# DARA Receivers Exercise

Mike Jones & Jamie Leech

May 2021

## 1 Introduction

In this exercise we will analyse some data from a small radio telescope. You will transform the data from time domain to frequency domain, calibrate it to remove the instrumental gain variations and put the data in to physical units, measure the system temperature of the instrument, and then reduce an observation containing Galactic HI signals.

## 2 The telescope

The telescope (Figure 1) consists of a rectangular horn antenna, fitted with a low-noise amplifier. After a band-defining filter and two more gain stages, the signals are fed in to an RTL-SDR dongle (Figure 2). This is a commercial device intended to provide digital TV and radio to personal computers. In fact it is a very powerful software-defined radio, i.e. a device which provides real-time radio data that can be analysed in the computer using software. In this exercise, we have captured some data for you to analyse off-line.

Figure 3 shows the functions of the two chips that make up the RTL-SDR dongle. The E4000 tuner chip has a programmable gain stage, then down-converts the data using a synthesised signal source. The RTL2832 samples this data at a fixed sample rate of 28.8 MHz, and then digitally downconverts and re-samples the data. The USB interface that sends the data to the computer can only handle a much lower data rate, up to about 2.5 M samples/sec. The data are thus filtered to a smaller bandwidth and decimated (under-sampled) to send to the computer.

## 3 Data

You are provided with three data files, `dara_sky.dat`, `dara_load.dat` and `dara_galaxy.dat`. These all have identical format and consist of complex data samples, sampled at 2.4 M samples/second. There are 24,000,000 samples in each file, corresponding to exactly 10 s of data each. This number is also useful as it is a multiple of 512. The receiver was tuned to exactly 1420.0 MHz. For `dara_sky.dat`, the telescope was pointed to a place in the sky off the galactic plane, where no strong HI signal is expected. For `dara_load.dat`, a large piece of microwave absorber was placed across the antenna, completely blocking the sky signal. For `dara_galaxy.dat`, the telescope was pointed at the Galactic plane.

Figure 1: The horn antenna and mount.

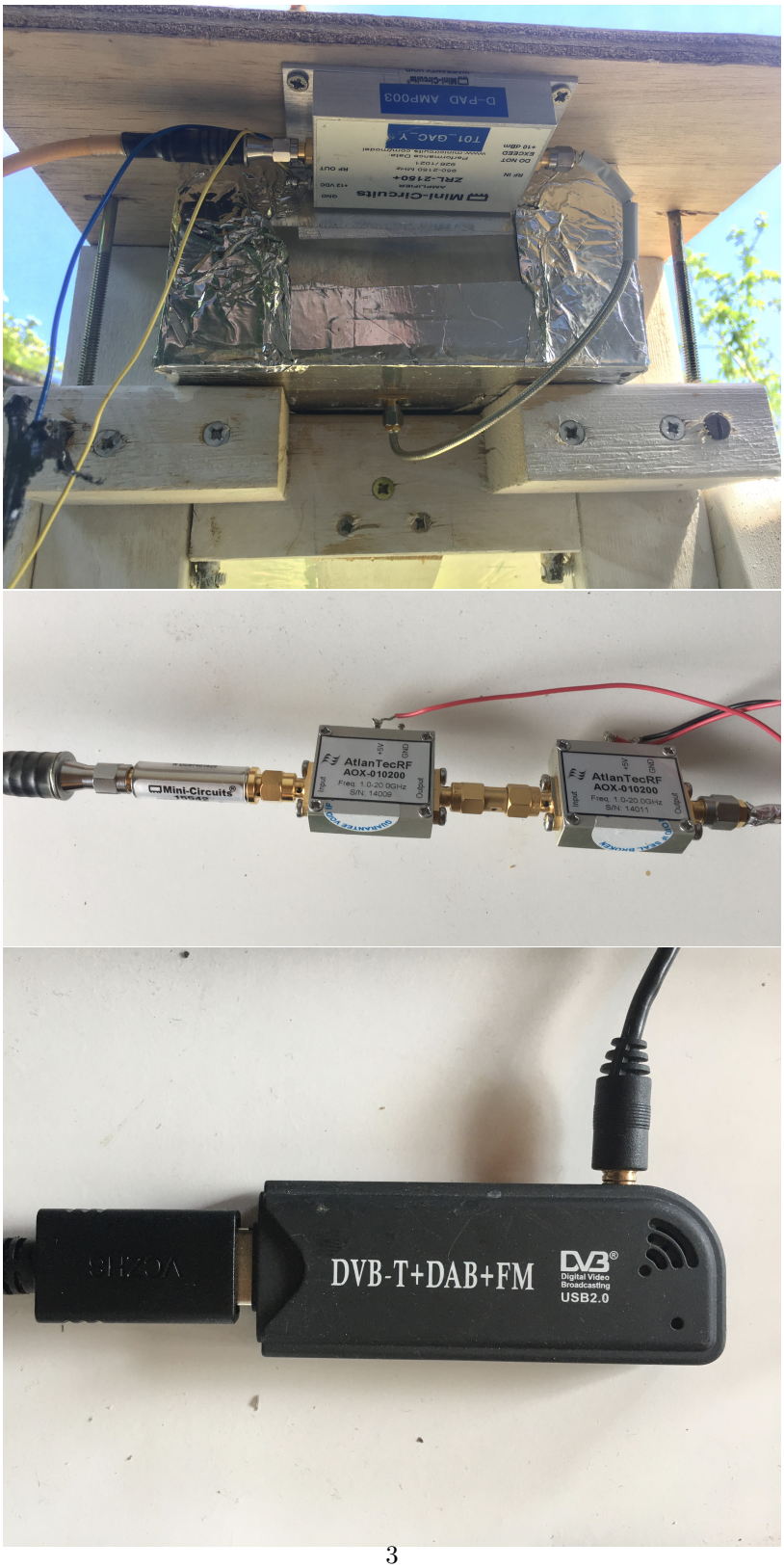
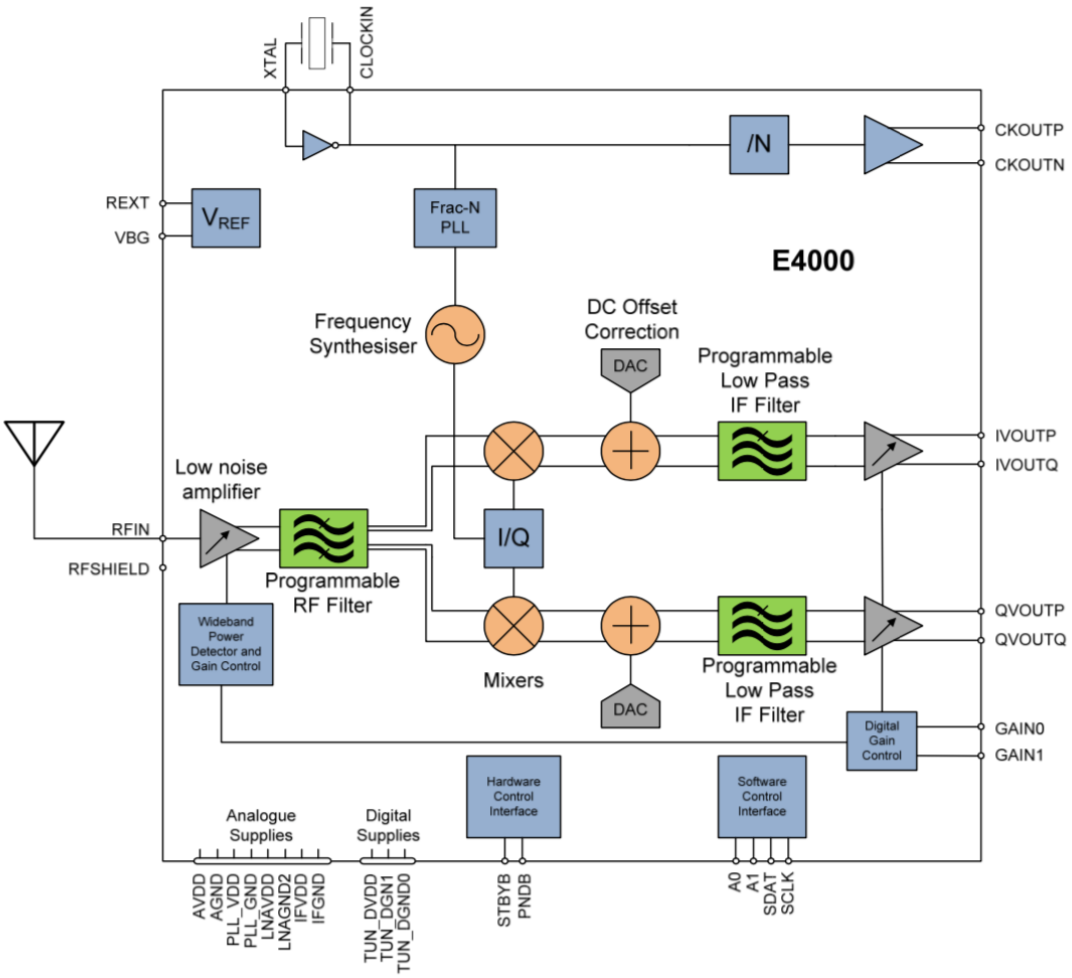





Figure 2: The three components of the receiver chain: (top) the low-noise amplifier; (middle) the band-defining filter and two gain stages; (bottom) the RTL-SDR dongle.



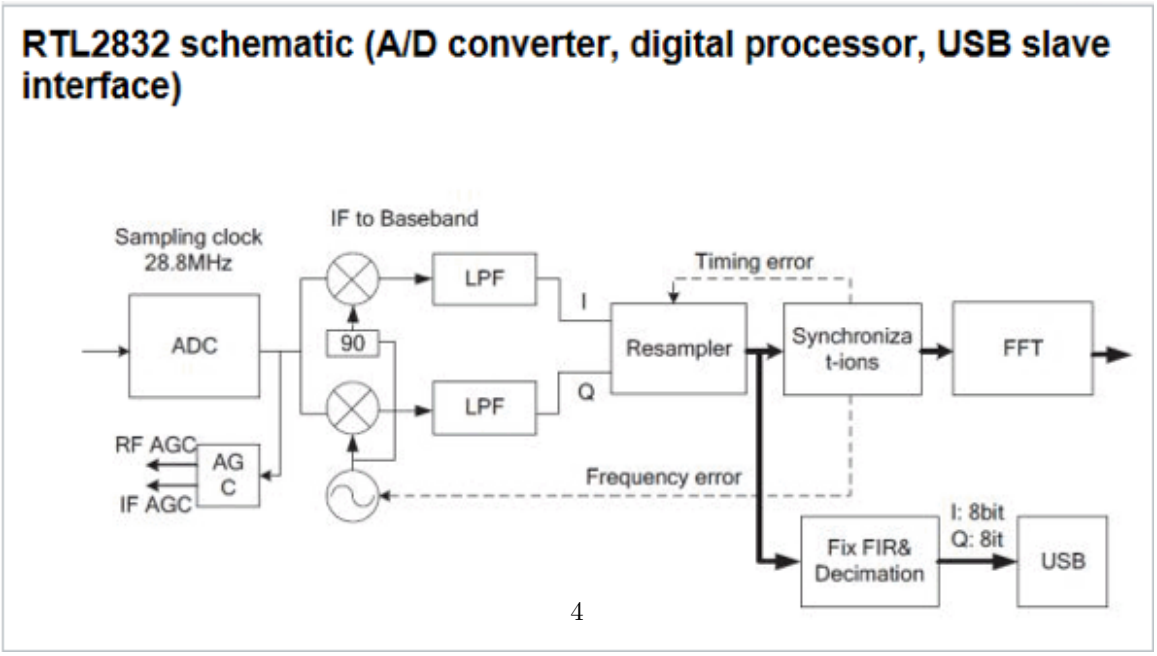






Figure 3: Block diagrams of the two chips that make up the RTL-SDR dongle.

# 4 Calibration theory

We need to take in to account the fact that the telescope electronics are generating noise, which we have to distinguish from the noise arriving from the sky. We will calibrate the data using a hot load, that is, an absorber placed in front of the antenna that is at ambient temperature. When the load is in front of the antenna, the observed temperature is

$$T_1 = G(T_{hot} + T_{rx})$$

where $T_{hot}$ is the temperature of the load, $T_{rx}$ is the noise temperature of the receiver and $G$ is the gain of the receiver. We then place a cold load in front of the antenna. The easiest cold load to use is the sky itself, which at L-band is not much hotter than the CMB - around 4 – 5 K.

$$T_2 = G(T_{cold} + T_{rx})$$

Note that these measurements are a function of frequency. We then take the ratio of these two measurements – this is usually called $Y$, and this method is the $Y$-factor method.

$$Y = T_1/T_2$$

By taking the ratio of these two measurements, we eliminate the gain of the receiver, and calibrate the data in to physical units (assuming our values for $T_{hot}$ and $T_{cold}$ are correct!).

Re-arranging:

$$T_{rx} = (T_{hot} - YT_{cold})/(Y - 1)$$

Assuming values for $T_{hot}$ and $T_{cold}$ – say 300K and 5K respectively – we can solve for $T_{rx}$.

# 5 Calibration and data analysis process

You are provided with a Python notebook for the exercise. There is a function provided to read the data files, but we should study how it works before we use it.

First, you need to read in the three data files `dara_sky.dat, dara_load.dat` and `dara_galaxy.dat`. These are time-ordered, complex data, stored as unsigned 8-bit integers, i.e. values between 0 and 255, with consecutive values representing the real and imaginary parts. These can be read in as follows (from the function provided in your Python notebook) :

```
# Read in the data. It is binary format, unsigned 8 bit integers 'unit8'
# With real value followed by imaginary value for each data point
data = np.fromfile(filename, dtype='uint8')

# This will be the number of distinct complex values, i.e. half the
# number of total values.
datasize = int(np.size(data)/2)

# Create a numpy array (initially filled with zeros) to hold
# the complex data
dataz = np.zeros(datasize,dtype=complex)
```

```
# Assigned the complex values, using numpy indexing trickery.
# real values start on 0 and step over 2
# imag values start on 1 and step over 2
dataz=data[0::2]+1j*data[1::2]

# Calculate the mean of the data set and subtract
offset=np.mean(dataz)
dataz-=offset
```

To convert them to frequency space, we need to apply a Fourier transform. We will use a transform of length 512. There are 46,875 x 512 samples in the files, so we need to sort the data in to blocks of 512 complex samples, and Fourier transform these using the FFT algorithm as shown below

```
# The sampled time chunks in the data are 512 elements long
# This will be the size of the fft we want to do
fftsize=512;

# nspec is the number of distinct time chunks in the time ordered data
# This will also be the number of output spectra
nspec=int(np.floor(datasize/fftsize))
nsamp=nspec*fftsize

# This makes a 2D array out of the 1-D data
# with nspec rows and fftsize columns

dataz= dataz.reshape((nspec,fftsize))
# This calculates the fft, but only of the column axis (axis =1, fftsize long).
fft_dataz= np.fft.fft(dataz,axis=1)
```

The output of the FFT algorithm starts with the zero frequency channel, goes up to the maximum positive frequency, then swaps to the maximum negative frequency, then comes down again. To put these in a more sensible order with zero in the middle, we use the fftshift routine:

```
# This reorders the fft data to that the 0 frequency channel is in the middle,
# (FFT routines expect and return "folded" data ordering 0 to f, -f to 0)
fft_dataz_shift= np.fft.fftshift(fft_dataz,axes=(1,))
```

Finally we need to average the 46,875 individual spectra together to make a single integrated spectrum for the whole 10s of data:

```
# This averages all the the individual 512 spectra together in the 2D array.
# Notice we specify axis = 0 , the opposite axis to which the FFT was performed
# over.
spec = np.mean(np.abs(fft_dataz_shift),axis=0)
```

To assign frequencies to the outputs of the FFT, we remember that the frequency step per channel is the inverse of the total length of each of the FFT blocks:

$$\tau = 1/(2.4 \times 10^6)$$

$$\Delta f = 1/512\tau$$

and the centre frequency, i.e. channel 256, is 1420.0 MHz. This is implemented here...

```
# Calculate the corresponding frequency axis
 samprate=2.4e6;
 fcentre=1420.0e6;
 tau=1/samprate;
 fstep=1/(tau*fftsize);
 freq=fcentre+(fstep*(np.arange(0,fftsize)-fftsize/2));
```

You are provided with a single Python function which wraps up these steps. You can supply a filename and it returns and spectrum and a frequency axis as 1-D arrays. For example, to read in the spectrum and reduce the data for the observation when the telescope was looking at the load, one can do the following:

```
(freq,spec_load) = read_spectrum('dara_load.dat')
```

**Issue this command in your Ipython/Jupyter notebook, to check it works. N.B. make sure that that data .dat file is in the same directory on your computer as the IPython notebook.**

On can then plot out the spectrum as follows,

```
plt.plot(freq, spec_load)
plt.show()
```

1. **Do the same thing for the 'sky' observation (in file `data_sky.dat`. Store the spectrum in a clear variable e.g. `spec_sky`. What features can you see? Any signs of radio frequency interference?**

2. **Do the same thing for the 'galaxy' observation (in file `data_galaxy.dat`. Store the spectrum in a clear filename e.g. `spec_galaxy`. What features can you see? Do you see an additional feature which might be from HI emission from the galaxy which was not present in load or sky data above?**

We would now like to make fully calibrated plots, with sky brightness measured in terms of the receiver noise temperature (i.e. on an antenna temperature scale in Kelvin).

1. **Use your sky and load spectra to calculate $Y$ as a function frequency. See Section 4 above. The Y factor is the ratio of the load and the sky observation. Plot this out.**

2. **Use your $Y$ factor to calculate $T_{rx}$ as a function of frequency, again consult Section 4 above. Plot this out.**

*Question: From your $T_{rx}$ plot, what are typical mean values for $T_{rx}$? Does this seem reasonable given that the noise temperature of the first RF amplifier (which is at room temperature in this system) is likely to be between 150 to 200 K? Think about whether the noise temperature of the second RF amplifier in the receiver chain is important in determining the expected receiver noise temperature.*

1. **Use your galaxy and load spectra to calculate $Y_{gal}$, the $Y$ factor with the galaxy present in the telescope beam as a function of frequency. Plot this out.**

2. **Use your $Ygal$ factor to calculate $T_{gal}$, the antenna temperature with the galaxy present in the telescope beam. Plot this out. Can you see the HI feature from the galaxy?**

**Finally we can subtract $T_{rx}$ from $T_{gal}$ to give a measure of the additional antenna temperature due to the emission from the galaxy alone. Plot this out. What is the brightness, in antenna temperature units, of just the emission from the HI spectral line of the galaxy?**

Remember when looking at these data that they are real observations! Not everything may be exactly as you expect, and there may be unexpected features such as interference.

Some more things to think about: How important is it that we have accurate measurements of $T_{hot}$ and $T_{cold}$? How much would our results change if one or other of them were incorrect by, say, 10%?