university of groningen

faculty of science and engineering

The Impact of Helium Pressure on Ion-guiding Efficiency at the Low Energy Branch of NEXT

Author:
René BLOMMAERT
(s5738598)

Supervisor:
Prof. Dr. Julia EVEN
Second examiner :
Dr. Catherine RIGOLLET

Bachelor's Thesis
To fulfill the requirements for the degree of
Bachelor of Science in Physics
at the University of Groningen

July 4, 2024

Contents

Abstract

The goal of this research project is to work on the pressure control and monitoring of the NEXT ion-guide, as to determine and recommend an optimal pressure for the highest ion transmission and bunching efficiency. A communication system between the Leybold GRAPHIX 2 gauge controller and the Influxdb database was established by creating new Python drivers. Initial measurements were performed from a range of 10^{-4} mbar to 10^{-2} mbar, using an offline version of the set-up. Measuring the Time of Flight (ToF) and counts up until the first MicroChannel Plate detector (MCP). A full analysis of the different parameters suggests an optimal pressure at approximately 1.0×10^{-2} mbar, giving almost 5 counts/ejection and a Full Width Half Maximum (FWHM) of 135 ns.

Acknowledgments

I am grateful to the individuals who have supported me throughout my involvement in the NEXT Project. I extend my thanks to Professor Doctor Julia Even, my supervisor, who clarified the project's objectives, the set-up and has pushed me to think more deeply about the goal of physics. I appreciate Briain Hartigan for his close mentorship and consistent assistance. Additionally, I would like to acknowledge the other team members of the NEXT project, Jasper Westbroek, Jennifer Cipagauta Mora, Marko Brajkoviç, Niels Bouwman, Arif Soylu, Lisa van der Werff and Puck Planje, for their warm welcome and for making my experience while completing my bachelor thesis highly pleasant. I would also like to thank my parents for always supporting me.

This research was funded by European Research Council Executive Agency (ERCEA), under the powers delegated by the European Commission through a Starting Grant number 803740dNEXTdERC-2018-STG.

1 Introduction

1.1 NEXT Motivation

The region of neutron rich nuclei is among the remaining challenging areas to explore on the table of nuclides. The short lifetimes of these nuclei in the neutron drip line have posed difficulties in obtaining precise measurements. Some facilities worldwide, such as IGISOL in Finland [1], have developed methods to produce these heavy exotic nuclei. However, this production comes with its own set of challenges, including a wide angular distribution [2].

The Neutron-rich EXotic heavy nuclei in multi-nucleon transfer reactions Project (NEXT Project), initiated in 2018, aims to collect a large angular distribution and prepare heavy nuclei for precision measurements [3]. This project will enable access to isotopes that are difficult to produce at other facilities, allowing for precise mass measurements of these low-energy heavy nuclei. Obtaining information about the mass will grant insight into the binding energy of these nuclei, doing so with a precision mass resolving power of $\times 10^{5-6}$. Such measurements are crucial for a better understanding of the nuclear chart and will provide valuable data for future experiments. Mass data from NEXT will also allow for more insight into the astrophysical r-process which is responsible for creating approximately half of the elements heavier than iron [4].

The study of exotic nuclei is particularly important in the field of nuclear astrophysics, as these nuclei are found in extreme astrophysical environments. Although we are aware that heavy nuclei are produced following the catastrophic collapse of massive stars, these elements decay rapidly. Therefore, in order to better understand the formation of heavy matter and its origins, it is essential to explore the characteristics of exotic neutron-rich nuclei [5].

1.2 NEXT Set-up

Figure 1 shows the NEXT set-up, designed to produce, thermalize, bunch, measure mass, and prepare heavy neutron-rich nuclei samples for decay spectroscopy. The AGOR Cyclotron will direct a beam of ions to a rotating target inside the solenoid magnet, this will produce the multi-nucleon transfer reaction. The reaction ^{136}Xe + ^{198}Pt at an energy of about 6 MeV/amu will give access to the N=126 neutron sub-shell. Similarly ^{48}Ca + ^{251}Cf at 6.1 MeV/amu will be used to produce transfermium nuclei. The solenoid separator will serve as a way of dealing with this large angular distribution by allowing for the separation of target-like transfer products from the primary beam and projectile-like fragments, removing any unwanted by-products with the use of a magnetic field. Furthermore, linking the Multi-Reflection Time-of-Flight Mass Spectrometer (MR-ToF MS) via a gas-catcher [6] and a stacked-ring ion-guide [7] will prepare samples for decay analysis and allow for precise mass measurements of certain highly exotic isotopes [3]. The experiment is set up with a differential pumping system to maintain different pressures, as will be discussed in more detail in Chapter 3.

The main objective of the NEXT set-up is to obtain precise measurements with low production rates. To accomplish this the efficiency of the ion-guide must be optimized. The ion-guide functions with a Radio Frequency (RF) to confine the ions radially, and a travelling wave to carry the ions from the gas-catcher to the next part of the set-up. Maintaining low helium pressure will amplify the effects of the RF, resulting in cleaner bunching. Simultaneously inside the ion-guide pressure will minimize interactions between ions and favor thermalization with the helium buffer gas, equally limiting production of undesirable molecules inside the line. Helium is used due to its purity, high ionization energy, and light mass [3].

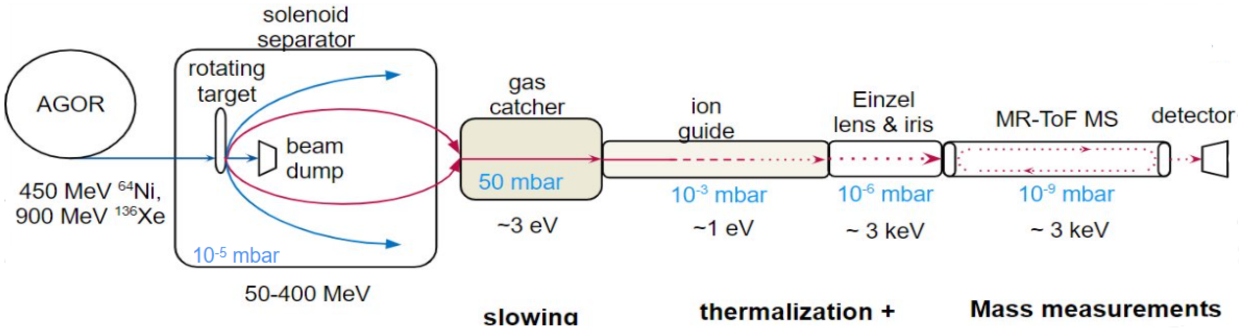

Figure 1: Schematic of NEXT, the blue arrow shows the trajectories of projectile-like fragments, while the red arrows show the trajectories of target-like fragments. The solid lines represent a continuous beam, while the dashed lines represent a bunched beam. The pressure requirements of each differential pumping stage and energy levels of the ions are also indicated in the figure.

1.3 Thesis Motivation

In this thesis the significance of pressure in the efficiency of the ion-guide in the NEXT project will be explored. Pressure plays a crucial role in the process of thermalizing ions, so determining the optimal pressure can have a compelling effect on the efficiency of the ion-guide and, as a result on the rest of the set-up.

Monitoring and logging pressure will allow for real-time analysis of results, while taking into account systematic errors with immediate data being available. Creating and deploying a tool to monitor pressure will also facilitate future enhancements to the set-up, as it will be designed to be versatile.

2 Computational Methods

2.1 Simulation Environment and Task Description

To start this chapter, we must first outline our objectives : the main goal is to develop a Python code capable of:

- Communicating with a database.
- Receiving information from a controller.
- Translating that information into the database protocol.
- Sending it to the database.

For this task Python 3.10.9, a Leybold GRAPHIX2 Controller, a RS-232 to USB cable and the Influxdb database were used. Understanding how the database and controller communicate is key to being able to develop the program correctly. Figure 3 gives an overview of the connection between the devices.

2.2 Developing and optimising a Python communication program

Some important factors when developing the code were for it to be readable and to keep it as simple as possible so that future modifications can be easily implemented. There exist Python libraries that allow for a connection between the Influxdb database and different hosts, unfortunately these libraries are not compatible with the Raspberry Pi OS we intend to use. Therefore, we need to find an alternative method, which involves using the HTTP protocol. HTTP, or Hypertext Transfer Protocol, the primary protocol for data transmission between a web browser and a website. To send information formatting the code correctly in HTTP and knowing how Influxdb expects to receive information is key. The Influxdb line protocol [8] is shown in figure 2.

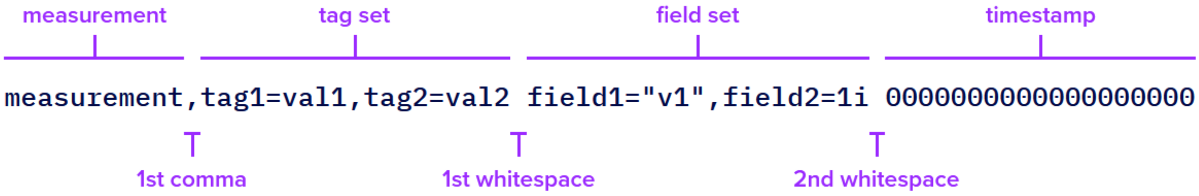
measurement
tag set
field set
timestamp
measurement,tag1=val1,tag2=val2 field1="v1",field2=1i 0000000000000000000
1st comma
1st whitespace
2nd whitespace

Figure 2: InfluxDB line protocol

Incorporating this line protocol into Python can be seen in listing 1, the header must include some necessary information such as:

- Authorization, token to establish connection with the database.
- Content-type, to make sure we correctly encode the data we send and receive.
- Accept, the json file formatting so the database can read our information.

Listing 1: influxdb line protocol in Python format

```python
# Database configuration
token = "
    uqosEoKFyYegI0keAIzkiDkpc8LYM6D4tCvENvwrWDg_xZHejeq8vuXFUHgYNkLNQw9kZ4DnTdMCaHJz2LseMw
    =="
url_post = "http://192.168.10.106:8086/api/v2/write?org=NextLab&bucket=Test&
    precision=ns"
measurement = "vacuum"
tags = {"device": "graphix2", "gauge": "pressure"}

headers = {
    "Authorization": "Token " + token,
    "Content-Type": "text/plain; charset=utf-8",
    "Accept": "application/json"
}
```

The URL-link, measurement and tags are all used in the post request function of the code. This is how the Influxdb database requires the information to be formatted.

This protocol successfully established a baseline connection between the database and the Raspberry Pi; moving forward it was much easier to add different aspects to the code, such as separating the data points and ensuring the timestamps match. Having correct timestamps is an important factor as having a delay or non-matching time with the other datasets would cause confusion and require additional analysis.

The next part is establishing a connection between the gauge controller and the Raspberry Pi. A complete overview of the gauge controller's manual was required, as to fully understand how the controller communicates [9]. The encoding to request data from the controller is done as follows :

- Read detection [0x0F]
- Parameter group [1,2,3,4,5]
- Separating character [";"]
- Parameter number [1-38]
- Checksum calculation [CRC]
- End of transmission [0x04]

Parameter group values one to three indicate the channel which is to be used, and the parameter number the data requested, for pressure this is n=29. The checksum calculation consists of an ASCII character, the byte value of which results as follows from the preceding characters of the send or receive string : CRC = 255 – [(Byte sum of all preceding characters) mod 256] If this value is lower than 32 (control character of the ASCII code), then 32 must be added [9]. If there are any errors either in the coding of the message or the calculation of the CRC the controller will send back an error code indicating the parameter value is not accepted [0x15].

If the message has no errors the written data will be sent back in the following format :

- Parameter value is accepted [0x06]
- Value (pressure in mbar)
- Checksum calculation [CRC]
- End of transmission [0x04]

From this message we want to extract only the pressure values, and insert it into the post request function to send to the database. This meant translating the ASCII message into utf-8 and removing [0x06] and [CRC][0x04]. It was important that the Raspberry Pi be able to identify the timings from the received data as to match a corresponding timestamp.

Adding to this extra options such as being able to choose which channel to communicate with and how to name them, makes the code intuitive to use. Creating a bunching function greatly increased the efficiency of sending data. This driver will facilitate future improvements to the code as there is no longer a need to research connection protocols between the different components of the set-up. The full code has been appended (listing 2). Once all the software was functional, the RS-232 to USB cable was welded on a 9-pin to connect the cable to the gauge controller and the Raspberry Pi. A second cable was also made as replacement or for future use. Once the information is sent from the gauge controller to the database, the Influxdb server takes care of the formatting and plotting of the batched data points.

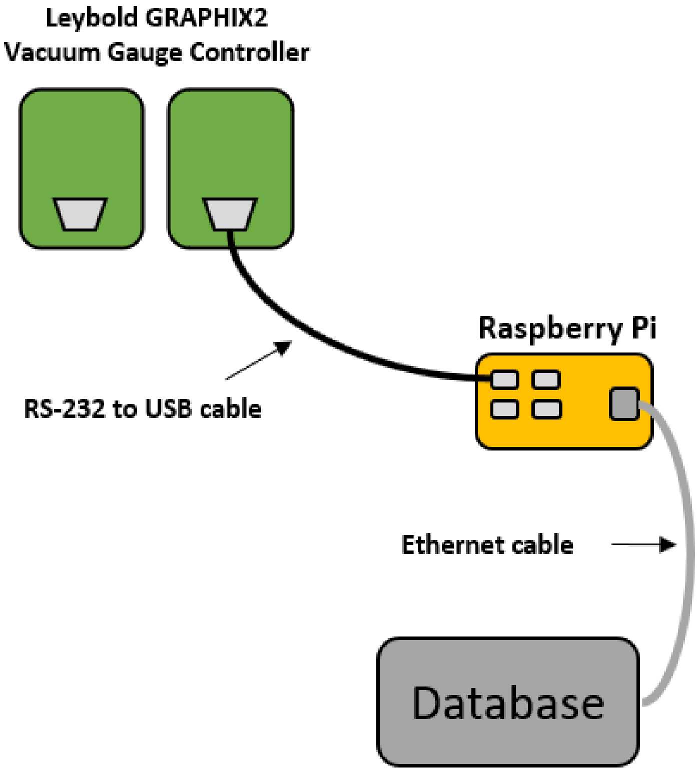

Figure 3: Schematic overview of the data being sent from the gauge controller to the database

3 Experimental Methods

3.1 Offline Set-up

Currently, the ion-guide buncher and MR-ToF MS are being commissioned in an offline version. A surface ion source provides $^{133}Cs^{+}$ ions. Figure 4 is a representation of the current offline set-up, whilst figure 5 shows the section that was used during these measurements up to the first MCP detector. At present the gas-catcher is separated from the set-up and the ion-source is placed in front of the ion guide. The source is supplied by a potential and heated by a current, this produces the ions that travel through the set-up. We should note that the source has some variations when delivering ions, in a short time frame (seconds/minutes) this variation is negligible at a standard deviation of 16% over 90 second measurements, over long periods of time (hours/days) the source ion current may increase or decrease more significantly. Once the ions leave the source they are sent through to the ion-guide. The first section of the ion-guide functions as a funnel, gradually pushing the ions to the beam axis by using electric fields, then they proceed into the thermalization section where they are confined by an RF of 160 Vpp at 3.97 MHz and carried by a travelling wave in the beam axis at 333 Hz and 3 Vpp, induced by deviating the duty cycle of square waves. The ions are refocused before reaching the ejection section where they are stored in a potential bucket where they are stored and bunched. An ejection voltage gradient is applied to the last 4 rings to properly eject the ions collected inside this bucket. They are further accelerated by a Pulsed Drift Tube (PDT) and will go into the MR-ToF MS passing through two Einzel lenses and an iris. The Einzel lenses focus and steer the ion bunch once after the ion-guide and once before entering the MR-ToF MS, having a well-focused ion bunch is essential in having accurate measurements. Any deviation by the ions caused by these lenses will create an offset angle when the ions hit the detector. This results in a larger measured FWHM of the ion bunches ToF distribution. The iris serves as a separator between the ion-guide and the MR-ToF MS because of the differential pumping. A first turbo-molecular pump (of capacity 350 L/s) is installed on the ion-guide chamber, in the later set-up this will be replaced by a 2200 L/s pump, to effectively withdraw the helium from the gas-catcher and create a supersonic jet flow into the ion-guide. This is the first stage to achieve a potential high vacuum later in the system. It is expected that 10^{-3} mbar is the optimal pressure for ion transmission [10]. After this a second pump (of capacity 350 L/s) is placed between the ion-guide and the MR-ToF MS where a PDT is placed. The pump

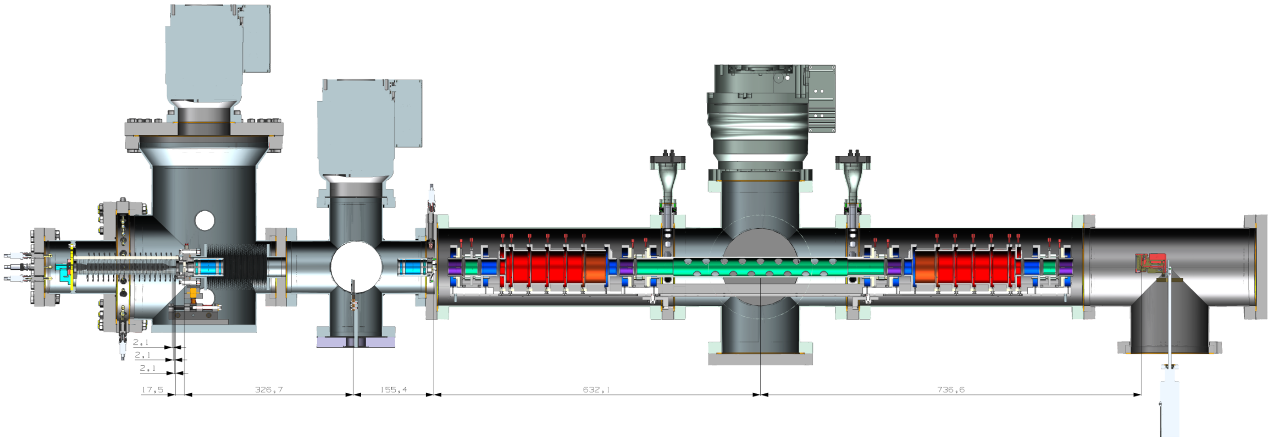

Figure 4: Offline set-up of the project in half-section view. Including : $^{133}Cs^{+}$ source, ion-guide, two Einzel lenses, one iris, two pulsed drift tubes, MR-ToF MS, and two MCP detectors

in this section creates a high vacuum at 10^{-6} mbar, this pressure limits the number of interactions between the ions. Finally, a third pump (of capacity 700 L/s) is placed on the MR-ToF MS as to keep a pressure of 10^{-9} mbar, needed for the device to function properly.

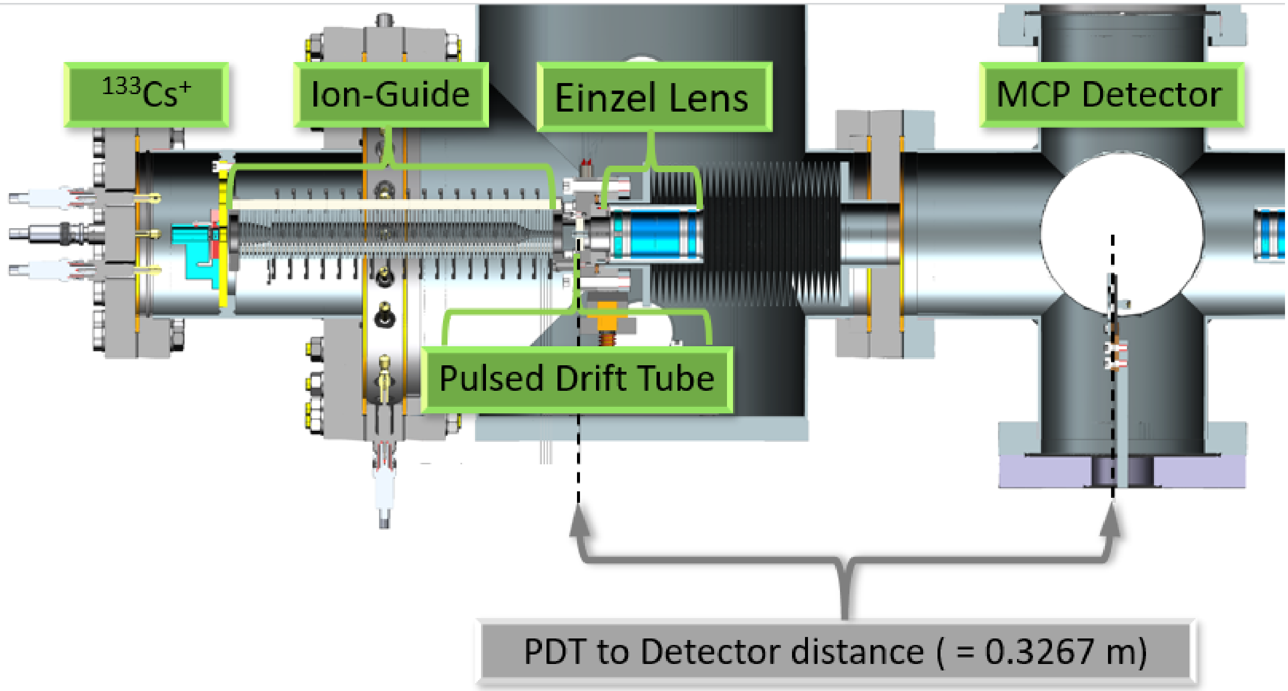

Figure 5: Offline set-up of the project in half-section view, this is the section, that was used during the measurements performed for this thesis. The measurements start their recordings when the ions leave the pulsed drift tube and travel to the detector, a distance for the rough-line of l = 326.7 mm

3.1.1 Mean free path

Using the kinetic theory of gases, we can estimate the mean free path of the ions passing through as a function of pressure. This will be done using the ideal gas law (1) because the gas inside the ion-guide is helium, which can be approximated as an ideal gas. We have the following variables:

- P = pressure (Pa)
- V = volume (L)
- N = number of molecules
- k_B = Boltzmann constant = 1.38×10^{-23} $m^2 s^{-2} kg^{-1}$
- T = temperature = 295 K
- λ = mean free path (m)
- n = number of moles of gas
- a_1 = van der Waals radius of cesium = 343 pm [11]
- a_2 = van der Waals radius of helium = 140 pm [11]

- σ = collision cross section = $\pi(a_1+a_2)^2 = \pi(343\times10^{-12}+140\times10^{-12})^2 = 7.39\times10^{-19}$ m^2

According to the ideal gas law:

$$PV = Nk_BT \tag{1}$$

The number density n is given by:

$$n = \frac{N}{V} \tag{2}$$

The mean free path λ can be expressed as:

$$\lambda = \frac{1}{\sqrt{2}n\sigma} \tag{3}$$

By substituting n from the ideal gas law, we get:

$$n = \frac{P}{k_BT} \tag{4}$$

Thus, the mean free path λ is:

$$\lambda = \frac{1}{\sqrt{2}\sigma}\cdot\frac{k_BT}{P} \tag{5}$$

Demonstrating that the mean free path is inversely proportional to the pressure of the system at constant temperature T. It follows that a pressure of $\times10^{-3}$ mbar corresponds to a mean free path of roughly 0.039 m or about 39 mm. The total length of the ion-guide is 190.8 mm [7]. Causing multiple collisions to thermalize the ions inside the ion-guide, this confirms that helium is an efficient buffer gas [12][10]. If we were to compare this to other noble gases, such as argon gas with a_3 = 188 pm, resulting in $\sigma = 8.86\times10^{-19}$ m^2. Giving a mean free path of λ = 32 mm. This seems to be better but when taking into account the thermal conductivity of argon (0.01772 W/mK) [13] it is much better to use helium (0.1513 W/mK) [13], as this outweighs the small difference in mean free path. In the Einzel lens section, with a pressure of $\times10^{-6}$ mbar, the mean free path in helium extends to 39 m.

Investigating the mean free path for the ions inside the MR-ToF MS , where there is no buffer gas, will allow for confirmation of the required pressure. The MR-ToF MS is kept in vacuum as variations in energy due to thermalization is undesirable. The time of flight of the ion bunch is measured at a known distance, an electric field is applied to the ion bunch to accelerate them to a certain kinetic energy, this is why thermalization is detrimental. This time of flight is dependant on the velocity and therefore it is a measure of mass-to-charge ratio.
Inside the MR-ToF MS Using equation 5 and inserting $\sigma = \pi(2\times a_1)^2 = \pi(2\times343\times10^{-12})^2 = 1.48\times10^{-18}$ m^2 gives us :

- for P = $\times10^{-8}$ mbar = $\times10^{-6}$ Pa
- λ = 1.95 km
- for P = $\times10^{-9}$ mbar = $\times10^{-7}$ Pa
- λ = 19.5 km

Resulting in a mean free path of the ions in the MR-ToF MS at vacuum between 2 and 20 km, considering the length travelled by the ion bunch inside is about 1 km, this pressure is more than sufficient for proper operation, even if there is a small drift in pressure.

3.2 Lab Measurements

To test the efficiency of the set-up, some measurements were performed with varying helium pressures, mainly looking at the rough-line, the section between the ion-guide and the first MCP detector. This was done using the lab measurement software in which different voltages and frequencies can be adjusted. Differing the helium inflow using the Pfeiffer inflow valve will change the helium pressure in the system, the pressure was varied from approximately 10^{-2} mbar to 10^{-4} mbar. The power supply was set at 80 V positive and 80 V negative and 3.97 MHz frequency on the RF, and the ejection from the ion-guide voltage was set at a gradient from 146 V to 85 V on the last ring. Two different source potentials were used, one at 2 V and one at 4 V. To get a detectable amount 0.9 V was applied to the extractor. Another set of measurements were carried out 2 weeks later, having changed some of the parameters. This time at 90 Vpp for the RF in the ion-guide. The source potential was varied from 2 V to 9 V, 1.8 A source current and an extraction voltage of 1.1 V.

4 Results

4.1 Pressure data from Raspberry Pi

Once the drivers were written and the logging system was functional, a trial over a three day period was established, the program ran without error or missing value points. Figure 6 is a comparative plot between what was collected, in the rough-line, and what is already being monitored by the database, the ion-guide pressure and the MR-ToF MS pressure which has been multiplied by $\times 10^5$. These plots come from the time the second set of measurements were made, on 29/05/2024. As can be seen the code functions without missing data points, roughly following the pressure of the ion-guide increase and decrease over time. The data has been batched and smoothed to save space so some of the pressure points where measurements were made are not directly visible. This does however showcase that the Raspberry Pi and code work correctly as expected.

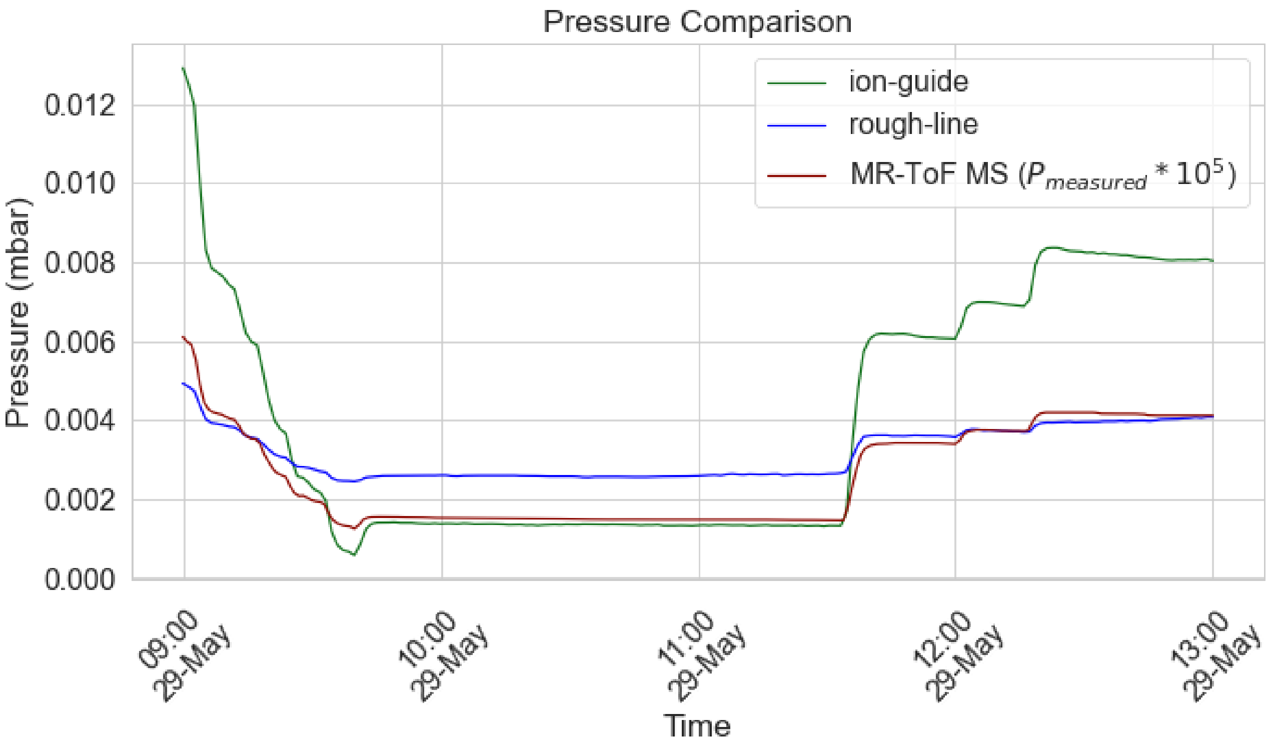

Figure 6: The data collected by the Raspberry Pi (rough-line) and Influxdb (ion-guide and MR-ToF MS) on 29/05/2024 during the measurements from 11:30 to 12:30

4.2 Lab Results

The experimental testing aims to identify how the pressure affects the count rate of the ions, the efficiency of the bunching and any time of flight variations. Some source fluctuation is expected but should be negligible over the time period of the measurements. Systematic errors are taken into account, these include some intrinsic delays from the electronic components, software and hardware. The most significant being the internal propagation delay time of the fast high voltage transistor switches 150 ns [14], including some cable delay and internal resistor delay we can approximate this to 184 ns. A small uncertainty on the distance exists but as this is on the order of ≈ 2% of 326.7 mm it can be ignored.

4.2.1 Time of Flight

The detector allows for a measurement of the time of flight, this data will eventually be used to determine the mass of the nuclei in the MR ToF MS. Through the kinetic energy of the ions (E_k), the mass ($m \approx 2.2 \times 10^{-25}$ kg) and their travel path distance ($l = 0.3267$ m), we can determine their time of flight (t_{ToF}) :

$$E_k = \frac{1}{2}mv^2 \tag{6}$$

$$v = \frac{d}{t} \tag{7}$$

$$E_k = \frac{m}{2} \times (\frac{d}{t})^2 \tag{8}$$

$$t^2 = \frac{m}{2E_k} \times d^2 \tag{9}$$

$$t = d\sqrt{\frac{m}{2E_k}} \tag{10}$$

$$t_{ToF} = l\sqrt{\frac{m}{2E_k}} \tag{11}$$

Using the time of flight it is possible to convert to energy, assuming the distance traveled by all the ions is the same, in reality this is not the case. However, for the sake of simplicity, we shall ignore the small variations in spatial distribution. By manipulating equation 11 we obtain :

$$E_k = (\frac{l^2}{{t_{ToF}}^2})\frac{m}{2} \tag{12}$$

We can then calculate the average kinetic energy of the ion bunch as it travels to the first MCP detector. This is not a good representation of the actual kinetic energy of the ion bunches, as the Einzel lens will create a variation in velocity.

Figures 7 and 8 are plots of the data collected with a 4 V source potential, the Python fit parameters can be found in table 1 and table 3 in the appendix. The software utilized in the laboratory applies a Gaussian fit for the chosen data, which is sub optimal. A Gaussian fit presumes a normal symmetrical distribution, which does not hold true in our situation. As can be seen on both figures, there is tailing right after the peak, due to some ions from the bunch arriving later that the core bunch. This is why a double Gaussian has been fitted on the data, with a skew. This allows for a more precise estimation of the different factors of interest. In theory a Boltzmann distribution fit would work best, using the energy distribution, this was tested but inconclusive. The double Gaussian does however present a good enough approximation, due to the tailing, we can consider two peaks. One peak at the tail and one peak at the ion bunch core, adding these two gets a better fit on the data and adding a skew accounts for the asymmetric nature. This is enough precision to be of interest. The fitting was done using the following equations in a Python script which reads the tab separated text files collected during measurements (listing 3).

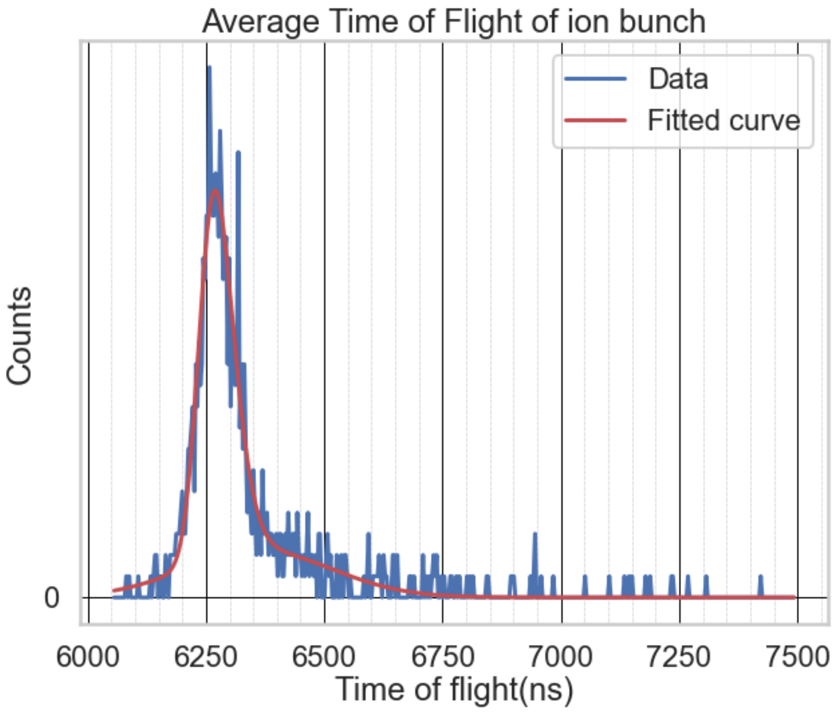

Figure 7: Plot at pressure = 1×10^{-3} mbar, double Gaussian curve fitted with skew. Parameters of the fitted curve : Mean tof = 6258 ns ; FWHM = 155 ns ; Amplitude = 70 counts, and a skew = 2.28

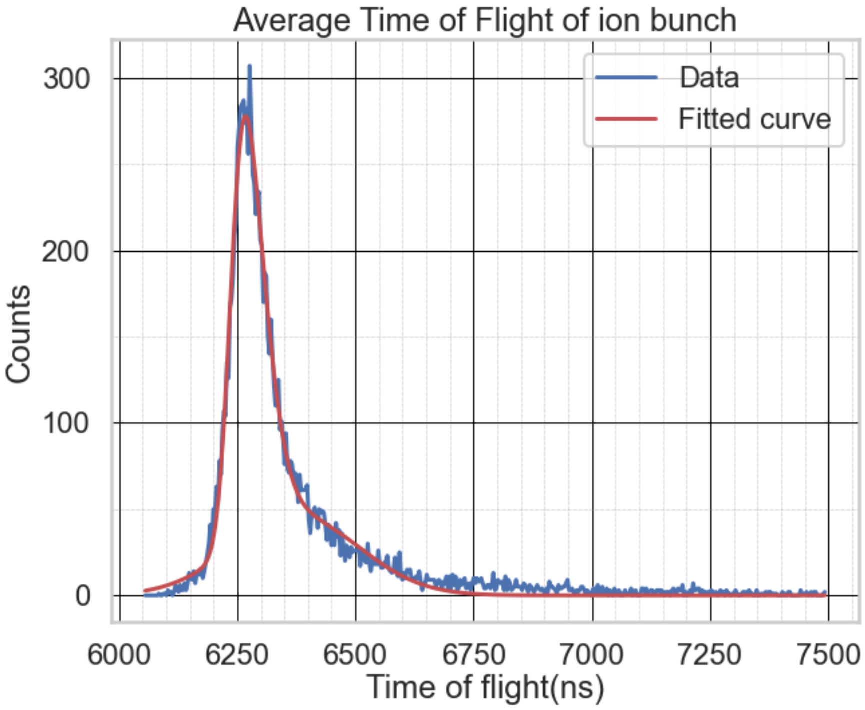

Figure 8: Plot at pressure = 1.07×10^{-2} mbar, double Gaussian fitted curve with skew. Parameters of the fitted curve : Mean tof = 6255 ns; FWHM = 133 ns ; Amplitude = 280 counts, and a skew = 2.09

To apply the double Gaussian distribution we have the following :

- x = Time of flight
- μ_i = Mean peak time of flight of Gaussian 1 and 2
- σ_i = Standard deviation of Gaussian 1 and 2 = FWHM (1, 2)
- A_i = Amplitude of the Gaussian 1 and 2
- a = Amplitude of the skew

The Gaussian function is given by:

$$g_i(x;\mu_i,\sigma_i,A_i) = A_i \cdot \exp\left(-0.5\left(\frac{x-\mu_i}{\sigma_i}\right)^2\right)$$

The skew function is:

$$s(x;\mu,\sigma,A,a) = A \cdot \exp\left(-0.5\left(\frac{x-\mu}{\sigma}\right)^2\right) \cdot \left(1-\operatorname{erfc}\left(\frac{a(x-\mu)}{\sigma\sqrt{2}}\right)\right)$$

The double Gaussian function combines two Gaussian functions:

$$G(x;\mu_1,\sigma_1,A_1,\mu_2,\sigma_2,A_2) = g_1(x;\mu_1,\sigma_1,A_1)+g_2(x;\mu_2,\sigma_2,A_2)$$

The fit bunch function combines the double Gaussian and skew functions:

$$f(x;\mu_1,\sigma_1,A_1,\mu_2,\sigma_2,A_2,a) = G+s$$

4.2.2 Count rate

According to the data collected in figure 9 we can see that the higher count-rate and presumably better bunching happens at a helium pressure of around 1×10^{-2} mbar. The count rate slowly increases before 8×10^{-3} mbar, then we have a stable area around 1×10^{-2} mbar and finally a decrease when we allow more helium into the system. The slow rise in counts can be explained by the ions thermalizing less efficiently at lower helium pressure, these ions do not get thermalized fast enough and their energy is too high to be caught in the travelling wave inside the ion-guide, instead they possibly hit the gating voltage at the end of the ion-guide and reflect back and exit the ion-guide this results in less effective bunching and a lower count rate. On the other end, when approaching high helium pressures the ions are potentially thermalized too quickly resulting in their energies being too low before entering the ion-guide and being rejected at the entrance, additionally helium impurities should be taken into account at higher pressures. When the source potential is increased from 3 V to 4 V the count rate goes up more significantly for the higher helium pressure, this supports the hypothesis that the ions have too low energy at the entrance of the ion-guide.

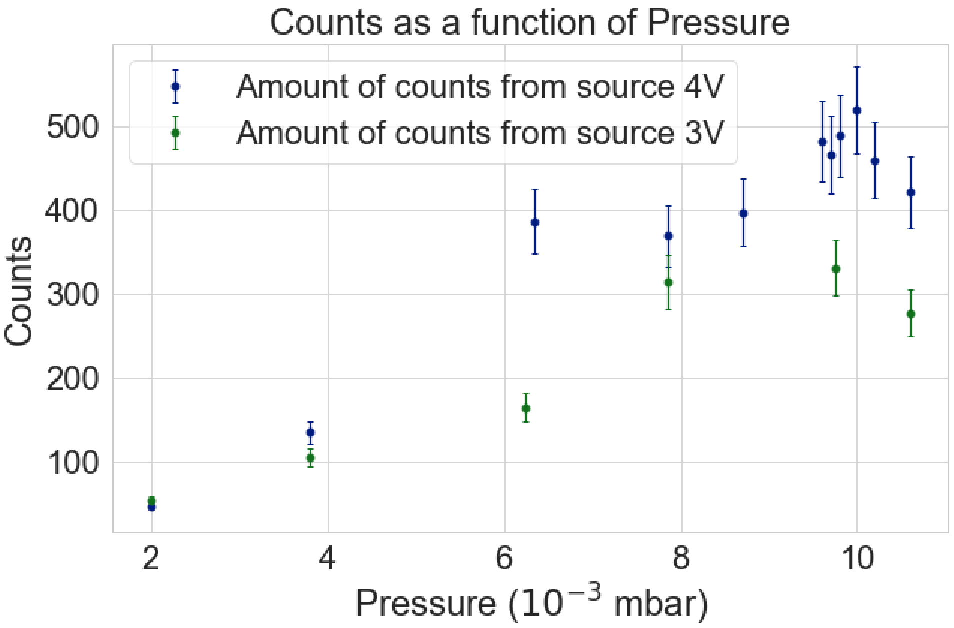

Figure 9: Plot showing the total counts collected during measurements from 13/06/2024

4.2.3 Full-Width at Half Maximum

Another interesting result to look at is the full width half maximum of the ion bunches during the measurements. By investigating the time of flight distribution, and applying a double Gaussian fit, the FWHM can be determined. This will quantify the efficiency of the ion bunching at the detector. This is relevant in understanding the efficiency of the ion-guide as this will display the variation and width of the peak. Meaning we know the shape of the ion bunches. A smaller FWHM indicates better bunching. The FWHM is defined by the following equation using the standard deviation (σ):

$$\text{FWHM} = 2\sqrt{2\ln(2)}\sigma = 2.335\sigma \tag{13}$$

The data in figure 10 suggests that there are 2 areas where minimal FWHM was reached at a pressure of roughly 5×10^{-3} mbar and 1×10^{-2} mbar. Both the 2 V and 4 V source potential present only small variations. Again we see that at lower helium pressures the data is not ideal, the large uncertainty seen can be explained. The thermalization process being inefficient and having a low count rate means that when the fitting is applied there's large inaccuracies due to low statistics.

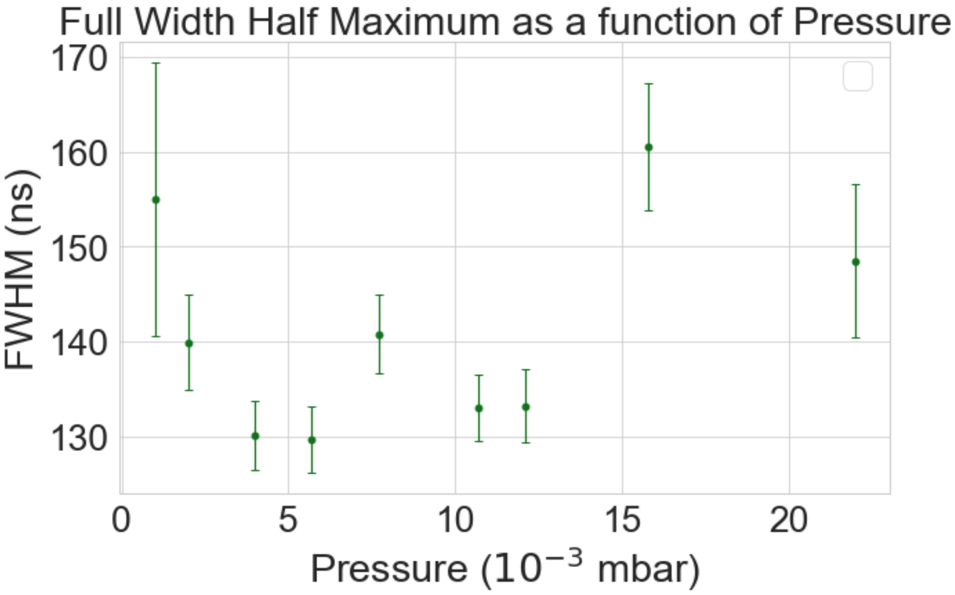

Figure 10: Plot showing the FWHM (ns) as a function of pressure at 4V source potential

According to the plots in figures 10 and 11, and the parameters calculated by the fitting we can assume that a pressure of 1.07×10^{-2} mbar is an ideal sweet spot. We want to maintain a relatively low FWHM but also get a sufficient count rate of ions and this seems to be the best pressure to do so.

5 Conclusion and discussion

To conclude, the objective of this research project has been firstly to develop a reliable and versatile pressure logging system that can be applied to different parts of the system in future improvements. Secondly to determine the ideal helium pressure for the ion-guide efficiency. Experimental measurements were carried out, from 10^{-4} mbar to 10^{-2} mbar, effectively narrowing the ideal helium pressure to 6×10^{-3} mbar to 1×10^{-2} mbar. A more detailed analysis suggests 1×10^{-2} mbar is an ideal sweet spot regarding count rate and bunching efficiency.

Additions such as the connection between the pressure gauge controller and the database are long-term advancements for the project. This will allow for more data to be collected and for a better understanding of the set-up.

Additional time would have been beneficial in being able to carry out further measurements concerning the interactions between different parameters and the pressure. Similar to measurements made with different source voltages. More measurements would have resulted in more accurate data analysis.

Additions such as thermal gauges would be beneficial in monitoring the set-up in more detail, implementing safety signals inside the Raspberry Pi could also be considered to improve overall safety of the experiment. This could avoid very low or high external temperatures from negatively impacting components unexpectedly. Although once the set-up is moved to the permanent line some of these issues would resolve themselves, temperature variation would be minimal in the new lab. Developing a computational method to control the helium pressure would accelerate measurements, the Raspberry Pi could be key to accomplish this task by creating a new script that interacts with the valves control system.

Bibliography

[1] T. Dickel, A. Kankainen, A. Spătaru, D. Amanbayev, O. Beliuskina, S. Beck, P. Constantin, D. Benyamin, H. Geissel, L. Gröf, C. Hornung, A. V. Karpov, I. Mardor, G. Münzenberg, D. Nichita, W. R. Plaß, I. Pohjalainen, S. Purushothaman, M. Reponen, A. Rotaru, V. V. Saiko, C. Scheidenberger, J. S. Winfield, and A. Zadvornaya, "Multi-nucleon transfer reactions at ion catcher facilities - a new way to produce and study heavy neutron-rich nuclei," vol. 1668, 2020.

[2] A. Türler and V. Pershina, "Advances in the production and chemistry of the heaviest elements," 2013.

[3] J. Even, X. Chen, A. Soylu, P. Fischer, A. Karpov, V. Saiko, J. Saren, M. Schlaich, T. Schlathölter, L. Schweikhard, J. Uusitalo, and F. Wienholtz, "The next project: Towards production and investigation of neutron-rich heavy nuclides," *Atoms*, vol. 10, 2022.

[4] B. Pfeiffer, K. L. Kratz, F. K. Thielemann, and W. B. Walters, "Nuclear structure studies for the astrophysical r-process," *Nuclear Physics A*, vol. 693, 2001.

[5] F. M. Nunes, "Why are theorists excited about exotic nuclei?," *Physics Today*, vol. 74, 2021.

[6] A. Mollaebrahimi, B. Anelić, J. Even, M. Block, M. Eibach, F. Giacoppo, N. Kalantar-Nayestanaki, O. Kaleja, H. R. Kremers, M. Laatiaoui, and S. Raeder, "A setup to develop novel chemical isobaric separation (cise)," *Nuclear Instruments and Methods in Physics Research, Section B: Beam Interactions with Materials and Atoms*, vol. 463, 2020.

[7] X. Chen, J. Even, P. Fischer, M. Schlaich, T. Schlathölter, L. Schweikhard, and A. Soylu, "Stacked-ring ion guide for cooling and bunching rare isotopes," *International Journal of Mass Spectrometry*, vol. 477, 2022.

[8] "Influxdb line protocol representation," 2024. `https://docs.influxdata.com/influxdb/v2/get-started/write/?t=InfluxDB+API`.

[9] L. GmbH, "Instruction manual," 2024. `https://www.leyboldproducts.com/media/pdf/0f/9e/dc/300550402_002_C1_GRAPHIX_123_EN.pdf`.

[10] D. J. Douglas and J. B. French, "Collisional focusing effects in radio frequency quadrupoles," *Journal of the American Society for Mass Spectrometry*, vol. 3, 1992.

[11] M. Mantina, A. C. Chamberlin, R. Valero, C. J. Cramer, and D. G. Truhlar, "Consistent van der waals radii for the whole main group," *Journal of Physical Chemistry A*, vol. 113, 2009.

[12] A. K. Hansen, O. O. Versolato, L. Kłosowski, S. B. Kristensen, A. Gingell, M. Schwarz, A. Windberger, J. Ullrich, J. R. López-Urrutia, and M. Drewsen, "Efficient rotational cooling of coulomb-crystallized molecular ions by a helium buffer gas," *Nature*, vol. 508, 2014.

[13] J. Dean, *Lange's Handbook Of Chemistry, 15th ed.* 1999.

[14] B. P. E. GmbH, "Behlke technical data," 2024. `https://www.behlke.com/pdf/61-03-gsm.pdf`.

Appendices

A Tables

Table 1: Parameters from the fitting on the data, 4 V Source potential

Pressure (mbar)	Mean FWHM (ns)	Mean peak TOF (ns)	skew
1.0×10^{-3}	155.328 ± 15.678	6258.578 ± 2.316	2.28
2.0×10^{-3}	140.819 ± 6.902	6288.872 ± 4.521	3.14
4.0×10^{-3}	130.313 ± 3.465	6308.607 ± 6.251	2.33
5.7×10^{-3}	127.636 ± 3.910	6255.319 ± 2.544	1.50
7.7×10^{-3}	143.251 ± 3.719	6258.648 ± 2.434	1.87
1.07×10^{-2}	133.842 ± 4.275	6255.811 ± 2.665	2.09
1.21×10^{-2}	136.898 ± 14.750	6250.793 ± 7.997	1.65
1.58×10^{-2}	162.624 ± 4.167	6270.983 ± 2.797	2.35
2.20×10^{-2}	146.168 ± 5.182	6252.072 ± 3.049	1.62

Table 2: Parameters from the fitting on the data, 2 V Source potential

Pressure (mbar)	Mean FWHM (ns)	Mean peak TOF (ns)
0.7×10^{-3}	119.017 ± 12.604	6327.964 ± 9.137
2.2×10^{-3}	138.057 ± 7.100	6307.948 ± 5.332
3.75×10^{-3}	121.796 ± 5.761	6305.415 ± 4.489
5.95×10^{-3}	138.233 ± 6.073	6302.553 ± 4.451
7.75×10^{-3}	145.924 ± 6.622	6305.109 ± 5.066
1.25×10^{-2}	194.468 ± 9.502	6222.922 ± 1.162
1.98×10^{-2}	173.523 ± 8.032	6196.426 ± 3.184

Table 3: Parameters from the fitting on the data collected on 13/06/2024, low count rate has caused high uncertainty in the fitting of some of the data

Pressure	Mean FWHM (ns)	Mean Peak TOF (ns)
3.8×10^{-3}	132.826 ± 19.718	6031.115 ± 10.656
6.35×10^{-3}	138.331 ± 17.231	6023.146 ± 11.919
7.75×10^{-3}	118.390 ± 1155.056	6051.014 ± 50092.090
7.85×10^{-3}	262.668 ± 28.511	6079.943 ± 5.329
8.7×10^{-3}	171.753 ± 8.932	6035.854 ± 4.683
9.5×10^{-3}	103.372 ± 2216.946	6054.884 ± 236695.600
9.65×10^{-3}	178.334 ± 9.932	6019.313 ± 11.546
9.7×10^{-3}	110.220 ± 7722.049	6059.347 ± 1362386.000
9.8×10^{-3}	320.686 ± 28.311	6062.352 ± 6.071
9.8×10^{-3}	157.922 ± 14.501	6040.938 ± 10.941
1.0×10^{-2}	205.577 ± 17.258	6020.739 ± 18.325
1.02×10^{-2}	322.041 ± 49.597	6261.365 ± 42.318
1.04×10^{-2}	163.565 ± 16.673	6023.118 ± 17.675
1.06×10^{-2}	227.260 ± 13.645	6062.283 ± 4.538
1.06×10^{-2}	221.820 ± 8.419	6068.989 ± 2.428

B Plots

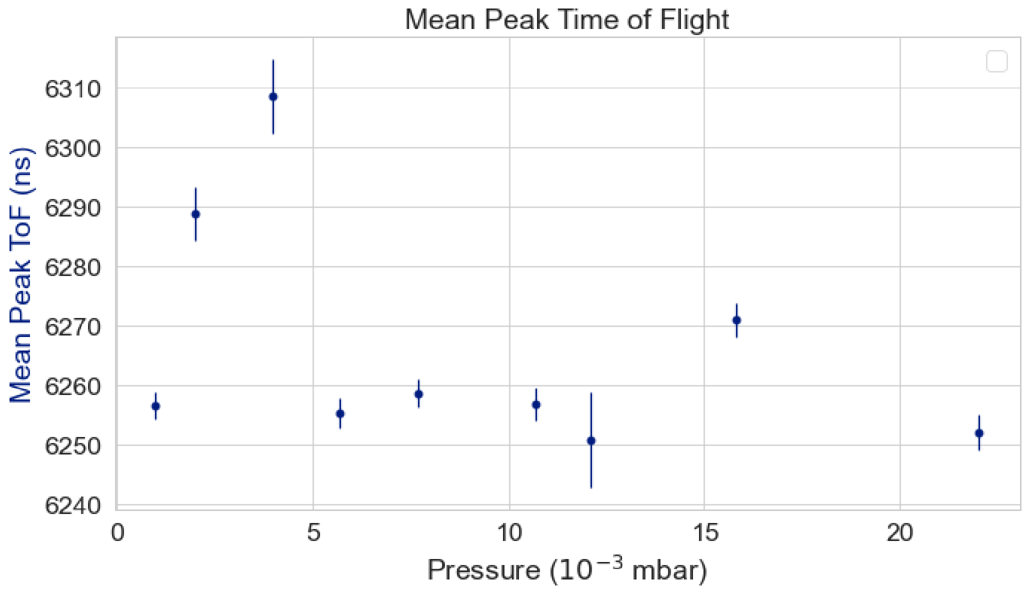

Figure 11: From Table 1 Mean TOF points from 28/05/24, 4V source potential, varying pressures

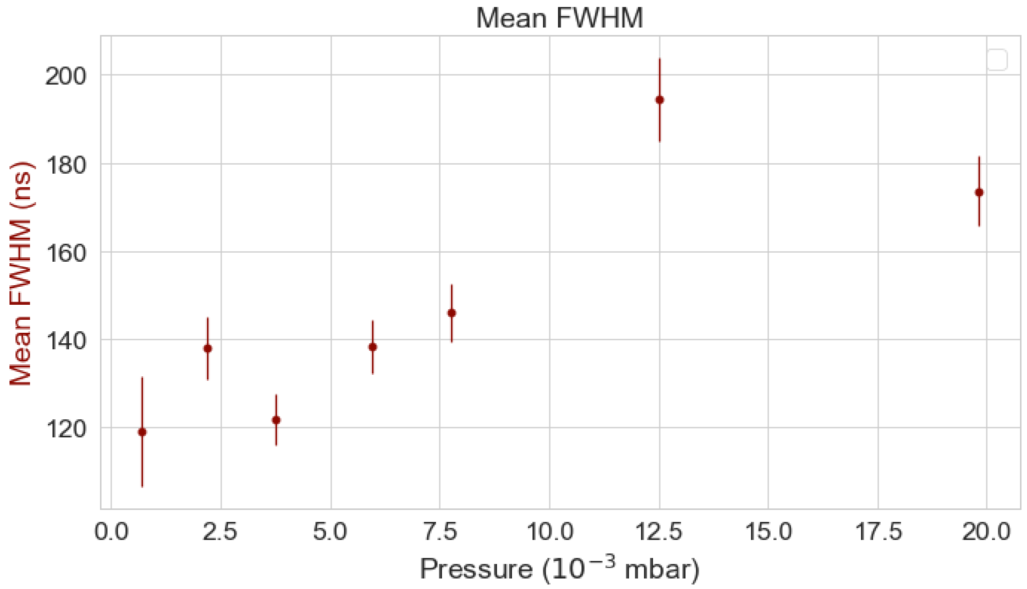

Figure 12: From Table 2 Mean FWHM points from 29/05/24, 2V source potential, varying pressures, low count rate

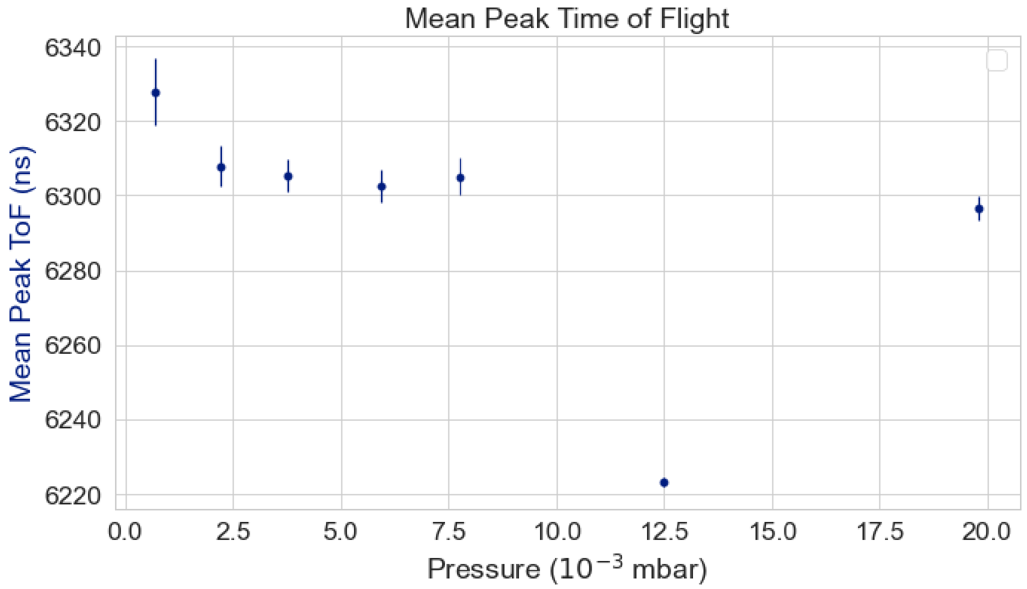

Figure 13: From Table 2 Mean TOF points from 29/05/24, 2V source potential, varying pressures, low count rate

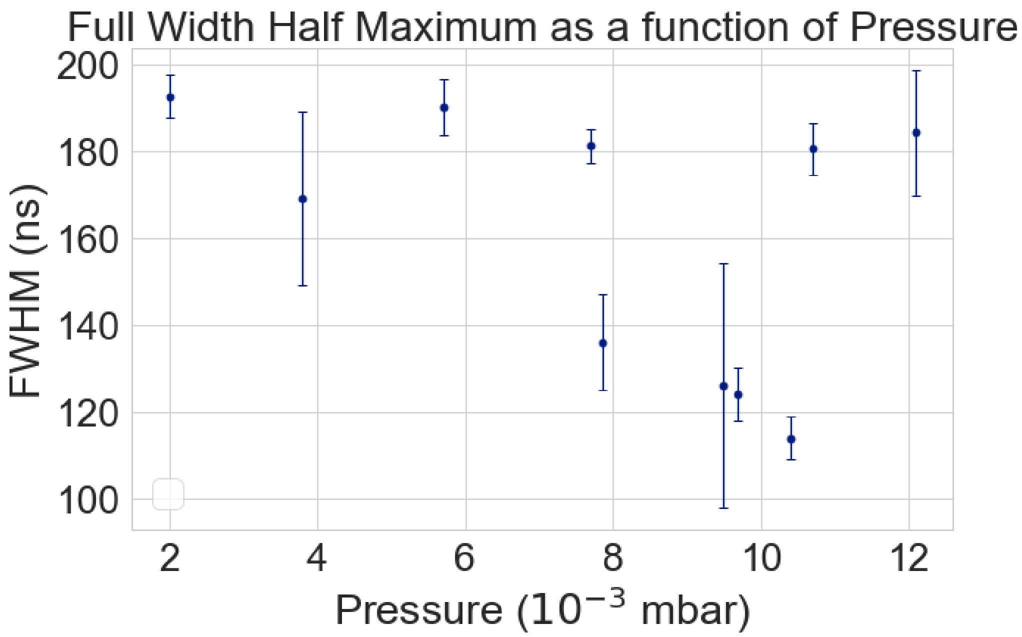

Figure 14: Mean FWHM points from 13/06/24, 4V Source potential, varying pressures, low count rate

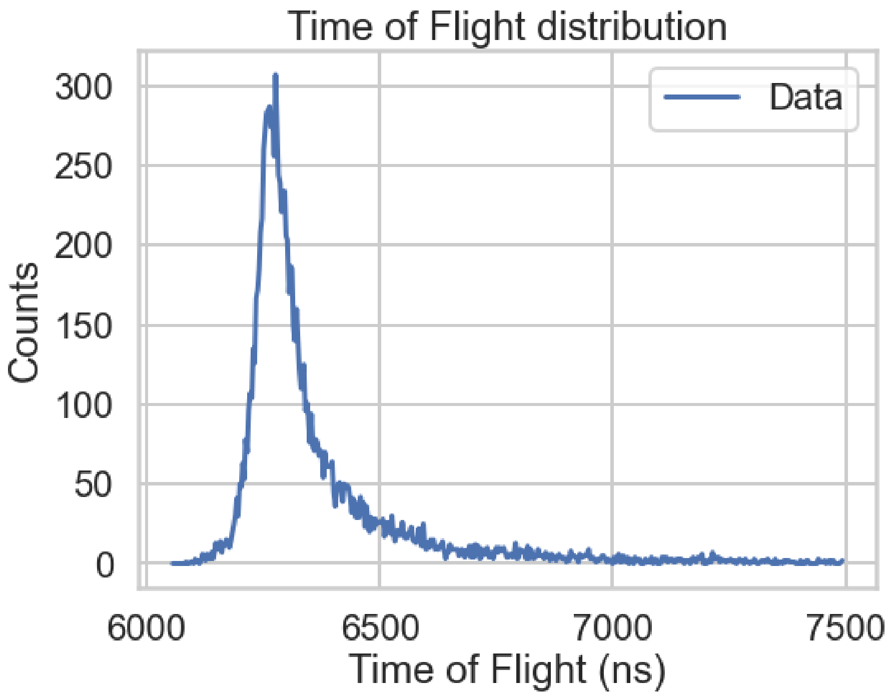

Figure 15: Raw data that was binned and plotted from 28/05/24, 4V Source potential, 1.07 $\times 10^{-3}$ mbar

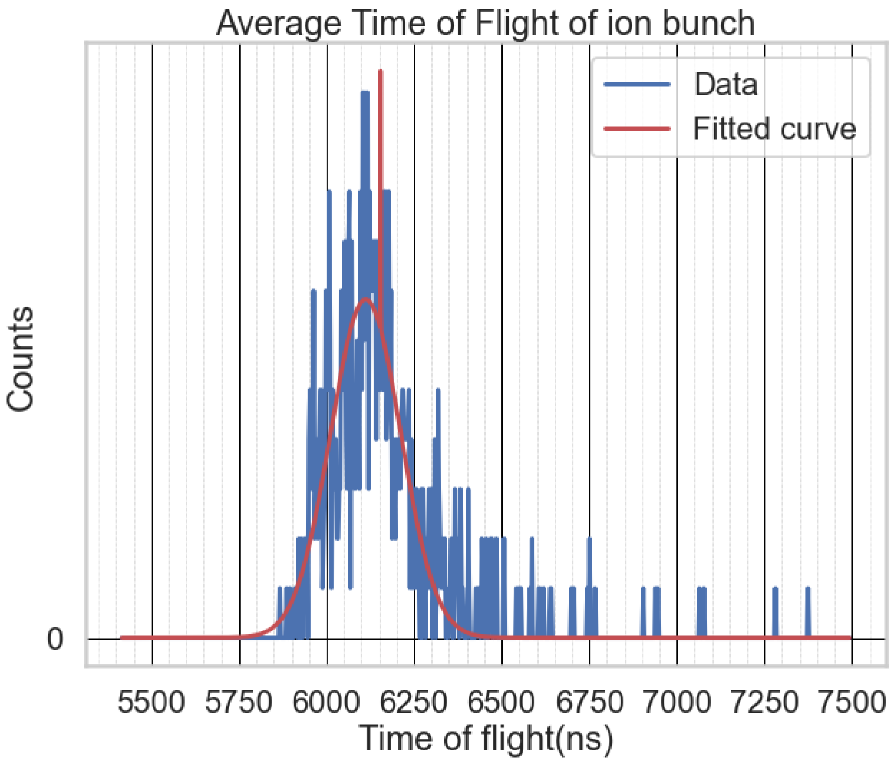

Figure 16: Example of low count data being fitted, numerous errors occur when the count rate is too low, adjusting one parameter will hinder another

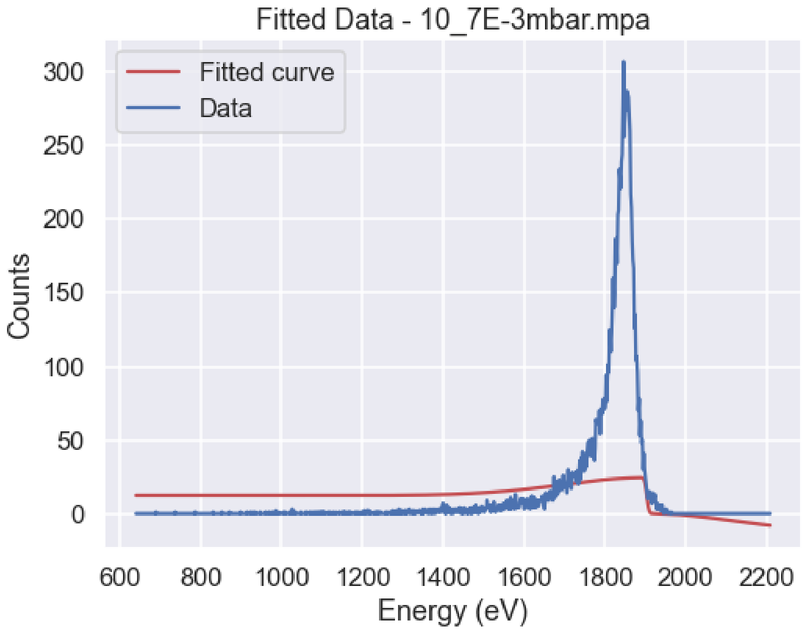

Figure 17: Example of Boltzmann distribution fit on time of flight converted to energy

C Code

Listing 2: Python code used for communication between the gauge controller and the database

```python
# -*- coding: utf-8 -*-
"""
Created on Mon May  6 11:53:16 2024

@author: blomm
"""

import serial
import requests
import json
import time
from datetime import datetime

# Serial port configuration
ser = serial.Serial(port=’/dev/ttyUSB0’, baudrate=9600, bytesize=serial.EIGHTBITS, parity=serial.PARITY_NONE, stopbits=serial.STOPBITS_ONE, timeout=1)

# Function to calculate checksum
def calculate_checksum(data):
    byte_sum = sum(ord(char) for char in data)
    checksum = 255 - ((byte_sum) % 256)
    if checksum < 32:
        checksum += 32
    return chr(checksum)

# Function to send command to the device
def send_command(command):
    checksum = calculate_checksum(command)
    full_command = f"{command}{checksum}\x04"
    ser.write(full_command.encode(’ascii’))

# Function to receive response from the device
def receive_response():
    response = ser.readline().decode(’ascii’)
    return response

# Functions to request sensor data
def request_pressure_value():
    send_command(’\x0F1;29’)
    response = receive_response()
    return response

def request_channels():
    send_command(’\x0F5;8’)
    response = receive_response()
    return response

def sensor_status():
    send_command(’\x0F1;24’)
    response = receive_response()
    return response

```

```
# Database configuration
token = "
    uqosEoKFyYegI0keAIzkiDkpc8LYM6D4tCvENvwrWDg_xZHejeq8vuXFUHgYNkLNQw9kZ4DnTdMCaHJz2LseMw
    =="
url_post = "http://192.168.10.106:8086/api/v2/write?org=NextLab&bucket=Test&
    precision=ns"
measurement = "vacuum"
tags = {"device": "graphix2", "gauge": "pressure"}

headers = {
    "Authorization": "Token " + token,
    "Content-Type": "text/plain; charset=utf-8",
    "Accept": "application/json"
}

# Batching configuration
batch_size = 10  # Number of readings to batch before sending
batch_interval = 60  # Time interval (in seconds) for sending a batch

# Initialize batch data
batch_data = []

# Loop for continuous data reading and sending
while True:
    try:
        # Read sensor data
        pressure_value = request_pressure_value()

        # Prepare data for batch
        fields = {"pressure": pressure_value}
        now = datetime.now()
        timestamp = int(datetime.timestamp(now))
        timestamp = str(timestamp*1000000000)
        line_protocol = f"{measurement},{','.join([f'{key}={value}' for key,
    value in tags.items()])} "
        line_protocol += f"{','.join([f'{key}={value}' for key, value in fields.
    items()])} {timestamp}"

        # Append data to batch
        batch_data.append(line_protocol)

        # If batch size is reached or batch interval has passed, send batch
        if len(batch_data) >= batch_size or (len(batch_data) > 0 and time.time()
     - batch_data[0] >= batch_interval):
            batch_payload = "\n".join(batch_data)
            response_post = requests.post(url_post, headers=headers, data=
    batch_payload)
            print("Data batch sent to database. Response code:", response_post.
    status_code)
            # Clear batch data after sending
            batch_data = []

        # Wait for some time before next iteration
        time.sleep(1)

    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        break

# Close serial port
ser.close()
```

Listing 3: Python code used for fitting and converting raw data from measurements initial fitting code made by Briain Hartigan double gaussian and skew were added also a time delay

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 21 10:49:27 2024
@author: S5738598
"""
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sc
from scipy.optimize import curve_fit
import math
import os
import glob
import re
import shutil

plt.rcParams.update({’font.size’:22})
sns.set_theme(context="poster", style="whitegrid", palette="deep", font_scale=1)

def sum_counts_for_rows_range(df, start_row, end_row):
    selected_rows = df.iloc[start_row:end_row + 1]
    total_counts = selected_rows[’Counts’].sum()
    return total_counts

def gauss(x, mu, sigma, amp):
    return amp * np.exp(-0.5 * ((x - mu) / sigma)**2.0)

def skew(x, mu, sigma, amp, a):
    return amp * np.exp(-0.5 * ((x - mu) / sigma)**2) * (1 - sc.special.erfc((a * (x - mu)) / (sigma * np.sqrt(2))))

def double_gaussian(x, mu1, sigma1, amp1, mu2, sigma2, amp2):
    return gauss(x, mu1, sigma1, amp1) + gauss(x, mu2, sigma2, amp2)

def fit_bunch(x, mu1, sigma1, amp1, mu2, sigma2, amp2, a):
    return double_gaussian(x, mu1, sigma1, amp1, mu2, sigma2, amp2) + skew(x, mu1, sigma1, amp1, a)

def is_float(value):
    try:
        float(value)
        return True
    except ValueError:
        return False

td_value = None
vpp_value = None
p_value = None
```

```

# Define folder locations
folder_loc = r’X:/My Desktop/BRP Blommaert/data/final_data/reffiting240529’
move_fol = r’X:/My Desktop/BRP Blommaert/data/final_data/reffiting240529’

# Initialize results file
output_file = os.path.join(folder_loc, "_results.csv")
with open(output_file, ’w’) as f:
    f.write("td (ns),Vpp (V),pressure (mbar),Total Counts,center1 (ns),sigma1 (
   ns),amp1 (Counts/bin),center2 (ns),sigma2 (ns),amp2 (Counts/bin),a,mean FWHM,
   mean FWHM uncertainty,mean peak energy,mean peak energy uncertainty\n")

single_folder = os.path.join(folder_loc, ’*.mpa’)
for file in glob.glob(single_folder):
    with open(file, ’r’, errors=’replace’) as dataset:
        data = dataset.readlines()

    line_num = next(i + 1 for i, line in enumerate(data) if ’[TDAT0’ in line)
    dataframe = pd.read_csv(file, sep=’\t’, skiprows=line_num, names=[’time(ns)’
   , ’Counts’])
    pattern = r"td(\d+)ticks_Vpp(\d+)V_P(\d+e[-+]?\d+)"
    matches = re.search(pattern, file)

    if matches:
        td_value = int(matches.group(1)) * 6.25
        vpp_value = int(matches.group(2))
        p_value = "{:.2E}".format(float(matches.group(3).replace(’e’, ’E’)))
        print(f"Extracted values: td={td_value}, vpp={vpp_value}, p={p_value}")

    counts_in_peak = sum_counts_for_rows_range(dataframe, 5800, 6400)

    selected_data = dataframe.iloc[1950:2400]
    selected_data = selected_data[selected_data[’time(ns)’].apply(is_float)]

    x_data_tof = (selected_data[’time(ns)’].values.astype(float) - 184)
    y_data_tof = selected_data[’Counts’].values.astype(float)

    plt.figure(figsize=(8, 6))
    plt.plot(x_data_tof, y_data_tof, ’b-’, label=’Data’)
    plt.xlabel(’Time of Flight (ns)’)
    plt.ylabel(’Counts’)
    plt.title(’Time of Flight distribution’)
    plt.legend()

    initial_guess_tof = [6255,100,160,6280,120,100,4]
    bounds_tof = ([6200, 0, 0, 6200, 0, 0, -5], [6500, 250, 350, 6500, 250, 350,
     50]) #bounds to constrain the fit
    fit_params, cov_matrix = curve_fit(fit_bunch, x_data_tof, y_data_tof, p0=
   initial_guess_tof, bounds=bounds_tof, maxfev=10000000000)
    fit_params_error = np.sqrt(np.diag(cov_matrix))

    print("Fitted parameters:")
    for i, param in enumerate(fit_params):
        print(f"param{i+1}: {param}    {fit_params_error[i]}")

    x_fit = np.linspace(min(x_data_tof), max(x_data_tof), 2000)
    y_fit = fit_bunch(x_fit, *fit_params)
```

```python

    plt.figure(figsize=(10, 8))
    plt.plot(x_data_tof, y_data_tof, 'b-', label='Data')
    plt.plot(x_fit, y_fit, 'r-', label='Fitted curve')
    plt.xlabel('Time of flight(ns)')
    plt.ylabel('Counts')
    plt.title('Average Time of Flight of ion bunch')
    plt.legend()

    # Increase grid density
    plt.minorticks_on()
    plt.grid(True, which='both', linestyle='-', linewidth='0.5')
    plt.grid(which='minor', linestyle=':', linewidth='0.5', color='grey')
    plt.grid(which='major', linestyle='-', linewidth='1', color='black')
    plt.gca().xaxis.set_major_locator(plt.MultipleLocator(250))  # Adjust major x ticks
    plt.gca().xaxis.set_minor_locator(plt.MultipleLocator(50))    # Adjust minor x ticks
    plt.gca().yaxis.set_major_locator(plt.MultipleLocator(100)) # Adjust major y ticks
    plt.gca().yaxis.set_minor_locator(plt.MultipleLocator(50))  # Adjust minor y ticks

    graphname = os.path.join(folder_loc, f"{os.path.basename(file)}.png")
    plt.savefig(graphname)
    plt.show()

    # Calculate mean FWHM and mean peak tof
    mean_fwhm = fit_params[1] + fit_params[4]
    mean_peak_tof = (fit_params[0] + fit_params[3]) / 2

    # Calculate uncertainties for mean FWHM and mean peak tof
    mean_fwhm_uncertainty = np.sqrt(fit_params_error[1]**2 + fit_params_error[4]**2)
    mean_peak_tof_uncertainty = np.sqrt(fit_params_error[0]**2 + fit_params_error[3]**2) / 2

    print(f"Mean FWHM: {mean_fwhm} +/- {mean_fwhm_uncertainty}")
    print(f"Mean peak tof: {mean_peak_tof} +/- {mean_peak_tof_uncertainty}")
    print(f"{os.path.basename(file)}")
    with open(output_file, 'a') as f:
        f.write(f"{td_value},{vpp_value},{p_value},{counts_in_peak},{fit_params[0]},{fit_params[1]},{fit_params[2]},{fit_params[3]},{fit_params[4]},{fit_params[5]},{fit_params[6]},{mean_fwhm},{mean_fwhm_uncertainty},{mean_peak_tof},{mean_peak_tof_uncertainty}\n")

    file_name = os.path.basename(file)
    destination_file_path = os.path.join(move_fol, file_name)
    shutil.move(file, destination_file_path)
```