

IoT Platforms using ESP8266 and ESP32

IoT Platforms using ESP8266 and ESP32

Contents

Objectives
Arduino IoT Cloud
Thingsboard

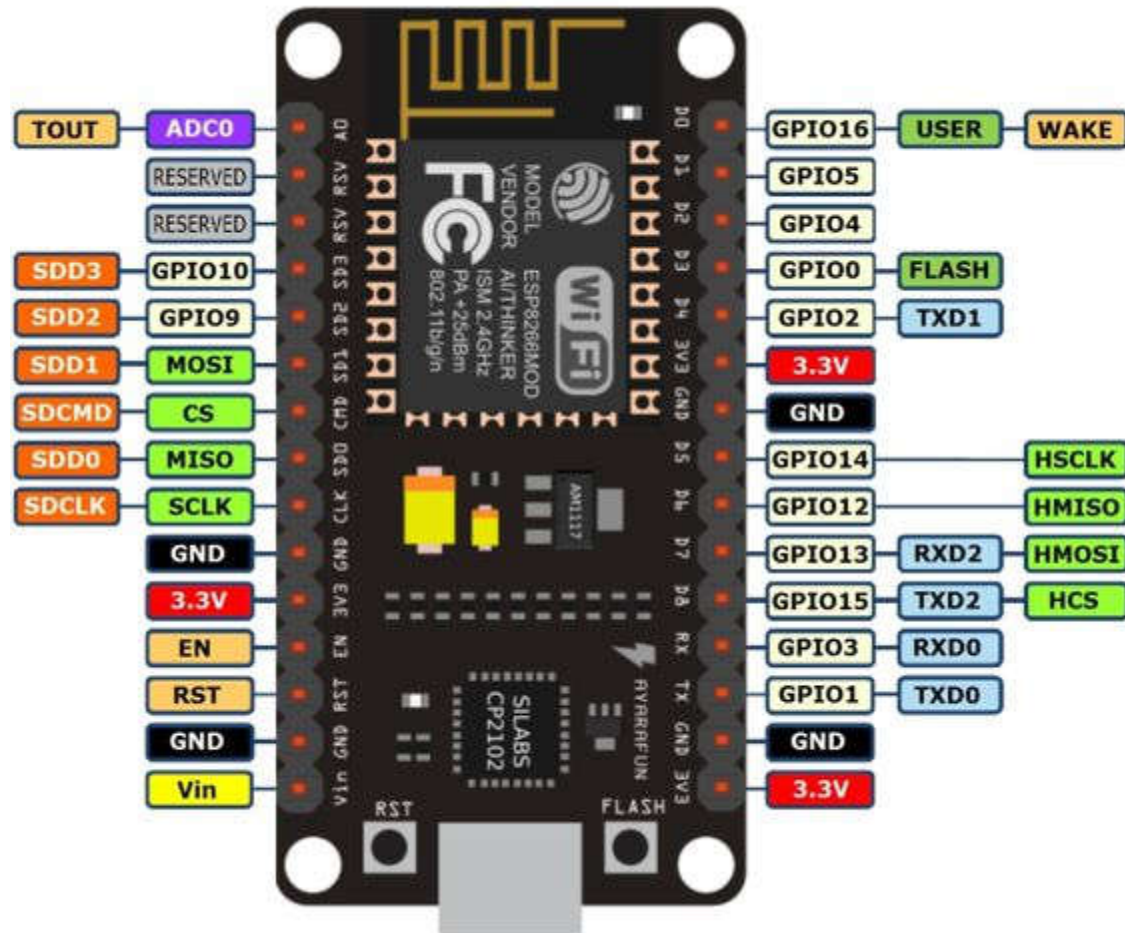


Objectives

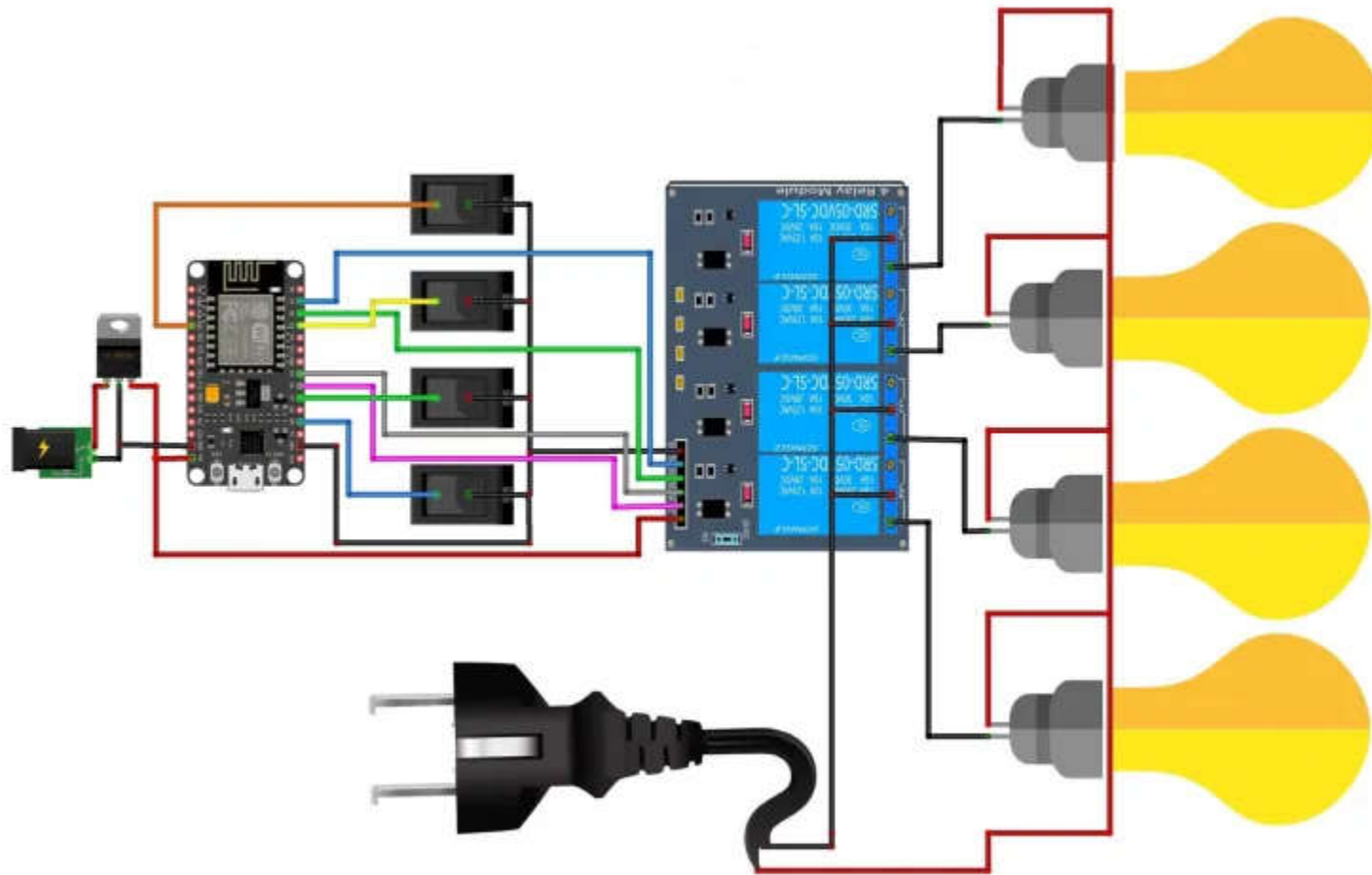
- In this tutorial, you will learn:
 - To familiarize Arduino IoT Cloud using ESP8266 and ESP32.
 - To understand basic Thingsboard dashboard using ESP8266 and ESP32.



IoT using ESP8266 WiFi Module



IoT using ESP8266 WiFi Module



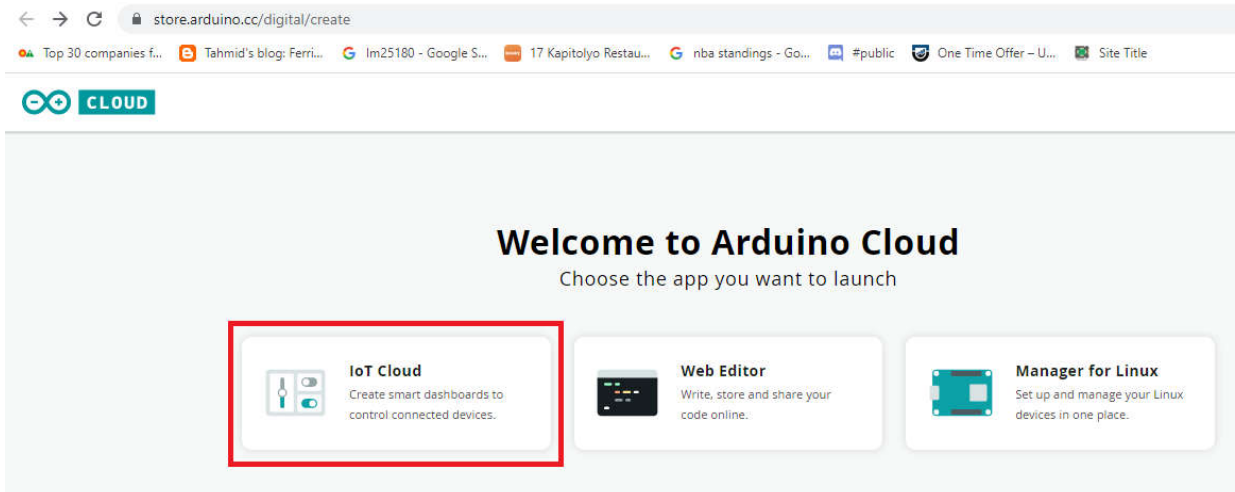
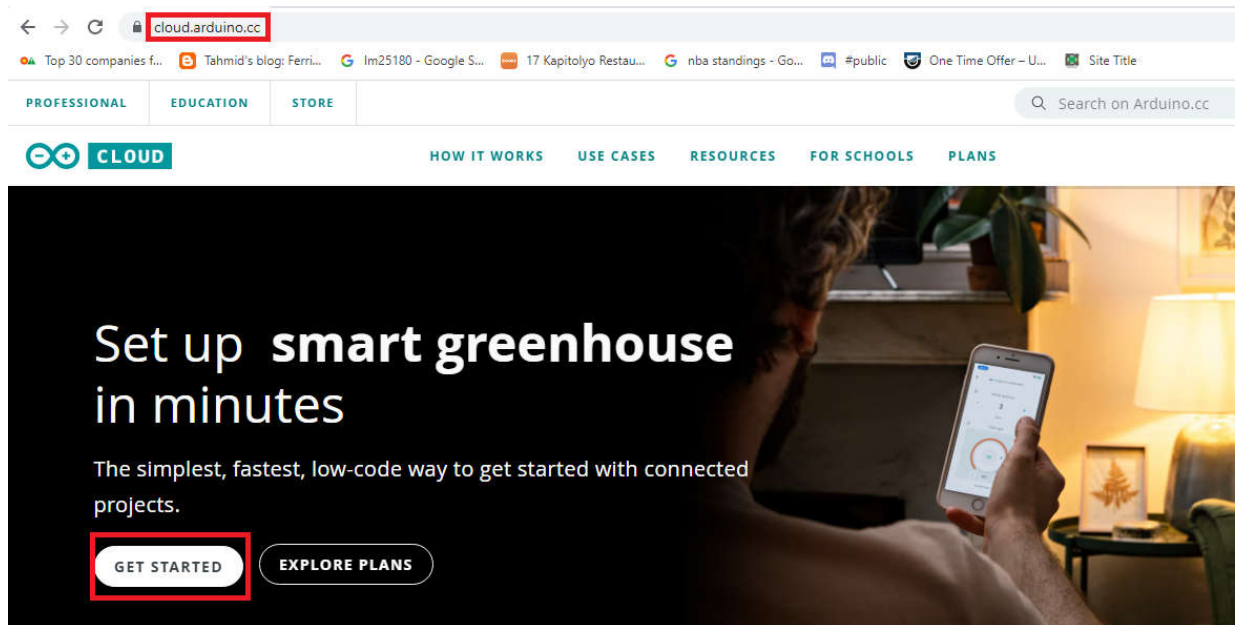
IoT using ESP8266 WiFi Module

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

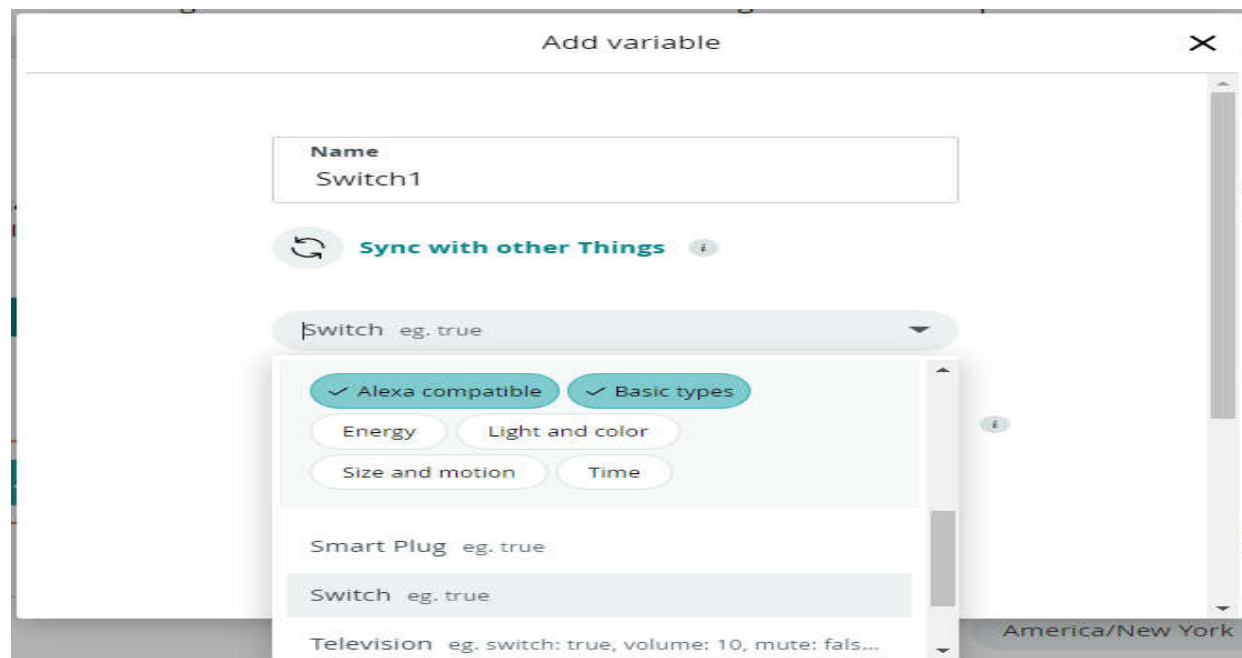
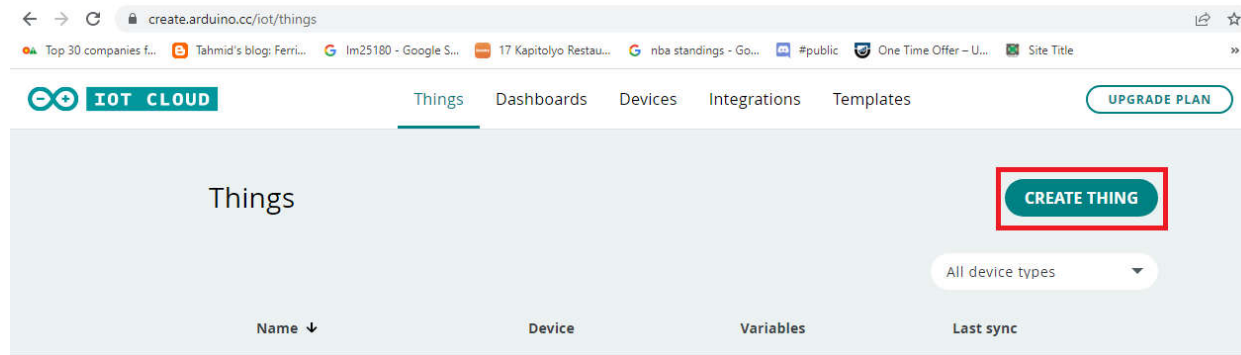
1. Click get started.
2. Click IoT Cloud.
3. Create Thing.
4. Add variable.
5. Name Variable Switch1.
6. Variable type Cloud Switch.
7. Click Add variable to create first variable.
8. Select Device then ESP8266 then NodeMCU1.0 ESP-12E Module.
9. Click continue and name it IECEP_SmartHome.
10. Get device 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

Add variable

Name

Switch1

↺

Sync with other Things

ⓘ

Switch

eg. true

▼

Declaration

CloudSwitch switch1;

ⓘ

Variable Permission

ⓘ

☒ Read & Write

☐ Read Only

Variable Update Policy

ⓘ

☒ On change

☐ Periodically

ADD VARIABLE

CANCEL

IoT using ESP8266 WiFi Module

Setup

Variables

ADD

Name ↓	Last Value	Last Update
<input type="checkbox"/> Switch1 CloudSwitch switch1;	-	-


Variables

ADD

Name ↓	Last Value	Last Update
<input type="checkbox"/> Switch1 CloudSwitch switch1;	-	-
<input type="checkbox"/> Switch2 CloudSwitch switch2;	-	-
<input type="checkbox"/> Switch3 CloudSwitch switch3;	-	-

Device

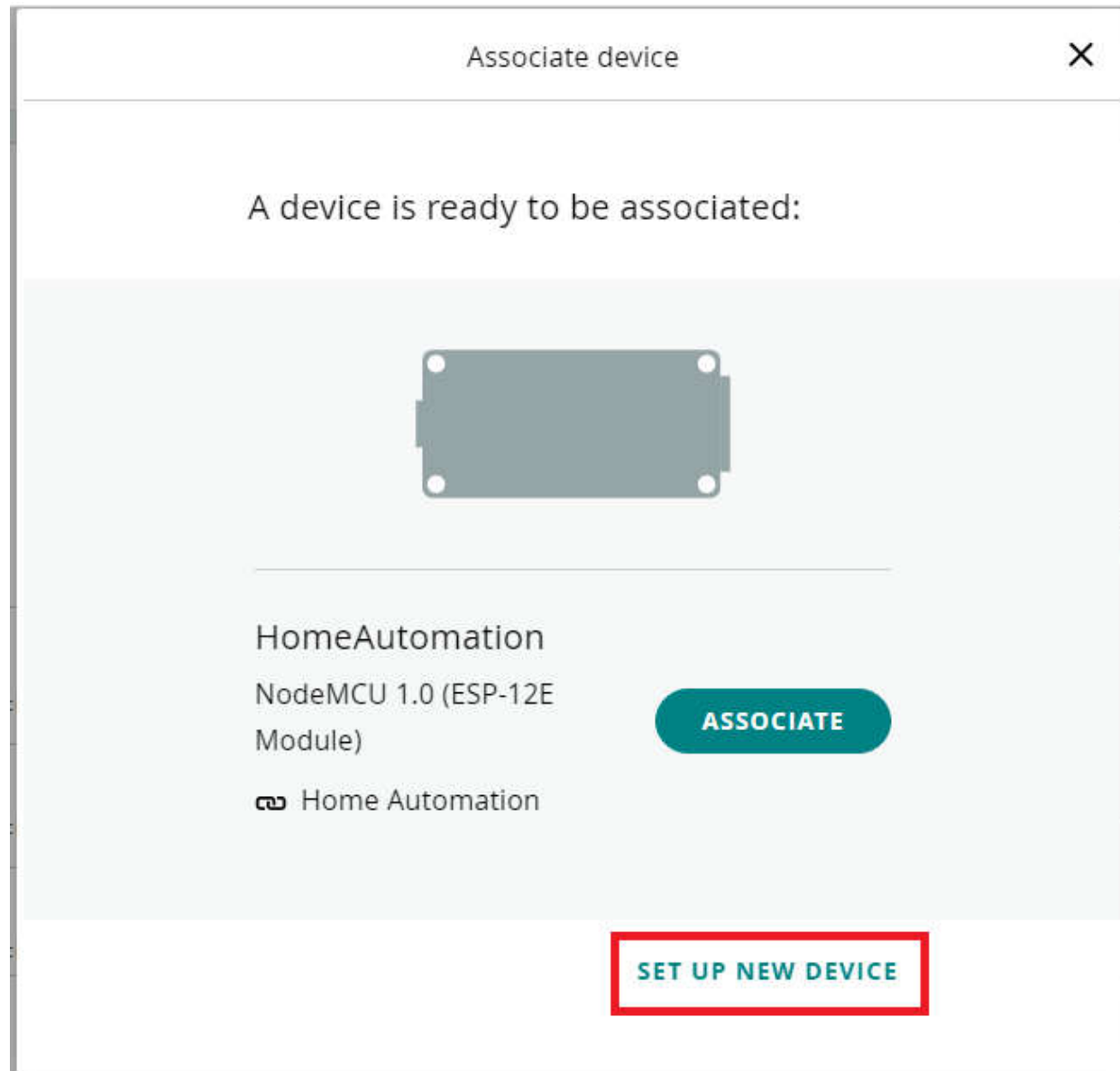
Select the device you want to use or configure a new one.



Select Device

Network

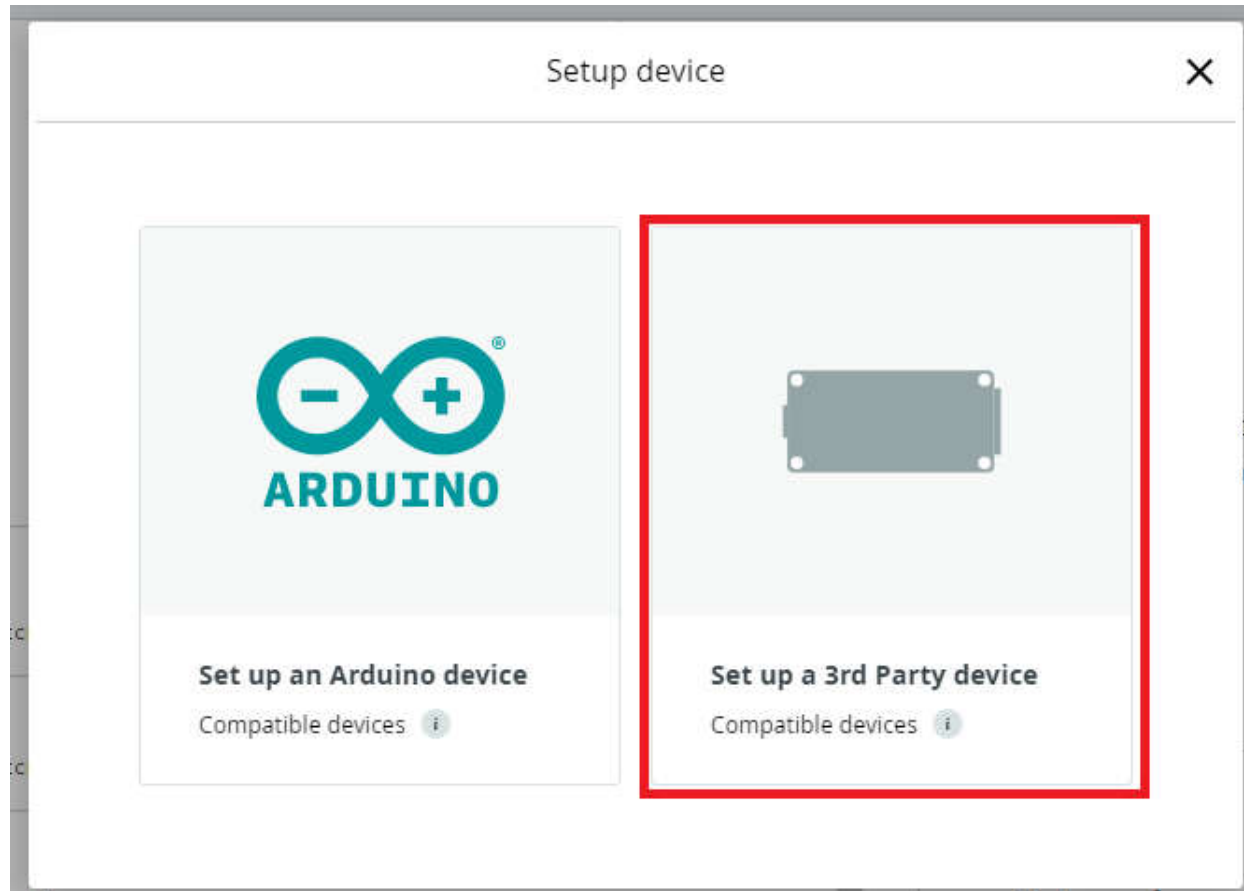
IoT using ESP8266 WiFi Module





IoT using ESP8266 WiFi Module



IoT using ESP8266 WiFi Module



IoT using ESP8266 WiFi Module

 Setup device 

Select device type

Please select the device type and model you want to configure

☒ ESP8266 ☐ ESP32 ☐ LoRaWAN

NodeMCU 1.0 (ESP-12E Module) ▼


CONTINUE

IoT using ESP8266 WiFi Module

←

Setup device

×



Give your device a name

Name your device so you will be able to recognize it.

Device Name

IECEP_Smart_Home

↻

NEXT

IoT using ESP8266 WiFi Module

Setup device


Make your device IoT-ready

To use this board you will need a Device ID and a Secret Key, please copy and save them or [download the PDF](#).

Also, keep in mind that this device authentication has a lower security level compared to other Arduino devices.

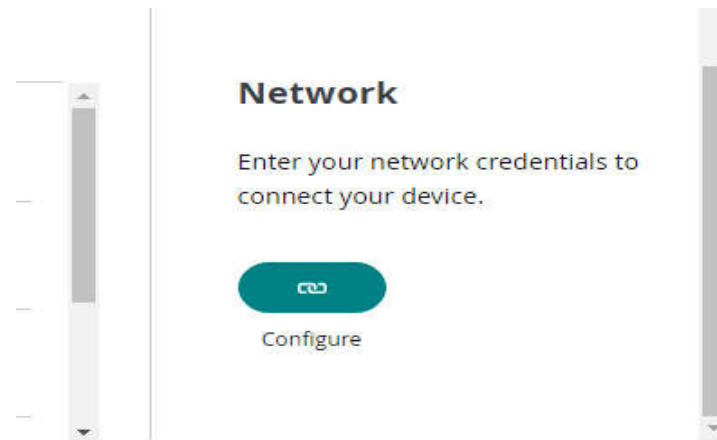
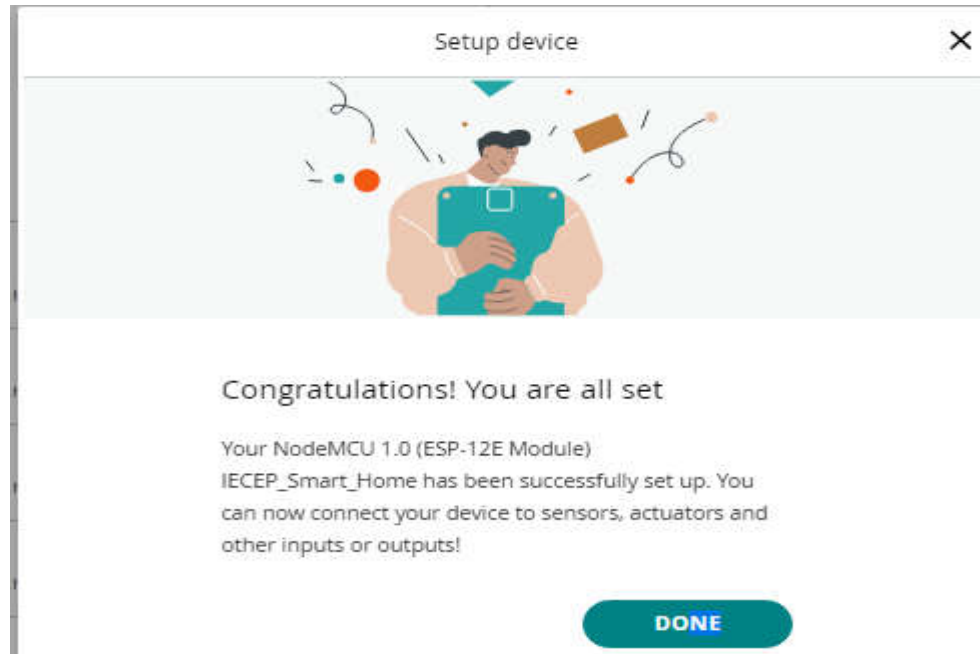
Device ID
00f47b4e-610a-42c3-a6ce-55b3f82ef974

Secret Key
SPG3L7SJYR6IR89VVEQE

**Secret key cannot be recovered**
Please keep it safe, if you lose it you will have to delete and setup your device again.

☒ I saved my device ID and Secret Key **CONTINUE**

IoT using ESP8266 WiFi Module

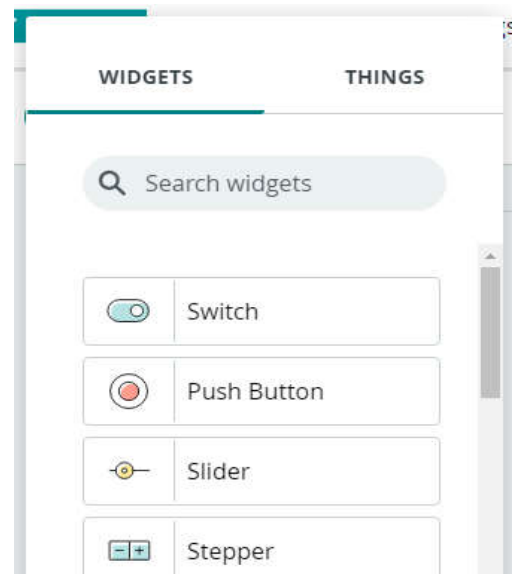
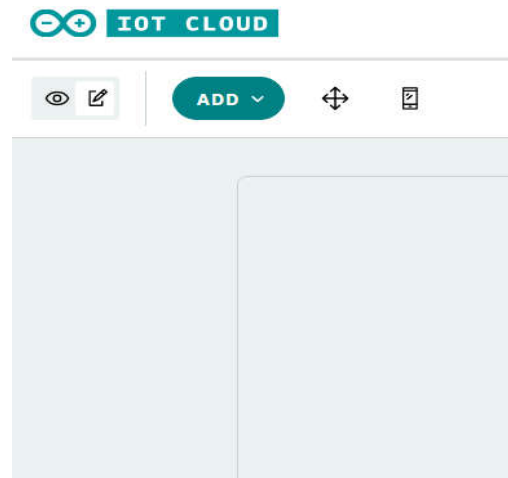


IoT using ESP8266 WiFi Module

The screenshot displays the IOT CLOUD interface. At the top, there's a navigation bar with 'Things', 'Dashboards' (highlighted with a red box), 'Devices', 'Integrations', and 'Templates'. A 'UPGRADE PLAN' button is on the right. Below the navigation bar, there are two tabs: 'Setup' and 'Sketch'. The 'Setup' tab is active, showing a 'Variables' section with an 'ADD' button and a table of variables. The 'Sketch' tab shows 'Change' and 'Detach' buttons. Below the tabs, there's a 'Network' section with fields for 'Wi-Fi Name', 'Password', and 'Secret Key', and a 'Change' button. At the bottom, there's a 'Dashboards' section with a 'BUILD DASHBOARD' button (highlighted with a red box) and a dropdown menu for 'All dashboards'. Below this, there's a table of dashboards.

Name ↓	Last modified	Owned by
<input type="checkbox"/> Home Automation	16 Feb 2022 13:59:24	spectrum17

IoT using ESP8266 WiFi Module



IoT using ESP8266 WiFi Module

The image shows two screenshots from an IoT dashboard interface.

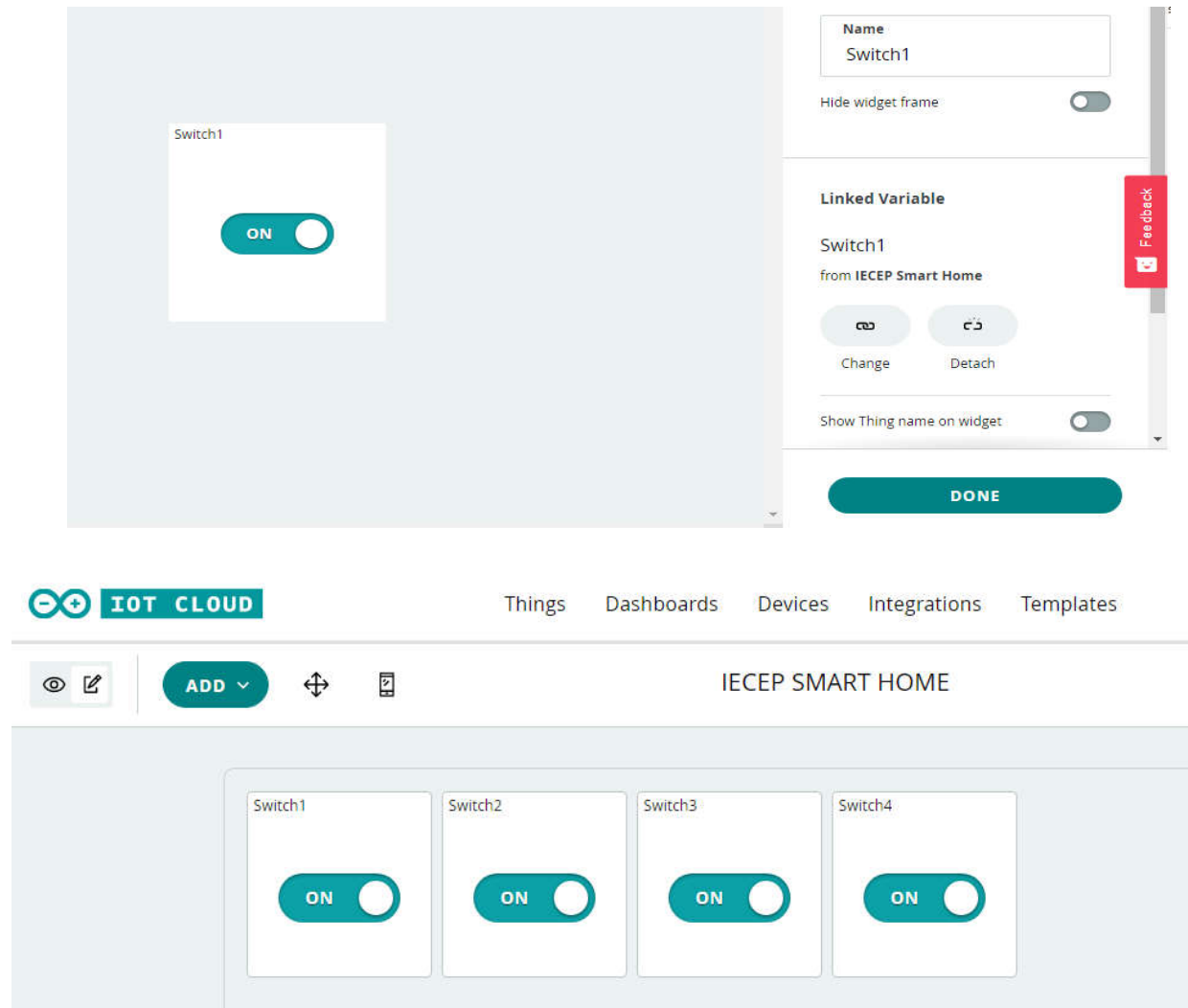
Top Screenshot: Widget Settings

- Widget Settings:**
 - Name: Switch1
 - Hide widget frame: ☐
- Linked Variable:**
 - This widget is displaying example data. Select a source variable to display its value.
 - A red box highlights a teal button with a link icon.

Bottom Screenshot: Link Variable to Switch1

- Things:**
 - Home Automation
 - IECEP Smart Home (selected)
- Variables:**
 - Switch1 (selected)
 - Switch2
 - Switch3
 - Switch4
- Switch1 Details:**
 - Thing: IECEP Smart Home
 - Type: Switch
 - Last value: -
 - Permission: Read/Write
 - Update policy: On change
 - Last update: null
- A red box highlights a teal button labeled **LINK VARIABLE**.

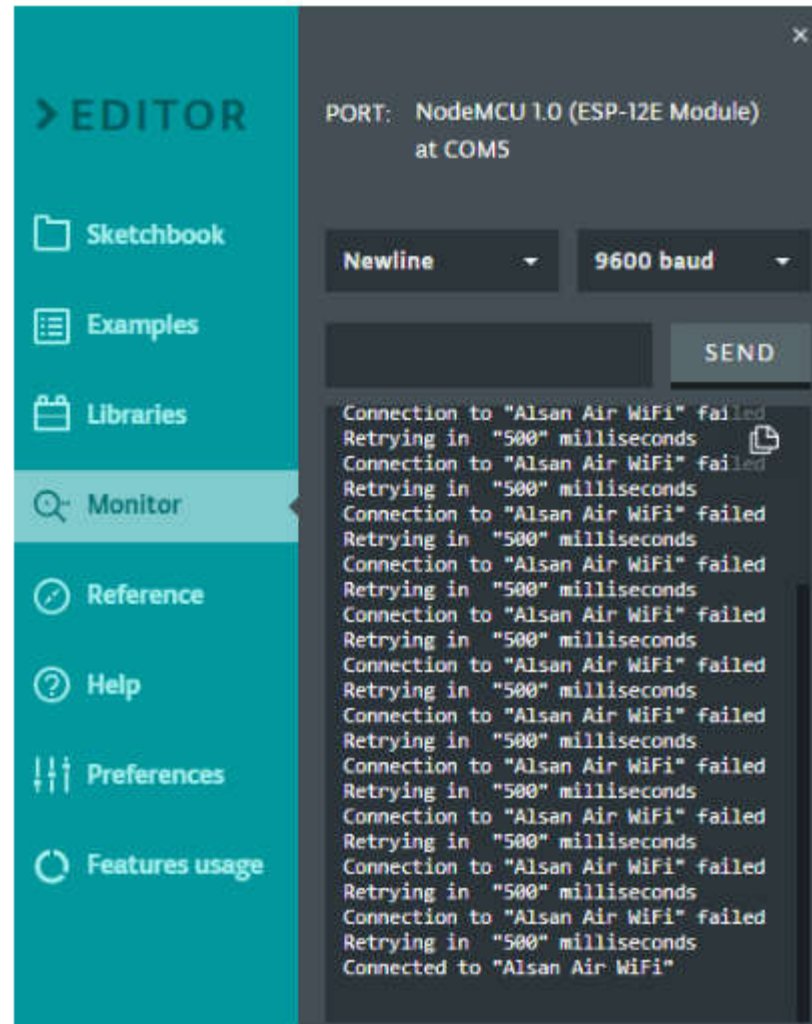
IoT using ESP8266 WiFi Module



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

28

```
//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);
```

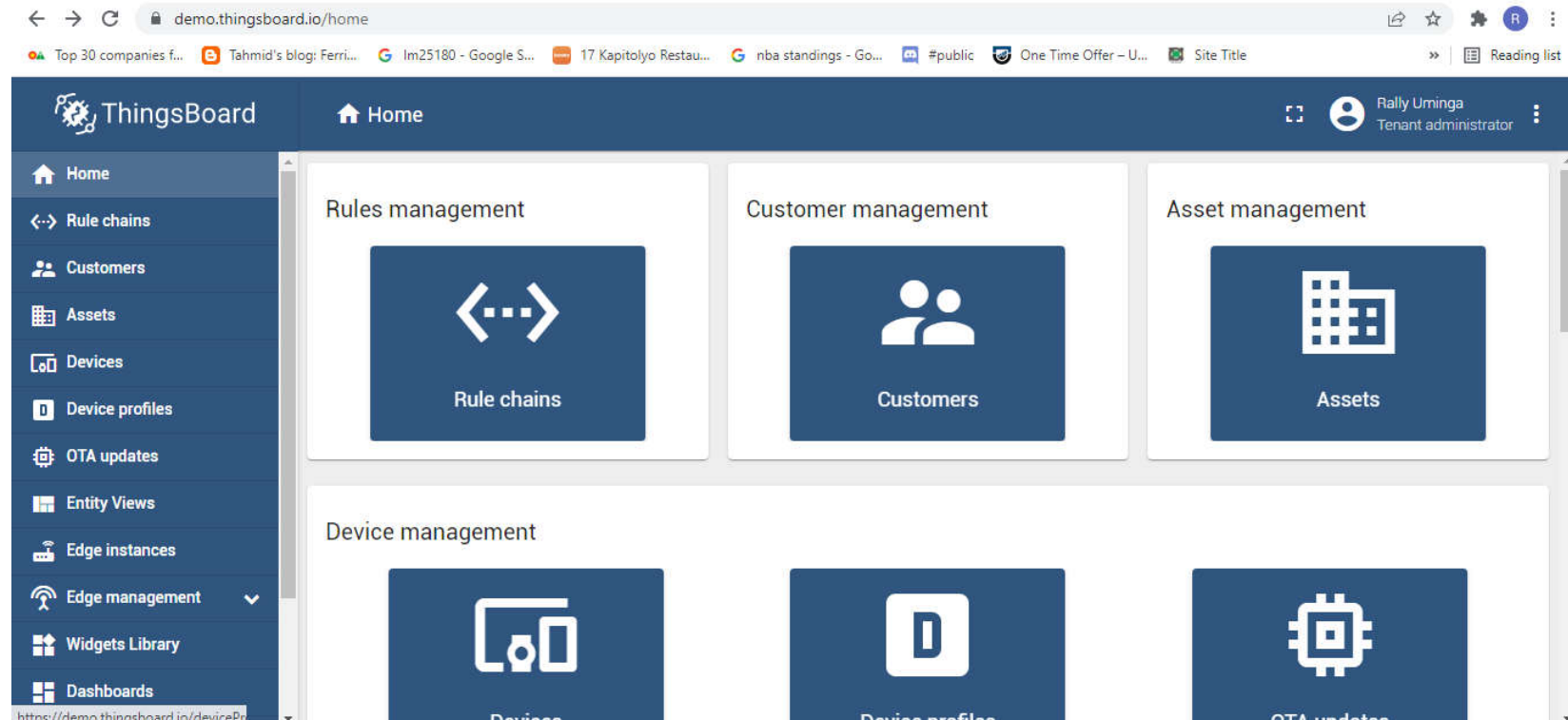
```
}
```

```
WiFiConnectionHandler ArduinoIoTPreferredConnection(SSID, PASS);
```



IoT using ESP8266 WiFi Module

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

39

demo.thingsboard.io/devices

ThingsBoard

Devices





















Rally Uminga
Tenant administrator

Home
Rule chains
Customers
Assets
Devices
Device profiles
OTA updates
Entity Views
Edge instances

Devices

Device profile: All

+ ↺ 🔍

<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway	
<input type="checkbox"/>	2022-03-18 20:36:26	bme280	default	weather_sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-03-15 00:31:37	BH1750	default	Lux Meter	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-03-14 23:32:22	DHT11 Sensor	default	DHT11 Sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	    
<input type="checkbox"/>	2022-02-17 16:44:03	ldrsensor	default	LDR SENSOR		<input type="checkbox"/>	<input type="checkbox"/>	    



IoT using ESP8266 WiFi Module

40

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

41

Add new device ? X

1 Device details

2 Credentials
Optional

3 Customer
Optional

Name *

DHT11 TempHum

Label

DHT11 TempHum

☒ Select existing device profile

Device profile *

default X

☐ Create new device profile

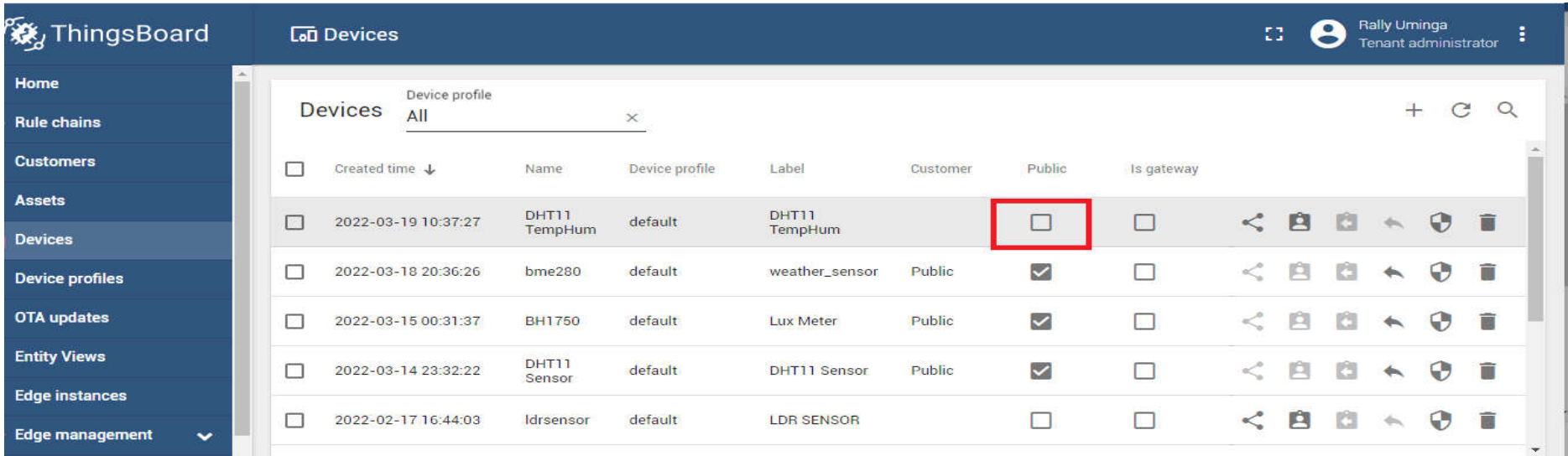
☐ Is gateway

Next: Credentials

Cancel Add

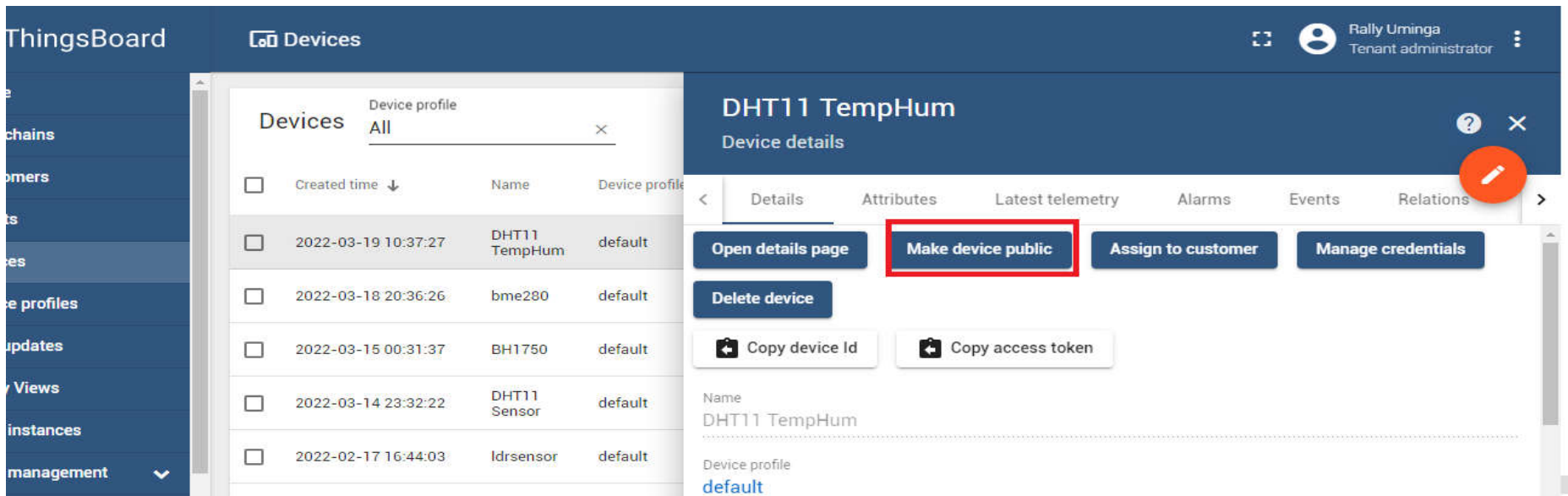
IoT using ESP8266 WiFi Module

42



The screenshot shows the ThingsBoard interface with the 'Devices' tab selected. A table lists several devices. The first device, 'DHT11 TempHum', has its 'Public' checkbox highlighted with a red box. The table columns are: Created time, Name, Device profile, Label, Customer, Public, and Is gateway.

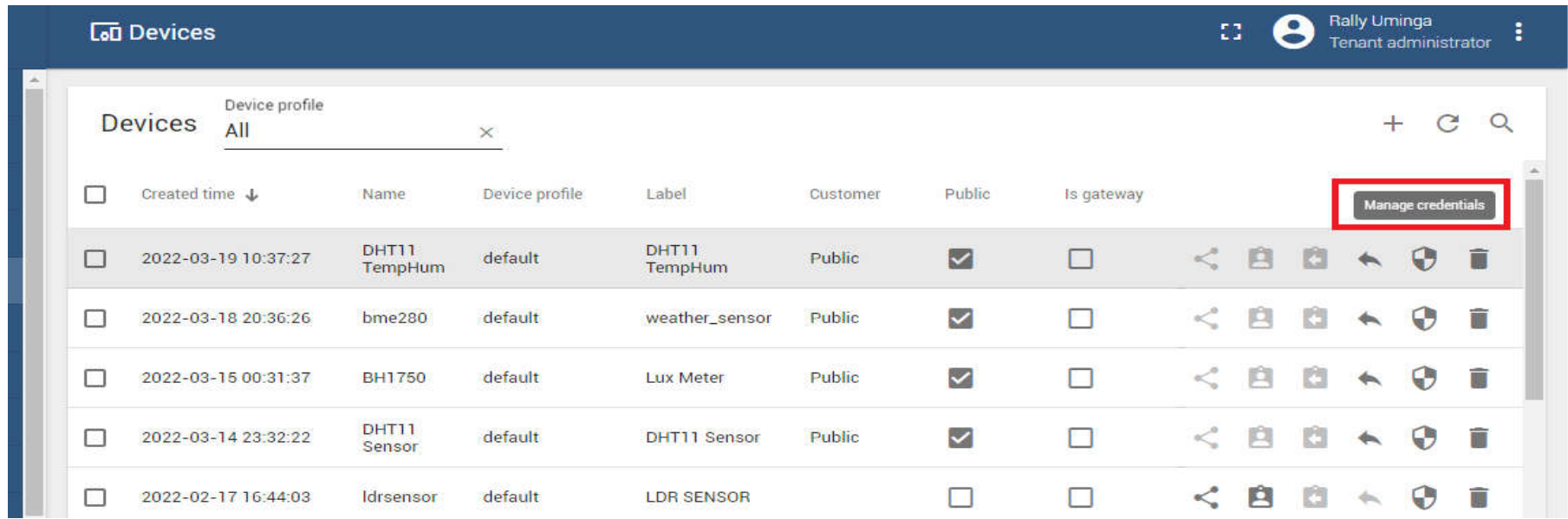
Created time	Name	Device profile	Label	Customer	Public	Is gateway
2022-03-19 10:37:27	DHT11 TempHum	default	DHT11 TempHum		<input type="checkbox"/>	<input type="checkbox"/>
2022-03-18 20:36:26	bme280	default	weather_sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-03-15 00:31:37	BH1750	default	Lux Meter	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-03-14 23:32:22	DHT11 Sensor	default	DHT11 Sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-02-17 16:44:03	ldrsensor	default	LDR SENSOR		<input type="checkbox"/>	<input type="checkbox"/>



The screenshot shows the 'DHT11 TempHum' device details page. The 'Make device public' button is highlighted with a red box. The page includes tabs for Details, Attributes, Latest telemetry, Alarms, Events, and Relations. Below the tabs are buttons for 'Open details page', 'Make device public', 'Assign to customer', 'Manage credentials', and 'Delete device'. There are also buttons for 'Copy device Id' and 'Copy access token'.

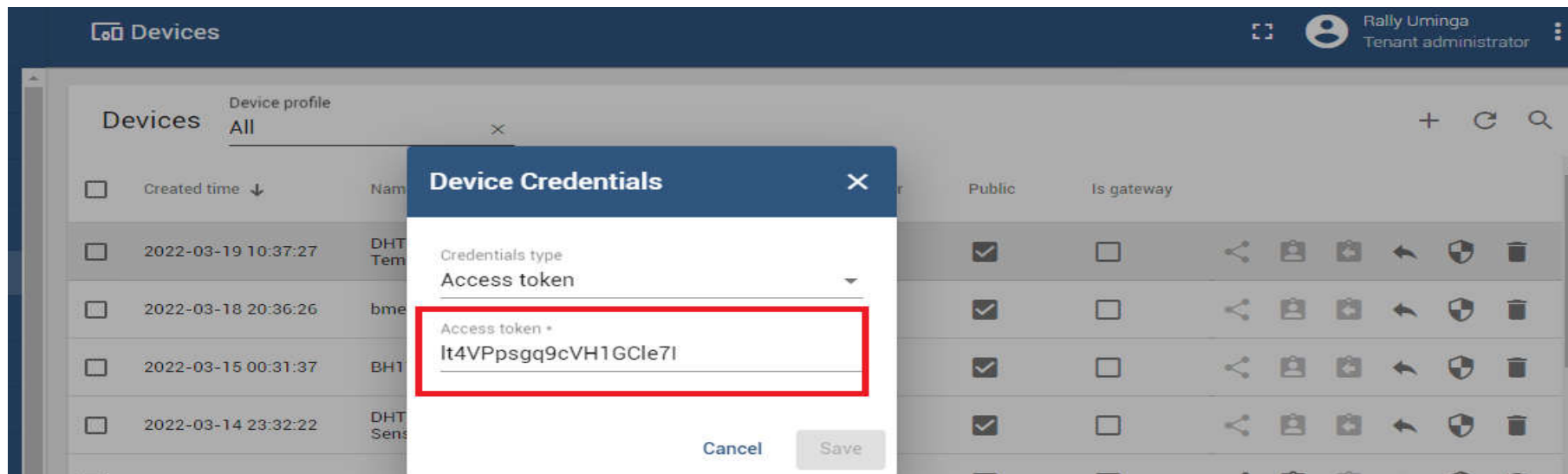
IoT using ESP8266 WiFi Module

43



The screenshot shows the 'Devices' management page. At the top right, the user 'Rally Uminga' is identified as a 'Tenant administrator'. Below the header, there's a 'Devices' section with a filter for 'All'. A table lists several devices with columns for 'Created time', 'Name', 'Device profile', 'Label', 'Customer', 'Public', and 'Is gateway'. A red box highlights the 'Manage credentials' button in the top right corner of the device list.

Created time	Name	Device profile	Label	Customer	Public	Is gateway
2022-03-19 10:37:27	DHT11 TempHum	default	DHT11 TempHum	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-03-18 20:36:26	bme280	default	weather_sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-03-15 00:31:37	BH1750	default	Lux Meter	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-03-14 23:32:22	DHT11 Sensor	default	DHT11 Sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2022-02-17 16:44:03	ldrsensor	default	LDR SENSOR		<input type="checkbox"/>	<input type="checkbox"/>



The screenshot shows the 'Device Credentials' dialog box. The 'Credentials type' is set to 'Access token'. The 'Access token' field is highlighted with a red box and contains the token 'lt4VPpsgq9cVH1GCle7l'. The dialog has 'Cancel' and 'Save' buttons at the bottom.

Device Credentials

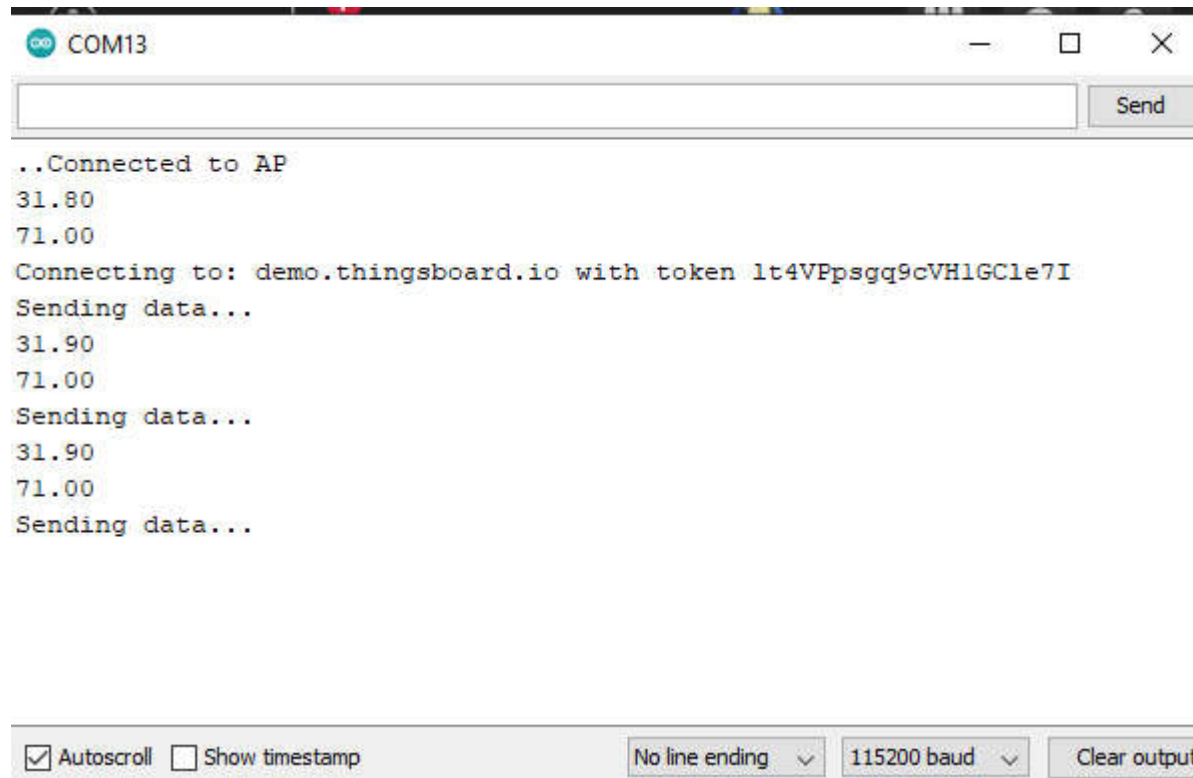
Credentials type: Access token

Access token: lt4VPpsgq9cVH1GCle7l

Cancel Save

IoT using ESP8266 WiFi Module

44



```
COM13
Send

..Connected to AP
31.80
71.00
Connecting to: demo.thingsboard.io with token lt4VPsgq9cVH1GC1e7I
Sending data...
31.90
71.00
Sending data...
31.90
71.00
Sending data...
```

☒ Autoscroll ☐ Show timestamp No line ending 115200 baud Clear output



IoT using ESP8266 WiFi Module

45

The screenshot displays a web application interface for managing IoT devices. On the left, a sidebar titled 'Devices' shows a list of devices with columns for 'Created time', 'Name', and 'Device profile'. The main content area is titled 'DHT11 TempHum' and 'Device details'. It features a tabbed interface with 'Latest telemetry' selected. The 'Latest telemetry' tab shows a table of data points, with the first two rows highlighted by a red box. The first row shows a humidity reading of 71, and the second row shows a temperature reading of 31.89999962. The interface also includes a search bar, a pagination control showing 'Items per page: 10' and '1 - 2 of 2', and a red circular button with a pencil icon for editing.

Last update time	Key ↑	Value
2022-03-19 10:56:23	humidty	71
2022-03-19 10:56:22	temperature	31.89999962



IoT using ESP8266 WiFi Module

46

ThingsBoard

Devices

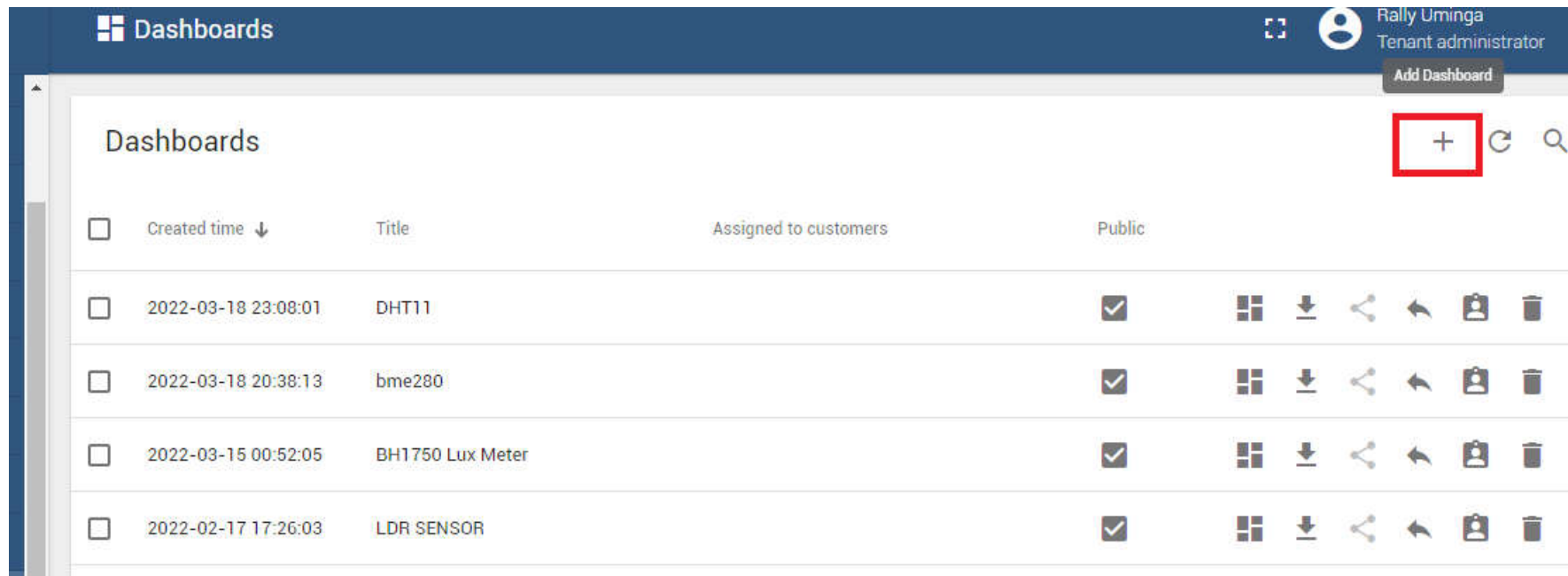
Device profile: All

<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway
<input type="checkbox"/>	2022-03-19 10:37:27	DHT11 TempHum	default	DHT11 TempHum	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	2022-03-18 20:36:26	bme280	default	weather_sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	2022-03-15 00:31:37	BH1750	default	Lux Meter	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	2022-03-14 23:32:22	DHT11 Sensor	default	DHT11 Sensor	Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	2022-02-17 16:44:03	ldrsensor	default	LDR SENSOR		<input type="checkbox"/>	<input type="checkbox"/>



























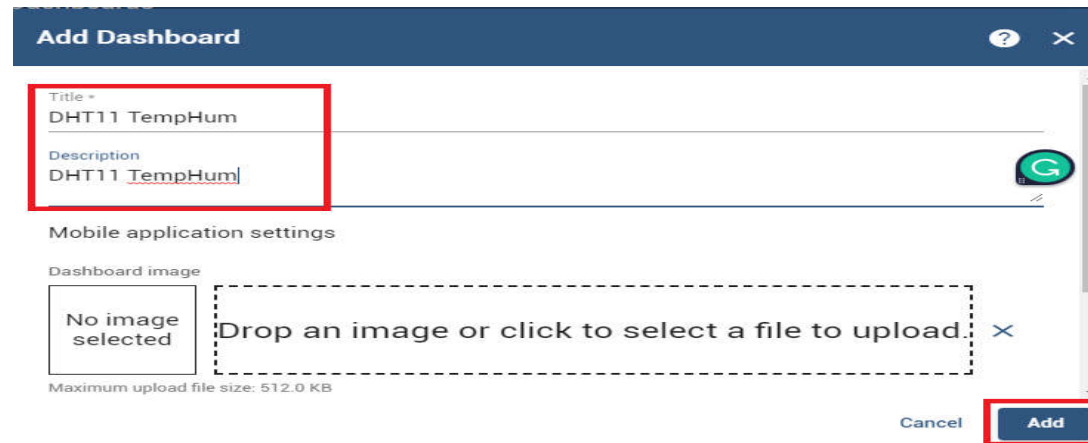
IoT using ESP8266 WiFi Module

47



The screenshot shows a web interface for managing dashboards. At the top, there's a header with a 'Dashboards' menu icon, a user profile for 'Rally Uminga' (Tenant administrator), and an 'Add Dashboard' button. Below the header, a red box highlights a '+' icon next to a refresh and search icon. The main area contains a table of existing dashboards.

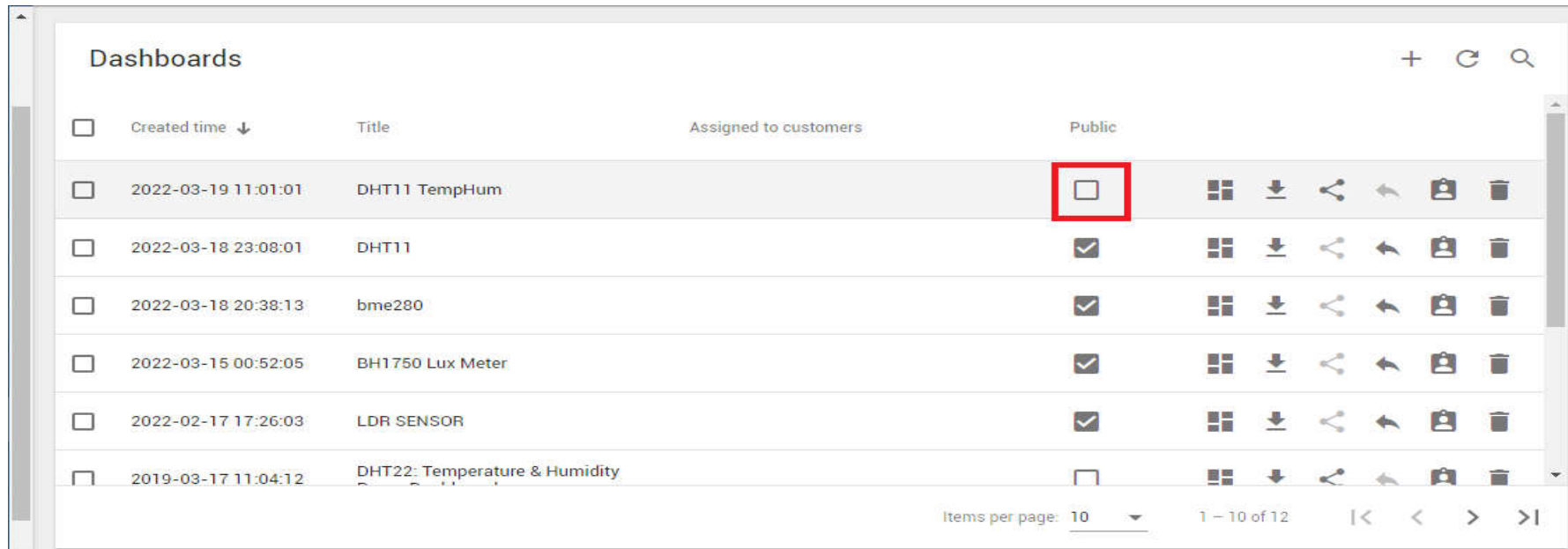
<input type="checkbox"/>	Created time ↓	Title	Assigned to customers	Public	
<input type="checkbox"/>	2022-03-18 23:08:01	DHT11		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2022-03-18 20:38:13	bme280		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2022-03-15 00:52:05	BH1750 Lux Meter		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2022-02-17 17:26:03	LDR SENSOR		<input checked="" type="checkbox"/>	     







































The screenshot shows a modal window titled 'Add Dashboard'. It contains input fields for 'Title' and 'Description', both containing the text 'DHT11 TempHum'. Below these is a section for 'Mobile application settings'. Under 'Dashboard image', there is a placeholder box with the text 'No image selected' and a larger dashed box with the text 'Drop an image or click to select a file to upload.'. At the bottom right, there are 'Cancel' and 'Add' buttons, with the 'Add' button highlighted by a red box.

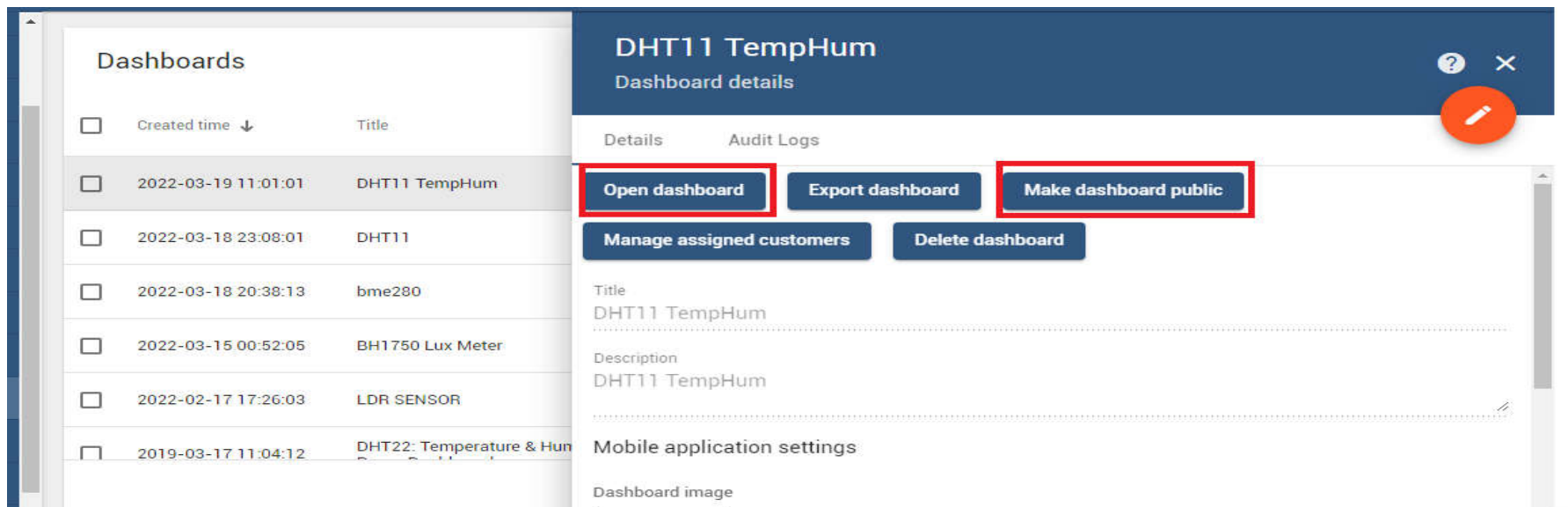
IoT using ESP8266 WiFi Module

48



<input type="checkbox"/>	Created time ↓	Title	Assigned to customers	Public	
<input type="checkbox"/>	2022-03-19 11:01:01	DHT11 TempHum		<input type="checkbox"/>	     
<input type="checkbox"/>	2022-03-18 23:08:01	DHT11		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2022-03-18 20:38:13	bme280		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2022-03-15 00:52:05	BH1750 Lux Meter		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2022-02-17 17:26:03	LDR SENSOR		<input checked="" type="checkbox"/>	     
<input type="checkbox"/>	2019-03-17 11:04:12	DHT22: Temperature & Humidity		<input type="checkbox"/>	     

Items per page: 10 1 – 10 of 12



DHT11 TempHum

Dashboard details

Details Audit Logs

Open dashboard **Export dashboard** **Make dashboard public**

Manage assigned customers **Delete dashboard**

Title
DHT11 TempHum

Description
DHT11 TempHum

Mobile application settings

Dashboard image

IoT using ESP8266 WiFi Module

49

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](#)



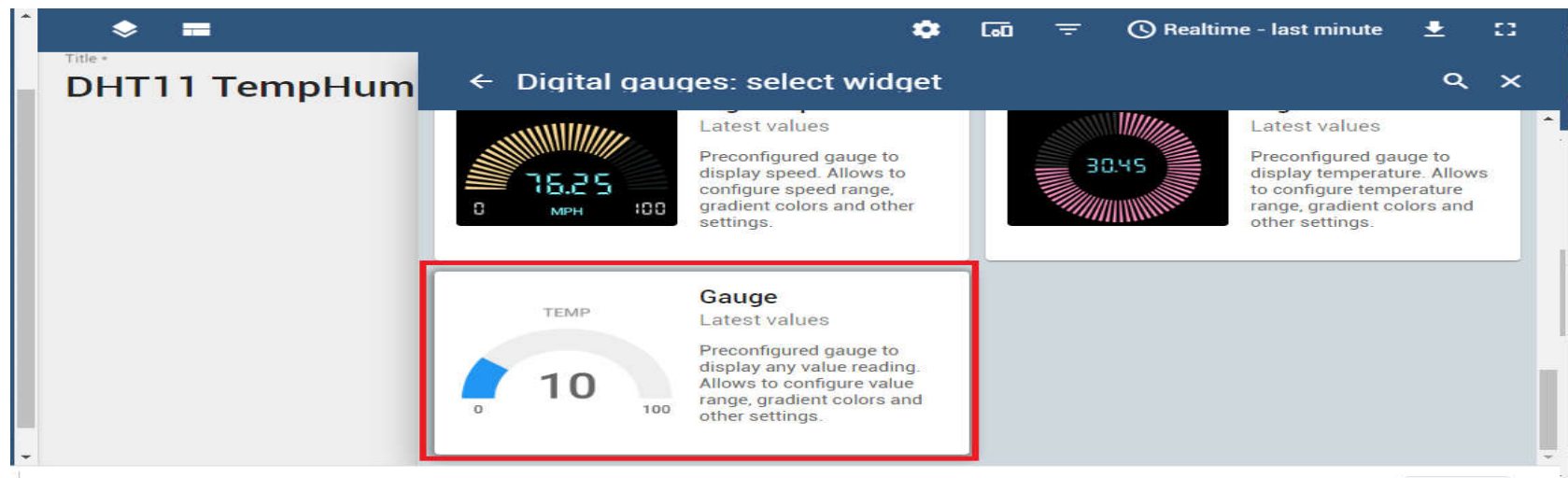
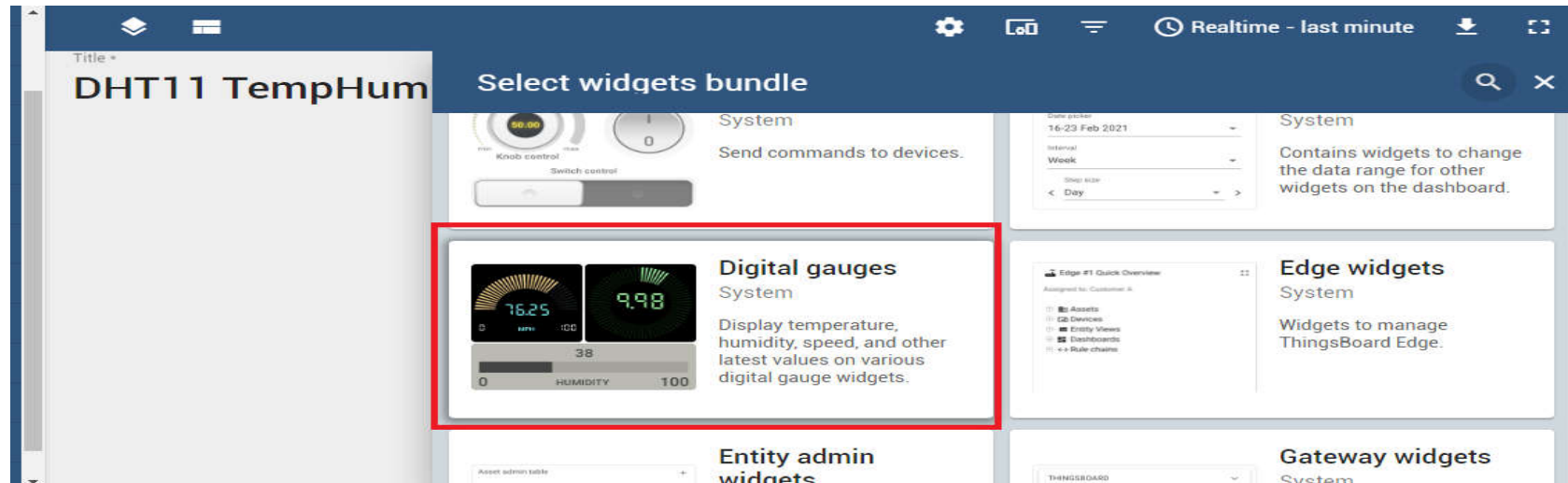
IoT using ESP8266 WiFi Module

50



IoT using ESP8266 WiFi Module

51



IoT using ESP8266 WiFi Module

52

Add Widget: Gauge ? X

Data Settings Advanced Actions

Datasources
Maximum 1 datasource is allowed.

Add datasource

Please add datasource

+ Add

Data settings

Cancel Add

Add Widget: Gauge ? X

Data Settings Advanced Actions

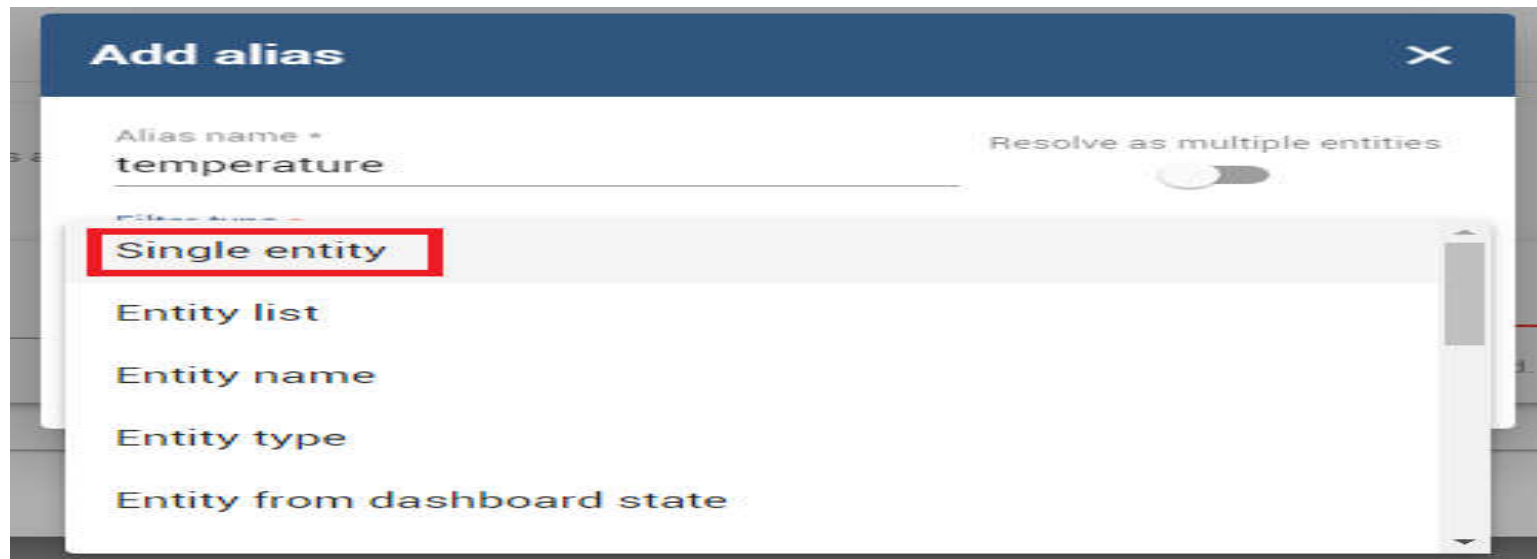
Datasources
Maximum 1 datasource is allowed.

Type	Parameters
= 1. Entity	<div>Entity alias * temperature</div> <div>'temper...' not found. Create a new one!</div>

Entity timeseries/attributes are required.
Maximum 1 timeseries/attribute is allowed.

IoT using ESP8266 WiFi Module

53



Add alias [X]

Alias name *
temperature

Resolve as multiple entities [Toggle]

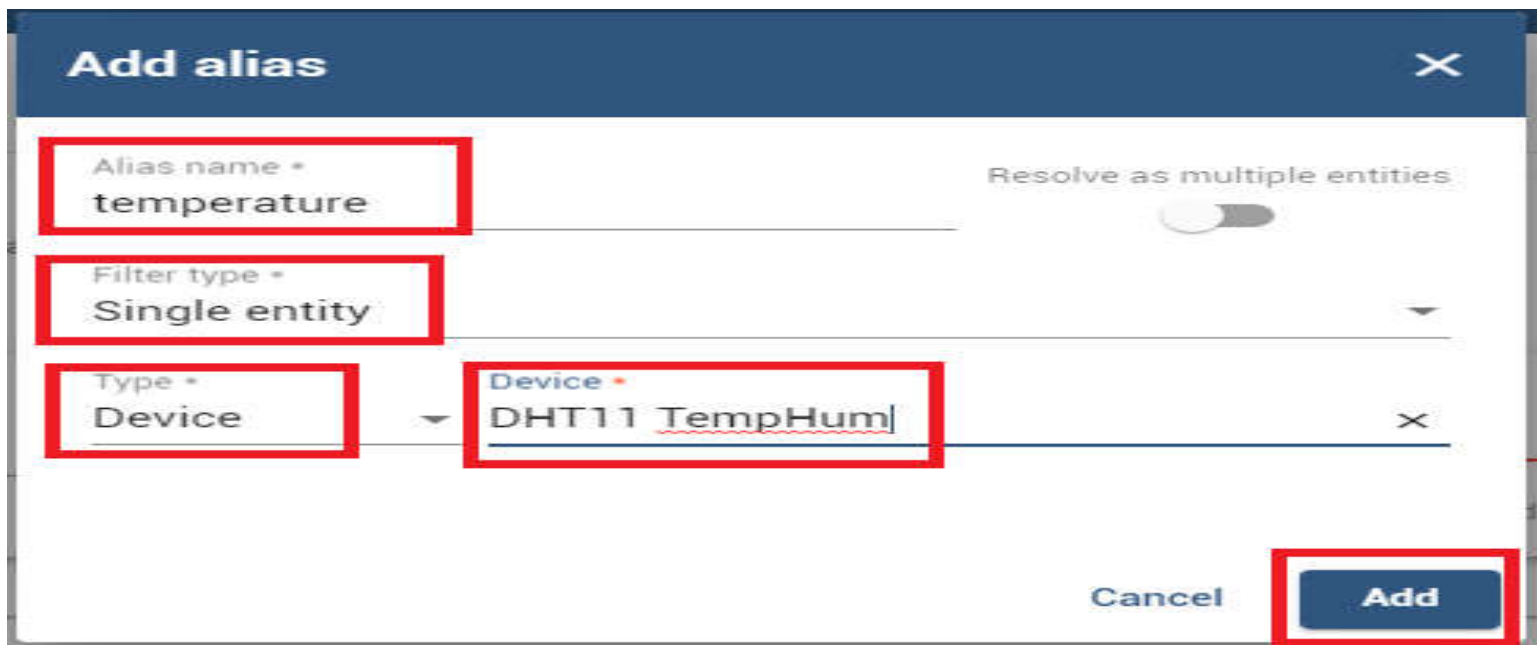
Filter type *
Single entity

Entity list

Entity name

Entity type

Entity from dashboard state



Add alias [X]

Alias name *
temperature

Resolve as multiple entities [Toggle]

Filter type *
Single entity

Type *
Device

Device *
DHT11 TempHum

Cancel Add

IoT using ESP8266 WiFi Module

54

Add Widget: Gauge

humidity
temperature
active
inactivityAlarmTime
lastActivityTime
lastConnectTime

Datasources
Maximum 1 datasource is allowed.

Type	Parameters
1. Entity	<div>Entity alias * temperature</div> <div>Filter</div>

Entity timeseries/attributes are required.
Maximum 1 timeseries/attribute is allowed.

Cancel Add

IoT using ESP8266 WiFi Module

55

Add Widget: Gauge

Data

Settings

Advanced

Actions

Datasources

Maximum 1 datasource is allowed.

Type

Parameters

= 1. Entity

Entity alias *

temperature

Filter


=

temper... : tempera...


Cancel

Add

RALLY UMINGA 2022

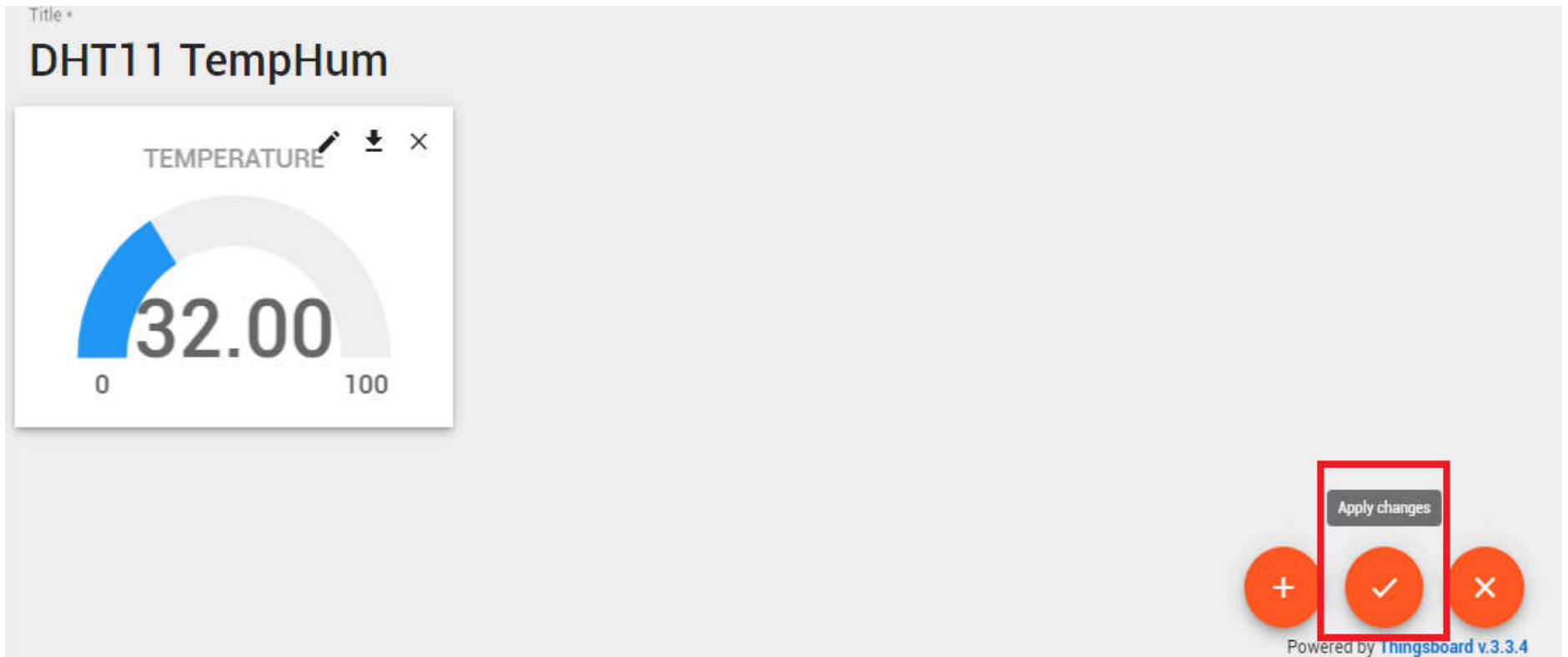


IECEP RIZAL



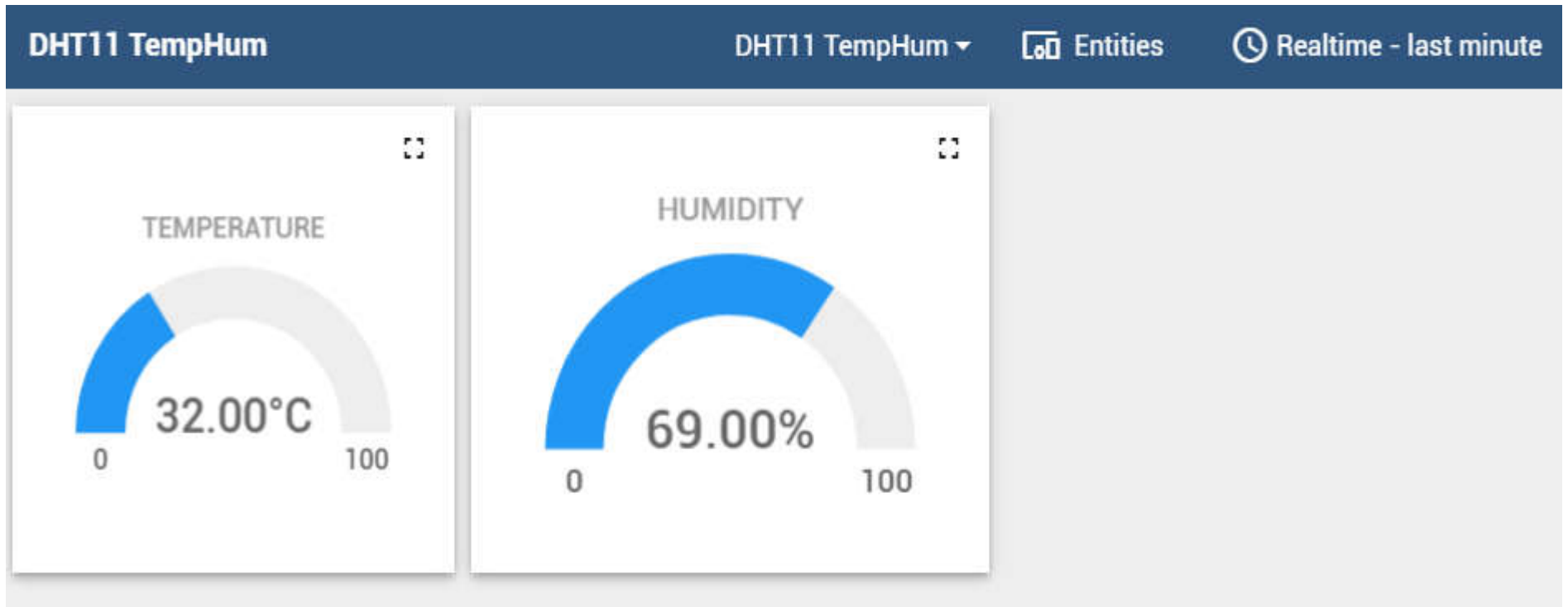
IoT using ESP8266 WiFi Module

56



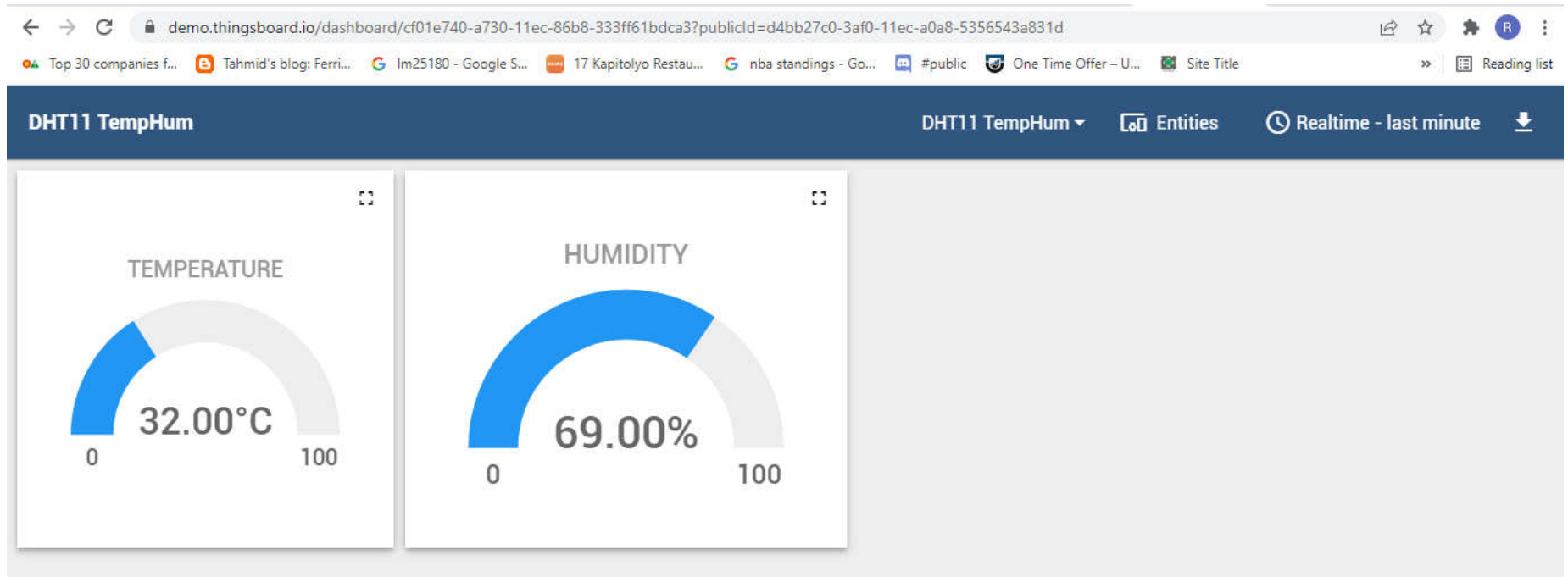
IoT using ESP8266 WiFi Module

57



IoT using ESP8266 WiFi Module

58



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                "w5i8L0likhcpioanKx2g"
#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("humidity", 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

63

```
//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

64

```
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

65

```
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

66

```
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

