



## ESP8266 Instructions

Document version **1.0.0**  
Date Released **22/03/2023**  
Author **Anthony Musgrove**  
Contact **anthony@labworx.au**

So you've heard the great news .. OpenChill can now be installed on a simple mini ESP8266. Well you've heard right. Despite the small footprint, the ESP8266 is actually quite a powerful unit, encapsulating WIFI! So the considerations when porting to ESP8266 were that this chip doesn't have enough pins for an LCD screen with all our other stuff – not a problem, we eliminate that requirement by publishing a simple web portal over wifi so you can monitor OpenChill on your phone or computer, or connect the json data stream directly into compatible software that you can make do with it how you please.

At this stage, the firmware only provides a simple web portal, with statistics output – there's no configuration changes, etc, so need not worry that it isn't yet password protected. This can come later.

The other consideration we had to note was the fact that this chip is 3.3v level logic – which means it will NOT switch a 5 volt relay for the compressor – no big issues here though. By far the simplest and most elegant solution here would be to use a solid state relay such as the SSR40 (yeah yeah, 40 amps, a bit overkill Anthony), oh well, they're cheap as chips – \$8-\$15 AUD will get you an SSR40 – which gives you the ability to switch up to 40 amps (yes, 40!) at 240volts AC, and will support a switching gate voltage of just 3 volts! Perfect ...!

So lets get down to the nitty gritty – this document isn't yet complete. It'll be updated after I finish moving house! 😊

## Editing the firmware to support ESP8266

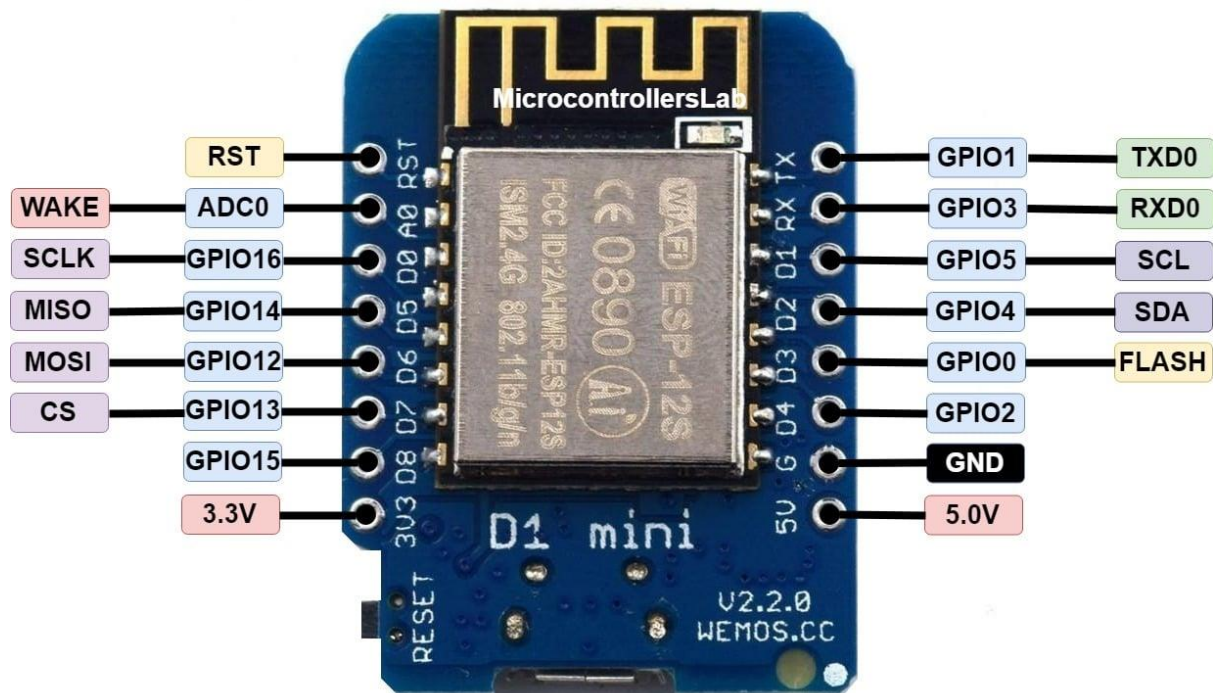
First step is to check that the firmware is correctly configured, before you compile it. With the ESP8266, it is a requirement that you set the SSID and Password for your WIFI before compiling and flashing the firmware (at this stage anyway – later we can integrate the settings into the EEPROM and have them modifiable via the web portal – all will come in good time).

**First step – define the board name** `#define OPENCHILL_ESP8266`

(this is **instead** of any of the others, for example `OPENCHILL_HARDWARE_V1` or `MKS_BASE_1_0`).

By defining the board type as `OPENCHILL_ESP8266`, the firmware will then include and compile the ESP8266 libraries, Wifi client, Webserver and DNS libraries). It will also define the CPU map (pinout wiring) as follows :-

<b>D0</b>	Unused	Spare (for now)
<b>D1</b>	Unused	Spare (for now)
<b>D2</b>	Unused	Spare (for now)
<b>D3</b>	Override Switch	Input Pullup, switch LOW to activate Override switch – this will keep the fridge compressor ON (or D4 -> HIGH) for as long as the switch is enabled.
<b>D4</b>	Fridge/Compressor Relay	3.3v output logic for the Solid State Relay gate
<b>D5</b>	Buzzer/Audio	For alerts/alarms. If you don't add a buzzer, no big deal – you just won't hear it if it detects the reservoir temperature above the maximum allowed value
<b>D6</b>	Safety Cutoff Signal	Not yet used – will be soon
<b>D7</b>	Laser In Use Signal	Not yet used – will be soon
<b>D8</b>	DHT11/22/etc Sensor Signal	The (S)ignal pin from your DHT11 or DHT22 sensor
<b>A0</b>	Reservoir Temperature Signal	Analog Input pin for a temperature probe – I made my probe from a simple thermocouple in heatshrink then covered in foam. Depending on your thermocouple you use you may need to change the mapping code manually to support its range – we need to make this more versatile in the future



You need to configure your wifi SSID and PASSWORD. If you're unsure of these values they may be on a sticker on your router.

Scroll down to line 183 (approx.) and configure the following lines :-

```
#define WIFI_SSID      "YourWIFISSIDHere"
#define WIFI_PASSWD    "YourWIFIPasswordHere"
```

Along with configuring the wifi SSID and PASSWORD – you can also specify a static lan IP address if you wish to have the web portal / json data stream open on the same IP address always – you can also override the default web port 80.

## To set a static IP

Scroll down to line 186 (approx.) of the firmware and find the line

```
#define IS_STATIC_IP
```

Once you've told OpenChill firmware to use a static IP, you need to set the values from approx. line 191 to line 212. They include IP, DNS, Gateway and Subnet Mask, example shown below:

```
/* Portal IP Address (Example: 192.168.1.75) */
#define IP_0 192
#define IP_1 168
#define IP_2 1
#define IP_3 85

/* DNS IP Address (Example: 192.168.1.1) */
#define DNS_0 192
#define DNS_1 168
#define DNS_2 1
#define DNS_3 1

/* Gateway/Router Address (Example: 192.168.1.1) */
#define GW_0 192
#define GW_1 168
#define GW_2 1
#define GW_3 1

/* Subnet Mask (Example: 255.255.255.0) */
#define SUBN_0 255
#define SUBN_1 255
#define SUBN_2 255
#define SUBN_3 0
```

This will tell the firmware to publish the simple web portal on IP address 192.168.1.85, using DNS server 192.168.1.1, Gateway (or your wifi router address) 192.168.1.1, and a subnet mask of 255.255.255.0 – networking configuration is out of scope for this document – If you need to clarify any of these settings you need to either speak with your network administrator or read up on your router settings (and local area network settings). Who knows, we might do some more in depth tutorials on this stuff at a later stage.

## Changing the default port for the portal

If you want to take the portal off the default web (http) port (80), you can modify the line (approx. 214):

```
#define WEB_PORT 80
```

Once you've got your settings modified, compile the firmware using the Arduino IDE, while your ESP8266 is connected via USB (or however you wish to achieve programming your ESP8266).

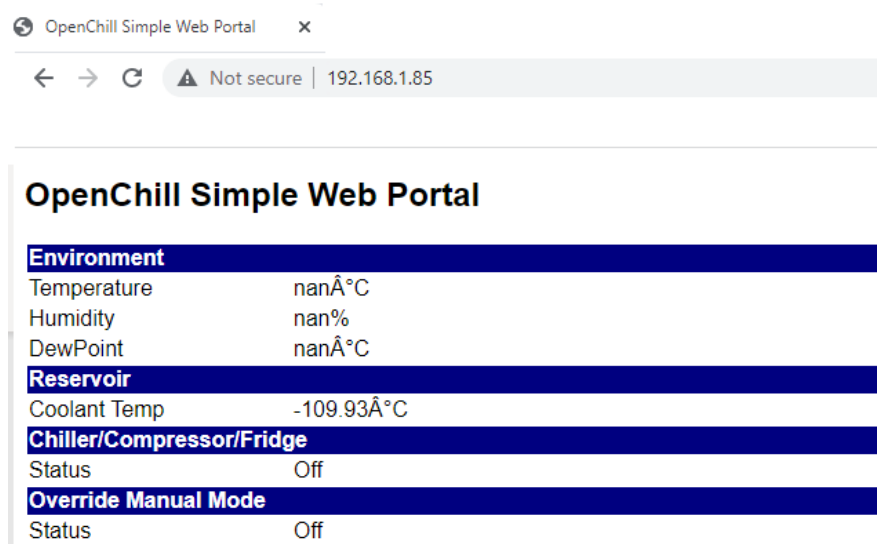
When you boot it up, you should see a simple web portal on your portal address :

[http://static\\_configured\\_ip:port](http://static_configured_ip:port)

Note – if your port is left at default 80, you do not need to specify a port, simply [http://ip\\_address/](http://ip_address/)

Also note – if you did not choose to use a static IP address, you'll need to locate the device on your local network by either IP scanning your net range, or checking 'connected devices' or the like on your wifi router/modem.

The simple portal appears somewhat like the following :-



OpenChill Simple Web Portal	
<b>Environment</b>	
Temperature	nanÅ°C
Humidity	nan%
DewPoint	nanÅ°C
<b>Reservoir</b>	
Coolant Temp	-109.93Å°C
<b>Chiller/Compressor/Fridge</b>	
Status	Off
<b>Override Manual Mode</b>	
Status	Off

**Environment Temperature** and **Humidity** come from your DHT11/22 sensor. The dewpoint is calculated using the dewpoint formula ( $T_d = T \cdot 0((100 - RH/5.)$  where  $T_d$  is dew point temperature (in degrees Celsius),  $T$  is observed temperature (in degrees Celsius), and  $RH$  is relative humidity (in percent).

The values shown in my example above are what you'll typically see without your DHT11/22 sensor connected (nan = Not A Number), and the firmware knows not to attempt to switch on the compressor if these invalid values are received).

**Reservoir** Coolant Temperature is the temperature value received by your temperature probe installed inside your chiller – remember this may be inaccurate and may need calibrating (and even some code changes to the mapping etc) – right now this isn't very versatile so this is a massive TODO for the near future – to get some predefined sensor types and their mapping values into the

firmware. For now though, use a BBQ probe or Meat probe and do some calculations to getting your particular sensor working correctly.

**Chiller/Compressor/Fridge Status** is the indicative value that your logic gate for your solid state relay is ON or OFF – and this subsequently will be what powers your compressor pump or whatever you choose to power with the solid state relay.

**Override Manual Mode Status** is the status of your manual override switch. If you choose to use a manual override switch – I added it as I found it handy when my temperature probe is inaccurate (because of calibration issues) and I really needed to get laser jobs done, I'd just flick it over to manual mode and monitor the temperature manually – however I rarely need this now as I've got my implementation configured/dialed in pretty well.

### How do I just access the raw stats?

If you just want a JSON formatted feed of the vitals from openchill, such that you can connect it to your laser software or some other software that can use this data, etc, you can open the stats URL directly at:

[http://openchill\\_ip\\_address/stats](http://openchill_ip_address/stats) (if you changed your default web port from 80, you'll also need to specify the port – ie: [http://openchill\\_ip\\_address:port/stats](http://openchill_ip_address:port/stats)).

You'll be presented with the JSON formatted stats data which will appear like the following -:

```
{ "environment":  
  { "temperature": "nan",  
    "humidity": "nan",  
    "dewpoint": "nan"},  
  "reservior_temperature": "-109.93",  
  "fridge_status": "Off",  
  "override_status": "Off" }
```

When you have sensors connected and its working correctly, you'll see values instead of nan (not a number) for temperature, humidity and dewpoint.