Personal Rave Machine

Final project for ITM 492, Spring 2015

Rytis Bizauskas

Illinois Institute of Technology

Contact: ray@rytis.biz

Name and purpose of device

The device name is Personal Rave Machine. It’s a super easy to use music lighting device utilizing the popular WS2811 LEDs in small form factors. Traditionally, music lighting programming is a tedious task requiring large equipment, bulky controllers, and generally meant for the well technically inclined. The protocol for this control is called DMX, or Digital Multiplexer. It wasn’t really meant to be directly to hooked up to music- it’s just a protocol to control lighting fixtures. And that protocol works for large lighting applications but what about personal lighting where you perhaps still want to have the light show experience without the whole big stage setup? The Personal Rave Machine is your answer. Instead of DMX, it uses the same protocol that most digitally connected instruments and softwares already use: MIDI, or the Musical Instrument Digital Interface. What does that mean to the regular user? It means that you can create lighting shows the same way you create music on your computer. And you can choose just about any software to do so. If you already have music composed in your software, all you have to do is send the MIDI signal to the Personal Rave Machine and you’re all set. If you’d like more control over you lights, you can setup the standard Control Changes (it does the same thing as a knob and you can even automate it over time) to control how the LEDs behave and the possibilities are truly endless. Currently, only colors can be changes through these Control Changes but many more things such as fading and animation can be added later on.

Product and its operations

The Personal Rave Machine employs the use of the Teensy 3.1 microcontroller, which has the ability to emulate a class-compliant USB MIDI device. That means that it can do almost anything that a regular MIDI instrument can do. The Teensy 3.1 is similar to Arduino and in fact even works with most Arduino code. The biggest difference compared to a regular Arduino is that it has a 32-bit ARM processor (PRJC, 2014 and ARM, 2015), which allows for faster operation as well as the ability to emulate various USB devices.

In order to make the project more portable, I utilized the PJRC OctoWS2811 adapter for the Teensy, which has two standard Ethernet ports for data out to drive any type or combination of the WS2811 individually addressable LEDs. This device does not actually use any Internet capabilities; it solely provides a data and ground line to the controller. And a very nice thing about an Ethernet cable is that is has 4 data pairs, which means you have 4 data lines (data & ground) to work with one, rugged cable. Using Ethernet cable means that you can setup an Ethernet coupler on the LED side of the circuit so you effectively have two Ethernet ports and the length of the cable connecting the controller and the lighting device is essentially whatever you want it to be. So it could be the case that you perhaps want to mount the Personal Rave Machine in a far away corner on the ceiling. All you have to do is provide power to the LEDs and control the LEDs via a ridiculously long Ethernet cable, which would be connected to the Teensy wherever your computer happens to be.

Now I move on to the LEDs themselves, the WS8211 type. If you were to take the regular off the shelf LEDs with two leads and wanted to perhaps control a hundred of them, you'd need about a hundred data pins, which you don't have nor would be efficient use of processing power. In order to control multiple LEDs with a shared data line, you would need to set up multiplexing and many chips, wires, and soldering would ensure. I was originally considering that route but the WS8211 LEDs brought joy to my life. Each LED is *individually* addressable because it already has a control chip built it and they are chainable which means you can control as many pixels as you want with just one data line. The WS8211 LEDs are most popular in the strip format, so they range from anywhere to 60 to 144 LEDs per one meter long strip- all addressable via just one data line without any extra components (except a capacitor between the external power source to prevent voltage surge when the supply is powered on since the LEDs can't handle anything much over 5 volts). In my setup, I utilized a 12-count and 24-count NeoPixel rings (the Adafruit brand of WS8211s), in addition to a 144-count NeoPixel strip. I put 12-count inside the 24 count and created a nice and sturdy case for it. In order to best spread around the light, I set a water bottle right on top of the pixels which creates a very cool lighting effect. However, the LEDs are almost blindingly bright so I had to create some sort of block in front of the LEDs so you couldn't directly see the LEDs so you don't go blind. Since my case already had screws sticking out in the corners, I figured I could put some 1" foam on top of them securely. So I used a CNC milled to cut out a shape of my case with a large center hole so I can insert my bottle inside it.

The circuit for the LED rings is rather simple. First, I have an external power source jack on a breadboard, which is first directs its power to a 2200uF, 10V capacitor to smooth out voltage spikes. In order for the whole setup to work, all the grounds must meet together: so the external power source ground needs to meet the ground of the controller. To achieve that, I soldered my 2 ground lines from the controller next to the ground from the capacitor and I was set up there. In order to get my LEDs to work, I had create another wire from my joined grounds, gather the 2 data lines, as well as the 5V power line, and solder them accordingly to the NeoPixel rings. The NeoPixel rings conveniently have holes for soldering wires and I can even share the ground and power from the larger ring with the smaller ring by simple and short (no, not the electrical short) connections. See Figure 1 and Figure 2 for a visual representation of the ring setup. A similar circuit is in place for the 144-pixel strip but it is much less complex and not displayed below.

Figure 1: The LED ring circuit

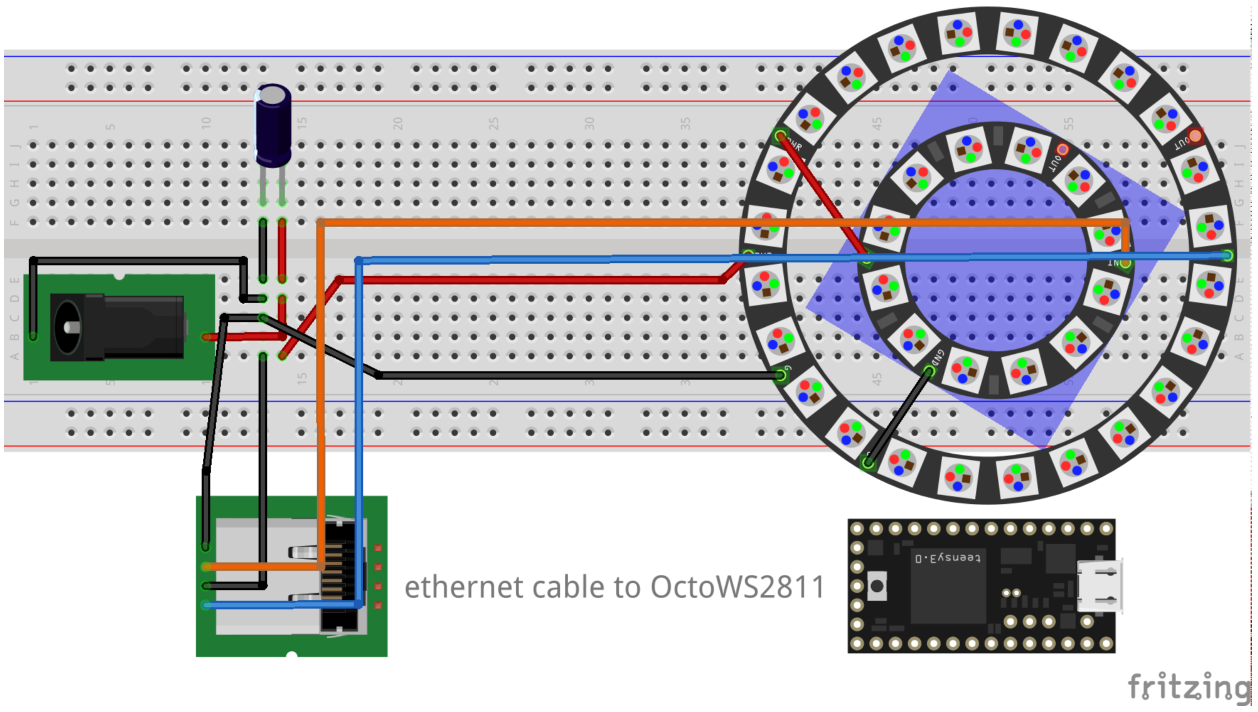
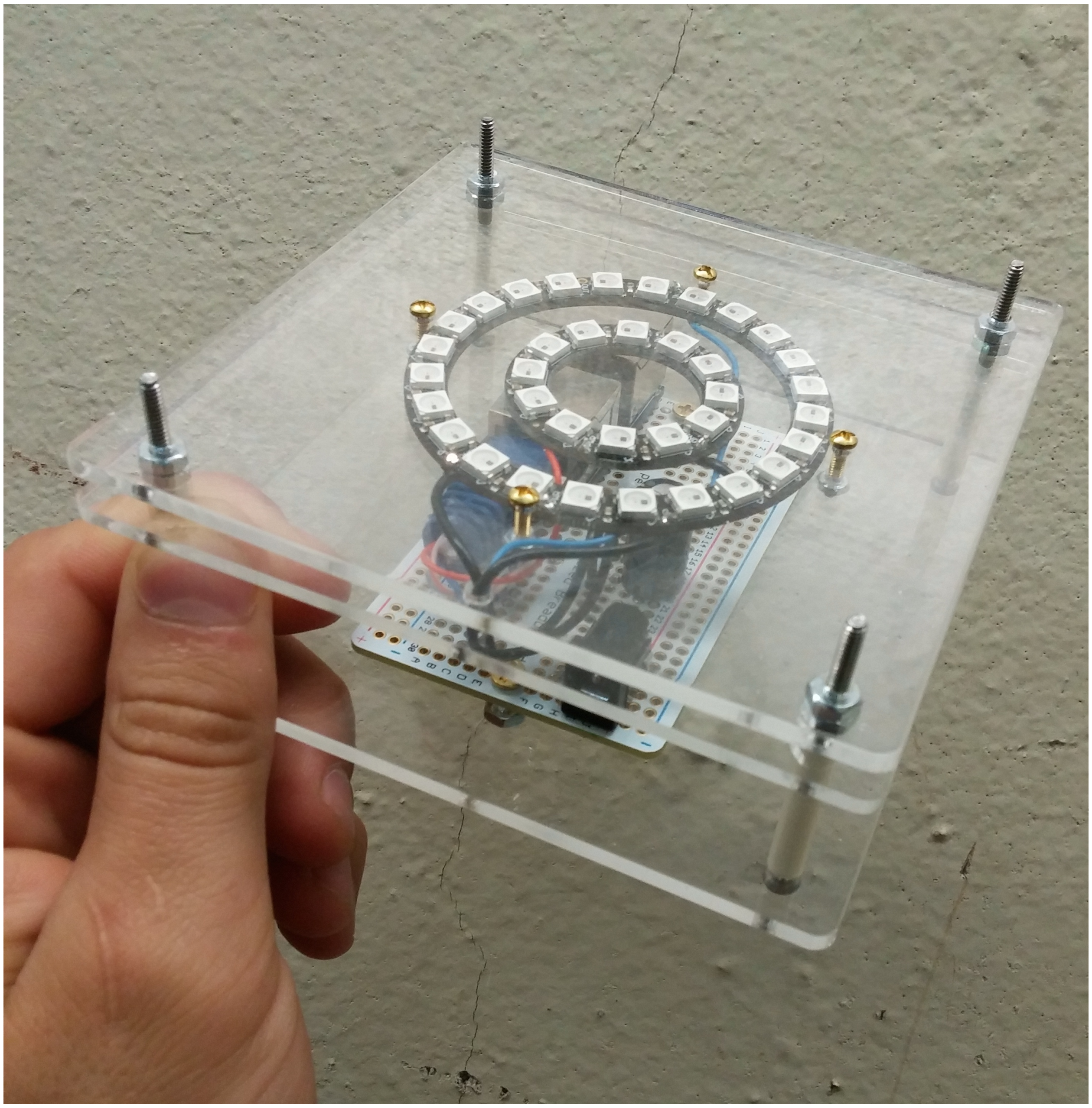


Figure 2: Picture of LED circuit + case (Teensy and power adapter not connected)



Observations

All in all, I was very happy to have the device working, accepting, and most importantly processing MIDI signal, although sometimes it would take a few tries and restarts to get everything to register with the computer.

When I started building out this device about a year ago, I originally had my LED rings directly wired to power and to an Arduino. The biggest problem for me was the wires breaking off because I was putting stress on them by carrying the whole wired mess together. I wanted to instead create a sturdy case that it is modular so no cables are sticking out nor is any stress is being applied to them everything the device is moved. I was able to create the case successfully and after a trip to Home Depot and a few trips to the IdeaShop, I was built a case composed of layered components and laser cut acrylic plates with mounting holes to make the whole device nearly indestructible. However, I did not get enough time to build such a structure for my newly acquired LED strip and as misfortune would have it, one of the ground wires snapped on the strip during my demo so I was not able to effectively convey the awesomeness of my project although afterwards after fiddling with the connection I was able to get it working again. To fix this issue, I am planning in the future to separate the strip into half-meter lengths (so two, 72 pixel strips) and attached them to a rigid surface, which will make them portable, and at the same time more rugged. I hope to be able to join those pixels in some fashion so I can have one long strip that can be quickly detached into 2 parts.

From the software side, I encountered a few issues with consistent operation but at the same time there is a lot of data flowing through different sources and it's a bit difficult to keep track of it all. One lifesaver was the Serial print line that I could perform so I knew exactly what happened in my software because I have a rather hefty amount if statements orchestrating all of the operations. For example, if I played a sequence of lights and started playing it again, the old settings would stick and the new information would be added on top of the old information, which is undesirable. The way I temporarily went around this issue was to power cycle the Teensy so the previous values are cleared. The slight problem I ran into was that I did not know when the Teensy was ready to receive MIDI signal, so it took a few tries to get it going. In addition, I often had to restart my audio engine in my music production software to get the signal to be sent properly. In the future, I am planning to demystify the startup sequence in addition to adding more functionality such as resetting the device software-side instead of removing/inserting the USB cable consistently. I noticed that the Teensy 3.1 has a reset solder pad so I may be able to programmatically reset my Teensy.

Future work

It is my goal to continue this project and hopefully create something that I can sell. From the technical standpoint, I will have employ some advanced computer programming techniques to create all the effects that I want to with the LEDs. In addition, I hope to source a low-cost supplier of parts because my prototypes currently run at least a $100 in costs alone. But sourcing parts themselves won't be enough: I will have to actually create a custom circuit board with my controller and all other components professionally designed and made as my lab prototype version is certainly not something a serious consumer would buy, nor would I like to sell myself as picking it away would be a bit too easy.

References

Cortex-M4 Processor. (n.d.). Retrieved May 5, 2015, from http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php

Teensy 3.1 - New Features. (2015). Retrieved May 5, 2015, from https://www.pjrc.com/teensy/teensy31.html