Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Andreas Bernd Thoma

born in Marktredwitz (Germany)

2024

Test of a helium-3 evaporation fridge for the Lsym experiment

This Bachelor Thesis has been carried out by Andreas Bernd Thoma at
the
Max-Planck Institut für Kernphysik in Heidelberg
under the supervision of
Priv. Doz. Dr. Sven Sturm

ABSTRAKT

LSYM is eine Penning Falle, welche aktuell am MPIK Heidelberg entwickelt wird. Das Ziel des Experiments ist es nach möglichen CPT Verletztungen zu suchen, indem man die Differenz der Larmor-Frequenz eines Positrons und eines gebundenen Elektrons mit einer Genauigkeit von 10^{-14} bestimmt. Dazu muss die Cyclotron Bewegung des Positrons in den Grundzustand gebracht werden, was durch koppeln der Bewegung an das Schwarzkörper Feld der Falle geschieht. Diese Falle wird mit einem Helium-3 Evaporationskühler auf eine Temperatur von weniger als 500mK gebracht. Der Kühler reduziert dabei den Druck über einem Bad aus flüssigem Helium, um den Siedepunkt des Heliums zu reduzieren. Insgesamt besteht der Kühler aus vier Pumpsystemen, zwei davon nutzen Helium-4 und zwei Helium-3. Zunächst wird durch pumpen des Helium-4 Systems die Temperatur auf etwa 0.8K reduziert und danach mit dem Helium-3 System auf etwa 300mK. Mithilfe von zwei Modulen lässt sich die Falle kontinuierlich auf unter 500mK kühlen. Bevor der Kühler in das Experiment eingebaut wird, wird dieser getestet, um sicher zu gehen, dass dieser wie erwartet funktioniert und der Last des LSYM-Experiments standhalten kann.

ABSTRACT

LSYM is a Penning trap which is currently being developed at the MPIK Heidelberg. Its goal is to test for possible CPT violations in the lepton sector by determining the difference in Larmor frequencies of a positron and a bound electron up to a magnitude of 10^{-14}. To reach this precision the cyclotron motion of the positron must be cooled to the ground state, which is done by coupling it to the black body field of the trap. This trap will be cooled with a helium-3 evaporation fridge to a temperature of less than 500mK. This refrigerator reduces the pressure above a liquid helium bath with charcoal pumps to reduce the boiling point of the helium. The fridge contains a total of four of these Pump systems with two helium-3 and two helium-4 parts. First pumping the helium-4 reduces the temperature to approximately 0.8K then pumping the helium-3 reduces it further down to approximately 300mK. Having two of these modules allows the trap to stay below 500mK continuously. Before the fridge is implemented into the LSYM experiment, it is tested to ensure that it works as expected and that it can handle the expected heat load of the LSYM experiment.

Contents

List of Figures

List of Tables

1 Motivation and Basics

One of the currently unsolved questions in physics is the difference in matter and antimatter in the observable universe in comparison to the difference of the initially created particle/antiparticle pairs, which can be determined by measuring the cosmic microwave background . One possible explanation for this would be a violation of the CPT invariance [1].

1.1 LSym

LSYM is an experiment which is currently being developed at the MPIK Heidelberg to test for possible CPT violations in the lepton sector by comparing the difference of the spin precession frequency of a positron and a bound electron inside a Penning trap and thereby determining the difference of g-factor and charge to mass ratio up to a relative precision of 10^{-14}.

1.2 Penning traps

Penning traps allow high-precision measurements on charged particles by trapping them inside a homogeneous magnetic and an electric quadrupole field.

This superposition of electric and magnetic field leads to three independent oscillations [2]:
(i) harmonic oscillation along the magnetic field with frequency:

$$\omega_z = \sqrt{\left(\frac{qU_{dc}}{md^2}\right)}, \tag{1.1}$$

where $d = \sqrt{\frac{1}{2}\left[z_0^2 + \frac{1}{2}\rho_0^2\right]}$ is the characteristic trap dimension, where z_0 and ρ_0 are the minimal radial and axial distances to the electrodes [4],

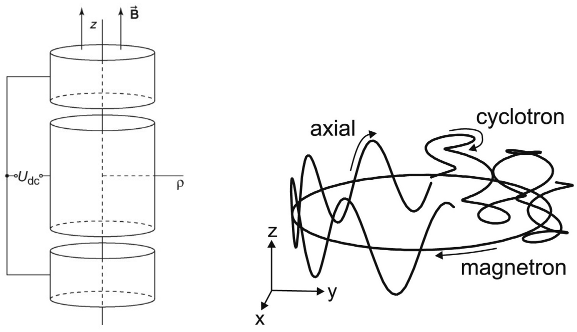

FIGURE 1.1:
Left: Geometry of the cylindrical Penning trap electrodes [2]
Right: Motion inside Penning trap [3]

[5] and U_{dc} the trapping voltage.
(ii) modified radial cyclotron motion:

$$\omega_+ = \frac{\omega_c}{2} + \sqrt{\frac{\omega_c^2}{4} - \frac{\omega_z^2}{2}}, \tag{1.2}$$

(iii) magnetron motion:

$$\omega_- = \frac{\omega_c}{2} - \sqrt{\frac{\omega_c^2}{4} - \frac{\omega_z^2}{2}}, \tag{1.3}$$

where $\omega_c = \frac{qB}{m}$ is the unmodified cyclotron motion with charge q, magnetic field B and mass m.

In LSYM the magnetic field is will be about 5T and is created by a superconducting magnet while the electric field is produced by applying a constant voltage to the endcaps of the traps and multiple cylindrical electrodes. The magnetic field leads to a cyclotron frequency of the positron around 140GHz and about 19 MHz for the helium ion that serves as a surrogate for a free electron. While the axial frequency will be approximately 50MHz for the positron and 600kHz for the helium ion.

To measure these frequencies one can determine the oscillation of the mirror charge inside of the trap electrodes which is induced by the motion of the charged particle. This current in the electrodes has the same frequency as the motion of the charged particle and is typically in the order fA. To measure this cryogenic tank circuits and amplifiers are needed, where the to determined motion is coupled to [6].

1.3 Larmor frequency

LSYM will not use the cyclotron frequency to determine the difference of positron and electron. Instead we will measure the difference of the Larmor-frequencies, which is the precession frequency of the spin of the electron or positron inside a magnetic field. After ensuring the spin of the bound electron and the one of the positron are in the same state, they will be brought onto the equator of the Bloch-sphere by a $\pi/2$ pulse, where they precess with the Larmor frequency [7]:

$$\omega_L = \frac{g}{2}\frac{q}{m}B \tag{1.4}$$

After applying another $\pi/2$ pulse the spin orientation of both particles is determined. Due to the weak binding of the electron to the helium nucleus the precession frequency of the electron is slightly shifted but this shift can be calculated precisely by quantum electrodynamics [8].

1.4 Relativistic frequency shifts

To achieve the precision of 10^{-14} relativistic shift of the frequency due to the motion of the particle also need to be accounted for which can be calculated based on the energy of the motion [9].

1.4.1 Eigenstates of motion

The linear harmonic oscillations of the particle explained in section 1.2 can be quantisized leading to eigenstates of the individual motions [9]:

$$\text{cyclotron: } E_n = \left(n + \frac{1}{2}\right)\hbar\omega_+, \tag{1.5}$$

$$\text{axial: } E_k = \left(k + \frac{1}{2}\right)\hbar\omega_z, \tag{1.6}$$

$$\text{magnetron: } E_l = -\left(l + \frac{1}{2}\right)\hbar\omega_m, \tag{1.7}$$

where n, k, l are the quantum numbers for the cyclotron, axial and magnetron motion and $\hbar$ is the reduced Planck constant.

Additionally, the Thomas precession needs to be accounted for. It describes the precession of the spin due to acceleration of the particle inside of the magnetic field leading to a correction of the Larmor precession [7]. The relative relativistic shift of the Larmor frequency was derived by L.S. Brown and G. Gabrielse and published in [9]:

$$\frac{\Delta\omega_L}{\omega_L} = \frac{1}{mc^2}\left(\frac{\omega_+}{\omega_L}E_n + \frac{1}{2}E_k - \left(\frac{\omega_z}{\omega_+}\right)^2 E_l\right), \tag{1.8}$$

where m is the mass of the helium ion or positron and E_i the energy of the oscillation and ω_i their frequency.

The relative difference of the Larmor frequency when being in the first excited cyclotron state instead of the ground state for the helium ion is therefore:

$$\frac{\Delta\omega_L}{\omega_L} = \frac{1}{mc^2}\frac{\omega_+}{\omega_L}\left(E_{n=1} - E_{n=0}\right) \approx 2.9 \cdot 10^{-21}, \tag{1.9}$$

however the lower mass of the positron and the higher frequencies lead to higher relativistic shifts:

$$\frac{\Delta\omega_L}{\omega_L} = \frac{1}{mc^2}\frac{\omega_+}{\omega_L}\left(E_{n=1} - E_{n=0}\right) \approx 1.13 \cdot 10^{-9}, \tag{1.10}$$

which is by far not enough to reach a precision of 10^{-14}. In this case even the shift from the first excited axial state is:

$$\frac{\Delta\omega_L}{\omega_L} = \frac{1}{mc^2}\frac{1}{2}\left(E_{k=1} - E_{k=0}\right) \approx 1.87 \cdot 10^{-13}. \tag{1.11}$$

Therefore the positron needs to be both in the cyclotron as well as in the axial ground state.

The cyclotron motion couples to the radiation field of black body photons of the surrounding trap and the cyclotron state distributions follows a Boltzmann distribution [9]. The probability to not be in the ground state is proportional to the equilibrium mean thermal photon number [10] :

$$| < n \neq 0 > | \propto \frac{1}{\exp\left(\frac{\hbar\omega_c}{k_B T}\right) - 1}, \tag{1.12}$$

where k_B is the Boltzmann constant and T is the temperature of the trap.

To reduce the impact of the excited cyclotron states the trap temperature needs to be at least 500mK which reduces the factor above to

$1.46 \cdot 10^{-6}$. The axial motion is reduced by cavity assisted side band cooling which couples the axial motion to the cyclotron motion in the trap which leads to both being on average in the same excited state. Further information on this principle can be found in Phys. Rev. Lett. 99, 093902 by Florian Marquardt, Joe P. Chen, A. A. Clerk, and S. M. Girvin [11], in Nature, vol. 475, no. 7356 by J. D. Teufel, T. Donner, D. Li, et al. [12] and Phys. Rev. A 41, 312 by Eric A. Cornell, Robert M. Weisskoff, Kevin R. Boyce, and David E. Pritchard [13].

1.5 Heat transfer

As explained in section 1.4 the trap needs to be cooled to 500mK to reach the ground state of the cyclotron motion. To achieve this temperature the incoming and cooling power on the trap need to be considered. The incoming power can be split into (i) the thermal radiation, (ii) the heat transfer due to conduction and (iii) gas convection.

(i) Thermal radiation:
A black body which is able to absorb radiation of all frequencies will emit energy in form of electromagnetic radiation based on its temperature. This radiation has a continuous frequency spectrum also known as Planck spectrum. Integrating over the whole spectrum leads to the Stefan-Boltzmann law which allows to calculate the total power P emitted per area A with a factor ϵ which describes the fraction of the power which a real body will emit in comparison to an ideal black body [14]:

$$P = \epsilon \sigma A T^4. \tag{1.13}$$

Here, the Stefan-Boltzmann constant $\sigma = 5.670373 \cdot 10^{-8} \frac{W}{m^2 K^4}$ and the temperature T of the body.
To reduce this power a combination of material with low emissivity and multiple shielding layers are used. Placing shields between the cool stage and the warmer stages reduce the power radiated between single layer where it can be dissipated better while also decreasing due to the lower temperature of the inner shields. This process will be further explained in chapter 3.

(ii) Conduction describes the heat flow in material due to collision and diffusion during their random motion.
Differential form of Fourier´s law of thermal conduction gives the local heat flux density q [15]:

$$q = -k\Delta T, \tag{1.14}$$

where k is the material's conductivity which changes with temperature and ΔT is the temperature gradient.
Due to the frequency dependence of k, no simple formula can be applied in cryogenic applications, but for example NIST listed the conductivity values for often used materials and provides an integration tool, which calculates q based on the material, the start and the end temperature [16]. The heat transfer due to conduction in the experiment will be further explained in chapter 3.
(iii) Gas convection: Gas convection describes the heat transfer of a gas due to its movement due to buoyancy. The amount of heat transferred is depends on the viscosity of the gas and its specific heat. The heat conductivity can be calculated by [17]:

$$k = K \cdot h \cdot c_v, \tag{1.15}$$

Where K is a gas dependent constant , h is the viscosity of the gas and c_V its specific heat.
This formula only holds until the mean free path of the gas particle is more than the distance of the two surfaces containing the gas. Otherwise the conductivity get proportional to the gas pressure and can be highly reduced by creating an insulation vacuum [18].

To cool the trap an evaporation cooler which reduces its temperature by evaporating ^{3}He is used. It has a cooling power of $200\mu W$ at 0.4K and it will be further explained in chapter 2.

2 Sorption Cooler

The refrigerator used in LSym to cool the trap and parts of the electronics to the required sub-Kelvin temperature is the sorption cooler "CC7" manufactured by Chase Cryogenic Research.
It can cool below 300mK by reducing the vapour pressure by adsorbing gas above a liquid helium-3 bath. This reduction results into part of the helium evaporating and therefore reducing the temperature of the cold end of the refrigerator. The evaporated helium is then also adsorbed by the charcoal and can be desorbed by heating these, which allows to reuse the expensive helium. Vapour pressure and the evaporation process will be further explained in section 2.2.

2.1 Physical description

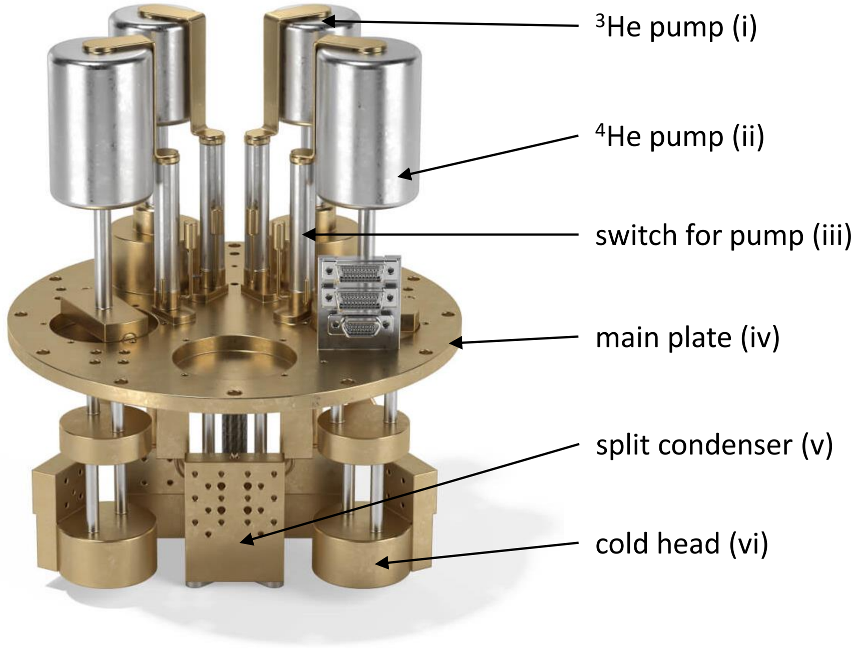

Figure 2.1: CC7 sorption cooler
adapted from Chase Research Cryogenic [19]

The sorption cooler can be split into different parts. First, the so-called "main plate" (iv). It is a gold-plated copper pate with which the refrigerator is mounted to an independent pre-cooler which is kept around 4 Kelvin. This could be a pulse-tube cooler as in the test setup (see chapter 3) used in this work, or a liquid helium-4 bath as it will be in the LSYM experiment. The "main plate" is the "warm" end of the refrigerator and transports the excess heat generated in the recycling process (see section 2.3) to the pre-cooler.

The cooler is made out of two modules, named A and B, which are of identical design. Each of these modules includes two pumps (i,ii), two heat switches (iii) and two cold heads (vi), where one of these belongs to a part which uses ^{4}He and the other one uses ^{3}He for cooling.

During the active operation cycle (see section 2.3) the cold heads (iv) are filled with a small amount of liquid helium, which cools to low temperatures when reducing the vapour pressure wit the soprtion pumps (i,ii). Each module has one cold head filled with ^{4}He and the other one with ^{3}He.

The ^{4}He head can cool down to about 0.8K while the ^{3}He head reaches temperatures down to 300mK. The helium evaporates from the heads into the vacuum up to the charcoal where it is adsorbed taking energy out of the heads to the pump.

In between the ^{3}He-heads of module A and B is the split condenser (iv). It is a golden plated copper plate connected to ^{3}He- heads via passive switches. The temperature of the split condenser decreases when the temperature of the ^{3}He head drops. The switches allow the split condenser to continuously stay at 300mK by only being connected to the heads when their temperature is low and therefore one module can be recycled while the other one cools. When switching between the modules a small temperature spike appears which can be removed by heating the split condenser with a low power heater to a temperature slightly above the peak resulting in a higher but more stable temperature of the split condenser.

The pumps (i,ii) are located on the top end of the cooler and are filled with charcoal. Each of them has a high power heater and is connected to the "main plate" via a heat switch. Heating them allows the helium which was adsorbed by the piece of charcoal during the active cooling to be removed from that charcoal making the helium reusable.

The last part are the heat switches which are normally thermally insulating, but which become conductive when activated by a weak heating.

2.2 Working Principle

2.2.1 Vapour pressure and evaporation process

Vapour pressure describes the relation between pressure and temperature in thermodynamic equilibrium due to vaporization of a substance in a closed system and under the assumption of being a ideal gas can be calculated with the Clausius-Clapeyron Equation:

$$\frac{dp_0}{dT_0} = \frac{p_0 \Delta H}{RT^2}, \tag{2.1}$$

where p_0 is the vapour pressure at temperature T_0, $R = 8.3145\frac{J}{Kmol}$ is the molar gas constant and ΔH is the latent heat of vaporization per mole [20].

The sorption cooler changes the pressure above the liquid helium by pumping with charcoal reducing the pressure. When reducing the pressure, the liquid and gas are not in equilibrium anymore causing some of the liquid helium to evaporate [21].

When the helium evaporates it undergoes a phase transition from liquid to gas. To do so the particle in the liquid must have more energy than its binding energy in the liquid. The required energy for this is also called latent heat of vaporization. When looking at the surface of a liquid the energy of the single molecules is distributed. Therefore some of these have enough energy to leave the surface. When doing so the molecules take energy out of the liquid and effectively cool the liquid [22].

Figure 2.2 shows the temperature and vapor pressure relation for helium-4 and helium-3 below their critical point.

The vapor pressure for helium-4 was plotted by fitting through the measurement points published by R.A Erickson and L.D. Roberts [23], while the curve for helium-3 is taken from Y.H. Huang and G.B. Chen [24].

The temperatures reached with the frigerator used are approximately

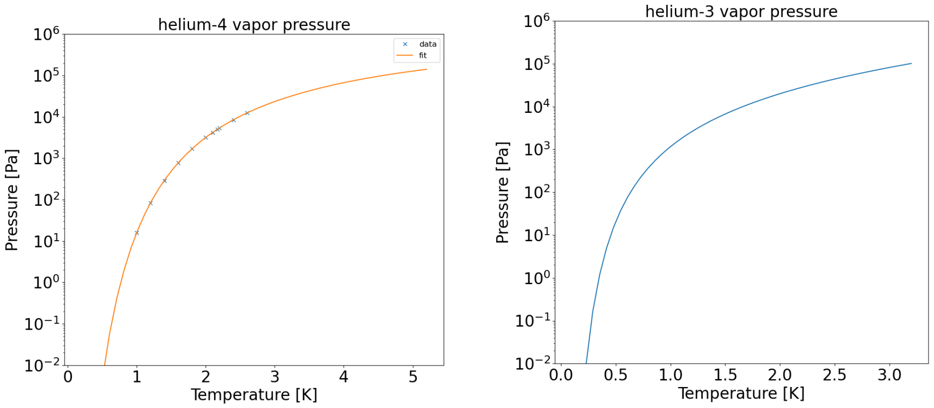

FIGURE 2.2: Vapor pressure of helium-4 and helium-3

0.8K for helium-4 and 300mK for helium-3 [25].

Another method to cool to even lower temperatures with helium are dilution refrigerators. These remove heat from their cold head by diluting highly concentrated helium-3 into a dilute phase where the helium-3 concentration is about 6.5%. This dilute phase flows into a so called still which is keept at around 0.7K. In there mainly helium-3 evaporates due to its lower boiling point and is pumped. After cleaning the helium from possible impurities it is condensed with a 1K bath from where it flows trough a flow impedance and a heat exchange, which allows heat transfer with the diluted phase and thereby cools the helium-3 before it is feed back to the mixing chamber [26].

2.3 Explanation of the cooling cycle

step	^{4}He pump A	^{3}He pump A	^{4}He switch A	^{3}He switch A	^{4}He pump B	^{3}He pump B	^{4}He switch B	^{3}He switch B	comments
0	off	off	off	off	off	off	off	off	waiting to reach operation temperature
1	on	on	off	off	on	on	off	off	recycling of both modules
2	on	on	off	off	on	on	off	off	recycling of both modules
3	off	on	on	off	off	on	on	off	cooling of ^{4}He part both modules
4	off	off	on	on	off	off	on	on	cooling of ^{3}He part both modules
5	on	on	off	off	off	off	on	on	recycling module A, while B cold
6	off	on	on	off	off	off	on	on	cooling ^{4}He part of module A, while B cold
7	off	off	on	on	off	off	on	on	cooling ^{3}He part of module A, while B cold
8	off	off	on	on	on	on	off	off	recycling module B, while A cold
9	off	off	on	on	off	on	on	off	cooling ^{4}He part of module B, while A cold
10	off	off	on	on	off	off	on	on	cooling ^{3}He part of module B, while A cold

TABLE 2.1: Steps of the cooling cycle
adapted from the cooler manual [25]

The cooler uses the principles explained above to continuously reach temperatures below 300mK by alternating the two modules and recycling one while the other one cools. The manual provided by Chase Cryogenic Research splits the operation into 11 steps [25]:

Step 0: Initial cool down
The first step is to pre-cool the sorption cooler with an external source such as a pulse tube cryocooler and wait until all switches and the "main plate" have a temperature of about 4K. This disables the switches and helium gas gets adsorbed by the charcoal in the pump. During this time the pumps are already cold enough to adsorb the gas while the helium is still above its critical temperature.

Step 1: Heat all pumps to 45K
First a voltage of 2V are applied to all pumps to slowly increase the pump temperatures, which is then increased to 15V for the ^{4}He-pumps and 12V for the ^{3}He-pumps, until the pumps reach a temperature of 45K. After this a stabilising voltage of 3V is applied to each pump.

Step 2: Wait for heads to stabilise
Keep the pumps at 45K until most of the helium is removed from the charcoal. During this time the ^{4}He and ^{3}He-heads cool to the "main plate" temperature. During this step it is important to ensure the ^{4}He-head temperature cools well below the critical point 5.2K in order to liquefy a significant fraction of the helium gas.

Step 3: Turn off ^{4}He Pumps and turn on ^{4}He switches
The pump temperature decreases rapidly, causing the helium gas to adsorb onto the charcoal surface, which reduces the the pressure above drastically, which has previously been collected in the ^{4}He-head. The reduced pressure results in a reduced boiling point of the helium, which starts boiling and therefore reduces the temperature of the remaining ^{4}He liquid as well as the passively cooled ^{3}He-head. The minimum temperature reached is approximately 0.8K.

Step 4: Turn off both ^{3}He Pumps and turn on both ^{3}He switches
Once the ^{3}He-head is at 1.6K, which is well below its critical point at 3.3K and therefore liquefied, the voltage to the ^{3}-He pump is set to zero while the voltage for the switch is increased from 1.8V to 3V over time. The ^{3}He Pump temperature decreases quickly,once the switch opens, which also decreases the vapour pressure in the ^{3}He system. The liquid helium in the head evaporates which cools the head and also the split

condenser down to about 300mK.

Step 5: Turn off both switches and turn on both pumps of module A and keep them at 45K until both heads stabilised
Start recycling module A by turning off the ^{4}He and ^{3}He switch and again heating ^{4}He and ^{3}He pump to 45K and keep them at this temperature for some time while the switches of module B stay on. The helium inside the ^{4}He and ^{3}He pump of module A is removed from the charcoal again and the vapour pressure rises.

Step 6: Turn off ^{4}He pump A and turn on ^{4}He switch A
The pump temperature and vapour pressure in the ^{4}He system of module A drop and both heads cool to 0.8K.

Step 7: Turn off ^{3}He pump A and turn on ^{3}He switch A
The pump temperature and vapour pressure in the ^{3}He A system drop and ^{3}He heads cool to 300mK. The head is reconnected to the split condenser and contributes to cooling of the split condenser.

Step 8: Turn on both pumps of module B and keep them at 45K until both heads stabilised
Once the ^{3}He-head of module B runs out of helium, its temperature increases slightly. At this point both pumps of module B are turned on and kept at a temperature of 45K. The vapour pressure increases again due to the helium being desorbed from the charcoal and the heads adjust to the "main plate" temperature.

Step 9: Turn off ^{4}He pump turn on ^{4}He switch of module B
The ^{4}He pump temperature and vapour pressure decrease and both head temperatures drop to 0.8K.

Step 10: Turn off ^{3}He Pump and turn on ^{3}He switch
The ^{3}He pump temperature and vapour pressure decreases. The ^{3}He head temperature drops to 300mK and thermalises the split condenser.

To keep a continuous 300mK heat sink steps 5 to 10 are repeated as often as needed.

2.4 Temperature sensors

When cooling down the device we are also interested in the temperature at different parts of the sorption cooler. Two different types of sensors were used for this.
(i) Diode thermometers at the "main plate", the pumps and the switches: diode thermometers use the temperature dependent voltage drop at a constant current to determine the temperature [27]. The diodes used at the pumps, switches and "main plate" are silicon diodes calibrated between 320 and 0.8K and have their highest sensitivity at around 7K [25]. Figure 2.3 shows the temperature dependent voltage at an excitation current of 10μA and the sensitivity at these temperatures.

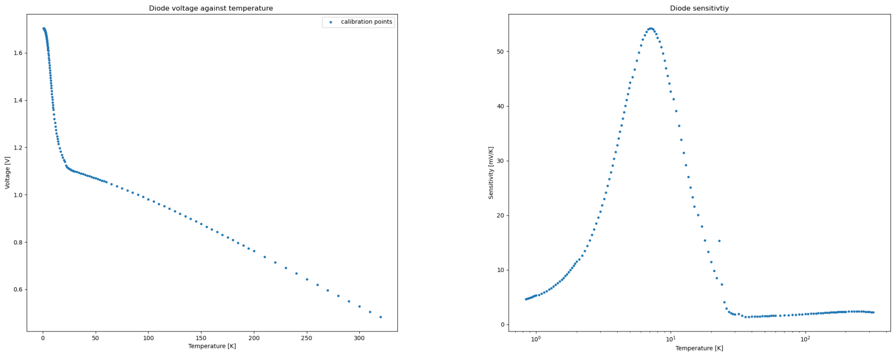

FIGURE 2.3:
Left: Calibration points of the diode sensors
Right: Sensitivity of the diode sensors

(ii) Resistance thermometer at the split condenser and the heads: resistance thermometers use the temperature dependent resistance of a sensor to determine its temperature. There are two types of theses sensors. One where the resistance increases with temperature, therefore they have a positive temperature coefficient sensors (PTC) and the other one have a negative temperature coefficient (NTC) [28].
To connect the sensor a cable is needed which also has a resistance and therefore has to be subtracted from the determined resistance to get the sensor resistance. The Resistance of the cable itself is often also temperature dependent. One way to encounter this problem is by connecting a total of four wires where on each side one is connected to the sensor directly and the other one is connected to this wire close to the sensor. This way the resistance of both wires can be determined without

including the resistance of the sensor then the total resistance of wire and sensor is measured. By subtracting the wire resistance from the total resistance, the sensor resistance is determined [29]. When measuring like this the resistor acts as a heater which could potentially increase the temperature [30].

The resistance sensors used at the heads and split condenser are made out of ruthenium oxide and are NTC sensors with a relatively low resistance of a few $k\Omega$ at 300mK [25]. When the excitation current is set to 100nA the heating power of the sensor at the split condenser is:

$$P = R \cdot I^2 = 5615\Omega \cdot (100nA)^2 = 56.15pW, \tag{2.2}$$

where R is the resistance of the sensor and I the current [30]. Compared to the few μW cooling power, the heating due to measuring the temperature sensor is small.

The resulting self heating of the sensor was determined by determining the measured temperature at different excitation currents, when the split condenser was at approximately 312 mK. With the slope of a linear fit, the self heating at the lowest excitation current can be determined [30]. Figure 2.4 shows the excitation current and the measured temperature.

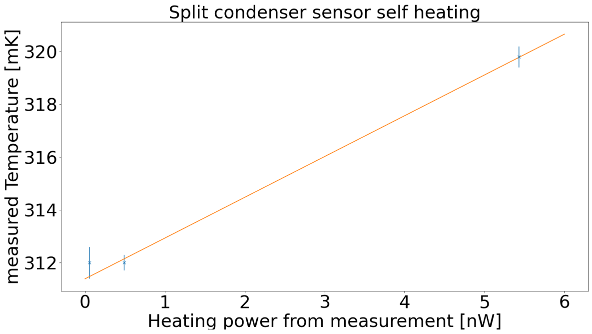

Figure 2.4: Split condenser self heating

The slope m of this fit is:

$$m = (1.55 \pm 0.1)\frac{mK}{nW} \tag{2.3}$$

An excitation current of 100nA results into a self heating ΔT_{SH} of:

$$\Delta T_{SH} = (0.087 \pm 0.005)mK \tag{2.4}$$

The individual calibration points and the sensitivity of this sensor can be seen in figure 2.5.

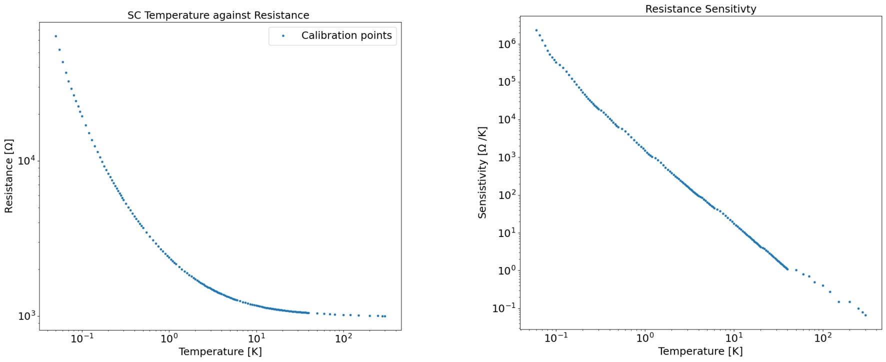

Figure 2.5:
Left: Calibration points of the split condenser sensor
Right: Sensitivity of the split condenser sensor

3 Test setup

Before the refrigerator, which was introduced in chapter 2, will be installed into LSYM, it needs to be tested and optimized. For this, a test setup is needed, which has to be able to cool the "main plate" of the fridge to around 4K, supply voltages to all of the heaters and read out the temperatures at the split condenser, pumps and heads. This setup can be seen in figure 3.1 and it will be further explained in this chapter.

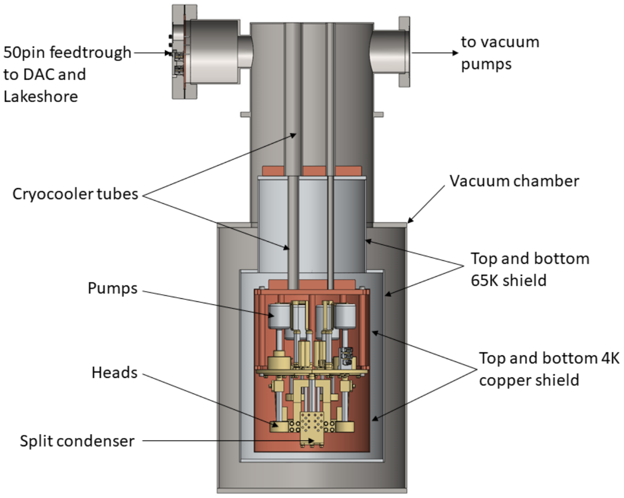

FIGURE 3.1: Test setup of the evaporation fridge

3.1 Devices

The test setup includes multiple devices to control the cooler and analyse the temperature. To ensure minimal heat transfer via conduction and convection and to avoid freezing moist air the setup is placed inside a vacuum chamber where the air was pumped out with first a rotary pump down to a pressure of about 1×10^{-2} mbar and further with a turbo molecular pump down to a pressure of about 5×10^{-8} mbar.

For pre-cooling the refrigerator a RP-062B 4K Pulse Tube Cryocooler

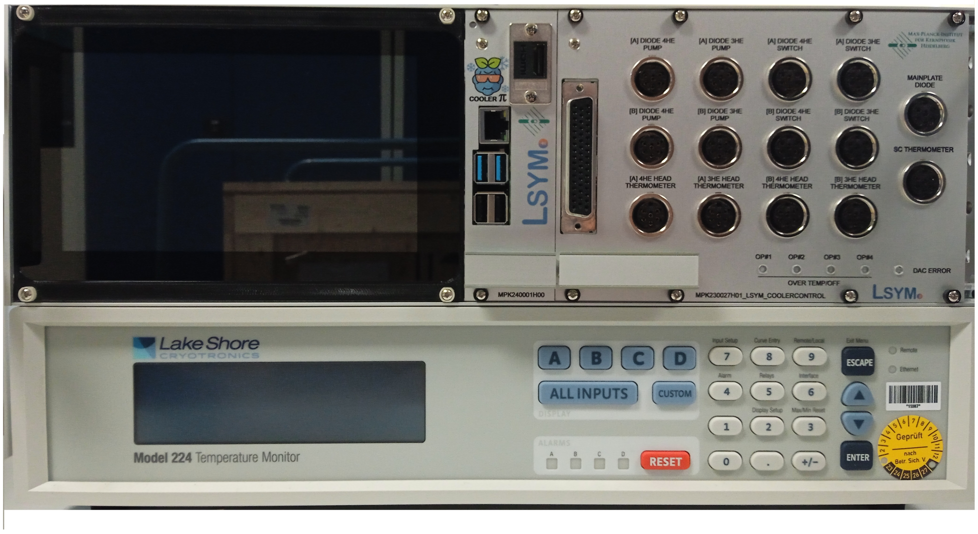

FIGURE 3.2: Lakeshore Model 224 and DAC

was used. It has two cold stages, one at 65K with a cooling power of 30W and one at 4.2K with a cooling power of 0.5W. The colder stage can reach less then 3.0K according to the manufacturer and its refrigeration power depends on the temperatures of both stages [31]. The vacuum chamber consist of two parts, an smaller cylinder on top and a larger cylinder on the bottom. The dimensions can be seen in table 3.1.

To measure the temperatures at the different parts of the refrigerator a total of 14 temperature sensors are installed on it and ten of them are needed for the operation [25]. These sensors are connected via a custom made cable to the outside and from there to a lakeshore Model 224 temperature monitor (see figure 3.2). This device allows to read out the measurements of the sensors in real time. The manufacturer provides python drivers with which temperature dependent steps can be implemented.

The voltage supply to the sorption cooler was realized with a DAC (Digital to Analog Converter, see figure 3.2) which was built by the electronics department of the MPIK. It can be controlled with a Raspberry Pi and has a total of nine voltage outputs where all of these are power amplified and can supply up to 10.5 Volt. Four of these outputs are additionally voltage amplified with a factor of 6 to support the high power heaters which can take up to 25V. The programs, which control the voltage output will be explained in chapter 4.

In order to operate at sub-kelvin temperatures the heat radiations should be minimized. This is achieved by two radiation shields. These are cylinders which are physically connected to the cold head stages and absorb and reflect some of the thermal radiation coming from warmer stages. One at the 4.2K stage, which completely surrounds the 300mK cooler and the other one is connected to the 65K stage and surrounds the inner shield. Around the 65K shield is the vacuum chamber at room temperature with a gap of a few centimeters in between. Each shield can be split into a top and a bottom cylinder. The dimensions of each shield can be seen in table 3.1.

stage		radius [cm]	height [cm]
300K	top	10.45±0.05	17.0±0.1
	bottom	13.05±0.05	40.5±0.1
65K	top	8.75±0.05	8.5±0.1
	bottom	11.5±0.05	31.0±0.1
4k	top	9.25±0.05	12.1±0.1
	bottom	9.25±0.05	14.0±0.1

TABLE 3.1: Shield dimension at the 300K, 65K and 4K stage

3.2 65 Kelvin stage

The shield at 65K is made out of aluminium which has an emissivity of about 0.1 [32] and is additionally covered in multi layer insulation which contains a total of 10 layers of aluminium foil which are separated by spacer layers with low thermal conductivity material, typically woven fibers. The emissivities depend on the temperature as well as the finishing process of the material and the oxidation of the material over time. Additionally multiple sources give different values. For example in [33] the emissivity for mechanically polished aluminium at 65K is approximately 0.8%. For this calculation the highest value was used, to not underestimate the actual heat radiation. The shield is connected to the 65K stage of the cold head which has a cooling power of 30W at this temperature. To ensure that the temperature of the shield approaches that value, the incoming thermal radiation power from the room-temperature stage has to be sufficiently low and the thermal conductivity from the shield to the cold stage has to be sufficiently high.

The power emitted between the vacuum tubes and the multi layer insulation towards the aluminum shield can be estimated by assuming two infinitely large heat reservoirs at temperature $T_1 = (290 \pm 5)$ and $T_2 = (65 \pm 1)K$ with multiple layers in between: The temperature of these layers depends on the incoming and out going radiation. The incoming radiation e.g. $P_{1,down}$ from the vacuum tubes to the first layer of the MLI depends on the fraction of photons that get absorbed in the MLI. The fraction that doesn't get absorbed will be reflected back to the vacuum chamber, where it can also either be absorbed or reflected back to the MLI. This leads to an effective emissivity of:

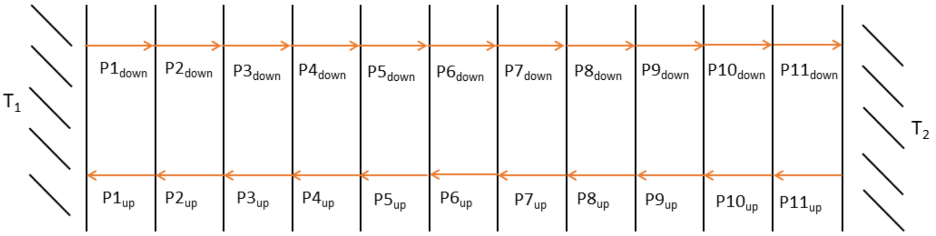

Figure 3.3: Heat radiation in multi layer insulation

$$\frac{1}{\epsilon} = \frac{1}{\epsilon_1} + \frac{1}{\epsilon_2} - 1, \tag{3.1}$$

where ϵ_1 and ϵ_2 are the emissivities of both materials [32]. At thermal equilibrium the sum of these powers P needs to be zero at each layer. Additionally the power going from one stage to the next colder one and next warmer layer is the same. Therefore:

$$P_{i,down} + P_{i+1,up} - P_{i+1,down} - P_{i,up} = 0, \qquad \text{for } i \in [1,10] \tag{3.2}$$

$$P_{i+1,down} = P_{i,up}, \qquad \text{for } j \in [1,10] \tag{3.3}$$

Considering the incoming and outgoing radiation of all layers, leads to a power $P_{in,65K}$, which is the radiation power, that flows from the last insulation layer to the aluminum shield and is absorbed there [32]:

$$P_{in,65K} = \frac{P_{1,down}}{N+1} = (0.98 \pm 0.03)W, \tag{3.4}$$

where $P_{1,down}$ is the heat radiation coming from the vacuum tubes which have an emissivity of 0.0919 [34] that is absorbed by the aluminum shield, and N is the number of layers in the MLI. The power was calculated with

equation (1.13) and (3.2) and its error by Gaussian error propagation of the error in temperature and surface area.
Additionally to the heat radiation, the spacer will transfer heat via solid conduction. The solid conductivity per unit thickness is [35]:

$$k_s = C_2 \cdot f \cdot k/\Delta X, \tag{3.5}$$

where C_2 is an empirical constant, f is the relative density of the separator compared to solid material, k is the separator material thermal conductivity compared to solid material and ΔX is the thickness of separator between reflectors.
The manufacturer gives a total heat flux of more than $1.00\,\mathrm{W/m^2}$ for 10 layers of aluminium and 10 spacers in a temperature range of 300K to 77K [36].

3.3 4 Kelvin stage

The next stage is at 4.2K, where the cold head has a cooling power of 0.5W. The shield is made out of copper which has an emissivity of 0.02 [32] at 4K. Here the value is also different depending on the surface finnish and different sources give different values (see [33]). Using an emissivity of 0.02 leads to an incoming power of:

$$P_{in,4K} = (6.9 \pm 0.2) mW. \tag{3.6}$$

The power from the copper shield towards the inside, which is partly absorbed by the split condenser is:

$$P_{out,4K} = (0.039 \pm 0.008) \mu W. \tag{3.7}$$

The other heat source mentioned in chapter 1 is conduction. In the experiment we need to be able to measure the temperature and provide different voltages to the cooler. For this, electrical connections are needed, which will transfer heat depending on the material and length of the cable. In the LSYM experiment the cable will be approximately 30cm.
The cable used for the low power heater and the temperature sensors was made out manganin and has a diameter of 200 microns. In total

Figure 3.4: Left: Aluminum shield with multi layer insulation which is at the 65K stage
Right: Copper shield which is at the 4K stage

there are 40 connections of this type. Additionally the connections for the four high power heaters are made out of eight 100 micron copper beryllium wires. Adding all wires together leads to a total heat transfer of:

$$P_{cable} = (n_{Mn} \cdot \lambda_{Mn} \cdot A_{Mn} + n_{CuBe} \cdot \lambda_{CuBe} \cdot A_{CuBe})/l \approx 3.6mW, \tag{3.8}$$

where n is the number of cables, A is the cross-section of the wire and l its length. λ is the thermal conductivity assuming the cable is thermalised at the ends, where one end is at 4.2K and the other one at 77K. To calculate the heat flux, the temperature dependent thermal conductivity must be integrated over the temperatures at both ends [37]. For copper beryllium this integral is $\lambda_{CuBe} = 1513\,\frac{\mathrm{W}}{\mathrm{m}}$ [38] and for manganin $\lambda_{mn} = 568.2\,\frac{\mathrm{W}}{\mathrm{m}}$ [37].

In case of the test setup no calculation was done. The cable here was approximately 1m long and is wrapped around the tube of the pulse tube cryocooler before being connected to the refrigerator. This additional connection makes it complex to calculate, because the exact temperature at the connection point as well as the amount of heat transfer to the tube is not known. However the cable is about three times as long, therefore it can be said that the heat transfer will be less than the available cooling power of 0.5W at the 4K stage.

stage	power influx	available cooling power
65K	(0.98 ± 0.03)W	30W
4.2K	(6.9 ± 0.2)mW	500mW
300mK	$(0.038 \pm 0.008)\mu W$	$100\mu W$

Table 3.2: Heating due to radiation and cooling power at different stages of the test setup

Table 3.2 shows the incoming heating power radiation compared to the cooling power at different stages of the setup. At all stages the heat influx is less than the available cooling. This is especially important for the 300mK stage, which has by far the smallest cooling power and would otherwise not be able to reach this temperature.

4 Control Programs

To automate the cooling cycle of the refrigerator (see chapter 2) a control system is needed, that sets the voltages needed for each heater depending on the current cooling step and that save all data. To this end, python scripts have been developed within this work which control the heater voltages (DAC control) and the cooling cycles (Fridge Control), provide a graphical user interface (Fridge GUI) and store the recorded cycle data in a common database. The details of this control system will be explained in this chapter. One benefit of using python is Lakeshore provides python drivers for the Model 224 temperature monitor. An overview of the system can be seen in figure 4.1.

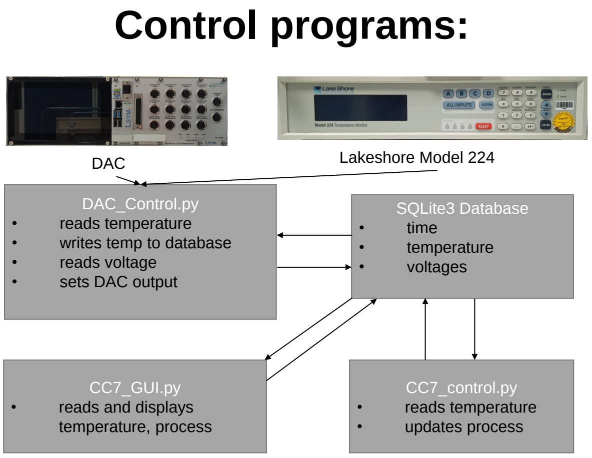

FIGURE 4.1: Visual representation of programs communicating with each other and their main purposes

4.1 sqlite3 database

To store the voltages, temperatures step and time, and to allow different programs to communicate with each other a common SQLite3 [39] database was used. The structure of this database can be seen in table 4.1.

column name	unit	explanation
time	s	time beginning when the DAC control starts
step		current step of the operation as explained in section 2.3
SC_temp	K	temperature of the split condenser
MP_temp	K	temperature of the "main plate"
Pump_4A_Temp	K	temperature of the ^{4}He-pump of module A
Pump_3A_Temp	K	temperature of the ^{3}He-pump of module A
Pump_4B_Temp	K	temperature of the ^{4}He-pump of module B
Pump_3B_Temp	K	temperature of the ^{3}He-pump of module B
Head_4A_Temp	K	temperature of the ^{4}He-head of module A
Head_3A_Temp	K	temperature of the ^{3}He-head of module A
Head_4B_Temp	K	temperature of the ^{4}He-head of module B
Head_3B_Temp	K	temperature of the ^{3}He-head of module B
Pump_4A_Voltage	V	voltage applied to ^{4}He-pump of module A
Pump_3A_Voltage	V	voltage applied to ^{3}He-pump of module A
Pump_4B_Voltage	V	voltage applied to ^{4}He-pump of module B
Pump_3B_Voltage	V	voltage applied to ^{3}He-pump of module B
Switch_4A_Voltage	V	voltage applied to the switch for the ^{4}He-pump of module A
Switch_3A_Voltage	V	voltage applied to the switch for the ^{3}He-pump of module A
Switch_4B_Voltage	V	voltage applied to the switch for the ^{4}He-pump of module B
Switch_3B_Voltage	V	voltage applied to the switch for the ^{3}He-pump of module B
PID_Voltage	V	voltage applied to heater on the split condenser

Table 4.1: Stored Information in SQL database

4.2 DAC Control

The main purpose of the DAC Control is to:

- set all output voltages
- time since start
- read the temperature from the Lakeshore device and write these to the database

In this script the current temperatures are read from the Lakeshore monitor device and saved in the database. Then, the control voltages are updated and saved as well.
One cycle in this script:

- updates the timer
- reads the temperatures from the Lakeshore
- reads the new voltage (requested by the FridgeControl script) from the database
- writes the time and temperatures to the database

- updates the DAC output

An example of the table inside the database after starting the DAC control:

time	step	Pump_4A_Temp	Head_4A_Temp	Pump_4A_Voltage	Switch_4A_Voltage	PID_Voltage
0	0	3.0000	3.2400	0	0	0
2	0	3.0000	3.2300	0	0	0

TABLE 4.2: Database after starting DAC Control

To keep the example simple only the values for the ^{4}He-part of module A were included.

Additionally some safety features were implemented in this program to handle possible errors. These include creating the database if it doesn't exist and writing zeros as standard values for the voltages in the database to prevent overheating in case the Fridge control crashes.
The python code is reproduced in appendix A.

4.3 Fridge Control

While the "DAC control" handles the communication with the actual devices, the "Fridge Control" handles the steps of the cooling cycle as explained in section 2.3. Its main purpose is to:

- read temperature from database
- handle current step of the cooling cycle
- write voltages to database

The steps themselves are implemented as explained in section 2.3 inside one python class, where the individual voltages are determined and stored in the database every two seconds.
It starts at step 0 where it uses the temperature from the database to wait for the "main plate" to cool to 4K. After that the next step is started and the script goes through the sub-steps of each step. These sub-steps depend on a fixed time or the temperatures. For example the stabilising of the helium-4 heads needs a certain amount of time but also a certain helium-4 head temperature. A timer is started once the stabilising begins. Every two seconds the timer and temperature are updated and the script tests if the head temperature is below 4.2K and if the time spend stabilising is longer than 900 seconds.

One potential error during the run is if a sensor connected to the Lakeshore temperature monitor is currently at a temperature outside of its set temperature range. Then instead of the correct temperature a 0 would be written to the database. To prevent misinterpreting the temperature the individual steps check if the temperature is above zero when needed. If both is fulfilled the next sub step begins.

Adding the start of the fridge control to table 4.2:

time	step	Pump_4A_Temp	Head_4A_Temp	Pump_4A_Voltage	Switch_4A_Voltage	PID_Voltage
0	0	3.0000	3.2400	0	0	0
2	0	3.0000	3.2300	0	0	0
4	1	3.2000	3.2250	2	0	0
6	1	3.2050	3.2100	2	0	0

Table 4.3: Database after starting Fridge Control

The python code is reproduced in appendix B

4.4 Fridge GUI

The fridge GUI gives a visual representation of the current cooler status. It reads the current split condenser temperature from the database and displays this in mK graphically using a tkinter [40] interface. To provide immediate intuitive user feedback, the background changes between green and red depending on the temperature. It is green when the temperature is in the wanted interval of the set point otherwise red. The python code is reproduced in appendix C.

The combination of these three programs allows to continuously supply voltage to the heaters on the refrigerator without supervision while also saving this data for analysis. The test of these programs and the cooler will be discussed in the next chapter.

5 Test results

With the test setup (see chapter 3) and the control system (see chapter 4) the refrigerator had to be tested to ensure the cooler and the programs work as expected and that the cooler can handle the expected heat load of the LSYM experiment before it is implemented into the LSYM experiment. The test results will be presented in this chapter.

5.1 First Cooldown

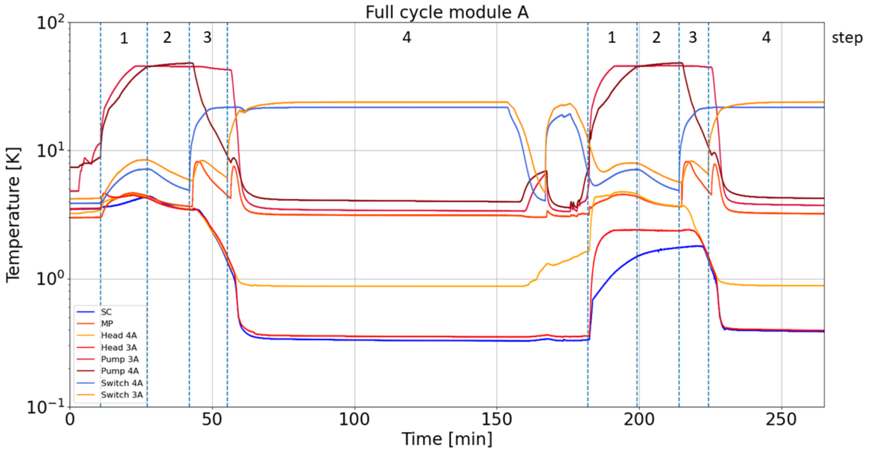

FIGURE 5.1: Full run of Module A
In blue the split condenser, in red the ^{3}He-head, in orange the ^{4}He-head, in orange red the "main plate", the ^{4}He-pump in crimson, the ^{3}He-pump in dark red, the switch for the ^{4}He-pump in royal blue and the switch for the ^{3}He-pump in dark orange.

Figure 5.1 shows the plot of temperature over time of one run with module A, where the individual steps (see section 2.3) were performed. The second module could not be run in the first cooldown, due to a damaged helium-4 pump heater of module B.

Step 0, which is waiting for the "main plate" to cool below 4K is not visible in this plot.

Step 1 (minutes 0 to 27.2) increases the voltage of the ^{4}He-pump to 15V and the voltage of the ^{3}He-pump to 12V. Their temperatures now rise until they reach a temperature of 45K. Then a stabilising voltage of 2.8V for the ^{4}He-pump and 3.8V for ^{3}He-pump is applied. During this time the "main plate" and head temperatures spike up to 4.7K and the switches up to 8.5K. It is important for the switches to stay below 12K in this step otherwise the pumps would dissipate heat through the switches to the "main plate" and heat the complete system.

Step 2 (minutes 27.2 to 42.1) The stabilising voltages for both pumps are kept on for 200 seconds. During that time the heads need to be cooled well below 5.2K, otherwise the helium-4 would be readsorbed in the next step before it is liquefied. In our case the heads reached a maximum temperature of 4.67K and settled at 3.51K before the heaters were turned off.

Step 3 (minutes 42.1 to 55.3) The heater of the ^{4}He-pump is turned off while the switch is slowly turned on. Slowly in this case means at first a voltage of 1.8V was applied and increased to a maximum of 3V over 180s. When the switch temperature reaches 12K at minute 43 the pump temperature starts to drop down to 8.0K. Once the pump temperature reaches approximately 30K the temperature of both heads begin to fall. At minute 55 the heads and the split condenser reached a temperature of 1.5K. During this step a temperature difference of up to 0.1K between the split condenser and the helium-3 head was measured.

Step 4 (minutes 55.3 to 153.7) The heating voltage of the ^{3}He-pump is set to zero while the switch voltage is slowly increased starting from 1.8V to a maximum of 3V. Once the switch reaches 12K at min 56, the pump temperature falls while the "main plate" temperature temporarily increases. Even the ^{4}He-pump temperature shortly increases when the "main plate" hits its maximum. At minute 58 the charcoal starts to adsorb helium-3 and the ^{3}He-head as well as the split condenser cool down. The initial cooling to 470mK takes place within 2 minutes (see at minute 60). After this the temperature further decreases to 328mK over a time span of 103 minutes.

At minute 154 both switch heaters are turned off. Once these cool below 12K both pumps increased their temperature until minute 167, where the switches are turned on again, allowing heat to flow from the pumps to the "main plate" which therefore also increased slightly in temperature to 3.4K. Additionally the ^{4}He-head increased in temperature once the pumps heat up. At minute 154 both switch heaters are turned off. As a result they cool to the "main plate" temperature and reduce the heat transfer from the pumps to the "main plate" significantly. During this time the helium-4 head temperature starts increasing and the switches are heated again at minute 167.1.

step		SC T[mK]	MP T[K]	^{4}He-Pump T[K]	^{3}He-Pump T[K]	^{4}He-Head T[mK]	^{3}He-Head T[mK]	^{4}He-switch T[K]	^{3}He-switch T[K]
1	min	3465 ±80	2.988±0.016	7.402±0.012	4.846±0.014	3232±45	3515±50	3.866±0.014	4.194±0.014
	max	3554 ±89	3.023±0.016	8.921±0.012	11.4±0.015	3427±48	3575±51	3.896±0.014	4.2452±0.014
2	min	3527 ±87	3.006±0.016	8.930±0.012	11.4±0.015	3428±48	3576±51	3.896±0.014	4.245±0.014
	max	4410 ±110	4.675±0.014	45.1±0.09	45.7±0.10	4548±80	4656±93	7.186±0.012	8.4673±0.012
3	min	3453 ±80	3.671±0.015	45.1±0.09	45.4±0.10	3502±50	3455±49	4.901±0.014	5.904±0.012
	max	4418 ±110	4.445±0.014	48.1±0.10	45.6±0.10	4322±80	4250±77	7.187±0.012	8.449±0.012
4	min	1363 ±13	3.659±0.015	9.134±0.012	42.8±0.10	1411±9	1514±9	4.899±0.014	5.806±0.012
	max	3481 ±82	8.197±0.012	48.1±0.10	45.4±0.10	3538±52	3530±50	21.7±0.02	8.378±0.012
5	min	328 ±3	3.112±0.016	3.997±0.013	3.388±0.015	874±6	353±3	19.9±0.017	6.081±0.012
	max	1357 ±13	7.545±0.012	9.096±0.012	42.8±0.10	1405±9	1508±9	21.8±0.02	23.9±0.02

TABLE 5.1: Temperatures during each step of the cooling cycle

Table 5.1 shows the minimum and maximum temperature of each sensor during each step from step 1 to step 5.

The uncertainty included in the temperature measurement of the "main plate", switches and pumps are:

i) the uncertainty from fitting Chebychev polynomials [41] to the sensor calibration data (see figure 2.3) in the corresponding temperature intervals [41]

ii) the uncertainty from the readout which is ($80\ \mu V \pm 0.005\% rdg$) [29] converted in an uncertainty of temperature by dividing through the sensitivity (see figure 2.3).

iii) the typical calibration uncertainty of the sensor which is 12mK at a temperature of 4.2, 12mk at 12K, 14mK at 20K, and 22mK at 77K [41]

The uncertainty included in the temperature measurement of the split condenser and heads are:

i) the uncertainty from fitting Chebychev polynomials [42] to the sensor calibration data (see figure 2.5) in corresponding temperature intervals [41]

ii) the uncertainty from the readout which is ($10\ \Omega$ + 0.04% rdg) [29] converted in an uncertainty of temperature by dividing trough the sensitivity at this temperature (see figure 2.5).

iii) the typical calibration uncertainty of the sensor which is 3mK at a temperature of 300mK, 5mK at 4.2K and 6mK at 10K [41]

Another source of uncertainty is the resistance of the cryogenic 2-wire cabling, which is subtracted from the measured resistance and which differs by up to 20% [37] between room temperature and 4K.

In step 5 the difference in temperature between the split condenser and the ^{3}He-head of module A is more than 3σ. The test provided by the manufacturer also had this difference of approximately 20mK between the helium-3 head and split condenser senors. They stated in the manual that the temperatures from the sensors at the heads should be regarded as indicative [25].

The minimum temperature reached in the manufacturers test was approximately 325mK, when running with only one module, which is comparable to the temperature reached in this test.

5.2 Second Test

After the damaged heater was repaired, the individual steps are repeated the same way as in section 5.1, first for both modules simultaneously, then with an offset of about two hours. The result can be seen in figure 5.2, while figure 5.3 shows one cycle in this run, including recycling of both modules one time each.

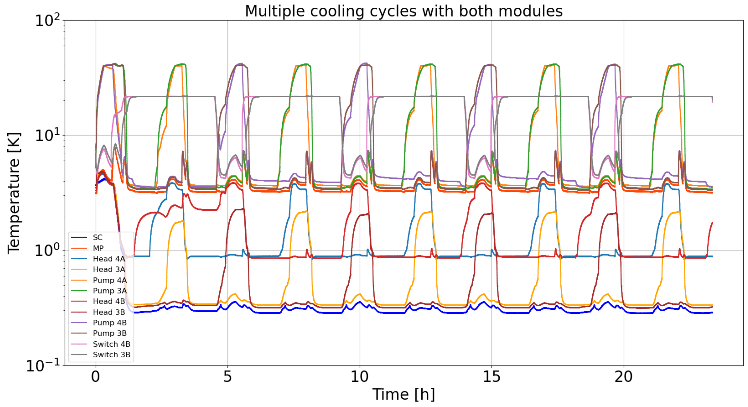

FIGURE 5.2: Multiple cycles with both modules

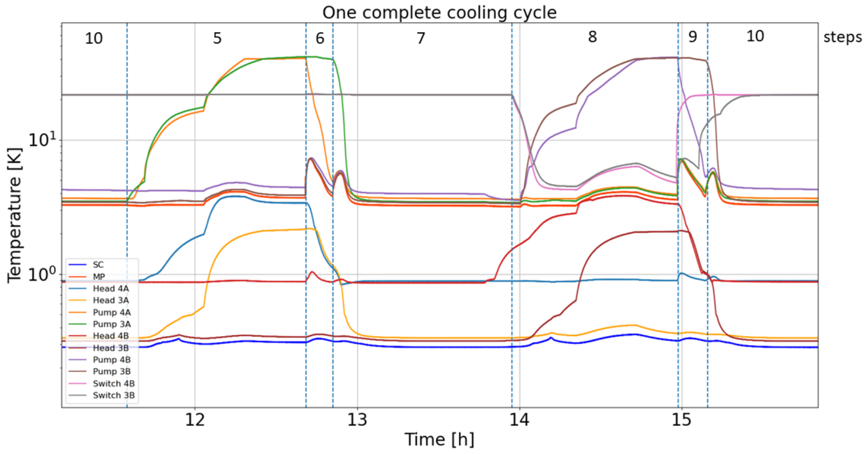

FIGURE 5.3: One complete cooling cycle with both modules being recycled

A few changes have been made in comparison to the first test. The maximum temperature of the pumps during the recycling process was reduced to 40K and the switch opening time was increased, which lead to a lower "main plate" temperature, when cooling the pumps. The stabilising time was increased to 900s, to increase the time before the helium-4 heads run out. The individual minimum and maximum temperatures during each step can be seen in table 5.2.

step		SC T[mK]	MP T[K]	^{4}He-Pump A T[K]	^{3}He-Pump A T[K]	^{4}He-Head A T[mK]	^{3}He-Head A T[mK]
5	min	285±3	3.150±0.014	3.477±0.015	3.4376±0.015	892±6	335±3
	max	333±3	4.179±0.014	40.6169±0.10	41.76±0.10	3857±62	2174±20
6/7	min	285±3	3.158±0.015	3.6057±0.015	3.3993±0.015	831±6	335±3
	max	343±3	7.244±0.012	40.6209±0.10	41.80±0.10	3411±48	2204 ±20
8	min	286±3	3.153±0.015	3.5993±0.015	3.4003±0.015	874±6	336 ±3
	max	359±3	4.129±0.014	4.4804±0.014	4.4184±0.014	917±6	420 ± 3
9/10	min	284±3	3.232±0.015	3.6495±0.015	3.4642±0.015	888±6	335± 3
	max	337±3	7.303±0.012	7.317±0.012	7.328±0.012	1022±7	372±3

step		^{4}He-Pump B T[K]	^{3}He-Pump B T[K]	^{4}He-Head B T[mK]	^{3}He-Head B T[mK]	^{4}He-switch B T[K]	^{3}He-switch B T[K]
5	min	3.520±0.015	3.372±0.015	854±6	316.8±3	21.71±0.11	21.72±0.11
	max	4.826±0.014	4.338±0.014	2479±25	360.9±3	21.78±0.11	21.81±0.11
6/7	min	3.534±0.015	3.348±0.015	856±6	317.1±3	21.71±0.11	21.72±0.11
	max	7.391±0.012	7.289±0.012	3110±39	370.1±3	21.92±0.11	21.99±0.11
8	min	3.518±0.015	3.347±0.015	856±6	318.5±3	4.247±0.014	4.50±0.014
	max	42.391±0.10	40.89±0.10	3870±60	2271±21	21.72±0.11	21.73±0.11
9/10	min	4.007±0.014	3.420±0.015	854±6	316.7±3	4.784±0.014	5.21±0.013
	max	42.39 ±0.10	40.91±0.10	3383±47	2302±21	21.78±0.11	21.73±0.11

Table 5.2: Temperatures during each step of the repeated cooling cycle

Step 6/7 and 9/10 are given as one, because the peaks resulting from step 6 or 9 emerge when the program is already in step 7 or 10.

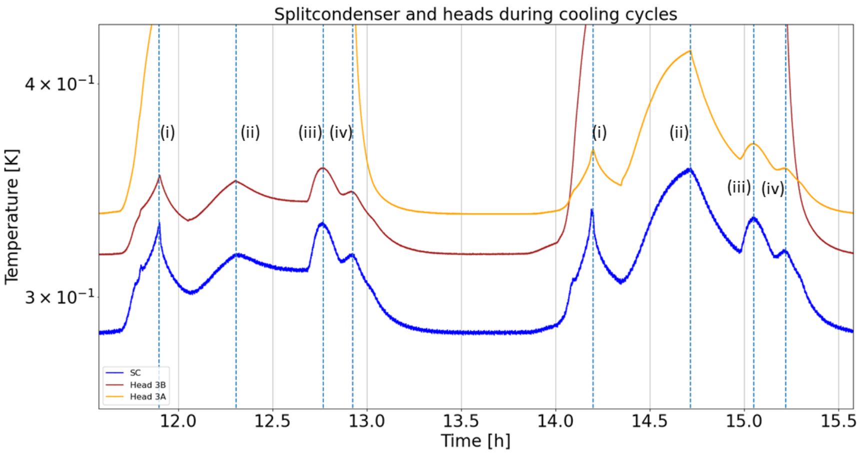

Figure 5.4: Split condenser peaks during recycling of the modules

Figure 5.4 shows the split condenser temperature during one complete cooling cycle. On the left (between 11.7h and 13.1h) module A was recycled while on the right (between 13.9h and 15.5h) module B was recycled. In total there are four different peaks, when recycling one module:

(i) disconnecting the helium-3 head of the module that's being recycled

(ii) heating the pump of the recycled module to 40K

(iii) cooling the helium-4 pump of the recycled module

(iv) cooling the helium-3 pump of the recycled module

Peak	recycled Module	Time [h]	SC Temp [mK]	ΔT [mK]
(i)	A	11.9	332	3
(ii)	A	12.3	318	3
(iii)	A	12.8	331	3
(iv)	A	12.9	317	3
(i)	B	14.2	338	3
(ii)	B	14.7	357	3
(iii)	B	15.0	334	3
(iv)	B	15.2	320	3

Table 5.3: split condenser peaks during recycling of each module

The highest peak, to about 360mK, is the one while heating the pumps of module B to 40K. The test run send by the manufacturer has its highest peak at about 370mK, when disconnecting the helium-3 head of module B from the split condenser. The manufacturers test had no peak, when heating the pumps. One possible explanation is, that the cables connected to the pump heaters get slightly warmer during this time transferring more heat to the cables, which are connected to the split condenser sensor.

5.3 Load test

During the operation of Lsym the heat load on the parts connected to the split condenser will be approximately $100\mu W$. This includes the thermal radiation and heating from electronics. To test if the fridge can stay below 500mK in the experiment the PID heater on the split condenser was used to imitate a load by applying a certain voltage. The voltage was 4.55V with an error of 0.01V which leads to a heating power of:

$$P = \frac{U^2}{R} = (98.2 \pm 0.3)\mu W, \tag{5.1}$$

where $R = (211.0 \pm 0.1)k\Omega$ is the resistance of the PID heater and U is the supplied voltage [30].

This voltage was applied over multiple cycles. The split condenser and helium-3 head temperatures during one of these can be seen in figure 5.5.

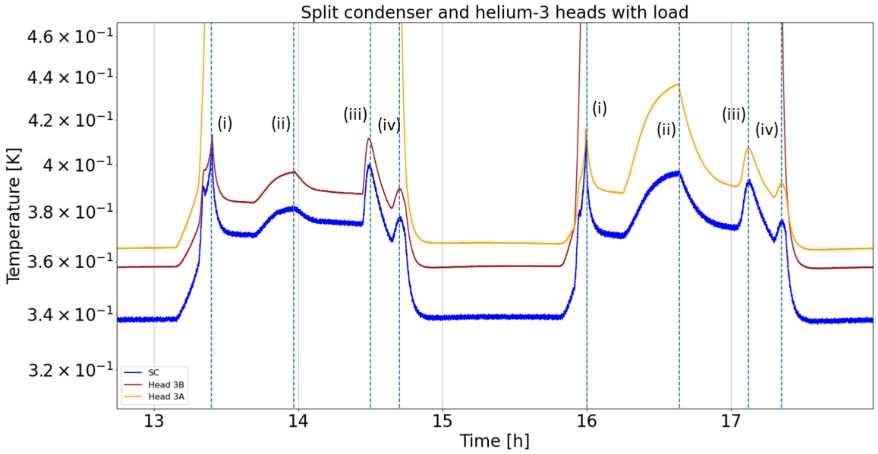

FIGURE 5.5: One complete cooling cycle with both modules being recycled with a load of 98.2 μW

As expected the minimum temperature as well as the individual peaks during the recycling of one module increased. Table 5.4 shows these temperatures where the peaks are defined as in table 5.3 and min is the minimum temperature reached, when both modules were cold.

Peak	recycled Module	Time [h]	SC Temp [mK]	ΔT [mK]
min		14.9-15.8	339	3
(i)	A	13.4	412	3
(ii)	A	14.0	381	3
(iii)	A	14.5	399	3
(iv)	A	14.7	377	3
(i)	B	16.0	415	3
(ii)	B	16.6	396	3
(iii)	B	17.1	393	3
(iv)	B	17.4	376	3

TABLE 5.4: Split condenser peaks and minimum temperature during recycling of each module with additional load

During the time of the load test the split condenser temperature was always below 500mK. This shows that once the cooler is in the experiment it will still be able to cool and maintain the LSYM cavity at a temperature less than 500mK and therefore the cyclotron motion can be cooled to the ground state. To test the cooling power as a function of applied load, multiple cycles were done. During each cycle a different load was applied and the average temperature reached, when both modules were cold, was taken. The cycle time was approximately 1 hour and the temperatures dependent on the applied load can be seen in figure 5.6.

The manufacturer also supplied a test similar to this. The main difference

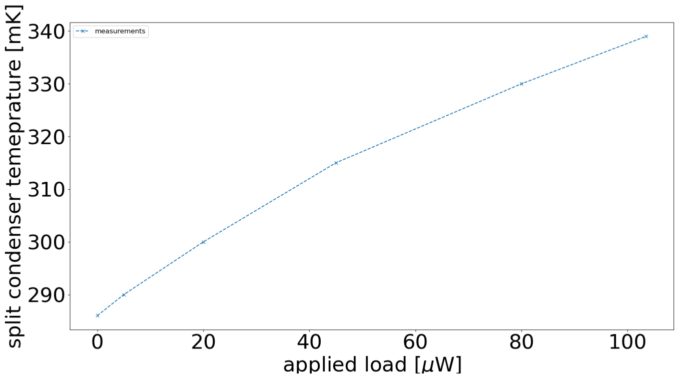

FIGURE 5.6: Split condenser temperatures during different applied loads

was, their test took the average temperature during one cycle with the applied load. This included the time where one of the modules was recycled. Additionally, their cycle time was set to approximately 2 hours. Due to this difference no comparison will be made.

6 Outlook and Conclusion

The LSYM experiment is an upcoming test of possible CPT violations in the lepton sector as introduced in chapter 1. It will store a positron and a bound electron simultaneously inside a Penning trap where the difference of their Larmor frequency will be measured. To reduce the uncertainty of the difference the cyclotron motion of the positron needs to be cooled to the ground state which is done by cooling the trap to at least 500mK. To achieve such temperatures a helium evaporation fridge will be used which is consists of two modules with one helium-4 and one helium-3 part each. Having two modules allows continuous cooling by recycling one module while the other one cools. The voltage output was realised with a DAC that was build by the in house electronics department. To automatise this cooling a control system was programmed in python which handles the voltages needed to heat certain parts of the cooler, while also saving the temperature and voltage data.
A test was set up to verify the working of the refrigerator and the control programs where the fridge was connected to a pulse tube cryocooler, surrounded by multiple radiation shields inside a vacuum chamber. In this test the split condenser reached a minimum of 287mK, with peaks during the recycling of one module of up to 360mK which is comparable to the test runs done by the manufacturer, Chase Cryogenic Research. Further cycle were used to test if the fridge can cope with the heat anticipated heat load of LSYM. There a minimum temperature of 338mK was achieved and the maximum peak was 415mK, showing the fridge will be able to cool and maintain the LSYM trap to less than 500mK.

Before the fridge can be integrated into the experiment an additional run with both modules needs to be done where a PID control is tested which will be used to stabilise the temperature of the split condenser.

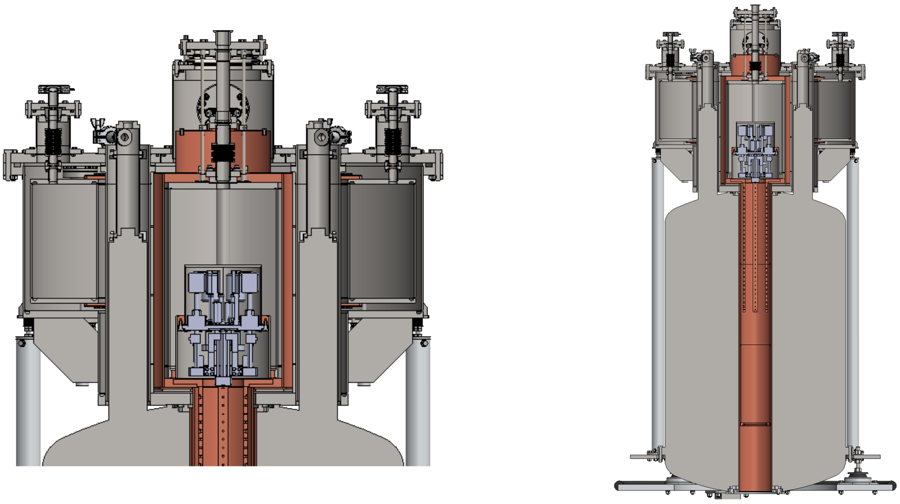

Figure 6.1: Sketch of the fridge inside the LSYM experiment

Figure 6.1 shows how the refrigerator will be placed inside the LSYM experiment. The "main plate" will be physically connected to the 4K stage, which is cooled with liquid helium. From there the cable which connects to the DAC and the Lakeshore Model 224 will be going to the next stage which is connected the 77K stage and is cooled with liquid nitrogen. For this recabling of the present refrigerator setup is necessary. The trap will be located well below the fridge and will be connected to the split condenser with heat stripes.

Once the refrigerator is in the experiment a couple of tests and a few changes have to be made, which are ensuring the cycle time is set correctly so stable temperatures can be achieved at the split condenser and update the PID parameters which will be slightly different due to the additional load.

A DAC Control

```python
#from ..src.self.dac.spi_LTC2688 import Ltc2688
import sqlite3
from lakeshore import Model224
import time
from timeloop import Timeloop
from datetime import timedelta
from options import *
from src.dac import spi_LTC2688
import numpy as np

class DAC_CONTROL():
    def __init__(self) -> None:
        "initialize Model224 , Dac and database"
        self.Temp_monitor = Model224(ip_address="149.217.89.108")
        self.dac = spi_LTC2688.Ltc2688(clock_pin=11, mosi_pin=10, miso_pin=9, select_pin=8)
        He_4_A_pump_channel = 0
        He_3_A_pump_channel = 1
        He_4_B_pump_channel = 2
        He_3_B_pump_channel = 3

        He_4_A_switch_channel = 4
        He_3_A_switch_channel = 5
        He_4_B_switch_channel = 6
        He_3_B_switch_channel = 7

        PID_Heater_channel = 8

        "set the channel output voltage range"
        self.dac.set_ch_range(He_4_A_pump_channel ,0) #Pump heaters need up to 30 Volts
        self.dac.set_ch_range(He_3_A_pump_channel ,0) #the output of 0 is up to 5 Volts but the output gets amplified
        self.dac.set_ch_range(He_4_B_pump_channel ,0) #with a factor of 6
        self.dac.set_ch_range(He_3_B_pump_channel ,0)
```

```
        self.dac.set_ch_range(He_4_A_switch_channel, 0) # switch heater need up to 5 Volts
        self.dac.set_ch_range(He_3_A_switch_channel, 0)
        self.dac.set_ch_range(He_4_B_switch_channel, 0)
        self.dac.set_ch_range(He_3_B_switch_channel, 0)
        self.dac.set_ch_range(PID_Heater_channel, 1) #PID heater takes up to 10 Volts

        self.starting_time = time.time()
        con = sqlite3.connect("fridge_database.db")
        cur = con.cursor()
        cur.execute("CREATE TABLE IF NOT EXISTS fridge (time, step, SC_Temp, MP_Temp, Pump_4A_Temp, Pump_3A_Temp, Pump_4B_Temp, Pump_3B_Temp, Head_4A_Temp, Head_3A_Temp, Head_4B_Temp, Head_3B_Temp, Switch_4B_Temp, Switch_3B_Temp, Pump_4A_Voltage, Pump_3A_Voltage, Pump_4B_Voltage, Pump_3B_Voltage, Switch_4A_Voltage, Switch_3A_Voltage, Switch_4B_Voltage, Switch_3B_Voltage, PID_Voltage)")
        cur.execute("""INSERT INTO fridge (time, SC_Temp, MP_Temp, Pump_4A_Temp, Pump_3A_Temp, Pump_4B_Temp, Pump_3B_Temp, Head_4A_Temp, Head_3A_Temp, Head_4B_Temp, Head_3B_Temp,Switch_4B_Temp,Switch_3B_Temp , Pump_4A_Voltage, Pump_3A_Voltage, Pump_4B_Voltage, Pump_3B_Voltage, Switch_4A_Voltage, Switch_3A_Voltage, Switch_4B_Voltage, Switch_3B_Voltage,PID_Voltage)
                             VALUES (0,0,0,0,0,0,0,0,0,0,0,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.)""")
        con.commit()

    def set_Voltage(self, channel, voltage):
        if(type(voltage) == (int or float)):
            self.dac.set_ch_conv(channel, voltage)
        elif (type(voltage) != (int or float)):
            print('no voltage set')
            print(type(voltage))
            self.dac.set_ch_conv(channel, 0)

    def run(self):
        tl = Timeloop()

        @tl.job(interval= timedelta(seconds=timestep))
        def write_Temperature_and_DAC():
            "update the timer and read the temperatures from the Lakeshore"
```

```
            timer = time.time()-self.starting_time
            SC_Temp = self.Temp_monitor.get_kelvin_reading(SC_Thermometer_channel)
            MP_Temp = self.Temp_monitor.get_kelvin_reading(Mainplate_Diode_channel)
            Pump_4A_Temp = self.Temp_monitor.get_kelvin_reading(He_4_A_Pump_Diode_channel)
            Pump_3A_Temp = self.Temp_monitor.get_kelvin_reading(He_3_A_Pump_Diode_channel)
            Pump_4B_Temp = self.Temp_monitor.get_kelvin_reading(He_4_B_Pump_Diode_channel)
            Pump_3B_Temp = self.Temp_monitor.get_kelvin_reading(He_3_B_Pump_Diode_channel)
            Head_4A_Temp = self.Temp_monitor.get_kelvin_reading(He_4_A_Thermometer_channel)
            Head_3A_Temp = self.Temp_monitor.get_kelvin_reading(He_3_A_Thermometer_channel)
            Head_4B_Temp = self.Temp_monitor.get_kelvin_reading(He_4_B_Thermometer_channel)
            Head_3B_Temp = self.Temp_monitor.get_kelvin_reading(He_3_B_Thermometer_channel)
            Switch_4B_Temp = self.Temp_monitor.get_kelvin_reading(He_4_A_Switch_Diode_channel)
            Switch_3B_Temp = self.Temp_monitor.get_kelvin_reading(He_3_A_Switch_Diode_channel)
            "connect the database and write values into it"
            con = sqlite3.connect("fridge_database.db")
            cur = con.cursor()
            Pump_4A_Voltage = cur.execute("SELECT Pump_4A_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            Pump_3A_Voltage = cur.execute("SELECT Pump_3A_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            Pump_4B_Voltage = cur.execute("SELECT Pump_4B_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            Pump_3B_Voltage = cur.execute("SELECT Pump_3B_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            Switch_4A_Voltage = cur.execute("SELECT Switch_4A_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
```

```
            Switch_3A_Voltage = cur.execute("SELECT Switch_3A_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            Switch_4B_Voltage = cur.execute("SELECT Switch_4B_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            Switch_3B_Voltage = cur.execute("SELECT Switch_3B_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]
            PID_Voltage = cur.execute("SELECT PID_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge))").fetchone()[0]

            cur.execute("""INSERT INTO fridge (time, SC_Temp, MP_Temp, Pump_4A_Temp, Pump_3A_Temp, Pump_4B_Temp, Pump_3B_Temp, Head_4A_Temp, Head_3A_Temp,  Head_4B_Temp, Head_3B_Temp,Switch_4B_Temp, Switch_3B_Temp,Pump_4A_Voltage, Pump_3A_Voltage, Pump_4B_Voltage, Pump_3B_Voltage, Switch_4A_Voltage, Switch_3A_Voltage, Switch_4B_Voltage, Switch_3B_Voltage ,PID_Voltage)
                        VALUES ({0},{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11},{12},0.,0.,0.,0.,0.,0
""".format(timer,SC_Temp, MP_Temp, Pump_4A_Temp, Pump_3A_Temp, Pump_4B_Temp, Pump_3B_Temp, Head_4A_Temp,  Head_3A_Temp, Head_4B_Temp, Head_3B_Temp, Switch_4B_Temp, Switch_3B_Temp))
            con.commit()


            "write voltages from database to Dac"
            self.dac.set_ch_conv(He_4_A_pump_channel, Pump_4A_Voltage/6, update = True)
            "Here it is important to divide by 6 because the Voltages of the pumps"
            self.dac.set_ch_conv(He_3_A_pump_channel, Pump_3A_Voltage/6, update = True)
            "get amplified by 6"
            self.dac.set_ch_conv(He_4_B_pump_channel, Pump_4B_Voltage/6, update = True)
            self.dac.set_ch_conv(He_3_B_pump_channel, Pump_3B_Voltage/6, update = True)
            self.dac.set_ch_conv(He_4_A_switch_channel, Switch_4A_Voltage, update = True)
            self.dac.set_ch_conv(He_3_A_switch_channel, Switch_3A_Voltage, update = True)
```

```
            self.dac.set_ch_conv(He_4_B_switch_channel, Switch_4B_Voltage, update = True)
            self.dac.set_ch_conv(He_3_B_switch_channel, Switch_3B_Voltage, update = True)
            self.dac.set_ch_conv(PID_Heater_channel, PID_Voltage, update = True)

        def DAC_turn_off():
            self.dac.set_ch_conv(He_4_A_pump_channel, 0, update = True)
            self.dac.set_ch_conv(He_3_A_pump_channel, 0, update = True)
            self.dac.set_ch_conv(He_4_B_pump_channel, 0, update = True)
            self.dac.set_ch_conv(He_3_B_pump_channel, 0, update = True)
            self.dac.set_ch_conv(He_4_A_switch_channel, 0, update = True)
            self.dac.set_ch_conv(He_3_A_switch_channel, 0, update = True)
            self.dac.set_ch_conv(He_4_B_switch_channel, 0, update = True)
            self.dac.set_ch_conv(He_3_B_switch_channel, 0, update = True)
            self.dac.set_ch_conv(PID_Heater_channel,0, update = True)

        tl.start()
        while(True):
            try:
                time.sleep(1)
            except KeyboardInterrupt:
                tl.stop()
                DAC_turn_off()
                print("Dac turned off")
                break

if __name__ == "__main__":
    Dac_Control = DAC_CONTROL()
    Dac_Control.run()
```

B Fridge Control

```python
import sqlite3
from datetime import timedelta
from timeloop import Timeloop
from options import *
from options import timestep
import time
from time import sleep
import numpy as np

class coolercontrol():
    tl = Timeloop()

    def __init__(self):
        "set the start step to 0 "
        self.step = 4
        self.max_pump_temp = 40

        "save voltages in variables that get written to the database"
        self.Pump_4A_Voltage = 0
        self.Pump_3A_Voltage = 0
        self.Pump_4B_Voltage = 0
        self.Pump_3B_Voltage = 0
        self.Switch_4A_Voltage = 0
        self.Switch_3A_Voltage = 0
        self.Switch_4B_Voltage = 0
        self.Switch_3B_Voltage = 0

        "save temperatures in variables that get read from the database"
        self.Pump_4A_Temp = 0
        self.Pump_3A_Temp = 0
        self.Pump_4B_Temp = 0
        self.Pump_3B_Temp = 0
        self.Head_4A_Temp = 0
        self.Head_3A_Temp = 0
        self.Head_4B_Temp = 0
```

```

        self.Head_3B_Temp = 0
        self.Switch_4A_Temp = 0
        self.Switch_3A_Temp = 0
        self.MP_Temp = 0
        self.SC_Temp = 0



    def step_0(self):
        """wait until mainplate and heat switch diodes cooled down to about 4K"""
        while(self.Switch_4A_Temp == (0 or None) or self.Switch_3A_Temp == (0 or None) or self.MP_Temp == (0 or None)):
            sleep(2)
        while(self.MP_Temp > 5 or self.MP_Temp == 0):
            sleep(2)
        self.step = 1

    def step_1(self):
        """Heat pumps to 40-45K and wait until all 4 heads are colder than 4k"""
        timer_pump = time.time()
        self.Pump_4A_Voltage = 2.
        self.Pump_3A_Voltage = 2.
        self.Pump_4B_Voltage = 2.
        self.Pump_3B_Voltage = 2.
        while(time.time()-timer_pump < 0):
            sleep(2)

        self.Pump_4A_Voltage = 11
        self.Pump_3A_Voltage = 8
        self.Pump_4B_Voltage = 12
        self.Pump_3B_Voltage = 8
        while(self.Pump_4A_Temp < self.max_pump_temp or
              self.Pump_3A_Temp < self.max_pump_temp or
              self.Pump_4B_Temp < self.max_pump_temp or
              self.Pump_3B_Temp < self.max_pump_temp
              ):
            if(self.Pump_4A_Temp > self.max_pump_temp):
                self.Pump_4A_Voltage = 2.5
            if(self.Pump_3A_Temp > self.max_pump_temp):
                self.Pump_3A_Voltage = 3.
            if(self.Pump_4B_Temp > (self.max_pump_temp-1)):
```

```
            self.Pump_4B_Voltage = 0
        if(self.Pump_3B_Temp > self.max_pump_temp):
            self.Pump_3B_Voltage = 3.
        sleep(1)
    self.step = 2

 def step_2(self):
    """wait until head temperature stabilizes"""
    timer_stabalize = time.time()
    self.Pump_4A_Voltage = 2.5
    self.Pump_3A_Voltage = 4.5
    self.Pump_4B_Voltage = 0
    self.Pump_3B_Voltage = 3.5
    while((time.time()- timer_stabalize) < 900 or (self.Head_4A_Temp > 3.8 or self.Head_4B_Temp > 3.8)):
        sleep(2)
    self.step = 3

 def step_3(self):
    """turn on 4He switches and wait until both 3 Heads cooled to about 1.5 K"""
    self.Pump_4A_Voltage = 0
    self.Pump_3A_Voltage = 5
    self.Pump_4B_Voltage = 0
    self.Pump_3B_Voltage = 4.5

    self.Switch_4A_Voltage = 2.2
    "2.2"
    self.Switch_4B_Voltage = 1.8
    "1.8"
    sleep(0)
    while(self.Switch_4A_Voltage < 3.1 or self.Switch_4B_Voltage < 2.7):
        self.Switch_4A_Voltage += 0.1
        self.Switch_4B_Voltage += 0.1
        sleep(120)
    self.Switch_4A_Voltage = 4
    self.Switch_4B_Voltage = 4
    while(self.Head_3A_Temp > 1.3 or self.Head_3B_Temp > 1.3):
        sleep(2)
    self.step = 4

 def step_4(self):
    """turn off 3He pumps, turn on both 3He switches """
```

```
        self.Switch_4B_Voltage = 4
        self.Switch_4A_Voltage = 4

        self.Pump_3A_Voltage = 0
        self.Pump_3B_Voltage = 0
        self.Switch_3A_Voltage = 2.6
        "2.6"
        self.Switch_3B_Voltage = 1.7
        "1.8"
        while(self.Switch_3A_Voltage < 3.2 and self.Switch_3B_Voltage < 2.3):
            self.Switch_3A_Voltage += 0.1
            self.Switch_3B_Voltage += 0.1
            sleep(120)
        self.Switch_3A_Voltage = 4
        self.Switch_3B_Voltage = 3
        while(self.Head_3A_Temp > 0.37 or self.Head_3B_Temp > 0.36):
            sleep(2)

        sleep(3600)
        self.step = 5

    def step_5(self):
        """slowly increase 4 pump and then 3 pump temperature of module A"""
        self.Switch_4B_Voltage = 4
        self.Switch_3B_Voltage = 3
        self.Switch_4A_Voltage = 0
        self.Switch_3A_Voltage = 0
        sleep(200)

        self.Pump_4A_Voltage = 2
        sleep(400)
        self.Pump_3A_Voltage = 2

        while(self.Pump_3A_Temp < 17):
          sleep(2)
        sleep(200)
        self.Pump_4A_Voltage = 12
        self.Pump_3A_Voltage = 6.5

        while(self.Pump_4A_Temp < self.max_pump_temp or
              self.Pump_3A_Temp < self.max_pump_temp
              ):
            if(self.Pump_4A_Temp > self.max_pump_temp):
```

```
                self.Pump_4A_Voltage = 3
            if(self.Pump_3A_Temp > self.max_pump_temp):
                self.Pump_3A_Voltage = 4.5
            sleep(2)
        timer_pump = time.time()
        self.Pump_4A_Voltage = 3
        self.Pump_3A_Voltage = 4.5

        while(self.Head_4A_Temp > 4.2 or ((time.time()-
   timer_pump) < 900)):
            sleep(2)
        self.step = 6

    def step_6(self):
        """cycle 4 head of module A"""
        self.Switch_4B_Voltage = 4
        self.Switch_3B_Voltage = 3
        self.Pump_3A_Voltage = 4

        self.Pump_4A_Voltage = 0
        self.Switch_4A_Voltage = 3.5
        while(self.Switch_4A_Voltage < 3.5):
            self.Switch_4A_Voltage += 0.1
            sleep(120)
        self.Switch_4A_Voltage = 4

        self.step = 7

    def step_7(self):
        """cycle 3 head of module A"""
        self.Switch_4B_Voltage = 4
        self.Switch_3B_Voltage = 3
        self.Switch_4A_Voltage = 4

        while(self.Head_3A_Temp > 1.3):
          sleep(2)
        self.Pump_3A_Voltage = 0
        self.Switch_3A_Voltage = 3
        while(self.Switch_3A_Voltage < 3.6):
            self.Switch_3A_Voltage += 0.1
            sleep(80)
        self.Switch_3A_Voltage = 4

        """while(self.SC_Temp < 0.4):
            sleep(2)"""

```

```
     """if(self.SC_Temp >= 0.4):
       self.Switch_4B_Voltage = 0
       self.Switch_3B_Voltage = 0
       self.Switch_4A_Voltage = 0
       self.Switch_3A_Voltage = 0
       self.step = -1"""

     sleep(3600)
     self.step = 8

 def step_8(self):
     """slowly heat 4 and 3 pump of moduel B to 45K"""
     self.Switch_4A_Voltage = 4
     self.Switch_3A_Voltage = 3.5
     self.Switch_4B_Voltage = 0
     self.Switch_3B_Voltage = 0
     sleep(200)
     self.Pump_4B_Voltage = 2.2
     sleep(400)
     self.Pump_3B_Voltage = 2

     while(self.Pump_3B_Temp < 17.5):
         sleep(2)
     sleep(300)
     self.Pump_4B_Voltage = 12
     self.Pump_3B_Voltage = 6
     while(self.Pump_4B_Temp < (self.max_pump_temp-1)
or
           self.Pump_3B_Temp < self.max_pump_temp
           ):
         if(self.Pump_4B_Temp > self.max_pump_temp):
             self.Pump_4B_Voltage = 0
         if(self.Pump_3B_Temp > self.max_pump_temp):
             self.Pump_3B_Voltage = 3.5
         sleep(2)

     self.Pump_4B_Voltage = 0
     self.Pump_3B_Voltage = 3.3
     timer_pump=time.time()
     while(self.Head_4B_Temp > 4.2 or ((time.time()-
timer_pump) < 900)):
         sleep(2)

     self.step = 9

 def step_9(self):
```

```
        self.Pump_3B_Voltage = 3.3
        self.Switch_4A_Voltage = 4.0
        self.Switch_3A_Voltage = 3.5

        self.Pump_4B_Voltage = 0
        self.Switch_4B_Voltage = 1.8
        while(self.Switch_4B_Voltage < 2.3):
            self.Switch_4B_Voltage += 0.1
            sleep(120)
        self.Switch_4B_Voltage = 4
        self.step = 10

    def step_10(self):
        self.Switch_4A_Voltage = 4
        self.Switch_3A_Voltage = 3.5
        self.Switch_4B_Voltage = 4

        while(self.Head_3B_Temp > 1.3):
            sleep(2)

        self.Pump_3B_Voltage = 0
        self.Switch_3B_Voltage = 1.8
        "1.8"
        while(self.Switch_3B_Voltage < 2.2):
            self.Switch_3B_Voltage += 0.1
            sleep(120)
        self.Switch_3B_Voltage = 3
        while(self.Head_3B_Temp > 0.36 and self.Head_3A_Temp < 0.38):
            sleep(1)
        sleep(3600)
        self.step = 5


    "writes 0 to all voltages in the database "
    def voltage_turn_off(self):
        con =sqlite3.connect(’fridge_database.db’)
        cur = con.cursor()
        cur.execute("UPDATE fridge SET(step, Pump_4A_Voltage, Pump_3A_Voltage, Pump_4B_Voltage, Pump_3B_Voltage, Switch_4A_Voltage, Switch_3A_Voltage, Switch_4B_Voltage, Switch_3B_Voltage) =(-1,0,0,0,0,0,0,0,0) WHERE(rowid = (SELECT MAX(rowid) FROM fridge))""")
        con.commit()

```

```
    def run(self):
        tl = Timeloop()

        @tl.job(interval = timedelta(seconds = timestep))
        def routine():
            """
            -read the temperatures from the database
            -write the Voltages into the database
            """
            con =sqlite3.connect('fridge_database.db')
            cur = con.cursor()
            self.Pump_4A_Temp, self.Pump_3A_Temp, self.
   Pump_4B_Temp, self.Pump_3B_Temp, self.Head_4A_Temp,
   self.Head_3A_Temp, self.Head_4B_Temp, self.Head_3B_Temp
   , self.Switch_4B_Temp, self.Switch_3B_Temp, self.
   MP_Temp, self.SC_Temp = cur.execute("SELECT
   Pump_4A_Temp, Pump_3A_Temp, Pump_4B_Temp, Pump_3B_Temp,
    Head_4A_Temp, Head_3A_Temp,  Head_4B_Temp,
   Head_3B_Temp,Switch_4B_Temp, Switch_3B_Temp, MP_Temp,
   SC_Temp FROM fridge WHERE (rowid = (SELECT MAX(rowid)
   FROM fridge))").fetchall()[0]
            cur.execute("UPDATE fridge SET(step,
   Pump_4A_Voltage, Pump_3A_Voltage, Pump_4B_Voltage,
   Pump_3B_Voltage, Switch_4A_Voltage, Switch_3A_Voltage,
   Switch_4B_Voltage, Switch_3B_Voltage)
   =({0},{1},{2},{3},{4},{5},{6},{7},{8}) WHERE(rowid = (
   SELECT MAX(rowid) FROM fridge))""".format(self.step,
   self.Pump_4A_Voltage, self.Pump_3A_Voltage, self.
   Pump_4B_Voltage, self.Pump_3B_Voltage, self.
   Switch_4A_Voltage, self.Switch_3A_Voltage, self.
   Switch_4B_Voltage, self.Switch_3B_Voltage))
            con.commit()

        self.running = True
        tl.start()
        while(self.running):
            try:
                if(self.step == 0):
                    self.step_0()
                elif(self.step ==1):
                    self.step_1()
                elif(self.step == 2):
                    self.step_2()
                elif(self.step == 3):
                    self.step_3()
                elif(self.step == 4):
```

```
                    self.step_4()

                elif(self.step ==5):
                    self.step_5()
                elif(self.step == 6):
                    self.step_6()
                elif(self.step == 7):
                    self.step_7()
                elif(self.step == 8):
                    self.step_8()
                elif(self.step ==9):
                    self.step_9()
                elif(self.step == 10):
                    self.step_10()
            except KeyboardInterrupt:
                self.running = False
                self.step = -1
                tl.stop()
                self.voltage_turn_off()
                break


if __name__ == "__main__":
    Coolercontrol = coolercontrol()
    Coolercontrol.run()
```

C Fridge GUI

```
import sqlite3
import time
from timeloop import Timeloop
from datetime import timedelta
from options import *
from tkinter import *
from tkinter import ttk

class Fridge_Gui:
    def __init__(self) -> None:
        self.SC_Temp = 0
        self.Pump_4A_Voltage = 0
        self.Pump_3A_Voltage = 0
        self.Pump_4B_Voltage = 0
        self.Pump_3B_Voltage = 0
        self.Switch_4A_Voltage = 0
        self.Switch_3A_Voltage = 0
        self.Switch_4B_Voltage = 0
        self.Switch_3B_Voltage = 0
        self.PID_Voltage = 0
        self.time = 0

    def run(self):
        tl = Timeloop()

        "create window with tkinter"
        self.root = Tk()

        self.Temp_frm = ttk.Frame(self.root, padding=10)#, style='red')
        self.Temp_frm.grid()
        self.Temp_frm.anchor("center")
        self.Volt_frm = ttk.Frame(self.root, padding=10)#, style='red')
        self.Volt_frm.grid()
        self.Volt_frm.anchor("center")
        self.root.title('Fridge GUI')
```

```

        "define how to handle closing the window"
        def close_Window():
            tl.stop()
            self.root.quit()
            self.root.destroy()
            quit()

        self.root.protocol("WM_DELETE_WINDOW",
close_Window)



        @tl.job(interval = timedelta(seconds=2))
        def update_window():
            "read data from database"
            con = sqlite3.connect("fridge_database.db")
            cur = con.cursor()
            self.SC_Temp = round(1e3*cur.execute("SELECT
SC_Temp FROM fridge WHERE (rowid = (SELECT MAX(rowid)
FROM fridge))").fetchone()[0],2)

            self.PID_Voltage = round(cur.execute("SELECT
PID_Voltage FROM fridge WHERE (rowid = (SELECT MAX(
rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Pump_4A_Voltage = round(cur.execute("
SELECT Pump_4A_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Pump_3A_Voltage = round(cur.execute("
SELECT Pump_3A_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Pump_4B_Voltage = round(cur.execute("
SELECT Pump_4B_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Pump_3B_Voltage = round(cur.execute("
SELECT Pump_3B_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Switch_4A_Voltage = round(cur.execute("
SELECT Switch_4A_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Switch_3A_Voltage = round(cur.execute("
SELECT Switch_3A_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
            self.Switch_4B_Voltage = round(cur.execute("
SELECT Switch_4B_Voltage FROM fridge WHERE (rowid = (
SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
```

```python
        self.Switch_3B_Voltage = round(cur.execute("SELECT Switch_3B_Voltage FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0],2)
        self.time = int(cur.execute("SELECT time FROM fridge WHERE (rowid = (SELECT MAX(rowid) FROM fridge)-1)").fetchone()[0])
        con.commit()

        if(self.SC_Temp < 365 and self.SC_Temp > 355):
            bgColor = "green"
        else:
            bgColor = "red"

        frame_list = self.root.grid_slaves()
        label_list = []
        for fr in frame_list:
            label_list = fr.grid_slaves()
            for l in label_list:
                l.destroy()


        "SC Temperature"
        SCLabel1 = ttk.Label(self.Temp_frm, text ="SC Temperature:", background = bgColor,font='Arial 30').grid(column=0,row=0)
        SCLabel2 = ttk.Label(self.Temp_frm, text="mK", background = bgColor,font='Arial 50').grid(column=1,row=1)
        SCLabel3 = ttk.Label(self.Temp_frm, text="\t{0}\t".format(self.SC_Temp), background = bgColor, font='Arial 50').grid(column=0,row=1)
        "Show Voltages in GUI"
        "Pumps"
        PumpLabel1 = ttk.Label(self.Volt_frm, text ="4 A Pump  ",font='Arial 15').grid(column=0,row=2)
        PumpLabel2 = ttk.Label(self.Volt_frm, text="{0}".format(self.Pump_4A_Voltage),  font='Arial 15').grid(column=0,row=3)
        PumpLabel3 = ttk.Label(self.Volt_frm, text ="3 A Pump  ", font='Arial 15').grid(column=1,row=2)
        PumpLabel4 = ttk.Label(self.Volt_frm, text="{0}".format(self.Pump_3A_Voltage),  font='Arial 15').grid(column=1,row=3)
        PumpLabel5 = ttk.Label(self.Volt_frm, text ="4 B Pump  ", font='Arial 15').grid(column=2,row=2)
```

```
        PumpLabel6 = ttk.Label(self.Volt_frm, text="{0}".format(self.Pump_4B_Voltage),  font='Arial 15').grid(column=2,row=3)
        PumpLabel7 = ttk.Label(self.Volt_frm, text ="3B Pump  ", font='Arial 15').grid(column=3,row=2)
        PumpLabel8 = ttk.Label(self.Volt_frm, text="{0}".format(self.Pump_3B_Voltage),  font='Arial 15').grid(column=3,row=3)

        "switches"
        SwitchLabel1 = ttk.Label(self.Volt_frm, text = "4A Switch ", font='Arial 15').grid(column=4,row=2)
        SwitchLabel2 = ttk.Label(self.Volt_frm, text="{0}".format(self.Switch_4A_Voltage),  font='Arial 15').grid(column=4,row=3)
        SwitchLabel3 = ttk.Label(self.Volt_frm, text = "3A Switch ", font='Arial 15').grid(column=5,row=2)
        SwitchLabel4 = ttk.Label(self.Volt_frm, text="{0}".format(self.Switch_3A_Voltage),  font='Arial 15').grid(column=5,row=3)
        SwitchLabel5 = ttk.Label(self.Volt_frm, text = "4B Switch ", font='Arial 15').grid(column=6,row=2)
        SwitchLabel6 = ttk.Label(self.Volt_frm, text="{0}".format(self.Switch_4B_Voltage),  font='Arial 15').grid(column=6,row=3)
        SwitchLabel7 = ttk.Label(self.Volt_frm, text = "3B Switch ", font='Arial 15').grid(column=7,row=2)
        SwitchLabel8 = ttk.Label(self.Volt_frm, text="{0}".format(self.Switch_3B_Voltage),  font='Arial 15').grid(column=7,row=3)

        "time and PID"
        timeLabel1 = ttk.Label(self.Volt_frm, text ="Time: ", font='Arial 15').grid(column=4,row=4)
        timeLabel1 = ttk.Label(self.Volt_frm, text="{0}".format(self.time),background="white",  font='Arial 15').grid(column=4,row=5)
        PIDLabel1 = ttk.Label(self.Volt_frm, text ="PID:  ",font='Arial 15').grid(column=3,row=4)
        PIDLabel2 = ttk.Label(self.Volt_frm, text="{0}".format(self.PID_Voltage),  font='Arial 15').grid(column=3,row=5)


    tl.start()
    while True:
```

```python
            try:
                self.root.mainloop()
            except KeyboardInterrupt:
                self.root.quit()
                tl.stop()
                break


if __name__ == "__main__":
    GUI = Fridge_Gui()
    GUI.run()
```

Bibliography

[1] T. Cheng, M. Lindner, and M. Sen, “Implications of a matter-antimatter mass asymmetry in penning-trap experiments,” *Physics Letters B*, vol. 844, p. 138 068, 2023.

[2] Y. N. N. K. Blaum and G. Werth, “Penning traps as a versatile tool for precise experiments in fundamental physics,” *Contemporary Physics*, vol. 51, no. 2, pp. 149–175, 2010.

[3] D. Beck, K. Blaum, G. Bollen, *et al.*, “Electric and magnetic field optimization procedure for penning trap mass spectrometers,” *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 598, no. 2, pp. 635–641, 2009.

[4] G. Gabrielse and F. C. MacKintosh, “Cylindrical penning traps with orthogonalized anharmonicity compensation,” *International Journal of Mass Spectrometry and Ion Processes*, vol. 57, no. 1, pp. 1–17, 1984.

[5] G. Gabrielse, “Relaxation calculation of the electrostatic properties of compensated penning traps with hyperbolic electrodes,” *Physical Review A*, vol. 27, no. 5, p. 2277, 1983.

[6] S. Sturm, I. Arapoglou, A. Egl, *et al.*, “The alphatrap experiment,” *The European Physical Journal Special Topics*, vol. 227, pp. 1425–1491, 2019.

[7] G. P. Fisher, “The thomas precession,” *American Journal of Physics*, vol. 40, no. 12, pp. 1772–1781, 1972.

[8] A. Czarnecki, M. Dowling, J. Piclum, and R. Szafron, “Two-loop binding corrections to the electron gyromagnetic factor,” *Physical Review Letters*, vol. 120, no. 4, p. 043 203, 2018.

[9] L. S. Brown and G. Gabrielse, “Geonium theory: Physics of a single electron or ion in a penning trap,” *Reviews of Modern Physics*, vol. 58, no. 1, p. 233, 1986.

[10] C. Genes, D. Vitali, P. Tombesi, S. Gigan, and M. Aspelmeyer, “Ground-state cooling of a micromechanical oscillator: Comparing cold damping and cavity-assisted cooling schemes,” *Phys. Rev. A*, vol. 77, p. 033 804, 3 2008.

[11] F. Marquardt, J. P. Chen, A. A. Clerk, and S. M. Girvin, “Quantum theory of cavity-assisted sideband cooling of mechanical motion,” *Phys. Rev. Lett.*, vol. 99, p. 093 902, 9 2007. DOI: `10.1103/PhysRevLett.99.093902`.

[12] J. D. Teufel, T. Donner, D. Li, *et al.*, “Sideband cooling of micromechanical motion to the quantum ground state,” *Nature*, vol. 475, no. 7356, pp. 359–363, 2011.

[13] E. A. Cornell, R. M. Weisskoff, K. R. Boyce, and D. E. Pritchard, “Mode coupling in a penning trap: π Pulses and a classical avoided crossing,” *Phys. Rev. A*, vol. 41, pp. 312–315, 1 1990.

[14] M. Wellons, “The stefan-boltzmann law,” *Physics Department, The College of Wooster, Wooster, Ohio*, vol. 44691, p. 25, 2007.

[15] O. M. Necati *et al.*, *Heat conduction*, 1993.

[16] P Bradley, R Radebaugh, and M Lewis, “Cryogenic material properties database, update 2006,” in *Proceedings of ICMC’06 Twenty First International Cryogenic Engineering Conference and 9th Cryogenics*, 2006, pp. 13–21.

[17] I. Langmuir, “Convection and conduction of heat in gases,” *Physical Review (Series I)*, vol. 34, no. 6, p. 401, 1912.

[18] A. Trowbridge, “Thermal conductivity of air at low pressures,” *Phys. Rev.*, vol. 2, pp. 58–64, 1 1913.

[19] *Product range - chase research cryogenic*, `https://www.chasecryogenics.com/products`, Accessed: 2024-04-29.

[20] S. Torquato and G Stell, “Latent heat of vaporization of a fluid,” *The Journal of Physical Chemistry*, vol. 85, no. 21, pp. 3029–3030, 1981.

[21] R. Bhatia, S. Chase, S. Edgington, *et al.*, “A three-stage helium sorption refrigerator for cooling of infrared detectors to 280 mk,” *Cryogenics*, vol. 40, no. 11, pp. 685–691, 2000.

[22] W. J. Shuttleworth, “Evaporation,” 1979.

[23] R. Erickson and L. Roberts, “The measurement and the calculation of the liquid helium vapor pressure-temperature scale from 1° to 4.2° k,” *Physical Review*, vol. 93, no. 5, p. 957, 1954.

[24] Y. Huang and G. Chen, “A practical vapor pressure equation for helium-3 from 0.01 k to the critical point,” *Cryogenics*, vol. 46, no. 12, pp. 833–839, 2006.

[25] *Continous 300mk sorption cooler type cc7: Generic installation and operating instructions*, Chase Research Cryogenics (2023).

[26] C. Enss and S. Hunklinger, *Low-temperature physics.* Springer Science & Business Media, 2005.

[27] N Sclar and D. Pollock, “On diode thermometers,” *Solid-State Electronics*, vol. 15, no. 5, pp. 473–480, 1972.

[28] Y. Lan, L. Yu, G. Chen, S. Yang, and A. Chang, “Construction and characterization of ntc thermistors at low temperature,” *International Journal of Thermophysics*, vol. 31, pp. 1456–1465, 2010.

[29] *User’s manual model 224 temperature monitor*, Lake Shore Cryotronics, Inc.(June 2020).

[30] S Scott Courts, W. Davenport, and D Scott Holmes, “Thermal resistances of cryogenic temperature sensors from 1–300 k,” in *Advances in cryogenic engineering*, Springer, 2000, pp. 1849–1856.

[31] *Rp-062b 4k pulse tube cryocooler series*, `https://www.shicryogenics.com/product/rp-062b-4k-pulse-tube-cryocooler-series/`, Accessed: 2024-07-15.

[32] V. Parma, “Cryostat design,” *arXiv preprint arXiv:1501.07154*, 2015.

[33] J. Frolec, T. Králík, V. Musilová, P. Hanzelka, A. Srnka, and J. Jelínek, “A database of metallic materials emissivities and absorptivities for cryogenics,” *Cryogenics*, vol. 97, pp. 85–99, 2019.

[34] S. I. Woods, T. M. Jung, D. R. Sears, and J. Yu, “Emissivity of silver and stainless steel from 80 k to 300 k: Application to iter thermal shields,” *Cryogenics*, vol. 60, pp. 44–48, 2014.

[35] G. E. McIntosh, “Layer by layer mli calculation using a separated mode equation,” in *Advances in cryogenic engineering*, Springer, 1994, pp. 1683–1690.

[36] *Beyond gravitiy - cryogenic insulation products*, Accessed: 2024-07-11.

[37] P. Duthil, “Material properties at low temperature,” *arXiv preprint arXiv:1501.07100*, 2015.

[38] *Cryogenics - property graphing/integrating tool*, `https://trc.nist.gov/cryogenics/calculators/graphcalc.html`, Accessed: 2024-04-29.

[39] *Sqlite3 homepage*, `https://www.sqlite.org/`, Accessed: 2024-06-25.

[40] *Tkinter — python interface to tcl/tk*, `https://docs.python.org/3/library/tkinter.html`, Accessed: 2024-06-25.

[41] *Sensor calibration accuarcies*, `https://www.lakeshore.com/docs/default-source/product-downloads/lstc_appendixd_l.pdf?sfvrsn=ef0ab655_2`, Accessed: 2024-06-18.

[42] T. J. Rivlin, *Chebyshev polynomials*. Courier Dover Publications, 2020.