Hacking the DFW Hackers Badge

This hack has been brought to you by **Gregory A. Miller** of **TheLab.ms**

Acknowledgement

I first wish to acknowledge the work of Richard Gowen and all the excellent work and support he has selflessly provided in the form of promotion, mentorship and momentum boosts for the fine organizations so named on the face of the DFW Hackers Badge. TheLab.ms is but one of the organizations in which he has donated time and money to and I think I speak for all membership of TheLab.ms, we appreciate everything that he contributes to its continued success.

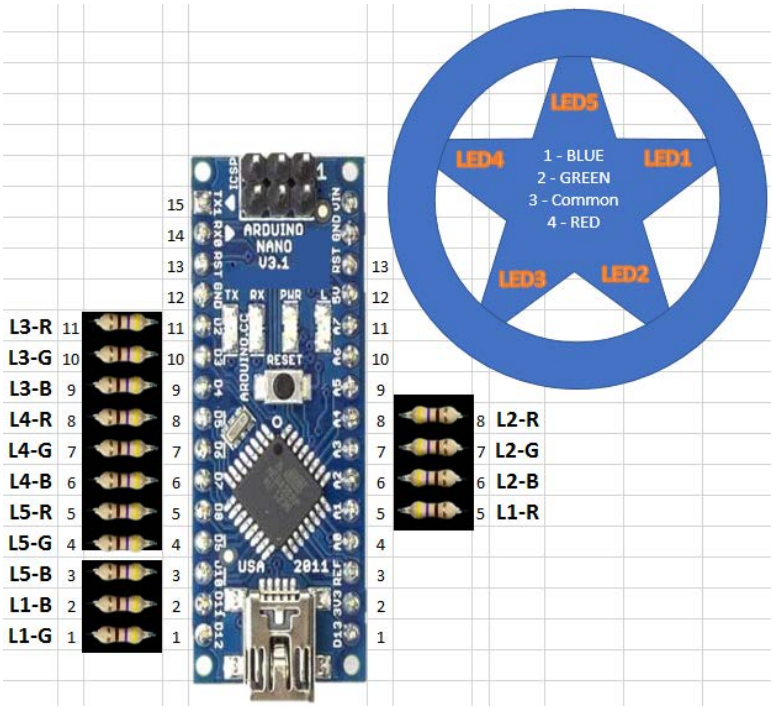
Objective

There are four objectives:

1. Use a larger smart LED for a bolder appearance and more visual options.
2. Be able to reverse configure the badge using the original RGB LEDs and quickly switch between the two configurations without the need for soldering.
3. Have a single code base for all the above.
4. Everything should be reusable. No major components are permanent or altered.

Engineering Considerations

The first step was to Identify the pins used on the Arduino and determine if the redesign was even possible. The leads were traced from the RGB LEDs back to each corresponding pin on the Arduino and created a diagram of the layout to reference during future revisions.



The project calls for a common cathode (common ground) RGB LED device. For the ease of identifying the pins the LEDs were numbered on the star pattern as shown above. The LED connections to the PCB are numbered from 1-4 to support the following functions:

1. BLUE
2. GREEN
3. GROUND
4. RED

Thus, as you read the labels surrounding the Arduino image, L1-R is LED #1 and the RED lead, Pin #4, where L2-B is LED #2 and the BLUE lead, Pin #1.

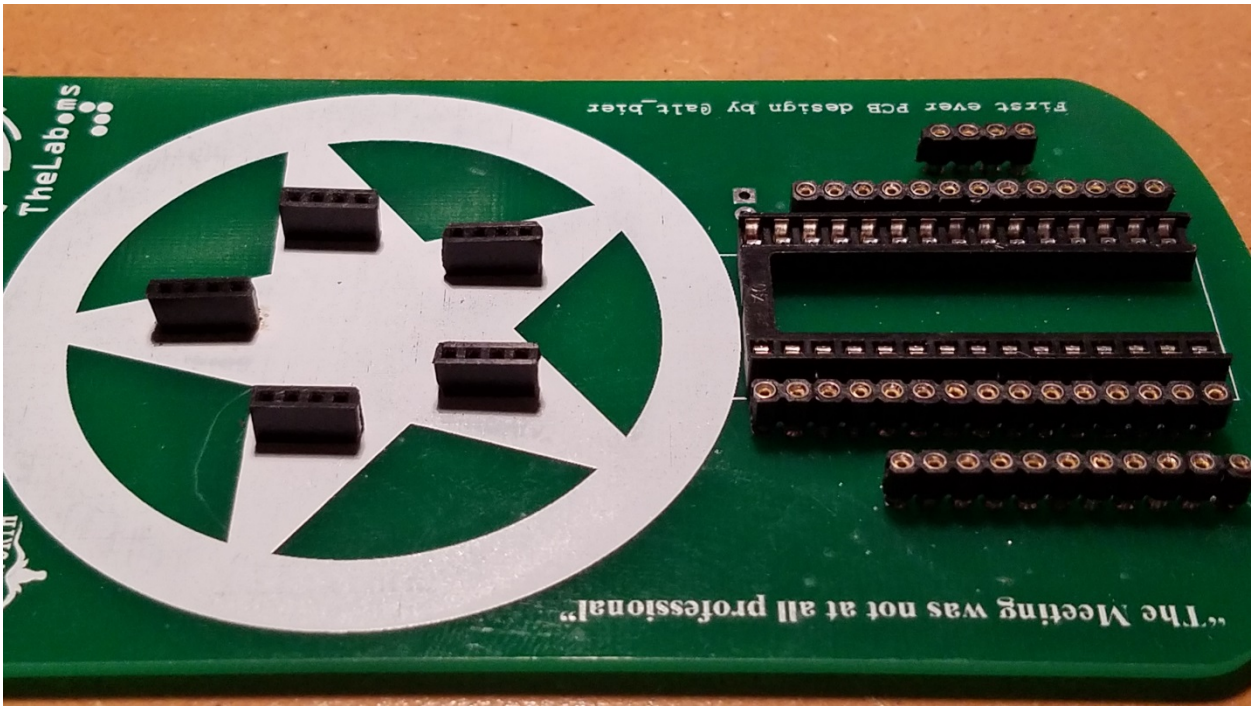
These labels are not useful in the original configuration build as all the hard work was completed by Richard and the only thing necessary to do was plug it in and turn it on… All of this pre-work came in handy on the redesign process to help identify how the LEDs would be tied together.

Build Process

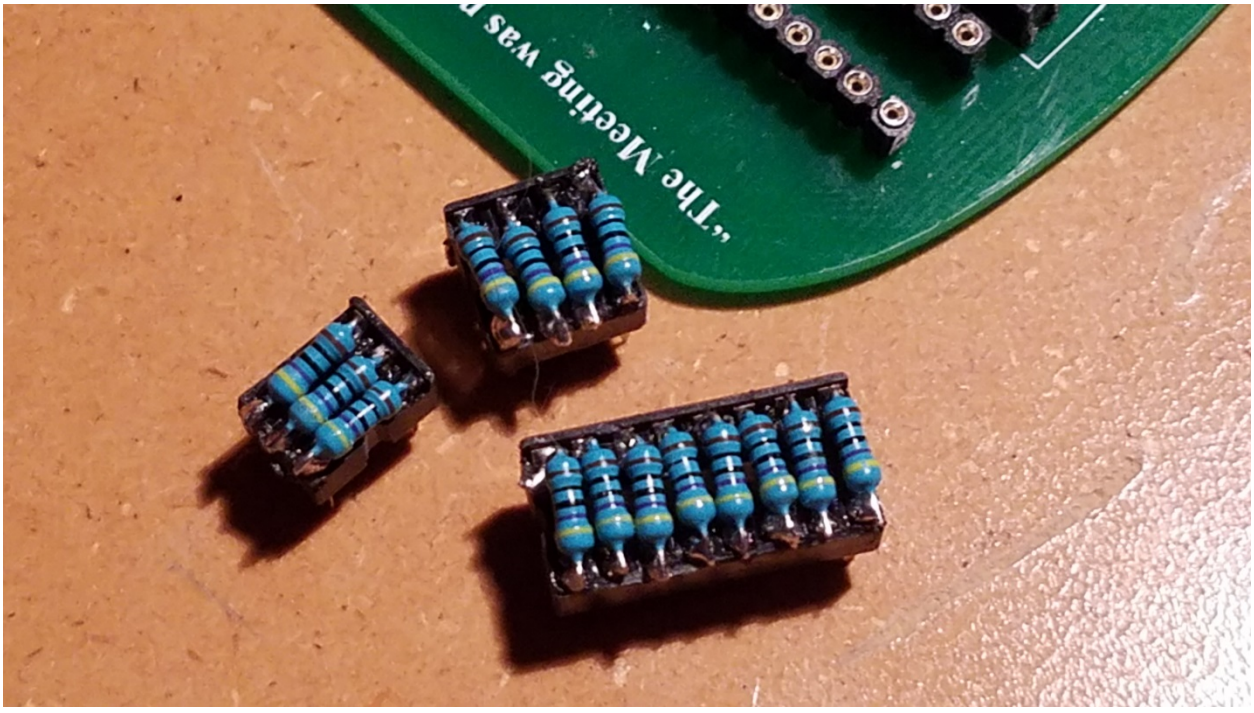
Nearly Original Build

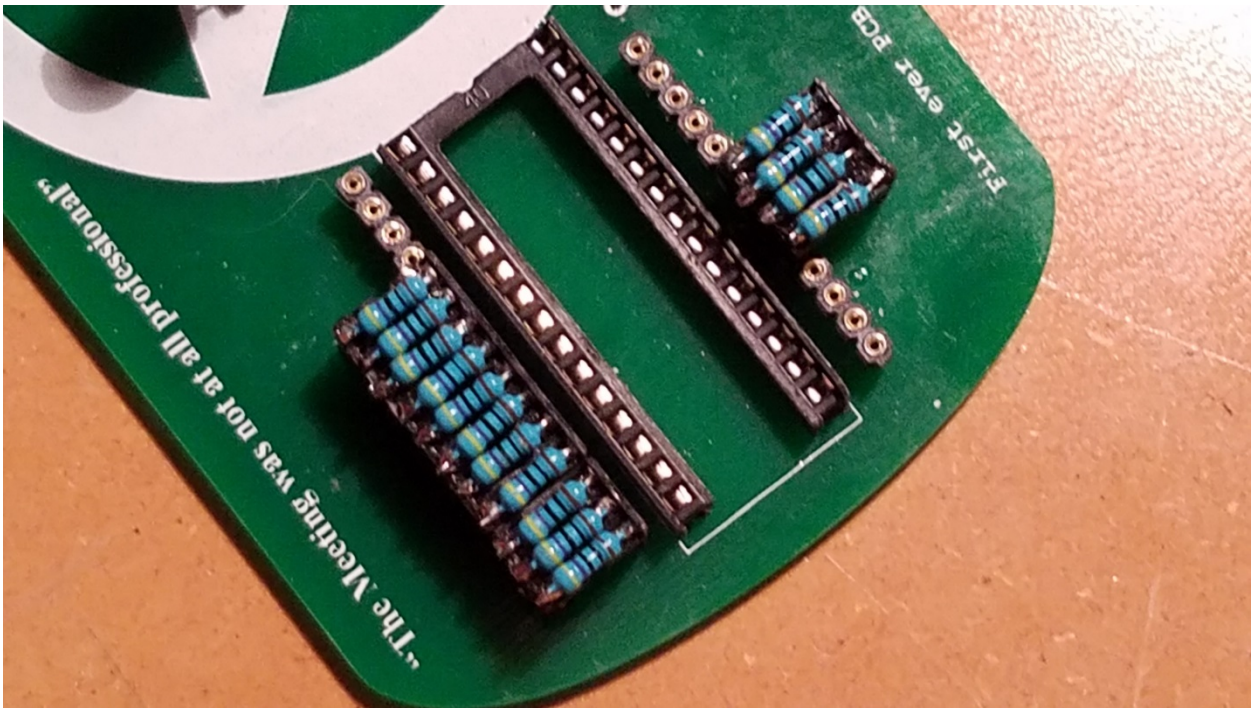
In the above drawing, you see three black rectangles surrounding some resistors. These black rectangles represent sockets that hold resistors which are needed in the build. This was done to stay true to the list of objectives. In fact, everything was socketed in such a way the devices on the PCB could be quickly and easily removed and replaced if needed.

The Arduino has an odd number of pins so a slightly larger socket was custom cut and adapted for use. A standard 2.54mm pitch pin header strip was cut to length to fit the holes on both sides of the Arduino. The LED sockets began life as a 40 pin, 2mm pitch, header strip which was cut down to size and edges cleaned up before soldered into place.

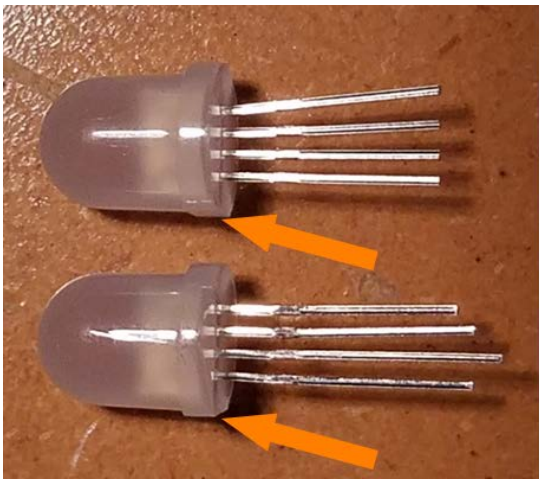


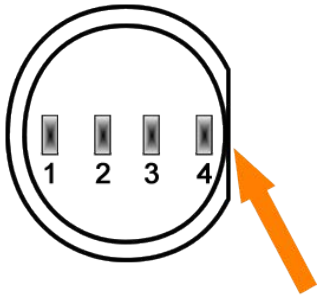
Matching sockets were fitted with the necessary resistors as called out by the original design. The resistors are not going to be reusable for anything other than this project but since resistors are less than a penny each in bulk, this small loss of reusability was acceptable overall.





The RGB LEDs come with staggered leads which need to be all cut to a similar length so they can be inserted into the socket in a uniform fashion. Don't cut too short it makes it hard to get them inserted but short enough so they are not getting their pins bent when bumped into. Approximately 0.5" long worked well. Also keep in mind the LEDs are polarized and need to maintain a specific orientation. The flat edge is how to know which direction to plug them in, after the leads are cut. The flat edge will always be oriented towards the bottom.

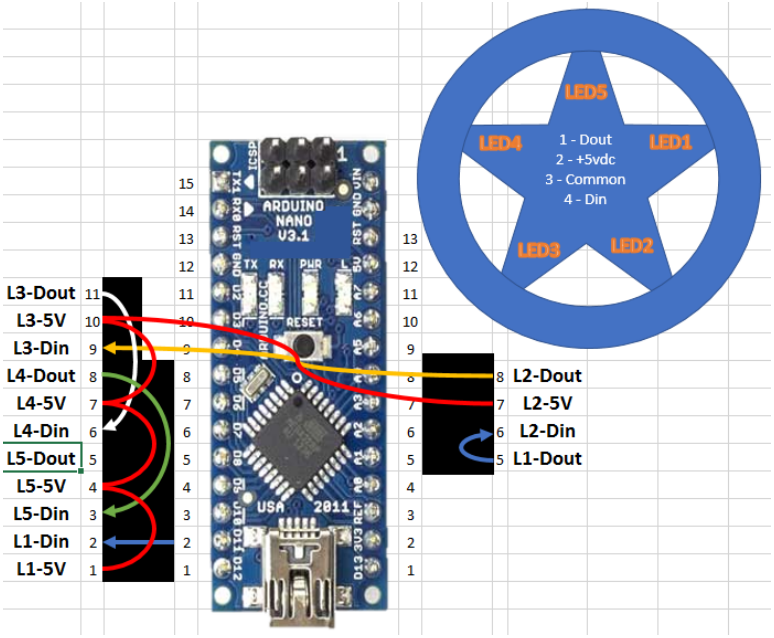




Everything was then plugged in according to the original project documentation and away it went. Now let's improve on what we have.

Next Generation Build

After tracing the wiring connections in the previous build, it was determined that this project could be rewired to change its personality in a modular fashion thus keeping with the original objectives. Here is the revised wiring diagram:



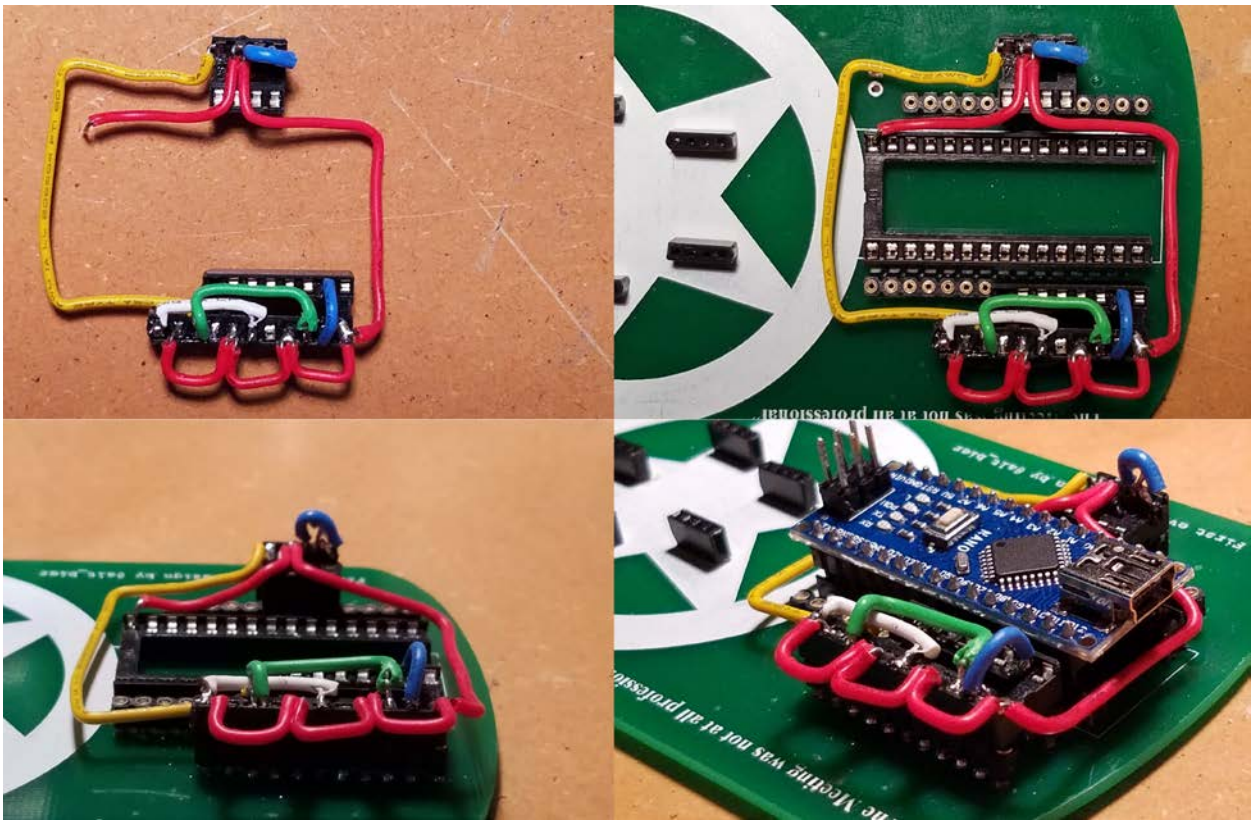
PL9823 Pinouts

1. Data Out
2. +5vdc Vcc
3. GROUND
4. Data In

While this device looks like an RGB LED, it is wired and controlled very differently.

Since the PL9823 LEDs operate on a PWM serial link principle, the data in and data out pins were identified and the resistors, in the original build, were replaced with jumper wires soldered into sockets so the data paths retain my numbering definitions of LED1 → LED5

In the photo above, you can see there is an arrow on the data lines which notate the direction the serial data is flowing. I built the wiring modules (shown below) using a wire color to match.



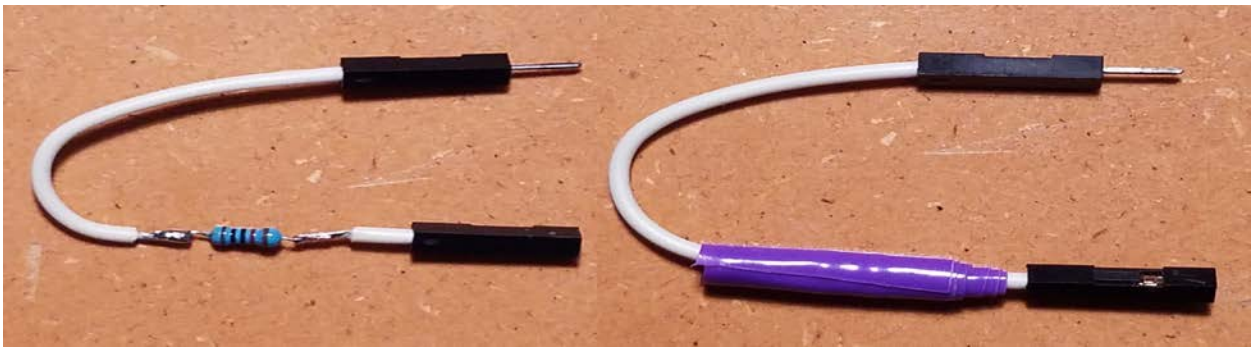
The PL9832 LEDs work similarly to the popular Adafruit NeoPixel and WS2812 devices but at a significantly lower per unit cost, at least lower at the time of purchase.

Software Design

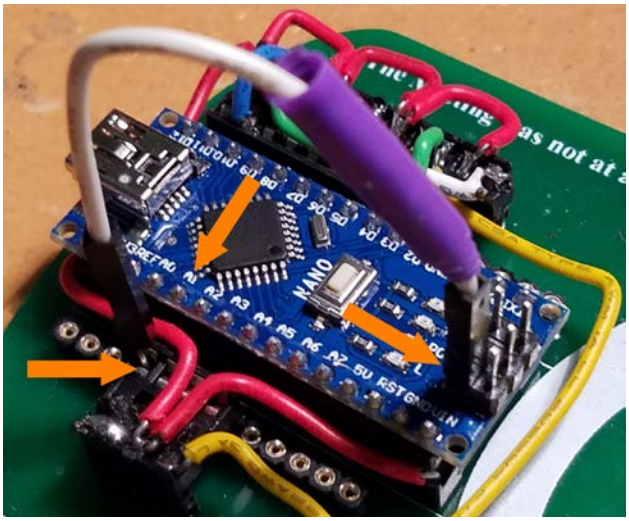
The goals behind the software was to have the ability to have a single codebase to drive the original RGB LEDs and the LP9823 devices. This would enable the change of desired functionality without the need to have a computer handy to reprogram the Arduino Nano.

The goal is to use the unused GPIO as a selector switch of sorts. Reading about current draw on the Arduino pins, it is determined that a direct link from a GPIO to ground or power would create a direct short and potentially zap the Arduino device nearly immediately. Shorting two GPIO pins together could also, in certain circumstances, cause the same short condition. This means a resistor must be used.

A safe way to do this is to build a custom M-F jumper wire that has a 10Kohm resister soldered in line with some tape or shrink tubing to protect it. This provides an easy way to choose between one of eight analog pins on the Nano without making any permanent modifications to the board.



The customized jumper is used to make a demo selection by plugging the female end onto the Vcc pin of the ICSP port on the Nano and the other end into any of the available Analog pins exposed. The image below shows it plugged into the Analog A1 pin of the socket.



This connection only needs to be made when power is applied or when a reset is performed and then can be removed if desired. The code is designed to read the selection as it is setting everything up and then it does not read it again unless the Arduino is reset or power cycled.

***** CAUTION :: This is a Live Vcc connection so be careful not to short anything out. *****

Code Section actually starts here…

The Adafruit NeoPixel libraries were used in this demo code for simple implementation but you can use whatever libraries you want or simply write your own. I also started with the Example code named “strandtest”, which comes with the Adafruit NeoPixel library, as a foundation for creating several fast demos for this project. This example code is designed for use with pre-cut lengths of the NeoPixel strips but is easily modified for this application.

The hack project only has 5 PL9823 devices to work with so we do not have much to play with as far as eye candy demos are concerned (see Final Thoughts section) but here is how our code and demos work:

First step is to add the Adafruit NeoPixel libraries to your code:

```
// This sketch requires the Adafruit NeoPixel Library v1.1.6 or higher
#include <Adafruit_NeoPixel.h>
```

```
// Depending on your device model and Mfr. the parameter in the following
line can be either NEO_GRB or NEO_RGB.
// If the colors are all wrong, try switching this parameter. All it does is
change the bit order on the colors output.
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(DEVICECOUNT, PIN, NEO_RGB +
NEO_KHZ800);
```

The remaining items are to define the pin to be used for the PWM output signal. It seem that a variable the number of device is silly but you can expand this from 5 to several hundred if desired.

```
// Which Arduino pin is the PWM Output? Nano options include 3,5,6,9,10,11
// PIN 5 was chosen by default to match the Hacked schematic.
#define PIN             11
// This defines how many LED devices you have on the string.
#define DEVICECOUNT     5
// If you experience flicker or random flashes of color, increase this delay
to make it go away.
int iDelay_ms = 500;   // Time delay in miliseconds
```

There were minor tweaks to the original RGB code so it would be able to coexist with the Hack code. I only took out the original Setup() function and replaced the original Loop() function with a while() loop. The whole thing is then wrapped up in a function called OriginalBadgeCode(). In the following code blocks the **RED** code was removed, **PURPLE** was modified.

```
void OriginalBadgeCode() {

//void setup()    // Original Setup function was commented out. The remaining
//{               // initialization of vars was merged into calling function.
                  // Reminder of original code was unchanged.
…
Original Setup() code goes here in its entirety
…
//}
```

```
// Main Loop      // Original Main function was commented out. The remaining
//void loop()     // code of the loop was placed into a never ending while()
                  // to emulate the original Main loop.
while(1)
{
…
Original Setup() code goes here in its entirety
…
}
}
```

The new Setup and Main functions control the actions of the Hacked Badge. In Setup there is a call to a function called ConfigurationSet() which determines which personality the badge should be operating as:

```
void ConfigurationSet() {
// Determine if the RGB configuration is active or the Hacked configuration.
      for(int a=0;a<8;a++){
      // The Nano has 8 analog pins which can be used for selection testing.
      // Test each one and find one that has an output value of "1023"
            Serial.println(analogRead(a));
            if(analogRead(a)==1023){
                  iSelected = a;
                  break;
            }else {
                  iSelected = 99;
            }
      }
}
```

Based on the value placed into the iSelected variable determines which personality the Main loop will begin executing:

```
Void loop() {
  switch(iSelected) {
    case 0:
      demoBasic();
      break;
    case 1:
      // This demos two different techniques for fading
      demoFade1(5);
      demoFade2(5);
      break;
    case 2:
      demoSpinningWheel();
      break;
    case 3:
      demoRandomBlinky(500);
      break;
    case 4:
      demoPatriotic();
```

```
      break;
    case 5:
      demoFunColors(1000);
      break;
    case 6:
      demoRandomColors(100);
      break;
    case 7:
      // This selection runs all the demos in order
      demoBasic();
      demoFade1(5);
      demoFade2(5);
      demoRandomBlinky(500);
      demoPatriotic();
      demoFunColors(1000);
      demoRandomColors(100);
      break;
    default:
      // This is only run if the badge is in RGB mode
      Serial.println("RGB Configuration");
      OriginalBadgeCode();
      break;
  }
}
```

Final Thoughts

You can easily extend this project with more PL9823 devices by using the L5-Dout pin to another Din on a 6th device or an entire string of devices. The Nano should be able to drive several hundred devices, given enough power to light them all.

This was an incredibly fun project. There are so many things that can be done to the badge beyond blinking lights and we will be exploring some additional ideas soon.

Pitfalls & Lessons Learned

1. When soldering your socket to the PCB, use only enough solder to attach the socket to the via. The sockets I used had an open bottom so I had the unfavorable issue if solder filling the socket and thus decreasing the grip effect and depth of insertion. This caused my LEDs to not make a good circuit connection after getting jostled around a bit.
2. Use solid core wire so its rigid properties can be exploited in keeping the sockets in place during use.
3. The Arduino Nano cannot drive too much current and will fail catastrophically.
4. If odd and unexplainable things are happening, check for shorts and open connections.
5. The LEDs do not keep good connection in the sockets.
6. Do NOT tie the GPIO directly to Vcc or GND as this will cause a direct short and can cause the release of the magic smoke.
7. Batteries can get short circuited if not handled properly. =<

References

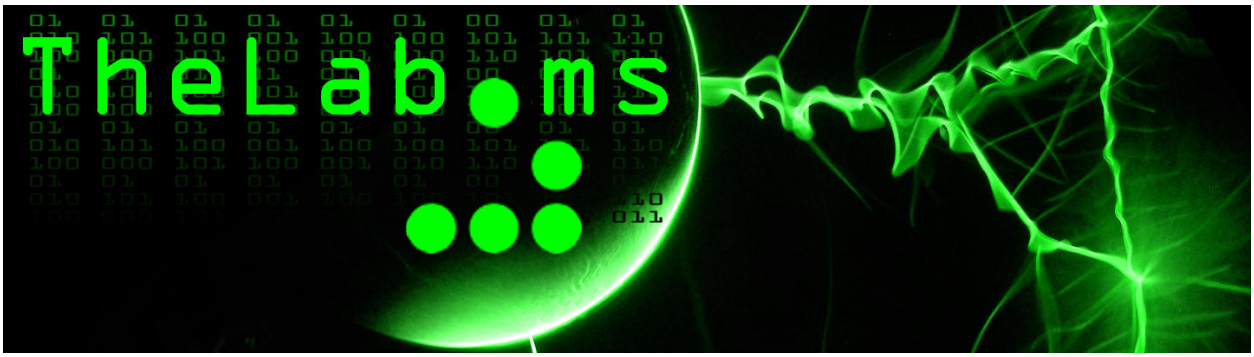
Adafruit - NeoPixel Diffused 8mm Through-Hole LED

NeoPixel is a trademark of Adafruit. You can get these directly from Adafruit as well as some distributors stock them as well.

Amazon.com - 8mm WS2812 Addressable LED – RGB

Amazon.com – 8mm PL9823 Addressable LED – RGB

I provide a link to Amazon search for the WS2812 and PL9823 devices but there are many places these can be sourced from and if you can wait you can get them directly from China really cheap.



TheLab.ms is an all-volunteer, 501(c)(3) non-profit organization and community resource. For more information, please visit our website and consider becoming a supporting member.