

LỜI CẢM ƠN

Chúng em xin chân thành gửi lời cảm ơn sâu sắc đến Thầy Phạm Văn Hưởng. Trong suốt quá trình thực hiện đề tài này, Thầy luôn là người đồng hành cùng chúng em với sự chỉ dẫn tận tình, những góp ý nhận xét và những kinh nghiệm quý báu được Thầy truyền đạt lại đã tạo điều kiện thuận lợi để chúng em hoàn thành tốt đề tài.

Chúng em xin cảm ơn các Thầy, Cô trong khoa Công nghệ Thông tin đã giảng dạy cho chúng em những kiến thức nền tảng để chúng em có tiền đề để thực hiện đề tài này.

Xin chân thành cảm ơn!

NGƯỜI THỰC HIỆN

Cao Thị Thùy Linh

Phạm Bá Hiếu

Trương Việt Hoàng

LỜI NÓI ĐẦU

Trong thời đại hiện nay, công nghệ phát triển luôn đem không ít lợi ích cho con người và kèm theo đó cũng là áp lực. Tuy nhiên mọi thứ cũng đều có tính công bằng nó. Hãy thử tưởng tượng khi chúng ta mệt mỏi và bước chân về nhà, thứ chào đón chính là không khí mát lạnh của chiếc điều hòa đã được bật từ 30 phút trước, ánh đèn tự động bật lên và điều chỉnh mức sáng để đỡ cảm thấy nóng bức hoặc chỉ đơn giản là đỡ chói mắt, tiếng nhạc du dương được mở mà bản thân không phải mất công tìm điều khiển hay đến tận nơi bấm nút máy phát nhạc, nước đã được chuẩn bị chỉ cần thả mình,...điều đó thật thú vị phải không.

Thực ra những điều nói trên chính là một phần ứng dụng của công nghệ IOT( Internet of Things) đang được rất nhiều nhà công nghệ quan tâm hiện nay, cụ thể là lắp đặt các thiết bị điện, điện tử có thể được điều khiển hoặc tự động hoá hoặc bán tự động. Thay thế con người trong thực hiện một hoặc một số thao tác quản lý, điều khiển. Hệ thống điện tử này giao tiếp với người dùng thông qua bảng điện tử đặt trong nhà, ứng dụng trên điện thoại di động, máy tính bảng hoặc một giao diện web.

Khi điều khiển thiết bị điện bằng ThingsBoard, có nghĩa là chúng ta đang dùng công nghệ để làm cuộc sống thoải mái hơn. Chúng sẽ giảm khối lượng công việc của con người. Giúp chúng ta có nhiều thời gian thư giãn chứ không làm chúng ta lười đi. Đó chính là lợi ích của công nghệ! Vậy nên để bắt đầu tìm hiểu và phát triển những thứ vô cùng thân thiện và thiết thực này. Chúng ta sẽ bắt nghiên cứu từ những thứ ở mức đơn giản, cụ thể là dự án “Thiết kế, thi công hệ thống điều khiển thiết bị điện bằng ThingsBoard”được trình bày chi tiết ở dưới đây.

# 

# 

MỤC LỤC

[**CHƯƠNG 1: THIẾT KẾ**](#_dj07th68ckeq) **6**

[1.1. Phân tích, lựa chọn công nghệ](#_9geht5g7fst3) 6

[1.1.1. Tổng quan về thingsBoard](#_8r8o0535fl0t) 6

[1.1.2. Phương thức giao tiếp(Hiếu)](#_cdabu0eqgr) 8

[1.1.2.1. CoAP](#_itccn7fymu85) 9

[1.1.2.1.1: Cơ bản về CoAP](#_eiih73rlstlw) 9

[1.1.2.1.2: Xác thực CoAP và mã lỗi](#_yuwkqu619ppe) 9

[1.1.2.1.3: Đinh dạng key-value](#_nkhzg9dmo092) 9

[1.1.2.2. MQTT](#_lu0xk9or5sdy) 10

[1.2. Sơ đồ](#_usn3yrth49v) 10

[1.2.1. Sơ đồ tổng thể](#_c6jy2c748yey) 10

[1.2.2. Sơ đồ phần cứng](#_4ajeouckrp5e) 10

[1.2.3. Sơ đồ luồng dữ liệu](#_yv1ay7oxy545) 11

[1.3. Thiết kế và thi công phần cứng](#_v3rxkagabwlu) 11

[1.3.1. Danh sách các linh kiện](#_d8vd597dz7a6) 11

[1.3.2. Giới thiệu 1 số linh kiện](#_lrnrr0t29bml) 12

[1.3.2.1. Arduino WeMos D1 WiFi UNO ESP8266](#_obg3of8xjgd1) 12

[1.3.2.2. Module relay 4 chân](#_tkosyg4udndo) 15

[1.3.3. Thi công lắp ráp mô hình](#_o85ahgtds50x) 16

[1.4. Thiết kế bộ điều khiển trên thingsboard](#_ovsgf9pqgl00) 17

[1.4.1. Cài đặt thingsboard trên windows](#_ik2xujw92hsn) 17

[1.4.2. Cấu hình trên ThingsBoard server](#_k8syrmdiqtwf) 22

[1.5. Phát triển phần mềm](#_2v0acunbhkt0) 29

[1.5.1. Phát triển phần mềm trên arduino](#_e5pq1uc42s8l) 29

[1.5.2. Phát triển phần mềm trên mobile(Hoàng)](#_23py6q6sx4t2) 33

[1.5.2.1. Tổng quan về công nghệ](#_jbi127ecgcet) 33

[Ngôn ngữ lập trình Dart](#_4z9z9ms4ajac) 34

[Framework Flutter](#_g6hh9qst9uvb) 34

[Bộ công cụ nhận dạng giọng nói Vosk API](#_vhq2y3cu9z0n) 35

[1.5.2.2. Sơ đồ giao tiếp](#_zi6n2ouw80hj) 35

[1.5.3. Phát triển phần mềm trên web(Hiếu)](#_m0s84qdvm6va) 35

[**CHƯƠNG 2: TRIỂN KHAI**](#_6cengjr4uixu) **36**

[**CHƯƠNG 3: HƯỚNG DẪN SỬ DỤNG(Hoàng + Hiếu)**](#_a000dcf6jmi) **38**

[**CHƯƠNG 4: MÔ TẢ CODE**](#_h10qnxnegvya) **39**

[4.1. Code trên arduino](#_umgxq057rk4w) 39

[4.2. Code trên mobile(Hoàng)](#_j9omjaxueixv) 42

[4.3. Code trên web(Hiếu)](#_atcir523h1y5) 42

# 

# **CHƯƠNG 1: THIẾT KẾ**

## **1.1.** **Phân tích, lựa chọn công nghệ**

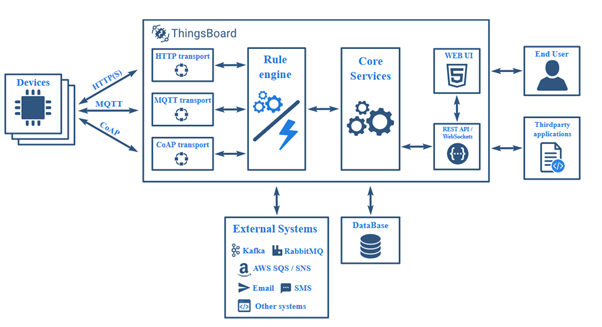
### 1.1.1. Tổng quan về thingsBoard

**Định nghĩa**:

ThingsBoard là một nền tảng IoT nguồn mở cho phép phát triển, quản lý và nhân rộng nhanh chóng các dự án IoT. Với nền tảng ThingsBoard, bạn có thể thu thập, xử lý, hiện thực quan và quản lý thiết bị

**Kiến trúc hệ thống**:

* Monolithic:



Hình 2.1 Khối kiến trúc ThingsBoard

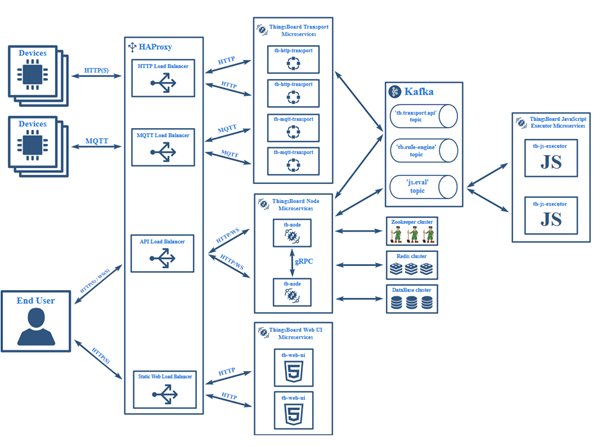
Gộp tất cả thành phần của hệ thống như phần giao diện người dùng, xử lý logic và database vào một package

*Ưu điểm:*

* Dễ dàng phát triển với ứng dụng nhỏ
* Triển khai dễ dàng, nhanh chóng
* Yêu cầu cơ sở hạ tầng đơn giản

*Nhược điểm*:

* Khó khăn trong việc bảo trì, thành viên mới sẽ khó tiếp cận vì nó là 1 khối lớn
* Các nhóm tham gia vào dự án phụ thuộc lẫn nhau, khó mở rộng quy mô nhóm
* Tốn thời gian xây dựng do có sự thay đổi nào thì hệ thống lại xây dựng lại cả dự án
* Tính ổn định không cao
* Microservices



Hình 2.2 Khối service ThingsBoard

Tách nhỏ các chức năng ra thành các service riêng lẻ độc lập. Các service giao tiếp với nhau thông qua các phương thức như: rest API, GRPC, lambda,...

*Ưu điểm:*

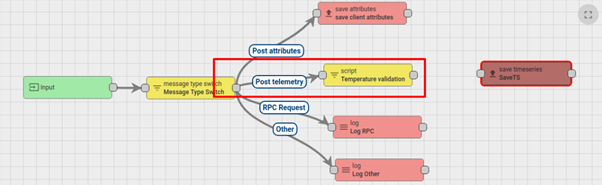
* Các tính năng được phát triển độc lập
* Dễ bảo trì, dễ tiếp cận với người mới
* Dễ hình dung được các chức năng của ứng dụng

*Nhược điểm*:

* Thời gian phát triển ban đầu chậm
* Khó khăn trong việc quản lý service
* Yêu cầu cơ sở hạ tầng tương đối

**Tính năng**

* **Thu thập dữ liệu từ xa**(collect telemetry data): ThingsBoard sẽ hỗ trợ thu thập và lưu trữ dữ liệu từ xa theo cách đáng tin cậy. Ta có thể truy cập dữ liệu đã thu được bằng cách sử dụng tổng quan web tùy chỉnh hoặc API phía máy chủ
* **Hiển thị trực quan dữ liệu thu thập**(visualize telemetry data): Bao gồm các khả năng trực quan hóa dữ liệu: Widgets, Dashboards , DashBoard states.
* **Phân tích dữ liệu và đưa ra báo động phù hợp**(alarms): ThingsBoard cung cấp khả năng tạo và quản lý các cảnh báo liên quan đến các thực thể: device, asset,.. Cho phép giám sát báo động theo thời gian thực và báo động cho việc phân cấp các thực thể liên quan
* **Công cụ tạo chuỗi quy tắc kéo thả**(rule chains): Xử lý dữ liệu thiết bị đến bằng chuỗi quy tắc linh hoạt dựa trên thuộc tính thực thể hoặc nội dung tin nhắn. Chuyển tiếp dữ liệu tới hệ thống bên ngoài hoặc kích hoạt báo thức bằng cách sử dụng logic tùy chỉnh. Định cấu hình chuỗi thông báo phức tạp trên các thông báo. Làm phong phú thêm chức năng phía máy chủ hoặc điều khiển thiết bị của ta bằng các quy tắc có thể tùy chỉnh cao



Hình 2.3. Chuỗi quy tắc trong Thingsboard

* **Quản lý thiết bị(RPC):** API và widget để đẩy các lệnh từ ứng dụng và bảng điều khiển của ta sang thiết bị và ngược lại.

### 1.1.2. Phương thức giao tiếp

* Giao tiếp với thingboard dựa trên 3 phương thức:
* CoAP
* MQTT
* HTTP/HTTPS

#### 1.1.2.1. CoAP

##### 1.1.2.1.1. Cơ bản về CoAP

* CoAP là một giao thức IoT trọng lượng nhẹ dành cho các thiết bị bị ràng buộc.
* Giao thức CoAP dựa trên UDP, nhưng tương tự như HTTP, nó sử dụng mô hình phản hồi yêu cầu. [Tùy chọn](https://tools.ietf.org/html/rfc7641) CoAP quan sát cho phép đăng ký tài nguyên và nhận thông báo về sự thay đổi tài nguyên.

##### 1.1.2.1.2. Xác thực CoAP và mã lỗi

* Chúng tôi sẽ sử dụng *thông* tin đăng nhập thiết bị *mã thông báo truy cập* trong bài viết này và sau này chúng sẽ được gọi là $ ACCESS\_TOKEN . Ứng dụng cần bao gồm $ ACCESS\_TOKEN làm tham số đường dẫn vào mỗi yêu cầu CoAP. Các mã lỗi có thể xảy ra và lý do của chúng:
* 4.00 Yêu cầu không hợp lệ - URL, thông số yêu cầu hoặc nội dung không hợp lệ.
* 4.01 Trái phép - $ ACCESS\_TOKEN không hợp lệ .
* 4.04 Không tìm thấy - Không tìm thấy tài nguyên.

##### 1.1.2.1.3. Đinh dạng key-value

* Theo mặc định, ThingsBoard hỗ trợ nội dung khóa-giá trị trong JSON. Khóa luôn là một chuỗi, trong khi giá trị có thể là chuỗi, boolean, double, long hoặc JSON. Cũng có thể sử dụng định dạng nhị phân tùy chỉnh hoặc một số khuôn khổ tuần tự hóa. Xem [tùy chỉnh giao thức](https://thingsboard.io/docs/reference/coap-api/#protocol-customization) để biết thêm chi tiết. Ví dụ:

**{**

**"stringKey":"value1",**

**"booleanKey":true,**

**"doubleKey":42.0,**

**"longKey":73,**

**"jsonKey": {**

**"someNumber": 42,**

**"someArray": [1,2,3],**

**"someNestedObject": {"key": "value"}**

**}**

**}**

#### 1.1.2.2. MQTT

##### 1.1.2.2.1. Cơ bản về mqtt

* MQTT là một giao thức nhắn tin đăng ký xuất bản nhẹ có thể làm cho nó phù hợp nhất với các thiết bị IoT khác nhau.
* Các nút máy chủ ThingsBoard hoạt động như một Nhà môi giới MQTT hỗ trợ QoS mức 0 (nhiều nhất một lần) và 1 (ít nhất một lần) và một tập hợp các chủ đề có thể [định cấu hình](https://thingsboard.io/docs/user-guide/device-profiles/#mqtt-device-topic-filters) .

##### 1.1.2.2.2. Kết nối MQTT

* Chúng tôi sẽ sử dụng *thông* tin đăng nhập thiết bị *mã thông báo truy cập* trong bài viết này và sau này chúng sẽ được gọi là $ ACCESS\_TOKEN . Ứng dụng cần gửi tin nhắn MQTT CONNECT với tên người dùng có chứa $ ACCESS\_TOKEN . Tùy chọn thay thế là sử dụng [Thông tin đăng nhập MQTT Cơ bản](https://thingsboard.io/docs/user-guide/basic-mqtt/) - sự kết hợp của id khách hàng, tên người dùng và mật khẩu;
* Các mã trả lại có thể có và lý do của chúng trong quá trình kết nối:
* Đã kết nối 0x00 - Đã kết nối thành công với máy chủ MQTT ThingsBoard.
* Kết nối 0x04 Bị từ chối, tên người dùng hoặc mật khẩu không hợp lệ - Tên người dùng trống.
* Kết nối 0x05 Bị từ chối, không được ủy quyền - Tên người dùng chứa $ ACCESS\_TOKEN không hợp lệ .

##### 1.1.2.2.3. Định dạng Key-value

* Theo mặc định, ThingsBoard hỗ trợ nội dung khóa-giá trị trong JSON. Khóa luôn là một chuỗi, trong khi giá trị có thể là chuỗi, boolean, double, long hoặc JSON.
* Cũng có thể sử dụng định dạng nhị phân tùy chỉnh hoặc một số khuôn khổ tuần tự hóa. Xem [tùy chỉnh giao thức](https://thingsboard.io/docs/reference/mqtt-api/#protocol-customization) để biết thêm chi tiết. Ví dụ:

**{**

**"stringKey":"value1",**

**"booleanKey":true,**

**"doubleKey":42.0,**

**"longKey":73,**

**"jsonKey": {**

**"someNumber": 42,**

**"someArray": [1,2,3],**

**"someNestedObject": {"key": "value"}**

**}**

**}**

#### 1.1.2.2. HTTP/HTTPS

##### 1.1.2.2.1. Cơ bản về HTTP/HTTPS

* [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) là một giao thức mạng có mục đích chung có thể được sử dụng trong các ứng dụng IoT.
* Các nút máy chủ ThingsBoard hoạt động như một Máy chủ HTTP hỗ trợ cả giao thức HTTP và HTTPS.

##### 1.1.2.2.2. Xác thực HTTP và mã lỗi

* Chúng tôi sẽ sử dụng *thông* tin đăng nhập thiết bị *mã thông báo truy cập* trong bài viết này và sau này chúng sẽ được gọi là $ ACCESS\_TOKEN . Ứng dụng cần bao gồm $ ACCESS\_TOKEN làm tham số đường dẫn trong mỗi yêu cầu HTTP. Các mã lỗi có thể xảy ra và lý do của chúng:
* 400 Yêu cầu không hợp lệ - URL, thông số yêu cầu hoặc nội dung không hợp lệ.
* 401 Không được phép - $ ACCESS\_TOKEN không hợp lệ .
* 404 Not Found - Không tìm thấy tài nguyên.

##### 1.1.2.2.3. Xác thực HTTP và mã lỗi

* Theo mặc định, ThingsBoard hỗ trợ nội dung khóa-giá trị trong JSON. Khóa luôn là một chuỗi, trong khi giá trị có thể là chuỗi, boolean, double, long hoặc JSON.
* Cũng có thể sử dụng định dạng nhị phân tùy chỉnh hoặc một số khuôn khổ tuần tự hóa. Xem [tùy chỉnh giao thức](https://thingsboard.io/docs/reference/http-api/#protocol-customization) để biết thêm chi tiết. Ví dụ:

**{**

**"stringKey":"value1",**

**"booleanKey":true,**

**"doubleKey":42.0,**

**"longKey":73,**

**"jsonKey": {**

**"someNumber": 42,**

**"someArray": [1,2,3],**

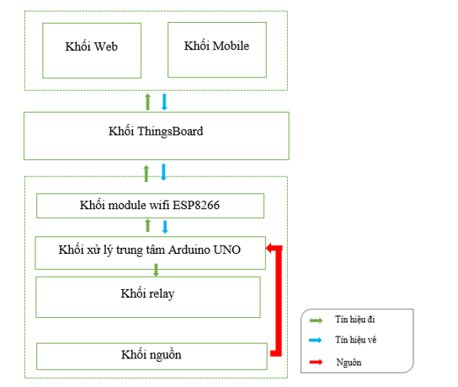
**"someNestedObject": {"key": "value"}**

**}**

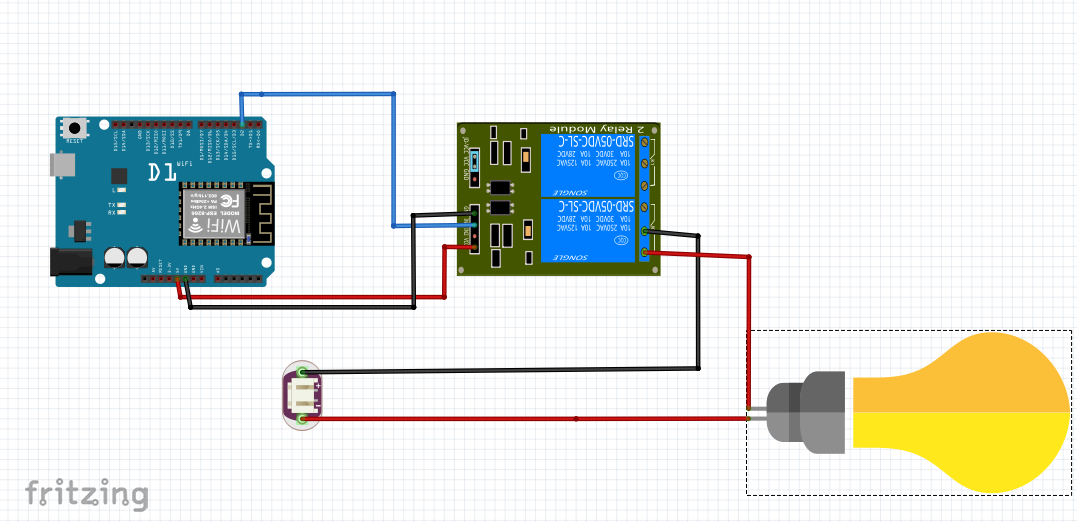
**}**

## **1.2. Sơ đồ**

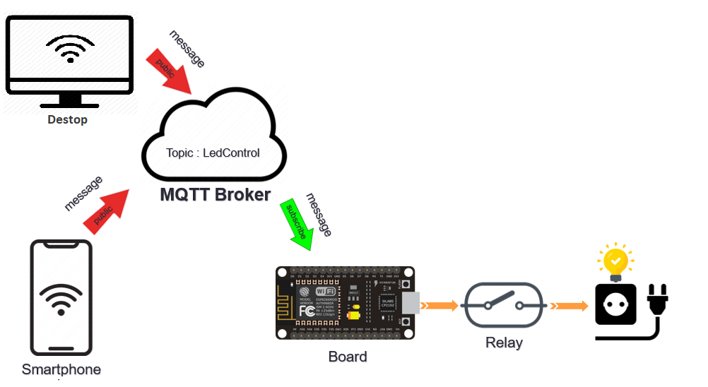
### 1.2.1. Sơ đồ tổng thể



### 1.2.2. Sơ đồ phần cứng



### 1.2.3. Sơ đồ luồng dữ liệu



## **1.3. Thiết kế và thi công phần cứng**

### 1.3.1. Danh sách các linh kiện

| **STT** | **Tên thiết bị, linh kiện** | **Số lượng** |
| --- | --- | --- |
| 1 | Arduino WeMos D1 WiFi UNO ESP8266 | 1 |
| 2 | Module relay 4 chân | 2 |
| 3 | Dây điện | 3m |
| 4 | Bóng đèn tròn + đui đèn | 1 |
| 5 | Bìa cứng | 1 tấm |

### 1.3.2. Giới thiệu 1 số linh kiện

#### 1.3.2.1. Arduino WeMos D1 WiFi UNO ESP8266



WEMOS D1 Wifi là kit phát triển phiên ta mới nhất từ WeMos, kit được thiết kế với hình dáng tương tự Arduino Uno nhưng trung tâm lại là module wifi Soc ESP8266EX được build lại firmware để có thể chạy với chương trình Arduino. Kit thích hợp và dễ dàng thực hiện các ứng dụng thu thập dữ liệu và điều khiển qua Wifi.

Đặc điểm:

* Vi điều khiển: ESP8266EX
* Điện áp hoạt động: 3.3V
* Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One-wire, trừ chân D0)
* Số chân Analog Input: 1 (điện áp vào tối đa 3.3V)
* Bộ nhớ Flash: 4MB
* Điện áp vào: 9-24V
* Điện áp ra: 5V - Dòng max: 1A
* Giao tiếp: [Cable Micro USB](https://iotmaker.vn/cable-micro-usb-b-80cm.html)
* Wifi: 2.4 GHz
* Hỗ trợ bảo mật: WPA/WPA2
* Tích hợp giao thức TCP/IP
* Kích thước: 68.6mm x 53.4mm (2.701" x 2.102")
* Lập trình trên các ngôn ngữ: C/C++, Micropython, NodeMCU - Lua

Pinout:

| **Pin** | **Tính năng** | **ESP8266 Pin** |
| --- | --- | --- |
| TX | TXD | TXD |
| RX | RXD | RXD |
| A0 | Analog Input | A0 |
| D0 | I/O | GPIO16 |
| D1 | I/O, SCL | GPIO5 |
| D2 | I/O, SDA | GPIO4 |
| D3 | I/O, 10k pull-up | GPIO0 |
| D4 | I/O, 10k pull-up, BUILTIN\_LED | GPIO2 |
| D5 | I/O, SCK | GPIO14 |
| D6 | I/O, MISO | GPIO12 |
| D7 | I/O, MOSI | GPIO13 |
| D8 | I/O, 10k pull-down, SS | GPIO15 |
| GND | Ground | GND |
| 5V | 5V |  |
| 3V3 | 3.3V | 3.3V |
| RST | Reset | RST |

#### 1.3.2.2. Module relay 4 chân



Relay hay còn gọi Rơ-le là một công tắc (khóa K). Nhưng khác với công tắc ở một chỗ cơ bản, rơ-le được kích hoạt bằng điện thay vì dùng tay người. Chính vì lẽ đó, rơ-le được dùng làm công tắc điện tử! Vì rơ-le là một công tắc nên nó có 2 trạng thái: đóng và mở.

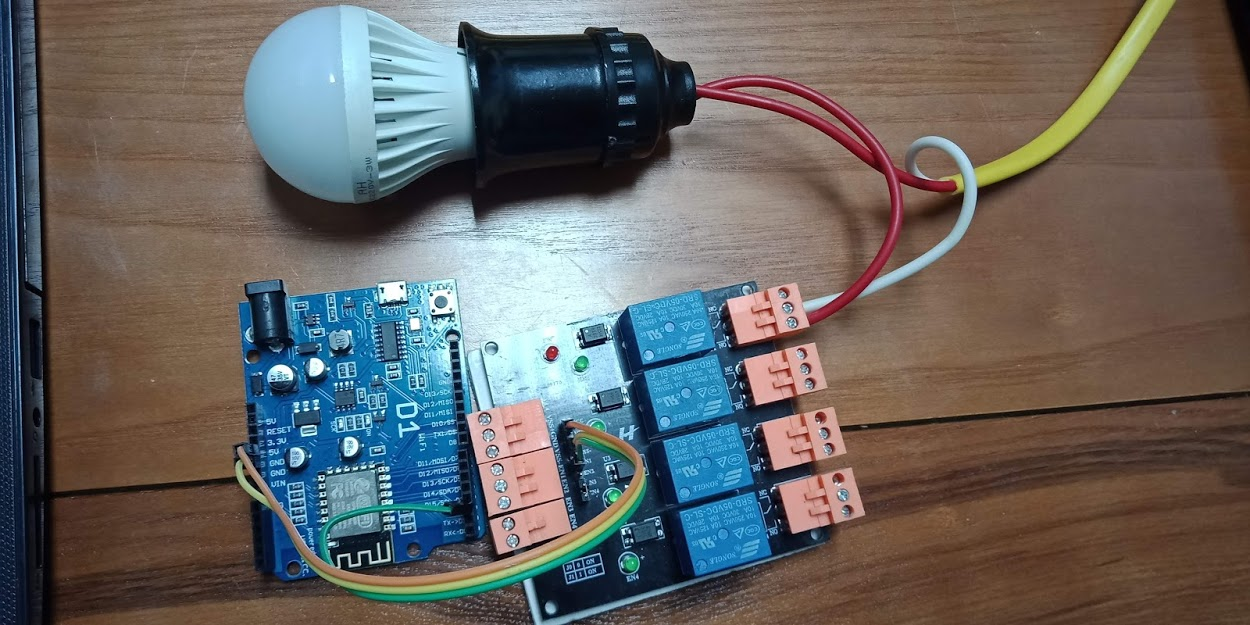
**Tính năng của relay 4 chân**

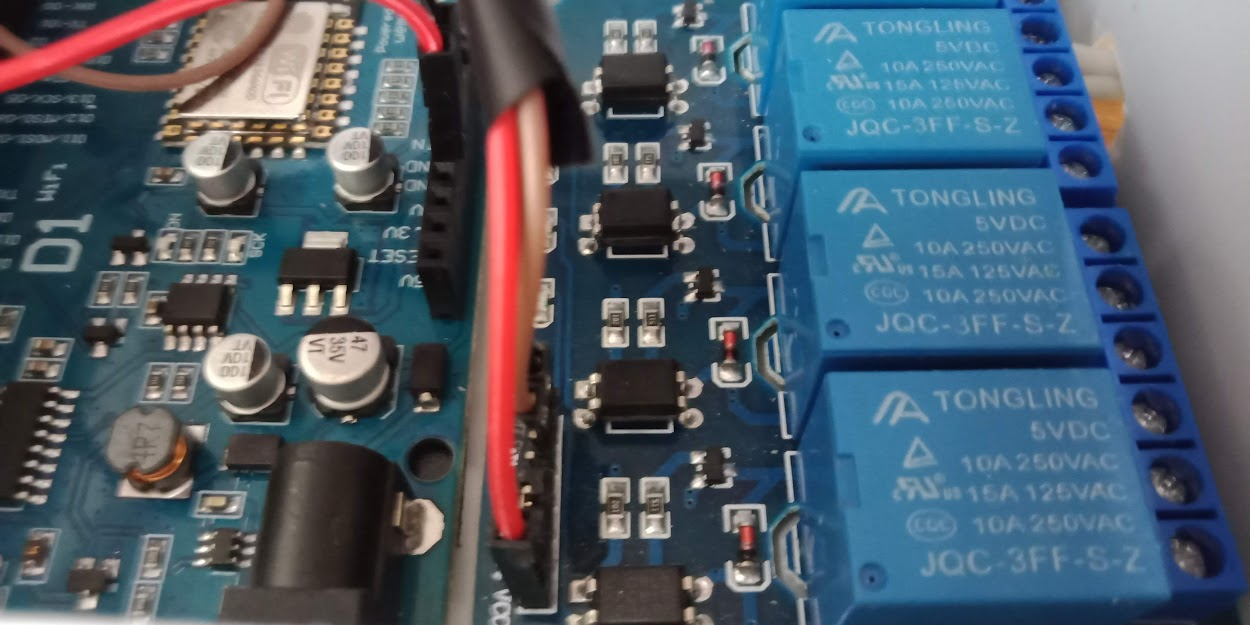
* Module Relay 4 kênh 12V Âm Kích được sử dụng để điều khiển thiết bị điện tử công suất cao như đèn, quạt điện và điều hòa không khí,...
* Module Relay 4 kênh 12V Âm Kích được sử dụng rộng rãi và phổ biến trong các hệ thống IoT, nhà thông minh, vườn thông minh,... để điều khiển thiết bị tắt mở một cách dễ dàng và nhanh chóng.

**Thông số kỹ thuật**

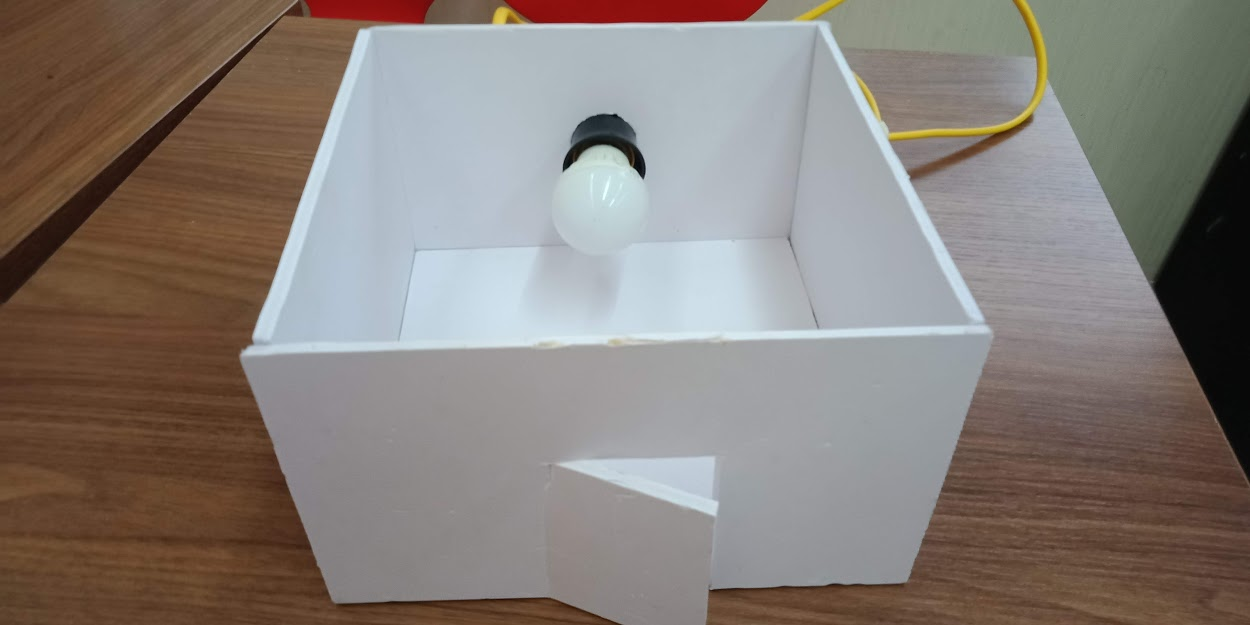
* Rơle chất lượng cao, tải tối đa: AC250V / 10A, DC 30V / 10A.
* Sử dụng bộ ghép nối cách ly, khả năng điều khiển rất mạnh, hiệu suất ổn định, kích hoạt chỉ 5mA.
* Số lượng Rơle: 4 Rơle
* Điện áp hoạt động: 12VDC.
* Có thể thay đổi chân kích: âm kích.
* Thiết kế fault-tolerant, khi dòng điều khiển bị hỏng, relay sẽ không hoạt động, tránh hỏng thiết bị sau.
* Thiết kế giao diện thân thiện với người dùng . Tất cả các kết nối có thể được kết nối trực tiếp qua khối đầu cuối. Rất thuận tiện.
* Kích thước: 71,5 \* 50 \* 18,5 mm
* Các chân của Module Relay 8 Kênh:
* DC +: Kết nối cực dương của nguồn điện.
* DC-: Kết nối cực âm của nguồn điện.
* IN1-IN4: Theo cài đặt của người dùng, nó có thể ở mức cao hoặc thấp.
* NO1 - NO4: Cổng Thường Mở
* COM1 - COM4: Cổng chung rơle
* NC1 - NC4: Cổng Thường Đóng

### 1.3.3. Thi công lắp ráp mô hình









## **1.4. Thiết kế bộ điều khiển trên thingsboard**

### 1.4.1. Cài đặt thingsboard trên windows

**Điều kiện cài đặt**: Windows 10/8.1/8/7 32-bit/64-bit. Yêu cầu phần cứng phụ thuộc vào cơ sở dữ liệu đã chọn và số lượng thiết bị được kết nối với hệ thống. Để chạy ThingsBoard và PostgreSQL trên một máy duy nhất, bạn sẽ cần ít nhất 2Gb RAM. Để chạy ThingsBoard và Cassandra trên một máy duy nhất, bạn sẽ cần ít nhất 8Gb RAM.

**Bước 1: Cài đặt Java 8 (OpenJDK)**

* Dịch vụ ThingsBoard đang chạy trên Java 8. Làm theo hướng dẫn này để cài đặt OpenJDK 8.

Truy cập Trang Tải xuống JDK [AdoptOpenJDK - Open source, prebuilt OpenJDK binaries].Mở để tải xuống gói MSI OpenJDK 8 (LTS) mới nhất.

Chạy gói MSI đã tải xuống và làm theo hướng dẫn. Hãy chắc chắn rằng bạn đã chọn **“Add to PATH**” và **“Set JAVA\_HOME variable”** tùy chọn để trạng thái "Sẽ được cài đặt trên ổ cứng local".

Bạn có thể kiểm tra cài đặt bằng cách sử dụng lệnh sau (sử dụng Dấu nhắc Lệnh):

*java -version*

Đầu ra dự kiến là:

*C:\Users\User>java -version*

*openjdk version "1.8.0\_212"*

*OpenJDK Runtime Environment (AdoptOpenJDK)(build 1.8.0\_212-b04)*

*OpenJDK 64-Bit Server VM (AdoptOpenJDK)(build 25.212-b04, mixed mode)*

**Bước 2: Cài đặt Server ThingsBoard**

Tải về và trích xuất các gói phần mềm.

[https://github.com/thingsboard/thingsboard/releases/download/v3.2/thingsboard-windows-3.2.zip]

Lưu ý: Chúng tôi giả sử bạn đã giải nén gói ThingsBoard đến vị trí mặc định: C:\Program Files (x86)\thingsboard

**Bước 3: Cấu hình cơ sở dữ liệu**

ThingsBoard có thể sử dụng phương pháp cơ sở dữ liệu SQL hoặc kết hợp. PostgreSQL là lựa chọn phổ biến nhất

Cài đặt PostgreSQL

Tải xuống tệp cài đặt (PostgreSQL 11.7 hoặc các bản phát hành mới hơn)[Download PostgreSQL Database for Windows, Linux and MacOS & 32-bit or 64-bit Versions | EDB (enterprisedb.com)] và làm theo hướng dẫn cài đặt.

Trong quá trình cài đặt PostgreSQL, bạn sẽ được nhắc nhập mật khẩu superuser (postgres). Đừng quên mật khẩu này. Nó sẽ được sử dụng sau này. Để đơn giản, chúng tôi sẽ thay thế nó bằng "postgres".

Tạo Cơ sở dữ liệu ThingsBoard

Sau khi cài đặt, khởi động phần mềm "pgAdmin" và đăng nhập như superuser (postgres). Mở máy chủ của bạn và tạo cơ sở dữ liệu "thingsboard" với chủ sở hữu "postgres".

Cấu hình ThingsBoard

Trong trường hợp bạn đã chỉ định mật khẩu siêu người dùng PostgreSQL là "postgres", bạn có thể bỏ qua bước này.

Mở Notepad hoặc trình soạn thảo khác với tư cách là người dùng quản trị viên (nhấp chuột phải vào biểu tượng ứng dụng và chọn "Chạy với tư cách administrator").

Mở tệp sau để chỉnh sửa (chọn "All Files" thay vì "Text Documents" trong hộp thoại chọn tệp, mã hóa là UTF-8):

C:\Program Files (x86)\thingsboard\conf\thingsboard.yml

và xác định vị trí "# SQL DAO cấu hình" khối. Đừng quên thay thế "postgres" bằng mật khẩu người dùng postgres thực sự của bạn:

*# SQL DAO Configuration*

*spring:*

*data:*

*jpa:*

*repositories:*

*enabled: "true"*

*jpa:*

*open-in-view: "false"*

*hibernate:*

*ddl-auto: "none"*

*database-platform: "${SPRING\_JPA\_DATABASE\_PLATFORM:org.hibernate.dialect.PostgreSQLDialect}"*

*datasource:*

*driverClassName: "${SPRING\_DRIVER\_CLASS\_NAME:org.postgresql.Driver}"*

*url: "${SPRING\_DATASOURCE\_URL:jdbc:postgresql://localhost:5000/thingsboard}"*

*username: "${SPRING\_DATASOURCE\_USERNAME:postgres}"*

*password: "${SPRING\_DATASOURCE\_PASSWORD:hoanghieulinh}"*

*hikari:*

*maximumPoolSize: "${SPRING\_DATASOURCE\_MAXIMUM\_POOL\_SIZE:5}"*

xác định vị trí tham số "SQL\_POSTGRES\_TS\_KV\_PARTITIONING" để ghi đè giá trị mặc định cho kích thước phân vùng lưu trữ khóa-giá trị dấu thời gian:

*sql:*

*postgres:*

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

*ts\_key\_value\_partitioning: "${SQL\_POSTGRES\_TS\_KV\_PARTITIONING:MONTHS}"*

**Bước 4: Chọn ThingsBoard queue service**

ThingsBoard có thể sử dụng các messaging systems/brokers để lưu trữ tin nhắn và giao tiếp giữa các dịch vụ Của ThingsBoard. Chọn queue phù hợp

**Bước 5: [Tùy chọn] Cập nhật bộ nhớ cho máy chậm (1GB RAM)**

Mở Notepad hoặc trình soạn thảo khác với tư cách là người dùng quản trị viên (nhấp chuột phải vào biểu tượng ứng dụng và chọn "Chạy với tư cách administrator").

Mở tệp sau để chỉnh sửa (chọn "All Files" thay vì "Text Documents" trong hộp thoại chọn tệp, mã hóa là UTF-8):

C:\Program Files (x86)\thingsboard\thingsboard.xml

Copy to clipboard

Định vị các dòng sau vào tệp cấu hình.

*<startargument>-Xms512m</startargument>*

*<startargument>-Xmx1024m</startargument>*

và thay đổi chúng thành

*<startargument>-Xms256m</startargument>*

*<startargument>-Xmx256m</startargument>*

**Bước 6. Chạy tập lệnh cài đặt**

Khởi chạy windows shell (Command Prompt) với tư cách Administratorị. Thay đổi thư mục vào thư mục cài đặt ThingsBoard.

Thực hiện tập lệnh ***install.bat*** để cài đặt ThingsBoard như một dịch vụ Windows (hoặc chạy "***install.bat --loadDemo***" để cài đặt và thêm dữ liệu demo). Điều này có nghĩa là nó sẽ được tự động bắt đầu khi khởi động hệ thống. Tương tự, ***uninstall.bat*** sẽ xóa ThingsBoard khỏi các dịch vụ Windows. Đầu ra phải tương tự như đầu ra này:

*C:\Program Files (x86)\thingsboard>install.bat --loadDemo*

*Detecting Java version installed.*

*CurrentVersion 18*

*Java 1.8 found!*

*Installing thingsboard ...*

*...*

*ThingsBoard installed successfully!*

**Bước 7: Khởi động ThingsBoard service**

Mở command prompt với tư cách Administrator và thực thi lệnh sau:

*net start thingsboard*

Đầu ra dự kiến:

*The ThingsBoard Server Application service is starting.*

*The ThingsBoard Server Application service was started successfully*.

Để khởi động lại ThingsBoard server, bạn có thể thực hiện các lệnh sau:

*net stop thingsboard*

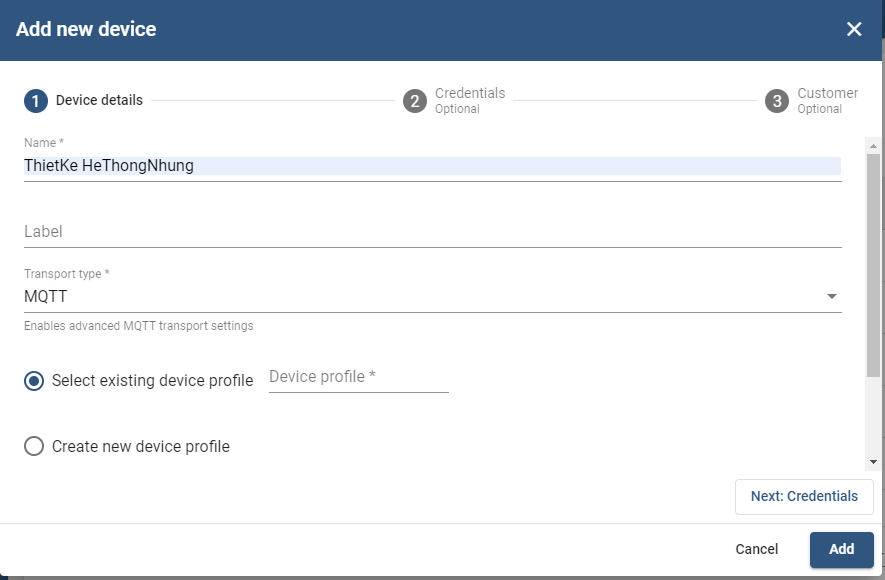
*net start thingsboard*

Sau khi bắt đầu, bạn sẽ có thể mở giao diện người dùng Web bằng cách sử dụng liên kết sau:

http://localhost:8080/

### 1.4.2. Cấu hình trên ThingsBoard server

Sau khi đăng nhập vào Thingsboard bằng tài khoản Tenant administrator. Vào phần Device nhấp vào “add device” hay “+” ở góc trên bên phải màn hình.



Điền đầy đủ thông tin về tên,thể loại và mô tả về thiết bị rồi nhấn ADD. Để quản lý chi tiết, click vào thẻ thiết bị mới được tạo ra:

Chú ý **ACCESS TOKEN** của nó:

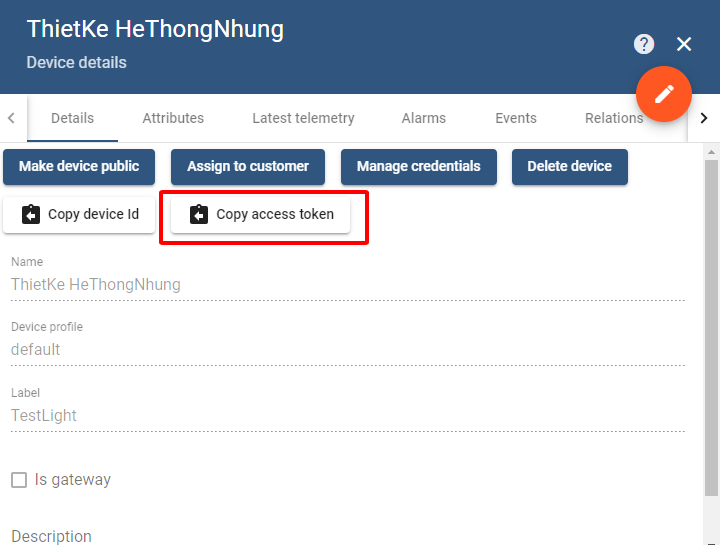
$**ACCESS\_TOKEN** là một mã(tối đa 20 ký tự) chứng thực của thiết bị đó nó sẽ được tạo ngẫu nhiên khi người sử dụng vừa tạo ra thiết bị đó tuy nhiên có thể sửa đổi nó.

Click vào COPY DEVICE ID để lấy ID của thiết bị.

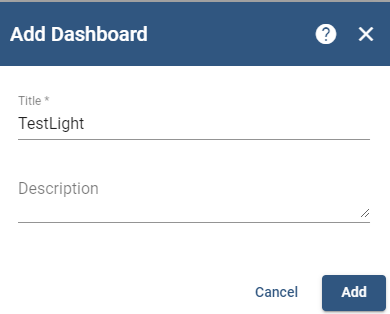
Click vào MAKE DEVICE PUBLIC cho nó thành thiết bị công khai.(tất cả các customer đều xem được).

Click vào MANAGE CREDENTIALS để có thể thay đổi $ **ACCESS TOKEN**.

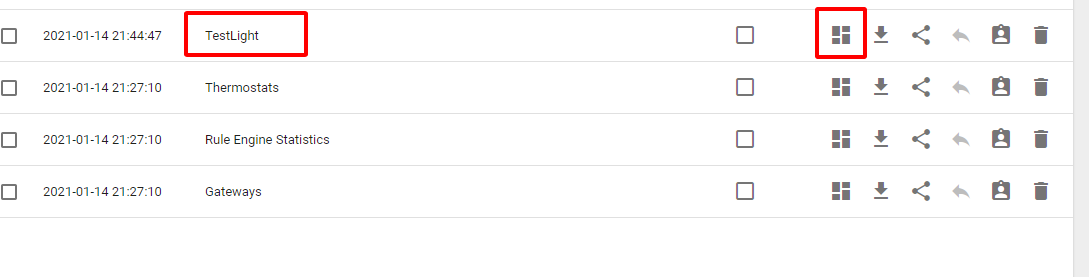
Nếu muốn sửa lại tên lúc đầu ta đặt tích vào cái nút hình cái bút chì màu đỏ góc trên bên phải màn hình.



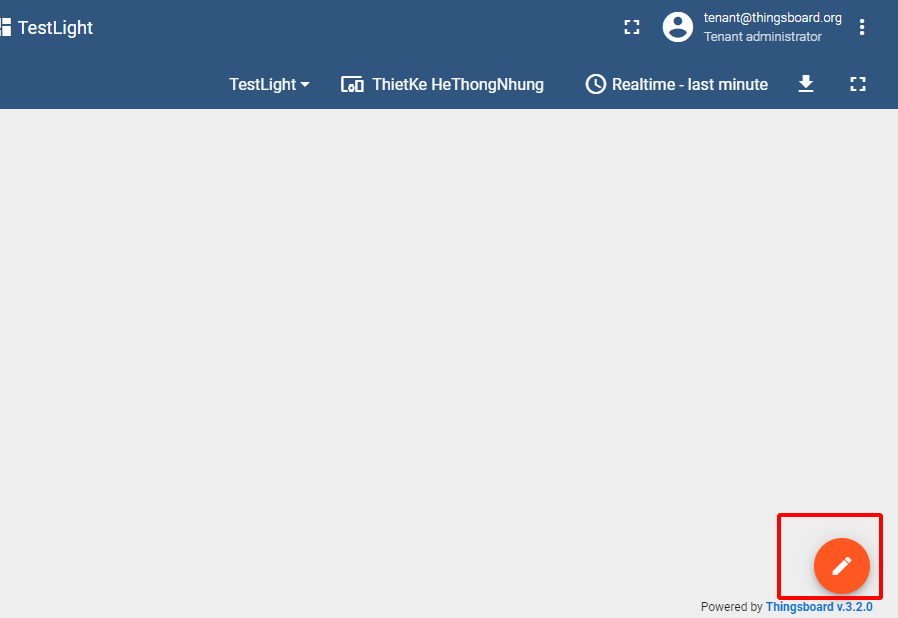
Sau đó tạo một Dashboard(bảng điều khiển) để cho người sử dụng điều khiển



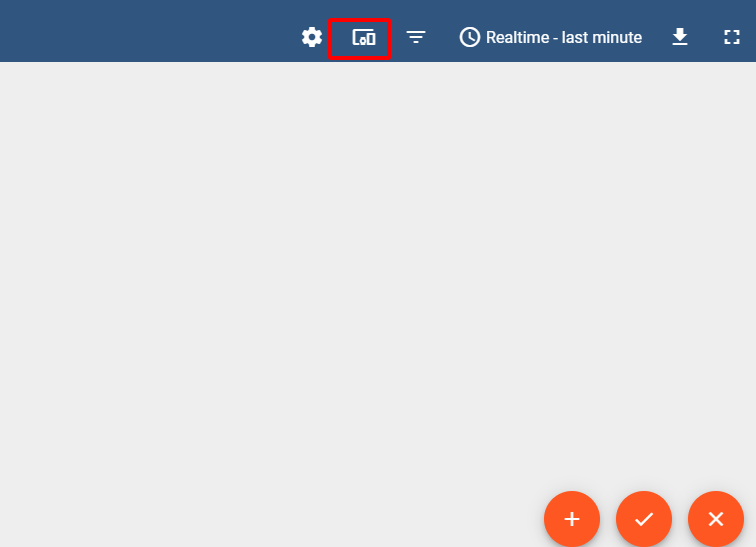
Điền thông tin đầy đủ rồi nhấn “ADD”. Khi đó ta có 1 dashboard. Để thực thi trên dashboard, ta click vào “open dashboard”, hoặc ký hiệu được đánh dấu.

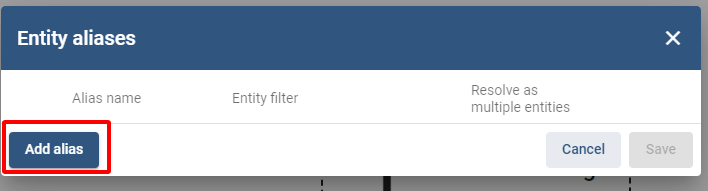


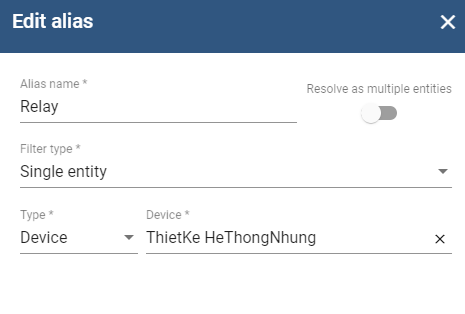
Sau đó chọn “edit”



Các tiện ích control sẽ cần được cung cấp một thiết bị đích. Trong trường hợp này thiết bị đích chính là thiết bị ảo(device) ta tạo trên. Tuy nhiên ta cần gán bí danh cho thực thể thiết bị đích này để dễ dàng sử dụng:







Cấu hình như hình trên với:

• *Alias name*: Bí danh của thực thể đặt tên tùy ý.

• *Filter type*: Cách thức lọc lấy dữ liệu, có nhiều cách lọc dữ liệu tuy nhiên ở đây lấy theo thực thể đơn độc.

• *Type:* Lấy dữ liệu theo loại thực thể là Device.

• *Device:* Tên của Device cần lấy dữ liệu(chính là tên thiết bị tạo lúc đầu).

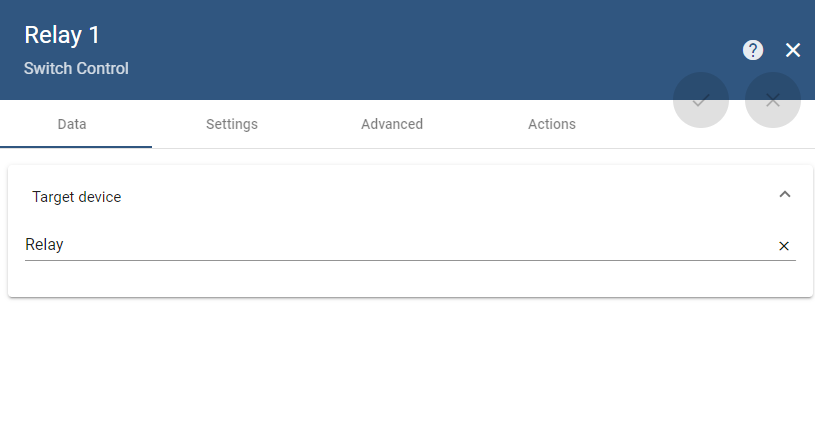
Hướng dẫn này sử dụng tiện ích Switch control:

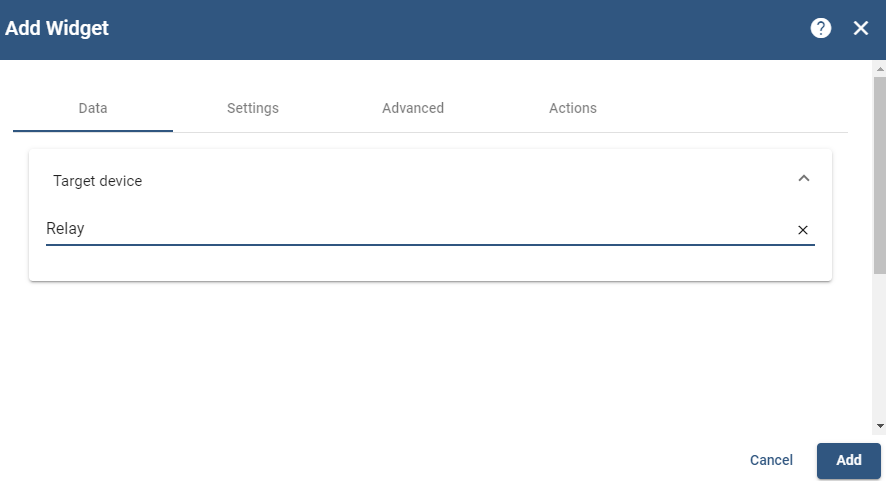


Sử dụng 5 nút nhấn Relay mỗi nút điều khiển 1 thiết bị, và các Status Delay để cập nhật trạng thái cho từng thiết bị.



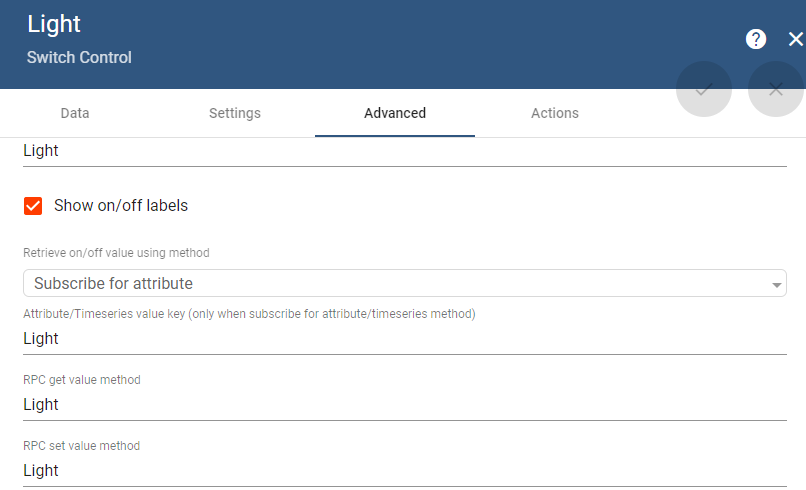
Tất cả tiện ích trên đều có Target device là bí danh khi nãy đặt





Với tiện ích Switch control, trong tab ADVANCED hãy chú ý tới một số phần như hình sau:

* 5 Switch control đầu dùng để bật tắt các thiết bị: khi đó tích vào mục 1



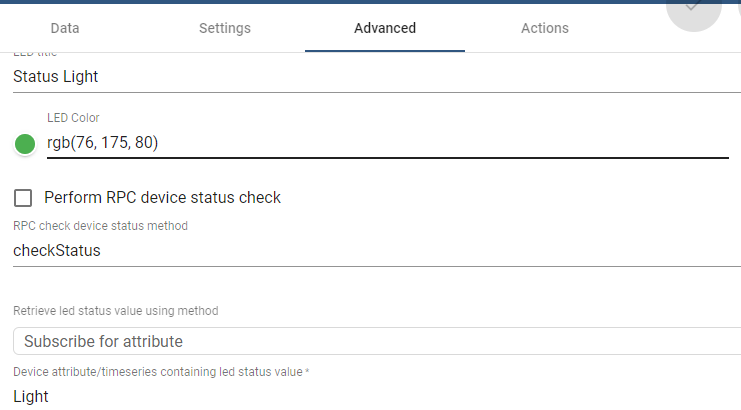
Khi người sử dụng điều khiển bật hoặc tắt trên giao diện web thì Thingsboard sẽ gửi vào topic “*v1/devices/me/rpc/request/+*” một message(thông điệp) có dạng như sau: *{“method”:”x”,”params”:y}* với :

* x: là giá trị điền vào ô số 3 trên hình trên
* y: là true hoặc false(true nếu bật false nếu tắt)

Thực chất khi đó Thingsboard server đang gửi một câu lệnh rpc(remote procedure call).

Và nó sẽ cập nhật trạng thái của nút nhấn trên giao diện dựa trên giá trị chúng ta điền vào ô số 2,và giá trị này phải được publish tới topic “*v1/devices/me/attributes*” nếu ô số 1 được chọn như hình. Nếu ở ô đó ta chọn “*timeseries*” thì giá trị đó cần publish tới topic “*v1/devices/me/telemetry*“

Với tiện ích LED Indicator, trong tab ADVANCED hãy chú ý tới một số phần như hình sau



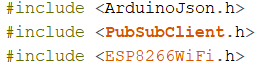
Trạng thái bật tắt của đèn này sẽ cập nhật dựa trên giá trị ta điền vào ô số 2 nếu như giá trị này được publish tới topic “*v1/devices/me/attributes* “.

## **1.5. Phát triển phần mềm**

### 1.5.1. Phát triển phần mềm trên arduino

Node mcu sẽ kết nối đến Thingsboard bằng giao thức MQTT. Nó sẽ subscribe vào topic sau : “*v1/devices/me/rpc/request/+*” để nhận được tín hiệu điều khiển khi người sử dụng tác động trên giao diện web. Sau đó dựa vào tín hiệu nhận được đó để điều khiển trạng thái các thiết bị. Sau đó sẽ publish lại trạng thái của các đèn vào topic “*v1/devices/me/attributes*” để cập nhật trạng thái của nút nhấn trên Thingsboard. Ban đầu node mcu sẽ gửi trạng thái của tất cả các thiết bị là tắt( false).

* Khai báo các thư viện cho chương trình:
* *ArduinoJson*: thư viện để giải nén các chuỗi json
* *PubSubClient*: thư viện publish và subscribe lên mqtt
* *ESP8266WiFi:* thư viện kết nối esp8266



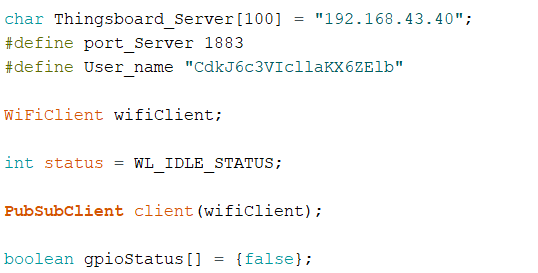
* Khai báo các chân nối với relay



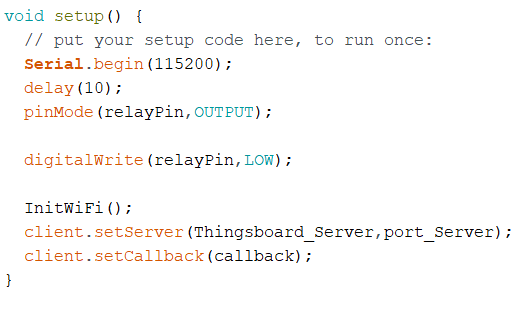
* Khai báo mật khẩu và tên wifi kết tới qua esp8266



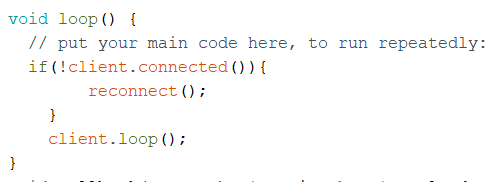
* Khai báo access token và địa chỉ server , port của thingsboard



* Hàm khai báo các chân tín hiệu ra, kết nối wifi, kết nối với thingsboard



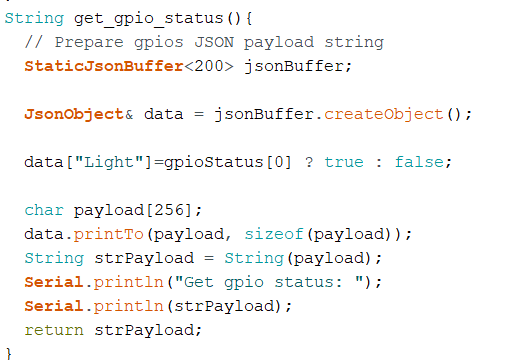
* Hàm lặp kết nối wifi



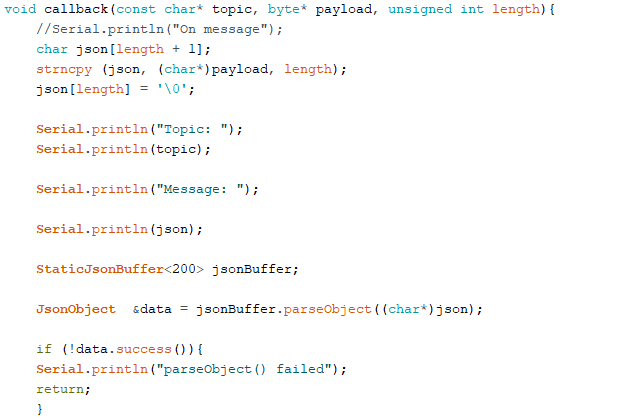
* Sau mỗi kết nối arduino sẽ lấy dữ liệu từ mqtt







* Sau mỗi lần lấy dữ liệu arduino sẽ so sánh và publish lại thông tin hiện tại light



Kiểm tra dữ liệu hàm Light và set lại giá trị tương ứng vào mảng gpioStatus



### 1.5.2. Phát triển phần mềm trên mobile

#### 1.5.2.1. Tổng quan về công nghệ

##### Ngôn ngữ lập trình Dart

Dart là ngôn ngữ lập trình đa mục đích ban đầu được phát triển bởi Google và sau đó được Ecma (ECMA-408) phê chuẩn làm tiêu chuẩn. Nó được sử dụng để xây dựng các ứng dụng web, server, máy tính để bàn và thiết bị di động. Dart là một ngôn ngữ hướng đối tượng, được xác định theo lớp, với cơ chế garbage-collected, sử dụng cú pháp kiểu C để dịch mã tùy ý sang JavaScript. Nó hỗ trợ interface, mixin, abstract, generic, static typing và sound type (2 cái cuối có thể hiểu là type-safe). Dart là ngôn ngữ mã nguồn mở và miễn phí, được phát triển trên [GitHub](https://github.com/dart-lang). Hiện nay Dart đã release phiên bản 2.7.

Các nhà phát triển tại Google và các nơi khác sử dụng Dart để tạo các ứng dụng chất lượng cao, quan trọng cho iOS, Android và web. Với các tính năng nhắm đến sự phát triển phía khách hàng, Dart rất phù hợp cho cả ứng dụng di động và web. Dart giúp bạn tạo ra những trải nghiệm đẹp, chất lượng cao trên tất cả các màn hình, với:

* Một ngôn ngữ được tối ưu hóa cho client
* Framework mạnh mẽ
* Công cụ linh hoạt

##### Framework Flutter

Flutter là SDK của Google, giúp tạo các ứng dụng di động cho iOS và Android bằng cách sử dụng một cơ sở mã (gần như vậy). Nó là một người mới trong phát triển ứng dụng di động đa nền tảng và không giống như các frameworks khác ví dụ React Native, nó không sử dụng JavaScript làm Ngôn ngữ lập trình.

Flutter bao gồm Reactive framework và công nghệ hiển thị 2D (2D rendering engine)và các công cụ phát triển(development tool). Các thành phần này làm việc cùng nhau giúp ta thiết kế, xây dựng, test, debug ứng dụng.

**Tại sao nên sử dụng Flutter ?**

Phát triển ứng dụng nhanh chóng: Tính năng hot reload của nó giúp bạn nhanh chóng và dễ dàng thử nghiệm, xây dựng giao diện người dùng, thêm tính năng và sửa lỗi nhanh hơn. Trải nghiệm tải lại lần thứ hai, mà không làm mất trạng thái, trên emulator, simulator và device cho iOS và Android.

UI đẹp và biểu cảm: Thỏa mãn người dùng của bạn với các widget

built-in đẹp mắt theo Material Design và Cupertino (iOS-flavor), các API chuyển động phong phú, scroll tự nhiên mượt mà và tự nhận thức được nền tảng.

##### Bộ công cụ nhận dạng giọng nói Vosk API

Vosk là một bộ công cụ nhận dạng giọng nói mã nguồn mở ngoại tuyến. Nó cung cấp các mô hình nhận dạng giọng nói cho 16 ngôn ngữ: tiếng Anh, tiếng Anh Ấn Độ, tiếng Đức, tiếng Pháp, tiếng Tây Ban Nha, tiếng Bồ Đào Nha, tiếng Trung, tiếng Nga, tiếng Thổ Nhĩ Kỳ, tiếng Việt, tiếng Ý, tiếng Hà Lan, tiếng Catalan, tiếng Ả Rập, tiếng Hy Lạp, tiếng Farsi.

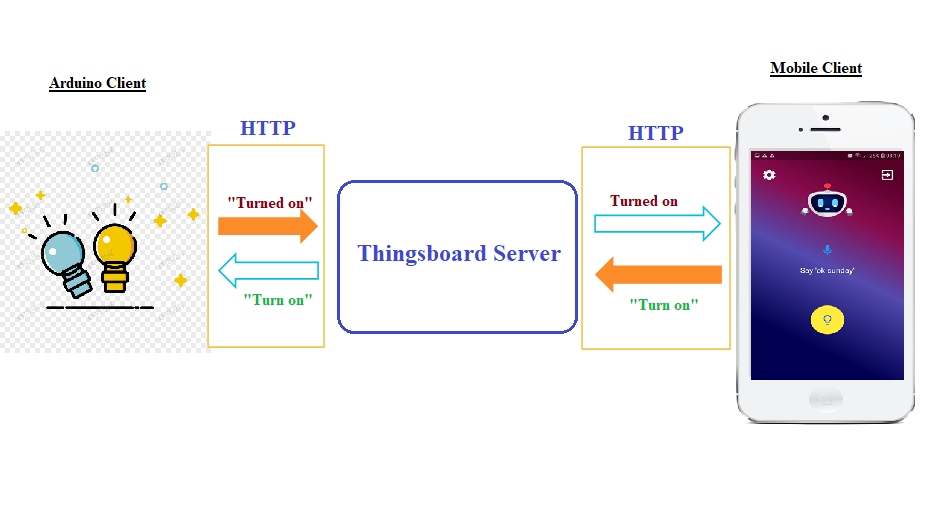
Các mô hình Vosk nhỏ (50Mb) nhưng cung cấp phiên âm từ vựng lớn liên tục, phản hồi không có độ trễ với API phát trực tuyến, từ vựng có thể cấu hình lại và nhận dạng người nói.

Các ràng buộc nhận dạng giọng nói được triển khai cho các ngôn ngữ lập trình khác nhau như Python, Java, Node.JS, C #, C ++ và các ngôn ngữ khác.

Vosk cung cấp tính năng nhận dạng giọng nói cho chatbots, thiết bị gia dụng thông minh, trợ lý ảo. Nó cũng có thể tạo phụ đề cho phim, phiên âm cho các bài giảng và phỏng vấn.

Vosk mở rộng quy mô từ các thiết bị nhỏ như Raspberry Pi hoặc điện thoại thông minh Android đến các cụm lớn.

#### 1.5.2.2. Sơ đồ giao tiếp



* Thiết bị điện đóng vai trò là một subscriber tới thingsboard. Nó sẽ subscribe để nhận dữ liệu điều khiển. Khi nhận được dữ liệu, bộ xử lý sẽ thực hiện hành động tương ứng.
* Mobile client gửi các dữ liệu điều khiển tới Thingsboard server để từ đó thiết bị điện lấy dữ liệu.

#### 1.5.2.2. Phân tích, thiết kế ứng dụng điều khiển

###### 1.5.2.2.1. Yêu cầu

* Xây dựng ứng dụng điều khiển thiết bị điện trên cả 2 nền tảng di động lớn là Android và iOS (Demo trên Android).
* Sử dụng giao thức HTTP để truyền nhận dữ liệu.
* Giao diện trực quan, dễ sử dụng.
* Chức năng của ứng dụng: Điều khiển bóng đèn bật/tắt tùy ý.

###### 1.5.2.2.2. Đặc tả bài toán

Hiện nay với xu hướng công nghệ hóa, các hệ thống điều khiển thiết bị đã rất phổ biến, nhất là các hệ thống về IOT. Các hệ thống đó sử dụng nhiều công nghệ giao tiếp đặc thù khác nhau như HTTP/HTTPS, Bluetooth, TCP IP, MQTT…

Để đơn giản hóa và thuận tiện cho việc điều khiển, ta sẽ sử dụng điện thoại thông minh thay thế cho các thiết bị điều khiển vật lý và các phần mềm điều khiển trên máy vi tính.

Áp dụng vào hệ thống này, phần mềm điều khiển thiết bị điện sẽ có thể điều khiển thiết bị theo ý muốn của người sử dụng ở mọi lúc, mọi nơi, mọi thời điểm, chỉ cần có kết nối Internet.

###### 1.5.2.2.3. Đặc tả chức năng

Hệ thống hiện tại sẽ tập trung vào chức năng điều khiển thiết bị theo ý muốn của người dùng, nên ứng dụng cũng chỉ có 2 đặc tả chức năng chính đó là: Điều khiển bằng nút và điều khiển qua giọng nói.

* **Chức năng điều khiển bằng nút**
* Mô tả chức năng: Chức năng này cho phép người dùng điều khiển bật/tắt thiết bị điện bằng các nút hiển thị trên màn hình điện thoại.
* Mô tả hoạt động:
* Khi người dùng mở ứng dụng, màn hình sẽ hiển thị các nút điều khiển tương ứng với các hành động bật/tắt của thiết bị.
* Khi ấn vào nút điều khiển, ứng dụng sẽ gửi tín hiệu điều khiển đến hệ thống thiết bị để thực hiện hành động mà người dùng đã chọn.
* **Chức năng điều khiển bằng giọng nói**
* Mô tả chức năng: Chức năng này cho phép người dùng ngoài điều khiển thiết bị bằng giọng nói (tiếng Anh) theo các câu lệnh điều khiển đã được định nghĩa.
* Mô tả hoạt động:
* Ứng dụng sẽ có nút bật/tắt chế độ điều khiển bằng giọng nói.
* Khi chức năng điều khiển giọng nói được bật, ứng dụng sẽ chuyển sang chế độ “Sleep”, lắng nghe câu lệnh “wake-up”.
* Sẽ có câu lệnh “Wake-up” để khởi động chế độ lắng nghe mệnh lệnh.
* Sau khi được wake-up, ứng dụng chuyển sang chế độ lắng nghe mệnh lệnh trong vòng 5 giây.
* Nếu người dùng nói ra lệnh điều khiển, ứng dụng sẽ gửi tín hiệu điều khiển tương ứng tới thiết bị, và hệ thống sẽ xử lý sẽ thực hiện hành động đó.
* Nếu trong 5 giây lắng nghe lệnh điều khiển, người dùng không nói lệnh điều khiển, ứng dụng sẽ không gửi tín hiệu điều khiển nào.
* Qua 5 giây, ứng dụng sẽ trở lại chế độ “Sleep”, lắng nghe lệnh “Wakeup”.

### 1.5.3. Phát triển phần mềm trên web(Hiếu)

# **CHƯƠNG 2: TRIỂN KHAI**

## **2.1. Lịch trình thực hiện**

| **Tuần/Ngày** | **Nội dung** | **Xác nhận** |
| --- | --- | --- |
| Tuần 1  (02/11/2020 đến 08/11/2020) | - Chọn đề tài |  |
| Tuần 2  (09/11/2020 đến 15/11/2020) | - Viết đề cương chi tiết  - Trình bày phương án  - Phân chia công việc cho từng thành viên |  |
| Tuần 3  (16/11/2020 đến 22/11/2020) | - Tìm hiểu hoạt động, nguyên lý làm việc và test thử các module( board arduino, ThingsBoard, ...) |  |
| Tuần 4  (23/11/2020 đến 29/11/2020) | - Kết nối board arduino với các module, kiểm tra độ chạy  - Lắp ráp các linh kiện lên mô hình  - Tạo cấu hình trên ThingsBoard |  |
| Tuần 5  (30/11/2020 đến 06/12/2020) | - Tìm hiểu các ngôn ngữ viết chương trình điều khiển và các tool phát triển  - Nghiên cứu các giao thức kết nối ThingsBoard |  |
| Tuần 6  (07/12/2020 đến 13/12/2020) | - Nghiên cứu, thiết kế cơ bản về sơ đồ nguyên lý, cách thức hoạt động  - Nghiên cứu, thiết kế cơ bản về sơ đồ luồng điều khiển |  |
| Tuần 7  (14/12/2020 đến 20/12/2020) | - Viết chương trình điều khiển qua arduino, nạp code và chạy thử |  |
| Tuần 8  (21/12/2020 đến 27/12/2020) | - Viết chương trình điều khiển qua mobile, chạy thử |  |
| Tuần 9  (28/12/2020 đến 03/01/2021) | - Viết chương trình điều khiển qua web, chạy thử |  |
| Tuần 10  (04/01/2021 đến 09/01/2021) | - Lập trình hoàn chỉnh  - Cân chỉnh và tối ưu sản phẩm |  |
| Tuần 11  (10/01/2021 đến 16/01/2021) | - Viết và chỉnh sửa báo cáo  - In báo cáo  - Bảo vệ đề tài |  |

## **2.2. Thực nghiệm và kết luận**

### 2.2.1. Mô tả thực nghiệm

Cấp nguồn cho thiết bị và tiến hành thực hiện điều khiển thiết bị thông qua web và ứng dụng di động. Kiểm tra tính chính xác và ổn định của hệ thống.

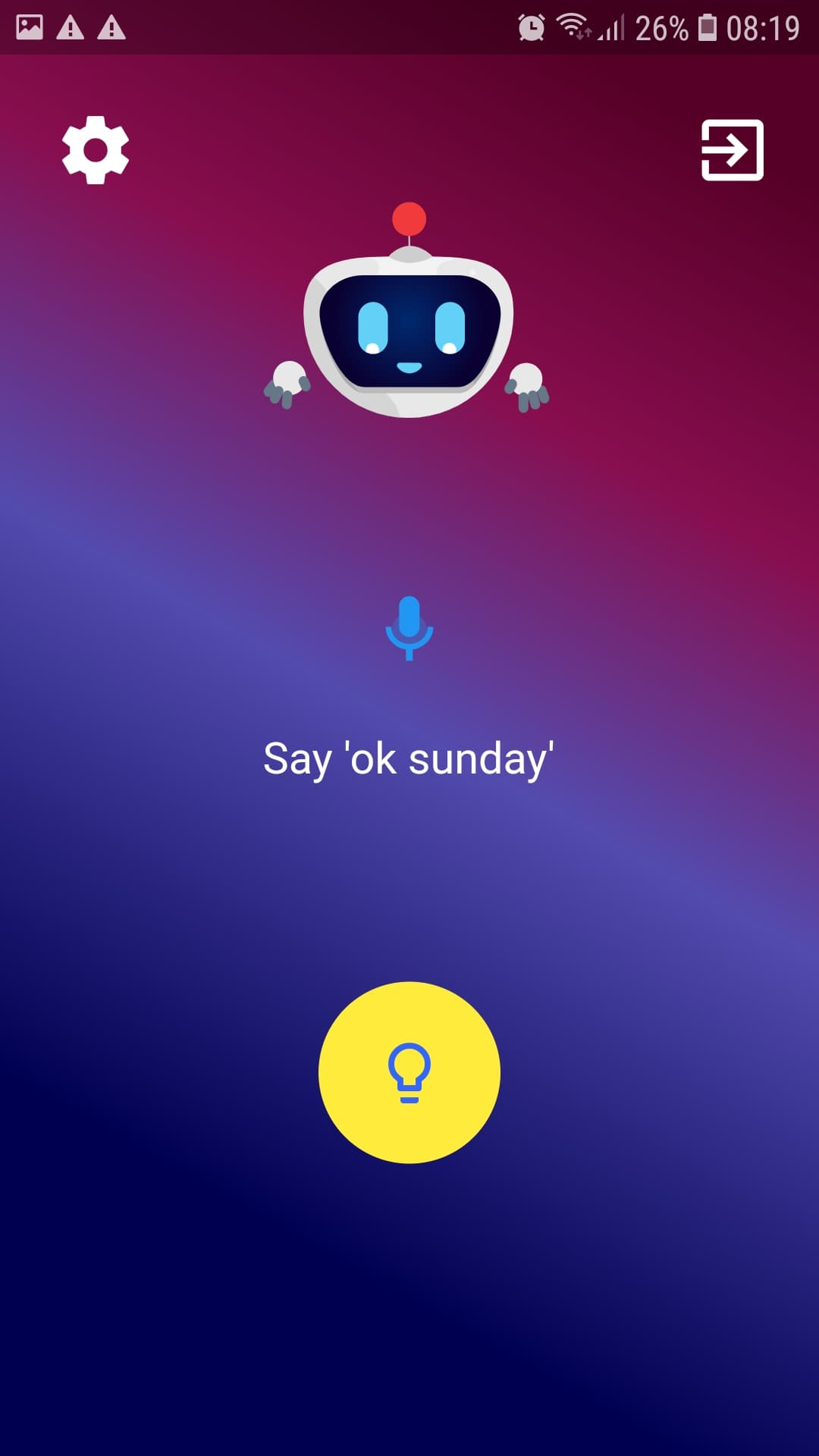
### 2.2.2. Kết quả, nhận xét và đánh giá

#### 2.2.2.1. Kết quả làm được

**Web**

* Server : Đã hoạt động ổn định nhận được các trạng thái của các thiết bị gửi đến qua Broker một các ổn định và liên tục
* Client : Điều khiển được các thiết bị và kiểm tra luôn được trạng thái của chúng thông qua đèn màu và nút điều khiển Switch ( ví dụ nếu đèn bật thì button đang ở nút On ,còn đèn đang tắt thì button đang ở nút Off ,đèn màu cũng sẽ hiện màu sáng /tối tương tự với trạng thái bật / tắt của thiết bị )
* Tổng quan : Hệ thống hoạt động khá ổn định có thể sử dụng một cách dễ dàng với giao diện dễ sử dụng

**Mobile**



* UI: Đã xây dựng hai màn hình giao diện cho phép người dùng tương tác, đồng thời cũng hiển thị rõ các giá trị và hình ảnh trả về khi người dùng nhận được kết quả tương tác.
* Back-end: Các hàm xử lý đã gửi request thành công đến server để điều khiển thiết bị trong hệ thống thông qua ứng dụng và trả về kết quả để thực hiện thay đổi bên phía giao diện. Ngoài ra, cũng đã truy cập thành công vào phần cứng thiết bị (micro) để lấy và xử lý giọng nói nhận được từ người dùng, từ đó cho phép người dùng có thể điều khiển thông qua giọng nói.
* Tổng quan: Hệ thống dễ tương tác và sử dụng, đồng thời cũng đã hoạt động một cách ổn.

#### 2.2.2.2. Kết quả thực nghiệm

* Web : Hoạt động ổn định , dễ sử dụng . Khi điều khiển thì bóng điện ,cửa,..nhận tín hiệu một cách khá nhanh ,đồng thời khả năng cập nhật trạng thái có độ delay thấp và liên tục cập nhật được trạng thái của các thiết bị . Có khả năng điều khiển tương tự Dashboard của Broker Thingsboard
* App : Hoạt động tốt. Trong điều kiện mạng ổn định, có thể điều khiển rất nhanh, khá trực quan và tiện lợi cho người sử dụng.

#### 2.2.2.3. Nhận xét

Qua 11 tuần thực hiện, nhóm đã hoàn thành hệ thống đáp ứng cơ bản những yêu cầu ban đầu đặt ra. Dưới đây là những nhận xét:

* Ưu điểm
  + Kết nối thành công giữa các module
  + Hệ thống hoạt động ổn định qua nhiều lần thử nghiệm
  + Giao diện điều khiển trực quan
* Nhược điểm
  + Hệ thống phục thuộc vào tốc độ wifi
  + Hệ thống phục thuộc vào nguồn 220V

#### 2.2.2.4. Đánh giá

Trong quá trình vận hành hệ thống, nhóm đã ghi nhận lại kết quả được tổng hợp

| Công việc | Số lần thao tác | Số lần thành công | Thời gian đáp ứng | Đánh giá |
| --- | --- | --- | --- | --- |
| Điều khiển thiết bị qua smartPhone | 22 | 22 | 1-2 giây | Tốt |
| Điều khiển thiết bị qua website | 23 | 21 | 2 -3 giây | Tốt |
| Cập nhật trạng thái thiết bị | 20 | 20 | 1-2 giây | Tốt |
| **Đánh giá chung** | | | | Tốt |

Qua các số liệu được thống kê ở bảng trên, nhóm đánh giá hệ thống về cơ bản đã đáp ứng mục tiêu đặt ra. Hệ thống hoạt động ổn định qua nhiều lần chạy. Mô hình đảm bảo tính an toàn.

### 2.2.3. Hướng phát triển

* Thêm nhiều thiết bị phù hợp
* In mạch điều khiển smarthome để nhìn đẹp mắt, dễ di chuyển, lắp đặt
* Phát triển thêm các options thiết thực và tiện dụng hơn trên phần mềm.

# **CHƯƠNG 3: HƯỚNG DẪN SỬ DỤNG**

# **CHƯƠNG 4: MÔ TẢ CODE**

## **4.1. Code trên arduino**

#include <ArduinoJson.h>

#include <PubSubClient.h>

#include <ESP8266WiFi.h>

#define relayPin D2 //do

#define WIFI\_AP "nokia110i"

#define WIFI\_PASSWORD "12345678"

char Thingsboard\_Server[100] = "192.168.43.40";

#define port\_Server 1883

#define User\_name "CdkJ6c3VIcllaKX6ZElb"

WiFiClient wifiClient;

int status = WL\_IDLE\_STATUS;

PubSubClient client(wifiClient);

boolean gpioStatus[] = {false};

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

delay(10);

pinMode(relayPin,OUTPUT);

digitalWrite(relayPin,LOW);

InitWiFi();

client.setServer(Thingsboard\_Server,port\_Server);

client.setCallback(callback);

}

void loop() {

// put your main code here, to run repeatedly:

if(!client.connected()){

reconnect();

}

client.loop();

}

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

//Serial.println("On message");

char json[length + 1];

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

json[length] = '\0';

Serial.println("Topic: ");

Serial.println(topic);

Serial.println("Message: ");

Serial.println(json);

StaticJsonBuffer<200> jsonBuffer;

JsonObject &data = jsonBuffer.parseObject((char\*)json);

if (!data.success()){

Serial.println("parseObject() failed");

return;

}

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

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

if(data["params"]==true){

digitalWrite(relayPin,LOW);

gpioStatus[0]=true;

}

else{

digitalWrite(relayPin,HIGH);

gpioStatus[0]=false;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

}

void reconnect(){

// Loop until we're reconnected

while (!client.connected()){

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");

}

Serial.print("Connecting to Thingsboard node ...");

// Attempt to connect (clientId, username, password)

if(client.connect("TESTRELAY",User\_name,NULL)){

Serial.println( "[DONE]" );

client.subscribe("v1/devices/me/rpc/request/+");//dang ky nhan cas lenh rpc tu cloud

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

}

else{

Serial.print( "[FAILED] [ rc = " );

Serial.print( client.state());

Serial.println(" : retrying in 5 seconds]");

// Wait 5 seconds before retrying

delay( 5000 );

}

}

}

String get\_gpio\_status(){

// Prepare gpios JSON payload string

StaticJsonBuffer<200> jsonBuffer;

JsonObject& data = jsonBuffer.createObject();

data["Light"]=gpioStatus[0] ? true : false;

char payload[256];

data.printTo(payload, sizeof(payload));

String strPayload = String(payload);

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

Serial.println(strPayload);

return strPayload;

}

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");

}

## **4.2. Code trên mobile**

### 4.2.1. Code login

class AuthenticationProvider {

Client client;

AuthenticationProvider(this.client);

Future<Token> login(Account account) async {

try {

var response = await client.post(URLConstants.*login*,

headers: {"Content-Type": "application/json"},

body: jsonEncode(account.toJson()));

print(URLConstants.*login*);

print(response.body);

print(account.toJson());

if (response.statusCode == 200) {

var jsonResponse = jsonDecode(response.body);

return Token.fromJson(jsonResponse);

} else {

return null;

}

} catch (e) {

throw e;

}

}

}

### 4.2.2. Code giao tiếp điều khiển HTTP

class ControlDeviceProvider {

final Client client;

ControlDeviceProvider(this.client);

Future<bool> sendRPCDeviceRequest(String token, String method, dynamic params,

{String id = "3e85b630-28a4-11eb-85ee-f936949cce2a"}) async {

try {

var response = await client.post("${URLConstants.*rpcOneWay*}/$id",

headers: {

"Content-Type": "application/json",

"X-Authorization": "Bearer $token"

},

body: jsonEncode({"method": method, "params": params}));

print(response.body);

if (response.statusCode == 200) {

return true;

} else {

return false;

}

} catch (e) {

throw e;

}

}

}

## **4.3. Code trên web**

### 4.3.1. Phần login.

* Login vào thingboard qua HTTP bằng cách gửi lên thingboard tài thông tin mật khẩu và tên đăng nhập

export const login = (username, password, callback) => {

axios.post(authLoginUrl, { username, password }).then(callback).catch(error => {

console.log("Error with login: ", error)

alert('Login failse')

})

}

export const logout = (callback) => {

axios.post(authLogoutUrl).then(callback).catch(error => {

console.log(

"Error with logout: ", error)

})

}

* Lấy lại thông tin mã xác thực:

const loginApp = (username, password, callback = loginCallback) => {

login(username, password, callback)

setLogin(true)

}

### 4.3.2. Điều khiển thiết bị qua rpc api:

* Khi điều khiển ta gửi lên thông tin device key, mã xác thực thiết bị

export const remoteDevice = (deviceKey, authToken, deviceName, callback) => {

const rpcDeviceUrl = `${rpcRomte}${deviceKey}`

axios.post(rpcDeviceUrl, { params: `${deviceName}` }, {

headers: {

'Content-Type': 'application/json',

'X-Authorization': `Bearer ${authToken}`

},

}).then(callback).catch(error => {

console.log(

"Error with rpc: ", error)

})

}