

ESPixelStick V2 Assembly and Usage

OVERVIEW

The ESPixelStick is a wireless (802.11g/n) pixel controller that interfaces as a standard sACN / E1.31 controller and supports a variety of pixel types. It also supports serial (RS-232) output of DMX¹ and Renard protocols. The firmware is open source and written in the EPS8266 Arduino environment to allow the user to modify and experiment with the firmware as they see fit. The ESPixelStick is provided in kit form with all SMD components pre-soldered. It is a fairly easy kit to solder with a low component count.

KIT CONTENTS

For each ESPixelStick, your kit will contain the following:

Quantity	Part Description
1	ESPixelStick V2 Main Board (SMD pre-soldered)
2	6x6mm Tactile Switch
1	1x3 Male Header
1	2x4 Female Header
1	6 Pin Plug
1	6 Pin Right Angle Socket
1	Mini ATM Fuse Holder
1	3A Mini ATM Fuse
1	ESP-01 Module ² (optional)

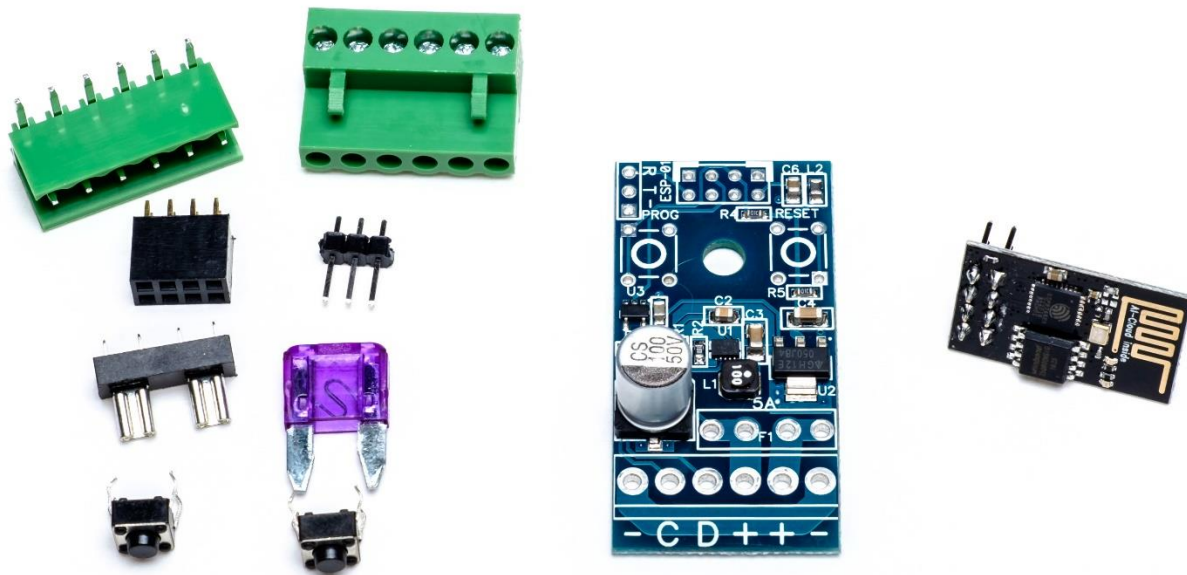


Figure 1 – ESPixelStick V2 BOM

¹ For connection to DMX devices, you will need to bypass the RS-485 interface, or use a RS-232 to RS-485 converter.

² The ESP-01 Module is required for completion, but is optional when purchasing the kit.

ASSEMBLY INSTRUCTIONS

Before beginning assembly, it is recommended that you fully read the Assembly Instructions section. Verify your BOM (Bill of Materials / Kit Contents above) and visually inspect the ESPixelStick Main Board. Verify that there are no issues with the board such as broken traces or shorted components. The ESPixelStick Main Board has already been visually inspected, but it's always recommended to perform a final check before assembly. The order of assembly really isn't that important, but I usually work from the smallest components to the largest.

- ☐ Install and solder **6x6mm Tactile Switch**, at **PROG** and **RESET**. Note the orientation of the switches below.
- ☐ Install and solder **2x4 Female Header**, at **ESP-01**.
- ☐ Install and solder **1x3 Male Header**, next to **ESP-01** and **PROG**. This is the programming header.
- ☐ Install and solder **Mini ATM Fuse Holder**, next to where it says "5A".
- ☐ Install and solder **6 Pin Right Angle Socket** at the end of the board, left of the fuse holder.
- ☐ Insert the **ESP-01 Module** into the 2x8 Header, at **ESP-01**. The module must be inserted with the antenna portion (gold zig-zag) oriented away from the board.
- ☐ This completes assembly of the ESPixelStick.

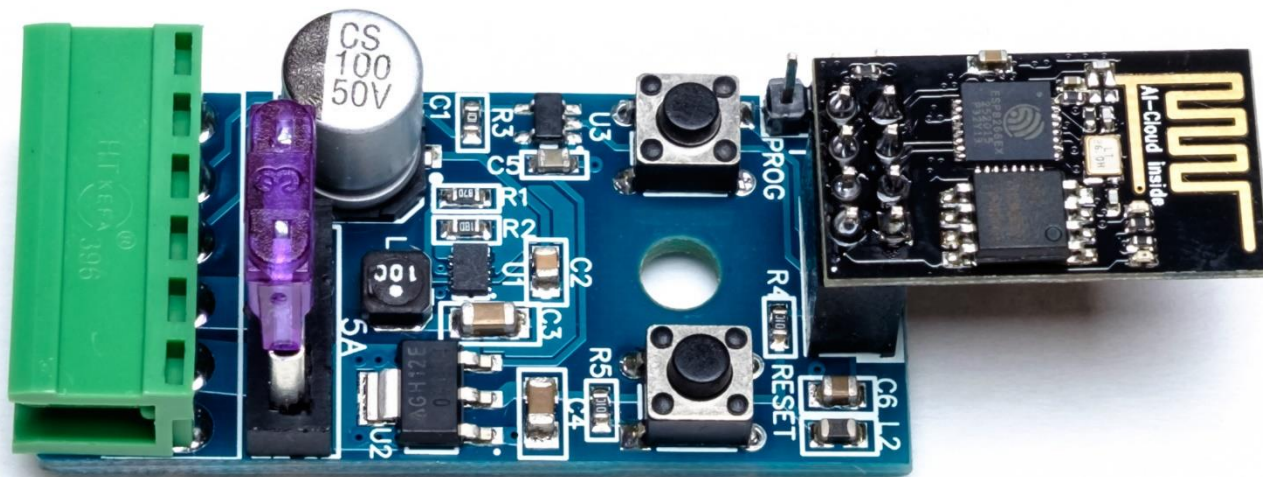


Figure 2 - Completed ESPixelStick

All power and data connections are made through the 6 Pin connector located on the end of the board. Wiring designators are indicated on the bottom of the ESPixelStick. The ESPixelStick is designed to be located near the element that it's controlling, eliminating the issue of having to deal with long pixel transmission lines and null pixels.



Figure 3 - Bottom of ESPixelStick

- **VIN -** : Power Supply Ground
- **VIN +** : Power Supply Positive
- **Pixels +** : Pixel Positive
- **Pixels D** : Pixel Data Line / Serial TX Line
- ***Pixels C** : Pixel Clock Line
- **Pixels -** : Pixel Ground / Serial Ground

*Note that clock-less 3-wire pixels (WS2811/2, GECE, etc.) will not use the **Pixels C** connection. It is reserved for future support of clocked pixels such as WS2801.

Usage of the onboard fusing and power distribution of the ESPixelStick is optional. If your pixels are being fed power directly by a separate power supply or fuse block, the ESPixelStick will be back-fed power by the pixel power connection (**Pixels +** and **Pixels -**). The following examples show V1 boards, however the connections are the same.

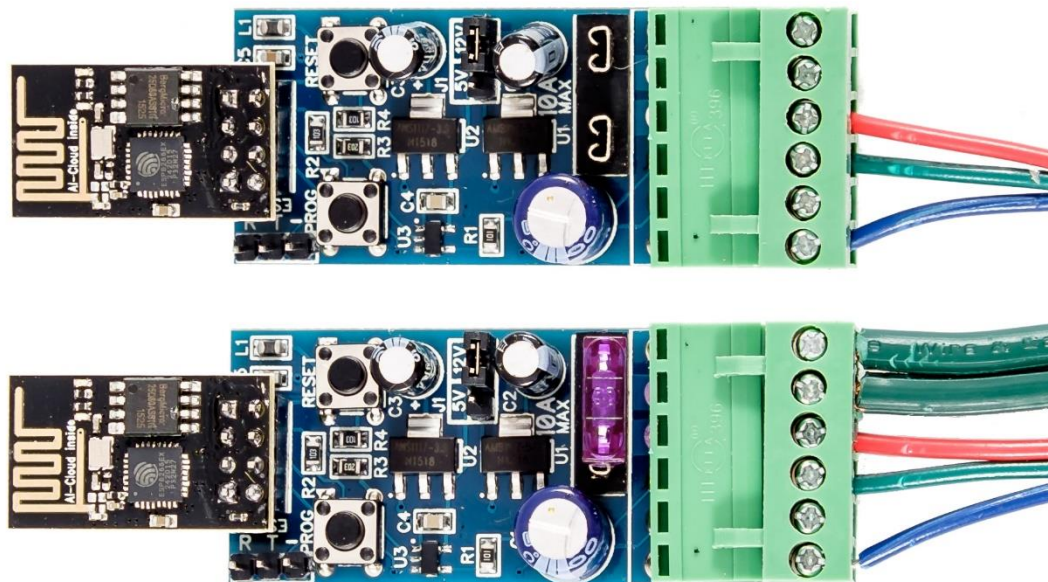


Figure 4 – Back fed power (top) and power fed through ESPixelStick (bottom)

PROGRAMMING

Programming can be done by compiling from source in the Arduino environment, or by using pre-compiled binaries and ESPSFlashTool. This guide will cover using ESPSFlashTool to flash pre-compiled binaries on a Windows machine and is the recommended method for programming your ESPixelStick. If utilizing a Linux or Mac machine, be sure you have write access to your serial port. If you would like to build from source yourself, please refer to the ESPixelStick GitHub page - <http://github.com/forkineye/ESPixelStick>.

Programming your ESPixelStick requires a suitable USB to Serial adapter with the GND, TX, and RX lines exposed. Such adapters are commonly referred to as “USB to TTL” and will advertise serial chips such as FTDI, PL2303, CP2102, and CH340G. CP2102 based devices are recommended and they can be bought for just a few dollars online. Others, such as Prolific and FTDI are notorious for having clones on the market and will fail if the driver thinks a clone is connected. We’ll go over some requirements, connecting the ESPixelStick and finally programming the ESPixelStick.

REQUIREMENTS

- ☐ USB to TTL Serial Cable
- ☐ Java 8 runtime - <http://www.java.com>
- ☐ Latest ESPixelStick Firmware package - <http://github.com/forkineye/ESPixelStick/releases>

PROGRAMMING INTERFACE

The ESPixelStick must be connected to a power supply through the main plug for programming. From the programming connector, connect the **R**, **T**, and **-** lines to the **RX**, **TX**, and **GND** connections of your USB to TTL Serial Cable. The following examples shows a V1 board, however the connections are the same.

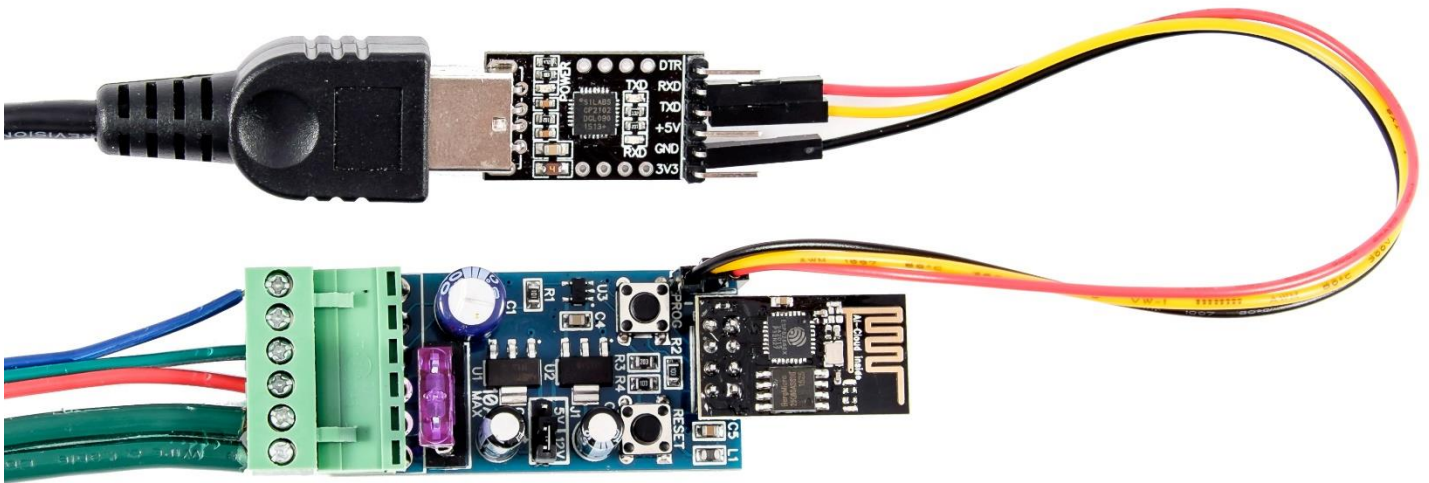


Figure 5 - Programming Connections

Tip – If programming multiple controllers, you may find it easier to dedicate one controller as the programmer until done. All code and configuration is stored on the ESP-01 module itself, so you can just swap out modules on the ESPixelStick Main Board that you're using as the programmer. Note however, it is recommended that you disconnect power to the ESPixelStick while swapping modules. The programming cables however may stay connected.

- ☐ Download the latest ESPixelStick Firmware release from <http://github.com/forkineye/ESPixelStick/releases>. Ignore the source packages listed, you only need ESPixelStick_Firmware-*.zip.
- ☐ Extract the ESPixelStick_Firmware zip file into its own directory.
- ☐ Launch ESPSFlashTool by double-clicking on **ESPSFlashTool.jar**.
- ☐ Enter the **SSID** and **Passphrase** for your access point and change **Device ID** if you desire.
 - **Device ID** is just a plain text identifier to help you tell your ESPixelSticks apart. It can also be changed via the web interface once programmed. Typically, locations or element names make good ID's (i.e. – Lower Windows, Mini Tree 1, Matrix, etc.)
- ☐ Select **Device Mode** to choose if you want this to be a *Pixel* or *Serial* device. This cannot be changed at runtime, however you can flash your device over the web with *Pixel* or *Serial* firmware once it is programmed.
- ☐ Select your **Serial Port**.
- ☐ Enter programming mode on the ESPixelStick using the following procedure
 - Hold down the **PROG** button.
 - Press and release the **RESET** button.
 - Wait 2 seconds.
 - Release the **PROG** button.
 - The ESPixelStick is now in programming mode.
- ☐ Click **Upload** to program your ESPixelStick.
- ☐ Once the upload is complete, the ESPixelStick will relay its configuration status to the *Serial Output* window. You can now connect to the ESPixelStick with your web browser for further configuration if required. It's a good idea to record the IP Address as chances are, you'll need it!
- ☐ Your ESPixelStick is now programmed.

CONFIGURATION AND USAGE

Configuration and usage of the ESPixelStick is pretty straight forward for anyone familiar with 802.11 wireless networks and sACN / E1.31. Here are some details to get you up and running.

A NOTE ON 802.11 WIRELESS NETWORKS FOR SHOW USAGE

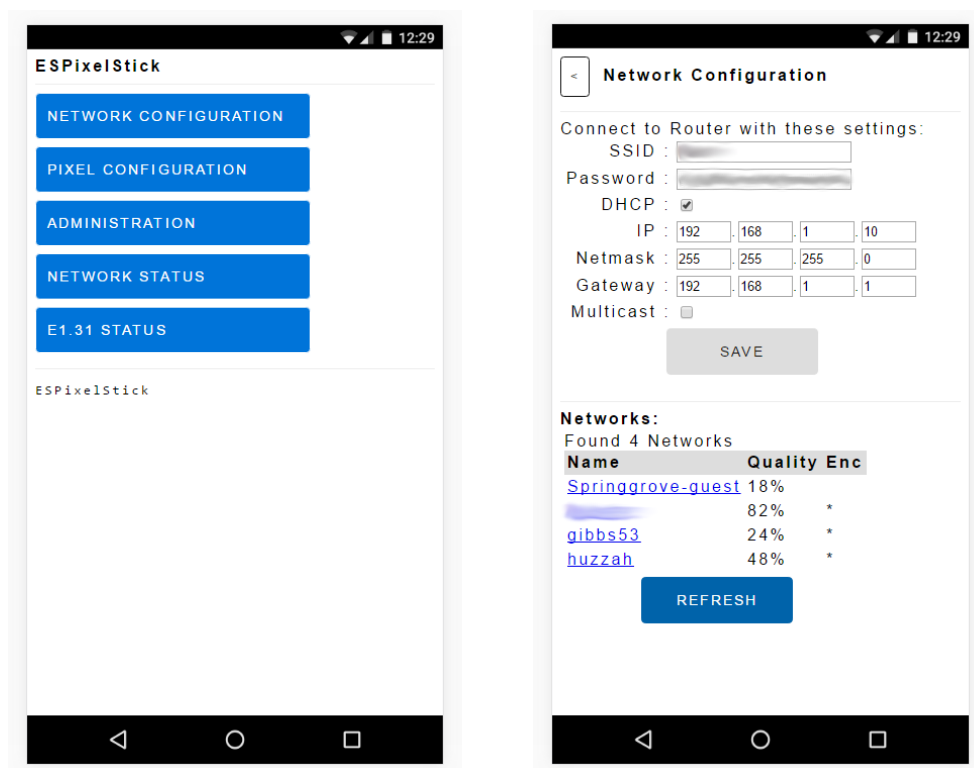
Just like a wired show network, your wireless show network should be separate from your home network and dedicated solely for your show. Due to the simplex nature of 802.11, your sACN source (Vixen, xLights, FPP, etc.) **MUST** be hardwired or you will experience significant show lag as the access point will become overloaded very quickly.

If you're planning to use Multicast, be aware of the issues that may be caused by non-show devices attempting to access your network and how to mitigate them. Mainly, keeping all non-show devices off the network and disabling SSID broadcast for your show AP. Mobile devices in particular connecting to a WiFi network can significantly degrade Multicast transmission rates and affect show performance.

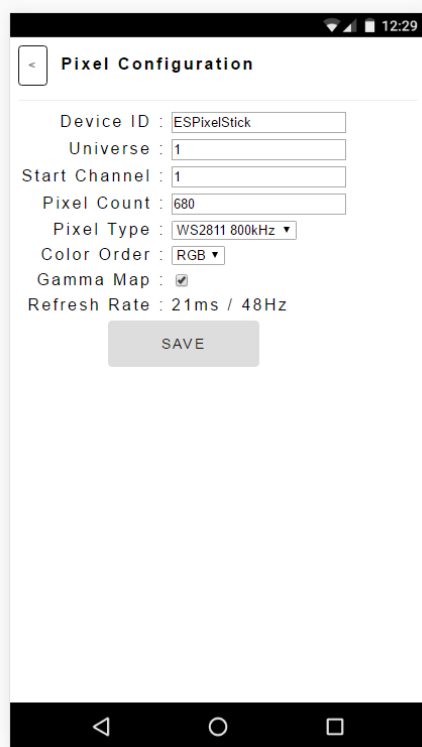
If you must use your home network, Unicast only is recommended and be aware that other devices on your network may degrade your show performance.

WEB BASED ADMINISTRATION

The ESPixelStick has a mobile friendly web-based administration interface that provides configuration and status reporting. To connect, simply browse to the IP Address reported by your ESPixelStick during programming. The examples in this section are based on the ESPixelStick v2.0 Firmware.



The web interface home page and Network Configuration page are self-explanatory. Note however, **Multicast** is only supported for a single universe at this time.



Pixel Configuration

Device ID : ESPixelStick

Universe : 1

Start Channel : 1

Pixel Count : 680

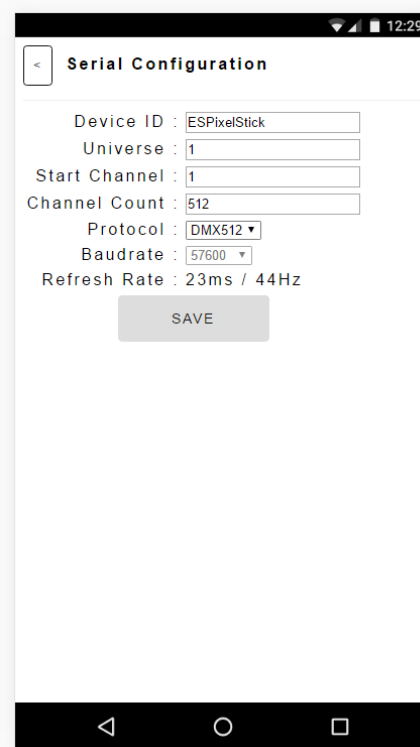
Pixel Type : WS2811 800kHz ▾

Color Order : RGB ▾

Gamma Map : ☒

Refresh Rate : 21ms / 48Hz

SAVE



Serial Configuration

Device ID : ESPixelStick

Universe : 1

Start Channel : 1

Channel Count : 512

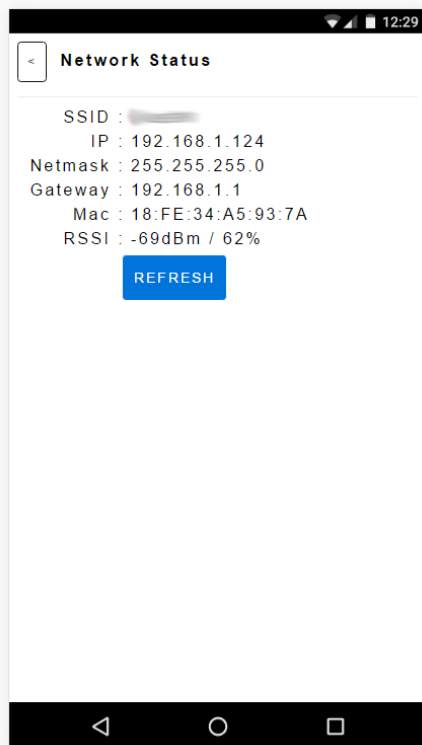
Protocol : DMX512 ▾

Baudrate : 57600 ▾

Refresh Rate : 23ms / 44Hz

SAVE

The **Device ID** is a free-form field to help you know which ESPixelStick you are logged into it. The text here is displayed at the bottom of the homepage and in the title bar of your web browser. **Gamma Map** applies only to 8 bit pixels and will apply a pseudo-gamma correction of 2.2. You may want to use this if you don't already have custom dimming curves setup in your sequencer. The **Refresh Rate** is an estimated maximum refresh rate based off the protocol and number of channels you are using.



Network Status

SSID : [REDACTED]

IP : 192.168.1.124

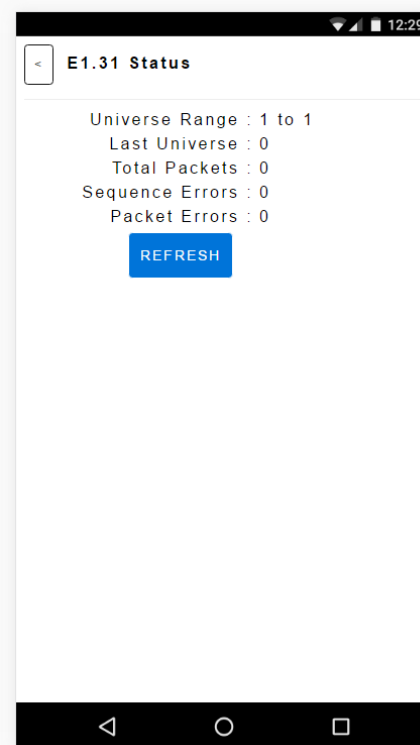
Netmask : 255.255.255.0

Gateway : 192.168.1.1

Mac : 18:FE:34:A5:93:7A

RSSI : -69dBm / 62%

REFRESH



E1.31 Status

Universe Range : 1 to 1

Last Universe : 0

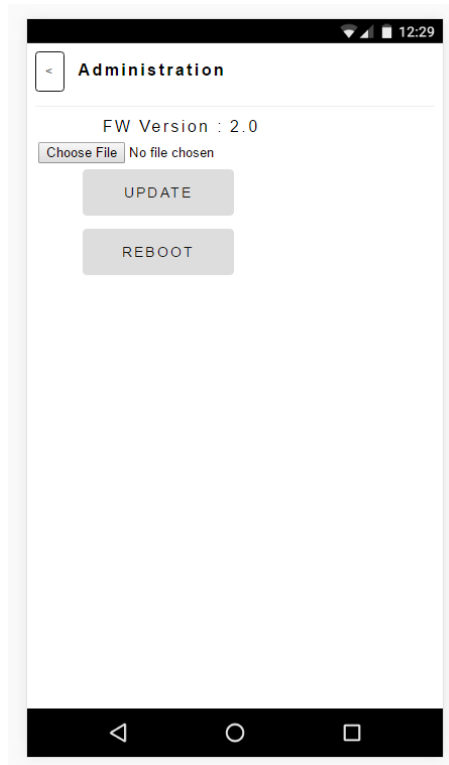
Total Packets : 0

Sequence Errors : 0

Packet Errors : 0

REFRESH

The Network Status and E1.31 Status pages are self-explanatory.



The Administration page allows you to update your firmware over the web using .EFU (ESP Firmware Update) files. These update files will be included in all firmware binary releases from 2.0 forward.

SEQUENCER CONFIGURATION

Directions will vary for all the various sequencing programs available. Follow your product's instructions for setting up an "E1.31" or "sACN (Streaming ACN)" controller.

