Internet of things
What? Why? And How?

By: Eng. Javier Hernández

About Me

- Computer Engineer
- Developer
- Nerd
- Twitter @soynerdito
- soynerdito@gmail.com

- Interests (no particular order)
  - Programming
  - Making
  - Security
  - Hardware
  - Tech communities
  - Just now - Agriculture

@Soynerdito

Goals

- What is Iot?
- Why is so popular?

- Gain awareness of challenges

- Focus on making our own
- Motivate getting hands dirty into making stuff at hobby or business
- Slides at:
- https://github.com/soynerdito/InternetOfThingsTalk

Iot Background

Definition (What is?)

What is iot? Source: Wikipedia

- The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity which enables these things to connect and exchange data, creating opportunities for more direct integration of the physical world into computer-based systems, resulting in efficiency improvements, economic benefits, and reduced human exertions

What is iot? Source: Wikipedia

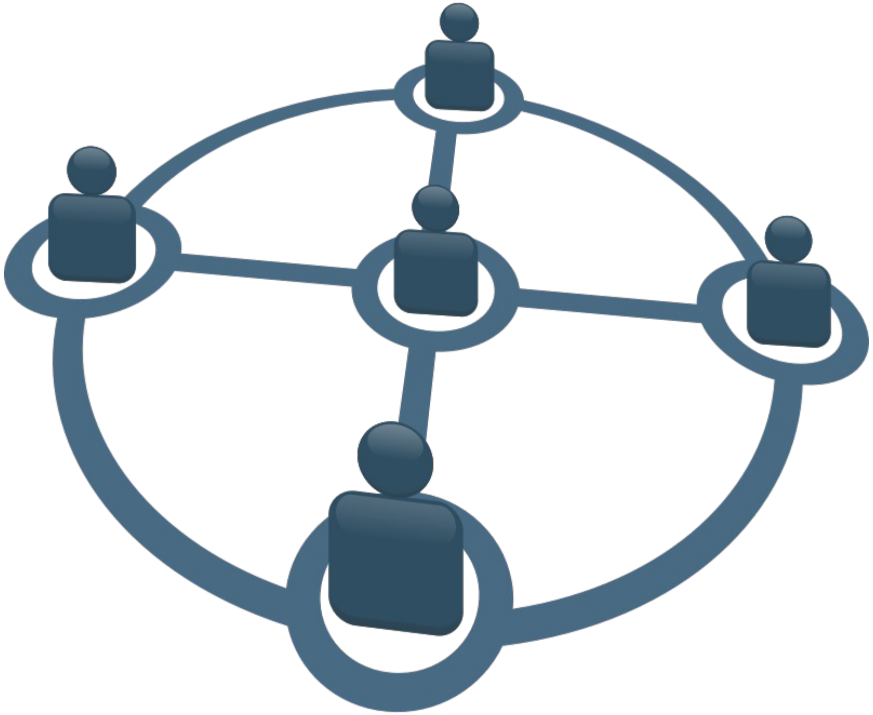
- network of physical devices
  - vehicles, home appliances, and other items
- embedded with electronics
  - software, sensors, actuators, and connectivity
- enables things to connect and exchange data
- direct integration of the physical world into computer-based systems

What is a device

- Mobile phones, tablets, laptops
- Transportation: Car, Bus, bikes
- Home appliances: Fridge, TV
- Traffic lights
- Doors
- Clothes
- Furniture
- Anything and everything

Every device has network connectivity

- Some examples may be:
  - Wireless
    - Bluetooth
    - Wifi
    - RF
    - IRDA
    - LoraWAN
  - Wired
    - Ethernet
    - RS232/RS485
    - USB

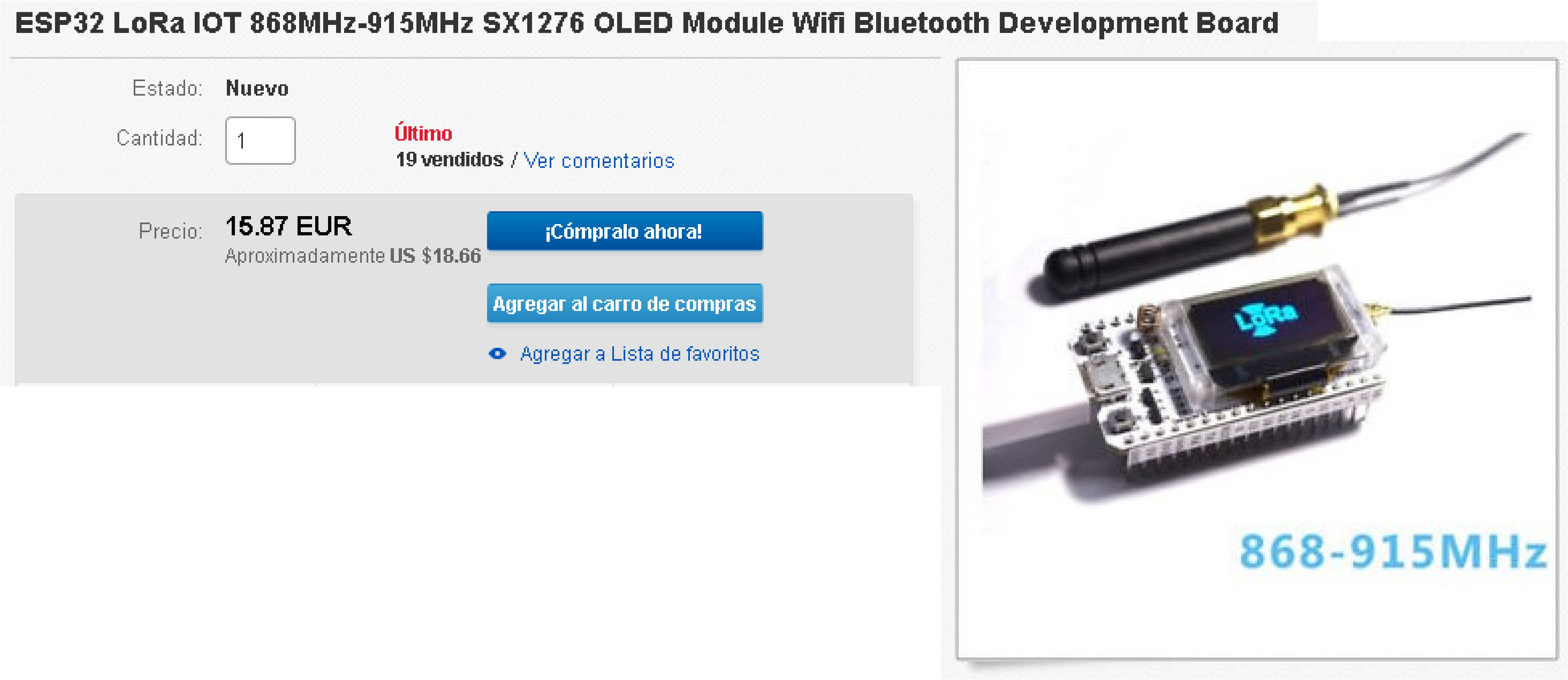


LoRa (Long Range)

- uses license-free sub-gigahertz radio frequency bands like 169 MHz, 433 MHz, 868 MHz (Europe) and 915 MHz (North America).
- Enables very-long-range transmissions (more than 10 km in rural areas) with low power consumption.The technology is presented in two parts — LoRa, the physical layer and LoRaWAN (Long Range Wide Area Network), the upper layers.

Source: https://en.wikipedia.org/wiki/LoRa

LoRa example



ESP32 LoRa IOT 868MHz-915MHz SX1276 OLED Module Wifi Bluetooth Development Board

Estado: Nuevo

Cantidad: 1

Último

19 vendidos / Ver comentarios

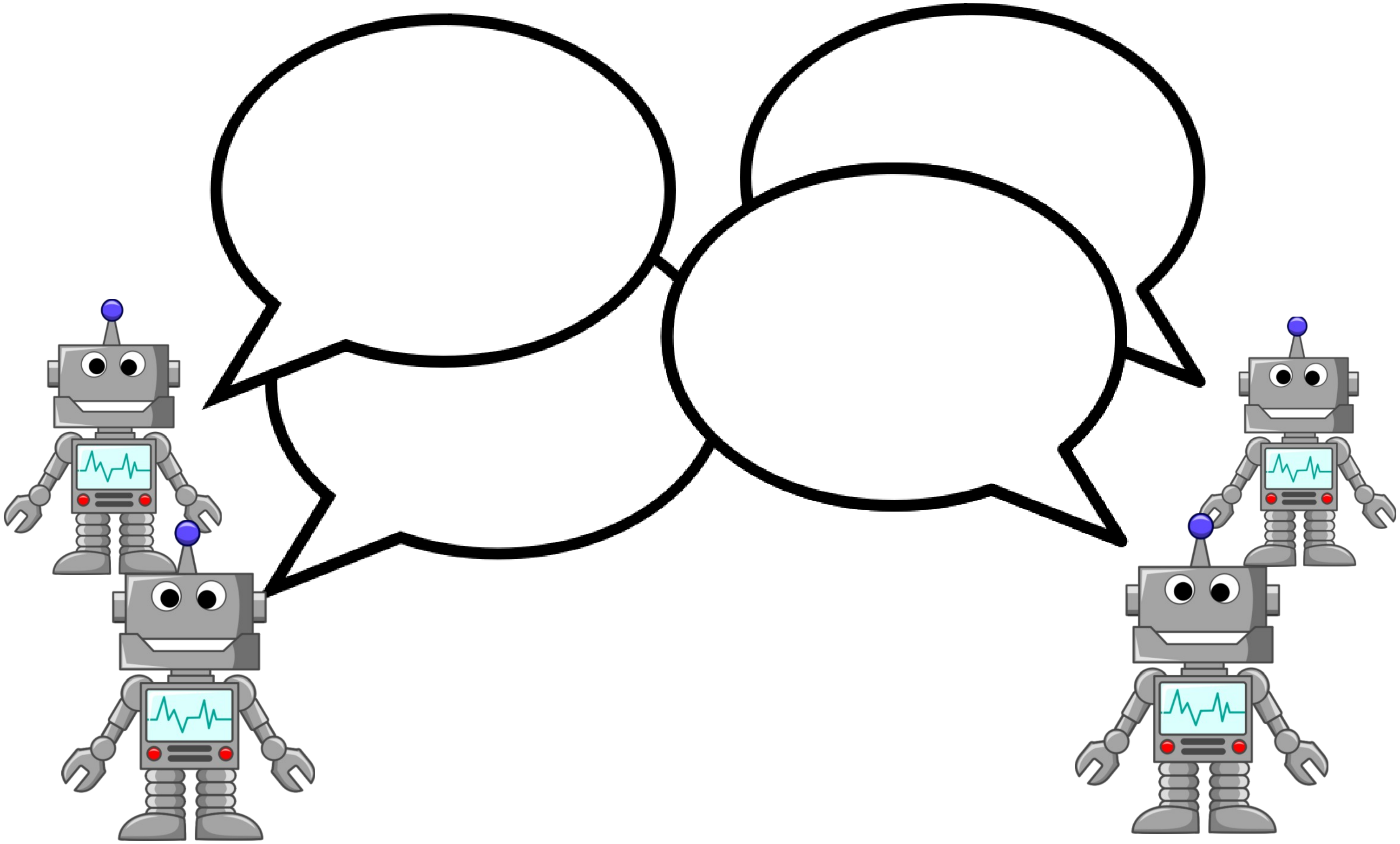
Precio: 15.87 EUR

Aproximadamente US $18.66

Source: ebay.com

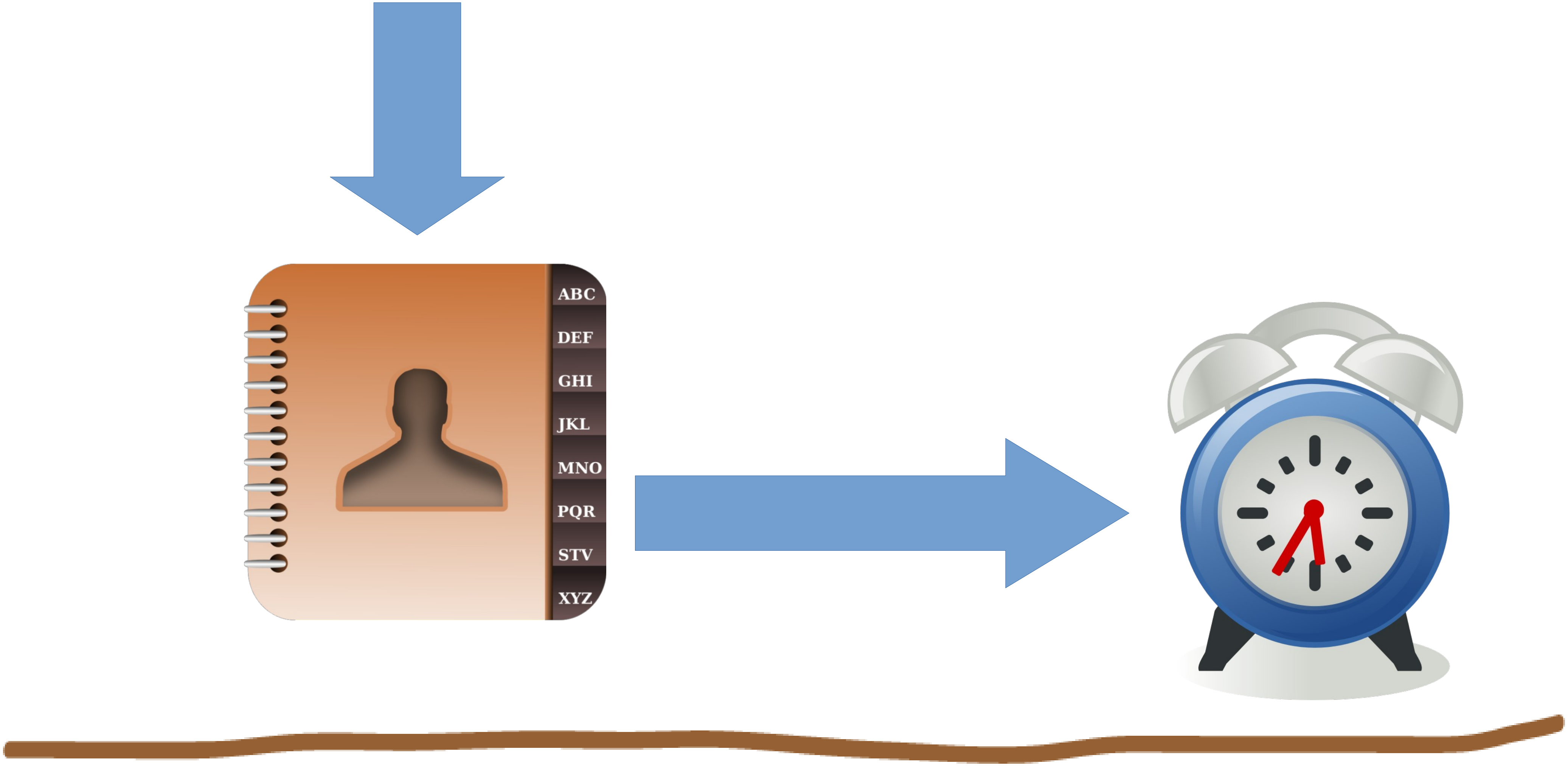
Send and Receive Data

- The devices can talk!



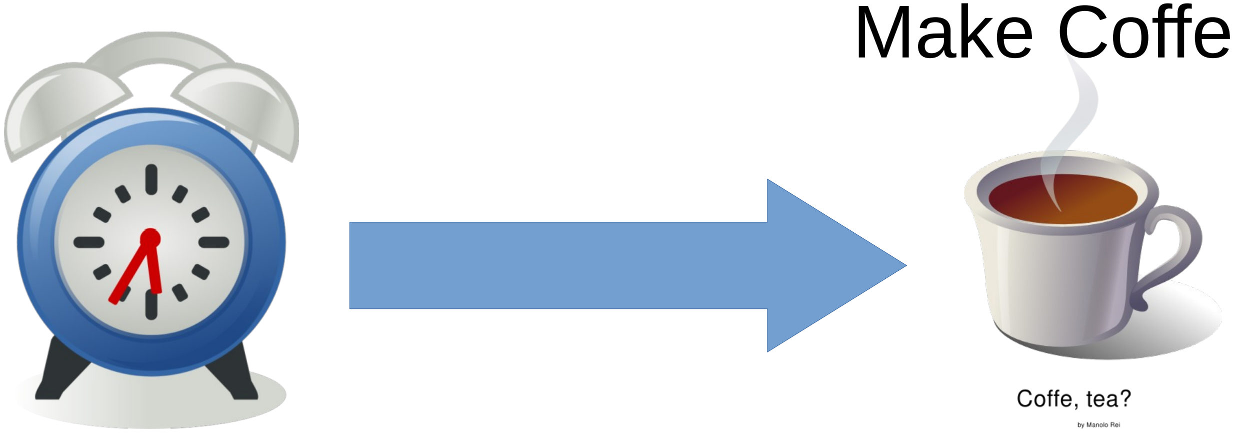
Example

- Monday 8AM meeting @Office



Monday

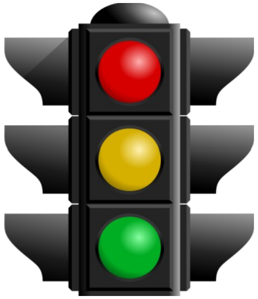
- Morning
  - Wake up on time
  - Automatically make coffee



Make Coffee

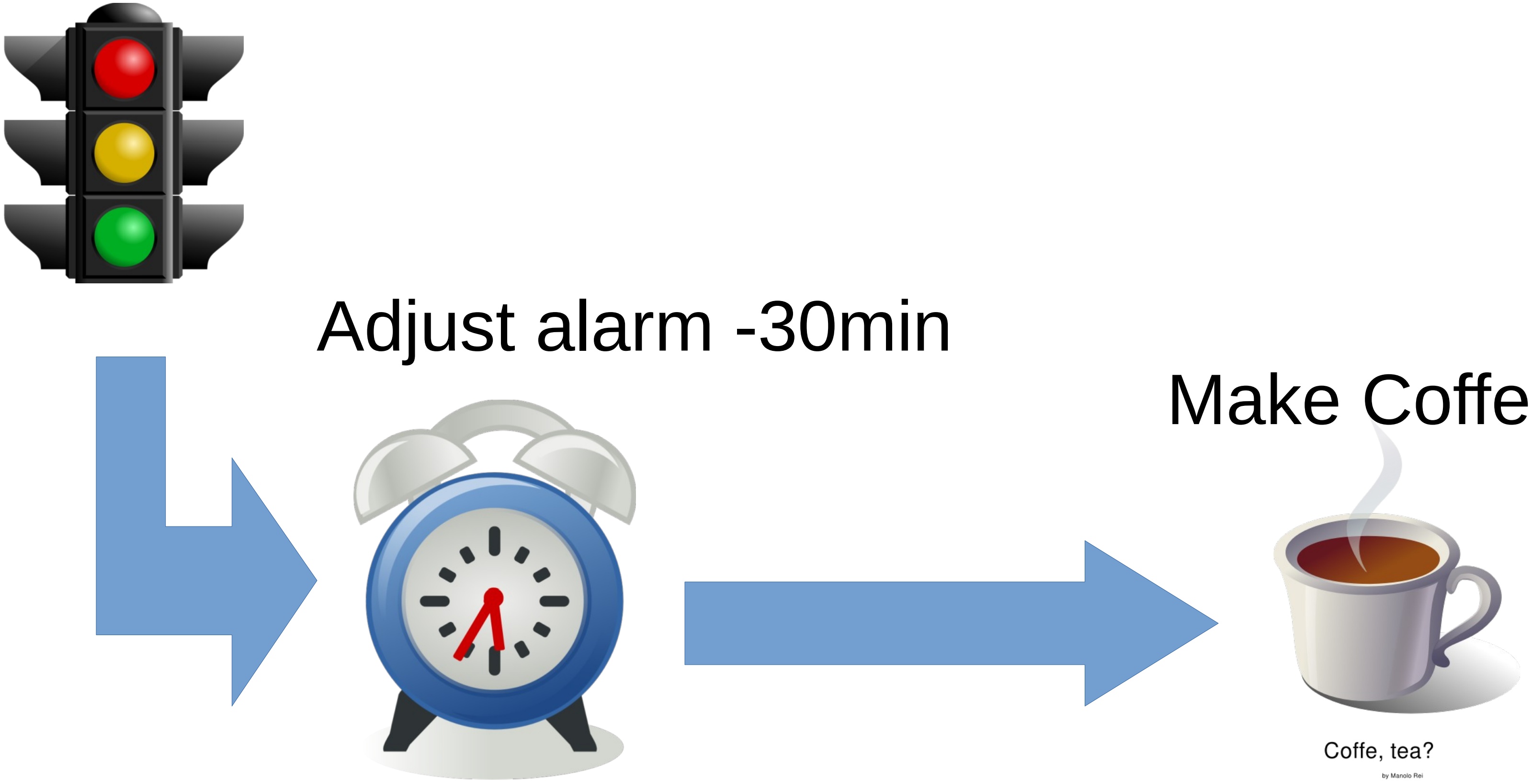
Monday

- Traffic report
- Estimated morning commute delay 30min



Monday

- Traffic report
  - Estimated morning commute delay 30min



Adjust alarm -30min

Make Coffee

Traditionally

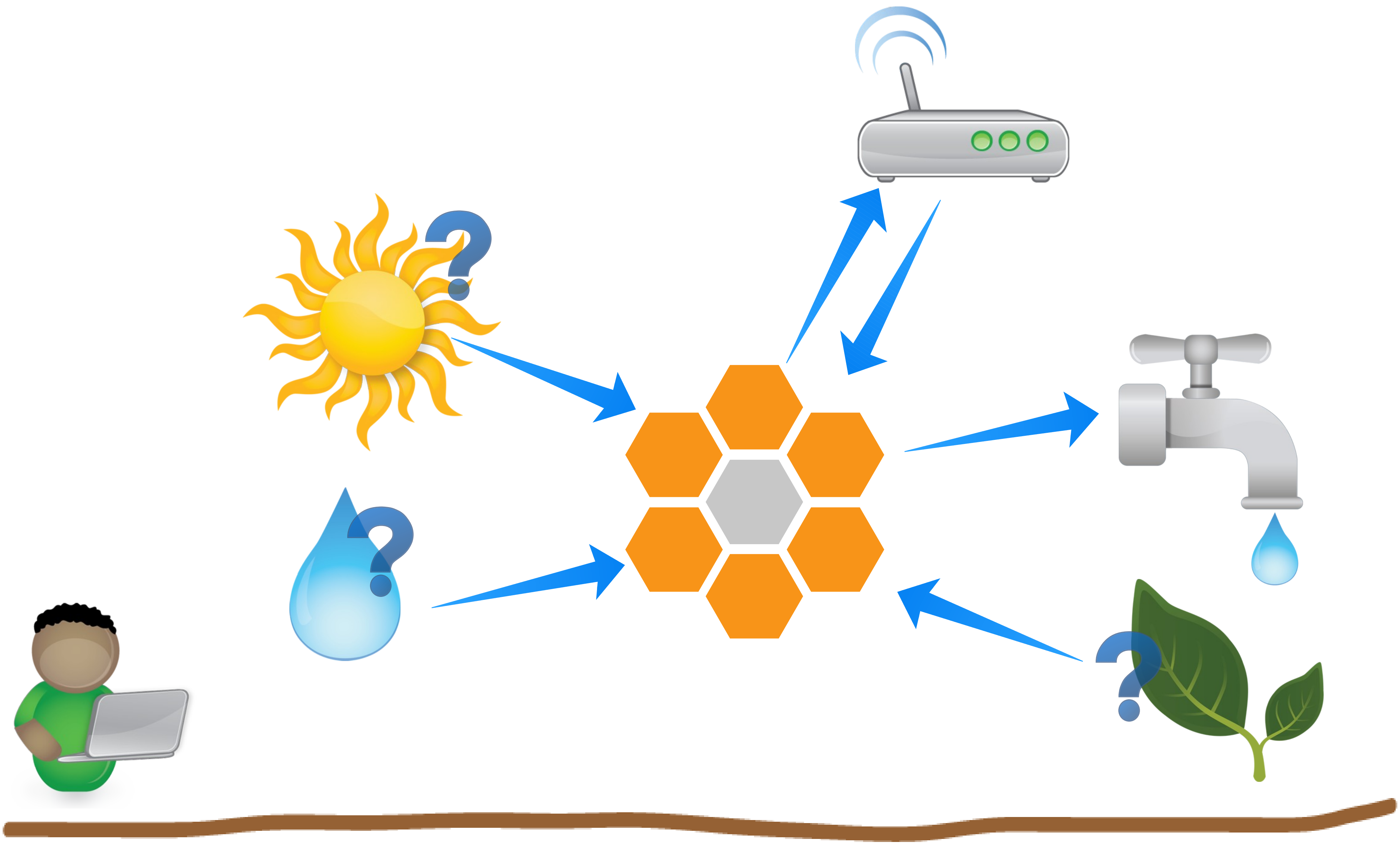
- Person set water by timer

Traditionally

- Person set water by timer

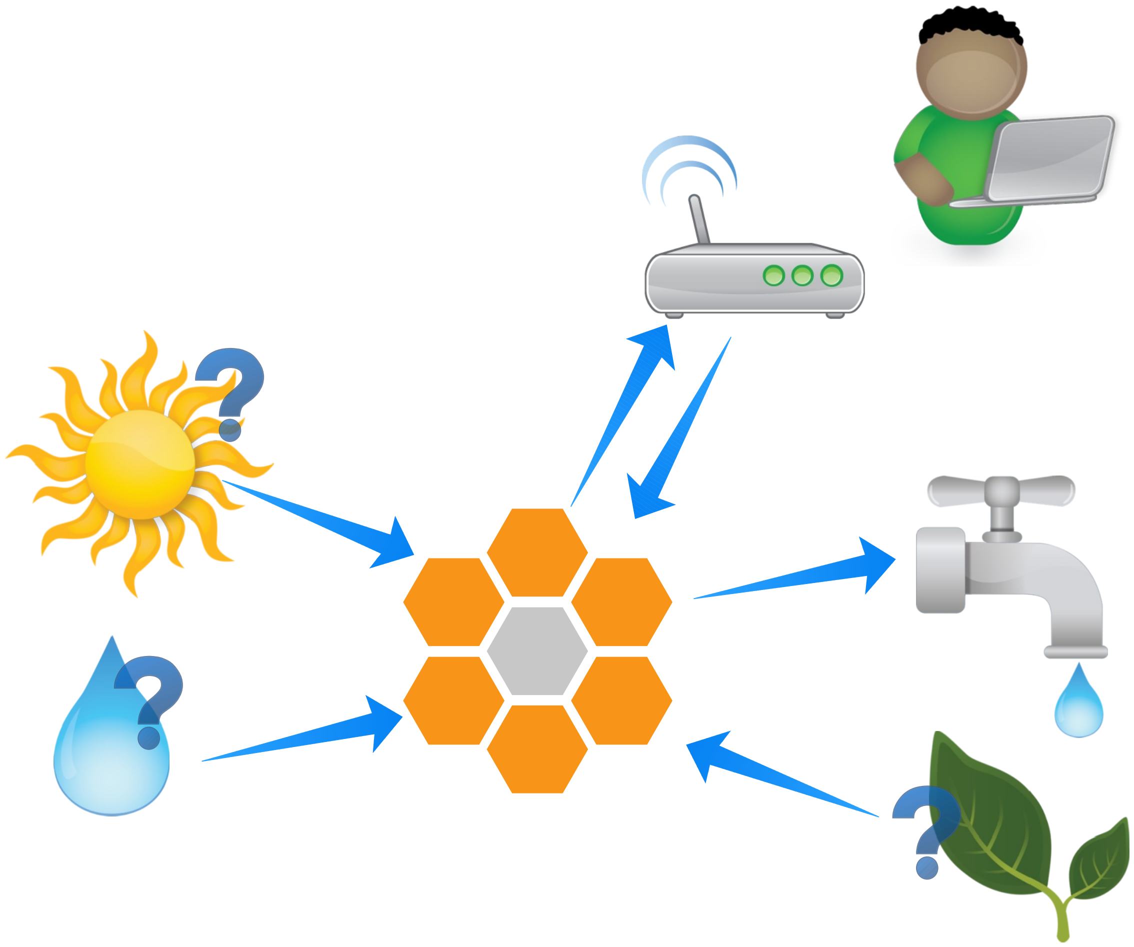
- Is really needed?
- Was the plant dry?
- Did it rained?
- Will there be rain latter today?
- Feedback, was it activated?

More integrated with internet



“Iot” like solution

- Sensors provide information
  - Light
  - Temperature
  - Water
  - Soil moisture
  - Weather forecast
  - Is water available
- User feedback



Example: Weather station

- Sensor
  - Wind speed
  - Wind direction
  - Rain
  - Temperature
  - Humidity
  - Barometer
- Connectivity
  - Wifi
- Interaction
  - App
  - Display

Source: Amazon.com

https://www.amazon.com/Ambient-Weather-Professional-Monitoring-Compatible/dp/B01N5TEHLI/ref=sr_1_3?ie=UTF8&qid=1535421699&sr=8-3&keywords=weather+station

Iot Background

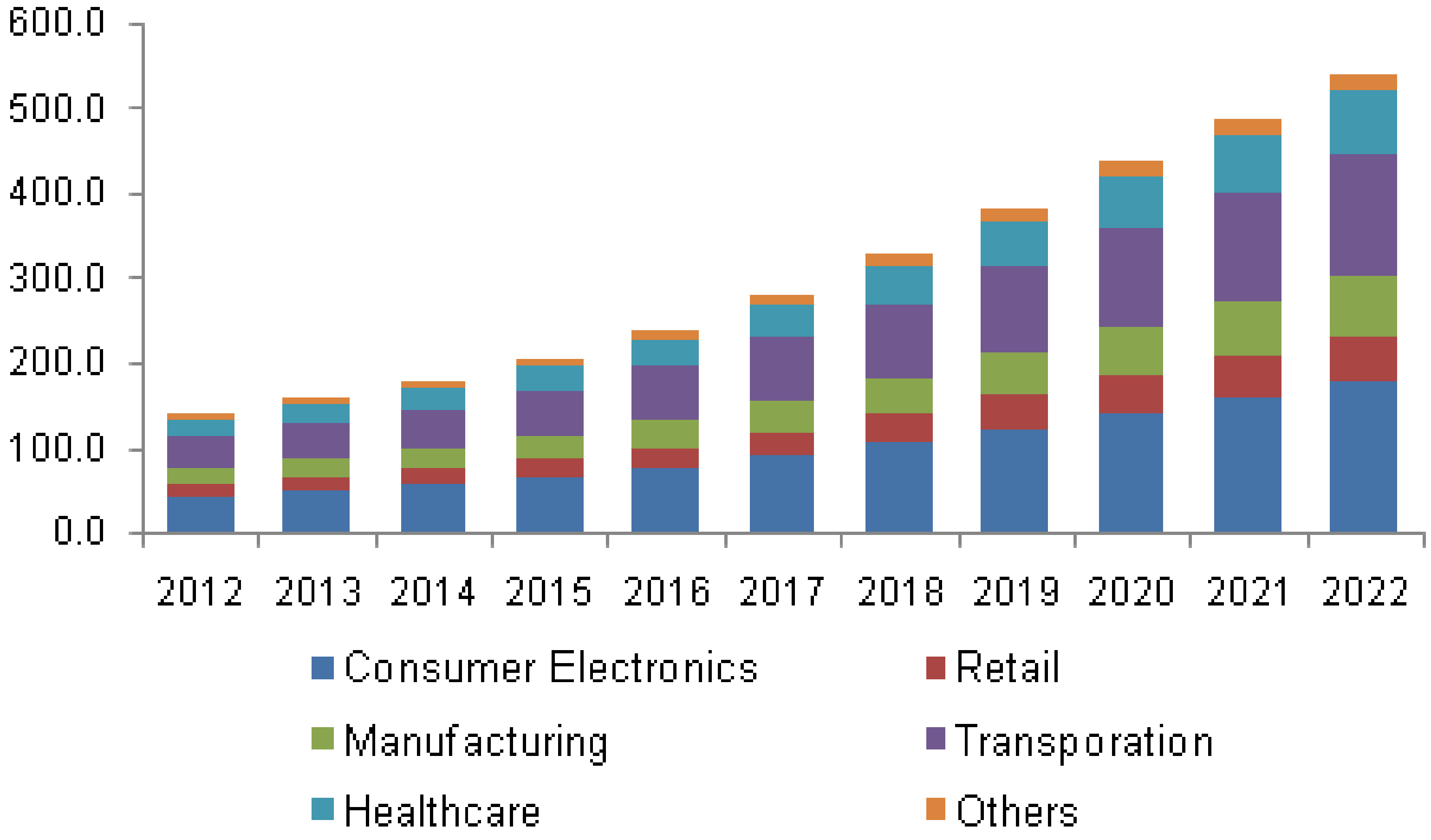
Why is so popular?

Industry Insights

- The global Internet of Things (IoT) market size was valued at **USD 605.69 billion in 2014.** Technological proliferation and increasing investments are expected to drive the global market over the next seven years. Significant advances in telecommunication and growing penetration of internet and broadband services have further propelled the growth of Internet of Things (IoT) industry.

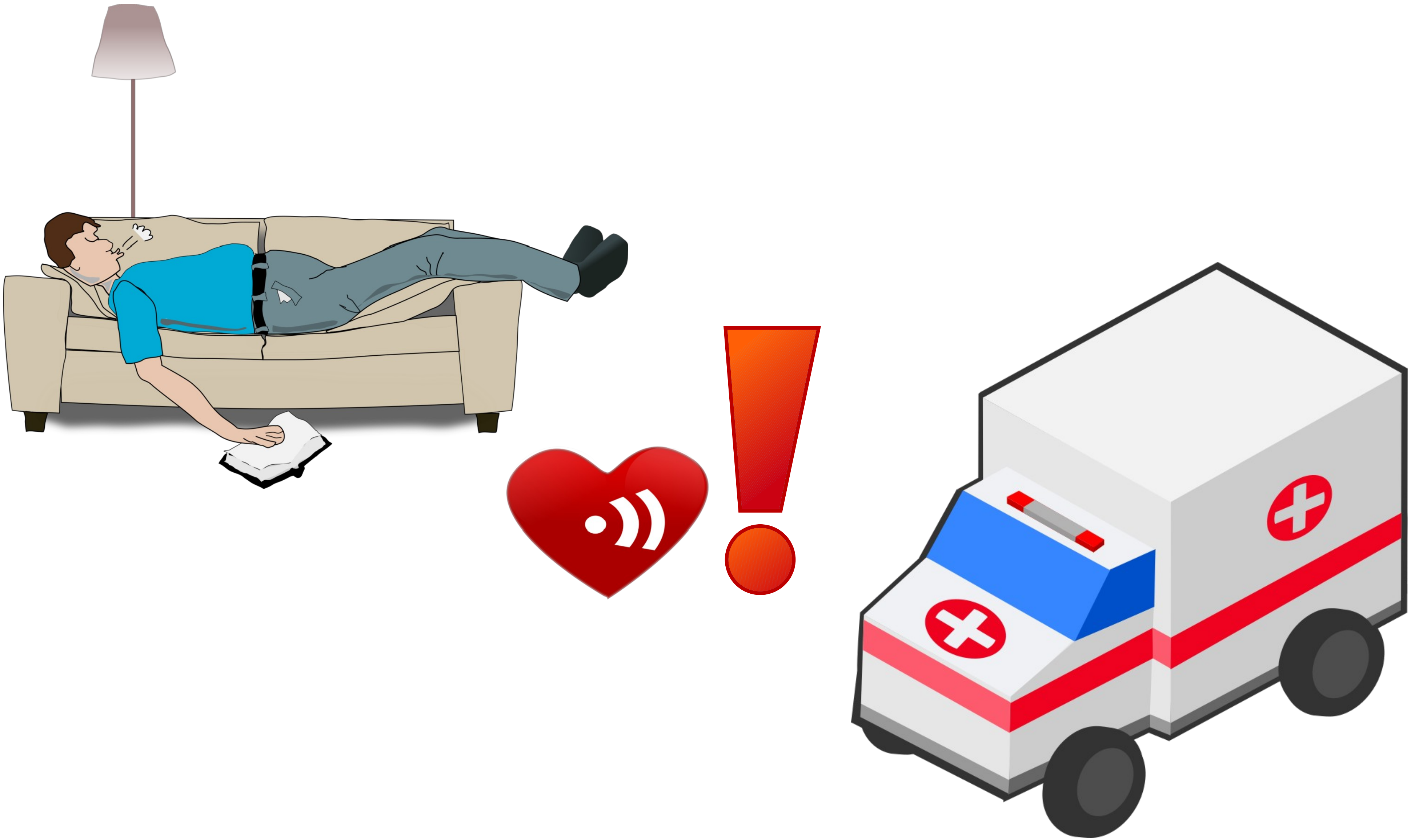
Source: https://www.grandviewresearch.com/industry-analysis/iot-market

North America IoT market revenue by application, 2012 - 2022 (USD Billion)



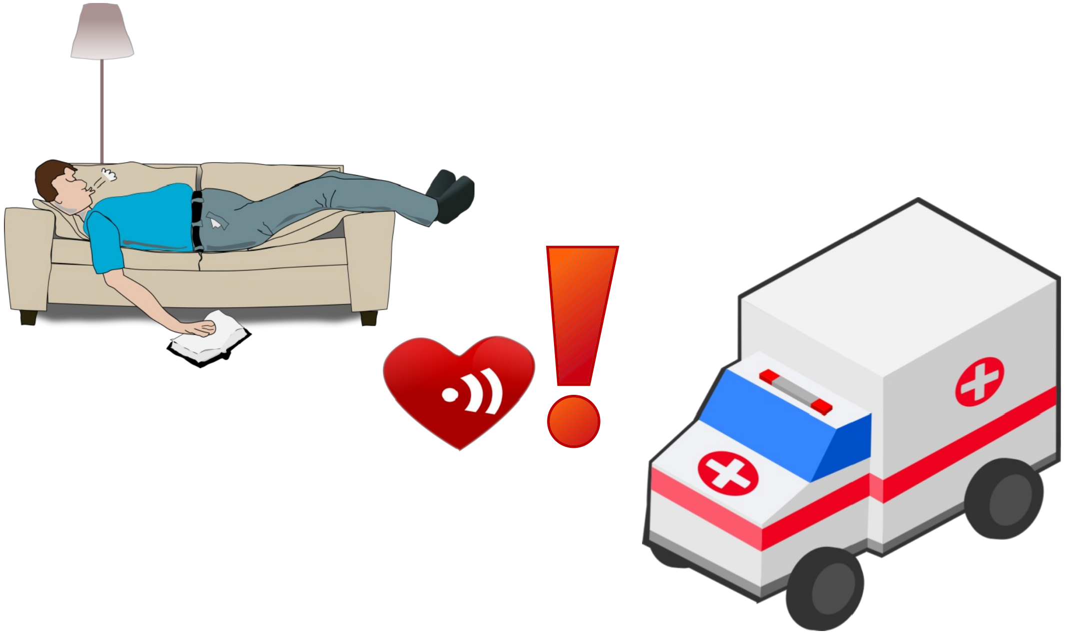
Source: https://www.grandviewresearch.com/industry-analysis/iot-market

Health



Health

- Heart attack
- Call ambulance before it happens
- Patient with doctor while the attack

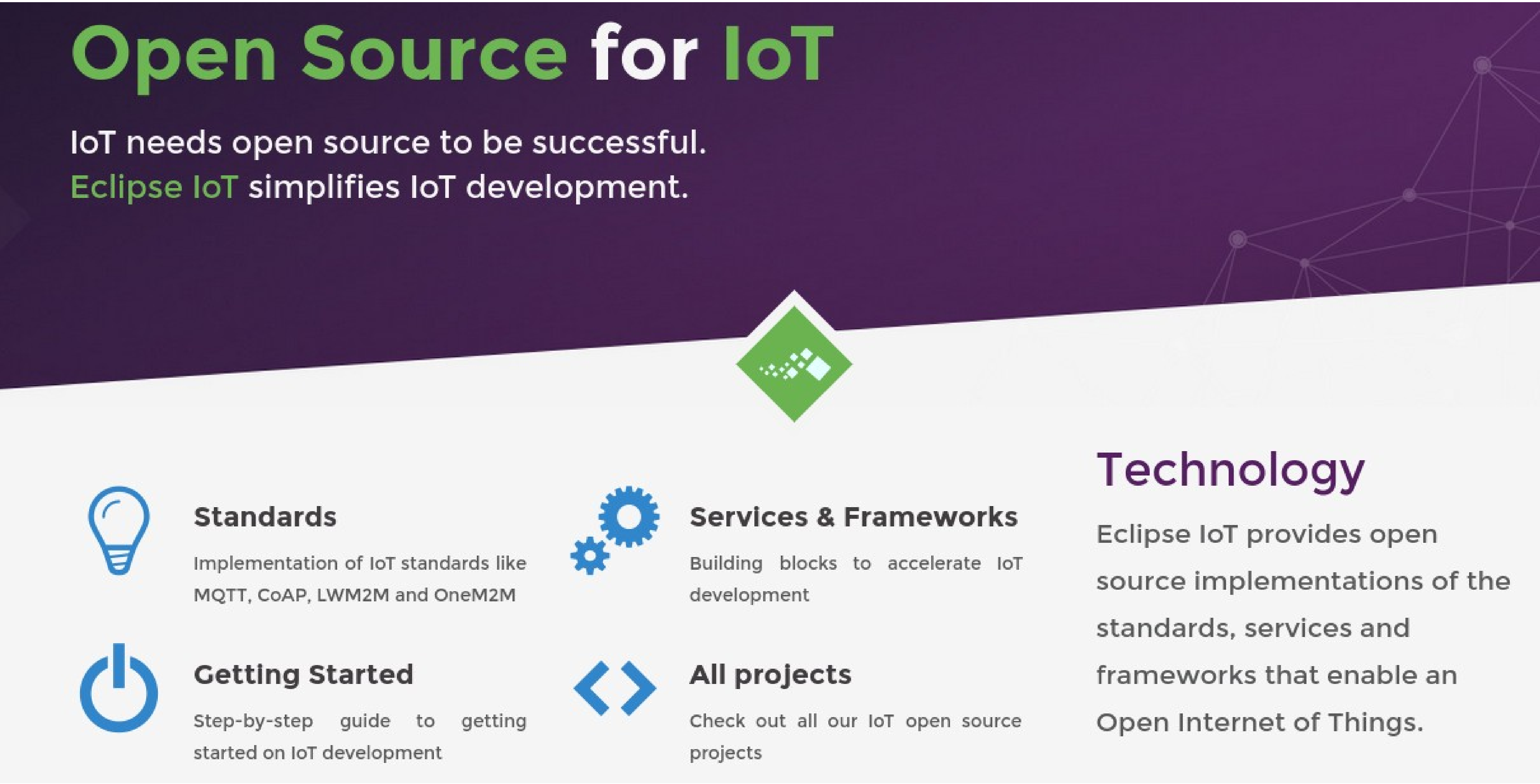


How to Connect things?

Machine to machine

- Protocols
- Data access
- Compatibility
- Brands
- Device type
- Security
- Open or proprietary
- Bandwidth

Eclipse http://iot.eclipse.org/



Open Source for IoT

IoT needs open source to be successful.
Eclipse IoT simplifies IoT development.

Standards

Implementation of IoT standards like MQTT, CoAP, LWM2M and OneM2M

Services & Frameworks

Building blocks to accelerate IoT development

Getting Started

Step-by-step guide to getting started on IoT development

All projects

Check out all our IoT open source projects

Technology

Eclipse IoT provides open source implementations of the standards, services and frameworks that enable an Open Internet of Things.

Standards

- OMA LightweightM2M
- LightweightM2M is principly a device management protocol, but it should be designed to be able to extend to meet the requirements of applications. LightweightM2M is not restricted to device management, it should be able transfer service / application data.
- OMA LightweightM2M (LWM2M) is an industry standard for device management of M2M/IoT devices. It heavily relies on CoAP and therefore is optimized for communications over sensor or cellular networks.
- OMA LWM2M provides an extensible object model that allows to enable application data exchanges in addition to the core device management features (firmware upgrade, connectivity monitoring, …)

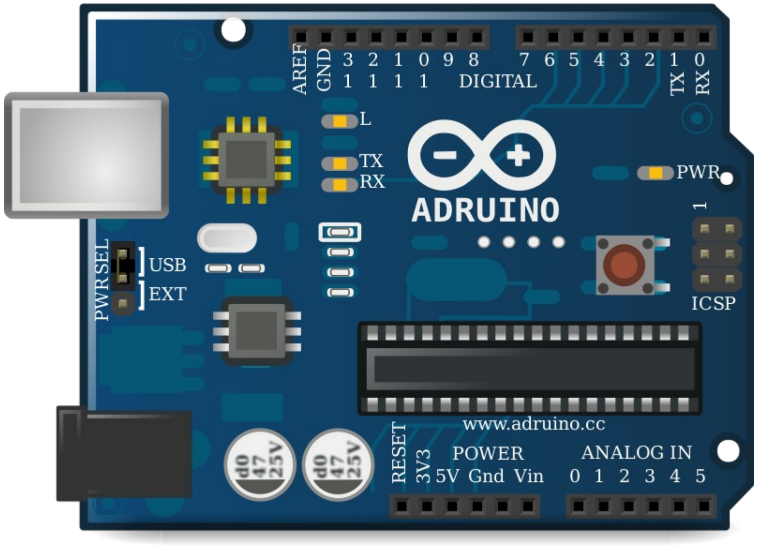
Standards

- CoAP
  - CoAP (Constrained Application Protocol) is a protocol specialized for use with constrained nodes and networks.
  - It implements the REST architectural style, it can be transparently mapped to HTTP. However, CoAP also provides features that go beyond HTTP such as native push notifications and group communication.

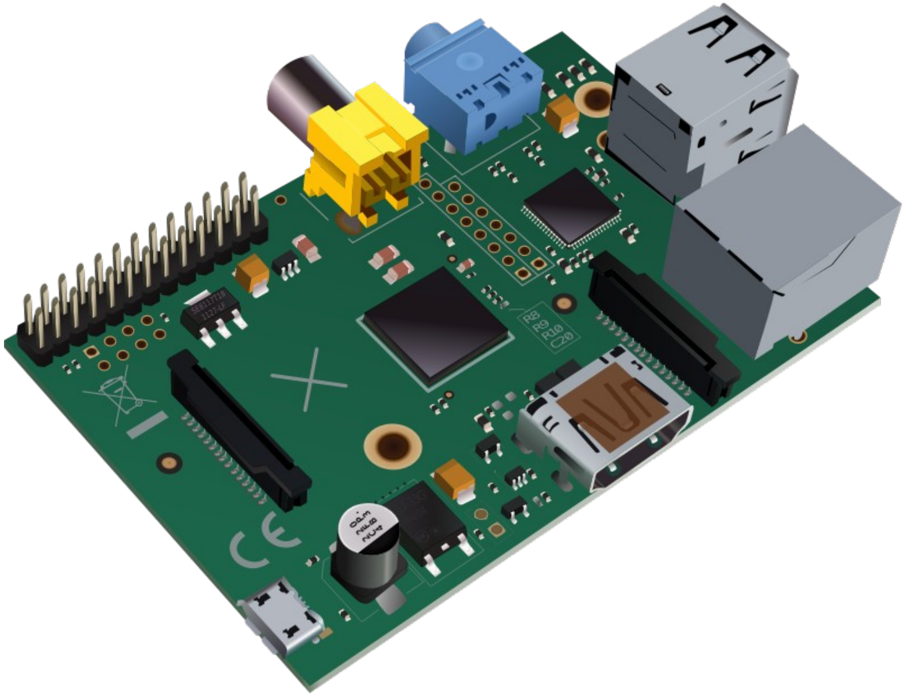
Standards

- Message Queuing Telemetry Transport (MQTT)
  - Protocol designed to connect the physical world devices and networks, with applications and middleware used in IT and Web development.
  - Lightweight
  - Publish/Subscribe
  - Wired/Wireless
  - Protocol used by Facebook chat!

Common hardware

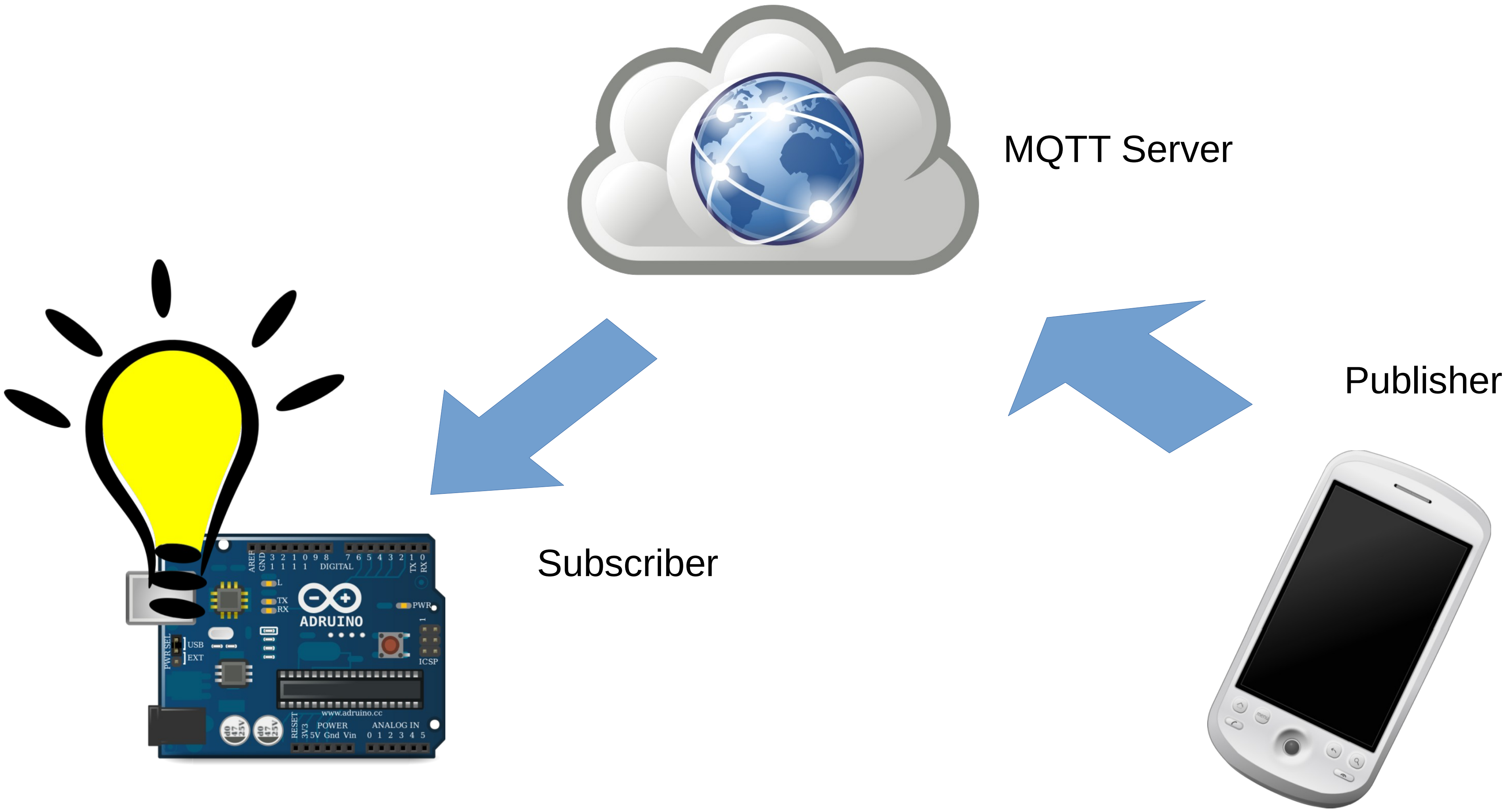


Arduino



Raspberry Pi

MQTT Implementation



MQTT Server

Publisher

Subscriber

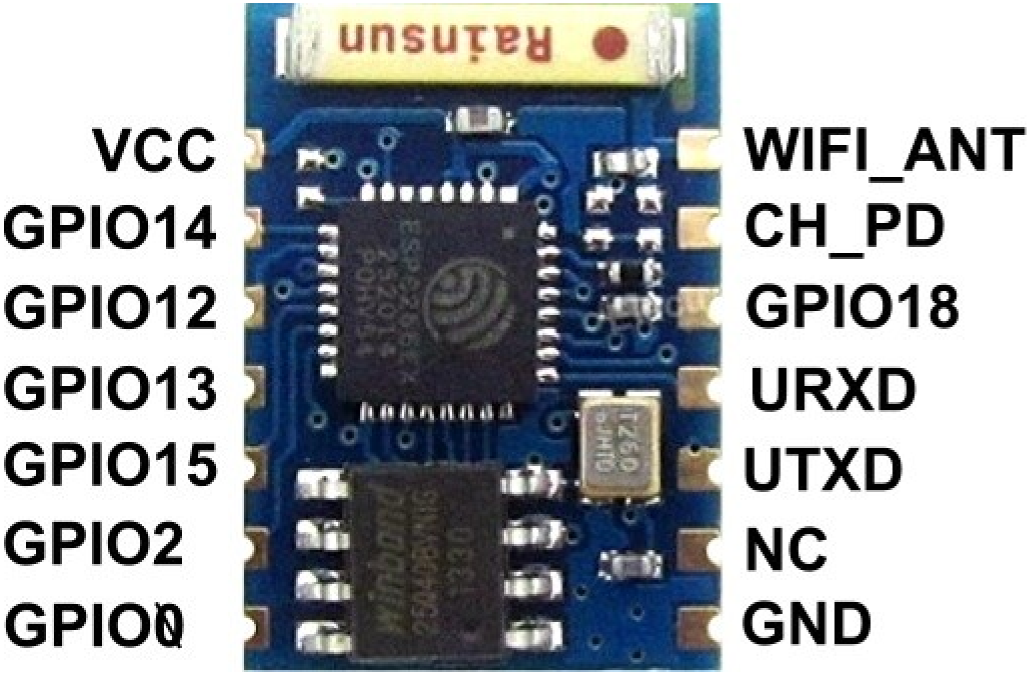
Create Mqtt Server

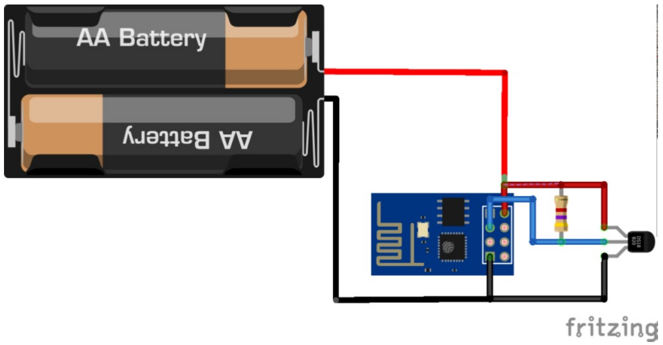
- Mosquitto Server
  - Open source
  - Can be installed locally

- Use a public server
  - Adafruit IO

How to get started

- Wifi is cheap
- Multiple options
- ESP8266 < $5





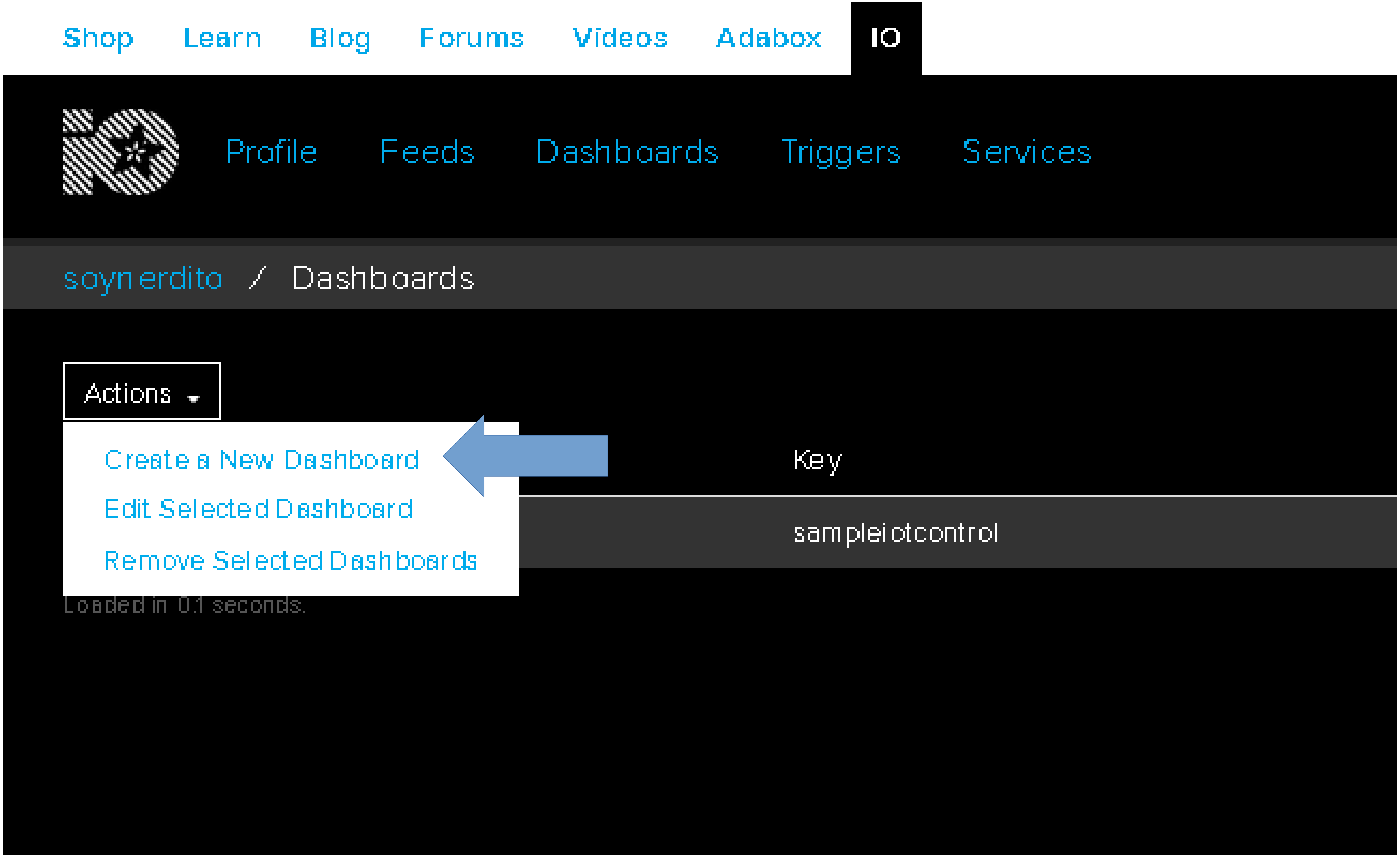
Workshop Goals

- Understand simple client/server architecture
- Know board ESP8266
  - Variants
- Install Arduino
- Compile project
- Connect Wifi
- Publish/Subscribe for Mqtt

Register/Create account

- https://io.adafruit.com
  - Create account

Create a Dashboard

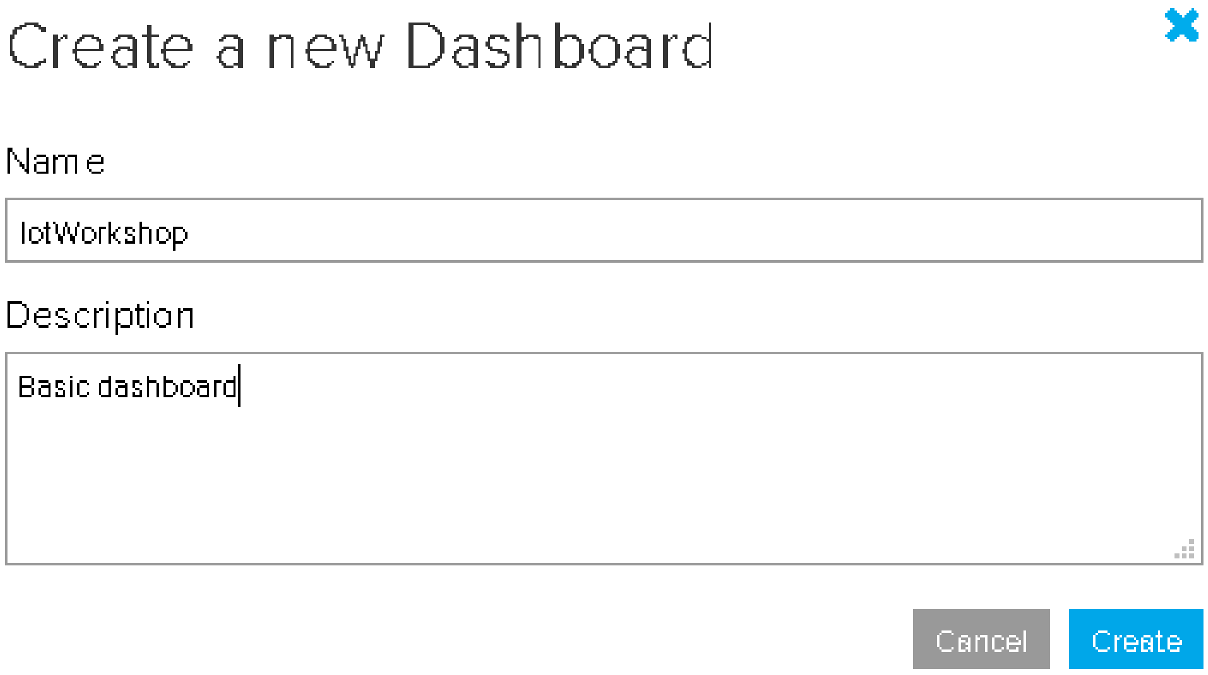


soynerdito / Dashboards

Key
sampleiotcontrol

Loaded in 0.1 seconds.

Follow the steps



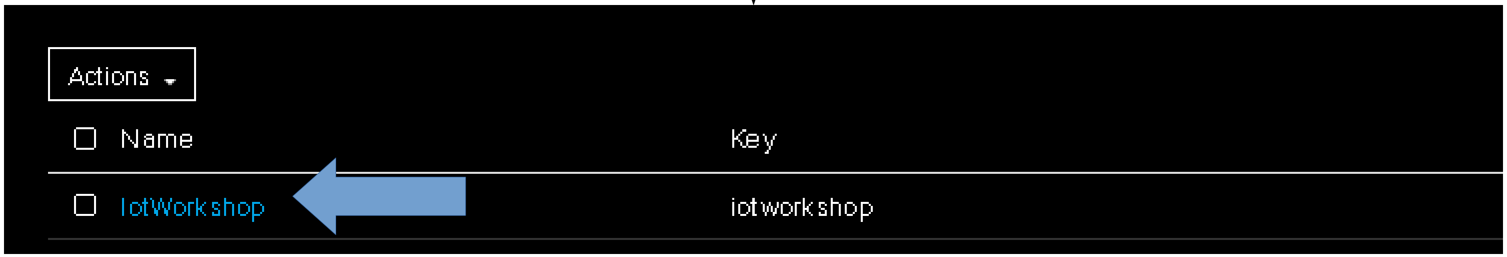
Create a new Dashboard

Name

IotWorkshop

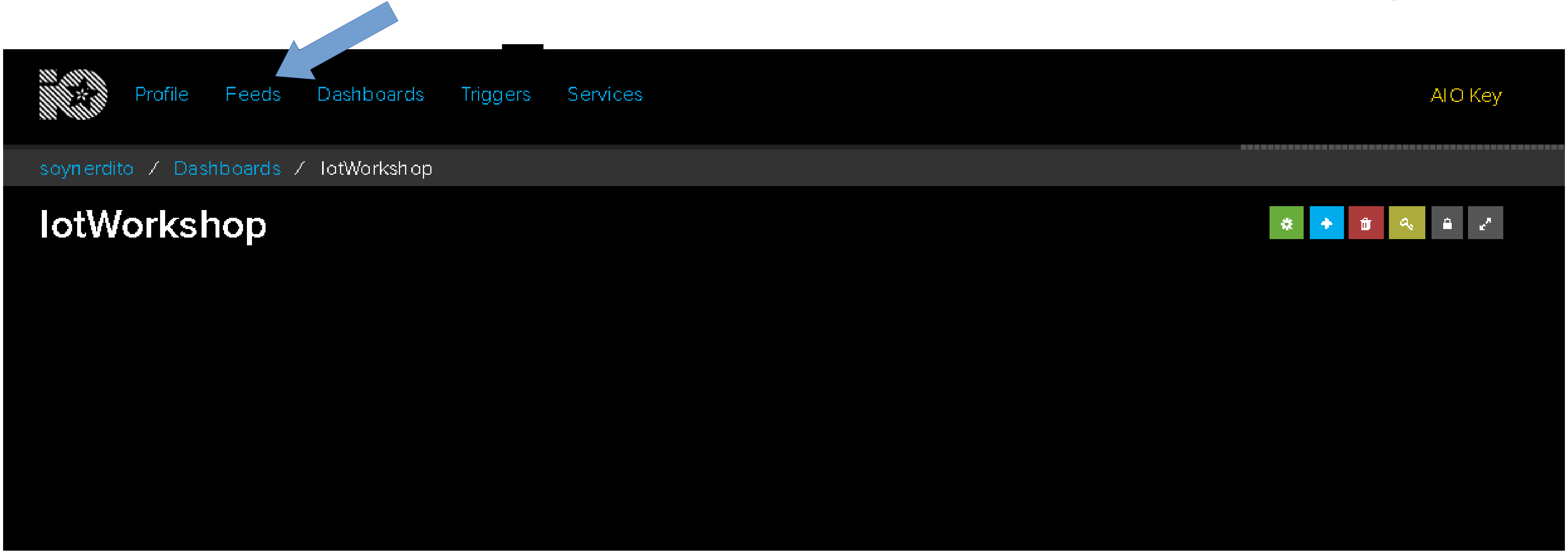
Description

Basic dashboard



□ Name	Key
□ IotWorkshop	iotworkshop

Configure



Profile Feeds Dashboards Triggers Services

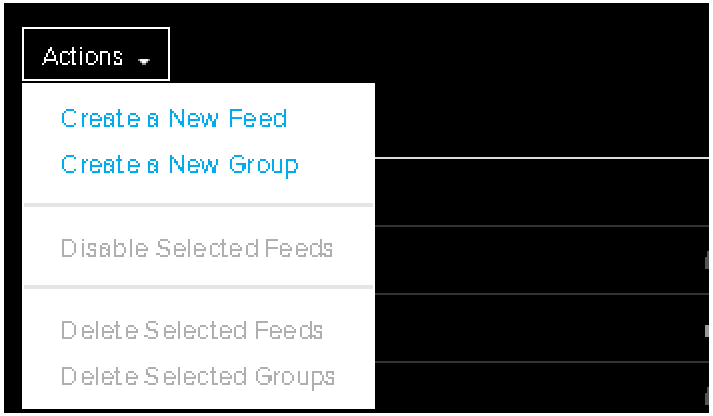
AIO Key

soynerdito / Dashboards / IotWorkshop

IotWorkshop

- Create a Feed to connect devices

New Feed



- A feed is used to represent a value

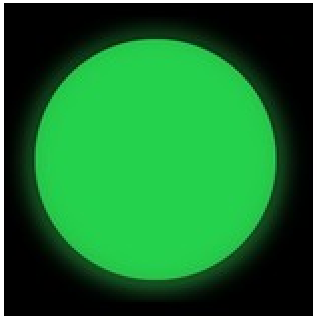
Mqtt Qos
What is Quality of Service?

- The Quality of Service (QoS) level is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message. There are 3 QoS levels in MQTT:
  - At most once (0)
  - At least once (1)
  - Exactly once (2).
- When you talk about QoS in MQTT, you need to consider the two sides of message delivery:
  - Message delivery form the publishing client to the broker.
  - Message delivery from the broker to the subscribing client.

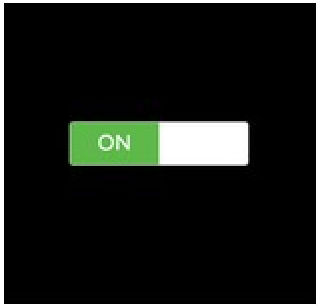
https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

Add controls to Dashboard

- Receive control



- Publish control



Internet Users by Country

- In 2014, nearly 75% (2.1 billion) of all internet users in the world (2.8 billion) live in the top 20 countries.
- The remaining 25% (0.7 billion) is distributed among the other 178 countries, each representing less than 1% of total users.

Live demos

Build demo
Making it blink

Wemos D1 mini

- ESP8266
- Programming with arduino
- Led Connected to pin D0

Dotnet new console

```
using System;
using System.Text;
using System.Threading;
using MQTTnet;
using MQTTnet.Client;
using MQTTnet.Implementations;

namespace mqttConsole
{
    class Program
    {
        static async System.Threading.Tasks.Task Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            // Use TCP connection.
            var options = new MqttClientOptionsBuilder()
                //.WithTcpServer("broker.hivemq.com", 1883) // Port is optional
                //.WithTcpServer("iot.eclipse.org", 1883) // Port is optional
                .WithTcpServer("test.mosquitto.org", 1883) // Port is optional
                .Build();
            // Create a new MQTT client.
            var factory = new MqttFactory();
            var mqttClient = factory.CreateMqttClient();


            mqttClient.ApplicationMessageReceived += (s, e) =>
            {
                Console.WriteLine("### RECEIVED APPLICATION MESSAGE ###");
                Console.WriteLine($"+ Topic = {e.ApplicationMessage.Topic}");
                Console.WriteLine($"+ Payload = {Encoding.UTF8.GetString(e.ApplicationMessage.Payload)}");
                Console.WriteLine($"+ QoS = {e.ApplicationMessage.QualityOfServiceLevel}");
                Console.WriteLine($"+ Retain = {e.ApplicationMessage.Retain}");
                Console.WriteLine();
            };

            mqttClient.Connected += async (s, e) =>
           {
               Console.WriteLine("### CONNECTED WITH SERVER ###");

                // Subscribe to a topic
                await mqttClient.SubscribeAsync(new TopicFilterBuilder().WithTopic("outTopic").Build());

               Console.WriteLine("### SUBSCRIBED ###");
           };
            await mqttClient.ConnectAsync(options);



            string inputLine = "";
            do
            {
                inputLine = Console.ReadLine();
                if (inputLine?.Length > 0)
                {
                    await mqttClient.PublishAsync(
                        new MqttApplicationMessageBuilder()
                        .WithTopic("inTopic")
                        .WithPayload(inputLine)
                        .Build() );
                }

            }
            while (inputLine?.Length > 0);

        }
    }
}

```

Project file

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <LangVersion>latest</LangVersion>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp2.1</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="MQTTnet" Version="2.8.4"/>
  </ItemGroup>

</Project>
```

Arduino example

```
/*
 Basic ESP8266 MQTT example

 This sketch demonstrates the capabilities of the pubsub library in combination
 with the ESP8266 board/library.

 It connects to an MQTT server then:
  - publishes "hello world" to the topic "outTopic" every two seconds
  - subscribes to the topic "inTopic", printing out any messages
    it receives. NB - it assumes the received payloads are strings not binary
  - If the first character of the topic "inTopic" is an 1, switch ON the ESP Led,
    else switch it off

 It will reconnect to the server if the connection is lost using a blocking
 reconnect function. See the 'mqtt_reconnect_nonblocking' example for how to
 achieve the same result without blocking the main loop.

 To install the ESP8266 board, (using Arduino 1.6.4+):
  - Add the following 3rd party board manager under "File -> Preferences -> Additional Boards Manager URLs":
       http://arduino.esp8266.com/stable/package_esp8266com_index.json
  - Open the "Tools -> Board -> Board Manager" and click install for the ESP8266"
  - Select your ESP8266 in "Tools -> Board"

*/

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.

const char* ssid = "....";
const char* password = "....";
const char* mqtt_server = "test.mosquitto.org";

WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;

void setup() {
  pinMode(BUILTIN_LED, OUTPUT);     // Initialize the BUILTIN_LED pin as an output.
  pinMode(D0, OUTPUT);
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    client.publish("outTopic", "Appagando");
    digitalWrite(D0, LOW);   // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is acive low on the ESP-01)
  } else {
    digitalWrite(D0, HIGH);  // Turn the LED off by making the voltage HIGH
    client.publish("outTopic", "Prendiendo");
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      //client.publish("outTopic", "hello world");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    ++value;
    //snprintf (msg, 75, "hello world #%ld", value);
   // Serial.print("Publish message: ");
    //Serial.println(msg);
    //client.publish("outTopic", msg);
  }
}
```

Arduino source wifiManager

```
#include <ESP8266WiFi.h>          //https://github.com/esp8266/Arduino

//needed for library
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>          //https://github.com/tzapu/WiFiManager

#include <ESP8266WiFi.h>
#include <PubSubClient.h>
//Mqtt stuff
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[50];
int value = 0;
const char* mqtt_server = "test.mosquitto.org";

//for LED status
#include <Ticker.h>
Ticker ticker;

void tick()
{
  //toggle state
  int state = digitalRead(BUILTIN_LED);  // get the current state of GPIO1 pin
  digitalWrite(BUILTIN_LED, !state);     // set pin to the opposite state
}

//gets called when WiFiManager enters configuration mode
void configModeCallback (WiFiManager *myWiFiManager) {
  Serial.println("Entered config mode");
  Serial.println(WiFi.softAPIP());
  //if you used auto generated SSID, print it
  Serial.println(myWiFiManager->getConfigPortalSSID());
  //entered config mode, make led toggle faster
  ticker.attach(0.2, tick);
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  //For mqtt stuff
   pinMode(D0, OUTPUT);

  //set led pin as output
  pinMode(BUILTIN_LED, OUTPUT);
  // start ticker with 0.5 because we start in AP mode and try to connect
  ticker.attach(0.6, tick);

  //WiFiManager
  //Local intialization. Once its business is done, there is no need to keep it around
  WiFiManager wifiManager;
  //reset settings - for testing
  //wifiManager.resetSettings();

  //set callback that gets called when connecting to previous WiFi fails, and enters Access Point mode
  wifiManager.setAPCallback(configModeCallback);

  //fetches ssid and pass and tries to connect
  //if it does not connect it starts an access point with the specified name
  //here  "AutoConnectAP"
  //and goes into a blocking loop awaiting configuration
  if (!wifiManager.autoConnect()) {
    Serial.println("failed to connect and hit timeout");
    //reset and try again, or maybe put it to deep sleep
    ESP.reset();
    delay(1000);
  }

  //if you get here you have connected to the WiFi
  Serial.println("connected...yeey :)");
  ticker.detach();
  //keep LED on
  digitalWrite(BUILTIN_LED, LOW);

  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}


void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  // Switch on the LED if an 1 was received as first character
  if ((char)payload[0] == '1') {
    client.publish("outTopic", "Appagando");
    digitalWrite(D0, LOW);   // Turn the LED on (Note that LOW is the voltage level
    // but actually the LED is on; this is because
    // it is acive low on the ESP-01)
  } else {
    digitalWrite(D0, HIGH);  // Turn the LED off by making the voltage HIGH
    client.publish("outTopic", "Prendiendo");
  }

}

void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      // Once connected, publish an announcement...
      //client.publish("outTopic", "hello world");
      // ... and resubscribe
      client.subscribe("inTopic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  long now = millis();
  if (now - lastMsg > 2000) {
    lastMsg = now;
    ++value;
    //snprintf (msg, 75, "hello world #%ld", value);
   // Serial.print("Publish message: ");
    //Serial.println(msg);
    //client.publish("outTopic", msg);
  }
}
```