Characterizing Arrays of Kinetic Inductance Detector

AUTUMN DIARY ENTRY

Ashley Thean

Spring Diaries starting pg 42 of document

<u>Week 1: 6/10/2021 – Wednesday</u>

1. **Outline of meeting and tasks**

Meeting with supervisor, Dr. Simon Doyle, on Zoom to discuss the details of the project and to give a brief introduction to the project along with people related to the research such as Dr Sam Rowe and Dr Tom Brien who are part of the research team for this project topic and will assist me.

A brief introduction to the project was provided. In this project, we will be working on understanding the principle of superconductivity and its microwave properties, understand how superconductivity principle can be applied to create a kinetic inductance detector (KID) and thus using the above knowledge to characterize a detector array. The real-world application of the detector would be in an SFAB airport security camera. The reading material provided is Dr. Simon Doyle's Thesis titled "Lumped Element Kinetic Inductance Detectors"

The inductance of a superconducting material forms the operating basis of the detector, hence the name Kinetic Inductance Detector. As such, the task for this week consists of going through the topics related to the theory of superconductivity and how the kinetic inductance of a material can come about due to it. The recommended literature for this is the thesis mentioned above from Chapter 3.1 to Chapter 3.4, page 13-25.

2. **Notes and Materials Covered**

Not much was covered in terms of project material. General introductory session.

Week 2: 13/10/2021 – Wednesday

1. Outline of meeting and tasks

Prior to the meeting on the 12th, I had the opportunity to visit the camera in North Building on the lower ground floor. Tom Brien explained the briefly basics of the detector and how the 2 stage-active cooling of camera allowed the temperature to remain in the ranges of mK.

The meeting with my supervisor was to discuss the aims and objectives of the project and to complete the "Aims and Objectives and safety overview" sheet for submission. The discussed aims and title are shown in the following section. The risk assessment and safety section of the sheet was also discussed.

After this, seeing as there was time, my supervisor proceeded with explaining the topic of superconductivity and how it relates to the detector. The topics covered and further reading material is given in the final section of this week's diary.

My supervisor also provided the literature to read for a better understanding of the topic. The literature given was: "Lumped Element Kinetic Inductance Detectors" by Dr. Simon Doyle, and the recommended section to read was Chapter 3.1 to 3.4, page 13-25. Further outline of the materials elaborated in the final section of this diary.

2. Aims and Objectives Sheet

The title of the project is ***Characterizing Arrays of Kinetic Inductance Detectors.*** The Kinetic Inductance Detectors (KIDs) as mentioned will be used in an SFAB Security Imaging System. The discussed aims of the experiment are:

To fully characterise the SFAB Security imaging system in terms of detector sensitivity and yield. This will involve.

1) Learning and understanding the microwave properties of superconductors

2) Learning and understanding how a Lumped Element Kinetic Inductance Detector works

3) Be able to form a simple simulation of a LEKID detector

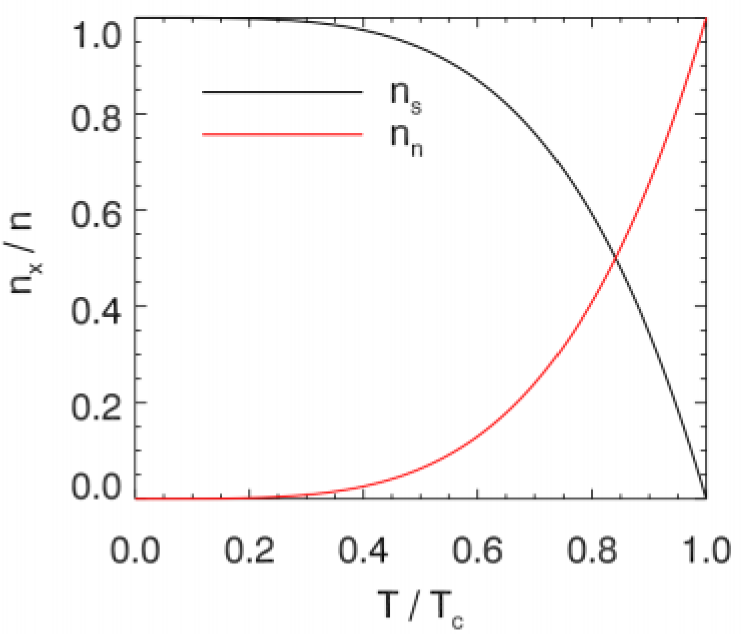
4) Understand how to analyse data from a LEKID array and to make sensitivity measurements from this data

5) Be able to compare the data of each detector in the array to the expected photon noise limit.

3. Notes and Materials Covered

Based on the literature provided in the pages outlined above, from the theory of superconductivity the phenomenon of kinetic inductance arises from it and as such is an important concept as a prerequisite to the fundamentals of the detector. A simple notes and explanation of superconductivity and kinetic inductance is given below:

In some elemental metals such as Niobium, Aluminium and Tin, the DC resistance falls to zero below a certain temperature known as the critical temperature, and as such these materials are known as superconductors. In normal metals, electrons scattering off the ions in the lattice cause a loss in kinetic energy and this leads to non-zero resistance. However, in superconductors below the critical temperature T_c, a fraction of the electron population start to "pair-up" into Cooper-pairs and these pairs are immune to scattering and as such have 0 resistance. The ratio, $\frac{n_s}{n_n}$ of the population of superconducting Cooper pairs n_s to unpaired electrons n_n is inversely proportional to the temperature T below the critical temperature. As T decreases, the number of cooper pairs increases. A plot of this relationship is shown:



Source: Simon Doyle Understanding Superconductivity and KIDs Slides

When an electrical current is applied on a superconducting material, the current simply passes through the population of superconducting electrons and thus arises zero resistance. When the temperature increases, the kinetic energy of the electrons break Cooper pairs and the population of n_n increases.

Another important concept about superconductivity is the Meissner Effect. Essentially, a superconducting material will completely expel any magnetic flux density from within the bulk by creating surface currents to cancel any flux that penetrates it. The relationship of the field into the material is characterized by the 2 London equations:

$$\frac{dJ}{dt} = \frac{n_s e^2}{\omega m}$$

$$\lambda_L = \sqrt{\frac{m}{\mu_0 n_s e^2}}$$

λ_L is the London Penetration depth. It is the value at which the field decays to ${}^{1}/_{e}$ of its value at the surface within the distance $\sqrt{m/\mu_0 n_s e^2}$. The result of the London equations are essential in characterizing the inductance of the material.

The next portion touches on the two fluid model of the behaviour of the electrons in the material. As mentioned previously, below T_c, the populations of the electrons split into 2 different populations n_s and n_n which coexist within the lattice. n_n is affected by the usually scattering and exhibits loss. n_s on the other hand, does not undergo scattering and thus no loss. The Cooper pairs are bound together with an energy gap of 2Δ (1 from each electron). The model takes into account that the current in the superconductor has 2 paths to travel through, n_s and n_n. The ratio of n_s/n (n being the total conducting electrons) is given by:

$$\frac{n_s}{n} = 1 - (\frac{T}{T_c})^4$$

$$n_n = n - n_s$$

The above gives the temperature dependence of the population of the superconducting electrons, and will be essential in future calculations for the conductivity of the material. The conductivity of n_s can be denoted by σ_s and conductivity of n_n is denoted by σ_n. Taken directly from the notes: "At low frequencies, the σ_s is far greater than σ_n, thus displaying zero resistance. At higher frequencies however, especially in the microwave region, σ_n can play a considerable part in the conductivity. This is due to the kinetic inductance of the superconducting electrons. The inertia of these electrons produces a reactance giving us a large impedance at high frequencies. This effect is likened to an inductance as the energy drawn from the field E is stored in the kinetic energy of thse non-scattering electrons." As the temperature or frequency increases, more of the current will be shunted through the non-superconducting resistive path. From this, derives the temperature dependence of the London Penetration Depth, LPD:

$$\lambda_L = \lambda_L(0)\ [1 - \left(\frac{T}{T_c}\right)^4]^{-0,5}$$

Where $\lambda_L(0)$ is the LPD at 0 Kelvin.

By summing up the kinetic energies of all the superconducting electrons, we can use this to determine the kinetic inductance of the material given by the following expression:

$$L_k = \frac{\mu_0\lambda^2}{Wt}$$

Where W is the width of the sheet and t is the thickness. It is also further simplified by calculating the L_k of a square of the material and thus the W term vanishes. We are also often working in the limits of $t << \lambda$ and $t >> \lambda$, thus we need to perform surface integrals for current over the entire cross-sectional area to take into account variations in current density. L_k and L_m, the magnetic inductance is given by:

$$L_k = \frac{\mu_0\lambda^2}{4}[\coth\left(\frac{t}{2\lambda}\right) + \left(\frac{t}{2\lambda}\right) cosech^2\left(\frac{t}{2\lambda}\right)$$

$$L_m = \frac{\mu_0\lambda^2}{4}[\coth\left(\frac{t}{2\lambda}\right) - \left(\frac{t}{2\lambda}\right) cosech^2\left(\frac{t}{2\lambda}\right)$$

The total internal inductance is thus given by:

$$L_{int} = L_m + L_k = \frac{\mu_0\lambda}{2}\coth\left(\frac{t}{2\lambda}\right)$$

Week 3: 20/10/2021 – Wednesday

1. Outline of meeting

The meeting was carried out on Zoom. The meeting was a discussion about the topics discussed previously, and as they are dense topics, a recap of the theory of superconductivity and the associated inductances. The discussed recap and outline of the task is given in section 3 of this diary. The meeting also outlined a task to recreate Figure 3.3 from Dr. Simon Doyle's thesis titled "Lumped Element Kinetic Inductance Detectors" and to calculate the internal inductance of an aluminium sheet given specific parameters. The specification of the task is given in the next section.

2. Specification of Tasks

i) Using the equations and theory discussed, recreate the plot from the thesis, Figure 3.3

ii) Calculate the internal inductance for an aluminium square with the following properties:

$$Thickness = 50nm$$
$$n_{electrons} = 18 \times 10^{28}$$
$$Critical\ temperature, T_c = 1.4\ K$$
$$Temperature, T = 0.3\ K$$

3. Outline of Theory and Methodology for Task

Using the equations 3.19 and 3.20 from last week to determine the kinetic and magnetic inductance takes the surface integral for current over the entire cross-sectional area to consider variations in current density:

$$L_k = \frac{\mu_0 \lambda^2}{4W}[\coth\left(\frac{t}{2\lambda}\right) + \left(\frac{t}{2\lambda}\right) cosech^2\left(\frac{t}{2\lambda}\right)$$
$$L_m = \frac{\mu_0 \lambda^2}{4W}[\coth\left(\frac{t}{2\lambda}\right) - \left(\frac{t}{2\lambda}\right) cosech^2\left(\frac{t}{2\lambda}\right)$$

The kinetic and magnetic inductances per square can be determined using the equations. As discussed in the meeting, the inductances can be simplified by calculating the value per square of the material instead and the W term vanishes. To find the ratio of L_k and L_m to L_{int}, the value of L_{int} was found using the equation 3.21 from the thesis:

$$L_{int} = L_m + L_k = \frac{\mu_0 \lambda}{2}\coth\left(\frac{t}{2\lambda}\right)$$

The λ term was a fixed value of 50nm. The thickness was varied for a range of 0nm to 300nm. The ratios were found and the inductances were plotted against each other, the plot is show in the section below.

The next part of the task was to calculate the internal inductance of a thin aluminium sheet using the parameters given. The first step is to calculate the London Penetration Depth at T=0K. The 2nd London Equation can be used to find the value:

$$\lambda_L = \sqrt{\frac{m}{\mu_0 n_s e^2}}$$

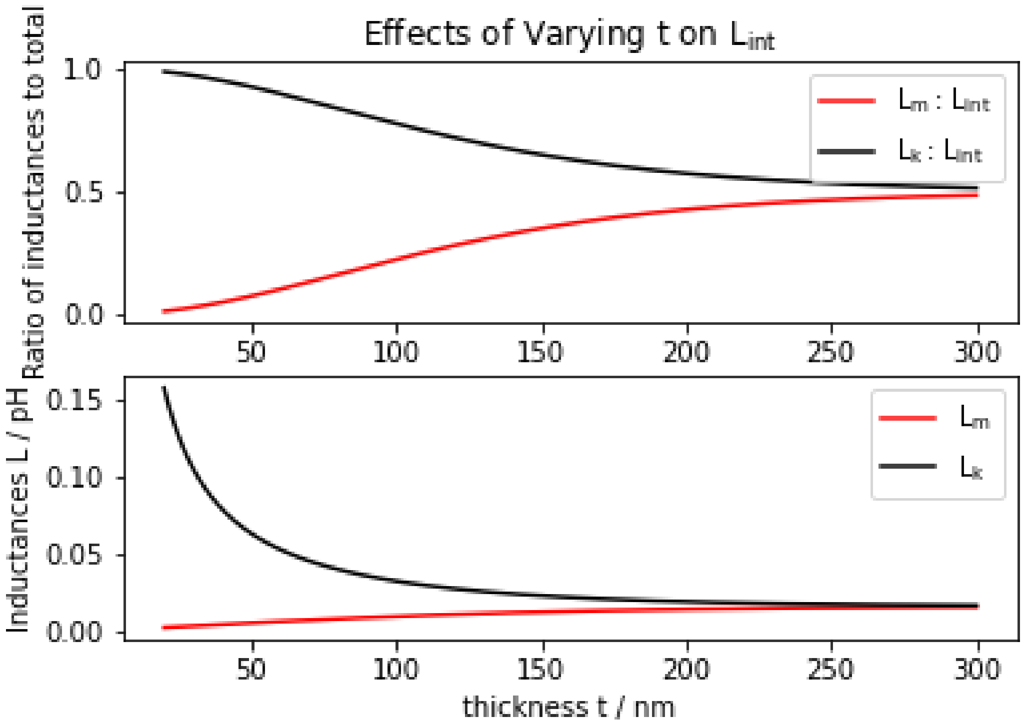
As the temperature decreases, the population of n_s increases and n_n decreases. The electrons will form Cooper-pairs as the temperature decreases. At 0K, ALL the electrons will have paired up and thus, $n_{electrons} = n_s$. Thus, we can calculate the LDP at 0K using this.

The temperature dependence of λ_L is given by equation 3.13 in the thesis:

$$\lambda_L(T) = \lambda_L(0)\ [1 - \left(\frac{T}{T_c}\right)^4]$$

From this, we can calculate the internal inductance using the parameters provided. The result is given in section 5.

4. Recreated Plots of Figure 3.3 of Thesis



5. Calculated Value for The Internal Inductance

$$L_{int} = 0.47 \frac{pH}{square}$$

6. Code for Calculating Inductances

```
'''
#####################################################################################
1. Effects of varying film thickness for $L_{m}$ and $L_{k}$ on a square film
#####################################################################################
'''
#Imports
import numpy as np
import matplotlib.pyplot as plt

def coth(x):
  return np.cosh(x)/np.sinh(x)
```

```
def cosec(x):
   return 1/np.sin(x)
def cosech(x):
   return 1/np.sinh(x)

#Defining lambda, miu_0 and lower and upper limit of t
lam = 50*10**-9
miu_0 = 1.25663706212*10**-6
lowerLimit, upperLimit = 20e-9, 300e-9

#Create array of varying thickness t
t = np.linspace(lowerLimit, upperLimit, num=1000)

#Define a fraction for neater code
fraction = t/(2*lam)

#Calculating lk
lk = (miu_0 *lam/4)*(coth(fraction) + fraction*(cosech(fraction))**2)
lm = (miu_0 *lam/4)*(coth(fraction) - fraction*(cosech(fraction))**2)
# ### The total internal inductance is given by: $L_{int} =
\frac{\mu_{0}\lambda}{2}coth(\frac{t}{2\lambda})$

#Calculating total inductance
l_int = (miu_0 * lam/2)*coth(t/(2*lam))

#Ratios
ratio_lm = lm/l_int
ratio_lk = lk/l_int

#Subplots
plt.subplot(2,1,1)
params = {'mathtext.default': 'regular' }
plt.rcParams.update(params)
plt.plot(t*10**9, ratio_lm, 'r-', label = '$L_{m}:L_{int}$')
plt.plot(t*10**9, ratio_lk, 'k-', label = '$L_{k}:L_{int}$')
plt.legend(loc='best')
plt.ylabel('Ratio of inductances to total')
plt.grid()
plt.title('Effects of varying t on $L_{int}$')

#Subplot 2
plt.subplot(2,1,2)
plt.plot(t*10**9, lm*10**12, 'r-', label = '$L_m$')
plt.plot(t*10**9, lk*10**12, 'k-', label = '$L_k$')
plt.legend(loc='best')
plt.xlabel('thickness t / nm')
plt.ylabel('Inductances L / pH')
plt.grid()
plt.savefig("Effects of Varying t on L_int")

'''
```

```
###################################################################################
2. Internal Inductance For Aluminium Square
###################################################################################
'''

#Define variables
n = 1.8e28
temp_c = 1.4
temp = 0.3
t_al = 50e-9
e = 1.6e-19
me = 9.11e-31

# ### We can then determine the London Penetration Depth
lam0 = (me/(miu_0*n*e**2))**0.5
lamL = lam0*(1 - (temp/temp_c)**4)**-0.5

#Calculate L_int
l_int_al = (miu_0 * lamL/2)*coth(t_al/(2*lamL))
print(l_int_al)
```

Week 4: 27/10/2021 – Wednesday

1. **Outline of meeting**

The meeting was held on Zoom. The first half of the meeting was dedicated to bug-fixing the code from the previous week and correct any misunderstandings from the topic. The second half of the meeting was dedicated to the Mattis Bardeen Theory. A discussion was had regarding the topic and a careful elaboration to the best of my understanding was given. An outline of the Mattis Bardeen Theory will be elaborated in section 3 of this diary. Essentially, Mattis Bardeen theory is derived from the fundamental principles of superconductivity and takes the band gap into consideration. From this, the task that arose from this is to create the plots of L_{int} vs T using Mattis Bardeen Approximations to find the L_{int} from the band gap energy. Then, the resistive part of the impedance can be found, and a plot of R vs T can be made. The outline of the theory and task specification is given below.

2. **Specification of Tasks**

i) Plot out L_{int} over a temperature range of 0.05 – 4 K from Mattis Bardeen Approximations using equation 3.33 for a superconducting film with the following properties:
   - Normal state conductivity: $\sigma_n = 6.0 \times 10^7$
   - Thickness: $t = 20 \times 10^{-9}\ m$
   - Critical temperature: $T_c = 1.5\ K$
   - Frequency: $f = 500 \times 10^6\ Hz$

ii) Use equation 3.38 to calculate R as a function of temperature

3. **Outline of Theory and Methodology for Task**

Mattis Bardeen Theory: *"The London Equations derived in previous weeks hold well but they are not derived from any fundamental principles of superconductivity and does not take into account the idea of a band gap. Another assumption of the London Model is that it assumes that the electrons in the superconducting state are just simply electrons which do not scatter and will all be accelerated independently if an electric field is applied."*

Source: Lumped Element Kinetic Inductance Detector – Dr. Simon Doyle Thesis

Building from Mattis Bardeen Theory, the full effects of the band gap and non-local treatment of Cooper pairs leads to the Mattis Bardeen Integrals:

$$\frac{\sigma_1}{\sigma_n} = \frac{2}{\hbar\omega}\int_{\Delta}^{\infty}[f(E) - f(E+\hbar\omega)]g(E)dE + \frac{1}{\hbar\omega}\int_{\Delta-\hbar\omega}^{\Delta}[1 - f(E+\hbar\omega)]g(E)dE$$

$$\frac{\sigma_2}{\sigma_n} = \frac{1}{\hbar\omega}\int_{\Delta-\hbar\omega,-\Delta}^{\Delta}\frac{[1-2f(E+\hbar\omega)][E^2+\Delta^2+\hbar\omega\mathrm{E}]}{[\Delta^2-E^2]^{\frac{1}{2}}[(E+\hbar\omega)^2-\Delta^2]^{\frac{1}{2}}}$$

To simplify, the integrals can be approximated when in the limits $k_BT << \Delta(0)$ and $\hbar\omega << \Delta(0)$ to the Mattis Bardeen Approximations:

$$\frac{\sigma_1}{\sigma_n} = \frac{2\Delta(\mathrm{T})}{\hbar\omega}\exp(\left(-\frac{\Delta(0)}{\mathrm{k_B T}}\right)\mathrm{K_0}(\frac{\hbar\omega}{2\mathrm{k_B T}})[2\sinh(\frac{\hbar\omega}{2\mathrm{k_B T}})]$$

$$\frac{\sigma_2}{\sigma_n} = \frac{\pi\Delta(\mathrm{T})}{\hbar\omega}[1 - 2\exp(\left(-\frac{\Delta(0)}{\mathrm{k_B T}}\right)\exp\left(\frac{-\hbar\omega}{2\mathrm{k_B T}}\right)\mathrm{I_0}(\frac{\hbar\omega}{2\mathrm{k_B T}})]$$

Where Δ is the band gap energy, I_0 and k_0 are modified Bessel functions of the first and second kind respectively. $\Delta(\mathrm{T})$ in this case was approximated to $\Delta(0)$ since $T \cong 0$.

Following this, the London Penetration depth can be found by first determining the electron density n_s using the following relation:

$$n_s = \sigma_s \omega m_e / e^2$$

Where σ_s = σ_2 in this case. Using the result, the Penetration depth can be found:

$$\lambda_{MB} = \sqrt{\frac{m}{\mu_0 n_s e^2}}$$

Plugging λ into the expression for total internal inductance:

$$L_{int} = \frac{\mu_0 \lambda}{2}\coth\left(\frac{t}{2\lambda}\right)$$

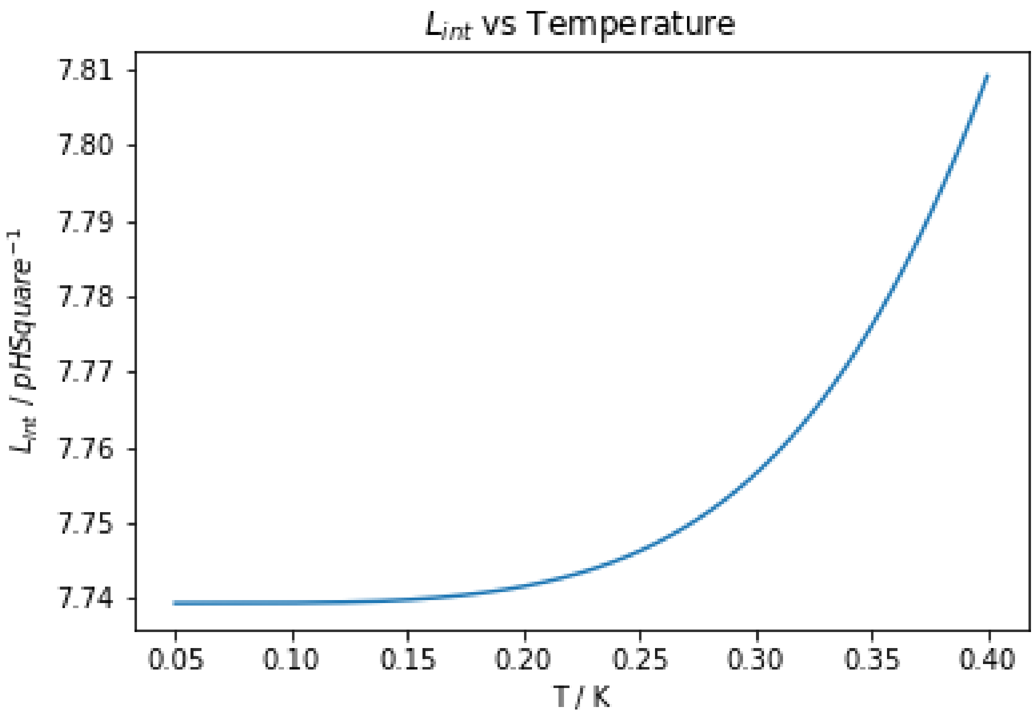
The graph of L_{int} vs T can be plotted.

Using the expression for the resistive part from equation 3.38 in the thesis:

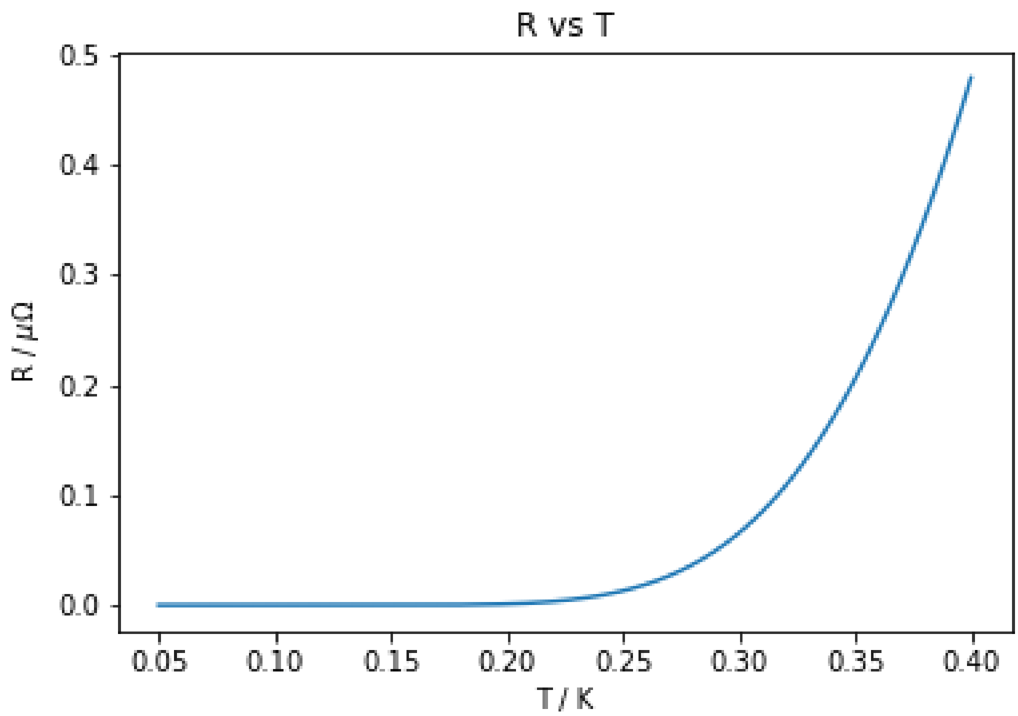
$$R = L_k\,\omega\frac{\sigma_1}{\sigma_2}$$

A plot of R vs T can also be made. The plots and code for the tasks are shown in the following sections.

4. Plot of L_{int} vs T Using Mattis Bardeen Approximations



5. Plot of R vs T Using Mattis Bardeen Approximations



6. Code for Calculating L_{int} and R

```
#imports
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy.special import iv as I0
from scipy.special import kv as K0
import scipy.constants as const

#Define function
def coth(x):
    return np.cosh(x)/np.sinh(x)

def csch(x):
    return 1/np.sinh(x)

#Define constants
TC = 1.5
Delta_0 = (3.5*const.Boltzmann*TC)/2
sigma_n = 6.0e7         # Normal stae conductvity if superconducting film
Thick = 20e-9           # Thickness of superconducting fil
f = 500e6
w = 2 * np.pi * f
me = const.m_e
miu_0 = 4*np.pi*10**-7

#Varying range of temperature
T = np.linspace(0.05, 0.4, num=500)

#An interpolation formula for delta_T (Cheating a bit by using an interpolation formula, ideally
should be integrated)
#Source: https://physics.stackexchange.com/questions/192416/interpolation-formula-for-bcs-
superconducting-gap#mjx-eqn-eq2
delta_T = Delta_0*np.tanh(1.74*np.sqrt((TC/T)-1))

#Define constants to simplify eqn
multiplying_constant = delta_T/(const.hbar * w)
e_const_1 = - Delta_0/(const.Boltzmann*T)
e_const_2 = (const.hbar*w)/(2*const.Boltzmann*T)

#Parts of the sigma1 Ratio
A = 2*multiplying_constant
B = np.exp(e_const_1)
C = K0(0, e_const_2)
D = 2*(np.sinh(e_const_2))

#Find Sigma 1 and Sigma 2
sigma1Ratio = A * B * C * D
sigma2Ratio = np.pi*multiplying_constant*(1 - (2*np.exp(e_const_1)*np.exp(-
e_const_2)*I0(0,e_const_2)))
sigma2 = sigma2Ratio * sigma_n
sigma1 = sigma1Ratio * sigma_n
```

```
# Sanity Check
# plt.subplot(2,1,1)
# plt.plot(T, sigma1Ratio)
# plt.ylabel("$\sigma_{1}/\sigma_n$")
# plt.yscale("log")
# plt.subplot(2,1,2)
# plt.plot(T, sigma2Ratio)
# plt.ylabel("$\sigma_{2}/\sigma_n$")
# plt.show()
# plt.figure()


#Depth
lower_fraction = miu_0*sigma2*w
Lambda_T_MB = (1/lower_fraction)**0.5

#Internal Inductance
fraction = Thick/(2*Lambda_T_MB)
L_int = (miu_0*Lambda_T_MB/2)*coth(fraction)
plt.plot(T, L_int*10e12)
plt.ylabel("$L_{int}$ / $pHSquare^{-1}$")
plt.xlabel("T / K")
plt.title("$L_{int}$ vs Temperature")
plt.savefig("L_int vs T Mattis Bardeen")
plt.figure()

#sigma 1 to sigma 2
sigma12Ratio = sigma1/sigma2

#Terms for lk
A = (miu_0*Lambda_T_MB)/4
B = coth(fraction)
C = fraction*(csch(fraction))**2

#R vs T
lk = A*(B+C)
R = lk * w * sigma12Ratio
plt.title("R vs T")
plt.plot(T, R*10**6)
plt.ylabel("R / $\mu\Omega$")
plt.xlabel("T / K")
plt.savefig("R vs T Mattis Bardeen")
```

Week 5: 03/11/2021 – Wednesday

1. Outline of Meeting

The meeting was carried out on Zoom. The first part of the meeting was dedicated to answering questions and clearing up any misconceptions from the previous week's topic, the Mattis Bardeen Theory. Following this, the basic concepts of KID microwave readout. In particular, the scattering parameters and the effects it has on microwave circuits. From this, the task for this week was specified which was to create a S21 plots for its amplitude and phase for a range of frequencies. More details on the theory and concepts is given in section 3 of this diary.

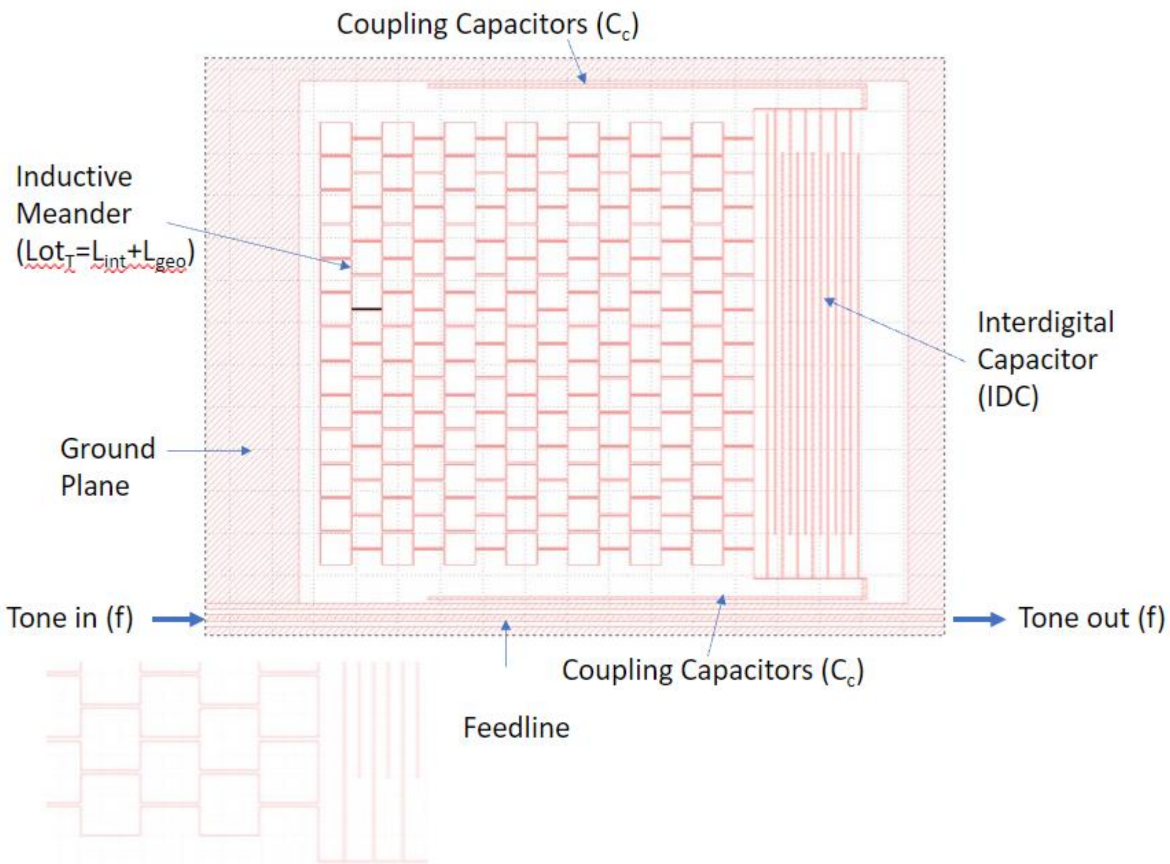
2. Specification of Tasks

i) Read through the Understanding_Kinetic_Inductance_Detector_Microwave_readout.pdf document on the one drive. Try and create the S21 plots for amplitude and phase. Use the following parameters:

- F_0 = 1 GHz
- Q_r = 9000
- Q_c = 10000

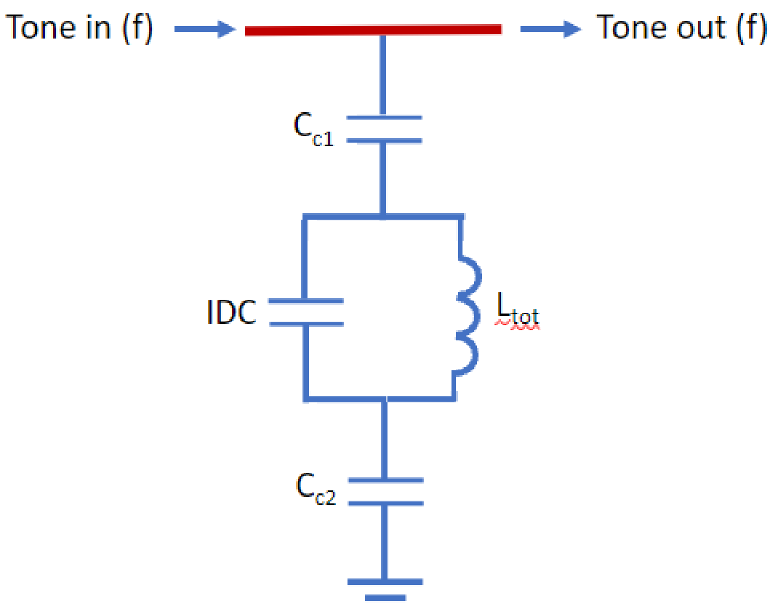
3. Outline of Theory and Task Methodology

A schematic of a single pixel on a KID is shown below:



Source: A schematic of a single pixel on a KID provided by Simon Doyle's Slides

The circuit sits on a silicon wafer (white part) and that wafer sits on an aluminium sheet (dashed lines) which is the ground plane. The inductive meander can be characterized as a single inductor with inductance L_{tot}. An equivalent circuit is given below:



Source: An equivalent circuit to the schematic of a single pixel on a KID provided by Simon Doyle's Slides

The basic mechanism of the circuit is as follows:
Incident photons will provide energy for Cooper pairs to overcome the band gap energy Δ. As such, the cooper pairs breaks and as a result the density of n_1 increases while the superconducting n_s density decreases. This increases σ_1 and as a result, the value of L_{int} will vary. The dimensions of the inductor does not change, as such will remain constant throughout. Thus, the total inductance L will change accordingly to a change in L_{int}.

For the capacitors, there are 2 coupling capacitors C_c that couple to the transmission feedline and ground. The total capacitance C_c from C_{c1} and C_{c2} is simply the series sum:

$$C_c = \frac{C_{c1}C_{c2}}{C_{c1} + C_{c2}}$$

The other capacitor is the Intedigital capacitor IDC which is coupled to the Inductive Meander L_{tot}. We can find the total capacitance of the whole circuit as a simple parallel sum of the capacitances:

$$C_{tot} = IDC + C_c$$

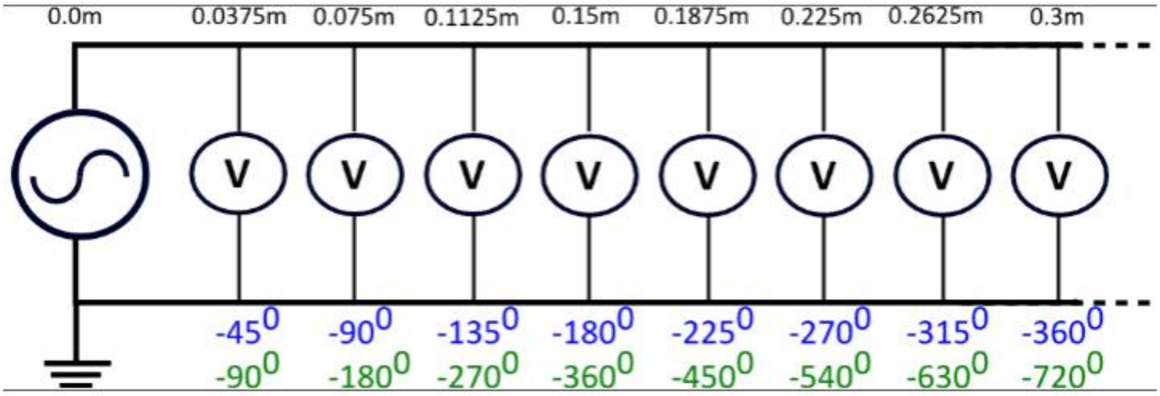
The circuit can be modelled as a basic LC circuit with a resonant frequency given by:

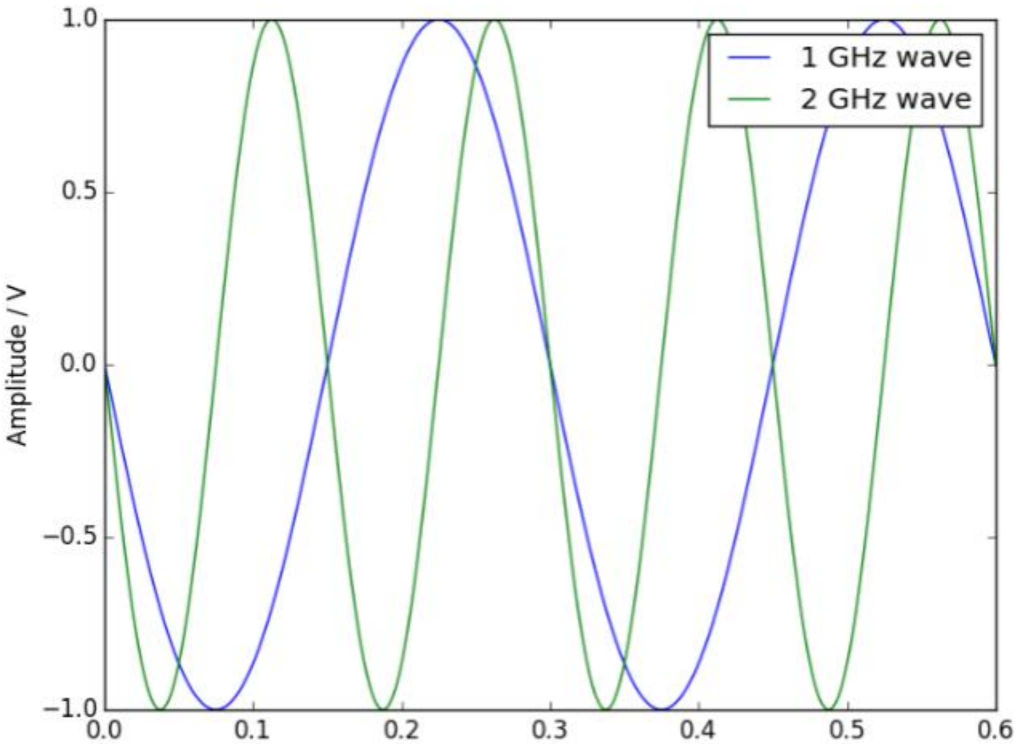
$$\omega_0 = \frac{1}{\sqrt{L_{tot}C_{tot}}}$$

Since L_{tot} varies with respect to photon intensity incident on the inductive meander, ω_0 varies as well, since the capacitors are coupled to the transmission feedline, therefore the incidence of the photons can be detected, thus a detector. More details on the relationship between incident photon energy and how the detector responds to it will be explored in future meetings and diaries. We have thus crudely characterized a single pixel of a KID.

Another phenomenon faced by the KID:
KIDs work in the microwave frequency range *"Unlike low frequency electronics, working at microwave frequencies typically requires one to treat a signal as a wave rather than simply voltages and currents that one may be used to. This is because at higher frequencies, electronic components become similar in size as the wavelength of the signals being measured."*
Source: Understanding_Kinetic_Inductance_Detector_Microwave_readout – Simon Doyle

Since the wavelengths of the signals being measured is comparable to the length of the wire, it would be useful to be able to characterize the phase at a certain point of the wire. Due to the long wavelength, the phase plays a significant role as some points of the circuit will be out of phase. This is illustrated on the diagram below:





Source: Understanding_Kinetic_Inductance_Detector_Microwave_readout – Simon Doyle

Due to this, it is useful to define a microwave circuit in terms of their scattering parameters. These define the voltage waves entering and leaving a microwave circuit via two ports. We denote the scattering parameter as S21 and the S21 for a KID is given as follows:

$$S_{21} = 1 - \frac{Q_r}{Q_c}\frac{1}{1 + 2jQ_r x}$$

Where Q_c and Q_r are known as the coupling Q and resonator Q respectively. These are set by the physical properties of the resonator. J is the complex number $-1^{0.5}$. x is given by:

$$x = \frac{F - F_0}{F_0}$$

Where F_0 is the resonant frequency of the KID and F is the wave propagated along the feedline. Using S21, the Amplitude and Phase of S21 against frequency can be plotted, where:

$$|S21| = \sqrt{S21_{Real}^2 + S21_{Imaginary}^2}$$

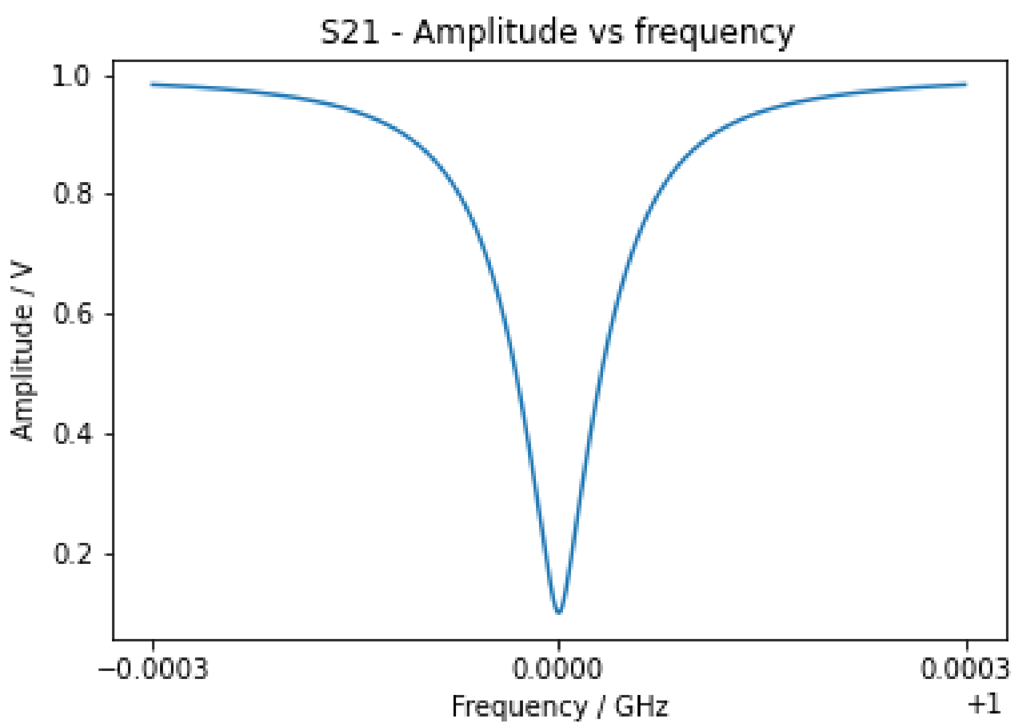
$$Phase_{S21} = Arctan(\frac{S21_{Imaginary}}{S21_{Real}})$$

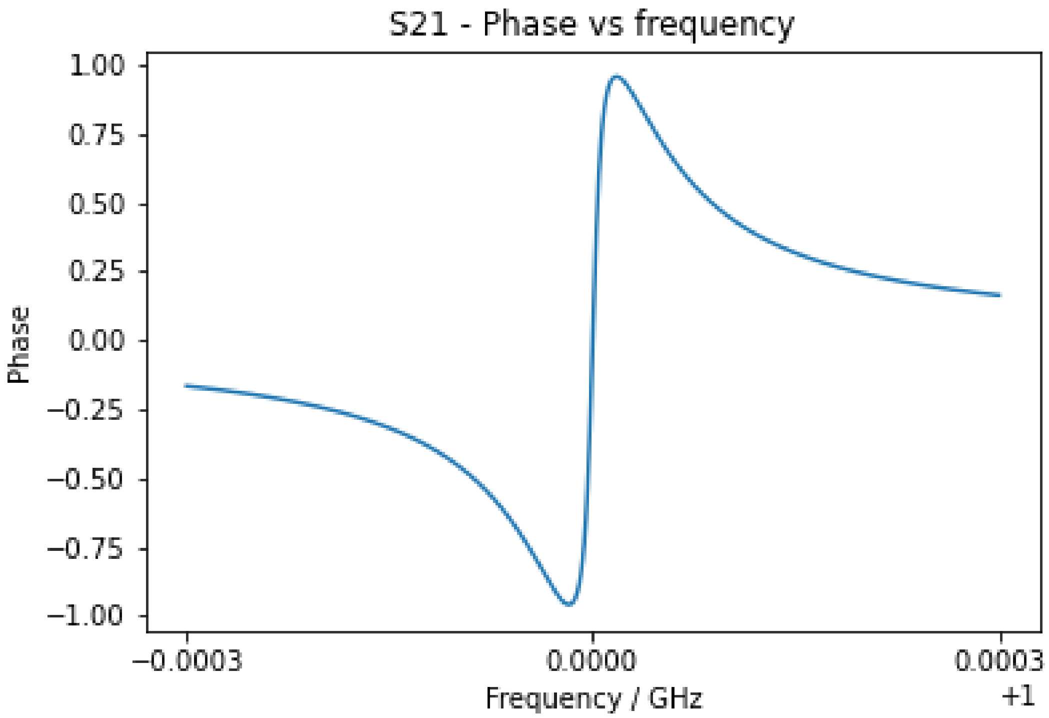
Where the imaginary and real components of S21 can be found it Python using the following code:

```
S21_Real = S21.real
S21_Imaginary = S21.imag
```

As part of the task, the plots of amplitude and phase for S21 against frequency in the bandwidth of 0.0006 GHz with F_0 at 1 GHz are given in the following section.

4. Plots of S21 for Amplitude and Phase vs Frequency





5. Code For Calculating the Components of S21

```
import numpy as np
import matplotlib.pyplot as plt
import cmath as cmath

#Function for calculating S21
def S21(Qr, Qc, f):
    x = (f-f0)/f0
    result = 1 - (Qr/Qc)*(1/(1+2j*Qr*x))
    return result

#Define values
Qr=9000
Qc=10000
f0 = 1e9

#range of frequencies
lower_f = f0-3e5
upper_f = f0+3e5
f = np.linspace(lower_f, upper_f, 1000)

#Compute S21
S21_value = S21(Qr, Qc, f)
S21_real = S21_value.real
S21_imag = S21_value.imag

#Amplitude
```

```
plt.figure()
S21_amplitude = (S21_real**2 + S21_imag**2)**0.5
plt.plot(f*10**-9, S21_amplitude)
plt.title("S21 - Amplitude vs frequency")
plt.xlabel("Frequency / GHz")
plt.ylabel("Amplitude / V")
plt.xticks([0.9997, 1.000, 1.0003])
plt.savefig("Amplitude S21 Plot")

#Phase
plt.figure()
S21_phase = np.arctan(S21_imag/S21_real)
plt.plot(f*10**-9, S21_phase)
plt.title("S21 - Phase vs frequency")
plt.xlabel("Frequency / GHz")
plt.ylabel("Phase")
plt.xticks([0.9997, 1.000, 1.0003])
plt.savefig("Phase S21 Plot")
```

Week 6: 10/11/2021 – Wednesday

1. Outline of Meeting

The meeting was held on Zoom. Following from last week's topic, the meeting was focused on characterizing a KID. Building from last week's theory, this week extends for characterizing the measurement of a KID. By using the scattering parameters equations for the KID, by determining the ABCD values which are related to the transmission line, the S21 value can be found for the detector. This week's task was focused on using a Python function given by my supervisor and given parameters to model a KID for a certain temperature range. The result of which is a plot of cascading frequencies each curve corresponding to a detector.

2. Specification of Task

i) Find Lint using Mattis Bardeen Approximations for Aluminium (use values given last time) at a temperature of 0.2K. Multiply this by "Squares" to get total Lint.

ii) Find the IDC value for the lowest temperature and make this fixed. Found from F0 and (Lint+Lgeo) and C_couple.

$$\omega_0 = \frac{1}{\sqrt{(L_{int}+L_{Geo})(C_{IDC}+C_{Couple})}}$$

iii) Run simulations code below to obtain S21 and plot this as a function of frequency

iv) Define a temperature step (say 0.02K) and calculate a new Lint using Mattis Bardeen Approximations at this slightly higher temperature.

v) Simulate again changing only the value of Lint (IDC, L_geo and C_couple remain fixed)

vi) Repeat for all temperatures up to 0.35K

3. Outline of Theory and Task Methodology

Last week discussed about the scattering parameters of the KID. This week, we used the Mattis Bardeen Approximations to determine the population change of the electrons and apply this to L_{int} for the change in internal inductance for a range of temperatures. Following this, the KID can be modelled for a temperature change, corresponding to a detection.

Detailed steps:

i) First, we used the Mattis-Bardeen Approximation for a range of temperatures (from 0.2 to 0.35K with step of 0.02K) and constants defined. Equations:

$$\frac{\sigma_1}{\sigma_n} = \frac{2\Delta(\mathrm{T})}{\hbar\omega}\exp(\left(-\frac{\Delta(0)}{\mathrm{k_B T}}\right)\mathrm{K_0}(\frac{\hbar\omega}{2\mathrm{k_B T}})[2\sinh(\frac{\hbar\omega}{2\mathrm{k_B T}})]$$

$$\frac{\sigma_2}{\sigma_n} = \frac{\pi\Delta(\mathrm{T})}{\hbar\omega}[1-2\exp(\left(-\frac{\Delta(0)}{\mathrm{k_B T}}\right)\exp\left(\frac{-\hbar\omega}{2\mathrm{k_B T}}\right)\mathrm{I_0}(\frac{\hbar\omega}{2\mathrm{k_B T}})]$$

ii) From this, we can calculate the change in σ_2 and σ_1 for a change in temperature.

iii) Following this, we can substitute the values of σ_2 into the LPD expression:

$$\sigma_s = -j\frac{n_s e^2}{\omega m}$$

We take the magnitude of this, so the imaginary number can be ignored. Using this, found values for Cooper-pair popuation n_s.

iv) Then, the value of n_s is substituted into the LPD expression to obtain the LPD:

$$\lambda_L = \sqrt{\frac{m}{\mu_0 n_s e^2}}$$

v) Using the LPD, substitute into the expression for L_{int}:

$$L_{int} = \frac{\mu_0 \lambda}{2} \coth\left(\frac{t}{2\lambda}\right)$$

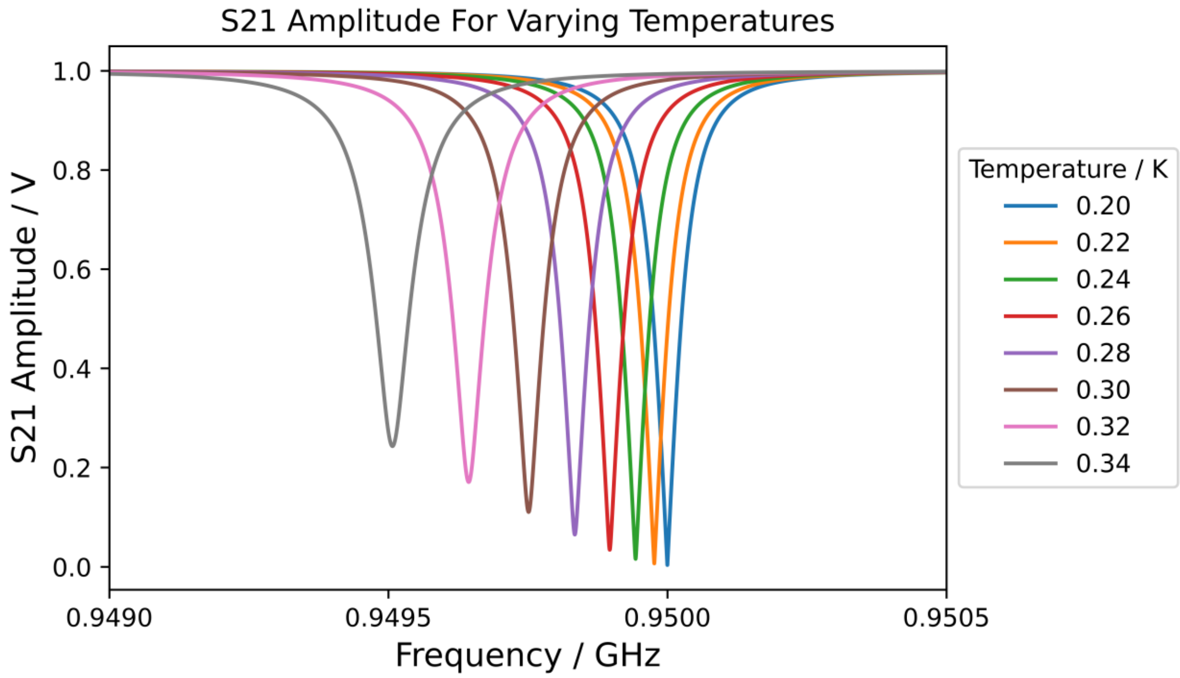
vi) Next, we can find the resistive part R using the following expression:

$$R = L_k \omega \frac{\sigma_1}{\sigma_2}$$

This R accounts for the loss faced by the current when propagating across the quasi-particle as this path is resistive compared to the superconducting n_s.

vii) Finally, plug the R and L_{int} values into the Python function given to obtain S21 and plot the amplitude of S21 against frequency. This figure is generated in the following section.

4. Plot of S21 vs Temperature



This figure illustrates the S21 Amplitude variation with temperature. This is effectively a model of the detector, and this allows us to study and understand the effects of the parameters on its output.

5. Python Function Given:

```
def Capacitive_Res_Sim(F0, C_couple, Z0, L_geo, L_int, Res, Sweep_BW, Sweep_points, Capacitance):
  j=complex(0,1)
  Cc=C_couple
  F_min=F0-(Sweep_BW/2.0)
```

```
    F_max=F0+(Sweep_BW/2.0)
    Sweep=np.linspace(F_min, F_max, Sweep_points)
    W=Sweep*2.0*pi
    W0=2.0*pi*F0
    L=L_geo+L_int
    C=Capacitance
    Zres= 1.0/((1./((j*W*L)+Res))+(j*W*C))   # Impedance of resonator section
    Zc=1.0/(j*W*Cc)  #impedance of coupler
    ZT=Zres+Zc
    YT=1.0/ZT
    S21 = 2.0/(2.0+(YT*Z0))
    I_raw=S21.real
    Q_raw=S21.imag
    shift=((1.0-min(I_raw))/2.0)+min(I_raw)
    I_cent=I_raw-shift
    Q_cent=Q_raw
    Phase=Atan(abs(Q_cent/I_cent))
    QU=(W0*L)/Res
    QL=(C*2)/(W0*(Cc**2)*Z0)
    S21_Volt=abs(S21)
    I_offset=shift
    return  (Sweep, S21_Volt, Phase, I_raw, Q_raw, I_cent, Q_cent, QU, QL, I_offset)
```

6. Python Code for Task

```
#imports
import numpy as np
import matplotlib.pyplot as plt
import scipy.constants as const
from scipy.special import iv as I0
from scipy.special import kv as K0

#Define Global Variables
L_geo = 55.6e-9
Z0 = 50.0
F0_base = 0.95e9       #At lowest Temp
squares= 27223
c_couple = 1.5e-14
TC = 1.5
Delta_0 = (3.5*const.Boltzmann*TC)/2
sigma_n = 6.0e7         # Normal stae conductvity if superconducting film
Thick = 20e-9          # Thickness of superconducting fil
w = 2 * np.pi * F0_base
me = const.m_e
miu_0 = 4*np.pi*10**-7
pi = np.pi

#Main code
def main():
    #Define temperature range with step 0.01K
    step = 0.02
    temp = np.arange(0.20, 0.35, step)
```

```
    #Find sigma1 and sigma 2 and Lint
    sigma1, sigma2 = find_sigma1_sigma2(sigma_n ,Thick, TC, Delta_0, w, temp)
    Lint = find_Lint_square(Thick, w, sigma2) * squares

    #Find lk
    Lk = find_lk(Thick, w, sigma2)

    #Find Res
    sigma12Ratio = sigma1/sigma2
    Res = Lk*w*sigma12Ratio *squares

    #IDC for Lowest Temp (0.2K)
    Ltot_lowest = Lint[0] + L_geo
    IDC = find_IDC(w, Ltot_lowest, c_couple)

    #Find S21
    Sweep_points = 20000
    BW = 5e6
    I_raw = np.zeros((Sweep_points, len(temp)), dtype="float")
    Q_raw = np.copy(I_raw)
    Phase = np.copy(Q_raw)
    S21_Volt = np.copy(I_raw)
    for i in range(0, len(Lint)):
        Sweep, S21_Volt[:,i], Phase[:,i], I_raw[:,i], Q_raw[:,i],_,_,_,_,_ = Capacitive_Res_Sim(F0_base,
c_couple, Z0, L_geo, Lint[i], Res[i], BW, Sweep_points, IDC)
        plt.plot(Sweep/1e9, S21_Volt[:,i], label=str("{:.2f}".format(temp[i])))

    #Graph labels and title
    plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fancybox=True, title="Temperature / K")
    plt.xlabel('Frequency / GHz', fontsize=13)

    plt.ylabel('S21 Amplitude / V', fontsize=13);
    plt.title("S21 Amplitude For Varying Temperatures")
    plt.xlim(0.9490, 0.9505)
    plt.locator_params(nbins=6)
    plt.savefig("S21 Plot with Resistance")
    plt.rcParams['figure.dpi'] = 300
#KID Simulating Function
def Capacitive_Res_Sim(F0, C_couple, Z0, L_geo, L_int, Res, Sweep_BW, Sweep_points,
Capacitance):
    """ Help file here"""
    j=complex(0,1)
    Cc=C_couple
    F_min=F0-(Sweep_BW/2.0)
    F_max=F0+(Sweep_BW/2.0)
    Sweep=np.linspace(F_min, F_max, Sweep_points)
    W=Sweep*2.0*pi
    W0=2.0*pi*F0
    L=L_geo+L_int
    C=Capacitance
```

```
    Zres= 1.0/((1./((j*W*L)+Res))+(j*W*C))   # Impedance of resonator section
    Zc=1.0/(j*W*Cc)  #impedance of coupler
    ZT=Zres+Zc
    YT=1.0/ZT
    S21 = 2.0/(2.0+(YT*Z0))
    I_raw=S21.real
    Q_raw=S21.imag
    shift=((1.0-min(I_raw))/2.0)+min(I_raw)
    I_cent=I_raw-shift
    Q_cent=Q_raw
    Phase=Atan(abs(Q_cent/I_cent))
    QU=(W0*L)/Res
    QL=(C*2)/(W0*(Cc**2)*Z0)
    S21_Volt=abs(S21)
    I_offset=shift
    return  (Sweep, S21_Volt, Phase, I_raw, Q_raw, I_cent, Q_cent, QU, QL, I_offset)

#Function to find sigma1 and sigma2
def find_sigma1_sigma2(sigma_n ,Thick, TC, Delta_0, w, T):
    #An interpolation formula for delta_T
    delta_T = Delta_0*np.tanh(1.74*np.sqrt((TC/T)-1))

    #Define constants to simplify eqn
    multiplying_constant = delta_T/(const.hbar * w)
    e_const_1 = - Delta_0/(const.Boltzmann*T)
    e_const_2 = (const.hbar*w)/(2*const.Boltzmann*T)

    #Parts of the sigma1 Ratio
    A = 2*multiplying_constant
    B = np.exp(e_const_1)
    C = K0(0, e_const_2)
    D = 2*(np.sinh(e_const_2))

    #Find Sigma 1 and Sigma 2
    sigma1Ratio = A * B * C * D
    sigma2Ratio = np.pi*multiplying_constant*(1 - (2*np.exp(e_const_1)*np.exp(-
e_const_2)*I0(0,e_const_2)))
    sigma2 = sigma2Ratio * sigma_n
    sigma1 = sigma1Ratio * sigma_n
    return sigma1, sigma2

def find_lk(Thick, w, sigma2):
    #Depth
    lower_fraction = miu_0*sigma2*w
    Lambda_T_MB = (1/lower_fraction)**0.5
    fraction = Thick/(2*Lambda_T_MB)

    #Terms for lk
    A = (miu_0*Lambda_T_MB)/4
    B = coth(fraction)
    C = fraction*(csch(fraction))**2
```

```
    #R vs T
    lk = A*(B+C)
    return lk

def find_Lint_square(Thick, w, sigma2):
    #Depth
    lower_fraction = miu_0*sigma2*w
    Lambda_T_MB = (1/lower_fraction)**0.5

    #Internal Inductance
    fraction = Thick/(2*Lambda_T_MB)
    L_int = (miu_0*Lambda_T_MB/2)*coth(fraction)
    return L_int

#Define coth and csch
def coth(x):
    return np.cosh(x)/np.sinh(x)

def csch(x):
    return 1/np.sinh(x)

def Atan(x):
    return np.arctan(x)

#Find IDC function
def find_IDC(w0, Ltot, Cc):
    IDC = 1/((w0**2)*Ltot) - Cc
    return IDC

def Magic_Formula(di, dq, didf, dqdf):
    return (di*didf + dq*dqdf)/(didf**2 + dqdf**2)

main()
```

Week 7: 17/11/2021 – Wednesday

1. **Outline of Meeting**

The meeting was carried out on Zoom. The meeting was dedicated to explaining how we move forward with the model accomplished from the previous week. The project now moves onto reading out the measurement from the detector. This was accomplished by using the dF0 formula. Further details will be explained in the Outline of theory and methodology section. It essentially allows us to relate the detector output of S21 and the signal to measure.

2. **Specification of Tasks**

i) Add resistance to the ABCD KID model you have already made

ii) Take your model, and make Q vs I plots for the resonator at each temperature (I= real part of S21, Q= imaginary part of S21)

iii) For your lowest temperature KID, note F0 and the I and Q values at F0. Lets call this F0 F0_base

iv) add the I and Q values of each KID at the frequency F0_base to your Q vs I sweep plots as a symbol.

v) Make another plot of F0 vs temperature. Calculate F0 of each resonance as from the frequency corresponding to the minimum in S21 magnitude.

vi) Using the "Magic formula" equation equation 6 from the understanding KID readout document, plot the F0 calculated from this formula. Do this by looking at I and Q at F0 base and calculating df0 from the formula. The dI/dF and dQ/dF data should be calculated from the lowest temperature sweep. This will tell use the range over which this formula is valid

vii) calculate di/df and dQ by df at F0 for the lowest temperature sweep these will be single values

viii) on subsequent S21 data (taken at higher temperatures, plug in the vales for dI this is the change in I In the new temperature from the I at F0_base. Do the same for Q and this should give you a change in F0

3. **Outline of Theory and Methodology**

A KID typically measures the signal and outputs a I and Q value in units of Volts. This may not be entirely intuitive, as a voltage from an electronic circuit does not give much information in terms of its response to the detection.

The solution to this is therefore known as the $dF0$ formula. This formula allows us to convert I and Q values from the Sweep data (Data across the frequency domain) into a change in tone frequency F_0. The formula is given as follows:

$$\partial F_0 = \frac{\partial I(t)\frac{\partial I}{\partial F} + \partial Q(t)\frac{\partial Q}{\partial F}}{\left(\frac{\partial I}{\partial F}\right)^2 + \left(\frac{\partial Q}{\partial F}\right)^2}$$

where $\partial I(t)$ and $\partial Q(t)$ are the changes in I and Q values from the time stream data against the I and Q from the Sweep data, for a given time. In the case of the model, it is the change in I and Q for a given temperature compared to the base temperature at the tone frequency. $\left(\frac{\partial I}{\partial F}\right)$ and $\left(\frac{\partial Q}{\partial F}\right)$ are the numerical derivates of the minimum of the sweep data (For the model, the base temperature). This allows us to calculate a value for the change in the tone frequency.

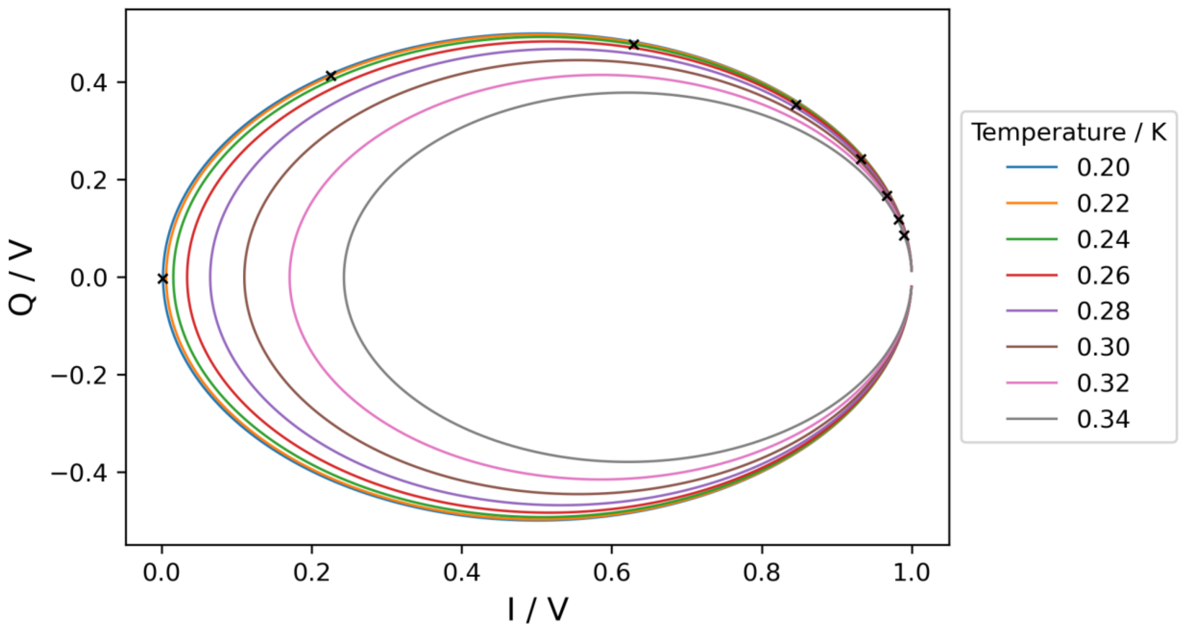
Moving on to the task, we first plot the I and Q values as given in section 4, the cross represents the I and Q value at the tone frequency, we can clearly observe that the I and Q values have changed.

Following this, we can find $\left(\frac{\partial I}{\partial F}\right)$ and $\left(\frac{\partial Q}{\partial F}\right)$ by finding the numerical derivate of the minimum of S21 at the base temperature. Then, we can find the change in I and Q values $\partial I(t)$ and $\partial Q(t)$ for temperatures above the base temperature by subtracting the I and Q values for the respective temperature from the I and Q of the base temperature, at the tone-frequency. (e.g. if we set tone at 0.95 GHz, find the I and Q values at 0.95GHz for both the base and new temperature and subtract each other).

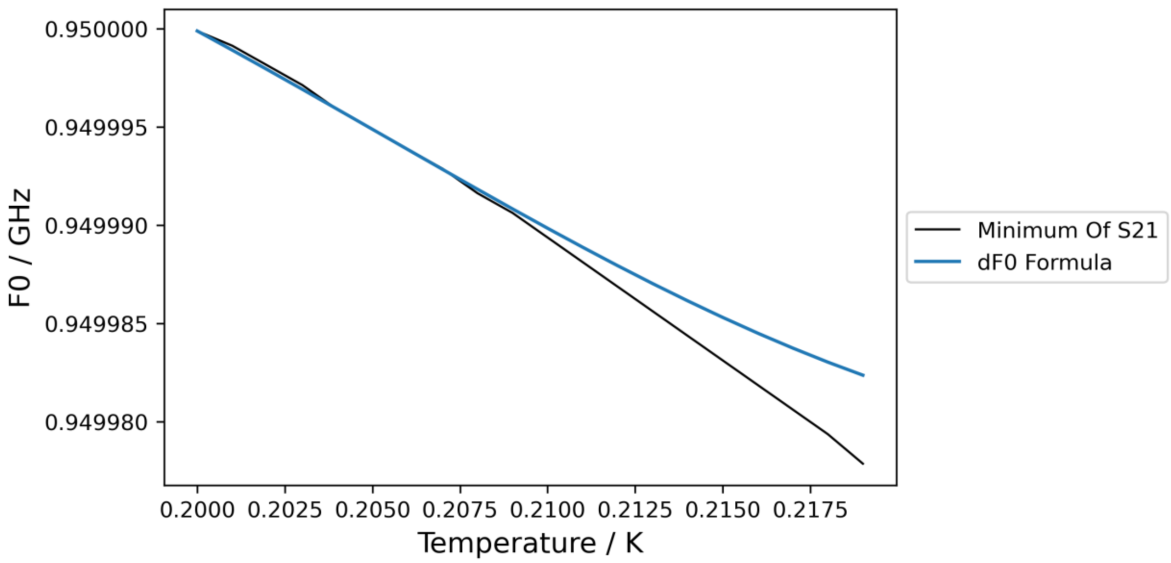
Then, we can use these values and plug them into the dF0 formula to obtain a dF0. We can subtract dF0 for each temperature from F0 and plot F0 against temperature. We can also find F0 for the minimums of S21 for varying temperatures and plot this along with dF0 formula. This is shown in section 5. The graph shows that the ∂F_0 formula holds remarkably well up till 0.21K. As such, the ∂F_0 formula is appropriate for low temperature variations, therefore the maximum change in F_0 for the approach to still be valid is $\pm\ 10000Hz$.

Important to note, the dF0 formula is quite important for characterising the response of the detector. This is because the dF0 allows the conversion of I and Q values from the output of the KID into a change in F0. We can then characterize the response as a change in F0 against a change in frequency, $dF0/dP$. This will be explored when we come to responsivity measurements in semester 2.

4. Plot of I vs Q



5. Plot of F0 against Temperature for dF0 formula and Minimum of S21



6. Python Code for Task

```
#imports
import numpy as np
import matplotlib.pyplot as plt
import scipy.constants as const
from scipy.special import iv as I0
from scipy.special import kv as K0

#Define Global Variables
```

```
L_geo = 55.6e-9
Z0 = 50.0
F0_base = 0.95e9      #At lowest Temp
squares= 27223
c_couple = 1.5e-14
TC = 1.5
Delta_0 = (3.5*const.Boltzmann*TC)/2
sigma_n = 6.0e7       # Normal stae conductvity if superconducting film
Thick = 20e-9         # Thickness of superconducting fil
w = 2 * np.pi * F0_base
me = const.m_e
miu_0 = 4*np.pi*10**-7
pi = np.pi

#Main code
def main():
    #Define temperature range with step 0.01K
    step = 0.02
    temp = np.arange(0.20, 0.35, step)

    #Find sigma1 and sigma 2 and Lint
    sigma1, sigma2 = find_sigma1_sigma2(sigma_n ,Thick, TC, Delta_0, w, temp)
    Lint = find_Lint_square(Thick, w, sigma2) * squares

    #Find lk
    Lk = find_lk(Thick, w, sigma2)

    #Find Res
    sigma12Ratio = sigma1/sigma2
    Res = Lk*w*sigma12Ratio *squares

    #IDC for Lowest Temp (0.2K)
    Ltot_lowest = Lint[0] + L_geo
    IDC = find_IDC(w, Ltot_lowest, c_couple)

    #Find S21
    Sweep_points = 20000
    BW = 5e6
    I_raw = np.zeros((Sweep_points, len(temp)), dtype="float")
    Q_raw = np.copy(I_raw)
    Phase = np.copy(Q_raw)
    S21_Volt = np.copy(I_raw)
    for i in range(0, len(Lint)):
        Sweep, S21_Volt[:,i], Phase[:,i], I_raw[:,i], Q_raw[:,i],_,_,_,_,_ = Capacitive_Res_Sim(F0_base,
c_couple, Z0, L_geo, Lint[i], Res[i], BW, Sweep_points, IDC)
        plt.plot(Sweep/1e9, S21_Volt[:,i], label=str("{:.2f}".format(temp[i])))

    #Graph labels and title
    plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fancybox=True, title="Temperature / K")
    plt.xlabel('Frequency / GHz', fontsize=13)
```

```
plt.ylabel('S21 Amplitude / V', fontsize=13);
plt.title("S21 Amplitude For Varying Temperatures")
plt.xlim(0.9490, 0.9505)
plt.locator_params(nbins=6)
plt.savefig("S21 Plot with Resistance")
plt.rcParams['figure.dpi'] = 300
plt.figure()

#Q vs I plots
for i in range(0, len(Lint)):
   plt.plot(I_raw[:,i], Q_raw[:,i], linewidth=1,label=str("{:.2f}".format(temp[i])))

#Minimum S21 at lowest temp
S21_Base = min(S21_Volt[:,0])
I_Base = np.zeros(len(temp), dtype="float")
Q_Base = np.copy(I_Base)

#Obtain F0_base and I and Q values for Lowest Temp
for i in range(0, len(S21_Volt[:,0])):
   if S21_Base == S21_Volt[i,0]:
      F0_Base = Sweep[i]

#Plot I and Q values at F0_Base
for i in range(0, len(temp)):
   for j in range(0, len(Sweep)):
      if F0_Base == Sweep[j]:
         I_Base[i] = I_raw[j,i]
         Q_Base[i] = Q_raw[j,i]
         plt.plot(I_Base[i], Q_Base[i], markersize=4, marker="x", color='black')

#labels
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fancybox=True, title="Temperature / K")
plt.xlabel('I / V', fontsize=13)
plt.ylabel('Q / V', fontsize=13);
#plt.title("Q vs I Plot for Varying Temperature")
plt.savefig("Q vs I plot for varying temp")
plt.figure()

#Finding F0 for the different Temperatures
F0 = np.zeros(len(temp))
for i in range(0, len(temp)):
   S21_min = min(S21_Volt[:,i])
   for j in range(0, len(Sweep)):
      if S21_min == S21_Volt[j,i]:
         F0[i] = Sweep[j]

#Plotting F0 vs Temp
plt.plot(temp, F0/1e9, color='k', linewidth="1", label="Minimum Of S21")
plt.xlabel('Temperature / K', fontsize=13)
plt.ylabel('F0 / GHz', fontsize=13);
plt.rcParams['figure.dpi'] = 300
```

```
    #plt.title("F0 vs Temperature")

    #Finding dI/dF and dQ/dF for lowest temperature
    #Using numerical derivatives
    step = abs((Sweep[0]-Sweep[-1])/Sweep_points)
    for i in range(0, len(Sweep)):
        if Sweep[i] == F0_Base:
            didf = (I_raw[i+1,0] - I_raw[i-1,0])/(2*step)
            dqdf = (Q_raw[i+1,0] - Q_raw[i-1,0])/(2*step)

    #Use Magic Formula
    di = np.zeros(len(temp))
    dq = np.copy(di)
    di = abs(I_Base - I_Base[0])
    dq = abs(Q_Base - Q_Base[0])
    dF0 = Magic_Formula(di, dq, didf, dqdf)

    #Find F0 for different temp
    F0_Magic = F0_Base - abs(dF0)
    plt.plot(temp, F0_Magic/1e9, label="dF0 Formula")
    plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fancybox=True)
    plt.ticklabel_format(useOffset=False)
    plt.rcParams['figure.dpi'] = 1000
    plt.xlim(0.20, 0.22)
    plt.ylim(0.949980, 0.95)
    plt.savefig("Magic Formula plot")

#KID Simulating Function
def Capacitive_Res_Sim(F0, C_couple, Z0, L_geo, L_int, Res, Sweep_BW, Sweep_points,
Capacitance):
    """ Help file here"""
    j=complex(0,1)
    Cc=C_couple
    F_min=F0-(Sweep_BW/2.0)
    F_max=F0+(Sweep_BW/2.0)
    Sweep=np.linspace(F_min, F_max, Sweep_points)
    W=Sweep*2.0*pi
    W0=2.0*pi*F0
    L=L_geo+L_int
    C=Capacitance
    Zres= 1.0/((1./((j*W*L)+Res))+(j*W*C))   # Impedance of resonator section
    Zc=1.0/(j*W*Cc)  #impedance of coupler
    ZT=Zres+Zc
    YT=1.0/ZT
    S21 = 2.0/(2.0+(YT*Z0))
    I_raw=S21.real
    Q_raw=S21.imag
    shift=((1.0-min(I_raw))/2.0)+min(I_raw)
    I_cent=I_raw-shift
    Q_cent=Q_raw
    Phase=Atan(abs(Q_cent/I_cent))
```

```
    QU=(W0*L)/Res
    QL=(C*2)/(W0*(Cc**2)*Z0)
    S21_Volt=abs(S21)
    I_offset=shift
    return  (Sweep, S21_Volt, Phase, I_raw, Q_raw, I_cent, Q_cent, QU, QL, I_offset)

#Function to find sigma1 and sigma2
def find_sigma1_sigma2(sigma_n ,Thick, TC, Delta_0, w, T):
    #An interpolation formula for delta_T
    delta_T = Delta_0*np.tanh(1.74*np.sqrt((TC/T)-1))

    #Define constants to simplify eqn
    multiplying_constant = delta_T/(const.hbar * w)
    e_const_1 = - Delta_0/(const.Boltzmann*T)
    e_const_2 = (const.hbar*w)/(2*const.Boltzmann*T)

    #Parts of the sigma1 Ratio
    A = 2*multiplying_constant
    B = np.exp(e_const_1)
    C = K0(0, e_const_2)
    D = 2*(np.sinh(e_const_2))

    #Find Sigma 1 and Sigma 2
    sigma1Ratio = A * B * C * D
    sigma2Ratio = np.pi*multiplying_constant*(1 - (2*np.exp(e_const_1)*np.exp(-
e_const_2)*I0(0,e_const_2)))
    sigma2 = sigma2Ratio * sigma_n
    sigma1 = sigma1Ratio * sigma_n
    return sigma1, sigma2

def find_lk(Thick, w, sigma2):
    #Depth
    lower_fraction = miu_0*sigma2*w
    Lambda_T_MB = (1/lower_fraction)**0.5
    fraction = Thick/(2*Lambda_T_MB)

    #Terms for lk
    A = (miu_0*Lambda_T_MB)/4
    B = coth(fraction)
    C = fraction*(csch(fraction))**2

    #R vs T
    lk = A*(B+C)
    return lk

def find_Lint_square(Thick, w, sigma2):
    #Depth
    lower_fraction = miu_0*sigma2*w
    Lambda_T_MB = (1/lower_fraction)**0.5

    #Internal Inductance
```

```
    fraction = Thick/(2*Lambda_T_MB)
    L_int = (miu_0*Lambda_T_MB/2)*coth(fraction)
    return L_int

#Define coth and csch
def coth(x):
    return np.cosh(x)/np.sinh(x)

def csch(x):
    return 1/np.sinh(x)

def Atan(x):
    return np.arctan(x)

#Find IDC function
def find_IDC(w0, Ltot, Cc):
    IDC = 1/((w0**2)*Ltot) - Cc
    return IDC

def Magic_Formula(di, dq, didf, dqdf):
    return (di*didf + dq*dqdf)/(didf**2 + dqdf**2)

main()
```

Week 8: 24/11/2021 – Wednesday

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**

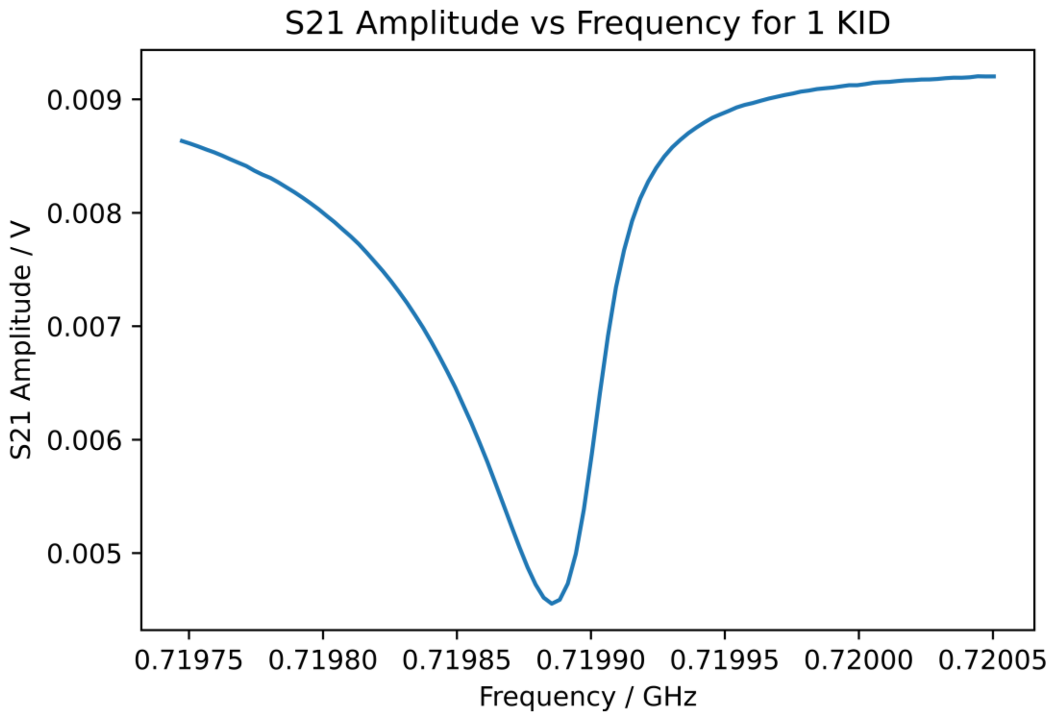
i) Using the FSAB data file, readout out the sweep and timestream data.
ii) Plot the sweep and timestreams data

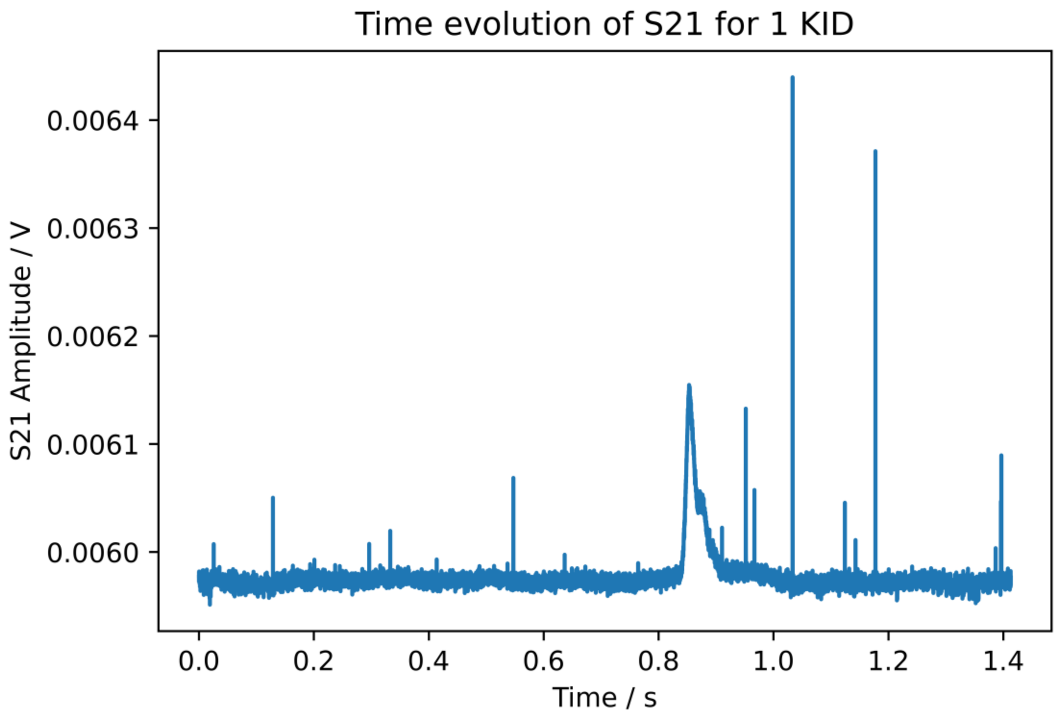
3. **Outline and Methodology**

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")
```

<u>Week 9: 1/12/2021 – Wednesday</u>

1. **Outline of Meeting**

This meeting was held on Zoom. It was dedicated to outline how we are going to move forward with noise analysis. Going back to the previous weeks, we can now use the dF0 formula and data measurements to do readings on noise. This can be accomplished by taking measurements of a known temperature object (e.g. a heated bar) and scanning across to measure the room (background). This gives us information on the responsivity and hence the NEP.

2. **Specification of Tasks**

a) **Noise Spectral Density:**

i) 1. Load code and understand how to produce a power spectrum and noise spectral density in Python

ii) 2. Use the Syntax to make similar spectral density plots for real KID data (need to convert real (I) and Imaginary (Q) data to a df time stream using sweep data)

iii) 3. Measure response of KID data for each KID. dI/dF + dQ/dF

b) **Interim Report**

i) Produce a skeleton and slides as a plan for the report.

Advised to prioritise this over the Noise spectral Density Task

3. **Outline and Methodology**

Building on the previous week's work. We are now able to measure the response of the KID using the data from last week. We do this by first finding the dI/dF0 and dQ/dF0 using the sweep data by obtaining the numerical derivative at the minimum point. Then, using each point on the timestream data as I(t) and Q(t), plug these values along with the dI/dF0 and dQ/dF0 to obtain an dF0. Subtract dF0 from the tone frequency and plot for varying times.

Prioritize Interim Report Planning. This can be accomplished after Interim Report submitted

Week 10: 8/12/2021 – Wednesday

1. **Outline of Meeting**

The meeting was held on Zoom. The meeting was focused on going through the slides for the plan for the interim report and the skeleton of it. It was discussed how to better structure the report, what to emphasize, and fitting in the aims and objectives. It was also discussed that the slides can be used for the Viva for next year. The draft report was agreed to be compiled by next week so that feedback can be given.

2. **Outline of Tasks**

i) Create a draft of the Interim Report by next week

The discussing of the interim report and how it should be structured, and which parts to emphasize took up most of the meeting and as such, not much in terms of theory was discussed in this meeting.

Week 11: 13/12/2021 – Monday

1. **Outline of Meeting**

The meeting was held on Zoom. The meeting was focused on going through the draft Interim Report for submission on Friday the 17th. My supervisor provided valuable feedback on what could be improved in the report. In terms of the word count, my report went quite over the limit and my supervisor gave me pointers on which parts to leave out and which parts to simplify and shorten. He also gave advice on how to structure the information so that it would flow smoother for the reader. It was also covered how the referencing could be improved to save words and better worded. Some misconceptions on the use of the dF0 formula was also covered. Overall, most of the meeting was spent on constructive feedback on the report.

SPRING DIARY ENTRIES

Ashley Thean

Week 1 - Week commencing 7/2/2022

Week Outline

In-person meeting to visit the camera, take simple measurements and rediscuss the aims and objectives of the project and how the project will move forward. This week is mostly a refresher into the project as this semester will focus more on analyzing experimental detector data than understanding any additional theory. The tasks for this week is to understand what the aims and objectives of the project moving forward and read in I and Q values, and attempt to convert to dF0 using the "Magic Formula". In addition, some photos and simple measurements were taken for future use.

Complete Python Code Attached at the End of Diary

Tasks Outline

- Obtain and read-in detector data
- Convert I & Q values to dF0 using "Magic Formula"
- Plot dF0 vs Time

Simple Measurements

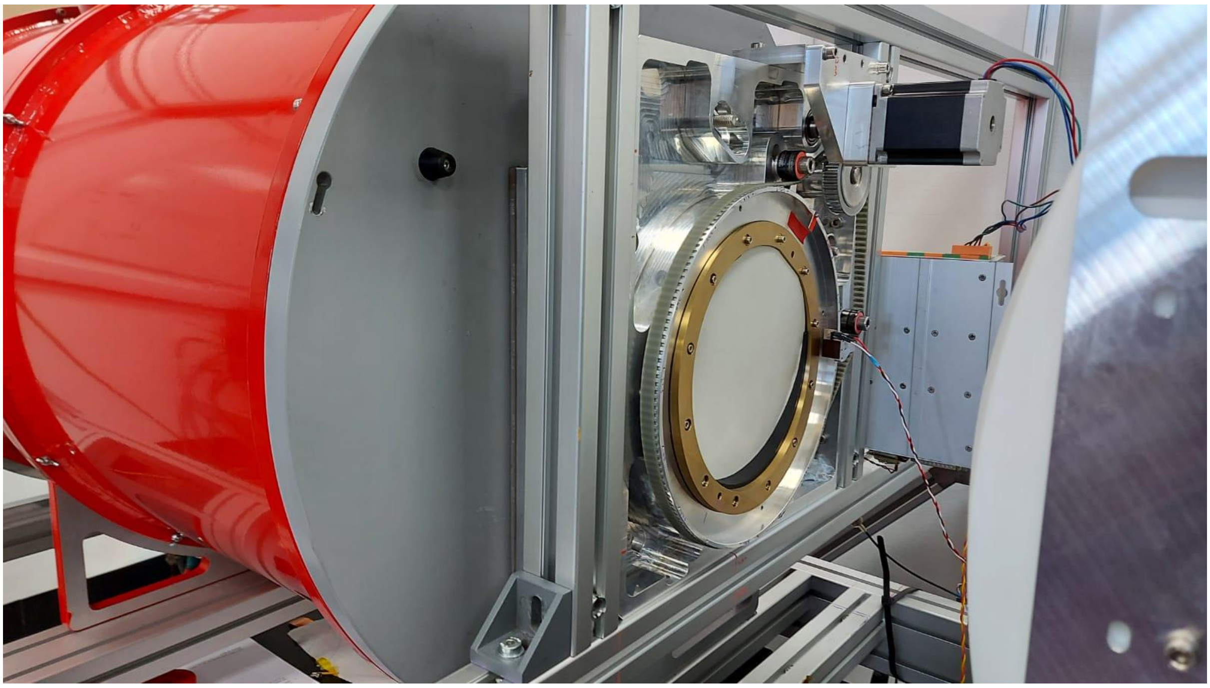
$$T_{hotbar} = 41.3\ °C$$

$$T_{surrounding} = 22.7\ °C$$

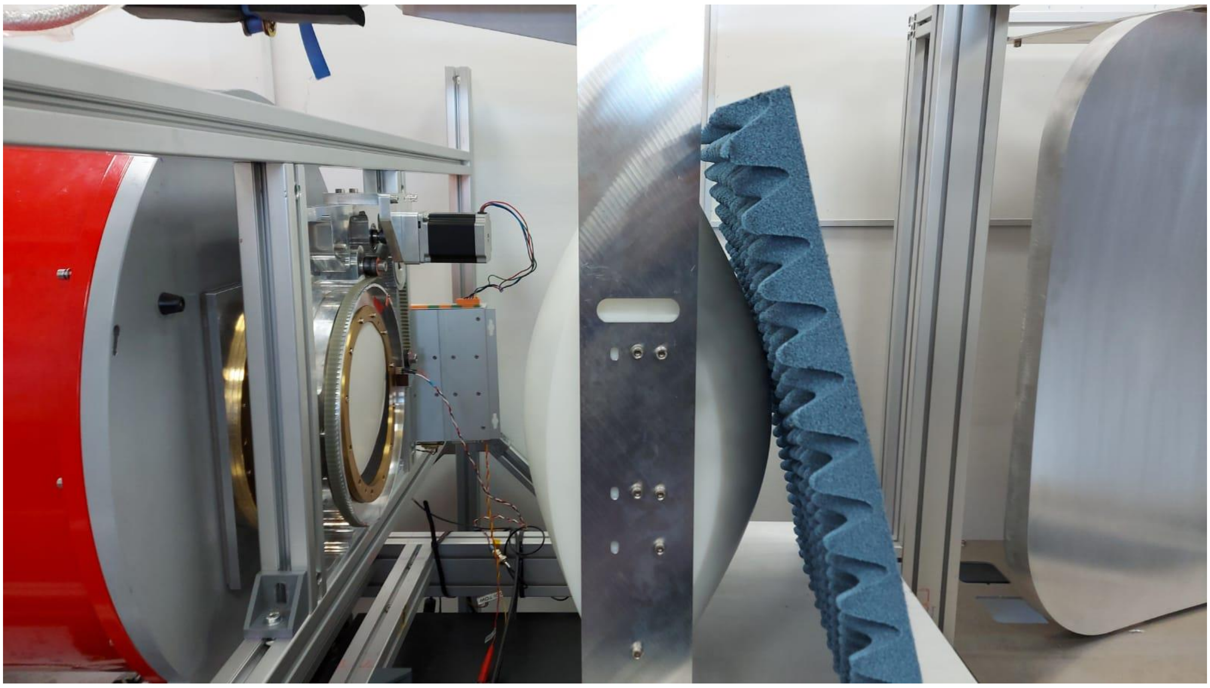
$$length\ of\ hotbar,\ L_{hotbar} = 100\ mm$$

$$width\ of\ hotbar,\ W_{hotbar} = 20\ mm$$

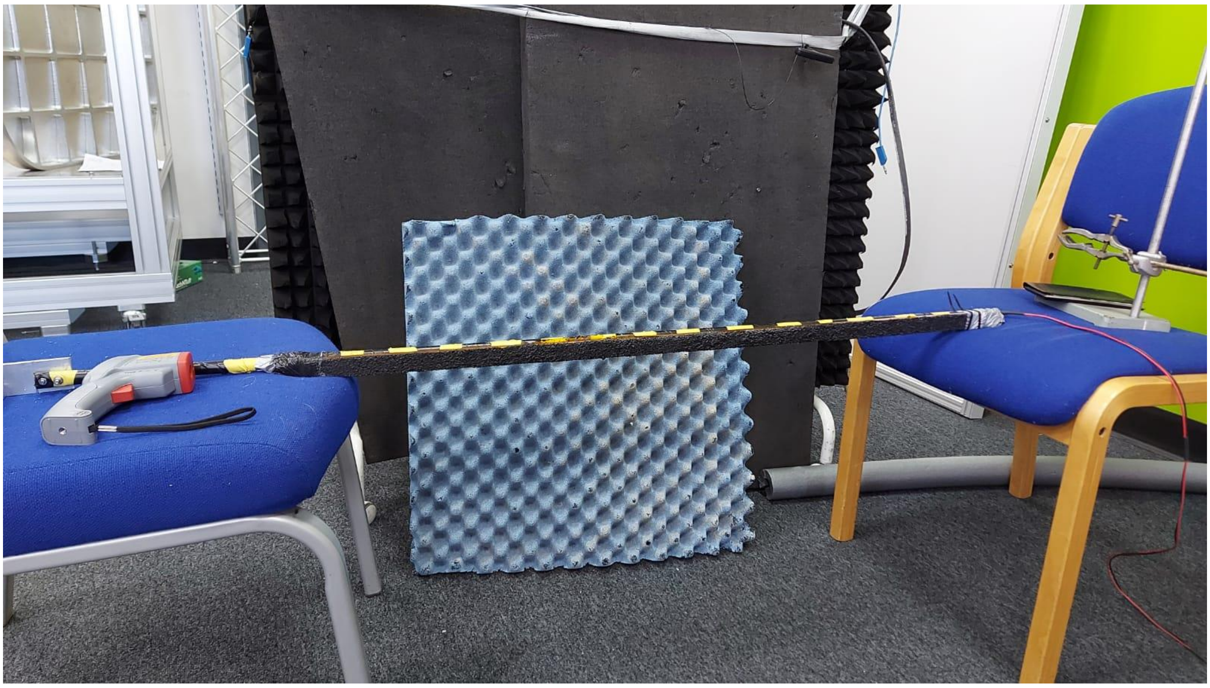
Photos of Camera Setup



Camera. The red drum contains the detector inside held at ~300mK.



Detector with foam screen.

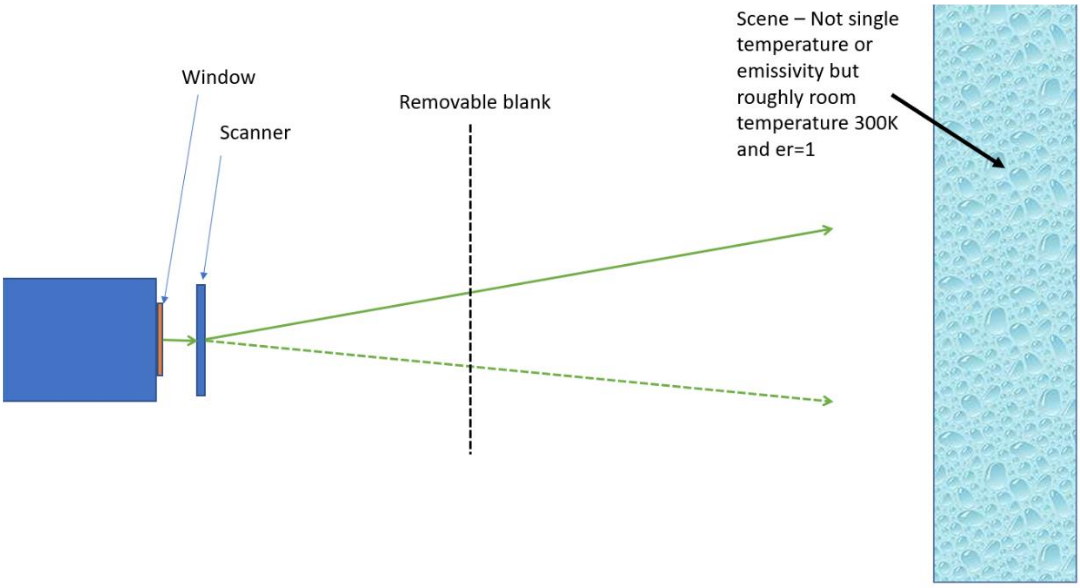


Setup of hot bar, heated electrically.

Measured Data

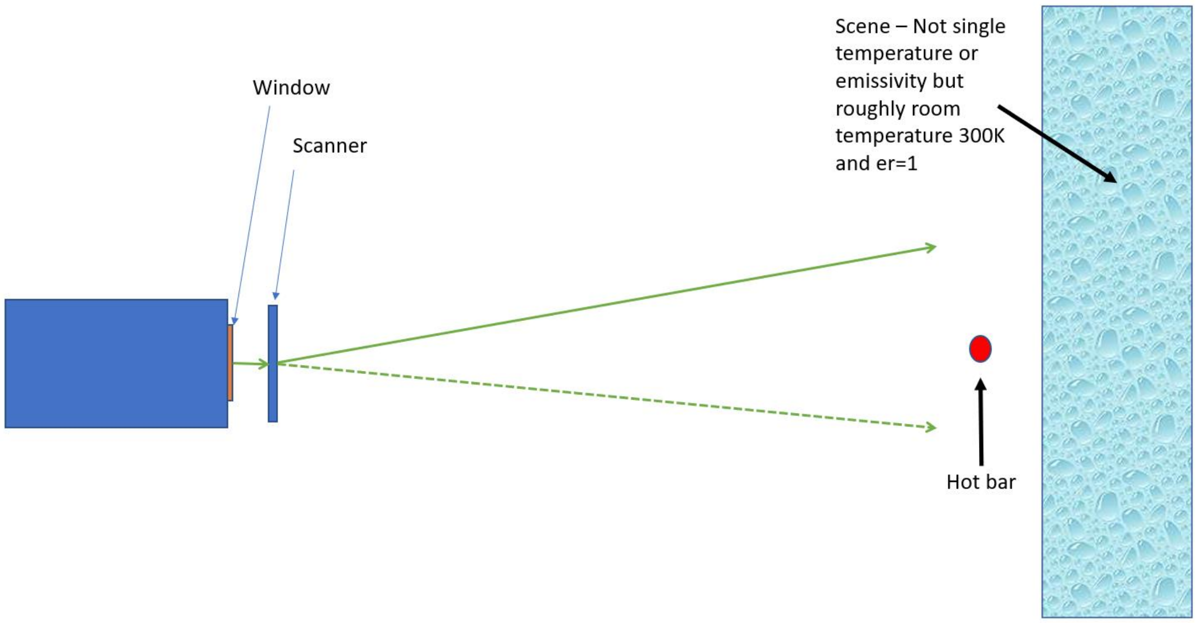
For the detector data, there are 2 sets of detector data that will be used throughout the project for analysis:

1. Measurement of a blank screen.



This data allows us to measure the background for noise analysis.

2. Measurement with a hot bar



This data allows us to characterize the response of the detector, with known temperature of the hotbar.

Week 2 - Week commencing 14/2/2022

Week Outline

Online meeting to discuss the next step of the data analysis and to clear up misconceptions about obtaining dF0 and responsivity of the KID. As explained, the next step is to read-in I and Q values and converting them to a dF0. Mostly as a continuation of last week's data and camera visit and project overview. By doing so, we will be able to use these dF0 values in finding a response by the detector.

Complete Python Code Attached at the End of Diary

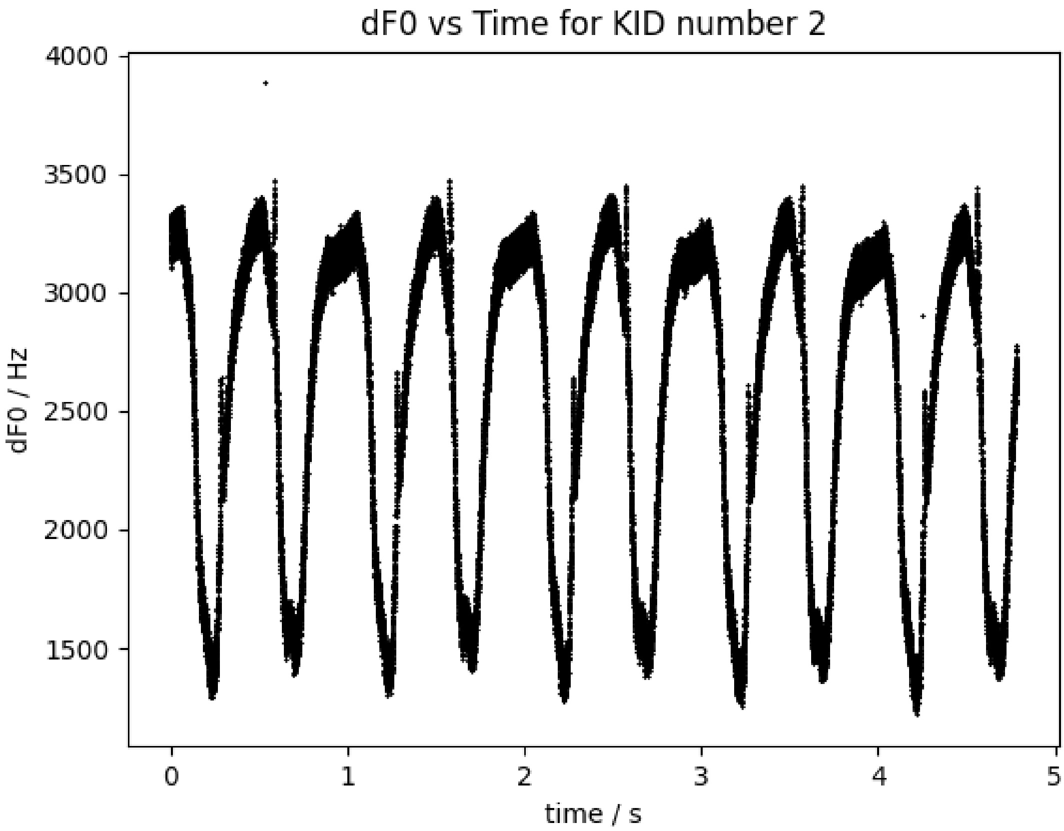
Tasks Outline

- Convert I and Q values to dF0 using the "Magic Formula" for hot bar data:

$$dF_0 = \frac{dI(t)\frac{dI}{dF} + dQ(t)\frac{dQ}{dF}}{\frac{dI}{dF}^2 + \frac{dQ}{dF}^2}$$

- Identify hot bar within data.

Graph of dF0 vs Time



This is **detector data for the hot bar for KID Number 2**. here are 2 distinct features visible, the large oscillating wave is the reflection of the detector screen as it oscillates. The small peaks present on the large waves are the hot bar. As the detector scans up and down, the hot bar falls in the camera's field of view momentarily. The hot bar will have a Gaussian curve with a peak height as the detector takes an integration time to "warm up" to the detection of the hot bar and it "cools down" again as it moves away.

One would expect the oscillating features to be symmetrical as it scans up and down or at a different time, however this is not the case. For example, the reflection and hot bar peaks and curves are not identical to itself in a different time. One of the main contributing reasons is of course the background as it is measured in a "un-ideal" scene with many furniture and objects, at fluctuating "room temperature", which may annoyingly "add extra features" to the curves.

The other reason is that the detector temperature fluctuates. One can imagine due to the internal electronics and the mechanism to regulate the heat itself may produce heat, and other factors such as temperature fluctuations may cause the detector temperature to not remain constant. The detector temperature was advised to only remain somewhat constant for around ~30 seconds before it deviates significantly. This may lead to some additional features to the curves as well.

Week 3 - Week commencing 21/2/2022

Outline

Online meeting to discuss the next step of the project: calculating the response and Noise – equivalent temperature (NET). Following from last week, the dF0 can now be used to find a response of the detector. Using the measurements from week 1, the hot bar is a known temperature T. We can obtain a responsivity based on dF0/dT.

The dF0 can be calculated by identifying the peak of hot bar that takes the shape of a Gaussian curve, where the location and where it is is explained in the previous week. The peak can have a Gaussian curve fitted on it and the peak height can be found, this is the dF0.

Next, the dT can be found by determining the change in temperature of the hotbar and the room. This is just the difference between the temperatures measured in week 1. The response can simply be calculated from these values and the process can be repeated for the other KIDs.

Finally, the NET is related to the response by:

$$NET = \frac{\sqrt{e_n}}{Response}$$

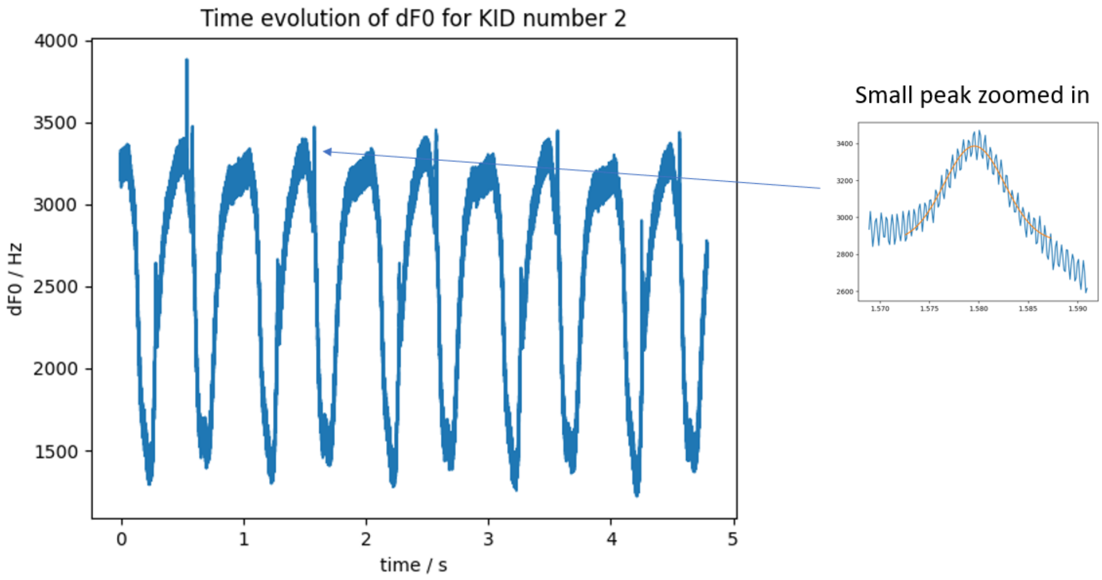
Where e_n is the spectral density. This value can be found by taking the Fourier Transform of the dF0 values and finding the intensities of each frequency bin, then use this value to calculate NET.

Complete Python Code Attached at the End of Diary

Tasks Outline

- Find the peak height of the hot bar feature.
- Calculate a response using the peak height and hotbar temperature
- Take the Fourier Transform of the dF0 data to find the spectral densities
- Find NET using spectral densities and response

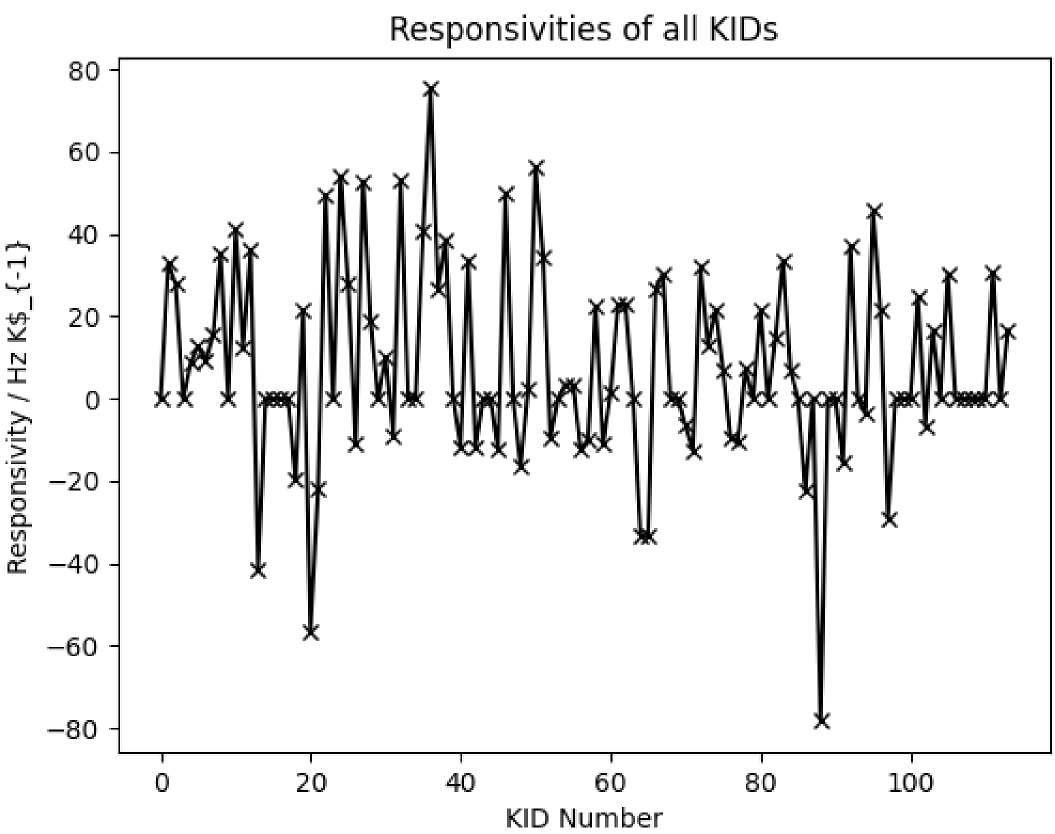
Curve Fit of Hot Bar Curve



A Gaussian was fitted to the hot bar feature of KID 2 and using the Python's curve fit function, the parameters for the height of the curve can be found, which is the peak height.

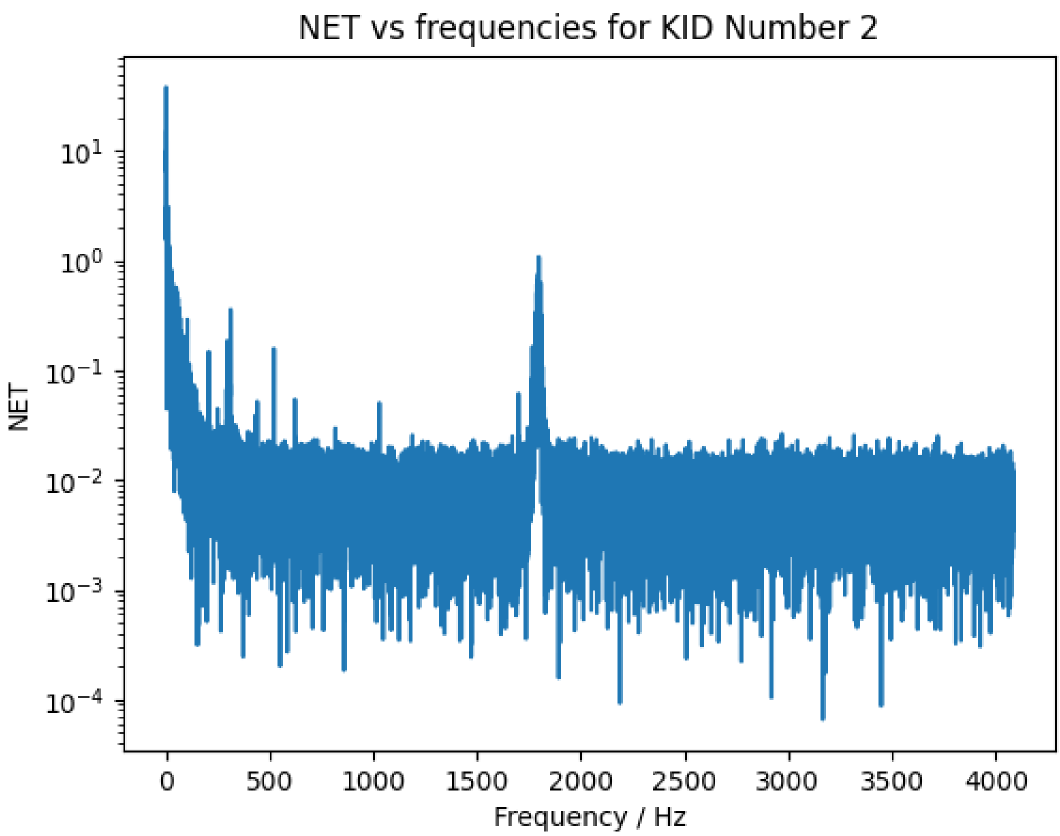
The response for KID 2 was found to be 27.5 Hz/K

Responsivities of all KIDs Graph



The process was looped in Python for the other KIDs. The values exceeding a magnitude of 100 was set to 0. This was probably due to a change in the shape of the curve causing distortions. This can be fixed in a future week.

Graph of NET vs Frequency for KID 2



The Fourier Transform of the dF0 data for KID 2 was taken and the spectral densities found. Then, using the densities, found NET using the mentioned formula. NOTE: NET =/= NEP. To find NEP, calculate power of bar. 1/f noise and hotbar feature evident. The white noise level is also deducible.

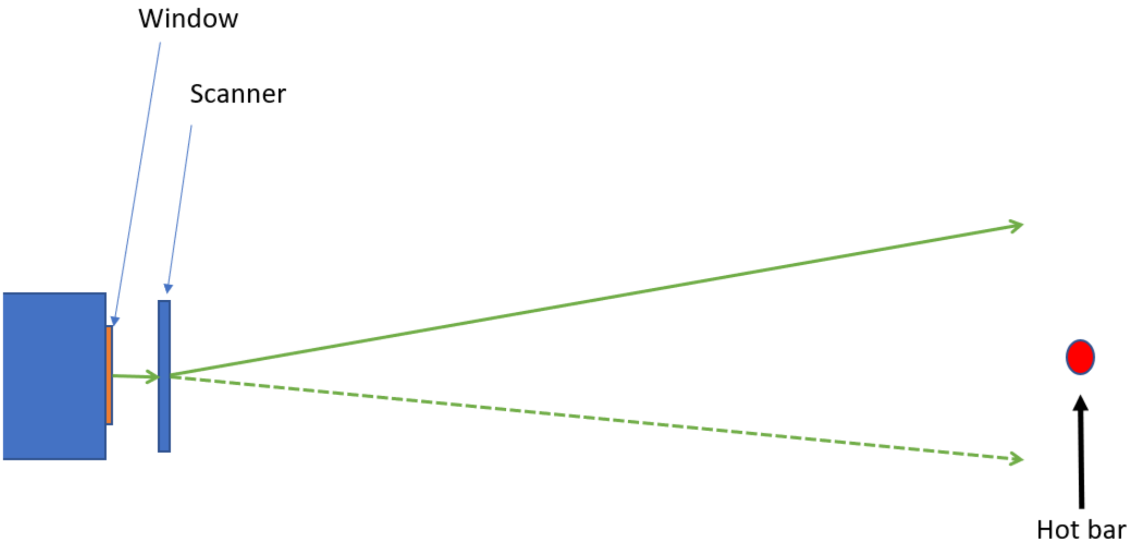
Week 4 - Week commencing 28/2/2022

Week Outline

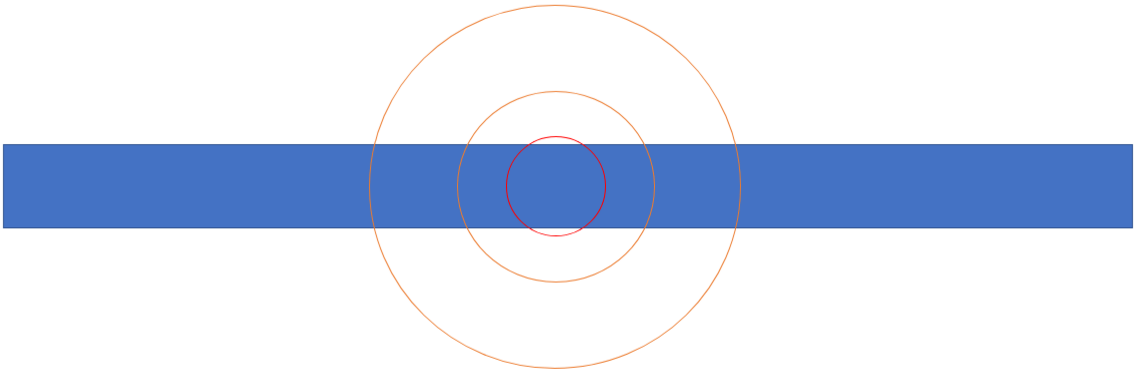
Online meeting discussing the implications and further analysis of the beam filling factor on the measured data from the detector.

The detector has a large solid angle, and thus the full beam of the detector is not entirely comprised of the radiance of the hot bar. An example diagram is given below:

Side view

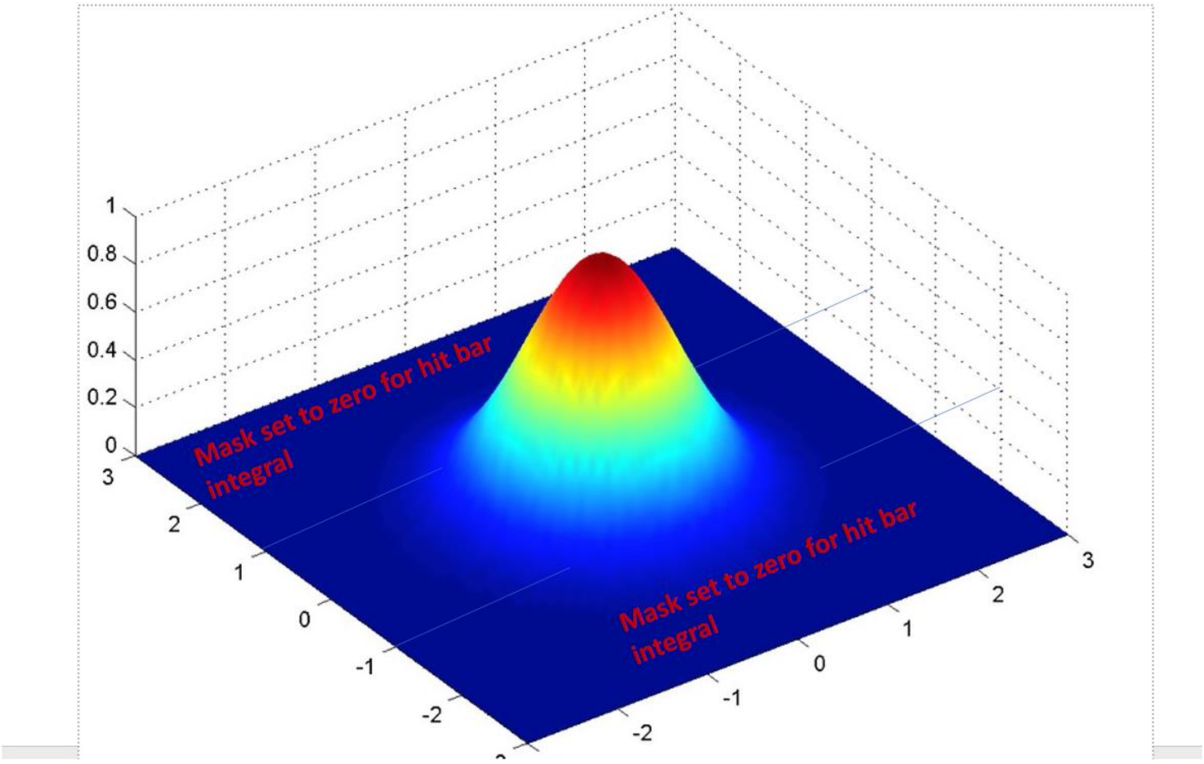


Front View



where the red circle is the beam. As observed, the beam covers parts of the detection that goes beyond the hotbar, and you can imagine that the 2-D Gaussian dF0 peak that forms from the detection for a given detector data at a given time comprises of “addition of power” from background sources and that not 100% of the peak of the curve used for calculating the response corresponds to the hot bar. This leads to the idea of a beam filling factor.

The 2D Gaussian corresponding to the detection of the hot bar can be modelled in such a way that if we use and “cut” a 2D Gaussian of length^2, to only the dimensions of the hot bar and set the outside of the curve to equal to 0. This is shown below:



We can then take the sum of the “cut” Gaussian curve and the “uncut” Gaussian curve and take the ratio of them, this gives a “Beam Filling Factor”:

$$BFF = \frac{\sum hot\ bar\ Gaussian}{\sum 2D\ Gaussian}$$

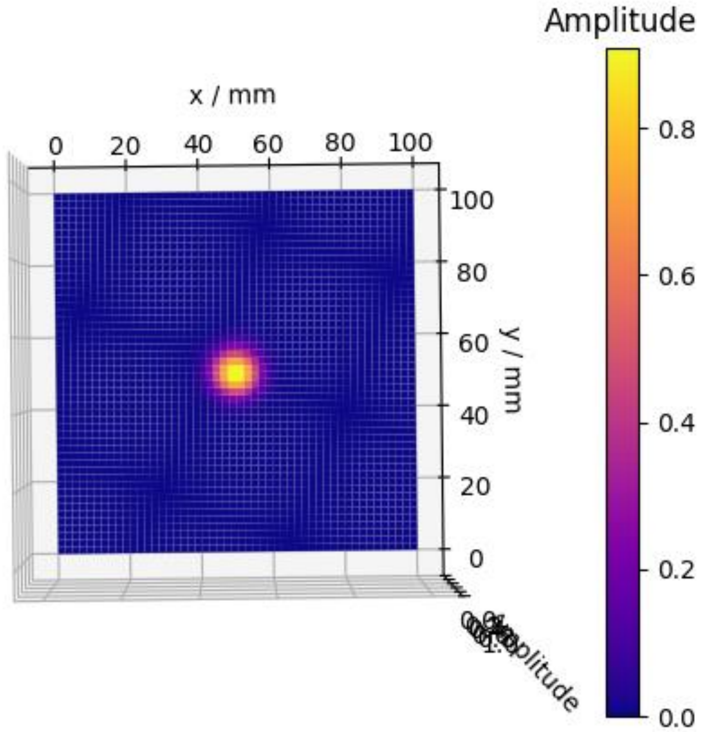
This beam filling factor essentially gives a ratio of the beam that is filled by the hot bar. This will be essential in calculating the power of the hot bar in the following weeks to find the NEP.

<u>Complete Python Code Attached at the End of Diary</u>

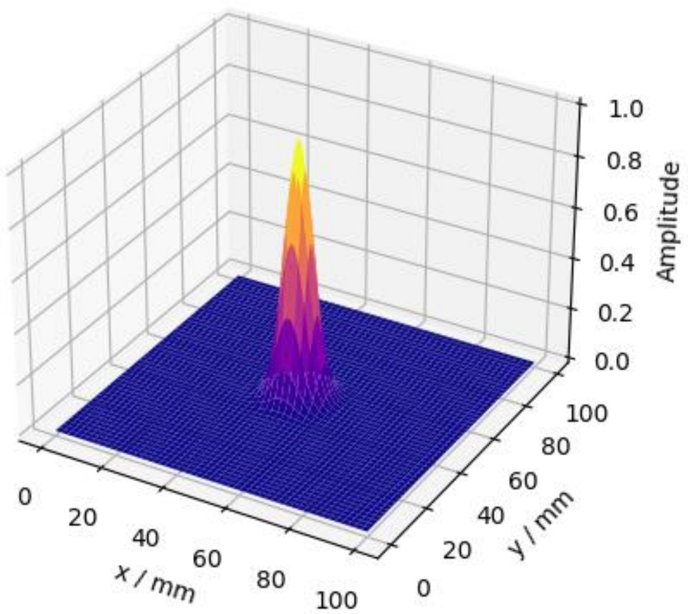
<u>Tasks Outline</u>

- Create a 2D Gaussian Curve with amplitude 1, fwhm = width of hot bar and plot this for the length of the hot bar squared.
- Take the sum of this Gaussian as Gaussian_Sum
- Set the points outside the hot bar dimensions as equal to 0
- Take the sum of this new “cut” Gaussian.
- Take the ratio of Gaussian_Sum to “cut” Gaussian sum as the Beam Filling Factor.

Top Down View of the cut Gaussian



Side View of the Cut Gaussian

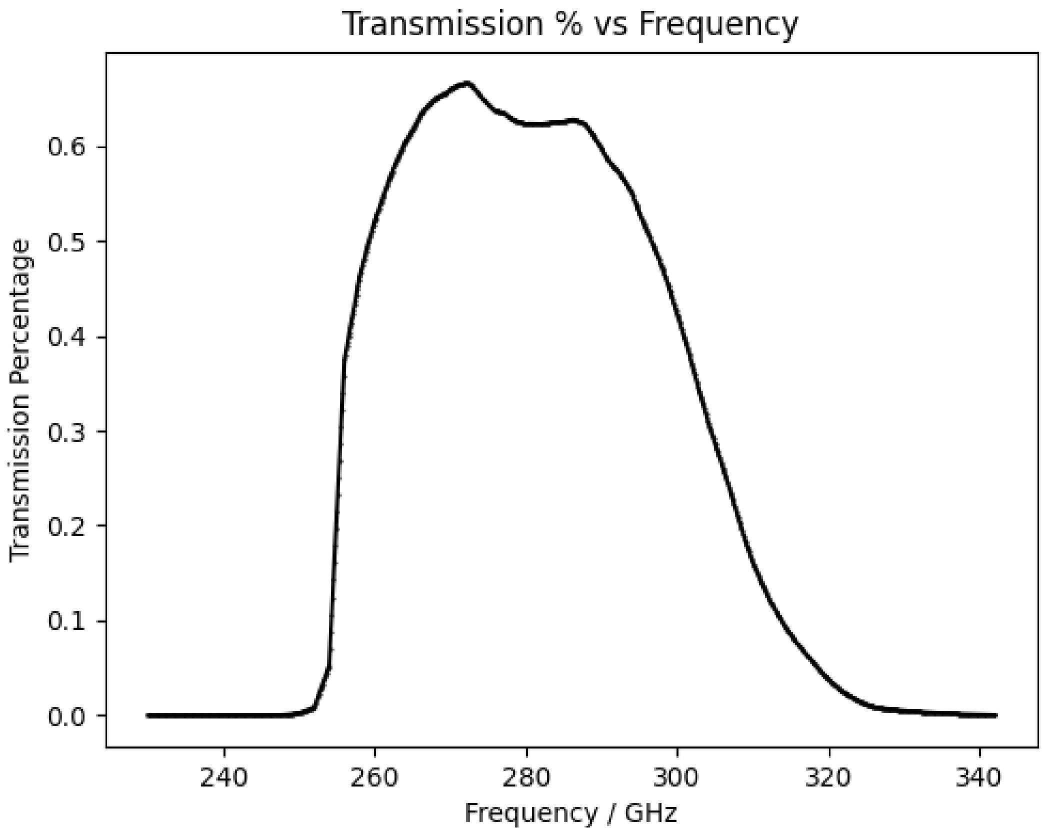


Beam Filling Factor was found to be: 0.9847

Week 5 - Week commencing 7/3/2022

Week Outline

Online meeting discussing the next steps of the project, calculating a power emitted by the background as a blackbody. Starting from the filter profiles of the detector, the filter profiles essentially acts as a “transmission” factor for a range of frequency ranges. These filter profiles were combined and given as data, plotted as shown:



This gives the overall power that is filterered away and can be multiplied with the power to give the received power by the detector.

The next step is to calculate the integral for the blackbody function over the filter bandwidth. We can model the background as a blackbody at temperature T measured in week 1 and by integrating over the bandwidth, gives us the power/solid angle.

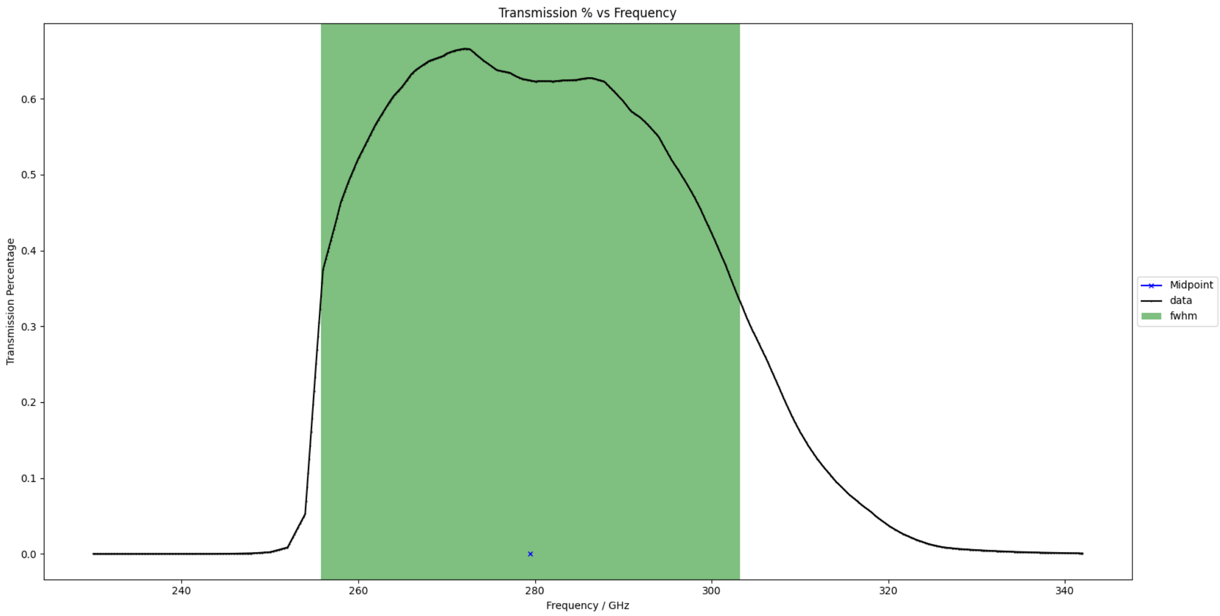
Finally, we can multiply this power/solid angle with λ^2 as the antenna throughput and obtain the power received by the detector from the room.

Complete Python Code Attached at the End of Diary

Tasks Outline

- Workout the Full-width-half-maximum of the filter profile and take the midpoint of it as the frequency v
- Workout dv as the final frequency point – first frequency point of the filter profile
- Create a Planck Function, use the temperature T of the room and frequency range as the inputs and integrate over the Planck Function over the first and last value of the frequency of the filter profile.
- Use this value and multiply by the transmission profile to get the filtered power/solid angle
- Finally, use this value and multiply by the throughput of the "antenna" λ^2 which is the midpoint frequency v converted to wavelength to get power received by the detector.

Transmission Profile with FWHM and Midpoint



Green bar = FWHM

Blue cross = midpoint

Results

Power received by the detector emitted by room at 293K over the filter profile bandwidth:

$$P = 2.430 \times 10^{-10}\, W$$

Or

$$P = 243\, pW$$

Distinction between the NEP and NET

Following from previously, we have calculated a NET based on detector data and the response. The NET gives a good description for the system as a whole, but the NEP gives a better description for th detector as NET does not consider the optical bandwidth but the NEP does.

A good way to think about it is to imagine a thermal camera. A thermal camera typically has a large bandwidth. Thus, this allows it to have a higher sensitivity to temperature gradients, e.g. low NET. However, having such a large bandwidth essentially “masks” the detector, as larger bandwidths means allowing for more noise to saturate the data, thus increasing the NEP. The NEP gives a better description of the fundamental limit of the detector when compared to the photon noise.

Week 6 - Week commencing 7/3/2022

Week Outline

Online meeting to cover error fixing in the Python code and to discuss the next step of the project. After obtaining the power received by the detector, we can now calculate the photon noise as a quadrature sum of the wave noise and shot noise. The formulas for calculating them are as follows:

$$shot\ noise = \sqrt{2P_{opt}hv}$$

Where v is the frequency, found from the midpoint of the bandwidth and P_{opt} is the optical power calculated from last week.

$$wave\ noise = \frac{P_{opt}}{\sqrt{2dv}}$$

Where dv is the bandwidth.

$$Total\ photon\ noise = \sqrt{WaveNoise^2 + ShotNoise^2}$$

Following this, we can also calculate the power emitted by the hot bar with room by using the same procedure as for the room power with the addition of the Beam Filling factor:

$$P = (BeamFillingFactor)\lambda^2 \int B(v)\, Transmission\ dv$$

Then, the response of the hot bar in terms of power:

$$Response = \frac{dF0}{P_{hotbar}}$$

Then, finally the NEP can be found:

$$NEP = \frac{\sqrt{e_n}}{Response}$$

Tasks Outline

- Calculate the shot noise and wave noise using the optical power P_{opt} obtained last week
- Use the shot and wave noise to calculate the total photon noise
- Find the power of the hot bar using the equation given previously
- Use this power to find the response of the detector in terms of the power
- Calculate the NEP

Results

The results was calculated using the given equations to find the NEP. KID 2 was used as it has the clearest data:

- Total Photon Noise:

$$Total\ Photon\ Noise = 8.4367 \times 10^{-16} \frac{W}{Hz^{0.5}}$$

- Modelled P:

$$P = 2.6105 \times 10^{-10}\ W$$

- Room power P_{room}:

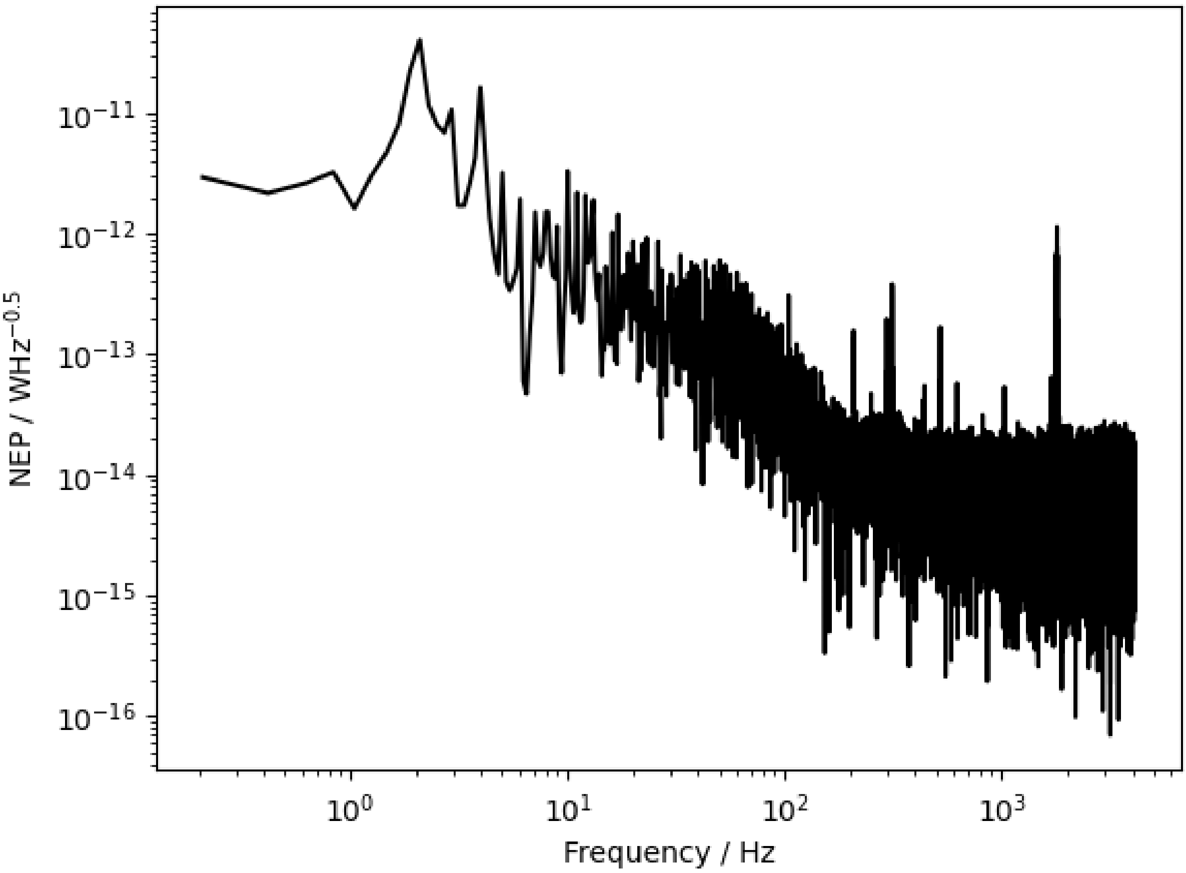
$$P_{room} = 2.4527 \times 10^{-10}\ W$$

- Hot bar power dp:

$$dP = P - P_{room} = 1.9800 \times 10^{-11}\ W$$

- NEP:

The NEP can now be calculated from the previous equations, dp and the noise spectral density. The NEP for KID 2 is given:



Week 7 - Week commencing 14/3/2022

Week Outline

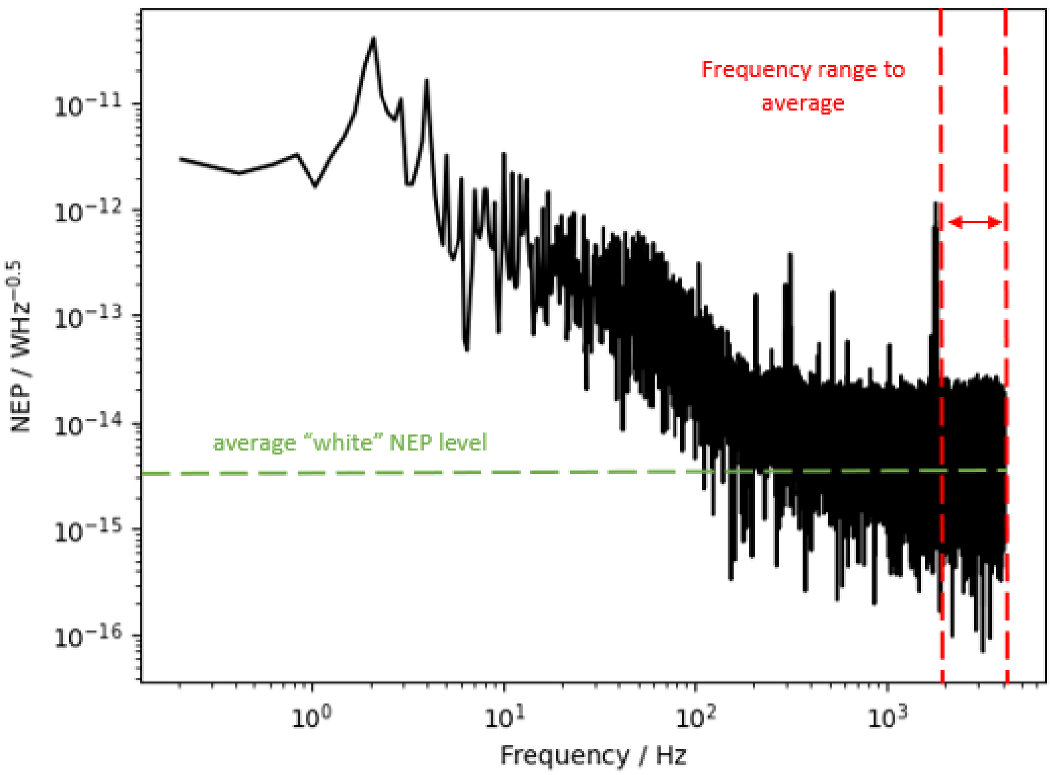
Online meeting discussing the next steps of the project. To recap, the aims of this project is to look at the SFAB detector array and characterize it in terms of its detector properties. The detector properties that we have calculated so far is the response, noise spectral density and the NEP. However, we want to do this for the whole array. This week focuses on obtaining a distribution of NEPs for the whole array

Task Outline

- Find the responsivity of the whole KID array
- Use the responsivity to find the NEP
- Find average "white" NEP for all KIDs in the array
- Plot a histogram of the NEPs of the array

Methodology

So, this week focuses on taking the previously found NEP and using it to find the closest approximation of the photon noise limited NEP to compare with the photon noise limit. This is done by taking the average of the NEP over a higher frequency range:



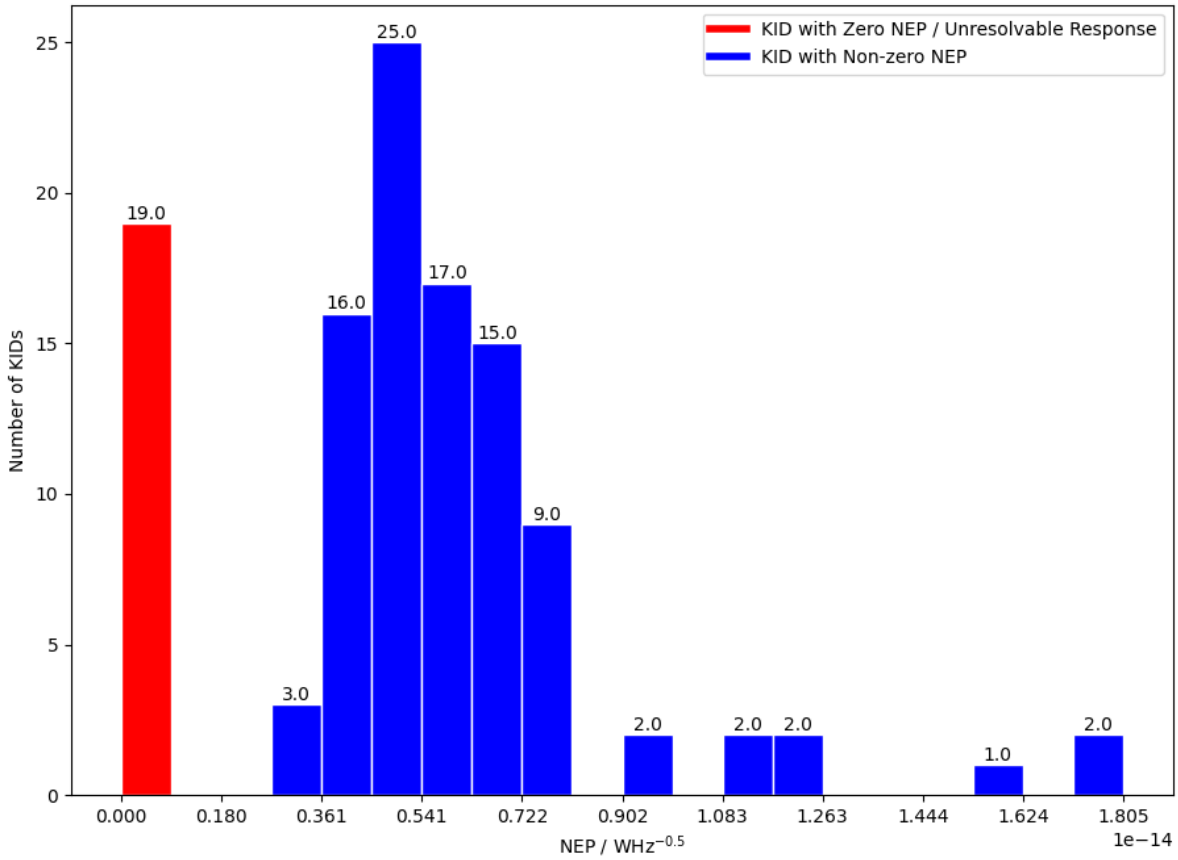
This figure is of KID 2, where the green is the average NEP level with a "white" feature and the red is the frequency range we are averaging over. The reason we do this is because the photon noise limit has a “white” feature, meaning it is a constant spectral density over all frequencies. This averaging allows us to find that same feature. This gave an average NEP of:

$$NEP_{ave} = 2.50 \times 10^{-15}\ WHz^{-0.5}$$

This was for KID 2. The responsivity of all KIDs can be found and the same procedures for finding the average NEP can be done. Once all the NEPs are found, a histogram can be plotted of the distribution of NEPs of the array.

PYTHON CODE AT END OF DIARY

Histogram Plot



The histogram plot was created. The red bars denote the KIDs that had responsivities that could not be found. Essentially, their hot bar peaks of dF0 was masked by the excess noise of the system. This could be due to interference, but there is no concrete evidence of this.

Week 8 - Week commencing 21/3/2022

Week Outline

Online meeting. Meeting involved checking over calculations and code from last week as it was not quite right. Following this, this meeting was focused on wrapping up the project and discussing why the NEP was much lower than the photon noise limit. This was mainly attributed to the underestimate of the response. The other detector property was also determined, which was the detector yield. Otherwise, this was mostly a explanation session for why the detector behaves the way it does.

Outline of Tasks

- Calculate the detector yield for the SFAB arrays of total KIDS = 176
- Compare the NEP to the photon noise limit

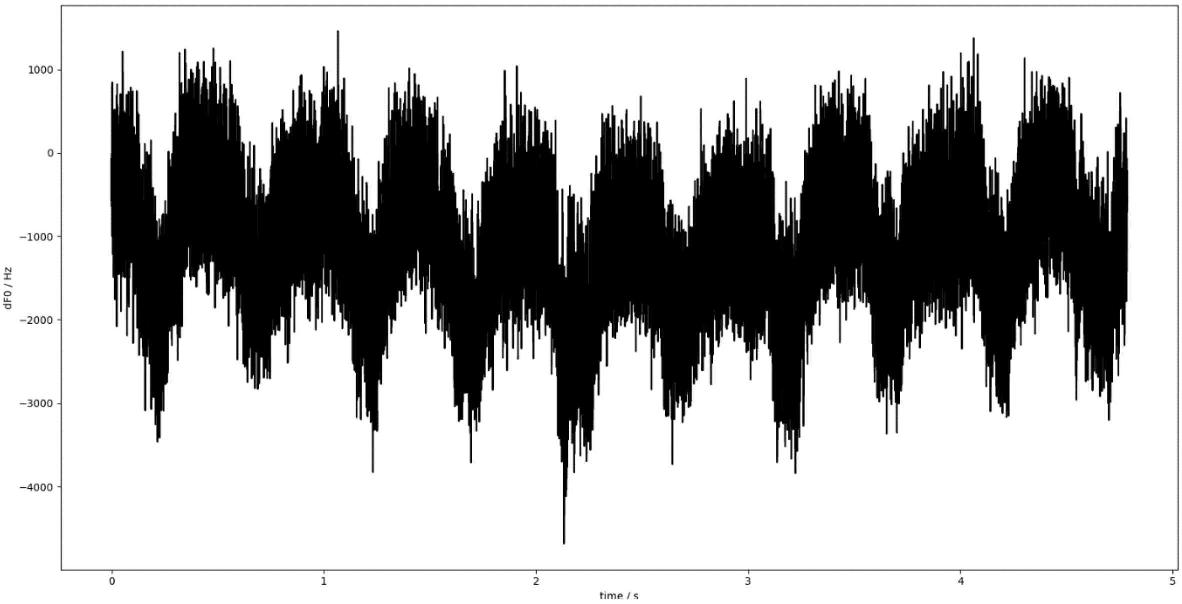
Detector Yield

The detector yield is given as follows:

$$Yield = \frac{n_{responsive}}{n_{KID}}$$

Where $n_{responsive}$ is the number of responsive KIDs in the arrays that gave a hot bar response. n_{KID} is the total number of KIDs in the array. For the SFAB system this was 176. From the NEP histogram from last week, it was found that there were $n_{responsive} = 94$. So, this led to a detector yield of 53.4%. Out of the 176 KIDs, only 113 KIDs gave data outputs. The remainder of KIDs did not give I and Q values, which could be due to a wide range of problems, such as damage, defects, excess temperature fluctuations, etc. It can't really be determined at this point in time, and it is also too advanced in terms of the report.

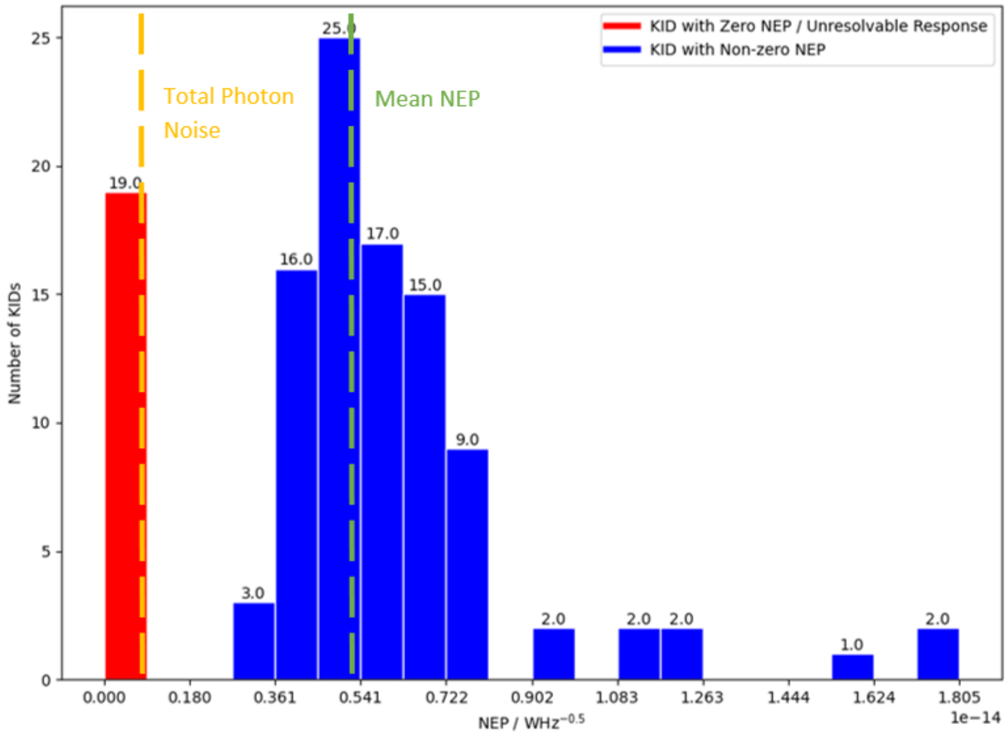
For the other detectors, they gave an unresolvable response, as shown by the red bars in the Histogram. These KIDs had dF0 data that could not have the hot bar peaks be identified. An example is shown:



The dF0 timestream graph above shows that there are no hot bar peaks to be found, or easily identifiable. Most likely, the hot bar peaks are there, but the peaks are masked by excess noise from/on the detector. This could be due to a whole range of factors such as temperature fluctuations of the KID, interference, background noise, etc. As such, these KIDs were given a "Unresolvable Response" tag on the Histogram.

NEP Comparison with the Photon Noise Limit

The NEP can be calculated as a mean from the histogram. This is illustrated:



The green line was taken as the mean NEP and this was used to compare with the total photon noise limit calculated last week.

The ratio of mean NEP of the KID array to the total photon noise limit was found to be 5.88. This means that the KID array NEP was 5.88 times larger than the total photon noise limit. Ideal detectors should have NEPs that are photon noise limited. This means that perfect detectors have NEP equal to the photon noise limit. The reason for the KID to stray from the total photon noise limit is due to several factors, the first being an overestimate of the response. This is most likely due to underestimation of dF0. One major reason is due to the losses in the lenses. Since the lenses are made of propylene, which has refractive index of 1.5, the refractive index change causes significant losses through refraction and absorption as explained by my supervisor.

Another reason could be due to modelling the hot bar as a perfect blackbody. In reality, objects tend to radiate perfectly due to the surface materials that have emissivity lower than 1. This will decrease the dP calculated.

Final reason, could be due to excess noise causing NEP to increase. Since the NEP calculated is based on the noise spectral density, noise sources such as thermal noise, generation recombination of cooper pairs, interference and varying scenes will all contribute a noise level to the detector and increase the NEP.

Week 9 – Week commencing 28/3/2022

Week Outline

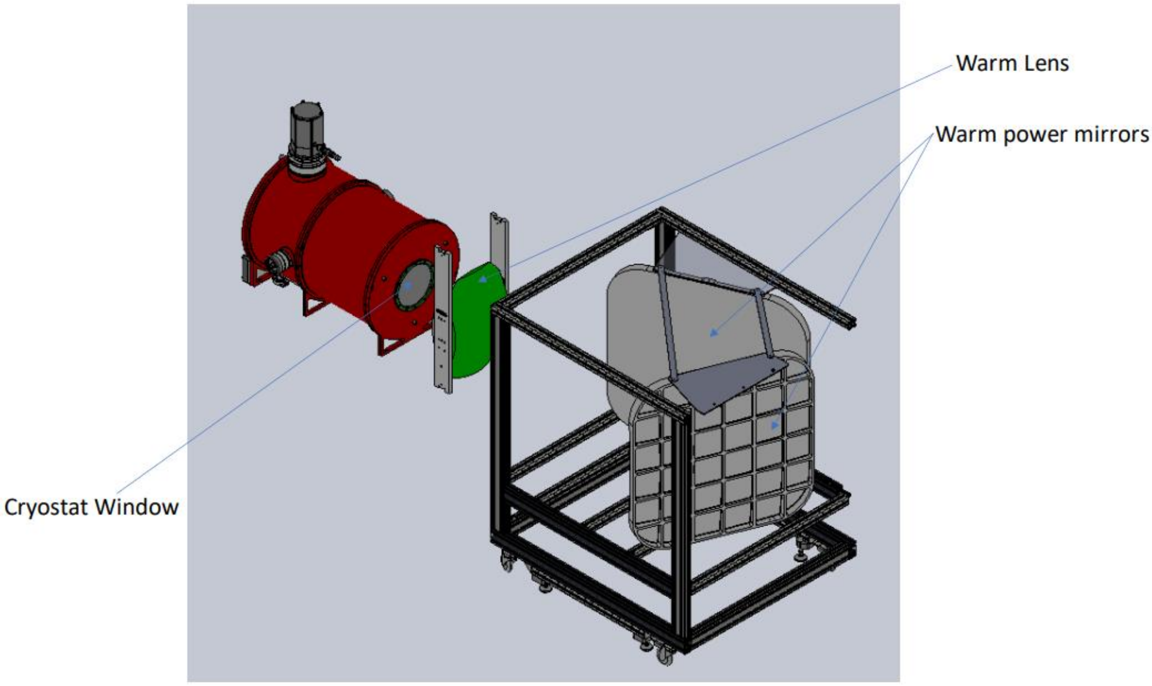
Online meeting. Meeting mainly for bug fixing the Python Code for the NEP calculations as values are slightly off. Next, giving an overview of the project and discussing the preparations for the presentations. Otherwise, there were no significant tasks in this weeks work other than to prepare for presentation with a plan and to start thinking about writing up the project. This was the week before Easter holidays, which would give ample time for preparing for the report writing and presentation.

Several schematics of the KIDs were also provided by Dr. Tom Brien that could be used in the final report or presentation.

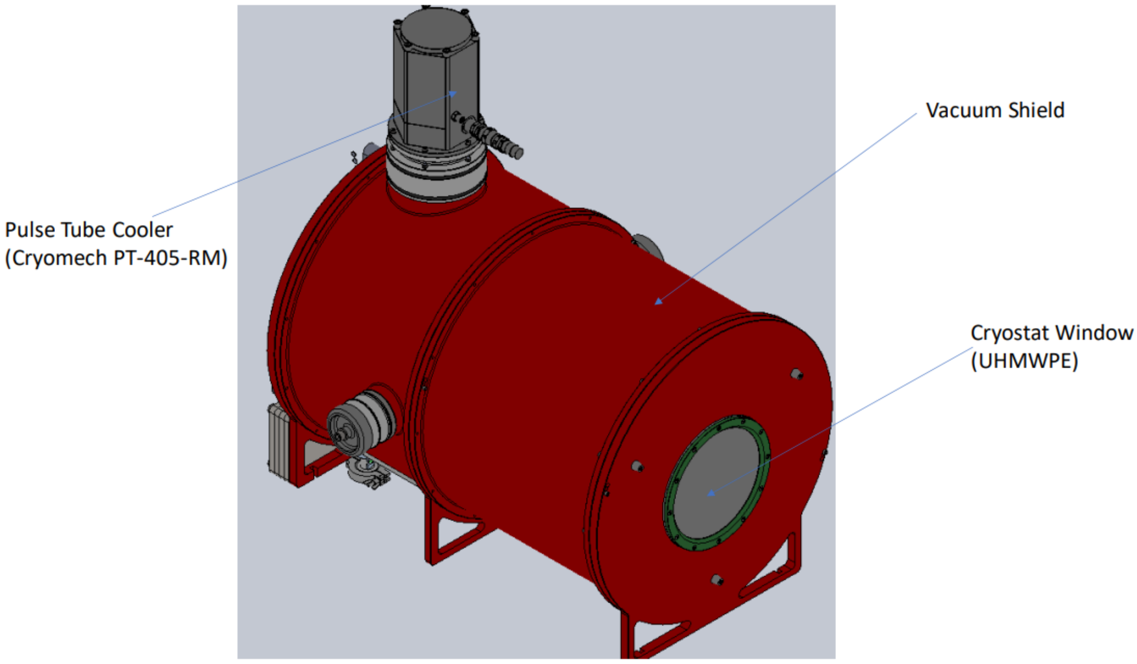
Outline of Tasks

- Create a plan/draft for the presentation
- Prepare for report write up

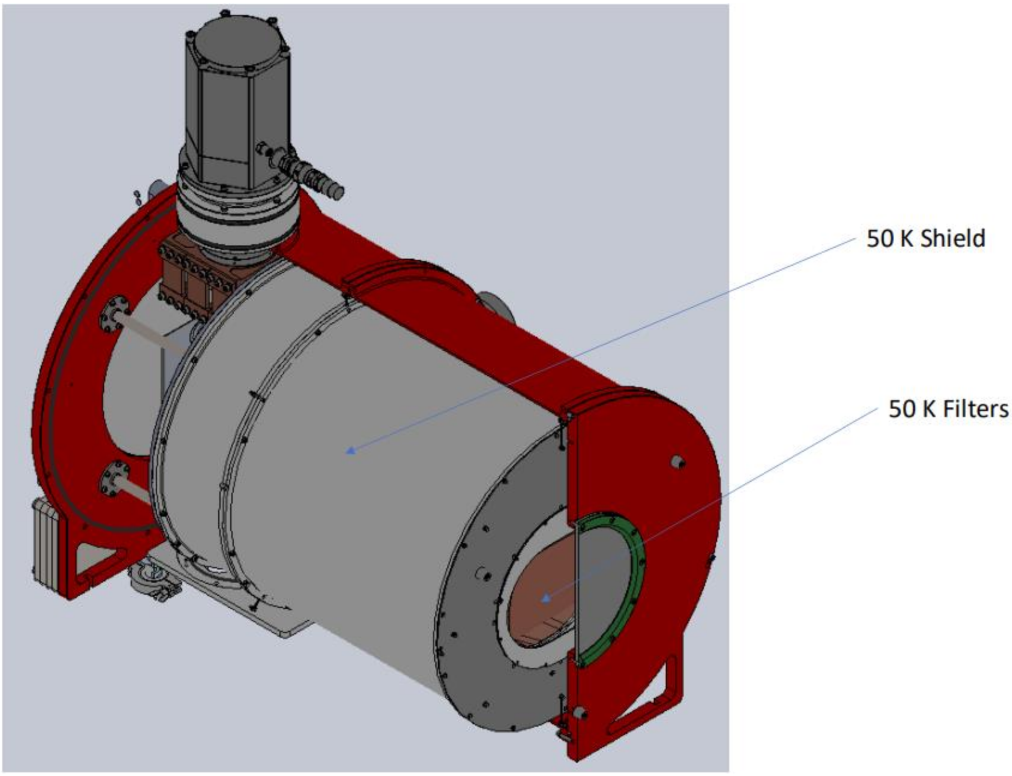
Schematics of KID



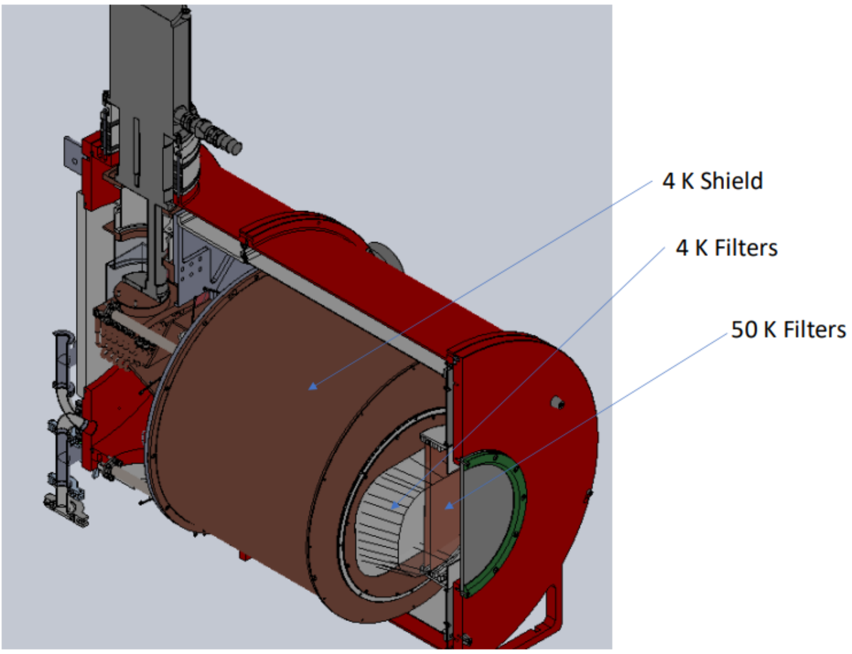
Schematic of the optics system set up for the SFAB system



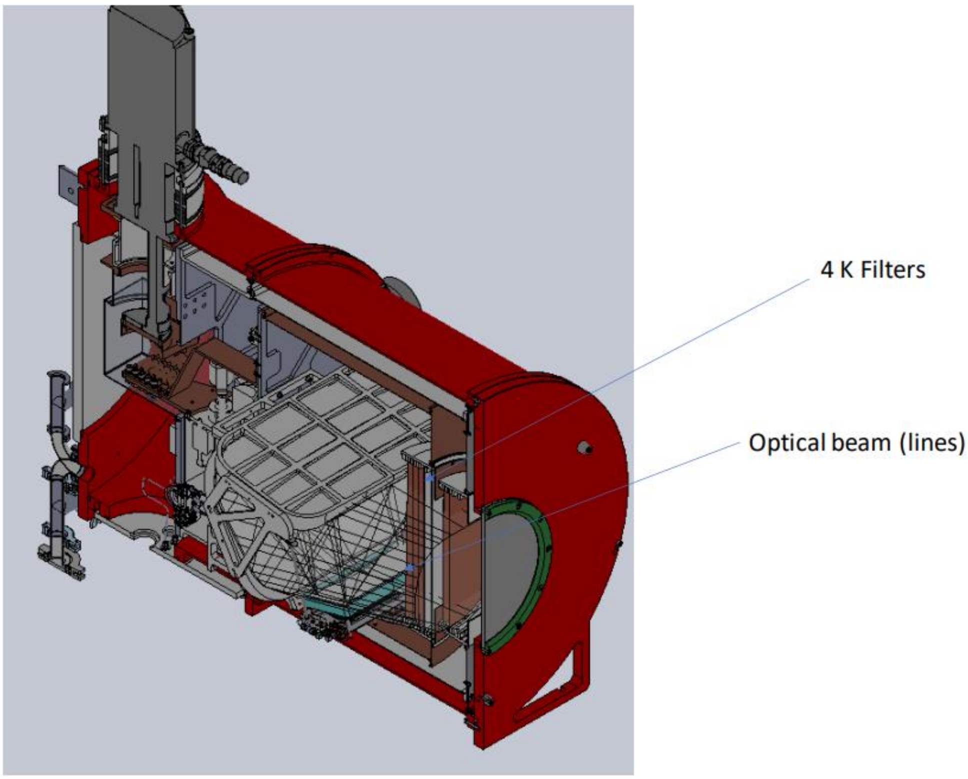
Labelled schematic of the camera



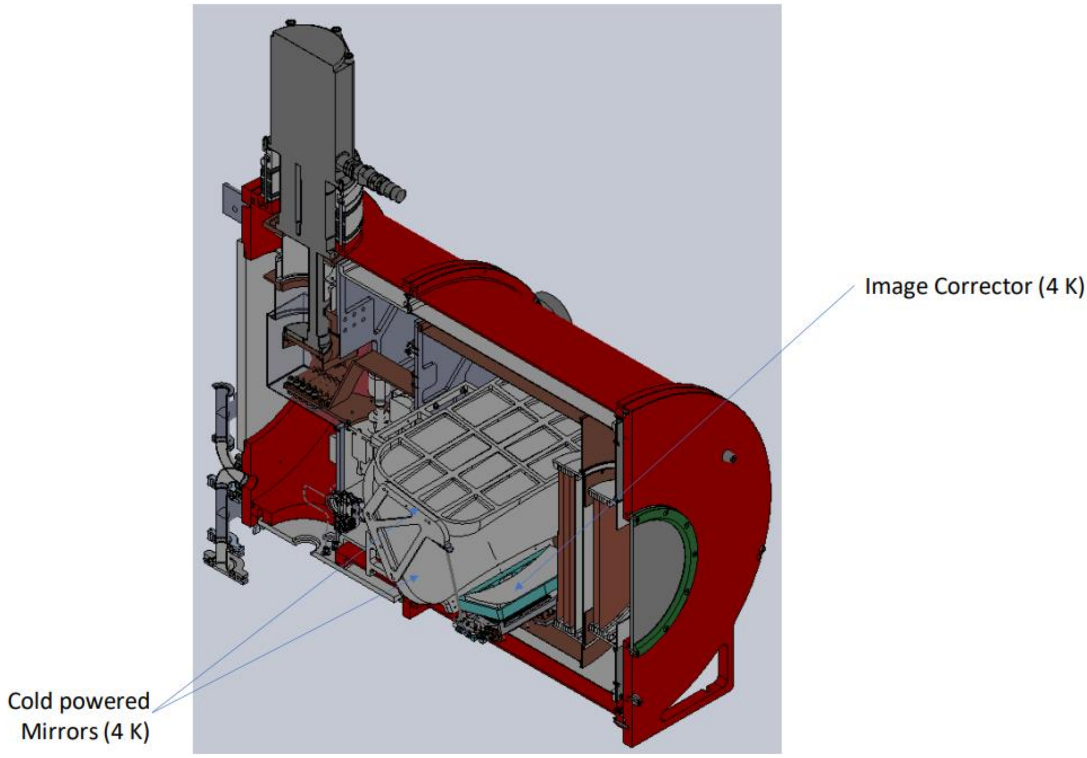
Labelled schematic of inner layer of camera



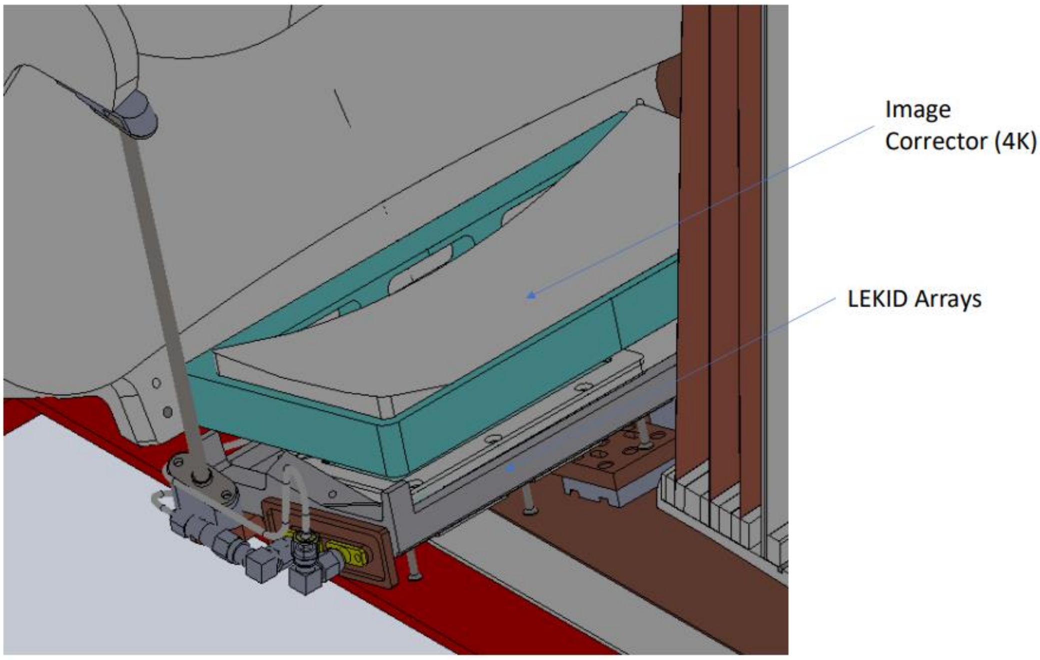
Schematic with filters and shield labelled within the camera



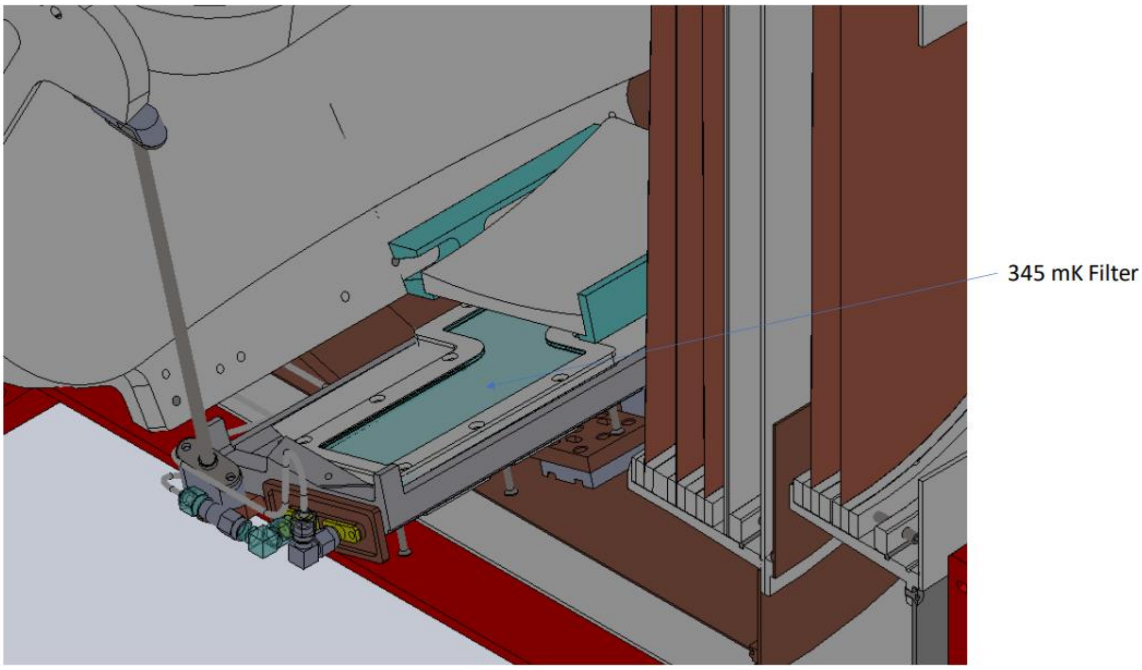
Schematic with filters and beam lines labelled within the camera



Schematic of interior of the camera



Schematic of the KID array



Schematic of the KID array

Week 10 – Week commencing 25/4/2022

Week Outline

Online meeting. This week focused on feedback from both my PowerPoint slides and my approach to presenting my PowerPoint. During the meeting, I went through the slides and my supervisor gave feedback on the presentation. This was the main theme of the meeting for this week.

Outline of Tasks

- Amend the presentation based on the recommendations of my supervisor
- Prepare for presentation and questions
- Create a plan for report

Recommendations

The following are recommendations given by my supervisor on the presentation slides:

- Add more detail on what you have done
- Fix few errors on terminology
- Fix wavelength ranges for detectors
- Add plots to aid explanations
- Explain more detail on superconductivity and how it works for a KID

Presentation

Before the meeting I created a PowerPoint for a presentation where I started with introducing the background of KIDs and its relevance, and then moving on to discussing detector properties, aims and objectives and findings. The slides were amended based on the recommendations and are attached below:

CHARACTERIZING ARRAYS OF KINETIC INDUCTANCE DETECTORS

ASHLEY THEAN

3RD YEAR BSC PROJECT PRESENTATION – 29TH APRIL 2022

WHAT IS THE MOTIVATION FOR DEVELOPMENT?

Challenges faced in longer wavelength detection:

1) **Limited pixel count**:

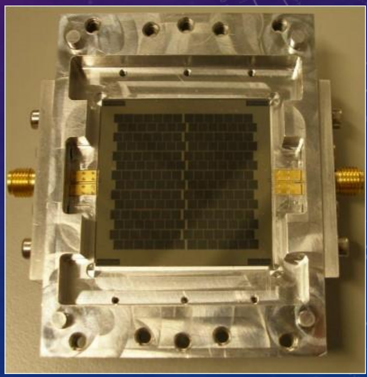
- Semiconductor tech (e.g. mobile cameras): Fit 10s of mega-pixels in a tiny square
- Best mm-wave cameras: Struggle to fit orders of thousands of pixels
- Detector physics tells us: $\boxed{Area\ of\ Detector \propto Wavelength^2}$
- Longer wavelength, larger detectors (Fewer pixels per area)

2) **Limited wavelength range**:

- Semiconductor physics: $\boxed{Photon\ Energy\ must\ be\ greater\ than\ band\ gap\ energy}$
- Lowest band gap energy semiconductor ~0.04eV (Wavelength limited ~30μm)

KINETIC INDUCTANCE DETECTORS (KID)

- Uses principles of superconducting-pair breaking
- Energy gap less than 1meV (wavelength up to 3mm)
- Developed for airport security cameras and in astronomy
- Compact: makes it easier for large arrays of detectors
- High sensitivity detector



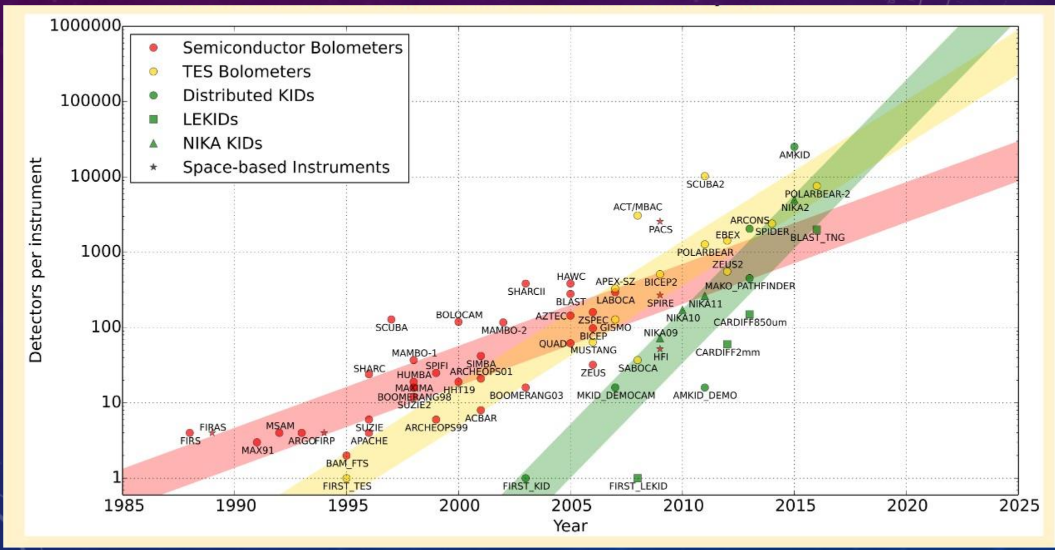
Source: Dr. Simon Doylés slides on intro to KIDs

COMPARISONS WITH COMMON DETECTORS WITHIN THE WAVELENGTH RANGE

Detector	Limitation	Comparing with KIDs
Semi-conductor technology	Limited to wavelengths of order $30\mu m$	Wavelengths up to 3mm
Heterodyne receivers	Typically noisy and not practical for large format imaging arrays	Compact, easy to create larger arrays
Bolometers	Have sensitivity but poor multiplexing ratios	Excellent multiplexing and sensitivity

Source: Dr. Simon Doyle's slides on introduction to Kinetic Inductance Detectors

PIXEL COUNTS IN ASTRONOMICAL RECEIVERS OVER THE YEARS



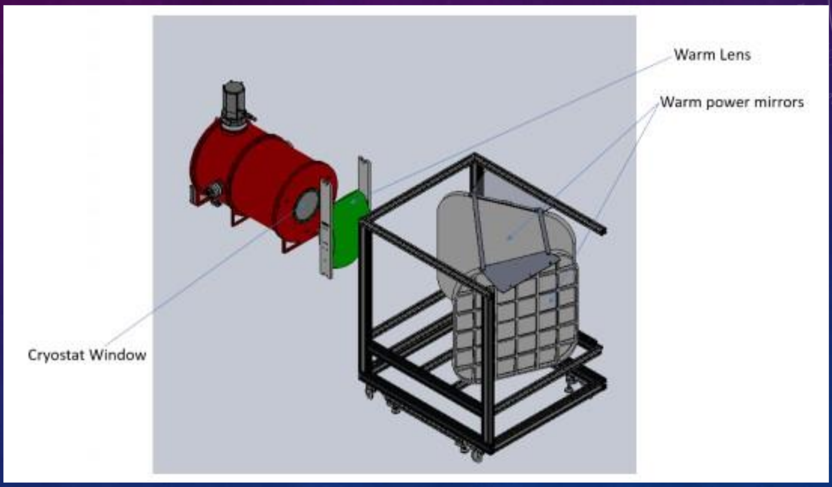
Source: Graph created by Dr. Sam Rowe

DETECTOR USED IN THIS PROJECT

- SFAB Security Imaging System
- Being developed in Cardiff University
- KID system developed for use as an airport security camera
- Range of frequencies used– transparent to bodies and clothing but not metals
- Detector kept at ~280mK

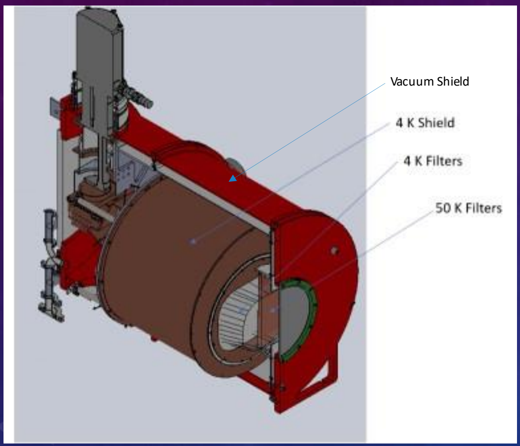
Source: Dr. Simon Doyle's slides on intro to KIDs

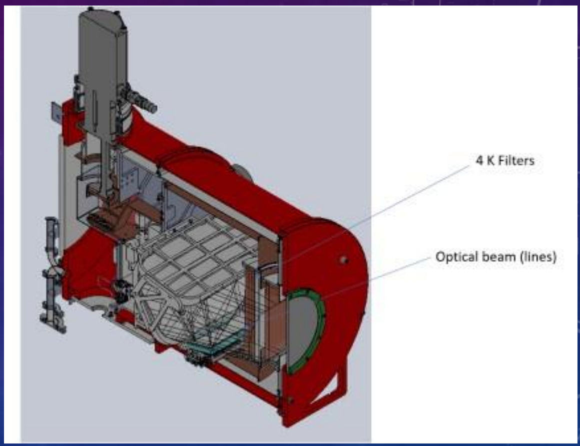
SCHEMATIC OF THE SFAB SECURITY IMAGING SYSTEM SET UP



Source: System schematic by Dr. Tom Brien

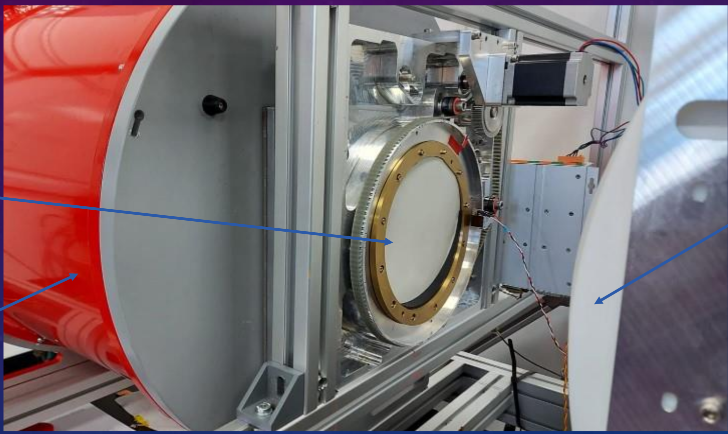
SCHEMATIC OF THE CAMERA INTERIOR





Source: Schematics created by Dr. Tom Brien

PHOTOGRAPH OF THE SFAB SECURITY IMAGING SYSTEM



AIMS AND OBJECTIVES

1. Explore and understand the basic **principles of superconductivity** and its **application to KIDs**
2. Create a simple simulation of an ideal KID for a change in temperature
3. Read and analyze real KID data
4. Use experimental data to characterize the noise and detector properties

CHARACTERIZING DETECTOR PROPERTIES

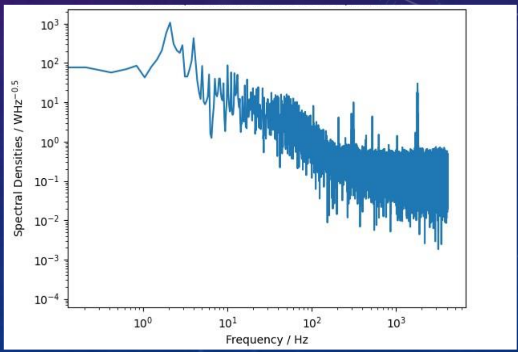
- Many performance metrics to compare between detectors
- In this report we focused on 4:

1. Responsivity:
Quantifying how the detector responds to a signal

2. Noise-Equivalent-Power (NEP):
Measure of sensitivity of a detector. Minimum detectable power

3. Detector System Yield:
Characterize losses throughout the system

4. Noise Spectrum:
Spectral density of the noise across the frequency



HOW DOES A KID WORK?

Superconductivity principles:

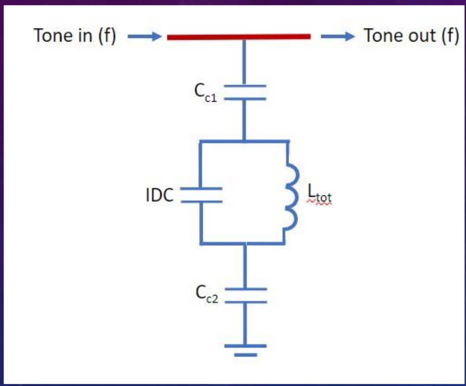
- At temperatures below T_c (depends on material), electrons start to pair up into superconducting electron -pairs
- When number of pairs varies, inductance will vary

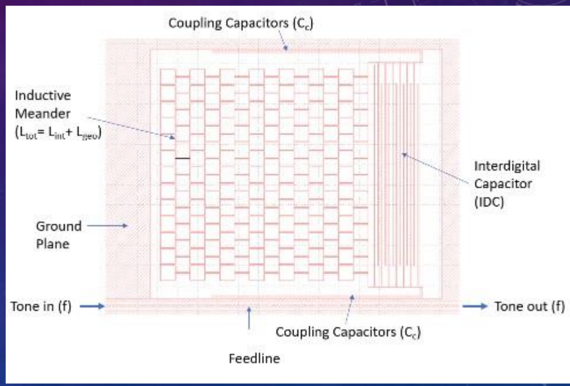
KID circuit:

- KID based on LC circuit
- LC circuit produces resonance when a signal (tone) is placed
- Inductor - superconductor and photon absorber
- Incident photons causes inductance to vary (energy absorbed to break pairs)

Able to detect photons from changes in resonance curve features!

CIRCUIT DIAGRAM OF KID

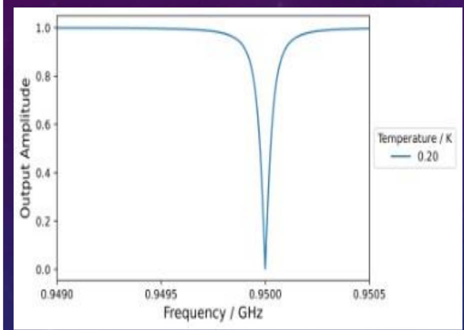




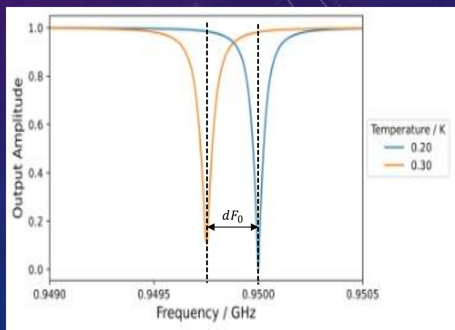
Source: Circuit diagram created by Dr. Simon Doyle

SIMULATION FOR CHANGE IN TEMPERATURE

Change in temperature has same effect as incident photons (energy break electron pairs)



Change in Temperature

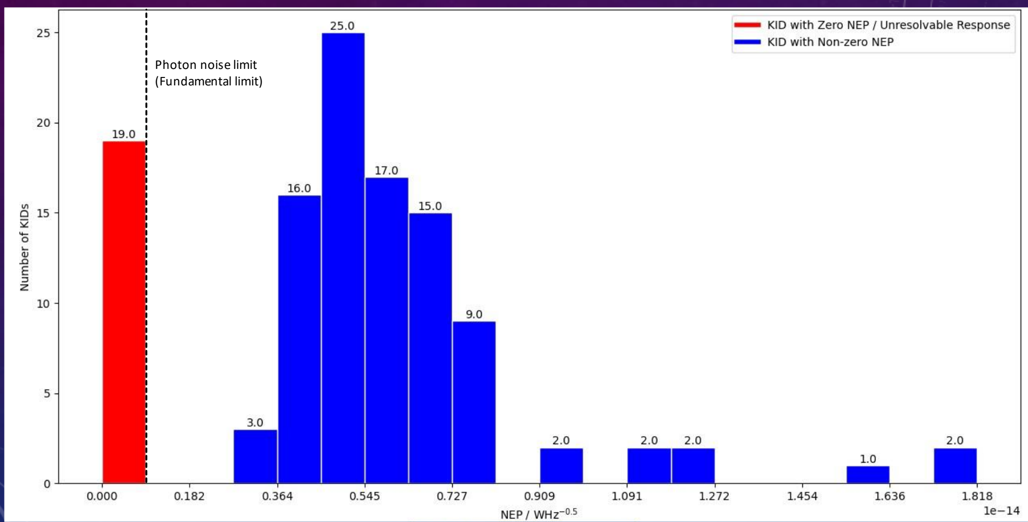


- Modelled using superconductivity theory
- Change in temperature causes resonance curve to shift position by dF_0

ANALYSIS OVERVIEW

1. Set up detector to measure a hot bar of known temperature
2. Scanned the hot bar to get detector response (hot bar power modelled as blackbody emission)
3. Compared this response to detector noise to determine minimum signal detectable (NEP)
4. Plot a histogram of sensitivity of all the KIDs in the array and compared it to the fundamental noise limit

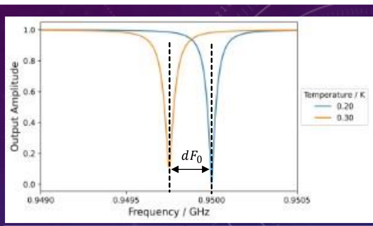
NEP DISTRIBUTION OF AN ARRAY OF KID

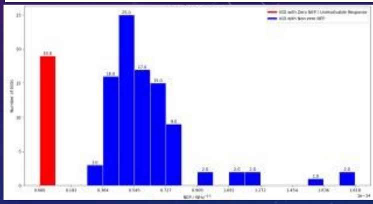


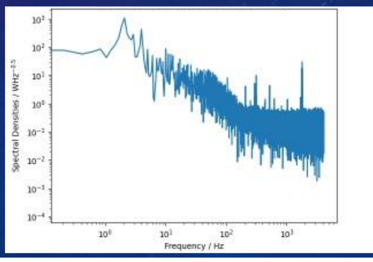
Total KIDs = 113

SUMMARY OF REPORT

1. Understood superconductivity and KIDs
2. Modelled the KID for a change in temperature
3. Took measurements of a hot bar of known temperature
4. Obtained analysis of the response, Noise Spectrum, NEP distribution
5. Found that NEP makes sense and only slightly above the fundamental limit. Relatively sensitive
6. Noise spectrum has all the right characteristics from the expected noise sources







THANK YOU FOR LISTENING!

Week 11 – Week commencing 2/5/2022

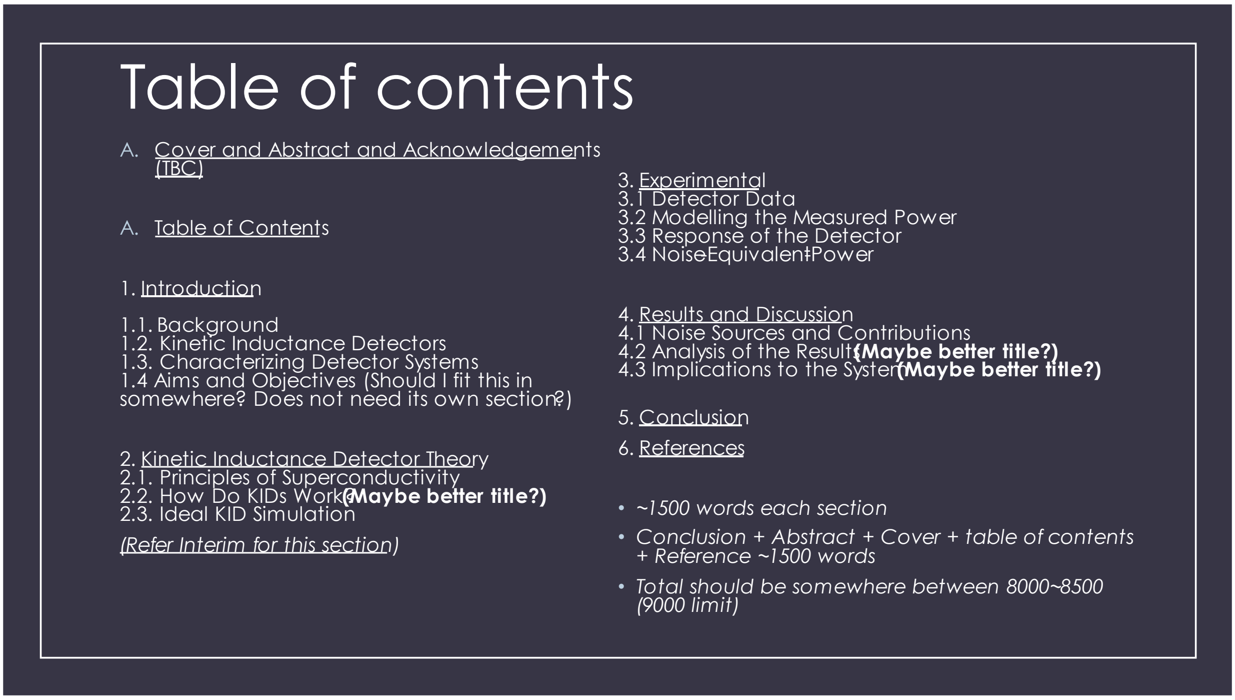
Week Outline

Online meeting. This meeting was mainly used to discuss the final report plan and what topics needed to be covered and what need to be removed. Advised to rephrase more of the superconductivity stuff from the interim report and summarize it into a smaller chunk. Talk more about the methodology and analysis of the project. Several diagrams was also included by my supervisor to include in my report.

Task Outline

- Using the report plan, create a final report
- Use the recommendations in the final report

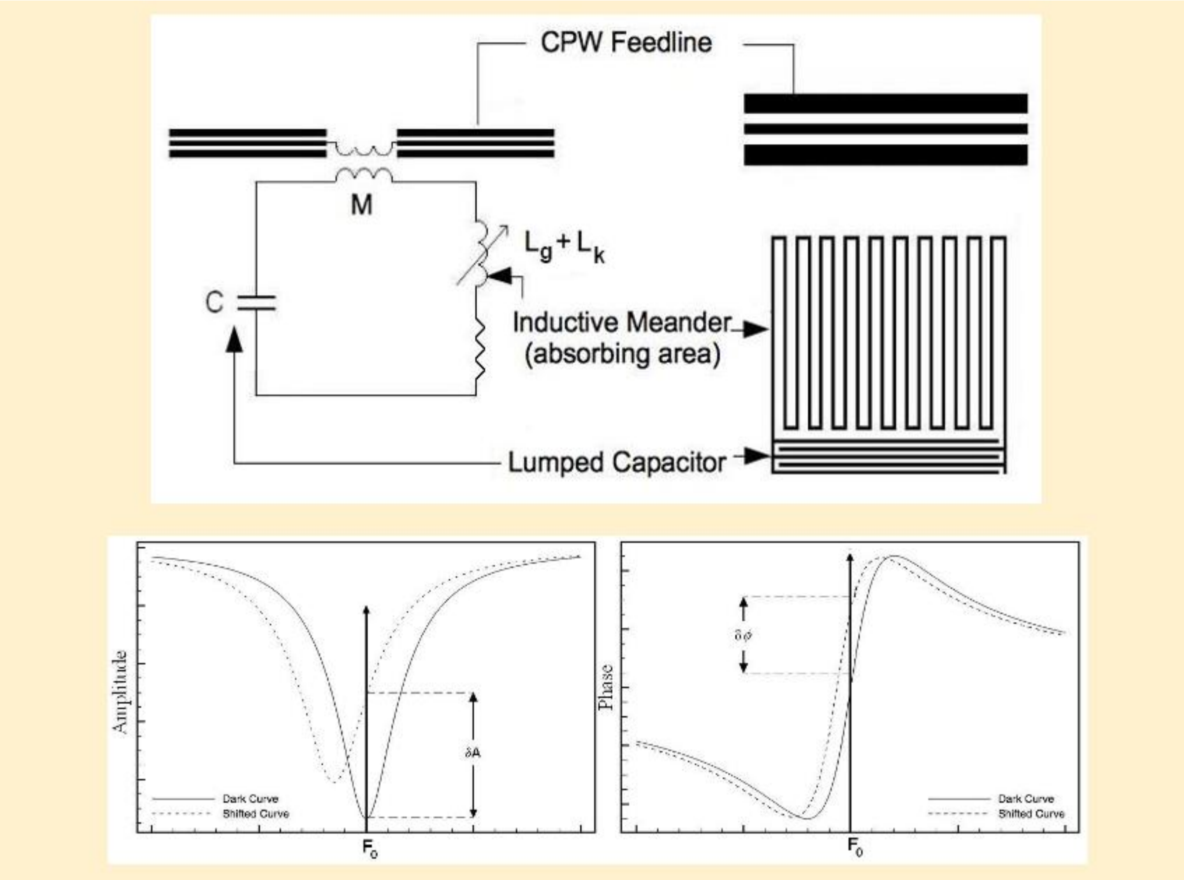
Planned table of contents



Recommendations

- Make sure under word limit by Turnitin
- Intro: describe photon noise, background, and KIDs overview
- Condense the superconductivity theory into less words
- Use lots of graphs and diagrams to aid explanations (words count, but diagram = 0 words)
- Touch upon more experimental stuff
- KID simulation and dF0 formula analysis is good to have in
- Explain photon noise limit and what implications if NEP higher? (excess noise, etc. Link to ERD)
- Noise sources? (explain?)

__**Diagram of KID response given by Supervisor**__



This diagram can be used to explain how a KID responds to light in the report

Week 12 – Week commencing 9/5/2022

Week Outline

Online meeting. Since final report due date is the 13th May, this meeting focuses on discussing the draft final report and corrections. Main problems of the report included:

- Over word limit
- Terms and symbols that are introduced not defined
- Not succinct

Recommendations:

- Removing sections that are not as consequential to lower word count (e.g. interference, excess noise dF0 data that masks hot bar response)
- More clarification on all introduced concepts and symbols
- Add diagram for airport security camera image, showing function
- Shorten language in exchange for succinctness. (e.g. As seen previously in Figure 3… -> Figure 3 shows…)

Outline of Tasks

- Correct the final report based on recommendations and advice given above.

<u>Complete Python Code For Whole Project</u>

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.special import iv as I0
from scipy.special import kv as K0
import fsab_dirfile_raw as fsab
import os
from scipy import signal
from scipy import constants as const
from scipy.interpolate import UnivariateSpline
import csv as csv
from matplotlib.lines import Line2D
from matplotlib import cm


#Constant Experiment variables
T_Hotbar = 41.3
T_Surrounding = 22.7
dT = T_Hotbar - T_Surrounding
T_Hotbar_K = T_Hotbar + 273
T_Surrounding_K = T_Surrounding + 273

#Hard Code User Inputs
data_folder_name = "ch1"
KID_Number = 2
lower_range = 1.5725
upper_range = 1.5875

def main():
    global KID_Number
    global lower_range
    global upper_range

    #Reading data
    local_file = os.getcwd()
    data = fsab.fsab_dirfile(local_file + "\\"  + data_folder_name)

    #Plotting Sweep
    user_input = input("Default analysis (y/n)? (KID = 2, 1.5725 < hotbar <
1.5875)\n")

    #Get vals
    if user_input.lower() == "n":
        KID_Number, upper_range, lower_range = get_user_input(data)
```

```
    I_Time, Q_Time, time = Get_IQ_Time(data, KID_Number)

    plt.plot(time, np.sqrt(I_Time**2+Q_Time**2), color='black')
    plt.ylabel("|S21| / V")
    plt.xlabel("time / s")
    plt.show()
    plt.figure()

    I, Q, sweep, tone_freq = Get_IQ_Sweep(data, KID_Number)
    print(tone_freq)
    plt.plot(sweep/1e9, np.sqrt(I**2+Q**2), color='black')
    plt.ylabel("|S21| / V")
    plt.xlabel("Frequency / GHz")
    plt.show()
    plt.figure()
    dF0, time = Get_dF0(data, KID_Number)


    plt.plot(time, dF0, color='black')
    plt.ylabel("dF0 / Hz")
    plt.xlabel("time / s")
    plt.ticklabel_format(useOffset=False)
    plt.show()
    plt.figure()


    #Get range of time for peak height
    peak_range_array = np.where(np.logical_and(time>=lower_range,
time<=upper_range))

    #Get index of upper and lower range for time
    lower_index = peak_range_array[0][0]
    upper_index = peak_range_array[0][-1]

    plt.plot(time[lower_index-40:upper_index+40], dF0[lower_index-
40:upper_index+40], marker="x", markersize=2, color="black", linewidth=0)
    plt.xlabel('Time / s')
    plt.ylabel('dF0 / Hz')
    plt.show()
    plt.figure()

    #Plot this range
    plt.plot(time[lower_index-40:upper_index+40], dF0[lower_index-
40:upper_index+40], marker="x", markersize=2, color="black", linewidth=0,
label="Data")

    #Curve fit
    #P_Guess
    height = 400
```

```
    mu = time[upper_index] - time[lower_index]
    sigma = 1
    c = 3000
    p_guess = [height, mu, sigma, c]

    #Gaussian curve
    popt_main, _ = curve_fit(Gaussian, time[lower_index:upper_index],
dF0[lower_index:upper_index], maxfev=20000, p0=p_guess)
    response_main = popt_main[0]/dT

    dF0_hotbar = popt_main[0]

    #Plot curve_fit
    plt.plot(time[lower_index-30:upper_index+30], Gaussian(time[lower_index-
30:upper_index+30], *popt_main), label="Curve Fit")
    plt.xlabel('Time / s')
    plt.ylabel('dF0 / Hz')
    plt.legend(loc='upper right', fancybox=True)
    plt.show()

    print(*popt_main)

    responsivity = list()


    #Loop for all KIDs
    number_of_KID = data.numkids

    #P_Guess
    height = 400
    mu = time[upper_index] - time[lower_index]
    sigma = 1
    c = 3000
    p_guess = [height, mu, sigma, c]


    #popt=[]
    # for KID_Num in range(0, number_of_KID):
    #     dF0, time = Get_dF0(data, KID_Num)

    #     #Get index of upper and lower range for time
    #     lower_index = peak_range_array[0][0]
    #     upper_index = peak_range_array[0][-1]

    #     #Curve Fit
    #     try:
    #         popt, _ = curve_fit(Gaussian, time[lower_index:upper_index],
dF0[lower_index:upper_index], maxfev=10000, p0=p_guess)
```

```
    #         #Get curve heigh
    #         dF0_hotbar = popt[0]

    #         #Get response
    #         response = dF0_hotbar/dT

    #         if abs(response) <= 100:
    #             responsivity.append(response)
    #         else:
    #             responsivity.append(0)

    #     except RuntimeError:
    #         responsivity.append(0)

    # plt.plot(responsivity, marker='x', color='black')
    # plt.title("Responsivities of all KIDs")
    # plt.ylabel("Responsivity / Hz K^-1")
    # plt.xlabel("KID Number")
    # plt.ticklabel_format(useOffset=False)
    # plt.show()
    # plt.figure()

    #Spectral_densities
    frequencies, spectral_densities = fourier_transform(data, KID_Number)
    plt.plot(frequencies, np.sqrt(spectral_densities), color='black')
    plt.semilogy()
    plt.title("Noise Spectral Densities vs Frequencies")
    plt.ylabel("Spectral Densities / WHz$^{-0.5}$")
    plt.xlabel("Frequency / Hz")
    plt.show()
    plt.figure()



    #NET
    Noise_Eq_T = np.sqrt(spectral_densities)/response_main
    plt.loglog(frequencies[1:], (Noise_Eq_T[1:]), color="black")
    plt.title("NET vs frequencies for KID Number " + str(KID_Number))
    plt.ylabel("NET / KHz$^{-0.5}$")
    plt.xlabel("Frequency / Hz")
    plt.show()
    plt.figure()

    #Get Transmission Factor
    transmission_frequency, transmission = np.loadtxt('MUSCAT_band.txt',
unpack=True)

    #Get dnu using fwhm
```

```
    fwhm, midpoint_frequency, lower_bandwidth, upper_bandwidth =
transmission_fwhm(transmission_frequency, transmission)
    print("BANDWIDTH")
    print(upper_bandwidth)
    print(lower_bandwidth)
    dnu = upper_bandwidth-lower_bandwidth
    print(dnu)
    wavelength = const.c/midpoint_frequency
    print("fwhm")
    print(fwhm)
    print("midpoint")
    print(midpoint_frequency)
    #Blackbody Intensities

    room_power = get_power(planck, transmission_frequency, transmission,
wavelength, dnu, BFF=1, T=T_Surrounding_K)
    #Beam filling factor
    Beam_Filling_Factor = calculate_Beam_Filling_Factor(length=100, width=20,
FWHM=20, points=len(transmission_frequency))
    print(Beam_Filling_Factor)
    hotbar_room_power = get_power(planck, transmission_frequency,
transmission, wavelength, dnu, BFF=Beam_Filling_Factor, T=T_Hotbar_K)
    print("Room optical power = " + str(hotbar_room_power) + " W")

    print("Room optical power = " + str(room_power) + " W")

    total_photon_noise = np.sqrt(shot_noise(room_power, midpoint_frequency)**2
+ wave_noise(room_power, fwhm)**2)
    print(wave_noise(room_power, fwhm))
    print(fwhm)
    print("Total Photon Noise = " + str(total_photon_noise) + " W/Hz^0.5")

    #NEP
    dp = hotbar_room_power - room_power

    Response_Power = dF0_hotbar/dp
    print("Hotbar power = " + str(dp) + " W")

    print("response " + str(Response_Power))

    Noise_Eq_Power = np.sqrt(spectral_densities)/Response_Power
    plt.loglog(frequencies[1:], Noise_Eq_Power[1:], color="black")
    plt.ylabel("NEP / WHz$^{-0.5}$")
    plt.xlabel("Frequency / Hz")
    plt.title("NEP vs Frequency for KID " + str(KID_Number))
    plt.show()
    response_dict = {}
    with open('response.csv') as csv_file:
```

```
        csv_reader = csv.reader(csv_file, delimiter=',')
        line_count = 0
        for row in csv_reader:
            if line_count == 0:
                line_count += 1
            else:
                temp = row
                if len(temp) == 0:
                    k_NUMBER = int(line_count)
                    response = 0
                    response_dict[k_NUMBER] = response

                else:
                    k_NUMBER = int(temp[0])
                    response = float(temp[1])
                    response_dict[k_NUMBER] = response
                line_count += 1

    for i in range(1, data.numkids):

        plt.plot(i, response_dict[i], marker='x', color='black', markersize=5)

    plt.title("Responsivities of all KIDs")
    plt.ylabel("Responsivity / Hz W^-1")
    plt.xlabel("KID Number")
    plt.ticklabel_format(useOffset=False)
    plt.show()
    plt.figure()

    two_hundred_Hertz_Index = np.argmax(frequencies > 2000)

    for i in range(1, data.numkids):
        if response_dict[i] != 0:
            Noise_Eq_Power = np.sqrt(spectral_densities)/response_dict[i]
            plt.loglog(frequencies, Noise_Eq_Power, label="KID " + str(i))
        else:
            Noise_Eq_Power = np.zeros(len(frequencies))
            plt.loglog(frequencies, Noise_Eq_Power, label="KID " + str(i))

    plt.title("NEP vs frequency")
    plt.ylabel("NEP")
    plt.xlabel("Frequency / Hz")
    plt.show()
    print("last = " + str(frequencies[-1]))

    all_kid_NEP = []

    for i in range(1, data.numkids):
```

```
        if response_dict[i] != 0:
            Noise_Eq_Power = np.sqrt(spectral_densities)/response_dict[i]
            ave_NEP = np.average(Noise_Eq_Power[two_hundred_Hertz_Index:])
            all_kid_NEP.append(ave_NEP)
        else:
            all_kid_NEP.append(0)

    fig, ax = plt.subplots()
    N, bins, patches = ax.hist(all_kid_NEP, bins=20 ,edgecolor='white',
linewidth=1)
    patches[0].set_facecolor('r')
    for i in range(1, len(patches)):
        patches[i].set_facecolor('b')

    bins_size = bins[1] - bins[0]
    plt.figtext(0.5, 0.01, f"Bin Size = ${bins_size:.2E}"  , ha="center",
fontsize=18, bbox={"facecolor":"orange", "alpha":0.5, "pad":5})
    plt.title("Histogram of KID NEP")
    plt.ylabel("Number of KIDs")
    plt.xlabel("NEP / WHz$^{-0.5}$")

    labels = N

    custom_lines = [Line2D([0], [0], color="r", lw=4),
                    Line2D([0], [0], color='b', lw=4)]

    # Make some labels.
    rects = ax.patches

    for rect, label in zip(rects, labels):
        height = rect.get_height()
        if label != 0:
            ax.text(rect.get_x() + rect.get_width() / 2, height+0.01, label,
                    ha='center', va='bottom')

    ax.legend(custom_lines, ["KID with Zero NEP / Unresolvable Response", "KID
with Non-zero NEP"])
    plt.xticks(bins[::2])


    # height = 25
    # mu = 0.5e-14
    # sigma = 1
    # c = 0
    # p_guess = [height, mu, sigma, c]

    print(N)
    # #Gaussian curve
```

```python
    popt_main, _ = curve_fit(Gaussian, bins[1:10], N[1:10], maxfev=20000)
    x = np.linspace(bins[1], bins[10], num=200)
    y = Gaussian(x, *popt_main)

    plt.plot(x,y)
    # response_main = popt_main[0]/dT
    plt.show()


def get_power(planck, transmission_frequency, transmission, wavelength, dnu,
T, BFF=1):
    blackbody_intensity = planck(transmission_frequency,transmission, T)
    points = len(blackbody_intensity)
    power = ((sum(blackbody_intensity))/points)*(dnu)
    power = (1/BFF)*power*wavelength**2
    return power


#define normalized 2D gaussian
def gaus2d(x, y, mean_x, mean_y, sigma_x, sigma_y, amplitude):
    first_frac = ((x-mean_x)**2)/(2*sigma_x*2)
    second_frac = ((y-mean_y)**2)/(2*sigma_y*2)
    return amplitude*np.exp(-(first_frac+second_frac))

def calculate_Beam_Filling_Factor(length=100, width=20, FWHM=20, points=5000):

    #Define x and y
    x = np.linspace(0, length, points)
    y = np.linspace(0, length, points)
    x, y = np.meshgrid(x, y) # get 2D variables instead of 1D

    #mean: square box of (longest length)/2
    mean = [length/2, length/2]

    #sigma = FWHM/(sqrt(8log2))
    divisor = np.sqrt(8*np.log(2))
    sigma = [FWHM/divisor, FWHM/divisor]
    z = gaus2d(x, y, mean[0], mean[1], sigma[0], sigma[1], 1)

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(x, y, z, cmap="coolwarm", linewidth=0)

    ax.set_xlabel('x / mm')
    ax.set_ylabel('y / mm')
    ax.set_zlabel('Intensity')
    m = cm.ScalarMappable(cmap="coolwarm")
    cbar = plt.colorbar(m)
    cbar.ax.set_title('Intensity')
```

```python
    plt.show()

    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_surface(x, y, z, cmap="coolwarm", linewidth=0)
    ax.set_zticklabels([])
    ax.set_xlabel('x / mm')
    ax.set_ylabel('y / mm')
    ax.grid(False)
    m = cm.ScalarMappable(cmap="coolwarm")
    cbar = plt.colorbar(m)
    cbar.ax.set_title('Intensity')

    plt.show()
    plt.figure()

    #Integral = sum over all points
    Gauss_integral = np.sum(z)

    #Bar dimensions centered at length/2
    sub = width/2
    first_cuttoff = int(((mean[0]-sub)/length)*points)
    second_cuttoff = int(((mean[0]+sub)/length)*points)

    #Set outside hotbar dimensions = 0
    z[0:first_cuttoff,:] = 0
    z[second_cuttoff:,:] = 0

    #Integrate over all points
    hotbar_integral = np.sum(z)

    #Calculate BFF
    Beam_Filling_Factor = hotbar_integral/Gauss_integral

    return Beam_Filling_Factor


def shot_noise(power, nu):
    shot = np.sqrt(2*power*const.h*nu)
    return shot

def wave_noise(power, dnu):
    wave = power/(np.sqrt(2*dnu))
    return wave

def planck(frequency,transmission, T=293):
    a = (2*const.h*frequency**3)/const.c**2
    b = 1/(np.exp((const.h*frequency)/(const.Boltzmann*T))-1)
    #intensity = a*b*transmission*wavelength**2
```

```
    intensity = a*b*transmission
    return intensity

def transmission_fwhm(x, y):
    #create a spline of x and freq-np.max(blue)/2 to find fwhm
    spline = UnivariateSpline(x, y-np.max(y)/2, s=0)
    r1, r2 = spline.roots() #find the roots
    fwhm = abs(r2 - r1)

    r1_r2_indices = np.where(np.logical_and(x>=r1, x<=r2))

    midpoint = fwhm/2 + r1
    plt.plot(x/1e9, y, marker = ".", color = "black", label = "data",
markersize = 1)
    plt.axvline(x=midpoint/1e9, label="Midpoint", color = "blue", linestyle="-
-")
    plt.axvspan(r1/1e9, r2/1e9, facecolor='g', alpha=0.5, label="fwhm")
    plt.xlabel("Frequency / GHz")
    plt.ylabel("Transmission Fraction")
    plt.legend(loc='center left', bbox_to_anchor=(1, 0.5), fancybox=True)
    plt.show()

    r1 = x[0]
    r2 = x[-1]
    return fwhm, midpoint, r1, r2

#navigate to which file. Default analysis is ch1 for hot bar data
def get_user_input(data):
    #Get Freq
    data_folder_name = input("Data file: ")
    local_file = os.getcwd()
    #Uncomment for use:
    #data_folder_name = input("Data file: ")
    data = fsab.fsab_dirfile(local_file + "\\"  + data_folder_name)

    #Plotting Sweep
    while True:
        Input = input("Know the range? (y/n):")
        if Input.lower() == "y":
            break
        KID_Number = int(input("KID Number?: "))

        #Plot dF0 v Time
        dF0, time = Get_dF0(data, KID_Number)
        plt.plot(time, dF0)
        plt.rcParams["figure.dpi"] = 400
        plt.ticklabel_format(useOffset=False)
        plt.xlabel("time / s")
```

```python
        plt.ylabel("dF0 / Hz")
        plt.title("Time evolution of dF0 for KID number " + str(KID_Number))
        plt.show()
        plt.figure()

        Input = input("Again? (y/n):")
        if Input.lower() == "n":
            break

    #Get dF0 and Time
    KID_Number = int(input("KID Number?: "))

    lower_range = float(input("Lower bound of time peak range?:"))
    upper_range = float(input("Upper bound of time peak range?:"))

    return KID_Number, upper_range, lower_range

def Gaussian(x, height, mu, sigma, c):
  f = c + height*np.exp((-(x-mu)**2)/(2*(sigma)**2))
  return f

def Get_IQ_Sweep(data, KID_Number):
    IQ_data = data.sweep[KID_Number]["z"]
    sweep = data.sweep[KID_Number]["f"]
    tone_freq = data.sweep[KID_Number]["tone_freq"]
    I = IQ_data.real
    Q = IQ_data.imag
    return I, Q, sweep, tone_freq

def Get_IQ_Time(data, KID_Number):
    t = (data.start_time - data.stop_time)
    IQ_data = data.get_iq_data(KID_Number)
    I = IQ_data.real
    Q = IQ_data.imag
    time = np.linspace(0,t, len(I))
    return I, Q, time

def Get_dF0(data, KID_Number):
    #Initialize data
    I_Sweep, Q_Sweep, sweep, tone_frequency = Get_IQ_Sweep(data, KID_Number)
    I_Time, Q_Time, time = Get_IQ_Time(data, KID_Number)

    #Get didf and dqdf
    step = sweep[1] - sweep[0]
    I_Base, Q_Base = 0, 0
    for i in range(0, len(sweep)):
        if sweep[i] == tone_frequency:
            I_Base = I_Sweep[i]
```

```
            Q_Base = Q_Sweep[i]
            didf = (I_Sweep[i+1] - I_Sweep[i-1])/(2*step)
            dqdf = (Q_Sweep[i+1] - Q_Sweep[i-1])/(2*step)

    #Get di and df
    di = I_Time - I_Base
    dq = Q_Time - Q_Base

    #Magic Formula
    dF0 = (di*didf + dq*dqdf)/(didf**2 + dqdf**2)

    #Return
    return dF0, time

def fourier_transform(data, KID_Number):

    #points
    dF0, time = Get_dF0(data, KID_Number)

    plt.plot(time,dF0, color="black", linewidth=3)
    plt.xlabel("Time / s")
    plt.ylabel("dF0 / Hz")
    plt.show()
    plt.figure()
    points = len(time)
    MU = time[-1] - time[0]
    FS = points/MU
    f, Pxx_den = signal.periodogram(dF0, FS)

    return f, Pxx_den

if __name__ == "__main__":
    main()
```