

Magnet Mapper Controls

Josh Kraan

TRIUMF

Abstract: Two new user-friendly control systems for TRIUMF's new magnet mapper are detailed. The Python control system is best for local control, and the EPICS implementation allows mapping to be monitored and controlled remotely.

1 Introduction

A new magnet mapper system was designed and built for TRIUMF by a UBC capstone team in April 2022 [1]. A more user-friendly control system was desired, the design of which this report will detail.

Two similar control systems with different strengths were created with Python and EPICS. Ideally these would be combined into one product, however this was not feasible within the time constraints of the development.

The initial Python implementation is the most fully-featured and easy to use. The later EPICS version has fewer features and ease-of-use but allows for mapping to be remotely monitored and controlled. Both versions will be able to provide the same output data.

2 Common Concepts

2.1 Frames

The vertical stage of the mapper is mounted in an inverted configuration. This means that commands sent to the Zaber stages are resolved in the “Zaber” frame, shown in Figure 1a.

To produce a proper right-handed coordinate system, a new “Mapper” frame was defined in software, shown in Figure 1b. User-configurable offsets were added so that the center of a magnet can be set to $(0, 0, 0)$ in this frame. Note that the offsets are defined relative to the Zaber frame to aid in alignment, which can cause some confusion for the vertical axis.

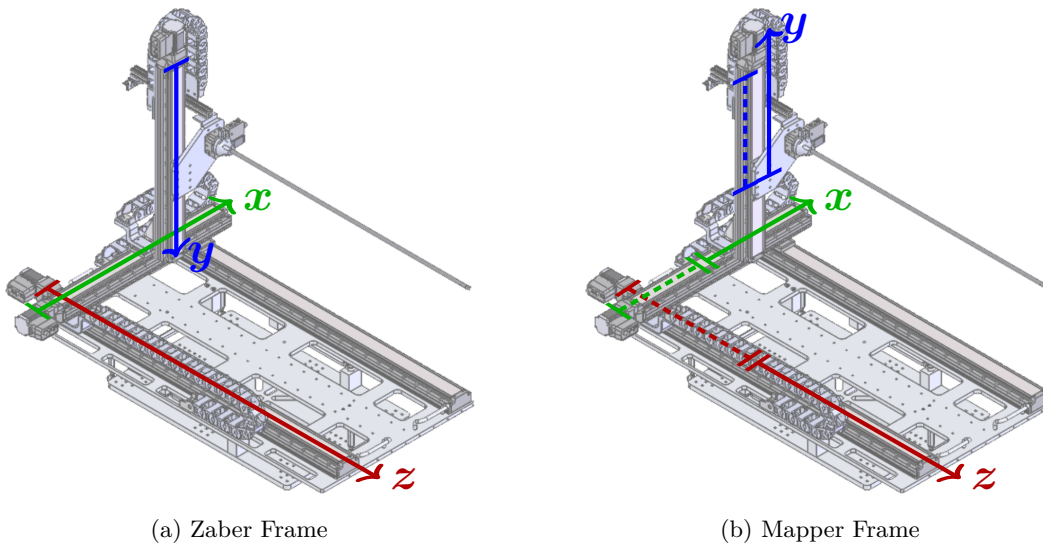


Figure 1: Diagrams of the “Zaber” and “Mapper” frames overlaid on a CAD model of the mapper [1]. For each axis the bar indicates the zero position, and the arrow indicates the positive direction. Axis offsets in the Mapper frame are shown with dashed lines.

2.2 Mapping Modes

All automated mapping is done in the Mapper frame.

2.2.1 Rectangular

In rectangular mode the probe is moved through a 3D rectangular grid. Multiple angles for the rotation stage can be specified for which the grid will be repeated.

The grid is defined by a range and spacing for each axis. The range is the maximum amount of displacement of a given axis. The maximum amount of points centered around and including 0 with the specified spacing that fit within the range are used. For example, a range of 20 and a spacing of 10 for an axis would result in the visited points -10, 0, 10, but a range of 30 would result in the same. Only when the range is increased to 40 would the visited points change to -20, -10, 0, 10, 20.

2.2.2 Cylinder

Cylinder mode is essentially the intersection of a 3D rectangular grid with a cylindrical bounding box. The cylinder axis is in the z direction, so the grid along z is defined identically to rectangular mode. The grid on the x - y plane is defined by the “ x - y spacing” and “radius” parameters, and is calculated as the points for a grid centered around and including $x, y = (0, 0)$ with the given spacing that fit within the radius.

Like rectangular mode, rotation stage angles for which the grid will be repeated can be specified.

2.2.3 2D Custom

In 2D custom mode a list of coordinates in x, y , and rotation are repeated at z locations specified by a range and spacing, which are defined the same as in rectangular mode.

2.2.4 3D Custom

In 3D custom mode a full list of coordinates to visit in x, y, z , and rotation is input.

2.3 Mapping Procedure

The following general procedure is recommended for mapping a magnet:

- Before placing the mapper near a magnet, home all axes.
- Zero the hall probe with a zero gauss chamber. Ensure that the Teslameter is set to measure the magnetic field in Gauss.
- Manually move the axes until the hall probe tip is at the desired origin of the magnet. For each linear stage, read off the position in the Zaber frame and set the axis offset to be the same. All linear axis positions should now be 0 in the Mapper frame.
- Manually move the mapper through the largest extents of the desired map, ensuring that all is within axis limits.
- Input the mapping information into the control software. Preview the generated coordinates to ensure they match what is desired.
- Begin mapping.

3 Python Implementation

3.1 Architecture

The Python-based mapper controller is run on a local computer that is directly connected to the mapper system, as shown in Figure 2. The Zaber system and the Teslameter are connected by USB to RS232 adapters.

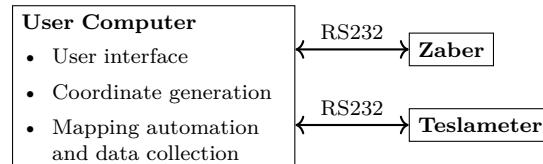


Figure 2: Architecture of the Python implementation of the mapper controller.

Identifying the serial port assigned to each USB to RS232 adapter is necessary to start the Python software. An example of finding the assigned serial ports with the Windows Device Manager is shown in Figure 3.

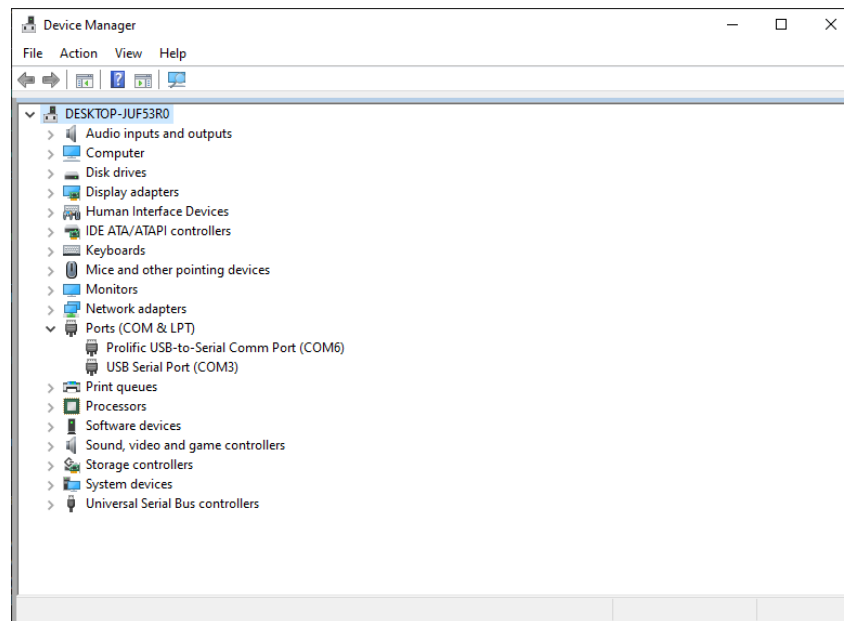


Figure 3: Example of identifying the correct serial ports using the Device Manager on Windows. The Teslameter uses a Prolific adapter; in this example it is assigned to the port COM6. The only other serial device here is the Zaber system on COM3.

3.2 High-Level Classes

The task of mapping a magnet was split into the subtasks of generating the coordinates to map, controlling the motion system, reading the field from the Teslameter, and saving the data as it is gathered. A high-level class was created for each subtask.

An example of using all four classes together to map a magnet is provided in Listing 1. For details on each class, see their docstrings.

```

1 from mapping import Mapping
2 from motion import Motion
3 from teslameter import Teslameter
4 from output import Output
5
6 mapping = Mapping()
7 motion = Motion("COM3")
8 teslameter = Teslameter("COM6")
9 output = Output()
10
11 mapping.shape = "rectangular"
12 mapping.range["x"] = 200
13 mapping.range["y"] = 0 # Only mapping on x-z plane
14 mapping.range["z"] = 100
15 mapping.spacing["x"] = 10
16 mapping.spacing["z"] = 10
17
18 mapping.settling_time = 5
19
20 output.create_file()
21
22 for coords in mapping.point_generator():
23     motion.move_mapper(*coords, mapping.settling_time)
24     reading, unit = teslameter.read()
25     output.write_line(*coords, reading, unit)

```

Listing 1: Example usage of the high-level classes.

The CSV file for the custom option should be formatted as shown in Listing 2, and saved as a text file with any extension (*.csv, *.txt, *.dat, etc.).

```

1 X (mm), Y (mm), R (deg)
2 0.00,-7.60,0
3 0.00,-5.60,0
4 -2.80,-4.85,30
5 -4.85,-2.80,60
6 -5.60,0.00,90
7 -4.85,2.80,120
8 -2.80,4.85,150
9 0.00,5.60,180
10 2.80,4.85,210
11 4.85,2.80,240
12 5.60,0.00,270
13 4.85,-2.80,300
14 2.80,-4.85,330

```

Listing 2: Example CSV file for the custom option. Note that the first line is skipped, so a header or placeholder must be included.

3.3 User Interface

The user interface for the Python-based mapper controller can be started by running `main.py`. Note that the serial ports assigned to the Teslameter and the Zaber system should be set at the beginning of this file, as shown in Figure 4. The Teslameter and Zaber system must be connected for the user interface to start.

```

11 # Comm ports:
12
13 ZABER_COMM_PORT = "COM3"
14 TESLAMETER_COMM_PORT = "COM6"

```

Figure 4: Serial port assignment in `main.py`. See Figure 3 for information on identifying the correct ports.

The screenshot shows the 'Mapper Controller' application window. It is divided into three main sections:

- Motion:** A table with columns for Stage, Mapper Position, Offset, Zaber Position, Home, and Warning Flags. The rows are for X, Y, Z, and R axes. Each cell contains a numerical value, a unit (mm or degrees), and a 'Home' button.
- Mapping Configuration:** A section with various input fields for mapping parameters: Settling Time (5.00 s), Z Range (100.00 mm), Z Spacing (5.00 mm), Shape Type (Rectangular), Y Range (10.00 mm), Y Spacing (5.00 mm), X Range (10.00 mm), X Spacing (5.00 mm), and Rotation Points (0.00, 90.00). A 'Preview Coordinates' button is present, along with a status message: 'With 378 points, estimated mapping time is 44 minutes.'
- Output Configuration:** A section with fields for Output Folder (C:/Users/MagMeasure/Desktop), Magnet Name (magnet), and Output File (C:/Users/MagMeasure/Desktop/magnet_2022-08-22T17-31.csv). A 'Browse' button is next to the Output Folder field.

At the bottom of the window is a large 'Begin Mapping' button.

Figure 5: Main screen of the Python interface. Positions in the “Motion” section can be set by the user and are also updated based on the live system positions. Selecting a different “Shape Type” than shown here will reveal different options in the “Mapping Configuration” section. User inputs are sent to the backend when the user presses the Enter key or changes field focus. When this main screen is closed, a stop command is sent to all axes.

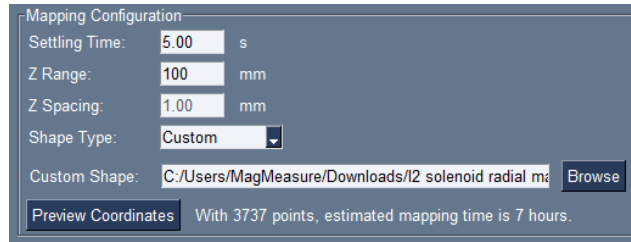


Figure 6: Example of the Mapping Configuration section with the Custom shape type selected. For information on formatting the input file see Listing 2.

	X	Y	Z	R
0	-5	-5	-50	0
1	-5	-5	-45	0
2	-5	-5	-40	0
3	-5	-5	-35	0
4	-5	-5	-30	0
5	-5	-5	-25	0
6	-5	-5	-20	0
7	-5	-5	-15	0
8	-5	-5	-10	0
9	-5	-5	-5	0
10	-5	-5	0	0
11	-5	-5	5	0
12	-5	-5	10	0
13	-5	-5	15	0
14	-5	-5	20	0
15	-5	-5	25	0
16	-5	-5	30	0
17	-5	-5	35	0
18	-5	-5	40	0
19	-5	-5	45	0
20	-5	-5	50	0
21	-5	0	50	0
22	-5	0	45	0
23	-5	0	40	0
24	-5	0	35	0
25	-5	0	30	0
26	-5	0	25	0
27	-5	0	20	0
28	-5	0	15	0
29	-5	0	10	0
30	-5	0	5	0

Figure 7: Coordinate preview screen of the Python interface, launched by the “Preview Coordinates” option in the main screen.

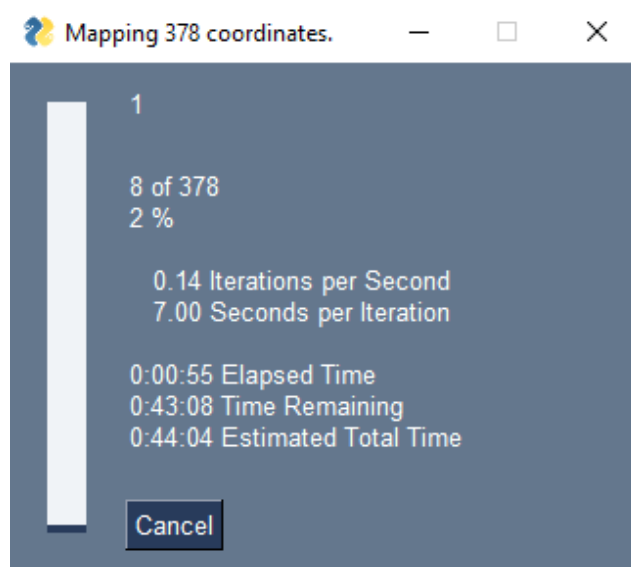


Figure 8: Mapping progress screen of the Python interface. Iteration time is recorded, so the estimate will get better over time.

4 EPICS Implementation

For further details on the EPICS implementation see Appendix A.

4.1 Architecture

The architecture of the EPICS controller is shown in Figure 9. Two additional layers over the Python implementation are present between the user computer and the mapper. The user remotely accesses the EDM server via SSH, and the server communicates with the IOC via channel access. The IOC is directly connected to the Zaber system and the Teslameter.

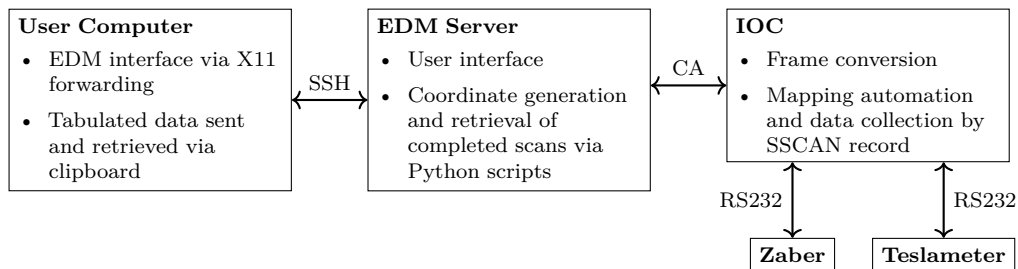


Figure 9: Architecture of the EPICS implementation of the mapper controller.

4.2 Channel Access

Offsets and stage positions in the mapper frame can be set via the Channel Access protocol, and the current reading of the hall probe reported by the Teslameter can be read. This can be useful if scan automation by the SSCAN record is not desired, such as if a high-level web application for mapping magnets was created. Example usage is shown in Listing 3.


```

1 # Note that the following applies to all axes: replace 'X' with 'Y', 'Z', or 'R'
2 caget MAPPER:X:POS_MAPPER # Read position in Mapper frame, in mm
3 caget MAPPER:X:POS_ZABER # Read position in Zaber frame, in mm
4 caget MAPPER:X:WARNINGS # Read warning codes active on this axis
5 caput MAPPER:X:OFFSET 250 # Set offset to 250mm
6 caput MAPPER:X:SET_POS_MAPPER 0 # Move to the given position in the Mapper frame
7 caput MAPPER:X:HOME 1 # Home axis
8
9 # Reading the current reading of the Teslameter, in gauss
10 caget MAPPER:TESLAMETER:FIELD

```

Listing 3: Examples of mapper control with Channel Access. Note that the `MAPPER:` prefix used for the record names may be changed in the future.

4.3 EDM

The following screenshots show the EDM user interface. Note that user input fields in the EDM interface will not be updated until the Enter key is pressed.

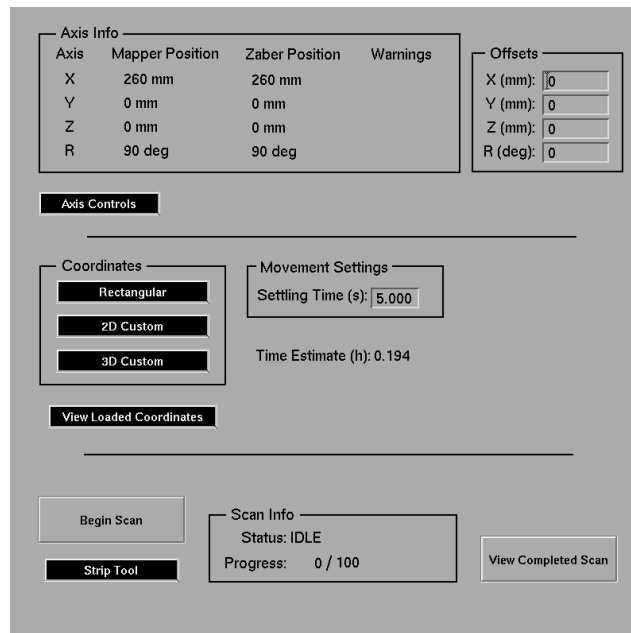


Figure 10: Main screen of the EDM interface.

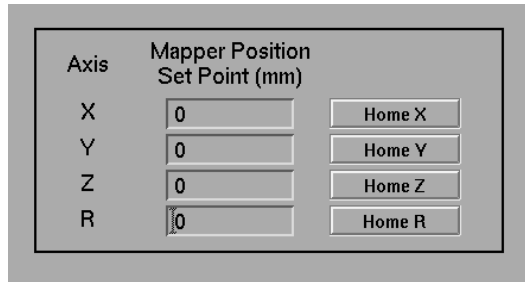


Figure 11: Axis Controls screen. Note that the displayed set points are not updated when axes are manually moved outside of EPICS.

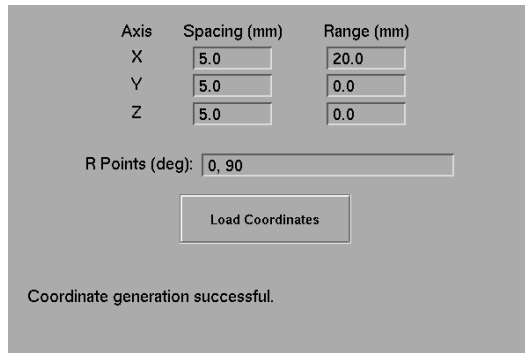


Figure 12: Rectangular coordinates screen. Rotation points should be entered as a comma-separated list of angles in degrees, as shown. Press “Load Coordinates” to load the specified settings into the SSCAN record.

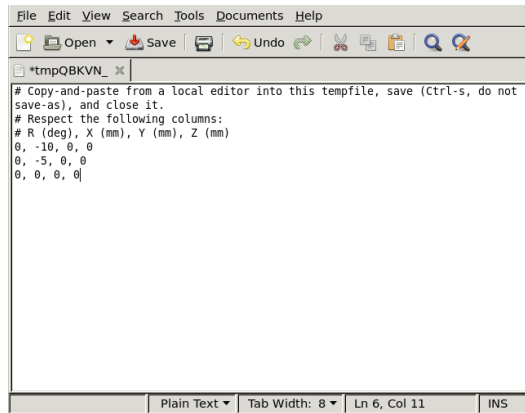
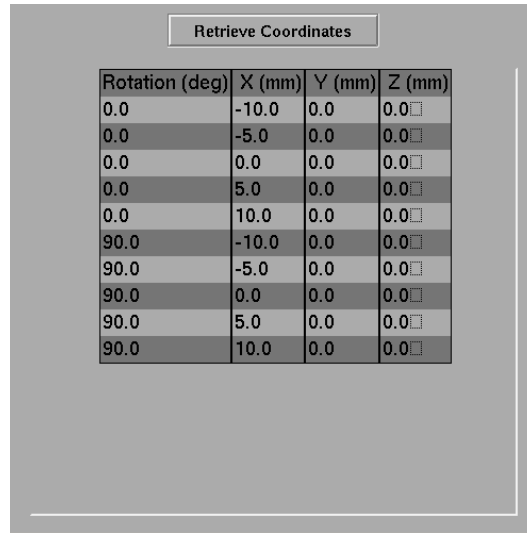


Figure 13: The Gedit window opened by the “3D Custom” option. This editor is run on the remote EDM server (see Section 4.1), and so it does not have access to local files. However, it does have access to the local clipboard, which can be used to transfer large CSVs.



The image shows a window titled "Retrieve Coordinates" containing a table with the following data:

Rotation (deg)	X (mm)	Y (mm)	Z (mm)
0.0	-10.0	0.0	0.0
0.0	-5.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	5.0	0.0	0.0
0.0	10.0	0.0	0.0
90.0	-10.0	0.0	0.0
90.0	-5.0	0.0	0.0
90.0	0.0	0.0	0.0
90.0	5.0	0.0	0.0
90.0	10.0	0.0	0.0

Figure 14: Preview of the coordinates that the mapper will map. These coordinates will not be updated until the “Retrieve Coordinates” button is pressed.

5 Conclusion

Both control systems shown here are operational and should be able to provide the same data. It is recommended that the Python implementation be used when the remote operation capability of the EPICS implementation is not needed.

A EPICS Implementation Details

A.1 Python Scripts

The following scripts are used by the EDM interface:

- `epics_tools.py` is essentially a Python wrapper of the command line tools `caget` and `caput`, and is imported into the other scripts.
- `gen_arrays.py` generates arrays of coordinates and inputs them into the SSCAN record depending on the command line arguments supplied. If a “Custom” mode is used, a new Gedit window is opened up, into which the user can paste a CSV file.
- `preview_coords.py` reads the coordinates currently loaded in the SSCAN record, formats them into a table, and writes the table to a temporary file to be read by the EDM “Table” widget.
- `scan_results.py` reads the scan results from the SSCAN record, formats them into a table, and writes them into a temporary file which is opened by Gedit. The user can copy the scan results from the Gedit window to their local computer.

A.2 Record Name Prefix

The record name prefix was left as a macro in the TDCT schematic and the EDM interface. In the Python scripts, it was defined in the beginning of each script that requires it as the

variable `MAPPER_PREFIX`.

The TDCT schematic can be compiled with macros with the command `java -jar /usr1/local/tdct/tdct.jar -s capfast/mapper.sch -db db/ -m MAPPER_PREFIX=MAPPER`.

The EDM interface can be launched with macros with the command `/usr1/extensions_bin/edm -eolc -x -m "MAPPER_PREFIX=MAPPER"`.

A.3 Misc

- The user running the IOC must be a member of the dialout group.
- Place the included udev rules `99-mapper.rules` in `/lib/udev/rules.d/` to ensure that serial ports are mapped correctly to `/dev/zaber` and `/dev/teslameter`.
- Be careful with Zaber device numbers, they are automatically renamed if additional devices are added in the daisy chain.
- `EPICS_CA_MAX_ARRAY_BYTES` should be set to 1000000, allowing for 125,000 data points to be collected.
- EDM shell button paths need to be updated if Python script or data files are relocated.
- `MAPPER_COORD_SCRIPT_DIR` should be set appropriately in `use_mapper.bash`.
- The current ASLO factor of $4.961e-4$ is used to convert Zaber commands from mm to microsteps. Note that this assumes the resolution setting is at its default value, giving a microstep size of 0.496 microns.
- The Python scripts were written to be compatible with both Python 2 and Python 3. May need to install `python-is-python3` or modify shebangs in Python files for Python3 compatibility.

B Source Code Location

Code for the Python implementation is located at <https://gitlab.triumf.ca/hla/student-projects/magnet-mapper-control>.

Code for the EPICS implementation is located at <https://gitlab.triumf.ca/hla/student-projects/magnet-mapper-control-epics>.

References

- [1] B. Brown, A. Kassab, J. Kraan, and A. Xiong, *16T Magnet Mapper*, Apr. 2022. [Online]. Available: <https://documents.triumf.ca/docushare/dsweb/Get/Document-222513/2204%20-%2016T%20Magnet%20Mapper%20-%20Final%20Report.pdf>.