yaq

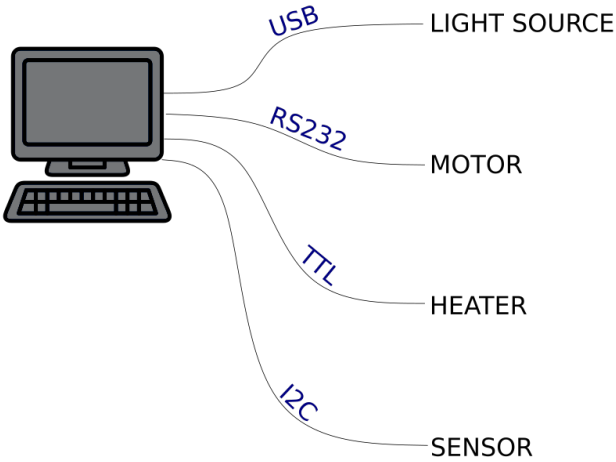
Instrument Shop Training

Blaise Thompson

University of Wisconsin–Madison

June 17, 2022

A Typical Instrument



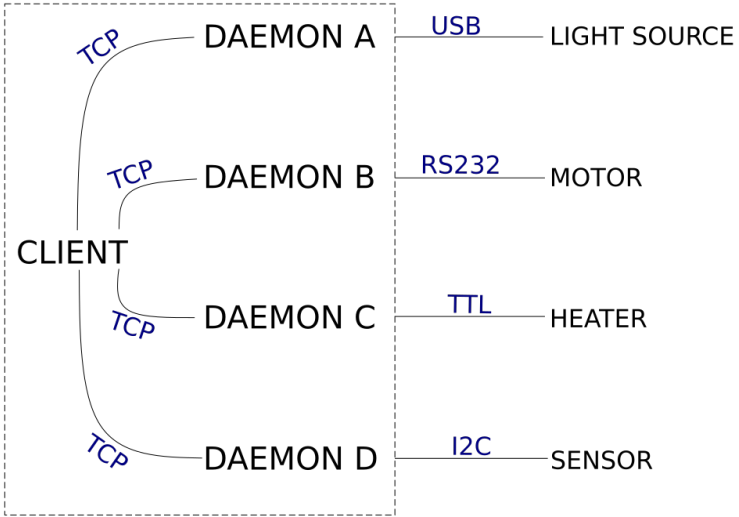
A Typical Instrument

typical instrumental software...

- monolithic
  - hard to develop—need everything to work for anything to work
  - any one piece of the software can crash everything
  - hard to reuse pieces in other instruments
- inflexible
  - lots of “baked in” assumptions about the particular hardware attached
  - takes a long time to change experimental approach

... limits experimentalist freedom!

The yaq Approach



The yaq Approach

distributed

- each daemon can be developed separately from everything else
- no piece of software can crash everything
- possibility of multiclient or networked access

portable

- reusable daemons
- benefit from existing ecosystem rather than “reinventing” hardware support
- use just what you need
- multilingual (in theory)

The yaq Approach

Guiding principle:

simplify client development as much as possible

YOU create clients

The yaq Approach

self describing

- daemon tells the client about itself in a very structured way
- good documentation: `https://yaq.fyi`

traits

- enforce consistency between similar daemons where possible
- optional, extensible

The yaq Approach

simplify timing control

- any daemon can be asked for “busy” state
- architecture removes annoyance of blocking hardware interfaces

yaq clients

using your package manager, install `yaqc`

- `client = yaqc.Client(host="192.168.1.1", port=38002)`
- `client.get_position`
- `client.set_position`
- `client.busy`

Try comparing across all machines!

yaq clients

using your package manager, install `yaqd-control`

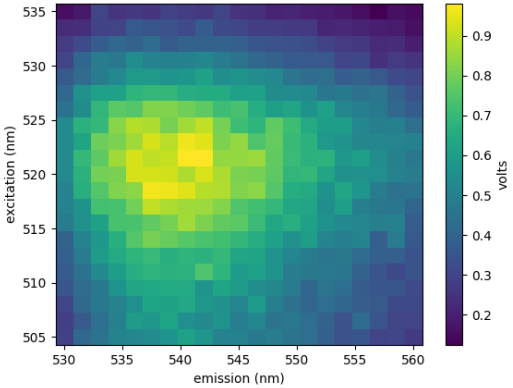
- `yaqd scan --host 192.168.1.1 --start 38000 --stop 38005`
- `yaqd status`

yaq clients

using your package manager, install `yaqc-qtpy`

graphical client based on traits

Test Hardware



fluorescein in ethanol fluorescence
using Python, take an emission slice for a given excitation wavelength

- numpy linspace
- for loop
- matplotlib.pyplot

Using yaq

yaq doesn't have "built in" orchestration software

- write scripts
- create experiment-specific GUIs
- utilize existing projects focused on orchestration

the best choice depends on your instrument

- scripts are flexible and lightweight—rapid development, few users
- experiment-specific GUIs are great for mature experiments with many users
- can introduce as much or as little sophistication as needed

importantly you can choose a combination of many options!

Bluesky

https://blueskyproject.io/