KPM Power

Anzen Modbus

Ashar Latif
ashar@kpmpower.com
August 6, 2021

1 Installation

For the rest of these instructions, it is assumed that `python3` and `pip` are installed. You can ensure this by typing the following (hitting `Enter` between lines and entering your password when requested):

```
sudo apt update
sudo apt install -y python3 python3-pip
```

If you are unsure if they are installed, run the commands anyway (it is idempotent if the dependencies already exist).

Once you are sure that `python3` and `pip` are installed, you can do install the virtual environment by invoking either of the following:

```
python3 -m pip install virtualenv
```

OR:

```
pip3 install virtualenv
```

If you are running this in WSL, close your terminal window and open a new one. If you are using native Linux, source your shell config file (`.bashrc` if using `bash`, `.zshrc` if using `zsh`, or `config.fish` if using `fish`; for any other shell please check the user manual).

If you are unsure if they are installed, run the command anyway (it is idempotent if the dependencies already exist).

The rest of this documentation assumes that you are working in a virtual environment, so ensure that this is done before proceeding.

1.1 Clone the Github Repository

If the repository is not present on the device, clone it by invoking:

```
git clone https://github.com/kpmpower/pymodbustcp-server
```

Enter your Github user name and password and the repository will be cloned. Navigate to the directory by typing:

```
cd pymodbustcp-server
```

If there has never been a Modbus server on the device, we will need to enable the services before we can proceed. You can do this by invoking the following (pressing `Enter` after each line):

```
sudo cp services/modbus_server.service /etc/systemd/system
sudo cp services/fbs_modbus.service /etc/systemd/system
sudo systemctl enable --now modbus_server.service
sudo systemctl enable --now fbs_modbus.service
```

1.2 Starting a Virtual Environment

If you just cloned the repository you now need to create a virtual environment, which you can do by invoking the following (fill in `{ENVIRONMENT_NAME}` with whatever you want to name you virtual environment. I like `venv`):

```
virtualenv {ENVIRONMENT_NAME}
```

If you named your virtual environment `venv`, you would invoke:

```
virtualenv venv
```

This will create a new environment. You can then activate it by invoking:

```
source venv/bin/activate
```

You will know if the environment is activated because the name of the environment will now precede your user info on the command line:

Before activation

```
anzen@zincfive-a023:~/pymodbustcp-server $
```

After activation

```
(venv) anzen@zincfive-a023:~/pymodbustcp-server $
```

1.3 Install Dependencies

The modbus server has the following dependencies that need to be installed:

- `python-benedict==0.24.0`
- `python-can==3.3.4`
- `umodbus==1.0.4`

If this is a new virtual environment, install the dependencies by copying each of the lines in the list of dependencies to your `requirements.txt` and install using the following invocation (**make sure the environment is activated**):

```
python3 -m pip install -r requirements.txt
```

If you are unsure if dependencies are installed, run the command anyway (it is idempotent if the dependencies already exist).

2 Starting Modbus Server

The following commends will only run on the device that is running the Modbus server (ie. the R3000 attached to the CAN bus). To run these, `ssh` to the remote device with the following invocation (where `192.168.20.xx` stands in for the specific IP of the server):

```
ssh anzen@192.168.20.xx
```

Enter the password when asked.

2.1 Checking Server Status

The Modbus server should start upon bootup of the R3000. If you are unsure if the Modbus server is running on the R3000 you are using, run the following command:

```
sudo systemctl status modbus_server.service
```

If it is running you should see something like the following:

```
●
modbus_server.service - Initializes Modbus server
  Loaded: loaded (/etc/systemd/system/modbus_server.service; enabled; vendor
      preset: enabled)
  Active: active (running) since Tue 2021-06-15 10:26:40 EDT; 44min ago
Main PID: 643 (python)
   Tasks: 1 (limit: 2062)
  CGroup: /system.slice/modbus_server.service└─
          643 /home/anzen/.virtualenv/forservices/bin/python /home/anzen/
              pymodbustcp-server/modbus_server.py
```

You also have to make sure that Modbus is enabled in the data collection. Run the following command:

```
sudo systemctl status fbs_modbus.service
```

You have to look closer to see if Modbus is enabled here:

```
●
fbs_modbus.service - Collect and transmit data on Modbus
  Loaded: loaded (/etc/systemd/system/fbs_modbus.service; enabled; vendor preset:
      enabled)
  Active: active (running) since Tue 2021-06-15 10:26:40 EDT; 3h 30min ago
Main PID: 648 (python)
   Tasks: 1 (limit: 2062)
  CGroup: /system.slice/fbs_modbus.service└─
          648 /home/anzen/.virtualenv/forservices/bin/python /home/anzen/
              pymodbustcp-server/fbs_modbus.py
```

2.2 Restarting Modbus Server

If for whatever reason you need to restart the service, restart it with the following set of invocations:

```
sudo systemctl stop fbs_modbus.service
sudo systemctl daemon-reload
sudo systemctl start fbs_modbus.service
```

Whenever enabling a service, always query its status to make sure it is in fact running.

3 Updating Modbus Service

If the Modbus service needs to be updated, first update the Github repository by invoking:

```
cd ~/pymodbustcp-server
git pull origin
```

Enter your Github user name and password when prompted and the repository will update itself to the latest version.

The services will need to then be restarted, which can be done with the following set of invocations:

```
sudo systemctl stop fbs_modbus.service
sudo systemctl stop modbus_server.service
sudo systemctl daemon-reload
sudo systemctl start fbs_modbus.service
sudo systemctl start modbus_server.service
```

4 Reading Modbus Information

The Modbus data store can be queried both locally on the R3000 or remotely from another computer.

4.1 Local Queries

Command Line Queries

If querying for information **on the same device the server is running on**, go to the folder containing the Modbus Github repository by invoking:

```
cd ~/pymodbustcp-server
```

or, alternatively, just invoke `modgit` and you will be taken to the relevant folder. If you are using a virtual environment then activate it now.

To read the data, start the modbus client with the following invocation:

```
python3 modbus_client.py -d
```

The `-d` flag indicates that you would like the returned information to be displayed on the terminal. Without this flag enabled the data will still be queried, jut not printed to `stdout`. You do not need to pass an address because the client will read from `localhost` by default.

TUI (Terminal User Interface)

There is also a rough preliminary graphical representation of the data available. It can be run by invoking (while in the virtual environment):

```
python3 tui-object.py
```

4.2 Remote Queries

If querying for information **on a different device than the server is running on**, first get the IP of the device the server is running on. For this example use the generic `192.168.20.xx`. Then, go to the folder where you cloned the Modbus Github repository. If you are using a virtual environment then activate it now.

To read the data, start the modbus client with the following invocation:

```
python3 modbus_client.py -a 192.168.20.xx -d
```

The `-d` flag indicates that you would like the returned information to be displayed on the terminal. Without this flag enabled the data will still be queried, jut not printed to `stdout`. The `-a` flag indicates that data will be read from an external device and the next argument is the server address.

4.3 Returned Information

The return will look like the following:

```
Values received successfully: [4974, 0, 244, 38]
Values received successfully: [74, 66, 72, 74, 103]
Values received successfully: [16000, 16000, 16000]
Values received successfully: [0, 7, 5, 61782, 17016, 16000, 16616, 32, 61, 2021,
    6, 15, 14, 47, 41]
Values received successfully: [13141, 13128, 13126, 13151, 13134, 13129, 13132,
    13128, 13096, 13107, 13110, 13118, 13112, 13122, 12894, 13092, 13137, 13106,
    13111, 13103, 13121, 13125, 13121, 13116, 13109, 13026, 13067, 13111, 13089,
    13120, 13118, 12460, 13104, 13126, 13129, 13119, 13128, 13130]
Values received successfully: [72, 72, 74, 70, 72, 72, 74, 71, 73, 73, 74, 71, 74,
    75, 74, 71, 73, 74, 73, 73, 73, 72, 74, 73, 74, 87, 75, 75, 73, 72, 73, 73, 72,
     75, 73, 73, 73, 71]
Values received successfully: [1, 0, 1, 0]
Values received successfully: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0]
```

The returns are as follows:

```
Registers 40000-40003
Registers 41000-41004
Registers 43000-43002
Registers 42000-42014
Registers 40004-40042 (Monoblock Voltages)
Registers 41005-41043 (Monoblock Temperatures)
Coils     40-43
Coils     1-26        (Errors and alarms)
```