adafruit learning system

Trinket / Gemma Blinky Eyes

Created by Mike Barela

Last updated on 2018-08-22 03:38:08 PM UTC

Guide Contents

Overview and Parts

You have one day to make a halloween decoration! What will you do!? Well, if you have a Trinket and a couple of LEDs + a photo cell you can build these randomly blinky eyes that turn on when it gets dark

A recent MAKE project, Spooky Blinky Eyes (https://adafru.it/cSe) by Bill Blumenthal, demonstrates an ATTiny45 processor fading a pair of LED eyes that randomly blink, giving a more realistic effect than standard "always on" LED eyes.

This guide was written for the Trinket Mini and Gemma v2 boards. It has been updated to also support the Trinket M0 and Gemma M0 using CircuitPython. We recommend the Trinket M0 or Gemma M0 as they are easier to use and are more compatible with modern computers!

The effect is due to come clever programming of the timers available on the ATTiny processors featured on the Adafruit Trinket and Gemma microcontrollers. Pins 0 and 1 are capable of pulse width modulation. The timers are set to fade the pins in and out by changing the pulse width back and forth. The blink effect is using an algorithm called a linear feedback shift register (https://adafru.it/cSf) (LFSR) to pseudo-randomly turn the eyes off and on quickly.

This project adapts the original code for use on the faster ATTiny85 processor and for the Arduino integrated development environment (IDE). It also adds a Cadmium Sulfide (CdS) photocell to allow the eyes to come on only below a certain light level. During sunlight, this will save battery power.

Parts List

- **Adafruit Gemma M0 (https://adafru.it/ytb), Trinket M0 (https://adafru.it/zya), Trinket Mini** (https://adafru.it/egk) or **Gemma v2** (http://adafru.it/1222) microcontroller board (if Trinket, either the 3.3V (http://adafru.it/1500) or 5V (http://adafru.it/1501)type works). We recommend using the **Trinket**

M0 (https://adafru.it/zya).

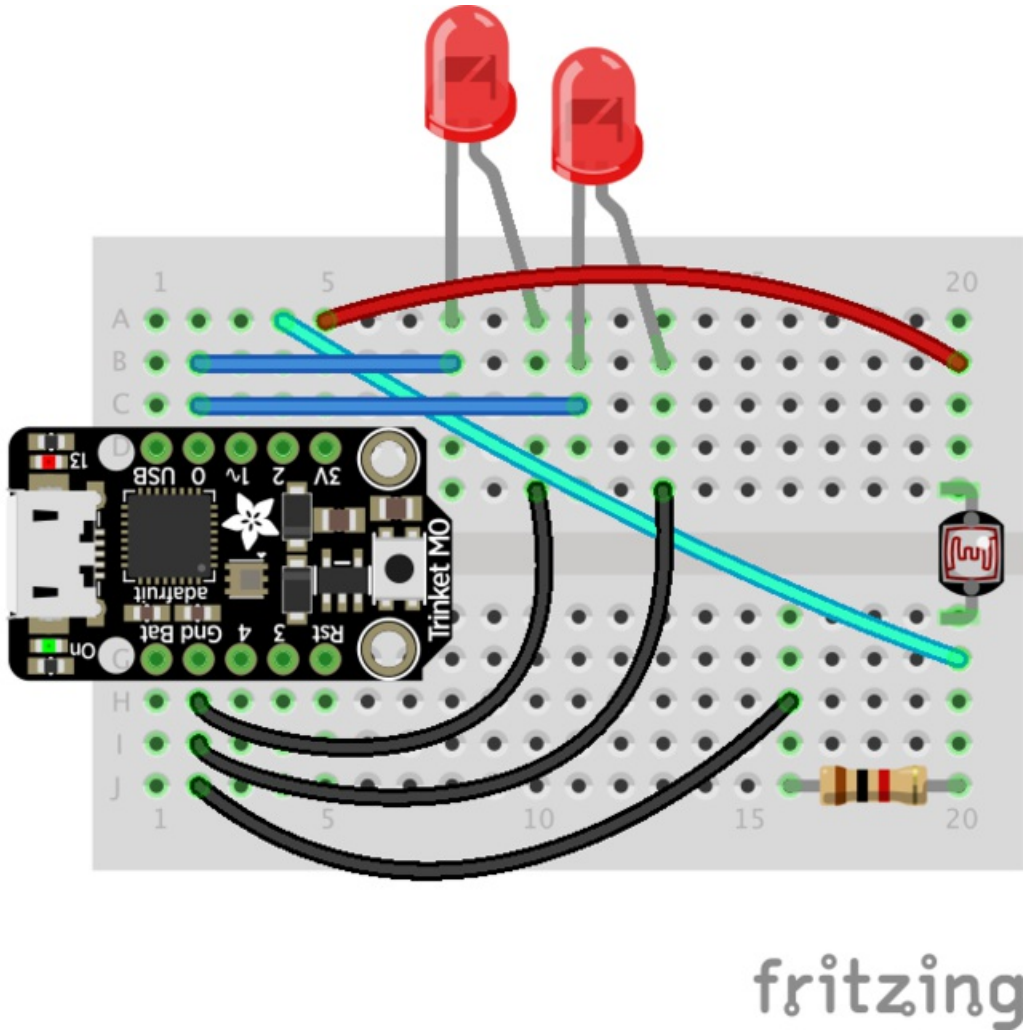
- Cadmium Sulfide Photocell (https://adafru.it/cSh)
- 2 LEDs (I used two 5 mm red (http://adafru.it/297) for the evil look, you can use any size or color)
- (1k) ohm resistor
- Tiny Breadboard (http://adafru.it/65) (or other suitable wiring surface)
- 6 volt coin cell battery pack (http://adafru.it/783)
- 2 CR2032 Batteries (http://adafru.it/654)
- A prop to put your circuit in

Wiring

The circuit is fairly straightforward and can be assembled well within an hour.

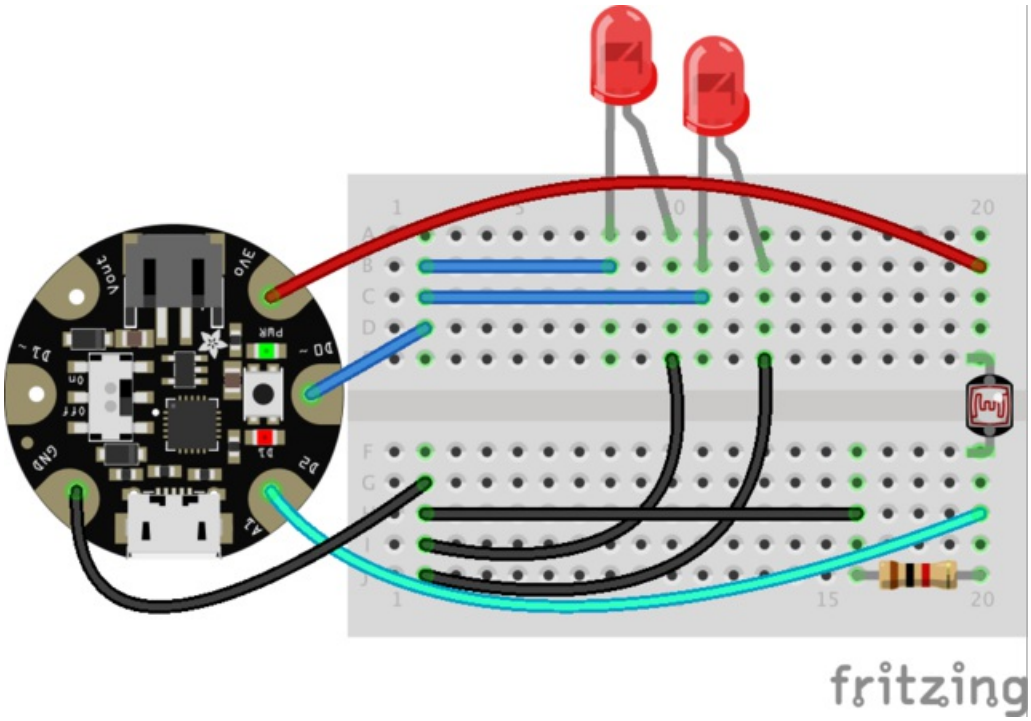
Headers (included with Trinket) may be soldered to facilitate attachment to a breadboard or proto board. Other parts may be pressed into the breadboard. Ready-made hookup wire or cut to fit wires make the connections.

These diagrams are the same for all versions of Trinket and Gemma.



The same circuit with Gemma. The LEDs share D0. D2/A1 is hooked to the junction of the photocell and a 1000 ohm (1K) resistor. If you wish to just rely on the power pack on/off switch, you can eliminate the photocell and 1K resistor.

The battery pack works well with wearables. It has a JST connector to plug directly into Gemma. It also has the on/off switch to save power when the circuit is not being used.


fritzing

Arduino Code

Preparing Your Trinket

Be sure you have a 3 volt Trinket or Gemma. Follow the Introducing Trinket (https://adafru.it/cEu) or Introducing Gemma (https://adafru.it/cHH) tutorials rather carefully to ensure your Arduino integrated development environment (IDE) is set up.

> The Arduino code presented below works well on the Trinket Mini and Gemma v2. If you have an M0 board you can use the CircuitPython code on the next page of this guide.

Remember you must press the hardware reset button on the Trinket / Gemma then quickly press upload in the Arduino software to upload a sketch. If you get an error, try the reset-upload process again. If you continually cannot load the blink sketch, check to make sure the Trinket / Gemma is connected (without any wires connected to pins #3 and #4) and the Arduino IDE software has all the required changes. This project does not use pins 3 and 4, they are shared with the USB connector (Gemma does not have these pins).

The following uses PWM for the eye brightness. Controlling the counter which helps random blinks still uses hardware Timer 1. A random functionality could be used but would not give as nice a "randomness" and possibly not be timed right given all that is going on with the state machine.

```
/*
Name: Blinking Eyes  - based on code by Brad Blumenthal, MAKE Magazine
License: GPLv3
Modified for 8 MHz ATTiny85 and low light photocell
October 2013 for Adafruit Learning System
*/

#define SENSITIVITY 550        // photocell sensitivity (changeable)
#define CELL_PIN    1          // CdS Photocell voltage divider on
                               //   Trinket GPIO #2 (A1), Gemma D1/A1
uint8_t eyes_open;
volatile uint8_t  blink_count;
volatile uint8_t  blink_flag;
volatile uint8_t  tick_flag;
volatile uint8_t  getting_brighter = 0;
const uint8_t    min_bright=16;
const uint8_t    max_bright=128;
volatile uint8_t brightness;
uint8_t          lfsr;            // Linear Feedback Shift Register
const uint8_t    min_blink = 64u; // don't blink more than once every 3 secs or so

void setup() {
  pinMode(0, OUTPUT);                                    // Eyes set as output
  pinMode(2, INPUT);                                     // Photocell as input
  analogWrite(0, max_bright);  analogWrite(1, max_bright);  // Light eyes
  eyes_open = 1;
  blink_flag = 0;
  lfsr = random(100);                                    // initialize "blinking"
  blink_count = max(blink_count, min_blink);
  lfsr = (lfsr >> 1) ^ (-(lfsr & 1u) & 0xF0u);           // pseudorandom blinking

  // Timer1 set to CK/1024  ~10 (8) hZ at 8 MHz clock rate for blinking action
  TCCR1 |= _BV(CS13) | _BV(CS11) | _BV(CS10);
  TIMSK |= _BV(TOIE1);  // Enable Timer/Counter1 Overflow Interrupt
```

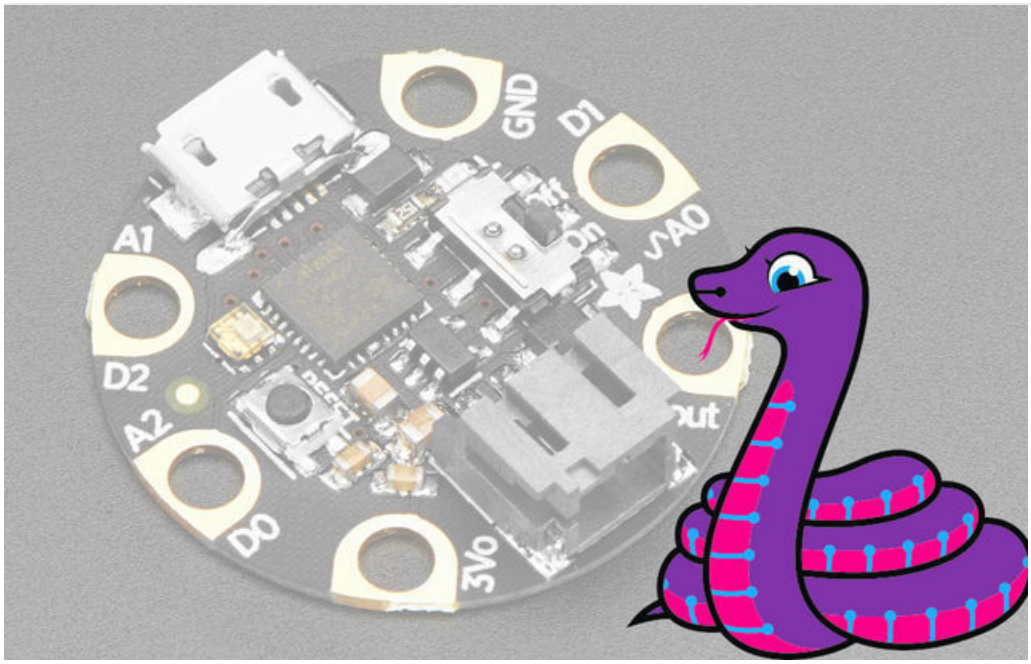
```
  TIMSK |= _BV(TOIE1);  // Enable Timer/Counter1 overflow interrupt
}

void loop() {
    uint16_t photocell;
    photocell = analogRead(CELL_PIN);
    if(photocell > SENSITIVITY) {  // if too light, shut down eyes until it gets darker on photocell
       tick_flag=0;
       analogWrite(0,0);  // Turn off eyes if too light out
      }
    if (tick_flag) {  // if too bright or we've counted enough ticks (clocks for blink)
      tick_flag = 0;
      if (blink_flag) {
        blink_flag = 0;
        if (eyes_open) {
          eyes_open = 0;
          analogWrite(0,0);  // Turn off eyes by stopping PWM
          blink_count = (lfsr & 0x01) + 1; // off for 1-2 ticks
          }
        else {
          eyes_open = 1;
          analogWrite(0,brightness);  // Turn eyes on
          blink_count = max(blink_count, min_blink);
          lfsr = (lfsr >> 1) ^ (-(lfsr & 1u) & 0xF0u);  // regenerate pseudorandom blink
          }
      }
      else { // One "tick,"  but we didn't blink...  work on brightness control
        if (getting_brighter) {
          brightness += 2;  // increase brightness
          analogWrite(0, brightness);
          if (brightness >= max_bright) getting_brighter = 0;
        } else {
          brightness -= 2;  // decrease brightness
          analogWrite(0, brightness);
          if (brightness <= min_bright) getting_brighter = 1;
        }
      }
    }
}

ISR (TIMER1_OVF_vect) {           // Every 64 times a second, check blink
  noInterrupts();
  tick_flag = 1;
  blink_count--;
  if (!blink_count) {
    blink_flag = 1;
  }
  interrupts();
}
```

The Arduino code compiles to 1626 bytes.

CircuitPython Code



Trinket M0 and Gemma M0 boards can run **CircuitPython** — a different approach to programming compared to Arduino sketches. In fact, **CircuitPython comes factory pre-loaded on Trinket M0**. If you've overwritten it with an Arduino sketch, or just want to learn the basics of setting up and using CircuitPython, this is explained in the **Adafruit Trinket M0 guide (https://adafru.it/zxF)** or **Adafruit Gemma M0 guide. (https://adafru.it/BeC)**

> These directions are specific to the "M0" boards. The original Trinket and Gemma with an 8-bit AVR microcontroller do not run CircuitPython...for those boards, use the Arduino sketch on the "Arduino code" page of this guide.

Below is CircuitPython code that works similarly (though not exactly the same) as the Arduino sketch shown on a prior page. To use this, plug the Trinket M0 into USB...it should show up on your computer as a small **flash drive**...then edit the file "**main.py**" with your text editor of choice. Select and copy the code below and paste it into that file, **entirely replacing its contents** (don't mix it in with lingering bits of old code). When you save the file, the code should **start running almost immediately** (if not, see notes at the bottom of this page).

If Trinket M0 doesn't show up as a drive, follow the Trinket M0 guide link above to prepare the board for CircuitPython.

```
"""
Blinking Eyes - based on code by Brad Blumenthal, MAKE Magazine
License: GPLv3
"""

import time

import analogio
import board
import pulseio

try:
```

```
    import urandom as random  # for v1.0 API support
except ImportError:
    import random

# Initialize photocell
photocell_pin = board.A1  # cds photocell connected to this ANALOG pin
darkness_max = (2 ** 16 / 2)  # more dark than light > 32k out of 64k
photocell = analogio.AnalogIn(photocell_pin)

# Initialize PWM
# PWM (fading) - Both LEDs are connected on D0
# (PWM not avail on D1)
pwm_leds = board.D0
pwm = pulseio.PWMOut(pwm_leds, frequency=1000, duty_cycle=0)
brightness = 0  # how bright the LED is
fade_amount = 1285  # 2% steping of 2^16
counter = 0  # counter to keep track of cycles

# blink delay
blink_delay = True
blink_freq_min = 3
blink_freq_max = 6

# Loop forever...
while True:

    # turn on LEDs if it is dark out
    if photocell.value < darkness_max:

        # blink frequency and timer
        if blink_delay:
            blink_delay = False
            blink_timer_start = time.monotonic()
            blink_freq = random.randint(blink_freq_min, blink_freq_max)

        # time to blink? Blink once every 3 - 6 seconds (random assingment)
        if (time.monotonic() - blink_timer_start) >= blink_freq:
            blink_delay = True
            pwm.duty_cycle = 0
            time.sleep(.1)

        # send to LED as PWM level
        pwm.duty_cycle = brightness

        # change the brightness for next time through the loop:
        brightness = brightness + fade_amount

        # reverse the direction of the fading at the ends of the fade:
        if brightness <= 0:
            fade_amount = -fade_amount
            counter += 1
        elif brightness >= 65535:
            fade_amount = -fade_amount
            counter += 1

        # wait for 15 ms to see the dimming effect
        time.sleep(.015)

    else:
```

```
        # shutoff LEDs, it is too bright
        pwm.duty_cycle = 0
```

Building Your Project

Blinky eyes may be placed into nearly anything. That is most of the fun in this project. Select where you would like to put them then build the appropriate lighted eyes, put them in and you have a complete project ready to go.

For my project, I went to that staple of American shopping experiences: Walmart. There in the Halloween section is the 97 cent box. I found an assortment of small sippy cups crying to be liberated from their straws and modded to be small, scary light sensitive balls of evil. You may chose to place the eyes into any item or even a board to hide in a window or the bushes.

Removing the straws, I drill small holes in the eyes then use a larger 5 mm but to size the holes the same as the LEDs I chose. The hole in the top where the straw was will serve as the place the photocell will be mounted to detect whether it is dark enough to turn on the eyes.

I placed the photocell and LEDs on the ends of male to female jumper wires. Any hookup wire will do, but the jumper wires hook right in to my breadboard and the components. You will want to trim the component leads a bit before placing in the female end to avoid the wires touching and shorting out.

Put the batteries in the holder and for Trinket plug into the breadboard with the red lead to the BAT+, the black to GND. If you use Gemma, plug the battery holder JST (white) connector into the Gemma battery socket. Lower the board into your selected container and thread the LEDs into the eye holes. If you want, a dab of glue will hold the LEDs in place. Thread the photocell through a hole in the top and secure with tape, Sugru (http://adafru.it/436), or other substance being careful not to twist the wires or make them touch.

Turn on the curcuit and put the lid on. Now when it is dark enough, the lights will fade in and out and "blink" (go on and off quickly). Enjoy your scary eyes project.

Low Light Sensitivity

If you find the photocell sensitivity too high or low (which may vary due to the cell's characteristics or the actual value

of the 1K resistor), change the value of SENSITIVITY near the top of the code. Valid numbers are zero to 1023, you will want to select a number from about 150 to 800. The lower the number, the darker it must be.