Photon Digital IO -- Input (and Debugging :-)

44-440/640-IoT

Objectives

- Students will be able to:
    - explain the purpose of digitalRead(), and incorporate it in code
    - deploy a switch
    - explain the purpose and behavior of pullup resistors
    - debug firmware using the power of Serial debugging

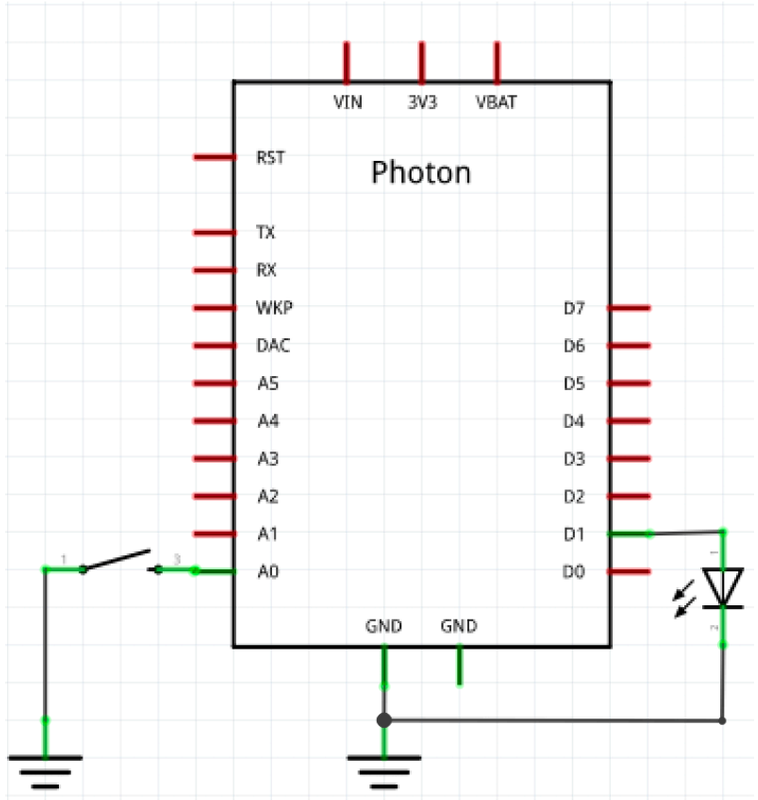
Digital Input

- The easiest component for digital input is a switch. We hook the switch to a pin set to a pin mode of **INPUT_PULLUP** (we'll explain why shortly) and then run it to ground.

- We detect switch presses with **digitalRead(*pin*)**.

- digitalRead() will return **LOW** if the button is pressed, **HIGH** if it is not

- Mnemonic: when you press a button, it moves LOWer 😍



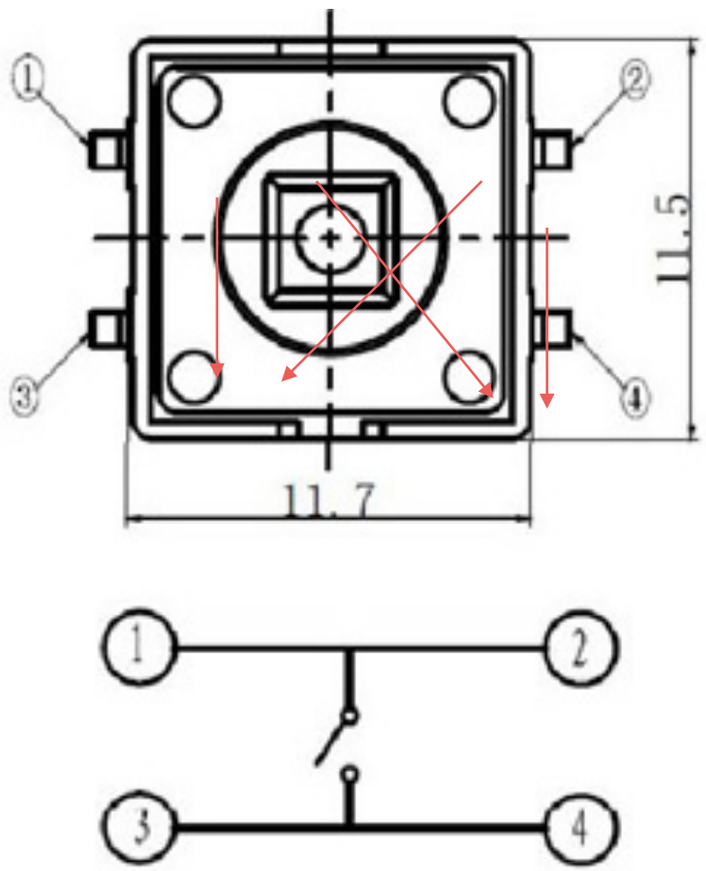

```
int switchy = A0;
void setup() {
   pinMode(switchy,INPUT_PULLUP);
   pinMode(leddy, OUTPUT);
   Serial.begin(9600);
}

void loop() {
    if(digitalRead(switchy)==LOW){
         Serial.printf("I am imPRESSed!\n");
     }
}
```

Digital Input

- Our switch is a push button switch, i.e., a button

- When depressed, the pins on both sides connect (and cross connect). For simplicity, just use pins 1 and 3 or 2 and 4

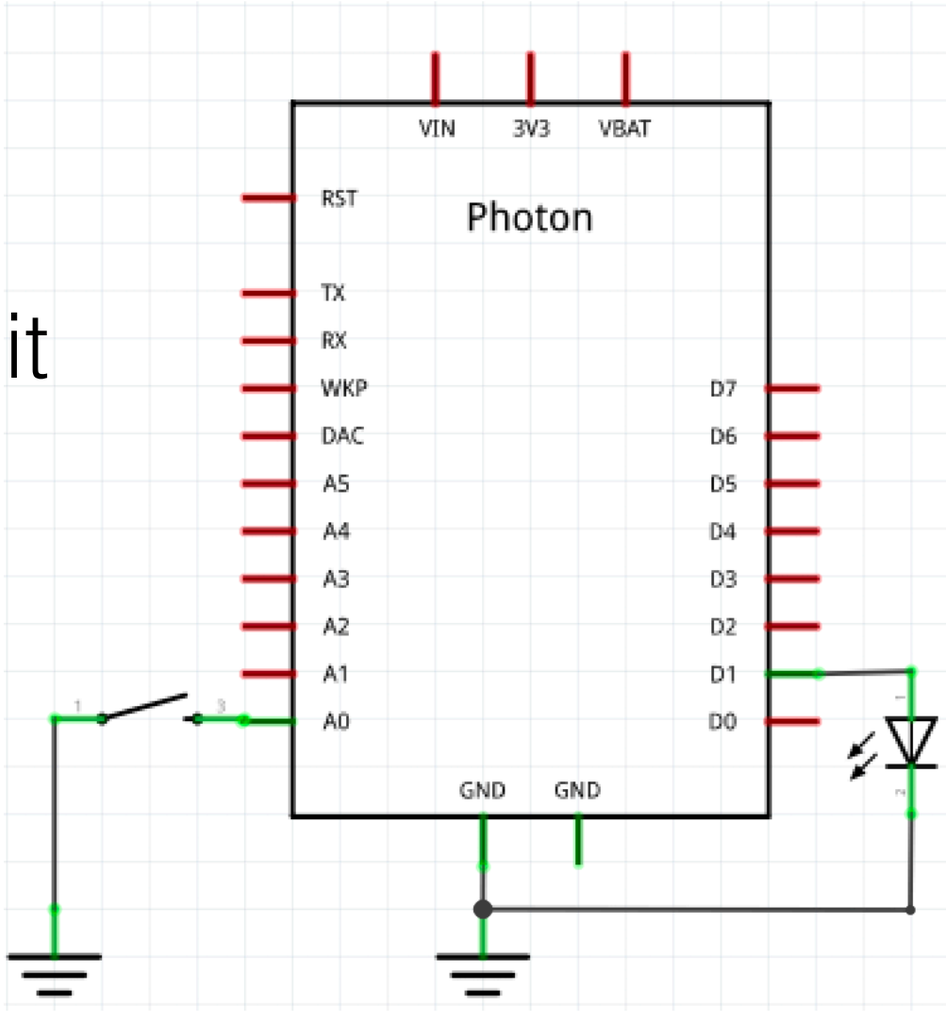
- Mnemonic: "switch on the side"



- Pins 1 & 2 are always connected, so are 3 & 4.
- Closing the switch connects 1-3 (and 4) and 2-4 (and 3)

Pull-up Resistors

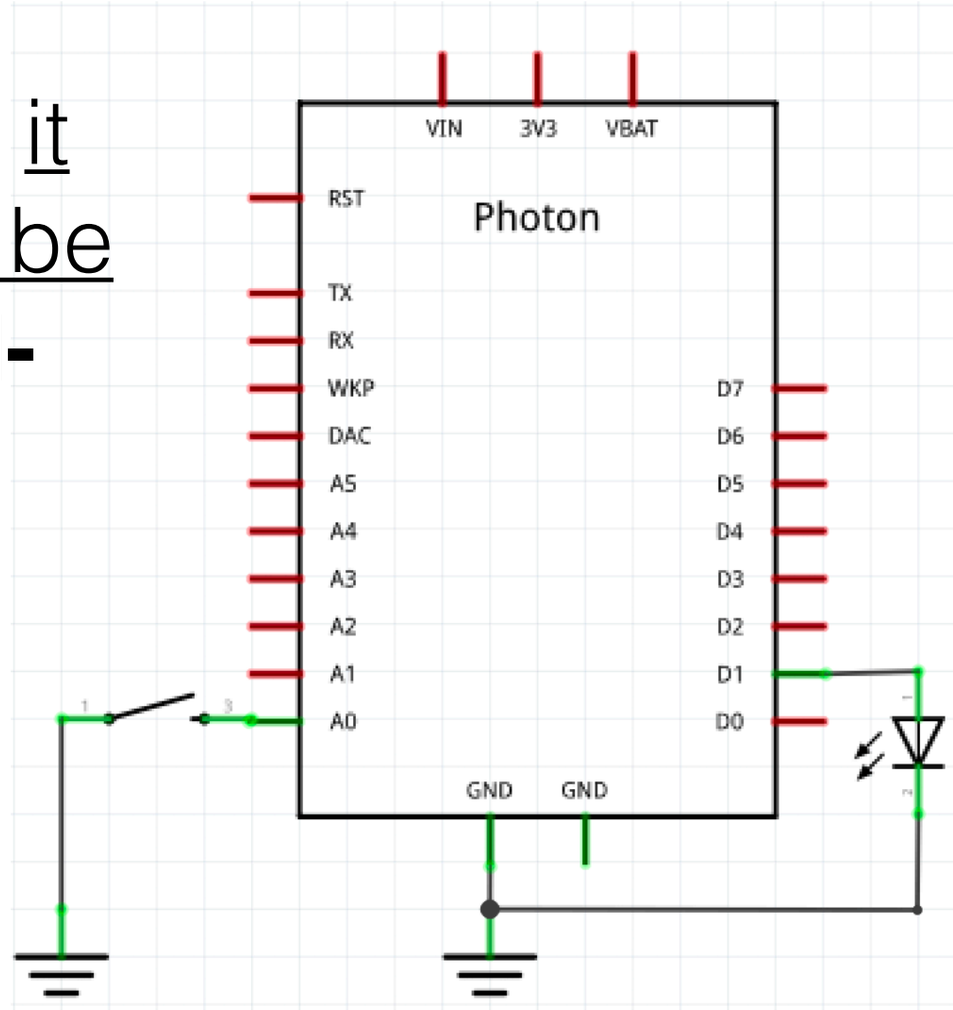
- Q: When reading from a pin configured as input, what should it read if nothing is connected? 🧐



Nothing's connected to A0 at present, as the switch is open. What should it read?

Pull-up Resistors

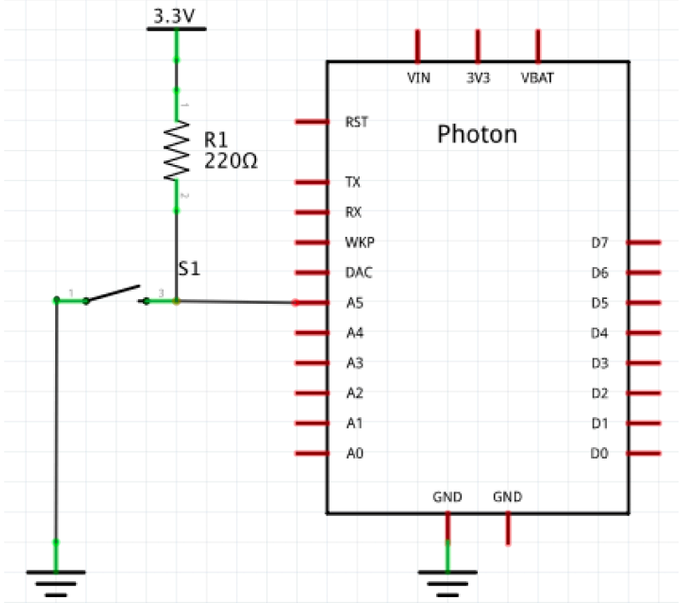
- A: You might think it will be 0, but <u>it depends, and cannot accurately be predicted</u> . The purpose of a **pull-up resistor** is to *remove that ambiguity*.




```
int switchy = A0; // A0-A7, D0-D7 can be used for digitalRead()
int leddy = D1;
void setup() {
   pinMode(switchy,INPUT_PULLUP);
   pinMode(leddy, OUTPUT);
   Serial.begin(9600); // enables debugging
}

void loop() {
    if(digitalRead(switchy)==LOW){
         digitalWrite(leddy, HIGH);
    } else {
         digitalWrite(leddy, LOW);
    }
}
```

Nothing's connected to A0 at present, as the switch is open. What should it read?

Pull-up Resistors



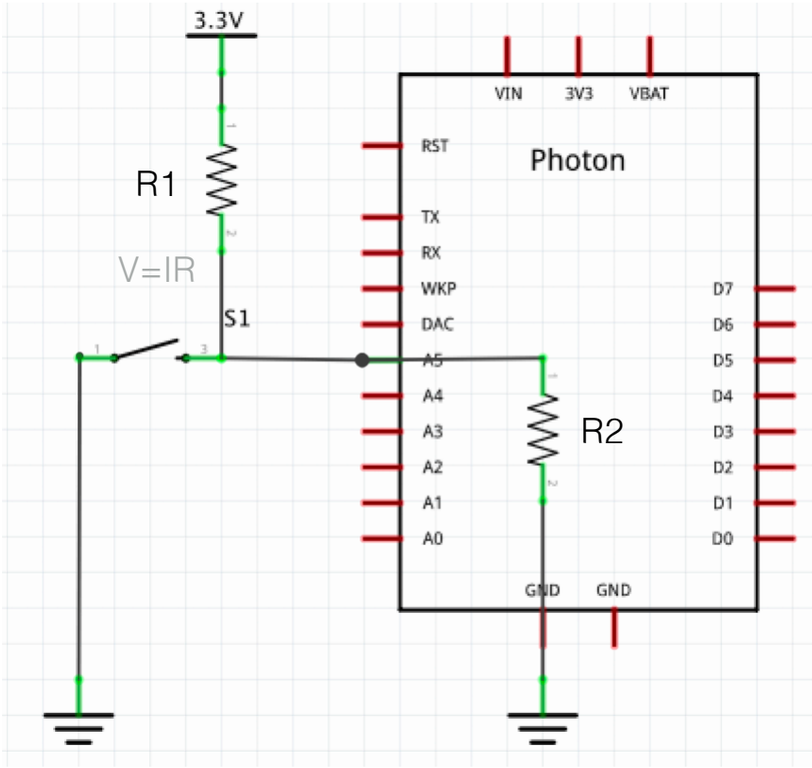
- In the diagram at right, R1 is a **pull-up resistor**. We wish to choose it so that
    - when the switch is open, the input pin reads HIGH (*close* to VCC = 3.3V). This implies a *small* voltage drop across R1, implying a small current (since V = IR1)
    - when the switch is closed, the current will preferentially go through the button to ground, voltage at the pin will be *almost* 0, and so digitalRead() will return LOW.
- Why does current flow **into** the input pin? Because beyond the input pin, which has a certain resistance indicated by R2, is *ground*, and conventional current flow states that charge flows from VCC to ground.

Pull-Up Resistors — Picking the Resistance

- What resistance should R1 and R2 be? R2 is fixed by the manufacturer, so the question really is what should R1 be? We want it so that:

  a) when the switch is **closed**, voltage at the input pin will drop to (almost?) 0, and R1 controls how much current will flow to GND. Therefore, **R1 cannot be too small, otherwise we would have excessive current**.

  b) when the switch is **open**, R1 controls the voltage on the input pin (there will be a voltage drop across R1: the voltage at the input pin is Vcc - the voltage drop). Therefore **it cannot be too large, otherwise there will be hardly any voltage**, making detecting the closed switch difficult.



- To satisfy b), a rule-of-thumb is to make R1 1/10 that of R2.

- Let's try R1 = 100kΩ. What happens? When the switch is closed, the current to ground is 3.3V/100kΩ = 0.03 mA

- When the switch is open, well, this is another **voltage divider circuit**, and we can see that the voltage at the input pin would be 3V, definitely high enough.

RT = 100kΩ + 1MΩ = 1.1MΩ
I = 3.3/1.1MΩ = 3x10-6 A

At R1, V = IR ==> V = 3x10^-6 * 100x10^3 = 300x10-3 V = 0.3V
At R2, V = IR ==> V = 3x10^-6 * 1x10^6 = 3V

So the voltage at the input pin would be 3.3-0.3 = 3V, definitely HIGH.

Close Enough …

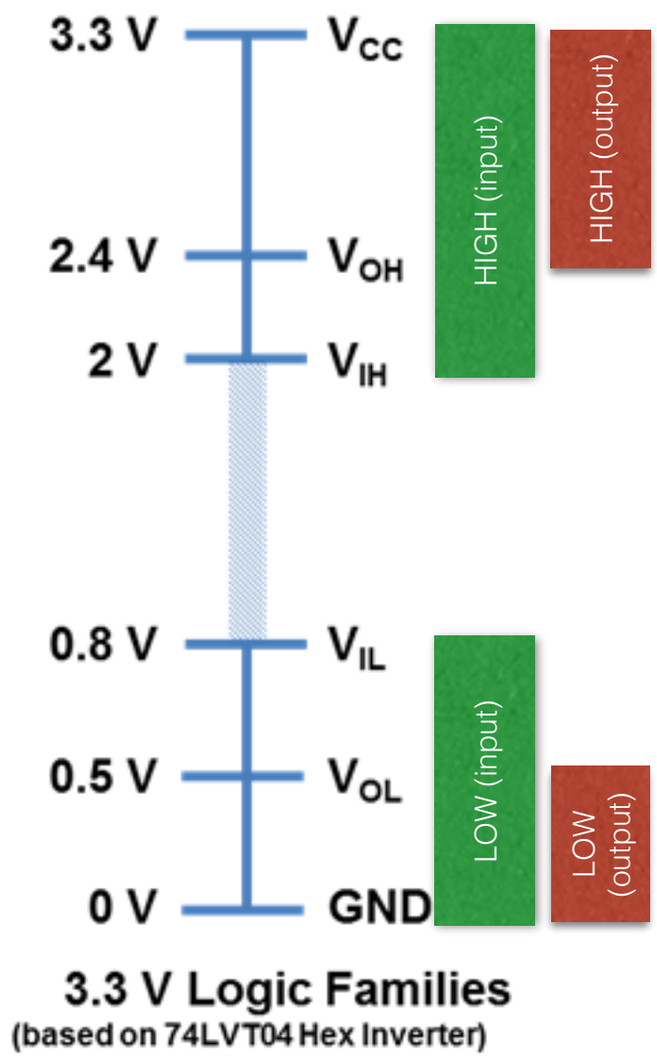
- Because of the imprecision of electronic components, we cannot simply say a digital value of HIGH == 3.3V, and a digital value of LOW == 0V. Instead, we need to define intervals:

- V_{OH} = Minimum output voltage that will be provided when pin is set to HIGH

- V_{IH} = Minimum input voltage that will be construed as HIGH

- V_{OL} = Maximum output voltage that will be supplied when pin is set to LOW

- V_{IL} = Maximum input voltage that will be construed as LOW



Pull-Up Resistors on the Photon

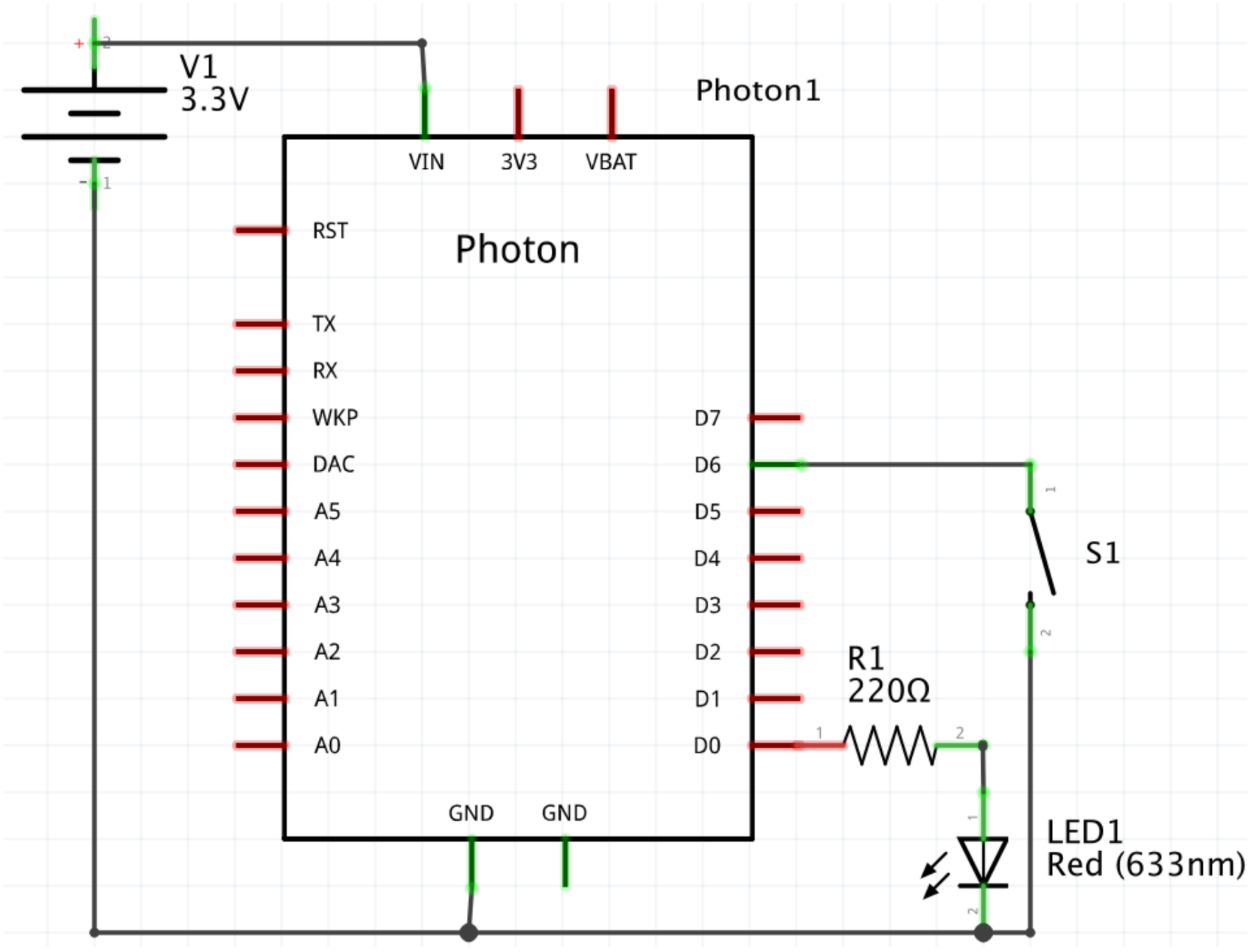
- On the Photon, 40K pull-up resistors are built-in, and programmatically enabled by writing, in setup,

```
pinMode(pin, INPUT_PULLUP)
```

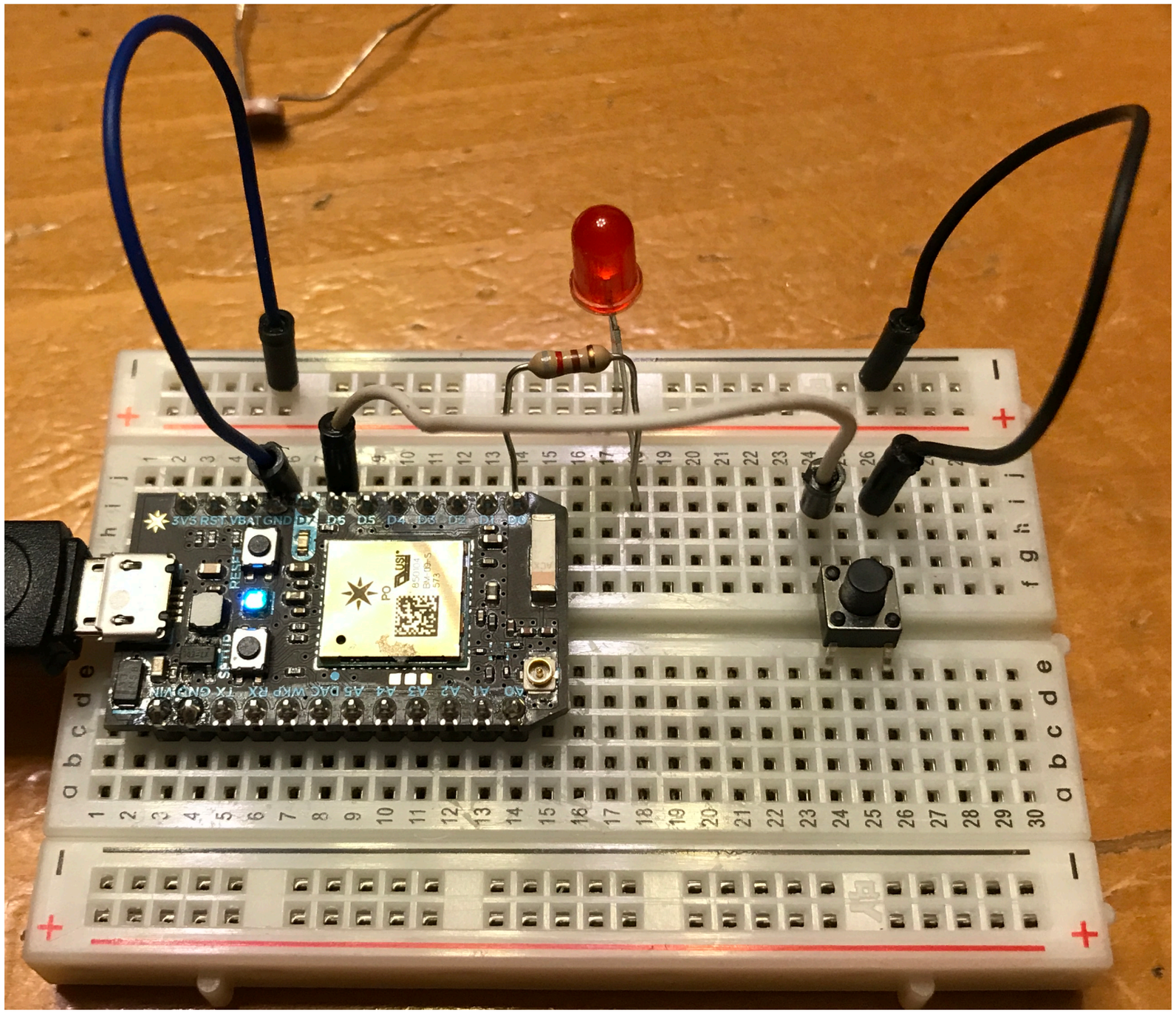
- If for some reason 40K isn't large enough, you could add another resistor leading to the input pin

- Use INPUT instead of INPUT_PULLUP to disable the pull-up resistors.

Hook 'em Up!: Switches



ICE: Breadboarding the Circuit



Handling Switches in Code

```
// Name: simple-switch.ino
// Author: MP Rogers
// Description: demonstrates how to use a switch

int switchy = D6;
int led = D0;

void setup() {
    pinMode(switchy,INPUT_PULLUP);
    pinMode(led,OUTPUT);
    Serial.begin(9600); // enables debugging
}

void loop() {
    if(digitalRead(switchy)==LOW){ // button pressed
       digitalWrite(led,HIGH); // turn on both LEDs
       Serial.println("Switch pressed"); // how many times should this
print?
    } else {
       digitalWrite(led,LOW);
    // Serial.println("Switch released"); // how about this?
   }
}
```

Printing both Switch pressed and Switch released makes it almost impossible to see when Switch pressed is called, hence the //

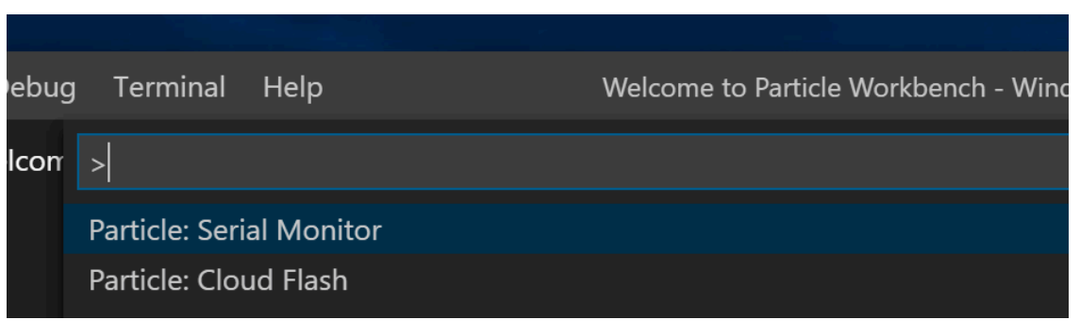
Debugging Code

- When connected to USB, text from the Photon can be sent, over that serial connection, to the computer.

- **Code**: Use Serial.begin() to set the com speed, and Serial.println() or Serial.printf() to send data

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    Serial.println("Marco");
    delay(500);
    digitalWrite(yellowLED,LOW);
    delay(500);
}
```

- **Visual Studio Code**: Select Particle: Serial Monitor from the command palette



ICE: Time, Ladies & Gents!

- Research the **Time** class (on docs.particle.io, in reference) and when the button is pressed, modify the above code to print:

- the time (in ms)

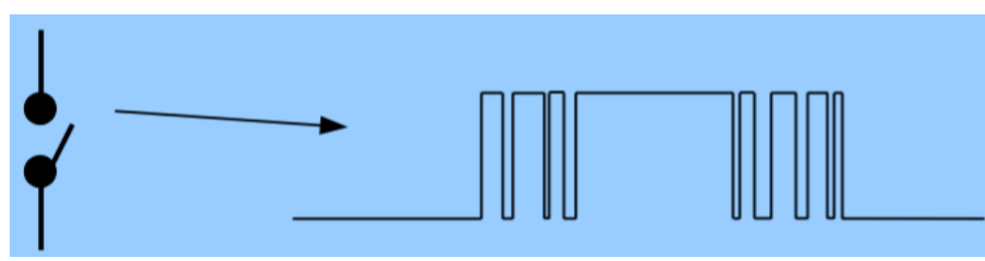
- the date (in the format day/month/year)

Fast Loops & Bouncing Switches

- The keen observer will have noticed that when you press the switch, "Switch pressed" appears multiple times in the console. There are two reasons for this:

  - **fast MCUs** -- the hand may be quicker than the eye, but the MCU is faster than both, so the loop() will be invoked several times before the user can remove their finger, invoking Serial.println() multiple times.

  - **bouncing switches** -- when a button is depressed, due to mechanical imperfections it does not close cleanly. Instead, it repeatedly closes and opens, within a few milliseconds before finally stabilizing



Solving the Debouncing Dilemma

- In the previous circuit, debouncing is not a problem, but most of the time we wish a button push to be detected just once (we will see examples of this, involving publishing, in a day or two).

- Strategy: When a closed switch is detected, see if the last closure happened more than a few nanoseconds ago — if so, deem it a separate event.

A Little Code

```
int timeButtonWasLastPressed;
int switchy = A0;
int led = D1;

void setup() {
    pinMode(switchy,INPUT_PULLUP);
    pinMode(led,OUTPUT);
    Serial.begin(9600);
    Particle.subscribe("Testing-Meh",handleMeh);
    timeButtonWasLastPressed = Time.now();
}
```

What does Time.now() do? Consult the docs - docs.particle.io

```
void loop() {
    // button pressed, and it's been at least 1 second since it was last pressed
    if(digitalRead(switchy)==LOW && Time.now() - timeButtonWasLastPressed >= 1){
        digitalWrite(led,HIGH);
        Particle.publish("Testing-Meh",String(Time.now()));
        timeButtonWasLastPressed = Time.now();
   } else {
        digitalWrite(led,LOW);
    }
}

void handleMeh(String eventName, String data){
    Serial.println(eventName);
    Serial.println(data);
}
```

We will explain the subscribe() and publish() mechanism in a few days, but the important point here is that when you push the button, publish() is only called once (unless you hold the button down for more than 1 second)

Resources

- https://www.sparkfun.com/tutorials/219
- https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds
- http://www.electronics-tutorials.ws/diode/diode_8.html
- https://learn.sparkfun.com/tutorials/pull-up-resistors
- http://apple.stackexchange.com/questions/124152/how-do-you-paste-syntax-highlighted-code-into-keynote-13
- http://markup.su/highlighter/
- http://www.cc.gatech.edu/~hadi/teaching/cs3220/02-2015fa/doc/debounce.pdf
- https://learn.adafruit.com/all-about-leds/forward-voltage-and-kvl
- http://dangerousprototypes.com/docs/Basic_Light_Emitting_Diode_guide
- https://www.baldengineer.com/led-basics.html
- https://learn.sparkfun.com/tutorials/logic-levels/ttl-logic-levels