# A Week in the Life of an I-LOFAR Chief Observer

This is a more simplified version of some of the pages David made and is meant to on board new chief observer's (CO) onto. It details the pipeline used to turn voltages to filter-banks, clean and search them before archival on DIAS or Archie.

## Scheduling

`ilofar@LGC:/home/ilofar/scheduling/archive/old_sched/`

1. 
2. `sched`
3. `cd tmp`
4. `bash test_schedule.sh your_schedule.sh`
5. `mv your_schedule.sh ../`

```
2024-01-23T08:01 - 2024-01-23T08:24 : J1541+47 [4.106468842, 0.821206502, 'J2000']
2024-01-23T08:25 - 2024-01-23T11:09 : J1836+51_PSR [4.877558323, 0.906481216, 'J2000']
2024-01-23T11:10 - 2024-01-23T11:56 : J1931+4229 [5.110286485, 0.741570367, 'J2000']
2024-01-23T11:57 - 2024-01-23T12:55 : J2153+44 [5.730934043, 0.78444308, 'J2000']
The STOPSTART command on the next line will pause observations for this block.
2024-01-23T15:05 - 2024-01-23T18:14 : J0103+54 [0.274724653, 0.941371287, 'J2000'] STOPSTART
2024-01-23T18:15 - 2024-01-23T19:23 : J0220+3622_PSR [0.613937673, 0.636146372, 'J2000']
2024-01-23T19:24 - 2024-01-23T20:06 : J0355+28_PSR [1.027046438, 0.499947595, 'J2000']
2024-01-23T20:07 - 2024-01-24T00:17 : J0746+55 [2.036742674, 0.964118391, 'J2000']
2024-01-24T00:18 - 2024-01-24T02:14 : J0939+45 [2.528618476, 0.789761487, 'J2000']
2024-01-24T02:15 - 2024-01-24T05:19 : J1218+47 [3.230943057, 0.824859043, 'J2000']
2024-01-24T05:20 - 2024-01-24T08:19 : J1541+47 [4.106468842, 0.821206502, 'J2000']
2024-01-24T08:20 - 2024-01-24T11:07 : J1836+51_PSR [4.877558323, 0.906481216, 'J2000']
```

```
2024-01-24T11:08 - 2024-01-24T11:51 : J1931+4229 [5.110286485, 0.741570367, 'J2000']
2024-01-24T11:52 - 2024-01-24T13:15 : J2153+44 [5.730934043, 0.78444308, 'J2000']
```

# Disk Space

There is a lack of disk space locally in Birr. By default the observing is stored in `/mnt/ucc1_recording2/data/observations/`. This  is one of the reasons that **UCC1 shouldn't be used during observations**. Keep an eye on the data rates throughout observation, if there is no space then there is no observations. This can be done through the terminal using df -h . Also there is a script setup to email the observing mailing list when the space on the disk drops below 20%.

The data rates for the observations are as follows, this is a good way to see if you have enough space left for the remaining observation time.

| Observation | Data Rate (GB/hr) |
| --- | --- |
| BF-ed HBT | ~550 |
| KBT (Sun) | ~700 |

This means that for a given full day of observation, about 14 TB is needed, **KBT dependent**.

**NOTE:** It is important that the disks remain free, if the disk looks like it will fill before the observations end, please email the CO list.

# Processing

Each week the collected data needs to be processed. This can be started during obs as it takes place on the other UCC nodes. This step should be started at the latest right after observations end. (usually 3pm on Wednesdays).

## Filter-banking & Folding

First things first is to do some folding on the known pulsars that were observed. To begin the processing you will need to log into either ucc3 or 4 as obs.

```
ssh obs@ucc4
```

The next thing to do is setup a tmux session, these sessions are just terminal instances that persist when you close your local terminal.

```
tmux new -t processing
```

Then `cd` to the directory where you want to generate the filterbanks, currently this is done in `/mnt/ucc4_data2/data/David` under a sub folder with the observation date. You will need to make a new directory for the most recent observation run, i.e. `mkdir 2024_02_09`.

Next thing is to load the docker with all the processing software. This is done using dckrgpu command. If successfully loaded you will see the following output.

```
:: initializing oneAPI environment ...
   BASH version = 4.4.20(1)-release
:: mkl -- latest
:: debugger -- latest
:: dev-utilities -- latest
:: ipp -- latest
:: mpi -- latest
:: inspector -- latest
:: compiler -- latest
:: ippcp -- latest
:: vpl -- latest
:: dpl -- latest
:: advisor -- latest
:: tbb -- latest
:: ccl -- latest
:: clck -- latest
:: dpcpp-ct -- latest
:: itac -- latest
:: vtune -- latest
:: oneAPI environment initialized ::
```

Then the `cdmtProc.py` script needs to be copied to current directory (where the observations are stored).

```
cp ../cdmtProc.py
```

This script calls CDMT and digifil and makes shell scripts for non-Solar observations to create filterbanks.

```
for i in {0..4}; do python cdmtProc.py -i /mnt/ucc1_recording2/data/observations/20231114T073100/ --extra
${i}; bash cdmtProc_${i}.sh; python ../generatePrepfolds.py; pushd folds; bash fold_${i}.sh & popd; done
```

You also can add a sleep statement (`sleep $delay`) such that the processing starts when observations end. This takes about the amount of time it took to take the observation (~1:1 processing time).

## RFI Cleaning

This is interactive but fast, you also need X display for this. So when you ssh you need to use the `-X` flag.

```
ssh -X obs@ucc4
cd /mnt/ucc4_data2/data/David/.....
python ../generateHeimdall.py
```

After a minute or two (during which your stdout will show some progress bars zipping along, reading filterbanks), if you have X, this will load up a bandpass plot like below,
Screenshot 2024-02-12 at 16.02.59.png

Some bad channels are automatically identified and marked in red. You need to mark any others that were missed. You can just click on the channel(s). You can cover a range by clicking on the left side of that range and then moving your mouse right and typing ‘a’. This will zap every channel between the click and the point where you clicked ‘a’. Note this works left to right only, not right to left! When you are happy with your zapping just click the x on the window to close it. You will keep getting fed such bandpasses until you’ve gone through all the filterbanks. The output of your clicks are fed into `-kill_chans` flag values for heimdall calls in a script that is generated.

This step is quick and goes and fast as you can select the channels.

## Heimdall Processing

The above generates a script which calls heimdall for every filterbank file, zapping the channels identified. Now you have to actually run this. This is, like step 1, quick to set running, but then you don’t have anything to do for hours as it chugs away processing.

```
tmux new
dckrgpu
cd /mnt/ucc4_data2/data/David/rrat_2023_10_16/
cd cands
bash heimdall_0.sh
<detach from the tmux>
```

This processing typically takes ~50% of real time (~1:0.5 processing).

Note: GPUs have limits! You can’t, for instance, be CDMT-ing a bunch of files AND running heimdall both on the ucc4 GPU or one/both will crash and give you memory errors. This is most important when you have a backlog and are not up to date with processing. Remember to run ‘nvidia-smi’ to get a status of what is happening on the GPU. If ucc4 GPU is busy, for example, doing CDMT or something else, then you can run heimdall on ucc3 as follows,

```
cd /mnt/ucc3_data1/data/Evan
mkdir rrat_2023_10_16; cd rrat_2023_10_16
mkdir cands; cd cands
cp /mnt/ucc4_data2/data/David/rrat_2023_10_16/cands/heimdall_0.sh ./heimdall_0_ucc3.sh
sed -i -e s:"/mnt/ucc4_data2/data/David/rrat_2023_10_16/./cands/":"./":g heimdall_0_ucc3.sh
```

Probably worthwhile diff-ing to make sure the ucc3 version was correctly changed etc. Then you just run as above, just on ucc3, like,

```
tmux new
dckrgpu
cd /mnt/ucc3_data1/data/Evan/rrat_2023_10_16/cands
cp /mnt/ucc4_data2/data/David/rrat_2023_10_16/cands/heimdall_0_ucc3.sh .
bash heimdall_0_ucc3.sh
```

## Heimdall Plots

The heimdall candidates can be plotted like so,

```
cd /mnt/ucc2_data1/data/dmckenna/working_cands/; mkdir rrat_2023_10_16
cd rrat_2023_10_16; cp ../frb_v4.py ./
python3.8 ../generateCands.py -l /mnt/ucc4_data2/data/David/rrat_2023_10_16/ -o proc_ucc4_1.sh; bash
proc_ucc4_1.sh
```

You will need to sudo all the above commands. If any files don't get made just re-run the last line of commands,

```
python3.8 ../generateCands.py -l /mnt/ucc4_data2/data/David/rrat_2023_10_16/ -o proc_ucc4_1.sh; bash
proc_ucc4_1.sh
```

Two sets of plots are generated: _cands are the narrow search set, and _full_cands are the wide search set (10 - 500 pc/cc), the wide search set always flag any candidates near the cDM variable in the file names.

## 4-Polarisation and Solar Processing

`screen -ls`

```
20240227T070000-4pol: 1 windows (created Wed Feb 28 15:00:00 2024)
20240227T070000-sun: 1 windows (created Wed Feb 28 15:00:00 2024)
```

These scripts live in the following directories:

- Non-solar Processing: `/mnt/recording1_2/data/rrats`
- Solar Processing: `/mnt/recording1_2/data/sun`

To restart a failed processing session run the following commands.

```
runPipelines.sh /home/ilofar/scheduling/work/20241105T180600/ENVs_sched_2024_11_05.log 1 0
```

That would run solar only. The zero is 4pol. You can do "1 1" if you want to do both or "0 1" just for 4pol.

## Shipping the data off-world (Archie and DIAS)

If all the filterbanks have been formed AND the 4-pol processing has completes AND the solar processing has completed, then you can backup and delete the relevant voltages. Typically we back up the Crab voltages and delete the rest but for any given project we might want to keep some voltages, e.g. for LOFAR-wide projects doing scintellometry/VLBI we certainly need to keep the voltages.

We back the this data to the archive based in DIAS and our tape archive based in TUS Athlone.

---

Revision #16
Created 6 February 2024 13:54:13 by Owen
Updated 15 November 2024 20:27:30 by Owen