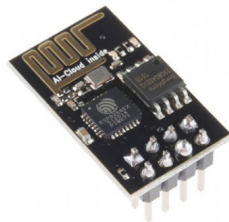




# Getting started with mqTrains Servo Board

## mqTrains OVERVIEW

ESP01 microprocessors are an inexpensive option for controlling model railroad accessories such as turnouts, sensors and signals. They connect quite easily to JMRI using the ESP01's WiFi interface and the MQTT messaging protocol. ESP01s can be purchased from online stores for about a couple of bucks each.

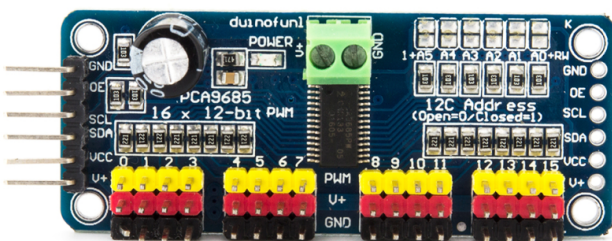


mqTrains sketches have been developed or are in development to handle some common model train requirements such as turnout control, signalling and occupancy detection. Visit [mqTrains.com](http://mqTrains.com) for more information.

ESP devices only use the 2.4 GHz WiFi band. Some common domestic WiFi routers are limited to 16 or 32 concurrent sessions. Check your router specifications if you intend to have a high number of these devices.

## mqTrains Servo Board

The mqTrainsServoPCA sketch is designed to drive turnouts using servo motors attached to a PCA9685 16-channel PWM board, allowing up to 16 servo motors.



## LET'S GET STARTED

These are the main steps to get started with mqTrains Servo Board. More complex features are covered in the mqTrains User Reference.

### 1. Install the MQTT Server Software

#### Linux, Mac and Pi users:

We recommend using a Raspberry Pi for the MQTT Server Software. The Pi Zero W is quite adequate and inexpensive. Follow the vendor instructions to set up the Raspberry Pi. To install Mosquitto Server, use the commands:

```
sudo apt-get update
sudo apt-get install mosquitto
mosquitto-clients
```

Mosquitto version 2 (since December 2020) uses tighter security rules. Create this config file:

```
sudo nano
/etc/mosquitto/conf.d/mosquitto.conf
```

Insert these two lines then press ctrl-o <enter> and ctrl-x

```
listener 1883
allow_anonymous true
```

and restart mosquitto with

```
sudo service mosquitto restart
```

#### Windows users:

Mosquitto can also be installed on most Windows computers. Refer to the [Mosquitto website](http://mosquitto.org) for details. If you install on Windows, you will most likely need to grant Firewall permission. Instructions for granting permission with the Windows Defender Firewall are shown at the end of this document. If you have a Firewall with non-Microsoft security software, refer to the vendor instructions.

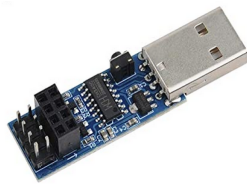
★ At the time of printing, standard Linux installs are mostly version 1.5 or 1.6 and the Windows download is version 2.0.9. These are all fine.

## 2. Install mqTrains on your ESP01

Download the mqTrains binary file from here:  
[mqTrains.com/downloads/](https://mqTrains.com/downloads/)

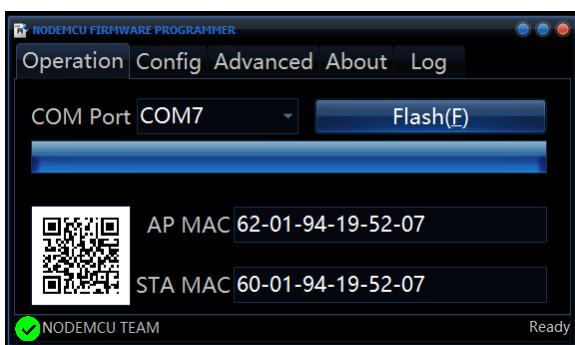
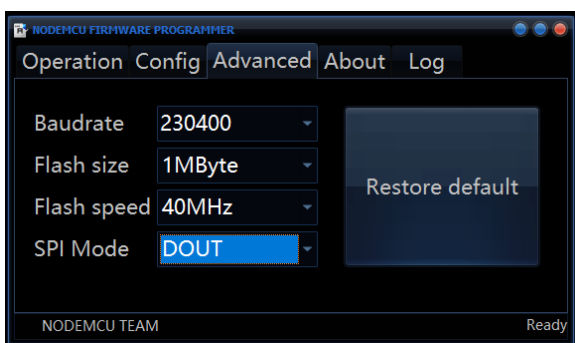
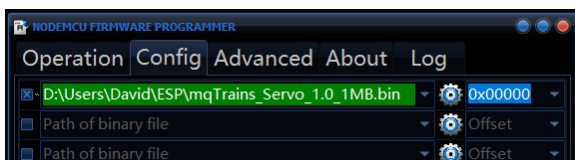
**Windows users:** Download the [nodemcu-flasher](#) program from the appropriate link here: [x64 version](#) (most common) or [x32 version](#) (for old 32-bit computers).

The ESP01 requires an adapter to connect to a computer. We recommend the one shown [here](#). Point the ESP01 towards the USB connector.



Run the installer (double click the exe file downloaded), select the Config tab, click on the wheel in the first row, then navigate to and select the binary file you downloaded.

Select the Advanced tab and select DOUT for SPI Mode:



On the Operation tab, the correct COM Port will most likely be selected for you, change if need be, then click on Flash to start the upload. The blue bar will show the upload progress which takes about 3 minutes. Wait for the green tick and "Ready" message at the bottom of the screen.

**Linux, Mac and Pi users:**

```
sudo apt-get update
```

```
sudo apt-get install python python-pip
```

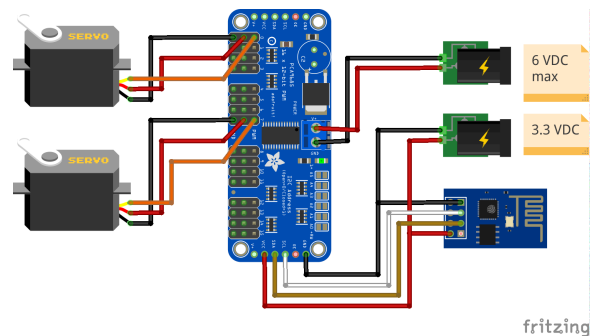
```
sudo pip install esptool
```

And in one line in the folder with the .bin file:

```
esptool.py --chip esp8266 --port /dev/ttyUSB0 --baud 460800 --before default_reset --after hard_reset write_flash 0x0 mqTrains_ServoPCA_0.98_1MB.bin
```

## 3. Wiring your ESP01

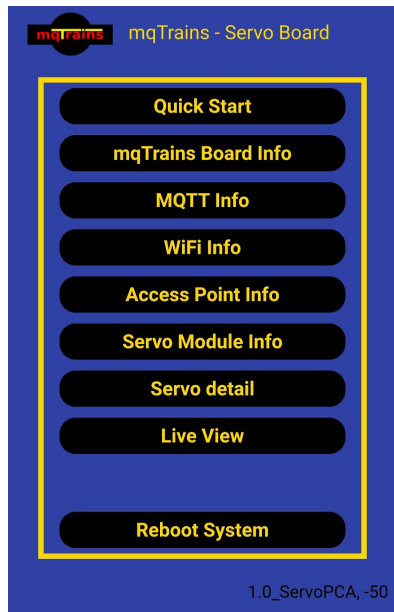
Connect a 3.3 Vdc power supply, as shown below to power the ESP01 and a 5 or 6 Vdc power supply to the PWM board to power the PCA board and the servo motors.



Connect a servo motor to one of the PWM sockets on the PWM board.

## 4. Configuring your ESP01

Using a smartphone or computer, look on your WiFi available networks list for the mqTrains Access Point (AP) network service started by the ESP01. The default name is mqTrains-0001.



Once connected to the AP, use a web browser to connect to the mqTrains configuration page using <http://2.2.2.1>.

Select the Quick Start option from the main menu and change each of these values to suit your needs.

Set a 4 digit Serial Number for the board. If you have more than one mqTrains board, each should have a unique serial number.

This will be used as part of the MQTT messaging. We recommend not leaving the default value, since the next one will also be 0001. Some use 1000s for turnout boards and 2000s for sensor

boards, others use two numbers for a location on the layout and two more for a serial number at that place, so why not change it to 1001?

Set the IP address of the computer on which your MQTT Server (Broker) is installed and the port number it uses. 1883 is usually the default port.

Set the base portion of the MQTT topic. The full topic for each servo will have the board serial number and the servo number appended to the base. For example, using the default base topic, the full MQTT topic for servo number /03 on board number 0002 will be myLayout/turnout/0002/03

Set the WiFi SSID and Key to connect to your desired WiFi service.

Press SAVE+REBOOT to save all these values to the ESP01's internal storage and restart it using those values.

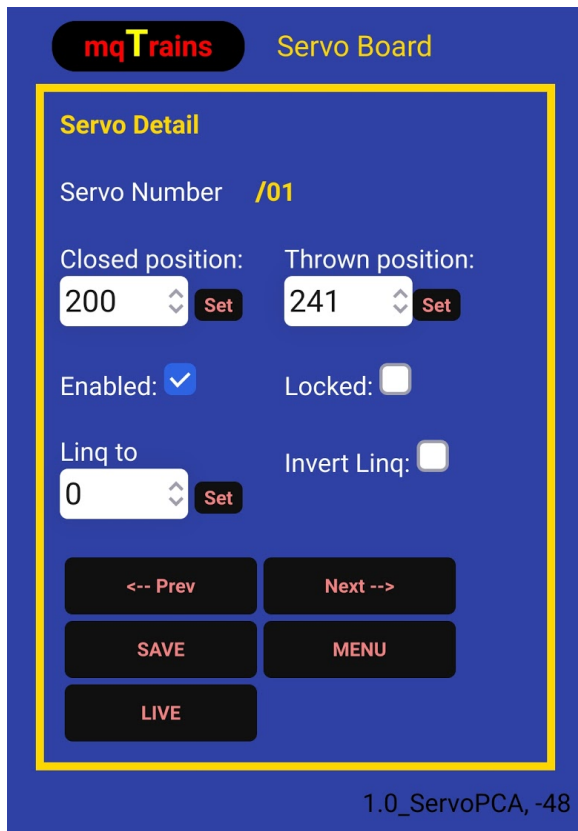
Using a utility such as MQTT Explorer or an MQTT subscribe line command, look for messages for topics beginning with the Base Topic. Expect to see a message like "192.168.1.10 is here!" This is the IP Address of the ESP01 on your WiFi network. Use this to connect to the configuration page to complete the set up or to make changes. The AP will no longer be needed and will shut down once the ESP01 is able to post a message to the MQTT Server.

## 5. Configure the servo positions

Servo numbers are assigned according to position on the PCA module into which the motor is plugged. For example, /03 is the third position, though it's labelled 2 on the module. The labels on the module start counting at zero.

The *Enabled* box must be ticked for servos being used. Ignore *Locked* and *Linq* settings for now, these are for special situations.

Servo motor settings for the closed and thrown positions are set using the *Servo Detail* page. Change the setting and press *Set*.



The values have to be between 115 and 500. Each time this is done, the limit setting is altered, the turnout is moved to the new closed or thrown position and an MQTT message is published.

Settings for each servo motor will be slightly different. Find the best settings by starting in the middle and adjust in small increments until you are happy. They can always be adjusted a little more if need be.

The inexpensive 9G Servo motors we commonly use rotate about half a revolution. We only need a small part of that range for moving turnouts. The settings used for the closed and thrown positions will vary for each servo motor depending on track scale, how the motor has been mounted, variations between servo motors and minor alignment variations during installation. Some motor orientations will result in Thrown position having a higher setting than the Closed position and some the reverse.

Plug in and configure one servo at a time to reduce the risk of damage by having inappropriate settings forcing motors beyond the operating range.

Attempting to force the point blade too far could damage the motor or dislodge your throw bar or track.

The method I use to set the Closed and Thrown positions for each servo is as follows:

Mount the servo motor into position without the horn attached. Connect the motor cable to the PWM board, power up the ESP01 and lastly, connect the 6 Vdc power to the PCA board.

Select the Servo Detail option on the main menu. Press "**Next**" to reach the servo you wish to configure.

Set the Closed and Thrown positions to about the middle of its rotation limits (307) Press the **Set** button for each. This will move the motor to a mid-range position.

Disconnect the servo's power source and the motor cable so that we don't move the motor position while the motor is live.

While trying not to change the servo's position, attach the horn to the motor and connect it to the turnout so that it puts the point blades approximately halfway between closed and thrown. Try to get this as accurate as you can so that when we power up again, our limits are within the desired Closed-Thrown range.

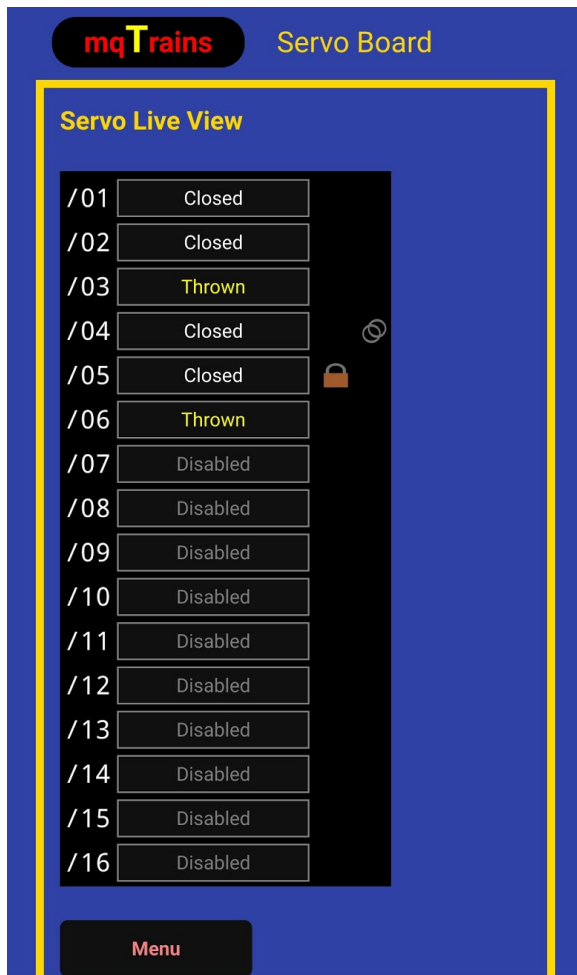
Slowly adjust the Closed limit by lowering the position setting in small increments away from the middle point. If it moves the wrong way, then increase instead. Remember to press "**Set**" with each move.

Continue in small increments until the point blade is properly aligned.

Repeat for the Thrown position, adjusting the opposite direction.

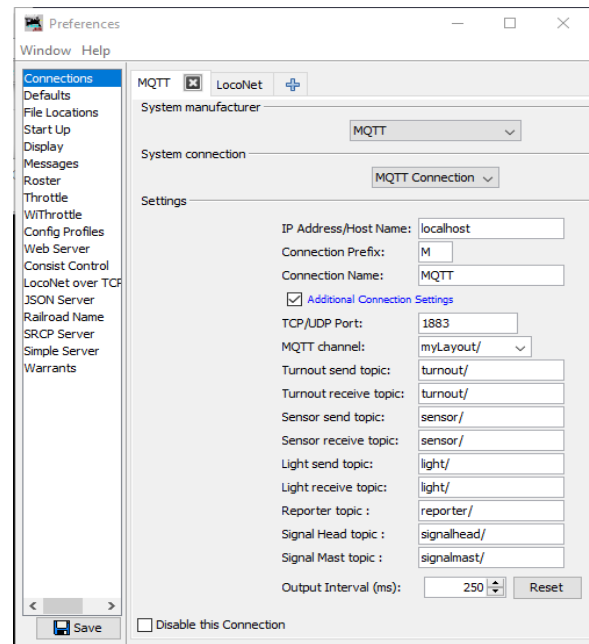
Once happy with the settings, press "**SAVE**".

Once servos are configured, they can be monitored using the Live View page. You can also toggle the turnout by tapping on the Closed/Thrown box. This will generate an MQTT message so JMRI can see that the turnout state has changed and update accordingly.



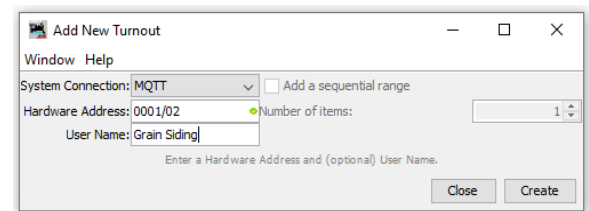
## Set JMRI to talk with your ESP01

In JMRI, using Edit -> Preferences -> Connections, create an MQTT connection with the settings as shown on the right, though make sure you specify the host on which you have installed the MQTT Server. You can specify the host name or the IP address as you did on the ESP01.



JMRI Connections (v 4.21.2)

Create MQTT turnouts using the board serial number and the position number for the hardware address, for example, the second pin on the PWM module for board number 0001 would have a hardware address of 0001/02 as shown below.



JMRI Add New Turnout

## Notes about RC Servos and the PWM values to control them

As described on Wikipedia [here](#) and [here](#), modern RC servos are not exactly PWM controlled devices. They don't really care about the 50 Hz rate of pulses coming in, they only care about the width of the pulses. A typical, inexpensive servo will move to its middle position when receiving a 1.5 ms (millisecond) pulse. To the far left with a 1 ms pulse and to the far right with a 2 ms pulse. Not all servos work exactly to pulses with these widths, but close enough for the mere 1 or 2 mm (millimeters) we need to the servo's horn to move the throwbar.

With a 12-bit control value used for each output on the PCA9685, and the PWM frequency set to

50 Hz (thus a 20 ms period), each increment in the PWM value will increase the pulse width by 4.8 us (microseconds). ( $0.020 / 2^{12} = 0.000,004,882$ )

So, to generate a 1 ms pulse, you need a PWM value of about 205, (from  $1/0.004,88$ ) and to generate a 2 ms pulse, twice as much, or 410. The middle, at 1.5 ms, would be 307 or maybe 308.

mqTrains limits the values from 115 to 500, which should be more than enough to make a servo travel in its full range. Try to install the

servo horn when the servo position is in the middle and the turnout points are in the middle and work with values starting close to 307.

When the mqTrains Servo board powers up, it will always send each of the servos enabled to the middle position calculated from the Closed and Thrown positions, so your points will loosen up as the throwbar moves to the middle of the travel. As soon as the MQTT subscription is successful, all the servos will receive their Closed or Thrown commands and move accordingly.

## Try mqTrains from the command line

The mqTrains web pages allow you to see and change turnout states and the setting limits for Closed and Thrown. On a command line, with the mosquitto tools installed, you can see the messages being published for those changes while mqTrains is connected to the MQTT Server:

```
prompt> mosquitto_sub -v -h localhost -t "myLayout/turnout/#"
```

```
"c:\Program Files\mosquitto\mosquitto_sub.exe" -v -h localhost -t "myLayout/turnout/#"
```

```
-v means verbose - show the full message (topic and payload).  
-t means topic  
-h means the MQTT Server hostname or IP address  
-r means retain the message in the Server.  
-m means message, aka payload.
```

You can also publish messages (CLOSED or THROWN) to the MQTT Server and see the turnout position change on the web page:

```
prompt> mosquitto_pub -h localhost -r -t "myLayout/turnout/0001/02" -m "THROWN"
```

You can also publish messages to alter the Closed and Thrown position values and save:

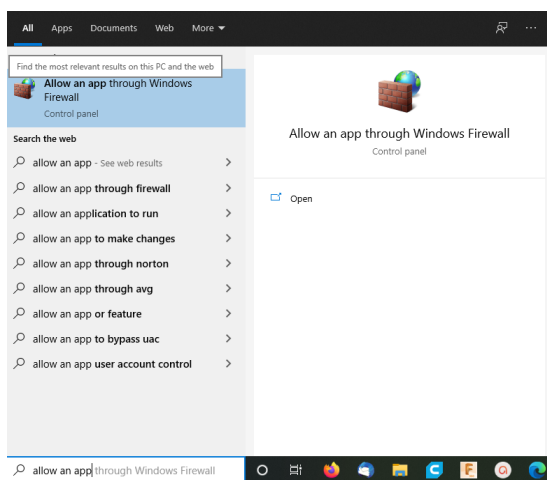
```
prompt> mosquitto_pub -h localhost -r -t "myLayout/turnout/0001/02" -m "T+5"  
prompt> mosquitto_pub -h localhost -r -t "myLayout/turnout/0001/02" -m "C-5"  
prompt> mosquitto_pub -h localhost -r -t "myLayout/turnout/0001/00" -m "SAV!"
```



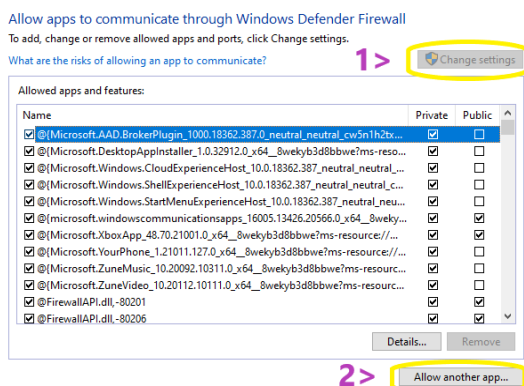
## Burning through the Windows Firewall

If you install MQTT Server Software (mosquitto) on a Windows computer, such as your JMRI machine, you may find that the ESP nodes cannot connect to the MQTT Server due to the Windows Defender Firewall blocking WiFi traffic coming into the computer. This section describes how to alter the Firewall to allow that MQTT messages to pass through.

Type "allow an app" into the Windows search bar. The "Allow an app through the Windows Firewall" will appear.

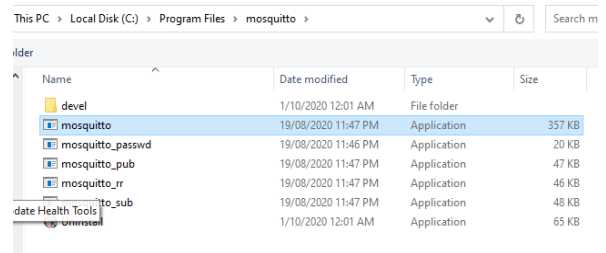


Click on this and then, on "Change Settings" and "Allow another app".

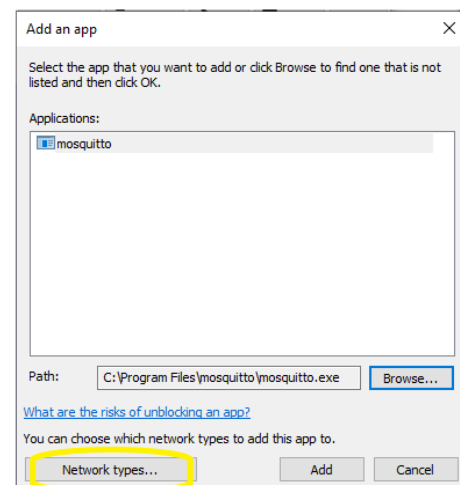


With Browse, select mosquitto.exe from C:\Program Files\mosquitto

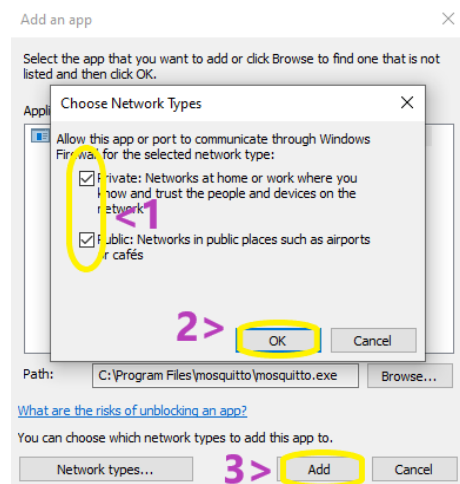
(32-bit machines would have X86 in that folder name)



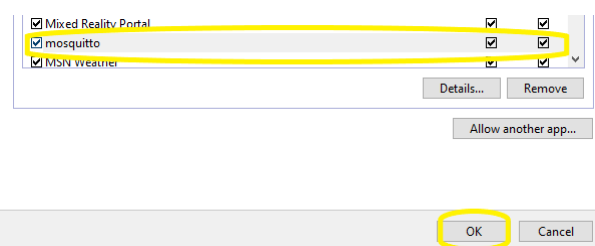
And click on "Open", then click on "Network types".



Depending on how your WiFi network is configured, you may require both Private and Public options so enable both anyway. Then click on "OK" and "Add".



You will then see mosquitto added to the list. Now press OK to finish.





---

mqTrains uses, without modifying any of their  
respective code as available on github:

ArduinoJSON by Bernoit Blanchon  
Arduino Queue by Einar Arnason  
Adafruit MCP23017 Arduino Library

Adafruit PWM Servo Driver Library  
Adafruit NeoPixel  
Adafruit SSD1306  
PubSubClient by Nick O'Leary  
Esp8266\_mdns by mrdunk  
LittleFS file system from [earlephilhowe](#)

ESPAsyncTCP and ESPAsyncWebServer from  
[me-no-dev](#)