Remote control

The aim of this project was to create a program, which presents a user interface, in which a user can control or monitor ‘instruments’ running on a Raspberry Pi, or similar device.

By instrument, I mean a Python program you have written, or may be thinking of writing that controls GPIO pins or attached hardware.

After a bit of research I found a specification exists ‘INDI’ – which defines a simple protocol for transmitting instrument data, such a switch controls, or measured numeric values to an attached client.

https://en.wikipedia.org/wiki/Instrument_Neutral_Distributed_Interface

This specification and protocol is widely used in the amateur astronomical world for controlling telescopes, however it is a general purpose protocol and can be used for any form of instrumentation. It defines the data to be divided into ‘switches’, ‘lights’, ‘numbers’, ‘text’ and ‘BLOBs’ (binary large objects), which covers a wide swathe of typical uses.

The basic idea is that a driver has to be written, interfacing to your own instrument code and organising any data you present into a standard format which includes things such as labels and numeric format strings. This is then served on a port to which a client can connect. The client could be remote, or could run locally.

The client uses the labels and names to present the data, and to send controlling information, such as switch On or Off. The client can be anything from an automating script, terminal client, distributed displays or anything which can be written which understands the protocol. In my case I was aiming at a terminal and a web display.

As the client learns everything from the presented protocol, it can be general purpose, the same client being able to connect to any INDI service. A more targetted client could also be written to control a specific instrument if required.

The project turned into the following Python programs:

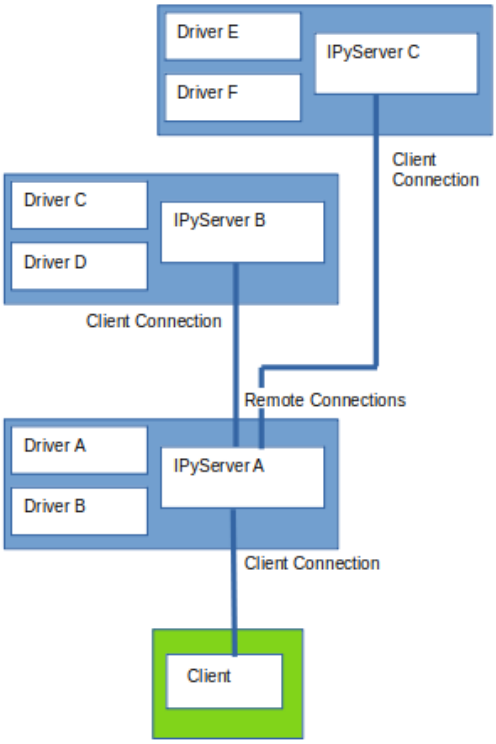
indipydriver

This provides classes you use to create a driver for your instrumentation which sets, updates and reads data.

indipyserver

This provides a class to serve your driver (or multiple drivers) on a port. It also has capability to make calls to other INDI servers, and broadcasts the INDI protocol between them. This enables a tree network of servers and drivers to be connected together, and a client connected to any can control any instrument on the network.

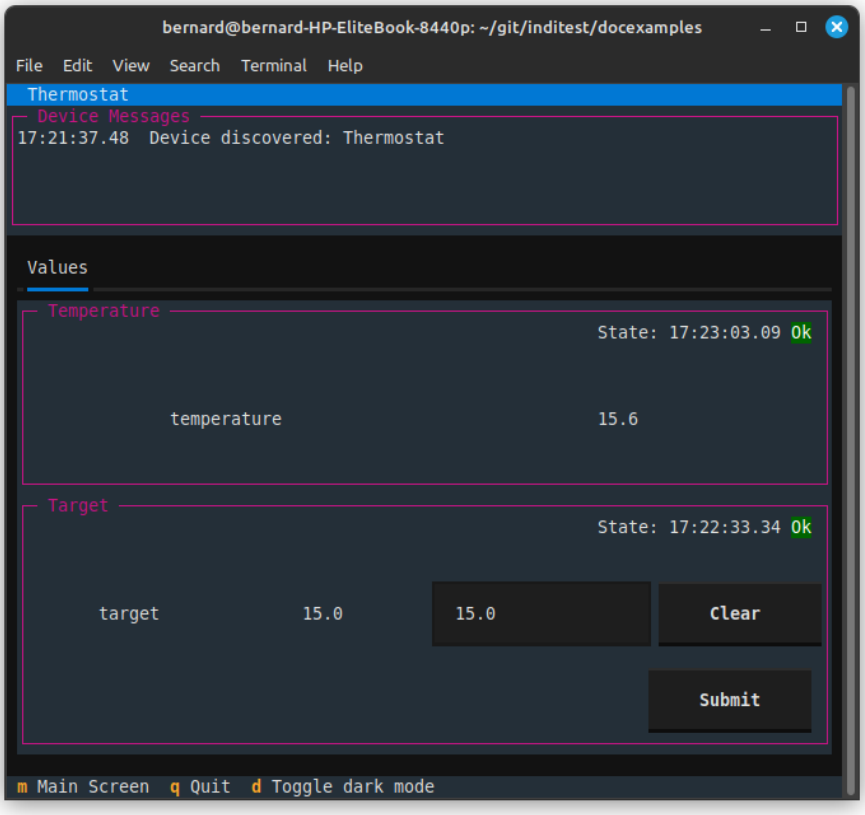
A network of servers and drivers could look like:



indipyterm

This is a general purpose terminal client, it is simply run, connects to an indiserver port, and displays and controls the instrument parameters it learns. There is no user programming involved.

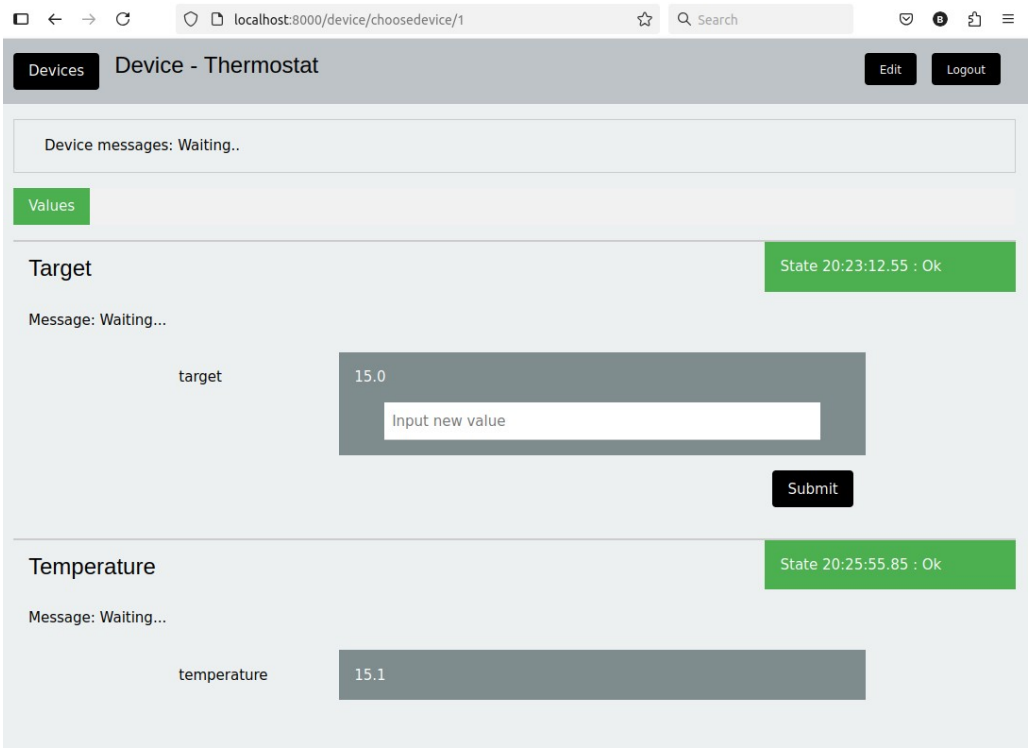
A terminal interface is simple for a headless device, to which you can connect by SSH.



indipyweb

This is another client, when run it connects to an indipyserver port, and then acts as a web server, and users can connect with their browser which displays and controls the instrument parameters.

Indipyweb also has the ability to add ‘admin’ and ‘user’ logins. Without login, a user has read-only access, with login a user has read-write access. The web server also provides a read-only JSON output showing instrument status.



Indipyclient

This is a Python package that most users will not run directly, it is used by both indipyterm and indipyweb to decode the INDI protocol and present the received data as Python classes. This package could be useful if a user wants to create his own clients or scripts for automated remote control.

Installing

All these packages are available on Pypi. Typically you would use pip to install these into a virtual environment.

If you use the uv tool, indipyterm can be very simply loaded from Pypi and run with the single command:

uvx indipyterm

Similarly indipyweb can also be run using:

uvx indipyweb

Rather than go through all the Python classes and their use in this article, if you are interested please see the following sites where there is a great deal of detail together with example code.

indipydriver sites:

https://github.com/bernie-skipole/indipydriver

https://indipydriver.readthedocs.io

https://pypi.org/project/indipydriver

indipyserver sites:

https://github.com/bernie-skipole/indipyserver

https://indipyserver.readthedocs.io

https://pypi.org/project/indipyserver

indipyclient sites:

https://github.com/bernie-skipole/indipyclient

https://indipyclient.readthedocs.io

https://pypi.org/project/indipyclient

indipyterm sites:

https://github.com/bernie-skipole/indipyterm

https://pypi.org/project/indipyterm

indipyweb sites:

https://github.com/bernie-skipole/indipyweb

https://pypi.org/project/indipyweb

Examples of clients and drivers are collected at:

https://github.com/bernie-skipole/inditest