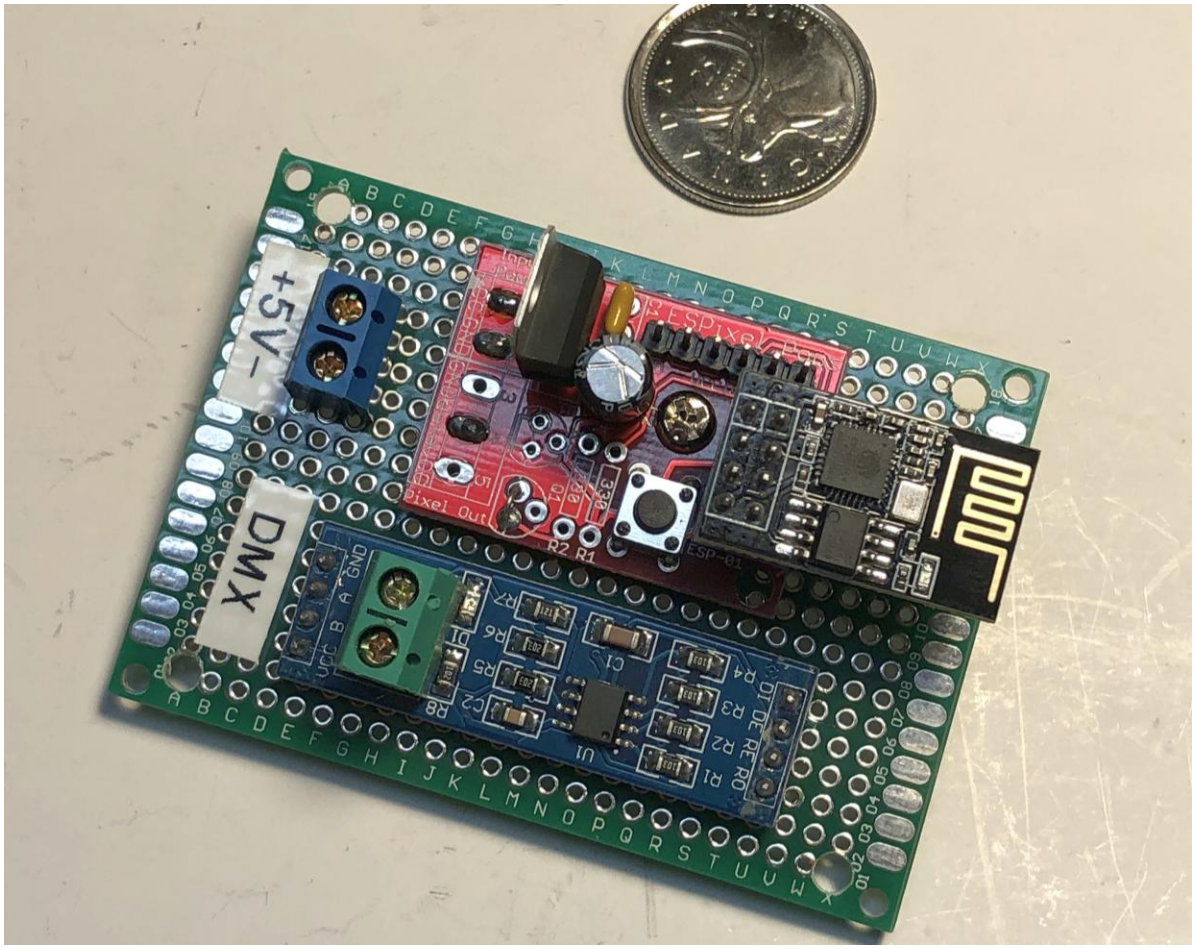
ESPixelPOPs DMX Build & Setup Guide v1.20.2022



In this guide we will put together a DMX version of the ESPixelPOPs. This isn't for controlling Pixels but instead will output a DMX signal for use as an E1.31 to DMX bridge, capable of streaming one Universe of DMX 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

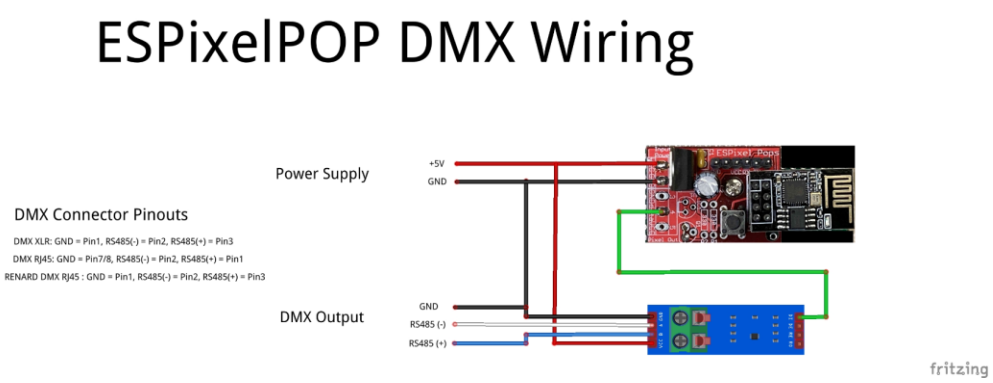
The ESPixelPOPs board has only a few through hole parts and is fairly easy to build. It is helpful if you have had previous soldering experience as it is a small PCB. A bright light and perhaps magnifying reading glasses may be helpful.

You will need the following tools and equipment:

- Drill & ⅛” Drill Bit
- Screwdriver
- Soldering Iron & Solder
- Diagonal Cutters
- USB to Serial TTL Adapter (3.3V or 5V Version)
- #22 or #24 Gauge Bare Wire

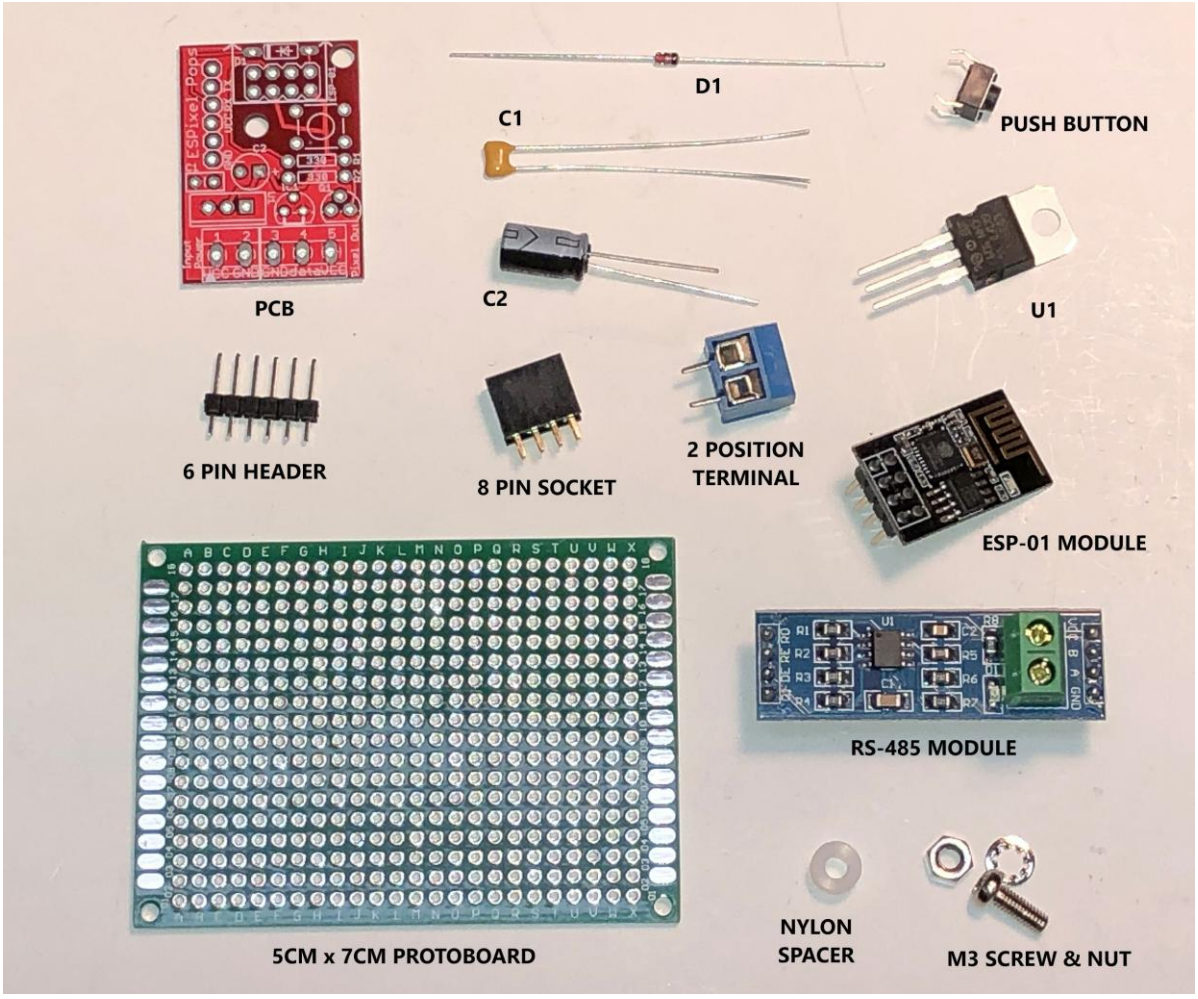
Note that this build is for a 5V power supply only.

Here is a wiring diagram:

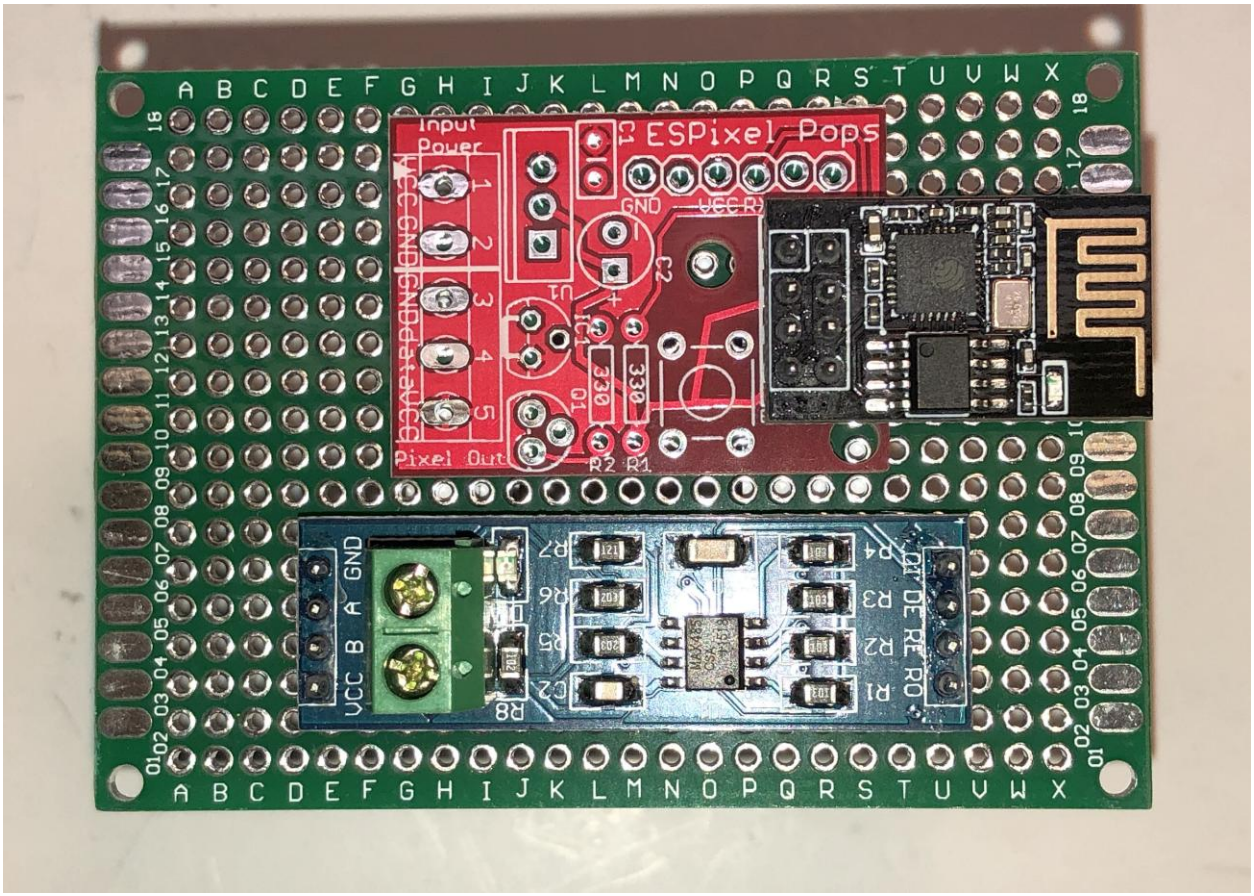


Step #1 - Referring to the picture below, make sure you have all the parts required:

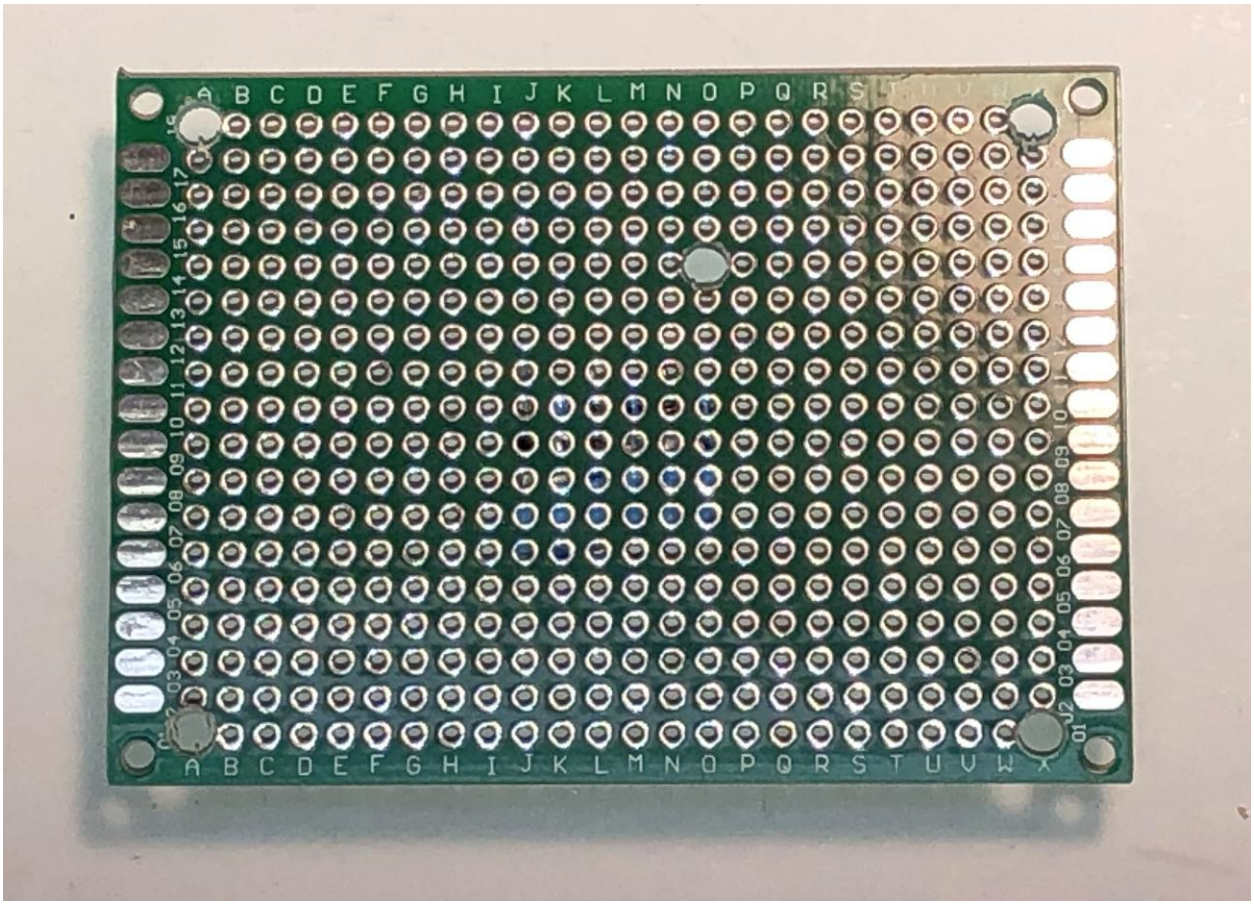
- ❑ 1x 5cm x 7cm Protoboard
- ❑ 1x ESPixelPOPs PCB
- ❑ 1x 1N4148 Diode (D1)
- ❑ 1x 0.1 uF Ceramic Capacitor (C1)
- ❑ 1x 6mm PCB Push Button
- ❑ 1x 2 Position 5.08mm Terminal Strip
- ❑ 1x 6 Pin 2.54mm Male Header Strip
- ❑ 1x 8 Pin (2x4) 2.54mm Socket
- ❑ 1x 220 uF 16V Electrolytic Capacitor (C2)
- ❑ 1x LD1117V33 Voltage Regulator (U1)
- ❑ 1x ESP-01 ESP8266 Module
- ❑ 1x MAX485 RS-485 Module
- ❑ 1x M3 x 3mm Nylon Spacer
- ❑ 1x M3 x 10mm Machine Screw and Nut



Step #2 - Layout the PCB and RS-485 module on the protoboard to determine a location for the mounting screw. If you place the ESP-01 modules pins through the PCB socket holes you can align the board. You also want the PCB terminal strips to roughly line up with holes in the protoboard. Make a note of where these holes line up (here they are in the H row). Mark the ESPixelPOPs center mounting hole (just left of C2 below) on the protoboard (it will be a bit off center).

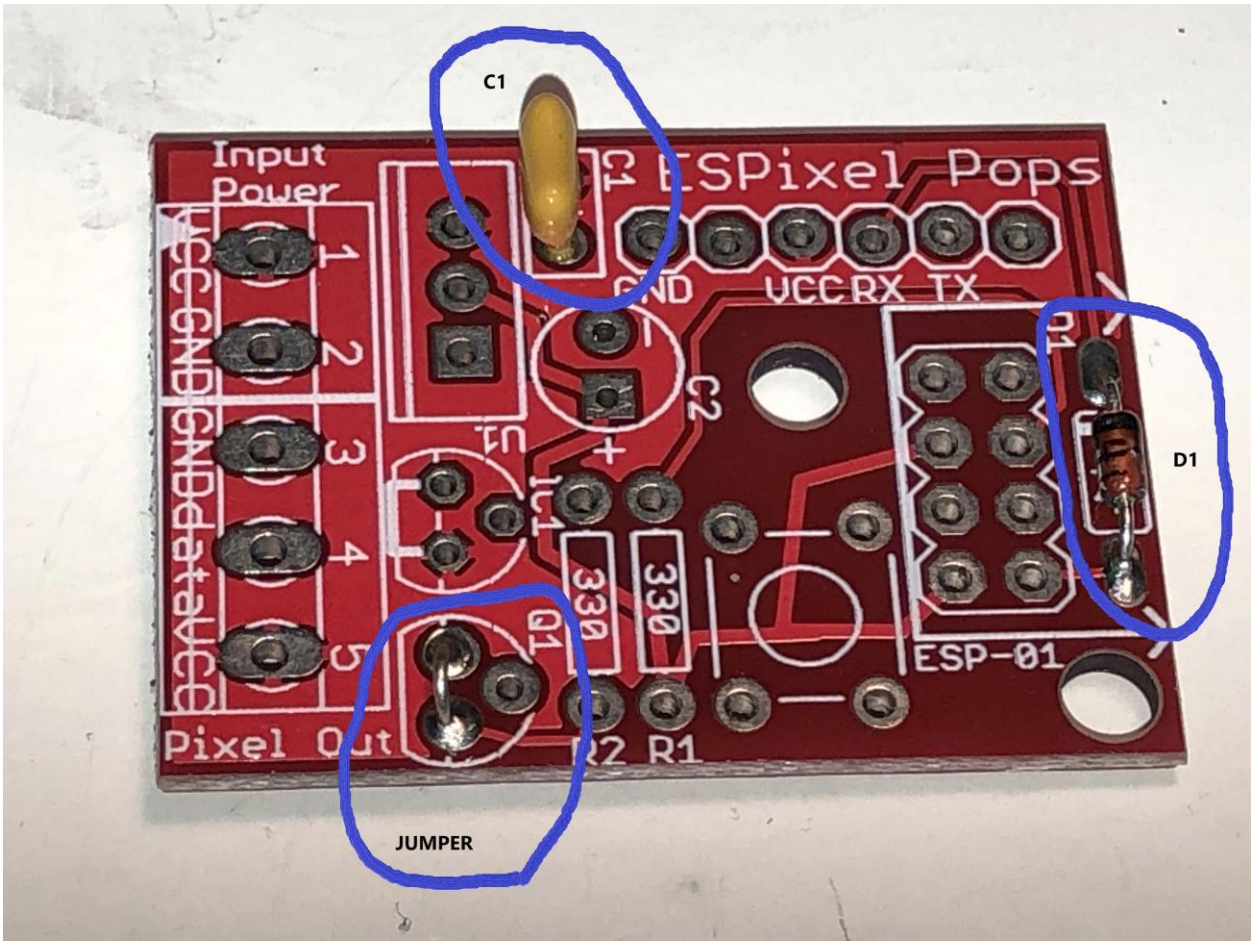


Step #3 - Using a ⅛” bit drill out the mounting hole on the protoboard. I also drill a new mounting hole in each corner of the board as I find the existing ones in the protoboard are too small and too close to the edge to be able to drill out larger.



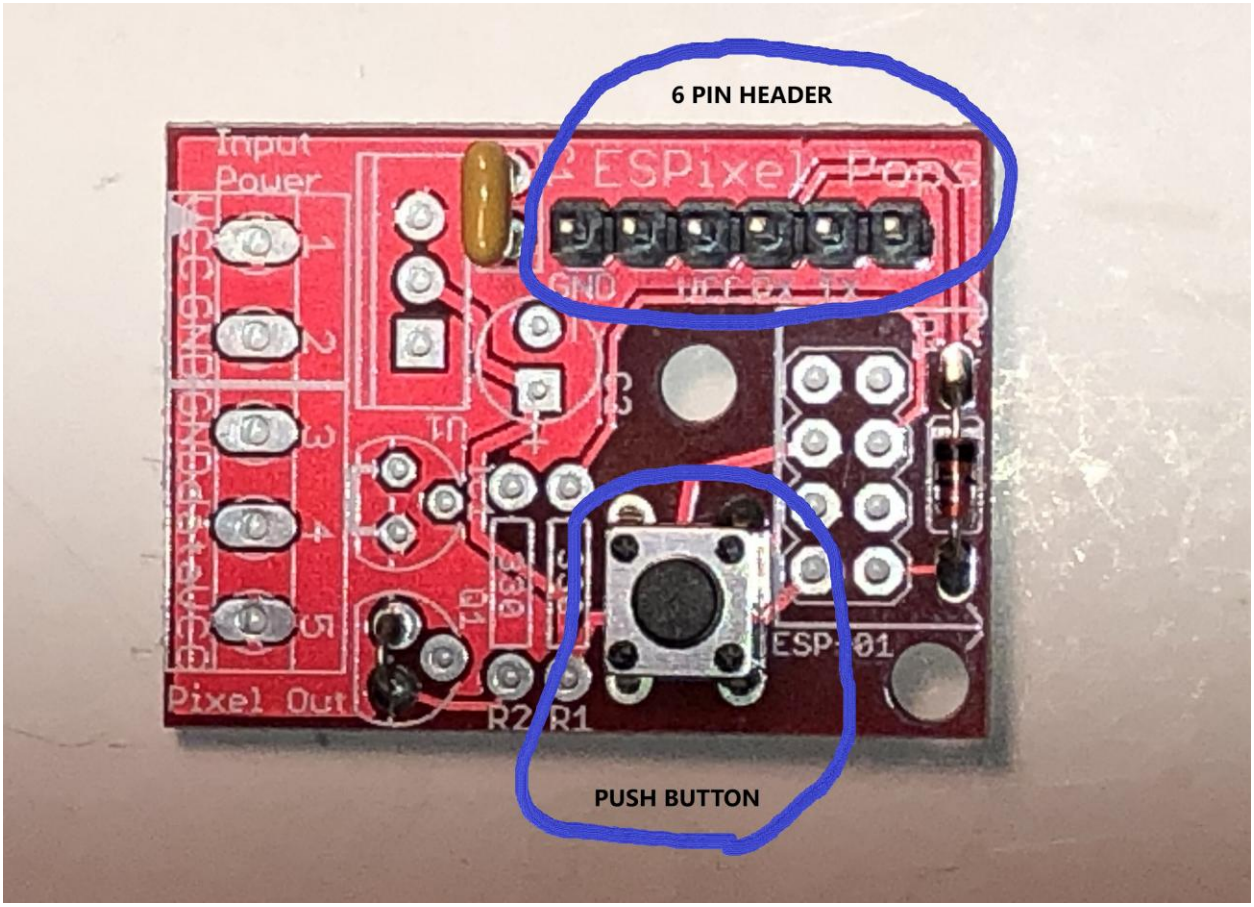
Step #4 - Now start working on the PCB. Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install D1 (ensure orientation matches symbol on PCB)
- ❏ Install C1 (orientation doesn't matter)
- ❏ Install jumper across Q1 terminals as shown. Ensure you jumper Q1 as shown, not IC1 as we do in the 5V Pixel build. Use a cut off lead from D1 or C1. You will need 3 other cut off leads later on so save them till you are completed.



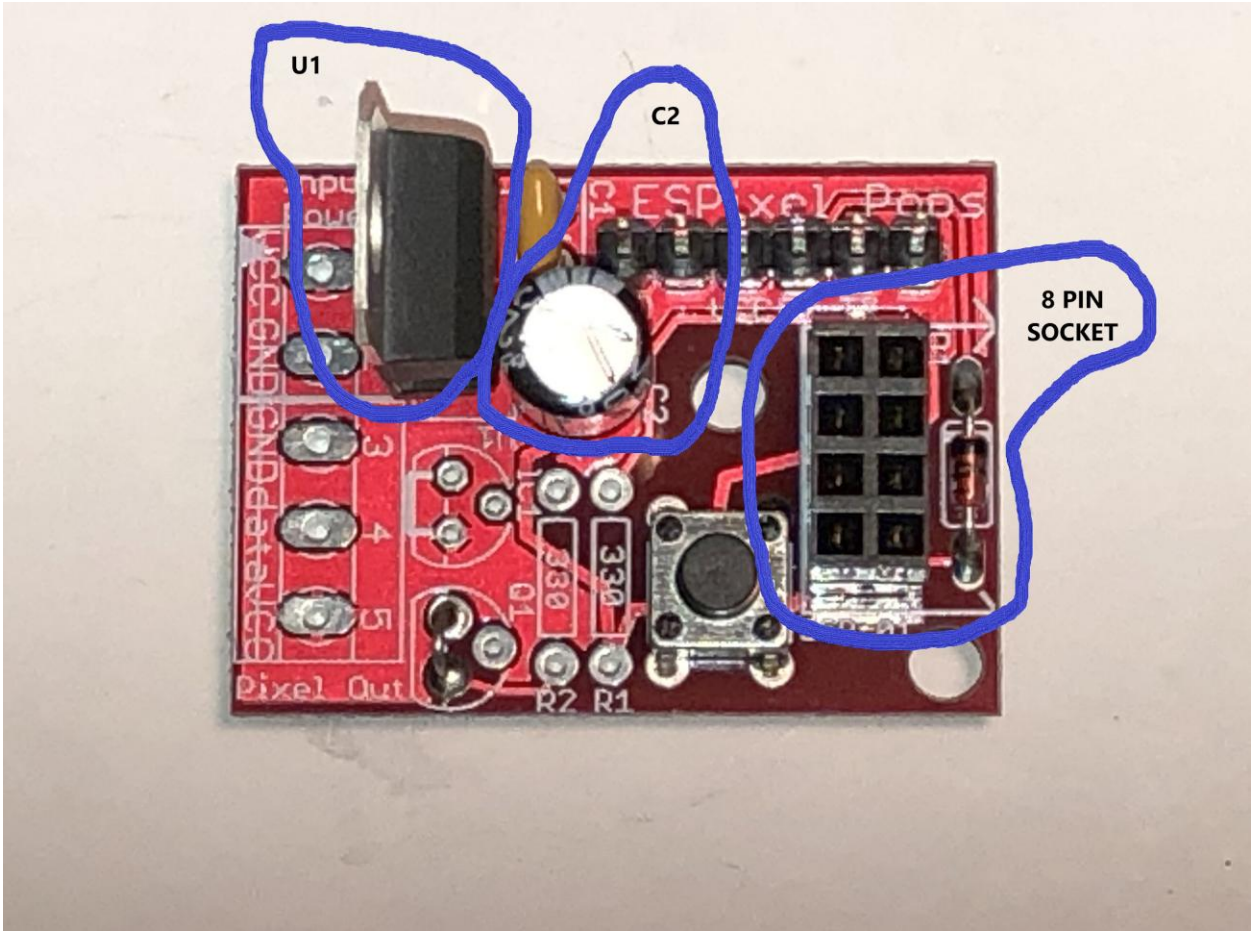
Step #5 - Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

- ❏ Install push button (will only fit in one direction)
- ❏ Install 6 Pin Male Header



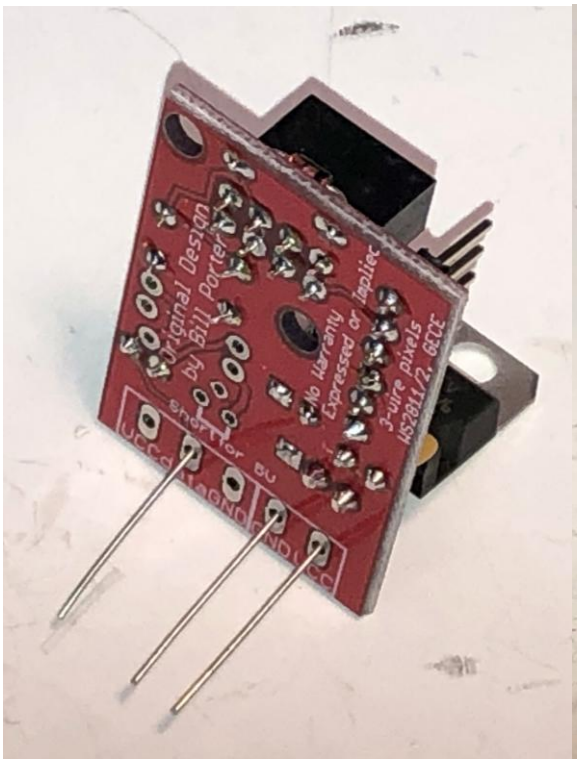
Step #6 - Following the steps below, insert each part, solder it in place and snip off the extra leads. Below each set of steps is a picture showing the parts location:

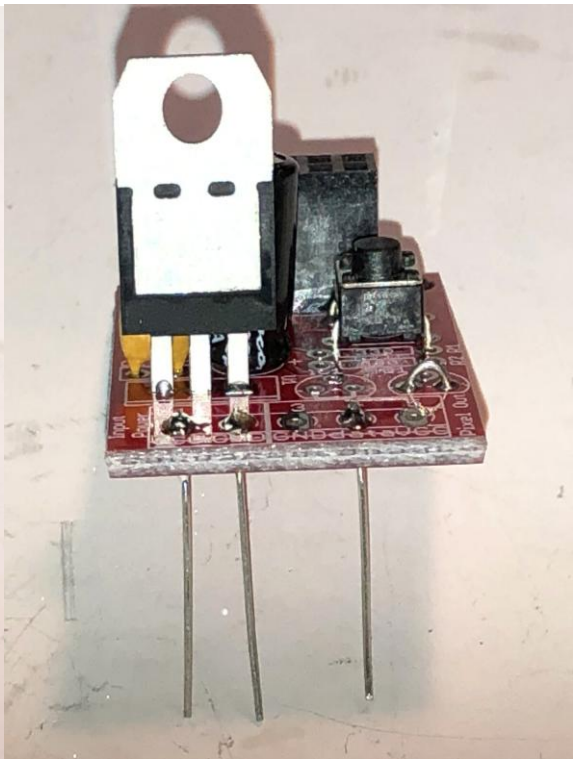
- ❏ Install 8 Pin Socket
- ❏ Install C2 (Ensure long lead goes in square + hole)
- ❏ Install U1 (Ensure metal heatsink faces terminal strip side of PCB)



Step #7 - This part is a bit tricky but you need to solder a short length of cut off lead into the terminal strip VCC, GND and DATA holes.

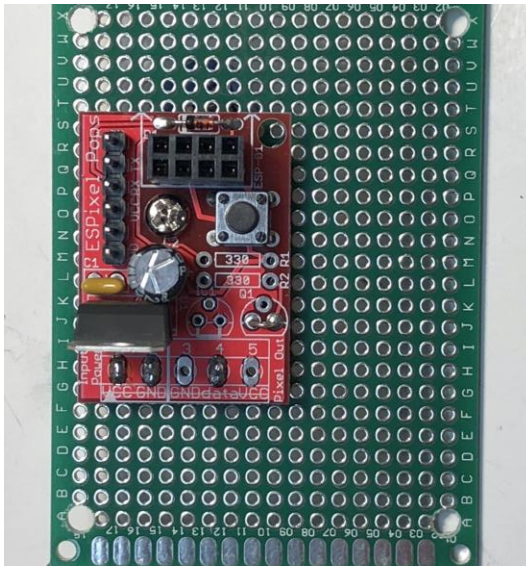
- ❏ Cut 3 pieces of lead each about 1” long (use pieces you have cut off from the other components)
- ❏ Lay the board on it’s edge with the terminal strip down and the parts away from you
- ❏ Position each lead into the hole about ⅛”, hang the longer bit out the bottom of the board
- ❏ Solder each in place. Don’t worry about them being straight as you can bend them into place as needed.
- ❏ Cut off any excess lead on the top side of the PCB

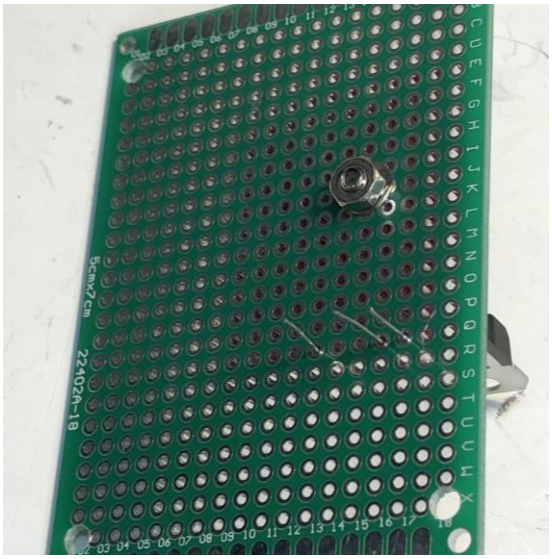




Step #8 - Mount the PCB on the protoboard

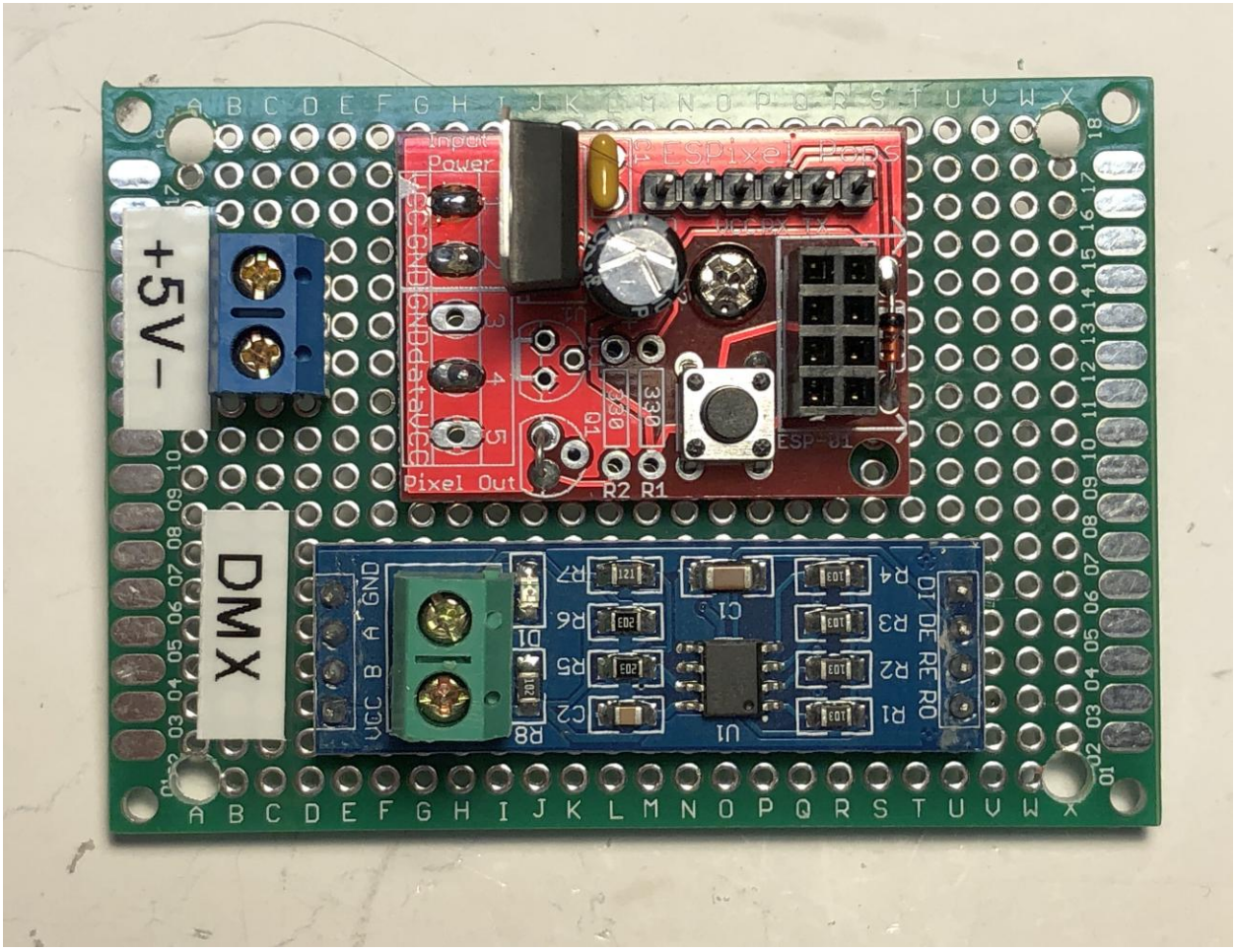
- ❏ Using the 3mm screw and the spacer, mount the ESPixelPOPs PCB to the protoboard.
- ❏ Thread the leads through the matching holes on the protoboard (you made note of the row in Step #2)
- ❏ The leads won't fit perfectly straight so bend as required. You want the PCB securely mounted with the 3 leads sticking out below the protoboard





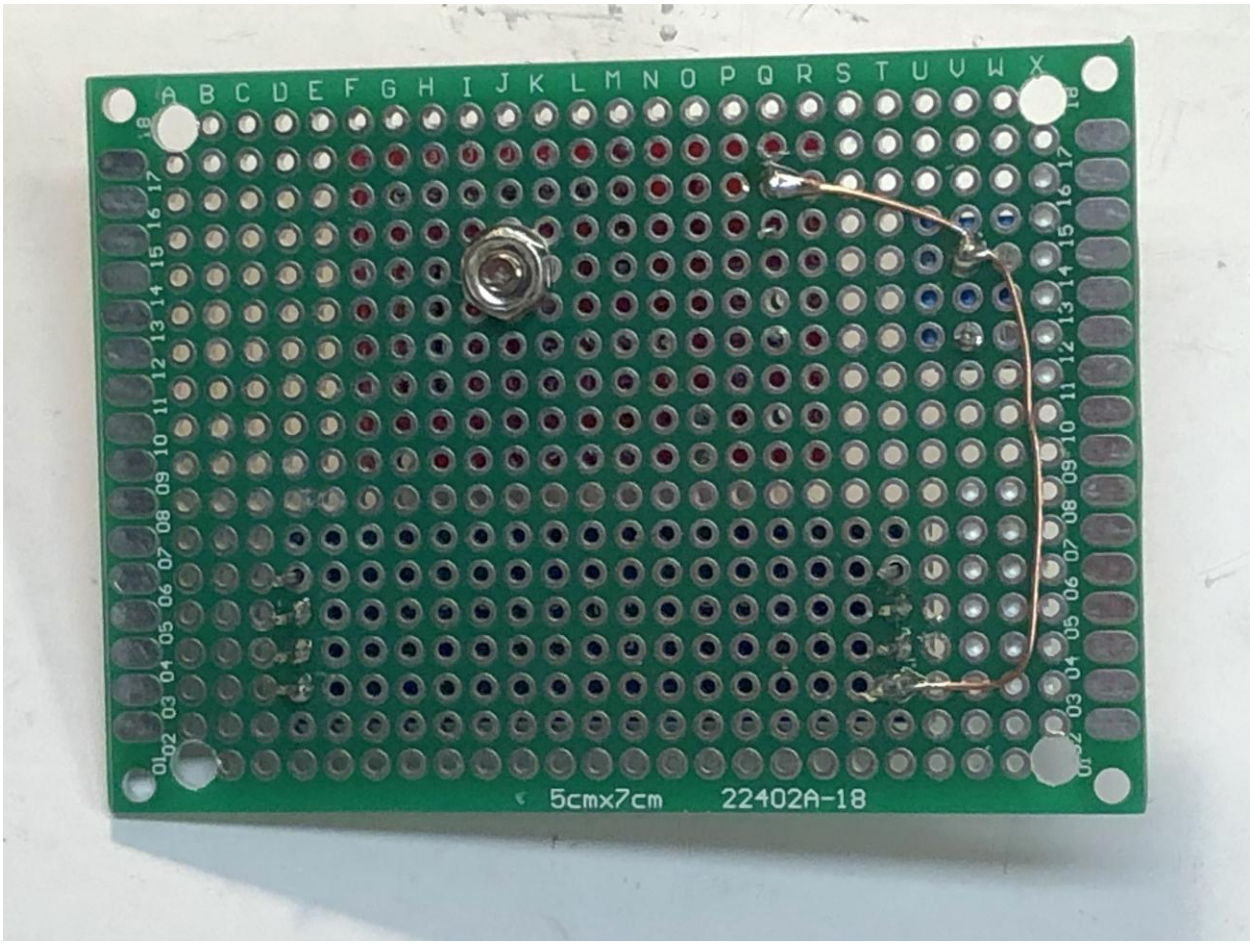
Step #9 - Now install the remaining components on the protoboard. I find it best to just tack these in place with a bit of solder to ensure they don't move while we finish the wiring.

- ❑ Install the 2 Position Terminal Strip
- ❑ Install RS-485 module
- ❑ Label the connections (helps prevent wiring incorrectly)

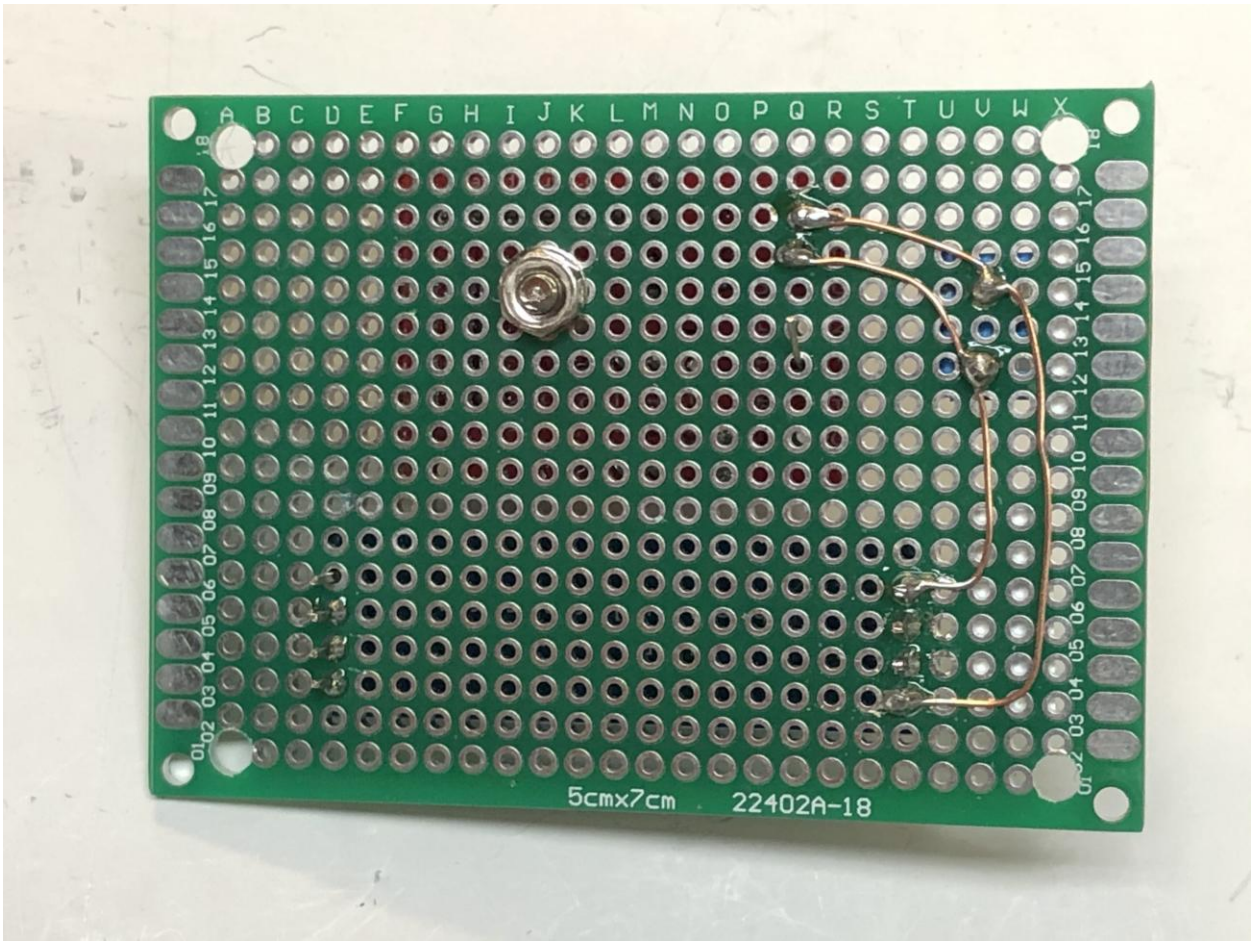


Step #10 - Next we just need to do a bit of wiring to connect it all together. I use 24 gauge solid copper wire from Cat5 cabling that I have scrounged over the years.

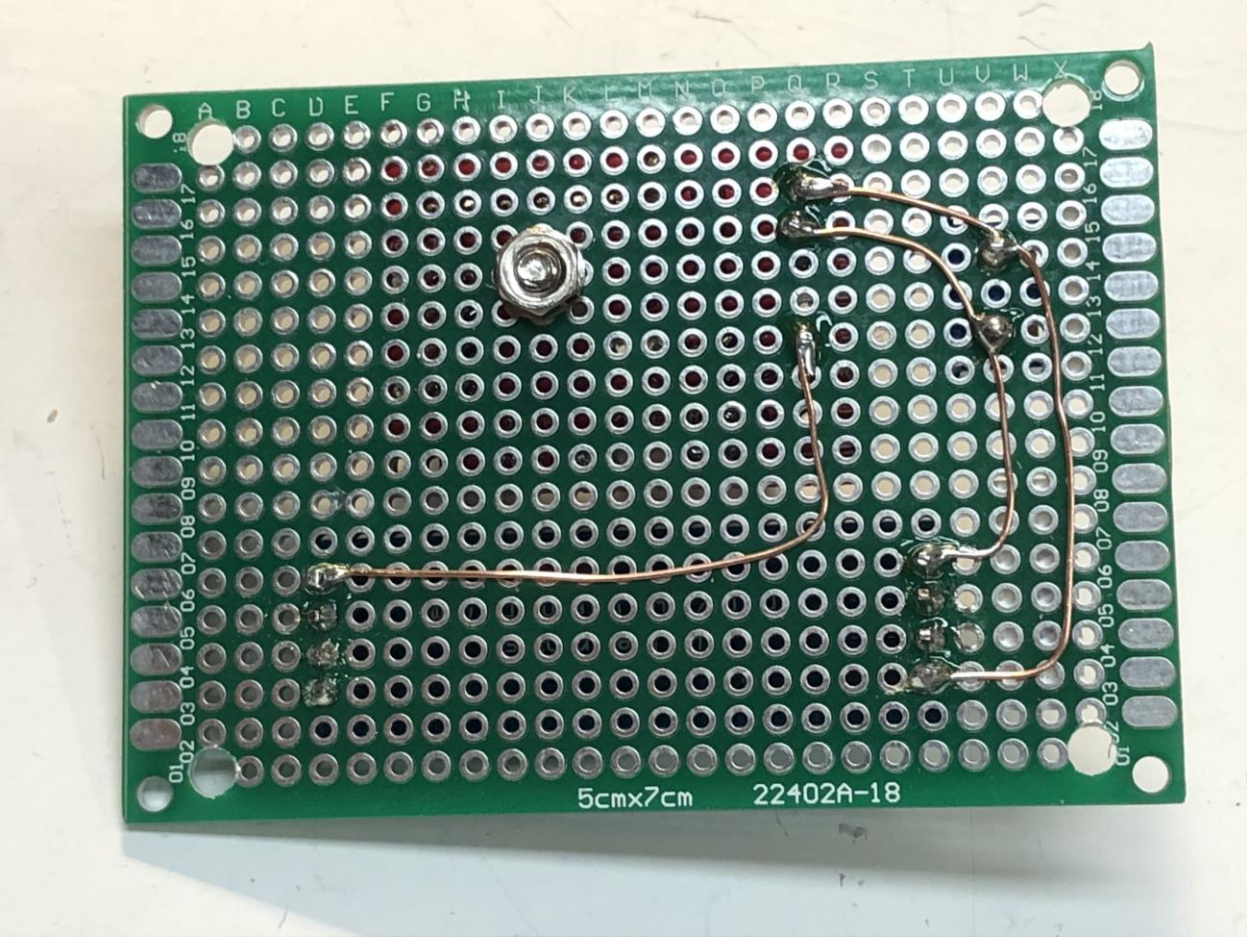
- ❏ Connect +5V from the power input terminal, to ESPixelPOP VCC terminal & to RS-485 VCC terminal.



- ❏ Connect GND from the power input terminal, to ESPixelPOP GND terminal & to RS-485 GND terminal.

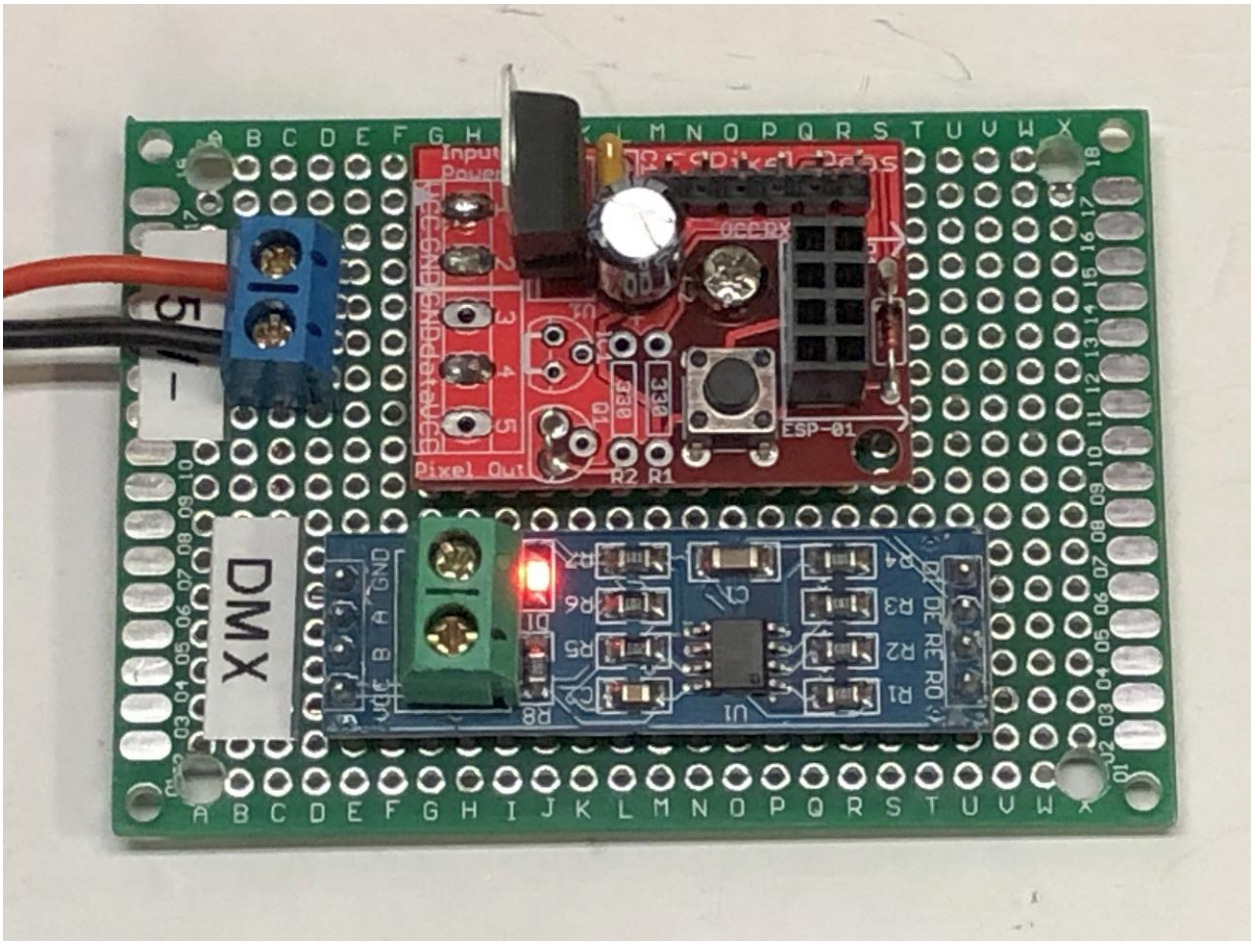


❏ Connect the DI pin from the RS-485 module to ESPixelPOP data terminal



Step #11 - Connect to power and a smoke test

- ❏ Connect the controller to your +5V power supply
- ❏ Ensure the power (D1)LED on the RS-485 module glows



Step #12 - Install ESP-01 module and upload the firmware

- ❏ Turn off the +5V power
- ❏ 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 method is by far the easiest route to follow and recommended unless you need to change something in the source code.

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
- ❏ Turn on the +5V power
- ❏ 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 “Serial(DMX/Renard) v3.1” selected
- ❏ 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.
- ❏ 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.

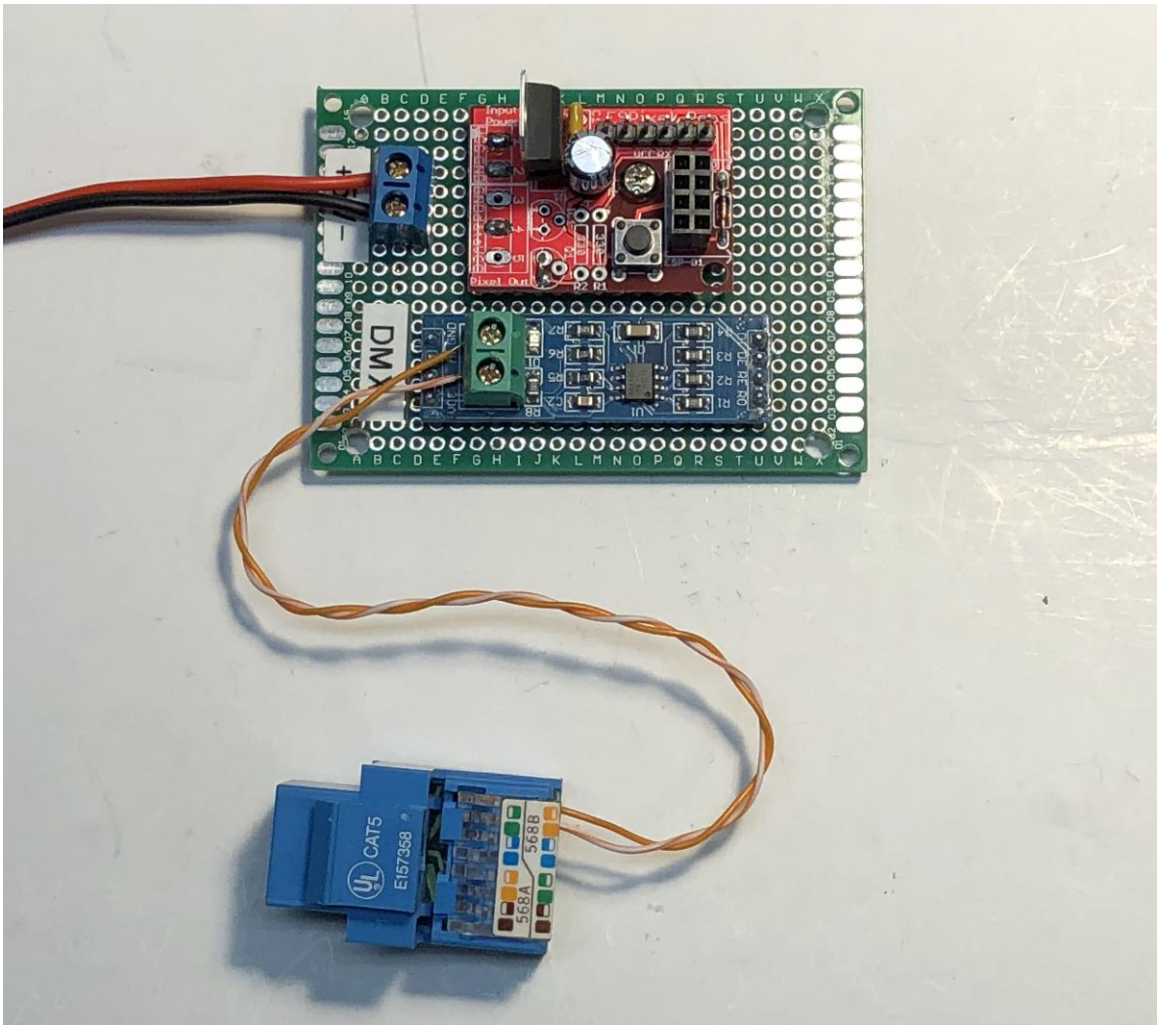
Arduino IDE Upload Method:

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) Click on the “Mode.h” tab and make the following changes:
    a) Comment out the “#define ESPS_MODE_PIXEL” line
    b) Uncomment out the “#define ESPS_MODE_SERIAL” line
6) Turn on the +5V power
7) You should see data displayed in the “Serial Output” window of the flash application as the ESP-01 boots up.
8) 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.
9) Compile and upload the code to the ESPixelPOP.
10) 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 #13 - There are various options for connecting your DMX device to the controller. Use the table below to determine how to wire the connector you want to use:

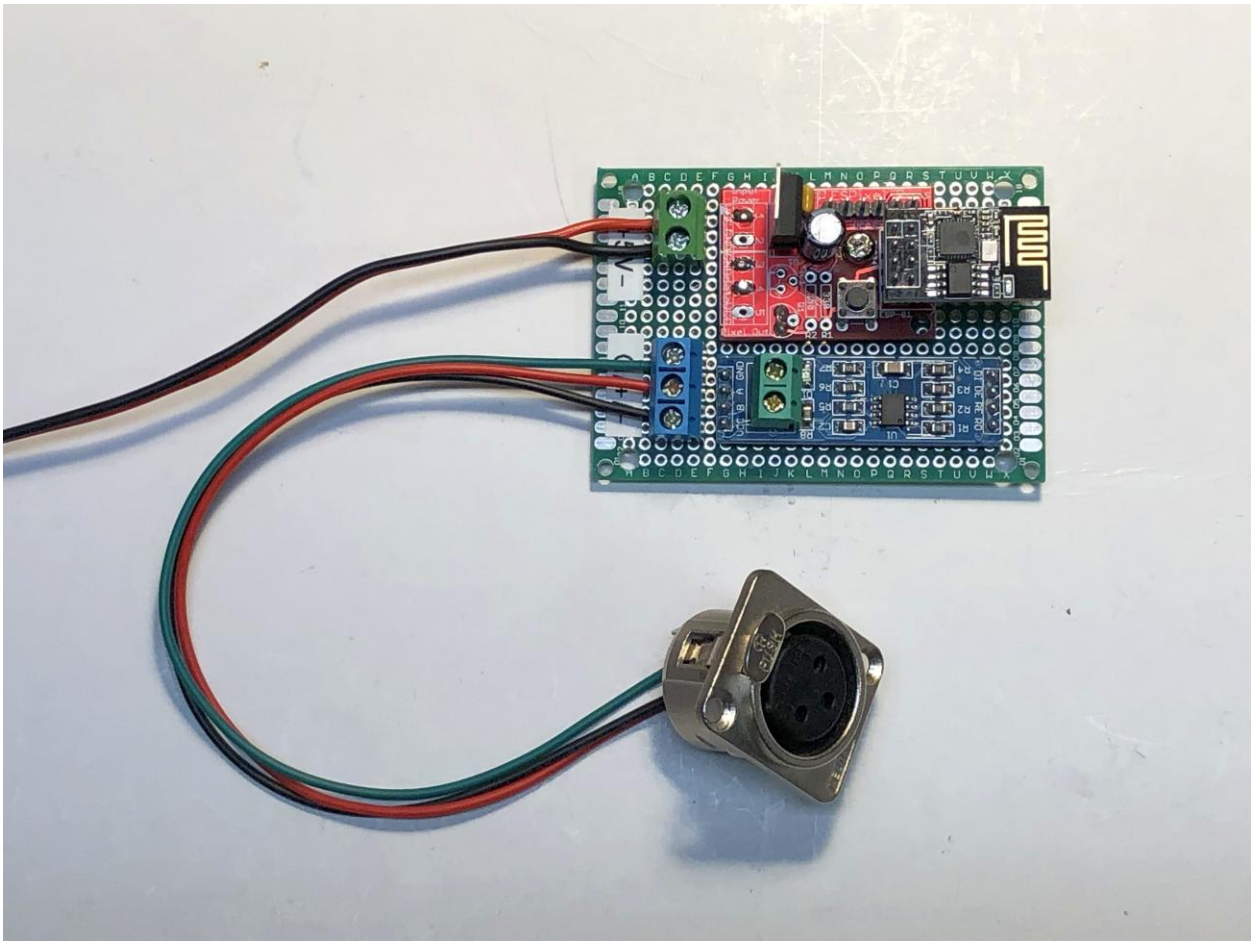
DMX Signal	MAX485 Pin	RS-485 Module Pin	XLR Pinout	RJ45 Pinout
Data - (Inverting)	**7**	**B**	**2**	**2**
Data + (Non inverting)	**6**	**A**	**3**	**1**
GND	**5**	**GND**	**1**	**7**

Below I have connected a RJ45 jack (using pins 1 & 2 as noted above). If the RJ45 jack is wired as TIA-568A then you use the green pair. If the RJ-45 jack is wired as TIA-568B then you use the orange pair.



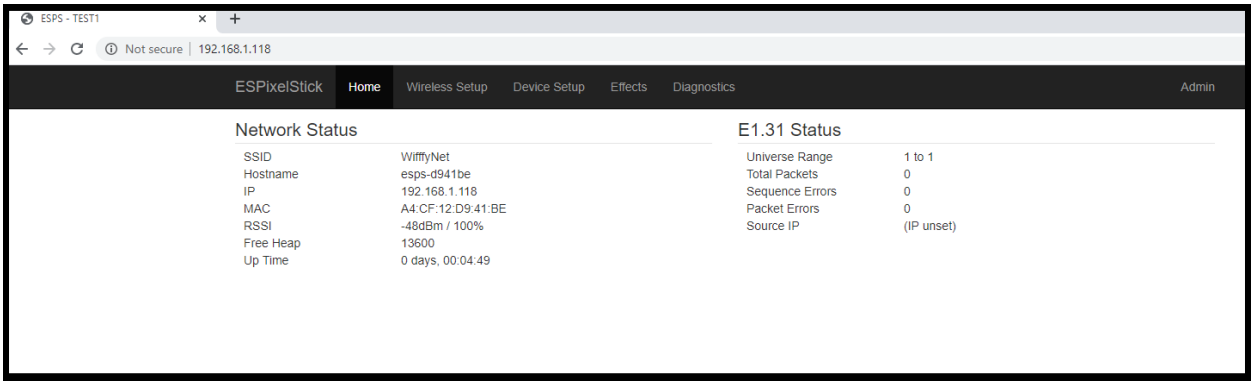
In the RJ-45 example above I have not connected the ground. If this is desired you can connect pin 7 (White/Brown) to the 5V power GND connection.

Below I have connected a 3 pin XLR jack. On this board I added a 3 pin terminal strip instead of using the 2 pin strip built into the RS-485 board.



Step #14 - Start using your ESPixelPOPs DMX

- ❏ Turn on power to your ESPixelPOPs
- ❏ Connect to the ESPixelPOP IP address (you will have noted it above) using an Internet Browser.
- ❏ Go to the **Wireless Setup** tab and configure it with a static IP for ease of connecting in the future.
- ❏ Goto the **Device Setup** tab and setup DMX info:
    - ❏ Make sure the Protocol is set for DMX (Baud Rate will automatically be set for 250000)
    - ❏ Select the Universe you want to use.
    - ❏ I recommend you leave the Start Channel to 1, the Universe Boundary to 512 and the Channel Count to the number of channels you plan to send.
- ❏ Go to the **Effects** tab and you will be able to send a variety of test sequences. Note that most test sequences are meant for RGB light channels and you may get strange results with DMX lights and devices.



You can also send test data to your ESPixelPOPs via E1.31 using either of these testing applications:

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

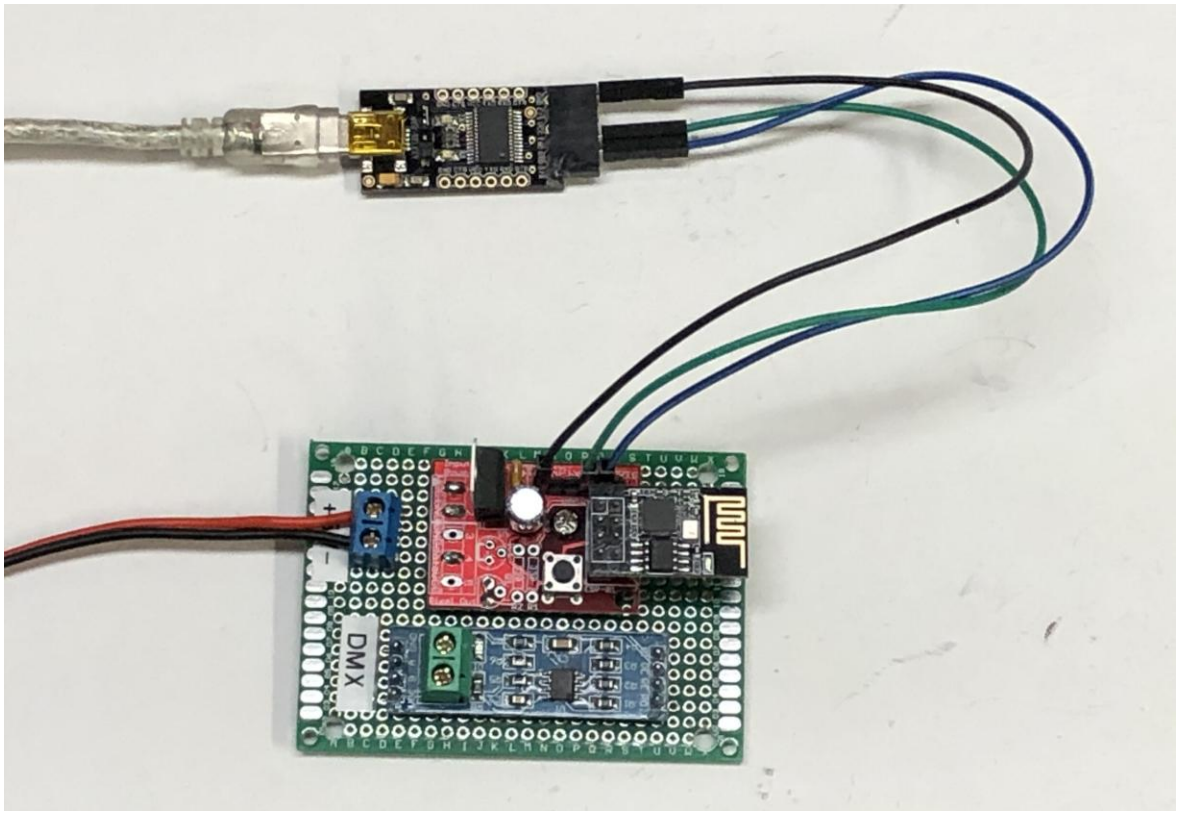
Naturally you can also send both test data and live sequencing data from xLights or any other sequencing application that supports E1.31.

ESP-01 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. Do not connect your USB-TTL adapter directly to the ESPixelPOPs board. Do not connect any of the other pins from the USB-TTL adapter to the ESPixelPOPs board.

You can use either a 3.3V or a 5V version of a USB-TTL adapter to program the ESP-01. 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 that 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.



ESPixelStick Configuration Settings (Serial DMX 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 controller is listening to
Start Channel - DMX channel the controller is starting at
Universe Boundary - The last channel used in the DMX universe.
Enable Multicast - Checked if you want to use Multicast.

Serial Configuration:

Channel Count - DMX channels used
Protocol - Select DMX512 or Renard output
Baud Rate - Adjust the baud rate for Renard output (Fixed at 250000 for DMX output)
Refresh Rate - Shows the maximum refresh rate you can get based on the settings you select.

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 channels
All leds same - Make all channels 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 channel brightness (1 square per channel). Note that the RGB colors may not match your DMX lighting output.
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 - FLash 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