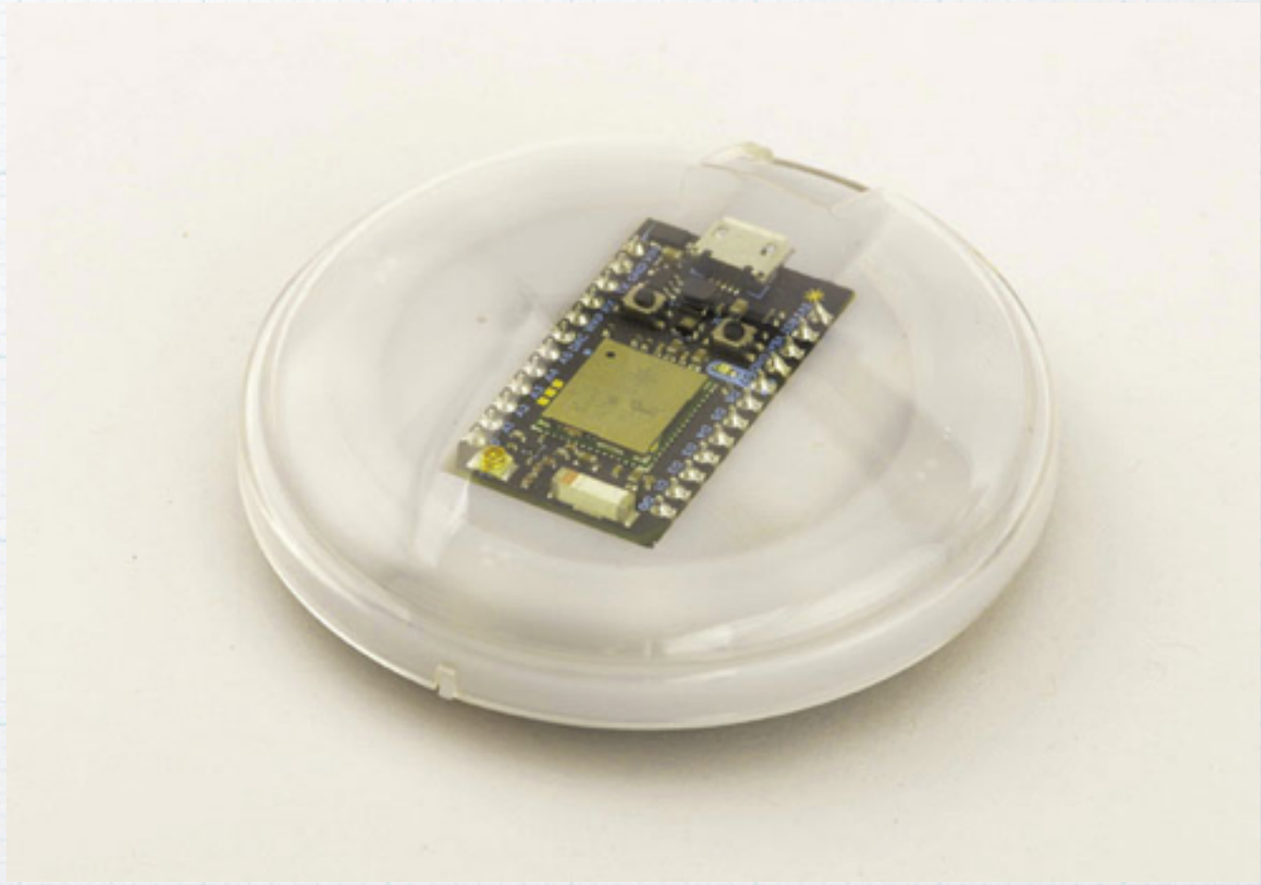
Anyone Can Code Photon/InternetButton

Lets get started

What are we going to learn??

* Setup the particle.io Photon
* Write programs for the Photon
* Write programs for Internet Button
* Learn how to use IFTTT
* Learn how to use Do button
* What is IoT?

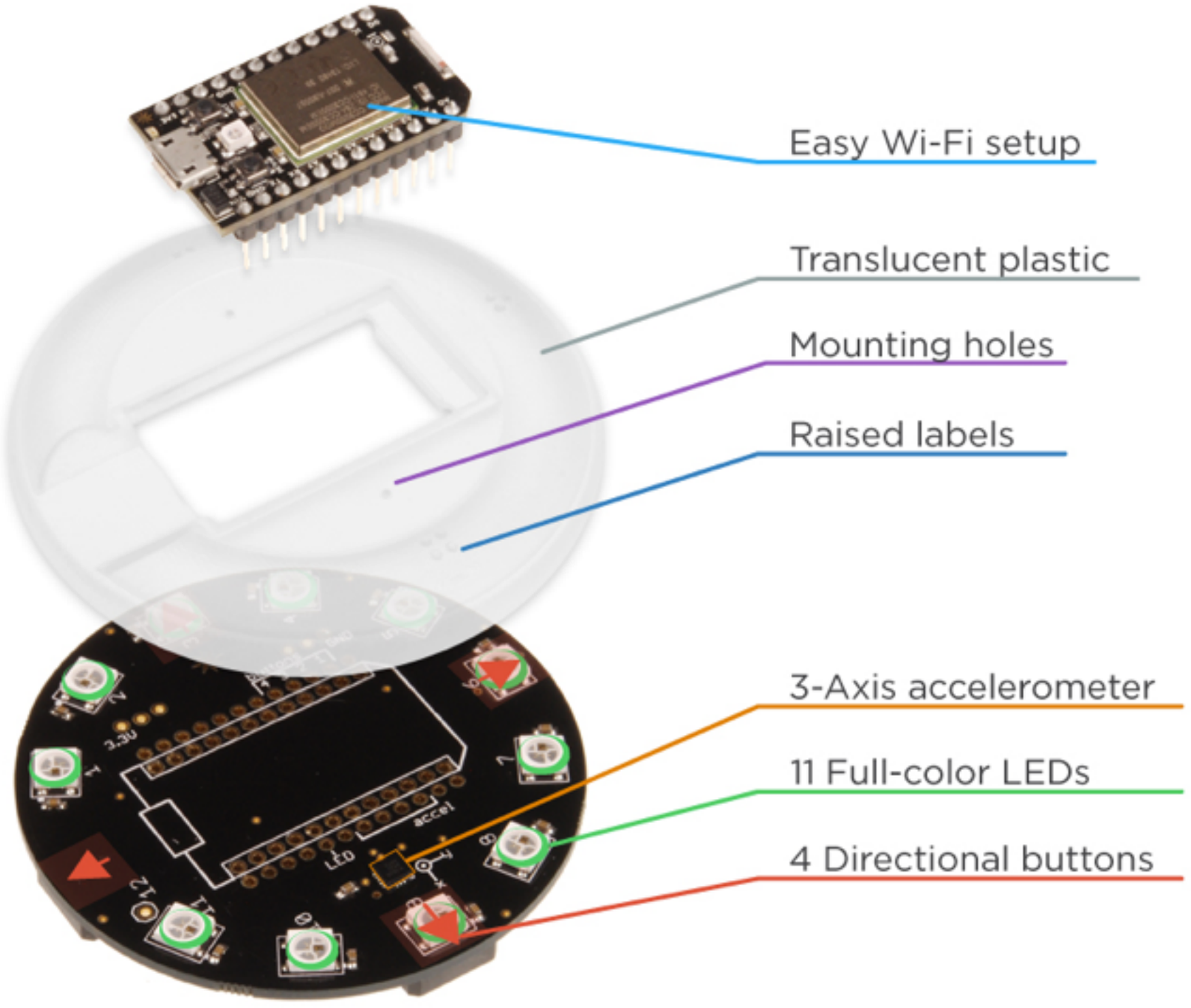


What's in the InternetButton

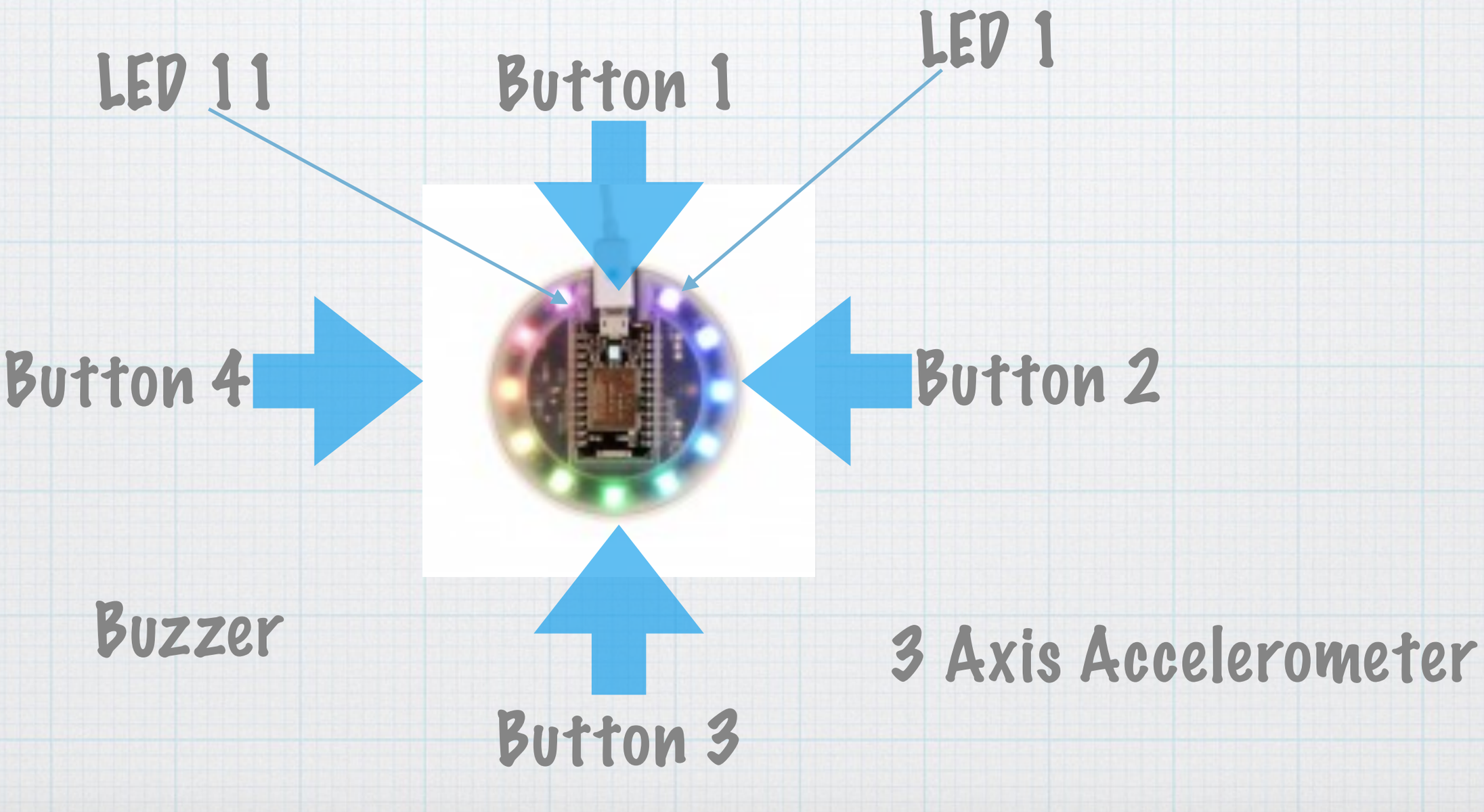
Includes Photon!
(yes, this is a Core)

Protective Cover

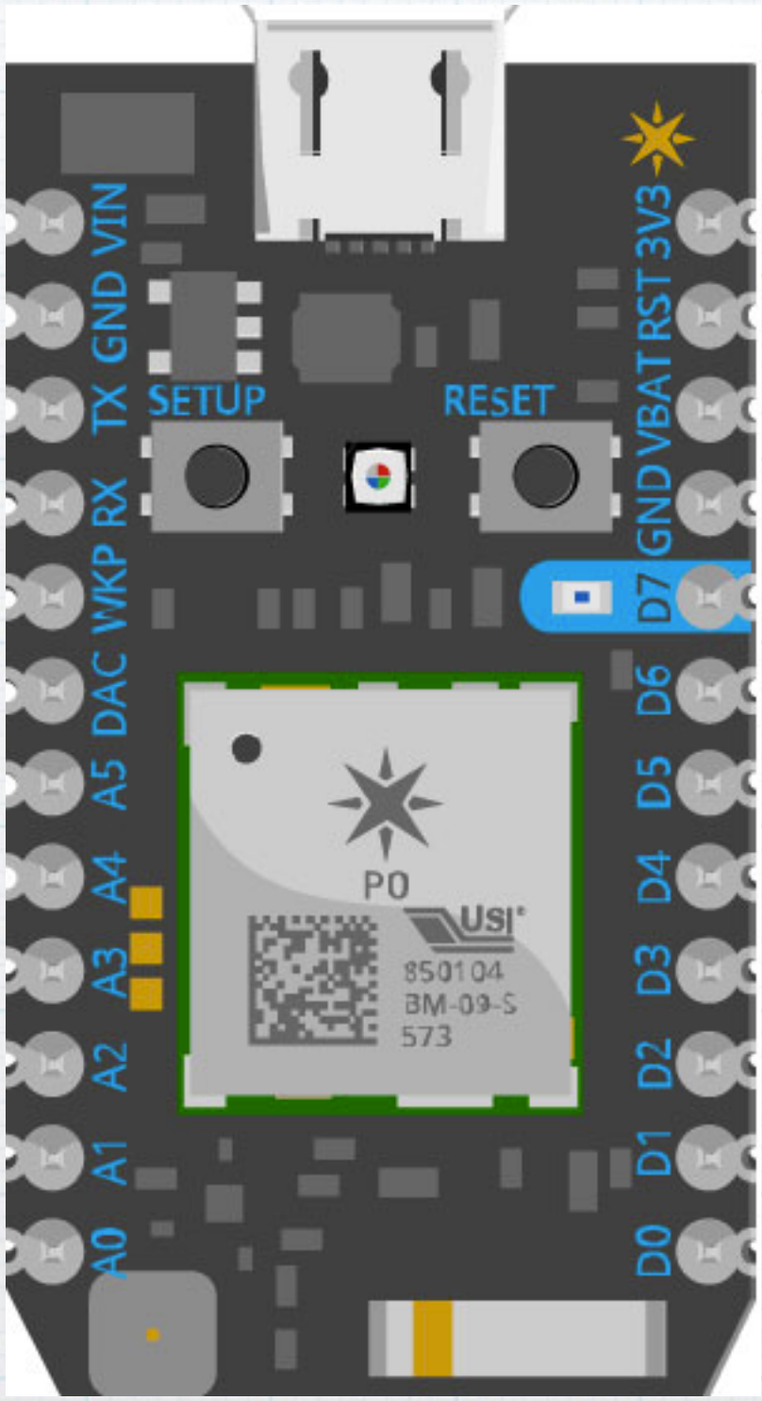
Button Components



What's in the InternetButton



What's in the Photon



WIFI enabled micro controller

3.3 Volt Inputs

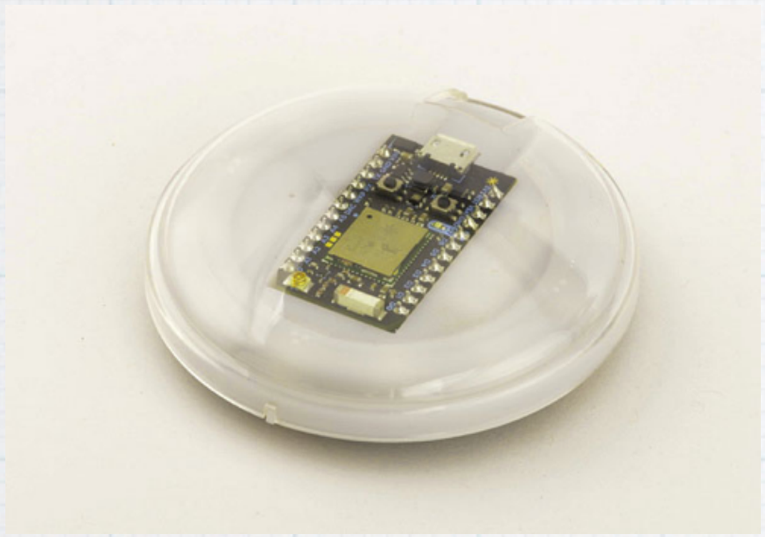
8 Analog Inputs

1 Digital to Analog output

8 Digital Input/Outputs

THE FUTURE
BELONGS TO
THE FEW OF US
STILL WILLING
TO GET OUR
HANDS DIRTY.

Tools

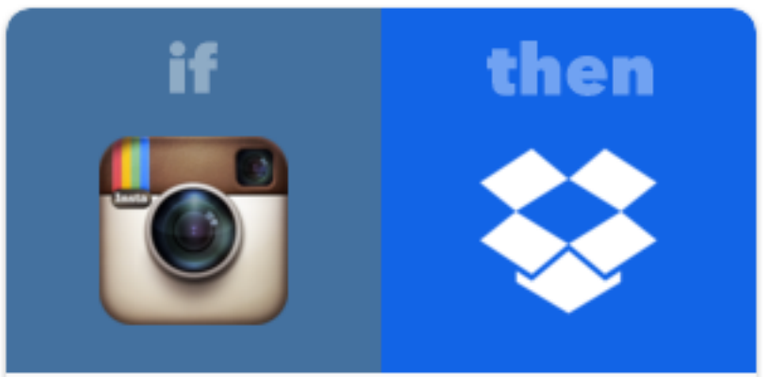


```
#include "InternetButton/InternetButton.h"
#include "math.h"

/*Did you know that the SparkButton can detect if
Specifically it can read when it's being accelera
is a constant acceleration and this becomes very

InternetButton b = InternetButton();

void setup() {
    //Tell b to get everything ready to go
    // Use b.begin(1); if you have the original S
    // to use, just add a '1' between the parenth
    b.begin();
}

void loop(){

    //How much are you moving in the x direction?
    int xValue = b.readX();

    //How about in the y direction?
    int yValue = b.readY();

    //And the z!
    int zValue = b.readZ();
```

DO RECIPE

Toggle Hue lights On/Off

IF RECIPE



If I post a picture on Instagram, save the photo to Dropbox

C/C++ programming Language

What we learn, and when

* Session 1
    * Create an account on Particle.io
    * Setup the Photon with Particle mobile application
    * Overview of the Particle web IDE
    * Write our first program and 'verify' program
    * Overview of Particle Dev IDE
    * Write our first program, verify program, flash program to InternetButton
* Session 2
    * Learn about Particle libraries and how to include them in our application
    * Write first InternetButton program, verify program and flash program to InternetButton
* Session 3
    * Learn about the particle.io events, variables and functions programing model
    * Learn how to publish and subscribe to custom events
    * Learn how to create watchable variables
    * Learn how to create functions that can be called from outside the Photon

What we learn, and when

* Session 4
  * Create IFTTT account
  * Create an IFTTT recipe
  * Learn how to use the InternetButton to publish an event to an IFTTT channel to send an email
  * Learn how to push a button to publish an event that will send a Pushover notification to your phone
* Session 5
  * Download the 'Do button' app and create a recipe that will cause the InternetButton to respond to a button press on your phone
  * Learn how to create a Do recipe that will allow you to push a button on your phone, and have it light up the InternetButton
* Session 6
  * You decide - create a project idea and implement it

What we are NOT going to learn

* Entire C / C++ Programming language constructs
* We will look at just enough C / C++ to program the InternetButton

30,000 Feet

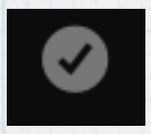
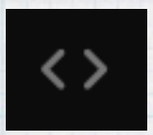
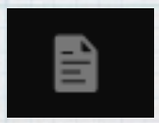
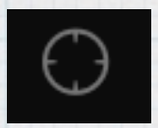
* What is IoT (Internet of Things)?
* What is the InternetButton and what is the Photon?
* Can you use the Photon outside of the InternetButton?
* What can I do with the Photon?

Session 1
Setup ParticleIO Account

* Do not plug in your InternetButton yet
* Create particle.io account
* Download Particle app to your phone

    * login into your account from Particle app
* One at a time!
    * plug in InternetButton into USB port
    * Photon should blink blue
        * If not, hold 'Setup' (on the left) button for 3 seconds or until it starts to flash blue
    * connect to Photon-XXXX wifi network on your phone
    * Go back to Particle App. If it does not connect then unplug usb and plug in again
    * Once the app connects to Photon, it will provide a list of WIFI points. Select one and login.
    * Name your InternetButton something you will remember (You will refer to it often)
* Your InternetButton should be 'breathing' a Cyan color
    * Your InternetButton is now on the Internet and registered with particle.io

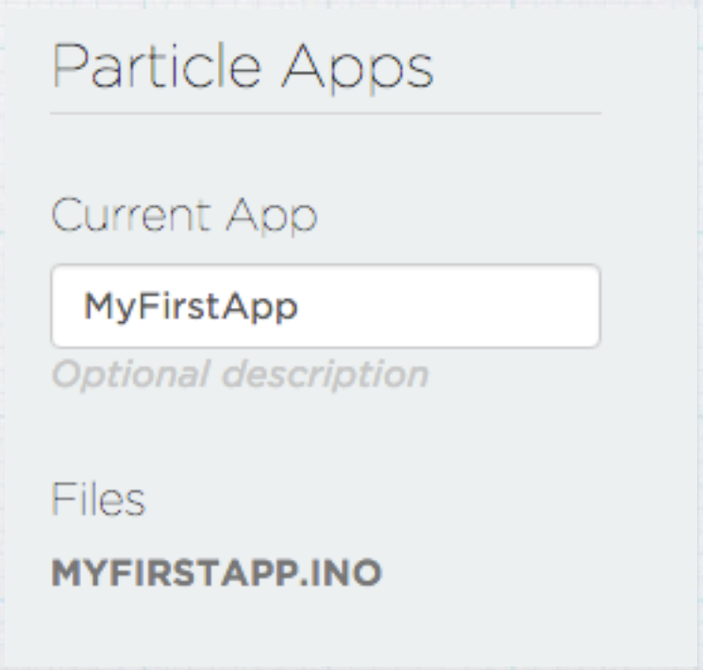
Session 1
ParticleIO IDE
https://build.particle.io

* Open particle build IDE in web browser:
*  Flash your program to the Photon
*  Verify/compile the program. Check syntax
*  Save any outstanding changes
*  Show / hide Code panel
*  Photon libraries
*  Photon programming documentation
*  Devices - important because you select your InternetButton
*  Settings

Session 1
Write our first program

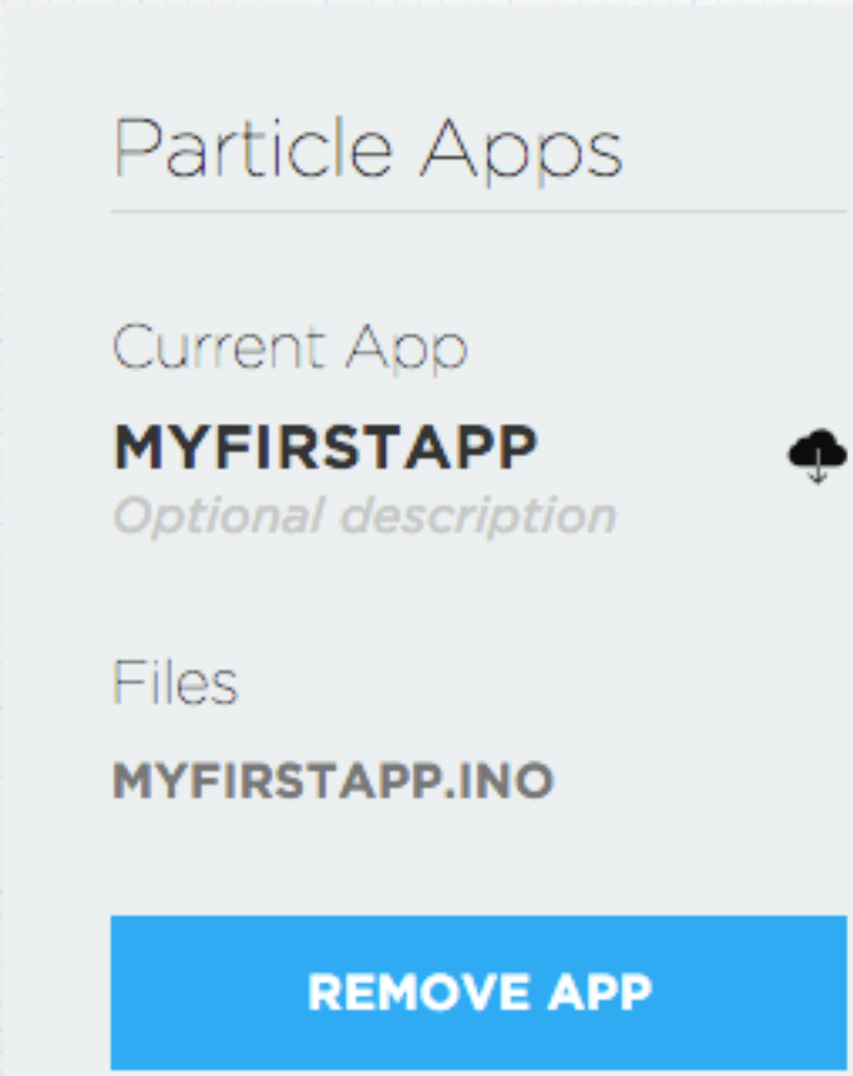
*  Select Devices
  * Make sure your device is selected
*  Show / hide Code Panel
* Select 

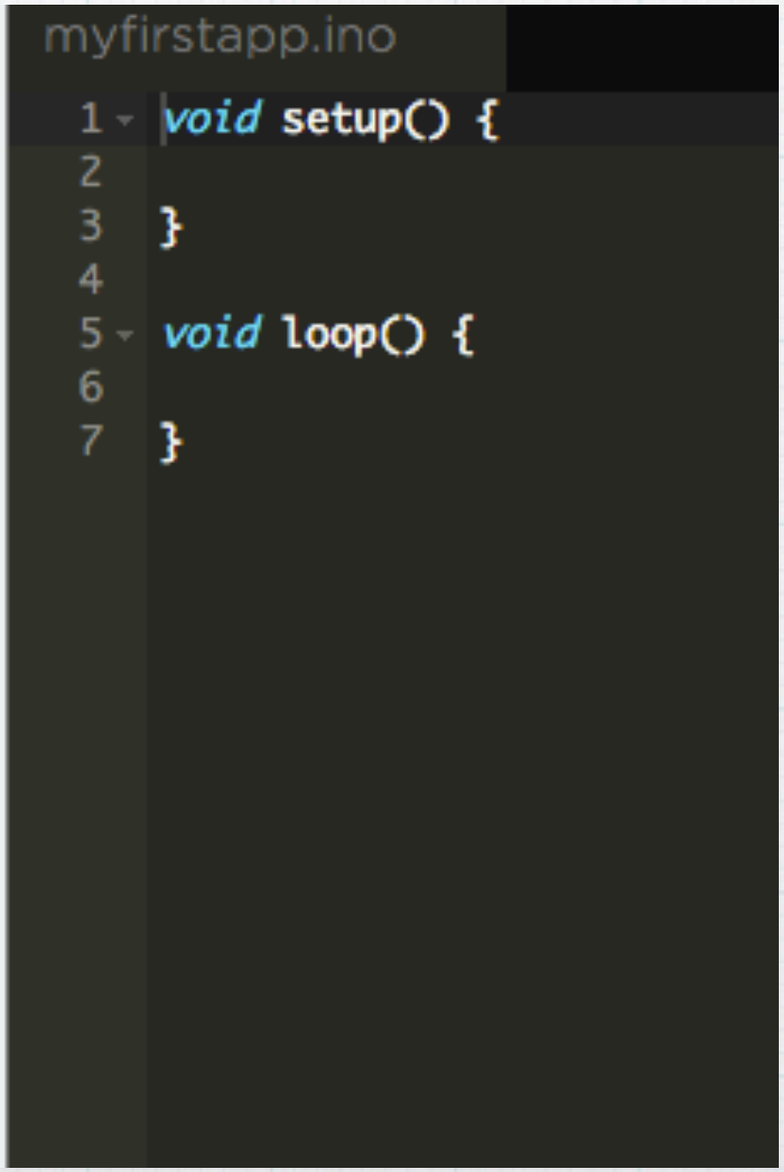
* Name you first app, hit return



Session 1
Write our first program

* Start of a new Photon application





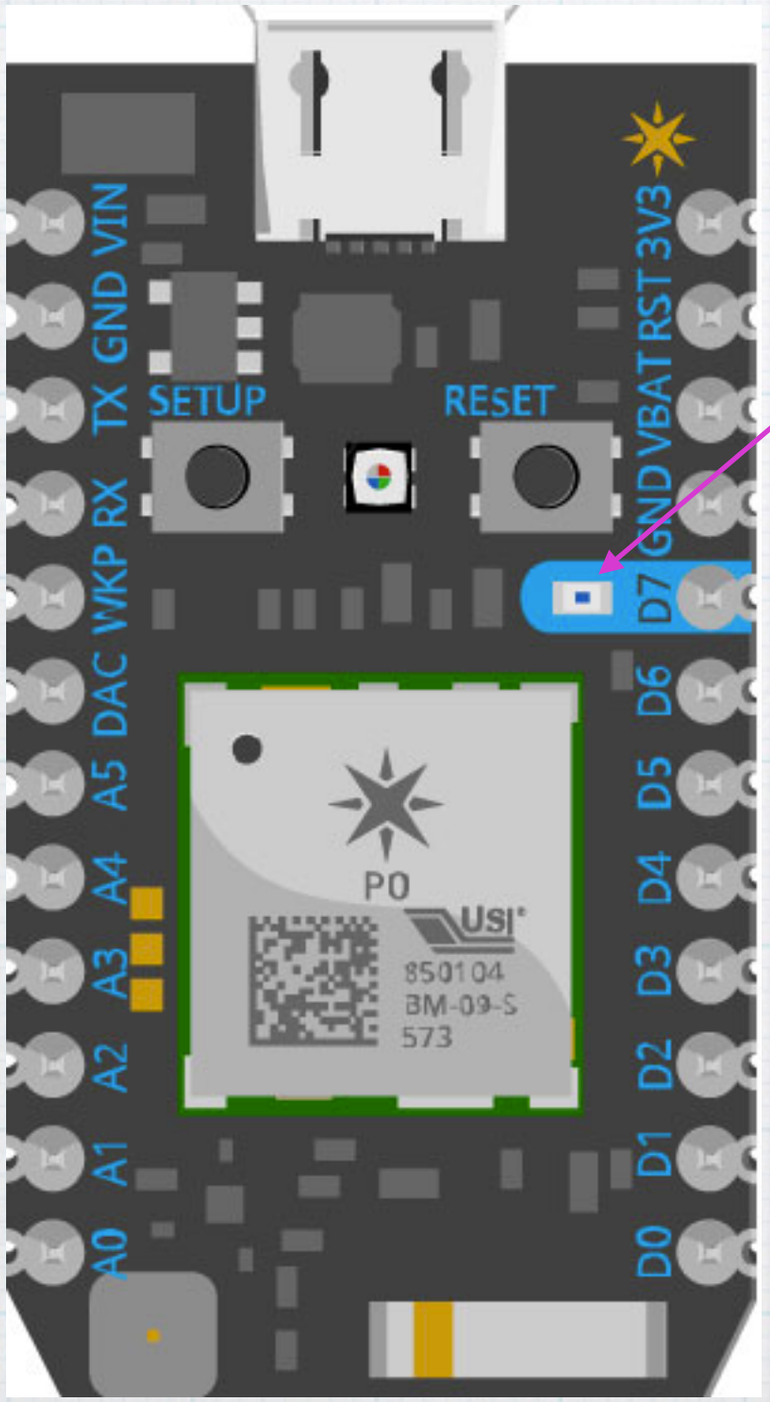
Session 1
Basic Program Structure

* void setup() { .... }
    * function called only once when the Photon starts up
    * Initialize variables, define variables.
    * Define pin direction (INPUT or OUTPUT)
    * Anything to properly set the initial state of the program and hardware
* void loop() { ..... }
    * function called repeatedly while the photon is on.
    * returning from loop will allow photon to take care of its business, then it calls loop() again as quickly as it can
    * you must return from loop periodically

Session 1
Using GPIO

* GPIO - General Purpose Input Output
* pinMode(pinId, INPUT or OUTPUT)
    * always execute pinMode in the setup function
    * Tells the photon whether to expect to write the to pin or read from the pin
    * pinId is of the form D0, D1, D2, D3, D4, D5, D6, D7
* digitalWrite(pinId, HIGH or LOW)
    * sets the voltage or value on pinId to either HIGH (3.3v) or LOW (0v)
    * HIGH means provide voltage to turn on LED. LOW means remove voltage and the LED will turn off.

Session 1
Photon



Digital pin, D7, has a built in blue LED

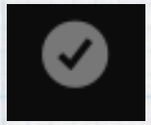
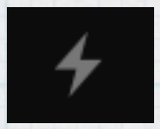
Our first program will turn the blue LED on and off

Session 1
Example Program

```
1
2  // This routine runs only once upon reset
3  void setup() {
4    // Initialize D7 pin as output
5    // It's important you do this here, inside the setup() function rather than outside it or in the loop function.
6    pinMode(D7, OUTPUT);
7  }
8
9  // This routine gets called repeatedly, like once every 5-15 milliseconds.
10 // Particle firmware interleaves background CPU activity associated with WiFi + Cloud activity with your code.
11 // Make sure none of your code delays or blocks for too long (like more than 5 seconds), or weird things can happen.
12 void loop() {
13   digitalWrite(D7, HIGH);
14   delay(1000);              // Wait for 1000mS = 1 second
15   digitalWrite(D7, LOW);
16   delay(1000);              // Wait for 1 second in off mode
17 }
```

Line 3: How you declare a function
Line 6: calling a function called pinMode
All lines end with semi-colon (;)
Every statement on its own line
A function is wrapped with { and ends with }

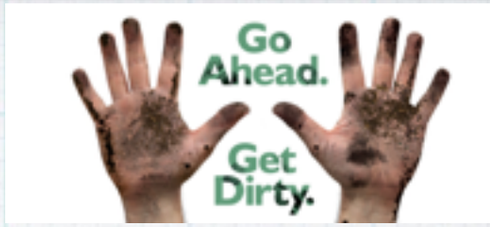
Session 1
Save, Verify, Flash

* Open particle build IDE in web browser
*  Save any outstanding changes
*  Verify/compile the program. Check syntax
  * Should see 'Code verified. Good Work!'
*  Flash your program to the Photon

```
Flash successful! Please wait a moment while your device is updated...
```

LED will turn Magenta, Off, White, flash Green, Breathing Cyan

Session 1 Project



* Create a program to blink the onboard LED at digital output D7 every second

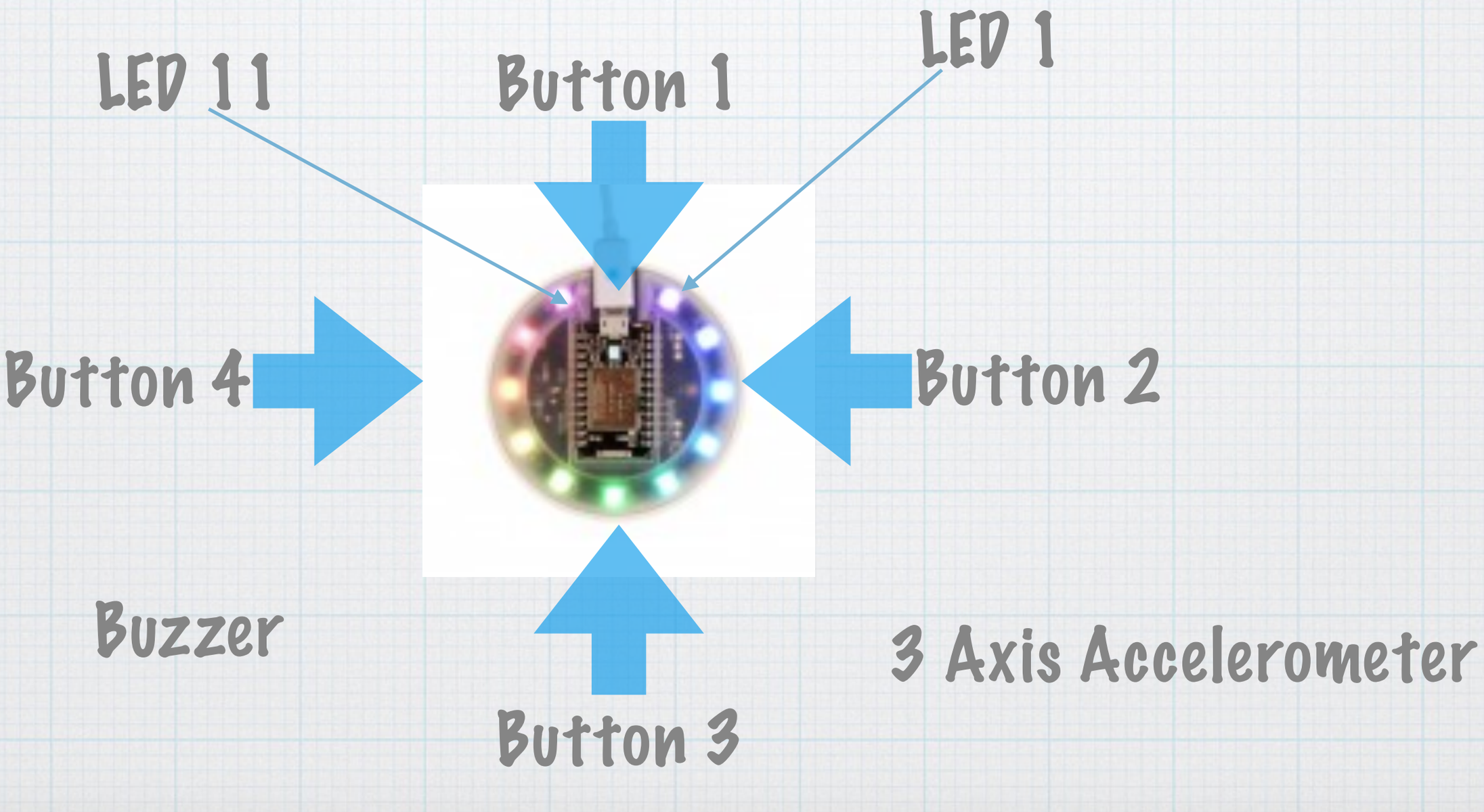
* Look back at the Example Program

* If it works the onboard BLUE LED should be blinking every second

Session 2
InternetButton

* Learn how to create InternetButton programs
* Learn how to include Particle libraries
* Learn how to interact with InternetButton LEDs
* Learn how to interact with InternetButton buttons

What's in the InternetButton



Session 2
Libraries

* Learn how to add Libraries to our applications
    * Are already written pieces of software that we can use.
* We do not have to write everything
    * learn to leverage the work of others
* We are going to learn to include the InternetButton library

Session 2
Libraries and the InternetButton

* <> Show / Hide Apps Panel

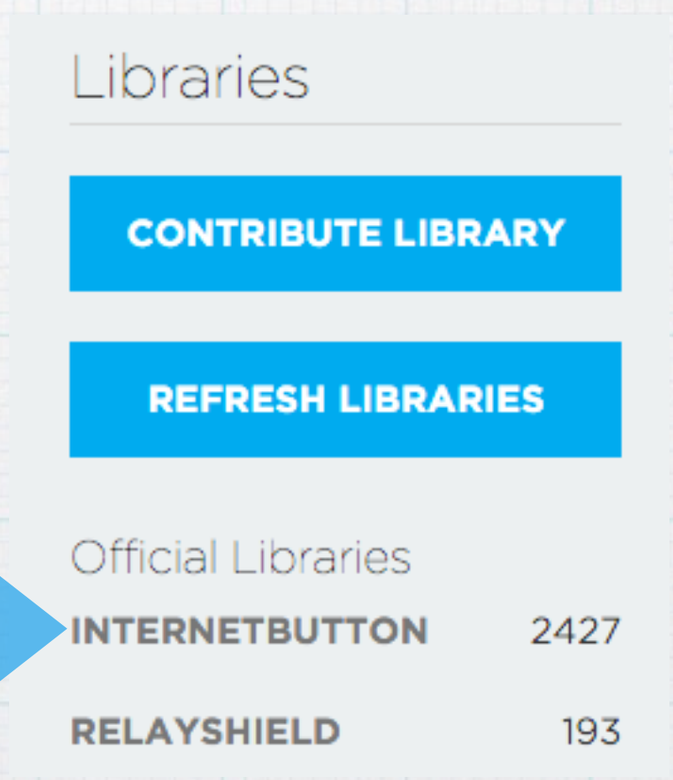
* Select CREATE NEW APP

Session 2
Libraries and the InternetButton

* To use the InternetButton we need to include the InternetButton library

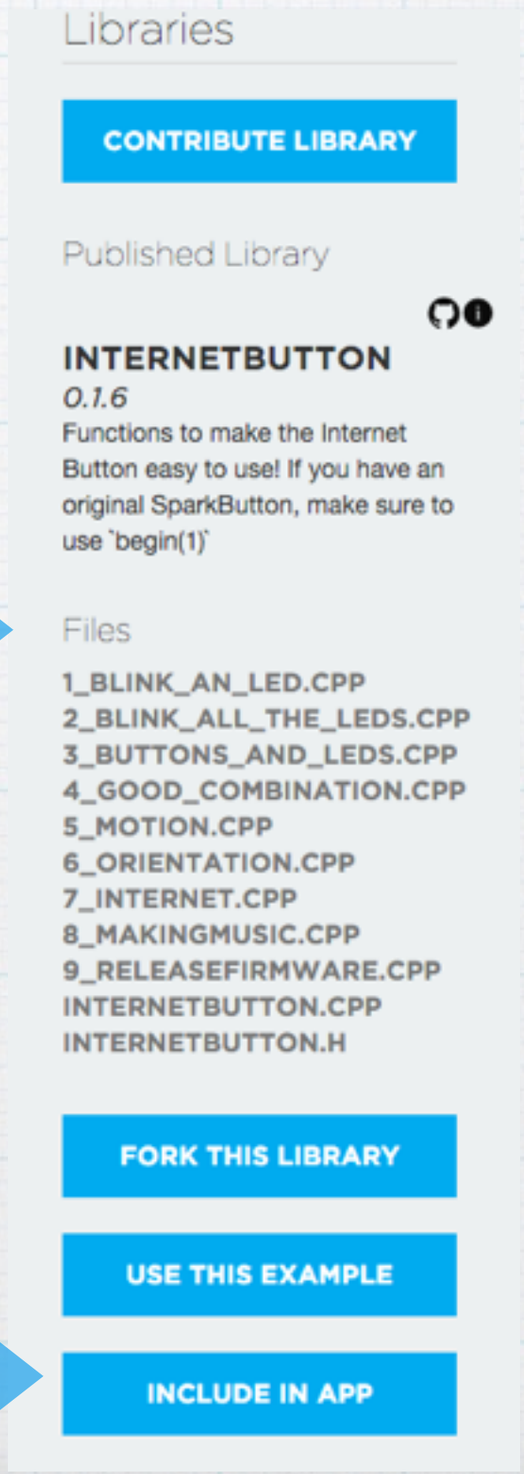
* Photon Libraries

Select InternetButton



Session 2
Libraries and the InternetButton

The '_' files are great examples



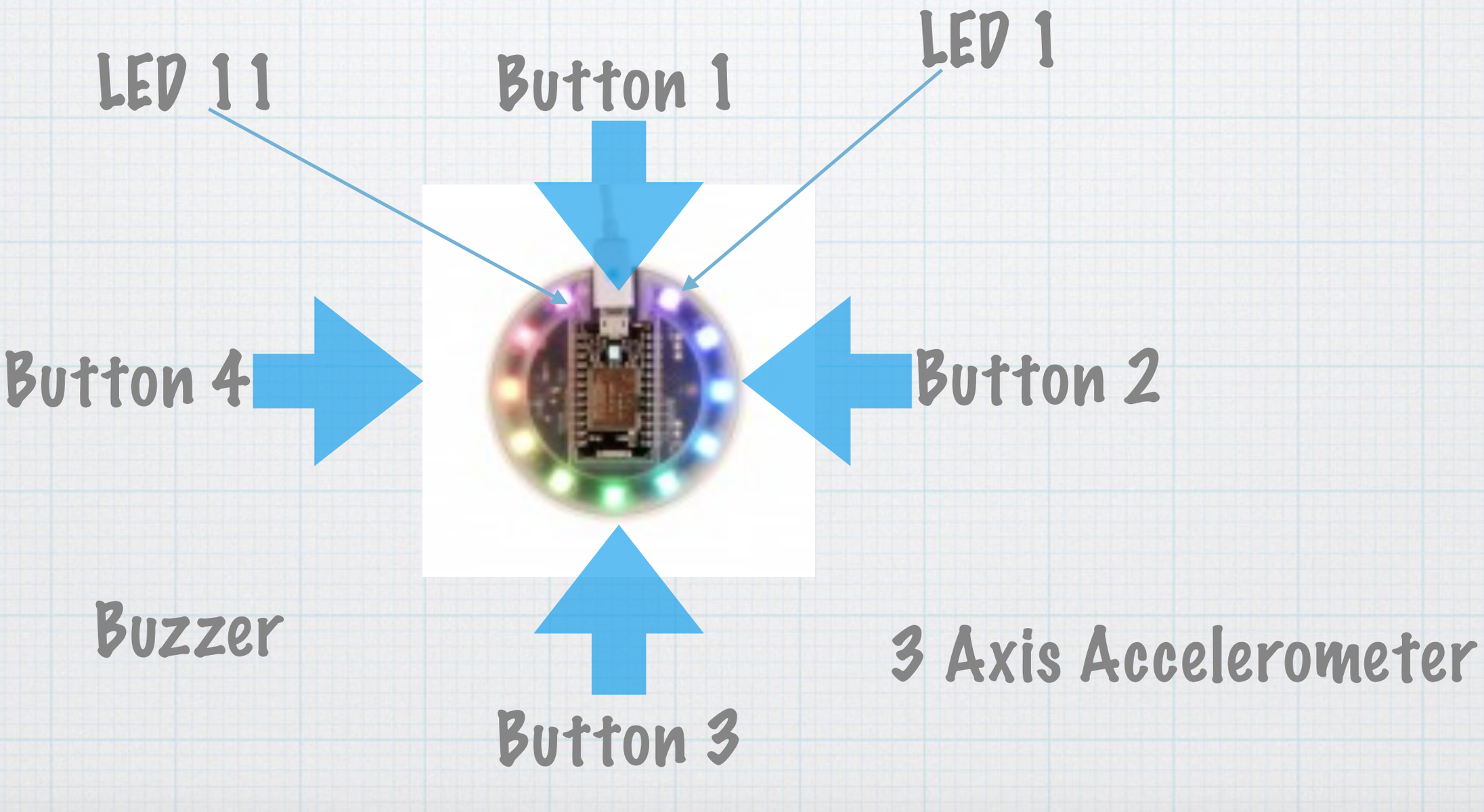
Select to include in application
Then select your application

Session 2
Libraries and the InternetButton

* The Web IDE should look something like:

Select 'INTERNETBUTTON' library to see the examples

What's in the InternetButton



Session 2
InternetButton instance

* Create a software instance representation of the InternetButton

mysecondapp.ino

```
1  #include "InternetButton/InternetButton.h"
2
3  InternetButton b = InternetButton();
```

* The variable 'b' is how you interact with the physical InternetButton
    * variable can be named anything:
        * b, interntButton, iButton, the_button

Session 2
Initialize InternetButton

* In setup(), initialize the software instance of the InternetButton

```
5  void setup() {
6      b.begin();
7  }
```

* b.begin() should only be called once
    * This is why we do this in setup()

Session 2
ledOn / ledOff

* b.ledOn(LED Number, Red, Green, Blue)

```
9  void loop() {
10     b.ledOn(6, 0, 0, 255);
11     // The format here is (LED, red, green, blue), so we're making a color with no red or green, but ALL the blue
12
13     delay(1000);
14
15     b.ledOff(6);
16     delay(1000);
17 }
```

* delay(milliseconds)
    * 1 second = 1000 milliseconds
* b.ledOff(LED number)

Always preface the calls with the InternetButton variable name. In this case 'b'

Session 2
Red / Green / Blue

* ledOn(Led #, RedValue,GreenValue,BlueValue)
* Each value is a number between 0 and 255
    * Red: 255, 0, 0
    * Green: 0, 255, 0
    * Blue: 0, 0, 255
    * Cyan: 0, 255, 255
    * Purple: 128, 0, 128
    * Yellow: 255, 255, 0
    * White: 255, 255, 255
* Make any color by changing the values

Session 2
ledOn / ledOff

Program will blink this LED blue

LED 6

Red: 0
Green: 0
Blue: 255

Session 2
allLedsOn / allLedsOff

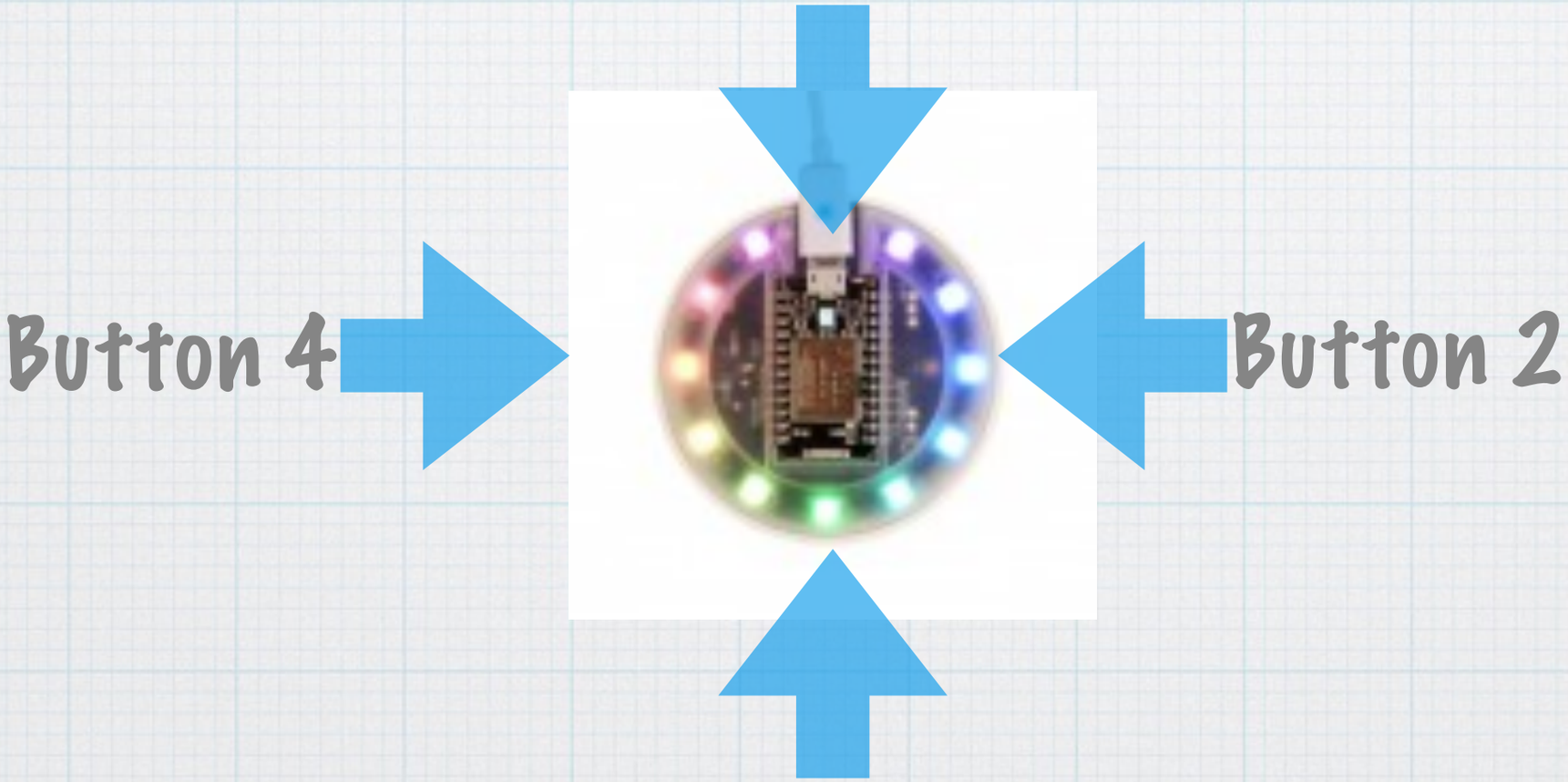
* Blink all LEDs - allLedsOn

mysecondapp.ino

```
1  #include "InternetButton/InternetButton.h"
2
3  InternetButton b = InternetButton();
4
5  void setup() {
6      b.begin();
7  }
8
9  void loop() {
10     b.allLedsOn(0,20,20);
11     delay(1000);
12     b.allLedsOff();
13     delay(1000);
14 }
```

Session 2
Buttons

Learn when a button is pressed



Session 2
Buttons

* Lets see how to turn on an LED when a button is pressed

  * buttonOn(button_number)

    * returns true if button is pressed

    * returns false if button is not pressed

Session 2
Buttons and LEDs

* Buttons and LEDs

mysecondapp.ino

```
1  #include "InternetButton/InternetButton.h"
2
3  InternetButton b = InternetButton();
4
5  void setup() {
6      b.begin();
7  }
8
9  void loop(){
10     // When you click the second button (at the 3 o'clock position) let's turn that LED on
11     if(b.buttonOn(2)){
12         b.ledOn(3, 255, 255, 255);
13     }
14     // And if the button's not on, then the LED should be off
15     else {
16         b.ledOff(3);
17     }
18 }
19
```

b.buttonOn(2) - returns true if button2 is pressed

b.ledOn(3,255,255,255) - turn on LED 3 as bright white

b.ledOff(3) - turn off LED 3

Session 2
If / else

* 'if' true
  * then execute code in the if { block }
* 'else'
  * execute the code in the else { block }
* if / else if / else
  * multiple if / else blocks

```
if( true ) {
  if block
} else {
  else block
}
```

```
if( true ) {
  if block
} else if(true) {
  else if block
} else {
  else block
}
```

Session 2
if / else Example

```
void x() {
    if (b.buttonOn(1)) {
        // if block
    } else if (b.buttonOn(2)) {
        // else if block
    } else {
        // else block
    }
}

void y() {
    if (b.buttonOn(1)) {
        // if block
    }
    if (b.buttonOn(2)) {
        // if block
    } else {
        // else block
    }
}
```

Session 2
Buttons and LEDs

mysecondapp.ino

```
#include "InternetButton/InternetButton.h"

InternetButton b = InternetButton();

void setup() {
    b.begin();
}

void loop(){
    // When you click the second button (at the 3 o'clock position) let's turn that LED on
    if(b.buttonOn(2)){
        b.ledOn(3, 255, 255, 255);
    }
    // And if the button's not on, then the LED should be off
    else {
        b.ledOff(3);
    }
}

```

b.buttonOn(2) - returns true if button2 is pressed
b.ledOn(3,255,255,255) - turn on LED 3 as bright white
b.ledOff(3) - turn off LED 3

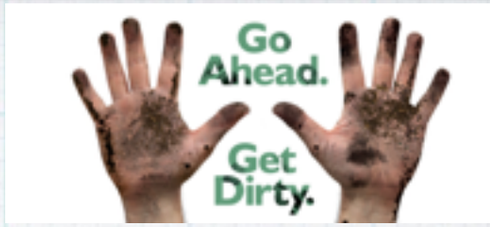
Session 2
Rainbow

* Built in method to show a rainbow of colors
  * rainbow(time)
    * e.g. b.rainbow(10)
* Will cause the InternetButton to turn on all of the LEDs in different colors.

Session 2 Summary

* To turn on single LED
    * ledOn(led_number, red,green,blue)
* To turn on ALL LEDs
    * allLedsOn(red,green,blue)
* To check for a button press
    * buttonOn(button_number)
* Rainbow
    * rainbow(time in ms)
    * built in delay after showing rainbow
    * remember to turn off the less

Session 2
Project 1



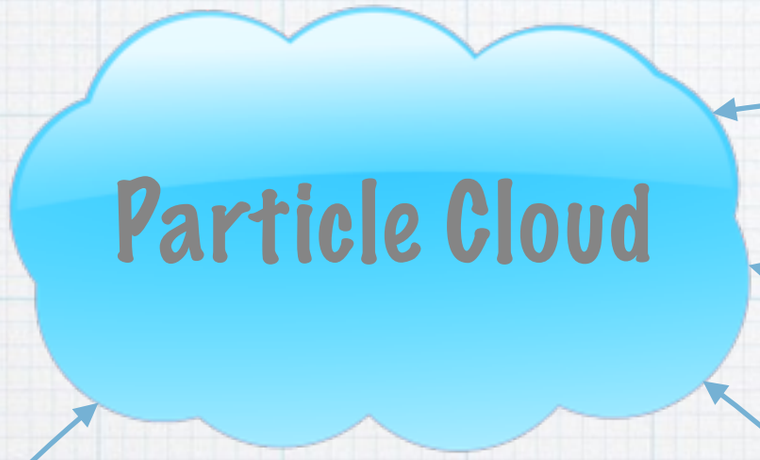
* Create a program to blink all of the LEDs your favorite color

* very similar to the first project, but instead of blinking the onboard single LED - blink all of the Red/Green/Blue LEDs of the internet button

Session 2
Project 2

* Initially start with all of the LEDs off
* When button 1 is pressed, turn all LEDs red.
* When button 2 is pressed, turn all LEDs green.
* When button 3 is pressed, turn all LEDs blue.
* When button 4 is pressed, then on the rainbow.

Session 3
Connecting to the Cloud



Events

Functions

Variables

Session 3
Particle Events

* Publish events to internet services
* Subscribe to events from internet services
* Particle.publish
  * String eventName (63 character max)
  * String data (255 bytes max) - Optional
  * int ttl - time to live in seconds
  * PRIVATE - flag to make the event private to your devices
* Events are published when you decide to publish them

Session 3
Particle Publish Events

* Examples - called from loop() method
  * Particle.publish("Door Open");
  * Particle.publish("Door Open", "Front");
  * Particle.publish("Door Open", "Front", 60, PRIVATE);
* **Caution**:
  * loop() is called frequently so you must guard the Particle.publish method so it is not called more than once ever few seconds. Typically once per 15 seconds
* E.g. When a button is pressed, publish an event

Session 3
Particle Publish Events

* Buttons and Published Events

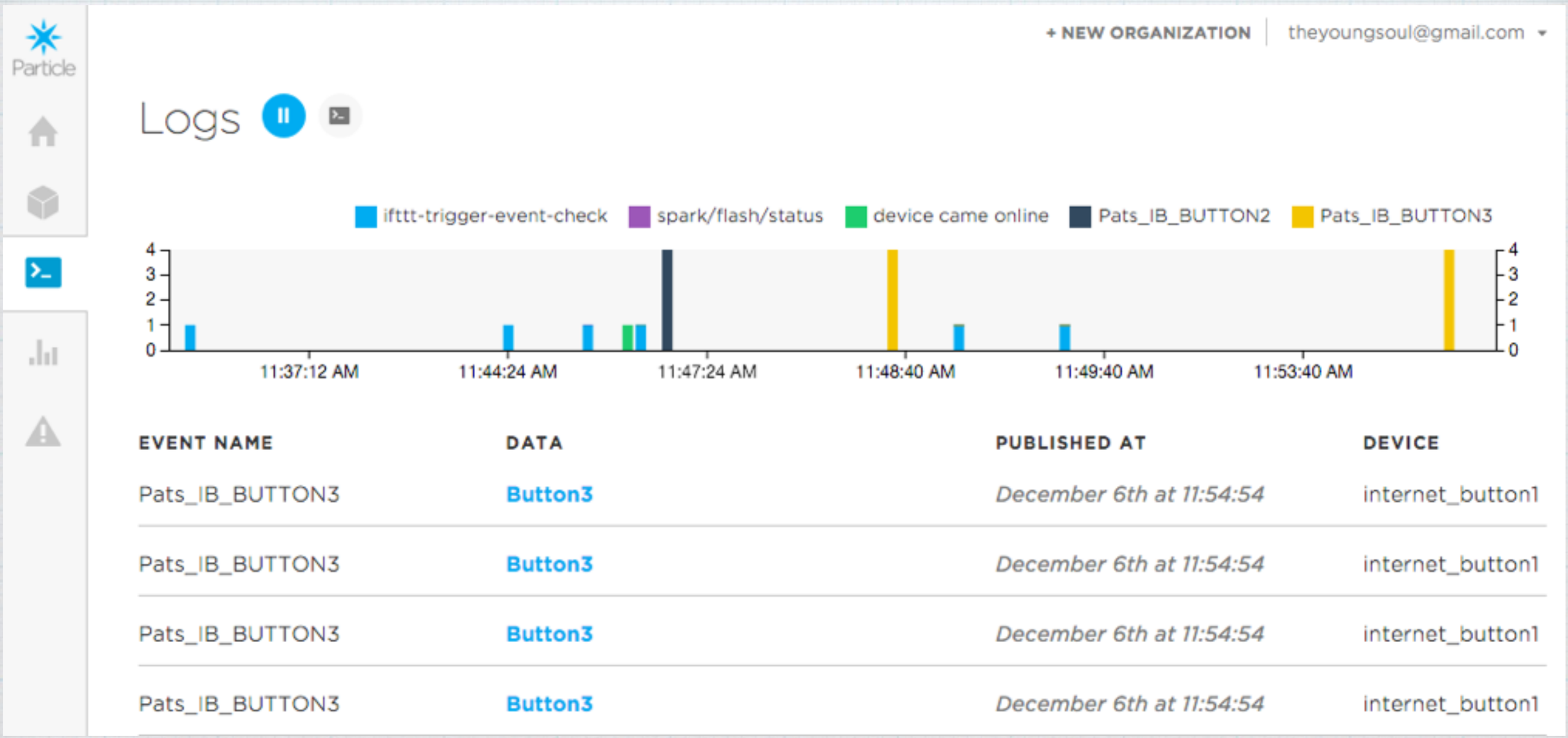
Warning!

This example will publish multiple events very quickly

```
#include "InternetButton/InternetButton.h"

InternetButton b = InternetButton();

void setup() {
    b.begin();
}

void loop(){

    // as written, multiple events will be published because the loop() is called
    // very quickly while the button is still 'on'
    if(b.buttonOn(2)) {
        Particle.publish("Pats_IB_BUTTON2");
    }

    if(b.buttonOn(3)) {
        Particle.publish("Pats_IB_BUTTON3", "Button3");
    }

    if(b.buttonOn(4)) {
        Particle.publish("Pats_IB_BUTTON4", "Button4", 60, PRIVATE);
    }

}
```

Session 3
Particle Publish Events

* Particle Dashboard



Session 3
Particle Publish Events

Buttons and Published Events

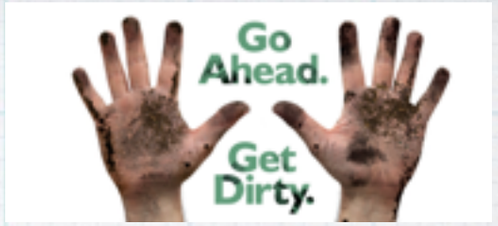
Create flag to know when button is pressed

Check to see if the button is already pressed

If not, then publish event and set flag to true

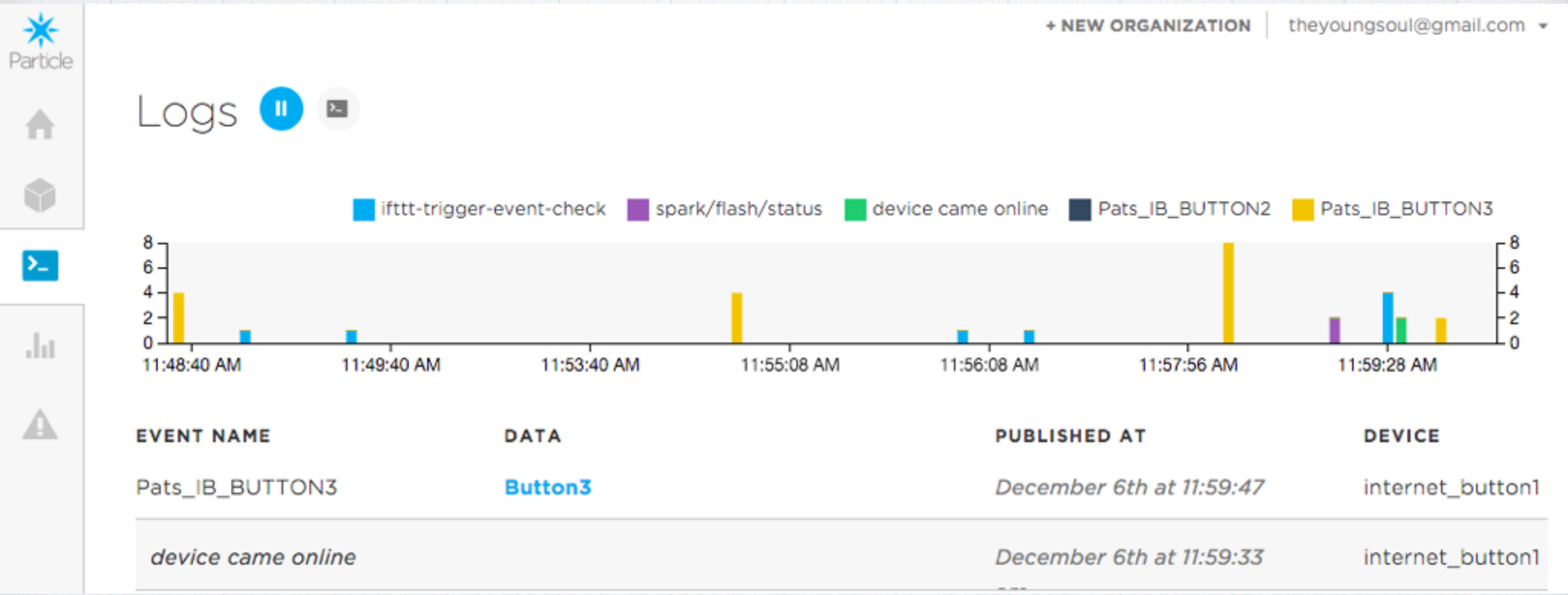
If all buttons are not pressed, then reset all flags

```
1  #include "InternetButton/InternetButton.h"
2
3  InternetButton b = InternetButton();
4
5  void setup() {
6      b.begin();
7  }
8
9  bool button2Pressed = false;
10 bool button3Pressed = false;
11 bool button4Pressed = false;
12
13 void loop(){
14
15     // will not allow for multiple events per button press
16     if(b.buttonOn(2) && button2Pressed == false) {
17         Particle.publish("Pats_IB_BUTTON2");
18         button2Pressed = true;
19     }
20
21     if(b.buttonOn(3) && button3Pressed == false) {
22         Particle.publish("Pats_IB_BUTTON3", "Button3");
23         button3Pressed = true;
24     }
25
26     if(b.buttonOn(4) && button3Pressed == false) {
27         Particle.publish("Pats_IB_BUTTON4", "Button4", 60, PRIVATE);
28         button4Pressed = true;
29     }
30
31     if(b.allButtonsOff()) {
32         button2Pressed = false;
33         button3Pressed = false;
34         button4Pressed = false;
35     }
36 }
```



Session 3
Particle Publish Events

* Particle Dashboard



Session 3
Particle Subscribe Events

- Listen for when events are published, and execute a function
- Subscribe to a maximum 4 events
- Examples - called from loop() method
  - Particle.subscribe("Door Open", myHandler);
  - Typically called in setup() - only needed to be done once
- void myHandler(String event, String data)
  - event - name of the event that was published.
  - data - data string, if any, that was published
  - Particle.publish("Pats_IB_Button3", "Button3");

Particle.publish("Pats_IB_Button3", "Button3"); -> Particle.Subscribe("Pats_IB_Button3", myHandler);

Session 3
Particle Variables

* Allows the value of a variable to be monitored by another service
* The other service 'pulls' or 'reads' the value when the service is ready.
    * unlike events that are pushed when the Photon program decides to send them
* Events are pushed, variable values are pulled.
* 3 Supported Data Types
    * INT
    * Double
    * String (622 bytes max)
* Maximum of 10 cloud variables per Photon
* Variables names have a maximum length of 12 characters

Session 3
Particle Variables

```
1  #include "InternetButton/InternetButton.h"
2
3  InternetButton b = InternetButton();
4
5  bool button2Pressed = false;
6  int button2PressCount = 0;
7  String message = "Initial Message";
8
9  void setup() {
10     b.begin();
11     Particle.variable("buttonCount", button2PressCount);
12     Particle.variable("status", message);
13 }
14
15
16 void loop(){
17
18     // will not allow for multiple events per button press
19     if(b.buttonOn(2) && button2Pressed == false) {
20         button2PressCount = button2PressCount + 1;
21         message = "Button 2 is Pressed";
22     }
23
24     if(b.allButtonsOff()) {
25         message = "Button 2 is not Pressed";
26     }
27 }
```

Session 3
Particle Variables

* We will see how to read variable values in the IFTTT section.

Session 3
Particle Functions

* Exposes a function that can be called by another service
* Maximum of 15 cloud functions per Photon
* Function name maximum is 12 characters
* Takes a single String argument with maximum of 63 characters
* Functions return an integer
    * negative numbers typically indicate an error

Session 3
Particle Functions

```
#include "InternetButton/InternetButton.h"

InternetButton b = InternetButton();

int turnOnLed(String ledNumber);
int turnOffLed(String ledNumber);

void setup() {
    b.begin();
    Particle.function("ledOn", turnOnLed);
    Particle.function("ledOff", turnOffLed);
}


void loop(){
    // NOTICE... there is nothing to do in the loop method
    // we are waiting for the defined functions to be
    // called.
}

int turnOnLed(String ledNumber) {
    if( ledNumber == "led1") {
        b.ledOn(1, 0,0,64);
    } else if( ledNumber == "led2") {
        b.ledOn(2, 0,0,64);
    }
    // continue with the rest
}
int turnOffLed(String ledNumber) {
    if( ledNumber == "led1") {
        b.ledOff(1);
    } else if( ledNumber == "led2") {
        b.ledOff(2);
    }
    // continue with the rest
}
```

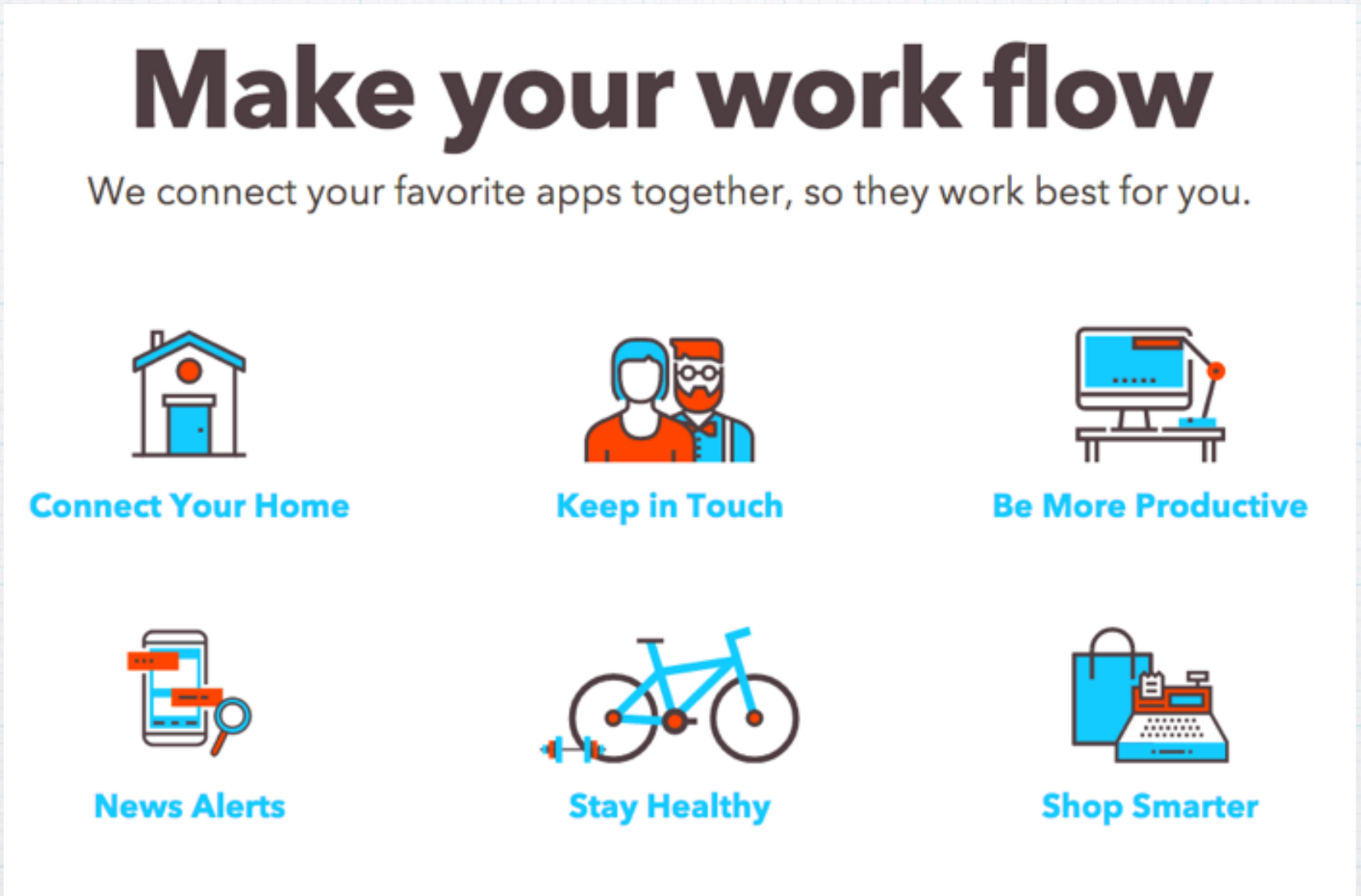
Session 3
Particle Functions

* Note: the exposed name of the function is not the same as the actual function name
    * exposed function name: ledOn
        * actual function name: turnOnLed
* Function can be arbitrarily complex
* Defers the overhead of complex, time consuming computation to when the service needs them

Session 3
Project 1

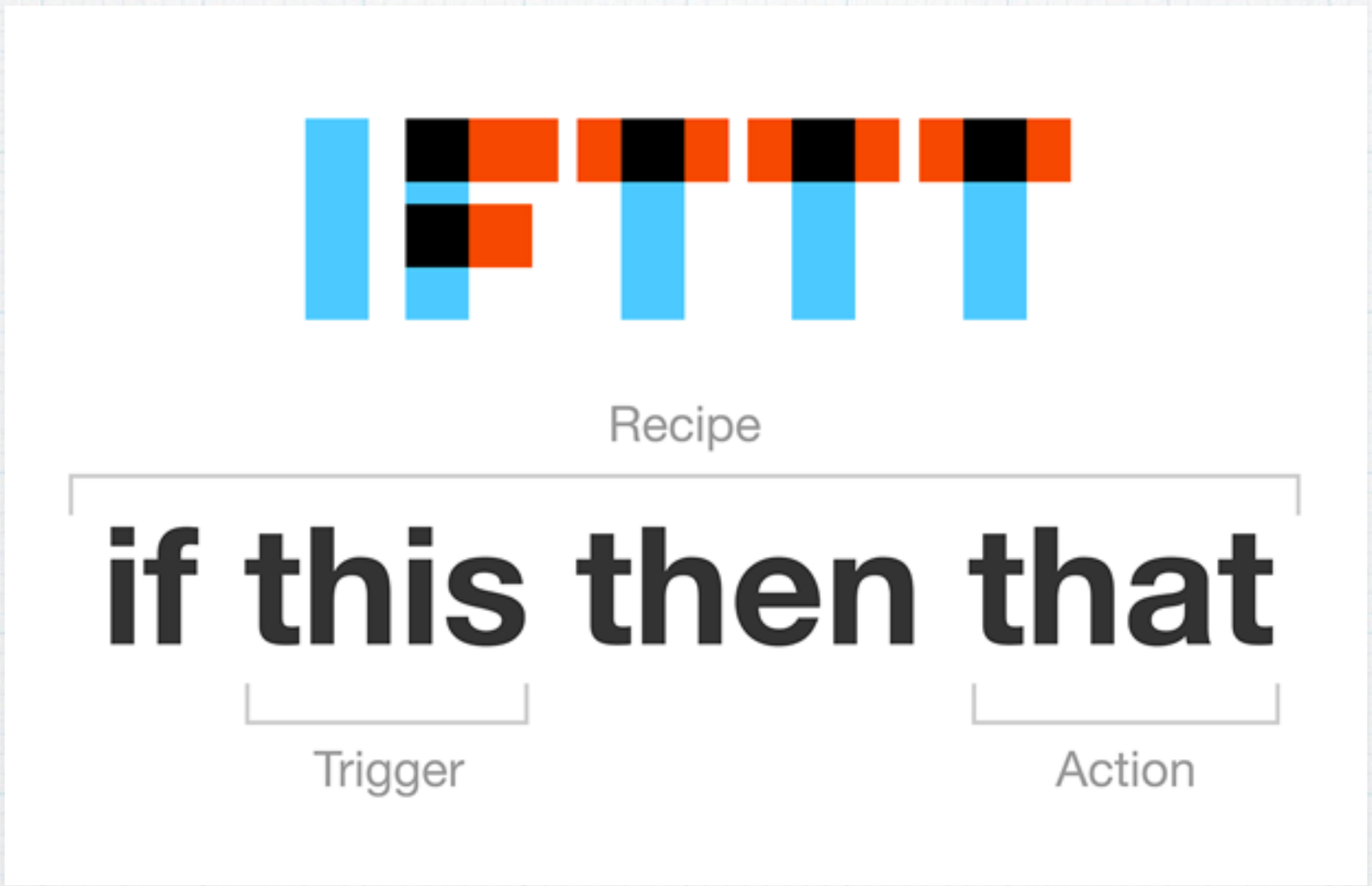
* Create a program that will create an integer variable, a function called 'showRainbow' and publish an event.
    * Pressing button 2 should increment the integer variable.
    * Pressing button 3 should publish event.
* Suggestions
    * When a button is pressed, flash an LED to indicate you received the button press

Session 4
IFTTT



https://ifttt.com

Session 4
IFTTT

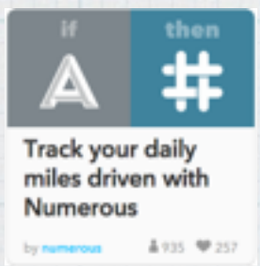


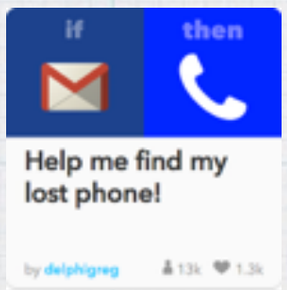
Recipe Trigger Action

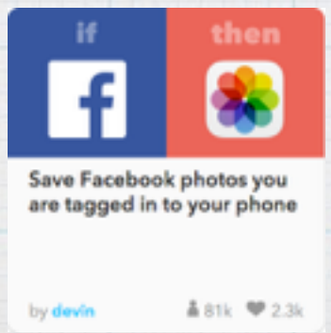
IFTTT lets you create connections between Channels

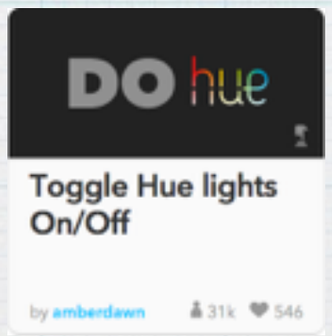
Session 4
IFTTT Channels

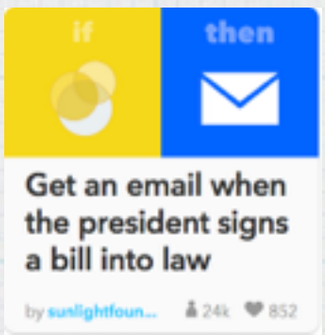
* Creates connections between Channels
* about 245 defined channels
* Most of the channels are pre-defined
* https://ifttt.com/channel



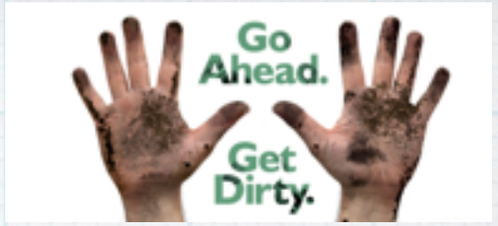




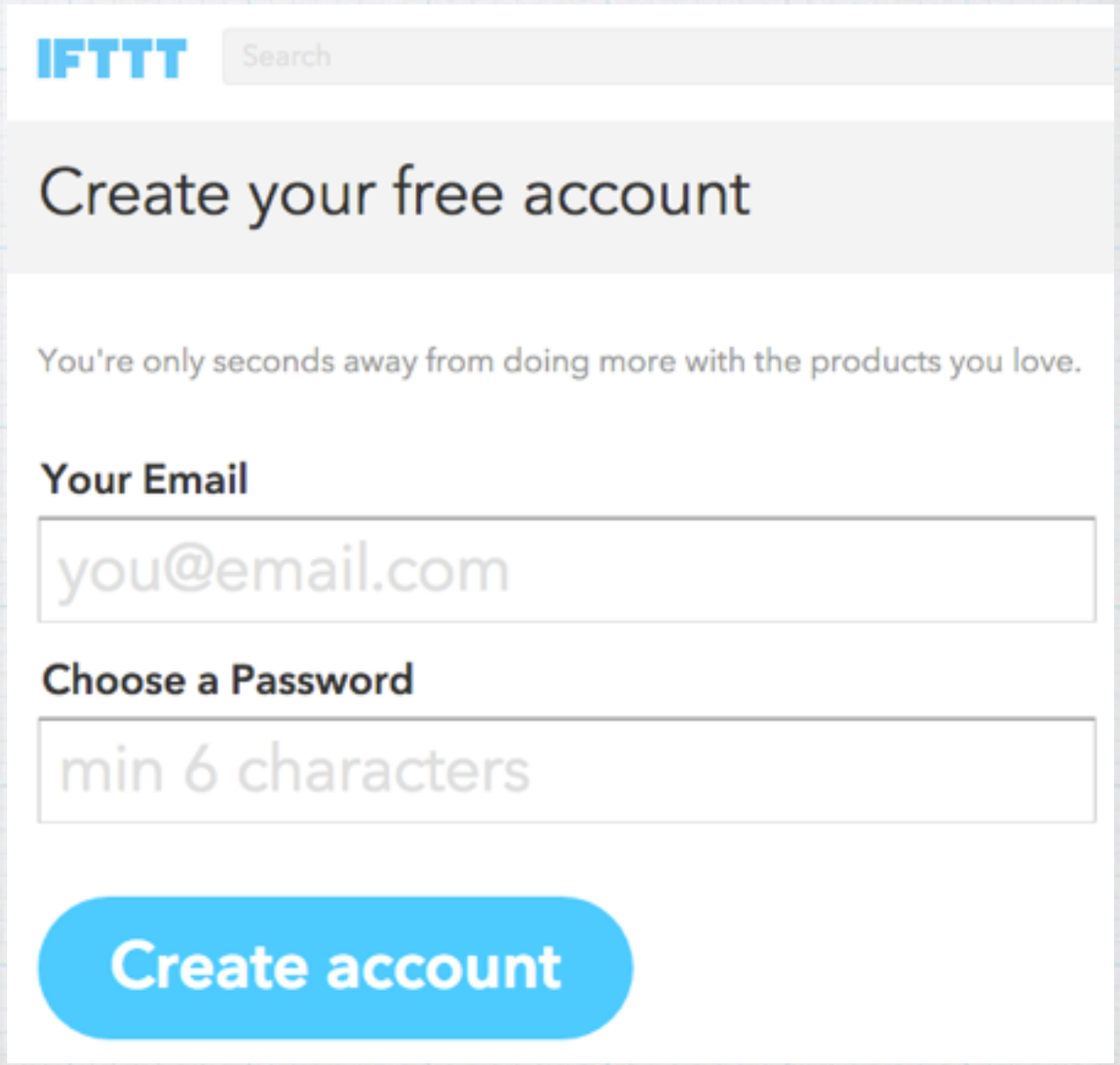


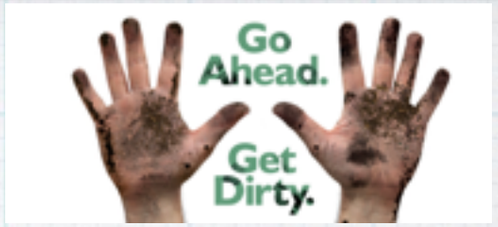


Session 4
IFTTT



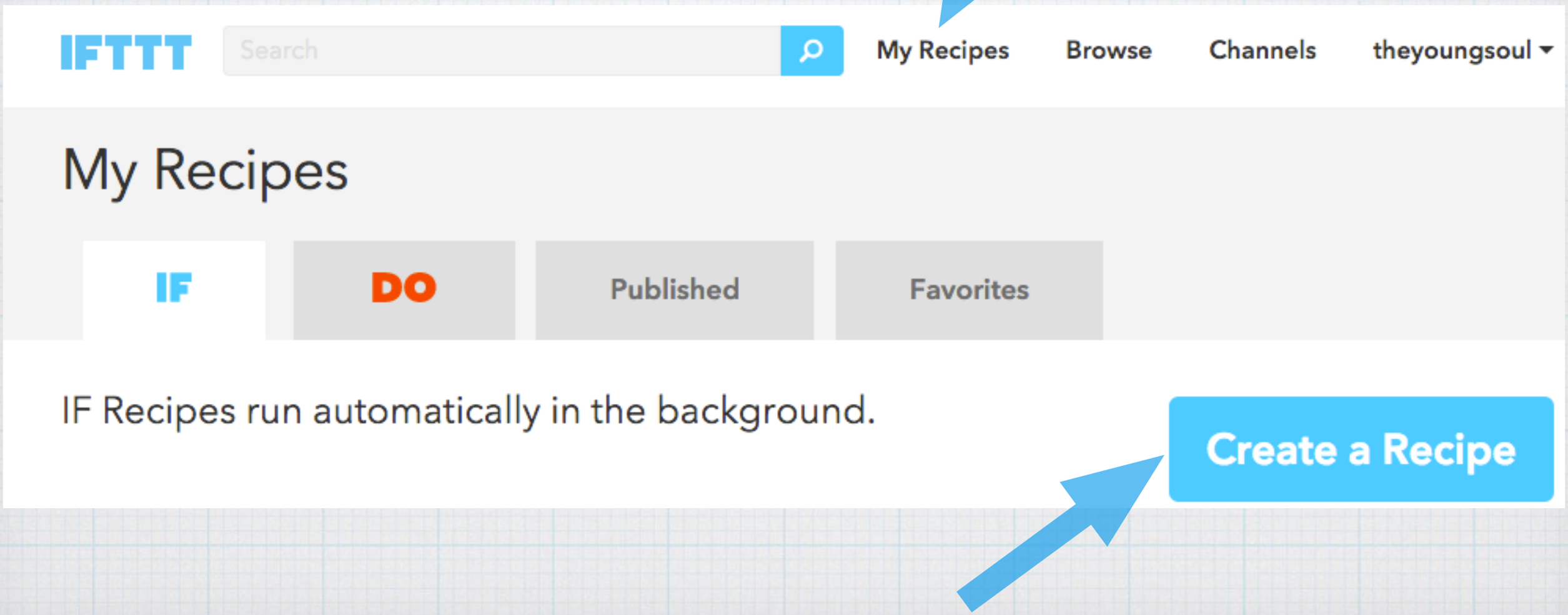
* Open browser and go to:
  * http://ifttt.com
* Create and 'Sign Up'

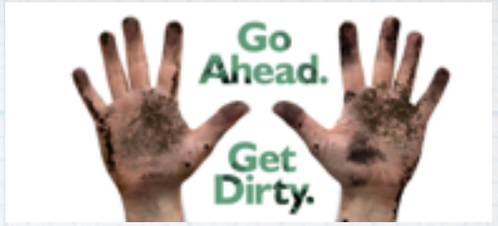




Session 4
IFTTT - Create a Recipe

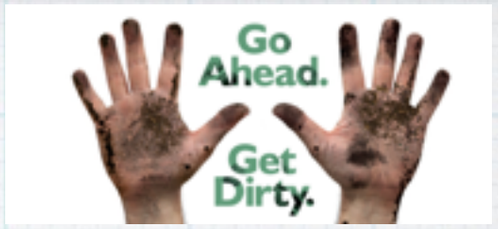
* Select 'My Recipes'
* 'Create a Recipe'





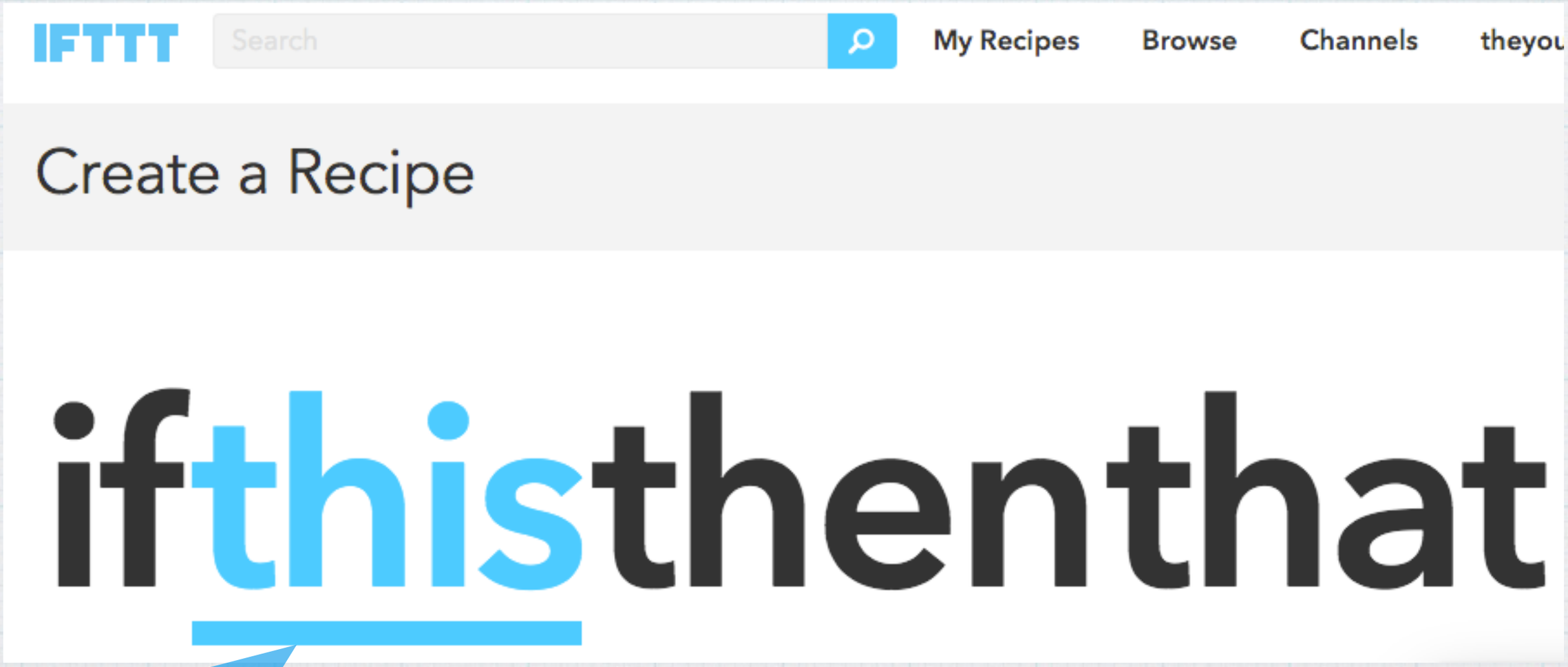
Session 4
Create a Recipe

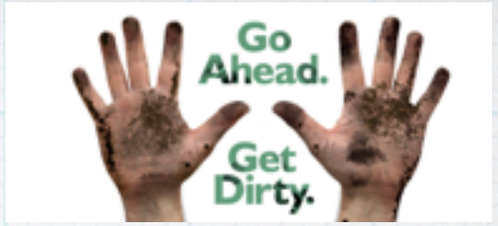
* Create a recipe to listen for one of the button events, and when we press the button send an email.



Session 4
IFTTT - Create a Recipe

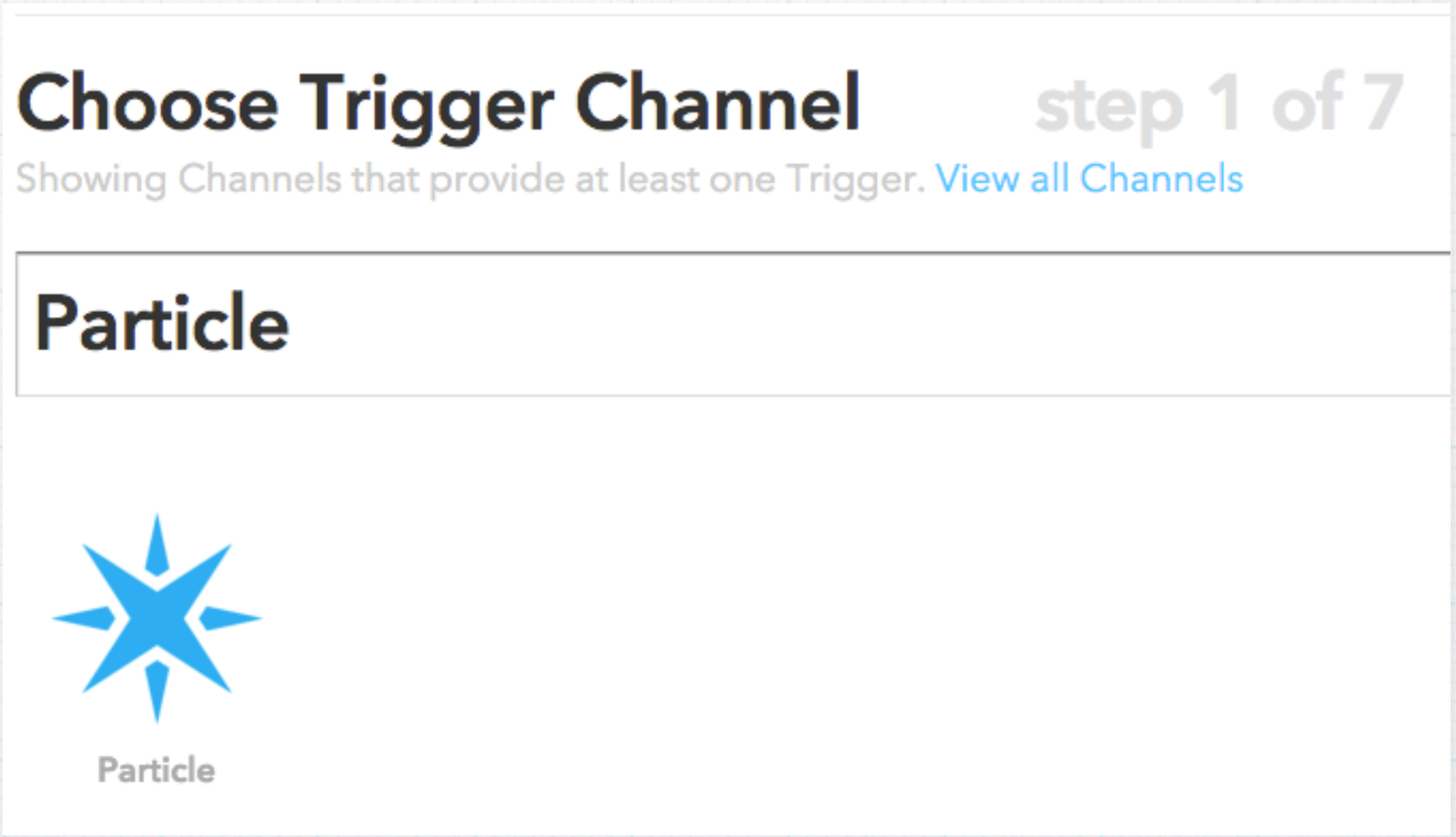
* Select 'this'



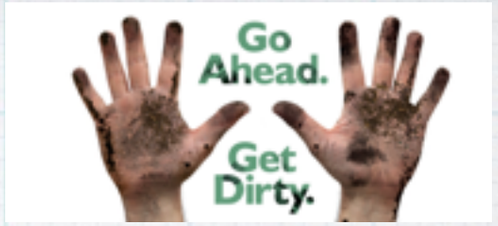


Session 4
IFTTT - Create a Recipe

* Choose Trigger Channel



Log into your particle.io Account



Session 4
IFTTT - Create a Recipe

* Choose 'New event published'

Choose a Trigger

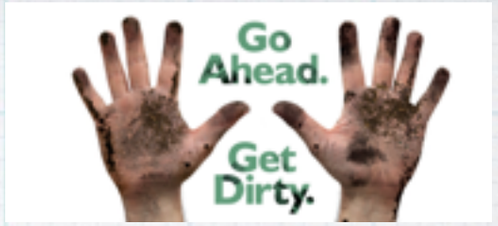
step 2 of 7

New event published
This Trigger fires when an interesting event comes from a particular device. Send events using Particle.publish.

Monitor a variable
This Trigger fires when a value on your Particle device changes to something interesting. Include particle.variable in your Particle code.

Monitor a function result
This Trigger checks a function device to see if something inte is happening.

Monitor your device status
This Trigger fires when your device changes states (i.e. when it goes online or offline.) Useful for detecting the power is out, when the internet is down, or when your Particle device sleeps much of the time.



Session 4
IFTTT - Create a Recipe

Complete Trigger Fields

step 3 of 7

New event published

If (Event Name)

button1

Fill in your published event name; ex: monitoring a washing machine? Event Name = Wash_Status

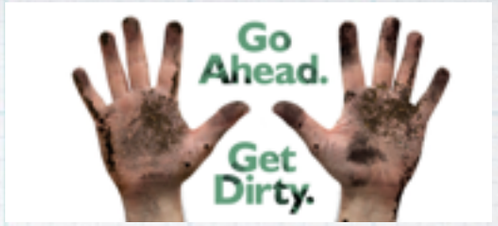
is (Event Contents)

The contents of the published event, "Data"; ex: monitoring a washing machine? Event Contents = Done

Device Name or ID

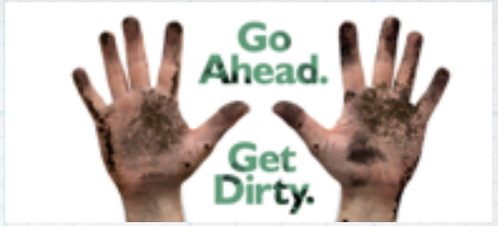
internet_button1

Create Trigger



Session 4
IFTTT - Create a Recipe

* Select 'that'



Session 4
IFTTT - Create a Recipe

* Choose Action Channel

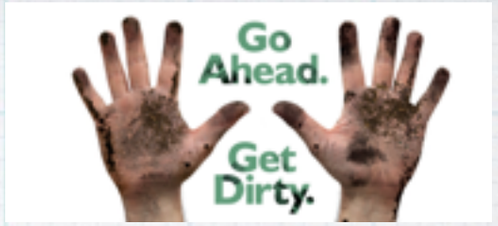
Choose Action Channel

step 4 of 7

Showing Channels that provide at least one Action. View all Channels

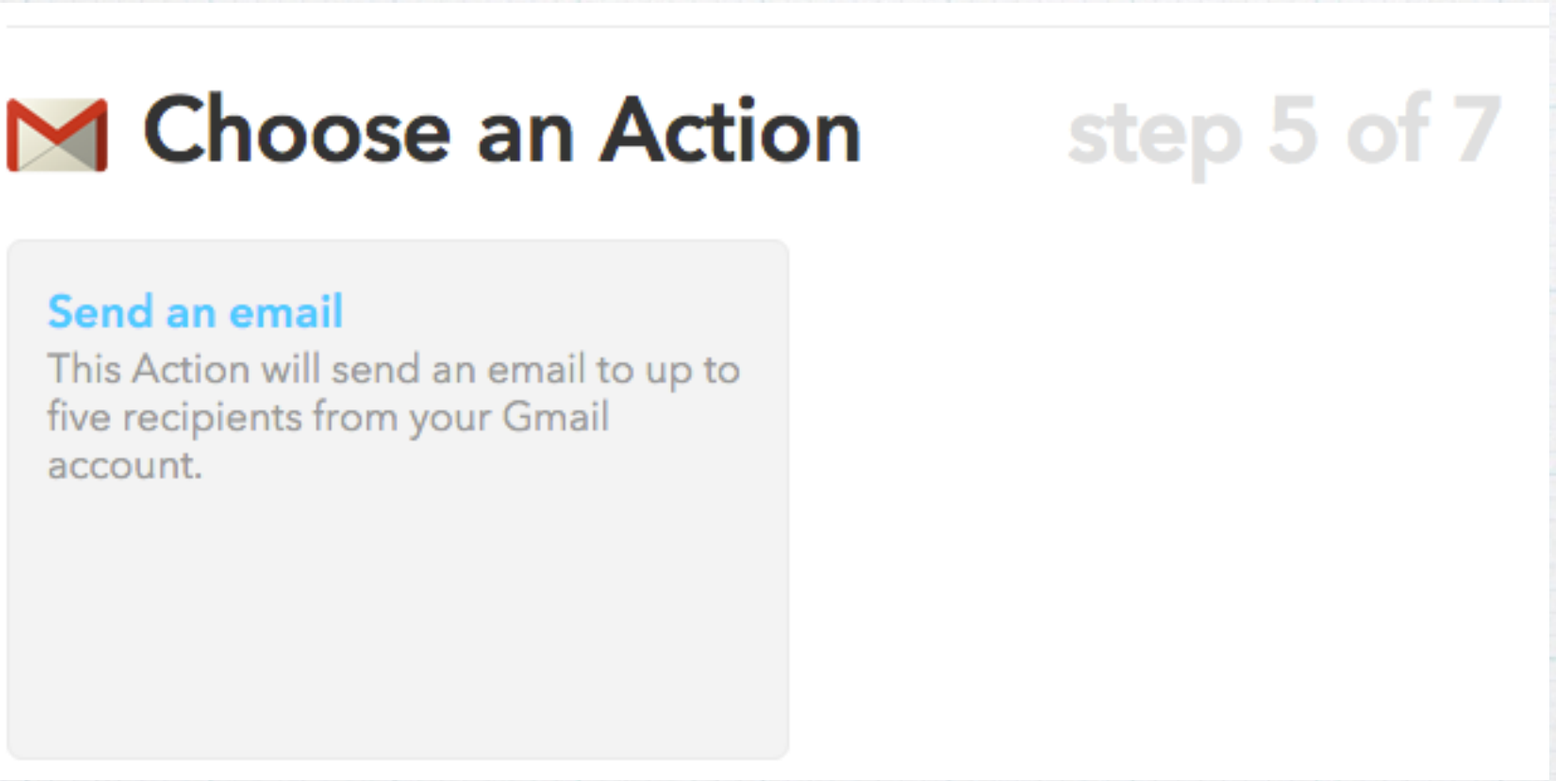
gmail

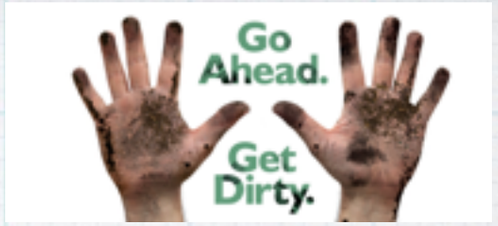
Gmail



Session 4
IFTTT - Create a Recipe

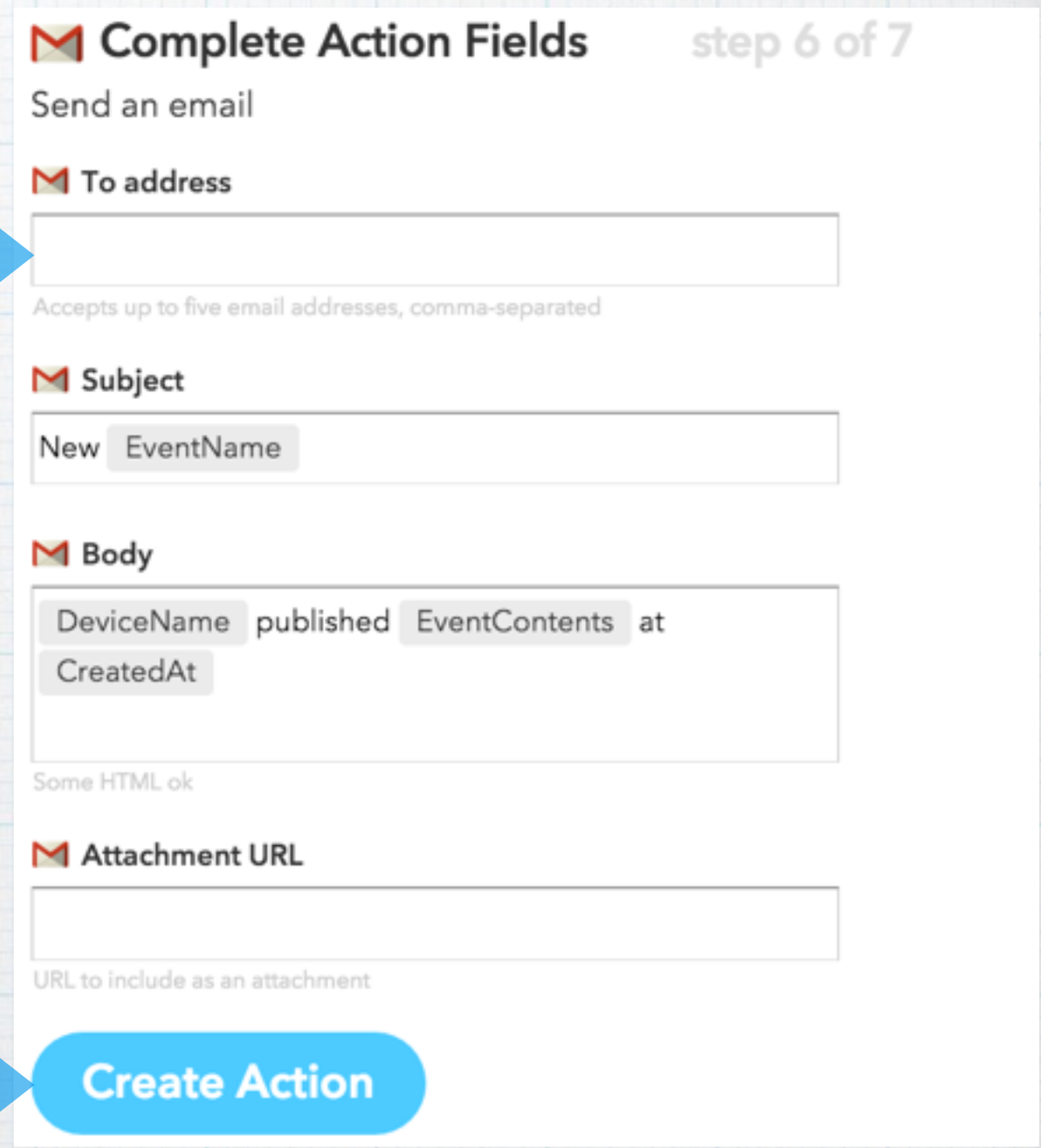
* Select 'Send an email'



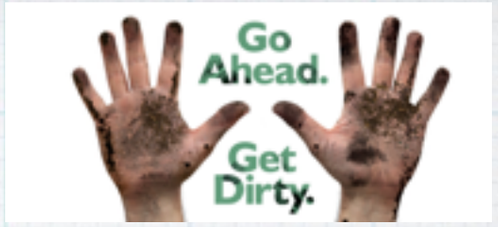


Session 4
IFTTT - Create a Recipe

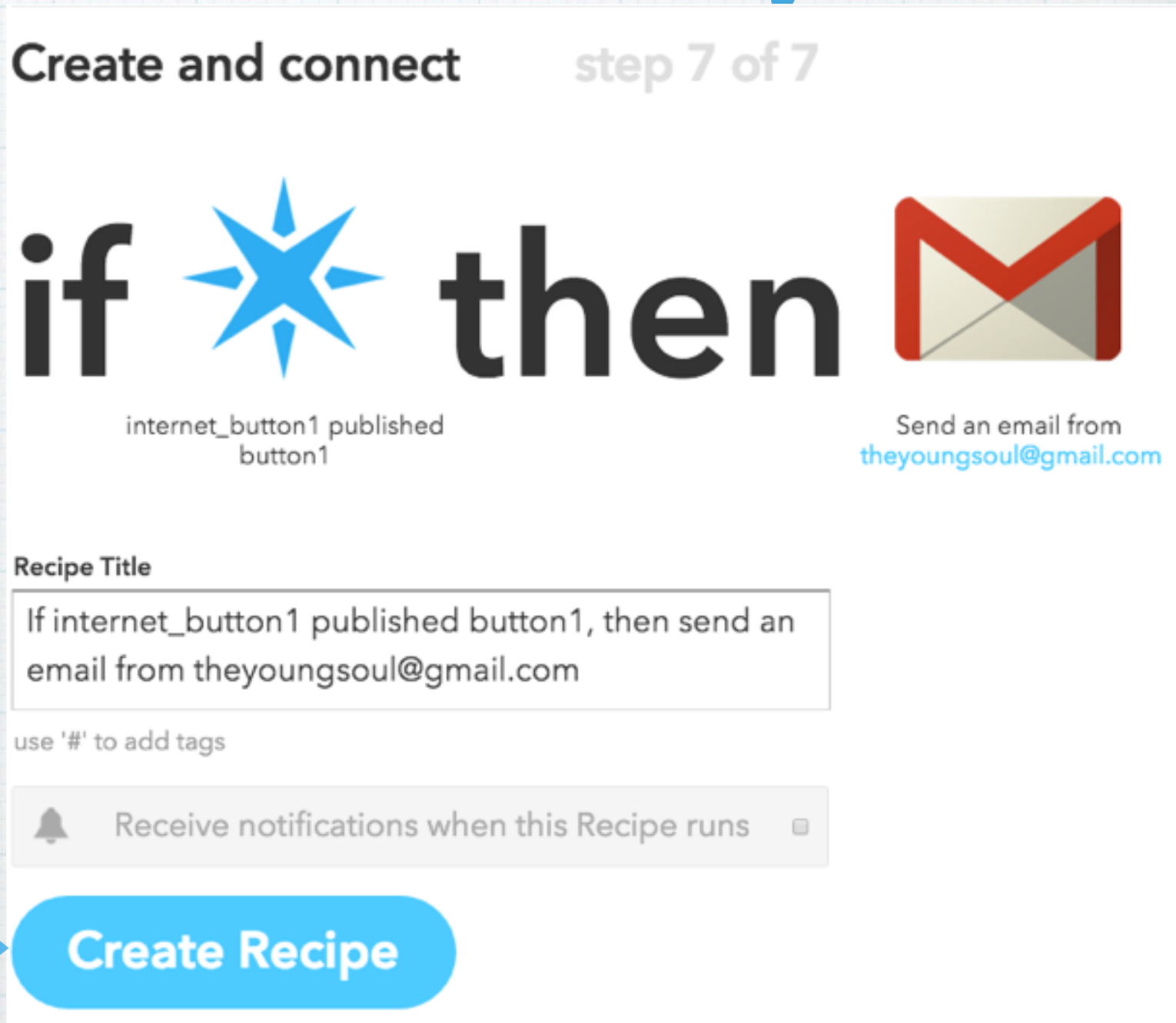
* Complete Action Fields



Session 4
IFTTT - Create a Recipe



* Create Recipe



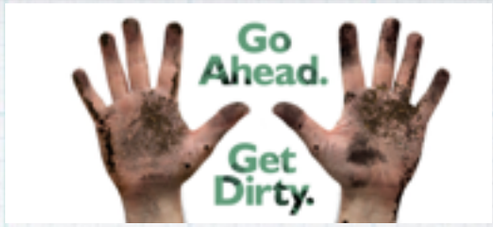
β

Session 5
DO Button

* Mobile application that creates a soft/ virtual button to access IFTTT channels

* Create a DO button to publish the 'doRainbow' event that the InternetButton is listen for

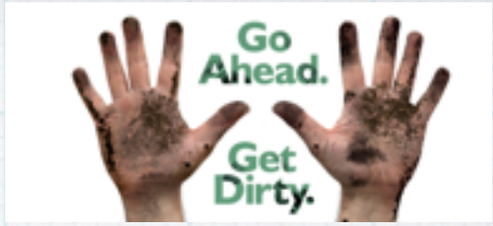
* Create a DO button to call a function that the InternetButton exposes.



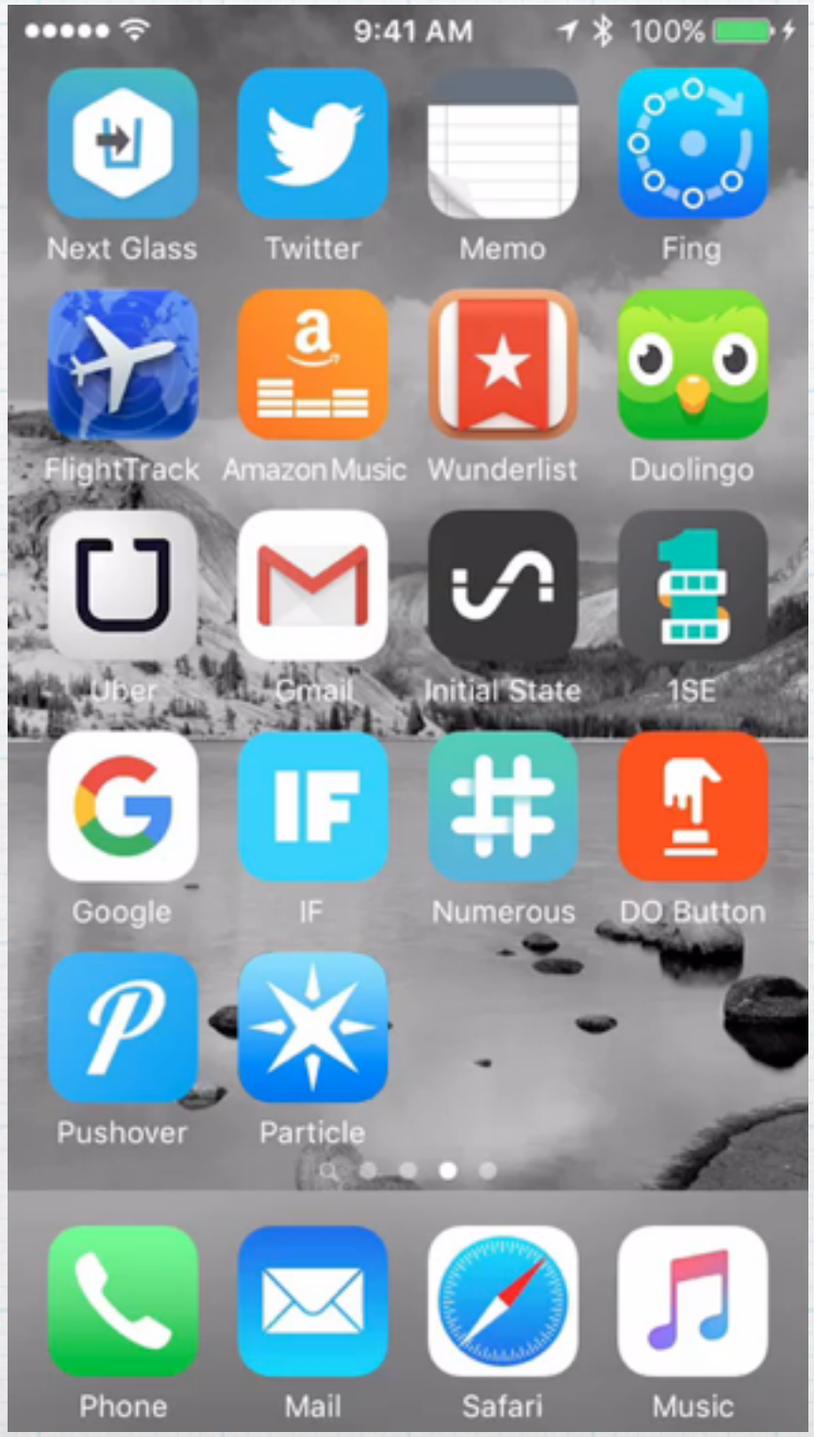
Session 5
DO Button

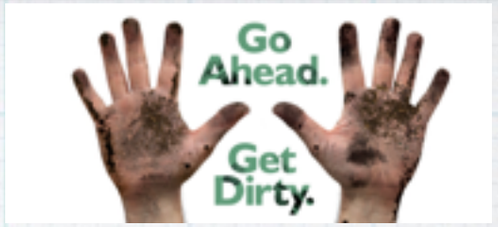
* Download the 'DO Button' application

* 'DO Button' recipes are created on the mobile device only
  * You cannot create them via the web
* Launch the 'DO Button' app
  * Log into your IFTTT account

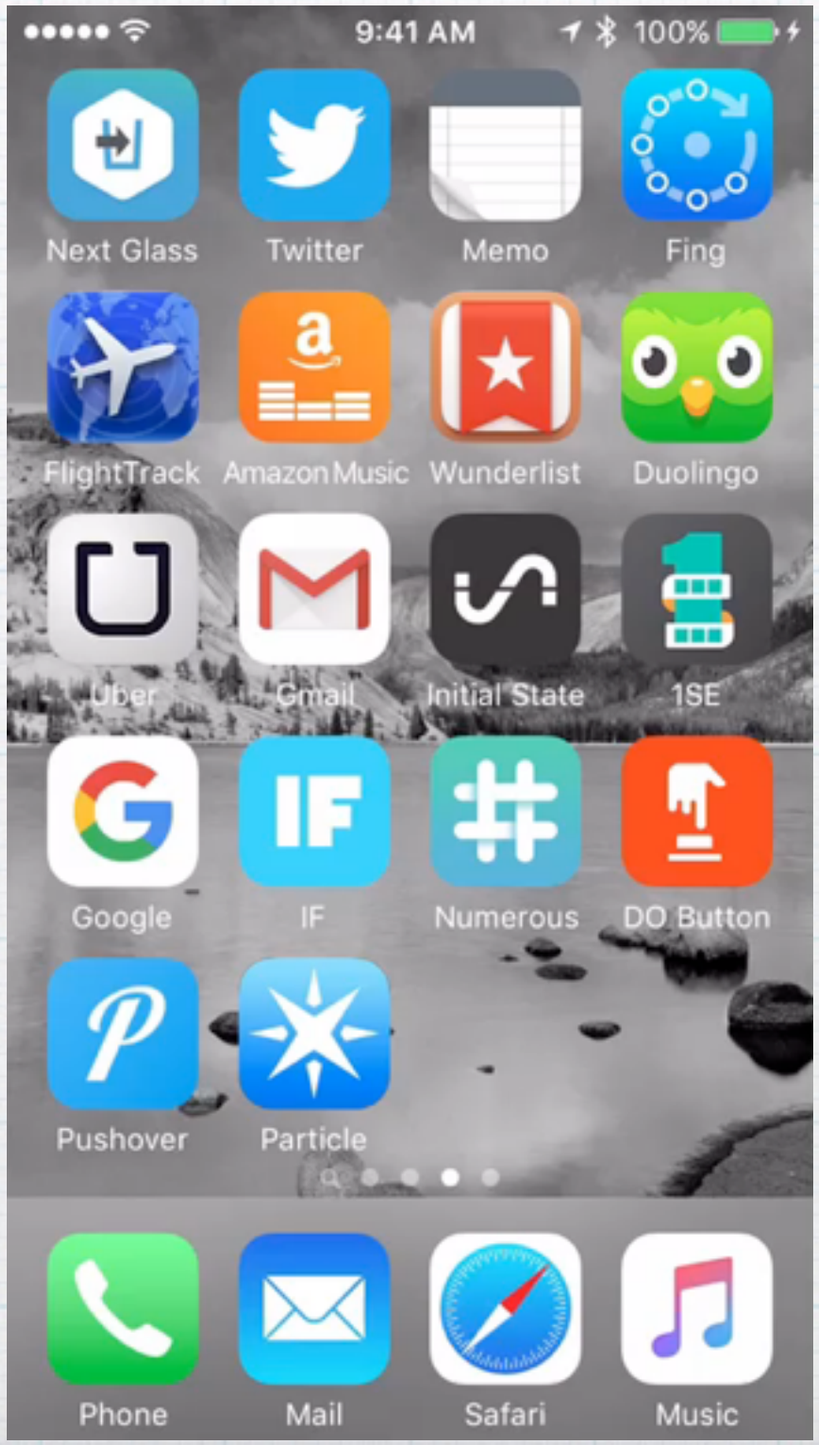


Session 5
DO Button - Events





Session 5
DO Button - Functions



Section 6
Blynk App

* www.blynk.cc
* Include blynk library
* In setup
  * Blynk.begin(auto key)
* In loop
  * Blynk.run

Section 6
Blynk App

* Write an app to turn LED widget on when InternetButton is pressed.
* Many controls and dashboard widgets
  * we are just focusing on the LED widget
  * WidgetLED blynkLed1(V1);
  * V1 - Blynk Virtual Input
  * blynkLed1.on()
  * blynkLed1.off()