

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



PHÁT TRIỂN ỨNG DỤNG INTERNET OF THINGS THỰC HÀNH
(CO3038)

LAB 01 SỬ DỤNG ADAFRUIT IO SERVER

BỘ MÔN: KỸ THUẬT MÁY TÍNH

GVHD: VŨ TRỌNG THIÊN

NHÓM LỚP: L01

—o0o—

SVTH: MAI THỊNH PHÁT (1914590)

TP. HỒ CHÍ MINH, 12/2023

Mục lục

1	Tổng quan	1
1.1	Giới thiệu về Adafruit IO server	1
1.1.1	Chức năng của Adafruit IO	1
1.1.2	Ưu điểm của dịch vụ Adafruit IO	1
1.2	Cấu trúc bài báo cáo	2
2	Khởi tạo dự án	3
2.1	Đăng ký tài khoản	3
2.2	Khởi tạo các kênh dữ liệu	4
3	Tạo giao diện theo dõi	7
3.1	Giao diện đồ thị đường	8
3.2	Giao diện Gauge	10
4	Kết nối với server Adafruit IO	13
4.1	Chuẩn bị	13
4.2	Hiện thực mã nguồn	14
4.2.1	Khai báo thư viện và khởi tạo khóa	14
4.2.2	Tạo hàm gọi lại khi có sự kiện xảy ra	14
4.2.3	Xử lý chính của chương trình	15
4.3	Kết quả	16
4.4	Repo mã nguồn	17
5	Repo mã nguồn	18

Danh sách hình vẽ

2.1	Ví dụ về tạo tài khoản trên Adafruit IO	3
2.2	Giao diện hệ thống sau khi đã đăng ký tài khoản thành công	4
2.3	tạo một kênh dữ liệu	4
2.4	tạo một kênh dữ liệu	5
2.5	Kênh dữ liệu mặc định Private	5
2.6	Chuyển kênh dữ liệu sang Public	6
2.7	Kết quả của việc khởi tạo các kênh dữ liệu	6
3.1	Tạo dashboard	7
3.2	Tạo dashboard	8
3.3	Tạo linechart	8
3.4	Liên kết giao diện với kênh dữ liệu cần thiết	9
3.5	Điều thông tin mô tả giao diện	9
3.6	Line chart	10
3.7	Liên kết giao diện với kênh dữ liệu cần thiết	10
3.8	Liên kết giao diện với kênh dữ liệu cần thiết	11
3.9	Điều thông tin mô tả giao diện	11
3.10	Gauge chart	12
3.11	Hoàn tất tạo các thành phần giao diện	12
4.1	Lấy key	13
4.2	Thêm thư viện	14
4.3	tạo callback kết nối	14
4.4	tạo callback đăng ký	14
4.5	tạo callback message	15
4.6	tạo callback disconnect	15
4.7	tạo client	15
4.8	Gửi dữ liệu	16
4.9	Result dashboard	17
4.10	Result terminal	17

Chương 1

Tổng quan

1.1 Giới thiệu về Adafruit IO server

Adafruit IO được tạo ra bởi Adafruit Industries, một công ty chuyên về các sản phẩm và dịch vụ dành cho Internet of Things (IoT). Nền tảng này được phát hành lần đầu tiên vào năm 2014, với mục tiêu cung cấp một cách dễ dàng và hiệu quả để kết nối các thiết bị IoT với internet.

Adafruit IO là một dịch vụ đám mây được thiết kế để kết nối thiết bị IoT (Internet of Things) và thu thập dữ liệu từ chúng. Dịch vụ này giúp đơn giản hóa việc gửi và nhận dữ liệu từ các thiết bị thông qua các giao thức như MQTT (Message Queuing Telemetry Transport).

1.1.1 Chức năng của Adafruit IO

Adafruit IO cung cấp một số chức năng chính sau:

- *Kết nối thiết bị với internet:* Adafruit IO cung cấp giao thức MQTT để giao tiếp giữa các thiết bị và nền tảng. Giao thức MQTT là một giao thức nhẹ và hiệu quả, phù hợp với các ứng dụng IoT.
- *Thu thập dữ liệu:* Adafruit IO cho phép bạn thu thập dữ liệu từ các thiết bị của mình. Dữ liệu có thể bao gồm nhiệt độ, độ ẩm, ánh sáng,...
- *Lưu trữ dữ liệu:* Adafruit IO lưu trữ dữ liệu trên hệ thống lưu trữ đám mây (Cloud storage) của Adafruit IO trong 30 hoặc 60 ngày.
- *Phân tích dữ liệu:* Adafruit IO cung cấp một số tính năng phân tích và tương tác với dữ liệu, giúp các lập trình viên, nhà quản trị hệ thống hiểu rõ hơn về dữ liệu mà họ thu thập.

1.1.2 Ưu điểm của dịch vụ Adafruit IO

Dưới đây là một số lý do làm cho Adafruit IO trở nên được ưa chuộng trong các hệ thống IOT:

- *Dễ sử dụng:* Adafruit IO có giao diện người dùng rất trực quan và dễ sử dụng.
- *Hỗ trợ nhiều loại thiết bị:* Adafruit IO hỗ trợ nhiều loại thiết bị IoT, bao gồm Arduino, Raspberry Pi, ESP8266,...

-
- *Giao thức MQTT*: Adafruit IO sử dụng giao thức MQTT để giao tiếp giữa các thiết bị và nền tảng. Giao thức MQTT là một giao thức nhẹ và hiệu quả, phù hợp với các ứng dụng IoT.
 - *Hỗ trợ Cloud storage* : Adafruit IO hỗ trợ lưu trữ dữ liệu bằng nền tảng Cloud storage trong 30 ngày hoặc 60 ngày.

1.2 Cấu trúc bài báo cáo

Kết cấu của bài thực hành này sẽ được trình bày trong báo cáo như sau:

- **Chương 1 Tổng quan**: Giới thiệu về Adafruit IO server
- **Chương 2 Khởi tạo dự án**: Giới thiệu về các bước khởi tạo các thành phần hỗ trợ cho bài thực hành
- **Chương 3 Phân tích và thiết kế hệ thống**: Tìm hiểu các hệ thống liên quan, rút ra ưu nhược của nó , đồng thời trình bày về các phân tích và thiết kế hệ thống đã sử dụng trong đề tài này.
- **Chương 4 Bài toán và giải pháp**: Trình bày các vấn đề , bài toán được đặt ra đối với đề tài và cách giải quyết chúng.
- **Chương 5 Hiện thực hệ thống**: Trình bày sơ lược kết quả của quá trình hiện thực hệ thống.
- **Chương 6 Kiểm thử hệ thống**: Trình bày về phương pháp kiểm thử đã áp dụng và kết quả thu được.
- **Chương 7 Tổng kết**: Tóm tắt các kết quả đạt được, hạn chế còn tồn đọng và hướng phát triển hệ thống trong tương lai.

Chương 2

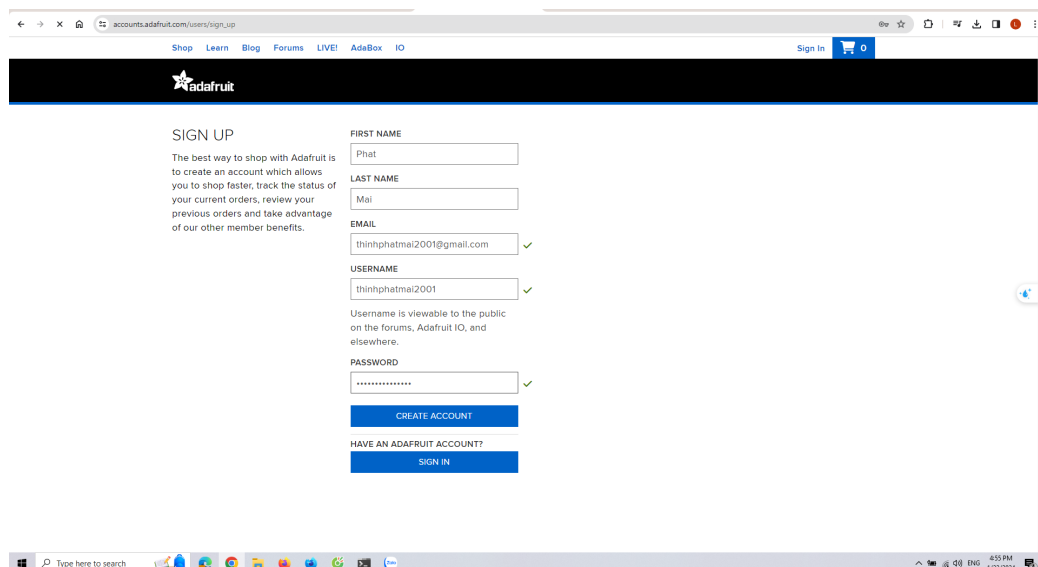
Khởi tạo dự án

Trong phần này em sẽ trình bày các phần phần cơ bản làm nền tảng cho các phần sau cụ thể là: Khởi tạo tài khoản trên Adafruit IO, tạo các kênh dữ liệu.

2.1 Đăng ký tài khoản

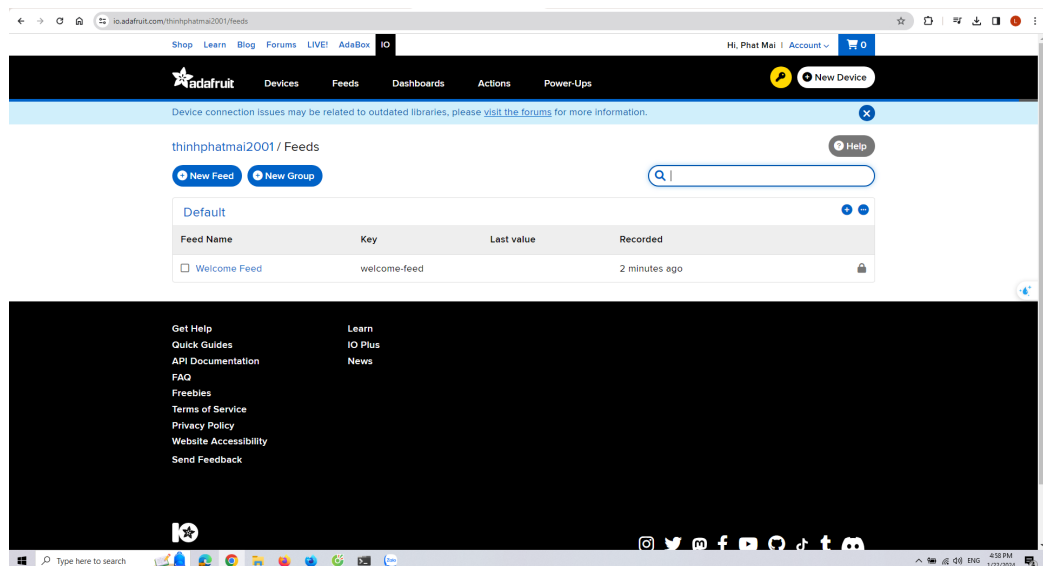
Để đăng ký tài khoản người dùng tiến hành vào trang đăng ký của *Adafruit IO* tại địa chỉ https://accounts.adafruit.com/users/sign_up để tiến hành tạo một tài khoản.

Sau khi đã vào được trang đăng ký người dùng tiến hành nhập các trường thông tin được yêu cầu để đăng ký tài khoản. Dưới đây là ví dụ:



Hình 2.1: Ví dụ về tạo tài khoản trên Adafruit IO

Sau khi đã tạo thành công tài khoản hệ thống sẽ tự động tiến hành đăng nhập bằng tài khoản mới tạo. Dưới đây là giao diện của trang chủ sau khi đã tạo thành công tài khoản.

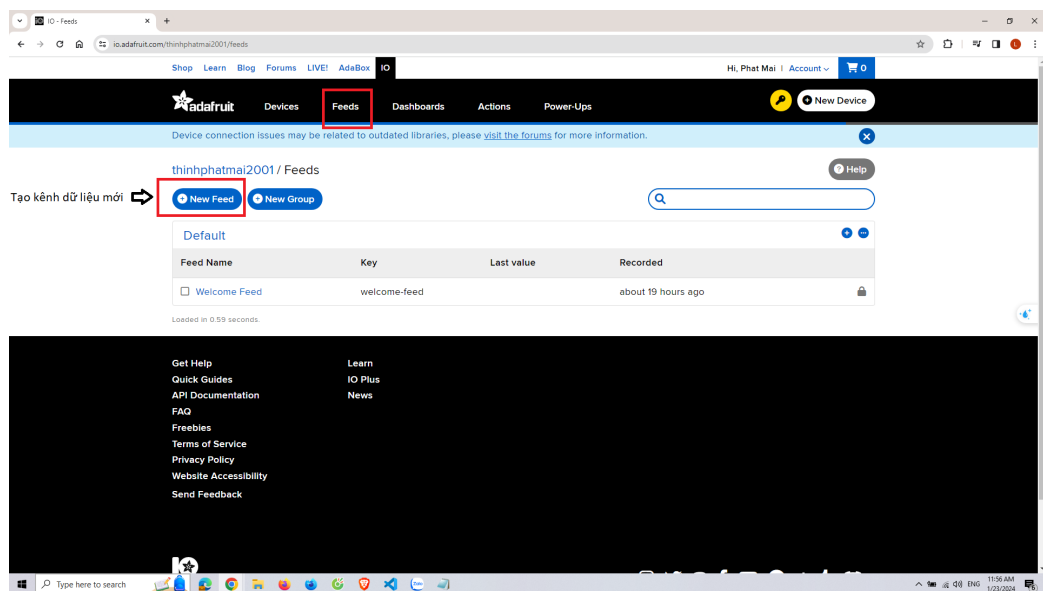


Hình 2.2: Giao diện hệ thống sau khi đã đăng ký tài khoản thành công

2.2 Khởi tạo các kênh dữ liệu

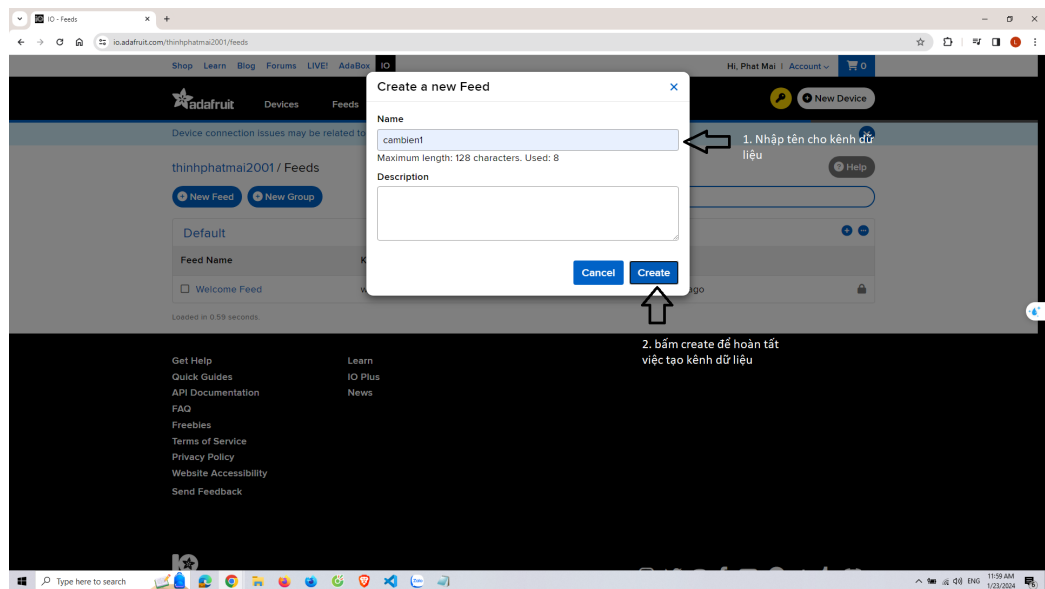
Trong phần này em sẽ trình bày các bước để khởi tạo một kênh dữ liệu trên Adafruit IO server. Để tạo một kênh dữ liệu ta sẽ lần lượt làm theo các bước sau:

1. chọn **Feeds** để mở danh sách các kênh dữ liệu hiện có. Sau đó chọn **New Feed** để tiến hành tạo một kênh dữ liệu mới



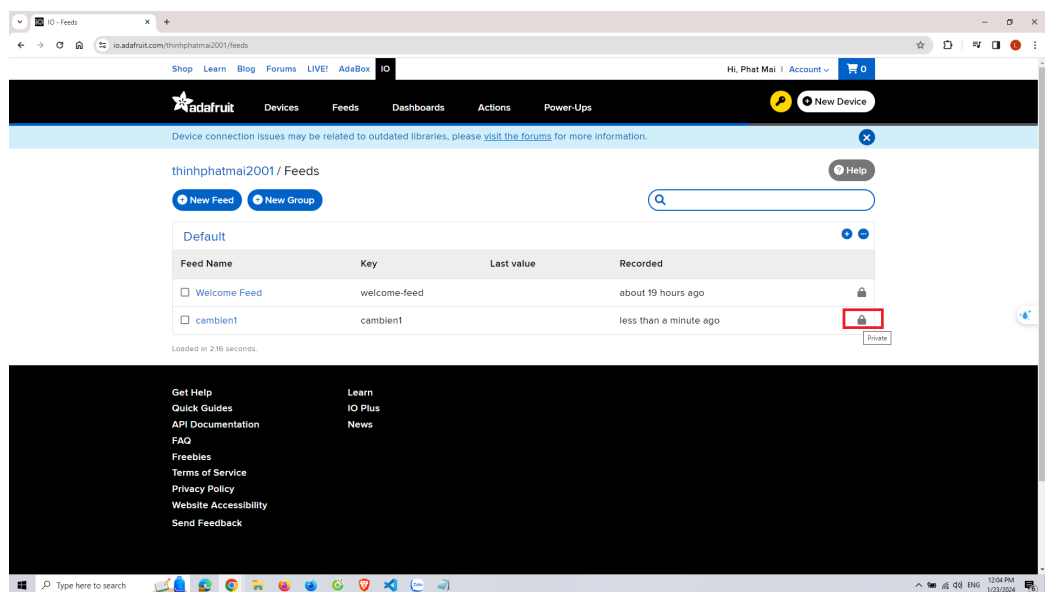
Hình 2.3: tạo một kênh dữ liệu

2. Tiến hành nhập tên của kênh dữ liệu sau đó nhấn **Create** để hoàn tất việc tạo kênh dữ liệu

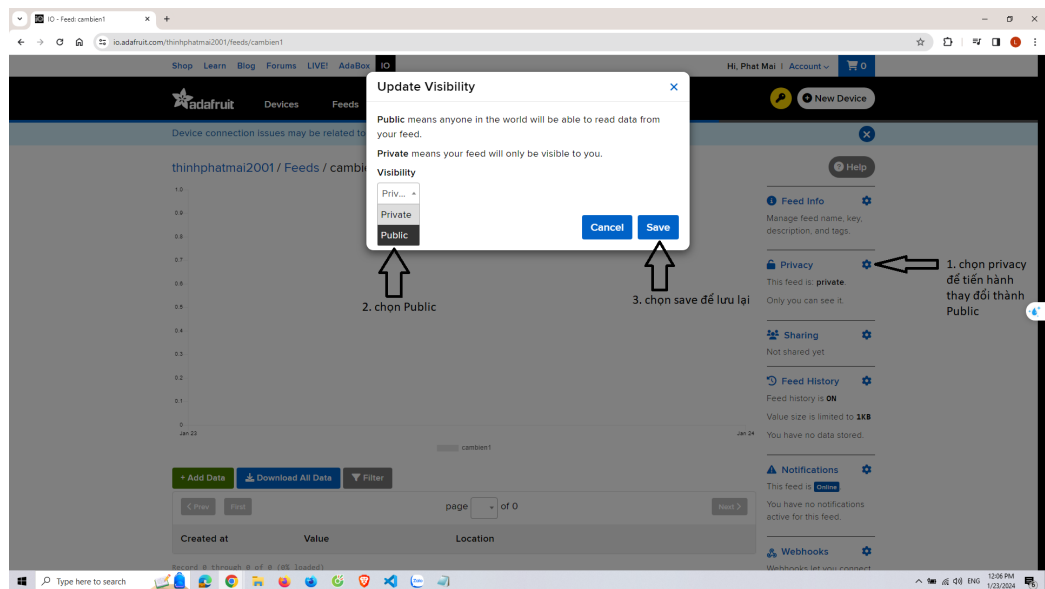


Hình 2.4: tạo một kênh dữ liệu

3. Thường các kênh dữ liệu mới tạo ra sẽ mặc định ở trạng thái là **Private** . Để thuận tiện cho các bước sau ta sẽ tiến hành chuyển thành **Public** như sau.

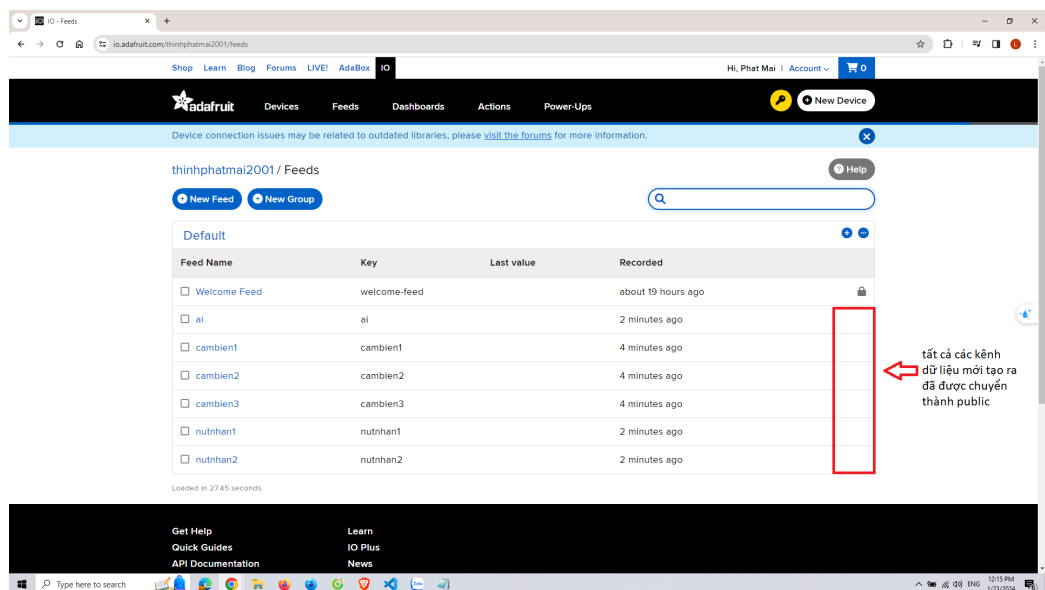


Hình 2.5: Kênh dữ liệu mặc định Private



Hình 2.6: Chuyển kênh dữ liệu sang Public

Sau khi đã khởi tạo các kênh dữ liệu cần thiết và chuyển các kênh dữ liệu sang *Public* ta thu được kết quả sau:



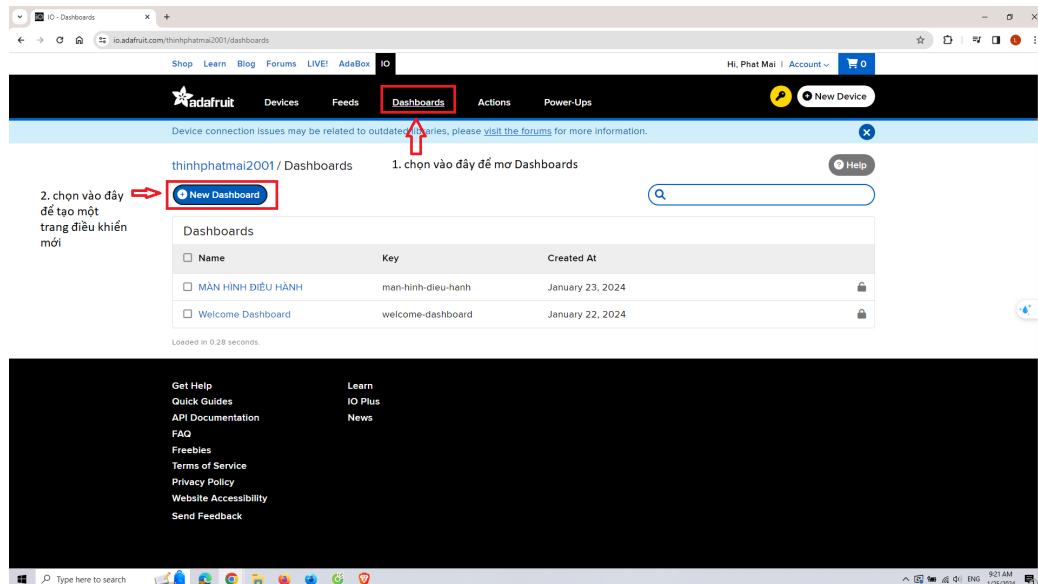
Hình 2.7: Kết quả của việc khởi tạo các kênh dữ liệu

Chương 3

Tạo giao diện theo dõi

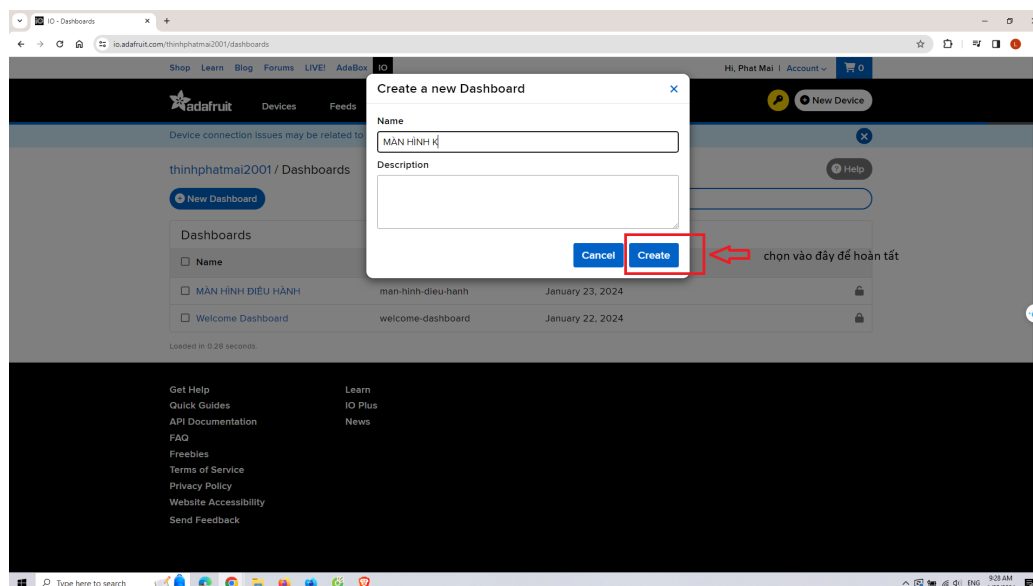
Trong phần này em sẽ trình bày các bước tạo giao diện điều khiển trên *Adafruit IO* để theo dõi các kênh dữ liệu đã tạo ra ở phần trước.

Để tạo mới một trang giao diện người dùng ta sẽ tiến hành vào **Dashboards** vào sau đó chọn **New dashboard** để tạo mới một trang giao diện.



Hình 3.1: Tạo dashboard

Sau đó ta tiến hành nhập tên cho Dashboard và chọn **create** để hoàn tất việc tạo trang giao diện

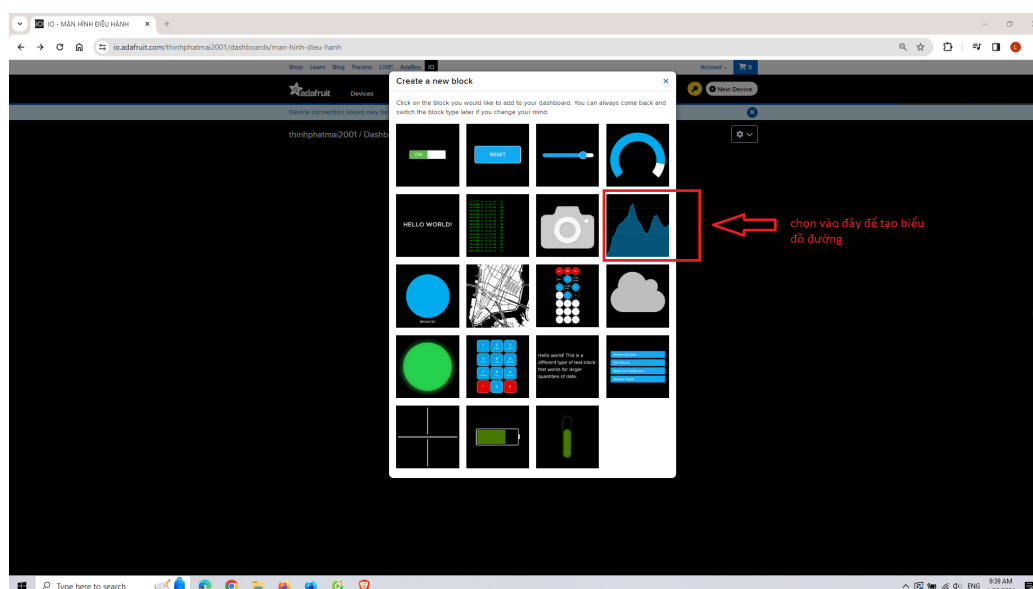


Hình 3.2: Tạo dashboard

Sau khi hoàn tất tạo trang giao diện chúng ta sẽ vào trang giao diện đó và tiến hành tạo từng thành phần cho giao diện. Bằng bấm vào biểu tượng bánh răng vào chọn **Create new block** để lựa chọn thành phần giao diện muốn hiển thị.

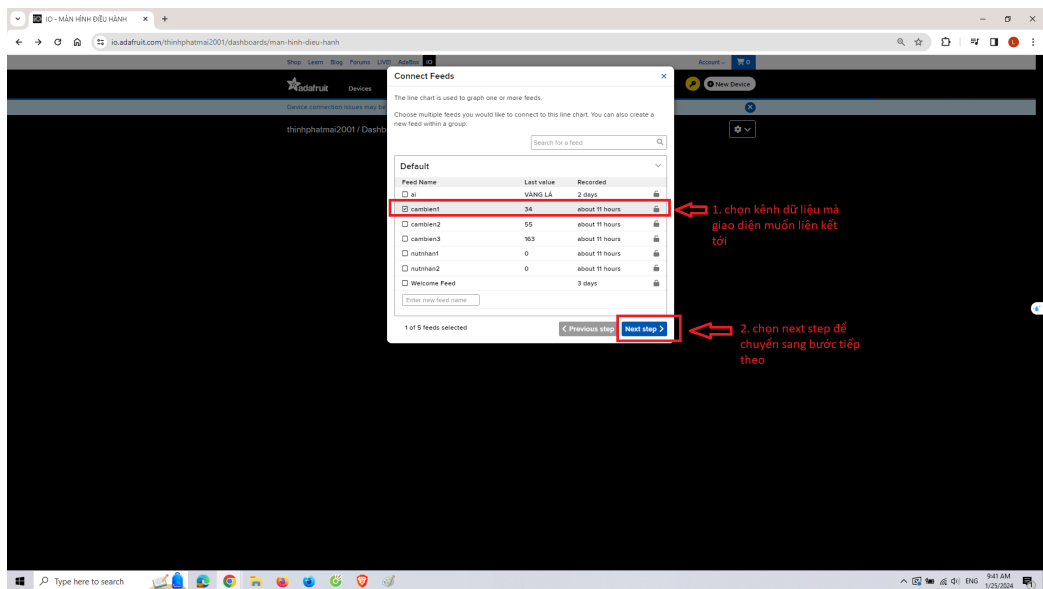
3.1 Giao diện đồ thị đường

Để tạo ra giao diện đồ thị đường ta chọn thành phần giao diện là **Line chart**.



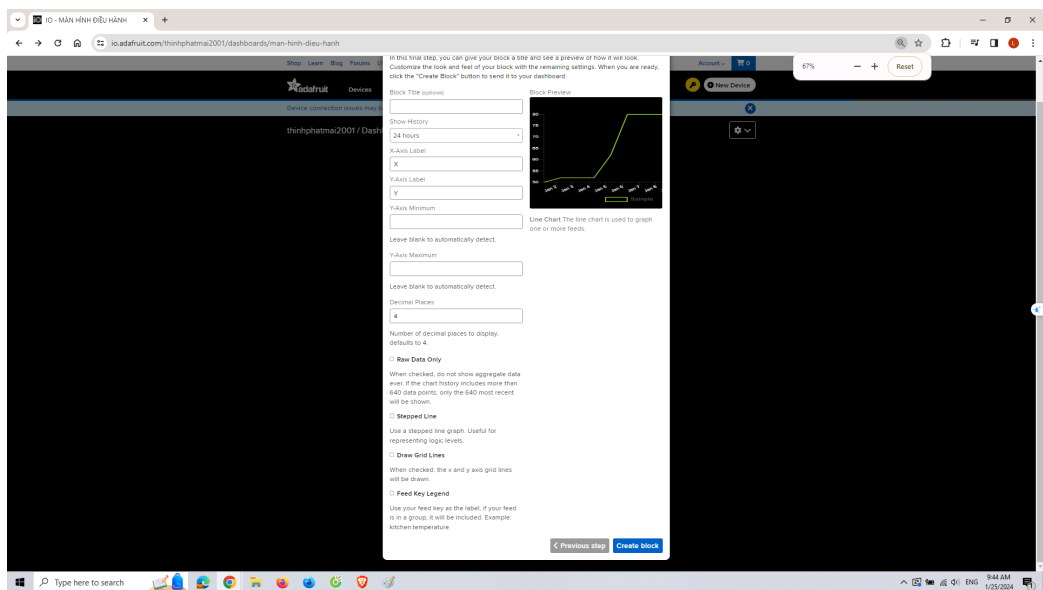
Hình 3.3: Tạo linechart

Tiếp theo kết nối giao diện với kênh dữ liệu bằng cách chọn vào kênh dữ liệu cần liên kết với vào diện vào chọn **Next step** để tiếp tục.



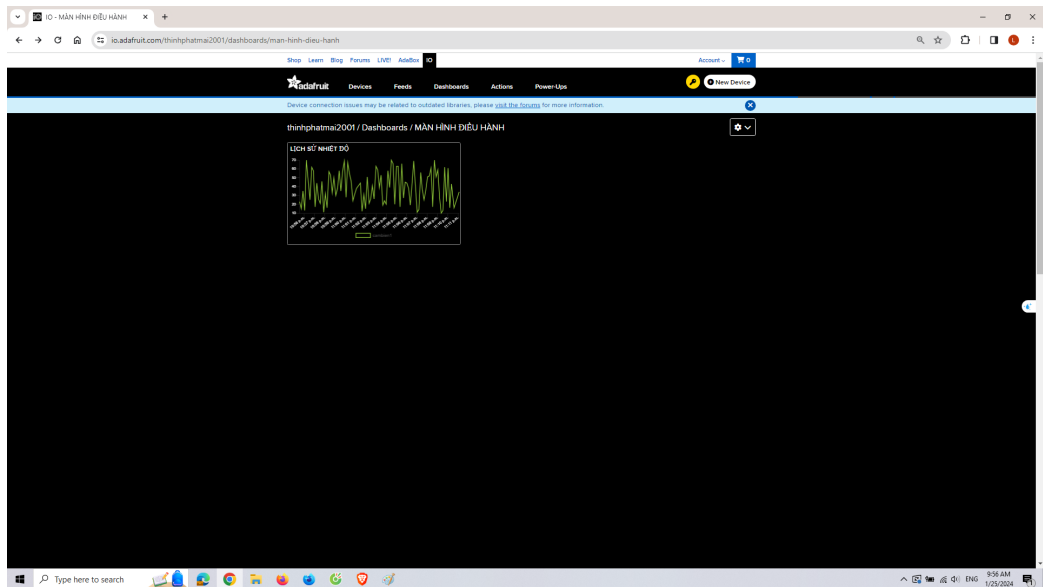
Hình 3.4: Liên kết giao diện với kênh dữ liệu cần thiết

Tiến hành nhập các thông tin hiển thị của giao diện và chọn **Create block** để hoàn tất tạo ra biểu đồ đường



Hình 3.5: Điều chỉnh thông tin mô tả giao diện

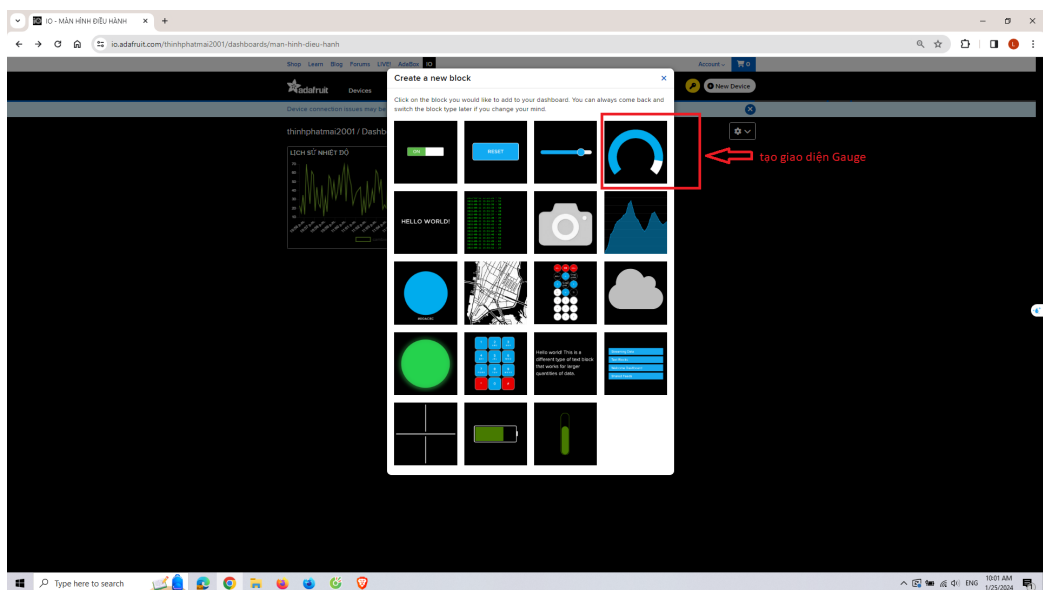
Kết quả biểu đồ đường đã được tạo ra



Hình 3.6: Line chart

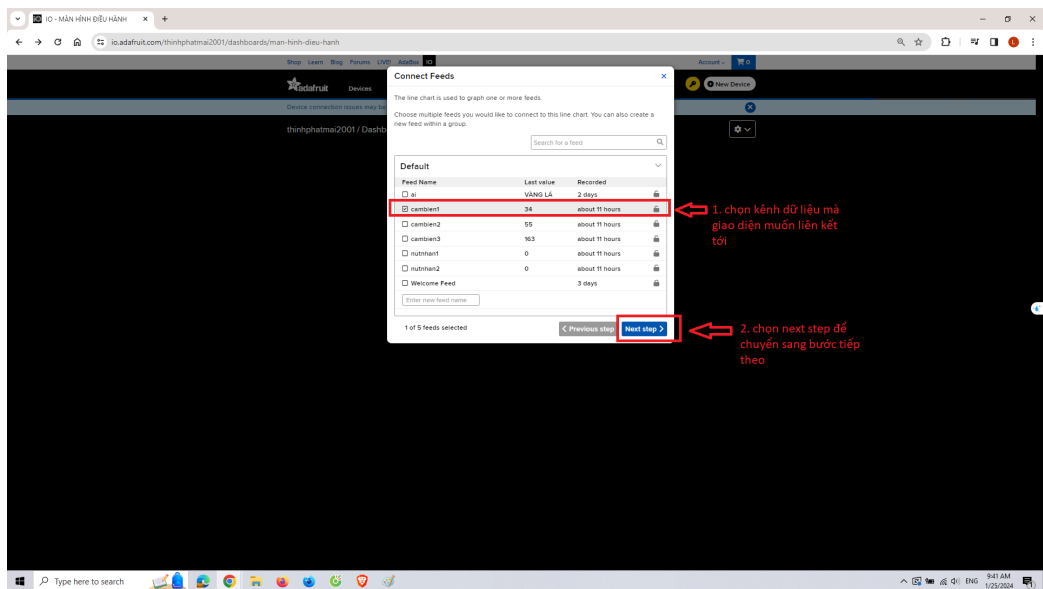
3.2 Giao diện Gauge

Ta chọn **Gauge** để tạo ra giao diện hiển thị số đo hiện tại.



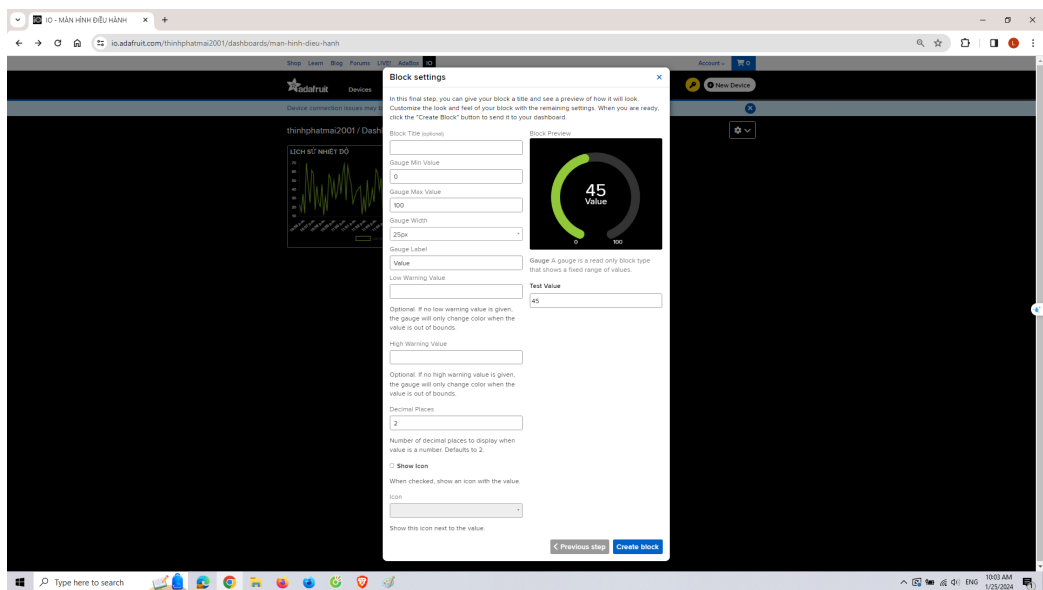
Hình 3.7: Liên kết giao diện với kênh dữ liệu cần thiết

Tiếp theo kết nối giao diện với kênh dữ liệu bằng cách chọn vào kênh dữ liệu cần liên kết với vào diện vào chọn **Next step** để tiếp tục.



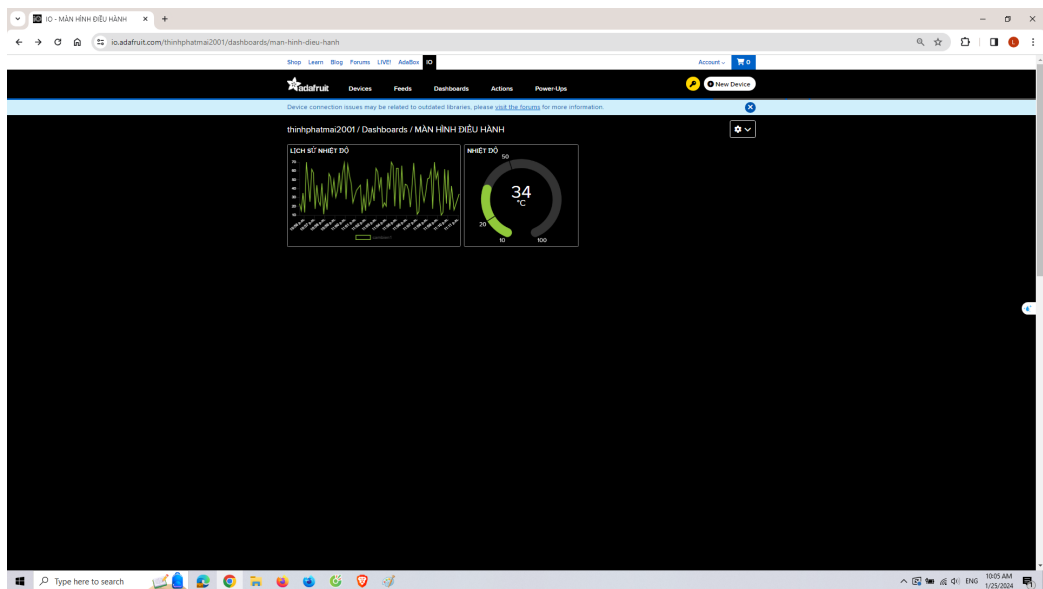
Hình 3.8: Liên kết giao diện với kênh dữ liệu cần thiết

Tiến hành nhập các thông tin hiển thị của giao diện và chọn **Create block** để hoàn tất tạo ra biểu đồ **Gauge**



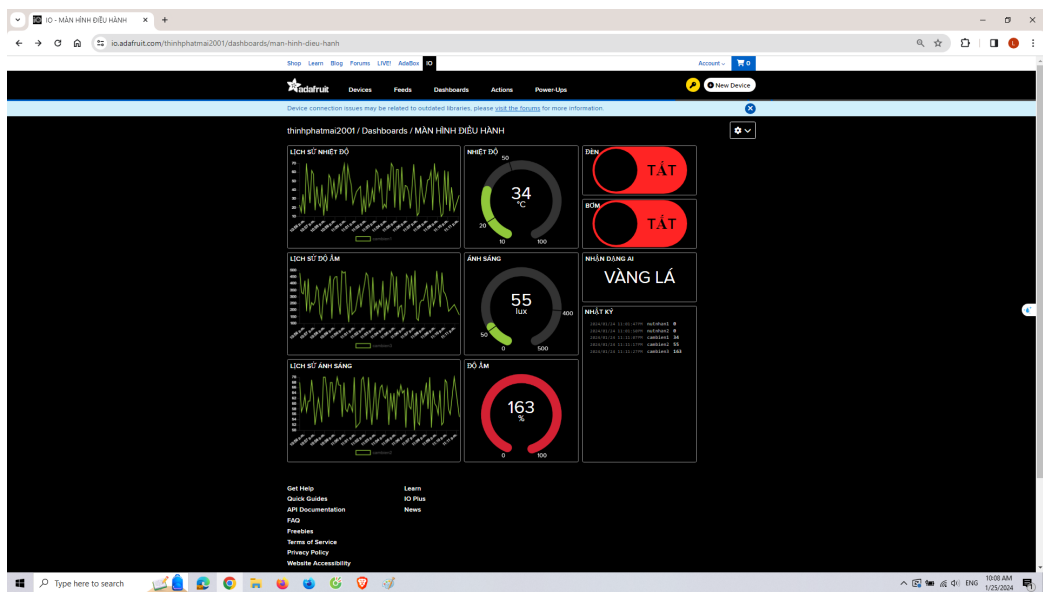
Hình 3.9: Điều chỉnh thông tin mô tả giao diện

Kết quả biểu đồ Gauge đã được tạo ra



Hình 3.10: Gauge chart

Tương tự cho các thành phần giao diện khác và sau đây là kết quả.



Hình 3.11: Hoàn tất tạo các thành phần giao diện

Chương 4

Kết nối với server Adafruit IO

Trong chương này em sẽ trình bày cách kết nối với server *Adafruit IO* bằng chương trình Python cơ bản

4.1 Chuẩn bị

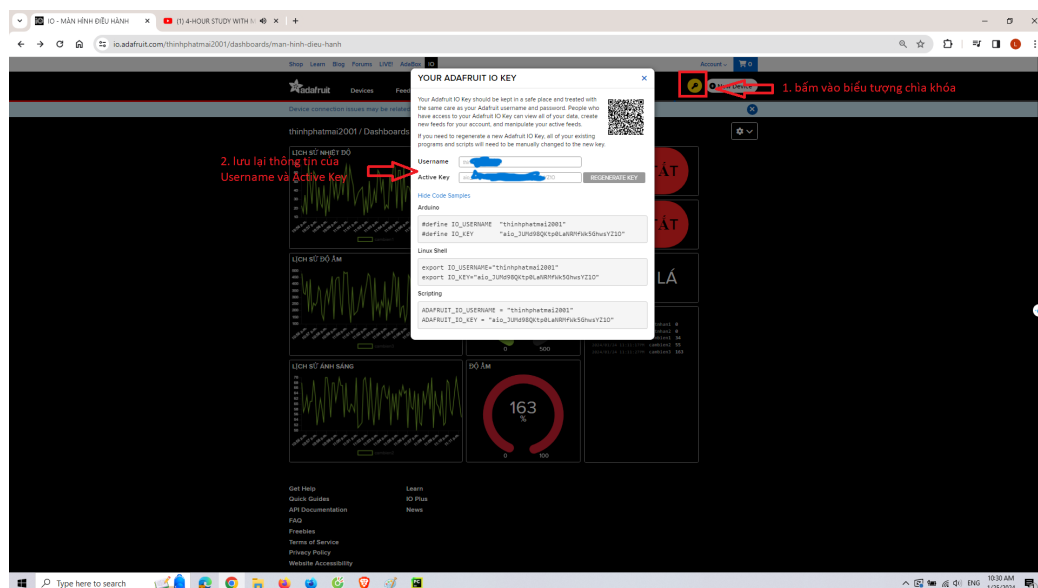
Để kết nối với server *Adafruit IO* ta cần chuẩn bị

1. cài đặt thư viện **adafruit-io**
2. Key của server

Để tiến hành cài đặt thư viện **adafruit-io** ta chạy câu lệnh sau.

```
pip install adafruit-io
```

Để tiến hành lấy key ta chọn vào biểu tượng chìa khóa và lưu lại thông tin ở trường **Username** và **Active Key**



Hình 4.1: Lấy key

4.2 Hiện thực mã nguồn

4.2.1 Khai báo thư viện và khởi tạo khóa

```
1 import sys
2 from Adafruit_IO import MQTTClient
3 import time
4 import random
5
6 AIO_FEED_IDS = ['nutnhan1', 'nutnhan2']
7 AIO_USERNAME = "th          1"
8 AIO_KEY = "aio_                                .0"
9
```

Hình 4.2: Thêm thư viện

Ở hình trên ta sẽ tiến hành thêm thư viện **adafruit-io** và lưu **Key** đã lấy ở bước trên vào hai biến **AIO_USERNAME** và **AIO_KEY** để khởi tạo kết nối ở bước sau. Biến **AIO_FEED_IDS** dùng để lưu danh sách các topic mà chúng ta sẽ subscribe.

4.2.2 Tạo hàm gọi lại khi có sự kiện xảy ra

Phần này chúng ta sẽ tạo ra các hàm gọi lại (*call back*) dùng để gọi khi có các sự kiện **kết nối, đăng ký (subscribe), nhận dữ liệu từ server, ngắt kết nối**.

- Đầu tiên là **hàm kết nối**.

```
def connected(client):
    print("Ket noi thanh cong ...")
    for topic in AIO_FEED_IDS:
        client.subscribe(topic)
```

Hình 4.3: tạo callback kết nối

Hàm này sẽ được chạy khi kết nối thành công. Ngay sau khi kết nối thành công chúng ta sẽ đăng ký (subscribe) vào hai kênh dữ liệu là **nutnhan1** và **nutnhan2** để có thể lắng nghe trạng thái từ hai kênh này

- **hàm đăng ký**

```
def subscribe(client, userdata, mid, granted_qos):
    print("Subscribe thanh cong ...")
```

Hình 4.4: tạo callback đăng ký

Hàm này sẽ được gọi ngay khi đăng ký thành công vào các kênh dữ liệu.

- hàm lắng nghe khi có dữ liệu đến

```
def message(client, feed_id, payload):  
    print("Nhan du lieu: " + payload + " , feed id:"+feed_id)
```

Hình 4.5: tạo callback message

Hàm này sẽ được gọi ngay có dữ liệu mới được trên server. Tham số **payload** lưu thông tin dữ liệu và **feed_id** lưu thông tin kênh dữ liệu

- hàm ngắt kết nối

```
def message(client, feed_id, payload):  
    print("Nhan du lieu: " + payload + " , feed id:"+feed_id)
```

Hình 4.6: tạo callback disconnect

Hàm này sẽ được gọi ngay khi kết nối giữ server và client bị ngắt.

4.2.3 Xử lý chính của chương trình

Ngay khi có các hàm callback để tiến hành kết nối với server ta sẽ tạo ra một client thuộc lớp *MQTTClient* và truyền các key của server vào làm tham số cho hàm khởi tạo đối tượng. Sau đó ta sẽ gán các hàm callback đã tạo ra ở bước trên vào các thuộc tính của đối tượng **client**

```
client = MQTTClient(AIO_USERNAME, AIO_KEY)  
client.on_connect = connected  
client.on_disconnect = disconnected  
client.on_message = message  
client.on_subscribe = subscribe  
client.connect()  
client.loop_background()
```

Hình 4.7: tạo client

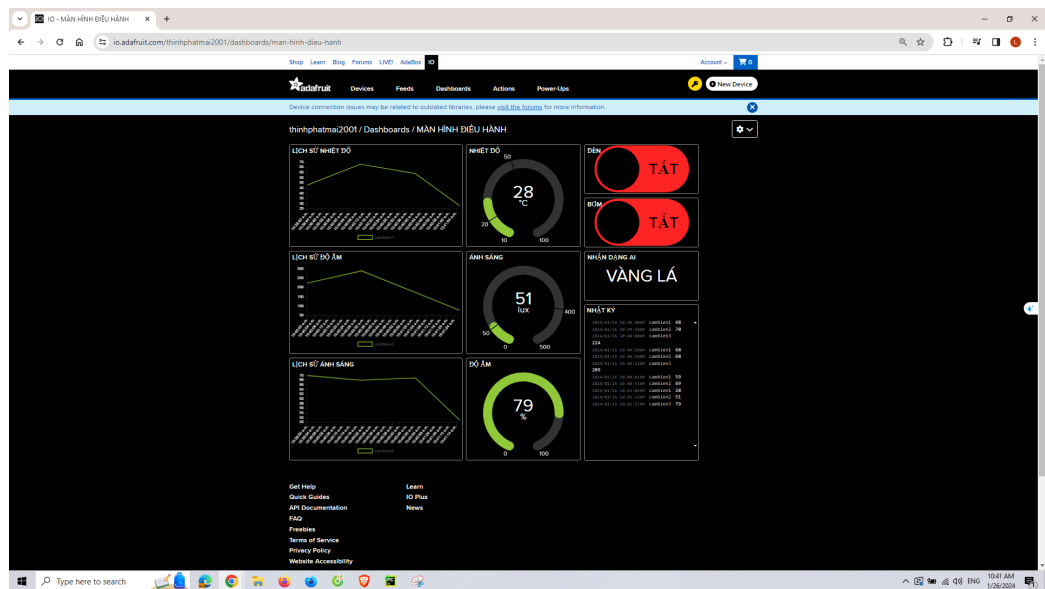
Sau đây ta sẽ tiến hành gửi các dữ liệu lên server tuần tự cho từng kênh dữ liệu cách nhau 10 giây:

```
counter = 10
sensor_type = 0
while True:
    counter -= 1
    if counter <= 0:
        counter = 10
        # todo
        print("Random data is publishing")
        if sensor_type == 0:
            print("temperture")
            temp = random.randint(a: 10, b: 70)
            client.publish(feed_id: 'cambien1', temp)
            sensor_type = 1
        elif sensor_type == 1:
            print("humidity")
            humi = random.randint(a: 50, b: 70)
            client.publish(feed_id: 'cambien2', humi)
            sensor_type = 2
        elif sensor_type == 2:
            print("light")
            light = random.randint(a: 100, b: 500)
            client.publish(feed_id: 'cambien3', light)
            sensor_type = 0
    time.sleep(1)
    pass
```

Hình 4.8: Gửi dữ liệu

4.3 Kết quả

Dưới đây là kết quả chụp được từ giao diện dashboard trên AdafruitIO



Hình 4.9: Result dashboard

Dưới đây là kết quả của việc nhận dữ liệu được server publish

```
Run main x
Random data is publishing
light
Nhan du lieu: 1 , feed id:nutnhan1
Random data is publishing
temperture
Nhan du lieu: 1 , feed id:nutnhan2
Nhan du lieu: 0 , feed id:nutnhan2
Nhan du lieu: 0 , feed id:nutnhan1
Nhan du lieu: 1 , feed id:nutnhan1
Random data is publishing
humidity
```

Hình 4.10: Result terminal

4.4 Repo mã nguồn

Mã nguồn và báo cáo đều được lưu trữ tại: aaaa

Chương 5

Repo mã nguồn

Tài liệu tham khảo

- [1] The Python Tutorial
<https://docs.python.org/3/tutorial/index.html> [Truy cập: 20-01-2024]
- [2] Python Tutorial
<https://www.w3schools.com/python/> [Truy cập: 20-01-2024]
- [3] adafruit-io python
<https://pypi.org/project/adafruit-io/> [Truy cập: 10-09-2023]