

# Getting Started using a Particle Photon

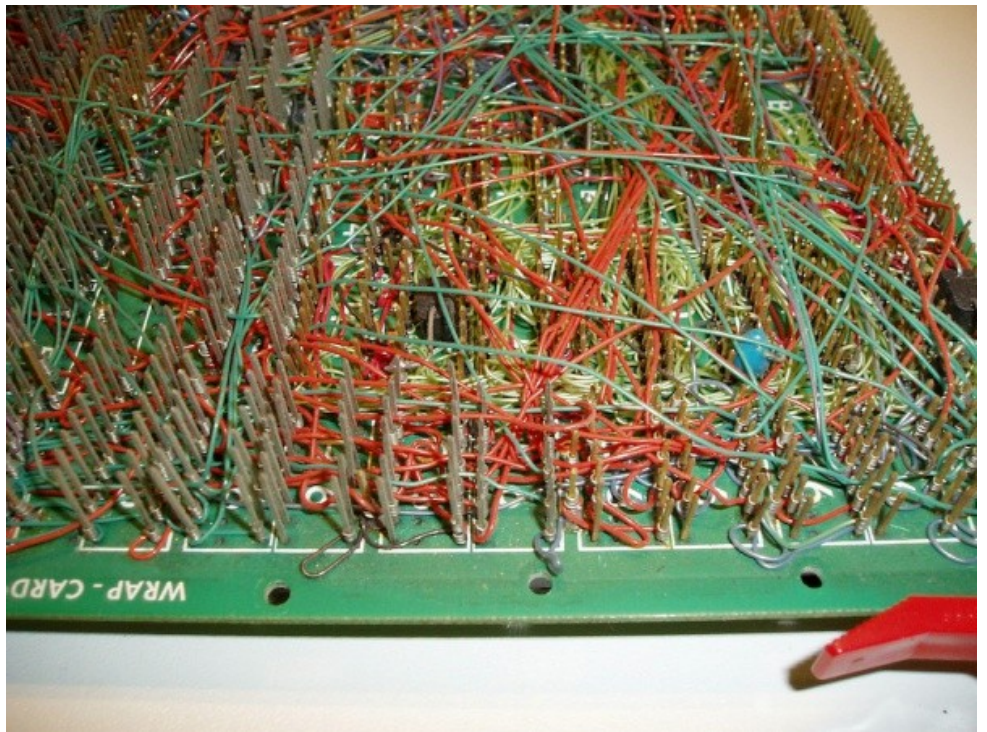
David Dixon  
April 13, 2016

# Objectives

- Learn to connect to, flash code to, read cloud data from, and perform other basic tasks with a small microcontroller (Particle Photon).
- Build a prototype for monitoring holding tank levels, using a variable resistor as a substitute for a pressure transducer

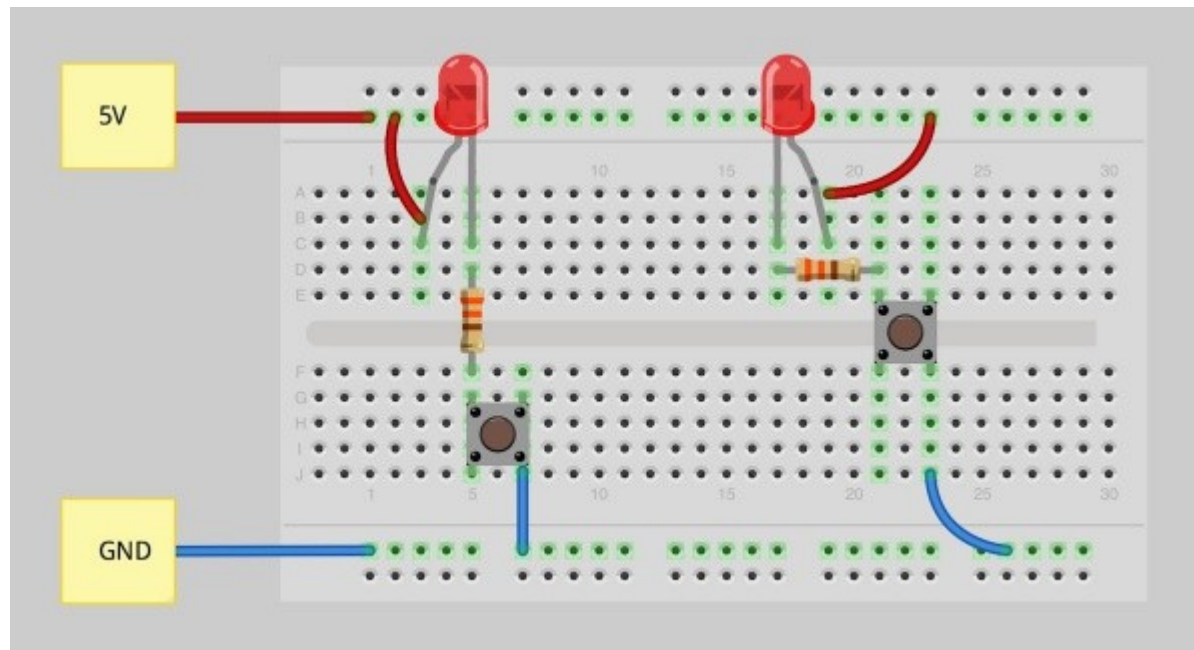
# Why a breadboard?

- Easy, solderless connections to test circuit designs
- You will make mistakes, forget things, and realize that your initial design has “bugs”
- It's a lot easier to yank a wire out than it is to desolder.



# How to use a breadboard

- <https://learn.sparkfun.com/tutorials/how-to-use-a-breadboard>

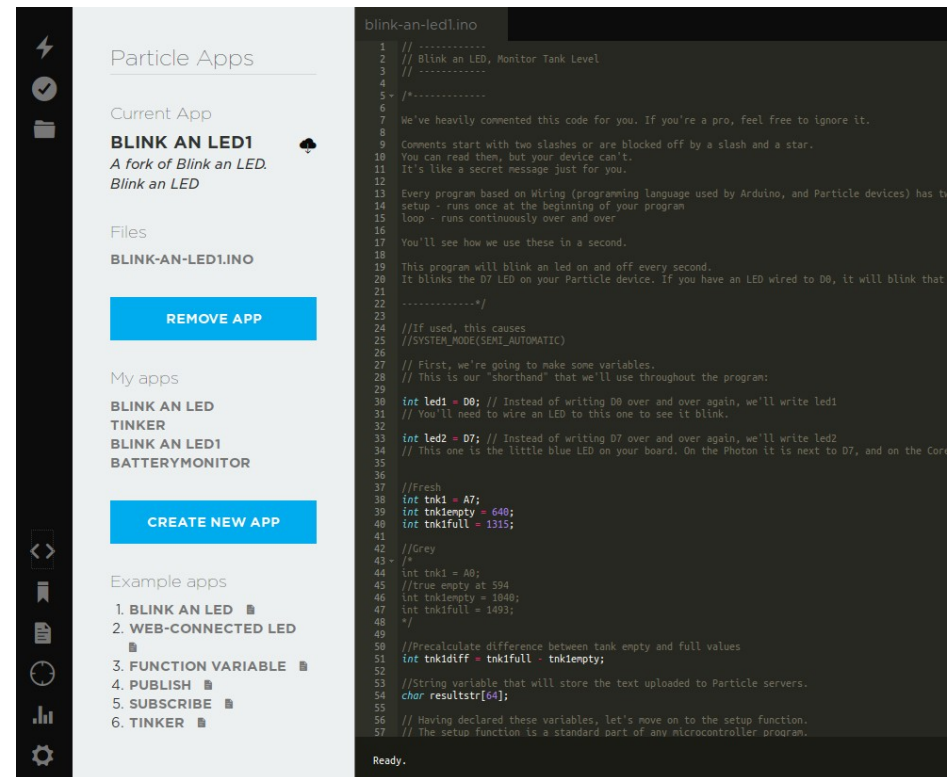


# Installing Photon in Breadboard

- One row of pins on either side of center divider
- MicroUSB connector at end of board
- Power on and cross fingers that Photon “breathes” cyan.

# Flashing Code to Photon

- Go to <http://eastcoasthdtrally.com>. There's a post at the top with a link labeled “Code” that has the firmware we're going to start with. Open this file.
- Open (preferably in another tab) <http://build.particle.io>. You'll need to log in, and should see an interface similar to the image on this slide



The screenshot shows the Particle.io web interface. On the left, there's a sidebar with navigation icons. The main content area is divided into sections: 'Particle Apps' with a 'Current App' section showing 'BLINK AN LED' (A fork of Blink an LED), 'Files' with 'BLINK-AN-LED1.INO', and 'My apps' with a list of apps including 'BLINK AN LED', 'TINKER', 'BLINK AN LED1', and 'BATTERYMONITOR'. There are 'REMOVE APP' and 'CREATE NEW APP' buttons. Below that is an 'Example apps' list. On the right, the source code for 'blink-an-led1.ino' is displayed, showing comments and code for setting up and flashing an LED.

```
blink-an-led1.ino
1 // -----
2 // Blink an LED, Monitor Tank Level
3 // -----
4
5 - /-----
6
7 We've heavily commented this code for you. If you're a pro, feel free to ignore it.
8
9 Comments start with two slashes or are blocked off by a slash and a star.
10 You can read them, but your device can't.
11 It's like a secret message just for you.
12
13 Every program based on Wiring (programming language used by Arduino, and Particle devices) has
14 setup - runs once at the beginning of your program
15 loop - runs continuously over and over
16
17 You'll see how we use these in a second.
18
19 This program will blink an led on and off every second.
20 It blinks the D7 LED on your Particle device. If you have an LED wired to D0, it will blink that
21
22 -----*/
23
24 //If used, this causes
25 //SYSTEM_NODE(SEMI_AUTOMATIC)
26
27 // First, we're going to make some variables.
28 // This is our "shorthand" that we'll use throughout the program:
29
30 int led1 = D0; // Instead of writing D0 over and over again, we'll write led1
31 // You'll need to wire an LED to this one to see it blink.
32
33 int led2 = D7; // Instead of writing D7 over and over again, we'll write led2
34 // This one is the little blue LED on your board. On the Photon it is next to D7, and on the Core
35
36
37 //fresh
38 int tk1 = A7;
39 int tkempty = 640;
40 int tkifull = 1315;
41
42 //Grey
43 - /
44 int tk1 = A0;
45 //true empty at 594
46 int tkempty = 1040;
47 int tkifull = 1493;
48
49
50 //Precalculate difference between tank empty and full values
51 int tkdiff = tkifull - tkempty;
52
53 //String variable that will store the text uploaded to Particle servers.
54 char resultstr[64];
55
56 // Having declared these variables, let's move on to the setup function.
57 // The setup function is a standard part of any microcontroller program.
58
59
60 Ready.
```

# Installing Variable Resistor

- Resistor should be oriented so that each of the three pins land on different breadboard rows.
- Connect one of the outer pins to 3.3V power, the other to ground.
- Third pin goes to an analog input on the Photon (A7 for now)





# Publishing Data

- Uncomment code block between lines 92 and 98
- Re-flash firmware
- Go to <http://dashboard.particle.io> and navigate to logs