MadMapper USB to LED Tutorial

Introduction

MadMapper 2.1 offers the possibility to send video pixels directly to LED devices.

When controlling LED devices from MadMapper you use « MadLight », which is inside MadMapper (activated by placing DMX Fixtures / DMX Lines on a video input). MadLight allows MadMapper to send pixels colors through a USB-DMX device (Enttec Pro or ShowJockey USB-DMX) or through ArtNet (the most popular DMX through ethernet protocol).
You then use an ArtNet compatible LED controller to convert DMX data to the LEDs protocol (generally WS2811 / WS2812 / APA102…). An example of reliable and cost effective controller is the PixLite from advateklights.

The new feature is good for small setups, when you just have a few strips to drive. It allows you to drive LEDs from MadMapper through the USB port with a USB-serial port device like an arduino. However, with arduino there are limitations due to the memory amount available and the CPU speed. So we actually recommand using Teensy 3.1 or 3.2 (about 20$), which is more than enough to drive a large amount of LEDs.
We provide the patch to upload in the Teensy and protocol documentation together with this document.
If you want to use it with a simple Arduino, you'll have to reduce the buffers size (so limit the number of pixels you can control).

Of course with USB cable, you cannot put the LEDs far from your computer, so driving LEDs from USB port is nice for workshops, prototyping or doing small sculptures where you can put a MacMini or let a small laptop next to your installation.

So, what are the steps to get it up and running ?

Step by step guide

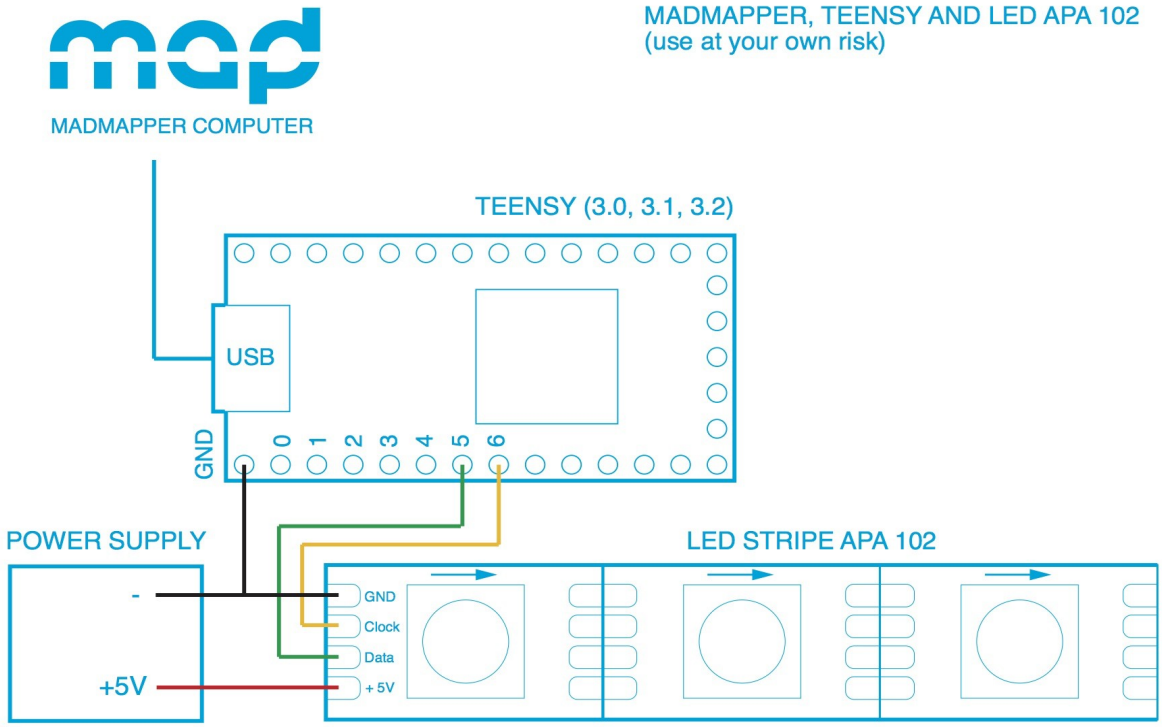
Step 1- Get all you need

- A Mac (10.7 or later) with MadMapper 2.1 and arduino studio with the add-on TeensyDuino and the library «FastLED» installed (download it here and put the decompressed folder in ~/Documents/Arduino/libraries)
- A teensy 3.1 or 3.2 (we recommand buying the teensy with pins!)
- A digital LED strip (WS2811, WS2812, APA102 are the most common ones). APA 102 has the advantage to have a clock wire, so it's more reliable for very long strips, but WS2811 / WS 2812 should be fine too (less expansive). Example : warm white APA 102, RGB APA 102
- An external alimentation if you whish to drive more than 200 LEDs (that might be less or more depending on the LEDs you use and if you put all at full white), 5V, 2A ?
- Some wires / connectors, ie http://www.adafruit.com/products/826
- You should normally have all files needed by this tutorial together with it

Step 2- Connect your LED Strip with the teensy

Your LED strip has 3 or 4 wires. When the strip has 3 wires, there are two wires for power (red & black) and a data wire (green generally). When there are 4 wires, it means the LEDs chip is using a clock, for instance APA102, LPD8806. If you have to choose a strip, prefer one with 4 wires, the clock is useful and makes controlling the strip easier from device electronic devices. When there is no clock wire, the controller must follow a strict timing or there would be glitches in the strip. I have no idea how much pixels we could control with the teensy using WS2811, or another 3 wires strip, before having glitches. From the tests I did it looked great.

Power supply : you can use the teensy power if you have less than 100 LEDs (of course it depends on your leds), after that, you should connect your strip « + » & « GND » cables to an external power supply (5V generally). In the case, the GND of the external alimentation must be connected with the GND on the Teensy.



The data pin, and the eventual clock pin, must be connected to digital pins on the teensy.

Step 3- Start Arduino application and open the file « FastLED-serial-only-single-line.ino »
Depending on the LED strip you have and the number of LEDs, you'll have to tweak the file :

In function « void setup() », you should comment the line

```
FastLED.addLeds<APA102, DATA_PIN, CLOCK_PIN, RGB, DATA_RATE_MHZ(2)>(leds, NUM_LEDS);
```

And uncomment the line that corresponds to your protocol.
Then depending on the number of LEDs and the pins you connect your LED strip to (DATA_PIN and CLOCK_PIN in case of a two wires protocol(with data and clock wires, like APA102 or LPD8886)), you will have to change the lines :

```
#define NUM_LEDS 500
#define DATA_PIN 5
#define CLOCK_PIN 6
```

Step 4- Connect you teensy and upload the patch (in Arduino app, File / Upload). Check in Arduino that there are no compilation errors and that Teensy application says « Reboot OK ».

Step 5- To test your LEDs directly withtout the need of MadMapper, add a line

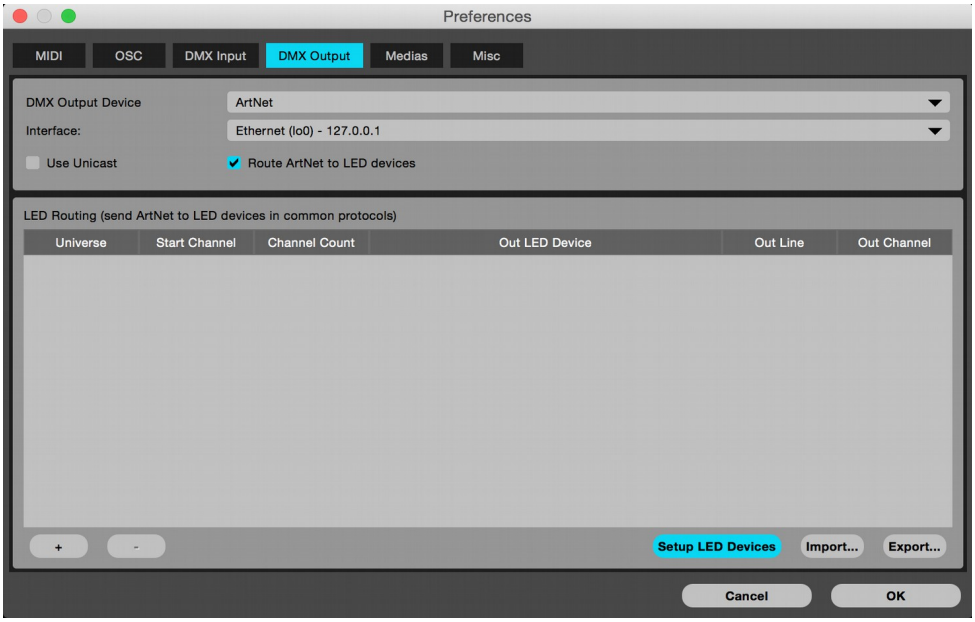
```
#define JUST_TEST_LEDS
```

at the beginning of your file. The LEDs should blink. If it blinks, remove this line and upload the sketch again. If it doesn't blink debug your setup using this simplier file: , check you followed correctly the previous lines, and try the « FastLED-simple-test.ino ».
If you can't get this to work, just your hardware connection, PIN numbers etc.

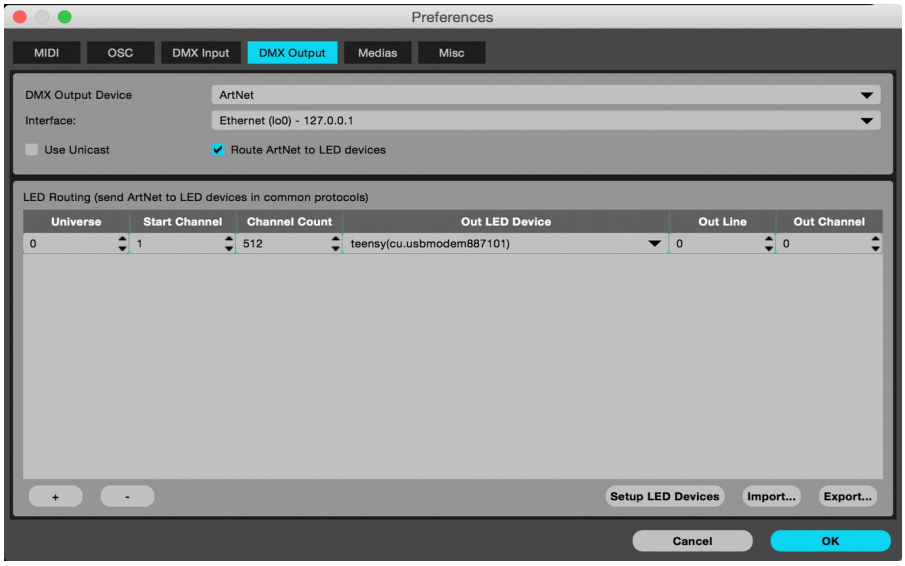
Now you have the teensy – LED setup ready.

Step 6- Setup in MadMapper preferences

Start MadMapper, open preferences / DMX Output, choose ArtNet output, network interface « localhost » (just not to pollute your wifi or LAN network with ArtNet packets), and check the option « Route ArtNet to LED devices » :

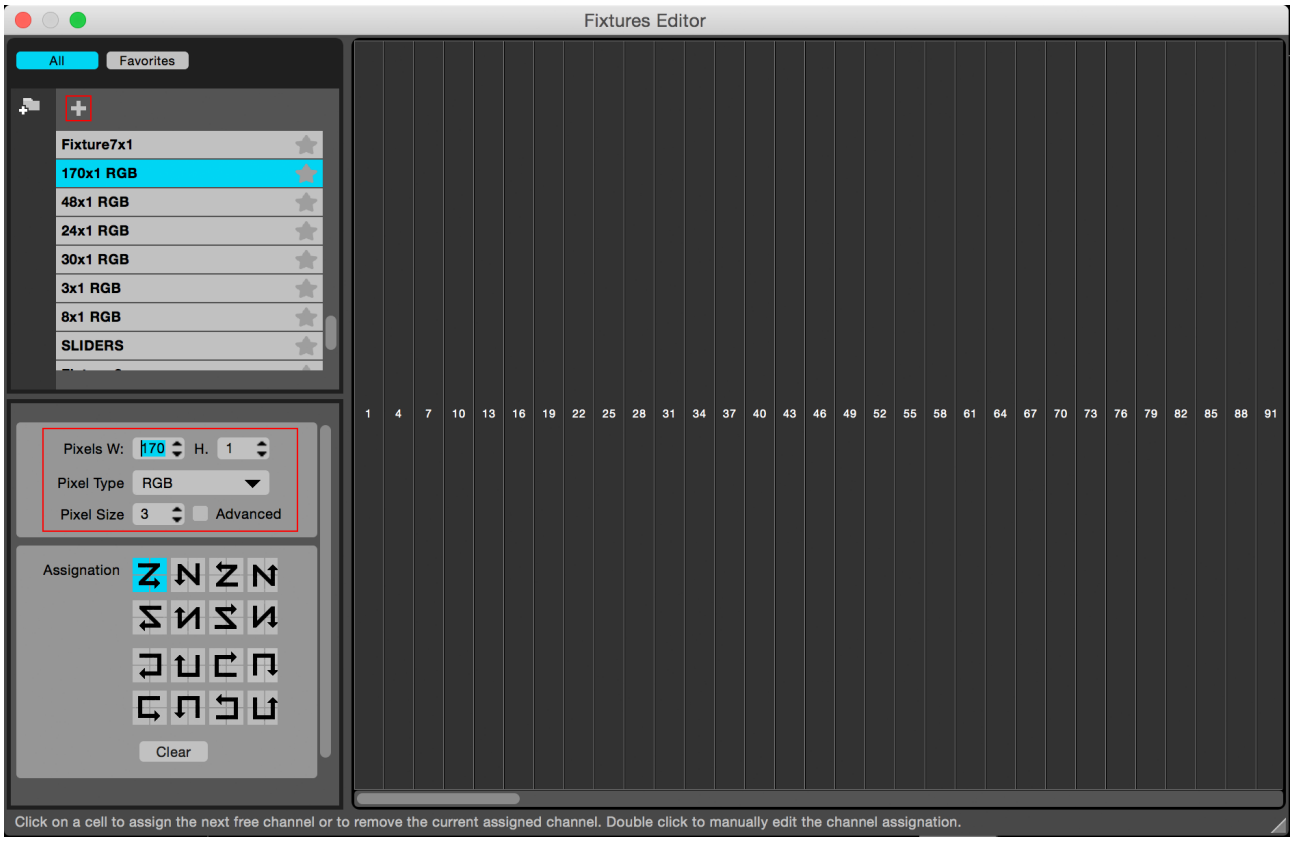


There is a table showing up below. Just press the + button on the bottom left, a new line will appear. Each line will allow sending a part of an ArtNet universe data to a LED device. Actually supported LED devices are : teensy (with MadLED protocol), arduino (with MadLED protocol) and BlinkyTile / BlinkyTape devices.
Setup your device as in the following picture:

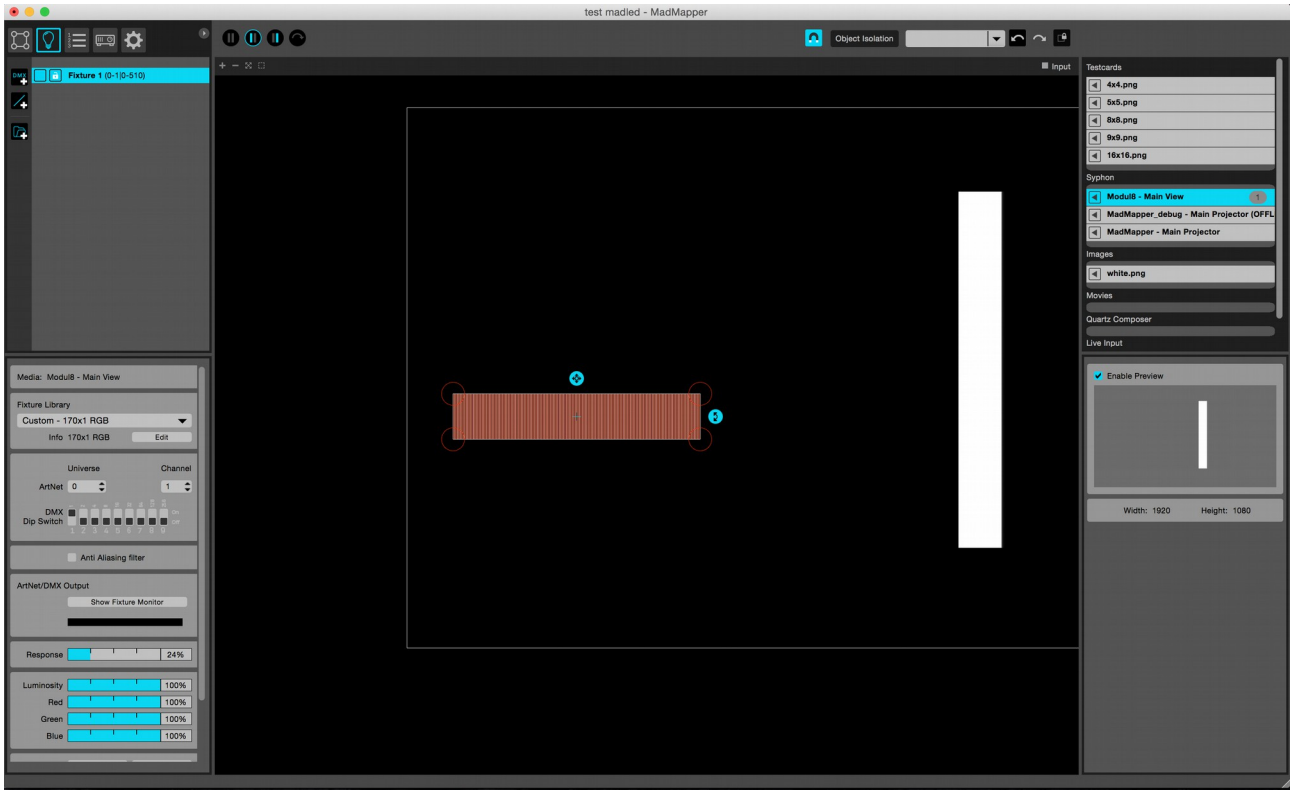


With this setup, MadMapper will forward ArtNet universe 0 data (channels 1 to 512, so the whole universe), to the teensy at channel 0. You could use ArtNet universe 1 to feed channels 512-1023 of the teensy.
The « Out Line » parameter is used if you want to control multiple strips from the same teensy (check the « FastLED-serial-only-two-lines.ino » file), but ignore that for the moment.

Step 7- Create a fixture definition for your strip from Tools / Fixture Editor. For instance for a 170 LEDs strip (a full ArtNet universe), press the « + » button and configure it as 170x1 RGB and rename it « 170x1 RGB »:



Step 8- Create a DMX fixture, choose the definition your created (170x1 RGB)



Don't change « universe » and « channel ». Universe 0 is the one sent to the teensy and channel 1 is the first channel that will control LEDs (see step 5)

Now choose a movie / syphon input to start playing video on your LEDs. You're done !

Demo Videos

https://www.youtube.com/watch?v=er_PZZwlWdM
https://www.youtube.com/watch?t=1&v=5kKivKnfhBE

In this demo we use 7 m APA102 strip 60 LEDs/meter. So 420 LEDs, each 3 channels, so it needs 420x3 = 1260 channels. Because each ArtNet universe has 512 channels, we need 3 universes. We patch 3 DMX Fixtures of 170x1 RGB pixels, on universe 0-channel 1, universe 1-channel 1, universe 2-channel 1. And in the preferences / Route ArtNet to LED devices the setup is：