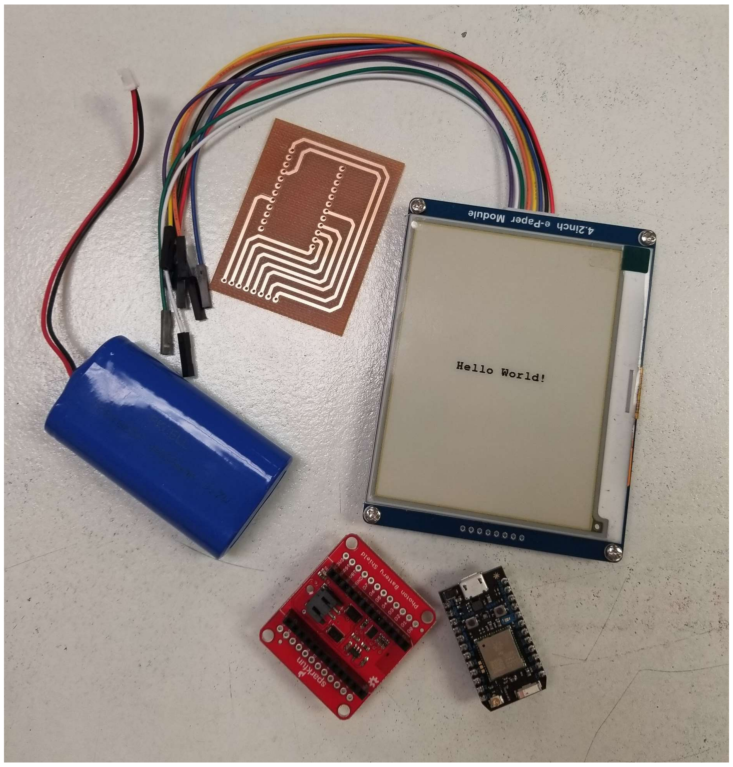
# DIY Office Network Board
# Instructions

Imagine a small, inexpensive device that can be mounted within an interior office window facing outwards (such as those by your professor's door) and used to display messages from the occupant of that office, such as "Back in 5 minutes", "Out to lunch", or "Offices hours cancelled due to illness".

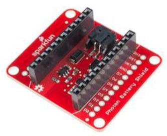
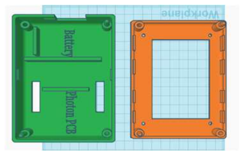
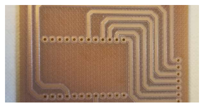


Now imagine that device untethered to any external network connection or wall-based power supply when in use, and also allowing the occupant to update the displayed message regardless of location.

With this project you will be able to create this device for yourself!

The code updates the screen whenever a new message is sent, and turns power off at the end of the day to save power.

## Parts

| | | | |
|---|---|---|---|
| Hardware | | | |
| | 4.2inch E-Ink display module | $35.99 | [1] |
| | Particle Photon with Headers | $19.00 | [2] |
| | Lithium Ion Battery Pack - 3.7V 4400mAh | $19.95 | [3] |
|  | SparkFun Photon Battery Shield | $13.95 | [4] |
| Software apps and Other Equipment Used | | | |
| | Particle Build Web IDE | | |
| | Soldering Iron and Solder | | |
| | Header Pins | | |
|  | 3D Printed Case | | |
|  | Printed PCB | | |

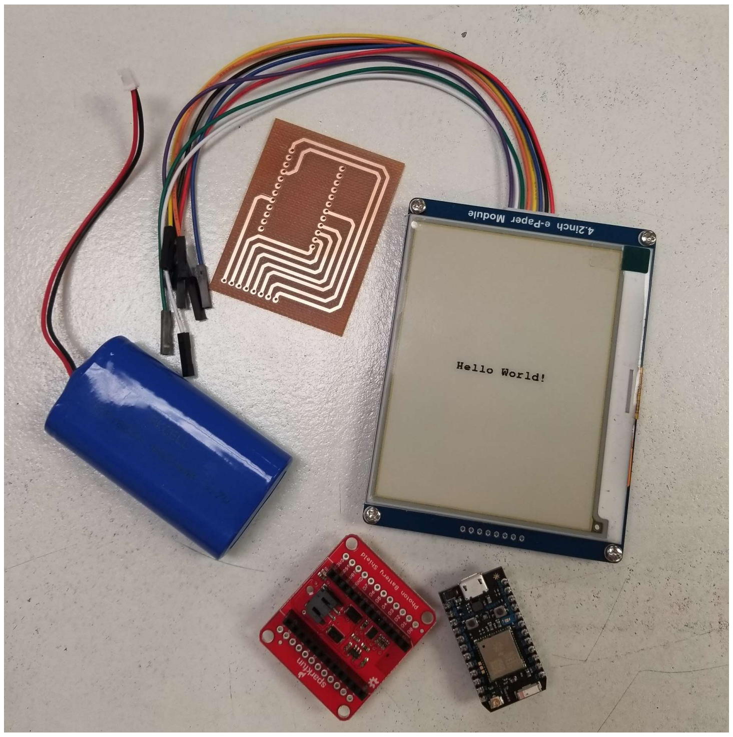
## Story

In the modern world of technology, convenience is key. People are constantly searching for easier ways to connect and communicate in today's fast paced society. The goal of our capstone project is to upgrade the basic concept of posting an update message on a Post-It-Note on your door to design and implement a wireless message board for office personnel or Do-It-Yourself (DIY) hobbyists. The use of this message board will allow a person to conveniently send or update a message to the board from their phone, no matter where that person is located, and the message sent will be displayed appropriately.

I have currently used a prototype of the final design to demonstrate the working of the project. The following are the steps to create this fun project, so let's get started :

## Step 1: Things required to Build the Project hardware

The things required to build this project are :



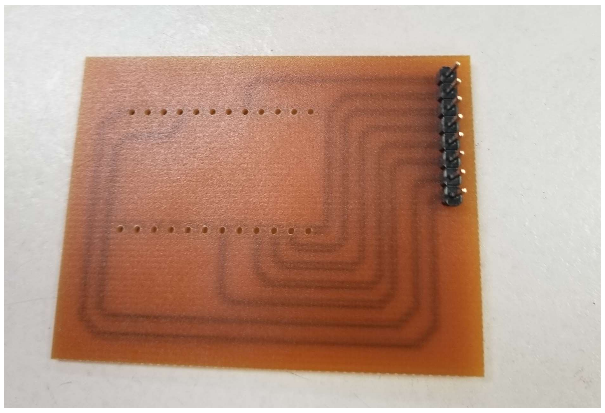
- Particle Photon, the brain of the project
- Waveshare 4.2inch E-Ink display module
- Lithium Ion Battery Pack - 3.7V 4400mAh
- SparkFun Photon Battery Shield
- Header Pins
- Printed PCB Board - Gerber Files within Repository
- 3D Printed Case - Files within Repository

Now that we have all the things required, lets start and assemble our project!

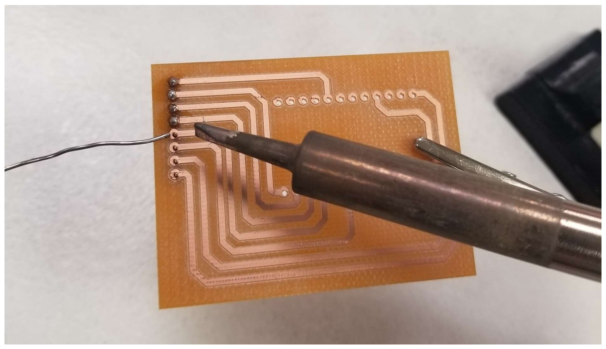
## Step 2: Assembly of the Hardware

Once all the parts are collected, we will prepare the materials to start assembling. First, the PCB will have the header pins soldered to it. The PCB has 8 pin connections at one side where the Waveshare screen wires will be connected.

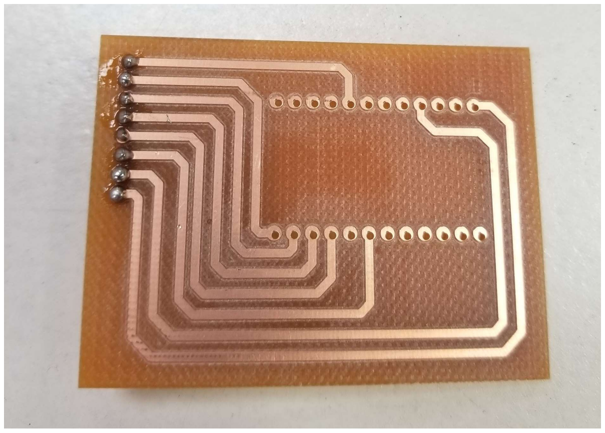
1)



2)

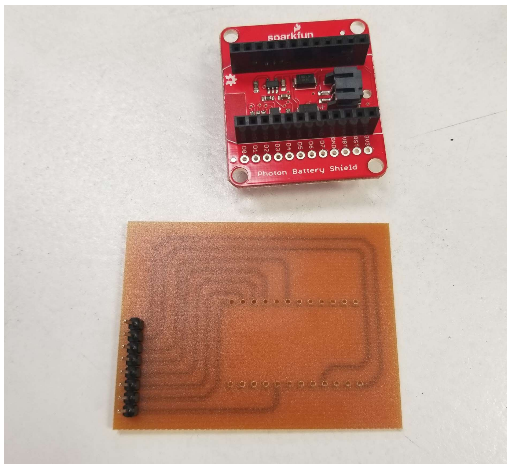


3)

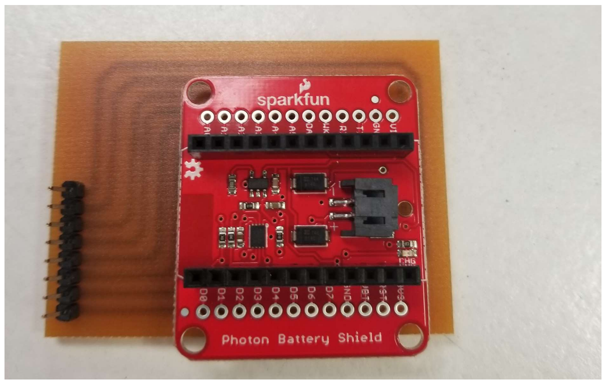


Next, the Photon Battery Shield will be soldered to the PCB. The PCB has a designated spot for the battery shield to sit. Place the battery shield so that the orientation of the battery connector is facing outwards from the PCB, or facing away from the header pins.

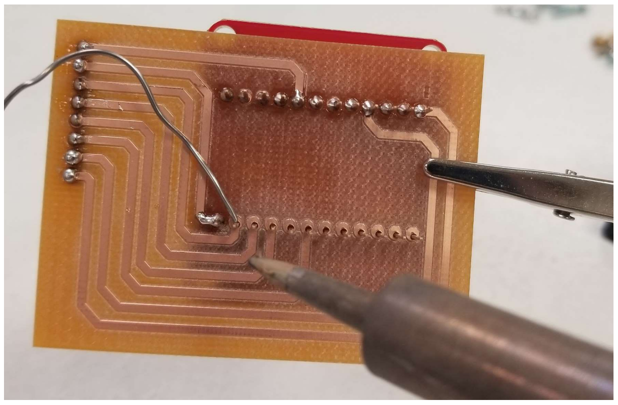
1)



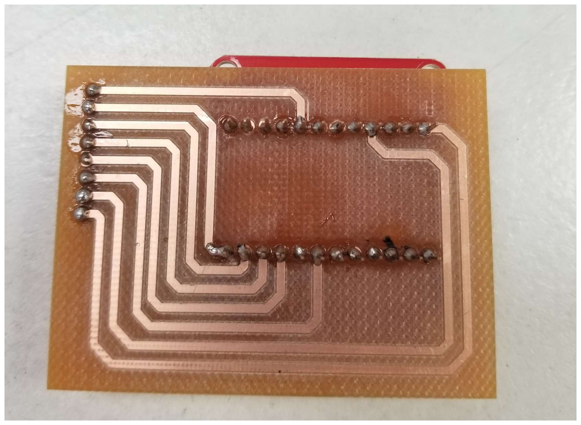
2)



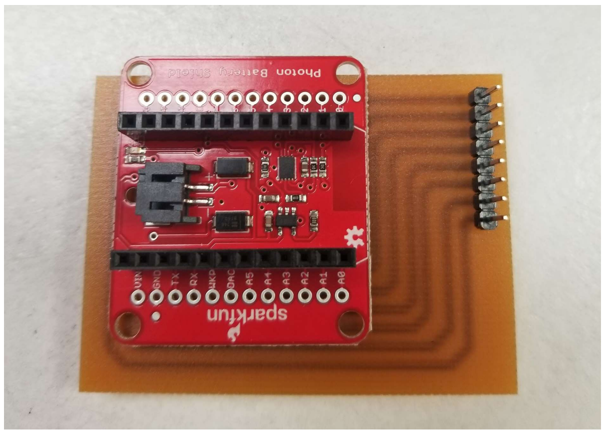
3)



4)



5) **



** Please note that the battery connector on the Battery Shield is facing outwards from the PCB, or away from the header pins.

Next we will do the connections as shown in the following schematic, Figure 1, and we will add the Particle Photon onto the Battery Shield. Place the Photon so that the orientation of the USB connector is facing outwards from the PCB, or facing away from the header pins. Table 1 shows the wire connections between the Waveshare screen and the Photon Board. These wires will be connected via the PCB headers that were just added.

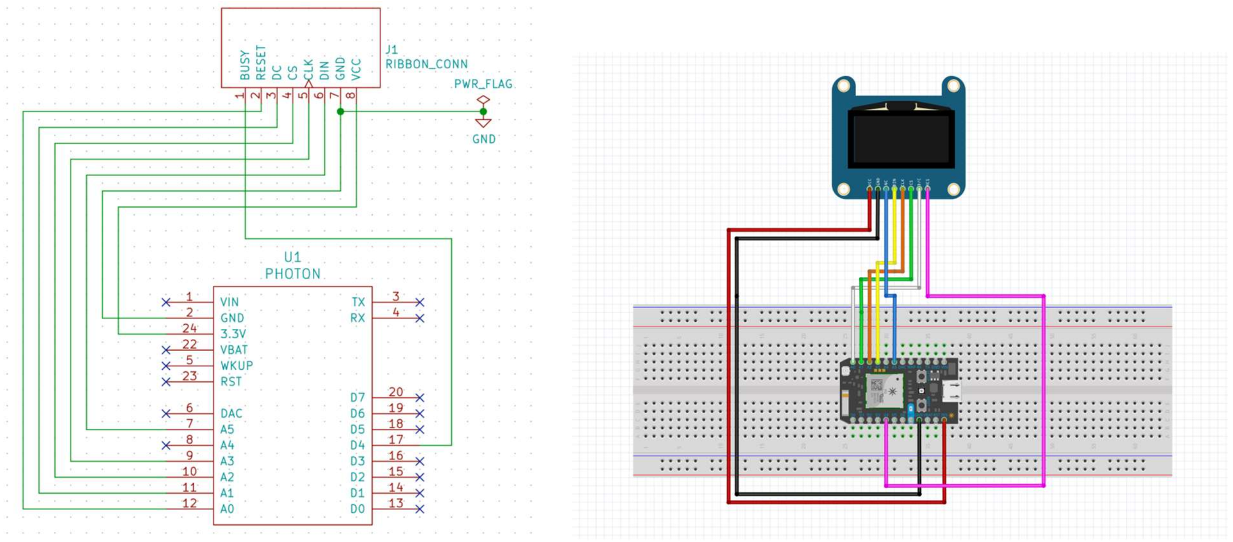


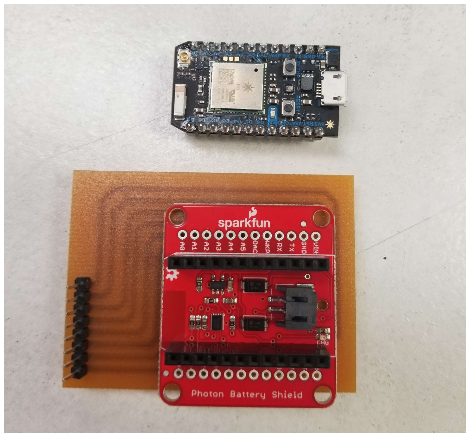
Figure 1: Breadboard Schematic of Connections

Table 1: Waveshare and Photon Wire Connections

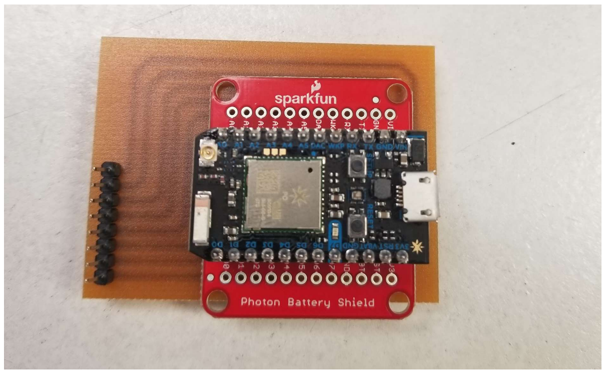
| Hookup | Wire Color | Screen Connection | Photon Connection |
|---|---|---|---|
| 3V | Red | 3V | 3V |
| Ground | Black | GND | GND |
| DIN | Blue | D11 | A5 |
| Clock | Yellow | D13 | A3 |
| CS | Orange | D10 | A2 |
| DC | Green | D9 | A1 |
| Reset | White | D8 | A0 |
| Busy | Purple | D7 | D4 |

After making these connections to the PCB your project will look like this:

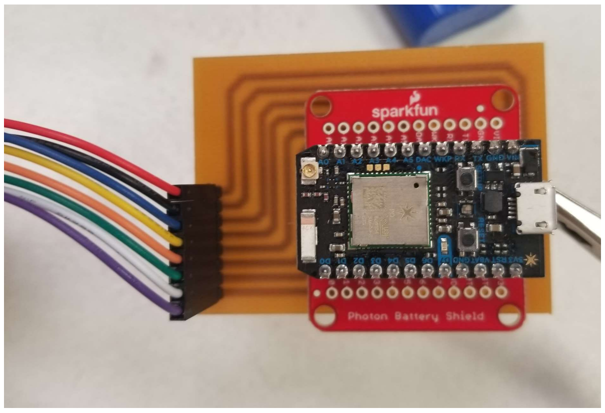
1)



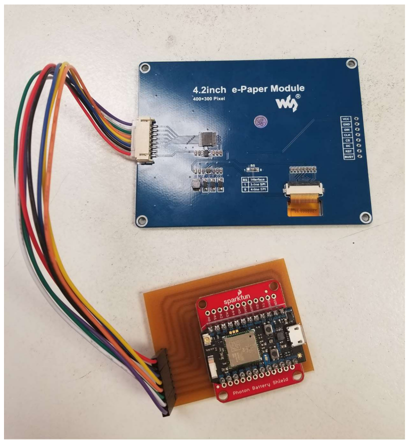
2)



3)

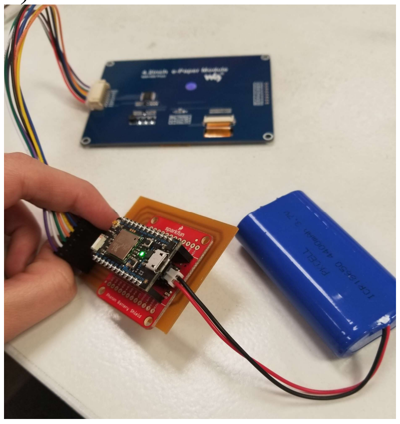


4)



Finally, the battery can be connected to the battery shield as shown here:

1)



**Step 3: Assembly of the Case with Circuit**

Before we can assemble the case with the circuit elements, first it must be printed using a 3D printer. The files for the top and bottom of the case are within the repository. Figure 2 shows the model of the case.

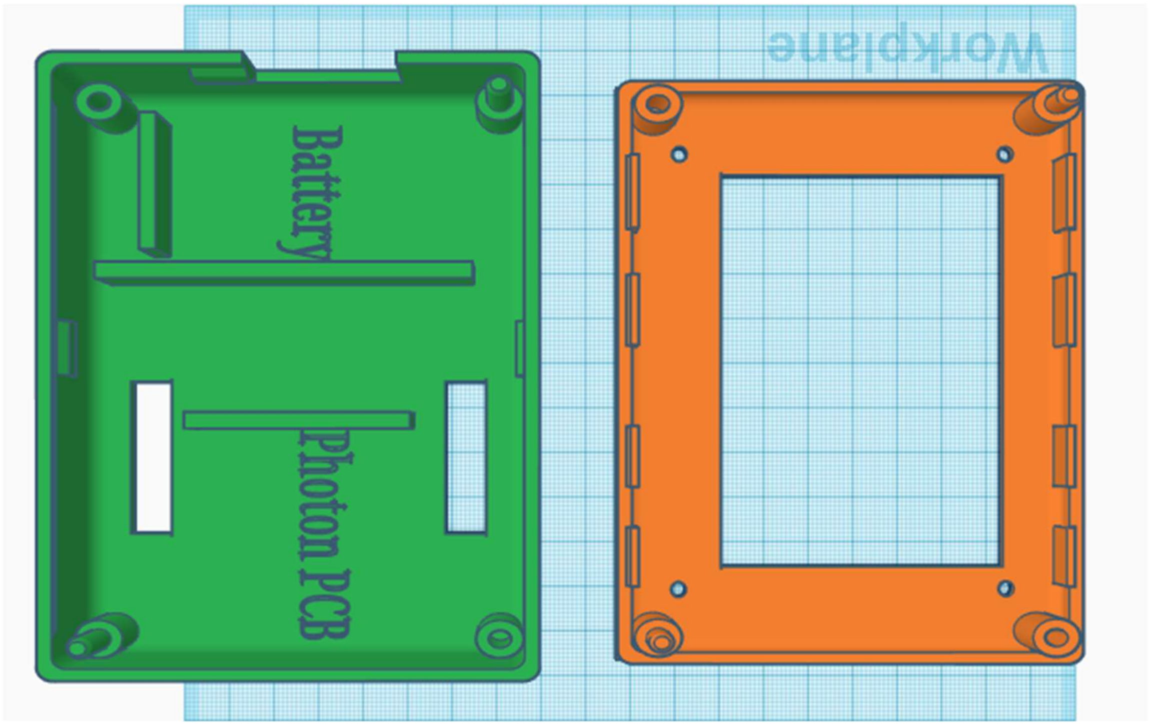
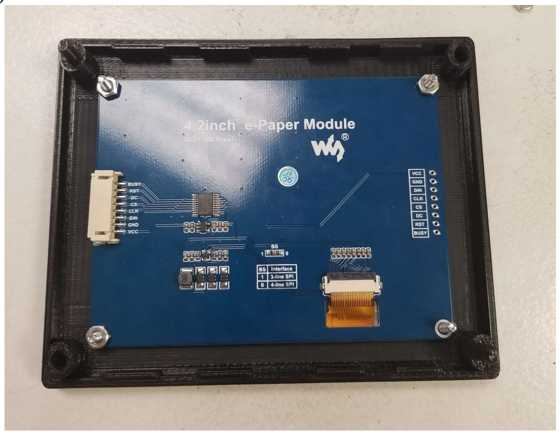


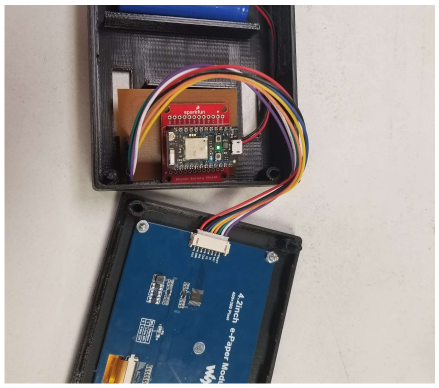
Figure 2: Final Case Model within TinkerCAD

Once the case is printed, screw the screen into the top portion. Place the battery and the Photon/PCB within the base of the case, as labeled on the drawing. Finally, connect the battery to the shield, and connect the waveshare screen wires from the PCB and the screen and close the case! You are now done assembling!

1)



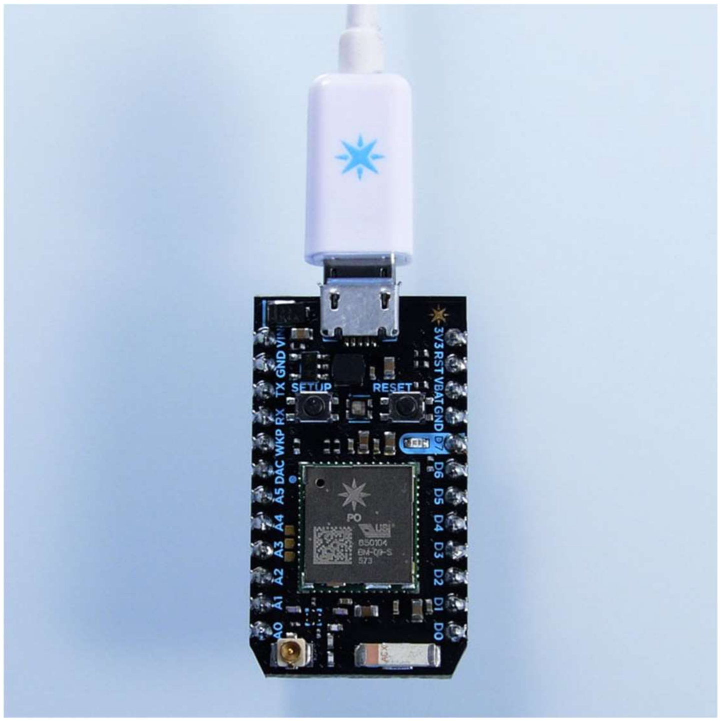
2)



**Step 4: Configuring and Flashing your Photon with the Code**

Here are the instructions for connecting to your Photon.

1. Power On Your Device



Plug the USB cable into your power source. (Your computer works perfectly for this purpose.) Your Particle device does not need your computer to connect to wifi. You could just as easily power your device with a power brick, a battery shield, or another power source wired to the VIN pin.

As soon as it is plugged in, the RGB LED on your device should begin blinking blue. If your device is not blinking blue, hold down the SETUP button.

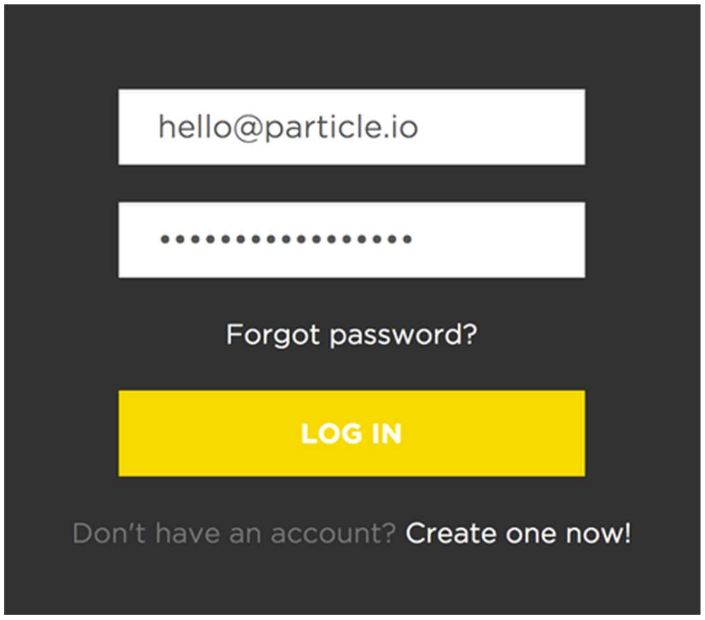
2. Connect your Photon to the Internet using the setup web application
   Go to setup.particle.io
   Click on Setup a Photon
   After clicking on NEXT, you should be presented with a file (photonsetup.html)
   Open the file
3. Connect your PC to the Photon, by connecting to the network named PHOTON-...
4. Configure your Wi-Fi credentials

Note: If you mistyped your credentials, the Photon will blink dark blue or green. You have to go through the process again (by refreshing the page or clicking on the retry process part)

5. Rename your device. You will also see a confirmation if the device was claimed or not

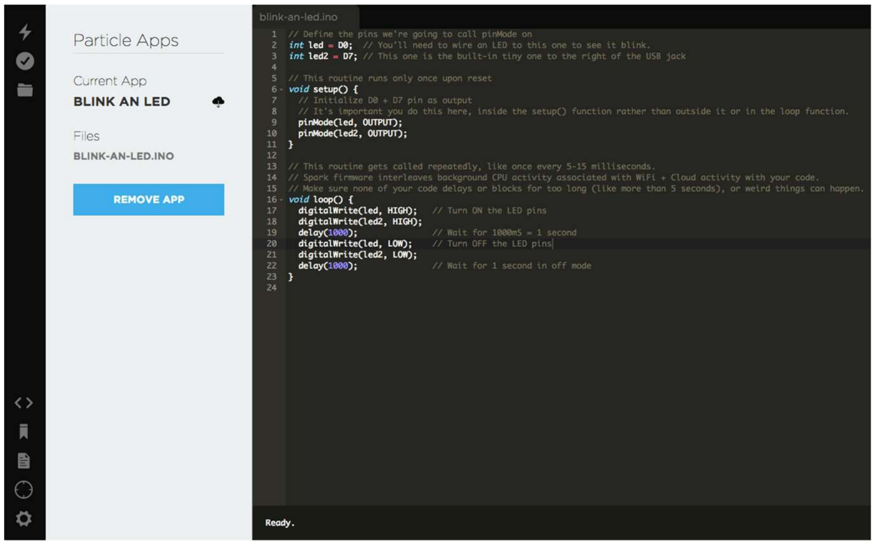
6. When you're ready to reprogram your device, head over to the Particle Web IDE. Once there, either sign in with your account or create one if you do not have one

Creating an account is a simple one-step process. When presented with the login screen, click the "create account" text and fill out the form including your email address (careful!) and desired account password. That's it! Remember the login information as that will be needed for future logins, using the app, or the console.



Particle Build is an Integrated Development Environment, or IDE; that means that you can do software development in an easy-to-use application, which just so happens to run in your web browser.

Particle Build starts with the navigation bar on the left. On the top, there are three buttons, which serve important functions:



```
// Define the pins we're going to call pinMode on
int led = D0;  // You'll need to wire an LED to this one to see it blink.
int led2 = D7; // This one is the built-in tiny one to the right of the USB jack

// This routine runs only once upon reset
void setup() {
  // Initialize D0 + D7 pin as output
  // It's important you do this here, inside the setup() function rather than outside it or in the loop function.
  pinMode(led, OUTPUT);
  pinMode(led2, OUTPUT);
}

// This routine gets called repeatedly, like once every 5-15 milliseconds.
// Spark firmware interleaves background CPU activity associated with WiFi + Cloud activity with your code.
// Make sure none of your code delays or blocks for too long (like more than 5 seconds), or weird things can happen.
void loop() {
  digitalWrite(led, HIGH);   // Turn ON the LED pins
  digitalWrite(led2, HIGH);
  delay(1000);               // Wait for 1000mS = 1 second
  digitalWrite(led, LOW);    // Turn OFF the LED pins
  digitalWrite(led2, LOW);
  delay(1000);               // Wait for 1 second in off mode
}
```

Flash:

Flashes the current code to the device. This initiates an over-the-air firmware update and loads the new software onto your device.

Verify:

This compiles your code without actually flashing it to the device; if there are any errors in your code, they will be shown in the debug console on the bottom of the screen.

Save:

Saves any changes you've made to your code.

At the bottom, there are four more buttons to navigate through the IDE:

Code:

Shows a list of your firmware applications and lets you select which one to edit/flash.

Library:

Explore libraries submitted by other users, and develop your own.

Docs:

Brings you to the documentation for Particle.

Devices:

Shows a list of your devices, so you can choose which to flash, and get more information on each device.

Settings:

Change your password, log out, or get your access token for API calls.

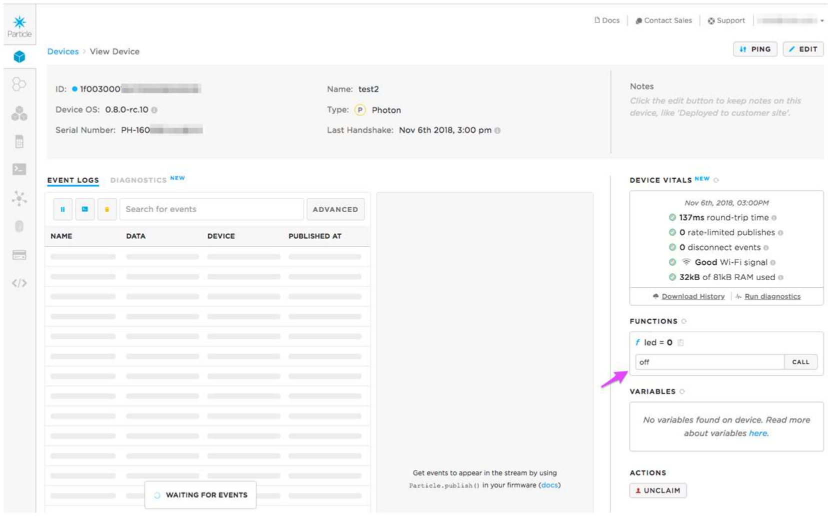
7. At this point create a new project within the IDE and copy the code into that file! The code can be found at the end of these instructions.
8. To get your final product working you must finally flash the code onto your Photon.
    a. ***Connect:*** Make sure your device is powered and "breathing" Cyan, which indicates that it's connected to the Particle Device Cloud and ready to be updated.

b. ***Select Your Device:*** If you have more than one device you have to make sure that you've selected which of your devices to flash code to. Click on the "Devices" icon at the bottom left side of the navigation pane, then when you hover over device name the star will appear on the left. Click on it to set the device you'd like to update (it won't be visible if you have only one device). Once you've selected a device, the star associated with it will turn yellow. (If you only have one device, there is no need to select it, you can continue on to the next step).
c. ***Flash:*** Click the "Flash" button, and your code will be sent wirelessly to your device. If the flash was successful, the LED on your device will begin flashing magenta.

## Step 5: Interacting with you Device

Now that your device is all build and your code is flashed you can now use your device to update your screen with a message of your choice!

To send a message you will need to use the Particle App or the Particle Web Console. With both the app or the console click on the device you are using, and go to functions.



ID: 1f003000
Device OS: 0.8.0-rc.10
Serial Number: PH-160
Name: test2
Type: Photon
Last Handshake: Nov 6th 2018, 3:00 pm

Notes
Click the edit button to keep notes on this device, like 'Deployed to customer site'.

Search for events

Get events to appear in the stream by using Particle.publish() in your firmware (docs)

Nov 6th, 2018, 03:00PM
137ms round-trip time
0 rate-limited publishes
0 disconnect events
Good Wi-Fi signal
32kB of 81kB RAM used

f led = 0
off

No variables found on device. Read more about variables here.

Within the functions you will see a function called 'UpdateMessage'. Fill in the text box with your message and press send. Your screen should now be updating with this message!

## Code

```
/*
 * Project: Office Network Board (ONB)
 * Description: An interactable message board to update a message of your choice!
 * Author: Georgia Snelling, Paul Kollat, and Jack Raney
 * Date: February 9th, 2019

 * Hookup:       Screen:          PHOTON:
 *  3V3 (red) ---> 3V3       ---> 3V3
 *  GND (blk) ---- GND       ---- GND
 *  DIN (blu) ---> MOSI(D11) ---> MOSI(A5)    MOSI = Master Out Slave In
 *  CLK (ylo) ---> SCLK(D13) ---> SCK (A3)
 *  CS  (ora) ---> SS  (D10) ---> SS  (A2)
 *  DC  (grn) ---> OUT (D9)  ---> OUT (A1)
 *  RST (wht) ---> OUT (D8)  ---> OUT (A0)
 *  BUSY(pur) <--- IN  (D7)  <--- IN  (D4)
 */


//Libraries to be included
#include <Adafruit_GFX_RK.h>
#include <GxEPD2_PP.h>

#define ENABLE_GxEPD2_GFX 0

#include <Arduino.h>
#include <Adafruit_GFX.h>
#include <FreeMonoBold9pt7b.h>

#include <GxEPD2_BW.h>
#include <GxEPD2_3C.h>


//Defining the I/O Pins
#define CS_PIN   A2
#define DC_PIN   A1
#define RST_PIN  A0
#define BUSY_PIN  A7
```

```
//4.2in Waveshare 3 Color Screen
GxEPD2_3C<GxEPD2_420c, GxEPD2_420c::HEIGHT> display(GxEPD2_420c(CS_PIN,
DC_PIN, RST_PIN, BUSY_PIN));


//Function Initialize Screen
//Set up all variables for screen and text settings
void initializeScreen() {
  // Clear the screen and initialize it with the template for the message
  display.init();
  display.setFont(&FreeMonoBold9pt7b);              //Set font
  display.setTextColor(GxEPD_BLACK);                //Set text colour as black.  Can change to
be black or yellow if desired
  display.setTextSize(1);                           //Setting text size to 1.  Change as needed
  display.setRotation(0);                           //Change to 1 for portrait view, 0 for landscape
  display.fillScreen(GxEPD_BLACK);                  //Setting background colour to be white.
Can be black or yellow if desired
}


//Setup Function
//Defines the function 'Update Message To Screen'
void setup() {
  Serial.begin(9600);
  initializeScreen();
  bool function = Particle.function("MessageUpdate", updateMessageToScreen);
  Serial.println("setup ready");
}

//Loop
void loop() {
   if (Time.hour()==20) {
       System.sleep(SLEEP_MODE_DEEP,15*3600);
   }
}

//Function processNewMessage
//Updates the screen with the nex message
//Input Parameters: String newMessage is the new message to be updated on the screen
void processNewMessage(String newMessage){
  const String text = newMessage;
  int16_t tbx, tby; uint16_t tbw, tbh;                  // boundary box window
  display.getTextBounds(text, 0, 0, &tbx, &tby, &tbw, &tbh);
  uint16_t x = (display.width() - tbw) / 2;
  uint16_t y = (display.height() + tbh) / 2;
  display.setFullWindow();
  display.firstPage();
  do
  {
    display.setCursor(x, y);                          // set the postition to start printing text
    display.print(text);                              // print some text
    // end of part executed multiple times
  }
  while (display.nextPage());
  Serial.println("done");

}


//Function connected to Device
//Input parameters: String updatedMessage is the new message to be updated on the screen
//Returns 0
int updateMessageToScreen(String updatedMessage)
{
  String newMessage = updatedMessage;

  if (newMessage == "" || newMessage == "clear")
  {
    newMessage = "Dr. Estell's Office Board"; //default message, change as you please (:
  }

  Serial.println(newMessage);
  processNewMessage(newMessage);
  return 0;
}
```