

# 1 MacroQueue: Automating Measurements in 2 High-Dimensional Parameter Spaces

3 Brad M. Goff<sup>1</sup> and Jay A. Gupta<sup>1</sup>✉

4 <sup>1</sup> The Ohio State University, United States of America ✉ Corresponding author

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Open Journals](#) ↗

## Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright<sup>6</sup> and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))<sup>9</sup>

## 5 Summary

6 Laboratory measurements often use several instruments to fully explore the relevant parameter space; such as, an external lock-in amplifier, an electromagnet, a RF generator, etc.. Ordinarily, these instruments 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 coding ability. Currently, it includes functions to control the 3 major commercial scanning tunneling microscopes, CreaTec, RHK, and Scienta Omicron in combination with any other instruments that are a part of the systems. It can be extended to other control any instrument that can be controlled via Python.

15 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 arbitrary python function can be added, the base functions were created with the functional programming paradigm in mind, so the functions are small and each perform a single task. For example, the function *Set RF Frequency*, changes the frequency on the RF generator and records the new value. This allows the functions to be reused for many types of measurements.

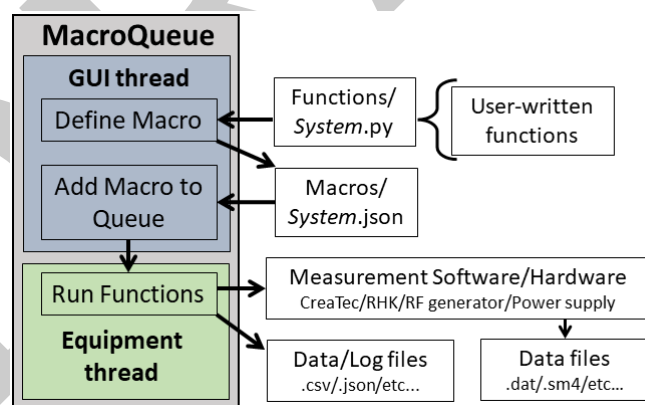


Figure 1: The general procedure for using MacroQueue.

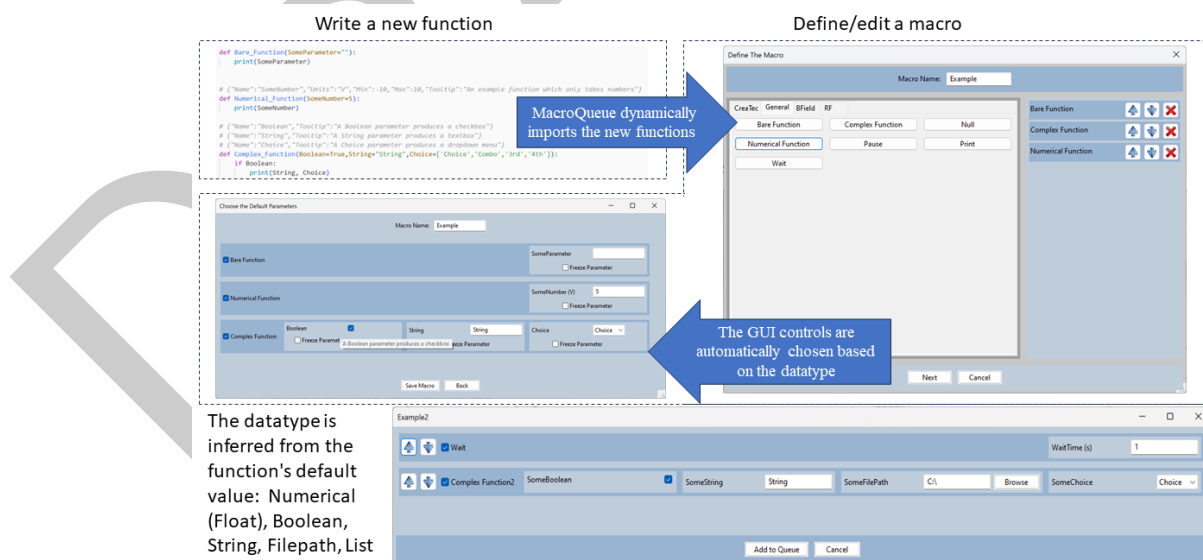
21 The functions are grouped into a macro for each type of measurement. Macros are added to  
22 a queue with different values for each parameter, e.g. bias, magnetic field, etc., to perform  
23 measurements throughout the parameter space. Each measurement is performed consecutively  
24 on a separate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

## 27 Statement of need

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

## 39 Overview

40 MacroQueue can be packaged into an executable, by either using the provided script or using  
41 PyInstaller directly, so that coding and using the command line is completely optional. To  
42 further allow the user experience to be as simple as possible, any arbitrary python function, no  
43 matter how basic, can be added to MacroQueue. Even as an executable, users can open the  
44 "source folder" via the File menu to access the python files that control various instruments.  
45 Upon launching, MacroQueue searches this folder for new files. Every function, from each  
46 python file, is dynamically imported using the package importlib, part of python's standard  
47 library. For each parameter in a function, MacroQueue reads the default value to interpret the  
48 datatype (e.g. string, numerical, boolean, list, filepath) and the appropriate control in the GUI  
49 (e.g. text box, numbers only text box, checkmark, dropdown menu, file browser respectively).  
50 **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.

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

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

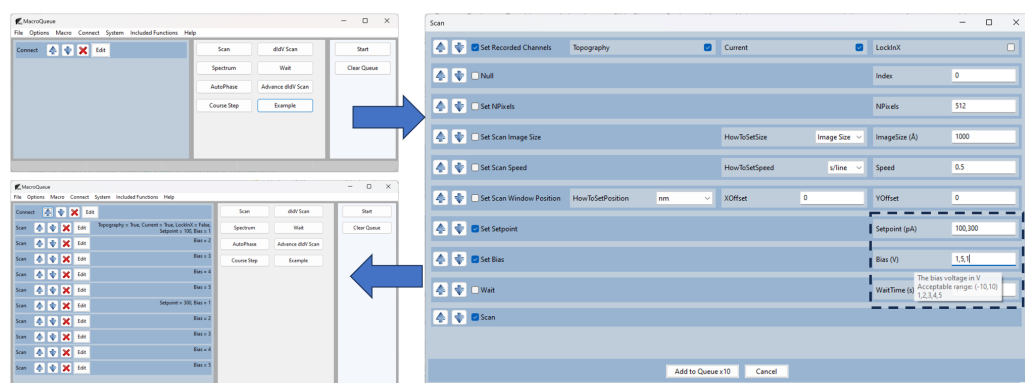


Figure 3: The workflow for adding macros to the queue.

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

## 72 Acknowledgements

73 This work was primarily supported by the Department of Energy (DOE) Basic Energy Sciences  
 74 under Grant No. DE-SC0016379.

## 75 References

- 76 Barnard, E. (2023). *ScopeFoundry*. GitHub. <https://www.scopefoundry.org/>
- 77 Buchner, C. (2022). *Python lab automation landscape catalog* (Version v0.9) [Data set].  
 78 Zenodo. <https://doi.org/10.5281/zenodo.6399528>
- 79 Goff, B., Zhou, W., Bishop, A. J., Bailey-Crandell, R., Robinson, K., Kawakami, R. K., & Gupta,  
 80 J. A. (2024). Scanning tunneling microscopy study of epitaxial Fe<sub>3</sub>GeTe<sub>2</sub> monolayers on  
 81 Bi<sub>2</sub>Te<sub>3</sub>. *2D Materials*, 11(2), 025012. <https://doi.org/10.1088/2053-1583/ad1c6d>
- 82 Jermain, C., minhhai, jmittelstaedt, dennisfeng2, StePhanino, unclmoe, Rowlands, G., Girard,  
 83 H.-L., Schippers, C., Schneider, M., chweiser, Buchner, C., Spirito, D., Feinstein, B.,  
 84 nowacklab-user, Tiger, S., Vaillant, G. A., Boxtel, T. van, Chan, M.-A., ... chrische-  
 85 xx. (2020). *Ralph-group/pymeasure: PyMeasure 0.8* (Version v0.8). Zenodo. <https://doi.org/10.5281/zenodo.3732545>

- 87 Koll, W., Urdaniz, C., Noh, K., Bae, Y., Wolf, C., & Gupta, J. (2024). *Formation of oriented*  
88 *bilayer motif – vanadyl phthalocyanine on ag(100)*. [https://doi.org/10.48550/arXiv.2403.](https://doi.org/10.48550/arXiv.2403.19821)  
89 [19821](https://doi.org/10.48550/arXiv.2403.19821)
- 90 Pasanen, R. D. H. (2023). *WxPython* (Version v4.2.1). GitHub. <https://docs.wxpython.org/>
- 91 Rakitin, M., AbbyGi, Bode, R., Morris, T., jklynch, Romasky, B., Maldonado, J., & Moeller,  
92 P. (2023). *NSLS-II/sirepo-bluesky: Sirepo-bluesky v0.7.2* (Version v0.7.2). Zenodo.  
93 <https://doi.org/10.5281/zenodo.8265981>

DRAFT