

The University of Oklahoma Nielsen Hall SRT

Operator's Manual

Version 0.2

JOHN TOBIN¹²³
NICKALAS REYNOLDS²³⁴

November 2, 2017

¹jjtobin@ou.edu

²GitHub: <https://github.com/OUsrc/srtn.git>, Group Email: nhnradiotelescope@groups.ou.edu

³Discord Chat: <https://discord.gg/vx536X2>, Website: <https://www.nhn.ou.edu/srt>

⁴nickreynolds@ou.edu

Abstract

The University of Oklahoma's Small Radio Telescope (3m) was sponsored by Dr. John Tobin as an outreach and educational tool for the Norman Community. The use of the telescope helps to aid in expanding one's knowledge of radio astronomy, engineering principles, and proper research methods. The kit was implemented off of MIT Haystack's SRT, fabricated through RF Ham Design, constructed by a team of students and professors¹, and the code was provided by MIT Haystack under the MIT Public License.

Information specifically regarding the MIT Haystack Observatory can be found at <http://www.haystack.mit.edu/edu/undergrad/srt/index.html>

Additional Information

The website² for this file contains:

- A link to the downloadable PDF and L^AT_EX code.
- The SRT source code.
- Useful scripts for observing and parsing the data

Acknowledgements

- Hardware design, development of software, and instructions: (www.haystack.mit.edu/edu/undergrad/srt/)
- Fabrication of the SRT parts: (<https://www.rfhamdesign.com>)
- Parse code and instructions: (<https://github.com/OUhrt/srtn.git>)

¹Prof: John Tobin; Grad students: Paul Canton, Hyunseop Choi, Nickalas Reynolds, Rajeeb Sharma, G-PSI; Undergrad Students: Jacob Gill, Lisa Patel, and Brian Stephensen; Staff: Barry Bergeron, Andy Feldt, Debi Schoenberger, John Snellings, Adrienne Wade, and Joel Young

²<https://github.com/OUhrt/srtn.git>

Contents

| | | |
|----------|-----------------------------------|-----------|
| 1 | Condensed Procedures | 3 |
| 2 | Overview | 4 |
| 3 | SRT Server Computer | 5 |
| 4 | SRT Software | 6 |
| 4.1 | Commands | 6 |
| 4.2 | Git/GitHub | 6 |
| 4.2.1 | Quick Command Breakdown | 6 |
| 4.2.2 | Proper Usage | 7 |
| 4.3 | Setup | 7 |
| 4.4 | Compiling | 7 |
| 4.5 | Files | 9 |
| 4.6 | Command Files | 10 |
| 4.7 | Example Procedure | 11 |
| 5 | Data Reduction | 13 |
| 5.1 | Parsing the Data Files | 13 |
| 5.2 | Gnuplot | 13 |
| 5.3 | Formulas | 13 |

Chapter 1

Condensed Procedures

This list of procedures is extremely condensed and it is expected you are fairly experienced with operating the telescope before using these. If you are unsure of something, ask.

Parameters (Not always needed)

- 1) Edit CALMODE in srt.cat file
- 2) CALMODE 0 - used for yfactor solving. Point to cold sky, cal, place absorber, get tsys and tcal
- 3) CALMODE 2 - only does bandpass correction, okay if no absorber or tree
- 4) CALMODE 3 - if stow looks at absorber like trees.
- 4) Verify target frequency and radial velocity
- 5) If velocity is $>50\text{km/s}$, use radvel.py to adjust the frequency if needed.
- 5) **Keep frequency in range 1300-1550MHz unless you have cleared it with Tobin.**
- 6) Input the Tcal (temperature of the absorber) into the srt.cal file.

Higher frequencies have a different instruction set and improper use can fry the receiver.

Pointing

- 1) Slew to very bright source (sun)
- 2) Use npoints to correct and input offset

Simple Cal (with absorber)

- 1) Slew to cold sky
- 2) hit cal on calmode 20
- 3) place absorber/or turn on noise-diode
- 4) should have flat bandpass

Simple Cal (without absorber)

- 1) Stow
- 2) hit cal on calmode 2
- 4) should have flat bandpass

Source

- 1) Reopen srtn (if you modified the frequency)
- 2) Slew to target and record
- 3) Hit the beamsw (Only for certain faint faint signals)
- 4) Wait for at least ~ 45 iterations (counter in bottom spectrum window)
- 5) re-hit beamsw to turn it off/ hit record to stop recording
- 6) This target is finished

End of Observation

- 1) Always stow at end of the observation
- 2) Exit the SRTN program, do **NOT** turn off the computer
- 3) Turn off the Controller, Receiver, and Tuner
- 4) Move data files to appropriate directories

Chapter 2

Overview

The capabilities of the SRT are as follow:

- Frequency Adjustments from 1420MHz
- Spectral Line Gathering
- Continuum
- Drift Scans
- Point Maps

| | |
|------------------------------|-----------------|
| Aperture | 3.0 meters |
| Frequency Range | 1420 – 1470 MHz |
| Mechanical Pointing Accuracy | 0.2 Degree |
| Software Pointing Accuracy | 0.5 Degree |
| Beam Width ¹ | 4.92 Degrees |
| Max wind speed | 120 km/h |

Chapter 3

SRT Server Computer

First know that the server computer, the SRTN software, and all of the following commands are assuming you are on a Linux OS. The SRTN software and the VNC software are available for other OS architectures but are not covered in the scope of this Manual. You will also have to download and install the VNC software¹.

Understand that access to the telescope is a privilege that is administered by Dr. Tobin² and any abuse of the system will result in that access being revoked.

The Nielsen Hall SRT is controlled through a network locked, local server. This server does not follow the standard authentication protocol followed by the University nor the department. In order to gain access to the telescope. You will have to populate and send your ssh public keys to Dr. Tobin. Follow these commands to do this:

```
1 # First verify you have ssh keys generated, or generate them
2 ~$ ls ~/.ssh/
3 # There should be a file named id_rsa.pub. If there is not a directory or not this file then use this
   command
4 ~$ ssh-keygen
5 # specify the path to download, typically in /home/username/.ssh/
6 # type in a password to use to unlock the file or leave blank
7 ~$ cp ~/.ssh/id_rsa.pub ~/id_rsa_NAME.pub # copy the file to your home directory with your name
```

Now email this file to Dr. Tobin or Nick Reynolds and wait until he confirms that he has added your public key to the server computer. Once you have access you can now login to the telescope computer. Please understand how the SRTN software works **before** controlling the telescope. It is also suggested to have a senior member of the group assist in the first run. To login to the computer use these commands:

```
1 ~$ vncviewer -Shared -ViewOnly -via jjtobin@reber.nhn.ou.edu reber:1 # Viewing the telescope
2 ~$ vncviewer -Shared -via jjtobin@reber.nhn.ou.edu reber:1 # Controlling the telescope
```

Since you are just starting out, use the command with the `-ViewOnly` flag just to see what the server looks like.

You will be prompted for a VNC server password which is `SRT1420MHz`.

To walk you through the commands: VNC is the program used that uses ssh protocol to provide a graphical interface between two computers, `vncviewer` is the command to view an existing VNC server on a host computer, `-Shared` allows multiple users to access the server at once, `-ViewOnly` does not send keyboard or mouse controls to the hosting server, `-via` specifies to route to the domain and specific server, `jjtobin@reber.ou.edu` is the domain under which the server presides, `reber:1` designates the computer and server window. There are probably a few VNC servers operating on the server computer at one time and can be listed with `vncserver -list`. However, unless you are the one that created those windows, please do not close them as they might be in use.

Once you have the display window open, you can now begin your observation. First, open up the Firefox session by either toggling the Icon in the application menu or by opening a Terminal window and issuing `firefox` into the prompt. This should open up several windows which are crucial for observations. The windows are: IP camera to display the telescope, the IP power supply which controls the power to the telescope/controller/noise generator, the SRT website, the Discord chat service, and the Intellicast website to view current weather conditions in Norman, OK.

To turn on the telescope, power first must be supplied to the telescope. Go to the Firefox window and to the IP power supply tab. Scroll down and toggle the socket icon next to the Telescope/Controller panel. You can now start the SRT software.

Make sure the Noise Generator is toggled off currently. We will revisit this later in order to properly calibrate the system.

“With power comes great responsibility”– Uncle Ben

¹<https://tigervnc.org>

²jjtobin@ou.edu

Chapter 4

SRT Software

Do not change the code of the SRT software on the SRT controller computer until you have thoroughly tested the code and passed it by the senior SRT operator. Only a single SRTN program can be running at any one time with a few exceptions noted below. The next few subsections are for trying out your own version of the software on your personal computer. However, you will need to create an account on the REBER computer in order to use the telescope. In order to do this, run the command `makeusr -help`, which will display the parameters for making a new account. Or you can simply run `makeusr` and follow the in-terminal prompts. The following message “Finished with all. Exiting program” should appear and your directory can be found in `~/srtn/`, fully populated. If any errors occurred, please contact the local operator.

Two versions of the code are currently in use and I will very briefly describe the differences. Version 5 is the more stable of the two programs however, it has a 1 degree tracking increment. Compared to the beam size of 5 degrees, this can yield a fairly large error if accurate position tracking is needed. Version 6 allows for 0.5 degree tracking, however, some issues with commands not passing through correctly have been noted. In order to work around this issue, simply close and reopen the program.

Multiple directories probably already exist on the computer. Most of these directories will be personal ones and please respect the integrity of other people’s work. There exist two simulation 5/6 directories which are for public use and can be used simultaneously with the main srtn program. These are for determining observation plans for later times and they can be used to simulate the positions of objects at any time.

4.1 Commands

For extended help on any commands, within the terminal window, you can type `man [COMMAND]` e.g. `man tar` and it will display the internal manual for the given command, tar in this case. You can also simply pass `-help` at the end of the command for a quick help section on how to execute the command.

```
1 cd # changes directory to home directory ~/ or if given input e.g.
2 cd ~/Downloads/ # will move to Downloads directory in home
3 ls # will list all of the files in the current director
4 mv /path/file1 /path/file2 # will move file1 to file2
5 mkdir srt # makes a new directory called srt, flag -p will copy directory structure
6 tar # (un)compression algorithm, x is uncompress, z is gunzip format, v is verbose, f specifies file
```

4.2 Git/GitHub

4.2.1 Quick Command Breakdown

This is meant for just a quick overview of Git/GitHub commands. Please understand the commands before using them.

```
1 new repository
2 ->git init :: initialize local repository inside the repository
3 ->git status :: list git files in current directory, tracked and untracked
4 ->git log :: shows a log of all new commits
5
6 new file
7 ->git add $1 :: $1 is name of file, adds the file to the staging environment. Think like indexing
8 ->git commit -m $1 :: $1 is the comment for the commit. Should be related to the commit.
9
10 new branches
11 ->git checkout -b $1 :: $1 is new branch name. Branching allows working on other projects without
    affecting master
12 ->git branch :: lists known branches and * tells which branch you are in
13 ->git checkout branch :: moves you to the branch defined
14
15 online
```

```

16 ->git remote add(rm) origin $1 :: $1 is website name. add will add $1 to the origin variable while rm
    will remove the variable
17 ->git push -u origin branch :: pushes the local branch repo to the origin branch repo
18 ->git revert $1 :: $1 is the hash code number. Command is used to revert the branch
19 ->git pull origin branch :: pulls the changes in the branch to your local. Prefer to use fetch and
    merge
20 ->git fetch origin :: remotely stages origin to merge locally
21 ->git merge origin/branch :: remotely merges the branch from origin to your current branch
22 ->git request-pull v1.0 origin master :: initiate a pull request upstream from your master branch to
    their master branch
23 ->git clone $1 :: will clone the specified online directory

```

4.2.2 Proper Usage

To properly use GitHub in collaboration with other colleagues please follow this guide.

To implement changes to the repository, you will have to make an account and request access online. Once this is finished, you will not have to check the website again. If you do not already have a directory you wish to use to contribute, make a directory now. You can use a pre-existing directory. Once in this directory, run the command `git init` to initialize the current directory.

To pull changes from the website, you first have to set the variable `origin` using command `git remote add origin $1` and input the name of the repository¹. Then use `git fetch origin` to remotely pull the most updated, official code from the GitHub page. You then merge the most up to date version into your local master with `git merge origin/master`.

You can then create local branches to work and merge projects using `git checkout -b $1` to create the branch and `git checkout $1` to move to a branch. The command `git branch` will tell you the list of current branches within the initialize directory you are in. Make the changes you wish to make and then issue the index command `git add $1`. This prepares the file for the commit stage. Once you are satisfied with all of the indexed files, commit the changes using `git commit -m "$1"` where \$1 in this case is a short comment on what this change is.

Once committed push the changes to the online repository with `git push -u origin branch`. If an error comes up regarding unable to push, probably one of two things happened: incorrect spelling of command/branch or someone else has since updated the branch. Like I said, you will be unable to push changes to the GitHub master branch so push changes to another branch and initiate a pull request.

4.3 Setup

First download the source code for the SRT either by downloading directly from the GitHub² or by using the command on line 3.

```

1 ~$ mkdir -p ~/srt/scripts/
2 ~$ cd ~/srt/
3 ~/srt$ git clone https://github.com/OUsrc/srtn.git # can ignore this line if you directly downloaded
4 ~/srt$ cp -r ./srtn/software/srtnver5_reber ./
5 ~/srt$ cd srtnver5_reber
6 ~/srt/srtnver5_reber$ ls

```

You should now be in the SRT main code directory and the contents should be on your screen. The code should be able to run immediately or recompiled as necessary.

4.4 Compiling

If there are issues with the C compiler not able to locate certain libraries, try searching for solutions to that problem or send me an email³. There should be a file called `srtnmake` within the directory. This is the main compiler file that will gather the other C files and compile them in the correct order. When you make changes to the other files within the directory (other than the `srt.cat` file) you will have to recompile the program so it can implement the changes.

```

1 ~/srt/srtnver5_reber$ ./srtnmake

```

¹In this case, the repository is <https://github.com/OUsrc/srtn.git>

²<https://github.com/OUsrc/srtn.git>

³nickreynolds@ou.edu

You will probably get a lot of warnings, this is fine and normal errors can look like this:

```
1 main.c: In function 'main':
2 main.c:62:12: warning: variable 'secstart' set but not used [-Wunused-but-set-variable]
3 main.c:60:12: warning: variable 'ii' set but not used [-Wunused-but-set-variable]
4 main.c:59:14: warning: variable 'color' set but not used [-Wunused-but-set-variable]
5 main.c: In function 'gauss':
6 main.c:555:16: warning: variable 'j' set but not used [-Wunused-but-set-variable]
7 vspectra_four.c: In function 'vspectra':
8 vspectra_four.c:33:28: warning: variable 'min' set but not used [-Wunused-but-set-variable]
9 vspectra_four.c:33:12: warning: variable 'avsig' set but not used [-Wunused-but-set-variable]
10 vspectra_four.c:31:43: warning: variable 'r' set but not used [-Wunused-but-set-variable]
11 disp.c: In function 'clearpaint':
12 disp.c:440:18: warning: variable 'update_rect' set but not used [-Wunused-but-set-variable]
13 outfile.c: In function 'outfile':
14 outfile.c:16:16: warning: variable 'n' set but not used [-Wunused-but-set-variable]
15 sport.c: In function 'rot2':
16 sport.c:402:25: warning: variable 'i' set but not used [-Wunused-but-set-variable]
17 sport.c:402:17: warning: variable 'status' set but not used [-Wunused-but-set-variable]
18 sport.c:408:15: warning: ignoring return value of 'system', declared with attribute warn_unused_result
   [-Wunused-result]
19 cal.c: In function 'cal':
20 cal.c:18:24: warning: variable 'ixe' set but not used [-Wunused-but-set-variable]
21 srthelp.c: In function 'display_help':
22 srthelp.c:39:14: warning: variable 'color' set but not used [-Wunused-but-set-variable]
23 srthelp.c: In function 'load_help':
24 srthelp.c:254:10: warning: ignoring return value of 'fgets', declared with attribute
   warn_unused_result [-Wunused-result]
25 srthelp.c:258:14: warning: ignoring return value of 'fgets', declared with attribute
   warn_unused_result [-Wunused-result]
26 velspec.c: In function 'velspec':
27 velspec.c:19:14: warning: variable 'color' set but not used [-Wunused-but-set-variable]
28 velspec.c: In function 'vplot':
29 velspec.c:162:15: warning: variable 'jmax' set but not used [-Wunused-but-set-variable]
30 librtlsdr.c: In function 'rtlsdr_open':
31 librtlsdr.c:1267:13: warning: variable 'rt' set but not used [-Wunused-but-set-variable]
32 tuner_r820t.c: In function 'R828_RfGainMode':
33 tuner_r820t.c:2859:11: warning: variable 'LnaGain' set but not used [-Wunused-but-set-variable]
34 tuner_r820t.c:2858:11: warning: variable 'MixerGain' set but not used [-Wunused-but-set-variable]
```

Now, within the same directory should be a file called `srtm` . This is the file that will run the GUI window for the telescope. To run it, simply type

```
1 ~/srt/srt_ver5$ ./srtm
```

and a GUI window should pop up that looks like this:

If you do not have both the receiver dongle and the telescope controller plugged into the computer, the program will fail to initialize. In order to get around this, for testing purposes, go into the `srt.cat` file and edit the lines from

```
1 *SIMULATE ANTENNA —> SIMULATE ANTENNA
2 *SIMULATE RECEIVER —> SIMULATE RECEIVER
```

This will allow you to open the program and test certain commands before committing commands to the telescope untested.

4.5 Files

Help on the various C code files in the source directory.

- 60-mcc.rules: udev rules that allows PCI, HID devices, and libusb to work with non-root users
 - acml.h: allows calling ACML routines via their C or FORTRAN interfaces
 - amdfft.c: fast-fourier transform adaptation software
 - calc.c: calibration code
 - cat.c: parses the srt.cat file
 - cmd.txt: the default file for the commands you want srtn to use
 - cmdfl.c: the code to interpret and execute cmd.txt
 - d1cons.h: declares values used in code
 - d1glob.h: globally declared variables for the code
 - d1proto.h: also declares variables for code
 - d1typ.h: declares a new struct
 - disp.c: code for the GUI
 - doindent2: code for parsing whitespace
- example_cmdfile: example command file for srtn
 - fftw2.c: fast-fourier transform
 - fftw3.c: fast-fourier transform
 - four.c: Handling the different coordinates to angles conversion
 - geom.c: for parsing the spherical geometries
 - hproto.h: help menu functions
 - librtlsdr.c: self consistent library for the dongle
 - main.c: the main C file for calling and compile the code
 - map.c: Deals with the GUI map controls and drawing.
 - outfile.c: writes the data to a file
- PCI-DAS4020-12.1.21.tgz: library for PCI boards
 - PCI-DAS4020.h: functions for the PCI boards
 - plot.c: plotting the spectra in the GUI
- README: basic readme file
- rtk-sdr.rules: udev rules to allow external rtl-sdr devices to work with non-root users
- rtl-sdr_export.h: packing the librtlsdr
 - rtlsdr-i2c.h: declaring variables from librtlsdr
 - sport.c: controls the antenna movement
 - srt.cat: the main config file
 - srt.hlp: the help file in the gui
 - srt help.c: displays the Help menu in the GUI and compiles the srt.hlp file
 - srtn: the program
 - srtnmake: used to compile the program
 - time.c: important for keeping track of timing
- tuner_r820t.c: the driver for the tuner
- tuner_r820t.h: declare variables for the tuner
 - velspec.c: plots the spectrum in the gui
- vspectra.c: inputs the data from the dongle
- vspectra_fftw.c: handles the data from the dongle
- vspectra_four.c: handles the data from the dongle
- vspectra_pci.c: also inputs data compatible with PCI boards
- vspectra_pci_fftw.c: handles the data from the dongle, compatible with PCI boards

4.6 Command Files

It is suggested to use input command files as this will add speed the entry of SRT commands as well as reduce command entry mistakes. By default the srtn software provides the syntax for setting up a cmd file if you mouse over the cmdfl button in the srtn software. The command file is ASCII formatted and can: accept instruction lines (active lines), blank lines (ignored), comment lines (begins with *, ignored), and comment sections (begins with /, ignored).

Instruction line: Must start with a time mark (either UT or LST) or a colon (current time).

For Example:

```
2005:148:00:00:00 (yyyy:ddd:hh:mm:ss (UT))
23:00:00 06:50:00 ra/dec
LST:06:00:00 (LST:hh:mm:ss)
:120 azel 120 30 (sss azel position)
```

Rules:

```
: cmd /execute the command and proceed to the next line
:120 cmd /execute the command and wait 120 seconds, taking data, before proceeding to the next line in the schedule
:120 /wait 120 seconds, taking data, before proceeding to the next line. This is a convenient way of increasing the
integration to more than one scan.
```

*Note: There is NO space allowed between the colon and a time “wait” command

```
LST:06:00:00 /wait until LST 06:00:00, taking data, before proceeding to the next line
2005:148:00:00:00 /wait until UT= yyyy:ddd:hh:mm:ss, before proceeding to the next line
```

Example Set: Command the SRT to take 1420.4 MHz hydrogen spectra in 5 degree spacing along a section of the galactic equator. The user must start data recording, unstow the telescope, calibrate the receiver, set the observing frequency center and frequency scan and then repeat the spectral line observations for ten points along the equator. Note: Allow a space between the colon and the command.

```
: record rotation.rad /(Start data recording of file rotation.rad)
: galactic 206 20 /(unstow and move to calibration position)
: freq 1419 /(Off-hydrogen calibration frequency)
: calibrate
: freq 1420.4 4 /(Set center frequency mode 4)
: galactic 205 0 /(Move to first data point)
: galactic 210 0 /(Next point)
: galactic 215 0
: galactic 220 0
: galactic 225 0
: freq 1419 /(Off-hydrogen calibration frequency)
: galactic 225 20 /(Calibration point)
: calibrate
: freq 1420.4 4 /(Set center frequency mode 4)
: galactic 230 0 /(Move to sixth data point)
: galactic 235 0 /(Next point)
: galactic 240 0
: galactic 245 0
: galactic 250 0
: roff /(End data recording)
```

If this input command file were named galactic.cmd, the user could initiate this observation by clicking in the command text box: galactic.cmd The SRT will read each line in turn and report the current line read as green text in the message board area. If, for example, the start time was 1400 Universal Time on March 15, 2002; and no output file name was entered, the default OUTPUT file would automatically be written and labeled 0214814.rad. Where the file label is: “yydddhh.rad”

4.7 Example Procedure

Calibrate the telescope:

First thing you will have to do before any observing will be to calibrate the telescope. Your intended source will determine the calibration procedure and the subsequent method of observing.

To transform your relative flux into absolute flux, you need to do a careful calibration. You can do this by pointing to a radio quiet patch of the sky and press the cal button. This will first sample this “cold load” source. The software will then prompt you to place the absorber in the feed. You can either turn on the noise source⁴ now by toggling the button in the IP Power supply Firefox window or you can go to the roof and physically place the carbon-impregnated foam absorber in front of the feed.

You will have to do this for roughly 10 seconds. Make sure you do not place anything in front of the antenna or the feed beforehand and that the feed is completely covered or supplied known noise for the full calibration time. Otherwise, your calibration will be off. In the terminal you used to open the srtn program, a line of parameters will appear when the calibration procedure is done. Pay special attention to these parameters.

Example line: `yfac 2.616403 tsys 174.554743 pwr 445.649653 pwrprev 169.683193 calmode 20 max_pwr 1420.362 MHz` . These are the parameters used to calibrate the telescope. You can look at the formula given in section 5.3. The typical tsys value for the telescope is 180 on a hot day and 175 on a cold day. Options are still being explored to better stabilize the telescope components. It is suggested to do several calibrations until you get one that is suitable. A new tsys value is not imposed unless the yfac is >1.5.

Changing frequencies will reset calibrations and pointing offsets.

Some sources have to use “beamsw” others cannot use this method.

For observing the Milk Way HI and for doing the sun radio profile do not use “beamsw” as this will corrupt your spectra. To observe the Sun you will have to adjust the frequency to 1418MHz.

Beam switching is used for faint sources like Cas A, Cygnus A, the Moon. M33 and Andromeda can use the beam switching method but it is currently untested if this will yield good results.

Generally you will not have to adjust the frequency unless doing continuum measurements, (1413MHz is clean of RFI), source is near another bright radio source, the source peaks > 2MHz away from 1420MHz, or the radio source has a high VLSR.

For observing Cas A, you’re going to need to perform the “advanced” calibration procedure.

Make sure that your center frequency is set to something that is OFF THE H1 LINE, 1415.00 MHz works well. Your center frequency must be off the H1 line because Cas A lies pretty much right in the galactic plane, so any H1 emissions would muddy up the Cas A signal.

Take note of your new Tsys and you’re ready to observe!

Observing Cas A:

Once the telescope is calibrated, slew over to Cas A.

Hit record.

Hit the “beamsw” button on the top bar of buttons in SRTN. This will start the beamswitching procedure, which takes the telescope on and off Cas A by a beamwidth on either side. This technique is used to tease out faint signals.

Wait till at least 40-45 beamswitches pass. If this is your only source then observe for as long as possible. There is a counter in the average spectrum (bottom spectrum) window for you to keep track of.

Hit “beamsw” again to turn the beamswitching off.

Hit record again to stop recording.

Stow the telescope and close SRTN (unless of course you aren’t done looking at stuff!).

This method is of course not the template to use for all sources and the methodologies will depend on current conditions, type of source, strength of source, etc.

It is also suggested to leave enough time to take multiple measurements sets for each source.

⁴The noise source temperature hasn’t been accurately determined yet but the current temperature profile works well

For extremely precise measurements, it is suggested to find the beam profile. This can be done by writing a script to map out many points up to 5 beam widths off of the sun and analyze the FWHM. There is code on the GitHub to do just that.

Chapter 5

Data Reduction

5.1 Parsing the Data Files

Parsing and Plotting the Spectra.

- Copy the data from the server computer to your personal computer to run the reduction sequence.
- The command `scp /file/to/move/ /destination/` will help.
- Make sure you acquire the parsing code¹ in order to parse the data.
- For continuum analysis:
 - The metadata parser is called `metaparse.py`. The `metaparse` script lists columns of data in the order: Date, obsn, az, el, center freq, Tsys, Tant, vlsr, galactic latitude, galactic longitude, azoff, eloff, source, fstart, fstop, spacing, bw, fbw, nfreq, nsam, npoint, integ, sigma, bsw.
- For spectrum analysis:
 - The spectrum parser is called `specparse.py`. The `specparse` script lists: frequency, vel, vel-vlsr, and Tant.
 - To run the files, use `python hispec.py -help`. Follow the input parameters or just simply run `python specparse.py`
 - Once you have your spectra parsed out and written to a new file, you can start to plot them all.
 - You can reduce and plot the spectra by running the code `python specplot.py -help`. This code will walk you through the reduction, but to summarize the steps you are flattening the baseline, removing RFI, and then finding the integrated intensity.

5.2 Gnuplot

Understanding, Interpreting, and changing the units of your plots into something useful.

- So the y-axis in your plot will have units of Kelvins. The numbers in the spectrum correspond to the power level measured at a given frequency, and these numbers are given in Kelvins.
- We want to convert Kelvins to Janskys with $kT_a = \frac{S_\nu \times A_{dish}}{2}$, T_a is Antenna temperature. S_{nu} is spectra flux at a certain frequency. A_{dish} , k is Boltzmann's constant. The value you get at this point will be in J/s/m²/Hz, and 1 Jansky = 1×10^{26} , so you'll need to multiply the value you just got by 1×10^{26} , and then you'll have your spectral flux density in Janskys. Now you can compare to established continuous spectra.
- The output of `spec_reduc.py` will put the data in a format that can be directly read into the Gnuplot software.

5.3 Formulas

$$\begin{aligned} \text{Diameter} &= 3 \text{ m} \\ \text{Frequency} &= 1.420401 \text{ GHz} \\ \text{Speed of Light} &= 0.299792458 \text{ giga-m/s} \\ \lambda &= \frac{c}{\nu} \\ \text{Beam Width} &= \frac{70 \times \lambda}{\text{Diameter}} \text{ degrees} \end{aligned} \tag{5.1}$$

¹<https://github.com/OUsrc/srtn.git>

#Solving for tsys

- 1) Follow the Simple Cal procedure
- 2) Move to blank sky patch and record
- 3) Average ~ 5 power measurements (from terminal). This is P_{cold}
- 5) Remove absorber and average 5 power measurements. This is P_{hot}
- 6) $T_{\text{cold}} \sim 3\text{k}$ T_{hot} is average ambient temperature or thermometer measurement of absorber
- 7) Run python script tsys.py to find tsys

(5.2)