

WiFi MQTT Control Relay Thermostat

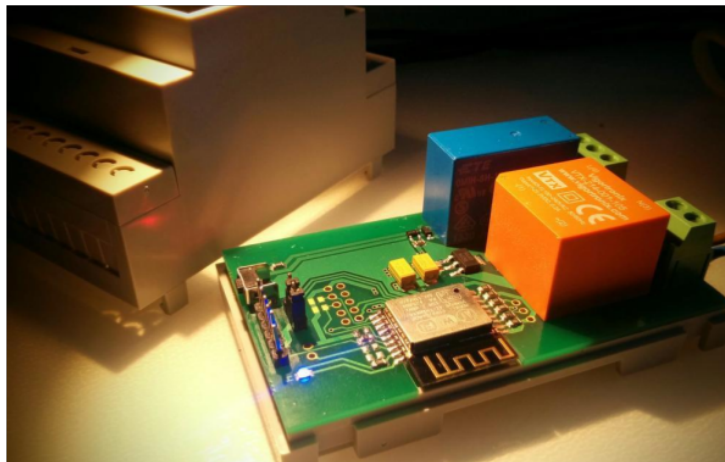
Multi-purpose Wifi connected relay control board. Applications include: remote heating an A/C systems control via nodeRED, openHAB and Android Tasker etc.

Overview

[View in Shop →](#)

- 1 x High quality 16A Relay (tested 100 switches @ 3x rated current)
- Powered by the popular [ESP8266](#) WiFi SoC
- OTA Firmware upload function
- Powered direct from 110-240V AC via isolated on-board PSU
- Built-in web server with mobile device friendly UI and
- HTTP Control API
- Thermostat function with weekly scheduling
- Manual relay control via web interface
- [MQTT](#) control
- [NTP](#) for network time and scheduling functionality
- Web server settings: including HTTP port & authentication setup
- Temperature sensor support - 1 x sensor included

MQTT ESP8266 WiFi Relay / Thermostat



MQTT Settings

MQTT enabled?: ☒

Host: 192.168.0.12

Port: 1883

Keepalive (sec): 120

Device ID: VESIV4Z46U

User: emonpi

Password: *****

Use TLS?: ☐

Subscribe topic: heating/control/#

Publish topic: heating/status/

WiFi Settings

Current WiFi mode: STA

To connect to a WiFi network, please select one of the detected networks...

- TALKTALK-C917B8
- Crumble

Current IP: 192.168.0.36

WiFi password: *****

Mode: Static IP

IP address: 192.168.0.36

IP mask: 255.255.255.0

GW address: 192.168.0.1

Heating

SET POINT:

- **19.5°C** +

Temperature: NaN°C

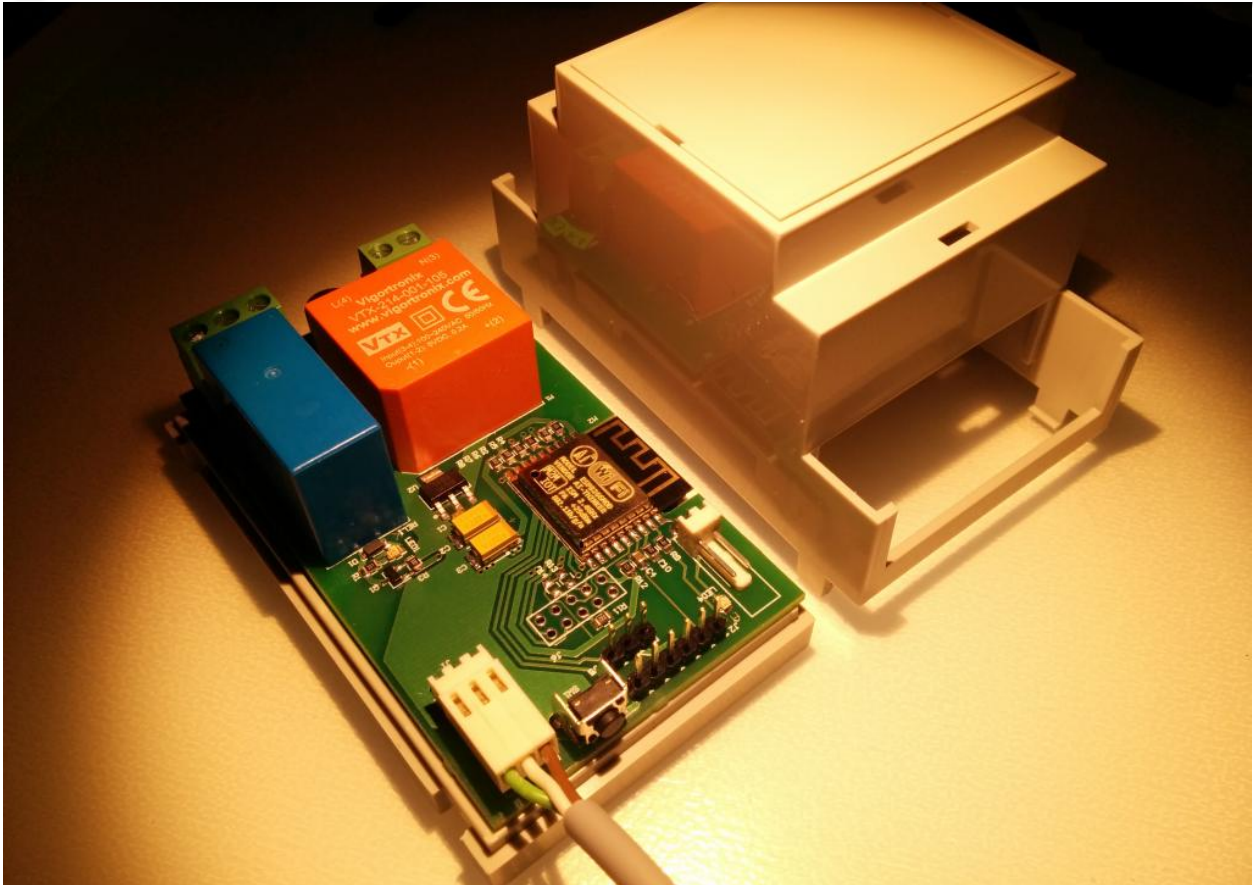
Schedule

TUE 22:43

MON	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TUE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
WED	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
THU	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FRI	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SAT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
SUN	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Relay can be controlled by publishing 1 or 0 to the MQTT control topic, default: `'heating/control/relay/1'`

<https://guide.openenergymonitor.org/integrations/mqtt-relay/>



Contents

[View in Shop →](#)

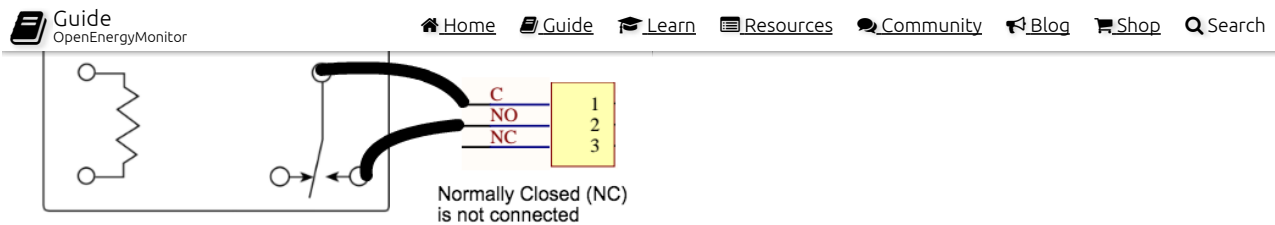
1. [Physical Connections](#)
2. [Network Setup](#)
3. [Web UI Control](#)
4. [Thermostat Scheduler](#)
5. [HTTP API](#)
6. [MQTT API](#)
7. [Other Settings](#)
8. [Firmware Update](#)
9. [Programming](#)
10. [Schematic](#)
11. [Dimensions](#)

Physical Connections

The board connects to and controls high voltage, knowledge and attention is required when installing to prevent electrical shock

1. Connect 110V-240V AC into the dual terminal block to power the unit (polarity does not matter) . There is a 0.25A fuse (MST250 slow blow) on the board, however an external fuse is also recommended.
2. The relay contacts are isolated from the main supply. The 3 x relay terminals are connected as follows:



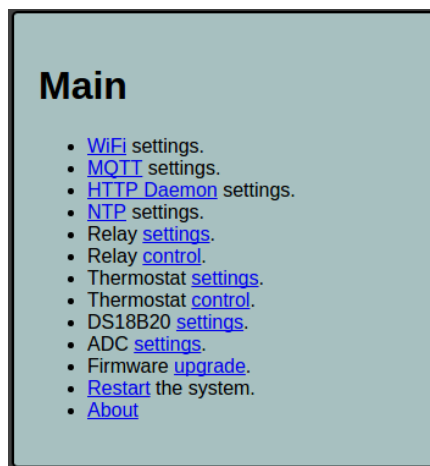


Your safety is your responsibility. Ensure all contacts are fully isolated before installing. If you have any doubts, seek professional assistance. Ensure power cables are securely wired into relay terminal blocks and are held captive externally.

Network Setup

On first power the unit will enter WiFi Access Point (AP) mode serving it's own WiFi hotspot called 'ESP_XXX', where XXX is the units MAC address. AP mode is only designed to setup the unit, operation of the thermostat requires connection to a WiFi network with internet access to obtain NTP time.

1. Connect to the 'OEM_XXX' WiFi network and browse to <http://192.168.10.1>.

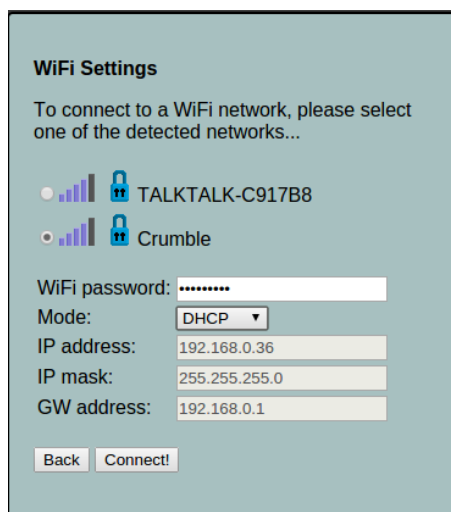


2. Select WiFi Settings, wait for auto scan to populate then select the network to connect to and enter password. If required enter static IP settings or choose DHCP (default).

3. Click Connect!

Unit will now reboot and turn off AP mode. Unit should not be connected to the choose WiFi network. Fing [Android](#) / [iOS](#) app can be used to find it's IP or run terminal command to scan for IP addresses of all connected ESP8266 modules:

```
sudo arp-scan -retry 7 -quiet -localnet -interface=wlan0 | grep -s -i 18:fe:34
```



Turn on WiFi AP / Restore Default Settings

Once the unit is connected to your local Wifi network for security the Wifi AP will automatically be turned off.

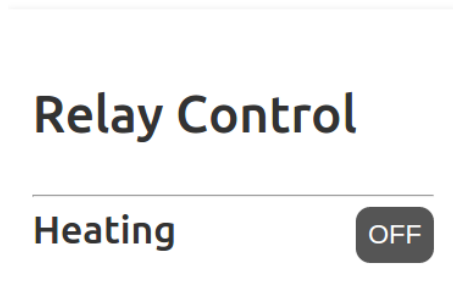
To turn the Wifi AP back on (e.g to scan and connect to a different network) press and hold the reset button for 3 seconds

When in Wifi AP mode a further 3 second press of the reset button will clear all setting and restore to default.

Relay can now be in four ways:

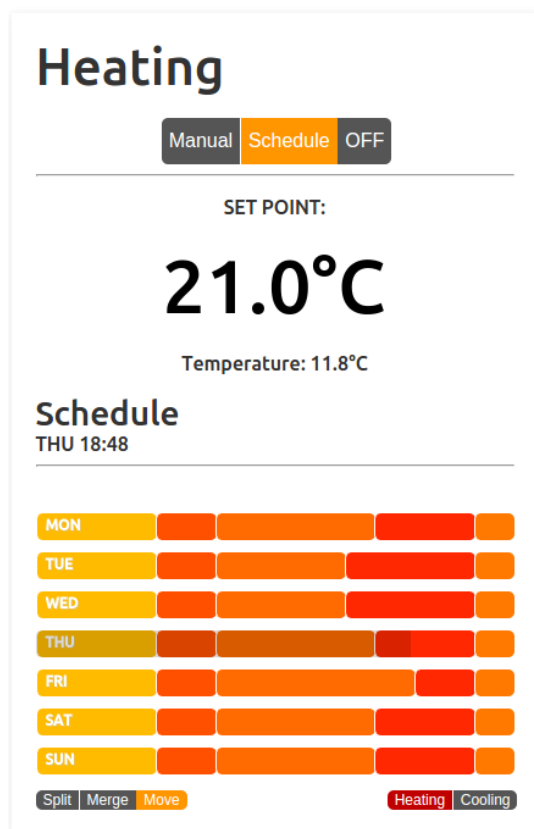
1. Web UI Simple interface

Select Relay Control via web interface. Or browse to <http://control/relay.html>



2. Thermostat Scheduler

Heating or A/C schedule and desired set point (from connected internal DS18B20 temperature sensor) can be set using the Thermostat Scheduler interface



The Thermostat Settings page allow setting of Thermostat Input (currently only DS18B20 sensor is implemented)

Hysteresis is a nice function that prevents the relay from switching on/off frequently around the setpoint. It is defined in 100ths of a degree, i.e. 500 means 5 degrees C. See fig below:

The “high” defines how much higher than the desired temperature the temperature reading has to be in order for the relay to switch off

The “low” defines how much lower than the desired temperature the temperature reading has to be in order for the relay to switch on

Guide
OpenEnergyMonitor

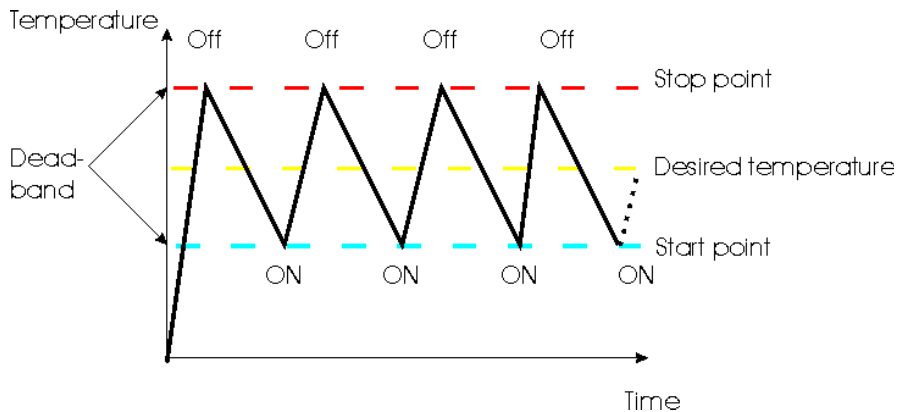
Home Guide Learn Resources Community Blog Shop Search

Thermostat Settings

Thermostat 1 input:

Thermostat 1 hysteresis high: (in hundreds of a degree, 50 means 0.5 degrees)

Thermostat 1 hysteresis low: (in hundreds of a degree, 50 means 0.5 degrees)



3. HTTP API

Return Status

`http://<IP-ADDRESS>/control/relay.cgi`

Returns JSON:

```
"relay1": 0, "relay1name": "Heating"}
```

Turn Relay on:

`http://<IP-ADDRESS>/control/relay.cgi?relay1=1`

Returns:

```
'OK'
```

HTTPD Settings allow the HTTP port to be set

Disable HTTPD in normal working mode - Turns off the HTTP service if not in "setup" mode, useful when you want to only use the MQTT interface and not worry about exposing the HTTP configuration UI to the wild

HTTP Auth allows for setting separate Admin and User password. Non-admin user cannot make any changes to the config.

HTTPD Settings

HTTPD Port:

Disable HTTPD in normal working mode?: ☐

HTTP Basic Auth enabled?: ☐

Admin user:

Password:

User:

Password:

4. MQTT API

The best (secure and lightweight) way IMHO to control the unit is via MQTT.

Status:

Unit periodically publishes DS18B20 temperature sensor value to the following topic:

heating/status/ds18b20/1

Relay Status after state is changed is published to:

heating/status/relay/1

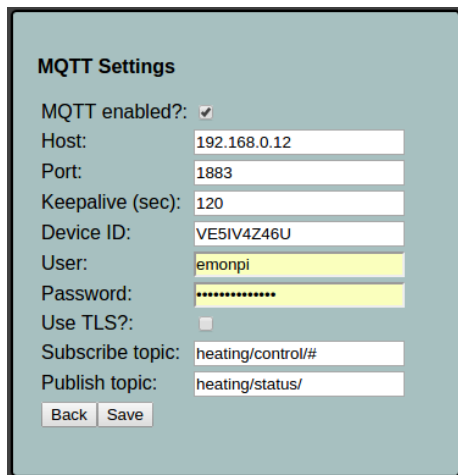
Control

Relay can be controlled by publishing 1 or 0 to the MQTT control topic, default:

heating/control/relay/1

After a control message has been received (either via MQTT or HTTP) relay will respond with a status MQTT message posted to the status topic (see above).

MQTT topic names are fully configurable, see MQTT Settings:



MQTT Settings

MQTT enabled?: ☒

Host: 192.168.0.12

Port: 1883

Keepalive (sec): 120

Device ID: VE5IV4Z46U

User: emonpi

Password: *****

Use TLS?: ☐

Subscribe topic: heating/control/#

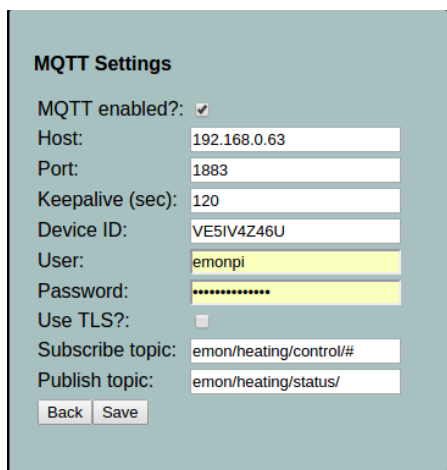
Publish topic: heating/status/

[Back](#) [Save](#)

MQTT with emonPi & Emoncms

The emonPi has a Mosquitto MQTT server running as standard on port 1883. To publish heating / temperature status to Emoncms publish to emonPi MQTT server with the emon/ MQTT prefix e.g:

emon/heating/status/ds18b20/1



MQTT Settings

MQTT enabled?: ☒

Host: 192.168.0.63

Port: 1883

Keepalive (sec): 120

Device ID: VE5IV4Z46U

User: emonpi

Password: *****

Use TLS?: ☐

Subscribe topic: emon/heating/control/#

Publish topic: emon/heating/status/

[Back](#) [Save](#)

Note: Older 1st gen emonPi's have an unauthenticated MQTT server running on localhost with port 1883 closed. Port can be opened and make persistently open with:

```
sudo iptables -A INPUT -p tcp -m tcp --dport 1883 -j ACCEPT
```

```
sudo apt-get install iptables-persistent
```

Newer emonPi's have the MQTT port open by default and authentication turned on with default _user: emonpi, password: emonpimqtt2016. _Port 1883 is closed by default on all home network routers therefor this port is not exposed to the web.

Other Settings

Relay latching - Restores relay state after reset (not recommended)

Relay name - Sets name of Relay to be used in web UI

Relay Pulse time: max "on" time in milliseconds. 0 means disabled (the default setting), setting it to 500 for example will result in the relay auto-switching off after 500ms of on-time. Useful when you want to pulse-contact something, say simulate a button press or make temporary make contact of the relay. Also useful when you want to ensure that the relay will be forced off after certain time, regardless of connectivity loss. I'd say if you control some heater, it would make sense to have the pulse time to 3hrs (10006060*3) for just in case.

DS18B20 Setting - A DS18B20 is used to inform the on-board thermostat. Changing these setting should not be required

ADC Poll - defaults to 0, it is the ADC polling interval. The two-wire connector is connected to the ADC, you can use it to measure temperature, light etc and take action accordingly. 3.3V Max

Firmware Upgrade (recommended way)

The unit has the ability to update firmware over WiFi network via the web interface. Firmware updates will be posted here. There are two branches of the relay board firmware the first is Martin's more advanced firmware based on his recent developments, Martin has not made source code available for this but has given us a compiled binary, available here:

[Current latest firmware: V3190](#) (Oct 2017)

Unzip then select the .bin into units interface then hit upload

Firmware

Firmware version: 2020 / Feb 16 2016 21:53:31 .

Free heap size:24896

Running user2.bin

Uptime: 13 days, 10 hours, 13 minutes, 24 seconds

Firmware file: No file chosen

Programming (advanced)

The ESP8266 can be programmed manually using 3.3V USB to UART / FTDI cable. Jumper 5 should be between 2 and 3 (closer to the LED side). Hold down the push button for the duration of the upload.

Pre compiled binary can be flashed using [esptool](#) :

Note: the recommended way to update firmware is via the web uploader (see above)

```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_freq 80m --flash_mode qio --flash_size 16m-c1 0x1000 oem_v2088.bin
```

The FTDI connection has additional jumper so second UART can be used as debug when the first one is used by the extension port (e.g. RFM69Pi - yet to be implemented).

Alternative Firmware

The second option is our adaptation of Martin's original 3 channel relay firmware, with relay 2 and 3 removed. [This is open source and available here:](#)

To upload this code, in the 1ch_relay directory run:

```
"line-height: 20.8px;">$ sudo make
```

and then to upload with a USB to UART cable:

```
$ sudo sh burn.sh
```

Hold down the push button for the duration of the upload, note there are 4 stages to the upload.

Further development: It is our intention to further develop this open source version of the firmware, if your interested in helping please get in touch.

Bootloader re-load

You can re-flash the bootloader ([download.zip](#)) like this:

```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_freq 80m --flash_mode qio --flash_size 16m-c1 0x00000 "boot_v1.5.bin"
```

flashing the firmware itself:

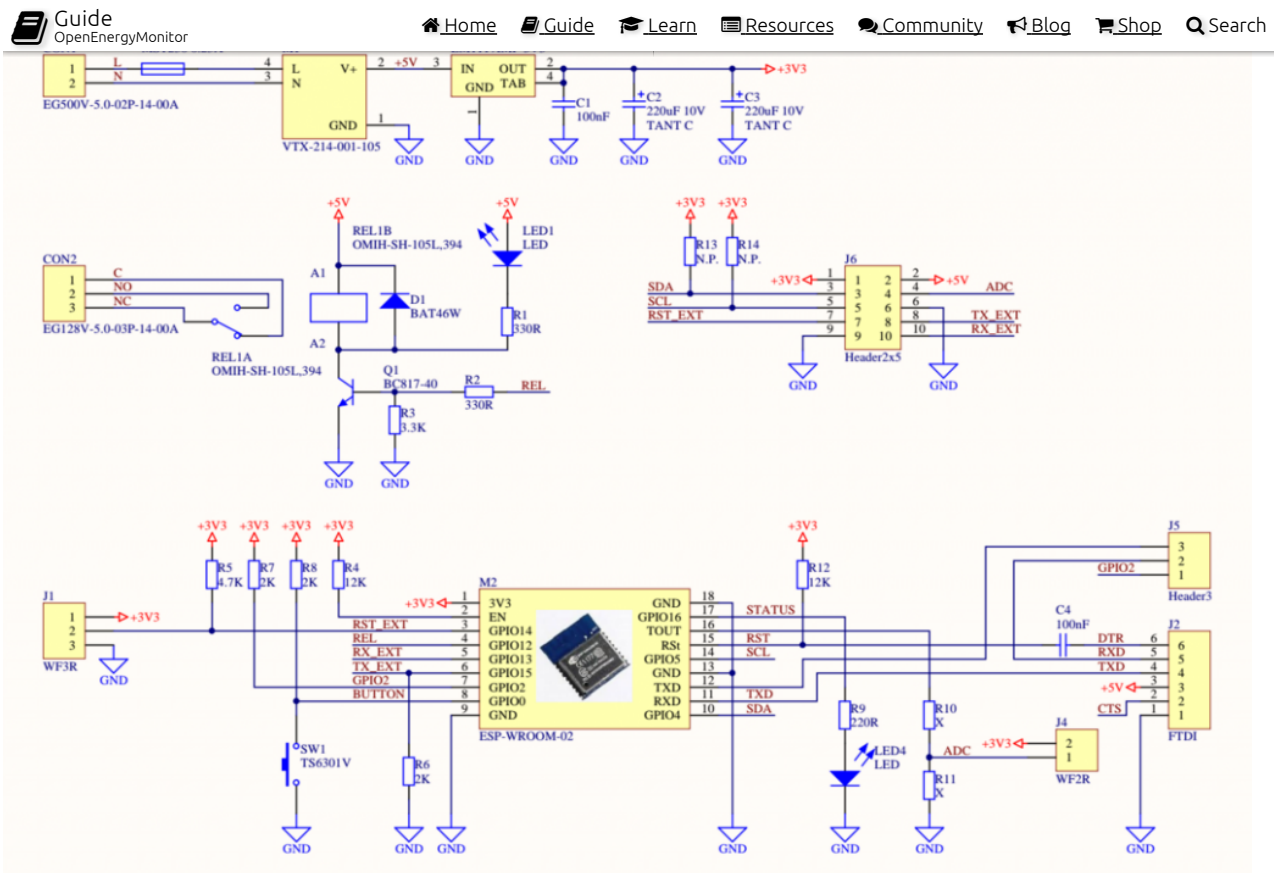
```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_freq 80m --flash_mode qio --flash_size 16m-c1 0x1000 oem.v2088.bin
```

If you have modified other system areas of the flash, this may also cause trouble.. try resetting them and then re-eating the bootloader-firmware update again:

```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_freq 80m --flash_mode qio --flash_size 16m-c1 \ 0x00000 blank.bin \ 0x01000 blank.bin \ 0x7C000 esp_init_data_default.bin \ 0x7D000 blank.bin \ 0x7E000 blank.bin \ 0x7F000 blank.bin \ 0x80000 blank.bin \ 0xFE000 blank.bin \ 0x100000 blank.bin \ 0x101000 blank.bin \ 0x1FC000 esp_init_data_default.bin \ 0x1FD000 blank.bin \ 0x1FE000 blank.bin \ 0x1FF000 blank.bin \ 0x3FC000 esp_init_data_default.bin \ 0x3FD000 blank.bin \ 0x3FE000 blank.bin \
```

Note: flash address 0xFC000 contains board identification information. do not erase or modify it. Check debug serial output.

Schematic



Physical Dimensions

PCB: 87mm x 50mm

Credits

This unit has been designed by [Martin Harizanov](#).

Code from the following sources has been used in this project:

- The ESP-HTTPD project by Jeroen, beerware license
- The ESP-MQTT project by Tuan PM, MIT license
- JSON jsmn library, Z Serge, MIT license
- Open heating controller/thermostat scheduler by Trystan Lea

[Edit on GitHub](#)[Edit on GitHub Prose](#)

Contents

- Setup
 - [1. System Overview](#)
 - [2. Connect](#)
 - [3. Install](#)
 - [4. Log Locally](#)
 - [5. Log Remotely](#)
 - [6. Dashboards](#)
 - [7. Energy Sensing Node\(s\)](#)
 - [emonTx](#)
 - [IoTaWatt](#)
 - [8. Temperature Node\(s\)](#)
 - [9. Add Optical Pulse Sensor](#)
 - [10. Import / Backup](#)
 - [11. Use in North America](#)
 - [12. Troubleshooting](#)
 - [13. Remote Access](#)
- Emoncms
 - [View Graphs](#)
 - [Daily kWh](#)
 - [Daily Averages](#)
 - [Exporting CSV](#)
 - [Histograms](#)
 - [Emoncms API](#)
- Applications



- [Integrations](#)
 - [Electric Vehicle Charging](#)
 - [User Guide Setup](#)
 - [Node-RED](#)
 - [OpenHAB](#)
 - [Control Relay](#)
 - [LightWave RF Control](#)
- [Technical](#)
 - [Overview](#)
 - [Specifications](#)
 - [Service Credentials](#)
 - [MQTT](#)
 - [Firmware Modification](#)
 - [Resources](#)
- [Support](#)
 - [Community Forum](#)
 - [Contact](#)

Website powered by [Jekyll](#) and [Oscalite](#).
Hosted by [GitHub](#) and served by [CloudFlare](#).

- [Home](#)
- [Guide](#)
- [Learn](#)
- [Resources](#)
- [Community](#)
- [Blog](#)
- [Shop](#)
- [Search](#)