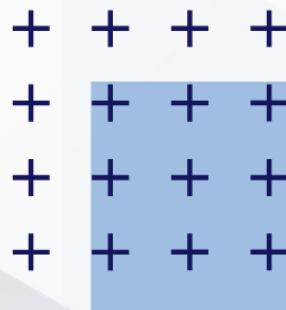
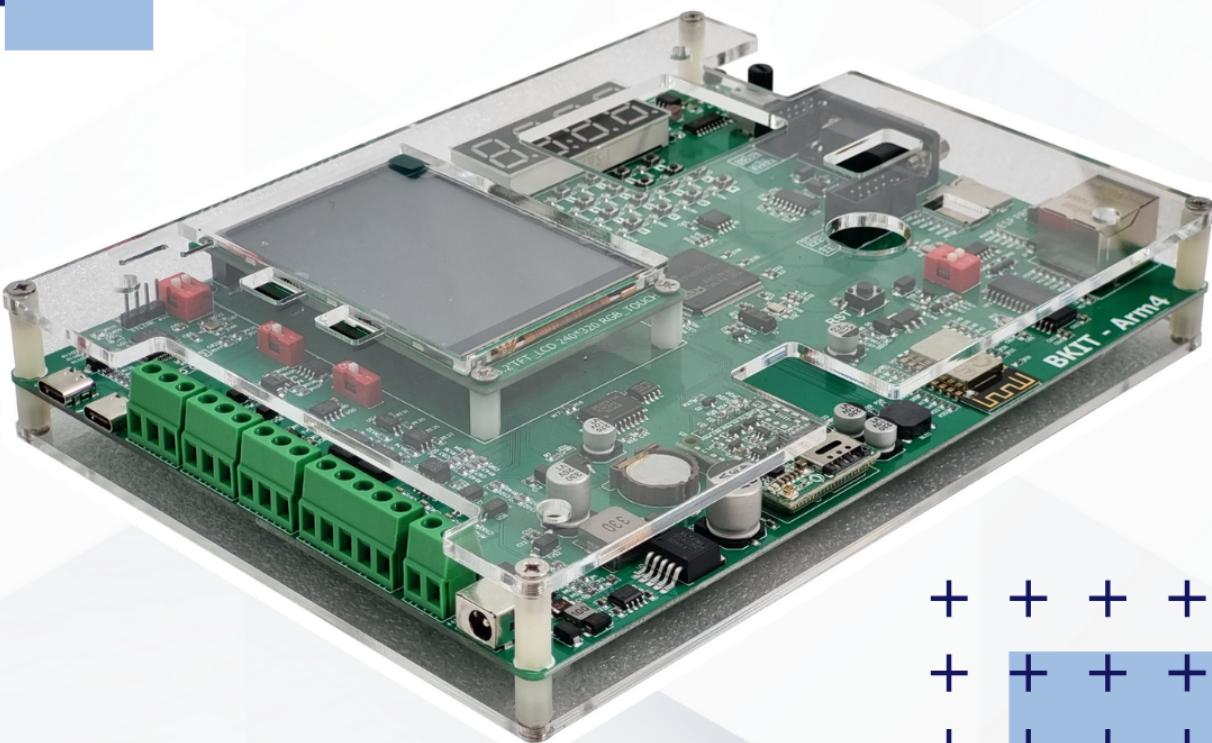


KIT THÍ NGHIỆM BKIT

ARM4

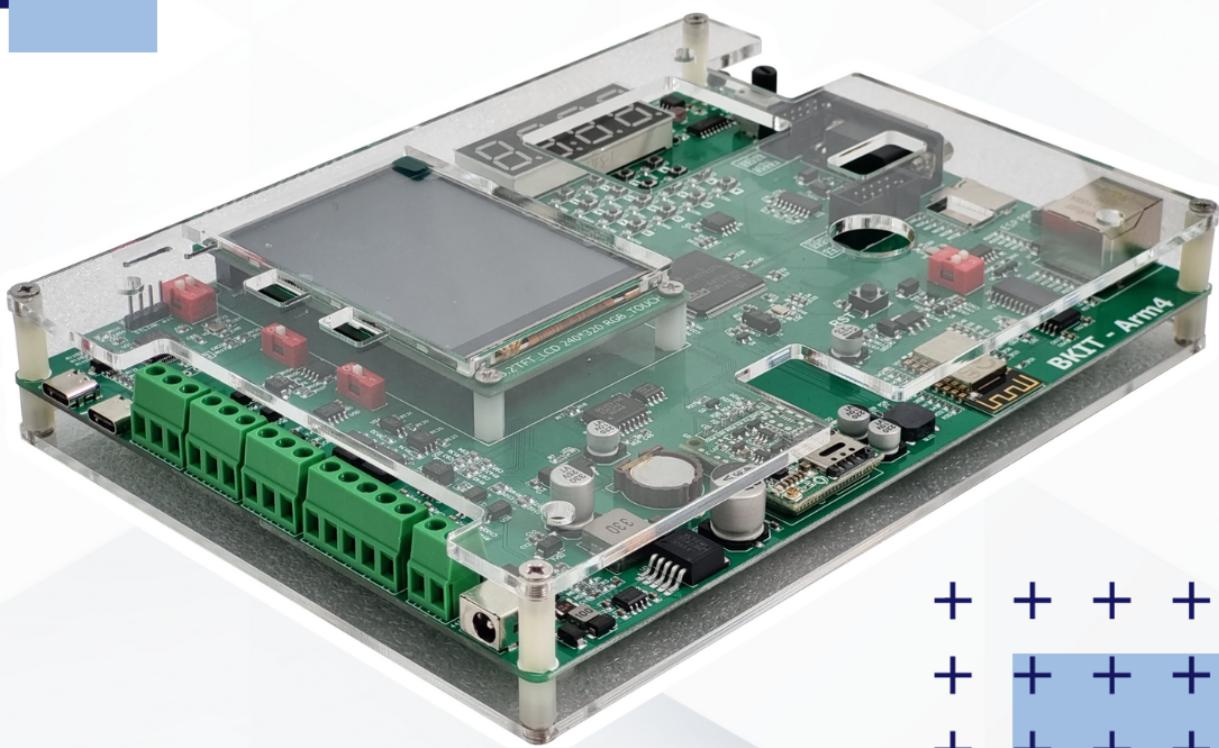


Mục lục

Chapter 8. ESP8266 - WIFI	3
1 Mục tiêu	4
2 Giới thiệu	4
3 Cơ sở lý thuyết	5
3.1 Giới thiệu về ESP8266	5
3.2 Kiến trúc 5 lớp của IoT	6
4 Hướng dẫn cấu hình	7
5 Hướng dẫn lập trình	8
5.1 Hướng dẫn lập trình ESP8266 trên Arduino IDE	9
5.2 Hướng dẫn làm việc với Adafruit	15
5.2.1 Tạo tài khoản và kênh dữ liệu	15
5.2.2 Tạo dashboard và kết nối với feed	17
5.3 Hướng dẫn thư viện STM	29
6 Bài tập và báo cáo	35

CHƯƠNG 8

ESP8266 - WIFI



1 Mục tiêu

- Tìm hiểu về ESP8266 ESP-12 của Ai-Thinker.
- Hướng dẫn sử dụng ESP8266 trên mạch và biết cách giao tiếp với chip chính.
- Kết nối Wifi và điều khiển thông qua Adafruit bằng giao thức MQTT.

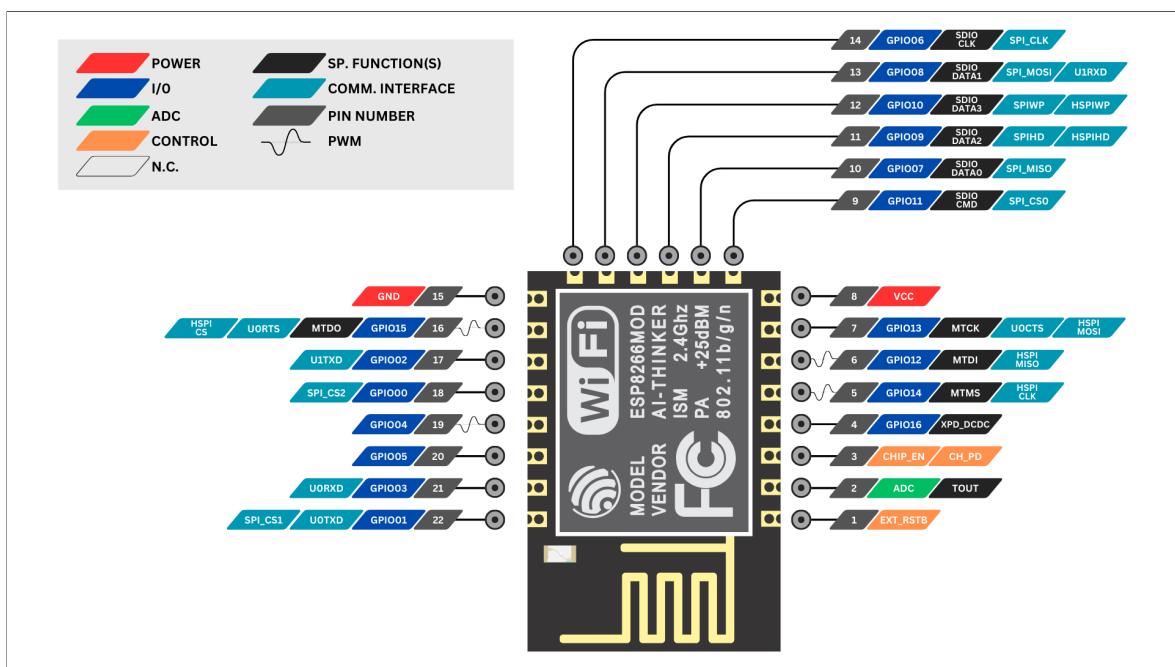
2 Giới thiệu

- Mạch thu phát Wifi ESP8266 ESP-12E của Ai-Thinker là một giải pháp Wifi tiết kiệm và hiệu quả cho các ứng dụng liên quan đến Internet và Wifi. Nó tích hợp thêm bộ khuếch đại công suất (PA) và bộ lọc tín hiệu RF (LNA), cải thiện khả năng thu phát sóng Wifi. Mạch có kích thước nhỏ gọn, chất lượng đáng tin cậy, vỏ kim loại chống nhiễu và anten Wifi PCB tích hợp. ESP8266 cũng hỗ trợ giao tiếp SPI, I2C, UART, ADC, PWM và các chức năng GPIO.
- Adafruit.io là một dịch vụ đám mây (cloud service) do Adafruit cung cấp, được thiết kế đặc biệt để hỗ trợ việc kết nối và quản lý dữ liệu từ các thiết bị IoT. Nó cung cấp giao diện đơn giản để thu thập, lưu trữ và truy xuất dữ liệu từ các cảm biến và thiết bị IoT, giúp người dùng dễ dàng tạo và theo dõi các ứng dụng IoT của mình. Trong bài lab này chúng ta sẽ sử dụng giao thức MQTT để giao tiếp giữa Adafruit.io và ESP8266. MQTT là một giao thức truyền thông nhẹ được sử dụng trong IoT, cho phép truyền dữ liệu giữa các thiết bị và máy chủ một cách hiệu quả thông qua cơ chế publish/subscribe.

3 Cơ sở lý thuyết

3.1 Giới thiệu về ESP8266

- Chip ESP8266EX sử dụng CPU 32-bit Tensilica Xtensa LX106, được tối ưu hóa cho các ứng dụng IoT. Bộ nhớ Flash tích hợp trực tiếp vào chip, giúp ESP8266EX lưu trữ chương trình và dữ liệu một cách hiệu quả. ESP-12E có 22 chân GPIO cho phép kết nối với nhiều thiết bị ngoại vi khác nhau. Mỗi chân có thể được cấu hình để thực hiện nhiều chức năng khác nhau, như GPIO, UART, SPI, I2C, ADC, PWM có thể kết nối với nhiều module khác.



Hình 8.1: Sơ đồ chân của ESP8266 ESP-12E

- ESP-12E hỗ trợ các chuẩn Wi-Fi phổ biến, bao gồm 802.11 b/g/n, giúp nó kết nối với nhiều loại mạng Wifi. Ngoài ra, nó có thể hoạt động ở chế độ Station (kết nối với Access Point), Access Point (tạo mạng Wifi), hoặc cả hai. ESP-12E thường đi kèm với Flash có dung lượng từ 4MB đến 16MB, cung cấp không gian lưu trữ đủ cho chương trình và dữ liệu. Sử dụng RAM để lưu trữ dữ liệu trong quá trình chạy chương trình.
 - Chế Độ Ngủ và Tiết Kiệm Năng Lượng:
 - Trong chế độ ngủ: ESP8266EX tiêu thụ rất ít năng lượng, thích hợp cho các ứng dụng yêu cầu tiết kiệm năng lượng cao.

- Trong chế độ tiết kiệm năng lượng: Cắt giảm một số chức năng để giảm tiêu thụ năng lượng, nhưng vẫn duy trì khả năng đánh thức nhanh. ESP8266 có thể được lập trình bằng Arduino IDE, môi trường lập trình phổ biến, giúp người dùng dễ dàng phát triển ứng dụng IoT. Bên cạnh đó, ESP8266 ESP-12E có sẵn nhiều thư viện hỗ trợ giúp lập trình viên tận dụng mọi khả năng của ESP8266.

3.2 Kiến trúc 5 lớp của IoT

- Theo tiến sĩ Timothy Chou, giảng viên Đại học Stanford, kiến trúc về ứng dụng thông minh dựa trên IoT được chia thành mô hình 5 lớp như sau:



Hình 8.2: Kiến trúc 5 tầng của IoT

- Chức năng chính của từng lớp trong kiến trúc này được khái quát như sau:
 - Things: Các thiết bị trong ứng dụng giám sát là rất đa dạng về chức năng và số lượng. Tùy vào ứng dụng, nhiều loại cảm biến khác nhau sẽ được sử dụng. Các nút cảm biến chủ yếu sử dụng giao tiếp không dây.
 - Connect: Chúng tôi thu thập dữ liệu từ các nút cảm biến. Đôi với mỗi loại ứng dụng, có nhiều tiêu chuẩn kết nối khác nhau, đòi hỏi lớp này phải hỗ trợ nhiều loại kết nối, từ Zigbee và Wifi trong các ứng dụng nhà thông minh, đến các giao tiếp rộng hơn như LoRa hay 3G/4G.
 - Collect: Dữ liệu thu thập được sẽ được gửi lên các máy chủ tập trung để lưu trữ. Tại đây, một lượng lớn dữ liệu sẽ được gửi về, đặt ra thách thức lớn cho máy chủ và cần áp dụng công nghệ Big Data.
 - Learn: Lớp này có nhiệm vụ lọc ra các thông tin đặc trưng, có ý nghĩa cụ thể cho từng loại ứng dụng. Công nghệ Machine Learning và hiện tại là Deep Learning sẽ được sử dụng ở đây.

- Do: Dựa vào các thông tin đặc trưng, hệ thống sẽ xây dựng các quy luật thích nghi với môi trường và đưa ra các quyết định cho hệ thống. Sự thực thi của mỗi quyết định sẽ được đo lường tự động, và sai lệch so với mục tiêu tối ưu sẽ được xem xét cho lần sau. Theo cách này, hệ thống sẽ tự học kinh nghiệm để ngày càng hoàn thiện hơn.

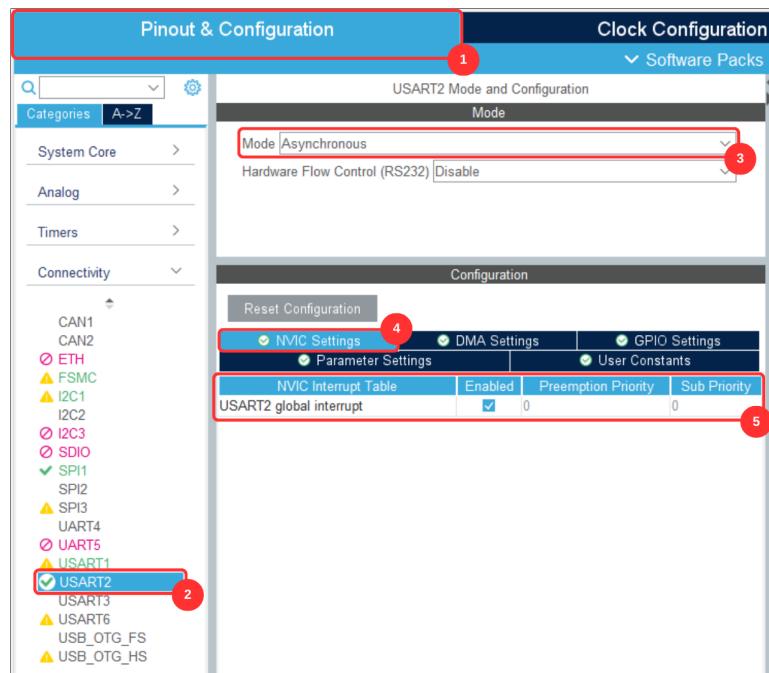
4 Hướng dẫn cấu hình

Trong Kit thí nghiệm, vi điều khiển sẽ cần phải giao tiếp với ESP8266 thông qua USART2 để truyền nhận dữ liệu. Các chân được cấu hình như bảng sau:

Ngoại vi	Chân vi điều khiển	Chức năng
ESP_POWER	PF10	GPIO Output
ESP_BUSY	PF9	GPIO Input
ESP_TX	PA3	USART2_RX
ESP_RX	PA2	USART2_TX

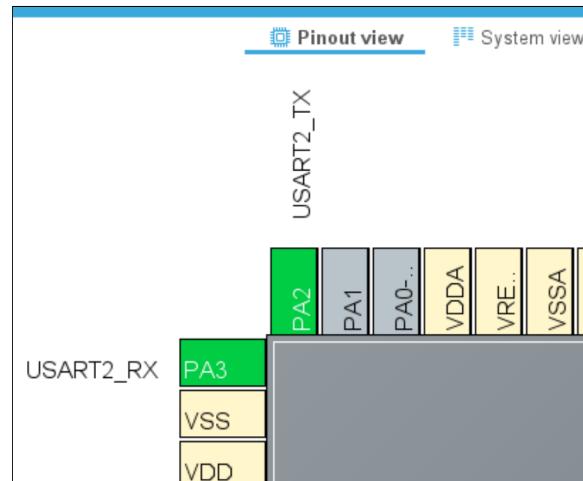
Bảng 8.1: Kết nối ngoại vi

Để chúng giao tiếp được với nhau, theo datasheet ta sẽ config USART2 như sau:



Hình 8.3: Config USART2

Sau khi config, ta có pinview out như hình sau:



Hình 8.4: Pinout view của USART2 sau khi config

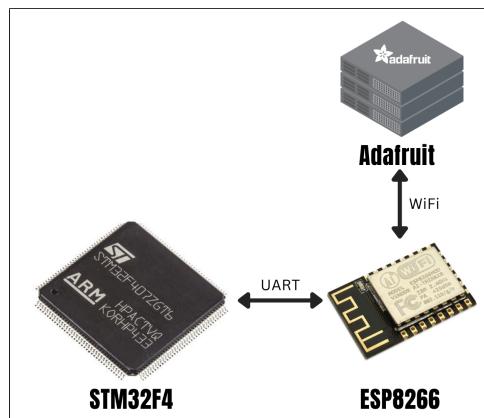
Hai chân ESP12_BUSY và ESP12_PWR sau khi được config theo bảng 8.1 sẽ có pin-view out như hình sau:



Hình 8.5: Config hai chân ESP12_BUSY và ESP12_PWR

5 Hướng dẫn lập trình

Trong bài lab này, chúng ta sẽ gửi tín hiệu chớp tắt đèn bằng nút nhấn hoặc từ server Adafruit theo mô hình sau:

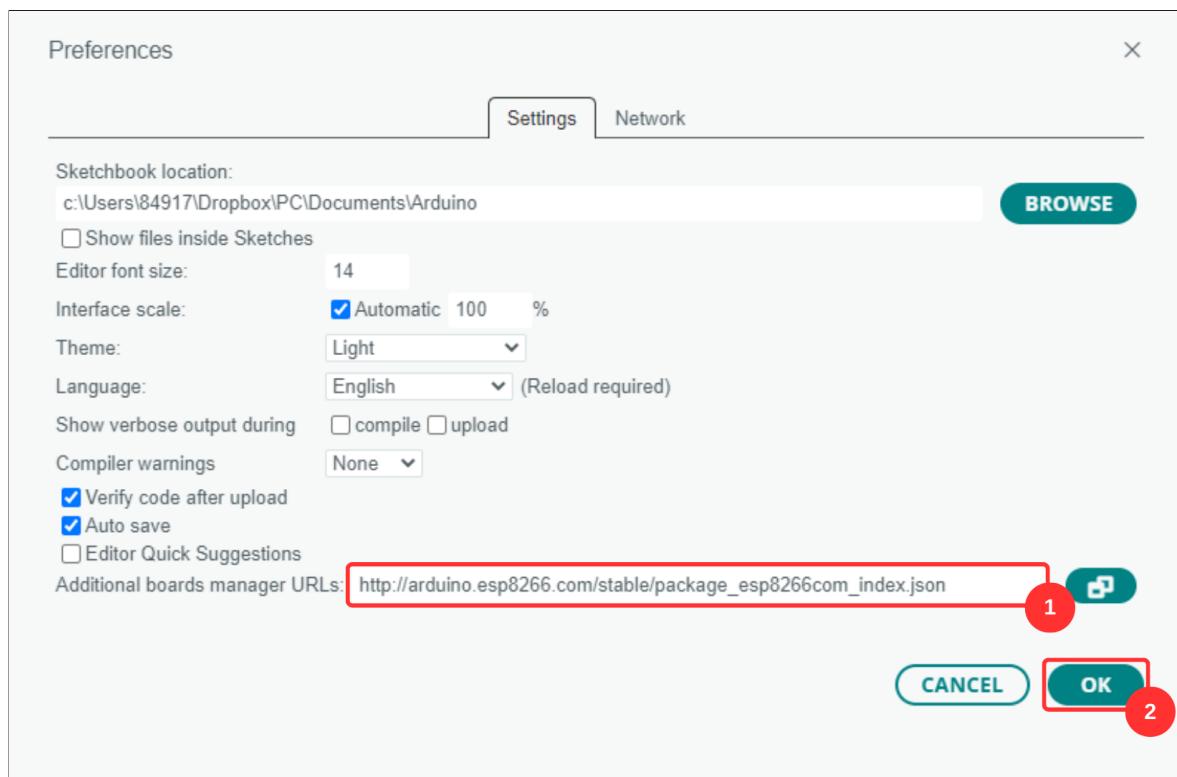


Hình 8.6: Mô hình của bài hướng dẫn

5.1 Hướng dẫn lập trình ESP8266 trên Arduino IDE

Trong phần này, chúng ta sẽ lập trình cho ESP8266 bằng Arduino IDE, bạn có thể tải IDE tại đây.

Ta sẽ chọn board **Generic ESP8266 Module**. Nếu không có sẵn board trên, bạn có thể thêm board bằng cách: File → Preferences và thêm url: http://arduino.esp8266.com/stable/package_esp8266com_index.json (hình 8.7).



Hình 8.7: Thêm board ESP8266

Dưới đây là cấu trúc chương trình khi lập trình trên Arduino IDE. Phần **setup** sẽ được dùng để thêm các đoạn code khởi tạo, chỉ chạy một lần. Phần **loop** sẽ chứa các đoạn code được lặp lại liên tục.

```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run repeatedly:  
7 }
```

Program 8.1: Cấu trúc chương trình



Hình 8.8: LED xanh được kết nối với chân GPIO2 trên ESP8266

Để kiểm tra hoạt động của ESP, ta sẽ thực hiện chương trình nháy LED. Trên ESP8266 có một đèn LED màu xanh được kết nối với chân GPIO2 (hình 8.8). Ta cần sử dụng 2 hàm cơ bản là **pinMode** để cấu hình chân và **digitalWrite** để ghi tín hiệu ra chân Output. Ngoài ra, ở cuối **loop** ta sẽ có câu lệnh **delay(10)**. Như vậy các câu lệnh ở trong loop sẽ được hiện thực mỗi 10ms. Tương tự như cách lập trình trên STM32, ta thêm biến đếm để thực hiện các công việc có chu kì lớn hơn. Đoạn code nhấp nháy LED mỗi 1s sẽ được minh họa dưới đây.

```

1 int led_counter = 0;
2 int led_status = HIGH;
3 void setup() {
4     // put your setup code here, to run once:
5     //set pin 2 as OUTPUT
6     pinMode(2, OUTPUT);
7 }
8
9 void loop() {
10    // put your main code here, to run repeatedly:
11    led_counter++;
12    if(led_counter == 100){
13        // every 1s
14        led_counter = 0;
15        //toggle LED
16        if(led_status == HIGH) led_status = LOW;
17        else led_status = HIGH;
18
19        digitalWrite(2, led_status);
20    }

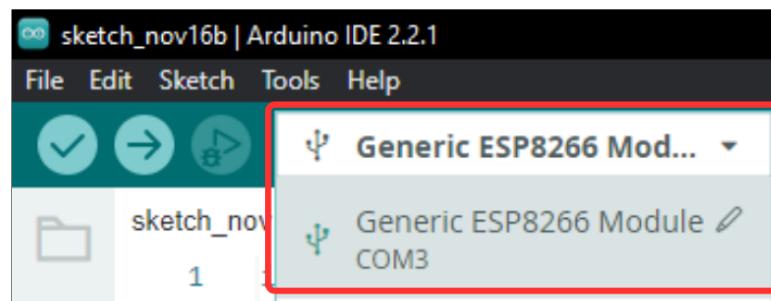
```

```

21
22     delay(10);
23 }
```

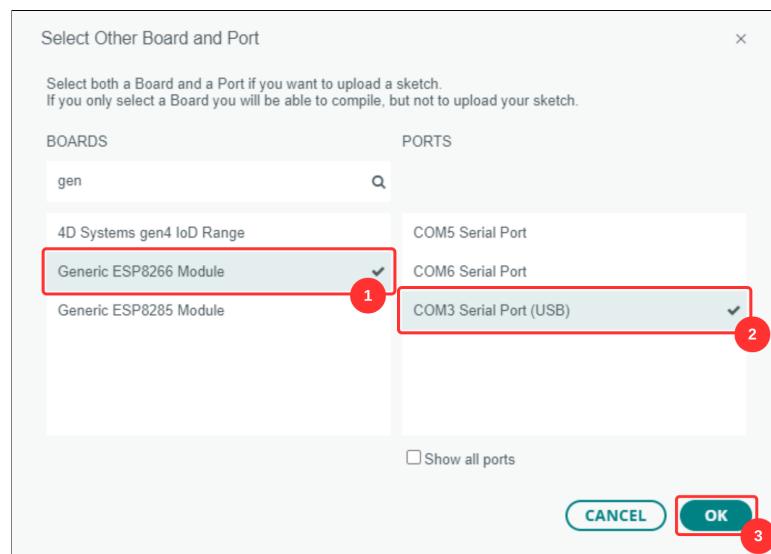
Program 8.2: Chương trình nháy LED

Để nạp chương trình cho ESP, ta cần kết nối cổng nạp **ESP PROG** với máy tính thông qua USB Type_C. Sau đó, ta điều chỉnh **SW2** trên Kit thí nghiệm ở trạng thái ON-ON. Tiếp theo, ta kết nối phần mềm Arduino IDE với ESP như hình sau:



Hình 8.9: Chọn board trong Arduino

Sau khi chọn board, cửa sổ chọn sau hiện ra để bạn biết máy tính của mình đang có thể kết nối với các thiết bị nào:



Hình 8.10: Chọn Board và Port cho ESP

Trường hợp không phát hiện được **COM Port** thì cần phải cài đặt Driver CH340 ở link sau: Driver CH340

Lưu ý: sau khi nạp chương trình thành công, ta phải điều chỉnh **SW2** trở lại trạng thái OFF-OFF thì ESP mới có thể giao tiếp với STM32.

ESP8266 ESP-12E được thiết kế trên Kit thí nghiệm với mục đích kết nối Internet. Trong bài lab này, ta sẽ kết nối Internet thông qua giao thức phổ biến nhất là Wifi.

```
1 #include <ESP8266WiFi.h>
2
3 //Wifi name
4 #define WLAN_SSID      "Your_Wifi_Name"
5 //Wifi password
6 #define WLAN_PASS      "Your_Wifi_Password"
7
8 int led_counter = 0;
9 int led_status = HIGH;
10
11 void setup() {
12     // put your setup code here, to run once:
13     //set pin 2,5 as OUTPUT
14     pinMode(2, OUTPUT);
15     pinMode(5, OUTPUT);
16     //set busy pin HIGH
17     digitalWrite(5, HIGH);
18
19     //connect Wifi
20     WiFi.begin(WLAN_SSID, WLAN_PASS);
21     while (WiFi.status() != WL_CONNECTED) {
22         delay(500);
23     }
24
25     //finish setup, set busy pin LOW
26     digitalWrite(5, LOW);
27
28 }
29
30 void loop() {
31     // put your main code here, to run repeatedly:
32     led_counter++;
33     if(led_counter == 100){
34         // every 1s
```

```

35     led_counter = 0;
36     //toggle LED
37     if(led_status == HIGH) led_status = LOW;
38     else led_status = HIGH;
39
40     digitalWrite(2, led_status);
41 }
42 delay(10);
43 }
```

Program 8.3: Kết nối Wifi

Chương trình trên là một ví dụ cho việc ESP8266 kết nối Internet bằng giao thức Wifi. Một số điểm cần lưu ý:

- Thay thế tên Wifi và password ở các dòng 4 và 6.
- Các dòng từ 20 → 23 được dùng để kết nối Wifi.
- Chân GPIO5 của ESP được cấu hình là Output và được sử dụng để báo cho STM32F4 biết ESP đang trong trạng thái BUSY hay không. Trong ví dụ này, tín hiệu từ GPIO5 ở mức HIGH thì nó đang trong trạng thái BUSY. Trên Kit thí nghiệm, trạng thái này được thể hiện ở **LED4**.
- Khi đoạn chương trình trên được thực hiện, **LED4** sẽ ở trạng thái sáng, sau khi Wifi được kết nối thành công thì **LED4** sẽ chuyển sang trạng thái tắt. Lúc này, LED xanh ở trên ESP sẽ chớp tắt.

Như đã đề cập ở phần trước, ESP và STM32 giao tiếp với nhau thông qua UART. Ngoài việc config USART2 trên STM32 như phần trên, ta cần thiết lập giao tiếp UART trên ESP trong đoạn code dưới đây:

```

1 void setup() {
2     // put your setup code here, to run once:
3
4     //set pin 2,5 as OUTPUT
5     pinMode(2, OUTPUT);
6     pinMode(5, OUTPUT);
7     //set busy pin HIGH
8     digitalWrite(5, HIGH);
9
10    Serial.begin(115200);
11 }
```

```

12 //connect Wifi
13 WiFi.begin(WLAN_SSID, WLAN_PASS);
14 while (WiFi.status() != WL_CONNECTED) {
15     delay(500);
16 }
17
18 //finish setup, set busy pin LOW
19 digitalWrite(5, LOW);
20
21 }
22
23 void loop() {
24     // put your main code here, to run repeatedly:
25
26     if(Serial.available()){
27         int msg = Serial.read();
28         if(msg == 'o') Serial.print('0');
29     }
30
31     led_counter++;
32     if(led_counter == 100){
33         // every 1s
34         led_counter = 0;
35         //toggle LED
36         if(led_status == HIGH) led_status = LOW;
37         else led_status = HIGH;
38
39         digitalWrite(2, led_status);
40     }
41     delay(10);
42 }

```

Program 8.4: Giao tiếp với STM32

Trong đoạn chương trình trên:

- Tại dòng 10, ta thiết lập giao tiếp UART với baudrate 115200, giá trị này phải thống nhất với giá trị đã được thiết lập trên STM32.
- Dòng 26 → 29, thực hiện việc nhận tín hiệu UART từ STM32 và xử lí. Trong ví dụ này, ta sẽ kiểm tra kết nối bằng cách gửi lại kí tự 'O' nếu nhận được kí tự

'o' từ STM32.

5.2 Hướng dẫn làm việc với Adafruit

5.2.1 Tạo tài khoản và kênh dữ liệu

Adafruit.io là một dịch vụ đám mây (cloud service) do Adafruit cung cấp, được thiết kế đặc biệt để hỗ trợ việc quản lý và truyền tải dữ liệu từ các thiết bị IoT. Server này chúng ta có thể sử dụng server miễn phí tại: <https://io.adafruit.com/>. Sau khi truy cập, ta chọn **Get Started for Free** để đăng ký. Nhập đầy đủ thông tin sau đó chọn **CREATE ACCOUNT** để tạo tài khoản.

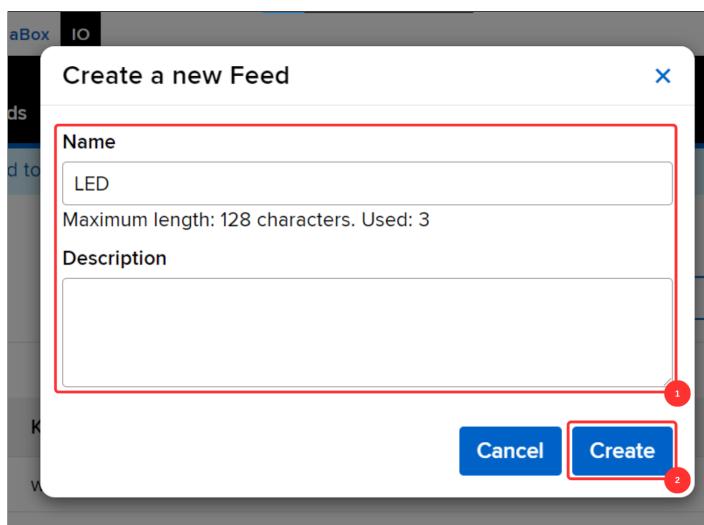
The screenshot shows the Adafruit sign-up page. At the top, there is a navigation bar with links: Shop, Learn, Blog, Forums, LIVE!, AdaBox, and IO. Below the navigation bar is the Adafruit logo. The main section is titled "SIGN UP". On the left, there is a descriptive text: "The best way to shop with Adafruit is to create an account which allows you to shop faster, track the status of your current orders, review your previous orders and take advantage of our other member benefits." To the right of this text are five input fields labeled "FIRST NAME", "LAST NAME", "EMAIL", "USERNAME", and "PASSWORD". Below the "USERNAME" field, there is a note: "Username is viewable to the public on the forums, Adafruit IO, and elsewhere." At the bottom of the form are two buttons: a blue "CREATE ACCOUNT" button and a blue "SIGN IN" button below it.

Hình 8.11: Màn hình đăng ký Adafruit

Trong bài lab này, chúng ta sẽ làm việc ở trong IO, trước tiên là trong phần kênh dữ liệu (feed). Chúng ta tạo một feed như sau:

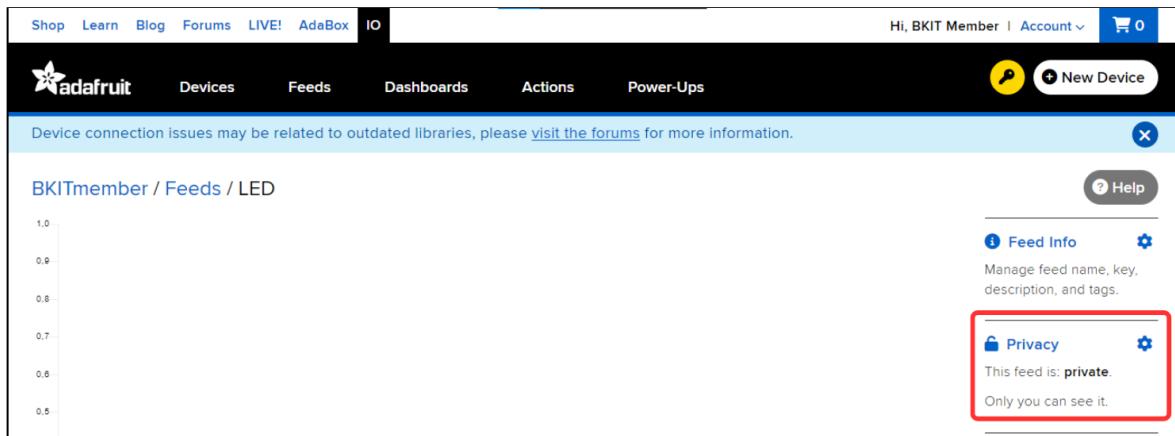
The screenshot shows the Adafruit IO web interface. At the top, there is a navigation bar with links for Shop, Learn, Blog, Forums, LIVE!, AdaBox, IO (highlighted with a red box and number 1), Devices, Feeds (highlighted with a red box and number 2), Dashboards, Actions, Power-Ups, and a user account section. Below the navigation is a message about device connection issues. The main area is titled 'BKITMember / Feeds' and contains two buttons: '+ New Feed' (highlighted with a red box and number 3) and '+ New Group'. A search bar is also present. The feed list table has columns for Feed Name, Key, Last value, and Recorded. One entry is shown: 'Welcome Feed' with key 'welcome-feed', last value 'welcome-feed', recorded 9 minutes ago, and a lock icon.

Hình 8.12: Tạo một feed mới

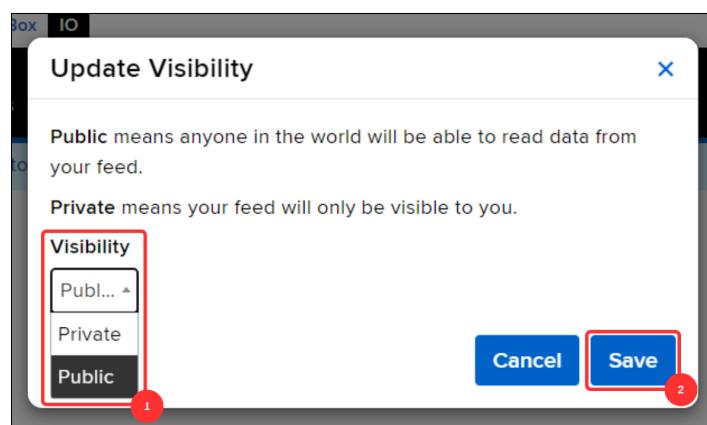


Hình 8.13: Đặt tên, mô tả và tạo feed

Sau khi tạo được một feed, chúng ta sẽ chọn feed và chuyển nó sang chế độ public (hình 8.14 và 8.15). Việc đổi chế độ này làm cho quá trình lập trình trở nên thuận tiện hơn.



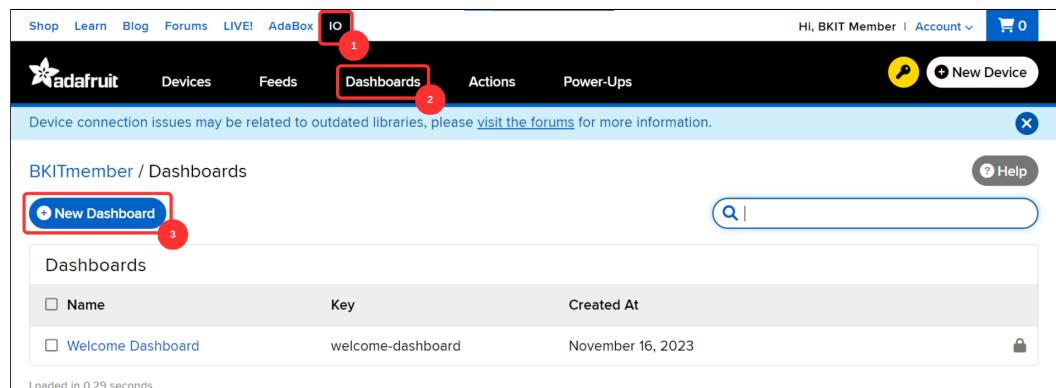
Hình 8.14: Chọn biểu tượng cài đặt tại mục Privacy



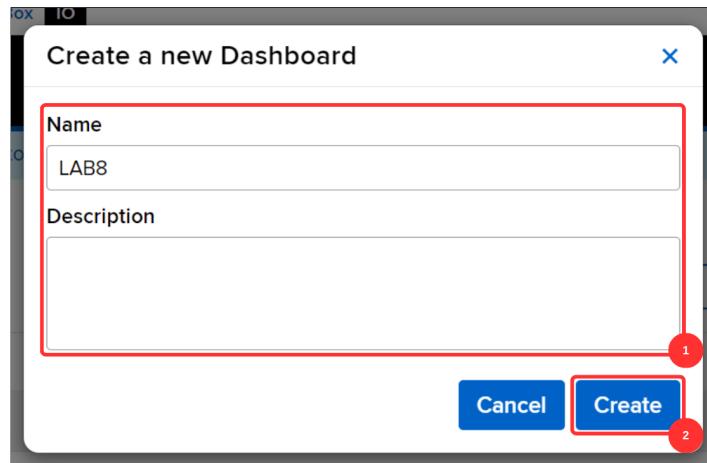
Hình 8.15: Chọn chế độ Public và Save

5.2.2 Tạo dashboard và kết nối với feed

Trong phần này, ta sẽ tạo một dashboard với một nút nhấn, sau đó liên kết dashboard và feed dữ liệu và kiểm tra tương tác giữa chúng. Trước tiên, chúng ta sẽ tạo một dashboard mới:

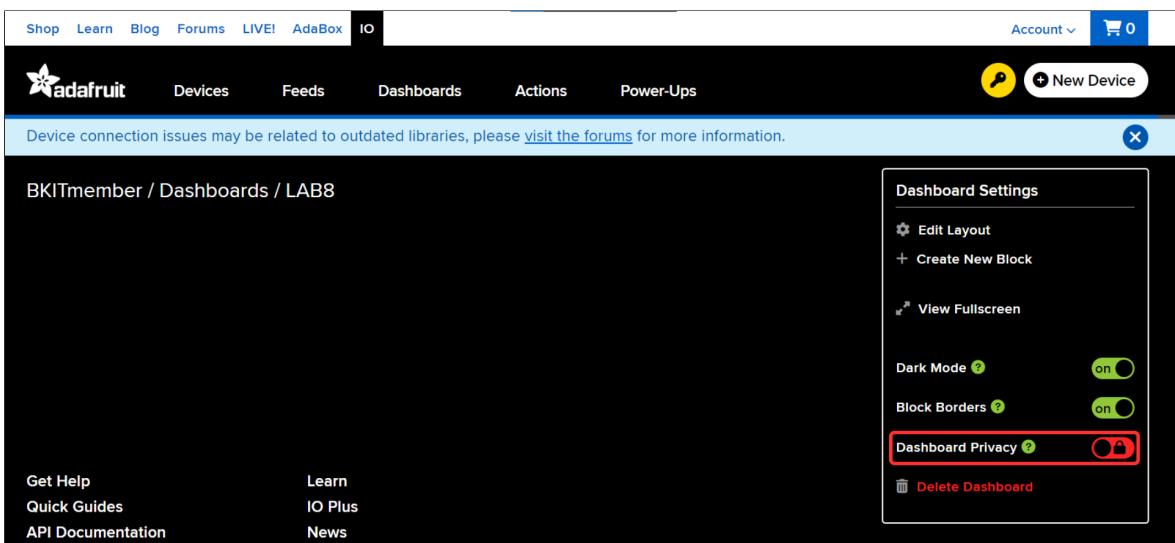


Hình 8.16: Tạo dashboard



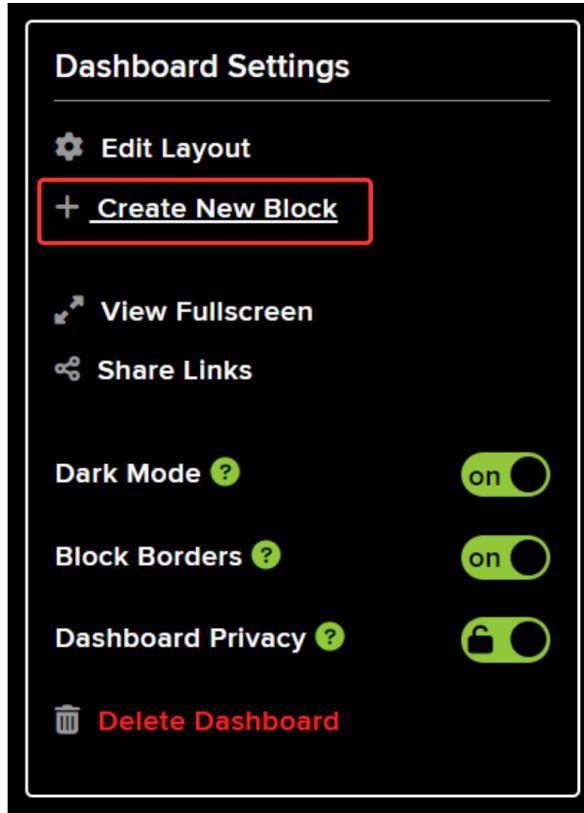
Hình 8.17: Đặt tên, mô tả và tạo dashboard

Sau khi tạo được một dashboard, chúng ta chọn dashboard và chuyển nó sang chế độ public (hình 8.18). Ta sẽ cần **Confirm** để xác nhận việc chuyển đổi. Việc này làm cho quá trình lập trình trở nên thuận tiện hơn.



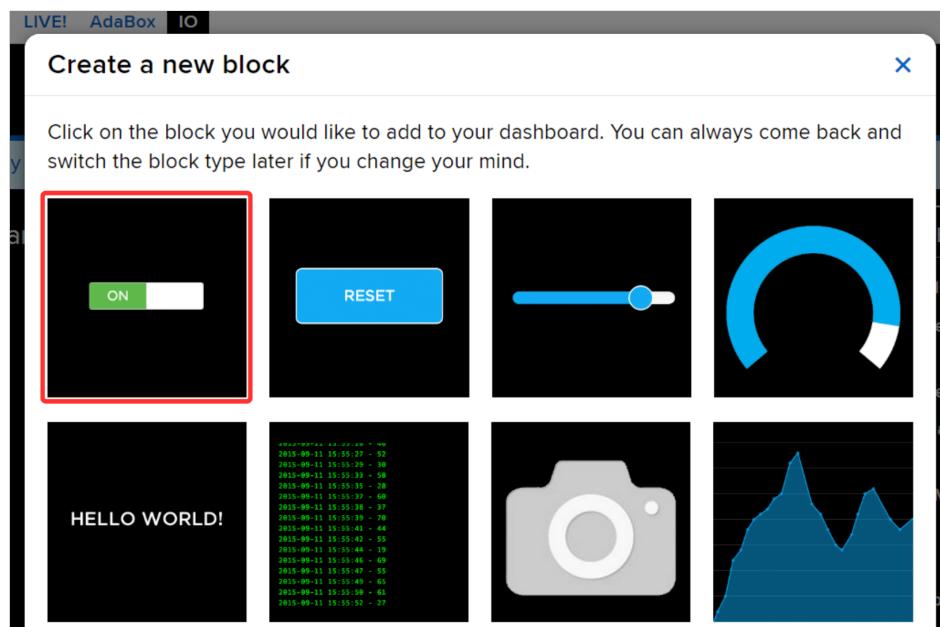
Hình 8.18: Chuyển dashboard sang public

Bây giờ, ta sẽ tiến hành tạo một block để thực hiện việc bật tắt đèn:



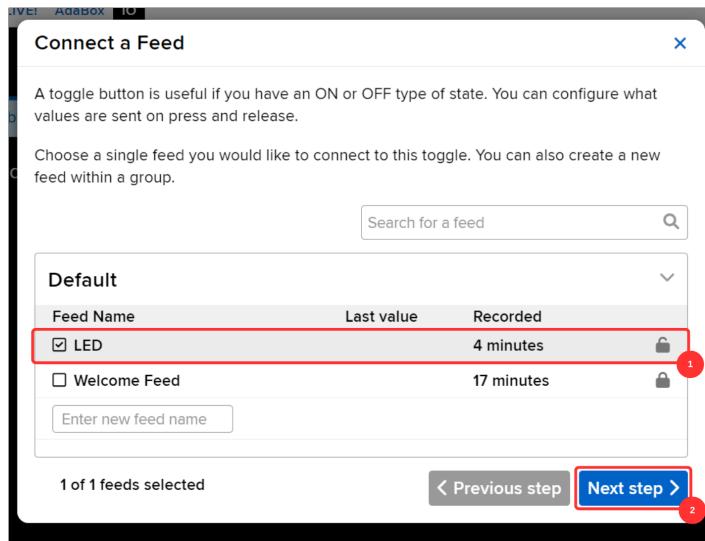
Hình 8.19: Tạo block mới

Sau khi chọn **Create New Block** thì hộp thoại như hình 8.20 sẽ xuất hiện, chúng ta có thể chọn những block phù hợp với project mà chúng ta đang làm. Trong bài lab này, chúng ta sẽ chọn Toggle button block, như hình dưới đây:



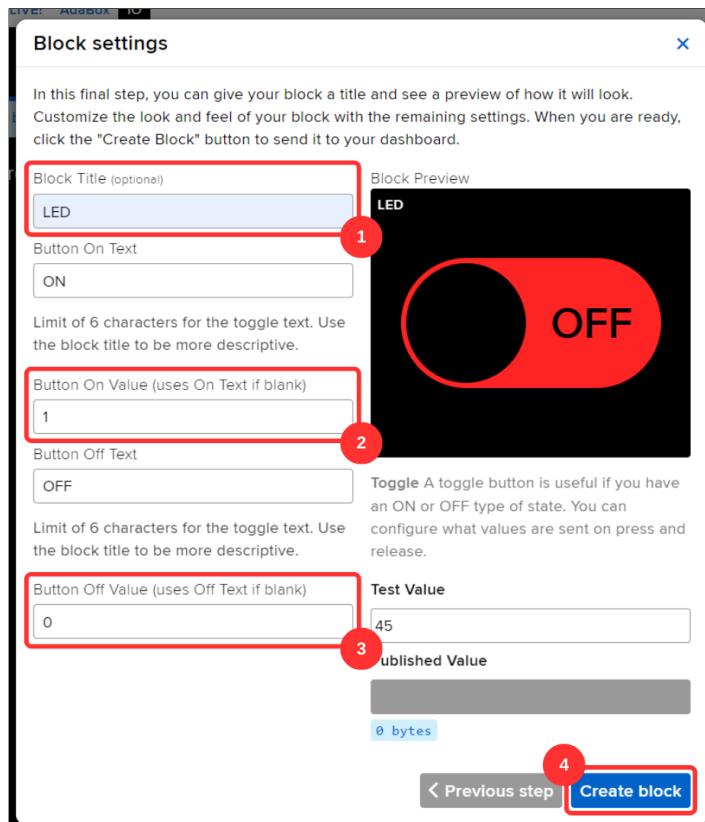
Hình 8.20: Hộp thoại chọn block

Sau khi chọn được block, ta sẽ liên kết block này với feed LED mà ta đã tạo lúc trước.



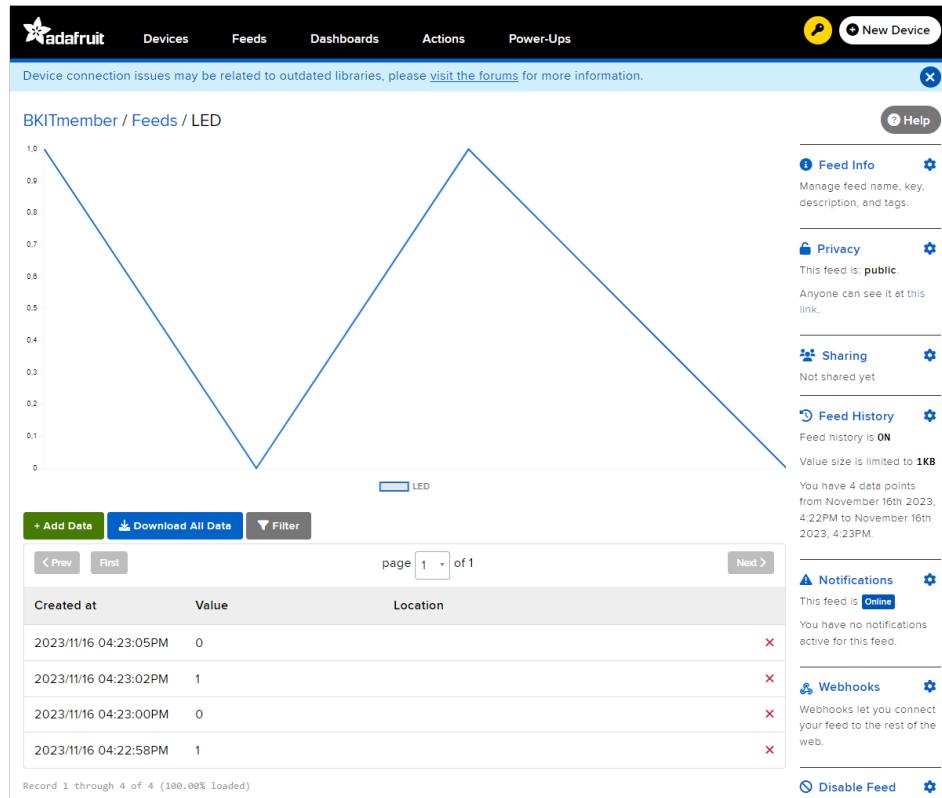
Hình 8.21: Kết nối block với feed

Định nghĩa các giá trị của toggle button, trong trường hợp này thì ta đang định nghĩa đèn bật thì dữ liệu gửi về là '1', đèn tắt thì dữ liệu gửi về là '0'. Sau đó chọn **Create block** để tạo block như hình 8.22.



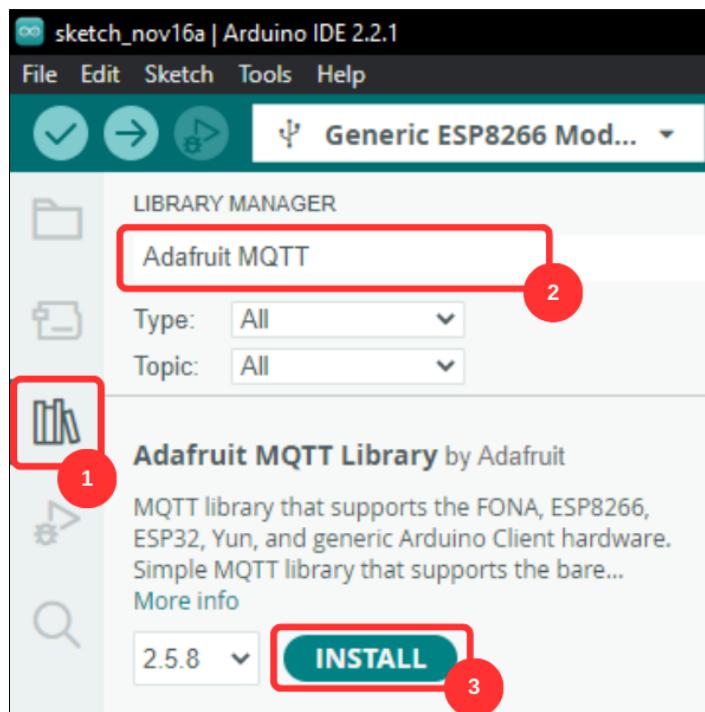
Hình 8.22: Định nghĩa các giá trị của block

Sau khi tạo xong dashboard, ta có thể chạy thử kết nối của dashboard với feed, biểu đồ giá trị trả về theo thời gian sẽ được thể hiện tại feed LED như hình 8.23.



Hình 8.23: Kiểm tra kết nối

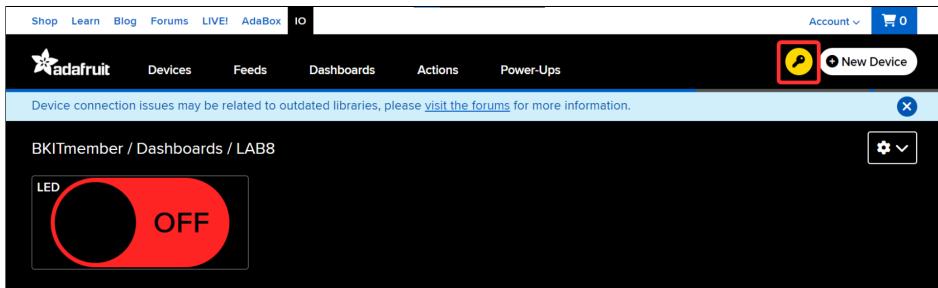
Để kết nối với server Adafruit, ta cần thêm thư viện Adafruit MQTT như sau:



Hình 8.24: Thêm thư viện trên Arduino

Sau đây là đoạn chương trình kết nối và đẩy dữ liệu lên Adafruit.

- Dòng 10 → 21 sẽ khai báo các tham số để kết nối với Adafruit thông qua MQTT. Để kết nối với server Adafruit ta cần lấy các thông tin **Username** và **Active Key** của tài khoản theo như hình 8.25 và 8.26.



Hình 8.25: Lấy key của Adafruit

YOUR ADAFRUIT IO KEY

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username: BKITmember

Active Key: aio_WqPO35igKT0CDtaUAr647rXFJxFs

REGENERATE KEY

[Hide Code Samples](#)

Arduino

```
#define IO_USERNAME "BKITmember"  
#define IO_KEY "aio_WqPO35igKT0CDtaUAr647rXFJxFs"
```

Linux Shell

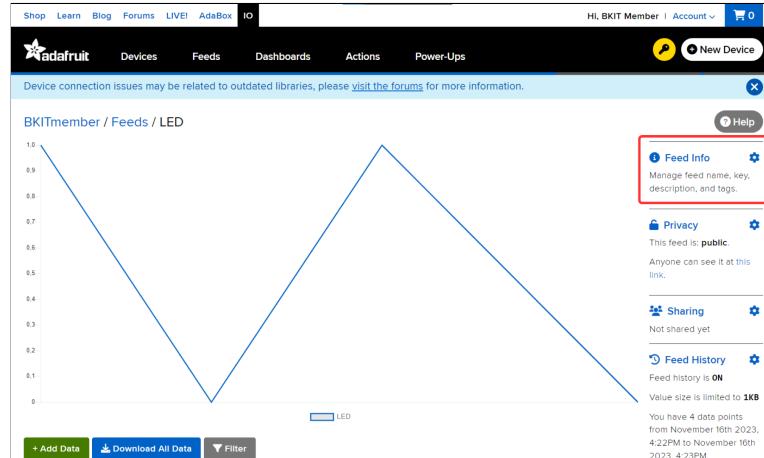
```
export IO_USERNAME="BKITmember"  
export IO_KEY="aio_WqPO35igKT0CDtaUAr647rXFJxFs"
```

Scripting

```
ADAFRUIT_IO_USERNAME = "BKITmember"  
ADAFRUIT_IO_KEY = "aio_WqPO35igKT0CDtaUAr647rXFJxFs"
```

Hình 8.26: Lấy key của Adafruit

- Dòng 23 sẽ khai báo biến nhằm đẩy dữ liệu lên một feed của Adafruit.



Hình 8.27: Lấy key của Feed

The screenshot shows a modal dialog titled "Create a new Feed". It has fields for "Name" (containing "LED") and "Key" (containing "led"). Below these are "Current Endpoints" for "Web" (URL: <https://io.adafruit.com/BKITmember/feeds/led>), "API" (URL: <https://io.adafruit.com/api/v2/BKITmember/feeds/led>), and "MQTT" (URL: <BKITmember/feeds/led>). The "MQTT by Key" URL is highlighted with a red box. There is also a "Description" field and "Cancel" and "Create" buttons at the bottom.

Hình 8.28: Lấy key của Feed

- Dòng 45 → 47 sẽ thiết lập kết nối với server Adafruit thông qua MQTT.
- Dòng 57 → 62 sẽ thêm đoạn chương trình giúp nhận dữ liệu thu thập từ STM32 và gửi lên server.
 - Nếu dữ liệu nhận được là 'a' thì ESP sẽ gửi giá trị 0 lên server nhằm báo hiệu đèn đã được tắt.
 - Nếu dữ liệu nhận được là 'A' thì ESP sẽ gửi giá trị 1 lên server nhằm báo hiệu đèn đã được bật.

```

1 #include <ESP8266WiFi.h>
2 #include "Adafruit_MQTT.h"
3 #include "Adafruit_MQTT_Client.h"
4
5 //Wifi name
6 #define WLAN_SSID          "Your_Wifi_Name"
7 //Wifi password
8 #define WLAN_PASS          "Your_Wifi_Password"
9
10 //setup Adafruit
11 #define AIO_SERVER          "io.adafruit.com"
12 #define AIO_SERVERPORT      1883
13 //fill your username
14 #define AIO_USERNAME        "Your_Adafruit_Name"
15 //fill your key
16 #define AIO_KEY              "Your_Adafruit_Password"
17
18 //setup MQTT
19 WiFiClient client;
20 Adafruit_MQTT_Client mqtt(&client, AIO_SERVER,
   AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
21
22 //set publish
23 Adafruit_MQTT_Publish light_pub = Adafruit_MQTT_Publish(&
   mqtt, AIO_USERNAME "/feeds/led");
24
25 int led_counter = 0;
26 int led_status = HIGH;
27
28 void setup() {
29   // put your setup code here, to run once:

```

```

30
31 //set pin 2,5 as OUTPUT
32 pinMode(2, OUTPUT);
33 pinMode(5, OUTPUT);
34 //set busy pin HIGH
35 digitalWrite(5, HIGH);

36
37 Serial.begin(115200);

38
39 //connect Wifi
40 WiFi.begin(WLAN_SSID, WLAN_PASS);
41 while (WiFi.status() != WL_CONNECTED) {
42     delay(500);
43 }

44
45 while (mqtt.connect() != 0) {
46     delay(500);
47 }

48
49 //finish setup, set busy pin LOW
50 digitalWrite(5, LOW);

51
52 }

53
54 void loop() {
55     // put your main code here, to run repeatedly:
56
57     if(Serial.available()){
58         int msg = Serial.read();
59         if(msg == 'o') Serial.print('0');
60         else if(msg == 'a') light_pub.publish(0);
61         else if(msg == 'A') light_pub.publish(1);
62     }
63
64     led_counter++;
65     if(led_counter == 100){
66         // every 1s
67         led_counter = 0;
68         //toggle LED

```

```

69     if(led_status == HIGH) led_status = LOW;
70     else led_status = HIGH;
71
72     digitalWrite(2, led_status);
73 }
74 delay(10);
75 }
```

Program 8.5: Gửi dữ liệu lên Adafruit

Sau khi gửi dữ liệu lên Adafruit thành công, ta cần hiện thực giao tiếp theo chiều ngược lại để ta có thể điều khiển đèn từ dashboard trên server. Khi ta thay đổi công tắc trên Adafruit dashboard, dữ liệu sẽ được gửi từ server xuống ESP, sau đó tùy vào dữ liệu nhận được mà ESP sẽ gửi dữ liệu đến STM32 để điều khiển đèn. Đoạn chương trình sau mô tả quá trình ESP nhận dữ liệu từ server. Trong đó:

- Dòng 26 được thêm vào để khởi tạo một biến giúp nhận dữ liệu từ một feed trên Adafruit.
- Dòng 31 → 34 là hàm sẽ được gọi mỗi khi nhận được dữ liệu điều khiển đèn từ Adafruit. Tùy theo dữ liệu nhận được mà ta sẽ gửi 'a' đến STM32 để tắt đèn, hoặc gửi 'A' để mở đèn.
- Dòng 53 → 54 nhằm mục đích đăng ký nhận tín hiệu mỗi khi có dữ liệu được gửi lên feed của Adafruit.
- Dòng 70 được thêm vào để liên tục cập nhật dữ liệu đã đăng ký từ server.

```

1 #include <ESP8266WiFi.h>
2 #include "Adafruit_MQTT.h"
3 #include "Adafruit_MQTT_Client.h"
4
5 //Wifi name
6 #define WLAN_SSID      "Your_Wifi_Name"
7 //Wifi password
8 #define WLAN_PASS      "Your_Wifi_Password"
9
10 //setup Adafruit
11 #define AIO_SERVER      "io.adafruit.com"
12 #define AIO_SERVERPORT  1883
13 //fill your username
14 #define AIO_USERNAME    "Your_Adafruit_Name"
```

```

15 //fill your key
16 #define AIO_KEY           "Your_Adafruit_Password"
17
18 //setup MQTT
19 WiFiClient client;
20 Adafruit_MQTT_Client mqtt(&client, AIO_SERVER,
   AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
21
22 //setup publish
23 Adafruit_MQTT_Publish light_pub = Adafruit_MQTT_Publish(&
   mqtt, AIO_USERNAME "/feeds/led");
24
25 //setup subscribe
26 Adafruit_MQTT_Subscribe light_sub = Adafruit_MQTT_Subscribe
   (&mqtt, AIO_USERNAME "/feeds/led", MQTT_QOS_1);
27
28 int led_counter = 0;
29 int led_status = HIGH;
30
31 void lightcallback(char* value, uint16_t len){
32   if(value[0] == '0') Serial.print('a');
33   if(value[0] == '1') Serial.print('A');
34 }
35
36 void setup() {
37   // put your setup code here, to run once:
38   //set pin 2,5 as OUTPUT
39   pinMode(2, OUTPUT);
40   pinMode(5, OUTPUT);
41   //set busy pin HIGH
42   digitalWrite(5, HIGH);
43
44   Serial.begin(115200);
45
46   //connect Wifi
47   WiFi.begin(WLAN_SSID, WLAN_PASS);
48   while (WiFi.status() != WL_CONNECTED) {
49     delay(500);
50   }

```

```

51
52 //subscribe light feed
53 light_sub.setCallback(lightcallback);
54 mqtt.subscribe(&light_sub);

55
56 //connect MQTT
57 while (mqtt.connect() != 0) {
58     mqtt.disconnect();
59     delay(500);
60 }

61
62 //finish setup, set busy pin LOW
63 digitalWrite(5, LOW);
64 }

65
66 void loop() {
67     // put your main code here, to run repeatedly:
68
69     //receive packet
70     mqtt.processPackets(1);

71
72     //read serial
73     if(Serial.available()){
74         int msg = Serial.read();
75         if(msg == 'o') Serial.print('0');
76         else if(msg == 'a') light_pub.publish(0);
77         else if(msg == 'A') light_pub.publish(1);
78     }

79
80     led_counter++;
81     if(led_counter == 100){
82         // every 1s
83         led_counter = 0;
84         //toggle LED
85         if(led_status == HIGH) led_status = LOW;
86         else led_status = HIGH;

87
88         digitalWrite(2, led_status);
89     }

```

```
90     delay(10);  
91 }
```

Program 8.6: Nhận dữ liệu từ Adafruit

5.3 Hướng dẫn thư viện STM

Vì STM32 và ESP giao tiếp với nhau thông qua UART nên các hàm liên quan đến ESP sẽ được thực thi bổ sung tại thư viện **uart.h**.

void uart_init_esp()

- **Mô tả:** Khởi tạo kết nối với ESP.
- **Tham số:** Không.
- **Giá trị trả về:** Không.

void uart_EspSendBytes(uint8_t* bytes, uint16_t size)

- **Mô tả:** Gửi các byte kí tự đến ESP.
- **Tham số:**
 - **bytes:** chuỗi dữ liệu.
 - **size:** độ dài chuỗi dữ liệu tính theo byte.
- **Giá trị trả về:** Không.

uint8_t uart_EspCheck()

- **Mô tả:** Kiểm tra kết nối đến ESP.
- **Tham số:** Không.
- **Giá trị trả về:**
 - **0:** Kết nối thất bại.
 - **1:** Kết nối thành công.

Để thuận tiện cho việc quản lý source code, ta sẽ thêm 2 file **light_control.h** và **light_control.c** để quản lý việc điều khiển đèn. Trong phần hướng dẫn này đèn tại **Output Y0** và nút nhấn "**0**" sẽ được dùng để mô phỏng.

```
1 #ifndef INC_LIGHT_CONTROL_H_
2 #define INC_LIGHT_CONTROL_H_
3
4 #include "gpio.h"
5 #include "uart.h"
6 #include "button.h"
7 #include "lcd.h"
8
9 extern uint8_t light_status;
10
11 void lightProcess();
12
13 void test_Esp();
```

Program 8.7: light_control.h

```
1 #include "light_control.h"
2
3 uint8_t light_status = 0;
4
5 void lightProcess(){
6     if(button_count[13] == 1){
7         light_status = 1 - light_status;
8         if(light_status == 1){
9             uart_EspSendBytes("A", 1);
10        } else {
11            uart_EspSendBytes("a", 1);
12        }
13    }
14    if(light_status == 1){
15        HAL_GPIO_WritePin(OUTPUT_Y0_GPIO_Port, OUTPUT_Y0_Pin,
16        1);
17    } else {
18        HAL_GPIO_WritePin(OUTPUT_Y0_GPIO_Port, OUTPUT_Y0_Pin,
19        0);
20    }
21}
22
23 void test_Esp(){
24     if(uart_EspCheck() == 0) uart_EspSendBytes("o", 1);
```

```

23     else lcd_ShowStr(10, 50, "ESP Connect", GREEN, BLACK, 24,
24         0);
}

```

Program 8.8: light_control.c

Trong đó, hàm **lightProcess** sẽ điều khiển đèn dựa trên biến trạng thái và thay đổi biến trạng thái, gửi dữ liệu đến ESP khi có tín hiệu nút nhấn. Hàm **test_Esp** sẽ gửi kí tự 'o' đến ESP để kiểm tra kết nối và in lên màn hình thông báo nếu trạng thái kết nối là thành công. Ngoài ra, để xử lý dữ liệu nhận được từ UART, ta phải thêm các dòng lệnh vào hàm interrupt ở file **uart.c**

```

1 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
2     if(huart->Instance == USART1){
3         //rs232 isr
4         //...
5     }
6
7     if(huart->Instance == USART2){
8         //esp_isr
9         if(receive_buffer2 == '0') check_esp = 1;
10        else if(receive_buffer2 == 'a') light_status = 0;
11        else if(receive_buffer2 == 'A') light_status = 1;
12
13        //enable receive
14        HAL_UART_Receive_IT(&huart2, &receive_buffer2, 1);
15    }
16}

```

Program 8.9: Hàm intterupt Uart

Hàm main sẽ gọi các hàm điều khiển đèn được thực thi ở trên.

```

1 //...
2 /* USER CODE BEGIN Includes */
3 #include "software_timer.h"
4 #include "led_7seg.h"
5 #include "button.h"
6 #include "lcd.h"
7 #include "picture.h"
8 #include "ds3231.h"
9 #include "sensor.h"
10 #include "buzzer.h"

```

```

11 #include "touch.h"
12 #include "uart.h"
13 #include "light_control.h"
14 /* USER CODE END Includes */
15
16 // ...
17
18 /* USER CODE BEGIN PFP */
19 void system_init();
20 void test_LedDebug();
21 /* USER CODE END PFP */
22
23 int main(void)
24 {
25     // ...
26
27     /* USER CODE BEGIN 2 */
28     system_init();
29     /* USER CODE END 2 */
30
31     /* Infinite loop */
32     /* USER CODE BEGIN WHILE */
33     lcd_Clear(BLACK);
34     while (1)
35     {
36         // 50ms task
37         if(flag_timer2 == 1){
38             flag_timer2 = 0;
39             button_Scan();
40             test_Esp();
41             lightProcess();
42             test_LedDebug();
43         }
44
45         /* USER CODE END WHILE */
46
47         /* USER CODE BEGIN 3 */
48     }
49     /* USER CODE END 3 */

```

```

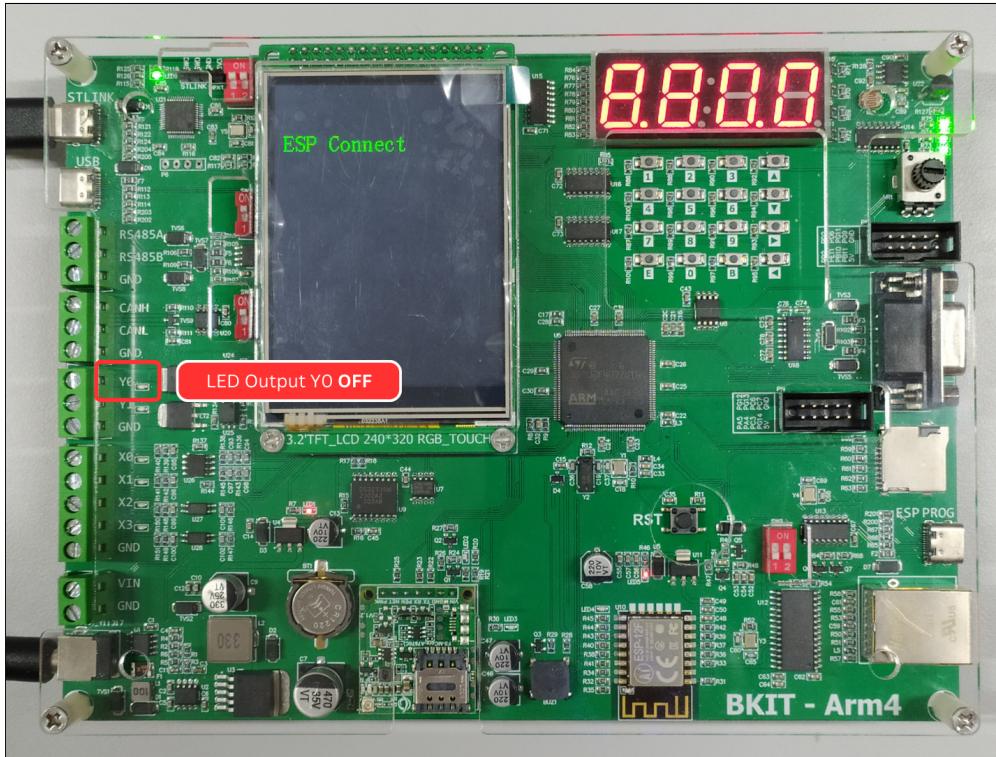
50 }
51
52 // ...
53
54 /* USER CODE BEGIN 4 */
55 void system_init(){
56     timer_init();
57     button_init();
58     lcd_init();
59     uart_init_esp();
60     setTimer2(50);
61 }
62
63 uint8_t count_led_debug = 0;
64
65 void test_LedDebug(){
66     count_led_debug = (count_led_debug + 1)%20;
67     if(count_led_debug == 0){
68         HAL_GPIO_TogglePin(DEBUG_LED_GPIO_Port , DEBUG_LED_Pin);
69     }
70 }
71
72 // ...

```

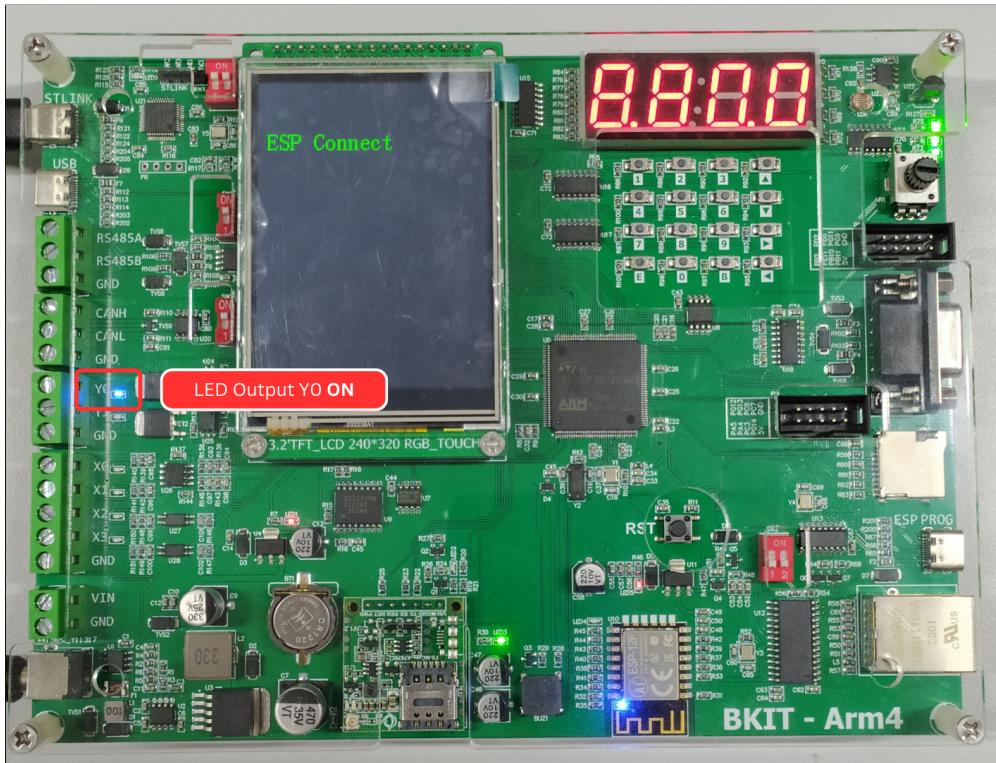
Program 8.10: File main.c

Khi hoàn thành chương trình trên, nếu kết nối ESP thành công, màn hình sẽ hiển thị chuỗi "ESP Connect". Tín hiệu Output Y0 sẽ có thể được điều khiển bởi nút nhấn "0" trên Kit thí nghiệm và bởi công tắc trên Adafruit dashboard. Trạng thái của Output Y0 sẽ được đồng bộ giữa Kit thí nghiệm và server.

Hình ảnh kết quả trên Kit thí nghiệm:



Hình 8.29: Kết quả hiện thực trên Kit thí nghiệm (1)



Hình 8.30: Kết quả hiện thực trên Kit thí nghiệm (2)

6 Bài tập và báo cáo

Thu thập dữ liệu từ cảm biến nhiệt độ trên Kit thí nghiệm và gửi lên server Adafruit mỗi 30s. Tạo dashboard trên Adafruit giúp hiển thị dữ liệu nhiệt độ theo dạng biểu đồ (Gợi ý: truyền dữ liệu nhiệt độ giữa STM và ESP dưới dạng "!TEMP:<Nhiệt độ>#").