Observing with the Allen Telescope Array (ATA)

Pranav Premnath
Wael Farah
Alexander Pollak
Sofia Sheikh

SETI Institute
Allen Telescope Array

February 24, 2022

1 Introduction

The Allen Telescope Array (ATA) at the Hat Creek Radio Observatory (HCRO) consists of 42 antennas, of which 23 have feeds installed in them. 20 of those antennas are connected to the RFSoc boards, and 12 of them connected to older the SNAP boards. We can observe many sources with different sub-arrays (or the entire array) from 1-10 GHz of bandwidth, at any elevation between 16.5^0 and 88^0. This document goes through how to operate the array to point at a source, and use the backend to record data. **Use this link to access the camera**. Please make sure to reset the camera to the default view if you change the view, as it is a shared camera. Remember that you need to be connected to the HCRO network to perform these observations.

2 Using the Control Machine

Before starting an observation, we can log on to the control machine for the ATA (`ssh obs@control -X -Y`). Note that this machine is separate from the different user machines at the ATA, and that you have to be on the VPN or directly connected to the HCRO network. There are many commands that can be run to check the status and various settings of the array, ranging from the LNA settings, to the focus frequency. Remember to parse in X11 when logging in (`-X -Y`), so that you can see any pop-ups and figures.

2.1 ATA Status

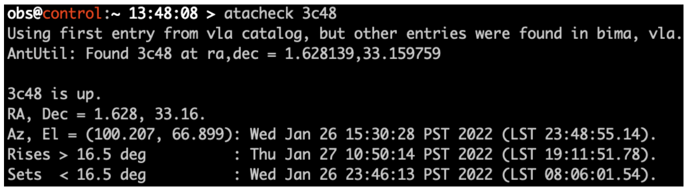
By typing `atastatus` on the command line of the control machine, we can see a display of all the antennas, which ones are currently running, where they are pointing to (Source, Azimuth and Elevation), the given Az/El command and the pointing error. On the bottom panel of the "Antenna Status" window, we can also see the various tunings and the frequencies they are set to.

2.2 Check Source Position

Before you can get ATA-specific observing characteristics for a source, you need to make sure the source is present in the ATA database. You can explore the database with the `atalistcatalog` command, which also allows the user to specify filtering options. For example `atalistcatalog --list` will provide the full list of sourcenames available in the catalog at present.

The command to check the position of the source you wish to study from the ATA's location is then simply `atacheck sourcename`. This command also tells you when the source Rises/Sets in both Pacific Time and LST, as seen in Figure 1. Additionally, if it is a more well known source, we can check the website `radioskyguide` Radioskyguide was designed by Gurmehar Singh. It can be used to track a source through the night, so we can know its azimuth and elevation during any particular time. We can also check its closeness to the Sun and Moon. **Note that observations should not be conducted if the source is within 10^0 of either the Sun or Moon.**

If the source you wish to observe is *not* contained in the catalog, `atacheck` will display a message that says "The source, [name], was not found." In this case, you can a) add the source to the catalog or b) use the sources known RA/Dec to track it in your observing script.



```
obs@control:~ 13:48:08 > atacheck 3c48
Using first entry from vla catalog, but other entries were found in bima, vla.
AntUtil: Found 3c48 at ra,dec = 1.628139,33.159759

3c48 is up.
RA, Dec = 1.628, 33.16.
Az, El = (100.207, 66.899): Wed Jan 26 15:30:28 PST 2022 (LST 23:48:55.14).
Rises > 16.5 deg          : Thu Jan 27 10:50:14 PST 2022 (LST 19:11:51.78).
Sets  < 16.5 deg          : Wed Jan 26 23:46:13 PST 2022 (LST 08:06:01.54).
```

Figure 1: The output of the command line call to check a source contained in the ATA database.

3 Starting the Backend

Currently, we have four different backend modes to record data, the spectrometer mode, voltage mode, and the correlator mode, and the beamformer mode. The data products are discussed further in the Recording Data section, but these commands will help you start the backend for the mode you would like to use. Note that the beamformer mode is currently under development.

3.1 Spectrometer Mode

```
bash /home/sonata/src/ata_snap_rfsoc/program_rfsoc_spec.sh
```

3.2 Voltage Mode

```
ansible-playbook
/home/sonata/src/ansible_playbooks/hashpipe/voltage_record.yml -K
```

3.3 Correlator Mode

```
ansible-playbook
/home/sonata/src/ansible_playbooks/hashpipe/xgpu_record.yml -K
```

3.4 Beamformer Mode

```
ansible-playbook /home/sonata/src/ansible_playbooks/hashpipe/beamformer_record.yml -K
```

4 Controlling the Array

To conduct an observation, the user must log on to the VNC session that we use for the ATA. Below, we list the current VNC session that is online.

```
obs-node1:5904
```

The VNC session has multiple tabs open, from the `atastatus`, the pipeline monitor, terminals with the delay engine running for each LO, the control machine, and two terminals with `obs-node1`; one to run your observing script, and the other to record data if the command is not included in the former. By pressing `Ctrl-Cmd` followed by an arrow, we can go to another window. We have a window dedicated to keeping the observing log, displaying the camera and `radioskyguide`. Another window has `CASA`, a python based data reduction software, open. The final window is free to use to run any scripts necessary with your data, as long as you don't interrupt another user. Note that you should only use the VNC session if you are the authorized observer at the time.

4.1 Choose the Antennas for the Observation

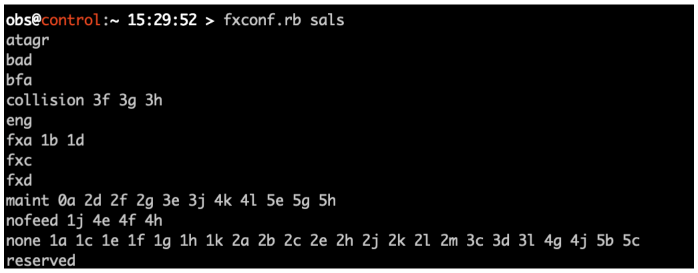
First, we must think about which antennas we want to use. Back in the control machine the command `fxconf.rb sals` lists which groups the antennas belong in, as seen in Figure 2. The group *none* contains the antennas which have not been assigned for any observation at the time, and have feeds capable of performing observations in them. We can then create a list of the antennas to be reserved for this observation.

```
ant_list = ["1c", "1g", "1h", "1k", "1e", "2a", "2b", "2c",
           "2e", "2h", "2j", "2k", "2l", "2m", "3c", "3d",
           "3l", "4j", "5b", "4g"]
```

.

4.2 LO and LO frequency

We can then select which Local Oscillator (LO) to use. This can enable us to select different frequencies for the array if we choose multiple LOs. Note the capitalization conventions: capital "LO" while calling the `antlo_list` and lower case "lo" when setting the frequency using `ata_control.set_freq`. In the future, we plan to make it such that either of them will be case-insensitive.



```
obs@control:~ 15:29:52 > fxconf.rb sals
atagr
bad
bfa
collision 3f 3g 3h
eng
fxa 1b 1d
fxc
fxd
maint 0a 2d 2f 2g 3e 3j 4k 4l 5e 5g 5h
nofeed 1j 4e 4f 4h
none 1a 1c 1e 1f 1g 1h 1k 2a 2b 2c 2e 2h 2j 2k 2l 2m 3c 3d 3l 4g 4j 5b 5c
reserved
```

Figure 2: The various groups the antennas are placed in. *bfa* stands for "beam former A" and is used as the group for reserved antennas for observations. *none* are antennas free to be used and have feeds in them. *no feed* are antennas that are retro-fitted, but don't have feeds in them yet.

LOb and LOc are connected to the RFSoc boards, and LOa is connected to the SNAP boards.

4.3 Reserve and Release Antennas

The next step is to reserve the antennas we selected before. This will throw an error if any specified antenna is already reserved for some other observation. Check the control machine once again if this happens.

Another important step is to use the `register` function from the `atexit` module to ensure that the observing script releases the reservation of the antennas when the observation concludes. This module will activate even in the case of an error/abort of the observing script, thus making it a more reliable way of releasing control than putting a statement at the very end of the script. If you optionally wish to park the antennas after the script is run, you can change the Boolean in the `register` function to `True`. Note that if you are recording data independent of the script, i.e., you are pointing the antennas with one script and recording with another), you don't want to park the antennas in the pointing script, as it will execute the park command prior to your data being recorded.

```
ata_control.reserve_antennas(ant_list)
atexit.register(ata_control.release_antennas, ant_list, False)
```

4.4 Set Frequency for different LOs for the Sub-Arrays

Once we know which LO we want to use, and at what frequency we want to observe at, relay it to the antennas. This can be done with mutiple LOs, by calling the same argument below for each LO and frequency that you want to observe at.

```
ata_control.set_freq(freqs, ant_list, lo=’b’)
```

4.5 Choose and Track Source, RA/Dec or Az/El

Depending on the type of observation, we need to tell the antennas where to point. If we have a particular source in mind, then we can define it as a string, as long as it is present in the ATA database.

```
source = ‘‘something’’
ata_control.make_and_track_ephems(source, ant_list)
```

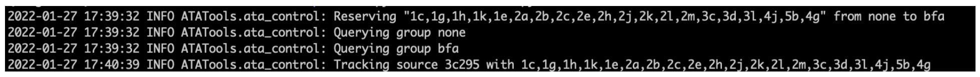
Additionally, if the source is not present in the database, we can specify the Right Ascension (RA) and Declination (Dec) of our target.

```
ata_control.make_and_track_ra_dec(ra, dec, ant_list)
```

On the other hand, we can also set the azimuth and elevation manually, which would be useful for geostationary satellite observations, as well as survey modes.

```
ata_control.set_az_el(ant_list, az, el)
```

Figure 3 show the output of a typical observing script, post reserving antennas. We will see them move from the *None* group to the *bfa* group for this observation. Once that is done, the antennas will start tracking the source, slewing to the position and then actively tracking the ephemeris file.



```
2022-01-27 17:39:32 INFO ATATools.ata_control: Reserving "1c,1g,1h,1k,1e,2a,2b,2c,2e,2h,2j,2k,2l,2m,3c,3d,3l,4j,5b,4g" from none to bfa
2022-01-27 17:39:32 INFO ATATools.ata_control: Querying group none
2022-01-27 17:39:32 INFO ATATools.ata_control: Querying group bfa
2022-01-27 17:40:39 INFO ATATools.ata_control: Tracking source 3c295 with 1c,1g,1h,1k,1e,2a,2b,2c,2e,2h,2j,2k,2l,2m,3c,3d,3l,4j,5b,4g
```

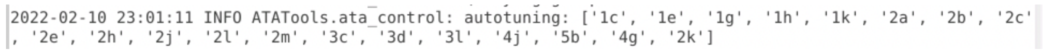
Figure 3: The output of the obaserving script once the antennas are reserved and they start pointing towards the source position

.

4.6 Autotune and IF Tune

“Tuning” refers to adjusting the attenuation at different stages of the signal pipeline to ensure a power level with a suitable gain and dynamic range. We have two different functions for tuning, which occur in two different places in the pipeline. Once the antennas are pointed at the source, we autotune the array, and a message like Figure 4 should appear. The `autotune` function tunes the power across the entire bandwidth at the PAX boards (in the field, in an aluminum box behind the feed).

```
ata_control.autotune(ant_list)
```

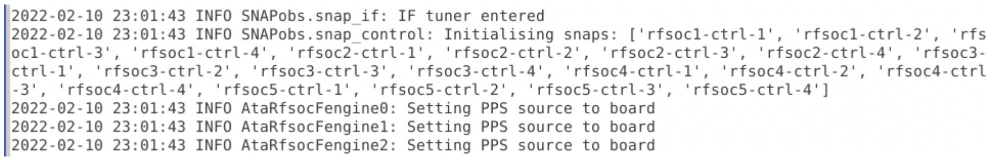


```
2022-02-10 23:01:11 INFO ATATools.ata_control: autotuning: ['1c', '1e', '1g', '1h', '1k', '2a', '2b', '2c'
, '2e', '2h', '2j', '2l', '2m', '3c', '3d', '3l', '4j', '5b', '4g', '2k']
```

Figure 4: Indication that Autotuning of the array has started

The `tune_if_antslo` function tunes the power for each LO individually at the IF stage, in the signal processing room (near digitization). This performs a tuning for a single LO (across $\sim$ 700 MHz), which will be different at different positions across the band. We should see an output like Figure 5 once the IF tuning has started.

```
snap_if.tune_if_antslo(antlo_list)
```



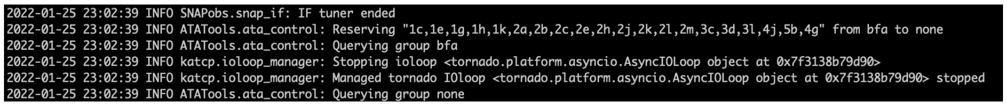
```
2022-02-10 23:01:43 INFO SNAPobs.snap_if: IF tuner entered
2022-02-10 23:01:43 INFO SNAPobs.snap_control: Initialising snaps: ['rfsoc1-ctrl-1', 'rfsoc1-ctrl-2', 'rfs
oc1-ctrl-3', 'rfsoc1-ctrl-4', 'rfsoc2-ctrl-1', 'rfsoc2-ctrl-2', 'rfsoc2-ctrl-3', 'rfsoc2-ctrl-4', 'rfsoc3-
ctrl-1', 'rfsoc3-ctrl-2', 'rfsoc3-ctrl-3', 'rfsoc3-ctrl-4', 'rfsoc4-ctrl-1', 'rfsoc4-ctrl-2', 'rfsoc4-ctrl
-3', 'rfsoc4-ctrl-4', 'rfsoc5-ctrl-1', 'rfsoc5-ctrl-2', 'rfsoc5-ctrl-3', 'rfsoc5-ctrl-4']
2022-02-10 23:01:43 INFO AtaRfsocFengine0: Setting PPS source to board
2022-02-10 23:01:43 INFO AtaRfsocFengine1: Setting PPS source to board
2022-02-10 23:01:43 INFO AtaRfsocFengine2: Setting PPS source to board
```

Figure 5: Indication that the IF tuner has entered

5 Start Recording Data

We got the array pointed at the target, and the antennas tuned accordingly. The output of the script should look like Figure 6, indicating the script has ended, and the data recording process is contained in another script. You will also see this output once the data recording has finished.

We can finally record some data once the tunings are done. Depending on the different modes available, we can set the boards to record data.



```
2022-01-25 23:02:39 INFO SNAPobs.snap_if: IF tuner ended
2022-01-25 23:02:39 INFO ATATools.ata_control: Reserving "1c,1e,1g,1h,1k,2a,2b,2c,2e,2h,2j,2k,2l,2m,3c,3d,3l,4j,5b,4g" from bfa to none
2022-01-25 23:02:39 INFO ATATools.ata_control: Querying group bfa
2022-01-25 23:02:39 INFO katcp.ioloop_manager: Stopping ioloop <tornado.platform.asyncio.AsyncIOLoop object at 0x7f3138b79d90>
2022-01-25 23:02:39 INFO katcp.ioloop_manager: Managed tornado IOloop <tornado.platform.asyncio.AsyncIOLoop object at 0x7f3138b79d90> stopped
2022-01-25 23:02:39 INFO ATATools.ata_control: Querying group none
```

Figure 6: The expected output once the script is run

5.1 Spectrometer Mode

In the spectrometer mode, we use the module `snap_dada` to record data. We can specify the observation time and the accumulation length. Typically, this command records data to `/mnt/buf0/obs` in the machine you run it from. However, that can be changed with a system call. The data is recorded in the `Filterbank` format. One such library that you can use to read the filterbank file is **Sigpyproc.**

```
obs_time = 300
utc = snap_dada.start_recording(antlo_list, obs_time,
disable_rfi=True, npolout=1, acclen=120)
```

5.2 Correlator Mode

In the correlator mode, we record data directly as a UVH5 file. We can also use the delay engine for these observations. The new RFSoc correlator for the ATA operates with 672 MHz of effective bandwidth.

Once the delay engine starts and enters the while loop, we can start recording data on the source. The nodes to which the data is recorded to must be specified (-H), along with the observation time in seconds (n). Note that this can be done independently or within your observing script. Be sure to check the pipeline monitor to see if all nodes are recording data and that not many observation samples are being dropped by the nodes.

```
start_record_in_x.py -H 1 2 3 4 5 6 7 8 -i 20 -n 300
```

5.3 Voltage Mode

In the voltage mode, we record voltages directly to the disk, as *raw* files. We record data with the delay engine and similar commands to the correlator mode.

6 Sample observation script

As a summary to the above, a typical observing script can be found in **this link.** Additionally, the GitHub repository for all the tools we use to observe with the ATA can be found **here.**