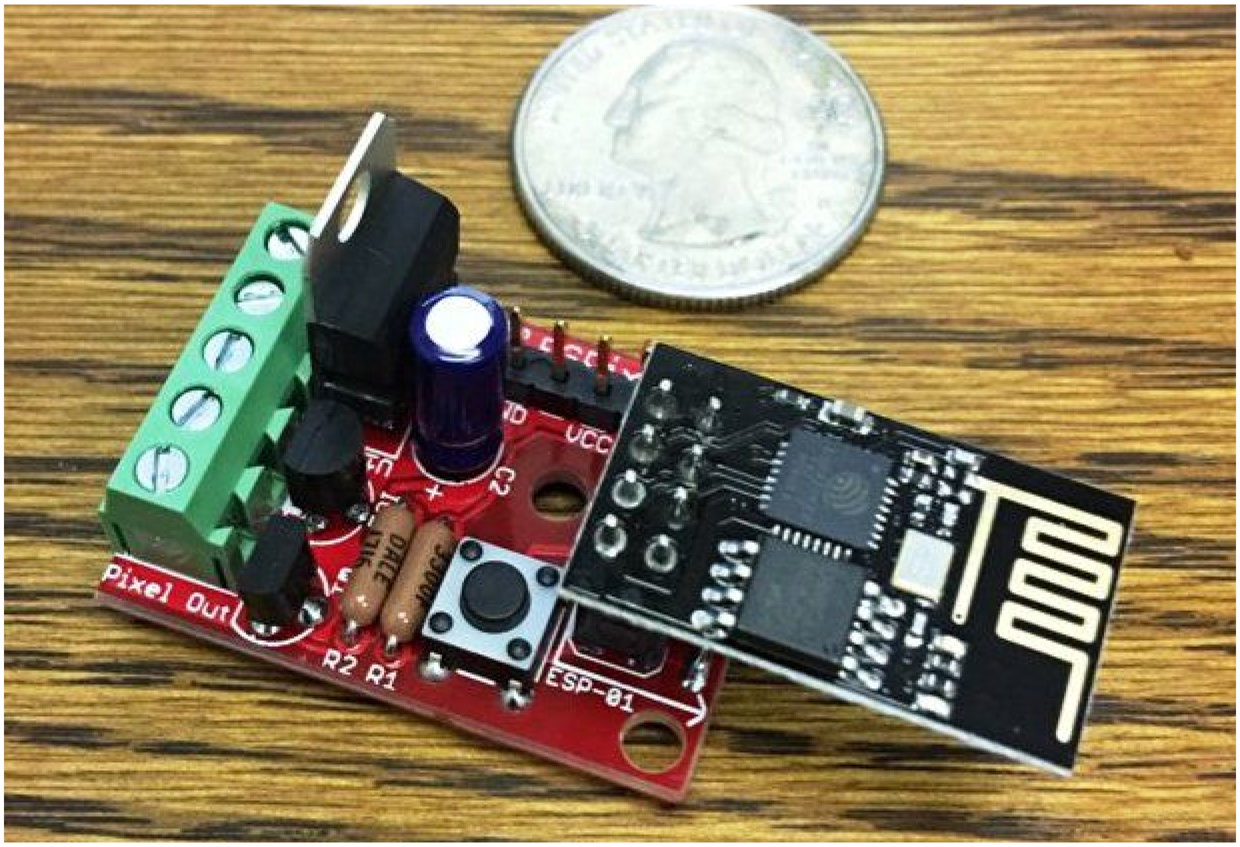
ESPixelPOPs Build & Setup Guide v3.20.2020



The ESPixelPOPs board is a small Wi-Fi based E1.31 (DMX over Ethernet) to Pixel (WS281x) controller that can handle up to 680 pixels of E1.31/sACN (4 DMX Universes) 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

I have put together this guide based on my experiences with multiple ESPixelPOP builds and while I expect it is all correct, there is no guarantee. If you find any errors or something that should be added please let me know and I will revise it as required.

packetbob@gmail.com

The ESPixelPOPs board has only a few through hole parts and is fairly easy to build. It is helpful if you have had previous soldering experience as it is a small PCB. A bright light and perhaps magnifying reading glasses may be helpful.

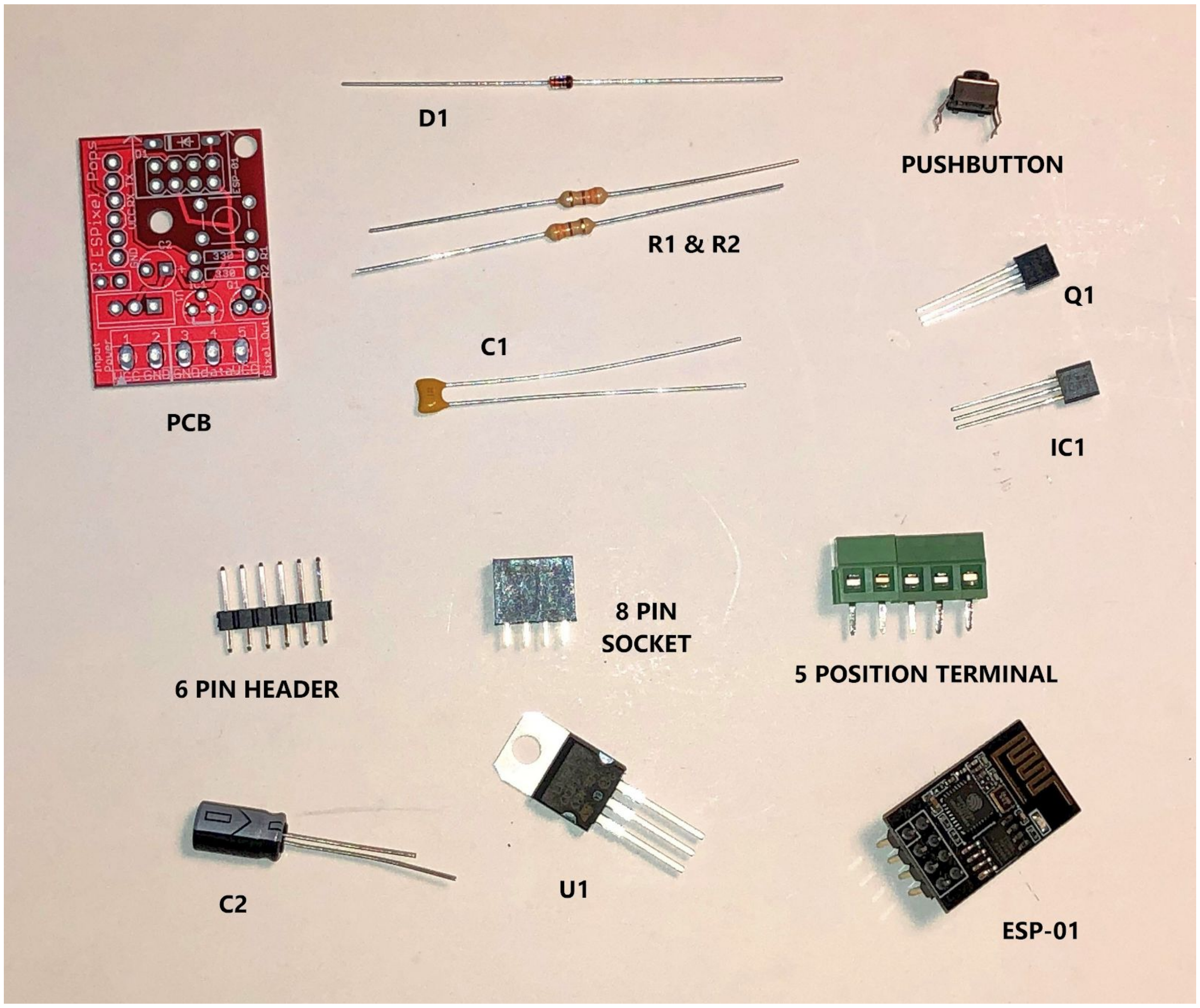
You will need the following tools and equipment:

- Soldering Iron & Solder
- Diagonal Cutters
- USB to Serial TTL Adapter (3.3V or 5V Version)
- Small bladed screwdriver
- Wire Strippers
- WS2811 Pixels

5V/12V Notes:
As per the instructions below you can build the ESPixelPOPs for either 5V or 12V operation. This is solely dependent on the requirements of your WS2811 pixel string. If you have 5V pixels you need a 5V power source and build the board for 5V operation. If you have 12V pixels you need a 12V power source and build the board for 12V operation. The only difference is the requirement for IC1 (5V Regulator) for 12V operation or using a jumper in it/s place for 5V operation. Note that a ESPixelPOP built for 12V will not work for 5V pixels and vice versa. You risk damaging your pixels if you try..

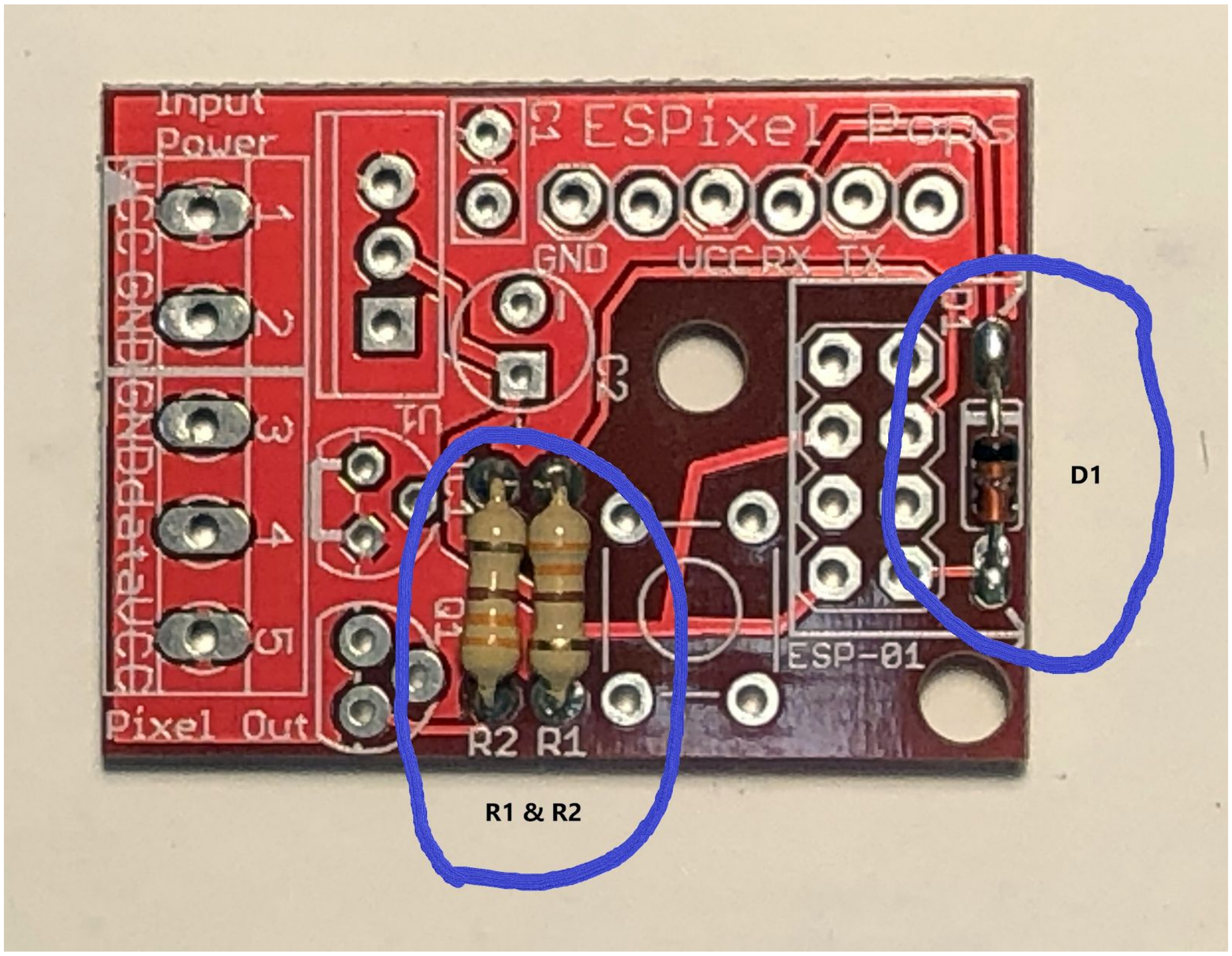
Step #1 - Referring to the picture below, make sure you have all the parts required:

- ❏ 1x ESPixelPOPs PCB
- ❏ 1x 1N4148 Diode (D1)
- ❏ 2x 330 1/4W Ohm Resistor (R1 & R2)
- ❏ 1x 0.1 uF Ceramic Capacitor (C1)
- ❏ 1x 6mm PCB Push Button
- ❏ 1x 2N7000 MOSFET (Q1)
- ❏ 1x 6 Pin 2.54mm Male Header Strip
- ❏ 1x 8 Pin (2x4) 2.54mm Socket
- ❏ 1x 5 Position 3.81mm SCrew Terminal Strip
- ❏ 1x 220 uF 16V Electrolytic Capacitor (C2)
- ❏ 1x LD1117V33 Voltage Regulator (U1)
- ❏ 1x ESP-01 ESP8266 Module
- ❏ 1x LM78L05 Voltage Regulator (IC1) - Only needed for 12V version



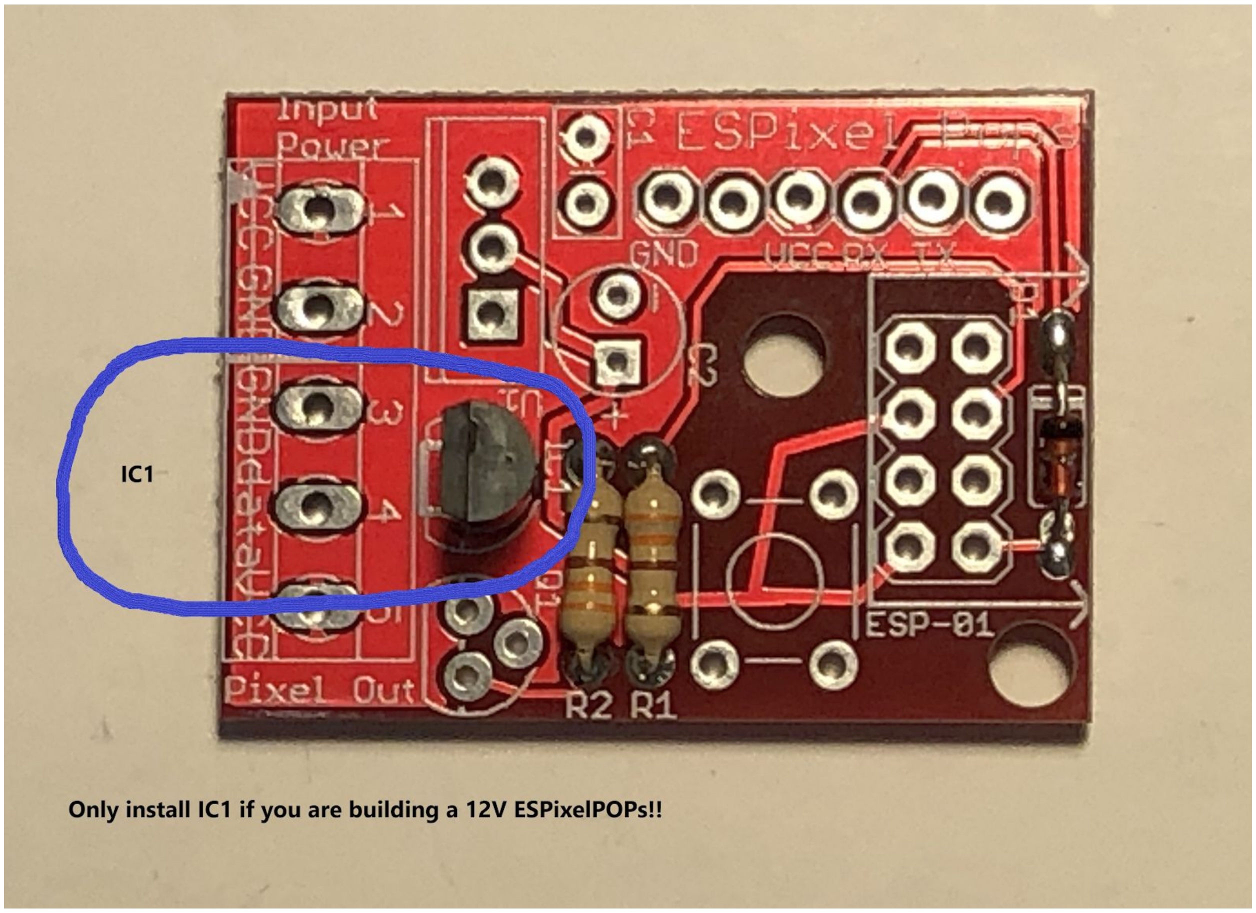
Step #2 - Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

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



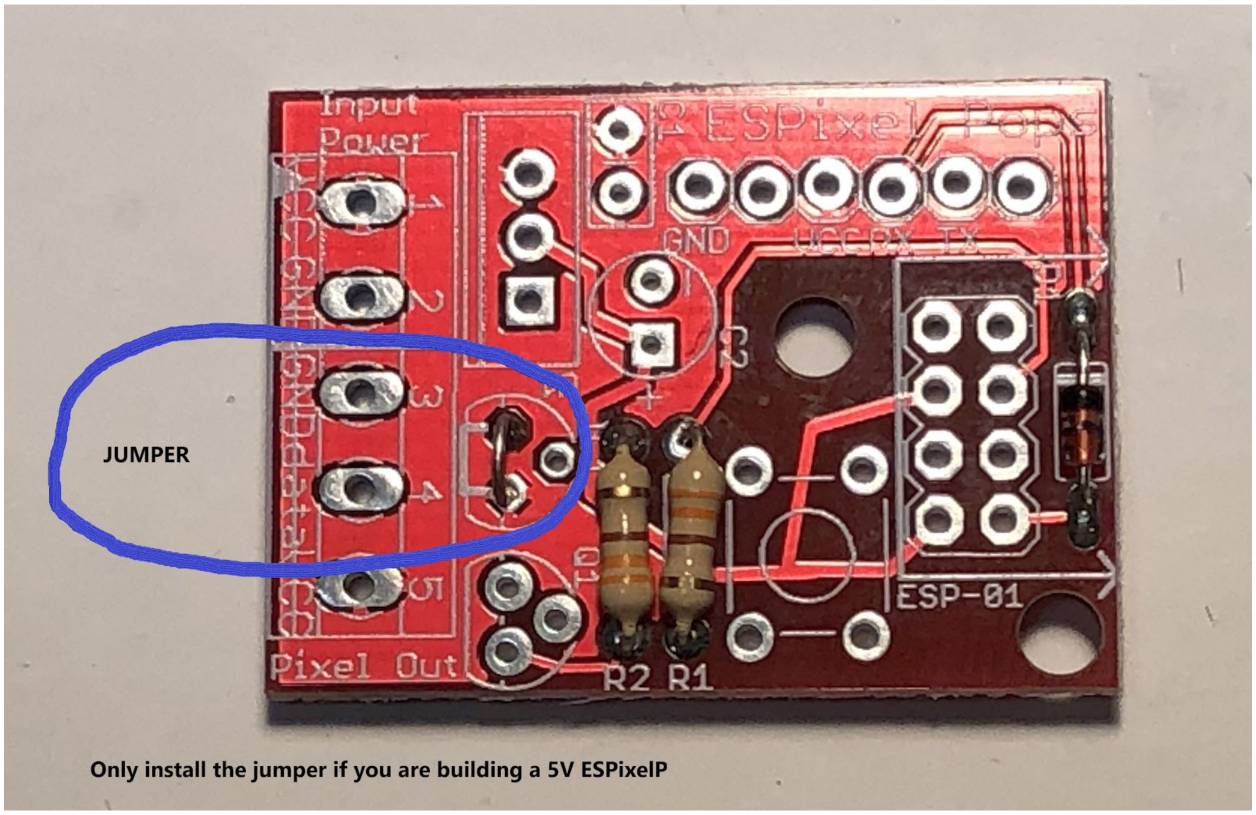
Step #3 - Only install this part if you are building a **12V** version

- ❏ If you are building a **12V** version then install IC1 (ensure orientation matches symbol on PCB)
- ❏ Skip to **Step #5** after you have done this



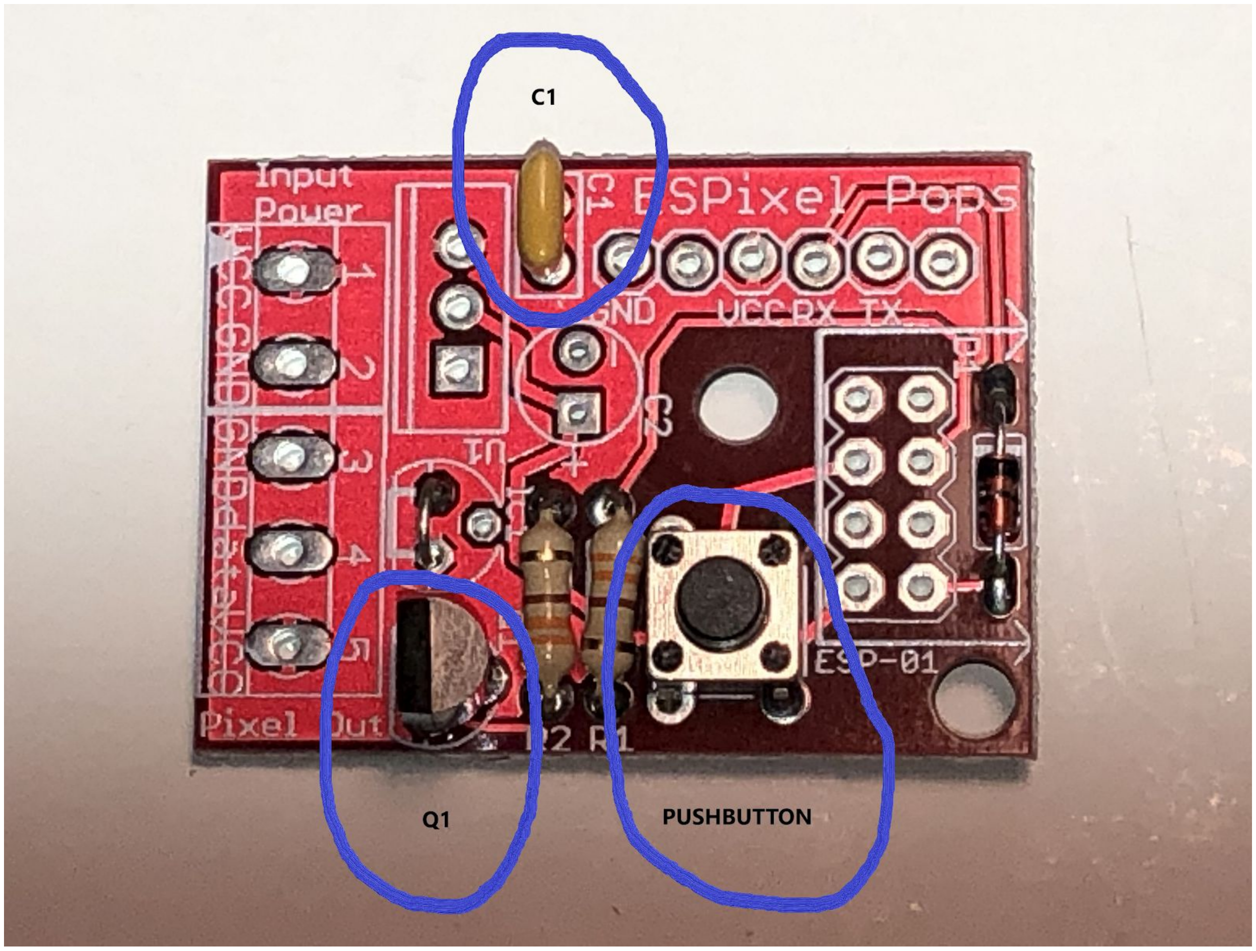
Step #4 - Only install this part if you are building a **5V** version

- ❏ If you are building a **5V** version then Jumper IC1 with cut off lead from D1 (jumper the holes as indicated on the PCB).



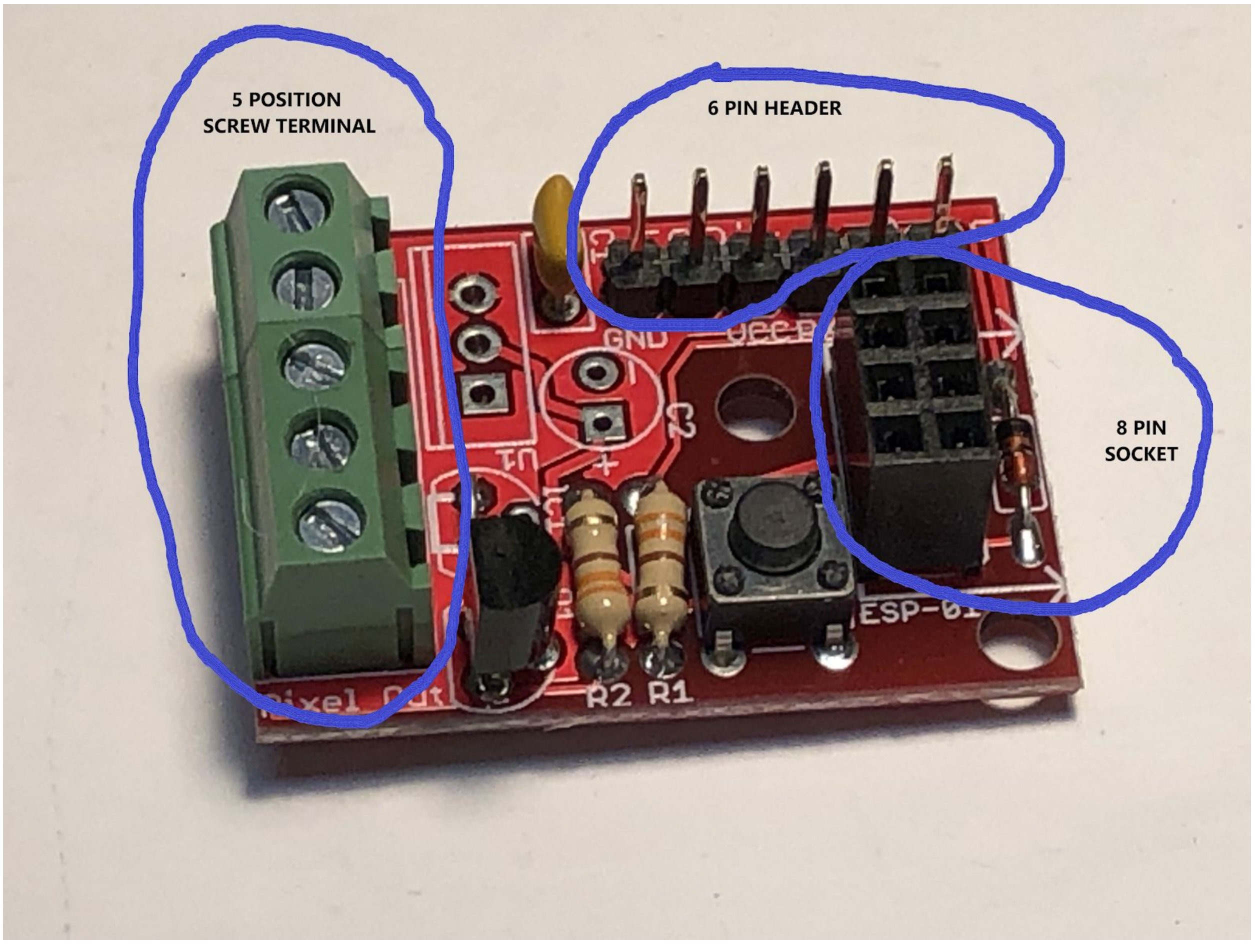
Step #5 - Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install C1 (orientation doesn't matter)
- ❏ Install push button (will only fit in one direction)
- ❏ Install Q1 (ensure orientation matches symbol on PCB)



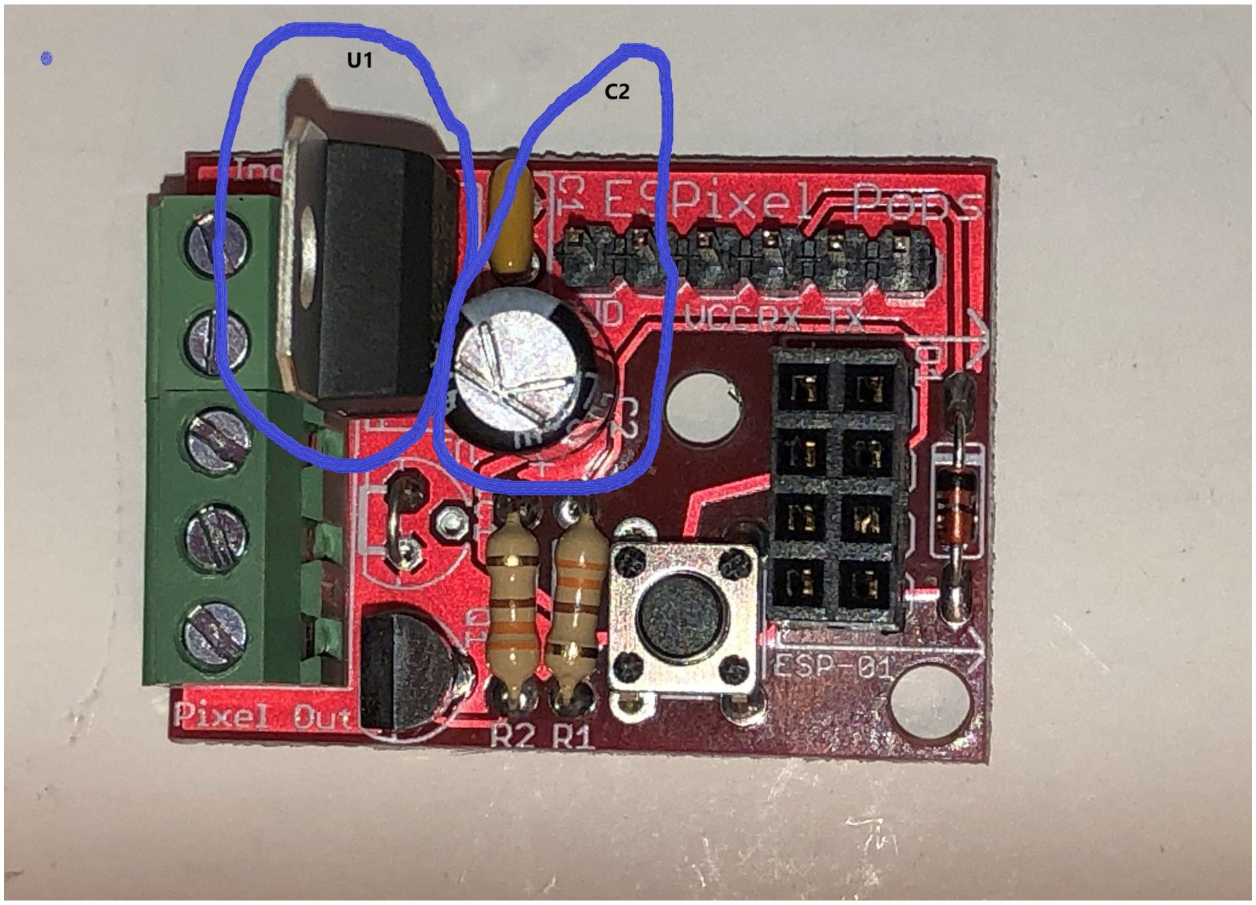
Step #6 - Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

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



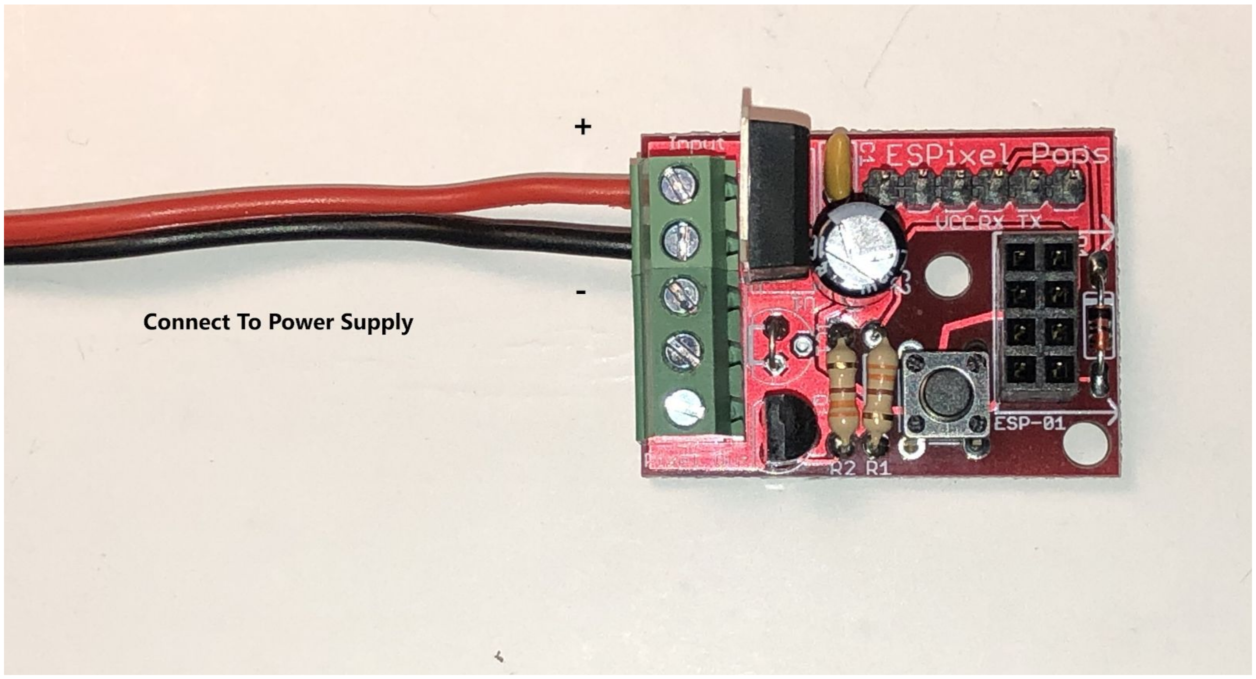
Step #7 - Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

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



Step #8 - Power up your and smoke test your ESPixelPOP

- ❏ Double check your parts placement and soldering (ensure there are no solder bridges).
- ❏ Connect +5V power supply connections to terminals 1 & 2 (VCC & GND)
- ❏ Double check connections to make sure the polarity is correct
- ❏ Turn on power and ensure all is well (if no smoke then all is good)
- ❏ Turn off power to ESPixelPOP

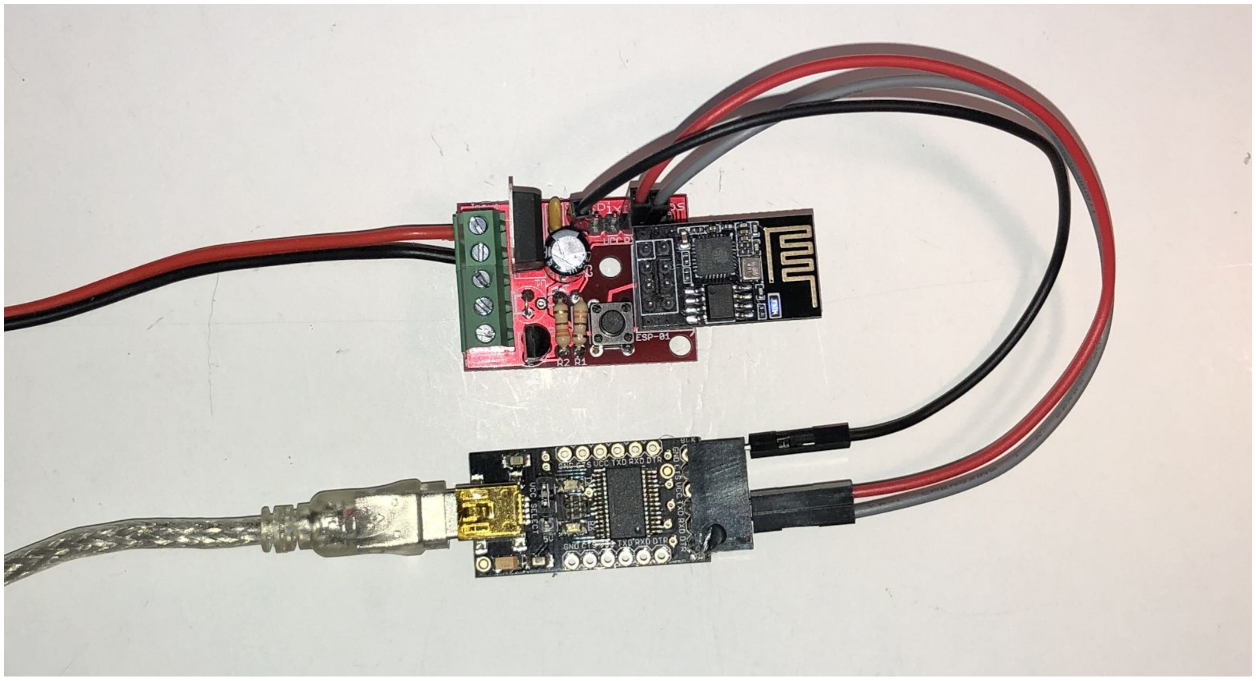


USB-TTL Adapter Programming Notes:
Do not use your USB-TTL adapter to power the ESPixelPOP during programming as it cannot supply enough current to reliably power the ESP-01 module. Make sure you power the ESPixelPOP board from a separate source that can supply at least 300 mA.

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.



Step #9 - Install ESP-01 module and upload the firmware

- ❏ Install ESP-01 module in 8 pin socket (ensure antenna side of module is away from the PCB)
- ❏ 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

Use one of the methods below to load the firmware. Using the precompiled firmware is by far the easiest route to follow. I would avoid using the IDE and compiling the code unless you know what you are doing.

Precompiled Firmware Uploader Method:

- ❏ 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.
- ❏ 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 power 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 ESPixelPOP 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 make a note of what IP it gets via DHCP from your network.

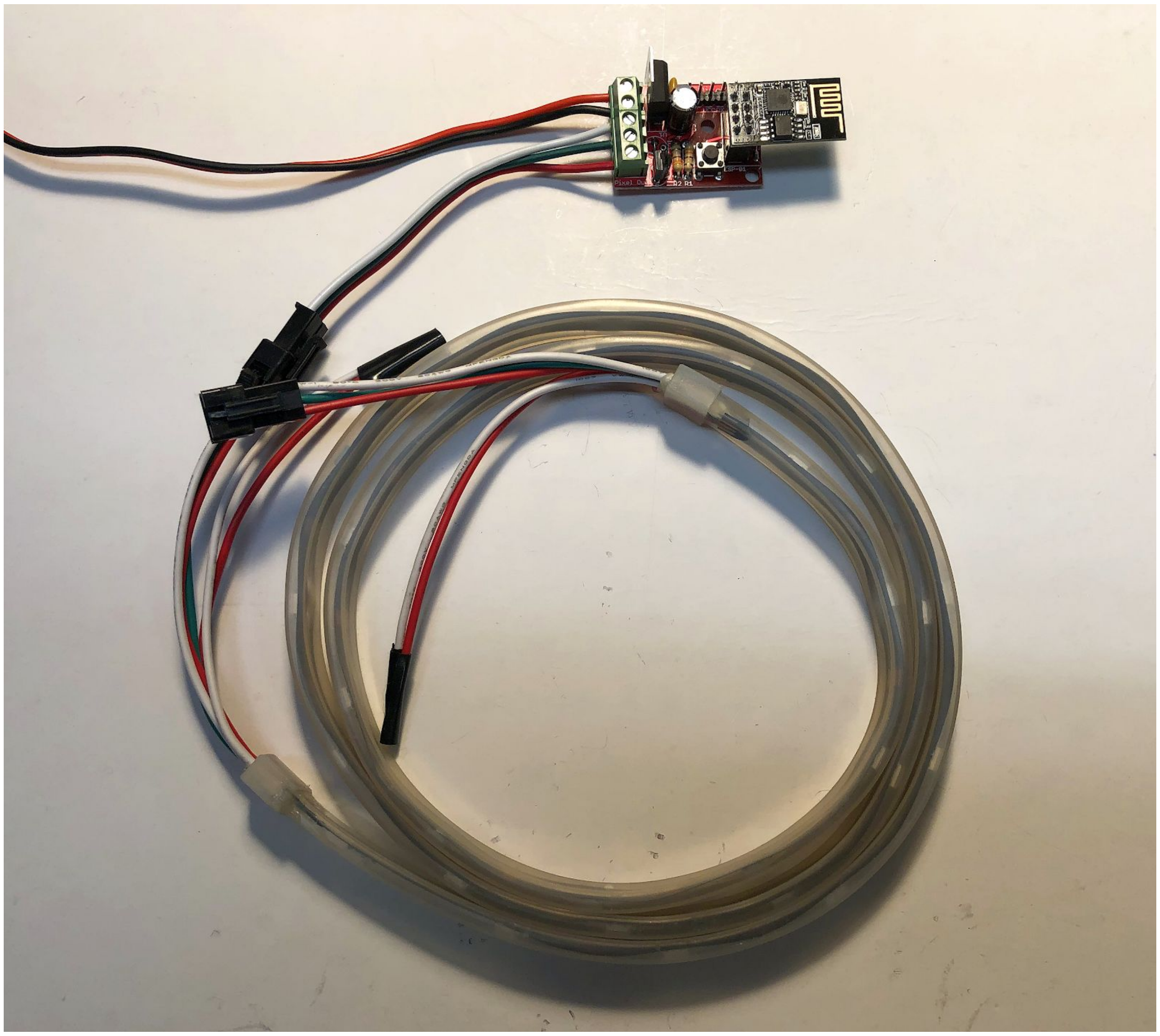
Arduino IDE Upload Method:

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) Check the page (https://github.com/forkineye/ESPixelStick) and ensure you get the required libraries and follow the procedures for the ESPixelStick code
3) Open the source code folder and open the “ESPixelStick.ino” file with your Arduino IDE
4) Click on the “ESPixelStick” tab and update the Wi-Fi credentials for your network.
5) Turn on the +5V power
6) You should see data displayed in the “Serial Output” window of the flash application as the ESP-01 boots up.
7) Power cycle the ESPixelPOP but this time hold down the push button on the ESPixelPOP PCB and release it a few seconds after you have powered up the board.
8) Compile and upload the code to the ESPixelPOP.
9) Reboot the ESPixelPop and if you watch the Serial Monitor window you will be able to see the connection details and make a note of what IP it gets via DHCP from your network.

Step #10 - Connect up your pixels….

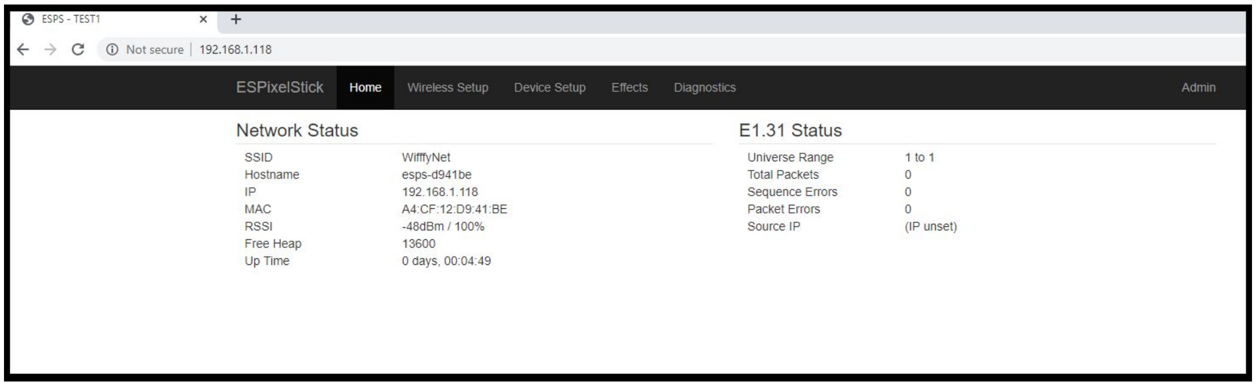
- ❏ Turn off the power to your ESPixelPOP
- ❏ Connect Pixel wires to terminals 3, 4 & 5 as follows:
    - ❏ Pixel GND Wire to ESPixelPOP GND Terminal
    - ❏ PIXEL DATA Wire to ESPixelPOP DATA Terminal
    - ❏ PIXEL Power Wire to ESPixelPOPs VCC Terminal

Note that different pixels may have different coloring codes for the wires. Confirm what yours are before connecting.



Step #11 - Light up your Pixels….

- ❏ Turn on power to your ESPixelPOPs
- ❏ Connect to the ESPixelPOP IP address (you will have noted it above) using an Internet Browser.



Network Status

SSID	WifffyNet
Hostname	esps-d941be
IP	192.168.1.118
MAC	A4:CF:12:D9:41:BE
RSSI	-48dBm / 100%
Free Heap	13600
Up Time	0 days, 00:04:49

E1.31 Status

Universe Range	1 to 1
Total Packets	0
Sequence Errors	0
Packet Errors	0
Source IP	(IP unset)

- ❏ Check out the various tabs for configuration options. I recommend you do the following:
    - ❏ Configure a static IP (makes it much easier to connect when you know the IP)
    - ❏ Configure the Color Order to match your Pixels (Use the Solid Color effect and ensure the LEDs go Red when you select Red)
    - ❏ Configure the Pixel Count
- ❏ Goto the Effects Tab and you will be able to send a variety of test sequences.

You now have control over your Pixels. You can send test E1.31 data from either of these testing applications:

- ❏ da_Tester - https://www.da-share.com/software/da_tester/
- ❏ sACNView = https://sacnview.org/

And of course you can also send both test data and live sequencing data from xLights or any other sequencing application that supports E1.31.

ESPixelStick Configuration Settings (Pixel WS2811 v3.1 Firmware)
Here are all the configuration options along with a quick explanation of what they do:

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 pixels
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.
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 3 channels), Channel shows LED brightness (1 square per channel)
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