Connecting to the 4m telescope VNC server

- Connecting from the guest terminal

As the computers in the observer building are connected to the MPIfR internal network, one can simply access the VNC by entering

```
vncviewer 134.104.70.120:99
```

into a unix terminal and then entering the password.

A window should appear with the VNC desktop. This is the telescope control computer "`4mteleskop`".

- Connecting from outside the MPIfR network

You need to establish an SSH tunnel to the MPIfR servers. For this you need an MPIfR account. With this you need to log into username@portal.mpifr-bonn.mpg.de when establishing the tunnel.

For Windows *MobaXterm* has a VNC client and a tunneling option in a single program for convenience.

Starting the telescope control programs

1. The clipboard is not shared with the VNC. Thus, it helps if you open the page containing the most important commands through a browser from within the VNC.
2. Open a terminal on: `4mteleskop`
3. If not already there, go to the directory: `/home/operateure/`
4. If you want to specify different folders to save your spectroscopy and continuum data in, you can do so by editing `/home/operateure/start4m.sh` through any text editor.
5. Use the command: `./start4m.sh` This should open all the necessary windows to operate the telescope.
6. Minimize or resize the windows you don't immediately need so the `4mcontrol.py` window with the blue input line is visible. This is from where you can control the telescope.

4mcontrol.py is the main interface to the telescope where one can enter commands. skripttelcontrol4m.py makes the calculations for the telescope positioning and constantly returns imprecise values about where the telescope is pointing. The VLC Media Player window has a live feed from a webcam showing the telescope. The Regler window displays the datastream from the motors of the telescope. The windows with the data plots pop up automatically when a new batch of data is saved. datawriter.py writes the data for the continuum backend, while fftsdatwriter does the same for the FFTS backend. pbeControl displays data activity from the telescope.

Preparing the measurement

- **Continuum backend**

The continuum backend measures the total flux coming from the telescope over the whole range of Frequencies received. It is usually used to quickly map out where radio sources are. It can be started by entering:

```
pbebackend('cmdusedchannels 1 1','cmdnumphases 1','cmdmode
external','configure')
```

- **FFTS backend**

The Fast Fourier Transformer backend separates the signal from the telescope into its constituent frequencies. This allows the detection of physical processes radiating at distinct frequencies in specific regions of the sky. It can be started by entering:

```
fitwriter('band1:cmdnumspecchan 8192','band2:cmdnumspecchan
8192','cmdusedsections 1 1','cmdmode int','cmdsynctime 200000','cmdblanktime
2','cmdnumphases 1','configure')
```

1. **Starting the observation**

In both cases the selection of the backend must be followed up by

```
send()
```

to actually send the command through the network to the appropriate devices. After this you can start the observation by entering

```
start_messung()
```

Observing with the telescope

Limits

The telescope can only move within certain limits. These are:

Elevation	0°-90°
Azimuth	-70°-270°

Be careful not to let the telescope cross over these limits.

Observation Commands

- **Point**

In order to point the telescope towards a specific heading - for example when shutting it down - you can use the command:

```
set_position_azel(integration = 0, el = (posel,unitel), offsety =
(offy,unitoffy), az = (posaz,unitaz), offsetx = (offx,unitoffx), velaz =
None, velel = None)
```

The correct shutdown position is Elevation: 35, Azimuth: 130,

- **Track**

In order to track an object across the sky, you can use the command:

```
track(messobjekt, integration = None, offsetx = (offx,unitoffx), offsety
=(offy,unitoffy))
```

- **Crosscan**

To detect the maximum intensity and precise position of sources, you can use the command:

```
crossscan(integration,messobjekt = None, az = (posaz,unitaz), el =
(posel,unitel), lengthx = (lengthaz,unitlenx), lengthy = (lengthel,unitleny),
offsetx = (offx,unitoffx), offsety =(offy,unitoffy))
```

The speed at which the telescope moves is determined by the integration time given, and the length given for the crossscan. The telescope will move to approximately to "position - 1/2 length" and then sweep over to approximately "position + 1/2 length" over the time frame given. It adds a couple of seconds of movement at the beginning and the end to avoid sudden jolts of the telescope.

- **Calibration**

The calibration diode is used to give a reference flux to determine the actual signal strength by comparing the incoming signal to it. For the time being it has to be turned on and off manually at the start of each scan. It should only need to be turned on for a few seconds. This can be done via:

```
cal_on()
```

and

```
cal_off()
```

- **Multiple tasks**

Currently the only way to queue up multiple tasks is to write a separate short python script and execute it via the `execfile("/path/name.py")` command. Take care to space out your scans with `time.sleep()` to allow for the full integration time as well as telescope movement!

- **Example**

This was an azimuth crossscan over the Sun via `crossscan(50000,"sun",lengthx=(14,'d'))`, with both continuum and spectroscopy backends enabled.

Variables

The commands for the telescope have several variables that you change for each command. If you leave out a variable, the control program will use the default value. (Mostly 0.)

Variable	Purpose	Default Value	Unit	Other
`integration=None,`	sets the time for collecting radio flux from the specified source	None=infinite	`ms`	0 means no data collected
`offsetx=(offx,unitoffx),`	sets an offset in azimuth to point beside the given coordinates	0, 0,	`unitoffx` with the format `'d'` for degrees	can be used to fine tune pointing
`offsety=(offy,unitoffy),`	sets an offset in elevation to point beside the given coordinates	0, 0,	`unitoffy` in the format `'d'` for degrees	can be used to fine tune pointing
`velaz=None,`	sets the velocity at which the telescope moves along the azimuth axis	'Schleichgang'	'Schleichgang': slow, 'Eilgang': fast, 1: 1°/sec	velocity is calculated automatically for crosscans
`velel=None,`	sets the velocity at which the telescope moves along the elevation axis	'Schleichgang'	'Schleichgang': slow, 'Eilgang': fast, 1: 1°/sec	velocity is calculated automatically for crosscans
`coorsys = 'J2000.0',`	specifies in reference to which point in time the coordinates were given	'J2000.0'	'J2000.0' OR 'B1950'	
`lng = (l, unitl),`	sets the longitude position of the telescope in galactic coordinates	0, 0,	`unitl` with the format `'d'` for degrees	
`br = (b, unitb),`	sets the longitude position of the telescope in galactic coordinates	0, 0,	`unitb` with the format `'d'` for degrees	

Variable	Purpose	Default Value	Unit	Other
`ra = (rec, unitrec),`	sets the right ascension position of the telescope in equatorial coordinates	0, 0,	`unitrec` with the format `'d'` for degrees	
`dec = (decl, unitdec),`	sets the declination position of the telescope in equatorial coordinates	0, 0,	`unitdec` with the format `'d'` for degrees	
`el = (posel,unitel),`	sets the elevation position of the telescope in local coordinates	0, 0,	`unitel` with the format `'d'` for degrees	with this option the telescope does not turn with the rotation of the earth
`az = (posaz,unitaz),`	sets the azimuth position of the telescope in local coordinates	0, 0,	`unitaz` with the format `'d'` for degrees	with this option the telescope does not turn with the rotation of the earth
`messobjekt,`	specifies which source to point towards	None	—	names looked up in in CDS name resolver
`lengthx = (lengthaz,unitlenx),`	specifies the distance the telescope should move in azimuth during a crossscan	0, 0,	`unitoffx` with the format `'d'` for degrees	
`lengthy = (lengthel,unitleny),`	specifies the distance the telescope should move in elevation during a crossscan	0, 0,	`unitoffy` with the format `'d'` for degrees	

Other commands

- get_position()

Displays the current positioning of the telescope in az/el.

- get_status()

Displays whether or not the telescope is already pointing at the desired target.

Bugs

- **Telescope not moving, not giving out data**

This indicates that the telescope control programs froze for some reason and need to be restarted. Close all windows by pressing `Ctrl+C` or the X button on each of them. Restart the telescope control programs via `./start4m.sh`.

- **Telescope still not moving, optionally giving error messages**

If a crossscan was started close to the hardcoded Azimuth endpoints of the telescope, it is possible that the telescope crossed over this endpoint. This means the actuator controls will have to be manually restarted by a technician.

- **Multiple backends online at the same time, don't turn off on restart**

This happens occasionally, and while the plotting windows can be a nuisance, the data is unaffected and measurements can continue normally.

- **134.104.70.120:99 is not accepting the password**

First the validity of the password needs to be checked. If it is still not allowing a connection the wrong password may have been entered too many times, in which case a technician will need to restart the VNC server to reset the lockout.

- **Other Error**

If the telescope does not function due to a different error, you should contact a technician.