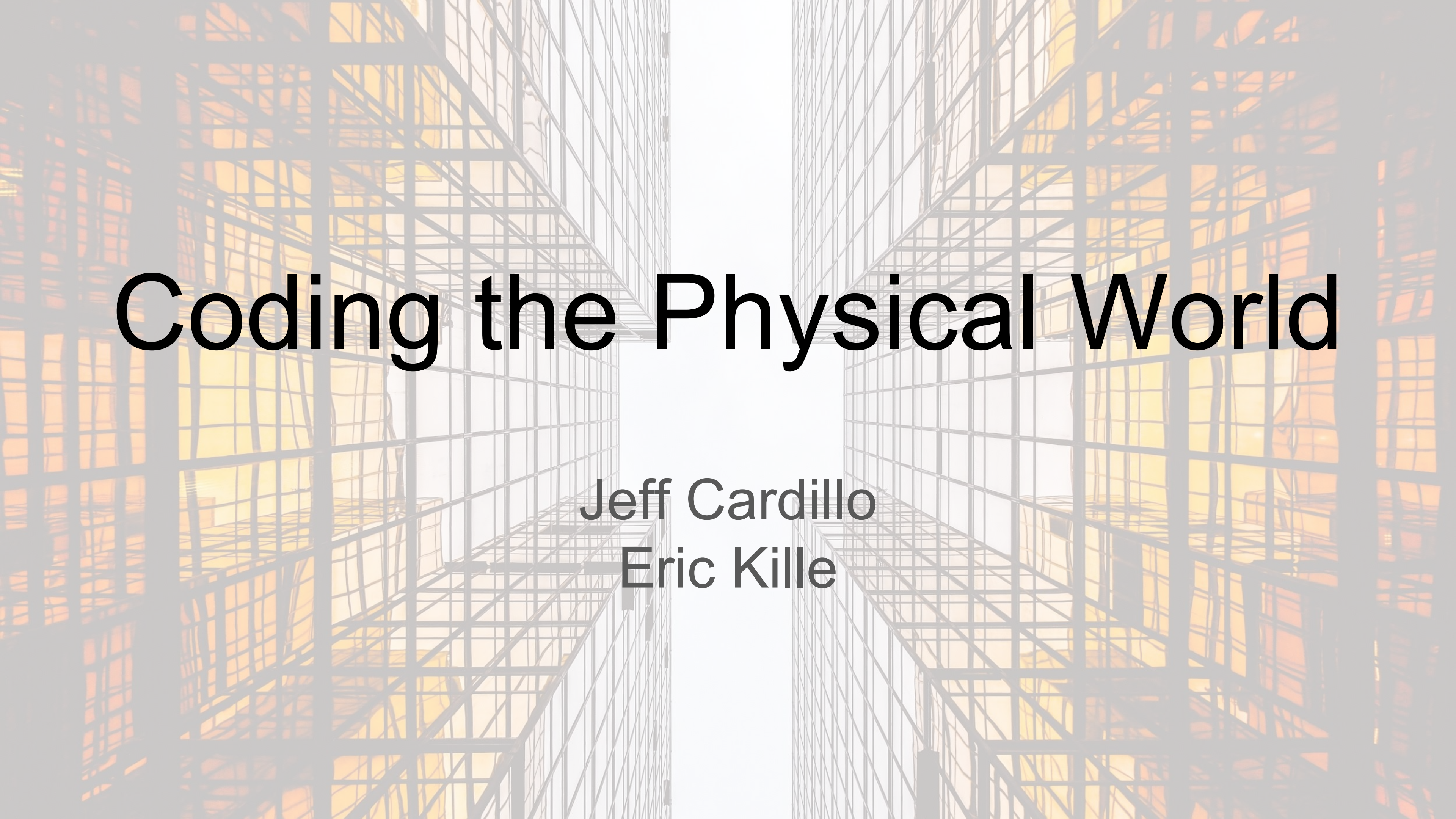


# Coding the Physical World

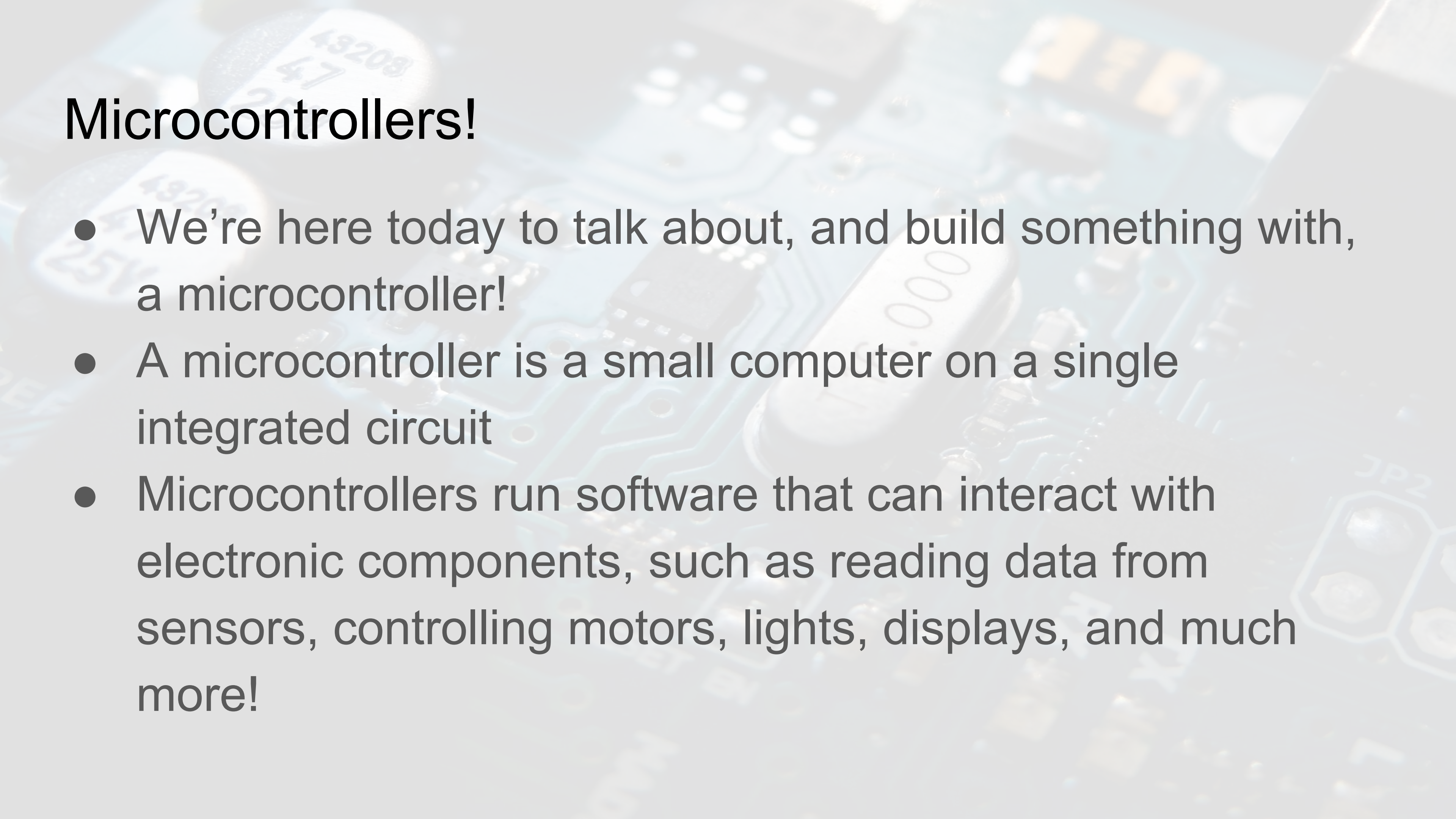
Jeff Cardillo
Eric Kille

## Useful links

Sample code, circuit schematic, and this presentation:
http://github.com/jeffcardillo/HopeworksCodeDay

Particle IO’s Website
https://www.particle.io/

Contact Us
jeffrey.cardillo@gmail.com
ekille93@gmail.com

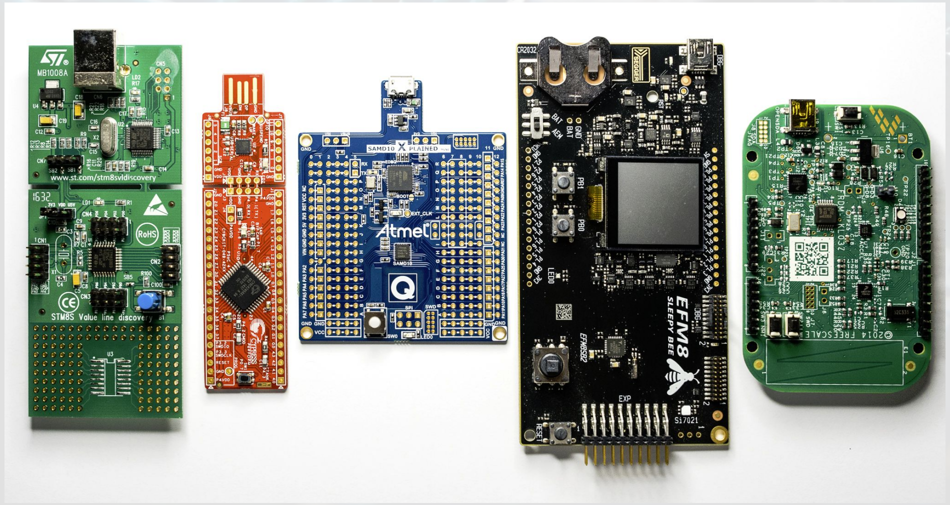


## Microcontrollers!

- We’re here today to talk about, and build something with, a microcontroller!
- A microcontroller is a small computer on a single integrated circuit
- Microcontrollers run software that can interact with electronic components, such as reading data from sensors, controlling motors, lights, displays, and much more!

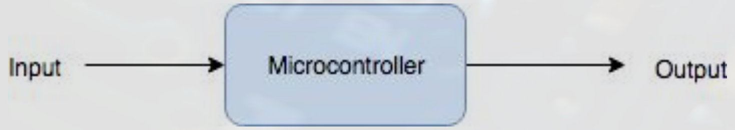
## Microcontrollers are everywhere!

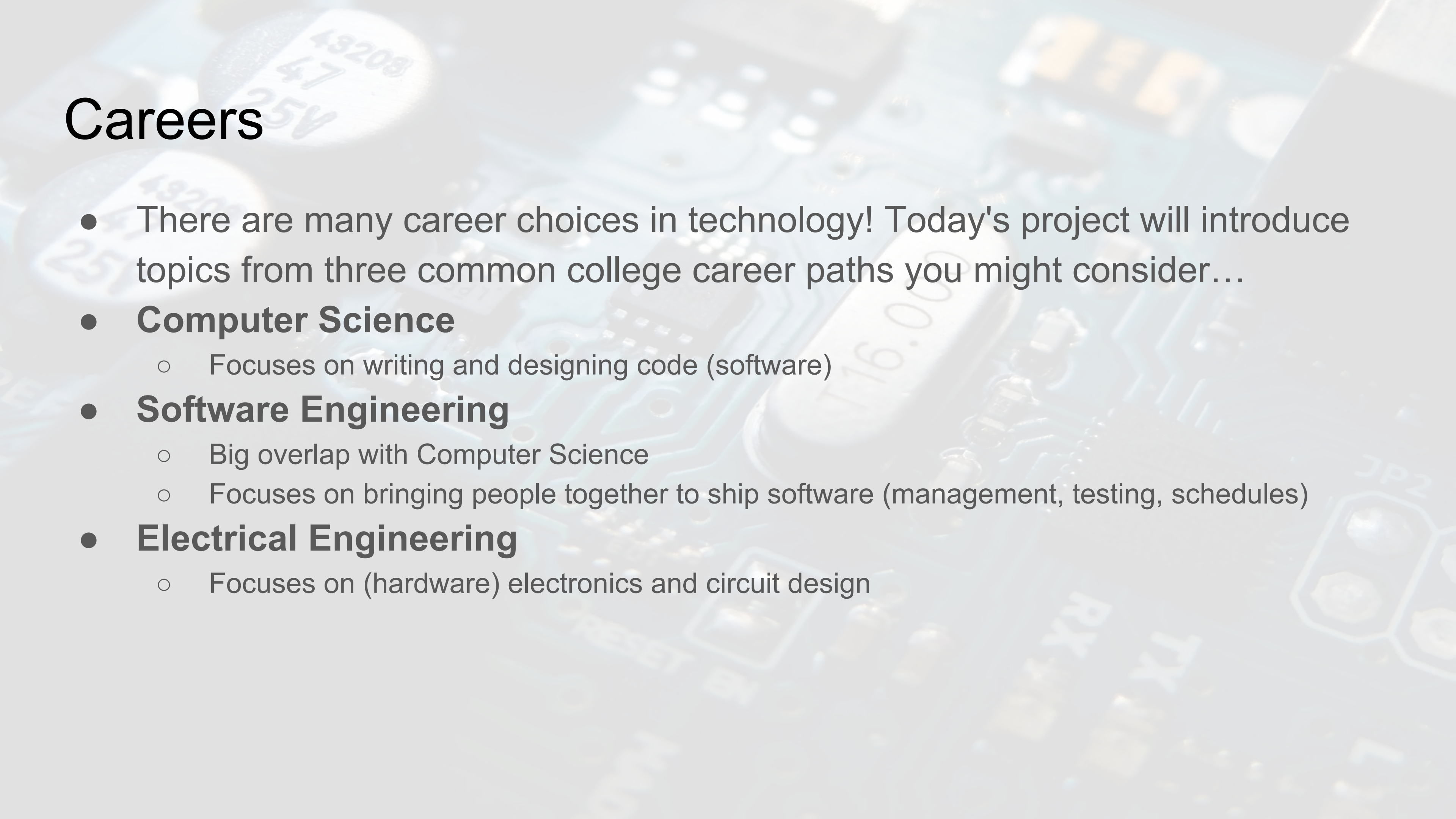
- Common places you'll find microcontrollers are gaming controllers, automobiles, remote controls, cameras, toys, etc.



# Microcontrollers are typically single task

- Microcontrollers differ from something like a smartphone in that a smartphone is more like a general computer capable of running many different programs (apps)
- A microcontroller typically has a single software program that it runs continuously
- The software program on a microcontroller is often called firmware because it doesn’t change once set. Though, the microcontroller can often be “flashed” or updated with newer firmware



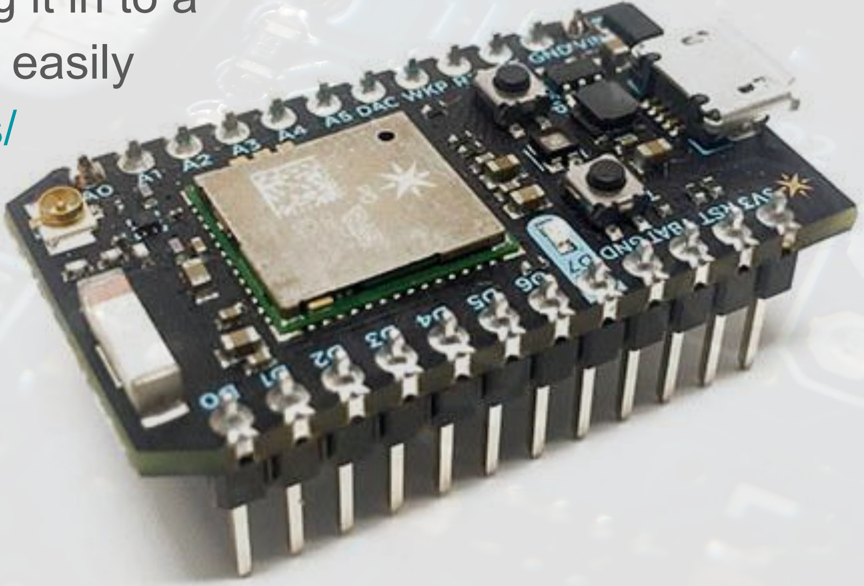


## Careers

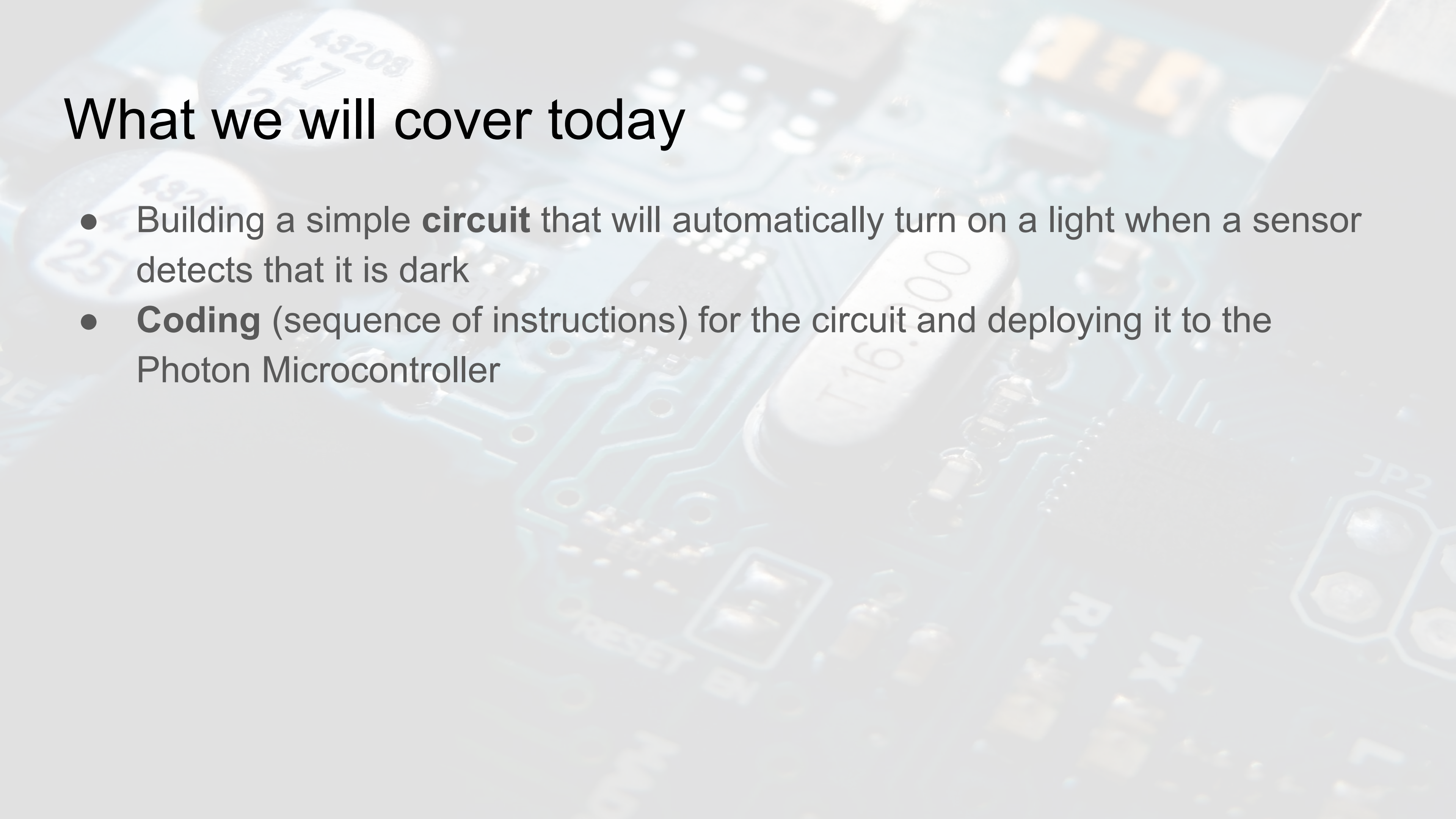
- There are many career choices in technology! Today's project will introduce topics from three common college career paths you might consider…
- **Computer Science**
  - Focuses on writing and designing code (software)
- **Software Engineering**
  - Big overlap with Computer Science
  - Focuses on bringing people together to ship software (management, testing, schedules)
- **Electrical Engineering**
  - Focuses on (hardware) electronics and circuit design

## Introducing the Particle Photon

- The Photon is a powerful, yet low cost hobbyist microcontroller
- It is easy to code software and push it to the Photon
- Photon boards are programmed using a C style language
- The Photon’s header pins allow us to plug it in to a breadboard and build simple circuits very easily
- More info https://www.particle.io/products/hardware/photon-wifi/



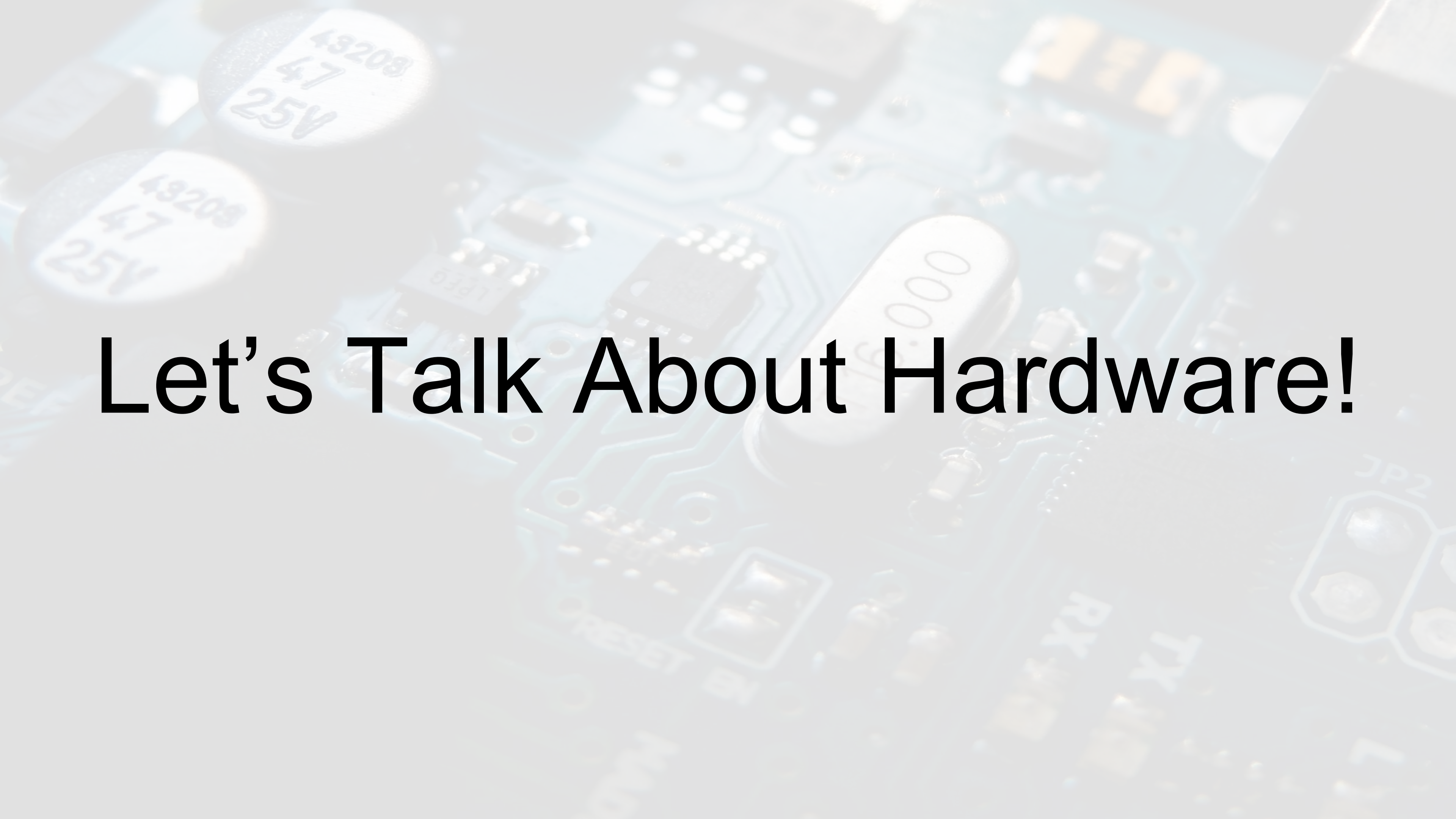
## What we will cover today



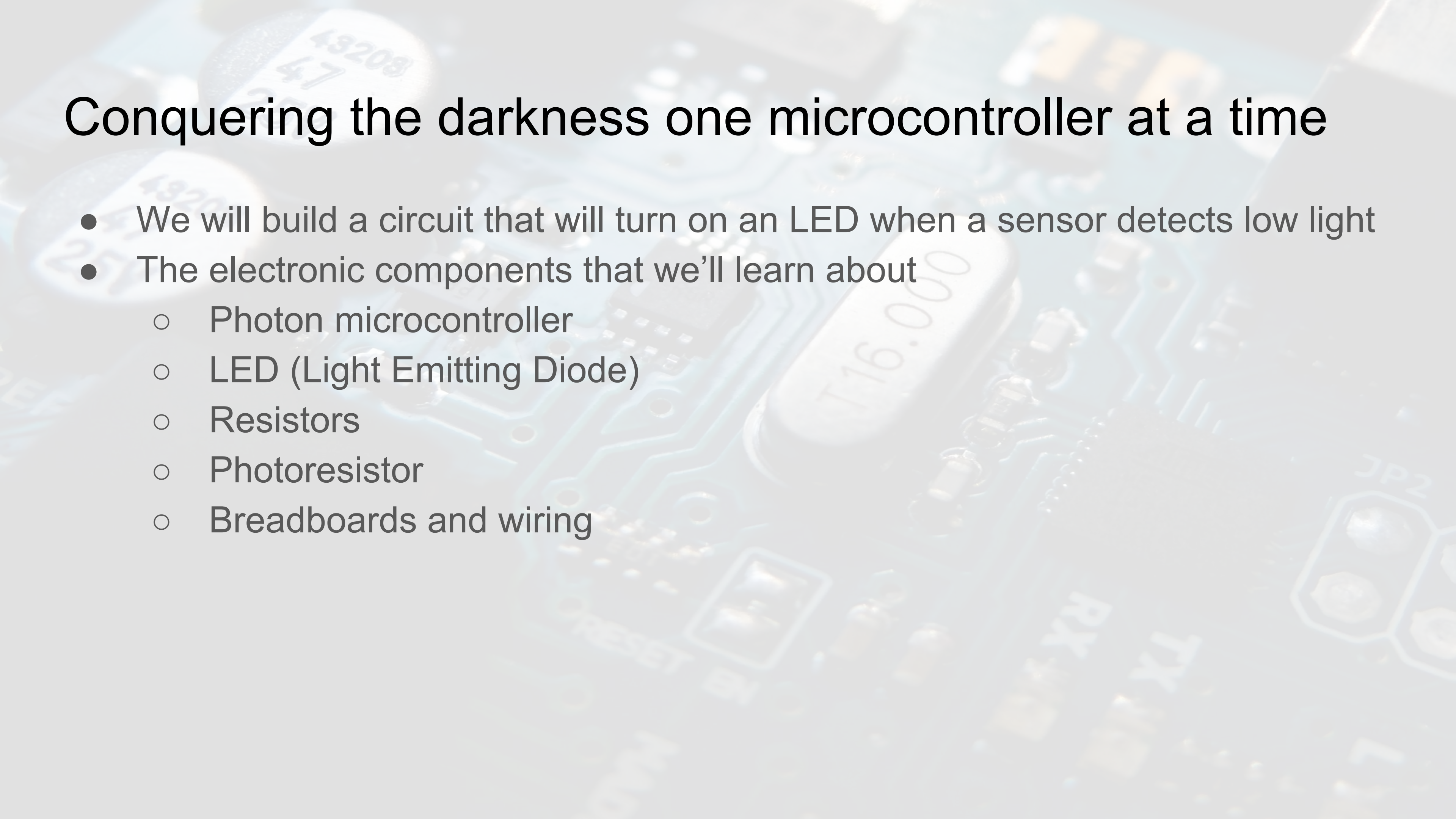
- Building a simple **circuit** that will automatically turn on a light when a sensor detects that it is dark
- **Coding** (sequence of instructions) for the circuit and deploying it to the Photon Microcontroller

## Let’s break out in to groups

- We have lab kits for up to 6 groups
- Each group should have at least one member that has a computer



## Let’s Talk About Hardware!

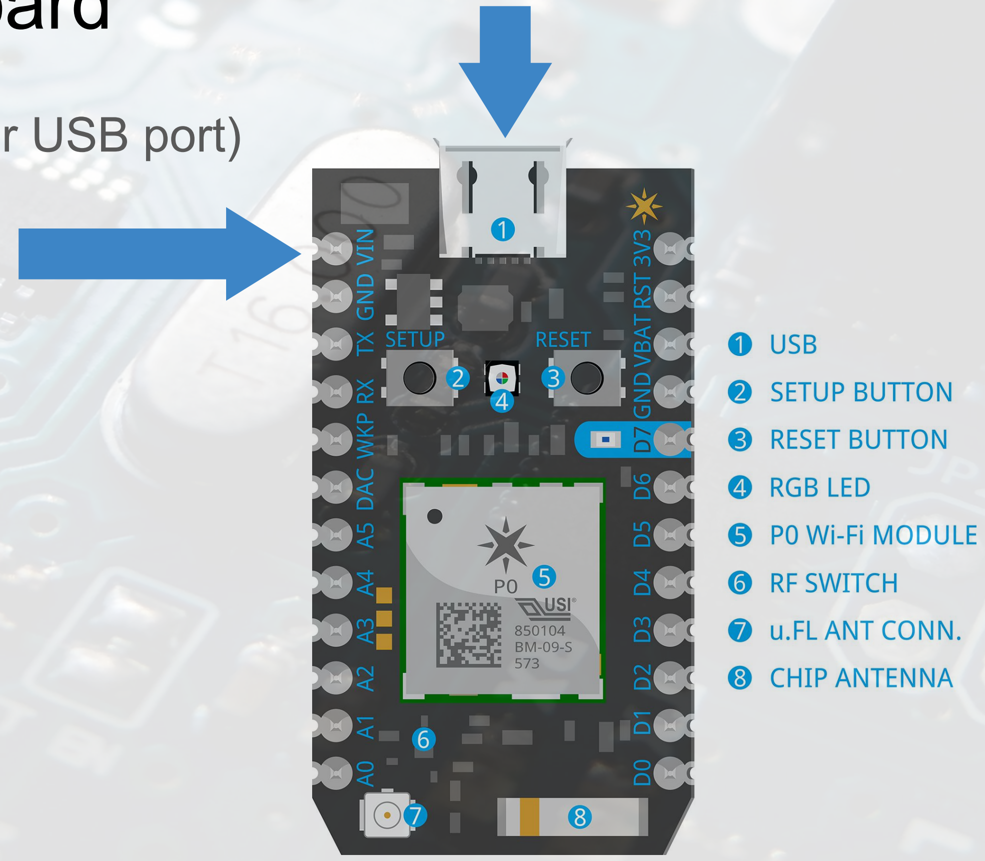


## Conquering the darkness one microcontroller at a time

- We will build a circuit that will turn on an LED when a sensor detects low light
- The electronic components that we’ll learn about
  - Photon microcontroller
  - LED (Light Emitting Diode)
  - Resistors
  - Photoresistor
  - Breadboards and wiring

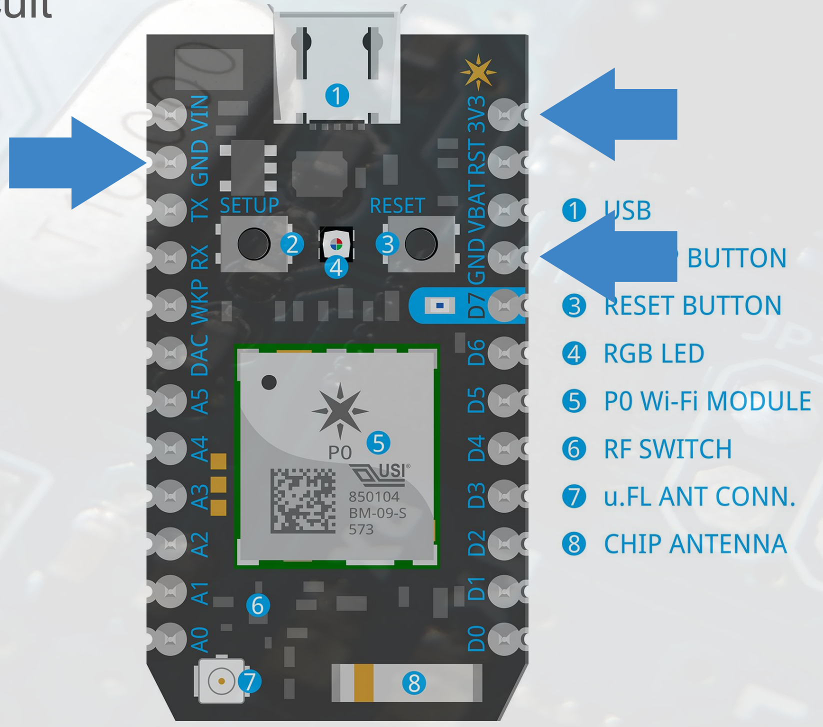
# Anatomy of a Photon board

- Board gets power through VIN (or USB port)



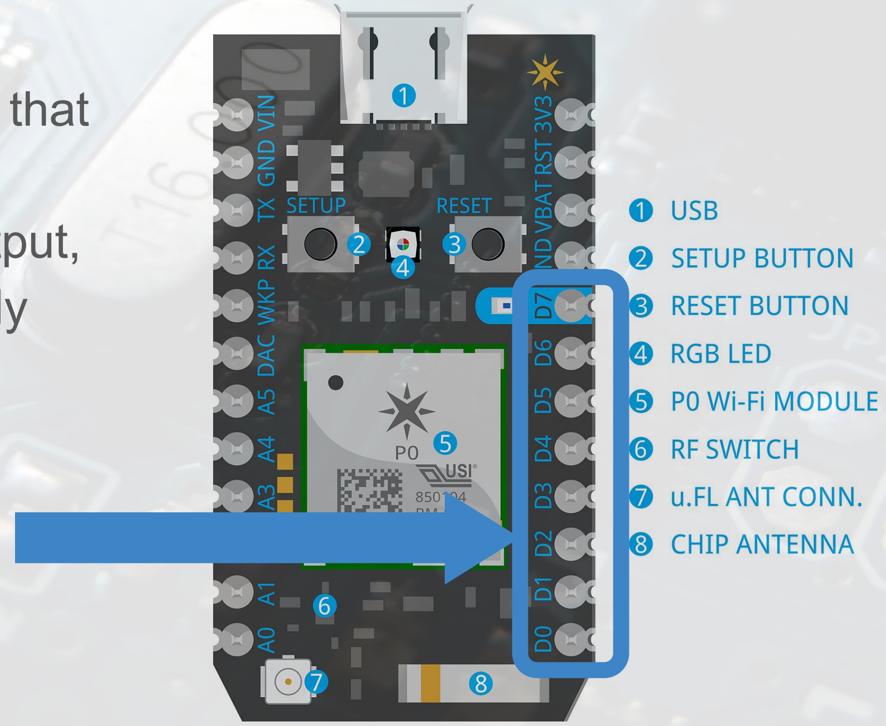
# Anatomy of a Photon board

- Board provides power to breadboard circuit through 3V3 / GND pins



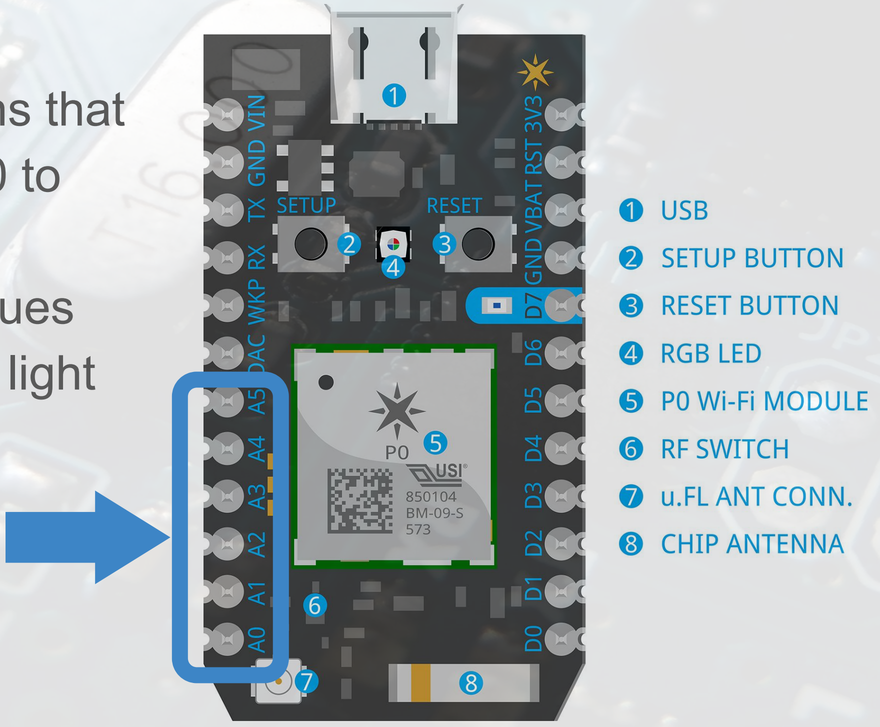
# Anatomy of a Photon board

- Pins named with a “D” are digital pins that are binary, ON or OFF
- These are used for both input and output, but we will be using one for output only

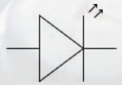


# Anatomy of a Photon board

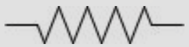
- Pins named with “A” are for analog pins that provide varying values (ranging from 0 to 1023 different values)
- Analog pins are used to read input values from sensors. We will use one to read light levels



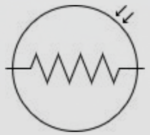
# LED (Light Emitting Diode)

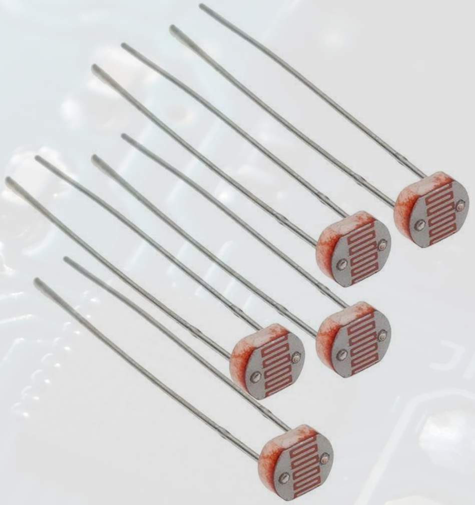
- Diodes are a component that allow electrons to flow through in only one direction, like a one way street!
- LEDs are a special kind of Diode that emits light as electrons pass through
- The schematic symbol for an LED is: 
- One wire is longer than the other denoting the + lead. The LED needs to be placed in the circuit in the correct direction

## Resistors

- Resistors “resist” electrical current (limits the flow of electrons)
- We will use one resistor to limit electrical flow through the LED, and another to create a voltage divider
- The schematic diagram of a resistor looks like a squiggly line: 
- Resistors have bands of color on them to tell you their resistance (chart in Appendix A)

## Photoresistor

- Photoresistors are a special kind of resistor whose resistance changes with the amount of light hitting the sensor
- The schematic diagram for a photoresistor appears as: 
- Resistors (and photoresistors) do not have to be placed in a certain direction

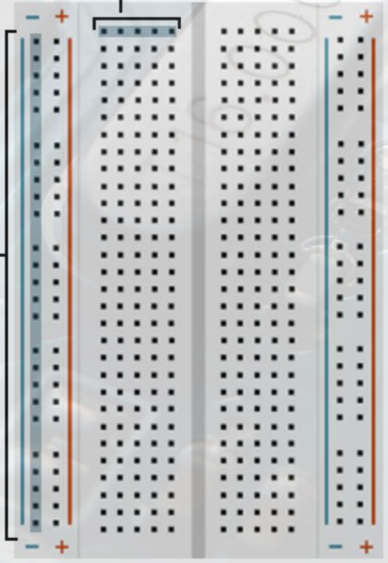


# Breadboards

The 5 holes in each horizontal row are connected electrically through metal strips inside the breadboard.

The middle row breaks the connection between the two sides of the board.

The vertical strips that run the length of the breadboard are electrically connected. The strips are usually used for power and ground connections.



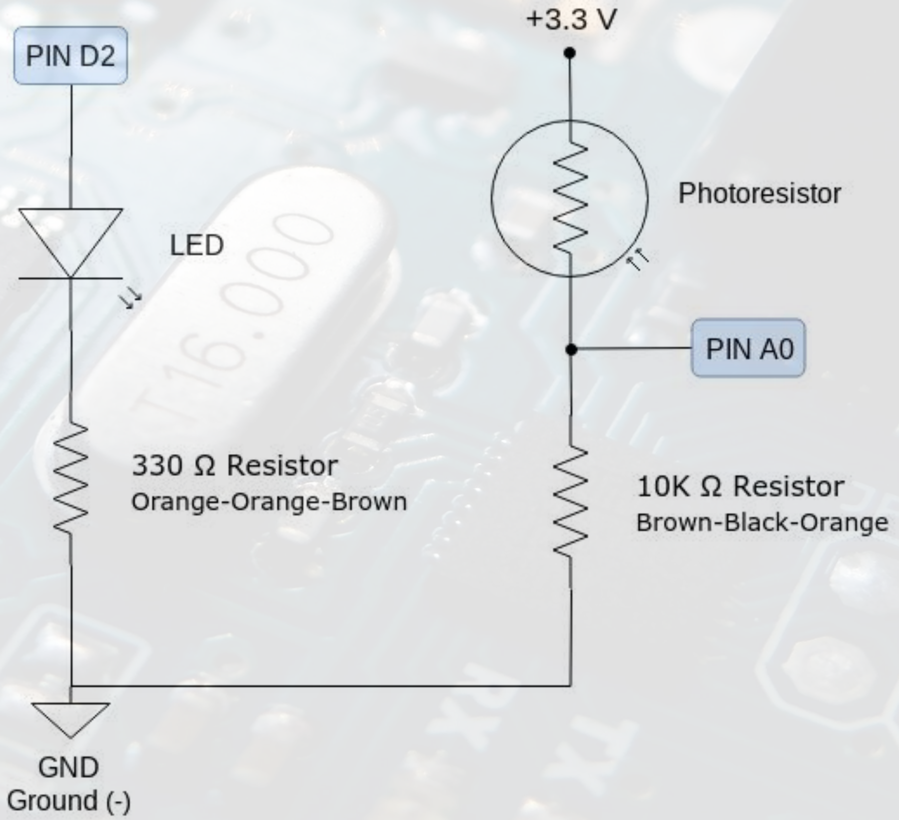
POWER BUS

POWER BUS

PROTOTYPING AREA

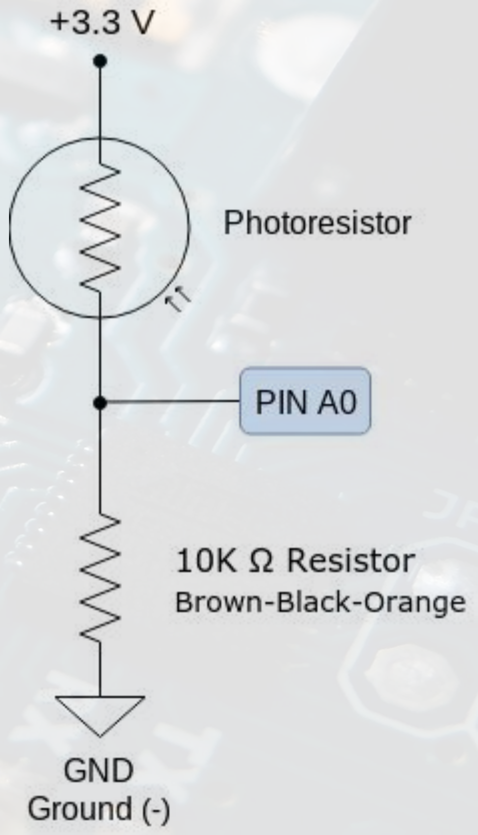
## Circuit Overview

- PIN D2 and A0 will connect to the Photon board
- +3.3 Volts and GND will also connect to the Photon Board
- “D” stands for “Digital” and “A” stands for “Analog”



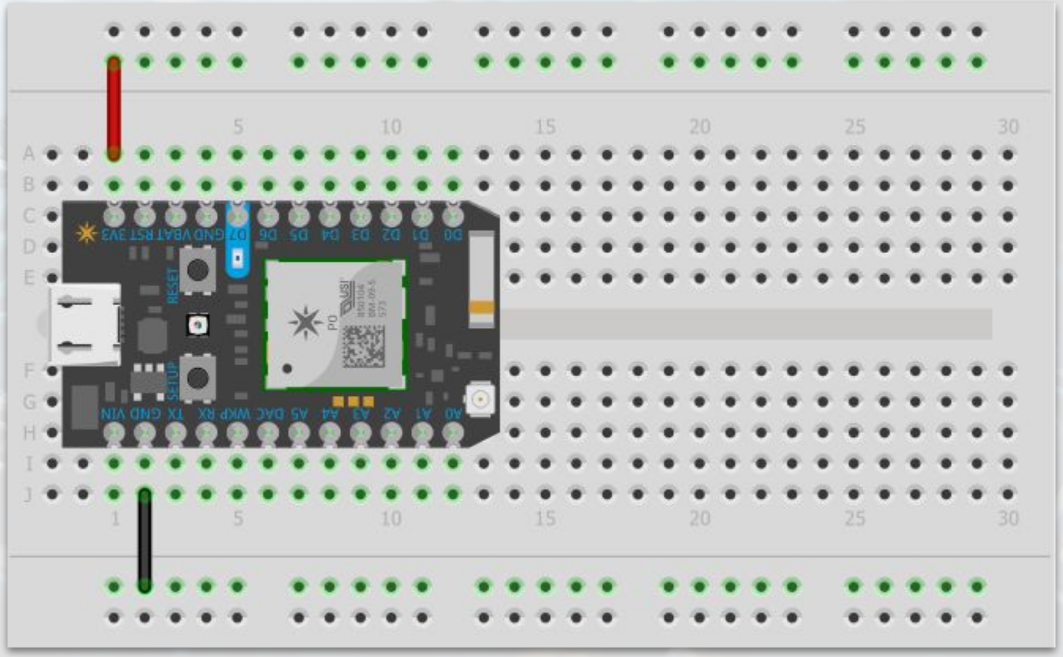
## Voltage Divider

- A Photoresistor modifies current by changing resistance, but the Photon cannot read variations in current. It can, however, read variations in voltage
- The voltage divider circuit (shown to the right) will cause the voltage at PIN A0 to fluctuate depending on the resistance of the Photoresistor
- The voltage variation is caused by relative voltage drops across the resistor components
- More details in Appendix B



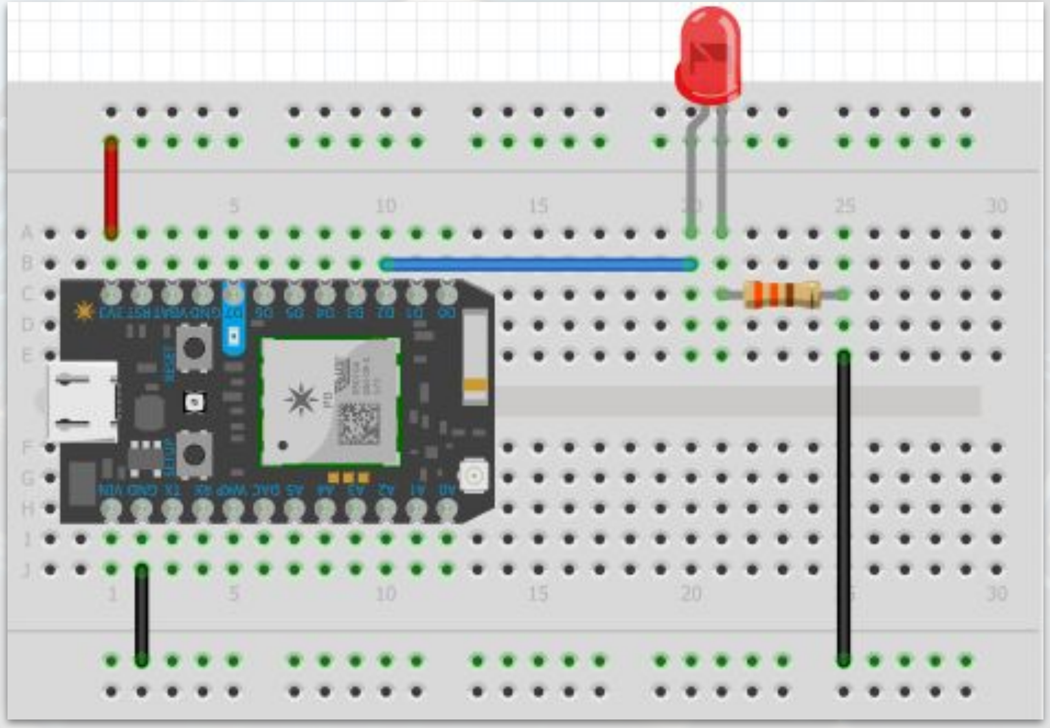
## Wiring the circuit - position the Photon Board

- Place the Photon on the breadboard with the USB port toward one side
- Place a red wire from 3v3 to the top bus
- Place a black wire at the bottom GND pin to a bottom bus



# Wiring the circuit - adding the LED circuit

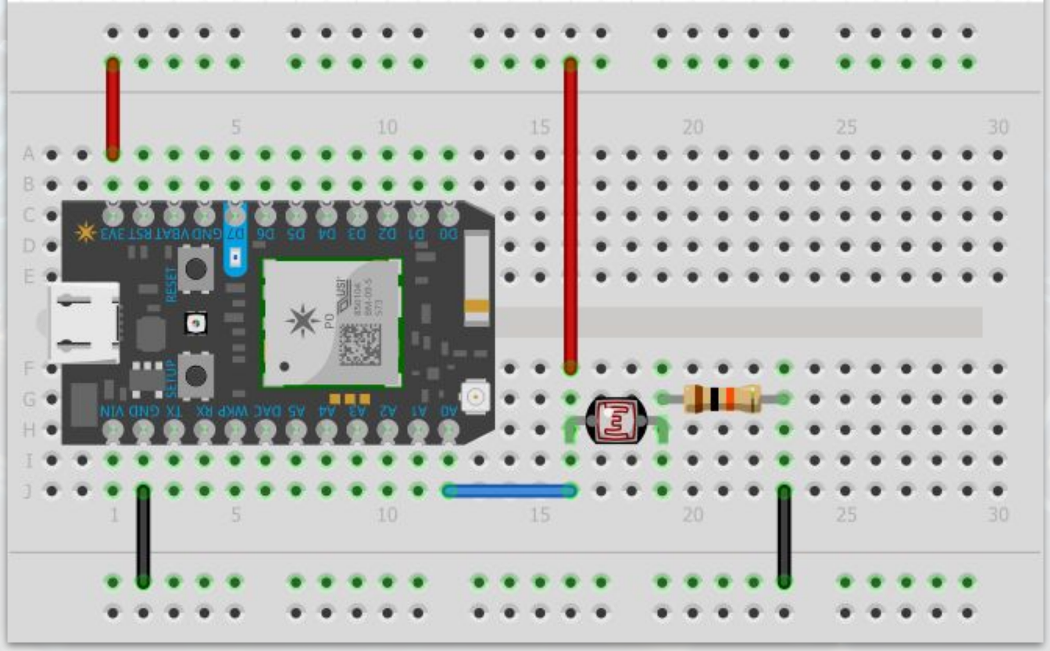
- Place a blue wire adjacent to D2 to open space on the board
- Place an LED with long wire inline with the blue wire
- Place the 330Ω resistor inline with the short LED wire to open space
- Place black wire inline with resistor to the GND bus



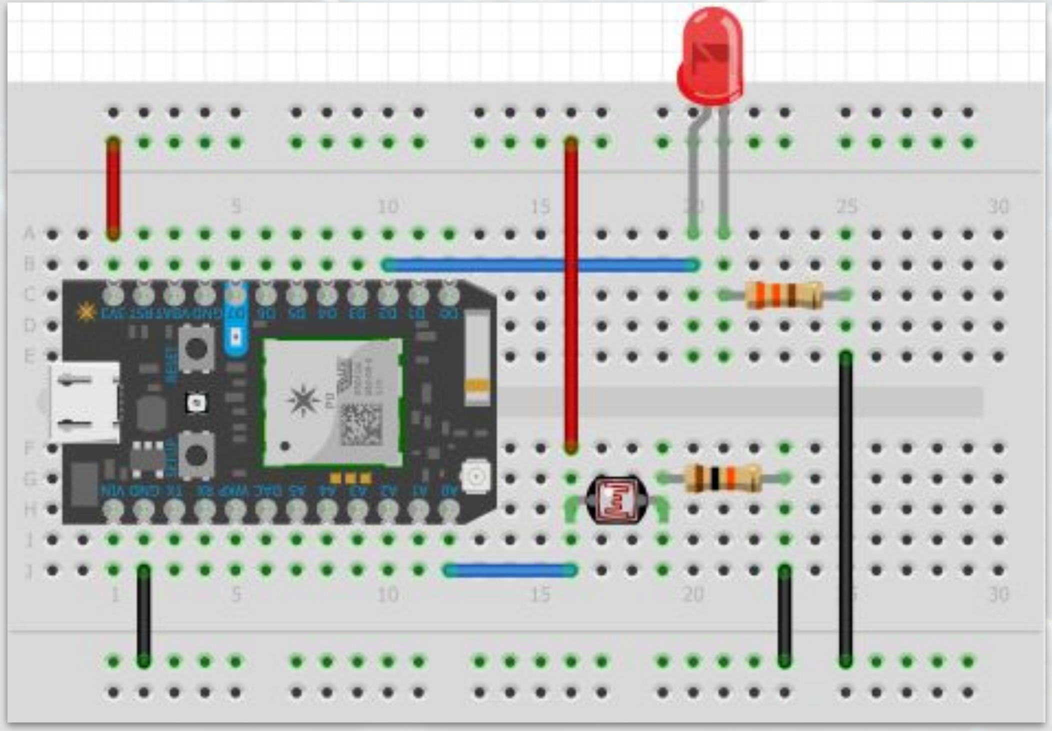
# Wiring the circuit - add photoresistor and v. divider

- Place a blue wire adjacent to A0 to open space, and a red wire inline with blue wire to the 3v3 bus
- Place photoresistor inline with blue and red wires, out to open space
- Connect 10KΩ inline with photoresistor to open space
- Link the end of the resistor to the GND bus

LED circuit removed for clarity but leave in place



## Wiring the circuit - full circuit



3V3 RST VBAT GND D7 D6 D5 D4 D3 D2 D1 D0
VIN GND TX RX WKP DAC A5 A4 A3 A2 A1 A0
RESET
SETUP

## Let’s Talk About Software

## Basic programming constructs - Variables

- Variables store values

```
int age = 18;

bool isHappy = true;
```

## Basic programming constructs - Variables

- Variables have a **type** like **int** (-1,0,1, 2, 3) or **bool** (true, false)

**type**

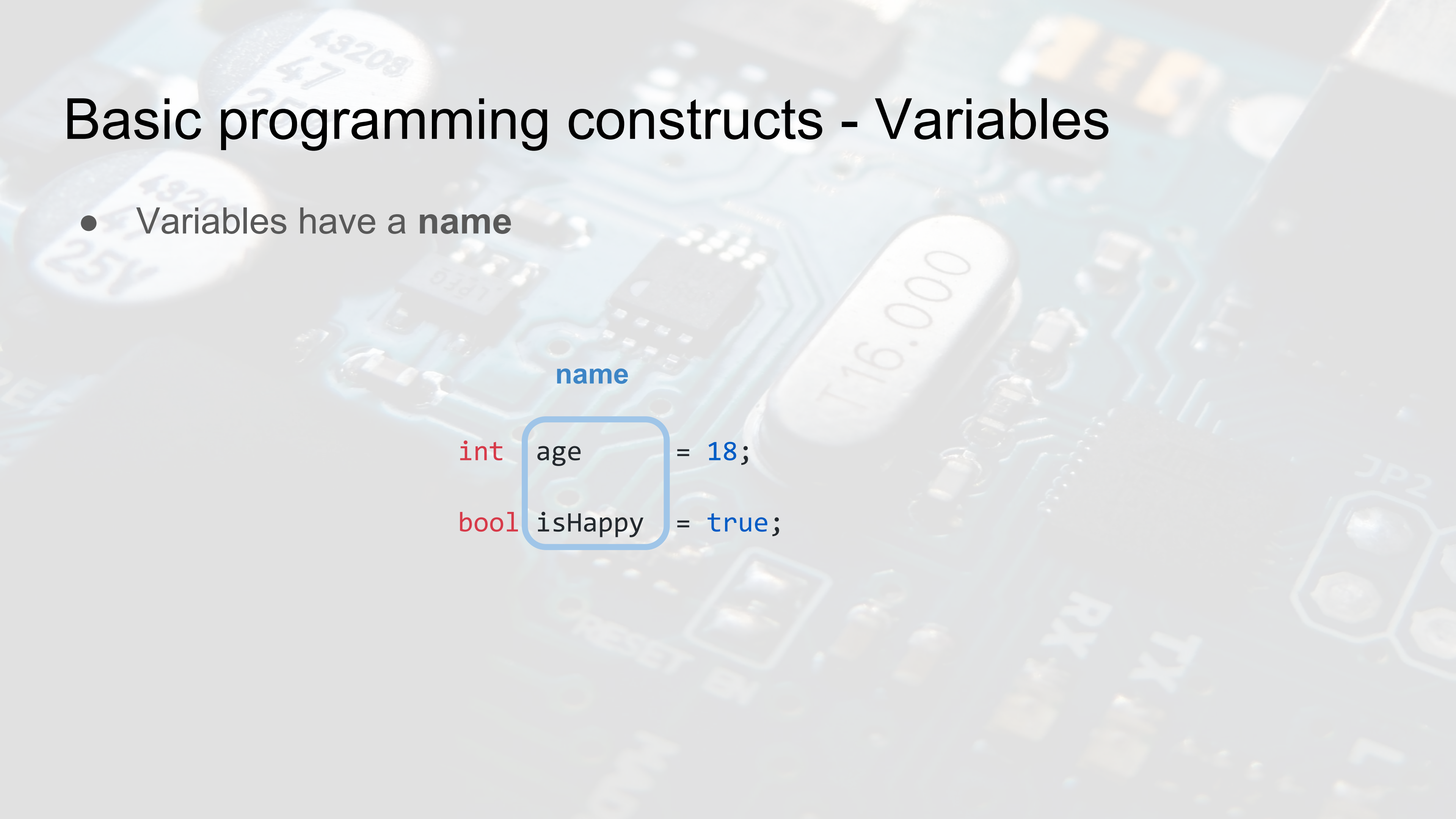
```
int  age     = 18;

bool isHappy = true;
```

## Basic programming constructs - Variables

- Variables have a **name**

name

```
int  age      = 18;

bool isHappy  = true;
```



## Basic programming constructs - Variables

- Variables have a **value**

**value**

```
int  age      = 18;

bool isHappy  = true;
```

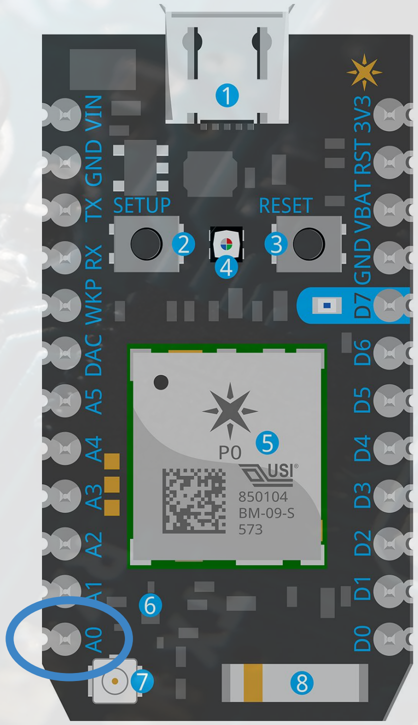
## Basic programming constructs - Variables

- Values of variables can be inspected to make decisions using “if” statements

```
bool isHappy = true;

if (isHappy == true) {
    clapHands(); // this would call a method called clapHands()
} else {
    stompFeet(); // this would call a method called stompFeet()
}
```

# Variables with the Photon

```
int  photoresistor = 0;


  photoresistor = analogRead(A0);


  // sets photoresistor to a number between 0 and 1023
```



## Using variables with a microcontroller

```
int threshold = 750;
int photoresistor = analogRead(A0); // value based on voltage on pin A0 (0 - 1023)

if (photoresistor < threshold) {
   // do something when photoresistor less than threshold value
} else {
   // do something when photoresistor greater than threshold value
}
```

## Basic programming constructs - methods

- A method is a collection of statements that can be called by name
- An Proton program has 2 mandatory methods, setup and loop
- The setup method is where you initialize your program, such as defining which pins will be used for input and output
- The loop method repeats itself over and over and is where the main work is done
- You can write your own methods to help organize your code, but we won’t need to for our project

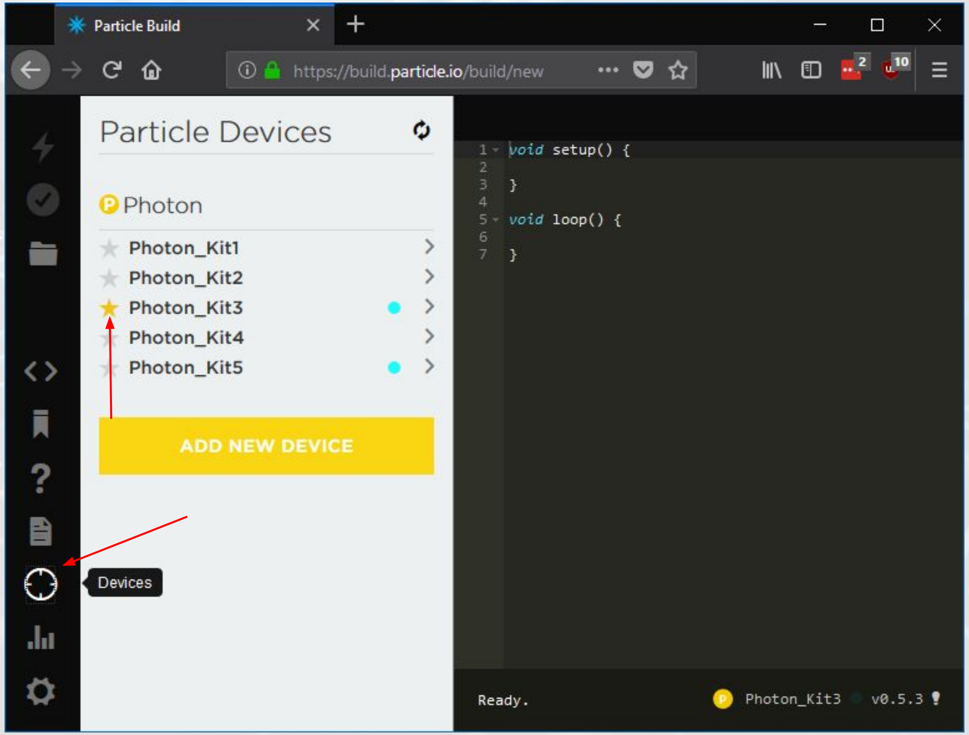
## Let’s get to the code!

- Go to the Particle Console and Log in
  - http://build.particle.io/build
  - User: `lab@buddingtechnologist.com` Password: `l@bdem0!`
  - Each group will select a Photon that is already paired to this account

- You do not need to copy or write source code now because we will load it from an existing project under the above user. If you are doing this project at home, you can get the source code at any time from the Github Project: http://github.com/jeffcardillo/HopeworksCodeDay

## Particle IO - select your group’s device

- In the build console, on the left bar you will see an icon that looks like crosshairs. Click it to see a list of Photons on the account.
- Select the Kit corresponding to your group by clicking the star, such as Photon_Kit3.

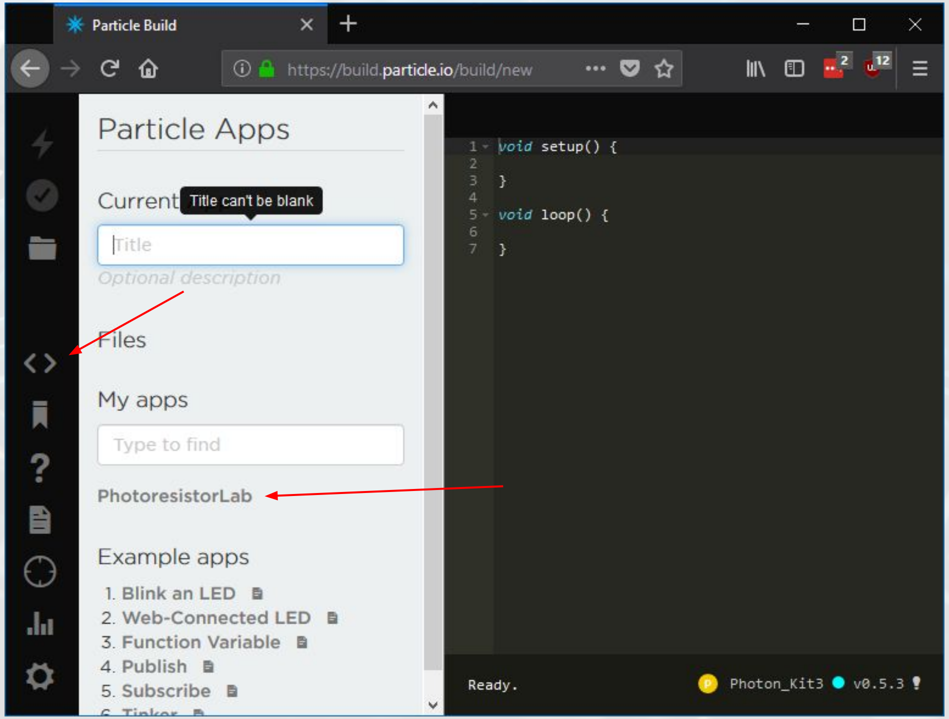


```
void setup() {

}

void loop() {

}
```

## Particle IO - Load the source code for our project

- Now click the “code” icon <>
- Then select the project “PhotoresistorLab” to load the source code

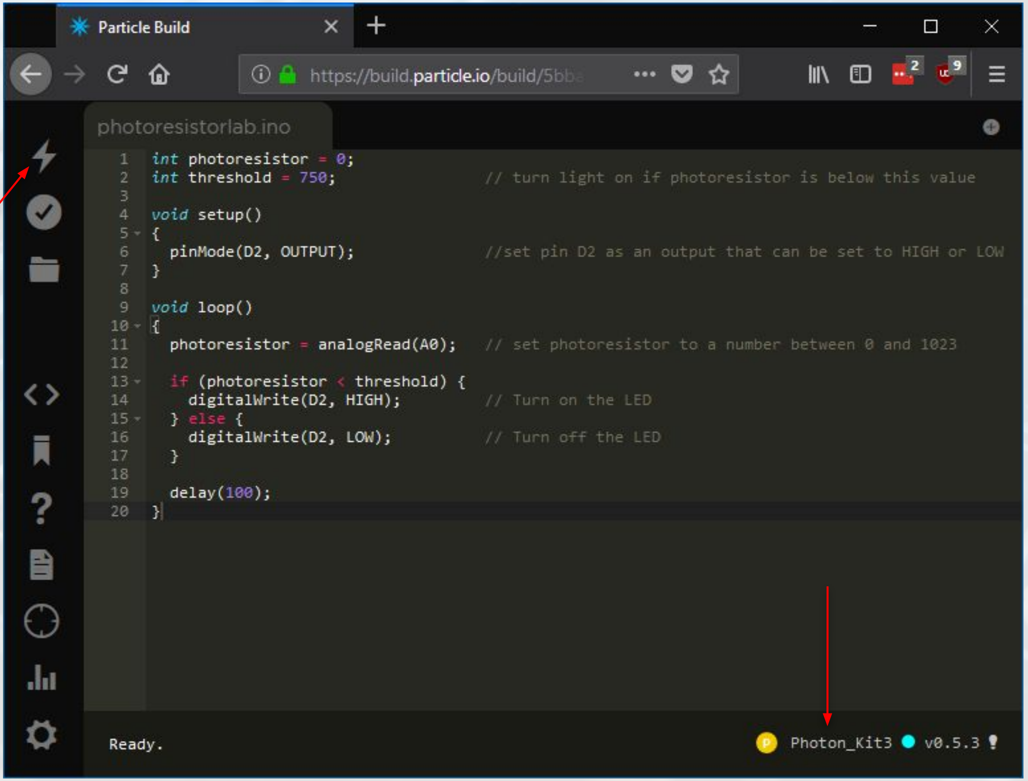
- *Note: If you want to build this project at home, you will need to be using your own Particle IO account. In this case, you can copy the code from the github link provided at the beginning of the presentation and paste it in to the editor*



```
void setup() {

}

void loop() {

}
```

# Particle IO’s build console

- Verify that your device is selected in the bottom right.
- Tap the lightning bolt “flash” button in the top left to download source code to Photon



```
int photoresistor = 0;
int threshold = 750;                // turn light on if photoresistor is below this value

void setup()
{
  pinMode(D2, OUTPUT);              //set pin D2 as an output that can be set to HIGH or LOW
}

void loop()
{
  photoresistor = analogRead(A0);   // set photoresistor to a number between 0 and 1023

  if (photoresistor < threshold) {
    digitalWrite(D2, HIGH);         // Turn on the LED
  } else {
    digitalWrite(D2, LOW);          // Turn off the LED
  }

  delay(100);
}
```

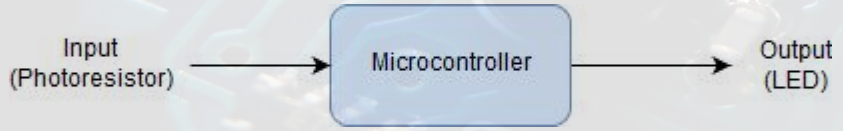
```
Ready.                                                Photon_Kit3  v0.5.3
```

## Flashing the Photon’s memory

- When the Photon is being flashed you will see the onboard LED change to a red & blue color (from afar it will look pink)
- The photon will reboot automatically and the onboard LED will blink green when booting
- Once complete, the onboard LED will slowly blink a bluish white color

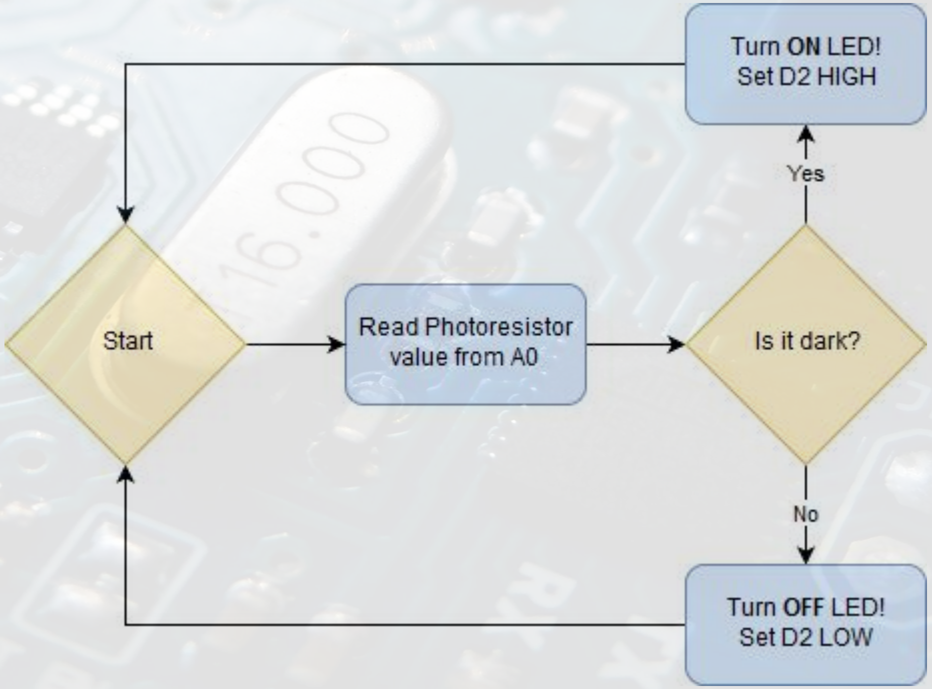
## How it works - Hardware Flow

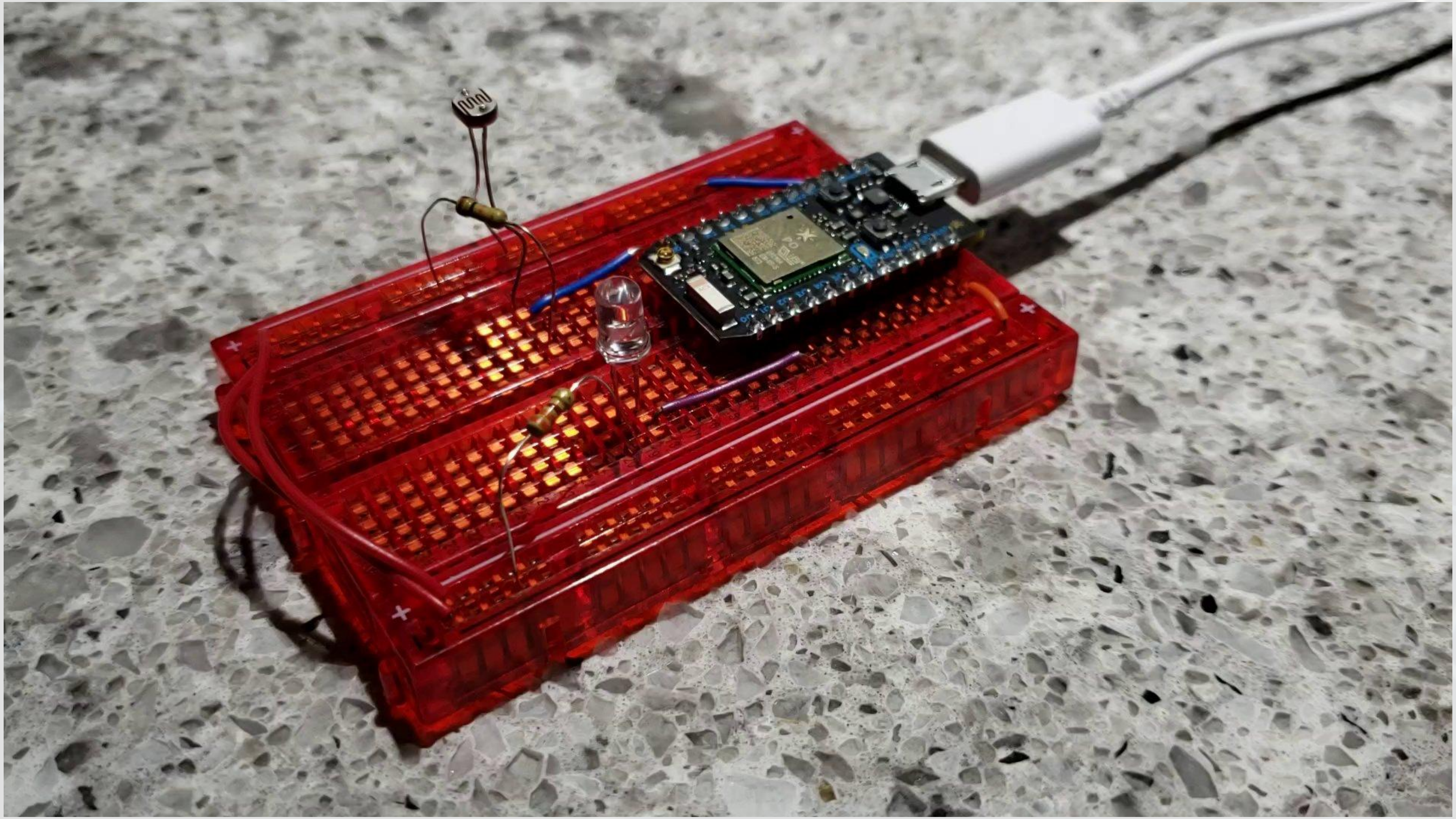
- We have a microcontroller that reads input values from A0 (the photoresistor circuit) and then generates output to an LED



## How it works - Code Flow

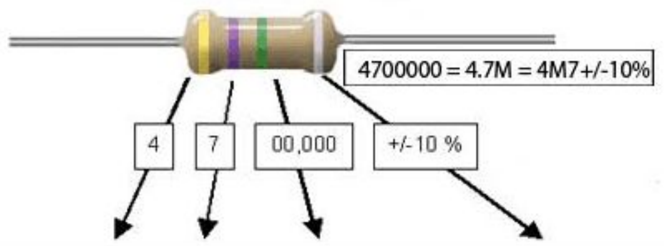
- The loop method starts by reading a value from the photoresistor on pin A0
- The next step is to decide if the value read is “dark” or “light”. Take action accordingly
- Return to the beginning of loop and start again





## Appendix

# Appendix A - Reading Resistor Value



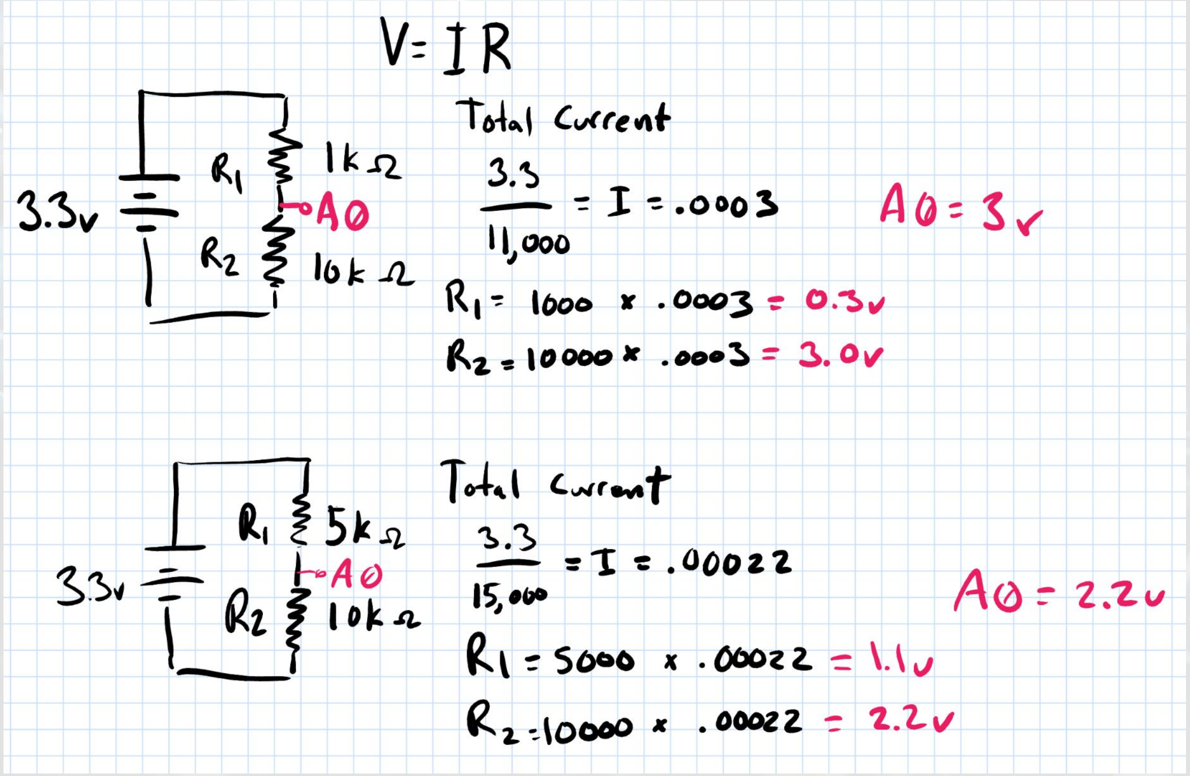
| Band | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Meaning | 1st Digit | 2nd Digit | (No. of zeros) | Tolerance % (No band +/- 20%) |
| Silver | | | .00 (divide by 100) | +/-10% |
| Gold | | | .0 (divide by 10) | +/-5% |
| Black | 0 | 0 | No Zeros | |
| Brown | 1 | 1 | 0 | +/-1% |
| Red | 2 | 2 | 00 | +/-2% |
| Orange | 3 | 3 | ,000 | |
| Yellow | 4 | 4 | 0,000 | |
| Green | 5 | 5 | 00,000 | +/-0.5% |
| Blue | 6 | 6 | ,000,000 | +/-0.25% |
| Violet | 7 | 7 | 0,000,000 | +/-0.1% |
| Grey | 8 | 8 | | +/-0.05% |
| White | 9 | 9 | | |

## Appendix B - Voltage Divider

- A voltage divider uses the voltage drop across resistors to convert varying resistances to a voltage value
- This is done with a fixed resistance opposed to a variable resistance, like the photoresistor
- We can use Ohm's law to calculate: V = I*R (Voltage = Current * Resistance)
- Steps to calculate
  a. Calculate total Voltage Divider resistance
  b. Calculate current
  c. Multiply current by individual resistances to get voltage drops
- Example ...

## Appendix B - Voltage Divider (continued)

- In the examples we are using 3.3v total and R1 is variable resistance
- Voltage at A0 is the total voltage (3.3v) minus the drop across R1



V= I R
3.3v
R1
1kΩ
A0
R2
10kΩ
Total Current
3.3 / 11,000 = I = .0003
A0 = 3v
R1 = 1000 x .0003 = 0.3v
R2 = 10000 x .0003 = 3.0v
3.3v
R1
5kΩ
A0
R2
10kΩ
Total current
3.3 / 15,000 = I = .00022
A0 = 2.2v
R1 = 5000 x .00022 = 1.1v
R2 = 10000 x .00022 = 2.2v