

Setting up TSC - alpha with a Raspberry Pi 3 Model B v. 0.0.1

TSC (TwoStepperControl) is a software project by Wolfgang Birkfellner and Steffen Elste to realize an open integrated control system for astronomical telescopes. The basic idea is to use a Raspberry Pi to control a telescope and to provide basic autoguiding functionality without additional external computers. As Raspbian Jessie is not an RTOS, it cannot control stepper drivers directly like a microcontroller; therefore, extra hardware is necessary which receives basic parameters (number of steps, velocity, acceleration) from the software. Currently, I am using a Phidgets 1067 bipolar driver board although other alternatives are under consideration. The steppers are monitored by the Pi in separate Qt-Concurrent threads so that software functionality is maintained during stepper operation. So far, only fork mounts are supported.

Software repository:

<https://github.com/selste/TwoStepperControl>

Basic components:

- Operating System: Raspbian Jessie - <https://www.raspberrypi.org/downloads/noobs/>
- Currently, the locale should be english.
- Raspberry Pi 3 Model B, with at least a 16 GB MicroSD card and a 2A micro-USB power adapter. Cost ~ 70 €.
- Be aware that one needs a lot of power to drive the Phidget boards. Get yourself a sufficient power supply for everything else but the Pi.
- Buy cooling bodies for the Pi, especially the ARM CPU (~5 €), and plan to get a fan for the Pi and the Phidget boards.
- Stepper control boards: 1067_0 - PhidgetStepper Bipolar HC, available from www.phidgets.com or www.nodna.de. Two boards are necessary. These operate one bipolar stepper each with up to 4 A coil current. 1/16 microsteps are available, the board is controlled via USB 2.0. Cost for each board is currently ~90 €. A development with the SparkFun AutoDriver - Stepper Motor Driver (v13) – www.sparkfun.com - using STMicro L6470 dSPIN drivers capable of 3A per coil and 1/128 microsteps addressed via SPI is planned but not yet available. As Raspbian is not a RTOS, it cannot control the drivers directly.
- A HDMI cable and an HDMI monitor. Development right now aims at using the Adafruit 5" HDMI 800*480 Backpack Touchscreen (~80 €). Basically, it is also possible to connect via W-LAN or WI-FI direct and a VNC viewer to the Raspberry using a mobile or a tablet – check <https://www.raspberrypi.org/documentation/remote-access/vnc/>
- A wireless mouse and keyboard. Cost ~20 €.
- A RS232 connection to connect to Cartes du Ciel and other programs. For this purpose, it is necessary to get a simple level shifter circuit like this one: <https://www.conrad.at/de/rs232-erweiterungs-platine-fuer-den-raspberry-pi-1337093.html>. However, if you don't have a serial

port and a NullModem – cable, it might make sense to use one of these although I have not tried it yet: <https://www.adafruit.com/product/954> . Finding out about available RS232 port can be done in UNIX-type systems by typing

```
dmesg | grep tty
```

and the command to make the port readable for all users is

```
sudo chmod a+rw /dev/ttyUSB...
```

- A Raspberry Pi compatible USB 2.0 hub with separate power. Not all hubs work, so be careful. I found the USB port in the lower right of the Pi to be the most stable.
- Development environment is Qt and QtCreator using C++. Be aware that some versions of Jessie might give trouble when starting the QtCreator – try `qtcreator -noload Welcome` and give the correct paths to compilers and debuggers in the Options – menu. Installing qtCreator from the repositories of Raspbian is currently not trivial. Once you have a running System, do not upgrade for the time being. Some info can be found – for instance on missing libxcb libraries- here: https://wiki.qt.io/Native_Build_of_Qt_5.4.1_on_a_Raspberry_Pi
- For the bluetooth handbox, one also needs an Arduino Uno (24 € for the original one (strongly recommended) or a few coins for chinese copies - other models like the Nano might work as well) and a HC-05 bluetooth board (3-6 € + postage on ebay). Also, a lever switch and four touch switches, wires, a 9V power supply, a housing and 5 10k resistors are necessary. Putting the handbox on a prototype shield is feasible. An Arduino sketch can be found in the repository called “`sketch_handbox_TSC.ino`” for operating the handbox
- **Software requirements:**
 - Install the Phidgets – library and set the right access privileges for the USB port if you don't want to run the program in `root` mode. Check <http://www.instructables.com/id/Getting-Started-with-Phidgets-on-the-Raspberry-Pi/> ... Basically, you need the libusb 1.0 and libphidgets 2.1. Changing the udev rules is also necessary (this is taken from <http://tordwessman.blogspot.co.at/2012/01/running-phidgets-under-linuxubuntu.html>):
 - `$ sudo nano /etc/udev/rules.d/80_phidget.rules`
 - Add the following content:
`SUBSYSTEMS=="usb", ACTION=="add", ATTRS{idVendor}=="06c2", ATTRS{idProduct}=="00[3-a][0-f]", MODE="666"`
 - Set `USB_DEVFS_PATH` to `/dev/bus/usb` by adding to `~/.bashrc`:
`export USB_DEVFS_PATH=/dev/bus/usb`
 - Restart udev:
`$ services udev restart`
 - If it doesn't work. Try the good old:
`$ sudo reboot`

- Currently, only image acquisition of a QHY5 CCD-camera is realized via INDI; other cameras may follow. However, one needs INDI, and a tutorial for installing it on the Pi is found under <http://indilib.org/support/tutorials/139-indi-library-on-raspberry-pi.html>
- In order to utilize the serial port, one has to disable it's current purpose – serving as a console to the pi.
 - Edit /boot/config.txt by typing `enable_uart=1`.
 - Disable console output from Jessie by typing:

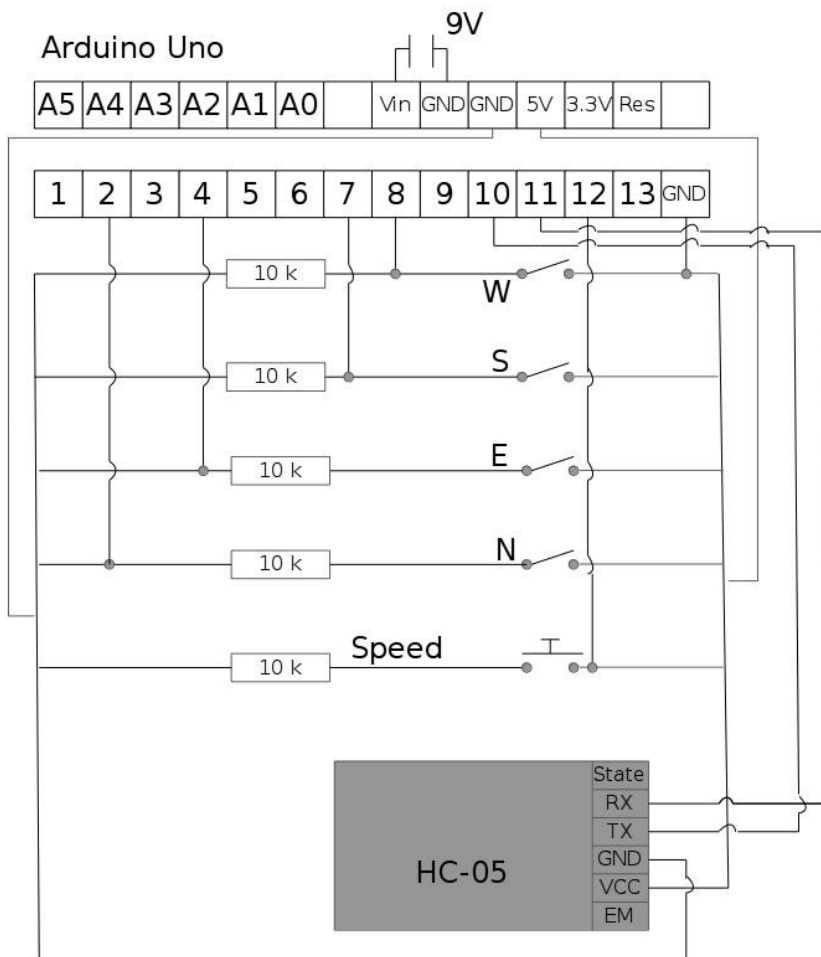

```
$ sudo systemctl stop serial-getty@ttyS0.service
$ sudo systemctl disable serial-getty@ttyS0.service
```
 - Type `sudo nano /boot/cmdline.txt` and remove the statement `console=serial0,115200` from that text. After reboot, /dev/ttyS0 should be available. More can be found under <http://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3/>
 - Make the port readable for all: `sudo chmod a+rw /dev/ttyS0`

The Bluetooth handbox:

The handbox features five switches, one on/off switch for speed selection and four switches for motion. It features an Arduino Uno and a HC-05 Bluetooth interface. A simplified schematic is shown.

Setting up the HC-05:

- I follow the instructions from <https://myraspberryyandme.wordpress.com/2013/11/20/bluetooth-serial-communication-with-hc-05/>
- For setup, the HC-05 “STATE” has to be connected to 5V. The sketch “*sketch_HC05_setup.ino*” is used to configure the HC-05. CR/LF have to be enabled for communicating via the serial interface.
- Send the following commands:
 - `AT+NAME:TSC`
 - `AT+ADDR?` gives the MAC address of the device – write it down!
 - `AT+UART:9600,1,0` sets the communication parameters
 - `AT+ROLE: 0` makes the device a slave
- Disconnect the wire from 5V to the STATE pin.



- Make sure that Bluez is downloaded - <https://learn.adafruit.com/install-bluez-on-the-raspberry-pi/installation>
- Write a file `rfcomm.conf` in `/etc/bluetooth`; it should read:

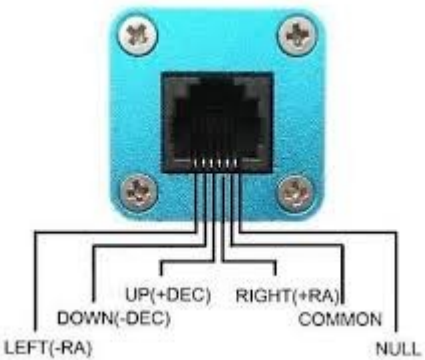
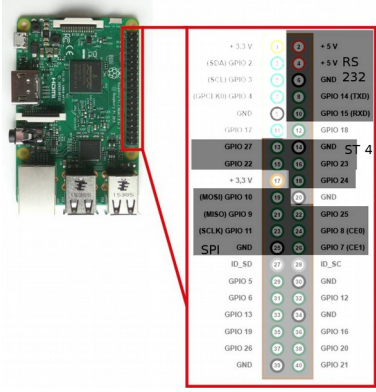

```
rfcomm0 {
    # Automatically bind the device at startup
    bind yes;
    # Bluetooth address of the device
    device 98:D3:31:FB:2A:8C
    # your MAC address for the HC05 should go here!
    # RFCOMM channel for the connection
    channel 1;
    # Description of the connection
    comment "TSC";
}
```

- Write a file “pincodes” in “/var/lib/bluetooth/xx:xx:xx:xx:xx:xx/” where the PIN code of the HC05 is stored; this is usually 1234, it can be edited in setup, but this is default.
- Try to pair the HC05 by using the BT-tool ... it should work now.

The ST 4 interface:

ST 4 can be used. For this purpose, a standard RJ 11 jack has to be wired to the GPIO port of the Raspberry Pi 3 as follows:

Direction	Direction	Wiring Pi Port	BCM Pin	Pi Pin	ST 4 Pin
Declination +	North	2	27	13	5
Declination -	South	4	23	16	4
Right Ascension +	West	5	24	18	3
Right Ascension -	East	3	22	15	6
GND				14	2

	
<p>The ST 4 Pins, as seen from the jack. Numbering goes from right to left.</p>	<p>The GPIO pins of the Raspberry Pi3; the program needs the pins in the gray areas.</p>

Current features:

- Independent control of two Phidget steppers. Mount and drive parameters can be directly edited and saved in the GUI of TSC.
- Motion of the mount at variable speed using a handbox-style interface.
- Bluetooth handbox with variable speed.
- Pulse Guiding and ST4 interface.
- GoTo using an internal set of catalogues; the catalogues are named .tsc but are in fact .CSV files that can be edited with every spreadsheet like Excel. They have the following structure

- Number of entries in the first cell of the first row.
- Epoch in the first cell of the second row.
- Constellation, Object name, RA hour, RA minute, RA second, sign of the declination, declination degrees, declination minutes, declination seconds.
- Example (start of the Messier list):

110 , , , , , , , ,

2000 , , , , , , , ,

Taurus , Messier 1 , 5 , 34 , 30 , + , 22 , 1 , 0

Aquarius , Messier 2 , 21 , 33 , 30 , - , 0 , 49 , 0

and so on ...

- Epoch can be corrected to the current year.
- Be aware that after building, the .tsc catalog files have to be copied to the build folder.
- Interfacing to a local INDI server for CCD-camera acquisition and acquisition of images from a guiding camera – basic. Currently, only the QHY5 camera is supported. An adequate indiserver can be started in the GUI. Experimental autoguidng is implemented.
- Support for Sync and Slew over serial RS232 communication from CdC (tested under Ubuntu 16.04 and Windows 7) and CA2 (Windows 7). A perfectly robust basic LX200 communication is currently possible with CdC (SkyChart) and Computer Aided Astronomy (CA2). Communication via ASCOM and INDI is not fully debugged yet. Make sure to activate “High Precision Display” in the CdC Telescope Control Panel (that is a bug in CdC – upon establishing communication, it sets the response format of the controller to high but does not do the same thing in it’s internal representation).