

# Hello, Photon!

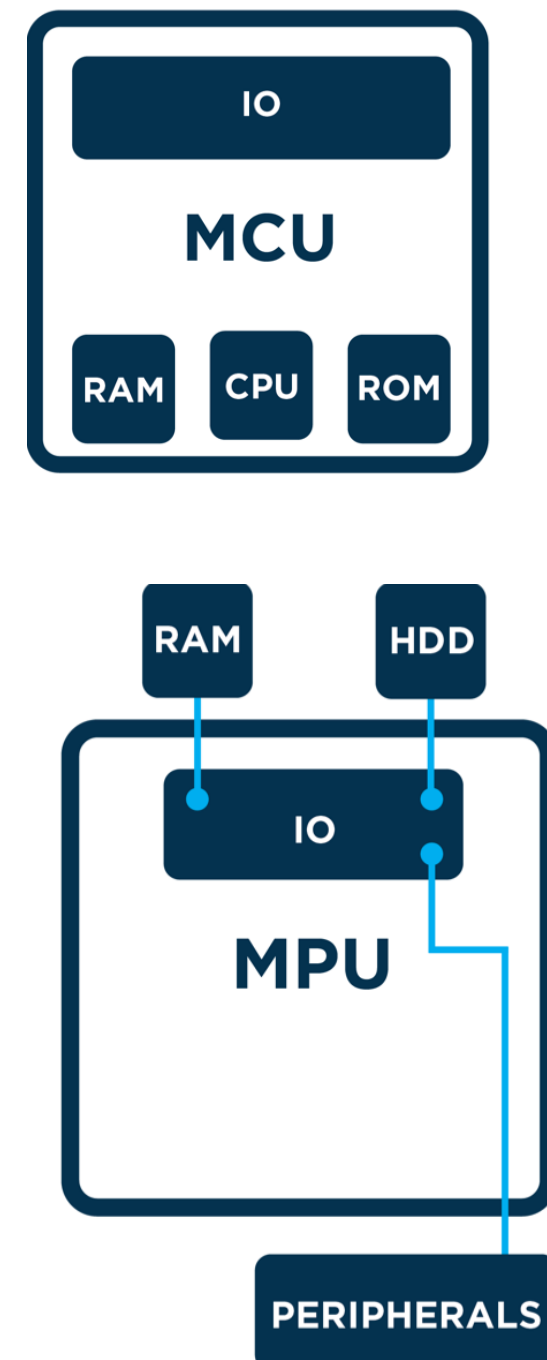
44-440/640-IoT

# Objectives

- Students will be able to:
  - explain in general terms the difference between a microcontroller and a microprocessor
  - describe the major components of the Photon development board
  - configure the Photon to connect to the internet
  - develop a simple program in [build.particle.io](https://build.particle.io) and run it on the Photon
  - define "firmware"

# MCU v. MPU

- A **microcontroller**, of which the Photon is an example, is essentially a computer on a chip (SOC), comprised of a processor (CPU), with RAM, ROM, and I/O ports. They tend to be devoted to a single task.
- An MCU has a minimalist OS (if any), and is easy to connect to sensors and servos, to program, and to use: turn it on, load **firmware**, and it's ready to go. Because of the simpler OS, they are less prone to hacks. They are also cheaper.
- A **microprocessor** only contains a processor -- everything else must be added. Microprocessors are more powerful, better suitable for general purpose computing.
- If your mission is complex (e.g., hardware for Home Pod) you'll want a microprocessor; if your mission is simple (control a thermostat), an MCU would be a better choice.



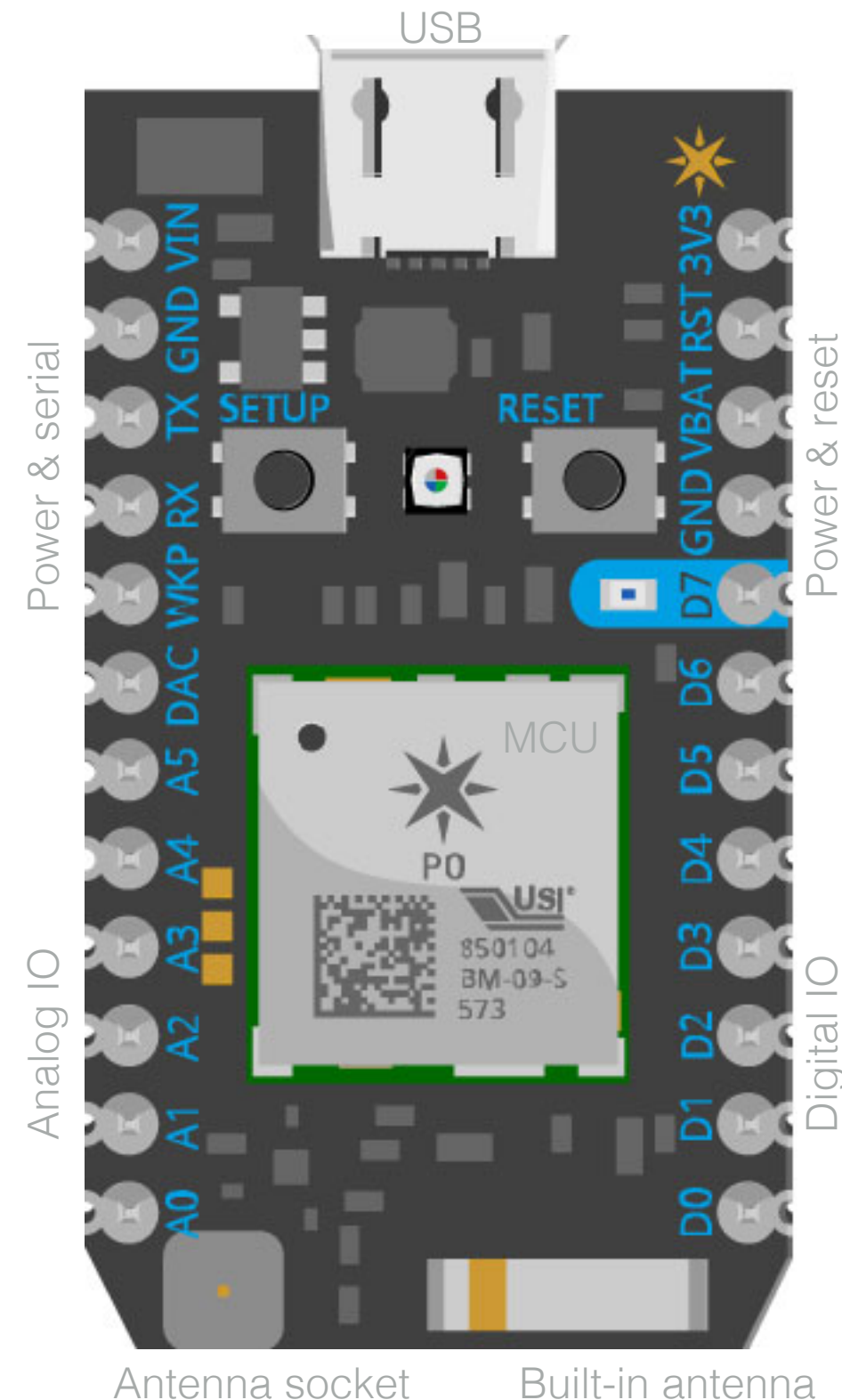
# Firmware

- **firmware** ['fermwer]. Noun. The software that resides on a microcontroller.
- It is a slight abuse of terminology to call the source code you write firmware — technically you are coding in C++, and the firmware is the machine code version of the same (linked with any other required functions). Photons speak ARM.
- When you flash your program onto the Photon, almost all of the software on the device gets updated — only Device OS remains unchanged
- Fun fact: It used to take minutes to flash your code over the air (so to speak: it will also work in a vacuum), but when Particle replaced the original Spark with the Photon, they decreased the time to just a few seconds.

# A Quick Photon Tour & Reference

You won't understand all this yet ... but you will, eventually

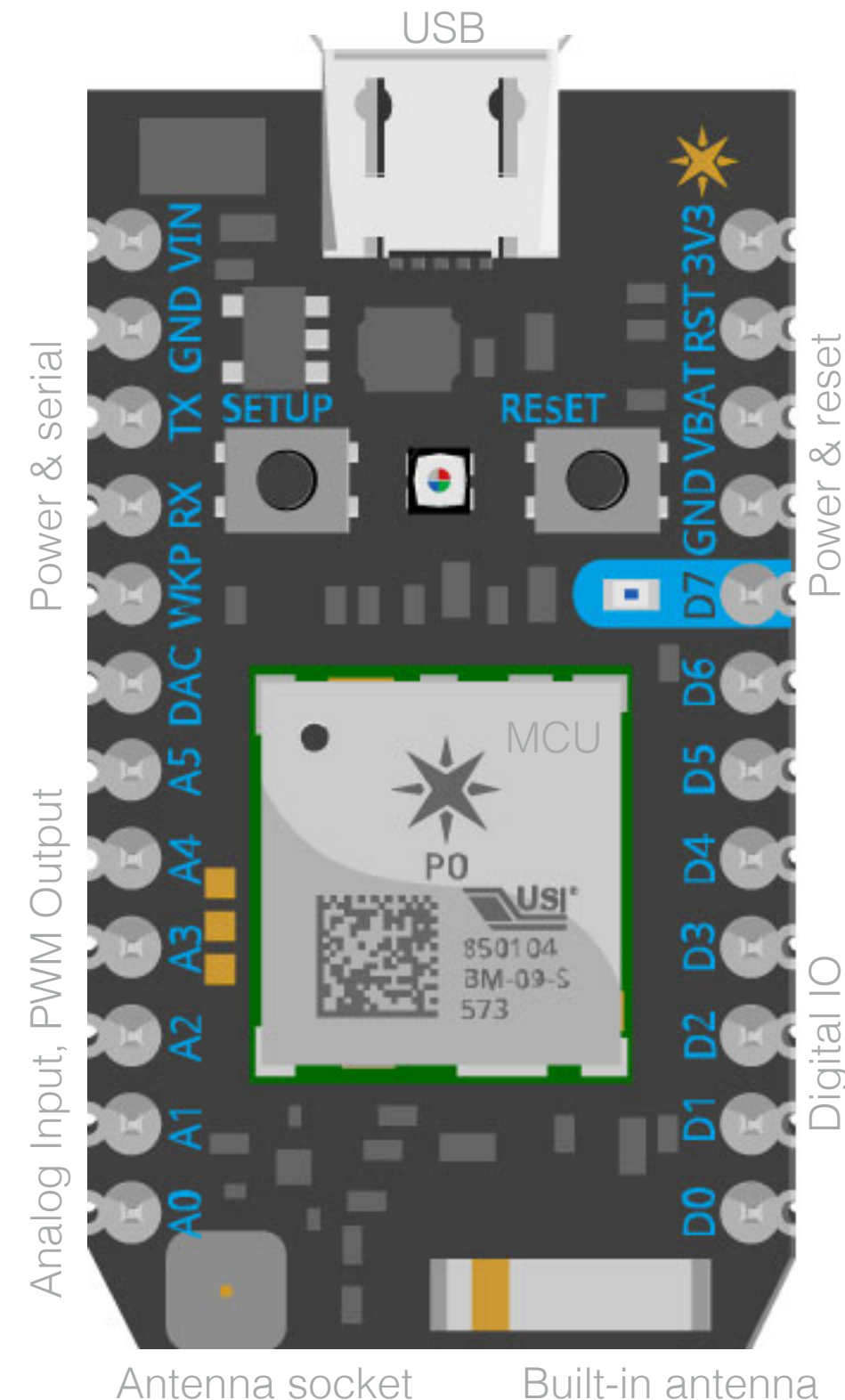
- The Photon is a development board, consisting of microcontroller and a Wi-Fi chip supporting 802.11b/g/n.
- SETUP puts the Photon into listening mode so you can put in new Wi-Fi credentials; RESET restarts the application firmware; together they reset the board to factory specs.
- VIN is used as input (3.6-5.5VDC to power the photon). It can also be used as output when powered by USB (it emits 4.8 VDC, with a max load of 1A)
- 3V3 can be used as a source of 3.3VDC (max load of 100 mA) when the Photon is powered by USB or by VIN. It can also power the Photon (with a max of 3.3VDC).



# A Quick Photon Tour & Reference

You won't understand all this yet ... but you will, eventually

- D0-D7 are digital only GPIO (General Purpose IO) pins: they can be either LOW or HIGH. D0-D3 may also be used for PWM (e.g., to control servos).
- A0-A7 can be used as digital only GPIO pins or as analog inputs (measuring a voltage between 0-3.3VDC, and converting it to an integer between 0-4095) or outputs (using pulse-width modulation: this is not truly analog, the voltage is either 0 or 3.3VDC, but the width of the duty cycle changes )
- DAC (digital-analog converter) is truly analog: it maps 0-4095 to 0-3.3VDC
- RST can be used to programmatically reset the board.
- The Status LED in the center is used by the Photon to display its mode (but it can be controlled by the programmer as well): another LED, attached to D7, can be controlled programmatically.
- At the bottom left is a connection for an external antenna; at the bottom right is a built-in antenna that works fine for our purposes.





# Techy Aside: All The Pins



Pin	Description
VIN	This pin can be used as an input or output. As an input, supply 3.6 to 5.5VDC to power the Photon. When the Photon is powered via the USB port, this pin will output a voltage of approximately 4.8VDC due to a reverse polarity protection series Schottky diode between VUSB and VIN. When used as an output, the max load on VIN is 1A.
RST	Active-low reset input. On-board circuitry contains a 1k ohm pull-up resistor between RST and 3V3, and 0.1uF capacitor between RST and GND.
VBAT	Supply to the internal RTC, backup registers and SRAM when 3V3 is not present (1.65 to 3.6VDC).
3V3	This pin is the output of the on-board regulator and is internally connected to the VDD of the Wi-Fi module. When powering the Photon via VIN or the USB port, this pin will output a voltage of 3.3VDC. This pin can also be used to power the Photon directly (max input 3.3VDC). When used as an output, the max load on 3V3 is 100mA. NOTE: When powering the Photon via this pin, ensure power is disconnected from VIN and USB.
RX	Primarily used as UART RX, but can also be used as a digital GPIO or PWM <sup>[2]</sup> .
TX	Primarily used as UART TX, but can also be used as a digital GPIO or PWM <sup>[2]</sup> .
WKP	Active-high wakeup pin, wakes the module from sleep/standby modes. When not used as a WAKEUP, this pin can also be used as a digital GPIO, ADC input or PWM <sup>[2]</sup> . Can be referred to as <b>A7</b> when used as an ADC.
DAC	12-bit Digital-to-Analog (D/A) output (0-4095), referred to as <b>DAC</b> or <b>DAC1</b> in software. Can also be used as a digital GPIO or ADC. Can be referred to as <b>A6</b> when used as an ADC. A3 is a second DAC output used as <b>DAC2</b> in software.
A0~A7	12-bit Analog-to-Digital (A/D) inputs (0-4095), and also digital GPIOs. <b>A6</b> and <b>A7</b> are code convenience mappings, which means pins are not actually labeled as such but you may use code like <code>analogRead(A7)</code> . <b>A6</b> maps to the DAC pin and <b>A7</b> maps to the WKP pin. A4,A5,A7 may also be used as a PWM <sup>[2]</sup> output.
D0~D7	Digital only GPIO pins. D0~D3 may also be used as a PWM <sup>[2]</sup> output.

# Reference: Device Modes

- The Photon has numerous device modes, which affect the status LED:
  - Blinking green -- trying to connect to the internet
  - Breathing cyan -- connected to the internet, life is good
  - Blinking blue -- in listening mode, ready to configure so it can connect to Wi-Fi
  - Blinking magenta (red & blue) -- loading an app or a newer version of Device OS



# Techy Details You Don't Need To Know

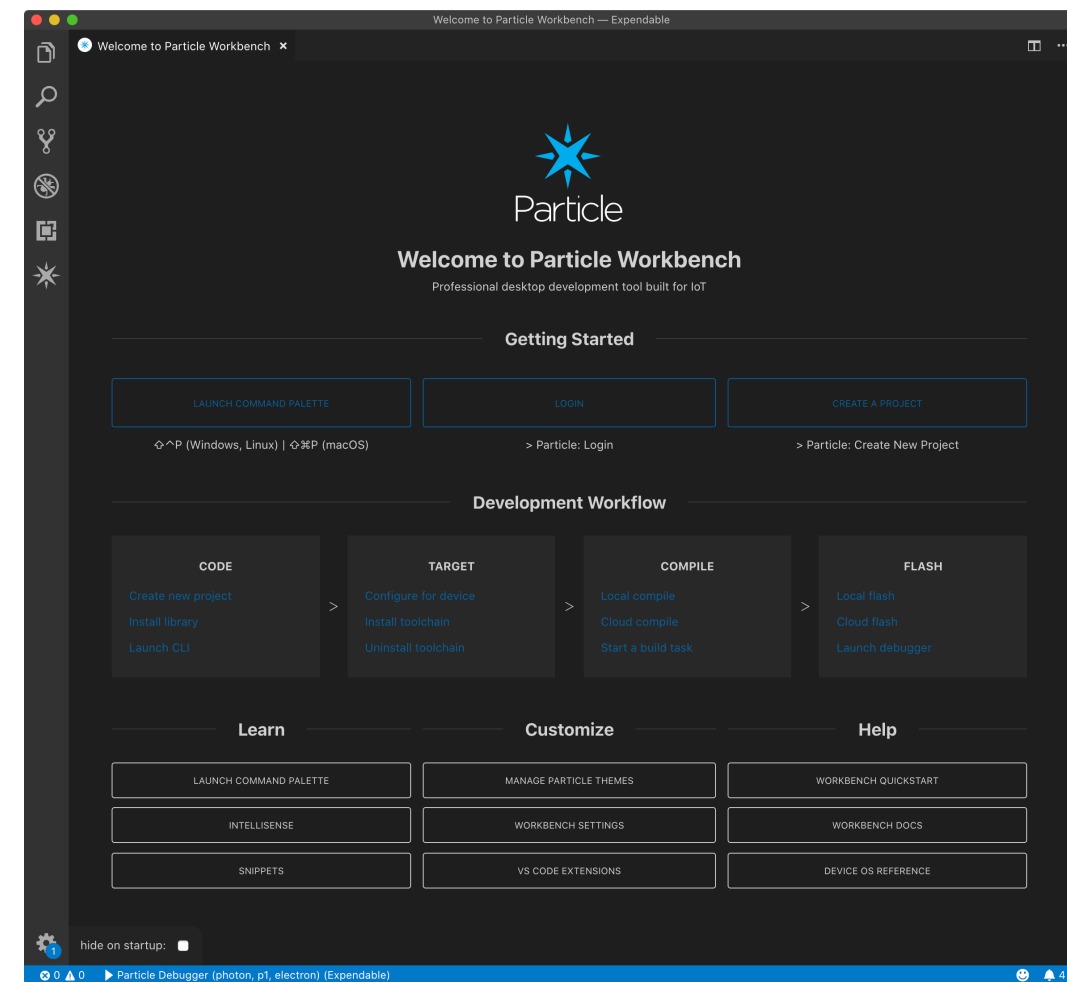
- The micro-controller is an STM32F205 120MHz **ARM** Cortex M3. It uses Harvard Architecture and contains a Free RTOS (Real Time Operating System).
- 1MB flash (non volatile), 128KB RAM (volatile)
- Single band 2.4GHz IEEE 802.11b/g/n with a built-in antenna 
- Supports wireless data rates of up to 65 Mbit/s 
- Ultra low power sleep, stand-by and stop modes
- Supports Open, WEP, WAPI, WPA and WPA2-PSK WiFi security modes
- Recommended Input voltage - +3.6-+5.5V
- ARM JTAG Hook-up

# Getting Connected

1. Sign up for an account at [particle.io](https://particle.io) (hopefully done before class)
2. Follow steps 1 & 2 under [Connect Your Photon](#)

# Programming the Photon

- Download and install Particle Workbench
- This process installs both Visual Studio Code (if not already on your system) and Workbench Dependencies
- Launch Particle Workbench
- We will take a tour during class



# Programming Hello, Photon!

- 1.LOGIN to Particle
- 2.Click on CREATE A PROJECT
- 3.Choose the project's parent folder (if prompted)
- 4.Give it a title, **Hello-Photon**
- 5.In Hello-Photon.ino write the code at right
- 6.Choose Configure for device
  1. Choose DeviceOS (1.2.1.), Target (Photon)
  2. Enter Device Name
- 7.Click on the flash to flash it to your Photon - after a brief light show (including flashing magenta), the little blue LED will start to do its thing

```
// Hello, Photon!  
void setup() {  
    pinMode(D7,OUTPUT);  
}  
  
void loop() {  
    digitalWrite(D7,HIGH);  
    delay(500);  
    digitalWrite(D7,LOW);  
    delay(500);  
}
```

# A Brief Explanation

- setup(), a C function, is called once, when the firmware loads
- loop() is called ad-infinitum
- pinMode() determines the behavior of a specific pin
- digitalWrite() writes a signal (HIGH or LOW) to the specified pin
- delay() specifies a delay, in ms

```
// Hello, Photon!  
void setup() {  
    pinMode(D7, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(D7, HIGH);  
    delay(500);  
    digitalWrite(D7, LOW);  
    delay(500);  
}
```

N.B.: after each call to loop(), the Photon must ping the cloud. To ensure that this happens frequently enough, do not put a lengthy (or infinite) loop inside loop()

# Exercises

- Write/verify/flash a program that blinks D7 on for 1.5 seconds and off for 0.5 seconds
- Write/verify/flash a program that blinks D7 on for 1 second, then 2 seconds, then 3 seconds ... 4 seconds (with a delay of 1 second between on cycles)
- Modify the previous program so that it counts 1,2,3,4,3,2,1,2,3,4 ...
- Explain the difference between a microcontroller and a microprocessor.
  - Is a Raspberry Pi a Microcontroller or Microprocessor?
- True or false: the Photon must be plugged in to USB in order to program it

# In Case of Emergency ...

- **Resetting Wi-Fi:** hold SETUP for 10 seconds until the LED blinks blue rapidly
- **Safe Mode** (stops application firmware from loading - useful if you make a mistake that prevents loop() from executing):
  1. Hold both SETUP and RESET
  2. Release RESET (continue to hold SETUP)
  3. Wait until LED blinks magenta
  4. Release SETUP





# Resources

- [docs.particle.io](https://docs.particle.io)
- <https://docs.particle.io/datasheets/photon-datasheet/#functional-description>
- <http://www.freertos.org/about-RTOS.html>