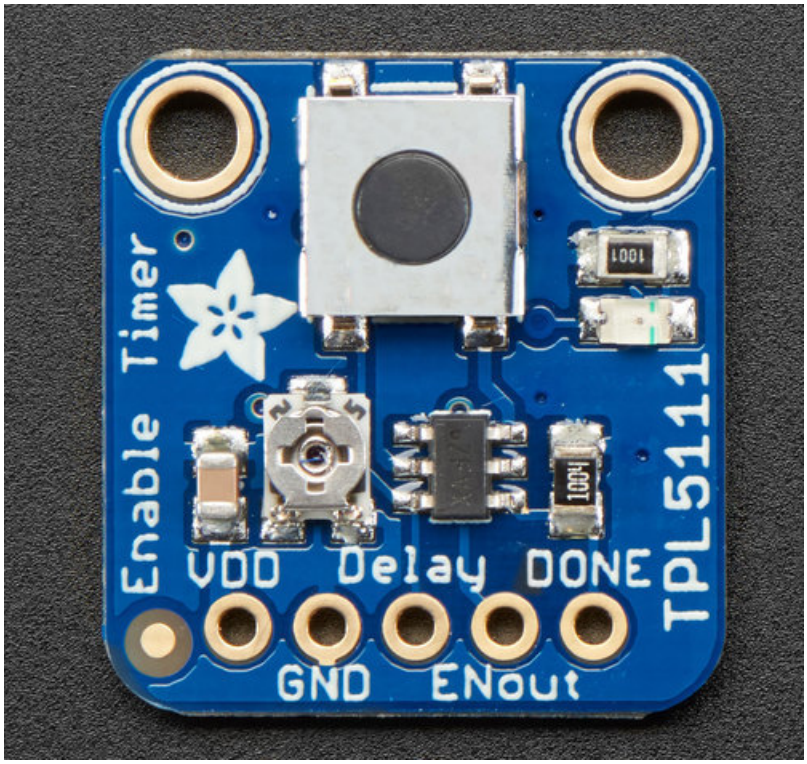
adafruit learning system

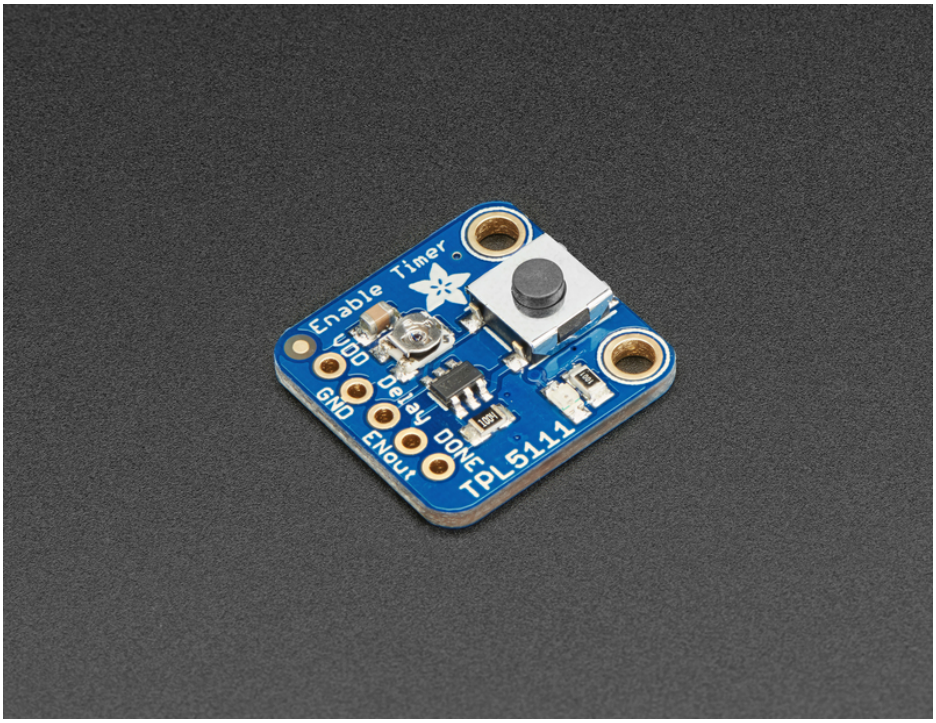
Adafruit TPL5111 Reset Enable Timer Breakout

Created by lady ada



Last updated on 2019-10-02 06:41:49 PM UTC

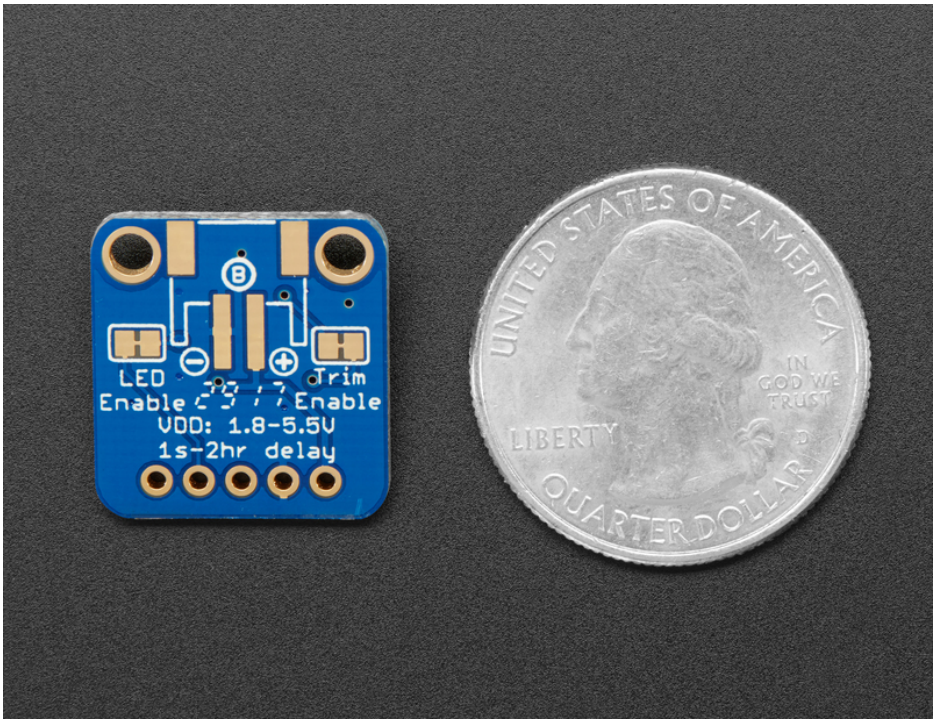
Overview



With some development boards, low power usage is an afterthought. Especially when price and usability is the main selling point. So what should you do when its time to turn around and make that project of yours run on a battery or solar? Sure you *could* try to hot-air that regulator off, or you could jerry-rig a relay. Or, use a 555? Ugh, the options aren't that great.

The **Adafruit TPL5111 Reset Enable Timer** is a stand-alone breakout that will turn *any* electronics into low-power electronics! It will take care of enabling & disabling your electronics using a built in timer that can vary from once-every 100ms up to once every two hours. Basically, the TPL will set an enable pin high periodically, adjustable by potentiometer or resistor, and turn on your project's power. It will then wait until a signal is received from the project to tell the TPL that it can safely disable the project by setting the enable pin low. If the TPL does not receive a signal by the set time-out, it will reset the device like a watchdog timer.

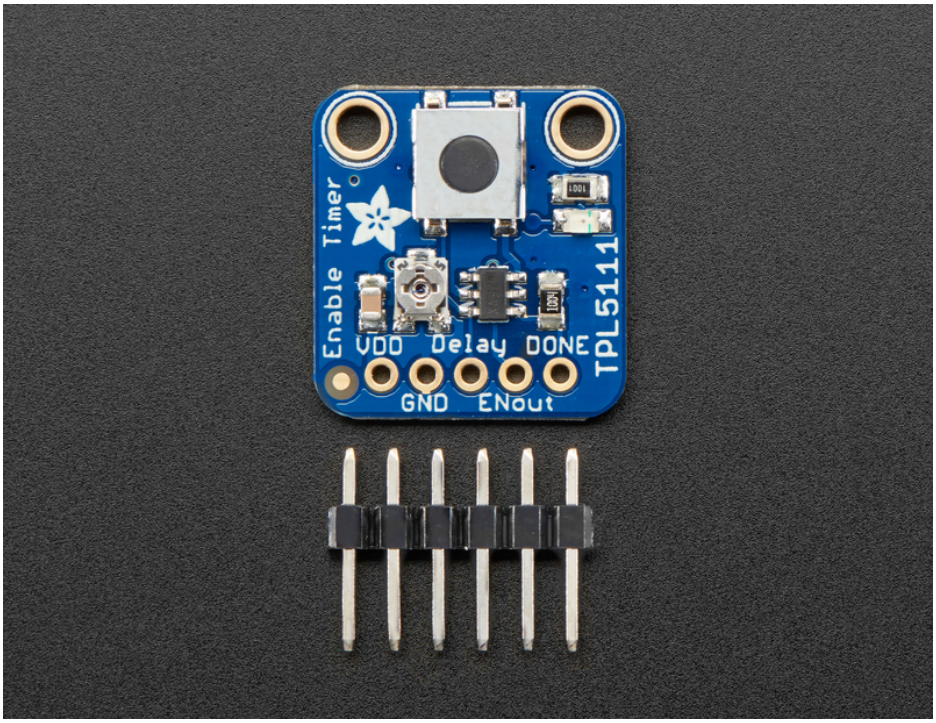
We also have a **TPL5110** breakout, which rather than setting an enable pin high/low, connects and disconnects power. The **TPL5111** breakout is best when you have an enable pin you can control, the **TPL5110** is best when you are able to 'break' the power input line to place the **TPL5110** between the power supply and Vin.



Usage is easy. First, set your desired delay but adjusting the on-board trim pot: all the way to the left is once-per-100ms an all the way to the right is once-every-2-hours. Then, connect VDD up to your 3-5V project power supply, and then your project's enable pin to the Enable pin. Finally, select a signal pin from your project to the Done pin. In your project's code or design, just make sure that it sets the Done pin high once it is completed with it's task. That's it!

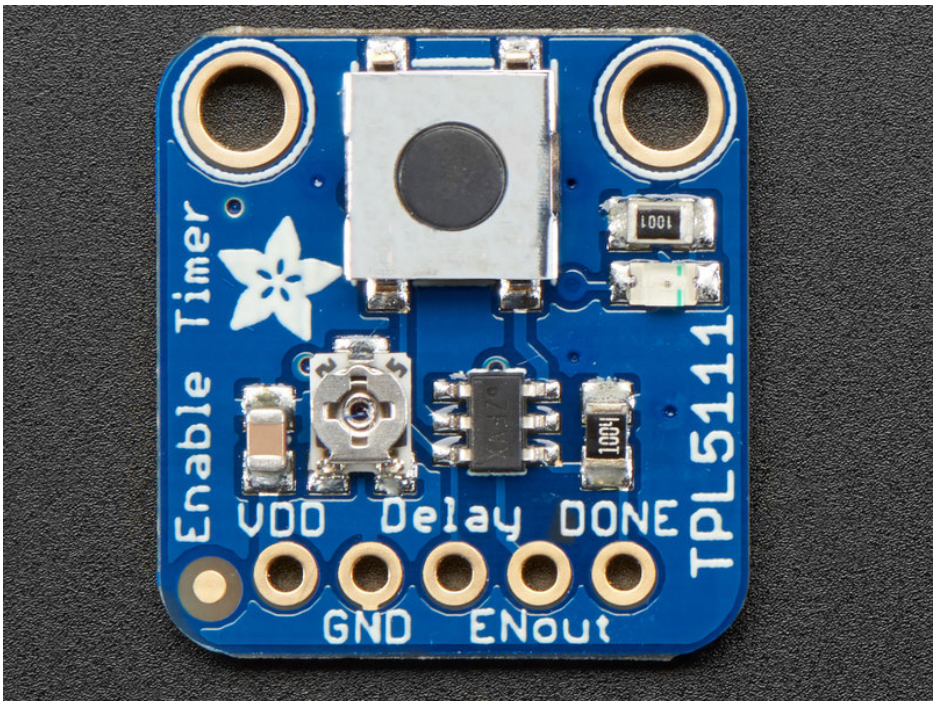
While the TPL5111 is running (but the remainder of the project is de-powered) the current draw is about 20uA (according to our Monsoon Power Meter)

If you want to turn the device on by hand, you can also activate the TPL by pressing the onboard tactile switch (or wire your own switch to the Delay pin) See? Your power problems just got *solved!*



Comes as a fully assembled breakout board with a TPL5111 chip, all components on-board, and some header. A little light soldering is required to put it together for breadboard use.

Pinouts



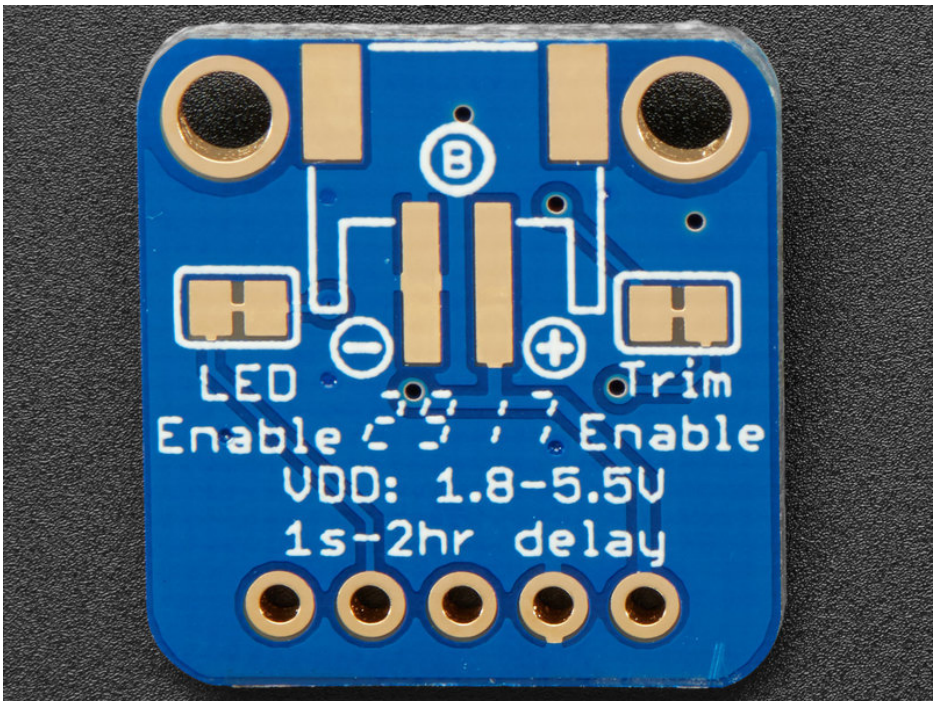
There's a lot going on with this compact breakout, and many ways to do one thing so reviewing the pinouts is a really good idea!

Power Pins

- **VDD** - this is the power input pin for the TPL5111 chip. Make sure this matches the logic level for the device you want to control. Make sure this is always connected and powered. This has to be 3-5VDC so don't give it 12VDC power!
- **GND** - this is shared ground for power and signal.

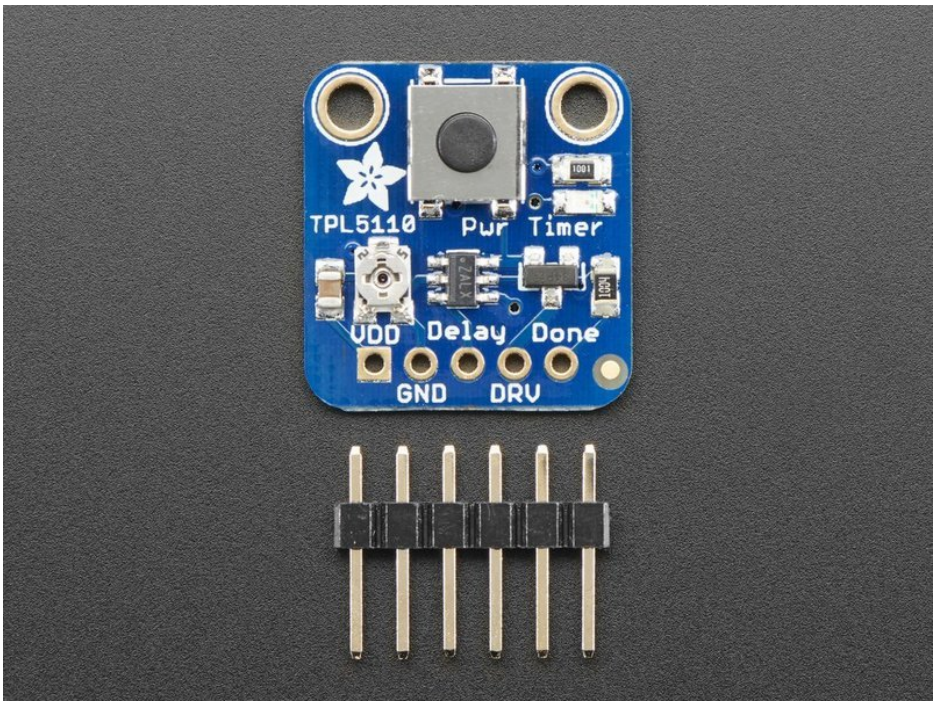
Control Pins

- **Delay** - this is the delay control pin. By adjusting the *resistance* (not voltage!) connected to this pin through to ground, you can change the delay between powerings. By default this is connected to the onboard trim potentiometer. However, you can cut the trace on the back and then connect your own resistor between **Delay** and **GND.** Also, if this pin is connected directly to VDD (say through a switch) it will activate the output instantly. Note this pin is *not* continuously sampled. You need to remove and re-apply power once you change the resistance!
- **ENout** - this is the Enable output pin, the TPL5111 will set this pin high with the same voltage as from VDD when the timer activates
- **Done** - This is the signal pin *from the driven electronics* back to the TPL5111 to let it know that it is 'done' with whatever it had to do, and the TPL5111 can turn it off by setting the ENout pin low

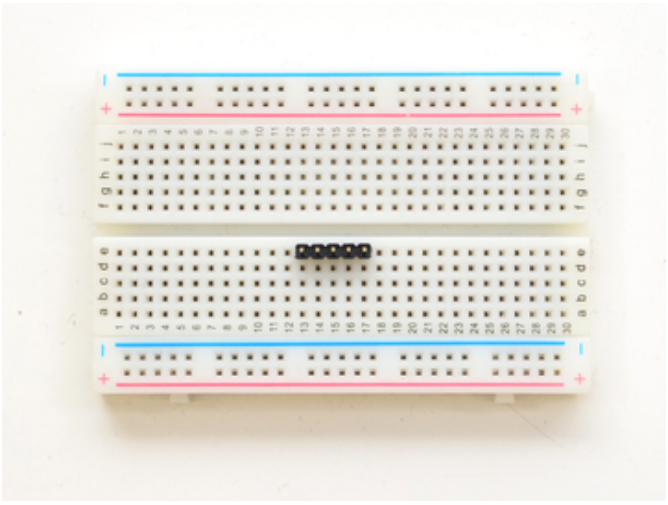


There is also an 'active' LED in the top right. This will let you know when the **ENout** pin is powered. It does draw some current so if you need ultra-low current draw, cut the trace behind the PCB!

Assembly

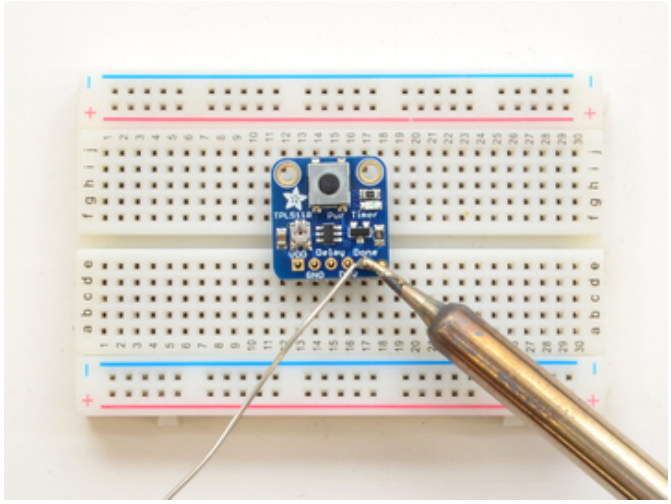


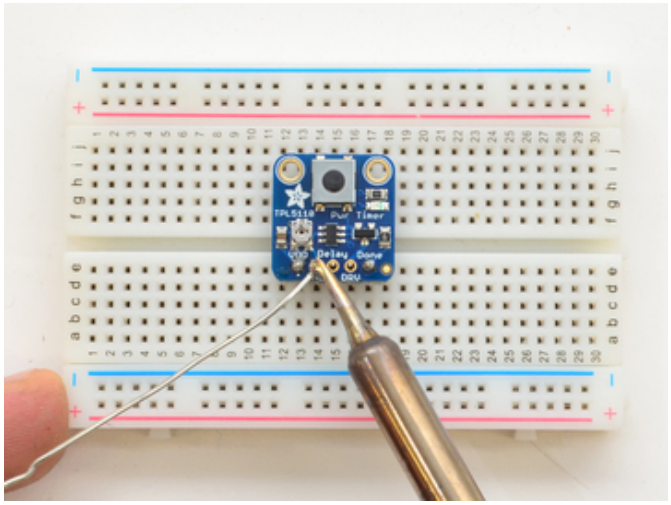
This page shows the TPL5110 but the TPL5111 procedure is identical!



Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



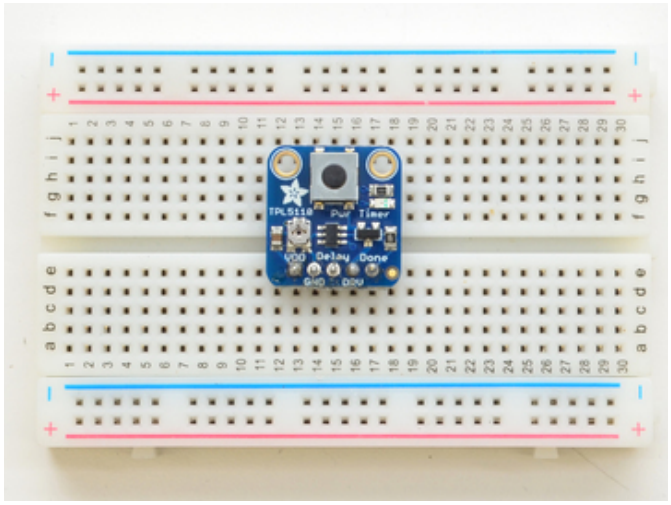


Add the breakout board and Solder!

Place the breakout board over the pins so that the short pins poke through the breakout pads

Be sure to solder all 5 pins for reliable electrical contact.

(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk).



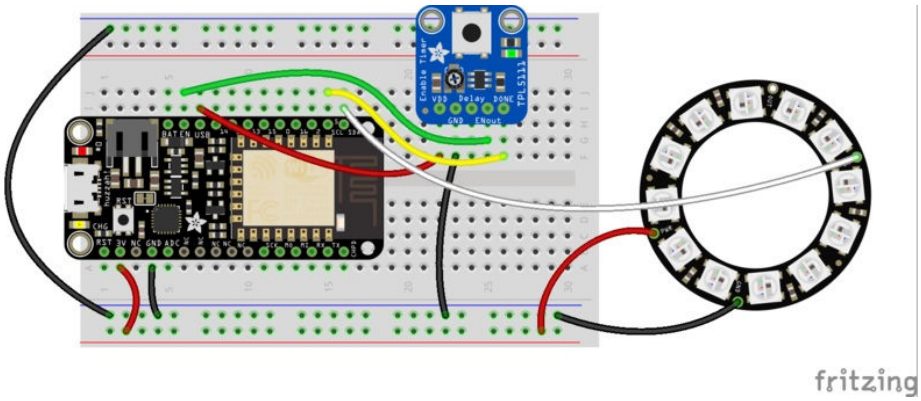
You're done! Check your solder joints visually and continue onto the next steps

Usage

Using the TPL5111 isn't too hard but there's a few things to watch out for. First up, do not give it 9V power, use 3-5V only!

Use the TPL5111 with any board that has an **Enable** pin. If an Enable pin isn't available, the **Reset** pin can sometimes be used - but often won't give you as much power savings!

Here's an example with an ESP8266 Feather. Power the Feather by USB or battery as normal



Make sure to provide the power to the **VDD** and **GND** pins.

Remember The VDD pin has to be always powered so you cannot connect it to the power pin you are enabling/disabling!

Then connect your project's Enable pin to the **ENout** pin, and also connect up the **GND** pin. Use a **DONE** pin from your microcontroller to signal when the TPL can disable power: when the **DONE** pin goes from low to high, that will turn off the TPL's power transistor.

Turn the trim potentiometer *clockwise* to make the resistance lower - measure the resistance between **Delay** and **GND** and check against the table below to make sure you get the timing you want

In this project I'm using the ESP8266 Feather's digital #5 as the **DONE** signal pin. It is lightly pulled down so just set to an Output and High when you're done!

```
#include <Adafruit_NeoPixel.h>

#if defined(ESP8266)
  #define NEOPIX   4
  #define DONEPIN  5
#else
  #include <Adafruit_SleepyDog.h>
  #define NEOPIX   13
  #define DONEPIN  12
#endif

Adafruit_NeoPixel strip = Adafruit_NeoPixel(12, NEOPIX, NEO_GRB + NEO_KHZ800);


void setup() {
  pinMode(DONEPIN, OUTPUT);
```

```
  digitalWrite(DONEPIN, LOW);
  Serial.begin(115200);
  Serial.println("Light up NeoPixels!");

  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
  strip.setBrightness(20);
}

void loop() {
  rainbowCycle(5);

  Serial.println("NeoPixels done, sleeping");

  // toggle DONE so TPL knows to cut power!
  while (1) {
    digitalWrite(DONEPIN, HIGH);
    delay(1);
    digitalWrite(DONEPIN, LOW);
    delay(1);
  }

  Serial.println("Awake!");
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*1; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
  }
}


// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if(WheelPos < 85) {
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

For the code, we toggle the DONE pin high and low forever to make sure it gets 'caught' by the TPL (it may not be necessary but it doesn't hurt!)

If the TPL *doesn't* get a DONE signal, it will reset the board with a short ENABLE toggle when the timeout is reached

(e.g. before the next cycle)

Note that if you disable the board you nay also disable the USB/Serial chip - so you have to unplug the TPL5111 while programming!

Notes on the Delay Pin

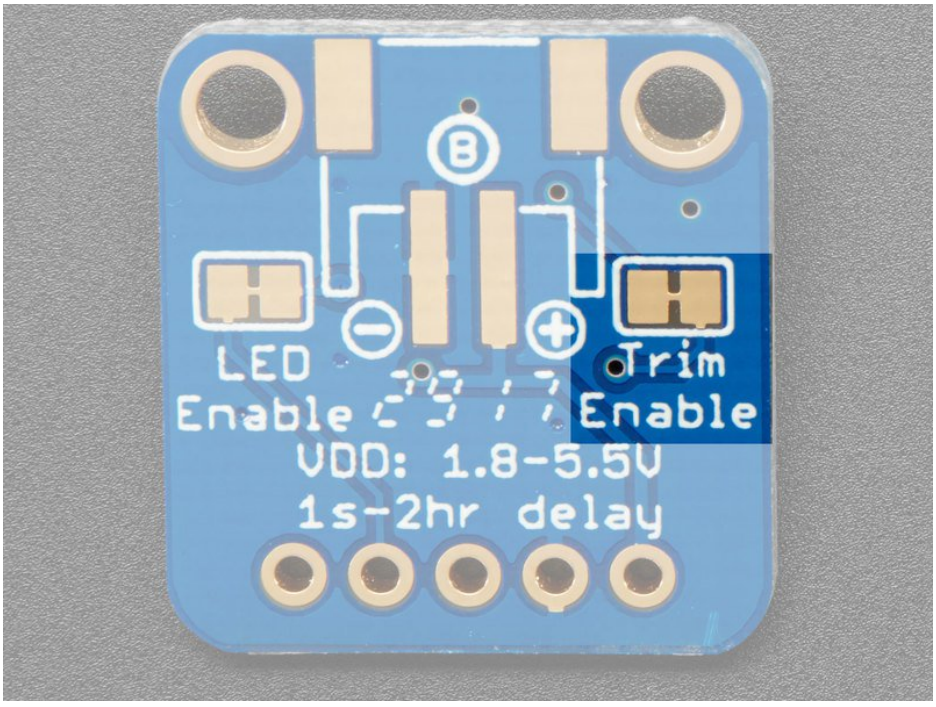
The delay pin is a little more complicated than you may first think!

- First, do not connect a voltage here, instead it uses a resistor to determine the delay timing.
- Second, it does not continuously sample the resistor, it only does it *once* when power is applied. So set the delay you want, then power up the breakout.
- Third, you can instantly turn on the project by connecting **Delay** to ground. By default we have a pushbutton on board, you can connect your own button if you like
- Fourth, the resistance is not linear with the time delay, rather there is a complex algorithm to set the time based on resistance. You can check the datasheet for the precise calculation, or use this rough table:

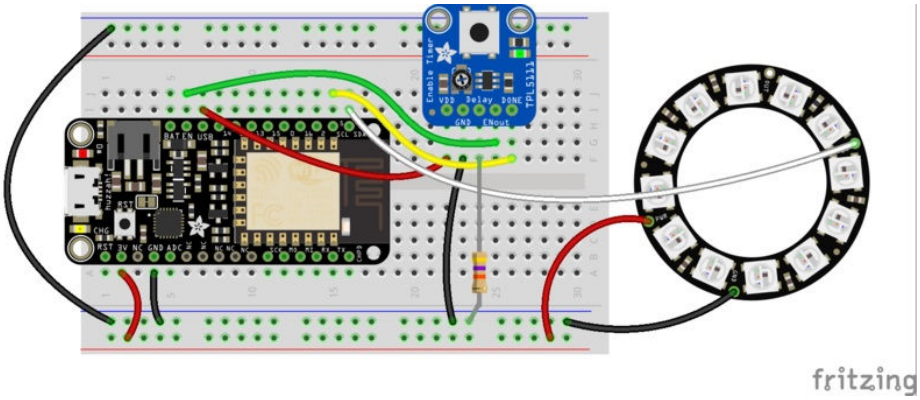
1 Seconds	5.2 KΩ
2 Seconds	6.79 kΩ
3 Seconds	7.64 kΩ
4 Seconds	8.3 kΩ
5 Seconds	8.85 kΩ
6 Seconds	9.26 kΩ
7 Seconds	9.71 kΩ
8 Seconds	10.18 kΩ
9 Seconds	10.68 kΩ
10 Seconds	11.2 kΩ
20 Seconds	14.41 kΩ
30 Seconds	16.78 kΩ
40 Seconds	18.75 kΩ
50 Seconds	20.047 kΩ
1 Minute	22.02 kΩ
2 Minutes	29.35 kΩ
3 Minutes	34.73 kΩ

4 Minutes	39.11 kΩ
5 Minutes	42.90 kΩ
6 Minutes	46.29 kΩ
7 Minutes	49.38 kΩ
8 Minutes	52.24 kΩ
9 Minutes	54.92 kΩ
10 Minutes	57.44 kΩ
20 Minutes	77.57 kΩ
30 Minutes	92.43 kΩ
40 Minutes	104.67 kΩ
50 Minutes	115.33 kΩ
1 Hour	124.91 kΩ
1 Hour 30 Minutes	149.39 kΩ
2 Hours	170 kΩ

Given that we put a 200 kΩ trimpot on the board, you may find it difficult to get precise timing if you need short delays. In that case, you can use any resistor you want. First, cut the trace on the back of the PCB



Then install your desired resistor:



Don't forget to hard-reset the full setup!

Downloads

Files

- Fritzing object in Adafruit Fritzing library (https://adafru.it/aP3)
- EagleCAD PCB files on GitHub (https://adafru.it/yxC)
- TPL5111 Datasheet (https://adafru.it/yxD)

Schematic and Fabrication Print

