The code for the Brownsburg High School Winter Percussion 2023 props is built for an ESP32 Dev Module. There are various sources of information for how to add support for the ESP32 to the Arduino environment and they generally involve adding <https://dl.espressif.com/dl/package_esp32_index.json> to the board manager.

The transmitter makes use of the MIDI Library by Francois Best version 5.0.2.

Both the transmitter and receiver rely on FastLED by Daniel Garcia version 3.3.3. There are newer versions of this library available but I found that there is a bug in them that affects the ability to transmit to more than four LED strips simultaneously. The ESP32 has the capability to control up to eight strips at the same time, but the newer version of the FastLED library will only transmit up to four at time. If you try more than four like the props have, it will do four then the remaining one sequentially effectively doubling the total transmission time and slowing down the frame rate. A slower frame rate can lead to less smooth transitions and effects.

Here is the math: each LED on the strip takes 24 bits of color information at 800 KHz. With 300 LEDs that results in 9 msec of transmission time. With the buggy version it would double to 18 msec. Since the number of LEDs was increased to 600 for more brightness, the buggy version would be 36 msec, but the 3.3.3 version would only be 18 msec. I’m being a little nit-picky and the slower version would probably still work, but it would bug me that the ESP32 has a capability that we couldn’t access with the new version of the library. Also a frame rate of 36 msec starts to get into the noticeable region.

There are a few different approaches to driving multiple strips with FastLED and the examples that come with the Arduino library are very good. The prop code is currently using the same memory buffer for each of the five strips to keep things simple. This means that each of the five segments of the props will be the same as each other. If we ever want to do some sort of chasing or flashing effect where segments of the prop are different from each other, different memory buffers will be needed for each segment. It will also take some processing time to fill each buffer each time the frame is updated. The FastLED library is built for speed but the time should be measured if we go this route.

The code makes use of “dynamic gradients” for defining colors. They aren’t really needed for solid color effects, but could be a powerful tool for more complex color designs. See <https://cssgradient.io/> for a tool used to create color gradients. Here is an example that has mostly red with a light band followed by a dark band which slowly goes back to red. On the screen you see it spatially but on the LEDs, a single region can cycle through the gradient and when it transitions from light to dark quickly, it is a twinkle effect. This is how the effect labelled “CHUNKY” works. It is based on example code but I modified it to break up the strip in to chunks of ten LEDs that were seeded with a random number then each chunk cycles through the gradient. It may need to be modified to produce a more dramatic effect. Also note that the cssgradient tool uses a scale from 0 to 100 but FastLED uses 0 to 255 so the numbers need to be scaled up when they are put into the code. The structure for sending commands from the transmitter to the receiver limits the gradient size to be six points but could be increased if needed.

