<u>Week 8: 24/11/2021 – Wednesday</u>

1. Outline of Meeting

Prior to the meeting, I had met Sam Rowe in the Sequestim Lab in North Building to visit the camera and have a brief introduction to how data measurements are made using the camera. Following this, the meeting with my supervisor in the next day consisted of explaining how the measurements can be extracted from the FSAB file and outlined how the data will be used for noise analysis in the upcoming sections of the project.

2. Specification of Tasks

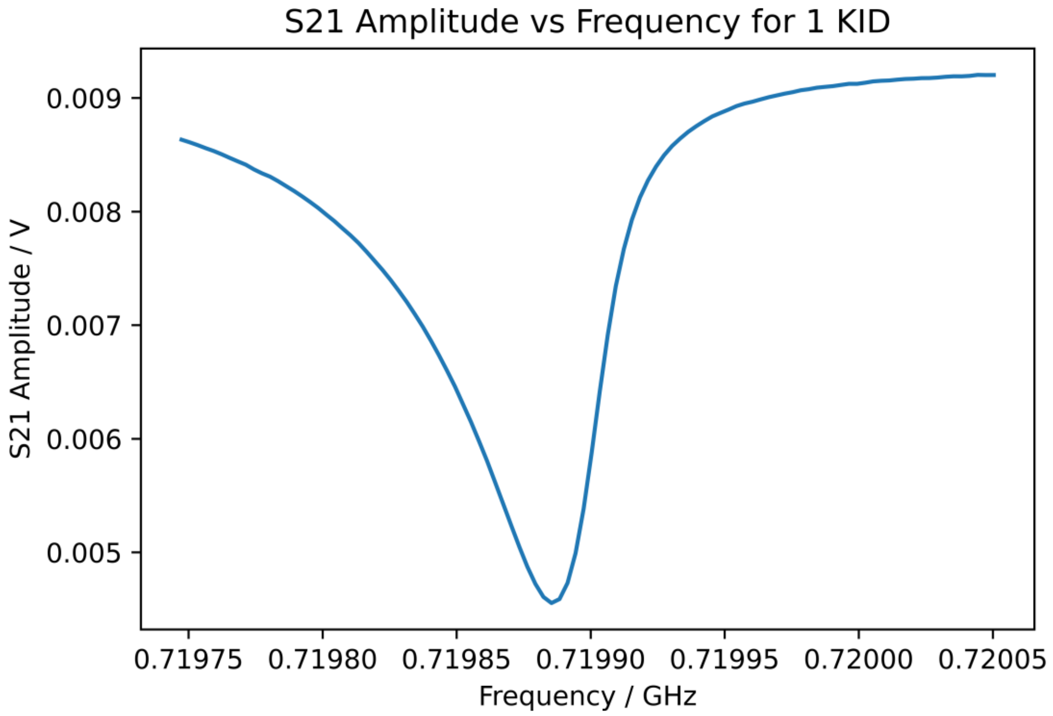
i) Using the FSAB data file, readout out the sweep and timestream data.
ii) Plot the sweep and timestreams data

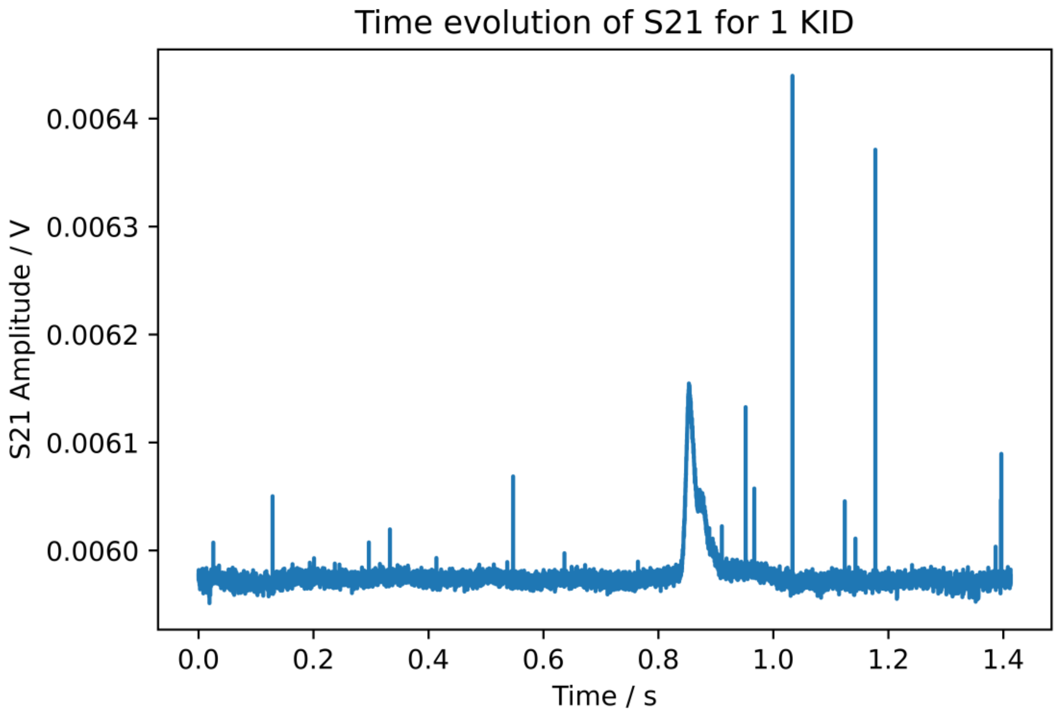
3. Outline and Methodology

The data reading can be quite complicated as the detector outputs the measurements as a timestream and a frequency sweep for the whole array of ~1000 KIDs. The Python Code for extracting the FSAB file was given by Sam Rowe and using this function, we can extract the S21 data from the timestream and sweep data separately. Following this, we can plot the data as S21 vs frequency for the sweep and S21 vs time for the time stream. This is given in section 4. The peaks are when an object is placed in front of the detector as a control.

It is important that this can be accomplished in a robust and automated way and overall we will look at around 300 detectors and therefore cannot do this by hand. Ultimately we will use the sweeps to determine dI/dF and dQ/dF and apply these to the magic formula to get F0 as a function of time. We will later observe two temperatures to determine the responsivity of each device and hence from the noise the NEP.

4. Plots of Sweep and Timestream Data





5. Python Code for extracting data from FSAB file

```
import os
import numpy as np


class fsab_dirfile():
    """
    For analysing FSAB data if libgetdata is not available.
```

```
"""
def __init__(self, path, reference='I0000'):
    if not os.path.exists(path):
        raise FileNotFoundError(path)
    self.path = path

    #parse sweep data:
    self.sweep={}
    with open(os.path.join(path,'sweep'),'r') as file:
        lines=file.readlines()
    #ignore comments
    lines=[i for i in lines if not i.startswith('#')]
    #ignore metadata
    lines=[i for i in lines if not i.startswith('/')]
    #ignore empty
    lines=[i for i in lines if len(i)>1]
    for line in lines:
        line = line.strip().split(' ')
        fieldname,fieldtype,datatype=line[:3]
        data = np.array(line[3:],dtype=np.float)
        _,fiq,num = fieldname.split('_')
        kidnum=int(num)
        if not kidnum in self.sweep.keys():
            self.sweep[kidnum]={}
        if fiq == 'f':
            self.sweep[kidnum]['f'] = data
        if fiq == 'i':
            if not 'z' in self.sweep[kidnum].keys():
                self.sweep[kidnum]['z'] = data.astype(np.cdouble)
            else:
                self.sweep[kidnum]['z'] += data.astype(np.cdouble)
        if fiq == 'q':
            if not 'z' in self.sweep[kidnum].keys():
                self.sweep[kidnum]['z'] = 1j*data.astype(np.cdouble)
            else:
                self.sweep[kidnum]['z'] += 1j*data.astype(np.cdouble)

    self.numkids = len(self.sweep)
    print(self.numkids)

    #load tone freqs
    with open(os.path.join(path,'calibration'),'r') as file:
        lines=file.readlines()
        lines=[i for i in lines if i.startswith('_cal_tone_freq')]
        for line in lines:
            line = line.strip().split(' ')
            fieldname,fieldtype,datatype,data=line
            kidnum = int(fieldname[-4:])
            tonefreq = float(data)
            print(self.sweep.keys())
            self.sweep[kidnum]['tone_freq'] = tonefreq

    self.start_time = np.loadtxt(os.path.join(self.path,'time_start.txt')).item()
    self.stop_time = np.loadtxt(os.path.join(self.path,'time_stop.txt')).item()

    print('Ready %s'%(self.path))



def get_iq_data(self,kidnum):
    assert kidnum < self.numkids
    filename_i = os.path.join(self.path,'I%04d'%kidnum)
    filename_q = os.path.join(self.path,'Q%04d'%kidnum)

    i = np.fromfile(filename_i,dtype=np.float32)
    q = np.fromfile(filename_q,dtype=np.float32)

    return i + 1j*q


def get_sync_data(self):
    filename = os.path.join(self.path,'Q1023')
    return np.fromfile(filename,dtype=np.float32)
```

6. Python Code for Plotting Sweep and Timestream

```
import numpy as np
import sys as sys
import matplotlib.pyplot as plt
import scipy.constants as const
from scipy.special import iv as I0
from scipy.special import kv as K0
import fsab_dirfile_raw as fsab

#Reading data
local_file = "C:\\Users\\Andrew Thean\\Desktop\\Year 3 Project"
data_folder_name = "\_1638195670"
data = fsab.fsab_dirfile(local_file + data_folder_name)

#Frequency Sweep
IQ_data = data.sweep[1]["z"]
sweep = data.sweep[1]["f"]
I = IQ_data.real
Q = IQ_data.imag
```

```
plt.plot(sweep/1e9, (I**2+Q**2)**0.5)
plt.ticklabel_format(useOffset=False)
plt.xlabel("Frequency / GHz")
plt.ylabel("S21 Amplitude / V")
plt.title("S21 Amplitude vs Frequency for 1 KID")
plt.figure()

#Time stream
t = (data.stop_time - data.start_time)/10
for i in range(1, 2):
    IQ_data = data.get_iq_data(i)
    I = IQ_data.real
    Q = IQ_data.imag
    time = np.linspace(0,t, len(I))
    plt.plot(time, (I**2+Q**2)**0.5)
    plt.ticklabel_format(useOffset=False)
    plt.xlabel("Time / s")
    plt.ylabel("S21 Amplitude / V")
    plt.title("Time evolution of S21 for 1 KID")
```