



2017 WEEK OF LEARNING

INTERNET OF THINGS

Rick Suel

CLASS PREREQUISITES

1. Some programming experience (i.e. Arduinos class, etc).
2. Download *Particle* mobile phone app:
 - ▶ iPhone: <https://itunes.apple.com/us/app/particle-build-photon-electron/id991459054?ls=1&mt=8>
 - ▶ Android: <https://play.google.com/store/apps/details?id=io.particle.android.app>
3. Create an account on <https://www.particle.io>
4. Create an account on <https://ifttt.com>
5. Create an amazon account if you don't already have one: <https://www.amazon.com>
6. Log into <https://echosim.io> with your amazon account.
7. Install the atom code editor <https://atom.io/> (do not use notepad for this class!)
8. Have the chrome browser installed and ready to go. I have not tested Internet Explorer and will not do so :).

INTRODUCTION

CLASS MATERIALS

https://github.com/rdsuel/iot_class

The screenshot shows the GitHub repository page for `rdsuel / iot_class`. The repository is described as "The IoT class that I teach at GE Appliances." It has 27 commits, 1 branch, 0 releases, and 1 contributor. The repository is currently on the `master` branch. A "Clone or download" button is visible, which has opened a dropdown menu showing options to "Clone with HTTPS" (using the URL `https://github.com/rdsuel/iot_class.git`), "Use SSH", "Open in Desktop", and "Download ZIP" (4 months ago). The repository contains the following files and folders:

File/Folder	Commit Message	Time Ago
<code>source</code>	Preparing for public github.	
<code>.gitignore</code>	Preparing for public github.	
<code>IoT_Presentation.key</code>	Updates.	
<code>IoT_Presentation.pdf</code>	Updates.	
<code>README.md</code>	Update README.md	11 months ago

The `README.md` file is displayed below the file list. It contains the title "IoT Class Materials" and the authors "by Rick Suel and Juan Espinosa".

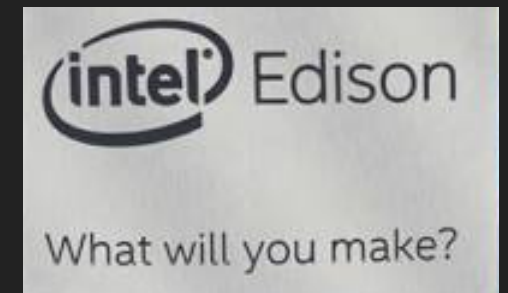
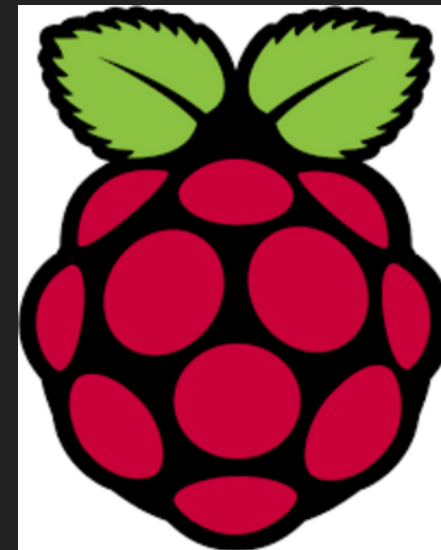
WHAT IS THE “INTERNET OF THINGS” (IOT)?

The Internet of Things includes “connected” embedded devices, sensors, etc that collect and share data, typically via a cloud service, where the data can be shared, analyzed etc. Some applications include:

- ▶ Smart home (nest, automation, etc)
- ▶ Industrial IoT (supply chain, sensors, GE predix, etc)
- ▶ Wearables (smart watches, fitness trackers, etc)
- ▶ Energy (smart meters)
- ▶ Healthcare
- ▶ And many, many more ...

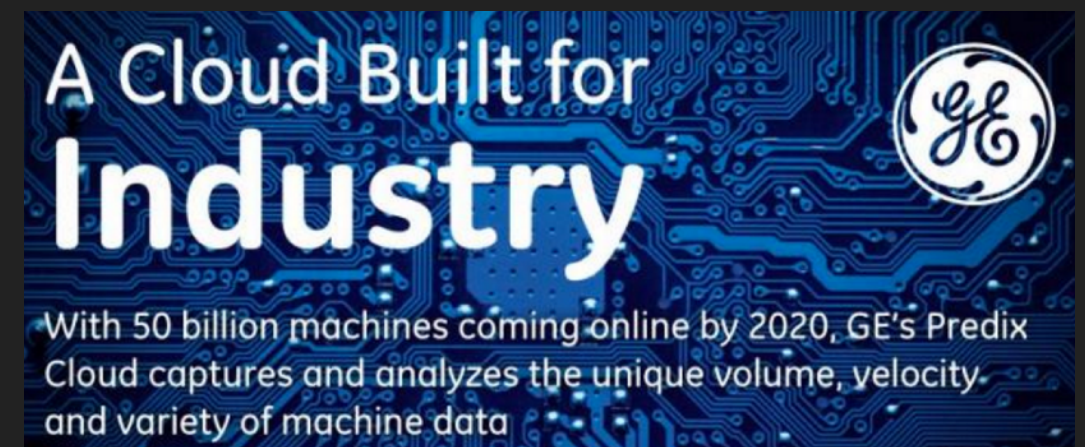
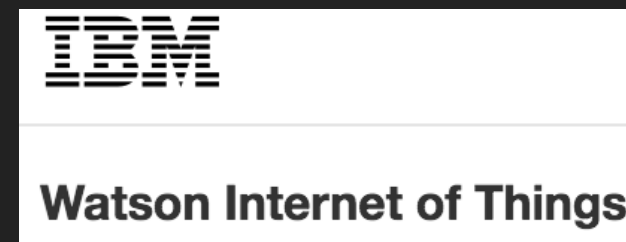
SOME DEVICE MANUFACTURERS

- ▶ Almost every embedded device manufacturer has an IoT offering. These are just a few such manufacturers:
- ▶ Texas Instruments, Intel, NXP, Raspberry Pi, Arduino, Onion, Particle, Electric Imp, and many, many more ...



SOME CLOUD SERVICE PROVIDERS

- ▶ Similar to device manufacturers, many familiar software companies are now offering IoT cloud solutions:
- ▶ Microsoft Azure, Amazon Web Services, Salesforce.com, Google Cloud Platform, IBM Watson, GE Predix, etc ...

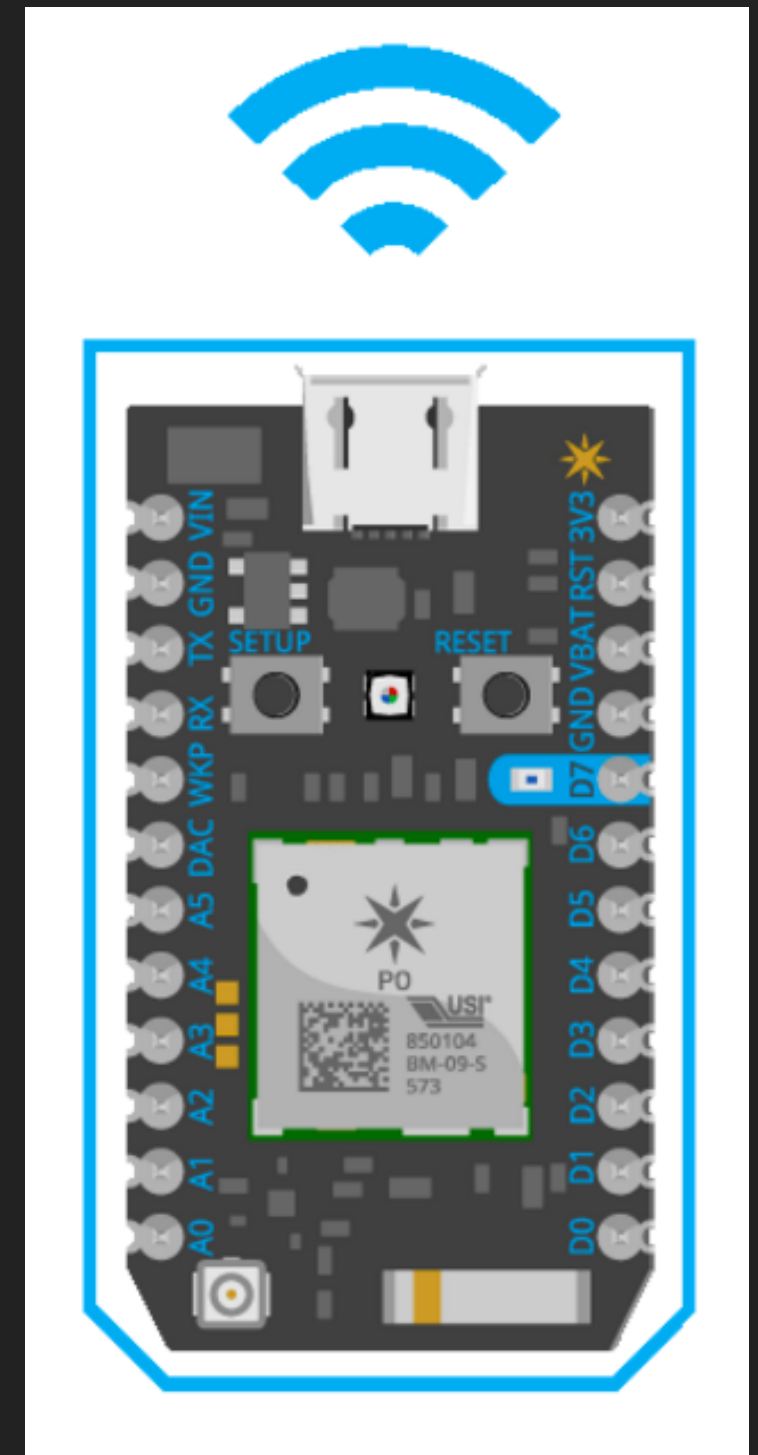


INTRODUCTION

PARTICLE PHOTON (\$19)

- ▶ We will be using the www.particle.io photon device and particle cloud for the duration of this class.
- ▶ This device is very simple to get started with, and even uses "Arduino-style" coding right in your web browser.
- ▶ Start here: <https://docs.particle.io/guide/getting-started/intro/photon/>
- ▶ Photon Features:
 - ▶ 802.11b/g/n Wifi
 - ▶ 120MHz Arm Cortex M3 micro
 - ▶ 1MB Flash, 128KB RAM
 - ▶ 18 Mixed-signal GPIO pins and advanced peripherals (ADC, DAC, SPI, I2C, CAN, USB, PWM).

<https://docs.particle.io/datasheets/photon-datasheet/>



WHERE TO FIND HELP

PARTICLE CLOUD API

Where to find Particle API documentation:

<https://docs.particle.io/reference/firmware/photon/>

The screenshot shows the Particle Cloud API documentation page for the Photon firmware. The page has a navigation bar with links to GUIDE, TUTORIALS, FAQ, REFERENCE (highlighted), DATASHEETS, and SUPPORT. A search bar is located in the top right corner. The left sidebar shows a tree view with 'Firmware' expanded, and 'Cloud Functions' selected. The main content area is titled 'Particle.function()' and describes how to expose a function through the Cloud. It includes a note about function names and an example of how to register a cloud function. The right sidebar shows a code block with the syntax for the Particle.function() method.

Particle.function()

Expose a *function* through the Cloud so that it can be called with **POST** `/v1/devices/{DEVICE_ID}/{FUNCTION}`.

Up to 15 cloud functions may be registered and each function name is limited to a maximum of 12 characters.

Note: Only use letters, numbers, underscores and dashes in function names. Special characters may be escaped by different tools and libraries causing unexpected results.

In order to register a cloud function, the user provides the **funcKey**, which is the string name used to make a POST request and a **funcName**, which is the actual name of the function that gets called in your app. The cloud function can return any integer; **-1** is commonly used for a failed function call.

A cloud function is set up to take one argument of the **String** datatype. This argument length is limited to a max of 63 characters.

```
// SYNTAX
bool success = Particle.function("funcKey", funcName);

// Cloud functions must return int and take one String
int funcName(String extra) {
    return 0;
}
```

```
// EXAMPLE USAGE
```


EXERCISE 1

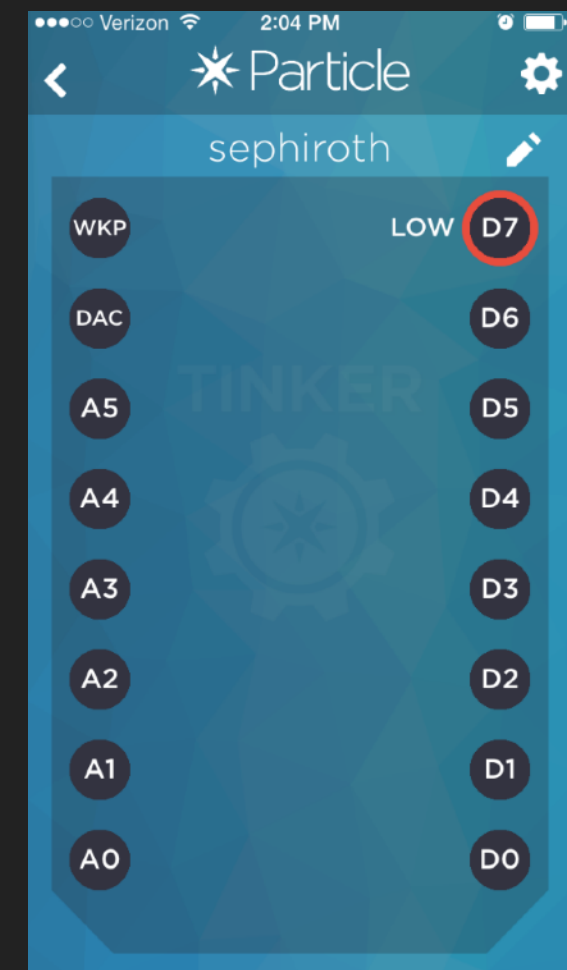
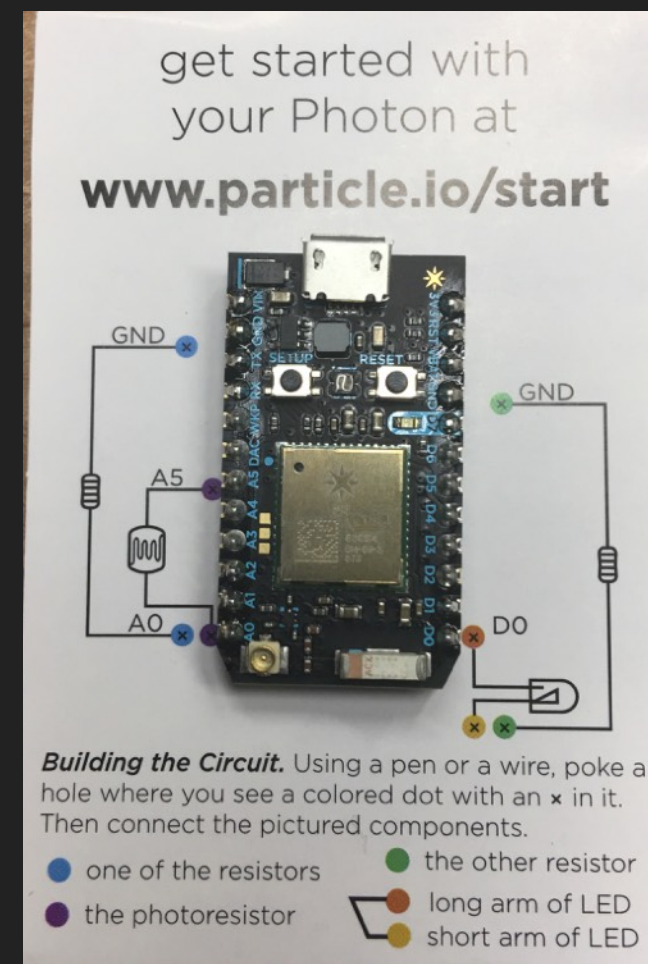
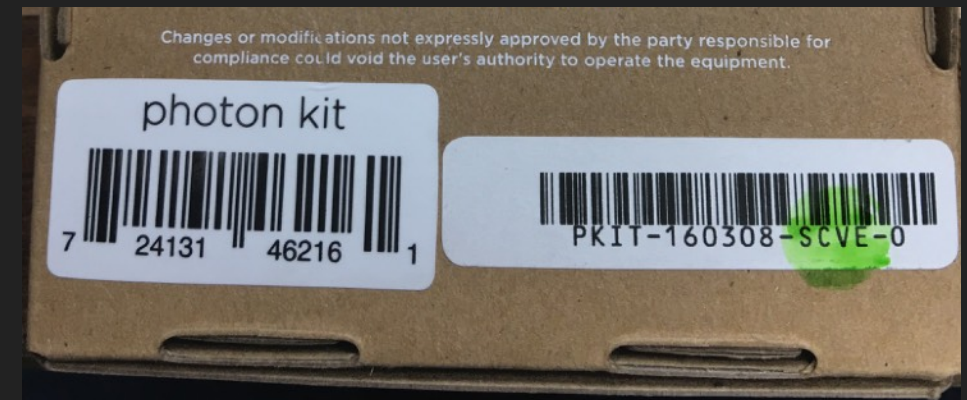
SETUP AND “TINKER”

Goals:

- ▶ Connect your photon to wifi.
- ▶ Interact with the device over the internet using your smart phone.

Steps:

1. Wire the circuit as shown in the kit.
2. Connect your photon to the “photon” wifi network using the Particle mobile app. The password is “password”.
3. Use the “Tinker” utility built into the Particle app to drive the LED and read the light sensor in your circuit.



EXERCISE 2

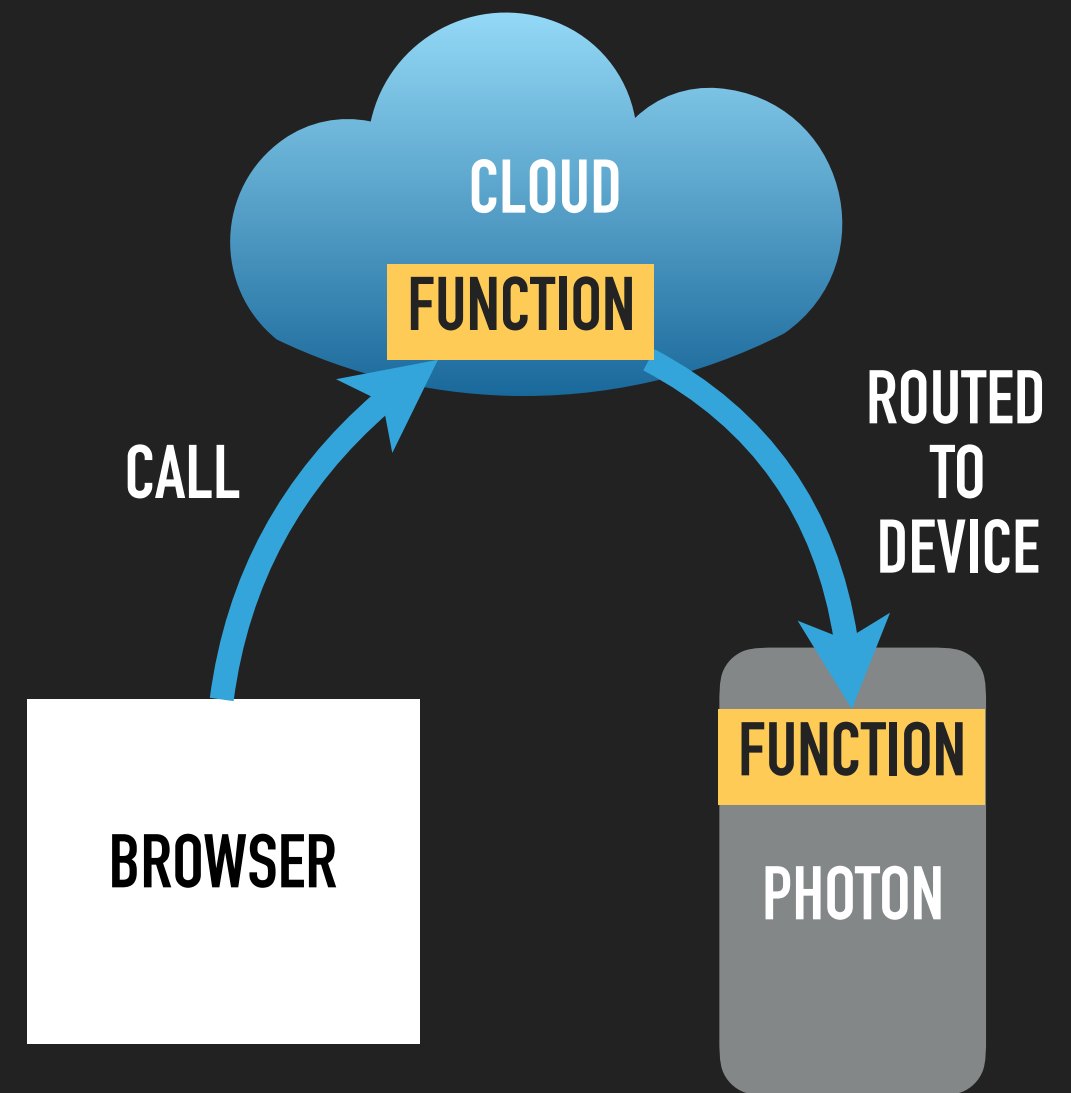
CONTROLLING AN LED WITH YOUR BROWSER

Goals

- ▶ Publish a “Web Function” from your photon.
- ▶ Call that function using a web browser to blink an LED on the photon.

Steps:

1. Write the firmware to add a “web function” to your particle called “led”. This function will be used to control the LED over the web by writing “ON” and “OFF” to the web function.
2. Add your device ID and access key to the HTML template.
3. Add HTML buttons to turn the LED on and off.



Firmware: <https://docs.particle.io/reference/firmware/photon/#particle-function>

HTML API: <https://docs.particle.io/reference/api/#call-a-function>

EXERCISE 3

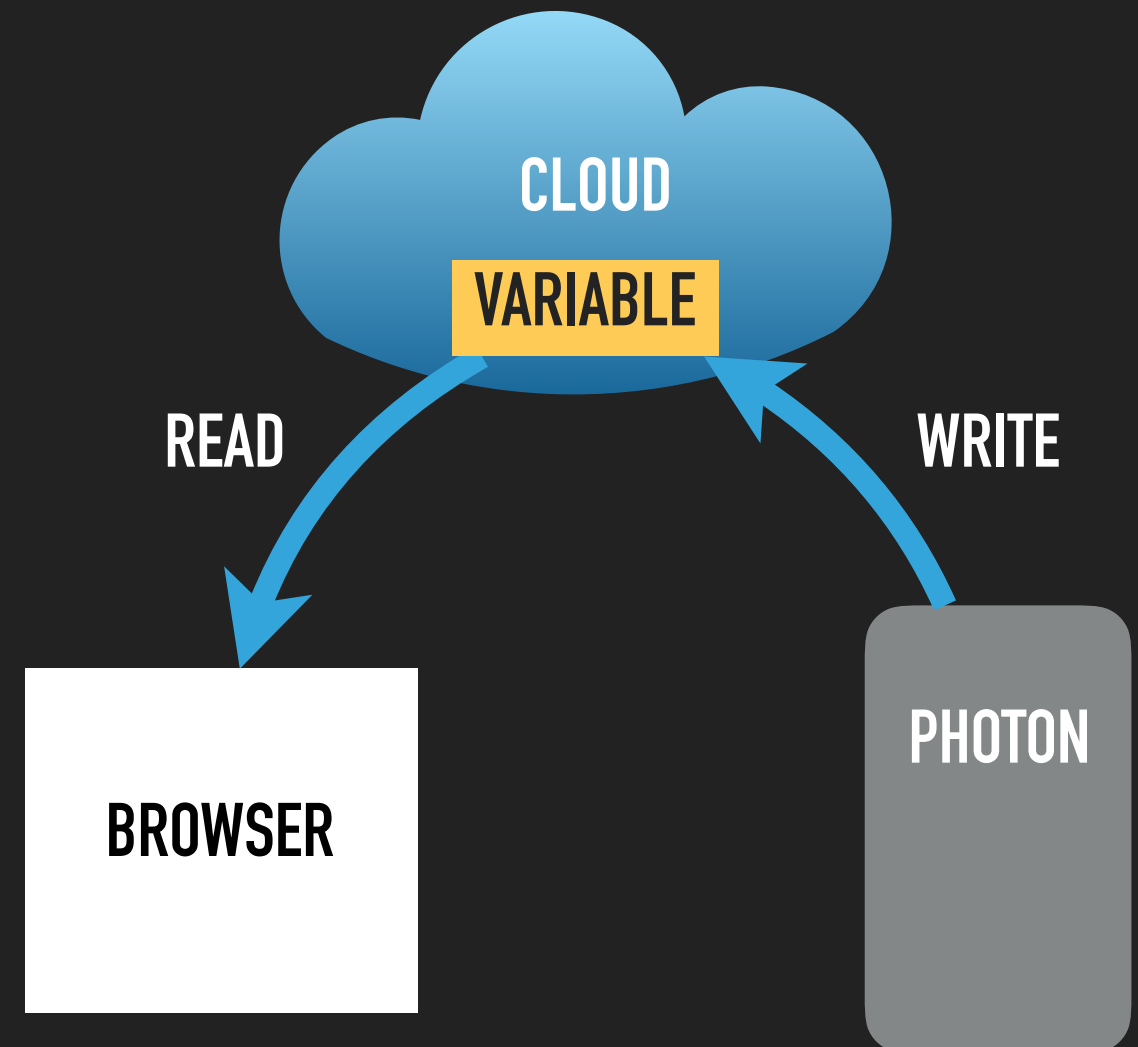
READ A SENSOR WITH YOUR BROWSER

Goals:

- ▶ Publish a "Variable" to the cloud that holds the light sensor value.
- ▶ Read the "Variable" from the cloud with your web browser.

Steps:

1. Write the firmware to add a "web variable" called "photoValue" to your particle. Update the variable with the photo sensor value periodically.
2. Update the HTML template to add your device ID and access key (your photon's "address" in the cloud).
3. Update the HTML to periodically read the photo sensor value from the cloud and displays it in the browser window.



Firmware: <https://docs.particle.io/reference/firmware/photon/#particle-variable>

HTML API: <https://docs.particle.io/reference/api/#get-a-variable-value>

EXERCISE 4

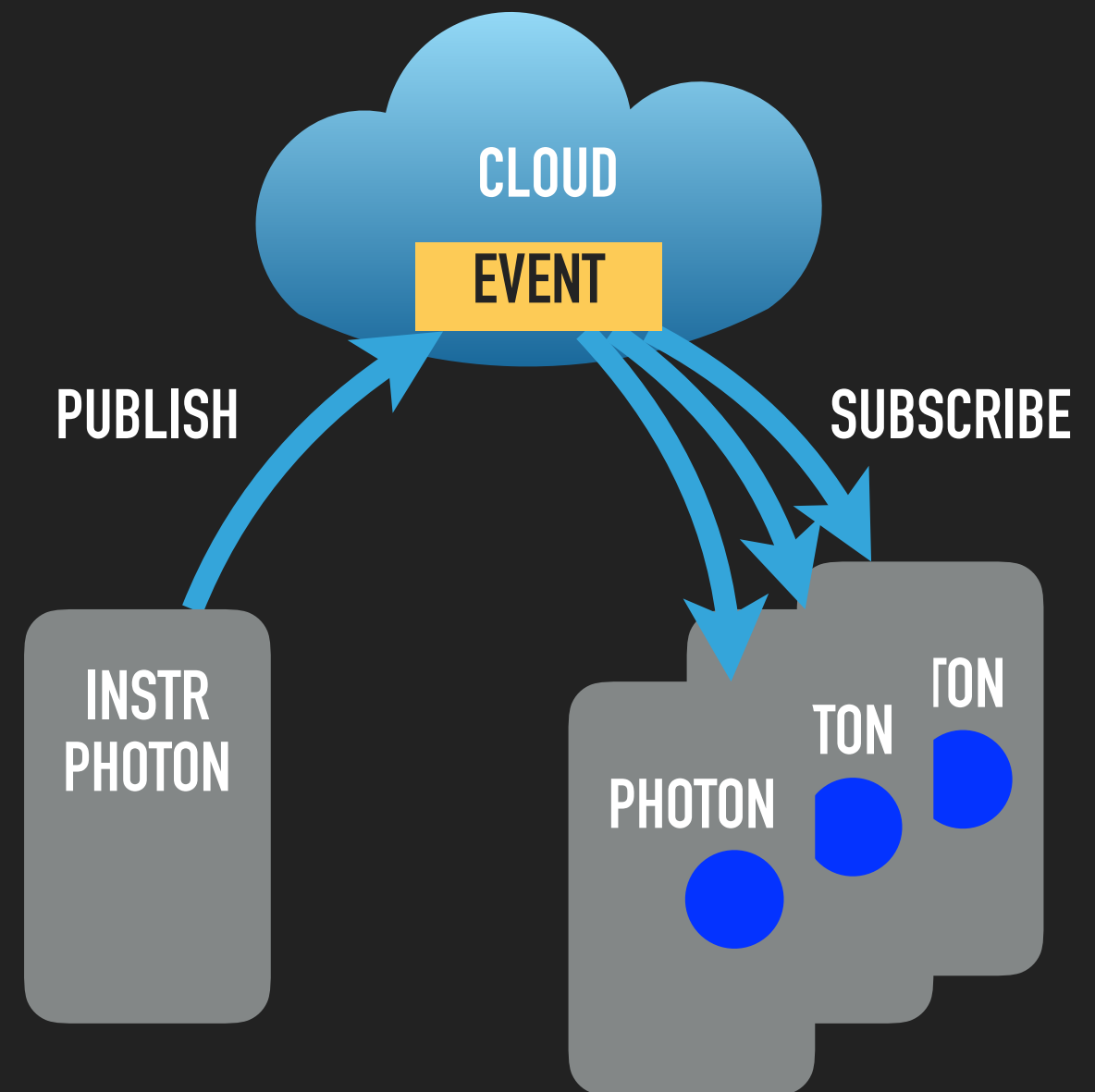
DEVICE-TO-DEVICE COMMUNICATION

Goals:

- ▶ Publish a "Web Event" from a photon to the cloud (Instructors).
- ▶ Subscribe to the "Web Event" and control an LED (students).

Steps:

1. Add a `Particle.subscribe()` handler for the "photoStatus" event. When the value is "NIGHT" turn on your board LED (D7), when the value is "DAY", turn off your LED (D0).
2. The instructors will create `Particle.publish()` for the "photoStatus" event on their photon. When the photo sensor value is below 300, the event argument will be "NIGHT", and when above 400, it will be "DAY".
3. The instructors can then control all photon LEDs in the class by interacting with the light sensor on the instructor's photon.



Subscribe: <https://docs.particle.io/reference/firmware/photon/#particle-subscribe>

Publish: <https://docs.particle.io/reference/firmware/photon/#particle-publish>

EXAMPLE 5

CONTROLLING AN LED WITH ALEXA AND IFTTT

Goals:

- ▶ Create an IFTTT recipe that turns on a photon LED when an Alexa command is issued.

Steps:

1. Log into ifttt.com and create a new applet.
2. Select "Amazon Alexa" for *this* and choose the "Say a specific phrase" Trigger.
3. Set the phrase to "light on" and press "create trigger" button.
4. Select "Particle" for *that* and choose "Call a function" for the Action.
5. Select your "led" function on your photon device.
6. Type "ON" in the "with input (Function Input)" field.
7. Press the "Create Action" button.
8. On the final screen hit "Create Recipe".
9. Go to echosim.io and say "Alexa trigger light on".



If You say "Alexa trigger light on", then call a function



HELPFUL LINKS

- ▶ Particle documentation: <https://docs.particle.io/guide/getting-started/intro/photon/>
- ▶ Particle web IDE: <https://build.particle.io/build>
- ▶ Particle console: <https://console.particle.io/devices>
- ▶ Particle community: <https://community.particle.io/>
- ▶ Project ideas: <https://www.hackster.io/particle>
- ▶ Great site for finding/understanding web APIs: <http://www.programmableweb.com/>
- ▶ Losant is powerful cloud platform. Here is how to connect your photon to it: <https://docs.losant.com/getting-started/losant-iot-dev-kits/builder-kit-particle/>
- ▶ [plot.ly](https://images.plot.ly/plotly-documentation/images/plotly_js_cheat_sheet.pdf) cheat sheet: https://images.plot.ly/plotly-documentation/images/plotly_js_cheat_sheet.pdf