

Lab 8: Particle Photon WebHookTutorial

Due: Tuesday, November 05, 11:59 PM

1. Setup Particle Photon

Requirements:

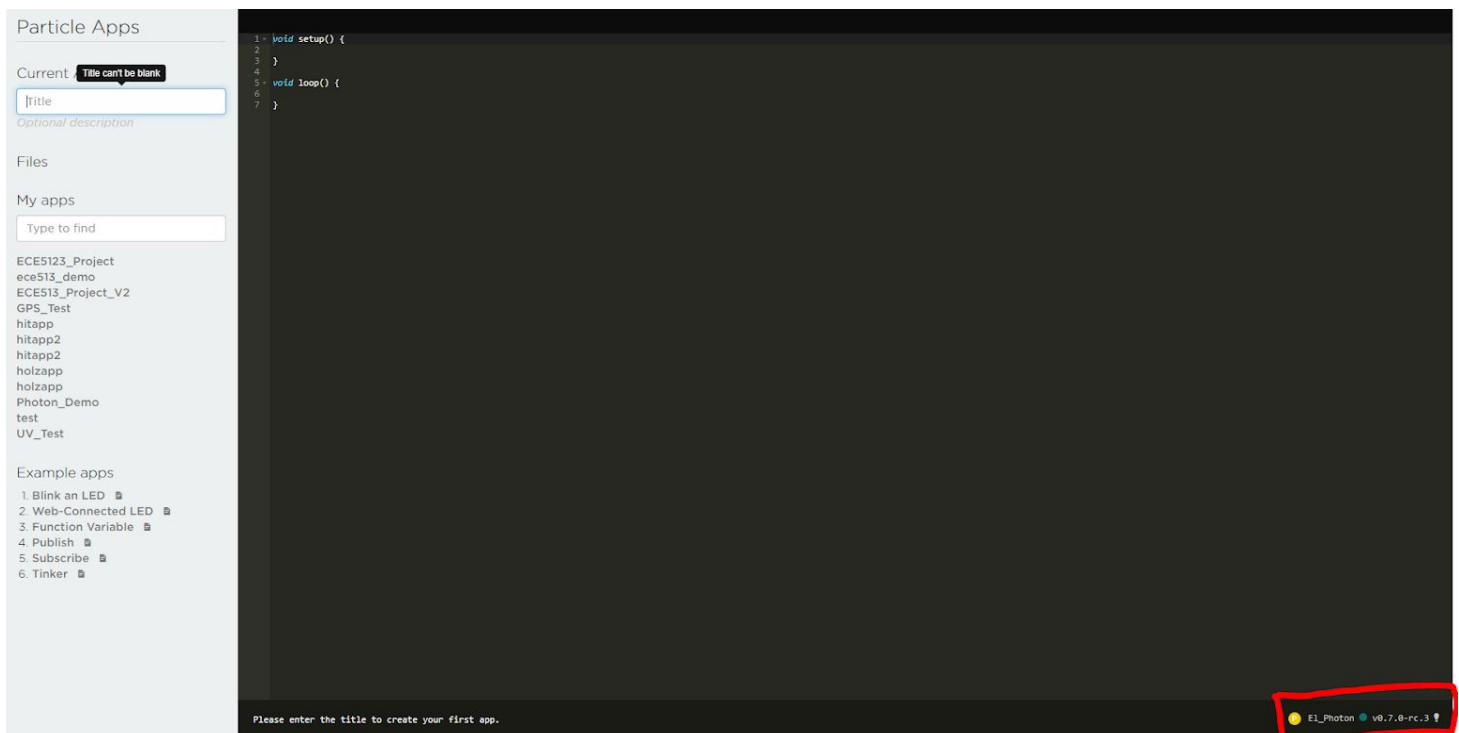
- Particle Photon device
- Internet connection with username and password authentication (UA WiFi not usable)

Optional Tutorials:

- [Particle Photon Basic Setup with GPS and UV Sensor](#)
- [Particle Photon Basic Setup Video Tutorial](#)

2. Get Device ID to Retrieve a Valid API key

- 1) There are two methods of finding your deviceId. Please keep this device ID for later use.
 - a) The easiest is to simply go to first follow <https://build.particle.io> and look for the name of your device in the lower right section of the browser. Click on it, then click on the “>” symbol in the top left section, and you will find the device ID. The following snippets illustrate this process:





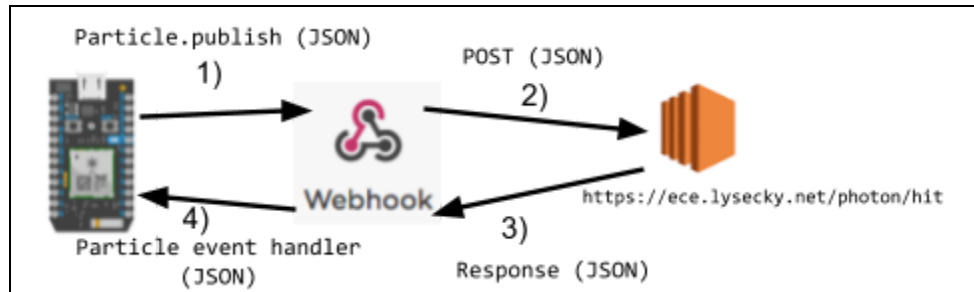
- b) The second method is using the particle CLI previously installed in your computer, follow the instructions in order to retrieve your photon device ID <https://docs.particle.io/support/particle-devices-faq/finding-device-id/> Note that the device is blinking blue (setup mode) for this purpose.

- 2) Go to <https://ece.lysecky.net/json/jsonposter.html> to register an API key for your device. Keep this value for later use.
 - a) Use URL: <https://ece.lysecky.net/devices/register>
 - b) Enter your Photon's device ID and your UA email in the JSON Data
 - i) **Note 1:** You must use your UA email address when registering your device.
 - ii) **Note 2:** One Photon device per student is required.
 - iii) Example:


```
{
                "deviceId": "03265156100a513651635323536",
                "email": "luke_skywalker@email.arizona.edu"
              }
```

3. Publish Event from Photon to Server

The idea is that the Photon board will send information to the Photon server (WebHook) and the WebHook will automatically relay this information with any desired changes to a third party endpoint. The endpoint can send back information to the webhook and the webhook can return this information to the Photon board.



The goal of this tutorial/lab is to create an application for the Photon device that will publish data to a WebHook that forwards the request to the ECE 413/513 server.

- 1) Photon App: detecting a button press and when detected, it publish pre-defined *longitude* and *latitude* to the Webhook

```
{ "longitude": -110.954099, "latitude": 32.234999 }
```

- 2) WebHook then sends a POST request with JSON data to the endpoint
Parameters (JSON)

```
{
  "apikey": "1234",
  "deviceId": "887665535435778776",
  "longitude": "-110.954099",
  "latitude": "32.234999",
  "time": "2017-10-27T06:09:48.825Z"
}
```

Note that you need a valid apikey to access the api. The apikey is embedded in Webhook level instead of device level for better security.

- 3) The endpoint we will be sending this information to is */photon/hit*. This endpoint processes POST requests, check that the received apikey and deviceId are valid, and return responses as follows:

Responses (JSON):

Success (201):


```
{
  "status": "OK",
  "message": "Data saved in db with object ID
59f37c7b1f85ac64671f0ffc.",
}
```

Failure (201):

```
{
  "status": "ERROR",
  "message": "Invalid apikey for device ID 1234",
}
```

- 4) The Webhook will then receive the JSON response from step 3) and send it back to the device that made the request in step 1).

4. Defining a WebHook


1. Go to <https://console.particle.io/integrations>
2. Choose **NEW INTEGRATION** () button
3. Choose Webhook (Push Particle device data to other web services in real-time)
 - a. Event Name: hit **<Note: the event name is case-sensitive.>**
 - b. URL: <https://ece.lysecky.net/photon/hit>
 - c. Request Type: POST
 - d. Request Format: JSON
 - e. Device: Any
4. Select Advanced Setting
 - a. Select "custom" for JSON data format and specify format in textarea

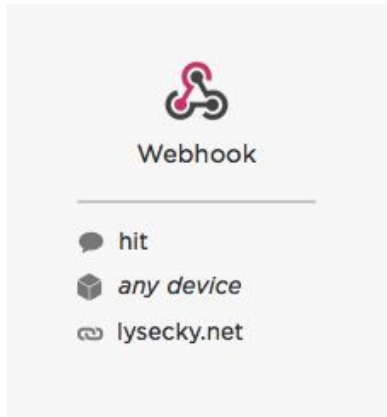
```
{
  "apikey": "PLEASE EMBED YOUR APIKEY HERE",
  "deviceId": "{{PARTICLE_DEVICE_ID}}",
  "longitude": "{{longitude}}",
  "latitude": "{{latitude}}"
}
```

As a quick reference, these are the pre-defined webhook variables available for you to use:

- `{{PARTICLE_DEVICE_ID}}`: The ID of the device that triggered the webhook
- `{{PARTICLE_EVENT_NAME}}`: The name of the event that triggers the webhook
- `{{PARTICLE_EVENT_VALUE}}`: The data associated with the webhook event
- `{{PARTICLE_PUBLISHED_AT}}`: When the webhook was sent

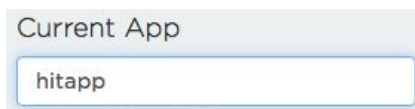
5. Press **CREATE WEBHOOK** (CREATE WEBHOOK) button




6. You can confirm that the webhook is created by pressing the () button (on the left side) to see your list of webhooks.



5. Particle App Code

1. Go to <https://build.particle.io> and log in if needed
2. Type app name: hitapp



3. Copy and paste (overwrite) the following code into editor
4. Press save button ()
5. Press verify button ()
6. If no error, then flash code (), once you see **Flash successful! Your device is being updated..** then it's good to go.

```
1  /*****
2   * Button hit logger
3   * Embedded Systems Design Lab 10/21/2019
4   */
5
6  // assign SETUP button's pin
7  int button = BTN;
8
9  // Setup
10 void setup() {
11     // Setup serial port
12     Serial.begin(9600);
13     // Setup pin mode for button
14     pinMode(button, INPUT);
15     // Setup webhook subscribe
16     Particle.subscribe("hook-response/hit", myHandler, MY_DEVICES);
17 }
18
19 // main loop
20 void loop() {
21     // read button and if it is pressed
22     if (digitalRead(button) == 0) { // pulldown resistor, 0: Pressed
```

```

23         // Construct json string
24         String data = String("{ \"longitude\": -110.954099, \"latitude\": 32.234999 }");
25         // Log to serial console
26         Serial.println("button pressed!");
27         // Publish to webhook
28         Particle.publish("hit", data, PRIVATE);
29     }
30     // delay .5 second to block continuous input
31     delay(500);
32 }
33
34 // When obtain response from the publish
35 void myHandler(const char *event, const char *data) {
36     // Formatting output
37     String output = String::format("Response from Post:\n  %s\n", data);
38     // Log to serial console
39     Serial.println(output);
40 }

```

Some details of the code:

- Line 7 assigns button with BTN (SETUP button on Photon)
- Line 12 initializes serial port communication with baud rate 9600 (for serial debugging purpose)
- Line 16 registers (subscribes) *hit* webhook and its handler (line 31).
- Line 24 defines JSON data that will be injected into publish in line 28. In this process, the WebHook replaces **data** with the JSON format defined in the WebHook creation. **NOTE that double quotation mark should be escaped as above.**
- Lines 34-4 format the response data from the server (through the WebHook) and print the response to the serial port.

5. Demo and Lab Submission

1. Open a new terminal to connect to your device via serial port using previously installed particle-cli by typing "particle serial monitor".
2. Press the setup button on the Photon.
3. If you can get the following JSON data, it means your request was successfully saved in the database.

In host computer that is connected with Photon
(Terminal or command prompt)

```

~$ particle serial monitor
Opening serial monitor for com port: "/dev/cu.usbmodem1481"
Serial monitor opened successfully:
Press SETUP button on Photon
button pressed!
Response from Post:
{"status":"OK","message":"Data saved in db with object ID 59f3988894685769b93c78e3."}

```

References

- Cloud functions: <https://docs.particle.io/reference/firmware/photon/#cloud-functions>
- Language syntax: <https://docs.particle.io/reference/firmware/photon/#language-syntax>
- C standard library used on the Photon: <https://sourceware.org/newlib/libc.html>
- WebHooks: <https://docs.particle.io/reference/webhooks/>