

# IoT Workshop



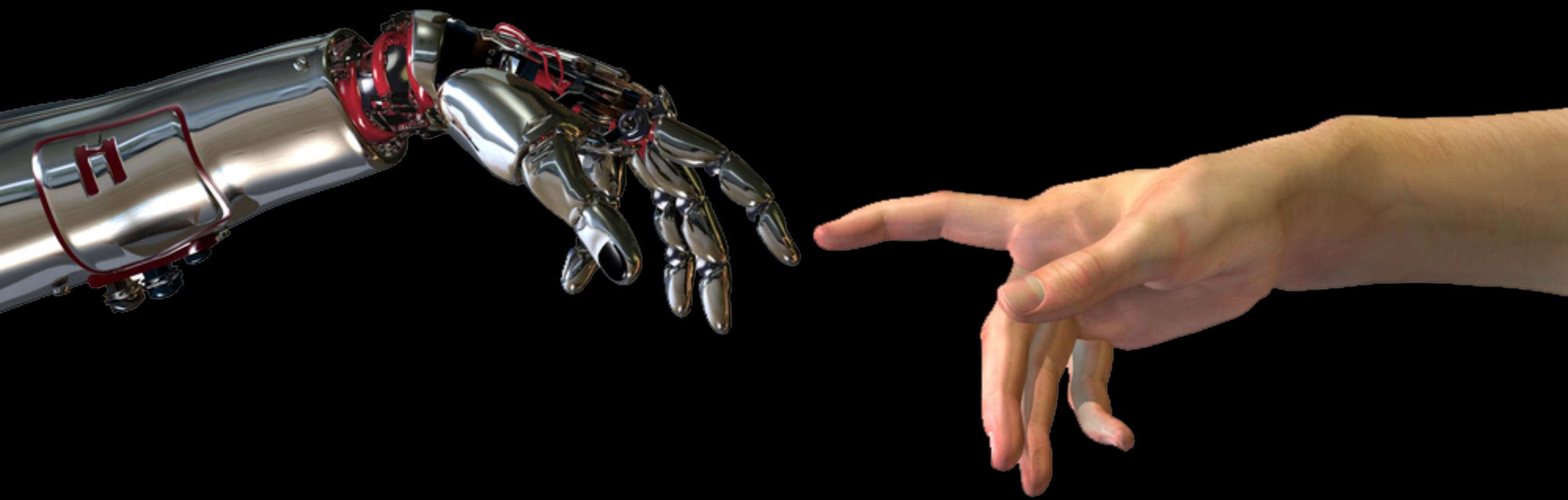
**Imanol Gómez**  
[imanolgomez.net](http://imanolgomez.net)  
7th - 8th January 2019

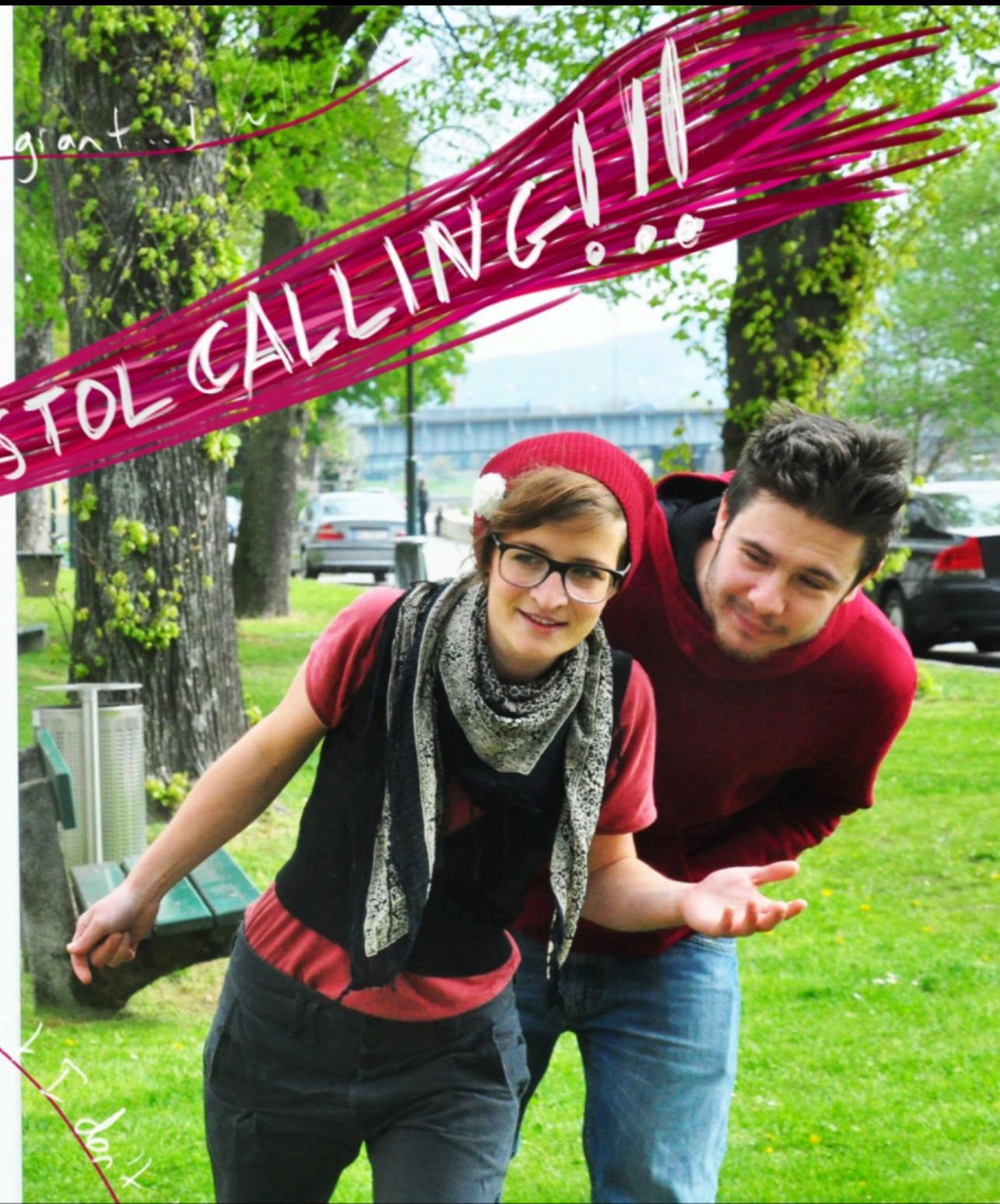
# 1. Introduction

Who am I?

# Imanol Gómez









# FontanaArte







Who are you?

# What will you learn?

1. What is IoT?
2. Arduino
3. ESP8266
4. Blynk
5. Sensors
6. Hackathon
7. Future Work

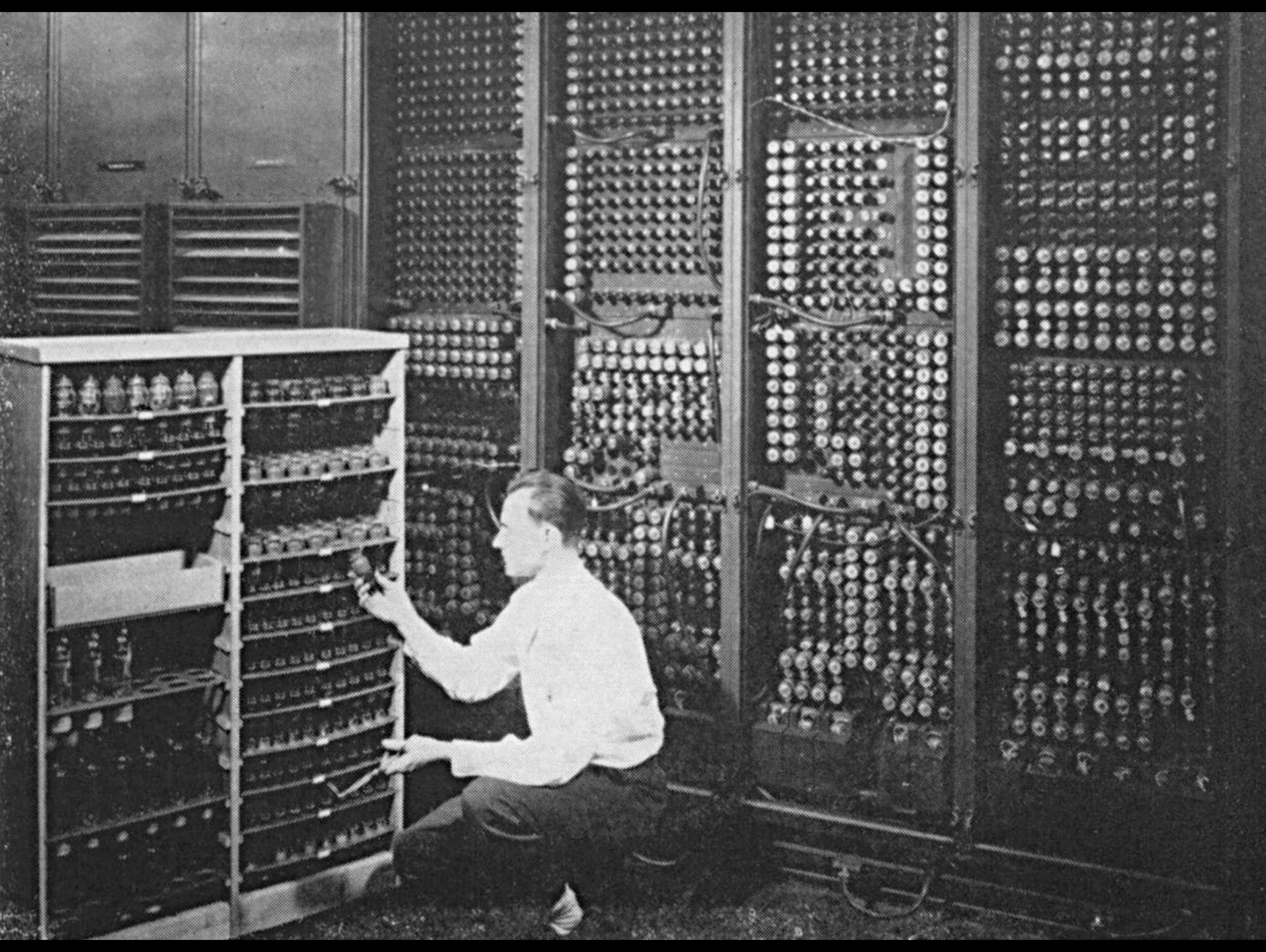
# Repository

[https://github.com/ImanolGo/  
IoTWorkshop2019](https://github.com/ImanolGo/IoTWorkshop2019)

# What is IoT?

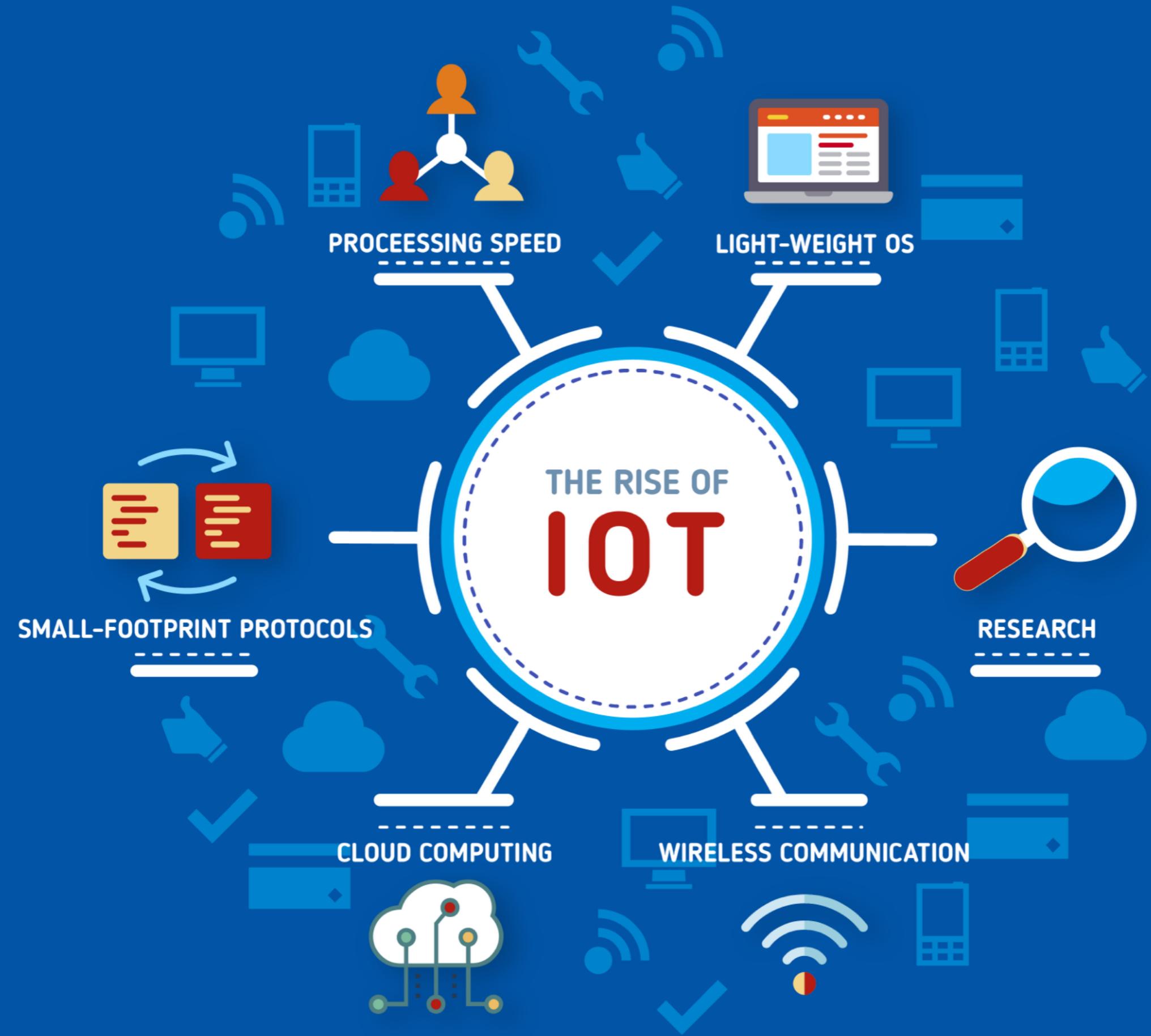


**Neither the ‘Thing’ nor the ‘Internet’ is new**

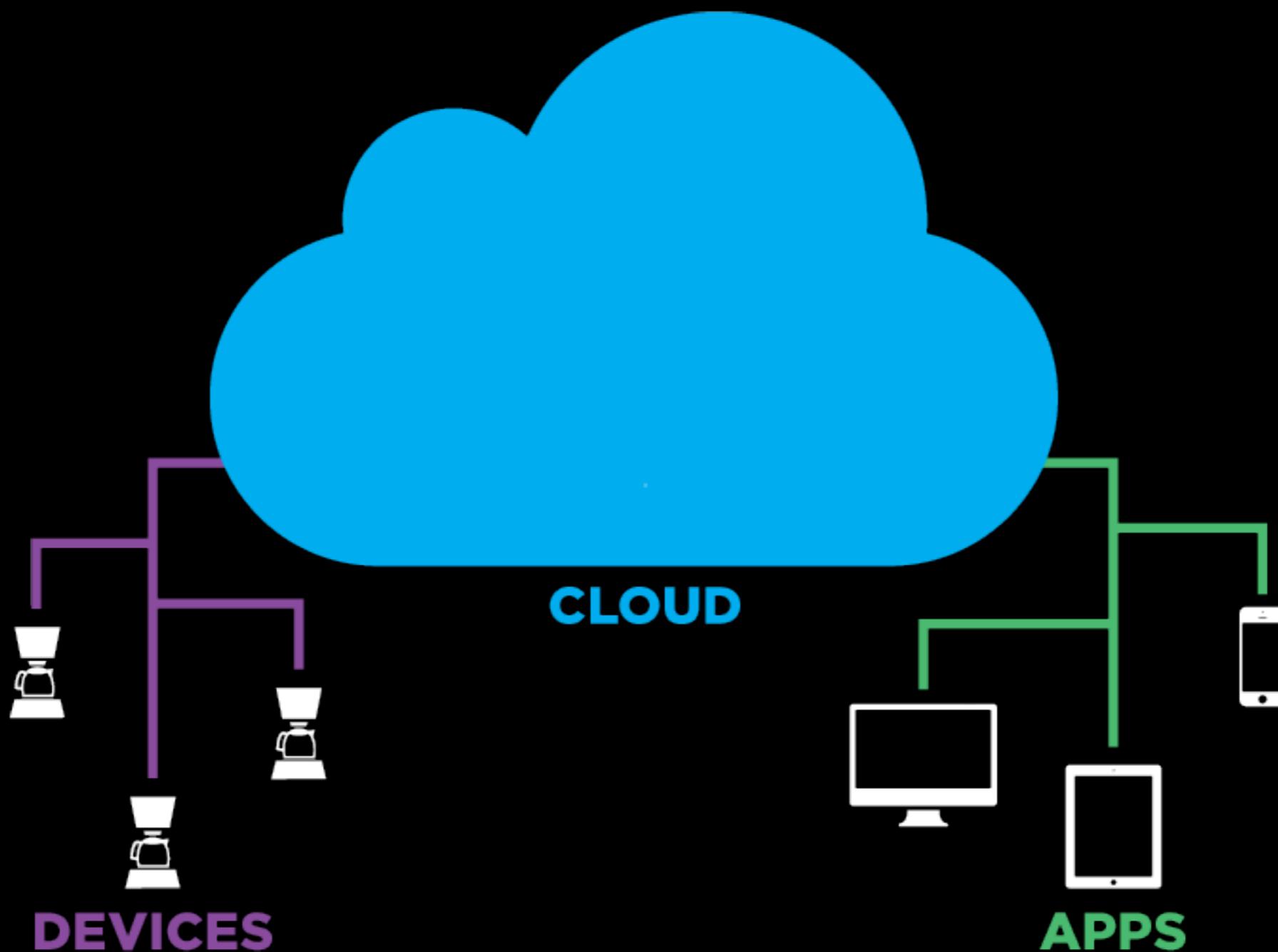


# First Internet Message : 1969





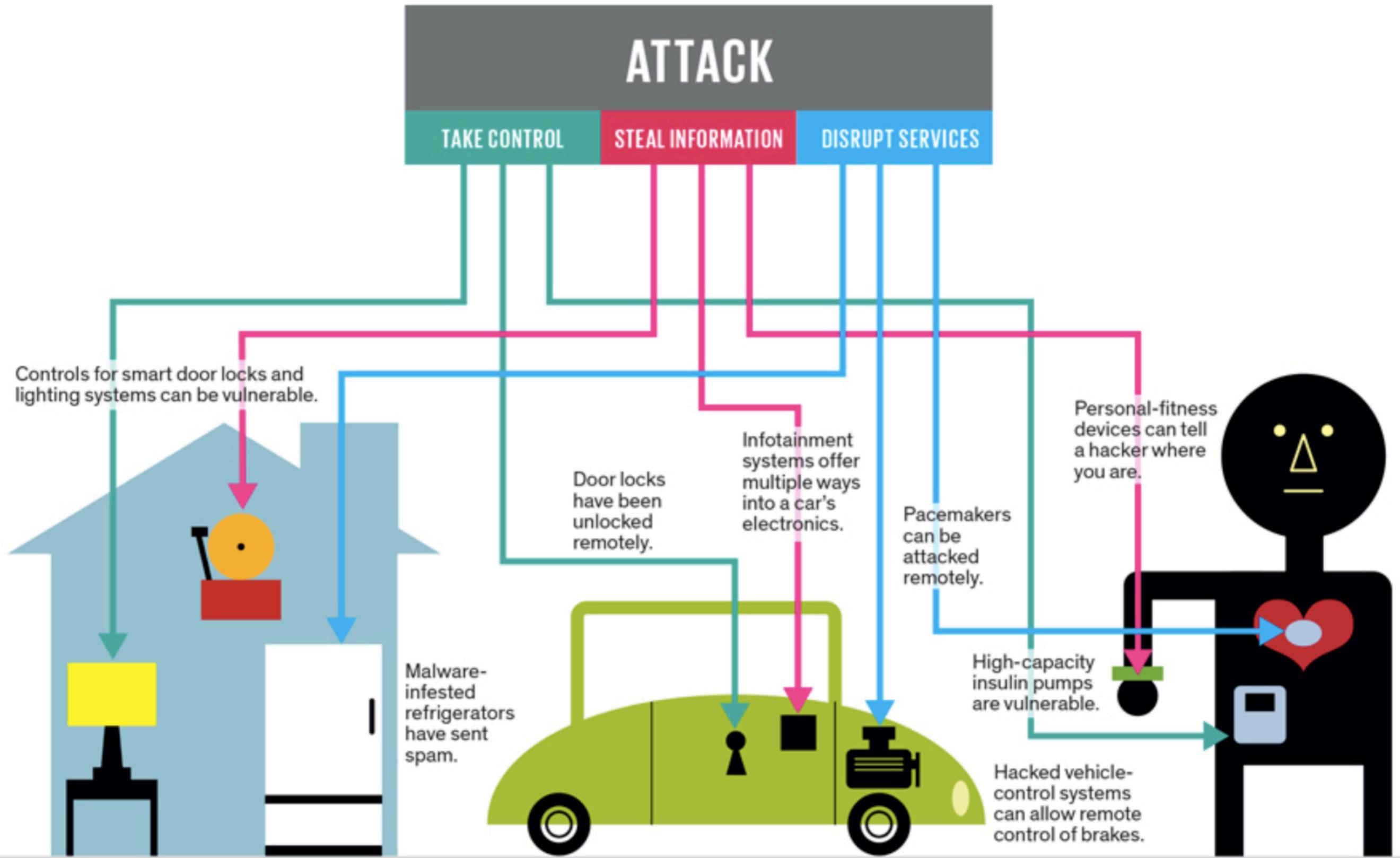
# How does it work?



# The Cloud



# Security



# Examples

# Products



SMART THERMOSTATS

nest™



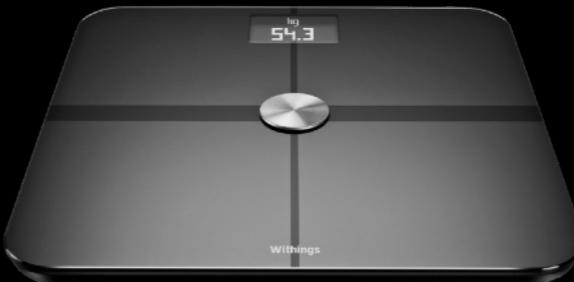
ACTIVITY TRACKERS

NIKE



SMART LIGHTS

PHILIPS



CONNECTED SCALE

Withings

# Applications



FASHION  
LIFESTYLE

TRANSPORT  
MOBILITY/TRAFFIC

RETAIL  
INVENTORY

near-world contexts for your apps  
Developer Preview Kit  
10 Bluetooth Low Energy Beacons



**LOGISTICS**  
REAL TIME



**MANUFACTURING**  
M2M



**CITIES**  
INDUSTRY



**HEALTH**  
BODY



**ENVIRONMENT**  
PREVENTION



**BUILDING**  
INFRASTRUCTURE  




**AGRICULTURE**  
CONTROL



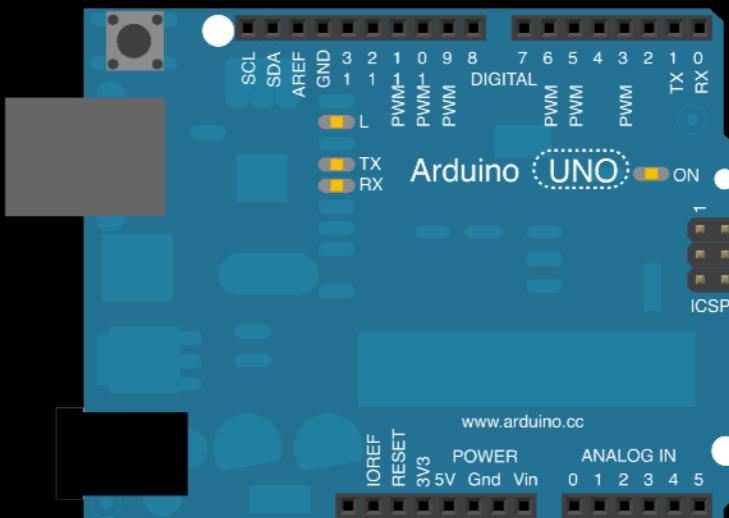
**SECURITY**  
DETECTION

## 2. Arduino



Blink | Arduino 1.6.7

```
1 //*
2 * Blink
3
4 Turns an LED on for one second, then off for one second, repeatedly.
5
6 Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7 it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8 the correct LED pin independent of which board is used.
9 If you want to know what the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27 // initialize digital pin LED_BUILTIN as an output.
28 pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
34 delay(1000); // wait for a second
35 digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
36 delay(1000); // wait for a second
37 }
```



Software

Hardware

# Arduino

- Inexpensive
- Cross-platform
- Simple, clear programming environment
- Open source and extensible software
- Open source and extensible hardware



The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.7". The main window displays the "Blink" sketch code. The code is a classic example that turns an LED on for one second and off for one second, repeatedly. It includes comments explaining the purpose of the LED and how to find its physical connection on different boards. The code also credits the original authors and contributors. At the bottom of the code area, there is a note about the example being in the public domain and a link to the tutorial. The Arduino board selection dropdown at the bottom right shows "NUCLEO-F446RE".

```
/*
  Blink
  Turns an LED on for one second, then off for one second, repeatedly.

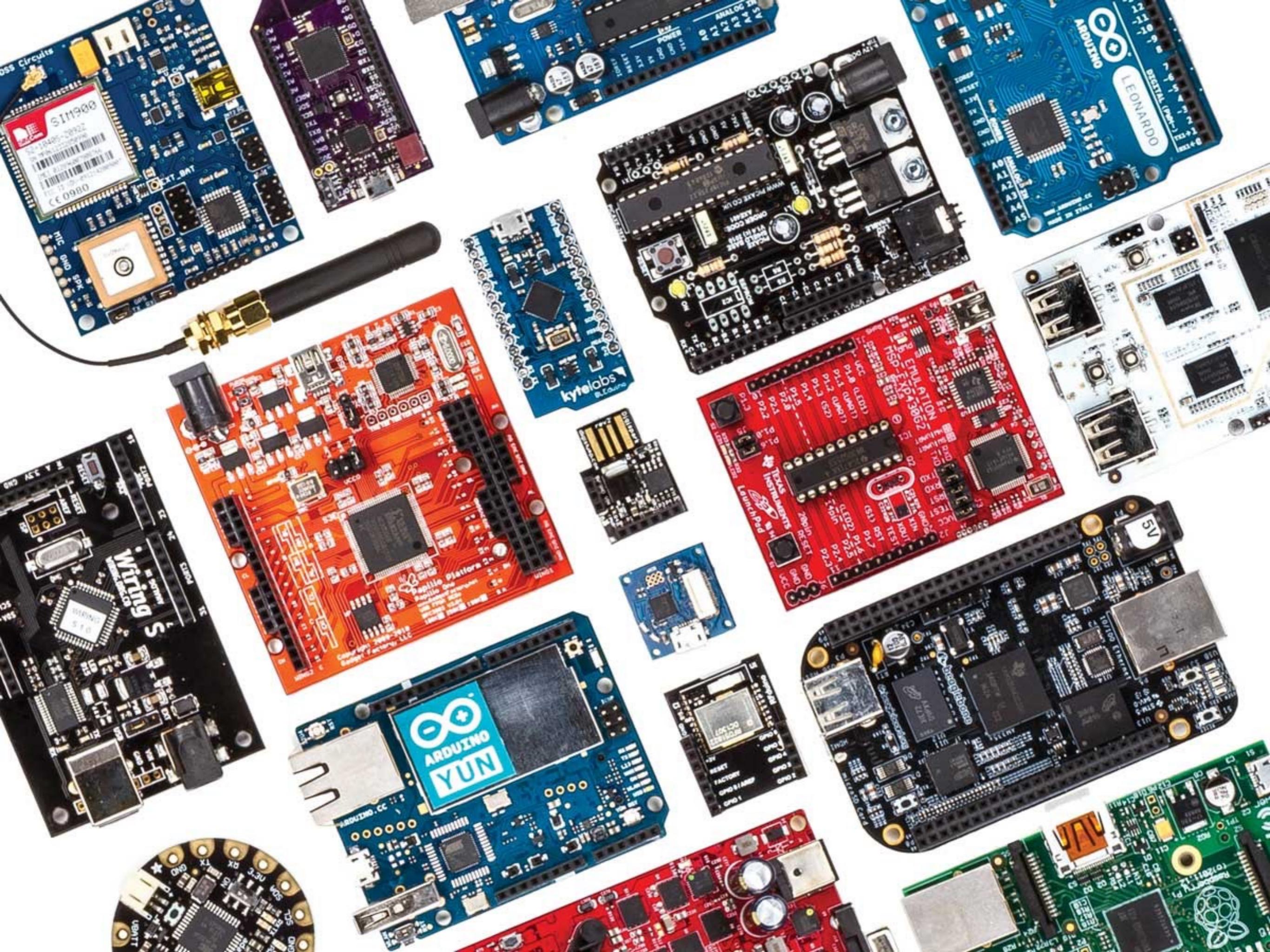
  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```



# Installation

# Install Arduino IDE

## Download the Arduino IDE



### ARDUINO 1.8.8

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



**Mac OS X** 10.8 Mountain Lion or newer

**Linux** 32 bits

**Linux** 64 bits

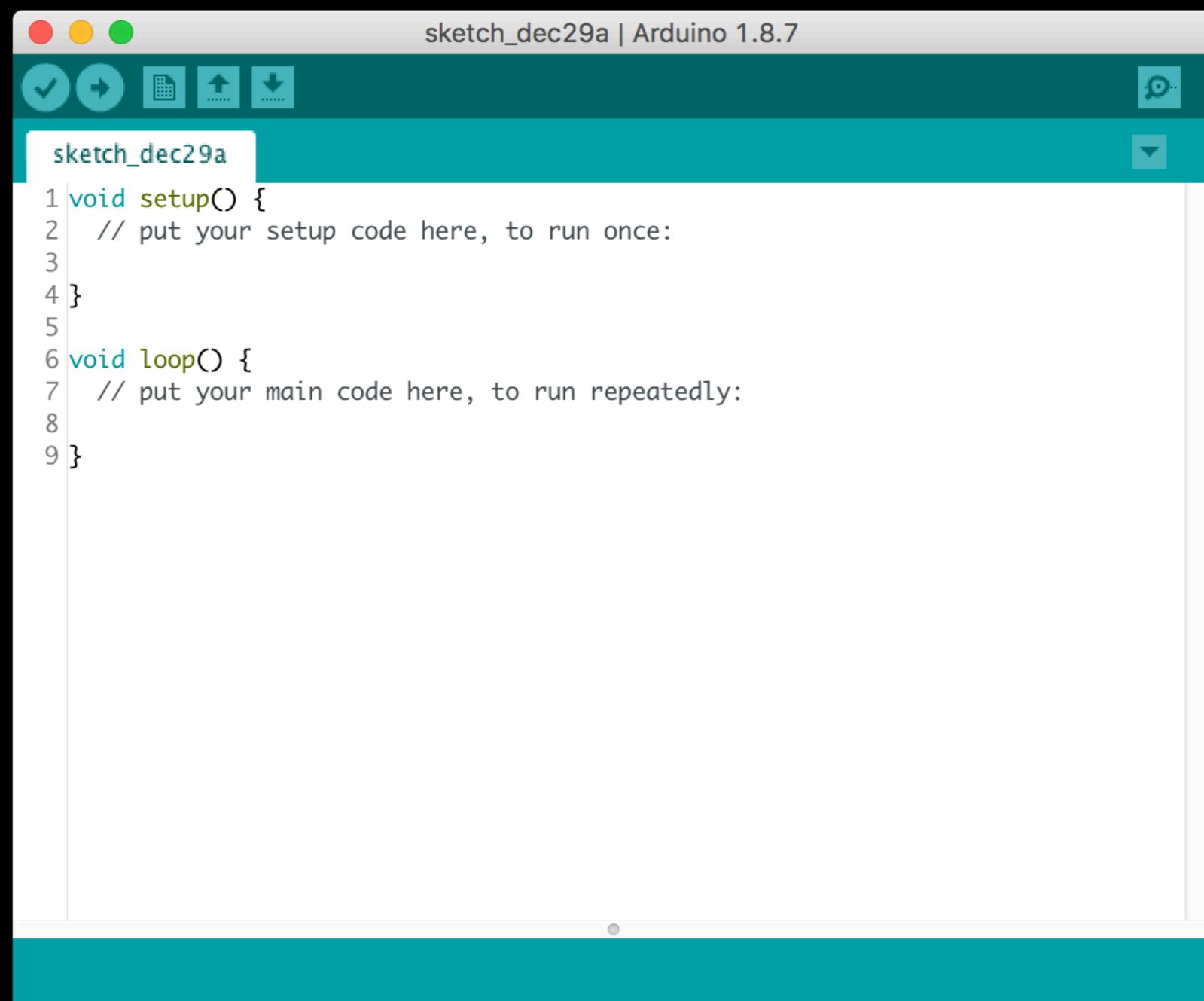
**Linux** ARM

[Release Notes](#)

[Source Code](#)

[Checksums \(sha512\)](#)

# Open Arduino IDE



# The Semicolon;

A semicolon needs to follow every statement written in the Arduino programming language.

```
int LEDpin = 9;
```

# The Double Backslash For Single Line Comments //

- Comments are what you use to annotate code.
- Comments are meant to inform you and anyone else
- Comments will be ignored by the compiler

```
//This is the pin on the Arduino that the LED is plugged into  
int LEDpin = 9
```

# Curly Braces { }

- Curly braces are used to enclose further instructions carried out by a function
- There is always an opening curly bracket and a closing curly bracket

```
void loop() { //this curly brace opens  
//way cool program here  
} //this curly brace closes
```

# Functions ()

```
pinMode(13, OUTPUT);
//Sets the mode of an Arduino pin
```

```
millis();
//Retrieves the length of time in milliseconds that the Arduino has been running
```

# Void Setup()

1. setup() only runs once.
2. setup() needs to be the first function in your Arduino sketch.
3. setup() must have opening and closing curly braces.

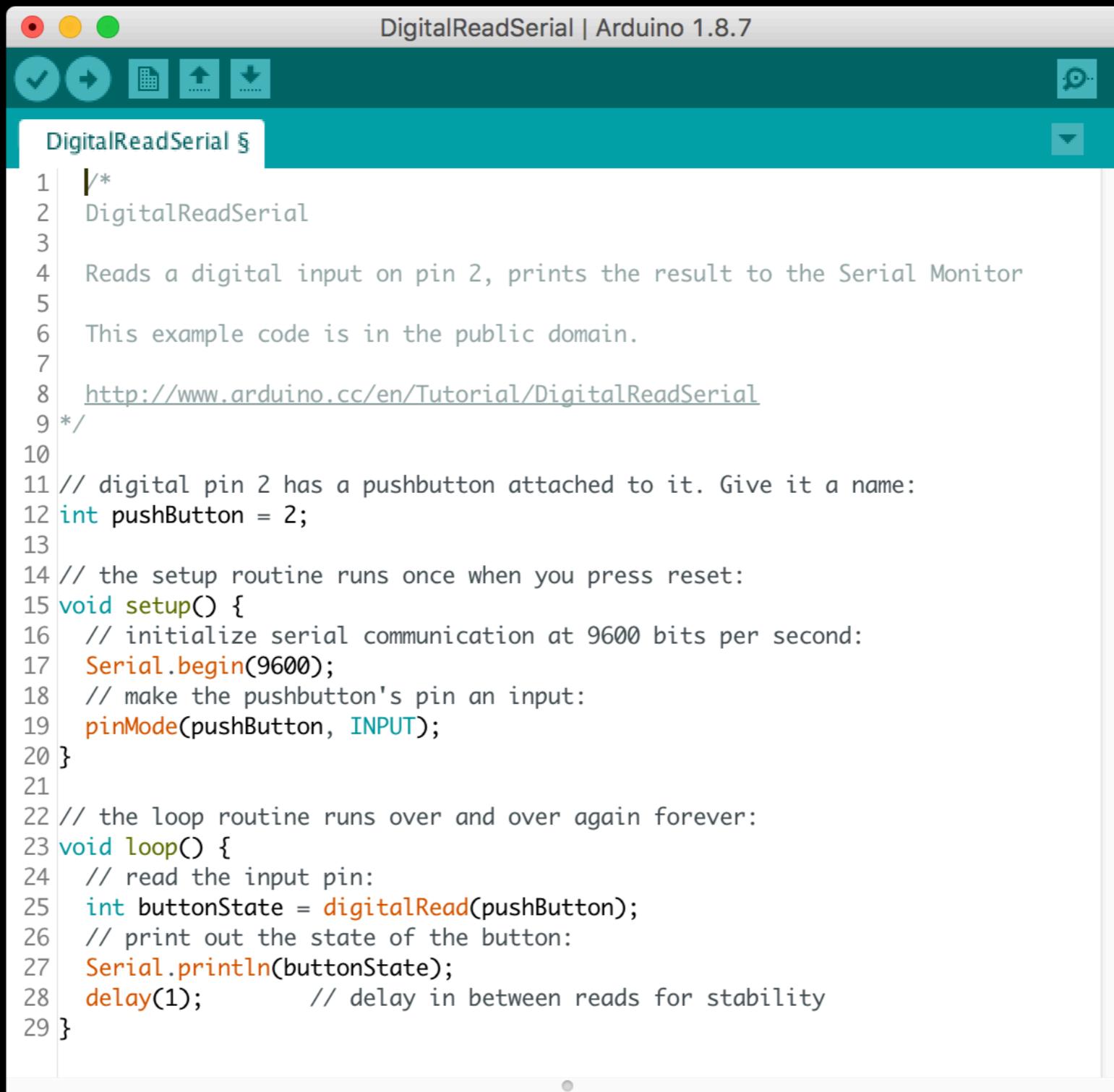
```
void setup( ) { //the code between the curly braces is only run once for setup()
```

# Void Loop( )

- loop() is repeated over and over again
- loop() function is where the body of your program will reside.

```
void loop( ) { //whatever code you put here is executed over and over}
```

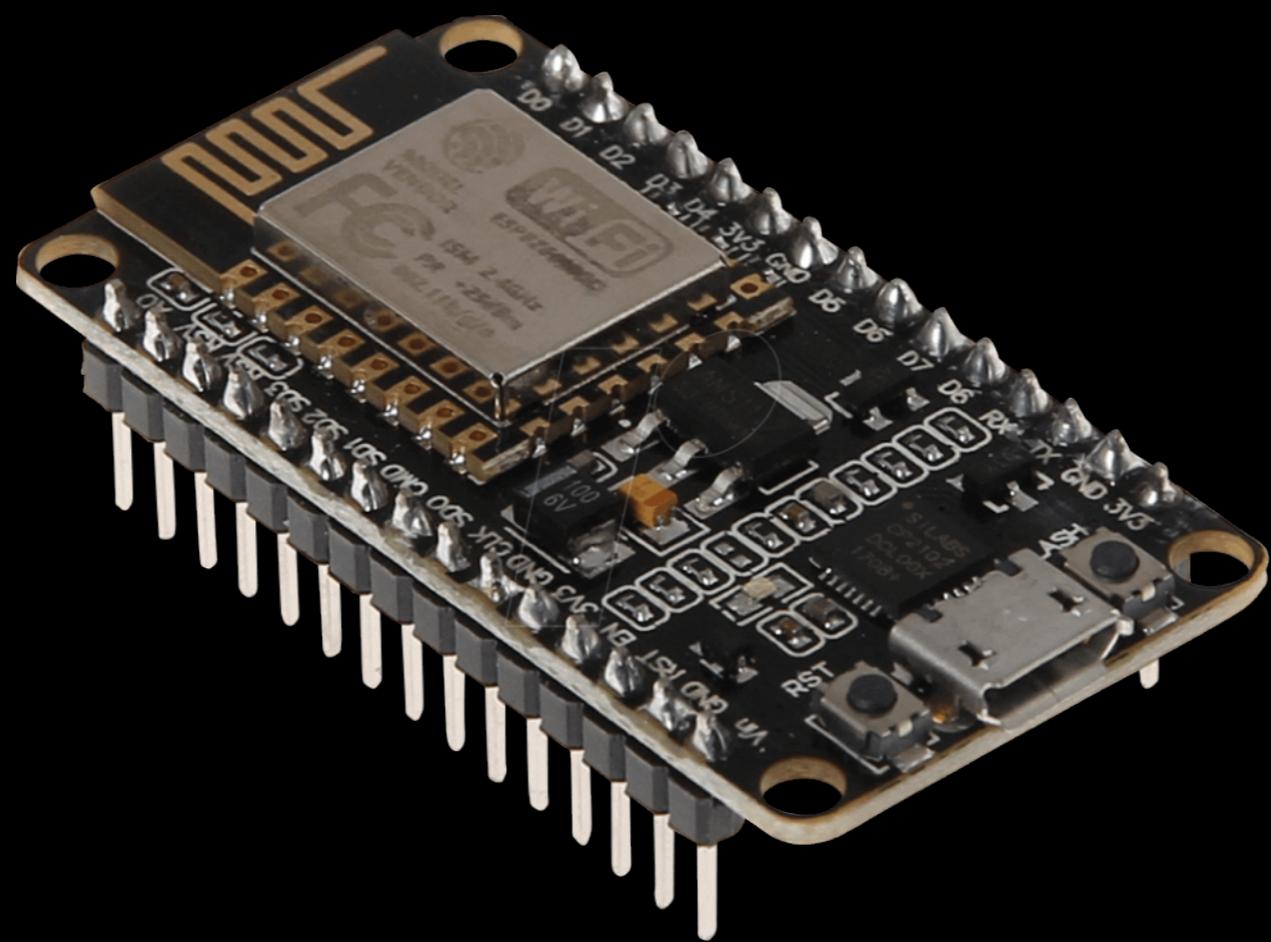
# Example Syntax



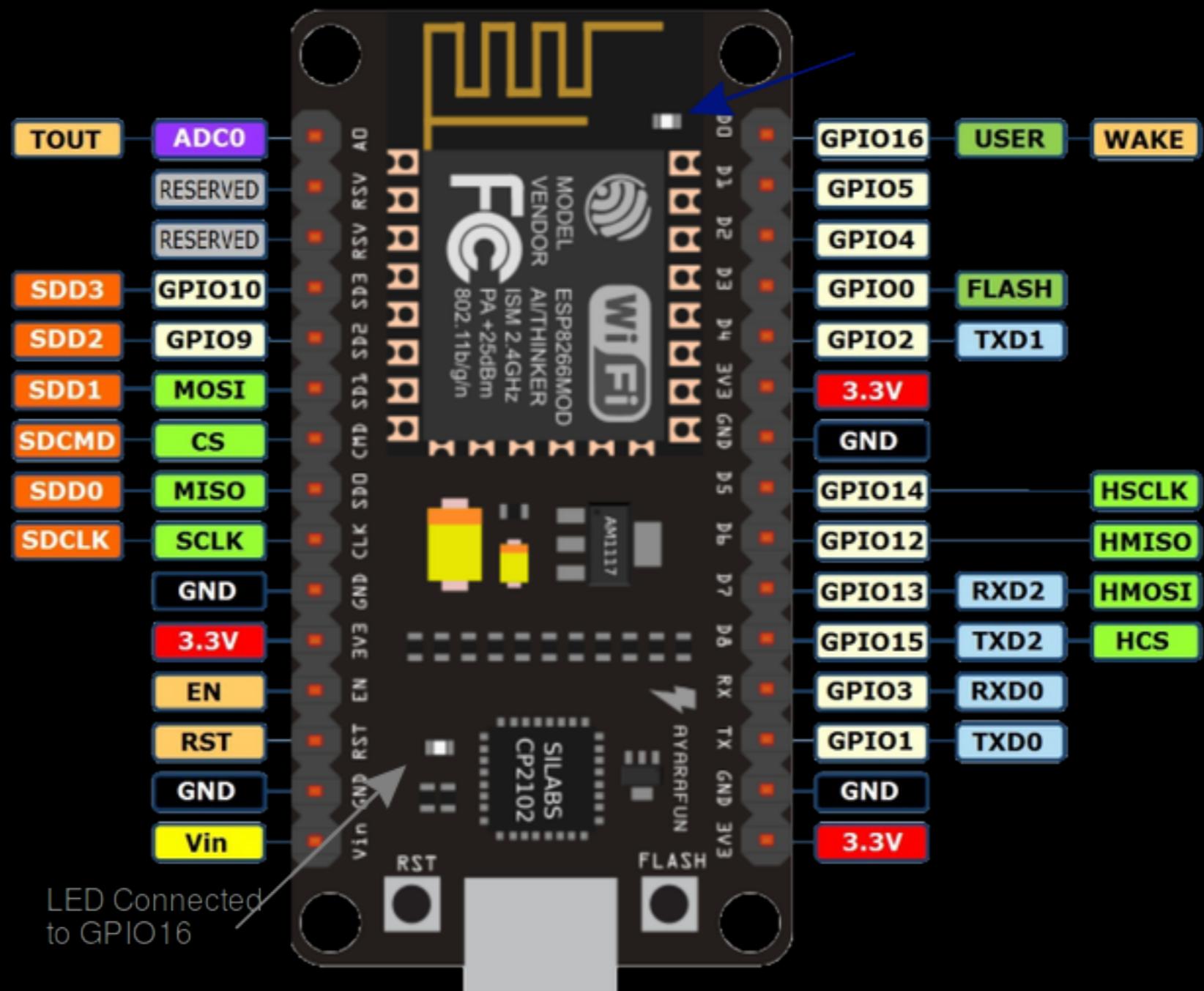
The screenshot shows the Arduino IDE interface with the title bar "DigitalReadSerial | Arduino 1.8.7". Below the title bar is a toolbar with icons for file operations like Open, Save, and Print. The main window displays the "DigitalReadSerial §" sketch. The code is as follows:

```
1  /*
2   * DigitalReadSerial
3
4   * Reads a digital input on pin 2, prints the result to the Serial Monitor
5
6   * This example code is in the public domain.
7
8   * http://www.arduino.cc/en/Tutorial/DigitalReadSerial
9  */
10
11 // digital pin 2 has a pushbutton attached to it. Give it a name:
12 int pushButton = 2;
13
14 // the setup routine runs once when you press reset:
15 void setup() {
16   // initialize serial communication at 9600 bits per second:
17   Serial.begin(9600);
18   // make the pushbutton's pin an input:
19   pinMode(pushButton, INPUT);
20 }
21
22 // the loop routine runs over and over again forever:
23 void loop() {
24   // read the input pin:
25   int buttonState = digitalRead(pushButton);
26   // print out the state of the button:
27   Serial.println(buttonState);
28   delay(1);      // delay in between reads for stability
29 }
```

# ESP8266



# ESP8266 Pinout

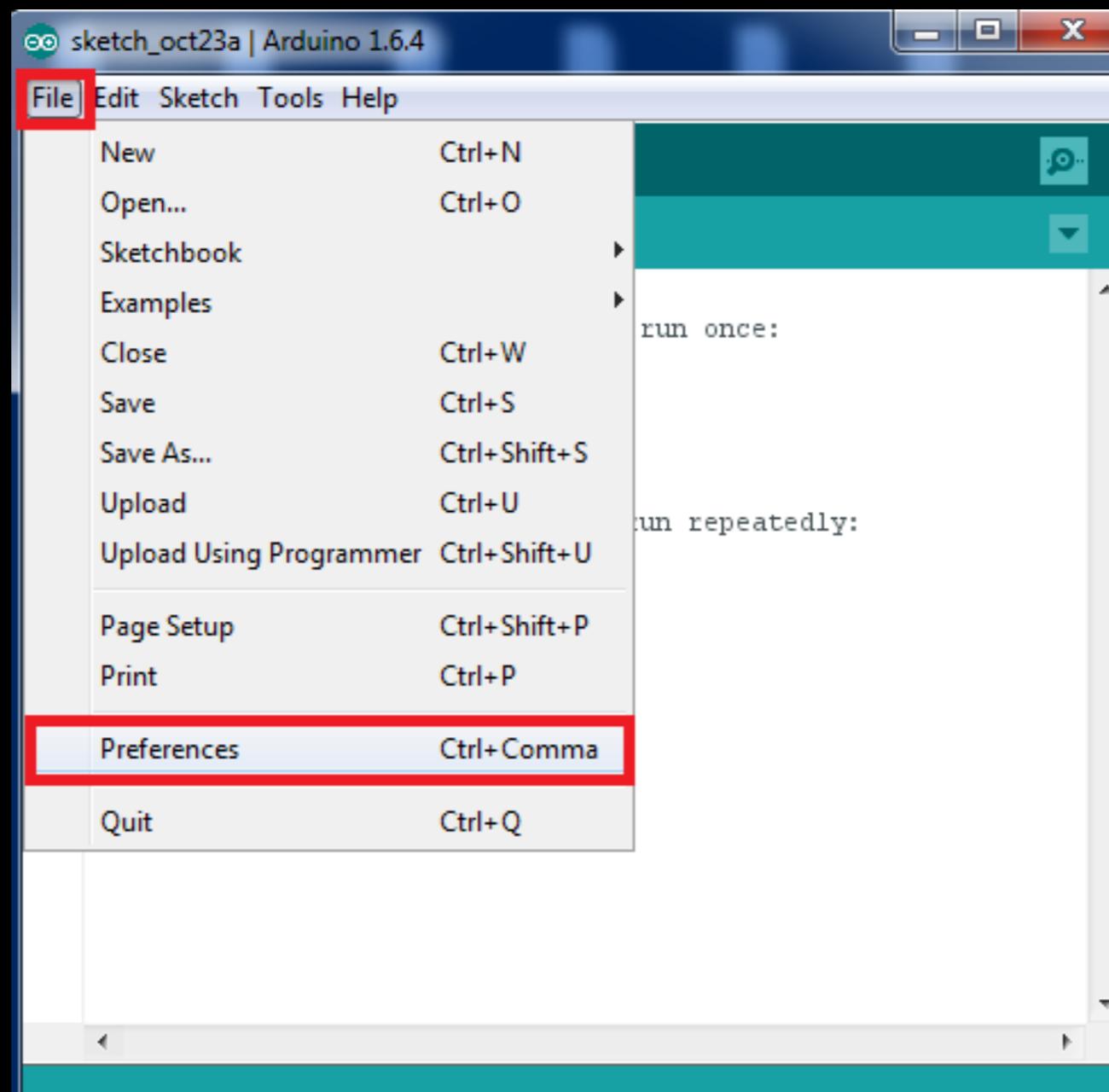


# Installing the Drivers

<https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers>

# Installing the ESP8266 Board

1. Open the preferences window from the Arduino IDE. Go to File > Preferences

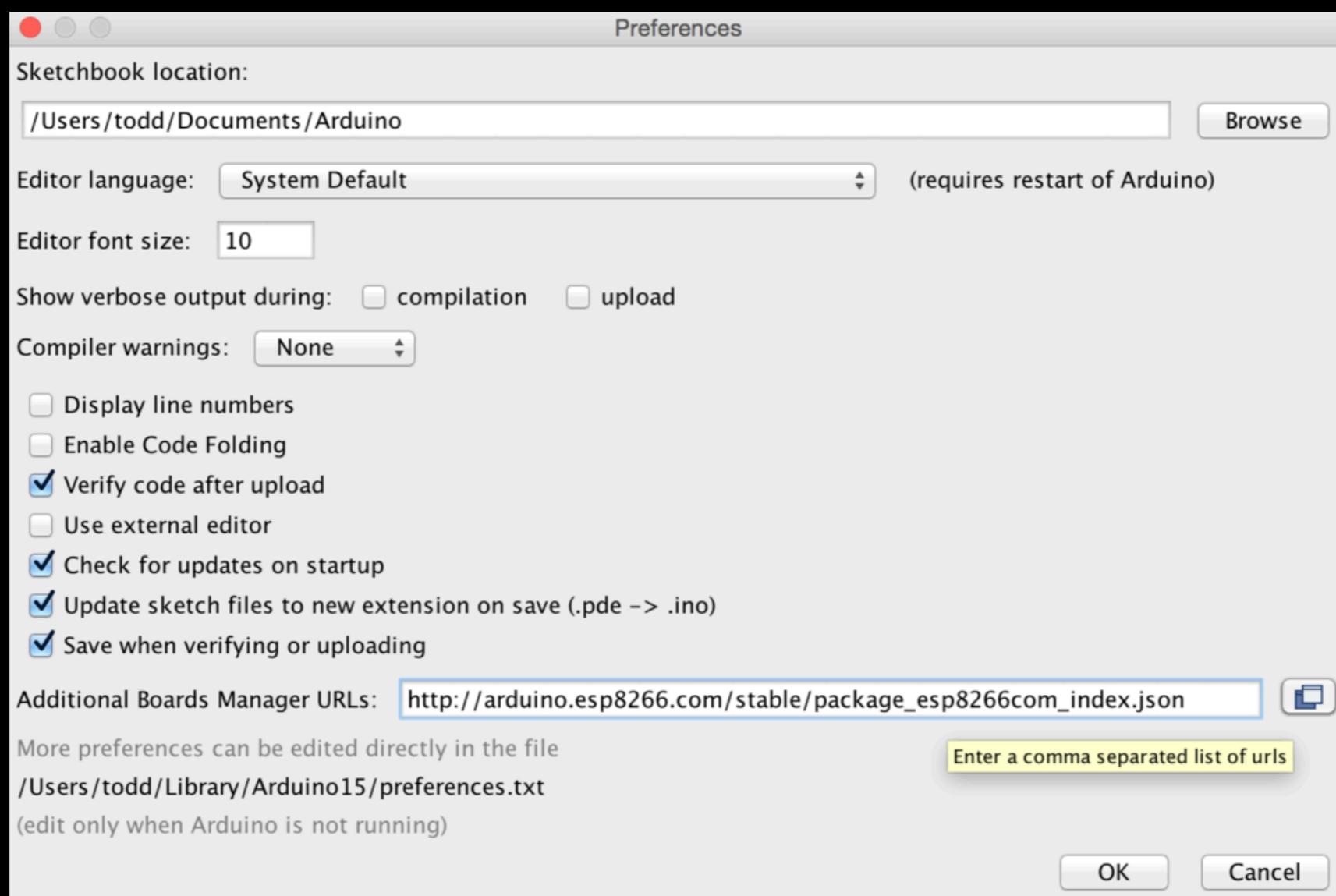


# Installing the ESP8266 Board

2. Enter

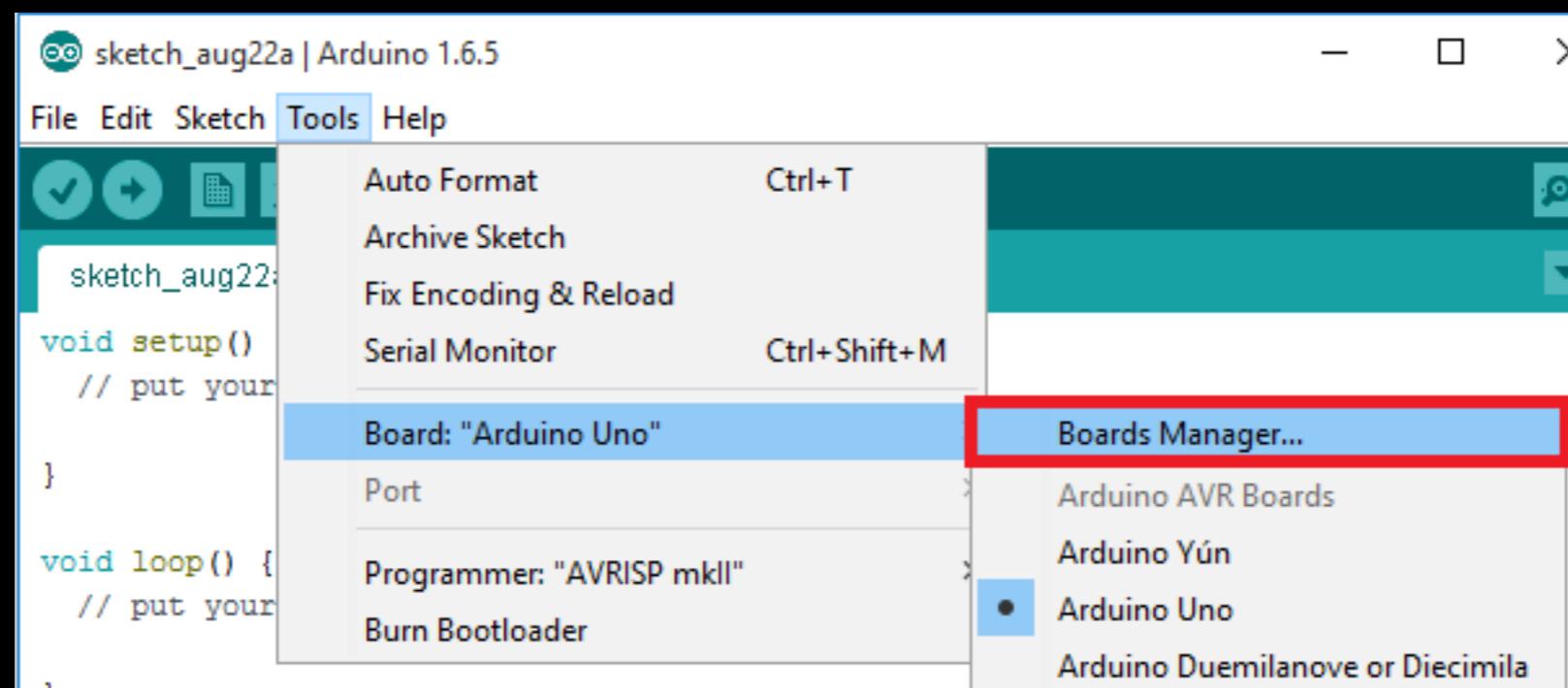
***http://arduino.esp8266.com/stable/package\_esp8266com\_index.json***

into the “Additional Board Manager URLs” field as shown in the figure below. Then, click the “OK” button.



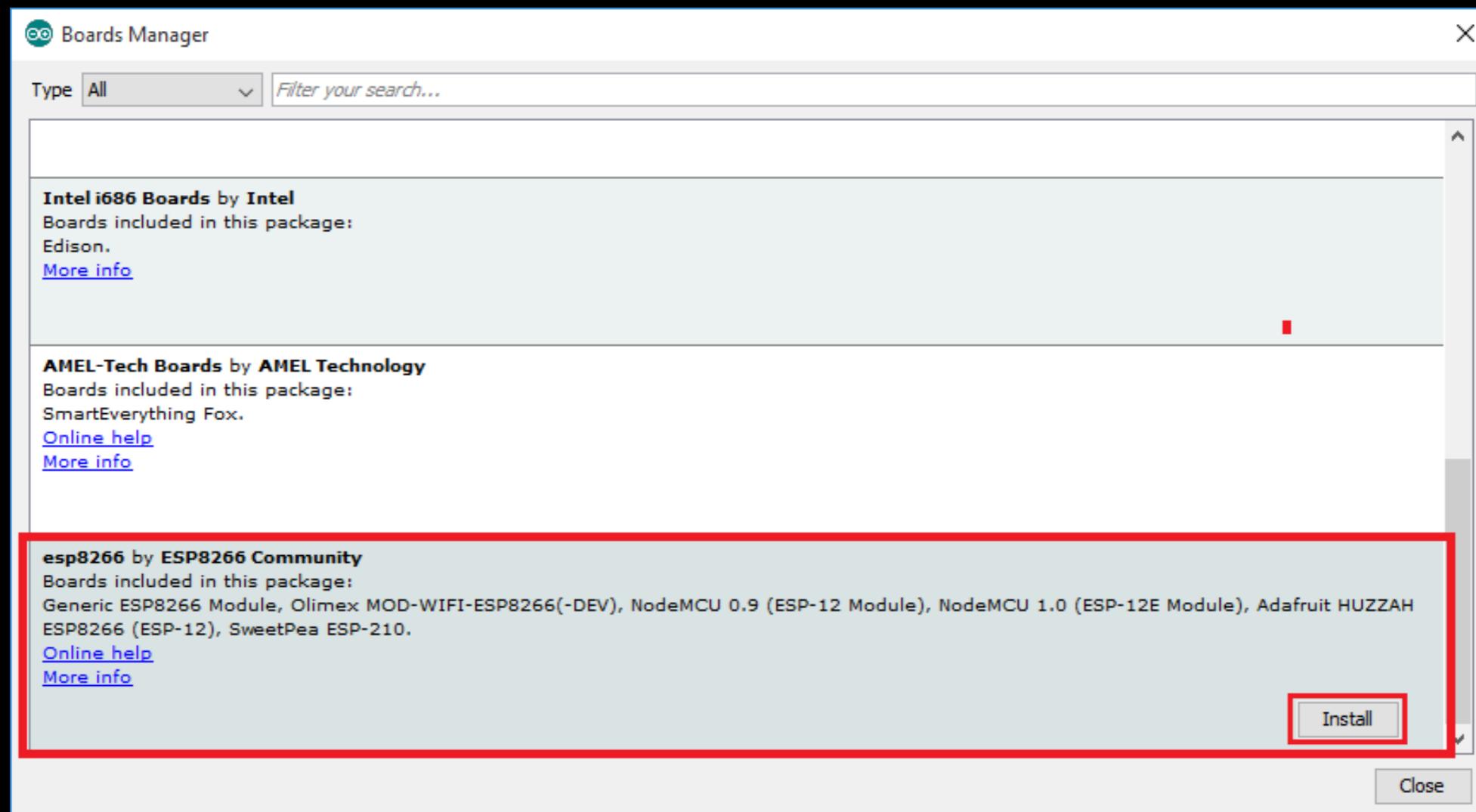
# Installing the ESP8266 Board

3. Open boards manager. Go to Tools > Board > Boards Manager...



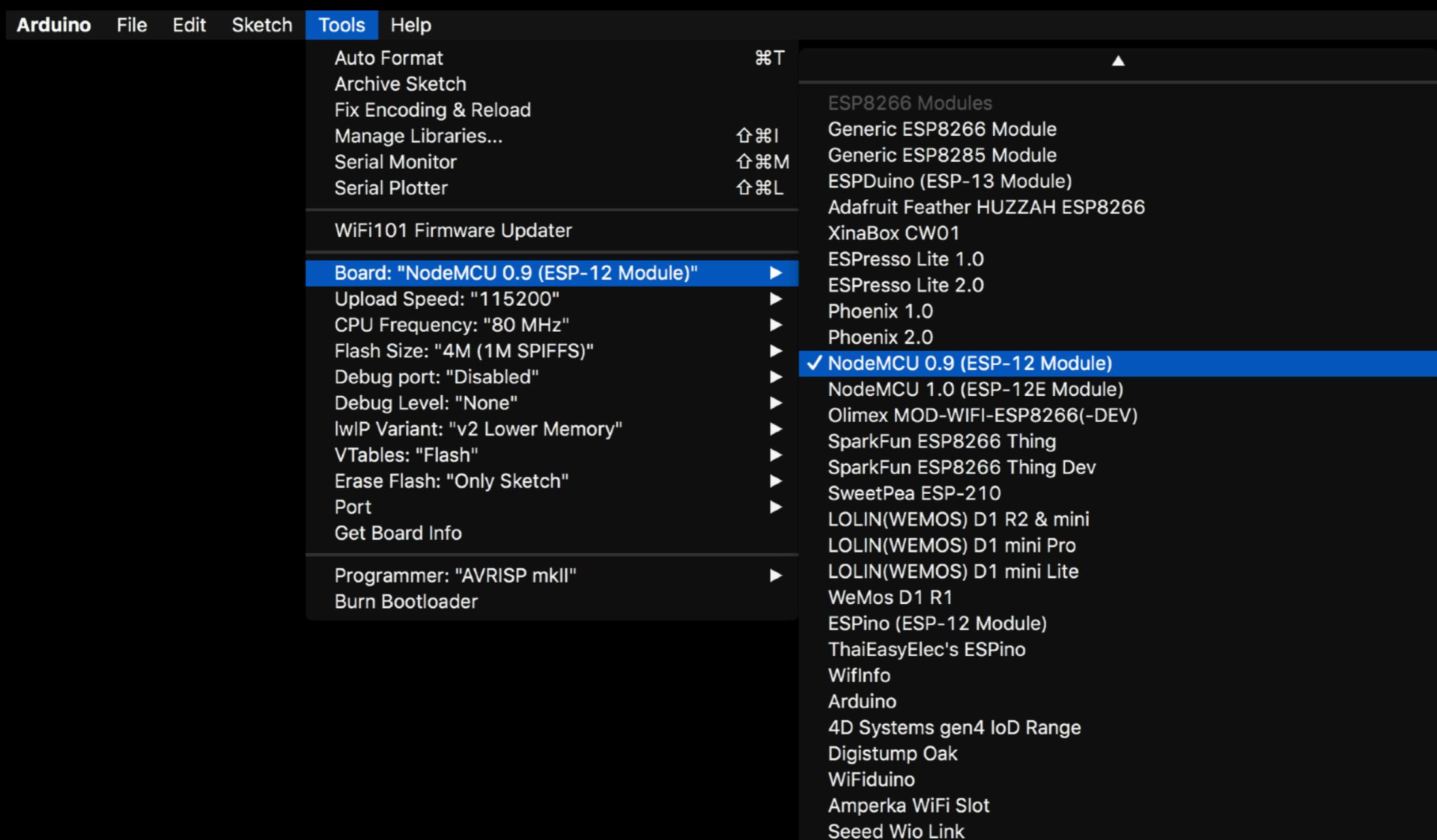
# Installing the ESP8266 Board

4. Scroll down, select the ESP8266 board menu and install “esp8266”



# Installing the ESP8266 Board

5. Choose your ESP8266 board from Tools > Board > NodeMCU 0.9



# Hello World!



Blink | Arduino 1.8.7

Sketch: Blink

```
1 /*
2  * Blink
3  *
4  * Turns an LED on for one second, then off for one second, repeatedly.
5  *
6  * Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  * it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  * the correct LED pin independent of which board is used.
9  * If you want to know what pin the on-board LED is connected to on your Arduino
10 model, check the Technical Specs of your board at:
11 https://www.arduino.cc/en/Main/Products
12
13 modified 8 May 2014
14 by Scott Fitzgerald
15 modified 2 Sep 2016
16 by Arturo Guadalupi
17 modified 8 Sep 2016
18 by Colby Newman
19
20 This example code is in the public domain.
21
22 http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
34     delay(1000);                      // wait for a second
35     digitalWrite(LED_BUILTIN, LOW);       // turn the LED off by making the voltage LOW
36     delay(1000);                      // wait for a second
37 }
```

# 3. Blynk

# IoT System

1. A complete IoT system needs **hardware**
2. A complete IoT system needs **connectivity**
3. A complete IoT system needs **software**.
4. Finally, a complete IoT system needs a user **interface**.

# IoT Platform

1. A complete IoT system needs **hardware**
2. A complete IoT system needs **connectivity**
3. A complete IoT system needs **software**.
4. Finally, a complete IoT system needs a user **interface**.

# IoT Platform

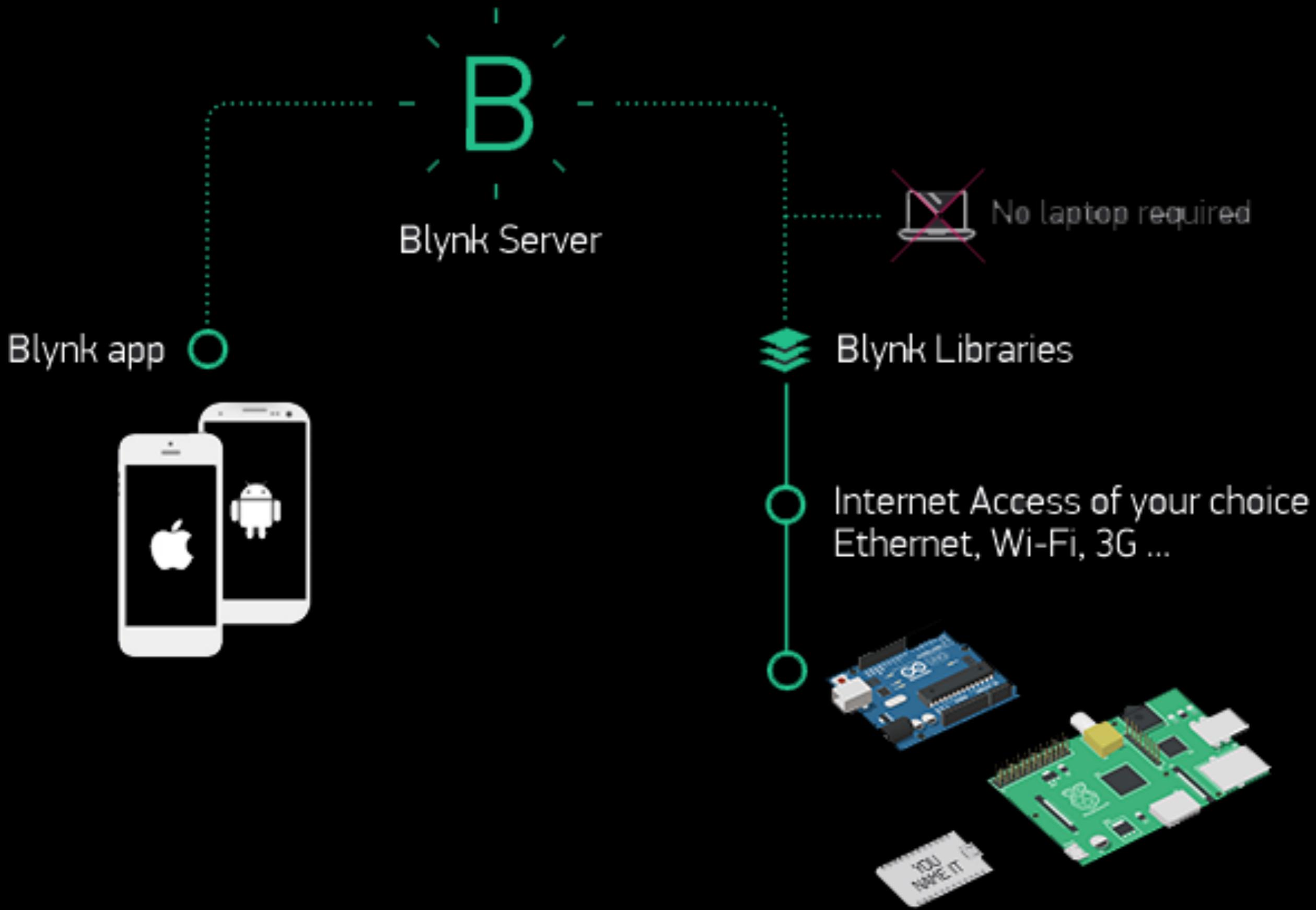
- Connect hardware
- Handle different communication protocols
- Provide security and authentication for devices and users
- Collect, visualize, and analyze data
- Integrate with other web services

# Platforms

1. **PlatformIO** - <https://platformio.org/>
2. **ThingerIO** - <https://thinger.io/>
3. **Oracle Cloud IoT** - <https://cloud.oracle.com/iot>
4. **Azure IoT Hub** - [https://azure.microsoft.com/en-us/  
services/iot-hub/](https://azure.microsoft.com/en-us/services/iot-hub/)
5. **IBM IoT Watson** - <https://www.ibm.com/internet-of-thing>
6. **The Samsung ARTIK** - <https://www.artik.io/>

# Blynk





# Blynk Documentation

The screenshot shows the Blynk Documentation website. On the left, a dark sidebar contains a navigation menu:

- Blynk**
- INTRO**
  - [How Blynk Works](#)
  - Features
  - What do I need to Blynk?
- DOWNLOADS**
  - Blynk Apps for iOS or Android
  - Blynk Library
- GETTING STARTED**
  - Getting Started With The Blynk App
    - 1. Create a Blynk Account
    - 2. Create a New Project
    - 3. Choose Your Hardware
    - 4. Auth Token
    - 5. Add a Widget
    - 6. Run The Project
  - Getting Started With Hardware
    - How To Use an Example Sketch
    - Auth Token
    - Blynking
- HARDWARE SET-UPS**
  - Arduino over USB (no shield)
  - Raspberry Pi
  - ESP8266 Standalone
  - NodeMCU
  - Arduino + ESP8266 WiFi with AT commands
  - Particle
- BLYNK MAIN OPERATIONS**
  - Virtual Pins
  - Send data from app to hardware
    - Sending array from Widget
  - Get data from hardware

## Intro

This guide will help you understand how to get started using Blynk and give a comprehensive overview of all the features.

If you want to jump straight into playing with Blynk, check out Getting Started.

[GETTING STARTED](#)

## How Blynk Works

Blynk was designed for the Internet of Things. It can control hardware remotely, it can display sensor data, it can store data, visualize it and do many other cool things.

There are three major components in the platform:

- Blynk App - allows to you create amazing interfaces for your projects using various widgets we provide.
- Blynk Server - responsible for all the communications between the smartphone and hardware. You can use our Blynk Cloud or run your [private Blynk server](#) locally. It's open-source, could easily handle thousands of devices and can even be launched on a Raspberry Pi.
- Blynk Libraries - for all the popular hardware platforms - enable communication with the server and process all the incoming and outgoing commands.

Now imagine: every time you press a Button in the Blynk app, the message travels to space the Blynk Cloud, where it magically finds its way to your hardware. It works the same in the opposite direction and everything happens in a blink of an eye.

# Steps Arduino

## 1. Install Arduino Software

<http://www.arduino.org/software>

## 2. Download Blynk Library

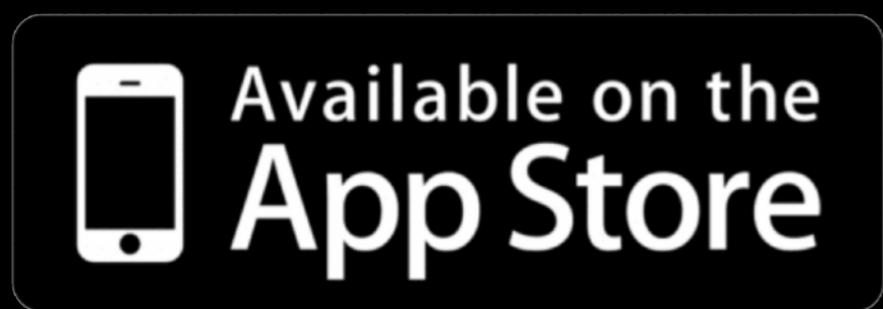
[https://github.com/blynkkk/blynk-library/releases/  
latest](https://github.com/blynkkk/blynk-library/releases/latest)

## 3. Install Library

<http://www.arduino.cc/en/guide/libraries>

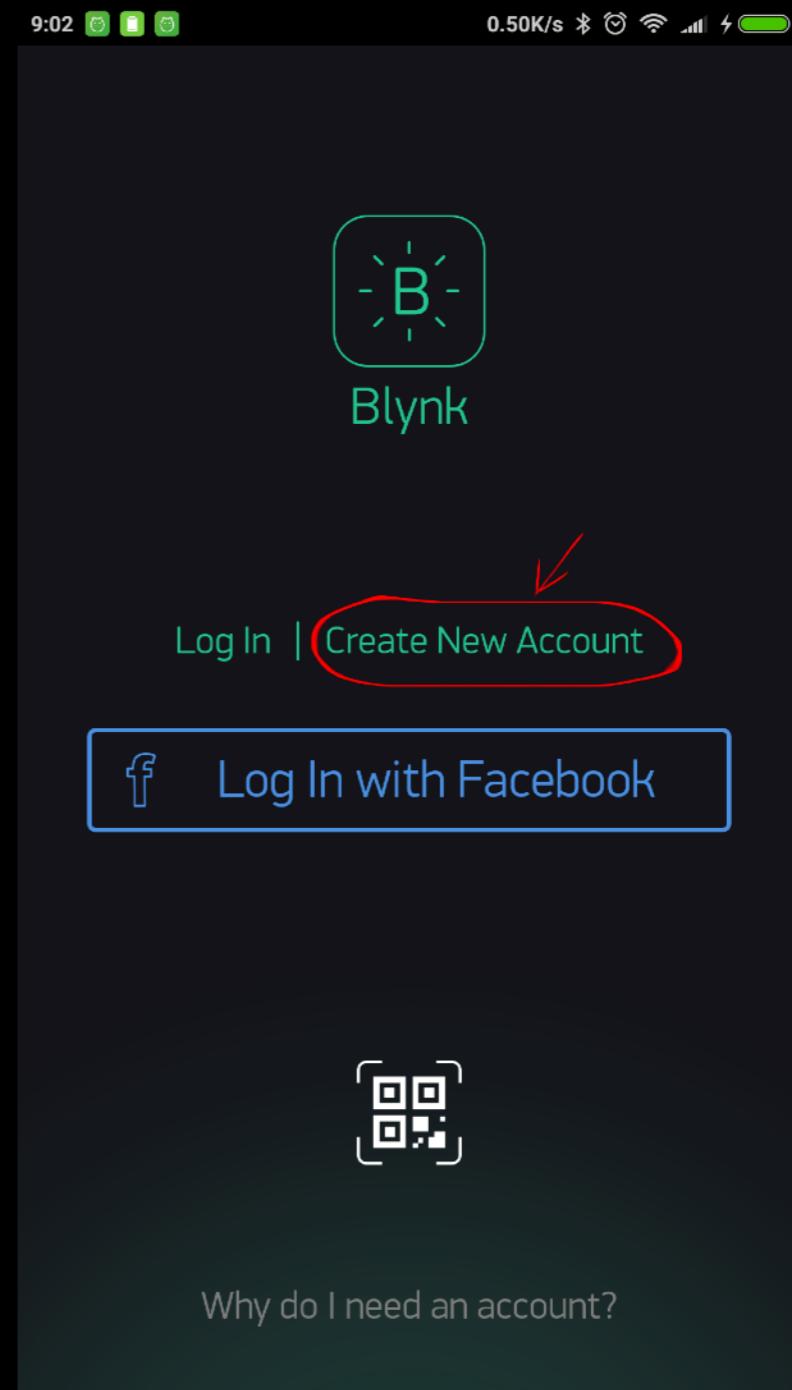
# Steps Blynk

1. Install Blynk App for iOS or Android.



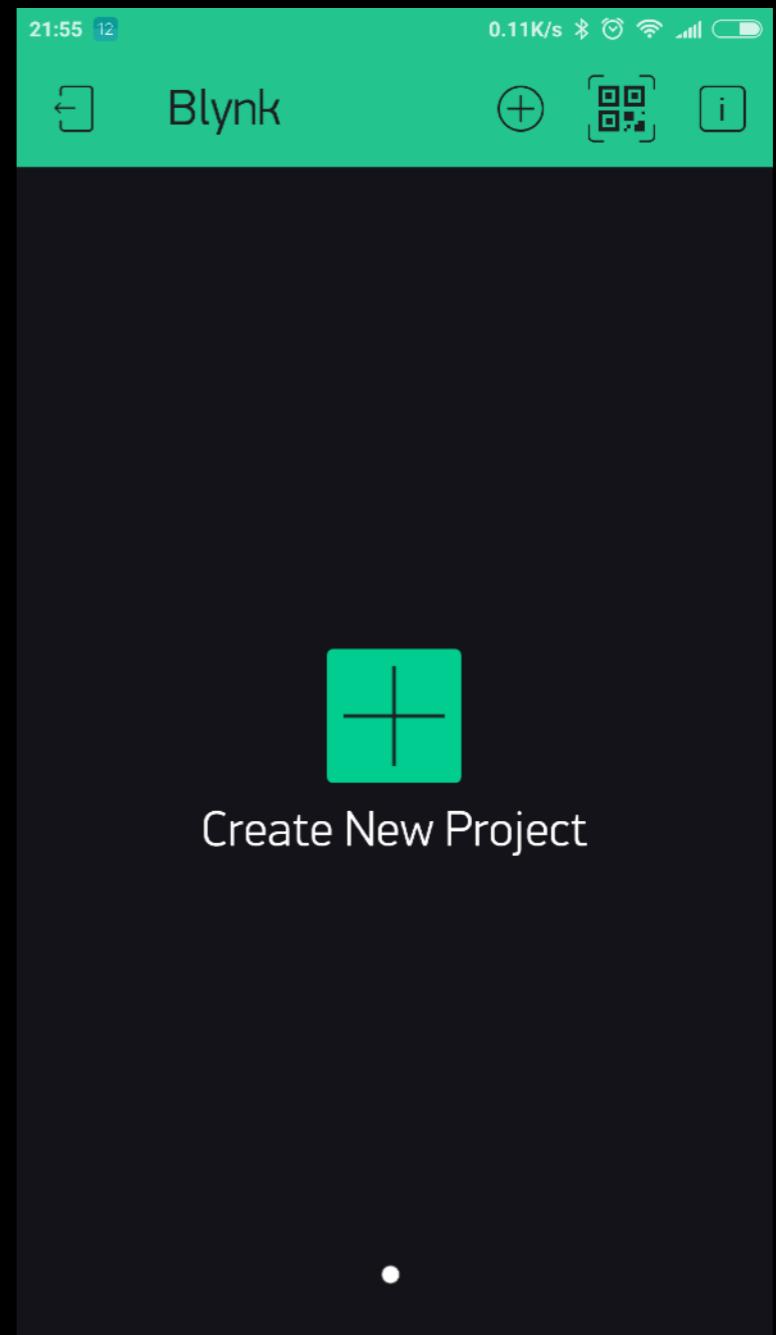
# Steps Blynk

## 2 . Create Blynk Account



# Steps Blynk

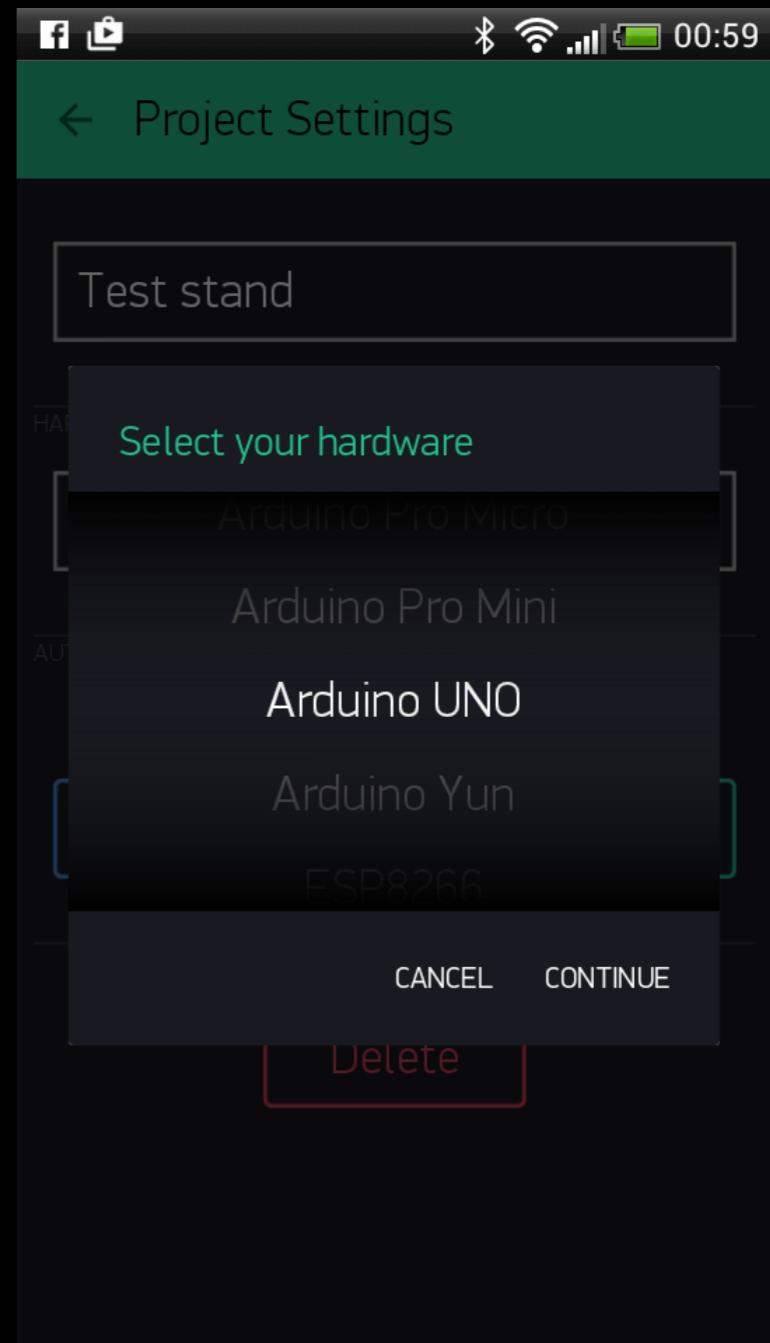
3 . Create a New Project



# Steps Blynk

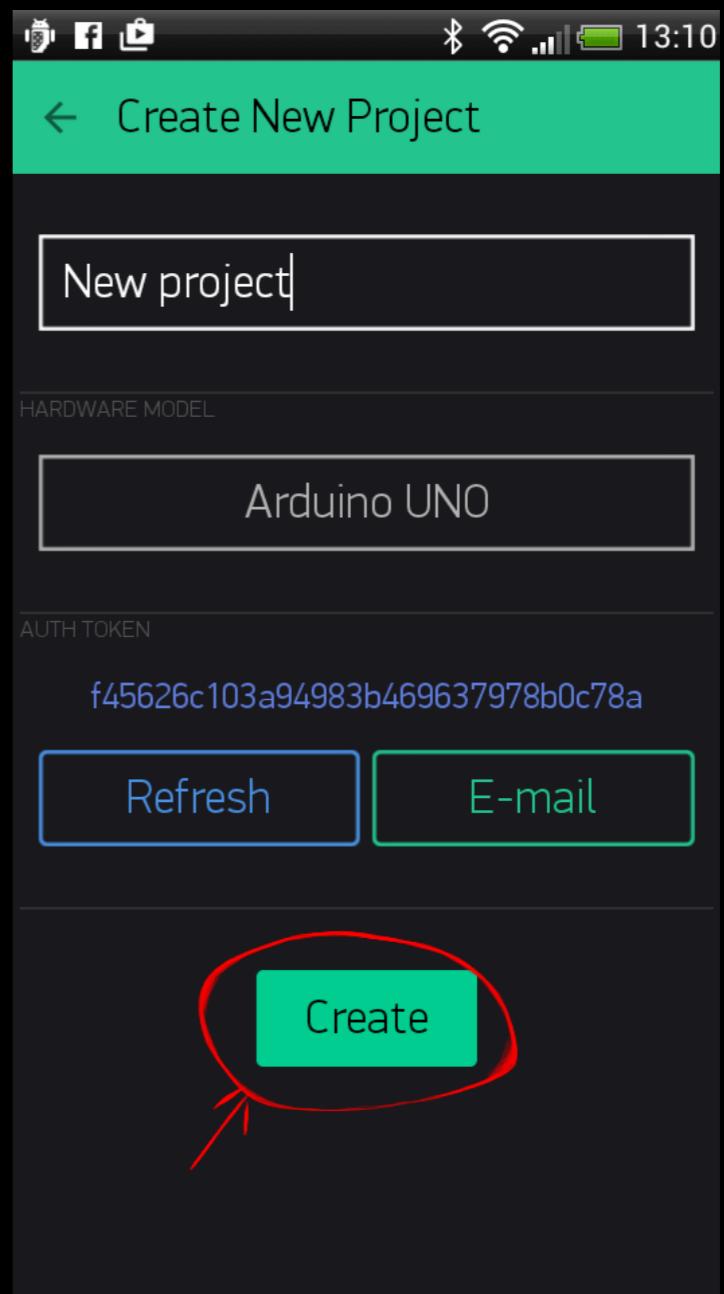
4 . Choose your hardware:

**ESP8266**



# Steps Blynk

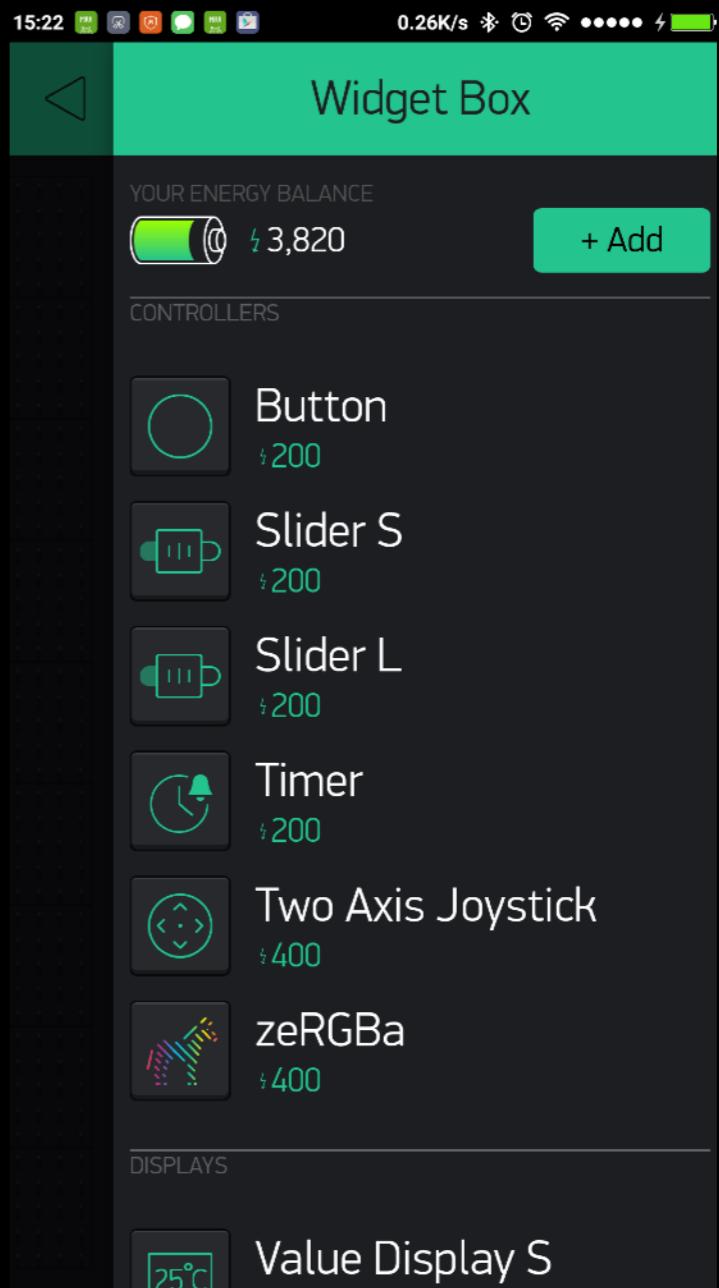
## 5 . Auth Token



# Steps Blynk

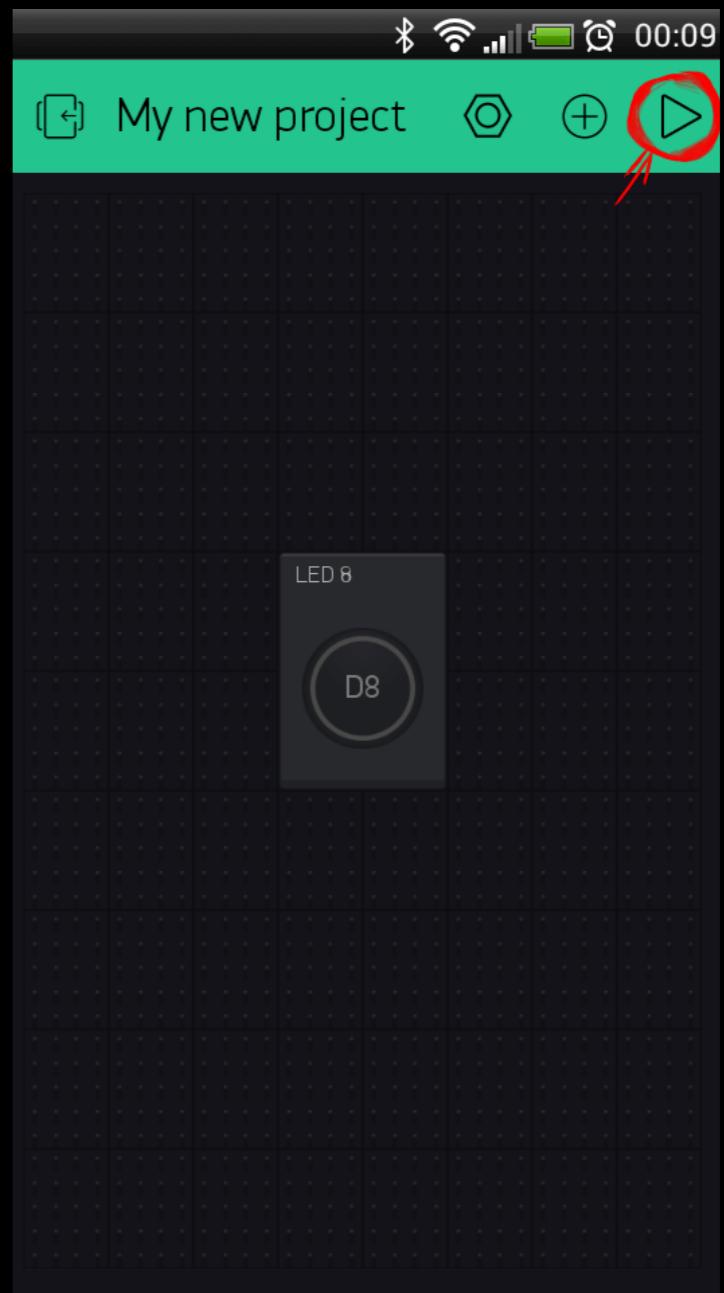
## 6 . Add a Widget

1. Drag-n-Drop Tap and hold the Button Widget to drag it to the new position.
2. Widget Settings Each Widget has it's own settings. Tap on the widget to get to them.
3. Set PIN to Digital -> D2 or GP2



# Steps Blynk

7 . Run The Project



# Steps Code

## 1. Sketch Code Builder (<https://examples.blynk.cc>)

1. Choose Your board: ESP8266
2. Choose your connection: ESP8266 WiFi

The image shows a screenshot of the Blynk Sketch Code Builder. On the left, there's a green sidebar with configuration options: Board (ESP8266), Connection (ESP8266 WiFi), Auth Token (optional), and Example (Blynk Blink). The main area displays the generated sketch code:

```
/*
Download latest Blynk library here:
https://github.com/blynkkk/blynk-library/releases/latest

Blynk is a platform with iOS and Android apps to control
Arduino, Raspberry Pi and the likes over the Internet.
You can easily build graphic interfaces for all your
projects by simply dragging and dropping widgets.

Downloads, docs, tutorials: http://www.blynk.cc
Sketch generator: http://examples.blynk.cc
Blynk community: http://community.blynk.cc
Follow us: http://www.fb.com/blynkapp
http://twitter.com/blynk\_app

Blynk library is licensed under MIT license
This example code is in public domain.

You'll need:
- Blynk App (download from AppStore or Google Play)
- ESP8266 board
- Decide how to connect to Blynk
  (USB, Ethernet, Wi-Fi, Bluetooth, ...)

There is a bunch of great example sketches included to show you how to get
started. Think of them as LEGO bricks and combine them as you wish.
For example, take the Ethernet Shield sketch and combine it with the
Servo example, or choose a USB sketch and add a code from SendData
example.

*/
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

void setup()
{
    // Debug console
    Serial.begin(9600);

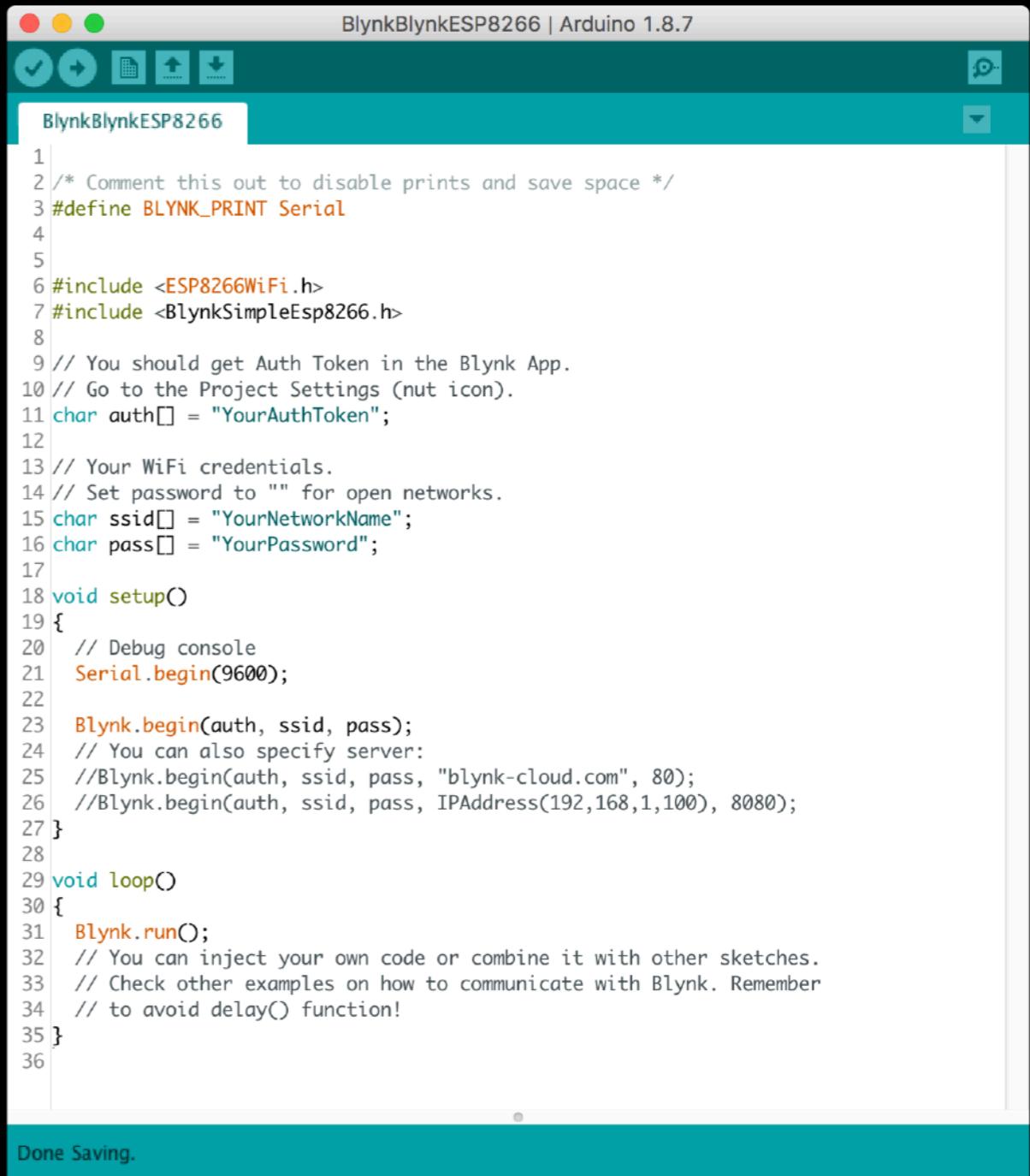
    Blynk.begin(auth, ssid, pass);
    // You can also specify server:
    //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
    //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
}

void loop()
{
    Blynk.run();
    // You can inject your own code or combine it with other sketches.
    // Check other examples on how to communicate with Blynk. Remember
    // to avoid delay() function!
}
```

# Steps Code

## 2 . Create new sketch

1. Copy example code
2. Auth Token
3. WiFi Credentials



The screenshot shows the Arduino IDE interface with the title bar "BlynkBlynkESP8266 | Arduino 1.8.7". The main window displays the following C++ code for a Blynk ESP8266 sketch:

```
1  /* Comment this out to disable prints and save space */
2  #define BLYNK_PRINT Serial
3
4
5
6 #include <ESP8266WiFi.h>
7 #include <BlynkSimpleEsp8266.h>
8
9 // You should get Auth Token in the Blynk App.
10 // Go to the Project Settings (nut icon).
11 char auth[] = "YourAuthToken";
12
13 // Your WiFi credentials.
14 // Set password to "" for open networks.
15 char ssid[] = "YourNetworkName";
16 char pass[] = "YourPassword";
17
18 void setup()
19 {
20     // Debug console
21     Serial.begin(9600);
22
23     Blynk.begin(auth, ssid, pass);
24     // You can also specify server:
25     //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
26     //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
27 }
28
29 void loop()
30 {
31     Blynk.run();
32     // You can inject your own code or combine it with other sketches.
33     // Check other examples on how to communicate with Blynk. Remember
34     // to avoid delay() function!
35 }
36
```

At the bottom of the code editor, a message "Done Saving." is displayed.

# Steps Code

## 3 . Upload Sketch

1. Open serial monitor in Arduino IDE.

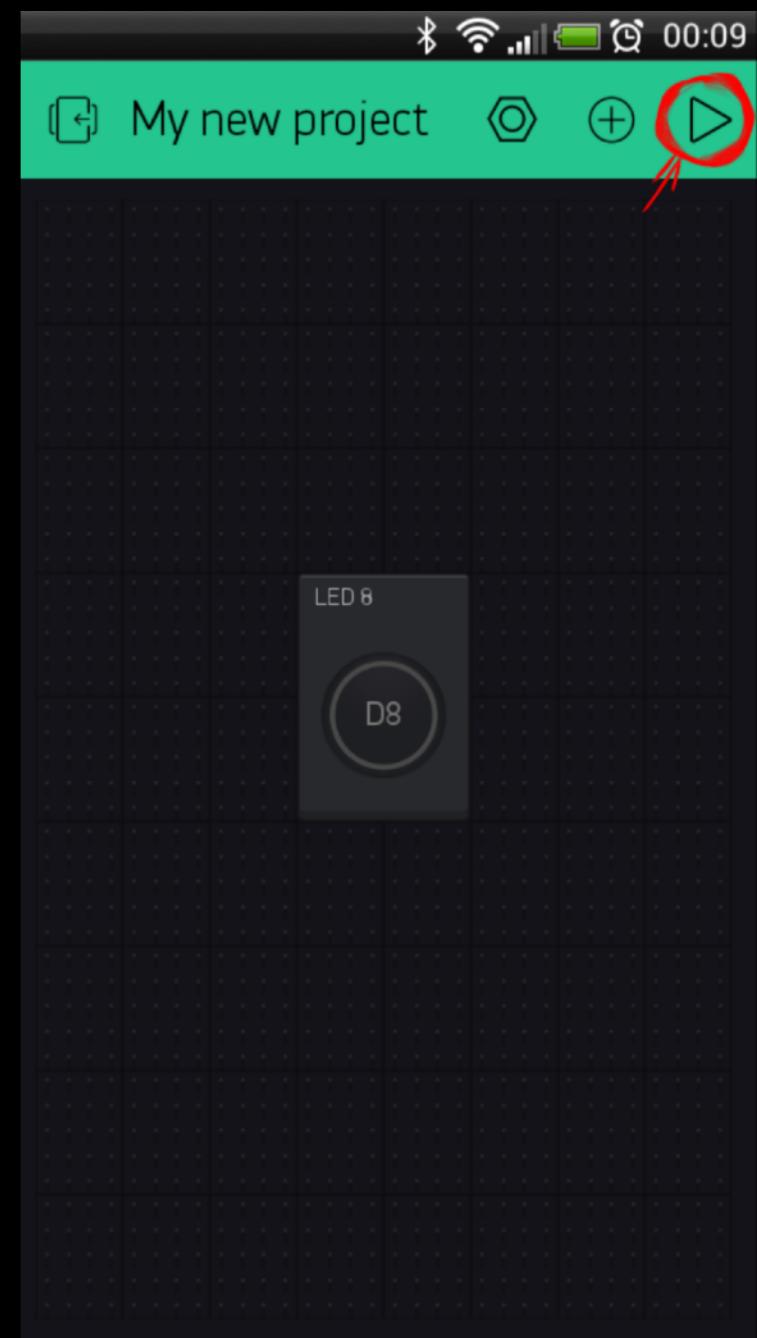


```
1  /* Comment this out to disable prints and save space */
2  #define BLYNK_PRINT Serial
3
4
5
6 #include <ESP8266WiFi.h>
7 #include <BlynkSimpleEsp8266.h>
8
9 // You should get Auth Token in the Blynk App.
10 // Go to the Project Settings (nut icon).
11 char auth[] = "YourAuthToken";
12
13 // Your WiFi credentials.
14 // Set password to "" for open networks.
15 char ssid[] = "YourNetworkName";
16 char pass[] = "YourPassword";
17
18 void setup()
19 {
20     // Debug console
21     Serial.begin(9600);
22
23     Blynk.begin(auth, ssid, pass);
24     // You can also specify server:
25     //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
26     //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
27 }
28
29 void loop()
30 {
31     Blynk.run();
32     // You can inject your own code or combine it with other sketches.
33     // Check other examples on how to communicate with Blynk. Remember
34     // to avoid delay() function!
35 }
36
```

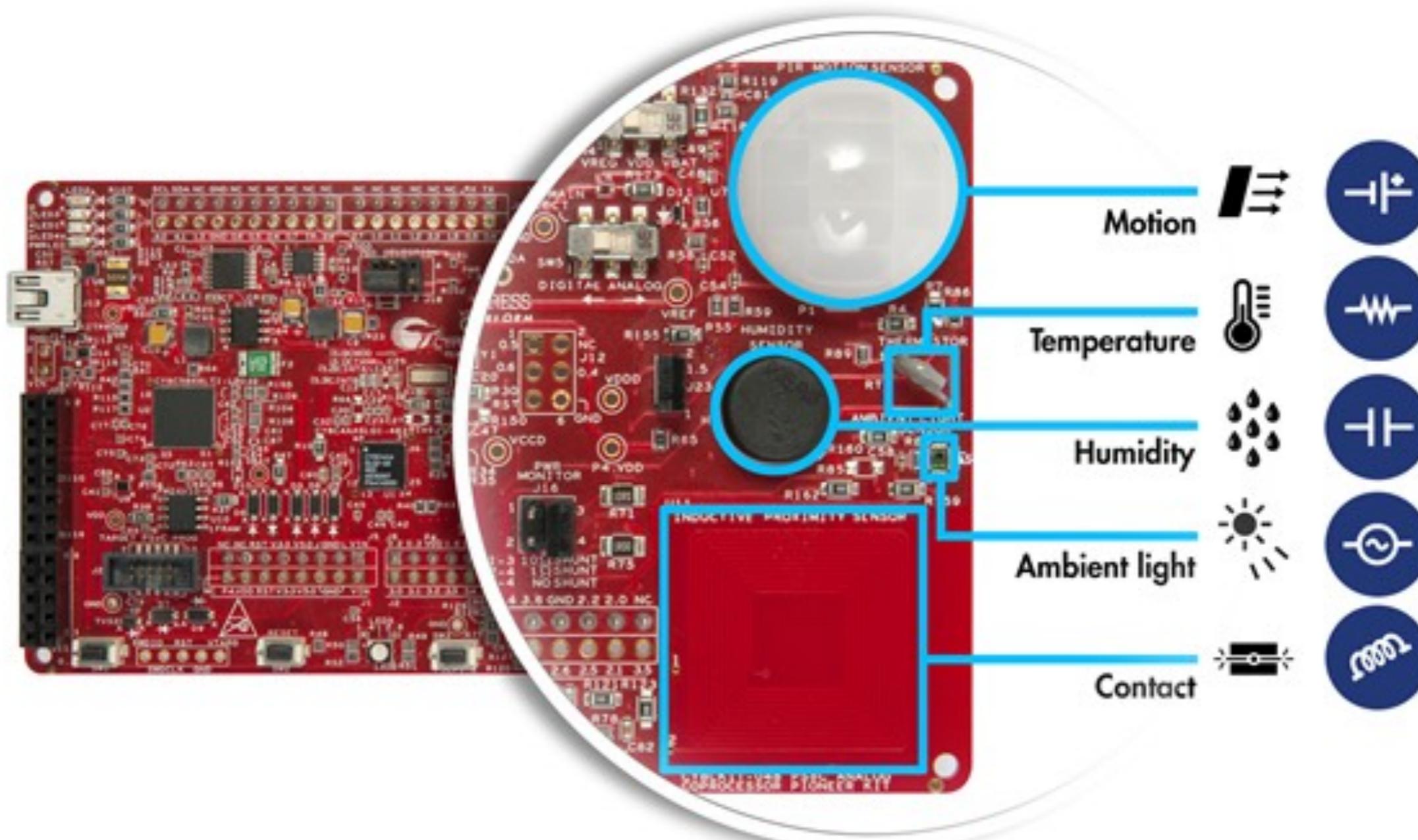
Compiling sketch...

# Steps Code

4 . Run the Blynk project

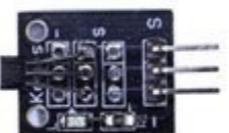


# 4. Sensors





Finger-Pulse sensor Module



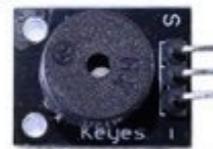
Hall sensor Module



Microphone sensor Module



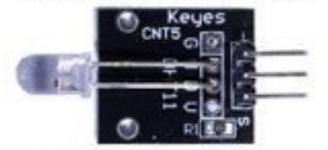
Obstacle avoidance sensor Module



Passive Buzzer Module



Flame sensor Module



Colorful Auto-flash Module



infrared-transmit Module



common-cathode RED&GREEN LED Module



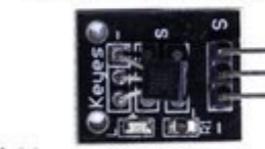
Linear-Hall Sensor Module



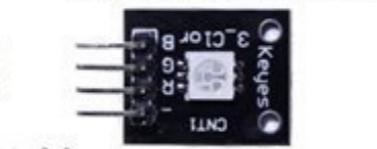
Active buzzer Module



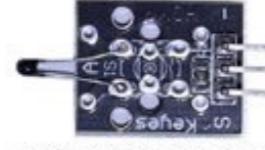
Magnet-ring sensor Module



18b20 temperature sensor Module



RGB LED Module



Analog-temperature sensor Module



Magic-ring Module



tilt-switch Module



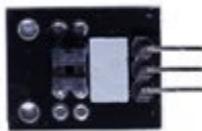
Shock-switch sensor Module



Rotate-encode Module



Relay Module



Light break sensor Module



Magnetic spring Module



Digital-Temperature sensor Module



High-sensitive voice sensor Module



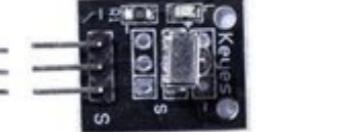
Photo resistor sensor Module



Metal touch sensor Module



hydrgyrum-switch sensor Module



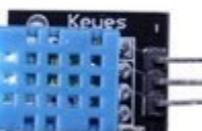
Infrared-receive sensor Module



RGB LED Module



Push button Module



humiture sensor Module



Laser-transmit Module



two-color commoncathode LED Module



Knock sensor module



Tracking sensor Module

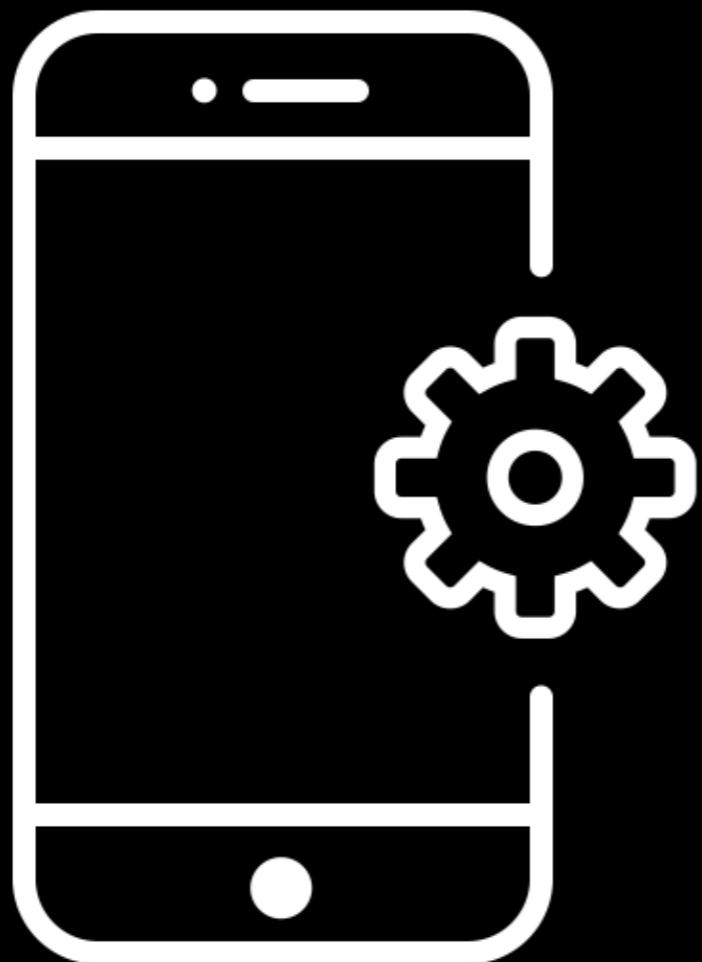


Analogy-Hall sensor Module



Joystick PS2 Module

# Smartphone Sensors



- Accelerometer
- Gravity
- Barometer/Pressure
- Humidity
- Light
- Proximity
- Temperature
- GPS

# Libelium Smart World

## Air Pollution

Control of CO<sub>2</sub> emissions of factories, pollution emitted by cars and toxic gases generated in farms.

## Forest Fire Detection

Monitoring of combustion gases and preemptive fire conditions to define alert zones.

## Wine Quality Enhancing

Monitoring soil moisture and trunk diameter in vineyards to control the amount of sugar in grapes and grapevine health.

## Offspring Care

Control of growing conditions of the offspring in animal farms to ensure its survival and health.

## Sportsmen Care

Vital signs monitoring in high performance centers and fields.

## Structural Health

Monitoring of vibrations and material conditions in buildings, bridges and historical monuments.

## Quality of Shipment Conditions

Monitoring of vibrations, strokes, container openings or cold chain maintenance for insurance purposes.

## Smartphones Detection

Detect iPhone and Android devices and in general any device which works with WiFi or Bluetooth interfaces.

## Perimeter Access Control

Access control to restricted areas and detection of people in non-authorized areas.

## Radiation Levels

Distributed measurement of radiation levels in nuclear power stations surroundings to generate leakage alerts.

## Electromagnetic Levels

Measurement of the energy radiated by cell stations and WiFi routers.

## Traffic Congestion

Monitoring of vehicles and pedestrian affluence to optimize driving and walking routes.

## Water Quality

Study of water suitability in rivers and the sea for fauna and eligibility for drinkable use.

## Smart Roads

Warning messages and diversions according to climate conditions and unexpected events like accidents or traffic jams.

## Smart Lighting

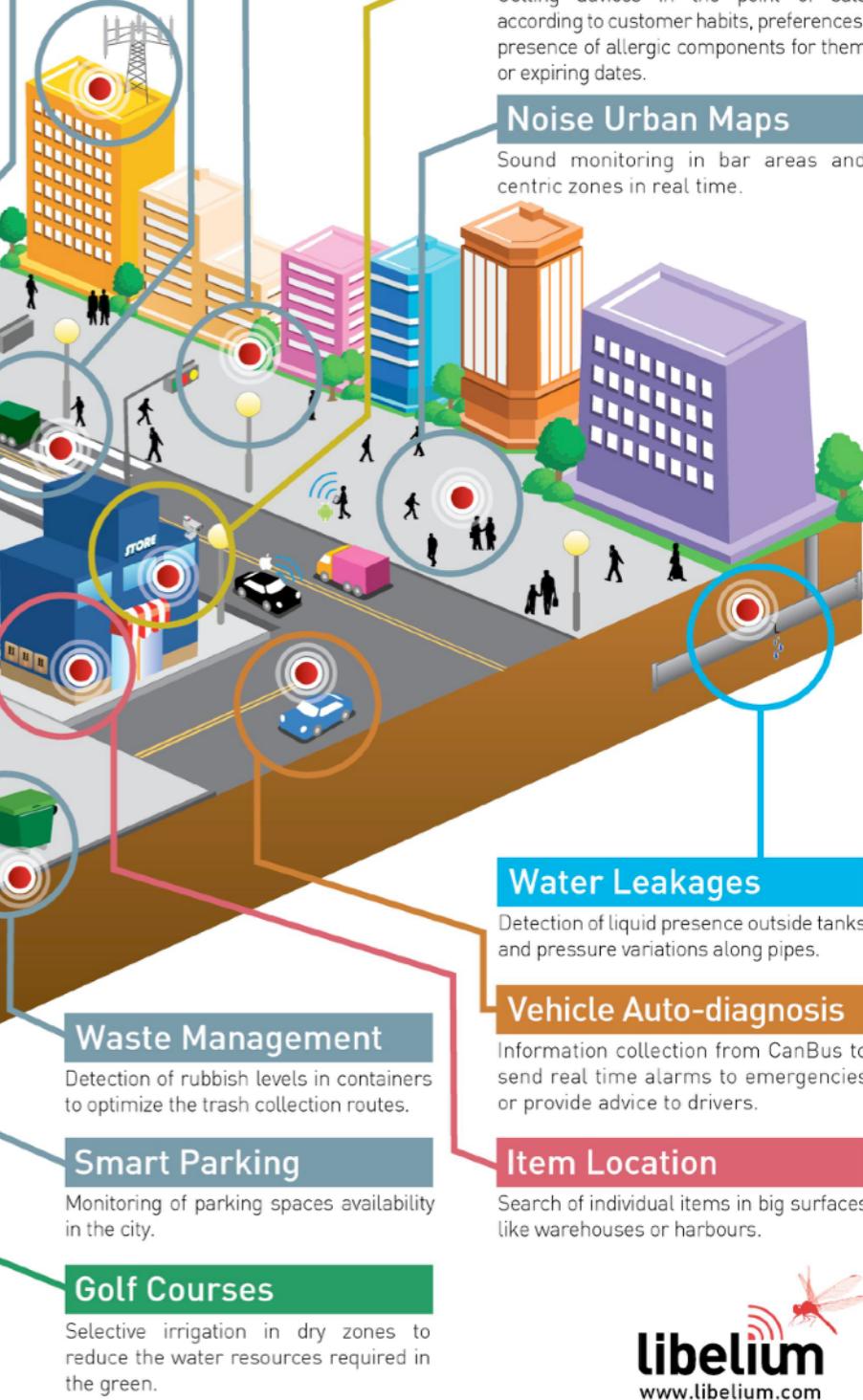
Intelligent and weather adaptive lighting in street lights.

## Intelligent Shopping

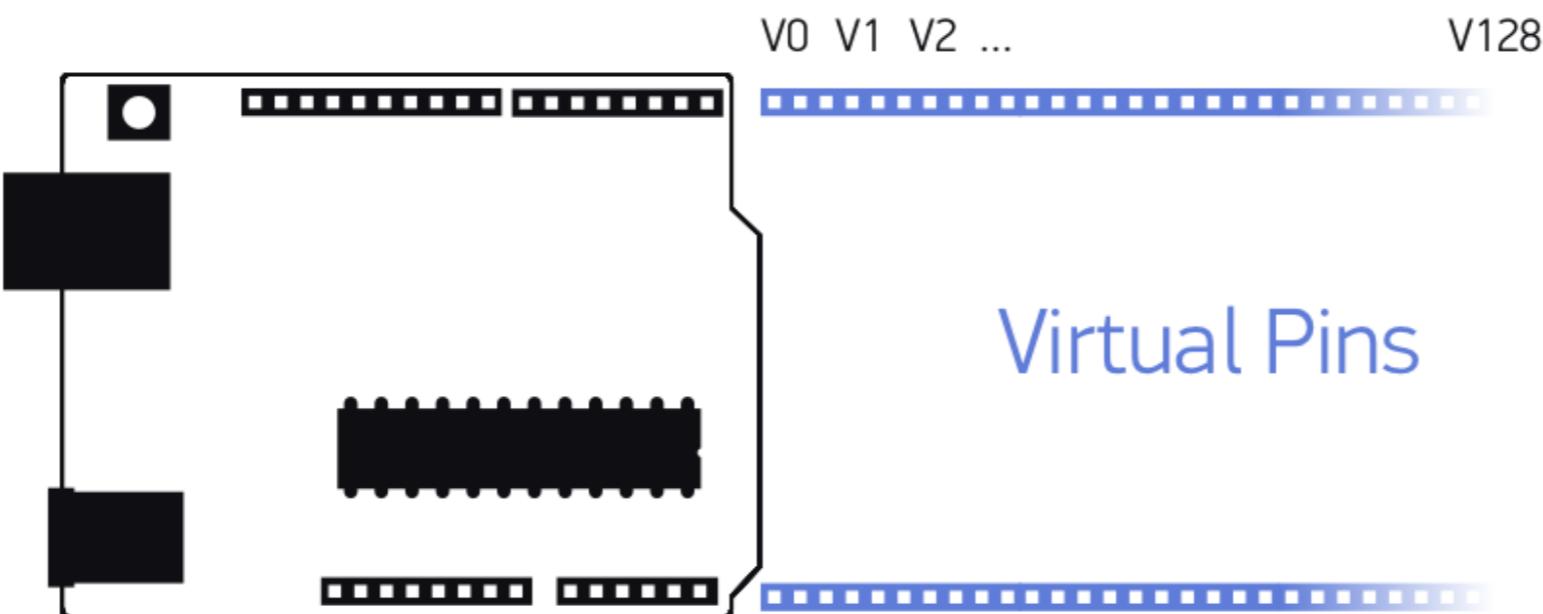
Getting advices in the point of sale according to customer habits, preferences, presence of allergic components for them or expiring dates.

## Noise Urban Maps

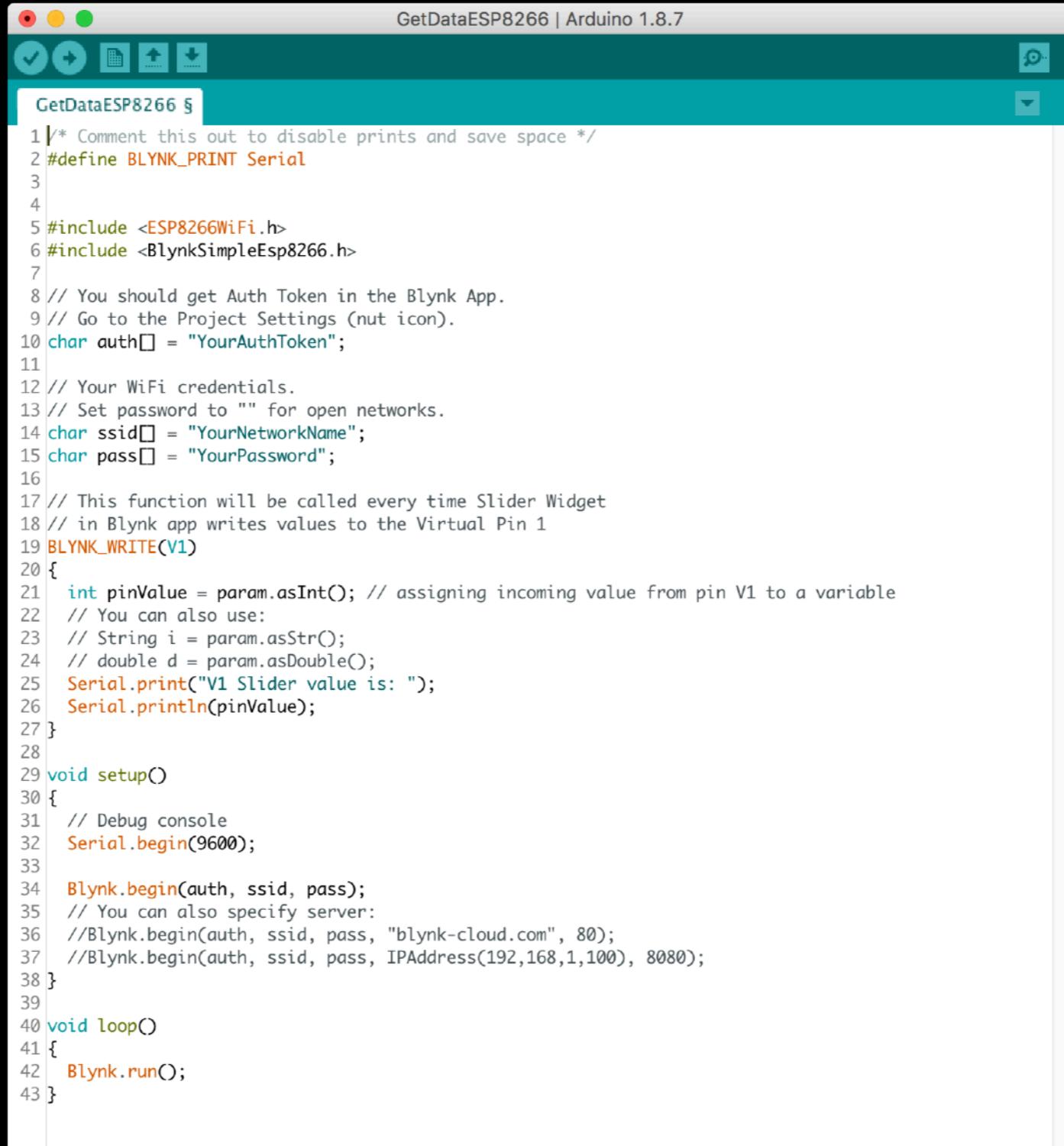
Sound monitoring in bar areas and centric zones in real time.



# Virtual Pins



# Get Data



The screenshot shows the Arduino IDE interface with a sketch named "GetDataESP8266". The code implements a simple Blynk application for an ESP8266. It includes configuration for WiFi credentials and a Blynk token, and defines a function to handle slider input from a virtual pin.

```
/* Comment this out to disable prints and save space */
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// You should get Auth Token in the Blynk App.
// Go to the Project Settings (nut icon).
char auth[] = "YourAuthToken";

// Your WiFi credentials.
// Set password to "" for open networks.
char ssid[] = "YourNetworkName";
char pass[] = "YourPassword";

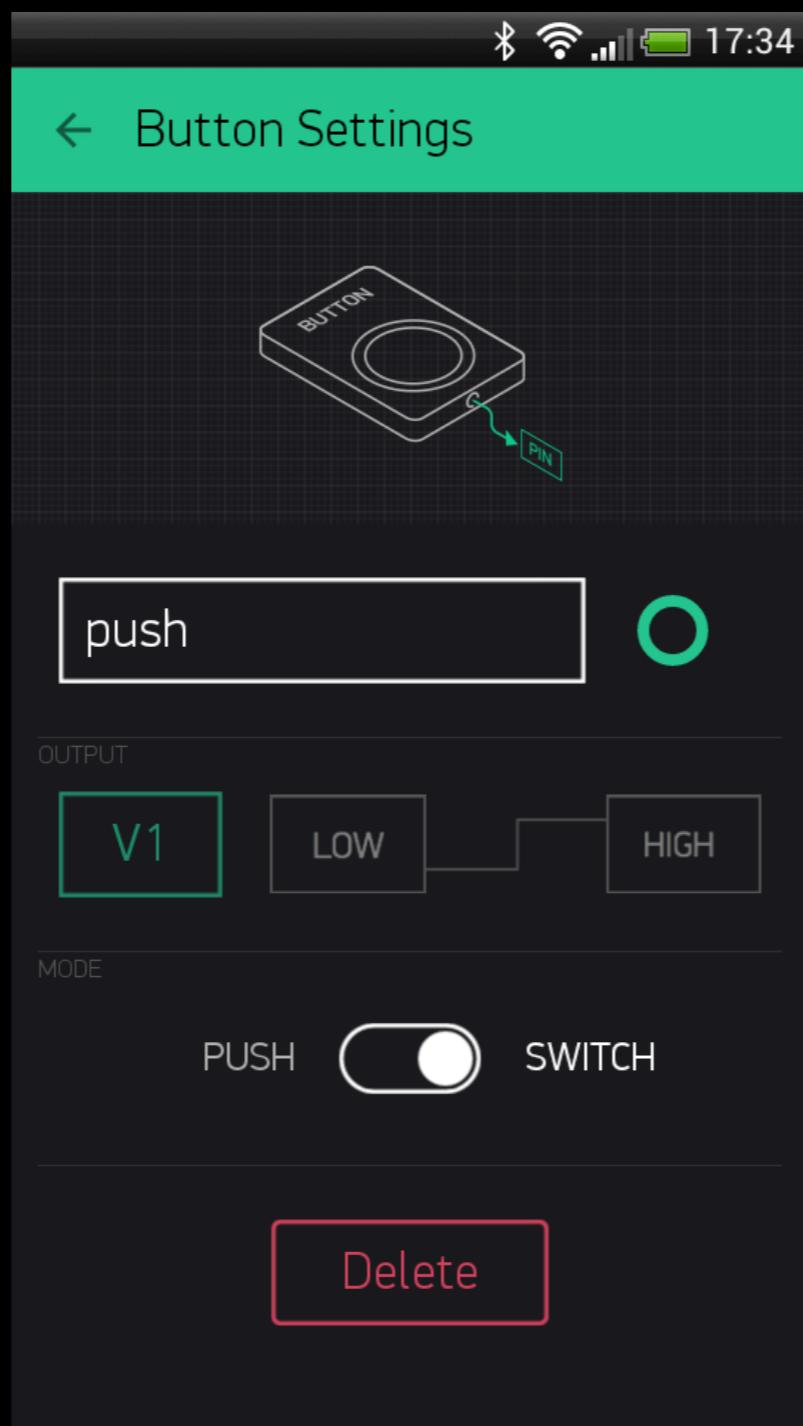
// This function will be called every time Slider Widget
// in Blynk app writes values to the Virtual Pin 1
BLYNK_WRITE(V1)
{
    int pinValue = param.asInt(); // assigning incoming value from pin V1 to a variable
    // You can also use:
    // String i = param.asStr();
    // double d = param.asDouble();
    Serial.print("V1 Slider value is: ");
    Serial.println(pinValue);
}

void setup()
{
    // Debug console
    Serial.begin(9600);

    Blynk.begin(auth, ssid, pass);
    // You can also specify server:
    //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
    //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
}

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

# Get Data



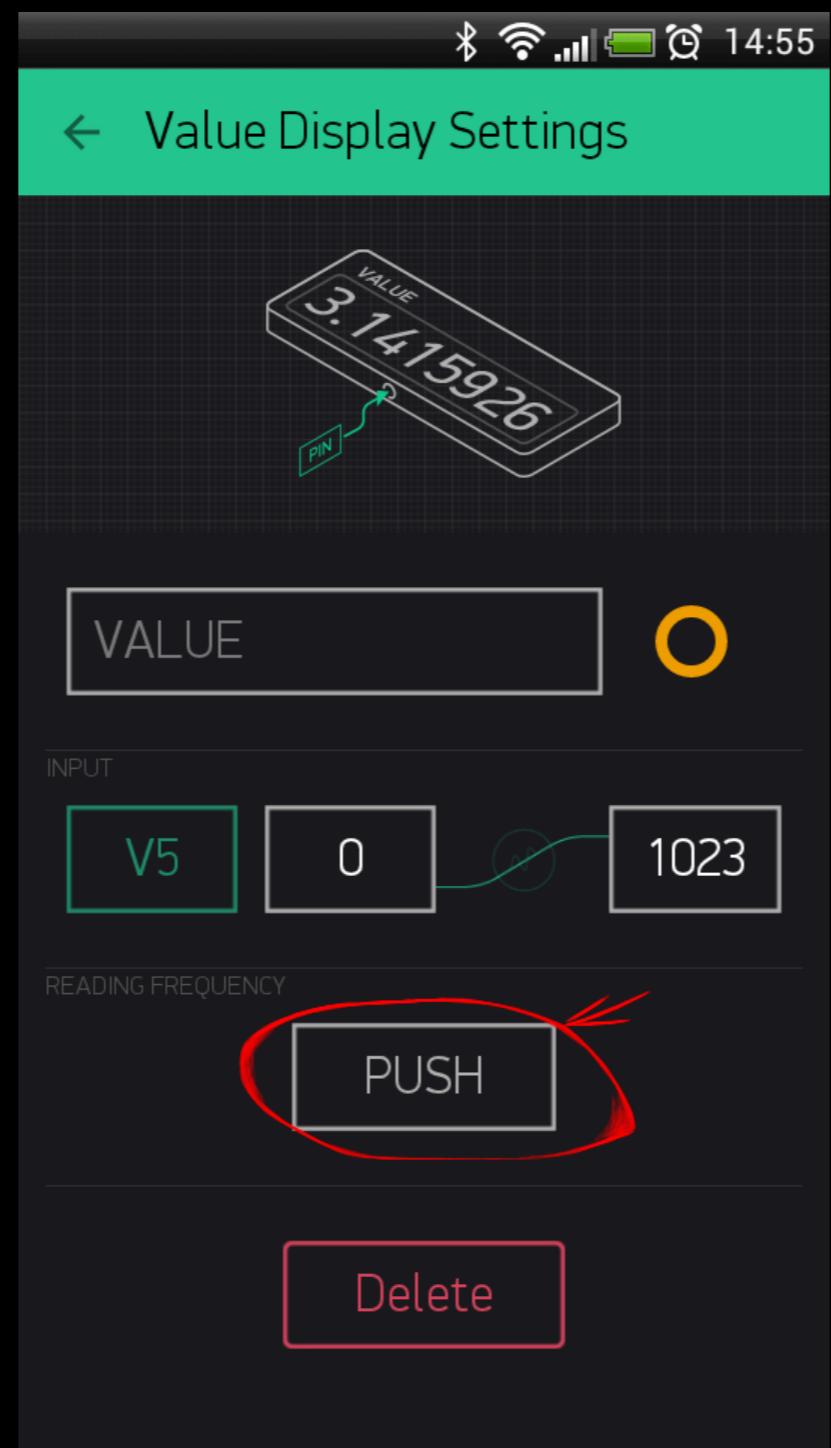
# Push Data



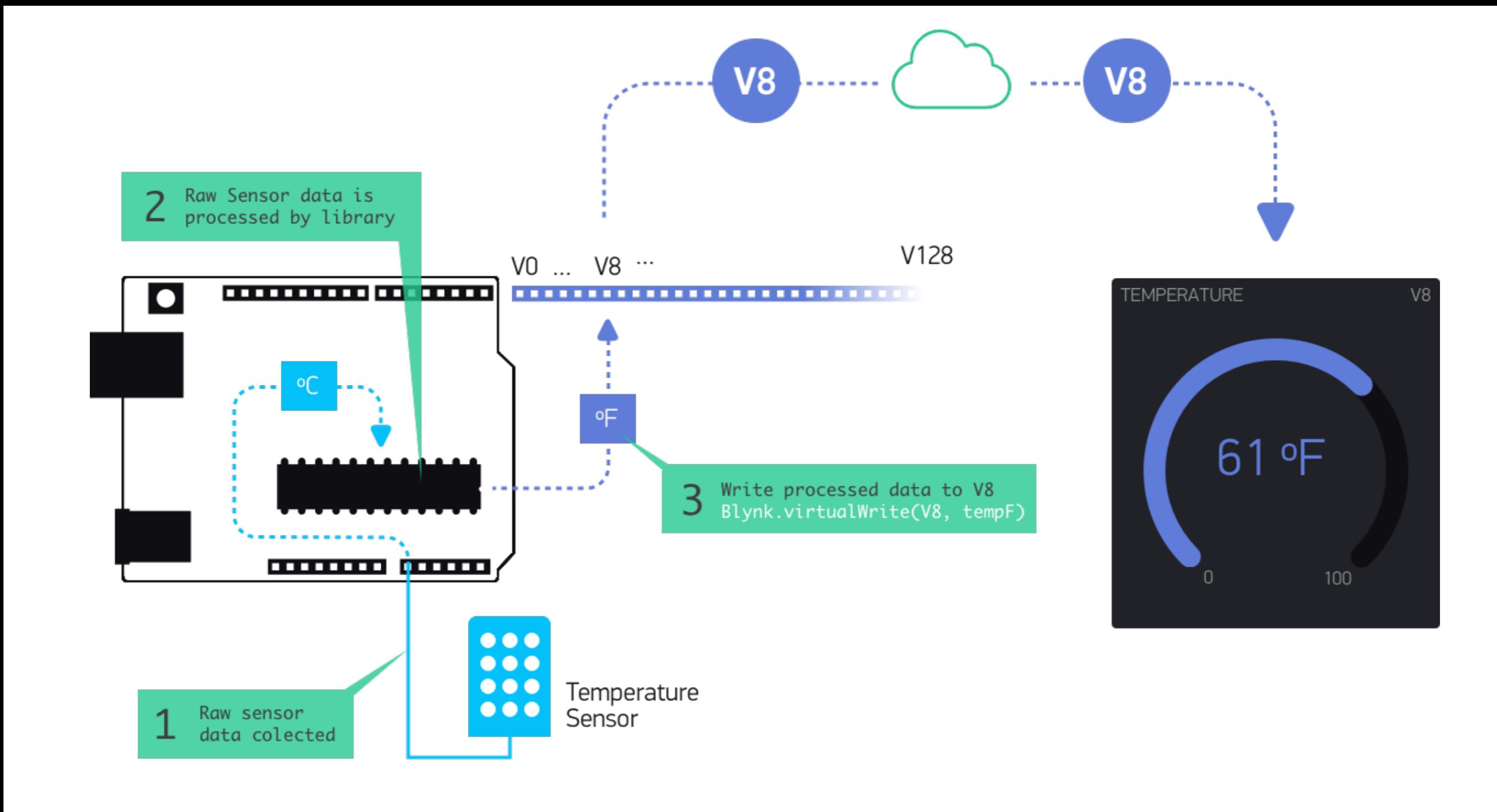
The screenshot shows the Arduino IDE interface with the title bar "PushDataESP8266 | Arduino 1.8.7". The main window displays the code for the "PushDataESP8266" sketch. The code is written in C++ and uses the Blynk library to connect an ESP8266 to a Blynk account. It defines WiFi credentials and sets up a virtual pin for uptime display.

```
1 /* Comment this out to disable prints and save space */
2 #define BLYNK_PRINT Serial
3
4
5
6 #include <ESP8266WiFi.h>
7 #include <BlynkSimpleEsp8266.h>
8
9 // You should get Auth Token in the Blynk App.
10 // Go to the Project Settings (nut icon).
11 char auth[] = "YourAuthToken";
12
13 // Your WiFi credentials.
14 // Set password to "" for open networks.
15 char ssid[] = "YourNetworkName";
16 char pass[] = "YourPassword";
17
18 // Use Virtual pin 5 for uptime display
19 #define PIN_UPTIME V5
20
21 // This function tells Arduino what to do if there is a Widget
22 // which is requesting data for Virtual Pin (5)
23 BLYNK_READ(PIN_UPTIME)
24 {
25     // This command writes Arduino's uptime in seconds to Virtual Pin (5)
26     Blynk.virtualWrite(PIN_UPTIME, millis() / 1000);
27 }
28
29 void setup()
30 {
31     // Debug console
32     Serial.begin(9600);
33
34     Blynk.begin(auth, ssid, pass);
35     // You can also specify server:
36     //Blynk.begin(auth, ssid, pass, "blynk-cloud.com", 80);
37     //Blynk.begin(auth, ssid, pass, IPAddress(192,168,1,100), 8080);
38 }
39
40 void loop()
41 {
42     Blynk.run();
43 }
```

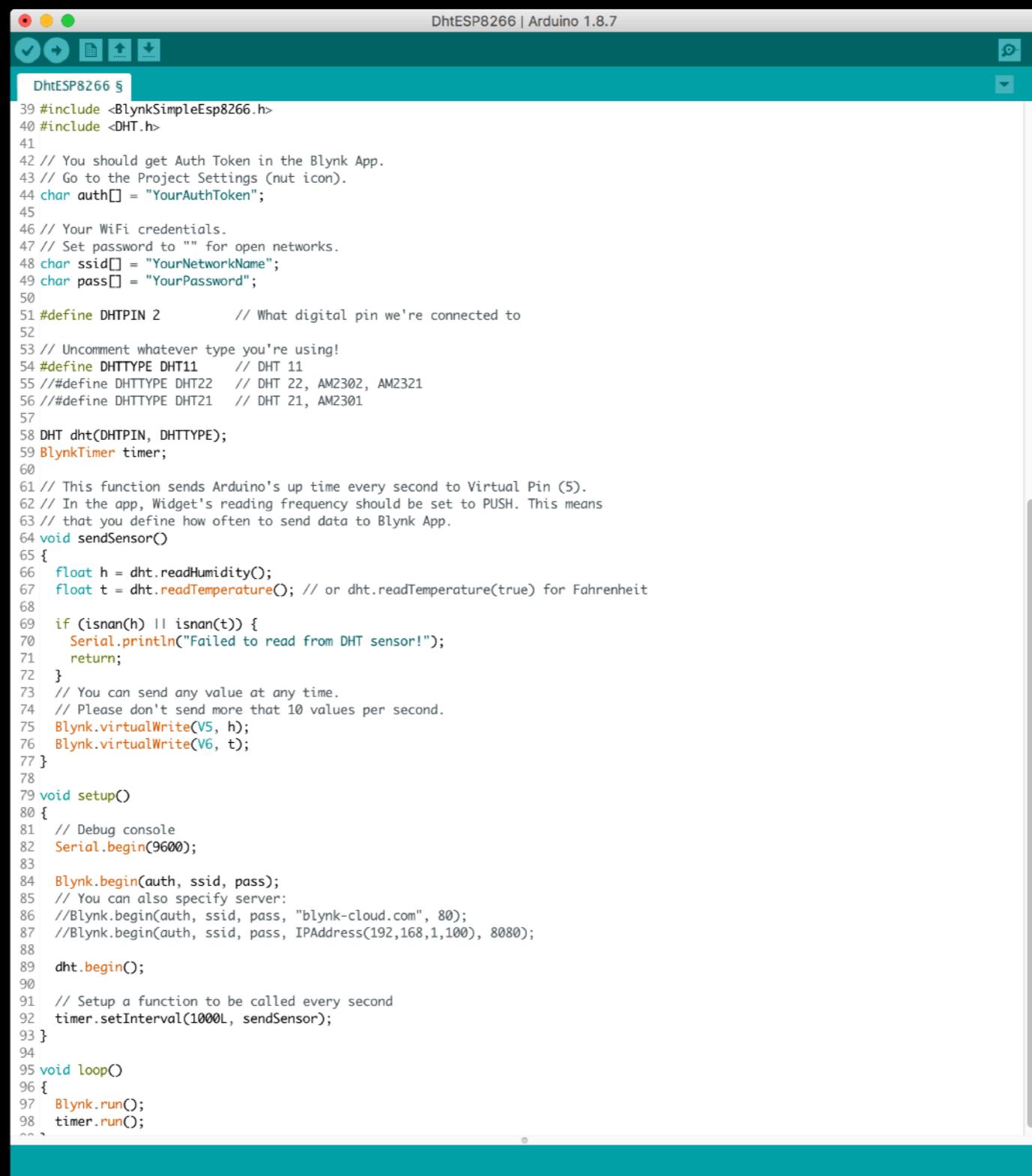
# Push Data



# Temperature



# Temperature



The screenshot shows the Arduino IDE interface with the title bar "DhtESP8266 | Arduino 1.8.7". The main area displays the C++ code for a sketch named "DhtESP8266". The code includes #include statements for BlynkSimpleEsp8266.h and DHT.h, defines for WiFi credentials (authToken, ssid, pass), and pins (DHTPIN). It uses #defines for DHT types (DHT11, DHT22, DHT21) and initializes a DHT object (dht) and a BlynkTimer (timer). The setup() function sets up the serial port at 9600 bps and begins communication with the Blynk cloud using the provided auth token and WiFi credentials. It also starts a timer to call the sendSensor() function every second. The loop() function runs the Blynk run loop and the timer run loop.

```
DhtESP8266 | Arduino 1.8.7

DhtESP8266 §

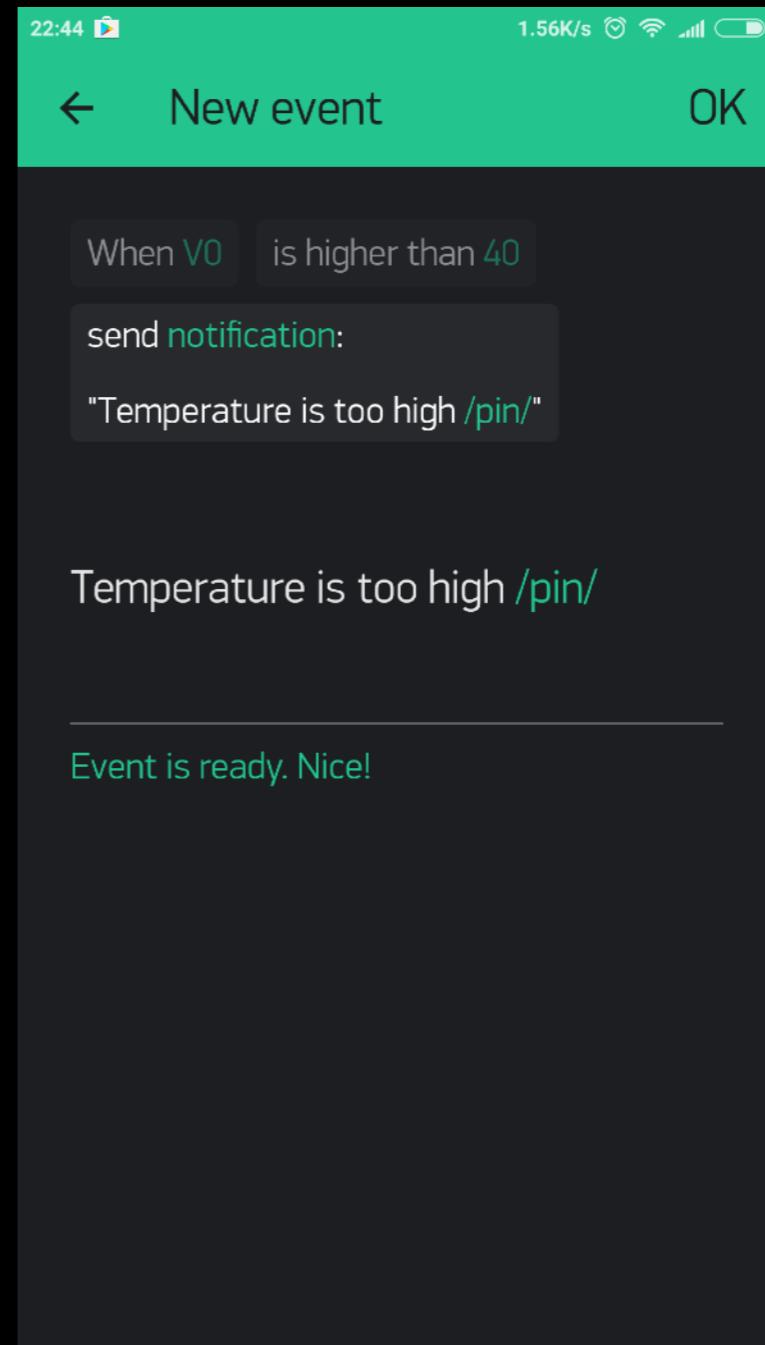
39 #include <BlynkSimpleEsp8266.h>
40 #include <DHT.h>
41
42 // You should get Auth Token in the Blynk App.
43 // Go to the Project Settings (nut icon).
44 char authToken = "YourAuthToken";
45
46 // Your WiFi credentials.
47 // Set password to "" for open networks.
48 char ssid = "YourNetworkName";
49 char pass = "YourPassword";
50
51 #define DHTPIN 2           // What digital pin we're connected to
52
53 // Uncomment whatever type you're using!
54 #define DHTTYPE DHT11     // DHT 11
55 // #define DHTTYPE DHT22   // DHT 22, AM2302, AM2321
56 // #define DHTTYPE DHT21   // DHT 21, AM2301
57
58 DHT dht(DHTPIN, DHTTYPE);
59 BlynkTimer timer;
60
61 // This function sends Arduino's up time every second to Virtual Pin (5).
62 // In the app, Widget's reading frequency should be set to PUSH. This means
63 // that you define how often to send data to Blynk App.
64 void sendSensor()
65 {
66     float h = dht.readHumidity();
67     float t = dht.readTemperature(); // or dht.readTemperature(true) for Fahrenheit
68
69     if (isnan(h) || isnan(t)) {
70         Serial.println("Failed to read from DHT sensor!");
71         return;
72     }
73     // You can send any value at any time.
74     // Please don't send more than 10 values per second.
75     Blynk.virtualWrite(V5, h);
76     Blynk.virtualWrite(V6, t);
77 }
78
79 void setup()
80 {
81     // Debug console
82     Serial.begin(9600);
83
84     Blynk.begin(authToken, ssid, pass);
85     // You can also specify server:
86     //Blynk.begin(authToken, ssid, pass, "blynk-cloud.com", 80);
87     //Blynk.begin(authToken, ssid, pass, IPAddress(192,168,1,100), 8080);
88
89     dht.begin();
90
91     // Setup a function to be called every second
92     timer.setInterval(1000L, sendSensor);
93 }
94
95 void loop()
96 {
97     Blynk.run();
98     timer.run();
99 }
```

# Eventor

```
float t = dht.readTemperature();
if (isnan(t)) {
    return;
}
if (t > 40) {
    Blynk.notify(String("Temperature is too high: ") + t);
}
```

# Eventor

```
float t = dht.readTemperature();
Blynk.virtualWrite(V0, t);
```



# Dash



# 5. Hackathon

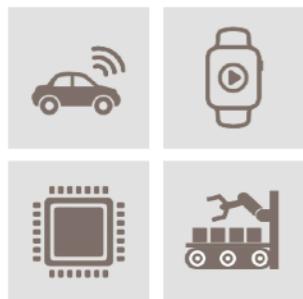


# 6. Future Work

# Parts of an IoT System

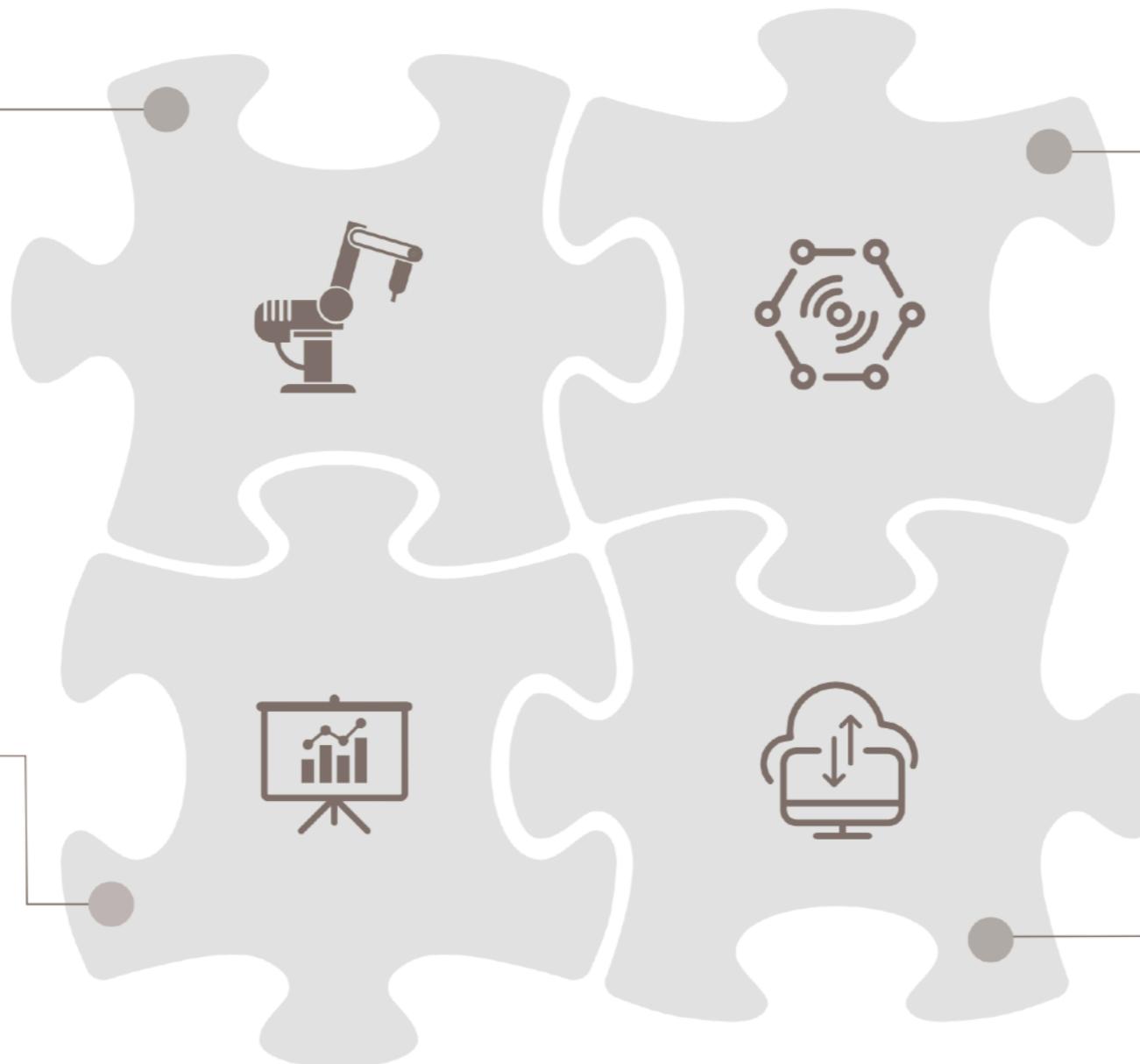
## Devices

- Sensors
- Actuators
- Machines



## Business Model

- Outcome driven
- Product as a service
- Leverage existing investment



## Connectivity

- Network
- Protocol
- Security



## Platform

- Storage
- Analytics
- Visualization
- Integration



# Platforms

1. **PlatformIO** - <https://platformio.org/>
2. **ThingerIO** - <https://thinger.io/>
3. **Oracle Cloud IoT** - <https://cloud.oracle.com/iot>
4. **Azure IoT Hub** - [https://azure.microsoft.com/en-us/  
services/iot-hub/](https://azure.microsoft.com/en-us/services/iot-hub/)
5. **IBM IoT Watson** - <https://www.ibm.com/internet-of-thing>
6. **The Samsung ARTIK** - <https://www.artik.io/>

# Devices

1. **Reliability** - warranty
2. **Scalability:** Getting ready for the big time
3. **Costs:** Counting Pennies
4. **Power consumption:** Every milliamp matters
5. **Security:** You will be attacked

# Connectivity

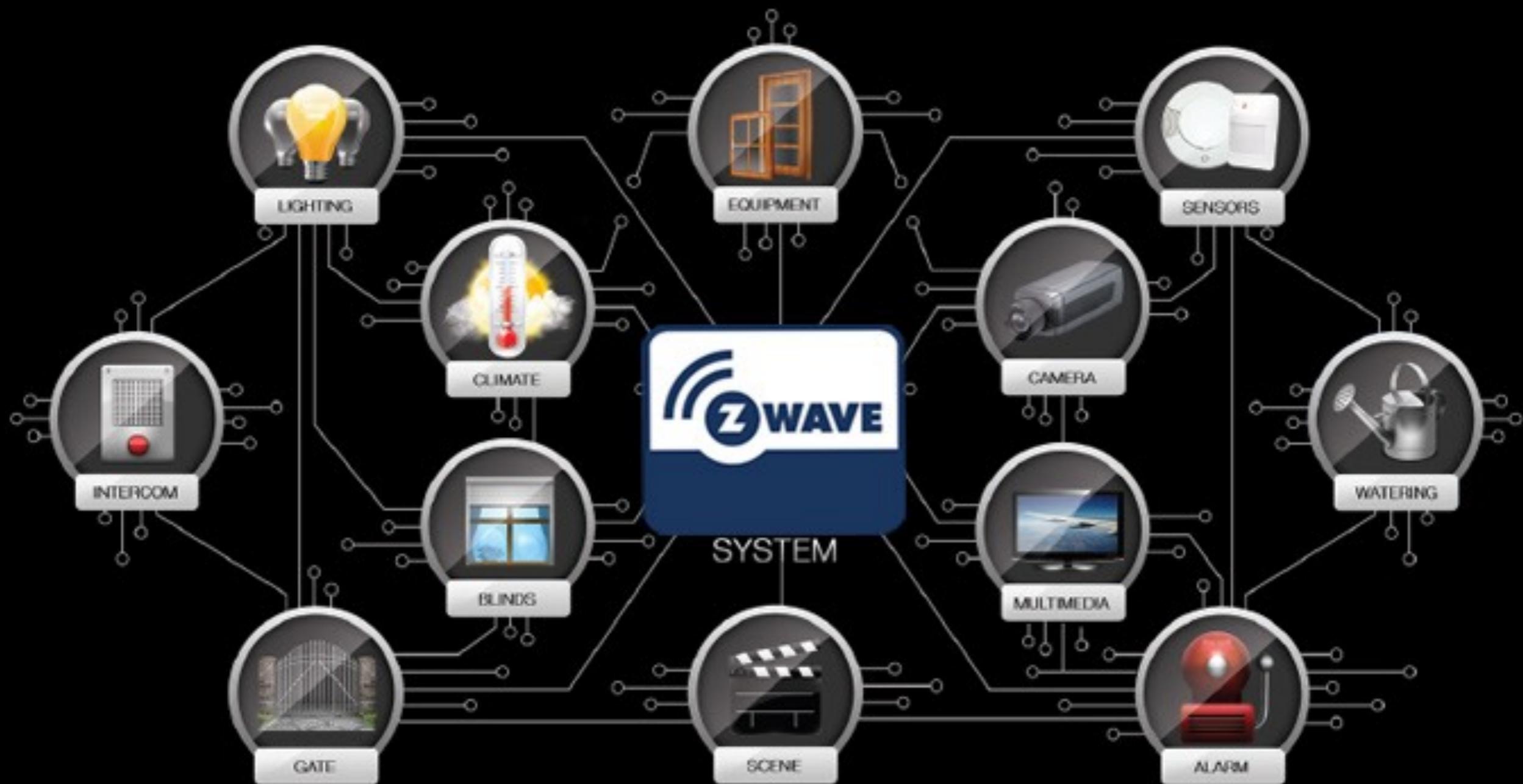
# LoRaWAN



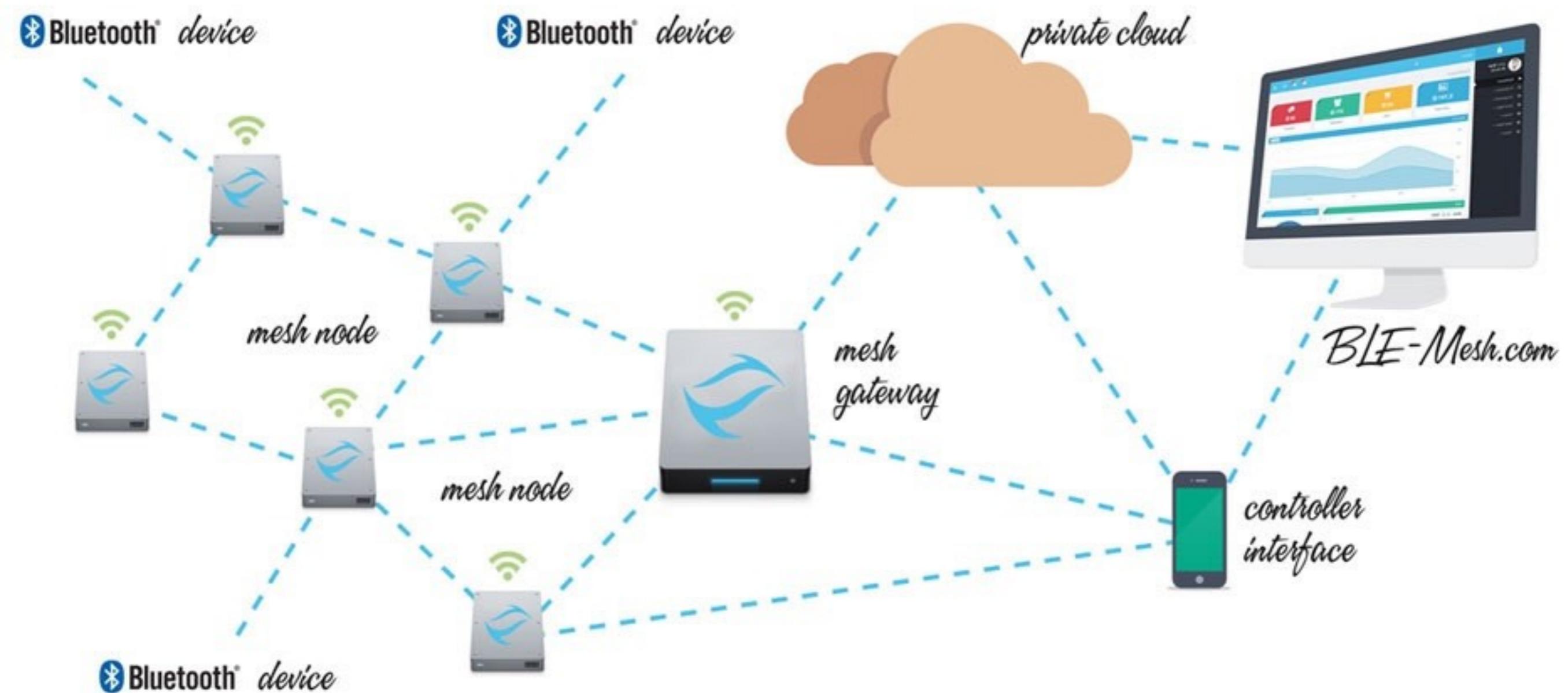
# ZigBee



# Z-Wave



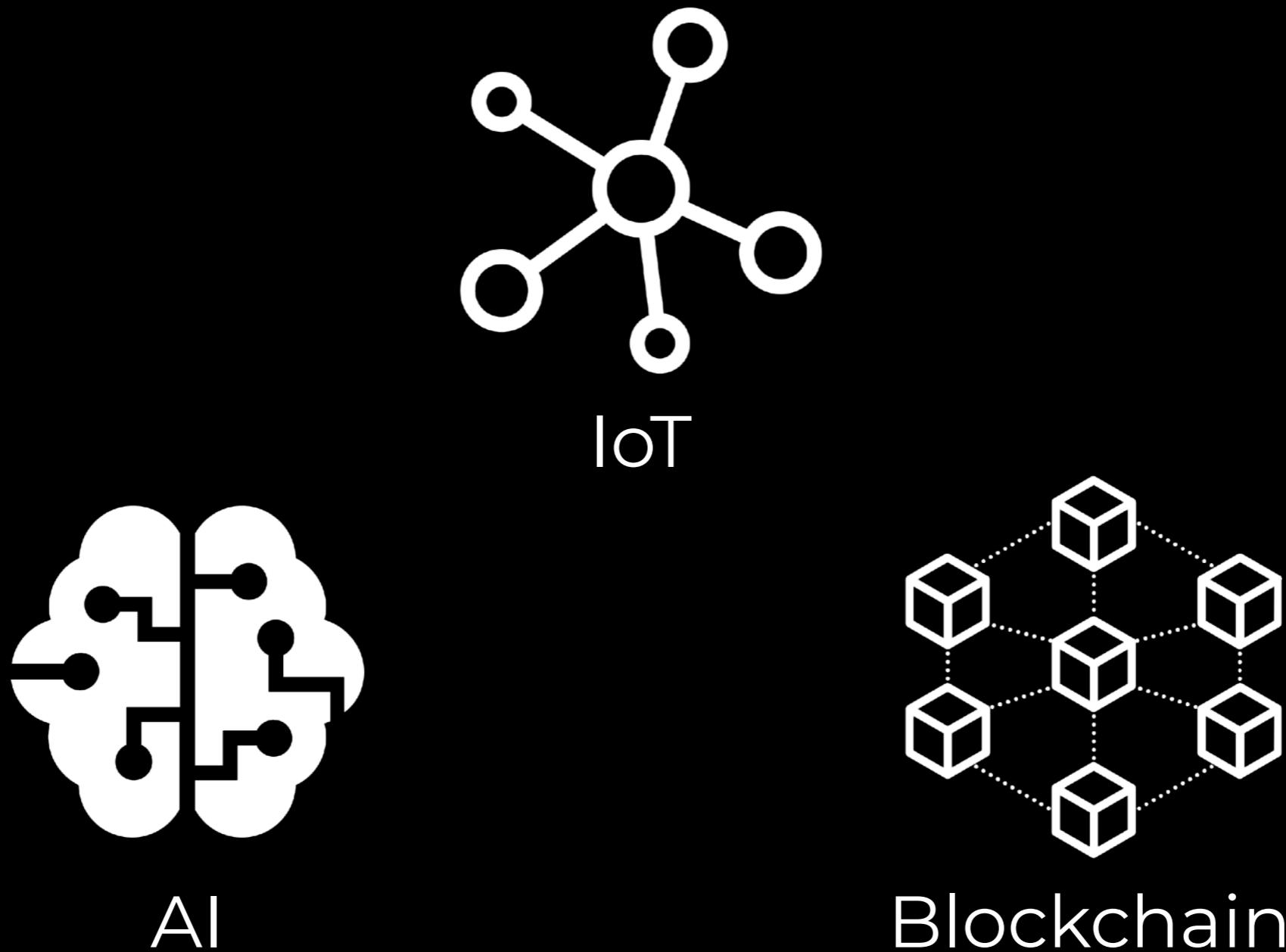
# BLE Mesh



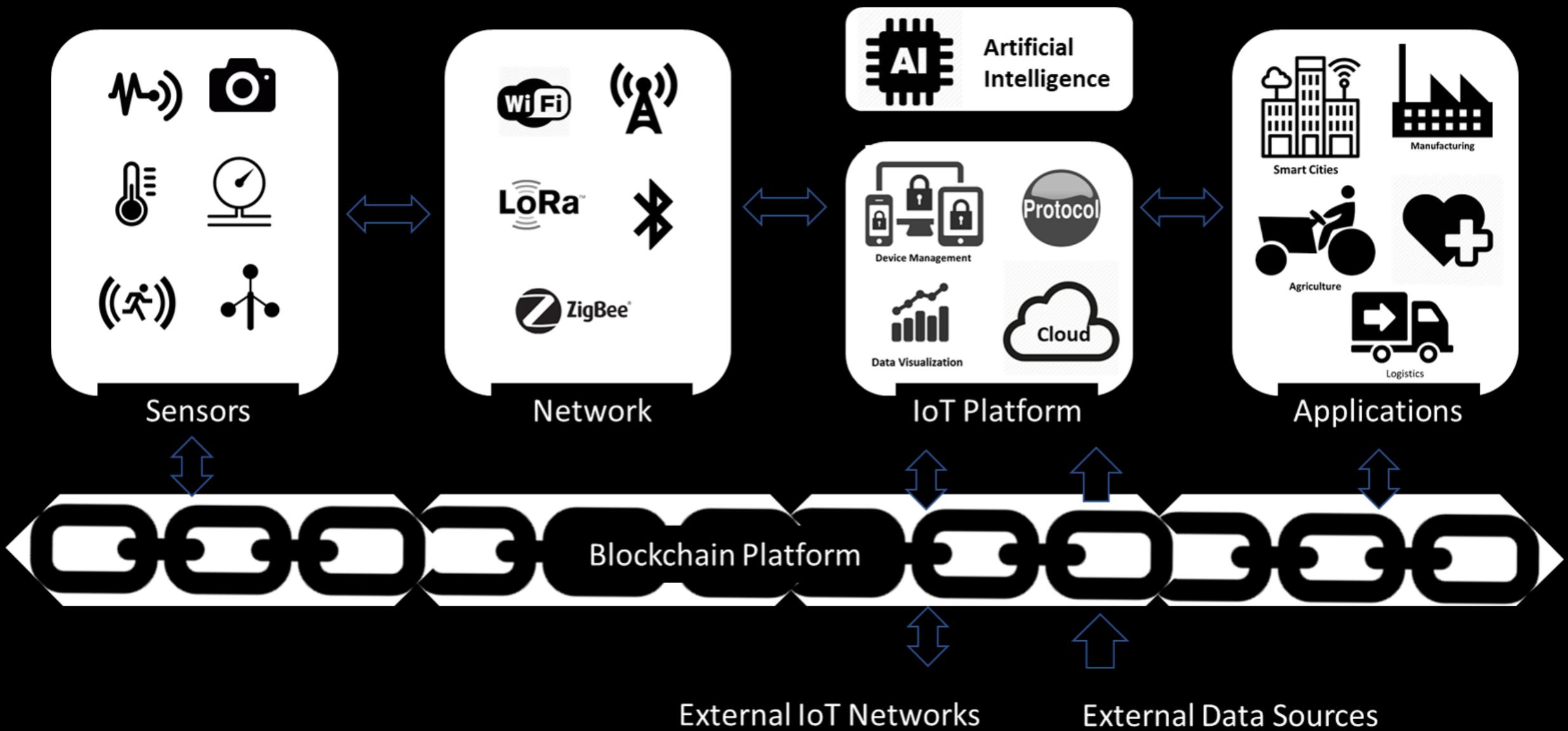
# Business Model

1. **Subscription Model**
2. **Outcome-Based Model**
3. **Asset-Sharing Model**
4. **IoT Products as a Proxy to Sell Another Product**
5. **IoT Products as a Vehicle to Monetize Data**

# New Holy Trinity



# Summary



*“In theory, theory and practice are the same. In practice, they are not.”*

- Albert Einstein -

# Questions?

Imanol Gómez

[imanolgomez.net](http://imanolgomez.net)

[yo@imanolgomez.net](mailto:yo@imanolgomez.net)