8minute SOLAR ENERGY

Modbusy:

Modbus for mortals

August 30, 2019

Who am I?

Brandon Carpenter
Principal Controls
Software Engineer
brandon@8minute.com

8minute Solar Energy
4370 Town Center Blvd, Ste 110
El Dorado Hills, CA 95762
www.8minute.com

What is Modbus?

- Low-level electronics control protocol
  - Originally developed to control PLCs over serial connections
- Updated to work over TCP/IP
- Typically involves many slaves controlling real hardware
  - One master controls one or more slaves
  - Each slave controlled by only one master

Using real Modbus devices for development is problematic

- Environment is usually isolated and difficult to access
- Controlling real hardware has real-world consequences
- Not designed for simultaneous access
- Difficult to reproduce results

Why use an emulated Modbus device?

- Each developer can have their own instance
- Can run on the developer's system
- No real-world consequences
- Easy to reproduce results
- Simple to reset to a known state
- Apparent behavior of hardware can be modified

What about other Python Modbus libraries?

- Few Python Modbus libraries exist
  - pymodbus and modbus_tk are most notable
- All require more than a familiarity with the Modbus protocol
- Require more code for even simple slaves

What is modbusy?

- Pronounced “mod-bussy” or “mod-busy”
- Python library based on uModbus (µModbus)
- Uses decorators and high-level helpers
- Requires little knowledge of Modbus
- Includes a helper to create slaves from a CSV file and a YAML configuration
- "So easy a caveman could do it"

uModbus example

```python
# umodbus_example.py
from socketserver import TCPServer
from umodbus import conf
from umodbus.server.tcp import RequestHandler, get_server

conf.SIGNED_VALUES = True  # Enable values to be signed (default is False)

def main():
    TCPServer.allow_reuse_address = True
    app = get_server(TCPServer, ('localhost', 502), RequestHandler)
    values = [0, 0]

    @app.route(slave_ids=[1], function_codes=[3, 4], addresses=[0, 1])
    def read_data_store(slave_id, function_code, address):
        return values[address]

    @app.route(slave_ids=[1], function_codes=[6, 16], addresses=[0, 1])
    def write_data_store(slave_id, function_code, address, value):
        values[address] = value

    try:
        app.serve_forever()
    finally:
        app.shutdown()
        app.server_close()

if __name__ == '__main__':
    main()
```

Modbusy example

```python
# modbusy_example.py
# Execution: python modbusy_example.py --address 127.0.0.1 --port 5020 12345

import contextlib
import click
import modbusy

# The order of decorators is important
@modbusy.tcp_app()
@click.argument('value', type=int)
@contextlib.contextmanager
def main(app, value) -> None:
    '''Modbus emulator to serve a single signed 32-bit value'''

    @app.register(0, modbusy.INT32)
    def read_value(for_write):
        return value

    @read_value.setter
    def write_value(new_value):
        nonlocal value
        value = new_value

    yield

if __name__ == '__main__':
    main()
```

Create a Modbus slave from time series data

sample_pv_1min.csv

```
utc_timestamp,active_power_total,dc_voltage,dc_unclipped_power
2018-03-01 00:00:00,71431,686.505555555555,72056
2018-03-01 00:01:00,70676,687.975925925926,71287
2018-03-01 00:02:00,69835,687.888888888889,70436
2018-03-01 00:05:00,67641,687.964814814815,68207
2018-03-01 00:06:00,67039,688.537037037037,67603
2018-03-01 00:07:00,66345,688.925925925926,66898
2018-03-01 00:08:00,65831,689.492592592593,66375
2018-03-01 00:10:00,64436,690.283333333333,64957
...
```

pv.yaml

```
registers:
  - address: 0
    column: active_power_total
    type: float
    trigger: yes
  - address: 2
    column: dc_voltage
    type: float
  - address: 4
    column: dc_unclipped_power
    type: float
  - address: 100
    type: string[19]
    column: utc_timestamp
```

```
python -m modbusy.csvslave --address 127.0.0.1 --port 5020 pv.yaml sample_pv_1min.csv
```

Conclusion

- Modbusy makes writing Modbus slaves easy
- Released under open-source license
  - BSD 3-clause license
- Source code available on Bitbucket: https://bitbucket.com/8minutenergy/modbusy

Brandon Carpenter
Principal Controls
Software Engineer
brandon@8minute.com

8minute Solar Energy
4370 Town Center Blvd, Ste 110
El Dorado Hills, CA 95762
www.8minute.com

Contact

Brandon Carpenter
Principal Controls
Software Engineer
brandon@8minute.com

8minute Solar Energy
4370 Town Center Blvd, Ste 110
El Dorado Hills, CA 95762
www.8minute.com