Project description

Emu-Api

This library allows communication with a Rainforest Automation EMU-2 device. It is a rewrite of Rainforest's Emu-Serial-API in Python 3, and follows a similar architectural pattern where possible.

Installation

This library is distributed via pip and may be installed using `pip install emu-power`.

Usage

This API can be used in asynchronous mode or synchronous mode. Technically, communication always happens asynchronously with the Emu, but this library allows for a synchronous API to emulated.

As in Rainforest's library, we do not automatically start communication once the main object is instantiated. This is done because the EMU-2 constantly pushes data to us over the serial connection, so we want to give the consumer control over when this channel is opened. Before any commands are issued, the `start_serial(<serial port>)` method must be called to open the serial port and begin receiving data.

The serial port name is the platform-specific device that you wish to use. Note that unlike Rainforest's library, we do not attempt to detect the host platform or set any port prefixes - this means that the full name of the device (prefixed with `COM` for Windows or `/dev` for OSX and Linux, usually) must be used.

Example Usage

Synchronous

```
from emu_power import Emu

api = Emu(synchronous=True)
api.start_serial("/dev/tty.usbmodem146101")

# This will return an instance of InstantaneousUsage, or None on timeout.
```

```
response = api.get_instantaneous_usage()
```

Asynchronous

```
from emu_power import Emu
from emu_power.response_entities import InstantaneousUsage
import time

api = Emu()
api.start_serial("/dev/tty.usbmodem146101")

# This will return immediately. The response data will become available
# when the device responds.
api.get_instantaneous_usage()
time.sleep(5)
response = api.get_data(InstantaneousUsage)
```

Note: In real programs using asynchronous mode, it would probably make sense to make use of the schedule function of the EMU-2. This sets the frequency that certain events are sent, and allows for data to be received without constant polling. The schedule may be set using the set_schedule(...) method of the API. After the schedule is set up, the consuming program may periodically call the get_data method to access the most recent received data.

Contributing

Contributions are welcome! Not all commands have been thoroughly tested yet, since I haven't have a reason to use some of them. This library was written both to provide a more convenient interface with the EMU-2, as well as to get some experience writing Python modules. Functionality and style improvements suggestions are welcome.

This library is based off of the XML spec for the Rainforest RAVEN, which may be found on Rainforest Automation's website. This spec is similar but not identical to the API that the EMU-2 uses. Some commands not listed in this document were implemented based off of the official Python 2 library, which may be found on Github.