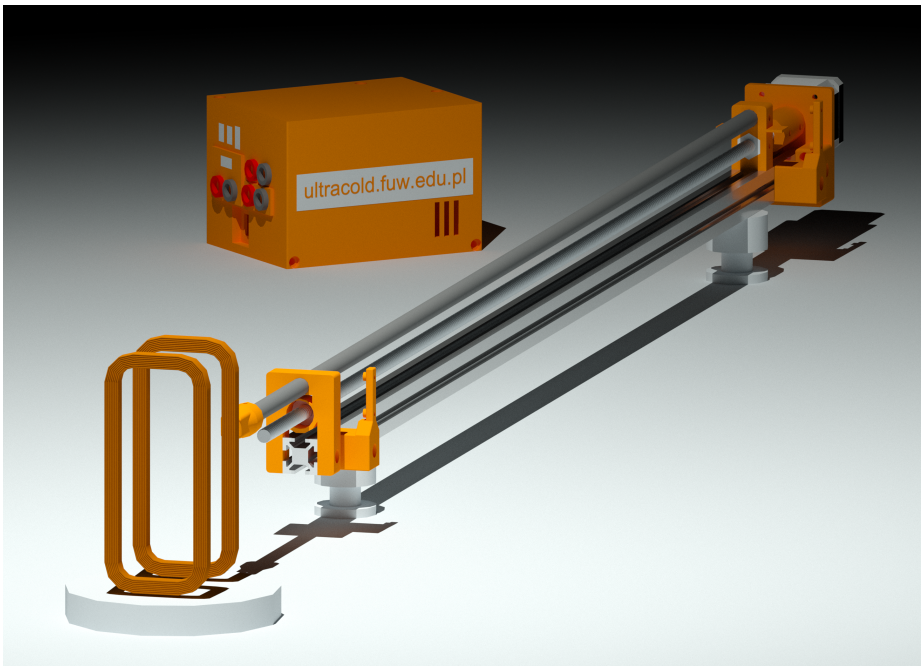
User Manual

Automated Magnetometer



Quantum Gases Laboratory
University of Warsaw

Koray Dinçer
2023

Contents

About

This document is prepared for the "Open Source Automated Magnetometer" designed to characterize magnetic field sources. This manual provides detailed instructions on how to effectively utilize our system designed specifically for experiments in ultracold physics. The Automated Magnetometer combines a high-precision Hall sensor and a stepper motor to enable precise characterization of magnetic field sources along a one-dimensional range. With this system, one can accurately measure and analyze magnetic fields, making it an essential tool for characterization of permanent and electromagnets. This manual aims to guide users through the installation, setup and operation while ensuring optimal performance and reliable results.

All necessary files can be found at https://github.com/uqglab/automag.

1 Hardware

The choice of hardware solutions has been driven by the desire to make the entire magnetometer cheap and easy to assemble while providing high performance in terms of achievable noise and resolution of the magnetic field measurement. It is common in academic setting that many tasks needed to build a new experiment are performed by inexperience students (e.g. undergraduate students just entering the field of research). With this in mind we have taken care to make our design easy to reproduce. The mechanical assembly uses easily available commercial components like T-slotted aluminum profiles or threaded steel rods and 3D printed parts that can be made in-house.

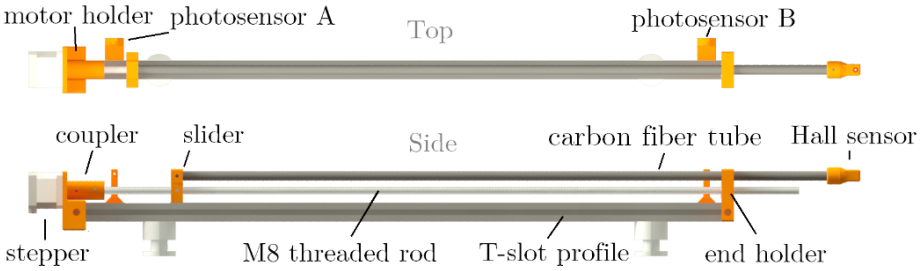


Figure 1: The layout of the automated magnetometer as seen from the top and from the side. Orange-colored parts are 3D printed. The rotation of the stepper motor shaft is coupled to the threaded rod via a coupler. A M8 nut is embedded in the slider to convert the rotation of the rod into the translational motion of the carbon fiber tube which supports the Hall sensor.

The aluminum profile is the "backbone" of the setup and everything is attached to it (Fig. 1) but neither the profile nor the details of the remaining components need to follow our recommendations; as long as the general approach we present is followed the remaining hardware and software will be fully usable. Below is a list of the hardware components included:

1x T-slot profile (20 mm×20 mm),
1x Carbon fiber tube,
1x Threaded rod, M8,
6x 3D Printed Parts,
1x Arduino Mega,
1x MMC5983MA (Qwiic),
1x L298 - Dual H-Bridge motor driver,
1x Stepper motor (JK42HM48-1684 or similar),
2x WSR-12225,
1x ST1168.

2 Assembly

2.1 Installation of mechanical parts

Installation of the mechanical parts are fairly simple by design. Please follow the below guide to finalize the assembly of mechanical parts. The list of specific screws and nuts with their quantities:

M3 screws: 2x 8mm (Plastic), 4x 8mm, 4x 5mm,
M4 screws: 8x 6mm (set-screw), 7x 10mm,
M4 T-nuts: 7x T-nut,
M8 nuts: 3x nut.

1. Position the motor-holder at the desired end of the T-slot aluminum profile. The motor-holder has with three screw holes: two on the sides and one at the bottom. Secure the motor-holder to the T-slot aluminum profile using three 10 mm M4 screws and three M4 T-nuts. Ensure that the motor-holder is tightly fastened for proper stability and support.

2. Place the stepper motor into the motor-holder, ensuring that the motor shaft aligns with the central hole of the motor holder. Use four 8 mm M3 screws to securely fasten the stepper motor to the motor-holder. Take care to align the stepper motor's rod at the center of the motor-holder's central hole. By aligning the motor shaft at the center of the central hole, you will achieve optimal positioning and stability for the stepper motor.

3. Insert the coupler onto the stepper's rod, ensuring that the smaller entry of the coupler aligns with the stepper's rod. Fasten two 6 mm M4 set screws securely to hold the stepper's rod in place within the coupler. Tighten the set screws firmly but avoid over tightening to prevent damage to the components.

4. Take an M8 nut and insert it onto the threaded M8 rod. Screw the M8 nut onto the rod until it aligns with one end of the threaded rod. Carefully insert the threaded rod into the coupler, ensuring that the rod enters the hole in the coupler. As you insert the rod, the threaded hole in the coupler will be threaded by the rod, providing stability and alignment with the stepper's rod. Use the previously placed M8 nut as a guide to help thread the rod into the hole of the coupler properly. Adjust the nut as needed to ensure smooth threading. The threaded must be at least 15 mm inside the coupler. the Fasten two 6 mm M4 set screws securely to hold the threaded rod in place within the coupler. Tighten the set screws firmly but avoid over tightening to prevent damage to the components.

5. Move the M8 nut that is already attached to the threaded rod away from the coupler. Attach the slider to the T-slot profile, positioning it near the coupler. Slide the slider towards the coupler, aligning it with the M8 nut.

Use two 6mm M4 set screws to fasten the M8 nut to the slider. Insert the set screws through the designated holes in the slider, ensuring they engage with the M8 nut. Tighten the set screws securely to hold the M8 nut in place.

6. Place an M8 nut from the opposite end of the threaded rod and then place a washer (which can be 3D-printed) onto the rod. Attach the flanged ball bearing with an outer diameter of 18 mm on the flanged side (16 mm on the other side), ensuring that the flanged side faces towards the stepper motor. Add another washer and another M8 nut to the rod. Screw the last added nut to have approximately 7 mm of the threaded rod protruding from the end of the T-slot aluminum profile. Attach the end-holder to the end of the T-slot profile using 2x 8 mm M4 screws. Tighten them, but do not fully tighten yet. Adjust the position of the flanged ball bearing so that it makes contact with both the end-holder and the last added M8 nut. Tighten the flanged bearing by tightening the M8 nut that is closer to the stepper motor. Ensure that the flanged ball bearing is securely tightened between these two M8 nuts. Finally, tighten the M4 screws, ensuring that the end-holder is firmly attached and stable.

7. Insert the carbon fiber rod through the end-plate and into the slider. Use two 6mm M4 set screws to secure the carbon fiber rod in place. Tighten the set screws firmly to ensure a secure attachment of the carbon fiber rod.

8. Attach the cable of the MMC5983MA board to the appropriate connector. Slide the board inside the 3D-printed Hall sensor cap, ensuring it fits securely. Take a plastic screw M3 8mm and carefully fasten it to secure the board in place within the cap. Use another M3 8mm screw to attach the Hall sensor cap to the carbon fiber rod, ensuring it is securely tightened.

9. Take four 5 mm M3 screws and position one photosensor on each photosensor holder. Insert the M3 screws through the holes on the photosensors and tighten them. Once the photosensors are attached to their holders, proceed to install the holders on the T-slot profile. Take two 10 mm M4 screws and two M4 T-nuts. Position one photosensor holder on the desired location along the T-slot profile. Insert one M4 screw through a hole on the photosensor holder, aligning it with a T-slot opening. Insert an M4 T-nut into the T-slot opening, behind the photosensor holder. Repeat the above steps for the second photosensor holder. Once both photosensor holders are loosely attached, slide one of the holders closer to the coupler, but leave it loose for now. Slide the other photosensor holder to the desired position, adjusting the distance between the two holders to determine the range of the hall sensor. Once you have set the desired position and distance, tighten the screws on both photosensor holders securely.

2.2 Installation of electronic parts

We have designed a housing to ensure secure connections between electronic devices (see Fig 3). The existing pins of the L298 dual-H Bridge stepper driver and the ST1168 MOSFET-based power switch are directly connected to the Arduino without the need of cables. For the rest of the connections, we have designed an interface on the front panel of the housing that includes a total of 11 ports (see Fig. 2). These ports consist of 4 for MOSFET connections (input and output currents), 2 for photosensors, 1 for a hall sensor, 1 for the stepper motor, 2 for powering the stepper motor, and 1 for establishing a serial connection between the Arduino and a computer.

Figure 2: Ports of the assembled electronics module. A) PhotosensorA, B) PhotosensorB, C) Hall Sensor, D) stepper motor connections, E) 12V power supply (stepper), F) power supply (current for coils), G) coils H) computer.

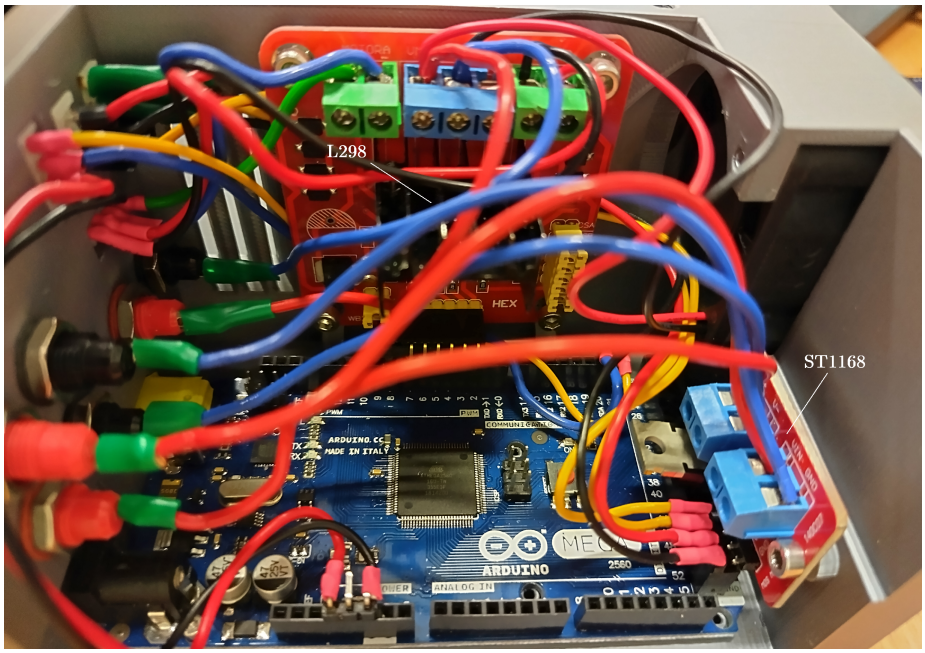


Figure 3: Layout of the electronic board placements inside the housing.

The power supply (stepper), power supply (current for coils), and coil connections are made via standard banana connectors. Photosensors, on the other hand, utilize 3-pin male JST XH2.54 ports, while the stepper motor and Hall sensor connections utilize 4-pin male JST XH2.54 ports. The JST connectors are glued into the openings of the housing.

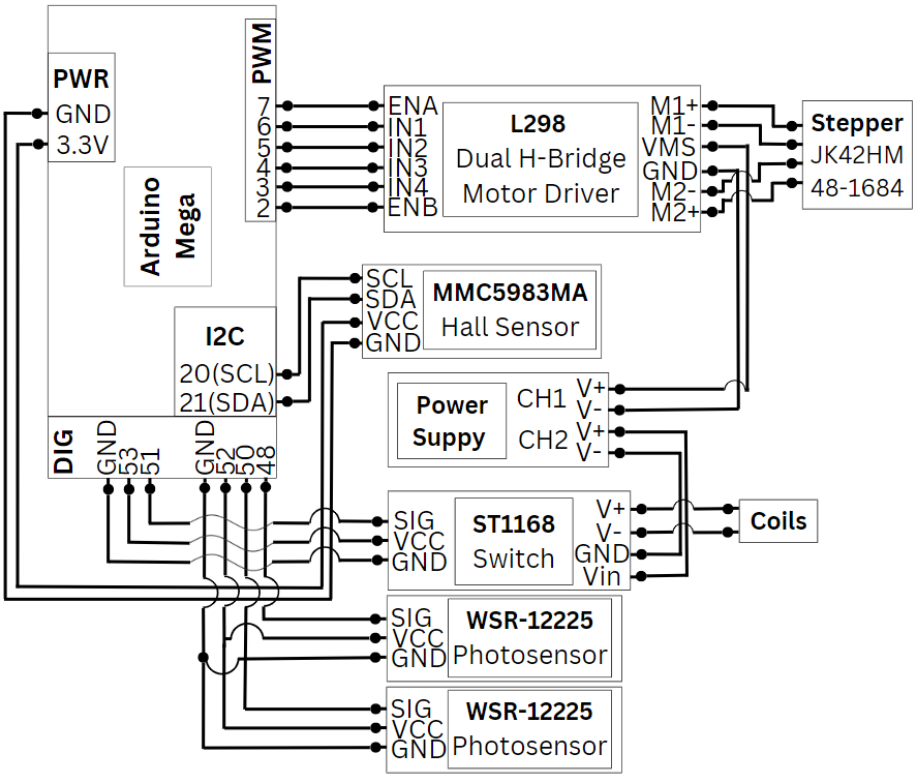


Figure 4: Schematic of the connections for the electronic components of the automated magnetometer. The Arduino Mega is controlled via a USB serial connection to a computer.

The electronic connections required for the project are depicted in Figure 4. If necessary, you can customize the default pin assignments of the microcontroller by modifying the source code of the project. The default parameters are loaded into the Python scripts from JSON files. To make adjustments to these files, you can use a simple text editor or run the 'config_creator.py' script, which generates the default configuration based on your modifications.

2.3 Micro-controller firmware

The electronic components of the automated magnetometer are managed by an Arduino Mega single-board micro-controller (see Fig. 4 for the schematics). Typically, micro-controllers are programmed with firmware designed to carry out specific operations, which can pose a challenge if modifications to the setup are desired, particularly in the case of open-source projects. To overcome this limitation, we have chosen to use a generic firmware based on the Firmata protocol, which facilitates communication between the software running on a computer

and the Arduino board. We utilized the Telemetrix4Arduino firmware. This firmware can be easily uploaded to the microcontroller using the Arduino IDE Library manager. The source code for the firmware can be accessed at the following GitHub repository: https://github.com/MrYsLab/Telemetrix4Arduino.

3 Operating the Automated Magnetometer

Operating the Automated Magnetometer is a straightforward process thanks to its user-friendly GUI. This chapter serves as a guide to help you navigate and utilize the GUI effectively.

3.1 Connecting to the microcontroller

Figure 5: The Automated Magnetometer GUI on initialization.

The connection to the microcontroller is established using the built-in functions of the Telemetrix-AIO library. When the initialization button (see Fig. 5 is pressed in the GUI, the software will attempt to establish a connection with the Arduino. If the connection is successful, a confirmation window will appear, indicating that the connection has been established. However, if the connection fails for any reason, an error will be raised, notifying the user of the issue. This ensures that the communication between the software and the Arduino is properly established before proceeding with any further operations. Once the connection is established, the Arduino will, by default, run a start-up sequence in which the selected pins in the configuration file will be configured to power up the devices.

3.2 Calibration

The calibration process is a critical step in ensuring consistent units for tracking the position of the hall sensor. By calibrating the system, you can determine the displacement of the sensor for a single step of the stepper motor. This value, for

user convenience, is presented as displacement in millimeters per revolution of the stepper in the Calibration frame of the GUI. This parameter is then used by the Python script to convert user inputs in millimeters into corresponding steps of the stepper motor. Consequently, during the measurement process, the script can accurately convert the position of the hall sensor back into millimeters. This calibration ensures precise and reliable measurements throughout the operation of the Automated Magnetometer.

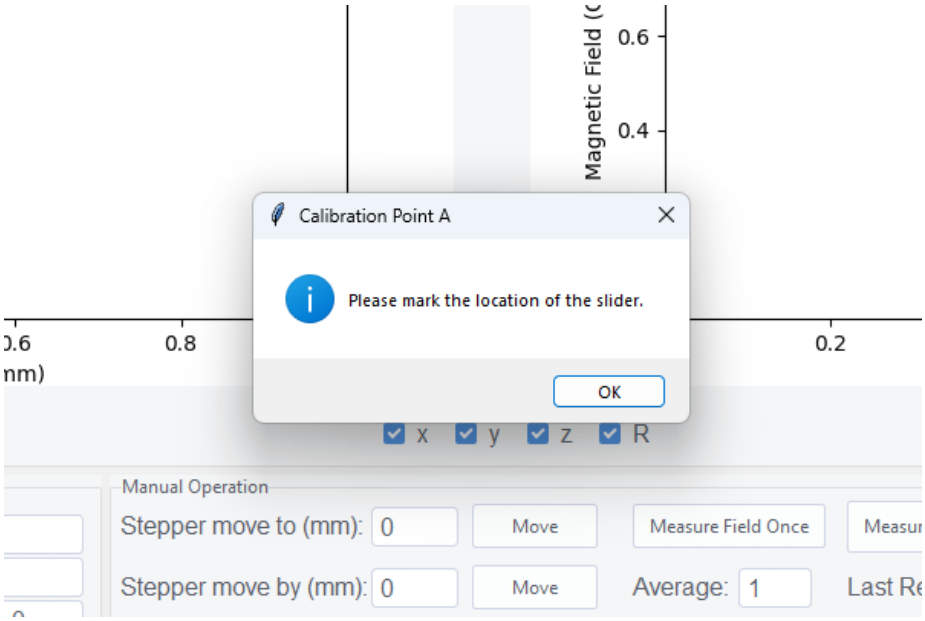


Figure 6: User is asked to mark the position of the slider.

The *Calibrate* button (see Fig. 6 in the GUI of the Automated Magnetometer initiates the calibration process. When the user clicks on the *Calibrate* button, the stepper motor will begin driving the slider away from itself continuously until it reaches the *photosensorB*.

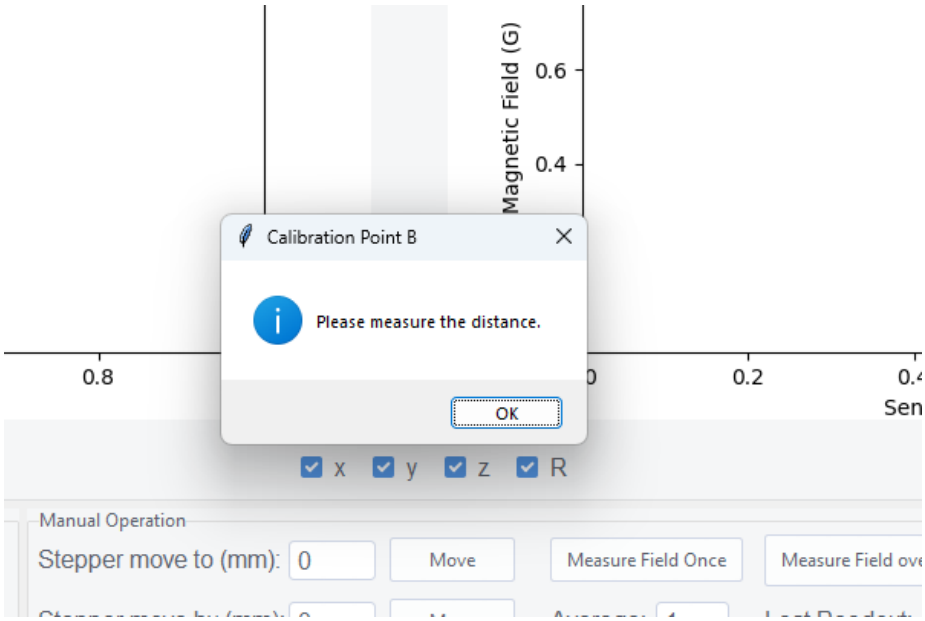


Figure 7: User is asked to measure the distance between the marked position and current position of the slider.

When the slider blocks the *photosensorB*, the user will be prompted to mark the location of the slider. Once the user confirms the marked position, the stepper motor will drive the slider to photosensorA. When the slider arrives at photosensorA, the user will be asked to measure the distance between the current position of the slider and the previously marked position (Fig. 7). Upon confirming the measured distance, the user will be prompted to submit the distance in millimeter (mm) units

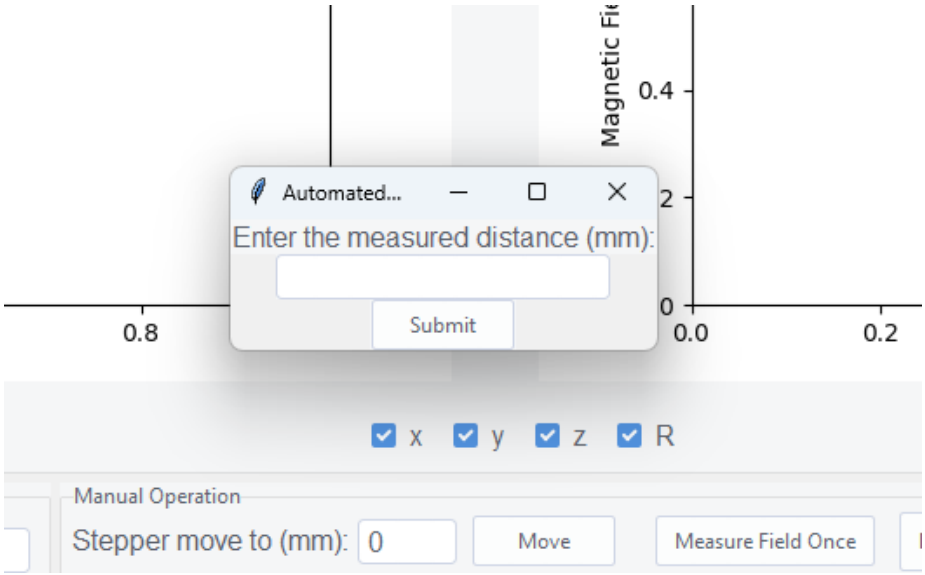


Figure 8: User is asked to submit the measured distance.

(Fig. 8). Once the distance is submitted, the GUI will calculate the distance that the slider will traverse in one revolution of the stepper motor. This calculated distance will be displayed in the Calibration frame (see Fig. 9.

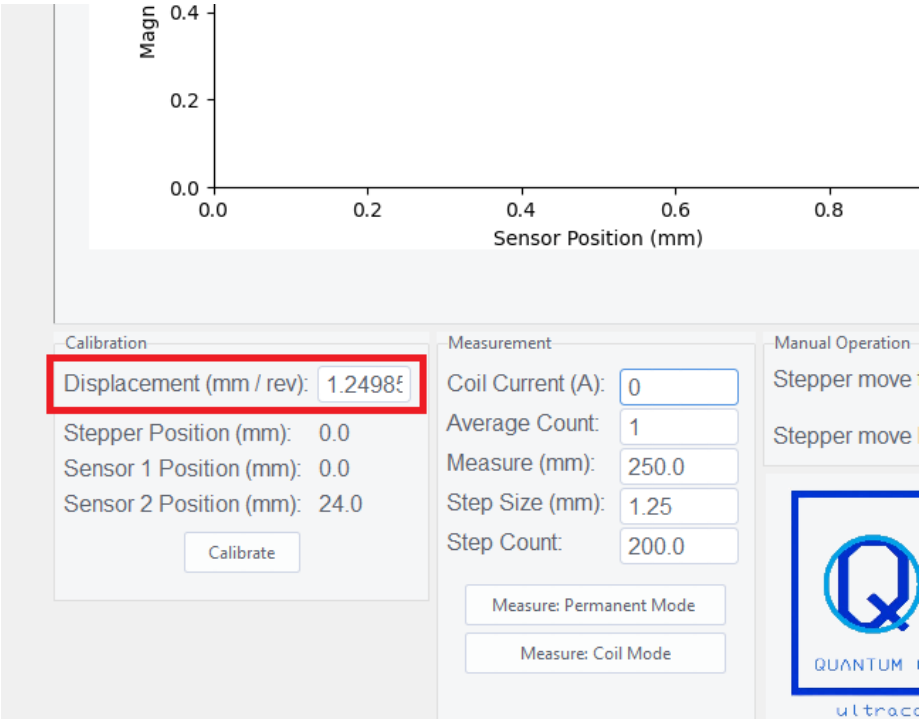


Figure 9: Calibrate frame of the GUI after calibration with the new determined displacement per revolution value.

The calculated displacement in mm per revolution is expected to be close to or equal to the pitch size of the threaded rod. This information is important for accurately converting user input in mm to the corresponding number of steps for the stepper motor, as well as converting the position of the hall sensor to mm during the measurement process.

3.3 Measurements

In the Measurement section of the GUI, the user is required to set several parameters. The first parameter is the 'measure' parameter, which represents the range of measurement from the current position of the stepper motor. To adjust the initial position of the stepper motor, the user can utilize the manual operation commands provided.

Once the measurement range is set, the user can proceed to adjust the step size or resolution. The step size determines the distance between the points where the magnetic field will be measured. After defining the desired parameters, the user can press the 'Enter' button. The GUI will then responsively validate and potentially modify the defined parameters to ensure they are within feasible limits. These parameters are interconnected and also subject to the

minimum displacement that can be achieved with the stepper motor.

By adjusting these parameters, the user can customize the measurement process according to their specific requirements, by namely adjusting the precision of the magnetic field measurements. Additionally, users are prompted to define the averaging count, which determines the number of data points that the hall sensor will collect during the measurement process. This averaging count is utilized to calculate both the stray magnetic field (without magnets) and the magnetic field in the presence of magnets. By specifying the averaging count, users can control the level of accuracy and stability in the measured magnetic field readings. A higher averaging count generally leads to smoother and more reliable results, while a lower count may provide faster measurements but with potentially increased noise or fluctuations. The GUI allows users to experiment and find the optimal averaging count based on their specific measurement requirements and the desired balance between measurement accuracy and speed.

With the "measure: Coil-Mode" and "measure: Permanent-Mode" buttons, users can initiate the measurement process. The collected data will be displayed on the data plots within the GUI interface. The GUI provides two separate plots to visualize the magnetic field measurements.

Figure 10: The view of the GUI while a measurement with the permanent-mode is on going. The plots are color-coded with x,y,z and R corresponding to Red, Green, Blue and Black data points.

On the right side of the GUI, the stray field plot represents the magnetic field readings without any magnets present and on the left side of the GUI, the field from the magnet is displayed (see Fig. 10).

3.4 Manual commands

Within the manual operation frame of the GUI, users have various options to control and interact with the stepper motor and hall sensor (see Fig. 11). These options include:

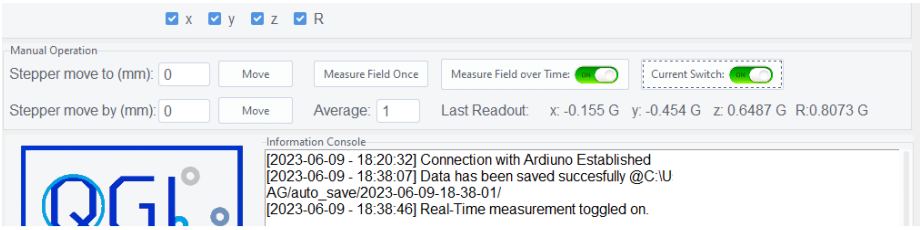


Figure 11: The view of the GUI's manual operation frame.

1. Move to Targeted Position: Users can specify a specific position to which they want to move the stepper motor. By entering the desired position, the GUI will command the stepper motor to move to that targeted position.

2. Move Relative to Current Position: Users can define a relative movement distance or step count to be performed by the stepper motor. This allows users to move the motor by a specific increment either forward or backward from its current position.

3. Single Measurement: Users can request a one-time measurement from the hall sensor. This will trigger the sensor to capture and display the magnetic field reading at the current position of the stepper motor.

4. Real-time Measurement: Users have the option to initiate a continuous measurement. The hall sensor will continuously collect data points and display them on the right plot of the GUI. This real-time measurement enables users to observe the changes in the magnetic field over time. This measurement will only show measurements taken in the last one minute.

5. Averaging: Users can specify the number of measurements to be averaged for both one-time and real-time measurements. By defining an averaging value greater than 1, the GUI will calculate the average magnetic field reading based on the specified number of measurements. This helps to obtain more stable and accurate results by reducing measurement variations if needed.

6. MOSFET Power Switch: Users can toggle the state of the MOSFET power switch. This feature allows users to turn the power supply to the connected devices (such as magnets or coils) on or off, providing control over the magnetic field generation or other related functionalities.

The manual operation tools provided in the GUI can be extremely useful for aligning the coils to be measured with the movement of the hall sensor. By tracking the data displayed on the GUI while using the manual commands to move the stepper motor, users can properly align the position the hall sensor to be crossing from the center of the coils.

3.5 I/O Handling

The application stores all the arduino pin configuration and user input variables in JSON files. Upon launching the application, these parameters are loaded from the JSON files. When the application is closed, all the parameters are saved back to the JSON files, ensuring that they are available for the next launch. The measurement results, along with the raw data points, are automatically saved to the folder specified by the user in the top-right section of the GUI. If the user doesn't specify a folder path, the application will save the data to a default folder located in the same directory as the Python script. Additionally, users have the option to manually initiate the saving process by pressing the "Save" button.

4 Comments on Modifications

Users can modify and/or improve this open-source Python code. The recommended approach for suggesting changes is through GitHub. The code consists of two main scripts: *AMGUI* and *AMDev*. *AMGUI* serves as the main script where GUI windows are created, and user commands are passed to the *AMDev* script. Both scripts are designed to work asynchronously.

It is important to note that the GUI in the current version is not handled in a separate thread or core. The variables are passed between the two scripts in a cleaner manner compared to other methods that require queue functions. The *AMDev* script handles device operations and runs Telemetrix-aio commands. New sensors, tasks, and operations can be easily added to the *AMDev* script.

On the *AMGUI* side, GUI elements are created, and the GUI updater function, known as *"mainHandler"*, is called iteratively. The *mainHandler* function is responsible for checking if specific events have occurred. When an Event is set, the *mainHandler* first clears the event and then executes the intended task associated with the occurrence of this event. Therefore, any new functionality added to the *AMDev* script can be incorporated by adding the corresponding function to the *mainHandler's* event handling script.