

# YOLO:HOME

AI VÀ IOT CHO NHÀ THÔNG MINH





---

# Mục lục

---

<b>I Các Thao Tác Cơ Bản Trên Yolo:Bit</b>	<b>13</b>
<b>Chương 1. Chương trình đầu tiên trên Yolo:Bit</b>	<b>15</b>
1 Giới thiệu Yolo:Bit . . . . .	16
2 Cài đặt driver Yolo:Bit . . . . .	17
3 Môi trường lập trình . . . . .	19
4 Chương trình đầu tiên . . . . .	21
5 Khôi phục cài đặt gốc . . . . .	22
6 Chạy chương trình trên Yolo:Bit . . . . .	22
7 Nạp chương trình vào Yolo:Bit . . . . .	24
8 Lưu, mở và chia sẻ chương trình . . . . .	25
<b>Chương 2. Ngoại vi cơ bản trên Yolo:Bit</b>	<b>27</b>
1 Giới thiệu . . . . .	28
2 Hiển thị thông tin trên Yolo:Bit . . . . .	29
2.1 Hiện chữ - Hiện số . . . . .	29
2.2 Hiện hình ảnh . . . . .	29
2.3 Xóa màn hình . . . . .	31
3 Nút nhấn trên Yolo:Bit . . . . .	31
4 Cảm biến trên Yolo:Bit . . . . .	32
4.1 Nhiệt độ và Ánh sáng . . . . .	32
4.2 Cảm biến hành vi . . . . .	33
<b>Chương 3. Đồng hồ thông minh Internet</b>	<b>35</b>
1 Giới thiệu . . . . .	36
2 Thư viện lập trình cho thời gian . . . . .	36
3 Lập trình trên Yolo:Bit . . . . .	38
3.1 Lập trình cho nút A và B . . . . .	38
3.2 Khởi tạo đồng hồ Internet . . . . .	38
3.3 Hiển thị thông tin thời gian . . . . .	39
<b>II Kết Nối Mở Rộng Cho Yolo:Bit</b>	<b>43</b>
<b>Chương 4. Mạch Mở Rộng - Đèn RGB</b>	<b>45</b>
1 Giới thiệu . . . . .	46
2 Đèn chiếu sáng RGB . . . . .	47

3	Kết nối với Yolo:Bit . . . . .	47
4	Thêm thư viện lập trình AIOT . . . . .	48
5	Lập trình bật tắt đèn LED RGB . . . . .	49
6	Khe cắm trên mạch mở rộng . . . . .	50
<b>Chương 5. Cảm Biến Nhiệt Độ - Độ Ẩm DHT20</b>		<b>53</b>
1	Giới thiệu . . . . .	54
2	Kết nối với DHT20 . . . . .	54
3	Lập trình lấy dữ liệu từ DHT20 . . . . .	55
<b>Chương 6. Màn Hình LCD</b>		<b>59</b>
1	Giới thiệu . . . . .	60
2	Kết nối I2C với LCD . . . . .	60
3	Hiển thị dữ liệu trên LCD . . . . .	61
4	Cập nhật thông tin lên LCD . . . . .	63
5	Hiển thị nâng cao . . . . .	64
<b>Chương 7. Động cơ - Quạt mini</b>		<b>67</b>
1	Giới thiệu . . . . .	68
2	Nguyên lý điều khiển tốc độ cho động cơ . . . . .	68
3	Kết nối với Yolo:Bit . . . . .	69
4	Lập trình điều khiển quạt . . . . .	70
<b>Chương 8. Công Tắc Relay</b>		<b>73</b>
1	Giới thiệu . . . . .	74
2	Nguyên lý hoạt động của Relay . . . . .	75
3	Kết nối Relay với Yolo:Bit . . . . .	75
<b>Chương 9. Cảm biến chuyển động</b>		<b>79</b>
1	Giới thiệu . . . . .	80
2	Nguyên lý hoạt động của cảm biến PIR . . . . .	80
3	Kết nối với Yolo:Bit . . . . .	81
4	Lập trình với cảm biến chuyển động . . . . .	82
<b>Chương 10. Cảm Biến Ánh Sáng</b>		<b>85</b>
1	Giới thiệu . . . . .	86
2	Nguyên lý hoạt động của cảm biến trở kháng . . . . .	87
3	Kết nối với Yolo:Bit . . . . .	88
4	Lập trình đọc độ ẩm từ cảm biến . . . . .	88
4.1	Sử dụng thư viện AIOT-KIT . . . . .	88
4.2	Đọc điện áp trực tiếp . . . . .	89
4.3	Chuyển đổi giá trị cho cảm biến . . . . .	90
<b>Chương 11. Remote Và Mắt Nhận Hồng Ngoại</b>		<b>93</b>
1	Giới thiệu . . . . .	94
2	Kết nối với mắt nhận hồng ngoại . . . . .	94
3	Lập trình nhận dữ liệu từ remote . . . . .	95
4	Bật tắt đèn bằng remote . . . . .	96

<b>Chương 12. Động cơ RC Servo</b>	<b>99</b>
1    Giới thiệu . . . . .	100
2    Kết nối với động cơ . . . . .	101
3    Lập trình điều khiển động cơ . . . . .	101
<b>III    Kết Nối Vạn Vật với OhStem</b>	<b>105</b>
<b>Chương 13. Bảng Điều Khiển IoT</b>	<b>107</b>
1    Giới thiệu . . . . .	108
2    Chia sẻ dữ liệu trên OhStem server . . . . .	109
3    Thiết kế bảng điều khiển . . . . .	109
<b>Chương 14. Nhận dữ liệu trên Yolo:Bit</b>	<b>115</b>
1    Giới thiệu . . . . .	116
2    Thêm khôi mở rộng MQTT . . . . .	117
3    Lập trình trên Yolo:Bit . . . . .	118
3.1    Kết nối vào mạng WiFi . . . . .	119
3.2    Kết nối với OhStem server . . . . .	119
3.3    Đăng ký kênh dữ liệu . . . . .	120
3.4    Liên kết định kỳ với Server . . . . .	121
4    Xử lý dữ liệu nhận được từ server . . . . .	121
5    Mở rộng chương trình . . . . .	122
<b>Chương 15. Hiển thị thông tin IoT</b>	<b>125</b>
1    Giới thiệu . . . . .	126
2    Giao diện Thông tin và Đồ thị . . . . .	126
2.1    Cấu hình cho Thông tin . . . . .	127
2.2    Cấu hình cho Đồ thị . . . . .	128
3    Hoàn thiện bảng giám sát . . . . .	129
<b>Chương 16. Gửi dữ liệu lên OhStem server</b>	<b>131</b>
1    Giới thiệu . . . . .	132
2    Kết nối với OhStem server . . . . .	133
3    Thư viện lập trình Sự kiện . . . . .	134
4    Gửi dữ liệu lên server . . . . .	136
<b>Chương 17. Giao diện nâng cao</b>	<b>137</b>
1    Giới thiệu . . . . .	138
2    Tự kiểm thử giao diện . . . . .	138
3    Cấu hình cho Thanh trượt . . . . .	140
4    Giao diện Đồng hồ đo . . . . .	141
<b>Chương 18. Gửi tin nhắn với Telegram</b>	<b>145</b>
1    Giới thiệu . . . . .	146
2    Cài đặt Telegram . . . . .	147
3    Liên kết ứng dụng Telegram trên Website . . . . .	148
4    Tạo BotFather . . . . .	149
5    Tạo nhóm chat với BotFather . . . . .	152
6    Lập trình gửi tin nhắn trên Yolo:Bit . . . . .	154

## **IV Trí Tuệ Nhân Tạo** 157

<b>Chương 19. Nhận diện giọng nói</b>	<b>159</b>
1 Giới thiệu . . . . .	160
2 Kiến trúc ứng dụng AI . . . . .	160
3 Môi trường lập trình AI . . . . .	161
4 Lập trình nhận diện giọng nói . . . . .	162
<b>Chương 20. Nhận và xử lý lệnh AI</b>	<b>167</b>
1 Giới thiệu . . . . .	168
2 Xây dựng chương trình cho Yolo:Bit . . . . .	169
3 Lưu chương trình vào thiết bị . . . . .	172
4 Kết nối với trí tuệ nhân tạo . . . . .	173
<b>Chương 21. Huấn luyện trí tuệ nhân tạo</b>	<b>175</b>
1 Giới thiệu . . . . .	176
2 Trang chủ của trí tuệ nhân tạo . . . . .	177
3 Huấn luyện hệ thống . . . . .	177
4 Hiện thực dự án AI . . . . .	179
<b>Chương 22. Kết hợp công nghệ AI</b>	<b>183</b>
1 Giới thiệu . . . . .	184
2 Chuyển đổi giọng nói - văn bản . . . . .	184
2.1 Môi trường doanh nghiệp . . . . .	185
2.2 Nhà thông minh và Robot . . . . .	185
2.3 Trong giáo dục . . . . .	186
3 Học máy với Google . . . . .	186
4 Kết hợp hai công nghệ AI . . . . .	187
4.4 Nhận dạng giọng nói . . . . .	187
4.5 Nhận dạng hình ảnh . . . . .	189
<b>Chương 23. Khung chương trình cho AI và IoT</b>	<b>191</b>
1 Giới thiệu . . . . .	192
2 Thư viện cho ứng dụng AIoT . . . . .	192
3 Khung chương trình IoT . . . . .	193
4 Tích hợp tính năng AI . . . . .	195

## **V Hiện Thực Dự Án Yolo:Home** 197

<b>Chương 24. Tổng quan dự án</b>	<b>199</b>
1 Giới thiệu . . . . .	200
2 Giám sát môi trường . . . . .	201
3 Cửa mật mã . . . . .	201
4 Điều khiển thiết bị bằng giọng nói . . . . .	202
5 Nhận diện khuôn mặt - FaceAI . . . . .	203

<b>Chương 25. Khởi động dự án</b>	<b>205</b>
1    Giới thiệu . . . . .	206
2    Kết nối phần cứng . . . . .	206
3    Thiết kế bảng điều khiển IoT . . . . .	208
4    Khung chương trình AIoT . . . . .	209
<b>Chương 26. Giám sát chất lượng không khí</b>	<b>211</b>
1    Giới thiệu . . . . .	212
2    Hiển thị thông tin trên LCD . . . . .	212
3    Hiện thực chương trình . . . . .	213
4    Bật đèn tự động . . . . .	216
<b>Chương 27. Cửa mật mã</b>	<b>219</b>
1    Giới thiệu . . . . .	220
2    Kiểm tra mật mã . . . . .	220
3    Thay đổi mật mã . . . . .	223
<b>Chương 28. Nhà thông minh IoT</b>	<b>227</b>
1    Giới thiệu . . . . .	228
2    Gửi dữ liệu lên bảng điều khiển . . . . .	228
3    Nhận dữ liệu và điều khiển thiết bị . . . . .	229
<b>Chương 29. Điều khiển bằng giọng nói</b>	<b>233</b>
1    Giới thiệu . . . . .	234
2    Kiến trúc điều khiển AIoT . . . . .	235
3    Chương trình nhận diện giọng nói . . . . .	235
<b>Chương 30. Nhận dạng khuôn mặt FaceAI</b>	<b>239</b>
1    Giới thiệu . . . . .	240
2    Khối nhận diện AI và thời gian . . . . .	241
3    Hiện thực FaceAI . . . . .	241
4    Tích hợp xử lý trên Yolo:Bit . . . . .	245
5    Kết luận và các hướng mở rộng . . . . .	246

## LỜI MỞ ĐẦU

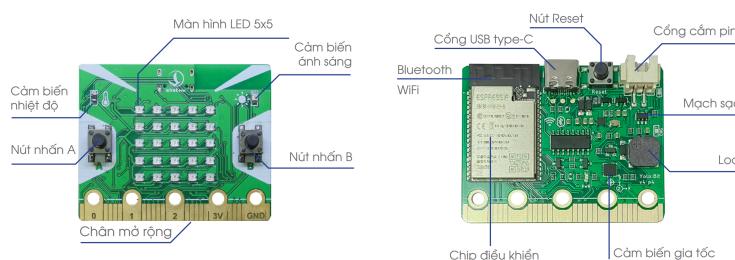
Cuộc cách mạng khoa học công nghệ lần thứ 4 (Industrial 4.0), với các thành tựu tiêu biểu như Kết nối vạn vật, Trí tuệ nhân tạo, Khoa học dữ liệu và bảo mật Blockchain, đã góp phần nâng cao chất lượng cho các ứng dụng tự động hóa, như nhà thông minh, nông nghiệp thông minh hay giáo dục thông minh. Nhờ sự chia sẻ dữ liệu mạnh mẽ trên nền tảng kết nối vạn vật, cộng với sự mềm dẻo của Trí tuệ nhân tạo, các ứng dụng hiện tại đang dần chuyển mình từ tự động (Automation) sang tự hành (Autonomous).

Trong giáo trình này, chúng tôi áp dụng các công nghệ nổi bật vào ứng dụng **Yolo-Home - nhà thông minh**, vốn là chủ đề khá phổ biến trong giáo dục STEM. Tuy nhiên, bên cạnh các tính năng tự động, khi kết hợp với Kết nối vạn vật và Trí tuệ nhân tạo, nhiều tính năng hấp dẫn sẽ được tích hợp vào dự án YoloHome. Bên cạnh việc nhận dạng và điều khiển thiết bị bằng giọng nói, chúng ta có thể theo dõi các thông tin về điều kiện trong nhà, như nhiệt độ và độ ẩm không khí.Thêm nữa, việc nhận dạng khuôn mặt có thể bổ sung thêm một tính năng nổi trội cho ứng dụng của thông minh trong chủ đề nhà thông minh.

Dự án trong giáo trình, tuy không phải là một sản phẩm thực tế, nhưng chúng tôi hy vọng nó sẽ cung cấp cho bạn đọc các kiến thức cơ bản về việc áp dụng công nghệ vào một ứng dụng cụ thể. Nội dung trong giáo trình này được chia làm nhiều phần và theo kinh nghiệm của chúng tôi, nó thích hợp để áp dụng cho học sinh ở bậc trung học.

Chi tiết của mỗi phần được tóm tắt như bên dưới:

### Phần 1: Các Thao Tác Cơ Bản Trên Yolo:Bit



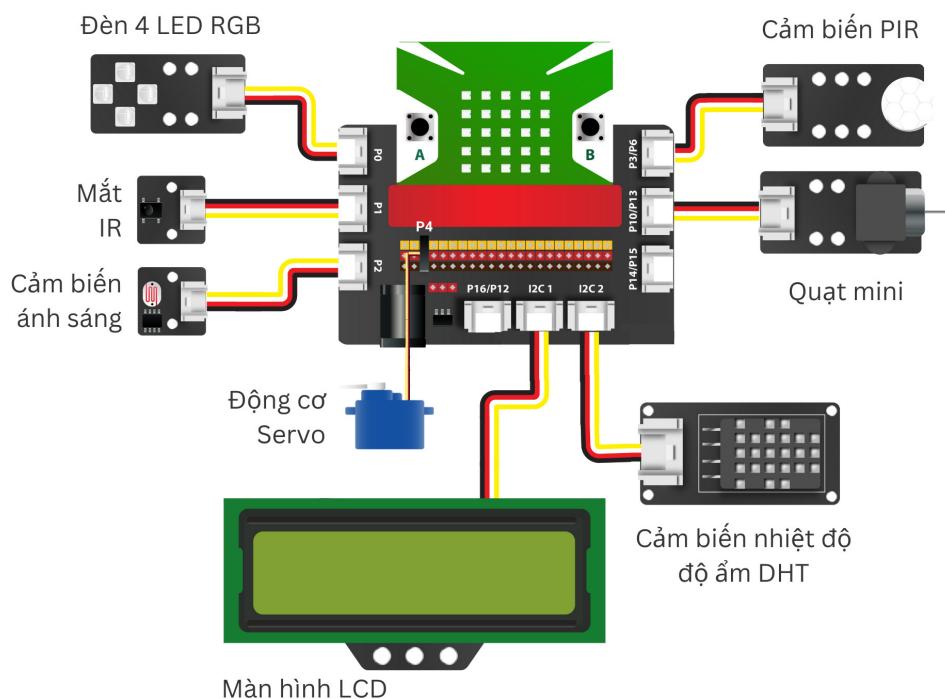
Hình 1: Mạch lập trình Yolo:Bit sử dụng trong giáo trình

Trong phần này sẽ trình bày các bước cơ bản để cài đặt phần mềm và làm quen với mạch lập trình Yolo:Bit. Với vai trò là bộ xử lý trung tâm trong dự án Yolo:Home,

bạn đọc sẽ làm quen với các ngoại vi cơ bản của nó, trước khi bắt đầu tích hợp thêm các ngoại vi phức tạp hơn cho dự án. Trong phần này, dự án đồng hồ thông minh sẽ được trình bày, giúp bạn đọc hiểu rõ hơn khả năng của mạch lập trình Yolo:Bit. Bằng khả năng kết nối vào mạng Internet, mạch Yolo:Bit dễ dàng có được thông tin chính xác về thời gian.

## Phần 2: Kết Nối Mở Rộng Cho Yolo:Bit

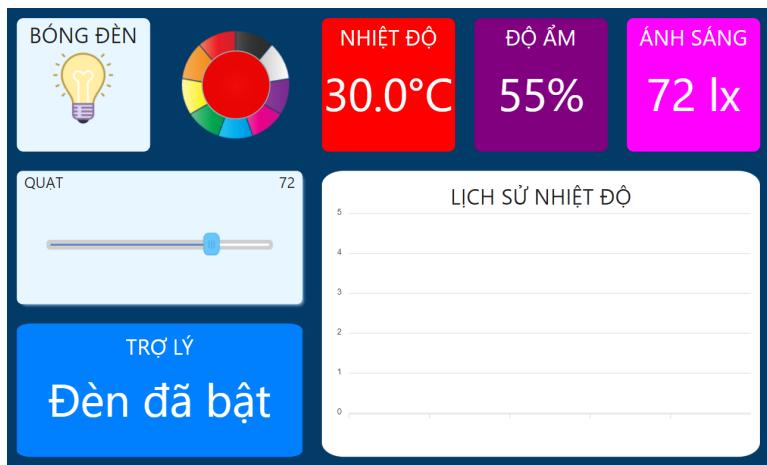
Tùy vào yêu cầu của mỗi dự án, các thiết bị ngoại vi khác nhau cần được kết nối vào mạch Yolo:Bit. Trong phần này, chúng tôi trình bày những kiến thức cơ bản nhất về việc kết nối các cảm biến (ánh sáng, độ ẩm và nhiệt độ không khí) và thiết bị điều khiển (quạt mini, đèn). Phần này tập trung trình bày chi tiết về đặc tính điện, cách kết nối cũng như lập trình với nhiều thiết bị khác nhau. Mặc dù tập trung cho các thiết bị cần thiết cho dự án về nhà thông minh, phần này có thể dễ dàng áp dụng cho các dự án khác trong tương lai.



Hình 2: Kết nối thêm ngoại vi cho Yolo:Bit

## Phần 3: Kết Nối Vạn Vật với OhStem

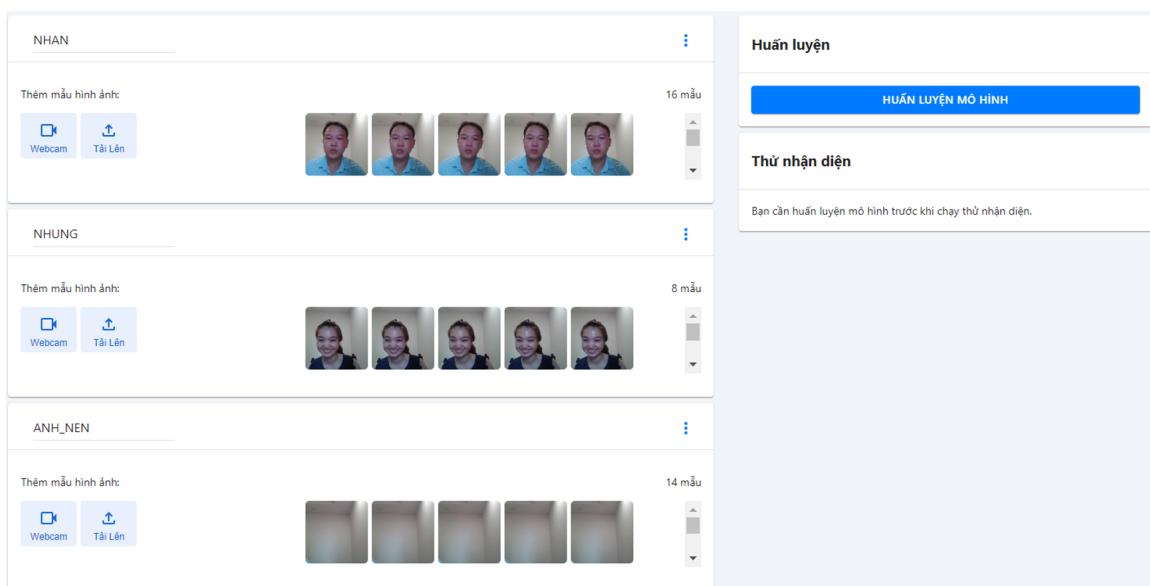
Phần tiếp theo này, là bước chuyển mình của công nghệ, từ thời kì tự động hóa sang kết nối vạn vật. Với sự hỗ trợ đặc biệt của nền tảng mạng thế hệ thứ 4, dữ liệu từ các cảm biến được dễ dàng chia sẻ lên mạng Internet. Trong chương này, chúng tôi giới thiệu server OhStem, được thiết kế đặc biệt cho các dự án liên quan đến kết nối vạn vật. Với sự hỗ trợ của những giao diện đẹp mắt và đặc biệt là không cần tạo tài khoản, bạn đọc có thể dễ dàng tạo ra một ứng dụng thu thập dữ liệu và điều khiển từ xa dựa trên nền tảng kết nối vạn vật.



Hình 3: OhStem - Server kết nối vạn vật

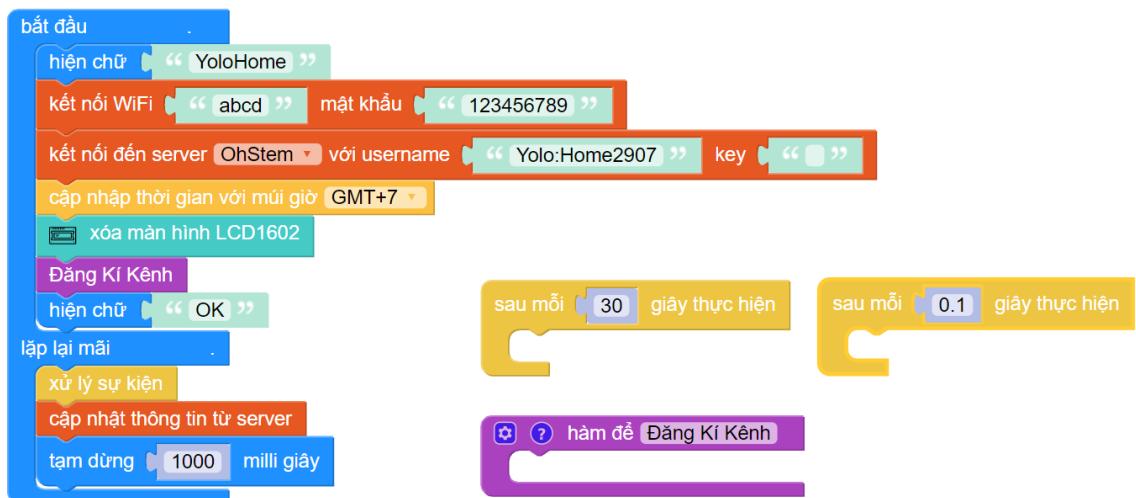
#### Phần 4: Trí Tuệ Nhân Tạo

Cùng với sự phát triển của nền tảng kết nối vạn vật, chúng ta có công nghệ nổi trội bậc nhất hiện tại, là trí tuệ nhân tạo. Sử dụng mã nguồn mở từ Google, chúng tôi tích hợp công nghệ này vào hệ sinh thái OhStem và cung cấp công cụ cho bạn đọc thu thập dữ liệu, huấn luyện hệ thống và sử dụng nó cho các ứng dụng của mình. Đối với dự án trong giáo trình này, chúng ta sẽ huấn luyện hệ thống để có thể nhận diện ra ai đang đứng trước Webcam. Chúng tôi định nghĩa tính năng này là FaceAI. Sự linh động của trí tuệ nhân tạo giúp nó dễ dàng ứng dụng vào nhiều dự án khác nhau mà nhà thông minh chỉ là một ứng dụng minh họa. Bạn đọc có thể áp dụng nó trong nhận dạng và phân loại rác hay thậm chí là phát hiện sâu bệnh ở cây trồng.



Hình 4: Huấn luyện trí tuệ nhân tạo cho chức năng FaceAI

## Phần 5: Hiện Thực Dự Án Yolo:Home



Hình 5: Tổ chức chương trình cho dự án

Khác với các phần trước, trong phần này, bên cạnh việc tích hợp các kiến thức cũ để hiện thực một dự án hoàn chỉnh, chúng tôi trình bày một phương pháp hiện đại để tiếp cận với kiến thức công nghệ, đó là học tập qua dự án (project based learning). Khả năng tổ chức và phát triển chương trình qua mỗi bài hướng dẫn là mục tiêu chính của phần này. Nhiều tính năng thông minh và gần gũi với thực tế được đưa vào dự án, như theo dõi các thông tin liên quan đến môi trường trong nhà, điều khiển thiết bị thông minh bằng giọng nói nhân tạo, khóa cửa mật mã và đặc biệt là tính năng FaceAI để nhận diện khuôn mặt.

Với việc chia giáo trình thành 5 phần riêng biệt, chúng tôi hy vọng có thể giúp các thầy cô phân bổ bài giảng hợp lý với học sinh khối trung học. Với học sinh lớp 6, các em chỉ nên tiếp cận với các thao tác cơ bản ở Phần 1. Đến lớp 7, khi các em đã có thêm kiến thức trong môn Vật lý, Phần 2 sẽ thích hợp để minh họa thêm các kiến thức về điện tử. Kết nối vạn vật và Trí tuệ nhân tạo (Phần 3 và 4) là vừa tầm với học sinh lớp 8. Cuối cùng, các hướng dẫn cho một dự án hoàn chỉnh là thích hợp với học sinh lớp 9.

Cuối cùng, chúng tôi hy vọng rằng, thông qua giáo trình này, bạn đọc sẽ hình thành cho mình một nền tảng chung cho nhiều ứng dụng trong tương lai.

**OhStem Education**  
**Khơi Nguồn Sáng Tạo**



## **Phần I**

# **Các Thao Tác Cơ Bản Trên Yolo:Bit**



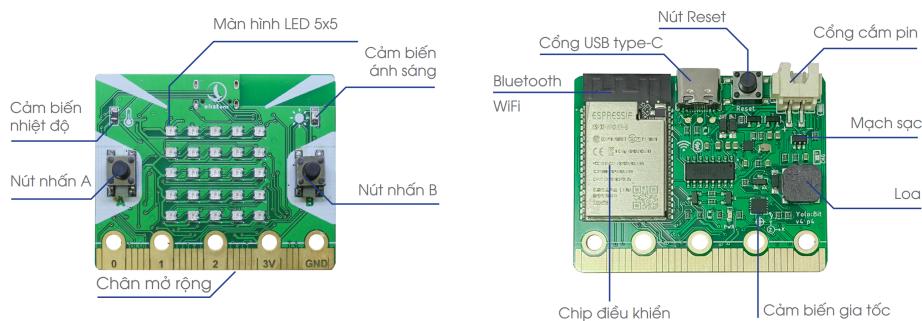
# CHƯƠNG 1

## Chương trình đầu tiên trên Yolo:Bit



# 1 Giới thiệu Yolo:Bit

Yolo:Bit là một mạch lập trình mini được thiết kế để phục vụ việc dạy và học lập trình ở mọi cấp độ, từ học sinh cấp 2, cấp 3, thậm chí là cao đẳng và đại học. Bản thân Yolo:Bit là một máy tính nhúng có hệ điều hành nhỏ (tên là Micro Python), được chạy trên nền tảng chip ESP32. Với việc lập trình trên mạch Yolo:Bit, chúng tôi muốn truyền tải đến bạn đọc khái niệm **lập trình ứng dụng**, vốn có nhiều khác biệt so với lập trình giải thuật - thường áp dụng trên máy tính đơn thuần.



Hình 1.1: Mạch lập trình Yolo:Bit

Máy tính lập trình Yolo:Bit có lối thiết kế đặc biệt, phù hợp để dạy lập trình cho trẻ nhỏ. Hình dạng và mạch điện được thiết kế dựa theo chuẩn giáo dục của các nước tiên tiến, giúp đảm bảo an toàn cho các em. Mạch Yolo:Bit sử dụng điện áp thấp 3.3V, các góc cạnh được bo tròn, giúp hạn chế tối đa các nguy cơ ảnh hưởng tới trẻ nhỏ khi làm việc với các mạch điện tử.

Bên cạnh nhiều ngoại vi và cảm biến tích hợp, Yolo:Bit còn tích hợp sẵn khả năng kết nối không dây (WiFi và Bluetooth), nhằm hỗ trợ cho các ứng dụng kết nối vạn vật - Internet of Things - một công nghệ rất phổ biến trong các ứng dụng ngày nay.



Hình 1.2: Điện toán đám mây cho kết nối vạn vật

Các mạch như Yolo:Bit có thể chia sẻ dữ liệu giữa nhiều thiết bị tham gia vào mạng và lưu trữ dữ liệu này lên các máy chủ, điện toán đám mây - nơi mà các thiết bị khác có thể nhìn thấy và sử dụng. Đây giống như là một kho kiến thức bổ sung cho các công nghệ về trí tuệ nhân tạo trong tương lai, như minh họa ở hình bên trên.

Trước khi đi vào các ứng dụng nâng cao với Yolo:Bit, chúng ta cần phải thiết lập việc kết nối và lập trình với mạch này. Nội dung chính trong bài đầu tiên bao gồm:

- Cài đặt driver cho mạch Yolo:Bit
- Môi trường lập trình trực tuyến
- Chương trình đầu tiên trên Yolo:Bit
- Chạy thử và nạp chương trình

## 2 Cài đặt driver Yolo:Bit

Với sự hỗ trợ của chip tích hợp ESP32, có nhiều cách để chúng ta kết nối và lập trình cho mạch Yolo:Bit. Tuy nhiên, trong giáo trình này, chúng tôi ưu tiên kết nối máy tính với mạch Yolo:Bit thông qua dây USB, để việc kết nối và lập trình được ổn định. Do đó, chúng ta cần phải **cài đặt driver** cho mạch Yolo:Bit (bỏ qua bước này nếu bạn đã kết nối Yolo:Bit với máy tính bằng cáp USB trước đó). Các bước chi tiết được trình bày như bên dưới:

### Bước 1: Tải file cài đặt driver cho mạch Yolo:Bit

Mạch Yolo:Bit sử dụng driver USB có tên gọi là CH340. Đây là driver rất phổ biến cho các hệ thống Arduino. Do đó, nếu máy tính nào đã từng lập trình được với bo mạch Arduino, thường đã có sẵn driver này và bạn không cần phải cài đặt lại. Bạn có thể tự kiểm tra điều này như hướng dẫn ở Bước 3.

Nếu là lần đầu sử dụng máy tính lập trình Yolo:Bit, bạn cần truy cập đường dẫn sau đây và tải file về

<https://ohstem.vn/driver>

#### Cách cài đặt driver

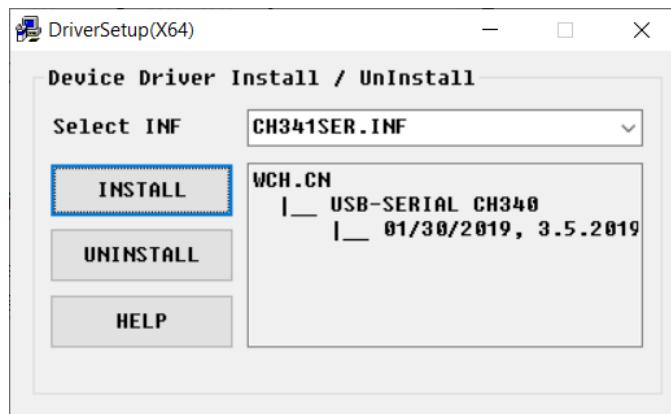
Để máy tính có thể kết nối và làm việc được với các thiết bị sử dụng chip USB CH340 như Yolo:Bit v4, robot xBot hay mạch điều khiển xController bằng dây cáp USB, bạn cần cài đặt driver theo các bước sau:

- Tải File cài đặt driver [tại đây](#). 
- Unzip File đã được download và chạy file **CH341SER.EXE** để tiến hành cài đặt.
- Click vào **INSTALL**:

Hình 1.3: Chọn tải driver CH340

### Bước 2: Tiến hành cài đặt driver

Sau khi đã tải file thành công, bạn giải nén và chạy file cài đặt **CH341SER.EXE** bằng cách **nhấn chuột phải vào file này**, và chọn tiếp vào **Run as Administrator**. Lúc này, giao diện hiển thị thông báo mới, bạn nhấn **Yes** để tới được giao diện cài đặt sau đây:

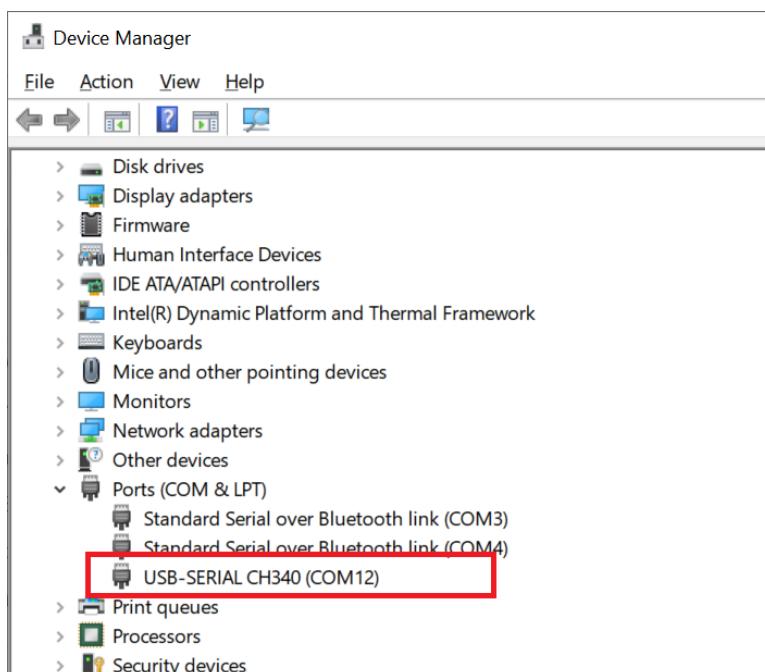


Hình 1.4: Cài đặt driver CH340 cho Yolo:Bit

Cuối cùng, bạn nhấn tiếp vào nút **Install** và tiến hành cài đặt bình thường.

### Bước 3: Kiểm tra thiết bị trong Device Manager

Để kiểm tra xem driver CH340 đã được cài đặt hay chưa, bạn cần kết nối Yolo:Bit với máy tính qua cổng USB. Sau đó, mở cửa sổ Device Manager để kiểm tra, bằng cách nhấn phím tắt **Windows +X** và chọn vào **Device Manager** trên danh sách:



Hình 1.5: Kiểm tra thiết bị trong Device Manager

Khi cài đặt driver thành công, cổng COM kết nối với mạch Yolo:Bit sẽ xuất hiện trên cửa sổ Device Manager như minh họa ở hình trên. Mỗi máy tính sẽ có 1 cổng COM kết nối khác nhau, tùy thuộc vào tài nguyên sử dụng cổng COM của máy tính. Nếu driver chưa được cài đặt, hoặc cài đặt chưa thành công, máy tính sẽ hiển thị “unknown device” tại mục Device Manager. Trong trường hợp này, bạn cần phải tải về và cài lại driver.

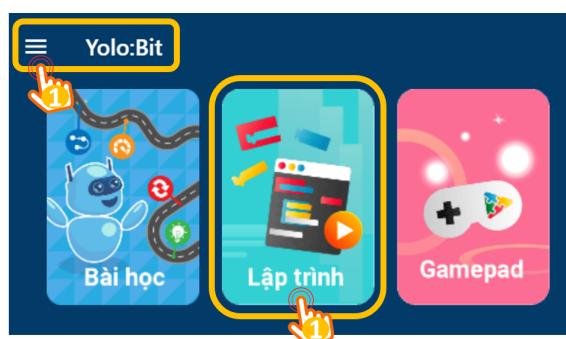
### 3 Môi trường lập trình

Để việc lập trình trở nên dễ dàng và thuận tiện, các môi trường lập trình hiện tại đã chuyển dần lên hình thức lập trình trực tuyến. Tức là bạn đọc không cần cài đặt phần mềm nữa, mà chỉ cần dùng trình duyệt web để bắt đầu lập trình. Điều này cũng có nghĩa là bên cạnh máy tính, các thiết bị di động như điện thoại thông minh hay máy tính bảng đều có thể được sử dụng để lập trình cho mạch YoloBit.

Để bắt đầu lập trình với Yolo:Bit, chúng ta cần vào trang lập trình trực tuyến sau đây:

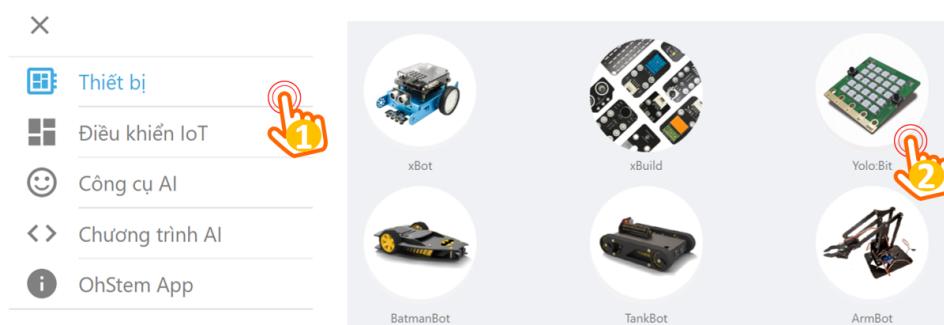
<https://app.ohstem.vn/>

Lần đầu tiên vào môi trường lập trình này, trang lập trình sẽ mặc định chọn thiết bị phần cứng là mạch YoloBit. Bạn có thể nhận ra điều này khi quan sát thấy dòng chữ **Yolo:Bit** ở gốc trên bên trái của màn hình.



Hình 1.6: Trang lập trình trực tuyến cho Yolo:Bit

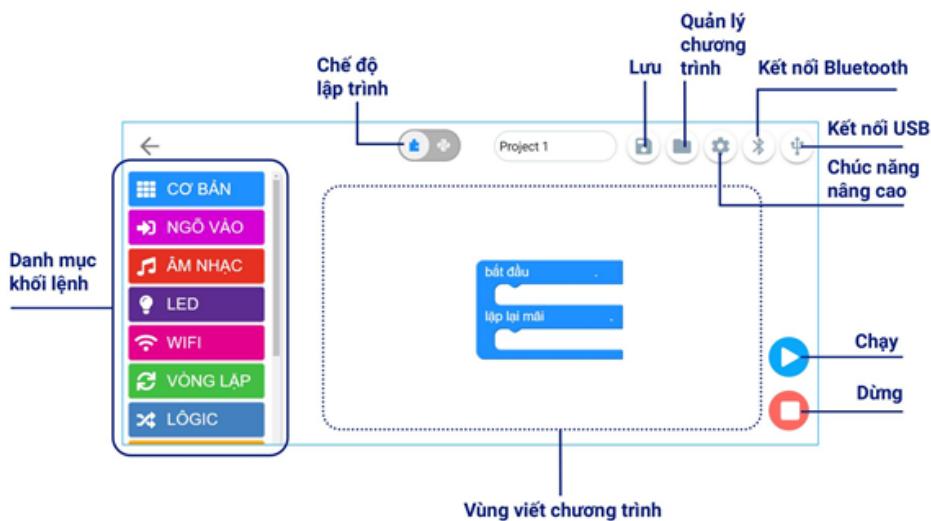
Nếu đúng là dòng chữ Yolo:Bit, bạn chỉ cần chọn tiếp vào **Lập trình**, là có thể bắt đầu các bước tiếp theo. Tuy nhiên, nếu thiết bị hiển thị không phải là YoloBit, bạn đọc hãy chọn lại thiết bị, bằng cách nhấn vào biểu tượng **Menu**, xuất hiện ở góc trên bên trái của giao diện lập trình. Giao diện sau đây sẽ hiện ra:



Hình 1.7: Hệ sinh thái các thiết bị dựa trên Yolo:Bit

Giao diện trên đây liệt kê tất cả các thiết bị trong hệ sinh thái Yolo:Bit. Bạn đọc chỉ cần chọn thiết bị cho chính xác, trong trường hợp này là mạch Yolo:Bit, thì có thể tới trang lập trình cho nó, như đã trình bày ở Hình 1.6.

Sau khi đã chọn đúng thiết bị, và chọn tiếp vào **Lập trình**, giao diện lập trình chính sẽ như sau:



Hình 1.8: Giao diện lập trình kéo thả

Môi trường tích hợp này cho phép chúng ta lập trình với nhiều ngôn ngữ khác nhau. Tuy nhiên, trong giáo trình này, chúng tôi sẽ tập trung vào ngôn ngữ "**kéo - thả**", vốn rất dễ tiếp cận đối với người mới bắt đầu, nhưng vẫn không làm mất tính logic trong ngôn ngữ lập trình.

Được kế thừa từ ngôn ngữ Blockly của Google, môi trường lập trình của Yolo:Bit cũng tuân theo những quy luật cơ bản của một ngôn ngữ kéo thả, có thể tóm tắt như sau:

- **Danh mục khối lệnh:** Các câu lệnh được sắp theo từng nhóm, mỗi nhóm có một màu riêng. Điều này giúp bạn dễ dàng tìm kiếm các câu lệnh mình cần. Dựa theo màu sắc của câu lệnh, chúng ta sẽ biết nó nằm ở nhóm lệnh nào.
- **Vùng viết chương trình:** Đây là nơi các câu lệnh được kết nối với nhau để tạo thành một chương trình chạy trên mạch Yolo:Bit.

Môi trường lập trình này có thể hỗ trợ được trên nhiều thiết bị như máy tính bảng, điện thoại di động. Tuy nhiên, nếu lập trình bằng điện thoại hoặc máy tính bảng, bạn cần kết nối với Yolo:Bit bằng Bluetooth.

Trong giáo trình này, chúng tôi sẽ tập trung hướng dẫn việc lập trình trên máy tính, vì chúng ổn định và dễ tìm lỗi hơn.

## 4 Chương trình đầu tiên

Đối với các mạch phần cứng nói chung và Yolo:Bit nói riêng, chương trình đầu tiên thường là nhấp nháy hoặc chớp tắt đèn. Đây có thể coi là chương trình đơn giản nhất, nhưng ý nghĩa của nó thì lớn hơn thế. Vì đây là chương trình đơn giản nên xác suất để chúng ta viết sai chương trình là rất thấp. Do đó, bạn có thể dùng chương trình này để kiểm tra mạch Yolo:Bit có thực sự hoạt động chính xác hay không.

Chương trình trên Yolo:Bit được tổ chức thành 2 phần với ý nghĩa như sau:

- **Khối bắt đầu:** Những câu lệnh trong khối này sẽ được thực hiện trước tiên, ngay khi bật nguồn hoặc nhấn nút Reset trên mạch Yolo:Bit (chỉ thực hiện 1 lần)
- **Khối lặp lại mãi:** Các câu lệnh trong khối này sẽ được thực hiện ngay sau đó. Nhưng sau khi thực hiện xong, nó sẽ được thực hiện lại.

Đây là dạng kiến trúc mang tính đặc trưng của lập trình ứng dụng: chương trình của bạn phải được lặp lại liên tục sau khi thiết bị hoạt động. Nếu chương trình dừng lại, điều này cũng có nghĩa là hệ thống của bạn đã chết.

Đây là điểm khác biệt lớn nhất khi bạn làm việc trên mạch lập trình Yolo:Bit so với việc lập trình trên máy tính. Một bên là chương trình mang thiên hướng ứng dụng, một bên là chương trình mang thiên hướng nghiên cứu. Chương trình dành cho một ứng dụng cần phải chính xác, hoạt động bền bỉ 24/7 và không bao giờ được dừng lại, cho đến khi phần cứng của hệ thống bị lỗi.

Chương trình đầu tiên của chúng ta khá đơn giản, chỉ sử dụng 2 câu lệnh là **hiện chữ** và **hiện hình ảnh**. Tất cả các câu lệnh này đều có màu xanh dương trong mục **CƠ BẢN**, như sau:



Hình 1.9: Chương trình đầu tiên trên mạch Yolo:Bit

Đối với câu lệnh **hiện chữ** trong mục **bắt đầu**, bạn đọc cần lưu ý rằng Yolo:Bit chỉ **hiển thị được kí tự không dấu** mà thôi. Đối với câu lệnh thứ 2 trong mục **lặp mãi mãi**, bạn cần thay đổi lựa chọn trong câu lệnh **hiện hình ảnh** bằng cách nhấn vào biểu tượng tam giác ở phía cuối câu lệnh.

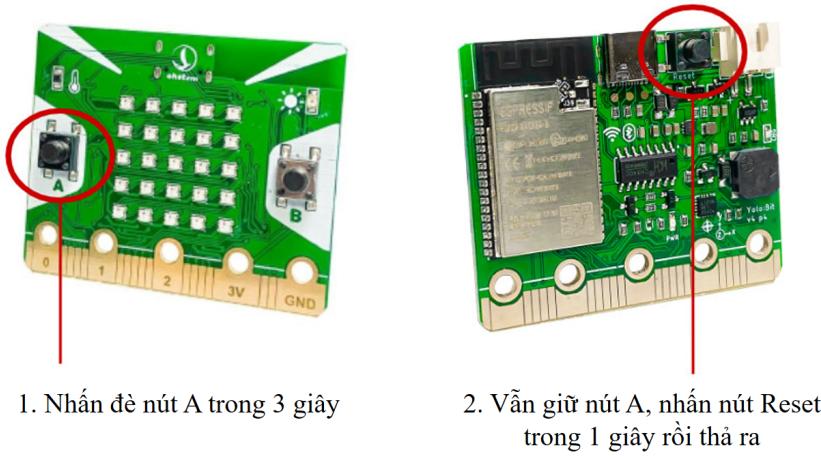
## 5 Khôi phục cài đặt gốc

Trước khi kết nối mạch Yolo:Bit với máy tính để chạy chương trình, chúng ta nên khôi phục cài đặt gốc của mạch. Đây là thao tác nên thực hiện ngay cả khi bạn đang có trên tay mạch Yolo:Bit mới. Thao tác này sẽ giảm thiểu tối đa các sai sót khi chúng ta nạp chương trình đầu tiên cho mạch Yolo:Bit.

Để khôi phục cài đặt gốc, bạn làm theo trình tự như sau:

- Nhấn đè nút A trên mạch trong khoảng 3 giây.
- Vẫn nhấn đè nút A, nhấn đè nút Reset trong khoảng 1 giây
- Thả nút Reset, vẫn giữ đè nút A
- Chờ cho đến khi đèn trên mạch Yolo:Bit chớp tắt 3 lần

Hình ảnh minh họa các thao tác trên được trình bày như sau:



*Hình 1.10: Khôi phục cài đặt gốc cho Yolo:Bit*

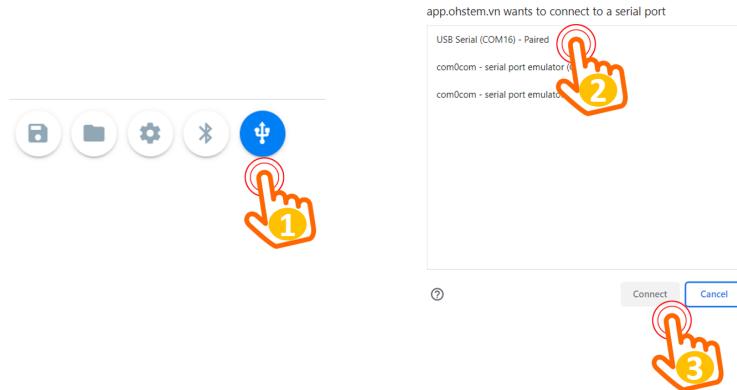
Sau khi mạch Yolo:Bit chớp tắt đèn 3 lần, nó sẽ được tự động reset lại. Khoảng 5 giây sau, một hiệu ứng đèn xoắn ốc sẽ xuất hiện. Đến lúc này, mạch Yolo:Bit đã được khôi phục cài đặt gốc thành công và sẵn sàng cho các bước tiếp theo.

## 6 Chạy chương trình trên Yolo:Bit

Sau khi đã thực hiện xong, chúng ta cần phải chuyển chương trình này từ máy tính vào Yolo:Bit để nó có thể hoạt động được. Quy trình này gồm có 2 bước cơ bản như sau:

## Bước 1: Kết nối với Yolo:Bit

Từ thanh công cụ của chương trình, bạn chọn vào biểu tượng USB, chọn cổng kết nối với mạch Yolo:Bit và chọn tiếp vào nút **Connect**, như minh họa ở hình bên dưới:



Hình 1.11: Kết nối với Yolo:Bit thông qua USB

Sau khi thực hiện thành công, biểu tượng kết nối USB sẽ chuyển sang màu xanh dương. Trong trường hợp không thấy mạch Yolo:Bit xuất hiện ở bước 2, có thể là do việc cài đặt driver không thành công. Bạn cần kiểm tra lại trong mục Device Manager của máy tính (nhấn tổ hợp phím **Windows + X**) để xem máy đã nhận dạng được mạch Yolo:Bit hay chưa.

## Bước 2: Chạy thử chương trình

Ở bước này, bạn chỉ cần nhấn vào biểu tượng nút **Play** (màu xanh dương) ở góc bên phải để chạy thử chương trình. Trong trường hợp muốn dừng, bạn nhấn vào nút **Stop** ở bên dưới. Bạn cần lưu ý rằng, đây mới chỉ là bước chạy thử, dùng để kiểm tra chương trình là chính.

Trong trường hợp chương trình không chạy trên mạch Yolo:Bit, cũng giống như một máy tính, chúng ta có thể **Reset** lại nó. Để làm việc này, chúng ta chọn vào biểu tượng **Cài Đặt**, chọn tiếp vào lựa chọn **Nhập Lệnh**, như minh họa ở hình bên dưới:



Hình 1.12: Mở cửa sổ tương tác lệnh với Yolo:Bit

Khi cửa sổ console (giao diện trắng đen như hệ điều hành MS DOS) hiện lên, chúng ta nhấp chuột vào đây và nhấn tiếp tổ hợp phím **Ctrl + D**. Đây cũng là điều mà các lập trình viên phát triển phần mềm cho Yolo:Bit thường xuyên làm để kiểm tra hệ thống.

Việc chạy thử chương trình gặp lỗi cũng thường xuyên xảy ra. Khi có 1 chương trình tương đối phức tạp (có giao tiếp với thiết bị nào đó) được nạp trực tiếp vào máy tính Yolo:Bit, việc chạy thử của chúng ta sẽ gặp vấn đề. Do đó, bạn cần phải thường xuyên mở cửa sổ console và reset lại mạch trong quá trình làm việc với nó.

## 7 Nạp chương trình vào Yolo:Bit

Với việc chạy thử ở phần trên, chương trình chỉ mới được gửi tạm tới Yolo:Bit. Nói một cách khác, chương trình này sẽ không tồn tại trên mạch Yolo:Bit mỗi khi chúng ta tắt nguồn và bật lại, tương tự như bộ nhớ tạm (RAM) của máy tính thông thường.

Để chương trình luôn được lưu lại trong Yolo:Bit và vận hành như một ứng dụng thực tế, bạn cần nạp nó vào mạch. Quy trình thực hiện cũng khá tương đồng với bước chạy thử, nhưng lần này, chúng ta sẽ chọn **Lưu dự án vào thiết bị**, như minh họa ở hình bên dưới:



Hình 1.13: Nạp chương trình cho mạch Yolo:Bit

Chúng ta nên reset lại mạch ở bước này, để khởi động lại máy tính Yolo:Bit. Có 2 cách để chúng ta reset lại mạch, là nhấn vào nút Reset (nằm gần khe cắm nguồn USB) hoặc rút khe cắm USB và gắn lại vào mạch Yolo:Bit.

Mạch Yolo:Bit sẽ cần khoảng 5 giây để khởi động lại hệ thống. Do đó, việc chúng ta xài một câu lệnh **hiện chữ** trong phần **bắt đầu** có ý nghĩa vô cùng lớn. Khi thông tin này hiện ra đèn, tức là mạch Yolo:Bit đã khởi động xong và chương trình của chúng ta bắt đầu thực thi. Bạn đọc hãy tận dụng điều này mỗi khi bắt đầu một dự án với Yolo:Bit.

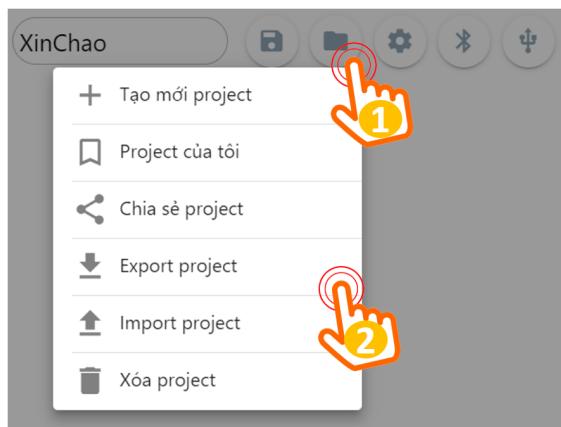
Một điều lưu ý quan trọng, là đôi khi vì chương trình mà chúng ta lưu vào thiết bị, làm cho việc chương trình mới có thể không thực thi thành công trên mạch Yolo:Bit. Lý do chủ yếu của việc này là do xung đột bộ nhớ chương trình. Trong trường hợp này, bạn đọc chỉ cần khôi phục lại cài đặt gốc của mạch Yolo:Bit và thử lại chương trình của mình.

## 8 Lưu, mở và chia sẻ chương trình

Lưu và mở lại chương trình là 2 tính năng quan trọng trong việc lập trình, và chúng cũng không ngoại lệ khi làm việc với mạch Yolo:Bit. Tính năng này được tích hợp sẵn trên thanh công cụ của môi trường lập trình:

- **Export:** Lưu chương trình (dưới dạng file .json)
- **Import:** Mở chương trình cũ (bằng file. json)

Bên dưới là hình minh họa về các công cụ liên quan đến việc lưu và mở lại dự án:

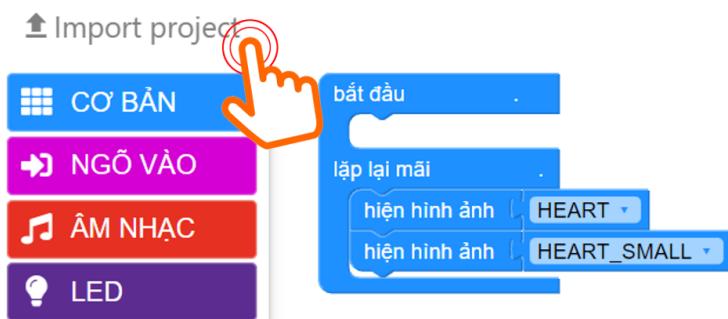


Hình 1.14: Lưu và mở lại chương trình

Ngoài ra, trong thanh công cụ này, bạn có thể dễ dàng tìm thấy tính năng **Chia sẻ project**. Tính năng này giúp bạn thuận tiện hơn trong việc trao đổi chương trình, đặc biệt là khi bạn phải giảng dạy online từ xa. Khi nhấn vào chia sẻ project, hệ thống sẽ cung cấp cho bạn một đường dẫn đến chương trình của mình, ví dụ như:

<https://app.ohstem.vn/#!/share/yolobit/25gsyDhbO0OEZewWfAtAvnCWZKq>

Khi mở đường dẫn này ở trình duyệt trên một máy tính khác, chương trình của chúng ta sẽ hiện lên. Bạn chỉ cần nhấn vào nút **Import Project** là chương trình sẽ tự động được tải về môi trường lập trình trên máy. Bạn có thể tham khảo và chỉnh sửa lại chúng, như minh họa ở hình bên dưới:



Hình 1.15: Chia sẻ chương trình trên Yolo:Bit

Trong các bài hướng dẫn tiếp theo, chúng tôi cũng sẽ tận dụng tính năng này để chia sẻ chương trình trong giáo trình đến bạn, để bạn dễ dàng tham khảo.



# CHƯƠNG 2

## Ngoại vi cơ bản trên Yolo:Bit



# 1 Giới thiệu

Giống như mô hình máy tính cơ bản, Yolo:Bit là một mạch tích hợp với bộ xử lý trung tâm, các thiết bị xuất dữ liệu và nhập dữ liệu. Nhờ việc thiết kế tích hợp này, bản thân mạch Yolo:Bit có thể dùng để làm nhiều dự án đơn giản mà không cần thêm các thiết bị ngoại vi khác cũng như các kết nối mở rộng. Tuy nhiên, trong phạm vi của giáo trình này, chúng tôi chỉ trình bày sơ lược một số ngoại vi quan trọng trên mạch Yolo:Bit, trước khi tập trung vào các ứng dụng liên quan đến Kết nối vạn vật và Trí tuệ nhân tạo.

Trong đa số các hệ thống điều khiển và xử lý nói chung, ngoại vi sẽ được chia làm 2 loại: Xuất dữ liệu và Nhập dữ liệu. Tuy nhiên ở mức độ cơ bản, chúng tôi tập trung vào 3 phân loại sau đây trên mạch Yolo:Bit:

- Màn hình hiển thị: Với 25 đèn hiển thị độc lập, đây sẽ là công cụ hữu ích để làm việc trên mọi dự án và chương trình trên Yolo:Bit. Màn hình hiển thị là công cụ quan trọng để hiện thị kết quả. Rõ ràng, nó là 1 ngoại vi thuộc nhóm xuất dữ liệu.
- Nút nhấn: Trên mạch Yolo:Bit hỗ trợ 2 nút nhấn tín hiệu là A và B. Nút nhấn Reset không thuộc trong nhóm này, nó là 1 dạng nút nhấn thuộc về hệ thống và không thể lập trình được. Nút nhấn thuộc nhóm nhập dữ liệu.
- Cảm biến: cũng thuộc nhóm nhập dữ liệu. Tuy nhiên, chúng tôi tách nó thành 1 nhóm mới để việc hướng dẫn được tập trung hơn. Cảm biến là các thiết bị cung cấp dữ liệu từ môi trường theo một điều kiện nào đó. Cảm biến chính là thông tin đầu vào cho các ứng dụng thông minh mà chúng ta gặp rất nhiều trong đời sống hiện nay. Một điểm thú vị trong mục này, là các cảm biến **hành vi**, chẳng hạn như lắc mạch, nghiêng trái hay nghiêng phải, được tích hợp sẵn trên mạch Yolo:Bit

Trong bài hướng dẫn này, chúng tôi sẽ trình bày các câu lệnh chính liên quan đến 3 nhóm thiết bị nói trên. Các chương trình trong bài hướng dẫn này sẽ là một công cụ kiểm tra cần thiết khi bạn đọc tích hợp thêm các tính năng mới và xây dựng một dự án lớn trong tương lai. Các mục tiêu chính trong bài hướng dẫn này được tóm tắt như sau:

- Hiển thị thông tin trên Yolo:Bit
- Làm việc với nút nhấn A và B
- Truy xuất giá trị cảm biến trên Yolo:Bit

Các câu lệnh sử dụng trong bài hướng dẫn này thuộc 2 nhóm chính là **CƠ BẢN** và **NGÕ VÀO**. Đây có thể xem là 2 nhóm lệnh liên quan đến xuất kết quả và đọc dữ liệu đầu vào trên mạch Yolo:Bit. Đối với việc lập trình trên các mạch điện phần cứng, việc nắm rõ vai trò của một thiết bị (đầu vào hay đầu ra) là rất quan trọng để phát triển các chương trình trong tương lai.

## 2 Hiển thị thông tin trên Yolo:Bit

Nói chung, các ngoại vi xuất kết quả thường có độ phức tạp thấp hơn nhiều so với các ngoại vi nhập dữ liệu. Trên mạch Yolo:Bit cũng không phải là ngoại lệ, khi các câu lệnh dùng cho việc hiển thị nằm trong nhóm **CƠ BẢN**. Các câu lệnh chính mà chúng ta thường sử dụng được trình bày như sau.

### 2.1 Hiển chữ - Hiển số

Cho dù là chương trình đơn giản hay phức tạp, bạn đọc hãy luôn luôn hiển ra 1 thông tin ra màn hình bằng câu lệnh **hiển chữ**. Câu lệnh này sẽ được đặt đầu tiên trong phần **bắt đầu**. Điểm lưu ý quan trọng là Yolo:Bit **không hiển thị được tiếng Việt có dấu**.



Hình 2.1: Câu lệnh hiển chữ và hiển số

Khác với câu lệnh **hiển số**, câu lệnh **hiển chữ** có tầm ảnh hưởng rộng hơn. Phần nội dung của nó có thể được ghép với 1 con số cũng được. Ngược lại, câu lệnh **hiển số** chỉ có thể xuất được 1 con số ra màn hình mà thôi. Do đó, trong trường hợp không chắc chắn dữ liệu là chuỗi hay là số, bạn có thể dùng câu lệnh **hiển chữ** cho an toàn.

Để hiểu rõ hơn tính năng của 2 câu lệnh trên, bạn đọc có thể hiện thực các chương trình sau đây để kiểm tra tính năng của nó. Để thay đổi nội dung trong 2 câu lệnh này, chúng ta đơn giản là nhấp chuột vào và gõ nội dung mới.



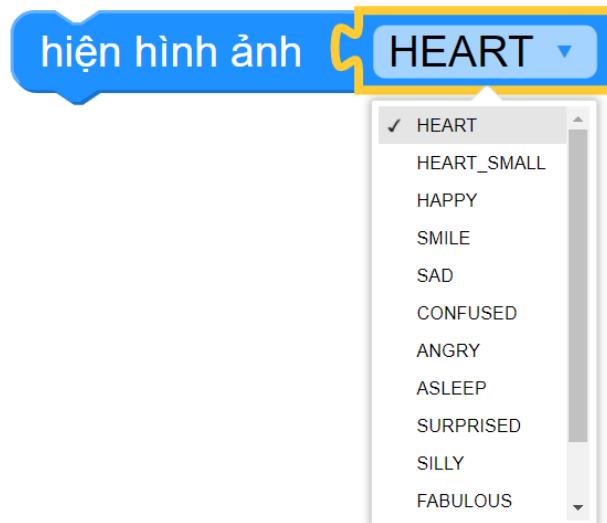
Hình 2.2: Kiểm tra tính năng của câu lệnh hiển chữ và hiển số

Với 2 câu lệnh này, khi thông tin chỉ có 1 chữ số hoặc 1 kí tự, nó sẽ được hiển thị cố định trên màn hình. Với các thông tin có hơn 2 chữ số hoặc 2 kí tự, nó sẽ được dịch chuyển từ phải qua trái. Câu lệnh hiển số có thể hiển thị được số âm lẫn số thập phân.

### 2.2 Hiển hình ảnh

Hiển hình ảnh là câu lệnh phổ biến tiếp theo khi làm việc với mạch Yolo:Bit. Trên Yolo:Bit, chúng ta có 2 phiên bản của câu lệnh này. Đầu tiên, trong câu lệnh thứ

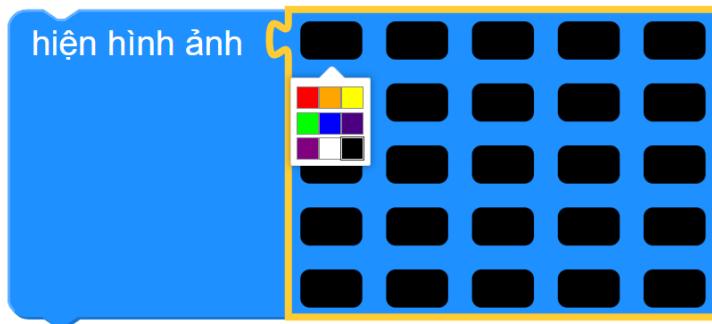
nhất, chúng ta có một danh sách các hình ảnh có thể được chọn lựa, như trình bày ở hình bên dưới:



Hình 2.3: Câu lệnh hiện hình ảnh thứ nhất trên Yolo:Bit

Các hình ảnh thường sử dụng như là mặt cười (HAPPY, SMILE) hoặc mặt khóc (SAD, ANGRY). Bạn đọc hoàn toàn có thể chủ động kiểm tra các hình ảnh này bằng cách sử dụng câu lệnh hiện hình ảnh trong phần bắt đầu của chương trình.

Trong câu lệnh hiện hình ảnh thứ 2, được trình bày như hình bên dưới, chúng ta có một màn hình 25 đèn để có thể chủ động tạo ra hình ảnh cho riêng mình.Thêm nữa, tại mỗi đèn, cũng có rất nhiều màu sắc khác nhau cho chúng ta lựa chọn.



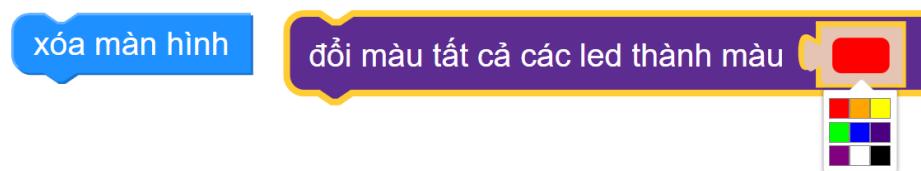
Hình 2.4: Câu lệnh hiện hình ảnh thứ 2 Yolo:Bit

Màu đen tức là đèn sẽ tắt, các lựa chọn còn lại dùng để chỉnh màu của từng bóng đèn.

Về mức độ phổ biến khi lập trình, câu lệnh thứ nhất thường sẽ được dùng nhiều hơn, do nó đơn giản và hiệu quả hơn so với câu lệnh thứ 2. Trừ khi muốn tạo ra các hình ảnh riêng với màu sắc tùy chọn, câu lệnh thứ 2 mới được sử dụng.

### 2.3 Xóa màn hình

Bên cạnh việc hiển thị thông tin, xóa màn hình cũng là tính năng phổ biến trong các dự án. Chẳng hạn như khi xây dựng 1 ứng dụng nhập mật mã, chúng ta sẽ cho mật mã hiển thị ra trong 1 thời gian ngắn rồi xóa nó đi. Câu lệnh này được trình bày như hình bên dưới.

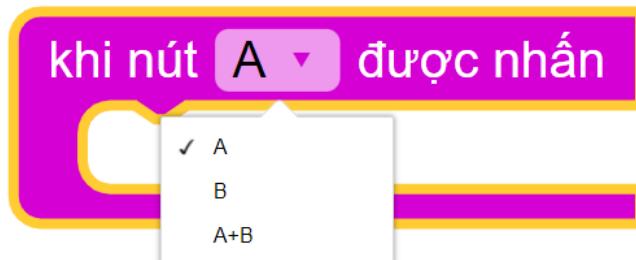


Hình 2.5: Câu lệnh xóa màn hình trên Yolo:Bit

Bên cạnh câu lệnh thứ nhất, xóa màn hình, câu lệnh thứ 2 cũng là một câu lệnh tiện dụng, câu lệnh **đổi màu tắt cả các led thành màu**. Câu lệnh này nằm trong nhóm **LED**. Với lựa chọn màu đen, tắt cả các đèn sẽ tắt.

### 3 Nút nhấn trên Yolo:Bit

Trên mạch Yolo:Bit hỗ trợ 2 nút nhấn là A và B. Để có thể biết được khi nào một nút được nhấn, môi trường lập trình hỗ trợ cho chúng ta câu lệnh **khi nút được nhấn**. Câu lệnh này nằm cuối cùng trong nhóm lệnh **NGÕ VÀO**, như được trình bày ở hình bên dưới:



Hình 2.6: Câu lệnh nút nhấn trên Yolo:Bit

Đây là câu lệnh có 3 lựa chọn, dành cho nút A, nút B hoặc khi cả A và B được nhấn đồng thời. Điều đặc biệt của khối lệnh này là nó là một dạng câu lệnh sự kiện. Tức là mặc dù chương trình đang thực hiện tính năng nào đó trong phần **lắp mái mái**, mỗi khi có tín hiệu nút nhấn, chương trình sẽ tạm ngưng lại để thực hiện các câu lệnh trong phần **sự kiện**.

Cũng bởi vì câu lệnh này là câu lệnh sự kiện, nó có thể đứng riêng lẻ trong môi trường lập trình mà không cần thiết phải được ghép với 1 câu lệnh nào cả, như minh họa ở chương trình sau đây:



Hình 2.7: Chương trình ví dụ cho nút nhấn

Với chương trình ví dụ này, trong khi hiển thị hình trái tim luân phiên nhau, mỗi khi nút nhấn được nhấn, chương trình sẽ ngay lập tức chuyển sang thực hiện các câu lệnh bên trong sự kiện tương ứng, rồi mới quay lại thực hiện tiếp các câu lệnh trong phần **lặp mãi mãi**. Chương trình trong các khối sự kiện nút nhấn còn gọi là chương trình ngắn. Đây là một kiến trúc rất đặc trưng trên các hệ thống vi điều khiển nói chung, và mạch Yolo:Bit nói riêng. Nhờ kiến trúc này, trong khi đang thực hiện 1 chức năng trong lặp mãi mãi, nó có thể tạm dừng để thực hiện 1 số chức năng ưu tiên trong các khối lệnh sự kiện khi một nút nhấn được nhấn.

## 4 Cảm biến trên Yolo:Bit

### 4.1 Nhiệt độ và Ánh sáng

Tích hợp sẵn trên mạch Yolo:Bit là 2 cảm biến về **nhiệt độ** và **cường độ sáng**. Hai thông tin này đều nằm trong mục **NGÕ VÀO**. Một chương trình để kiểm tra 2 thông tin này được gợi ý như sau:



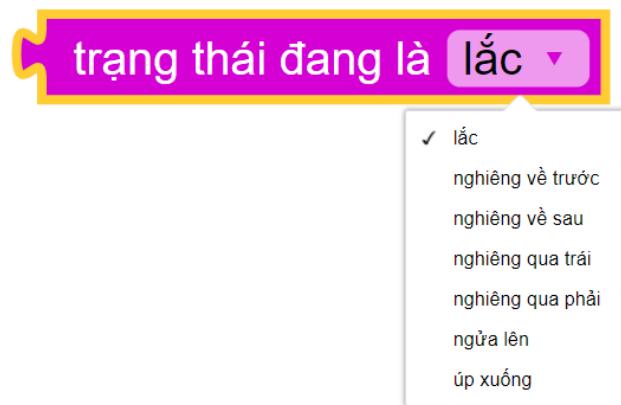
Hình 2.8: Chương trình ví dụ cho cảm biến

Do thông tin cảm biến đều là dữ liệu số, do đó chúng ta phải xài câu lệnh **hiện số** để hiển thị kết quả. Một lưu ý quan trọng dành cho thông tin nhiệt độ, vốn là

nhiệt độ trên bo mạch Yolo:Bit và không phải là nhiệt độ của môi trường. Do đó, nó thường sẽ cao hơn nhiệt độ môi trường không khí. Tuy nhiên, trong các ví dụ minh họa, bạn cũng có thể sử dụng thông tin này và trừ đi 1 lượng nhỏ để biến diễn nhiệt độ không khí.

## 4.2 Cảm biến hành vi

Nhờ cảm biến gia tốc và gốc xoay tích hợp sẵn trên mạch, Yolo:Bit có khả năng nhận biết hành vi tương tác của người dùng. Đây là công nghệ được ứng dụng trong các điện thoại thông minh, khi nó có khả năng nhận biết bạn đang đi bộ, leo cầu thang hay đang chạy bộ. Khối lệnh về cảm biến hành vi được trình bày bên dưới, với nhiều lựa chọn khác nhau cho các ứng dụng tương tác.



Hình 2.9: Câu lệnh cho cảm biến hành vi

Tuy nhiên, khối lệnh hành vi này là một dạng điều kiện, với kết quả là **Đúng** hoặc **Sai**. Cho nên để sử dụng nó, chúng ta sẽ phải xài kết hợp với câu lệnh **nếu ... thực hiện**. Câu lệnh này nằm trong mục **LOGIC**.

Thêm nữa, việc kiểm tra hành vi tương tác sẽ phải làm thường xuyên. Do đó, các câu lệnh **nếu ... thực hiện** sẽ được đặt trong khối **lặp mãi mãi**. Tuy nhiên, để hệ thống ổn định hơn, bạn nên đặt thêm 1 khối lệnh **tạm dừng 100 mili giây**. Đôi với người bình thường, 100ms là 1 khoảng thời gian rất nhỏ và khó nhận ra. Tuy nhiên đối với hệ thống xử lý, nó là một khoảng thời gian rất dài để nó có thể khởi tạo lại các thông số cảm biến.

Một chương trình gợi ý để sử dụng câu lệnh này như sau:



Hình 2.10: Sử dụng câu lệnh hành vi

Với chương trình này, bạn có thể cầm và lắc mạch Yolo:Bit để nó có thể hiện ra dòng chữ LAC hoặc nghiêng mạch sang trái hay sang phải. Trong tương lai, tương tác kiểu hành vi này có thể dùng cho việc hiện thực 1 tay cầm điều khiển Robot từ xa hoặc điều khiển thiết bị bằng hành vi, mà không cần phải nhấn trực tiếp vào nút nhấn.

## CHƯƠNG 3

# Đồng hồ thông minh Internet



# 1 Giới thiệu

Đồng hồ thông minh là một dự án cực kì hữu ích cho những ai muốn tiếp cận với việc lập trình trên các mạch điện như Yolo:Bit. Khi bạn có thể hiện thực thành công dự án này, có thể kết luận rằng bạn đã rất thuần thục về việc xử lý đầu vào (input) và đầu ra (output) trên hệ thống phần cứng. Sở dĩ có thể kết luận như vậy, là vì chỉ với một màn hình hiển thị đơn giản và 2 nút nhấn, chúng ta phải làm rất nhiều thì mới có thể xây dựng đủ các chức năng của một đồng hồ, bao gồm việc chỉnh giờ, ngày tháng hoặc là hẹn giờ chẳng hạn.

Tuy nhiên, khác với các dự án về đồng hồ khác, mạch Yolo:Bit vốn đã có sẵn khôi kết nối với mạng Internet. Đây là lợi thế vô cùng lớn của Yolo:Bit khi làm ứng dụng này, bởi chúng ta sẽ có thể lấy thông tin về thời gian trên mạng và xử lý trực tiếp. Do đó, việc lập trình sẽ trở nên đơn giản hơn và bạn đọc có thể phát triển nhiều tính năng hơn cho đồng hồ của mình. Đối với các hệ thống khác, khi không hỗ trợ kết nối Internet, việc giúp thiết bị chạy đúng giờ là không hề đơn giản.

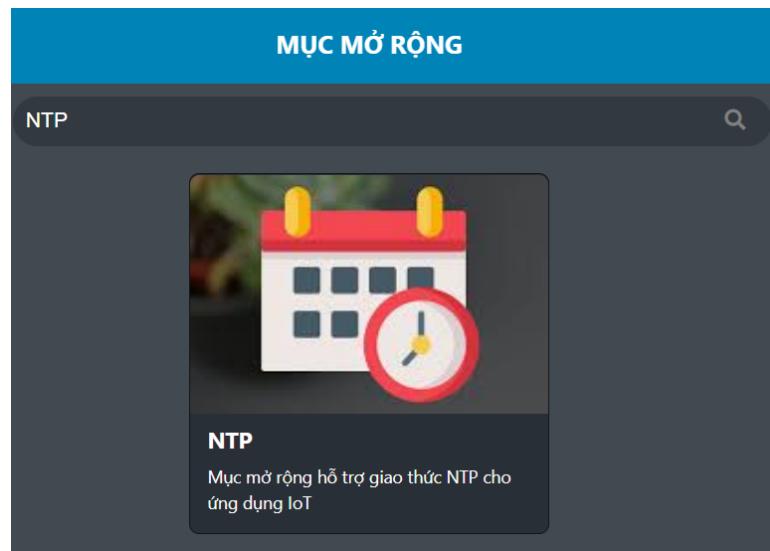
Với 2 nút nhấn A và B cùng màn hình gồm 25 đèn LED, chúng ta có thể xây dựng một số chức năng của đồng hồ thông minh như sau:

- Nhấn vào nút A để xem thông tin về nhiệt độ
- Nhấn vào nút B để xem thông tin về ánh sáng
- Chạm tay vào chân 0 để xem thông tin về thời gian
- Chạm tay vào chân 1 để xem thông tin về ngày tháng

# 2 Thư viện lập trình cho thời gian

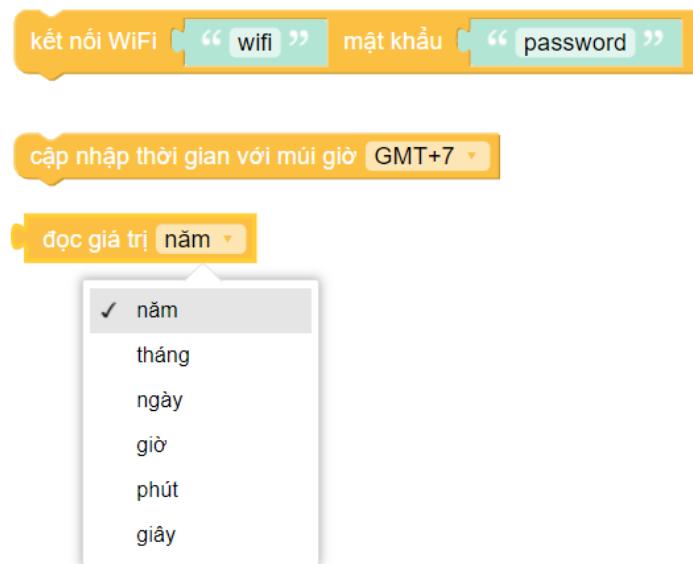
Thời gian trên Yolo:Bit được xây dựng dựa trên khái niệm Network Time Protocol (giao thức thời gian trên mạng). Network Time Protocol (NTP) là một thuật toán phần mềm, có công dụng giữ cho các máy tính và các thiết bị công nghệ khác nhau có thể đồng bộ hóa thời gian với nhau. NTP đã đạt được thành công trong việc đồng bộ hóa thời gian các thiết bị hiệu quả, chỉ trong vài milli giây (1 giây = 1000 milli giây).

Để thêm thư viện lập trình NTP, chúng ta chọn vào mục **MỞ RỘNG**, sau đó nhập từ khóa **NTP** vào ô tìm kiếm và nhấn Enter (hoặc nhấn vào biểu tượng tìm kiếm). Kết quả của việc tìm kiếm sẽ hiện ra như sau:



Hình 3.1: Thư viện lập trình NTP cho thời gian

Sau khi chọn vào kết quả tìm kiếm, khôi lệnh mới trong NTP sẽ được thêm vào chương trình, với 3 khôi lệnh chính như sau:



Hình 3.2: Các câu lệnh trong khôi NTP

**Câu lệnh 1:** Dùng để kết nối vào mạng Internet. Bạn cần cung cấp thông tin về tên mạng WiFi cũng như mật khẩu để mạch Yolo:Bit có thể đăng nhập vào mạng Internet.

**Câu lệnh 2:** Dùng để cập nhật múi giờ của hệ thống. Đây là câu lệnh mà bạn bắt buộc phải dùng, và bạn nên đặt nó trong phần **bắt đầu**, bởi nó mang tính chất cấu hình và chỉ cần được thực hiện 1 lần.

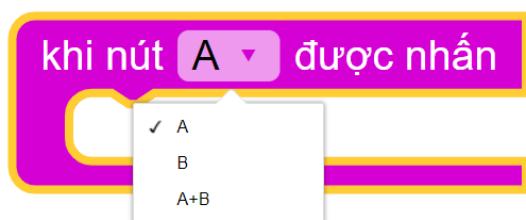
**Câu lệnh 3:** Có rất nhiều tùy chọn về thời gian giúp bạn đọc có thể thoải mái sử dụng trong ứng dụng của mình.

### 3 Lập trình trên Yolo:Bit

Sau khi đã thêm đầy đủ thư viện, chúng ta bắt đầu lập trình cho mạch Yolo:Bit. Với các tính năng đã liệt kê ra ở phần giới thiệu, chúng ta sẽ bắt đầu với 2 nút nhấn A và B.

#### 3.1 Lập trình cho nút A và B

Nhờ công cụ lập trình được hỗ trợ sẵn, khôi sự kiện cho nút A và B - **khôi lệnh khi nút được nhấn** trong mục **NGÕ VÀO** sẽ được sử dụng, như minh họa trong hình bên dưới:



Hình 3.3: Khôi lệnh sự kiện cho nút nhấn A và B

Đây là khôi lệnh có thể chọn lựa được, với 3 tùy chọn khác nhau, dành cho nút A, nút B và khi nhấn 2 nút cùng lúc. Với yêu cầu của dự án đồng hồ thông minh, chúng ta chỉ cần 2 khôi cho 2 nút A và B mà thôi. Chương trình gợi ý cho việc hiển thị thông tin khi nhấn nút sẽ như sau:



Hình 3.4: Chương trình cho nút nhấn A và B

Với các thông tin về nhiệt độ và ánh sáng, bạn có thể tìm chúng trong phần **NGÕ VÀO**. Lưu ý: Nhiệt độ ở đây là nhiệt độ của mạch Yolo:Bit. Nó khá gần với nhiệt độ môi trường (cao hơn khoảng 1 - 2 độ) và có thể dùng để minh họa cho thông tin về nhiệt độ.

#### 3.2 Khởi tạo đồng hồ Internet

Để có được thông tin về thời gian, mạch Yolo:Bit cần được kết nối vào mạng và chỉnh lại múi giờ tại Việt Nam (là **GMT+7**). Thao tác này chỉ cần thực hiện 1 lần, và

sẽ được đặt trong khối **bắt đầu**, như gợi ý sau đây:

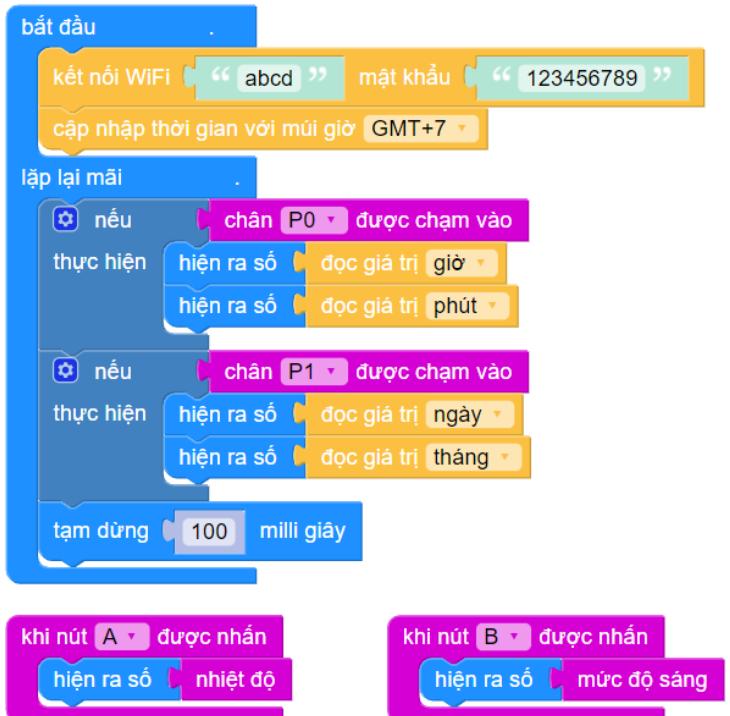


Hình 3.5: Chương trình cho nút nhấn A và B

Trong chương trình này, chúng ta kết nối với mạng WiFi và cấu hình cho múi giờ tại Việt Nam, là GMT+7.

### 3.3 Hiển thị thông tin thời gian

Mạch Yolo:Bit hỗ trợ cảm biến chạm, chúng ta có thể dùng ngón tay chạm vào các chân **P0**, **P1** và **P2** (kí hiệu 0, 1 và 2 trên mạch). Trong phần này, chúng ta sẽ thường xuyên kiểm tra xem người dùng có chạm vào các chân này hay không, để hiển thị ra các thông tin tương ứng. Do đó, tính năng này sẽ được lập trình trong khối **lặp mãi mãi**, như sau:



Hình 3.6: Hiển thị thông tin thời gian

Thực ra, việc kiểm tra các cảm biến chạm không cần thiết phải thực hiện liên tục. Do đó, một câu lệnh đợi 100ms được thêm vào ở cuối vòng lặp mãi mãi. Tuy nhiên, đối với người dùng, 100ms là rất nhỏ. Chúng ta sẽ có cảm giác rằng mỗi khi chạm vào, ngay lập tức thông tin về thời gian sẽ được hiển thị. Chương trình trên được chia sẻ ở đường dẫn sau:

<https://app.ohstem.vn/#!/share/yolobit/26Y7l09iigK0hFKeocD1yPB2bhr>

Để cho việc hiển thị đẹp hơn, bạn đọc có thể sử dụng câu lệnh ghép chuỗi trong phần **NÂNG CAO** và chọn tiếp **CHỮ VIẾT**. Câu lệnh mà chúng ta sẽ sử dụng là **tạo chuỗi từ**, như minh họa ở hình bên dưới:

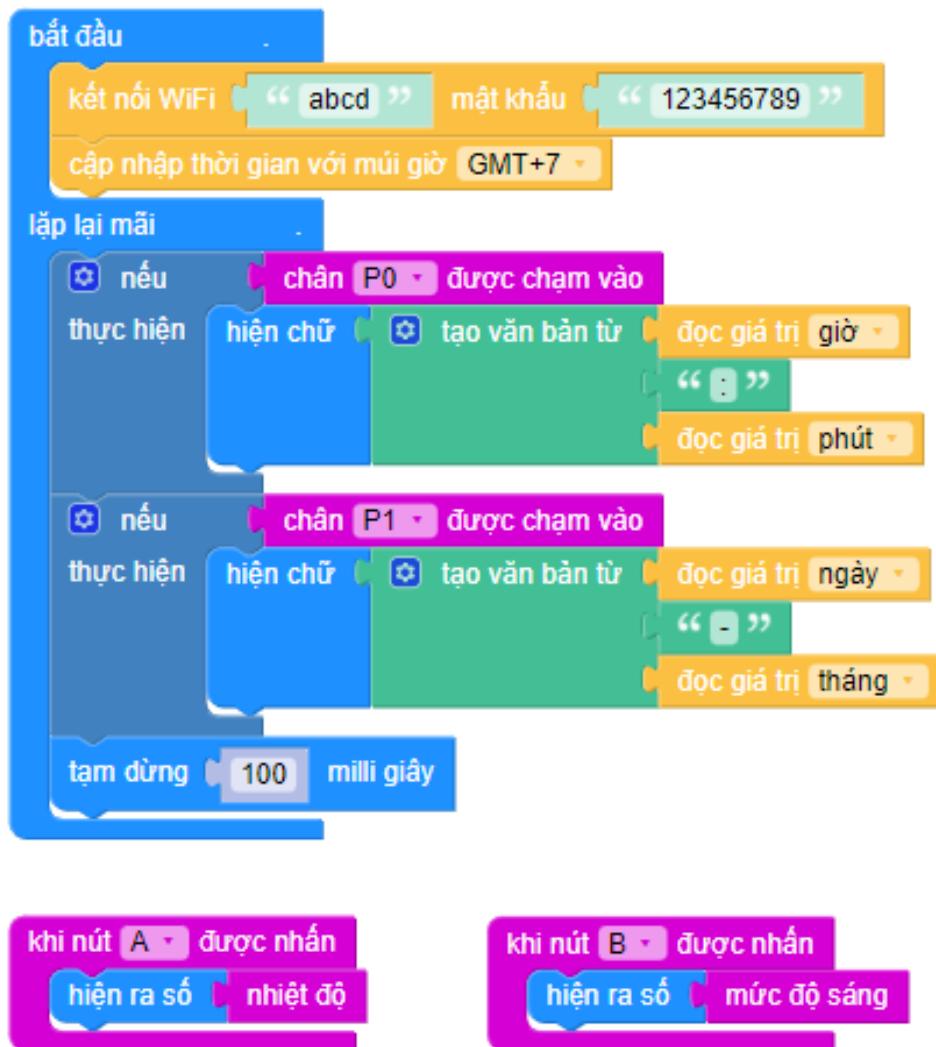


Hình 3.7: Câu lệnh ghép chuỗi

Tuy nhiên, mặc định câu lệnh này chỉ **ghép được 2 thông tin**. Trong trường hợp chúng ta cần 3 thông tin, chẳng hạn như **giờ**, **dầu hai chấm** rồi **tới phút**, chúng ta sẽ thực hiện:

- Nhấn vào biểu tượng cài đặt của câu lệnh
- Kéo thêm đối tượng **vật** ghép vào bên phải, như minh họa ở hình trên
- Sau đó, nhấn lại một lần nữa vào biểu tượng cài đặt để tắt chức năng này đi.

Hình ảnh của chương trình được chia sẻ như hình bên dưới:



Hình 3.8: Chương trình sau khi hiệu chỉnh phần hiển thị

Khi thực hiện câu lệnh ghép chuỗi, chúng ta phải dùng câu lệnh **hiển chữ**, thay vì hiển số như chương trình ban đầu. Chương trình sau khi đã hiệu chỉnh phần hiển thị, được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/26Y8nXJGC5Io1LorfXbJsTssgEL>



## **Phần II**

# **Kết Nối Mở Rộng Cho Yolo:Bit**



# CHƯƠNG 4

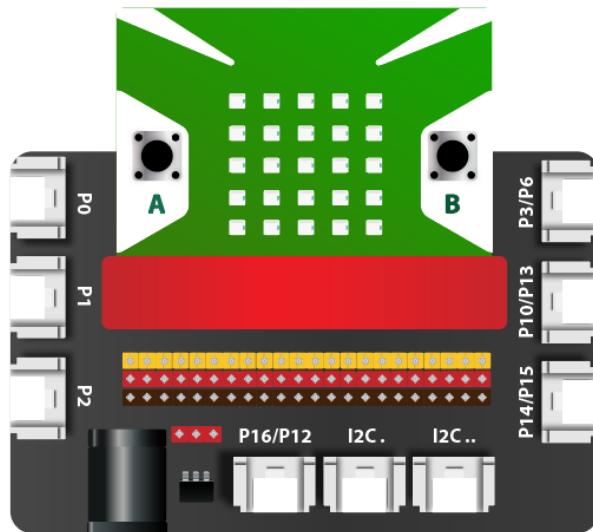
## Mạch Mở Rộng - Đèn RGB



# 1 Giới thiệu

Chỉ với 1 bản mạch tích hợp Yolo:Bit, bạn đã có thể làm nhiều ứng dụng với nó. Mặc dù trên mạch có sẵn các cảm biến tích hợp, như cảm biến nhiệt độ, ánh sáng hay cảm biến gia tốc, nhưng đôi lúc ta cần kết nối thêm các cảm biến bên ngoài để làm các ứng dụng thú vị và gần với thực tế hơn. Chẳng hạn, với ứng dụng Nhà Thông Minh trong giáo trình này, chúng ta sẽ cần kết nối thêm với các cảm biến đo chất lượng không khí, cảnh báo cháy, hiển thị thông tin trên màn hình LCD hoặc là bật tắt các công tắc và điều khiển các thiết bị tương ứng.

Để đơn giản hóa việc kết nối giữa Yolo:Bit và các cảm biến bên ngoài, giúp bạn có thể tiếp cận dễ dàng hơn, từ bài hướng dẫn này, chúng tôi sẽ kết hợp mạch Yolo:Bit với một mạch mở rộng (như hình ảnh minh họa bên dưới). Nhờ mạch mở rộng, các kết nối với thiết bị bổ sung đã được chuẩn hóa và rất thuận tiện cho bạn đọc. Khi gắn Yolo:Bit vào mạch mở rộng, **25 bóng đèn của nó phải hướng lên trên**.



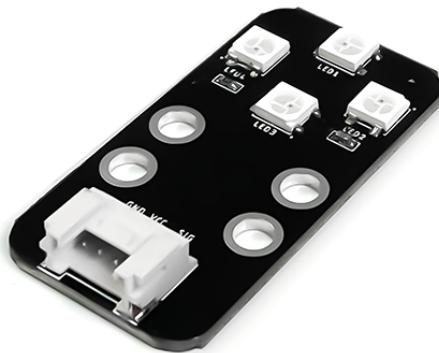
Hình 4.1: Mạch mở rộng cho Yolo:Bit

Không chỉ hỗ trợ về mặt kết nối, chúng tôi cũng sẽ giới thiệu đến bạn đọc các bộ thư viện thông dụng, giúp làm việc với các thiết bị kết nối thêm vào hệ thống (thường gọi là ngoại vi). Bộ thư viện sẽ đơn giản hóa việc điều khiển các thiết bị ngoại vi này. Trong bài đầu tiên về các thiết bị mở rộng, chúng ta sẽ thử kết nối và điều khiển với đèn 3 màu RGB. Các mục tiêu hướng dẫn trong bài này như sau:

- Kết nối Yolo:Bit và đèn màu RGB
- Thêm thư viện lập trình HOME:BIT V3
- Điều khiển đèn màu RGB ở mức đơn giản
- Thay đổi kết nối với đèn RGB

## 2 Đèn chiếu sáng RGB

Một trong những thiết bị đầu tiên mà chúng ta sẽ làm quen là đèn chiếu sáng RGB (còn gọi là module 4 LED RGB), viết tắt của Red - Green - Blue (tương ứng với 3 màu là đỏ, xanh lá và xanh dương). Đây là 3 màu cơ bản dùng để phô ra rất nhiều màu trong thực thế. Do đó, với thiết bị này, bạn có thể sáng tạo ra nhiều ánh sáng đẹp mắt cho ứng dụng của mình.



Hình 4.2: Hình ảnh module 4 LED RGB

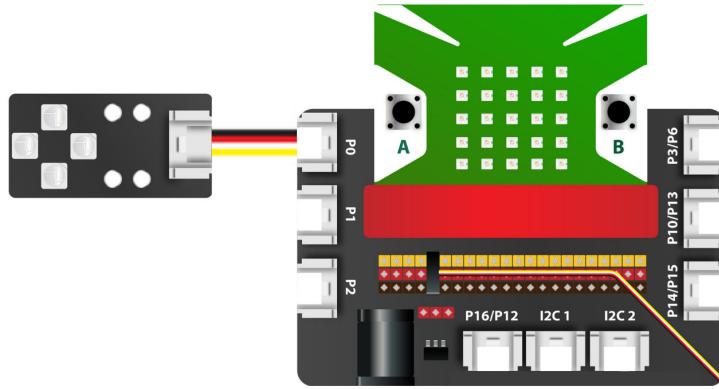
Trên thực tế, module trên là một dãy 4 đèn nối tiếp với nhau, gọi là NeoPixel. Đây là một sản phẩm được công ty Adafruit IO đưa ra, nhằm tránh lỗi chân cắm và giúp làm đơn giản hóa mạch điện. Toàn bộ dây LED có thể mở rộng lên đến 144 đèn, mỗi đèn có thể điều khiển được màu sắc riêng của nó từ 3 màu cơ bản RGB. Bạn hoàn toàn có thể sử dụng chúng để làm các hiệu ứng ánh sáng trong dự án của mình.

## 3 Kết nối với Yolo:Bit

Trên mạch mở rộng của Yolo:Bit, mỗi khe cắm đều sẽ có tên của chân kết nối. Đây là thông tin quan trọng cho việc lập trình sau này. Trước khi kết nối, bạn cần chuẩn bị những thiết bị sau đây:

- Yolo:Bit và mạch mở rộng
- Module 4 LED RGB
- Dây kết nối

Tiếp theo, bạn có thể kết nối như hình minh họa bên dưới (module 4 LED RGB được nối với chân P0 của Yolo:Bit):



Hình 4.3: Kết nối đèn RGB vào chân P0

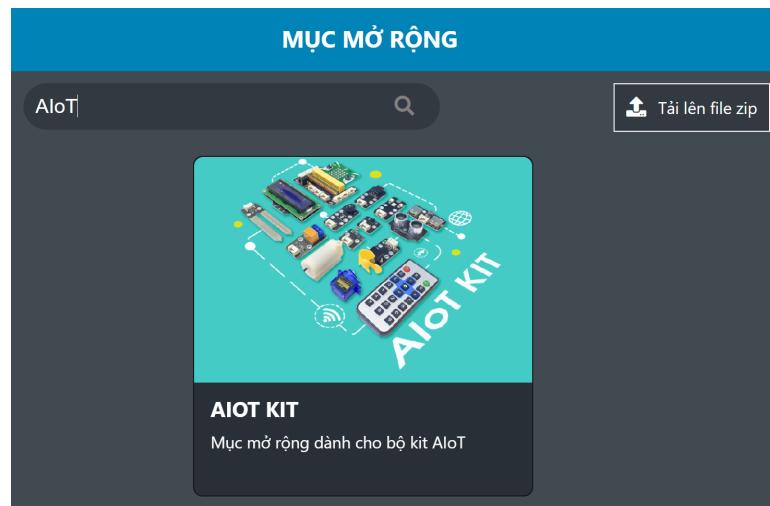
Nhờ các dây điện đã chuẩn hóa về nguồn đất và tín hiệu điều khiển, bạn gần như không cần phải lo lắng về việc kết nối các thiết bị với nhau. Các dây kết nối cũng chỉ có 1 chiều cắm, nhằm hạn chế tối đa việc cắm ngược, gây chập nguồn và hư hỏng thiết bị. Đây là một lợi thế vô cùng lớn của hệ thống mạch điện khi đã được chuẩn hóa. Lúc này, xác suất gặp lỗi sẽ ít hơn và bạn có thể tập trung vào việc lập trình của mình, thay vì phải tốn thời gian cho việc tìm lỗi từ mạch điện.

## 4 Thêm thư viện lập trình AIOT

Như đã trình bày ở trên, đèn 4 LED RGB là một thiết bị tích hợp. Cụ thể, nó được điều khiển bằng tín hiệu hình xung, vốn là một khái niệm khá phức tạp trong việc lập trình dành cho các thiết bị điện tử. Do đó, để thuận lợi cho người dùng, OhStem có hỗ trợ thư viện lập trình dùng cho các thiết bị ngoại vi của Yolo:Bit, có tên là **AIOT**.

Không chỉ sử dụng trong bài này, **AIOT** còn sẽ được sử dụng xuyên suốt trong các phần hướng dẫn tiếp theo. Dựa vào thư viện này, bạn có thể lập trình với rất nhiều thiết bị cao cấp khác, như quạt (motor), cảm biến nhiệt độ và độ ẩm không khí, cảm biến độ ẩm đất hay các công tắc điện tử, màn hình LCD.

Để thêm thư viện AIOT, bạn cần **kết nối mạch Yolo:Bit với môi trường lập trình**, chọn tiếp vào **MỞ RỘNG** và tìm kiếm từ khóa **AIOT**, giao diện sau đây sẽ hiện ra.



Hình 4.4: Thư viện mở rộng AIOT

Sau khi nhấn chọn vào thư viện này, một nhóm lệnh mới sẽ được đồng thời thêm vào môi trường lập trình trực tuyến và **tải vào mạch Yolo:Bit**, như hình dưới:



Hình 4.5: Các câu lệnh thuộc thư viện HOME:BIT V3

Trong trường hợp việc tải thư viện vào mạch Yolo:Bit không thành công, bạn đọc có thể chủ động tải lại bằng một trong 2 cách sau đây:

- Vào lại mục **MỞ RỘNG** và thêm lại thư viện **AIOT**
- Từ biểu tượng cài đặt (hình bánh răng), chọn vào **Tải Thư Viện**

## 5 Lập trình bật tắt đèn LED RGB

Chúng ta sẽ sử dụng câu lệnh bên dưới để điều khiển đèn RGB:



Hình 4.6: Câu lệnh điều khiển đèn LED RGB

Câu lệnh này có 3 phần tùy chọn có thể thay đổi:

- Chân kết nối: Bạn cần phải chọn đúng chân mà đèn 4 LED RGB đang kết nối với Yolo:Bit. Như trong bài này, chân kết nối là **P0**.
- Vị trí đèn: Trên thiết bị có tất cả 4 đèn, bạn có thể chọn điều khiển riêng lẻ từng đèn, hoặc đơn giản là chọn **tất cả**.
- Màu đèn: Có rất nhiều màu để bạn chọn (lưu ý: màu **đen** được dùng để tắt đèn).

Một chương trình đơn giản để bật tắt đèn RGB sẽ như sau:



Hình 4.7: Chương trình điều khiển đèn RGB đơn giản

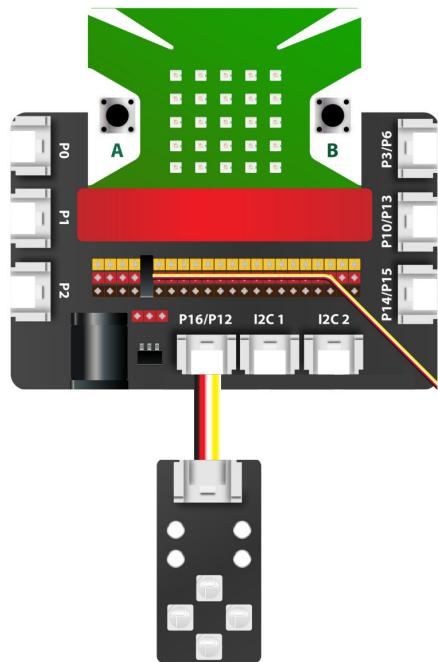
## 6 Khe cắm trên mạch mở rộng

Trên mạch mở rộng của Yolo:Bit có tổng cộng 9 khe cắm khác nhau, được chúng tôi phân loại như sau:

- Khe cắm đa dụng một tín hiệu: P0, P1 và P2.
- Khe cắm hai tín hiệu: P16/12, P14/15, P10/13, P3/6.
- Khe giao tiếp I2C: I2C1 và I2C2, cùng là chân P19/20.

Chúng tôi gọi P0, P1 và P2 là khe cắm đa dụng, bởi nó có thể kết nối được với nhiều dạng thiết bị khác nhau. Trong các bài sau chúng tôi sẽ trình bày chi tiết hơn, đặc biệt là khi nó được dùng để **kết nối với các cảm biến**, vốn là đặc trưng quan trọng cho các ứng dụng thông minh.

Ngược lại, các khe cắm 2 tín hiệu còn lại, kể cả chân I2C, chỉ có thể kết nối được với 1 số thiết bị mà thôi. Đặc biệt là khe I2C, một số thiết bị như cảm biến nhiệt độ độ ẩm DHT20 hay màn hình LCD kí tự, bắt buộc phải cắm vào cổng này.



Hình 4.8: Kết nối đèn RGB vào khe cắm hai tín hiệu P16/P12

Với thiết bị đèn RGB chúng ta đang sử dụng ở bài này, vì nó là thiết bị đơn giản để điều khiển, nên nó có thể cắm vào tất cả các khe cắm trên mạch mở rộng của Yolo:Bit. Khi cắm vào khe cắm có 2 tín hiệu, **chân dùng để lập trình là tín hiệu đầu tiên**. Chẳng hạn như khi cắm vào khe cắm P16/12, chân để điều khiển đèn RGB là P16. Chương trình gợi ý cho trường hợp này sẽ như sau:



Hình 4.9: Đổi đèn RGB sang cổng P16/12

Bạn đọc có thể tiếp tục đổi cổng kết nối với đèn RGB để kiểm tra thử với tất cả các cổng kết nối khả dĩ với nó. Việc kết nối được với nhiều cổng khác nhau, giúp thiết bị mở rộng có nhiều khả năng sử dụng vào trong 1 dự án. Thông thường, với đèn RGB nó sẽ được ưu tiên kết nối thấp nhất trong dự án, để dành sự ưu tiên cho các thiết bị có khả năng kết nối ít hơn với mạch mở rộng.



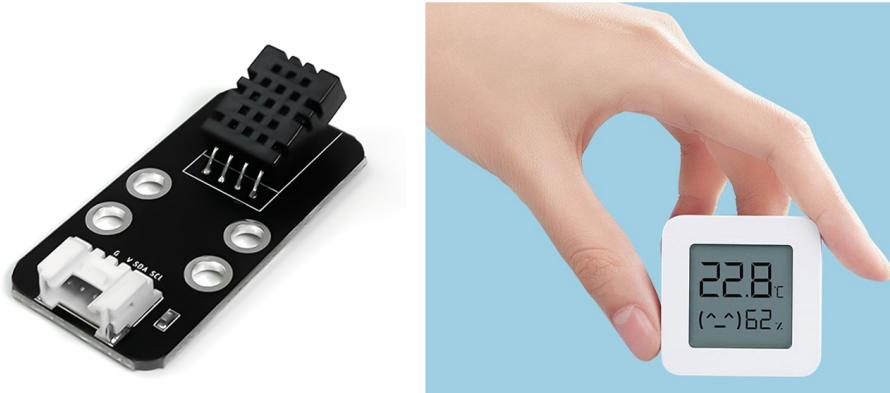
## CHƯƠNG 5

### Cảm Biến Nhiệt Độ - Độ Ẩm DHT20



## 1 Giới thiệu

Trong bài hướng dẫn này, chúng ta sẽ làm việc với một thiết bị khá thông dụng cho các ứng dụng thông minh. Thiết bị này dùng để đo **nhiệt độ và độ ẩm không khí**, có tên gọi là DHT20. Trong một ứng dụng nhàn thông minh của giáo trình này, thông tin về nhiệt độ và độ ẩm phản ánh điều kiện không khí trong nhà, là thông tin quan trọng cho cuộc sống hiện đại.



Hình 5.1: Cảm biến đo nhiệt độ và độ ẩm không khí (DHT20)

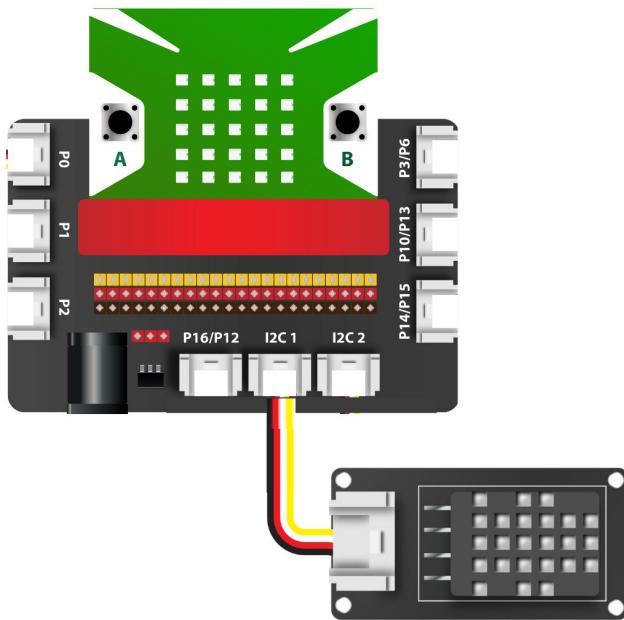
DHT20 là một thiết bị cảm biến rất phổ biến đối với các ứng dụng dân dụng, với ưu điểm là giá thành thấp và cách sử dụng đơn giản. Trong một số trường hợp, DHT20 còn có thể sử dụng cho một số máy sấy thực phẩm. Dãy đo của cảm biến khá phù hợp trong môi trường bình thường không có biến động lớn, với độ ẩm trong khoảng 20 - 90% và nhiệt độ là 0 - 50°C.

Trong bài hướng dẫn này, chúng ta sẽ lấy thông tin về nhiệt độ và độ ẩm từ cảm biến DHT20 và hiển thị nó trên màn hình 25 đèn của Yolo:Bit. Nội dung chính của bài học gồm:

- Kết nối với cảm biến DHT20
- Lập trình lấy dữ liệu từ DHT20
- Hiển thị dữ liệu lên 25 đèn của Yolo:Bit

## 2 Kết nối với DHT20

Khác với các thiết bị mở rộng khác, DHT20 sử dụng một giao thứ đặc biệt để điều khiển, có tên gọi là I2C ( Inter – Integrated Circuit). Đây là một giao thức giao tiếp nối tiếp đồng bộ được phát triển bởi Philips Semiconductors, được sử dụng để truyền nhận dữ liệu giữa các thiết bị điện tử. Do đó, khi kết nối vào mạch mở rộng, chúng ta cần phải kết nối vào **đúng cổng dành cho I2C**. Trên mạch mở rộng Yolo:Bit, chúng ta có 2 cổng I2C, được đặt tên là **I2C1** và **I2C2**. Một thiết bị giao tiếp I2C như DHT20 có thể cắm vào bất kỳ một trong hai cổng này. Trong hình minh họa ở hình bên dưới, DHT20 được kết nối vào cổng **I2C1**.



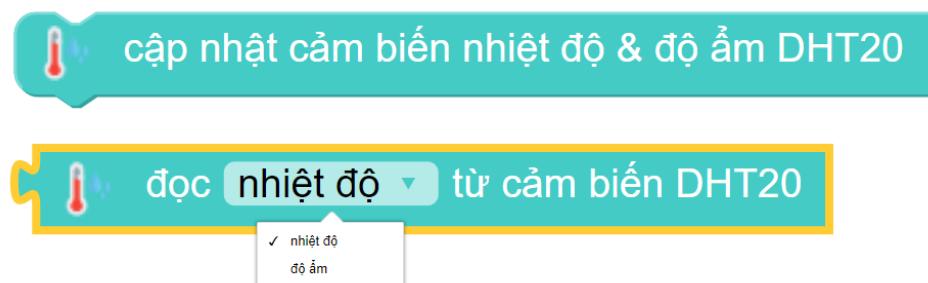
Hình 5.2: Kết nối DHT20 vào mạch mở rộng

Bạn hoàn toàn có thể đổi vị trí cảm của cảm biến DHT20 mà không ảnh hưởng đến hoạt động của hệ thống, do cơ chế của I2C là giao tiếp dựa vào địa chỉ. Đây là nguyên lý giúp cho chân I2C có thể kết nối với 128 thiết bị khác nhau.

Mặc dù vậy, do thiết kế phần cứng của hệ thống, mạch mở rộng Yolo:Bit chỉ hỗ trợ 2 cổng I2C kết nối nhanh, và 1 cổng dự phòng trong tương lai, ở chân **P19** và **P20**.

### 3 Lập trình lấy dữ liệu từ DHT20

Các câu lệnh để lập trình với DHT20 nằm trong nhóm lệnh **HOME\_BIT V3** đã sử dụng ở bài trước. DHT20 là một dạng thiết bị, chỉ khi bạn hỏi nó, thì nó mới trả lời lại thông tin về nhiệt độ và độ ẩm. Vì vậy, quy trình tương tác với cảm biến này gồm 2 bước, như minh họa ở hình bên dưới:



Hình 5.3: Hai câu lệnh chính để làm việc với DHT20

Bạn phải sử dụng câu lệnh **cập nhật cảm biến nhiệt độ & độ ẩm DHT20** trước, sau đó mới có thể truy xuất tới các thông tin về nhiệt độ và độ ẩm. Nếu không sử dụng câu lệnh này, thông tin chúng ta có là thông tin cũ, không phải là giá trị hợp lệ. Chương trình đơn giản để hiện thị thông tin từ cảm biến này được gợi ý như sau:



Hình 5.4: Chương trình đọc thông tin từ DHT20

Trong chương trình này, cứ mỗi 5 giây chúng ta lại cập nhật lại thông tin cảm biến từ DHT20 và hiện nó ra màn hình 25 đèn của Yolo:Bit. Bạn đọc cũng có thể chủ động thêm thông tin đi kèm với 2 giá trị nhiệt độ và độ ẩm để việc hiển thị được sinh động hơn, chẳng hạn như hiển thị thêm đơn vị cho từng giá trị cảm biến. Chương trình gợi ý của chúng tôi cho tính năng này như sau:



Hình 5.5: Chương trình hiển thị thêm đơn vị

Ở đây chúng tôi sử dụng câu lệnh **tạo văn bản từ**, nằm trong mục **NÂNG CAO, CHỮ VIẾT**. Câu lệnh này sẽ tạo ra một chuỗi mới bằng việc ghép 2 chuỗi con. Đơn vị của cảm biến được tạo ra bằng khối lệnh đầu tiên trong phần **CHỮ VIẾT**. Thông tin bây giờ là **kiểu chuỗi**, do đó câu lệnh **hiện chữ** sẽ được dùng để hiển thị kết quả ra màn hình 25 đèn của Yolo:Bit.

Vai trò của 2 cổng kết nối I2C là như nhau, bạn đọc có thể chủ động thay đổi kết nối của cảm biến DHT20 sang cổng thứ 2 và kiểm tra lại chương trình của mình.



# CHƯƠNG 6

## Màn Hình LCD



## 1 Giới thiệu

Trong bài hướng dẫn này, chúng ta sẽ làm việc với một thiết bị dùng cho việc hiển thị, gọi là màn hình LCD. Thiết bị này còn được gọi với tên khác là LCD kí tự, bởi nó chủ yếu được dùng để hiển thị kí tự. Thiết bị mà chúng ta sử dụng có thể hiển thị 2 dòng với mỗi dòng tối đa 16 ký tự, nên còn được gọi là màn hình LCD 16 x 2. Với LCD kí tự, việc hiển thị thông tin sẽ phong phú hơn nếu so với màn hình 25 đèn của Yolo:Bit. Một thông tin dài sẽ trôi qua màn hình của Yolo:Bit và không thuận tiện quan sát trong một số trường hợp. Điểm bất lợi này sẽ được khắc phục hoàn toàn trên LCD kí tự.

Màn hình LCD rất phổ biến trên hệ thống vi điều khiển, trong đó có Yolo:Bit. Tuy nhiên, để điều khiển được thiết bị này, cần phải sử dụng rất nhiều chân kết nối (tối thiểu 3 chân điều khiển và 4 chân tín hiệu). Do vậy, để tối ưu trong việc thiết kế phần cứng, một mạch bổ trợ có tên là I2C đã được thiết kế đi kèm với màn hình LCD.



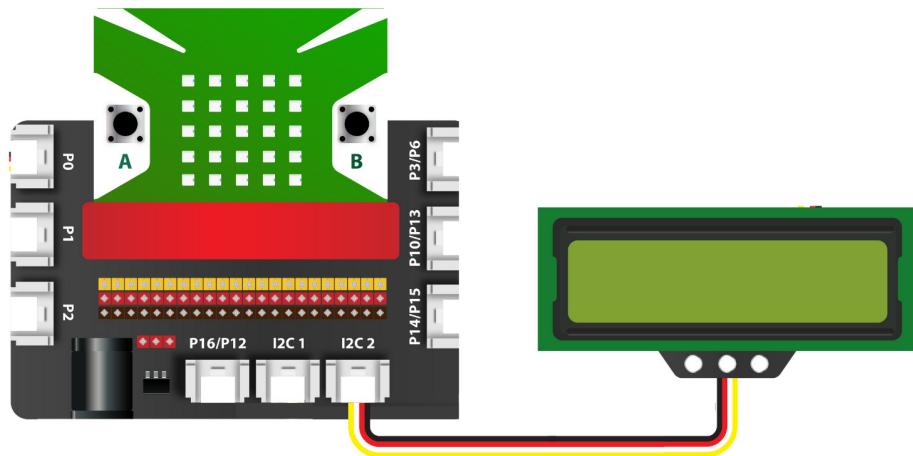
Hình 6.1: Màn hình LCD tích hợp mạch giao tiếp I2C

Với mục tiêu là khảo sát từng thiết bị đơn lẻ, trong bài này, chúng ta sẽ cho hiển thị thông tin từ 2 cảm biến có sẵn trên mạch Yolo:Bit (là nhiệt độ và cường độ ánh sáng). **Các thông tin này sẽ được cập nhật định kì sau mỗi 5 giây**. Qua bài hướng dẫn này, bạn sẽ có thể:

- Kết nối được màn hình LCD với mạch mở rộng
- Hiểu được nguyên lý của giao tiếp I2C
- Viết được chương trình xuất dữ liệu lên LCD

## 2 Kết nối I2C với LCD

Do I2C là một dạng giao tiếp đặc biệt, nên chúng ta chỉ có thể cắm dây nối giữa màn hình LCD và mạch mở rộng tại chân cắm I2C (như hình minh họa). Chúng ta cần lưu ý là trên mạch mở rộng của Yolo:Bit chỉ hỗ trợ tối đa 2 kết nối I2C cùng một lúc.

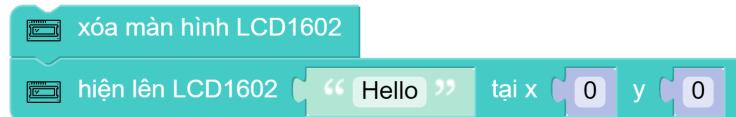


Hình 6.2: Kết nối màn hình LCD vào hệ thống

Một ưu điểm lớn của I2C là nó có thể hỗ trợ kết nối lên đến 128 thiết bị khác nhau, miễn là chúng **khác địa chỉ**. Với khe I2C còn lại trên mạch mở rộng, chúng ta hoàn toàn có thể cắm thêm 1 thiết bị khác, chẳng hạn như cảm biến đo thân nhiệt MLX90614, hay cảm biến nhiệt độ độ ẩm DHT20, phần này sẽ được trình bày ở các bài hướng dẫn tiếp theo.

### 3 Hiển thị dữ liệu trên LCD

Để hiển thị dữ liệu lên màn hình LCD, bạn hãy sử dụng 1 cặp gồm 2 câu lệnh trong nhóm HOME\_BIT V3, như hình:



Hình 6.3: Câu lệnh để hiển thị lên LCD

Câu lệnh đầu tiên dùng để xóa toàn bộ màn hình, trong khi câu lệnh thứ 2 sẽ được dùng để hiển thị thông tin mới lên tại tọa độ x và y:

- x (cột): có giá trị từ **0** **đến** **15** - tương ứng cho 16 cột
- y (hàng): có giá trị từ **0** **đến** **1** - tương ứng cho 2 dòng

Một chương trình đơn giản để hiển thị thông tin trên LCD như sau (lưu ý rằng LCD chỉ hiển thị được tiếng Việt không dấu và không có ký tự đặc biệt):



Hình 6.4: Câu lệnh để hiển thị lên LCD

Mặc dù đã có LCD để hiển thị, trong phần **bắt đầu** của Yolo:Bit, bạn đọc vẫn nên tận dụng 25 đèn để hiện ra thông tin gì đó. Điều này giúp bạn đọc có thể tự kiểm tra xem chương trình mình đang thiết kế, có thực sự được gửi thành công tới mạch YoloBit hay không.

Trong trường hợp dòng chữ **LCD** không hiện ra trên 25 đèn, khả năng lớn là thư viện **HOME\_BIT V3** chưa được tải thành công lên Yolo:Bit. Bạn đọc hãy tải lại thư viện bằng cách vào biểu tượng cài đặt và chọn **Tải thư viện**. Nếu cần thận hơn bạn đọc có thể reset lại mạch sau khi tải thư viện thành công, trước khi kết nối lại với Yolo:Bit và gửi chương trình cho nó.

Trong trường hợp dòng **LCD** đã hiện ra trên 25 đèn, nhưng không thấy xuất hiện chữ trên màn hình LCD, bạn đọc cũng đừng hoang mang và kết luận rằng chương trình bị lỗi. Thông thường, điều này xảy ra do độ tương phản của màu chữ và màu nền chưa hợp lý.

Để khắc phục vấn đề này, bạn có thể sử dụng một tuốc nơ vít nhỏ, vặn một đầu vít có ghi chữ **CONTRAST** (như hình minh họa bên dưới), bạn sẽ thấy chữ dần dần hiện lên.



Hình 6.5: Chỉnh tương phản cho việc hiển thị trên LCD

## 4 Cập nhật thông tin lên LCD

Tận dụng 2 cảm biến có sẵn trên mạch Yolo:Bit, chúng ta có thể hiển thị thông tin về nhiệt độ và ánh sáng trên màn hình LCD. Chương trình gợi ý cho tính năng này như sau:



Hình 6.6: Hiển thị thông tin cảm biến lên LCD

Để làm việc với LCD một cách đơn giản và hiệu quả, bạn đọc hãy xóa hết toàn bộ thông tin trước khi hiển thị thông tin mới. Trong trường hợp muốn hiển thị thông tin đi kèm, bạn đọc có thể linh động sử dụng kết hợp câu lệnh ghép chuỗi, như gợi ý ở chương trình sau đây:



Hình 6.7: Ghép chuỗi khi hiển thị với LCD

Trong chương trình trên, chúng tôi đã sử dụng thêm câu lệnh **tạo văn bản từ**, nằm trong mục **CHỮ VIẾT**. Trong trường hợp bạn chưa thấy nhóm lệnh CHỮ VIẾT xuất hiện bên trái trong môi trường lập trình, bạn hãy Refresh lại trang lập trình (hoặc nhấn tổ hợp phím nóng Ctrl + F5). Khôi lệnh để tạo dòng chữ **NHIET DO:** hay **ANH SANG** được tạo ra từ khối đầu tiên trong mục **CHỮ VIẾT**. Lưu ý rằng, LCD kí tự có thể hiển thị được kí tự **khoảng trắng**. Kí tự này sẽ tạo ra một khoảng trống giữ các thông tin, giúp cho việc đọc thông tin dễ dàng hơn.

## 5 Hiển thị nâng cao

Trong các chương trình đã gợi ý, chúng ta sẽ luôn luôn xóa cả màn hình trước khi hiển thị thông tin mới. Điều này sẽ tạo một khoảnh khắc nhỏ, màn hình LCD không hề hiển thị thông tin nào cả. Trong trường hợp muốn tối ưu, chúng ta chỉ muốn xóa và hiển lại thông tin bị thay đổi mà thôi.

Trong ứng dụng ví dụ ở trên, các dòng chữ **NHIET DO:** và **ANH SANG:** là không thay đổi trong cả ứng dụng của chúng ta. Chỉ thông tin hiển thị đằng sau 2 dòng chữ này mới bị thay đổi. Để tính ra vị trí cần xóa, chúng ta sẽ phải đếm xem dòng chữ cố định có bao nhiêu kí tự. Trong trường hợp này là 10 dành cho cả 2 dòng chữ **NHIET DO:** và **ANH SANG:** (một kí tự khoảng trắng đã được thêm vào sau dấu hai chấm).

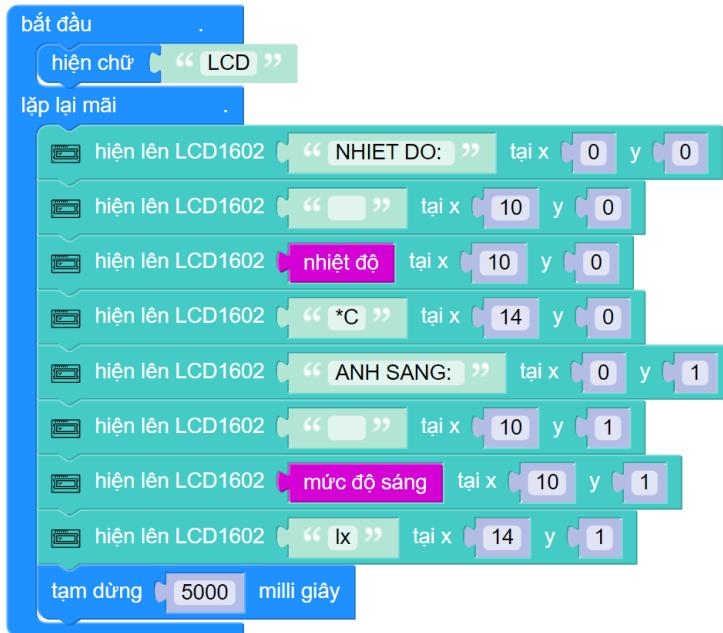
Từ tọa độ 10 trên mỗi dòng, chúng ta sẽ hiển thị ra khoảng trắng để xóa thông tin cũ, trước khi hiển thị thông tin mới ra màn hình. Chương trình gợi ý cho tính năng này sẽ như sau:



Hình 6.8: Hiển thị nâng cao với LCD

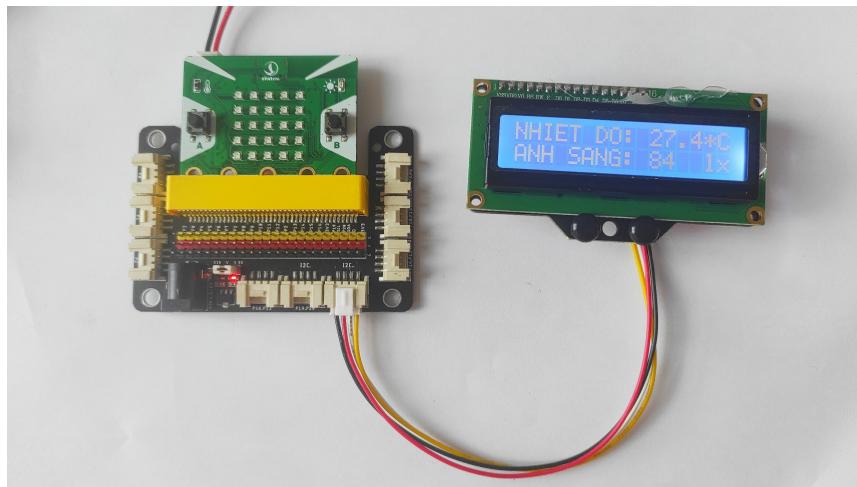
Chúng tôi sẽ dùng **4 kí tự** để hiển thị thông tin về nhiệt độ và ánh sáng. Do đó, trong câu lệnh thứ 2, có 4 kí tự khoảng trắng được sử dụng, nhằm mục đích xóa thông tin cũ trước khi hiển thị thông tin mới ra màn hình LCD. Bạn đọc hãy lưu ý về thông tin các tọa độ của x và y khi tối ưu việc hiển thị trong phần hướng dẫn này.

Bằng cách tính tọa độ, nếu muốn hiển thị thêm đơn vị, bạn sẽ bắt đầu hiển thị kết quả ở tọa độ trục x là 14 (vì 4 kí tự dành cho hiển thị thông tin). Ở vị trí 14, chúng ta cũng chỉ hiển thị thêm được 2 kí tự nữa mà thôi, lý do là LCD chỉ hiển thị được 16 kí tự (đánh dấu từ 0 đến 15). Do đó, chúng tôi chọn đơn vị cho nhiệt độ là \*C và ánh sáng là lx (viết tắt của lux). Chương trình gợi ý như sau:



Hình 6.9: Hiển thị thêm đơn vị

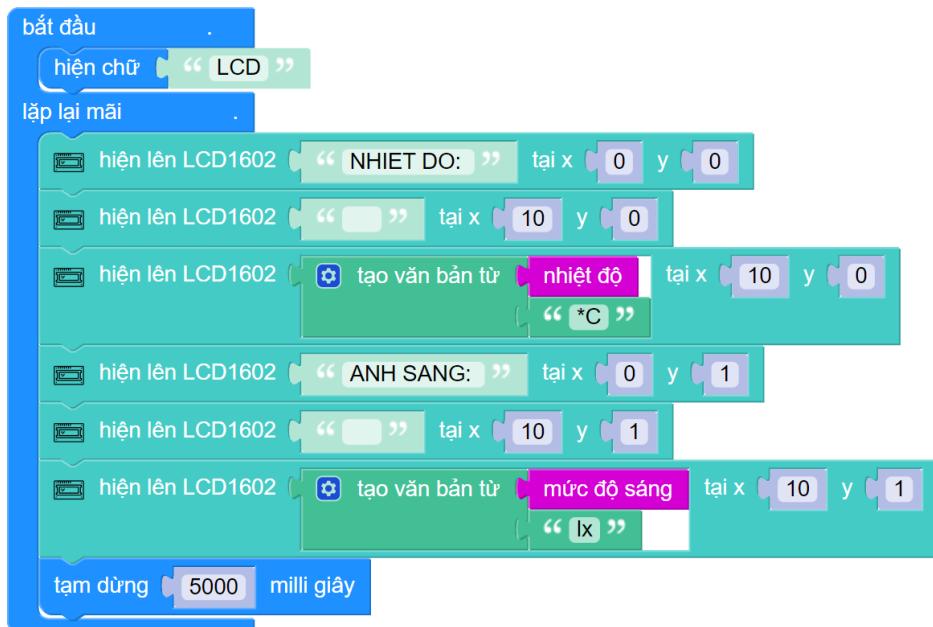
Mặc dù tiêu đề và đơn vị là các thông tin cố định, bạn đọc nên để nó trong phần **lặp mãi mãi** thay vì chỉ thực hiện một lần trong **bắt đầu**. Khi mới khởi động, Yolo:Bit có thể hoạt động không ổn định ở một lệnh nào đó, làm cho việc hiển thị bị lỗi, và điều đó sẽ không được khôi phục lại. Khi để trong lặp mãi mãi, nó sẽ được thực hiện ổn định hơn. Điểm tối ưu trong cách lập trình ở đây, là chúng ta chỉ cần xóa thông tin bị thay đổi (bằng cách hiện ra kí tự khoảng trắng) rồi cập nhật thông tin mới lên mà thôi. Kết quả của chương trình khi chạy trên LCD sẽ như sau:



Hình 6.10: Chương trình khi chạy trên LCD

Trong hướng dẫn ở bài này, chúng ta quy định thông tin về nhiệt độ và cường độ ánh sáng có 4 kí số. Điều này khá hợp lý với nhiệt độ, khi thông thường nó sẽ có 2 kí số, thêm dấu chấm và 1 chữ số thập phân nữa, là 4 kí số. Tuy nhiên, độ sáng lại có số lượng kí số thay đổi do khoảng biến thiên của nó khá rộng. Đôi khi, độ sáng chỉ là số có 2 chữ số, và do đó sẽ có 2 kí tự trống trước đơn vị của nó.

Để việc hiển thị trở nên hoàn hảo hơn, chúng tôi gợi ý một chương trình, với đơn vị được điền sát bên vào giá trị. Tuy nhiên, khi xóa thông tin cũ, chúng ta phải hiển thị 6 khoảng trắng, cho trường hợp giá trị từ cảm biến ánh sáng thực sự có 4 kí số.



Hình 6.11: Hiển thị đơn vị sát vào thông tin

Tác dụng của chương trình này giống như việc canh trái trên màn hình hiển thị LCD. Bạn đọc cần lưu ý là trong câu lệnh dùng để xóa thông tin cũ, **6 kí tự khoảng trắng được sử dụng**. Trong trường hợp muốn canh phải thông tin cảm biến, bạn đọc cần phải xử lý nhiều hơn. Tính năng này chúng tôi không trình bày ở đây và để dành phần thử thách này cho bạn đọc.

# CHƯƠNG 7

## Động cơ - Quạt mini



## 1 Giới thiệu

Quạt là một thiết bị cần thiết cho ứng dụng nhà thông minh. Tích hợp thêm với các tính năng điều khiển tự động hoặc giọng nói, chúng ta sẽ có một ứng dụng hấp dẫn trong nhà. Trong bài hướng dẫn này, chúng tôi sẽ sử dụng thiết bị **Quạt mini** để minh họa cho việc điều khiển một động cơ máy quạt. Với việc sử dụng động cơ, chúng ta có thể điều khiển được tốc độ của quạt tùy theo mục đích của ứng dụng. Hình ảnh của thiết bị này được trình bày như bên dưới.



Hình 7.1: Động cơ - Quạt mini

Thiết bị quạt mini này sử dụng động cơ điện một chiều, hay còn gọi là động cơ DC (Direct Current - Dòng điện một chiều). Không giống như các thiết bị chỉ có nhu cầu bật và tắt (chẳng hạn như bóng đèn), quạt là thiết bị cần phải điều khiển được tốc độ. Trong bài hướng dẫn này, chúng ta sẽ kết nối thiết bị quạt mini vào mạch mở rộng Yolo:Bit và lập trình để điều khiển tốc độ của nó. Các mục tiêu trong bài hướng dẫn này như sau:

- Hiểu được nguyên lý hoạt động của động cơ
- Kết nối quạt mini vào mạch mở rộng Yolo:Bit
- Lập trình điều khiển quạt mini.

## 2 Nguyên lý điều khiển tốc độ cho động cơ

Động cơ mini sử dụng trong thiết bị quạt mini là một dạng động cơ DC. Do đó, để động cơ có thể quay được, nó cần một dòng điện 1 chiều đi qua động cơ. Lúc này, động cơ đóng vai trò như một điện trở. Khi dòng điện đi qua càng lớn, động cơ sẽ quay nhanh hơn và ngược lại.

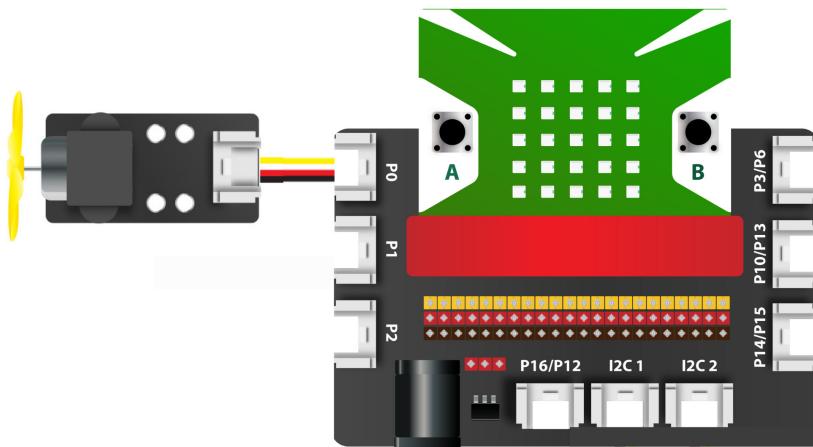
Điều áp hay điều xung, là phương pháp cơ bản để điều khiển tốc độ của động cơ. Nhờ kĩ thuật này, điện thế tại một chân của Yolo:Bit có thể thay đổi được điện áp, từ đó thay đổi cường độ dòng điện qua động cơ và thay đổi tốc độ quay của động cơ. Tín hiệu điều xung còn được gọi là tín hiệu analog, vì nó có thể xuất ra nhiều mức điện áp khác nhau. Điều xung là một kĩ thuật cao cấp trong điều khiển thiết

bị. Lúc đó, tín hiệu tại chân của thiết bị sẽ bật (mức 3.3V) và tắt (mức 0V) với tần suất rất nhanh, tạo ra hiệu ứng điện áp thay đổi ở đầu ra.

Trong một số ứng dụng cao cấp hơn, chúng ta còn có thể điều khiển được chiều quay của động cơ. Tuy nhiên, tín hiệu điều khiển chiều thường chỉ có 2 mức, 0V và 3.3V (điện áp tối đa của một chân điều khiển). Tín hiệu của chân điều khiển chiều thường được gọi là tín hiệu digital, vì nó chỉ có 2 mức ý nghĩa.

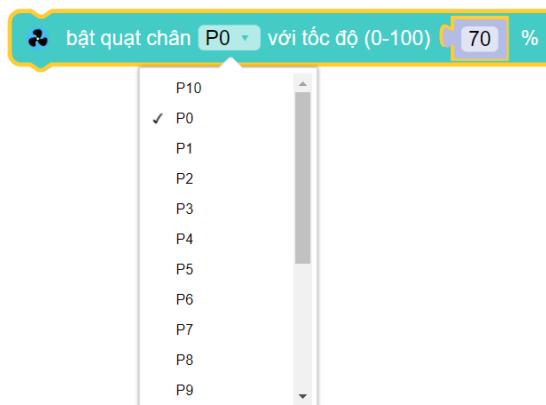
### 3 Kết nối với Yolo:Bit

Thiết bị quạt mini trong hệ sinh thái OhStem chỉ hỗ trợ điều khiển tốc độ và chỉ cần một chân điều khiển. Do đó, bạn có thể cắm nó vào gần như tất cả các cổng trên mạch mở rộng của Yolo:Bit. Chúng ta sẽ bắt đầu thử nghiệm thiết bị này với chân thông dụng nhất, là **P0**, như kết nối ở hình bên dưới.



Hình 7.2: Kết nối quạt mini với mạch mở rộng

Chúng ta có hai câu lệnh để điều khiển tốc độ của quạt. Với câu lệnh thứ nhất, bạn có thể tìm trong nhóm **AIoT KIT**, như trình bày bên dưới:



Hình 7.3: Câu lệnh điều khiển tốc độ của quạt

Câu lệnh này có 2 tùy chọn, bao gồm chân kết nối với thiết bị và tốc độ của nó. Câu lệnh này sẽ dễ sử dụng hơn cho bạn đọc, khi nó hỗ trợ thông tin tốc độ theo đơn vị phần trăm.

Một câu lệnh khác, thường dành cho bạn đọc có nhu cầu tìm hiểu sâu hơn, nằm trong mục **NÂNG CAO**, và chọn tiếp vào **CHÂN CẮM**, như trình bày ở hình bên dưới.



Hình 7.4: Câu lệnh điều xung tại một chân

Trong câu lệnh này, phần đầu tiên dùng để điều khiển, trong khi phần thứ hai sẽ là chân kết nối với thiết bị. Tuy nhiên thông tin về tốc độ là một con số, trải dài từ 0 đến 4095. Một cách nói khác, thông số điều xung có 12 bit ( $2^{12} - 1 = 4095$ ). Khi sử dụng câu lệnh này, bạn đọc phải tự ánh xạ nó sang phần trăm nếu muốn diễn giải theo từ ngữ thông dụng. Việc ánh xạ là tuyến tính với 0 tương ứng với 0% và 4095 tương ứng với 100%. Con số ở giữa, 2045 là khoảng 50%.

## 4 Lập trình điều khiển quạt

Trong phần này, chúng tôi sẽ dùng câu lệnh trong nhóm AIOT KIT để điều khiển quạt bởi nó thân thiện hơn với đa số bạn đọc. Chương trình để bật tắt quạt nối với chân P0 có thể được hiện thực đơn giản như sau:



Hình 7.5: Chương trình bật tắt quạt mini nối với P0

Bạn đọc có thể chủ động thay đổi tốc độ để kiểm tra sự thay đổi của cánh quạt. Cần lưu ý với các tốc độ nhỏ, lực quay sẽ không đủ để làm cánh quạt chuyển động. Đây là ma sát nghỉ thường được trình bày trong môn Vật lý. Với tốc độ lớn hơn 30% hoặc thậm chí lớn hơn, cánh quạt mới bắt đầu quay. Lúc đó lực phát động từ động cơ mới đủ thắng lực ma sát nghỉ của chính nó và làm cánh quạt quay.

Thiết bị quạt mini hoàn toàn có thể kết nối với các cổng 2 chân, và việc đọc chân kết nối cũng giống với đèn RGB đã trình bày ở bài trước. Ví dụ như khi cắm vào cổng P10/P13, chân kết nối sẽ là **P10** và chương trình cần hiệu chỉnh như sau:



Hình 7.6: *Đổi chân kết nối sang P10/13*

Lưu ý rằng, các cổng kết nối I2C (P19/20), có độ tương thích không tốt với thiết bị quạt. Do đó, bạn đọc không nên kết nối nó vào 2 cổng này.



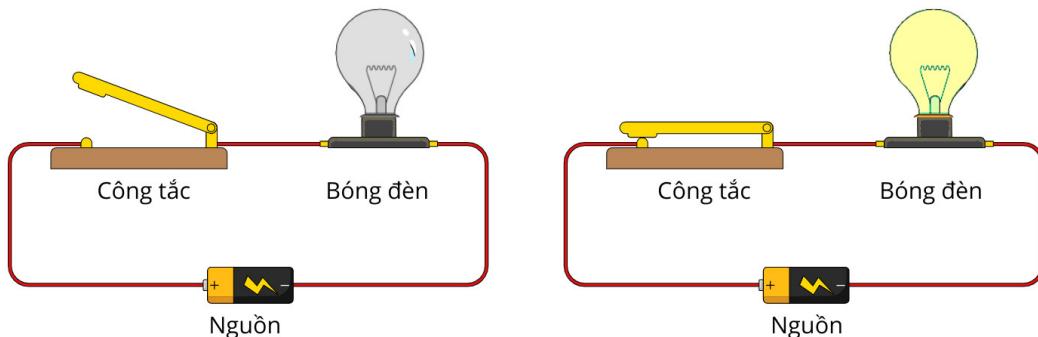
# CHƯƠNG 8

## Công Tắc Relay



# 1 Giới thiệu

Công tắc điện tử, hay còn gọi là Relay, là một thiết bị dùng để đóng tắt nguồn cho một thiết bị. Về nguyên lý, nó hoàn toàn giống với công tắc cơ mà chúng ta đang xài trong dân dụng. Điểm khác biệt ở đây, là nó có thể được điều khiển việc đóng ngắt bằng việc lập trình. Hình minh họa bên dưới là cách sử dụng một công tắc để điều khiển (bật hoặc tắt) một thiết bị điện.



Hình 8.1: Công tắc trong mạch điện

Công tắc, **có 2 chân**, được mắc nối tiếp với thiết bị tiêu thụ điện. Trong ví dụ ở trên, nguồn dương từ pin đi qua công tắc, trước khi đi qua tải, là bóng đèn. Khi công tắc mở (Hình bên trái), dòng điện không đi qua đèn và đèn không sáng. Trong trường hợp còn lại, công tắc đang đóng và có dòng điện đi qua đèn, giúp nó sáng. Rõ ràng, công tắc đang sử dụng rất nhiều trong nhà, để bật tắt đèn hay máy quạt.

Tuy nhiên, trong các ứng dụng thông minh, chúng ta cần một dạng công tắc có thể điều khiển tự động thay vì phải bật tắt bằng tay. Bằng cách sử dụng ngôn ngữ lập trình, chúng ta có một thiết bị có thể điều khiển được, gọi là rơ le (từ chuyên ngành là Relay), có hình ảnh như bên dưới.



Hình 8.2: Công tắc điện tử (Relay)

Trong bài hướng dẫn này, chúng ta sẽ tìm hiểu nguyên lý hoạt động của công tắc điện tử Relay và sử dụng mạch Yolo:Bit để điều khiển nó. Trong tương lai, tính năng này sẽ được dùng để điều khiển một thiết bị tải lớn, chẳng hạn như đèn điện 220V,

vốn được sử dụng nhiều trong các ứng dụng nông nghiệp thông minh để cung cấp thêm ánh sáng quang hợp cho thực vật. Chẳng hạn như việc trồng thanh long trái mùa, nhà vườn sẽ bật thêm đèn vào buổi tối để kích thích thanh long phát triển.

## 2 Nguyên lý hoạt động của Relay

Như được trình bày ở Hình 8.2, mạch điều khiển Relay có 2 đầu kết nối: một đầu dùng để kết nối với mạch lập trình, chẳng hạn như Yolo:Bit, đầu còn lại đóng vai trò là một công tắc để nối với thiết bị cần điều khiển. Tuy nhiên, điểm khác biệt ở đây là **Relay có 3 chân đầu ra**, thay vì chỉ 2 chân như công tắc trong mạch điện ở Hình 8.1. Tên và ý nghĩa của 3 chân này như sau:

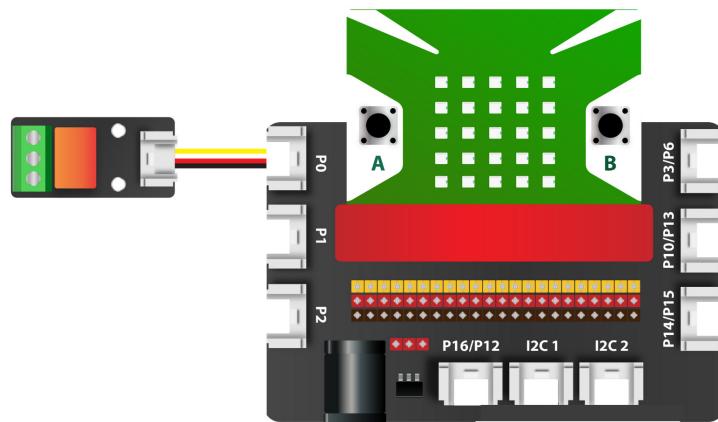
- Chân COM: viết tắt của chữ Common, là chân chung và chỉ nối với 1 trong 2 chân còn lại.
- Chân NC: viết tắt của chữ Normally Closed, là chân thường đóng. Khi không có điện ở phần điều khiển, COM và NC được nối với nhau.
- Chân NO: viết tắt của chữ Normally Open, là chân thường mở. Khi có điện ở phần điều khiển, COM được nối qua NO. Hiển nhiên, lúc này COM và NC không nối với nhau nữa.

Bằng việc lập trình, chúng ta sẽ cung cấp điện từ phần điều khiển. Thông thường chúng ta hay gọi việc này là **kích chân Relay**. Một dòng điện nhỏ từ phần điều khiển sẽ làm cuộn cảm bên trong Relay hoạt động. Lúc này, một lực nam châm sẽ hút chân COM từ NC về NO. Nói cách khác, **châm COM sẽ nối với NO và không nối với NC** nữa.

Như vậy, với 3 chân của Relay, để sử dụng như một công tắc bình thường trình bày ở Hình 8.1, chúng ta chỉ cần dùng 2 chân COM và NO là đủ. Rất ít các ứng dụng cần hành vi ngược lại (chân COM và NC), tức là công tắc sẽ thường xuyên đóng, chỉ khi nào cần kích hoạt thì nó mới mở.

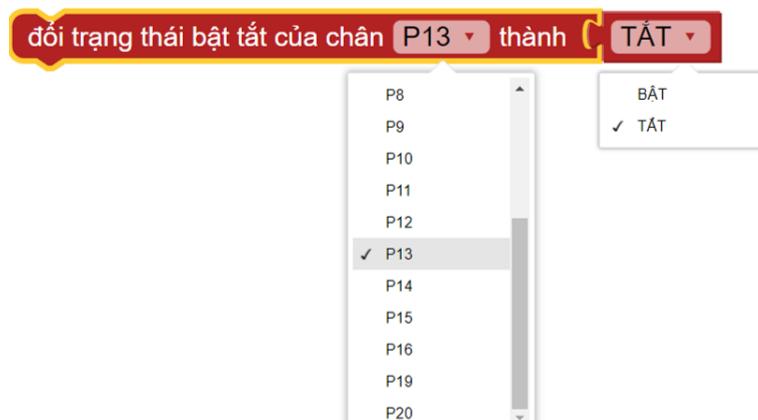
## 3 Kết nối Relay với Yolo:Bit

Cũng như các thiết bị ngoại vi khác, mạch Relay được nối với Yolo:Bit thông qua mạch mở rộng, như minh họa ở Hình 8.3. Nguyên lý điều khiển Relay khá đơn giản, khi nó chỉ cần **một tín hiệu để lập trình**. Cũng như mạch đèn RGB, Relay có thể được cắm vào bất kì khe mở rộng nào của mạch mở rộng. Điều này giúp Relay trở thành một thiết bị đa dụng cho rất nhiều dự án khác nhau với Yolo:Bit. Trong hình minh họa bên dưới, Relay được kết nối vào chân **P0** của Yolo:Bit.



Hình 8.3: Kết nối Relay với Yolo:Bit thông qua mạch mở rộng

Trong trường hợp Relay được cắm vào cổng điều khiển có 2 tín hiệu, chẳng hạn như cổng P3/P6, với nguyên lý xác định chân giống hệt với đèn RGB, chân lập trình để điều khiển nó là P3. Câu lệnh chính để điều khiển Relay được tìm thấy trong mục **NÂNG CAO**, và chọn tiếp vào **CHÂN CẤM**, như trình bày ở hình bên dưới.



Hình 8.4: Câu lệnh bật tắt một chân lập trình

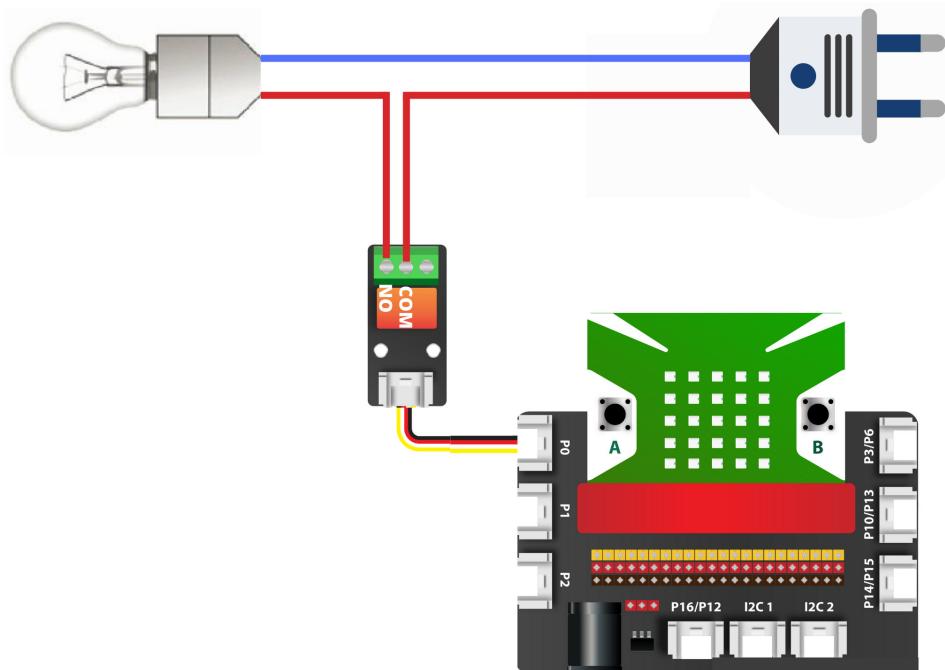
Hoàn toàn giống với hướng dẫn ở bài trước, câu lệnh này có 2 tùy chọn, bao gồm chân điều khiển và trạng thái **BẬT** hay **TẮT**. Khi chọn là **BẬT**, tín hiệu từ chân điều khiển sẽ kích một lực nam châm để bật Relay (COM nối với NO). Với kết nối như Hình 8.3, chương trình đơn giản để bật tắt Relay như sau:



Hình 8.5: Chương trình bật tắt Relay

Trong chương trình trên, chúng tôi cho hiện lên 2 hình ảnh khác nhau (câu lệnh **hiện hình ảnh YES/NO**), tương ứng với 2 trạng thái bật và tắt của Relay. Bạn đọc hãy để ý mỗi khi hình ảnh trên 25 đèn của Yolo:Bit thay đổi, sẽ có một âm thanh nhẹ phát ra. Đó là lúc chân COM thay đổi kết nối với NO hoặc NC. Đây cũng là âm thanh đặc trưng để nhận biết rằng Relay đang hoạt động.

Khi kết nối với tải có nguồn điện lớn hơn, bạn đọc phải hết sức cẩn thận. Chẳng hạn như khi kết nối với đèn 220V, một chân của đèn sẽ được nối vô nguồn. Chân còn lại sẽ được tách ra làm 2, để nối vào COM và NO của Relay, như minh họa ở hình sau đây:



Hình 8.6: Kết nối công tắc và đèn 220V

Tuy nhiên, khi kết nối với các thiết bị sử dụng động cơ 220V như máy bơm hoặc máy quạt công nghiệp, chúng ta phải nối Relay thông qua một công tắc chuyên dụng hơn cho điện 220V, gọi là **khởi động từ** hoặc **công tắc tơ**. Thiết bị này có khả năng khử nhiễu từ trường đối với các thiết bị sử dụng động cơ 220V. Relay chỉ có thể hoạt động được với các thiết bị có công suất nhỏ, trong tầm 100W đến 150W mà thôi. Với các thiết bị điện có công suất lớn hơn, bạn đọc cần tham khảo thêm các kiến thức về điện để sử dụng kết hợp Relay với công tắc tơ. Trong giới hạn của chương trình này, **chúng tôi không khuyến khích việc kết nối với thiết bị có công suất lớn.**

# CHƯƠNG 9

## Cảm biến chuyển động



## 1 Giới thiệu

Trong cuộc sống hiện đại, cảm biến phát hiện có người hay cảm biến chuyển động, được sử dụng rất phổ biến trong các ứng dụng tự động. Dù có biết đèn nó hay không, bạn đã tiếp xúc với nó nhiều hơn bạn nghĩ, chẳng hạn như cửa tự động mở khi đèn ngân hàng, hoặc đèn tự động bật sáng khi có người. Hình ảnh của cảm biến phát hiện chuyển động được trình bày như hình bên dưới.



Hình 9.1: Cảm biến phát hiện chuyển động

Cảm biến phát hiện có người này, còn được gọi với từ ngữ kỹ thuật là cảm biến PIR (Passive Infrared), tức là cảm biến hồng ngoại thụ động. Cơ thể sống nói chung, và con người chúng ta nói riêng, đều phát ra ánh sáng hồng ngoại, vốn không được nhìn thấy bằng mắt thường. Dựa vào ánh sáng này, cùng một số cơ chế đặc biệt trên cảm biến, nó có thể phát hiện ra một người đang di chuyển và kích hoạt các cơ chế điều khiển thông minh nhờ vào việc lập trình, để tạo ra các ứng dụng hiện đại.

Cụ thể hơn, bên trong cảm biến PIR sẽ có 2 bộ phát hiện tia hồng ngoại. Khi có sự di chuyển của vật thể, tín hiệu từ 2 bộ phát hiện tia hồng ngoại này sẽ khác biệt nhau và cảm biến quyết định là có sự chuyển động. Vì vậy, khi không có vật thể hoặc khi vật thể đứng im, cảm biến sẽ trả về là không có sự chuyển động.

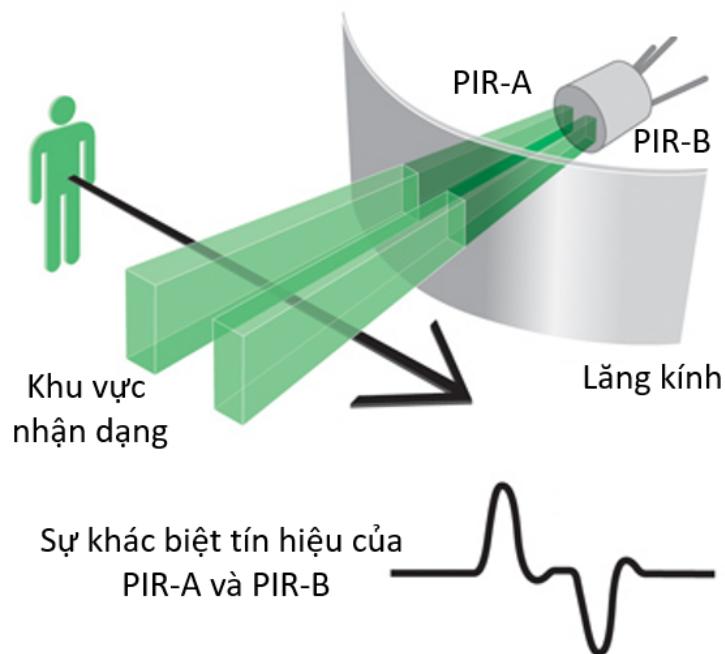
Trong bài hướng dẫn này, chúng ta sẽ tập trung vào các mục tiêu sau đây:

- Nguyên lý hoạt động của cảm biến PIR
- Kết nối cảm biến vật cản với Yolo:Bit
- Lập trình lấy dữ liệu từ cảm biến

## 2 Nguyên lý hoạt động của cảm biến PIR

Bên trong lăng kính của cảm biến PIR, có 2 cảm biến nhỏ để phát hiện tia hồng ngoại. Tín hiệu của 2 cảm biến này sẽ thay đổi khi có tia hồng ngoại hướng tới cảm

biến. Chúng ta gọi 2 cảm biến này là PIR-A và PIR-B, như minh họa ở hình bên dưới.

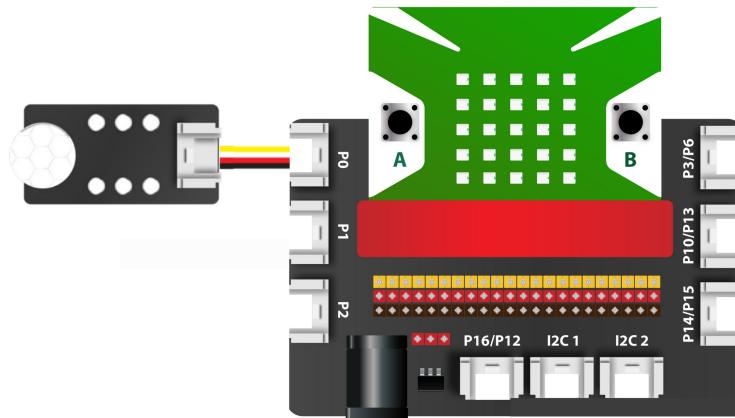


Hình 9.2: Nguyên lý hoạt động của cảm biến PIR

Nguyên lý để phát hiện chuyển động của cảm biến là dựa vào sự khác biệt về tín hiệu của A và B. Khi có 1 người di chuyển ngang qua cả 2 cảm biến, nhờ các góc phủ tín hiệu khác nhau, mà tín hiệu từ A sẽ tích cực trước và B sẽ tích cực sau, trong trường hợp có một người di chuyển từ phía A sang phía B. Cảm biến sẽ tích cực cho đến khi một người di chuyển ra ngoài vùng phủ của cả PIR-A và PIR-B.

### 3 Kết nối với Yolo:Bit

Cảm biến chuyển động là dạng thiết bị đa dụng, khi nó chỉ có 2 trạng thái là có và không. Do đó, nó rất dễ dàng để kết nối với các chân của hệ thống điều khiển nói chung, và Yolo:Bit nói riêng. Ở đây, để kiểm tra dữ liệu từ cảm biến, chúng tôi chọn kết nối vào chân **P0**, như hình vẽ sau đây:



Hình 9.3: Kết nối với cảm biến chuyển động

Khi kết nối cảm biến với các cổng có 2 tín hiệu, cách xác định chân lập trình của cảm biến chuyển động hoàn toàn giống với đèn RGB hoặc Relay ở các bài hướng dẫn trước. Chẳng hạn, khi kết nối cảm biến vào cổng **P16/P12**, chân lập trình của nó sẽ là **P16**.

## 4 Lập trình với cảm biến chuyển động

Khối lệnh chính để tương tác với cảm biến chuyển động có thể được tìm thấy trong nhóm AIOT-KIT, như trình bày ở hình bên dưới:

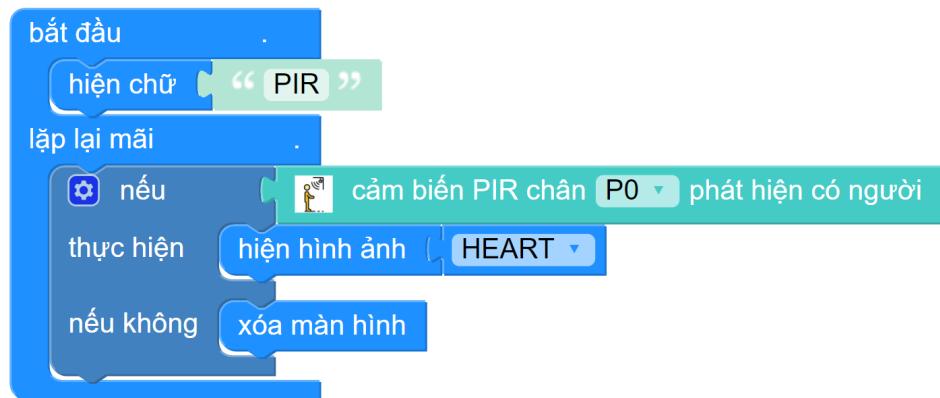


Hình 9.4: Khối lệnh lập trình cho cảm biến chuyển động

Với khối lệnh này, chúng ta cần lưu ý 2 thông tin quan trọng sau đây:

- Chân kết nối, hay chân lập trình, cần phải được lựa chọn cho chính xác.
- Khối lệnh này mang ý nghĩa đúng hoặc sai, nên nó phải được dùng chung với khối lệnh khác, thông dụng nhất là **nếu ... nếu không**

Một chương trình đơn giản để thử nghiệm với cảm biến chuyển động được trình bày như sau:



Hình 9.5: Chương trình cho cảm biến chuyển động

Khi sử dụng cảm biến phát hiện chuyển động, bạn có thể huơ tay trước cảm biến để thấy đèn trên mạch Yolo:Bit thay đổi ra sao. Bạn cũng có thể thử đổi khe cắm qua cổng P16/P12 và thay đổi chương trình cho phù hợp. Lúc này, chân lập trình dành cho cảm biến PIR sẽ là **P16**.



# CHƯƠNG 10

## Cảm Biến Ánh Sáng



# 1 Giới thiệu

Trong ngữ cảnh nhà thông minh, điều kiện ánh sáng là một thông tin hữu ích và cần theo dõi. Mặc dù trên mạch Yolo:Bit đã có tích hợp sẵn một cảm biến ánh sáng, việc sử dụng cảm biến ánh sáng rời sẽ dễ dàng hơn cho việc lắp đặt và kết quả đo đạc cũng sẽ chính xác hơn. Cảm biến ánh sáng sử dụng trong bài hướng dẫn này được trình bày như hình bên dưới.



Hình 10.1: Cảm biến ánh sáng

Một trong những đặc tính quan trọng của cảm biến sử dụng trong bài này là đầu ra của cảm biến là một dạng điện áp. Cụ thể hơn, nó dựa trên nguyên lý trở kháng để phản ánh thông số cần đo đạc. Với ánh sáng, đó là quang trở - thiết bị có điện trở thay đổi khi cường độ ánh sáng thay đổi. Nguyên lý này được áp dụng phổ biến cho nhiều cảm biến khác nhau, chẳng hạn như nhiệt điện trở cho cảm biến nhiệt độ hay như độ ẩm đất.

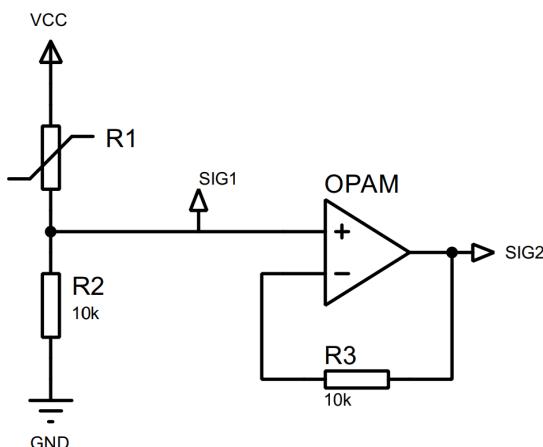
Nói chung, các cảm biến dựa trên nguyên lý trở kháng khá đơn giản về thiết kế mạch điện lẫn việc lập trình cho nó. Do tính chất đơn giản của nó, các hệ thống cảm biến cho các dự án nói chung, và đặc thù cho mục đích giáo dục, đều có hỗ trợ chuẩn đầu ra là điện áp. Thậm chí những cảm biến có độ phức tạp cao, vẫn có hỗ trợ chuẩn đầu ra điện áp, chẳng hạn như hệ thống cảm biến Analog Gravity từ DFRobot, từ các cảm biến đo chất lượng nước cho đến các cảm biến liên quan đến không khí. Các mục tiêu chính trong bài hướng dẫn này như sau:

- Nguyên lý hoạt động của cảm biến dựa vào trở kháng.
- Kết nối cảm biến ánh sáng với Yolo:Bit
- Lập trình lấy dữ liệu từ cảm biến ánh sáng

Trong hệ sinh thái của OhStem, bên cạnh cảm biến ánh sáng, các cảm biến khác như độ ẩm đất, cảm biến âm thanh, các loại cảm biến khí gas đều dựa trên nguyên lý trở kháng này.

## 2 Nguyên lý hoạt động của cảm biến trở kháng

Khác với cảm biến tích hợp chẳng hạn như DHT20, thường hỗ trợ nhiều thông tin trong một cảm biến, cảm biến dựa vào trở kháng chỉ hỗ trợ đo một thông số duy nhất. Tuy nhiên, đầu ra của cảm biến đơn giản là một mức điện áp. Nhờ vậy, chúng ta 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 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 4095 (giá trị 12 bit), khi lập trình trên mạch Yolo:Bit. 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 xuông đất. Ở đây, điện trở R1 tương trung 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:

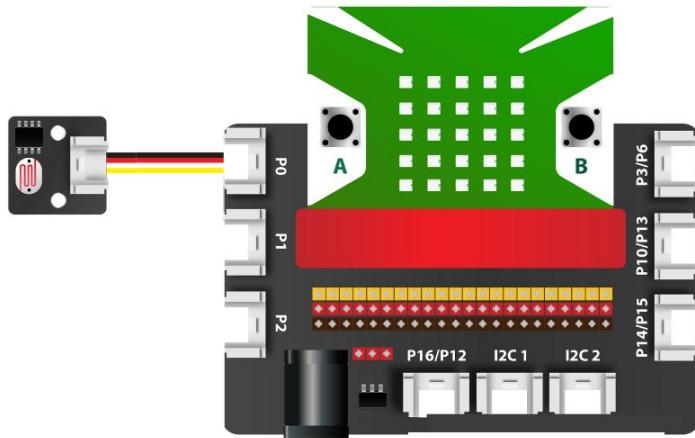
$$SIG1 = \frac{VCC * R2}{R1 + R2}$$

Rõ ràng, khi R1 thay đổi, SIG1 sẽ thay đổi theo. Ví dụ như cảm biến ánh sáng ở bài này, khi ánh sáng yếu 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 cường độ ánh sáng cao, 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.

Thông thường, 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 Yolo:Bit. Do đó, mạch Yolo:Bit sẽ rất bền và không bị nóng trong quá trình hoạt động.

### 3 Kết nối với Yolo:Bit

Chỉ một số chân trên Yolo:Bit mới có khả năng kết nối được với cảm biến có đầu ra là điện áp. Hiện tại, với mạch mở rộng, chỉ các chân **P0, P1 và P2** mới có thể kết nối được với cảm biến ánh sáng. Theo như kết nối ở hình dưới đây, cảm biến ánh sáng được nối với chân P0.



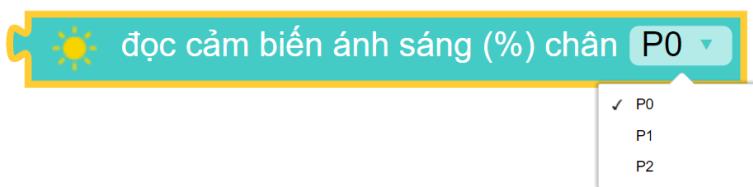
Hình 10.3: Kết nối cảm biến với Yolo:Bit

Để đọc dữ liệu từ cảm biến, chúng ta có 2 cách thực hiện: sử dụng câu lệnh hỗ trợ trong nhóm AIOT-KIT hoặc tự hiện thực bằng câu lệnh đọc giá trị analog trực tiếp từ chân của Yolo:Bit. Trong hướng dẫn của giáo trình này, chúng tôi sẽ trình bày cả 2 phương pháp để bạn đọc có cái nhìn đầy đủ nhất về cảm biến điện áp.

### 4 Lập trình đọc độ ẩm từ cảm biến

#### 4.1 Sử dụng thư viện AIOT-KIT

Trong thư viện AIOT-KIT, khôi lệnh sau đây sẽ được sử dụng để đọc giá trị từ cảm biến ánh sáng:



Hình 10.4: Câu lệnh đọc cảm biến ánh sáng trong AIOT-KIT

Trong câu lệnh này, chúng ta cần phải chỉ định rõ chân lập trình cho cảm biến. Ngoài ra, kết quả nhận được từ khôi lệnh này là một con số với đơn vị là phần trăm. Chương trình đơn giản sau đây sẽ hiển thị giá trị từ cảm biến ánh sáng mỗi 5 giây.



Hình 10.5: Đọc cảm biến ánh sáng mỗi 5 giây

## 4.2 Đọc điện áp trực tiếp

Trong trường hợp bạn muốn đọc trực tiếp giá trị điện áp, câu lệnh **đọc giá trị analog của chân**, như trình bày ở hình bên dưới, sẽ được sử dụng. Câu lệnh này được lấy trong mục **NÂNG CAO**, và chọn tiếp trong nhóm **CHÂN CẮM**.



Hình 10.6: Câu lệnh đọc dữ liệu điện áp từ cảm biến

Một chương trình đơn giản để hiển thị giá trị đọc được từ cảm biến có đầu ra là điện áp, đang được nối với chân P0, được hiện thực như sau:



Hình 10.7: Chương trình đọc dữ liệu từ cảm biến

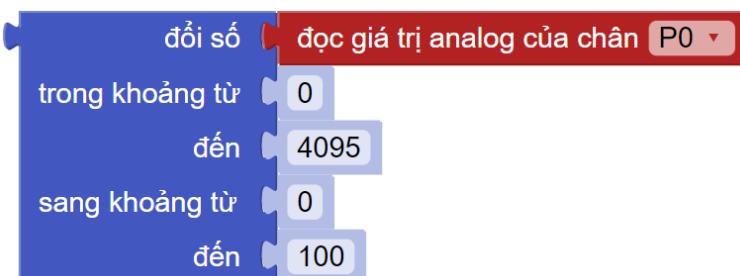
Mỗi 5 giây, giá trị của cảm biến sẽ được hiện ra màn hình 25 đèn của Yolo:Bit. Bạn đọc có thể kiểm tra sự thay đổi của cảm biến bằng cách thay đổi cường độ ánh sáng chiếu vào nó bằng cách dùng tay để che cảm biến lại.

### 4.3 Chuyển đổi giá trị cho cảm biến

Trong chương trình ở trên, giá trị của chúng ta nhận được là một con số, có tầm giá trị từ 0 đến 4095. Điều này được giải thích rằng, giá trị điện áp tại chân P0 (có tầm từ 0V đến 3.3V) sẽ được phân giải thành một con số 12 bit. Việc phân giải này là tuyến tính, nghĩa là 0V sẽ tương ứng với 0 và 3.3V sẽ tương ứng với số lớn nhất có 12 bit, là 4095 ( $2^{12} - 1 = 4095$ ). Bằng cách chuyển đổi tuyến tính, chúng ta sẽ có mức điện áp 1.65V sẽ có giá trị là một nửa của 4095, tầm 2048.

Tuy nhiên, như chúng ta đã biết, cường độ ánh sáng có thể diễn giải với đơn vị là phần trăm (%) và có tầm giá trị từ 0 đến 100 mà thôi. Điều quan trọng, và cũng là hạn chế của cảm biến ADC, là giá trị của nó không tuyến tính với mức điện áp. Do đặc điểm này, cảm biến ADC thường có độ chính xác vừa phải và giá thành cũng rẻ hơn so với các cảm biến sử dụng các công nghệ cao cấp hơn.

Mặc dù vậy, trong một số ứng dụng, chẳng hạn như giám sát ánh sáng trong nhà thông minh, chúng ta cũng không cần phải đo đạc quá chính xác. Giả sử rằng, giá trị điện áp cũng tỉ lệ tuyến tính với thông tin về độ ẩm. Theo tính chất bắt đầu, con số có giá trị từ 0 đến 4095 sẽ tuyến tính với tầm giá trị của độ ẩm, từ 0 đến 100. Với giả sử này, môi trường lập trình hỗ trợ cho chúng ta câu lệnh sau đây để chuyển đổi:



Hình 10.8: Câu lệnh chuyển đổi tuyến tính

Câu lệnh này sẽ có thông số đầu vào, nằm ở mục **đổi số**. Hai thông số tiếp theo là tầm trị của đầu vào và 2 thông số cuối cùng, là khoảng giá trị cần chuyển đổi. Trong ví dụ ở Hình 10.8, chúng ta đang đổi giá trị từ cảm biến độ ẩm đất (nối với chân P0) sang thông tin có đơn vị phần trăm. Chương trình minh họa cho việc chuyển đổi và hiện thị kết quả như sau:



Hình 10.9: Chương trình chuyển đổi kết quả tính toán

Trong chương trình trên, cứ mỗi 5 giây, dữ liệu thô sẽ được in ra. Sau đó, giá trị chuyển đổi của độ ẩm đất (với đơn vị là %) sẽ được in ra. Bạn đọc có thể tự tìm hiểu thêm về một đơn vị khác cho cảm biến ánh sáng là **lux** và xây dựng một công thức khác cho nó trong việc chuyển đổi giá trị điện áp sang lux.



# CHƯƠNG 11

## Remote Và Mắt Nhận Hồng Ngoại



## 1 Giới thiệu

Trong trường hợp muốn tăng thông tin tương tác với người dùng, bạn đọc có thể tích hợp thêm remote hồng ngoại. Trên remote có sẵn nhiều nút nhấn, bao gồm cả chữ và số. Đây là công cụ mở rộng hữu dụng trên Yolo:Bit nhằm tăng số lượng nút nhấn trong một ứng dụng. Bàn phím số này gồm có 2 phần: Remote điều khiển từ xa và mắt nhận hồng ngoại. Hình ảnh các thiết bị sử dụng trong bài hướng dẫn này được trình bày như hình bên dưới:



Hình 11.1: Remote và mắt nhận hồng ngoại

Nguyên lý hoạt động của nó cũng giống với các remote trong dân dụng, như remote tivi hoặc máy lạnh chẳng hạn. Khi sử dụng remote, bạn phải hướng nó vào mắt nhận thì việc kết nối ổn định và chính xác. Sóng hồng ngoại là sóng có hướng và có độ phản xạ tương đối thấp. Do đó, khi không hướng trực tiếp vào mắt nhận, xác suất nhận được tín hiệu là tương đối thấp.

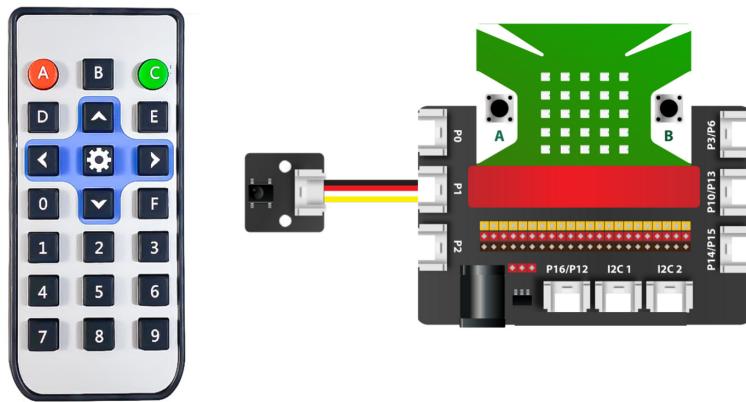
Bạn cần lưu ý lấy đúng mắt nhận hồng ngoại, bởi nó hơi giống với cảm biến ánh sáng. Đối với remote, bạn cần loại bỏ tấm chắn pin bằng plastic để pin có thể tiếp xúc với điện cực, lúc đó remote mới hoạt động. Khi sản xuất, tấm chắn plastic này hạn chế việc chạm phím khi vận chuyển, tránh hao pin hoặc gây hư hỏng remote điều khiển.

Các nội dung trong bài hướng dẫn này được tóm lược như sau:

- Kết nối mắt nhận hồng ngoại và mạch mở rộng
- Lập trình nhận dữ liệu từ Remote hồng ngoại
- Bật tắt đèn bằng remote

## 2 Kết nối với mắt nhận hồng ngoại

Để sử dụng bàn phím số hồng ngoại này, chúng ta cần kết nối mắt nhận hồng ngoại vào mạch mở rộng. Với gợi ý của chúng tôi, bạn đọc có thể kết nối nó vào cổng P1, như minh họa ở hình bên dưới:

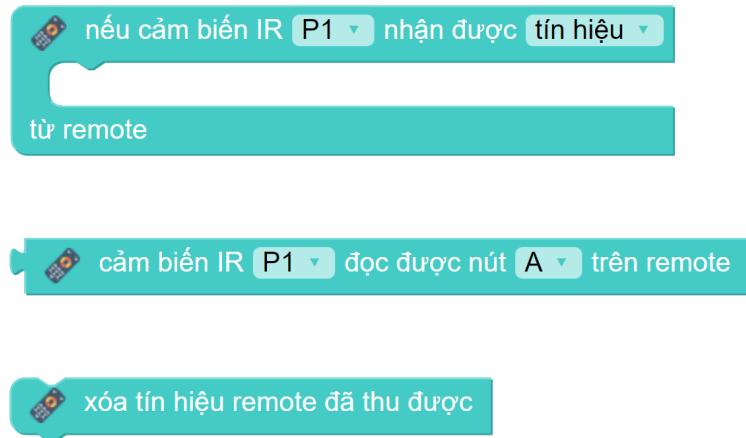


Hình 11.2: Kết nối mắt đọc hồng ngoại vào hệ thống

Mắt nhận hồng ngoại là một thiết bị đa dụng. Về cơ bản, nó có thể kết nối với đa số các cổng trên mạch mở rộng. Tuy nhiên, chúng ta nên hạn chế việc kết nối thiết bị vào 2 cổng I2C, vốn dành riêng cho các thiết bị chuyên dụng. Khi kết nối vào cổng 2 tín hiệu, việc đọc chân lập trình cho mắt hồng ngoại cũng tương tự như đèn RGB. Tức là nếu kết nối mắt hồng ngoại vào cổng P10/P13, chân lập trình cho nó sẽ là P10.

### 3 Lập trình nhận dữ liệu từ remote

Để có thể làm việc với remote và mắt nhận hồng ngoại, chúng ta cần 3 khối lệnh trong nhóm AIOT-KIT, như sau:

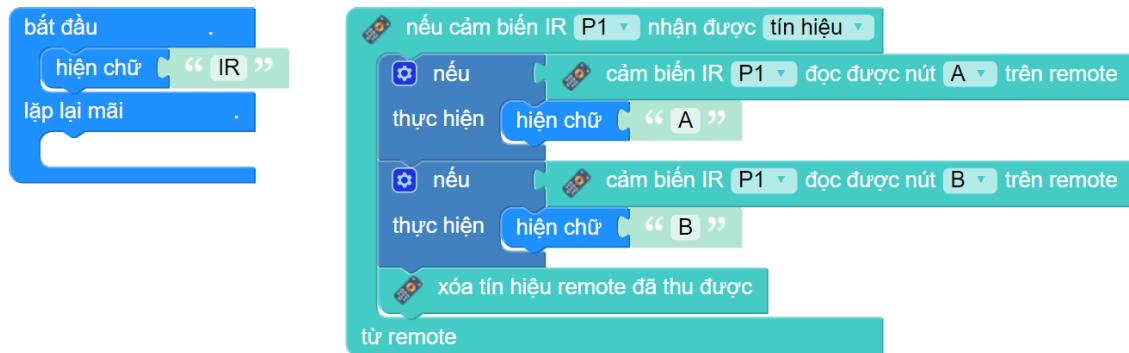


Hình 11.3: Các khối lệnh cần cho việc lập trình với Remote hồng ngoại

Với khối lệnh đầu tiên, nó là một dạng khối lệnh sự kiện. Khi nhận được tín hiệu từ một nút trên remote, chương trình sẽ tự động chạy các câu lệnh sẽ được hiện thực bên trong nó. Vai trò của khối lệnh này khá giống với khối lệnh nhấn nút A và nút B trên mạch Yolo:Bit.

Với khối lệnh thứ 2, chúng ta sẽ sử dụng kèm nó với câu lệnh nếu, để kiểm tra xem phím nào được nhấn, trước khi có thể thực hiện các tác vụ tương ứng.

Cuối cùng, khôi lệnh thứ 3 dùng để xóa dữ liệu của nút nhấn trong bộ nhớ của hệ thống sau khi đã xử lý xong, trước khi có thể nhận được tín hiệu mới từ remote hồng ngoại. Chương trình đơn giản để nhận tín hiệu từ remote được trình bày như sau:



Hình 11.4: Chương trình nhận và xử lý lệnh từ remote

Để kiểm tra xem dữ liệu nhận được từ remote là gì, cổng kết nối và tên của nút nhấn sẽ được lựa chọn và sử dụng kết hợp với câu lệnh **nếu**. Cuối cùng, sau khi đã xử lý xong cho tín hiệu từ remote, nó sẽ được xóa đi bằng khôi lệnh **xóa tín hiệu remote đã thu được**.

## 4 Bật tắt đèn bằng remote

Chúng ta sẽ sử dụng remote hiện có để làm một ứng dụng, là bật tắt đèn bằng màn hình 25 đèn của Yolo:Bit. Ở đây, nút A sẽ được sử dụng để hiện thực tính năng này. Khi đèn đang tắt, nhấn nút A nó sẽ sáng. Ngược lại, khi đèn đang sáng, nhấn nút A nó sẽ tắt đi. Đây là tính năng khá phổ biến trong việc điều khiển thiết bị trong dự án nhà thông minh trong tương lai.

Để có thể hiện thực tính năng này, chúng ta sẽ cần thêm một biến số, đặt tên là **status**. Khi biến này bằng 0, đèn đang tắt và khi bằng 1, đèn đang sáng. Dựa vào giá trị của biến này, chúng ta sẽ thực hiện các tác vụ tương ứng khi nhận được tín hiệu từ nút A.

Các bước chi tiết để tạo ra biến **status** được minh họa như hướng dẫn bên dưới. Từ mục **BIẾN**, nhấn vào nút **Tạo biến...**, đặt tên và nhấn nút **LƯU**.



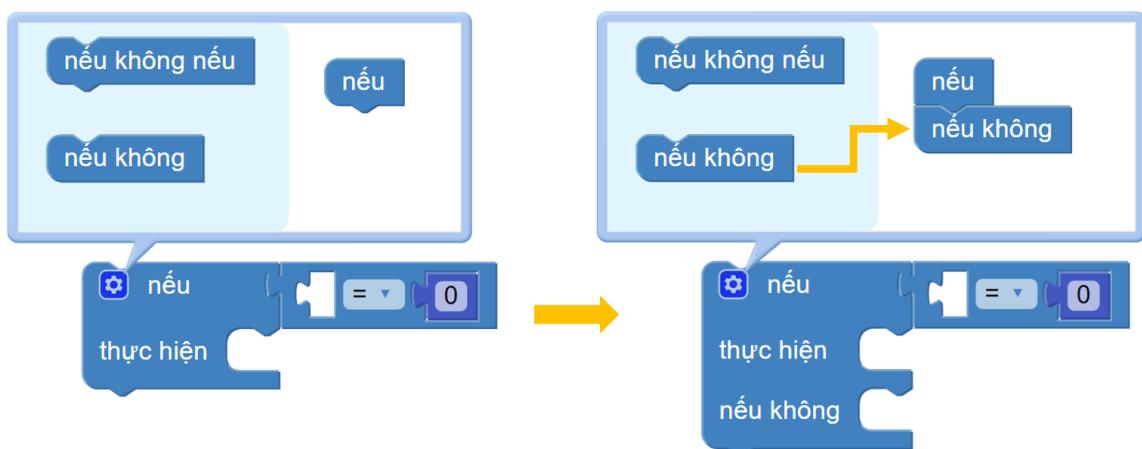
Hình 11.5: Tạo biến số status

Sau khi đã tạo xong biến status, chúng ta bắt đầu hiện thực chương trình, bao gồm việc khởi tạo giá trị cho nó ở phần bắt đầu. Giá trị 0 cho biến status có thể được tìm thấy trong nhóm lệnh **TÍNH TOÁN**, như sau:



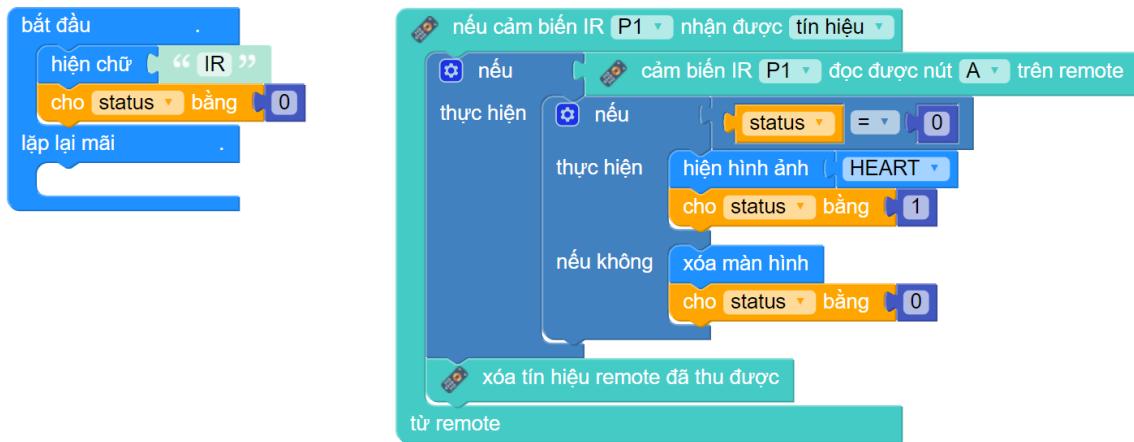
Hình 11.6: Khởi tạo giá trị cho biến status

Bước tiếp theo, chúng ta cần xây dựng câu lệnh nếu, để kiểm tra giá trị của biến status trước khi bật tắt đèn một cách tương ứng. Với khôi lệnh đầu tiên trong mục **LOGIC**, nhấn vào biểu tượng cài đặt của nó, để xây dựng thêm phần **nếu không**, như minh họa ở hình bên dưới:



Hình 11.7: Xây dựng khôi lệnh kiểm tra điều kiện

Mảnh ghép cho việc so sánh bằng sẽ là biến status đã tạo. Chương trình hoàn thiện của chúng ta sẽ như sau:



Hình 11.8: Chương trình bật tắt đèn bằng nút A trên remote

Sau khi bật tắt đèn, chúng ta cũng đồng thời gán giá trị cho biến status. Đây là câu lệnh hết sức quan trọng để đèn đảo trạng thái mỗi khi nhấn nút A trên remote.

# CHƯƠNG 12

## Động cơ RC Servo



# 1 Giới thiệu

Bên cạnh việc sử dụng đèn để hiển thị, các cơ cấu chuyển động của động cơ sẽ tạo nhiều hứng thú và là công cụ tốt cho các sản phẩm STEM. Tuy nhiên, khác với động cơ sử dụng cho máy quạt mini, RC Servo là một dạng động cơ được điều khiển theo góc. Do đó, nó khá thích hợp để xây dựng các ứng dụng cần sự điều khiển chính xác và có điểm dừng, chẳng hạn điều khiển đóng và mở cửa trong ứng dụng nhà thông minh.

Hoàn toàn trái ngược với động cơ cánh quạt, RC Servo có tốc độ thấp nhưng có lực khá mạnh. Bên trong cấu tạo của RC Servo thường đi kèm với bộ hộp số giảm tốc có tỉ số truyền lớn. Do vậy, chỉ cần một lực nhỏ ở phía động cơ, có thể truyền động ra một lực lớn ở cơ cấu truyền động. Hình ảnh của động cơ RC Servo được minh họa ở hình bên dưới:



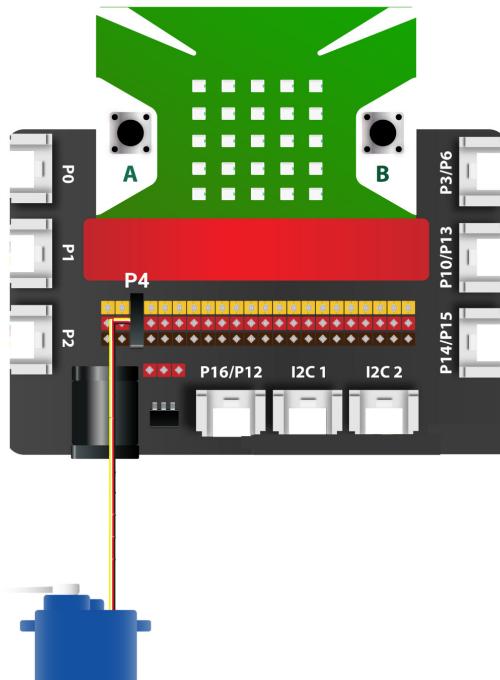
Hình 12.1: Động cơ RC Servo

Lưu ý là động cơ RC Servo vẫn có loại có thể xoay vòng tròn, thường được gọi là RC Servo 360 độ. Đối với các ứng dụng cần điều khiển theo góc hoặc đóng mở một cơ cấu, chúng ta sẽ xài động cơ RC Servo 180 độ. Trong bài hướng dẫn này, chúng ta sẽ tập trung vào các nội dung sau đây:

- Kết nối động cơ với mạch mở rộng
- Lập trình điều khiển động cơ
- Thủ nghiệm các chân kết nối khác

## 2 Kết nối với động cơ

Trên mạch mở rộng của Yolo:Bit hỗ trợ hàng chân cắm chuyên dụng cho việc tích hợp động cơ RC Servo, với 3 chân kết nối cơ bản là Nguồn - Đất và chân tín hiệu, hay còn gọi là chân lập trình, như minh họa ở hình bên dưới:



Hình 12.2: Kết nối động cơ với mạch mở rộng

Động cơ RC Servo được thiết kế khoa học, giúp bạn đọc dễ dàng sử dụng. Chúng có 3 chân là Nguồn (đỏ), đất (xám) và tín hiệu (vàng). Vì chân nguồn nằm ở giữa, nên việc cắm nhầm vào hàng 3 chân của mạch mở rộng cũng không làm hư hỏng động cơ. Tuy nhiên, **bạn đọc hãy lưu ý cắm chân màu vàng của động cơ vào chân màu vàng trên mạch mở rộng**. Lúc đó, chân màu xám của động cơ sẽ kết nối được với nguồn âm của mạch mở rộng (chân màu đen). Với kết nối như gợi ý ở trên, chân lập trình dùng để điều khiển động cơ RC Servo sẽ là **P4**.

## 3 Lập trình điều khiển động cơ

Các khối lệnh điều khiển động cơ RC Servo sẽ cần được lấy trong nhóm **NÂNG CAO**, và chọn tiếp vào **CHÂN CẮM**. Hai khối lệnh chính cho việc điều khiển động cơ được trình bày như hình ảnh bên dưới:

quay servo chân P4 ▾ đến góc (0-180 độ) 90

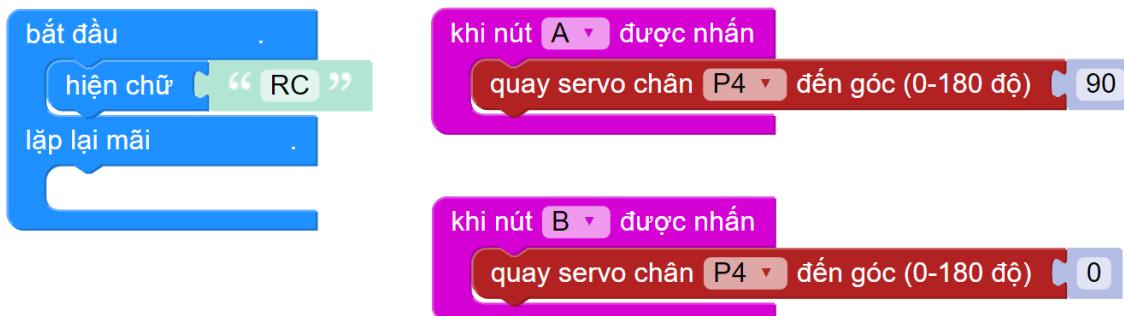
tắt điều khiển servo chân P4 ▾

Hình 12.3: Các khối lệnh điều khiển RC Servo

Khi sử dụng khối lệnh đầu tiên, một tín hiệu điều khiển sẽ gửi đến động cơ và giúp nó quay đến 1 góc chỉ định (chẳng hạn như 90 độ). Tín hiệu này sẽ liên tục duy trì và giúp cho động cơ ổn định ở góc xoay này. Chúng ta hay gọi trường hợp này là ghim xung cho động cơ.

Tuy nhiên, khi ghi xung liên tục, động cơ sẽ liên tục hoạt động và làm tốn điện tiêu thụ. Do đó, trong một số trường hợp, sau một khoảng thời gian ngắn (khoảng 1 đến 2 giây), chúng ta có thể tắt việc ghim xung này bằng khối lệnh thứ 2.

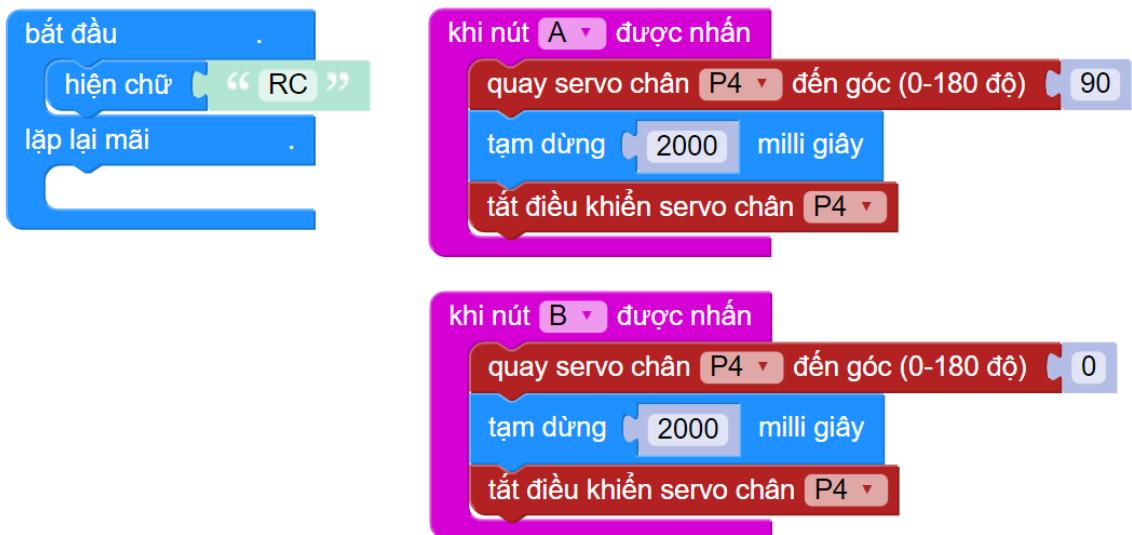
Khác với các chương trình kiểm thử thiết bị đã trình bày, chương trình để tương tác với động cơ RC Servo nên sử dụng kết hợp với 2 nút nhấn A và B, như gợi ý sau đây:



Hình 12.4: Chương trình kiểm tra góc xoay của động cơ

Chương trình trên là rất cần thiết để cân chỉnh động cơ về lại góc 0 của nó (mỗi khi nhấn nút B). Thông thường, động cơ sẽ được nối với 1 hệ thống cơ khí trong mô hình của một sản phẩm. Chính vì vậy, việc thao tác cơ khí và lập trình trong tương lai sẽ thuận tiện hơn khi động cơ đang ở góc 0 độ.

Một chương trình khác sau đây, minh họa cho việc tắt ghim xung ở động cơ sau khi nó quay đến góc mong muốn:



Hình 12.5: *Tắt ghim xung cho động cơ*

Bạn đọc có thể chủ động thay đổi chân cảm của động cơ với mạch mở rộng để có thể thử nghiệm thêm nhiều vị trí khác. Bạn hãy ghi nhận những chân có kết nối ổn định và sử dụng nó cho các dự án trong tương lai. Vì tín hiệu xuất ra cho động cơ là rất nhạy cảm, việc tiếp xúc không tốt giữa Yolo:Bit và mạch mở rộng có thể ảnh hưởng đến hoạt động của động cơ RC Servo.



# **Phần III**

## **Kết Nối Vạn Vật với OhStem**



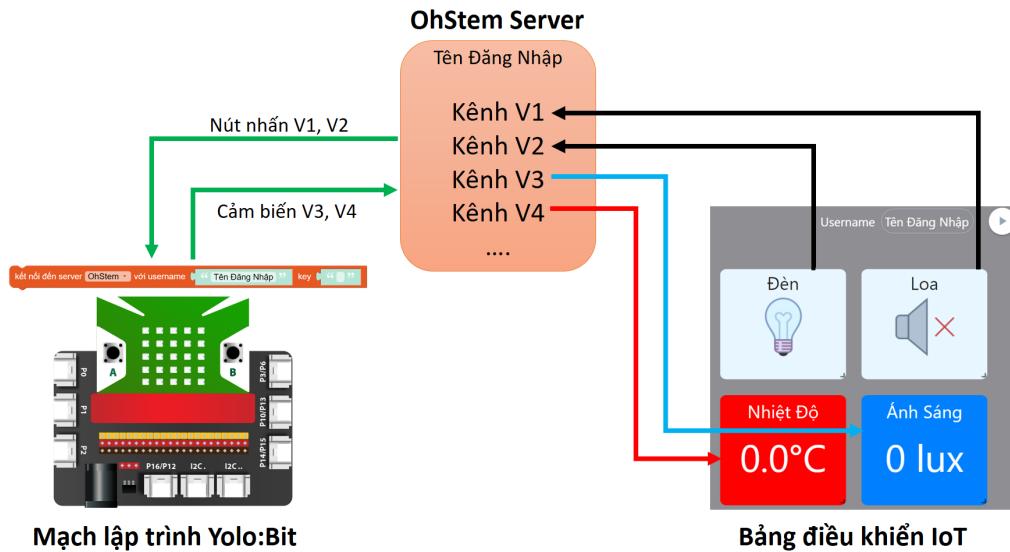
# CHƯƠNG 13

## Bảng Điều Khiển IoT



# 1 Giới thiệu

Để thiết bị có thể kết nối và chia sẻ dữ liệu trong thế giới kết nối vạn vật, chúng ta cần dùng đến điện toán đám mây - nơi sẽ lưu trữ dữ liệu của các thiết bị. Thông thường, chúng ta hay gọi đây là các máy chủ Server. Hiện tại, có rất nhiều Server thông dụng cho các ứng dụng kết nối vạn vật. Chúng ta có thể kể ra một vài cái tên như ThingSpeak, Blink hay Adafruit. Tuy nhiên, trong giáo trình này, chúng tôi sẽ sử dụng **OhStem Server** để các hướng dẫn có thể đồng nhất và thuận tiện trong các vấn đề cập nhật hoặc nâng cấp sau này.



Hình 13.1: Các thành phần trong một ứng dụng kết nối vạn vật

Vai trò của 3 thành phần trên trong một ứng dụng dựa trên công nghệ kết nối vạn vật được tóm tắt như sau:

- Mạch Yolo:Bit: Là thiết bị phần cứng đặc trưng cho khái niệm **Vật** (Things) trong kiến trúc vạn vật. Nó sẽ định kì cập nhật các giá trị cảm biến lên server đồng thời nhận các tín hiệu điều khiển từ server, chẳng hạn như để bật tắt một thiết bị nào đó.
- Bảng điều khiển IoT: Đây là một giao diện để chúng ta có thể theo dõi các thông tin cảm biến ở mạch Yolo:Bit một cách trực quan và sinh động. Chẳng hạn như thông tin về nhiệt độ hiện tại hay lịch sử của nhiệt độ. Ngoài ra sẽ có những nút điều khiển trên màn hình, để chúng ta có thể vận hành hệ thống của mình ở bất kì đâu, miễn là có mạng Internet.
- OhStem server: Thực ra đây chính là hạt nhân của kiến trúc vạn vật, khi tất cả dữ liệu từ mạch Yolo:Bit và bảng điều khiển IoT đều phải đi qua OhStem server.

Để đơn giản hóa cho việc tiếp cận của bạn đọc, chúng ta không cần phải đăng ký tài khoản khi sử dụng với OhStem server. Bằng việc tạo ra bảng điều khiển, và điền thông tin vào ô **Username**, một tài khoản trên OhStem server sẽ được tự động tạo ra. Thông tin Username khi tạo bảng điều khiển là rất quan trọng, bởi nó sẽ được dùng cho việc lập trình trên mạch Yolo:Bit trong tương lai. Trong bài hướng dẫn này, chúng tôi sẽ tập trung vào các nội dung sau đây:

- Nguyên lý chia sẻ dữ liệu của server OhStem
- Thiết kế bảng điều khiển đơn giản với 2 nút nhấn
- Cấu hình đăng ký kênh dữ liệu cho nút nhấn

## 2 Chia sẻ dữ liệu trên OhStem server

Giao tiếp với OhStem server được dựa trên một giao thức chuyên dụng cho các ứng dụng kết nối vật vật, có tên gọi là MQTT (Message Queuing Telemetry Transport). Giao thức này tận dụng việc một gói dữ liệu có kích thước nhỏ, để có thể luân chuyển nó giữa bảng điều khiển và thiết bị Yolo:Bit được nhanh nhất.

Cơ chế đầu tiên của giao thức MQTT, là gửi dữ liệu lên server OhStem, hay còn gọi là **Publish** dữ liệu. Bảng điều khiển lõi mạch Yolo:Bit đều có thể gửi dữ liệu lên server. Tuy nhiên, dữ liệu khi gửi lên server sẽ được phân ra từng kênh rõ rệt. Trên OhStem server, có 20 kênh dữ liệu khác nhau được hỗ trợ, có tên là **V1, V2, ..., V20**. Như minh họa ở Hình 13.1, khi một nút nhấn dùng để điều khiển bật/tắt một bóng đèn sẽ gửi dữ liệu lên server ở kênh dữ liệu V1 mỗi khi chúng ta nhấn nó. Tương tự như vậy, mạch Yolo:Bit có thể gửi thông tin về nhiệt độ và cường độ ánh sáng lên server, nhưng ở 2 kênh khác, là V3 và V4.

Cơ chế thứ hai của MQTT, dùng để nhận dữ liệu từ server, gọi là **Subscribe**. Cơ chế này giống hệt với việc bạn muốn nhận thông báo từ một kênh Youtube, thì phải đăng ký nó. Trong ngữ cảnh của ứng dụng kết nối vật vật, khi muốn nhận dữ liệu từ chủ đề (topic) nào, thiết bị cần phải đăng ký vào chủ đề đó.

Như vậy, để mạch Yolo:Bit biết nút nhấn điều khiển đèn đang là trạng thái bật hay tắt, **nó phải đăng ký vào kênh V1**, là kênh dữ liệu mà nút nhấn trên màn hình điều khiển sẽ gửi dữ liệu lên. Trong trường hợp ngược lại, khi một giao diện hiển thị được thông tin cảm biến do mạch Yolo:Bit gửi lên server, nó phải liên kết đúng với kênh dữ liệu. Nói một cách khác, con số trên màn hình điều khiển phải đăng ký vào kênh dữ liệu do mạch Yolo:Bit gửi lên, trong ví dụ ở Hình 13.1 là **V3 cho nhiệt độ** và **V4 cho ánh sáng**.

## 3 Thiết kế bảng điều khiển

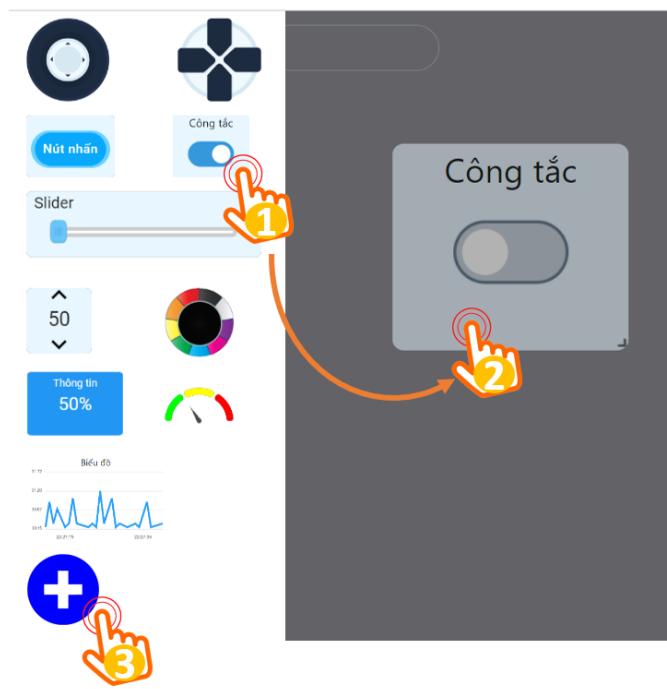
Trong hướng dẫn tiếp theo, chúng ta sẽ bắt đầu thiết kế một giao diện điều khiển đơn giản với 2 nút nhấn. Trình tự để thực hiện bảng điều khiển được trình bày chi tiết bên dưới.

**Bước 1:** Từ trang lập trình chính <https://app.ohstem.vn/>, chọn vào **Bảng điều khiển IoT**, sau đó nhấn tiếp vào nút **TẠO MỚI**, như hướng dẫn bên dưới.



Hình 13.2: Tạo mới một bảng điều khiển IoT

**Bước 2:** Với các giao diện được hỗ trợ trên bảng điều khiển, kéo thả một **Công tắc** ra màn hình, như minh họa ở hình bên dưới.



Hình 13.3: Kéo thả một công tắc ra màn hình

Trong trường hợp bạn nhấn vào màn hình, các đối tượng cho việc thiết kế sẽ ẩn đi. Lúc này, bạn chỉ cần nhấn vào biểu tượng ở góc dưới bên trái của mình hình (đánh dấu số 3 trên Hình 13.3 để mở nó lại).

Với mỗi đối tượng giao diện, khi muốn thay đổi kích thước, chúng ta di chuyển chuột vào góc dưới bên phải của đối tượng, nơi có 1 kí hiệu nhỏ. Biểu tượng thay đổi kích thước sẽ xuất hiện để chúng ta có thể kéo và thả.

**Bước 3:** Nhấn chuột trái trực tiếp vào đối tượng công tắc trên màn hình, giao diện sau đây sẽ hiện ra để chúng ta cấu hình thông tin cho công tắc.



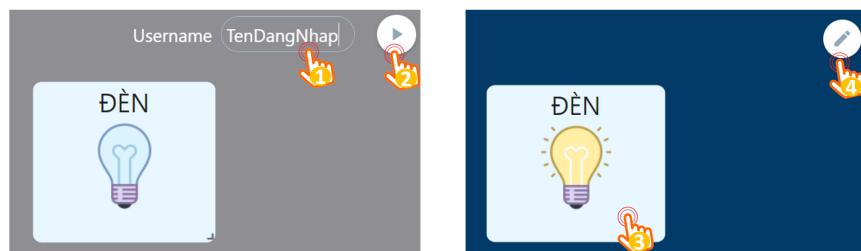
Hình 13.4: Cấu hình cho đối tượng công tắc

Một số thông tin quan trọng cho một công tắc được diễn giải như sau:

- Tên: Là dòng chữ hiển thị bên trên nút nhấn. Trong minh họa ở Hình 13.4, chúng tôi đã sửa lại thành **ĐÈN**.
- Kênh thông tin: Đây là cấu hình cực kì quan trọng cho một công tắc. Để cho đơn giản, bạn hãy dành **một kênh dữ liệu cho một công tắc**. Trong trường hợp này, chúng tôi chọn **V1**.
- Giá trị mở và tắt: Mặc định là 1 và 0, chúng ta nên để giá trị mặc định này và không thay đổi nó.
- Ánh gợi ý: Chọn lựa các hình ảnh đẹp cho một công tắc. Ở đây, chúng tôi đã chỉnh lại thành **Bóng đèn**

Cuối cùng, nhấn vào nút **OK** để tạo ra một công tắc điều khiển trên màn hình. Trường hợp muốn xóa một nút đã tạo, chúng ta nhấn chuột trái vào nó và chọn **XÓA**

**Bước 4:** Cung cấp thông tin **Username** và chạy thử màn hình điều khiển IoT.

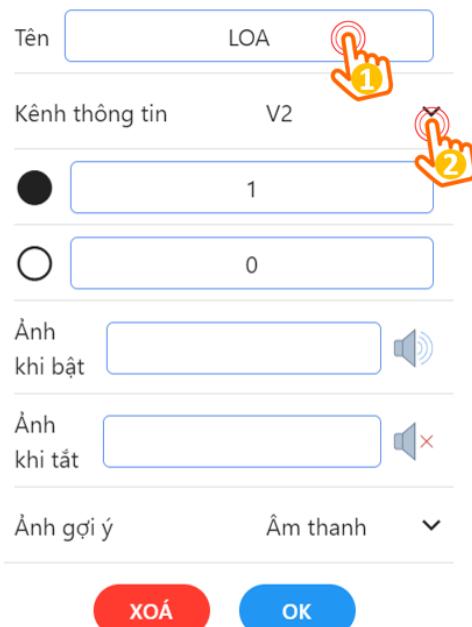


Hình 13.5: Đặt tên cho màn hình điều khiển

Như đã trình bày ở phần giới thiệu, thông tin ở trường **Username** dùng để đồng bộ các thiết bị trong ứng dụng của chúng ta. Bạn đọc hãy chọn 1 thông tin có sử dụng thêm các kí số đặc biệt để đảm bảo nó không trùng với ai, chẳng hạn như 6 số cuối của thẻ căn cước chẳng hạn.

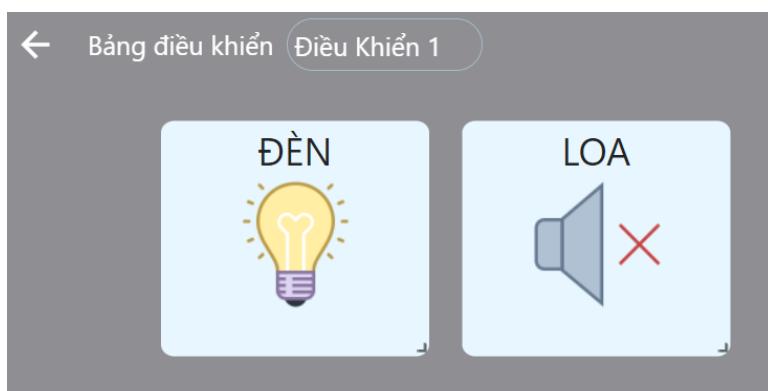
Bây giờ, bạn đọc có thể nhấn vào biểu tượng ở góc bên phải để chạy màn hình điều khiển (kí hiệu số 2 trên Hình 13.5), tương tác trên nút nhấn **ĐÈN** để thấy các hiệu ứng thay đổi mỗi khi nhấn vào nó. Cuối cùng, để trở lại màn hình thiết kế, chúng ta nhấn vào biểu tượng thay đổi ở góc bên phải (kí hiệu số 4 trên Hình 13.5).

**Bước 5:** Thiết kế thêm 1 nút nhấn, đặt tên là **LOA** và cấu hình cho **Kênh thông tin** là **V2**, như minh họa ở hình bên dưới.



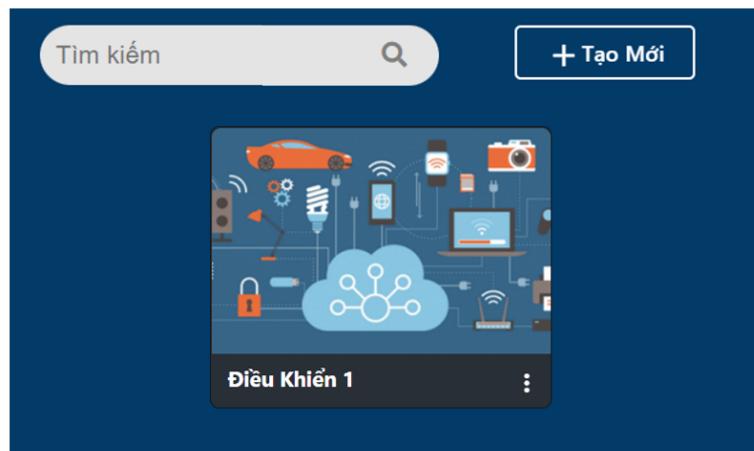
Hình 13.6: Tạo công tắc mới với kênh V2

**Bước 6:** Sắp xếp đối tượng trên màn hình và đặt tên cho bảng điều khiển, chúng ta có thể có được giao diện như sau.



Hình 13.7: Sắp xếp đối tượng giao diện trên màn hình

Lần tiếp theo, khi vào lại bảng điều khiển IoT, chúng ta sẽ có sẵn một bản điều khiển đã được lưu lại.



Hình 13.8: Các bảng điều khiển đã thiết kế

Trong bài hướng dẫn tiếp theo, chúng ta sẽ lập trình để nhận dữ liệu điều khiển trên mạch Yolo:Bit và điều khiển thiết bị tương ứng với trạng thái của 2 công tắc trên bảng điều khiển.



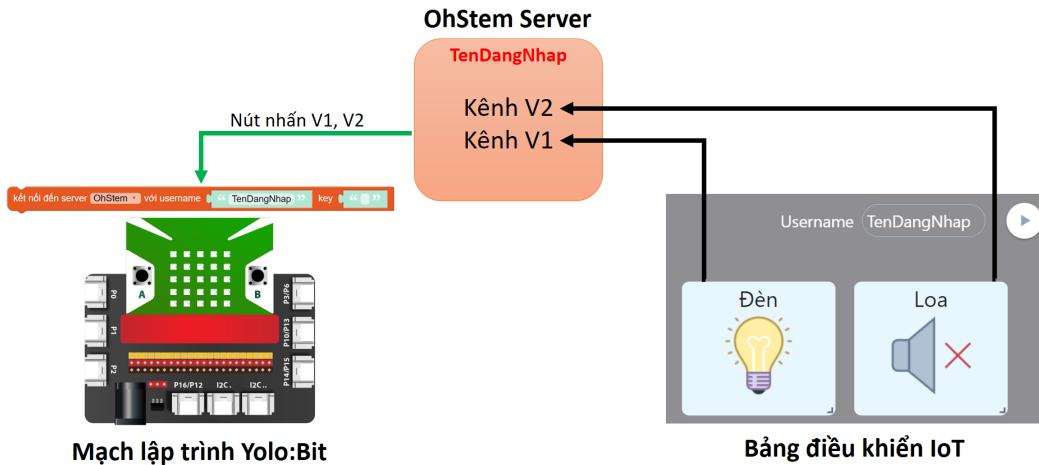
# CHƯƠNG 14

## Nhận dữ liệu trên Yolo:Bit



# 1 Giới thiệu

Sau khi đã tạo xong màn hình điều khiển với 2 nút nhấn đơn giản, chúng ta đã có thể lập trình cho Yolo:Bit, để nhận dữ liệu và thực sự điều khiển thiết bị tương ứng. Tuy nhiên, trước khi lập trình, chúng ta hãy cùng nhìn lại sơ đồ tổng quan của hệ thống kết nối vạn vật dựa trên giao thức MQTT mà chúng ta đang có:



Hình 14.1: Cấu trúc kết nối vạn vật với Yolo:Bit và OhStem server

Khi thiết kế ra màn hình điều khiển, thông tin Username là rất quan trọng để sử dụng cho việc lập trình tiếp theo. **Bạn đọc cần chọn cho mình 1 cái tên ít có khả năng trùng nhau.** Ở đây, để minh họa cho hướng dẫn, chúng tôi đang chọn là **Ten-DangNhap**. Bạn đọc cần chủ động chọn một tên khác để tránh bị trùng tên với người khác.

Khi thiết kế ra giao diện điều khiển, chúng ta đã sử dụng 2 kênh dữ liệu, là V1 cho nút nhấn điều khiển đèn và V2 cho nút nhấn điều khiển loa. Mỗi khi chúng ta nhấn vào một nút trên màn hình điều khiển, quy trình hoạt động của hệ thống sẽ như sau:

- Mỗi khi chúng ta nhấn nút nhấn, dữ liệu thô khi cấu hình (là 0 hoặc 1) sẽ được gửi lên OhStem server trước, và nó kết thúc nhiệm vụ của mình ở đó.
- Sau đó, OhStem server sẽ gửi thông tin này xuống mạch phần cứng của chúng ta, tức là Yolo:Bit, **nếu nó có đăng ký nhận dữ liệu từ V1 và V2.**

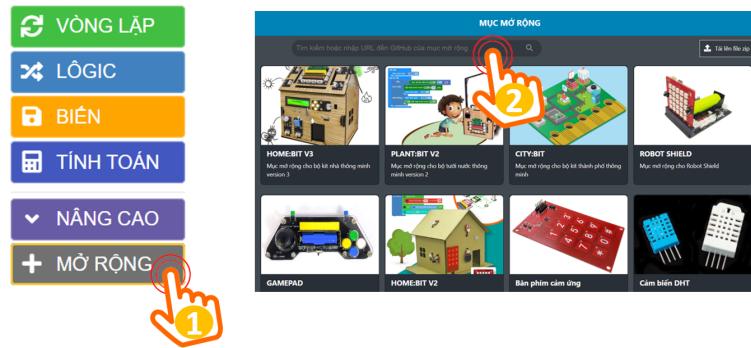
Thực ra, cơ chế đăng ký nhận dữ liệu (subscribe) đã tồn tại rất nhiều trong các ứng dụng mà chúng ta đang sử dụng hiện tại. Ví dụ: Mỗi khi bạn đăng ký một kênh trên Youtube, bạn sẽ tự động nhận được thông báo từ kênh đó. Bên dưới mạch Yolo:Bit đã thực hiện một cơ chế tương tự như việc bạn đăng ký 1 kênh trên Youtube. Nội dung hướng dẫn trong bài này tập trung vào lập trình để Yolo:Bit có thể nhận được thông tin điều khiển từ 2 kênh dữ liệu V1 và V2, cụ thể như sau:

- Thêm thư viện lập trình MQTT
- Kết nối vào mạng WiFi
- Đăng ký kênh dữ liệu V1 và V2
- Nhận và xử lý dữ liệu từ OhStem server

## 2 Thêm khôi mở rộng MQTT

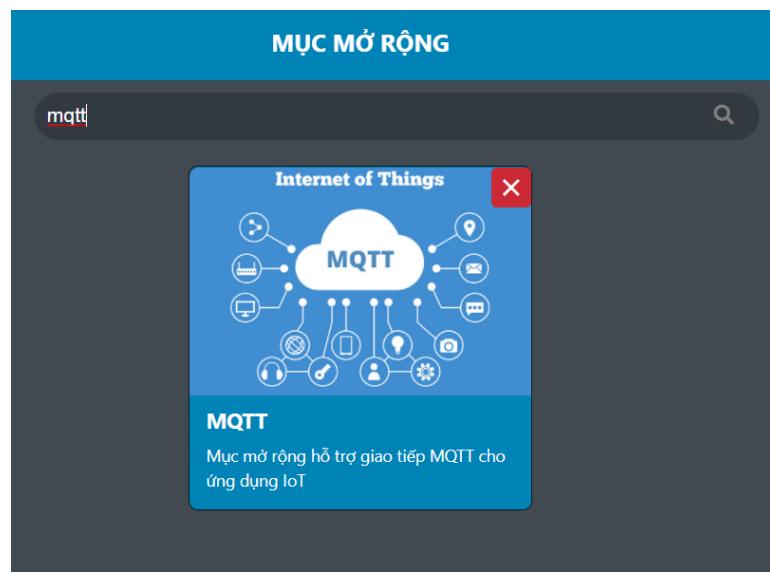
Để mạch Yolo:Bit có thể kết nối WiFi và thực hiện cơ chế nhận dữ liệu với OhStem server, chúng ta cần tải thêm thư viện lập trình mở rộng MQTT cho nó. Các bước để thêm thư viện khôi lệnh này được trình bày chi tiết bên dưới:

**Bước 1:** Trong danh mục khôi lệnh, chọn vào khôi **MỞ RỘNG** để mở các thư viện mở rộng, như minh họa ở hình dưới:



Hình 14.2: Thêm thư viện lập trình cho Yolo:Bit

**Bước 2:** Nhập từ khóa **mqtt** vào ô tìm kiếm, sau đó nhấn Enter. Kết quả của việc tìm kiếm sẽ xuất hiện như hình:



Hình 14.3: Thư viện lập trình MQTT cho Yolo:Bit

**Bước 3:** Nhấn vào MQTT để thêm thư viện. Khi thông báo sau đây xuất hiện, bạn chọn **OK**.

Tương tự như việc thêm thư viện mở rộng ở các bài hướng dẫn trước, bạn đọc nên kết nối mạch Yolo:Bit vào máy tính để tải thư viện mở rộng vào mạch. Nếu không, chúng ta sẽ phải tải lại thư viện sau đó (bằng cách chọn vào biểu tượng bánh răng

### Tải thư viện

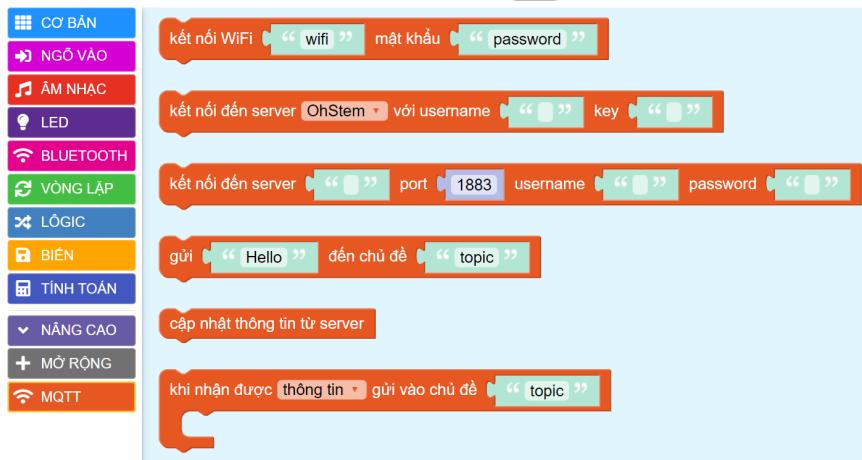
Mục mở rộng này cần thư viện đi kèm để sử dụng được. Bạn có muốn tải thư viện (thiết bị cần được kết nối)?

BỎ QUA

OK

Hình 14.4: Tải thư viện mở rộng MQTT cho Yolo:Bit

và chọn tiếp Tải thư viện). Cuối cùng, chúng ta sẽ có một nhóm khối lệnh mới như hình:



Hình 14.5: Khối lệnh trong thư viện MQTT

## 3 Lập trình trên Yolo:Bit

Việc lập trình trên Yolo:Bit khá giống với trình tự khi chúng ta sử dụng Youtube trên máy tính vậy. Việc so sánh này được tóm tắt như bảng bên dưới.

Trình tự	Yolo:Bit	Youtube
1	Đăng nhập vào mạng WiFi (tên mạng + mật khẩu)	Đăng nhập vào mạng WiFi (tên mạng + mật khẩu)
2	Đăng nhập vào tài khoản trên OhStem server	Đăng nhập vào tài khoản trên Youtube
3	Đăng ký kênh để nhận dữ liệu từ OhStem server	Đăng ký kênh Youtube để nhận thông tin mới về kênh

Với trình tự các bước như trên, bạn cũng chỉ **làm một lần duy nhất** khi sử dụng Youtube. Trên Yolo:Bit, nguyên lý hoạt động cũng tương tự như vậy, các bước này cũng sẽ được hiện thực trong phần **bắt đầu** của chương trình. Chi tiết hiện thực của từng bước sẽ được trình bày bên dưới.

### 3.1 Kết nối vào mạng WiFi

Đây là bước đầu mà chúng ta cần làm để thiết bị có thể kết nối với Internet. Cũng giống như máy tính, việc kết nối với mạng WiFi bất kỳ chỉ cần được thực hiện một lần. Do đó, chúng ta sẽ lập trình tính năng này trong phần **bắt đầu** của chương trình.

Câu lệnh chúng ta sử dụng là câu lệnh đầu tiên trong thư viện MQTT: **câu lệnh kết nối WiFi**, như sau:



Hình 14.6: Kết nối với mạng WiFi

Trong câu lệnh này, bạn cần cung cấp 2 thông tin là tên và mật khẩu của WiFi cho Yolo:Bit. Trong các cuộc thi, thí sinh thường sử dụng điện thoại để phát mạng 4G và cho Yolo:Bit kết nối vào mạng 4G đó. Trong trường hợp này, bạn nên đặt tên và mật khẩu đơn giản, không có các khoảng trắng hoặc các kí tự đặc biệt, để giảm bớt rủi ro như nhập sai mật khẩu.

Đối với một số điện thoại đời mới, sử dụng băng tần cho mạng 5G, mạch Yolo:Bit sẽ không kết nối được vào điểm phát sóng của điện thoại. Trong trường hợp này, bạn bắt buộc phải tìm một điện thoại khác phù hợp hơn. Thông thường, các điện thoại sử dụng hệ điều hành Android sẽ dễ kết nối hơn là iOS.

### 3.2 Kết nối với OhStem server

Sau khi kết nối với mạng WiFi, chúng ta sẽ lập trình để Yolo:Bit đăng nhập vào server OhStem. Câu lệnh mà bạn sẽ sử dụng là **câu lệnh thứ 2 trong MQTT**, như trình bày bên dưới:



Hình 14.7: Câu lệnh để đăng nhập vào OhStem server

Mặc dù câu lệnh này có 3 lựa chọn khác nhau, chúng ta chỉ cần điền thông tin vào phần **username** là đủ. Thông tin về server đã được chọn mặc định và mật khẩu (phần **key** là không cần thiết). Sau khi cung cấp chính xác **TenDangNhap**, chúng ta sẽ ghép câu lệnh này vào chương trình, như sau:

Các câu lệnh hiện chữ được sử dụng trong phần này có ý nghĩa quan trọng. Với câu lệnh đầu tiên, đó là dấu hiệu để biết rằng chương trình của chúng ta bắt đầu thực thi. Câu lệnh thứ hai, chữ **OK** sẽ báo hiệu rằng việc kết nối với mạng WiFi



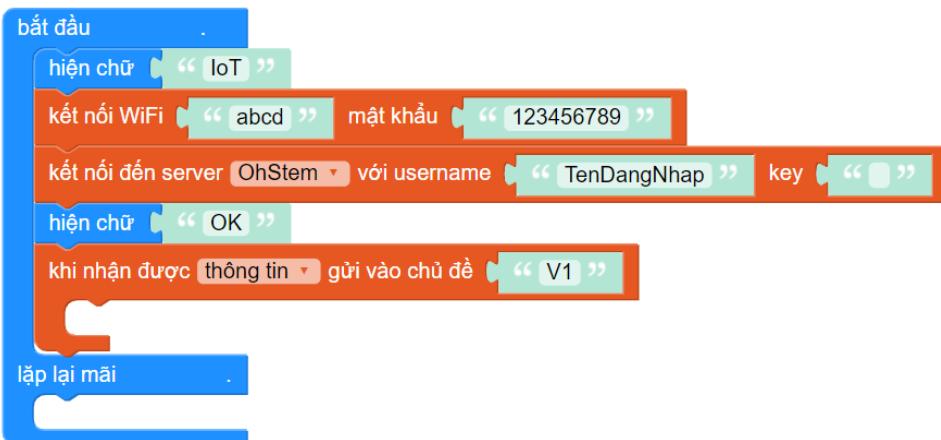
Hình 14.8: Đăng nhập vào OhStem server

và OhStem server là thành công. Thông thường, lỗi chính của phần này sẽ nằm ở phần kết nối với mạng Wifi (chữ **OK** không hiện lên mạch Yolo:Bit). Trình tự để khắc phục lỗi này như sau:

- Kiểm tra lại tên mạng và mật khẩu, không nên có kí tự đặc biệt cho cả 2 trường thông tin này.
- Reset lại mạch Yolo:Bit, nạp lại thư viện, nạp lại chương trình.
- Tìm một mạng Wifi khác để kiểm tra, có thể điện thoại phát mạng Wifi đang sử dụng băng tầng 5G và không tương thích với Yolo:Bit.

### 3.3 Đăng ký kênh dữ liệu

Cuối cùng, bạn cần đăng ký vào kênh dữ liệu mà chúng ta muốn nhận (hay còn gọi là chủ đề - topic) trên OhStem. Trong trường hợp này, chúng ta sẽ đăng ký trước vào kênh **V1**, như sau:

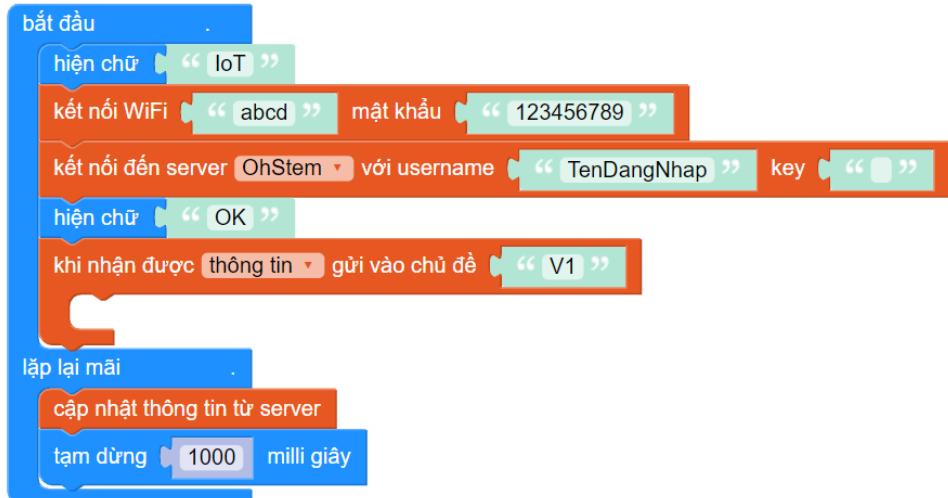


Hình 14.9: Đăng ký kênh nhận dữ liệu

Bạn cần lưu ý rằng, tên của kênh dữ liệu là kí tự viết hoa. Đến đây, mỗi khi nút nhấn trên màn hình điều khiển, dữ liệu sẽ được tự động lưu lại trong biến **thông tin**. Phần xử lý khi nhận dữ liệu sẽ được trình bày ở các phần sau.

### 3.4 Liên kết định kỳ với Server

Sau các bước cấu hình ở trên, chúng ta cần phải tạo một liên kết định kỳ với Server. Việc này được thực hiện lặp đi lặp lại liên tục, nên chúng ta cần phải hiện thực nó trong khối **lặp lại mãi**, như sau:



Hình 14.10: Liên kết định kỳ với Server

Chu kì kiểm tra kết nối với Server mà chúng tôi để xuất ở đây là 1 giây, tức là 1000ms (sử dụng câu lệnh **tạm dừng** trong mục **CƠ BẢN**). Thời gian dừng càng lớn thì việc nhận tín hiệu điều khiển khi nhấn nút sẽ chậm. Tuy nhiên, nếu thời gian dừng nhỏ thì chúng sẽ làm tốn tài nguyên của mạng Internet (do mạch Yolo:Bit phải thường xuyên truy cập và gửi dữ liệu lên Server OhStem).

Trong các ứng dụng hiện tại, chúng ta nên sử dụng độ trễ 1 giây.

## 4 Xử lý dữ liệu nhận được từ server

Để xử lý dữ liệu nhận được (lưu trong biến **thông tin**), chúng ta cần phải lập trình trong phần **bắt đầu**. Nhưng để đơn giản trong việc trình bày, chúng tôi chỉ trình bày chương trình cho bạn (chứ không đặt chúng trong phần bắt đầu). Chương trình sẽ như sau:



Hình 14.11: Xử lý dữ liệu nhận được

Ta thấy rằng đây là một chương trình có rất nhiều màu, chứng tỏ các câu lệnh được lấy từ nhiều nhóm lệnh khác nhau. Bạn có thể đi theo trình tự gợi ý sau để lập trình chương trình trên:

- Lấy **khối lệnh nếu thực hiện... nếu không** trong mục **LOGIC**
- Lấy **khối lệnh so sánh bằng**, cũng trong mục **LOGIC**
- Lấy **khối lệnh thông tin** có màu vàng, trong mục **BIÊN**
- Lấy **khối chuỗi ký tự rỗng** trong phần **NÂNG CAO**, mục **CHỮ VIẾT**, thay đổi nội dung bên trong nó thành chuỗi **1**
- Có nhiều cách để lập trình bật tắt đèn, nhưng ở đây, chúng tôi sử dụng **khối lệnh đổi màu đèn LED** trong mục **LED**

Do dữ liệu gửi trả về từ các server nói chung và OhStem nói riêng, là kiểu dữ liệu dạng chuỗi, do đó, trong phần xử lý này, chúng ta bắt buộc phải thực hiện phép so sánh chuỗi. Hình ảnh của chương trình lúc này như sau:



Hình 14.12: Chương trình nhận dữ liệu và xử lý trên Yolo:Bit

## 5 Mở rộng chương trình

Chúng ta sẽ tiếp tục làm thêm một khối lệnh đăng ký kênh V2 và xử lý cho nó, tương tự với việc bật và tắt đèn ở kênh V1. Phần hướng dẫn này sẽ không được trình bày chi tiết bởi nó khá tương tự ở trên. Chương trình gợi ý của chúng tôi cho bạn đọc như sau:



Hình 14.13: Chương trình đăng ký và xử lý 2 kênh dữ liệu

Bạn đọc cần lưu ý về **tầm vực** của 2 câu lệnh đăng ký kênh dữ liệu. Nó phải nằm thẳng hàng và đồng cấp với nhau. **Nếu câu lệnh đăng ký V2 nằm bên trong V1, chương trình sẽ bị sai về mặt luận lý.** Chương trình của bài này được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2ByR7pLN0bIAKIZ9WVPQJThcjS2>



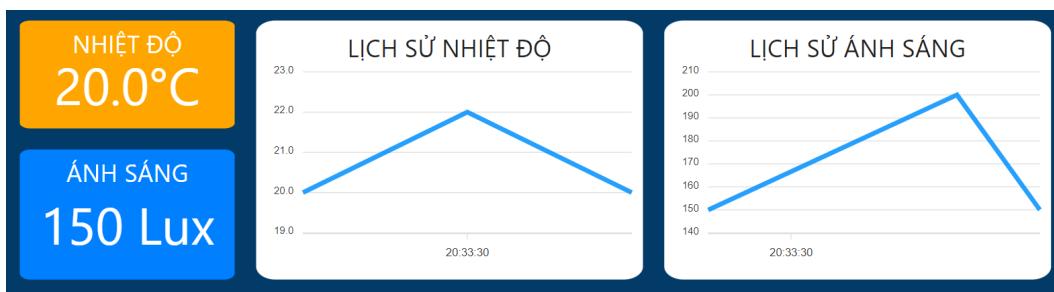
# CHƯƠNG 15

## Hiển thị thông tin IoT



## 1 Giới thiệu

Bên cạnh việc điều khiển thiết bị từ xa thông qua mạng với các dịch vụ đặc biệt cho kết nối vật vật, tính năng theo dõi thông tin cảm biến ở mạch Yolo:Bit cũng là một nhu cầu cần thiết cho các ứng dụng thông minh. Trong ngữ cảnh về vườn rau thông minh, chúng ta sẽ có nhu cầu giám sát thông tin về điều kiện nuôi trồng từ xa, chẳng hạn như thông tin về nhiệt độ không khí hay độ ẩm đất.



Hình 15.1: Giao diện quan trắc thông tin từ thiết bị

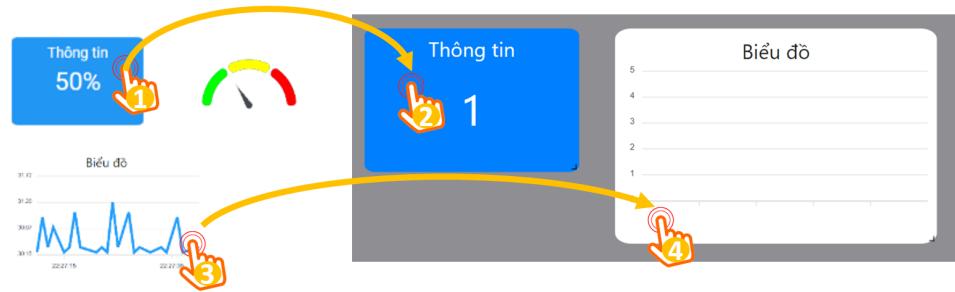
Trong bài hướng dẫn này, chúng ta sẽ thiết kế trước một giao diện trên màn hình để có thể giám sát trực quan nhất dữ liệu từ xa. Với những cảm biến hỗ trợ sẵn trên Yolo:Bit, chúng tôi sẽ thiết kế giao diện để theo dõi thông tin về nhiệt độ và cường độ ánh sáng. Trong tương lai, chúng ta hoàn toàn có thể thêm các thông tin theo dõi khác.

Để tránh đụng độ với 2 kênh dữ liệu ở bài trước, chúng ta sẽ chọn kênh V3 để lưu thông tin cho nhiệt độ và V4 cho cường độ ánh sáng. Mặc dù hướng dẫn ở bài này sẽ tạo mới hoàn toàn một giao diện điều khiển, bạn đọc hoàn toàn có thể mở bảng điều khiển ở bài trước để tích hợp chung vào 1 giao diện. Với mỗi thông số cảm biến, chúng ta sẽ dễ dàng quan sát được giá trị hiện tại cũng như lịch sử của nó. Các nội dung chi tiết của bài hướng dẫn này như sau:

- Tạo giao diện giám sát giá trị hiện tại
- Tạo giao diện giám sát giá trị lịch sử

## 2 Giao diện Thông tin và Đồ thị

Với các đối tượng hỗ trợ sẵn trên OhStem server, có 2 đối tượng rất thích hợp cho yêu cầu ở bài này, là **Thông tin** và **Đồ thị**. Một đối tượng sẽ được sử dụng để hiển thị thông tin hiện tại, trong khi đối tượng còn lại sẽ dùng để hiển thị lịch sử của dữ liệu. Tương tự như nút nhấn ở bài trước, bạn đọc có thể kéo thả 2 đối tượng này từ thanh công cụ bên trái vào màn hình điều hành, như minh họa ở hình bên dưới. Di chuyển chuột vào kí hiệu ở góc dưới bên phải của mỗi đối tượng, bạn đọc có thể thay đổi kích thước có nó. Nếu muốn thay đổi vị trí của nó, chúng ta nhấn đèn vòi đối tượng và kéo thả nó ra vị trí mới.



Hình 15.2: Giao diện Thông tin và Đồ thị

Chúng ta sẽ cấu hình cho hai đối tượng đang có trên màn hình để hiện thị cho phần **Nhiệt độ**. Phần ánh sáng bạn đọc có thể làm tương tự. Như đã quy định ở đầu bài, kênh thông tin dành cho nhiệt độ là V3. Các hướng dẫn bên dưới sẽ tập trung vào việc cấu hình từng đối tượng giao diện và liên kết nó với kênh V3.

## 2.1 Cấu hình cho Thông tin

Nhân chuột trái vào đối tượng Thông tin, giao diện dùng để cấu hình đối tượng này sẽ hiện ra, để bạn đọc có thể thay đổi các thông tin như minh họa dưới đây.



Hình 15.3: Cấu hình cho đối tượng Thông tin

Các thông số cấu hình sau đây là quan trọng và cần lưu ý:

- Tên: Thông tin sẽ hiển thị bên trên giá trị cảm biến. Mặc dù không phải là thông tin quan trọng, nó lại có ý nghĩa với người dùng. Bạn đọc nên thay đổi giá trị của nó cho phù hợp với cảm biến cần giám sát.
- Kênh thông tin: Đây là thông số cấu hình quan trọng nhất, khi bạn phải lựa chọn kênh thông tin chính xác. Trong trường hợp của chúng tôi là V3 dành cho nhiệt độ.
- Cách hiển thị: Thông số cảm biến có 1 chữ số thập phân hay được làm tròn hay đơn vị của nó, sẽ được tùy chỉnh ở phần này. Mặc dù bạn đọc có thể nhập

trực tiếp vào ô nhập liệu, chúng tôi khuyên bạn hãy lựa chọn trong danh sách có sẵn, bằng cách nhấn vào biểu tượng lựa chọn ở góc bên phải của phần này.

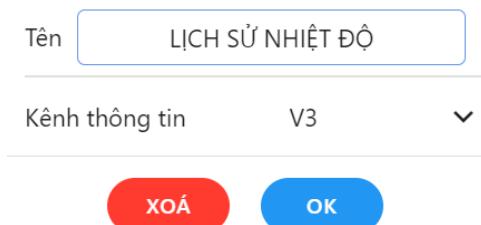
- Màu nền: Một số màu nền thông dụng cho bạn đọc lựa chọn cho đối tượng thông tin.

Cuối cùng, nhấn vào **OK** nếu như bạn đã cấu hình xong, hoặc nhấn vào **XÓA** để xóa hoàn toàn đối tượng giao diện này và tạo lại từ đầu.

Trong quá trình tạo một giao diện, nó có thể **xuất hiện một số dữ liệu rác**. Bạn đọc không cần phải lo lắng vì điều này. Khi mạch Yolo:Bit được chạy thật, các thông tin sẽ được cập nhật lại một cách chính xác.

## 2.2 Cấu hình cho Đồ thị

Việc cấu hình cho đồ thị lại đơn giản hơn nhiều so với giao diện Thông tin, khi chúng ta chỉ có 2 thông tin thường phải thay đổi, như minh họa ở hình bên dưới:



Hình 15.4: Cấu hình cho đối tượng Đồ thị

Thông tin quan trọng vẫn là Kênh thông tin, chúng ta cần phải lựa chọn là **V3** cho đúng với yêu cầu đã đặt ra. Với một thông tin về nhiệt độ, chúng ta đã biểu diễn nó dưới 2 dạng khác nhau, giúp cho việc giám sát trở nên thân thiện và thuận tiện hơn.

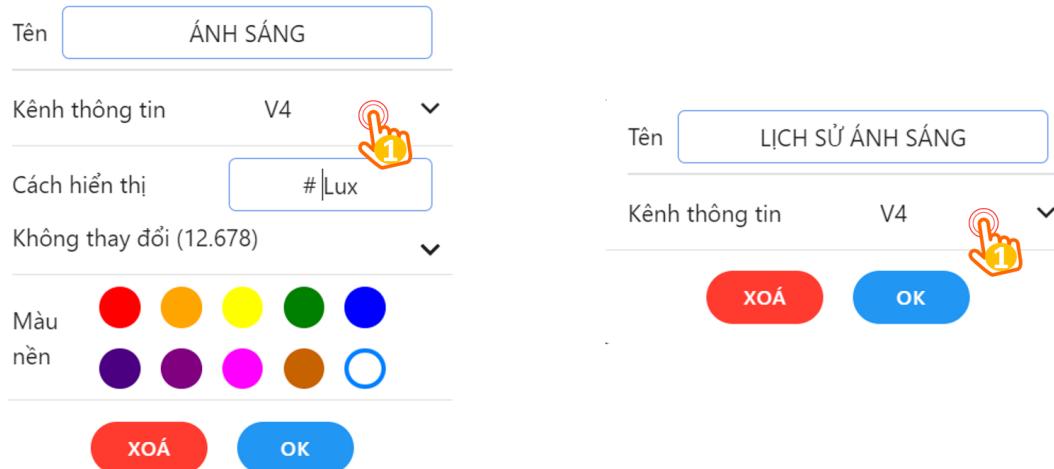
Cuối cùng, sắp xếp các giao diện một cách hợp lý và cung cấp thông tin cho trường Username, chúng ta sẽ có kết quả như sau:



Hình 15.5: Biểu diễn thông tin nhiệt độ trên bảng điều khiển

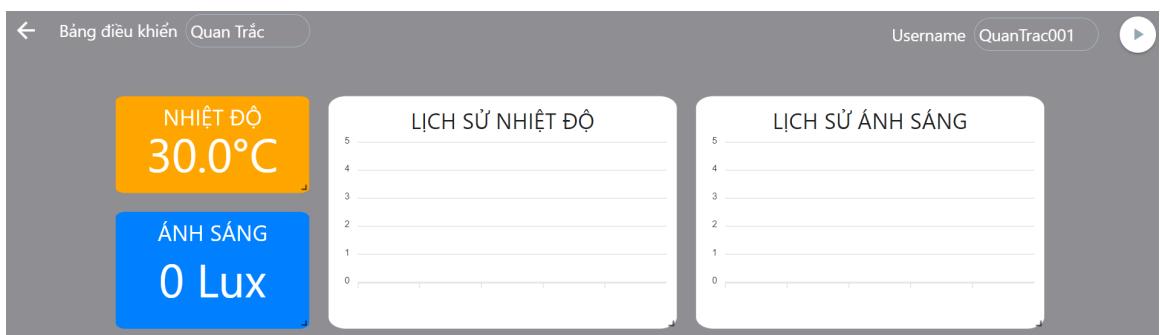
### 3 Hoàn thiện bảng giám sát

Thực hiện hoàn toàn tương tự như trên, bạn đọc có thể chủ động thêm 2 đối tượng giao diện để hiển diễn cho thông tin cảm biến về cường độ ánh sáng. Một lưu ý quan trọng là các giao diện mới sẽ liên kết với kênh dữ liệu V4. Ánh sáng có đơn vị là Lux.



Hình 15.6: Giao diện mới cho thông tin Ánh sáng

Bên cạnh đó, đơn vị cho ánh sáng không được hỗ trợ sẵn trong các lựa chọn trên OhStem, nên chúng ta cần phải gõ đơn vị này vào. Kết quả của màn hình giám sát sẽ như sau:



Hình 15.7: Giao diện quan trắc Nhiệt độ và Ánh sáng

Cuối cùng, chúng ta đặt tên cho màn hình điều khiển này, và thử chạy nó bằng cách nhấn vào nút Play ở góc trên bên phải của màn hình.



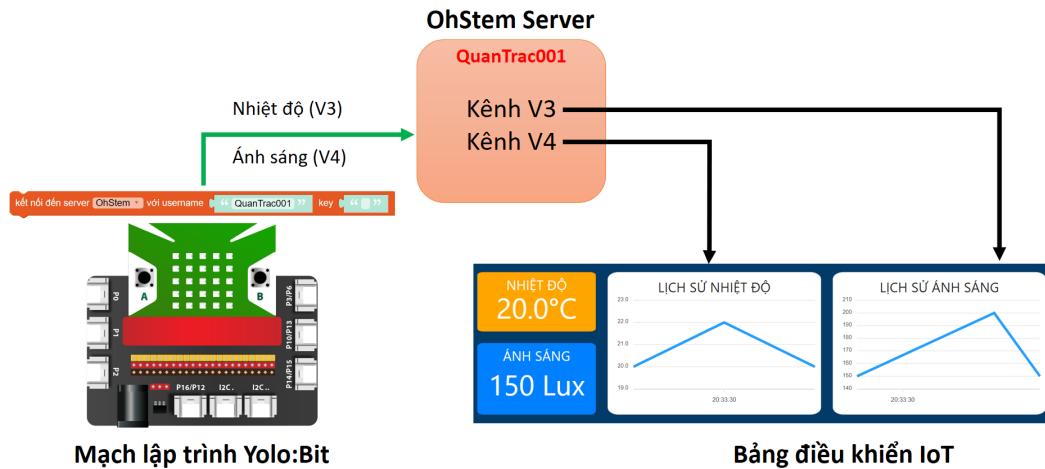
# CHƯƠNG 16

## Gửi dữ liệu lên OhStem server



# 1 Giới thiệu

Sau khi đã tạo xong các giao diện trên bảng điều khiển IoT, chúng ta đã có thể lập trình để Yolo:Bit gửi dữ liệu từ cảm biến lên OhStem server và biểu diễn giá trị này dưới dạng đồ thị cũng như thông tin hiện tại. Trước khi đi vào chi tiết của phần hiện thực, chúng ta hãy cùng nhìn lại sơ đồ tổng quan của hệ thống kết nối vạn vật mà chúng ta đang có, như trình bày ở hình bên dưới:



Hình 16.1: Cấu trúc kết nối vạn vật với Yolo:Bit và OhStem server

Rõ ràng, chiều đi của dữ liệu trong bài này sẽ ngược lại hoàn toàn với nút nhấn điều khiển thiết bị ở bài trước. Xuất phát từ mạch Yolo:Bit, thông tin cảm biến sẽ được gửi lên OhStem server ở 2 kênh dữ liệu là V3 và V4. Các đối tượng giao diện trên bảng điều khiển, khi cấu hình liên kết với các kênh dữ liệu này, nó sẽ được tự động cập nhật giá trị tương ứng.

Cần lưu ý rằng, chúng ta **không gửi dữ liệu trực tiếp từ Yolo:Bit tới bảng điều khiển**. Dữ liệu trao đổi giữa các thiết bị đầu cuối, trong trường hợp này là mạch Yolo:Bit và bảng điều khiển, đều phải đi qua OhStem server.

Với thiết kế tích hợp cho các ứng dụng hiện đại, Yolo:Bit có hỗ trợ nhiều thông tin cảm biến từ đơn giản đến phức tạp, như thông tin về nhiệt độ, cường độ ánh sáng, hay thậm chí là về gia tốc. Điều này giúp bạn thuận tiện hơn khi phát triển một ứng dụng nhỏ mà không cần tích hợp thêm các phần cứng thiết bị khác.

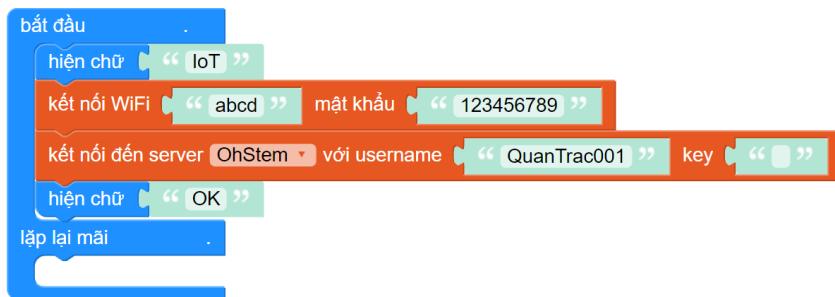
Trong bài hướng dẫn này, chúng tôi sẽ sử dụng cảm biến nhiệt độ có sẵn trên mạch Yolo:Bit để minh họa cho việc **gửi thông tin định kì lên Server** và theo dõi thông tin này trên đồ thị đã được thiết kế ở bài trước. Một khối lệnh mới sẽ được giới thiệu trong bài này, nhằm hỗ trợ trực tiếp cho việc gửi dữ liệu định kì được thuận tiện và dễ phát triển về sau.

Các nội dung hướng dẫn chính trong bài này như sau:

- Chương trình kết nối với OhStem server
- Thêm thư viện hỗ trợ cho khối lệnh định kì
- Gửi dữ liệu lên OhStem server

## 2 Kết nối với OhStem server

Để có thể sử dụng các dịch vụ mới về kết nối vạn vật, hiển nhiên mạch Yolo:Bit cần phải đăng nhập vào một mạng Internet, rồi sau đó mới kết nối vào server OhStem. Các chức năng này sẽ được hiện thực trong phần **bắt đầu** của chương trình, như sau:

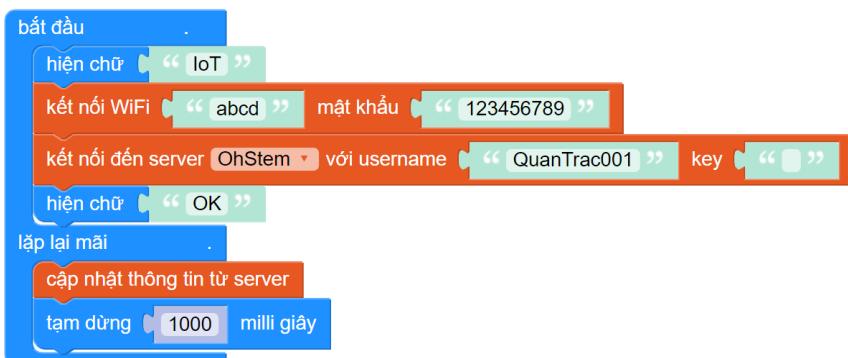


Hình 16.2: Kết nối mạng WiFi và OhStem server

Hai câu lệnh hiển thị được thêm vào để chúng ta kiểm soát tốt hơn chương trình. Thông thường, dòng chữ đầu tiên sẽ hiện ra. Sau đó khoảng 5 giây, dòng chữ thứ 2 (OK) sẽ xuất hiện, báo hiệu việc kết nối với mạng Internet và server OhStem thành công. Trong trường hợp chữ OK không xuất hiện, bạn đọc cần xem lại cách khắc phục đã trình bày ở Chương 3.

Thông tin quan trọng nhất trong 2 câu lệnh ở trên, là username để đăng nhập vào OhStem server. Thông tin này bắt buộc phải trùng khớp với màn hình điều khiển IoT mà bạn đã thiết kế. Trong trường hợp của chúng tôi là **QuanTrac001**.

Tiếp theo đó, sau khi đã kết nối thành công với server, chúng ta cần định kì giữ liên kết với nó. Với rất nhiều lý do liên quan đến hiệu suất của server, độ trễ của dữ liệu, chúng tôi đề xuất **chu kỳ giữ kết nối với server là 1 giây**. Với công việc phải thực hiện định kì, nó sẽ được hiện thực trong phần **lặp mãi mãi**, như ví dụ sau đây:



Hình 16.3: Định kì giữ kết nối với OhStem server

Chu kỳ 1 giây này là thông số rất cần thiết để hệ thống hoạt động ổn định. Do đó, chúng tôi sẽ cố định kiến trúc này cho toàn bộ hệ thống khi sử dụng tính năng kết nối vạn vật.

### 3 Thư viện lập trình Sự kiện

Sau khi cố định câu lệnh đợi 1 giây trong phần lặp mãi mãi, việc định kì gửi dữ liệu cảm biến lên server sẽ là trở ngại không nhỏ với bạn đọc. Lý do chính nằm ở chỗ chu kỳ cho việc gửi giá trị cảm biến thường là dài, khoảng 30 giây, 1 phút hoặc thậm chí lâu hơn, tùy theo ứng dụng. Do đó, để đơn giản hóa việc lập trình, chúng tôi hỗ trợ bạn đọc một thư viện mới, có tên là **SỰ KIỆN**. Trình tự thêm thư viện này vào môi trường lập trình được trình bày như bên dưới.

**Bước 1:** Nhấn vào nhóm lệnh **MỞ RỘNG**, và tìm kiếm thư viện **SỰ KIỆN** trong cửa sổ mới, như minh họa ở hình sau đây:



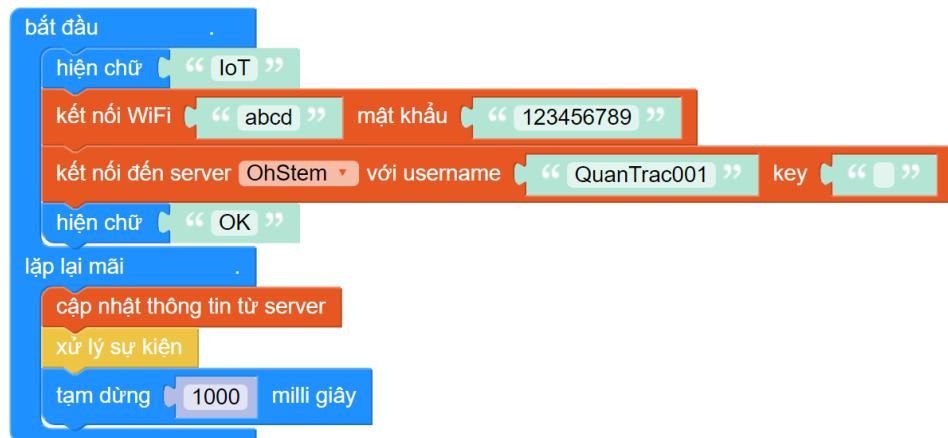
Hình 16.4: Thư viện lập trình SỰ KIỆN

Nhấn trực tiếp vào thư viện **SỰ KIỆN** để thêm nó vào môi trường lập trình và tải vào mạch Yolo:Bit. Như những bài hướng dẫn trước, bạn đọc nên cắm mạch Yolo:Bit vào máy tính trước khi nhấn thêm một thư viện mới vào. Hệ thống sẽ yêu cầu việc kết nối với mạch Yolo:Bit cho chức năng này. Khi thêm thư viện thành công, giao diện lập trình của chúng ta sẽ có thêm 1 nhóm lệnh mới, như sau:



Hình 16.5: Nhóm lệnh SỰ KIỆN được thêm vào

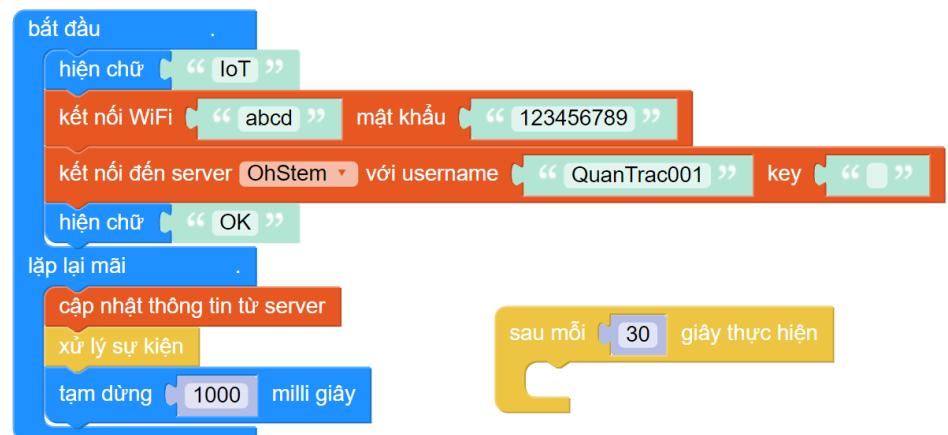
**Bước 2:** Thêm khối lệnh **xử lý sự kiện** vào phần lặp mãi mãi của chương trình, như sau:



Hình 16.6: Tích hợp xử lý sự kiện vào chương trình

Chúng ta cũng diễn giải cho chức năng vừa mới thêm vào giống với việc định kì liên kết với server, hệ thống cần định kì xử lý cho các khối lệnh liên quan đến sự kiện mà chúng ta sẽ dùng trong tương lai. Câu lệnh **tạm dừng 1000 mili giây** bắt buộc phải có trong phần **lặp mãi mãi**. Cho dù trong một số trường hợp đặc thù, có thể chúng ta không thực hiện một chức năng nào trong lặp mãi mãi, bạn vẫn nên có câu lệnh tạm dừng 1000 mili giây.

**Bước 3:** Sử dụng khối lệnh xử lý theo chu kì, câu lệnh **sau mỗi giây thực hiện**.



Hình 16.7: Sử dụng khối lệnh xử lý theo chu kì

Bây giờ, chúng ta đã có thể sử dụng khối lệnh xử lý theo chu kì, và chỉnh lại nó cho phù hợp với mục tiêu sử dụng. Trong hướng dẫn này, chúng tôi sẽ **định kì 30 giây** gửi dữ liệu lên OhStem server. Một lưu ý quan trọng, khối lệnh này chỉ được thực hiện khi có câu lệnh **xử lý sự kiện** trong phần **lặp mãi mãi**.

## 4 Gửi dữ liệu lên server

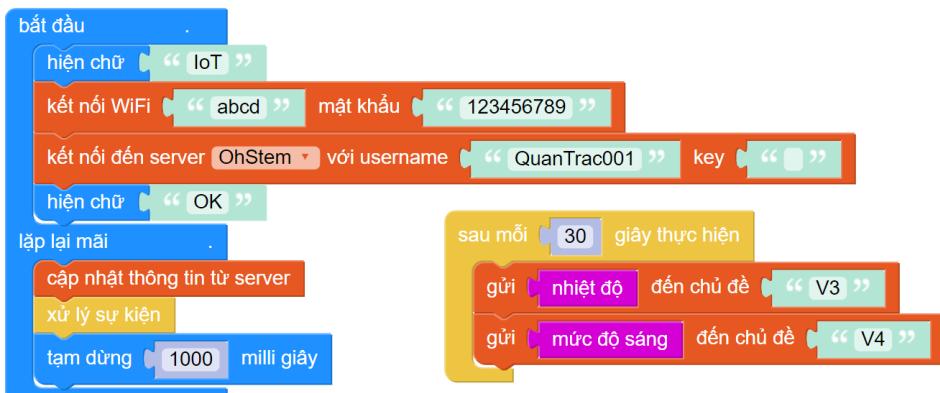
Sau khi đã có khôi lệnh xử lý theo chu kỳ, việc gửi dữ liệu lên OhStem server sẽ được đơn giản hóa. Các khôi lệnh chính để gửi dữ liệu nhiệt độ và ánh sáng lên OhStem server được trình bày như sau:



Hình 16.8: Câu lệnh gửi dữ liệu lên server

Các khôi lệnh liên quan đến cảm biến có thể được tìm thấy trong nhóm **NGÕ VÀO**. Câu lệnh còn lại nằm trong nhóm lệnh **MQTT**. Điều quan trọng nhất trong câu lệnh thứ 2 là bạn phải chỉ định chính xác **kênh dữ liệu** (hay còn gọi là chủ đề) trên server. Trong ví dụ của chúng ta là V3 cho nhiệt độ và V4 cho ánh sáng.

Ghép nối các khôi lệnh và đặt chúng vào phần xử lý định kỳ, chúng ta có chương trình hoàn chỉnh như sau:



Hình 16.9: Hoàn thiện chương trình

Nhóm lệnh **SỰ KIỆN** cung cấp cho bạn đọc một công cụ hiệu quả để xử lý các tác vụ liên quan đến thời gian. Bạn đọc hãy sử dụng nó một cách hiệu quả để có được kết quả tốt nhất. Chúng ta không nên tiếc thời gian tạm dừng 1 giây, mà làm cho hệ thống bị treo khi sử dụng thực tế. Chương trình cho bài này được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2C0x55QPuwTTD5pbrMHHqLKZAqD>

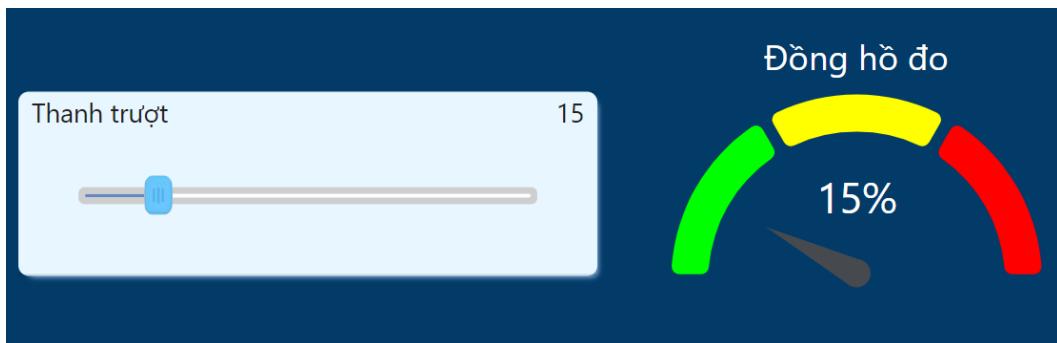
# CHƯƠNG 17

## Giao diện nâng cao



## 1 Giới thiệu

Trong bài hướng dẫn này, chúng ta sẽ làm việc với các đối tượng giao diện nâng cao trên bảng điều khiển IoT. Các thông tin về cấu hình của mỗi giao diện sẽ được trình bày chi tiết để bạn đọc có thể nắm bắt và vận dụng vào ứng dụng của mình. Tuy nhiên về căn bản, cấu hình cho **Kênh thông tin** là quan trọng nhất. Cụ thể, trong bài hướng dẫn này, chúng ta sẽ làm việc với 2 đối tượng giao diện mới, là **Thanh trượt** và **Đồng hồ đo**, như minh họa bên dưới:



Hình 17.1: Các giao diện nâng cao

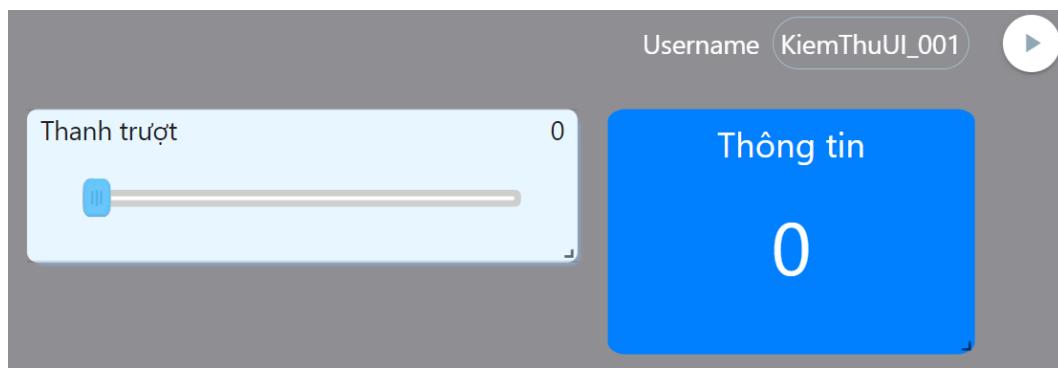
Kỹ năng quan trọng trong bài hướng dẫn này là cung cấp cho bạn đọc 1 công cụ để kiểm thử đối tượng giao diện mới, trước khi bắt đầu lập trình trên mạch Yolo:Bit. Điều này sẽ giúp bạn đọc có thể tự tìm hiểu và giảm sai sót khi lập trình.

Về mặt ứng dụng, **Thanh trượt** có thể dùng để điều khiển tốc độ của một động cơ, như máy quạt hoặc độ sáng của một bóng đèn. Trong khi đó, **Đồng hồ đo** khá thích hợp để hiển thị thông tin có giới hạn ngưỡng trên và dưới, chẳng hạn như độ ẩm không khí, có giá trị từ 0% đến 100%. Nội dung hướng dẫn của bài này tập trung vào các nội dung sau đây:

- Tìm hiểu đối tượng giao diện Thanh trượt
- Tự kiểm thử một đối tượng giao diện
- Tìm hiểu đối tượng giao diện Đồng hồ đo

## 2 Tự kiểm thử giao diện

Để bắt đầu tìm hiểu một đối tượng giao diện, bạn nên tạo mới một màn hình điều khiển IoT. Tuy nhiên, hãy kéo kèm nó ra màn hình với đối tượng giao diện **Thông tin**, như minh họa dưới đây:



Hình 17.2: Tự kiểm thử giao diện

Trong hình ảnh minh họa ở trên, chúng ta đã kéo ra đối tượng **Thanh trượt** và **Thông tin**. Bạn đọc không cần phải cầu hình bất kì thông tin gì của 2 đối tượng giao diện này, chỉ cần để nó mặc định là được.

Bước tiếp theo, chúng ta cần đặt **Username** cho bảng điều khiển, và sau đó có thể chạy thử màn hình điều khiển này. Bạn có thể bắt đầu tương tác với giao diện mới và kết quả sẽ tự động hiển thị trên **Thông tin**, như minh họa ở hình bên dưới:



Hình 17.3: Chạy thử bảng điều khiển IoT

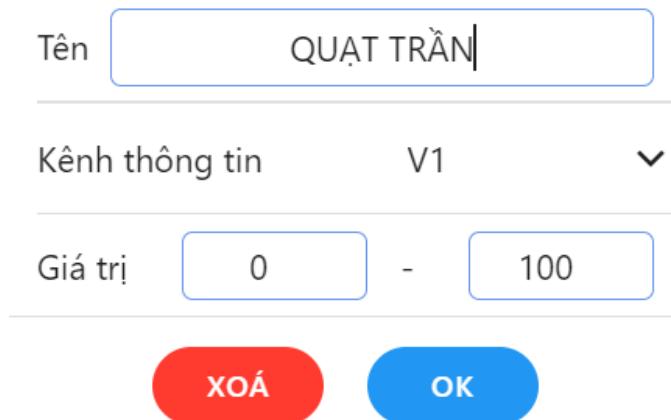
Điều này được giải thích như sau:

- Mỗi đối tượng giao diện được tạo ra trên màn hình điều khiển, mặc định nó sẽ **liên kết với kênh dữ liệu V1**.
- Với đối tượng có thể tương tác, như nút nhấn hay như thanh trượt trong bài này, nó sẽ **gửi dữ liệu lên kênh dữ liệu V1**.
- Giao diện Thông tin liên kết với V1, nên nó sẽ thay đổi dữ liệu theo tương ứng.

Khi lập trình cho mạch Yolo:Bit, thực ra nó cũng đóng vài trò hệt như đối tượng Thông tin. Những gì hiển thị trên Thông tin sẽ chính xác là những gì Yolo:Bit sẽ nhận được. Tuy nhiên, dữ liệu hiển thị dù là con số, vẫn là dữ liệu kiểu chuỗi. Khi điều khiển thiết bị, thông thường chúng ta phải đổi dữ liệu từ chuỗi sang số.

### 3 Cấu hình cho Thanh trượt

Cũng tương tự như các đối tượng giao diện khác, chúng ta cấu hình cho Thanh trượt bằng cách nhấn chuột trái vào nó, giao diện sau đây sẽ hiện lên:



Hình 17.4: Cấu hình cho Thanh trượt

Giao diện này chỉ có 3 thông số cần lưu ý:

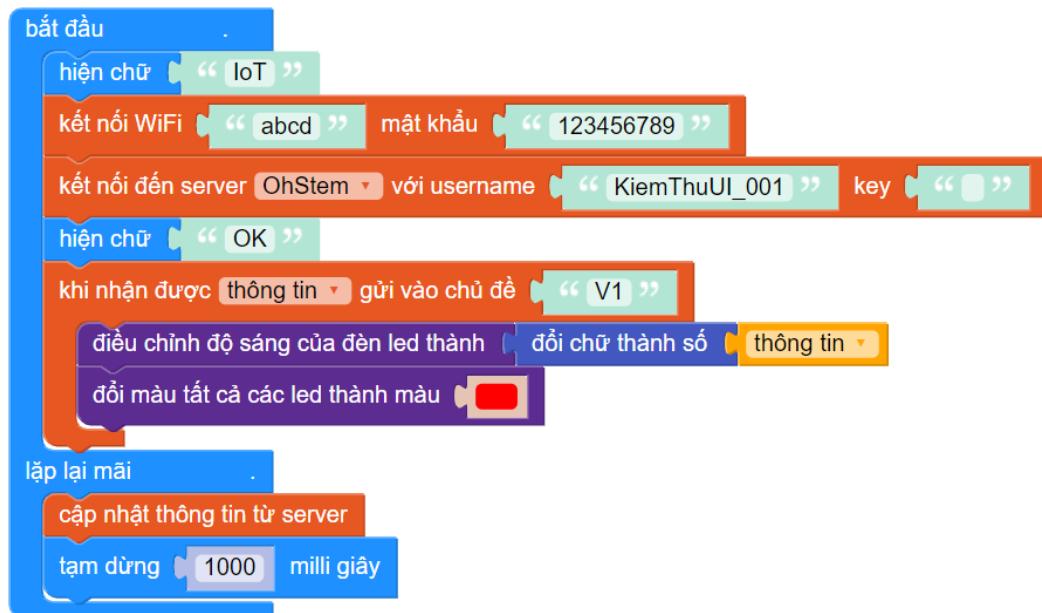
- Tên: là giá trị sẽ hiện như tiêu đề của Thanh trượt. Mặc dù không phải là thông tin dùng cho việc lập trình, nó lại là thông tin quan trọng để trình diễn. Bạn đọc sẽ thường thay đổi thông tin này để nó gần gũi nhất với thiết bị cần điều khiển ở mạch Yolo:Bit.
- Kênh dữ liệu: luôn luôn là thông tin quan trọng cho mỗi đối tượng giao diện. OhStem hỗ trợ 20 kênh dữ liệu khác nhau để bạn đọc có thể tùy chọn sử dụng cho dự án của mình.
- Giá trị: Cận trên và cận dưới của thanh trượt. Tùy vào ứng dụng mà bạn đọc có thể tùy chỉnh cho 2 giá trị này. Với một động cơ, giá trị mặc định của nó thường là 0 đến 100, tương trưng cho công suất tính theo phần trăm của động cơ.

Việc lập trình ở phía mạch Yolo:Bit khá đơn giản đối với thanh trượt, chỉ có một lưu ý quan trọng là giá trị lưu trong biến thông tin là **dữ liệu chuỗi**. Trong nhiều trường hợp, dữ liệu này thường được chuyển qua dữ liệu số, như ví dụ sau đây:



Hình 17.5: Xử lý cơ bản với thanh trượt

Câu lệnh **đổi chữ thành số** có thể được tìm thấy trong phần **TÍNH TOÁN**. Khi ghép khôi lệnh đăng kí này vào chương trình hoàn chỉnh, chúng ta có một chương trình minh họa như sau.

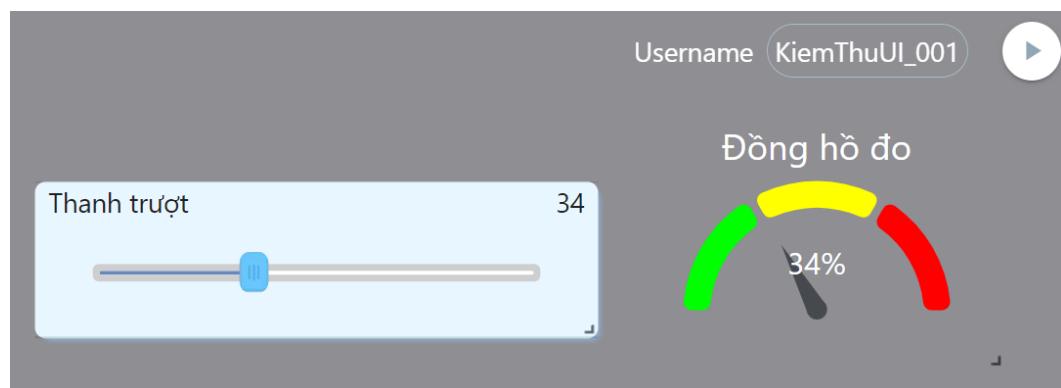


Hình 17.6: Chương trình xử lý trên Yolo:Bit

Trong chương trình ví dụ này, độ sáng của đèn trên Yolo:Bit sẽ được điều chỉnh theo thanh trượt. Bạn đọc cần lưu ý cung cấp thông tin username trong câu lệnh **kết nối đến server OhStem** cho chính xác là chương trình sẽ hoạt động tốt.

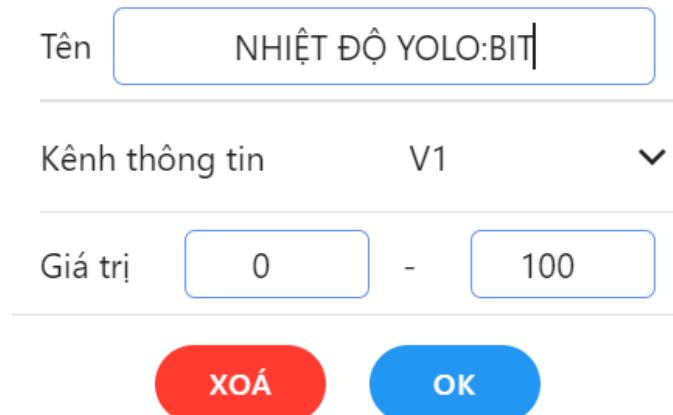
## 4 Giao diện Đồng hồ đo

Với đối tượng giao diện này, do chúng ta không tương tác được với nó (chẳng hạn như nhấn chuột hoặc kéo thả trên nó), nên không thể kéo thả kèm với giao diện thông tin. Thay vào đó, chúng ta sẽ kiểm tra nó với một đối tượng giao diện có thể tương tác được, và Thanh trượt là một ví dụ điển hình nhất. Bạn đọc có thể thiết kế một giao diện kiểm thử như sau:



Hình 17.7: Kiểm thử giao diện Đồng hồ đo

Đặt tên cho Username và bắt đầu chạy thử giao diện điều khiển IoT, chúng ta sẽ dễ dàng hình dung được tính năng của nó. Vì không tương tác được, Đồng hồ đo chủ yếu được sử dụng để hiện thông tin từ mạch Yolo:Bit gửi lên. Các thông số cấu hình quan trọng cho nó được trình bày như sau:

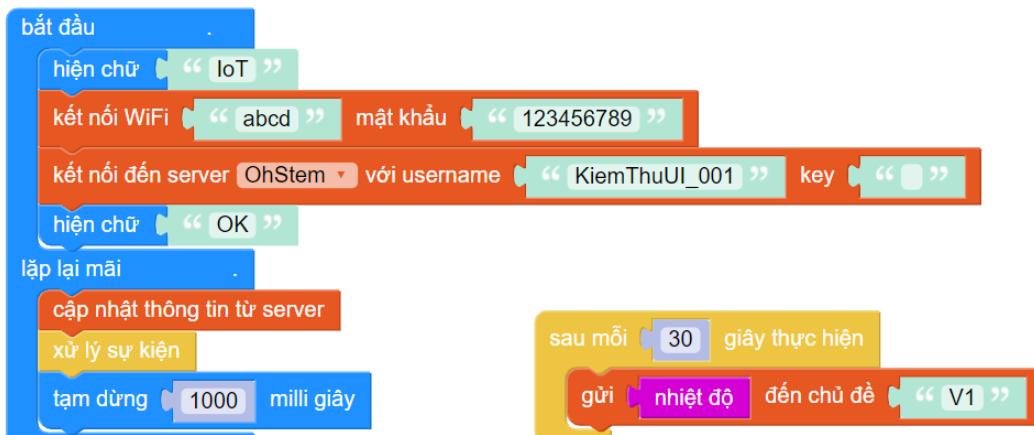


Hình 17.8: Cấu hình cho Đồng hồ đo

Giao diện này chỉ có 3 thông số cần lưu ý:

- Tên: là giá trị sẽ hiện như tiêu đề của giao diện. Bạn đọc sẽ thường thay đổi thông tin này để nó thể hiện được thông tin gửi lên từ Yolo:Bit một cách trực quan nhất.
- Kênh dữ liệu: luôn luôn là thông tin quan trọng cho mỗi đối tượng giao diện. OhStem hỗ trợ 20 kênh dữ liệu khác nhau để bạn đọc có thể tùy chọn sử dụng cho dự án của mình.
- Giá trị: Cận trên và cận dưới của thanh trượt. Tùy vào ứng dụng mà bạn đọc có thể tùy chỉnh cho 2 giá trị này. Với thông tin về độ ẩm, tầm giá trị của nó từ 0% đến 100%.

Để ví dụ cho việc gửi dữ liệu định kì lên giao diện đồng hồ, chúng tôi tạm sử dụng thông tin về nhiệt độ của Yolo:Bit, như sau:



Hình 17.9: Chương trình gửi dữ liệu lên kênh V1

Trong tương lai, khi gửi dữ liệu lên Đồng hồ đo, bạn nên chọn thông tin có đơn vị là phần trăm (%). Bạn đọc sẽ thấy có hiệu ứng đặc biệt, khi thanh trượt cũng sẽ tự động chạy theo mỗi khi giá trị về nhiệt độ từ Yolo:Bit được gửi lên, như minh họa ở hình bên dưới.



Hình 17.10: Thanh trượt cũng cập nhật theo Đồng hồ đo

Chúng ta có thể có một số kết luận sau đây:

- Đối với đối tượng giao diện không thể tương tác, nó sẽ thường được dùng để hiển thị thông tin từ mạch Yolo:Bit gửi lên.
- Đối với đối tượng giao diện có thể tương tác, nó thường sẽ được dùng để điều khiển một thiết bị trên Yolo:Bit. Tuy nhiên, nó vẫn có thể cập nhật hiệu ứng trong trường hợp mạch Yolo:Bit gửi dữ liệu lên đúng kênh dữ liệu mà đối tượng này đang liên kết.

Kết luận thứ 2 sẽ là thông tin quan trọng mà chúng ta sẽ sử dụng trong tương lai. Chẳng hạn như mạch Yolo:Bit đang nắm quyền bật một bóng đèn, nó sẽ gửi lên server thông tin về trạng thái của thiết bị (là 1 hoặc 0). Lúc này, nút nhấn vốn được thiết kế để điều khiển bóng đèn từ xa, sẽ cập nhập theo trạng thái của thiết bị bên dưới. Lúc đó hệ thống của chúng ta mới được gọi là đồng bộ và theo đúng kiến trúc của một ứng dụng kết nối vạn vật.



# CHƯƠNG 18

## Gửi tin nhắn với Telegram



## 1 Giới thiệu

Nếu như trước đây, chỉ có máy tính mới có thể kết nối được vào mạng Internet, thì hiện tại, nhờ sự ra đời của kết nối vạn vật, mà nhiều thiết bị đã có thể tham gia vào mạng Internet. Bên cạnh các điện thoại thông minh, máy tính bảng, các thiết bị phần cứng nhỏ gọn ngày nay, như mạch Yolo:Bit hoàn toàn đủ sức mạnh để tham gia vào thế giới kết nối 4.0 và sử dụng các dịch vụ tuyệt vời từ nó.



Hình 18.1: Gửi tin nhắn với Telegram

Nếu để ý kĩ hơn, bạn sẽ thấy thế giới kết nối vạn vật đã và đang thay đổi nhiều thói quen trong cuộc sống thường nhật. Chúng ta đã ít gọi điện thoại bằng số cố định, mà thay vào đó là các dịch vụ gọi trực tuyến như Zalo hay Viber. Chúng ta cũng ít nhắn tin văn bản SMS bằng sim điện thoại, mà thay vào đó là các ứng dụng rất phong phú về tin nhắn, như Facebook, Zalo và cả Telegram. Các ứng dụng nhắn tin hiện tại rất linh hoạt, cho phép chúng ta có thể sử dụng cùng số điện thoại để nhắn tin trên các thiết bị khác nhau, trong cùng một thời gian.

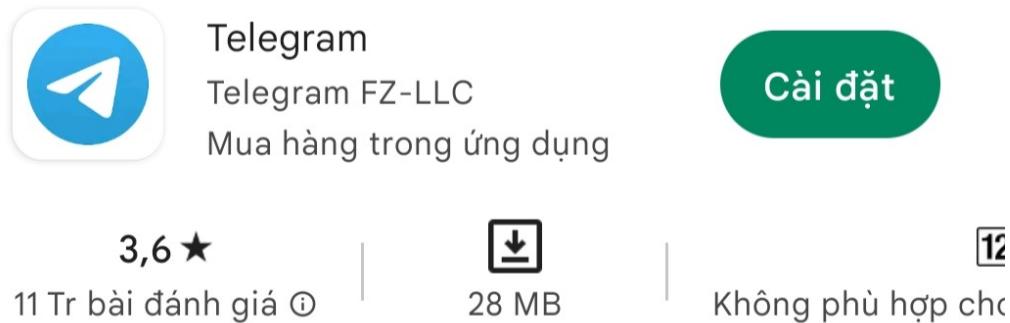
Trong bài viết này, chúng tôi sẽ hướng dẫn bạn đọc cách gửi tin nhắn lên Telegram một cách đơn giản, thông qua mạch lập trình mini Yolo:Bit. Tính năng này có thể được sử dụng để gửi thông báo đến người quản lý trong một số ứng dụng thông minh mà bạn hiện thực. Chẳng hạn như gửi tin nhắn nhắc nhở đã đến giờ uống thuốc hoặc gửi tin nhắn cảnh báo khi có vấn đề gì bất thường trong ứng dụng nhà thông minh. Các nội dung hướng dẫn trong bài này như sau:

- Cài đặt Telegram và đăng ký tài khoản
- Tạo nhóm chat với BotFather
- Hiện thực chương trình gửi tin nhắn trên Yolo:Bit

## 2 Cài đặt Telegram

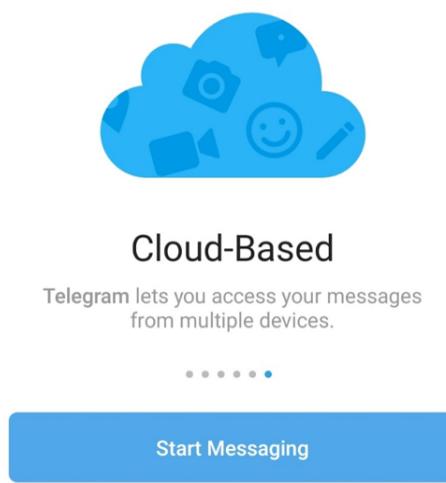
Về bản chất, phần mềm Telegram cũng giống như các phần mềm nhắn tin và gọi điện trực tuyến khác như Zalo, Viber hay WhatsApp. Trước tiên, bạn cần phải cài đặt phần mềm này trên thiết bị di động. Sau đó, bạn cần tạo một tài khoản và xác thực tài khoản của mình trước khi có thể bắt đầu nhắn tin và gọi điện thoại trực tuyến.

Phần mềm Telegram khá phổ biến cho điện thoại di động hoặc máy tính bảng. Nó hỗ trợ cho cả 2 hệ điều hành thông dụng hiện nay là Android và iOS. Trên thiết bị sử dụng Android, phần mềm có thể dễ dàng được tìm thấy trong CHPlay, như minh họa ở hình bên dưới:



Hình 18.2: Phần mềm Telegram trên cửa hàng ứng dụng

Sau khi tải và cài đặt ứng dụng thành công, màn hình đầu tiên của ứng dụng sẽ hiện ra như sau:



Hình 18.3: Giao diện đầu tiên của ứng dụng Telegram

Tiếp theo, bạn tiến hành nhấn vào **Start Messaging** và bắt đầu điền số điện thoại của mình để đăng ký tài khoản:



Hình 18.4: Đăng ký tài khoản

Sau khi đã nhập số điện thoại, bạn nhấn vào biểu tượng mũi tên bên dưới. Telegram sẽ gửi mã xác thực về số điện thoại bạn đã đăng ký. Bạn chỉ cần nhập mã xác thực vào là đã hoàn tất quá trình đăng ký tài khoản. Bây giờ, bạn có thể sử dụng Telegram để nhận tin nhắn từ mạch Yolo:Bit gửi lên.

### 3 Liên kết ứng dụng Telegram trên Website

Trước khi lập trình trên mạch Yolo:Bit chúng ta cần 2 thông tin quan trọng, liên quan đến token và id của nhóm chat. Tuy nhiên, 2 thông tin này chỉ có với phiên bản web của phần mềm Telegram. Do vậy, chúng ta cần phải liên kết ứng dụng Telegram đang cài trên điện thoại với phiên bản mềm trên website.

Đầu tiên, sử dụng máy tính, từ trình duyệt web thông thường, bạn tới đường dẫn <https://web.telegram.org/z/>. Giao diện sau đây sẽ hiện ra, với mã QR code ở giữa màn hình:

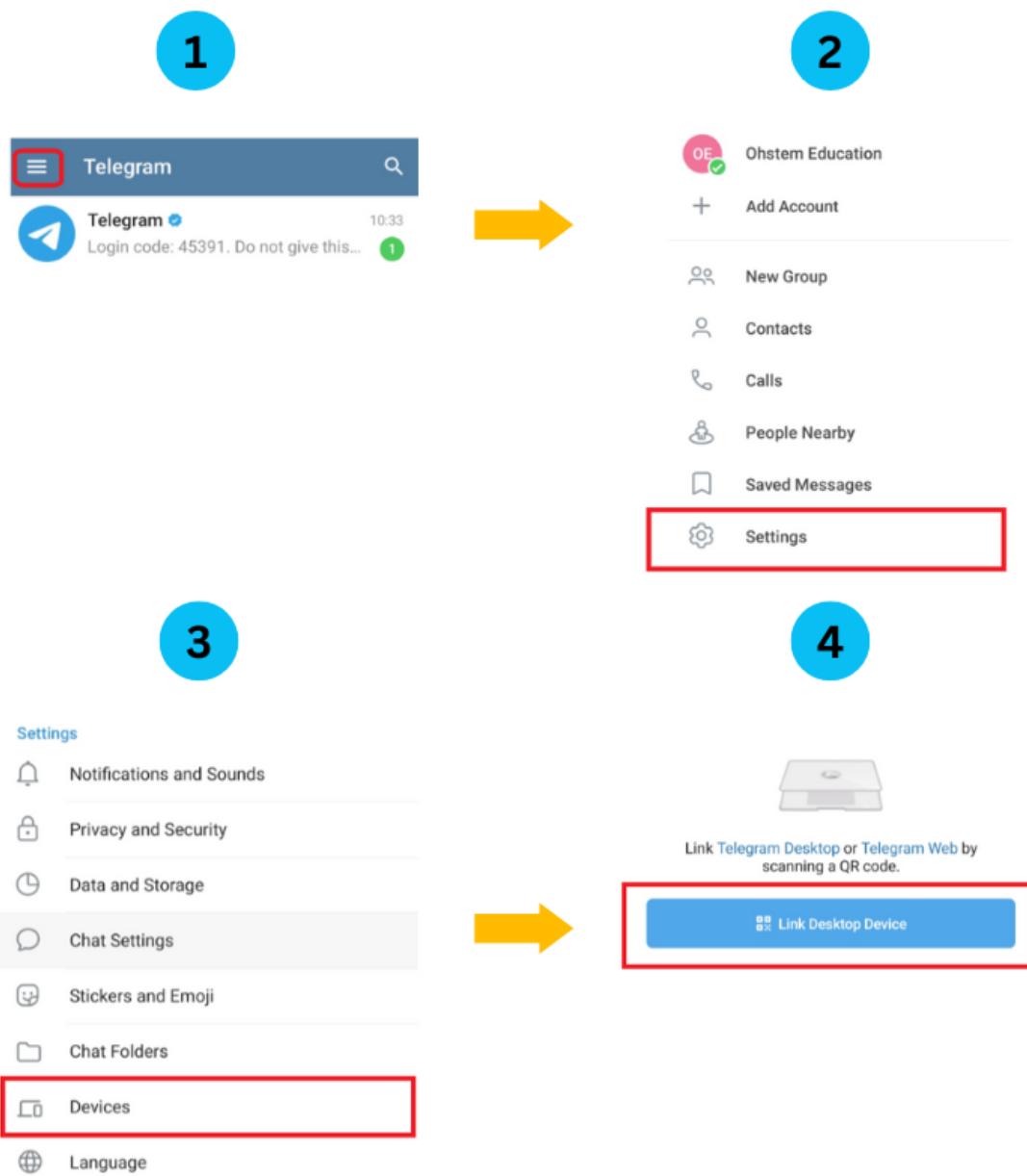


Log in to Telegram by QR Code

- 1 Open Telegram on your phone
- 2 Go to Settings → Devices → Link Desktop Device
- 3 Point your phone at this screen to confirm login

Hình 18.5: Liên kết đăng nhập với phiên bản web

Tiếp, chúng ta sẽ tiến hành liên kết thiết bị di động với Telegram trên Website. Để thực hiện, bạn nhấn vào phần biểu tượng menu (hình 3 dấu gạch ngang), sau đó chọn **Settings » Devices » Link Desktop Device**, như minh họa ở các bước sau đây:



Hình 18.6: Liên kết với Website

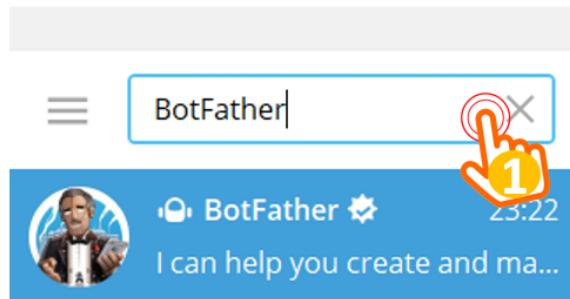
Tới đây, màn hình của thiết bị di động sẽ bật camera lên. Bạn cần quét mã QR trên màn hình máy tính của mình để đăng nhập Telegram trên máy tính. Quá trình liên kết giữa ứng dụng trên di động và trên website đến bước này là kết thúc.

## 4 Tạo BotFather

Để mач Yolo:Bit có thể gửi tin nhắn lên Telegram, nó sẽ đóng vai trò của một Bot-Father. BotFather có thể hiểu là một người ảo trên Telegram mà mач Yolo:Bit có thể liên kết với nó để nhận tin cho chúng ta. Quy trình để tạo ra một BotFather

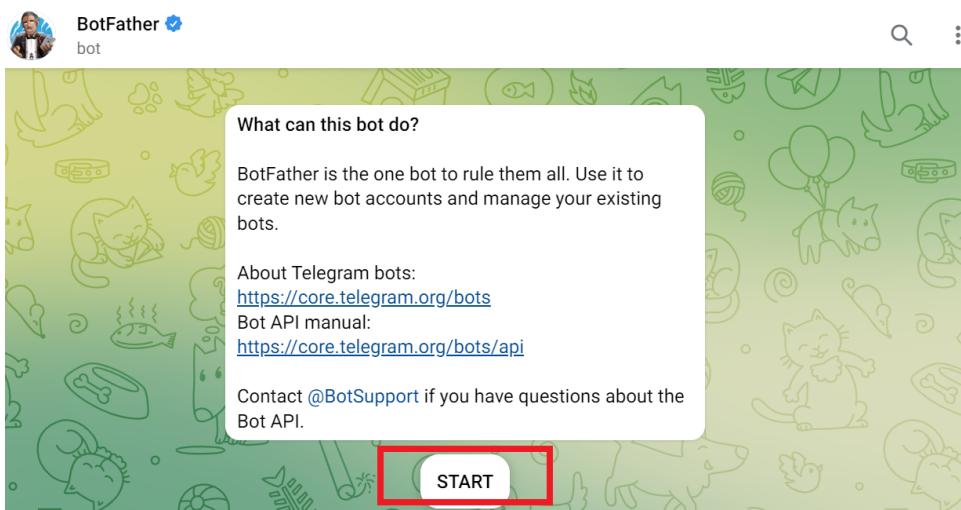
được trình bày từng bước bên dưới.

Từ ứng dụng Telegram trên di động, bạn tiến hành nhập từ khóa **BotFather** ở ô tìm kiếm, sau đó nhấp vào biểu tượng BotFather để cài đặt, như hướng dẫn sau đây.



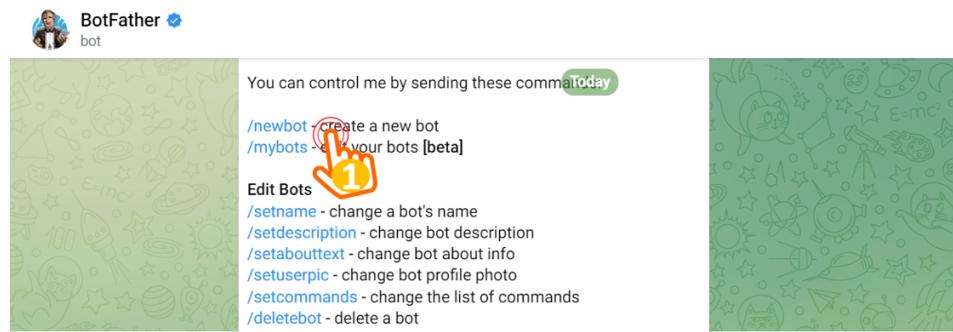
Hình 18.7: Tìm kiếm và cài đặt BotFather

Màn hình sẽ xuất hiện một giao diện mới. Tại đây, bạn hãy nhấn vào nút **START** để kích hoạt BotFather.



Hình 18.8: Nhấn nút START để bắt đầu

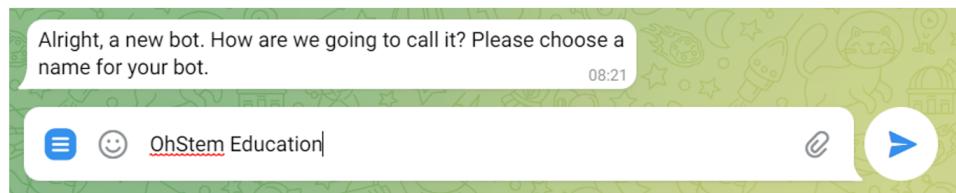
Bước này chỉ xuất hiện lần đầu tiên khi bạn cài BotFather. Sau bước này, BotFather trở thành 1 thành viên trong danh sách danh bạ Telegram của bạn. Tiếp theo, với giao diện mới xuất hiện, tiếp tục nhấp vào **new Bot** để tạo Bot mới:



Hình 18.9: Nhấn vào /newbot để tạo Bot mới

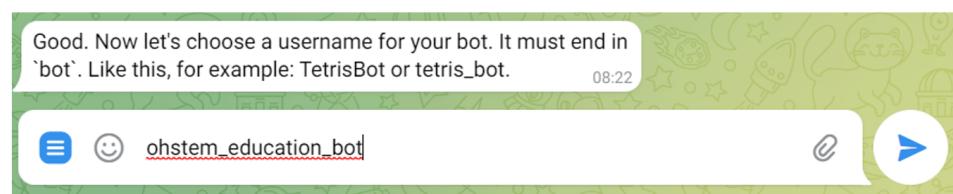
Thực ra việc tạo một Bot mới chỉ đơn giản là nhấp cho tài khoản BotFather dòng lệnh **/newbot**. Đây là một thao tác nhanh cho bạn trong tương lai, khi bạn muốn tạo thêm Bot mới cho ứng dụng của mình.

Tiếp theo, hệ thống sẽ hỏi bạn đặt tên cho Bot, ở đây chúng tôi lấy tên là **OhStem Education**, và gửi tin nhắn này cho nó như một câu chat bình thường, như minh họa ở hình sau:



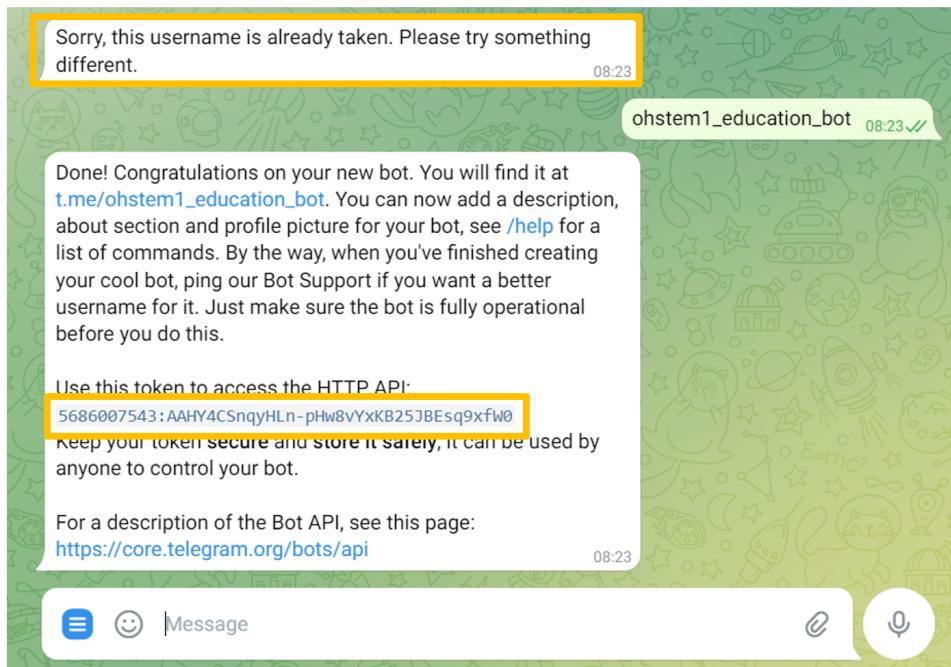
Hình 18.10: Đặt tên cho BotFather

Sau khi đặt tên cho BotFather, bạn cần chọn cho nó 1 tên đăng nhập (username) theo cú pháp **name\_bot**. Chủ yếu là bạn sẽ viết thành chữ thường, cách nhau bằng dấu gạch và kết thúc bằng **bot** là được. Như minh họa bên dưới, chúng tôi chọn tên đăng nhập cho nó là **ohstem\_education\_bot**:



Hình 18.11: Chọn username cho BotFather

Khác với tên, username cần phải chưa được sử dụng. Trong một số trường hợp, bạn sẽ gặp thông báo là username này đã có rồi và bắt buộc phải chọn username khác, như minh họa ở hình dưới đây:

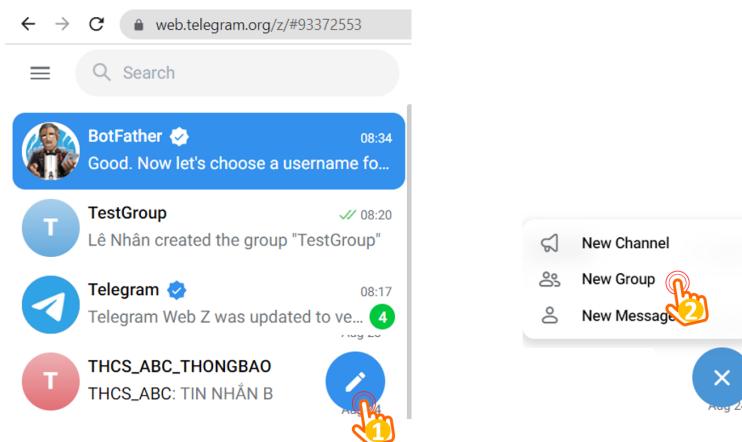


*Hình 18.12: Username bị trùng và phải chọn lại*

Sau khi chọn một username cho Bot thành công, dòng thông báo **Done! Congratulations** sẽ xuất hiện, bạn chú ý đến phần token của nó. Thông tin token sẽ được cung cấp trong thông báo của BotFather và nằm bên dưới dòng **Use this token....**. Bạn cần sao chép thông tin này lại và sử dụng nó cho việc lập trình.

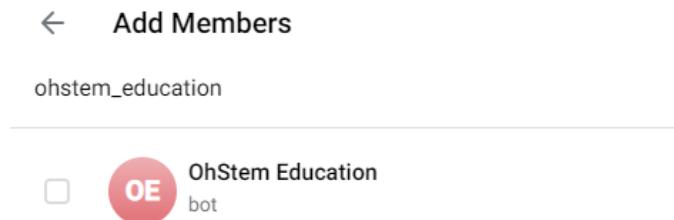
## 5 Tạo nhóm chat với BotFather

Thực ra BotFather, hay mач Yolo:Bit trong tương lai, nó không thể gửi tin nhắn đến từng thành viên riêng lẻ trong danh bạ của bạn được. Chúng ta bắt buộc phải tạo nhóm chat có BotFather và các thành viên được phép nhận tin nhắn cho một ứng dụng nhắn tin thông báo. Vẫn tiếp tục sử dụng phần mềm Telegram trên website, để tiện cho việc thao tác, bạn di chuyển chuột vào phần danh bạ, biểu tượng tạo nhóm chat sẽ xuất hiện, như minh họa sau đây:



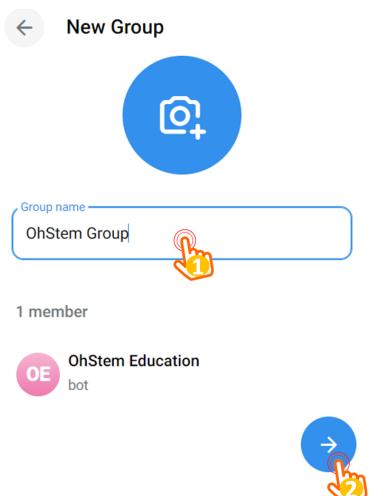
*Hình 18.13: Username bị trùng và phải chọn lại*

Bạn bắt đầu thêm các thành viên vào nhóm, và hiển nhiên phải thêm BotFather mà bạn đã tạo. Bạn hãy tìm BotFather của mình bằng **username** của nó để kết quả được chính xác nhất.



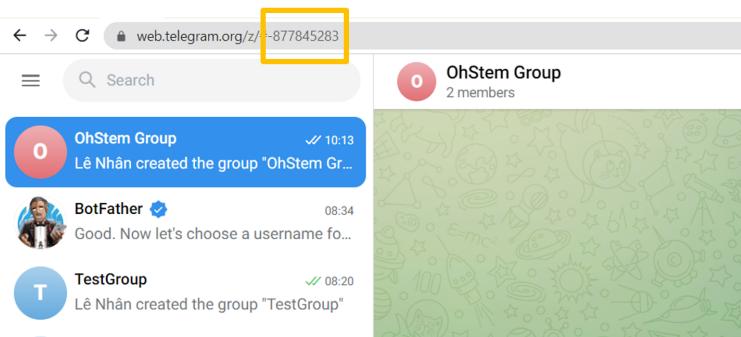
Hình 18.14: Thêm thành viên và BotFather vào nhóm chat

Sau khi đã thêm đủ thành viên, bạn đặt tên cho nhóm chat, như minh họa ở hình bên dưới:



Hình 18.15: Hoàn tất việc tạo nhóm chat trên Telegram

Sau khi tạo thành công nhóm chat, hãy để ý vào các con số trên địa chỉ của trình duyệt. Đây chính là mã của nhóm chat (ID kênh chat) mà bạn cần phải sao chép lại để lập trình. Đôi lúc mã chat này là một số âm, như thông tin trong hình dưới đây.



Hình 18.16: Chú ý giá trị ID

Như vậy là chúng ta đã cấu hình xong phần nhận tin nhắn ở Telegram. Chúng ta sẽ chuyển sang phần lập trình gửi tin nhắn ở Yolo:Bit. Hai thông tin mà bạn cần phải lưu lại trong quá trình này là **token** và **ID** của kênh chat.

## 6 Lập trình gửi tin nhắn trên Yolo:Bit

Để lập trình, bạn cần thư viện mở rộng để gửi tin nhắn. OhStem đã xây dựng sẵn thư viện mở rộng phục vụ mục đích này. Từ môi trường lập trình thông thường, bạn thêm thư viện bằng cách chọn vào **MỞ RỘNG**. Trong cửa sổ mới, hãy tìm đến thư viện **HTTP**, như minh họa ở hình ảnh bên dưới.



Hình 18.17: Thêm thư viện HTTP vào môi trường lập trình

Bạn tiến hành kết nối Yolo:Bit với máy tính, sau đó nhấn vào thư viện để tải chúng về. Chúng ta sẽ sử dụng thư viện này để gửi tin nhắn lên Telegram. Chúng ta chủ yếu sẽ làm việc với khôi lệnh sau đây:

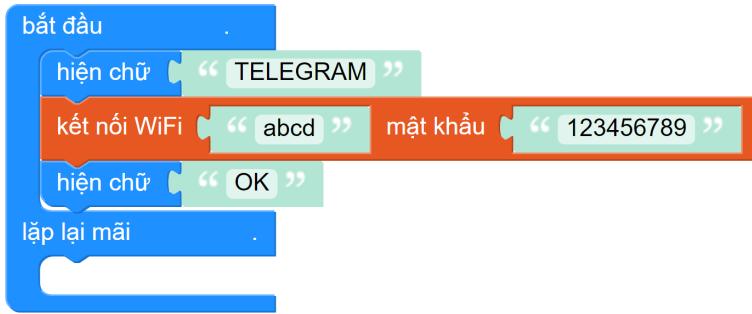
gửi lên Telegram token “ ” id “ ” tin nhắn “ ”

Hình 18.18: Khôi lệnh gửi tin nhắn lên Telegram

Để gửi tin nhắn lên Telegram, trước hết, chúng ta hãy kết nối Yolo:Bit với WiFi. Ở đây, bạn cần đổi tên WiFi và mật khẩu sao cho đúng với WiFi bạn đang sử dụng. Chúng tôi sẽ lấy ví dụ mẫu là WiFi có tên **abcd**, với mật khẩu là **123456789**, như sau:

Ngoài câu lệnh kết nối WiFi chính, các câu lệnh khác sẽ có tác dụng:

- Câu lệnh hiện chữ "TELEGRAM" sẽ giúp chúng ta biết được chương trình đã được gửi vào mạch Yolo:Bit.
- Câu lệnh hiện chữ 'OK' sẽ giúp chúng ta kiểm tra việc kết nối WiFi đã thành công hay chưa. Nếu mạch Yolo:Bit hiện lên dòng chữ OK thì thiết bị của chúng ta đã kết nối WiFi thành công. Ngược lại, nếu mạch không hiển thị "OK", chúng ta cần xem lại kết nối WiFi của mình.



Hình 18.19: Kết nối Yolo:Bit với WiFi

Sau khi đã kết nối WiFi thành công, chúng ta sẽ làm một chương trình đơn giản để kiểm tra thử tính năng gửi tin nhắn lên Telegram, bằng 2 nút A và B. Chương trình sẽ gửi các tin nhắn đơn giản khi nhấn nút A hoặc B.



Hình 18.20: Nhấn nút B để gửi tin nhắn

Cho đến bước này, hình ảnh hoàn chỉnh của chương trình sẽ như sau:



Hình 18.21: Chương trình hoàn chỉnh

Bây giờ, khi nhấn nút A hoặc B, bạn sẽ nhận được tin nhắn từ mạch Yolo:Bit, từ cả ứng dụng trên web lẫn phần mềm đang được cài trên điện thoại di động. Tính năng này sẽ rất hữu ích cho các ứng dụng thời đại, khi việc gửi thông báo hoặc cảnh báo sẽ trở nên dễ dàng hơn bao giờ hết.

Bạn đọc có thể tự phát triển thêm chương trình ở đây, bằng cách kết hợp thêm với thời gian, để làm nên ứng dụng nhắc nhở uống thuốc đúng giờ chẳng hạn. Phần kiểm tra thời gian và gửi tin nhắn sẽ được hiện thực trong **lặp mãi mãi**, với chu kỳ kiểm tra là 30 giây một lần. Đây là con số hợp lý cho một ứng dụng chỉ cần kiểm

tra giờ và phút. Bạn cũng cần phải đổi các thông tin về thời gian sang kiểu số, trước khi làm các bài toán so sánh. Hình ảnh gợi ý cho chương trình này như sau:



Hình 18.22: Chương trình nhắc nhở uống thuốc đúng giờ

Chương trình tham khảo cho bạn đọc được chia sẻ ở đường link sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2HIHBorNvrBIJcRt36TfTaCkZj5>

# **Phần IV**

## **Trí Tuệ Nhân Tạo**



# CHƯƠNG 19

## Nhận diện giọng nói



## 1 Giới thiệu

Trong hệ sinh thái Yolo:Bit, chúng tôi hỗ trợ đầy đủ các công nghệ 4.0 cho các hoạt động sáng tạo trong giáo dục nói chung và STEM nói riêng. Bên cạnh công nghệ việc tích hợp mạnh mẽ khả năng kết nối Internet cho các ứng dụng Kết nối vạn vật, Trí tuệ nhân tạo là công cụ không thể thiếu cho các ứng dụng thông minh và tự hành.

Thực ra, trí tuệ nhân tạo, hay còn được gọi là AI (Artificial Intelligence), có nhiều mức độ khác nhau. Trong bài hướng dẫn đầu tiên này, chúng ta sẽ tìm hiểu cách sử dụng công cụ chuyển đổi văn bản thành giọng nói. Đây là công nghệ được sử dụng trong công cụ tìm kiếm bằng giọng nói mà bạn có thể dễ dàng tìm thấy trong điện thoại hay tivi thông minh mà chúng ta đang sử dụng ngày nay. Mặc dù điều khiển bằng giọng nói chưa phải là một tính năng hấp dẫn trong một ứng dụng nông nghiệp thông minh, nó vẫn là một công cụ để bạn đọc sáng tạo trong tương lai.

Thực ra các mạch điện phần cứng nói chung và mạch Yolo:Bit nói riêng khó có khả năng thực hiện tính năng trí tuệ nhân tạo bởi sức mạnh và ngoại vi của nó không đủ. Chẳng hạn như với ứng dụng nhận diện giọng nói trong bài này, mạch Yolo:Bit không hề có micro để thu âm thanh. Tài nguyên phần cứng lẩn tốc độ bộ xử lý cũng không thể đủ để chạy **bộ não nhân tạo**.

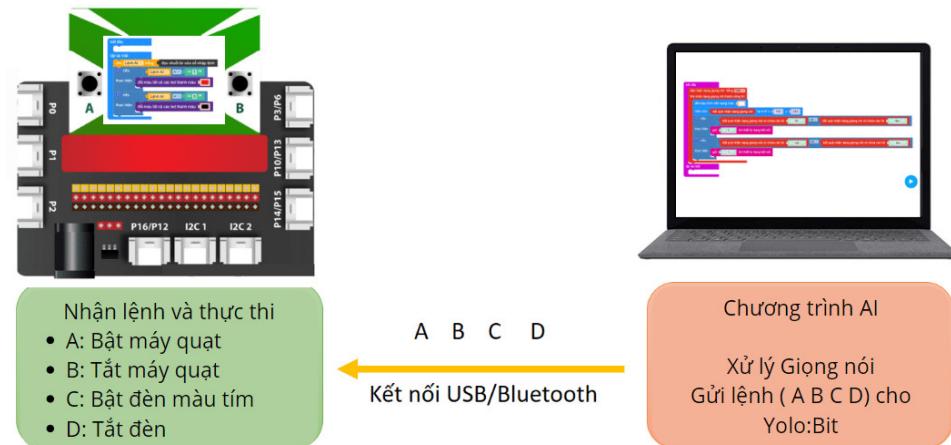
Trong bài hướng dẫn này, bên cạnh việc sử dụng công cụ chuyển đổi văn bản thành giọng nói, được hỗ trợ trong hệ sinh thái Yolo:Bit, chúng tôi cũng sẽ trình bày kiến trúc chung cho một ứng dụng trí tuệ nhân tạo. Chúng ta sẽ xây dựng một ứng dụng điều khiển bằng giọng nói, chẳng hạn như **bật máy bơm** hay **bật đèn màu tím**, những chức năng có thể ứng dụng được trong nông nghiệp thông minh. Các mục tiêu trong bài hướng dẫn này như sau:

- Kiến trúc của một ứng dụng AI
- Môi trường lập trình cho trí tuệ nhân tạo
- Lập trình nhận dạng giọng nói
- Lập trình nhận diện từ lệnh điều khiển

## 2 Kiến trúc ứng dụng AI

Để có thể thực thi một bộ não AI, chúng ta cần một hệ thống có tài nguyên đủ mạnh, bao gồm tốc độ CPU và bộ nhớ RAM. Một hệ thống như vậy mới có thể thực thi được các tác vụ phức tạp của bộ não AI (chủ yếu là phép nhân ma trận) trong thời gian cho phép. Do đó, máy tính hoặc điện thoại di động mới có thể phù hợp để thực thi bộ não AI.Thêm nữa, trên các thiết bị này thường có sẵn micro lẩn webcam, để có thể dễ dàng thu được dữ liệu đầu vào (âm thanh và hình ảnh) cho bộ não AI.

Yolo:Bit thì ngược lại, nó không có những ưu thế như máy tính hay điện thoại di động, nếu muốn so sánh về sức mạnh CPU lẫn bộ nhớ RAM. Yolo:Bit cũng khó có khả năng tích hợp thêm micro và webcam. Tuy nhiên, nó lại rất mạnh mẽ trong việc điều khiển thiết bị, chẳng hạn như bật đèn hoặc tắt một máy bơm.



Hình 19.1: Kiến trúc chung của ứng dụng dựa trên AI

Như vậy, một ứng dụng điều khiển thiết bị dựa trên công nghệ AI sẽ là sự kết hợp của 2 hệ thống trên: Máy tính sử dụng cho AI và Yolo:Bit để điều khiển thiết bị. Hai thiết bị này được liên kết với nhau thông qua kết nối USB hoặc Bluetooth. Chương trình AI chạy trên máy tính sẽ chịu trách nhiệm cho phần xử lý phức tạp, chẳng hạn như việc nhận dạng giọng nói. Sau đó, nó sẽ gửi các lệnh đơn giản cho Yolo:Bit dưới dạng các ký tự đơn giản, như là A, B, C hoặc D. Cuối cùng, mạch Yolo:Bit sẽ nhận lệnh và thực thi với các tác vụ tương ứng.

### 3 Môi trường lập trình AI

Chúng ta sẽ bắt đầu xây dựng chương trình AI trên máy tính trước. Việc chuyển nó sang điện thoại di động hoàn toàn tương tự và sẽ không được trình bày trong hướng dẫn này. Cùng mục đích với việc phổ biến việc học lập trình, môi trường lập trình cho trí tuệ nhân tạo cũng được triển khai trực tuyến trong cùng hệ sinh thái của OhStem. Để có thể bắt đầu lập trình nhận diện giọng nói, chúng ta truy cập vào đường dẫn sau đây:

<https://app.ohstem.vn/>

Với giao diện được hiện ra, chúng ta chọn tiếp vào phần **Lập trình AI**, như hướng dẫn ở hình sau đây:



Hình 19.2: Giao diện lập trình Trí tuệ nhân tạo

Giao diện lập trình kéo thả sau đây sẽ hiện ra, với khối lệnh đặt trưng **bắt đầu ...** **lặp mãi mãi** cho các ứng dụng của chúng ta.



Hình 19.3: Trang lập trình kéo thả cho AI

## 4 Lập trình nhận diện giọng nói

Các câu lệnh chính cho việc lập trình ở bài này sẽ thuộc nhóm lệnh **GIỌNG NÓI**, có màu đỏ. Từng bước để xây dựng chương trình được trình bày bên dưới.

**Bước 1:** Chuẩn bị các câu lệnh nhận diện giọng nói.

Bạn đọc hãy kéo 2 khối lệnh sau ra màn hình lập trình. Chúng ta chưa cần ghép vội chúng lại với nhau, như sau:

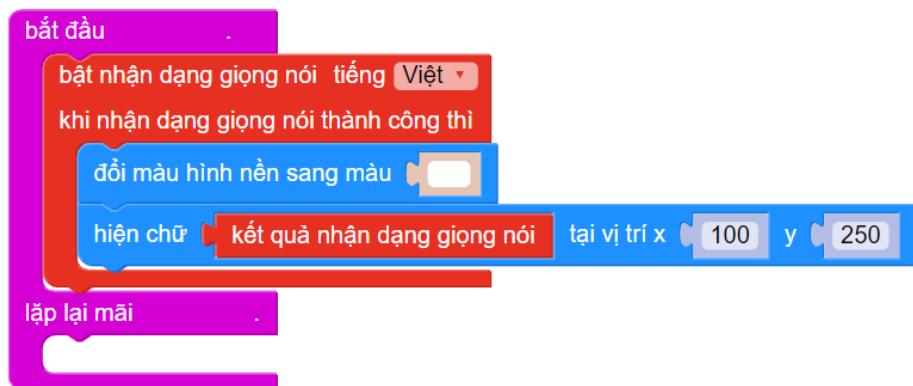


Hình 19.4: Khối lệnh nhận diện giọng nói

Câu lệnh đầu tiên có chức năng kích hoạt bộ chuyển đổi giọng nói thành văn bản, hay nói ngắn gọn là nhận diện giọng nói. Kết quả nhận dạng sẽ được lưu trong khối lệnh thứ 2.

### Bước 2: Xuất kết quả nhận diện ra màn hình.

Tiếp theo, sử dụng thêm với 2 câu lệnh trong mục **HIỂN THỊ** để có được chương trình sau đây.



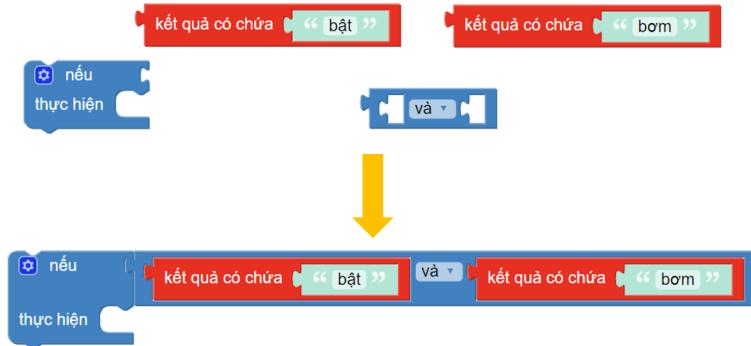
Hình 19.5: Hiển thị kết quả giọng nói

Câu lệnh đầu tiên, **đổi màu hình nền** sang màu trắng, có tác dụng xóa kết quả nhận dạng trước đó, để việc hiển thị kết quả tiếp theo không bị che mất. Câu lệnh thứ 2 sẽ hiển ra màn hình kết quả của việc nhận dạng giọng nói. Đến bước này, bạn có thể chạy chương trình và thử nói 1 vài câu trước máy tính. Kết quả của việc nhận dạng sẽ được hiển thị ra màn hình.

Một mẹo lưu ý nhỏ, là khi mới lần đầu chạy chương trình, bạn nên chờ khoảng 2 giây rồi mới bắt đầu nói. Đây là thời gian cần thiết cho bộ nhận diện giọng nói khởi tạo tạo xong. Khi nói, bạn nên nói thành 1 câu hoàn chỉnh vì bộ nhận dạng có khả năng phân tích ngữ pháp. Do đó, khi nói **Nông trại ơi, bật máy bơm lên** kết quả sẽ chính xác hơn so với việc chỉ nói **bật máy bơm** mà thôi.

### Bước 3: Xây dựng nhóm lệnh bắt từ khóa.

Thực ra, việc nhận dạng giọng nói khó có thể khớp từng lời so với thực tế. Nguyên nhân sai lệnh là do giọng điệu hay thậm chí là tiếng ồn xung quanh. Do đó, chúng ta sẽ bắt 1 số từ khóa chính, chẳng hạn như **bật** và **bơm**, thay vì so sánh nguyên cụm từ **bật máy bơm**. Tương tự như vậy, để điều khiển màu của đèn trên Yolo:Bit, chúng ta có thể bắt 3 từ khóa là **bật** và **đèn** và **tím** (ánh sáng tím rất tốt cho quang hợp của cây). Toàn bộ khối lệnh cho việc này được trình bày bên dưới:



Hình 19.6: *Bắt hai từ khóa để điều khiển máy bơm*

#### **Bước 4:** Gửi từ lệnh sau khi xử lý nhận dạng giọng nói.

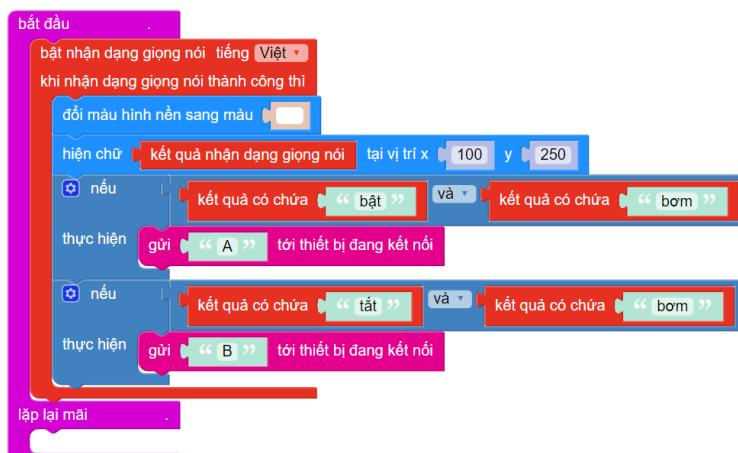
Khi bắt được từ lệnh đi thẳng, chúng ta sẽ gửi một **mã lệnh** tới thiết bị đang kết nối (có thể là xBot hoặc mạch Yolo:Bit), là **kí tự A** cho đơn giản. Câu lệnh để bắt từ khóa (**kết quả có chứa**) nằm trong nhóm **GIỌNG NÓI**. Câu lệnh **gửi 1** tới **thiết bị đang kết nối** nằm trong nhóm **GIAO TIẾP**.



Hình 19.7: *Gửi từ lệnh sau khi xử lý nhận dạng AI*

#### **Bước 5:** Ghép các câu lệnh vào chương trình.

Tiếp tục nhân bản toàn bộ câu lệnh ở bước trên, và ghép nó vào vị trí thích hợp để hoàn thiện chương trình như sau:



Hình 19.8: *Ghép các câu lệnh vào chương trình hoàn chỉnh*

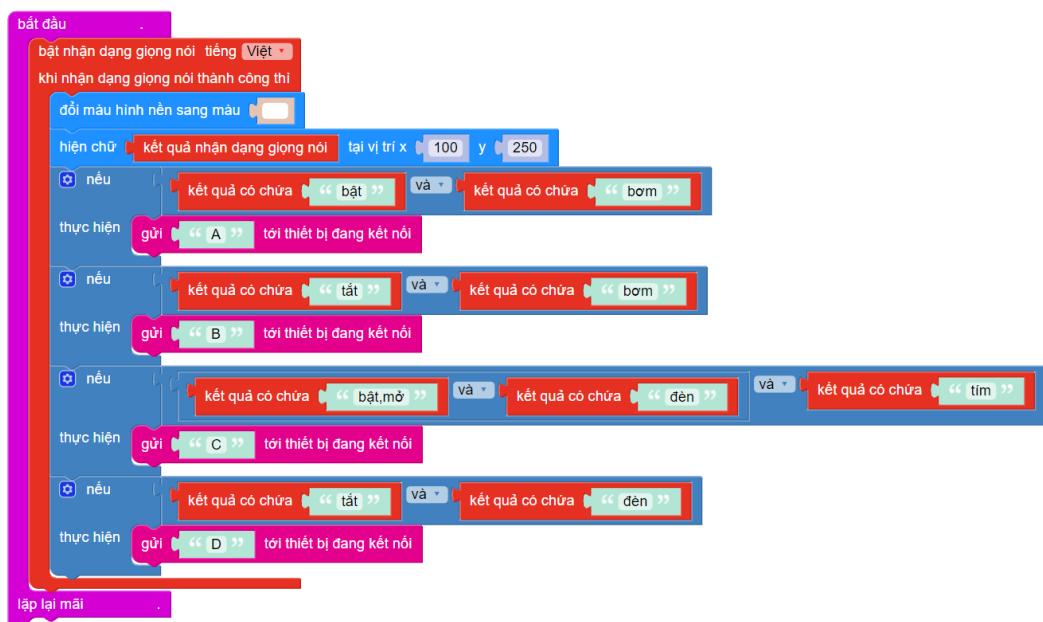
## Bước 6: Bắt nhiều từ khóa cùng lúc.

Trong trường hợp muốn bắt 3 từ khóa cùng 1 lúc, bạn đọc cần phải khéo léo xử lý kết hợp nhiều câu lệnh hơn. Trong gợi ý ở hướng dẫn này, chúng tôi sẽ dùng nhiều câu lệnh **và** cho việc này, ví dụ như sau



Hình 19.9: Bắt nhiều từ khóa trong xử lý giọng nói

Khi các từ khóa cách nhau bằng dấu phẩy trong câu lệnh **kết quả có chứa**, nó mang ý nghĩa là **hoặc**. Tức là trong câu lệnh trên, có 2 trường hợp nó sẽ đúng khi người dùng nói câu **bật đèn màu tím** hoặc **mở đèn màu tím**. Chương trình cho đến lúc này sẽ như sau:



Hình 19.10: Chương trình xử lý giọng nói dựa trên AI

Điểm nổi bật của hệ sinh thái Yolo:Bit là trang lập trình này hoàn toàn tương thích với điện thoại di động hay máy tính bảng. Bạn đọc hoàn toàn có thể hiện thực lại nó bằng điện thoại di động. Bài hướng dẫn tiếp theo sẽ tập trung vào việc nhận lệnh của Yolo:Bit để điều khiển tương ứng.

Với kiến trúc xử lý đặc biệt trên máy tính, chúng ta không có câu lệnh nào trong phần **lặp mãi mãi**. Cuối cùng, việc so sánh chuỗi là có phân biệt chữ hoa chữ thường, bạn đọc cần lưu ý thông tin này để hiện thực cho chính xác.



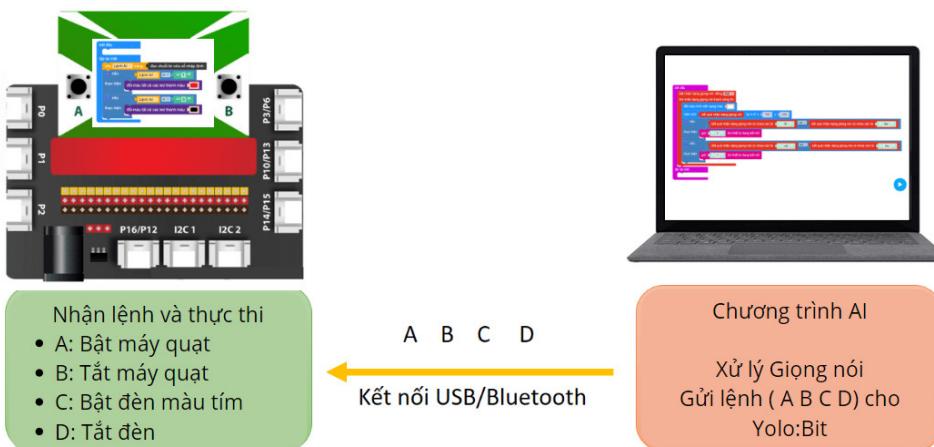
# CHƯƠNG 20

## Nhận và xử lý lệnh AI



# 1 Giới thiệu

Như đã trình bày ở bài hướng dẫn trước, trong bài này chúng ta sẽ hiện thực phần nhận lệnh ở mạch Yolo:Bit. Thực ra, các mạch phần cứng đa phần có tài nguyên khá hạn chế nên không có khả năng chạy các giải thuật trí tuệ nhân tạo. Do đó, trang web lập trình trực tuyến (<https://app.ohstem.vn>) đang thực thi dùm chúng ta các phần liên quan đến trí tuệ nhân tạo. Điều này hoàn toàn khả dĩ bởi máy tính hay thiết bị di động có tài nguyên khá lớn. Cuối cùng, các thiết bị phần cứng chỉ đơn giản là nhận lệnh và thực thi mà thôi.



Hình 20.1: Kiến trúc chung của ứng dụng dựa trên AI

Trong bài hướng dẫn này, chúng ta sẽ tập trung vào việc xây dựng chương trình bên phía mạch Yolo:Bit để nhận lệnh và thực thi. Việc kết nối với các thiết bị ngoại vi, chẳng hạn như máy bơm, bạn đọc có thể chủ động thực hiện dựa theo các bài hướng dẫn trước đó. Với chương trình AI ở bài trước, chúng ta có 4 lệnh cần phải xử lý, được tóm tắt như sau:

- A: Bật máy quạt
- B: Tắt máy quạt
- C: Bật đèn màu tím
- D: Tắt đèn

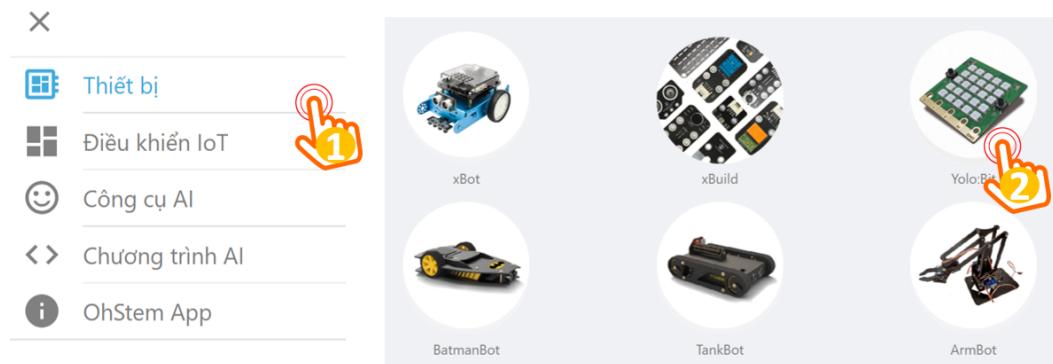
Chúng ta giả định rằng **quạt mini được kết nối với Yolo:Bit ở chân P0**. Trong khi đó, **25 đèn của Yolo:Bit** sẽ được sử dụng để phát sáng ra màu tương ứng. Các mục tiêu chính trong bài hướng dẫn này như sau:

- Xây dựng chương trình nhận lệnh trên Yolo:Bit
- Điều khiển ngoại vi theo từ lệnh
- Kết nối Yolo:Bit và hệ thống nhận dạng giọng nói

## 2 Xây dựng chương trình cho Yolo:Bit

### Bước 1: Mở trang lập trình cho Yolo:Bit.

Từ trang chủ của môi trường lập trình (<https://app.ohstem.vn>), chúng ta chọn thiết bị là Yolo:Bit như đã hướng dẫn ở các bài trước.



Hình 20.2: *Bắt đầu với chương trình trên Yolo:Bit*

### Bước 2: Khai báo biến Lệnh AI

Để tiện lợi cho các bước xử lý tiếp theo, bạn cần khai báo 1 biến số, đặt tên là **Lệnh AI**, để lưu lại lệnh được gửi tới xBot. Trình tự khai báo biến này được trình bày như hướng dẫn bên dưới:



Hình 20.3: *Khai báo biến Lệnh AI*

Từ nhóm lệnh **BIẾN**, chúng ta nhấp vào nút **Tạo biến...**, đặt tên cho nó và nhấp vào nút **Lưu**. Một biến mới sẽ được tạo ra, với các khối lệnh mới được tự động sinh ra trong phần này.

### Bước 3: Đọc lệnh AI từ cửa sổ lệnh.

Ở bước này, chúng ta cần **định kì kiểm tra** lệnh được gửi tới từ hệ thống AI. Do đó, phần lệnh này sẽ phải được hiện thực trong khối **lặp mãi mãi**. Chương trình gợi ý cho bạn đọc như sau:

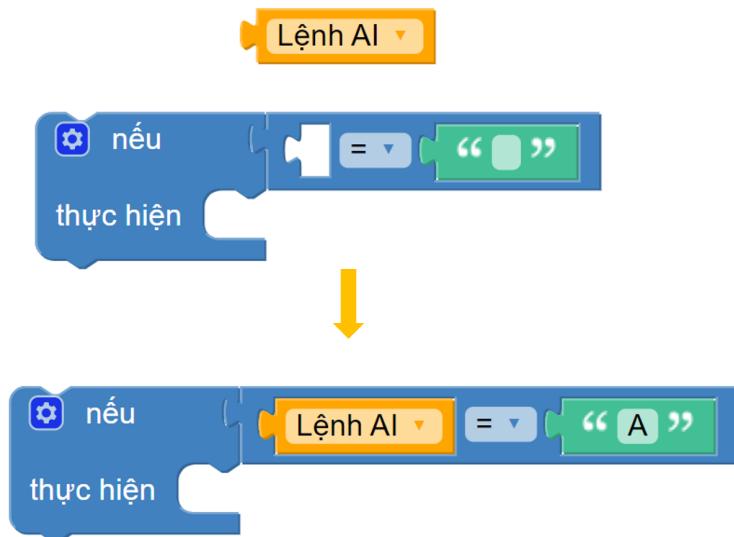


Hình 20.4: Đọc lệnh từ cửa sổ lệnh

Khối lệnh **đọc chuỗi từ cửa sổ lệnh** được lấy trong phần **NÂNG CAO**, và chọn tiếp nhóm lệnh **THIẾT BỊ**. Tại đây bạn sẽ tìm thấy được khối lệnh này.

#### Bước 4: Xây dựng phép so sánh chuỗi.

Biến **Lệnh AI** được đọc từ cửa sổ lệnh và có kiểu là chuỗi (còn gọi là xâu). Do đó, chúng ta cần phải xây dựng câu lệnh **nếu ... thực hiện** với phép so sánh chuỗi. Từng bước thực hiện cho phần này được minh họa như hình bên dưới:

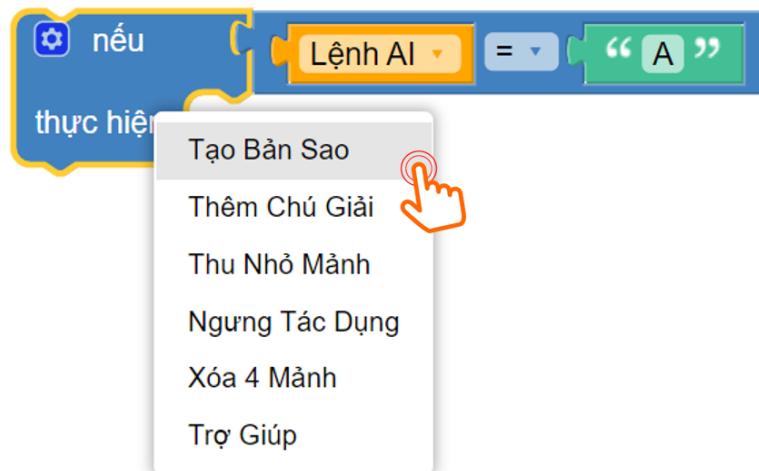


Hình 20.5: Xây dựng phép so sánh chuỗi

Từ câu lệnh được hỗ trợ trong phần **LOGIC** và khối **Lệnh AI** được lấy trong mục **BIẾN**, chúng ta sẽ xây dựng được cấu trúc điều kiện như trên.

#### Bước 5: Nhân bản và ghép nối chương trình.

Chúng ta sẽ nhân bản khối lệnh vừa thực hiện ở bước 4, bằng cách nhấn chuột phải vào khối lệnh **nếu ... thực hiện** (khối lệnh cha) và chọn vào **Tạo Bản Sao**, như sau:



Hình 20.6: Nhân bản câu lệnh so sánh chuỗi

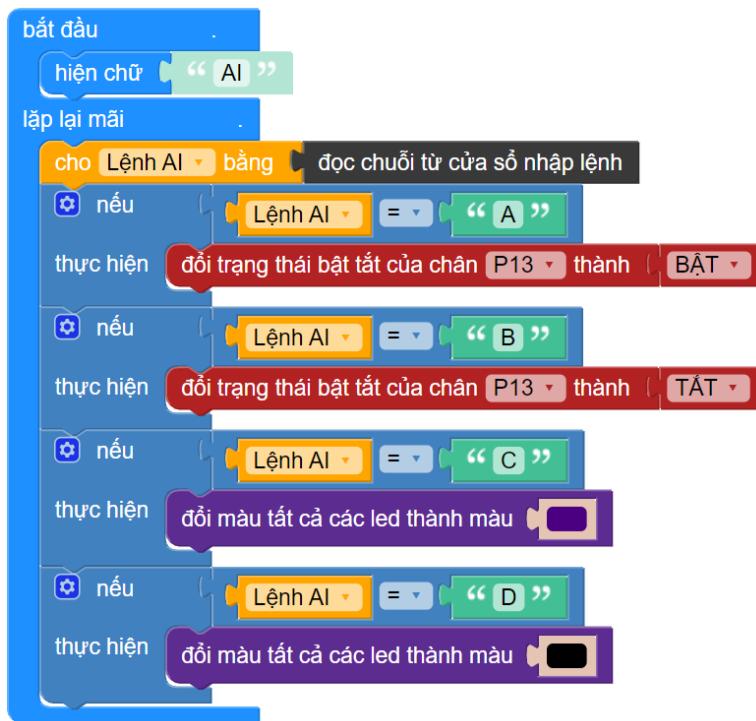
Cuối cùng, chúng ta ghép nối lại chương trình để có được cấu trúc chương trình như sau:



Hình 20.7: Cấu trúc chương trình nhận 4 lệnh

**Bước 6:** Thực hiện các di chuyển tương ứng.

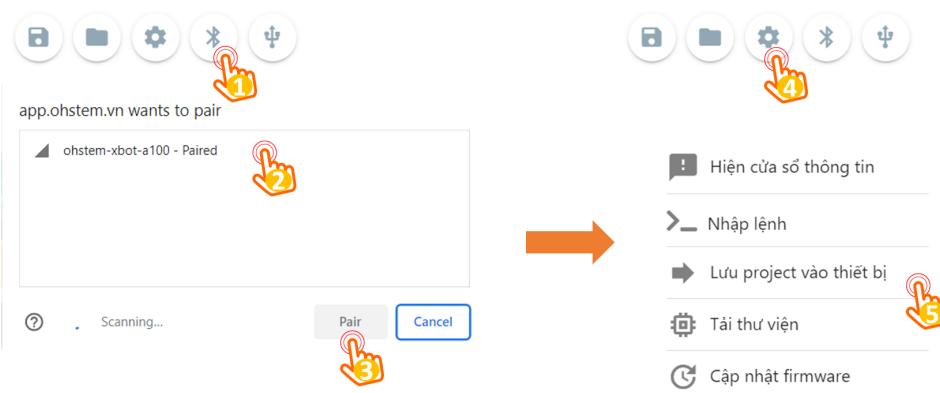
Cho đến bước này, chúng ta sẽ ghép thêm các câu lệnh điều khiển ngoại vi cho trùng khớp với các từ lệnh đã định nghĩa từ đầu. Các câu lệnh cần thiết có thể tìm thấy trong phần **NÂNG CAO**, **CHÂN CẤM** và **LED** cho chương trình sau đây.



Hình 20.8: Hoàn thiện chương trình

### 3 Lưu chương trình vào thiết bị

Sau khi biên soạn chương trình xong, chương trình cần được lưu cố định vào thiết bị, trong trường hợp này là mạch Yolo:Bit. Điều này sẽ rất cần thiết cho việc kết nối ổn định giữa Yolo:Bit và chương trình AI về sau.



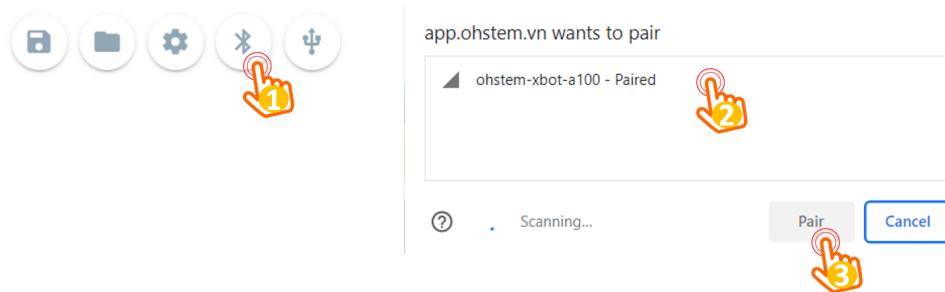
Hình 20.9: Kết nối và lưu chương trình vào Yolo:Bit

Việc lưu một chương trình vào trong xBot cũng được thực hiện tương tự như với mạch Yolo:Bit, đó là kết nối với thiết bị (bằng bluetooth hay USB đều được), sau đó từ biểu tượng cài đặt, chọn tiếp **Lưu project vào thiết bị**, như minh họa ở Hình

Cuối cùng, chúng ta cần **RESET** lại Yolo:Bit bằng cách nhấn nút reset hoặc tắt nguồn mở lại. Khi dòng chữ **AI** xuất hiện trên màn hình của Yolo:Bit, việc reset đã thành công. Khi reset Yolo:Bit, việc kết nối nó với môi trường lập trình cũng bị ngắt. Đây là điều cần thiết để nó có thể kết nối với phần trí tuệ nhân tạo ở phần tiếp theo.

## 4 Kết nối với trí tuệ nhân tạo

Từ trang lập trình dành cho trí tuệ nhân tạo, có 2 cách khả dĩ để kết nối với Yolo:Bit, bao gồm kết nối thông qua Bluetooth hoặc USB. Tùy vào điều kiện thiết lập ứng dụng mà bạn đọc có thể chọn một trong hai phương pháp kết nối với Yolo:Bit. Kết nối bằng dây thông qua USB sẽ ổn định hơn Bluetooth. Trong khi đó, kết nối không dây qua Bluetooth sẽ có khoảng cách xa hơn và thuận tiện hơn. Hình ảnh bên dưới minh họa cho việc kết nối thông qua Bluetooth.



Hình 20.10: Kết nối Bluetooth giữa AI và xBot

Cuối cùng, chúng ta nhấn vào nút chạy chương trình nhận diện giọng nói mà chúng ta đã làm ở bài trước. Bây giờ bạn đã có thể thử nói và xem Yolo:Bit có vận hành theo đúng chương trình mà mình đã thiết kế cho nó hay không.

Trong trường hợp có lỗi, bạn chọn cách xử lý như hướng dẫn sau:

- Trường hợp kết nối bằng Bluetooth: Reset lại mạch trước rồi kết nối lại bằng Bluetooth.
- Trường hợp kết nối bằng USB: Kết nối lại rồi mới Reset mạch Yolo:Bit

Trong cả 2 trường hợp, sau khi Reset mạch Yolo:Bit, bạn phải chờ dòng chữ **AI xuất hiện trên 25 đèn** của Yolo:Bit rồi mới thao tác tiếp. Cuối cùng, chạy lại chương trình AI để nhận diện giọng nói và gửi lệnh cho Yolo:Bit.



# CHƯƠNG 21

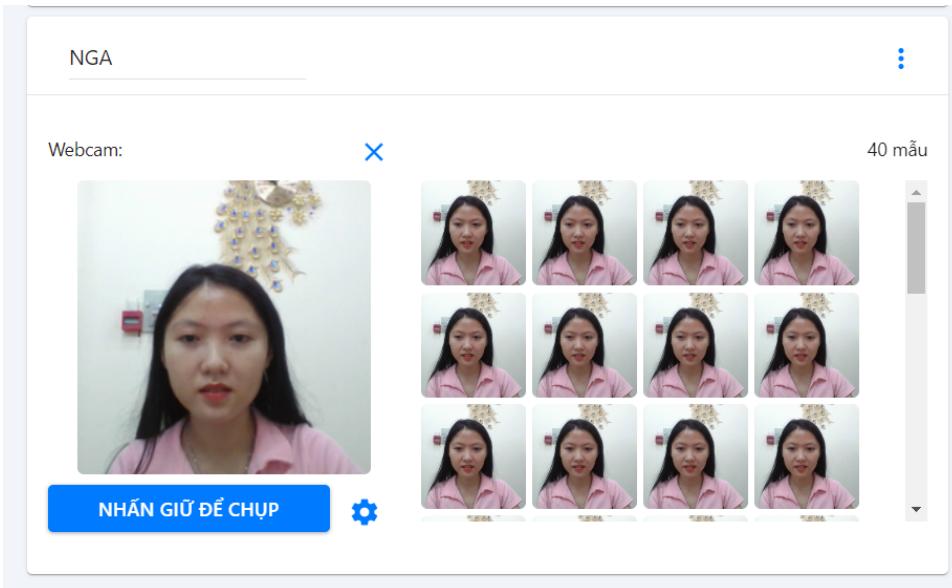
## Huấn luyện trí tuệ nhân tạo



# 1 Giới thiệu

Trong thời đại bùng nổ của cuộc cách mạng công nghệ 4.0, chắc hẳn chúng ta đã nghe nói nhiều về khái niệm trí tuệ nhân tạo, hay còn gọi là AI (Artificial Intelligence). Nhằm cung cấp công nghệ để người dùng có thể tự xây dựng một hệ thống AI, từ năm 2017, Google đã xây dựng một công cụ trực tuyến, có tên gọi là Teachable Machine. Chỉ cần vài thao tác đơn giản, bạn hoàn toàn có thể **tự xây dựng cho mình một bộ não nhân tạo**, với những kiến thức do bạn chủ động.

Hiện nay, với phiên bản 2.0, Teachable Machine đã trở thành một công cụ đắc lực trong việc tự thiết kế các ứng dụng về trí tuệ nhân tạo. Công cụ này đã hỗ trợ nhiều nền tảng lập trình khác nhau như Scratch (ngôn ngữ cho học sinh) và Python - ngôn ngữ lập trình mạnh mẽ, phù hợp cho các ứng dụng thời đại mới. Ngoài ra, còn có rất nhiều nền tảng AI khác đang được dựa trên lõi trung tâm là Google Teachable Machine. Điểm khác biệt của nó so với phần nhận diện giọng nói (sử dụng trí tuệ nhân tạo có sẵn của Google), là chúng ta tự xây dựng bộ não trí tuệ nhân tạo cho mình bằng công cụ do Google cung cấp.



Hình 21.1: Huấn luyện AI để phân biệt một số khuôn mặt

Trong bài hướng dẫn này, chúng tôi sẽ giới thiệu các thao tác cơ bản của Teachable Machine, được tích hợp trong phần mềm lập trình trực tuyến OhStem. Chúng ta sẽ huấn luyện hệ thống để nó có thể **phân biệt được ai là người đang đứng trước camera**, mô phỏng cho chức năng nhận dạng FaceID. Các nội dung chính trong bài đầu tiên bao gồm:

- Sử dụng được công cụ Teachable Machine trên trang lập trình trực tuyến OhStem
- Thu thập hình ảnh khuôn mặt của nhiều người khác nhau
- Huấn luyện hệ thống phân biệt được ai đang đứng trước camera

Trong phần còn lại của hướng dẫn, chúng tôi sẽ sử dụng thuật ngữ việt hóa cho Teachable Machine là "Học Máy Google" để thuận tiện hơn cho bạn đọc. Chúng tôi cũng định nghĩa cho tính năng phát triển trong bài này là **FaceAI**.

## 2 Trang chủ của trí tuệ nhân tạo

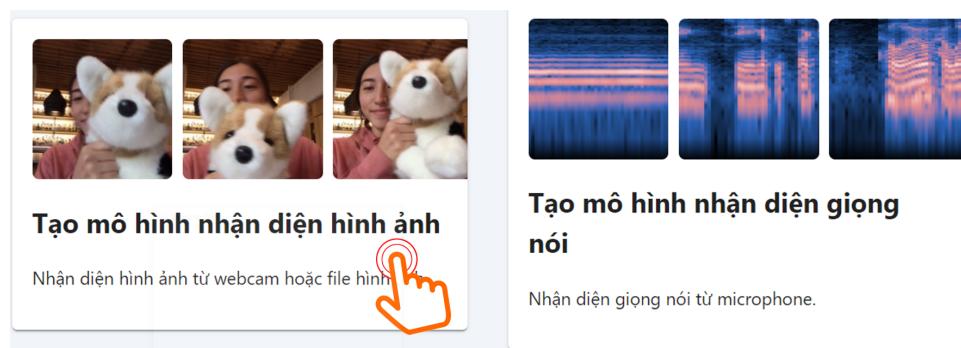
Để bắt đầu huấn luyện một hệ thống trí tuệ nhân tạo trong hệ sinh thái Yolo:Bit, bạn cần truy cập vào địa chỉ trực tuyến <https://app.ohstem.vn/>, trở về màn hình chủ của nó (bằng cách nhấn nút trở về trong giao diện của trang web), như minh họa ở hình bên dưới:



Hình 21.2: Giao diện trang chủ và tính năng huấn luyện mô hình AI

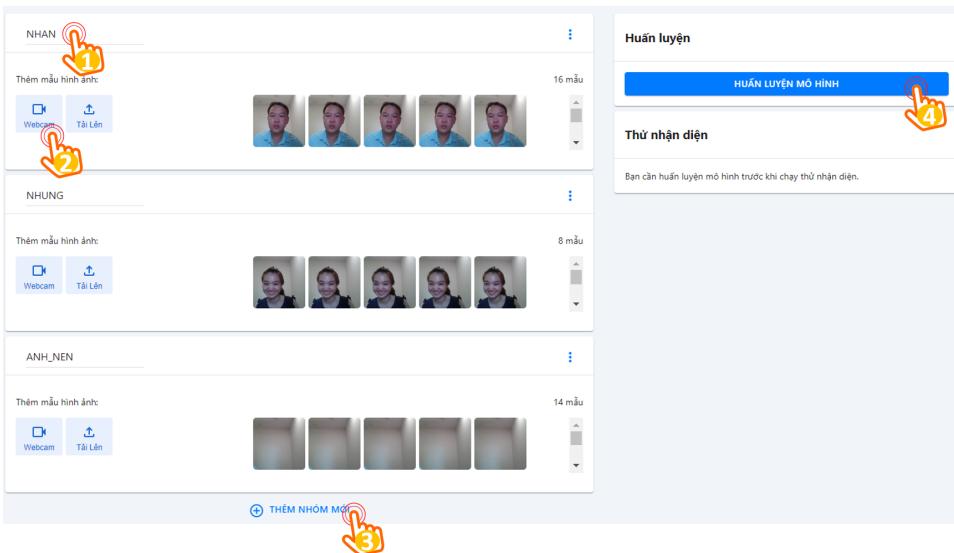
## 3 Huấn luyện hệ thống

**Bước 1:** Từ trang chủ trình bày ở Hình 21.2, bạn chọn vào **Mô hình AI**, chọn tiếp **Tạo mô hình nhận diện hình ảnh**, như minh họa ở hình bên dưới:



Hình 21.3: Tạo dự án nhận diện hình ảnh bằng AI

**Bước 2:** Với giao diện mới được hiện ra, bạn đặt tên cho kiến thức mà mình muốn huấn luyện, chẳng hạn như là **LaXanh** là hình ảnh các lá cây tươi tốt và **LaVang** cho hình ảnh các cây không tốt, như minh họa bên dưới.

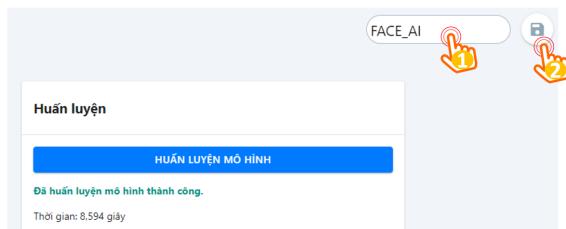


Hình 21.4: Huấn luyện hệ thống FaceAI cho việc nhận dạng

Trong phần này, bạn cần chụp hình những người muốn nhận dạng bằng Webcam của hệ thống (nhấn vào chức năng **WEBCAM**) hoặc tải ảnh có sẵn từ máy tính lên (nhấn vào chức năng **Tải Lên**). Điều quan trọng đối với việc thu thập dữ liệu trước khi huấn luyện, là các hình ảnh bên lề và không liên quan đến việc nhận dạng, tiêu biểu nhất là **ảnh nền**, với ý nghĩa là **không có ai** trước Webcam. Kiến thức **ANH\_NEN** được tạo thêm bằng cách nhấn vào **THÊM NHÓM MỚI**, như minh họa ở hình bên trên.

Cuối cùng, nhấn vào nút **HUẤN LUYỆN MÔ HÌNH** và chờ cho đến khi việc huấn luyện kết thúc.

### Bước 3: Đặt tên và lưu lại mô hình AI.



Hình 21.5: Đặt tên và lưu lại dự án FaceAI

Cung cấp tên cho mô hình AI mà chúng ta mới huấn luyện, sau đó nhấn vào biểu tượng lưu lại. Giao diện sau đây sẽ xuất hiện khi việc lưu thành công.

Sau khi huấn luyện hệ thống xong, bạn đọc có thể nhấn vào biểu tượng nút **Back** để trở về màn hình chính của hệ sinh thái Yolo:Bit.

Lưu mô hình "FACE\_AI" thành công!

OK

Hình 21.6: Lưu dự án FaceAI thành công

## 4 Hiện thực dự án AI

**Bước 1:** Từ màn hình chính của trang lập trình, lần này chúng ta sẽ chọn vào **Lập trình AI**, như minh họa ở hình bên dưới:



Hình 21.7: Giao diện lập trình Trí tuệ nhân tạo

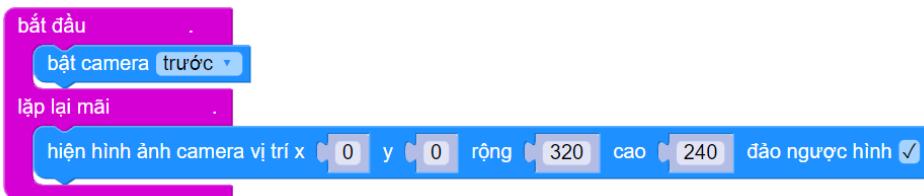
Các câu lệnh lập trình sẽ khá tương tự với phần nhận diện giọng nói, nhưng lần chúng ta sẽ xài các câu lệnh trong nhóm **M-LEARNING**.



Hình 21.8: Nhóm lệnh M-LEARNING cho mô hình AI tự xây dựng

**Bước 2:** Thực hiện các tính năng cơ bản để khởi động Camera và hiện hình ảnh từ Camera.

Hiện tại, với phiên bản chạy trên máy tính, bạn đọc cần khởi động **Camera trước**. Với câu lệnh trong phần **lặp mãi mãi**, bạn hãy đảm bảo rằng mình đã chọn vào

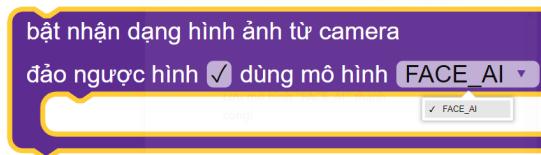


Hình 21.9: Khởi động camera nhận dạng

**đảo ngược hình** (do chúng ta đang sử dụng Camera trước của laptop).

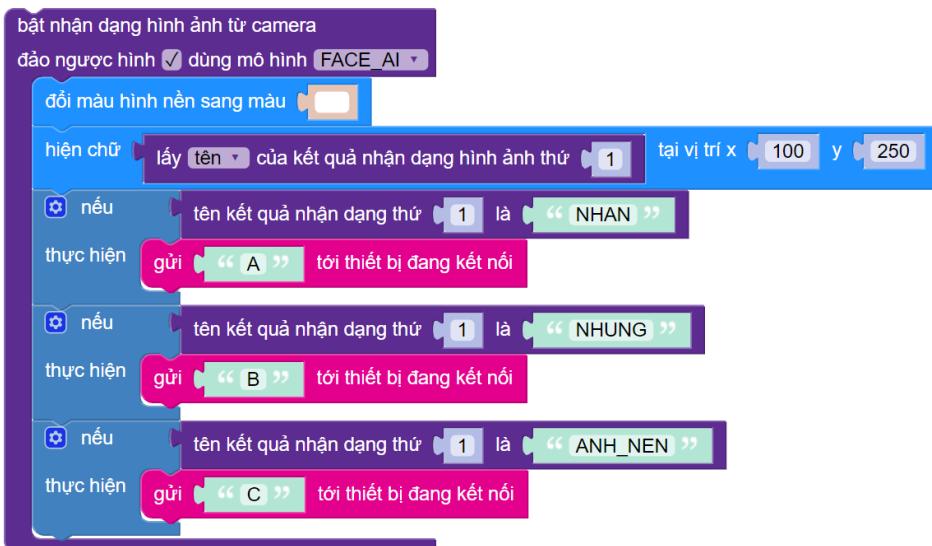
### Bước 3: Hiện thực khôi phục năng lực AI.

Câu lệnh chính để hiện thực khôi phục năng lực nhận diện bằng AI được trình bày bên dưới, nằm trong nhóm **M-LEARNING**. Đây là câu lệnh có thể chọn lựa và chúng ta cần chọn đúng mô hình AI đã huấn luyện ở phần trước.



Hình 21.10: Khởi lệnh nhận dạng bằng trí tuệ nhân tạo

Mặc định, câu lệnh này cũng đảo ngược hình cho tương đồng với cấu hình Camera trước. Chúng ta phân loại kết quả nhận dạng từ AI bằng các câu lệnh nếu, và gửi kết quả này xuống mạch Yolo:Bit cho những bước xử lý tiếp theo, như gợi ý ở phần bên dưới:



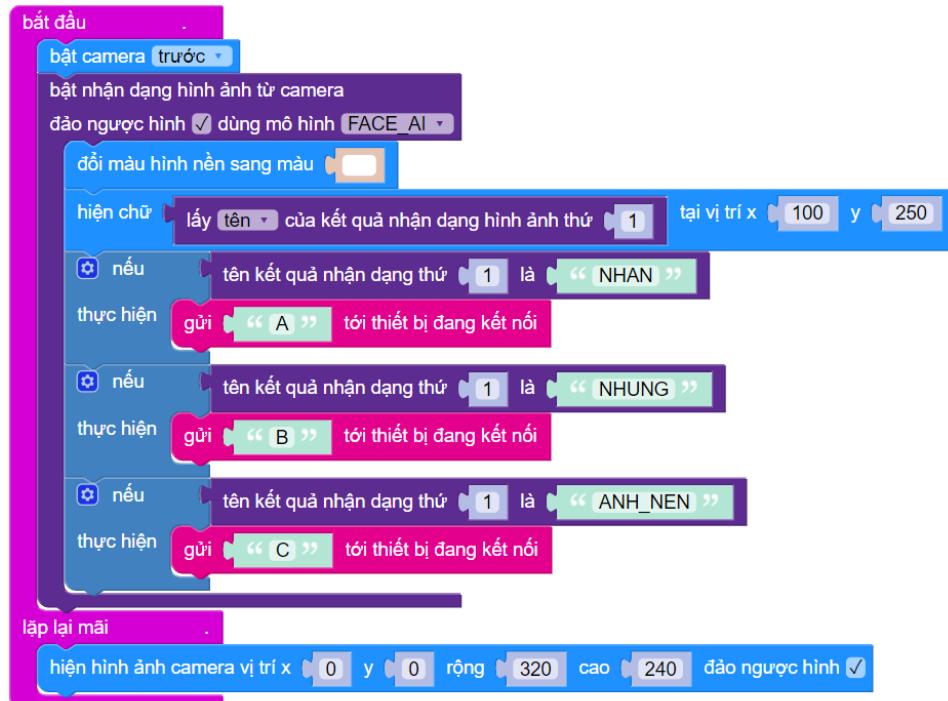
Hình 21.11: Kiểm tra kết quả nhận dạng và gửi lệnh tới Yolo:Bit

Lệnh **gửi tới thiết bị đang kết nối** nằm trong nhóm lệnh **GIAO TIẾP**. Trong câu

lệnh kiểm tra điều kiện, bạn phải dùng đúng tên biến đã đặt khi huấn luyện hệ thống (có phân biệt viết hoa thường). Để tiện cho việc xử lý trên mạch Yolo:Bit, chúng ta sẽ chỉ **gửi một chữ cái** mà thôi.

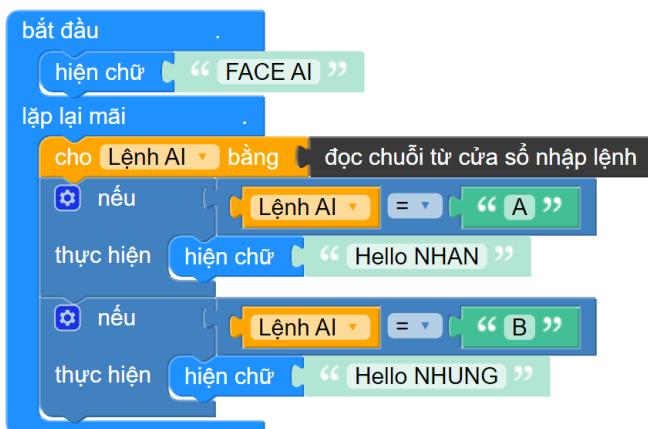
#### Bước 4: Hoàn thiện chương trình.

Với khôi lệnh đã chuẩn bị ở bước trên, chúng ta sẽ đặt nó vào trong phần **bắt đầu**. Chương trình hoàn chỉnh của chúng ta sẽ như sau:



Hình 21.12: Hoàn thiện chương trình FaceAI

Sau khi kiểm tra dự án hoàn chỉnh, câu lệnh trong lặp mãi mãi có thể bỏ qua, để giảm tải cho chương trình nếu cần. Phần chương trình trên mạch Yolo:Bit hoàn toàn tương tự với bài trước, nên sẽ không trình bày chi tiết ở hướng dẫn này. Chương trình gợi ý cho bạn đọc được trình bày như sau:



Hình 21.13: Chương trình nhận lệnh trên mạch Yolo:Bit



# CHƯƠNG 22

## Kết hợp công nghệ AI



## 1 Giới thiệu

Trong các bài hướng dẫn trước, chúng ta đã được trải nghiệm với 2 ứng dụng sử dụng công nghệ trí tuệ nhân tạo. Cụ thể, hệ thống nhận dạng tiếng Việt và nhận dạng các khuôn mặt khác nhau đã được trình bày.



Hình 22.1: Công nghệ nhận dạng giọng nói và Học máy với Google

Trong bài hướng dẫn này, chúng tôi sẽ trình bày sâu hơn về công nghệ bên dưới của 2 ứng dụng trên. Với ứng dụng nhận dạng giọng nói, chúng ta đang xài công nghệ chuyển đổi giọng nói thành văn bản. Đây là công nghệ rất phổ biến trong các ứng dụng tìm kiếm bằng giọng nói hoặc các hệ thống trợ lý ảo nhân tạo. Trong khi đó, để nhận dạng khuôn mặt, công nghệ học máy với Google đã được sử dụng. Với công nghệ này, chúng ta có thể tự tạo cho mình một bộ não nhân tạo cho từng ứng dụng riêng biệt của chúng ta. Tuy nhiên, cũng có một số hạn chế liên quan đến việc này, chủ yếu liên quan đến việc dữ liệu không đầy đủ.

Sau cùng, chúng tôi sẽ đề xuất một kiến trúc để bạn đọc có thể tích hợp cùng lúc 2 công nghệ này trong một ứng dụng. Trong một số trường hợp, có thể bạn đọc sẽ cần tính năng điều khiển bằng giọng nói kết hợp với bộ não nhân tạo tự huấn luyện.

## 2 Chuyển đổi giọng nói - văn bản

Chuyển đổi giọng nói thành văn bản (Speech to Text - Speech2Text) là một công nghệ giúp máy tính nhận dạng âm thanh của tiếng nói người và tạo ra chuỗi văn bản tương ứng. Khởi đầu, tiếng nói sẽ được ghi nhận qua microphone và lưu trữ trong máy tính dưới dạng các tín hiệu số. Để máy tính có thể nhận dạng dữ liệu tiếng nói, rất nhiều kỹ thuật xử lý tín hiệu số và xử lý ngôn ngữ tự nhiên được sử dụng. Gần đây, với sự tiến bộ của kỹ thuật học sâu (một nhánh đang phát triển rất mạnh của trí tuệ nhân tạo - AI), việc nhận dạng giọng nói riêng có thể đạt được độ chính xác rất cao trên rất nhiều ngôn ngữ.

Nhận dạng giọng nói là công nghệ quan trọng nhất để tạo nên các ứng dụng tương tác thông minh qua giọng nói, như tìm kiếm qua giọng nói trên điện thoại di động, tivi thông minh. Đặc biệt, công nghệ này ứng dụng rất nhiều trong các dịch vụ trợ lý ảo như Siri trên iOS, Cortana trên Windows, hoặc các trợ lý nhà thông minh như Alexa của Amazon và Google Home của Google. Công nghệ nhận dạng giọng nói mở ra kỷ nguyên mới cho việc tương tác người máy. Để yêu cầu thiết bị thông minh

thực hiện chỉ thị của mình, giờ đây bạn chỉ cần ra lệnh bằng giọng nói, thay vì phải thao tác qua nút ấn hoặc màn hình. Vậy rõ ràng cuộc sống cá nhân của con người được giúp ích rất nhiều với công nghệ nhận dạng giọng nói. Công nghệ nhận dạng giọng nói có thể áp dụng trong rất nhiều lĩnh vực của đời sống, từ doanh nghiệp, nhà thông minh, giao tiếp với Robot cho tới giáo dục.

## 2.1 Môi trường doanh nghiệp

Nhận diện giọng nói có thể đơn giản hóa các công việc thường ngày ở công ty bạn, giảm sai sót, tiết kiệm nhân công và giúp tăng hiệu quả công việc tại công ty. Một số lợi ích của nó có thể kể đến như:

- Giảm thời gian tương tác: Trong thực tế, thời gian để nói ra một câu luôn ngắn hơn thời gian để nhập câu đó vào máy tính. Cũng như vậy, thời gian để đọc được một tin nhắn luôn ngắn hơn thời gian để nghe tin nhắn thoại cùng một nội dung.
- Giảm thời gian nhập liệu: Nhập liệu là một công việc nhàn chán và dễ sai sót. Với tính năng chuyển đổi giọng nói thành văn bản, nhân viên nhập liệu có thể nhập dữ liệu trực tiếp bằng giọng nói của mình thay vì đọc dữ liệu từ tài liệu nguồn và gõ lại trên bàn phím.
- Tự động tạo biên bản cuộc họp: Mỗi ngày có rất nhiều cuộc họp quan trọng cần được ghi biên bản. Speech to Text có thể dễ dàng tự động chuyển đổi nội dung ghi âm của cuộc họp thành văn bản, nhờ đó giảm được áp lực và sai sót của thư ký ghi biên bản.
- Hiểu thông tin khách hàng để marketing hướng mục tiêu: Các công nghệ nhận dạng giọng nói hiện đại có thể dễ dàng đoán được độ tuổi, giới tính và vùng miền của con người thông qua giọng nói. Điều này rất có lợi cho công ty bạn khi cần biết một số thông tin cá nhân của khách hàng để thực hiện chiến lược marketing đúng mục tiêu.

## 2.2 Nhà thông minh và Robot

Giao tiếp bằng giọng nói được ứng dụng ngày càng nhiều trong 2 lĩnh vực này. Công nghệ nhận dạng giọng nói giúp khoảng cách giao tiếp giữa người và máy được rút ngắn. Việc sử dụng, ra lệnh, truy vấn thông tin ngày nay dễ dàng hơn rất nhiều cho người già, trẻ em hoặc người khuyết tật.

Các thiết bị IoT nhà thông minh luôn cần một thiết bị trợ lý ảo để điều khiển chúng. Công nghệ nhận dạng giọng nói hiện nay giúp chủ nhân có thể điều khiển ngôi nhà thông minh của mình mà không cần dùng một nút ấn nào. Công nghệ nhận dạng giọng nói thực sự giúp cho các ứng dụng trở nên thông minh và hoàn thiện hơn. Hiện tại, công nghệ này cũng đã ứng dụng trong ô tô.

Bên cạnh đó, giao tiếp bằng giọng nói với robots hiện nay không còn là ý tưởng nữa, mà đã được áp dụng thực tế. Rất nhiều robots gần đây có khả năng giao tiếp như người thật. Thêm chí vào năm 2017, Sophia, một robot có khả năng giao tiếp và biểu hiện sắc thái được phát triển từ Hong Kong vào năm 2016, còn được Arab

Saudi cấp quyền công dân. Tất cả khả năng giao tiếp của robot đều là thành tựu của trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên.

### 2.3 Trong giáo dục

Rất nhiều ứng dụng dạy ngôn ngữ hiện nay sử dụng trí tuệ nhân tạo và nhận dạng tiếng nói như công nghệ chính đánh giá khả năng ngôn ngữ của người học. Các bài kiểm tra sẽ được lập trình để sinh ra tự động câu hỏi. Khi người dùng trả lời, bộ não nhân tạo sẽ phân tích giọng nói thành văn bản, để có thể so sánh kết quả với đáp án và đưa ra con số đánh giá. Những kỹ thuật này có thể thay thế giáo viên trong việc đánh giá và hoàn thiện kỹ năng phát âm của học viên, một tiêu chí quan trọng khi học một ngoại ngữ.

Bên cạnh đó, các trợ lý ảo cũng được phát triển trong giáo dục, để có thể trò chuyện và tư vấn cho học sinh, chẳng hạn như vấn đề về chọn ngành học hay chọn trường thi. Với khả năng nhớ nhiều tri thức hơn, cùng bộ suy luận không thể nhầm lẫn, công nghệ nhận diện giọng nói kết hợp với trí tuệ nhân tạo sẽ thay đổi cách chúng ta tiếp xúc với máy tính và các thiết bị thông minh trong tương lai.

## 3 Học máy với Google

Machine Learning đang là một trong những từ khóa hot nhất của lĩnh vực công nghệ hiện nay. Đã có rất nhiều tài liệu cũng như những bài viết chuyên sâu, cụ thể về vấn đề này. Tuy nhiên, được tự mình trải nghiệm bao giờ cũng mang lại cảm giác rất khác biệt so với những gì mà bạn đọc được hoặc xem được. Nắm bắt được nhu cầu đó đó, Google đã phát triển một tính năng nhỏ có tên Teachable Machine tích hợp trên các trình duyệt web để giúp bạn thực hiện điều này. Teachable Machine sẽ cho bạn thấy những gì mà công nghệ AI hiện đại có thể và không thể làm được một cách chân thực, chính xác nhất.

Cụ thể, Teachable Machine sử dụng webcam của bạn để làm công cụ “dạy học” cho một chương trình AI cơ bản. Chỉ cần click vào các phần huấn luyện mô hình, chương trình này sẽ ghi lại và “học” tất cả những gì bạn cung cấp cho nó, bao gồm hình ảnh từ webcam hoặc hình ảnh được tải lên từ máy tính. Sau quá trình trên, chương trình AI đó sẽ có khả năng xác định, nhận diện những gì đã được huấn luyện, để bạn có thể tích nó vào trong hệ thống đang có, như là một cảm biến cao cấp sử dụng công nghệ AI.

Điểm thú vị lớn nhất với công nghệ học máy này, là bạn có thể chủ động dạy cho nó với các hình ảnh từ dự án cụ thể của mình. Tuy nhiên, điểm bất lợi là dữ liệu do bạn đọc cung cấp thường có số lượng ít, chưa đủ phong phú và đa dạng để xử lý nhiều trường hợp phát sinh trong thực tế. Chẳng hạn như, để quan sát 1 cây trồng có sinh trưởng bình thường hay không, chúng ta thường quan sát vào ban ngày và bỏ qua thời gian vào ban đêm. Việc thiếu dữ liệu vào ban đêm cũng ảnh hưởng một phần nào đó vào hiệu suất của hệ thống. Thông thường, để xử lý cho phần này, chúng ta sẽ kết hợp thêm thông tin về cảm biến trước khi kích hoạt việc nhận dạng bằng công nghệ học máy này.

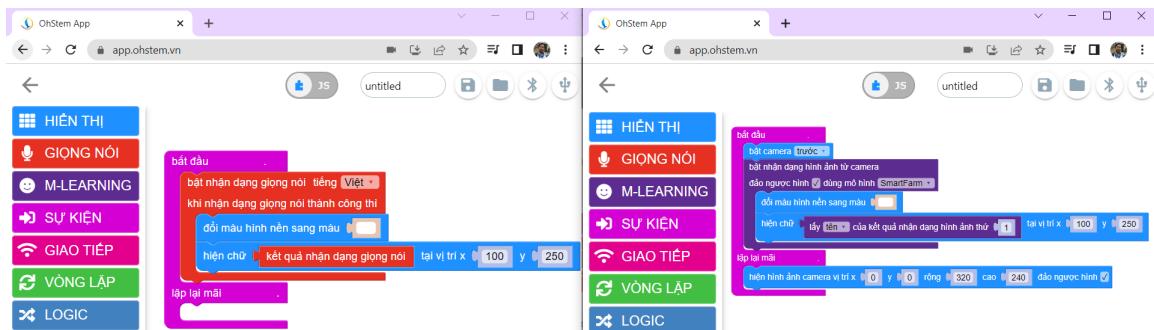
## 4 Kết hợp hai công nghệ AI

Hai công nghệ này có thể tích hợp chung trong một ứng dụng, bởi nó không bị mâu thuẫn về mặt phần cứng của hệ thống xử lý AI, cụ thể:

- Nhận diện giọng nói: Truy cập vào micro của thiết bị.
- Học máy với Google: Truy cập vào webcam của thiết bị.

Do đó, bạn có thể kết hợp nhận diện giọng nói và học máy với hình ảnh. Trường hợp nhận diện giọng nói và học máy với âm thanh sẽ không kết hợp được, bởi chúng bị xung đột về tài nguyên phần cứng.

Thiết bị xử lý AI trong trường hợp này **phải là máy tính**. Bởi máy tính là thiết bị có thể hỗ trợ chạy 2 chương trình AI cùng 1 lúc trên 2 cửa sổ của trình duyệt web. Mặc dù các thiết bị di động như điện thoại thông minh hay máy tính bảng có hỗ trợ xử lý đa luồng, nhưng phần mềm trên các thiết bị này không thể mở 2 trình duyệt web cùng 1 lúc. Hình ảnh bên dưới minh họa việc hiện thực 2 bộ xử lý AI trên 2 cửa sổ của trình duyệt web.



Hình 22.2: Hiện thực nhận dạng giọng nói và học máy với Google

Tuy nhiên, khi hai chương trình AI cùng chạy một lúc, chúng ta sẽ không thể kết nối đến Yolo:Bit bằng Bluetooth hoặc dây cáp USB. Hai công nghệ kết nối này chỉ hỗ trợ kết nối một chương trình với một thiết bị mà thôi. Do đó, thay vì gửi thông tin tới thiết bị đang kết nối, chúng ta sẽ sử dụng WIFI để gửi thông tin lên server OhStem. Mạch Yolo:Bit muốn nhận thông tin này sẽ phải sử dụng công nghệ kết nối vạn vật, dựa trên giao thức MQTT và đăng ký với server OhStem. Chương trình cho cả phần giọng nói và nhận dạng AI cần phải thay đổi để phù hợp với kiến trúc này, và được trình bày ở phần bên dưới.

### 4.4 Nhận dạng giọng nói

Trong chương trình nhận dạng giọng nói, trước khi bật khôi nhận dạng, chúng ta cần phải kết nối với server OhStem. Do máy tính thông thường đã được kết nối sẵn vào mạng, chúng ta không cần phải thực thi việc kết nối vào WiFi như mạch Yolo:Bit mà chỉ cần kết nối với OhStem server là đủ. Chương trình của chúng ta sẽ bắt đầu như sau.



Hình 22.3: Hiện thực nhận dạng giọng nói liên kết với OhStem server

Một câu lệnh quan trọng dùng để kết nối với server OhStem được ghép đầu tiên trong khối **bắt đầu**. Do không phải tạo tài khoản, bạn đọc chỉ cần chọn cho mình một **username đặc biệt** để không bị trùng với người khác.

Tiếp theo đó, việc hiện thực xử lý cho một ứng dụng điều khiển bằng giọng nói được hiện thực bình thường với câu lệnh nếu và bắt các từ khóa quan trọng. Tuy nhiên, trong phần **thực hiện**, chúng ta sẽ gửi thông tin tới một chủ đề trên server, như minh họa sau đây.



Hình 22.4: Xử lý cho giọng nói - Gửi kết quả lên server

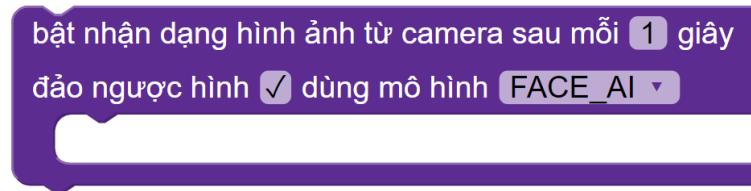
Hiện tại, OhStem hỗ trợ 20 kênh dữ liệu khác nhau cho bạn lựa chọn để gửi thông tin. Ở đây, chúng tôi sử dụng kênh V10 để gửi kết quả của việc xử lý giọng nói. Hình ảnh của chương trình sau khi hoàn thiện sẽ như sau:



Hình 22.5: Hoàn thiện chương trình giọng nói khi liên kết với OhStem server

## 4.5 Nhận dạng hình ảnh

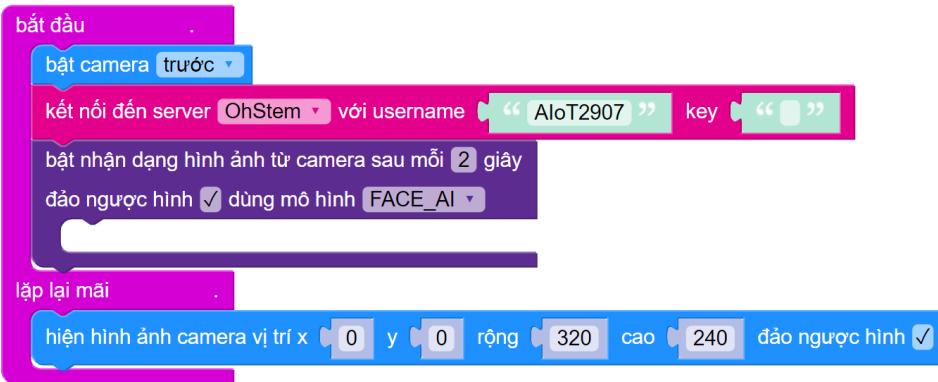
Với chương trình nhận dạng hình ảnh từ bộ não nhân tạo đã được huấn luyện trước, bạn cần sử dụng một khôi lệnh mới cho nó, với yếu tố thời gian đi kèm, như hình ảnh bên dưới:



Hình 22.6: Khôi lệnh nhận diện AI với thời gian

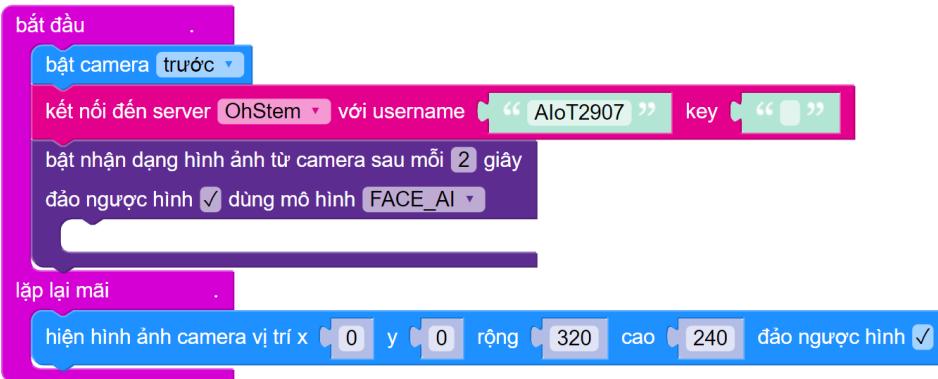
Khôi lệnh này được đặt ở vị trí cuối cùng trong nhóm **M-LEARNING**. Sở dĩ chúng ta phải sử dụng khôi lệnh có thời gian, là để hạn chế việc gửi dữ liệu lên tục lên server OhStem. Đối với một ứng dụng thông thường, bạn nên chọn thời gian là 2 giây hoặc thậm chí lâu hơn.

Cũng khá tương tự với chương trình liên quan đến giọng nói, chúng ta cũng bắt đầu chương trình nhận dạng với câu lệnh kết nối đến server OhStem, như sau:



Hình 22.7: Xây dựng chương trình nhận dạng AI với thời gian

Phần xử lý bên trong khôi nhận dạng AI khá tương tự với các bài hướng dẫn trước. Chúng ta chỉ có một thay đổi nhỏ, là thay vì gửi thông tin đến thiết bị đang kết nối, chúng ta sẽ gửi thông tin lên server OhStem. Trong ví dụ dưới đây, chúng tôi gửi kết quả nhận lện lên OhStem với kênh dữ liệu là **V11**.



Hình 22.8: Chương trình nhận dạng AI và gửi lên server OhStem

Bạn đọc có thể chủ động thiết kế một bảng điều khiển IoT đơn giản, với 2 khôi thông tin lần lượt liên kết với kênh dữ liệu V10 và V11. Sau đó, mở 2 trình duyệt độc lập để chạy đồng thời chương trình giọng nói và nhận dạng qua webcam. Kết quả nhận được sẽ như minh họa ở hình bên dưới.



Hình 22.9: Kết quả trên OhStem server

**Lưu ý:** Thông tin của **Username** của bảng điều khiển, chương trình giọng nói và chương trình AI phải giống nhau, trong ví dụ này là **AIoT2907**.

# CHƯƠNG 23

## Khung chương trình cho AI và IoT



## 1 Giới thiệu

Khi sự phát triển của AI và IoT đạt đến mức độ mà 2 công nghệ có thể giao thoa, các ứng dụng mới áp dụng được cả 2 công nghệ này càng nhiều, và chúng ta có một khái niệm mới khi nói về công nghệ 4.0, gọi là AIoT, tạm dịch là Trí tuệ nhân tạo của vạn vật kết nối. AIoT mở ra nhiều ứng dụng thông minh hơn với bộ não trí tuệ nhân tạo, bổ sung thêm thông tin cho các ứng dụng IoT, vốn dựa chính trên các cảm biến.

Kết nối vạn vật, là công nghệ giao tiếp những nhiều thiết bị với nhau (nhiều cảm biến giao tiếp với Yolo:Bit) và quan trọng, là những dữ liệu này được gửi đến một server kết nối vạn vật, để nhiều thiết bị khác, như điện thoại di động, như máy tính bảng có thể giám sát và điều hành. Trong giáo trình này, chúng ta đang sử dụng **OhStem server** để minh họa cho điều đó.

Trí tuệ nhân tạo, lại là một bộ não do chính chúng ta thiết kế và huấn luyện. Tùy vào ứng dụng cụ thể, trí tuệ nhân tạo có thể dùng để nhận dạng hình ảnh hoặc âm thanh. Nó bổ sung thêm cho hệ thống IoT một cảm biến cao cấp, và nâng cao cho các ứng dụng thời đại.

Một điều lưu ý vô cùng quan trọng là bộ não nhân tạo không được thực thi trên thiết bị vi điều khiển như Yolo:Bit, mà được thực hiện trên một máy tính hay di động, nơi có tài nguyên bộ nhớ và xử lý dồi dào hơn. Kết quả của việc xử lý sẽ được gửi định kì tới thiết bị, và các chức năng điều khiển sẽ được thực thi.

Để có thể đa dạng hóa các tính năng mới nói chung và các công nghệ nhân tạo nói riêng, cách tiếp cận của chúng ta là dựa vào server OhStem để làm cầu nối cho việc trung chuyển dữ liệu giữa chương trình AI và thiết bị Yolo:Bit sử dụng trong các ứng dụng thông minh.

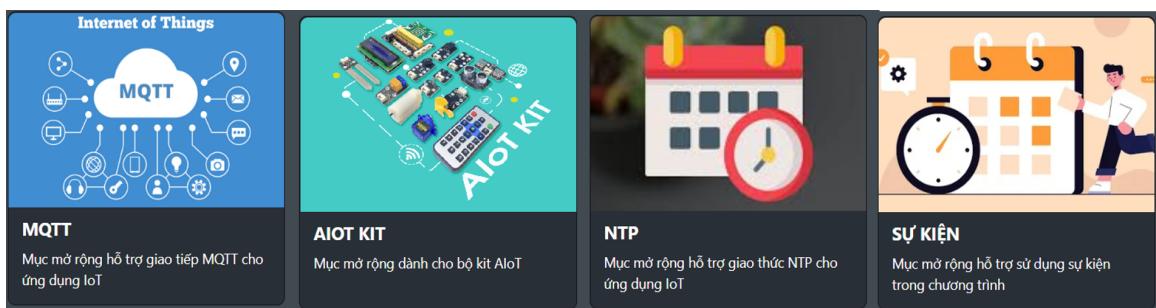
Các mục tiêu hướng dẫn trong bài này như sau:

- Thư viện cần thiết cho ứng dụng AIoT
- Tích hợp tính năng IoT vào chương trình
- Tích hợp tính năng AI vào chương trình

## 2 Thư viện cho ứng dụng AIoT

Thực ra, phần trí tuệ nhân tạo không được thực hiện ở mạch Yolo:Bit, mà được thực hiện trên máy tính hoặc các thiết bị di động. Do đó, các thư viện cần thiết cho bài hướng dẫn này, chủ yếu đến từ tính năng IoT và các thiết bị mở rộng là chính.

Đầu tiên, MQTT là thư viện không thể thiếu, do nó hỗ trợ trực tiếp các câu lệnh để kết nối vào mạng Internet và server IoT (trường hợp này là OhStem server). MQTT cũng hỗ trợ cho việc đăng ký kênh để nhận dữ liệu và gửi dữ liệu lên server.



Hình 23.1: Thư viện cho một ứng dụng IoT

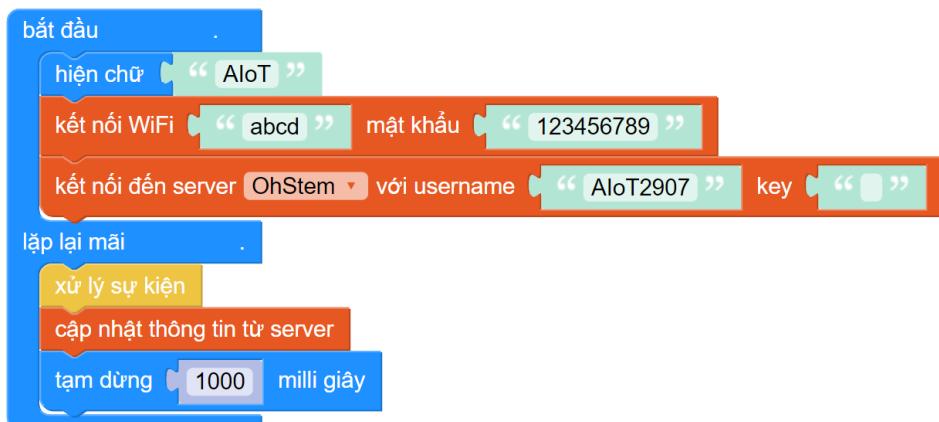
Tiếp theo, là thư viện **AIoT KIT**, hỗ trợ cho việc lập trình với thiết bị, bao gồm các cảm biến, màn hình LCD hay máy bơm. Thực ra, thư viện này được dựa hoàn toàn trên **HOME:BIT V3**. Bạn đọc có thể sử dụng thư viện cũ nếu như đã có kinh nghiệm với thư viện HOME:BIT V3.

Thư viện **SỰ KIỆN** với hỗ trợ chính là các câu lệnh định kì, sẽ đơn giản hóa cho việc hiện thực các tác vụ được thực hiện theo chu kỳ, vốn rất phổ biến trong các ứng dụng giám sát. Một số ví dụ về tính chu kỳ trong ứng dụng có thể kể ra như định kì gửi dữ liệu cảm biến lên server mỗi 30 giây hay định kì kiểm tra thời gian để bật máy bơm theo giờ.

Cuối cùng, là thư viện liên quan đến thời gian thực **NTP**. Mặc dù đây không phải là thư viện chính, nhưng sẽ bổ sung thêm một tính năng cho ứng dụng của bạn. Việc sử dụng thêm thư viện này cũng không làm ảnh hưởng quá nhiều đến hiệu suất của chương trình.

### 3 Khung chương trình IoT

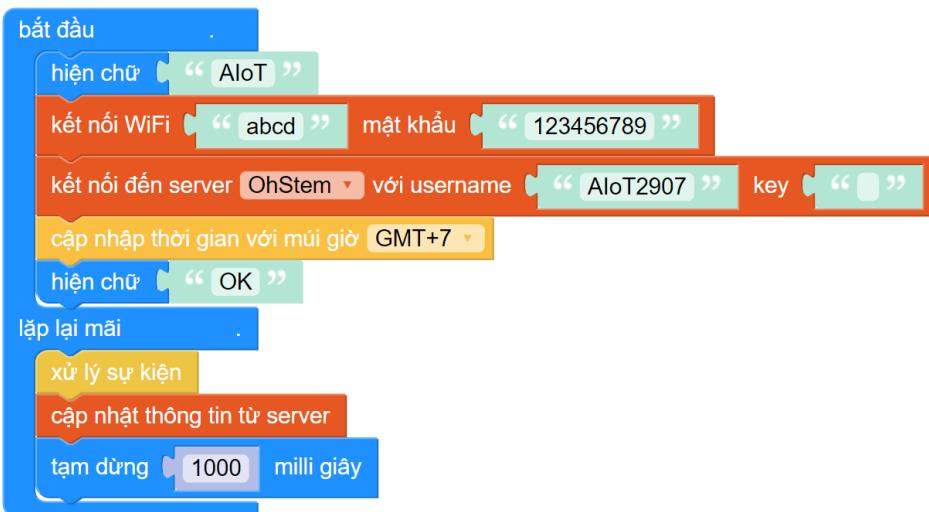
Chúng ta sẽ xây dựng chương trình cho phần IoT trước, với các chức năng kết nối Wifi và kết nối với OhStem server trong phần bắt đầu, như sau:



Hình 23.2: Các câu lệnh liên quan đến kết nối IoT

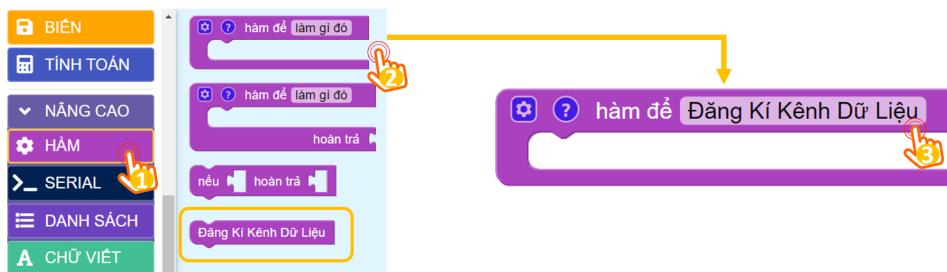
Trong từng dự án cụ thể, bạn đọc cần chọn thông tin **username** cho phù hợp với

bảng điều khiển IoT của mình. Tiếp theo, chúng ta sẽ bổ sung thêm các câu lệnh liên quan đến thời gian thực và sự kiện, như sau:



Hình 23.3: Khởi tạo cho thời gian thực và sự kiện

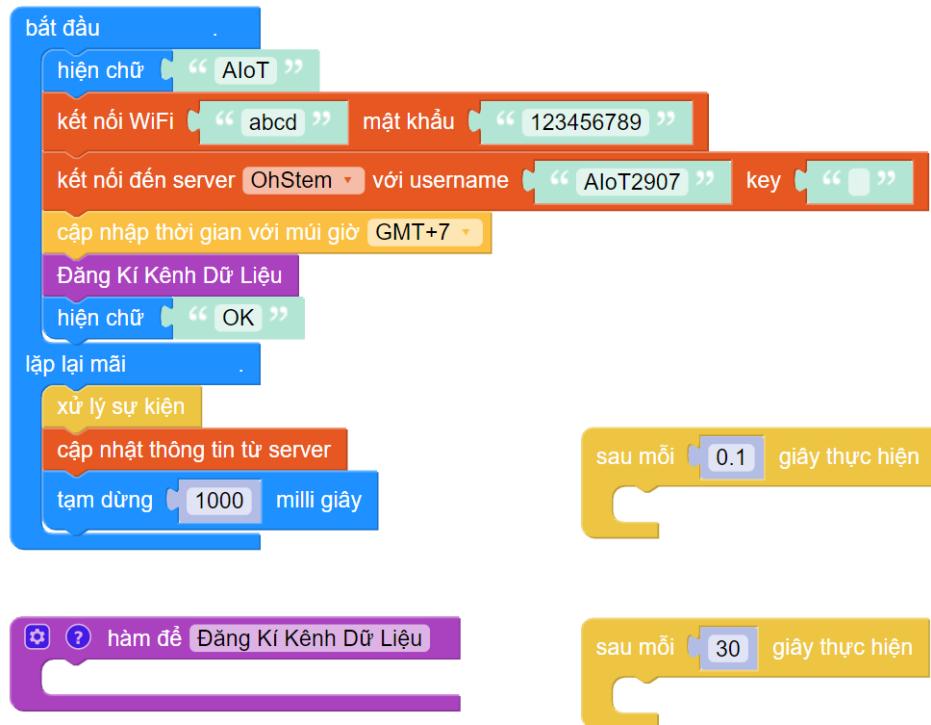
Sau phần khởi tạo, liên quan đến IoT, chúng ta cần 2 tính năng: đăng ký kênh nhận dữ liệu và định kì gửi dữ liệu lên server. Trước tiên, với việc nhận dữ liệu, do chúng ta chỉ cần thực hiện một lần, nên nó sẽ được hiện thực trong phần **bắt đầu**. Tuy nhiên để tiện cho việc tổ chức chương trình, chúng tôi sẽ xài một kĩ thuật, gọi là **chương trình con**, còn được gọi là **hàm** trên môi trường lập trình. Trình tự tạo một hàm được minh họa như sau:



Hình 23.4: Tạo hàm Đăng Kí Kênh Dữ Liệu

Bằng cách mở rộng khối **NÂNG CAO**, chọn vào **HÀM**, kéo thả khối đầu tiên ra màn hình lập trình, sau đó đổi tên hàm thành **Đăng Kí Kênh Dữ Liệu**, chúng ta sẽ có một khối lệnh mới xuất hiện trong phần **HÀM**, như minh họa ở hình trên.

Cuối cùng, sử dụng khối lệnh mới vào phần **bắt đầu**, đồng thời kéo thêm các khối lệnh chu kì trong phần **sự kiện**, chương trình của chúng ta sẽ như sau:



Hình 23.5: Tạo hàm Đăng Kí Kênh Dữ Liệu

Trong tương lai, việc định kí gửi dữ liệu lên server sẽ được hiện thực trong các khối lệnh định kí. Một chương trình sẽ có thể gồm **nhiều khối lệnh định kí**. Trong khi đó, việc đăng kí kênh và xử lý dữ liệu nhận được sẽ được hiện thực trong hàm **Đăng Kí Kênh Dữ Liệu**.

## 4 Tích hợp tính năng AI

Về phần AI, khi các kết quả đều đã được gửi lên OhStem server, việc tính hợp tính năng AI cho hệ thống thực sự rất đơn giản. Chúng ta chỉ cần đăng kí đúng kênh dữ liệu để nhận thông tin và xử lý nó, tương tự như xử lý thông tin cho nút nhấn trên bảng điều khiển IoT đã được hướng dẫn ở các bài trước.

Trong phần hướng dẫn này, chúng ta sẽ đăng kí 2 kênh dữ liệu là **V10** dành cho điều khiển giọng nói và kênh **V11** để nhận kết quả từ việc nhận dạng bằng hình ảnh. Ở đây, chúng tôi chỉ dùng màn hình hiển thị của Yolo:Bit để minh họa cho việc xử lý dữ liệu nhận được. Đối với kênh V10, chúng ta sẽ nhận được "A" hoặc "B" khi giọng nói có chứa những từ lệnh như bật đèn hay tắt đèn. Trong khi đó, với kênh dữ liệu V11, phần mềm nhận dạng AI từ hình ảnh sẽ cho 3 kết quả, là "NHAN", "NHUNG" và "ANH\_NEN".

Chương trình của chúng ta sẽ được hiện thực trong hàm **Đăng Kí Kênh Dữ Liệu**, như sau:



Hình 23.6: Tích hợp tính năng AI vào hệ thống

Trong chương trình trên, chúng tôi đang dùng những hình ảnh khác nhau trên mạch Yolo:Bit để minh họa cho nó có thể nhận được dữ liệu từ 2 chương trình AI trong một dự án. Việc tích hợp 2 công nghệ lớn là kết nối vạn vật và trí tuệ nhân tạo (nhận dạng âm thanh và hình ảnh) sẽ bổ sung đáng kể công cụ mới cho các dự án về STEM. Chương trình tham khảo cho phần hứng dẫn này được chia sẻ ở đường dẫn bên dưới.

<https://app.ohstem.vn/#!/share/yolobit/2HE4SpXc2finQHH58ASbBOSEZm6>

# **Phần V**

## **Hiện Thực Dự Án Yolo:Home**



# CHƯƠNG 24

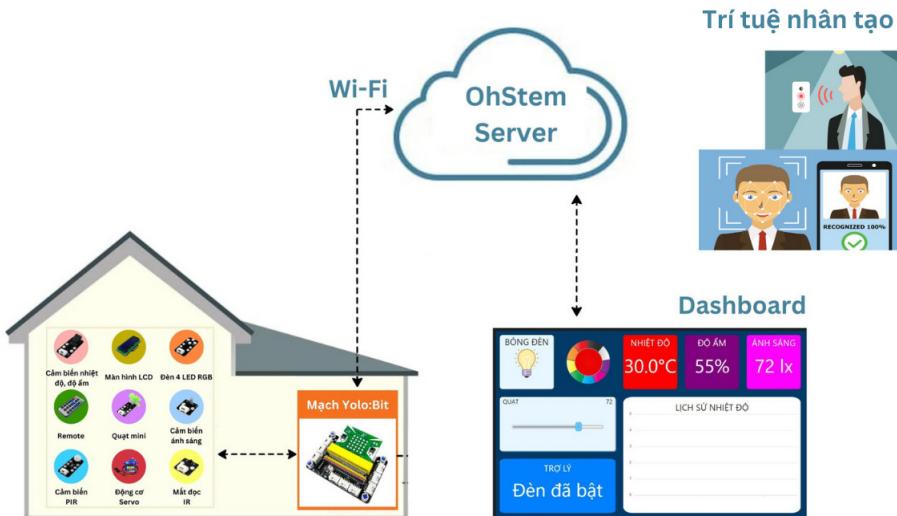
## Tổng quan dự án



# 1 Giới thiệu

Nhà thông minh SmartHome đã và đang là một chủ đề quá phổ biến trong giáo dục STEM. Trong giáo trình này, bên cạnh việc hiện thực các tính năng cơ bản, chúng tôi tích hợp thêm các dịch vụ mới liên quan đến kết nối vạn vật và trí tuệ nhân tạo.

Với những công nghệ mới này, dự án nhà thông minh sẽ trở nên sinh động và gần gũi hơn với bạn đọc, khi nó có thể nhận dạng giọng nói và cũng có thể phản hồi lại người sử dụng như một trợ lý ảo nhân tạo. Chúng tôi gọi tên cho dự án này là Yolo:Home.



Hình 24.1: Nhà thông minh dựa trên Kết nối vạn vật và Trí tuệ nhân tạo

Với các công nghệ được tích hợp vào trong dự án, chúng tôi hướng đến một nền tảng đa dụng cho các ứng dụng liên quan đến AI và IoT, hay còn gọi là AIoT. Với sự phát triển của khoa học và công nghệ 4.0, AIoT có thể được áp dụng trong nhiều lĩnh vực, nhà thông minh cũng là một trong số đó.

Bên cạnh các tính năng tự động hóa, như giám sát môi trường và điều khiển tự động các thiết bị, nhà thông minh hiện nay được bổ sung thêm công nghệ kết nối vạn vật, để nó có thể chia sẻ dữ liệu qua mạng Internet hay điều khiển thông minh với công nghệ trí tuệ nhân tạo.

Cũng trong dự án này, chúng tôi đưa ra một cách tiếp cận mới cho việc xây dựng chương trình kéo thả trên Yolo:Bit. Thực ra, nền tảng của mạch Yolo:Bit là hệ điều hành micro-python, do đó, nó có thể xử lý dựa trên các sự kiện, thay vì thực thi lệnh tuần tự như các cấu trúc lập trình khác. Kiến trúc mới này giúp bạn có thể dễ dàng mở rộng tính năng của hệ thống mà không gặp phải nhiều khó khăn.

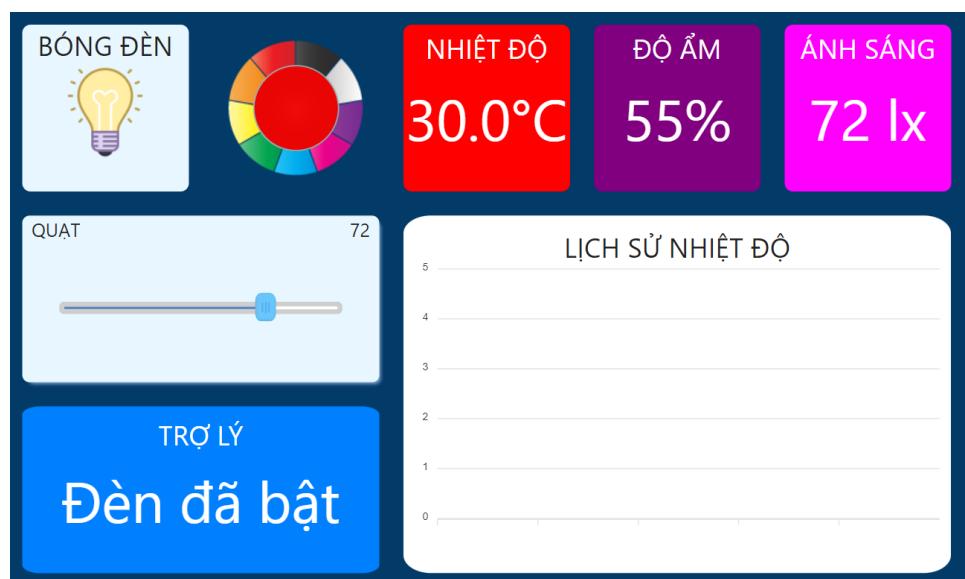
Phần tiếp theo của bài hướng dẫn này sẽ trình bày chi tiết các tính năng trong dự án. Với mỗi tính năng, phần cứng hoặc công nghệ liên quan sẽ được đề xuất và giới thiệu.

## 2 Giám sát môi trường

Đây là tính năng cơ bản nhất của ứng nhà thông minh. Các thông tin liên quan đến điều kiện trong nhà sẽ được giám sát một cách tự động và liên tục (chẳng hạn mỗi 5 phút một lần) để chúng ta có thể theo dõi điều kiện sống một cách liên tục và mọi lúc mọi nơi.

Trong dự án này, chúng tôi sẽ tập trung giám sát các thông tin liên quan đến môi trường không khí và ánh sáng, được trình bày chi tiết như sau:

- Nhiệt độ và độ ẩm không khí:** Đây là hai thông tin khá căn bản và cần thiết cho môi trường trong nhà. Với tính năng này, chúng ta sẽ sử dụng cảm biến DHT20 để giám sát.
- Điều kiện ánh sáng:** Mặc dù mạch Yolo:Bit đã có sẵn một cảm biến ánh sáng, nhưng chúng ta sẽ sử dụng cảm biến ánh sáng rời, dưới dạng một ngoại vi gắn vào mạch mở rộng. Không chỉ mang lại độ chính xác cao, việc lắp đặt và triển khai cảm biến này cũng sẽ dễ dàng hơn.



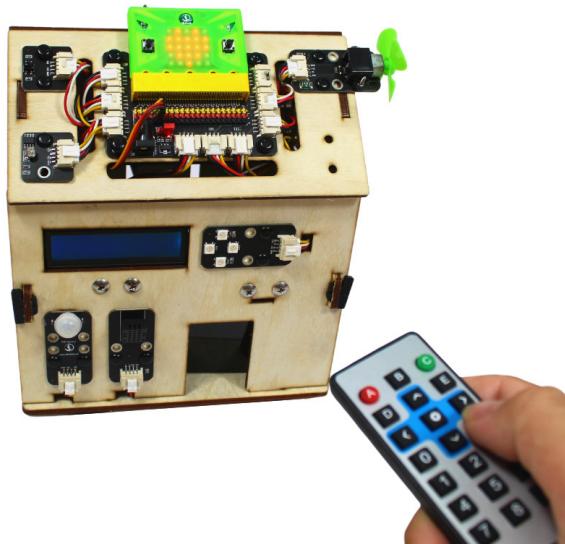
Hình 24.2: Bảng điều khiển IoT cho dự án Yolo:Home

Bên cạnh việc liên tục cập nhật các thông tin này lên OhStem Server để phục vụ nhu cầu giám sát từ xa, các thông tin sẽ được hiển thị trực quan tại ngôi nhà thông minh, thông qua màn hình kí tự LCD 16x2.

## 3 Cửa mật mã

Một trong những chủ đề thông dụng trong nhà thông minh, là khóa cửa với mật mã. Trong dự án này, chúng tôi tận dụng lại remote hồng ngoại làm công cụ để nhập mật mã.

Tuy nhiên, vấn đề của giao tiếp hồng ngoại là không ổn định. Để khắc phục vấn đề này, chúng tôi đưa ra một kiến trúc mới để xử lý nút nhấn với remote hồng ngoại và thêm phần âm thanh báo hiệu sau mỗi lần nút được nhấn, để gia tăng trải nghiệm cho người dùng. Theo trải nghiệm thực tế của chúng tôi, điều này là hoàn toàn chấp nhận được cho một dự án STEM.



Hình 24.3: Khóa cửa mật mã với Remote hồng ngoại

Bên cạnh đó, một tính năng cao cấp cho khóa cửa mật mã là khả năng thay đổi mật khẩu của hệ thống. Đây thực sự là tính năng phức tạp nhất trong cả dự án, nếu nói về việc xử lý chương trình.

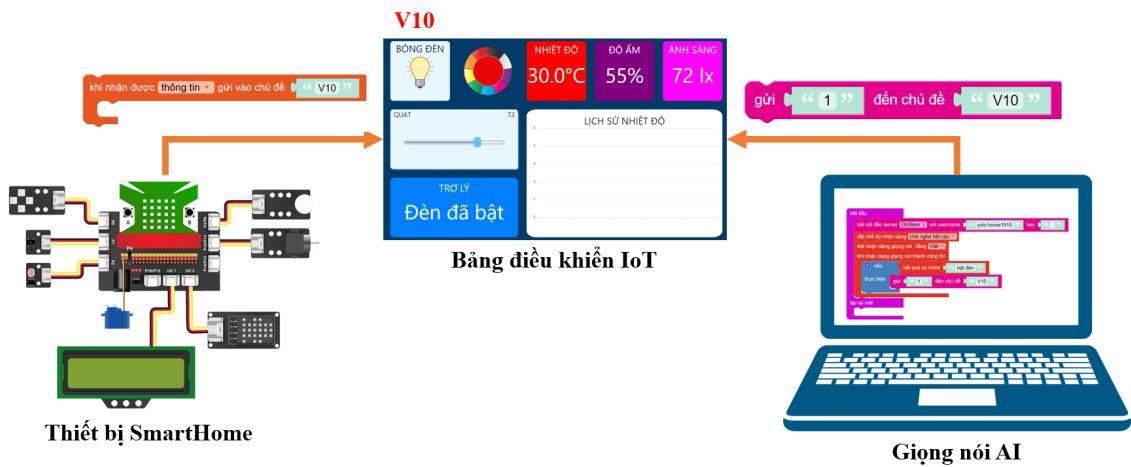
Trong tính năng này, chúng tôi đưa ra khái niệm về **quản lý trạng thái** trong chương trình. Hệ thống của chúng ta về căn bản sẽ có 2 trạng thái:

- Kiểm tra mật mã: Nếu đúng mật mã cửa sẽ mở, ngược lại, cửa sẽ đóng.
- Thay đổi mật mã: Nếu như mật mã bình thường để mở cửa được nhập 2 lần, hệ thống sẽ cho phép chúng ta nhập mật mã mới.

Việc chuyển đổi giữa 2 trạng thái này sẽ được định nghĩa bằng một điều kiện do chúng ta quy định. Chẳng hạn như khi bạn nhập lại mật khẩu 2 lần, và nhấn xác nhận, hệ thống sẽ chuyển từ trạng thái kiểm tra mật mã sang đổi mật mã.

## 4 Điều khiển thiết bị bằng giọng nói

Một trong những tính năng mới cho nhà thông minh, nhờ các công nghệ về Trí tuệ nhân tạo, là khả năng điều khiển thiết bị bằng giọng nói. Hiện nay, các công cụ chuyển đổi giọng nói sang văn bản đã khá phổ biến và nhận dạng tốt với tiếng Việt của chúng ta. Do đó, việc bắt một số từ khóa quan trọng để điều khiển thiết bị là hoàn toàn có thể.

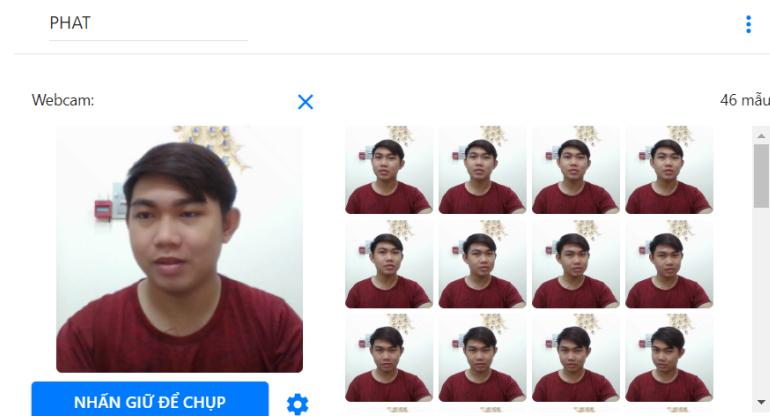


Hình 24.4: Kiến trúc hệ thống khi kết hợp AI và IoT

Chắc chắn những thế, chúng ta còn có thể kết hợp giữa Trí tuệ nhân tạo và Kết nối vạn vật, để hình thành nên ứng dụng mới theo kiến trúc AIoT. Với kiến trúc này, nhà thông minh đóng vai trò như một trợ lý ảo nhân tạo. Nó sẽ trở nên sinh động hơn và tương tác gần gũi hơn với chúng ta.

## 5 Nhận diện khuôn mặt - FaceAI

Một trong những lợi thế của trí tuệ nhân tạo, là tính linh hoạt của nó trong việc nhận dạng dựa vào hình ảnh. Chúng ta sẽ tận dụng tính năng này, để hệ thống có thể nhận dạng được một số định danh dựa vào hình ảnh khuôn mặt.



Hình 24.5: Nhận dạng khuôn mặt dựa vào trí tuệ nhân tạo

Mặc dù tính năng này không thể sánh với công nghệ FaceID đang được quảng bá rộng rãi, nó cũng một phần nào thể hiện được sức mạnh của dự án nhà thông minh, khi được tích hợp thêm với công nghệ mới. Điểm nổi bật của dự án này là việc nhận diện khuôn mặt và điều khiển bằng giọng nói được tiến hành song song.



# CHƯƠNG 25

## Khởi động dự án



## 1 Giới thiệu

Trước khi bắt đầu vào phần hiện thực dự án, chủ yếu là chương trình trên mạch Yolo:Bit, chúng ta cần chuẩn bị một số thiết lập ban đầu để sử dụng cho toàn bộ dự án.

Đầu tiên phải kể đến là kết nối phần cứng của các thiết bị. Thông thường đây sẽ là bước khó khăn với người mới bắt đầu, đặc biệt là trong việc lựa chọn thiết bị và chân kết nối với nó. Mỗi thiết bị có một đặc trưng về điện, và cần phải kết nối đúng nguyên lý thì mới có thể hoạt động được.



Hình 25.1: Mô hình cho dự án Nhà thông minh

Với các yêu cầu trong dự án, thiết kế giao diện trên bảng điều khiển IoT và lựa chọn các kênh dữ liệu cho cảm biến, quạt mini hay đèn chiếu sáng cũng là điều quan trọng, và cần thiết lập trước khi bắt đầu cho việc lập trình.

Cuối cùng, là khung chương trình sẽ sử dụng trong dự án. Khung chương trình cần được thiết kế chặt chẽ để có thể sử dụng nhiều công nghệ cùng 1 lúc, quan trọng nhất là IoT và AI.Thêm nữa, chương trình cần có khả năng mở rộng mà không bị ảnh hưởng đến các tính năng cũ đã hiện thực.

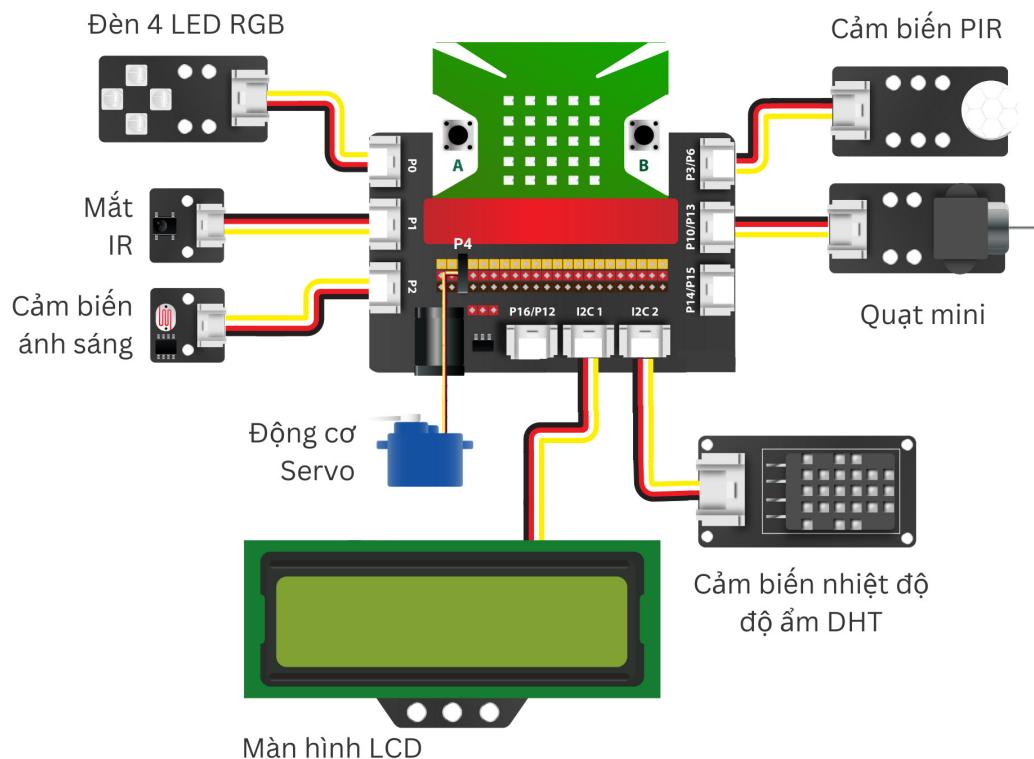
## 2 Kết nối phần cứng

Thiết bị phần cứng cho dự án chủ yếu tập trung ở các tính năng liên quan đến quan trắc môi trường và tự động hóa (tưới tiêu thông minh và phân loại trái cây). Dựa vào các thông tin về thiết bị ở Chương V, các thiết bị cần thiết cho dự án được tóm tắt trong bảng sau đây:

Bảng 25.1: Các thiết bị cần thiết cho dự án

	Chức năng	Thiết bị	Kết nối Khả dĩ	Kết nối Thực tế
Quan trắc môi trường	Nhiệt độ	DHT20	I2C1, I2C2	I2C2
	Độ ẩm			
	Ánh sáng	Cảm biến ánh sáng	P0, P1, P2	P2
	Hiển thị	LCD kí tự 16x2	I2C1, I2C2	I2C1
Cửa thông minh	Nhập mật mã	Remote và mắt nhận	Tất cả	P1
	Phát hiện có người	Cảm biến PIR	Tất cả	P3
	Đóng mở cửa	Động cơ RC	Tất cả	P4
Cảnh báo	Đèn báo hiệu	Đèn 3 màu RGB	Tất cả	P0

Chúng tôi đã bổ sung thêm một đèn RGB để làm báo hiệu cho dự án. Với khả năng hiển thị nhiều màu, bạn đọc có thể tận dụng nó để làm thông tin cảnh báo trong dự, chẳng hạn màu đỏ là cảnh báo bất thường hay màu xanh là báo hiệu cho việc nhận dạng chính xác một thông tin nào đó. Đối với các thiết bị cần kết nối chuyên dụng như DHT20, màn hình LCD hay các cảm biến, chúng ta sẽ ưu tiên lựa chọn kết nối cho nó trước. Trong khi đó, các ngoại vi có thể kết nối được với **tất cả** các khe mở rộng của Yolo:Bit, sẽ được chọn sau cùng. Dựa vào thông tin ở cột **Kết nối thực tế**, sơ đồ kết nối các thiết bị của chúng ta sẽ như trình bày dưới đây.



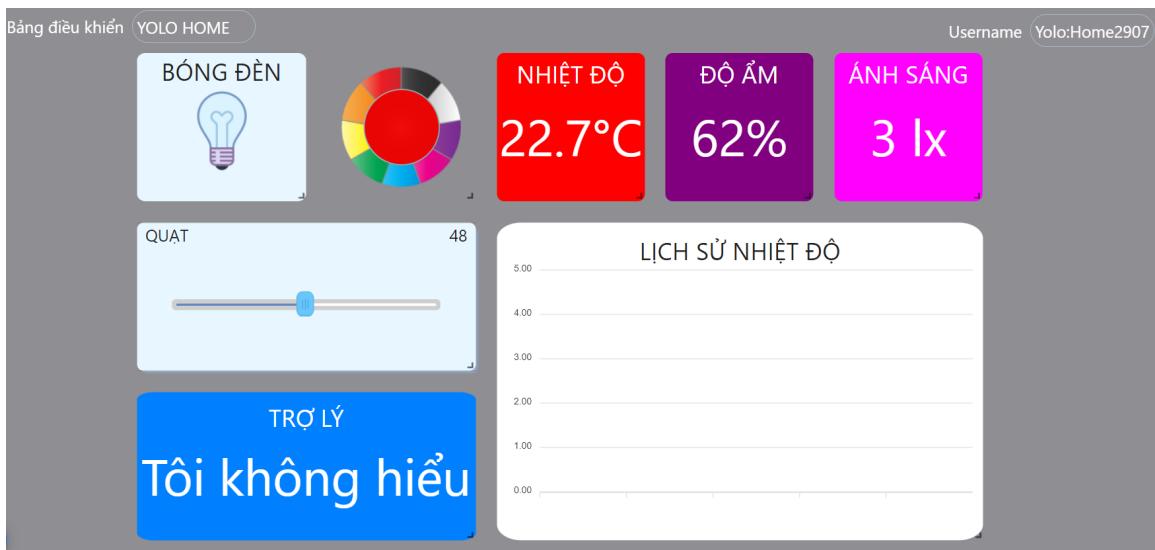
Hình 25.2: Kết nối phần cứng cho dự án

### 3 Thiết kế bảng điều khiển IoT

Ở phần này, chúng ta sẽ chuẩn bị cho giao diện trên bảng điều khiển IoT, với các tính năng giám sát dữ liệu cảm biến và điều khiển thiết bị. Phần giao diện này bạn đọc có thể chủ động sáng tạo. Chúng tôi đề xuất một giao diện với các tiêu chí sau đây:

- Các thông tin về cảm biến sẽ được biểu diễn một cách trực quan nhất, để người điều hành có thể thấy nhanh thông tin.
- Thông tin cần xem dưới dạng quá trình, chẳng hạn như nhiệt độ, sẽ được biểu diễn bằng đồ thị.
- Đèn sẽ được điều khiển bật tắt bằng nút nhấn và cũng có thể điều khiển được màu của nó.
- Quạt có thể điều khiển được tốc độ từ 0 đến 100.
- Một thông báo nhỏ, đóng vai trò như một trợ lý ảo của dự án.

Dựa vào các tiêu chí như trên, chúng tôi đề xuất giao diện trên bảng điều khiển IoT như sau:



Hình 25.3: Bảng điều khiển IoT cho dự án Yolo:Home

Sau khi chỉnh thông tin **Tên**, cách hiển thị cho mỗi đối tượng giao diện, thông tin tiếp theo mà bạn đọc phải lưu ý là **Kênh thông tin**. Với các đối tượng giao diện trên bảng điều khiển, chúng tôi đề xuất kênh thông tin như sau:

Bảng 25.2: Kênh thông tin cho các đối tượng giao diện

Đối tượng	Kênh thông tin
NHIỆT ĐỘ	V1
ĐỘ ẨM	V2
ÁNH SÁNG	V3
LỊCH SỬ NHIỆT ĐỘ	V1
BÓNG ĐÈN	V10
MÀU ĐÈN	V11
QUẠT	V12
TRỢ LÝ	V13

Trong bảng điều khiển, nhiệt độ không khí được biểu diễn dưới hai hình thức khác nhau: giá trị hiện tại và giá trị lịch sử ( thông qua đồ thị). Do đó, chúng sẽ lấy thông tin từ cùng một kênh dữ liệu, trong trường hợp này là **V1**.

Khi phân phối kênh dữ liệu cho các thiết bị điều khiển như đèn và máy quạt, chúng tôi chọn là **V10**, **V11** và **V12**. Điều này sẽ thuận tiện hơn cho việc mở rộng trong tương lai khi bạn đọc tích hợp thêm nhiều cảm biến, và có thể sử dụng V7, V8 hoặc V9. Điều đặc biệt trong bảng điều khiển này, là một ô hiển thị thông tin (kết nối với kênh **V13**), đóng vai trò như một trợ lý ảo trong dự án. Khi mạch Yolo:Bit muốn gửi thông tin thông báo gì đó đến người sử dụng, nó có thể gửi dữ liệu lên kênh này, chẳng hạn như thông báo "**Đèn đã bật**" hoặc "**Tôi không hiểu**" trong trường hợp nó không nhận được đúng lệnh cần xử lý.

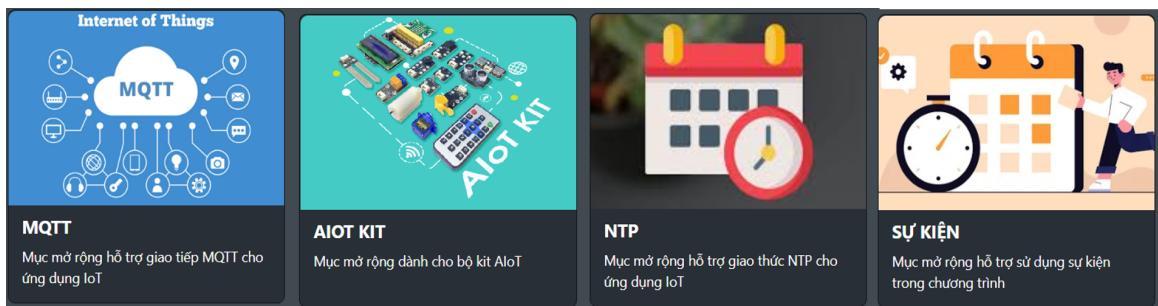
Cuối cùng, bạn cần đặt tên cho bảng điều khiển của mình, trong ví dụ của chúng tôi là **Yolo:Home2907**. Bạn đọc hãy đặt tên đi kèm với 1 mã số để hạn chế việc trùng nhau giữa nhiều bảng điều khiển IoT.

## 4 Khung chương trình AIoT

Với các yêu cầu về tính năng, chương trình của chúng ta sẽ cần 3 thư viện chính sau đây:

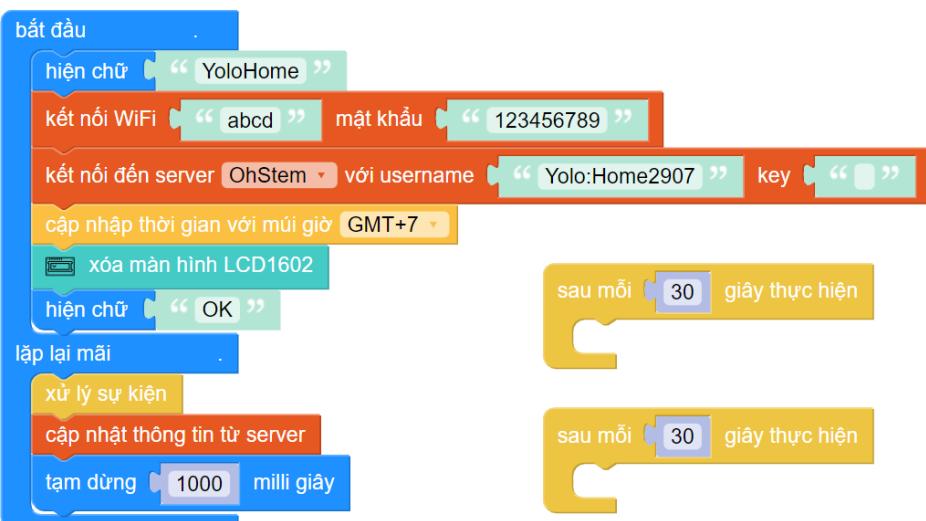
- AIoT KIT: Thư viện hỗ trợ chính cho việc tương tác với các thiết bị mở rộng, đọc giá trị từ cảm biến và xuất dữ liệu ra màn hình LCD
- MQTT: Thư viện hỗ trợ cho việc kết nối với mạng wifi và server OhStem.
- NTP: Thư viện hỗ trợ cho việc truy cập vào thời gian, giúp cho việc hiển thị thời gian hoặc ngày tháng.
- Sự kiện: Hỗ trợ khôi lệnh xử lý theo chu kỳ, hữu ích cho việc giám sát định kỳ các thông số liên quan đến cảm biến.

Khi chương trình có nhiều thư viện được sử dụng cùng một lúc, bạn đọc phải hết sức lưu ý đến việc khởi tạo nó cho đúng nguyên tắc. Ở phần bắt đầu, hệ thống cần



Hình 25.4: Thư viện cho việc lập trình trong dự án Yolo:Home

đăng nhập vào mạng Wifi, server OhStem, chỉnh múi giờ thành **GMT+7**. Bên cạnh đó, chúng ta cần có thêm 1 câu lệnh để xóa màn hình của LCD kí tự. Thực ra, câu lệnh này còn có một tác dụng nữa, là **khởi tạo LCD kí tự**. Thiếu câu lệnh này trong phần **bắt đầu**, LCD kí tự sẽ không hiển thị được.



Hình 25.5: Khởi tạo các tính năng chính trong chương trình

Khi sử dụng chương trình này, bạn đọc cần đảm bảo 4 thư viện liệt kê ở Hình 25.4 đã được thêm đầy đủ vào mạch Yolo:Bit. Thông tin về mạng Wifi cũng như tên của bảng điều khiển IoT (username) cần được hiệu chỉnh lại cho phù hợp. Chương trình trên được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2H8afw04WSeolzHuyvsUFmqs0qb>

# CHƯƠNG 26

## Giám sát chất lượng không khí



## 1 Giới thiệu

Trong bài hướng dẫn này, chúng ta sẽ tập trung vào tính năng đầu tiên của dự án Yolo:Home, là giám sát điều kiện môi trường trong nhà. Các thông số liên quan tới không khí (nhiệt độ và độ ẩm), và ánh sáng sẽ được giám sát định kì trong dự án. Bên cạnh đó, các thông tin phụ như ngày tháng năm và thời gian cũng sẽ được theo dõi trong ứng dụng.



Hình 26.1: Giám sát các thông tin về môi trường trong dự án Yolo:Home

Trong phần đầu của tính năng này, chúng ta sẽ tiếp cận với các tính năng truyền thống của một ứng dụng nhà thông minh, là hiển thị trực tiếp dữ liệu quan trắc lên màn hình LCD để người dùng có thể giám sát được thông tin.

Tiếp sau đó, một tính năng đặc thù cho khả năng xử lý tự động trong nhà thông minh sẽ được tích hợp vào hệ thống, đó là đèn sẽ tự động sáng khi có người chuyển động gần cảm biến PIR. Đây sẽ là tính năng tiêu biểu mà chúng tôi giới thiệu trong giáo trình này để dựa vào nó, bạn có thể chủ động tự phát triển thêm các tính năng tự động khác.

## 2 Hiển thị thông tin trên LCD

Thực ra, việc hiển thị thông tin lên LCD là không phức tạp khi chúng ta có sự hỗ trợ của thư viện lập trình AIOT-KIT. Tuy nhiên, chúng ta cần phải thiết kế bô cục trên màn hình có 16 dòng và 2 cột sao cho cân đối và đẹp mắt thì mới tạo ấn tượng cho ứng dụng. Trong giáo trình này, chúng tôi gợi ý bô cục như sau:

- Nhiệt độ và độ ẩm không khí và cường độ ánh sáng sẽ hiển thị ở dòng 1.

- Thông tin về ngày tháng năm và thời gian (giờ và phút) sẽ hiển thị ở dòng 2.

Do mỗi hàng chỉ hiển thị được tối đa 16 kí tự, chúng ta sẽ cố gắng sắp xếp thông tin cho vừa với màn hình. Chúng tôi gợi ý cho bạn đọc cách sắp xếp thông tin như sau:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	5	.	7	*	C			6	3	%			2	1	%
1	2	1	/	1	0	/	2	0	2	2		1	3	:	3	8

Hình 26.2: Hiển thị thông tin trên màn hình LCD kí tự

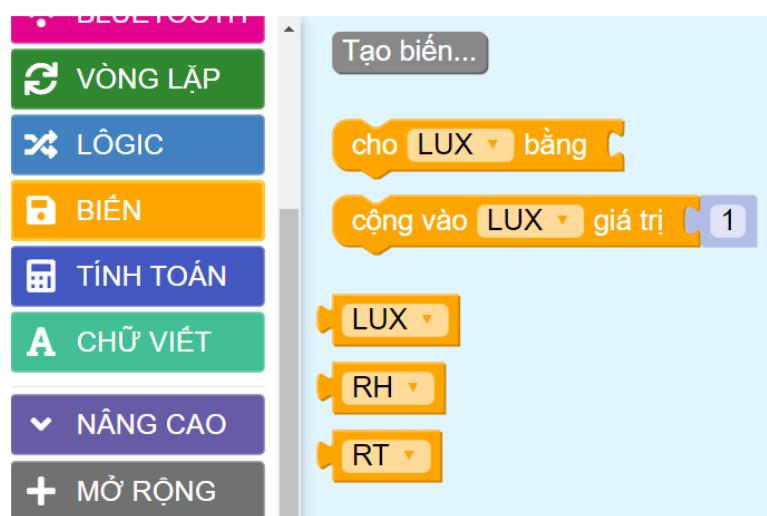
Trong cấu trúc trên, ở hàng 1 hiển thị lần lượt các giá trị về nhiệt độ, độ ẩm và cường độ ánh sáng. Trong khi đó, ở dòng thứ 2 sẽ là thông tin về thời gian.

### 3 Hiện thực chương trình

Các chức năng liên quan đến quan trắc thông số cảm biến sẽ được thực hiện định kì, với chu kỳ tương đối dài. Trong dự án này, chúng tôi chọn 30 giây sẽ quan trắc một lần. Trong thực tế, bạn đọc có thể tăng thời gian này lên nếu cần thiết. Trình tự hiện thực cho tính năng này được trình bày chi tiết như sau.

**Bước 1:** Tạo biến lưu trữ cho các thông tin cảm biến.

Từ chương trình khung, 3 biến số, lần lượt là **RT**, **RH** và **LUX** sẽ được tạo thêm để lưu các thông tin liên quan đến cảm biến, như kết quả ở hình sau đây:



Hình 26.3: Tạo biến số cho việc đọc thông tin từ cảm biến

**Bước 2:** Đọc thông tin từ các cảm biến.

Chức năng này sẽ được hiện thực trong khôi lệnh định kì 30 giây trong chương trình khung. Kết quả của chức năng này như sau.

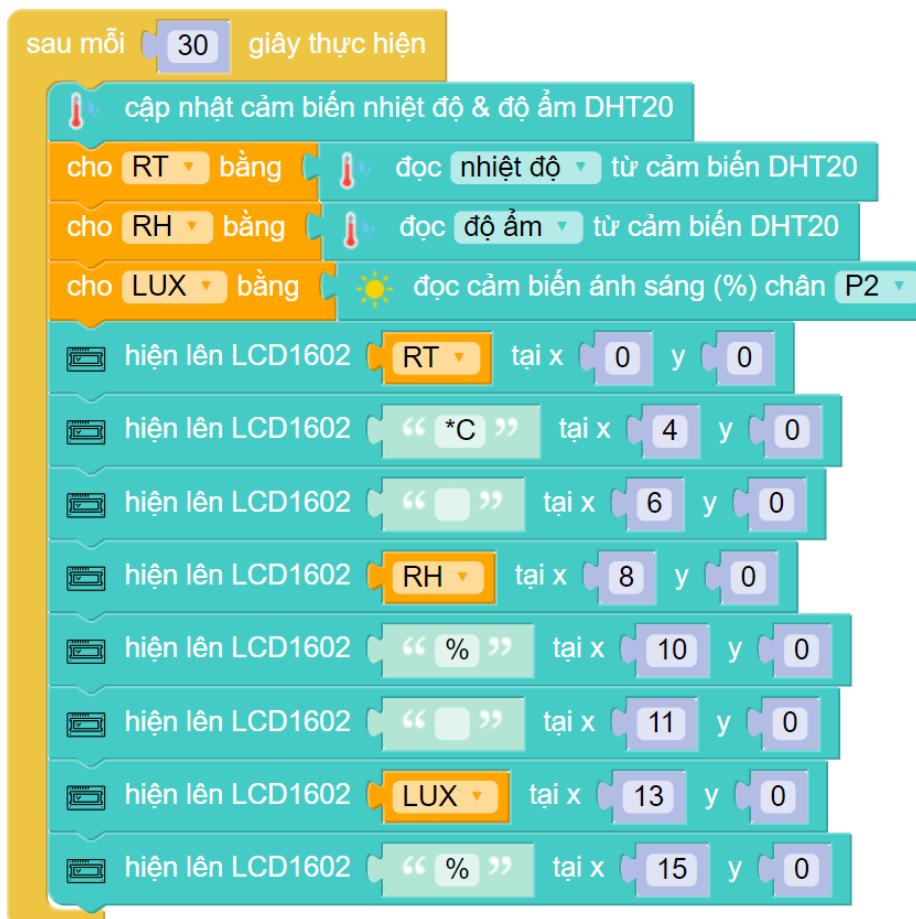


Hình 26.4: Đọc các giá trị cảm biến và lưu vào biến số

Nguyên tắc đọc giá trị của từng cảm biến đã được trình bày ở các bài trước đó, nên chúng tôi sẽ không trình bày lại chi tiết ở bài này.

### Bước 3: Hiển thị thông tin dòng thứ nhất trên LCD.

Theo cấu trúc được gợi ý trên Hình 26.2, bạn đọc có thể dễ dàng định vị được tọa độ để hiện thị các thông tin chính xác. Các câu lệnh cần thiết cho việc này được trình bày như sau:



Hình 26.5: Hiển thị dòng đầu tiên trên LCD

Bạn đọc để ý rằng, đằng sau đơn vị của nhiệt độ sẽ có thêm kí tự khoảng trắng.

Chúng ta tận dụng điều này để không phải xóa thông tin cũ khi hiển thị trên LCD, bởi trên dòng đầu tiên, chúng ta hiển thị hết 16 kí tự trên LCD.

#### Bước 4: Hiển thị thông tin dòng thứ hai trên LCD.

Bám sát theo cấu trúc hiển thị trình bày ở Hình 26.2, chúng ta sẽ tiếp tục hiện thực cho hàng thứ 2 trên LCD. Tuy nhiên, điểm thú vị ở đây là chúng tôi sẽ dùng một khối lệnh chu kì khác để làm việc này. Chương trình gợi ý để hiển thị dòng thứ 2 như sau:



Hình 26.6: Hiển thị dòng thứ hai trên LCD

Bản thân các khối lệnh sự kiện thực hiện theo chu kì, nó được thực hiện song song và độc lập nhau. Do đó, chúng ta có thể xử lý độc lập 2 dòng trên LCD với những chu kì khác nhau, chẳng hạn như 10 giây để cập nhật thông tin cảm biến, nhưng 30 giây chúng ta mới cập nhật thời gian trên dòng thứ 2.

Bạn hãy lưu ý rằng trong chương trình trên, chúng ta chưa hiển thị dấu hai chấm trong thời gian. Tính năng này sẽ được hiện thực ở bước tiếp theo

#### Bước 5: Hiện thực dấu hai chấm cho thời gian.

Dấu hai chấm trong chương trình sẽ được chớp tắt mỗi giây một lần. Và một lần nữa, chúng ta sẽ đối với nó một cách độc lập trong một khối lệnh chu kì khác. Chương trình gợi ý cho phần này như sau:



Hình 26.7: Gửi dữ liệu lên bảng điều khiển IoT

Việc lựa chọn chu kì 2 giây cho chức năng này được giải thích như sau. Đầu tiên, chúng ta sẽ cho hiện dấu hai chấm trong vòng 1 giây. Sau đó, **kí tự khoảng trắng** sẽ hiện ra cũng trong vòng 1 giây. Tổng cộng chu kì để hiện thực tính năng này là **2 giây**. Do đó, bạn đọc phải chọn chu kì của chức năng này là **2 giây trở lên** thì một lần thực hiện của nó mới được thực hiện trọn vẹn. Việc hiện **một kí tự khoảng trắng** sẽ tạo hiệu ứng ẩn dấu hai chấm đi trong 1 giây.

## 4 Bật đèn tự động

Với tính năng này, nhà thông minh sẽ tự động bật khi có người di chuyển gần với cảm biến phát hiện chuyển động. Chương trình phải thường xuyên kiểm tra giá trị từ cảm biến PIR và điều khiển đèn sáng tắt tương ứng. Ở đây, chúng tôi sẽ sử dụng 25 đèn của Yolo:Bit để bật tắt tương ứng với trạng thái của cảm biến PIR.

Với một tính năng tự động, cần kiểm tra thường xuyên giá trị từ cảm biến để điều khiển như trên, khôi lệnh xử lý theo chu kì lại một lần nữa, là một công cụ hữu ích cho bạn đọc. Một khôi xử lý theo chu kì nữa sẽ được thêm vào chương trình, với chương trình bên trong nó được gợi ý như sau:



Hình 26.8: Tự động bật đèn khi có người

Chu kì của chức năng này, phản ánh **mức độ đáp ứng** của hệ thống nhanh hay chậm khi có người. Chúng ta sẽ luôn mong muốn rằng hệ thống có thể đáp ứng nhanh nhất, với đèn được bật ngay lập tức khi có chuyển động. Do đó, một chu kì nhỏ sẽ được sử dụng trong trường hợp này, thay vì là vài giây.

Tuy nhiên, để cân bằng hiệu suất của hệ thống, chúng tôi đề xuất chu kì xử lý là **0.1 giây = 100 mili giây**. Với người sử dụng, độ trễ 100 mili giây là hoàn toàn chấp nhận được. Câu lệnh bật tắt đèn có thể tìm trong mục LED.

Tính năng này mở ra một phương pháp để hiện thực các tính năng tự động trong dự án. Chẳng hạn như phát ra loa nếu cảm biến lửa tích cực hay cảnh báo nếu như tràn nước ở bể chứa (cảm biến độ ẩm đất có thể được áp dụng trong trường hợp này). Tất cả những gì chúng ta cần làm là kéo thêm khối xử lý theo chu kì vào hệ thống. Việc tổ chức chương trình như hiện tại là rất linh động để mở rộng hệ thống. Hình ảnh của toàn bộ chương trình lúc này như sau:



Hình 26.9: Chương trình giám sát thông tin và điều khiển tự động

Chương trình của dự án đến lúc này được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2GT3KSHnUgyVbHPpT9jEaG4EPln>



# CHƯƠNG 27

## Cửa mật mã



## 1 Giới thiệu

Chủ đề cửa mật mã khá thông dụng trong ứng dụng nhà thông minh. Trong ứng dụng trình bày ở bài hướng dẫn này, remote hồng ngoại sẽ được sử dụng để nhập mật mã cho hệ thống. Tính năng đầu tiên của cửa mật mã, là cho phép người dùng nhập vào mật mã, và nhấn xác nhận. Nếu như mật mã chính xác, cửa sẽ được mở. Ngược lại, cửa sẽ không được mở và một thông báo sẽ hiện lên để cảnh báo. Một động cơ RC Servo kết nối với chân P4 sẽ được sử dụng để minh họa cho việc đóng mở cửa.

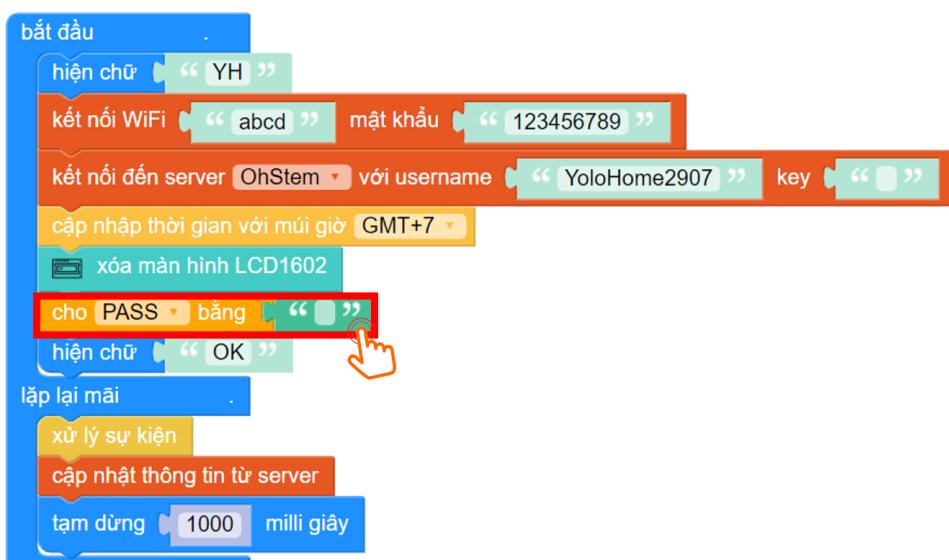
Tính năng thứ hai, trong ứng dụng này là việc thay đổi mật khẩu của hệ thống. Để hệ thống hợp lý hơn, chỉ khi nào nhập đúng mật mã, chúng ta mới được phép đổi mật mã. Sau khi nhấn một nút để xác nhận đổi mật khẩu, mật khẩu mới sẽ được lưu lại.

## 2 Kiểm tra mât mā

Đây là tính năng đầu tiên của hệ thống cửa mật mã. Tạm thời, chúng ta sẽ quy định mật khẩu là chuỗi kí tự cố định "**123**". Bạn đọc có thể tự mở rộng độ dài của mật khẩu và hỗ trợ nhiều kí số hơn trong tương lai. Chúng tôi cũng định nghĩa nút F trên bàn phím là nút dùng để kiểm tra mật khẩu.

Để có thể nhận được tín hiệu từ remote hồng ngoại một cách ổn định, chúng ta cần xử lý nó theo một hướng khác. Thay vì dùng khôi lệnh sự kiện, chúng ta sẽ dùng khôi lệnh chu kì để kiểm tra dữ liệu từ remote. Chi tiết từng bước để hiện thực được trình bày như bên dưới.

**Bước 1:** Tạo một biến số, đặt tên là **PASS** và khởi tạo nó trong phần **bắt đầu**.

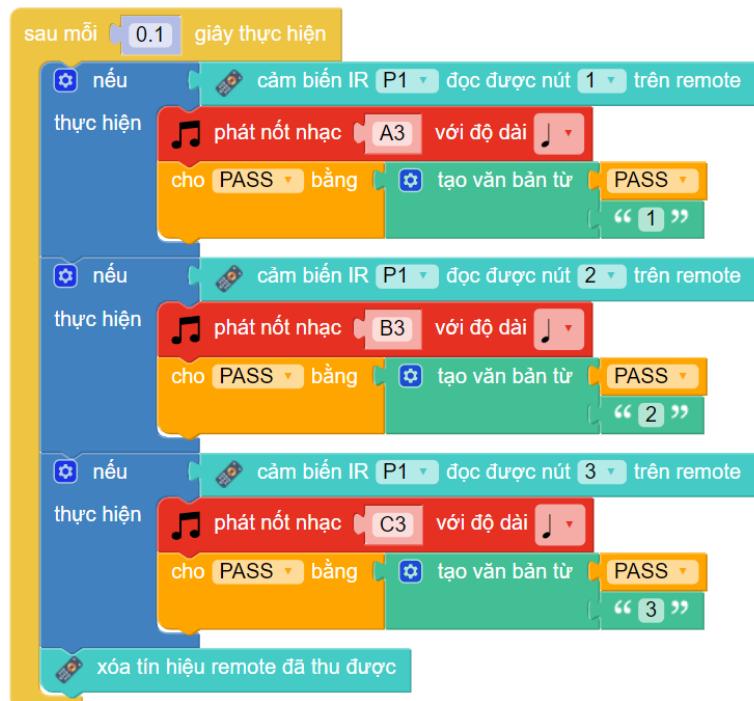


Hình 27.1: Khởi tạo cho biến PASS

Câu lệnh khởi tạo giá trị cho biến **PASS** nên được đặt trước câu lệnh hiển thị ra thông báo **OK**. Vì chúng ta sẽ xử lý trên chuỗi, nên giá trị khởi tạo cho biến sẽ là **chuỗi rỗng**, có thể được tìm thấy trong nhóm lệnh **CHỮ VIẾT**.

### Bước 2: Tạo khôi lệnh chu kì 0.1s cho phần nhận lệnh từ remote.

Với cách tiếp cận này, bạn đọc có thể mở rộng tính năng của chương trình rất dễ dàng và không làm ảnh hưởng đến các tính năng cũ. Đối với việc nhận lệnh bằng remote, chúng tôi sẽ sử dụng câu lệnh chu kì với thời gian là 0.1s, như minh họa ở hình bên dưới:



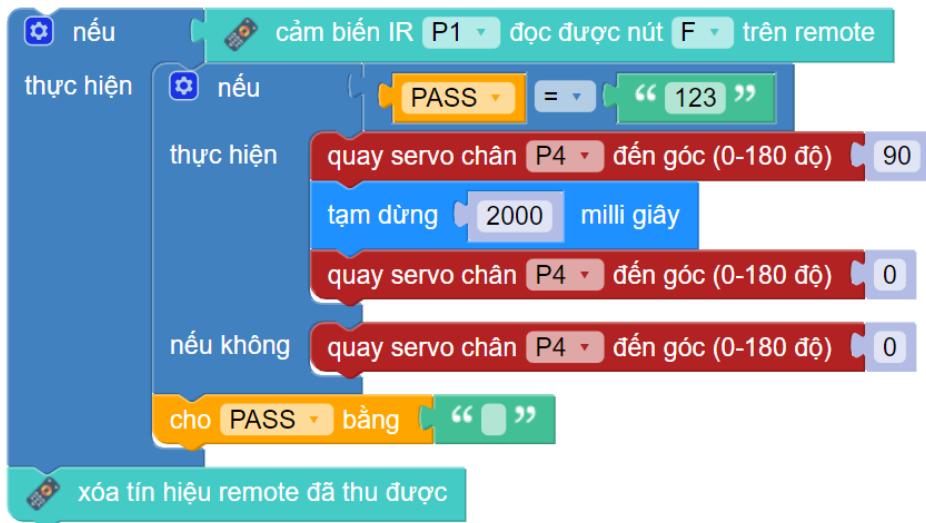
Hình 27.2: Nhận lệnh từ remote bằng câu lệnh sự kiện

Bạn đọc hãy lưu ý việc sử dụng thêm câu lệnh âm thanh, để tương tác của hệ thống với người sử dụng thêm sinh động và thân thiện. Mỗi khi nhận được lệnh, hệ thống sẽ phát ra một âm thanh khác nhau. Bạn cần lưu ý rằng, sẽ có một độ trễ nhỏ sau khi nhấn remote, bởi chu kỳ kiểm tra của chúng ta là 0.1s.

Cuối cùng, là để có thể nhận tiếp tín hiệu mới, chúng ta cần phải **xóa tín hiệu remote đã thu được** ở cuối khôi lệnh xử lý bên trên.

### Bước 3: Hiện thực khôi kiểm tra mật khẩu và đóng mở cửa cho nút F.

Theo định nghĩa cho tính năng của cửa mật khẩu, nút F trên bàn phím hồng ngoại sẽ được sử dụng để kiểm tra mật khẩu. Tuy nhiên, trước khi ráp vào khôi lệnh sự kiện bên trên, chúng ta nên hiện thực nó ở bên ngoài. Khi mật khẩu là chính xác, Yolo:Bit sẽ điều khiển một động cơ servo nối với chân P4 để mở cửa. Phần chương trình cho chức năng này sẽ được hiện thực như sau:



Hình 27.3: Các câu lệnh để kiểm tra mật khẩu và mở cửa

Trong chương trình trên, bạn cần lưu ý một số điểm quan trọng như sau:

- Nếu như mật khẩu là chính xác (trong ví dụ này là chuỗi 123), cửa sẽ được mở trong vòng 2 giây.
- Sau khi thực hiện việc điều khiển động cơ, chúng ta sẽ tắt nó trong 2 giây tiếp theo, để tiết kiệm điện cho hệ thống.
- Cho dù mật khẩu (lưu trong biến PASS) là đúng hay sai, chúng ta đều phải cho nó về chuỗi rỗng và chờ các tín hiệu điều khiển từ remote cho các lần tiếp theo.

**Bước 4:** Hoàn thiện chương trình cho cửa mật mã.

Cuối cùng, chúng ta sẽ ráp các khối lệnh cho phần kiểm tra mật mã vào bên trong khối lệnh chu kỳ. Bạn đọc hãy lưu ý về vị trí của câu lệnh **xóa tín hiệu remote đã thu được**, nó sẽ luôn nằm cuối trong chương trình xử lý với nút nhấn từ remote hồng ngoại. Hình ảnh của toàn bộ chương trình xử lý dành cho cửa mật mã sẽ như sau:



Hình 27.4: Hoàn thiện chức năng của mật mã

Bạn đọc có thể tự làm phong phú thêm chương trình với nhiều phím số có sẵn trên bàn phím. Ở ví dụ này, chúng tôi chỉ sử dụng các phím số 1, 2 và 3 mà thôi. Một mật mã dài hơn cũng có thể được lựa chọn cho hệ thống của bạn.

### 3 Thay đổi mật mã

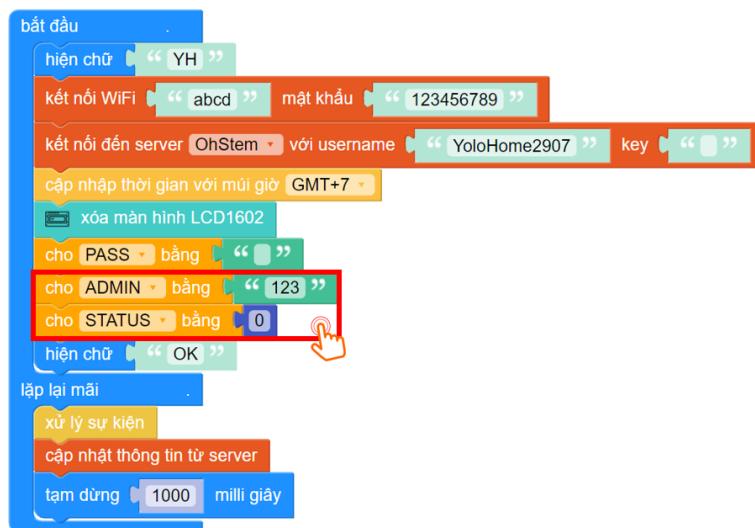
Trong một ứng dụng phức tạp hơn, chúng ta có thể bổ sung thêm tính năng thay đổi mật khẩu cho hệ thống cửa mật mã. Đây là tính năng tương đối phức tạp, vì chúng ta cần phải làm cho việc đổi mật mã này có độ bảo mật tương đối tốt. Ở đây, khi nhập mật mã 2 lần (đối với hệ thống hiện tại là chuỗi "123123") và nhấn nút F, hệ thống sẽ chuyển sang chế độ đổi mật mã. Khi nhấn nút F một lần nữa, hệ thống sẽ trở về chế độ kiểm tra mật mã thông thường. Bạn đọc có thể tự định nghĩa ra quy trình thay đổi mật mã cho riêng mình. Tuy nhiên, thông thường độ phức tạp của việc này sẽ cao hơn nhiều so với bước kiểm tra mật mã.

Với yêu cầu đặc tả như trên, các bước chính để hiện thực tính năng này như sau:

- Khai báo thêm biến **ADMIN**, dùng để lưu mật mã hiện tại của hệ thống.
- Khi nhấn nút F, chúng ta sẽ phải kiểm tra 2 điều kiện, là mật khẩu từ người dùng là để mở cửa thông thường hay là để đổi mật mã mới.
- Khai báo thêm biến **STATUS**, để quản lý trạng thái hiện tại của hệ thống, với 0 là trạng thái kiểm tra mật mã và 1 là trạng thái đổi mật mã.

Quá trình hiện thực thêm tính năng đổi mật mã được trình bày chi tiết như sau:

### Bước 1: Khởi tạo biến ADMIN và STATUS trong phần **bắt đầu**.

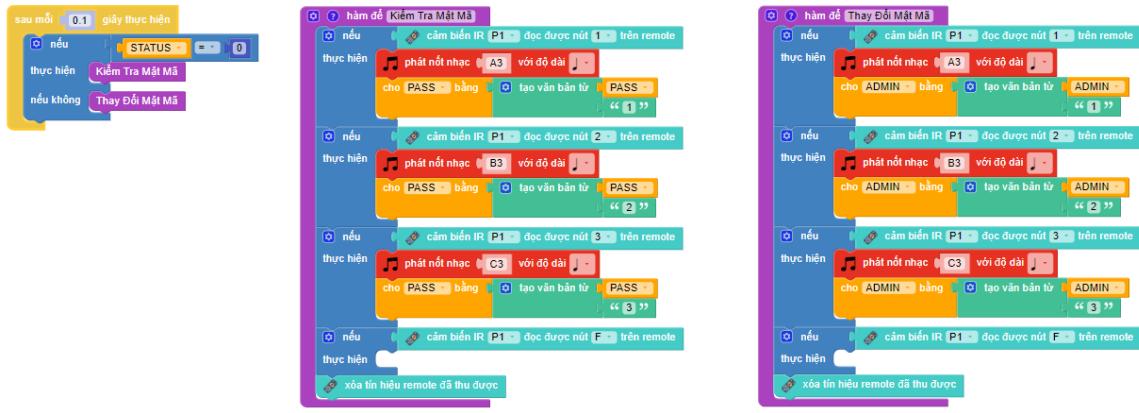


Hình 27.5: Khởi tạo giá trị cho biến ADMIN và STATUS

Biến **ADMIN** sẽ lưu giữ mật khẩu hiện tại của hệ thống. Ở trạng thái mặc định, hệ thống sẽ kiểm tra mật khẩu, với giá trị ban đầu của biến **STATUS** bằng 0.

### Bước 2: Phân loại xử lý trong khối lệnh định kì 0.1s.

Rõ ràng, với 2 giá trị khác nhau của biến **STATUS**, chúng ta cần có 2 khối xử lý độc lập, cho phần kiểm tra mật mã và thay đổi mật mã. Điều này sẽ được hiện thực bằng hai khối hàm tương ứng, như minh họa ở hình bên dưới:



Hình 27.6: Hai khối hàm cho phần xử lý định kì

**Khối hàm để** có thể được tìm thấy trong phần **NÂNG CAO**, và chọn tiếp vào **HÀM**. Việc xử lý ban đầu cho 2 khối này khá giống nhau, ngoài việc biến xử lý cho phần **Kiểm Tra Mật Mã** là **PASS** còn khối **Thay Đổi Mật Mã** là **ADMIN**.

### Bước 3: Hiện thực nút F cho phần **Kiểm Tra Mật Mã**.

Nút **F** cho phần này sẽ khá phức tạp, bởi nó có 2 nhiệm vụ cần phải kiểm tra. Nếu như người dùng nhập vào 2 lần mật mã, hệ thống sẽ chuyển qua trạng thái mới, là **Thay Đổi Mật Mã**. Ngược lại, chương trình sẽ chỉ kiểm tra mật mã (bằng cách so sánh với biến **ADMIN**) như bình thường. Chương trình gợi ý cho phần này sẽ như sau:



Hình 27.7: Xử lý nút F cho phần **Kiểm Tra Mật Mã**

Bạn đọc hãy lưu ý vào câu lệnh gán giá trị cho biến **STATUS**. Đây là câu lệnh khá quan trọng trong hệ thống của chúng ta, tạm gọi là câu lệnh chuyển trạng thái.

### Bước 4: Hiện thực nút F cho phần **Thay Đổi Mật Khẩu**.

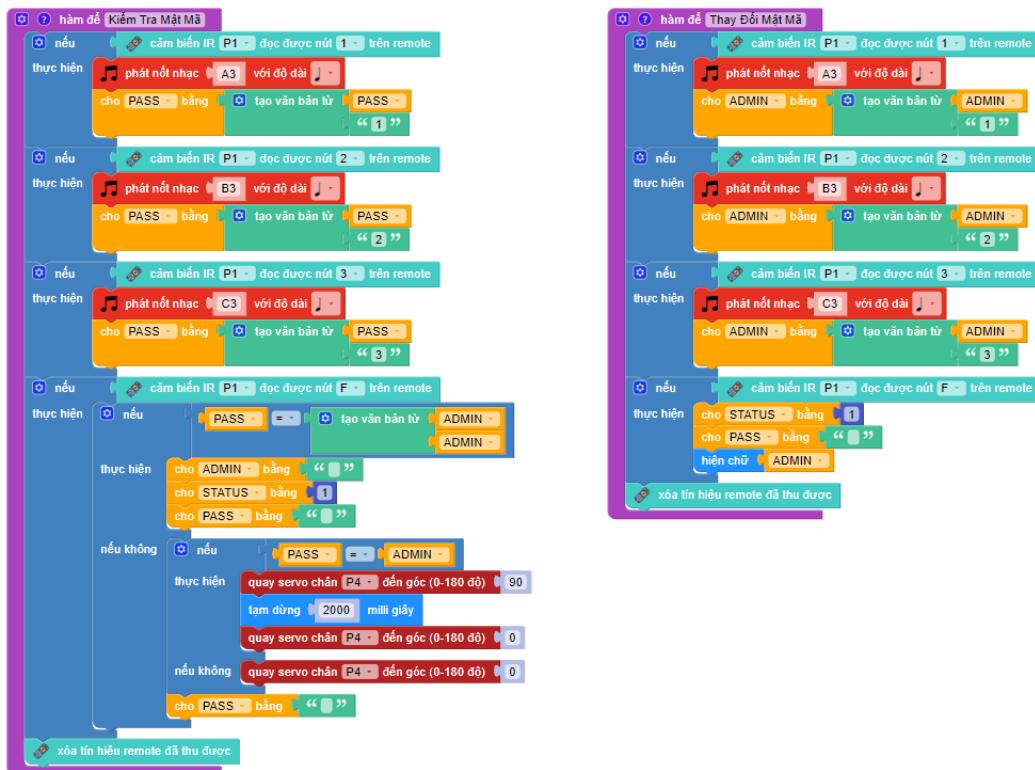
Phần chương trình cho bước này khá đơn giản, khi người sử dụng đã nhập xong mật mã mới và nhấn nút F để quay về trạng thái kiểm tra mật mã. Gợi ý chương trình cho phần này như sau:



Hình 27.8: Xử lý nút F cho phần Thay Đổi Mật Mã

Chúng ta cũng khéo léo hiển thị ra mật mã mới, để người sử dụng có thể ghi nhớ mật mã mới này trước khi sử dụng lại hệ thống với mật mã mới.

### Bước 5: Hoàn thiện chương trình.



Hình 27.9: Hoàn thiện chương trình Cử mật mã

Cuối cùng, các khối lệnh được lắp vào từng nút F tương ứng để hoàn thiện tính năng kiểm tra và thay đổi mật mã của bài hướng dẫn này. Chương trình hiện tại của hệ thống được chia sẻ ở đường dẫn sau đây.

<https://app.ohstem.vn/#!/share/yolobit/2HX5AQfymdnpSm9zGKjggglZln4>

# CHƯƠNG 28

## Nhà thông minh IoT



## 1 Giới thiệu

Với sự ra đời của nền tảng kết nối vạn vật IoT, các ứng dụng thông minh có thể được nâng tầm lên một bước mới. Khi đó, dữ liệu từ các cảm biến có thể được gửi lên mạng phục vụ cho việc giám sát và điều khiển thiết bị từ xa. Trong ngữ cảnh nhà thông minh, khi áp dụng vào kết nối vạn vật, nó có thể cho người sử dụng biết được trình trạng của ngôi nhà hiện tại (nhiệt độ, ánh sáng và độ ẩm). Thêm vào đó, người sử dụng có thể chủ động bật tắt các thiết bị từ xa thông qua kết nối mạng Internet. Một ví dụ điển hình cho dịch vụ này, là bạn có thể bật trước máy nước nóng trước khi về nhà khoảng 10 phút.



Hình 28.1: Nhà thông minh với Kết nối vạn vật

Thực ra, tính năng gửi dữ liệu lên server phục vụ cho việc giám sát từ xa có thể làm được từ những công nghệ ban đầu của mạng Internet. Có rất nhiều server hỗ trợ cho dịch vụ này, mà ThingSpeak là một ví dụ điển hình. Tuy nhiên, tính năng thứ 2, là điều khiển ngược lại thiết bị từ xa, mới là điểm nổi bật của kết nối vạn vật. Tính năng đặc thù này chỉ được hỗ trợ bởi 1 số server mới hiện nay, như là Adafruit IO hay như server OhStem mà chúng tôi tự xây dựng.

Trong bài hướng dẫn này, chúng ta sẽ tập trung cho các tính năng sau đây:

- Gửi dữ liệu cảm biến lên server, phục vụ cho việc giám sát từ xa trên bảng điều khiển IoT.
- Nhận dữ liệu từ bảng điều khiển IoT và điều khiển thiết bị tương ứng, bao gồm đèn và quạt mini.

## 2 Gửi dữ liệu lên bảng điều khiển

Với khung chương trình được tổ chức tốt, tính năng này có thể bổ sung vô cùng dễ dàng và thuận lợi. Chúng ta chỉ cần thêm vào cuối khối lệnh định kì đọc cảm biến

của chương trình các câu lệnh trong nhóm MQTT, là có thể gửi được dữ liệu lên server. Bạn đọc hãy lưu ý 3 khối lệnh mới trong chương trình sau đây.



Hình 28.2: Gửi dữ liệu cảm biến lên server OhStem

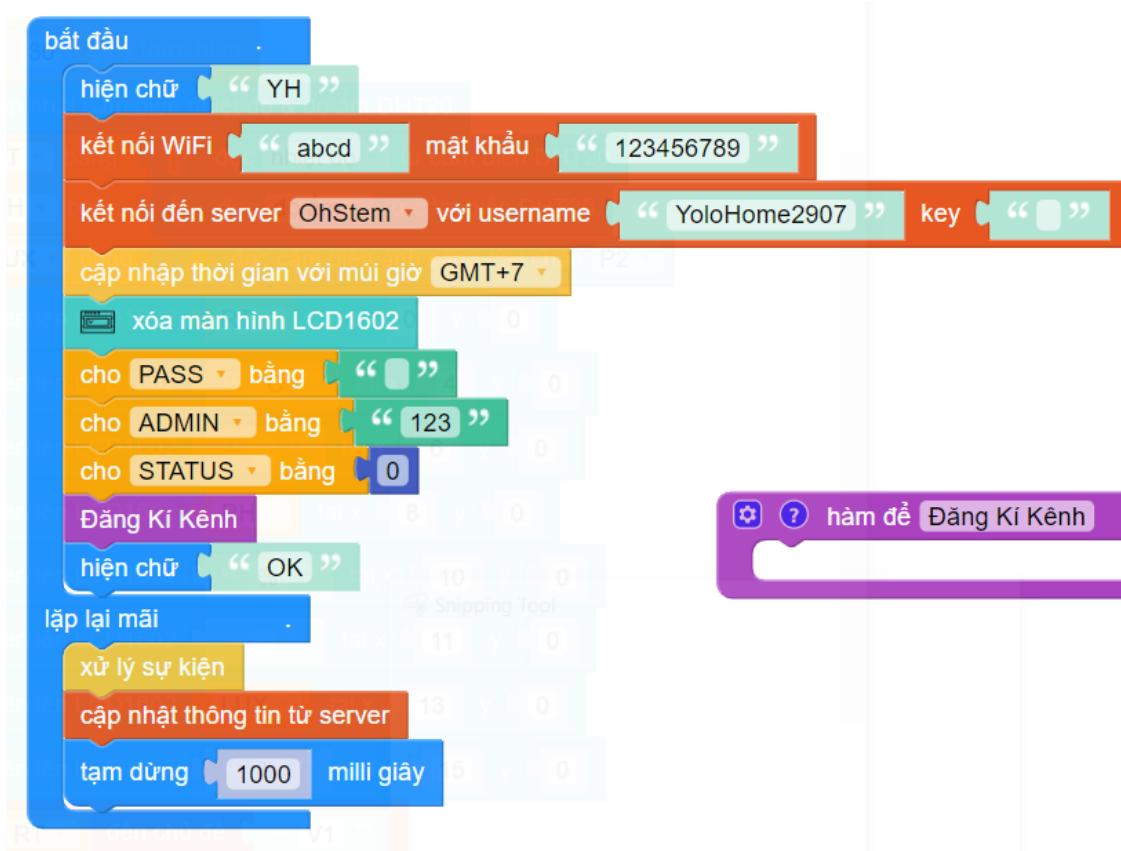
Điều quan trọng nhất là gửi dữ liệu cảm biến lên đúng kênh dữ liệu của nó, với lần lượt là Nhiệt độ (RT) cho kênh V1, RH cho kênh V2 và LUX cho kênh V3. Khi đã gửi đúng, bảng điều khiển IoT sẽ tự động cập nhật các giá trị này lên các đối tượng giao diện tương ứng mà chúng ta đã thiết kế. Một kết quả trên bảng điều khiển IoT sẽ như sau:

### 3 Nhận dữ liệu và điều khiển thiết bị

Để có thể nhận được tín hiệu điều khiển từ bảng điều khiển IoT, chúng ta phải đăng ký kênh dữ liệu tương ứng. Theo như thiết kế ở dự án này, chúng ta có 3 kênh dữ liệu dành cho việc điều khiển, lần lượt là **V10** dành cho bật tắt đèn, **V11** dành cho việc thay đổi màu của đèn và **V12** dành cho việc điều khiển tốc độ của quạt. Chi tiết quá trình hiện thực được trình như bên dưới.

**Bước 1:** Tạo hàm **Đăng Kí Kênh** và gọi trong **bắt đầu**.

Việc đăng kí kênh để nhận dữ liệu chỉ cần thực hiện một lần mà thôi. Do đó, để tiện lợi trong việc quản lý và phát triển chương trình, chúng ta sẽ tạo một chương



Hình 28.3: Tạo thêm chương trình con Đăng Kí Kênh

trình con, đặt tên là **Đăng Kí Kênh** và gọi nó trong khối **bắt đầu**. Bạn đọc hãy khéo léo đặt khối **Đăng Kí Kênh** trước việc hiện ra chữ **OK** ở cuối, như minh họa ở trên.

#### Bước 2: Đăng kí các kênh dữ liệu cần thiết.

Tiếp theo, chúng ta cần đăng kí 3 kênh dữ liệu cho việc điều khiển thiết bị. Bạn hãy lưu ý tầm vực của các câu lệnh này, nó cần phải được sắp xếp thẳng hàng trong khối chương trình **Đăng Kí Kênh**, như minh họa ở hình sau đây:



Hình 28.4: Đăng kí các kênh dữ liệu

### Bước 3: Xử lý dữ liệu trên biến thông tin.

Tiếp theo, chúng ta cần điều khiển các thiết bị tương ứng với biến thông tin nhận được cho mỗi kênh dữ liệu, như sau:

- Kênh V10: Bật tắt đèn, dữ liệu nhận được là chuỗi "1" hoặc "0".
- Kênh V11: Đổi màu của đèn, dữ liệu nhận được là một mã màu.
- Kênh V12: Điều khiển tốc độ của quạt, dữ liệu nhận được là một chuỗi kí số, cần phải đổi qua kiểu số.

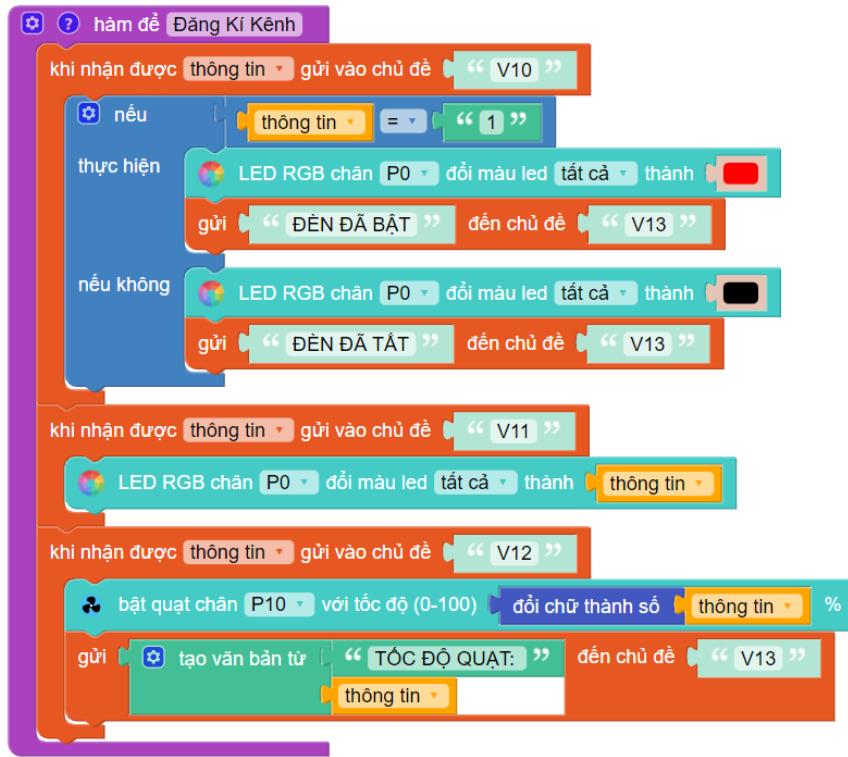
Chương trình xử lý cho 3 kênh dữ liệu được trình bày như sau:



Hình 28.5: Điều khiển thiết bị tương ứng với giá trị của thông tin nhận được

### Bước 4: Hồi đáp lại người sử dụng như một trợ lý ảo.

Chúng ta hoàn toàn có thể cho mạch Yolo:Bit hồi đáp lại tín hiệu điều khiển của chúng ta, giúp nó thực sự sinh động như một trợ lý ảo thông minh. Bằng cách gửi dữ liệu lên kênh **V13**, vốn đang được thiết kế cho phần hiển thị thông tin trên màn hình điều hành, chúng ta sẽ giúp cho dự án sinh động hơn rất nhiều. Chương trình gợi ý cho phần này được trình bày như bên dưới.



Hình 28.6: Yolo:Bit hồi đáp lại trên màn hình điều hành

Việc phản hồi từ mạch Yolo:Bit lên bảng điều khiển IoT sẽ giúp cho hệ thống sinh động hơn. Trong một số trường hợp, bạn có thể tích hợp thêm loa phát ra âm thanh, giúp cho ứng dụng gần gũi hơn với người sử dụng. Chương trình của hệ thống đến bài hướng dẫn này được chia sẻ ở đường dẫn bên dưới để bạn đọc tiện tham khảo.

<https://app.ohstem.vn/#/share/yolobit/2HX5zNDAK7IoTceSF0eQ8Ib8ZAI>

# CHƯƠNG 29

## Điều khiển bằng giọng nói



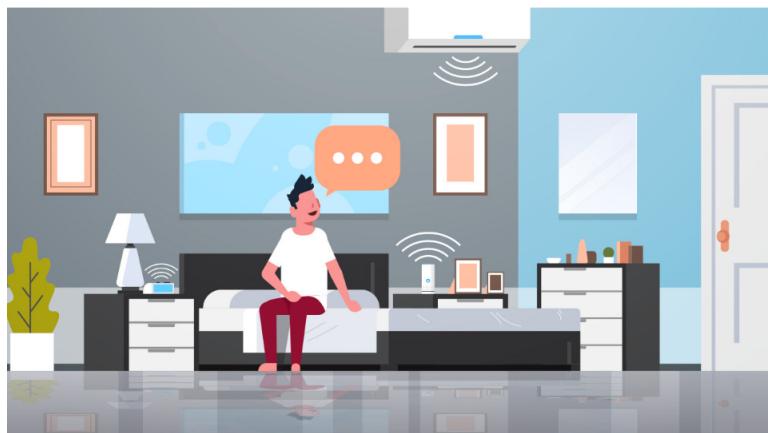
# 1 Giới thiệu

Một trong những tính năng nổi bật trong nhà thông minh hiện tại, là khả năng điều khiển các thiết bị bằng giọng nói. Nhờ sự hỗ trợ của công nghệ trí tuệ nhân tạo, mà việc nhận dạng giọng nói hiện nay đã khá chính xác cho tiếng Việt của chúng ta. Trên thị trường hiện tại, đã có nhiều sản phẩm liên quan đến trợ lý ảo trong nhà sử dụng giọng nói tiếng Việt, như Google Home hay Alexa.

Tại Việt Nam, cũng đang bắt đầu có những trợ lý ảo thuần việt đầu tiên, như VIVI trong hệ thống ô tô của VINFast hoặc loa thông minh OLLI MAIKA (<https://olli.vn/>)

- Một sản phẩm ngoài việc phát âm thanh thông thường, thì còn được ứng dụng trong điều khiển các thiết bị trong nhà chỉ bằng giọng nói mà không cần chạm tay. Công nghệ nhận dạng giọng nói này còn hay được gọi là **Speech To Text**, tạm dịch là chuyển đổi giọng nói thành văn bản.

Khả năng nhận diện giọng nói tiếng Việt mở ra một các tương tác hoàn toàn mới trong ngữ cảnh nhà thông minh, không chỉ có thể điều khiển thiết bị một cách trực tiếp bằng giọng nói, hệ thống còn có thể hiểu được ngữ cảnh và đưa ra những điều khiển thông minh, chẳng hạn như "**trời nóng quá**" sẽ được hồi đáp lại bằng hành vi bật quạt hoặc máy lạnh từ ngôi nhà thông minh.



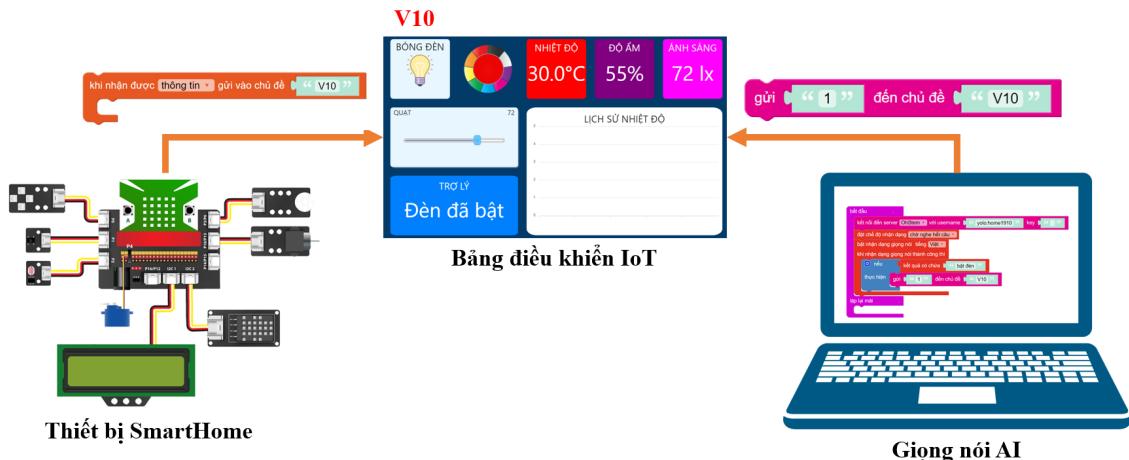
Hình 29.1: Công nghệ nhận dạng giọng nói trong nhà thông minh

Trong bài hướng dẫn này, chúng ta sẽ tích hợp việc điều khiển thiết bị đang có trong dự án nhà thông minh bằng giọng nói. Các mục tiêu cụ thể của bài hướng dẫn này như sau:

- Xây dựng kiến trúc điều khiển giọng nói cho hệ thống nhà thông minh
- Hiện thực chương trình nhận dạng giọng nói trên máy tính
- Tích hợp chương trình giọng nói để điều khiển thiết bị

## 2 Kiến trúc điều khiển AIoT

Hiện tại, các thiết bị trong nhà thông minh đang được đồng bộ với bảng điều khiển IoT. Chẳng hạn như với một bóng đèn, nó sẽ được liên kết với kênh dữ liệu V10 trên bảng điều khiển IoT, như minh họa ở hình bên dưới.



Hình 29.2: Kiến trúc hệ thống khi kết hợp với AI

Điều lưu ý quan trọng là, mỗi khi đèn được bật, dữ liệu "1" từ bảng điều khiển sẽ được gửi xuống thiết bị. Trường hợp đèn tắt, dữ liệu "0" sẽ được gửi đi. Do đó, chương trình AI nhận diện giọng nói, nó chỉ cần gửi đúng dữ liệu (0 hoặc 1 cho bóng đèn) và đúng kênh (chẳng hạn là V10), là thiết bị trong hệ thống sẽ được điều khiển theo đúng nguyên lý của nó.

Theo nguyên lý này, chương trình AI sẽ hoàn toàn không kết nối gì với thiết bị nữa, mà nó chỉ quan tâm đến việc gửi dữ liệu điều khiển lên bảng điều khiển IoT mà thôi. Điều này sẽ đơn giản hóa đi rất nhiều các kết nối Bluetooth, vôn kén thiết bị hơn và thường phụ thuộc và hệ điều hành. Trong khi đó, WIFI lại là công nghệ phổ dụng hiện nay và rất dễ dàng để gửi dữ liệu đến OhStem server. Với kiến trúc mới này, khi AI và IoT có thể được tính hợp chung trong một dự án, người ta thường gọi là kiến trúc AIoT.

## 3 Chương trình nhận diện giọng nói

Trong phần này, chúng ta sẽ từng bước xây dựng chương trình nhận diện giọng nói, bắt các từ khóa điều khiển chính và **gửi dữ liệu lên kênh dữ liệu phù hợp**, phục vụ cho việc điều khiển thiết bị trong dự án nhà thông minh. Chương trình trong phần hướng dẫn tiếp theo được chọn từ màn hình chính của OhStem, vào phần **Lập trình AI**. Từng bước để hiện thực chương trình này được trình bày như bên dưới.

**Bước 1:** Kết nối với bảng điều khiển IoT thông qua username.



Hình 29.3: Kết nối với màn hình điều khiển IoT

Thông tin về **username** dùng trong chương trình này phải trùng khớp hoàn toàn với username của bảng điều khiển, thì nó mới có thể gửi đúng dữ liệu lên server và phục vụ cho việc điều khiển thiết bị nhà thông minh bằng giọng nói.

**Bước 2:** Bật tính năng nhận diện giọng nói tiếng Việt.



Hình 29.4: Bật nhận dạng giọng nói tiếng Việt

Trong bước thứ 2 này, bạn lưu ý chọn chế độ nhận dạng cho phù hợp với ứng dụng của mình. Trong môi trường ít tiếng ồn, thì việc lựa chọn **chờ nghe hết câu** sẽ hợp lý hơn. Tuy nhiên, nếu như môi trường xung quanh có quá nhiều tiếng ồn, sự kiện này sẽ khó có thể xảy ra. Trong trường hợp này, lựa chọn chế độ **sau mỗi 2 giây** sẽ hợp lý hơn.

**Lưu ý:** Câu lệnh đặt chế độ cần phải đặt sau câu lệnh bật nhận dạng giọng nói.

**Bước 3:** Bắt từ khóa và gửi dữ liệu lên server.

Trong bước này, bạn bắt đầu hiện thực việc bắt từ khóa và gửi dữ liệu cho đúng. Dữ liệu phải gửi đi có 2 thông tin quan trọng:

- Kênh dữ liệu: Để thiết bị có thể nhận được tín hiệu, chương trình AI phải gửi lên kênh dữ liệu mà thiết bị đang đăng kí với server.
- Dữ liệu để điều khiển: Tùy thiết bị đang lập trình, dữ liệu này có thể mang ý nghĩa là bật (thường là chuỗi "1") hoặc tốc độ (một con số cho máy quạt).

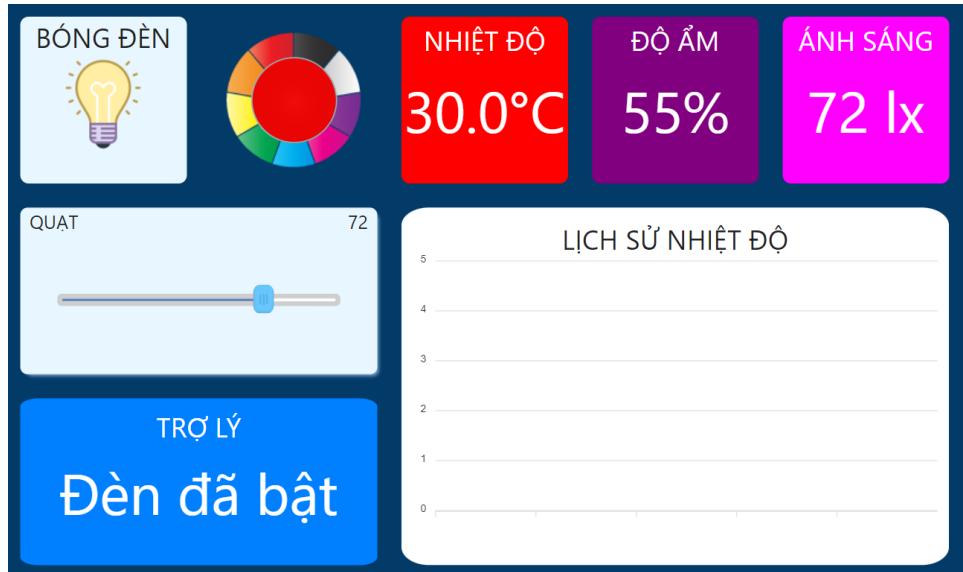
Trong chương trình bên trên, chúng ta đang thử bắt từ khóa "**bật đèn**" để điều khiển thiết bị. Theo thiết kế của dự án, đèn RGB sẽ đăng kí vào kênh **V10**, và dữ



Hình 29.5: Bắt từ khóa và gửi dữ liệu điều khiển lên server

liệu cho nó sẽ là "1" hoặc "0", tương ứng với bật và tắt.

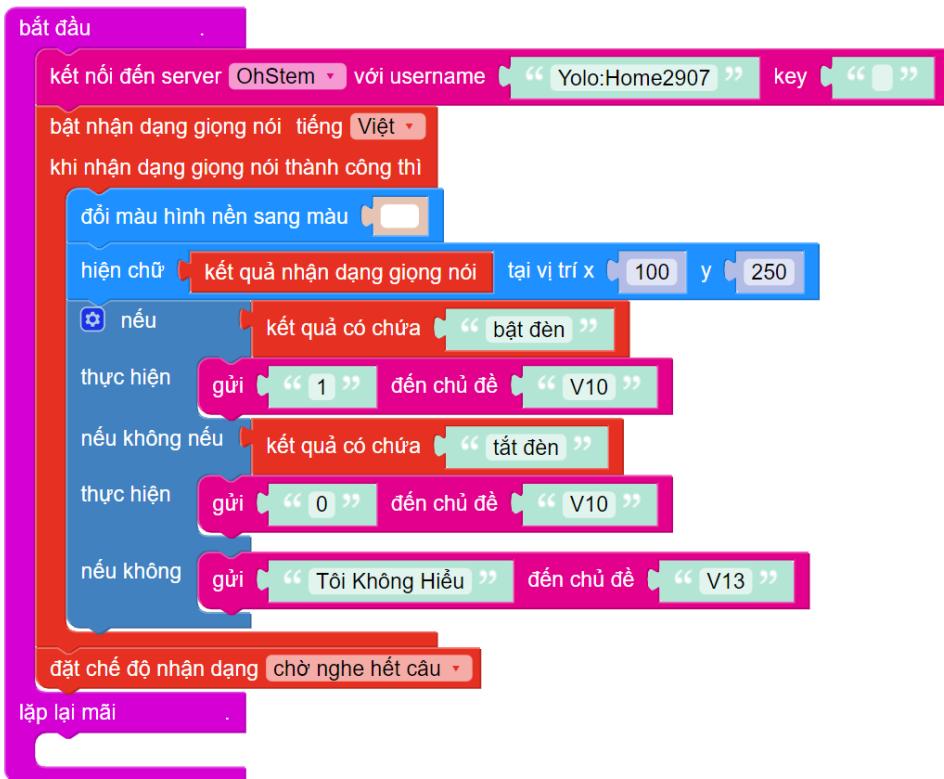
Điều thú vị trong việc tích hợp này, là khi bạn thực sự nói "**bật đèn**", trạng thái của nút nhấn dành cho đèn và trợ lý ảo của chúng ta cũng tự động cập nhật theo, như minh họa ở hình bên dưới:



Hình 29.6: Thông tin được cập nhật trên bảng điều khiển

**Bước 4:** Mở rộng và phát triển chương trình.

Bạn đọc có thể tự phát triển chương trình ở bước này, để bắt thêm nhiều từ khóa cho việc điều khiển thiết bị trong nhà thông minh. Tuy nhiên, lưu ý ở đây là bạn hãy sử dụng **câu lệnh nếu nhiều điều kiện**, với phần **nếu không** ở dưới cùng, như minh họa dưới đây.



Hình 29.7: Mở rộng tính năng điều khiển bằng giọng nói

Với việc có thêm khôi lệnh **nếu không**, với những câu điều khiển nằm ngoài những tính năng bạn đã hiện thực, bạn có thể gửi 1 thông báo ngắn lên kênh **V13** để thông báo với người sử dụng. Những tính năng này tuy đơn giản, nhưng sẽ góp phần lớn trong việc cải thiện trải nghiệm của người sử dụng. Đây cũng là một trong những tiêu chí mà chúng tôi hướng tới trong việc làm mới ứng dụng nhà thông minh, vốn đã rất phổ biến trong cộng đồng STEM.

# CHƯƠNG 30

## Nhận dạng khuôn mặt FaceAI



# 1 Giới thiệu

Với sự phát triển của khoa học công nghệ, các điện thoại thông minh đã tích hợp tính năng nhận diện khuôn mặt như là một dịch vụ, dựa trên nhiều camera đang có. Công nghệ ngày gọi là FaceID, khi nó có thể dựng lên hình ảnh 3D của một khuôn mặt và dùng nó cho việc xác thực. FaceID đang dần thay thế cảm biến vân tay, một công nghệ xác thực khác dựa vào sinh trắc học của một người.



Hình 30.1: Công nghệ nhận dạng định danh FaceID

Mặc dù hiện tại, hệ sinh thái OhStem chưa hỗ trợ chính xác công nghệ FaceID trong môi trường lập trình, chúng ta vẫn có thể sử dụng công cụ trí tuệ nhân tạo hiện có để tự xây dựng cho mình một hệ thống đơn giản để nhận diện khuôn mặt. Chúng tôi định gọi tính năng này là FaceAI, với công nghệ trí tuệ nhân tạo dựa trên học máy của Google.

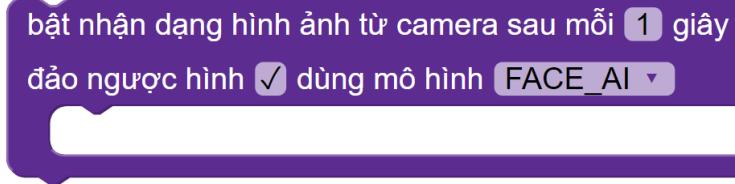
Việc xác thực rõ ràng là phải ổn định và chính xác. Do đó, chúng tôi đặt ra yêu cầu cho bài toán xác thực bằng khuôn mặt là hệ thống phải nhận diện ra 1 người ổn định trong 3 lần liên tiếp, mỗi lần cách nhau một giây, thì mới xác nhận là một người đã được huấn luyện.Thêm nữa, xác suất nhận dạng đúng một người phải cao hơn 70% thì mới bắt đầu xử lý.

Tính năng FaceAI trong bài này sẽ tiếp tục tích hợp vào hệ thống nhà thông minh. Như vậy, cả 2 công nghệ về trí tuệ nhân tạo sẽ được chạy song song trong dự án Yolo:Home, bao gồm điều khiển bằng giọng nói và nhận dạng khuôn mặt. Server IoT sẽ được dùng là cầu nối trung gian để thông báo cho mạch Yolo:Bit biết rằng có một người đang truy cập vào nhà thông minh và đưa ra những xử lý thích hợp. Các nội dung hướng dẫn trong bài này như sau:

- Sử dụng khôi nhận diện AI với ràng buộc thời gian
- Hiện thực chương trình nhận diện một khuôn mặt FaceAI
- Tích hợp điều khiển với mạch Yolo:Bit

## 2 Khởi nhận diện AI và thời gian

Với yêu cầu của việc nhận diện khuôn mặt ổn định trong 3 giây, chúng ta sẽ cần một khôi lệnh mới phục vụ cho tính năng này, như trình bày ở hình bên dưới:



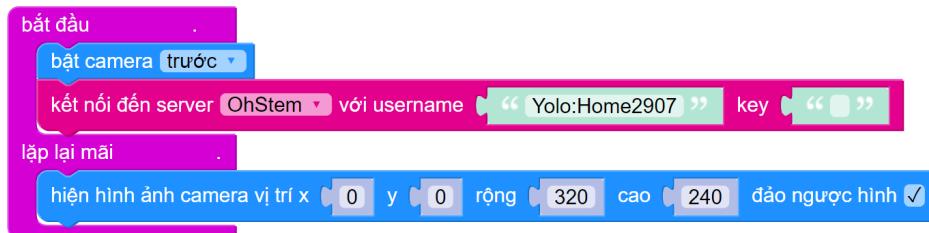
Hình 30.2: Khôi lệnh nhận diện AI với thời gian

Khôi lệnh này nằm ở cuối cùng trong nhóm **M-LEARNING**. Về ý nghĩa, nó sẽ thực hiện việc nhận diện hình ảnh theo chu kỳ, là con số được chỉ định ở cuối câu lệnh. Mặc định, chu kỳ này là một giây.

Như vậy, khi áp dụng vào yêu cầu của tính năng FaceAI, chúng ta sẽ cần khai báo thêm biến số để cộng dồn cho nó. Khi giá trị của biến số bằng 3, là lúc chúng ta xác nhận được khuôn mặt đã được huấn luyện. Từng bước hiện thức tính năng FaceAI được trình bày ở phần bên dưới.

## 3 Hiện thực FaceAI

**Bước 1:** Khởi tạo chương trình.



Hình 30.3: Phân loại trái cây khi thu hoạch

Trong bước này, chúng ta sẽ bật camera và quan trọng là kết nối với server OhStem bằng tài khoản mà chúng ta đã sử dụng cho dự án Yolo:Home.

**Bước 2:** Xây dựng khôi lệnh AI với thông tin về thời gian.

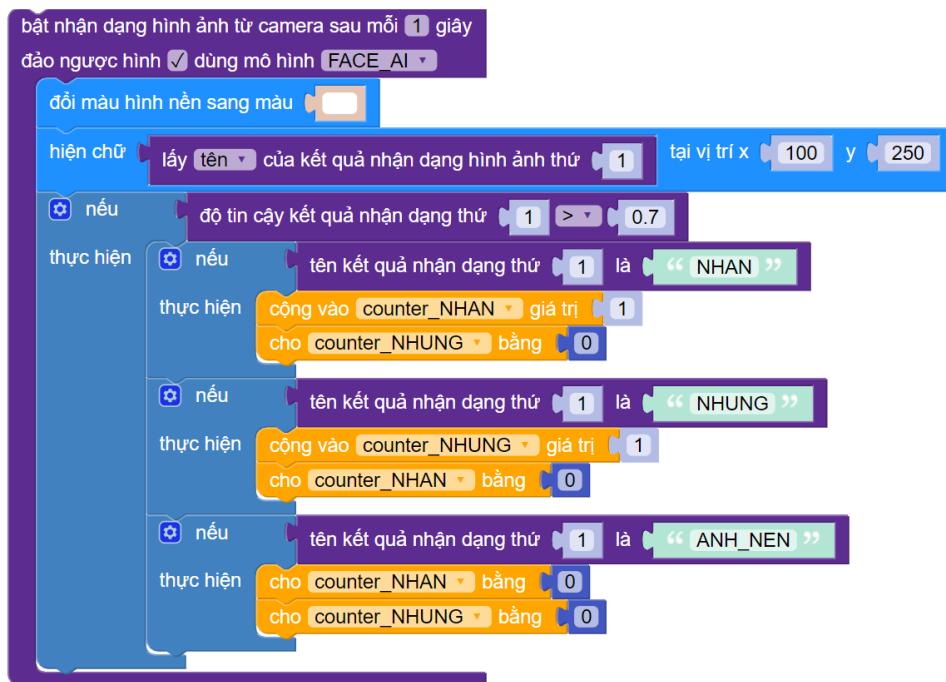
Trước khi ghép vào phần **bắt đầu** của chương trình, chúng ta sẽ xây dựng các tác vụ liên quan đến AI trước. Đầu tiên là khôi nhận dạng AI và các câu lệnh cơ bản để hiển thị kết quả ra màn hình, như sau:

**Bước 3:** Xử lý các biến số tương ứng với các kiến thức nhận dạng.



Hình 30.4: Câu lệnh nhận dạng AI với thời gian

Nhiều câu lệnh **nếu ... thực hiện** sẽ được sử dụng để xử lý cho các kiến thức nhận dạng được. Với mỗi kiến thức, chúng ta sẽ khai báo thêm 1 biến đếm tương ứng cho việc kiểm tra rằng nó có được nhận dạng ổn định trong liên tiếp 3 giây hay không.



Hình 30.5: Xử lý trên biến số với từng kiến thức nhận dạng được

Trong chương trình trên, chúng ta có 2 kiến thức về việc nhận dạng khuôn mặt, là "NHAN" và "NHUNG". Mỗi khi nhận được kiến thức nào, biến đếm cho nó sẽ được cộng lên 1 trong khi các biến đếm còn lại sẽ được chỉnh về 0. Và hiển nhiên, khi không có khuôn mặt nào (kiến thức "ANH\_NEN" được nhận) tất cả các biến được gán về 0.

Câu lệnh **nếu ... thực hiện** bao trùm tất cả các kiến thức đảm bảo rằng xác suất nhận diện mỗi kiến thức là cao, trên 70%.

**Bước 4:** Hiện thực tính ổn định của việc nhận dạng.

Với kiến thức đầu tiên, là "NHAN", chúng ta sẽ kiểm tra biến "counter\_NHAN". Khi giá trị của nó là 3, tức là trong 3 giây, chúng ta đã nhận dạng kiến thức này một

cách ổn định (3 lần). Các xử lý sẽ được thực hiện trong một câu lệnh **nếu ... thực hiện**, như sau:



Hình 30.6: Phân loại trái cây khi thu hoạch

Bạn đọc hãy lưu ý một số điểm quan trọng trong việc xử lý ở trên:

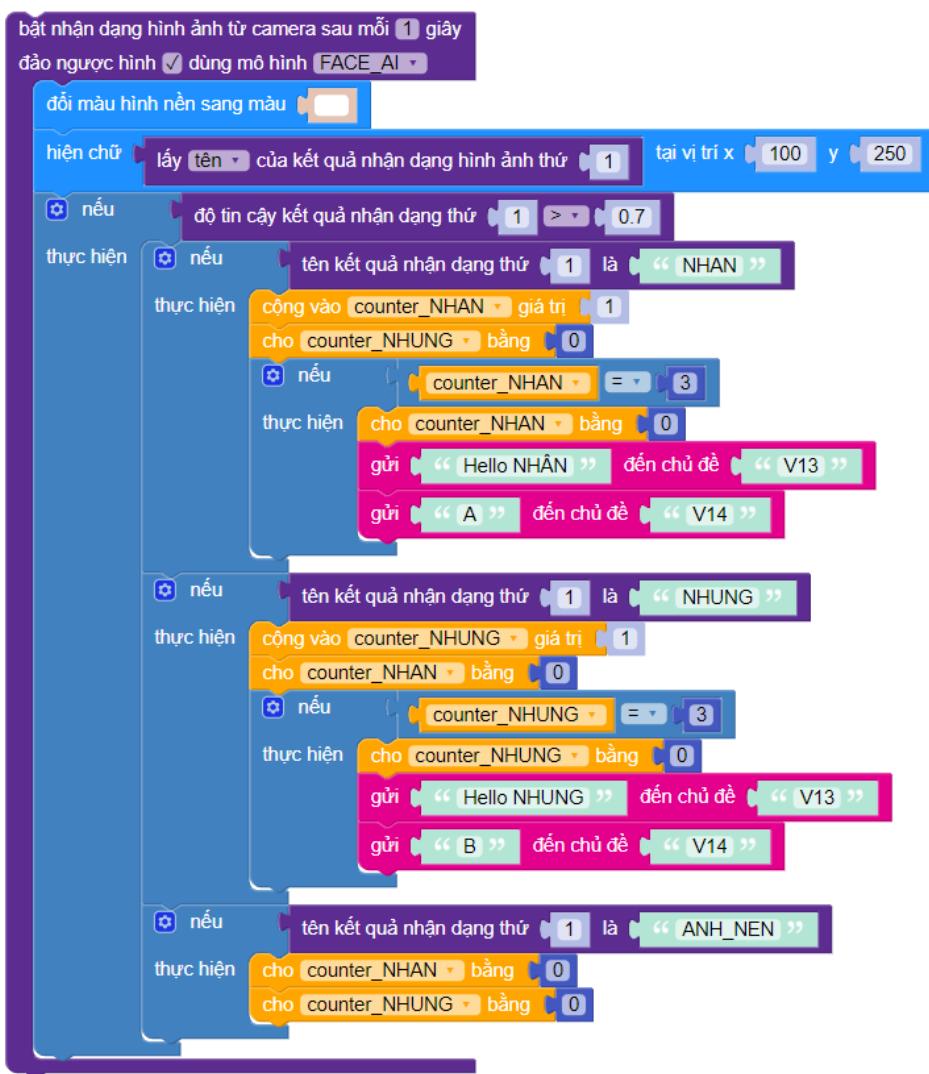
- Biến đếm phải được chỉnh về 0 cho lần nhận dạng ổn định tiếp theo.
- Một thông tin sẽ được gửi lệnh kênh dữ liệu **V13**, đây là kênh dành cho trợ lý ảo của màn hình điều khiển IoT.
- Tuy nhiên, một thông tin đơn giản hơn sẽ được gửi lên **V14** để mạch Yolo:Bit có thể xử lý đơn giản. Ở đây, khi nhận được chuỗi "A", khuôn mặt được đánh nhãn "NHAN" đã được xác thực.

Mặc dù kênh dữ liệu **V14** không hề được liên kết với đối tượng giao diện nào trên màn hình điều hành IoT, nó vẫn được lưu lại trên OhStem server. Mạch Yolo:Bit chỉ cần đăng ký kênh này là có thể nhận được các thông tin cần thiết cho việc xử lý FaceAI, như mở cửa nếu nhận ra được người chủ của ngôi nhà chẳng hạn.

Nếu như không có kênh V14, chúng ta vẫn có thể đăng ký kênh V13 để xử lý. Tuy nhiên trường hợp này, dữ liệu phải so sánh sẽ nhiều hơn và không thuận tiện cho thao tác trên một mạch điện nhỏ như Yolo:Bit. Kinh nghiệm của chúng tôi là hãy là cho thông tin tới mạch Yolo:Bit được đơn giản và gọn nhẹ nhất. Điều này sẽ giúp cho hệ thống hoạt động tin cậy hơn.

## Bước 5: Hoàn thiện khối xử lý AI.

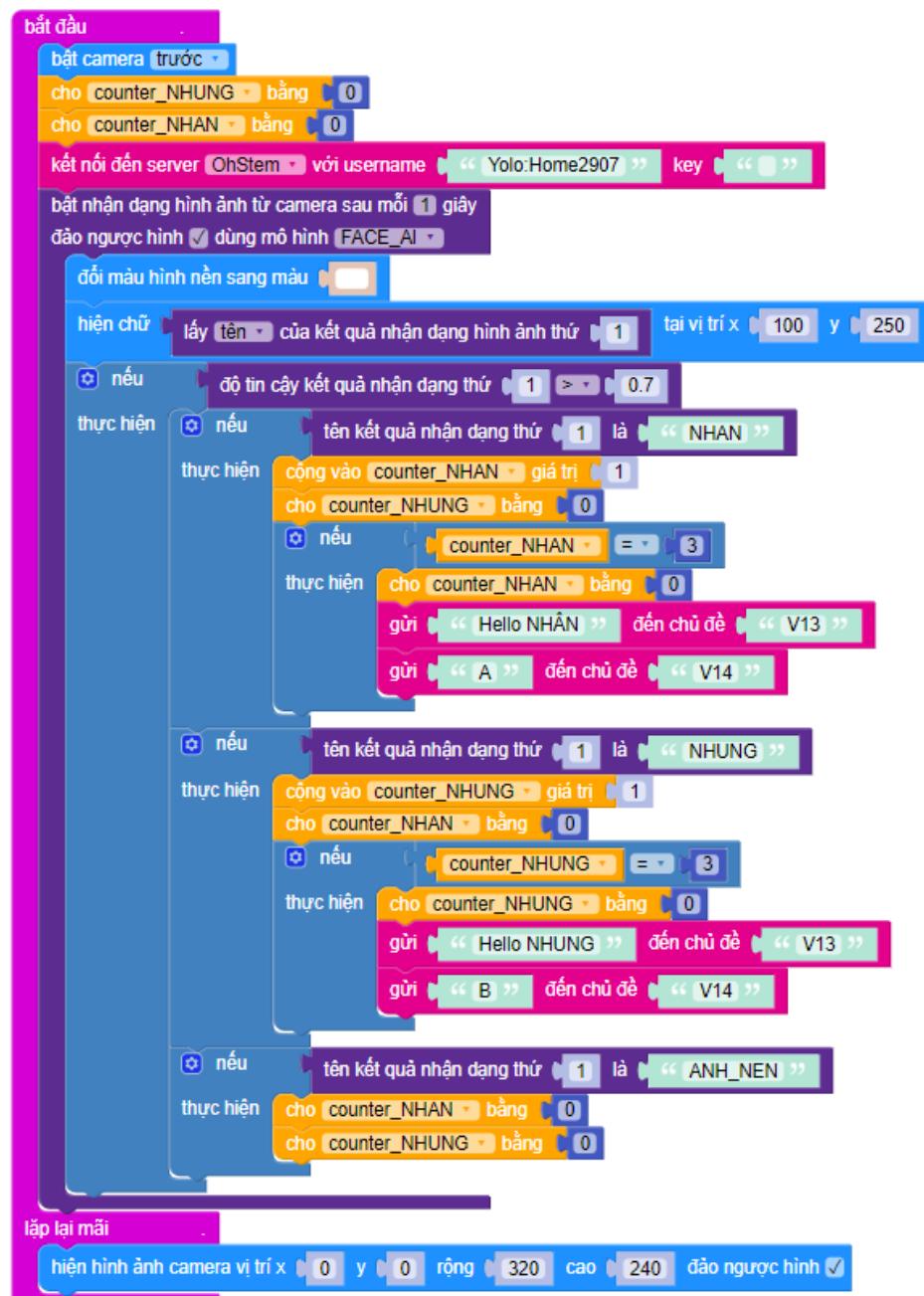
Tương tự như phần xử lý cho kiến thức đầu tiên, các kiến thức còn lại sẽ được hiện thực tương tự. Khi hệ thống có nhiều khuôn mặt cần nhận dạng, bạn cần hiện thực thêm nhiều câu lệnh nếu cũng như khai báo thêm biến số cho từng kiến thức để đếm số lần nhận dạng liên tục. Kết quả của bước này sẽ như minh họa ở hình ảnh bên dưới.



Hình 30.7: Hoàn thiện khối xử lý AI

## Bước 6: Hoàn thiện chương trình nhận dạng AI.

Cuối cùng, chúng ta sẽ ghép khôi nhận dạng AI này vào luồng xử lý của chương trình. Chúng ta cũng không quên khởi tạo giá trị cho các biến đếm.



Hình 30.8: Hoàn thiện chương trình FaceAI

## 4 Tích hợp xử lý trên Yolo:Bit

Trong trường hợp muốn xử lý thêm mạch Yolo:Bit, chẳng hạn như mở cửa khi nhận dạng được đúng người đã huấn luyện, bạn chỉ đơn giản là đăng ký thêm kênh **V14** cho mạch Yolo:Bit trong khôi hàm **Đăng Kí Kênh** mà chúng ta đã tạo ra ở các

bài trước. Trong ví dụ của chúng tôi, chuỗi kí tự "A" và "B" sẽ được gửi tới mạch Yolo:Bit.



Hình 30.9: Chương trình nhận lệnh FaceAI trên Yolo:Bit

Bạn đọc có thể chủ động phát triển thêm nhiều tính năng mới cho việc nhận dạng ra một khuôn mặt. Chẳng hạn như hệ thống sẽ chủ động mở cửa nếu nhận được "A". Tuy nhiên, khi nhận được "B" nó sẽ gửi một tin nhắn đến người A để mở cửa chẳng hạn. Chương trình cho phần tham khảo của bạn đọc được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2HCcI1CXE9DAqBe0wDjWr3MChw1>

## 5 Kết luận và các hướng mở rộng

Qua chuỗi bài hướng dẫn trong tài liệu này, chúng tôi hy vọng có thể xây dựng cho bạn đọc một cách tiếp cận hoàn chỉnh cho một dự án thông minh. Việc tổ chức chương trình là điều vô cùng cần thiết với các dự án cần độ tích hợp cao, từ điều khiển tự động, kết nối vạn vật cho đến trí tuệ nhân tạo. Dựa vào dự án trong giáo trình này, chúng tôi hy vọng bạn đọc có thể tìm thấy nhiều sự tương đồng của nó với nhiều dự án khác trong tương lai với sự tích hợp của tự động hóa, kết nối vạn vật và trí tuệ nhân tạo.

Điều cốt lõi trong ngôn ngữ lập trình ở giáo trình này, là các chức năng được hiện thực trong các khối lệnh sự kiện. Tính cấu trúc của chương trình (các câu lệnh được thực hiện từ trên xuống dưới) không còn là điều bắt buộc đối với các ngôn ngữ lập trình cấp cao. Trong phần **lặp mãi mãi**, rõ ràng chúng ta chỉ có 1 số câu lệnh mang tính chất duy trì cấu hình của hệ thống. Tất cả các tính năng của hệ thống được hiện thực một cách độc lập, và được thực hiện song song trong các khối lệnh sự kiện. Đây là đặc điểm đặc trưng của một khái niệm mới trong ngôn ngữ lập trình cấp cao, đó là lập trình hướng đối tượng.

Trong giáo trình này, một cách gián tiếp, chúng tôi cho các bạn tiếp cận với chương trình của một **máy trạng thái** khi thực hiện tính năng đổi mật khẩu ở mạch Yolo:Bit hay như nhận dạng FaceAI cho phần trí tuệ nhân tạo. Trong một chương trình hướng theo máy trạng thái, việc khai báo biến là bước đầu tiên. Giá trị của biến đại diện cho trạng thái của hệ thống. Trong mỗi khối điều kiện nếu, các chức năng tương ứng của một trạng thái sẽ được hiện thực. Cuối cùng, điều kiện để chuyển sang một trạng thái mới sẽ được hiện thực bằng một câu lệnh nếu khác.

# Kết nối cộng đồng

OhStem hiểu việc chỉ nghiên cứu một cuốn sách sẽ không đủ để bạn hiểu hết về các chủ đề này, hơn nữa, có thể bạn sẽ gặp nhiều khó khăn trong thực hành. Do đó, OhStem đã tạo một **cộng đồng các giáo viên dạy về STEM**, để trao đổi và hỗ trợ nhau trong quá trình thực hành.

Trong group cộng đồng này sẽ có đội ngũ kỹ thuật cũng như các thầy cô vấn chuyên môn nhiều kinh nghiệm, cùng đóng đảo các giáo viên STEM để hỗ trợ bạn.

Bạn có thể tham gia vào group tại:

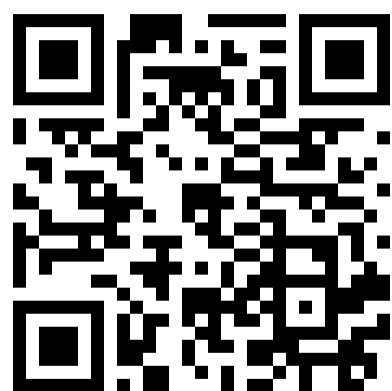
## 1. Cộng đồng trên Facebook

Link: <https://www.facebook.com/groups/dayvahocsteam>



## 2. Cộng đồng trên Zalo

Link: <https://zalo.me/g/lnlumg837>



# Thông tin liên hệ

Bạn có thể liên hệ với OhStem qua:

- Fanpage: <https://www.facebook.com/ohstem.aitt>
- Hotline: 08.6666.8168
- Email: contact@ohstem.vn