Hardware-Control: Instrument control and automation package

Grant Giesbrecht[1], **Ariel Amsellem**[1], **Timo Bauer**[1,2], **Brian Mak**[1], **Brian Wynne**[1], **Zhihao Qin**[1], and **Arun Persaud**[1]

1 Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA **2** Technische Universität Darmstadt, 64289 Darmstadt, Hesse, Germany

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Tim Tröndle

Reviewers:

- @aquilesc
- @untzag
- @garrettj403

Submitted: 10 September 2020
Published: unpublished

Summary

Conducting experimental research often relies on the control of laboratory instruments to, for example, control power supplies, move stages, and measure data. Tasks, such as data logging or parameter scans, often needs to be automated. Being able to easily create a user interface to the hardware and to be able to reuse code is highly desirable.

`Hardware-Control` is a Python package for instrument control and automation. It provides reusable user interfaces and instrument drivers to simplify writing control programs. `Hardware-Control` uses PyQt5, a GUI framework (Riverbank Computing Limited, 2020), to create fast and efficient user interfaces compatible with most major operating systems. `Hardware-Control` is also designed so that new drivers can be easily added for new hardware and used with existing user interfaces. The package also provides means for simplifying data collection with automatic data logging, plotting, and several export formats. `Hardware-Control` was designed with the use case of experiments in our laboratory in mind. Normally, this involves reading and setting voltages, controlling power supplies and oscilloscopes, updating values on external triggers, and automatically updating these values on a timer with time scales of about a second. The program is currently not well-fitted to do any real-time or near real-time communications with instruments or to handle data that is, for example, being streamed from a camera. The program interacts best with message-based devices or devices that already have pythons modules available to control them. The program's instrument drivers can send messages to sockets, usb ports, serial ports, etc. or call a python function to read or write a setting or call a command on the interface. Most of the drivers rely on the excellent pyvisa (PyVISA Authors, 2021) library, but drivers can also utilize pymodbus (Pymodbus Authors, 2021), or the built-in socket library.

For handling the data, we rely on numpy (Harris et al., 2020), pandas (The pandas development team, 2021), and many other common libraries that are available on pypi to do data analysis.

For communications within the library we rely on ZMQ (ZMQ Authors, 2021).

Statement of need

Commercial systems, such as LabVIEW, already exist, and they often do provide a wide range of instrument drivers (either directly or from the manufacturer of the devices). However, we have found that the resulting code is often hard to version control (LabVIEW files are binary, so code reviews and pull requests on services such as bitbucket and github are therefore difficult). Furthermore, although backend code can be shared between projects, complex user interfaces cannot easily be reused. The software package presented here tries to address these issues. Specifically, it makes reusing frontend code easier, integrates well with git, and provides an easy built-in scripting solution (via an optional python REPL that has full access to the GUI and all backends). The control through python during execution makes one-off complex parameter scans easy to implement. The software also makes it easy to develop and test the code without any hardware connected to the system by running a program in 'dummy mode' in which the hardware does not need to be present. When in dummy mode, the user can specify return values for certain opperations in order to test the program.

Similar packages already exist (see Scopefoundry.org (Barnard et al., 2020), pymeasure (Jermain & al., 2021), yaq (Yaq authors, 2021), etc.). A good overview of available python packages can be found at Ref (Buchner, 2021). However, for our experiments we had a specific usecase in mind, and we therefore decided to implement the provided solution. This software is currently actively used in our group at Lawrence Berkeley National Laboratory. Although our group is using `Hardware-Control` specifically in beam physics applications, we believe that the code can be useful for other experiments too.

In principle, we plan to continue to develop the code in the future by adding more instrument drivers and custom widgets since we believe that the current solution provides us several benefits compared to prior solutions and we have working setups at our test stands. Therefore, we are also open to community contributions. However, we are also open to merge our code into one of the pre-existing code bases and plan to explore these options going forward.

Acknowledgements

The information, data, or work presented herein was funded by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Contract No. DE-AC02-05CH11231.

References

Barnard, E. S., Buckley, A., Borys, N., Ogletree, F., Ursprung, B., Aiello, C., & Wu, H. (2020). *A python platform for controlling custom laboratory experiments and visualizing scientific data*. http://www.scopefoundry.org/.

Buchner, C. (2021). *Catalog the python lab automation landscape*. https://github.com/LabPy/labpy-discussion/issues/23.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Jermain, C., & al., G. R. et. (2021). *PyMeasure scientific package*. https://pymeasure.readthedocs.io/en/latest/index.html.

Pymodbus Authors. (2021). *Pymodbus documentation*. https://pymodbus.readthedocs.io/en/latest/index.html

PyVISA Authors. (2021). *PyVISA documentation*. https://pyvisa.readthedocs.io/en/stable/

Riverbank Computing Limited. (2020). *Python bindings for the qt cross platform application toolkit*. https://pypi.org/project/PyQt5/.

The pandas development team. (2021). *Pandas-dev/pandas: pandas* (latest) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.5574486

Yaq authors. (2021). *A modular and extensible instrument control framework*. https://yaq.fyi.

ZMQ Authors. (2021). *ZMQ documentation*. https://zeromq.org/