The TSC Telescope controller: Assembly und Usage

Wolfgang Birkfellner and Steffen Elste,
Home repository:
https://github.com/selste/TwoStepperControl
Homepage: tscatm.wordpress.com
Contact: wbirkfellner@gmail.com

July 27, 2018

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 350 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 350 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 currently based on two industrial grade stepper controllers (Phidget 1067 B bipolar stepper controller, www.phidgets.com, see also Fig. 3) with a maximum coil current of 4A and 30V maximum supply; the resolution is fixed with 1/16 microsteps. The boards are controlled via USB 2. The controllers are connected to a Raspberry Pi 3 B or a Raspberry Pi 3 B+ running Raspian Stretch (https://www.raspberrypi.org/). Fig. 2 shows the TSC controller in a improvised housing, together with one of the two wireless handboxes available.

The programming environment used for development of TSC is Qt[2] 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.
- 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.
- support for telescope control and GoTo via the LX200 protocol and common planetarium programs such as Cartes du Ciel, Stellarium, SkySafari and KStars. Connection is provided via autonomous WLAN (which is established if no other access point is available), Ethernet and USB. Currently, TSC operates with permanent coil currents of 1.8A at 12V with my telescope, resulting in a GoTo speed of 110× sidereal speed. The top speed is dependent on the drives and gears used. On smaller drives, top speeds of 500× sidereal speed were realized. LX200 is also supported in the ASCOM 6.3 environment, therefore all programs utilizing ASCOM 6.3 such as SkyTechX can be used.

[1] Check out `www.stellarjourney.com` or `rduinoscope.byethost24.com`.
[2] `www.qt.io/`

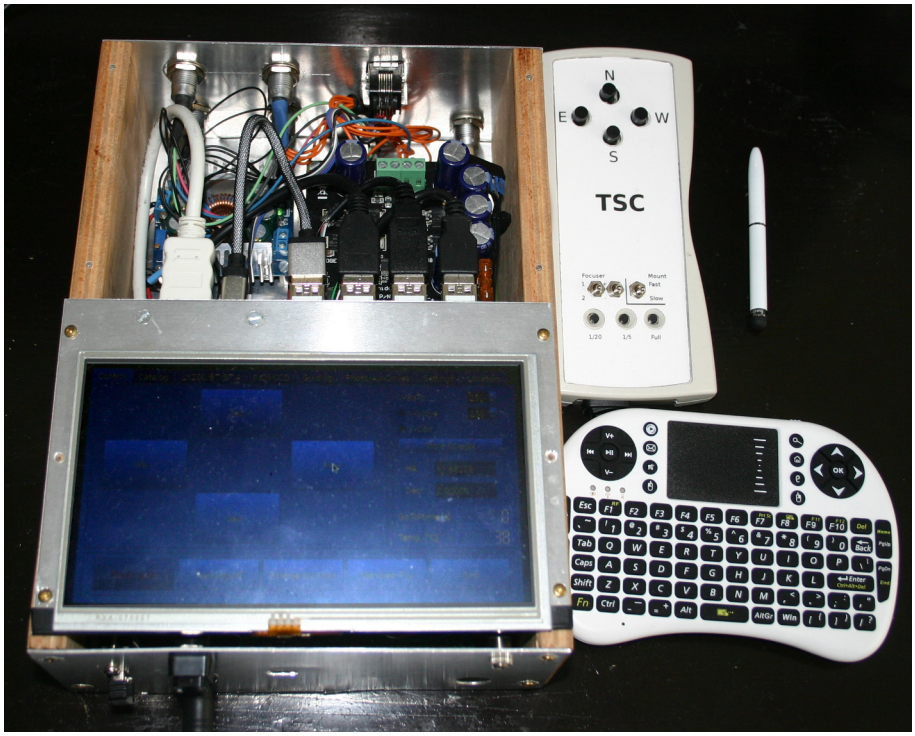


Figure 2: TSC in a version with a 7" touchscreen display, two high power stepper drivers, a secondary motor driver board, an internal 12V-5V block converter, and one of the two wireless handbox designs available. A miniature keyboard as shown is needed for configuration but is not required in routine 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. The housing is a preliminary one used during development.

- support for a custom wireless Bluetooth or 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[3].
- 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.

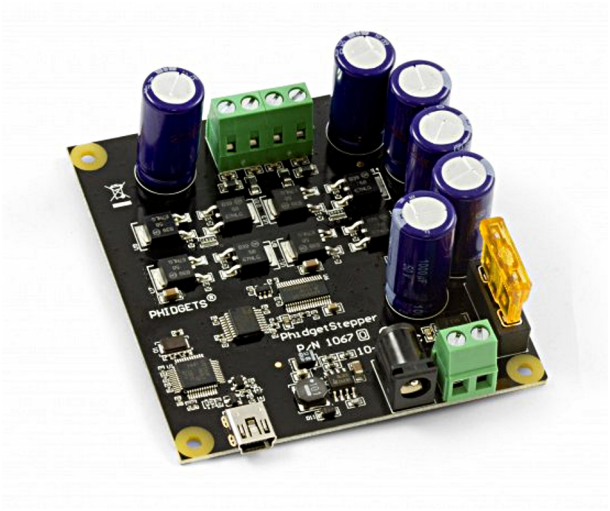


Figure 3: The Phidget 1067 bipolar stepper controller. These drivers for 2-phase bipolar stepper motors can deliver coil currents of up to 4A with a voltage of up to 30V, making the suitable for driving even very large telescopes via friction or worm wheels. Image taken from www.phidgets.com.

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.

[3] `www.indilib.org`

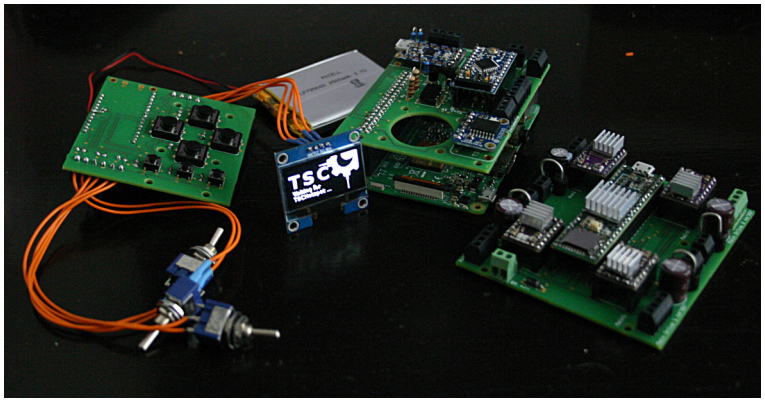


Figure 4: Some of the PCBs that make up TSC, including a Raspberry Pi 3 Model B, which forms the central component of the controller.

The overall cost, probably not including all small materials like screws, and wires is approximately 135 € for the Raspberry Pi 3 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 Phidget stepper drivers are the single most expensive components with 200 € for both. A potentially cheaper alternative for smaller mounts is presented as a future perspective in Sect. 7.1.

A HDMI touchscreen with 800 × 480 resolution costs between 35 and 80 €. It is not compulsory but recommended. The wireless handbox accounts for approximately 50 €. If you want to use the focuser board, too, this is another 30 € 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.

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 10 months by now. This section introduces the main components:

3.1 The main screen

Figure 5: The main screen of TSC. Basic functionality is motion in four directions, adjustment of correction and GoTo speeds, current position as hour angle and declination, and buttons for starting and stopping tracking, stopping all drive motion immediately, manually carrying out a meridian flip and for terminating the program. In GoTo-mode during a slew, the estimated time of arrival is also displayed.

Fig. 5 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-mov**), which can be freely adjusted in multiples of the basic speed **V-corr**). The maximum speed is **V-GoTo**, which can be adjusted but is available only in

GoTo mode, not via handbox operation. The pushbutton **Store Speeds** writes these settings to the preferences file.

The buttons on the lower part of the dialog allow for manually starting and stopping compensation motion in right ascension, doing a manual meridian flip, emergency stopping and for exiting TSC.

On the left hand side, one finds also displays for the actual position given as 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

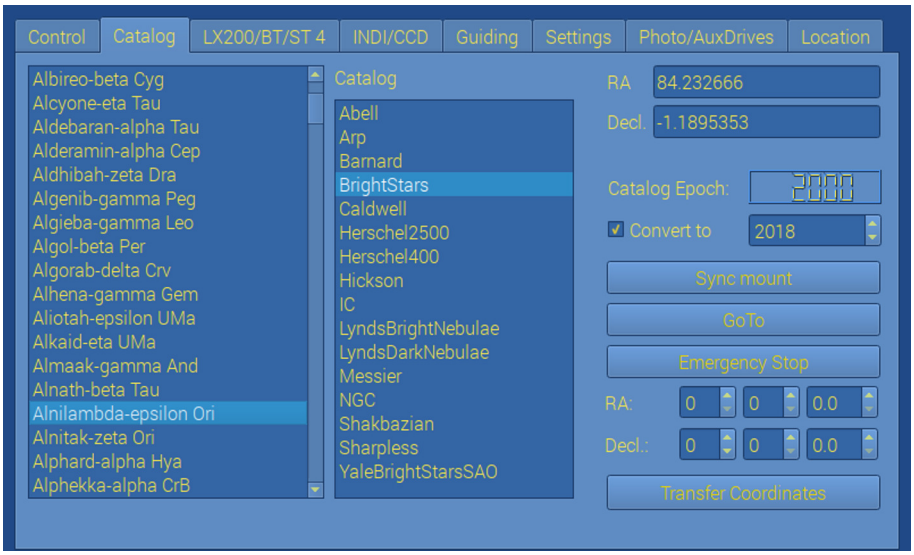


Figure 6: The screen for choosing objects, locations, and for one-star alignment and GoTo. It features an interface to manually editable catalog files. These positions, which are automatically converted to the current date, can be used for syncing the mount and for GoTo. It is also possible to enter an arbitrary location in right ascension and declination.

Fig. 6 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.
- If the slew in one of the directions is shorter than five seconds, the GoTo motion is not carried out simultaneously. Rather than that, the short slew is carried out first, followed by the longer slew in the other direction.
- 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. 5).

3.3 The interface dialogue

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 a patch cable[4] or a small router using the Ethernet port of the Raspberry Pi, and via WLAN. Denote that the Raspberry opens an WLAN access point of its own if no router is within reach, therefore this functionality is available in remote places without WLAN coverage. Sect. 9.3 gives more details on this functionality.

[4]This feature was, however, not tested extensively so far.

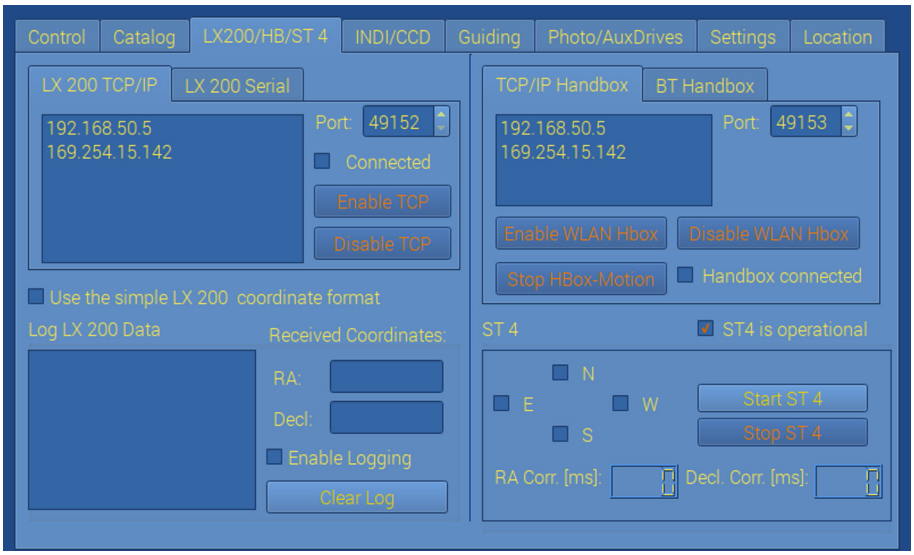


Figure 7: The configuration dialogue of TSC; LX200 via USB or WLAN and Ethernet, the connection to the custom Bluetooth- or TCP/IP handbox and for ST4 is configured here.

- Bluetooth or TCP/IP for connecting a custom 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].
- INDI (www.indilib.org) is used to interface to guiding cameras; the configuration of INDI is described separately in Sect. 3.4.

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 (with a small limitation), KStars, various programs supporting ASCOM 6.3 and SkySafari on 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. 7 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 common 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.

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

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. 8. After pushing this button, LX200 is available via USB.

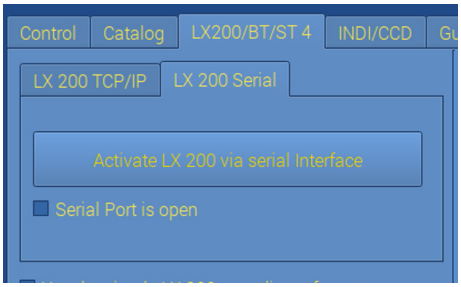


Figure 8: The dialogue to start LX200 interaction via the USB-to-RS232 converter of TSC.

Plans for two handboxes are provided with TSC – a more sophisticated one featuring a small OLED display using the internal WLAN hotspot of TSC, or a second handbox using Bluetooth. Both are configured using either the **TCP/IP Handbox** or the **BT 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 the Bluetooth-handbox is similar (Fig. 9). If the MAC-address is of the Bluetooth module is once determined and stored (using the **Save BT MAC Address button**), all one needs to do is to push the **Connect BT** button *while the handbox is powered up*. If connection is not established (indicated by the **BT port up** checkbox, powering up the handbox and pressing the **Try BT restart** button after a few seconds might work. Bluetooth is a serial protocol and somewhat frail, admittedly. Both handboxes are described in detail in Sect. 12.

The most simple interface is the ST4 interface. ST4 commands are received

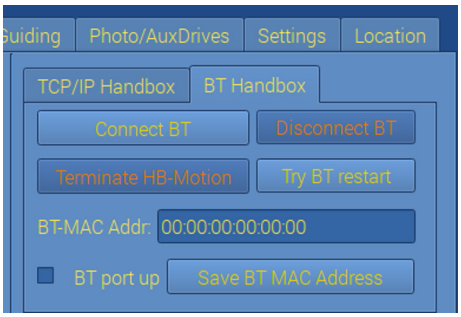


Figure 9: The dialogue to configure operation of the Bluetooth handbox.

by a dedicated Arduino Mini Pro microcontroller on the HAT (see also Sect. 10.4) 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.

3.4 INDI server configuration

Figure 10: Dialogue for starting an INDI server and for connecting a guidecamera.

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. **??** 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 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 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** dialgue (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 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.

3.5 Guiding camera operation

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

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

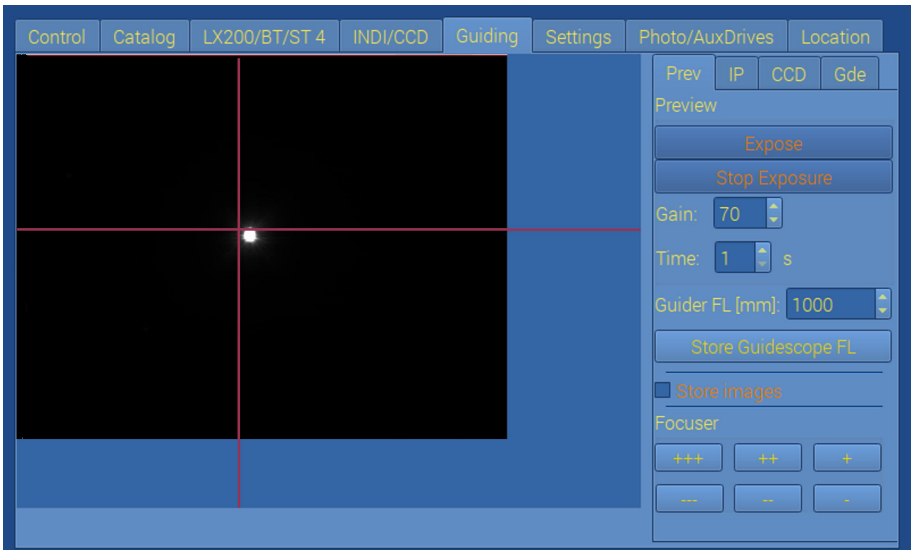


Figure 11: The main dialog for operating TSC as a standalone autoguider. If a camera is connected via INDI, exposure can be started; camera gain is adjusted if this feature is available for your camera, and the exposure time is also set. By clicking into the image, a crosshair can be positioned on the 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.

- **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

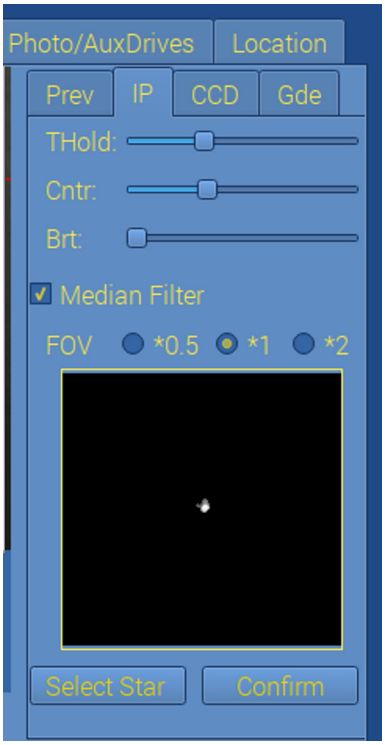


Figure 12: The **IP** sub-dialog for image processing of a guide star selected in the **Guiding** main screen. The guide star is pre-selected by clicking on the camera image. A final confirmation is done by pressing the **Select Star** pushbutton. Brightness threshold **THold**, image contrast **Cntr** 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 star (this is the **THold** slider), and basic adjustment for contrast (**Cntr**) and brightness (**Brt**) of the image. A 3×3 median filter removing salt-and-pepper noise (that is hot or cold pixels on the guiding camera) is activated by default. 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

Fig. 13 shows the **CCD** calibration dialogue. For testing purposes, the mount can be moved for a given number of milliseconds that is defined using the **Pulse [ms]** spinbox. using the **Dec+**, **Dec-**, **RA+** and **RA-** buttons. 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. **Emergency Stop** allows to terminate the procedure. 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

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

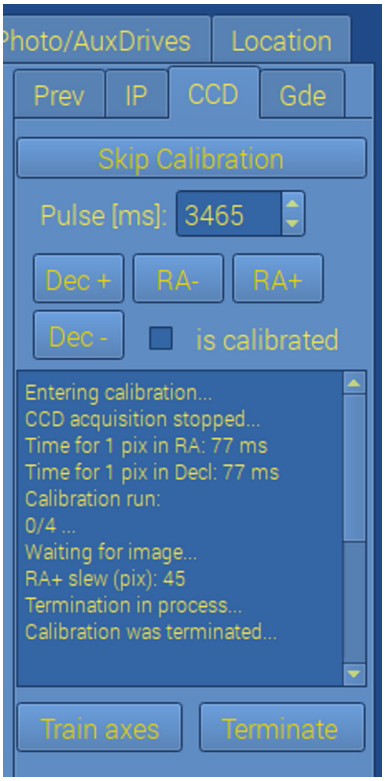


Figure 13: The autoguider calibration dialog labelled **CCD**. Motion of the mount for a given length im milliseconds (to be adjusted in the **Pulse [ms]** spinbox) can be tested using the buttons below. The **Train the axes** button triggers calibration of the mount, which runs automatically and is logged in the text field. This step can be skipped for testing purposes, and the procedure can be terminated prematurely.

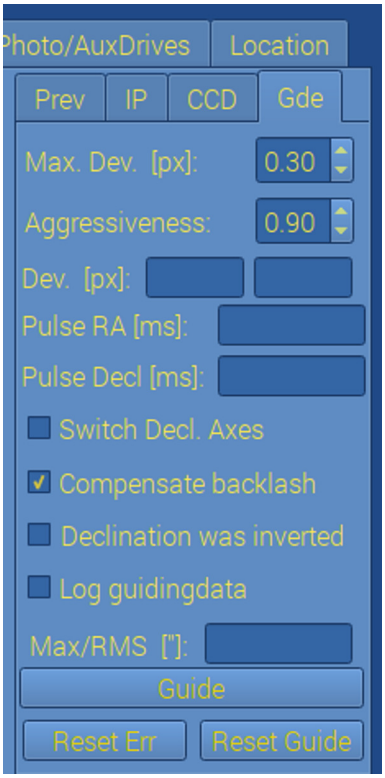


Figure 14: The dialog that starts the actual guiding procedure. A maximum permissible error in guide camera pixels can be set. By pressing the **Guide** button, autoguiding is started. Current guiding errors, correction move durations and the maximum error during guiding are displayed. If the declination 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.

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.

3.6 DSLR operation and focuser motor setup

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).

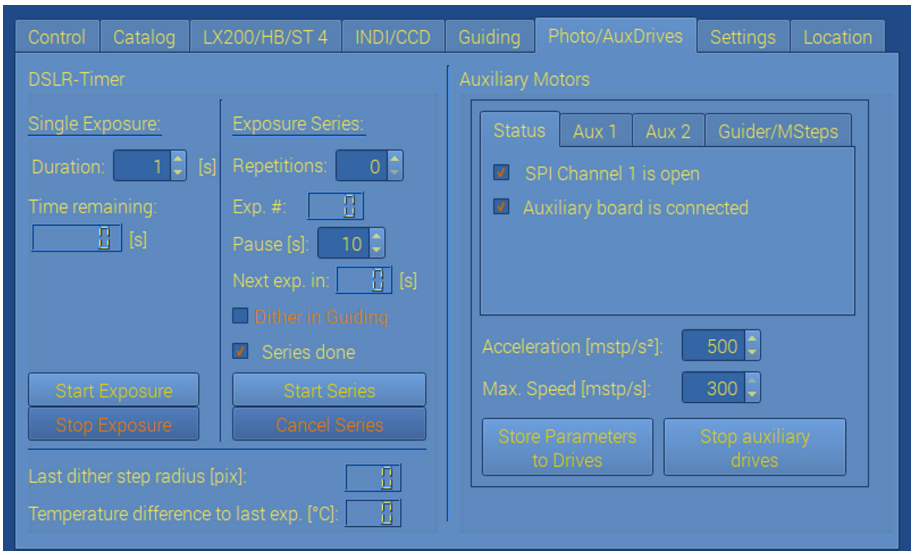


Figure 16: The dialog for controlling the DSLR in single exposures and exposure series, and for setting up the optional focuser drives.

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 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.

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

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

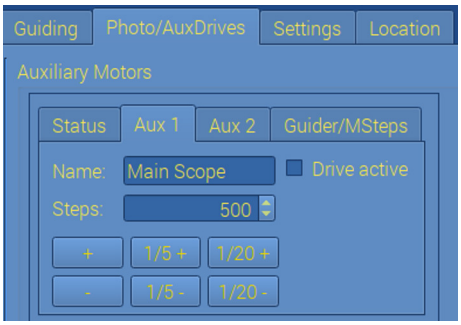


Figure 17: One of the two tabs in the **Auxiliary drives** group that allows for configuring the additional steppers fo the focusboard. It is possible to give a name to the drive, and to define the basic travel in the **Steps** field. The full travel, $\frac{1}{5}$ or $\frac{1}{20}$ can be tested in both directions. As long as the drive is active, this is indicated by the **Drive active** checkbox.

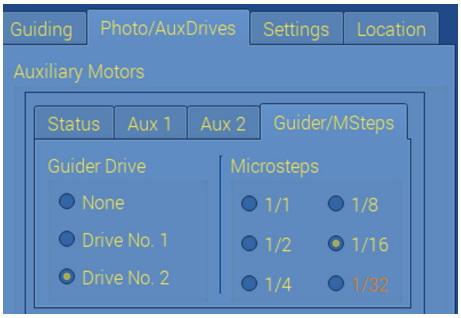


Figure 18: The contents of the **Guider/MSteps** tab.

is also implemented, the **Stop auxiliary drives** button.

3.7 The Settings dialog

Figure 19: 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. In addition, certain DSLR parameters for dithering in imaging series is also possible.

Fig. 19 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 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°. The number of microsteps is fixed to 16 for the Phidget 1067 driver, therefore this value cannot be changed. Denote that changing the parameters is not feasible when the drive is running. Storage of parameters is triggered when the **Store gear settings** button is pushed.

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. 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.

Finally, the group **DSLR Settings** allows to configure a few buttons for dithering. Dithering is a technique to shift the telescope for a few arcseconds in between subsequent exposures. In order to limit the dither range (that is the random displacemnent of the telescope during exposure), the diagonal pixel size of the DSLR in microns has to be set using the **Pixel diag. size [mu]** spinbox. The focal length of the main telescope is set in the **Telescope FL [mm]** spinbox. The minimum and maximum travel for dithering, which is determined using a random number generator is set using the **Dither range min [pix]** and **Dither range max [pix]** spinboxes. The button **Store DSLR settings** saves those settings.

3.8 The Location dialog

This is probably the most simple of all setup dialogs. Here, the current date, time, Julian date and sidereal time is 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 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 ascesnion from the clock and the the stored position and if you scope was not moved in between, the mount should be synced again.

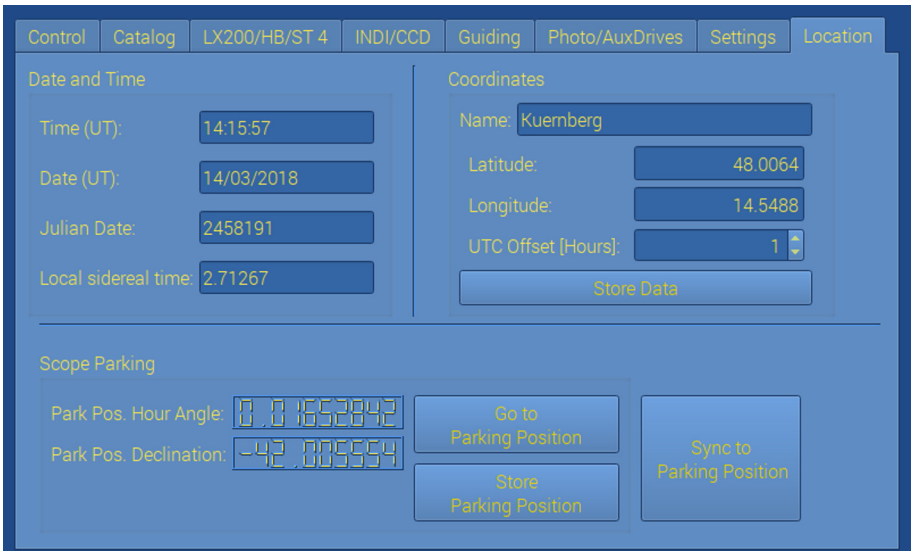


Figure 20: The location dialog. Current readings from TSCs internal hardware clock are displayed, and the observation site coordinates are defined. If the mount is synchronized, it is also possible to define a parking position and to make the telescope return to that parking position.

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. 22).
- 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. 21 shows this on a ASUS ZenPad running

Android.

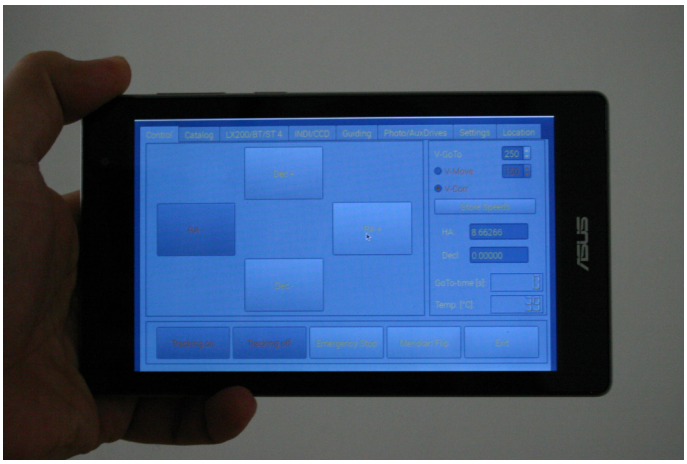


Figure 21: Remote control of TSC using VNC and an Android-based tablet.

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 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 aforementioned Adafruit USBFriend (see also Fig. 45) 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. 8. 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

[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`

tablets can connect to it. The default name for this autonomous WLAN access point is `TSCHotspot`. Fig 22 shows the looks of accessing TSC.

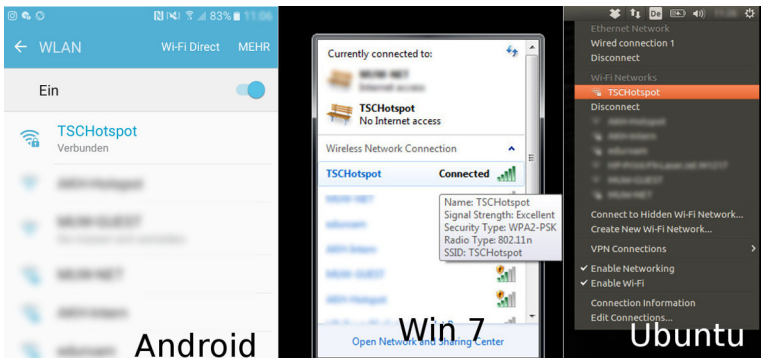


Figure 22: Various views of the autonomous `TSCHotspot` WLAN access point spawn by TSC.

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 the address starting with `169` is usually the address of 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. 23 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.

5.2 SkyChart/Cartes du Ciel, KStars and Stellarium

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. 24 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. 25). After this, CdC controls your telescope, either via a virtual handbox or via the *Slew* command (see also Fig. 26). CdC was

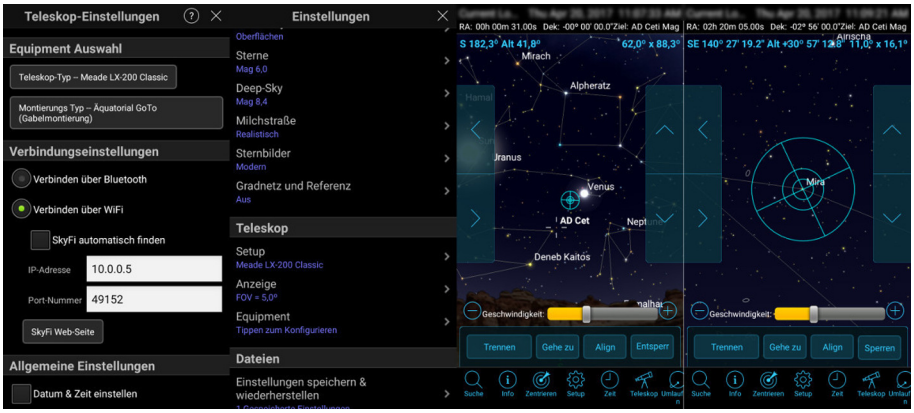


Figure 23: Connecting SkySafari Pro via WLAN to TSC allows for controlling the tescope. The actual position of the telescope is also updated on SkySafari.

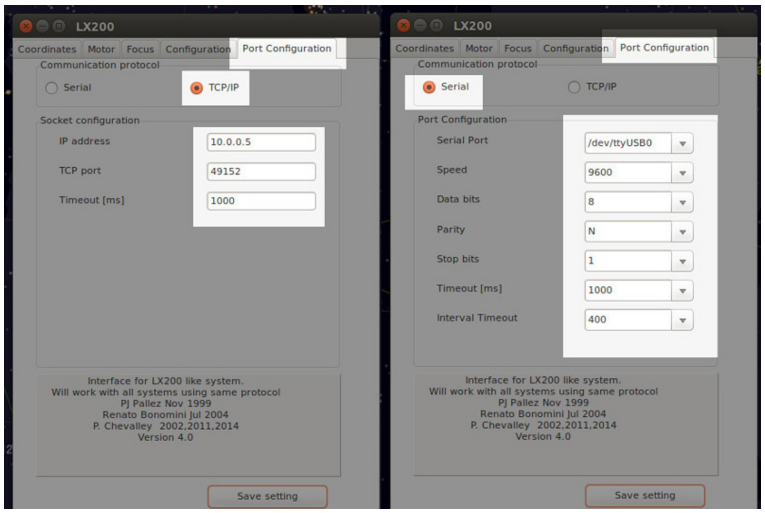


Figure 24: Dialogues in SkyChart/Cartes duCiel for setting up TCP/IP or serial communication for LX200 control.

tested with Windows 7, MacOS and Linux and works perfectly fine with these parameters

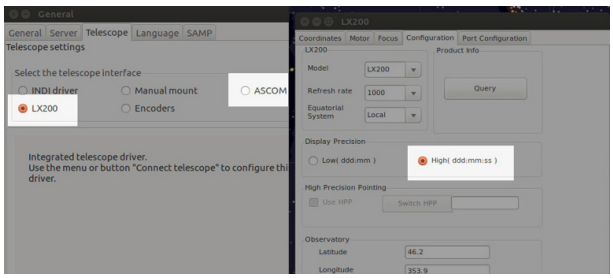


Figure 25: The further configuration steps necessary for connecting TSC to SkyChart – take care that the *Display Precision* is set to **high**.

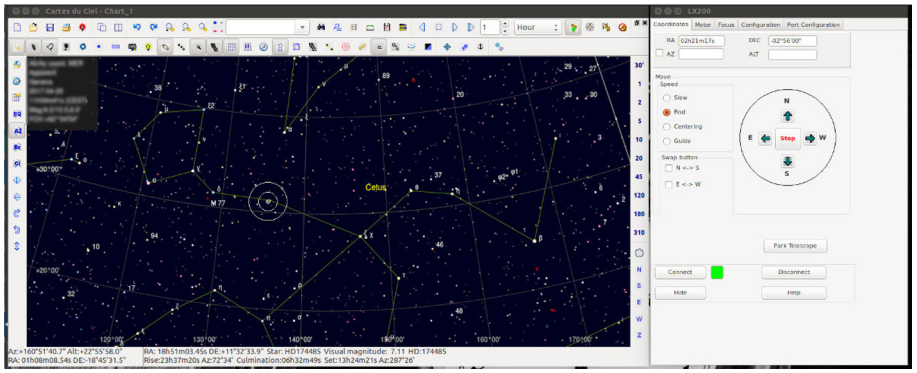


Figure 26: Connection of TSC and CdC. The telescope can be controlled via a virtual handbox or the *Slew* command of CdC; the actual position of the telescope is displayed on the screen with a crosshair.

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. 27 shows two screenshots of this process. After connection, KStars allows for control of telescope position using TSC, just like the other programs. Fig. 28 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. 29.

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:

[12] `https://pixelstelescopes.wordpress.com/advanced-lx200/`

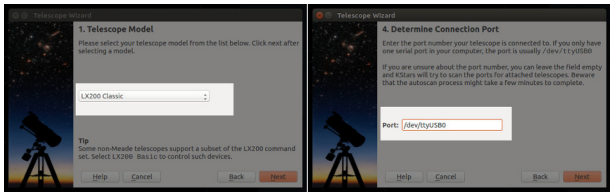


Figure 27: Screenshots of KStars set up using the serial connection under Linux.

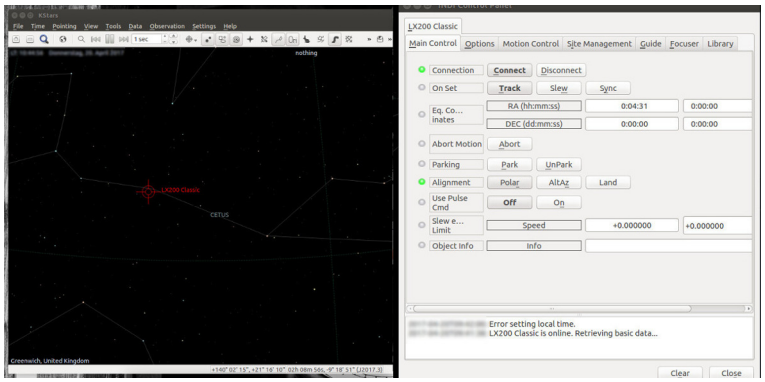


Figure 28: Screenshot of TSC connected to KStars using the *Meade LX200 Classic* protocol.

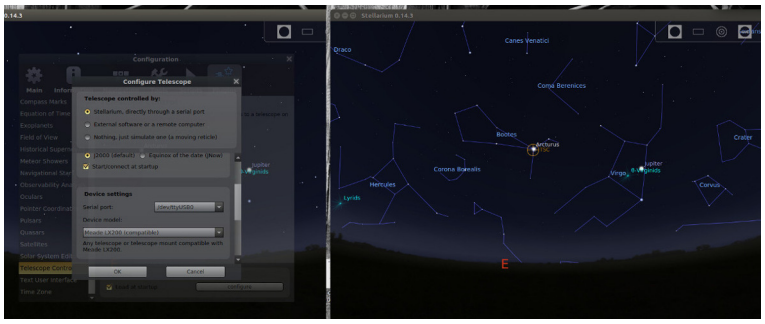


Figure 29: Configuration and display of telescope positions using serial communications in Stellarium.

- 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. 30 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.

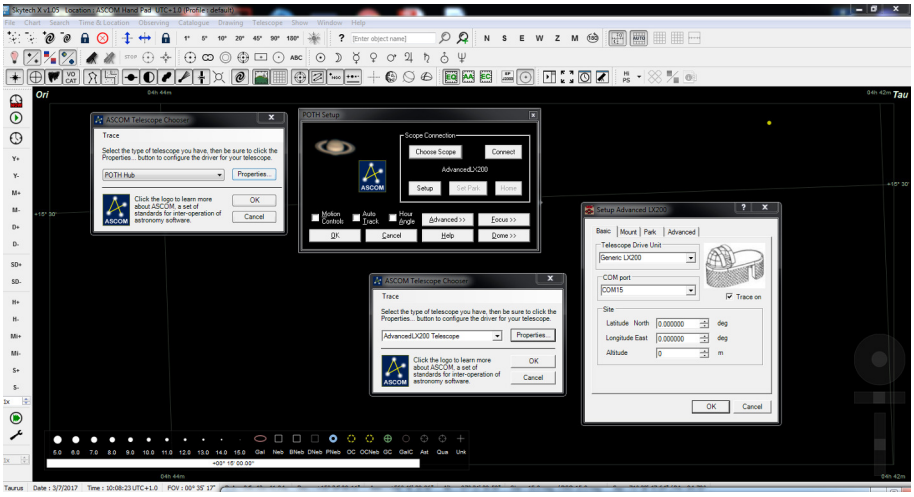


Figure 30: Setting up ASCOM 6.3 to work with the Advanced LX200 interface. The **Advanced LX200 Telescope** driver has to be selected via the **POTH** interface of ASCOM.

[13] `http://skytechx.eu/`
[14] `http://www.astrosurf.com/c2a/english/`
[15] `http://www.hnsky.org/software.htm`

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:

- Phidget 1067 board serial number for the right ascension drive.
- Phidget 1067 board serial number for the declination drive.
- The gear ratio for planetary connected to RA-stepper.
- 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 can do. This is 16 for the Phidget 1067 board.
- 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.

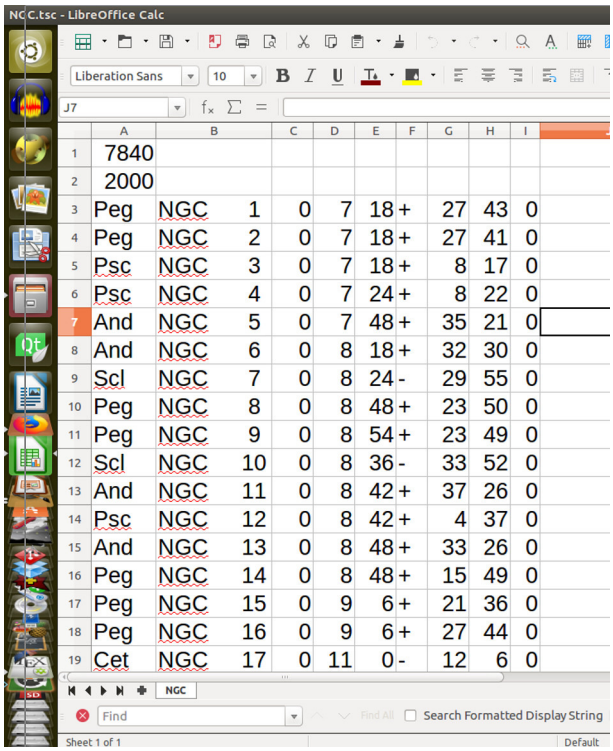
6.2 Catalog files

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. 31 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 31: 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

As pointed out initially, TSC was developed for large heavy stationary telescopes; there is also a wide range of cost-efficient DIY solutions for small telescopes available. However, downscaling is in the queue. In fact, the only difference lies in the use of different driver boards, which of course also brings down the cost as one of the Phidget boards is approximately 100€.

If you check the directory `Hardware/USB_TSC_TeensySlave`, you will find a PCB and software for controlling four small independent stepper drivers with the well-known Polulu style pinout. Various drivers with microstepping ratios from $\frac{1}{16}$ to

$\frac{1}{256}$ can be used, but coil current provided over a longer period of time by these drivers is inferior to drivers like the Phidget 1067. Still, for smaller telescopes with drives not larger than NEMA 17 this might become an alternative. The microcontroller used in this case is an extremely powerful Teensy 3.6[17] that outperforms standard systems like the Arduino Uno by far and also allows for fast direct USB 2.0 communication. Fig. 32 shows a prototype of the driver board.

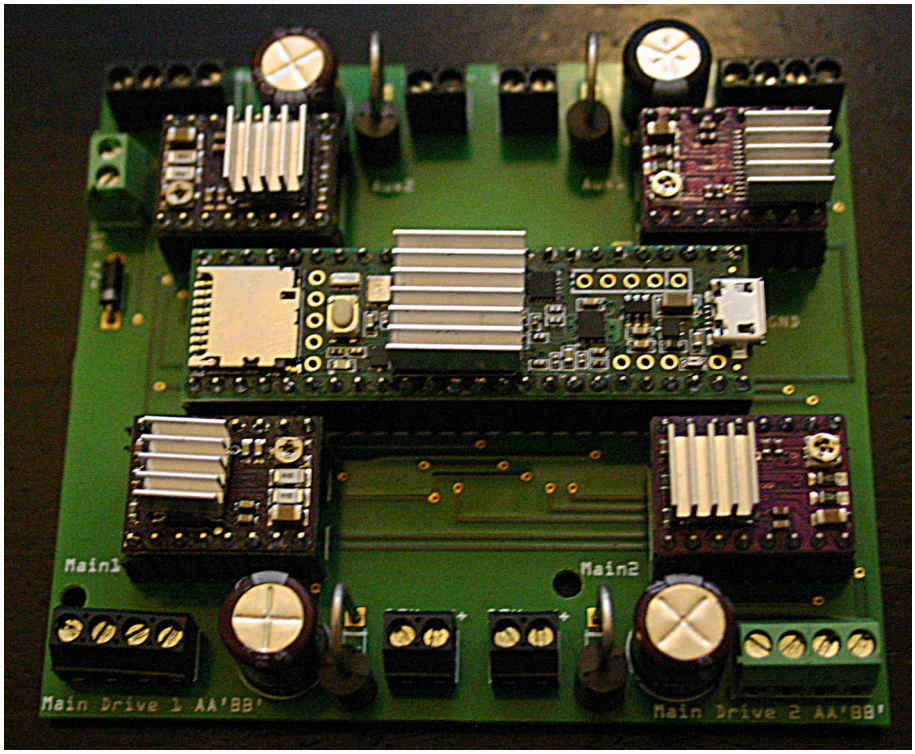


Figure 32: A protoype of a smaller and cheaper stepper driver board to be controlled by TSC. Both the main and the focuser drivers are assembled on one PCB, and all are driven by a single Teensy 3.6 microcontroller. The maximum coil current available here is way below the power that can be provided by the Phidget drivers but is sufficient to drive steppers up to the NEMA 17 class.

Another important subject for mobile use is precise time signals and exact location for quick and unambigious n-star alignment; for this purpose, a GPS receiver is also in the queue of hardware to be supported. As the current version of TSC is aimed at stationary systems and the cost of a GPS module is considerable, this option was omitted so far in favour of the hardware clock solution.

[17] `https://www.pjrc.com/store/teensy36.html`

7.2 Three star alignment and Alt/Az mode

This is certainly an interesting option, not only for Dobsonian telescopes but also for large stationary telescopes; one of the auxiliary drives could also be used for field derotation in that case.

7.3 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. 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.

7.4 Control via a dedicated app for tablets and smart phones

A shown in Sect. 4 there is a possibility to interface to TSC using VNC; a more comfortable solution is certainly doable. However, a handbox with physical switches is in my opinion the superior interface device in darkness compared to a touchscreen.

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[18] 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 the authors.

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

- 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[19].
- a fast 16 GB SD card, which costs approximately 10 €.
- an image of Raspian Stretch 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

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

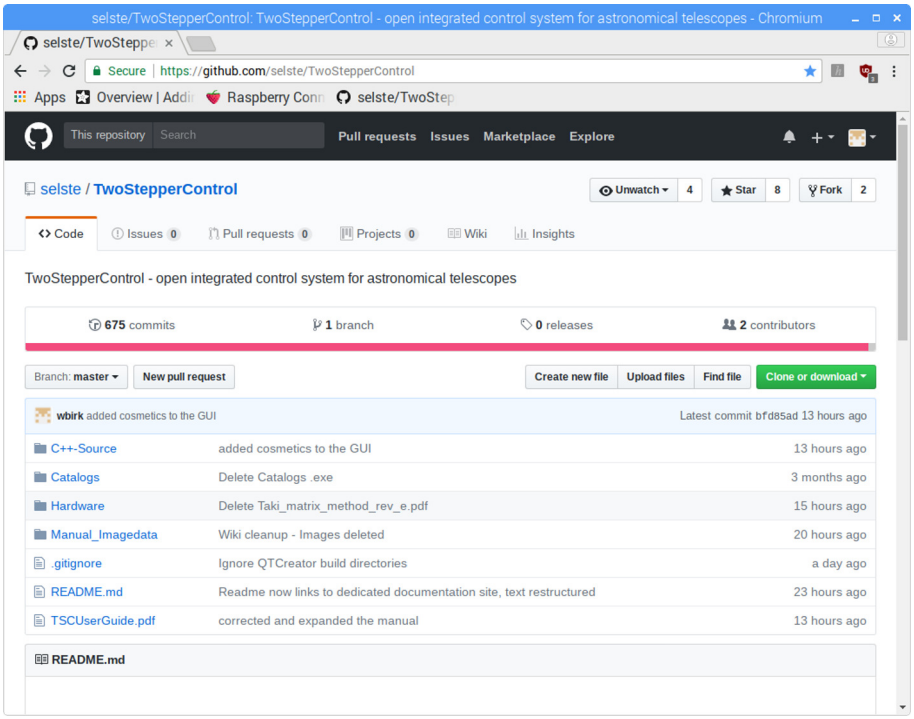


Figure 33: Screenshot of the home site for TSC on github. In the directory `C++-Source`, the sourcecode for TSC is found. The project can be directly compiled using Qt-Creator, installed on the Raspian image; however, a compiled version is also provided alongside with an an SD-card image of the Raspberry Pi operating system. `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.

SD-card image named `TSC.img` is located and type
`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.

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

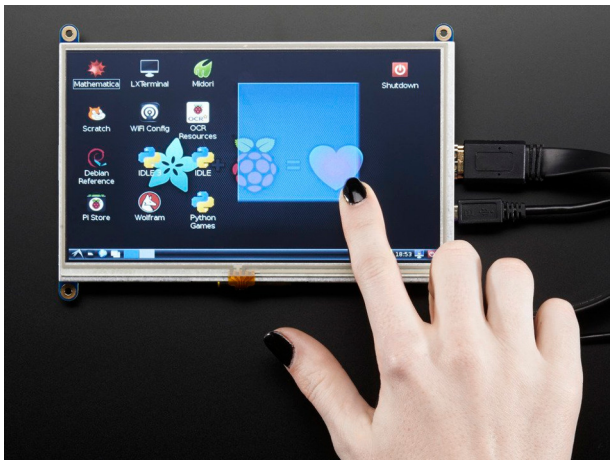


Figure 34: The HDMI touchscreen display available from Adafruit.com, which can be used in connection with TSC. Photo taken from www.adafruit.com.

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. 34 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. 35). 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.

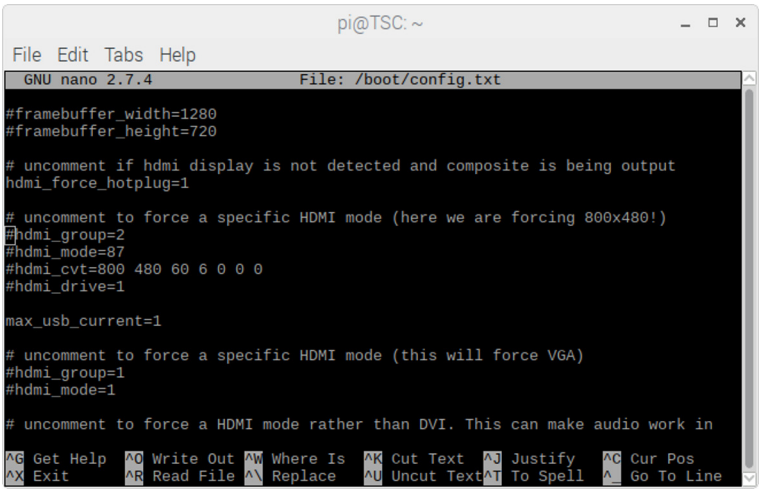


Figure 35: Configuration of the screen resolution of the Raspberry using VNC.

9.3 Setting up autonomous WLAN

TSC is set up following the recommendations from `www.raspberryconnect.com`[20]. If no known access point is available, it opens up its own WLAN with the aforementioned access point`TSCHotspot`. This is indicated by two opposing arrows in the menu bar – see also Fig. 36.

Figure 36: If the autonomous WLAN with the access point `TSCHotspot` is opened, this is indicated by two opposing arrows.

9.3.1 Adding your WLAN

In order to connect to your home WLAN network if you are in reach, one more configuration step is necessary. Again, you have to open a terminal and type `pi@TSC: $ sudo nano /usr/bin/autohotspotN`.

[20] `www.raspberryconnect.com/network/item/330-raspberry-pi-auto-wifi-hotspot-switch-internet`

This is the script that manages the search for known SSIDs. Scroll to the line `ssids=('...')` as shown in Fig. 37 and insert the name of your access point. In Fig. 37, the known access points are `TP-Link_Extender_2.4GHz` and `A1-156481`. Save the file by typing `Ctrl-O` and exit by typing `Ctrl-X`. After a reboot, the Raspberry connects to your WLAN if within reach and opens the `TSCHotspot` otherwise. Fig. 37 shows the editor.

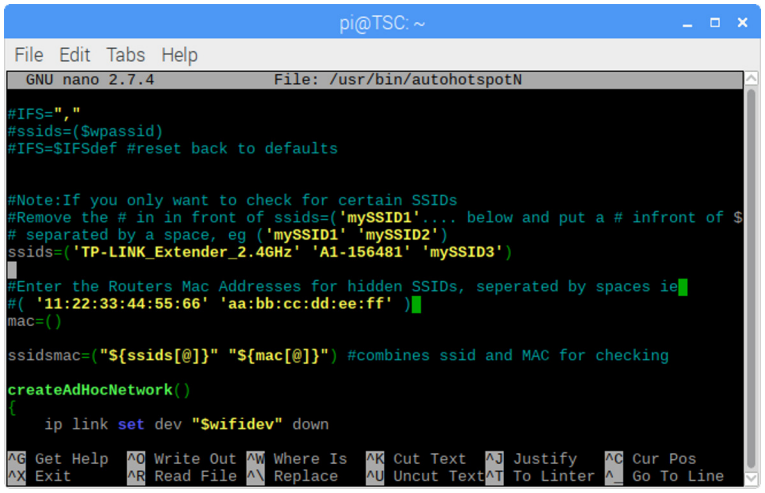


Figure 37: Configuration of your local WLAN accesspoint on the Raspberry. The name of your WLAN router is needed in the line `ssids=....`

If TSC is connect to an external access point, the WLAN-fan icon replaces the opposing arrows from Fig. 36; this can be seen if Fig. 38.

Figure 38: Appeareance of the toolbar if TSC is connected to a known access point.

9.3.2 Troubleshooting

Figure 39: Appeareance of the toolbar if WLAN is off.

There is a little caveat in this procedure: If your router is to slow to issue an IP-address, it might happen that TSC connects but it does not route properly. That is, you see a situation like in Fig. 38, but no internet access is feasible. In that case, right-click on the fan icon and choose `Turn Off Wi-Fi`. The icon

changes to the situation as shown in Fig. 39. Click this one again and choose your SSID – that should establish access.

9.3.3 Turning the Hotspot off again

So once access to the internet via your router is secured, you can do two things:

- Update your copy of TSC from github.
- Synchronize the clock module on the TSC-HAT via a timeserver (happens automatically).
- Make updates to Raspian, for instance by updating VNC once in a while.

Otherwise, you may want to turn WLAN access off and utilize the hotspot, maybe because you want to use the TCP/IP handbox (Sect. 12.1) or you don't want to change the IP-address in the planetarium program all the time. In this case, repeat the steps as shown in Sect. 9.3.1 and put a `#` in front of `ssids=('...')`. After reboot, you are in your own little WLAN-world again.

9.4 Modifications to Raspian Stretch

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 `http://www.raspberryconnect.com`[21].
- The latest INDI-version 1.6 was installed[22].
- Qt 5.7 and the OpenCV developer version were installed from Raspian repositories.
- The Arduino IDE V. 1.8.5 was installed from `www.arduino.cc`.
- 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.

[21] http://www.raspberryconnect.com/network/item/330-raspberry-pi-auto-wifi-hotspot-switch-internet

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

In a next step, the directory `TwoStepperControl-master` is being extracted by clicking on the zip-Archive. Fig. 40 illustrates this.

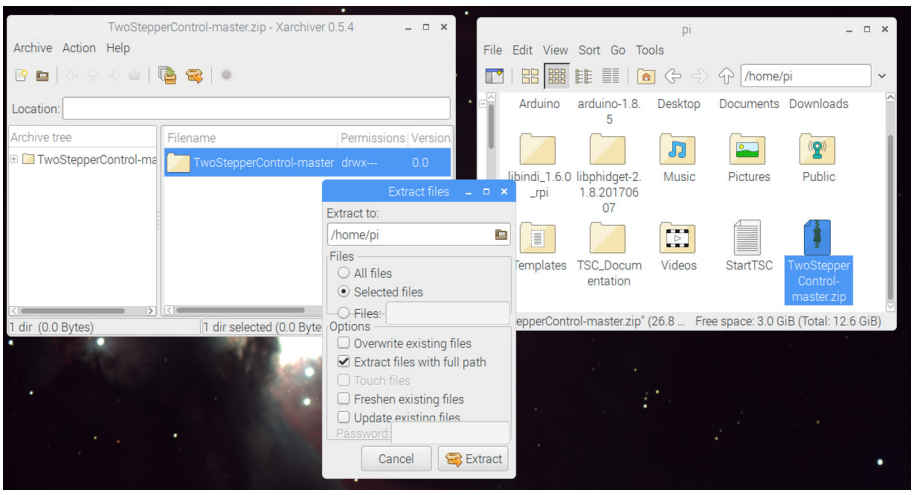


Figure 40: Extraction of a new copy of the TSC-repository. The result is a new `TwoStepperControl-master` archive with up to date materials.

A new version can be compiled by clicking `TwoStepperControl.pro` in the folder `C++-Source`. This opens the QT-Designer IDE; the menu point
`Build` → `Build Project "TwoStepperControl"`
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.

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 and each command that triggers a motion of the stepper motors will lead to unpredicatable behaviour. 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 phidget drivers to the power supply.

10 Basic setup: HAT assembly and power supply

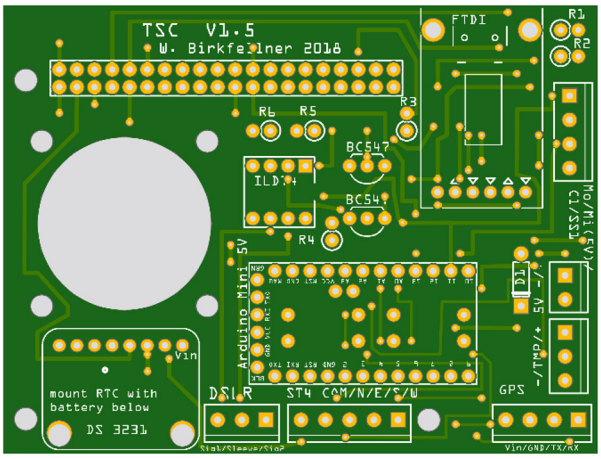


Figure 41: Top view of the PCB, where all components aside from the 40 pin connector to the Raspberry Pi are mounted. The PCB can be manufactured from the Fritzing manufacturing service, which is linked in the Fritzing software.

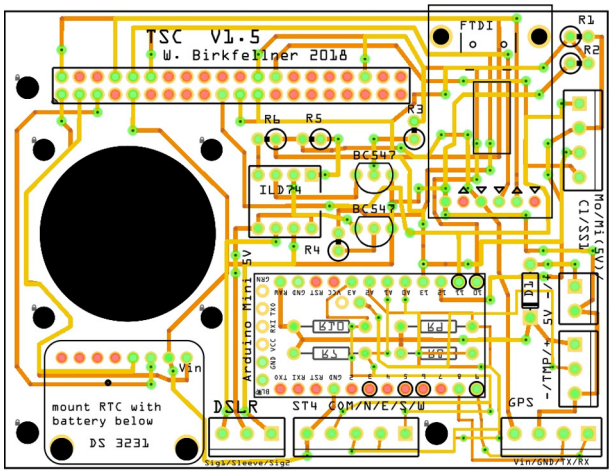


Figure 42: PCB Layout of the HAT. Visible is the large opening for the fan and the location of resitors, diodes and components.

The next step is to build the basic additional hardware of TSC and to connect the Phidget 1067 bipolar 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 `TSC_PiHAT`. Here, a Fritzing file (currently `TSC_PiHat_ILD_V1_5.fzz`

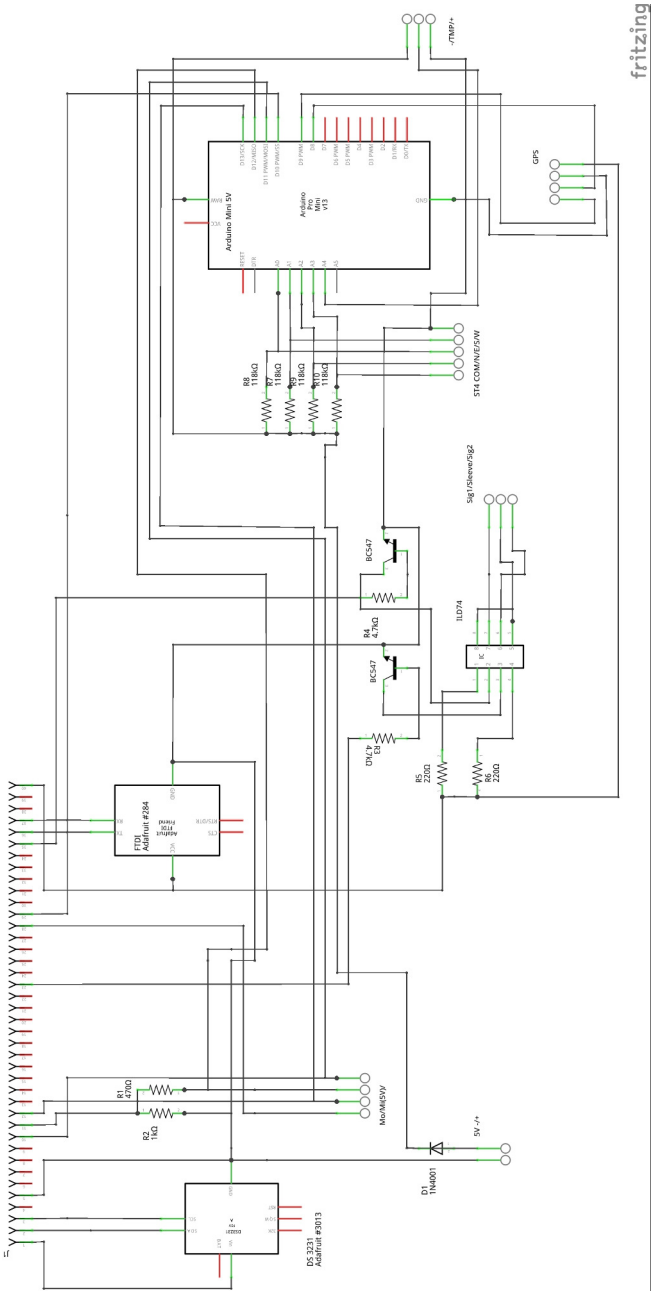


Figure 43: The schematic of the TSC HAT mounted on the Raspberry Pi.

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. 44 shows this component which is about 15 €. A CR1210 battery is also needed.
- 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. 45 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 and ST4 signals.
- A 50 × 50 mm fan for cooling the Raspberry Pi CPU. TSC is threaded; in GoTo mode, it may occupy up to 3 physical cores of the Raspberry. In narrow cases, this has caused heat problems.

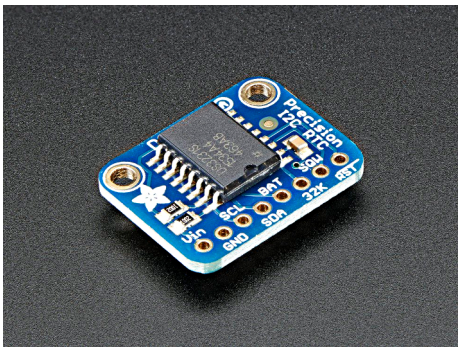


Figure 44: The DS 3231 hardware clock from Adafruit. Photo taken from www.adafruit.com.

Denote that there is also an earlier version, V. 1.4, available on GitHub. The only difference in V. 1.5 is an additional connection for an optional GPS receiver connected to the Arduini Mini Pro. This feature is, as of now, however not supported.

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

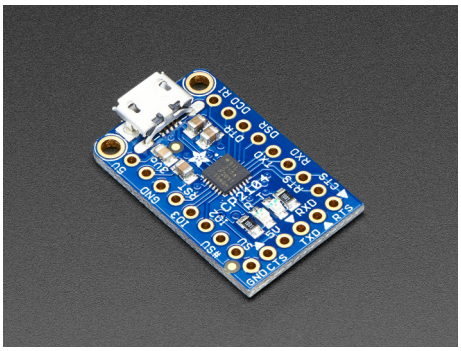


Figure 45: The Adafruit USB-friend mounted to the HAT. It connects an extrnal computer via USB to the RS232 serial interface of the Raspberry Pi and allows for LX200 operation from external programs such as Cartes du Ciel. Photograph from www.adafruit.com.

repository as an Excel-file called `TSC_PartsList.xls`. Here, the resistor values are also noted. Fig. 43 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 to be connected to a socket as some resistors are to be placed below the Arduino. All components aside from the connector to the Raspberry Pi are top mounted. Figs. 41 and 42 show the PCB. Making three PCBs using the manufacturing service of Fritzing accounts for approximately 40 €.

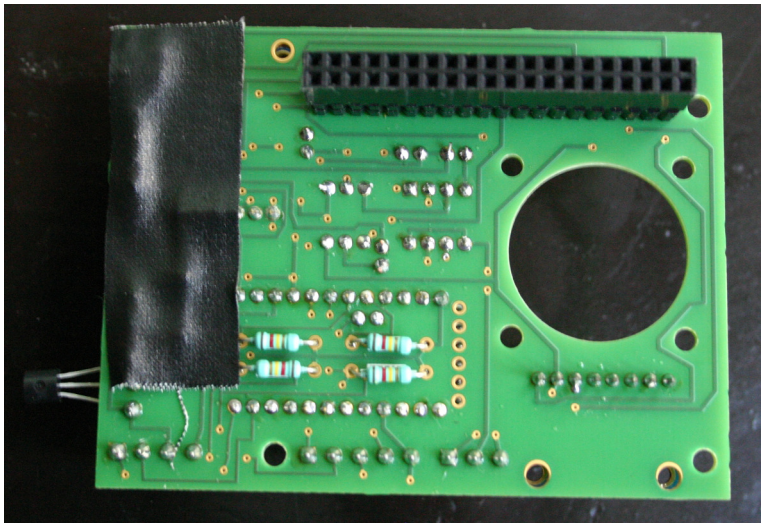


Figure 46: The bottom ot the PCB. Only the 40 pin connector for the Raspberry Pi 3 and four resistors are placed placed here. In order to avoid short circuiting the power supply for the Arduino Mini Pro, a strip of insulating tape was put here as well.

Soldering should start with the 40 pin connector to the Raspberry (see also Fig. 46). *Take care that these are of proper height - the total height of the HAT over the Raspberry Pi mainboard should be no less than 20 mm, otherwise the HAT will short circuit on the Ethernet/USB jacks of the Pi.* Next are the resistors. Denote that aside from R7 to R10, all resistors are mounted in a standup fashion. The following table lists 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.

The sockets for the Arduino Mini follow. 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 sockets as seen in Fig. 47). This is somewhat delicate. As pinstrips need to be soldered to the Arduino as well, it is advisable to do this first and to solder the small two pin connector with the Arduino in place. *Do not solder the Arduino directly onto the PCB as four resistors are placed below.*

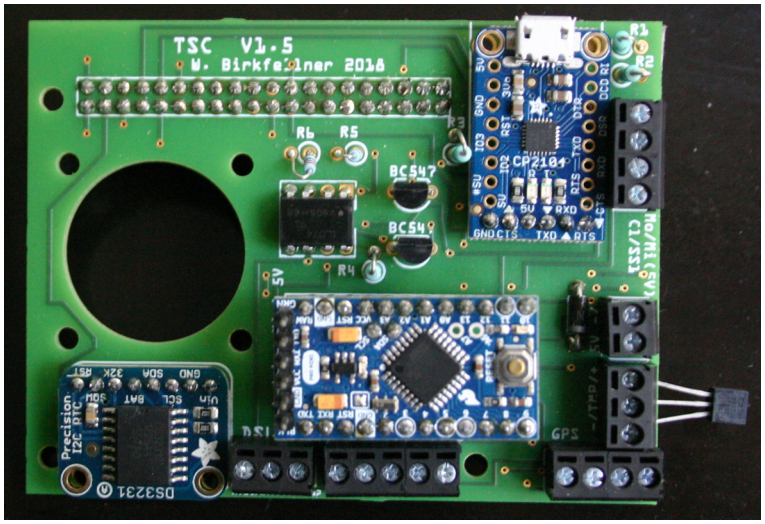


Figure 47: Assembly of the components on the HAT. Denote the sockets for the Arduino Mini Pro, which also feature a small third socket strip for connecting inputs A4 and A5 of the Arduino.

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. 49). 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. Fig. 48 shows how to attach these.

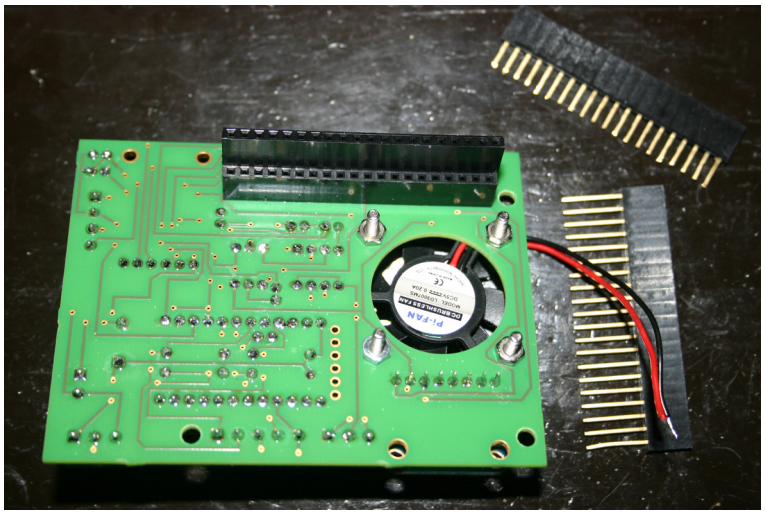


Figure 48: To connect the HAT to the Raspberry, a second row of connector strips is needed. These are shown here – take care that the length and total height of 20 mm is sufficient.

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) is needed. 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/TSC_PIHAT`. 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. 49 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

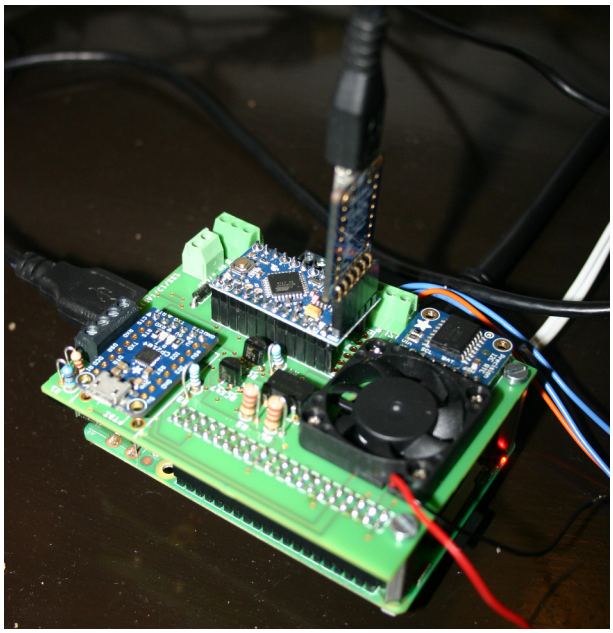


Figure 49: The Arduino on the HAT (already attached to the Raspberry in this illustration) acts as a Analog-Digital converter. It reads voltage levels from ST4 input and the temperature sensor. These data are transferred via SPI channel 0 to the Raspberry Pi. For programming, a sketch (`ST4_Temp.ino`) 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.

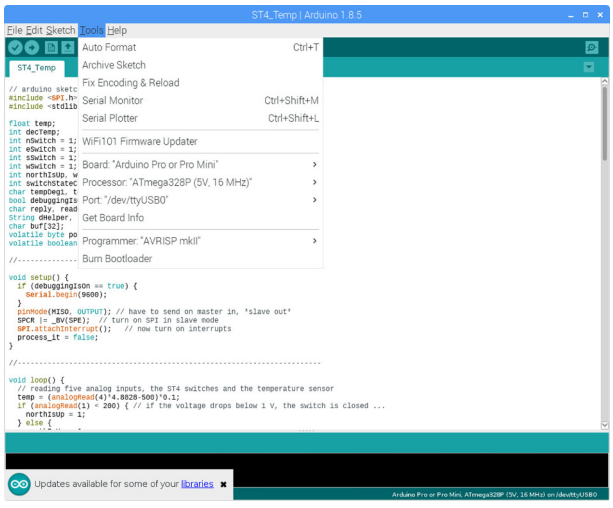


Figure 50: Settings for loading a sketch to the Arduino Mini Pro. Make sure that your USB/Serial converter is connected properly. The Arduino is a Mini Pro 5V/16 Hz - make sure that it is selected like in this illustration here. Once the → button in the icon list is checked, the sketch is uploaded to the Arduino.

the sketch. More on this follows in Chapter 10.6.

10.2 Power supplies and connecting a USB hub

Communication with the guiding camera, the Phidget motor drivers and the touchscreen is managed via USB. Therefore, a powered 4× USB 2.0 Hub is necessary. I have bought one for approximately 10 € and removed the case (Fig. 51). This one can be, for instance attached to the sheet metal holding the display using two-sided adhesive tape (see also Figs. 2, 57 and 58).

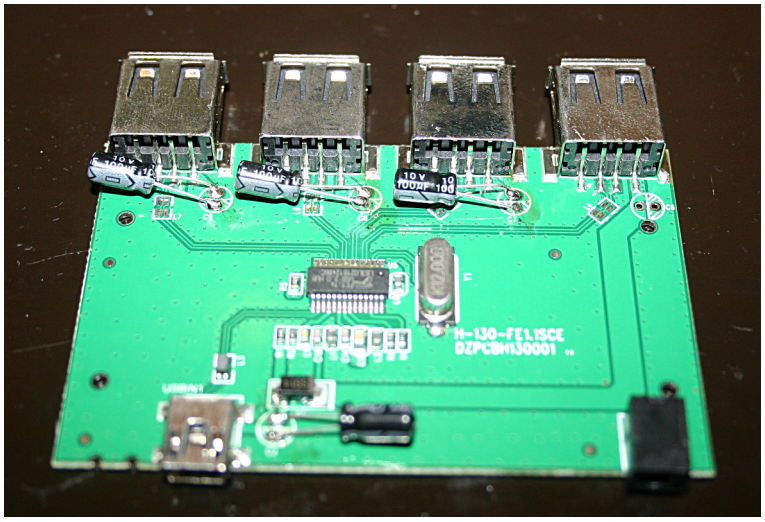


Figure 51: A standard USB - Hub with the case removed. Denote that separate 5V power input is needed. This hub handles communication with the touchscreen, the two Phidget motor drivers and the guiding camera.

For connecting the other basic components, you will also need

- A sufficient 12V Power supply. Something with 10-12A should be fine.
- A 12V – 5V converter providing at least 3A.
- A panel mount USB jack like the Adafruit #908 adapter (see also Fig. 52).
- Jacks for connecting the stepper motors. One can use the breakout board as described in Sect. 10.8.
- An On/Off switch.
- A 2.5 mm jack for connecting the DSLR.
- A jack for the temperature sensor.
- Optional, jacks for the focuser drives if the breakout board is not used.

- A micro-USB cable with sufficient wire diameter to power the Raspberry 3.
- A cable and plug to power the USB hub.
- Short Mini- and Micro-USB cables to connect the Phidgets, the touchscreen and the USB hub.
- A 6 pin RJ12 jack for ST4 if the RJ12 breakout board is not used.

Figure 52: 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; 12V need go to the two Phidgets, the focuser motorboard (if needed) and, of course, the 12V – 5V converter. 5V have to be supplied to the USB hub, the Raspberry Pi itself (take care to use a micro USB cable with sufficient wire diameter), the HAT and the optional focuser board . All powerlines should be connected *in parallel*. Take care to adjust the voltage level on the converter to 5.2 V maximum. Fig. 53 shows a diagram of all components to be connected.

10.3 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' 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 terminals A and B ot the Phidget boards, and B and B' to C and D. **If you want to invert the default direction of your stepper, switch the pairs when connecting them to the Phidget drivers or the focuser board.**

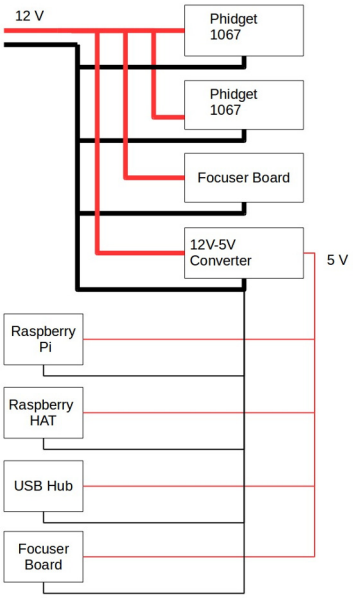


Figure 53: Wiring of power lines; take care that all lines are connected in parallel, and that they all share the same ground. The 12V – 5V voltage regulator should be at 5.2V maximum. This should be measured as some of these regulators do have a display which is not necessarily very accurate.

10.4 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)	Decl. + (N)	Decl. - (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.

10.5 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.

10.6 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.

10.7 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. 54 shows a solution with $250 \times 190 \times 82$ mm where a sliding cover for the 7" Adafruit touchscreen was realized.

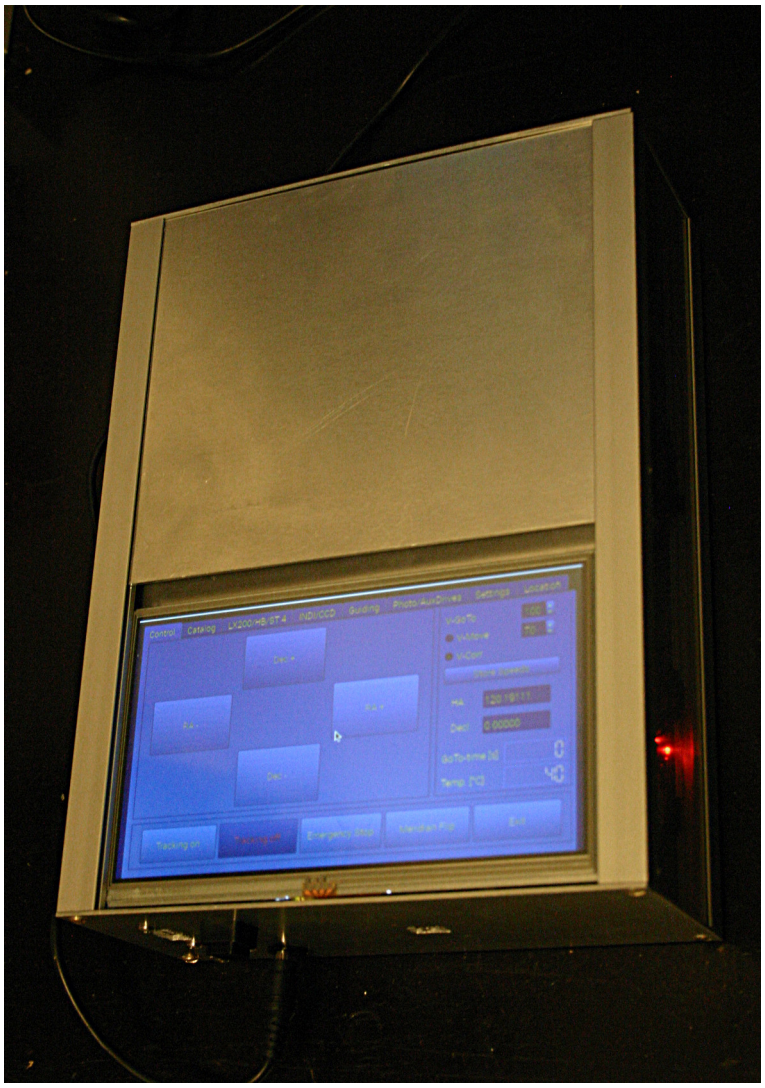


Figure 54: A professional case for TSC; overall dimensions are 250×190×82 mm. Here, Fischer Elektronik GB 1/250/ME profiles were used. The top cover can be used as a lid for the 7" touchscreen. The sides are made of 2 mm transparent plastic. As TSC communicates via WLAN and Bluetooth, it is recommended not to make a full metal case.

10.8 The RJ12 breakout board

An additional PCB called `ConnectorBreakout.fzz` is available for connecting your drives, the temperature sensor and ST4 via standard RJ12 jacks. Fig. 55 shows this PCB. RJ12 plugs are easily connected using a crimping tool and a flat cable with six wires. This PCB allows for a simple connection of the HAT and the focus motor board to external components.

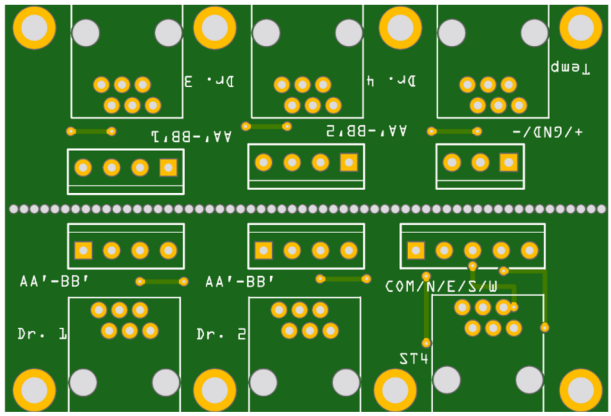


Figure 55: A simple additional PCB providing six RJ12 sockets for connecting motors, focus motors, the temperature sensor and ST4. The PCB can be broken in the middle and mounted to a case for connecting external components to the HAT and the focus motors.

As said before, this breakout board is not mandatory but simplifies connecting components. An assembled version can be found in Fig. 10.8.

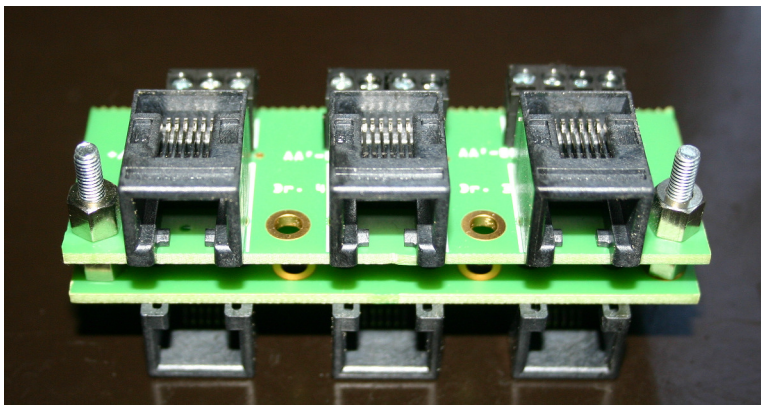


Figure 56: The assembled RJ12 breakout board for connecting motors, the temperature sensor and the ST4 interface. After breaking the PCB in two parts, a possible way to combine these two rows is to mount one atop of the other using standoffs. Internal connections to the HAT are feasible by the provided terminals.

10.9 Synchronizing the hardware clock

Your raspberry has now a hardware clock 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.*

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 cannot be used as it also requires I^2C access, which is already occupied by the realtime clock. 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. 57 and 58 illustrate this setup.

In Fig. 57, 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 handboxes

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 57: The Adafruit 7" display, also shown in Fig. 34, 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.

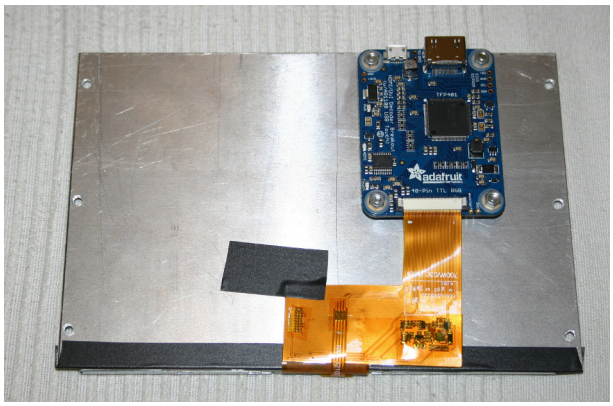


Figure 58: Bottom of the 7" display shown in Fig. 57; the flexpint is secured with a short strip of tape, the HDMI converter is attached by 4 M3 screws.

Two handbox designs exist – one of them uses Bluetooth, the other one operates via the autonomous WLAN access point of TSC. Both support 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. On the other side, it is only operational via the internal WLAN – if your version of TSC operates within a larger existing WLAN network, its operation and setup may become cumbersome as it uses a fixed IP-address. The bluetooth handbox does not have a display, the hardware is slightly cheaper (overall cost is however in the range of 40-50 €), but it is cumbersome to assemble and program and communication is less stable. The reason for the latter problem 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. But it operates independently of SSID-availability.

12.1 The TCP/IP handbox

12.1.1 PCB assembly

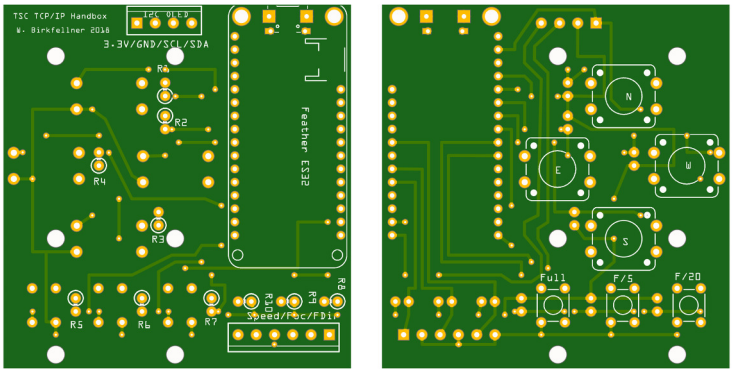


Figure 59: PCBs of the TCP/IP handbox. All momentary switches are integrated on the PCB, the microcontroller (an Adafruit Huzzah32 Feather ESP 32) is powered by a Li/Po rechargeable battery. Permanent switches for handbox motion speed, focuser selection and -direction and a I^2C driven OLED display are connected via terminals.

Little is to be said about PCB assembly, Figs. 59 and 60 show the setup. The Fritzing file for the schematic is to be found in the `Hardware` folder under `TCP_Handbox/TCP_Handbox.fzz`. Basically, seven momentary switches are mounted to the top of the PCB as it can be seen in Fig. 61. 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 of the focusers in increments. On the backside (Fig. 62), a microcontroller with integrated WLAN (the Adafruit Huzzah 32 Feather

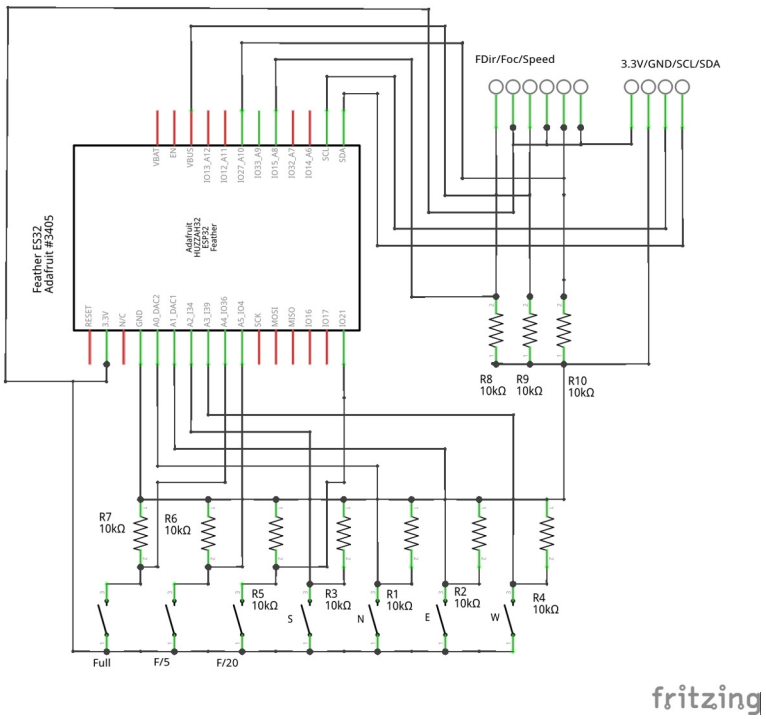


Figure 60: Schematic of the TCP/IP handbox. Basically, ten switches and pull-down resistors are connected to the inputs of the Adafruit Huzzah32 Feather ESP32. An OLED display is driven via I^2C.

ESP32), 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 Adafruit ESP32 becomes necessary. The PCB is not laid out for OLED displays requiring a 3-wire or 4-wire SPI interface.

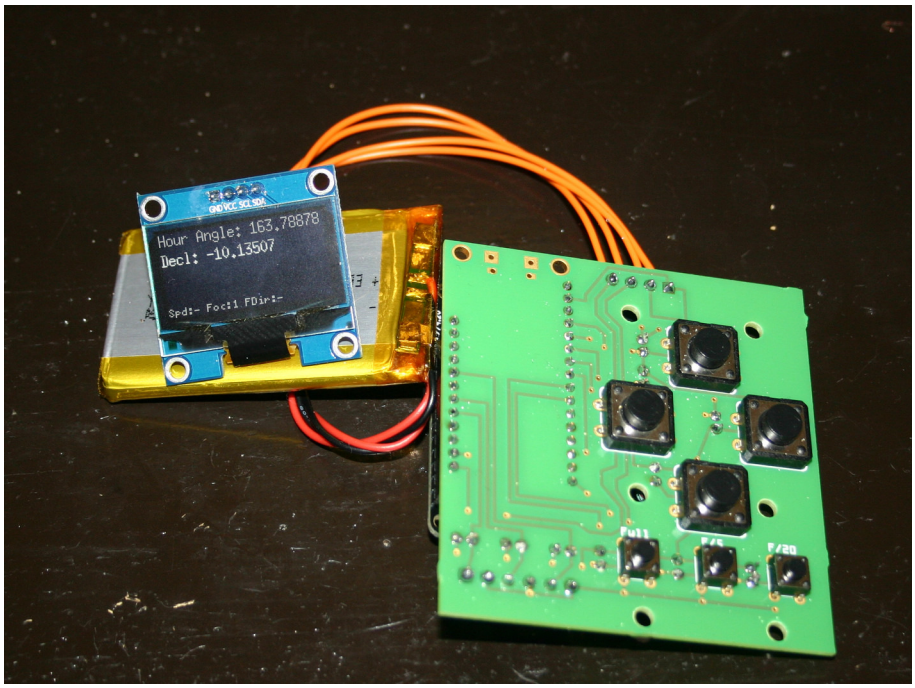


Figure 61: Front assembly of the TCP/IP (or WLAN) handbox. Connected is also a 1.3" OLED display running off 3.3V and an I^2C interface. The program on the Adafruit Huzzah32 ESP32 assumes that the display is controlled by an SH1106 chip and has a resolution of 128×64 pixels. Changes are feasible in the `TCP_Handbox.ino` sketch. The handbox driven by a LiPo battery which can be charged via the USB connector of the microcontroller.

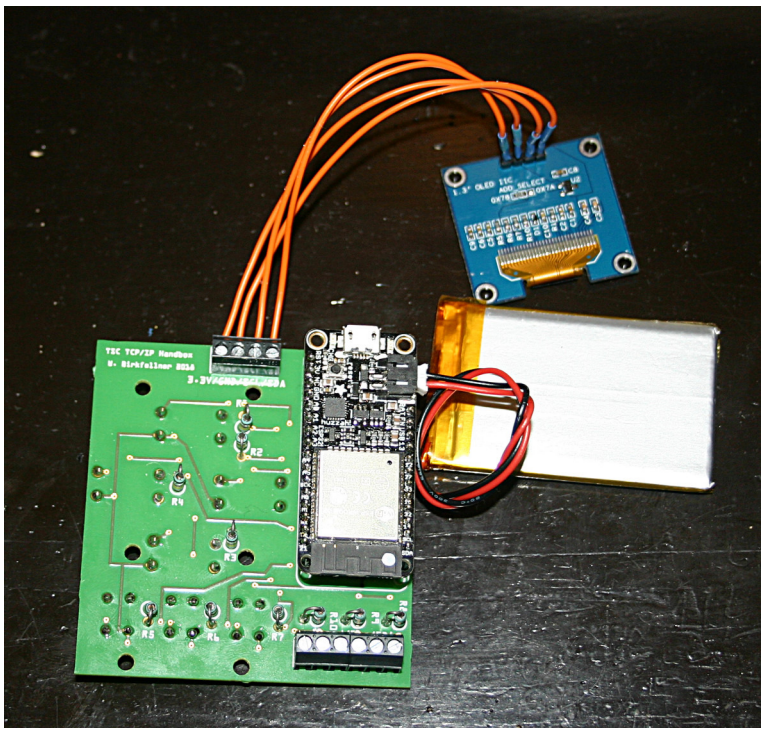


Figure 62: Backside of the PCB. Visible is the Adafruit ESP32, the pull down resistors and the terminals for the three selection switches for motion speed, focuser selection and focuser direction.

12.1.2 Programming the Adafruit Huzzah32 Feather ESP32

In principle, programming the Adafruit Feather 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. And here a little inconvenience occurs – so far, I was unable to install the programming environment for the Feather on Raspian. Installing the adequate environment on another operating system like Windows or Linux is, however, no problem. The details are to be found on the website of the manufacturer, `www.adafruit.com`. As usual, there is also extensive documentation on compiling sketches for the given Huzzah32 Feather ESP32.

The sketch for the handbox is to be found on github in the `Hardware` folder in the `TCP_Handbox` folder as `TCP_Handbox.ino`. As the Feather has a direct USB connector, not additional adapters are necessary. The proper settings for the compile are shown in Fig. 63.

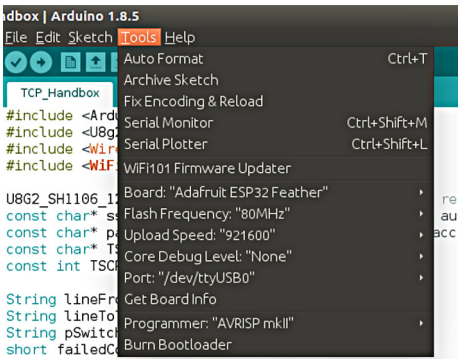


Figure 63: Proper settings for programming the Adafruit Feather ES32 for the TCP/IP handbox using the Arduino IDE.

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. 64.

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. 64.

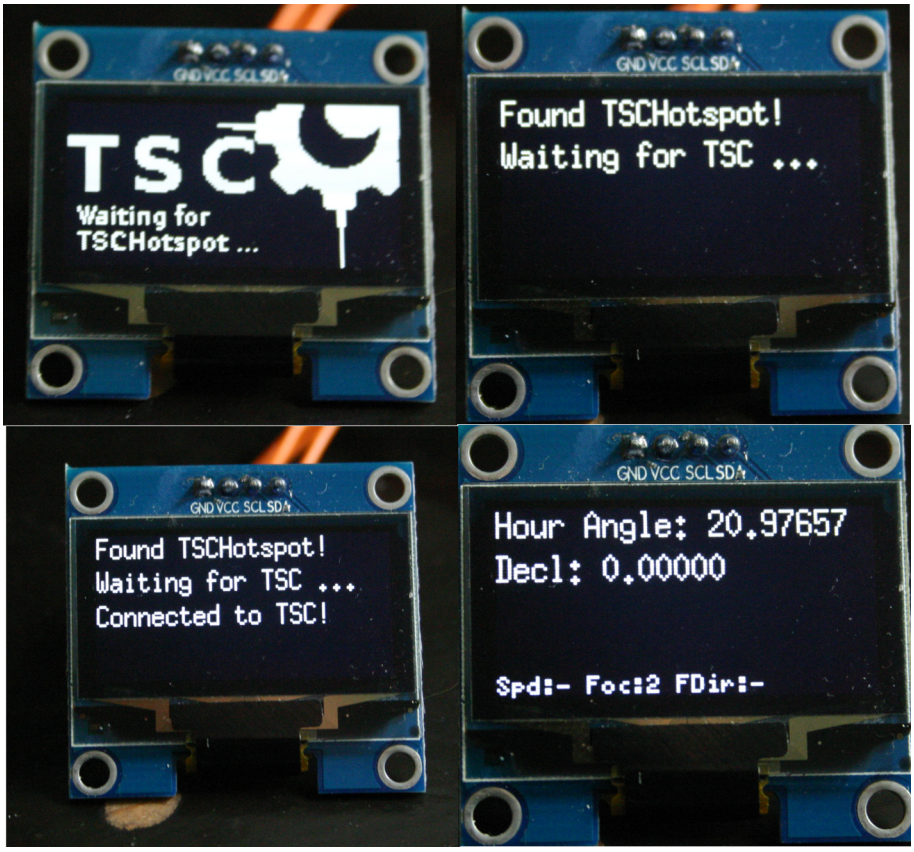


Figure 64: Four screenshots of the initialisation phase of the handbox. First, the handbox tries to establish a connection to the hotspot of TSC, and once this was successful, it waits for TSC to open a server socket. If communciation is established, the topical position is displayed as hour angle and declination.

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. 65 shows the hardcoded values in the sketch. This leads to an important question – what to do if you want to operate TSC within a greater network where fixed IP addresses might be a problem? In this case, it might be preferrable to resort to the bluetooth handbox described in Sect. 12.2.

Figure 65: Settings for a working connection between TSC and the handbox. Denote also that the type of OLED display is fixed in line 6.

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 the focuser was operated, this is also reported. Fig. 66 shows a few screenshots.

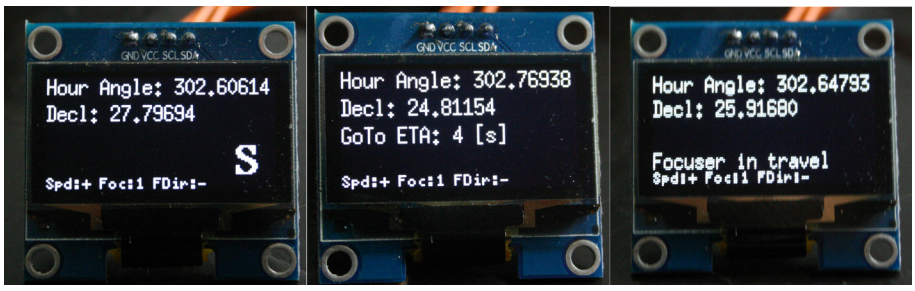


Figure 66: Status messages during operation. When the handbox is operated, the direction is displayed (and position is updated every 2 seconds). In GoTo, the time to arrival is shown. Operating one of the focuser switches is also confirmed.

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. 67). If the internal autoguider of TSC is used, guiding errors are reported as well.

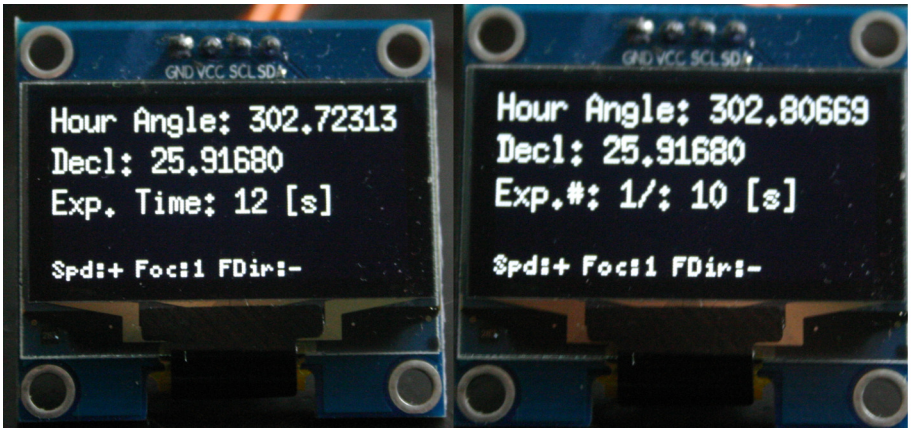


Figure 67: Two more status messages displayed in the TCP/IP handbox. During camera exposures, the remaining exposure time is displayed; in an exposure series, the number of exposures and the remaining time is shown.

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.

12.2 The Bluetooth handbox

This alternative handbox is using a standard case from Conrad (Bopla BOS STREAMLINE BS 400 F-7035) with a battery case for 3 AA batteries. A PDF of the faceplate can also be found on github (`Handbox.pdf`). It features a speed selection switch, an on/off switch, four momentary direction switches, and five switches for motorfocus operation. Fig. 69 shows the assembled unit.

12.2.1 PCB assembly

The handbox is very simple; on the bottom of the PCB, whose Fritzing-design is found in the folder `Hardware/BluetoothHandbox`, one finds a Diode D1 for polarity protection, several terminal strips for connecting the Bluetooth-module (a standard HC-05, available for a few € on ebay), additional switches, an Arduino Mini Pro with 3.3V and 8 MHz and ten 10 kΩ pull-down resistors. On the top side, four momentary switches with a footprint of 12×12 mm like the SparkFun COM-09190 are found, which act as direction switches. Fig. 72 shows the PCB mounted in the handbox.

Due to space limitations, this Arduino Mini Pro has to be soldered directly onto the PCB, and for programming, the serial interface side (the connectors on the short side of the board) should also be connected by thin wires, not by a header. The same holds true for the speed switch, the focuser selection and direction

Figure 68: The TCP/IP handbox in a standard aluminum case with 42 mm height.

Figure 69: The assembled handbox in a standard case. Aside from momentary direction operation, it features a speed selection switch and an interface to the motorfokus drivers. Power is supplied by three AA batteries.

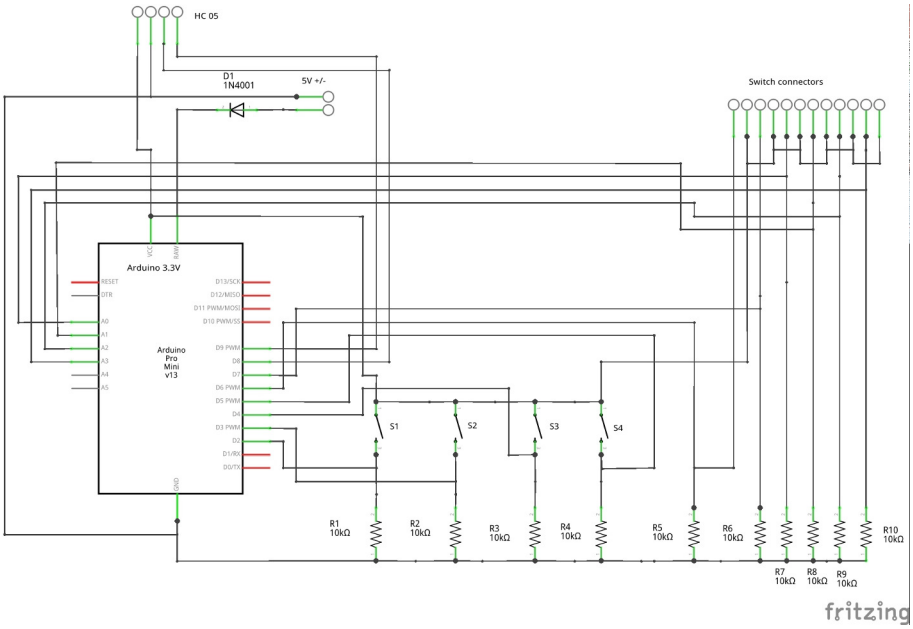


Figure 70: Schematic of the handbox; basically, it is ten switches with 10 pull down resitors read by the Arduino Mini Pro.

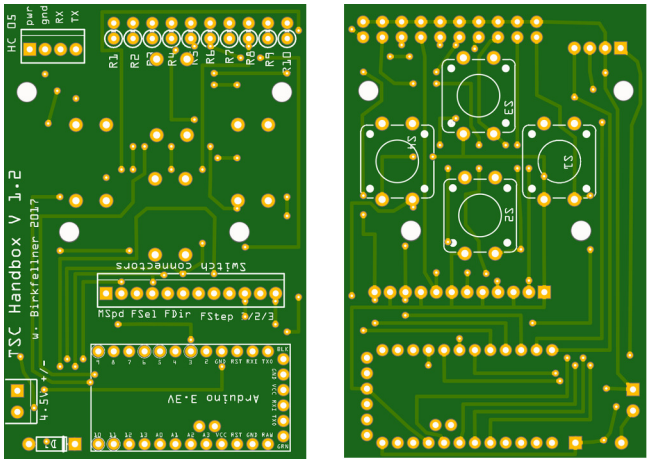


Figure 71: The handbox PCB. On the bottom, terminal strips for power, the microswitches for focuser management and the bluetooth module are found, together with a set of 10 kΩ pull down resistors, a diode D1 for polarity protection and a 3.3V 8 MHz Arduino Mini Pro. The top carries 4 momentary switches with a 12×12 mm footprint. Mounting holes are adopted to the Bopla BOS STREAMLINE BS 400 F-7035 case.

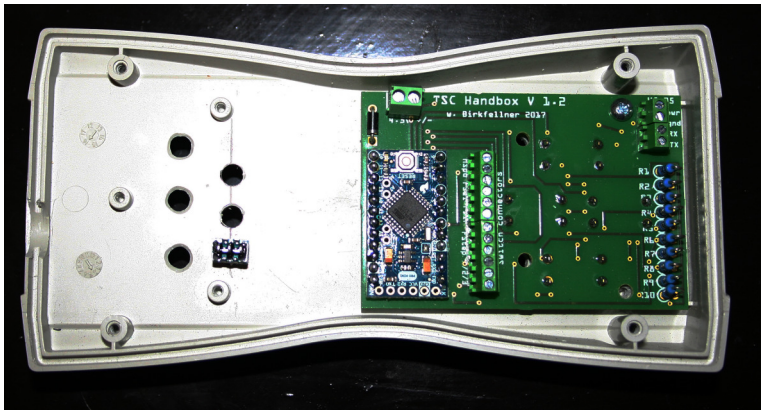


Figure 72: The handbox PCB with components mounted on the top half of the case. Denote that due to the height of the case, the Arduino is directly soldered onto the PCB. In the github-repository, one also finds a PDF file that can be used for the front label and as a drilling template.

switch, and the three miniature monetary tactile switches operating the focuser. If you do not want to use the focuser at all, you only have to connect the speed switch. Here, thin wires are connected to the switches and the central 12 pin terminal strip. This terminal strip is labeled (from left to right) in the following fashion:

- `MSpeed` for the permanent switch selecting the drive speed (labelled `Mount Fast/Slow`. This one controls the travel speed of the handbox motion. It selects values between sidereal speed and the value set for **V-Move** in the **Control** tab of TSC.
- `FSel` selects the focuser motor – this is either drive 1 or drive 2. The permanent switch selecting this is labelled `Focuser 1/2` on the handbox.
- `FDir` connects to the permanent switch selection the direcction of focuser motion. The switch is labelled `Focuser +/-` on the handbox.
- The six terminals labelled `FStep 1/2/3` connect the momentary switches labelled `Full`, `1/5` and `1/20` in the same order. These trigger the motion of the focuser drives when pressed.
- A permament power switch needs to be placed between the + connector of the battery case and the `4.5 V +/-` terminal strip on the PCB.

The three momentary switches `Full`, `1/5` and `1/20` are small buttons with a footprint of 6× 6 mm like the COM-00097 from SparkFun. There is no dedicated PCB for these, rather than that a small experimenting board is prepared – Fig. 73 shows an example.

Connect these switches with very thin flexible wires, and tin their ends. The switches are always connected pairwise to the 12 pin terminal. In the end, the handbox looks as presented in Fig. 74.

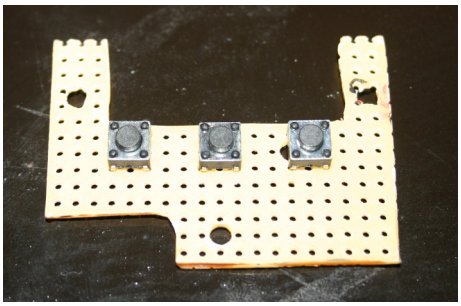


Figure 73: A hand-carved tiny experimenting board holding the three momentary switches for remote focuser operation in place. Grid distance is 2.54 mm.

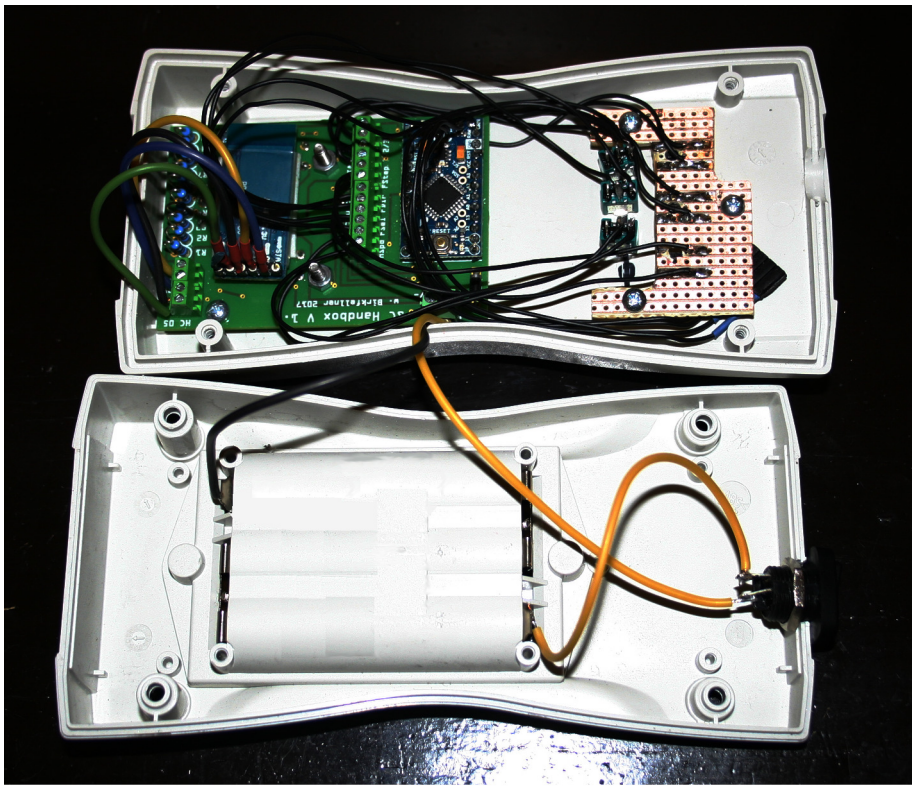


Figure 74: The fully assembled handbox. The HC-05 is connected to the third terminal strip and taped to the PCB by double sided adhesive tape. Once the HC-05 is configured and the the Arduino is programmed, it is best to close this case without destroying any of the wires and leave it like that.

12.2.2 Configuration of the HC-05

Configuration of the Bluetooth module HC-05 is another step to be taken; various instructions exist on the internet to do this with the help of another Arduino, but in fact it is easier to configure the module using a Serial/USB converter (like the Adafruit USB friend used on the HAT and for programming the Arduinos). For this purpose, it is necessary to build up a little circuit on a breadboard as shown in Fig. 75.

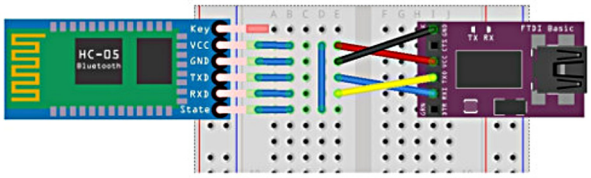


Figure 75: Circuit used to configure a HC-05 with an FTDI-compliant Serial/USB converter like the Adafruit USB friend. After setting this up, it is possible to configure the HC-05 with a common terminal program.

Next, connect to the HC-05 using a common terminal program; the Raspian image holding TSC comes with such a program named `CuteCom`. Configuration of a Bluetooth module takes place via the old Hayes AT commandset. Start `CuteCom` as a superuser by typing `sudo cutecom` in a terminal window and type the following commands:

- Connect to the HC-05 after pressing the HC-05 RESET button.
- Type `AT+NAME:TSC` – this is the name of your BT-device now.
- Type `AT+ADDR?` – this requests the unique MAC address or your HC-05. The response is, in the case as shown in Fig. 76 `+ADDR:2016:2:22:0061`. This needs to be decoded into a common MAC address of the type `XX:XX:XX:XX:XX:XX`. In this case the MAC address is `20:16:02:22:00:61`. Write your address down as we will need it later on.
- Type `AT+UART:9600,1,0` – this is the communication parameters set for the HC-05.
- Type `AT+ROLE:0` – this sets your BT-device in SLAVE mode.

Finally, you have to make the HC-05 known to the Raspberry and TSC; type
`sudo nano /etc/bluetooth/rfcomm.conf`
and change the line
`device XX:XX:XX:XX:XX:XX`
where `XX` is a number from your MAC address for the HC-05.

Next, the device should be registered with Raspian. Remove the wire connecting the `State` pin to `VCC` on the breadboard and reboot the HC-05 with the reset button. Reboot The Raspberry and go to the main menu of Raspian, click

on the Bluetooth symbol and search for devices. A device `TSC` should now be visible, and you will be asked for a password. This is usually `1234`. Raspian will claim that the device is now connected, but no usable services are found. That is ok.

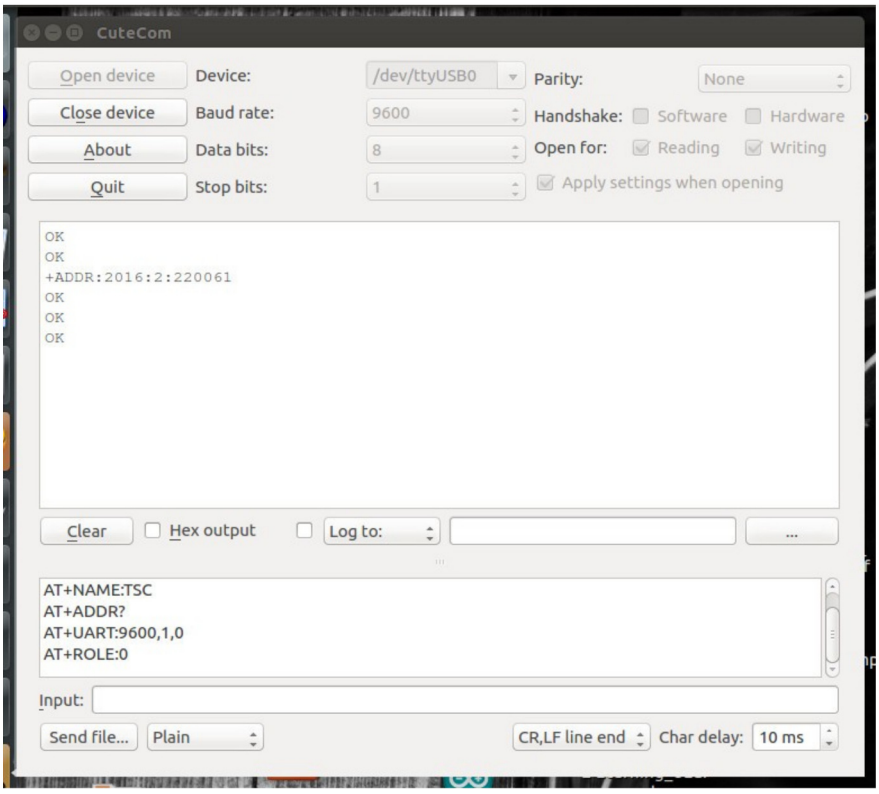


Figure 76: View of the terminal program `CuteCom` connected to the HC-05. After reset of the HC-05 (which is done by a button on the device), communication can be established. Take care that the right device (`/dev/ddyUSB0`) is chosen, that the right communication parameters are set, and that the `Char delay:` is set to 10 ms.

12.2.3 Programming the 3,3V Arduino Mini Pro of the BT-handbox

This Arduino used in the handbox is different – it is also an Arduino Mini Pro from SparkFun, it looks the same as the one on the HAT, but it isn't. It is a low power variant with 8 MHz and 3.3 V power level. This has advantages – power consumption is lower and only 3 instead of 4 AA batteries are necessary. But the Arduino cannot be programmed like the 5 V version as the supplied voltage from the USB/Serial converter would overheat this one. Therefore, Fig. 77 shows a Fritzing diagram for a breadboard setup to connect the Adafruit USB friend to the 3.3 V Arduino while supplying separate power to the microcontroller.

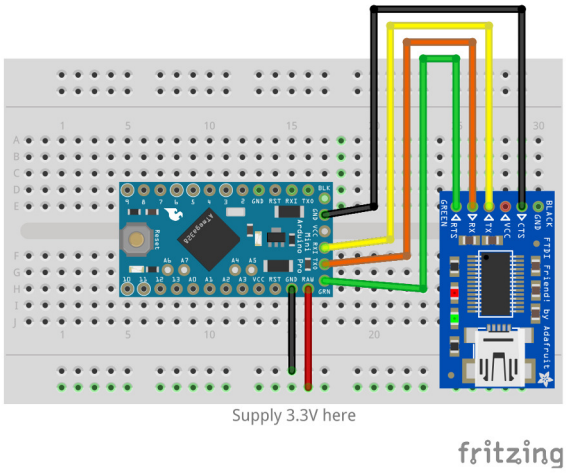


Figure 77: A breadboard schematic illustrating how to upload a sketch to a 3.3V Arduino Mini Pro. the 5V VCC from the USB converter would otherwise damage the microcontroller. Separate 3.3V power has to be supplied via the breadboard.

The sketch for the handbox is found on github in the `Hardware` folder in `BluetoothHandbox`. *Don't forget to change the type of board to* **3.3 V, 8 MHz** *in the Arduino IDE.*

12.2.4 Registering the handbox with TSC

Once everything is set up and programmed, you will need your MAC address of the HC-05 once again (it is advisable to write it on a small label and stick that to the handbox). Startup TSC and go to the **LX200/HB/ST 4**-tab shown in Fig. 7. Move to the **BT- Handbox** tab. Type the MAC address in the field **BT-MAC Addr** and press the `Save BT MAC Address` button. Power up the handbox and reboot TSC – after pressing the **Connect BT** button, the checkbox **BT port up** should be checked and the handbox is ready. **Always turn on the handbox before starting TSC**. There is also a button **Try BT restart** if connection does not work – but it has to be said that the result is not necessarily reliable. As said before, this is somehow connected to the nature of Bluetooth.

12.3 A recommendation

The TCP-handbox is easier to build, easier to use, smaller due to the use of a rechargeable battery and provides more functionality. As said before, its only limitation is the use of the autonomous `TSCHotspot` WLAN of TSC.

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 Phidget 1067 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/Focusmotors` directory. Figs. 78 and 79 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. 80 shows the assembled board. The power of these drivers cannot be compared the Phidget 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 Phidget 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

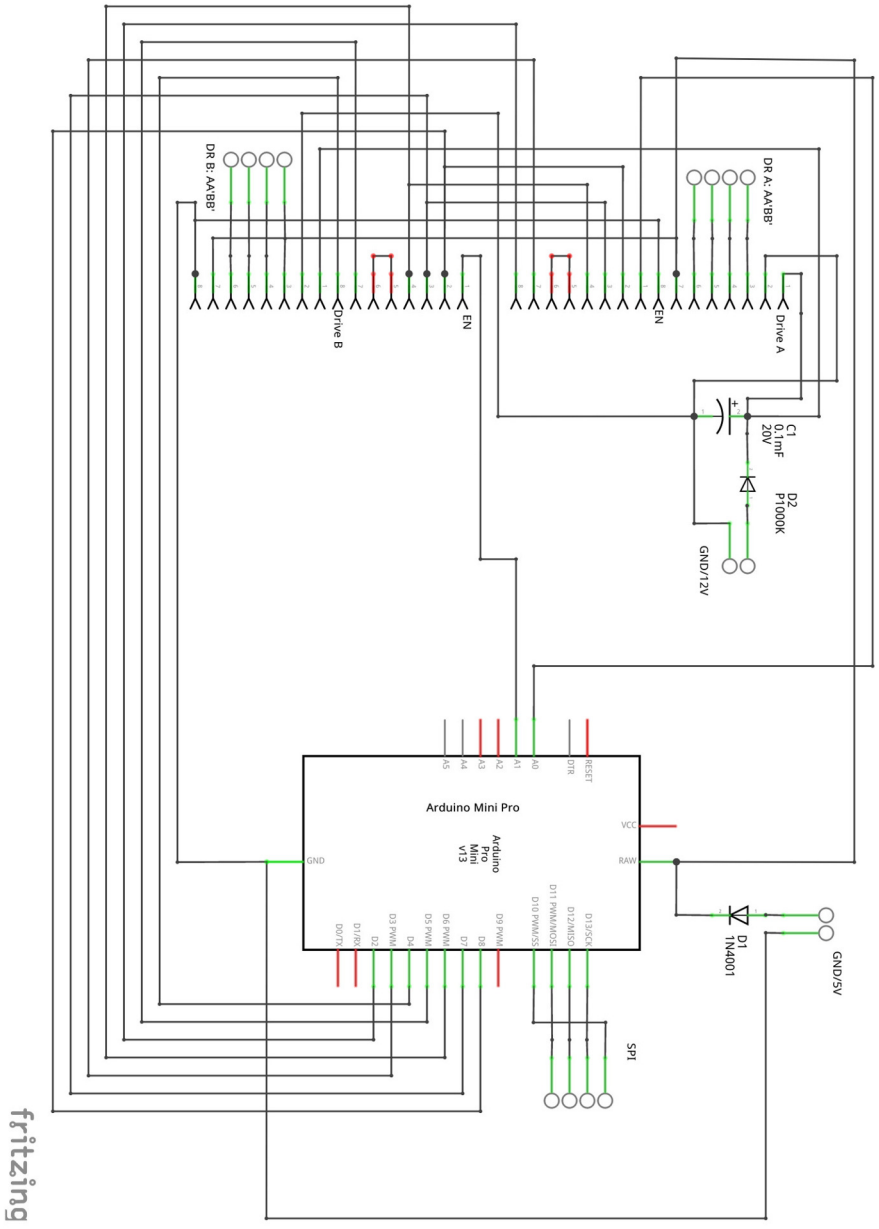


Figure 78: The schematic for the focuser board. The stepper drivers are controlled by an Arduino Mini Pro (16 MHz 5V version), which is controlled via SPI by TSC.

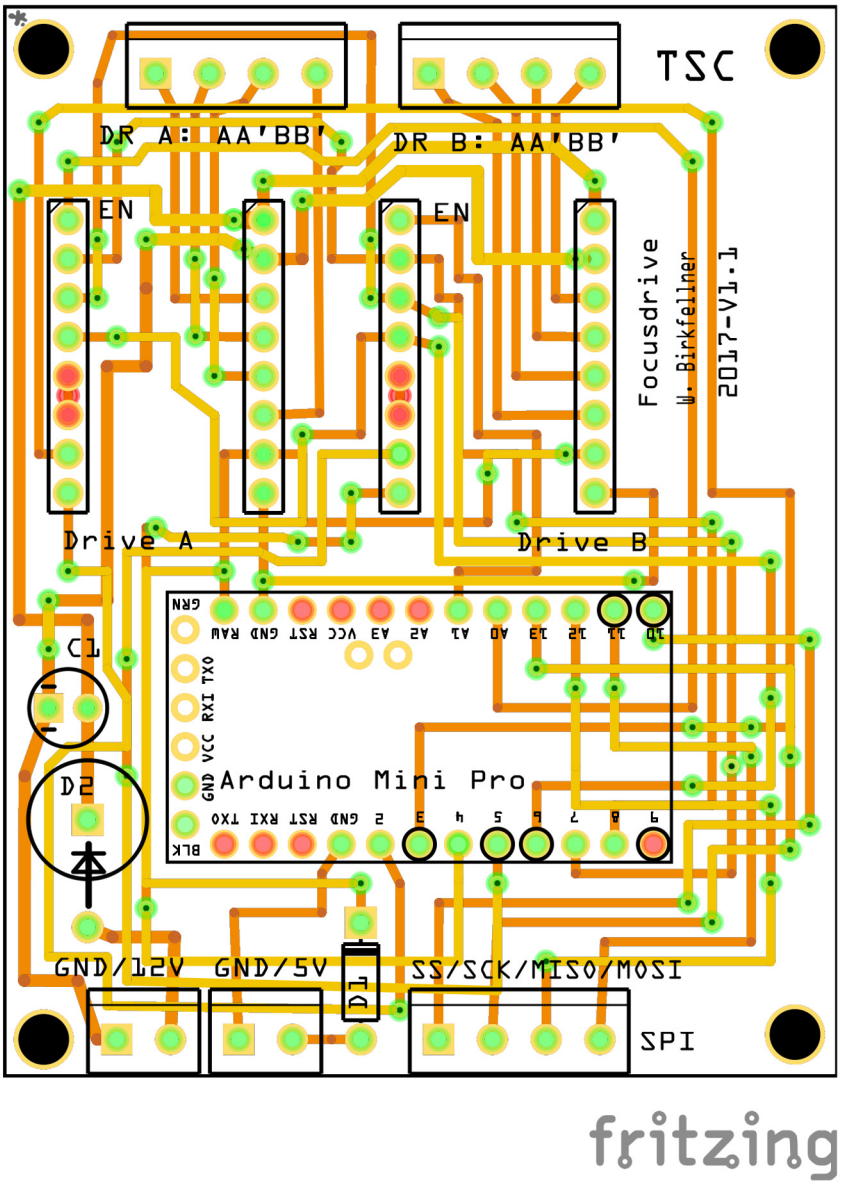


Figure 79: The PCB of the focuserbaord. The stepper controllers (Polulu A4988 or Polulu DRV 8825) are placed on sockets, and so is the Arduino Mini Pro.

depends on the size of the motors. Bipolar stepper motors up to NEMA 17 size can be operated easily by the focusboards.

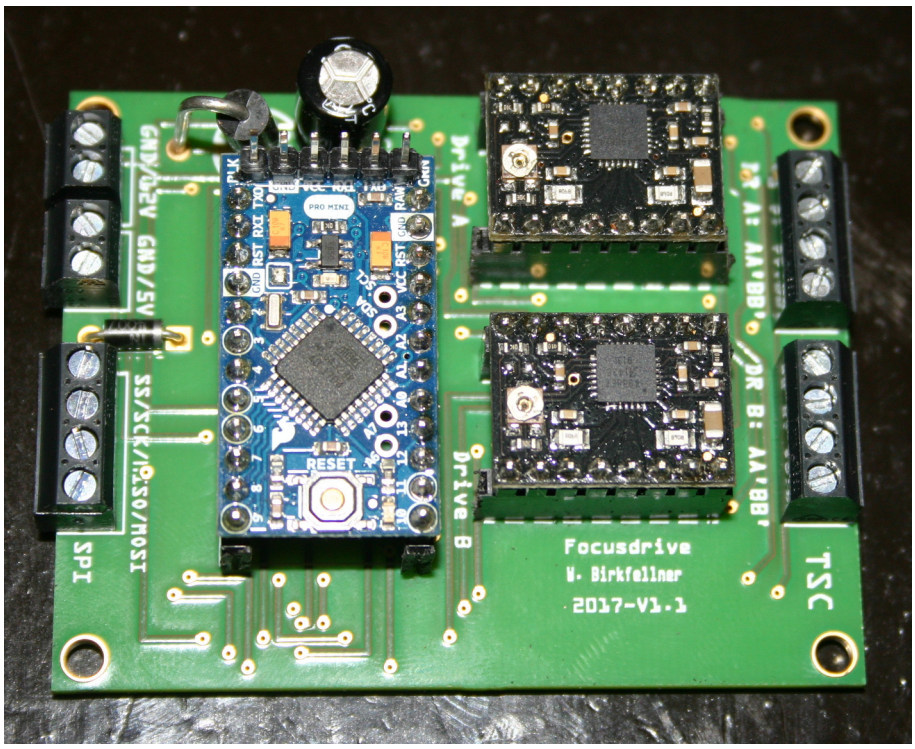


Figure 80: The assembled focuser board. In this case, Polulu A4988 drivers were used. One can clearly see the two potentiometers in the lower left part of the drivers – these are used to set the coil current of the additional stepper motors.

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. 53), 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

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.
- **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.
- **HAT** Hardware Attached to Top. A common name for additional hardware stacekd onto a Raspberry Pi.
- **HC-05** The name of the Bluetooth-module used in the BT-handbox.
- **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.
- **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.