# IoT Platforms using ESP8266 and ESP32

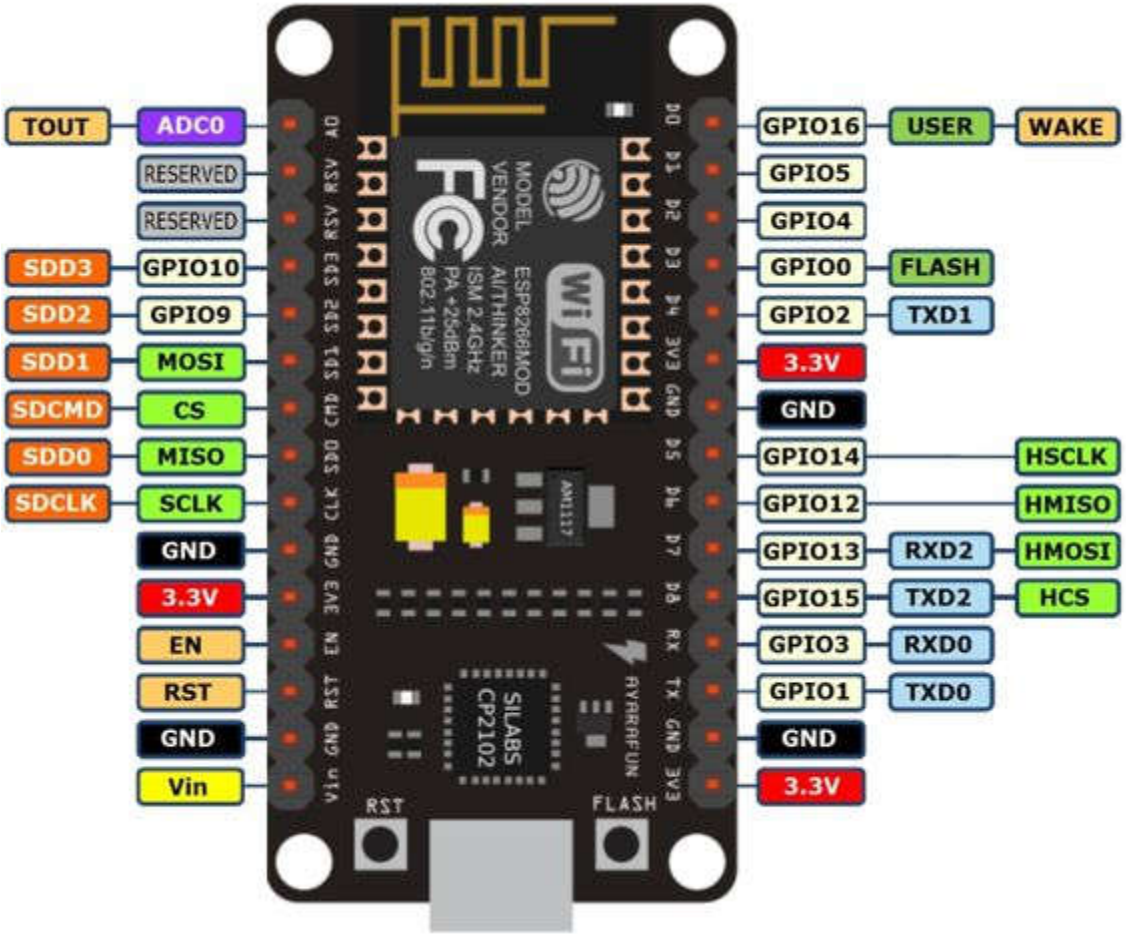
# IoT Platforms using ESP8266 and ESP32

**Contents**

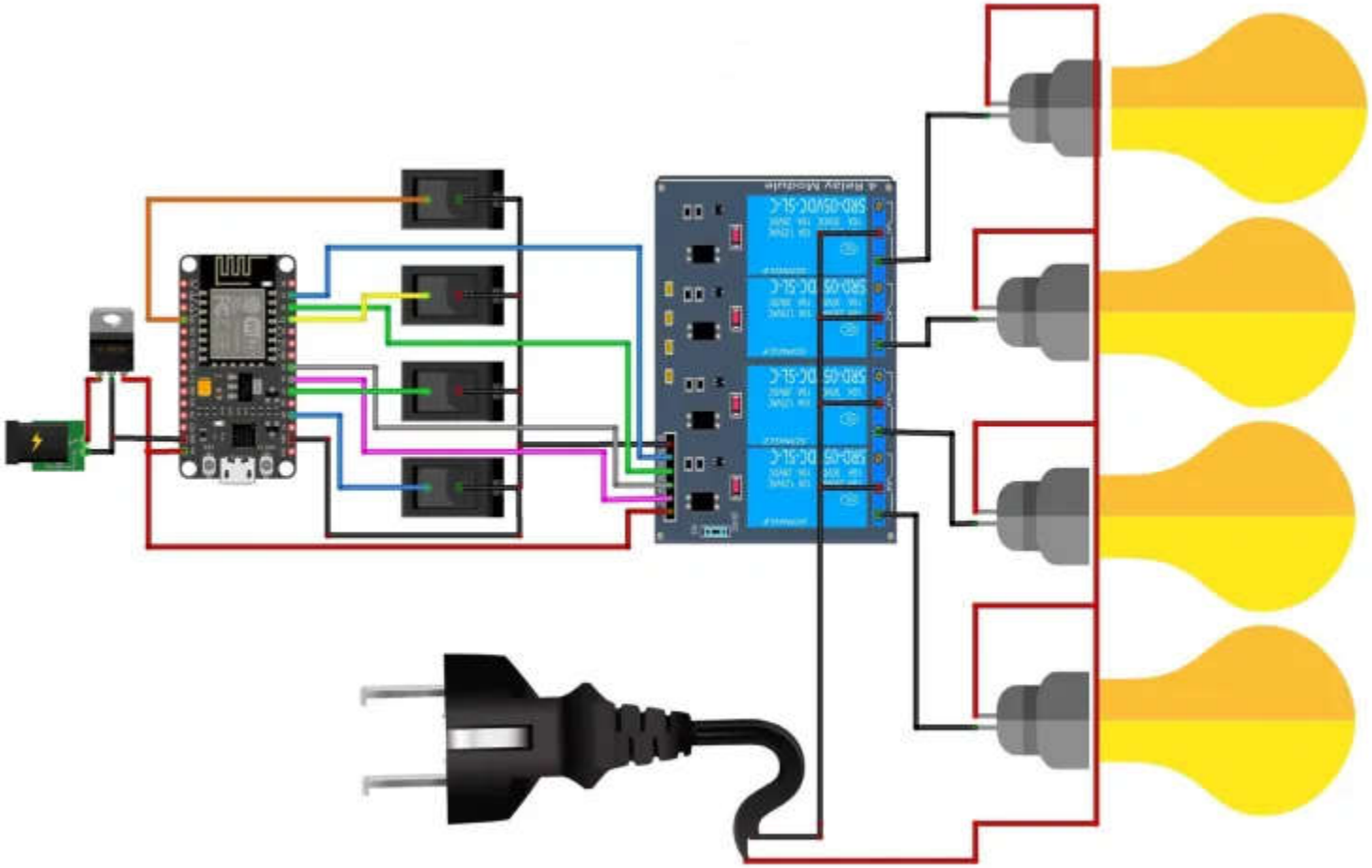
# Objectives

- In this tutorial, you will learn:
  - To familiarize Arduino IoT Cloud using ESP8266 and ESP32.
  - To understand basic Thingsboard dashboard using ESP8266 and ESP32.

# IoT using ESP8266 WiFi Module



# IoT using ESP8266 WiFi Module

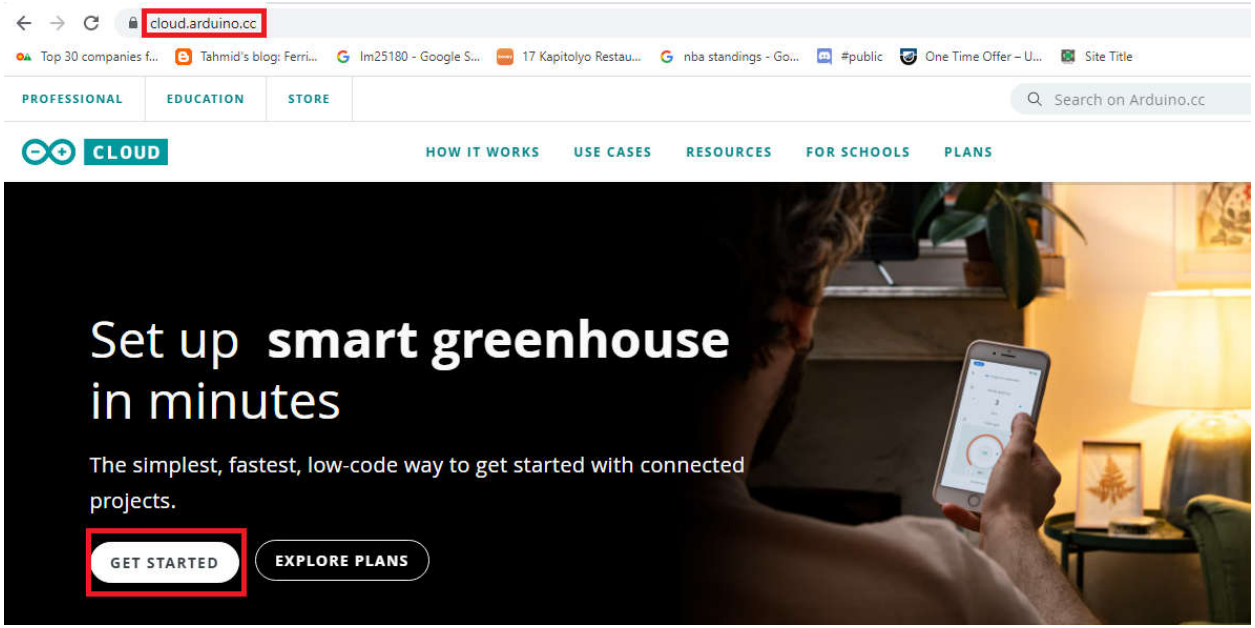


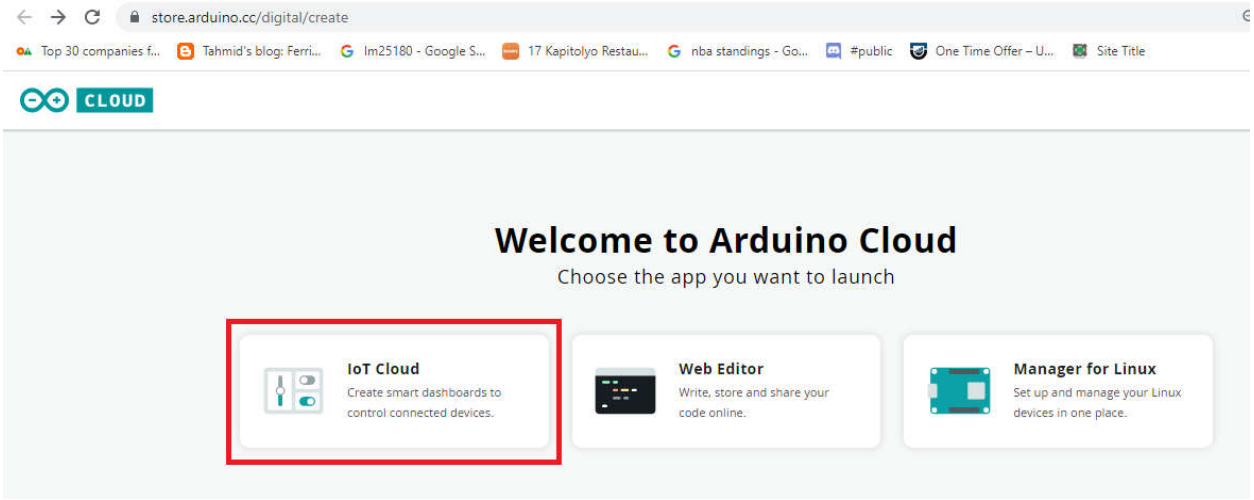
# IoT using ESP8266 WiFi Module

Go to https://cloud.arduino.cc/

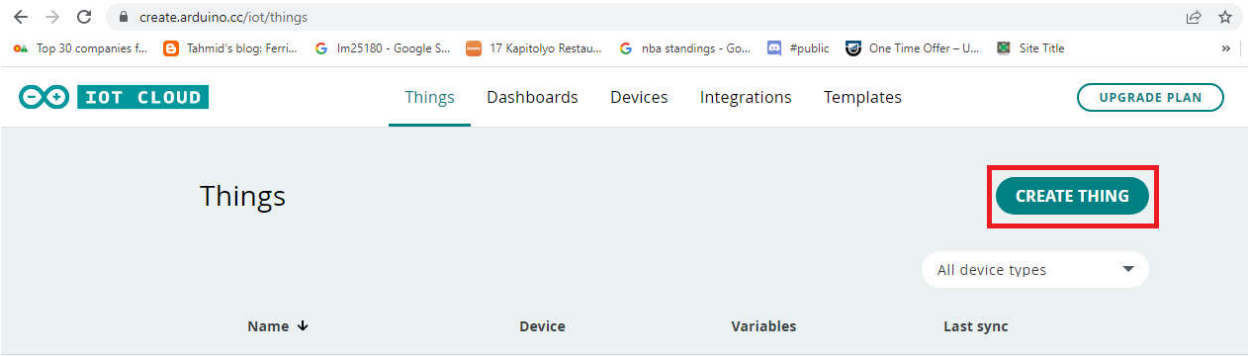
1.Click get started.
2.Click IoT Cloud.
3.Create Thing.
4.Add variable.
5.Name Variable Switch1.
6.Variable type Cloud Switch.
7.Click Add variable to create first variable.
8.Select Device then ESP8266 then NodeMCU1.0 ESP-12E Module.
9.Click continue and name it IECEP_SmartHome.
10.Get device ID and secret key
11.Setup Network credentials: SSID, Password, and Secret Key
12.Goto dashboard
13.got to edit then add variable and name it switch1 then link a variable
14.Click Done and add 3 more variables and link variable to them
15.Goto Things->Sketch
16.Install Arduino create agent
17.Edit code then compile and upload
18.Test using Monitor
19.Go to Dashboard to test the project
20. Download the Arduino IoT Cloud Remote in Playstore.

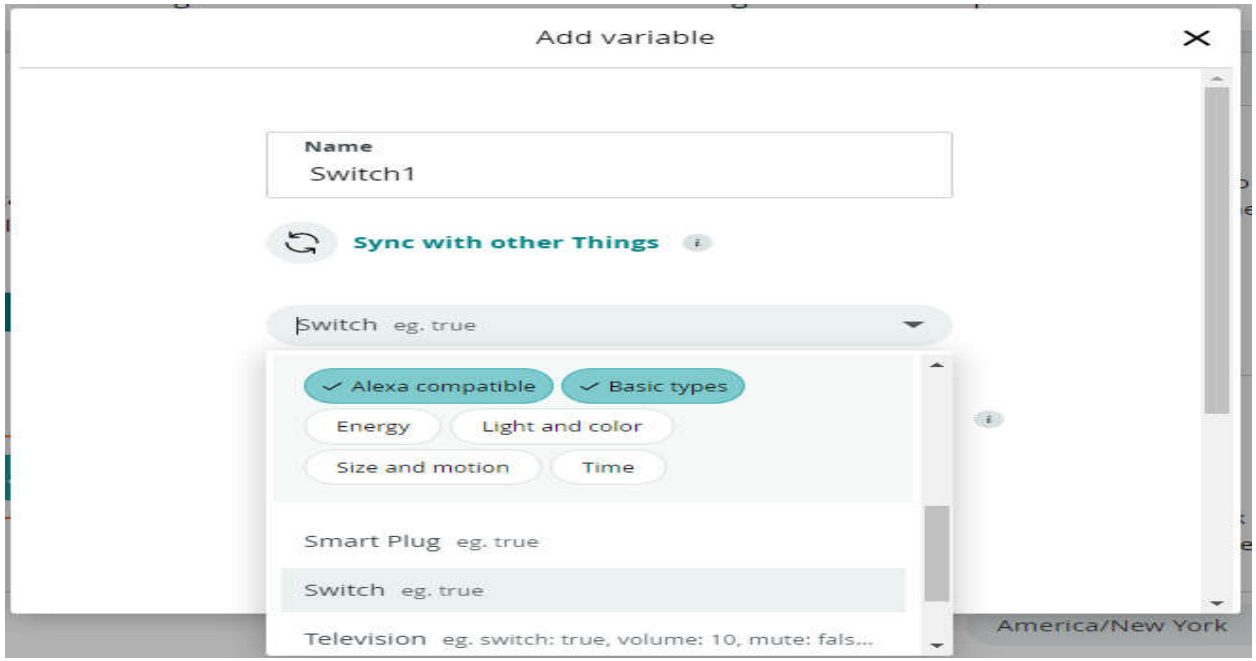
# IoT using ESP8266 WiFi Module



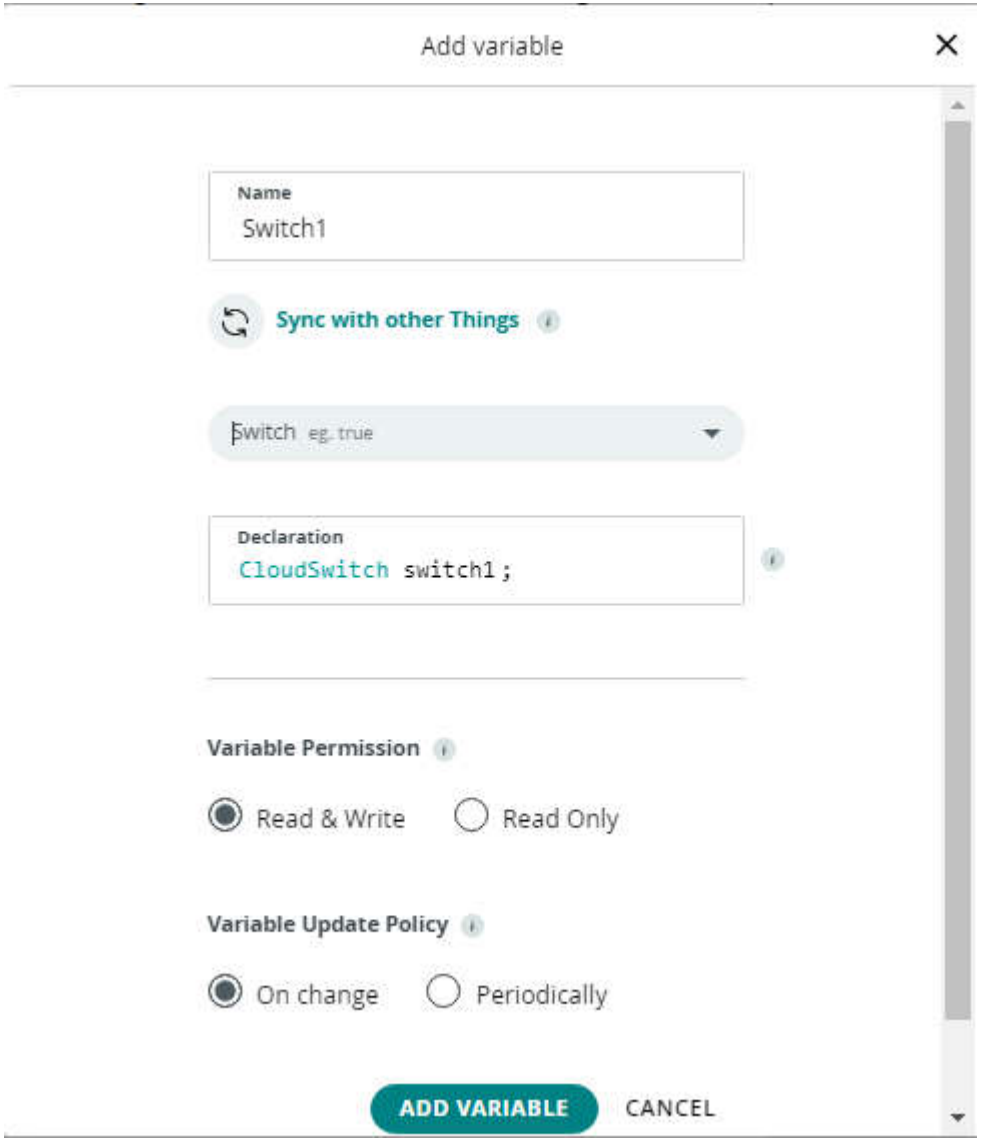



# IoT using ESP8266 WiFi Module

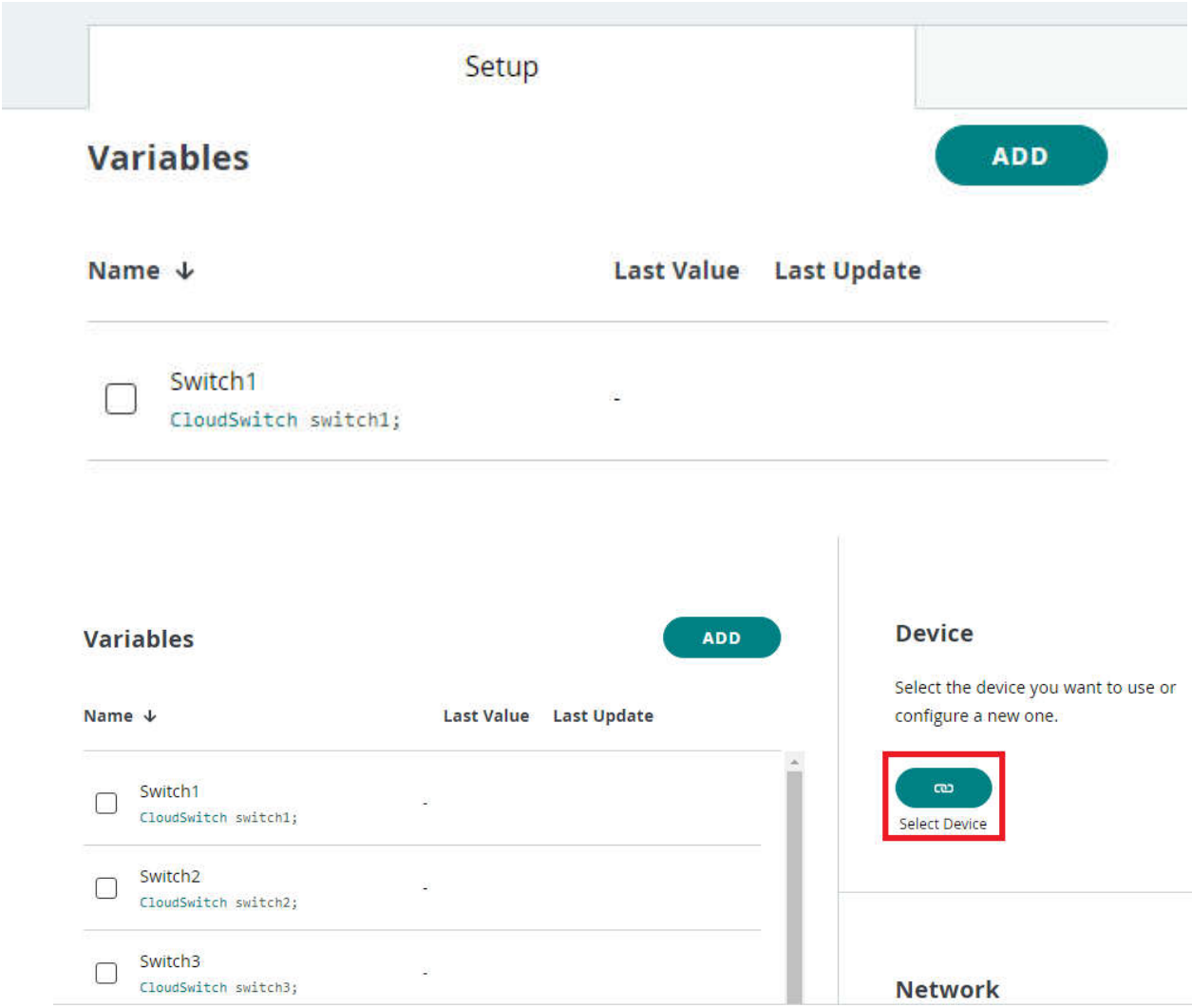


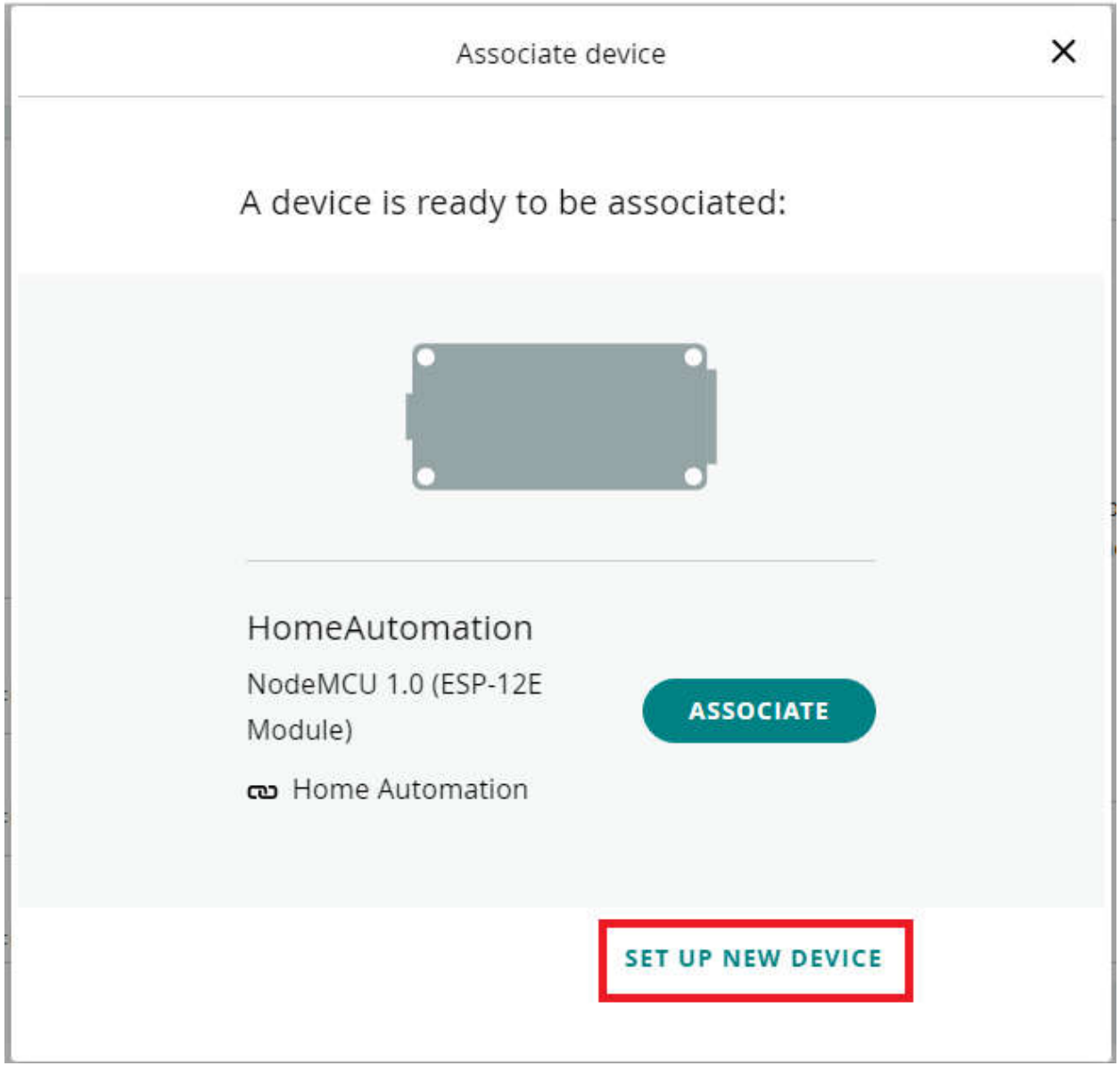
# IoT using ESP8266 WiFi Module



# IoT using ESP8266 WiFi Module

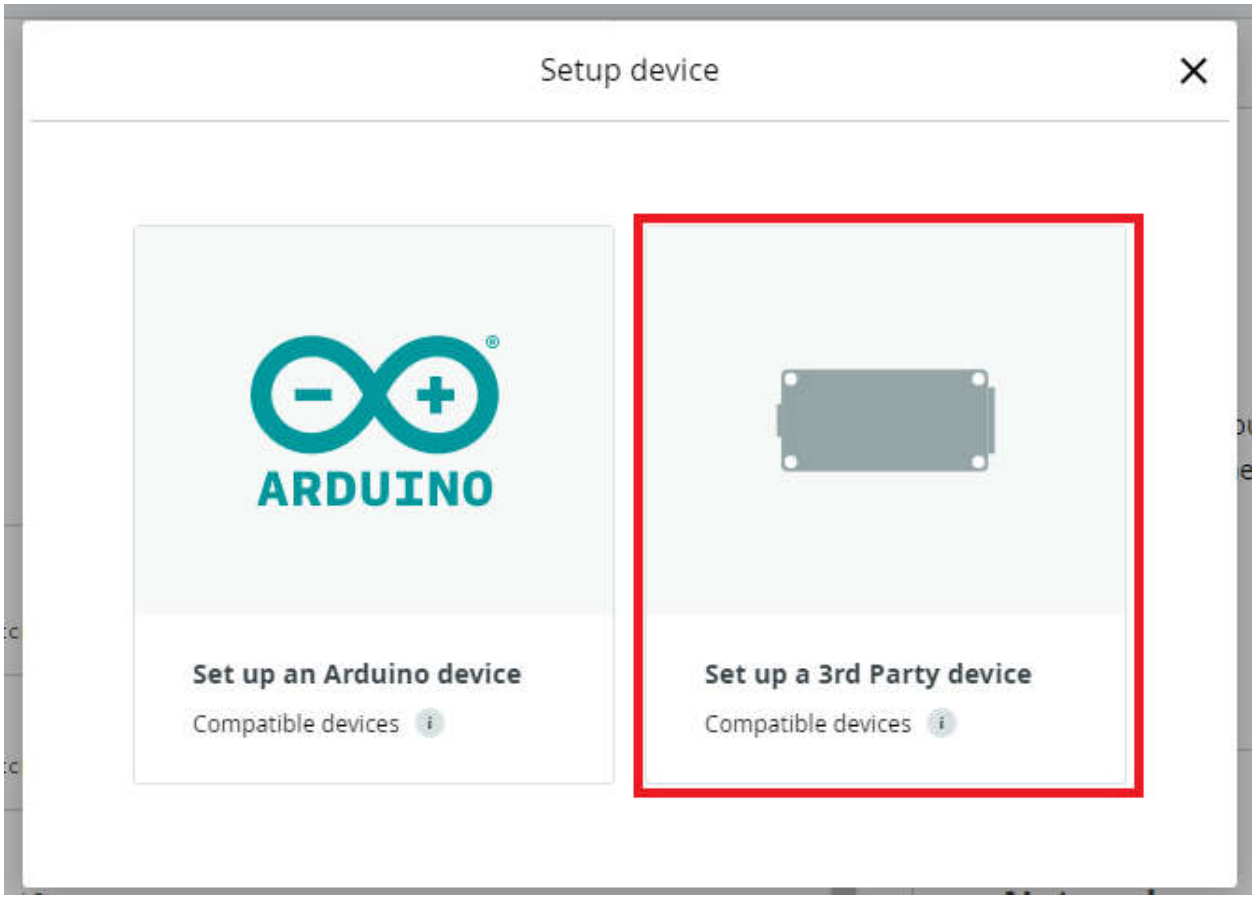


# IoT using ESP8266 WiFi Module

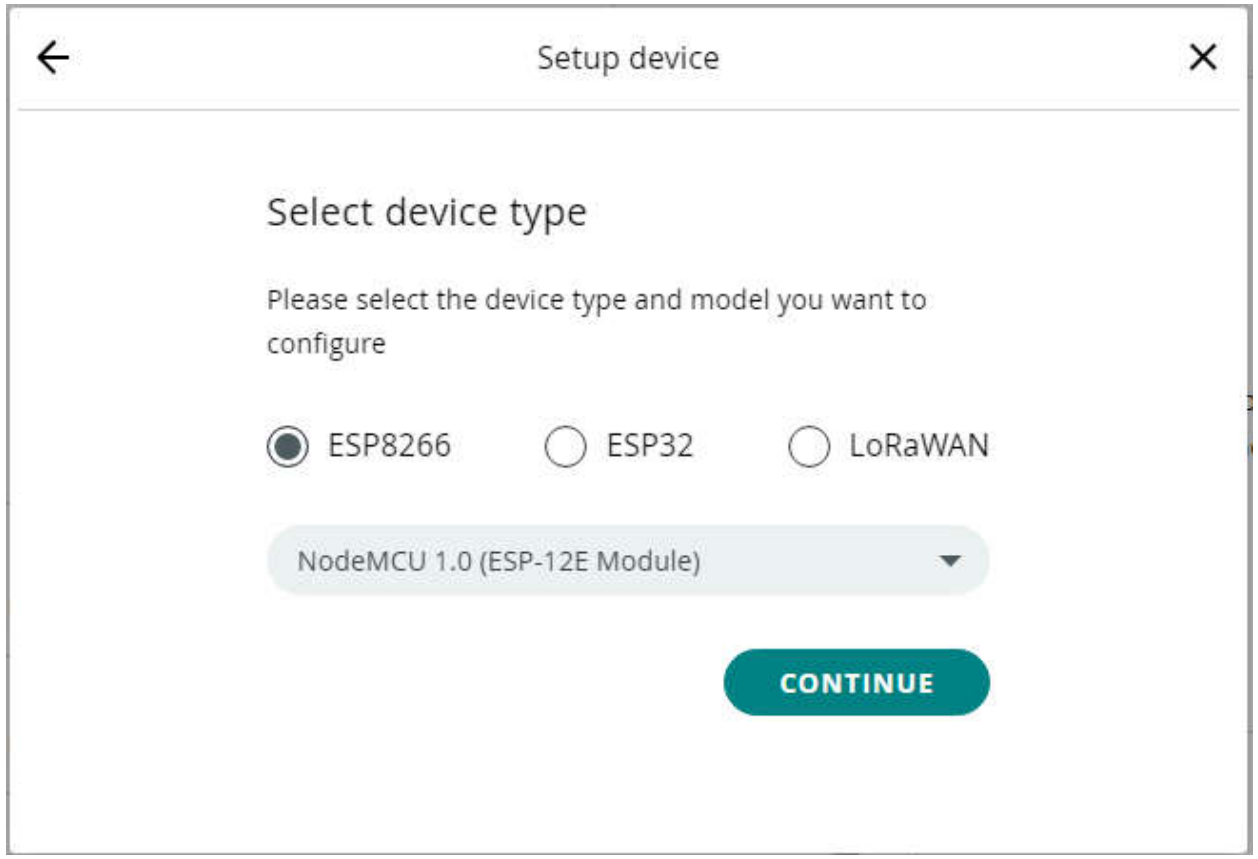


# IoT using ESP8266 WiFi Module

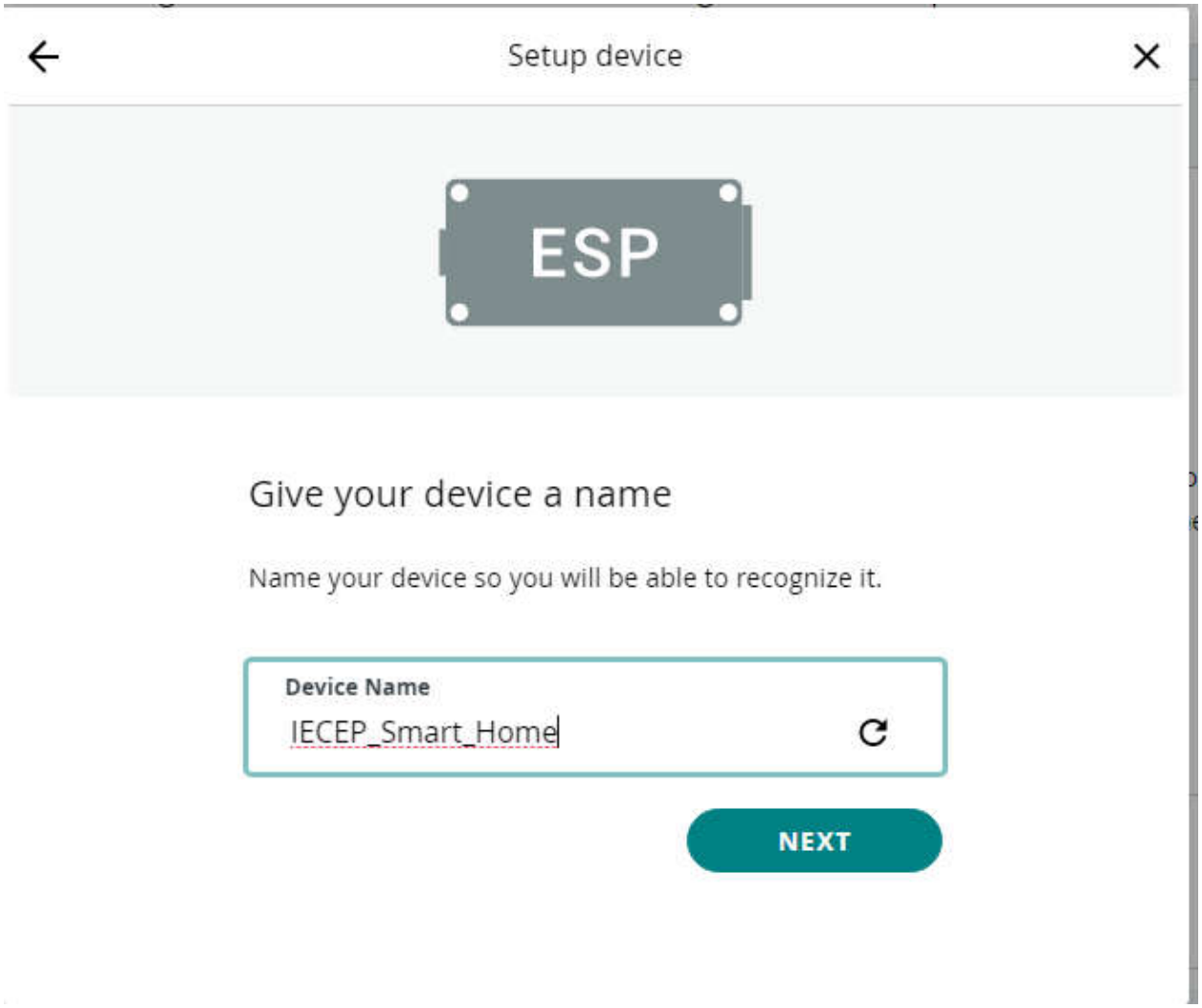
# IoT using ESP8266 WiFi Module



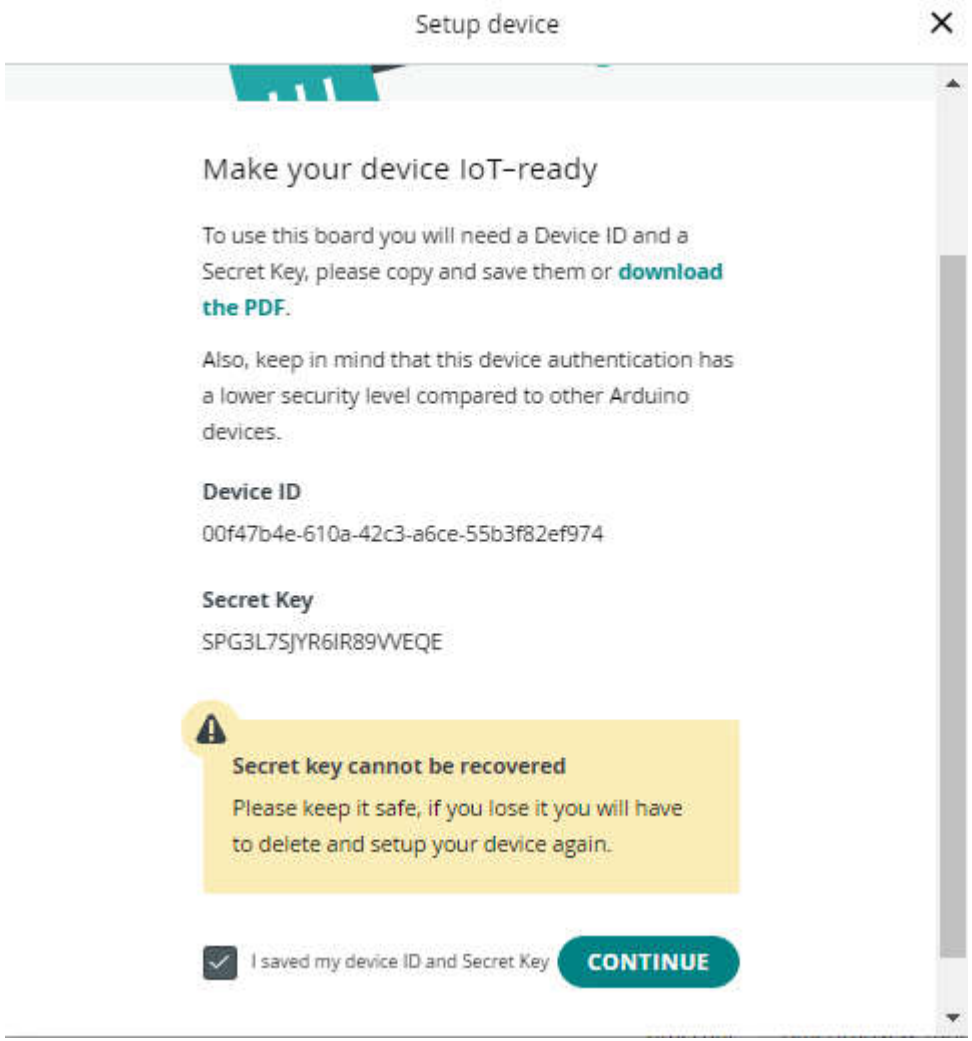
# IoT using ESP8266 WiFi Module



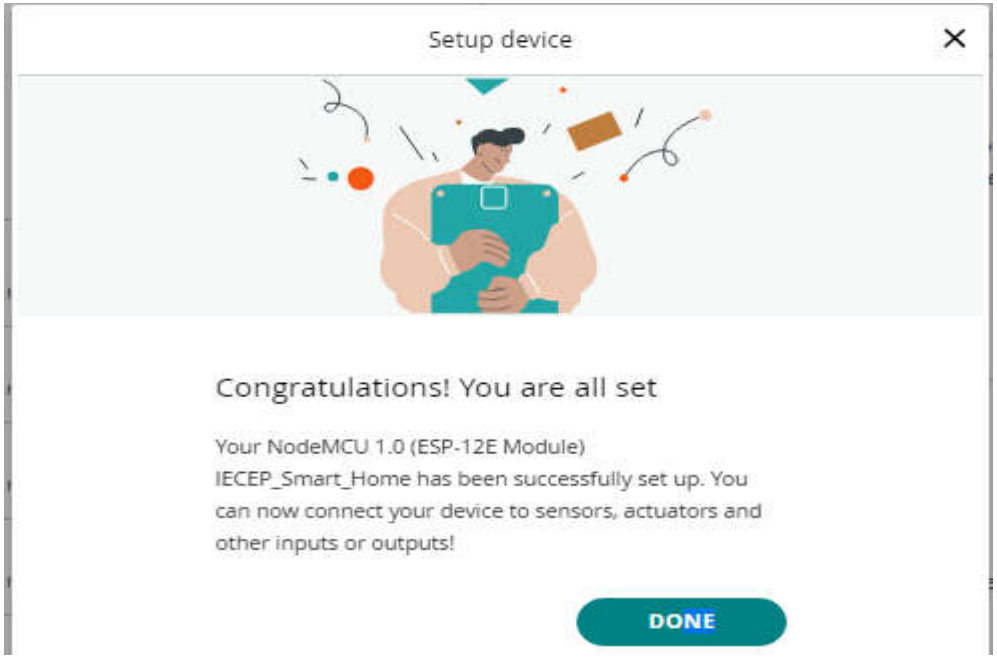
# IoT using ESP8266 WiFi Module

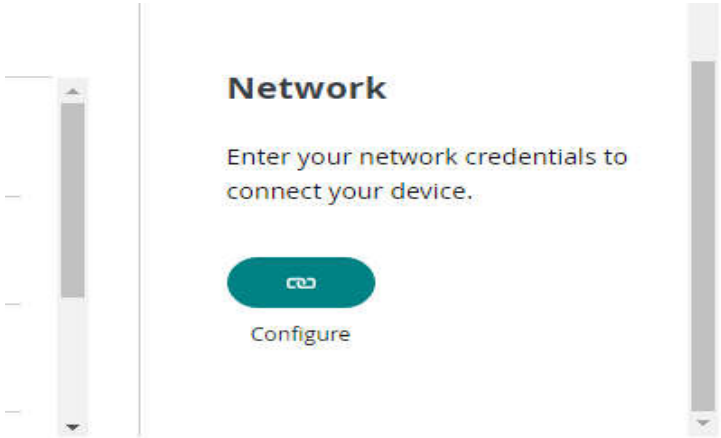


# IoT using ESP8266 WiFi Module

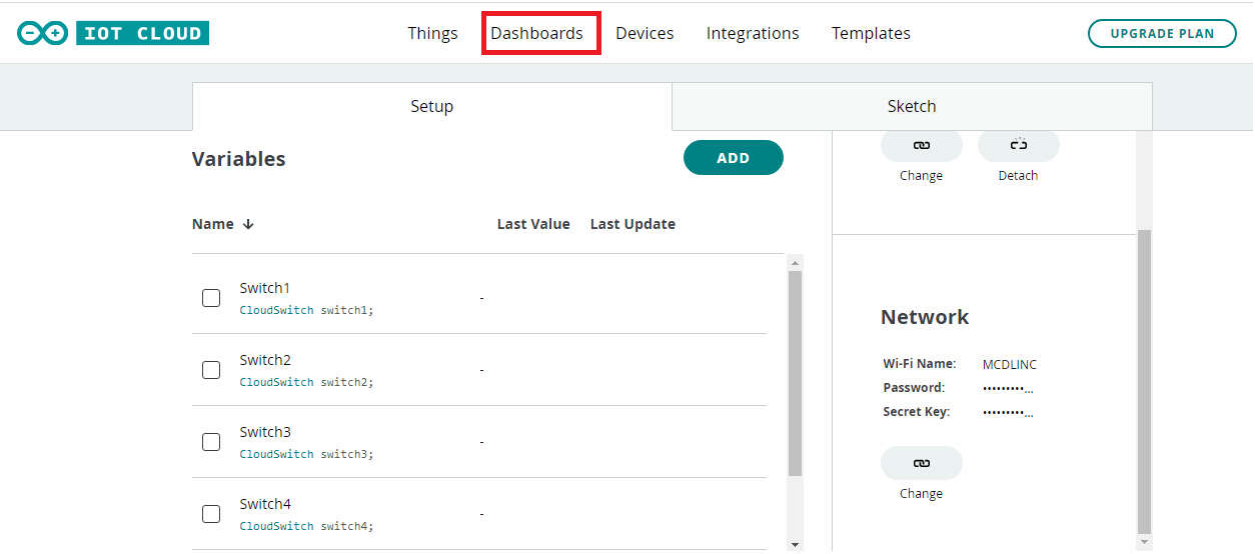


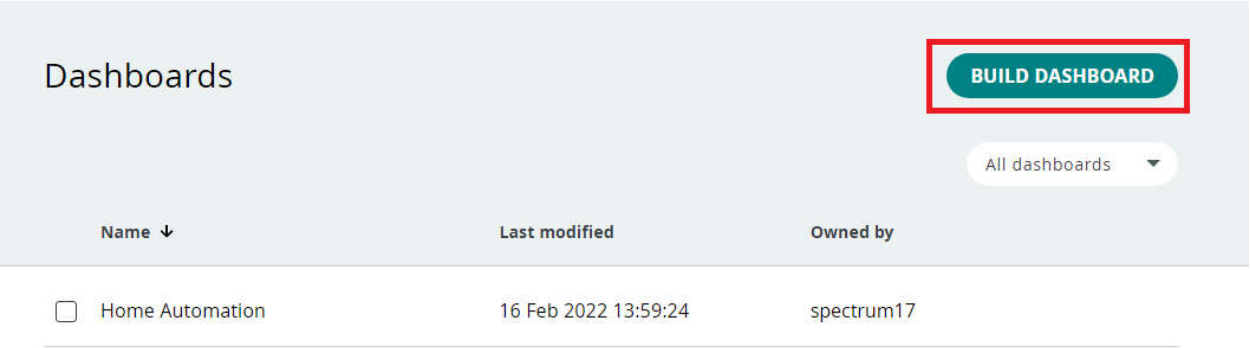
# IoT using ESP8266 WiFi Module



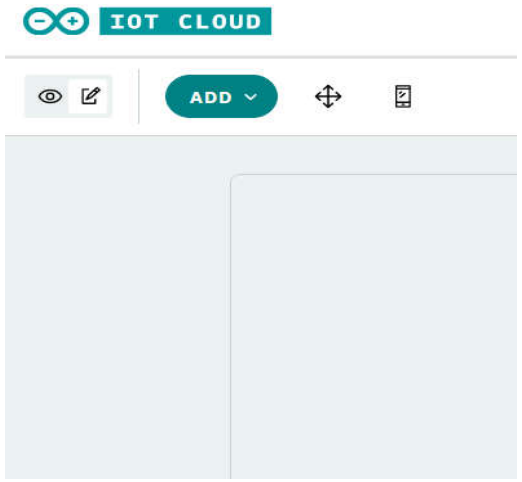



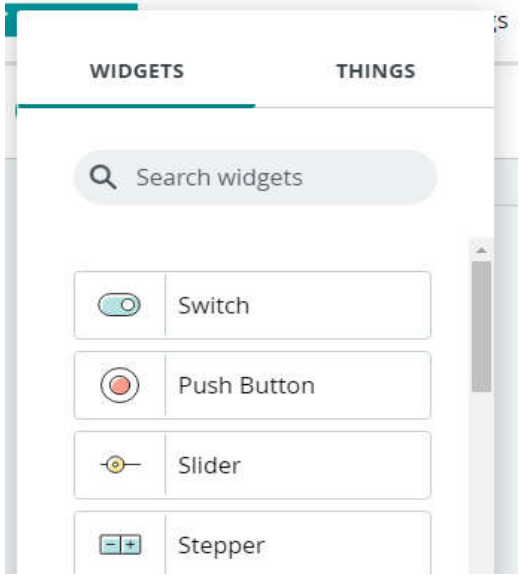
# IoT using ESP8266 WiFi Module



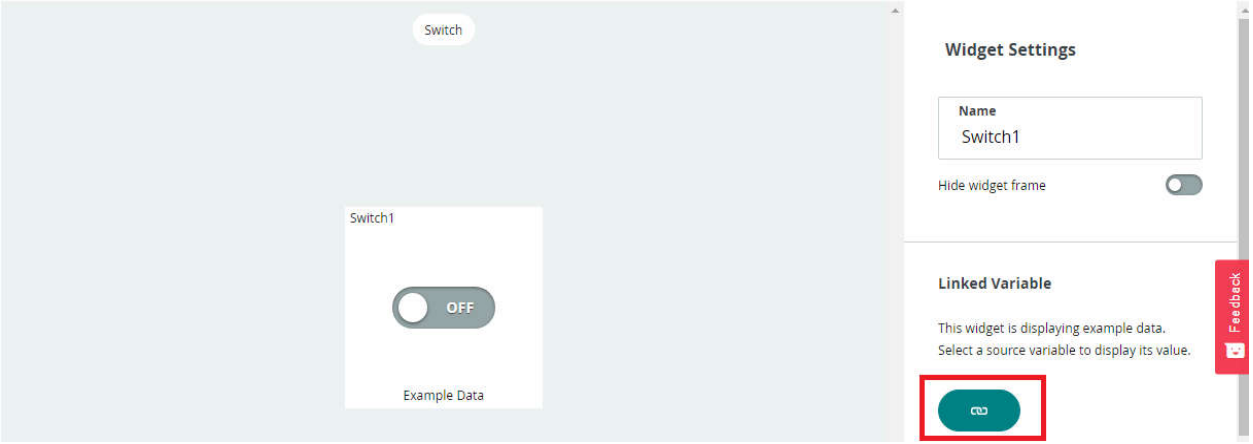



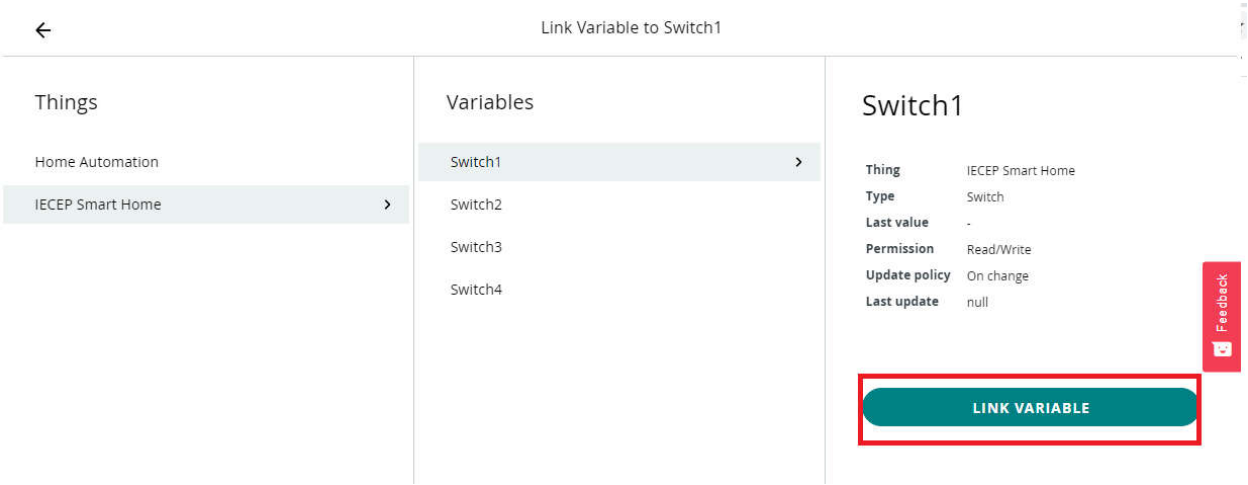
# IoT using ESP8266 WiFi Module



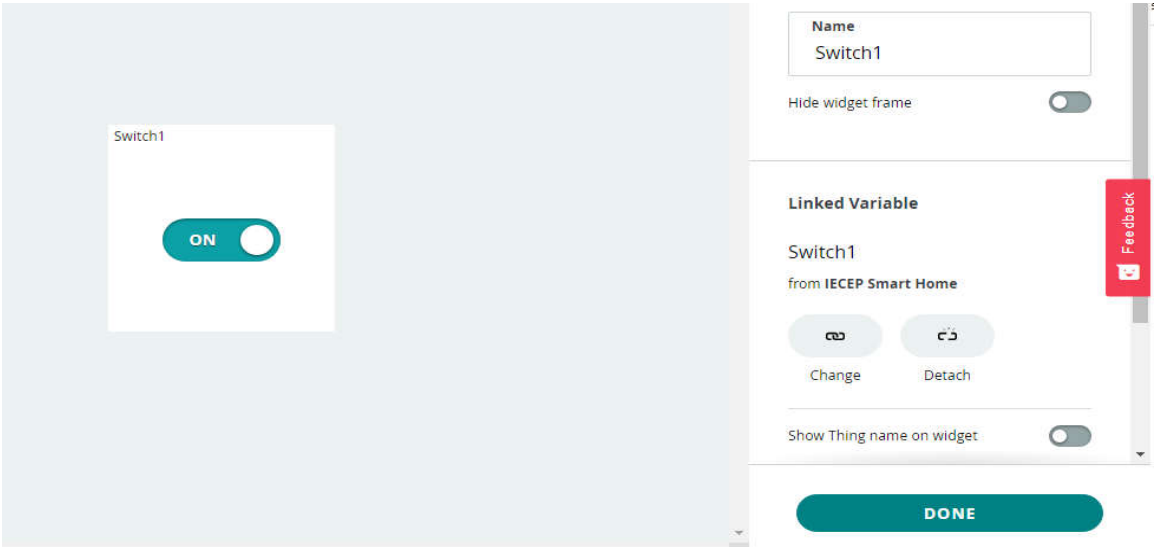



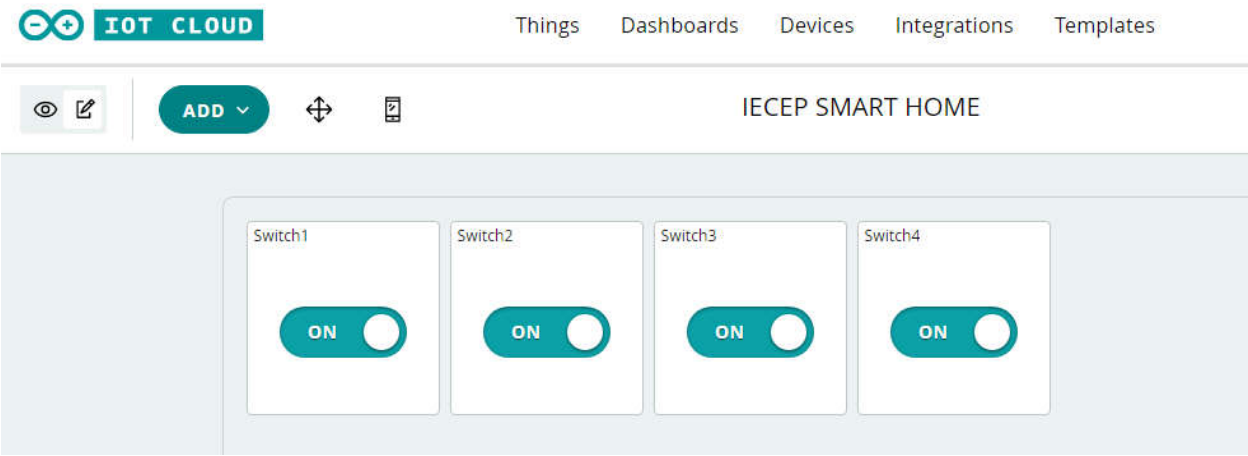
# IoT using ESP8266 WiFi Module



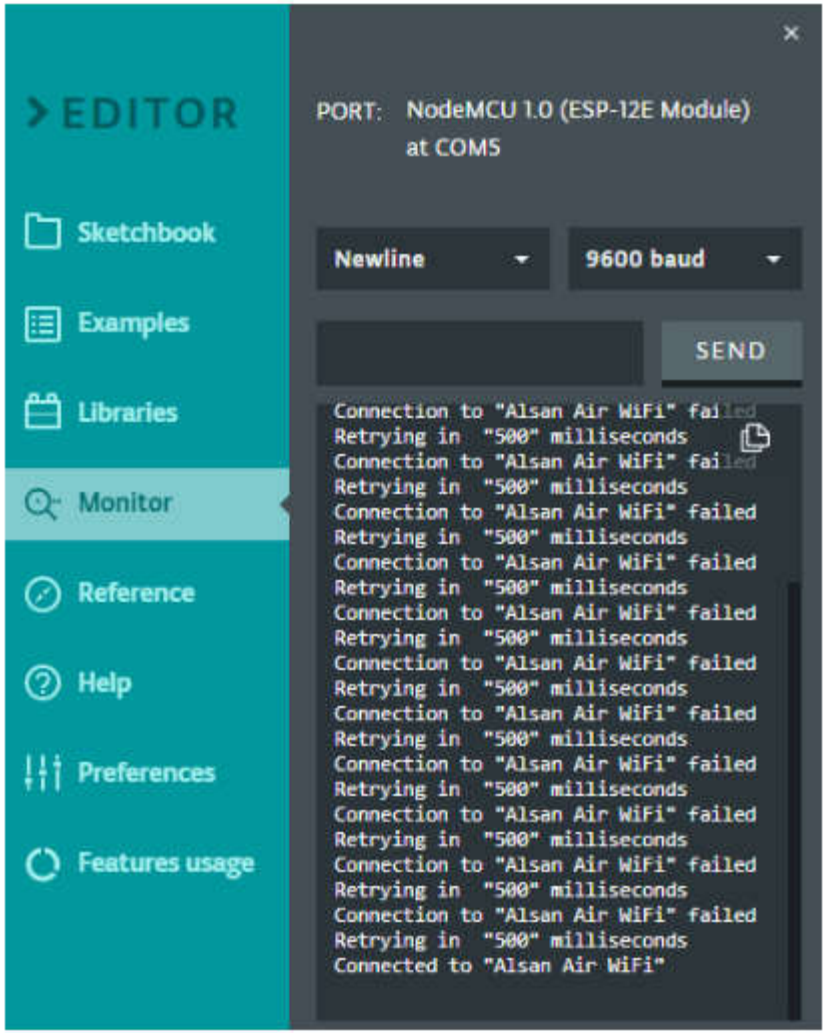
# IoT using ESP8266 WiFi Module



# IoT using ESP8266 WiFi Module

# IoT using ESP8266 WiFi Module



# IoT using ESP8266 WiFi Module

```
/*
  Sketch generated by the Arduino IoT Cloud Thing "IECEP_Smart_Home.ino"
  https://create.arduino.cc/cloud/things/3456fd1a-f508-4eef-bdd9-4a766d9ed64f

  Arduino IoT Cloud Variables description

  The following variables are automatically generated and updated when changes are made to the
Thing

  CloudSwitch switch1;
  CloudSwitch switch2;
  CloudSwitch switch3;
  CloudSwitch switch4;

  Variables which are marked as READ/WRITE in the Cloud Thing will also have functions
  which are called when their values are changed from the Dashboard.
  These functions are generated with the Thing and added at the end of this sketch.
*/

#include "thingProperties.h"
#include <AceButton.h>
using namespace ace_button;

// define the GPIO connected with Relays and switches
#define RelayPin1 14    //D5
#define RelayPin2 2     //D4
#define RelayPin3 15    //D8
#define RelayPin4 12    //D6

#define SwitchPin1 10  //SD3
#define SwitchPin2 0   //D3
#define SwitchPin3 13  //D7
#define SwitchPin4 3   //RX

#define wifiLed   16   //D0
```

# IoT using ESP8266 WiFi Module

```
int toggleState_1 = 0; //Define integer to remember the toggle state for relay 1
int toggleState_2 = 0; //Define integer to remember the toggle state for relay 2
int toggleState_3 = 0; //Define integer to remember the toggle state for relay 3
int toggleState_4 = 0; //Define integer to remember the toggle state for relay 4

ButtonConfig config1;
AceButton button1(&config1);
ButtonConfig config2;
AceButton button2(&config2);
ButtonConfig config3;
AceButton button3(&config3);
ButtonConfig config4;
AceButton button4(&config4);

void handleEvent1(AceButton*, uint8_t, uint8_t);
void handleEvent2(AceButton*, uint8_t, uint8_t);
void handleEvent3(AceButton*, uint8_t, uint8_t);
void handleEvent4(AceButton*, uint8_t, uint8_t);

void relayOnOff(int relay) {

  switch (relay) {
    case 1:
      if (toggleState_1 == 0) {
        digitalWrite(RelayPin1, LOW); // turn on relay 1
        toggleState_1 = 1;
        Serial.println("Device1 ON");
      }
      else {
        digitalWrite(RelayPin1, HIGH); // turn off relay 1
        toggleState_1 = 0;
        Serial.println("Device1 OFF");
      }
      delay(100);
```

# IoT using ESP8266 WiFi Module

```
break;
   case 2:
     if (toggleState_2 == 0) {
       digitalWrite(RelayPin2, LOW); // turn on relay 2
       toggleState_2 = 1;
       Serial.println("Device2 ON");
     }
     else {
       digitalWrite(RelayPin2, HIGH); // turn off relay 2
       toggleState_2 = 0;
       Serial.println("Device2 OFF");
     }
     delay(100);
     break;
   case 3:
     if (toggleState_3 == 0) {
       digitalWrite(RelayPin3, LOW); // turn on relay 3
       toggleState_3 = 1;
       Serial.println("Device3 ON");
     } else {
       digitalWrite(RelayPin3, HIGH); // turn off relay 3
       toggleState_3 = 0;
       Serial.println("Device3 OFF");
     }
     delay(100);
     break;
   case 4:
     if (toggleState_4 == 0) {
       digitalWrite(RelayPin4, LOW); // turn on relay 4
       toggleState_4 = 1;
       Serial.println("Device4 ON");
     }
     else {
       digitalWrite(RelayPin4, HIGH); // turn off relay 4
       toggleState_4 = 0;
```

# IoT using ESP8266 WiFi Module

```
Serial.println("Device4 OFF");
      }
      delay(100);
      break;
    default : break;
  }
}

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  pinMode(RelayPin1, OUTPUT);
  pinMode(RelayPin2, OUTPUT);
  pinMode(RelayPin3, OUTPUT);
  pinMode(RelayPin4, OUTPUT);

  pinMode(wifiLed, OUTPUT);

  pinMode(SwitchPin1, INPUT_PULLUP);
  pinMode(SwitchPin2, INPUT_PULLUP);
  pinMode(SwitchPin3, INPUT_PULLUP);
  pinMode(SwitchPin4, INPUT_PULLUP);
```

# IoT using ESP8266 WiFi Module



```
//During Starting all Relays should TURN OFF
  digitalWrite(RelayPin1, HIGH);
  digitalWrite(RelayPin2, HIGH);
  digitalWrite(RelayPin3, HIGH);
  digitalWrite(RelayPin4, HIGH);

  digitalWrite(wifiLed, HIGH);  //Turn OFF WiFi LED

  config1.setEventHandler(button1Handler);
  config2.setEventHandler(button2Handler);
  config3.setEventHandler(button3Handler);
  config4.setEventHandler(button4Handler);


}

void loop() {
  ArduinoCloud.update();

      //Manual Switch Control
    button1.check();
    button2.check();
    button3.check();
    button4.check();

  if (WiFi.status() != WL_CONNECTED)
  {
    digitalWrite(wifiLed, HIGH); //Turn OFF WiFi LED
  }
  else{
    digitalWrite(wifiLed, LOW); //Turn ON WiFi LED
  }
}
```

# IoT using ESP8266 WiFi Module

```
void onSwitch1Change() {
  if (switch1 == 1)
  {
    digitalWrite(RelayPin1, LOW);
    Serial.println("Device1 ON");
    toggleState_1 = 1;
  }
  else
  {
    digitalWrite(RelayPin1, HIGH);
    Serial.println("Device1 OFF");
    toggleState_1 = 0;
  }
}

void onSwitch2Change() {
  if (switch2 == 1)
  {
    digitalWrite(RelayPin2, LOW);
    Serial.println("Device2 ON");
    toggleState_2 = 1;
  }
  else
  {
    digitalWrite(RelayPin2, HIGH);
    Serial.println("Device2 OFF");
    toggleState_2 = 0;
  }
}
```

# IoT using ESP8266 WiFi Module

```
void onSwitch3Change() {
  if (switch3 == 1)
  {
    digitalWrite(RelayPin3, LOW);
    Serial.println("Device2 ON");
    toggleState_3 = 1;
  }
  else
  {
    digitalWrite(RelayPin3, HIGH);
    Serial.println("Device3 OFF");
    toggleState_3 = 0;
  }
}

void onSwitch4Change() {
  if (switch4 == 1)
  {
    digitalWrite(RelayPin4, LOW);
    Serial.println("Device4 ON");
    toggleState_4 = 1;
  }
  else
  {
    digitalWrite(RelayPin4, HIGH);
    Serial.println("Device4 OFF");
    toggleState_4 = 0;
  }
}
```

# IoT using ESP8266 WiFi Module

```
void button1Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT1");
  relayOnOff(1);
}
void button2Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT2");
  relayOnOff(2);
}
void button3Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT3");
  relayOnOff(3);
}
void button4Handler(AceButton* button, uint8_t eventType, uint8_t buttonState) {
  Serial.println("EVENT4");
  relayOnOff(4);
}
```

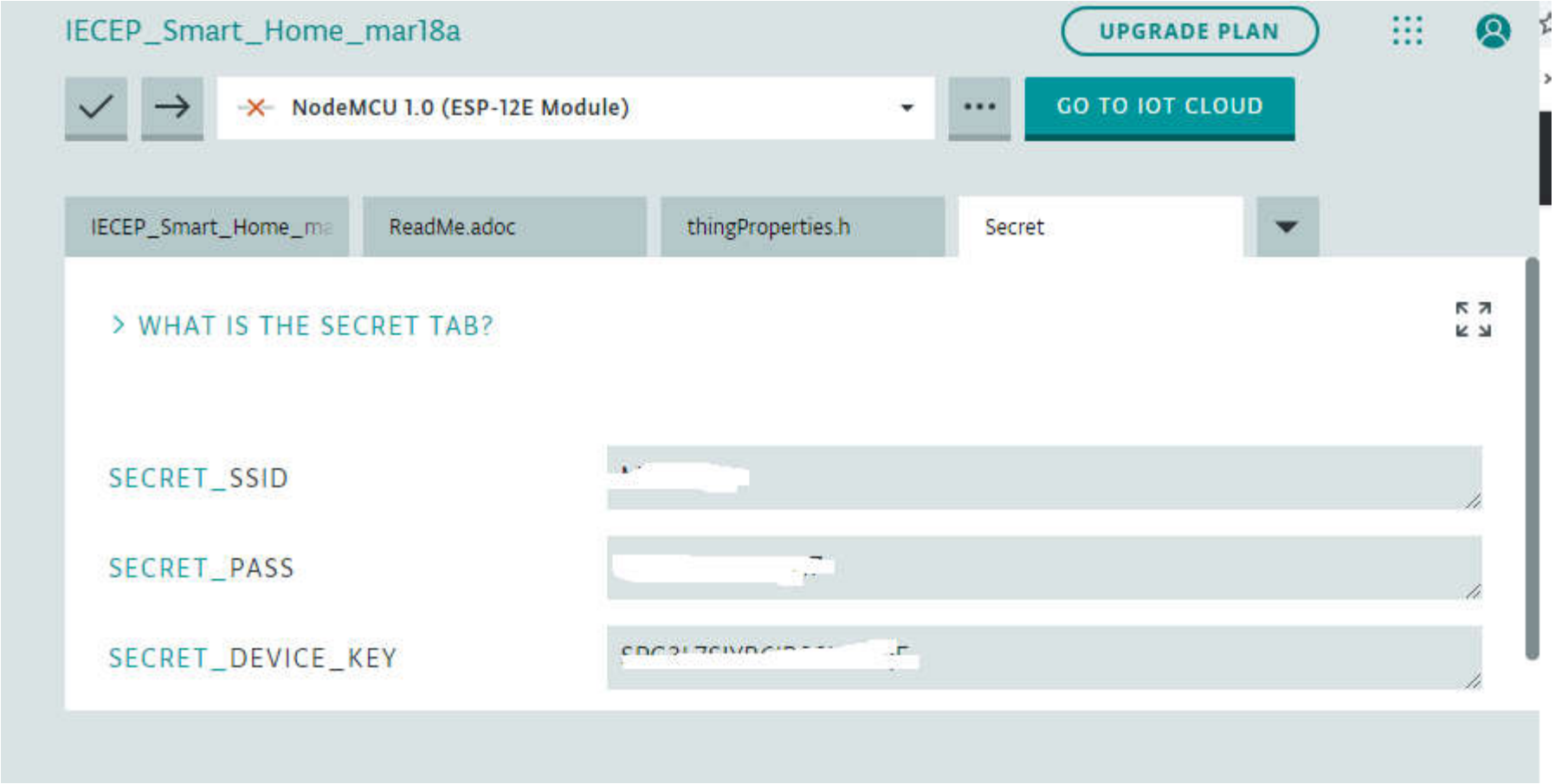
# IoT using ESP8266 WiFi Module

## thingsProperties.h

```
// Code generated by Arduino IoT Cloud, DO NOT EDIT.

#include <ArduinoIoTCloud.h>
#include <Arduino_ConnectionHandler.h>

const char DEVICE_LOGIN_NAME[]  = "00f47b4e-610a-42c3-a6ce-55b3f82ef974";

const char SSID[]               = SECRET_SSID;    // Network SSID (name)
const char PASS[]               = SECRET_PASS;    // Network password (use for WPA, or use as
key for WEP)
const char DEVICE_KEY[]  = SECRET_DEVICE_KEY;    // Secret device password

void onSwitch1Change();
void onSwitch2Change();
void onSwitch3Change();
void onSwitch4Change();

CloudSwitch switch1;
CloudSwitch switch2;
CloudSwitch switch3;
CloudSwitch switch4;

void initProperties(){

  ArduinoCloud.setBoardId(DEVICE_LOGIN_NAME);
  ArduinoCloud.setSecretDeviceKey(DEVICE_KEY);
  ArduinoCloud.addProperty(switch1, READWRITE, ON_CHANGE, onSwitch1Change);
  ArduinoCloud.addProperty(switch2, READWRITE, ON_CHANGE, onSwitch2Change);
  ArduinoCloud.addProperty(switch3, READWRITE, ON_CHANGE, onSwitch3Change);
  ArduinoCloud.addProperty(switch4, READWRITE, ON_CHANGE, onSwitch4Change);

}

WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```

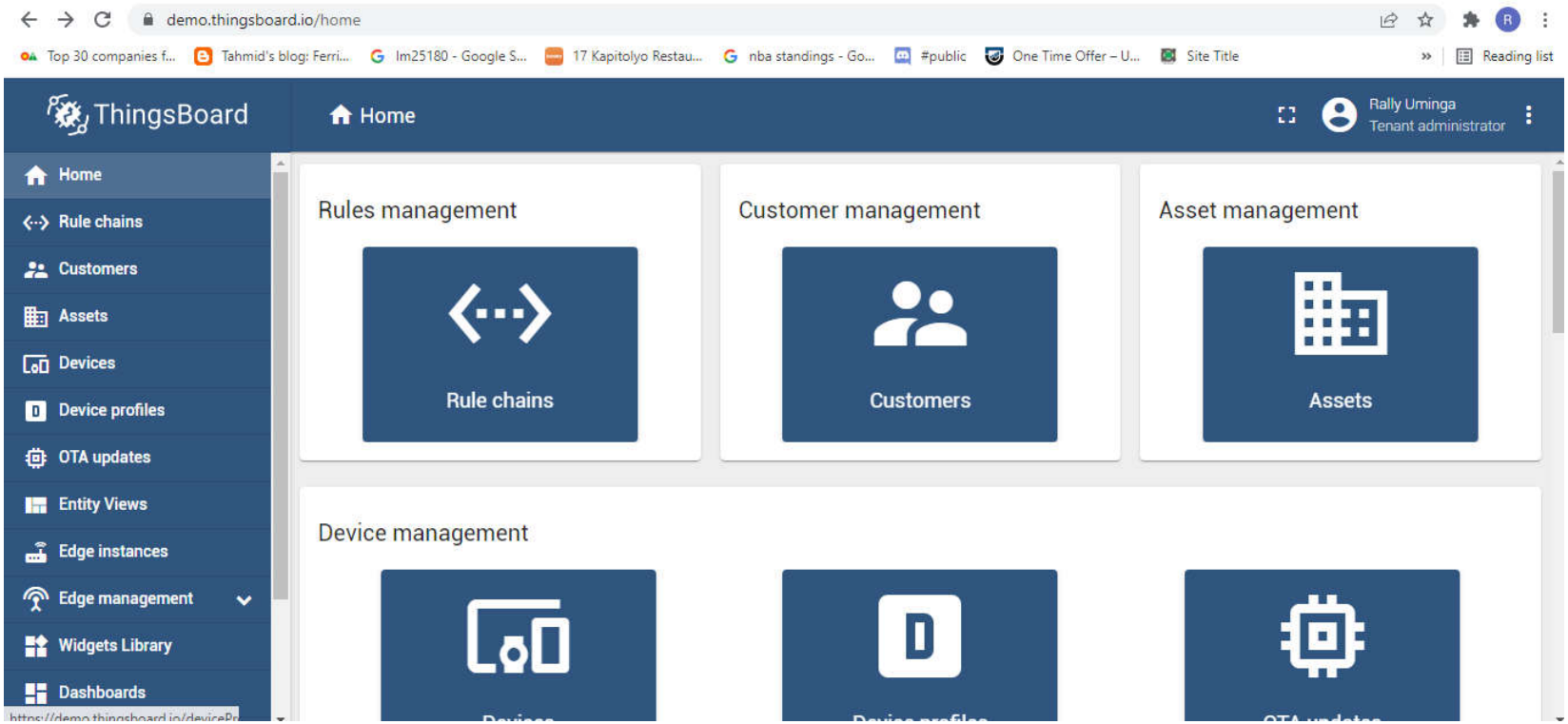
# IoT using ESP8266 WiFi Module

## Secret



# IoT using ESP8266 WiFi Module

## ThingsBoard



# IoT using ESP8266 WiFi Module

```
//DHT11_Thingsboard.ino
#include "ThingsBoard.h"
#include <ESP8266WiFi.h>    // for ESP8266
//#include <WiFi.h>         // for ESP32

#define WIFI_AP             “SSID"
#define WIFI_PASSWORD       “PASSWORD"

#include <DHT.h>

// See https://thingsboard.io/docs/getting-started-guides/helloworld/
// to understand how to obtain an access token
#define TOKEN               "JVpZ5m8xPKb6GJzMeKoE"
#define THINGSBOARD_SERVER  "demo.thingsboard.io"
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD   115200
#define DHT11_PIN 0
// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11     // DHT 11
//#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

DHT dht(DHT11_PIN, DHTTYPE);

// current temperature & humidity, updated in loop()
float t = 0.0;
float h = 0.0;

// Initialize ThingsBoard client
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;
```

# IoT using ESP8266 WiFi Module

```
unsigned long delayTime;

void setup(void)
{
// initialize serial for debugging
  Serial.begin(SERIAL_DEBUG_BAUD);
  dht.begin();

  bool status;

  Serial.println("-- Default Test --");
  delayTime = 1000;

  Serial.println();
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();

}

void loop()
{
  delay(10000);            // wait a second
  // Read temperature as Celsius (the default)
    float newT = dht.readTemperature();
    // Read temperature as Fahrenheit (isFahrenheit = true)
    //float newT = dht.readTemperature(true);
    // if temperature read failed, don't change t value
    if (isnan(newT)) {
      Serial.println("Failed to read from DHT sensor!");
    }
    else {
      t = newT;
      Serial.println(t);
    }
```

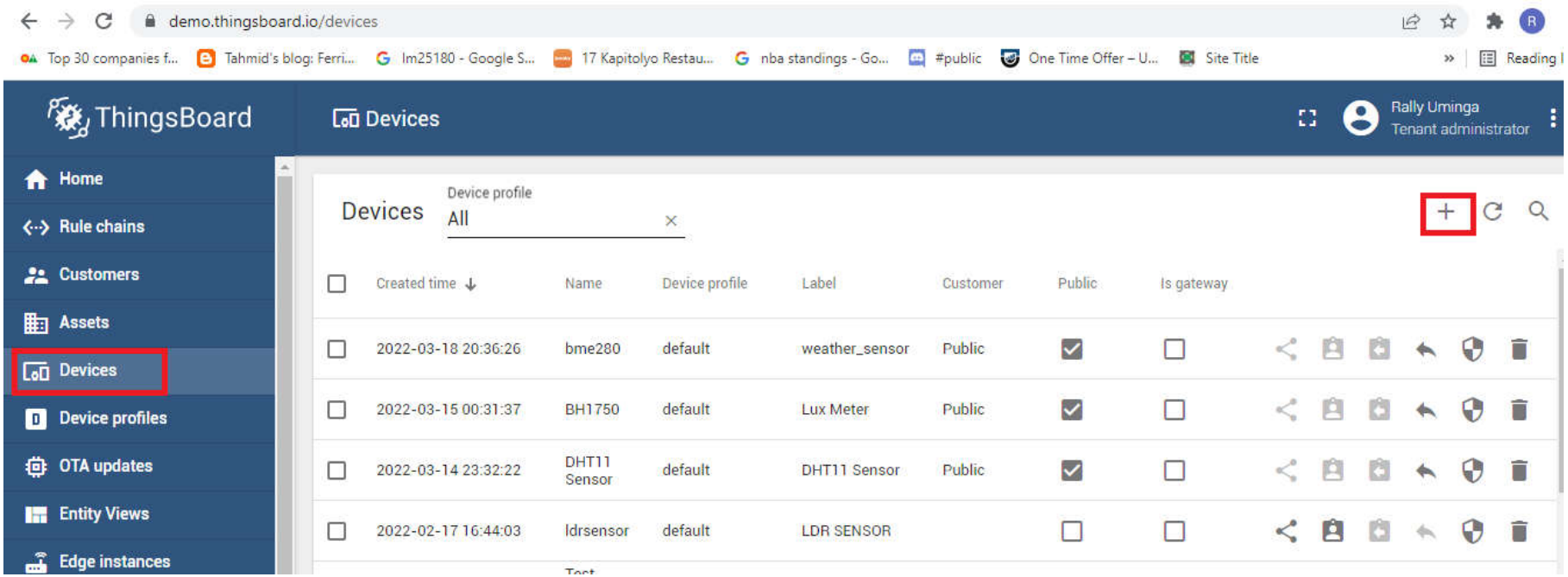
# IoT using ESP8266 WiFi Module

```
// Read Humidity
   float newH = dht.readHumidity();
   // if humidity read failed, don't change h value
   if (isnan(newH)) {
     Serial.println("Failed to read from DHT sensor!");
   }
   else {
     h = newH;
     Serial.println(h);
   }

 if (WiFi.status() != WL_CONNECTED) {
   reconnect();
 }

 if (!tb.connected()) {
   // Connect to the ThingsBoard
   Serial.print("Connecting to: ");
   Serial.print(THINGSBOARD_SERVER);
   Serial.print(" with token ");
   Serial.println(TOKEN);
   if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
     Serial.println("Failed to connect");
     return;
   }
 }

 Serial.println("Sending data...");

 // Uploads new telemetry to ThingsBoard using MQTT.
 // See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
 // for more details

 //tb.sendTelemetryInt("temperature", 22);
 //tb.sendTelemetryFloat("lux", lux);
```

# IoT using ESP8266 WiFi Module

```
tb.sendTelemetryFloat("temperature", t);
  tb.sendTelemetryFloat("humidty", h);

  tb.loop();

}

void InitWiFi()
{
  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  // Loop until we're reconnected
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to AP");
  }
}
```

# IoT using ESP8266 WiFi Module

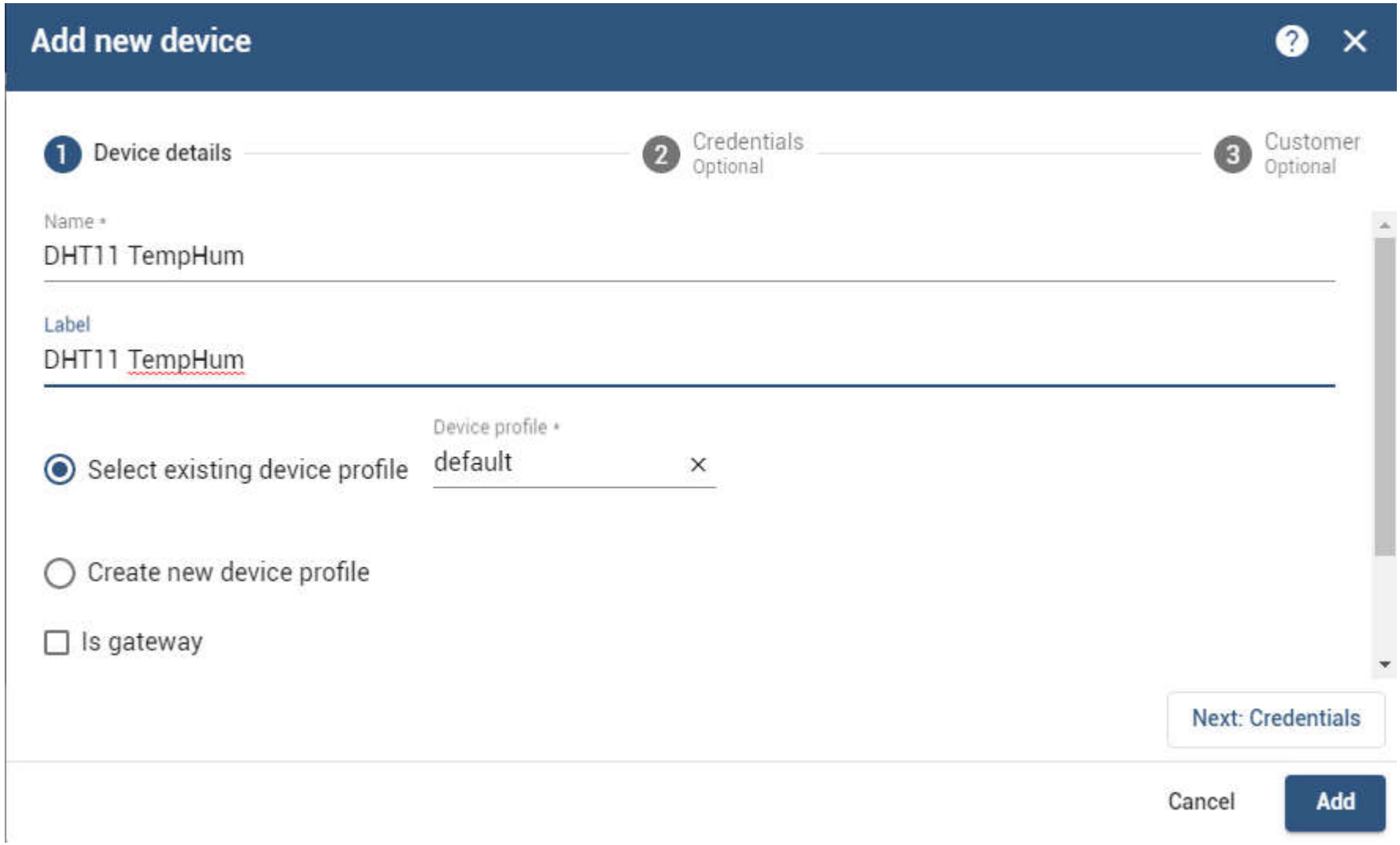


# IoT using ESP8266 WiFi Module

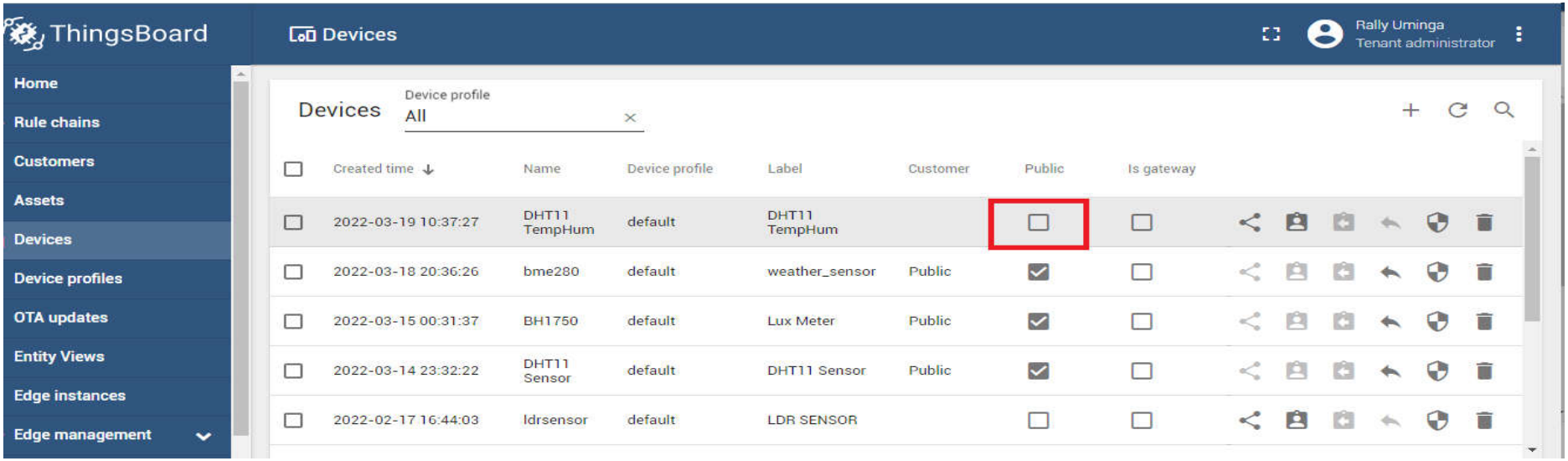
## Steps in Making Dashboard in ThingsBoard

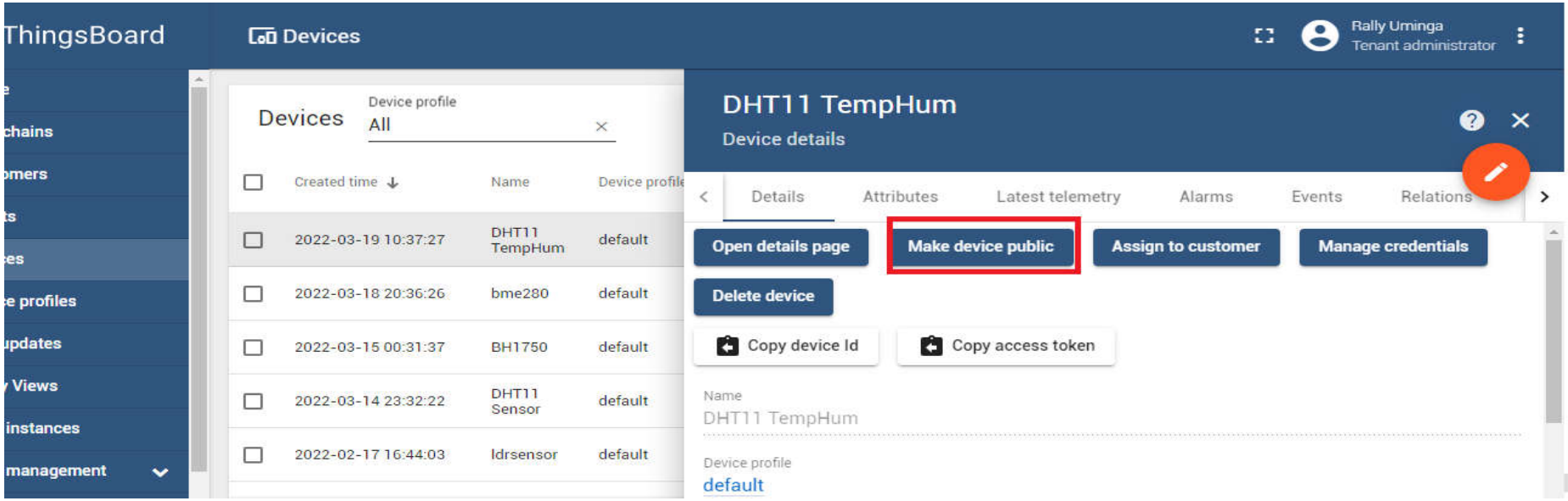
Go to https://demo.thingsboard.io/.
Go to devices then Add Device and Add New Device then fill-up
Make public the device
Click Manage credentials then copy access token
Click the Device then go to latest telemetry
Wait for data to come after uploading the code/sketch
Go to Dashboards
Add Dashboard then Create New Dashboard and fill-up dashboard name and description
Make it Public then copy public link
Click Open Dashboard
Click Add New Widget then Create New Widget
Choose Digital Gauges or any widgets you like
Choose Gauge for this example
Click Add then fill-up Entity alias to temperature(example)
Fill-up Filter type for Single Entity and Type as Device and Device is DHT11 TempHum(example) then click Add
Add Entity time series the temperature alias then click Add
You can now Edit the temperature widget and add also add Humidity widget

# IoT using ESP8266 WiFi Module

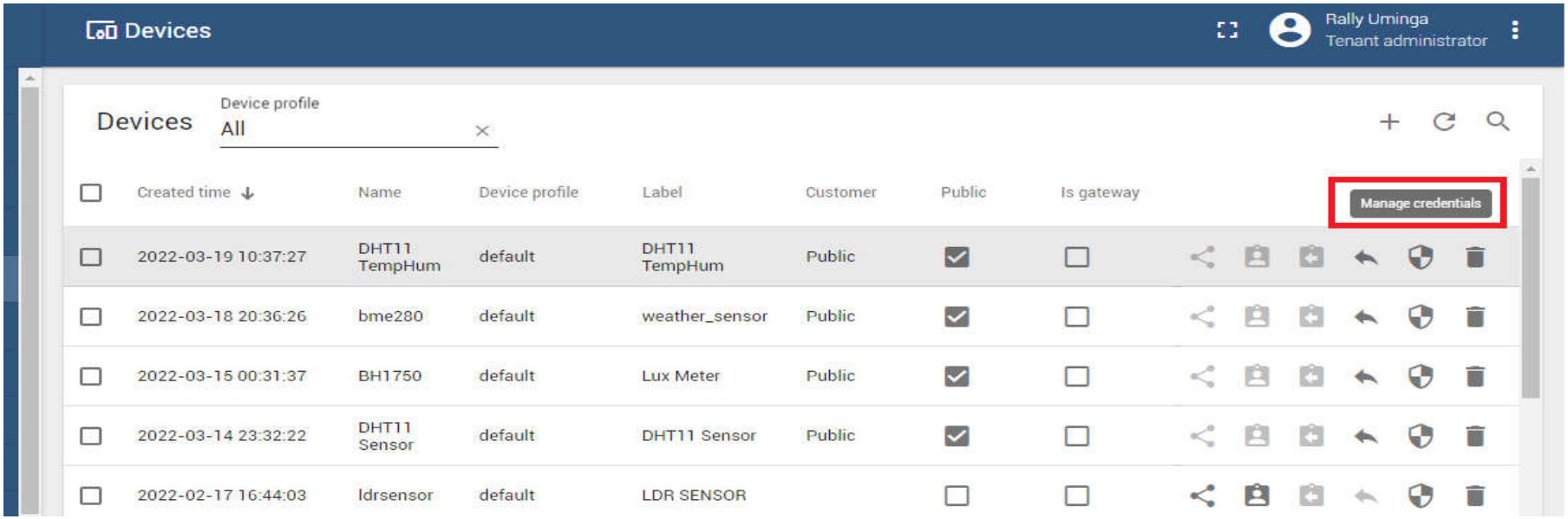


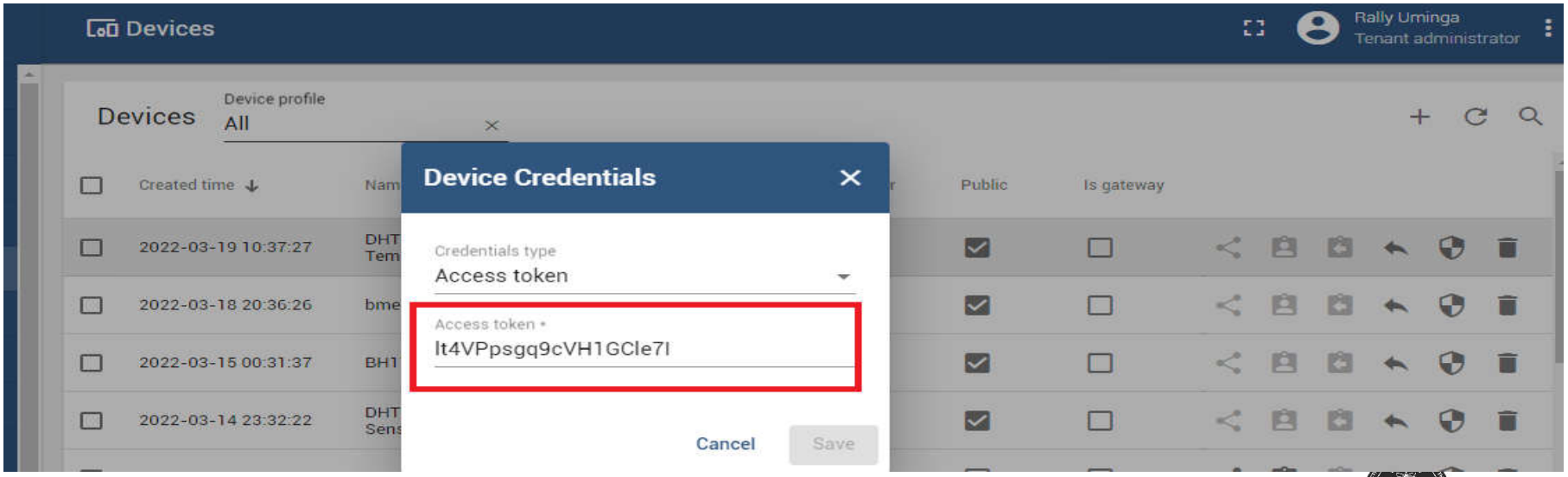
# IoT using ESP8266 WiFi Module



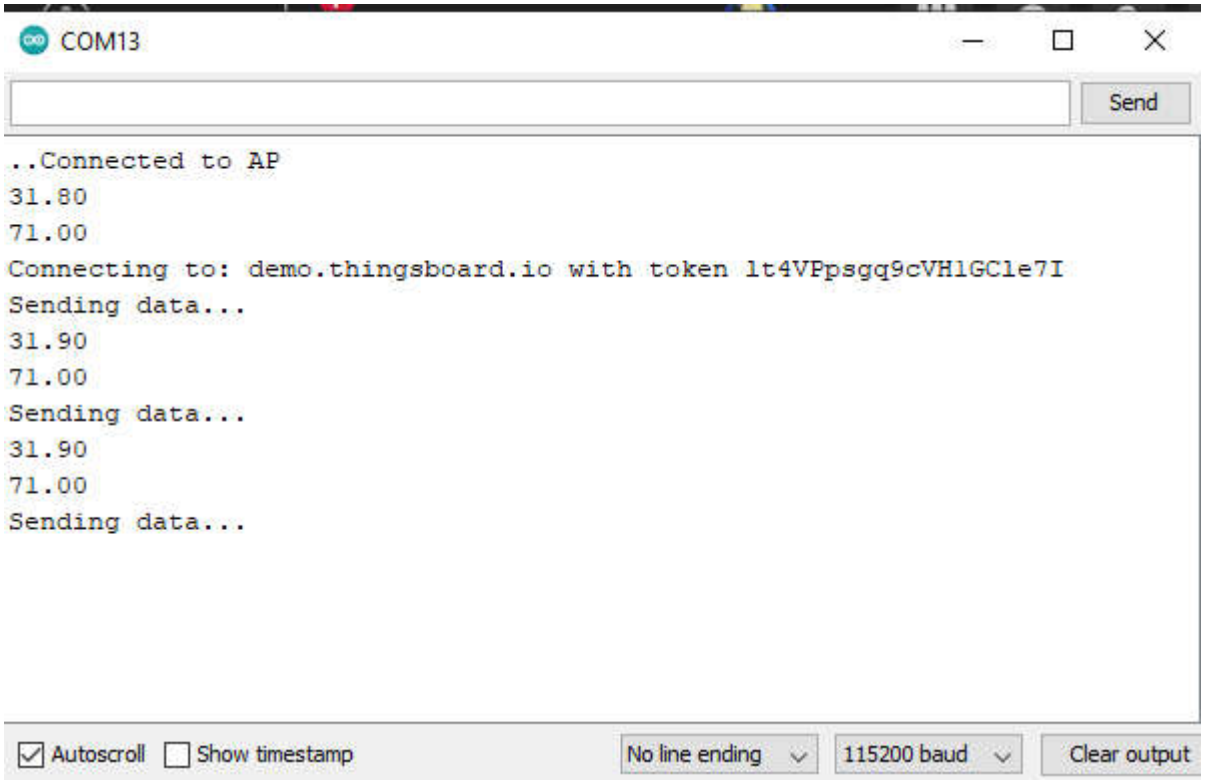
# IoT using ESP8266 WiFi Module



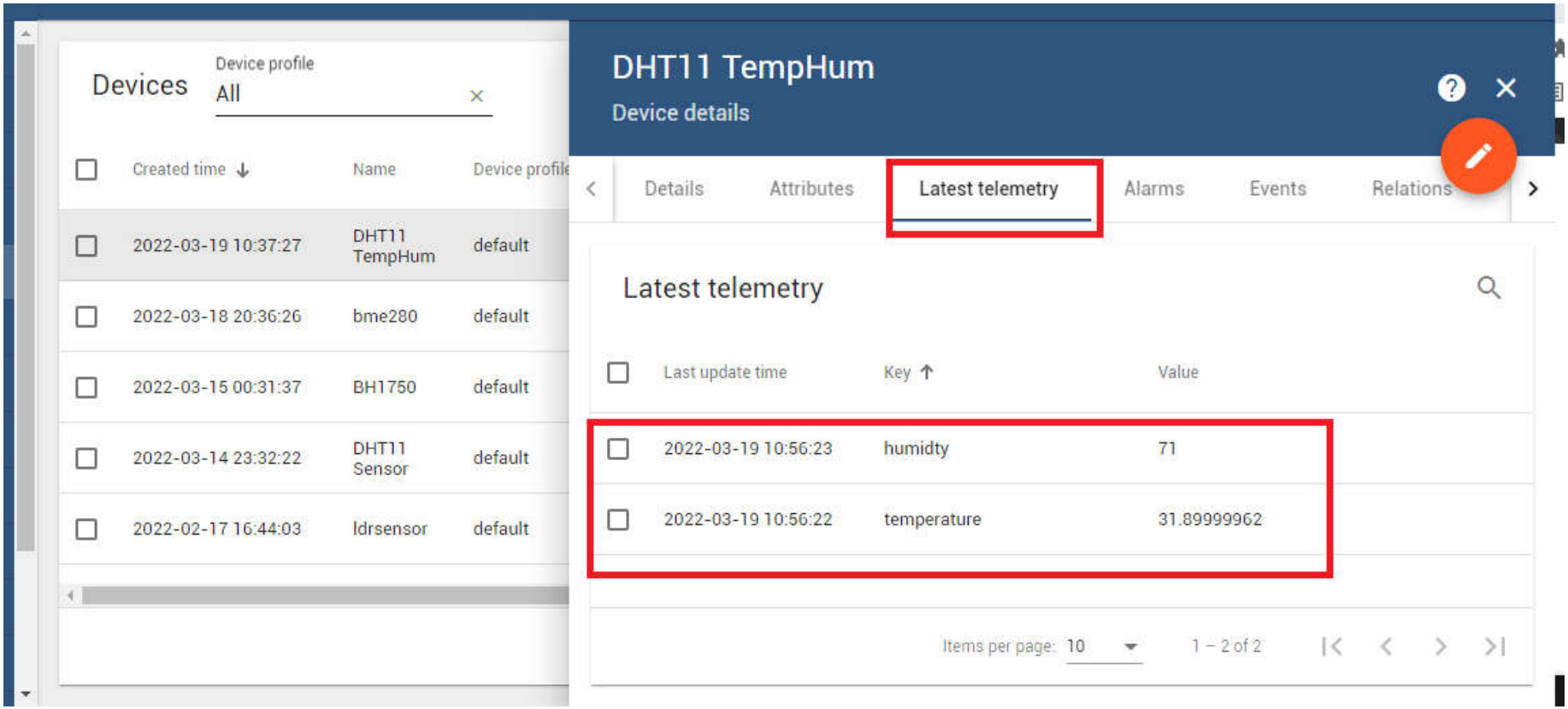
# IoT using ESP8266 WiFi Module





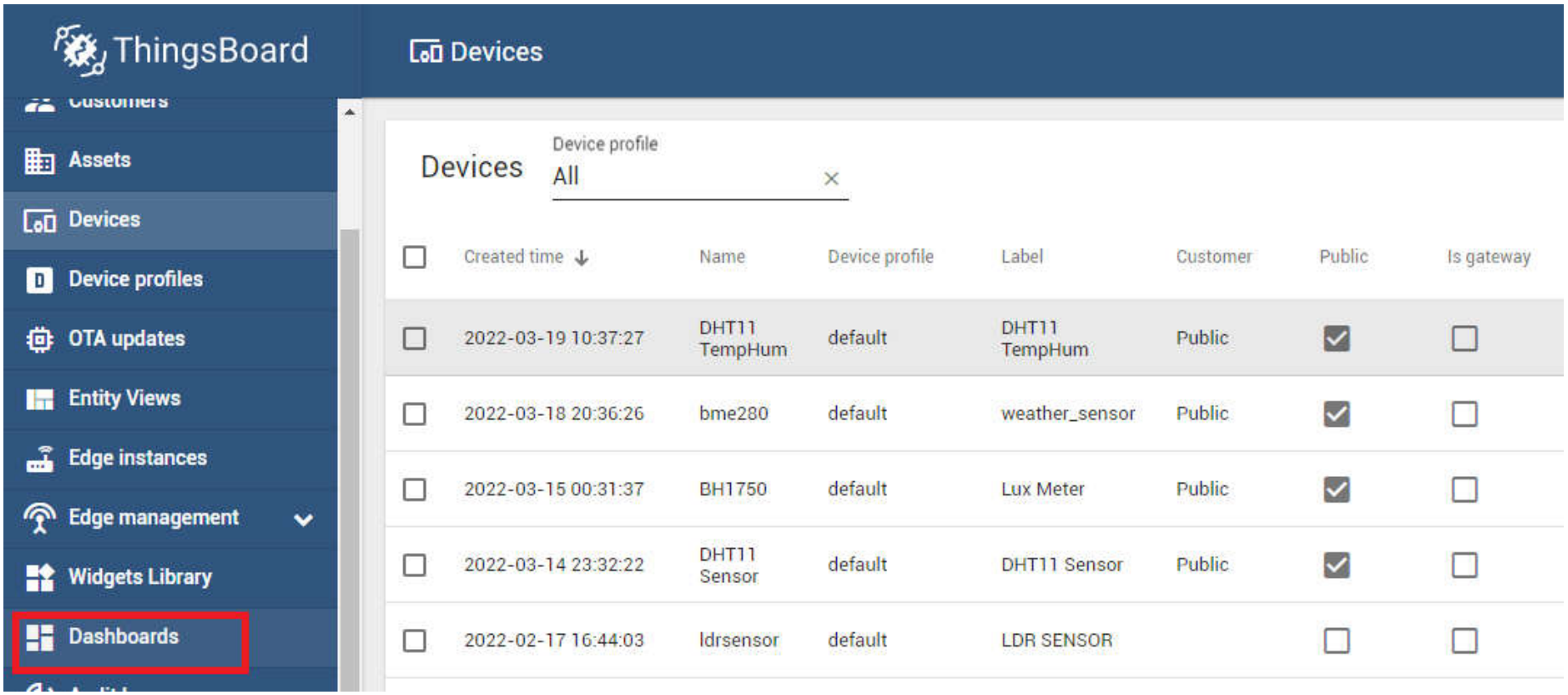
# IoT using ESP8266 WiFi Module



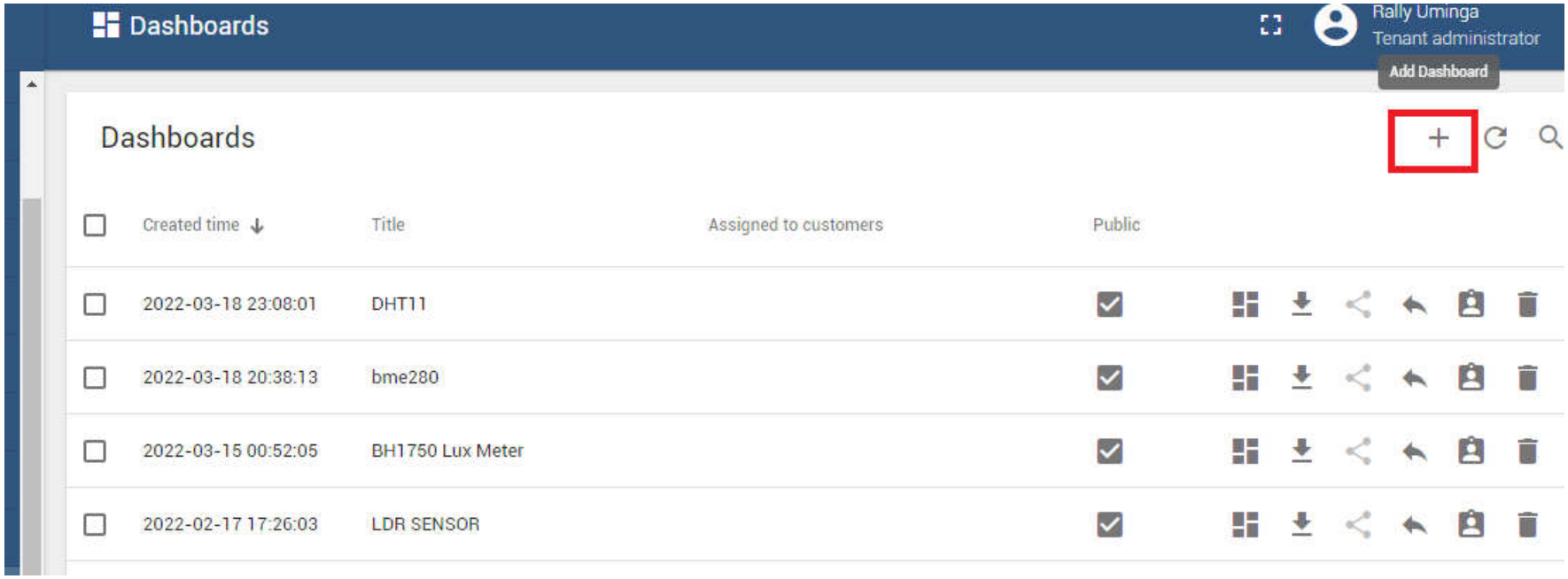


# IoT using ESP8266 WiFi Module

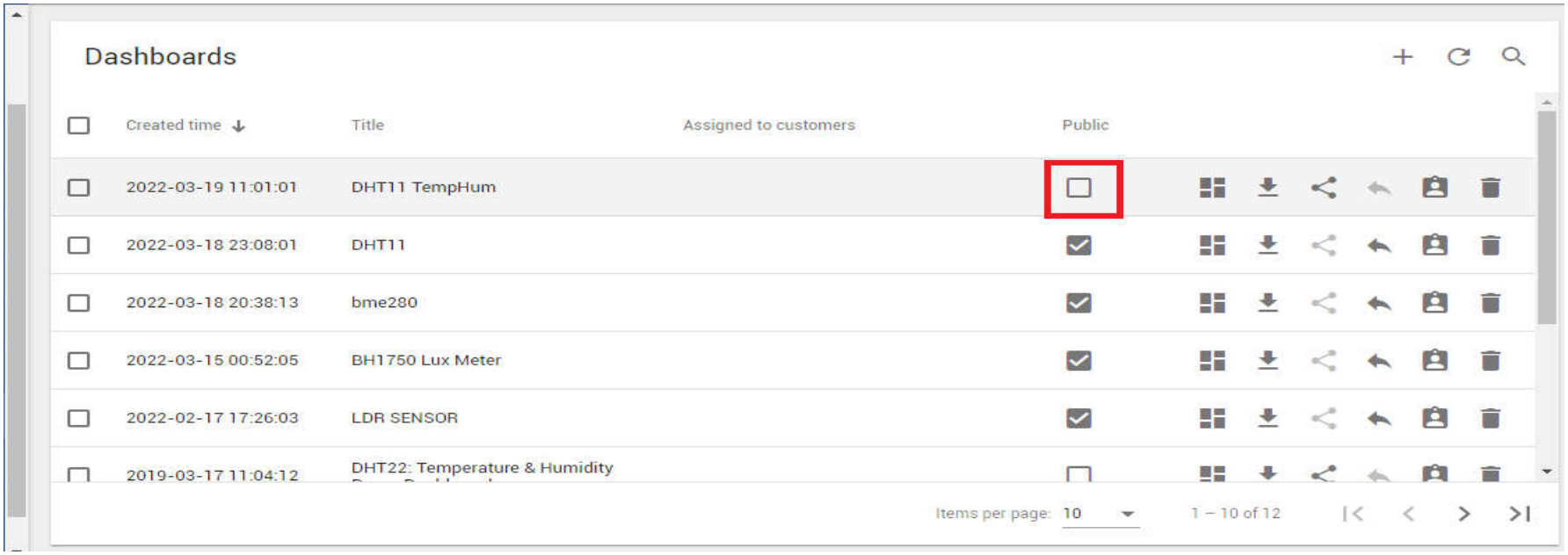


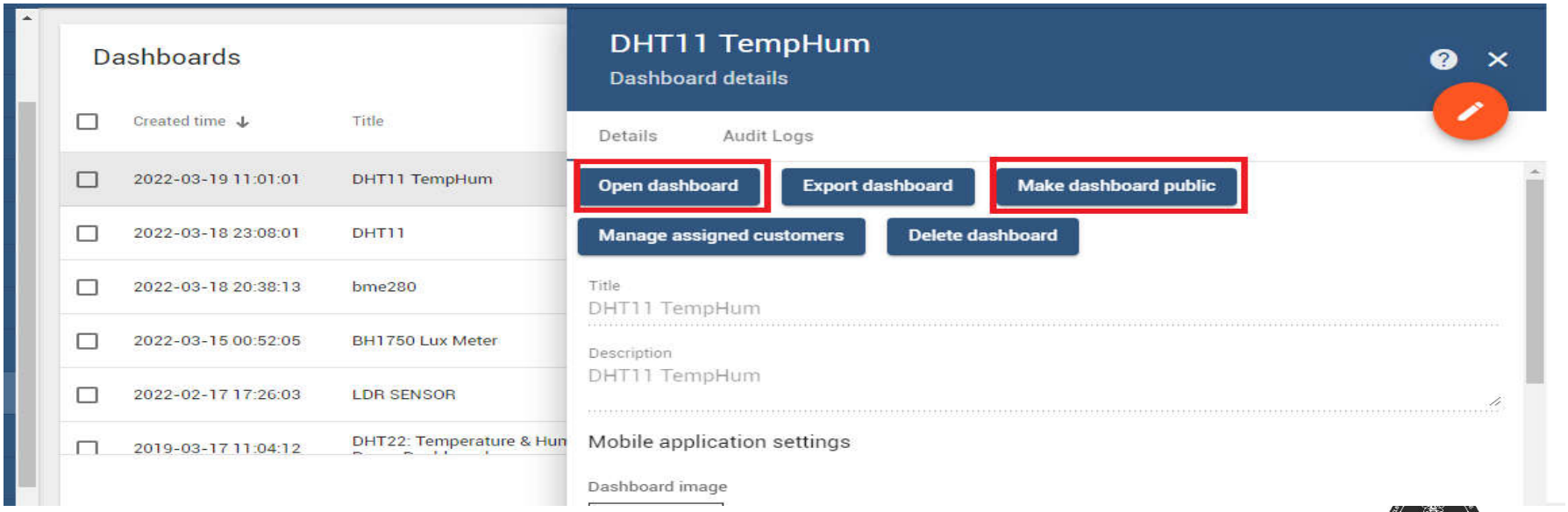


# IoT using ESP8266 WiFi Module




# IoT using ESP8266 WiFi Module



# IoT using ESP8266 WiFi Module



**Dashboard is now public**

Your dashboard **DHT11 TempHum** is now public and accessible via next public link:

https://demo.thingsboard.io/dashboard/cf01e740-a730-11ec-86b8-333ff61bdca3?publicId=d4bb27c0-3af0-11ec-a0a8-5356

**Note:** Do not forget to make related devices public in order to access their data.

OK

**DHT11 TempHum** DHT11 TempHum Realtime - last minute

No widgets configured

Enter edit mode

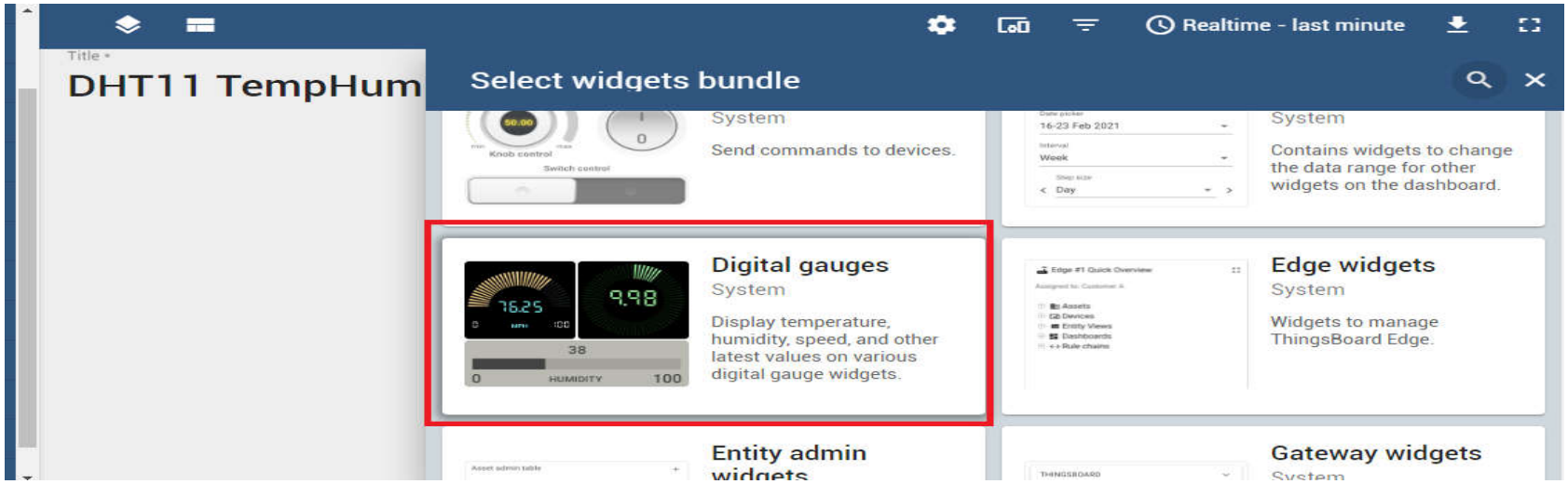
Powered by Thingsboard v.3.3.4

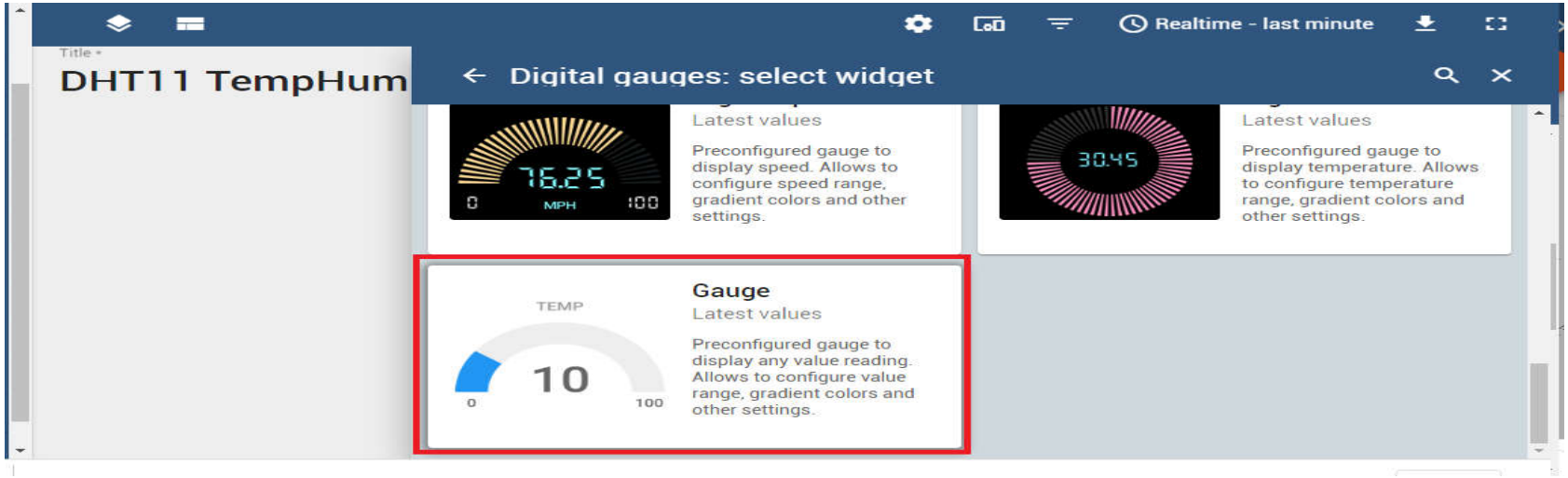
# IoT using ESP8266 WiFi Module



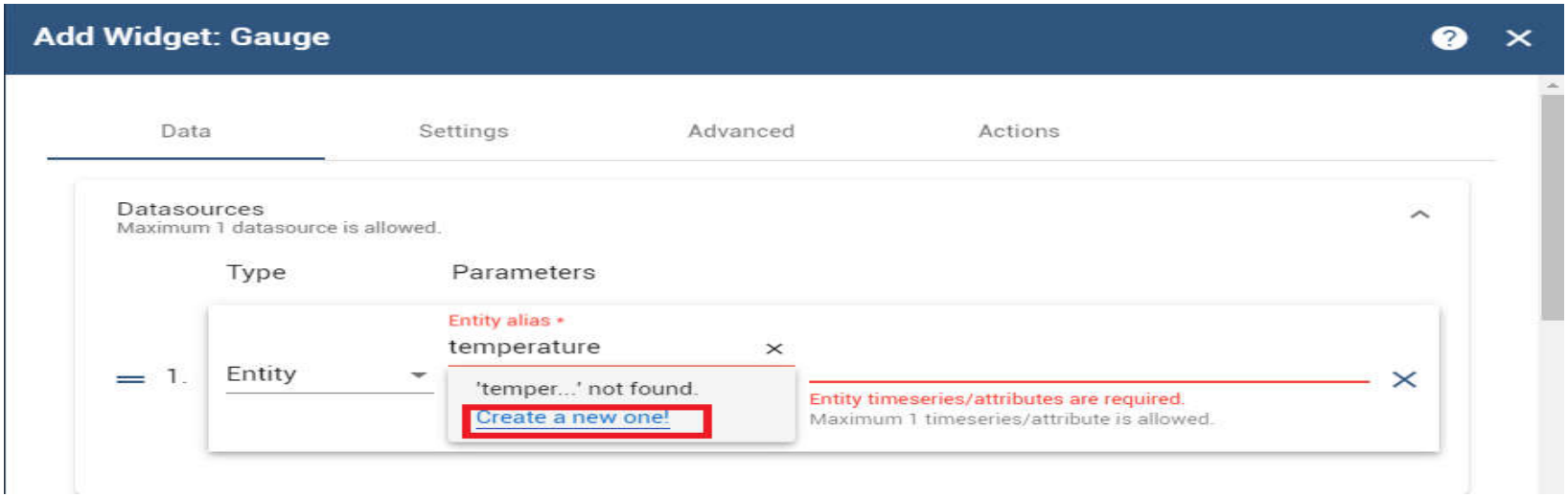

# IoT using ESP8266 WiFi Module



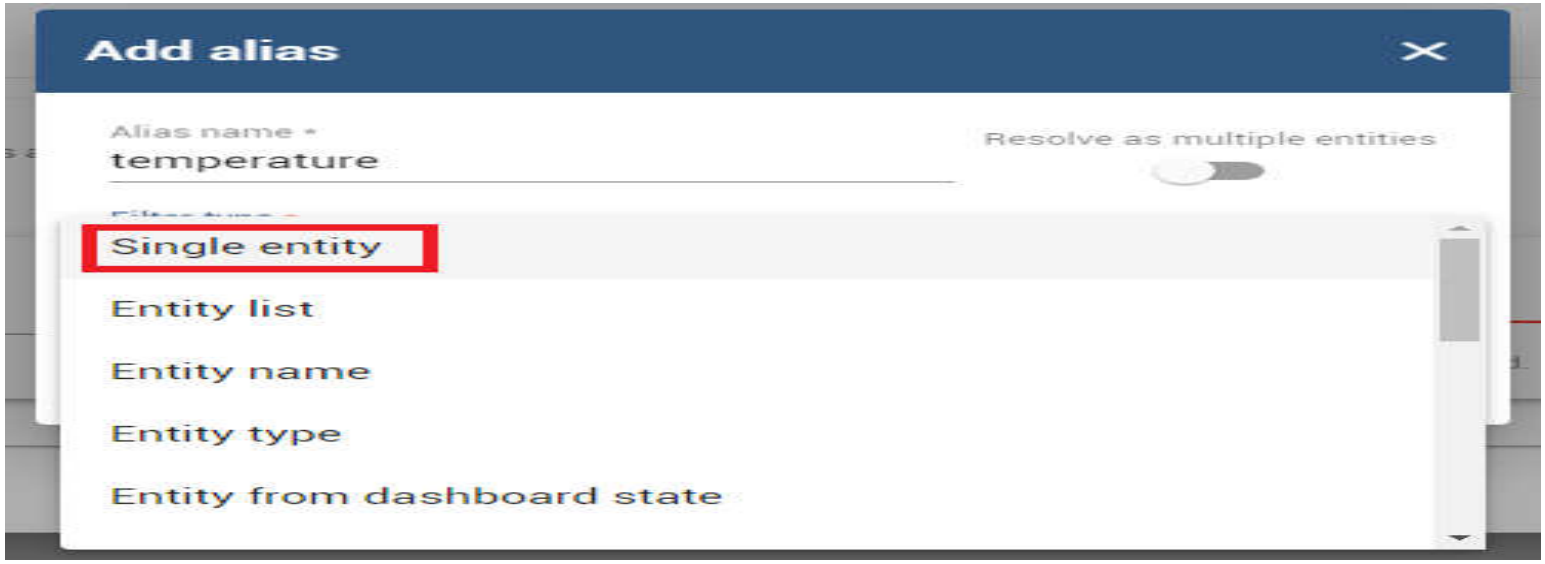


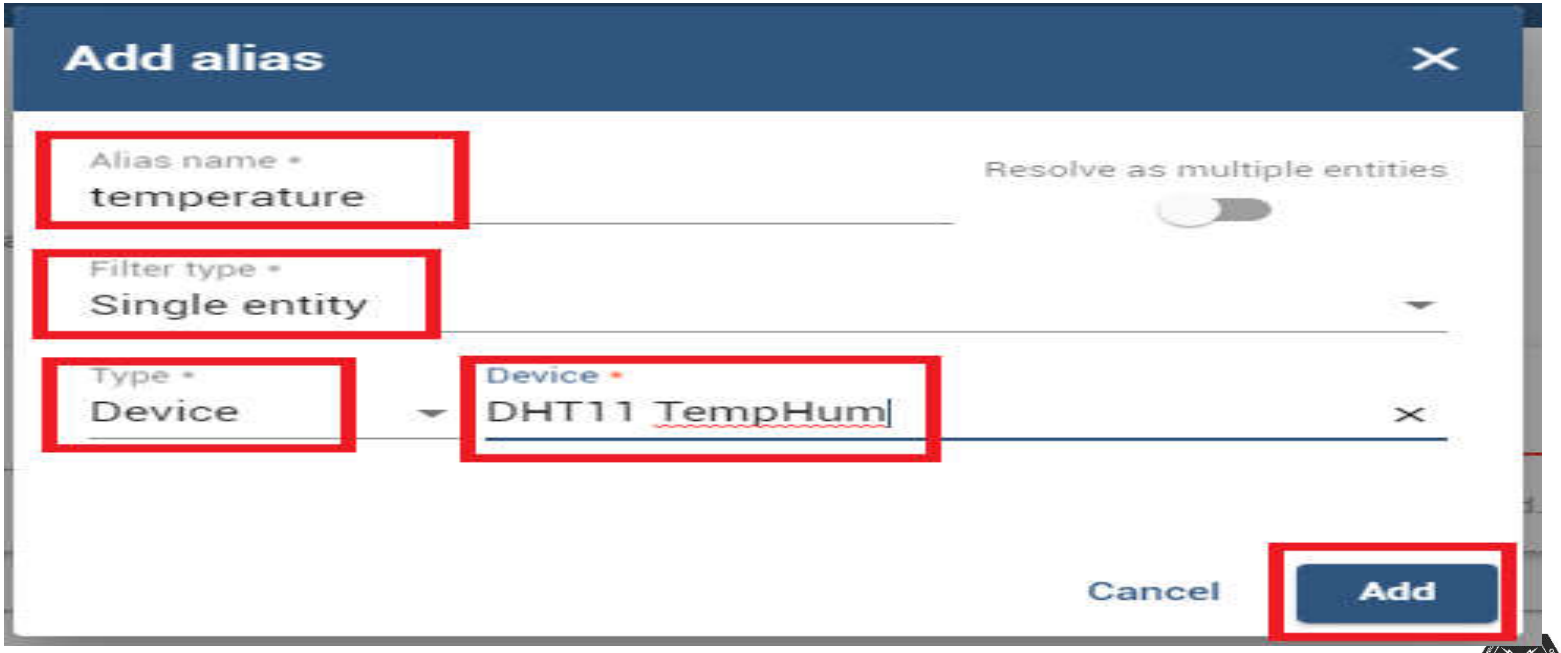



# IoT using ESP8266 WiFi Module

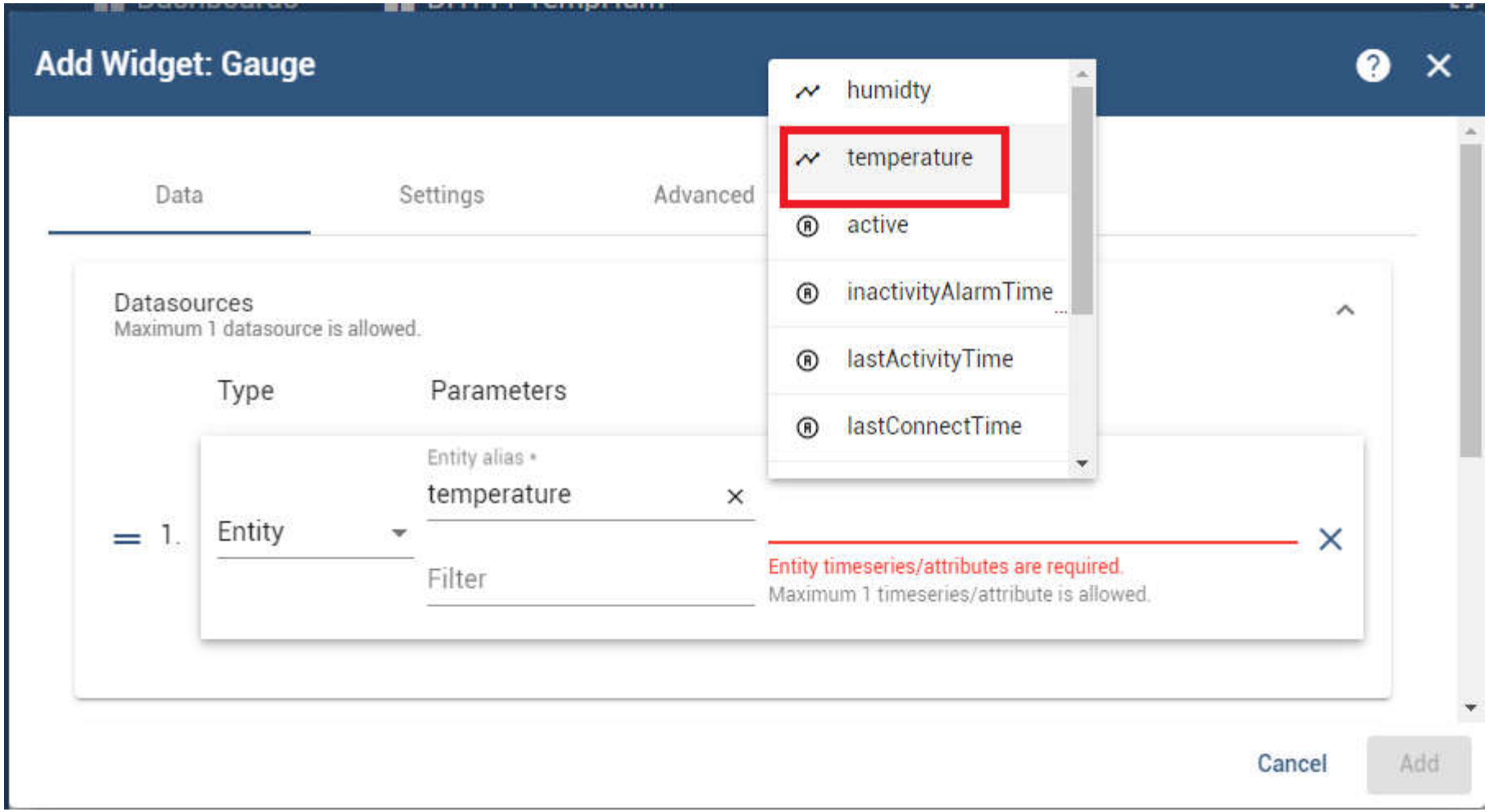
# IoT using ESP8266 WiFi Module



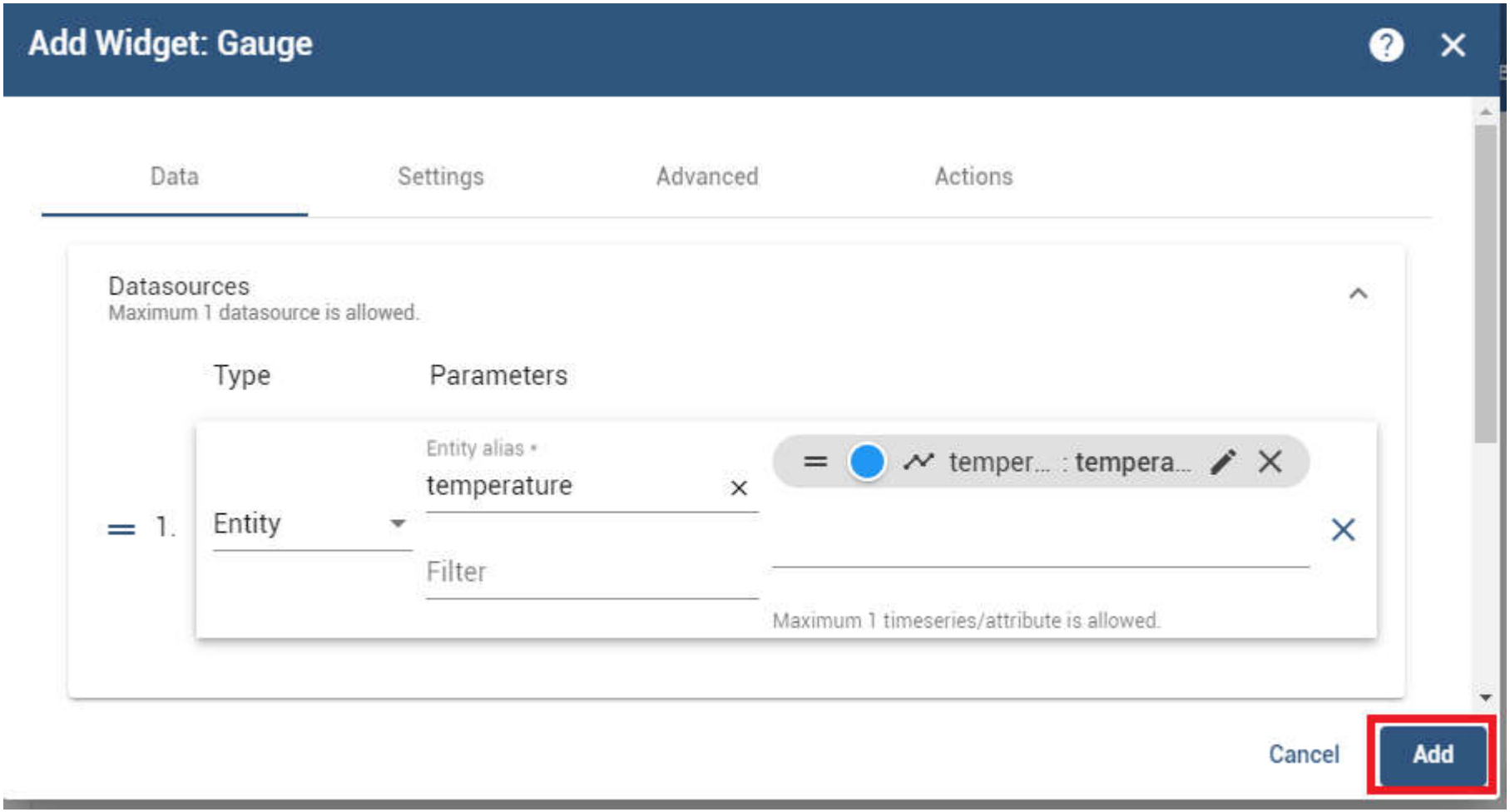
# IoT using ESP8266 WiFi Module





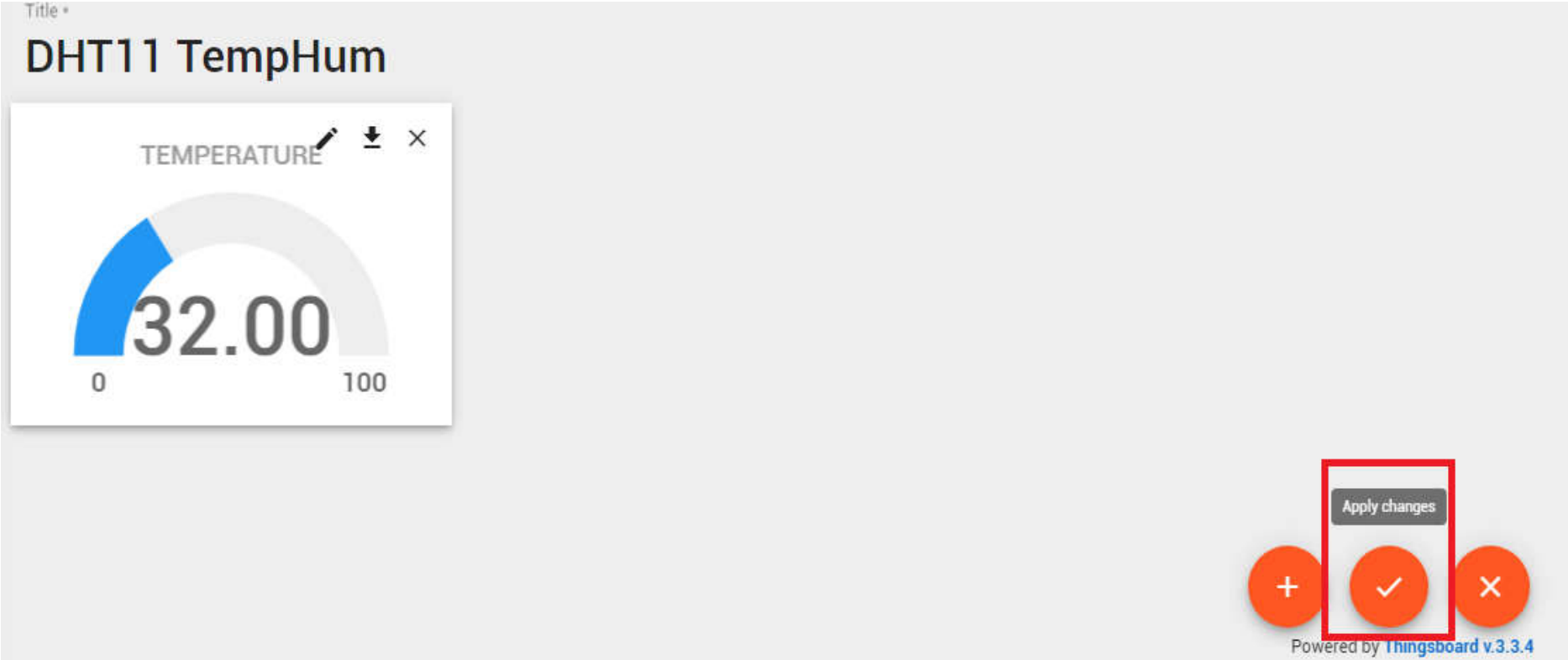
# IoT using ESP8266 WiFi Module





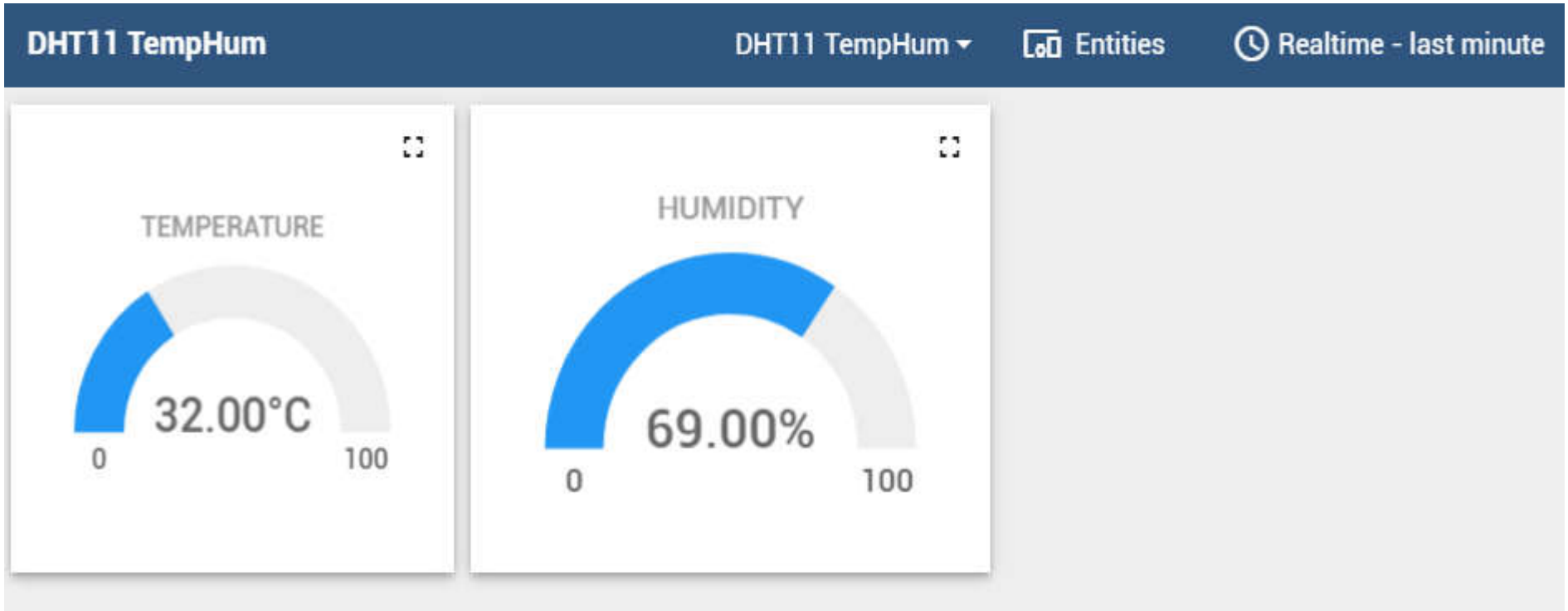
# IoT using ESP8266 WiFi Module





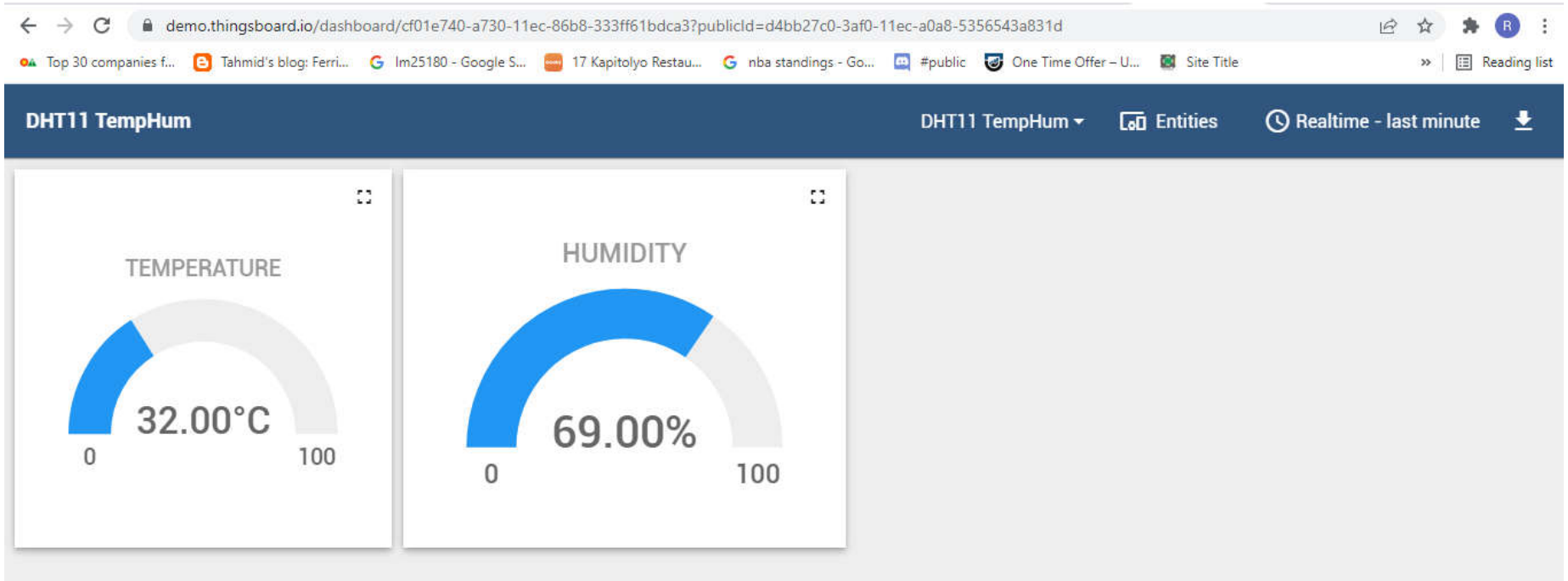
# IoT using ESP8266 WiFi Module





# IoT using ESP8266 WiFi Module






**PUBLIC LINK OF THE DHT11 TempHum SENSOR**

# IoT using ESP8266 WiFi Module

```
//BME280_ThingsBoard.ino
#include <Wire.h>              // include Arduino wire library (required for I2C devices)
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include "ThingsBoard.h"
#include <ESP8266WiFi.h>    // for ESP8266
//#include <WiFi.h>             // for ESP32
#define SEALEVELPRESSURE_HPA (1013.25)

#define WIFI_AP             “SSID"
#define WIFI_PASSWORD       “PASSWORD"

// See https://thingsboard.io/docs/getting-started-guides/helloworld/
// to understand how to obtain an access token
#define TOKEN               "W5i8LOlikhcpioanKx2g"
#define THINGSBOARD_SERVER  "demo.thingsboard.io"
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD   115200

Adafruit_BME280 bme; // I2C

// Initialize ThingsBoard client
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;

unsigned long delayTime;
```

# IoT using ESP8266 WiFi Module

```
void setup(void)
{
// initialize serial for debugging
  Serial.begin(SERIAL_DEBUG_BAUD);
  Serial.println(F("BME280 test"));

  bool status;

  // default settings
  // (you can also pass in a Wire library object like &Wire2)
  status = bme.begin(0x76);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  Serial.println("-- Default Test --");
  delayTime = 1000;

  Serial.println();
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();

  Wire.begin();

}

void loop()
{
  delay(10000);            // wait a second
  //float lux = lightMeter.readLightLevel();
  float temp = bme.readTemperature();
  float pres = bme.readPressure() / 100.0F;
  float alt = bme.readAltitude(SEALEVELPRESSURE_HPA);
  float hum = bme.readHumidity();
```

# IoT using ESP8266 WiFi Module

```
if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }

  if (!tb.connected()) {
    // Connect to the ThingsBoard
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }

  Serial.println("Sending data...");

  // Uploads new telemetry to ThingsBoard using MQTT.
  // See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
  // for more details

  //tb.sendTelemetryInt("temperature", 22);
  //tb.sendTelemetryFloat("lux", lux);
  tb.sendTelemetryFloat("temperature", temp);
  tb.sendTelemetryFloat("pressure", pres);
  tb.sendTelemetryFloat("altitude", alt);
  tb.sendTelemetryFloat("humidty", hum);

  tb.loop();

}
```

# IoT using ESP8266 WiFi Module

```
void InitWiFi()
{
  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  // Loop until we're reconnected
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to AP");
  }
}
```

# IoT using ESP8266 WiFi Module



```
//BH1750_OLED_ESP8266.ino
#include <Wire.h>              // include Arduino wire library (required for I2C devices)
#include <Adafruit_GFX.h>      // include Adafruit graphics library
#include <Adafruit_SSD1306.h>  // include Adafruit SSD1306 OLED display driver
#include <BH1750.h>
#include "ThingsBoard.h"
#include <ESP8266WiFi.h>

#define WIFI_AP             “SSID"
#define WIFI_PASSWORD       “PASSWORD"
// See https://thingsboard.io/docs/getting-started-guides/helloworld/
// to understand how to obtain an access token
#define TOKEN               "tLivHjcOobUuz3f6lidz"
#define THINGSBOARD_SERVER  "demo.thingsboard.io"
// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD   115200
#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 64 // OLED display height, in pixels
#define OLED_RESET  -1    // define display reset pin
#define SCREEN_ADDRESS 0x3C  //0x3D ///< See datasheet for Address; 0x3D for 128x64, 0x3C for
128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
//Adafruit_SSD1306 display(-1);  // initialize Adafruit display library

// Initialize ThingsBoard client
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;

BH1750 lightMeter;
```

# IoT using ESP8266 WiFi Module



```
void setup(void)
{
// initialize serial for debugging
  Serial.begin(SERIAL_DEBUG_BAUD);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();
  // initialize the SSD1306 OLED display with I2C address = 0x3C
  display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS);
  Wire.begin();
  lightMeter.begin();

  // clear the display buffer.
  display.clearDisplay();
  display.setTextSize(1);   // text size = 1
  display.setTextColor(WHITE, BLACK);  // set text color to white and black background
  display.setTextWrap(false);           // disable text wrap
  display.setCursor(29, 0);
  display.print("BH1750 TEST");
  display.setCursor(0, 16);
  display.print("Light:");
  display.setCursor(80, 16);
  display.print(" lx");
  display.display();        // update the display

}

char _buffer[12];

void loop()
{
  delay(10000);            // wait a second
  float lux = lightMeter.readLightLevel();
  sprintf(_buffer, " %02u.%02u ", (int)lux, (int)(lux * 100) % 100 );
  display.setCursor(37, 16);
  display.print(_buffer);
  display.display();      // update the display
```

# IoT using ESP8266 WiFi Module



```
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }

  if (!tb.connected()) {
    // Connect to the ThingsBoard
    Serial.print("Connecting to: ");
    Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token ");
    Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }

  Serial.println("Sending data...");

  // Uploads new telemetry to ThingsBoard using MQTT.
  // See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
  // for more details

  //tb.sendTelemetryInt("temperature", 22);
  tb.sendTelemetryFloat("lux", lux);

  tb.loop();

}
```

# IoT using ESP8266 WiFi Module



```
  void InitWiFi()
{
  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  // Loop until we're reconnected
  status = WiFi.status();
  if ( status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to AP");
  }
}
```

# THANK YOU !

# Q&A