



November 9-11, 2021
www.dotnetconf.net

.NET Conf

Discover the world of .NET

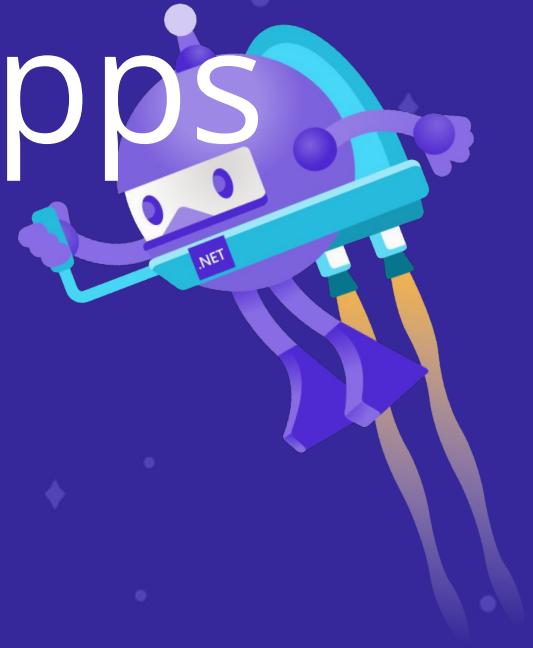


Raspberry-Pi hand sanitizer controlled by Mobile Apps

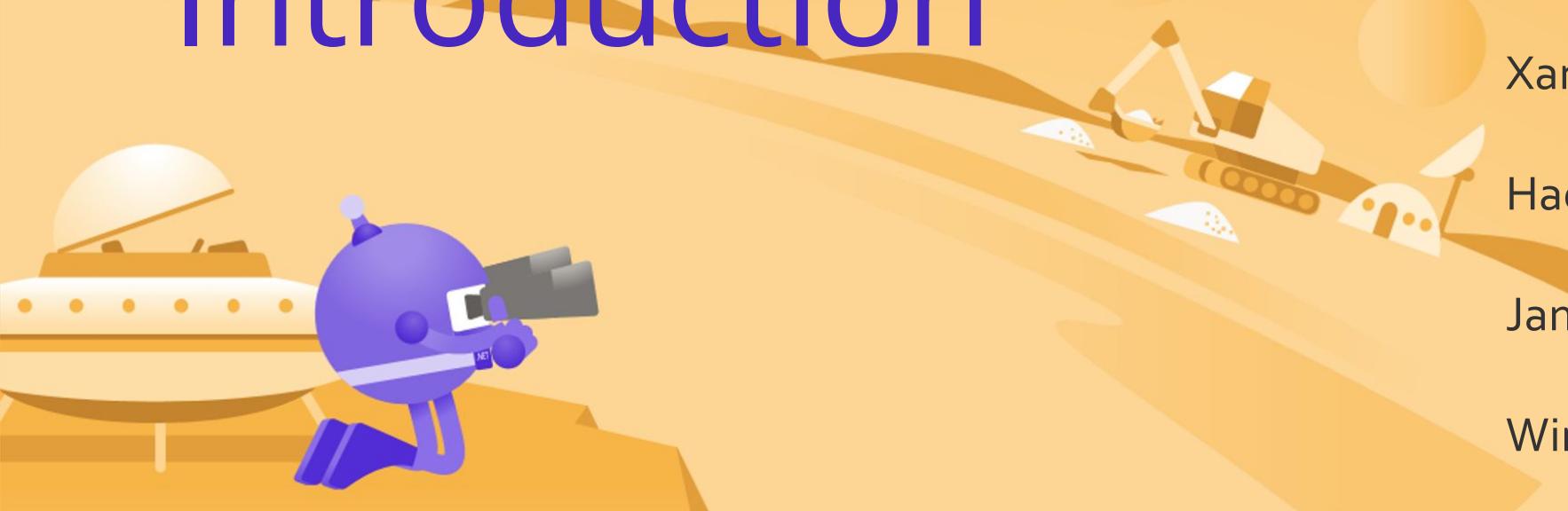
Saamer Mansoor

Founder, The First Prototype

11/8/2021



Introduction



Me



Omar A.



Arduino



EpicU



C#



Exigo



Xamarin



Hackathon



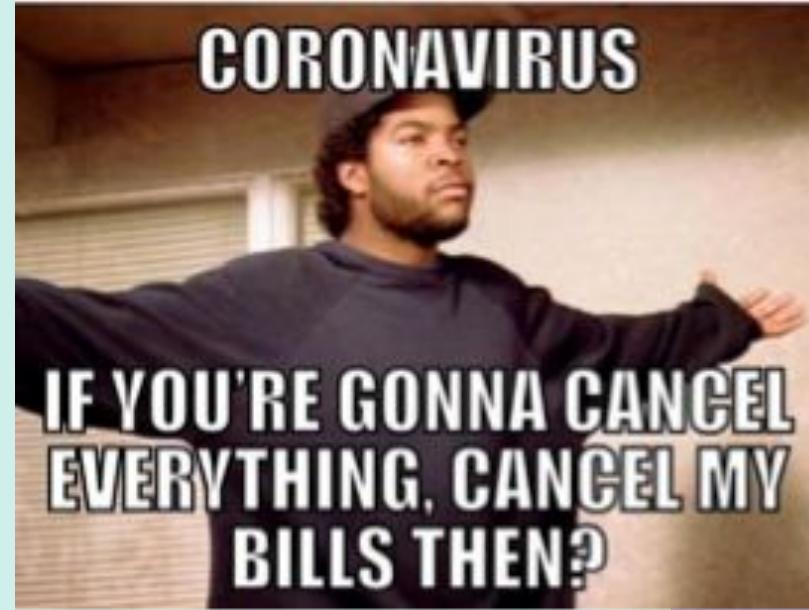
James Q



Windsor



Inspiration



DAYLIGHT SAVINGS IS HERE



CLOCK IN MY CAR IS CORRECT AGAIN



Xamarin Basics

Part 1

No Mobile Development experience? No problem!



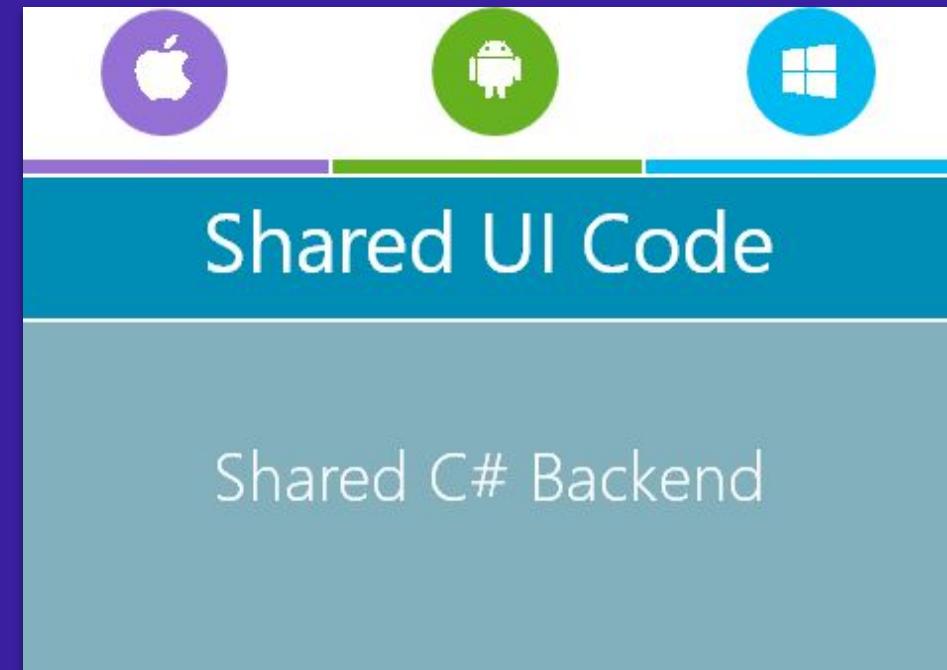
Hybrid vs Cross Platform

- Many definitions, not much consensus
- Hybrid apps are developed using more than one language/technology.
- Does not always mean it will be cross-platform.
- One can use both Objective-c and Swift in single iOS app, both languages talk with each other by using a bridge.
- Commonly accepted as Hybrid:
Apache Cordova or PhoneGap



Basic electronic components

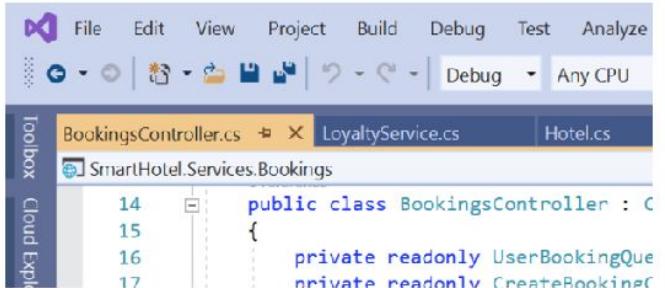
- **Xamarin Native** - “Traditional Xamarin”
 - Anything you can do in Objective-C/Swift for iOS, or Java/Kotlin in Android possible in C#
 - Higher learning curve
- **Xamarin Forms** - “Instant App Maker”
 - Even the UI can be edited just once, for the most part.
 - Basic C# skills suffice for maintenance.



Getting Started

Visual Studio

Full-featured IDE to code, debug, test, and deploy to any platform. [Learn more](#)

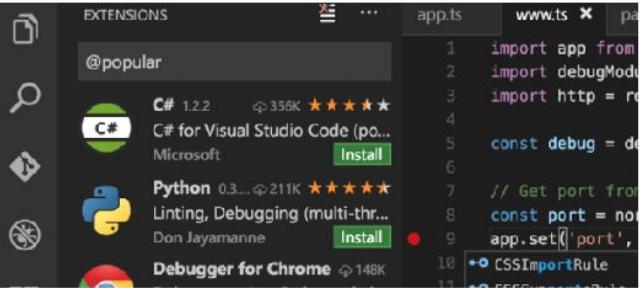


File Edit View Project Build Debug Test Analyze
Debug Any CPU
Toolbox Cloud Explorer
BookingsController.cs LoyaltyService.cs Hotel.cs
SmartHotel.Services.Bookings
14 public class BookingsController : C
15 {
16 private readonly UserBookingQue
17 private readonly CreateBookingC

[Download Visual Studio](#)

Visual Studio Code

Editing and debugging on any OS. [Learn more](#)



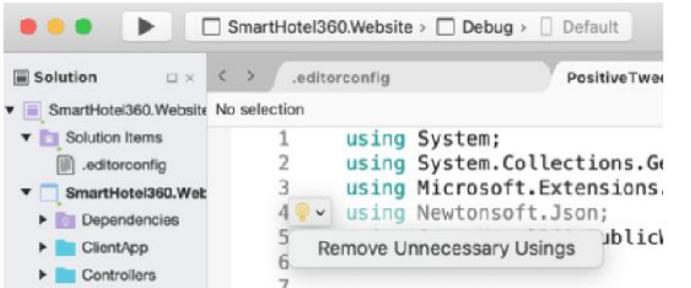
EXTENSIONS
@popular
C# 1.2.2 356K Microsoft Install
C# for Visual Studio Code (po...
Python 0.3... 211K Don Jayamanne Install
Linting, Debugging (multi-thr...
Debugger for Chrome 148K
app.ts
www.ts
1 import app from
2 import debugModule
3 import http = require('http');
4 const debug = d...
5 // Get port from
6 const port = no...
7 app.set('port',
8 process.env.P...
9 port || 3001);
10 app.listen(p...
11 () => console.log(`...`))
12 }
13
14 module.exports = app;

By using Visual Studio Code you agree to its [license](#) and [privacy statement](#)

[Download Visual Studio Code](#)

Visual Studio for Mac

Develop apps and games for iOS, Android, and web using .NET. [Learn more](#)

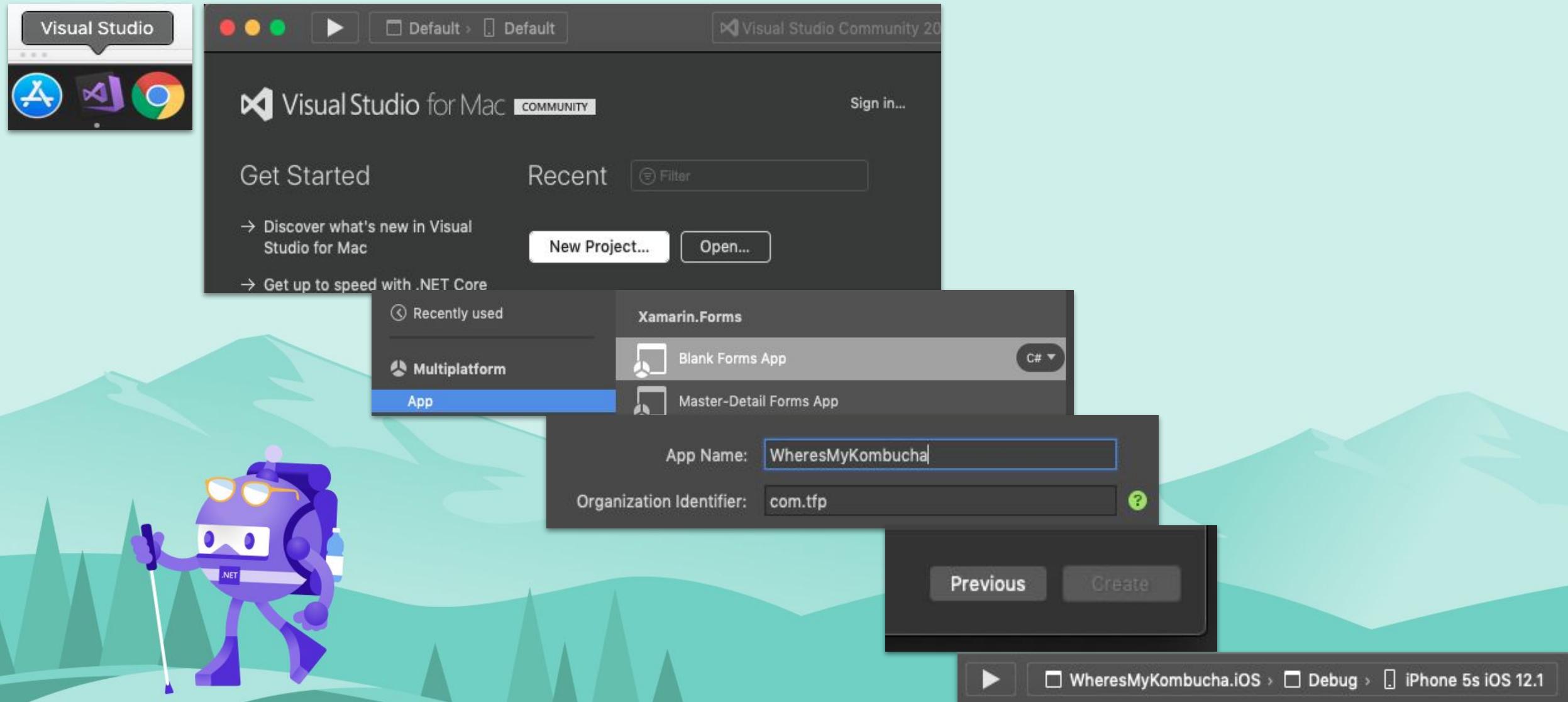


SmartHotel360.Website > Debug > Default
Solution
SmartHotel360.Website No selection
Solution Items
.editorconfig
SmartHotel360.Web
Dependencies
ClientApp
Controllers
1 using System;
2 using System.Collections.Generic;
3 using Microsoft.Extensions.DependencyInjection;
4 using Newtonsoft.Json;
5 Remove Unnecessary Usings
6
7

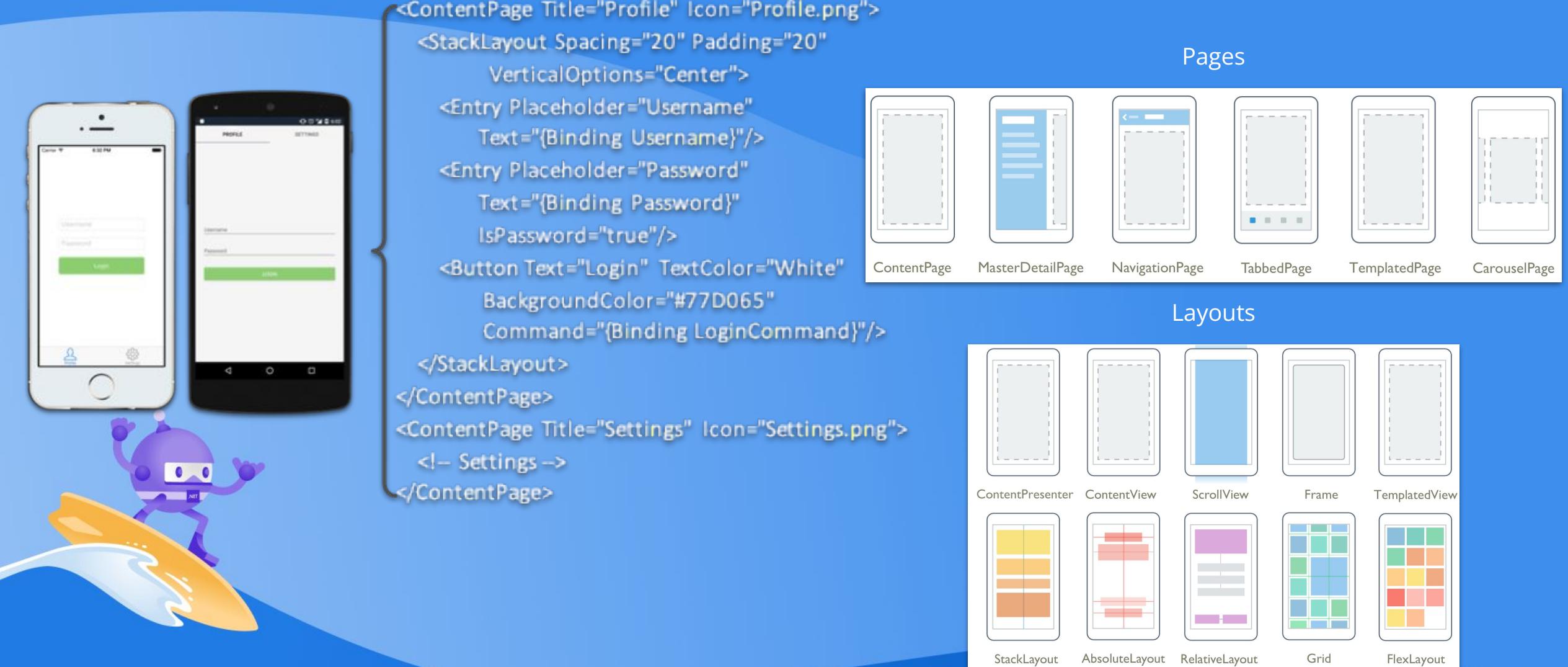
Read more about activating your license

[Download Visual Studio for Mac](#)

After Installation



Xamarin Forms Controls



The image features a cartoon purple robot with a white visor and a small screen on its chest, riding a yellow surfboard on a white wave. Above the robot are two smartphones: one white and one black, both displaying mobile application interfaces.

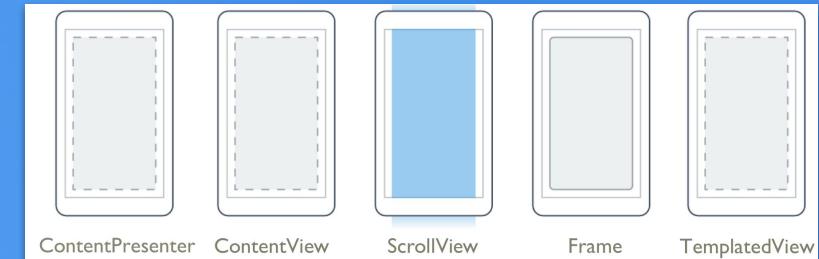
`<ContentPage Title="Profile" Icon="Profile.png">
 <StackLayout Spacing="20" Padding="20"
 VerticalOptions="Center">
 <Entry Placeholder="Username"
 Text="{Binding Username}"/>
 <Entry Placeholder="Password"
 Text="{Binding Password}"/>
 <IsPassword="true"/>
 <Button Text="Login" TextColor="White"
 BackgroundColor="#77D065"
 Command="{Binding LoginCommand}"/>
 </StackLayout>
</ContentPage>
<ContentPage Title="Settings" Icon="Settings.png">
 <!-- Settings -->
</ContentPage>`

Pages



ContentPage MasterDetailPage NavigationPage TabbedPage TemplatedPage CarouselPage

Layouts



ContentPresenter ContentView ScrollView Frame TemplatedView



StackLayout AbsoluteLayout RelativeLayout Grid FlexLayout

Quick Changes

- Choose to use XAML and designer files or just C#.
- Add views inside a layout, and set that inside a page.
- Notice the ease of making updates:



```
public class SimpleContentPage : ContentPage
{
    public SimpleContentPage ()
    {
        var label1 = new Label {
            Text = "This is my label",
            Font = Font.BoldSystemFontOfSize(NamedSize.Large)
        };

        var entry1 = new Entry {
            Placeholder = "Type something here"
        };

        var button1 = new Button {
            Text = "Click Me!"
        };

        button1.Clicked += (sender, e) => {
            label1.Text = "User typed: " + entry1.Text;
        };

        Content = new StackLayout {
            Padding = 30,
            Spacing = 10,
            Children = {label1, entry1, button1}
        };
    }
}
```



Connected Electronics

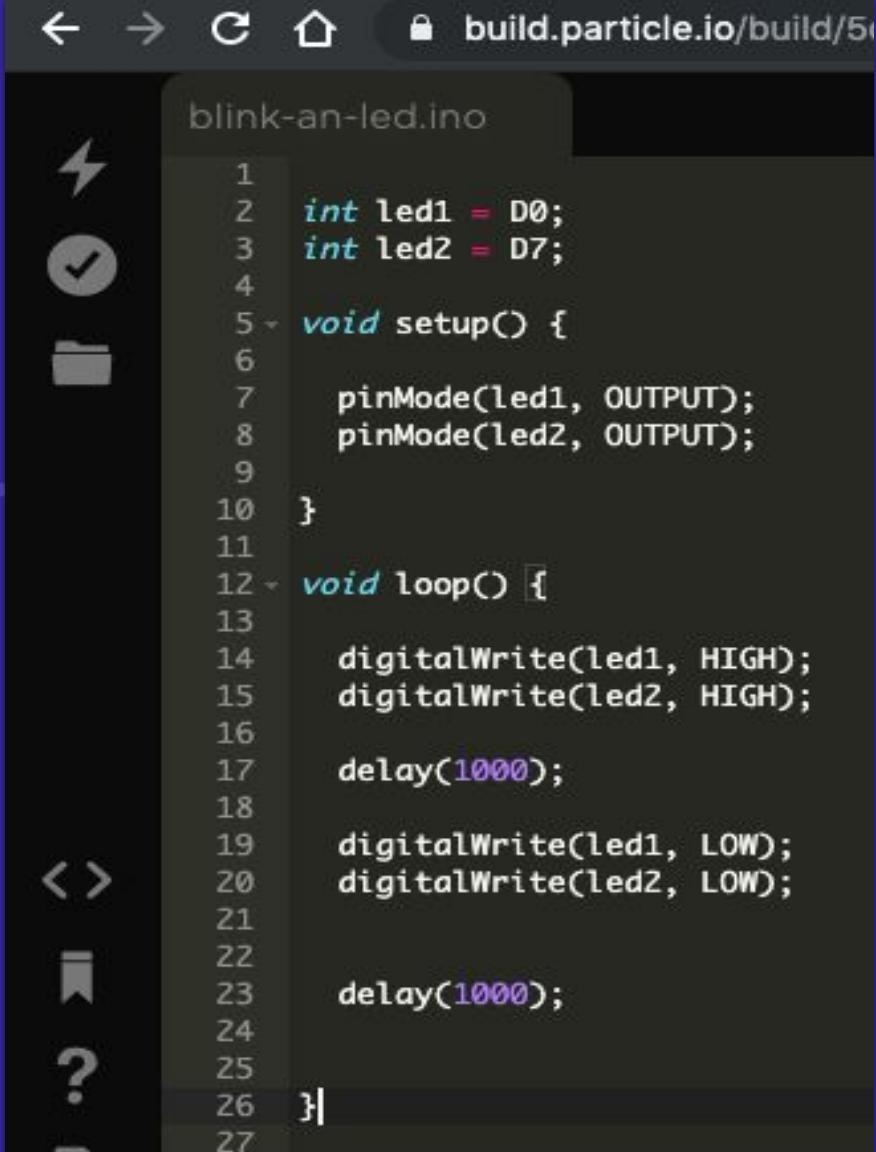
Part 2

Connecting apps with the physical world



Brief Intro to Electronics

- Microcontrollers vs Microprocessors
- Direct Current - Ground = Black = Negative Photon
- Particle vs Arduino vs Node ESP 32
- Led - Shorter End is negative, It won't work if you flip the leads
- Code through the browser
- C++, setup() & loop()
- Particle Function, Variable & Events
- Download Particle & connect it to wifi
- Deviceld & `Particle Token Create`



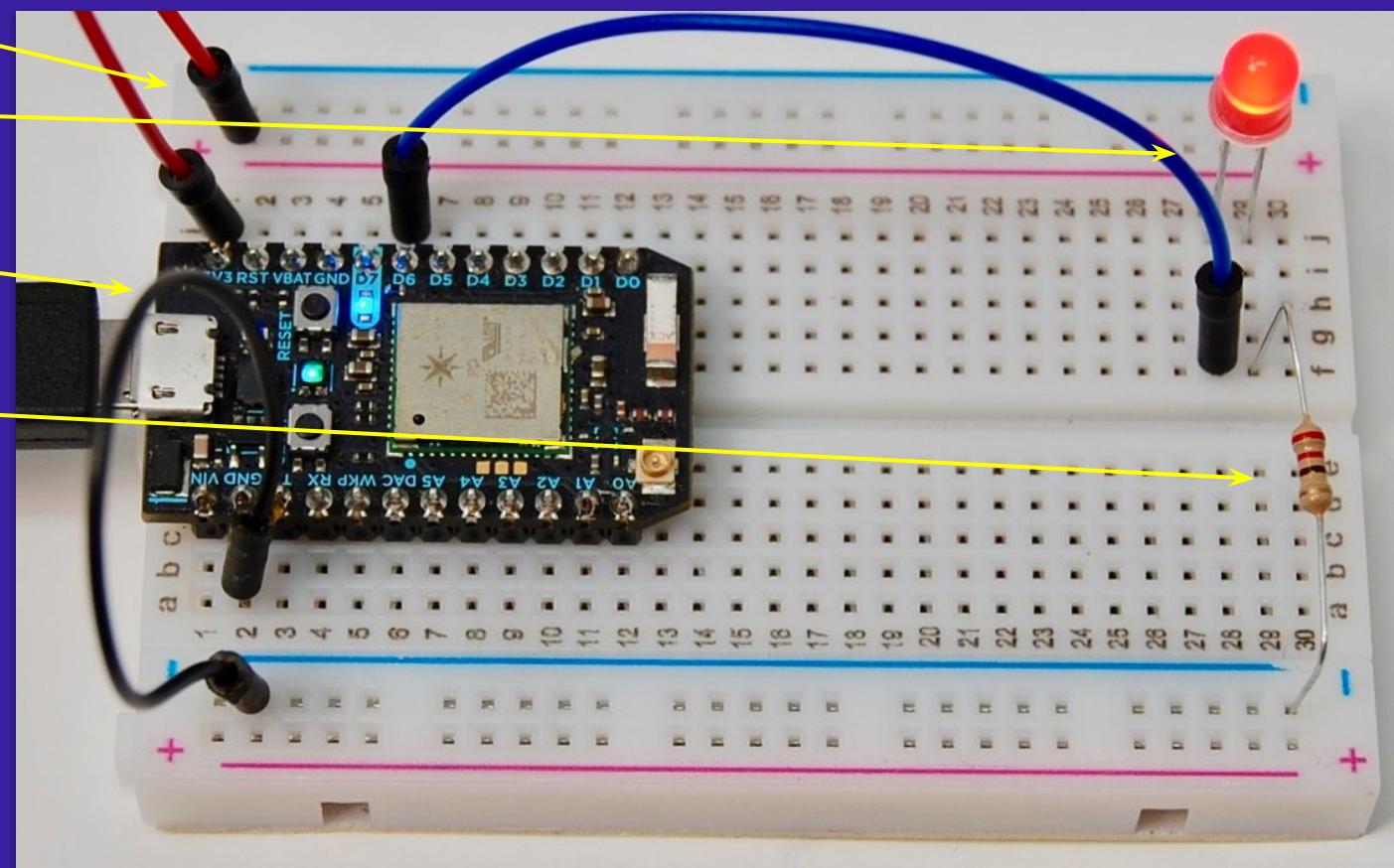
The screenshot shows a web-based development environment for Particle devices. At the top, there are navigation icons (back, forward, refresh, home) and a URL bar showing "build.particle.io/build/5...". Below the URL is the file name "blink-an-led.ino". The main area contains the C++ code for the sketch:

```
1 int led1 = D0;
2 int led2 = D7;
3
4 void setup() {
5     pinMode(led1, OUTPUT);
6     pinMode(led2, OUTPUT);
7 }
8
9 void loop() {
10
11     digitalWrite(led1, HIGH);
12     digitalWrite(led2, HIGH);
13
14     delay(1000);
15
16     digitalWrite(led1, LOW);
17     digitalWrite(led2, LOW);
18
19     delay(1000);
20
21
22
23
24
25
26 }
27
```

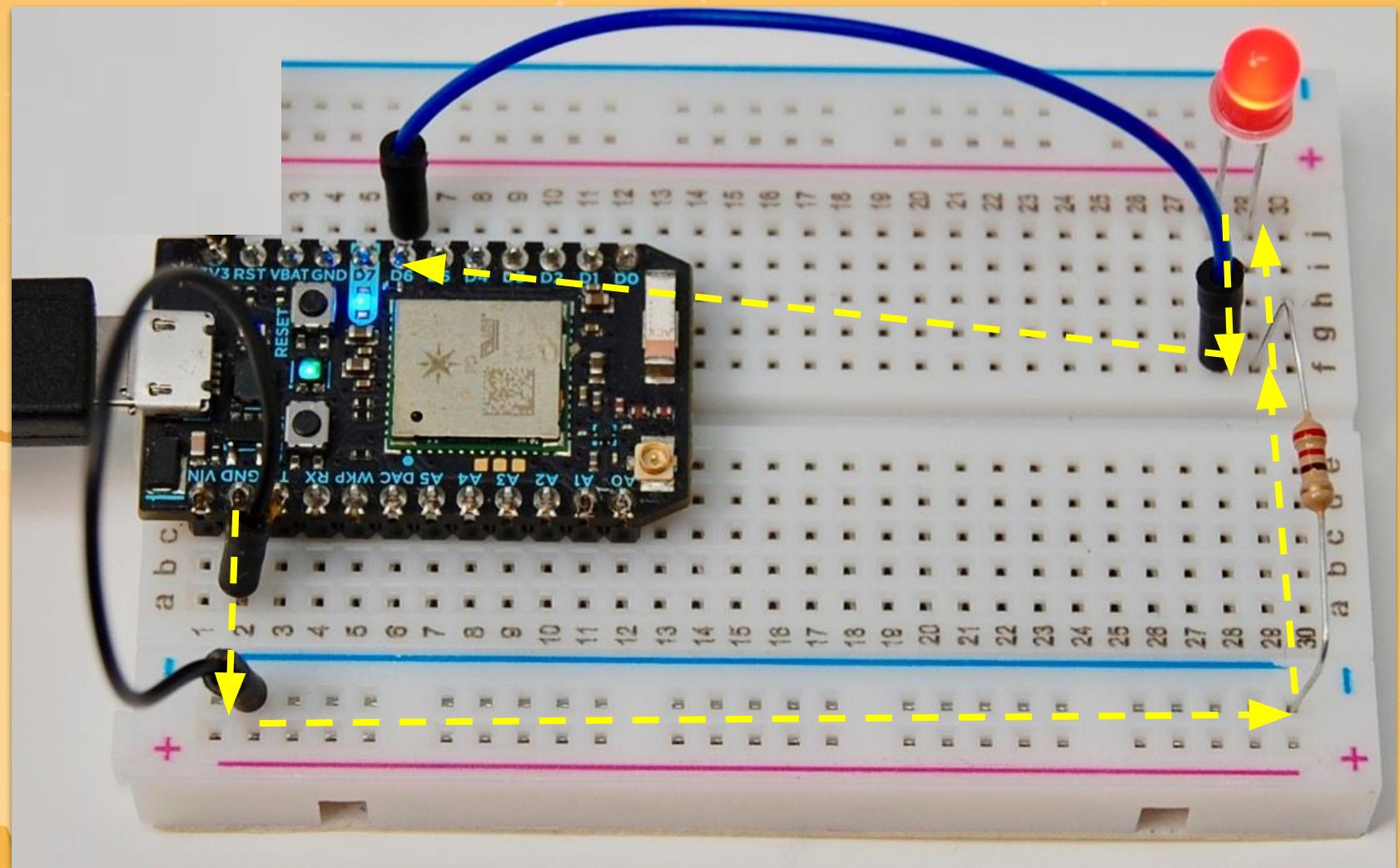
The code initializes two pins, D0 and D7, as outputs. In the setup() function, the pins are set to output mode. In the loop(), the LEDs are alternately turned on and off every 1000 milliseconds.

Basic electronic components

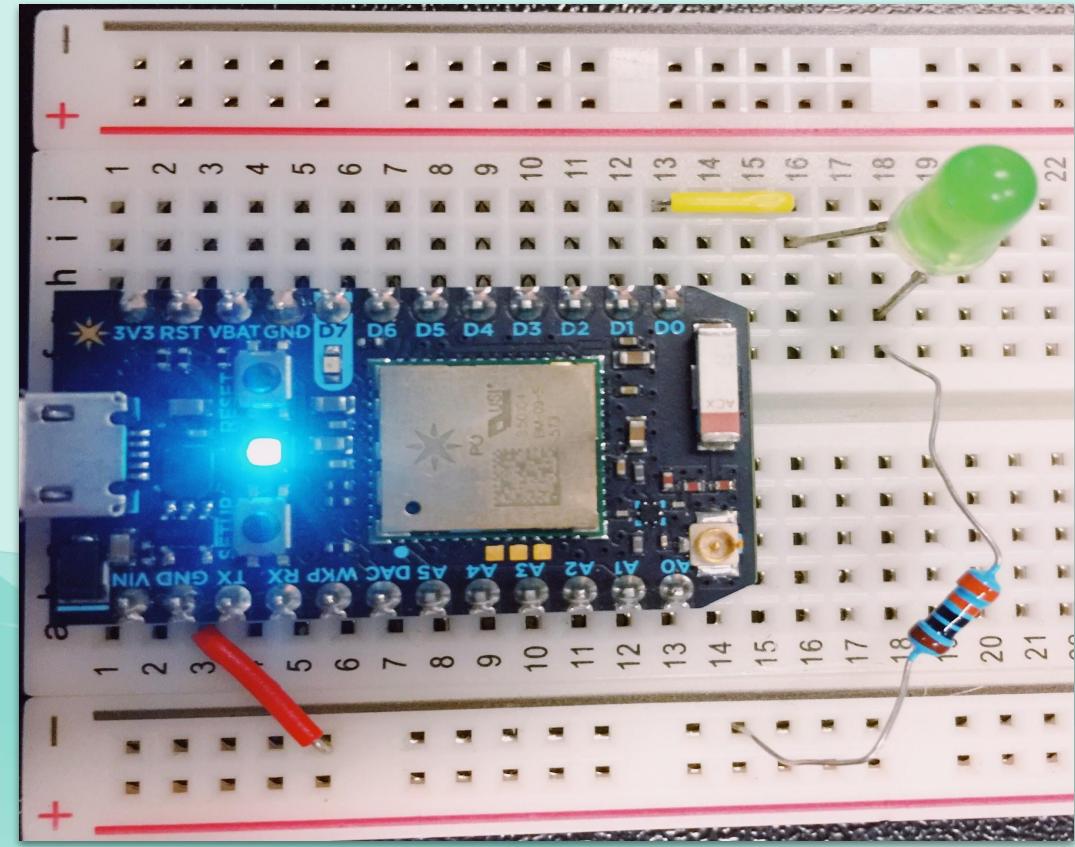
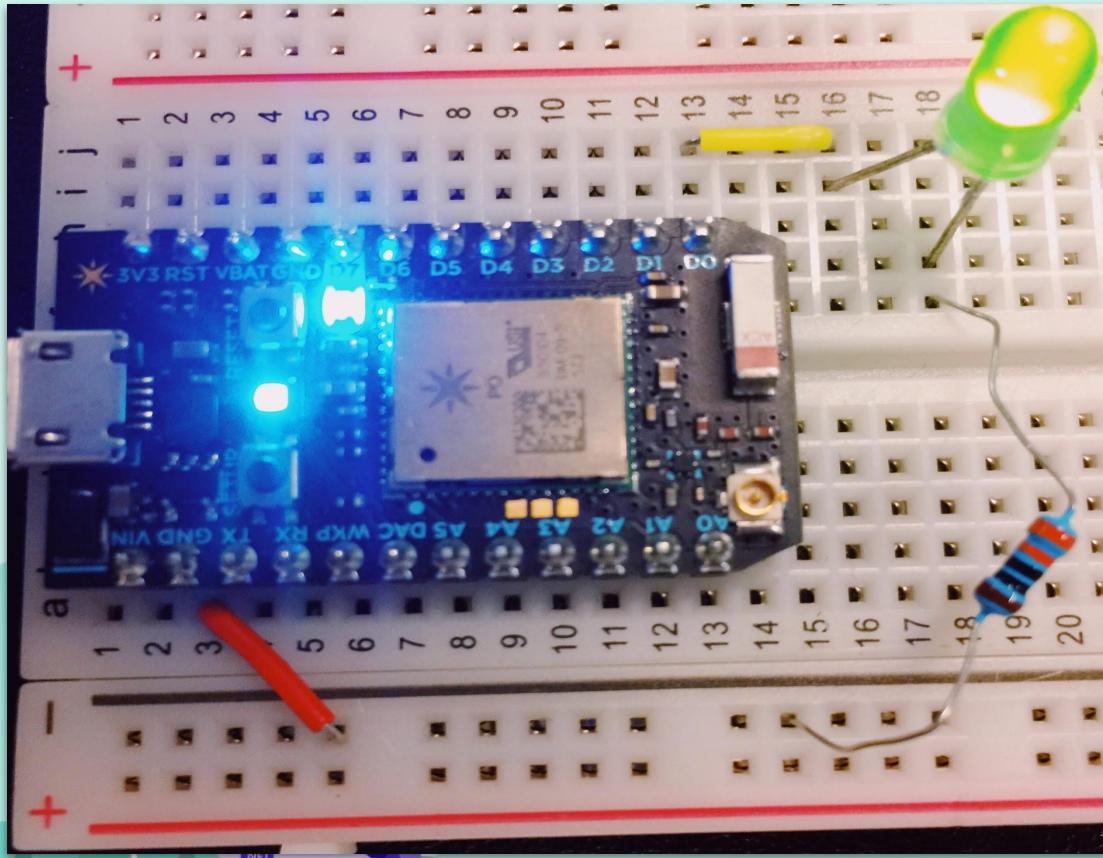
- Breadboard
- LED
- Microcontroller
- Resistor
- Voltage



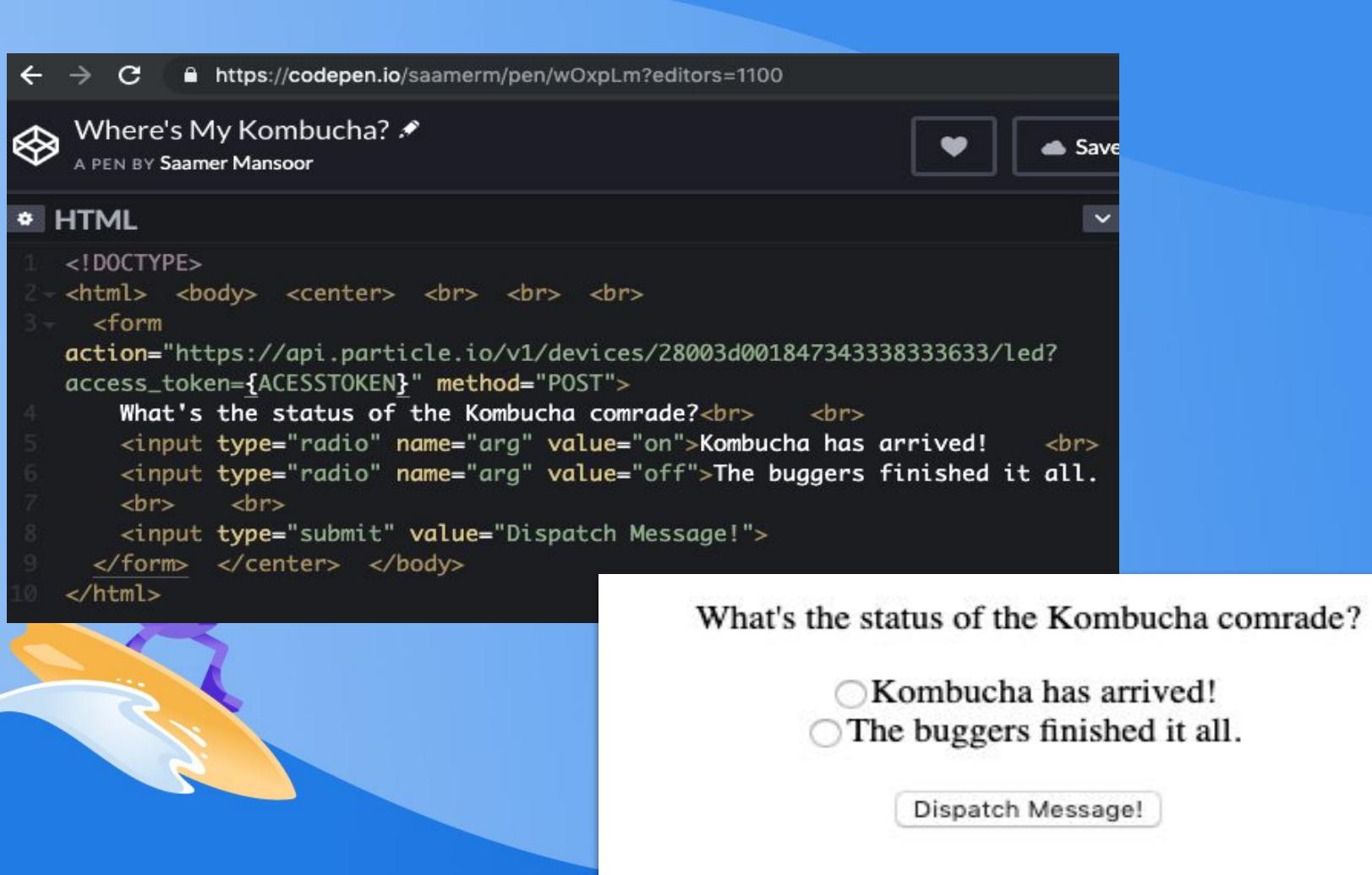
Current Flow Direction



On-Off States



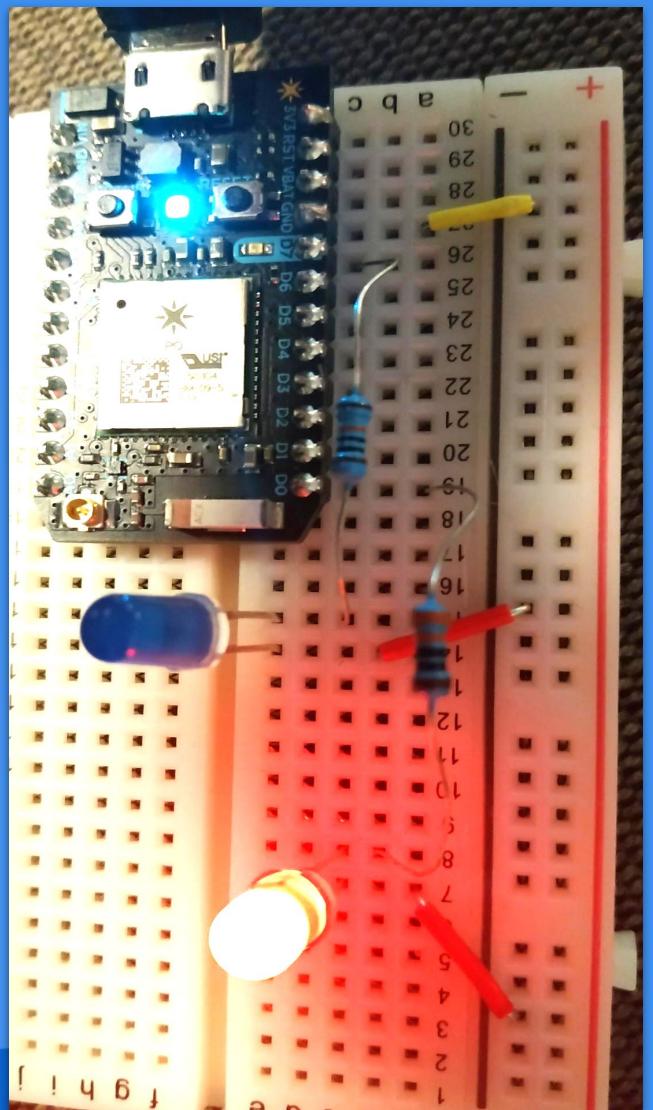
Check Real-time status from Web



The image shows a screenshot of a CodePen interface. The title of the pen is "Where's My Kombucha?". The code editor contains the following HTML:

```
<!DOCTYPE>
<html> <body> <center> <br> <br> <br>
<form action="https://api.particle.io/v1/devices/28003d001847343338333633/led?access_token={ACESSTOKEN}" method="POST">
  What's the status of the Kombucha comrade?<br> <br>
  <input type="radio" name="arg" value="on">Kombucha has arrived! <br>
  <input type="radio" name="arg" value="off">The buggers finished it all.
  <br> <br>
  <input type="submit" value="Dispatch Message!">
</form> </center> </body>
</html>
```

The preview window shows a form with the question "What's the status of the Kombucha comrade?". It contains two radio buttons: "Kombucha has arrived!" and "The buggers finished it all.". Below the radio buttons is a "Dispatch Message!" button.

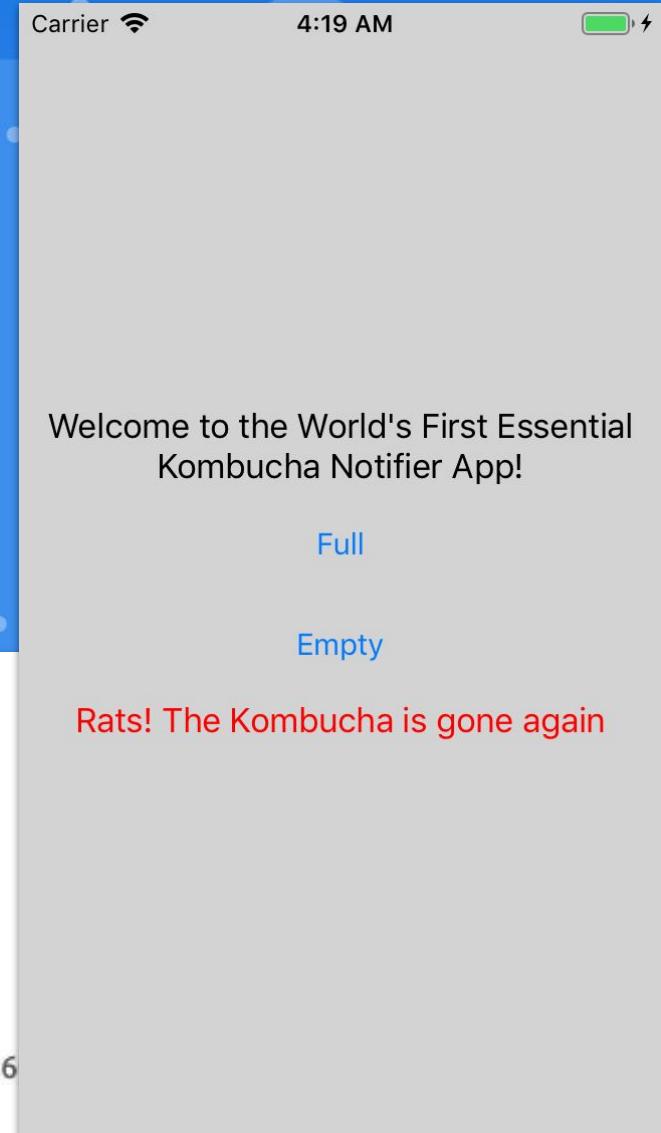


Particle Variable

Get Current Status of the Cloud Variable by calling a GET call to this URL:

https://api.particle.io/v1/devices/{DeviceID}/digitalValue?access_token={AccessToken}

```
{ "cmd": "VarReturn",
  "name": "digitalValue",
  "result": 0,
  "coreInfo": {
    "last_app": "",
    "last_heard": "2019-03-19T07:48:13.640Z",
    "connected": true,
    "last_handshake_at": "2019-03-19T07:37:12.36",
    "deviceID": "28003d001847343338333633",
    "product_id": 6
  }
}
```



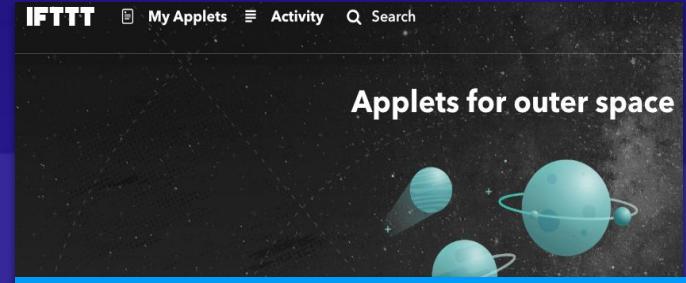


IOT - Internet of Things

Part 3

Leveraging the power of other connected services



A horizontal sequence of three screenshots from the IFTTT mobile application. 1. 'Choose a service' (Step 1 of 6): Shows a search bar with 'Particle' typed in, and a result card for 'Particle' with its logo. 2. 'Choose trigger' (Step 2 of 6): Shows four trigger options: 'New event published', 'Monitor a variable', 'Monitor a function result', and 'Monitor your device status'. 3. 'Choose action service' (Step 3 of 6): Shows a search bar with 'Search services' and a grid of action icons: 'SMS' (green), 'Email' (blue), 'Phone Call (US only)' (red), and 'Del' (dark blue).

Choose a service

Step 1 of 6

Q Particle

Particle

Choose trigger

Step 2 of 6

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

Choose action service

Step 3 of 6

Search services

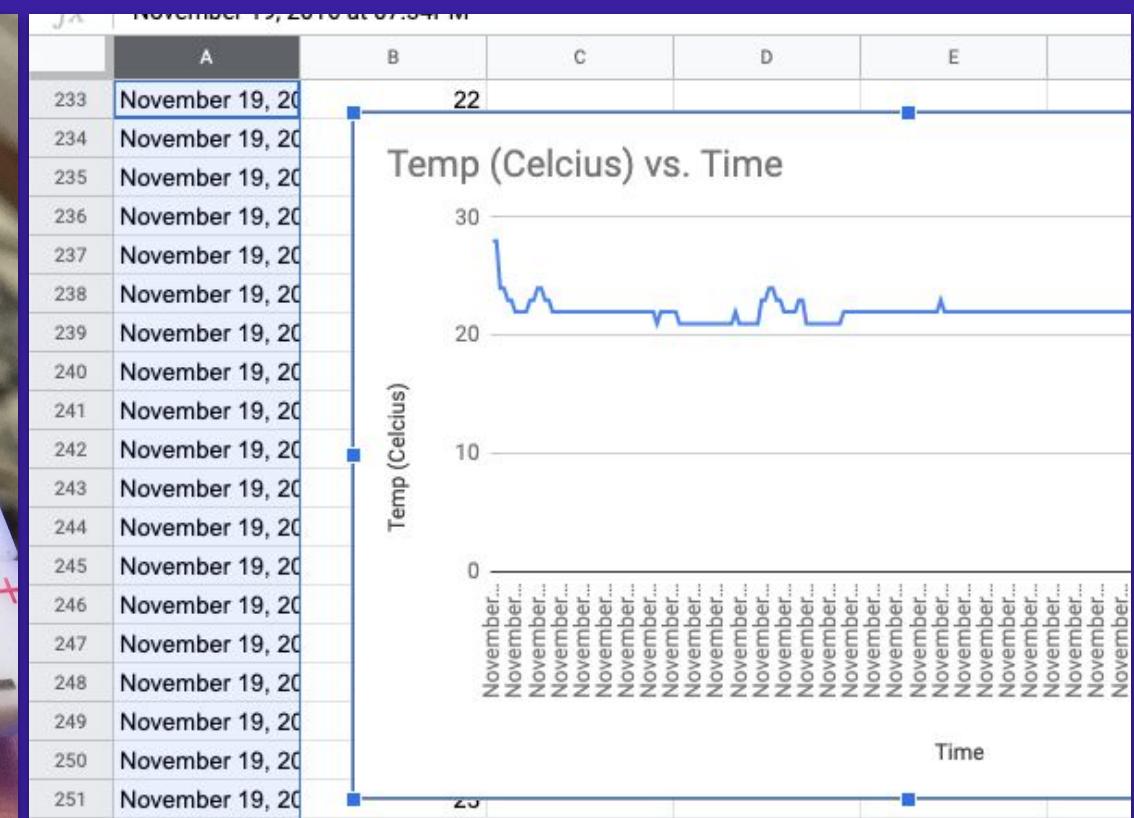
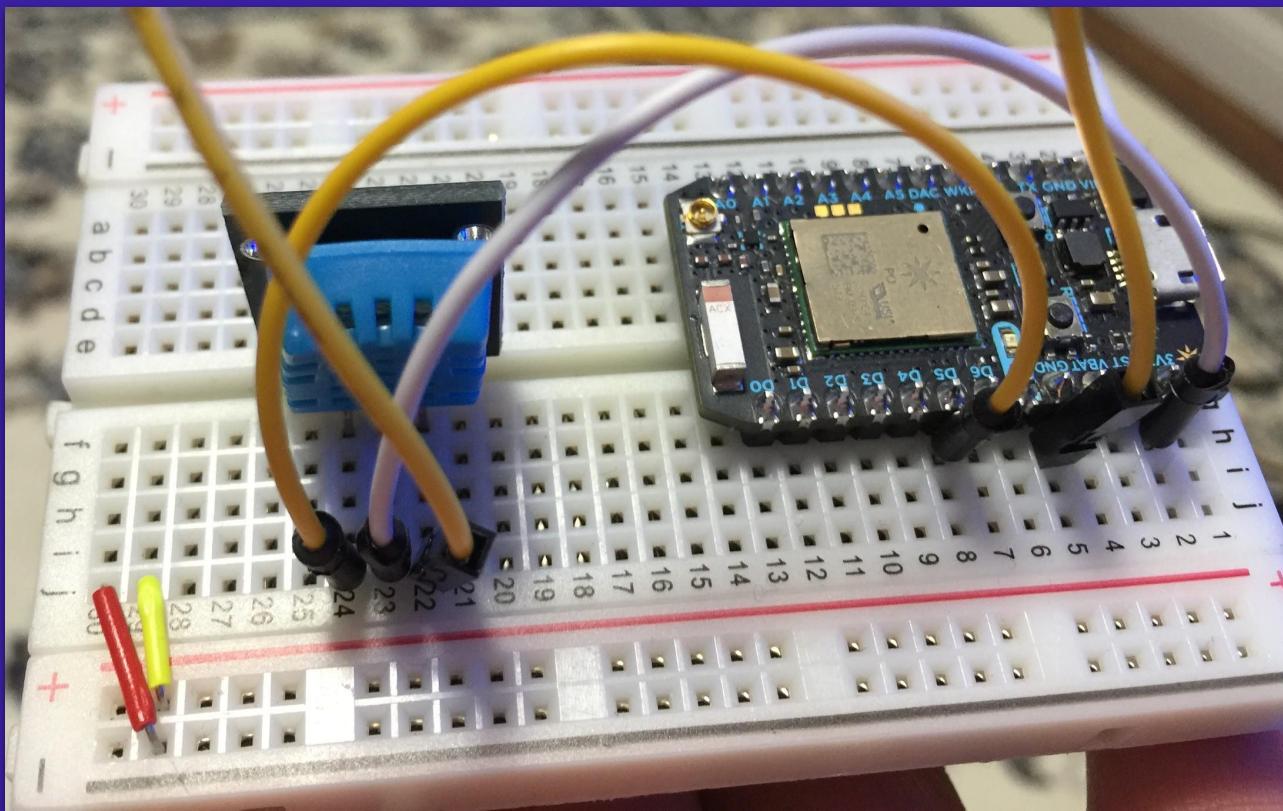
sms

Email

Phone Call (US only)

Del

Experimenting with Temperature





Using Raspberry Pi's instead

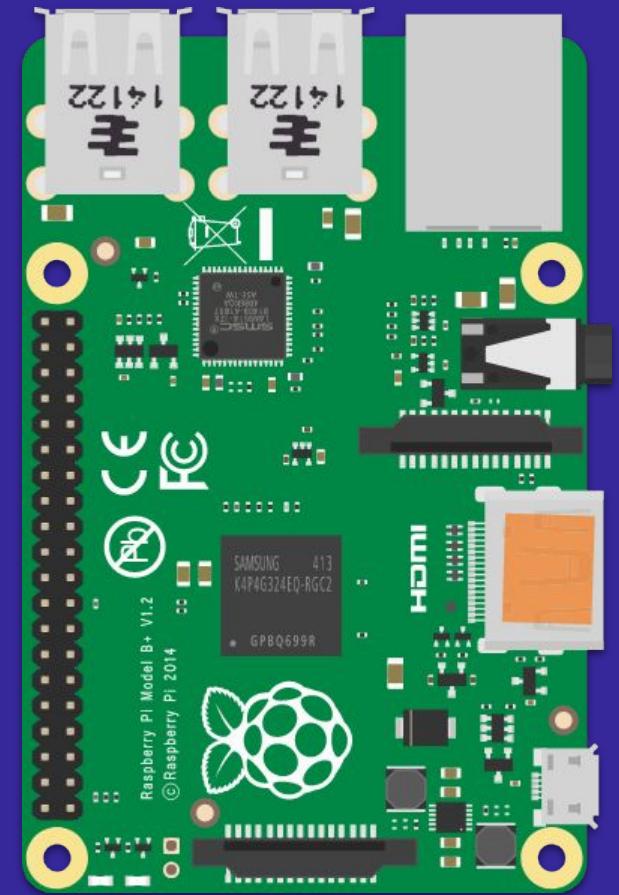
Part 4

Swapping electronics to create the sanitizer



What do we need to change?

- Microprocessor-Can run multiple applications
- Can be coded with python
- 4 USBs, Audio Out, HDMI, WiFi



Raspberry Pi Setup - From scratch

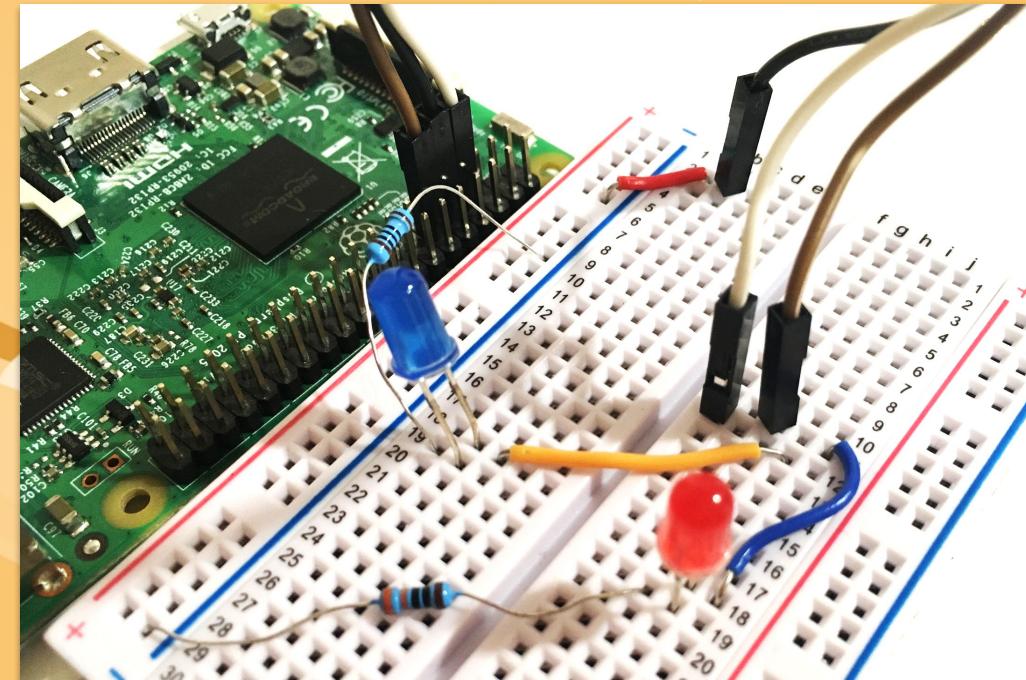
- Install Raspbian OS on your Raspberry Pi's micro SD Card:
<https://www.raspberrypi.org/software/>
- Open the terminal, install particle library:
`bash <(curl -sL https://particle.io/install-pi)`
- Create account here: build.particle.io
- Add your raspberry pi to the Particle Cloud:
“particle-agent setup” and login.
- Change the Device type in the Particle Web IDE, so code flashes to the Raspberry Pi
- Don't forget to update the DevicelD in your Xamarin code
- Create a token : `$ particle token create`

What do we need to change?

- Search for Pin-Out Diagram
- Connect D0, GND & D1 of the Raspberry Pi
- Deploy same code from Build IDE

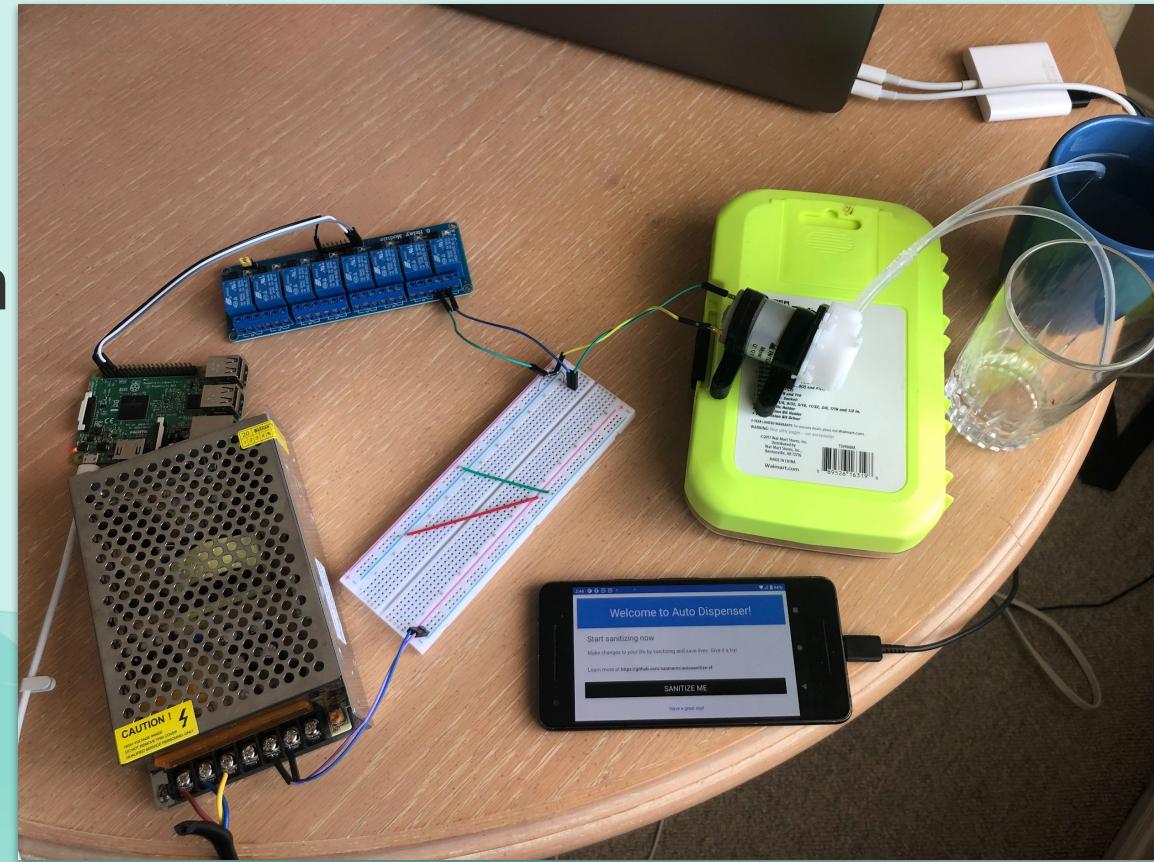


Peripherals	GPIO	Particle	Pin #	Pin #	Particle	GPIO	Peripherals
		3.3V	1	X	X	2	5V
I2C	GPIO2	SDA	3	X	X	4	5V
	GPIO3	SCL	5	X	X	6	GND
Digital I/O	GPIO4	D0	7	X	X	8	TX
			9	X	X	10	RX
Digital I/O	GPIO17	D1	11	X	X	12	D9/A0
Digital I/O	GPIO27	D2	13	X	X	14	GND
Digital I/O	GPIO22	D3	15	X	X	16	D10/A1
		3.3V	17	X	X	18	D11/A2
SPI	GPIO10	MOSI	19	X	X	20	GND
	GPIO9	MISO	21	X	X	22	D12/A3
	GPIO11	SCK	23	X	X	24	CE0
		GND	25	X	X	26	CE1
DO NOT USE	ID_SD	DO NOT USE	27	X	X	28	DO NOT USE
Digital I/O	GPIO5	D4	29	X	X	30	GND
Digital I/O	GPIO6	D5	31	X	X	32	D13/A4
PWM 2	GPIO13	D6	33	X	X	34	GND
PWM 2	GPIO19	D7	35	X	X	36	D14/A5
Digital I/O	GPIO26	D8	37	X	X	38	D15/A6
		GND	39	X	X	40	D16/A7
							GPIO21
							Digital I/O



What do we need to change?

- Power supply, Relay, Pump,
- Diode & Sanitizer for protection
- Code changes
- RaspberryPi



Thanks for joining!

Questions?

GitHub: github.com/saamer/WheresMyKombucha

Email: saamer@thefirstprototype.com

LinkedIn: linkedin.com/in/saamer

Twitter: @saamer



SLIDES BEYOND THIS TO BE
DELETED ONCE FINISHED



Single column of content

Two columns of content

Demo

Presenter

Code Sample

Code sample

RELEASED

Announcemnet

- Value prop 1
- Value prop 2
- Value prop 3



