Two Graphical User Interfaces for ISS

Summer Student Project

Edith Franziska Baader

December 10, 2018

Abstract

The ISOLDE Solenoidal Spectrometer (ISS) is one of the experiments of the HIE-ISOLDE section. Inside the solenoidal magnetic field, inelastic scattering and different transfer reactions will be studied. This report describes the design and the functionalities of the graphical user interface that was developed for controlling the drive system inside the magnet of ISS, as well as the user interface for the multiple Mesytec MHV-4 bias supply units. In the section of the GUI for controlling the drive system, the measurements of the alignment and the corresponding conclusions are included.

supervised by
Liam Gaffney
Joonas Konki

Contents

1 A GUI for controlling the drive System at ISS

1.1 Technical details of the drive System

The drive system at ISS consists of three different trails and four moving objects. Objects 1 and 2 are attached to the trail on the beam axis (see figure 1), 3 and 4 can be moved on axes perpendicular to the beam axis which are located on object 1 as can be seen in figure 2. Moreover, each axis has its own encoder strip for reading out the position (see figure 2). At the end of this strip, a stopper which prevents the objects from moving over the boundaries of the axis can be found.

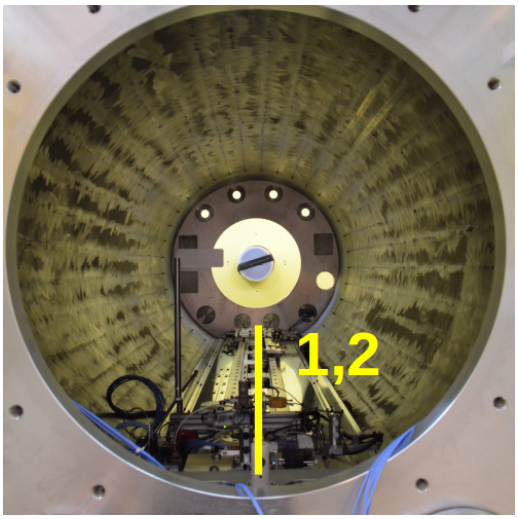

Figure 1: The beam axis which coincides with axis number 1 and 2

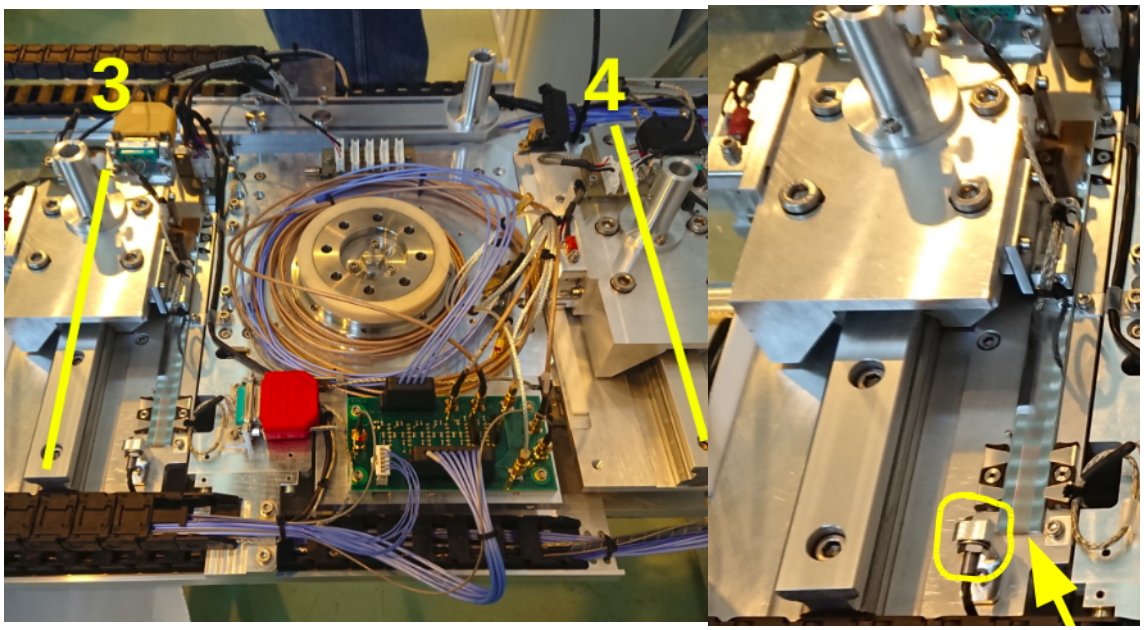

Figure 2: Axis 3 and 4 on the trolley (left) and zoom on the encoder strip and the stopper (right)

Each axis has a so-called 'datum position' whose encoder position is set to zero. The motors can move the attached objects in steps of $1/200$ mm $= 0.005$ mm. That means, that an encoder position of e.g. 600 represents a 3 mm distance from the datum position in positive direction. Axis 1 and axis 2 lie on the beam axis, but are orientated in the opposite way. This means that a movement in positive direction is in opposition to the beam direction. Axis 3 and 4 are perpendicular to the beam axis, their orientation can be found in figure 3.

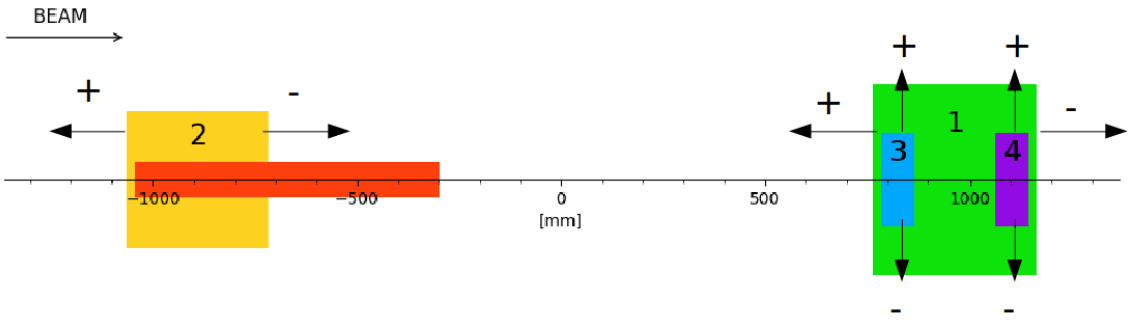

Figure 3: The orientation of the four different axes

The motors can be moved in two different ways. One option is to use the joystick which can be found inside the shielding cage of ISS. It allows to choose an axis and move the motor in slow or fast mode in positive or negative direction.
The other way is to send commands in ASCII format, which can be done by using the pySerial library of Python and knowing the following properties of the drive system:

Port name	/dev/ttyS0
Baudrate	9600
Bytesize	7 Bits
Parity	even
Stop bit	1
Timeout	3 s

Table 1: Properties of the communication with the drive system [1]

The objects which are attached to the drive system are the following (see also figure 4 and 5):

- **Axis 1:** Trolley with axis 3 and axis 4
- **Axis 2:** Silicon array
- **Axis 3:** Target ladder
- **Axis 4:** zero degree dE/E detector and Faraday cup

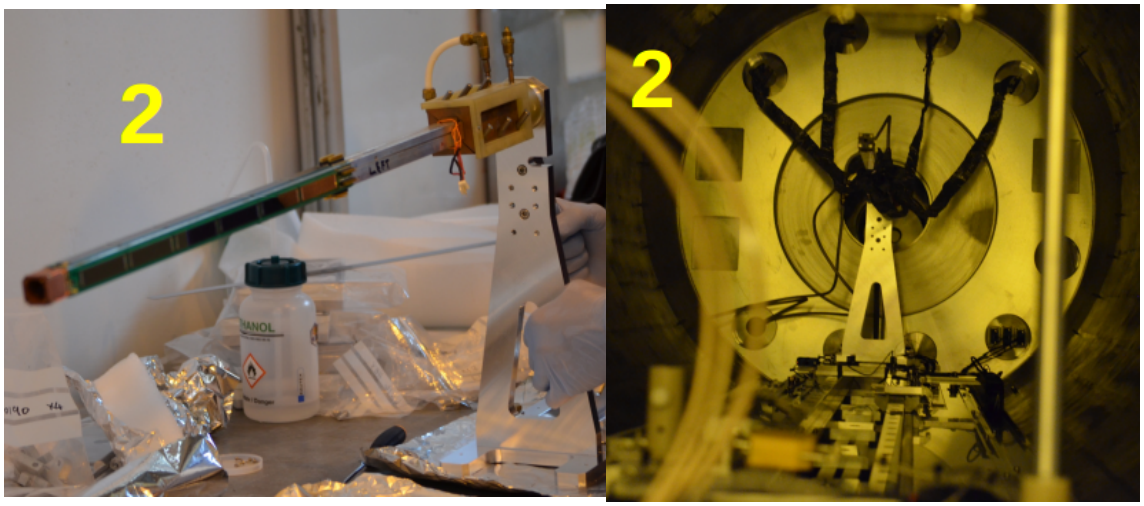

Figure 4: The silicon array which is attached to axis 2

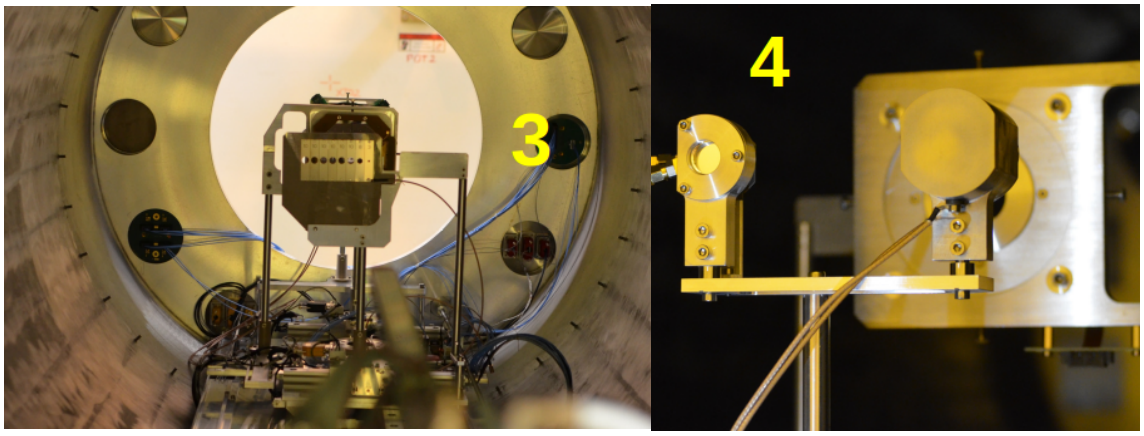

Figure 5: Target ladder on axis 3 and beam diagnostics on axis 4

1.2 Results from the alignment

1.2.1 Relative distances

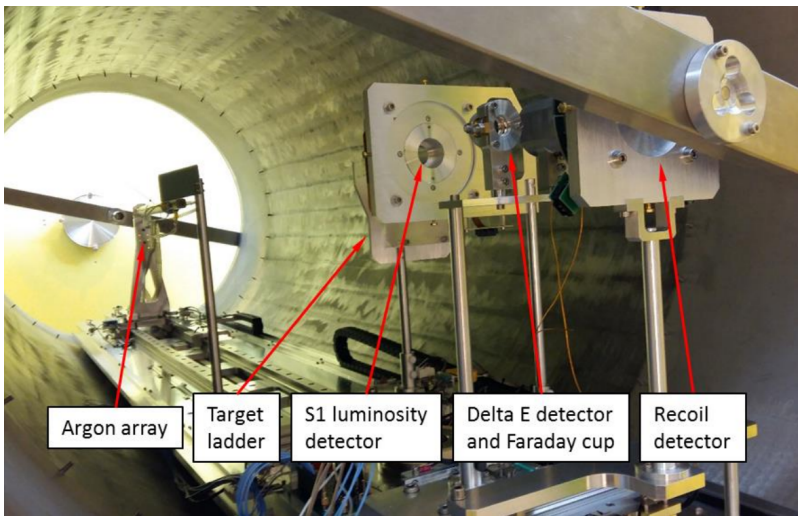

Figure 6: The objects whose relative distances to each other were determined during the alignment [3]

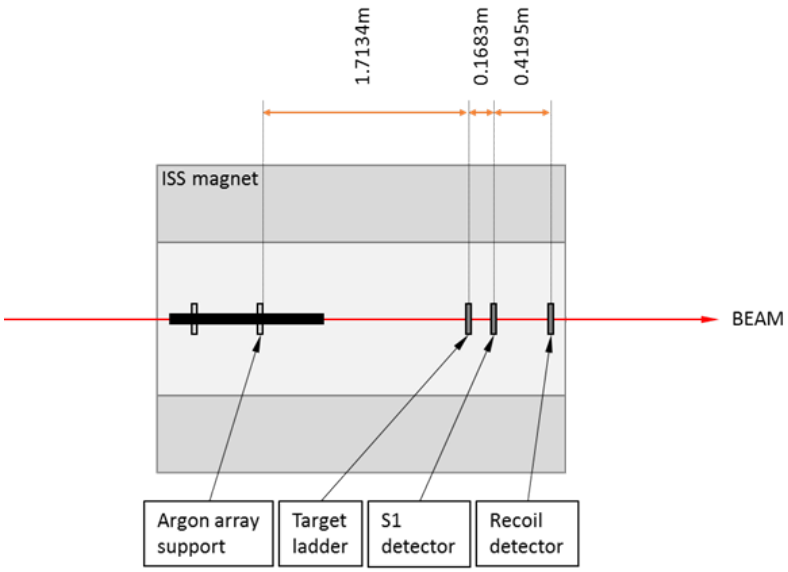

Figure 7: The measured relative distances given with a precision of 0.5 mm at 1 σ level [3]

Figure 6 shows the objects whose relative distances were determined during the alignment that was carried out by the survey team on August 30th, 2018. With the help of a survey theodolite that has been installed close to the beam line, downstream of the setup, the detectors inside the ISS magnet have been optically aligned. The longitudinal positions have been measured by observing a retro reflective target placed on the objects of interest. The relative distances can be found in figure 7 which are given with a precision of 0.5 mm at 1 σ level. All measurements were done against the beam direction. Because of the width of the objects and the measuring cross further calculations has to be done to determine the relative distances of interest. The required lengths which were measured with a caliper are in table 3.

	Length [mm]
Measuring cross	
Cross width	0.75
Cross holder width	8.35
Cross holder+cross	9.10
S1 detector	
Distance from back of S1 detector to its active surface	33.5
Recoil detector	
Distance to dE detector	32.225
Distance between dE and E layer	14.28
Width of layer	2.21
Distance between E, dE surface	12.07
Silicon array	
Distance from measuring point to the end of the array	748.5
Distance from end of array to edge of silicon detector	71.53

Table 2: Offsets

From these values the following distances of interest can be calculated:

	Distance [mm]
Target - Edge of Si Array	1713.4+0.75-748.5+71.53=1037.18
Target - S1	168.3-33.5-9.1=125.7
Target - Recoil(dE)	168.3+419.5-(32.225+9.1)=546.475

Table 3: Relative differences

The measurements were done at the following encoder positions:

- **Axis 1:** -101708
- **Axis 2:** 35115
- **Axis 3:** -5751
- **Axis 4:** 600

This can be used to calculate the distance between the edge of the silicon array and the target for given encoder positions $p_{1,enc.}$ and $p_{2,enc.}$:

$$d_{2,3} = 1037.18 + (-101708 - p_{1,enc.} + p_{2,enc.} - 35115)/200. \tag{1.1}$$

1.2.2 Position measurement on axis 3 (target ladder)

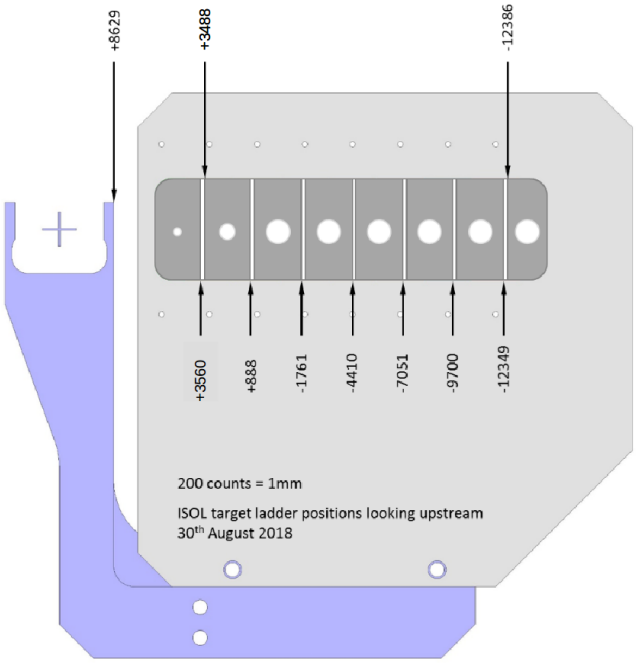

Figure 8: Measurement of the target ladder [3]

Figure 8 shows the measured encoder positions. Apparently, the width of one target holder is

$$(3488 - 888) \text{ steps} = 2600 \text{ steps} = 13 \text{ mm}.$$

Supposing that all target holders have the same width, the center of each target position can be calculated by adding (or subtracting in case of position 8) half of the width to the measured values. Furthermore, the position of the center of the alpha source (blue cross on the left) can be calculated by knowing that the distance from the measured edge of the holder until the center is 16 mm. The obtained values can be found in table 4.

Position	Encoder position of the center	
	[steps]	[mm]
α Source	11829	59.145
1	4860	24.300
2	2188	10.940
3	-461	-2.305
4	-3110	-15.550
5	-5751	-28.755
6	-8400	-42.000
7	-11049	-55.245
8	-13686	-68.430

Table 4: Measured target positions

1.2.3 Position measurement on axis 4 (beam diagnostics)

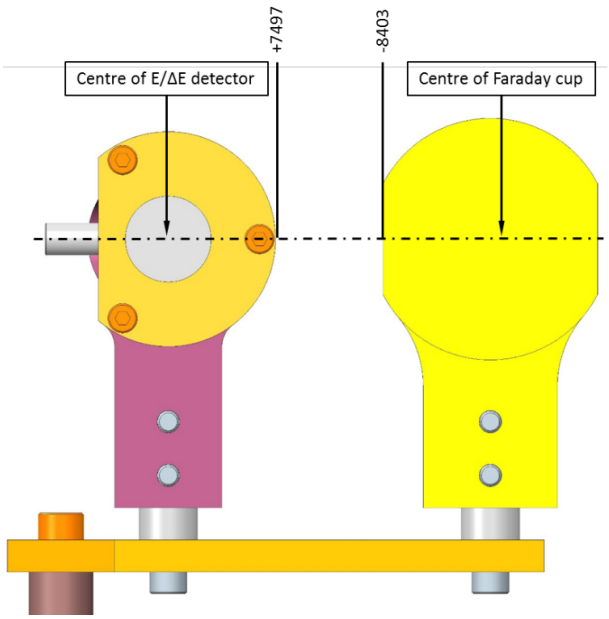

Figure 9: Encoder positions of the edges of the Faraday cup and the zero degree dE/E detector [3]

On axis 4, the encoder positions of the edges of the Faraday cup and the zero degree dE/E detector were measured (see figure 9). By knowing the distances to the center, the corresponding encoder position can be calculated. Table 5 shows the results.

Object	Position of edge [steps]	Distance to center [mm]	Position of center [steps]	Position of center [mm]
zero degree dE/E detector	7497	20	11497	57.485
Faraday cup	-8403	21	-12603	-63.015

Table 5: Measurement of the beam diagnostics

Using these results, the position of the center between Faraday cup and zero degree dE/E detector is calculated to

$$-8403 + \frac{1}{2}(7497 + 8403) = -553 = -2.765 \text{ mm}.$$

1.3 The GUI's design

The GUI is written in Python using the wrapper wxPython. It consists of the wx.Frame *DriveSystemGUI* which is split in two wx.Panels: the upper panel *ControlView* which allows the user to send commands to the drive system and the lower panel *MatplotPanel* which includes the Matplotlib figure *DriveView*. The panel *ControlView* starts the thread *CheckPositions* (see section 1.4) and is owner of the object *DriveSystem* which is responsible for connecting to the drive system and sending commands to it. A screenshot of the whole graphical user interface is shown by figure 10.

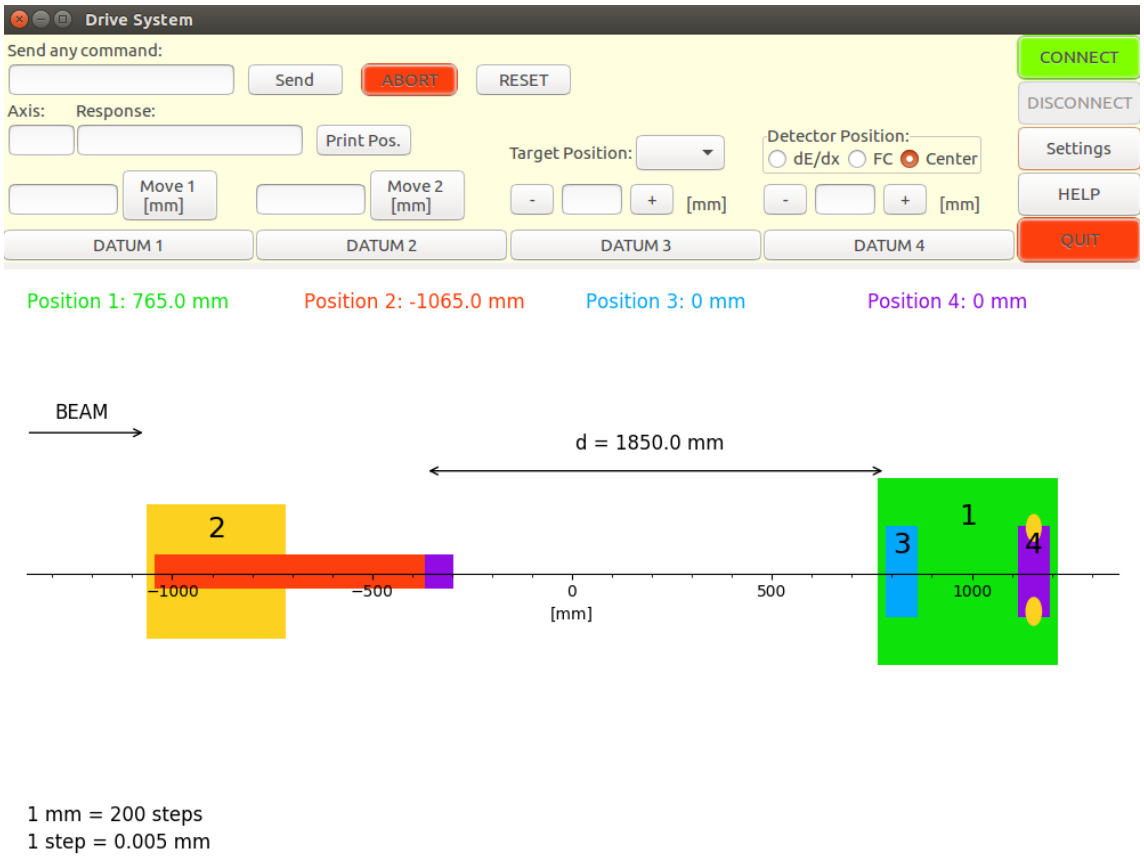

Figure 10: Screenshot of the GUI for controlling the drive system

1.3.1 The Matplot figure *DriveView*

The Matplotlib figure *DriveView* shows the positions of the four different objects in a graphical way. They are represented by colored rectangles whose positions change according to the encoders' positions which are checked every second. Moreover, it shows the position of the Faraday cup and the zero degree dE/E detector on axis 4 (yellow ovals) and the distance between the end of the array and and the edge of the silicon detector which is represented by the purple rectangular on the red rectangular. The coordinate system is in millimeters and centered around the middle of the magnet which has a length of 2730mm which determines the limits to be -1365 and +1365 mm. At the top of the figure, the distance from the datum position in millimeters is displayed. In case of axes 3 and 4 this is the equal to the encoder position, in case of axes 1 and 2 the encoder position is multiplied by -1. Furthermore, it shows the beam direction and the distance between the silicon array and the target calculated with formula 1.1.
From a further measurement, it is known that the end of the array is at a distance of 1150 mm to the wall of the magnet at the encoder position of 35129. That means that the position of the end of the array in the GUI's coordinate system can be calculated by

$$P_2 = -L_{mag}/2 + 1150 + (35129 - p_{2,enc.})/200 \text{ with } L_{mag} = 2730. \tag{1.2}$$

The distance between the edge of the silicon array and the target $d_{2,3}$ is calculated with formula 1.1 and the position of the target therefore

$$P_3 = P_2 - 71.01 + d_{2,3}. \tag{1.3}$$

The position of the trolley is then estimated with a 2 cm distance from the target and the Faraday cup with a 2 cm distance from the edge of the trolley. This gives:

$$P_1 = P_3 - 20 \text{ and} \tag{1.4}$$

$$P_4 = P_1 + L_1 - 20 \tag{1.5}$$

with L_1 being the length of the trolley which was measured with a ruler to approximately 45 cm.

1.3.2 The panel for controlling the drive system

In the upper part of the GUI, the following functions for controlling the drive system are offered:

- **Send any command:** In the top left corner of the GUI, there is the opportunity to send any kind of command to the drive system (all available commands can be found in the drive System's handbook [1] from page 69 onwards).
- **Abort and reset:** Clicking the abort button will abort the command on all axes, any new command will be blocked. The reset button cancel this blocking and commands can be sent again.
- **Print positions:** If the *Print Pos.* is pressed, the encoder positions will be printed in the terminal.
- **Change position on axis 1 and 2:** The *Move 1* and *Move 2* buttons are thought to move the objects relatively according to the GUI's coordinate system. Since the motor on axis 1 is currently defect, an error message will be printed.
- **Choose a target position:** The user can choose between eight different target positions and the position of the α-source from a list. By choosing one position, the motor on axis 3 will move the target ladder to the corresponding position.
- **Choose a position on axis 4:** A selection between the Faraday cup, the zero degree dE/E detector and their center is offered. When one of the items is selected, the motor on axis 4 will move the corresponding item into the beam line.
- **Change position on axis 3 and 4:** By entering a distance in millimeter and clicking on the + or - button, the motors will move the attached objects on axis 3 or 4 by the entered distance.
- **Change saved positions on axis 3 and 4:** The corresponding positions of the target positions and the positions of the beam diagnostics can be changed via the window *Settings*. By clicking on the *Ok* Button, the new positions will be saved to a file. When the GUI is closed and opened again these positions will be read in from the file.
- **Connect/Disconnect to/from the port:** The GUI connects automatically to the port when opened. The connect button is then disabled until the disconnect button is pressed.
- **Help:** A window with useful information about the GUI is opened by the *Help* button.
- **Quit:** The GUI will be closed when this button is pressed.
- **Datum search:** The datum/zero position on the axis will be searched when clicking the corresponding button. When the datum position is found, this position can be set to zero with the two commands OD (output datum position) and DA-x, with x being the datum position. This is also described in the *Help* window.

Except of the *Help* and *Settings* button (and *Connect* and *Quit* if the port is closed), all these actions will not be executed immediately after clicking the button, but are handled by the thread *CheckPositions*.

1.4 The thread *CheckPositions*

The thread which is called by the panel *ControlView* handles the communication with the drive System. The following pseudocode illustrates the its way of working:

```
while True:
    if driveSystem.portOpen == True and CommandsAborted==False:
        driveSystem.checkEncoderPositions()
        maptlotFigure.updatePositions()
        t=0
        while t<UPDATE_TIME:
            CheckQueue() #queue of clicked buttons
            time.sleep(0.1)
            t=t+0.1
    else:
        time.sleep(1)
```

The thread starts always with checking the encoder positions and updating them in the matplotlib figure. After that it checks every 0.1 seconds if any button was clicked and if so, sends the corresponding command to the drive system until a specific amount of time passed. This time is 1 second, and can be changed via the variable UPDATE_TIME.

1.5 Conclusions

The GUI for controlling the drive system is working stable and reliable. All basic movements which might be necessary while performing an experiment can be carried out by clicking on the corresponding button. If any other command is needed, it can be sent easily via the interface.

2 A GUI to control the Mesytec MHV-4 bias supply units

2.1 Technical Details

The necessary values for connecting to the unit can be found in table 6.

Baudrate	9600
Parity	odd
Bytesize	8 Bits
Stop bit	1
Timeout	1 s

Table 6: Properties of the communication with the MHV-4 unit [2]

2.2 Functions offered by the GUI

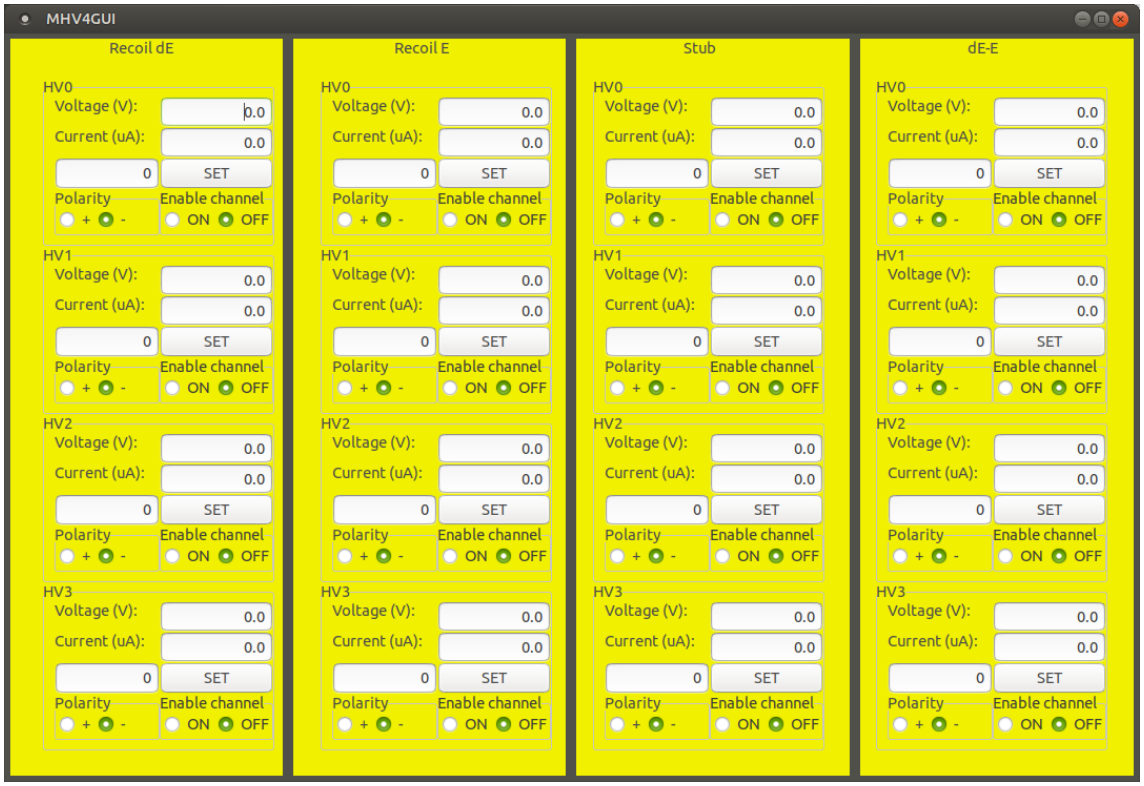

Figure 11: A screenshot of the GUI for controlling the MHV-4 units

Figure 11 shows a screenshot of the GUI. As the GUI for controlling the drive system, it is written in Python, using the library pySerial and the wrapper wx.Python. The actions which can be carried out by this interface are:

- Turn a channel on: The channel will be turned on and a voltage of 0.1 V will be set (since channels with a voltage of zero are considered to be turned off).
- Turn a channel off: The voltage will be ramped down to zero before turning the channel off.
- Change the polarity of a channel
- Set a new voltage by inserting the desired value in the text field and click on the SET button. The voltage will then be changed with a velocity of 0.5 V/s. This is achieved by changing the voltage about 1 V and then wait for 2 s.

Furthermore, the voltages and currents of all channels are read out every 3 seconds and are then updated in the GUI. Therefore, to prevent that two commands are sent at the same time to the unit, a thread is needed which handles the different requested actions.

2.3 The thread *CheckAndUpdater*

If a button is clicked in the GUI, the action will not be carried out immediately, but is appended to a queue. There is one queue for voltage changes and one for changing the polarity and turning a channel on. Every unit has its own thread *CheckAndUpdater* which checks these two queues, waits for 2 seconds (RAMP_WAIT_TIME=2) and then checks if already 3 seconds have passed since the last update (UPDATE_TIME=3). This is illustrated by the following pseudo code:

```
while True:
    if Vqueue.isEmpty()==False:       #Check if any voltage change was requested
        changeVoltage()
    if Pqueue.isEmpty()==False:       # Check if any polarity change or enabling a channel
        changePolarities()              # was requested
        enableChannel()
    time.sleep(RAMP_WAIT_TIME)      # The GUI sleeps for 2 s (time between two voltage
    updateCounter=updateCounter+RAMP_WAIT_TIME      # changes)
    if updateCounter>=UPDATE_TIME:      # Check if more then 3 s passed since the last
        updateCounter=0                   # Update
        readCurrentsAndVoltages()
        updateGUI()
```

2.4 Safety features

The following safetey checks are implemented in the GUI:

- A channel can be turned ON only if the voltage is zero.
- If a channel is turned OFF, an element is put to the voltage queue and the wanted value for the voltage is set to zero.
- The polarity can be changed only if the channel is OFF.
- When a channel is turned ON, the voltage is set to 0.1 V. When starting the GUI, all channels with a voltage of zero are turned OFF. This assures that a channel is turned OFF if its voltage is zero and ON if it is not equal to zero. This was implemented this way, because there is no way to ask a channel directly whether it is ON or OFF.
- Preset voltages are set to zero when starting the GUI.
- Ramping to a certain voltage is only continued if the voltage is higher than zero (channel is defintely turned ON).
- Every three seconds the current voltages and currents are updated in the GUI.
- The maximal voltage which can be applied is 100 V. If a higher voltage is requested, the GUI sends a message saying that the voltage is higher than the voltage limit.

2.5 Conclusions

The GUI can be used to bias the recoil detector as well as the S1 detector and the zero degree dE/E detector. The GUI must then not block the bus, because in this case the only thing one can do is pulling out the unit and put it back in again. This could damage the connected detectors. When the GUI was used during the setup of ISS, it was observed once, that the bus was blocked after turning all four channels of a unit on. Since there is only one thread which sends commands to the unit, two commands are never sent at once in theory. Nevertheless, the problem could have been that two commands were sent too close to each other. To prevent this, the command *time.sleep(0.1)* was inserted after every sent command. In the case that the bus is blocked again in the future, some further debugging should be done.

3 Conclusions and acknowledgements

With the two built interfaces it is possible to control the drive system and to bias the used detectors easily while carrying out an experiment. Both interfaces were already used for successful experiments in September (IS621) and October (IS631) 2018. The whole source code can be found under the links of reference [4] and [5].

A special thanks goes to my supervisors Liam and Joonas. They motivated me with their great enthusiasm for the experiment and by placing their trust in me and my work. Particularly, I am very happy to have been also given the opportunity to work in the laboratory where I learned a lot thanks to their explanations. Furthermore, I want to thank all the members of ISOLDE who made me feel like part of the group, especially since they gave all summer students the chance to present their project in one of the weekly meetings which was a great experience.

References

[1] *PM600 Motion Controller USER Manual*
Mclennan Servo Supplies Ltd,
V5.19a (2009)

[2] *mesytec MHV-4 datasheet*
mesytec GmbH & Co. KG,
V4.0_02
https://www.mesytec.com/products/datasheets/MHV-4.pdf

[3] *HIE-ISOLDE ALIGNMENT OF ISS MAGNET DETECTORS ON XT02 BEAM LINE*
Benoit CUMER, Alexandre BEYNEL
https://edms.cern.ch/document/2021548

[4] Source code of the GUI for controlling the drive system:
https://github.com/ISOLDESolenoidalSpectrometer/DriveSystemGUI

[5] Source code of the GUI for controlling the Mesytec MHV-4 bias supply units:
https://github.com/ISOLDESolenoidalSpectrometer/VoltageGUI