



Photon Basics & Setup

Hardware Overview

Particle Photon: A small Wi-Fi enabled development kit for creating connected projects.

Key Features:

- Broadcom Wi-Fi chip
- STM32 ARM Cortex M3 microcontroller
- Built-in RGB LED (D7)
- Headers for easy prototyping

Setting up the Photon

1. **Particle Account:** Create an account at particle.io.
2. **Particle CLI:** Install the Particle Command Line Interface (CLI) using npm:

```
npm install -g particle-cli
```

3. ****Claiming the Device:**** Use the Particle CLI to claim the device to your account. Put your Photon in listening mode (blinking blue) by holding the SETUP button. Then run: `bash particle setup`
`...`
 Follow the prompts to connect to your Wi-Fi network.

Important Pins

D0-D7:	Digital I/O pins
A0-A7:	Analog input pins
TX/RX:	Serial communication pins
VIN:	Voltage Input (3.6V - 12V)
3V3:	3.3V output
GND:	Ground

Coding with the Photon

Basic Structure

```
// Setup function (runs once)
void setup() {
  // Initialize code here
}

// Loop function (runs repeatedly)
void loop() {
  // Main program code here
}
```

Digital I/O

<code>pinMode(pin, mode);</code>	Sets the mode of a digital pin (INPUT, OUTPUT, INPUT_PULLUP)
<code>digitalWrite(pin, value);</code>	Writes HIGH or LOW to a digital pin
<code>digitalRead(pin);</code>	Reads the value (HIGH or LOW) from a digital pin
Example:	<pre>pinMode(D7, OUTPUT); digitalWrite(D7, HIGH); // Turn LED on delay(1000); digitalWrite(D7, LOW); // Turn LED off</pre>

Analog I/O

<code>analogRead(pin);</code>	Reads the analog value (0-4095) from an analog pin
<code>analogWrite(pin, value);</code>	Writes an analog value (PWM) to a digital pin (0-255)
Example:	<pre>int sensorValue = analogRead(A0); analogWrite(D0, sensorValue / 16); // Map 0-4095 to 0- 255</pre>

Timing

<code>delay(ms);</code>	Pauses the program for a specified number of milliseconds
<code>millis();</code>	Returns the number of milliseconds since the program started running
<code>micros();</code>	Returns the number of microseconds since the program started running

Cloud Functions & Variables

Cloud Functions

Cloud functions allow you to call functions on your Photon from the Particle Cloud API.

```
int myFunction(String command) {
  // Function logic here
  return 1; // Return a value
}

// Register the function
Particle.function("myFunction", myFunction);
```

Accessing the Function via the API:

```
particle call <device_name> myFunction
"argument"
```

Cloud Variables

Cloud variables allow you to read variables from your Photon via the Particle Cloud API.

```
int myVariable = 0;

// Register the variable
Particle.variable("myVariable", myVariable);
```

Accessing the Variable via the API:

```
particle get <device_name> myVariable
```

Publishing Events

Publishing events allows you to send data from your Photon to the Particle Cloud.

```
Particle.publish("eventName", "data");
```

You can also specify the event's privacy:

```
Particle.publish("eventName", "data",
PRIVATE);
```

Options are `PRIVATE` or `PUBLIC`.

Networking & System

Checking Connection Status

`WiFi.ready()` Returns `true` if the Wi-Fi connection is established and the Photon is connected to the Particle Cloud.

`Particle.connected()` Returns `true` if the Photon is connected to the Particle Cloud.

System Functions

`System.reset();` Resets the Photon.

`System.dfu();` Puts the Photon in DFU (Device Firmware Upgrade) mode for flashing firmware via USB.

`System.sleep(mode, sleep_time);` Puts the Photon in a low-power sleep mode. `sleep_mode` can be `SLEEP_MODE_DEEP` or `SLEEP_MODE_SOFTAP`. `sleep_time` is in seconds.

Error Handling

Use `TRY()` and `CATCH()` to handle errors. `assert()` also can be used to check if the statement returns true.