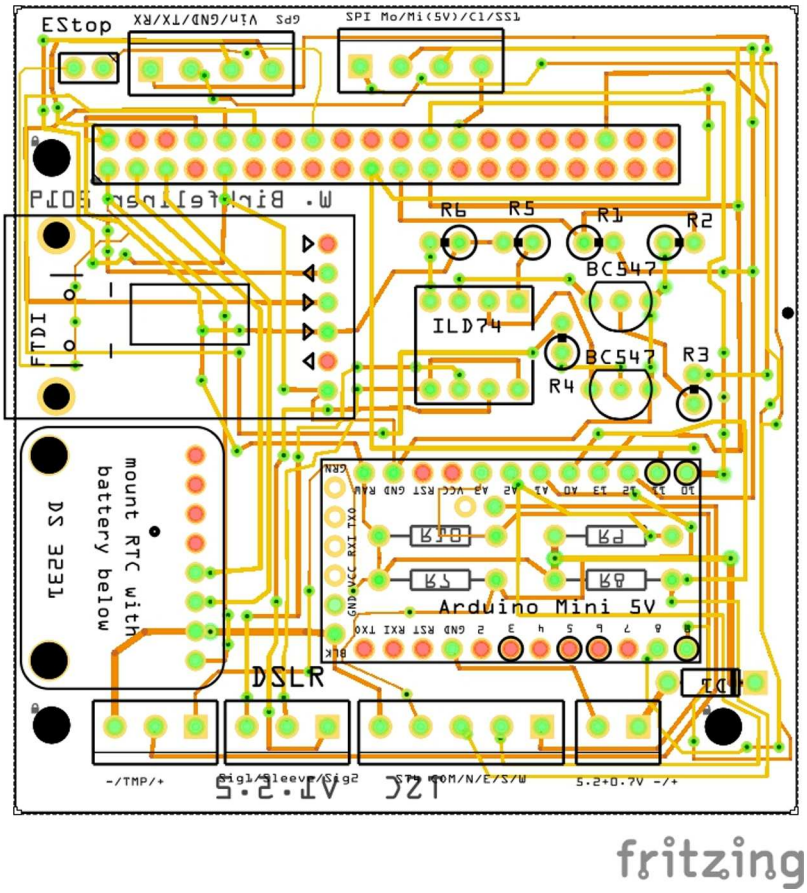


Figure 39: Top view of the PCB, where all components aside from the 40 pin connector to the Raspberry Pi are mounted. The PCB can be manufactured from the Fritzing manufacturing service, which is linked in the Fritzing software.

The next step is to build the basic additional hardware of TSC and to connect the stepper driver boards and the 12V and 5V power supply for the single components. The basic interfaces are positioned on a dedicated printed circuit board (PCB) called the TSC HAT; the PCB layout is to be found on the repository in the `Hardware` directory. The subdirectory of interest is called `1_TSC_PiHAT_GPS`. Here, a Fritzing file (currently `TSC_PiHat_ILD_V1_5.2.fzz` holds the breadboard layout, the schematic and the PCB layout. The HAT holds the following components:

- A DS 3231 precision hardware clock from Adafruit; this is a battery powered clock featuring temperature compensation. Fig. 41 shows this component which is about 15 €. A CR1210 battery is also needed.

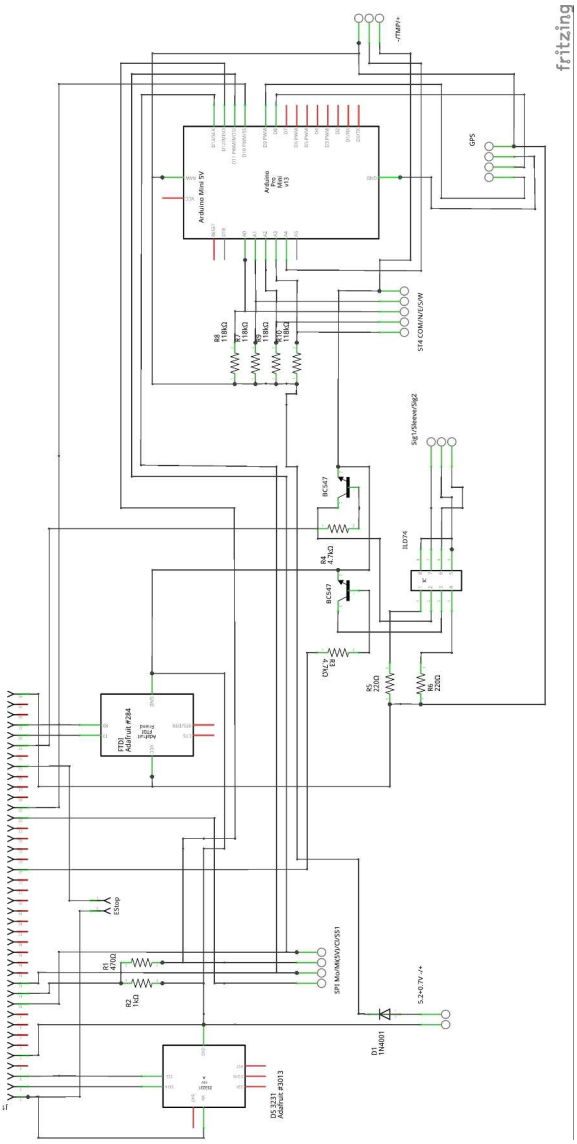


Figure 40: The schematic of the TSC HAT mounted on the Raspberry Pi.

- A connector for SPI, which is used to control the optional focuser motor board. The HAT features a voltage divider which converts the 5V MISO-signal to 3.3V for the SPI connector of the Raspberry Pi.
- A FTDI – compliant USB to RS232 converter for LX200 connection. Here, the Adafruit CP2104 Friend – USB to Serial Converter(cost is about 6.5 €) is used. Fig. 42 shows this device.
- A 5V 16 MHz Arduino Mini Pro from Sparkfun. Cost is approximately 10 €. This microcontroller connects internally via SPI to the Raspberry and is used for reading external ST4 commands for autoguiding. In addition, the temperature sensor is read by the Arduino. In short, it acts as a somewhat clever analog-digital converter.
- A connector for releasing a DSLR such as the Canon EOS series. The release is triggered by the Raspberry Pi GPIO pins and is isolated via a ILD 74 optocoupler from the DSLR.
- Connectors for external 5V supply voltages, ST4 signals, a (not yet realized) connection for an external GPS receiver, an emergency stop button and the emperature sensor.

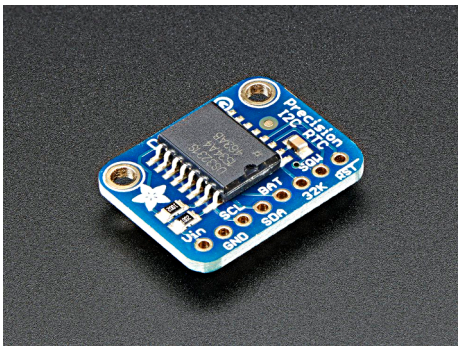


Figure 41: The DS 3231 hardware clock from Adafruit. Photo taken from www.adafruit.com.

For assembly of the HAT, a PCB and several components besides those listed above are needed. A detailed list if found in the `Hardware` directory of the repository as an Excel-file called `TSC_PartsList.xls`. Here, the resistor values are also noted. Fig. 40 shows the schematic of the HAT.

Soldering the HAT is simple as no SMD components and so on are used. The Arduino Mini Pro needs special attention as prior to soldering it, the four resistors R7-R10 need to be placed on the bootm. All other components aside from the connector to the Raspberry Pi are top mounted. Figs. 39 show the PCB. Making three PCBs using the manufacturing service of Fritzing accounts for approximately 30 €.

Soldering should start with the 40 pin connector to the Raspberry (see also Fig. 43). Next are the resistors R7 to R10 to be sodlered to the bottom of the PCB. All other resistors are mounted in a standup fashion. The following table lists

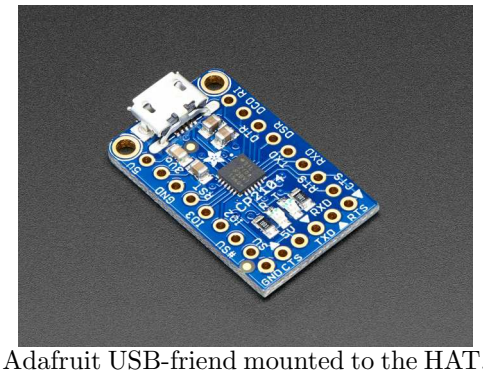


Figure 42: The Adafruit USB-friend mounted to the HAT. It connects an extrnal computer via USB to the RS232 serial interface of the Raspberry Pi and allows for LX200 operation from external programs such as Cartes du Ciel. Photograph from www.adafruit.com.

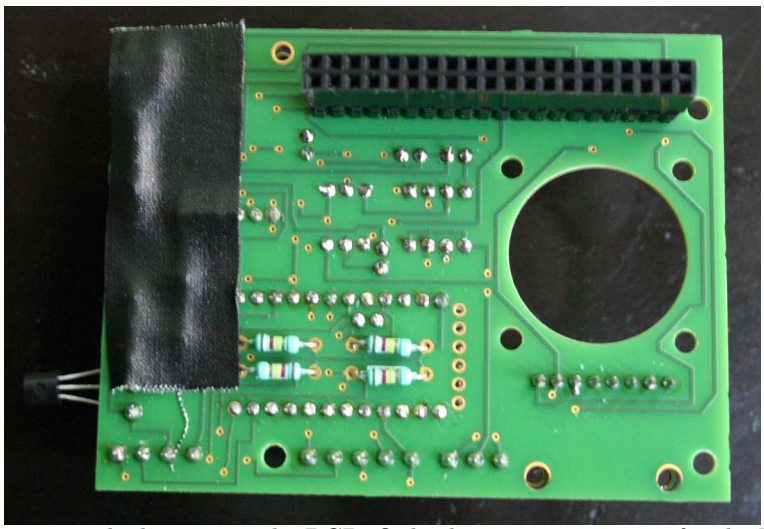


Figure 43: The bottom ot the PCB. Only the 40 pin connector for the Raspberry Pi 3 and four resistors are placed placed here. In order to avoid short circuiting the power supply for the Arduino Mini Pro, a strip of insulating tape was put here as well.

all resistor values:

Resistor	Value [Ω]
R1	470
R2	1000
R3, R4	4700
R5, R6	220
R7-10	118000

The optocoupler (ILD 74) for DSLR release – denote the proper orientation – the NPN transistors (BC547) and the polarity protection diode D1 are next.

Next is the Arduino Mini. Denote that the short side of the Arduino is not soldered to the PCB. As an eccentric pin (the analog pin A4) is also used for reading the temperature sensor, it is necessary to add two connectors as seen in Fig. 44). This is somewhat delicate.

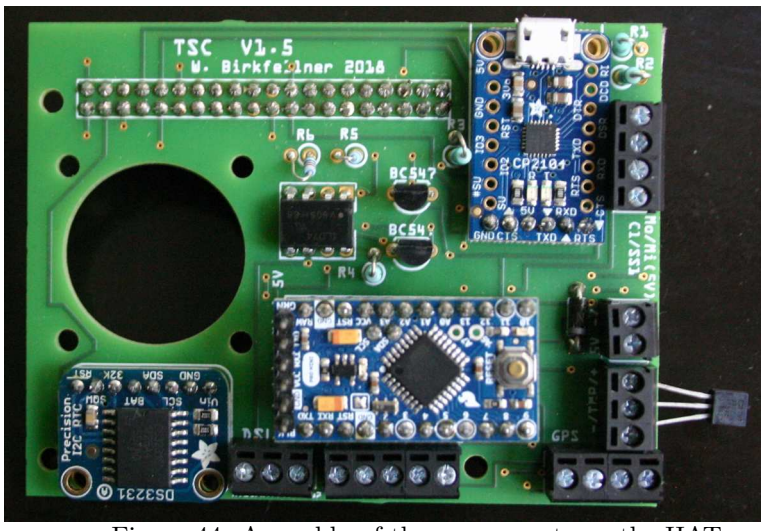


Figure 44: Assembly of the components on the HAT.

A fourth pinstripe with pins pointing away from the PCB is to be soldered to the Arduino Mini Pro for Programming the Arduino (see also Fig. 45). Denote that for programming the Arduino, the USB Friend converter is needed. As you might need that device later as well, it would be recommended to buy a second one of these converters. Finally, the DS 3231 and the USB friend (denoted as FTDI on the PCB) are mounted. If you want to use a socket on these is up to you. Unsoldering these devices is however difficult. In the end, the fan for the Raspberry Pi CPU is to be mounted.

Denote that the 40pin-connector on the HAT is not high enough; two more strips with sufficient length are necessary.

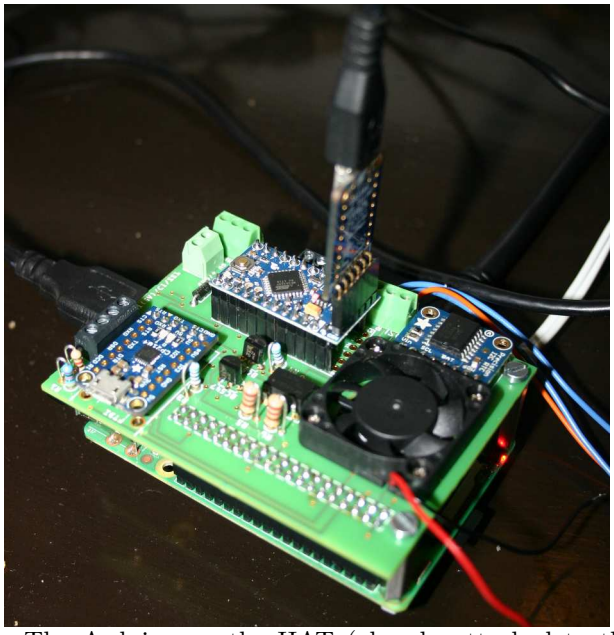


Figure 45: The Arduino on the HAT (already attached to the Raspberry in this illustration) acts as a Analog-Digital converter. It reads voltage levels from ST4 input and the temperature sensor. These data are transferred via SPI channel 0 to the Raspberry Pi. For programming, a sketch (`ST4_Temp.ino`) is to be uploaded using the Arduino IDE, which is installed on the Raspian system supplied alongside with TSC. A USB/Serial converter acts as the interface of the Arduino to the IDE.

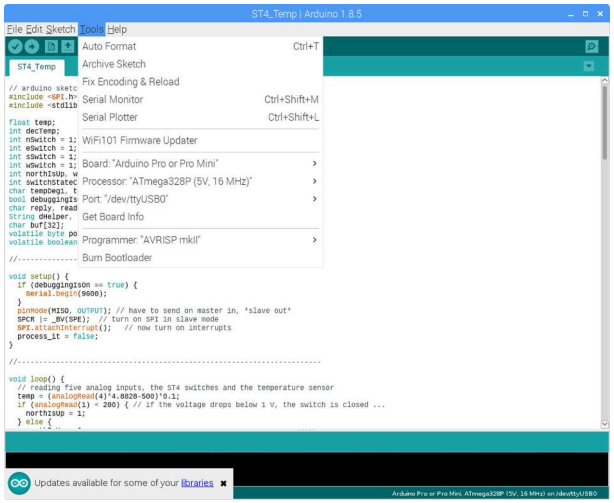


Figure 46: Settings for loading a sketch to the Arduino Mini Pro. Make sure that your USB/Serial converter is connected properly. The Arduino is a Mini Pro 5V/16 Hz - make sure that it is selected like in this illustration here. Once the → button in the icon list is checked, the sketch is uploaded to the Arduino.

10.1 Programming the Arduino Mini Pro

In order to program the Arduino, you need a USB-to-RS232 converter (just like the CP2104 Friend – USB converter used on the PCB) and the Arduino integrated development environment (IDE). The IDE can either be downloaded from www.arduino.cc, or the installed Arduino IDE on the Raspberry Pi itself can be used. Connect the Arduino to your computer using the USB-friend and open the Arduino IDE. A sketch named `ST4_Temp.ino` is located in the folder `Hardware/1_TSC _PiHAT_GPS`. This sketch is to be uploaded to the Arduino mounted to the HAT – it is responsible for measuring ST4 voltage levels and reading out the temperature sensor. Make sure to select the 5V/16MHz version of the Arduino Mini Pro in the Arduino IDE. It communicates with the Raspberry Pi using SPI channel 0. Denote that on the HAT, a voltage divider is used to shift the MISO level of the Arduino (5 V) to the 3.3 V required by the Raspberry. Fig. 45 shows the completed HAT mounted to the Raspberry Pi and the attached USB-converter for uploading. Denote that it might be necessary to adjust some values like the reference voltage for the temperature sensor in the sketch. More on this follows in Chapter 10.8.

10.2 The driver board

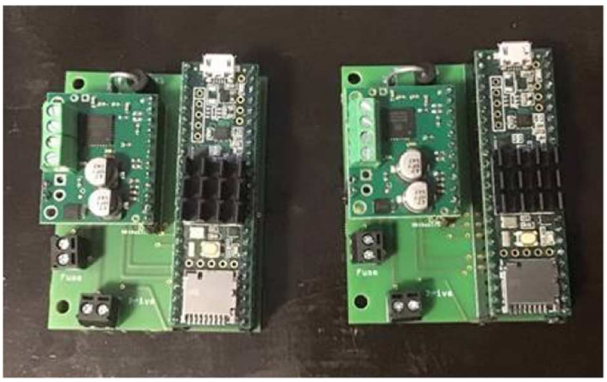


Figure 47: The assembled right ascension and declination drivers. Connection to the Raspberry is realized with short USB cables running from the Teensies to the Raspberry Pi.

There is little hardware for the driver boards necessary; on needs

- Two Teensy 4 microcontroller with headers
- Two Pololu AMIS 30543 breakout boards
- Two 3.5 mm terminals with 2 inputs
- Two 5.2 mm terminals with 4 inputs for connecting the steppers.
- Two 5×20 glass fuses and holders with 3 A maximum current.
- One rectifier diode with maximum 6 A current.

- Terminals and a RJ12 conenctor for ST4.

Basically, the hardware consists of the Teensies, the Pololu PCB, the terminals and a diode for reverse coltage protection. Alternatively, the motors can be attached to a terminal with four outlets on the Pololu board. Fig. 48 show the layout. The two Teensys need to be programmed with two different sketches found in `TwoStepperControl/Hardware/AMIS_Drives`; they are named `AMIS_Stepperinterface_DE` and `AMIS_Stepperinterface_RA` for the right ascension and declination drives. A Fritzing file for the PCB is also to be found there. The driver boards are simply connected via short USB cables to the Raspberry Pi. *Use feamle headers to connect the AMIS boards as these can blow and need to be replaced in such a case.*

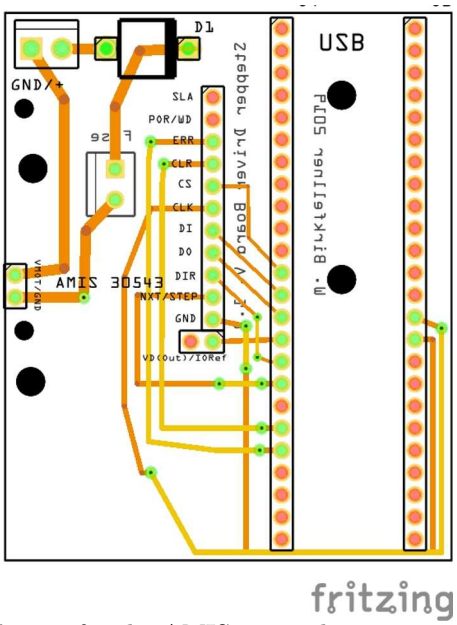


Figure 48: PCB layout for the AMIS 30543 drivers; orientation of the Teensy is indicated by the USB - mark. The bipolar steppers are connected with terminals mounted on the Pololu breakout board.

10.3 Connecting the HAT and the driver board

The Hat is simply put onto the Raspberry Pi; connection to the Teensies on the driver is realized via short USB cables. If you use a Raspberry 4, take care that these connectors occupy the USB2 ports. The driver board can be mounted atop of the HAT, but the screws for the standoffs are partially under the AMIS breakouts; however, the AMIS boards need to be mounted with headers anyhow as mentioned before. The ST4 connector needs to be connected to the ST4 terminal on the HAT with short wires.

10.4 Power supply

For connecting basic components, you will need

- A sufficient 12V Power supply.
- A 12V – 5V converter providing up to 3A.
- A panel mount USB jack like the Adafruit #908 adapter (see also Fig. 49).
- Jacks for connecting the stepper motors.
- An On/Off switch.
- A 2.5 mm jack for connecting the DSLR.
- A jack for the temperature sensor.
- Short Mini- and Micro-USB cables to connect the stepper driver boards and the touchscreen.

Figure 49: A panel mount USB jack needed for connecting the guiding camera – this one is sold by Adafruit. Image courtesy of Adafruit.

In general, two stable voltage levels need to be supplied; at least 12V need go to the stepper drivers (higher voltages up to 30 V allow for faster travel), the focuser motorboard (if needed) and, of course, the 12V – 5V converter. 5V have to be supplied to the Raspberry Pi itself, the HAT and the optional focuser board . All powerlines should be connected *in parallel*. Fig. 50 shows a diagram of all components to be connected. The regulated power from the converter goes into the HAT and, if necessary, into the focuser board; as both are protected by rectifier diodes, the minimum voltage is approximately 5.7 V. *If the Raspberry shows the undervoltage warning (a lightning symbol on the right hand side of the screen), this voltage needs to be slighlty increased. No more thatn 5.2V should be supplied to the Rasperry!*

10.5 Connecting stepper motors

TSC operates bipolar stepper motors. In its basic form, a bipolar stepper has four wires coming out of the motor body; these are usually labeled like A-A'

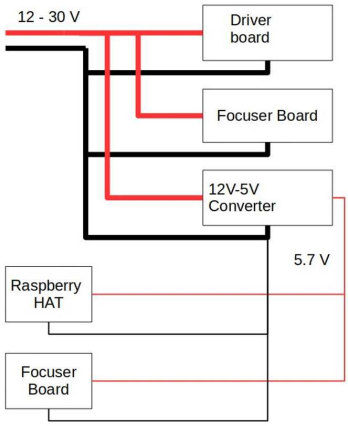


Figure 50: Wiring of power lines; take care that all lines are connected in parallel, and that they all share the same ground. The 12V – 5V voltage regulator should be at 5.2V maximum. This should be measured as some of these regulators do have a display which is not necessarily very accurate.

and B-B'. These pairs belong to one coil, and therefore the resistance between the pairs is very low. So if you put a multimeter on A and A' (or B and B'), resistance drops to a comparatively low value. If you have more than four wires coming out of your stepper, you either have an unipolar stepper in front of you or a bipolar stepper with sense lines. In these cases, consult the documentation. Connect A and A' to the two neighboring terminals of the stepper driver boards, and B and B' to the other two. **If you want to invert the default direction of your stepper, switch the pairs when connecting them to the drivers or the focuser board.**

10.6 Connecting ST4

Basically, ST 4 consists of a combination of four wires and a common ground. If one of the wires is short-circuited to ground, this direction is active. In reality, this is not as simple as that as some ST4 adapters do not drop to 0 Ω and the voltage between ground and signal is not zero. Originally, the ST4 interface consisted of mechanical relays which powered up a motor. In addition, the pinout of the connector is not well defined – nowadays, a RJ 12 connector and a six wire ribbon cable are common. TSC does not have a RJ 12 connector on the HAT, but rather than that, the 5 wires (**N**orth, **E**ast, **S**outh, **W**est and Ground (GND) or *Common* are connected with a terminal strip. These wires are connected to a jack and therefore, it is possible to customize the ST4 interface for your particular adapter. The common connection for the RJ12 jack is:

Connector	1	2	3	4	5	6
Direction	-	GND	RA + (W)	Dec. + (N)	Dec. - (S)	RA - (E)

The ST4 levels are read by the analog inputs A0 – A3 of the Arduino Mini Pro on the HAT and transferred to the Pi via SPI channel 0.

10.7 Connecting a DSLR

Another 3 pin terminal strip (labeled **DSLR**) allows to connect a camera such as a Canon EOS to TSC. TSC can trigger exposures and exposure series using the internal GPIO pins of the Raspberry Pi. These connectors are isolated by the ILD 74 optocoupler on the HAT. As there are many possible connections for different camera models, it is suggested to find out about your specific shutter release. For the Canon EOS DSLR, the tip of the plug triggers exposure, ring is meant for focusing and sleeve is connected to ground (GND). Currently, only the tip-contact is triggered by TSC. As the cables are connected via a terminal strip and the input is isolated, no damage can be done to the camera if the signals are connected the wrong way - an erratic camera behaviour is, however, the result. You can test this by making a single exposure using TSC with manual (**M**) and **Bulb** mode on the Canon EOS series.

10.8 Connect the temperature sensor

Finally, an Analog Devices TMP36 temperature sensor (also available from Adafruit) can be connected using the **-/TMP/+** terminal strip on the HAT. Make sure that the three pins of the TMP36 are connected properly, otherwise the sensor will be damaged. If you don't want to use the temperature sensor, you can short circuit **-** and **TMP**. TSC will then always display -50^{o}C.

It is important to calibrate the sensor. It is connected to an analog input (A4) of the Arduino Mini Pro. In dependence of the 5V supply voltage, the readings may vary. If 5.2 V are supplied by the DC-DC converter, the Diode D1 will most likely make 4.6 V out of this. The Arduino can cope with that, but the reference voltage on the ADC is wrong. This can be adjusted by measuring the voltage on the RAW input of the Arduino Mini. The correct voltage has to be stored in the `ST4_Temp.ino`. Search for the line

```
float VRef = 4.52;
```

in the sketch (it is found in the initialization part) and insert your reference voltage before uploading the sketch to the Arduino.

The sensor is accurate to 2^{o}C. The temperature is reported all 30 seconds. In order to make these measurements stable, it is recommended to package the sensor in a metallic housing with high thermal capacity. This avoids jitter in the measurements, which is inevitable in such a setup.

10.9 Housing TSC

All of this needs some housing. The easiest solution is to put everything in a box like in Fig. 2; a more elegant solution is the use of Fischer Elektronik profiles like the GB 1/250/ME series. Height should be in the range of 80 mm. Fig. 51 shows a solution with $250 \times 190 \times 82$ mm where a sliding cover for the 7" Adafruit touchscreen was realized.

Figure 51: A professional case for TSC; overall dimensions are $250 \times 190 \times 82$ mm. Here, Fischer Elektronik GB 1/250/ME profiles were used. The top cover can be used as a lid for the 7" touchscreen. The sides are made of 2 mm transparent plastic. As TSC communicates via WLAN, it is recommended not to make a full metal case.

10.10 Synchronizing the hardware clock

Your raspberry has now a hardware clock (RTC) once all is assembled. *You need to give that device a battery and connect once to an external network as described in Sect. 9.3.1 in order to set the clock.* Synchronization of the hardware clock can take place in three ways:

- Manually by opening a terminal window and typing `timedatectl set-time 'yyyy-mm-dd hh:mm:ss'` followed by `sudo hwclock -w`.
- Connecting your Raspberry Pi to the Internet with a Ethernet cable and wait for synchronization.
- Use LX220 as described in Sect. 3.8.

Details on setting up a RTC on the Raspberry Pi can be found in `https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi`.

Part IV

Beyond basic operation – optional add-ons

The HAT is sufficient for most of TSCs functionality; however, a few ad-ons exist.

11 Choosing a display and a keyboard

Once TSC is setup, assembled and powered up, basic operation is possible. You can connect an external keyboard to the USB interface, and you can connect any monitor with a HDMI interface to the Raspberry Pi.

However, TSC is supposed to be a standalone device. HMDI monitors with a minimum resolution of 800 × 480 pixels are avalable from Adafruit and other vendors like Waveshare. Personally, we have tested the Adafruit 5" and 7" HDMI displays, but others ranging from 5" to 10" are available with touchscreen capability. 7" are completely sufficient, we recommend the 7.0" 40-pin TFT Display 800x480 with Touchscreen and the Adafruit TFP401 HDMI/DVI Decoder to 40-pin TTL Display. The cost for such a display is about 80 €– and it is worth every cent. The genuine Raspberry Pi display of the Raspberry Pi foundation requires I^2C access, and so does the realtime clock. I do not know if the address (0x68 for the RTC) is available in this case. For integrating the display into a case, it is recommend to get a 1.5 mm stainless steel plate of sufficient size and to attach the display with flexible double sided adhesive tape. As the displays are often connected with a flexprint, it makes sense to protect the flexprint from the edge of the sheet metal with some tape as well. On the bottom side, the HDMI-converter is attached. In a later stage the USB-hub was also attached to the bottom of the metal plate. Figs. 52 and 53 illustrate this setup.

In Fig. 52, a small wireless keyboard is also shown. While both the display and the keyboard are not absolutely needed as VNC already provides an interface, they are highly recommended. You will love them.

12 The wireless handbox

Another feature of TSC is the support of a wireless handbox. While manual operation is feasible with planetarium programs like Sky Safari or Cartes du Ciel and via the user interface, touchscreens are a cumbersome and difficult to operate piece of equipment in darkness. Handboxes with four direction buttons provide a proven interface. However, many of them are wired, and this is avoided in the case of TSC by a custom wireless handbox.

Figure 52: The Adafruit 7” display, also shown in Fig. 35, taped to a thin stainless steel plate. A miniature wireless keyboard is also shown in the image. Denote the protective strip of adhesive top on the lower edge of the plate. It protects the connecting flexprint from damage.

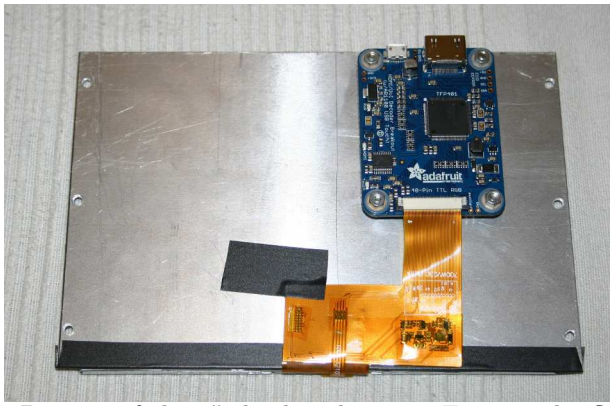


Figure 53: Bottom of the 7” display shown in Fig. 52; the flexpint is secured with a short strip of tape, the HDMI converter is attached by 4 M3 screws.

Originally, two handbox designs existed – one of them used Bluetooth, the other one operates via the autonomous WLAN access point of TSC. Both supported slow and fast motion in four directions and provide an interface to the focuser motors (see also Sect. 13). The development of the two handboxes has historical reasons. The TCP/IP handbox features feedback via a small OLED display, it has the more powerful processor, and provides a stable communication via TCP/IP. It is also easier to assemble than the Bluetooth handbox. Therefore, the Bluetooth handbox was finally removed from TSC. The main reason for this decision lies in the fact that Bluetooth is, essentially, serial communication like in the 1960s. Loss of connection is not easy to handle and may force one to restart TSC in the worst of all cases.

12.1 The TCP/IP handbox

12.1.1 PCB assembly

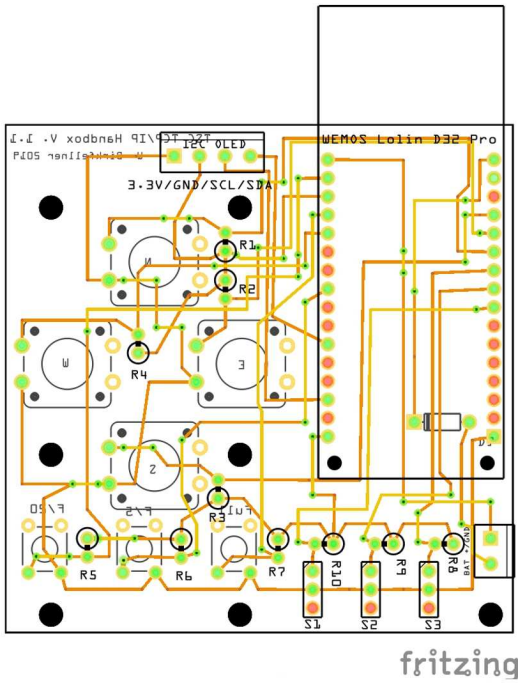


Figure 54: PCBs of the TCP/IP handbox. All momentary switches are integrated on the PCB, the microcontroller (a WEMOS LOLIN PRO D2) is powered by a Li/Po rechargeable battery. Permanent switches for handbox motion speed, focuser selection and -direction and a I^2C driven OLED display are connected via terminals.

Little is to be said about PCB assembly, Figs. 54 and 55 show the setup. The Fritzing file for the schematic is to be found in the `Hardware` folder under `3_TCP_Handbox`. Basically, seven momentary switches are mounted to the top of the PCB as it can be seen in Fig. 56. The bigger ones with a $12 \times 12\text{mm}^2$ footprint are the direction switches, and the three smaller ones allow to move one

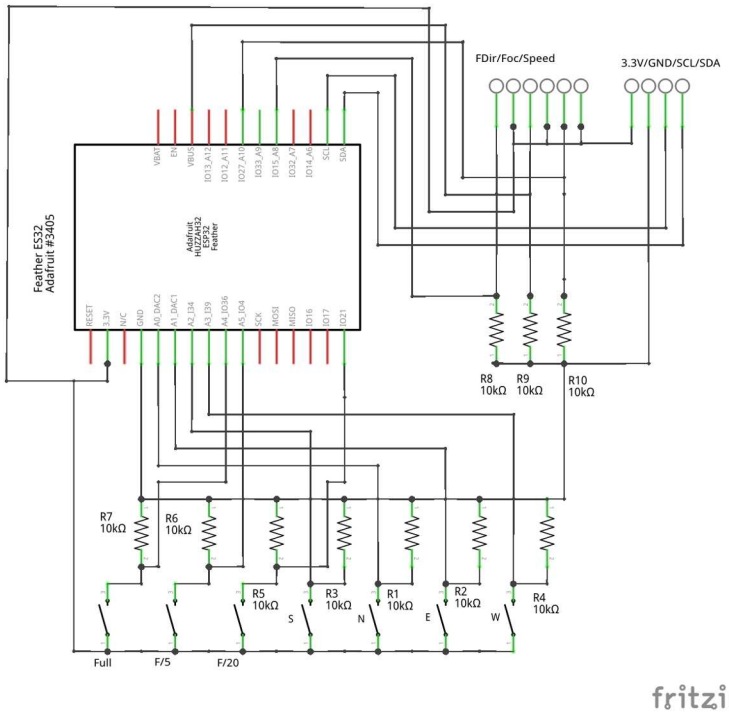


Figure 55: Schematic of the TCP/IP handbox. Basically, ten switches and pull-down resistors are connected to the inputs of the WEMOS LOLIN PRO D2. An OLED display is driven via I^2C.

of the focusers in increments. On the backside (Fig. 57), a microcontroller with integrated WLAN (the WEMOS LOLIN POR D2), ten pull-down resistors (all with the same rating) and two terminals are located. The terminals are meant for connecting three toggle switches (for selecting handbox motion speed, the focuser and the focuser direction) and for connecting a I^2C driven OLED display with 128×64 pixels. Power is supplied by a 3.7V LiPo rechargeable battery. These are connected to the JST connector on the Adafruit Feather. Charging takes place by connecting the Feather to a USB power source via the micro USB adapter of the microcontroller. Such battery packs are widely available, for instance from Adafruit itself or the German company Eckstein components for less than 10 €.

The display is available from ebay or similar sources for 4 – 10 €. It should utilize the SSH1106 chipset, otherwise reprogramming the sketch for the LOLIN becomes necessary. The PCB is not laid out for OLED displays requiring a 3-wire or 4-wire SPI interface.

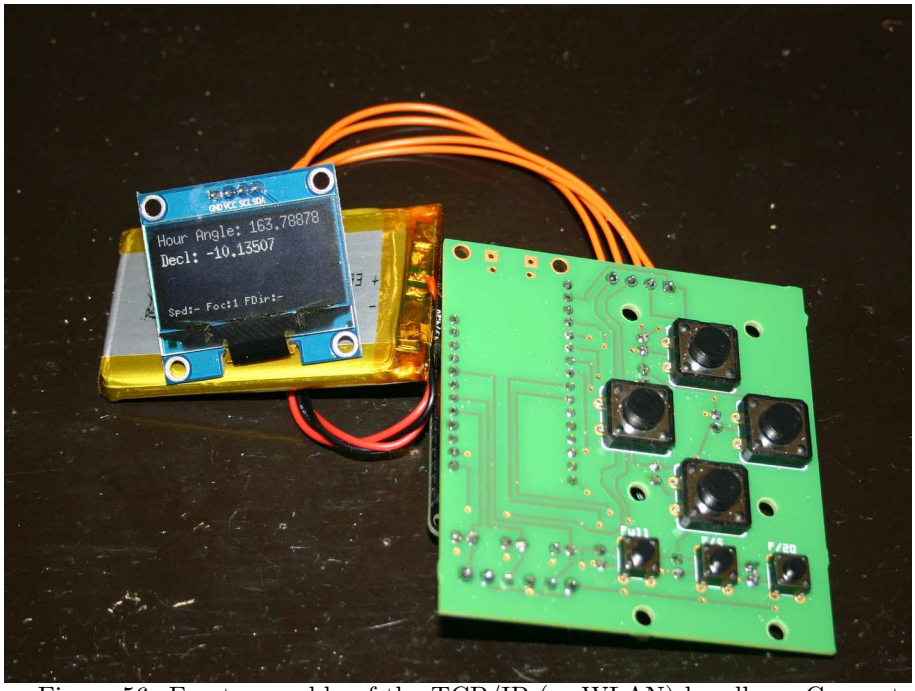


Figure 56: Front assembly of the TCP/IP (or WLAN) handbox. Connected is also a 1.3" OLED display running off 3.3V and an I^2C interface. The program on the Adafruit Huzzah32 ESP32 assumes that the display is controlled by an SH1106 chip and has a resolution of 128×64 pixels. Changes are feasible in the `TCP_Handbox.ino` sketch. The handbox driven by a LiPo battery which can be charged via the USB connector of the microcontroller.

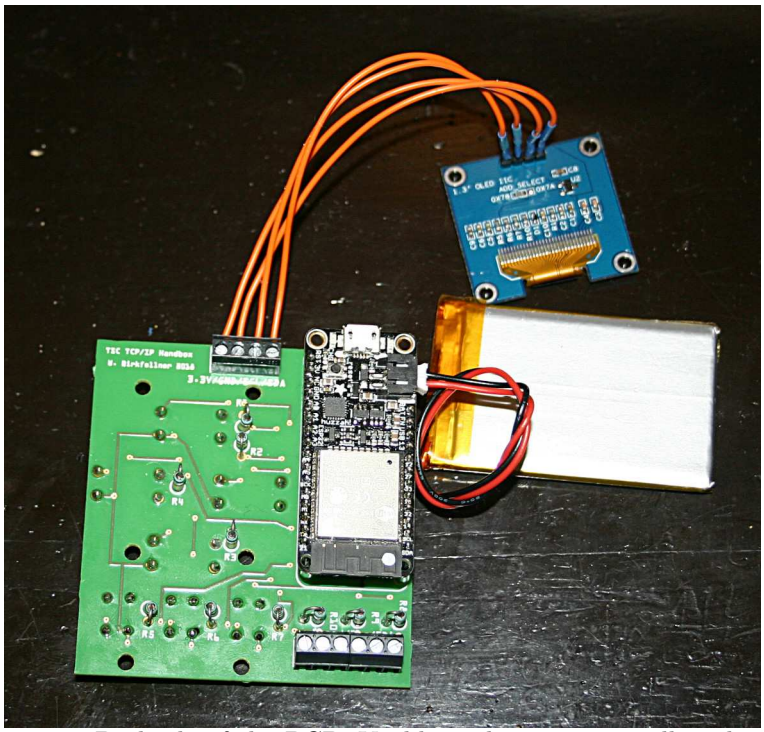


Figure 57: Backside of the PCB. Visible is the microcontroller, the pull down resistors and the terminals for the three selection switches for motion speed, focuser selection and focuser direction.

12.1.2 Programming the WEMOS LOLIN PRO D2

In principle, programming the WEMOS is not different from programming the Arduino Mini for the HAT. However, a different microcontroller is used, and therefore a different compiler has to be uploaded to the Arduino IDE.

The sketch for the handbox is to be found on github in the `Hardware` folder in the `3_TCP_Handbox` folder as `TCP_HandboxWemos.ino`. As the WEMOS has a direct USB connector, not additional adapters are necessary.

12.1.3 Connection settings and functions of the TCP/IP handbox

Under the following circumstances, the TCP/IP handbox should connect itself to TSC:

- TSC is in standalone hotspot mode and has spawned an access point named *TSCHotspot*. This is set in `/etc/hostapd/hostapd.conf`.
- The password is set to `TSCRaspi`. This is also set in the file `/etc/hostapd/hostapd.conf`.

In this case, the handbox finds the hotspot after a few seconds and claims that it is *Waiting for TSC...* – see also Fig. 58.

Next, TSC has to open a server socket for the handbox. This is done in TSC by operating the **TCP/IP Handbox** dialogue as described in Sect. 3.3. In short, one has to

- select the internal IP address `192.68.50.5` in the list of provided IP addresses and
- click the **Enable WLAN HBox** pushbutton.

The **Handbox connected** checkbox should go up and the handbox gives a message *Connected to TSC!*. After that, the state of the three stationary switches (handbox motion speed, focuser selection and focuser direction) together with the topical position of the mount is updated every 2 seconds (see also Fig. 58.

Denote that these values are hardcoded in the `TSC_Handbox` sketch – a different port number for the server socket for instance will cause failure of the connection. Fig. 59 shows the hardcoded values in the sketch.

And now for the real cool part: *If you loose connection, the handbox reboots and waits until it can re-establish contact to the hotspot, and then it re-connects to TSC.*

12.1.4 Other status messages from the TCP/IP handbox

By utilizing the high data rate of TCP/IP, it is possible to use the display of the handbox as a small status monitor. If you operate a direction switch, the corresponding direction is displayed in the OLED. During a GoTo, the time to arrival is displayed (so you can go outside of the observatory and have a beer under the stars while the telescope slews). And if one of the motion buttons for

Figure 58: Four screenshots of the initialisation phase of the handbox. First, the handbox tries to establish a connection to the hotspot of TSC, and once this was successful, it waits for TSC to open a server socket. If communciation is established, the topical position is displayed as hour angle and declination.

File Edit Sketch Tools Help

TCP_Handbox

```
#include <Arduino.h>
#include <U8g2lib.h>
#include <Wire.h>
#include <WiFi.h>

U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);
const char* ssid     = "TSCHotspot"; // name of the autonomous hotspot of TSC
const char* password = "TSCRaspi"; // password for access to TSCHotspot
const char* TSCServer = "192.168.50.5";
const int TSCPort = 49153;
```

Figure 59: Settings for a working connection between TSC and the handbox. Denote also that the type of OLED display is fixed in line 6.

the focuser was operated, this is also reported. Fig. 60 shows a few screenshots.

Figure 60: Status messages during operation. When the handbox is operated, the direction is displayed (and position is updated every 2 seconds). In GoTo, the time to arrival is shown. Operating one of the focuser switches is also confirmed.

During DSLR exposure series, the remaining exposure time or the number of exposures already done and the time remaining for the running exposure is displayed (see also Fig. 61). If the internal autoguider of TSC is used, guiding errors are reported as well.

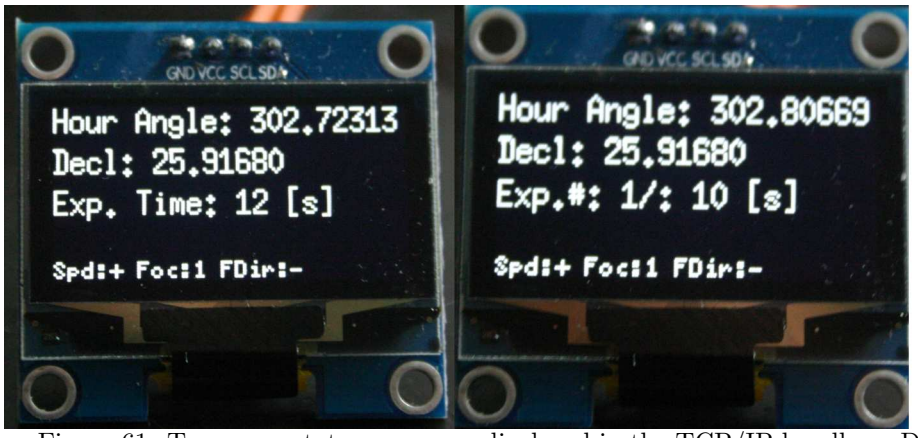


Figure 61: Two more status messages displayed in the TCP/IP handbox. During camera exposures, the remaining exposure time is displayed; in an exposure series, the number of exposures and the remaining time is shown.

12.1.5 Housing the TCP/IP handbox

Fig. 12.1 shows a housing solution. A standard 42 mm high aluminum case was used to to hold the PCB, the display and the LiPo-battery.

Figure 62: The TCP/IP handbox in a standard aluminum case with 42 mm height.

13 The focuser motorboard

The second optional (but useful) piece of hardware connected to TSC is the focuser board. TSC can control two additional steppers, which can, for instance, serve as focuser drives. These motors are not driven by the AMIS 30543 board but by small, cost-efficient and widely available drivers, the Polulu A4988 (with 16 microsteps) or the Polulu DRV 8825 (with 32 microsteps). These drivers can operate with coil currents in the range of 1A. A third microcontroller, again an Arduino Mini Pro (16 MHz, 5V version) operates these controllers. TSC communicates with this microcontroller over SPI channel 1. If the board is not connected, the corresponding GUI (as seen in Fig. 16) is deactivated. Denote that the operating voltage of SPI is 5V on the Arduino and 3.3V on the Raspberry; a voltage divider on the HAT takes care of regulating the voltage of signals coming from the Arduino.

The focuser drives are not meant for permanent operation; rather than that, one can choose a basic number of microsteps and a corresponding microstepping rate. The handbox or the configuration screen buttons labelled +, **1/5 +**, **1/20 +** and the repective buttons with the minus sign move the selected drive by that number of steps or a fraction thereof. If one of the drives is connected to the guiding scope, it is also possible to operate this drive from the **Guiding** part of TSC; the buttons are marked +++, ++ and + or — and so on; Fig. 14 shows these elements.

13.1 PCB assembly and motor current control

The Fritzing file for the board is found in GitHub in the `Hardware/4_FocusMotors` directory. Figs. 63 and 64 show the schematic and the PCB layout. The board is comparatively simple. It features a single capacitor, two diodes for polarity protection, the Arduino Mini Pro and several terminals for connecting 5V and 12V power, for connecting to the SPI-terminal on the TSC HAT, and for connecting two bipolar stepper drivers. Again, wires that belong to the same coil should be connected to the terminals denoted as `AA'` and `BB'` respectively.

Soldering the board is unproblematic. As stepper driver stages of this type tend to get hot and can break, it is advisable to put them on sockets. Fig. 65 shows the assembled board. The power of these drivers cannot be compared the AMIS 30543 boards, and the clock speed of the Arduino Mini Pro does not allow for very high speeds. In addition, the coil current cannot be set by software like in the case of the AMIS 30543 drivers – rather than that, a small potentiometer on the Polulu boards has to be adjusted. For this purpose, it is necessary to connect the steppers and to turn the potentiometers until smooth operation is achieved. If the current is set too high, the boards and motors will overheat and will be noisy. If it is too low, a sufficient torque is not achieved or the motors will not turn at all. Setting the current is done best with a controlled laboratory power supply. Or one can simple hold the motor spindle. Optimal current is set when it becomes difficult to hold the spindle by manual force, which of course also depends on the size of the motors. Bipolar stepper motors up to NEMA 17 size can be operated easily by the focusboards.

13.2 Programming the Arduino Mini Pro

The sketch for the Arduino Mini Pro is found in `Hardware/Focusmotors`; it is named `SPI_TSC_Slave.ino`. Uploading the sketch is simple as this is also a 5V Arduino Mini Pro – uploading takes place in the same manner as in Sect. 10.1. However, the type of driver board has to be given in the sketch as the A4988 (which is sufficient and, above all, a very robust board) features $\frac{1}{16}$ microstepping, whereas the DRV8825 can go down to $\frac{1}{32}$ microstepping. This is done in the line

```
const char whatDriver = 'A';
```

`A` stands for the A4988, whereas `D` denotes that a DRV8825 is used. For testing purposes, a sketch `Focusstepper_Test.ino` is also available in the repository; this one is for testing the board, and it just moves the motors back and forth.

13.3 Connecting the focuser board to the TSC HAT

Connecting the board to the HAT is quite easy. Attach the 5V and 12V connectors in a parallel fashion to the corresponding power sources (see also Fig. 50), and connect the four SPI lines from the focuser board to the HAT; the order is as follows:

Focuser Board	SS	SCK	MISO	MOSI
HAT	SS1	Cl	Mi (5V)	Mo

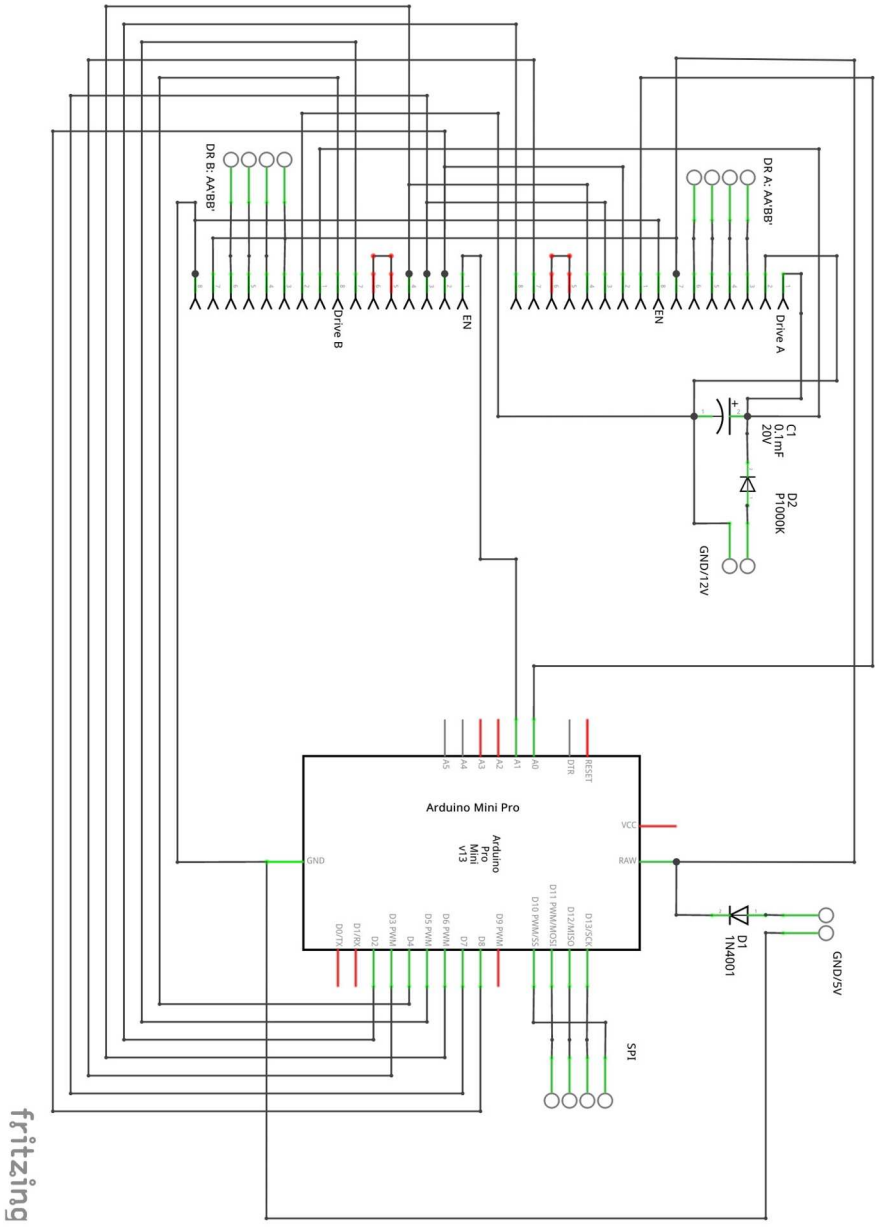


Figure 63: The schematic for the focuser board. The stepper drivers are controlled by an Arduino Mini Pro (16 MHz 5V version), which is controlled via SPI by TSC.

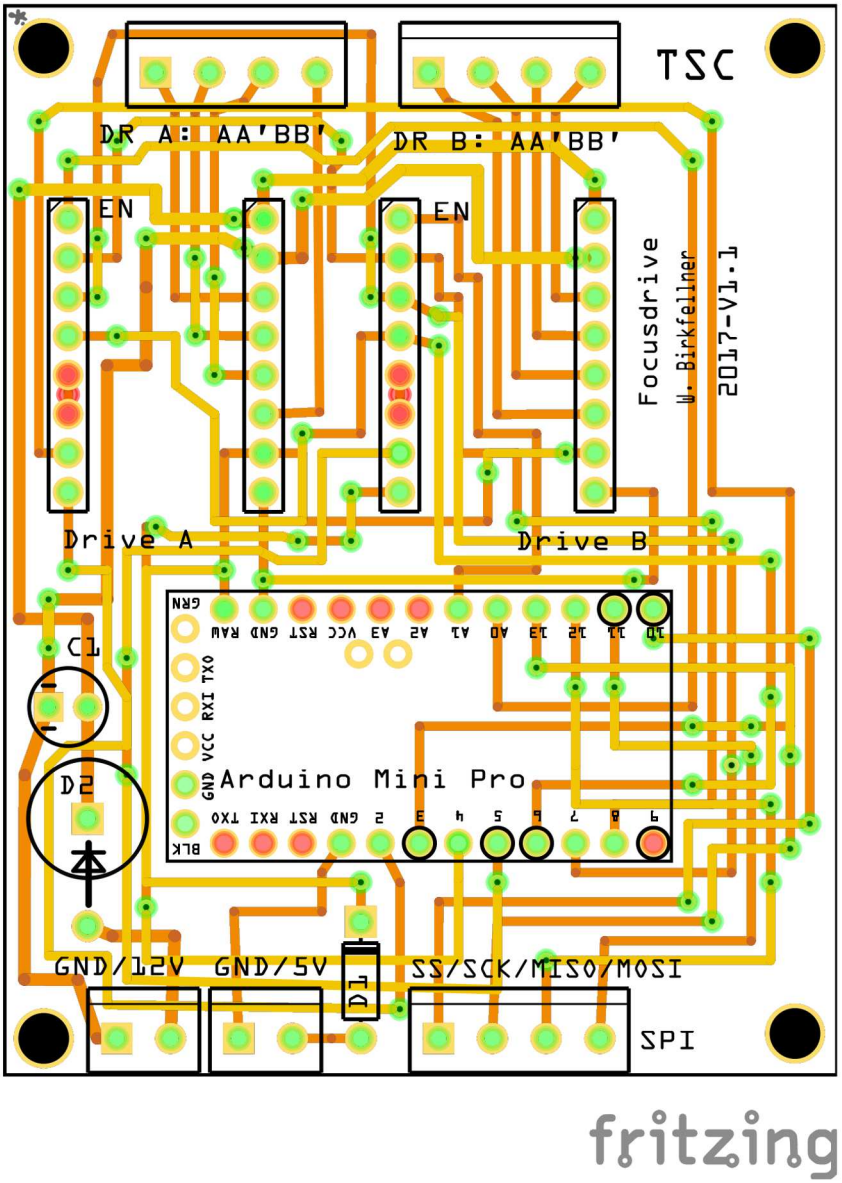


Figure 64: The PCB of the focuserbaord. The stepper controllers (Polulu A4988 or Polulu DRV 8825) are placed on sockets, and so is the Arduino Mini Pro.

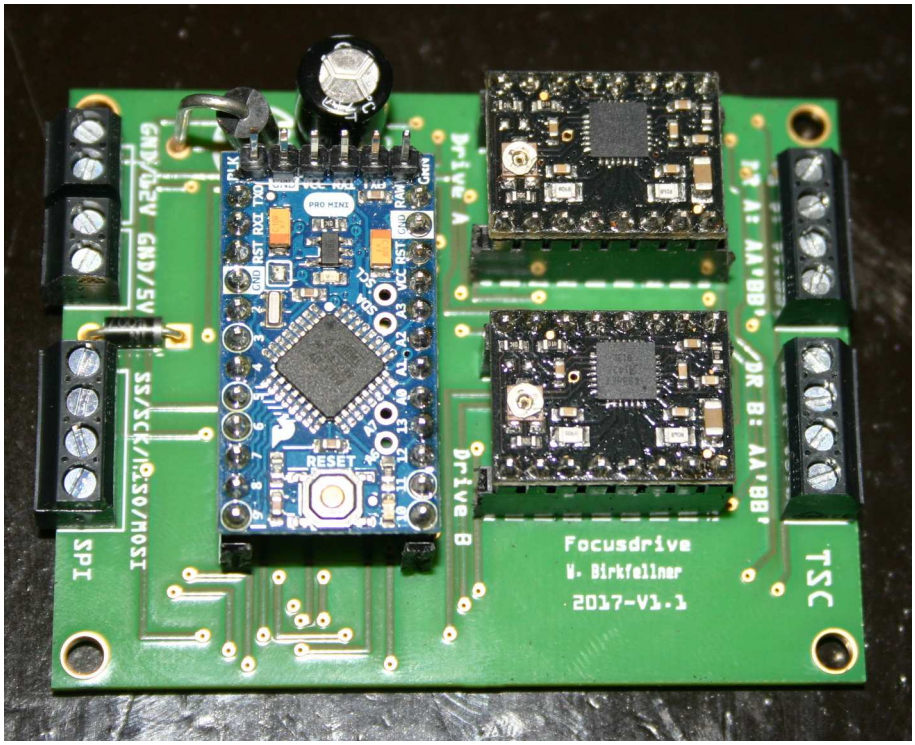


Figure 65: The assembled focuser board. In this case, Polulu A4988 drivers were used. One can clearly see the two potentiometers in the lower left part of the drivers – these are used to set the coil current of the additional stepper motors.

When powering up TSC, the software should now recognize the focuser board – that is, the configuration panel as shown in Fig. 16 is enabled – and two additional small bipolar steppers can be operated using TSC.

14 Glossary

- **AA** A common size for 1.5 V batteries.
- **BT** Bluetooth, a wireless serial communication protocol.
- **CdC** Cartes du Ciel or Sky Chart, a planetarium program.
- **CNC** Computer Numerical Control, an acronym from machinig for computer controlled tools.
- **COM** An old interface descriptor for a serial communication port.
- **CPU** Central Processing Unit, probably the most important part of a computer.
- **CSV** Comma-Separated Varaibles, a text style file format readable by many spreadsheet programs.
- **DSLR** Digital Single Lens Reflecting camera.
- **FTDI** Future Technology Devices International, a company that produces semiconductor devices; mainly known as a USB/RS232 converter synonym.
- **GEM** German equatorial mount.
- **GPIO** General Purpose Input/Output. Basicaly, a set of contacts on the Raspberry Pi to write or read digital commands.
- **GPS** Global Positioning System – an interface to satelites determining your actual location.
- **GUI** Graphical User Interface.
- **HA** Hour angle.
- **HAT** Hardware Attached to Top. A common name for additional hardware stacekd onto a Raspberry Pi.
- **HDMI** High-Definition Multimedia Interface. A video standard used by the Raspberry Pi.
- **I^2C** Inter-Integrated circuit. A simple serial communications protocol for microcontrollers.
- **IDE** Integrated Development Environment. A developement platform for computer programs.
- **INDI** Instrument Neutral Distributed Interface. A protocol for communication between astronomy hardware.
- **JST** Japan Solderless Terminal – a frequent and usually incorrect term for a PCB-mounted plastic jack.
- **Li/Po** Lithium–Polymer. A type of rechargeable battery.

- **LX200** Originally the name of a telescope series by Meade Corp. Here, it is used for a standard protocol for communication with a telescope.
- **MAC address** Media Access Control Address. A unique identifier for controllers in a network.
- **MISO** Master In-Slave Out. A communication line used in SPI.
- **MOSI** Master Out-Slave In. A communication line used in SPI.
- **NEMA** National Electrical Manufacturers Association. In the context of stepper motors, the NEMA size refers to a standardized size of drives so that drives can be exchanged easily.
- **NGC** New General Catalogue. A list of non-stellar objects.
- **NPN** Negative-Positive-Negative. A type of bipolar transistor.
- **OLED** organic Light-Emitting Diode. A type of small displays used in microcontrollers.
- **PCB** Printed Circuit Board.
- **PDF** Portable Document Format.
- **Qt** A class library for developing user interfaces (and more) in C++. Check `www.qt.io` for more information.
- **RJ12** Registered Jack 12 - a standard plug in telecommunications, in this case with 6 contacts.
- **RMS** Root-Mean-Square. The arithmetic mean of squares fo a set of numbers.
- **RS 232** Recommended Standard 232. A serial communications protocol introduced in the 1960s.
- **RTC** Realtime clock.
- **SAO** Smithsonian Astrophysical Observatory.
- **SD** Secure Digital. A non-volatile memory card.
- **SMD** Surface Mounted Device. A standard for miniature electronics components.
- **SPI** Serial Peripheral Interface bus. A standard for serial communication between embedded devices.
- **SSID** Service Set Network Identifiyer - usually the name of a device providing network services.
- **ST 4** A standard introduced by SBIG Corp. It is a protocol to control short telescope movements in autoguiding.
- **TCP/IP** Transmission Control Protocol/Internet Protocol. The communication standard for exchanging data via Ethernet or WLAN.

- **TFT** Thin-film transistor liquid crystal display. Another technolgoy for small displays.
- **TSC** TwinStepperControl – an acronym for the controlling software presented in this manual based ona Raspberry Pi3.
- **USB** Universial Serial Bus. A standard serial communications protocol.
- **VNC** Virtual Network Computing. A system to allow remote access to the GUI of other computers.
- **WLAN** Wireless Local Area Network.