# Giao thức MQTT:

# Tổng quan về MQTT

## Định nghĩa

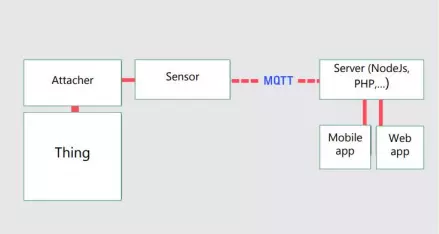
MQTT (Message Queuing Telemetry Transport) là giao thức truyền thông điệp (message) theo mô hình publish/subscribe, được sử dụng cho các thiết bị IoT với băng thông thấp, độ tin cậy cao và khả năng được sử dụng trong mạng lưới không ổn định. Giao thức này cho phép các thiết bị IOT gửi và nhận các tin nhắn nhỏ và đơn giản qua mạng Internet.

MQTT là lựa chọn lý tưởng trong các môi trường như:

* Những nơi mà giá mạng viễn thông đắt đỏ hoặc băng thông thấp hay thiếu tin cậy.
* Khi chạy trên thiết bị nhúng bị giới hạn về tài nguyên tốc độ và bộ nhớ.
* Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng M2M (Machine to Machine).
* MQTT cũng là giao thức được sử dụng trong Facebook Messenger

## Vị trí của MQTT trong mô hình IoT

Một số ưu điểm nổi bật của MQTT như: băng thông thấp, độ tin cậy cao và có thể sử dụng ngay cả khi hệ thống mạng không ổn định, tốn rất ít byte cho việc kết nối với server và connection có thể giữ trạng thái open xuyên suốt, có thể kết nối nhiều thiết bị (MQTT client) thông qua một MQTT server (broker). Bởi vì giao thức này sử dụng băng thông thấp trong môi trường có độ trễ cao nên nó là một giao thức lý tưởng cho các ứng dụng IoT.



Hình 1: Vị trí MQTT trong mô hình IoT

Từ hình ảnh trên có thể hiểu đơn giản MQTT như là cầu nối để giao tiếp Sensor với Server với mục đích đơn giản như gửi/nhận dữ liệu.

## Tính năng và đặc điểm nổi bật

Một số tính năng và đặc điểm nổi bật của giao thức MQTT

* Dạng truyền thông điệp theo mô hình Pub/Sub cung cấp việc truyền tin phân tán một chiều, tách biệt với phần ứng dụng.
* Việc truyền thông điệp là ngay lập tức, không quan tâm đến nội dung được truyền.
* Sử dụng TCP/IP là giao thức nền.
* Tồn tại ba mức độ tin cậy cho việc truyền dữ liệu (QoS: Quality of service)
  + QoS 0: Broker/client sẽ gửi dữ liệu đúng một lần, quá trình gửi được xác nhận bởi chỉ giao thức TCP/IP.
  + QoS 1: Broker/client sẽ gửi dữ liệu với ít nhất một lần xác nhận từ đầu kia, nghĩa là có thể có nhiều hơn 1 lần xác nhận đã nhận được dữ liệu.
  + QoS 2: Broker/client đảm bảo khi gửi dữ liệu thì phía nhận chỉ nhận được đúng một lần, quá trình này phải trải qua 4 bước bắt tay.
* Phần bao bọc dữ liệu truyền nhỏ và được giảm đến mức tối thiểu để giảm tải cho đường truyền.

## Ưu điểm của MQTT

Với những tính năng, đặc điểm nổi bật trên, MQTT mang lại nhiều lợi ích nhất là trong hệ thống SCADA (Supervisory Control And Data Acquisition) khi truy cập dữ liệu IoT.

* Truyền thông tin hiệu quả hơn.
* Tăng khả năng mở rộng.
* Giảm đáng kể tiêu thụ băng thông mạng.
* Rất phù hợp cho điều khiển và do thám.
* Tối đa hóa băng thông có sẵn.
* Chi phí thấp.
* Rất an toàn, bảo mật.
* Được sử dụng trong các ngành công nghiệp dầu khí, các công ty lớn như Amazon, Facebook, ....
* Tiết kiệm thời gian phát triển.
* Giao thức publish/subscribe thu thập nhiều dữ liệu hơn và tốn ít băng thông hơn so với giao thức cũ.

# Mô hình Pub/Sub và cơ chế hoạt động của MQTT

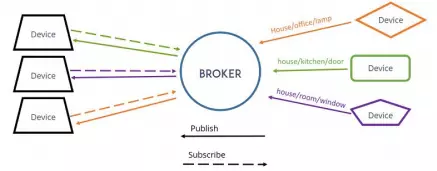
## Mô hình Pub/Sub

### Thành phần

* Client
  + Publisher - Nơi gửi thông điệp
  + Subscriber - Nơi nhận thông điệp
* Broker - Máy chủ môi giới

Trong đó Broker được coi như trung tâm, nó là điểm giao của tất cả các kết nối đến từ Client (Publisher/Subscriber). Nhiệm vụ chính của Broker là nhận thông điệp (message) từ Publisher, xếp vào hàng đợi rồi chuyển đến một địa điểm cụ thể. Ngoài ra, nó có thể đảm nhận thêm một vài tính năng liên quan tới quá trình truyền thông như: bảo mật message, lưu trữ message, logs, ....

Client thì được chia thành hai nhóm là Publisher và Subscriber. Client chỉ làm ít nhất một trong 2 việc là publish các thông điệp (message) lên một/nhiều topic cụ thể hoặc subscribe một/nhiều topic nào đó để nhận message từ topic này.



Hình 2: Mô hình Pub/Sub của MQTT

MQTT Clients tương thích với hầu hết các nền tảng hệ điều hành hiện có: MAC OS, Windows, Linux, Android, iOS, ....

### Ưu điểm và nhược điểm

Ưu điểm của mô hình này:

* Kết nối riêng rẽ, độc lập.
* Khả năng mở rộng.
* Thời gian tách biệt (Time decoupling).
* Đồng bộ riêng rẽ (Synchronization decoupling).

Nhược điểm của mô hình:

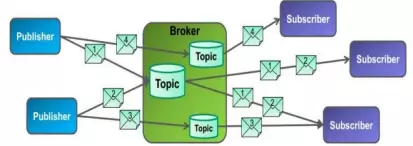
* Kết nối riêng rẽ, độc lập.
* Khả năng mở rộng.
* Thời gian tách biệt (Time decoupling).
* Đồng bộ riêng rẽ (Synchronization decoupling).

## Cơ chế hoạt động của MQTT theo mô hình Pub/Sub

### Tính chất và những đặc điểm riêng

* Tính chất:
  + Space decoupling (Không gian tách biệt)
  + Time decoupling (Thời gian tách biệt)
  + Synchronization decoupling (Sự đồng bộ riêng rẽ)
* Đặc điểm riêng:
  + MQTT sử dụng cơ chế lọc thông điệp dựa vào tiêu đề (subject-based)
  + MQTT có một tầng gọi là chất lượng dịch vụ (Quality of Services – QoS). Nó giúp cho dễ dàng nhận biết được là message có được truyền thành công hay không.

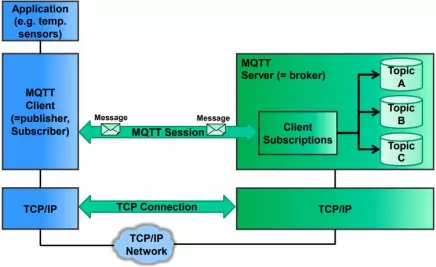
### Cơ chế tổng quan



Hình 3: Cơ chế hoạt động của MQTT (client/server)

* MQTT hoạt động theo cơ chế client/server, nơi mà mỗi cảm biến là một khách hàng (client) và kết nối đến một máy chủ, có thể hiểu như một Máy chủ môi giới (broker), thông qua giao thức TCP (Transmission Control Protocol). Broker chịu trách nhiệm điều phối tất cả các thông điệp giữa phía gửi đến đúng phía nhận.
* MQTT là giao thức định hướng bản tin. Mỗi bản tin là một đoạn rời rạc của tín hiệu và broker không thể nhìn thấy. Mỗi bản tin được publish một địa chỉ, có thể hiểu như một kênh (Topic). Client đăng kí vào một vài kênh để nhận/gửi dữ liệu, gọi là subscribe. Client có thể subscribe vào nhiều kênh. Mỗi client sẽ nhận được dữ liệu khi bất kỳ trạm nào khác gửi dữ liệu vào kênh đã đăng ký. Khi một client gửi một bản tin đến một kênh nào đó gọi là publish.

### Kiến trúc thành phần



Hình 4: Kiến trúc giao thức MQTT

* Thành phần chính của MQTT là Client (Publisher/Subscriber), Server (Broker), Sessions (tạm dịch là Phiên làm việc), Subscriptions và Topics.
* MQTT Client (Publisher/Subscriber): Clients sẽ subscribe một hoặc nhiều topics để gửi và nhận thông điệp từ những topic tương ứng.
* MQTT Server (Broker): Broker nhận những thông tin subscribe (Subscriptions) từ client, nhận thông điệp, chuyển những thông điệp đến các Subscriber tương ứng dựa trên Subscriptions từ client.
* Topic: Có thể coi Topic là một hàng đợi các thông điệp, và có sẵn khuôn mẫu dành cho Subscriber hoặc Publisher. Một cách logic thì các topic cho phép Client trao đổi thông tin với những ngữ nghĩa đã được định nghĩa sẵn. Ví dụ: Dữ liệu cảm biến nhiệt độ của một tòa nhà.
* Session: Một session được định nghĩa là kết nối từ client đến server. Tất cả các giao tiếp giữa client và server đều là 1 phần của session.
* Subscription: Không giống như session, subscription về mặt logic là kết nối từ client đến topic. Khi đã subscribe một topic, Client có thể nhận/gửi thông điệp (message) với topic đó.

# Lập trình MQTT với ESP32

Trong phần này ta sẽ demo một ví dụ cơ bản để kết nối MQTT và ESP32 để giao tiếp với nhau. Cụ thể là điều khiển thiết bị đèn trong nhà và cảnh báo khóa cửa, là một phần nhỏ trong dự án Smart Lock hay nói rộng hơn thì chính là Smart Home.

## Phần cứng và phần mềm

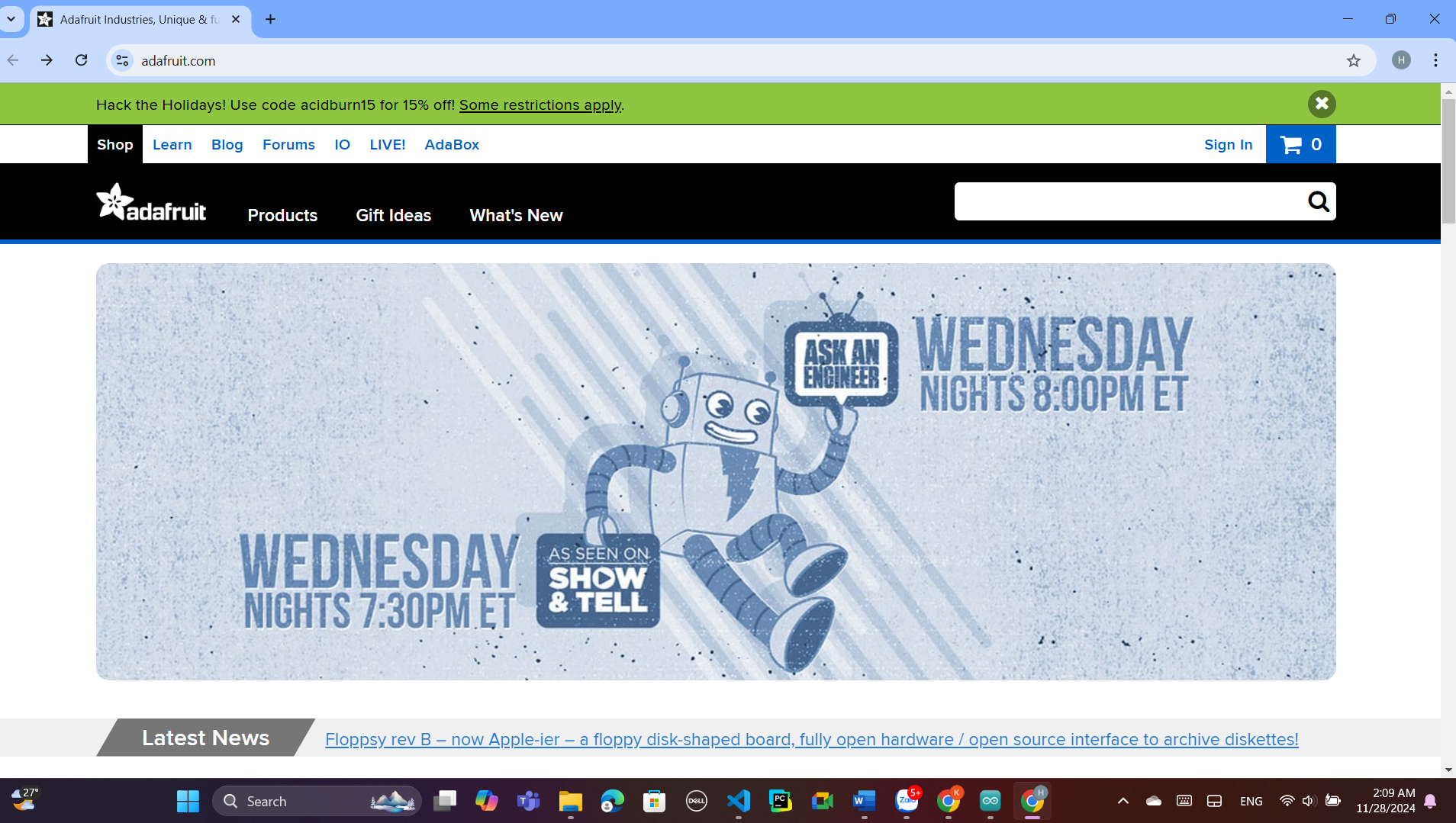
Chúng ta cần trang bị các thành phần sau:

* Một ESP32 và cáp USB (hoặc Type C tùy vào loại ESP32 mà bạn chọn) để kết nối với máy tính
* Chọn một **broker MQTT** trên mạng Internet (**HiveMQ**, **Mosquitto** hoặc **CloudMQTT, …**), ở đây tôi chọn AdafruitIO.
* Một Buzzer và hai đèn LED

## Các bước cần thực hiện

**Bước 1: Tạo broker MQTT:**

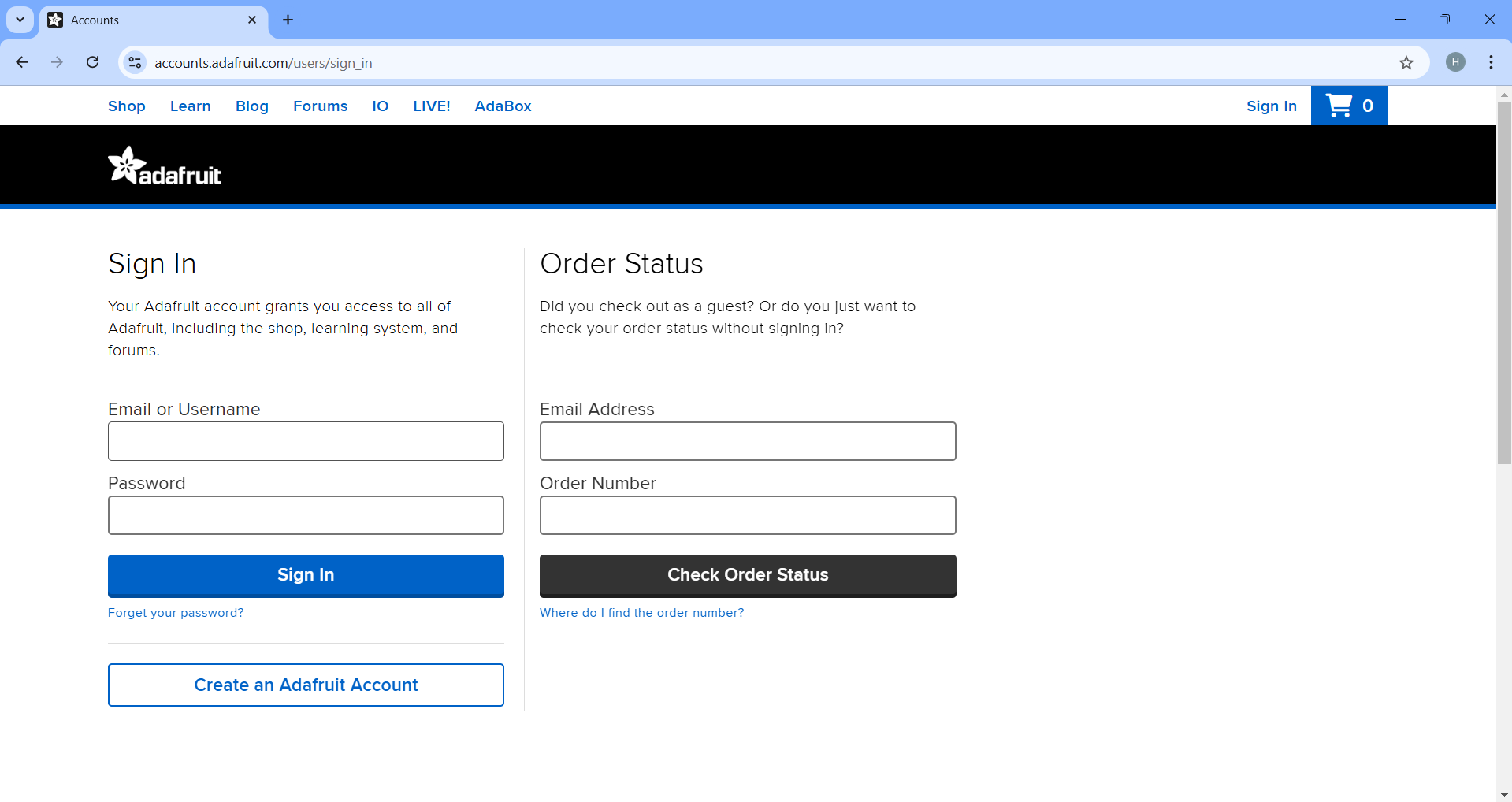
* Trong ví dụ này mình sử dụng **Adafruit broker**. Bạn truy cập vào website: <https://www.adafruit.com> sau đó chọn **Sign in**

****



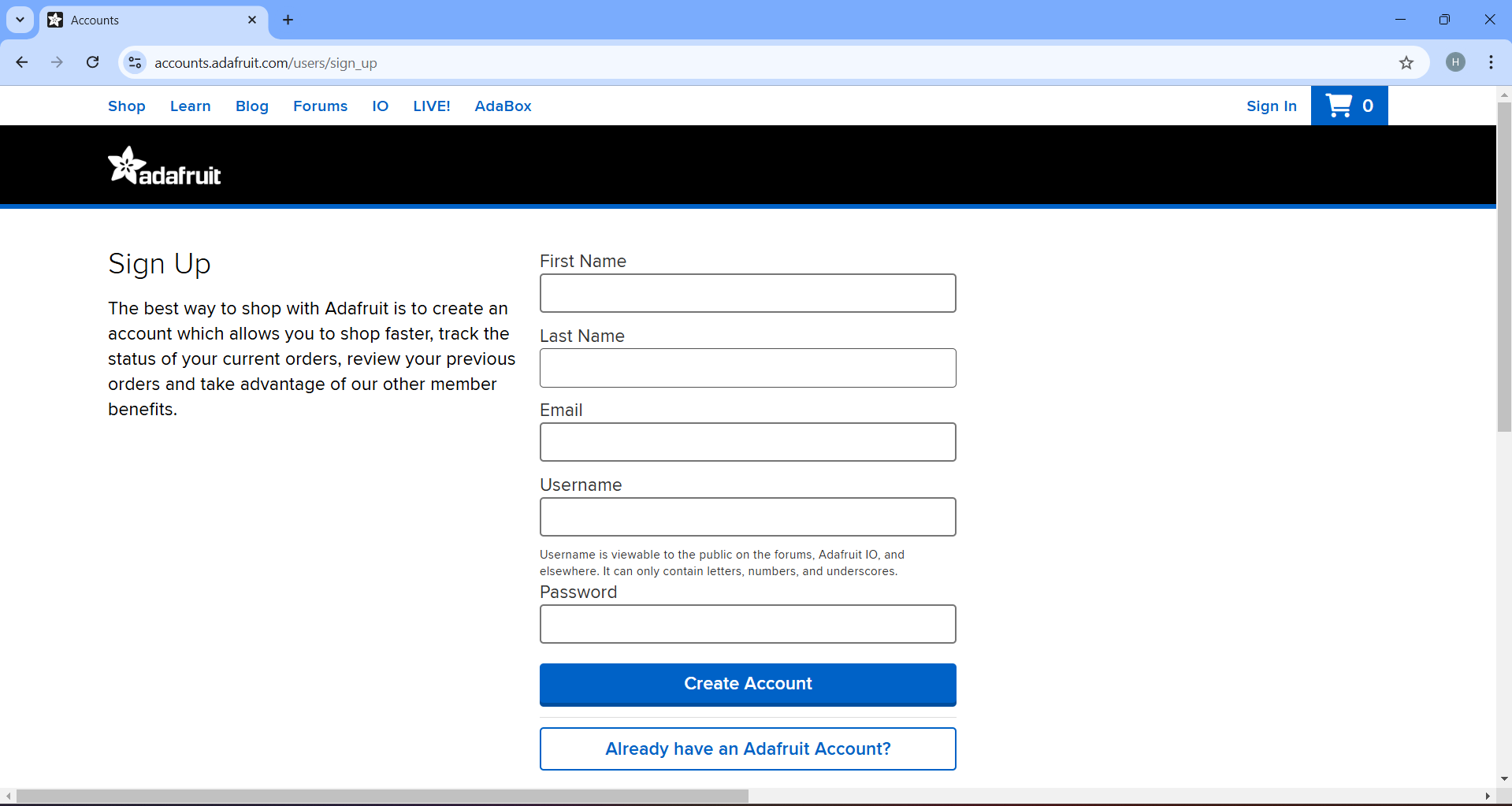
* Chọn tiếp **Create an Adafruit Account**







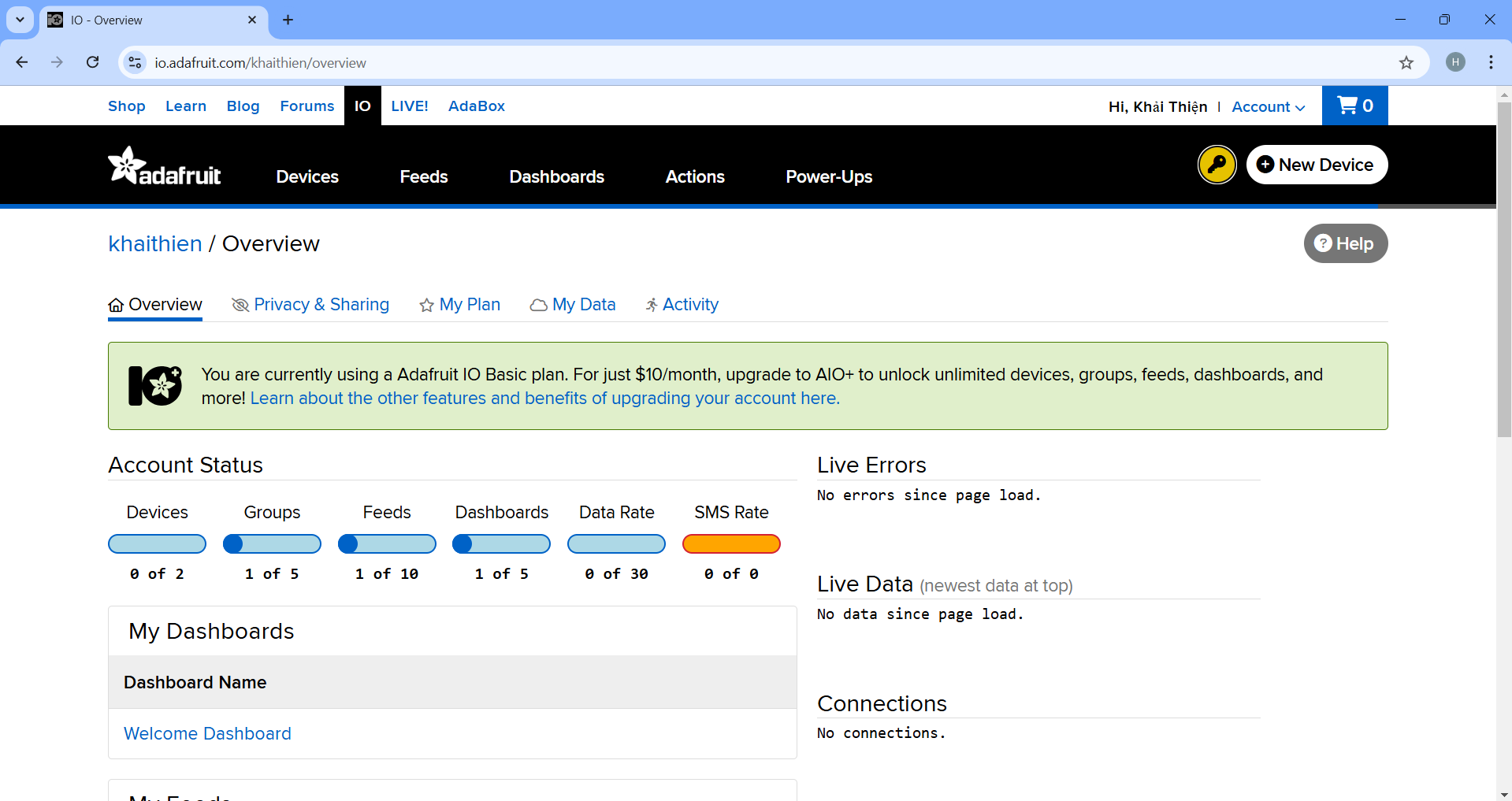
* Tiến hành nhập các trường thông tin bên dưới và nhấp vào **Create Account.**



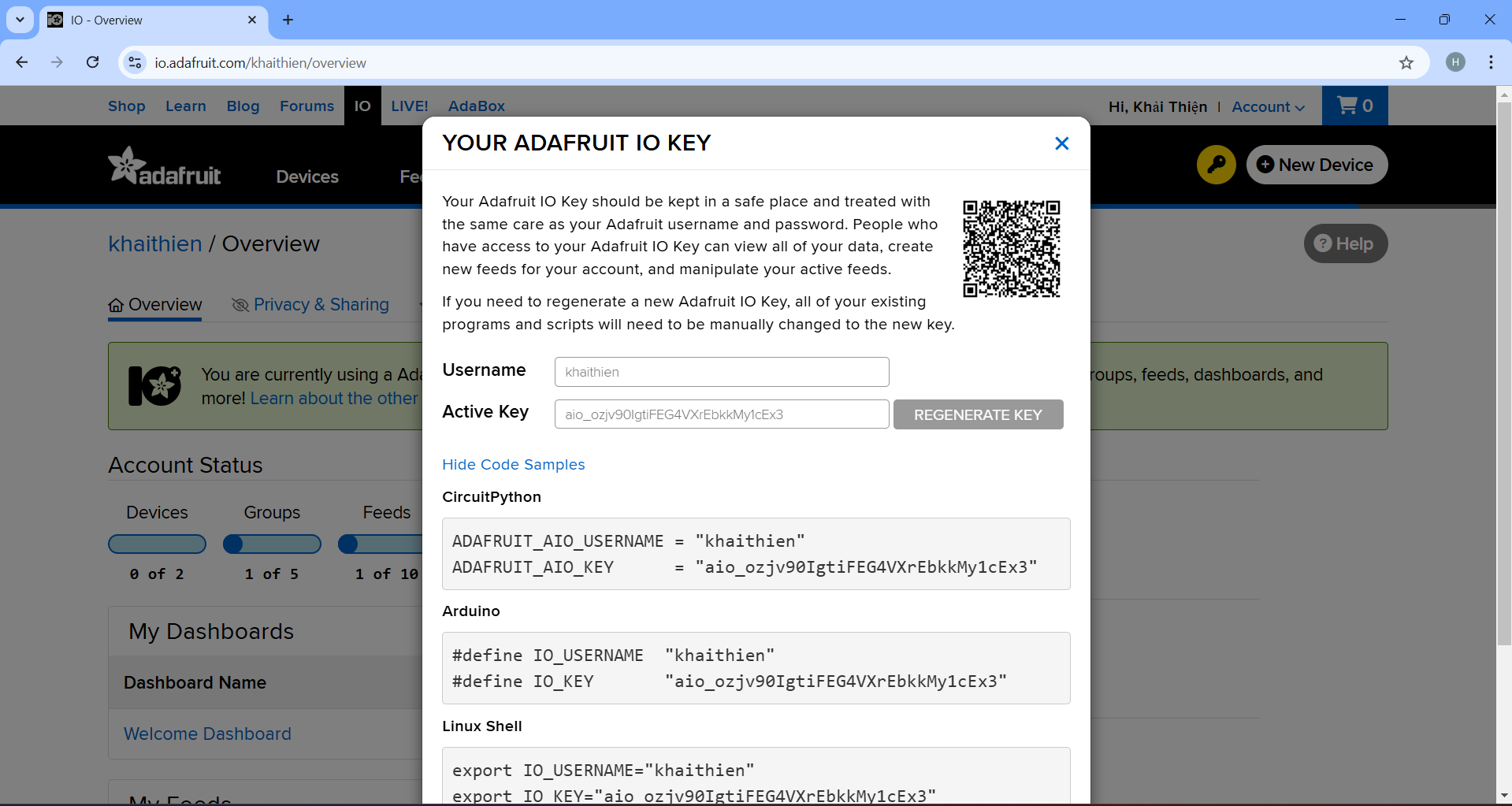


* Nhấp chọn **IO** và chọn **API Key** để lấy mã cho phần code.









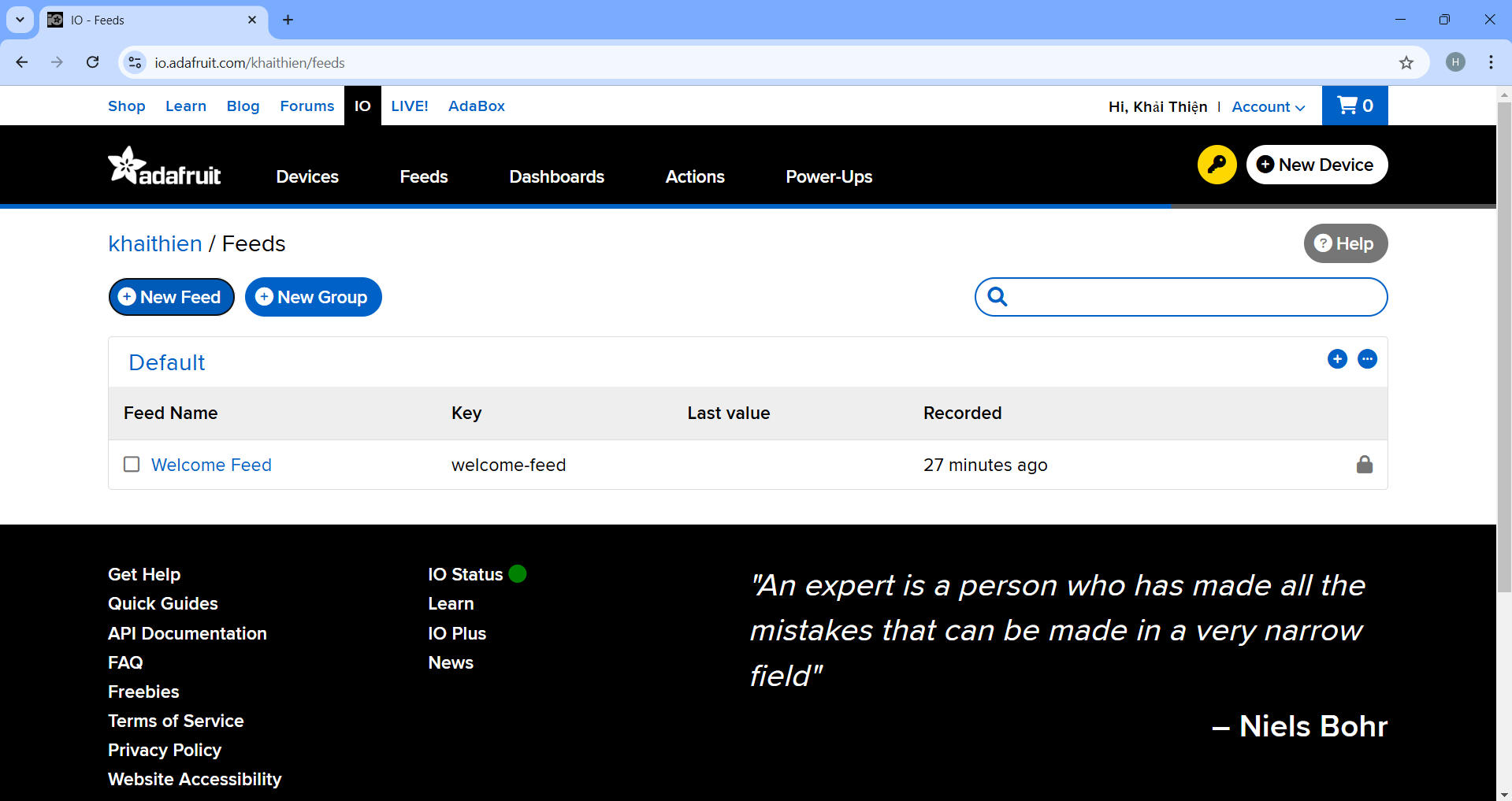
Copy đoạn code này



**Bước 2:Tạo Feeds:**

* Nhấp vào **Feeds** và chọn **New Feed**

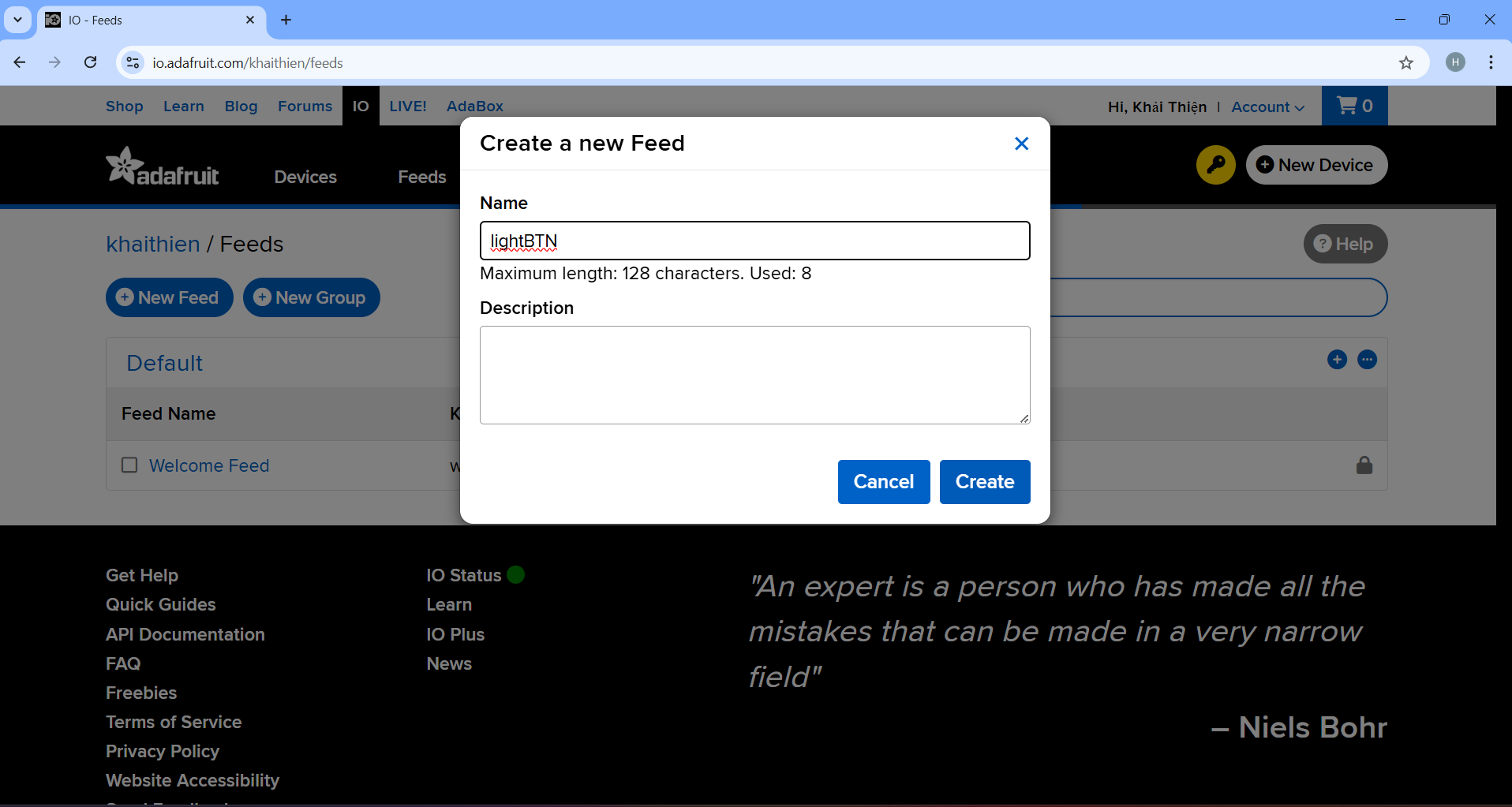






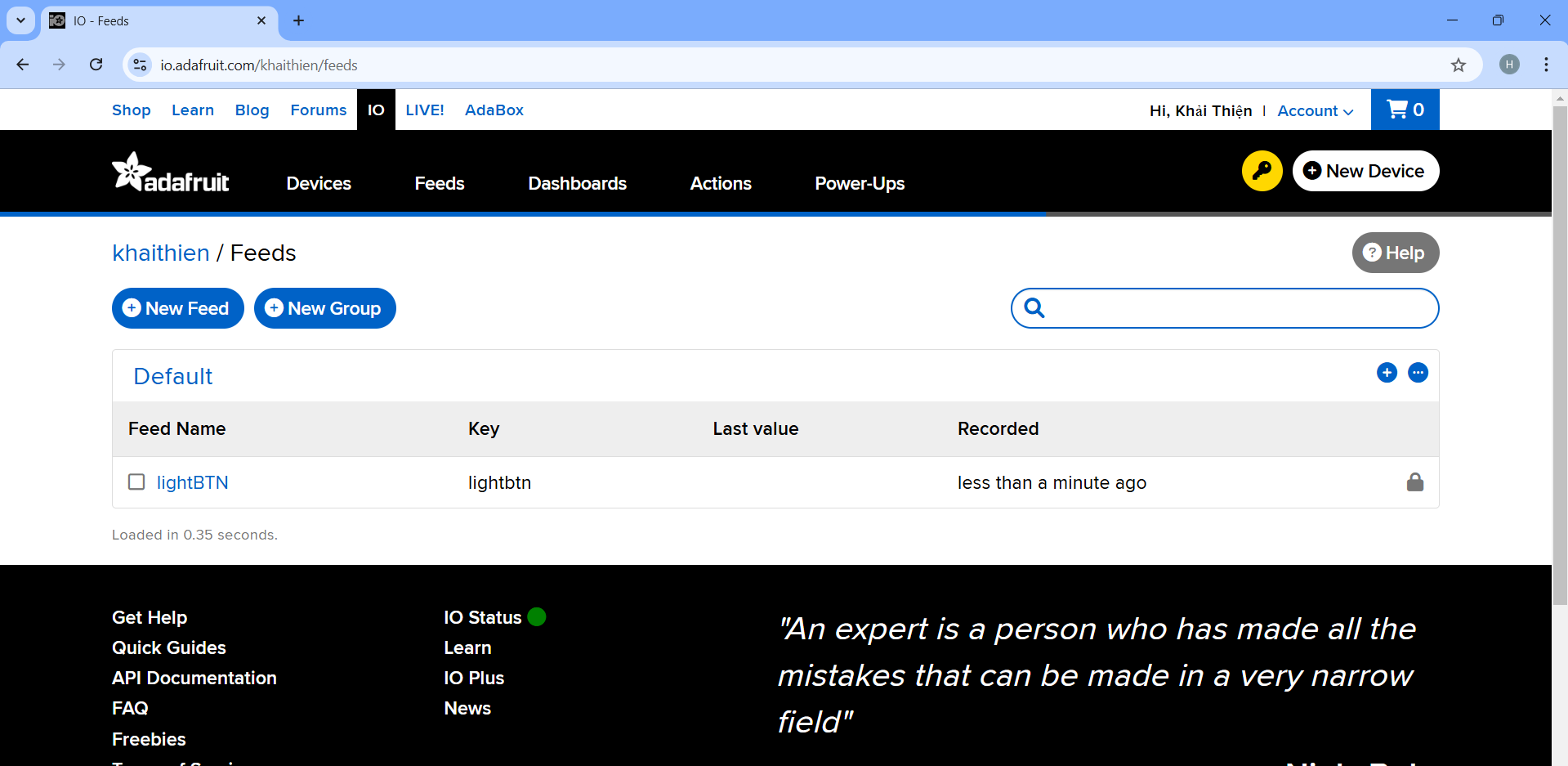
* Nhập tên và nhấn **Create**



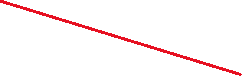




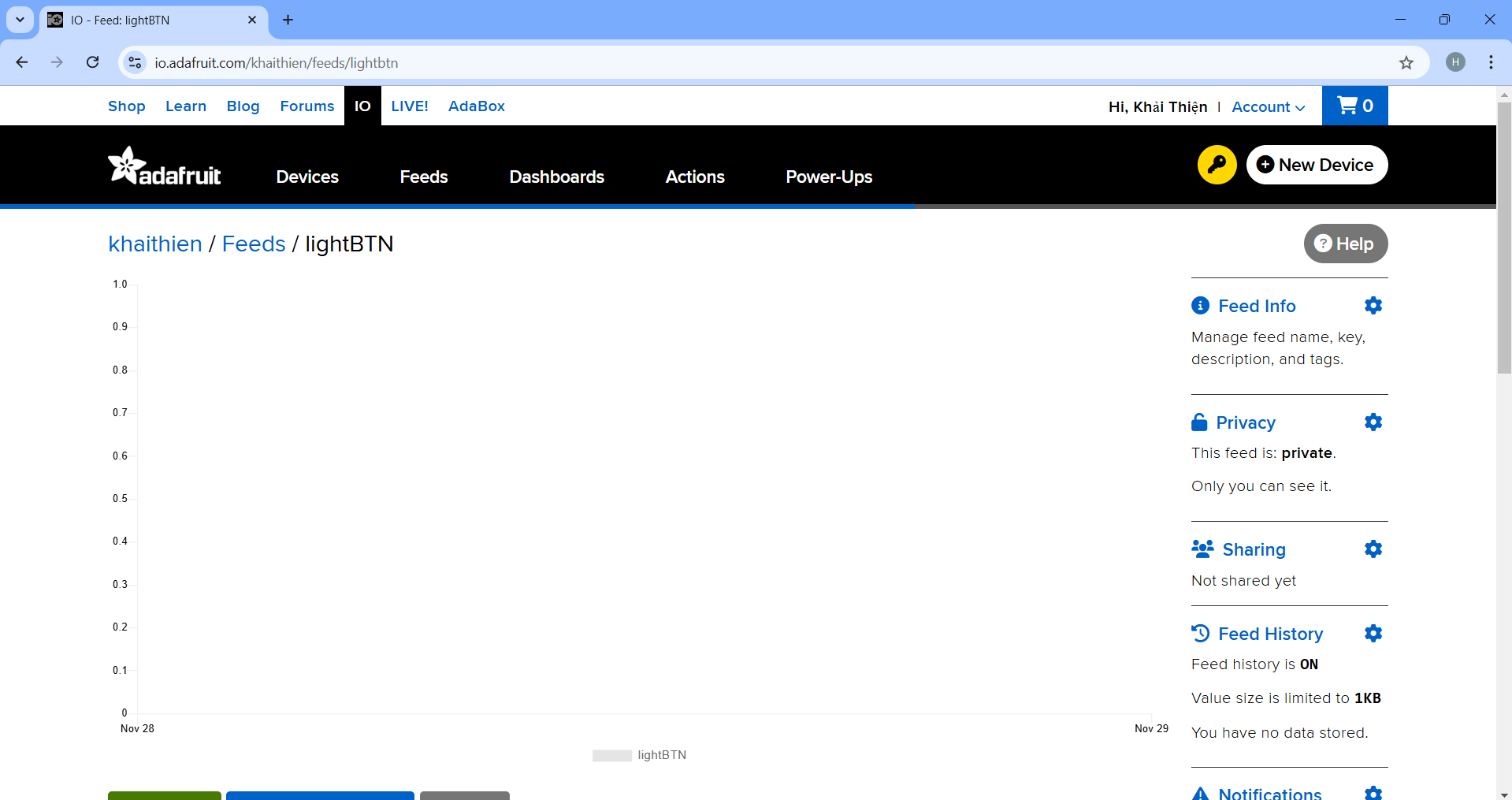
* Lúc này bạn sẽ thấy một feed mới được tạo với tên **(name)** là lightBTN với phần **key** (Phần tên để dùng trong code là **key**) được tạo.



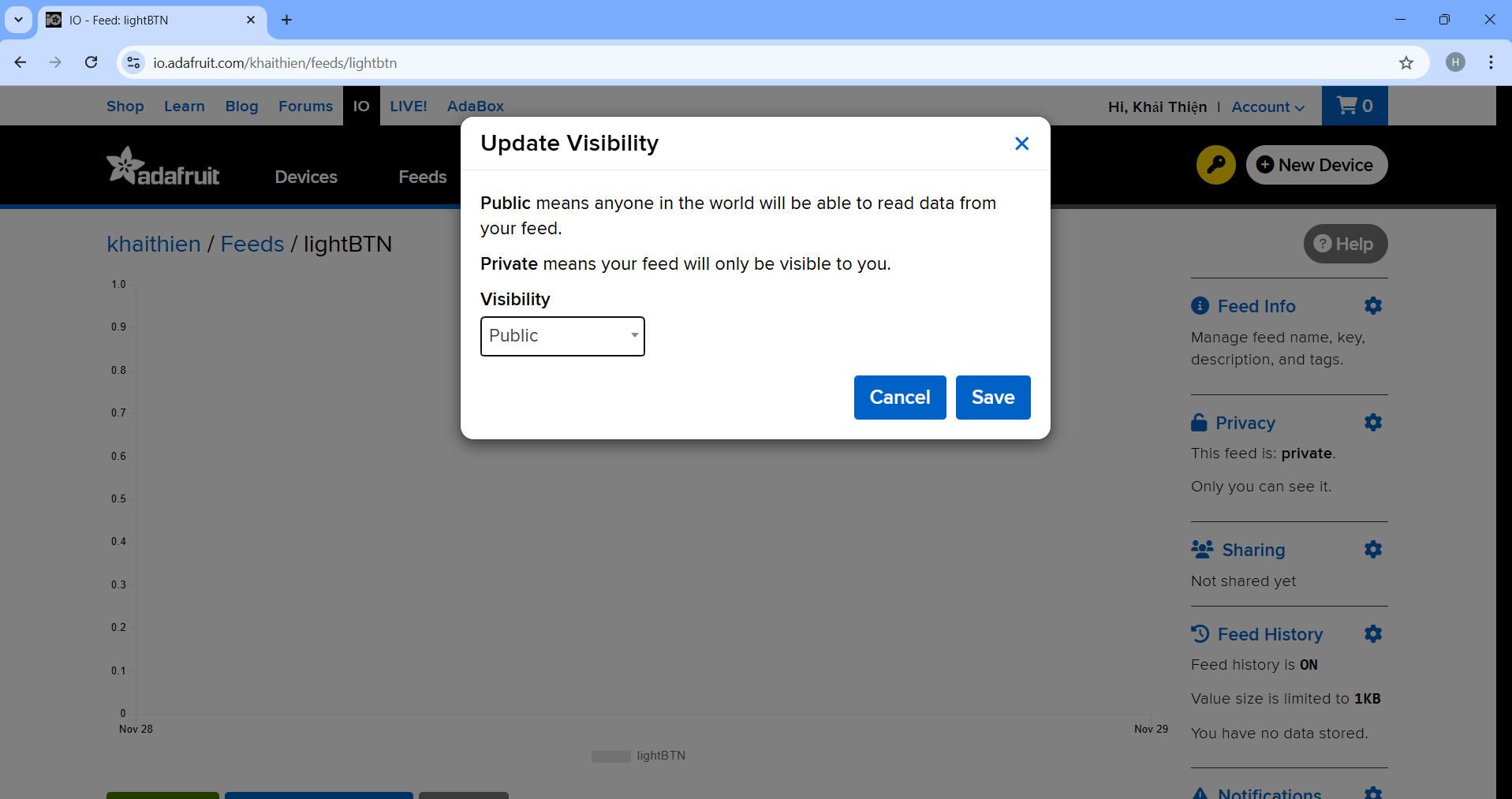
Chế độ: private



* Bấm vào tên của Feed và chỉnh chế độ **Privacy** thành **Public**



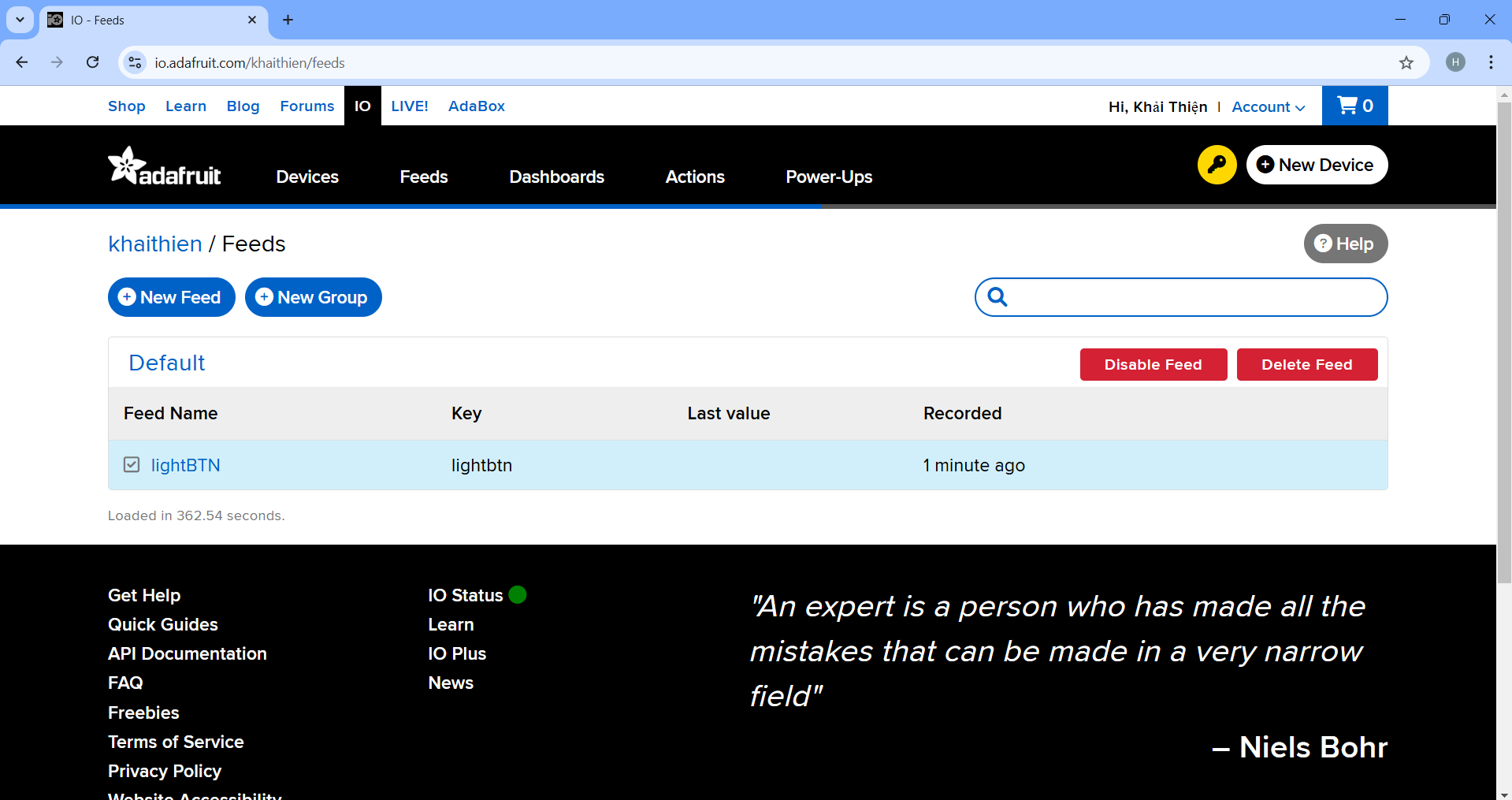




* Sau đó bấm **Save** bạn sẽ thấy có sự thay đổi.



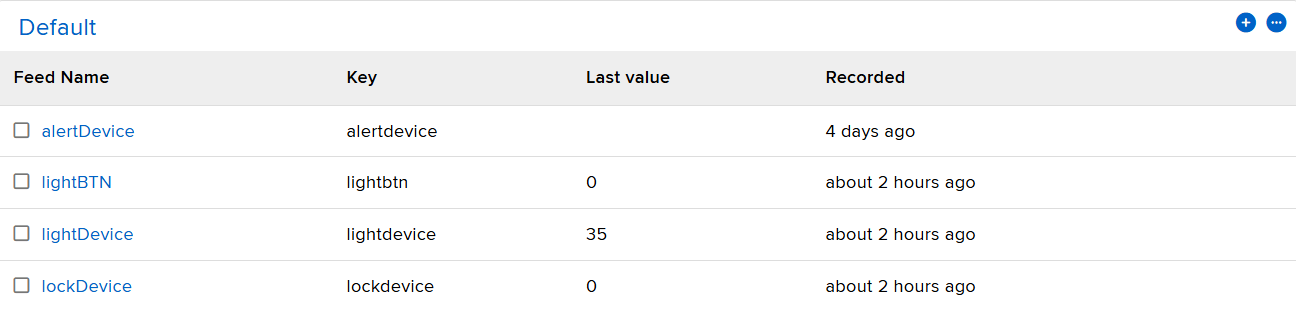




Chế độ: public

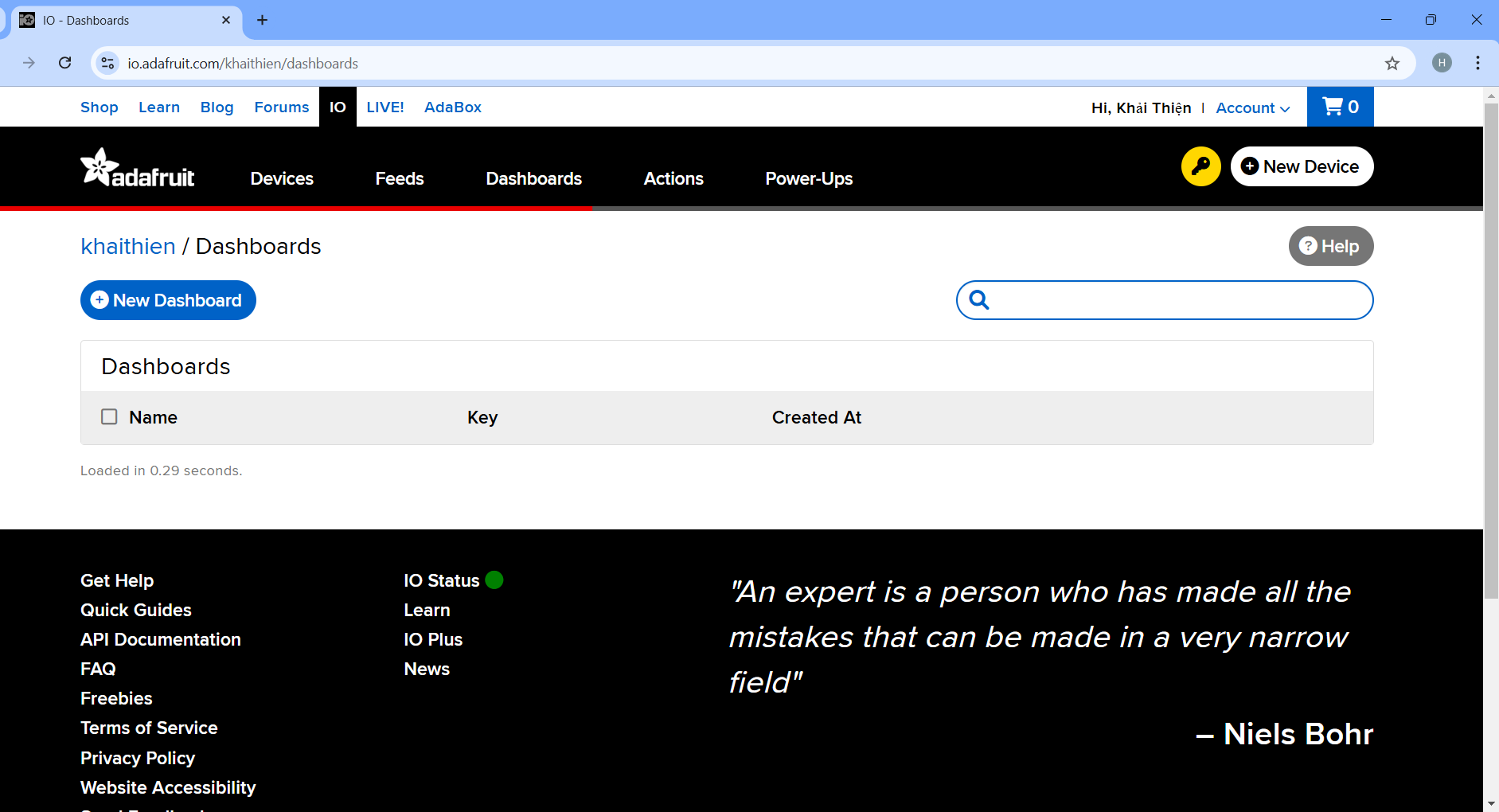


* Tạo các **Feeds** sau đó tương tự như vậy:



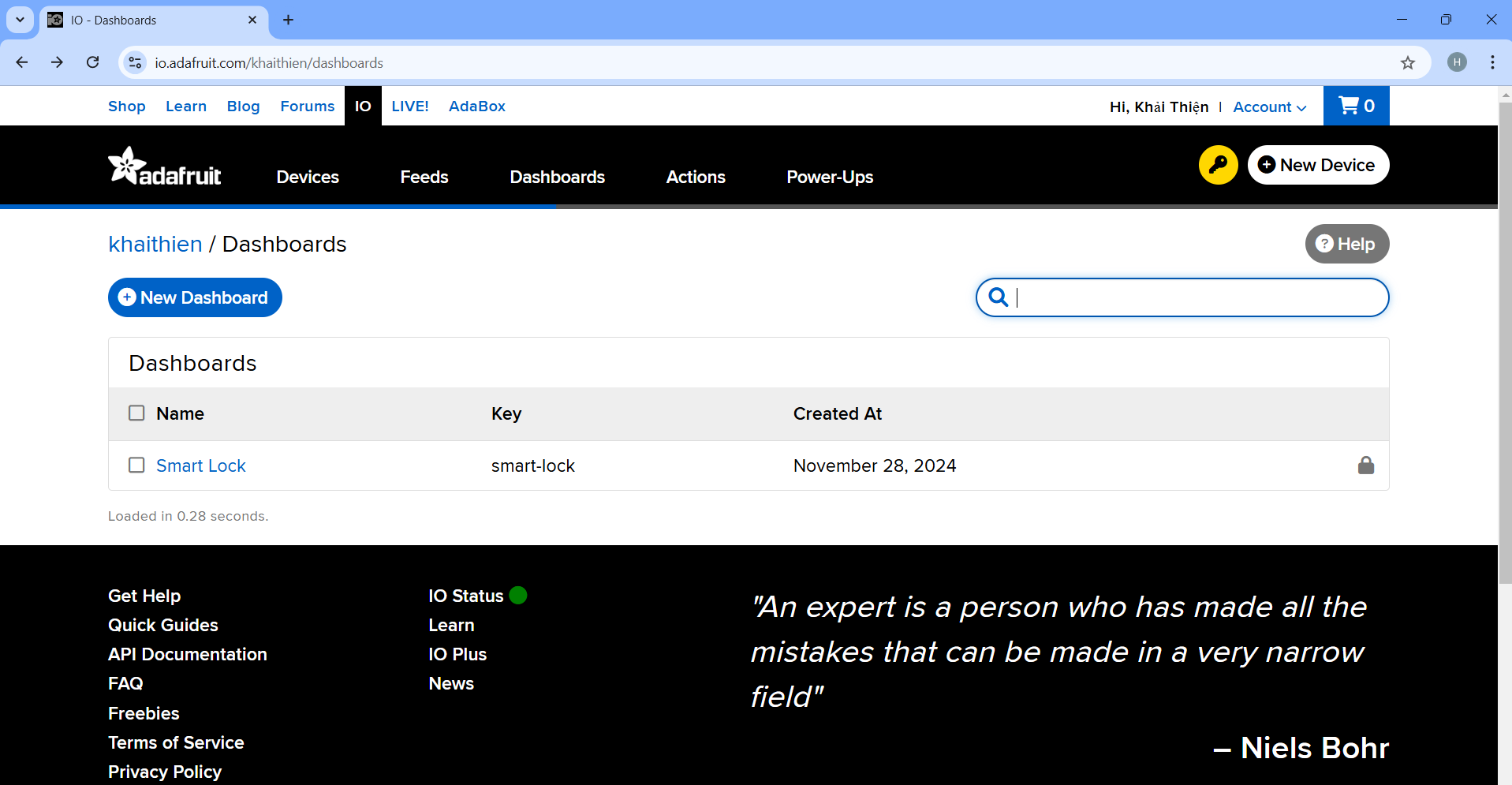
**Bước 3: Tạo Dashboard**

* Trước tiên vào phần **Dashboard** và chọn **New Dashboard**

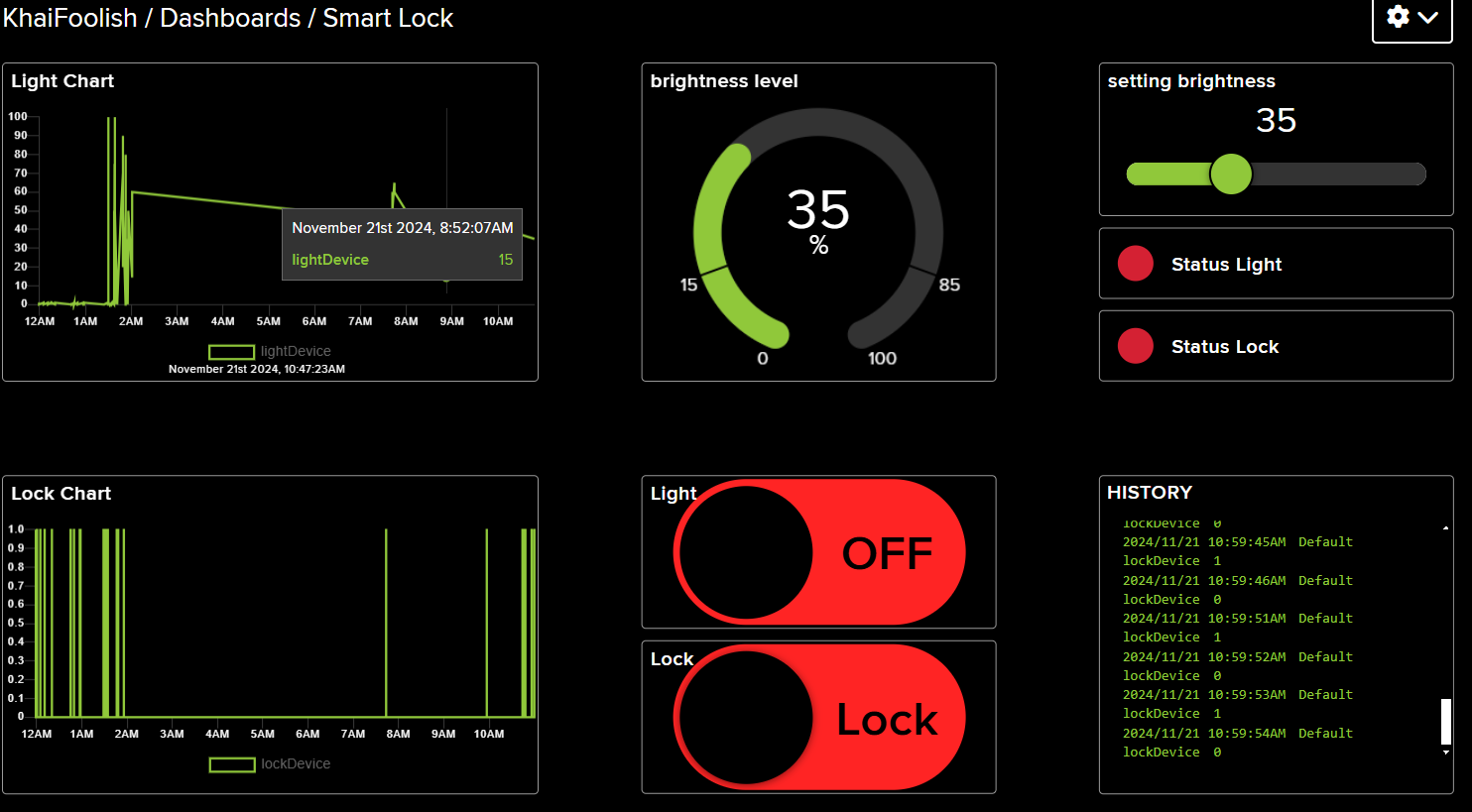




* Tiến hành nhập tên và tạo giống như phần **Feed**.

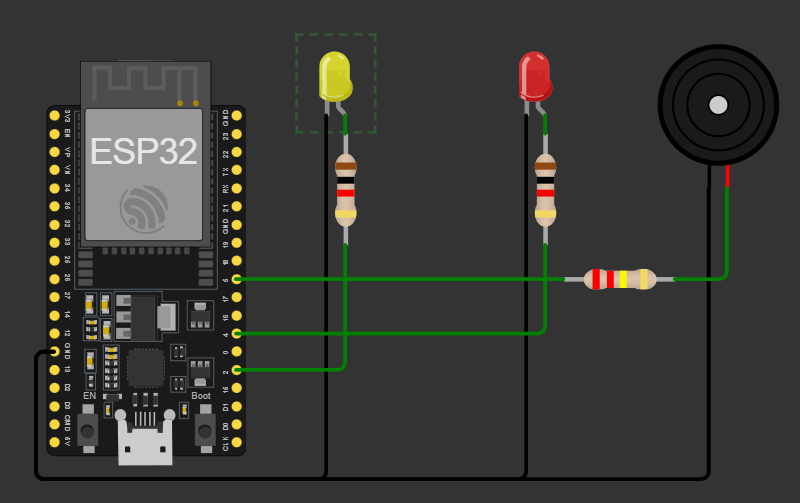


* Sau khi tạo xong thì nhấn vào và tạo dashboard. Và đây là phần Dashboard demo của mình



**Bước 4: Kết nối phần cứng và nạp code cho Esp32**

* Cài đặt thư viện **Adafruit IO Arduino** cho **Arduino IDE**
* Kết nối phần cứng như sơ đồ dưới đây:

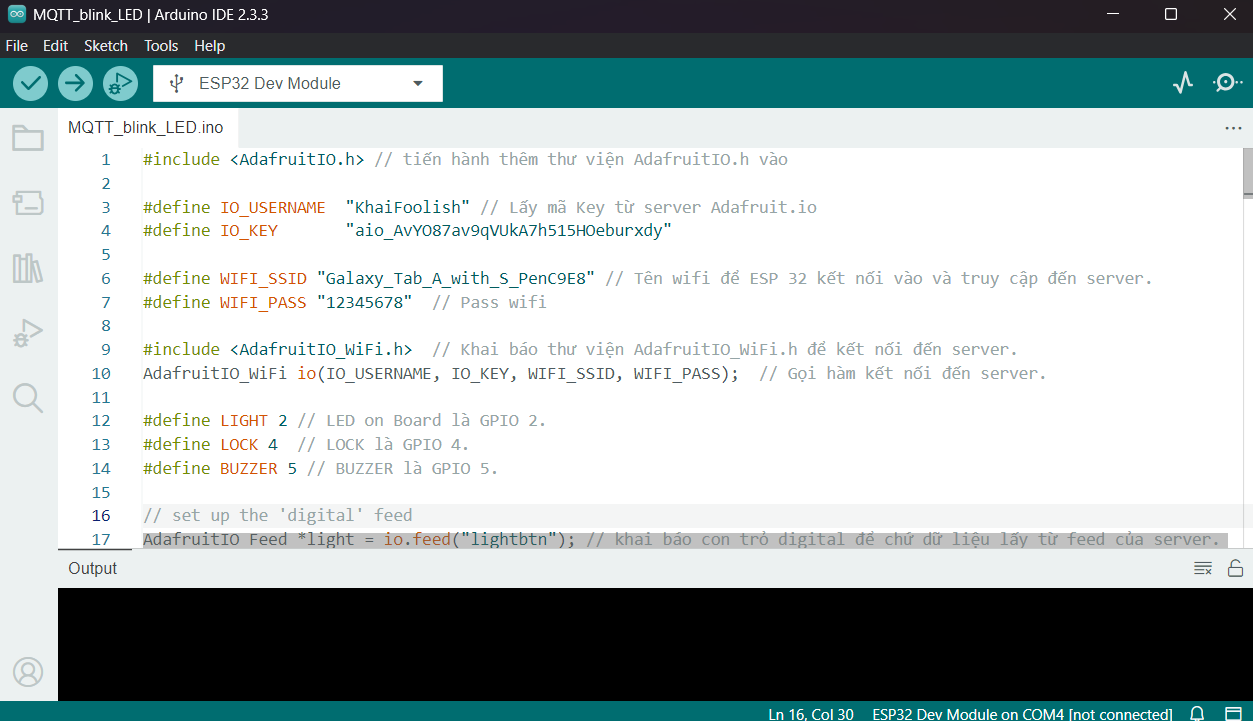
****

* Led màu vàng: nối GPIO 2 thể hiện đèn trong nhà
* Led màu đỏ: nối GPIO 4 thể hiện cho cửa (đèn sáng: cửa mở, đèn tắt: cửa đóng)
* Buzzer: nối GPIO 5 để báo hiệu tiếng khi cửa mở/đóng và khi cửa mở quá lâu sẽ phát tiếng kêu cảnh báo.
* Mở **Arduino IDE** và tạo một sketch mới với code như sau:

|  |
| --- |
| #include <AdafruitIO.h> // tiến hành thêm thư viện AdafruitIO.h vào  #define IO\_USERNAME  "KhaiFoolish" // Lấy mã Key từ server Adafruit.io  #define IO\_KEY       "aio\_AvYO87av9qVUkA7h515HOeburxdy"  #define WIFI\_SSID "Galaxy\_Tab\_A\_with\_S\_PenC9E8" // Tên wifi để ESP 32 kết nối vào và truy cập đến server.  #define WIFI\_PASS "12345678"  // Pass wifi  #include <AdafruitIO\_WiFi.h>  // Khai báo thư viện AdafruitIO\_WiFi.h để kết nối đến server.  AdafruitIO\_WiFi io(IO\_USERNAME, IO\_KEY, WIFI\_SSID, WIFI\_PASS);  // Gọi hàm kết nối đến server.  #define LIGHT 2 // LED on Board là GPIO 2.  #define LOCK 4  // LOCK là GPIO 4.  #define BUZZER 5 // BUZZER là GPIO 5.  // set up the 'digital' feed  AdafruitIO\_Feed \*light = io.feed("lightbtn"); // khai báo con trỏ digital để chứ dữ liệu lấy từ feed của server.  AdafruitIO\_Feed \*lock = io.feed("lockdevice");  bool doorOpen = false; // Trạng thái cửa  unsigned long startTime = 0;  bool waitingForClose = false; // Đợi đóng cửa  void setup() {      // set led pin as a digital output    pinMode(LIGHT, OUTPUT); // Khai báo output.    pinMode(LOCK, OUTPUT);    pinMode(BUZZER, OUTPUT); // Cấu hình buzzer (không bắt buộc, nhưng tốt cho khởi tạo)    noTone(BUZZER); // Tắt buzzer ban đầu      // start the serial connection    Serial.begin(115200);      // wait for serial monitor to open    while(! Serial);      // connect to io.adafruit.com    Serial.print("Connecting to Adafruit IO"); // tiến hành kết nối đến server.    io.connect(); // Gọi hàm kết nối        // wait for a connection    while(io.status() < AIO\_CONNECTED) {      Serial.print("."); // Nếu chưa kết nối được đến server sẽ tiến hành xuất ra màn hình đấu "."      delay(500);    }      // we are connected    Serial.println();    Serial.println(io.statusText()); // Nếu đã kết nối thành công tiến hành xuất ra màn hình trạng thái.    // set up a message handler for the 'digital' feed.    // the handleMessage function (defined below)    // will be called whenever a message is    // received from adafruit io.    light->onMessage(handleMessage); // Gọi hàm đọc dữ liệu và tiến hành điều khiển led và xuất ra trạng thái trên màn hình.    lock->onMessage(handleMessage);  }  void loop() {    io.run(); // gọi hàm Run.    if (waitingForClose && (millis() - startTime > 10000)) {      if (doorOpen) {        // Sau 10 giây cửa vẫn mở        Serial.println("Door still open, sounding alternating tones.");        while (doorOpen) { // Nếu cửa mở, kêu với tần số 1000Hz và 500Hz cách nhau 500ms          tone(BUZZER, 1000);          delay(500);          tone(BUZZER, 500);          delay(350);          io.run(); // Tiếp tục xử lý các feed trong lúc loop        }      }    }  }  // this function is called whenever an 'digital' feed message  // is received from Adafruit IO. it was attached to  // the 'digital' feed in the setup() function above.  void handleMessage(AdafruitIO\_Data \*data) { // hàm handleMessage để đọc dữ liệu.    Serial.print("Received <- ");    Serial.println(data->value()); // In ra giá trị nhận được từ feed    // Kiểm tra giá trị nhận được từ feed và điều khiển tương ứng    if (strcmp(data->feedName(), "lightbtn") == 0) {      // Điều khiển thiết bị 'lightbtn'      digitalWrite(LIGHT, data->toPinLevel());      Serial.print("Light is ");      Serial.println(data->toPinLevel() == HIGH ? "ON" : "OFF");    }    else if (strcmp(data->feedName(), "lockdevice") == 0) {      // Điều khiển khóa      digitalWrite(LOCK, data->toPinLevel());      Serial.print("Lock is ");      Serial.println(data->toPinLevel() == HIGH ? "UNLOCKED" : "LOCKED");      doorOpen = (data->toPinLevel() == HIGH); // TRUE nếu cửa mở      if (doorOpen) { // Cửa mở        Serial.println("Buzzer ON (HIGH Frequency) - Door is OPEN");        // Buzzer phát âm tần số cao (1000 Hz)        for (int i = 0; i < 2; i++){          tone(BUZZER, 1000);          delay(50);          noTone(BUZZER);          delay(50);        }        startTime = millis(); // Bắt đầu đếm 10 giây        waitingForClose = true;      } else { // Cửa đóng        waitingForClose = false;        Serial.println("Buzzer ON (LOW Frequency) - Door is CLOSED");        // Buzzer phát âm tần số thấp (750 Hz)        for (int i = 0; i < 2; i++){          tone(BUZZER, 750);          delay(50);          noTone(BUZZER);          delay(50);        }      }    }    else {      Serial.println("Unknown device command.");    }  } |

* Thay thế phần tên wifi, mật khẩu wifi, IO\_USERNAME và IO\_KEY







* Kiểm tra kết quả chương trình:
  + Nạp sketch vào ESP32
  + Mở serial monitor để xem các thông báo từ ESP32
  + Mở và kết nối với broker MQTT (**Adafruit**)