LED Campfire
Created by Erin St Blaine

https://learn.adafruit.com/led-campfire

Last updated on 2023-08-29 03:10:42 PM EDT
# Table of Contents

**Introduction**  
- Materials  

**The Code**  
- Troubleshooting  

**Wiring Diagram**  
- Option 1  
- Option 2  

**Prep Dotstars**  

**Wiring - Option 1**  
- If you encounter trouble...  

**Wiring - Option 2**  

**Build the Fire**  

**Fiber Optic Sparks**
Introduction

Summertime means being outdoors, looking at the stars, and spending time with friends and family. Traditionally, a campfire or backyard fire has been a place to gather and relax after a hard day's hike.

Many campsites don't allow individual fires -- and in my world, it's just not true camping without a campfire. Nothing can replace a real fire, but this surprisingly realistic LED fire comes really close. Guaranteed to confuse and annoy your kids.

This is a great beginner project -- you can keep it really simple and build it in one afternoon, or get more complicated with onboard chargers and fiber optics. Marshmallows not included.

Materials

- Pro Trinket 5v () microcontroller
- Around 20 Dotstar LEDs -- the 30/m spacing () works great
- Driftwood or fire-sized logs
- Deck screws
- Silicone Glue
There are two different options for battery power, depending on your preference and budget...

Wiring option 1 (lower cost):

- Male JST Connector (http://adafru.it/1769)
- 3-cell AAA battery holder with switch (http://adafru.it/727) -or-
- Instead of the 3x AAA holder, you can substitute a rechargeable Lithium-Polymer battery (http://adafru.it/328), but it won’t charge over USB with this configuration — you’ll need a separate LiPoly charger (http://adafru.it/1304) for this.

Wiring option 2 (allows USB charging):

- A rechargeable Lithium Polymer battery (http://adafru.it/15782500)
- LiPoly Backpack charger (http://adafru.it/2124)
- Clicky on/off switch (http://adafru.it/1092)

Optional: to add fiber optic sparks:

- 1-2 Glowbys () in red or orange
- 1-2 100 ohm resistors

You'll also need a good soldering iron and some solder.

The Code

Though nothing is assembled yet, let’s get the code onto the microcontroller first. Then the project’s ready to test once all the connections are made.

Get out your Pro Trinket and plug it into your computer via the USB port.

Software Setup

If this is your first time using Pro Trinket, take a look at Introducing Pro Trinket () to get a guided tour. This walks you through installing the software necessary to use this board. Get the starter “blink” sketch working to confirm that the Arduino IDE is properly set up and speaking to the board.
Once you’ve got your Pro Trinket up and running with Arduino(), you’ll need to install the FastLED library...

FastLED is a fast, efficient, easy-to-use Arduino library for programming addressable LED strips and pixels. It has a lot of features to get your animations up and running fast -- and it has a lot of code samples available if you’re just learning to code.

Use the Library Manager in the Arduino IDE to install this (Sketch→Include Library→Manage Libraries...). Scroll down or use the search field to locate FastLED.

All about Arduino Libraries() will tell you everything you ever wanted to know about libraries, including more detailed installation instructions.

Once your curiosity is satiated and the FastLED library is installed, copy and paste the code below into your Arduino window.

Go to the Tools menu and select "Pro Trinket 5V USB" from the list of boards. Plug your Pro Trinket into your computer via the onboard USB port. Press the "reset" button on your Pro Trinket and wait for the blinky red light, then click the upload button in Arduino.

This wonderful Fire code was written by Mark Kriegsman, and is one of my favorite LED effects of all time.

```cpp
#include <FastLED.h>
#define LED_PIN     10
#define CLOCK_PIN   9
#define COLOR_ORDER BGR  //if your colors look incorrect, change the color order here
#define NUM_LEDS    20
#define BRIGHTNESS  255
#define FRAMES_PER_SECOND 20

CRGB leds[NUM_LEDS];

// Fire2012 with programmable Color Palette
//
// This code is the same fire simulation as the original "Fire2012",
// but each heat cell's temperature is translated to color through a FastLED
// programmable color palette, instead of through the "HeatColor(...)" function.
//
// Four different static color palettes are provided here, plus one dynamic one.
//
// The three static ones are:
// 1. the FastLED built-in HeatColors_p -- this is the default, and it looks
// 2. a gradient from black to red to yellow to white, which is
//
// The three static ones are:
// 1. the FastLED built-in HeatColors_p -- this is the default, and it looks
// 2. a gradient from black to red to yellow to white, which is
```

©Adafruit Industries
what the 'heat colors' palette is actually doing,
  3. a similar gradient, but in blue colors rather than red ones,
     i.e. from black to blue to aqua to white, which results in
     an "icy blue" fire effect,
  4. a simplified three-step gradient, from black to red to white, just to show
     that these gradients need not have four components; two or
     three are possible, too, even if they don't look quite as nice for fire.

The dynamic palette shows how you can change the basic 'hue' of the
color palette every time through the loop, producing "rainbow fire".

CRGBPalette16 gPal;

void setup() {
  delay(3000); // sanity delay
  FastLED.addLeds<DOTSTAR, LED_PIN, CLOCK_PIN, COLOR_ORDER>(leds, NUM_LEDS).setCorrection( TypicalLEDStrip );
  FastLED.setBrightness( BRIGHTNESS );

  // This first palette is the basic 'black body radiation' colors,
  // which run from black to red to bright yellow to white.
  // gPal = HeatColors_p;
  // These are other ways to set up the color palette for the 'fire'.
  // First, a gradient from black to red to yellow to white -- similar to
  HeatColors_p
  gPal = CRGBPalette16( CRGB::Black, CRGB::Red, CRGB::Yellow, CRGB::White);

  // Second, this palette is like the heat colors, but blue/aqua instead of red/yellow
  // gPal = CRGBPalette16( CRGB::Black, CRGB::Blue, CRGB::Aqua, CRGB::White);

  // Third, here's a simpler, three-step gradient, from black to red to white
  // gPal = CRGBPalette16( CRGB::Black, CRGB::Red, CRGB::White);
}

void loop() {
  // Add entropy to random number generator; we use a lot of it.
  random16_add_entropy( random());

  // Fourth, the most sophisticated: this one sets up a new palette every
  // time through the loop, based on a hue that changes every time.
  // The palette is a gradient from black, to a dark color based on the hue,
  // to a light color based on the hue, to white.
  //
  // static uint8_t hue = 0;
  // hue++;
  // CRGB darkcolor  = CHSV(hue,255,192); // pure hue, three-quarters brightness
  // CRGB lightcolor = CHSV(hue,128,255); // half 'whitened', full brightness
  // gPal = CRGBPalette16( CRGB::Black, darkcolor, lightcolor, CRGB::White);

  Fire2012WithPalette(); // run simulation frame, using palette colors

  Fire2012 by Mark Kriegsman, July 2012
  // as part of "Five Elements" shown here: http://youtu.be/knWiGsmgycY
  ///
  // This basic one-dimensional 'fire' simulation works roughly as follows:
  // There's a underlying array of 'heat' cells, that model the temperature
  // at each point along the line. Every cycle through the simulation,
  // four steps are performed:
  // 1) All cells cool down a little bit, losing heat to the air
  // 2) The heat from each cell drifts 'up' and diffuses a little
  // 3) Sometimes randomly new 'sparks' of heat are added at the bottom
  // 4) The heat from each cell is rendered as a color into the leds array
  // The heat-to-color mapping uses a black-body radiation approximation.
// Temperature is in arbitrary units from 0 (cold black) to 255 (white hot).
// This simulation scales itself a bit depending on NUM_LEDS; it should look
// "OK" on anywhere from 20 to 100 LEDs without too much tweaking.
// I recommend running this simulation at anywhere from 30-100 frames per second,
// meaning an interframe delay of about 10-35 milliseconds.
// Looks best on a high-density LED setup (60+ pixels/meter).
// There are two main parameters you can play with to control the look and
// feel of your fire: COOLING (used in step 1 above), and SPARKING (used
// in step 3 above).
// COOLING: How much does the air cool as it rises?
// Default 55, suggested range 20-100
#define COOLING  55
// SPARKING: What chance (out of 255) is there that a new spark will be lit?
// Higher chance = more roaring fire. Lower chance = more flickery fire.
// Default 120, suggested range 50-200.
#define SPARKING 120

void Fire2012WithPalette() {
  // Array of temperature readings at each simulation cell
  static byte heat[NUM_LEDS];

  // Step 1.  Cool down every cell a little
  for( int i = 0; i < NUM_LEDS; i++) {
    heat[i] = qsub8( heat[i], random8(0, ((COOLING * 10) / NUM_LEDS) + 2));
  }

  // Step 2.  Heat from each cell drifts 'up' and diffuses a little
  for( int k= NUM_LEDS - 3; k > 0; k--) {
    heat[k] = (heat[k - 1] + heat[k - 2] + heat[k - 2] ) / 3;
  }

  // Step 3.  Randomly ignite new 'sparks' of heat near the bottom
  if( random8() \&\& SPARKING ) {
    int y = random8(7);
    heat[y] = qadd8( heat[y], random8(160,255) );
  }

  // Step 4.  Map from heat cells to LED colors
  for( int j = 0; j < NUM_LEDS; j++) {
    // Scale the heat value from 0-255 down to 0-240
    // for best results with color palettes.
    byte colorindex = scale8( heat[j], 240);
    leds[j] = ColorFromPalette( gPal, colorindex);
  }
}

As soon as you get the "Upload Successful" notification in your Arduino window,
unplug the Pro Trinket and get ready for some soldering.
Troubleshooting

If you’re getting errors or having trouble uploading the code, here are a couple things to try:

1. Be sure you have “Pro Trinket 5v” selected from the Tools menu.
2. Make sure the FastLED library is installed.
3. Press the "reset" button on the Pro Trinket just after you hit the "upload" button in Arduino. When you press reset, the Pro Trinket will go into bootloader mode for only around 8 seconds, so you need to time things just right and upload the code during that window.
4. Try restarting your Arduino IDE.
5. If you’re still having trouble, try uploading the "Blink" sketch (File→Examples→Basics→Blink). This should blink the Pro Trinket's onboard LED. If this is working, you know your Arduino IDE and upload sequence are working, and that the problem lies elsewhere (e.g. missing library, or syntax error in the code).

Wiring Diagram

I’ve included two different diagrams: a simpler option with a AAA battery pack, and a slightly more advanced option with a LiPoly charger and Lithium Polymer battery pack.

Use Option 1 if you want a safer, versatile campfire that runs off disposable or rechargeable AAA batteries -- easy to find, hard to break, and can be switched out to last all night. Lithium Polymer batteries are great but they do have their dangers and challenges, so if kids will be using this campfire or you plan to let it bounce around in the back of your truck, this is the option for you.

Use Option 2 if you want to be able to recharge your campfire by plugging it in with a USB charger. If you use a good-sized Lithium Polymer battery, it will last for hours, but you’ll need to stop and recharge it when it dies (or, have a spare Lithium Polymer battery which can be changed out). If you do it this way, the fire can also run indefinitely while plugged in, so this is a great longer-term backyard solution.

You don’t need to actually solder any parts together yet, this is just a reference for what’s ahead...
Option 1

![Image of JST connector soldering to Pro Trinket and battery pack]

The JST connector solders to the back side of the Pro Trinket so you can plug your battery pack in.

Then the following connections are made between the microcontroller and Dotstar LED strip:

- Dotstar G --> Pro Trinket G
- Dotstar + --> Pro Trinket BAT+
- Dotstar C --> Pro Trinket 9
- Dotstar D --> Pro Trinket 10

Make sure you are connected to the “input” end of the LED strip. Examine the face of the strip and look for the arrows...these indicate the direction of data from “in” to “out” (as in the diagram above).
Option 2

The LiPoly Backpack is designed to “piggyback” atop the Pro Trinket board. The diagram shows them side-by-side to make the connections clear, but in reality just use a 3-pin header to stack them.

- LiPoly backpack Bat --> Pro Trinket Bat
- LiPoly backpack G --> Pro Trinket G
- LiPoly backpack 5v --> Pro Trinket Bus

The on/off switch connects into the two switch pads on the LiPoly backpack.

Four connections are made between the Pro Trinket and LED strip:

- Dotstar G --> Pro Trinket G pad on back
- Dotstar + --> Pro Trinket 5v pad on back
- Dotstar C --> Pro Trinket 9
- Dotstar D --> Pro Trinket 10
Your Dotstar strand should have connectors already soldered on to both the "in" and "out" end. The "in" end usually has a female connector and the "out" has a male. But not always! Use the arrows on the strip for reference, find the "out" end and cut that connector off, setting it aside for now. Later you'll solder this connector to the Pro Trinket, so you can plug it into the "in" end of your strip.

Cut your strip to length starting at the "in" end...measure 20 LEDs down the strip, then cut carefully through the silicone sheath and then cut the strip between solder pads. For a small to medium sized fire, 20 lights is a good length -- too many more and the battery will quickly drain.

Always Double Check Your Strand

Look at your strand before cutting! Use the arrows for reference...they point from “in” to “out.” The factory producing Dotstar strips has occasionally made changes between batches, so you shouldn’t assume a certain gender to the end connectors.

To seal the end of the strip and keep moisture and ants out, fill the cut end of the silicone sleeve with some hot glue. You can use a piece of heat shrink to hold it all neatly together.
Wiring - Option 1

The simplest way to hook up your Dotstars!

Get out the male connector you cut from the end of your Dotstars.
Strip about 1/8" of shielding from each wire. With your soldering iron, neatly and delicately "tin" each wire with just a tiny amount of solder. This will keep the wires from getting frayed and fuzzy and make it a little easier to fit them into the Pro Trinket's holes.

Find the 5v and G holes at the end of the Pro Trinket opposite the USB port. Solder the red wire into 5v and the black wire into G.

Then, solder the green wire into hole 10, and the yellow wire into hole 9.

It's a bit of a tight fit, and you may need to reduce the tin solder or trim the wires a bit, but they will fit (so don't give up).

Plug the connector into your Dotstar strip, and a USB cable into your Pro Trinket, and be sure the strand lights up with fiery light!
If you're happy powering from a USB battery or wall charger, you're done! Go build your fire. ()

If you want to be able to run the fire from a AAA battery pack or LiPoly battery, there's one more step.

Take some solder and tin the two pads on the back of your Pro Trinket, and carefully solder on a male JST connector.

Now you can plug in a battery of your choice.
If you encounter trouble...

Any time you hit a roadblock with a DotStar project, we’ll usually ask that you start with the “strandtest” example from our own Adafruit_DotStar library. This helps us narrow down whether it’s a hardware or software issue. The library is installed similarly to FastLED or any other — unzip, rename “Adafruit_DotStar” and place in your Arduino/Libraries folder, then restart the Arduino IDE. You’ll find the strandtest example under File→Sketchbook→Libraries→Adafruit_DotStar→strandtest

If strandtest fails to run, this suggests a hardware issue...for example, connecting to the wrong Pro Trinket pin.

If you’re new to Arduino programming and LEDs, we usually suggest starting with the Adafruit_DotStar library...it’s pretty basic, the strip declaration is more conventional, and we can stay on top of keeping it compatible with our own products and the most mainstream Arduino boards.

As FastLED is a more “bleeding edge” third-party library, we can’t always guarantee compatibility across versions or with specific boards. You can find help through their community on Google+(). This is potent stuff, written by people with a deep appreciation for LED art, and we wanted to showcase it.

Wiring - Option 2
Wiring with Battery Charging Support

This wiring method is slightly more complicated but may make your campfire easier to use and maintain.

We'll add an on/off switch and a LiPoly backpack charger. This means you can recharge your campfire's batteries the same way you'd charge your phone -- by simply plugging it into a USB charger. Hooray!

Just like with option 1, get out the male connector you cut from the end of your Dotstars.

Strip about 1/8" of shielding from each wire. With your soldering iron, neatly and delicately “tin” each wire with just a tiny amount of solder. This will keep the wires from getting frayed and fuzzy and make it a little easier to fit them into the Pro Trinket's holes.
Solder the green wire into hole 10, and the yellow wire into hole 9.

It's a bit of a tight fit, and you may need to reduce the tin solder or trim the wires a bit, but they will fit (so don't give up).

Then, flip the Pro Trinket over and add some solder to tin the + and - pads on the back (intended for the JST battery connector). Solder the red and black wires firmly onto these pads.
With a utility knife, carefully scratch the copper lead between the two switch pads on the LiPoly backpack.

Strip a little shielding from each of your switch leads and solder the two wires into these two holes. It doesn’t matter which wire goes into which hole.

On the back of the LiPoly backpack, bridge these two pads with a dot of solder. This allows faster charging with batteries over 500 mAh capacity such as we’re using here.
Solder the header that came with your LiPoly backpack into the remaining 3 holes from below. The easiest way to do this is using a solderless breadboard to help hold the pins in place.

Then, place the LiPoly backpack header pins into the Pro Trinket's Bat, G, and Bus holes (line up the G pin on the Pro Trinket with the one on the charger to be sure you've got this right). Solder the headers into place. Trim off any extra header that's sticking up.

Plug your LiPoly battery into the JST connector on the backpack, and plug your Dotstars into the Dotstar connector. Click the on/off switch and watch your fiery lights burn!

If the lights are dim or don't come on, double check all your wiring and make sure there are no fuzzy wires shorting across any pins. If you can't find anything wrong, check out the troubleshooting tips under option 1.
Now that your LED strip is flaming and sparking, it's time to add the logs to turn this project into a campfire.

Select logs and sticks with pleasing textures. Forked sticks work great, since you want the fire to pile and hold together nicely -- weaving the forked twigs in and around each other will give your fire architecture and height, and lots of nooks and crannies for hiding the LEDs.

Pile up the logs into a sturdy and aesthetically pleasing shape, then secure them together with long deck screws.
Turn on your LED strand and put your fire on top of it. Weave the strand in and around the logs. You're going for indirect light as much as possible here -- if you can see the bare LEDs it will ruin the effect. Use the logs to create artful diffusion.

Steel wool is another great diffusion medium for any spots your LED strip is still showing.

Glue the LED strip to the underside of the logs. Silicone glue works best for this -- it's about the only thing that will stick to the silicone sleeve on the Dotstars, and it fills in gaps in the wood nicely too.
Secure the Pro Trinket to an edge of the fire where you can reach the USB port and battery jack fairly easily, and glue it in place.

Secure the switch and battery somewhere inconspicuous, and you're ready for a smoke- and spark-free hootenanny.

If you've added the LiPoly backpack and charger, you can run the fire plugged in via the USB port on the Pro Trinket, and the battery will charge the whole time it's plugged in. Unplug it and it will run for hours on a charge -- mine lasted through the evening hours of a 4-day camping trip without needing to be recharged.

Fiber Optic Sparks
Adding fiber optic sparks to your fire takes it to the next level of awesomeness. Glowbys hair barettes are the perfect form factor and size, and can be easily added to your camp fire with a little spray glue.

Note: I'm using the older, discontinued version of Glowby hair clips since I had them on-hand. The newer versions will look slightly different, with an open/close lid instead of a sliding lid, but the wiring should still work the same way.

Glowbys run on two coin cell batteries, and they don't have an on/off switch (you have to remove the battery cover to turn them off). I wanted my fire to turn on with one button press and didn't want to have to fumble around with multiple batteries in the dark. Here's how I wired the Glowbys directly into my campfire's circuit.

Open the glowby up and pull out the batteries. Bend the LED leads up and solder a 100 ohm resistor to the positive lead, and a red wire to the other side of the resistor. Solder a black wire to the negative lead.

Bend the resistor around in an "S" shape so it fits neatly inside the battery space without shorting against itself.
Pull the metal bit out of the inside of the glowby's lid. Carefully close the lid, making sure your two wires are safely separated inside so they don't short out.

Now, splice your red and black wires into the red and black connector wires on your campfire.
Turn your campfire on and weave the fiber optics in and around your sticks. Gently secure them down with spray glue if needed.

Note: some spray glues may damage the fibers so be sure to test your glue on a small section first.

I also left a few sparks "hovering" above the fire. These sparks move and dance in a breeze or if I blow on the fire, adding another level of realism and awesomeness.