

Para Manana

1. Bring laptop if you have one
2. Download Particle App
3. Register for free Particle account (can do this through the app)
4. Download Blynk App
5. Bookmark <https://github.com/jlosaw/SDQcreates>





ENVENTYS
PARTNERS

PRODUCT DESIGN
BOOTCAMP

DAY 2: IoT
February, 2020

OUR PROGRAM

What Makes A Good IoT Product

Lab Time

History of IoT

How Easy is It

What is IoT

What Is IoT?

IOT



3 Layers

Application/Network/Perception



**Application Layer
(The Device)**



**Network Layer
(The Transport)**



**Perception Layer
(The App)**



Application

What you are doing

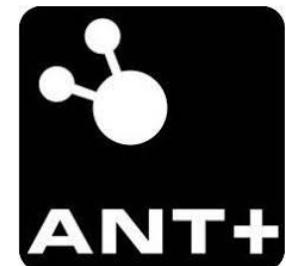
- Sensors (Temp, humidity, light, buttons, etc.)
- Actuators (motors, lights, speakers, etc.)



Network

How you transfer the data

- Ethernet
- WiFi
- Bluetooth
- Zigbee



Perception

The App or dashboard

- Web
- Mobile
- Tablet



All Together



History of IoT

Kevin Ashton

Coined the term Internet of Things in 1999 while working at Proctor and Gamble

- “The Internet of Things is about empowering computers...so they can see, hear and smell the world for themselves.”



128.2.209.43

Coke is IoT!

- In 1982, engineers at Carnegie Mellon connected a Coke machine to their network to monitor the temperature and inventory of the machine.

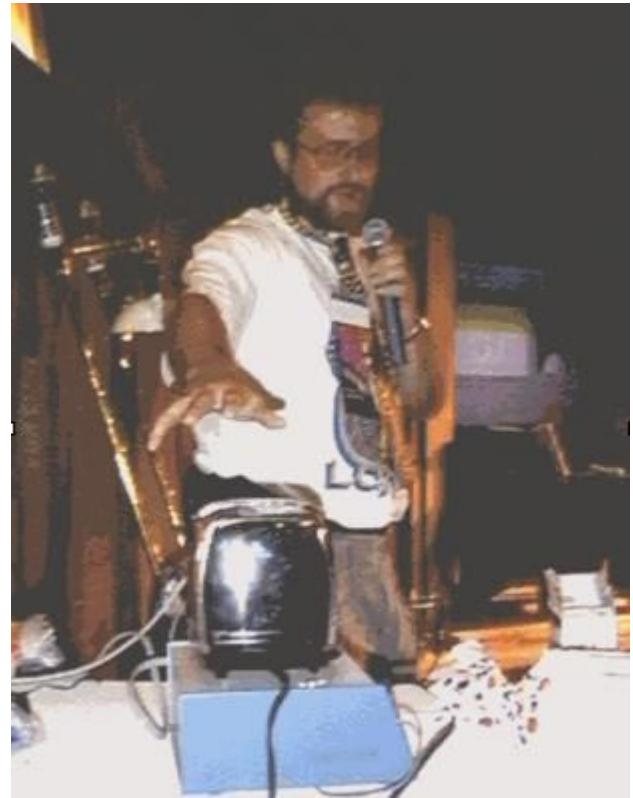


**Carnegie
Mellon
University**

Toast

1990. Simon Hackett and John Romkey connect the first commercial product to the internet (not a computer)

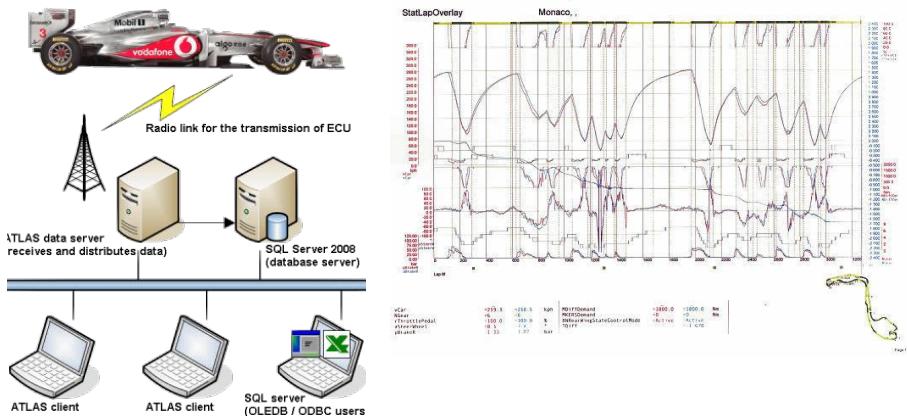
- Could be remotely turned on and off. Darkness of toast controlled by on-time
- In 1991 they added a crane to take the toast out.



Racing

Formula 1 Cars become mobile IoT labs

- First car to pitwall transmission in the late 80s
- By the 90s, bi-directional wireless data achieved



JenniCam

In 1996 Jenni Ringley uses her webcam to become the first internet celebrity using IoT.

- Broadcast her entire life in 15s increments
- World-wide attention lands her on Letterman



In the Year 2000!

LG Internet Digital DIOS IoT Refrigerator Debuts.

Took 55 people and a budget of ~\$45 million

- Measured temperature and freshness of food
- Integral webcam



Ambient Orb

Tracked stock prices and displayed color based on price

- MIT project that was launched in 2002.
- Still in production



Fitness

IoT gets buff

- 2006. Nike+ connects to iPod to measure running stats
- 2009 Fitbit announces their first fitness tracking device



Smartphones

Our necks have not been the same since

- 2007. First gen iPhone released
- A pocket full of possibilities



Smart Home

The call is coming from inside the house!

- 2011. Nest gen 1 launches
- Slew of smart home devices follow
- Amazon Echo released in late 2014



How Easy Is It?

Super Easy

You can be on the web in minutes

- Developer kits make it very easy to push data to the cloud
- Many boards are compatible with Arduino code and libraries
- Sample code is plentiful



Fire on Demand

No developer required

- Services like Blynk make it very easy to build a custom app for IoT hardware
- Integral solution that requires very little code for the device



And It's Cheap

Less than a venti mocha latte

- Wifi+ BTLE in a \$2.50 module
- Developer hardware for under \$20
- Cellular data plans for \$3/month



What Makes A Good IoT Product?

What Makes a Good IoT Device

Smart, Adaptable, Human centered, and Solve a Real Problem

- Strong Need
- Positive Cost/Benefit
- Robust architecture
- Easy to use



Strong Need

Don't do it just because you can

- Think of the Why, before the "I"



Benefit has to outweigh the cost

IoT versions of the same device can cost 5-10X their analog components

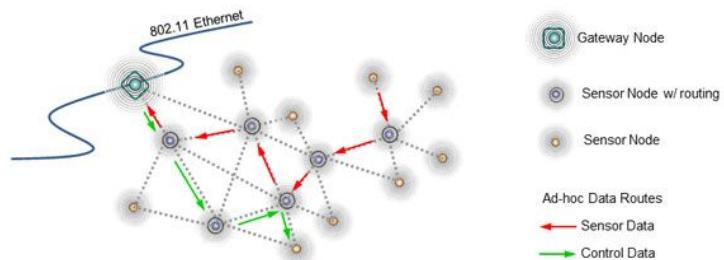
- Not to mention the time required to set them up

	\$215.99	\$59.95
	\$99.97	\$12.60
	\$179.94	\$39.99
	\$48.50	\$12.99
	\$49.99	\$0.00
	\$594.39	\$125.53
	RD\$31,772	RD\$6,686

Robust Architecture

Reliability is key to adoption

- Communication needs to work all the time
- Data has to be accurate
- Actuators need to work
- Adaptive networks



Easy to Use

Everyone wins when the UI matches the need

- Clear and intuitive UI
- Quality electromechanical assembly
- Easy installation



Easy-IoT Platforms

Goldilocks quality

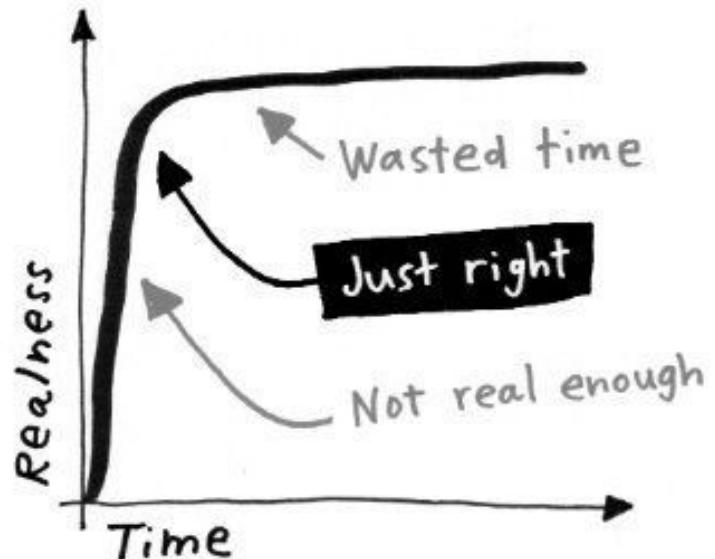


Image from *Sprint* by Jake Knapp

Goldilocks IoT

We can use pre-built tools like Tinker and Blynk to do minimal code with the maximum impact.

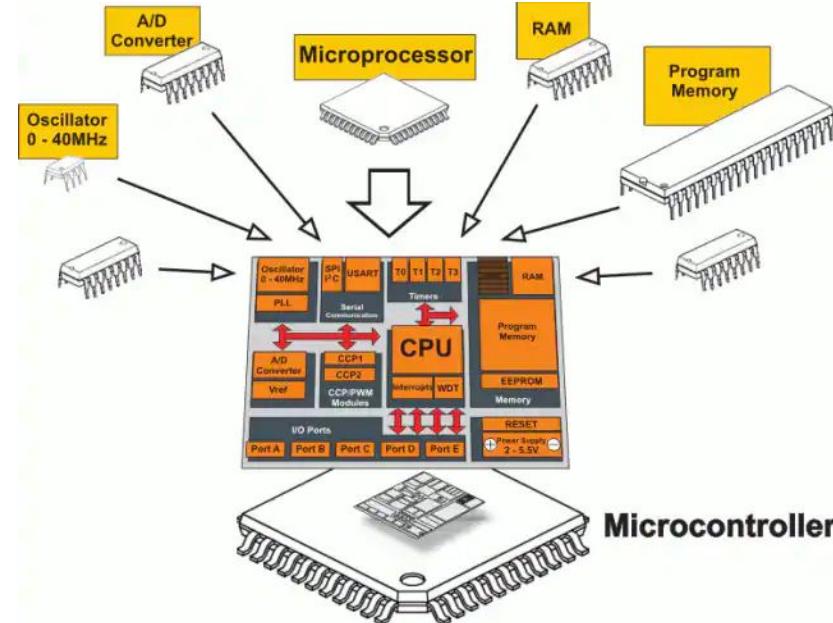


Microcontroller Basics

What is a microcontroller?

Mini computer

Processor, memory, and I/O in one integrated circuit



Development Board

Microcontroller that is expanded out onto a PCB that can easily be used for prototyping. Often too large for incorporation into a consumer product, but useful for prototyping.



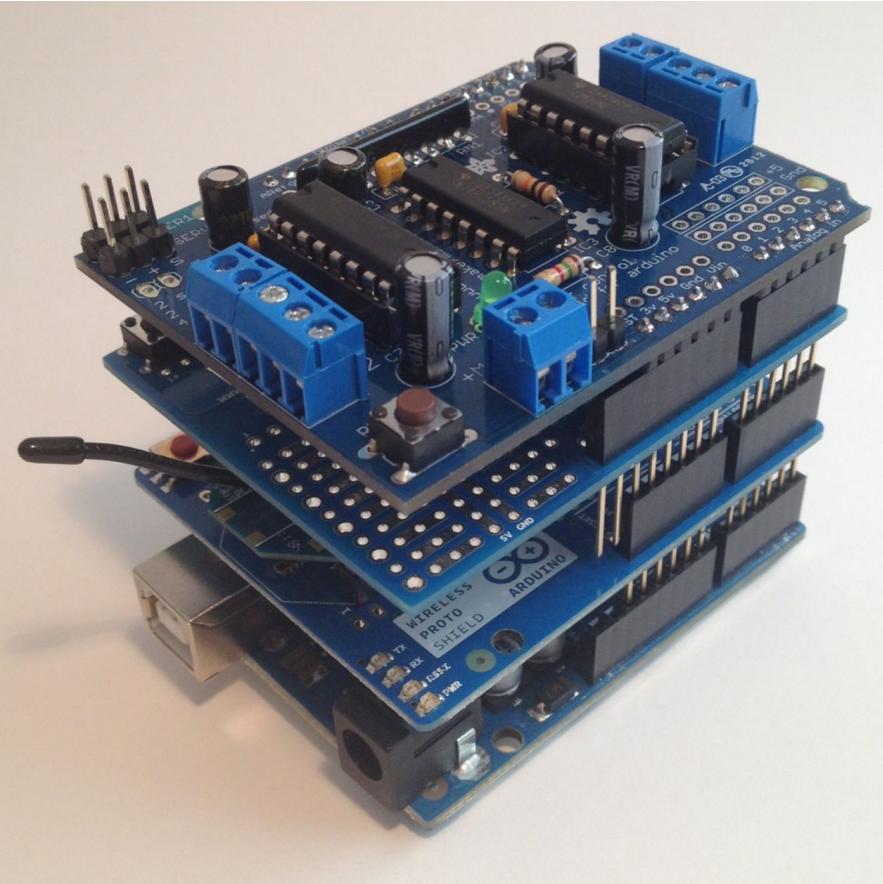
GreenIQ

Exception that proves the rule



Arduino

Arduino was first major platform that brought microcontrollers to the masses. Allowed the use of stackable “shields” to simplify wiring.

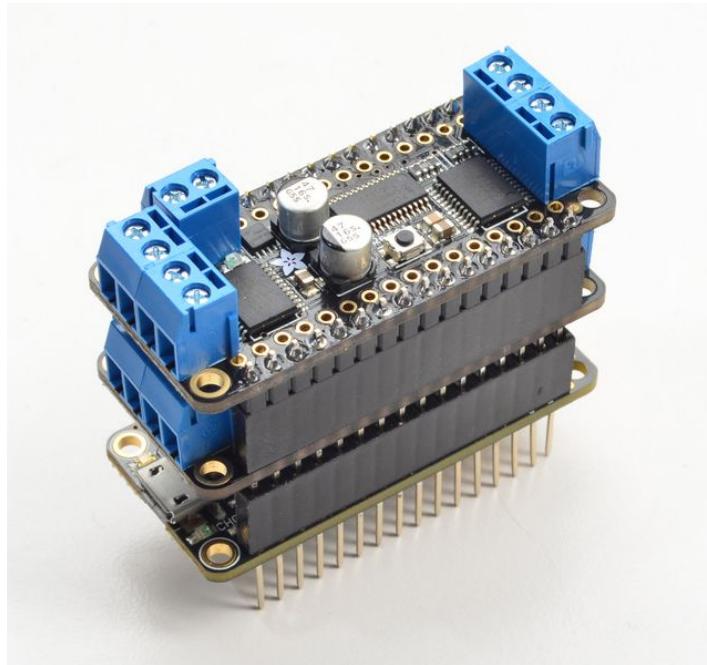


Feather

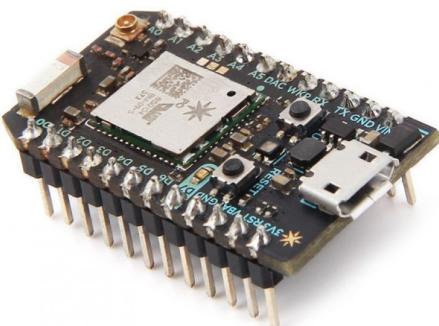
Feather platform is becoming the new standard for dev board footprints.

Most IoT dev boards are feather compatible.

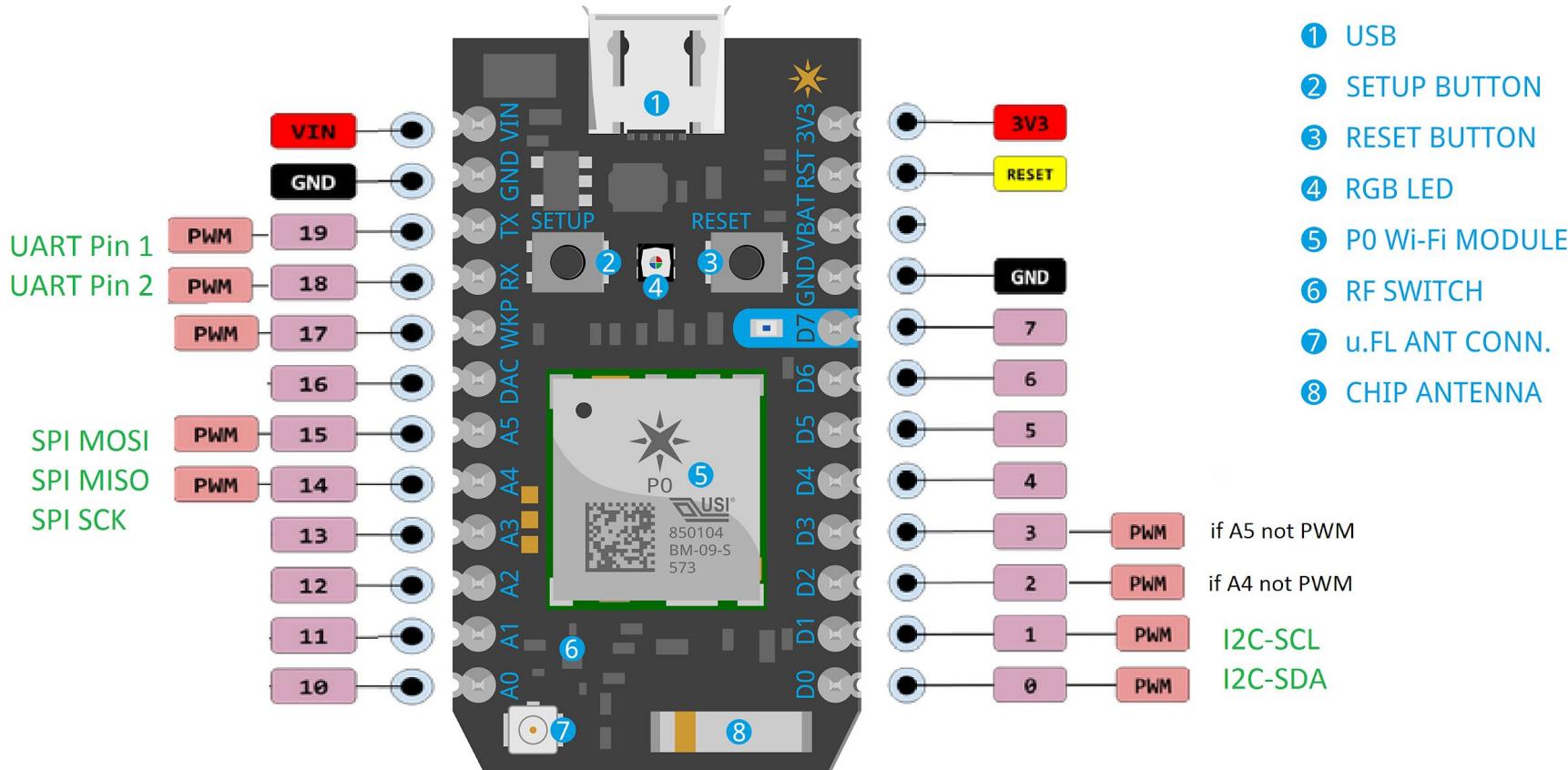
The Photon is not, but their new sister boards are.



Particle Photon



- Particle PØ Wi-Fi module
 - Broadcom
 - BCM43362 Wi-Fi chip
 - 802.11b/g/n Wi-Fi
 - STM32F205RGY6 120Mhz ARM Cortex M3
 - 1MB flash, 128KB RAM
- On-board RGB status LED (ext. drive provided)
- 18 Mixed-signal GPIO and advanced peripherals
- Open source design
- Real-time operating system (FreeRTOS)
- Soft AP setup
- FCC, CE and IC certified



One More Thing

Arduino

- Go to <https://www.arduino.cc/en/main/software>
- Download latest version of Arduino IDE
- Will use this only for the serial monitor

zAnd download Github Repo

<https://github.com/jlosaw/SDQcreates>



Download the Arduino IDE



ARDUINO 1.8.12

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.

This software can be used with any Arduino board. Refer to the [Getting Started](#) page for installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install
Windows app Requires Win 8.1 or 10
[Get](#)

Mac OS X 10.8 Mountain Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

Photon Connection and Programming

Particle App

Claim device. Download and follow Setup

- Setup (run once)
- WiFi
 - ID: EVENTO XOLUCTRONIC
 - PW:EVENTO2020
- Put device in “listening mode” and re-connect
- Photon will hold up to 5 sets of Wifi credentials, so you should not have to reconnect next class



Firmware Update

Photons need to be upgraded to latest firmware

- Open WebIDE from Particle.io
- Click code icon and select “Tinker”

The screenshot shows the Particle WebIDE interface. On the left, there's a vertical toolbar with various icons: a lightning bolt, a checkmark, a folder, a double arrow (highlighted with an orange circle), a bookmark, a question mark, a file, a clock, a bar chart, and a gear. Below the toolbar is a blue button labeled "CREATE NEW APP". To the right of the toolbar is a list of example apps, each with a small icon and a number. The "Tinker" app is highlighted with a large orange oval. The code editor on the right contains a snippet of "tinker.ino" code.

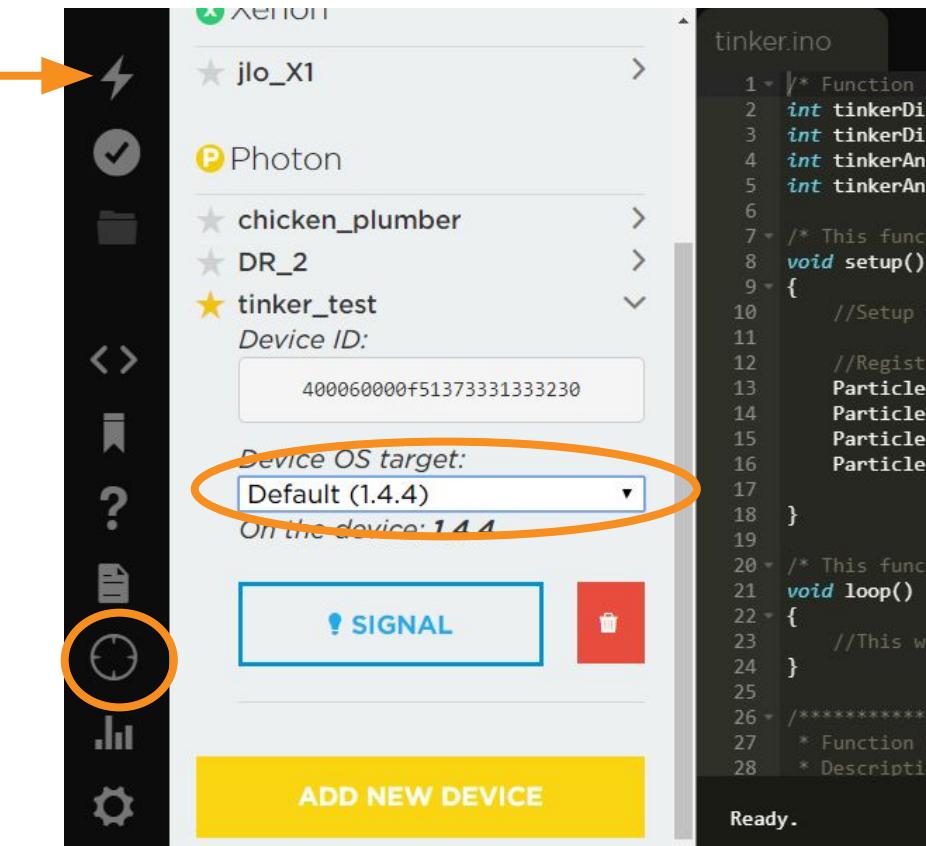
```
tinker.ino
1  /* Function
2  int tinkerDi
3  int tinkerDi
4  int tinkerAn
5  int tinkerAn
6
7  /* This func
8  void setup()
9  {
10    //Setup
11
12    //Registe
13    Particle
14    Particle
15    Particle
16    Particle
17
18  }
19
20  /* This func
21  void loop()
22  {
23    //This w
24  }
25
26  ****
27  * Functi
28  * Descripti
```

Ready.

Firmware Update

Step 2

- Select “Devices from the sidebar menu
- Set “Device OS target” to latest stable release (1.4.4)
- Click the lightning bolt icon to update firmware and flash the code



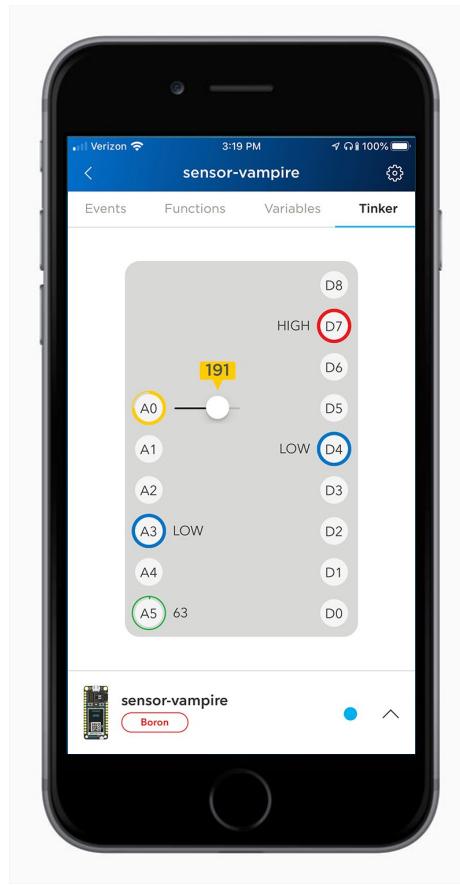
Tinker

Tinker Functions

Can use Particle app to control any of the Photon pins with a single touch.

4 Functions:

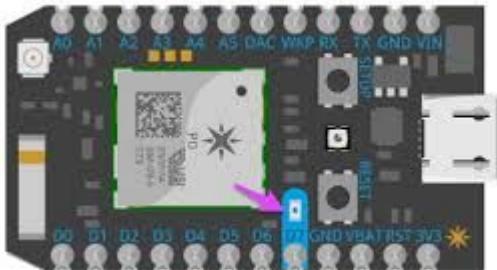
`digitalWrite`
`analogWrite`
`digitalRead`
`analogRead`



Try it Out

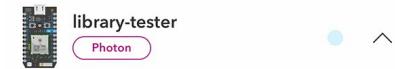
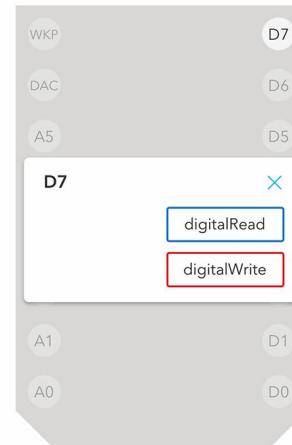
Blink the on-board LED

- 1: Choose your device.
- 2: Click the “Tinker” tab
3. Click D7 and set to “digital Write”
4. Toggle the pin to see on-board LED turn on and off



A screenshot of the library-tester app interface on an iPhone. The top status bar shows "Verizon" with signal strength, "11:52 AM", and "71%". The main screen has a blue header with the title "library-tester" and a gear icon. Below the header are tabs for "Events", "Functions", "Variables", and "Tinker", with "Tinker" being the active tab. The background is a light gray grid representing the pins on a board. A callout box is open over pin D7, showing two options: "digitalRead" (blue) and "digitalWrite" (red).

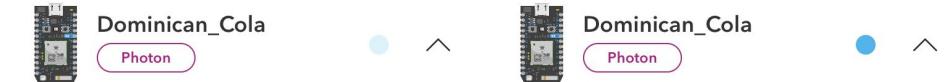
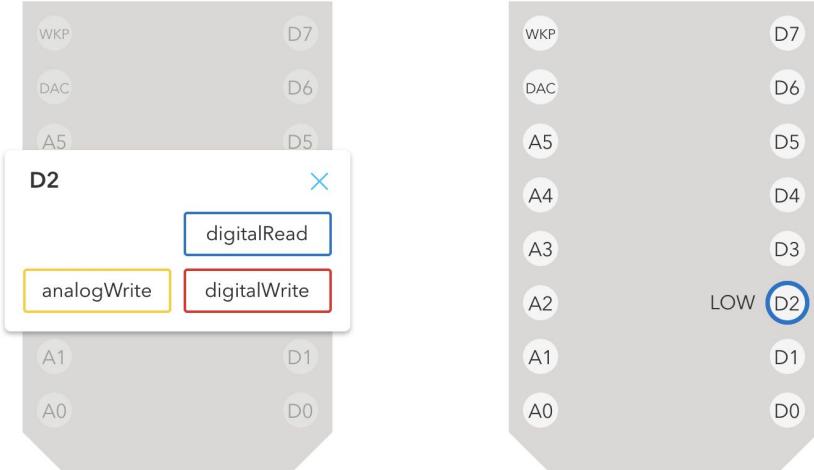
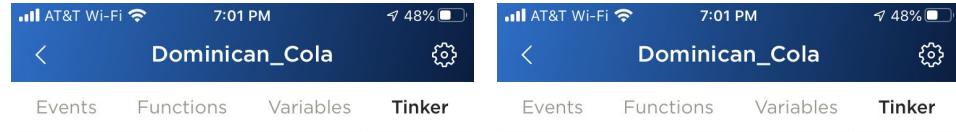
A screenshot of the library-tester app interface on an iPhone, similar to the previous one but with a red circle highlighting the "digitalWrite" button in the callout box over pin D7. The status bar shows "12:03 PM" and "69%". The "Tinker" tab is still active. The background grid shows the pins from D0 to D7.



Prototype Button

Read High/Low state of a pin

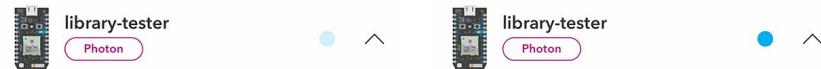
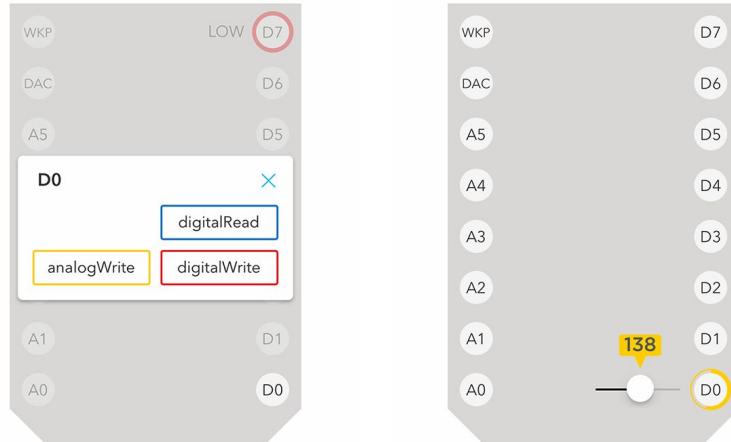
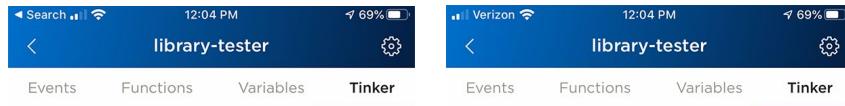
- 1: Set D2 to “digitalRead”
- 2: Note that the state is “LOW”
- 3: Run jumper wire from “3V3” pin to D2
- 4: Touch D2 and see how it reads “HIGH”



LED Dimmer

Analog Control

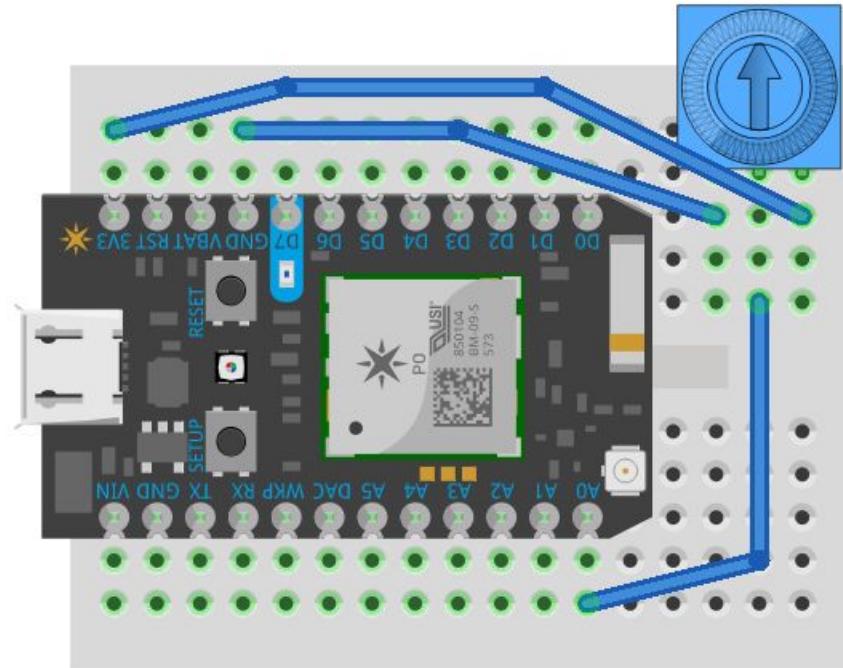
- 1: Connect LED from D0 to Ground
- 2: Change D0 to “analogWrite” in Tinker
3. Change slider to see LED change brightness



Read Potentiometer

Read a sensor

- 1: Connect potentiometer to A0
- 2: Change A0 to “analogRead” in Tinker
3. Turn the dial to change the values



fritzing

Programming The Photon

Modified C/C++

3 Main Parts

- Declarations
- Setup (run once)
- Loop (run forever)
- Subroutines

```
1 void setup() {  
2 }  
3  
4 void loop() {  
5 }  
6  
7 }
```

Ready.

GitHub

Website with sample code

- <https://github.com/jlosaw/SDQcreates>
- Sample files for all of our experiments

The screenshot shows a GitHub repository page. At the top, there's a header with the repository name "jlosaw / SDQcreates", a "Code" tab (which is selected), and various metrics: 4 commits, 1 branch, 0 packages, 0 releases, and 1 contributor. Below the header, there's a list of recent commits:

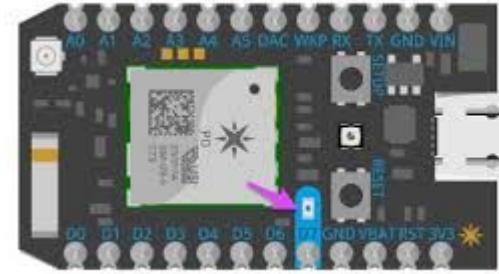
Author	Commit Message	Date
jlosaw	Create sdq_particle_variable.ino	Latest commit 504b0e4 3 days ago
	sdq_particle_variable.ino	3 days ago
	sdqblink.ino	Update sdqblink.ino
	sdqdht	Create sdqdht

At the bottom of the page, there's a note: "Help people interested in this repository understand your project by adding a README." followed by a green "Add a README" button.

Blink an LED

The “Hello World” of microcontrollers

- Often used for troubleshooting and debugging
- Photon has on-board LED connected to pin D7
- Download **sdqblink.ino** and run code



Particle Apps

Current Example
BLINK AN LED
Blink an LED

USE THIS EXAMPLE

Type to find

My apps

AlexoLEDTest
Blink an LED
blynk
dht-test
dht-test
DisplaytheAnalog Value
ifttt_servo
iftttLEDtest
Jlotest
NeoandDHT
rgbw-strandtest
servotests
singasong
Web-Connected LED
webneo

```
47 // This is our "shortend" that we'll use throughout the program:
48 // First, we're going to make some variables.
49 // This is our "orthand" that we'll use throughout the program:
50 int led1 = D8; // Instead of writing D8 over and over again, we'll write led1
51 // You'll need to wire an LED to this one to see it blink.
52 int led2 = D7; // Instead of writing D7 over and over again, we'll write led2
53 // 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.
54 // Having declared these variables, let's move on to the setup function.
55 // The setup function is a standard part of any microcontroller program.
56 // It runs only once when the device boots up or is reset.
57 void setup() {
58     // We are going to tell our device that D8 and D7 (which we named led1 and led2 respectively) are going to be output
59     // pins. That means that we will be sending voltage to them, rather than monitoring voltage that comes from them.
60     // It's important you do this here, inside the setup() function rather than outside it or in the loop function.
61     pinMode(led1, OUTPUT);
62     pinMode(led2, OUTPUT);
63 }
64 // Next we have the loop function, the other essential part of a microcontroller program.
65 // This routine gets repeated over and over, as quickly as possible and as many times as possible, after the setup function is called.
66 // Note: Code that blocks for too long (like more than 5 seconds), can cause weird things happen (like dropping the network connection). The
67 // loop function must be quick!
68 void loop() {
69     // Here we turn the LED on. Notice we'll turn it on...
70     digitalWrite(led1, HIGH);
71     // digitalWrite(led2, HIGH);
72     // Wait! Leave it on for 1 second...
73     delay(1000);
74 }
```

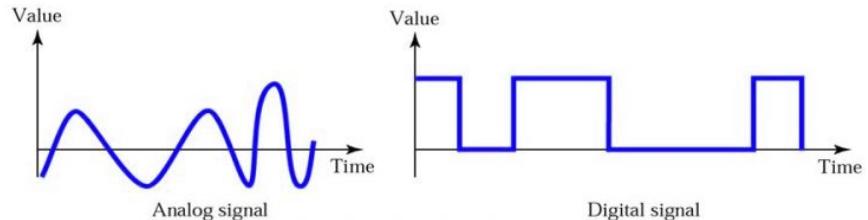
Ready.

Sensors

Types of Sensors

What kinds of signals

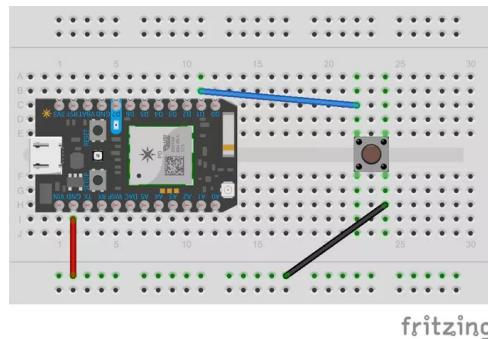
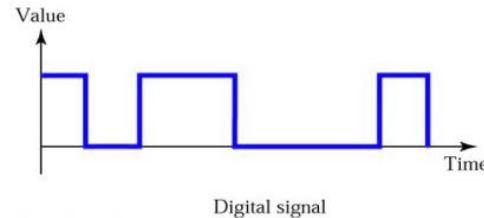
- Analog
- Digital
- Resistive
- Current
- Amplified



Digital

High or Low

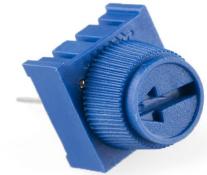
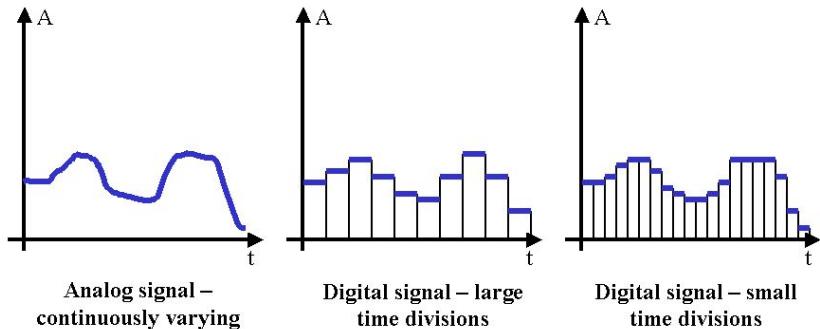
- Voltage high or low, or discrete
- Great for measuring on/off states
- Example: Limit switch on a 3D printer



Digital

Discrete values

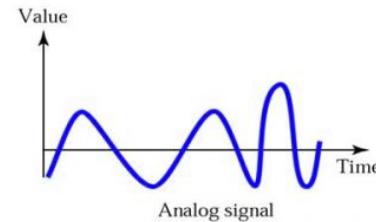
- Accuracy depends on the bit rate of the processor
- Example: 0-100 degree temp sensor with a 10 bit D/A converter reads in increments of roughly .1 degrees (100 degrees/1024 values)



Analog

Signals that are continuously variable

- Voltage output continuously variable with time
- Example: Potentiometer and a gas pedal



Analog to Digital Conversion

How many values can we read?

- Digital is base 2. 0 or 1 are the only values
- For example, 8 bit is $2^8 = 256$
- The Photon is 12 bit $2^{12} = 4096$



8 bit



16 bit

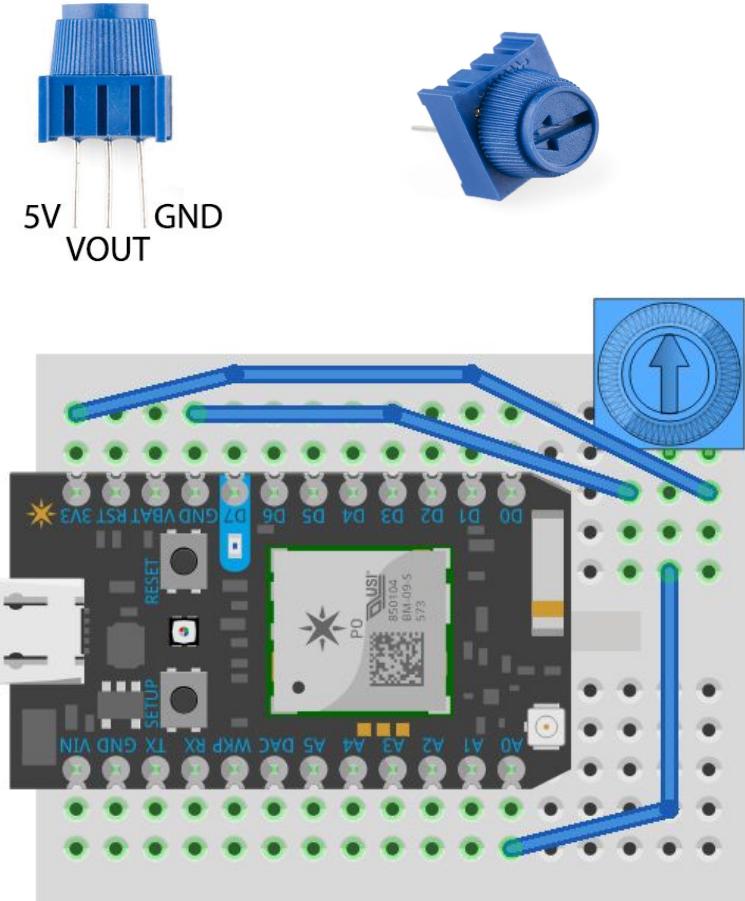


256 bit

Potentiometer

Get the analog value from the pot

- Download Arduino program
- Download [**sdq_potentiometer.ino**](#)
- Hook up the pot to A0 on the breakout board
- Remember, $V=IR$. We are varying R so the voltage will change
- The voltage output is infinitely variable, but why is output not 0- infinity

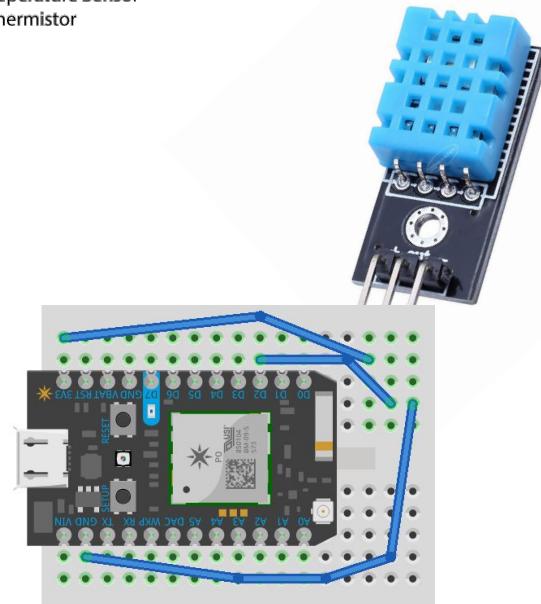
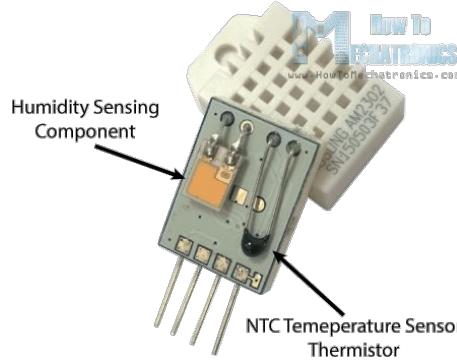


fritzing

DHT11

A more common sensor for temp and also get humidity

- Inside is an NTC thermistor and an electrode that changes resistance with humidity
- Download **sdqdht.ino**
- Use `#include "Adafruit_DHT_Particle.h"` library
- Data is aggregated and sent over 1 data wire
- Libraries for the sensor parse the data
- Setup the serial monitor from Arduino IDE to watch the data stream

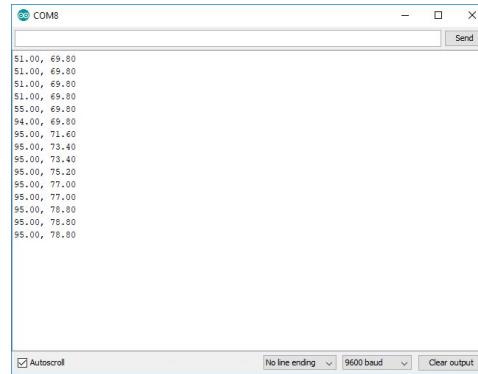


fritzing

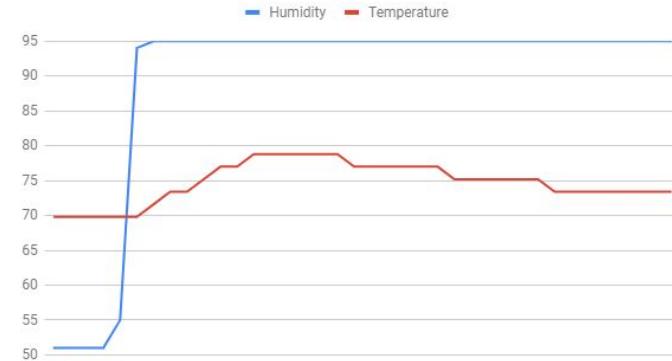
Pro Tip

Format serial output for easier plotting

```
Serial.print(h);
Serial.print(", ");
Serial.println(f);
```



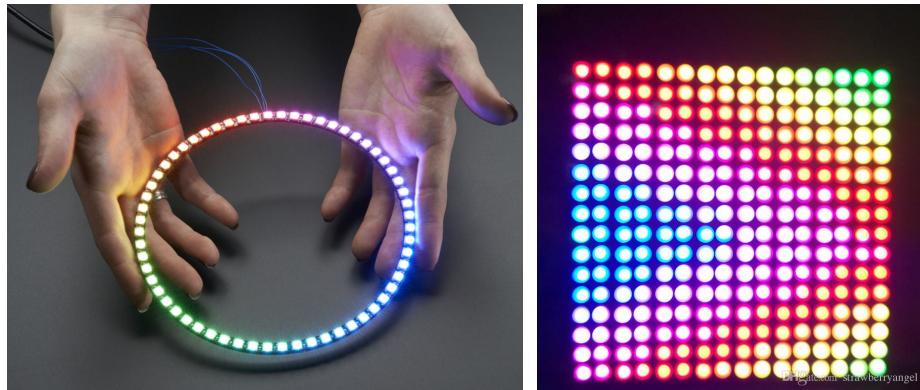
Humidity and Temperature



Addressable LEDs

Control each pixel in a chain or grid

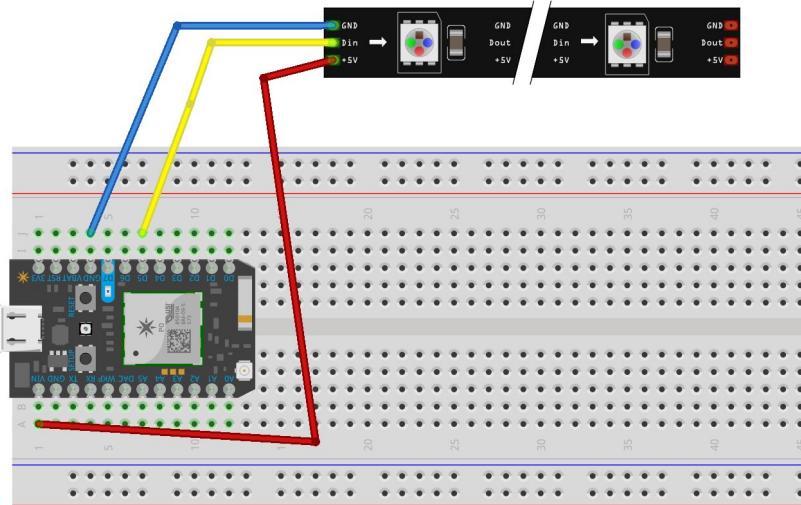
- Each LED has a small control chip that takes commands from the controller, displays its own value and then passes the next instruction down the line
- Allows for more complicated effects and customized output
- Comes at a price
- Two main types
 - Neopixels (3 wire)
 - Dotstars (4 wire)



Double Rainbow

Test out some samples from the library

- Use hookup wire to connect ring to data line to D5 on the board.
- Need 5V so hookup power to Vin
- Download and run **sdq_addressableLED.ino**
- Don't forget to re-add the Neopixel library
- Push the sample code to the board and note the effects



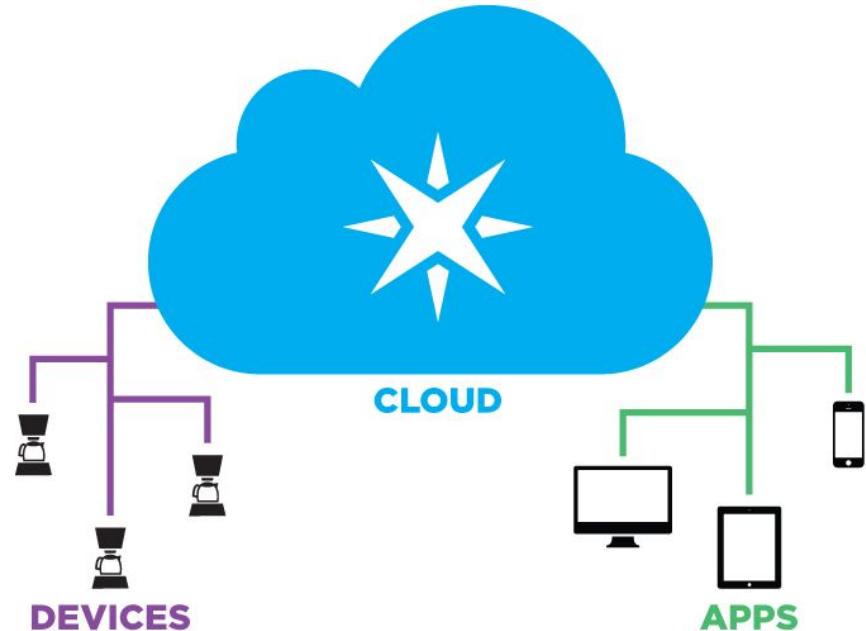
Particle Platform

Cloud Functions

Particle Cloud

4 Ways to interact with the Photon over the web

- Particle Functions
- Particle Variables
- Particle Publish
- Particle Subscribe
- Reference docs are here:
<https://docs.particle.io/reference/device-os/firmware/photon/>
- There are more, but this is what we are going to focus on



Particle Console

Your dashboard to see what is going on with your device

- **Devices**
- Products
- Networks (for mesh setups)
- Sim Cards (for LTE chips)
- **Events**
- Integrations
- Authentication
- Billing
- **Web IDE**

The screenshot shows the Particle Console interface. On the left is a vertical sidebar with icons for Devices, Products, Networks, Sim Cards, Events, Integrations, Authentication, Billing, and Web IDE. The 'Events' icon is highlighted with an orange circle. The main area displays the 'Events' page, which includes a search bar, an advanced filter button, and a table of event data. The table has columns for NAME, DATA, DEVICE, and PUBLISHED AT. A specific row for 'Temp_F' is highlighted with a blue selection bar and labeled '69.800003'. To the right of the table, there's a panel titled 'Temp_F' showing its details. The top right corner of the interface includes links for Docs, Contact Sales, and Help.

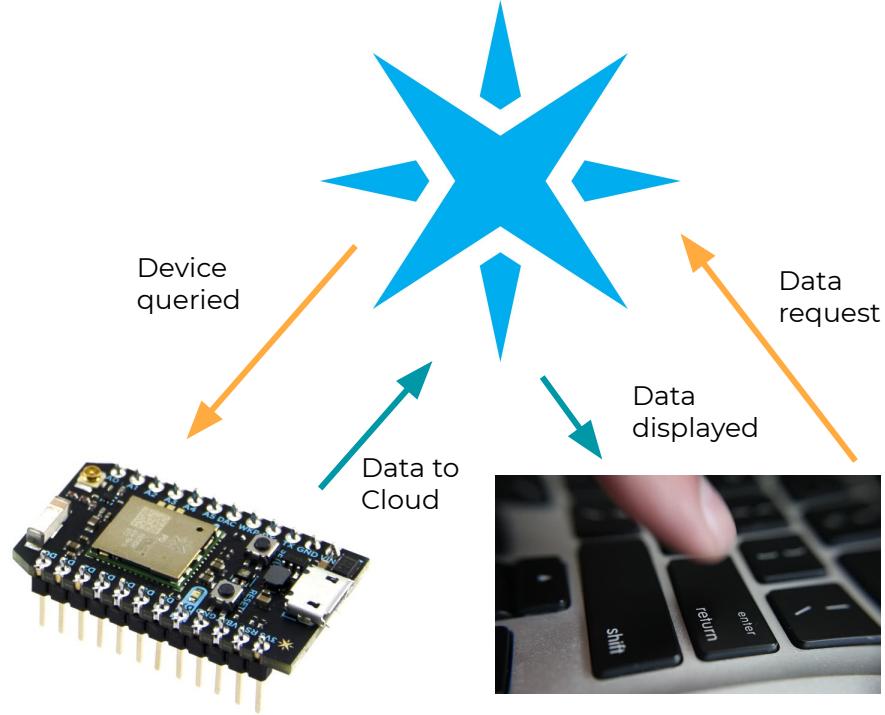
NAME	DATA	DEVICE	PUBLISHED AT
JLOGH_Temp	60	chicken_plumber	12/16/18 at 145:09 pm
Humidity	60.000000	chicken_plumber	12/16/18 at 145:09 pm
Temp_F	69.800003	chicken_plumber	12/16/18 at 145:09 pm
JLOGH_Temp	60	chicken_plumber	12/16/18 at 145:04 pm
Humidity	60.000000	chicken_plumber	12/16/18 at 145:04 pm
Temp_F	69.800003	chicken_plumber	12/16/18 at 145:03 pm
JLOGH_Temp	60	chicken_plumber	12/16/18 at 144:58 pm
Humidity	61.000000	chicken_plumber	12/16/18 at 144:58 pm
Temp_F	69.800003	chicken_plumber	12/16/18 at 144:58 pm
Humidity	60.000000	chicken_plumber	12/16/18 at 144:53 pm
JLOGH_Temp	60	chicken_plumber	12/16/18 at 144:53 pm
Temp_F	69.800003	chicken_plumber	12/16/18 at 144:53 pm

Particle.variable()

Particle Variable

Query data from a device through the cloud

- Opens a pipeline between the cloud and a device
- Does not constantly upload data to a dashboard
- Only allowed 20 variables and no more than 12 characters
- There are three supported data types:
 - INT
 - DOUBLE
 - STRING (maximum string length is 622 bytes)



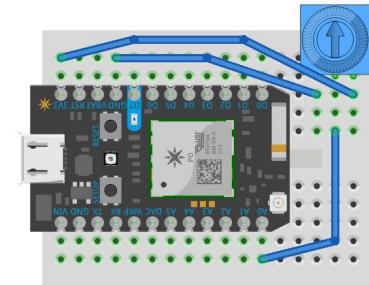
Particle Variable

Query data from console to get the value of the rotary pot

- Open **sdq_particle_variable.ino** from the repo
- Plug rotary pot into A0
- Upload code
- Login to your Particle Console
- Click the “Get” button on your variable to query the sensor and retrieve analog value

The screenshot shows the Particle Console interface for a device named "Photon". The top bar displays the URL <https://console.particle.io/devices/380046000c4736333053437>. The device details section shows "Device OS: 0.7.0", "Type: (P) Photon", "Serial Number: PH-171106-Q2TB-O", and "Last Handshake: Dec 16th 2018, 1:58 pm". The "EVENT LOGS" tab is selected, showing a table with columns NAME, DATA, DEVICE, and PUBLISHED AT. A note says "Get events to appear in the stream by using Particle.publish() in your firmware (docs)". The "VITALS" tab shows a note to "Update Device OS version to 0.8.0 or higher for device vital". The "FUNCTIONS" section indicates "No functions found on device. Read more about functions here.". The "VARIABLES" section lists "analogvalue (int32) = 714" with a "GET" button. The "ACTIONS" section has a "EMPTY AIM" button.

This screenshot shows the "VARIABLES" page in the Particle Console. It displays the variable "analogvalue (int32) = 714" with a "GET" button. An orange arrow points from the "Variables" section in the main screenshot down to this specific variable entry.

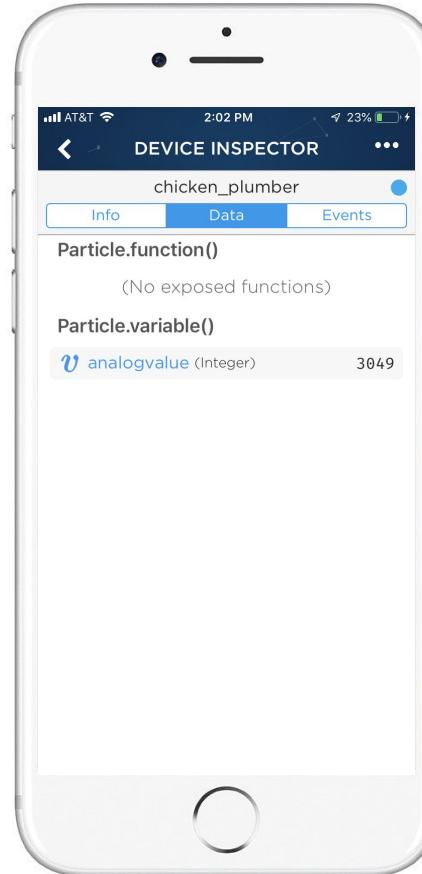


fritzing

Particle Variables with App

Make the query with your phone

- Open Particle app
- Navigate to your Photon
- Tap the “v” under the variable to retrieve the data

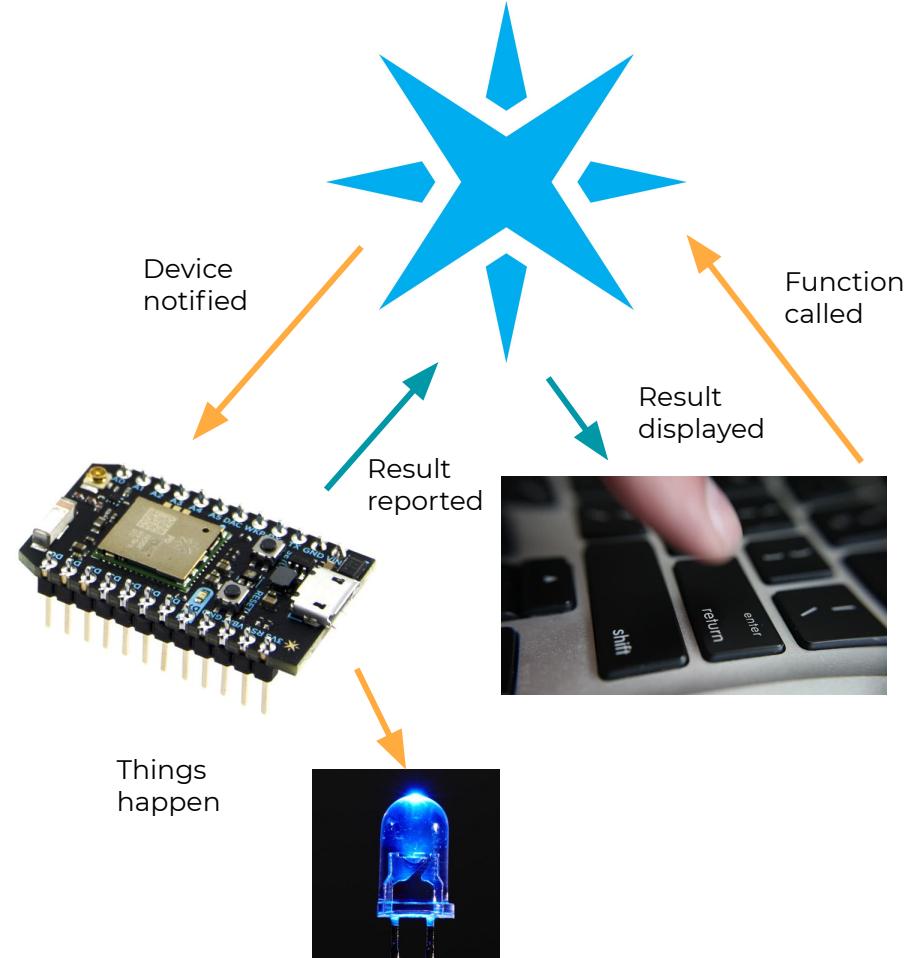


Particle.function()

Particle Function

Triggers a Particle device to run a routine

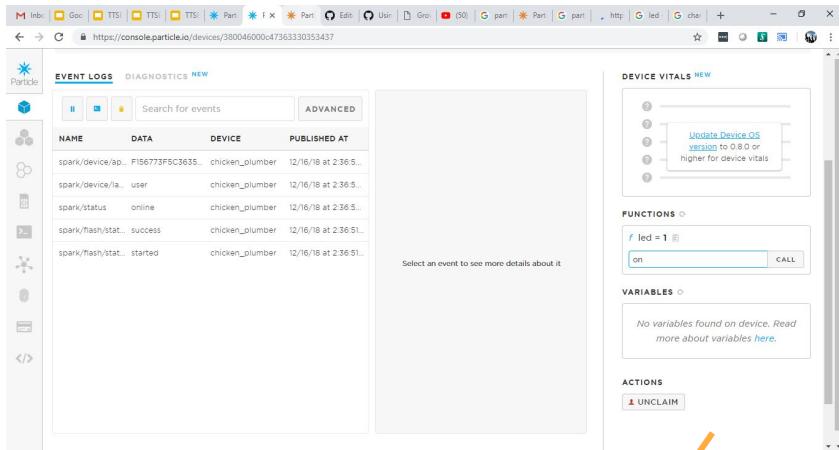
- Good for triggering an actuator
- Only allowed 15 variables and no more than 12 characters per name
- A cloud function is set up to take one argument of the String data type. This argument length is limited to a max of 63 characters.
- Returns an integer value based on the result
 - A non recognized input returns -1



Particle Function

Blink LED on command

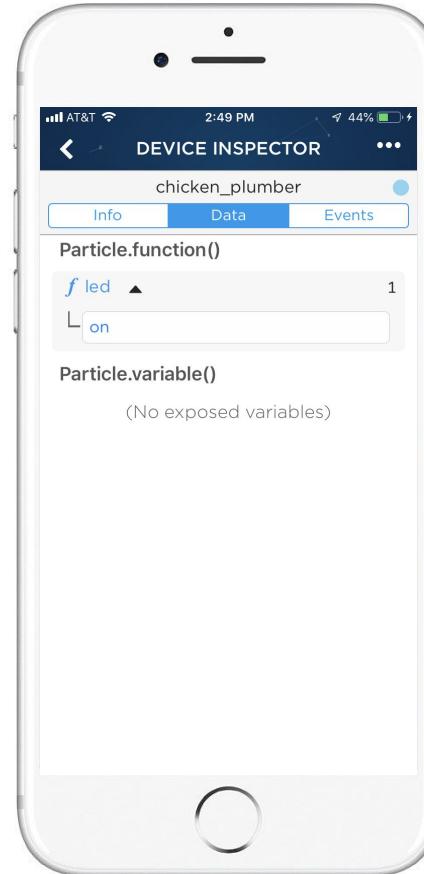
- Open **sdq_LED_function.ino** from the repo
- Upload code
- Login to your Particle Console
- Type “on” or “off” into the function box to control LED



Particle Functions with App

Make the command with your phone

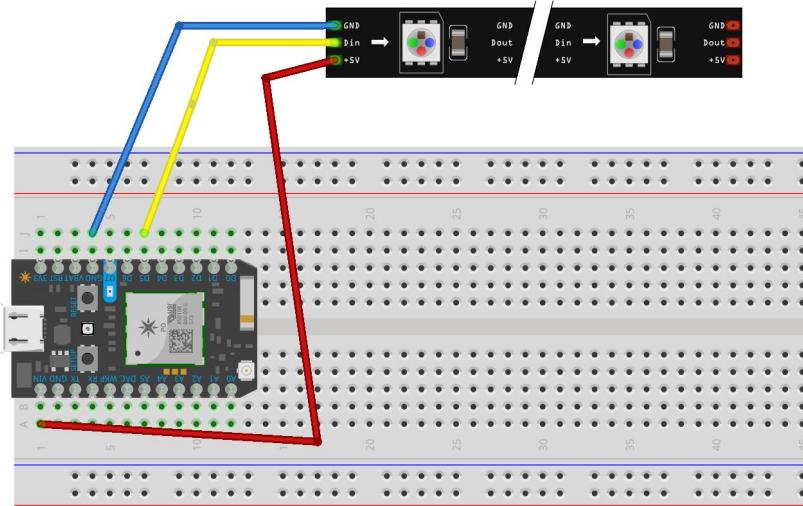
- Open Particle app
- Navigate to your Photon
- Type in the command to drive the output



Make it Rain

Let's hook up the neopixel ring and have some more fun

- Open **sdq_function_neo.ino** from the repo
- Hook up neopixel to D5
- Use "rain", "color", and off commands
- Class challenge: add a 4th function to drive a different effect

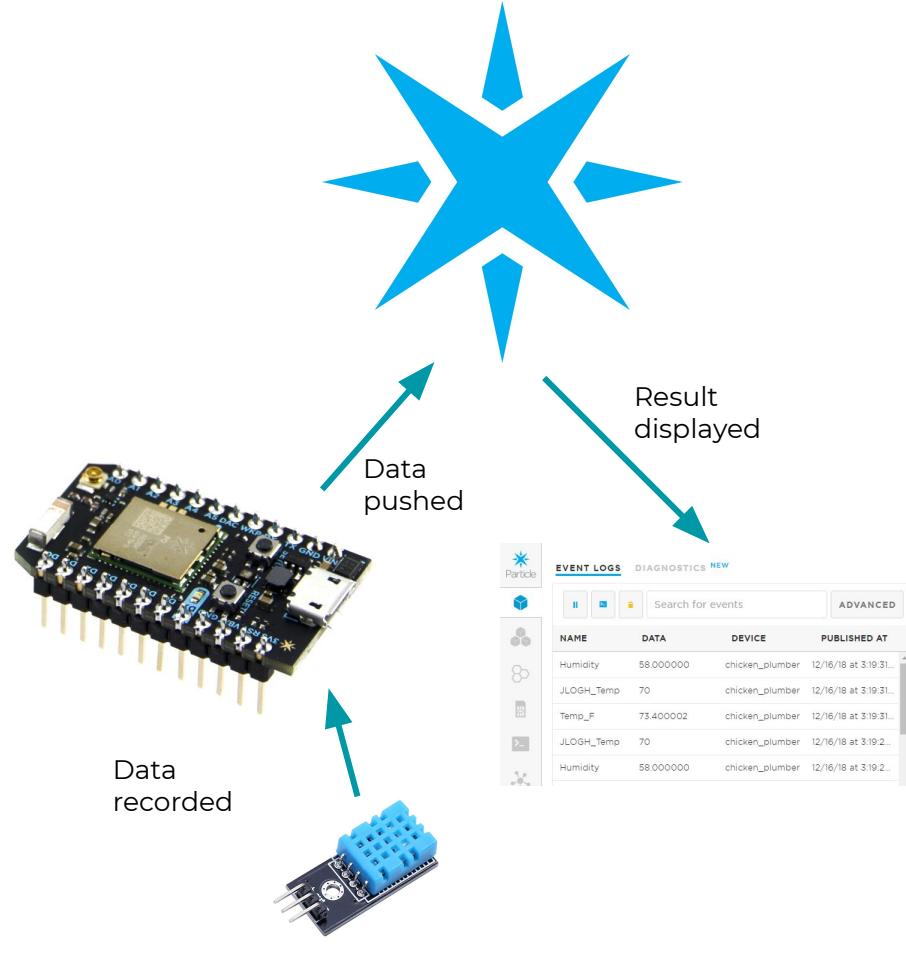


Particle.publish()

Particle Publish

Can push data continuously to the console

- Uses just a single line of code
- Event is forwarded to all registered listeners
- Pushes the value out of the device at a time controlled by the device firmware
- Publish events can be public or private
- Publish events can also be pulled if requested
- Always returns a string, so numeric data needs to be formatted on Photon firmware before pushing

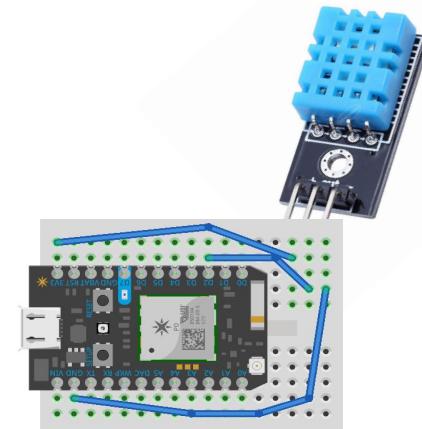


Particle Publish

Watch the data stream in

The screenshot shows the Particle Console interface. On the left is a sidebar with various icons. The main area has tabs for 'EVENT LOGS' (selected), 'DIAGNOSTICS' (NEW), and 'ADVANCED'. Under 'EVENT LOGS', there's a search bar and a table with columns: NAME, DATA, DEVICE, and PUBLISHED AT. The table lists several events, mostly from 'chicken_plumber' device, with data like 'Humidity: 58.000000' and 'Temp_F: 73.400002'. A message 'Select an event to see more details about it' is displayed below the table. To the right, under 'DEVICE VITALS' (NEW), there's a note to 'Update Device OS version to 0.8.0 or higher for device vitals'. Below that are sections for 'FUNCTIONS' (with a code snippet for 'led = 1'), 'VARIABLES' (empty), and 'ACTIONS' (with a 'UNCLAIM' button).

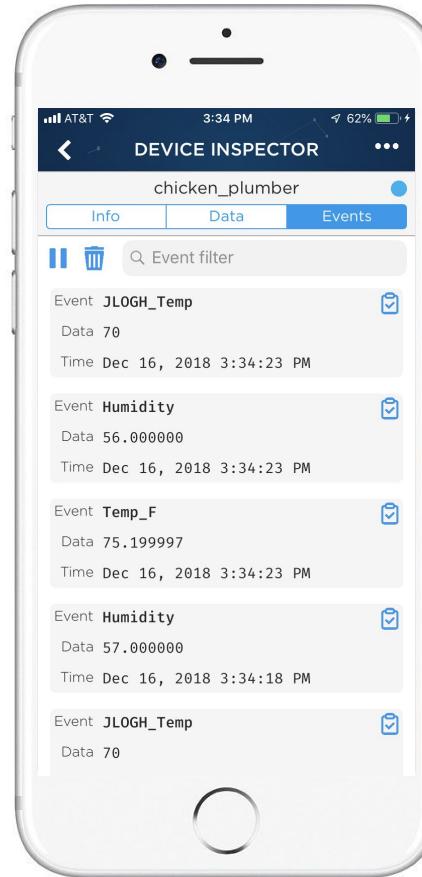
- Open **sdq_temp_publish.ino** from the repo
- Plug DHT11 into D2
- Add the following code after the serial print
 - //Publish the temp data to the console
 - Particle.publish("Temp C",String (t),60,PRIVATE);
 - Particle.publish("Humidity",String (h),60,PRIVATE)
- Upload code
- Login to your Particle Console
- Watch data stream in



Particle Publish with App

Watch your data live from anywhere

- Open Particle app
- Navigate to your Photon
- Select the “Events” tab and watch the data

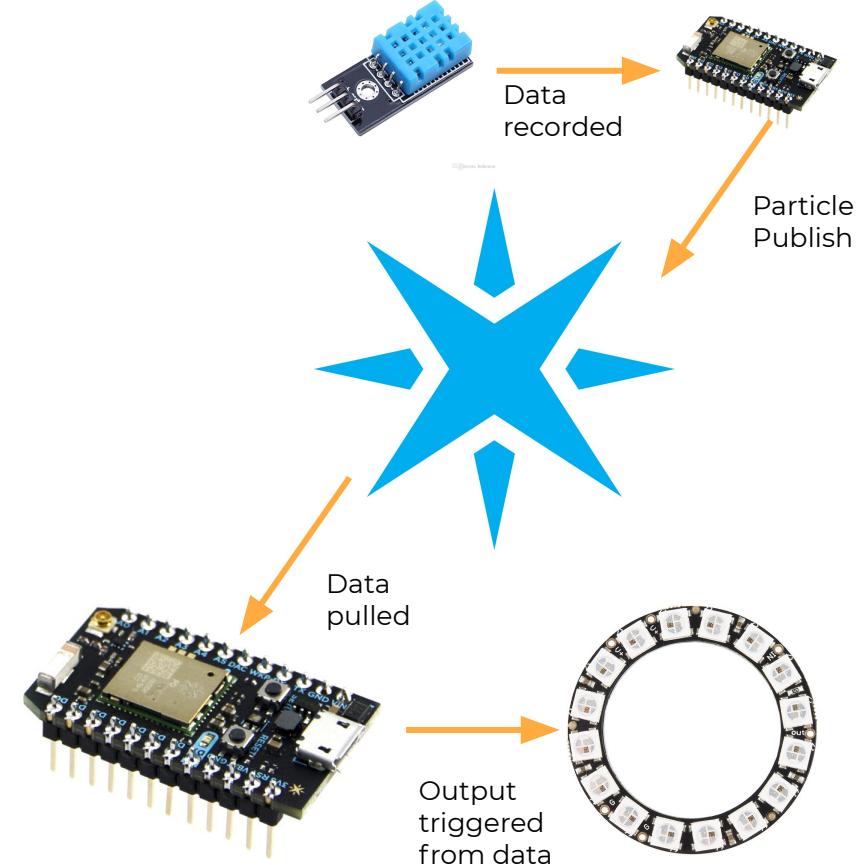


Particle.subscribe()

Particle Subscribe

Listen in for events published to the cloud

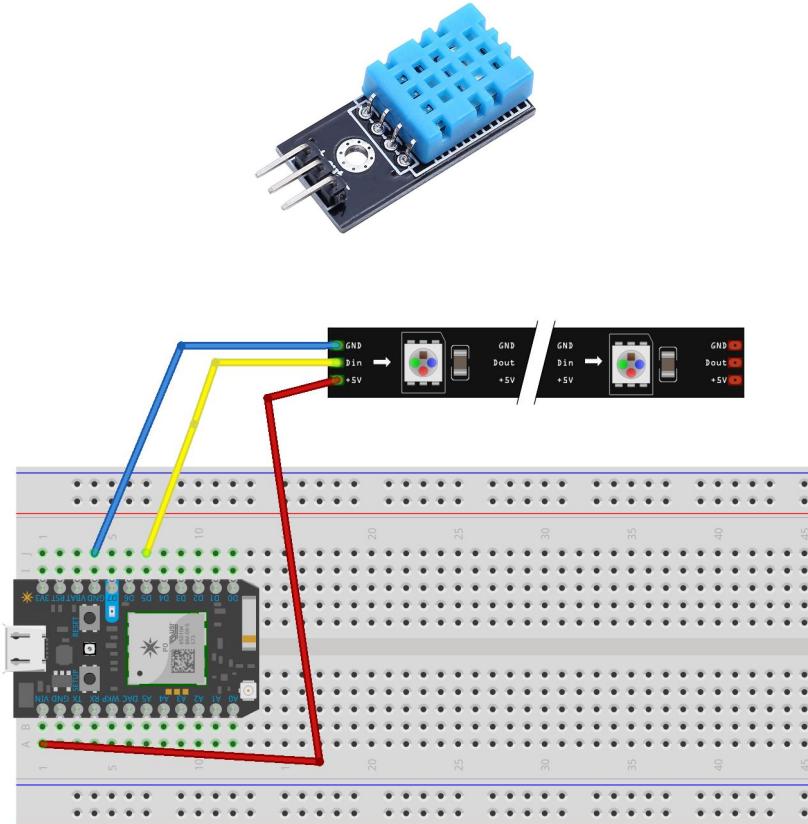
- Turns the Photon into a listening device that looks for events published to the particle cloud
- You can call `Particle.subscribe()` a maximum of 4 times; after that it will return false.
- Events are forwarded to all registered listeners so you can have many listening devices
- Need 2 devices to pull this off



Particle Subscribe

Listen in to another device

- Open **sdq_temp_subscribe.ino** from the repo
- Plug neopixel ring into D5
- Upload code
- Watch it light up when Jeremy's device reports temperature

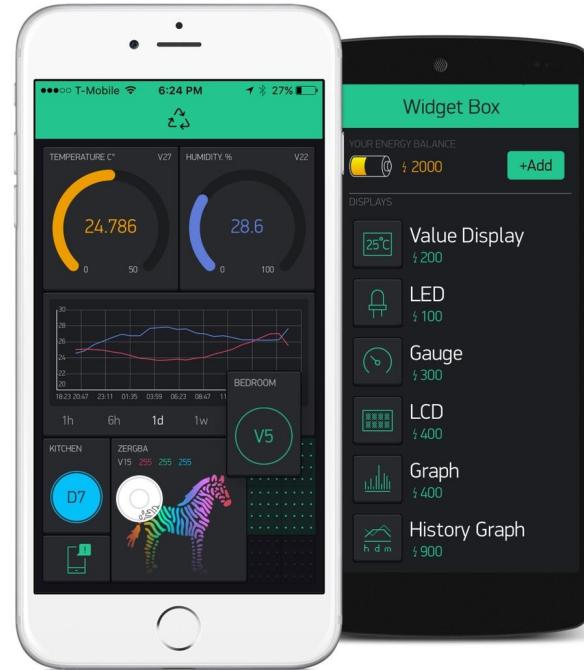


Easy-IoT Platforms

Blynk

Blynk

But what if I want more control? Blynk can unlock the power of the Photon.



Blynk Codebase

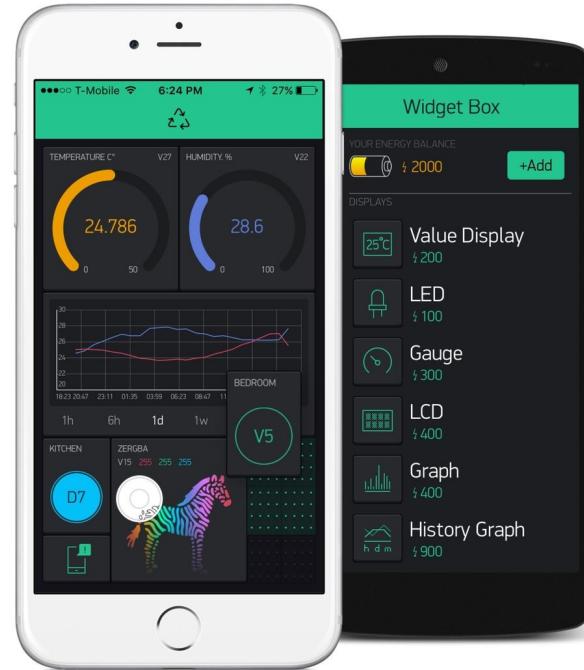
Authorization Token, Wifi Credentials, BOOM TOWN!

```
bareminimumblynkcode.ino
1 // This #include statement was automatically added by the Particle IDE.
2 #include <blynk.h>
3
4 // This #include statement was automatically added by the Particle IDE.
5 #define BLYNK_PRINT Serial
6
7 char auth[] = "XXXXXXXXXXXX";
8
9 void setup()
10 {
11     Serial.begin(9600);
12
13     delay(5000); // Allow board to settle
14
15     Blynk.begin(auth);
16
17 }
18
19 void loop() {
20
21     Blynk.run();
22
23 }
24
25 }
```

Blynk Setup

Get logged in

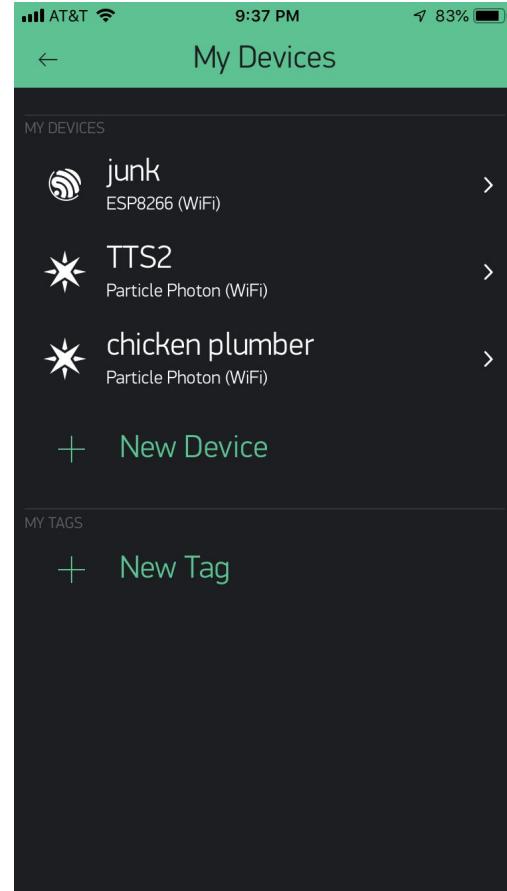
- Download Blynk app from app store
- Create an account
- Note that you get an “energy” balance to add widgets to your app



Add a Device

Connect the Photon to your app

- Click Settings>Devices> New Device
- Scroll down to “Photon”
- Have Blynk email your authentication token (it’s way too long to type in directly)



Load the Code

Connect your Photon to Blynk

- Upload **sdq_blynk.ino**
- This is all the code you need to start experimenting

```
#include <blynk.h>

#define BLYNK_PRINT Serial //#define
BLYNK_DEBUG

char auth[] =
"2fd02368df2749cbabafb6053834d03f";

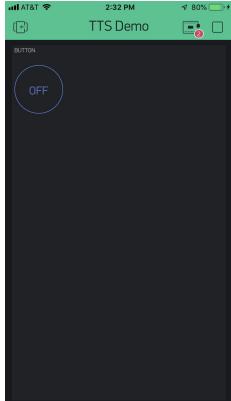
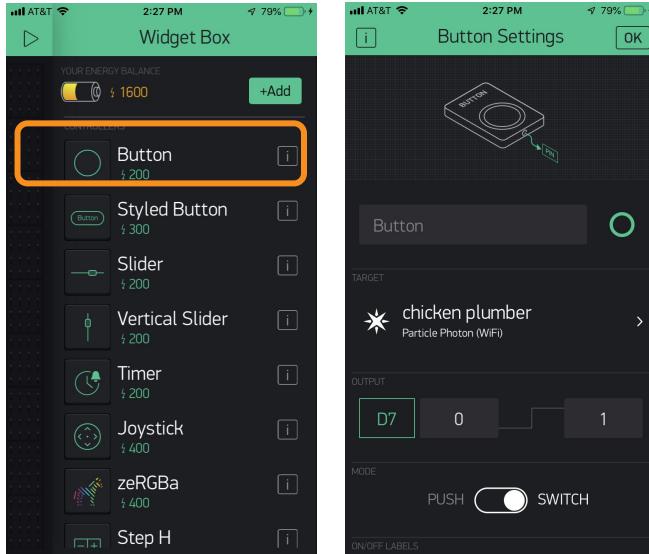
void setup()
{
    Blynk.begin(auth);
    delay (1000);
}

void loop()
{
    Blynk.run();
}
```

Add a Button

Blink the on-board LED to confirm it works

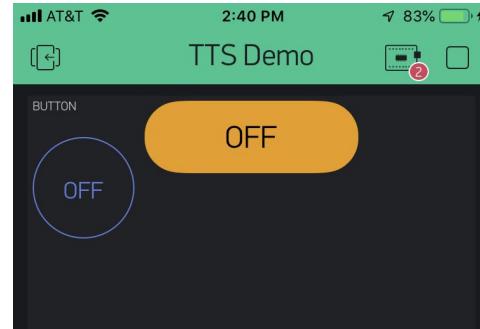
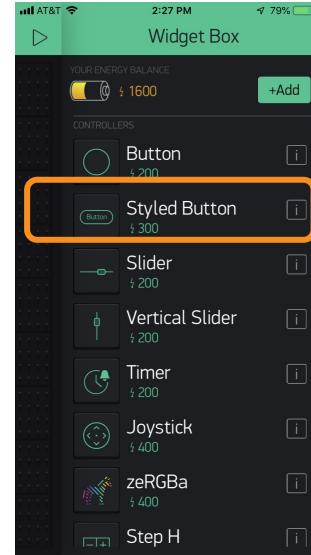
- Click the “nut” icon and select the “button” icon
- Set the output pin to D7
- Click OK to add it to your app
- Click the play button on the upper right to deploy the app
- Push the button and watch the LED light up



Stylized Button

Expanded options but costs more energy

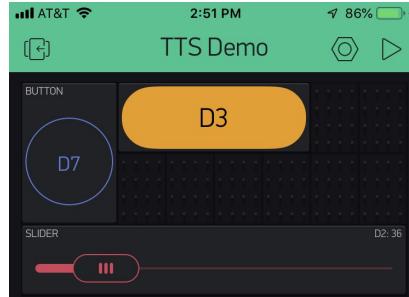
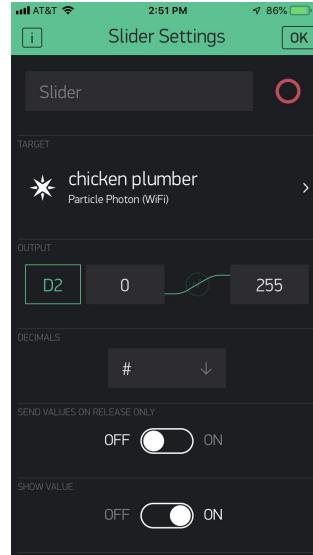
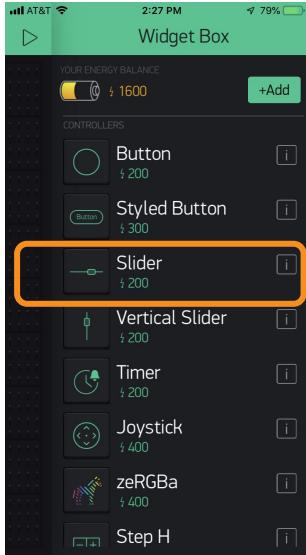
- Have control over the shape, and on/off color, and font size
- Same function as the standard button
- Note that you do not have the ability to choose D7 again until



Slider

Send a variable output to an actuator

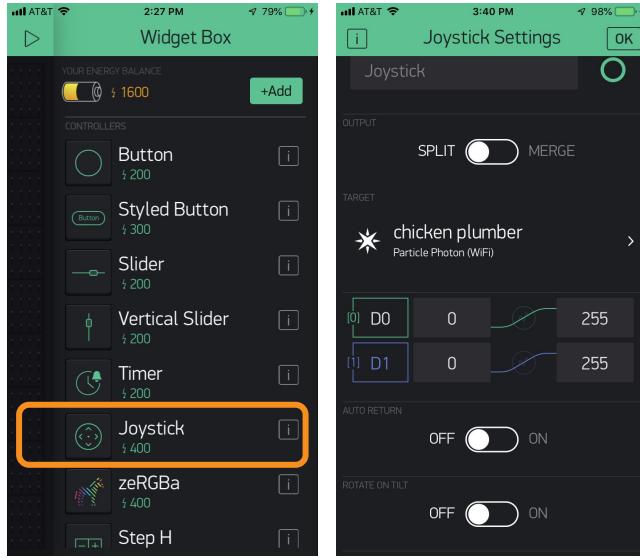
- Add slider widget to your canvas
- Change the output settings to 0-255
- Change “Send Values on Release Only” to OFF
- Set output to digital pin 3
- Add LED to pin 3 on your breadboard
- Watch the LED change brightness



Joystick

Bi-directional control

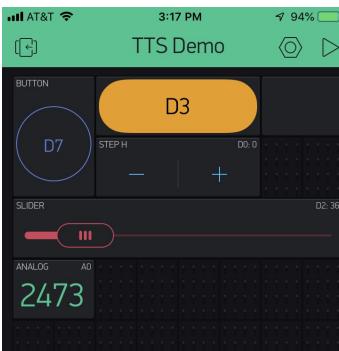
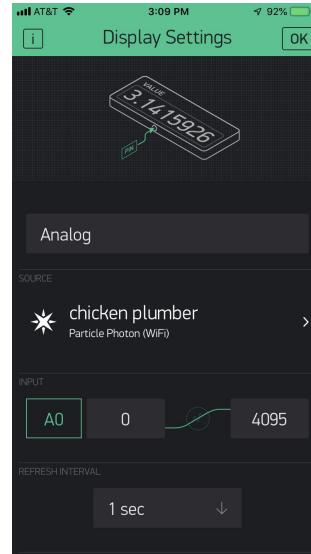
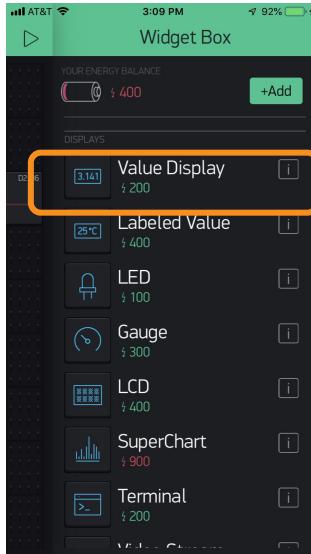
- Add Joystick widget to your canvas
- Change the output settings to D0 and D1 and both from 0-255
- Set auto-return to off
- Install LEDs on pins D0 and D1 on the Photon
- Run the app and move the joy stick widget around to see how the channels mix



Value Display

Read sensor data

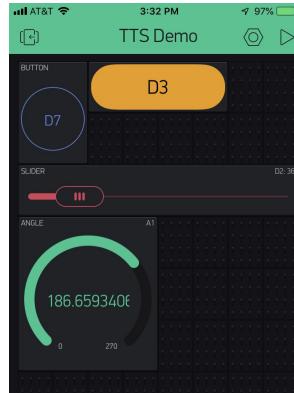
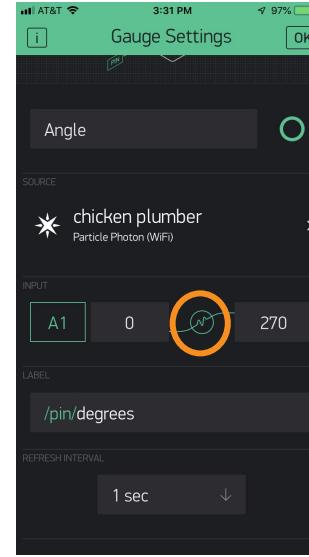
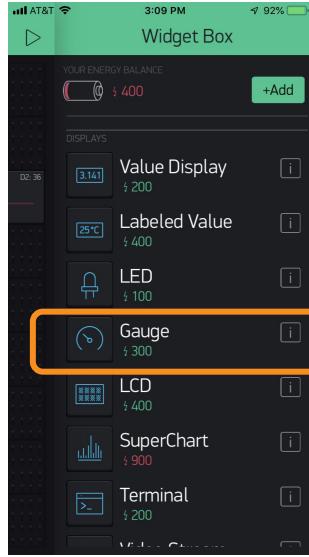
- Add a Value Display widget to your canvas
- Choose pin A0 and set range from 0-4095
- Hook up rotary pot to pin A0 on the Photon
- Move the dial and watch the values change



Gauge

Graphical sensor readout

- Add a Gauge Display widget to your canvas
- Choose pin A4 and set range from 0-270
- Click the wavy circle between the min and max values (this will scale the output and give us a calibrated output)
- Hook up rotary pot to pin A4 on the Photon breakout board
- Move the dial and watch the values change

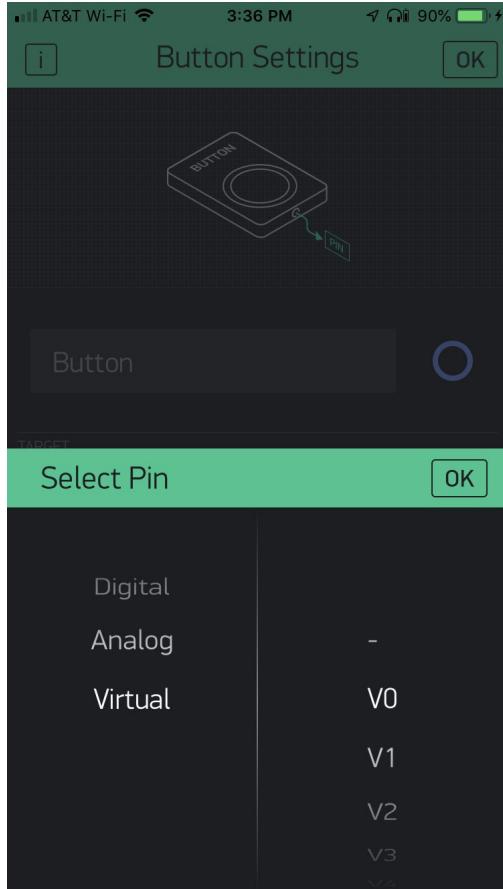


Blynk Virtual Pins

Virtual Pins

Add some jazz to your control

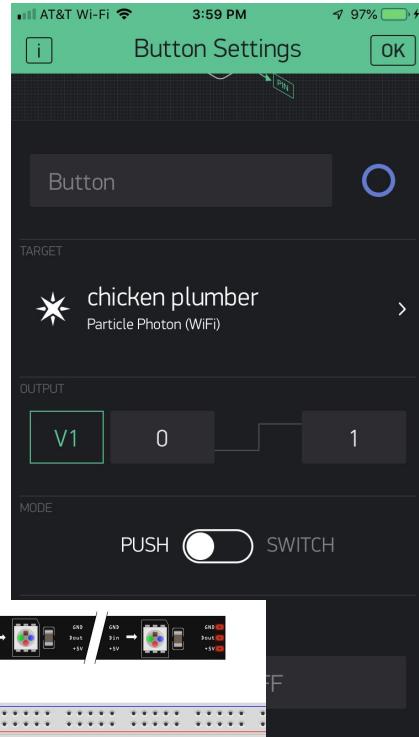
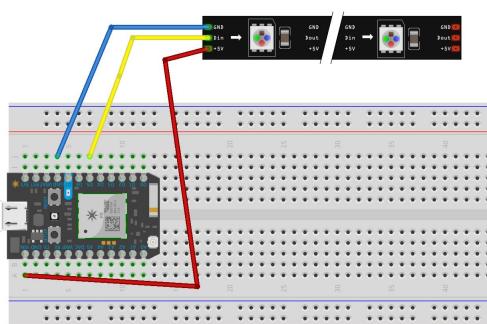
- Virtual pins allow Blynk to trigger advanced functions that can run on the Photon based on input from the app
- Called by simple code `BLYNK_WRITE(V1) {}` where V1 is the virtual pin
- Can have many many virtual pins in one app



Button to Drive Addressable LED

One touch fireworks

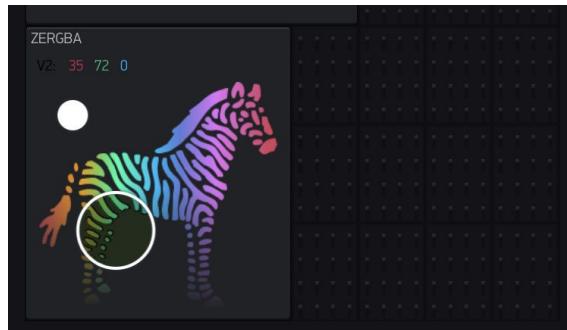
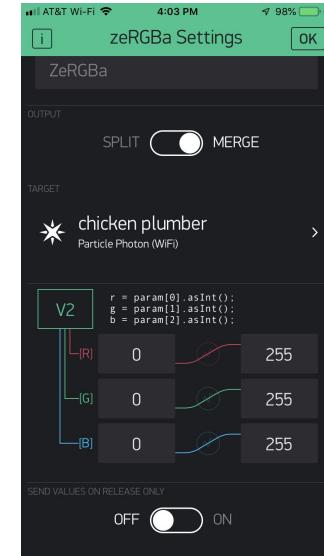
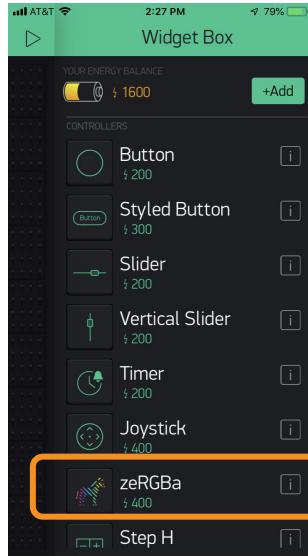
- Change the button on D7 to V1 and change the mode to “Push”
- Upload **sdq_blynk_write.ino** from the Github
- Connect LED ring to pin D6 and give it 5V power
- Start the Blynk app and push the button



RGB Picker

Add color on the fly

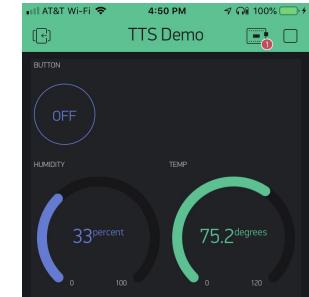
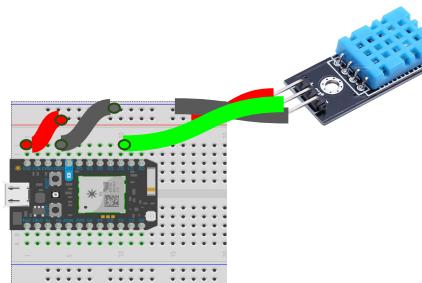
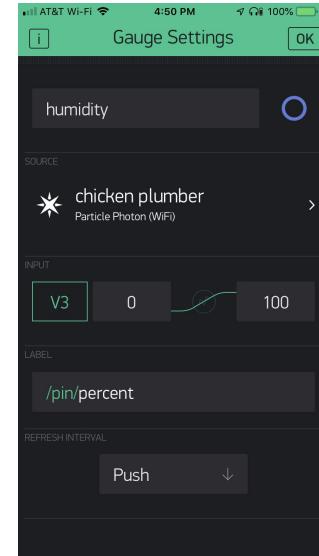
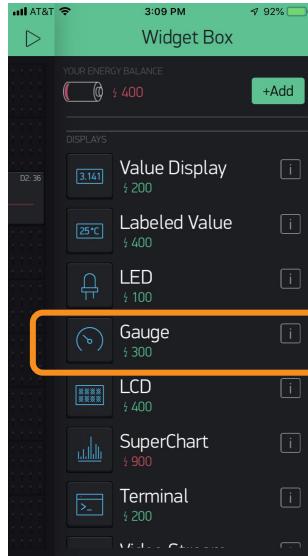
- Add a zeRGBA widget to your canvas
- Select V2 for the virtual pin and select “Merge” for the output option
- Turn “Send values on release only” to OFF
- Deploy widget and play app
- Swipe fingers across the zebra and notice the RGB LED on the Photon change color



Data Recording

Get DHT data to show in your app

- Push **sdq_blynk_dht11.ino** to your Photon. This pushes DHT data to virtual pins 3 and 4
- Add 2 new gauge widgets to your app. Select pin V3 for the humidity and V4 for the other for temperature

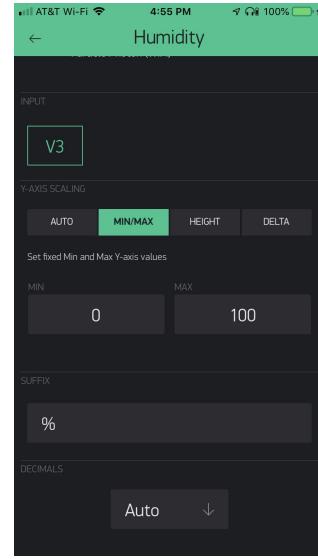
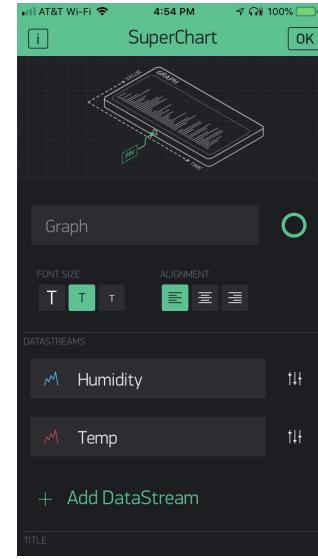
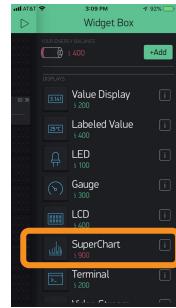


SuperChart

How good is this?

- Add SuperChart to your canvas (you may have to delete a couple of widgets to have enough energy (but it's worth it!)
- Edit the data stream to pull data from virtual pin V3
- Add a data stream and add temperature data from pin

V4



SuperChart

Play it and watch the magic

- Watch the values roll in
- Can rotate phone to landscape mode to get a better view
- Export your data in csv format



Thank you.

JEREMY LOSAW

jeremy.losaw@enventyspartners.com

www.enventypartners.com

www.jeremylosaw.com

