BÁO CÁO THỰC HÀNH

**Bài thực hành số 05: Giải pháp IoT hướng đến bảo mật**

**Môn học:** Bảo mật Internet of Things

**Lớp:** NT535.P21.1

**THÀNH VIÊN THỰC HIỆN (Nhóm xx):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |  | **Điểm tự đánh giá** |
| 1 | Phạm Thiều Gia Khang | 21520967 |  |
| 2 | Phan Văn B | 20520002 |
|  |  |  |  |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện |  |
| Phân chia công việc |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị |  |

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

MỤC LỤC

[A. BÁO CÁO CHI TIẾT 2](#_Toc177398261)

[1. Nội dung 1 2](#_Toc177398262)

[a. Nội dung a 2](#_Toc177398263)

[b. Nội dung b 3](#_Toc177398264)

[2. Nội dung 2 3](#_Toc177398265)

[a. Nội dung a 3](#_Toc177398266)

[B. TÀI LIỆU THAM KHẢO 3](#_Toc177398267)

# BÁO CÁO CHI TIẾT

## Triển khai ThingsBoard (https://thingsboard.io) để thực hiện giám sát, xử lý dữ liệu, trực quan hoá dữ liệu và quản lý thiết bị. Tham khảo hướng dẫn cài đặt tại https://thingsboard.io/docs/user-guide/install/ubuntu/

### Cài đặt Thingsboard:

Cài đặt dựa theo hướng dẫn của nhà phát triển tại: <https://thingsboard.io/docs/user-guide/install/ubuntu/?ubuntuThingsboardQueue=kafka-in-docker>.

Chọn tùy chọn là cài Thingsboard trên Docker + Kafka:

Tạo image Kafka bằng file docker-compose.yaml:

version: '3.2'

services:

kafka:

restart: always

image: bitnami/kafka:3.8.1

ports:

- 9092:9092 #to localhost:9092 from host machine

- 9093 #for Kraft

- 9094 #to kafka:9094 from within Docker network

environment:

ALLOW\_PLAINTEXT\_LISTENER: "yes"

KAFKA\_CFG\_LISTENERS: "OUTSIDE://:9092,CONTROLLER://:9093,INSIDE://:9094"

KAFKA\_CFG\_ADVERTISED\_LISTENERS: "OUTSIDE://localhost:9092,INSIDE://kafka:9094"

KAFKA\_CFG\_LISTENER\_SECURITY\_PROTOCOL\_MAP: "INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT,CONTROLLER:PLAINTEXT"

KAFKA\_CFG\_INTER\_BROKER\_LISTENER\_NAME: "INSIDE"

KAFKA\_CFG\_AUTO\_CREATE\_TOPICS\_ENABLE: "false"

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: "1"

KAFKA\_TRANSACTION\_STATE\_LOG\_MIN\_ISR: "1"

KAFKA\_TRANSACTION\_STATE\_LOG\_REPLICATION\_FACTOR: "1"

KAFKA\_CFG\_PROCESS\_ROLES: "controller,broker" #KRaft

KAFKA\_CFG\_NODE\_ID: "0" #KRaft

KAFKA\_CFG\_CONTROLLER\_LISTENER\_NAMES: "CONTROLLER" #KRaft

KAFKA\_CFG\_CONTROLLER\_QUORUM\_VOTERS: "0@kafka:9093" #KRaft

volumes:

- kafka-data:/bitnami

volumes:

kafka-data:

driver: local

Điều chỉnh file conf của Thingsboard:

#

# Copyright © 2016-2024 The Thingsboard Authors

#

# Licensed under the Apache License, Version 2.0 (the "License");

# you may not use this file except in compliance with the License.

# You may obtain a copy of the License at

#

# http://www.apache.org/licenses/LICENSE-2.0

#

# Unless required by applicable law or agreed to in writing, software

# distributed under the License is distributed on an "AS IS" BASIS,

# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

# See the License for the specific language governing permissions and

# limitations under the License.

#

# DB Configuration

export DATABASE\_TS\_TYPE=sql

export SPRING\_DATASOURCE\_URL=jdbc:postgresql://localhost:5432/thingsboard

export SPRING\_DATASOURCE\_USERNAME=postgres

export SPRING\_DATASOURCE\_PASSWORD=1

# Specify partitioning size for timestamp key-value storage. Allowed values: DAYS, MONTHS, YEARS, INDEFINITE.

export SQL\_POSTGRES\_TS\_KV\_PARTITIONING=MONTHS

export TB\_QUEUE\_TYPE=kafka

export TB\_KAFKA\_SERVERS=localhost:9092

# Cho phép Thingsboard lắng nghe Ipv4

export MQTT\_BIND\_ADDRESS=0.0.0.0

export JAVA\_OPTS="$JAVA\_OPTS -Dplatform=deb -Dinstall.data\_dir=/usr/share/thingsboard/data"

export JAVA\_OPTS="$JAVA\_OPTS -Xlog:gc\*,heap\*,age\*,safepoint=debug:file=/var/log/thingsboard/gc.log:time,uptime,level,tags:filecount=10,filesize=10M"

export JAVA\_OPTS="$JAVA\_OPTS -XX:+IgnoreUnrecognizedVMOptions -XX:+HeapDumpOnOutOfMemoryError"

export JAVA\_OPTS="$JAVA\_OPTS -XX:-UseBiasedLocking -XX:+UseTLAB -XX:+ResizeTLAB -XX:+PerfDisableSharedMem -XX:+UseCondCardMark"

export JAVA\_OPTS="$JAVA\_OPTS -XX:+UseG1GC -XX:MaxGCPauseMillis=500 -XX:+UseStringDeduplication -XX:+ParallelRefProcEnabled -XX:MaxTenuringThreshold=10"

export LOG\_FILENAME=thingsboard.out

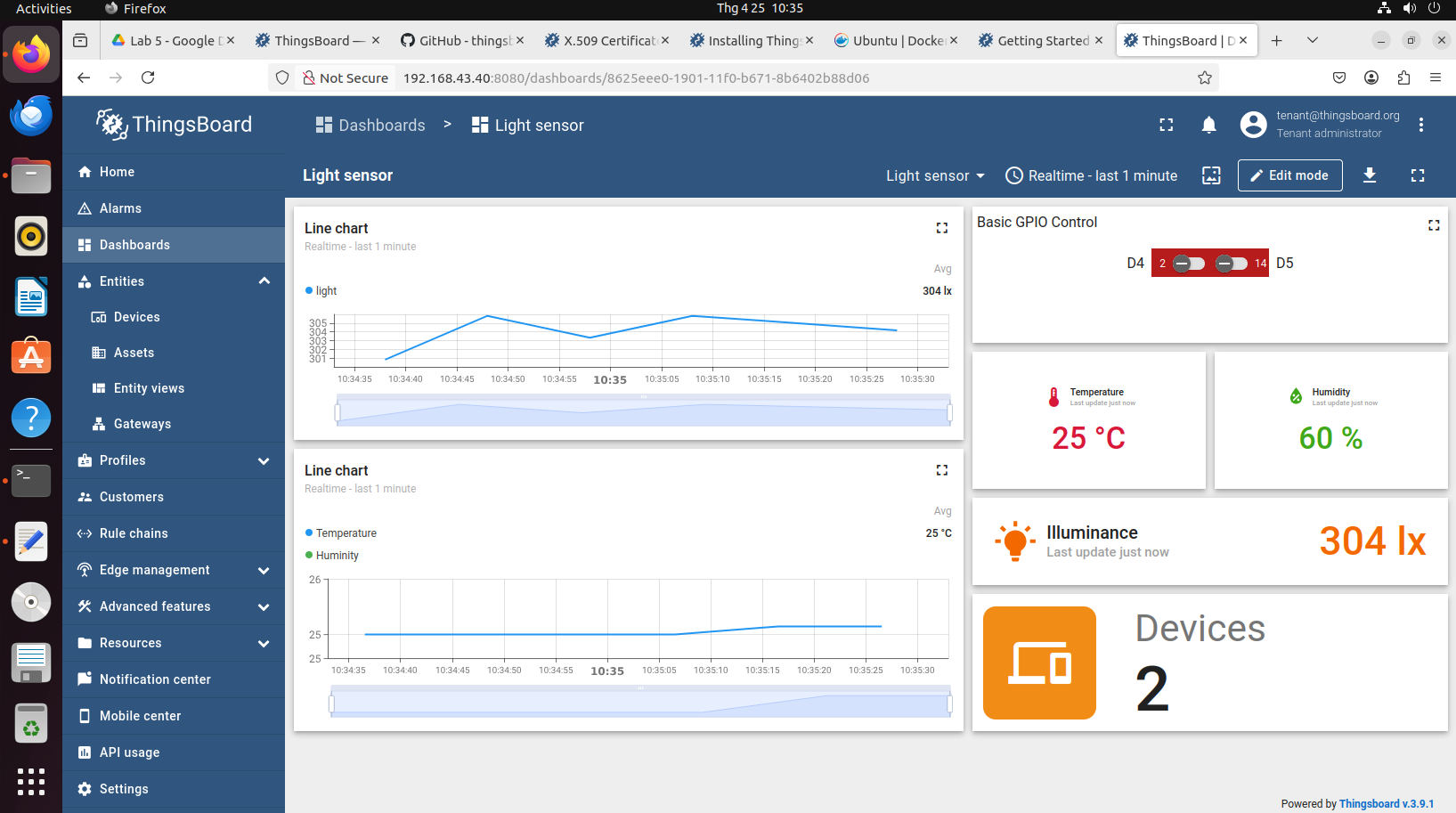
export LOADER\_PATH=/usr/share/thingsboard/conf,/usr/share/thingsboard/extensions

export SQL\_DATA\_FOLDER=/usr/share/thingsboard/data/sql

Kích hoạt Thingsboard và truy cập {HOST-IP}:8080/ để mở trang đăng nhập của Thingsboard:

$ sudo systemctl start thingsboard

Đăng nhập vào bằng tài khoản: Tenant Administrator: tenant@thingsboard.org / tenant

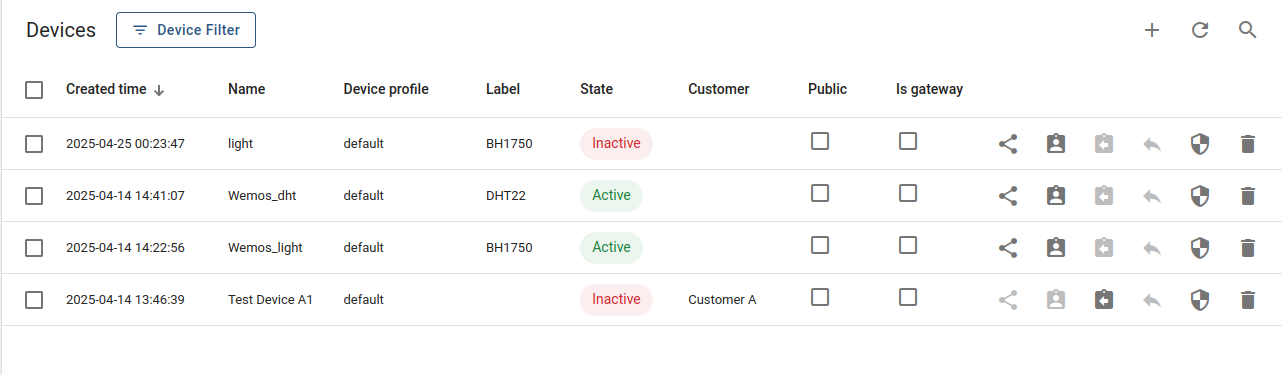


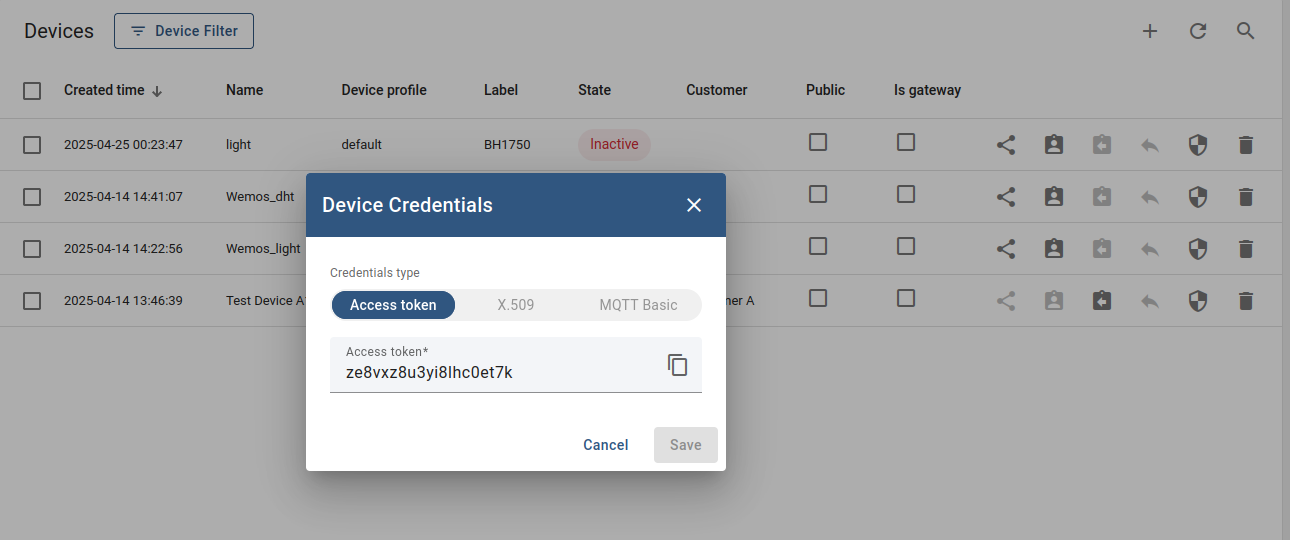
## Xây dựng một giải pháp IoT (có thể sử dụng thiết bị thật hoặc ảo hoá) bao gồm ít nhất 02 thiết bị có khả năng tạo dữ liệu (cảm biến, đọc thẻ RFID,…) và các thiết bị như đèn, LCD, âm thanh, servo,… Sử dụng ThingsBoard để xử lý dữ liệu thu thập được và ra tín hiệu điều khiển thiết bị / cảm biến.

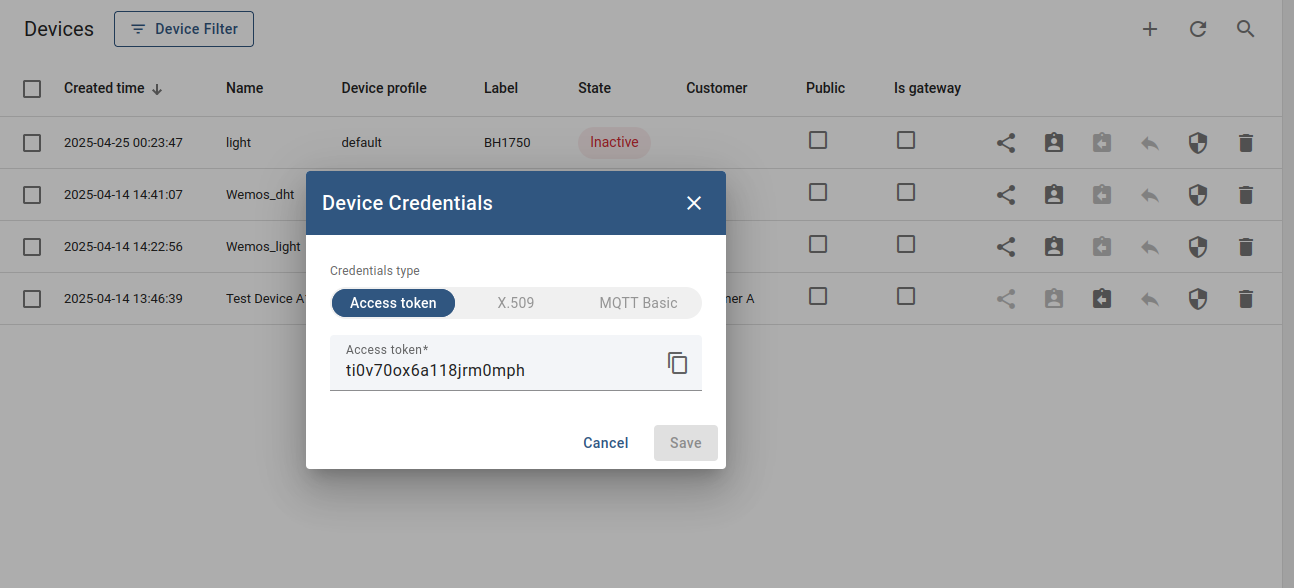
### Xây dựng kịch bản:

Kịch bản sẽ bao gồm 2 Wemos và 1 Host chạy Thingsboard, Node 1 sẽ đo cường độ ánh sang và nhận tín hiệu điều khiển đèn từ Platform, Node 2 sẽ đo nhiệt độ độ ẩm.

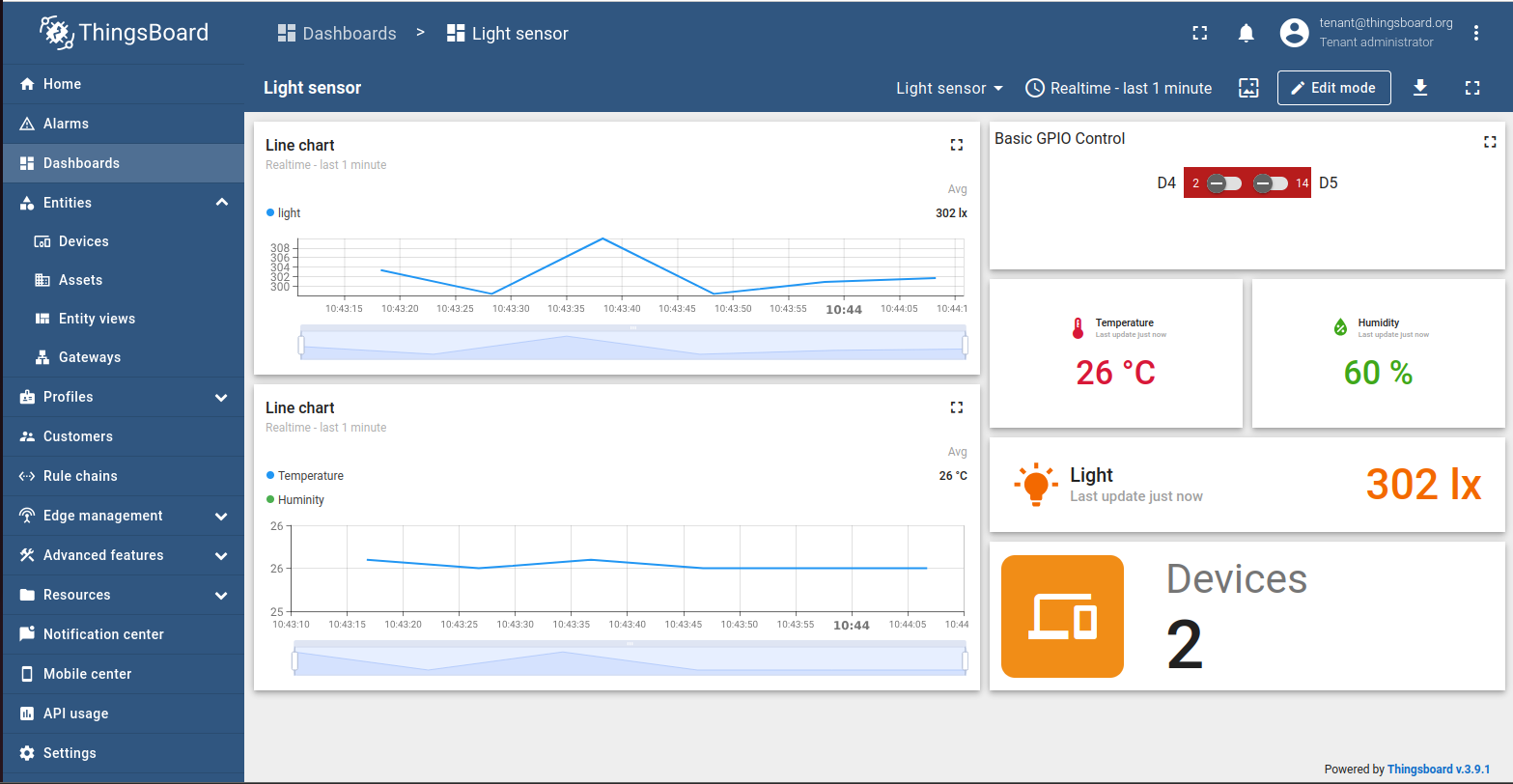
Tạo 1 2 Device Wemos\_light và Wemos\_dht trên Thingsboard, sử dụng Access Token được tạo ra để kết nối 2 thiết bị tới Platform:







Tạo một Dashboard gồm các thành phần như Line chart, Tempure card, Humidity card, Basic GPIO Control, Indoor horizontal illuminance card và total device:



Line Chart và các card tương ứng sẽ được sử dụng để hiển thị giá trị thực tế và theo thời gian, Basic GPIO dùng để điều khiển đèn.

### Code:

Trên Wemos\_dht:

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include "DHT.h"

#include <ArduinoJson.h>

// Địa chỉ truy cập Thingsboard

const char\* ssid = "WemosTest";

const char\* password = "01072003";

const char\* thingsboardServer = "192.168.43.40";

const int thingsboardPort = 1883;

const char\* accessToken = "ze8vxz8u3yi8lhc0et7k";

#define DHTPIN D5

#define DHTTYPE DHT22

WiFiClient wifiClient;

PubSubClient client(wifiClient);

DHT dht(DHTPIN, DHTTYPE);

void setup() {

Serial.begin(9600);

dht.begin();

WiFi.begin(ssid, password);

Serial.println("");

while (WiFi.status() != WL\_CONNECTED) {

delay(2000);

Serial.println("Connecting to WiFi...");

}

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

client.setServer(thingsboardServer, thingsboardPort);

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

// Đọc giá trị cảm biến

float h = dht.readHumidity();

float t = dht.readTemperature();

// In giá trị cảm biến ra Serial

Serial.print(F("Humidity: "));

Serial.print(h);

Serial.print(F("%, Temperature: "));

Serial.print(t);

Serial.println(F("°C."));

// Tạo gói Json chứa các giá trị

JsonDocument doc;

doc["temperature"] = t;

doc["humidity"] = h;

serializeJson(doc, Serial);

Serial.println();

// ĐÓng gói và publish gói json

String payload = "";

serializeJson(doc, payload);

client.publish("v1/devices/me/telemetry", payload.c\_str());

delay(10000);

}

void reconnect() {

while (!client.connected()) {

Serial.println("Connecting to ThingsBoard...");

if (client.connect("Wemos\_dht", accessToken, nullptr)) {

Serial.println("Connected to ThingsBoard");\

} else {

Serial.print("Failed to connect. Retrying in 5 seconds...");

delay(5000);

}

}

}

Trên Wemos\_light:

#include <Wire.h>

#include <BH1750.h>

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

// Truy cập vào Thingsboard

const char\* ssid = "WemosTest";

const char\* password = "01072003";

const char\* thingsboardServer = "192.168.43.40";

const int thingsboardPort = 1883;

const char\* accessToken = "ti0v70ox6a118jrm0mph";

#define GPIO2 D4

#define GPIO14 D5

#define GPIO2\_PIN 2

#define GPIO14\_PIN 14

// Điều chỉnh 2 GPIO về LOW

boolean gpioState[] = {false, false};

WiFiClient wifiClient;

PubSubClient client(wifiClient);

BH1750 lightMeter;

void setup() {

pinMode(GPIO2, OUTPUT);

pinMode(GPIO14, OUTPUT);

delay(20);

Serial.begin(9600);

Wire.begin(D2, D1);

lightMeter.begin();

Serial.println(F("BH1750 Test begin"));

WiFi.begin(ssid, password);

Serial.println("");

while (WiFi.status() != WL\_CONNECTED) {

delay(2000);

Serial.println("Connecting to WiFi...");

}

Serial.println("");

Serial.print("Connected to ");

Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

client.setServer(thingsboardServer, thingsboardPort);

client.setCallback(onMessage);

}

void loop() {

if (!client.connected()) {

reconnect();

}

client.loop();

float lux = lightMeter.readLightLevel();

Serial.println(lux);

String payload = "{\"lux\": " + String(lux) + "}";

client.publish("v1/devices/me/telemetry", payload.c\_str());

delay(10000);

}

void reconnect() {

while (!client.connected()) {

Serial.println("Connecting to ThingsBoard...");

if (client.connect("Wemos", accessToken, nullptr)) {

Serial.println("Connected to ThingsBoard");

client.subscribe("v1/devices/me/rpc/request/+");

client.subscribe("v1/devices/me/attributes/response/+");

Serial.println("Sending current GPIO status ...");

client.publish("v1/devices/me/attributes",get\_gpio\_status().c\_str());

} else {

Serial.print("Failed to connect. Retrying in 5 seconds...");

delay(5000);

}

}

}

void onMessage(const char\* topic, byte\* payload, unsigned int length) {

Serial.println("On recieved message");

char json[length + 1];

strncpy(json, (char\*)payload, length);

json[length] = '\0';

Serial.print("Topic: ");

Serial.println(topic);

Serial.print("Message: ");

Serial.println(json);

// Giả mã gói Json Request

DynamicJsonDocument jsonBuffer(200);

DeserializationError error = deserializeJson(jsonBuffer, json);

if (error) {

Serial.print("deserializeJson() failed: ");

Serial.println(error.c\_str());

return;

}

// In gói request ra Serial

String methodName = String((const char\*)jsonBuffer["method"]);

if (methodName.equals("getGpioStatus")) {

// Reply with GPIO status

String responseTopic = String(topic);

responseTopic.replace("request", "response");

String gpioStatus = get\_gpio\_status();

client.publish(responseTopic.c\_str(), gpioStatus.c\_str());

}

else if (methodName.equals("setGpioStatus")) {

// Update GPIO status và reply

int pin = jsonBuffer["params"]["pin"];

bool enabled = jsonBuffer["params"]["enabled"];

set\_gpio\_status(pin, enabled);

String responseTopic = String(topic);

responseTopic.replace("request", "response");

String gpioStatus = get\_gpio\_status();

client.publish(responseTopic.c\_str(), gpioStatus.c\_str());

client.publish("v1/devices/me/attributes", gpioStatus.c\_str());

}

}

String get\_gpio\_status() {

// Chuẩn bị gpios JSON payload string

DynamicJsonDocument jsonBuffer(200);

JsonObject data = jsonBuffer.to<JsonObject>();

data[String(GPIO2\_PIN)] = gpioState[0] ? true : false;

data[String(GPIO14\_PIN)] = gpioState[1] ? true : false;

String strPayload;

serializeJson(data, strPayload);

Serial.print("Get gpio status: ");

Serial.println(strPayload);

return strPayload;

}

void set\_gpio\_status(int pin, boolean enabled) {

if (pin == GPIO2\_PIN) {

// Output GPIOs state

digitalWrite(GPIO2, enabled ? HIGH : LOW);

// Update GPIOs state

gpioState[0] = enabled;

} else if (pin == GPIO14\_PIN) {

// Output GPIOs state

digitalWrite(GPIO14, enabled ? HIGH : LOW);

// Update GPIOs state

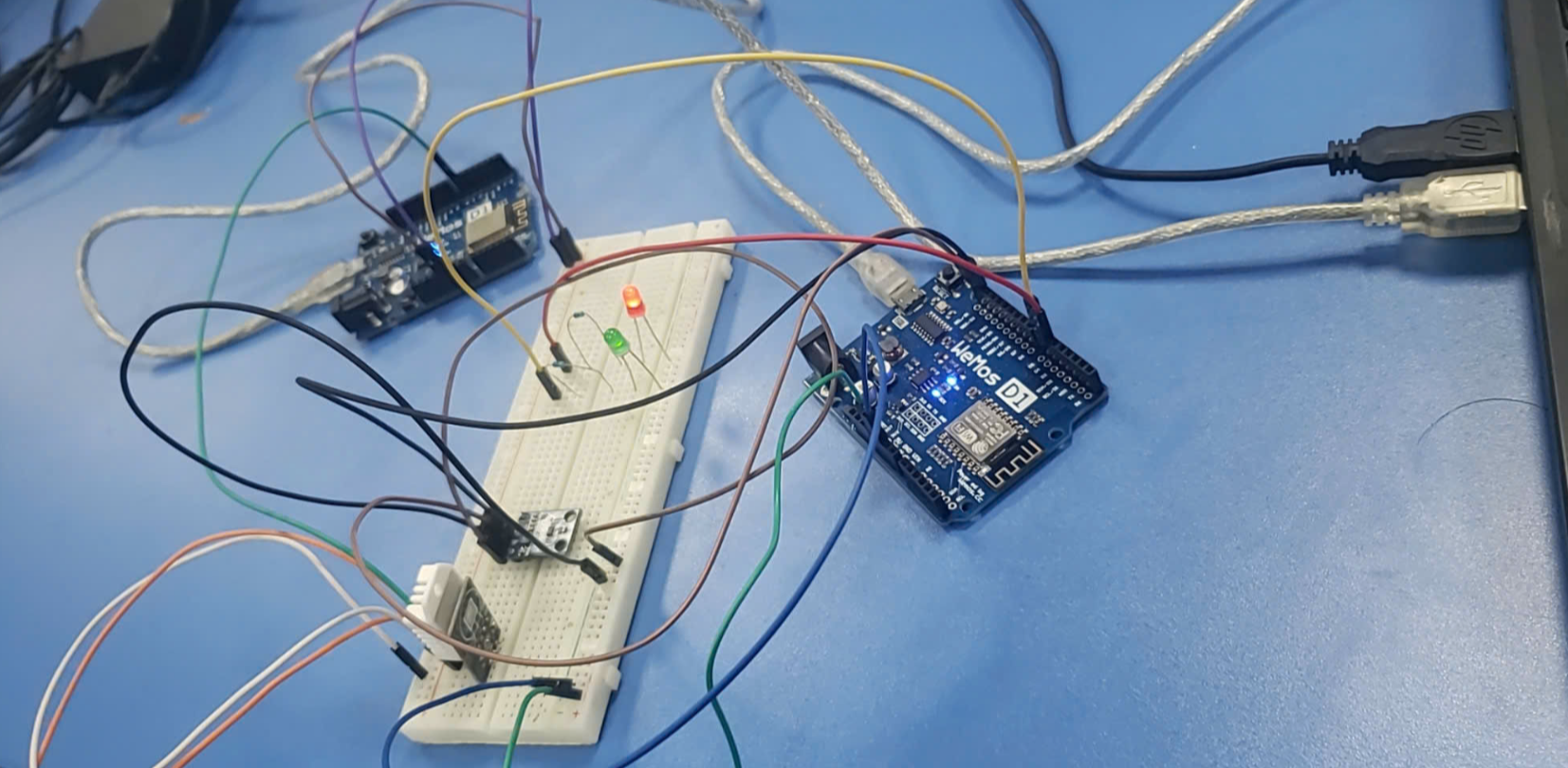
gpioState[1] = enabled;

}

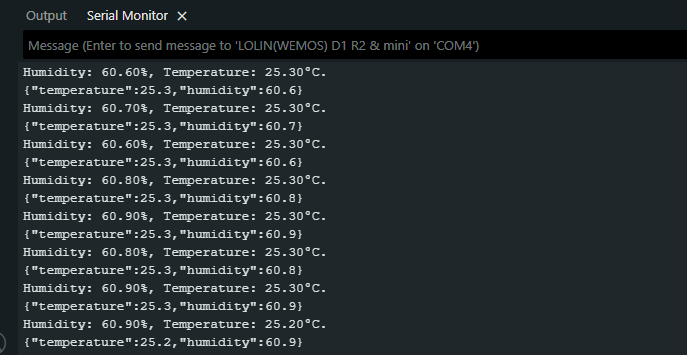
}

### Demo:

Tổng quan mô hình:



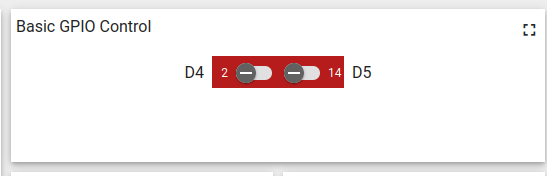
Các cảm biến sẽ liên tục gửi dữ liệu đến Thingsboard sau mỗi 10s:

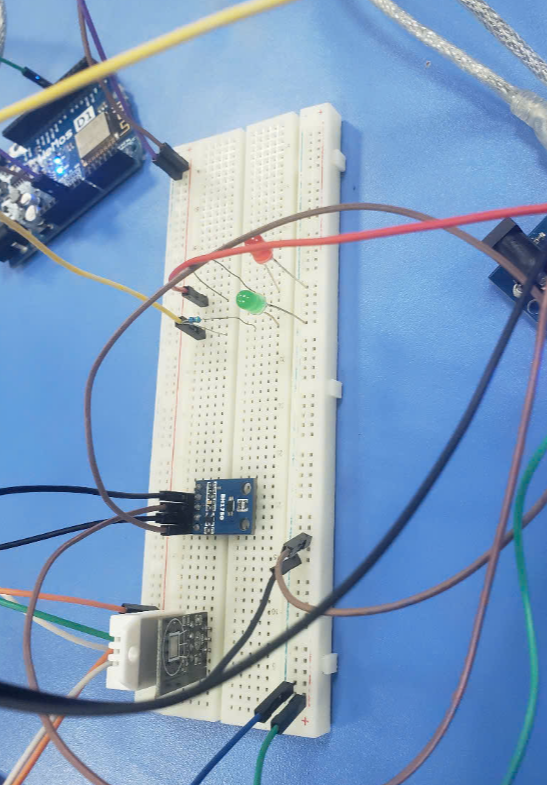




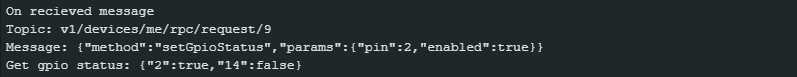
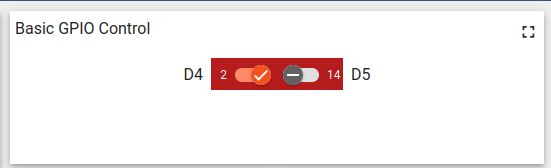
Khi bật tắt đèn trên Platform sẽ in ra gói request trên Serial và điều chỉnh đèn tương ứng:

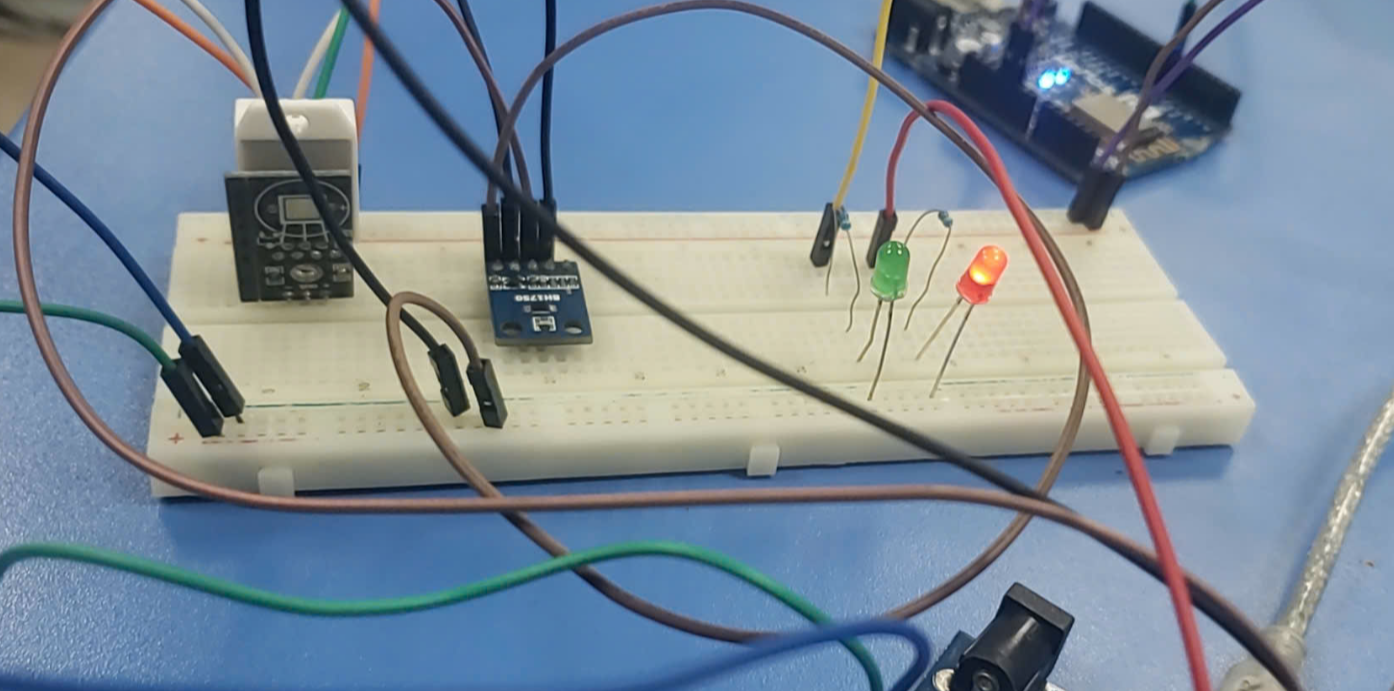
Khi chưa bật:



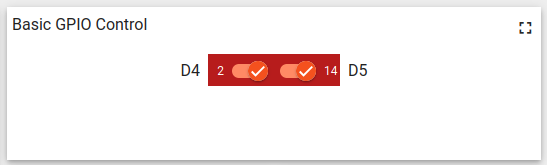


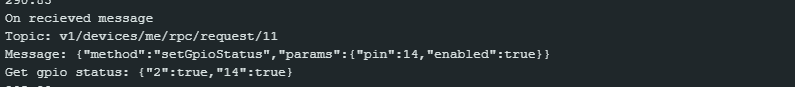
Khi bật D4:

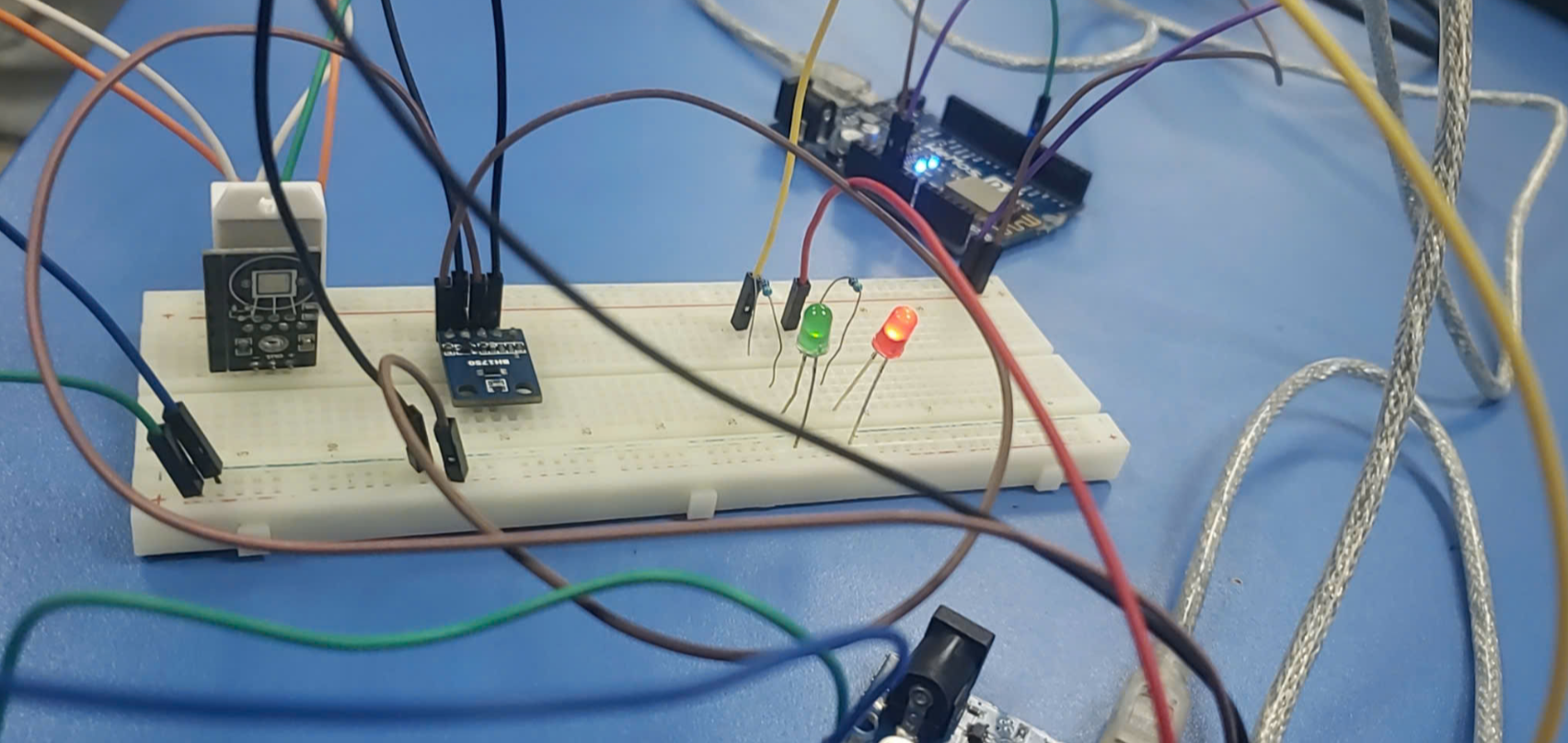




Khi bật D5:







Chi tiết video demo tại đây:

# TÀI LIỆU THAM KHẢO