Session 6

Agenda

01 **HTTP Protocol**

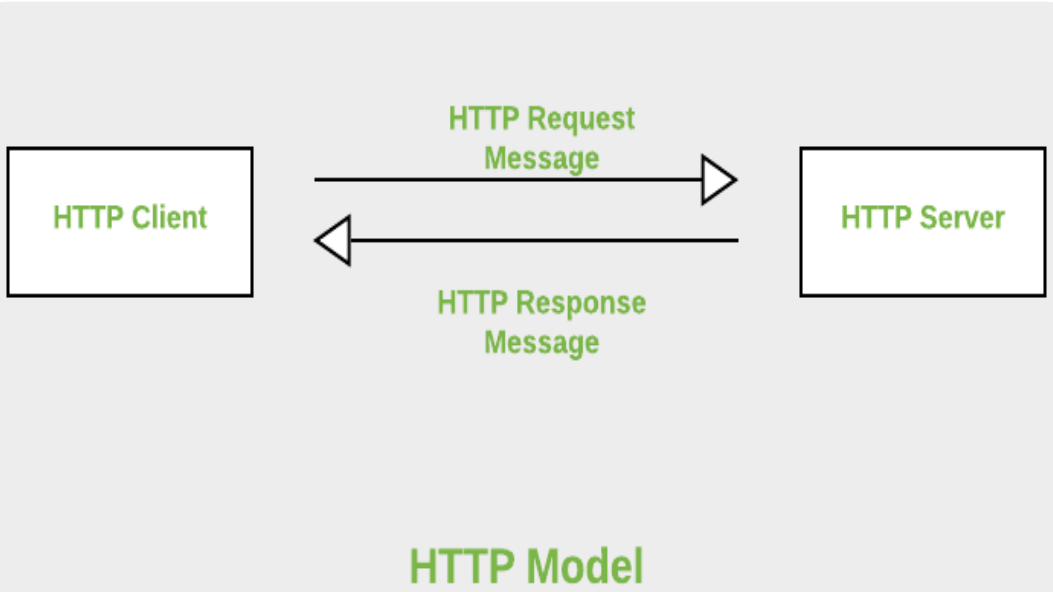
1. How it works
2. thingspeak.com

02 **MQTT Protocol**

How it works

ubidots.com

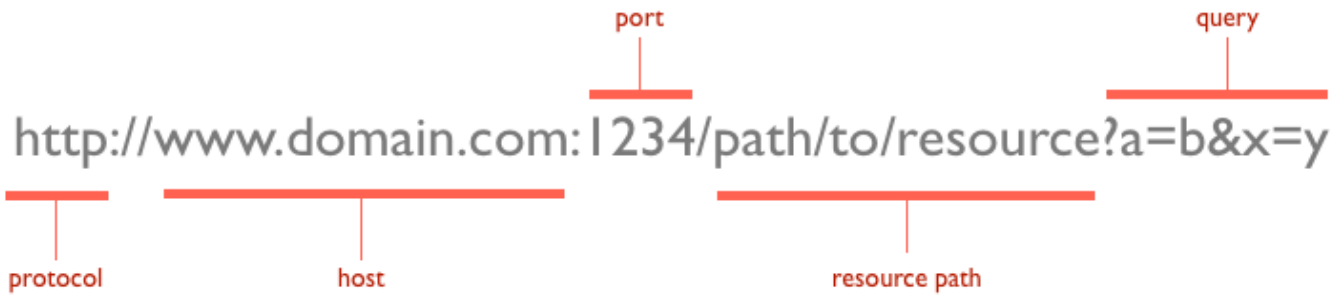
HTTP



- Hyper Text Transfer Protocol
- This protocol has formed the foundation of data communication over the web
- It is the most common protocol that is used for IoT devices when there is a lot of data to be published

HTTP

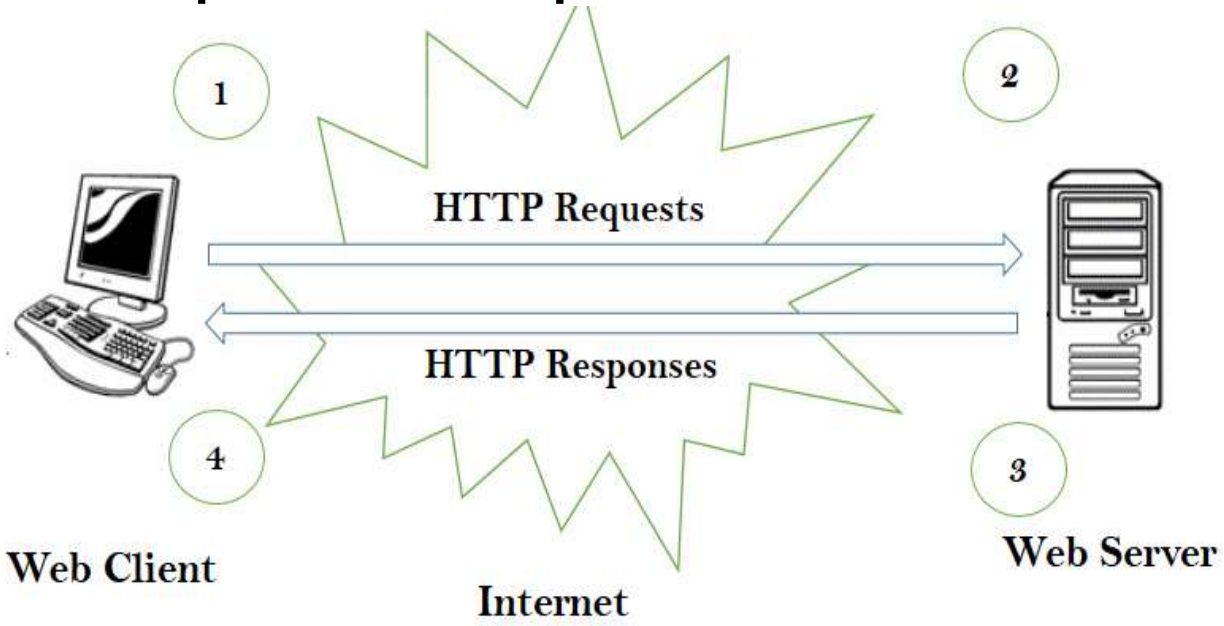
Stands for Uniform Resource Locator. A URL is nothing more than the address of a given unique resource on the Web.



- This protocol is responsible for the action that a server must take while sending information over the network
- When a URL is being entered into the browser, this protocol sends an HTTP request to the server and then an HTTP response is sent back to the browser
- This protocol is also responsible for the controlling of webpages on the World Wide Web

HTTP Request / Response

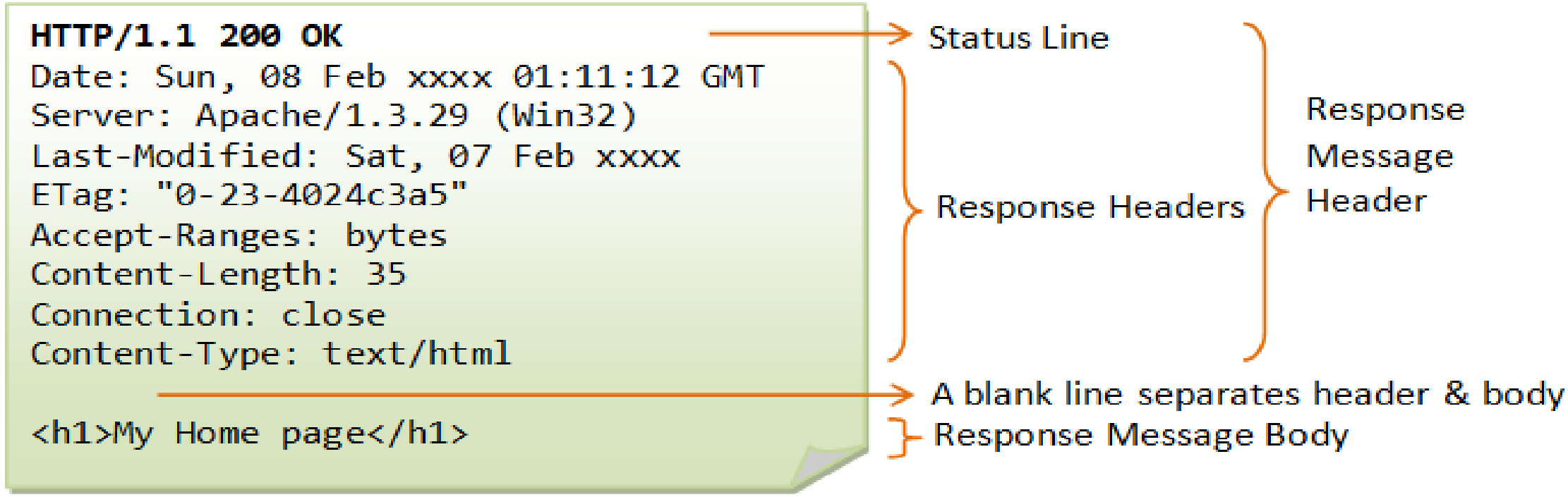
- **Communication between clients and servers is done by requests and responses:**



1. **A client (a browser) sends an HTTP request**
2. **A web server receives the request**
3. **The server runs an application to process the request**
4. **The server returns an HTTP response (output) to the browser**
5. **The client (the browser) receives the response**

HTTP GET Request

HTTP Response

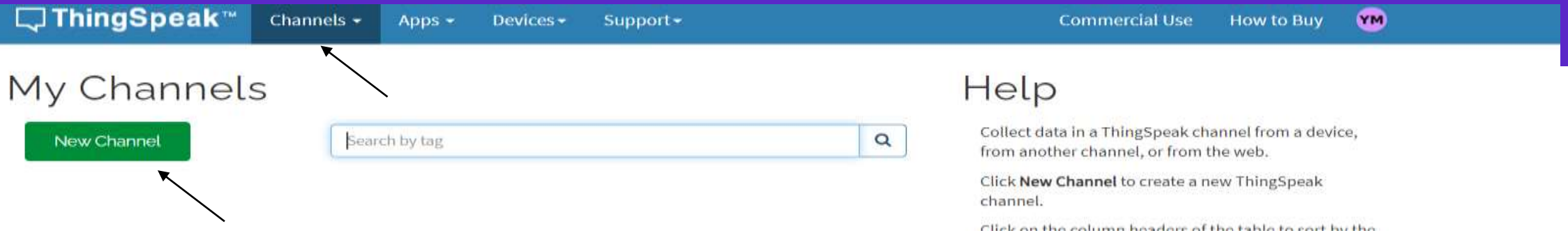


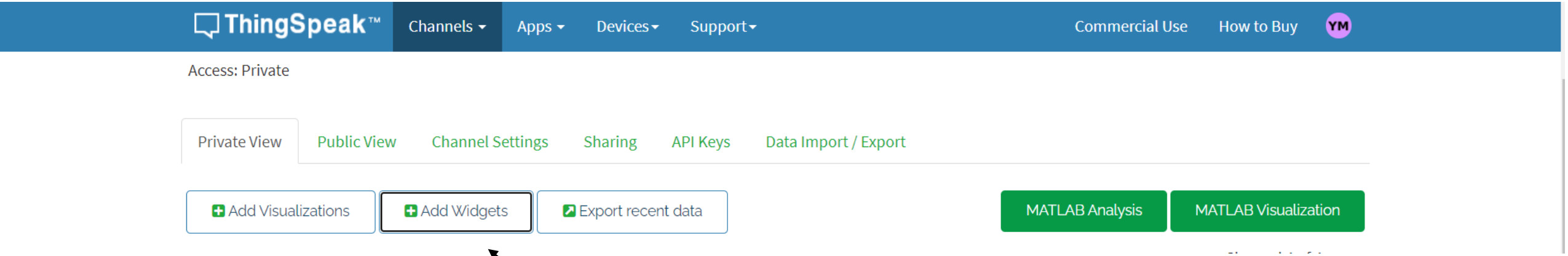
Thingspeak.com

Practice 1:

1. **Create account on Thingspeak.com.**
2. **Create a new channel as a server.**
3. **Get the API keys and write the distance read by an ultrasonic sensor in the channel.**

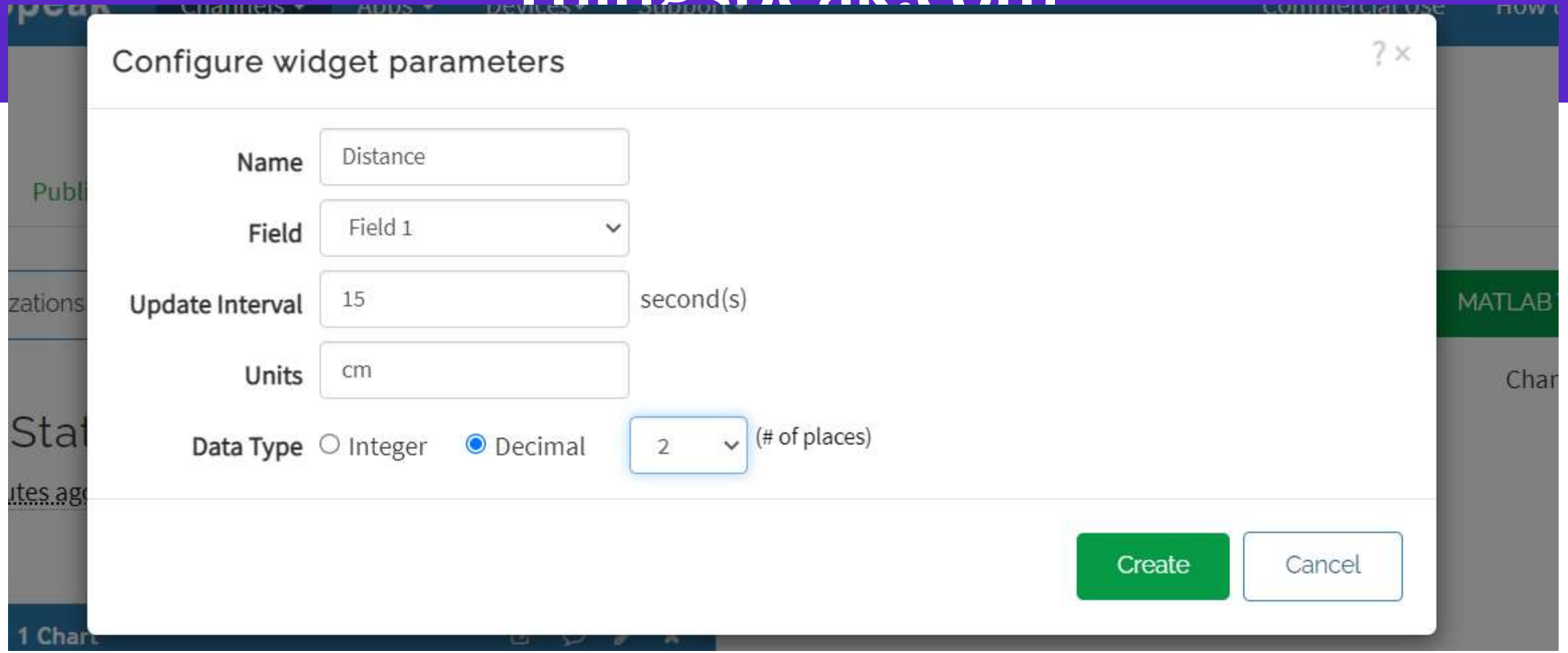
Thingspeak.com



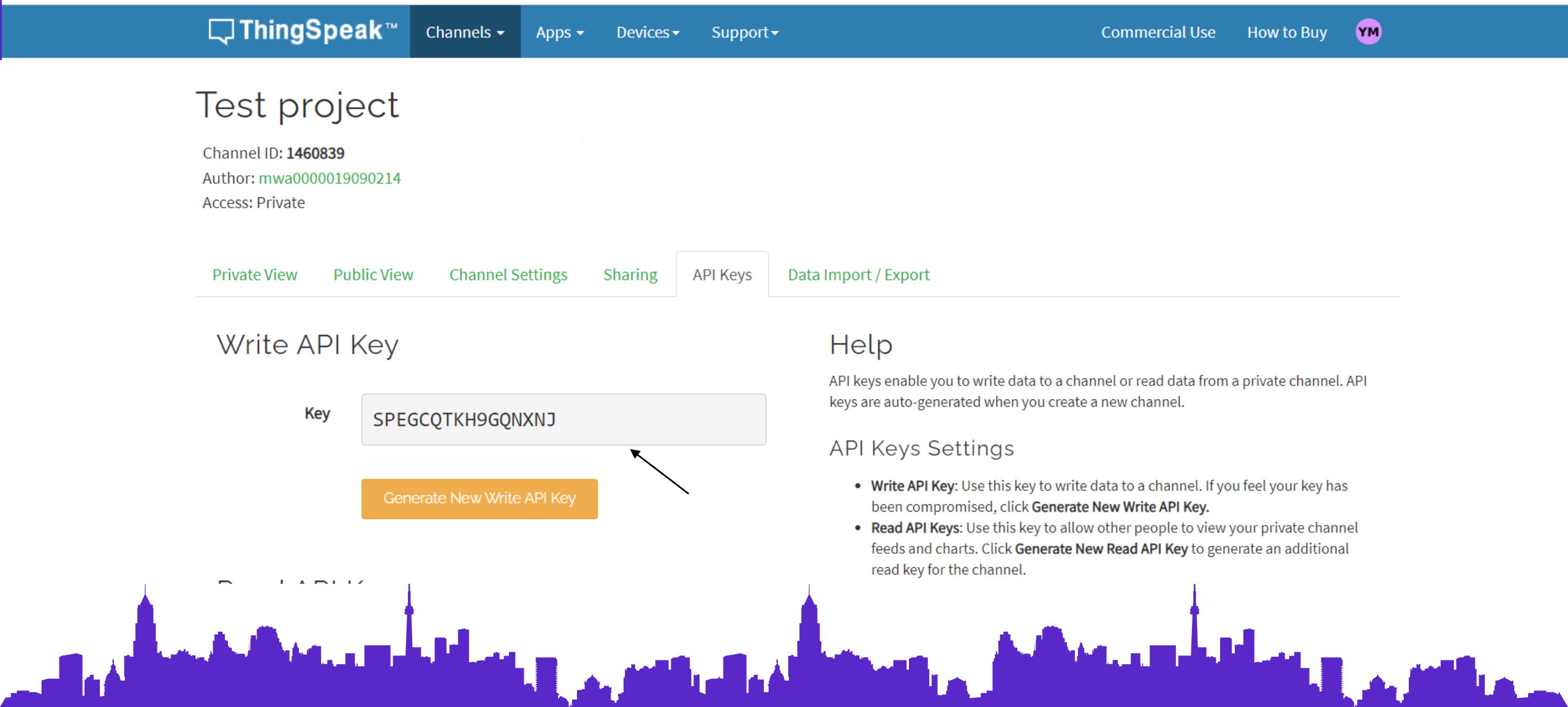


Thingspeak.com

Thingspeak.com



Thingspeak.com



Practice 1

Components:

- **nodeMCU**
- **Ultrasonic sensor**

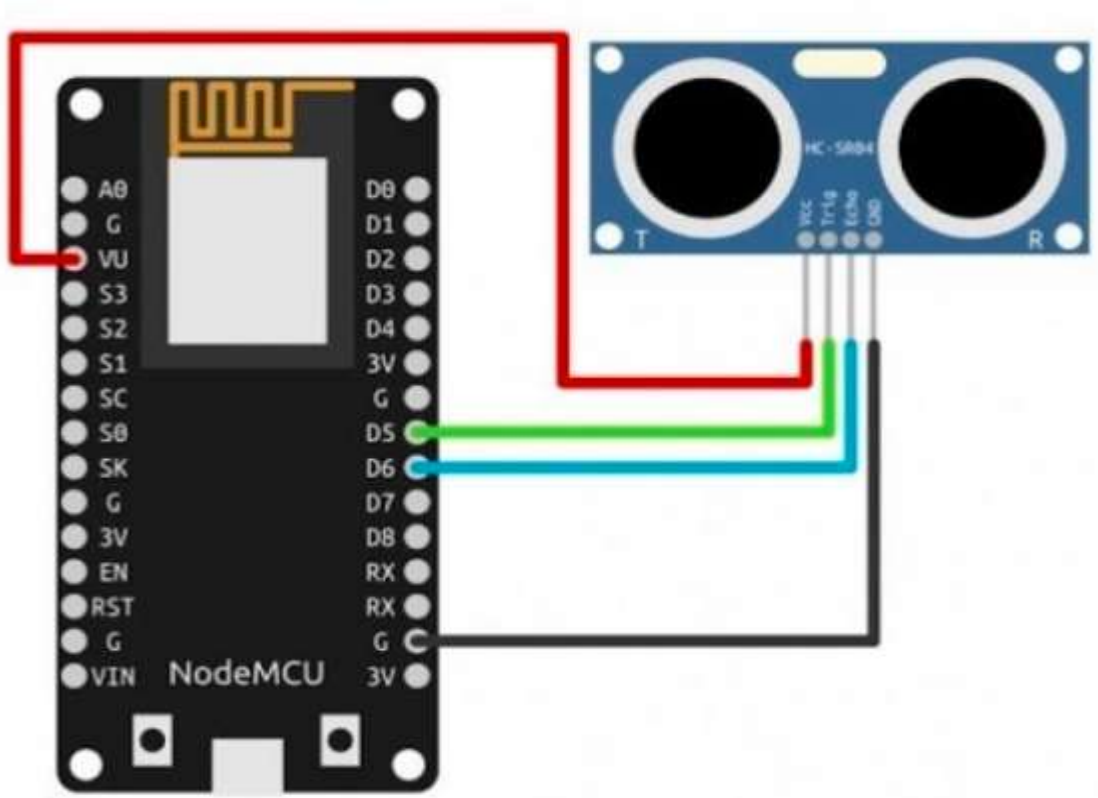
description:

- **Upload ultrasonic data to internet using thinkspeak**

Practice 1 Solution

sketch_aug21a §

```
#include <ESP8266WiFi.h>
String apiWritekey = "WRITE API"; // replace with
const char* ssid = "Baraka"; // your wifi SSID nam
const char* password = "00000000" ;// wifi pasword
const char* server = "api.thingspeak.com";
int echoPin=D6;
int trigPin=D5;
int duration;
int distance;
WiFiClient client;
```



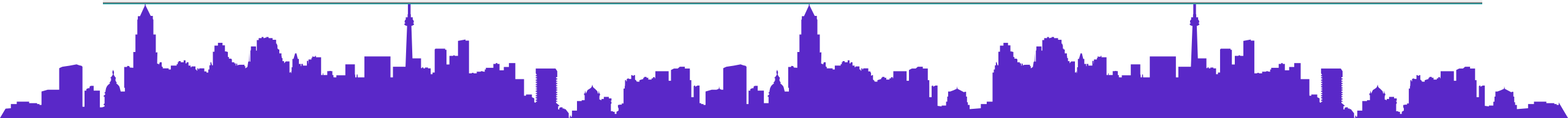
Practice 1 solution

```
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(115200);
  WiFi.disconnect();
  delay(10);
  WiFi.begin(ssid, password);

  Serial.println();
  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.print("NodeMcu connected to wifi...");
  Serial.println(ssid);
  Serial.println();
}
```

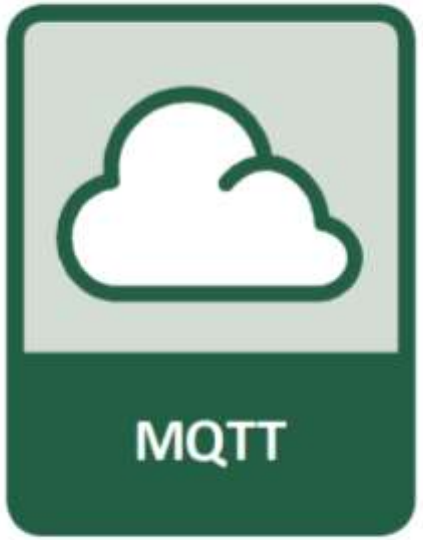
```
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2;
  //String val;
  if (client.connect(server,80))
  {  /*if (Serial.available()) {
   val=Serial.readString();
  }*/
    String tsData = apiWritekey;
           tsData +="&field1=";
           tsData += String(distance);
           tsData += "\r\n\r\n";
```

Practice 1 solution

```
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+apiWritekey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");  // the 2 carriage returns indicate closing of Header fields & starting of data
    client.print(tsData);
    Serial.print("Distance ");
    Serial.print(distance);
    Serial.println("uploaded to Thingspeak server....");
  }
  client.stop();
 // thingspeak needs minimum 15 sec delay between updates
// delay(15000);
}
```



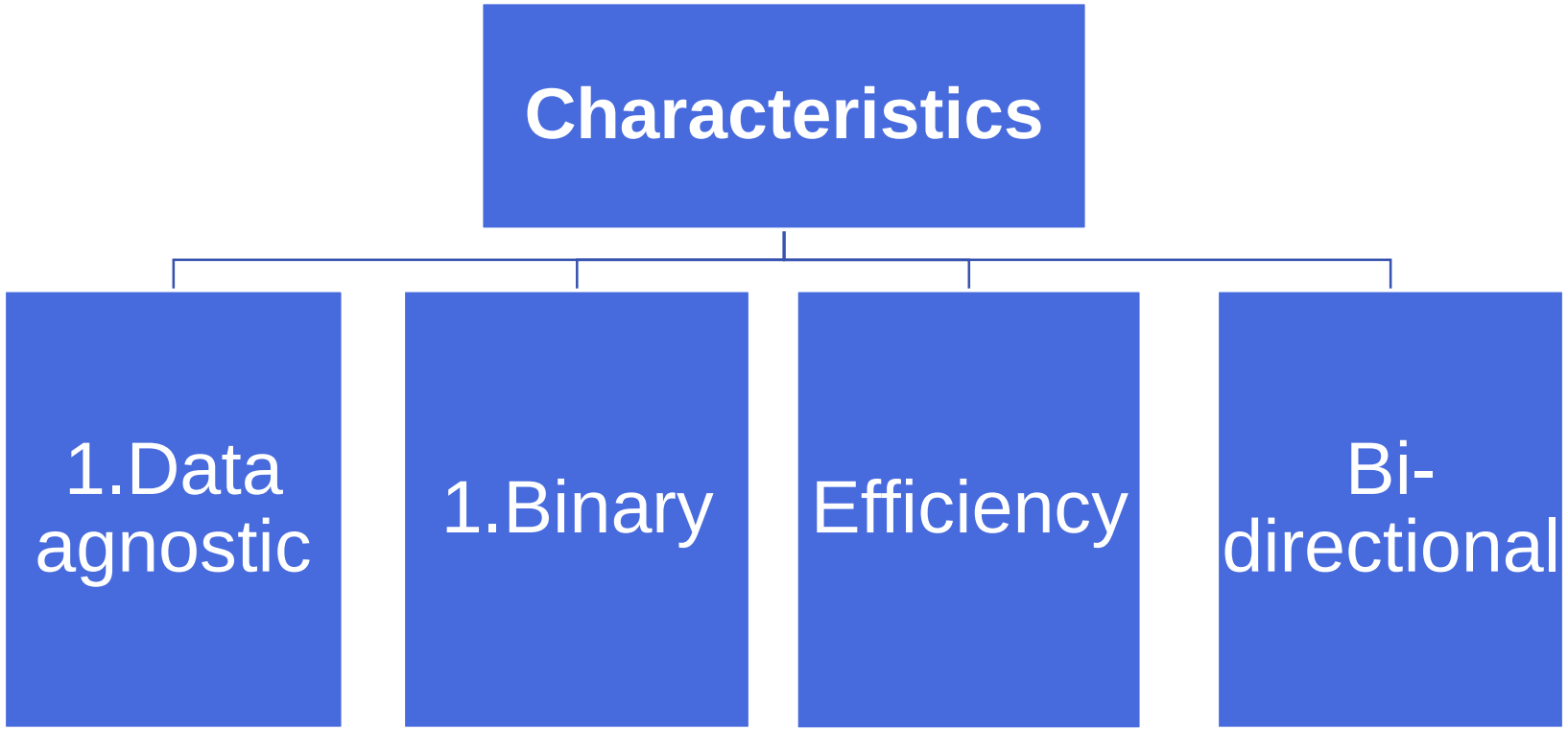
MQTT



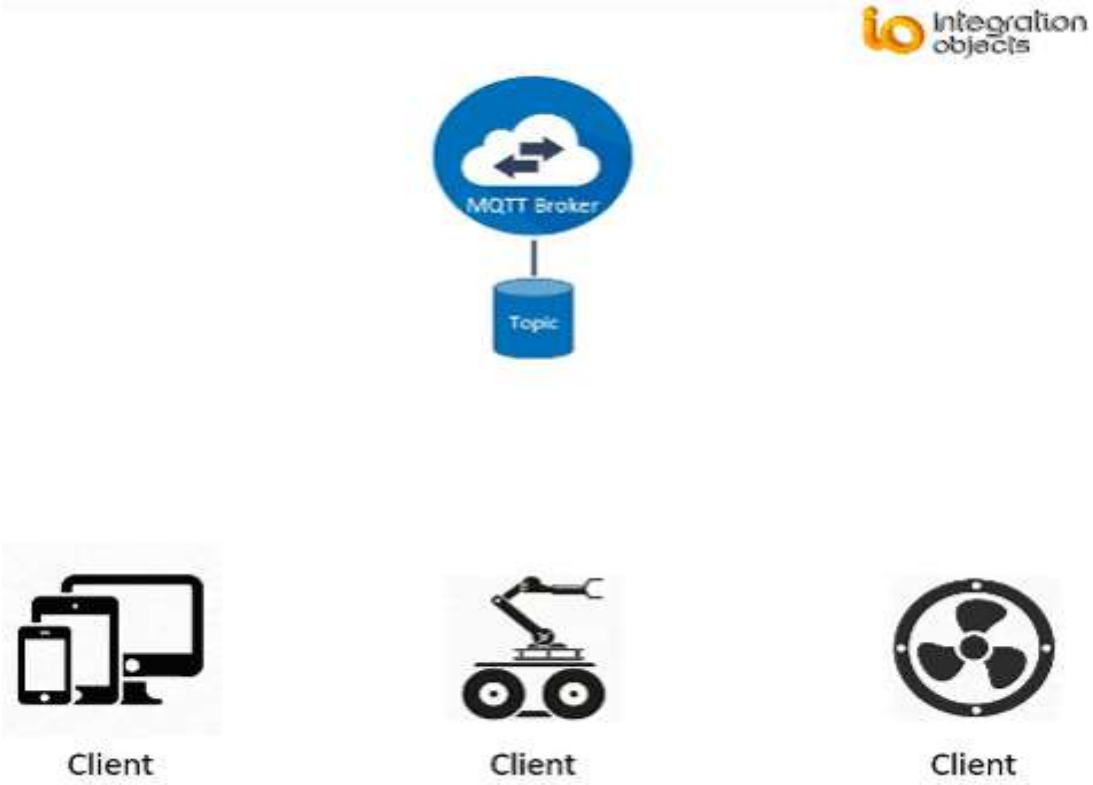
- **Message Queuing Telemetry Transport**
- **This protocol was invented because of the need of a protocol for minimal battery loss and minimal bandwidth**

Bandwidth is measured as the amount of data that can be transferred from one point to another within a network in a specific amount of time

MQTT

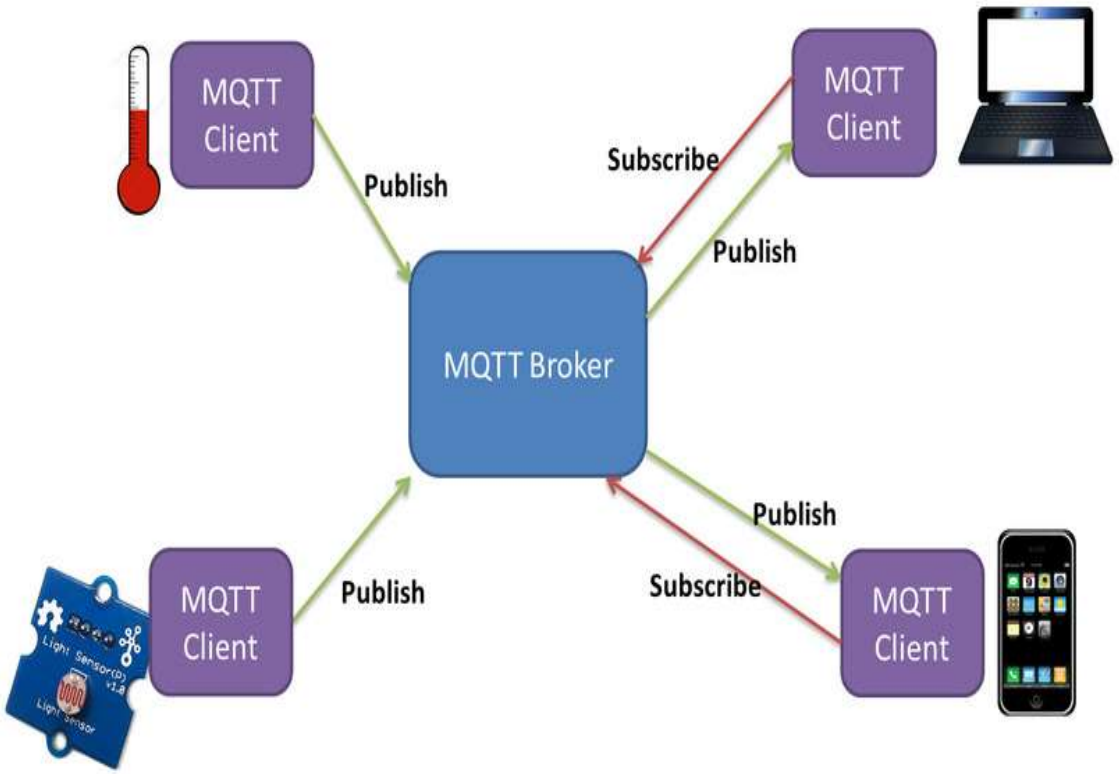

Characteristics
1.Data agnostic
1.Binary
Efficiency
Bi-directional

MQTT Publish / Subscribe



- The pub/sub model decouples the client that sends a message (the publisher) from the client or clients that receive the messages (the subscribers)
- The connection is handled by a third component (the broker)
- The job of the broker is to filter all incoming messages and distribute them correctly to subscribers

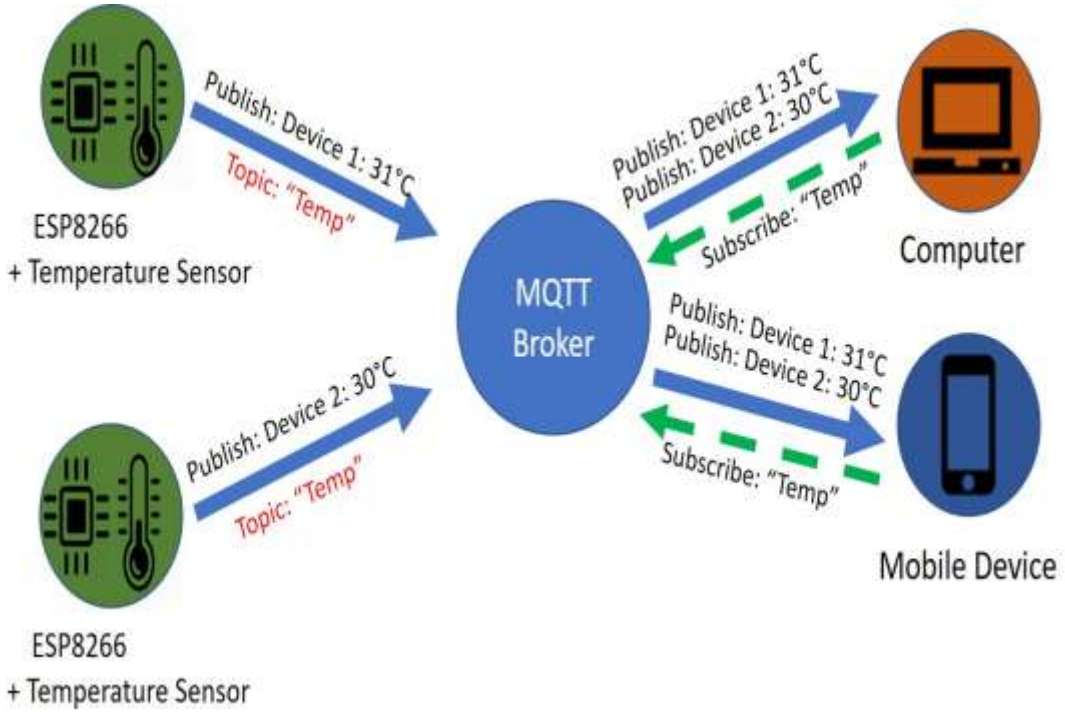
MQTT Publish / Subscribe



Clients:

- **Publish client.**
- **Subscribe client.**
- **Both publish and subscribe client.**

MQTT Publish / Subscribe



The most important aspect of pub/sub is the decoupling of the publisher of the message from the recipient (subscriber)

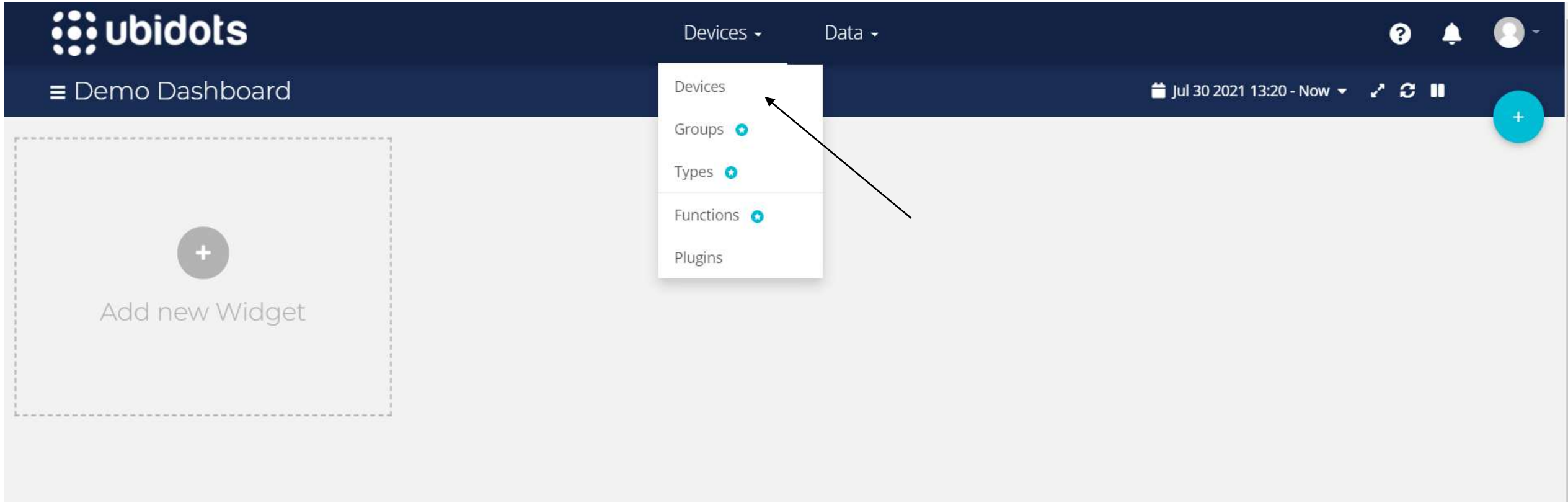
- **Space decoupling “no need to know the port”.**
- **Time decoupling.**
- **Synchronization decoupling.**

Ubidots.com

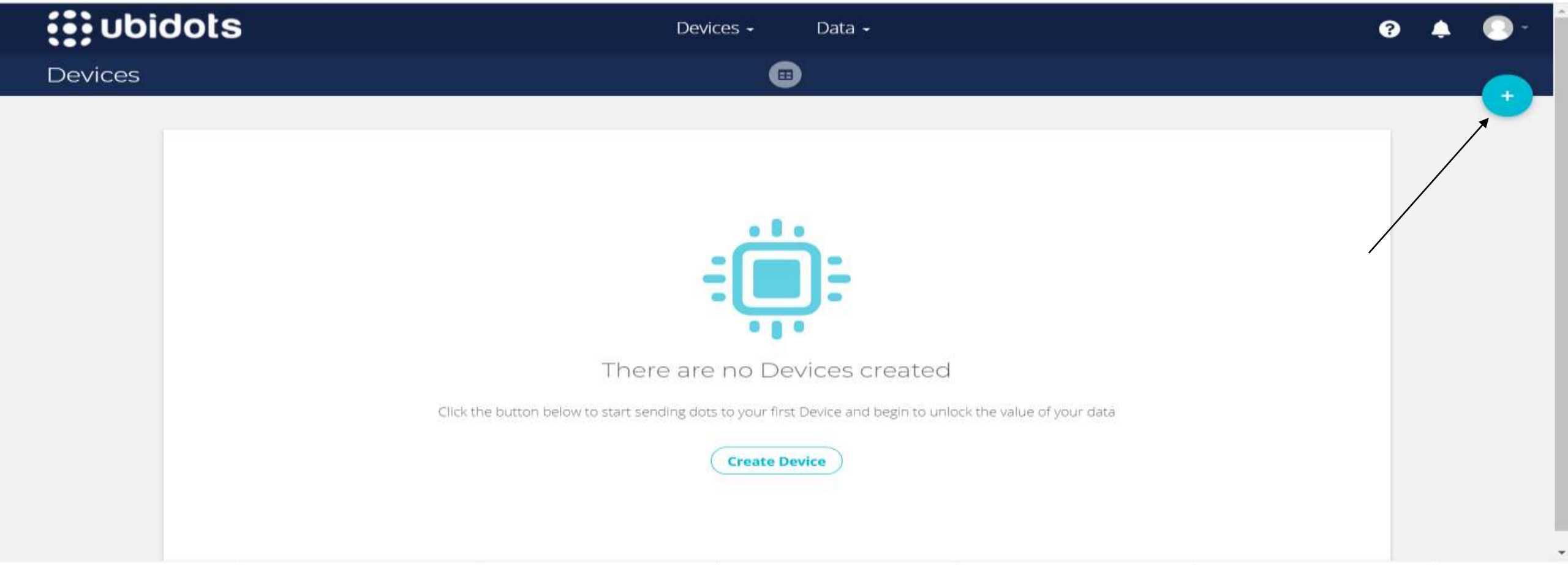
Practice 2:

1. **Create account on**
2. **Download Ubidots-MQTT-Esp library**
3. **Include the previous library in Arduino IDE**
4. **From Arduino IDE in Manage libraries download PubSubClient library**
5. **Enter a value on serial and see results on ubidots using ubidots widgets.**

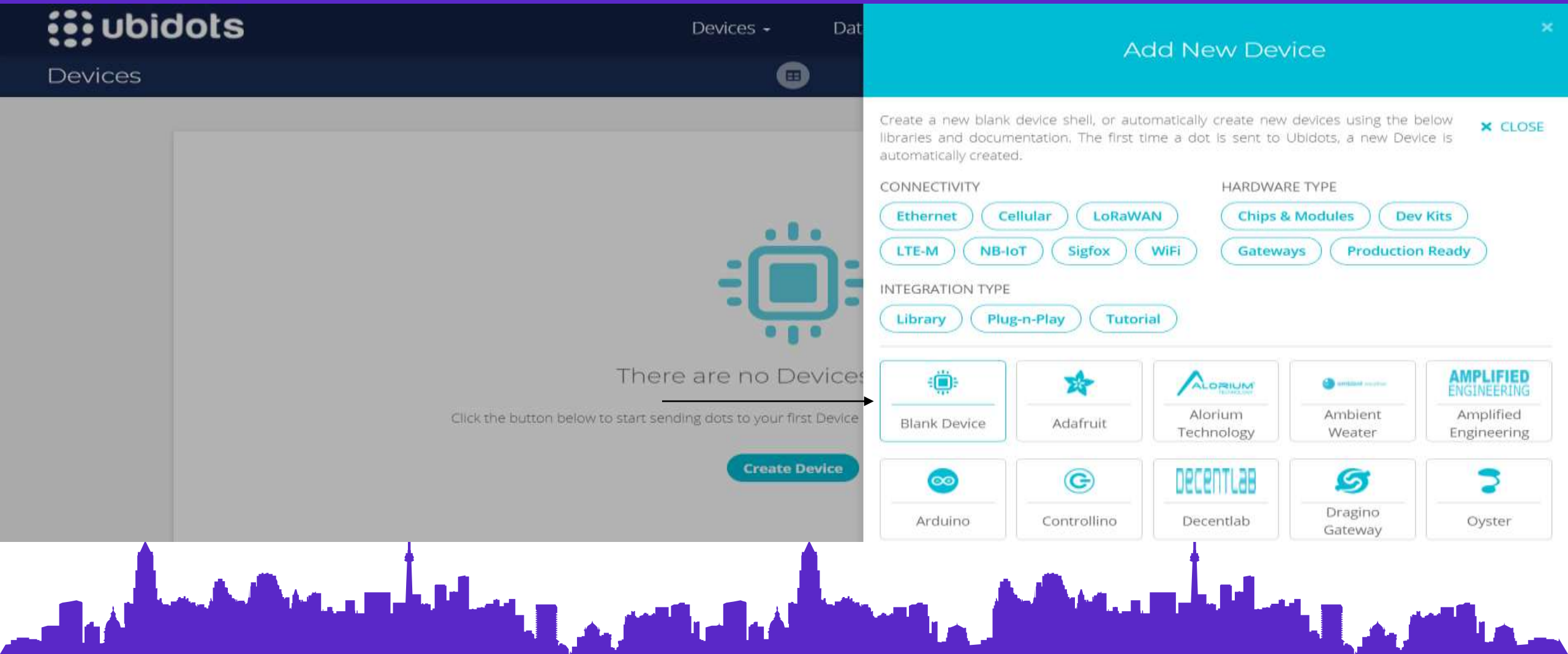
Ubidots.com


ubidots
Devices
Data
Demo Dashboard
Jul 30 2021 13:20 - Now
Devices
Groups
Types
Functions
Plugins
Add new Widget

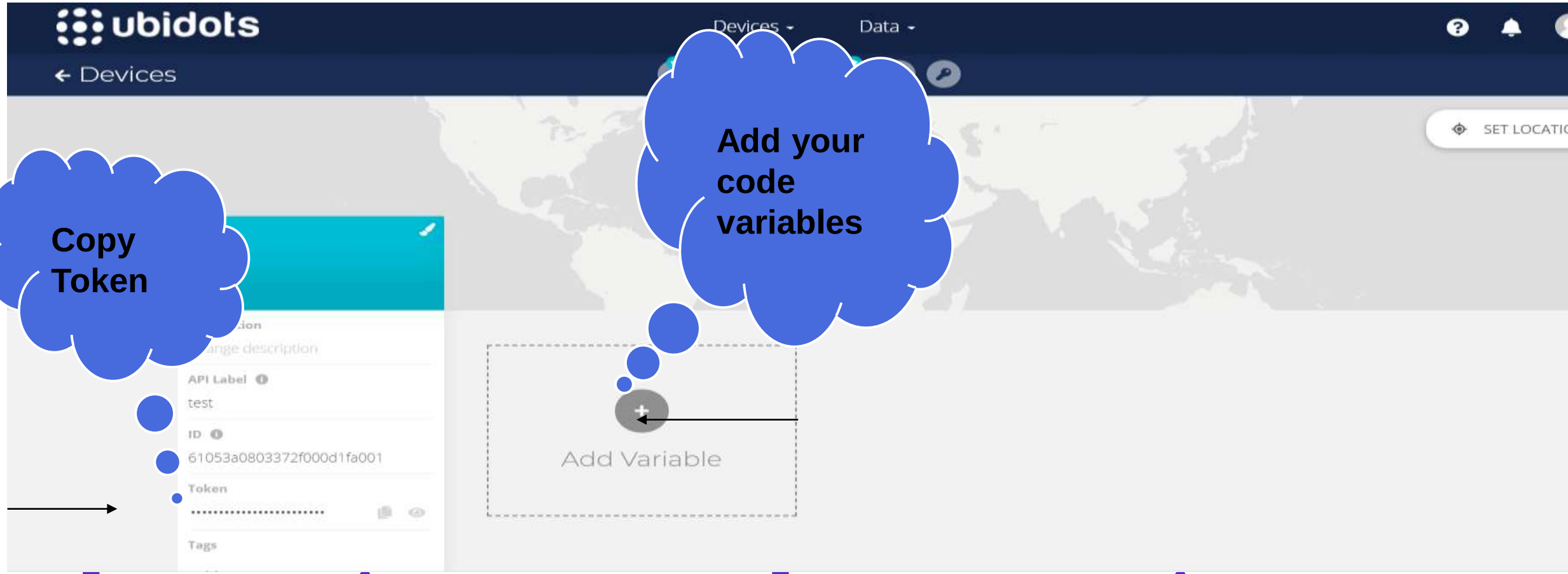
Ubidots.com


ubidots
Devices
Data
Devices
+
There are no Devices created
Click the button below to start sending dots to your first Device and begin to unlock the value of your data
Create Device

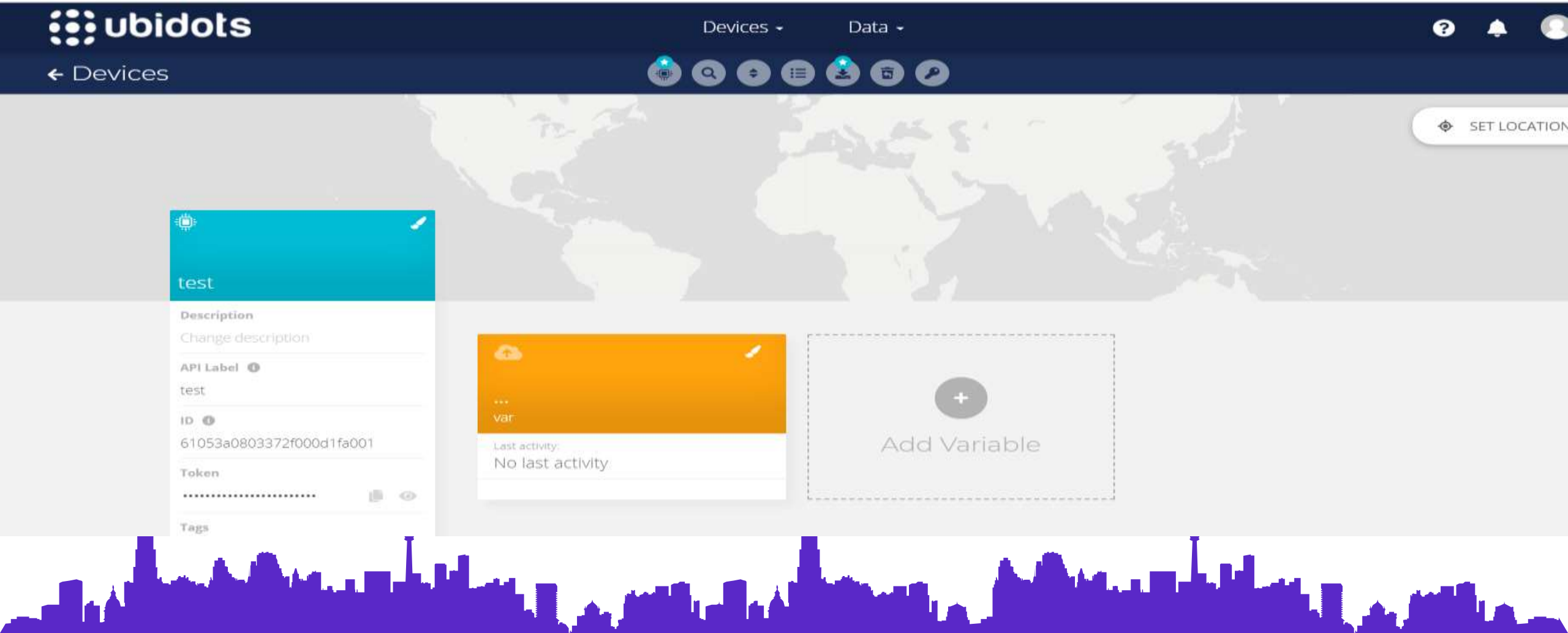
Ubidots.com


ubidots
Devices
Devices
There are no Devices
Click the button below to start sending dots to your first Device
Create Device
Add New Device
Create a new blank device shell, or automatically create new devices using the below libraries and documentation. The first time a dot is sent to Ubidots, a new Device is automatically created.
CLOSE
CONNECTIVITY
Ethernet
Cellular
LoRaWAN
LTE-M
NB-IoT
Sigfox
WiFi
HARDWARE TYPE
Chips & Modules
Dev Kits
Gateways
Production Ready
INTEGRATION TYPE
Library
Plug-n-Play
Tutorial
Blank Device
Adafruit
Alorium Technology
Ambient Weater
Amplified Engineering
Arduino
Controllino
Decentlab
Dragino Gateway
Oyster

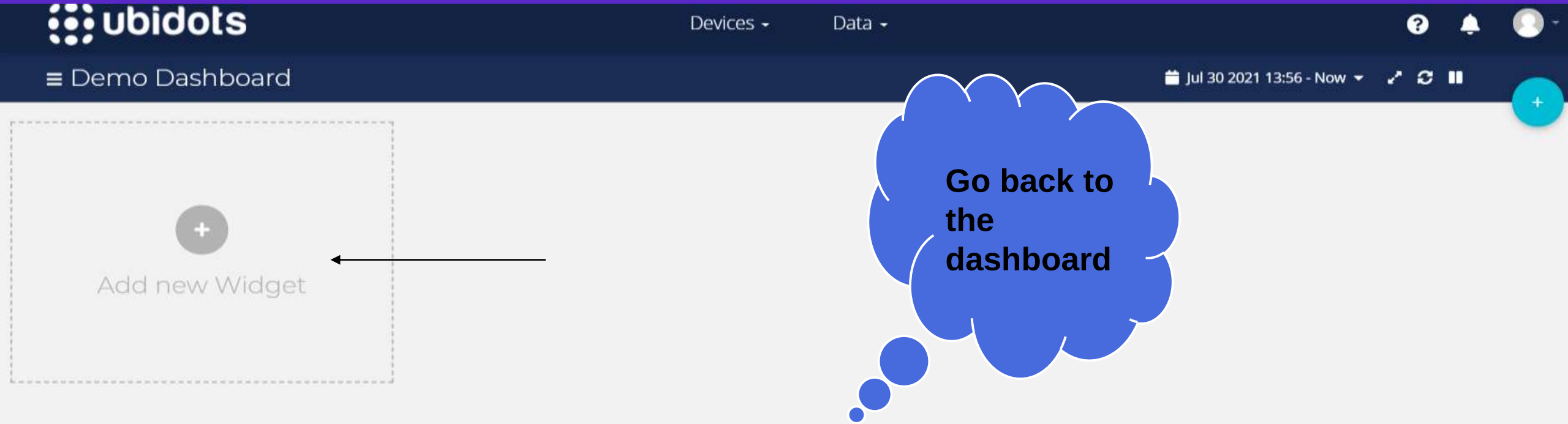
Ubidots.com


ubidots
Devices
Data
Devices
SET LOCATIO
Copy Token
Add your code variables
API Label
test
ID
61053a0803372f000d1fa001
Token
Tags
Add Variable

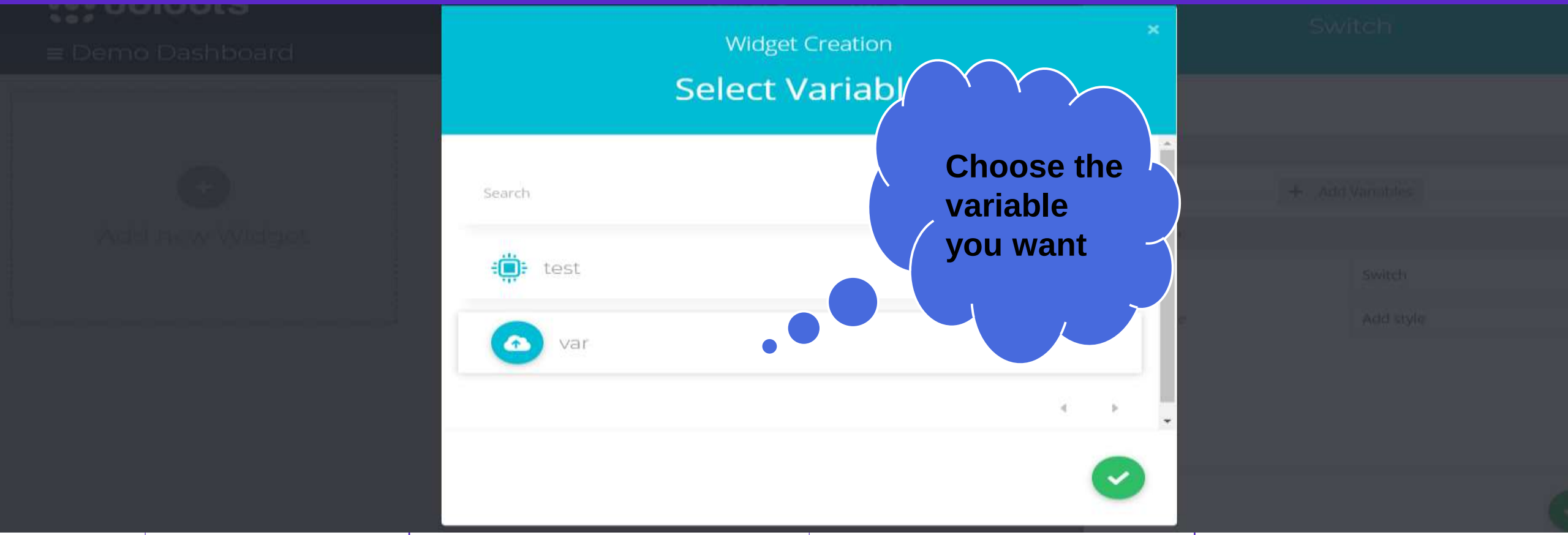
Ubidots.com


ubidots
Devices
Data
Devices
SET LOCATION
test
Description
Change description
API Label
test
ID
61053a0803372f000d1fa001
Token
Tags
var
Last activity:
No last activity
Add Variable

Ubidots.com


ubidots
Devices
Data
Demo Dashboard
Jul 30 2021 13:56 - Now
Add new Widget
Go back to the dashboard

Ubidots.com


Widget Creation
Select Variabl
Search
test
var
Choose the variable you want
Demo Dashboard
Switch
Add new Widget
Add Variables
Switch
Add style

Practice 2

Components:

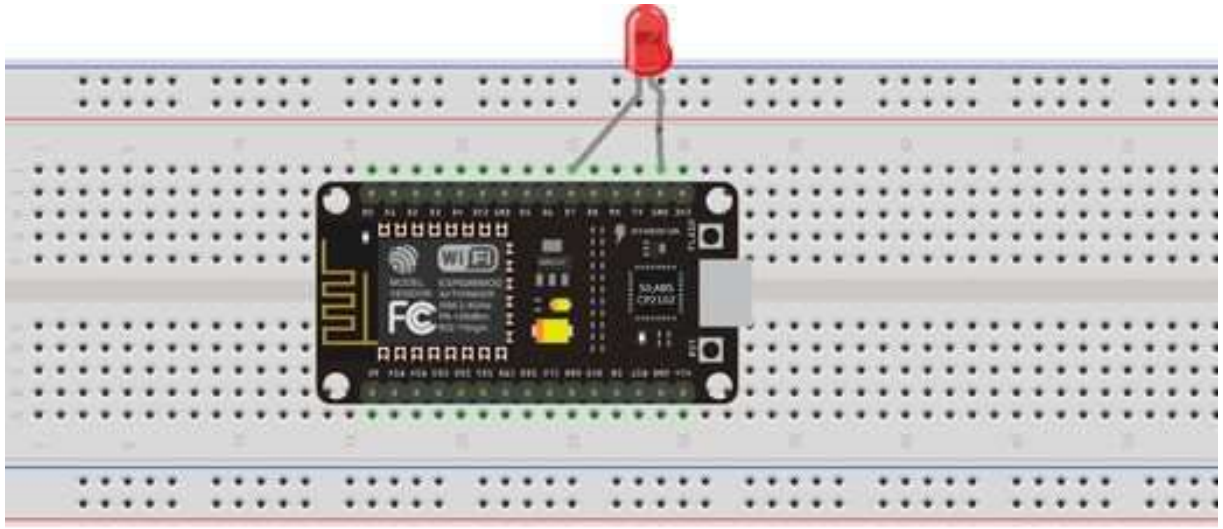
- **nodeMCU**
- **LED**

description:

- **Control the led from anywhere using ubidots**

Practice 2 solution

```
#include "UbidotsESPMQTT.h"
#define TOKEN "your token" // Your Ubidots TOKEN
#define WIFINAME "Baraka" //Your SSID
#define WIFIPASS "00000000" // Your Wifi Pass
#define DEVICE_LABEL "MCU"  // Put here your Ubidots device label
#define VARIABLE1 "test"  // Put here your Ubidots variable label
int ledPin = D0;
String value="";
Ubidots client(TOKEN);

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();
   if (payload[0]==1){
    digitalWrite(ledPin, HIGH);
    Serial.println("1");
  }
   else if (payload[0]==0){
    digitalWrite(ledPin, LOW);
    Serial.println("0");
  }
}
```



Practice 2 solution

```
void setup() {
  // put your setup code here, to run once:

  client.setDebug(true); // Pass a true or false bool value to activate debug messages
  Serial.begin(115200);
  client.wifiConnection(WIFINAME, WIFIPASS);
  client.begin(callback);
  client.ubidotsSubscribe(DEVICE_LABEL, VARIABLE1); //Insert the dataSource and Variable's Labels
  pinMode(ledPin, OUTPUT);

 }

void loop() {
  // put your main code here, to run repeatedly:
  if (!client.connected()) {
    client.reconnect();
  client.ubidotsSubscribe(DEVICE_LABEL, VARIABLE1); //Insert the dataSource and Variable's Labels

  }
 client.loop();
}
```

HTTP or MQTT ?

HTTP VS MQTT

VS

http://

	MQTT	HTTP
Full form :	**Message Queue Telemetry Transport**	**Hyper Text Transfer Protocol**
Architecture :	**publish/subscribe architecture**	**request/response (Client/Server) architecture**
message size :	**small**	**large**

HTTP VS MQTT

VS

http://

Message format :	**binary**	**ASCII format**
distribution	**1 to 0/1/N**	**one to one only**
Complexity :	**Simple**	**Client more complex**

The End

THANK YOU