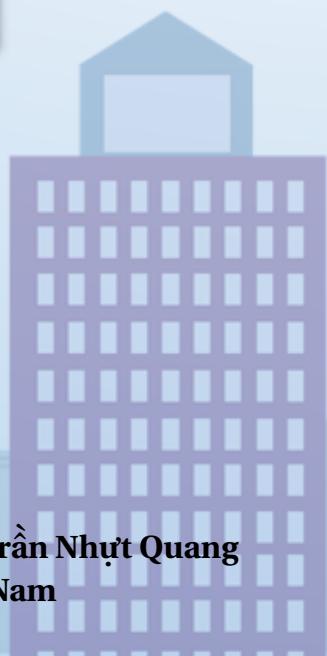




THE DARIU FOUNDATION

Phát triển Gateway IoT bằng Python



Nguyễn Thiên Ân - Phạm Thanh Danh - Huỳnh Nhữ Hùng - Trần Nhựt Quang
Nguyễn Văn Hạnh - Lê Trọng Nhân - Lê Phương Nam

Mục lục

Chương 1. Tạo tài khoản trên Adafruit IO	5
1 Giới thiệu	6
2 Kiến trúc 4 thành phần của ứng dụng	7
3 Tạo tài khoản trên Adafruit IO	8
4 Tạo kênh dữ liệu (Feed)	10
5 Chia sẻ Feed ở dạng Public	12
6 Câu hỏi ôn tập	14
Chương 2. Thiết kế Dashboard trên Adafruit IO	15
1 Giới thiệu	16
2 Tạo Dashboard mới	17
3 Thiết kế giao diện cho Dashboard	19
4 Chính sửa nút nhấn	22
5 Kiểm tra kết nối giữa Feed và Dashboard	23
6 Câu hỏi ôn tập	25
Chương 3. Hiện thực IoT Gateway bằng Python	27
1 Giới thiệu	28
2 Thông tin xác thực	29
3 Tạo dự án trên PyCharm	30
4 Cài đặt thư viện	31
4.1 Cài đặt bằng pip install	32
4.2 Cài đặt bằng GIT	32
5 Hiện thực chương trình	35
5.1 Import thư viện và khởi tạo	35
5.2 Hiện thực hàm chức năng	35
5.3 Cấu hình cho Gateway	36
5.4 Chạy thử chương trình	36
6 Câu hỏi ôn tập	38
Chương 4. Gửi dữ liệu lên Adafruit IO	39
1 Giới thiệu	40
2 Tạo Feed dữ liệu mới	41
3 Thêm đồ thị cho Dashboard	41
4 Kiểm tra tương tác giữa Feed và Dashboard	44
5 Lập trình cho Gateway	45
6 Câu hỏi ôn tập	47

Chương 5. Tích hợp với Microbit	49
1 Giới thiệu	50
2 Chương trình cho Microbit	51
3 Lập trình cho Gateway	51
3.1 Thêm thư viện lập trình	51
3.2 Thêm thư viện cho kết nối Serial	52
4 Tích hợp với Microbit	53
5 Câu hỏi ôn tập	55
Chương 6. Gửi dữ liệu từ Microbit tới Gateway	57
1 Giới thiệu	58
2 Chương trình cho Microbit	59
3 Hàm phân tách dữ liệu	59
4 Hàm đọc dữ liệu Serial	60
5 Tích hợp vào Gateway	60
6 Câu hỏi ôn tập	63
Chương 7. Nhiều nút nhấn trên Dashboard	65
1 Giới thiệu	66
2 Chương trình cho Microbit trung tâm	66
3 Tạo mới Feed dữ liệu	67
4 Tạo mới nút nhấn trên Dashboard	68
5 Cải tiến chương trình của Gateway	69
6 Câu hỏi ôn tập	73
Chương 8. Điều khiển ngoại vi trên nốt cảm biến	75
1 Giới thiệu	76
2 Kết nối mạch điện cho nốt cảm biến	77
3 Lập trình cho nốt cảm biến	78
4 Câu hỏi ôn tập	80
Chương 9. Cảm biến quan trắc tích hợp	81
1 Giới thiệu	82
2 Kết nối với DHT11	83
3 Nguyên lý hoạt động của DHT11	84
4 Lập trình với DHT11	84
5 Cải thiện chương trình ở Gateway	87
6 Các phiên bản nâng cấp của DHT11	87
6.1 Cảm biến DHT22	87
6.2 Cảm biến AM2305	88
7 Câu hỏi ôn tập	89
Chương 10. Cảm biến tương tự Analog	91
1 Giới thiệu	92
2 Nguyên lý thiết kế cảm biến Analog	93
3 Cảm biến khí Gas	94
4 Đọc dữ liệu từ cảm biến khí Gas	95
5 Cảm biến Analog ChiPi	96
6 Câu hỏi ôn tập	98

CHƯƠNG 1

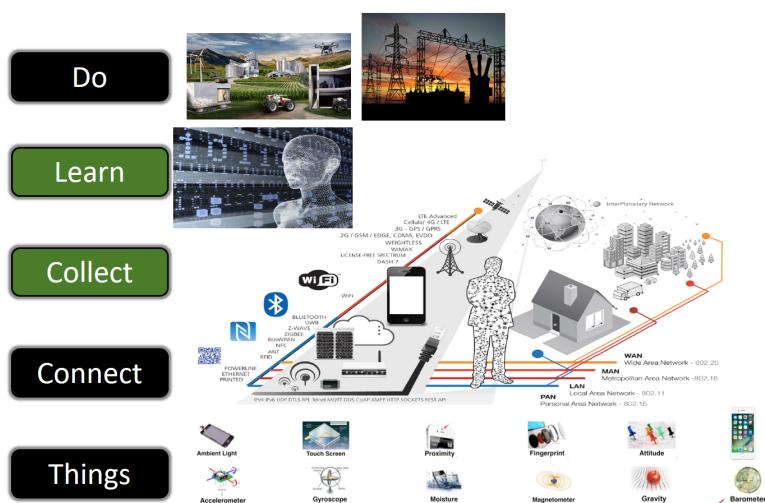
Tạo tài khoản trên Adafruit IO



1 Giới thiệu

Internet vạn vật, hay còn gọi là Internet of Things – IoTs, là một cuộc cách mạng trong việc kết nối giữa các thiết bị không dây với nhau. Ban đầu, chúng ta có mạng Internet, một thành tựu của cuộc cách mạng khoa học công nghệ lần thứ 3, cho phép các máy tính có thể kết nối và trao đổi thông tin toàn cầu. Tuy nhiên, với sự phát triển nhanh chóng của ngành vi cơ điện tử (Micro Electro Mechanical System), không chỉ máy tính, giờ đây rất nhiều các thiết bị có khả năng kết nối vào mạng Internet. Thông dụng nhất trong cuộc sống mà chúng ta có thể kể đến như các điện thoại thông minh, máy tính bảng, các loại thẻ thông minh (Smart cards) hay như các nốt trong mạng cảm biến không dây (Wireless Sensor Networks). Theo ước tính của cộng đồng khoa học, đến năm 2025 sẽ có 75 tỉ thiết bị có thể kết nối mạng Internet với nhau. Với những đặc tính đó, một thế hệ mạng mới đã được hình thành, và là sản phẩm đặc trưng cho cuộc cách mạng khoa học công nghệ lần thứ 4, mạng Internet vạn vật, hay còn gọi là IoT - Internet of Things.

Dựa trên mạng Internet vạn vật, các ứng dụng không còn ở khái niệm thông minh nữa, mà sẽ tiến lên một bước cao hơn, gọi là **tự hành (autonomous)**, chẳng hạn như các ứng dụng giám sát và tự động thích nghi trong việc điều khiển như các dịch vụ trong nhà, bãi giữ xe, hay các hệ thống quan trắc trong nông nghiệp, thủy hải sản. Theo diễn giả nổi tiếng Timothy Chou, kiến trúc về ứng dụng thông minh dựa trên Internet vạn vật của ông được chia thành mô hình 5 lớp, như mô tả ở hình bên dưới.



Hình 1.1: Kiến trúc 5 lớp của một ứng dụng Kết nối vạn vật

Chức năng chính của từng lớp trong kiến trúc này được khái quát như sau:

- **Things:** Các thiết bị trong ứng dụng giám sát. Chúng ta có thể thấy, đây là lớp rất phong phú về mặt số lượng và đa dạng về chức năng. Rất nhiều các loại cảm biến sẽ được dùng, tùy vào các ứng dụng giám sát. Bên cạnh đó, các nốt cảm biến sẽ chủ yếu dựa vào giao tiếp không dây
- **Connect:** Thu thập dữ liệu từ các nốt cảm biến. Do có rất nhiều tiêu chuẩn kết nối tùy theo từng loại ứng dụng, lớp này phải hỗ trợ nhiều loại kết nối, từ

giao tiếp Zigbee và Wifi trong các ứng dụng nhà thông minh, với khoảng cách giao tiếp ngắn cho đến các giao trên không gian rộng như LoRa hay 3G/4G.

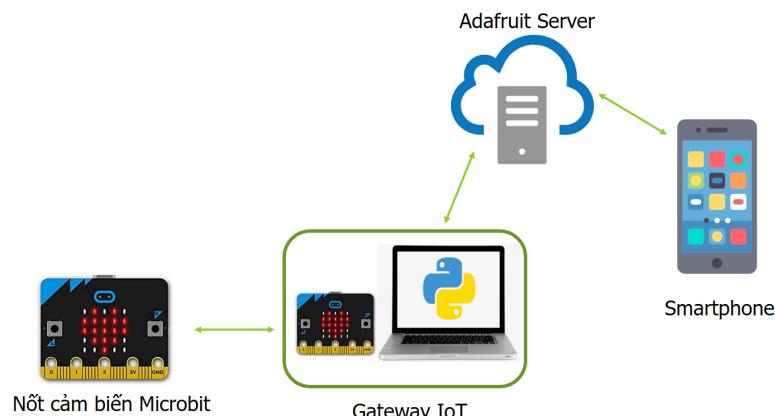
- **Collect:** Sau khi dữ liệu được thu thập, chúng sẽ được gửi lên các server tập trung để lưu trữ dữ liệu. Tại đây, một lượng lớn dữ liệu sẽ được đẩy về, tạo ra một thách thức không nhỏ cho các server và phải ứng dụng các công nghệ về Big Data (dữ liệu lớn) để xử lý.
- **Learn:** Nhiệm vụ của lớp này là lọc ra các thông tin đặc trưng, có ngữ nghĩa đặc thù cho từng loại ứng dụng. Các công nghệ về Học Máy và hiện tại là Học Sâu (Deep Learning) sẽ được áp dụng ở đây
- **Do:** Dựa vào các thông tin đặc trưng, hệ thống sẽ xây dựng nên những quy luật thích nghi theo ngoại cảnh, và đề xuất các quyết định cho hệ thống. Với mỗi quyết định, việc thực thi sẽ được đo đạc một cách tự động, và sai lệnh của quyết định đó so với mục tiêu tối ưu sẽ được xem xét lại cho lần sau. Theo cách này, hệ thống sẽ tự động tích lũy “kinh nghiệm” trong một thời gian dài, để ngày càng trở nên thông minh và hoàn thiện hơn.

Trong giáo trình này, hướng dẫn sẽ tập trung ở lớp **Connect**, với chức năng gọi là **Gateway IoTs**. Thiết bị đóng vai trò làm Gateway, sẽ tập hợp thông tin và gửi lên server, cũng như nhận điều khiển từ phía server ở lớp **Collect**. Chúng tôi sẽ hiện thực Gateway bằng ngôn ngữ lập trình Python. Các nội dung chính trong bài hướng dẫn này sẽ như sau:

- Kiến trúc 4 thành phần của ứng dụng dựa trên IoT
- Tạo tài khoản trên Adafruit IO Server
- Tạo kênh lưu trữ dữ liệu trên Adafruit IO

2 Kiến trúc 4 thành phần của ứng dụng

Dựa trên mô hình 5 lớp của kiến trúc IoTs, chúng ta sẽ phân ứng dụng thành 4 thành phần cơ bản, bao gồm nốt cảm biến, gateway trung tâm, server và các thiết bị để theo dõi dữ liệu và điều khiển từ xa, như trình bày ở hình bên dưới:



Hình 1.2: Kiến trúc 4 thành phần trong ứng dụng IoT

Hạt nhân trung tâm trong kiến trúc này là Gateway IoT, sẽ được xây dựng bằng cách kết hợp giữa máy tính và một mạch Microbit. Sở dĩ có sự kết hợp này, là vì chức năng của Gateway sẽ được hiện thực bằng ngôn ngữ Python. Máy tính của chúng ta là thiết bị mạnh mẽ và tiện dụng cho yêu cầu này. Thêm nữa, khi triển khai hệ thống, chương trình Python có thể dễ dàng sử dụng lên trên các máy tính nhúng, chẳng hạn như Raspberry PI chẳng hạn. Mạch Microbit được gắn thêm vào với nhu cầu mở rộng kết nối với nhiều mạch cảm biến khác bằng kỹ thuật giao tiếp không dây của mạch Microbit.

Nhiều mạch Microbit có thể đóng vai trò là nốt cảm biến, và gửi dữ liệu về mạch Microbit trung tâm, nơi nó sẽ chuyển dữ liệu lên máy tính. Tại đây, chương trình viết bằng ngôn ngữ Python sẽ gửi dữ liệu này lên server Adafruit IO. Từ đó, các thiết bị theo dõi từ xa như điện thoại di động hay thậm chí là một máy tính khác có thể theo dõi được dữ liệu của hệ thống. Trong trường hợp muốn điều khiển một thiết bị, luồng dữ liệu sẽ đi ngược từ phía thiết bị đầu cuối cho đến nốt cảm biến, để thi hành lệnh điều khiển.

Trong bài hướng dẫn đầu tiên này, chúng ta sẽ bắt đầu tạo tài khoản trên server Adafruit IO, trước khi có thể sử dụng nó để lưu trữ dữ liệu cảm biến và luân chuyển tín hiệu điều khiển.

3 Tạo tài khoản trên Adafruit IO

Bước 1: Vào trang web chính tại địa chỉ <https://io.adafruit.com/>. Giao diện sau đây ở hiện ra.



Hình 1.3: Trang chủ của Adafruit IO

Bạn đọc nhấn vào nút **Sign In** để đăng nhập vào hệ thống nếu như đã có tài khoản. Tuy nhiên, nút này cũng sẽ dẫn chúng ta đến trang tạo mới tài khoản ở bước tiếp theo.

Bước 2: Vì chúng ta chưa có tài khoản, nên ở bước này, chúng ta sẽ chọn tiếp **Sign Up** để đăng ký tài khoản.

SIGN IN

Your Adafruit account grants you access to all of Adafruit, including the shop, learning system, and forums.

EMAIL OR USERNAME

PASSWORD

[Forget your password?](#)

[SIGN IN](#)

NEED AN ADAFRUIT ACCOUNT?

[SIGN UP](#)

ORDER STATUS

Did you check out as a guest? Or do you just want to check your order status without signing in?

EMAIL ADDRESS

ORDER NUMBER

[Where do I find this?](#)

[CHECK ORDER STATUS](#)

Hình 1.4: Tạo tài khoản trên Adafruit IO

Bước 3: Cung cấp thông tin cá nhân(FIRST NAME và LAST NAME), Email, Tên đăng nhập và Mật khẩu. Riêng với **tên đăng nhập, bạn không được sử dụng kí tự đặc biệt**. Cuối cùng, chúng ta nhấn vào nút **CREATE ACCOUNT** để tạo tài khoản, như hướng dẫn bên dưới.

SIGN UP

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.

FIRST NAME

LAST NAME

EMAIL

 ✓

USERNAME

 ✓

Username is viewable to the public on the forums, Adafruit IO, and elsewhere.

PASSWORD

 ✓

[CREATE ACCOUNT](#)

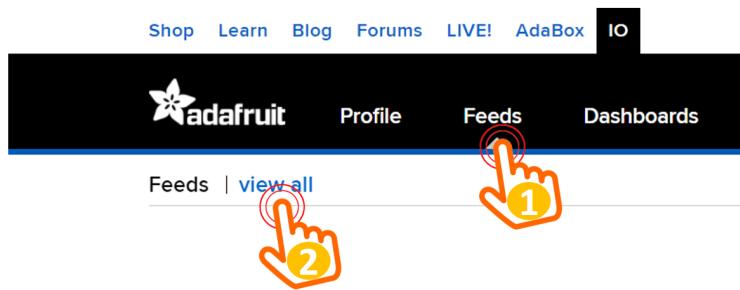
Hình 1.5: Giao diện đăng ký tài khoản

Sau khi tạo tài khoản thành công, hệ thống sẽ tự đăng nhập. Tuy nhiên, ở lần làm việc tiếp theo, bạn đọc hoàn toàn có thể đăng nhập vào hệ thống với chức năng **Sign In**.

4 Tạo kênh dữ liệu (Feed)

Để có thể lưu dữ liệu trên server, chúng ta cần phải phân loại cho nó. Thông thường, chúng ta gọi là một **kênh dữ liệu**, hay feed. Mỗi đối tượng trong hệ thống cũng sẽ thường có 1 kênh dữ liệu cho riêng nó. Chẳng hạn như để lưu trạng thái của một bóng đèn, chúng ta cần một kênh dữ liệu, tên là **BBC_LED** chẳng hạn. Sau khi đăng nhập vào hệ thống thành công, chúng ta bắt đầu tạo một kênh dữ liệu đầu tiên, với các bước hướng dẫn bên dưới.

Bước 1: Mở danh sách kênh dữ liệu, bằng cách nhấn vào Feeds, như minh họa ở hình bên dưới:



Hình 1.6: Truy cập vào các kênh dữ liệu

Tiếp theo đó, chọn tiếp vào tính năng **View all** để liệt kê tất cả các kênh dữ liệu đang có trong tài khoản, như minh họa ở hình sau đây:



Hình 1.7: Danh sách các kênh có sẵn

Trong minh họa ở hình trên, tài khoản của chúng ta chưa có một kênh dữ liệu nào cả, do nó mới được đăng ký lần đầu tiên. Một kênh dữ liệu sẽ tạo ra bằng cách nhấn vào nút **New Feed**.

Bước 2: Điền các thông tin cho kênh dữ liệu, như minh họa ở hình bên dưới.

Create a new Feed

X

Name

BBC_LED

Maximum length: 128 characters. Used: 7

Description

Kênh dữ liệu cho đèn LED

Cancel

Create



Hình 1.8: Диền thông tin cần thiết cho kênh dữ liệu

Quan trọng nhất là trường **Name** của kênh dữ liệu. Bạn đọc nên đặt tên cho nó đi kèm với 1 tiếp đầu ngữ, để có thể dễ dàng phân biệt kênh dữ liệu này là dành cho dự án nào. Trong hướng dẫn này, chúng tôi minh họa cho dữ liệu từ một đèn hiển thị trên Microbit, và đặt tên cho kênh là **BBC_LED**. Phần mô tả **Description** chỉ là tùy chọn, bạn đọc có thể ghi thêm các thông tin chú thích. Cuối cùng nhấn vào nút **Create**. Một kênh mới sẽ được tạo ra và giao diện của chúng ta bây giờ sẽ như sau:

The screenshot shows the BBC Feeds interface. At the top, there are buttons for '+ New Feed' and '+ New Group'. A search bar is on the right. Below, a table titled 'Default' lists one feed: 'BBC_LED' with 'Key' 'bbc-led', 'Last value' 'bbc-led', and 'Recorded' 'less than a minute ago'. There are buttons for '+ New Feed' and 'Group Settings' on the right. A note at the bottom says 'Loaded in 0.85 seconds.'

Hình 1.9: Kênh dữ liệu mới được tạo thành công

Trong trường hợp muốn xóa một kênh dữ liệu cũ, bạn đọc chỉ cần chọn nó và nhấn nút **Delete Feed**, như minh họa ở hình bên dưới:

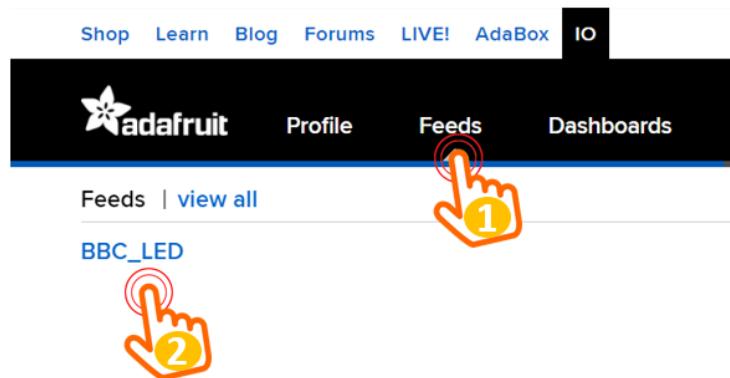
The screenshot shows the BBC Feeds interface. A feed named 'BBC_LED' is selected, indicated by a checked checkbox. A hand cursor icon (labeled 1) is shown clicking the checkbox. Another hand cursor icon (labeled 2) is shown clicking the 'Delete Feed' button. The table shows the feed details: 'Feed Name' BBC_LED, 'Key' bbc-led, 'Last value' bbc-led, and 'Recorded' 29 minutes ago.

Hình 1.10: Xóa một kênh dữ liệu đã có

5 Chia sẻ Feed ở dạng Public

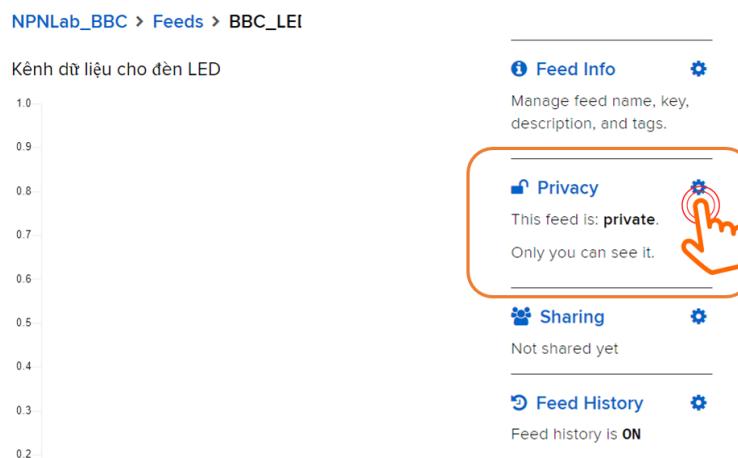
Khi một kênh dữ liệu (feed) được tạo ra, mặc định của nó là ở chế độ **private**, chỉ có tài khoản đăng nhập mới có thể truy cập và gửi dữ liệu lên nó. Để đơn giản hóa việc lập trình từ các thiết bị IoT gửi lên feed, chúng ta sẽ cấu hình cho nó là dạng **Public**.

Để làm được việc này, chúng ta cần phải truy xuất trực tiếp vào feed, bằng cách nhấp chuột trực tiếp vào tên feed, ở đây là **BBC_LED**. Có nhiều cách để bạn đọc có thể tìm thấy kênh của mình. Theo quy trình hiện tại, bạn đọc có thể nhấn trực tiếp vào **BBC_LED** ở [Hình 1.10](#). Trong trường hợp bạn vô lại tài khoản của mình, chỉ cần chọn Feeds, kênh dữ liệu này sẽ xuất hiện, như minh họa ở hình bên dưới:



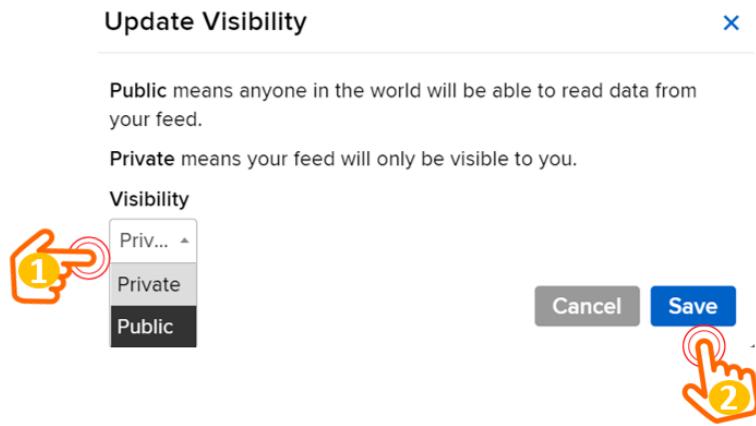
Hình 1.11: Một cách khác để truy cập vào feed

Thông tin chi tiết của feed dữ liệu sẽ được hiện ra như hình bên dưới:



Hình 1.12: Thông tin của feed dữ liệu

Chúng ta hãy để ý ở mục **Privacy** nằm ở khung bên phải, hiện tại nó đang ở chế độ **private**. Chúng ta nhấp vào biểu tượng cài đặt, để tới giao diện sau đây:



Hình 1.13: Cài đặt chia sẻ cho feed

Trong giao diện trên, chúng ta chọn **Public** ở phần **Visibility** và cuối cùng, nhấn vào nút **Save** để hoàn tất việc chỉnh kênh ở chế độ chia sẻ.

Bây giờ, thông tin ở mục **Privacy** đã thay đổi, với thêm thông tin chỉ dẫn **Anyone can see it at this link**. Bạn có thể chia sẻ kênh dữ liệu của mình với người khác bằng cách gửi đi đường liên kết này. Tuy nhiên, bước này chỉ cần thiết trong việc kiểm tra kênh dữ liệu có giao tiếp được tức thì hay không. Ngoài ra, nó cũng không thực sự là tính năng có ích trong các ứng dụng mà chúng ta sắp sửa hiện thực. Thông thường, bạn sẽ có nhu cầu che giấu kênh của mình để bảo vệ dữ liệu của hệ thống. Mục đích của chúng ta khi chỉnh kênh thành Public chỉ để đơn giản việc lập trình trong tương lai.

6 Câu hỏi ôn tập

1. Server được giới thiệu trong bài hướng dẫn có tên là gì?
 - A. ThingSpeak
 - B. Google
 - C. Amazon
 - D. Adafruit IO
2. Một kênh để lưu dữ liệu trên server còn được gọi là gì?
 - A. Client
 - B. Server
 - C. Feed
 - D. Channel
3. Kiến trúc ứng dụng kết nối vạn vật được đề xuất bởi Timothy Chou có mấy lớp?
 - A. 2
 - B. 3
 - C. 4
 - D. 5
4. Lớp các thiết bị như cảm biến, máy bơm, các mạch công suất, thuộc lớp nào trong mô hình IoT?
 - A. Things
 - B. Connect
 - C. Collect
 - D. Learn
5. Gateway IoT thuộc lớp nào trong các lớp dưới đây?
 - A. Things
 - B. Connect
 - C. Collect
 - D. Learn
6. Để tiện lợi cho việc lập trình trong tương lai, các thao tác nào là cần thiết?
 - A. Tạo kênh dữ liệu có tên gợi nhớ
 - B. Nên có tiếp đầu ngữ cho mỗi kênh dữ liệu
 - C. Chính kênh ở chế độ Public
 - D. Tất cả các thao tác trên
7. Để truy cập vào thông tin chi tiết của một feed, các thao tác cần thiết là gì?
 - A. Chọn Feeds, chọn kênh dữ liệu (tên feed)
 - B. Chọn Feeds, chọn view all, chọn kênh dữ liệu (tên feed)
 - C. Cả 2 thao tác trên đều được
 - D. Tất cả đều đúng

Đáp án

1. D 2. C 3. D 4. A 5. C 6. D 7. D

CHƯƠNG 2

Thiết kế Dashboard trên Adafruit IO

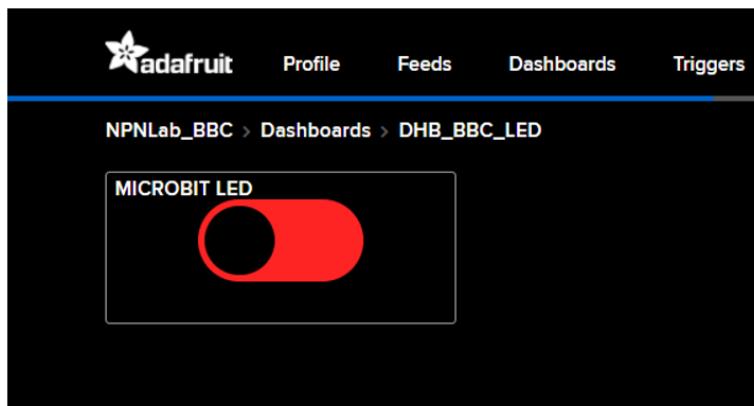


1 Giới thiệu

Dashboard có thể được hiểu là một bảng tổng hợp, hiển thị các thông tin cần thiết của một hệ thống. Trên dashboard, thông tin có thể được tổng hợp để hiển thị dưới dạng đồ thị với các dữ liệu lịch sử, thông tin hiện tại cũng như thống kê các giá trị nhỏ nhất, lớn nhất hay các giá trị trung bình. Ngoài ra, Dashboard còn có thể được sử dụng như một bản điều khiển thân thiện, để người dùng có thể tương tác và vận hành hệ thống từ xa. Tùy vào đặc thù của ứng dụng và nơi áp dụng, các yêu cầu của Dashboard có thể khác nhau. Trong các công ty, Dashboard đưa ra một cái nhìn tổng quát về năng suất của từng bộ phận, các xu hướng, các hoạt động, các chỉ số KPI (Key Performance Indicator – hay còn gọi là chỉ số đánh giá thực hiện công việc).

Với mục đích sử dụng khác nhau hoàn toàn, kênh dữ liệu Feed giới thiệu ở bài trước, thường dành cho người quản lý, để kiểm tra dữ liệu thô của hệ thống. Trong khi đó, Dashboard sẽ là một giao diện đẹp và thân thiện đối với người sử dụng. Hai đối tượng này, cũng thường được gọi là **Back End** dành cho Feed và **Front End** dành cho Dashboard. Và hiển nhiên, 2 đối tượng này sẽ liên kết chặt chẽ với nhau: Mỗi khi có dữ liệu gửi lên Feed, giao diện trên Dashboard sẽ được cập nhật tương ứng, và ngược lại, mỗi khi có tương tác trên Dashboard, thông tin cũng sẽ được lưu lại trên Feed.

Ở bài này, bạn đọc sẽ được hướng dẫn để tạo một Dashboard đơn giản trên Adafruit IO. Dashboard này là dùng để kết nối với Feed ta đã tạo ở bài trước và có một nút nhấn, để người dùng có thể bật tắt đèn trên mạch Microbit. Giao diện tương tác của người dùng trên Dashboard như sau:



Hình 2.1: Giao diện Dashboard điều khiển đèn

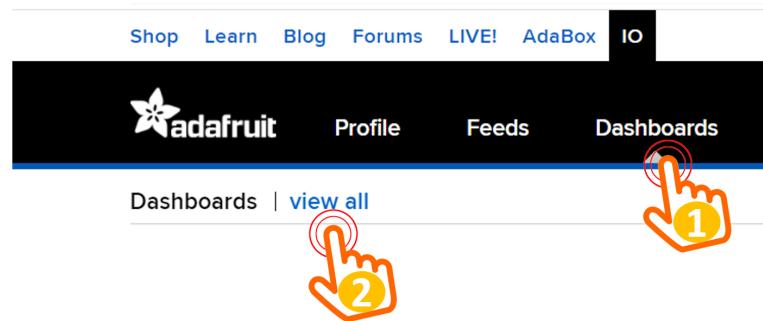
Các mục tiêu hướng dẫn trong bài này được tóm tắt như sau:

- Tạo một Dashboard với một nút nhấn
- Liên kết Dashboard và Feed dữ liệu
- Kiểm tra tương tác giữa Dashboard và Feed

2 Tạo Dashboard mới

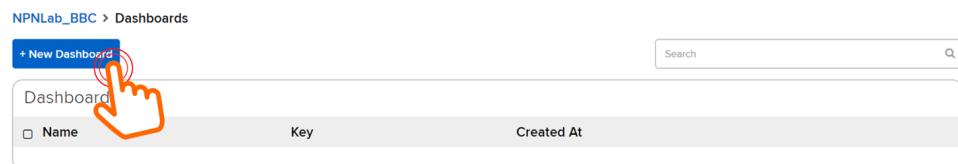
Trong phần này, hướng dẫn sẽ trình bày để bạn đọc có thể tạo ra 1 giao diện đơn giản, với một nút nhấn trên Dashboard, dùng để bật/tắt một thiết bị, chẳng hạn như là đèn hiển thị trên mạch Microbit. Giao diện và thao tác để tạo mới một Dashboard cũng khá tương tự với việc tạo mới một Feed ở bài trước, được trình bày chi tiết từng bước như dưới đây.

Bước 1: Sau khi đăng nhập vào Adafruit IO, bạn đọc chọn vào **Dashboard**, và chọn tiếp **View all**, như hướng dẫn bên dưới:



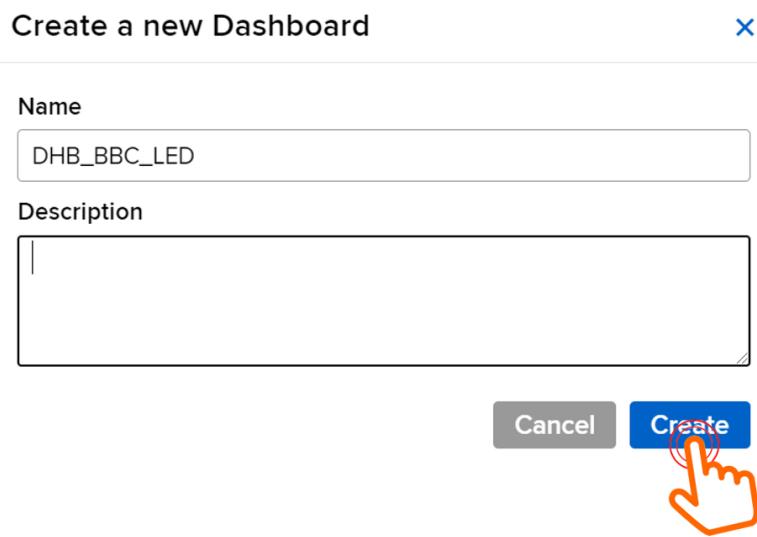
Hình 2.2: Truy cập vào Dashboard của tài khoản

Bước 2: Nhấn nút **+New Dashboard** để tạo mới một Dashboard trên Adafruit, như minh họa ở hình sau đây:



Hình 2.3: Tạo mới một Dashboard

Sau đó, một cửa sổ sẽ hiện ra cho ta ghi tên và mô tả cho Dashboard của mình. Lưu ý là trường **Name** thông tin bắt buộc, trong khi đó, trường **Description** (mô tả) là tùy chọn. Khi đặt tên cho Dashboard, bạn cũng nên thêm những tiếp đầu ngữ để dễ quản lý trong tương lai, như minh họa ở hình bên dưới:



Hình 2.4: Hoàn thiện thông tin cho Dashboard

Cuối cùng, nhấn vào nút **Create**, một Dashboard mới sẽ xuất hiện trong trang chủ Adafruit IO của bạn, như sau:

Name	Key	Created At
DHB_BBC_LED	dhb-bbc-led	August 9, 2021

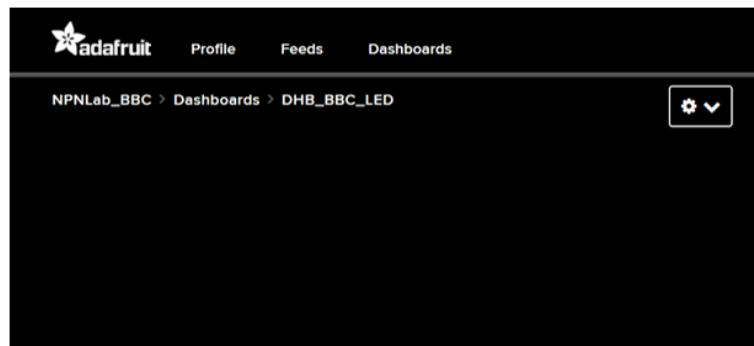
Hình 2.5: Một Dashboard mới đã được tạo

Trong trường hợp muốn xóa Dashboard, bạn đọc cần chọn nó trong danh sách ở trên, và nhấn vào nút **Delete Dashboard**, như minh họa ở hình sau:

Name	Key	Created At
DHB_BBC_LED	dhb-bbc-led	August 9, 2021

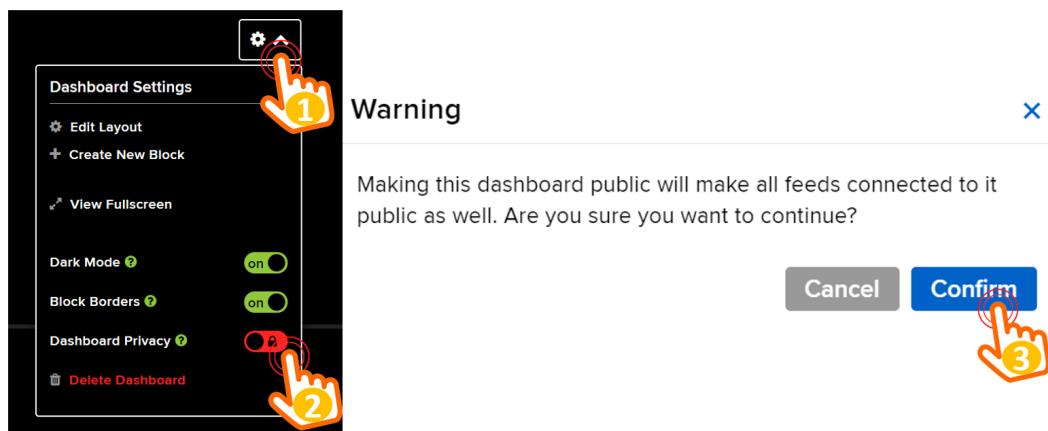
Hình 2.6: Xóa một Dashboard trong danh sách

Sau khi tạo thành công Dashboard ở các bước hướng dẫn bên trên, bạn đọc có thể truy xuất vào nó (bằng cách nhấp chuột trực tiếp vào tên Dashboard), một giao diện khởi tạo của Dashboard sẽ hiện ra, như sau:



Hình 2.7: Giao diện khởi tạo cho Dashboard

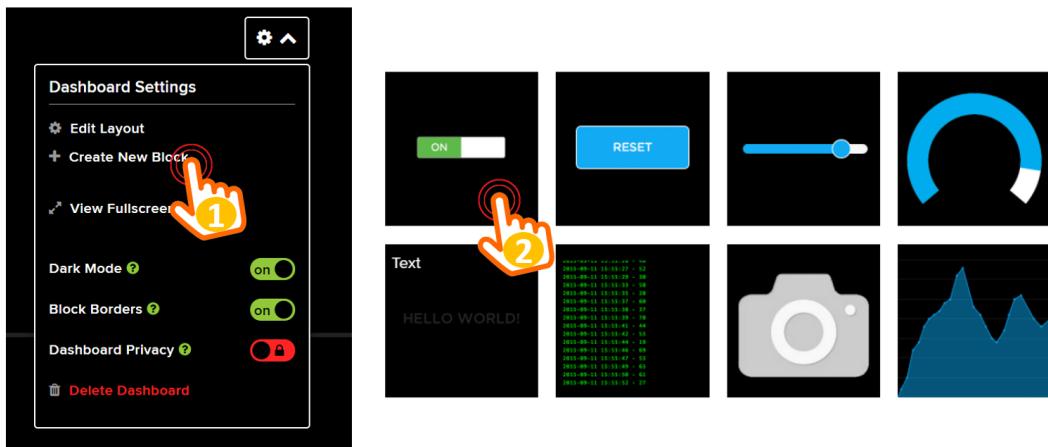
Trước khi thiết kế các đối tượng giao diện trên Dashboard (trong bài này là một nút nhấn), chúng ta cần cấu hình cho Dashboard ở dạng **Public**, để tiện chia sẻ trên nhiều thiết bị khác nhau, chẳng hạn như điện thoại hoặc máy tính bảng, để điều khiển và giám sát hệ thống từ xa. Bằng cách chọn vào biểu tượng cài đặt ở phía bên phải, chúng ta chọn tiếp vào **Dashboard Privacy**, như hướng dẫn sau đây:



Hình 2.8: Tùy chỉnh cho Dashboard ở chế độ Public

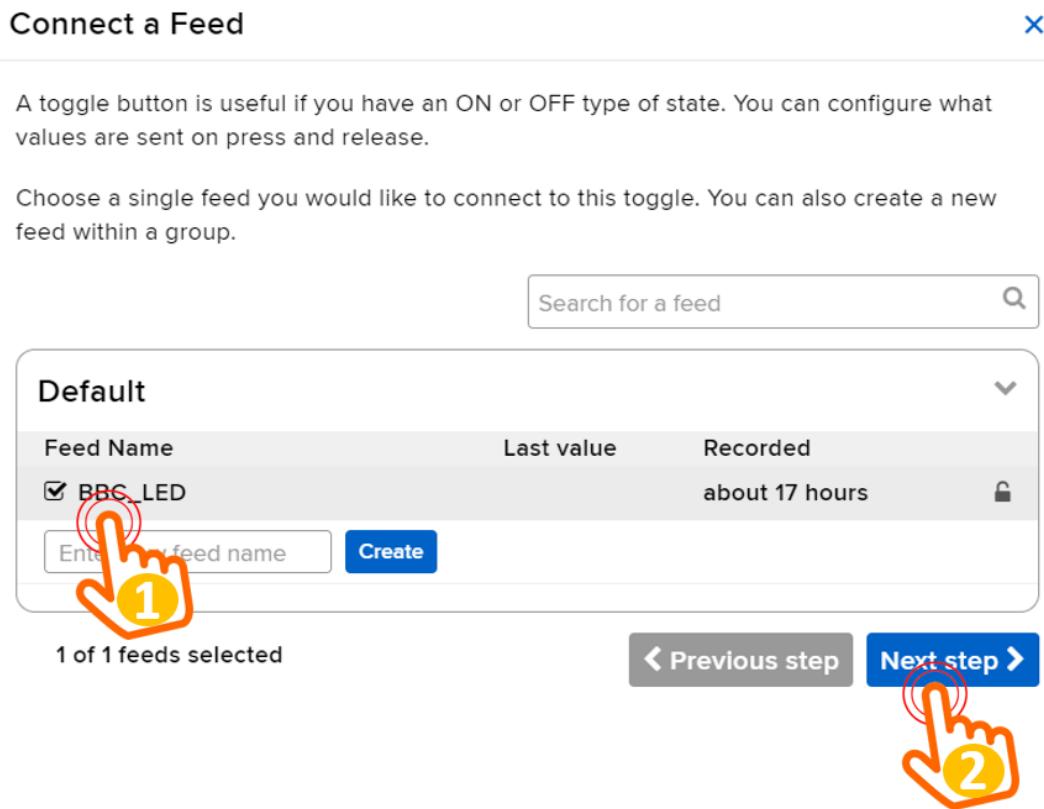
3 Thiết kế giao diện cho Dashboard

Giao diện mặc định của Dashboard khi mới được tạo ra là chưa có phần tử giao diện nào, chúng ta sẽ bắt đầu thêm một nút nhấn trên giao diện. Bằng cách vào lại biểu tượng cài đặt, và chọn **Create New Block**, như sau:



Hình 2.9: Thêm đối tượng giao diện vào Dashboard

Đối tượng giao diện hỗ trợ trên Dashboard là rất phong phú. Tuy nhiên trong bài hướng dẫn này, chúng ta sẽ chọn vào đối tượng đầu tiên. Đối tượng này có tên gọi là **Toggle Button**, và nó hoàn toàn phù hợp cho một ứng dụng điều khiển Bật/Tắt một thiết bị trong bài này. Sau khi chọn, giao diện sau đây sẽ hiện ra.



Hình 2.10: Kết nối với feed đã tạo

Đây là bước quan trọng nhất trong thiết kế giao diện cho Dashboard, khi chúng ta cần phải chọn liên kết cho nó với một Feed dữ liệu. Cho đến bài này, vì chúng ta chỉ có 1 feed dữ liệu (là BBC_LED), nên việc lựa chọn rất đơn giản. Trong trường hợp có nhiều feed dữ liệu, bạn đọc cần phải lựa chọn cho đúng. Sau đó chọn vào **Next step** và tiếp tục tùy chỉnh các thông tin cài đặt cho giao diện bên dưới.

Block settings

X

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title (optional)

MICROBIT LED

Button On Text

ON

Button On Value (uses On Text if blank)

1



Button Off Text

OFF

Button Off Value (uses Off Text if blank)

0



Block Preview

MICROBIT LED



Toggle A toggle button is useful if you have an ON or OFF type of state. You can configure what values are sent on press and release.

Test Value

1

Published

0 bytes

← Previous step

Create block



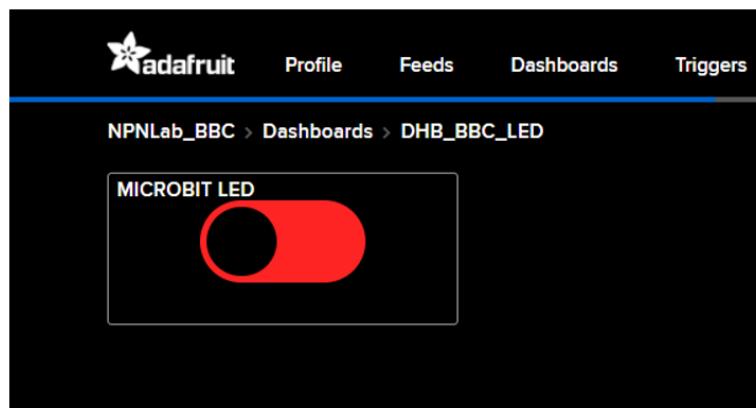
Hình 2.11: Các chức năng cài đặt cho đối tượng nút nhấn

Chức năng của các trường thông tin được tóm tắt như sau:

- **Block Title:** Trường này là tùy chọn thêm, không bắt buộc. Ở đây ta có thể đặt tên cho nút nhấn.
- **Button On Text:** Ở đây ta sẽ để những ký tự sẽ hiện lên trên nút nếu nút mở. Ở trạng thái mặc định không tùy chỉnh thì khi được mở, nút sẽ hiện từ "ON".
- **Button On Value:** Trường này cho ta tùy chỉnh dữ liệu được gửi đi khi nút này được bấm để trở thành trạng thái Bật. Để cho đơn giản, **chúng ta sẽ quy định dữ liệu cho nó là 1.**
- **Button Off Text:** Ở đây ta sẽ để những ký tự sẽ hiện lên trên nút nếu nút tắt. Ở trạng thái mặc định không tùy chỉnh thì khi được tắt, nút sẽ hiện từ "OFF".

- **Button Off Value:** Trường này cho ta tùy chỉnh dữ liệu được gửi đi khi nút này được bấm để trở thành trạng thái tắt. **Chúng ta sẽ quy định dữ liệu cho trường này là 0.**

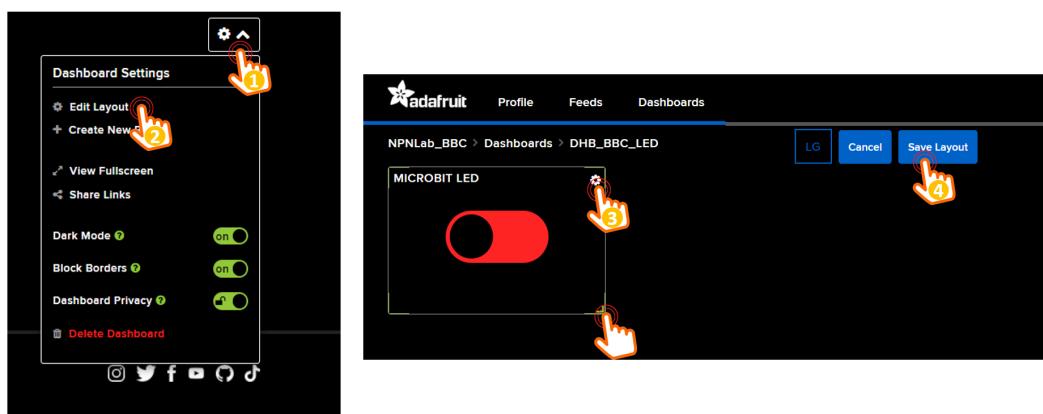
Ngoài ra, bạn đọc có thể kiểm tra việc thay đổi giao diện của nút nhấp, mỗi khi có dữ liệu mới được gửi lên Feed, bằng cách nhập giá trị 0 và 1 trong ô **Test Value**. Chức năng này đang mô phỏng hình ảnh của nút nhấp, khi vận hành trong thực tế. Cuối cùng, ta chọn **Create block** và một nút nhấp mới sẽ xuất hiện trong Dashboard của bạn, như sau.



Hình 2.12: Nút nhấp mới được tạo trên Dashboard

4 Chính sửa nút nhấp

Trong một số trường hợp, chúng ta sẽ có nhu cầu cài đặt lại cho đối tượng nút nhấp, chẳng hạn như làm cho kích thước của nó to hơn, thay đổi các giá trị cho trạng thái Bật/Tắt, hoặc thậm chí là xóa nút nhấp này. Tính năng hiệu chỉnh có thể được kích hoạt, bằng cách nhấp vào biểu tượng cài đặt, và chọn tiếp vào **Edit Layout**, như hướng dẫn bên dưới:



Hình 2.13: Chính sửa đối tượng nút nhấp

Các đối tượng giao diện trên Dashboard lúc này sẽ xuất hiện thêm một biểu tượng cài đặt riêng của nó. Khi nhấp vào biểu tượng này, và chọn tiếp **Edit Block**, giao diện cũ trình bày ở [Hình 2.11](#) sẽ xuất hiện để bạn đọc thay đổi các thông tin cấu

hình. Tính năng xóa đối tượng giao diện cũng sẽ xuất hiện trong danh sách lựa chọn, khi chúng ta nhấn vào biểu tượng cài đặt này.

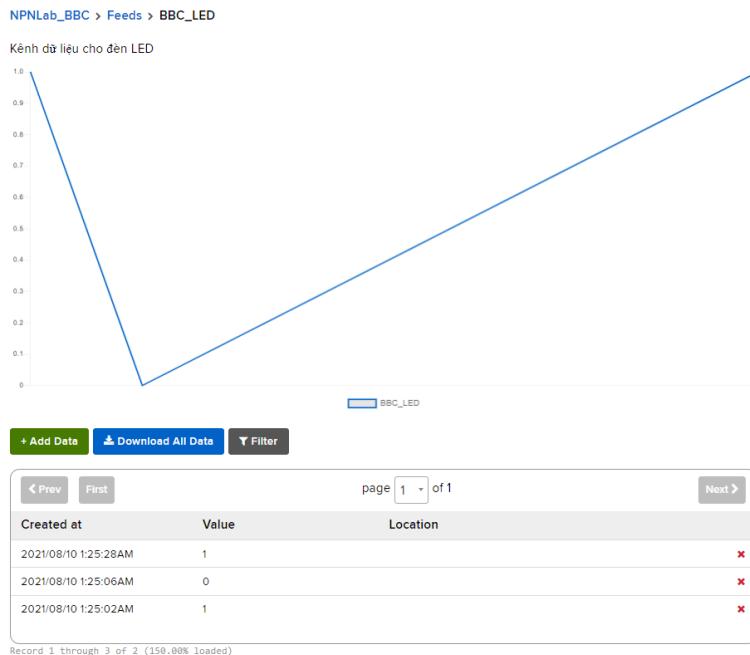
Trong trường hợp muốn thay đổi kích thước của đối tượng giao diện, bạn đọc chỉ đơn giản là kéo thả 4 góc của nó để thay đổi. Cuối cùng, nhấn vào nút **Save Layout**.

5 Kiểm tra kết nối giữa Feed và Dashboard

Đây là bước kiểm tra cuối cùng các cấu hình trên Adafruit IO, trước khi chúng ta bắt đầu lập trình gửi dữ liệu từ Gateway lên Feed. Quy trình kiểm tra sẽ theo trình tự như sau:

Bước 1: Từ giao diện Dashboard, bạn hãy nhấn thử nút nhấn trên giao diện, ghi nhớ lại số lần nhấn, tương ứng với các trạng thái ON và OFF của nút nhấn.

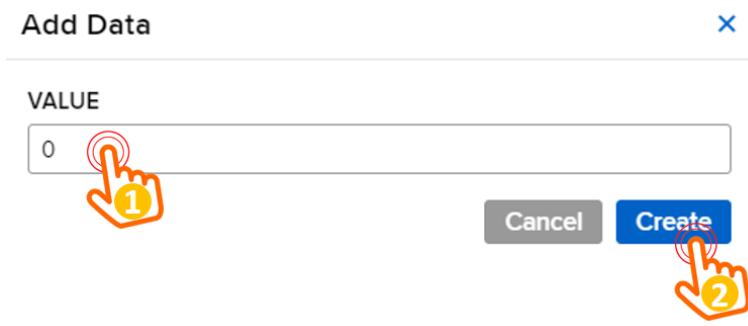
Bước 2: Mở lại giao diện của kênh dữ liệu Feed, bạn sẽ thấy giá trị thô, là 0 và 1 sẽ được gửi lên Feed, như minh họa ở hình sau.



Hình 2.14: Dữ liệu được gửi lên Feed khi tương tác trên Dashboard

Mặc dù một đồ thị được vẽ ra trên kênh dữ liệu, bạn đọc cần lưu ý dữ liệu thô của nó, được liệt kê trong một bảng bên dưới. Mỗi khi chúng ta nhấn nút, trạng thái của nó sẽ được lưu lại, cùng với thông số về thời gian. Thông tin thô này, sẽ được gửi xuống Gateway trong tương lai.

Bước 3: Tại trang dữ liệu Feed, hãy chọn và nút **+ Add Data**. Tính năng này đang mô phỏng việc gửi dữ liệu từ Gateway lên Feed trong tương lai. Giao diện sau đây sẽ hiện ra:



Hình 2.15: Thêm dữ liệu trên Feed bằng tay

Bạn đọc hãy điền những giá trị hợp lệ, trong trường hợp này là 0 hoặc 1, sau đó nhấn vào nút **Create**. Hiển nhiên, 1 dòng dữ liệu mới sẽ được thêm vào Feed. Nhưng song song đó, giao diện của Dashboard cũng được thay đổi tương ứng.

Bước 4: Kiểm tra độ trễ của hệ thống. Đây sẽ điều mới mẻ mà Adafruit IO server mang lại. Bạn đọc hãy thử chia sẻ đường link Dashboard của mình với người khác, để 2 bên có thể kiểm tra từ xa. Chúng ta sẽ thấy rằng, việc giao tiếp dữ liệu giữa Feed và Dashboard có độ trễ rất thấp.

Với Gateway IoT sẽ hiện thực ở bài sau, mỗi khi tương tác trên Dashboard, dữ liệu sẽ được gửi lên Feed. Bước tiếp theo đó, Feed sẽ tự động gửi xuống cho Gateway IoT để thực thi lệnh này. Trong trường hợp ngược lại, khi dữ liệu được gửi lên Feed từ Gateway IoT, giao diện của Dashboad cũng sẽ tự động cập nhật theo. Nhờ cơ chế này, mà server Adafruit IO mới phù hợp cho các ứng dụng quan trắc và điều khiển từ xa qua mạng.

6 Câu hỏi ôn tập

1. Đối tượng nào sau đây có thể được sử dụng để giám sát và điều khiển hệ thống?
 - A. Gateway IoT
 - B. Feed
 - C. Dashboard
 - D. Adafruit IO
2. Đối tượng nào có vai trò giống như một Back-End?
 - A. Gateway IoT
 - B. Feed
 - C. Dashboard
 - D. Adafruit IO
3. Đối tượng nào có vai trò giống như một Front-End?
 - A. Gateway IoT
 - B. Feed
 - C. Dashboard
 - D. Adafruit IO
4. Khi cấu hình một nút nhấn trên Dashboard, giá trị cho 2 trạng thái của nút nhấn như thế nào là phù hợp?
 - A. 0 và 1
 - B. 1 và 2
 - C. ON và OFF
 - D. Khác nhau là được
5. Độ trễ của việc giao tiếp dữ liệu giữa Feed và Dashboard là:
 - A. Nhanh
 - B. Chậm
 - C. Trung bình
 - D. Không xác định được
6. Độ trễ của việc giao tiếp dữ liệu giữa Feed và Gateway IoT là:
 - A. Nhanh
 - B. Chậm
 - C. Trung bình
 - D. Không xác định được
7. Độ trễ của việc giao tiếp dữ liệu giữa Dashboard và Gateway IoT là:
 - A. Nhanh
 - B. Chậm
 - C. Trung bình
 - D. Không xác định được

Đáp án

1. C
2. B
3. C
4. D
5. A
6. A
7. A

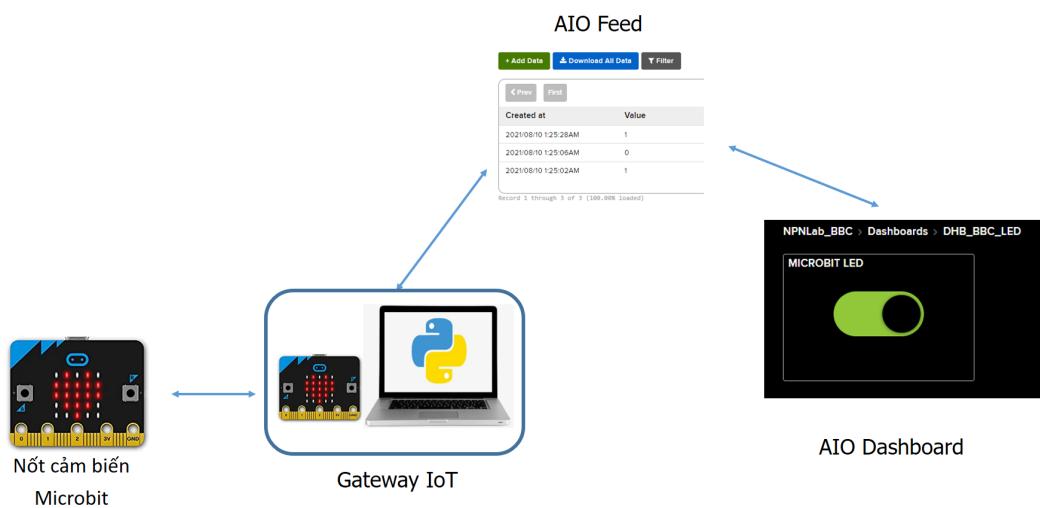
CHƯƠNG 3

Hiện thực IoT Gateway bằng Python



1 Giới thiệu

Với các bước chuẩn bị ở bài trước cho kênh dữ liệu (Feed) và một giao diện tương tác (Dashboard), trong bài này chúng ta sẽ bắt đầu hiện thực chức năng của một Gateway IoT. Mặc dù chúng ta đang dùng máy tính để hiện thực chức năng của một Gateway IoT, chương trình giới thiệu trong bài này hoàn toàn có thể được hiện thực trên các máy tính nhúng khi triển khai thành các ứng dụng thực tế như Raspberry PI hoặc Jetson Nano. Đây là lợi thế rất lớn của ngôn ngữ lập trình Python, khi nó có thể tương thích với nhiều nền tảng hệ điều hành khác nhau. Thậm chí, hiệu năng của chương trình Python trên máy tính nhúng còn tốt hơn trên máy tính mà chúng ta đang làm việc. Với hệ điều hành Linux trên các máy tính nhúng, mã nguồn Python sẽ được thực thi tối ưu hơn.



Hình 3.1: Kiến trúc 4 thành phần trong ứng dụng IoT

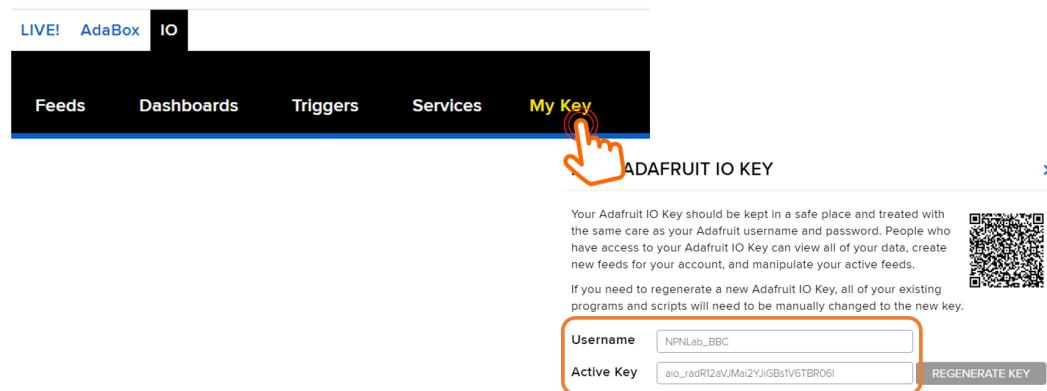
Kiến trúc 4 thành phần của ứng dụng dựa trên kết nối vạn vật, đến bài này đã cụ thể hóa hơn như trình bày ở hình trên. Feed dữ liệu trên Adafruit server đã trở thành nơi trung chuyển dữ liệu giữa Gateway IoT và Dashboard. Trong bài hướng dẫn này, chúng ta sẽ tập trung hiện thực trên máy tính một chương trình, sẽ nhận dữ liệu mỗi khi người dùng tương tác trên Dashboard. Đây là tính năng mới mà bạn đọc sẽ cần cho các ứng dụng điều khiển từ xa. Tính năng này, với các server thông thường, chẳng hạn như ThingSpeak, sẽ có độ trễ đáng kể. Nói một cách khác, khi người dùng muốn bật một bóng đèn với server ThingSpeak, tối thiểu 30 giây sau Gateway IoT mới có thể nhận ra lệnh này và thực hiện. Tuy nhiên, với server Adafruit IO, độ trễ này là rất thấp, và thông thường là dưới 1 giây.

Các mục tiêu hướng dẫn trong bài này như sau:

- Cài đặt thư viện lập trình cho Adafruit IO server
- Hiện thực chương trình Python trên máy tính
- Kiểm tra chương trình với Dashboard

2 Thông tin xác thực

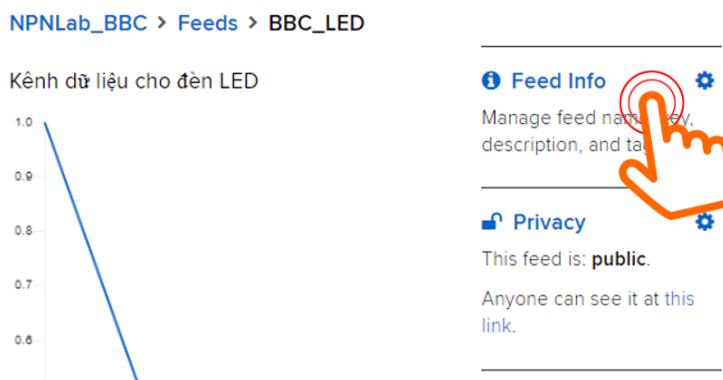
Để thông tin được gửi lên Feed một cách hợp lệ, Gateway cần phải có thông tin xác thực để kết nối với Feed. Từ thanh công cụ của trang chủ Adafruit IO, sau khi đăng nhập vào hệ thống, bạn sẽ có thông tin xác thực trong mục **My Key**, như minh họa sau đây:



Hình 3.2: Thông tin để xác thực tài khoản

Với giao diện hiện ra khi nhấn vào **My Key**, chúng ta cần lưu lại thông tin về **Username** và **Active Key**.

Tiếp theo, bạn cần kiểm tra tên của Feed dữ liệu khi truy xuất nó dưới dạng **Active Key**. Tên này sẽ được sinh ra tự động bởi hệ thống, và khác với tên chúng ta đặt khi tạo mới một Feed ở bài trước. Để truy xuất thông tin này, bạn chọn và **Feeds** trên thanh công cụ và nhấp chuột vào Feed dữ liệu đã tạo, trong hướng dẫn này là **BBC_LED**, giao diện sau đây sẽ hiện ra:



Hình 3.3: Kiểm tra thông tin của Feed

Sau khi nhấn vào mục **Feed Info**, một giao diện nữa sẽ hiện ra. Tại đây, chúng ta sẽ có tên của kênh dữ liệu để sử dụng cho phần lập trình, như minh họa ở hình bên dưới:

Create a new Feed X

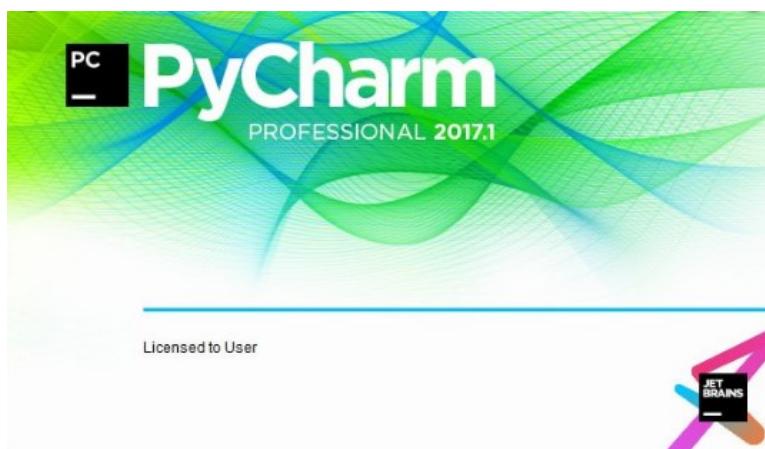
Name	<input type="text" value="BBC_LED"/>
Maximum length: 128 characters. Used: 7	
Key	<input type="text" value="bbc-led"/>
Changing the key will change API URLs and MQTT subscription topics. The only characters we permit are lower case english letters	

Hình 3.4: Tên truy cập của Feed dữ liệu

Như vậy, các thông tin cần lưu lại trước khi lập trình, bao gồm tên Feed ở dạng truy cập bằng khóa (thường là tên của Feed, nhưng ở dạng viết thường), Username và Active Key. Ba thông tin này sẽ được sử dụng khi hiện thực chương trình Python.

3 Tạo dự án trên PyCharm

Trong hướng dẫn này, chúng tôi sẽ sử dụng môi trường lập trình PyCharm để hiện thực chương trình chức năng cho Gateway IoT chạy trên máy tính. PyCharm được đánh giá là phần mềm thông dụng bậc nhất cho việc lập trình Python, bởi nó có thể tương thích với nhiều hệ điều hành khác nhau như Windows, Linux hay Mac OS. Phần mềm này cung cấp một môi trường ảo để chúng ta có thể dễ dàng cài đặt các thư viện mở rộng khi lập trình đến những tính năng cao cấp. Phiên bản trình thông dịch Python sử dụng trong hướng dẫn này là 3.8.

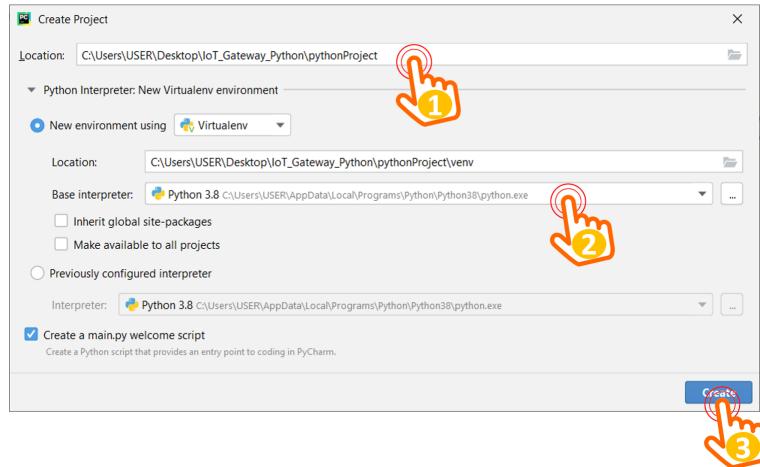


Hình 3.5: Phần mềm lập trình PyCharm

Hướng dẫn cài đặt phần mềm PyCharm cũng như chương trình biên dịch Python có thể được tìm thấy trong giáo trình Python Cơ Bản, được chia sẻ ở đường dẫn sau:

https://drive.google.com/file/d/1dJLE3CdRJvU2QUOfjqMcX6azHMBYBl_q/view

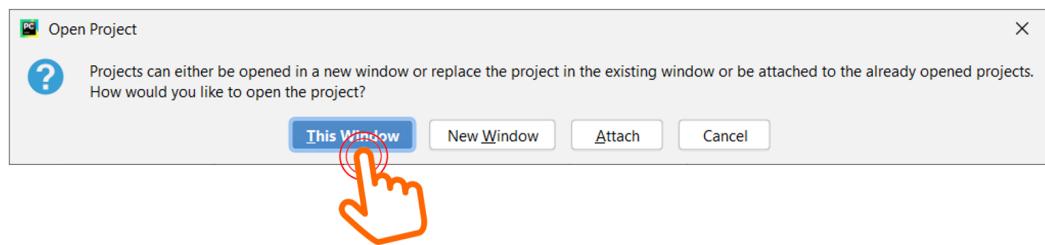
Sau khi khởi động PyCharm, chúng ta chọn **File/New Project**, như minh họa ở hình bên dưới:



Hình 3.6: Tạo mới một dự án trên Pycharm

Bạn đọc cần lưu ý đường dẫn để lưu dự án. Bạn đọc nên tạo trước một thư mục mới, để lưu tất cả dự án vào thư mục này. Chúng ta cũng không quên kiểm tra phiên bản của trình thông dịch Python trước khi nhấn vào nút **Create**. Trong trường hợp bạn cài nhiều phiên bản Python vào máy tính, sẽ có một danh sách chọn lựa cho bạn ở mục **Base interpreter**.

Một cửa sổ thông báo có thể sẽ hiện ra sau khi bạn nhấn vào **Create**. Bạn đọc hãy lựa chọn là tạo mới dự án tại cửa sổ hiện tại của PyCharm để không phải mở nhiều cửa sổ cùng lúc, bằng cách nhấn vào **This Window**, như minh họa sau:



Hình 3.7: Tạo dự án tại cửa sổ hiện tại của PyCharm

Khi dự án được tạo xong, bạn đọc có thể xóa hết nội dung chương trình mặc định trong **main.py**, để bắt đầu hiện thực chương trình cho IoT Gateway.

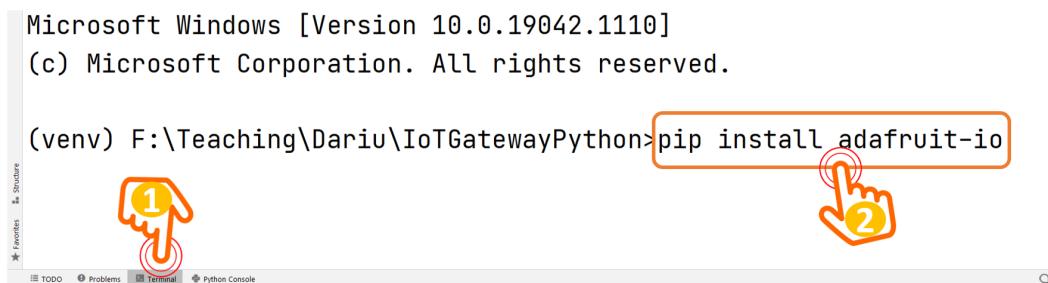
4 Cài đặt thư viện

Một trong những bước đầu tiên không thể thiếu cho các tính năng cao cấp với ngôn ngữ Python là cài đặt thư viện. Thư viện mở rộng mà chúng ta cần tích hợp là **adafruit-io**. Có 2 cách để chúng ta cài đặt thư viện này: Cài đặt trực tuyến bằng công cụ **pip install** và cài đặt từ mã nguồn GIT do chúng tôi tự phát triển. Đối với bạn đọc đã có kinh nghiệm thì phương pháp đầu tiên sẽ quen thuộc hơn. Tuy

nhiên, đối với những ai mới bắt đầu hoặc phương pháp 1 không thành công, bạn đọc có thể thử ở phương pháp thứ 2. Với server Git được chúng tôi bảo trì thường xuyên và cung cấp thư viện phù hợp nhất cho bạn đọc. Chi tiết cho các 2 phương pháp cài đặt thư viện được trình bày như sau:

4.1 Cài đặt bằng pip install

Đối với phương pháp này, chúng ta sẽ cần mở cửa sổ **Terminal** trên PyCharm, và gõ **pip install adafruit-io**, sau đó nhấn Enter, như minh họa bên dưới:

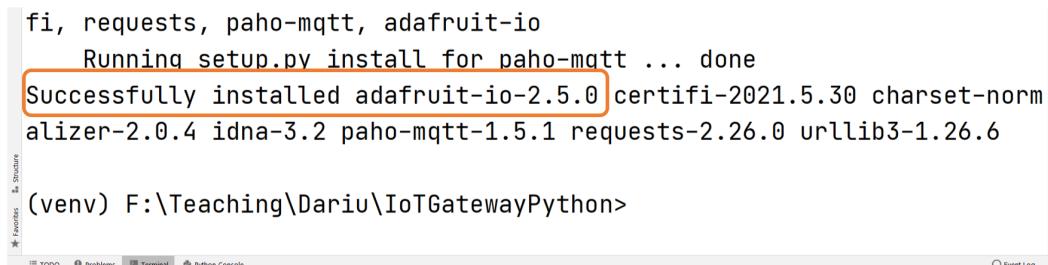


Microsoft Windows [Version 10.0.19042.1110]
(c) Microsoft Corporation. All rights reserved.

(venv) F:\Teaching\Darui\IoTGatewayPython>pip install adafruit-io
1 2

Hình 3.8: Cài đặt thư viện Adafruit IO

PyCharm sẽ tải thư viện về máy tính của chúng ta và cài đặt. Sau khi cài đặt thành công, sẽ có thông báo sau đây xuất hiện:



```
fi, requests, paho-mqtt, adafruit-io  
  Running setup.py install for paho-mqtt ... done  
Successfully installed adafruit-io-2.5.0 certifi-2021.5.30 charset-normalizer-2.0.4 idna-3.2 paho-mqtt-1.5.1 requests-2.26.0 urllib3-1.26.6  
  
(venv) F:\Teaching\Darui\IoTGatewayPython>
```

Hình 3.9: Cài đặt thư viện thành công

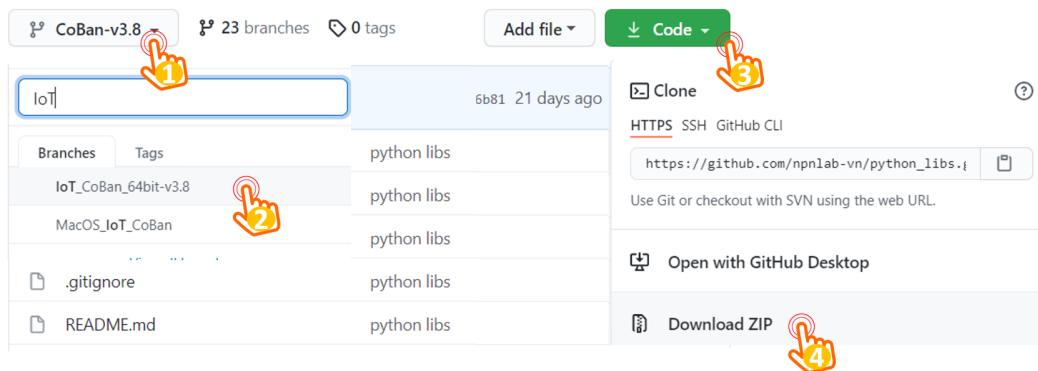
Cho đến bước này, chúng ta đã sẵn sàng để hiện thực chương trình. Nếu như việc cài đặt thư viện không thành công, bạn đọc có thể thử với cách thứ 2, ở phần tiếp theo.

4.2 Cài đặt bằng GIT

Đối với cách thứ 2 này, bạn cần tải một file ZIP từ server GIT do chúng tôi xây dựng. Đầu tiên, hãy truy cập vào trang chủ sau đây:

https://github.com/npnlab-vn/python_libs

Sau điện như hình bên dưới sẽ hiện lên, với rất nhiều thư viện do chúng tôi xây dựng sẵn theo từng giáo trình.



Hình 3.10: GIT lưu các thư viện lập trình Python

Bước 1: Bạn cần chọn nhánh cho đúng trước khi tải thư viện về máy tính, bằng cách nhấn vào lựa chọn ở vị trí 1. Sau đó, tìm đến nhánh **IoT_CoBan_64bit-v3.8**. Đối với máy có cấu hình khác, hoặc phiên bản Python khác, bạn đọc tự lựa chọn thư viện phù hợp với máy tính của mình.

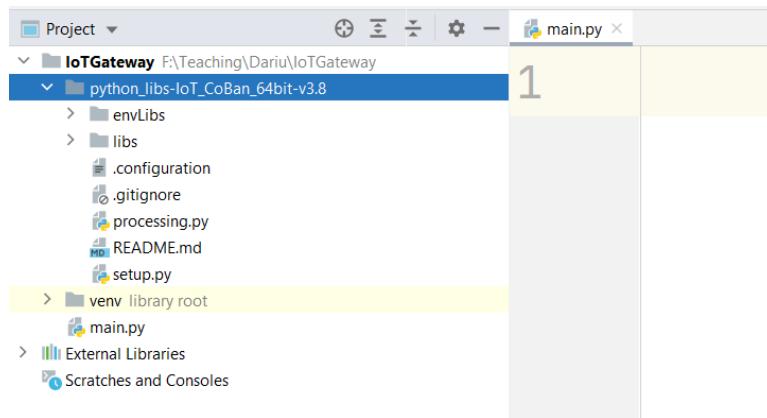
Bước 2: Sau khi đã chọn đúng nhánh, chúng ta đã có thể tải thư viện về, bằng cách nhấn vào nút **Code** và chọn chế độ tải file ZIP ở mục **Download ZIP**. Bạn đọc cần chọn đường dẫn để lưu lại file vừa mới tải về.

Bước 3: Giải nén file ZIP vừa tải về, bằng cách nhấn chuột phải và chọn **Extract Here**. Với thư mục vừa được giải nén, tiếp tục nhấn chuột phải để sao chép (hoặc nhấn phím nóng Ctrl+C), như minh họa bên dưới:



Hình 3.11: Giải nén thư viện bằng lệnh Extract Here

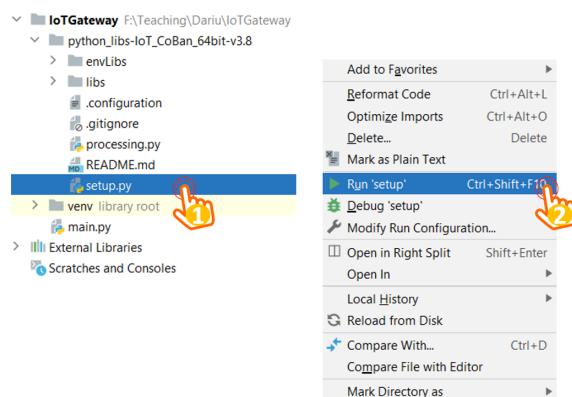
Bước 4: Sao chép thư mục đã giải nén, và dán nó vào thư mục lưu dự án đã tạo ở phần trước. Trong thư mục này, bạn sẽ thấy file **main.py**, như kết quả sau đây:



Hình 3.12: Thư viện đã được thêm vào dự án

Sau bước này, khi quay trở lại môi trường lập trình PyCharm, chúng ta sẽ thấy các thư viện cùng những tập tin đính kèm xuất hiện trong cửa sổ dự án.

Bước 5: Click chuột phải vào file **setup.py** và chọn **Run**, để tiến hành cài đặt thư viện, như minh họa ở hình bên dưới:



Hình 3.13: Cài đặt thư viện bằng cách chạy file setup.py

Sau khi việc cài đặt hoàn tất, thông báo sau đây sẽ xuất hiện:

```
Installing collected packages: urllib3, idna, charset1
Successfully installed adafruit-io-2.5.0 certifi-2021.10.6
=====
Successfully installed packages!!!
```

Process finished with exit code 0

Hình 3.14: Cài đặt thư viện thành công

Trong trường hợp việc cài đặt không thành công, bạn đọc cần kiểm tra lại phiên bản Python của mình. Khi tạo dự án, thông tin này sẽ nằm ở mục **Base Interpreter** để tải lại thư viện cho phù hợp.

5 Hiện thực chương trình

Bây giờ, chúng ta chuyển sang làm việc ở file **main.py** để hiện thực chức năng của chương trình. Từng bước hiện thực trong file này được trình bày chi tiết như bên dưới.

5.1 Import thư viện và khởi tạo

Ở bước này, quan trọng nhất là để kiểm tra lại việc cài đặt thư viện có thực sự hoàn tất hay không. Bên cạnh đó, bạn đọc cũng cần phải điền thông tin cho Feed dữ liệu của mình, cũng như Username và Key của tài khoản. Những dòng lệnh đầu tiên sẽ như sau:

```
1 import sys
2 from Adafruit_IO import MQTTClient
3
4 AIO_FEED_ID = "bbc-led"
5 AIO_USERNAME = "NPNLab_BBC"
6 AIO_KEY = "aio_radR12aVJMaI2YJiGBs1V6TBR061"
```

Chương trình 3.1: Import thư viện và khởi tạo

Bạn đọc có thể thực thi chương trình ở file **main.py**, bằng cách **nhấn chuột phải vào file này và chọn Run**. Từ lần thứ 2, bạn có thể chọn nút Run trên thanh công cụ của PyCharm.

5.2 Hiện thực hàm chức năng

Việc kết nối giữa IoT Gateway và Server Adafruit dựa trên một giao thức đặc biệt, gọi là MQTT (Message Queuing Telemetry Transport). Đây là một giao thức truyền thông dựa trên cơ chế publish/subscribe, chuyên dụng cho các thiết bị Internet of Things (IoT). Bốn hàm hiện thực tiếp sau đây, phục vụ cho việc vận hành giao thức MQTT tại Gateway IoT. Việc hiện thực các hàm này được trình bày như sau:

```
1 def connected(client):
2     print("Kết nối thành công...")
3     client.subscribe(AIO_FEED_ID)
4
5 def subscribe(client, userdata, mid, granted_qos):
6     print("Subscribe thành công...")
7
8 def disconnected(client):
9     print("Ngắt kết nối...")
10    sys.exit(1)
11
12 def message(client, feed_id, payload):
13     print("Nhận dữ liệu: " + payload)
```

Chương trình 3.2: Các hàm chức năng

Khi kết nối thành công với server, Gateway sẽ subscribe vô một kênh dữ liệu để nhận dữ liệu từ nó. Khi có dữ liệu từ bất cứ nguồn nào gửi lên Feed, dữ liệu này sẽ

được tự động gửi xuống Gateway IoT, và hàm message sẽ tự động chạy mà chúng ta không cần phải xử lý nhiều.

5.3 Cấu hình cho Gateway

Bước cuối cùng trong việc hiện thực Gateway, là tạo ra một đối tượng MQTT Client, để nó có thể liên kết với các hàm chức năng đã tạo ở trên. Các câu lệnh bổ sung vào chương trình được trình bày như sau:

```
1 client = MQTTCClient(AIO_USERNAME , AIO_KEY)
2 client.on_connect = connected
3 client.on_disconnect = disconnected
4 client.on_message = message
5 client.on_subscribe = subscribe
6 client.connect()
7 client.loop_background()
8
9 while True:
10     pass
```

Chương trình 3.3: Cấu hình cho đối tượng MQTT Client

Cuối chương trình, chúng ta cần **một vòng lặp vô tận** để chương trình không được kết thúc. Như vậy, nó mới lắng nghe các thông tin gửi về từ server Adafruit và tự động gọi hàm **message** cho chúng ta. Do trong vòng lặp vô tận này, chúng ta chưa cần thực hiện chức năng nào, nên lệnh pass được thêm vào cho đúng cấu pháp trong ngôn ngữ Python.

Chương trình ở bài này được chia sẻ ở đường dẫn sau đây:

https://github.com/nplab-vn/code-manager/blob/IoT_Lab3/IoT_Lab.py

5.4 Chạy thử chương trình

Bây giờ, chúng ta sẽ cho thực thi file **main.py** để kiểm tra tính năng của nó. Khi vừa được khởi chạy, Gateway sẽ kết nối với server Adafruit IO và sau đó đăng ký (Subscribe) vào kênh Feed **BBC_LED**. Thông tin như sau sẽ được in ra màn hình nếu 2 tác vụ này thành công:

```
F:\Teaching\DaRiu\IoTGatewayPython\venv\Scripts\python.exe F:/Teaching/Dariu/IoTGatewayPython/main.py
Connected to Adafruit IO!
Ket noi thanh cong...
Subscribe thanh cong...
|
```

Hình 3.15: Gateway kết nối và đăng ký kênh thành công

Bây giờ, bạn đọc có thể mở lại Dashboard của mình và tương tác trên nút nhấn để minh họa cho việc gửi lệnh Bật/Tắt. Gần như ngay lập tức, lệnh này sẽ được chuyển xuống Gateway IoT, như kết quả ở hình bên dưới:

```
F:\Teaching\Dariu\IoTGatewayPython\venv\Scripts\python.exe F:/Teaching/Dariu/IoTGatewayPython/main.py
Connected to Adafruit IO!
Ket noi thanh cong...
Subscribe thanh cong...
Nhan du lieu: 1
Nhan du lieu: 0
|
```

Hình 3.16: Gateway nhận được dữ liệu từ Dashboard

Hiển nhiên, tất cả các dữ liệu này đều được lưu lại trên Feed dữ liệu, mà bạn đọc có thể kiểm tra lại dễ dàng.

6 Câu hỏi ôn tập

1. Các thông tin nào là cần cho việc lập trình Gateway?
 - A. Tên Feed ở dạng Key
 - B. Username
 - C. Active Key
 - D. Tất cả các thông tin trên
2. Gateway IoT kết nối với đối tượng nào dưới đây?
 - A. Server Adafruit IO
 - B. Feed dữ liệu
 - C. Dashboard
 - D. Tất cả các đối tượng trên
3. Gateway IoT đăng kí với đối tượng nào dưới đây?
 - A. Server Adafruit IO
 - B. Feed dữ liệu
 - C. Dashboard
 - D. Tất cả các đối tượng trên
4. Khi cấu hình một nút nhấn trên Dashboard, giá trị được gửi cho đối tượng nào?
 - A. Gateway IoT
 - B. Feed dữ liệu
 - C. Dashboard
 - D. Tất cả các đối tượng trên
5. Giao thức kết nối giữa Gateway IoT và Feed là gì?
 - A. HTTP
 - B. HTTPS
 - C. MQTT
 - D. Tất cả đều đúng
6. Cơ chế trọng tâm của kết nối giữa Gateway IoT và Feed là gì?
 - A. request/response
 - B. ask/wait
 - C. publish/subscribe
 - D. Không xác định được
7. Quy trình hoạt động của Gateway IoT là gì?
 - A. Kết nối - Subscribe - Nhận dữ liệu
 - B. Kết nối - Subscribe - Gửi dữ liệu
 - C. Kết nối - Subscribe - Gửi dữ liệu hoặc Nhận dữ liệu
 - D. Không xác định được

Đáp án

1. D 2. A 3. B 4. B 5. C 6. A 7. C

CHƯƠNG 4

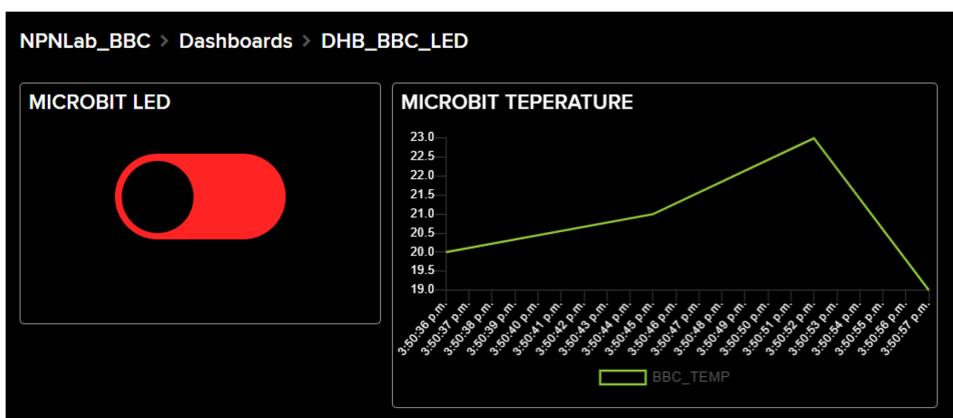
Gửi dữ liệu lên Adafruit IO



1 Giới thiệu

Trong bài hướng dẫn này, chiều giao tiếp ngược lại, với dữ liệu được gửi từ Gateway IoT lên Feed dữ liệu sẽ được trình bày. Nhu cầu này sẽ cần khi bạn đọc muốn xây dựng một ứng dụng giám sát định kì dữ liệu của hệ thống từ xa. Chẳng hạn như giám sát nhiệt độ của máy áp trứng mỗi 30 giây. Thông tin này sẽ được vẽ thành đồ thị trên Dashboard. Khi nhiệt độ đạt tới ngưỡng mong muốn, bạn đọc có thể tắt bộ phận gia nhiệt để tiết kiệm điện chẳng hạn. Bạn đọc có thể nghĩ rằng, một hệ thống thông minh có thể tự tắt khi đã đạt được nhiệt độ mong muốn. Tuy nhiên, với các hệ thống thực tế, nhu cầu điều khiển bằng tay vẫn luôn là cần thiết, cho những trường hợp ngoại lệ có thể xảy ra.

Trong bài này, chúng ta sẽ thiết kế thêm một giao diện trên Dashboard để hiện thị thông tin nhiệt độ. Tất nhiên bạn đọc hoàn toàn có thể gửi thông tin khác, tùy thuộc vào ứng dụng mà mình muốn triển khai. Giao diện trên Dashboard sẽ như minh họa sau đây:



Hình 4.1: Giao diện đồ thị trên Dashboard

Với giao thức MQTT, mỗi khi Gateway IoT gửi dữ liệu, bạn sẽ thấy thông tin này ngay lập tức cập nhật lên đồ thị. Đây là điểm rất khác biệt nếu so sánh với các server truyền thống như ThingSpeak. Thông thường, ThingSpeak có độ trễ rất lớn, khoảng hơn 5 giây dữ liệu mới được cập nhật trên đồ thị. Tuy nhiên, do chúng ta đang xài tài khoản miễn phí từ Adafruit IO, nên cũng có giới hạn về số lần gửi dữ liệu lên server. Hiện tại, chúng ta chỉ có thể gửi tối đa 1 gói tin trong 1 giây. Do vậy, đối với các ứng dụng quan trắc, vốn không nhất thiết phải gửi dữ liệu liên tục, bạn hãy gửi với chu kỳ 30 giây hoặc thậm chí là 60 giây. Chúng ta sẽ để dành tài nguyên mạng cho các nút nhấn điều khiển ngược lại thiết bị bên dưới Gateway IoT.

Mục tiêu hướng dẫn của bài được tóm tắt như sau:

- Tạo thêm một Feed dữ liệu
- Tạo giao diện đồ thị trên Dashboard
- Lập trình gửi dữ liệu từ Gateway lên Feed

2 Tạo Feed dữ liệu mới

Trước khi có thể tạo giao diện đồ thị trên Dashboard, bạn đọc cần tạo kênh dữ liệu (Feed) cho nó trước. Quy trình tạo thêm 1 kênh dữ liệu hoàn toàn tương tự như hướng dẫn ở bài trước. Trong bài này, bạn sẽ tạo thêm 1 kênh dữ liệu mới, đặt tên cho nó là **BBC_TEMP**. Bạn đọc cần vào lại mục **Feeds**, chọn tiếp vào **View All** và cuối cùng là **New Feed**.

The screenshot shows the Adafruit IO Feeds interface. At the top, there are buttons for '+ New Feed' and '+ New Group'. A search bar is on the right. Below is a table titled 'Default' with two rows:

Feed Name	Key	Last value	Recorded
BBC_LED	bbc-led	0	about 16 hours ago
BBC_TEMP	bbc-temp	19	30 minutes ago

Hình 4.2: Tạo thêm một Feed trong tài khoản

Sau khi tạo Feed mới thành công, kết quả sẽ được hiển thị như hình bên trên. Tuy nhiên bạn đọc cần truy xuất trực tiếp vào Feed để chỉnh chế độ trung cập của nó thành **Public**. Thao tác này đã được trình bày ở bài trước và sẽ không được trình bày chi tiết ở đây.

3 Thêm đồ thị cho Dashboard

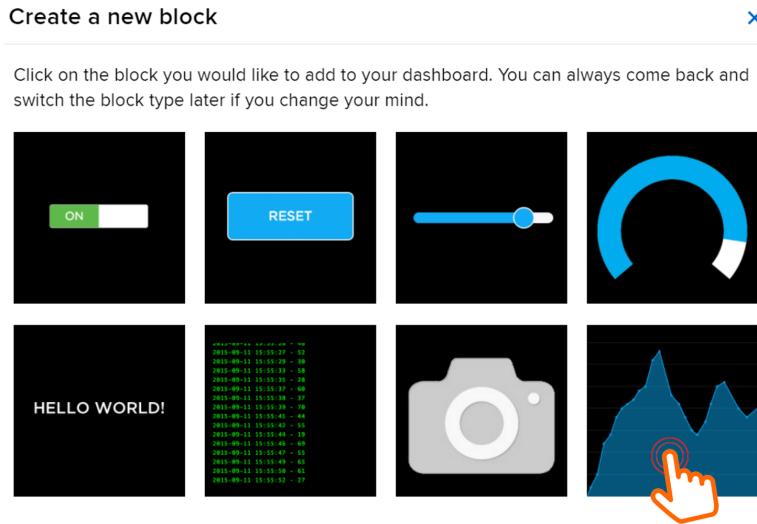
Sau khi tạo mới kênh dữ liệu cho dự án, bạn đọc có thể bắt đầu thiết kế Dashboard, với đối tượng đồ thị được thêm vào. Từ menu chính, chọn và **Dashboard** để truy cập vào chi tiết của Dashboard hiện tại (DHB_BBC_LED), như sau:

The screenshot shows the Adafruit IO Dashboard Settings menu. On the left is a preview of a dashboard block titled 'MICROBIT LED' with a red toggle switch. On the right is the 'Dashboard Settings' panel. A large orange callout with the number '1' points to the '+ Create New Block' button, which is highlighted with a yellow glow. Other settings include 'Edit Layout', 'View Fullscreen', 'Share Links', 'Dark Mode', 'Block Borders', 'Dashboard Privacy', and a 'Delete Dashboard' button.

Hình 4.3: Thiết kế giao diện cho Dashboard

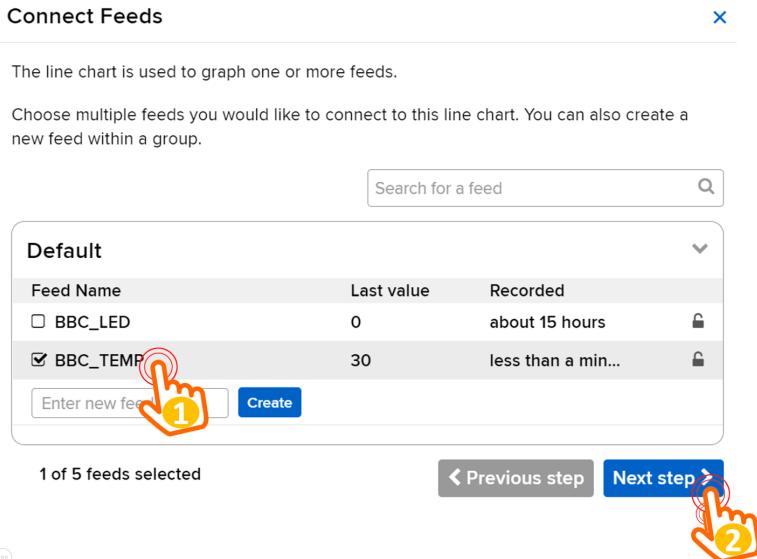
Từ biểu tượng cài đặt ở góc bên phải, bạn đọc chọn vào tính năng thêm đối tượng giao diện mới bằng cách nhấn chuột vào **+ Create New Block**. Với các đối tượng

giao diện hỗ trợ trên Adafruit IO, chúng ta sẽ chọn vào đối tượng đồ thị, như hướng dẫn bên dưới:



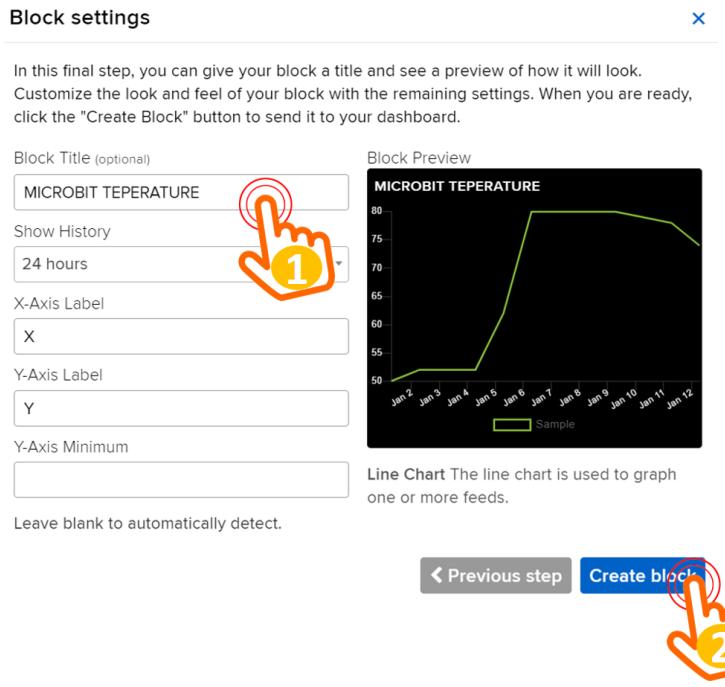
Hình 4.4: Chọn đồ thị giao diện

Tiếp theo, một giao diện sẽ được hiện ra, để chúng ta liên kết đồ thị với Feed dữ liệu. Ở đây, chúng ta sẽ chọn vào Feed **BBC_TEMPERATURE**, như minh họa ở hình bên dưới:



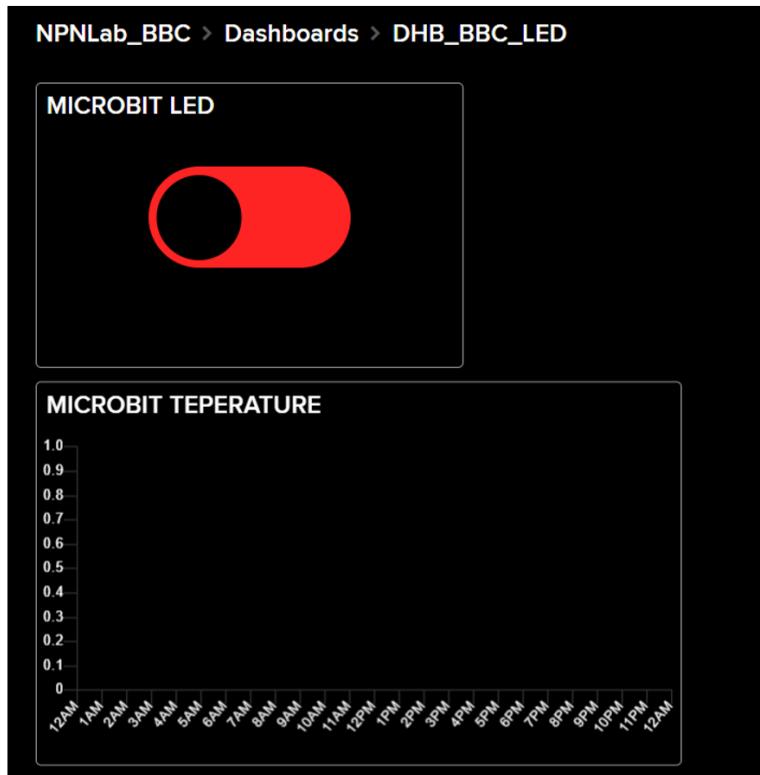
Hình 4.5: Chọn Feed dữ liệu liên kết với đồ thị

Nhấn vào nút **Next step** để cấu hình cho việc hiển thị của đồ thị. Giao diện bên dưới sẽ hiện ra:



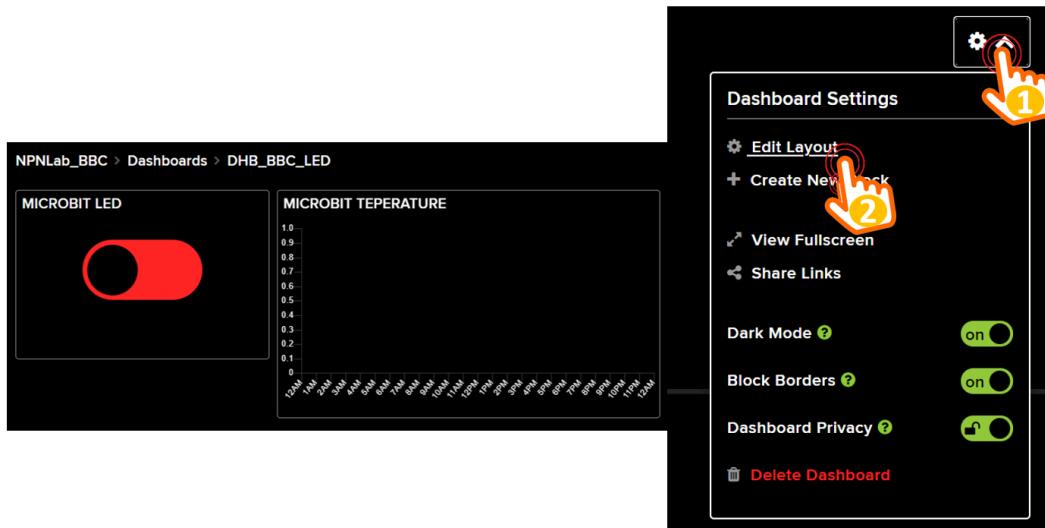
Hình 4.6: Cấu hình hiển thị trên đồ thị

Có rất nhiều thông tin mà bạn đọc có thể thay đổi. Tuy nhiên các thông tin cấu hình đều đang ở dạng mặc định mà bạn không cần phải thay đổi gì. Ở đây, chúng tôi chỉ đơn giản là thêm thông tin **Block Title**, và nhấn vào nút **Create block** ở cuối giao diện cài đặt này. Kết quả hiện tại, một đồ thị sẽ được thêm vào Dashboard như sau:



Hình 4.7: Cấu hình hiển thị trên đồ thị

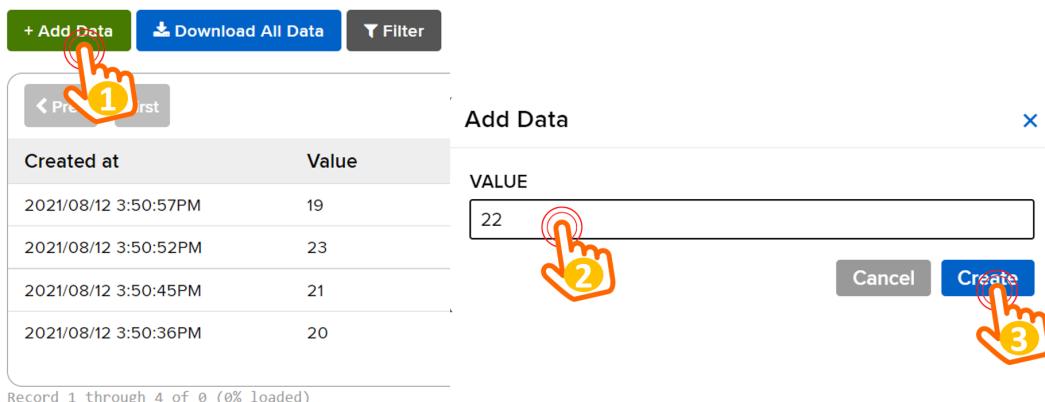
Theo chế độ mặc định, đồ thị mới thêm vào sẽ được xếp theo chiều dọc. Trong trường hợp bạn muốn sắp xếp lại nó ở vị trí khác, hay thậm chí thay đổi kích thước của đồ thị, thao tác cũng tương tự như khi chúng ta làm với nút nhấn: Chọn vào biểu tượng cài đặt, còn tiếp vào **Edit Layout**, sau đó kéo thả đối tượng trên Dashboard để thay đổi. Cuối cùng, nhấn vào nút **Save Layout** để kết thúc.



Hình 4.8: Thay đổi layout của giao diện trên Dashboard

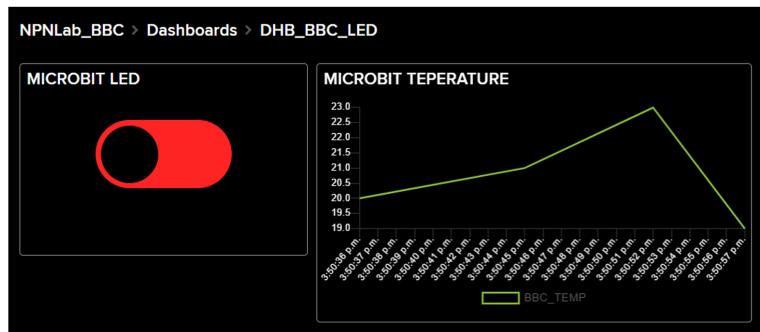
4 Kiểm tra tương tác giữa Feed và Dashboard

Đây là giai đoạn luôn cần thiết, trước khi bắt đầu lập trình cho Gateway. Chúng ta cần phải đảm bảo liên kết giữa Feed và Dashboard là hoàn thiện. Việc này được thực hiện dễ dàng bằng cách thêm dữ liệu bằng tay cho Feed dữ liệu **BBC_TEMP**, như hướng dẫn bên dưới:



Hình 4.9: Thêm dữ liệu bằng tay trên Feed

Khi đã thêm một vài dữ liệu kiểm tra, bạn đọc có thể mở lại Dashboard. Nếu việc cài đặt trên Dashboard là thành công, một đồ thị sẽ được vẽ ra, tương ứng với các giá trị thêm bằng tay trên Feed, như sau:



Hình 4.10: Giao diện đồ thị trên Dashboard

5 Lập trình cho Gateway

Sau khi mọi tác vụ trên server Adafruit, bao gồm Feed dữ liệu và Dashboard đã hoàn thiện, chúng ta đã có thể bắt đầu lập trình cho Gateway gửi dữ liệu định kì lên server. Như đã trình bày ở phần giới thiệu, chúng ta sẽ gửi một giá trị ngẫu nhiên (mô phỏng cho nhiệt độ) mỗi 30 giây. Bạn đọc cũng cần phải xem lại thông tin của Feed dữ liệu này để sử dụng cho việc lập trình. Trong trường hợp của chúng tôi, Feed dữ liệu mới có tên là "**bbc-temp**".

Khi gửi dữ liệu lên server, chúng ta sẽ không bắt buộc phải đăng ký vào kênh dữ liệu. Câu lệnh trong thư viện lập trình AdafruitIO cho phép chúng ta gửi dữ liệu (publish) bằng cách chỉnh định tên của Feed một cách trực tiếp.

Bước đầu tiên của việc lập trình, chúng ta cần thêm 2 thư viện có sẵn của hệ thống để định thời và lấy số ngẫu nhiên. Bạn đọc có thể thêm 2 câu lệnh này vào đầu chương trình:

```
1 import random
2 import time
```

Chương trình 4.1: Thêm thư viện hệ thống

Tiếp theo, chúng ta sẽ thay đổi chương trình trong vòng lặp vô tận ở cuối chương trình, như sau:

```
1 while True:
2     value = random.randint(0, 100)
3     print("Cap nhat:", value)
4     client.publish("bbc-temp", value)
5     time.sleep(30)
```

Chương trình 4.2: Thêm thư viện hệ thống

Bây giờ bạn đọc có thể kiểm tra trên Feed hoặc Dashboard. Cứ mỗi 30 giây, một giá trị ngẫu nhiên sẽ được gửi lên. Trong khi đó, mỗi khi nhấn vào nút nhấn trên Dashboard, thông tin sẽ được gửi ngay lập tức xuống Gateway. Chương trình hoàn chỉnh của chúng ta được chia sẻ ở đường dẫn sau đây:

https://github.com/nplab-vn/code-manager/blob/IoT_Lab4/IoT_Lab.py

Mã nguồn của chương trình được trình bày như bên dưới:

```
1 import random
2 import time
3 import sys
4 from Adafruit_IO import MQTTClient
5
6 AIO_FEED_ID = "bbc-led"
7 AIO_USERNAME = "NPNLab_BBC"
8 AIO_KEY = "aio_radR12aVJ Mai2YJiGBs1V6TBR061"
9
10 def connected(client):
11     print("Kết nối thành công...")
12     client.subscribe(AIO_FEED_ID)
13
14 def subscribe(client, userdata, mid, granted_qos):
15     print("Subscribe thành công...")
16
17 def disconnected(client):
18     print("Ngắt kết nối...")
19     sys.exit(1)
20
21 def message(client, feed_id, payload):
22     print("Nhận dữ liệu: " + payload)
23
24 client = MQTTClient(AIO_USERNAME, AIO_KEY)
25 client.on_connect = connected
26 client.on_disconnect = disconnected
27 client.on_message = message
28 client.on_subscribe = subscribe
29 client.connect()
30 client.loop_background()
31
32 while True:
33     value = random.randint(0, 100)
34     print("Cap nhat:", value)
35     client.publish("bbc-temp", value)
36     time.sleep(30)
```

Chương trình 4.3: Chương trình giao tiếp 2 chiều của Gateway với Server

6 Câu hỏi ôn tập

1. Cơ chế gửi dữ liệu từ Gateway lên server gọi là gì?
 - A. connect
 - B. subscribe
 - C. publish
 - D. Tất cả các thông tin trên
2. Phát biểu nào sau đây là sai?
 - A. Để nhận dữ liệu từ server, cần phải subscribe vào Feed
 - B. Để gửi dữ liệu lên server, cần phải subscribe vào Feed
 - C. Dashboard mặc định đã subscribe vào Feed
 - D. Dashboard mặc định là ở dạng Private
3. Việc gửi dữ liệu định kì trong bài này được thực hiện trong cấu trúc nào?
 - A. while True
 - B. connected()
 - C. Subscribe()
 - D. Tất cả đều đúng
4. Câu lệnh nào sau đây có hiệu ứng đợi 5 giây trên Python?
 - A. time.sleep(5)
 - B. time.sleep(5000)
 - C. delay(5000)
 - D. pause(5000)
5. Câu lệnh nào sau đây là đúng để gửi dữ liệu lên một Feed trên Adafruit IO?
 - A. client.publish("bbc-temp", 30)
 - B. client.publish("BBC-TEMP", 30)
 - C. client.publish("BBC-TEMP", "30")
 - D. Tất cả đều đúng
6. Để tạo ngẫu nhiên một số nguyên, thư viện nào đã được sử dụng?
 - A. import random
 - B. import randint
 - C. import rand
 - D. Tất cả đều sai
7. Thư viện cần thiết cho tạo hiệu ứng đợi là gì?
 - A. import time
 - B. import delay
 - C. import pause
 - D. Tất cả đều sai

Đáp án

1. C 2. B 3. A 4. A 5. A 6. A 7. A

CHƯƠNG 5

Tích hợp với Microbit



1 Giới thiệu

Đến bài này, chúng ta đã hoàn thành việc giao tiếp 2 chiều giữa Gateway chạy trên máy tính và Feed dữ liệu. Trong bài này, Gateway sẽ được tích hợp thêm việc kết nối với mạch Microbit bằng cổng kết nối USB. Mạch Microbit kết nối với máy tính sẽ được gọi là **mạch Microbit trung tâm** để phân biệt với mạch Microbit đóng vai trò làm nốt cảm biến. Cách tiếp cận này có một số lợi ích được liệt kê như sau:

- Gateway đang được hiện thực trên một máy tính. Trong tương lai, quy trình này sẽ được mang xuống máy tính nhúng. Đặc điểm chung của mọi hệ thống máy tính, đó là hệ thống thông dụng nhưng không chuyên dụng. Việc kết nối với một thiết bị chuyên dụng sẽ phải thông qua kết nối USB.
- Việc kết nối thêm thiết bị chuyên dụng, cung cấp tính năng mở rộng gần như vô hạn cho hệ thống máy tính. Trong trường hợp này, chúng ta muốn mở rộng khả năng giao tiếp không dây của mạch Microbit cho Gateway IoT trên máy tính.
- Với khả năng kết nối không dây từ Microbit, các nốt cảm biến sử dụng Microbit có thể dễ dàng gửi dữ liệu không dây về Gateway.

Trong bài này, chúng ta sẽ tập trung vào chiều giao tiếp từ Gateway xuống mạch Microbit nối trực tiếp vào Gateway. Cụ thể, khi điều khiển nút nhấn trên Dashboard, dữ liệu 0 và 1 sẽ được hiển thị trên mạch Microbit.

Mặc dù mạch Microbit được kết nối với máy tính thông qua kết nối USB, giao tiếp của nó thực ra có một tên gọi khác, gọi là giao tiếp nối tiếp qua cổng COM ảo. Đây là giao thức thông dụng nhất cho mọi hệ thống vi điều khiển (như mạch Microbit) và hệ thống máy tính hoặc máy tính nhúng. Giao tiếp nối tiếp cũng được hỗ trợ hoàn chỉnh trên các ngôn ngữ lập trình, từ kéo thả cho đến Python.

Các mục tiêu trong bài hướng dẫn này như sau:

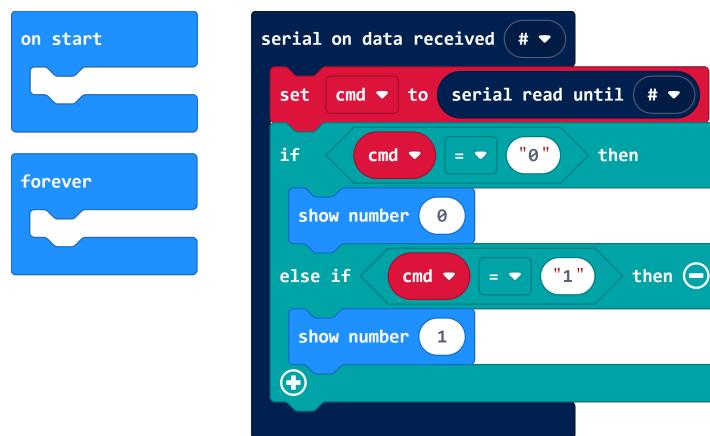
- Lập trình nhận dữ liệu trên mạch Microbit
- Cài đặt thư viện cho giao tiếp nối tiếp Serial
- Hiện thực chương trình gửi lệnh từ Gateway cho Microbit

Để thuận tiện cho bạn đọc, các đoạn chương trình Python nhỏ sẽ được giới thiệu ở từng phần. Tuy nhiên, ở phần cuối cùng của bài hướng dẫn này, chúng tôi sẽ trình bày toàn bộ chương trình. Trong quá trình đọc hiểu tài liệu, bạn đọc có thể tham khảo chương trình ở cuối để có thể tìm được vị trí tích hợp chương trình chức năng cho phù hợp.

2 Chương trình cho Microbit

Hiển nhiên, khi Gateway sẽ gửi lệnh, Microbit cần phải có một chương trình trên nó để nhận lệnh. Chúng ta sẽ hiện thực chương trình này trên mạch Microbit, trước khi tích hợp nó với Gateway. Chương trình này sẽ được hiện thực trên MakeCode, và mạch Microbit chỉ đơn giản là hiện số 0 hoặc 1 trên màn hình của nó.

Tuy nhiên, để việc lập trình trên MakeCode đơn giản hơn, chúng ta sẽ quy định một lệnh gửi từ GatewayIoT sẽ kết thúc bằng một kí tự đặc biệt #. Đây là kí thuật giao tiếp đơn giản, dựa trên một kí tự đặc biệt làm kí tự kết thúc câu lệnh. Chương trình trên MakeCode sẽ như sau:



Hình 5.1: Chương trình trên mạch Microbit trung tâm

Tạm thời, đối với mạch Microbit trung tâm, chúng ta chưa có các câu lệnh cần trong khối **on start** hay khối **forever**. Tuy nhiên bạn đọc nên giữ lại nó để phát triển thêm trong tương lai.

Một lưu ý quan trọng, dữ liệu nhận được từ giao tiếp nối tiếp là **dữ liệu chuỗi**. Do đó, khi muốn xử lý một lệnh, phép toán so sánh chuỗi cần được sử dụng cho chính xác (có kí tự "" đặc trưng cho dữ liệu chuỗi). Chương trình này được chia sẻ ở đường dẫn sau đây:

https://makecode.microbit.org/_5Rj8TeeM1RFb

Sau khi nạp chương trình này vào mạch Microbit, chúng ta đã có thể bắt đầu lập trình cho Gateway IoT, với các bước chi tiết được trình bày ở phần tiếp theo.

3 Lập trình cho Gateway

3.1 Thêm thư viện lập trình

Cũng như mọi tính năng cao cấp trên Python, việc thêm thư viện sẽ là bước đầu tiên để lập trình. Nếu như bạn đọc chọn việc cài đặt bằng tay, câu lệnh từ cửa sổ Terminal sẽ là **pip install pyserial**, như sau:

```
(venv) F:\Teaching\Dariu\IoTGatewayPython>pip install pyserial
Collecting pyserial
  Using cached pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
  Successfully installed pyserial-3.5
```

★ (venv) F:\Teaching\Dariu\IoTGatewayPython>

Hình 5.2: Cài đặt thư viện pyserial

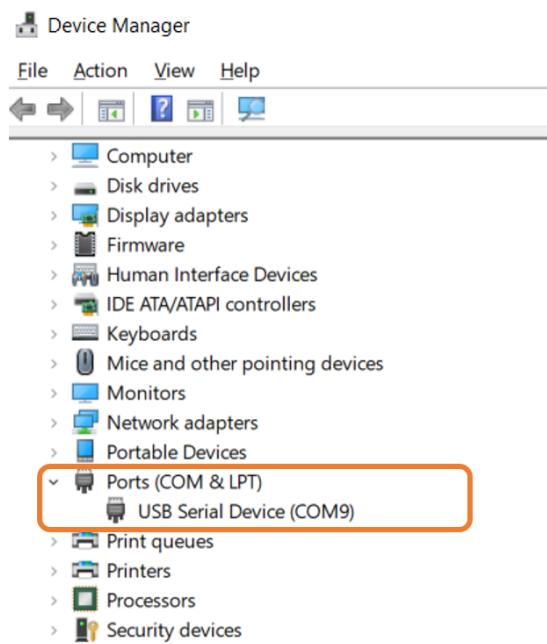
Sau khi cài đặt thư viện thành công, bạn đọc có thể thêm câu lệnh import thư viện bên dưới vào đầu chương trình.

```
1 import serial.tools.list_ports
```

Chương trình 5.1: Thêm thư viện lập trình Serial

3.2 Thêm thư viện cho kết nối Serial

Khi kết nối mạch Microbit với máy tính, nó sẽ được nhận dạng như một cổng COM ảo. Điều quan trọng nhất, là chúng ta phải xác định được tên của cổng COM này. Như minh họa ở hình dưới đây, trong Device Manager, mạch Microbit đang được kết nối với cổng COM9.



Hình 5.3: Tên cổng COM kết nối với mạch Microbit

Tuy nhiên, tên cổng COM này hoàn toàn có thể bị thay đổi, mỗi khi khởi động lại máy tính, hoặc khi chúng ta thay đổi cổng kết nối USB. Do vậy, chúng tôi hỗ trợ bạn đọc một hàm để tự động tìm ra cổng COM này, như hướng dẫn bên dưới:

```

1 def getPort():
2     ports = serial.tools.list_ports.comports()
3     N = len(ports)
4     commPort = "None"
5     for i in range(0, N):
6         port = ports[i]
7         strPort = str(port)
8         if "USB Serial Device" in strPort:
9             splitPort = strPort.split(" ")
10            commPort = (splitPort[0])
11
12 return commPort

```

Chương trình 5.2: Tìm tên cổng COM kết nối với Microbit

Tuy nhiên, hàm này chỉ hoạt động tốt khi có 1 mạch Microbit cắm vào hệ thống. Nhờ driver USB của mạch này, chương trình có thể tự động nhận dạng được. Trong trường hợp không nhận dạng được, bạn đọc cần kiểm tra lại trong Device Manager để xem tên của thiết bị có từ khóa "**USB Serial Device**" hay không.

Câu lệnh cho việc mở cổng kết nối Serial với mạch Microbit, sử dụng hàm trình bày ở trên sẽ như sau:

```

1 ser = serial.Serial( port=getPort() , baudrate=115200)

```

Chương trình 5.3: Mở kết nối với Microbit

Trong câu lệnh trên, chúng ta chỉ định ra tên cổng COM bằng cách gọi hàm phụ trợ **getPort()** và tốc độ giao tiếp, được cấu hình theo tốc độ mặc định của Microbit là 115200 bits/s.

Khi đã có thể mở kết nối thành công, việc gửi dữ liệu cho mạch Microbit rất đơn giản. Bạn đọc có thể tham khảo ở chương trình ở phần tiếp theo: Khi nhận được dữ liệu từ server, dữ liệu này sẽ gửi tiếp đến mạch Microbit.

4 Tích hợp với Microbit

Chương trình cho hệ thống bây giờ đã khá dài, được trình bày như bên dưới. Bạn đọc hãy chú ý câu lệnh quan trọng, ở dòng 24: Khi nhận được thông tin từ Feed dữ liệu, biến payload được đóng gói để gửi xuống mạch Microbit. Một kí tự đặc biệt # được thêm vào payload trước khi gửi xuống mạch Microbit. Chương trình được chia sẻ ở đường dẫn sau đây:

https://github.com/nplab-vn/code-manager/blob/IoT_Lab5/IoT_Lab.py

Mã nguồn của chương trình được trình bày như bên dưới:

```

1 import serial.tools.list_ports
2 import random
3 import time
4 import sys
5 from Adafruit_IO import MQTTClient
6

```

```

7 AIO_FEED_ID = "bbc-led"
8 AIO_USERNAME = "NPNLab_BBC"
9 AIO_KEY = "aio_radR12aVJ Mai2YJiGBs1V6TBR061"
10
11 def connected(client):
12     print("Kết nối thành công...")
13     client.subscribe(AIO_FEED_ID)
14
15 def subscribe(client , userdata , mid , granted_qos):
16     print("Subscribe thành công...")
17
18 def disconnected(client):
19     print("Ngắt kết nối...")
20     sys.exit (1)
21
22 def message(client , feed_id , payload):
23     print("Nhận dữ liệu: " + payload)
24     ser.write((str(payload) + "#").encode())
25
26 def getPort():
27     ports = serial.tools.list_ports.comports()
28     N = len(ports)
29     commPort = "None"
30     for i in range(0, N):
31         port = ports[i]
32         strPort = str(port)
33         if "USB Serial Device" in strPort:
34             splitPort = strPort.split(" ")
35             commPort = (splitPort[0])
36     return commPort
37
38 ser = serial.Serial( port=getPort() , baudrate=115200)
39
40 client = MQTTClient(AIO_USERNAME , AIO_KEY)
41 client.on_connect = connected
42 client.on_disconnect = disconnected
43 client.on_message = message
44 client.on_subscribe = subscribe
45 client.connect()
46 client.loop_background()
47
48 while True:
49     value = random.randint(0, 100)
50     print("Cap nhat:", value)
51     client.publish("bbc-temp" , value)
52     time.sleep(30)

```

Chương trình 5.4: Mở kết nối với Microbit

5 Câu hỏi ôn tập

1. Kết nối giữa Gateway và Microbit có tên là gì?
 - A. Serial
 - B. SPI
 - C. I2C
 - D. Tất cả đều đúng
2. Thư viện cài đặt thêm cho việc lập trình kết nối với Microbit là gì?
 - A. serial
 - B. uart
 - C. pyserial
 - D. pip
3. Tốc độ mặc định của giao tiếp serial với mạch Microbit là bao nhiêu?
 - A. 4800
 - B. 9600
 - C. 115200
 - D. Tất cả đều sai
4. Kí tự nào sau đây có thể được dùng làm kí tự kết thúc cho câu lệnh gửi từ Gateway xuống Microbit trung tâm?
 - A. #
 - B. :
 - C. \$
 - D. Tất cả đều có thể
5. Khối lệnh dùng để nhận dữ liệu trên Microbit là gì?
 - A. on radio received number
 - B. on radio received string
 - C. serial on data received #
 - D. Tất cả đều sai
6. Trên Microbit, dữ liệu nhận được từ serial có kiểu dữ liệu là?
 - A. số nguyên
 - B. số thực
 - C. chuỗi
 - D. Không xác định được
7. Câu lệnh gửi dữ liệu từ Gateway đến mạch Microbit trung tâm được hiện thực ở hàm nào?
 - A. connected
 - B. subscribe
 - C. message
 - D. getPort

Đáp án

1. A 2. C 3. C 4. D 5. C 6. C 7. C

CHƯƠNG 6

Gửi dữ liệu từ Microbit tới Gateway

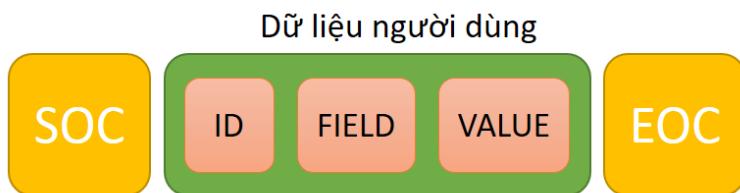


1 Giới thiệu

Trong bài này, chiêu giao tiếp thứ 2 của mạch Microbit tới Gateway IoT sẽ được hiện thực. Chức năng này sẽ có độ phức tạp lớn hơn nhiều so với chức năng hướng dẫn ở bài trước. Vấn đề ở đây là khi một chuỗi thông tin gửi từ Microbit lên Gateway, sẽ bị cắt ra chứ không được nhận một cách liên tục được. Gateway của chúng ta sẽ không biết mỗi lần nó nhận được bao nhiêu kí tự cũng như số lần nhận được. Do đó, thông tin gửi từ mạch Microbit lên cũng phải có quy định về kí tự bắt đầu và kí tự kết thúc, để thông báo là Gateway đã nhận đủ dữ liệu và bắt đầu xử lý thông tin.

Hạn chế trình bày ở trên là hạn chế chung của các hệ thống sử dụng hệ điều hành, như máy tính nói chung và máy tính nhúng nói riêng. Các hệ thống này, có những lợi thế như kết nối mạng ổn định, hỗ trợ nhiều dịch vụ mới trên mạng Internet. Tuy nhiên, một số tác vụ, chẳng hạn như nhận dữ liệu từ cổng nối tiếp, lại là thế giới riêng của Vi điều khiển như mạch Microbit.

Mặc dù vậy, chúng ta cũng có những mô hình kinh điển để xử lý cho việc này. Bằng việc đóng gói dữ liệu theo một định dạng có sẵn, như minh họa ở hình bên dưới:



Hình 6.1: Cấu trúc dữ liệu giao tiếp với máy tính

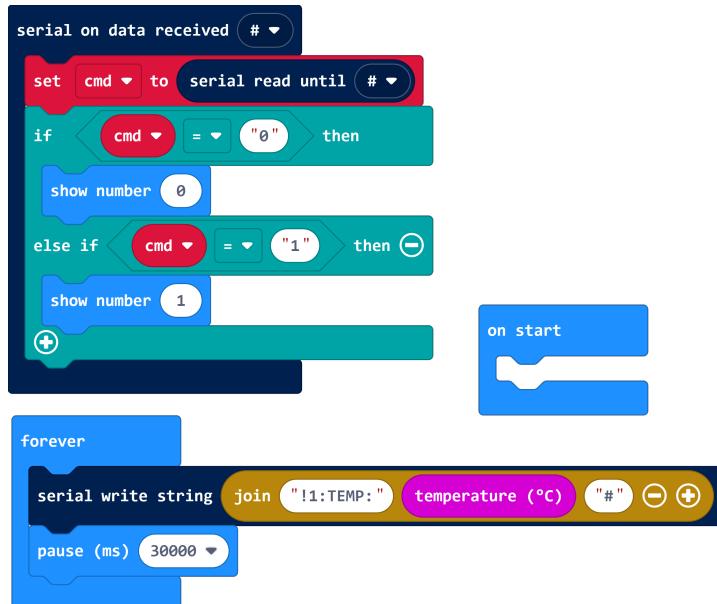
Cấu trúc ở trên là một dạng kinh điển trong giao tiếp máy tính (Computer Communications). Để áp dụng trong giáo trình này, chúng ta sẽ quy định kí tự bắt đầu (SOC = Start Of Character) là "!" và kí tự kết thúc (EOC = End Of Character) là "#". Trường thông tin ở giữa, chúng ta sẽ định nghĩa là **ID:FIELD:VALUE**, trong đó ID là định danh của một nốt, FIELD là thông tin định nghĩa và VALUE là giá trị cần gửi. Các trường này cách nhau bằng ":" - thuật ngữ chuyên ngành gọi là Escape Character.

Các mục đích chính của bài hướng dẫn này như sau:

- Hiện thực chương trình gửi dữ liệu nhiệt độ trên Microbit
- Đóng gói dữ liệu trên Microbit
- Giải mã dữ liệu trên Gateway
- Nhận và giả mã dữ liệu trên Gateway
- Gửi dữ liệu lên Adafruit server

2 Chương trình cho Microbit

Trước khi bắt đầu hiện thực chức năng nhận dữ liệu trên Gateway, chúng ta cần phải hiện thực chức năng gửi dữ liệu từ mạch Microbit. Ở đây, chúng tôi cho định kì 30 giây, thông tin nhiệt độ sẽ được gửi lên. Định dạng dữ liệu quy định cho việc giao tiếp là "!1:TEMP:xx#". Trong đó, 1 là thông tin ID của nốt Gateway, TEMP là từ khóa mà chúng ta tự định nghĩa cho thông tin nhiệt độ. Cuối cùng, là nhiệt độ đọc từ cảm biến của mạch Microbit. Gói tin kết thúc bằng kí tự #.



Hình 6.2: Chương trình gửi dữ liệu cho Microbit

Tính năng gửi dữ liệu định kì sẽ được hiện thực trong khối **forever**. Câu lệnh **serial write string** được lấy từ nhóm **Advance/Serial**. Chương trình trên được chia sẻ ở đường dẫn sau đây:

https://makecode.microbit.org/_AXYRzvURMJVY

3 Hàm phân tách dữ liệu

Như đã trình bày ở trên, dữ liệu chúng ta nhận được sẽ phải được phân tách ra, dựa vào kí tự đặc biệt là :. Các kí tự khác như kí tự bắt đầu ! và kí tự kết thúc # sẽ được loại bỏ. Tùy vào từ khóa của dữ liệu, mà thông tin sẽ được gửi lên feed tương ứng. Hàm này sẽ được hiện thực như sau:

```
1 def processData(data):
2     data = data.replace("!", "")
3     data = data.replace("#", "")
4     splitData = data.split(":")
5     print(splitData)
6     if splitData[1] == "TEMP":
7         client.publish("bbc-temp", splitData[2])
```

Chương trình 6.1: Phân tách dữ liệu và gửi lên feed

Trong trường hợp kĩ hơn, bạn đọc có thể tự xây dựng thêm các điều kiện của câu lệnh **if**, chẳng hạn như kiểm tra rằng số phần tử trong mảng **splitData** phải bằng 3 chẳng hạn. Khi các điều kiện kiểm tra đã hợp lệ, dữ liệu sẽ được gửi lên feed bằng câu lệnh **publish**. **Bạn đọc cần lưu ý về tham số đầu tiên trong lệnh publish, nó là FEED_ID của kênh dữ liệu.**

4 Hàm đọc dữ liệu Serial

Đây là hàm quan trọng nhất để nhận dữ liệu từ mạch Microbit. Vì tính chất đặc biệt của máy tính, là nó không biết dữ liệu sẽ được gửi lên bao nhiêu lần, và mỗi lần là bao nhiêu kí tự, việc xử lý ở hàm này tương đối phức tạp. Tuy nhiên, ý tưởng chính là chờ nhận đủ kí tự bắt đầu ("!") và kí tự kết thúc ("#"), chúng ta mới bắt đầu xử lý, bằng cách gọi hàm **processData** ở trên. Các câu lệnh sử dụng trong hàm này, chủ yếu là các tác vụ liên quan đến xử lý chuỗi.

```
1 mess = ""
2 def readSerial():
3     bytesToRead = ser.inWaiting()
4     if (bytesToRead > 0):
5         global mess
6         mess = mess + ser.read(bytesToRead).decode("UTF-8")
7         while ("#" in mess) and ("!" in mess):
8             start = mess.find("!")
9             end = mess.find("#")
10            processData(mess[start:end + 1])
11            if (end == len(mess)):
12                mess = ""
13            else:
14                mess = mess[end+1:]
```

Chương trình 6.2: Nhận dữ liệu từ Microbit

Trong hàm này, chúng ta cần 1 biến toàn cục, đặt tên là **mess** để cộng dồn dữ liệu nhận được từ Microbit. Bạn đọc hãy để ý về cách chỉ định một biến thành truy cập toàn cục bằng câu lệnh ở dòng 5, **global mess**.

5 Tích hợp vào Gateway

Cho đến lúc này, chúng ta đã có thể tích hợp vào chương trình Gateway, với vòng lặp vô tận sẽ định kì đọc dữ liệu từ mạch Microbit. Khi nào dữ liệu đầy đủ, nó sẽ tự động được xử lý và gửi lên server Adafruit. Thời gian đợi đang được chỉnh là 1 giây, bằng câu lệnh **time.sleep(1)**. Chương trình trọn vẹn của Gateway được chia sẻ ở đường dẫn sau đây:

https://github.com/npnlab-vn/code-manager/blob/IoT_Lab6/IoT_Lab.py

Mã nguồn cho chương trình đến bước này, được trình bày như sau:

```
1 import serial.tools.list_ports
2 import random
```

```

3 import time
4 import sys
5 from Adafruit_IO import MQTTClient
6
7 AIO_FEED_ID = "bbc-led"
8 AIO_USERNAME = "NPNLab_BBC"
9 AIO_KEY = "aio_radR12aVJMa12YJiGBs1V6TBR061"
10
11 def connected(client):
12     print("Ket noi thanh cong...")
13     client.subscribe(AIO_FEED_ID)
14
15 def subscribe(client , userdata , mid , granted_qos):
16     print("Subscribe thanh cong...")
17
18 def disconnected(client):
19     print("Ngat ket noi...")
20     sys.exit (1)
21
22 def message(client , feed_id , payload):
23     print("Nhan du lieu: " + payload)
24     ser.write((str(payload) + "#").encode())
25
26 client = MQTTClient(AIO_USERNAME , AIO_KEY)
27 client.on_connect = connected
28 client.on_disconnect = disconnected
29 client.on_message = message
30 client.on_subscribe = subscribe
31 client.connect()
32 client.loop_background()
33
34 def getPort():
35     ports = serial.tools.list_ports.comports()
36     N = len(ports)
37     commPort = "None"
38     for i in range(0, N):
39         port = ports[i]
40         strPort = str(port)
41         if "USB Serial Device" in strPort:
42             splitPort = strPort.split(" ")
43             commPort = (splitPort[0])
44     return commPort
45
46 ser = serial.Serial( port=getPort() , baudrate=115200)
47
48 mess = ""
49 def processData(data):
50     data = data.replace("!", "")
51     data = data.replace("#", "")

```

```

52     splitData = data.split(":")
53     print(splitData)
54     if splitData[1] == "TEMP":
55         client.publish("bbc-temp", splitData[2])
56
57 mess = ""
58 def readSerial():
59     bytesToRead = ser.inWaiting()
60     if (bytesToRead > 0):
61         global mess
62         mess = mess + ser.read(bytesToRead).decode("UTF-8")
63         while ("#" in mess) and ("!" in mess):
64             start = mess.find("!")
65             end = mess.find("#")
66             processData(mess[start:end + 1])
67             if (end == len(mess)):
68                 mess = ""
69             else:
70                 mess = mess[end+1:]
71
72 while True:
73     readSerial()
74     time.sleep(1)

```

Chương trình 6.3: Tích hợp chương trình vào Gateway IoT

6 Câu hỏi ôn tập

1. Để nhận được dữ liệu từ cổng nối tiếp, điểm hạn chế tại Gateway là gì?
 - A. Không biết dữ liệu sẽ nhận được bao nhiêu lần, mỗi lần bao nhiêu kí tự
 - B. Dữ liệu được nhận một lần duy nhất
 - C. Mất dữ liệu khi đọc từ cổng Serial
 - D. Tất cả đều đúng
2. Kĩ thuật xử lý nhận dữ liệu Serial ở Gateway là gì?
 - A. Khai báo biến toàn cục để cộng dồn dữ liệu
 - B. Xử lý thông tin dựa vào kí tự bắt đầu và kết thúc của thông tin
 - C. Cắt thông tin dựa vào các kí tự đặt biệt
 - D. Tất cả các kĩ thuật trên
3. Với định dạng "**!ID:KEY:VALUE#**", chuỗi kí tự nào sau đây là hợp lệ?
 - A. !1TEMP30#
 - B. !1:TEMP:30#
 - C. !2:HUMI:60#
 - D. Có 2 chuỗi hợp lệ
4. Tăng thời gian đợi trong vòng lặp while True có ảnh hưởng gì đến việc xử lý nhận dữ liệu từ Serial?
 - A. Làm mất dữ liệu
 - B. Làm tăng độ trễ khi gửi dữ liệu cảm biến lên server
 - C. Không ảnh hưởng gì cả
 - D. Tất cả đều sai
5. Để phân biệt dữ liệu được gửi từ nốt nào của mạng cảm biến, thông tin nào sau đây sẽ được xem xét?
 - A. ID
 - B. KEY
 - C. VALUE
 - D. Tất cả các thông tin trên
6. Để nhận biết ý nghĩa của dữ liệu (nhiệt độ hay độ ẩm), thông tin nào sau đây sẽ được xem xét?
 - A. ID
 - B. KEY
 - C. VALUE
 - D. Tất cả các thông tin trên
7. Để truy cập vào giá trị của cảm biến, thông tin nào sau đây sẽ được sử dụng?
 - A. ID
 - B. KEY
 - C. VALUE
 - D. Tất cả các thông tin trên

Đáp án

1. A 2. D 3. D 4. B 5. A 6. B 7. C

CHƯƠNG 7

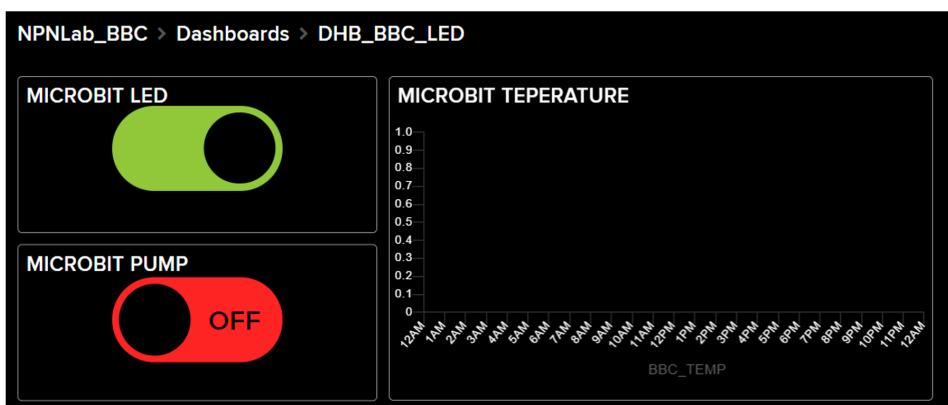
Nhiều nút nhấn trên Dashboard



1 Giới thiệu

Hiện tại, chúng ta đã xây dựng được hệ thống tương đối đầy đủ tính năng ở Gateway, với sự kết hợp của chương trình Python chạy trên máy tính và kết nối thêm mạch Microbit. Hệ thống đã có thể nhận được dữ liệu từ một nút nhấn khi người dùng tương tác trên Dashboard và dữ liệu sẽ được gửi tới mạch Microbit trung tâm - đang kết nối với chương trình Python trên máy tính.

Trong bài này, chúng ta sẽ mở rộng tính năng trên Dashboard, với thêm 1 nút nhấn nữa được thêm vào. Bạn đọc hãy lưu ý rằng, để nhận được dữ liệu từ Adafruit server, Gateway cần phải đăng kí (subscribe) với kênh dữ liệu, het như việc chúng ta phải đăng kí vào kênh Youtube để nhận được thông tin mới từ chủ kênh vậy. Ở các bài hướng dẫn trước, chúng ta đã đăng kí vào kênh dữ liệu liên quan đến việc bật tắt một bóng đèn. Bài này, chúng ta sẽ tạo một kênh mới, để bật tắt một máy bơm chẳng hạn.



Hình 7.1: Thêm nút nhấn điều khiển cho Dashboard

Chúng ta cũng sẽ hiện thực lại mạch Microbit trung trung cho nó hiện thực đúng vai trò của nó: Mỗi khi nhận được lệnh từ Gateway, nó sẽ dùng tính năng Radio để gửi lệnh này đến các nốt cảm biến. Ngược lại, mỗi khi nhận được bất kì thông tin nào từ các nốt cảm biến (qua giao tiếp Radio), nó sẽ gửi trọn vẹn thông tin đó lên Gateway để xử lý. Việc dùng mạch Microbit trung tâm ở các bài trước, chủ yếu là để kiểm tra tính năng giao tiếp của Gateway mà thôi.

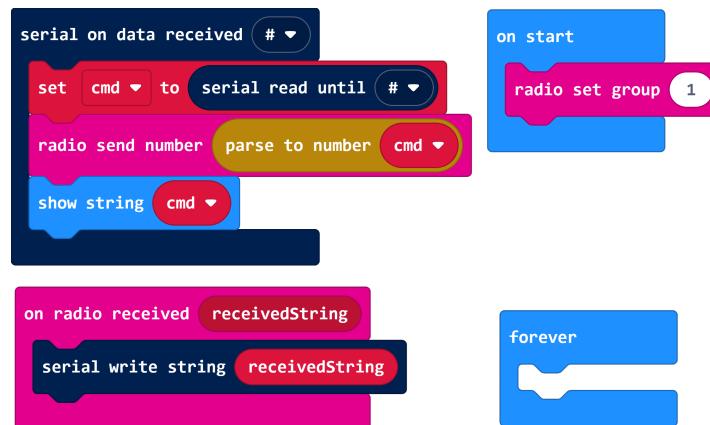
Các mục tiêu hướng dẫn trong bài này sẽ như sau:

- Hoàn thiện chương trình cho nốt Microbit trung tâm
- Tạo thêm một kênh dữ liệu và nút nhấn trên Dashboard
- Chỉnh sửa chương trình cho Gateway

2 Chương trình cho Microbit trung tâm

Với 2 tính năng định nghĩa ở phần giới thiệu, mạch trung tâm trở thành thiết bị thu động hoàn toàn. Nó không còn định kì gửi dữ liệu cảm biến của chính nó lên

Gateway nữa, mà sẽ chờ nhận dữ liệu từ các nốt cảm biến xung quanh. Song song đó, mỗi khi có lệnh từ Gateway (người dùng nhấn nút trên Dashboard), nó cũng chỉ đơn giản là chuyển tiếp lệnh đó đi cho các nốt cảm biến thực thi. Trong một số ứng dụng đơn giản, bạn đọc có thể tận dụng mạch Microbit trung tâm làm vai trò của mạch cảm biến và thực thi. Tuy nhiên trong hướng dẫn này, chúng tôi sẽ đi theo hướng tổng quát, và biến **Microbit trung tâm thành một cầu nối với các mạch cảm biến**. Chương trình mới cho mạch Microbit trung tâm sẽ như sau:



Hình 7.2: Chương trình cho mạch Microbit trung tâm

Chương trình trong khối **forever** sẽ không còn nữa. Việc đọc giá trị cảm biến và đóng gói nó, sẽ dành cho các nốt cảm biến, sẽ được trình bày ở bài sau. Trong khi đó, khi nhận được một lệnh từ Gateway, Microbit trung tâm chuyển dữ liệu này thành kiểu số, trước khi gửi đi bằng câu lệnh **radio send number**. Nhóm của các mạch Microbit trong cùng một hệ thống được quy định trong khối **on start** với câu lệnh **radio set group**.

Khi chuyển mạch Microbit trung tâm thành một cầu nối giao tiếp không dây, chúng ta hình thành nên một mạng cảm biến không dây chuyên dụng với nhu cầu của ứng dụng. Với cách tiếp cận này, nếu muốn chuyển sang các loại giao tiếp không dây khác, chẳng hạn như LoRa, cũng có thể làm được dễ dàng. Khi đó, chúng ta sẽ nối thêm thiết bị LoRa vào tất cả các mạch Microbit của hệ thống. Chương trình cho mạch Microbit trung tâm được chia sẻ ở đường dẫn sau đây:

https://makecode.microbit.org/_7dFKhAecaf4A

3 Tạo mới Feed dữ liệu

Đây là bước đầu tiên, trước khi bạn tạo giao diện ở Dashboard. Quy trình tạo mới một Feed cũng như bài trước và sẽ không trình bày chi tiết ở đây. Bạn đọc cần lưu ý là chỉnh chế độ truy cập **Public** cho Feed mới. Ở đây, chúng tôi tạo một Feed mới có tên là **BBC_PUMP**, dùng để lưu dữ liệu điều khiển cho máy bơm.

Feed Name	Key	Last value	Recorded
<input type="checkbox"/> BBC_LED	bbc-led	0	less than a minute ago
<input checked="" type="checkbox"/> BBC_PUMP	bbc-pump	3	less than a minute ago
<input type="checkbox"/> BBC_TEMP	bbc-temp	34	2 days ago

Hình 7.3: Thêm Feed dữ liệu cho Máy bơm

Bạn đọc hãy lưu ý về giá trị Key của Feed mới tạo ra, để sử dụng cho việc lập trình, trong trường hợp ở hình trên là **bbc-pump**.

4 Tạo mới nút nhấn trên Dashboard

Để thêm một nút nhấn mới trên Dashboard, bạn đọc cũng làm tương tự ở các bài trước: Nhấn vào biểu tượng cài đặt, và chọn và **Create New Block** để chọn vào biểu tượng nút nhấn. Tuy nhiên, lần này bạn sẽ chọn liên kết nút nhấn mới với Feed dữ liệu mới tạo ra ở bài trước, như kết quả ở hình sau đây:

Default

Feed Name	Last value	Recorded
<input type="checkbox"/> BBC_LED	0	19 minutes
<input checked="" type="checkbox"/> BBC_PUMP	3	19 minutes
<input type="checkbox"/> BBC_TEMP	34	2 days

Enter new feed name Create

1 of 1 feeds selected

[Previous step](#) [Next step](#)

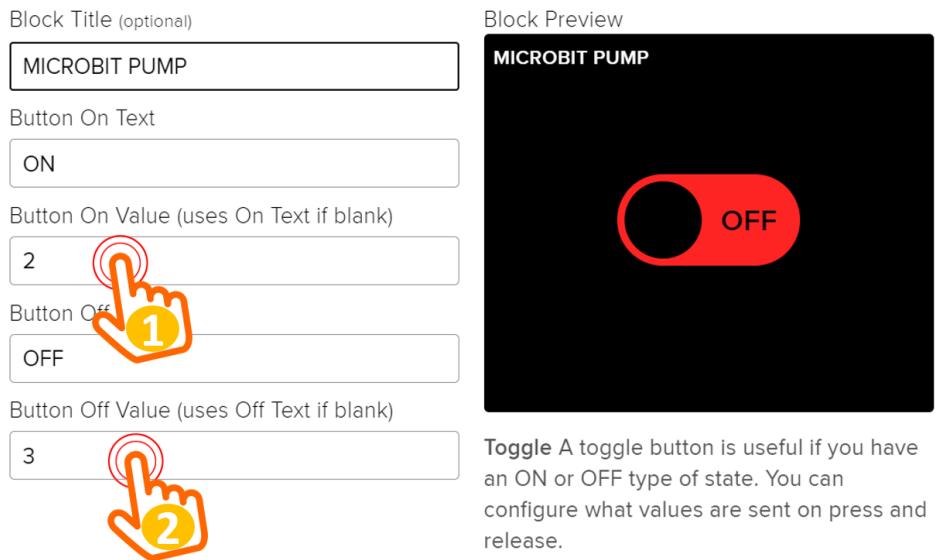
Hình 7.4: Thêm nút nhấn điều khiển trên Dashboard

Cuối cùng, bạn đọc nên cấu hình 2 giá trị khác nhau cho nút nhấn ở bài này. Với 2 giá trị 0 và 1 cho nút nhấn LED, chúng ta sẽ chọn 2 và 3 cho nút nhấn thứ này. Điều này sẽ đơn giản hóa việc lập trình trong tương lai, như minh họa ở hình bên dưới:

Block settings

X

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.



Hình 7.5: Thêm nút nhấn điều khiển trên Dashboard

Sau khi tạo xong nút nhấn mới, bạn đọc cũng phải kiểm tra lại việc giao tiếp dữ liệu giữa Dashboard và các Feed dữ liệu. **Trong trường hợp tương tác dữ liệu trên Dashboard không thành công, bạn đọc hãy thử Refresh lại trang này.** Trong lần đầu sau khi tạo mới một đối tượng giao diện mới, có thể Dashboard sẽ không hoạt động đúng chức năng.

5 Cài tiến chương trình của Gateway

Như đã trình bày ở phần giới thiệu, để có thể nhận dữ liệu từ một nút nhấn (hay một đối tượng điều khiển nào trên Dashboard), Gateway của chúng ta cần phải đăng kí vào kênh dữ liệu của nó. Hiện tại, chúng ta đang có 2 kênh dữ liệu, là "**bbc-led**" và "**bbc-pump**".

Để tổng quát việc lập trình trong tương lai của bạn đọc, với nhiều nút nhấn được thiết kế (tương ứng với nhiều kênh dữ liệu), chúng tôi đưa ra một khung chương trình tổng quát, với các điểm lưu ý quan trọng sau đây:

Khái báo mảng các Feeds cần đăng kí: Biến **AIO_FEED_ID** được điều chỉnh, để lưu một mảng gồm nhiều kênh dữ liệu cần đăng kí, như minh họa ở hình bên dưới:

```
1 AIO_FEED_ID = ["bbc-led", "bbc-pump"]
```

Chương trình 7.1: Khai báo biến AIO_FEED_ID thành mảng

Trong trường hợp có nhiều kênh hơn, bạn đọc cần liệt kê nó thêm trong mảng này.

Sửa lại hàm connected: Thay vì chỉ đăng kí một kênh như ban đầu, một vòng lặp **for** sẽ được sử dụng để đăng kí tất cả các kênh được liệt kê trong mảng **AIO_FEED_ID**

```
1 AIO_FEED_ID = ["bbc-led", "bbc-pump"]
2 def connected(client):
3     print("Kết nối thành công...")
4     for feed in AIO_FEED_ID:
5         client.subscribe(feed)
```

Chương trình 7.2: Khai báo biến AIO_FEED_ID thành mảng

Chương trình khi chạy, sẽ in ra kết quả như sau:

```
F:\Teaching\Dariu\IoTGatewayPython\venv\Scripts\python
Connected to Adafruit IO!
Kết nối thành công...
Subscribe thành công...
Subscribe thành công...
```

Hình 7.6: Thêm nút nhấn điều khiển trên Dashboard

Tùy vào việc bạn đọc đăng kí bao nhiêu kênh, sẽ có bấy nhiêu dòng thông báo được hiện ra như minh họa ở hình bên trên. Bây giờ, khi lần lượt nhấn vào 2 nút nhấn trên Dashboard, chúng ta sẽ nhận được 4 giá trị khác nhau, như kết quả sau đây:

```
Nhan du lieu: 1
Nhan du lieu: 0
Nhan du lieu: 2
Nhan du lieu: 3
|
```

Hình 7.7: Kết quả dữ liệu nhận được trên Gateway

Chương trình của Gateway được chia sẻ ở đường dẫn sau đây:

https://github.com/npnlab-vn/code-manager/blob/IoT_Lab7/IoT_Lab.py

Các lệnh điều kiện được bổ sung vào để trong trường hợp không kết nối với mạch Microbit, chương trình sẽ không bị lỗi. Mã nguồn của toàn bộ chương trình được trình bày bên dưới:

```
1 import serial.tools.list_ports
2 import random
3 import time
4 import sys
5 from Adafruit_IO import MQTTClient
6
7 AIO_FEED_ID = ["bbc-led", "bbc-pump"]
```

```

9
10 AIO_USERNAME = "NPNLab_BBC"
11 AIO_KEY = "aio_radR12aVJMa12YJiGBs1V6TBR061"
12
13 def connected(client):
14     print("Ket noi thanh cong...")
15     for feed in AIO_FEED_ID:
16         client.subscribe(feed)
17
18 def subscribe(client , userdata , mid , granted_qos):
19     print("Subscribe thanh cong...")
20
21 def disconnected(client):
22     print("Ngat ket noi...")
23     sys.exit (1)
24
25 def message(client , feed_id , payload):
26     print("Nhan du lieu: " + payload)
27     if isMicrobitConnected:
28         ser.write((str(payload) + "#").encode())
29
30 client = MQTTClient(AIO_USERNAME , AIO_KEY)
31 client.on_connect = connected
32 client.on_disconnect = disconnected
33 client.on_message = message
34 client.on_subscribe = subscribe
35 client.connect()
36 client.loop_background()
37
38 def getPort():
39     ports = serial.tools.list_ports.comports()
40     N = len(ports)
41     commPort = "None"
42     for i in range(0, N):
43         port = ports[i]
44         strPort = str(port)
45         if "USB Serial Device" in strPort:
46             splitPort = strPort.split(" ")
47             commPort = (splitPort[0])
48     return commPort
49
50 isMicrobitConnected = False
51 if getPort() != "None":
52     ser = serial.Serial( port=getPort() , baudrate=115200)
53     isMicrobitConnected = True
54
55
56 def processData(data):
57     data = data.replace("!", "")

```

```

58     data = data.replace("#", "")
59     splitData = data.split(":")
60     print(splitData)
61     if splitData[1] == "TEMP":
62         client.publish("bbc-temp", splitData[2])
63
63 mess = ""
64 def readSerial():
65     bytesToRead = ser.inWaiting()
66     if (bytesToRead > 0):
67         global mess
68         mess = mess + ser.read(bytesToRead).decode("UTF-8")
69         while ("#" in mess) and ("!" in mess):
70             start = mess.find("!")
71             end = mess.find("#")
72             processData(mess[start:end + 1])
73             if (end == len(mess)):
74                 mess = ""
75             else:
76                 mess = mess[end+1:]
77
78
79 while True:
80     if isMicrobitConnected:
81         readSerial()
82
83     time.sleep(1)

```

Chương trình 7.3: Chương trình xử lý nhiều nút nhấn

6 Câu hỏi ôn tập

1. Để nhận dữ liệu từ 2 nút nhấn, Gateway cần phải?
 - A. Đăng ký dữ liệu của 1 trong 2 nút nhấn
 - B. Đăng ký dữ liệu của cả 2 nút nhấn
 - C. Gửi dữ liệu cho 2 nút nhấn đó
 - D. Không cần làm gì cả, server sẽ tự động gửi
2. Để điều khiển 2 nút nhấn độc lập, chúng ta cần?
 - A. Tạo 1 feed dữ liệu dùng chung
 - B. Tạo 2 feed dữ liệu riêng biệt
 - C. Tạo 4 giá trị khác nhau cho các nút nhấn
 - D. Tất cả đều đúng
3. Để phân biệt nút nào được nhấn, chúng ta sẽ dựa vào trường dữ liệu nào ở phía Gateway?
 - A. payload
 - B. feed_id
 - C. Kết hợp cả 2 thông tin trên
 - D. Tất cả đều sai
4. Trong trường hợp nào, việc phân biệt dữ liệu từ các nút nhấn có thể chỉ dựa vào payload?
 - A. Khi chúng trả về 4 giá trị khác nhau
 - B. Khi chúng có 2 feed dữ liệu khác nhau
 - C. Khi chúng cùng nằm trên 1 Dashboard
 - D. Tất cả đều có thể
5. Phát biểu nào sau đây là đúng cho mạch Microbit trung tâm?
 - A. Nhận lệnh từ Gateway và gửi không dây tới nốt cảm biến
 - B. Nhận dữ liệu không dây cảm biến và gửi cho Gateway
 - C. Cả 2 tính năng trên
 - D. Chỉ 1 trong 2 tính năng trên
6. Câu lệnh nào dùng để chuyển chuỗi thành số trên Makecode?
 - A. parse to number
 - B. string to number
 - C. parse to string
 - D. number to string
7. Câu lệnh nào được sử dụng trong khối on start để gửi được dữ liệu bằng khôi lệnh Radio?
 - A. radio send number
 - B. radio send string
 - C. radio set group
 - D. Tất cả các câu lệnh trên

Đáp án

1. B 2. B 3. C 4. A 5. C 6. A 7. C

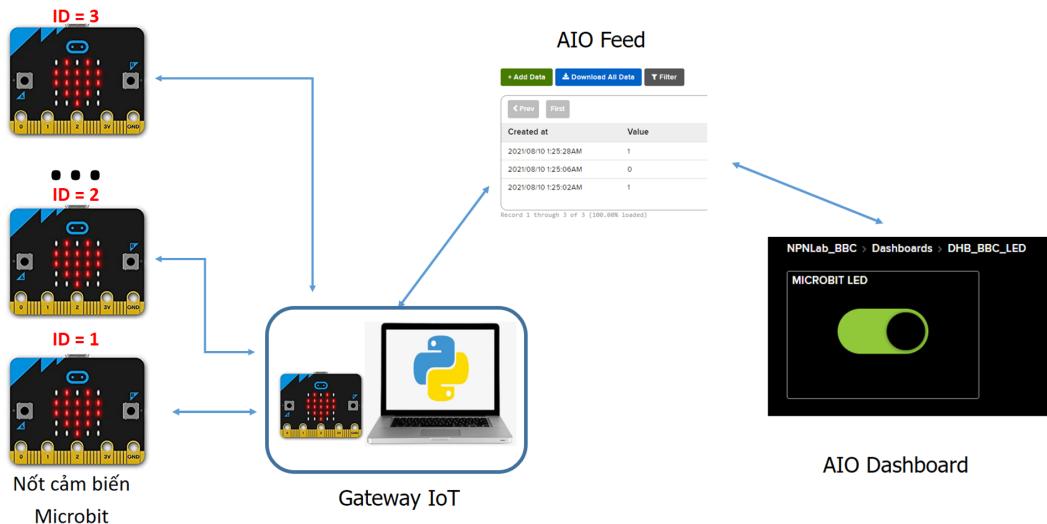
CHƯƠNG 8

Điều khiển ngoại vi trên nốt cảm biến



1 Giới thiệu

Trong bài này, chúng ta sẽ hiện thực phần cuối cùng của một ứng dụng kết nối vạn vật, các nốt cảm biến. Chúng ta sẽ sử dụng mạch Microbit, với nhiều tính năng đang được tích hợp sẵn như cảm biến nhiệt độ, ánh sáng và giao tiếp không dây, để hiện thực vai trò của một nốt cảm biến.



Hình 8.1: Nhiều nốt cảm biến trong hệ thống

Khi làm việc đến nốt cảm biến, chúng ta sẽ chạm tới phần khó khăn nhất trong toàn hệ thống. Mặc dù các công nghệ hiện thực trên nốt cảm biến không có gì mới, khó khăn của nó đến từ việc chúng ta có thể có nhiều, thậm chí là rất nhiều nốt cảm biến. Mỗi nốt cảm biến sẽ đảm nhiệm quan trắc 1 thông tin, hoặc cùng quan trắc một thông tin. Thêm nữa, sẽ có một số nốt cảm biến nhận vai trò điều khiển một hoặc một vài thiết bị nào đó. Như vậy, rõ ràng, thế giới cảm biến này là rất phong phú và đa dạng.

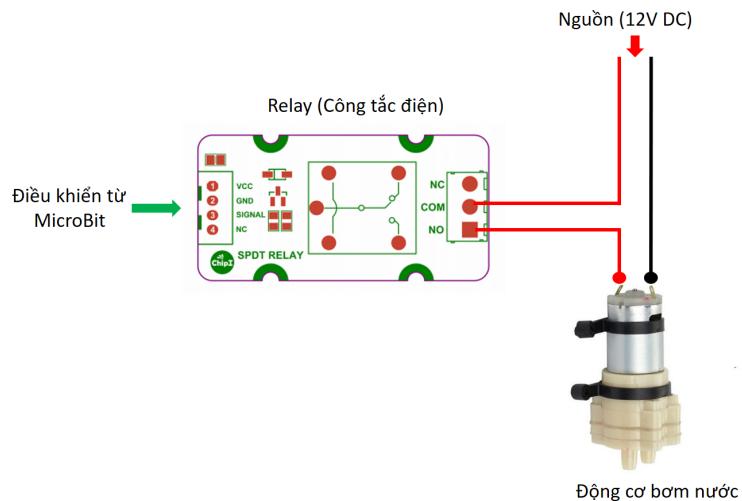
Điều quan trọng đối với các mạch cảm biến, là phải định nghĩa dữ liệu mà nó sẽ truyền đến Gateway, để phân biệt dữ liệu đó tới từ nốt nào, và dữ liệu đó mang theo thông tin gì (nhiệt độ hay độ ẩm chẳng hạn). Vấn đề này đã được trình bày ở bài trước, và bạn đọc sẽ thấy rõ hơn ở bài này, với nhiều nốt cảm biến cùng tương tác với Gateway.

Trong bài này, chúng ta sẽ hiện thực việc một nốt cảm biến nhận lệnh từ Gateway trung tâm. Cụ thể, 4 lệnh cơ bản sẽ được gửi xuống 1 nốt cảm biến để bật tắt đèn và bật tắt một máy bơm. Các mục tiêu chính của bài này như sau:

- Kết nối mạch điện: Relay điện tử, Máy bơm và đèn LED
- Nhận dữ liệu không dây ở nốt cảm biến
- Điều khiển ngoại vi tại nốt cảm biến

2 Kết nối mạch điện cho nút cảm biến

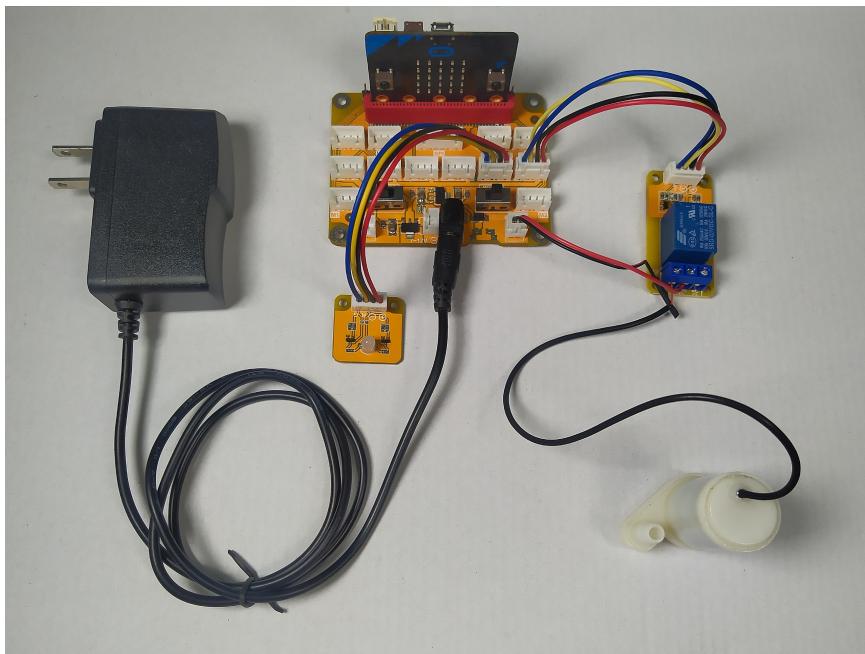
Đầu tiên, để máy bơm có thể hoạt động được, chúng ta cần phải sử dụng nguồn rời (chẳng hạn như PIN hoặc adapter) là vì nó tiêu thụ dòng lớn. Do đó, nguồn từ cổng USB của máy tính, cấp qua mạch MicroBit rồi cấp qua máy bơm là không đủ để nó vận hành. Module dùng để điều khiển động cơ bơm nước gọi là Relay, đóng vai trò như một công tắc điện tử, với 2 chân công tắc là COM và NO (Common và Normal Open). Công tắc này bình thường sẽ mở, với 2 chân COM và NO không kết nối.



Hình 8.2: Kết nối Relay và Máy bơm

Do đó, trong kết nối trên, chúng ta đã làm hở dây nguồn (cực dương) từ PIN, còn cực âm của PIN thì nối vào 1 cực của động cơ (động cơ bơm nước không có chiều). Nếu mắc mạch theo kiểu hở cực âm cũng không có ảnh hưởng gì đến việc hiện thực hệ thống. **Nếu như không có PIN rời, bạn có thể tận dụng khe cắm nguồn 2 chân trên mạch mở rộng ChiPi Base Shield.** Hai chân này cung cấp điện áp bằng với điện áp cấp vào từ adapter bên ngoài.

Cuối cùng, với 4 chân điều khiển của Relay, chúng ta sẽ nối nó vào cổng điều khiển VCC-GND-P0-P1 trên mạch mở rộng trên mạch ChiPi Base Shield. Thiết bị đèn hiển thị, sẽ được nối vào cổng kế bên, VCC-GND-P1-P2. Kết nối mạch điện của toàn hệ thống lúc này sẽ như sau:

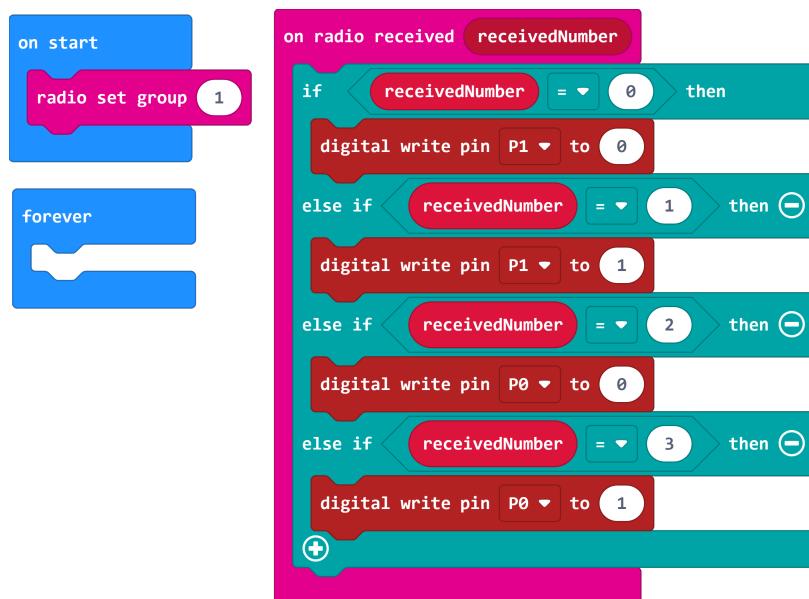


Hình 8.3: Kết nối mạch điện cho hệ thống

Như vậy, để điều khiển động cơ, chúng ta sẽ lập trình ở chân **P0**, và với đèn là 2 chân **P1, P2**.

3 Lập trình cho nốt cảm biến

Đầu tiên, để nhận dữ liệu từ Radio, nốt cảm biến phải có cùng nhóm với nốt trung tâm, trong trường hợp này sẽ là 1. Tiếp theo đó, việc nhận dữ liệu sẽ được thực hiện trong khối **on radio received receivedNnumber**. Việc lựa chọn khối nhận dữ liệu cần phải tương thích với việc gửi dữ liệu ở bài trước. Cấu trúc chương trình của chúng ta sẽ như sau:



Hình 8.4: Điều khiển đèn LED và Máy bơm

Việc điều khiển tắt mở một thiết bị, đối với việc lập trình trên Microbit là rất đơn giản. Chúng ta chỉ cần dùng một câu lệnh duy nhất là **digital write pin**, với 2 tùy chọn là 0 và 1 (tương ứng với Tắt và Bật). Bạn đọc cần phải xác định cho đúng thiết bị đó đang được kết nối với chân nào của Microbit.

Trong trường hợp bạn có 2 mạch Microbit, mà mạch thứ nhất nối với đèn còn mạch thứ 2 thì nối với Relay + Máy bơm, chúng ta chỉ cần giảm bớt số lượng điều kiện cần xử lý cho câu lệnh **if**. Một lưu ý rất quan trọng, là bạn đọc không được phép xài cấu trúc **else** khi hiện thực việc nhận lệnh và thực thi từ Gateway. Điều này sẽ gây rất nhiều nhập nhằng khi hệ thống có nhiều nốt. Bạn đọc hãy chỉ hiện thực chính xác các câu lệnh điều kiện mà thôi.

Chương trình gợi ý ở trên, được chia sẻ ở đường dẫn sau đây:

https://makecode.microbit.org/_0zoE2PhgUigm

Trong bài này, tạm thời chúng ta chưa hiện thực chức năng trong khối **forever**. Trong những bài sau, tính năng gửi giá trị quan trắc định kì sẽ được hiện thực trong **forever**.

4 Câu hỏi ôn tập

1. Thiết bị để bật/tắt máy bơm gọi là gì?
 - A. Công tắc điện tử
 - B. Relay
 - C. Rờ le
 - D. Tất cả đều đúng
2. Để bình thường máy bơm không hoạt động, 2 chân nào của Relay sẽ được kết nối?
 - A. COM và NO
 - B. COM và NC
 - C. NO và NC
 - D. Tất cả đều đúng
3. Để điều khiển bật/tắt một máy bơm, câu lệnh nào sau đây là phù hợp nhất?
 - A. digital write pin
 - B. set pull pin
 - C. Kết hợp cả 2 thông tin trên
 - D. Tất cả đều sai
4. Câu lệnh nào sau đây là đúng để nhận lệnh từ Microbit trung tâm?
 - A. seriel on data received
 - B. on radio received receivedNumber
 - C. on radio recevied receivedString
 - D. Tất cả đều có thể
5. Để điều khiển trạng thái của máy bơm, câu lệnh nào sau đây là phù hợp?
 - A. Một câu lệnh if else
 - B. Hai câu lệnh if
 - C. Một câu lệnh if else if
 - D. Có 2 câu đúng
6. Nguồn cấp cho động cơ bơm nước được lấy từ đâu là phù hợp?
 - A. Nguồn 3.3V từ Microbit
 - B. Nguồn 5V từ cổng USB của máy tính
 - C. Nguồn 5V từ adapter rời
 - D. Tất cả đều có thể sử dụng
7. Bao nhiêu chân của Microbit sẽ được sử dụng để điều khiển thiết bị ChiPi LED?
 - A. 1
 - B. 2
 - C. 3
 - D. 4

Đáp án

1. D 2. A 3. A 3. C 4. B 5. D 6. C 7. B

CHƯƠNG 9

Cảm biến quan trắc tích hợp



1 Giới thiệu

Một trong những ứng dụng mạnh mẽ mà nền tảng kết nối vạn vật mang lại, là các ứng dụng quan trắc môi trường. Nhờ khả năng kết nối không dây, nhiều nốt cảm biến có thể chia nhau quan trắc thông số môi trường và gửi về nốt trung tâm, để gửi lên server, phục vụ cho các ứng dụng thông minh, nâng tầm chất lượng cuộc sống, thúc đẩy các ứng dụng nghiên cứu lần triển khai, hướng đến các dịch vụ của thành phố thông minh trong tương lai.

Với các nốt cảm biến đã được thiết lập ở bài trước, trong bài này, chúng ta sẽ dùng nó để gửi thông tin về môi trường tới Gateway thông qua giao tiếp Radio. Vì vậy, nốt cảm biến, cho đến bài này mới thể hiện hết vai trò của nó, là phần đầu vào (còn gọi là **input**), thành phần vô cùng quan trọng của toàn hệ thống. Do đó, việc lựa chọn cảm biến cho một ứng dụng nói chung, cần phải được xem xét tỉ mỉ.



Hình 9.1: Cảm biến Nhiệt độ và độ ẩm DHT11

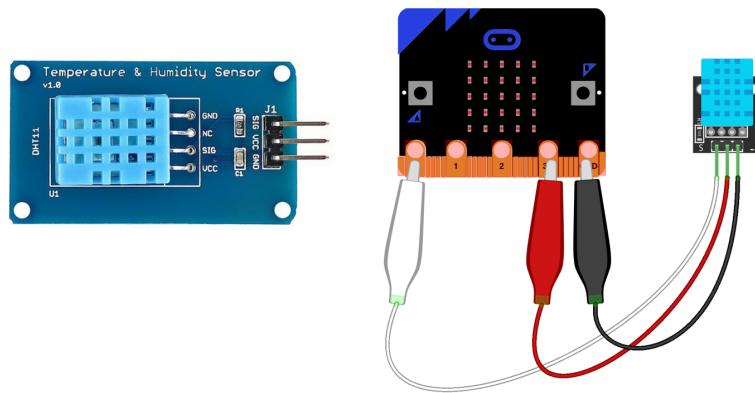
Trong bài hướng dẫn này, chúng tôi sẽ trình bày một cảm biến tích hợp, là cảm biến nhiệt độ - độ ẩm không khí DHT11, như trình bày ở hình trên. Sở dĩ gọi là tích hợp, bởi nó có thể cung cấp nhiều thông tin đồng thời, ở đây là 2 thông tin về môi trường. DHT11 là cảm biến khá đơn giản cho người mới bắt đầu, nhưng các phiên bản cao cấp của nó như DHT22 hay AM2305 là những sản phẩm có thể được dùng trong các ứng dụng thực tế. Và trên hết, các cảm biến này hoàn toàn tương thích về mặt chương trình, chỉ cần đổi thiết bị phần cứng, chúng ta sẽ có những thông tin về môi trường chính xác hơn và độ bền của thiết bị tốt hơn.

Các mục tiêu hướng dẫn trong bài này như sau:

- Kết nối với cảm biến DHT11
- Lập trình lấy dữ liệu từ DHT11
- Gửi dữ liệu lên Gateway
- Tìm hiểu các cảm biến cao cấp DHT22 và AM2305

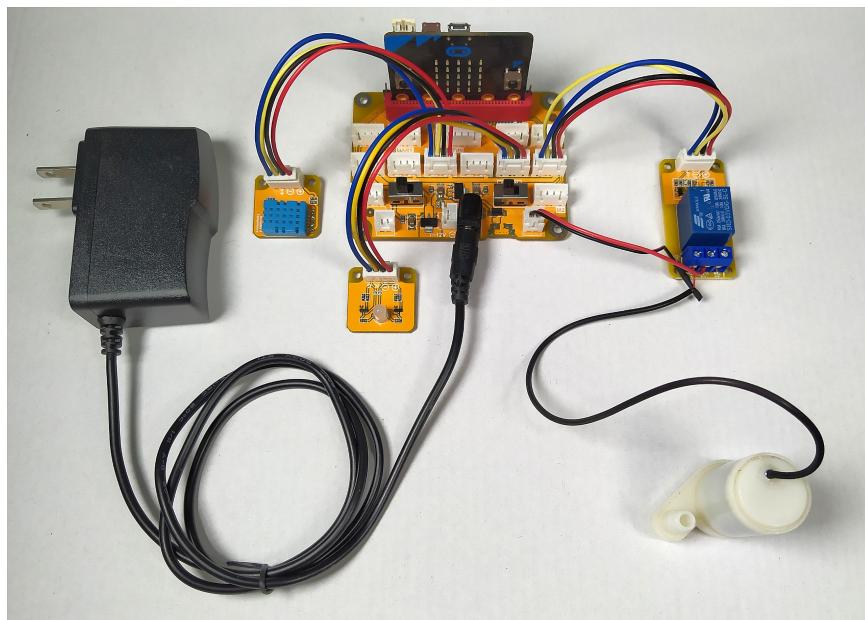
2 Kết nối với DHT11

Để kết nối với DHT11, chúng ta chỉ cần 3 chân kết nối, bao gồm Nguồn (từ 3V đến 5V), Đất (0V) và chân tín hiệu để lập trình. Rõ ràng, với việc tương thích điện áp rộng, cảm biến DHT11 được sử dụng nhiều trên các thiết bị lập trình phần cứng nói chung, và mạch Microbit nói riêng. Không khó để bạn đọc có thể kiểm các dự án liên quan giữa Microbit và cảm biến DHT11 trên mạng, kèm theo các hướng dẫn kết nối, như minh họa ở hình bên dưới:



Hình 9.2: Kết nối với DHT11 bằng 3 chân tín hiệu

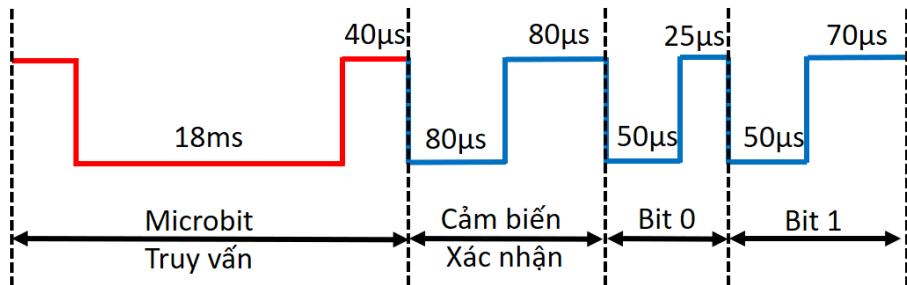
Trong bài hướng dẫn này, chúng tôi sử dụng thiết bị DHT11 đã được chuẩn hóa lại kết nối cho tương thích với mạch mở rộng ChiPi Base Shield. Chúng ta chỉ đơn giản là nối 1 dây có 4 chân vào mạch mở rộng. Khi kết nối vào mạch, bạn cần lưu ý chân tín hiệu của DHT11 đang kết nối với chân nào của Microbit. Thông tin này sẽ được dùng cho việc lập trình sắp tới. Tiếp tục tích hợp vào mạch cảm biến ở bài trước, chúng tôi sẽ kết nối cảm biến vào chân P3, như hình bên dưới.



Hình 9.3: Kết nối thêm cảm biến DHT11 vào nốt cảm biến

3 Nguyên lý hoạt động của DHT11

DHT11 là một dạng cảm biến tích hợp đơn giản và khá phổ biến với các ứng dụng dùng vi điều khiển nói chung, và Microbit nói riêng. Để có thể đọc dữ liệu từ nó, mạch Microbit phải gửi tín hiệu **truy vấn** (query). Khi nhận được tín hiệu này, cảm biến mới bắt đầu tính toán và gửi dữ liệu trả về cho mạch Microbit. Cũng chính vì lý do này, nếu như mạch Microbit quên gửi tín hiệu truy vấn, dữ liệu mà nó nhận được là dữ liệu cũ và không hợp lệ.



Hình 9.4: Nguyên lý giao tiếp với DHT11

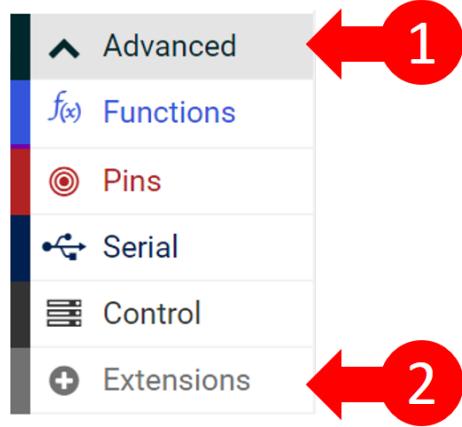
Mặc dù là cảm biến tích hợp đơn giản bậc nhất, việc lấy dữ liệu từ nó phức tạp hơn chúng ta nghĩ. Để bắt đầu việc giao tiếp, tín hiệu truy vấn từ Microbit gửi tới cảm biến là 1 xung mức thấp, kéo dài trong 18ms. Sau đó, Microbit sẽ nâng tín hiệu lên mức cao và chờ phản hồi từ DHT11. Sau khoảng 40 μ s, cảm biến sẽ xác nhận việc nhận lệnh bằng xung 80 μ s ở mức thấp, theo sau là xung 80 μ s nhưng ở mức cao. Cuối cùng, 40 bit dữ liệu sẽ được gửi lên Microbit, với bit 0 có hình xung là 50 μ s mức thấp và 25 μ s ở mức cao, còn bit 1 có hình xung là 50 μ s mức thấp và 70 μ s ở mức cao. Bạn đọc cần lưu ý về đơn vị thời gian cho quá trình giao tiếp này, được minh họa ở hình bên trên.

Cuối cùng, sau khi giải mã ra 40 bit dữ liệu, mạch Microbit phải tiếp tục xử lý để lấy ra thông tin cần thiết cho ứng dụng, với 16 bit đầu là thông tin cho độ ẩm, 16 bit tiếp theo là thông tin về nhiệt độ, và 8 bit cuối cùng là thông tin kiểm tra lỗi ($40 = 16 + 16 + 8$).

Vì tính chất phức tạp của cảm biến tín hiệu, chủ yếu là xử lý tín hiệu xung, đa số các ứng dụng trên Microbit sẽ sử dụng các thư viện lập trình hỗ trợ. Nhờ các thư viện này, việc lập trình sẽ đơn giản hơn rất nhiều và thuận tiện cho bạn đọc để tập trung xây dựng ứng dụng, hơn là việc can thiệp sâu vào hệ thống cho các tác vụ liên quan đến xử lý tín hiệu.

4 Lập trình với DHT11

Để hỗ trợ cho việc lập trình trong bài này, cũng như các bài còn lại trong toàn bộ giáo trình, thư viện IoTBitKit sẽ được thêm vào. Trình tự thêm thư viện này như sau: từ màn hình lập trình chọn vào **Advanced**, chọn tiếp **Extensions**, như hướng dẫn ở hình bên dưới:

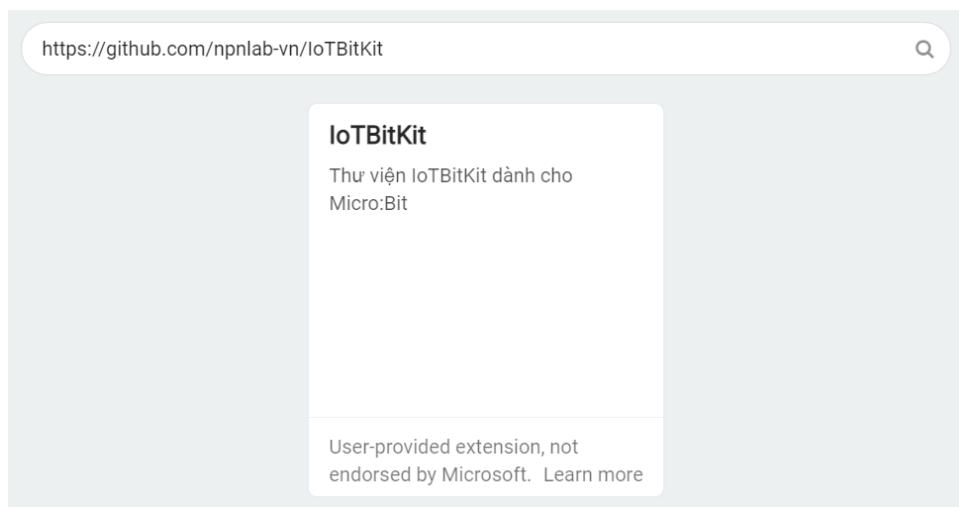


Hình 9.5: Thêm thư viện trên MakeCode

Với giao diện mới hiện ra, chúng ta nhập và đường dẫn sau đây:

<https://github.com/npnlab-vn/IoTBitKit>

Cuối cùng, nhấn vào nút tìm kiếm, chúng ta sẽ có kết quả như minh họa ở hình bên dưới:



Hình 9.6: Thêm thư viện IoTBitKit

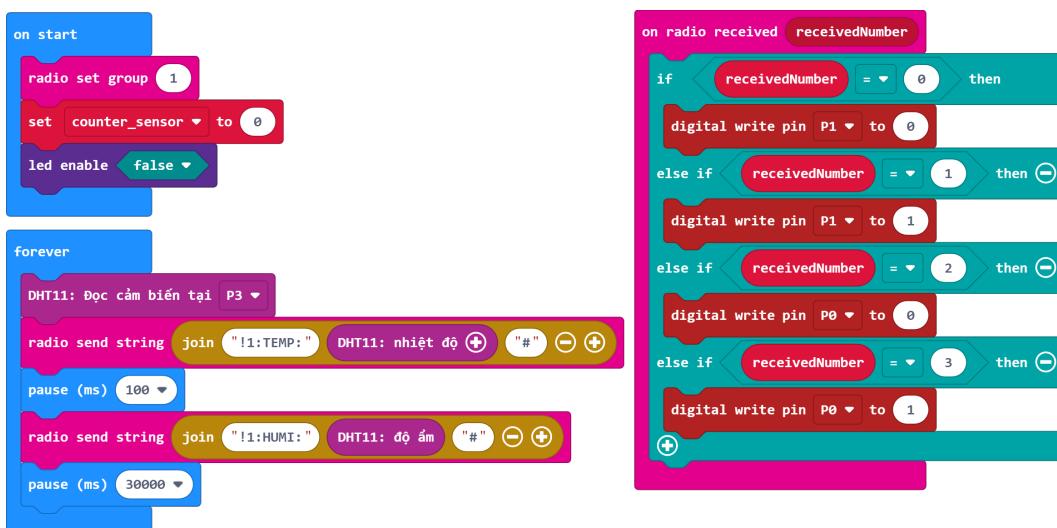
Rất nhiều nhóm lệnh của thư viện IoTBitKit sẽ được thêm vào hệ thống. Trong hướng dẫn này, các câu lệnh cần thiết cho việc lập trình lấy dữ liệu từ DHT11 được tích hợp sẵn trong nhóm lệnh NPNBitKit, như trình bày ở hình bên dưới:



Hình 9.7: Các câu lệnh liên quan đến DHT11 trong NPNBitKit

Hãy lưu ý rằng để nhận được giá trị cập nhật từ cảm biến, chúng ta sẽ phải truy vấn nó trước bằng câu lệnh **DHT11: Đọc cảm biến tại P3**. Trong trường hợp cảm biến được kết nối với chân khác, bạn đọc cần lựa chọn lại cho đúng chân kết nối. Sau khi truy vấn, chúng ta đã có thể sử dụng giá trị từ cảm biến này với 2 khối lập trình **DHT11: nhiệt độ** và **DHT11: độ ẩm**.

Sử dụng lại chương trình đã phát triển ở bài trước, chúng ta sẽ thêm tính năng đọc giá trị cảm biến mỗi 30 giây ở vòng lặp **forever**. Chương trình gợi ý sẽ như sau:



Hình 9.8: Chương trình đọc cảm biến DHT11

Các thông tin cảm biến sẽ được gửi đến nốt trung tâm theo định dạng mà chúng ta đã định nghĩa từ trước. Hai thông tin được gửi cách nhau một khoảng thời gian nhỏ, để giảm xác suất đụng độ của dữ liệu. Thêm nữa, để việc lập trình đơn giản, chúng tôi sử dụng câu lệnh đợi 30 giây ở cuối vòng lặp **forever**. Bạn đọc có thể chủ động xử lý để loại bỏ câu lệnh này, mà chỉ sử dụng một câu lệnh đợi **pause(100)** mà thôi. Với cách hiện thực thứ 2, hệ thống sẽ tốt hơn rất nhiều.

Bạn đọc cần lưu ý rằng, vì cảm biến DHT11 được nối vào chân P3, chúng ta phải thêm câu lệnh **led enable false** trong khối **on start**. Với câu lệnh này, 25 đèn của Microbit sẽ không còn sáng nữa. Nhưng bù lại, chúng ta có thể sử dụng các chân từ P3 của mạch Microbit. Chương trình được chia sẻ ở đường dẫn sau đây:

5 Cải thiện chương trình ở Gateway

Với 2 thông tin được gửi lên Gateway, bạn đọc có thể chủ động thêm một kênh dữ liệu nữa, để vẽ đồ thị cho tham số độ ẩm. Chương trình cho Gateway sẽ được cải tiến ở hàm **processData**, như sau:

```
1 def processData(data):
2     data = data.replace("!", "")
3     data = data.replace("#", "")
4     splitData = data.split(":")
5     print(splitData)
6     try:
7         if splitData[1] == "TEMP":
8             client.publish("bbc-temp", splitData[2])
9         elif splitData[1] == "HUMI":
10            client.publish("bbc-humi", splitData[2])
11     except:
12         pass
```

Chương trình 9.1: Cải thiện hàm xử lý dữ liệu

Khi có nhiều dữ liệu cùng gửi đến nốt trung tâm, việc nhận dữ liệu bị sai sót là có thể xảy ra. Do đó, câu lệnh **try except** được thêm vào để bỏ qua các lỗi do dữ liệu không hợp lệ. Đây là điều rất hữu ích cho các ứng dụng thực tế, khi chúng ta không thể lường trước được các lỗi có thể xảy ra khi hệ thống thực thi. Chương trình của Gateway được chia sẻ ở đường dẫn sau:

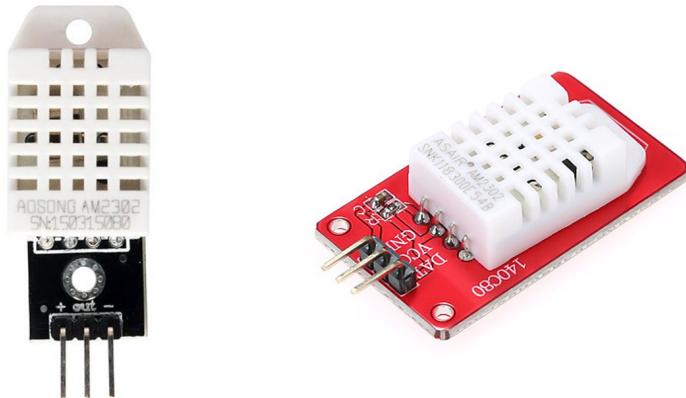
https://github.com/nplab-vn/code-manager/blob/IoT_Lab9/IoT_Lab.py

6 Các phiên bản nâng cấp của DHT11

Một trong những lợi thế của cảm biến DHT11 là nó có thể dễ dàng thay thế bằng các cảm biến cao cấp hơn, với độ chính xác tốt hơn, và hoàn toàn có thể sử dụng trong các ứng dụng công nghiệp, chẳng hạn như máy sấy trái cây hoặc máy ấp trứng. Trong phần cuối của bài này, chúng tôi sẽ trình bày thêm cho bạn đọc 2 loại cảm biến cao cấp hơn của DHT11. Các cảm biến này có cùng một kiểu giao tiếp. Chúng ta chỉ đơn giản là thay thế thiết bị phần cứng mà không cần phải lập trình lại.

6.1 Cảm biến DHT22

Đây là cảm biến có hình dạng rất giống với DHT11, nhưng thông thường nó sẽ có màu trắng để bạn đọc dễ phân biệt. Giá thành của nó thông thường cũng đắt hơn gấp đôi so với DHT11. Do đó, độ chính xác của nó hiển nhiên là tốt hơn DHT11, với chỉ khoảng 2% sai số khi đọc độ ẩm, so với 5% sai số của DHT11. Bên cạnh đó, nhiệt độ từ DHT22 có độ sai lệnh tối đa 0.5°C so với sai số 2°C của DHT11.

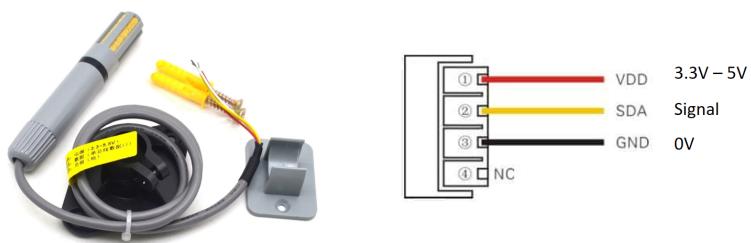


Hình 9.9: Cảm biến DHT22 trên thị trường

Do đó, đối với các ứng dụng mà việc đo nhiệt độ là thông tin quan trọng để vận hành hệ thống, chẳng hạn như máy áp trứng hoặc các máy sấy kích thước lớn, việc sử dụng DHT22 sẽ mang lại kết quả tốt hơn nhiều so với DHT11. Với kết nối chân hoàn toàn tương thích với DHT11, việc thay thế DHT22 vào hệ thống cũ sử dụng DHT11 là vô cùng dễ dàng và không có thay đổi nào về kết nối phần cứng lẫn chương trình trên phần mềm.

6.2 Cảm biến AM2305

Cảm biến nhiệt độ - độ ẩm này có đóng gói cứng cáp hơn và có độ bền tốt hơn so với DHT22. Thông thường, thiết bị này được sử dụng cho các sản phẩm trong công nghiệp đòi hỏi độ bền cao và chịu được nhiệt lớn. Hình ảnh của sản phẩm này được trình bày ở bên dưới.



Hình 9.10: Cảm biến AM2305 với 3 chân kết nối

Khi sử dụng sản phẩm này, bạn đọc cần lưu ý số lượng chân kết nối của cảm biến. Phiên bản nâng cấp của AM2305 là AM2315, có đến 4 chân kết nối, thay vì 3 chân như AM2305. Với chân nguồn tương thích với dãy điện áp rộng, từ 3.3V đến 5V, nên rất thích hợp với mạch Microbit. Tuy nhiên, trong các ứng dụng lớn, nguồn 5V nên được sử dụng cho cảm biến để đảm bảo độ ổn định. Độ chính xác của cảm biến AM2305 là tốt hơn hẳn so với DHT11, khi chỉ sai số trong khoảng 0.1°C và 1% sai số của độ ẩm.

7 Câu hỏi ôn tập

1. Phát biểu nào sau đây là đúng về cảm biến DHT11?
 - A. Cảm biến tích hợp
 - B. Đo được nhiệt độ và độ ẩm không khí
 - C. Là một dạng thiết bị đầu vào (input)
 - D. Tất cả đều đúng
2. Trước khi truy xuất giá trị của cảm biến DHT11, chúng ta cần làm gì?
 - A. Kéo chân cảm biến lên cao
 - B. Kéo chân cảm biến xuống thấp
 - C. Truy vấn cảm biến
 - D. Tất cả đều đúng
3. Nhóm lệnh hỗ trợ cho việc lập trình với DHT11 và LCD giới thiệu trong bài là gì?
 - A. DHT11
 - B. LCD
 - C. NPNBitKit
 - D. Tất cả đều đúng
4. Số chân kết nối giữa DHT11 và mạch mở rộng Microbit là bao nhiêu?
 - A. 1
 - B. 2
 - C. 3
 - D. 4
5. DHT11 hỗ trợ kết nối với nguồn có điện áp bao nhiêu Volt?
 - A. 3.3V
 - B. 4.1V
 - C. 5V
 - D. Tất cả đều đúng
6. Cảm biến nào sau đây là tương thích với chuẩn cắm dây với DHT11?
 - A. DHT22
 - B. AM2305
 - C. AM2315
 - D. Hai cảm biến đầu tiên
7. Cảm biến nào sau đây có độ chính xác cao nhất?
 - A. DHT11
 - B. DHT22
 - C. AM2305
 - D. Không xác định được

Đáp án

1. D 2. C 3. C 4. C 5. D 6. D 7. C

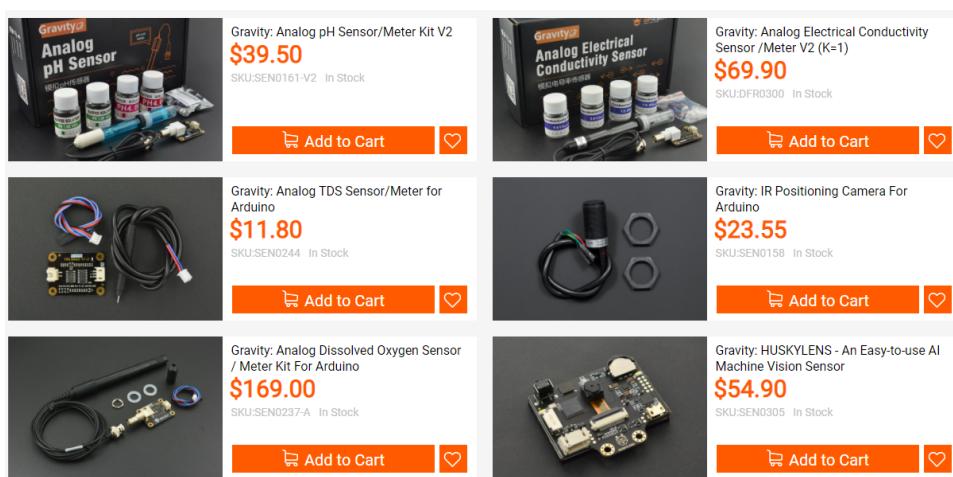
CHƯƠNG 10

Cảm biến tương tự Analog



1 Giới thiệu

Khác với cảm biến tích hợp giới thiệu ở bài trước, cảm biến tương tự, còn gọi là cảm biến Analog giới thiệu ở bài này, chỉ hỗ trợ đo một thông số duy nhất. Tuy nhiên, đầu ra của cảm biến là rất đơn giản, gọi là đầu ra tương tự, bạn đọc có thể dễ dàng đọc được dữ liệu từ cảm biến mà không cần sự hỗ trợ của các thư viện phức tạp. Việc đọc dữ liệu từ cảm biến tương tự còn được gọi là đọc ADC (Analog to Digital Converter). Và đúng như tên gọi ADC của nó, là chuyển từ tương tự sang số, giá trị mà chúng ta nhận được là một con số có giá trị từ 0 đến 1023, khi lập trình trên mạch Microbit.



Hình 10.1: Các cảm biến ADC từ DFRobot

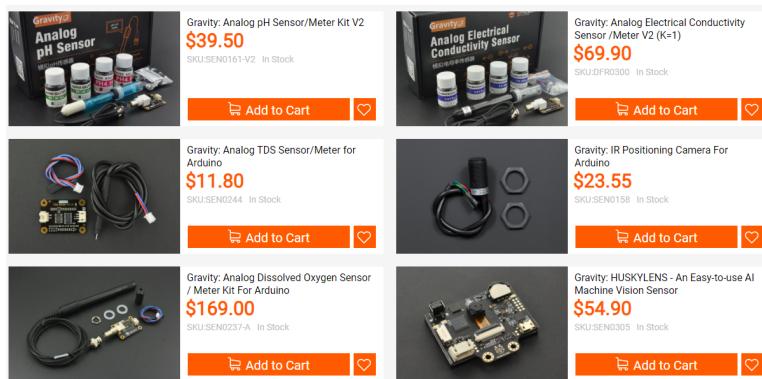
Do tính chất đơn giản của nó, các hệ thống cảm biến cho các dự án nói chung, và đặc thù cho mục đích giáo dục, đều có hỗ trợ chuẩn đầu ra là ADC. Thậm chí những cảm biến có độ phức tạp cao, vẫn có hỗ trợ chuẩn đầu ra là ADC, chẳng hạn như hệ thống cảm biến Analog Gravity từ DFRobot, với hàng trăm cảm biến hỗ trợ chuẩn đầu ra ADC. Một trong những yếu tố quan trọng của dạng cảm biến này, là bạn phải cung cấp đủ nguồn cho nó, bởi đơn giản, điện áp là thông tin quan trọng cho hoạt động của cảm biến.

Trong bài hướng dẫn này, chúng tôi sẽ giới thiệu về nguyên lý thiết kế của cảm biến ADC và minh họa trên cảm biến đo khí gas. Chúng tôi cũng sẽ giới thiệu về hệ thống cảm biến ChiPi, một hệ thống cảm biến đơn giản, với một chuẩn kết nối đồng bộ và rất phù hợp cho giáo dục ở lứa tuổi cấp 2. **Việc tích hợp cảm biến này vào hệ thống có sẵn sẽ được để dành cho bạn đọc sáng tạo.** Các mục tiêu trong bài hướng dẫn này như sau:

- Hiểu nguyên lý thiết kế của cảm biến Analog.
- Kết nối được cảm biến khí Gas.
- Hiện thực chương trình đọc dữ liệu từ cảm biến.
- Tìm hiểu các cảm biến ADC của hệ thống ChiPi.

2 Nguyên lý thiết kế cảm biến Analog

Cảm biến Analog khi kết nối với các hệ thống vi điều khiển khá đơn giản, bởi độ phức tạp của nó đã dồn qua việc thiết kế mạch. Thêm nữa, việc lập trình để đọc dữ liệu từ cảm biến cũng vô cùng thuận tiện. Các cảm biến loại này đều dựa vào đặc điểm là khi điều kiện giám sát thay đổi, điện trở của nó sẽ thay đổi theo. Ví dụ, cảm biến đo nhiệt độ sử dụng linh kiện, mà điện trở của nó sẽ thay đổi khi nhiệt độ thay đổi. Linh kiện này còn gọi là **nhiệt điện trở**. Tương tự như vậy, cảm biến ánh sáng sẽ sử dụng **quang trở**, thiết bị có điện trở thay đổi khi cường độ ánh sáng thay đổi. Nguyên lý kết nối các cảm biến này được trình bày như hình bên dưới:



Hình 10.2: Nguyên lý kết nối cảm biến ADC

Đầu tiên, bạn hãy để ý cầu điện trở ở bên trái, gồm 2 điện trở R1 và R2, nối từ nguồn xuống đất. Ở đây, điện trở R1 tượng trưng cho thiết bị cảm biến. Giá trị của R1 sẽ thay đổi tùy thuộc vào điện kiện của môi trường, trong khi đó, R2 sẽ có giá trị cố định. Dựa vào định luật Ohm, điện áp tại chân SIG1 sẽ được tính như sau:

$$SIG1 = \frac{VCC * R2}{R1 + R2}$$

Rõ ràng, khi R1 thay đổi, SIG1 sẽ thay đổi theo. Ví dụ như cảm biến độ ẩm đất ở bài trước, khi đất khô trở kháng R1 sẽ có giá trị lớn. Do giá trị R1 đang ở mấu số, giá trị điện áp ở chân SIG1 sẽ giảm. Ngược lại, khi đất ẩm, trở kháng R1 nhỏ, làm cho điện áp tại chân SIG1 tăng cao. Hành vi của hệ thống sẽ đảo ngược khi chúng ta đổi vị trí của R1 và R2 trên mạch điện trên.

Dựa vào nguyên lý này, rất nhiều thiết bị trong hệ thống ChiPi được thiế kế như trên. Do đó, đầu ra của cảm biến cũng chỉ cần 3 chân là VCC, GND và SIG. Tuy nhiên, trong hệ thống ChiPi, một mạch khuếch đại thuật toán OPAM được tích hợp thêm ở phía bên phải để tín hiệu đầu ra SIG2 được ổn định hơn. Mạch này được gọi là mạch hồi tiếp âm, do đầu ra được nối ngược lại vào chân âm của OPAM. Cách mắc này đảm bảo tín hiệu điện áp SIG2 và SIG1 là như nhau, nhưng với khả năng cách ly của OPAM, sẽ không có dòng điện đi vào mạch Microbit. Do đó, mạch Microbit sẽ rất bền và không bị nóng trong quá trình hoạt động. Trong trường hợp không có OPAM, sẽ có 1 dòng nhỏ đi từ SIG1 vào chân Microbit. Và trong trường hợp xấu, khi SIG1 tăng cao, do nhiều hoặc do thiết bị cảm biến bị lỗi, một dòng điện cao có thể làm hư mạch Microbit.

Với các cảm biến có đầu ra là điện áp, việc lập trình trên Microbit từ phía người sử dụng là vô cùng đơn giản, với câu lệnh **analog read**. Kết quả của phép đọc này chỉ là một con số, có tầm giá trị từ 0 đến 1023 (gọi là ADC 10 bit). Kết quả này không có đơn vị. Do đó, để chuyển nó sang đơn vị % của độ ẩm đất, hay **lux** của cường độ ánh sáng, thậm chí là **ppm** của nồng độ CO₂, chúng ta cần phải xây dựng thêm một hàm chuyển đổi nữa. Việc xây dựng hàm này sẽ tốn kém bởi chúng ta cần một thiết bị thương mại để có thể ánh xạ từ giá trị điện áp sang thông tin cần thiết. Trong phạm vi của giáo trình này, cũng như tính chất ứng dụng hiện tại còn đơn giản, chỉ cần phân biệt được có/không hoặc vượt ngưỡng, nên chúng tôi không đi vào chi tiết của hàm ánh xạ này.

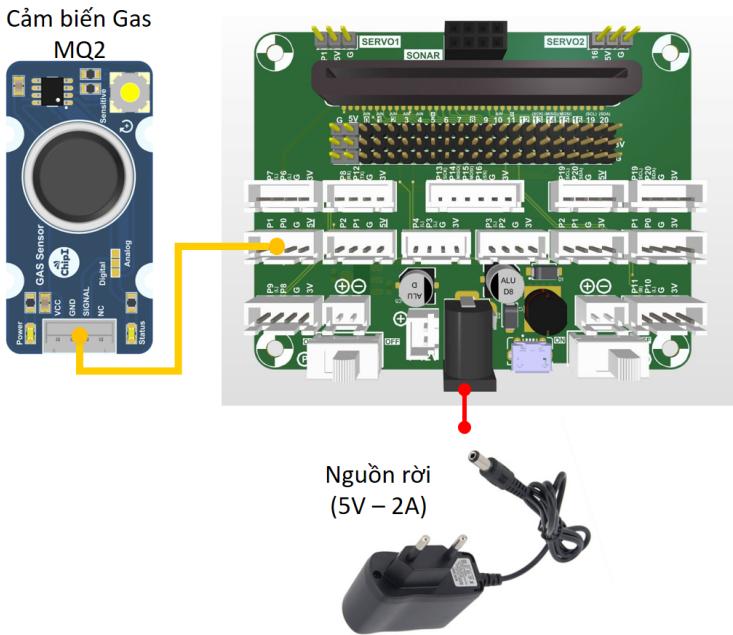
3 Cảm biến khí Gas

Cảm biến khí gas sử dụng trong bài hướng dẫn này có tên là MQ2, thuộc họ MQx, một công nghệ cảm biến khí đơn giản. Cảm biến loại này hoạt động dựa vào nguyên lý điện hóa. Chính vì vậy, nó cần một dòng điện có công suất cao, thì mới có thể phân tách được khí cần đo và trả về kết quả cho chúng ta. Cũng vì lý do này, mà MQ2 cần nguồn ngoài 5V-1A trở lên, và khi hoạt động, nó sẽ hơi ấm.

Trên thị trường, có rất nhiều cảm biến không khí được hiện thực bằng công nghệ điện hóa như MQ2, có thể tóm tắt ra một số thiết bị thông dụng như sau:

- MQ3: Alcohol, Ethanol, khói thuốc
- MQ4: Khí Methane
- MQ7: Carbon Monoxide CO
- MQ8: Khí hydro
- MQ9: Khí CO và khí độc hại

Do đó, hướng dẫn trong bài này có thể dễ dàng mở rộng ra nhiều ứng dụng khác nhau khi thay thế cảm biến MQ2 thành các cảm biến khác cùng họ MQx. Tuy nhiên, điều quan trọng là cảm biến này phải được cấp nguồn điện 5V, nên bạn đọc phải hết sức lưu ý thông tin này khi làm việc trên mạch Microbit và mạch mở rộng của nó. Trong hướng dẫn này, chúng tôi sử dụng mạch mở rộng ChiPi Based Shield V2, với sự hỗ trợ của 2 khe cắm 5V. Đường nhiên, để có điện áp 5V này, một nguồn ngoài (còn gọi là adapter) sẽ được sử dụng. Lúc này, cảm biến MQ2 kết nối với mạch mở rộng thông qua một dây 4 chân, như minh họa ở hình bên dưới.



Hình 10.3: Kết nối với cảm biến khí Gas MQ2

Theo kết nối ở trên, cảm biến khí Gas đang được kết nối với chân P0 của mạch Microbit. Để nhận ra vị trí chân kết nối bạn đọc cần chú ý màu dây đang nối với chân SIGNAL của cảm biến và mạch mở rộng ChiPi Base Shield V2.

4 Đọc dữ liệu từ cảm biến khí Gas

Việc lập trình lấy dữ liệu thô từ cảm biến ADC thực sự đơn giản, với một câu lệnh **analog read pin** trong mục **Pins** được sử dụng, chương trình gọi ý như sau:

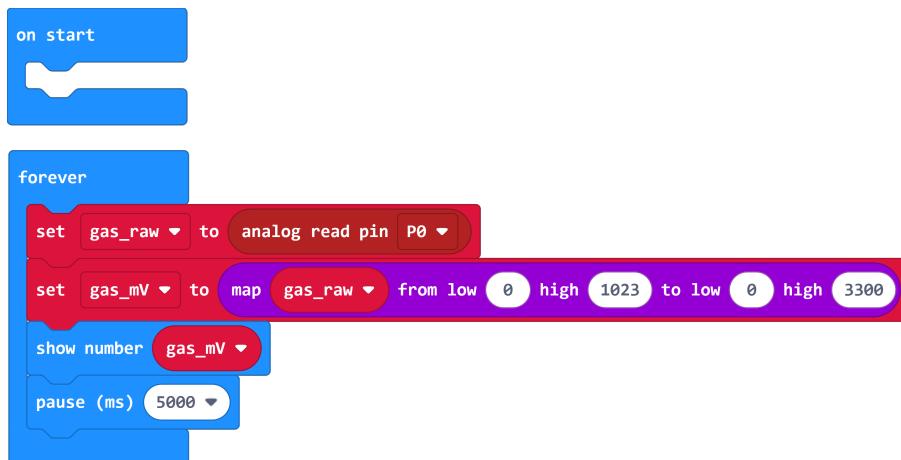


Hình 10.4: Đọc dữ liệu thô từ cảm biến ADC

Với các ứng dụng đơn giản, đoạn chương trình trên sẽ dùng để thống kê giá trị nhận được từ cảm biến. Bạn đọc cần chọn một ngưỡng so sánh để đưa ra các cảnh báo khi giá trị của cảm biến vượt ngưỡng cho phép. Chẳng hạn như với cảm biến khí Gas, một ngưỡng cần được chọn lựa kĩ càng, để đưa ra cảnh báo khi nồng độ khí Gas là nhiều hơn mức cho phép. Bạn đọc cũng có thể chủ động gửi dữ liệu này lên server ThingSpeak và kích hoạt tính năng gửi email cảnh báo, vốn đã hướng dẫn ở các bài trước, và sẽ không trình bày lại ở bài này.

Giá trị mà chúng ta nhận được từ mọi cảm biến ADC đều có giá trị từ 0 cho đến 1023, khi lập trình với Microbit. Nói một cách khác, bộ chuyển đổi ADC trên Microbit có 10 bit. Do đó, với điện áp đọc vào từ cảm biến, miền giá trị từ 0V đến 3.3V

sẽ được ánh xạ tuyến tính trên miền giá trị số 10 bit, từ 0 cho đến 1023 ($2^{10} - 1$). Do đó, trong trường hợp muốn tính ra chính xác giá trị của từng thông tin, chúng ta sẽ phải xem xét công thức chuyển đổi của từng thiết bị. Chúng tôi ví dụ rằng, **giá trị của khí Gas tính theo đơn vị ppm, là 2 lần của điện áp được tính ở đơn vị mili-Volt**. Với định nghĩ như thế, chúng ta cần đổi giá trị đọc được sang mV, bằng cách sử dụng khối lập trình **map** trong mục **Math**. Chương trình gợi ý cho việc chuyển đổi này như sau:



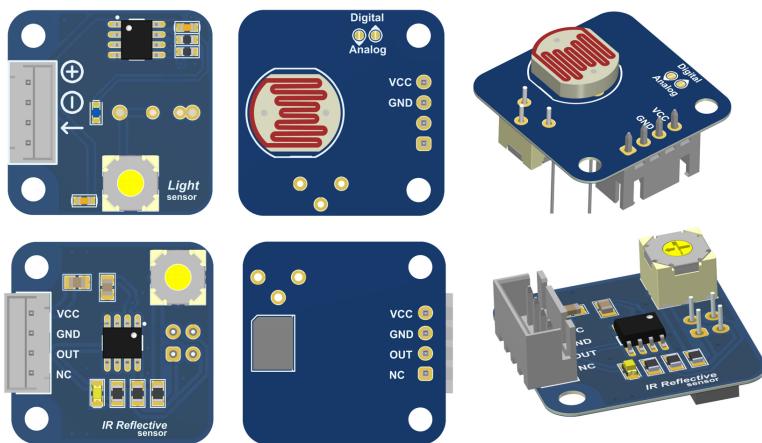
Hình 10.5: Chuyển đổi dữ liệu thô thành mV

Trong chương trình trên, 2 biến số đã được khai báo để lưu tạm các kết quả xử lý, bao gồm **gas_raw** để đọc dữ liệu thô và **gas_mV** là biến lưu giá trị sau khi đã chuyển sang mV. Do miền giá trị của **gas_raw** là từ 0 đến 1023, tương ứng với miền điện áp từ 0 đến 3300 mV (3.3V), nên các tham số của khối map được chỉnh lại như chương trình gợi ý ở trên.

Sau khi đã chuyển dữ liệu về đơn vị điện áp, bạn đọc sẽ phải sử dụng công thức chuyển đổi thêm 1 lần nữa, để tính ra đơn vị mình mong muốn. Công thức này sẽ phụ thuộc vào thiết bị cảm biến mà chúng ta sử dụng, nên sẽ không được hướng dẫn chi tiết ở bài này. Khi hiện thực công thức thứ 2 này, bạn đọc sẽ phải xài các câu lệnh tính toán trong nhóm Math. Phần này sẽ không được trình bày chi tiết trong hướng dẫn ở đây.

5 Cảm biến Analog ChiPi

Với mục đích tiện lợi cho bạn đọc mới bắt đầu với hệ thống cảm biến, vốn cần nhiều kinh nghiệm về điện tử, các cảm biến trong hệ thống ChiPi được thiết kế để đơn giản nhất cho bạn đọc với 1 kiểu kết nối duy nhất. Các khe cắm được thiết kế để bạn đọc không thể cắm nhầm. Tất cả những gì bạn đọc cần là xác định chân cảm cho chính xác, để lập trình lấy dữ liệu hợp lệ từ cảm biến. Một ví dụ về 2 thiết bị cảm biến khác, cũng ra dữ liệu dạng ADC, chẳng hạn như cảm biến cường độ ánh sáng và cảm biến vật cản hồng ngoại, như sau:



Hình 10.6: Một số cảm biến ADC trong chuẩn ChiPi

Đối với cảm biến vật cản hồng ngoại, khi vật thể ở gần, điện áp nhận được sẽ tăng (do ánh sáng phản xạ nhiều). Ngược lại, khi vật cản đi xa hoặc không có vật cản, điện áp sẽ giảm. Bên cạnh đó, còn có các cảm biến như độ ẩm đất, cảm biến cường độ âm thanh cũng thuộc nhóm này. Một số điểm lưu ý quan trọng khi kết nối với Microbit như sau:

- Cảm biến dạng analog chỉ hỗ trợ được cho các chân P0, P1, P2, P3, P4 và P10. Tức là chúng ta chỉ có thể kết nối được tối đa 6 cảm biến dạng analog vào một mạch Microbit. Trong trường hợp muốn kết nối nhiều hơn, bạn có thể sử dụng thêm 1 mạch Microbit khác và trao đổi dữ liệu giữa chúng thông qua giao tiếp không dây Radio.
- Trong trường hợp kết nối với các chân P3, P4 và P10, trong khôi **on start** cần sử dụng câu lệnh **led enable false** để lấy quyền điều khiển của 3 chân này. Khi đó, câu lệnh hiển thị ra màn hình của Microbit sẽ không sử dụng được nữa.
- Việc lựa chọn ngưỡng cần phải thực hiện tỉ mỉ. Khi không còn sử dụng được 25 đèn hiển thị trên Microbit (do câu lệnh **led enable false**), bạn có thể sử dụng như gửi dữ liệu lên máy tính hoặc tính năng Data Streamer của Excel.

Rất nhiều các cảm biến cao cấp, chẳng hạn như đo chất lượng nước và chất lượng không khí đều có hỗ trợ trên thị trường với chuẩn đầu ra là ADC, mà bạn đọc có thể tích hợp vào hệ thống sử dụng mạch Microbit. Khi đó, bạn đọc cần lưu ý điện áp nguồn cung cấp cho thiết bị để đảm bảo cảm biến hoạt động đúng chức năng.

Phần tích hợp cảm biến mới và gửi thông tin về nốt trung tâm, chúng tôi sẽ để cho bạn đọc tự phát triển. Bạn đọc cần định nghĩa một từ khóa khi gửi kèm giá trị của cảm biến đến nốt trung tâm, và xử lý nó ở hàm **processData** của Gateway. Không chỉ một, mà có thể nhiều nốt sẽ cùng gửi dữ liệu về nốt trung tâm. Trong trường hợp đó, bạn đọc cần định nghĩa thêm tham số **ID** cho nốt cảm biến, để nốt trung tâm có thể biết được nó nhận được giá trị này là từ nốt nào.

6 Câu hỏi ôn tập

1. Cảm biến CO₂ trong bài có đầu ra là dạng tín hiệu gì?
 - A. digital
 - B. analog
 - C. uart
 - D. Tất cả đều đúng
2. Điện áp đầu ra của cảm biến analog được dựa trên nguyên lý gì?
 - A. Cầu phân áp dùng 2 điện trở
 - B. Biến trở
 - C. Điện trở cố định
 - D. Tất cả đều đúng
3. Thiết kế cách ly đối với cảm biến analog, thiết bị nào thường được sử dụng?
 - A. Điện trở
 - B. Biến trở
 - C. OPAM
 - D. Tất cả đều đúng
4. Mạch Microbit hỗ trợ tối đa bao nhiêu chân cho kết nối với cảm biến analog?
 - A. 1
 - B. 2
 - C. 6
 - D. Tất cả đều sai
5. Câu lệnh để hỗ trợ cho việc chuyển đổi từ giá trị thô sang điện áp được hỗ trợ trên MakeCode là gì?
 - A. Math
 - B. map
 - C. linear
 - D. Tất cả đều sai
6. Khi kết nối cảm biến với chân P3, cần phải thực thi câu lệnh nào trong khối on start?
 - A. led enable true
 - B. led enable false
 - C. led turn on
 - D. led turn off
7. Khi tắt quyền hiển thị đèn trên mạch Microbit, các câu lệnh nào sẽ không còn tác dụng?
 - A. show number
 - B. show icon
 - C. show string
 - D. Tất cả các câu lệnh trên

Đáp án

1. B 2. A 3. C 4. B 5. B 6. B 7. D