MacroQueue: Automating Measurements in High-Dimensional Parameter Spaces

Brad M. Goff [1] **and Jay A. Gupta** [1]¶

1 The Ohio State University, United States of America ¶ Corresponding author

Summary

Laboratory measurements often use several instruments to fully explore the relavent parameter space; such as, an external lock-in amplifer, an electromagnet, a RF generator, etc.. Ordinarily, these intruments have to be individually controlled and their parameters have to be manually recorded. MacroQueue enables effortless measurements throughout these high-dimensional parameter space systems. MacroQueue is a modular software designed for controlling and automating various laboratory equipment in sync without requiring significant coding ability. It can be extended to automate any system that can be controlled via Python.

Users can easily add python functions to control new and existing auxiliary equipment, in addition to new and existing systems, as long as they can be controlled via python. Although any arbitary python function can be added, it is preferable that functions are written with the functional programming paradigm in mind, so they are small and each perform a single task. This allows the functions to be reused for many types of measurements.
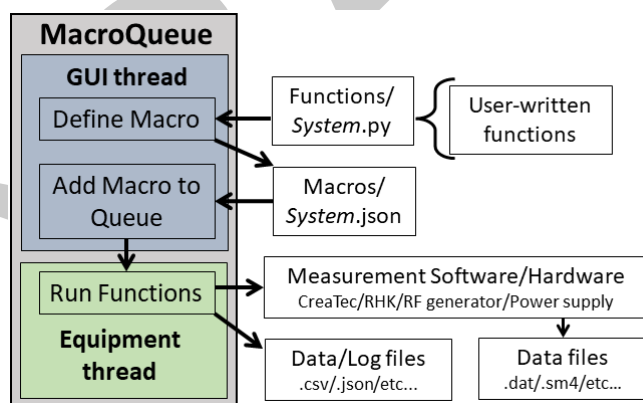
Figure 1: The general procedure for using MacroQueue.

The functions are grouped into a macro for each type of measurement. Macros are added to a queue with different values for each parameter, e.g. bias, magnetic field, etc., to perform measurements throughout the parameter space. Each measurement is performed consecutively on a seperate thread to allow measurements that are still in the queue to be modified. These features allow users to easily control several instruments in sync, add new instruments to a system, and perform long series of measurements with minimal input.

Statement of need

Numerous instruments have to be controlled in sync to access the full parameter space of a system. MacroQueue provides a simple GUI to allow users to perform measurements throughout the parameter space without coding. This software is currently in active use in several laboratories at The Ohio State University and the NSF NeXUS Facility. It assisted in the research of Goff et al. (2024) and Koll et al. (2024). Several similar packages and APIs already exist, such as Jermain et al. (2020), Rakitin et al. (2023), and Barnard (2023). A good overview of available Python packages can be found in Buchner (2022). The goal of MacroQueue is to provide a frontend GUI that allows users to perform measurements in high-dimensional parameter spaces without requiring the coding ability that is necessary to use the existing APIs while still providing advanced users the flexibility to write arbitrarily complex functions.

Overview

MacroQueue can be packaged into an executable, by either using the provided script or using PyInstaller directly, so that coding and using the command line is completely optional. To further allow the user experience to be as simple as possible, any arbitrary python function, no matter how basic, can be added to MacroQueue. Even as a executable, users can open the "source folder" via the File menu to access the python files that control various intstruments. Upon launching, MacroQueue searches this folder for new files. Every function, from each python file, is dynamically imported using the package importlib, part of python's standard library. For each parameter in a function, MacroQueue reads the default value to interpret the datatype (e.g. string, numerical, boolean, list, filepath) and the appriate control in the GUI (e.g. text box, numbers only text box, checkmark, dropdown menu, file browser respectively). Figure 2 shows example code and the various controls that are produced in the GUI.
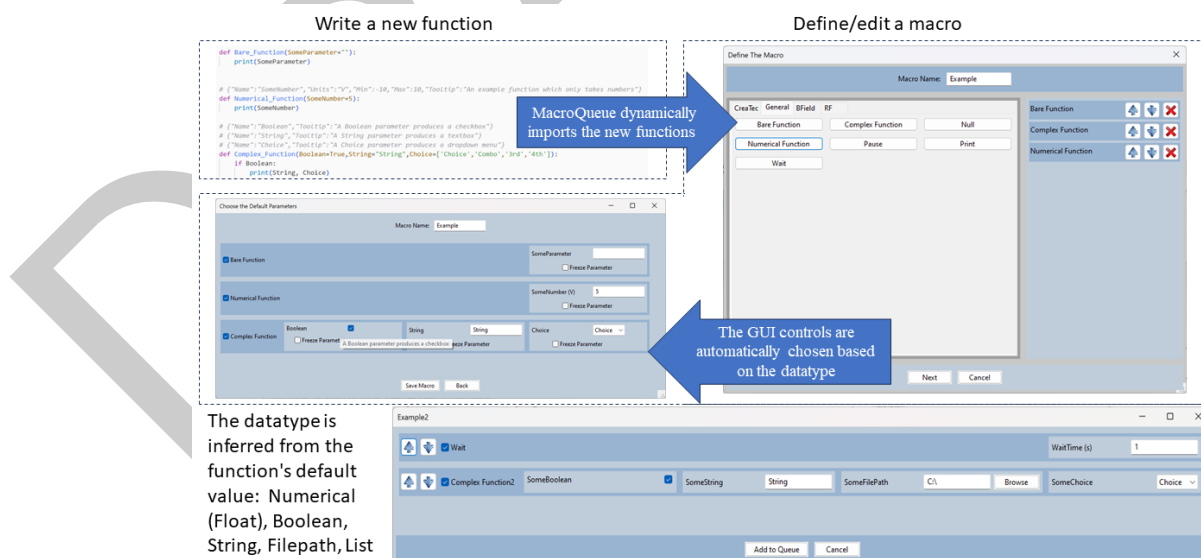

Figure 2: The workflow for adding a new function and defining a new macro.

For additional features, users can write metadata for each parameter in comments above each function. The metadata includes units and an explation of what the parameter does, which will be included in the parameters's hover tooltip. Numerical values can also have a soft minimum and/or maximum value in their metadata; when a users tries to input a value outside the range, there will be a pop-up warning to confirm that they want to proced. Hard limits can be

53 applied in individual function by raising an exception, such as a ValueError exception. If an
54 exception is thrown in any of the functions, intentional or otherwise, the queue will be paused,
55 the current macro will be canceled, and a pop-up will provide the user with the exception's
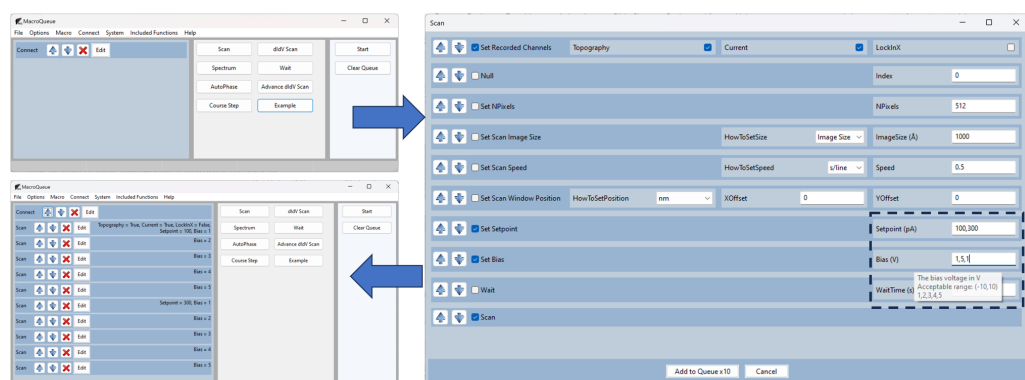56 details.

Figure 3: The workflow for adding macros to the queue.

57 MacroQueue makes a GUI, shown in Figure 3, using the WxPython toolkit Pasanen (2023).
58 The queue is on the left, showing the macros that will be run. The existing macros are on the
59 right. New macros can be created via the Macro menu or by modifying an existing Macro.
60 When you add a macro to the queue, there will be a menu where you can edit which functions
61 in the macro will be run and the values for each parameter that will be used. Users can input
62 multiple values for numerical parameters. MacroQueue will 'expand' the macro into multiple
63 macros, with each value, to the queue. The macros are created as if there is a for loop at
64 the expanding function and every function below the expanding function is inside the loop.
65 Figure 3 shows an example with 2 values for the setpoint and 5 values for the bias so 10 macros
66 are created. The setpoint function will run twice, and the bias function will run 10 times, 5
67 times per setpoint. This can be used to quickly add thousands of measurements throughout
68 the parameter space to the queue.

Acknowledgements

References

73 Barnard, E. (2023). *ScopeFoundry*. GitHub. https://www.scopefoundry.org/

74 Buchner, C. (2022). *Python lab automation landscape catalog* (Version v0.9) [Data set].
75 Zenodo. https://doi.org/10.5281/zenodo.6399528

76 Goff, B., Zhou, W., Bishop, A. J., Bailey-Crandell, R., Robinson, K., Kawakami, R. K., & Gupta,
77 J. A. (2024). Scanning tunneling microscopy study of epitaxial Fe3GeTe2 monolayers on
78 Bi2Te3. *2D Materials*, *11*(2), 025012. https://doi.org/10.1088/2053-1583/ad1c6d

79 Jermain, C., minhhai, jmittelstaedt, dennisfeng2, StePhanino, unclemoe, Rowlands, G., Girard,
80 H.-L., Schippers, C., Schneider, M., chweiser, Buchner, C., Spirito, D., Feinstein, B.,
81 nowacklab-user, Tiger, S., Vaillant, G. A., Boxtel, T. van, Chan, M.-A., ... chrische-
82 xx. (2020). *Ralph-group/pymeasure: PyMeasure 0.8* (Version v0.8). Zenodo. https:
83 //doi.org/10.5281/zenodo.3732545

84 Koll, W., Urdaniz, C., Noh, K., Bae, Y., Wolf, C., & Gupta, J. (2024). *Formation of oriented*
85 *bilayer motif – vanadyl phthalocyanine on ag(100)*. https://doi.org/10.48550/arXiv.2403.
86 19821

87 Pasanen, R. D. H. (2023). *WxPython* (Version v4.2.1). GitHub. https://docs.wxpython.org/

88 Rakitin, M., AbbyGi, Bode, R., Morris, T., jklynch, Romasky, B., Maldonado, J., & Moeller,
89 P. (2023). *NSLS-II/sirepo-bluesky: Sirepo-bluesky v0.7.2* (Version v0.7.2). Zenodo.
90 https://doi.org/10.5281/zenodo.8265981