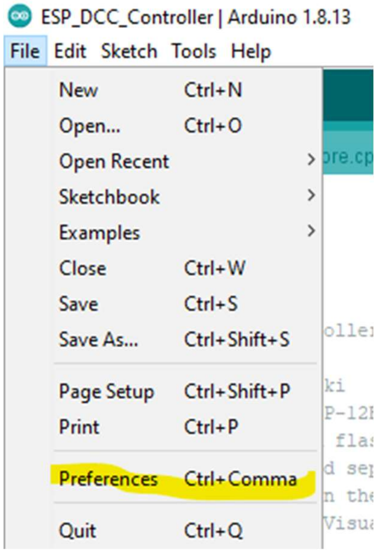
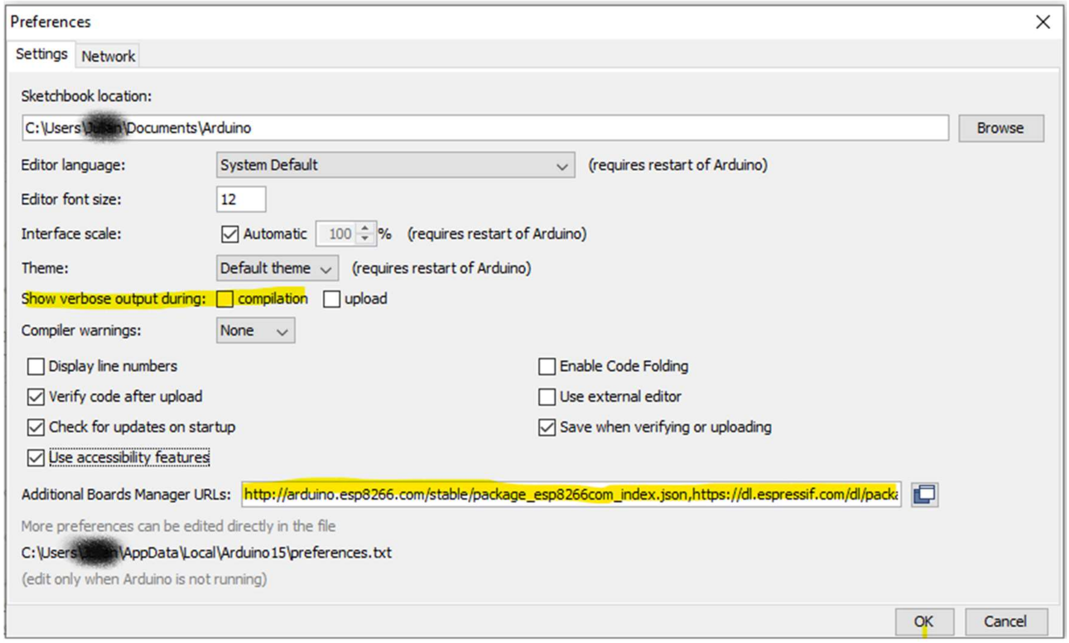
Arduino IDE set-up for DCC controller

Step 1. IDE environment set-up. Load the ESP boards.

When you first instal the Arduino IDE, it only supports ARM based boards. We need to add support for ESP based boards. Navigate to File… Preferences





Sketchbook location: C:\Users\[redacted]\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: □ compilation □ upload

Compiler warnings: None

□ Display line numbers
☑ Verify code after upload
☑ Check for updates on startup
☑ Use accessibility features
□ Enable Code Folding
□ Use external editor
☑ Save when verifying or uploading

Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json,https://dl.espressif.com/dl/packa

More preferences can be edited directly in the file
C:\Users\[redacted]\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

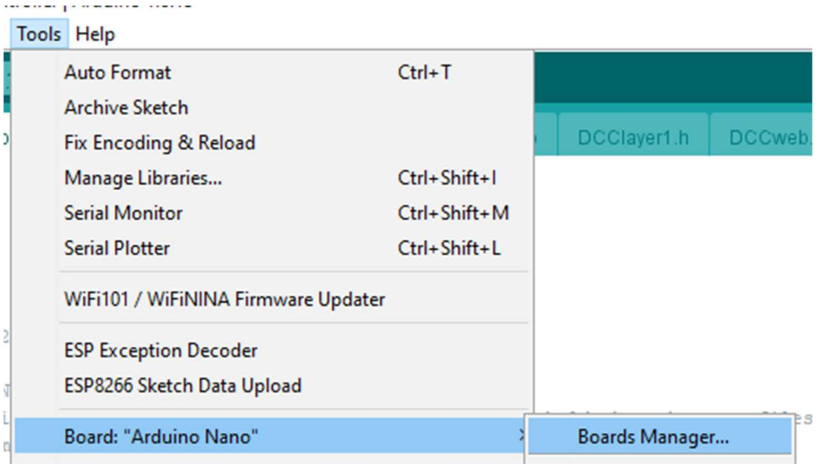
Type this line below into the Additional Boards Manager URLS box. Note there are underscores in it, no spaces.

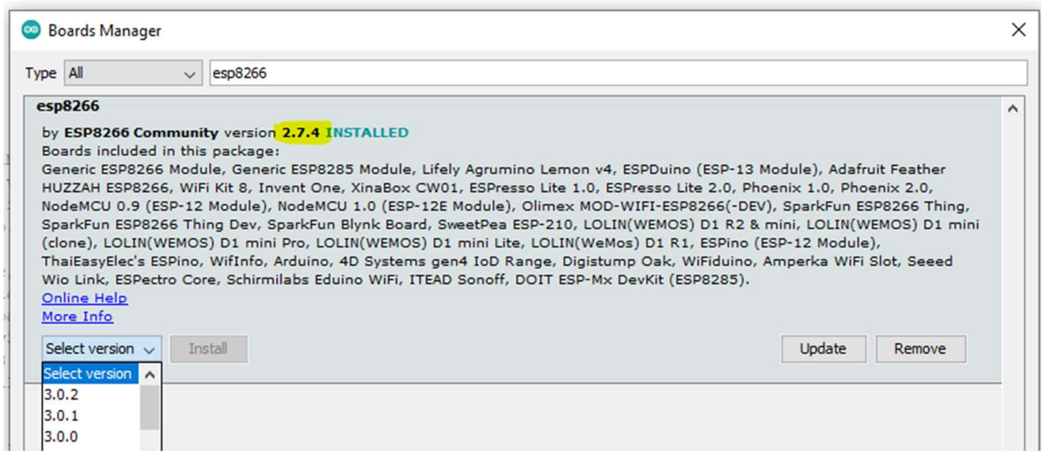
http://arduino.esp8266.com/stable/package_esp8266com_index.json,https://dl.espressif.com/dl/package_esp32_index.json

Also check the box that says Show Verbose during compilation. This gives us more information if something fails during the compilation.

Note that the line above adds support for both esp8266 devices and the newer esp32. The two json strings are separated by a comma.

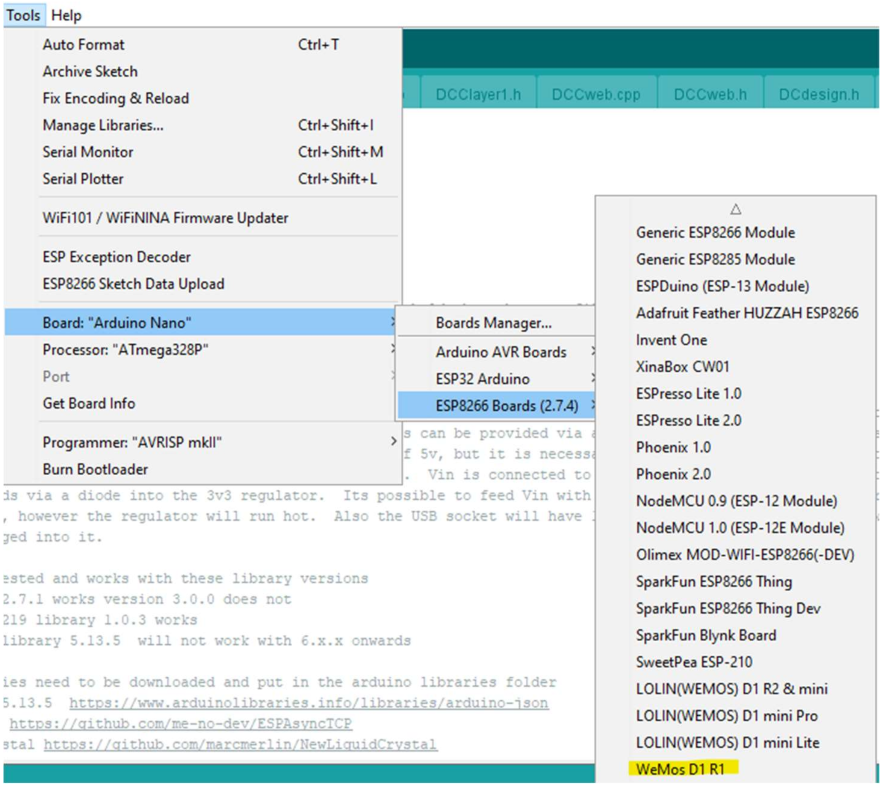
Now select board **version 2.7.4** from boards manager


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP Exception Decoder
ESP8266 Sketch Data Upload
Board: "Arduino Nano"
Boards Manager...
DCClayer1.h
DCCweb.



esp8266
by **ESP8266 Community** version **2.7.4** INSTALLED
Boards included in this package:
Generic ESP8266 Module, Generic ESP8285 Module, Lifely Agrumino Lemon v4, ESPDuino (ESP-13 Module), Adafruit Feather HUZZAH ESP8266, WiFi Kit 8, Invent One, XinaBox CW01, ESPresso Lite 1.0, ESPresso Lite 2.0, Phoenix 1.0, Phoenix 2.0, NodeMCU 0.9 (ESP-12 Module), NodeMCU 1.0 (ESP-12E Module), Olimex MOD-WIFI-ESP8266(-DEV), SparkFun ESP8266 Thing, SparkFun ESP8266 Thing Dev, SparkFun Blynk Board, SweetPea ESP-210, LOLIN(WEMOS) D1 R2 & mini, LOLIN(WEMOS) D1 mini (clone), LOLIN(WEMOS) D1 mini Pro, LOLIN(WEMOS) D1 mini Lite, LOLIN(WeMos) D1 R1, ESPino (ESP-12 Module), ThaiEasyElec's ESPino, WifInfo, Arduino, 4D Systems gen4 IoD Range, Digistump Oak, WiFiduino, Amperka WiFi Slot, Seeed Wio Link, ESPectro Core, Schirmilabs Eduino WiFi, ITEAD Sonoff, DOIT ESP-Mx DevKit (ESP8285).
Online Help
More Info

Install version 2.7.4. This works. Version 3.0.0 and higher does not work for this project.
Now, back in the Tools menu, select the board you will be using. For this project it will be either a nodeMCU 1.0 or a WeMos D1R1



```
s can be provided via
f 5v, but it is necess
.  Vin is connected to
ds via a diode into the 3v3 regulator.  Its possible to feed Vin with
, however the regulator will run hot.  Also the USB socket will have
ged into it.

ested and works with these library versions
2.7.1 works version 3.0.0 does not
219 library 1.0.3 works
library 5.13.5  will not work with 6.x.x onwards

ies need to be downloaded and put in the arduino libraries folder
5.13.5  https://www.arduinolibraries.info/libraries/arduino-json
 https://github.com/me-no-dev/ESPAsyncTCP
stal https://github.com/marcmerlin/NewLiquidCrystal
```

Here we select the WeMos D1R1. (changing this from the Nano)

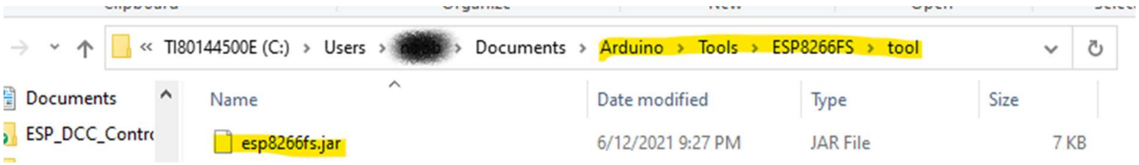
Step 2. IDE environment set-up. Load ESP8266 Sketch Data Upload add-in.

We need to load this add-in to allow us to publish (put) HTML pages and other files on the ESP device. These live in the data folder inside your project folder

https://github.com/esp8266/arduino-esp8266fs-plugin/releases

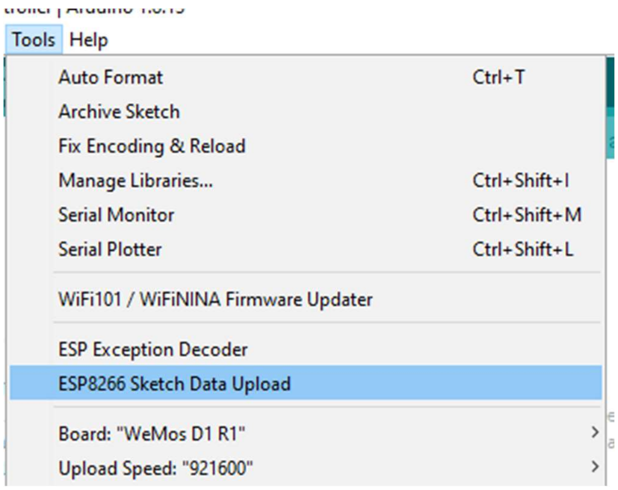
Go to the URL above and download ESP8266FS-0.5.0.zip

Create a Tools folder inside your Arduino folder. Unzip the contents of the zip file to this Tools folder. You should end up with this;


« TI80144500E (C:) › Users › [redacted] › Documents › Arduino › Tools › ESP8266FS › tool

Name	Date modified	Type	Size
esp8266fs.jar	6/12/2021 9:27 PM	JAR File	7 KB

And a new menu option will appear under Tools…


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP Exception Decoder
ESP8266 Sketch Data Upload
Board: "WeMos D1 R1"
Upload Speed: "921600"

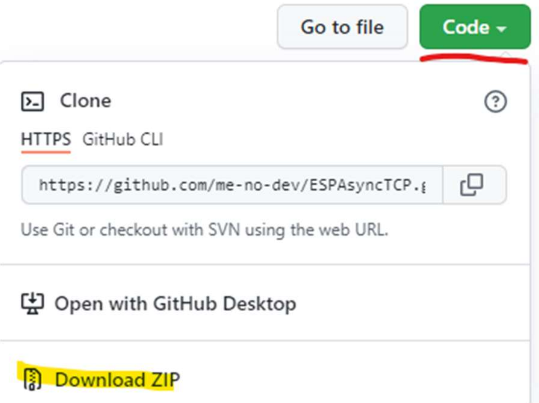
If you invoke that menu option, the IDE will upload the contents of the data folder to the board.

Ok so that's the IDE environment set up for general ESP8266 use, now we need to add some libraries to the Arduino/Libraries folder for this specific project.

Step 3. Download libraries and manually instal.

We need to download these libraries from Github;

https://github.com/me-no-dev/ESPAsyncTCP



```
https://github.com/me-no-dev/ESPAsyncTCP.
```

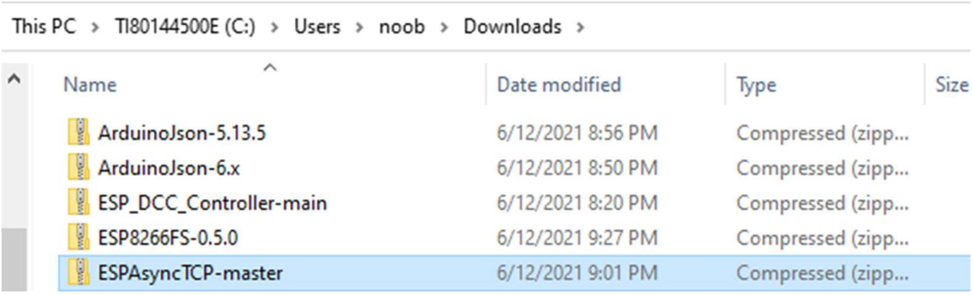
Use Git or checkout with SVN using the web URL.

Click on code, and then download zip. It will go to your downloads folder.

Go into downloads, find the zip, open it and drag the content folder "ESPAsyncTCP" to Arduino/libraries.

If the folder name ends with "-master", then rename it to remove "-master" from the end.

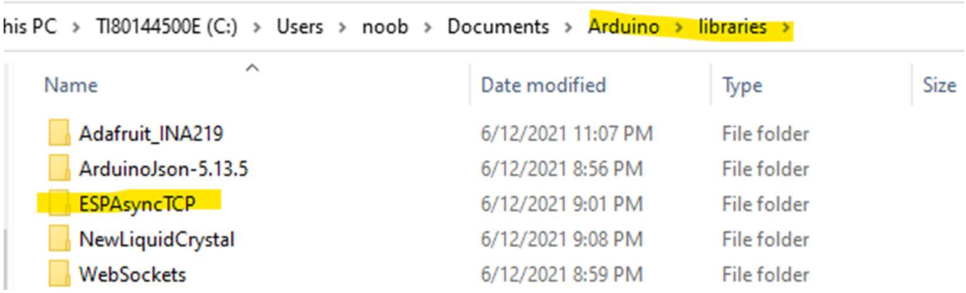
i.e. from downloads



This PC > TI80144500E (C:) > Users > noob > Downloads >

Name	Date modified	Type	Size
ArduinoJson-5.13.5	6/12/2021 8:56 PM	Compressed (zipp...	
ArduinoJson-6.x	6/12/2021 8:50 PM	Compressed (zipp...	
ESP_DCC_Controller-main	6/12/2021 8:20 PM	Compressed (zipp...	
ESP8266FS-0.5.0	6/12/2021 9:27 PM	Compressed (zipp...	
ESPAsyncTCP-master	6/12/2021 9:01 PM	Compressed (zipp...	

Open the .zip for ESPAsyncTCP-master, and drag ESPAsyncTCP-master folder from inside this to Arduino/Libraries



his PC > TI80144500E (C:) > Users > noob > Documents > Arduino > libraries >

Name	Date modified	Type	Size
Adafruit_INA219	6/12/2021 11:07 PM	File folder	
ArduinoJson-5.13.5	6/12/2021 8:56 PM	File folder	
ESPAsyncTCP	6/12/2021 9:01 PM	File folder	
NewLiquidCrystal	6/12/2021 9:08 PM	File folder	
WebSockets	6/12/2021 8:59 PM	File folder	

Note: Arduino/libraries cannot use the .zip version, you need to unzip (drag) the desired folder over.

We also need

https://github.com/fmalpartida/New-LiquidCrystal

Download the zip then drag its content to Arduino/libraries and remove -master ending.

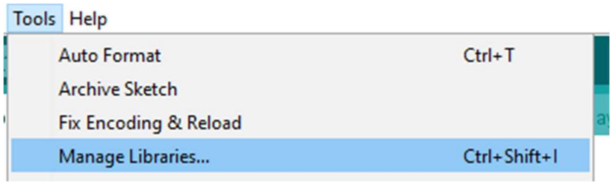
And finally, we need ArduinoJson-5.13.5.zip from the link below

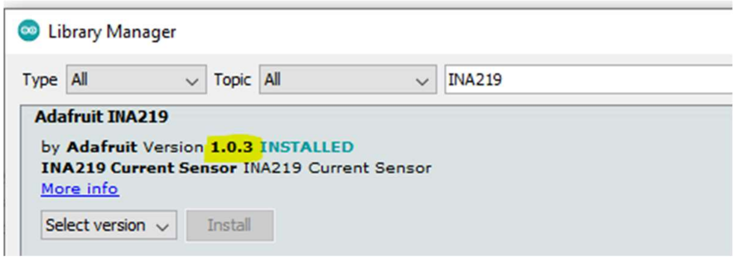
https://www.arduinolibraries.info/libraries/arduino-json

download and then drag the zip contents to Arduino/libraries

Step 4. Instal a couple more libraries using Arduino Library Manager.

We need two more libraries, and these come from the Arduino Library Manager which holds a selection of built-in libraries. Go to Tools… Manage Libraries…





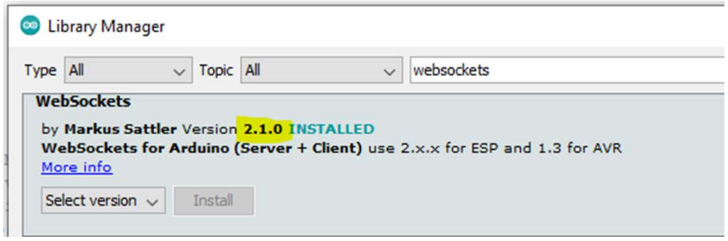
Type All Topic All INA219

Adafruit INA219

by **Adafruit** Version **1.0.3** INSTALLED
INA219 Current Sensor INA219 Current Sensor
More info

Use version 1.0.3 of Adafruit INA219. This works.

And also



Type All Topic All websockets

WebSockets

by **Markus Sattler** Version **2.1.0** INSTALLED
WebSockets for Arduino (Server + Client) use 2.x.x for ESP and 1.3 for AVR
More info

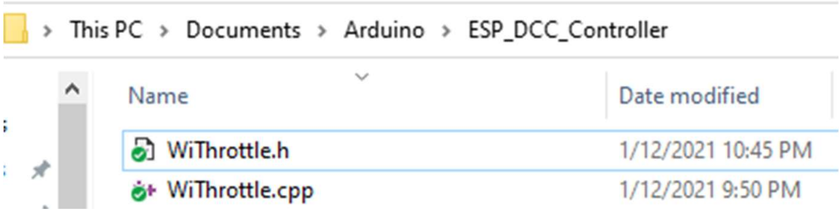
Use version 2.1.0 of WebSockets from **Markus Sattler**, this is tested and working. I have not tested later versions.

OK so that's all the libraries (aka references) that the IDE needs to compile this project.

Step 5. Download the ESP_DCC_Controller project from GitHub and open in IDE.

Go to GitHub and download
https://github.com/computski/ESP_DCC_controller

Click on the green "Code" button, and download the zip.   Then open the zip file and move its contents to the Arduino folder.   Rename the folder to remove the "-main" ending on the folder name. You should end up with a folder ESP_DCC_controller in your Arduino folder. It will contain an .INO file, various .H and .CPP files and a data folder.



Name	Date modified
WiThrottle.h	1/12/2021 10:45 PM
WiThrottle.cpp	1/12/2021 9:50 PM

Double click on the .INO file to open the project in the Arduino IDE.

Before we hit compile, we need to configure to your requirements….

Step 6. Set your requirements in Global.h

This project can support the nodeMCU or WeMos D1R1 and it also can support a number of different power board (motor shield) options, plus it can support devices on an I2C bus such as a current monitor, LCD display and keypad. And finally it can also support a jogwheel (rotary encoder). The most basic build you can do is a WeMos D1R1 and L298 motor shield.

Note the easiest way to disable an option is add a lowercase n in front of its name in the #define statement.

```
#define nNODEMCU_OPTION3
#define nBOARD_ESP12_SHIELD
#define WEMOS_D1R1_AND_L298_SHIELD
```

For example, above NODEMCU_OPTION3 has been disabled with n, the same for nBOARD_ESP12_SHIELD. WEMOS_D1R1_AND_L298_SHIELD is the active option, and this will cause the compiler to use the configuration for this as listed lower down.

To walk through this config:

```
#elif defined(WEMOS_D1R1_AND_L298_SHIELD)

/*Wemos D1-R1 stacked with L298 shield, note that the D1-R2 is a newer model with different pinouts*/
/*Cut the BRAKE jumpers on the L298 shield. These are not required and we don't want them driven by the I2C
pins as it will corrupt the DCC signal.

The board has an Arduino form factor, the pins are as follows
D0 GPIO3   RX
D1 GPIO1   TX
D2 GPIO16  heartbeat and jogwheel pushbutton (active hi)
D3 GPIO5   DCC enable (pwm)
D4 GPIO4   Jog1
D5 GPIO14  DCC signal (dir)
D6 GPIO12  DCC signal (dir)
D7 GPIO13  DCC enable (pwm)
D8 GPIO0   SDA, with 12k pullup
D9 GPIO2   SCL, with 12k pullup
D10 GPIO15 Jog2
```

the above are notes for humans, lets you know which ESP GPIOs will perform which functions. Note that the Arduino D1-D10 to GPIO mappings are different to the nodeMCU D1-D10 to GPIO mappings

```
*/

#define USE_ANALOG_MEASUREMENT

#define ANALOG_SCALING 3.9  //when using A and B in parallel  (2.36 to match multimeter RMS)
```

We will use the AD on the ESP and not an external I2C current monitoring device such as the INA219 disable this with nUSE_ANALOG_MEASUREMENT if you do wish to use an INA219

```
#define  PIN_HEARTBEAT 16  //and jogwheel pushbutton
```

```
#define DCC_PINS \
uint32 dcc_info[4] = { PERIPHS_IO_MUX_MTDI_U,  FUNC_GPIO12, 12 , 0 }; \
uint32 enable_info[4] = { PERIPHS_IO_MUX_MTDI_U,  FUNC_GPIO5, 5 , 0 }; \
uint32 dcc_infoA[4] = { PERIPHS_IO_MUX_MTDI_U,  FUNC_GPIO14, 14 , 0 }; \
uint32 enable_infoA[4] = { PERIPHS_IO_MUX_MTDI_U,  FUNC_GPIO13,13 , 0 };
```

Defines which pins will drive the DCC signals, we have two channels, running in-phase so we can common them together. A-channel is dcc_info[] and B-channel is dcc_infoA[]. These are defined as macros and the backslash is a line-continuation marker.

```
#define  PIN_SCL         2 //12k pullup
#define  PIN_SDA         0  //12k pullup
#define  PIN_JOG1        4
#define  PIN_JOG2        15 //12k pulldown
```

Define the pins (GPIOs) which drive the I2C SCL/SDA and then also the jogwheel inputs 1 and 2

```
#define KEYPAD_ADDRESS 0x21   //pcf8574
```

Used for the optional 4 x 4 matrix keypad, which is scanned using a pcf8574 chip

```
//addr, en,rw,rs,d4,d5,d6,d7,backlight, polarity.   we are using this as a 4 bit device
//my display pinout is rs,rw,e,d0-d7.  only d<4-7> are used. <210>  appears because bits <012> are //mapped
as EN,RW,RS and we need to reorder them per actual order on the hardware, 3 is mapped //to the backlight.
<4-7> appear in that order on the backpack and on the display.

#define BOOTUP_LCD LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  //YwRobot backpack
```

Used to define and configure the I2C backpack that drives the 1602 LCD display (optional), this is soft-configurable and there are several backpacks available whose pin configurations vary.

```
#endif
```

Step 7. Compile and upload to the board.

Now you have configured the board combo you intend using, you can compile the project. If you don't intend to use the 4x4 matrix keypad, and LCD, no problem, leave in their definitions as the software expects to configure them. The system will work fine over WiFi without them.

On the IDE, the tick symbol (verify) is actually "Compile". Click this and you will see various messages appear (provided you enabled Verbose compilation) as the system compiles the various libraries and links it all together. If all works well, and it should if you followed all steps above exactly, then you should see a success message appear. You are now ready to hit the right-arrow (upload) button, but before you do this, check you have selected the correct COM port for the board under the Tools menu.

After a successful upload (use a good quality USB cable) you also need to invoke the **Load ESP8266 Sketch Data** menu option under Tools. This will put the contents of the data folder onto the device (all the HTML pages).

You are done. Open the serial monitor, click the reset button and you should see the device boot and scan for I2C devices. You can now connect to it over Wifi, and its ready to wire up to its power board (motor shield).