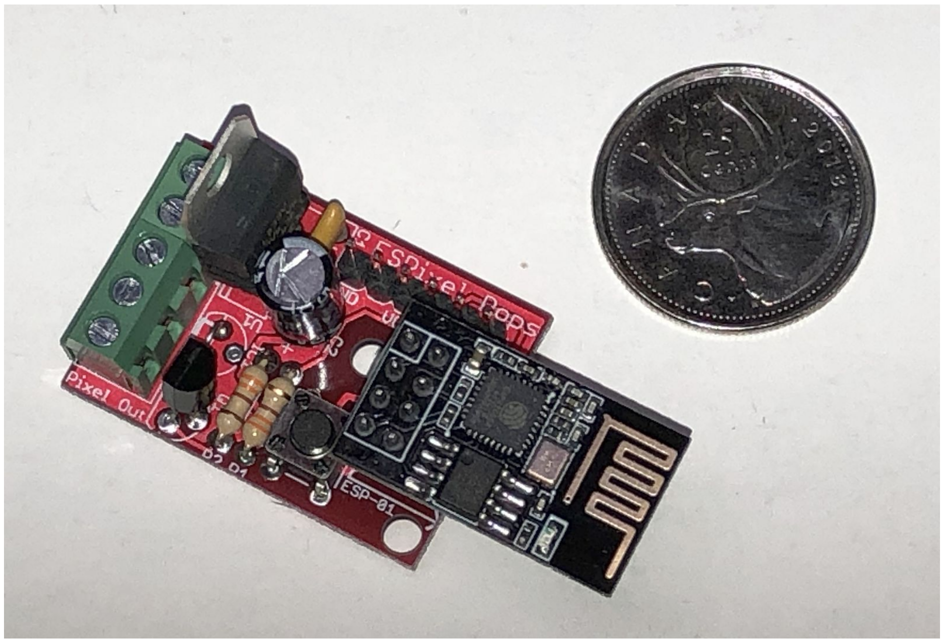
VHS ESPixelPOP Guide v3.20.2020



The ESPixelPOPs is a compact Wi-Fi based E1.31 (DMX over Ethernet) to Pixel (WS281x) controller that can handle up to 680 pixels (4 DMX Universes) of E1.31/sACN data. More information on ESPixel POPs can be found here:
https://www.doityourselfchristmas.com/wiki/index.php?title=ESPixel_Stick_%26_ESPixel_Pops
This controller uses the ESPixelStick software and you can find more information on it here:
https://github.com/forkineye/ESPixelStick

This guide was written to help people assemble a ESPixelPOPs kit that was offered as part of a workshop at my local maker space (**Vancouver Hack Space or VHS**). If you are not part of the VHS workshop this guide will still be helpful but some parts will not be relevant. This circuit board is not complicated but it is quite small (so previous soldering experience is handy) and the parts are not forgiving of being wired incorrectly (so some basic understanding of electronics is really handy). Please note that this build is designed for a 5V power supply and for 5V Pixels. It can be configured for 12V pixels but will require an additional part not included in the VHS kit.

You will need the following tools and equipment:

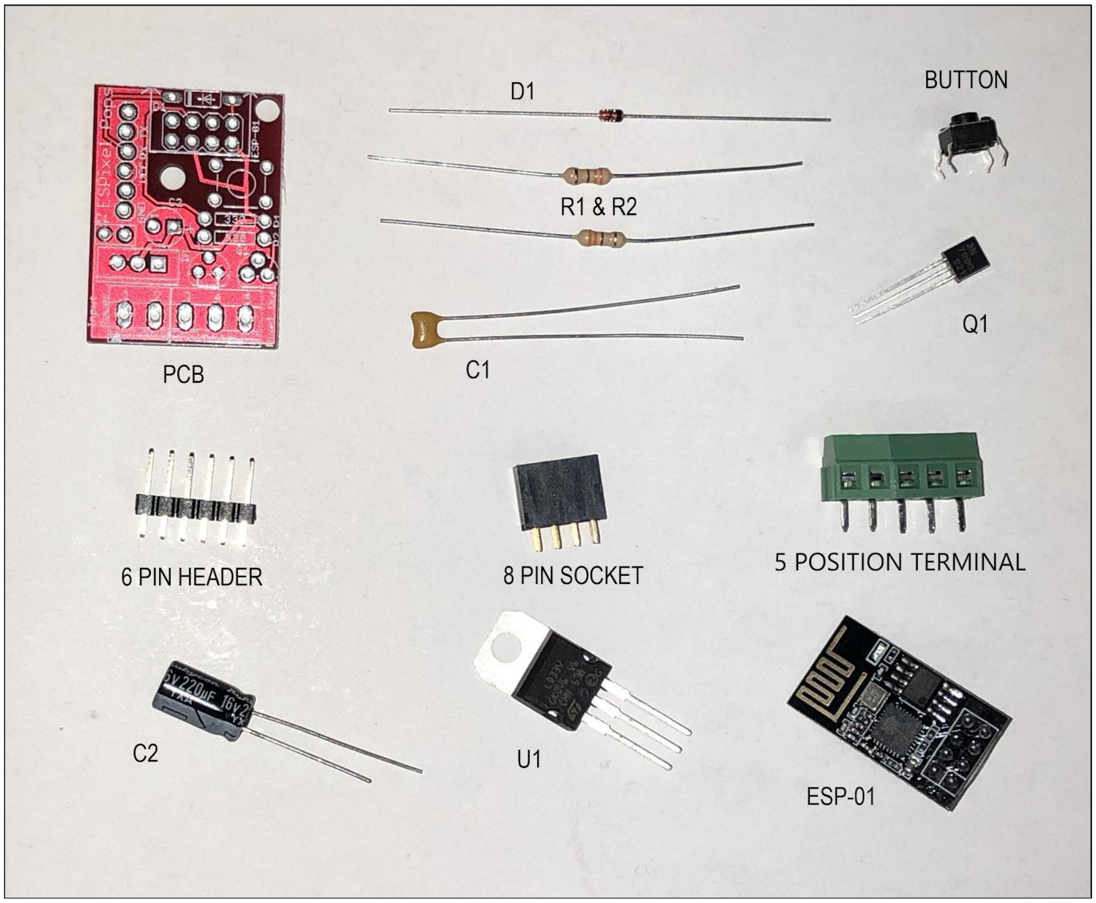
- Soldering Iron & Solder
- Diagonal Cutters
- Wire Strippers
- Electrical Tape
- Small Bladed Screwdriver
- 5 VDC Power Supply (Needs to source at least 2 Amps if you want to use full brightness on the included WS2812 pixel strip)

Step #1 - Referring to the picture below, check the list to ensure you have all the parts in your VHS kit. Q1 and the ESP-01 module will be in a protective antistatic bag, ensuring you take precautions with these parts as they are static sensitive.

- ❏ 1x ESPixel POPs PCB
- ❏ 1x 1N4148 Diode (D1)
- ❏ 2x 330 Ohm Resistor (R1 & R2)
- ❏ 1x 0.1 uF Ceramic Capacitor (C1)
- ❏ 1x PCB Push Button
- ❏ 1x 2N7000 MOSFET (Q1)
- ❏ 1x 6 Pin Male Header Strip
- ❏ 1x 8 Pin (2x4) Socket
- ❏ 1x 5 Position Terminal Strip
- ❏ 1x 220 uF Electrolytic Capacitor (C2)
- ❏ 1x LD1117V33 Voltage Regulator (U1)
- ❏ 1x ESP-01 Module (Preprogrammed with ESPixelStick firmware)

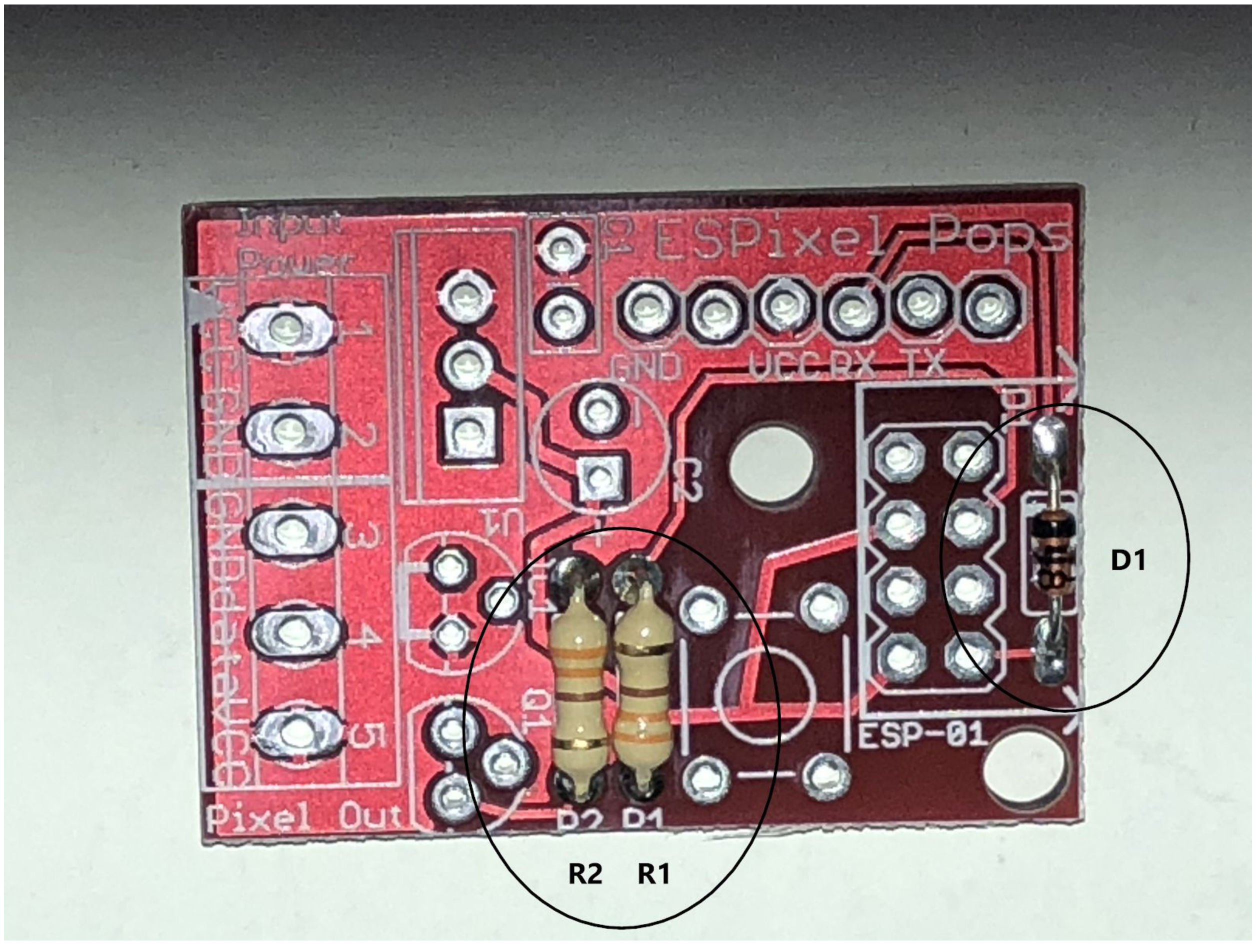
Also included in the kit but not shown in the picture are:

- ❏ 1x Pixel Strip Connector Cable
- ❏ 1x 30 LED WS2812 Pixel Strip



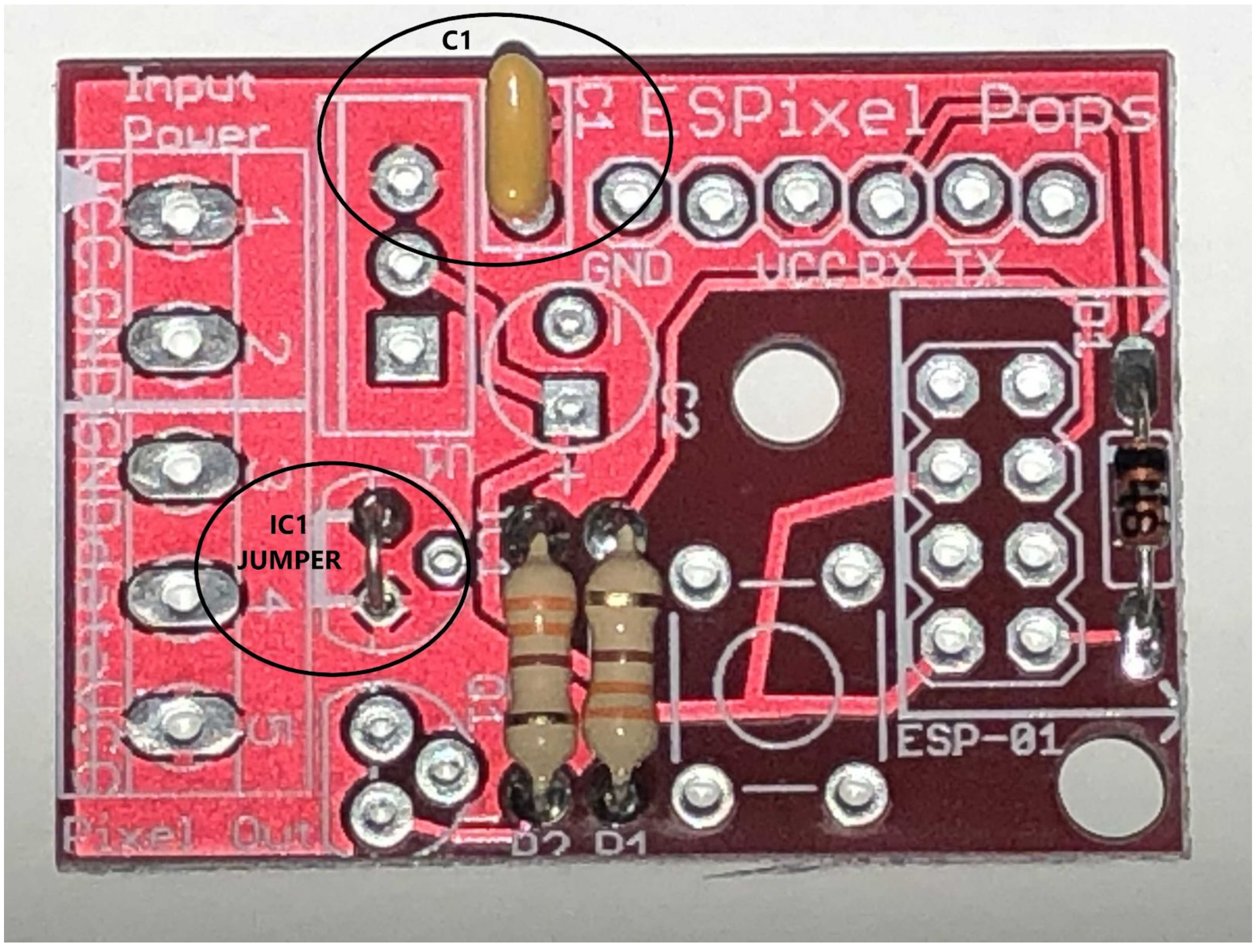
Step #2 - Following the steps below insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install D1 (ensure orientation matches symbol on PCB)
- ❏ Install R1 & R2 (orientation doesn't matter)



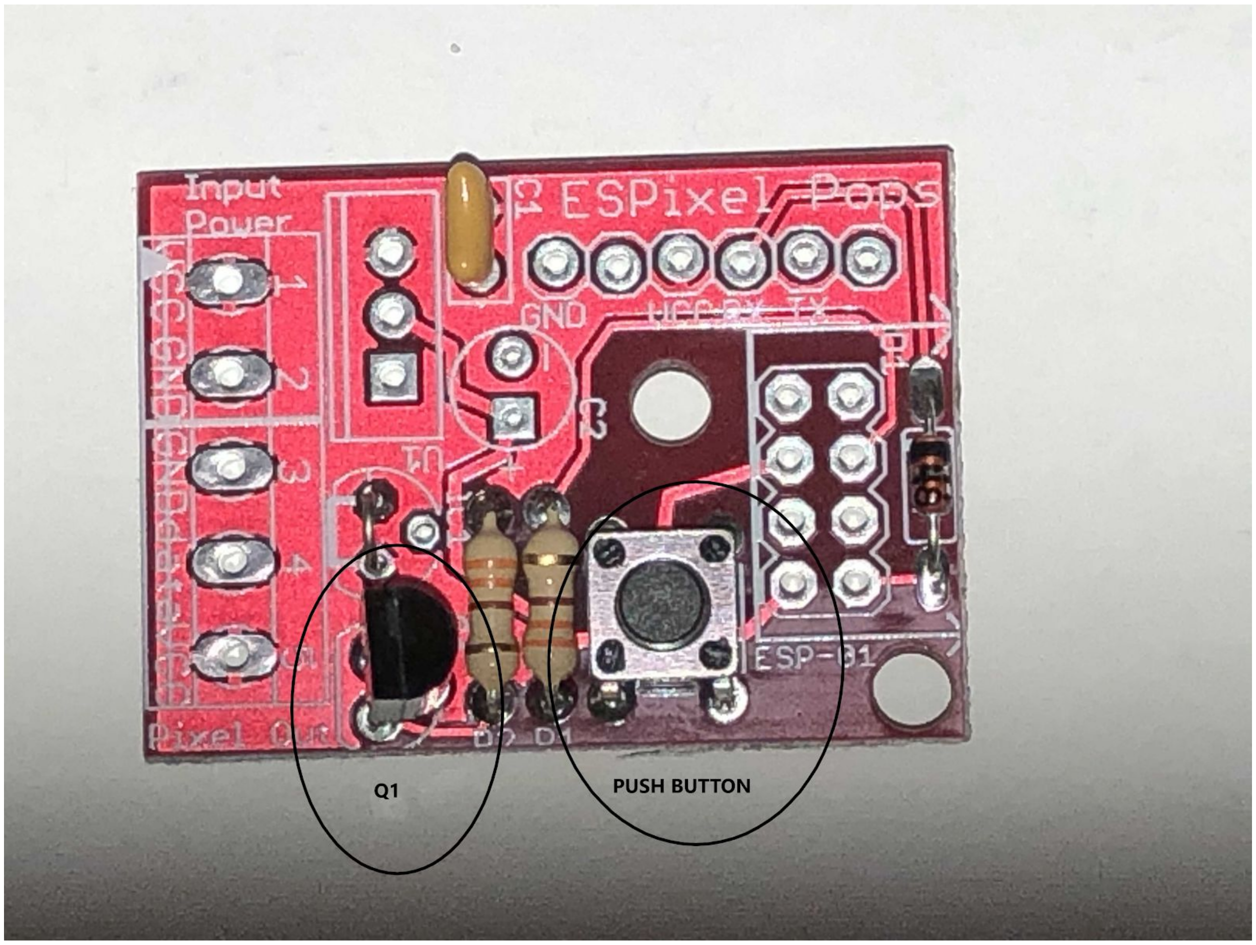
Step #3 - Following the steps below insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Jumper IC1 with cut off lead from D1 (jumper the holes as indicated on the PCB)
- ❏ Install C1 (orientation doesn't matter)



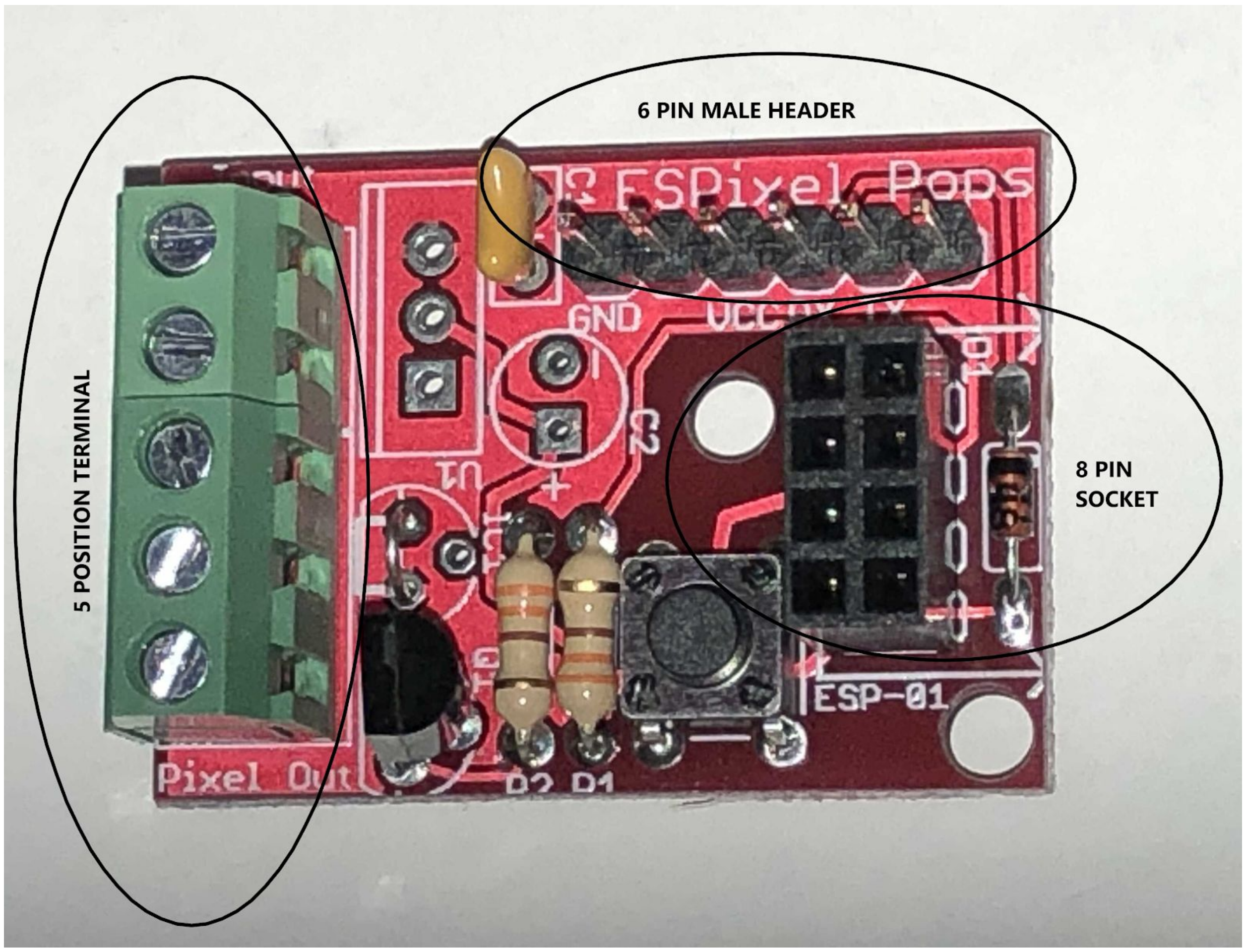
Step #4 - Following the steps below insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install push button
- ❏ Install Q1 (ensure orientation matches symbol on PCB)



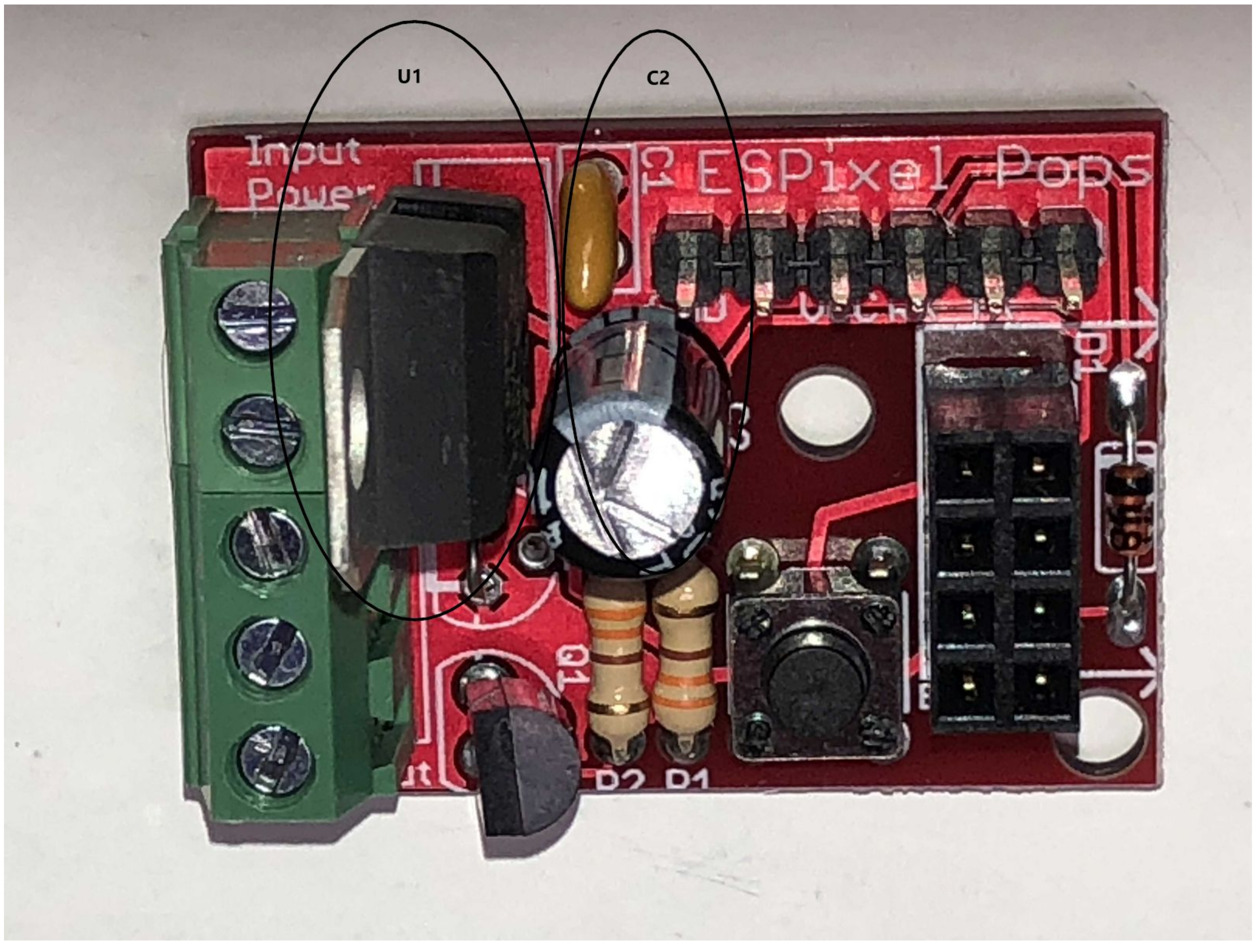
Step #5 - Following the steps below insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install 8 Pin Socket
- ❏ Install 6 Pin Male Header
- ❏ Install 5 Position Terminal (Ensure holes for wires face away from PCB)



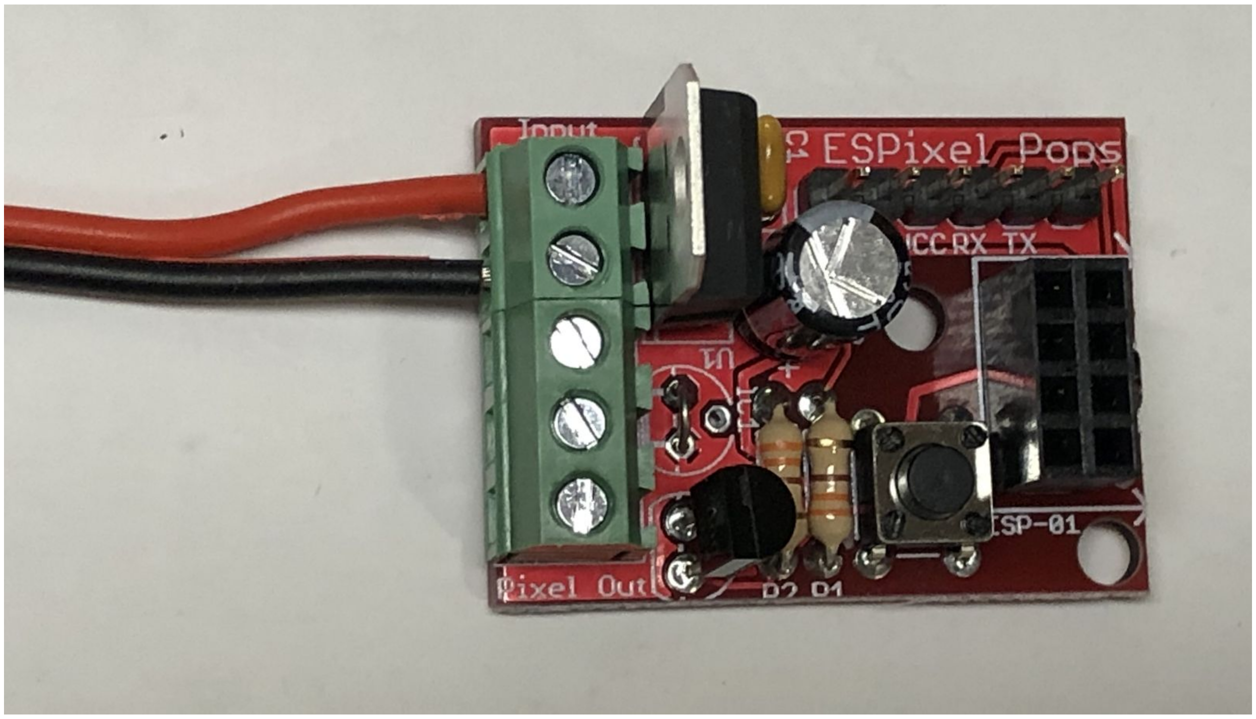
Step #6 - Following the steps below insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install C2 (Ensure long lead goes in square + hole)
- ❏ Install U1 (Ensure metal heatsink faces terminal strip)



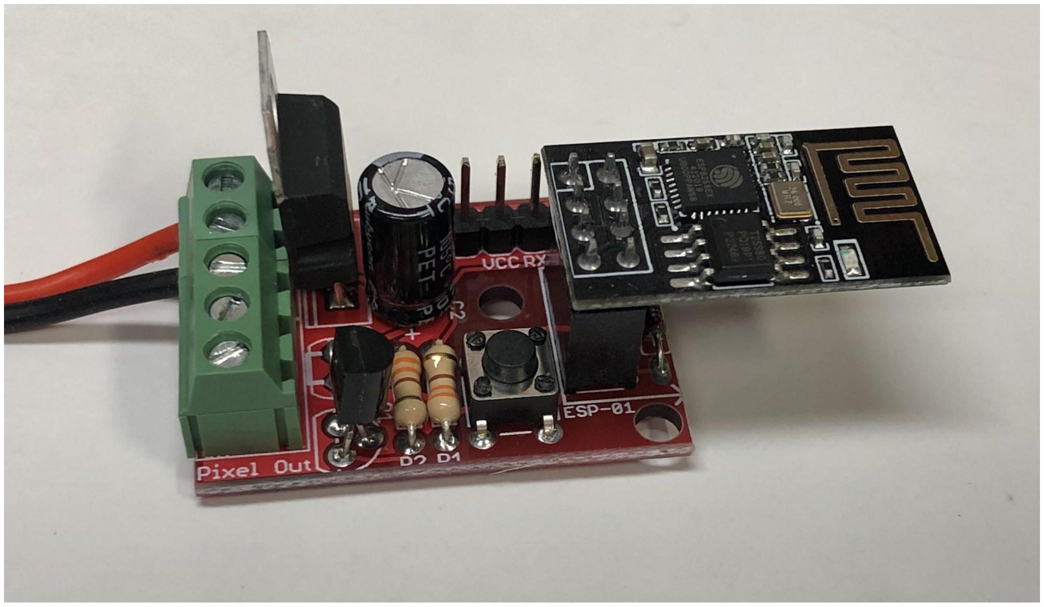
Step #7 - Power up and test your circuit

- ❑ Double check your parts placement and soldering (ensure there are no solder bridges).
- ❑ Connect +5V power supply to terminals 1 & 2 (VCC & GND)
- ❑ Double check to make sure the polarity is correct
- ❑ Turn on +5V power and ensure no magic smoke or other signs of problems
- ❑ Turn off +5V power

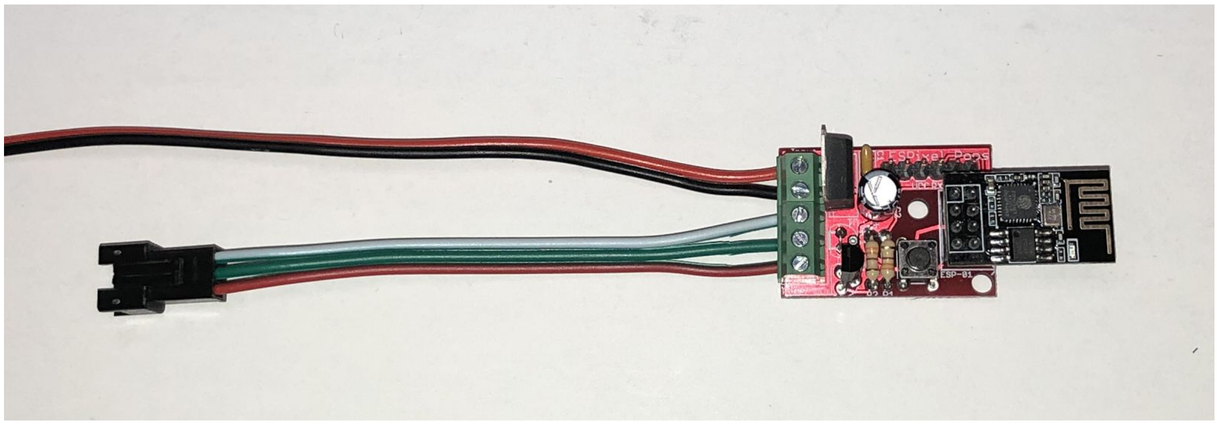


Step #8 - Install ESP-01 module and connect Pixel strip connector

- ❏ Install ESP-01 module in 8 pin socket as shown

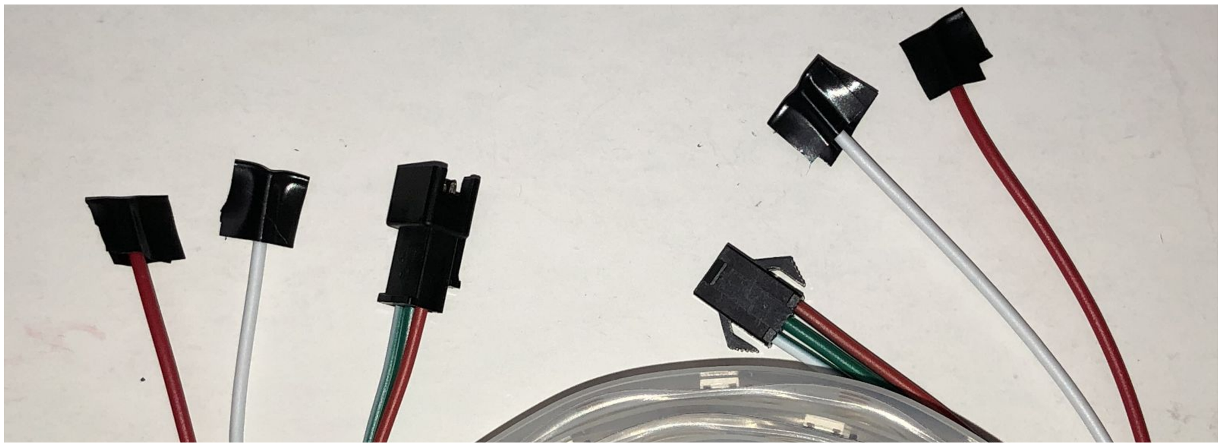


- ❏ Strip about 3/16” off the 3 wires that go to the Pixel Strip Connector. Twist the stranded end of each wire together and tin with solder to make it easier to insert into the PCB terminal connector. You may need to open the connector using your screwdriver before you can insert the wire and then tighten again. Connect Pixel Strip Connector wires to terminals 3, 4 & 5 as follows:
    - ❏ White Wire to GND (terminal 3)
    - ❏ Green Wire to Data (terminal 4)
    - ❏ Red Wire to VCC (terminal 5)

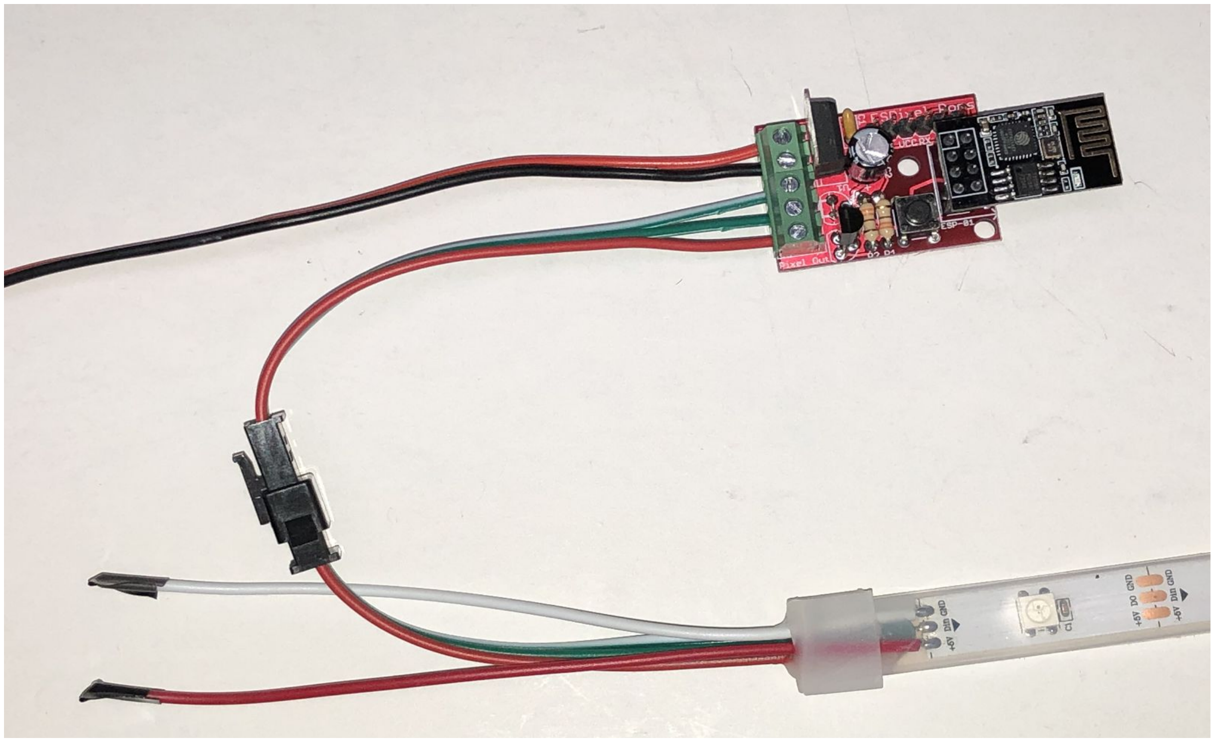


Step #9 - Connect WS2812 Pixel strip

❏ On each end of the Pixel Strip you will see a connector and two extra wires (one white and one red). These wires are for power injection and are not required for our project. Cover the end of each with some electrical tape to prevent any short circuits.



❏ Connect the input connector on the Pixel Strip (it will be on the end with the arrow facing into the strip) to the matching connector on the Pixel Strip Connector. It will only connect one way



Step #10 - Power up and connect to your network

- ❏ Turn on +5V power
- ❏ Use one of the methods below to connect your ESPixelPOP to your Wi-Fi network so you can start playing with pixels...

If you received this ESPixelPOP as a kit from a VHS workshop then the ESP-01 module has already been preprogrammed and you don't need to load the firmware. You can connect to it using these methods:

AP Setup Method:
Preprogrammed kits have been configured to connect to a Wi-Fi Access Point (AP) using these configurations:

SSID: PIXELNET
PW: pixelnet

If you set up your own Wi-Fi network using the above information your ESPixelPOP should be able to connect and get an IP address from your network. You will need to check your APs logs (It may show up as ESP-xxxx or PixelPOP" or scan your network to find the IP assigned to the pixel controller (or connect a USB-TTL adapter as noted in the methods below as the firmware sends out a status message showing the IP it gets). Once you determine the IP you should be able to log into the ESPixelPOP and configure it as required.

AP Fallback Method:
If your preprogrammed ESPixelPOP cannot connect to the Wi-Fi network (as noted above) then it will (after about 30 seconds) revert to AP mode. At this point the pixel controller will act as an AP and you should see an SSID called "PixelPOP", as a choice to connect to on your PC`s list of available Wi-Fi networks. Once you connect, (it will not need a password) your PC you should get an IP in the 192.168.4.x range. Now you can point your browser to 192.168.4.1 and get the configuration page of the ESPixelStick software. From here you can reconfigure the Wi-Fi to work on your own network. Reboot the ESPixelPOP and it should now connect to your network. You will need to check your APs logs or scan your network to find the IP assigned to the pixel controller (or connect a USB-TTL adapter as detailed in the methods below as the firmware sends out a status message showing the IP it gets).

If your ESPixelPOP kit did not have the firmware preprogrammed (or the above methods are not working) then you can use one of these methods below:

Some ESP-01 programming notes:

Do not use your USB-TTL adapter to power the ESPixelPOP during programming as it will not supply enough current to reliably power the ESP-01 module. Make sure you power the ESPixelPOP board from a separate 5V source that can supply at least 300 mA.

Only connect the USB -TTL adapter using the GND, RX & TX connections with jumper wires. If your USB-TTL adapter has sockets do not connect it directly to the pins on the ESPixelPOPs board. Do not connect any of the other pins from the USB-TTL adapter.

You can use either a 3.3V or a 5V version of a USB-TTL adapter. The diode in the ESPixelPOPs circuit acts as a level translator so the ESP-01 only sees the correct voltage either way.

I have found some cheap USB-TTL adapters do not seem to work as well as others for programming the ESP-01 module on the ESPixelPOP board. If you are having issues I recommend you try a different adapter. If you still have issues you may want to make or purchase a dedicated ESP-01 programmer.

Precompiled Firmware Uploader Method:
For this option you will need a USB to TTL adapter (ensure it is a 3.3V version not 5V and that you have the required drivers installed).

- ❏ Go to the ESPixelStick GitHub site (https://github.com/forkineye/ESPixelStick/releases) and download the ESPixelStick_Firmware-3.1.zip file. Uncompress it on your PC.
- ❏ Connect your USB-TTL adapter to your PC and ensure it is recognized by your PC.
- ❏ Connect your USB-TTL adapter pins to the header pins on the ESPixelPOP as noted:
  - ❏ USB Adapter GND to PixelPOP GND
  - ❏ USB Adapter TXD to PixelPOP RX
  - ❏ USB Adapter RXD to PixelPOP TX
- ❏ Open up the ESPixelStick firmware folder and launch the ESPFlashTool java applet
- ❏ Ensure the Serial Port dropdown box has the correct port selected for your USB-TTL adapter
- ❏ Connect the PixelPOP PCB to a +5V source
- ❏ You should see data displayed in the “Serial Output” window of the flash application as the ESP-01 boots up
- ❏ Fill in the SSID and Paraphrase boxes as appropriate for your Wi-Fi network
- ❏ Ensure the Firmware dropdown box has “Pixel(Ws2811/GECE) v3.1” selected
- ❏ Power cycle the ESPixelPOP but this time hold down the push button on the PixelPOP PCB and release after a few seconds
- ❏ Press the “Upload” button on the ESPFlashTool application
- ❏ You should see the file transfers progressing in the “Status” window of the applet
- ❏ If all goes well the unit will reboot after the firmware update and use the new SSID and credentials to connect to your network. You can watch in the “Serial Output” window for messages and see what IP it gets via DHCP from your network.

Arduino IDE Upload Method:
For this option you will need a USB to TTL adapter (ensure it is a 3.3V version not 5V and that you have the required drivers installed).

1) Go to the ESPixelStick GitHub site (https://github.com/forkineye/ESPixelStick/releases) and grab the Source code.zip file. Uncompress it on your PC.
2) Connect the USB TTL adapter to the header pins on the ESPIxelPOP as noted:
   a) USB Adapter GND to PixelPOP GND
   b) USB Adapter TXD to PixelPOP RX
   c) USB Adapter RXD to PixelPOP RX
2) Connect the PixelPOP PCB to a +5V source
3) Open the source code folder and open the “ESPixelStick.ino” file with your Arduino IDE
4) At the top of the page of code you will see a section for you Wi-Fi credentials.
5) Compile and upload the code to the ESP module
6) Reboot the ESPixelPop and if you watch the Serial Monitor window you will be able to see the connection details and see what IP it gets via DHCP from your network.

Step #11 - Configure and test your ESPIxelPOPs

Once you have the ESPixelPop configured and connected to your network you can point a browser to it's IP and bring up the status and configuration page.

You should set the IP of the ESPixelPOP to a static address as this will make using it much easier. You can do this from the Network Configuration tab.

For the WS2812 Pixel strips supplied in the VHS workshop kit you will need to change the Pixel Color Order from the default RGB to GRB. Once you have done this go to the Effects tab and select "Solid Color" and using the color picker choose each of the primary colors (Red, Green & Blue) in turn and ensure the LEDs turn the correct color.

You can use the Effects tab to run various test sequences to the attached pixel string you have with your kit. This will ensure that your pixels are connected correctly, that they are wired and powered correctly.

Next you can send E1.31 data to it and the attached WS2812 strip should respond. With the default settings of the ESPixelPOP you can send your data to the controllers IP address and the pixels should respond at Universe 1 Channels 1 to 90 (each of the strips 30 pixels uses 3 channels).

For further testing I recommend da_Tester (https://www.da-share.com/software/da_tester/) or sACNView (https://sacnview.org/). Both of these applications can send test sequences or give you slider control over all your channels.

To generate sACN data for your controller I recommend using Jinx (http://www.live-leds.de/) for DJ and club style lighting effects or xLights (http://xlights.org/) for christmas lighting animation.

Below is a quick overview of the various ESPixelStick configuration settings:

Home Tab:
Network Status:
SSID - Wi-Fi Network you are connecting to
Hostname - Hostname you have configured (or default based on MAC address)
IP - IP Address (either from DHCP or manually configured)
MAC - MAC address of ESP-01 module
RSSI - Wi-Fi signal strength
Free Heap - Free memory on ESP-01 module
Up Time - Length of time controller has been running
W1.31 Status:
Universe Range - Number of universes the Pixel Count covers
Total Packets - Number of E1.31 packets received
Sequence Errors - Number of out of order E1.31 packets received
Packet Errors - Number of errored E1.31 packets received
Source IP - IP address of node sending packets to controller

Wireless Setup Tab:
Network Configuration:
SSID - WiFi network SSID
Password - Wi-Fi Password
Hostname - Hostname for your controller (used when obtaining DHCP address)
Client Timeout - Duration controller tries to connect to configured Wi-Fi before reverting to AP mode (if AP Fallback enabled)
Use DHCP - Unselect this to manually configure the IP address.
AP Fallback - Select to allow for the device to fallback to AP mode if it can't connect to the configured network.

Device Setup Tab:
Device Configuration:
Device ID - Name for the controller
Universe - DMX universe the Pixels are starting at
Start Channel - DMX channel the Pixels are starting at
Universe Boundary - The last channel used in the DMX universe. While a DMX universe does have 512 channels, only 510 can be used for a number of pixels (3 x 170 = 510). Some systems use 510 for each universe while others use all 512. (needs to match what your are sending)
Enable Multicast - Checked if you want to use Multicast.
Pixel Configuration:
Pixel Count - Number of Pixels attached to controller
Pixel Type - Select WS2811 or GECE (You will most likely have WS2811)
ZigZag Count - Adjust for using zigzag in panels (rather than line runs)
Gamma Value - Lets you setup a custom dimming curve (LEDs don't dim linearly)
Refresh Rate - Shows the maximum refresh rate you can get based on the Pixel settings you select. Data sent faster than this will get missed.
Group Size - Sets pixels into groups (affects all pixels connected)
Color Order - Sets the color order as some pixels are not RGB (affects all pixels connected)
Brightness - Sets the maximum brightness for the string. Lets you limit the power used by the string and prolongs LED life.
MQTT Configuration:
Enable MQTT - Configuration options for using MQTT

Effects Tab:
Effect Options:
Effect - Select the desired effect and color you want to use
Reverse pattern - Reverse pattern direction
Mirror pattern - Mirror image pattern from middle of string
All leds same - Make all pixels identical
Effects Speed - Control speed of effect
Effects Brightness - Control brightness of effect
Effect Runtime:
Enable at startup - Runs selected effect when device starts (ignores and E1.31 data)
Enable when idle - Runs selected effect when no data being sent
Idle Timeout - Delay after no data before effect starts

Diagnostics Tab:
View Stream:
Display - RGB shows pixel colour (1 square per pixel), Channel shows LED brightness (1 square per LED)
Columns - How many squares in a row (display fills to match pixel count)

Admin Tab:
Administration:
FW Version - Firmware version
Build Date - Firmware date
Used Flash Size - Firmware size used by firmware
Real Flash Size - Actual flash size on ESP-01 (some modules have smaller flash than advertised)
Flash Chip ID - ID value from flash chip
Update Firmware - Lets you update firmware over the Wi-Fi connection. You need the EFU file from the Github site or you can create one from the FlashUpdater application.
Reboot - Reboot the controller

The above information should allow you to build, test and configure your ESPixelPOP and do some basic testing with the included pixel string. If you have any questions feel free to contact me on the VHS Talk site.