Background

Microcontrollers provide the first line of machine intelligence in today's everyday devices. For example, a Keurig coffee pot contains a microcontroller. This device takes input from the user (buttons), water level sensors, temperature sensors, clocks, and operates a series of pumps, motors, and valves to produce a cup of coffee when required. Everything from coffee pots to quadcopters have at least one microcontroller embedded in them. The microcontroller transforms everyday devices into smart devices.

The big idea behind the Internet of Things (IoT) is adding communication capabilities to items that we use every day. Smart devices are evolving to perform communications through wifi or cellular connectivity. Much like our entertainment, business, and our personal lives have changed significantly because of mass communication networks, adding communication opens up entirely new possibilities in smart devices. For example, a regular parking meter for a parking spot cannot tell you whether a car is parked in the spot unless you physically go to the spot and look. On the other hand, an internet-connected parking meter could report to you over the internet whether a car is parked in a spot, saving you the trouble of traveling to the spot and looking. Taking this one step further, a network of internet-connected parking meters could report parking availability for every parking spot in the city. A cellphone app looking at the information reported by the parking meters could tell you what is the closest available parking spot to your current location, saving you the trouble of trolling the streets to find a place to park. This is the power of the internet of things.

In this lab we'll be utilizing a microcontroller with an embedded WiFi interface as our communications capability. This will allow you to interact with your device, communicate with the outside world (or other devices), and program your device wirelessly.

Purpose

The purpose of this lab is to familiarize you with your new Particle microcontroller, perform first time configuration of your new Particle microcontroller, create a basic “Hello World!” program, and provide you with a basic knowledge of the Particle Photon microcontroller.

Supplemental Resources

https://www.particle.io
http://docs.particle.io/photon/
Photon - Common Issues (ECE maintained Photon debugging help)
Orientation
The Photon operates on power over USB (similar to how you charge your cell phone). It has several interface components described in the diagram to the right. The most important of these are:

- #2 the Setup button
  - Used to put the device into listen mode (over USB or Wifi)
- #3 the Reset Button.
  - Used to reboot the Photon
- #4 the RGB status LED
  - The color of this LED shows the status of the Photon
  - See appendix “Device Modes” for more details

There is a nice overview of the Photon available online at:
http://docs.particle.io/photon/

First-time Setup
This section describes how to get your Photon attached to our WiFi network and how to “claim” your photon. “Claiming” your Photon is the process by which you lock your Photon to your online account at particle.io. This will secure your Photon so that it only responds to you or someone you provide with a security key.

1. Plug in your Photon to the white breadboard (with lots of holes) as shown in the figure on the right.

   NOTE: The breadboard is oriented such that the blue line is on top.

2. Plug the small end of the USB cable into the Photon.

   NOTE: do not force the USB cable into the socket of your Photon board: you might be trying to connect it upside down and might break the socket. Do NOT plug in the other end of the USB cable to your computer yet.
The rest of this section is split into 2 parts:

- USB-based setup
- Claiming your Photon

Note: This document assumes you are in an area with access to the “CSE IoT” Wi-Fi network (ME or EE buildings). If you are getting an early start with your Photon (at home or in a dorm), please check out the Particle Photon - Getting Started page. If you are in a dorm it may be helpful to use a mobile hotspot for an internet connection.

USB-based Setup with Particle Workbench (All OSs)

3. Plug in your Photon to your computer via the USB cable.
4. Check that your photon is quickly flashing “blue” (not Cyan! See this [link](https://docs.particle.io/quickstart/workbench/) to view the different mode color flashes.)
   a. If it is flashing blue go to the next numbered step below.
   b. If it is *not* flashing blue, press and hold the SETUP button for 10 seconds. Your photon should start flashing blue.
   c. If your Photon still will not flash blue, contact your TA or Professor.
5. Install VS Code with the Particle Workbench extension:
   [https://docs.particle.io/quickstart/workbench/](https://docs.particle.io/quickstart/workbench/)
   (Using your own computer and already have VS Code installed? Use [this tutorial](https://docs.particle.io/quickstart/workbench/) for this step.)

   When VS Code asks you to also install Workbench and its dependencies, say NEXT and wait until the purple progress percentages at the bottom of the VS window show completion of all installations, and the small boxes at the bottom-right of the VS window stops saying “installing CLI dependencies”. Stop following the directions in the particle documentation (linked above) when you get to the heading “Create a Project”. We will do that later.

4. Wait for VS Code to start up, if you see the message “Please install Particle Workbench dependencies”, click “install”.
5. Wait for this install to complete, it may take a couple minutes. You should see a message that looks something like this:

   ![Installed CLI dependencies for v2.0.1](https://docs.particle.io/quickstart/workbench/)

6. Click the Particle Button [x] on the left side of the VS window.
8. Get your Device ID by typing the command:
particle serial identify

You should see something like this:

Example of retrieving Device ID

7. Cut-n-paste this Device ID number into a text file (we recommend Notepad) and save it because you will need it in the next section.
8. Next, connect your Photon to the campus IoT Wi-Fi network by typing this command:
   particle serial wifi
9. When asked to scan Wi-Fi networks, answer “n” for no.
   (Note: at home, say yes to see your home wifi)
10. For SSID, enter:
    CSE-IoT
    (Note: at home, select your home wifi)
    (case sensitive and the capital “i” in IoT is not an L!)
11. If asked for the security type, select using the arrow keys and hit enter:
    WPA2  (for cipher type select AES + TKiP)
12. For Wi-Fi Password, enter:
    Go!Gophers
    (also case sensitive)
    (Note: at home, enter your home wifi password)
12. It should look something like this if you did it correctly:
Example of a successful network setup using the Particle CLI in VS Code w/ Workbench

13. Your photon **should** now reboot on its own. Watch the LED closely!
   a. The light flashed blue a couple of times and your photon is blinking blue again. In this case press the RESET button and continue below.
   b. The light is blinking some combination of green-slow, green-fast, cyan-slow, cyan-fast then it is trying to connect. Wait a full 2 mins, then contact your TA if it doesn’t change.
   c. If it is flashing magenta. Your photon is trying to complete a firmware update, do not interrupt this stage! Just wait!
   d. The light is slowly “pulsing” cyan.
      Congratulations your Photon is connected to the network! Continue on to the lab section “Claim Your Photon”.

If your photon doesn’t pulse cyan (connected) and isn’t flashing magenta (firmware update), press RESET one more time and wait an additional 2 mins, then contact your TA for assistance.

**NOTE:** If you typed something incorrectly, press and hold the SETUP button for ~20secs. It will blink blue, then flash blue rapidly. Release the button and your Photon will clear its memory of all incorrect Wi-Fi information. Click the button, then begin again with step 5 in this procedure.

**Allowing Your Device to Connect to Additional Wifi Networks**

If you want to set up your Photon to connect to more Wi-Fi networks (e.g., your home network), then you can connect it to your computer’s USB port, and press and hold the SETUP button for about three seconds. The Photon will enter the “listen mode” again (more details can be found in the section called **Appendix - Resetting Your Device into Listen Mode** in this document).

**NOTE:** Every set of Wi-Fi settings, may add ~5 secs to the reboot time of your Photon.
What Just Happened?
Just to make sure you don’t think of any of this as black magic, here’s the summary of what we just did: the Photon board can do many things, including communicating with sensors and actuators in its environment, AND talking to other entities on the web. We have not utilized any of that yet. All we did was to ask the Photon to listen to the USB port. We sent a “command” by entering the command “particle serial identify”, and it responded by sending us its ID in the text format. The terminal command particle serial on our computer knew how to decipher the electric signals coming back on the USB cable and translate it to the message you saw on the screen.

As of now, your Photon board cannot talk to anyone on the web. In Lab 1, we will configure the board to use actuators. In Lab 2, we will explore how to listen to commands, how to send data, and take actions over the internet. Next, we will setup and explore the integrated development environment (IDE) for the Photon.

Claim Your Photon
In this section we will create a new user account at Particle.Io and claim your Photon. Doing this locks the unique ID of the Photon to your user account. This prevents other people from overwriting your code and taking control of your Photon.

1. Go to the website: https://build.particle.io/build (we recommend you bookmark this page.) You should see the following prompt the first time you connect:

2. Click “create account”
3. Type in your UMN (or other) email address and create a new password (better not to use your UMN password here for security reasons).
4. Once you fill in the form, the Particle web-based development environment will appear.
5. Select the “Devices” button

![Devices button](image)

6. Click “Add New Device” (left side, near bottom)
7. Type in your giant hex string device ID recorded previously; for example “54ff6b01112524821230167”
8. Click “Claim a Photon”
9. Name your Photon, your name should be short, unique, and memorable. For example your X.500 short name, followed by “Photon” (e.g., OrserPhoton)

What Just Happened (Wifi setup)?

You just asked the wireless router to assign an IP address to your Photon board. From this point on, your Photon has the ability to be powered by a power source (image on right) and not connected to a laptop, and still be able to function and be reprogrammed with new functionalities. We will not necessarily use that feature right now, but you will be doing so when you get more comfortable with the tools and programming.

The ID number you entered claims your Photon to your account. This allows secure communication between the particle.io servers, your IDE (Integrated Development Environment), and the Photon. Using this secure channel we can send and receive commands.

“Hello World!” - Web IDE

In this section we will create a very basic program to test the connectivity and function of your new Photon. Any code you write, needs to be “compiled”, which means it has to be translated from a human-readable text file to 0’s and 1’s that Photon understands. The code (also called a “program”) needs to be “flashed” on the Photon, which means the 0’s and 1’s have to be transferred to Photon’s memory so that the Photon board can execute the program.
In the left pane of the window of your browser (https://build.particle.io/), click on < > on the black bar on the left. In the gray area, you see a list of “Example apps”. These are pre-made applications developed by Photon designers.

1) Click on the “Blink and LED” example.

2) Click on the big blue button

![USE THIS EXAMPLE](image)

3) Click on the “Verify” button on the top-left corner.

4) Click on the “flash” button on the top-left corner of the screen.

5) You should see your Photon go through a series of changes: solid or blinking magenta, briefly showing white, then maybe multiple iterations of blinking magenta, white, off. During this process, the Photon might reset (also called restart) by blinking green and looking for wifi connections. Finally it will go back to “breathing” cyan. Congratulations! You have just programmed your Photon through the Web IDE.

“Hello World!” - Particle Workbench (Local IDE)

An alternative method of programming your Photon is to use the local compiler: VS Code + Photon benchmark that you setup at the beginning of this lab. We prefer this method because the IDE provides you with many interesting features, and also integrates a command-line interface (CLI) that allows you to monitor what is going on in the program.

1) Go back to the VS Code window (or open it again if you have closed it), and connect your Photon to your computer using the USB cable if it is not already.

2) Click on the Particle button (), the select “LOGIN”. It should look like this:
3) You will see a popup window at the top of the screen that looks like this:

![Login Window]

4) Enter your username, password, and (opt) auth token.

5) Go back to the particle setup documentation 
   https://docs.particle.io/quickstart/workbench/ and scroll down to the “Create a Project” heading. When the pop-up window appears, continue reading this item. Note that the particle extension on VS code does not allow spaces in the project path. Also, because 
   Google Drive has a space, you are not able to use a Google Drive Folder.

6) In the dialog box that shows up, you can -- if you want -- create a sub-folder under “Documents” such as VSCodePrj (blue mark in the image below) and click to go inside it. Finally, select the folder you are in as the parent for the project (red markings):

![Dialog Box]

7) Continue the instructions on Particle documentation and stop when you get to the heading “Compile a project”.

---

Copyright 2018
8) In the left pane, click on the small > sign next to “src”.

You can see that VS Code has already generated a code xxxx.ino (where xxxx is the name you gave your project) with some text filled in already. Click on the file name.

9) Select all text in the source file, delete, and replace with the following code:

```cpp
// Source from particle.io
// ------------
// Blink an LED
// ------------

/*-------------
We've heavily commented this code for you. If you're a pro, feel free to ignore it.
Comments start with two slashes or are blocked off by a slash and a star.
You can read them, but your device can't. It's like a secret message just for you.
Every program based on Wiring (programming language used by Arduino, and Particle devices) has two essential parts:
setup - runs once at the beginning of your program
loop - runs continuously over and over
You'll see how we use these in a second.
This program will blink an led on and off every second.
It blinks the D7 LED on your Particle device. If you have an LED wired to D0, it will blink that LED as well.
-------------*/
```

**STOP and READ this!**
Do not cut-n-paste code from the lab manuals! PDFs and many other text sources can contain hidden characters. When pasted into the IDE editor these hidden character will cause compilation errors. You have been warned.
// First, we're going to make some variables.  
// This is our "shorthand" that we'll use throughout the program:  

int led1 = D0;  // Instead of writing D0 over and over again, we'll write led1  
// You'll need to wire an LED to this one to see it blink.  

int led2 = D7;  // Instead of writing D7 over and over again, we'll write led2  
// This one is the little blue LED on your board. On the Photon it is next to D7, and on the Core it is next to the USB jack.  

// Having declared these variables, let's move on to the setup function.  
// The setup function is a standard part of any microcontroller program.  
// It runs only once when the device boots up or is reset.  

void setup() {  
  // We are going to tell our device that D0 and D7 (which we named led1 and led2 respectively) are going to be output  
  // (That means that we will be sending voltage to them, rather than monitoring voltage that comes from them)  
  // It's important you do this here, inside the setup() function rather than outside it or in the loop function.  
  pinMode(led1, OUTPUT);  
  pinMode(led2, OUTPUT);  
}

// Next we have the loop function, the other essential part of a microcontroller program.  
// This routine gets repeated over and over, as quickly as possible and as many times as possible, after the setup function is called.  
// Note: Code that blocks for too long (like more than 5 seconds), can make weird things happen (like dropping the network connection). The built-in delay function shown below safely interleaves required background activity, so arbitrarily long delays can safely be done if you need them.  

void loop() {  
  // To blink the LED, first we'll turn it on...  
  digitalWrite(led1, HIGH);  
  digitalWrite(led2, HIGH);  
  
  // We'll leave it on for 1 second...  
  delay(1000);  
  
  // Then we'll turn it off...  
  digitalWrite(led1, LOW);  
  digitalWrite(led2, LOW);  
  
  // Wait 1 second...
10) Before you can compile and flash the code, you need to tell VS Code which Photon device you want to use. Click on "<select device>" on the blue bar at the bottom of the window:

11) Enter the device name or the device ID that you had saved (when you had used “particle serial identify”). Note that the particle documentation you used to install VS Code shows an alternative way of entering the device id in the section “Compile a project” (https://docs.particle.io/quickstart/workbench/#compile-a-project).

12) To compile the code locally on your computer, you can click on the compile (was called verify in the Web IDE) button at the top right bar of the code window. If you don’t see this button, you probably need to click on the source file for the button to show. Compiling the application locally might take some time.

13) Once compilation is done, you can flash the code using the “flash” button on the top-right corner of the screen. If you see an error message about the USB connection not being recognized, talk to your instructor or TA. You might have noticed that you can launch the CLI window using the button in that group too.

14) An alternative way of compiling and flashing your code in VS Code is to use the “cloud” compilation and flash capabilities. The instructions for this type of compilation and flash are at the very bottom of the page in the particle documentation (https://docs.particle.io/quickstart/workbench/#compile-a-project). Command Palette can be invoked by pressing ⇧⌘P (macOS), ⇧P (Windows, Linux).

15) Do a cloud compile too, just to learn how to do it. Sometimes local compilation does not work.

15) Edit the code such that your LED blinks at two times per second. Demo this to your TA, show them your code, and answer any questions they might have.
Appendix - Photon Pinout Diagram
Many pins in the Photon can be utilized in a variety of ways. The following pin diagram may be useful when deciding which outputs to use on a project.

More details available at:
http://docs.particle.io/photon/photon-datasheet/#3-3-pin-out-diagrams

Appendix - Resetting Your Device into Listen Mode
Listen mode is indicated by a flashing blue (not cyan) light. This indicates the Photon is awaiting configuration instructions.

1.) Hold the SETUP button for 3-6 seconds, wait for the light to flash blue
2.) If this doesn’t work, press and release RESET, then press and hold the SETUP for 3-6 seconds.
3.) If your Photon is still unresponsive at this point, contact your TA.
Appendix - Clearing WiFi List

Every time you enter new wireless network credentials it adds them to a list inside the Photon. The Photon goes through this list every time it boots up and tries each connection until it finds one that works. This can take up to 10 seconds per network! If you enter incorrect information 10 times, you will have to wait 2 minutes to connect each time your Photon reboots!

Simply press and hold the setup button for a little over 10 seconds to clear the WiFi credentials list. It should flash blue rapidly and then switch into Listen Mode. You will now need to re-enter the correct Wifi credentials as instructed in the First-time Setup section.

Photon Documentation Page - Wi-Fi Network Reset

Appendix - Debugging Help

For advanced debugging help, including the flash green forever issue: Go to the document: https://docs.google.com/document/d/1dfAUDLg2COK4IPWFwb-x4yWlzfrGjCA-fOTO6G3U2Fk/edit?usp=sharing

Appendix - Device Modes

Quick and dirty mode list:
(refer to https://docs.particle.io/guide/getting-started/modes/photon/ for more details)
- Blinking blue: Listening for Wi-Fi setup
- Solid blue: Getting Wi-Fi info from app
- Blinking green: Connecting to the Wi-Fi network
- Blinking cyan: Connecting to the Particle Cloud
- Blinking magenta: Updating to the newest firmware
- Breathing cyan: Connected!

Reset diagram of Photon mode selection

To reset your photon, press and hold the SETUP and RESET buttons, release RESET...wait, then...release SETUP when the following colors show up to access that mode:
Appendix - (Optional) First-time Setup Outside of Lab

Should you be interested in getting a jump start on using your Photon in a location where the EE-Labs network is not available, this section and the Particle website will get you started.

Requirements

- Particle Photon
- USB Cable for Power Only
- Android or iOS based cell phone
- Access to a 2.4 GHz WPA2-PSK wireless network (UofM, UofM_Secure, etc. do not work!)

Note: You will want to make sure only one new Particle Photon is available on your local wireless network at one time.

To setup your new Particle Photon, follow the setup instructions available at http://docs.particle.io/photon/start/
Glossary of Frequently Used Terms

- **Serial Port** - a communication physical interface through which information transfers in or out one bit at a time --Wikipedia; Examples of serial ports are USB and RS-232
- **Terminal** - A text terminal, or often just terminal (sometimes text console) is a serial computer interface for text entry and display. --Wikipedia; Examples of terminal programs in modern computer are PuTTY or CoolTerm
- **Microcontroller** - A microcontroller (sometimes abbreviated µC, uC or MCU) is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. --Wikipedia
- **Firmware** - In electronic systems and computing, firmware is a type of software that provides control, monitoring and data manipulation of engineered products and systems. --Wikipedia
- **Forking a program** - In software engineering, a project fork happens when developers take a copy of the source code from one software package and start independent development on it, creating a distinct and separate piece of software. --Wikipedia
- **Flashing a microcontroller** - Flashing involves the overwriting of existing firmware or data in programmable memory (ie., EEPROM) present in an electronic device with new data.
- **Sensor** - In the broadest definition, a sensor is an object whose purpose is to detect events or changes in its environment, and then provide a corresponding output. --Wikipedia
- **Actuator** - An actuator is any device that makes physical changes in the real world (from LEDs to motor.)
- **API** - In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. --Wikipedia
- **Device ID (and how to get yours)** - In reference to your Photon: Click on

  ![Devices](image)

- **Access token (and how to get yours)** - In reference to your Photon Click on

  ![Settings](image)