

Publish/Subscribe and IFTTT

44-440/640-IoT

Objectives

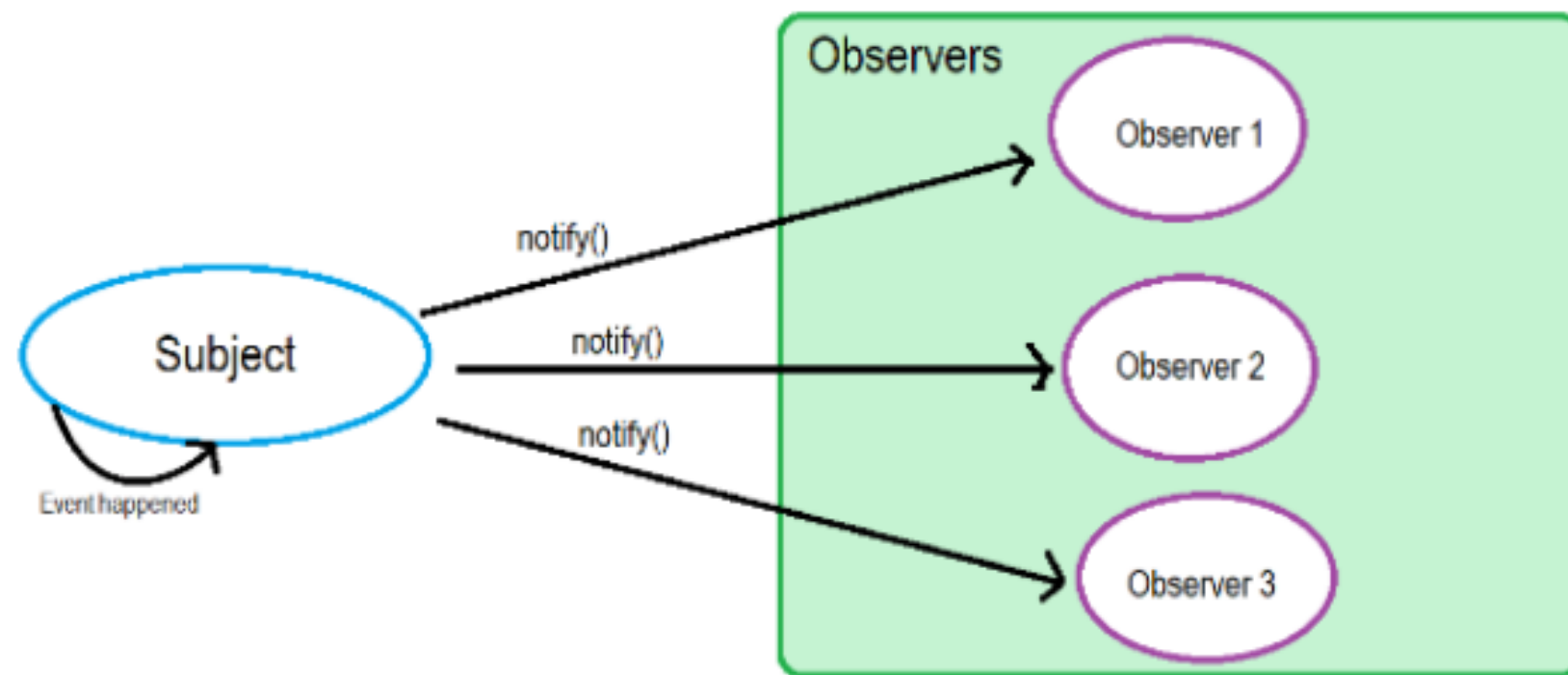
- Students will be able to:
 - describe the publish and subscribe design pattern and how it is implemented in Particle
 - describe the purpose and function of IFTTT
 - understand how to access variables and invoke functions using IFTTT
 - create novel applets using IFTTT
 - use IFTTT with events

Executive Summary:

IoT allows devices to talk to one another. In this keynote we take our first tentative steps to make that happen.

The Observer Pattern

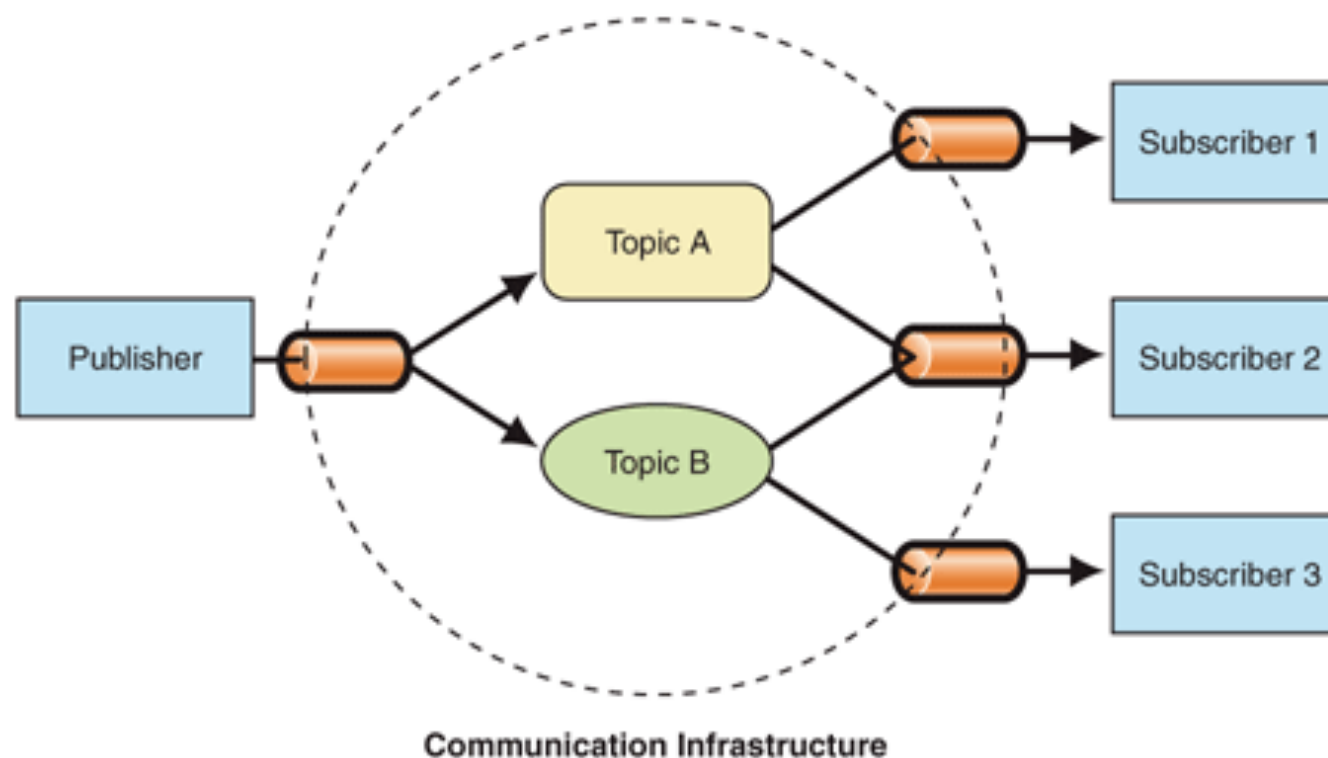
- In an observer pattern, the subject keeps track of its observers, and notifies them when any state changes, by invoking an appropriate method



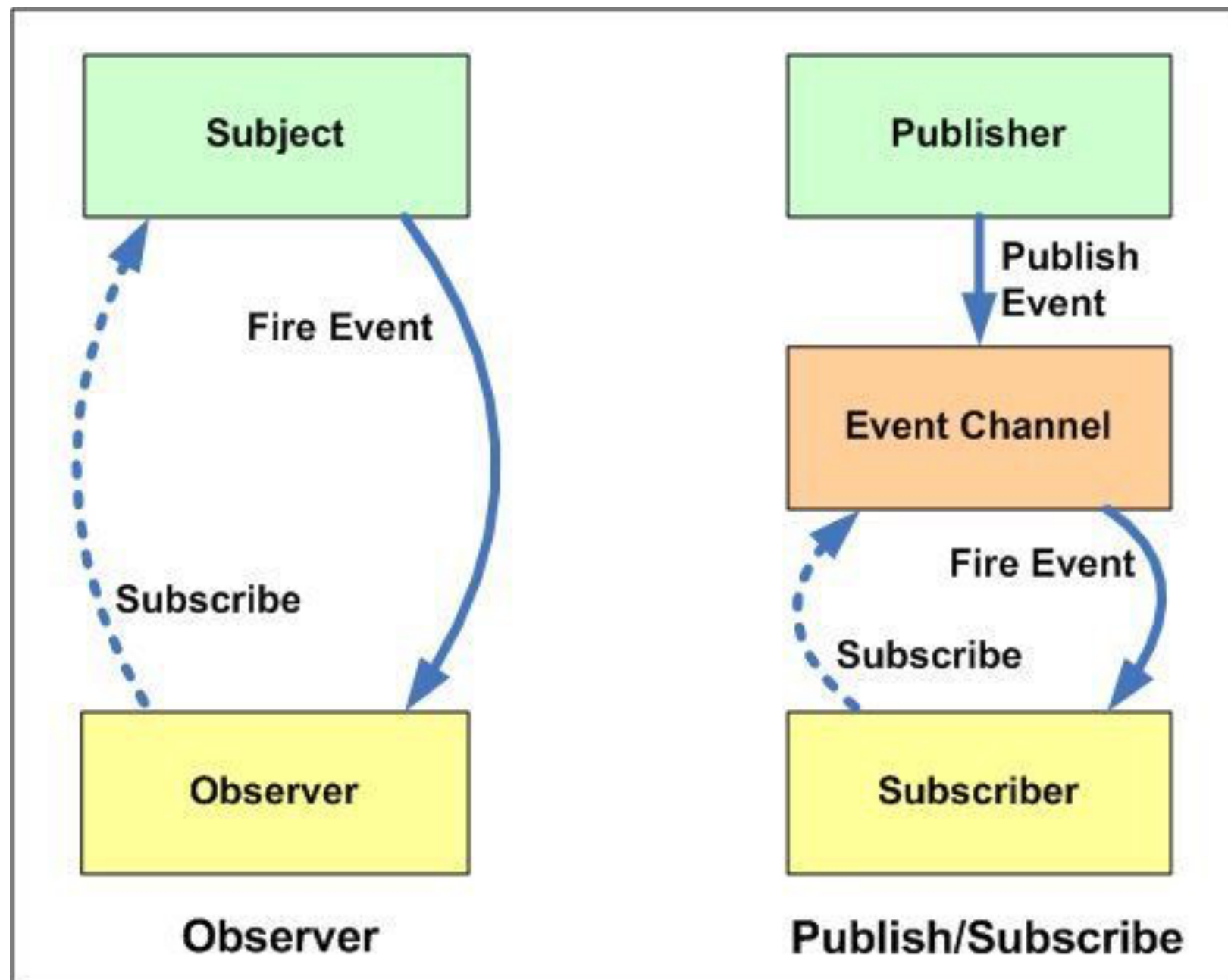
Observer design pattern (A bird's eye view :P)

Publish/Subscribe Pattern

- In this pattern, there is an intermediary between the subject (now called a publisher) and the observer (now called a subscriber)
- The publisher publishes an event; the subscribers learn about it; but the publisher doesn't know who its subscribers are, and vice-versa
- It is affectionately known as **pub/sub**
- The intermediary is known as a broker, message broker or event bus

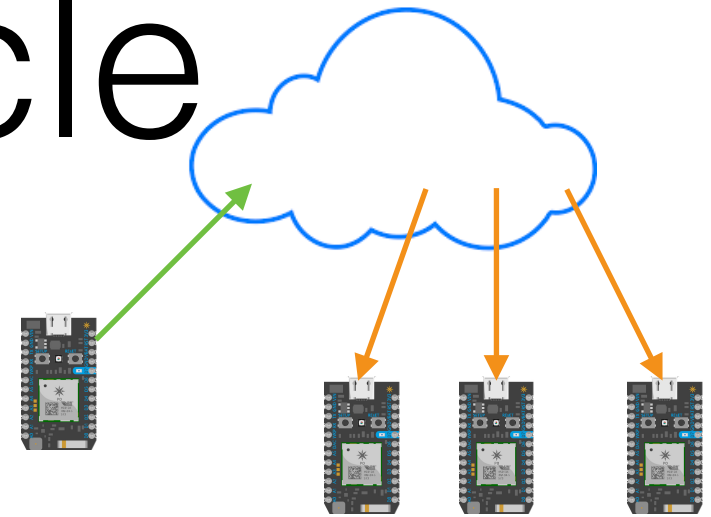


To Summarize ...



Events in Particle

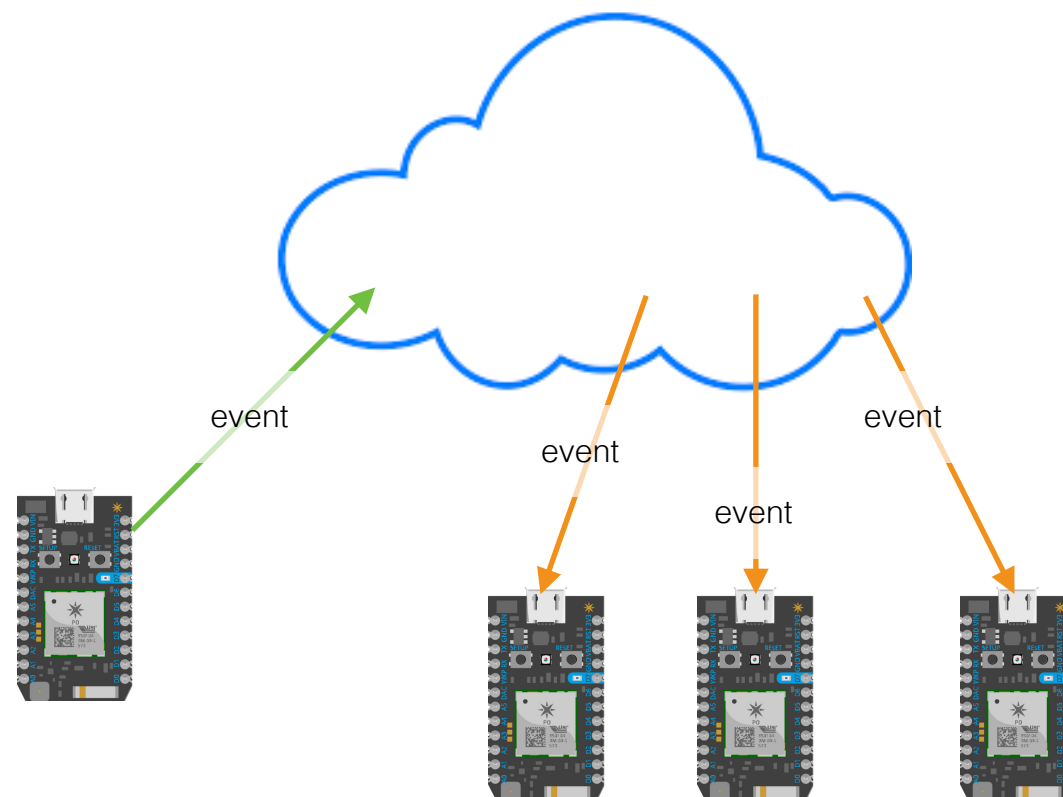
- In Particle, an event has 4 properties:
 - **name** (1-63 characters)
 - **data** (≤ 255 bytes)
 - **time to live** (0-16777215, but 60 is the *immutable* default time)
 - **access level** (public: *anyone* can subscribe. private: only the device owner can subscribe)



The data is entirely implementation/context dependent.

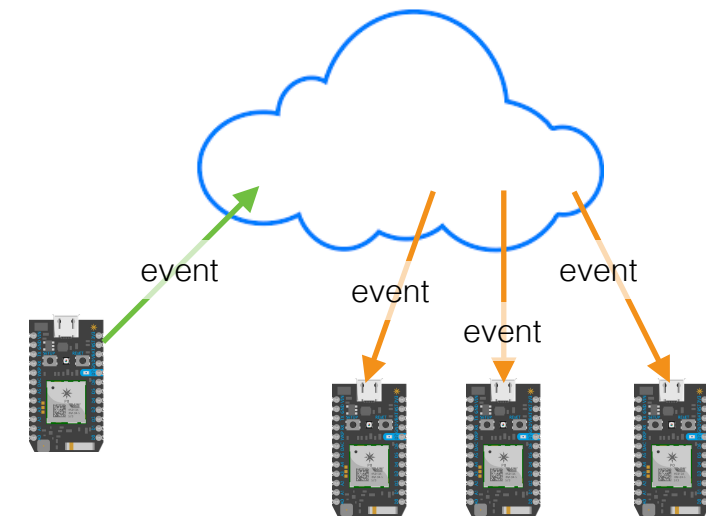
Events in Particle

- Events can be
 - published to the cloud, via **Particle.publish()**
 - subscribed to by other devices, via **Particle.subscribe()**



Events in Particle

- The ability for multiple Photons to receive events raises all *sorts* of interesting possibilities ...
- Events can be published at about the rate of 1 event / second. Don't flood the system!!!
- Use the [Particle console](#), under devices, to see events: (click on a log entry for raw data)



Particle.publish()

API:

- Particle.publish(const char * eventName, PublishFlags flags);
- Particle.publish(const char * eventName, const char * data, PublishFlags flags);
- Particle.publish(const char * eventName, const char * data, int ttl, PublishFlags flags);

Examples:

- Particle.publish("Low On Fuel", PRIVATE);
- Particle.publish("Garage Door Event", "Closed", PRIVATE);
- Particle.publish("Battery Charge", "2.8 V", 60, PUBLIC);

Anywhere const char * appears, you can also use String, it gets converted: see this [discussion](#).

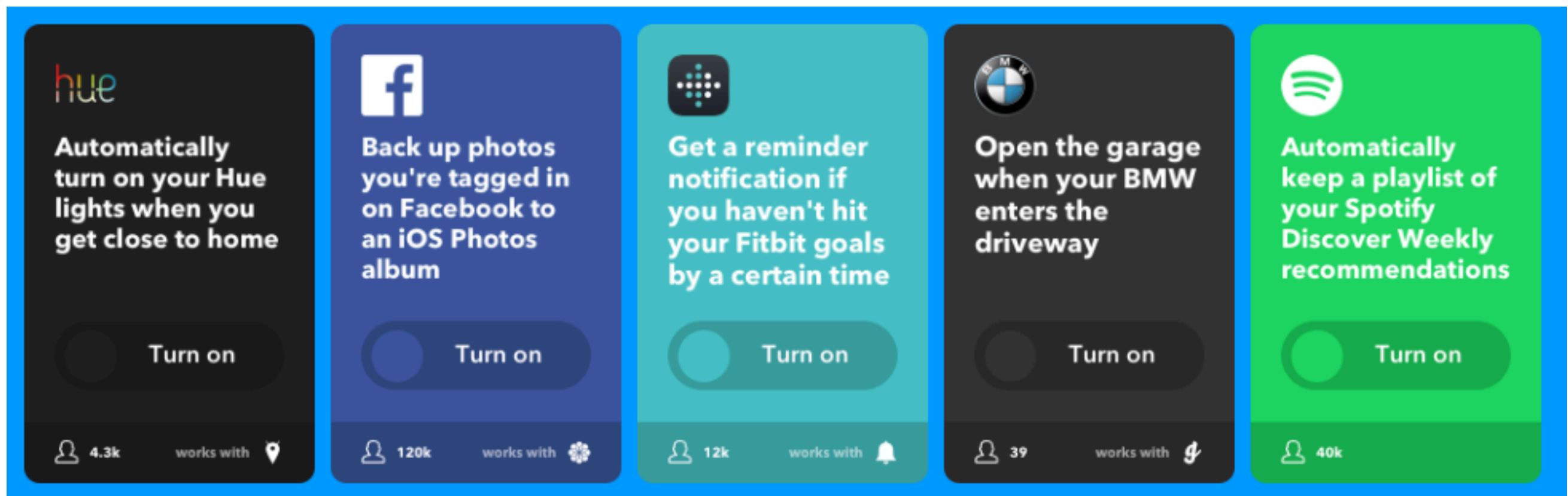
Particle.subscribe() Details

- Particle.subscribe(const char * eventName, EventHandler handler)
- Particle.subscribe(const char * eventName, EventHandler handler, Spark_Subscription_Scope_TypeDef scope=ALL_DEVICES)
 - This will subscribe you to any event that **starts** with eventName, e.g., if you subscribe to, say "gro", you will be notified when any event that starts with "gro" ("grok", "groan", "grocer", etc.)
 - The **event handler** is any void function with two C Strings (this is *different* than a Particle function: an int function with 1 C String). This function will be triggered when the event is published
 - You can subscribe to up to 4 event handlers

```
typedef void (*EventHandler)(const char* name, const char* data);  
  
typedef enum  
{  
    MY_DEVICES,  
    ALL_DEVICES  
} Spark_Subscription_Scope_TypeDef;
```

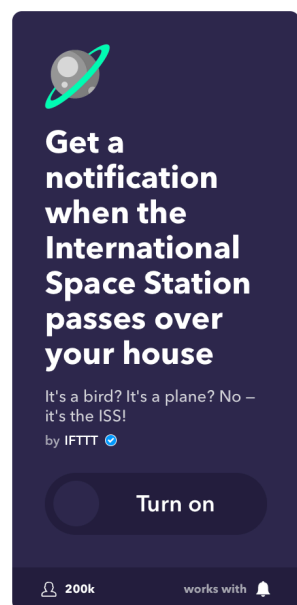
IFTTT Concepts

- IFTTT (pronounced like "Gift" without the G) stands for If This Then That. It connects web services together to create new functionality.

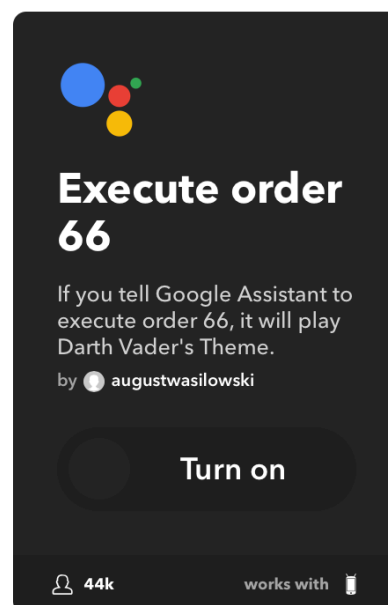


IFTTT Concepts

- IFTTT is based on **applets**. An applet consists of a **trigger** and **action**.
- When the trigger is fired, the action takes place
- Data, or **ingredients**, available in the trigger, can be passed to the action.
- Here are some possibilities: take a few moments to find others ...



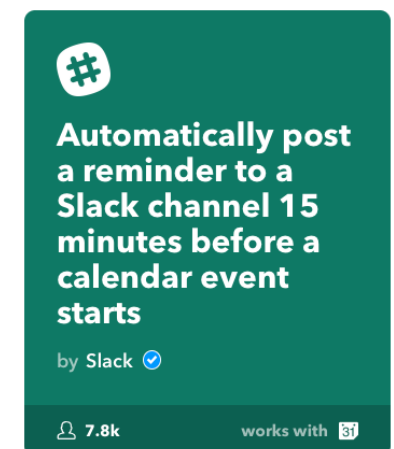
Trigger: The ISS is over your house
Action: Get a text message



Trigger: OK, Google, execute order 66
Action: Google plays Darth Vader's Theme



Trigger: Push a button
Action: Get a phone call



Trigger: 5 minutes until a Google Cal event
Action: Post a reminder to a Slack channel

Particle Triggers and Actions

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 on your device to see if something interesting 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.

Triggers

Publish an event

This Action publishes an event back to your Device(s), which you can catch with `particle.subscribe`.

Call a function

This Action will call a function on one of your Devices, triggering an action in the physical world.

Actions

In constructing an Applet, you will likely use a Particle Trigger or Action, but not both together.

IFTTT ICE: Preliminaries

1. Bring to class a completed circuit that contains:

1. a TMP 36
2. a button
3. an LED

2. Flash firmware that has the following capabilities:

1. when the button is pressed, it stores the temperature in a Particle variable, **temperature**, and writes it out to the console
2. Particle functions, **turnOn()** and **turnOff()** that turn the LED on and off, respectively.

3. Get an account on [IFTTT.com](https://ifttt.com) (enable Particle, Phone, Email and Twitter integration)

IFTTT ICE: New Event Published

1. Modify your firmware so that when the button is pushed, it publishes an event with two arguments:

1. "pushed" -- the event name
2. String(temperature) -- the data

2. Create an applet in IFTTT that will, when pushed is received, post a tweet that indicates the temperature

New event published

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

IFTTT ICE: New Event Published - Solution

```
double temperature = 0.0;
const int yellowLED = D0;
const int switchy = A0;
const int temperaturePin = A1;

void setup() {
  pinMode(yellowLED, OUTPUT);
  pinMode(switchy, INPUT_PULLUP);
  pinMode(temperaturePin, AN_INPUT);

  Particle.variable("temperature", temperature);
  Particle.function("turnOn", turnOn);
  Particle.function("turnOff", turnOff);
}

void loop() {
  if(digitalRead(switchy)==LOW){
    double voltage = analogRead(temperaturePin)/4095.0 * 3.3;
    temperature = 104.7*voltage - 53.1;
    Particle.publish("pushed", String(temperature, 2));
    delay(250);
  }
}

int turnOn(String data){
  digitalWrite(yellowLED, HIGH);
}

int turnOff(String data){
  digitalWrite(yellowLED, LOW);
}
```



New event published

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

If (Event Name)

pushed

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

trochee_wombat



An optional id for a particular device



Post a tweet

This Action will post a new tweet to your Twitter account. NOTE: Please adhere to Twitter's Rules and Terms of Service.

Tweet text

The temperature is currently
EventContents

IFTTT ICE: Monitor a Variable

- Create an applet such that when the temperature exceeds current temperature + 0.5°, it calls your phone (from 415-969-2754)
- Test it by breathing on your TMP36

Monitor a variable

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

IFTTT - ICE: Monitor a Variable - Solution

- No extra code is required, since temperature is a Particle variable

Monitor a variable

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

If (Variable Name)

temperature on "troch" ▼

The name of a `Spark.variable`; options will be pulled from your code once you run your firmware

is (Test Operation)

Greater ▼

How should the function result compare to our comparison value?

Comparison Value (Value to Test Against)

26.0

Ex: If you want your device to ping you if temperature is over 72; then 72 = Comparison Value

Call my phone

This Action will call your phone number and say a message.

Message to say

Hi! As of `CreatedAt` the `Variable` is `Value`

IFTTT ICE: Monitor Your Device Status


- Create an applet so that if your device goes offline, you get a text

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.


IFTTT ICE: Monitor Your Device Status - Solution

- It took a 1.5 minutes for this to get triggered
- Starting with DeviceOS 0.8.0, some of this functionality is available natively in Photon

 **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.


If (Device name or ID)

trochee_wombat 

The name or id of your device

is (Status of your device)

Offline 

 **Send me an SMS**

This Action will send an SMS to your mobile phone.

Message

DeviceName has just gone Status


IFTTT ICE: Call a Function

- Call a Function is an **action**, which means any trigger available through IFTTT can invoke a function on your Particle
- Write an applet so that when you send either
 - a) a Tweet with a hashtag #ILikePhoton,
 - b) an email with a subject #ILikePhoton
- `turnOn()` will be invoked
- Unsurprisingly, b) is much faster

Call a function

This Action will call a function on one of your Devices, triggering an action in the physical world.

IFTTT ICE: Call a Function - Solution


 **New tweet by you with hashtag**

This Trigger fires every time you post a new tweet with a specific hashtag.

Hashtag

#ILikePhoton

e.g. #hashtag

 **Call a function**

This Action will call a function on one of your Devices, triggering an action in the physical world.


Then call (Function Name)

turnOn on "trochee_w ▼

The name of the function you want to call;
ex: for your lighting project you may name your function LED

with input (Function Input) (optional)

"hello"


 **Send IFTTT an email tagged**

Send IFTTT an email at trigger@applet.ifttt.com with a hashtag in the subject (e.g. #IFTTT) and this Trigger fires. You can optionally add a single file attachment and IFTTT will create a public URL to the file as an Ingredient.

Tag

ILovePhoton

e.g. #hashtag

 **Call a function**

This Action will call a function on one of your Devices, triggering an action in the physical world.

Then call (Function Name)

turnOn on "trochee_w ▼

The name of the function you want to call;
ex: for your lighting project you may name your function LED

with input (Function Input) (optional)

Body

IFTTT ICE: Publish an Event

- Pending

IFTTT Polling Frequency

- IFTTT is not instantaneous. The **documentation states that** "Some applets run within seconds, while others run every 60 minutes. On average, most applets run within 15 minutes of being triggered."
- In your instructor's experience, [IFTTT.com](https://ifttt.com) under promises/over delivers: applets run *much* faster, about once a minute.

Particle via IFTTT 7:47 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:47PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Particle via IFTTT 7:47 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:46PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Particle via IFTTT 7:46 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:45PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Particle via IFTTT 7:44 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:43PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Particle via IFTTT 7:44 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:42PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Particle via IFTTT 7:44 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:44PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Particle via IFTTT 7:41 PM
magicNumber crossed our threshold! Inbox - iCloud
Good news, everyone! At September 27, 2017 at 07:41PM, invincible-dentist reported that magicNumber is now 6 If magicNumber on "invincible-dentist" is...

Exercises

- Write a function to flash the morse code equivalent of a String (e.g., HELP ==>-... .--.) on an LED, then invoke the function through IFTTT if an email is received with the subject line "HELP"

Resources

- IFTTT.com
- Monk, Simon. Make: Getting Started with the Photon.
- `firmware/wiring/inc/spark_wiring_cloud.h`
- <https://hackernoon.com/observer-vs-pub-sub-pattern-50d3b27f838c>