Reviewer Response Letter

Manuscript RSI22-AR-02455

The yaq Project: Standardized Software Enabling Flexible Instrumentation

The authors wish to thank both reviewers for their careful reviews! It's obvious from your questions that you have a strong background in this domain, and we feel that your comments have helped to improve our paper.

Reviewer 1

more traditional structure

The paper is compelling yet follows an untraditional organizational approach that generally makes it difficult to follow. There is discussion throughout about what is currently wrong with the field and too little focus on the yaq architecture and performance. The authors should focus on segregating the introduction and motivation away from methods and results. The methods and results should focus more on the advantages of yaq and the contributions to the field of yaq.

As of now, I am left with the conclusion of "yes, instrumentation control software is challenging and often done inefficiently." Instead, the authors should try to reorganize the paper so that my conclusion is "yaq presents an interesting method related to xyz and is capable of doing ABC". Before publication, the authors need to modify this paper into an article away from a commentary piece. More quantitative assessment of yaq would help; examples of yaq-enabled orchestration software would help even more.

Thank you to Reviewer 1 for this great point of critique. This has been a challenging paper for us to write, and the reviewer has eloquently summarized a tension we also feel. We originally tried to write a more technical paper, but found that when we shared it with our intended audience (instrument builders in our own department) they got lost in the details. Readers weren't able to grasp how the technical design of yaq could help them practically in their instrument building.

We decided to try writing a paper with the instrument builders in mind, and came up with the idea of structuring the paper around these relatable "challenges" that people in our department have faced. We have gotten good feedback about this structure from our intended audience. As we write, "This paper provides an overview of the concepts and motivations behind yaq. Refer to the yaq website for a formal specification of the yaq standard."

We agree with the reviewer that our paper needs improvement to serve the audience of informed instrument-software developers. It also needs to be clearer about comparisons between yaq and other software projects, as both reviewers point out. We would like to serve these readers while keeping the manuscript accessible to instrument builders who are not familiar with software architecture. To this end, we've added a lot to the SI, including quantitative assessment of yaq and examples of yaq-enabled orchestration. We have added references to this SI in the manuscript where appropriate. We have also expanded the comparisons between yaq and other software packages.

high-bandwidth

Significant challenges are often found with instruments that create high-bandwidth data streams that challenge the bandwidth of the communication method. Has yaq been demonstrated with cameras that generate video or has the rigol/mso1104 oscilloscope been configured to download screenshots?

yaq is limited by its network architecture. TCP is not particularly fast compared to direct hardware bus access. We have added a section in the SI that gives quantitative information about the scaling of yaq messages with size. Ultimately, we find that yaq is usable up to approximately 1 million camera pixels.

Some examples of potential future workarounds for high bandwidth applications are discussed in the SI. We have found that the advantages of our networked architecture are more important than high-bandwidth for our applications.

"Insurmountable" too extreme

We have rewritten this sentence, removing the word "insurmountable".

moving many motors

"A graduate student finds themself needing to master advanced concepts in concurrency in order to orchestrate many motors performantly."

The authors need to show how yaq fixes this. It seems that there would still be considerable complexity in driving multiple motors at once with yaq since the orchestration software needs to go back to check if the move was completed.

We intended this sentence to flow from the idea of "blocking interfaces". As the reviewer points out, orchestration software often needs to "go back and check" if motor motion is completed before moving on to the next step. Of course, this becomes more burdensome when the number of motors becomes large (10s of motors) and simultaneous motor motion is desired. In our experience, scientists typically try to solve this problem by adding multithreading to their

monolithic experimental software—and then they end up stuck with an unwieldy multithreaded mess which is challenging to debug.

The yaq architecture solves this by pushing all of the hardware interface complexity within the daemon. The yaq interface is performant enough that, in our experience, multithreading is not needed to “multiplex” addressing many motors simultaneously.

We have extensively rewritten the paragraph beginning “The third hardware interface barrier”. We have made this paragraph simpler and more direct in addressing what we think is important about the yaq interface with respect to addressing multiple pieces of hardware simultaneously.

The script examples and further discussion added to the SI also addresses this question.

define standards

“As we will show, usage of standards and the creation of tooling makes this architecture accessible to instrument builders outside of large facilities.”

Are these standards explained?

We’ve added a paragraph near the end of section 5 which highlights all of the features that make yaq more suitable for small research labs. These features were all explained in the manuscript, but we agree with the reviewer that the connection back to this point was unclear.

big data via avro

How is big data handled by Apache Avro? Images, videos, 10 MSPS ADC streams? Is there a way to save data locally?

Certainly “big data” is not the *primary* goal of yaq. That said, the choice of Avro in particular was made with array data in mind. Avro actually arises out of the Apache Hadoop community, so “big data” is very much in scope for Avro. We have added SI sections with some historical context around the choice of Avro and the path function yaq took to get there, as well as some quantitative analysis of how yaq performance scales with respect to large arrays (and options to improve performance).

Local storage is not something any current yaq daemon supports, though has been considered and probably will exist as cameras become more common in the ecosystem. We discuss this more in the SI.

extending traits

The authors should discuss how this trait concept is expanded to other instruments. Motors seem easy. How do I set the exposure time of a camera or AC couple channel 2 of an oscilloscope?

There are a few key ideas here.

The most foundational idea is that yaq supports arbitrary additional messages beyond those defined by the traits for a given protocol. For a camera you could simply define the message set_exposure_time for your specific protocol and that message would be available to anyone using your camera. Avro supports complex types and multi-argument messages. You could implement your AC coupling as set_coupling(dict(channel: coupling)) or set_coupling(enum channel, enum coupling). We do a pretty good job displaying complex types to the user graphically or through script clients.

As more and more hardware is added certain patterns might emerge. If there are several cameras with exposure time messages a new trait could be added “has-exposure-time”. This would provide a template so that all the protocols who wished to follow that standard message format could do so. A good example of this in practice is our existing has-turret trait which standardizes the approach for choosing gratings in monochromators that offer that feature. Traits are compositional so a given protocol can implement several traits simultaneously.

We’ve added a new paragraph to the manuscript to clarify how traits can be extended with unique messages.

synchronization via is_busy

“In order to know how long to wait, the “is-daemon“ trait provides a message called “is_busy”, which returns "true" while the long running action is not complete, and "false" once it is finished.”

It’s not clear if this is a solution or if it just kicks the can down the road.

We like the is_busy approach because it keeps scripts as simple as possible for scientists. This pattern of providing a busy getter does require polling, e.g. “while mymotor.is_busy(): time.sleep(0)”. Polling is less performant but easy to understand. Event structures and asynchronous patterns are a challenge for most of our users.

We’ve considered introducing a publish-subscribe pattern for alerting clients when the state of “busy” changes, but this has not become a priority. Polling is performant enough for all of the instruments we have built so far. We’re reluctant to add such features, the simplicity of yaq is one of its greatest strengths compared to other projects in this domain.

We've restructured the manuscript paragraph to mention less technical details and have added a section to the SI that goes into greater detail about timing and synchronization. These changes are also related to your comment about "moving many motors" above.

opinion piece

Seems very much like an opinion piece.

As we wrote in our response to the "more traditional structure" critique, we have attempted to highlight relatable challenges that people in our department have faced throughout the manuscript. We believe that this structure helps experimentalists and instrument builders connect the software design decisions to practical concerns they can understand. All of these examples are from our direct experiences.

orchestration example

(In context of figure 2) A more complete example would be very beneficial. Could you show reading, writing, etc. In other words, an orchestration example?

We have added several examples to the SI.

daemon discovery

Keeping track of these ports and IP addresses seems challenging. How is this done?

We have a lightweight tool, yaqd-control, that keeps a cache of daemons it has seen. Other tools in the ecosystem *e.g.* yaqc-qtpy can access that cache when looking for installed daemons. We've added a reference to this tool in the manuscript and a complete description in the SI.

Landis lightweight script example

Again much detail with limited yaq discussion. Could you provide an of a lightweight script example?

Is this mainly saying that creating a GUI early in the prototyping phase is a mistake? That conclusion is not seem to be yaq specific.

We have provided discussion and links to the Landis procedure scripts in the SI. We agree that creating a GUI early in the prototyping process is a mistake. This concept isn't yaq specific, but we've found that moving to yaq has helped researchers escape the early-GUI trap.

prior art around “separable and portable”

This paper has references to the literature in the introduction but then stops referencing other developments. There must be other works that create separate and modular interfaces to specific hardware. Could these be cited here? E.g. pyMeasure?

We have added the sentence “Separability, portability, and distributed development are advantages common to many open-source hardware interface projects.” We hope that the introduction makes it clear that yaq is just one open source project among many working in this domain.

tooling vs interface

I don't consider PyVISA and PySerial "tooling" libraries. Rather just interface libraries? If this is wrong the authors should define tooling.

We don’t have a specific definition in mind for “tooling”, we used the word loosely. At the reviewers suggestion we’ve removed that word.

“documentation is thankless” is opinion

An example of an unnecessary opinion sentence. I agree with this statement. Yet it is better omitted from the paper or saved for a discussion sentence that is clearly indicated as such.

We’ve rewritten this sentence and added a citation. We hope the reviewer will agree that the tone of the new sentence is better. The purpose of this sentence is simply to motivate the importance of our approach to documentation.

direct comparison to EPICS

What design choice makes yaq more feasible for small labs as compared to say EPICS?

We’ve added a paragraph near the end of section 5 which highlights all of the features that make yaq more suitable for small research labs.

Reviewer 2

1a - position in open source landscape

While clearly presenting the topic, the introduction lacks however a better description of how YAQ fits in the open-source software landscape and on what it really focuses. Indeed OpenSource instrumentation software can be put into two categories: hardware libraries,

instrumentation orchestration (or both). As is said later on in the manuscript, yaq focus mostly in hardware libraries via a standard interface. It would then be best to describe this focus in the introduction and also points to the reader how other cited software developments also fits into these categories. It would also be a good point to cite which are general and which focuses on particular domains. One could easily understand that a software focused only on hardware communication doesn't suffer by essence of thematic applicability.

We agree with the reviewer that orchestration and hardware interface support are two important aspects of instrumentation software. Our project focuses on hardware interfaces, there is no "built in" orchestration support. In our experience orchestration is always idiosyncratic, so the prudent approach is to support an ecosystem of different orchestration approaches rather than provide an "official" orchestration framework. This is the main takeaway of the "Experimental Flexibility" section of the manuscript, and we think the message is clear there.

As the reviewer points out, the introduction could be clearer about this point. To this end we have changed a sentence to highlight our focus on hardware support.

1b - unclear citations

Not all cited open-source software have proper citation (not even a website): ref 9, ref 10 and 15. While some can be found on the web I couldn't find anything on ref 10. There are also a lot of references that are PhDs thesis, are they findable on the net? Is there not a DOI going with them?

Here's links to the references you mentioned:

- Instrumental: A python-based library for controlling lab hardware https://doi.org/10.5281/zenodo.6591764
- TRSpectrometer documentation https://trspectrometer.readthedocs.io/en/latest/
- pyLabLib https://doi.org/10.5281/zenodo.7324876
- R. J. Carlson Ph.D. Thesis https://www.proquest.com/docview/303711007
- K. A. Meyer Ph.D. Thesis https://www.proquest.com/docview/305111493
- S. Kain Ph.D. Thesis https://www.proquest.com/docview/1985671488
- B. J. Thompson Ph.D. Thesis https://doi.org/10.5281/zenodo.7627321
- K. F. Sunden Ph.D. Thesis https://doi.org/10.5281/zenodo.7627331

Thanks for pointing this out, we agree that the references need work! We have tried and failed to get these links to appear using LaTeX and the revtex class (we did get the TRSpectrometer link to appear in the revision). Some of the citations are clickable hyperlinks if your PDF reader supports that. If this manuscript is accepted we will certainly work with the copy editor to make sure everything is as clear as possible!

We've gone through all of the references again and added all the information we can find.

2 - question about blocking

In page 2 related to figure 1. Daemons are services running over the network and various clients can connect to it. Nothing is said if various clients requests different actions on one client at the same time? For instance, if an orchestration software runs an acquisition but then some other client requests also data or a motor displacement. That would completely break the acquisition? Is there some kind of priority or mechanism to prevent this? This is also related to the paragraph on page 3: "Additionally, multiple clients can communicate with the same daemon simultaneously..."

Thank you for this insightful question. yaq is preemptive. If a motor receives a `set_position` command while it is actively moving, it will immediately begin moving to the new position. It shouldn't be possible to "confuse" the daemon by sending commands from multiple clients at the same time.

It's absolutely true that multiple clients simultaneously commanding a daemon might lead to unexpected behavior. At this time we have no plans to build access control into the yaq protocol. The additional complexity of access control would outweigh the benefit for the relatively small instruments that we are building. It is simple to avoid unexpected behavior when only one or two scientists are working in-person with a single instrument. We have prioritized simplicity where possible to keep deployment easy. In cases where access control is critical we feel that specialty network tools and/or client-side controls are probably sufficient *e.g.* CAgw at BESSY. We have added language to the manuscript explaining this.

3 - part II additional sentence

In page 3 the three last paragraph of part II, highlight the use of yaq with respect to the three barriers discussed earlier. I found this part starts abruptly. A sentence of introducing saying
that the authors are going now to discuss how yaq behaves with respect with these barriers will make the reading much easier.

We have added an introductory sentence, thank you for the suggestion.

4 - figure 2

On page 3 paragraph starting with : "At the bottom, we show several...". The authors are referring to the bottom of figure 2. I don't see what features they are talking about. Later in the same paragraph they quote again Figure 2, while they are already commenting about it in this paragraph. Also, I don't understand how we are supposed to see the parametrization of

the bluesky procedure and representation of the data?!

The reviewer is absolutely correct. We simplified Figure 2 but forgot to update this paragraph. We've rewritten this paragraph to describe the simplified figure. The original version of Figure 2 contained some simplified screenshots and graphics of yaq integrating with Bluesky. Since those don't appear in the manuscript anymore we've added expanded explanations and screenshots to the SI.

5 - layer vs interface vs software

On page 3 referring to translation layer. It is written "Similar translation layers [...] orchestration layers such as PyMoDAQ, Instrumental [...]". I believe the authors means orchestration software (not layers). But they should also pay attention that Instrumental is not an orchestration software but is actually very similar to YAQ as it focus on hardware interfaces. As for TRSpectrometer citation, I couldn't find anything about it.

We agree with the reviewer, Instrumental was a bad choice here. PyMeasure is a much better example of a package that contains orchestration procedures, so we have cited that instead. We've also removed the second instance of "layers", replacing that word with software.

Here is a link to the TRSpectrometer documentation
https://trspectrometer.readthedocs.io/en/latest/

6 - daemon discovery

I couldn't find if there are some mechanisms to "discover" which dameons are currently working. This would be important to use them by an orchestration software that would check
which are installed.

Please see our response to Reviewer 1 on this topic.

7 - additional traits

I could only find a minimal set of traits in the documentation, how a particularly complex hardware interface (such as the one for a CCD camera) can be used? They are modes of acquisition (image, vertical binning...), ROI, binning, triggering... Would the use of these require the creation of new traits? Something on this should be explained

Please see our response to Reviewer 1 on this topic.