<table>
<thead>
<tr>
<th>Guide Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td>Prototype &amp; Code</td>
<td>5</td>
</tr>
<tr>
<td>Circuit Diagram</td>
<td>10</td>
</tr>
<tr>
<td>Solder LED Strips</td>
<td>11</td>
</tr>
<tr>
<td>Solder Pro Trinket &amp; Keypad</td>
<td>14</td>
</tr>
<tr>
<td>Fabric Backing</td>
<td>19</td>
</tr>
<tr>
<td>Use it!</td>
<td>23</td>
</tr>
</tbody>
</table>
Overview

Using DotStar LED strips we can create a flexible, portable light source for video and photography. It can squish into small spaces, easily hang up on location, and it's fun and simple to make.

Different lighting situations call for a different color temperature of light, so we'll combine warm and cool white LED strips and program a Pro Trinket to adjust the light with a four-button membrane keypad.

We saw a crowdfunding campaign (https://adafru.it/fgi) a while back for something similar, and when it was canceled we still wanted one. Now that we have these DotStar LED strips in the shop we can build it together.

Before you begin, read/watch the following guides:

- Introducing Pro Trinket (https://adafru.it/nXF)
- Adafruit DotStar LEDs (https://adafru.it/zbD)
- Battery Powering your Wearable Electronics (https://adafru.it/e4c)

To build this project, you will need:

- Pro Trinket 5V (https://adafru.it/2000)
- Adafruit DotStar LED strip in warm and cool white (https://adafru.it/fgf) (1m each)
- Membrane keypad (https://adafru.it/elg)
- 4400mAh USB battery pack (https://adafru.it/dgz)
- Silicone coated 26ga stranded wire (https://adafru.it/dVt) in four colors
- Soldering iron and supplies (https://adafru.it/drl)
- Premium female jumper wires (https://adafru.it/fgg)
- Heat shrink tubing (https://adafru.it/erk)
- Silicone adhesive such as Permatex 66B (https://adafru.it/fgh)
- Durable fabric
- Velcro tape
- Sewing machine
- D-rings or clasps to make attachment points (optional)
- Scissors
- Needle & thread
This sketch shows the rough alignment of components for the project. Alternating strips of warm and cool white DotStar LEDs chain together to form a bank of light. For more detailed wiring instruction, visit the circuit diagram on the next page.
It pays to prototype your circuit on a solderless breadboard before soldering everything together. If you can manage to have a duplicate set of parts, that's even better-- you can use your working prototype as a model to work from when building the final soldered circuit.

Below is some rudimentary code for adjusting the brightness of the two LED strips using the membrane keypad as follows:

1: strip 1 brightness up
2: strip 1 brightness down
3: strip 2 brightness up
4: strip 2 brightness down

The brightness value incrementers don't have any thresholding, so values will "wrap around" (ie pressing 2 when the strips are off will raise that strip to full brightness).

Load the following code onto your Pro Trinket:

```cpp
#include <Adafruit_DotStar.h>
// Because conditional #includes don't work w/Arduino sketches...
#include <SPI.h>  // COMMENT OUT THIS LINE FOR GEMMA OR TRINKET
///#include <avr/power.h>  // ENABLE THIS LINE FOR GEMMA OR TRINKET

#define NUMwarmPIXELS 60  // Number of LEDs in strip
#define NUMcoolPIXELS 60  // Number of LEDs in strip
#define DATAPINwarm 6
#define CLOCKPINwarm 8
#define DATAPINcool 3
#define CLOCKPINcool 4
Adafruit_DotStar warmStrip = Adafruit_DotStar(NUMwarmPIXELS, DATAPINwarm, CLOCKPINwarm);
Adafruit_DotStar coolStrip = Adafruit_DotStar(NUMcoolPIXELS, DATAPINcool, CLOCKPINcool);
```
Adafruit_DotStar coolStrip = Adafruit_DotStar(NUMcoolPIXELS, DATAPINcool, CLOCKPINcool);

#define DEBOUNCE 10 // button debouncer, how many ms to debounce, 5+ ms is usually plenty

// here is where we define the buttons that we'll use. button "1" is the first, button "6" is the 6th, etc
byte buttons[] = {9, 10, 11, 12}; // the analog 0-5 pins are also known as 14-19
// This handy macro lets us determine how big the array up above is, by checking the size
#define NUMBUTTONS sizeof(buttons)
// we will track if a button is just pressed, just released, or 'currently pressed'
byte pressed[NUMBUTTONS], justpressed[NUMBUTTONS], justreleased[NUMBUTTONS];

int warmBrightness = 0;
int coolBrightness = 0;

void setup() {
  byte i;

  // Make input & enable pull-up resistors on switch pins
  for (i=0; i<NUMBUTTONS; i++){
    pinMode(buttons[i], INPUT_PULLUP);
  }

  // pin13 LED
  pinMode(13, OUTPUT);
  warmStrip.begin(); // Initialize pins for output
  warmStrip.show();  // Turn all LEDs off ASAP
  coolStrip.begin(); // Initialize pins for output
  coolStrip.show();  // Turn all LEDs off ASAP
  fillAll(warmStrip.Color(255, 255, 255));
  warmStrip.show();
  coolStrip.show();
}

void loop() {
  warmStrip.setBrightness(warmBrightness);
  coolStrip.setBrightness(coolBrightness);
  warmStrip.show();
  coolStrip.show();
  digitalWrite(13, LOW);

  check_switches();      // when we check the switches we'll get the current state

  for (byte i = 0; i<NUMBUTTONS; i++){
    if (pressed[i]) {
      digitalWrite(13, HIGH);
      // is the button pressed down at this moment
    }
    if (justreleased[i]) {
      if (i == 0){
        warmBrightness+=5;
      }else if (i == 1){
        warmBrightness-=5;
      }else if (i == 2){
        coolBrightness+=5;
      }else if (i == 3){
        coolBrightness-=5;
      }
    }
  }
}
for (byte i=0; i<NUMBUTTONS; i++){ // remember, check_switches() will necessitate clearing the 'just pressed' flag
    justpressed[i] = 0;
}

void check_switches()
{
    static byte previousstate[NUMBUTTONS];
    static byte currentstate[NUMBUTTONS];
    static long lasttime;
    byte index;

    if (millis() < lasttime){ // we wrapped around, lets just try again
        lasttime = millis();
    }

    if (((lasttime + DEBOUNCE) > millis()) {
        // not enough time has passed to debounce
        return;
    }

    // ok we have waited DEBOUNCE milliseconds, lets reset the timer
    lasttime = millis();

    for (index = 0; index<NUMBUTTONS; index++){
        // when we start, we clear out the "just" indicators
        justreleased[index] = 0;
        currentstate[index] = digitalRead(buttons[index]);   // read the button

        if (currentstate[index] == previousstate[index]) {
            if ((pressed[index] == LOW) && (currentstate[index] == LOW)) {
                // just pressed
                justpressed[index] = 1;
            }
            else if ((pressed[index] == HIGH) && (currentstate[index] == HIGH)) {
                // just released
                justreleased[index] = 1;
            }
        }
        pressed[index] = !currentstate[index];  // remember, digital HIGH means NOT pressed
    }

    //Serial.println(pressed[index], DEC); // keep a running tally of the buttons
    previousstate[index] = currentstate[index];
}

void fillAll(uint32_t c)
{
    for(uint16_t i=0; i<warmStrip.numPixels(); i++) {
        warmStrip.setPixelColor(i, c);
    }

    for(uint16_t i=0; i<coolStrip.numPixels(); i++) {
        coolStrip.setPixelColor(i, c);
    }
    warmStrip.show();
    coolStrip.show();
}
DotStar strips connected to pins 3 and 4 and 6 and 8. Membrane keypad buttons connected to pins 12, 11, 10, and 9. Battery connected via USB port.
Solder LED Strips

Cut your LED strips to your desired length and configuration. Our light is comprised of 3 20-LED sections of each color temperature strip. Be sure to preserve solder pads on each end of the strip.

If using the super dense 144/m strip, you may have a "waste pixel" between sections in order to get as much available solder pad as possible.
Strip and tin four different colored stranded wires. Tin the pads of the DotStar strip.

Solder every other wire on alternating sides of the LED strip to help avoid shorts.
Carefully trim the wires to length so that they make tidy junctions to the next strip. Remember to keep all the data arrows flowing in the same direction!

Your LED circuit will now look something like this. Test it with your solderless breadboard circuit before proceeding.
Solder Pro Trinket & Keypad

Solder data connections to Pro Trinket according to the circuit diagram.
Twist and tin the two power wires together and slide on a piece of heat shrink tubing. Solder to BUS on Pro Trinket, then slide and shrink the heat shrink over any remaining exposed wire.

Double check there are no stray strands of wire, or these could short out and possibly causing damage!
Repeat with the ground wires, but add one more wire to the bundle that will go to the membrane keypad's ground wire.

Plug your membrane keypad into some female jumper wires and lay out the assembly to gauge the appropriate length to cut the jumpers.
Solder the jumpers to digital inputs and ground as described in the circuit diagram. Use heat shrink tubing where applicable.

Test out your circuit to make sure everything works as expected before proceeding!
Cut two pieces of durable fabric to the size of your matrix plus an inch in each direction for seam allowance. With right sides together, stitch lines as shown using a sewing machine (or by hand if you must), then chop off the corners so everything will sit nice when you turn it right side out:

Iron the rectangle if it isn't sitting flat, then top stitch as shown below to make battery pouches:
You can optionally add D rings or clasps at this step to make attachment points for your light. Stitch small bits of fabric around them and include them in your topstitched seam, plus a little X for added support.

Use Velcro tape at the openings of the battery compartments.
When you're finished, the fabric backing will appear as below:

Sew your circuit elements onto the fabric backing with tack stitches-- that is, sew and tie off the thread in discrete junctions. That way if one stitch gets broken, the whole circuit doesn't fall off of the backing!
Use it!

Sample placement and test shots: