

YOLOFARM

NÔNG NGHIỆP CÔNG NGHỆ

Nông nghiệp dựa trên AI và IoT



Mục lục

I Các Thao Tác Cơ Bản Trên Yolo:Bit	13
Chương 1. Chương trình đầu tiên trên Yolo:Bit	15
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
Chương 2. Ngoại vi cơ bản trên Yolo:Bit	27
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
Chương 3. Đồng hồ thông minh Internet	35
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
II Kết Nối Mở Rộng Cho Yolo:Bit	43
Chương 4. Mạch Mở Rộng - Đèn RGB	45
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
Chương 5. Nhiệt Độ - Độ Ẩm DHT20		53
1	Giới thiệu	54
2	Kết nối với DHT20	54
3	Lập trình lấy dữ liệu từ DHT20	55
Chương 6. Màn Hình LCD		59
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
Chương 7. Máy Bơm Chìm		67
1	Giới thiệu	68
2	Kết nối với Yolo:Bit	69
3	Lập trình điều khiển máy bơm	70
Chương 8. Công Tắc Relay		71
1	Giới thiệu	72
2	Nguyên lý hoạt động của Relay	73
3	Kết nối Relay với Yolo:Bit	73
Chương 9. Cảm Biến Độ Ẩm Đất		77
1	Giới thiệu	78
2	Kết nối với Yolo:Bit	79
3	Lập trình đọc độ ẩm từ cảm biến	79
4	Chuyển đổi giá trị cho cảm biến	80
Chương 10. Cảm Biến Ánh Sáng		83
1	Giới thiệu	84
2	Công thức GDD	85
3	Hiện thực trên Yolo:Bit	85
III Kết Nối Vạn Vật với OhStem		89
Chương 11. Bảng Điều Khiển IoT		91
1	Giới thiệu	92
2	Chia sẻ dữ liệu trên OhStem server	93
3	Thiết kế bảng điều khiển	93

Chương 12. Nhận dữ liệu trên Yolo:Bit	99
1 Giới thiệu	100
2 Thêm khối mở rộng MQTT	101
3 Lập trình trên Yolo:Bit	102
3.1 Kết nối vào mạng WiFi	103
3.2 Kết nối với OhStem server	103
3.3 Đăng ký kênh dữ liệu	104
3.4 Liên kết định kỳ với Server	105
4 Xử lý dữ liệu nhận được từ server	105
5 Mở rộng chương trình	106
Chương 13. Hiển thị thông tin IoT	109
1 Giới thiệu	110
2 Giao diện Thông tin và Đồ thị	110
2.1 Cấu hình cho Thông tin	111
2.2 Cấu hình cho Đồ thị	112
3 Hoàn thiện bảng giám sát	113
Chương 14. Gửi dữ liệu lên OhStem server	115
1 Giới thiệu	116
2 Kết nối với OhStem server	117
3 Thư viện lập trình Sự kiện	118
4 Gửi dữ liệu lên server	120
Chương 15. Giao diện nâng cao	121
1 Giới thiệu	122
2 Tự kiểm thử giao diện	122
3 Cấu hình cho Thanh trượt	124
4 Giao diện Đồng hồ đo	125
IV Trí Tuệ Nhân Tạo	129
Chương 16. Nhận diện giọng nói	131
1 Giới thiệu	132
2 Kiến trúc ứng dụng AI	132
3 Môi trường lập trình AI	133
4 Lập trình nhận diện giọng nói	134
Chương 17. Nhận và xử lý lệnh AI	139
1 Giới thiệu	140
2 Xây dựng chương trình cho Yolo:Bit	141
3 Lưu chương trình vào thiết bị	144
4 Kết nối với trí tuệ nhân tạo	145
Chương 18. Huấn luyện trí tuệ nhân tạo	147
1 Giới thiệu	148
2 Trang chủ của trí tuệ nhân tạo	148
3 Huấn luyện hệ thống	149
4 Hiện thực dự án AI	151

Chương 19. Kết hợp công nghệ AI	155
1 Giới thiệu	156
2 Chuyển đổi giọng nói - văn bản	156
2.1 Môi trường doanh nghiệp	157
2.2 Nhà thông minh và Robot	157
2.3 Trong giáo dục	158
3 Học máy với Google	158
4 Kết hợp hai công nghệ AI	159
Chương 20. Khung chương trình cho AI và IoT	161
1 Giới thiệu	162
2 Thư viện cho ứng dụng AIoT	162
3 Khung chương trình IoT	163
4 Tích hợp tính năng AI	165
V Hiện Thực Dự Án Yolo:Farm	169
Chương 21. Tổng quan dự án	171
1 Giới thiệu	172
2 Giám sát môi trường	172
3 Tưới tiêu thông minh	173
4 Phân tích dữ liệu	174
5 Phát hiện bást thường ở cây trồng	174
6 Phân loại trái cây thu hoạch	175
Chương 22. Khởi động dự án	177
1 Giới thiệu	178
2 Kết nối phần cứng	178
3 Thiết kế bảng điều khiển IoT	179
4 Khung chương trình AIoT	181
Chương 23. Giám sát môi trường	183
1 Giới thiệu	184
2 Hiển thị thông tin trên LCD	184
3 Hiện thực chương trình	185
Chương 24. Tưới tiêu thông minh	189
1 Giới thiệu	190
2 Tưới tự động	191
3 Tưới theo giờ	191
4 Tưới thủ công	193
Chương 25. Phân tích dữ liệu	195
1 Giới thiệu	196
2 Cập nhật GDD	196
3 Phân tích dữ liệu	197

Chương 26. Phát hiện bất thường	201
1 Giới thiệu	202
2 Huấn luyện AI	203
3 Chương trình AI	203
4 Chương trình trên Yolo:Bit	205
Chương 27. Phân loại thu hoạch	207
1 Giới thiệu	208
2 Tổng quan hệ thống	209
3 Huấn luyện hệ thống AI	209
4 Hiện thực hệ thống	210
4.1 Xử lý AI và gửi lệnh	210
4.2 Nhận lệnh AI và điều khiển	211
5 Kết luận và các hướng mở rộng	212

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 nông nghiệp, vốn là lĩnh vực sản xuất quan trọng tại Việt Nam. Bằng việc áp dụng các cảm biến đặc thù, người nông dân có thể giám sát thường xuyên về điều kiện nuôi trồng, từ đó đưa ra các quyết định xử lý một cách kịp thời và hợp lý. Chúng tôi tin rằng dữ liệu, cùng với kiến thức và trực giác của người nông dân về trang trại của họ, có thể giúp tăng năng suất trang trại và cũng giúp giảm chi phí. Cộng với công nghệ về trí tuệ nhân tạo, những yêu cầu phức tạp trong nông nghiệp như phát hiện bất thường ở cây trồng, có thể được triển khai dễ dàng.

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



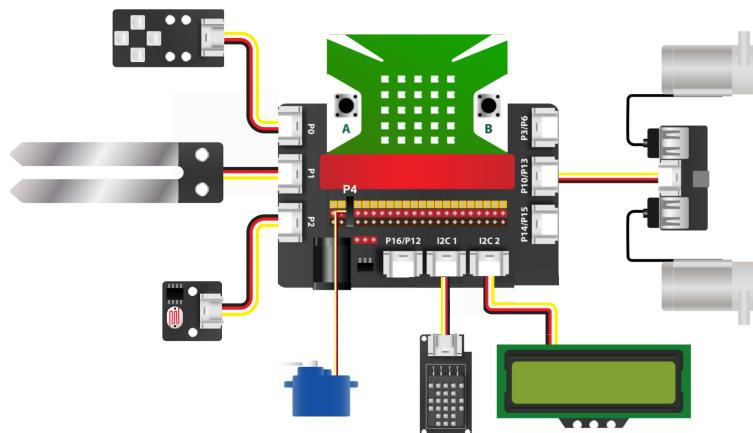
Hình 1: Mạch lập trình Yolo:Bit sử dụng trong giáo trình

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:Farm, 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 đất, nhiệt độ không khí) và thiết bị điều khiển (máy bơm, 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 nông nghiệp, phần này có thể dễ dàng áp dụng cho các dự án khác trong tương lai, chẳng hạn như nhà thông minh (Smart Home).



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

Phần 3: Kết Nối Vạn Vật với OhStem



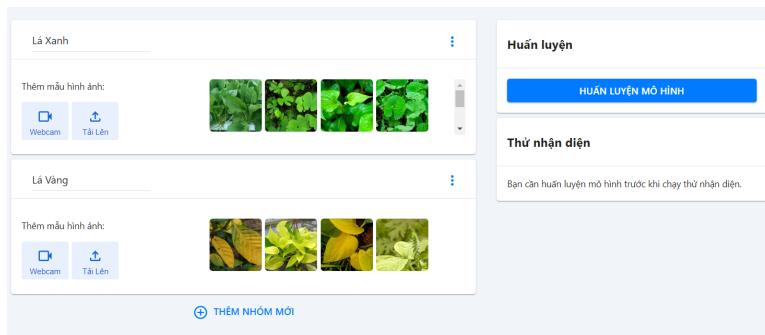
Hình 3: OhStem - Server kết nối vạn vật

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.

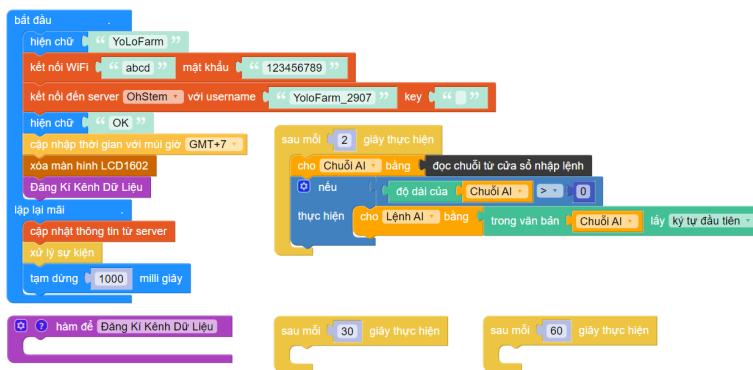
Phần 4: Trí Tuệ Nhân Tạo

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 phát hiện những bất thường ở cây trồng hoặc phân loại hoa quả sau khi thu hoạch. Sự linh động của trí tuệ nhân tạo giúp nó dễ dàng ứng dụng vào nhiều dự án khác nhau, chẳng hạn như phân loại rác thải chẳng hạn.



Hình 4: Huấn luyện trí tuệ nhân tạo

Phần 5: Hiện Thực Dự Án Yolo:Farm

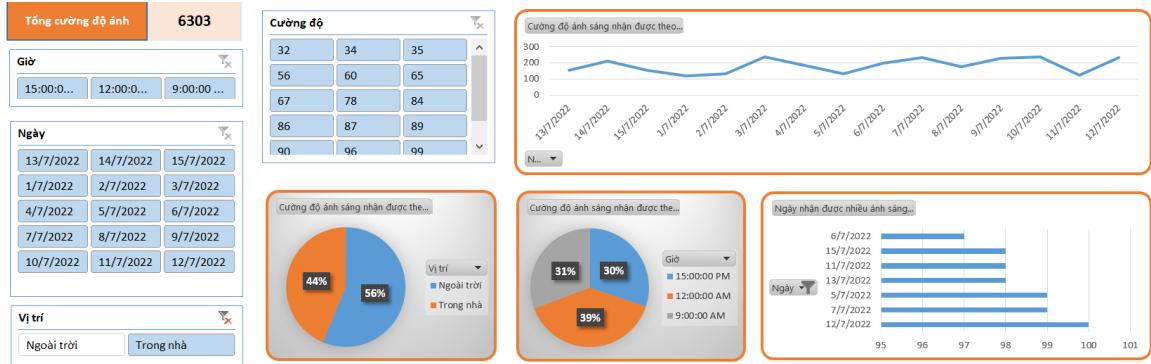


Hình 5: Tổ chức chương trình cho dự án

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 tưới tiêu theo giờ, đếm ngày tăng trưởng của cây hay

phân loại hoa quả sau khi thu hoạch.

Bên cạnh đó, chúng tôi cũng giới thiệu những khái niệm cơ bản nhất về khoa học dữ liệu. Cụ thể, các giá trị cảm biến thu thập được sẽ được trích xuất và phân tích bằng phần mềm Microsoft Excel, một công cụ vốn đã quen thuộc với đa số bạn đọc.



Hình 6: Phân tích dữ liệu với Microsoft Excel

Với việc chia giáo trình thành 5 phần riêng biệt, chúng tôi hy vọng có thể giúp các thầy cô phân bổ tải bài giảng hợp lý với học sinh khối THCS. Với học sinh lớp 6, 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.

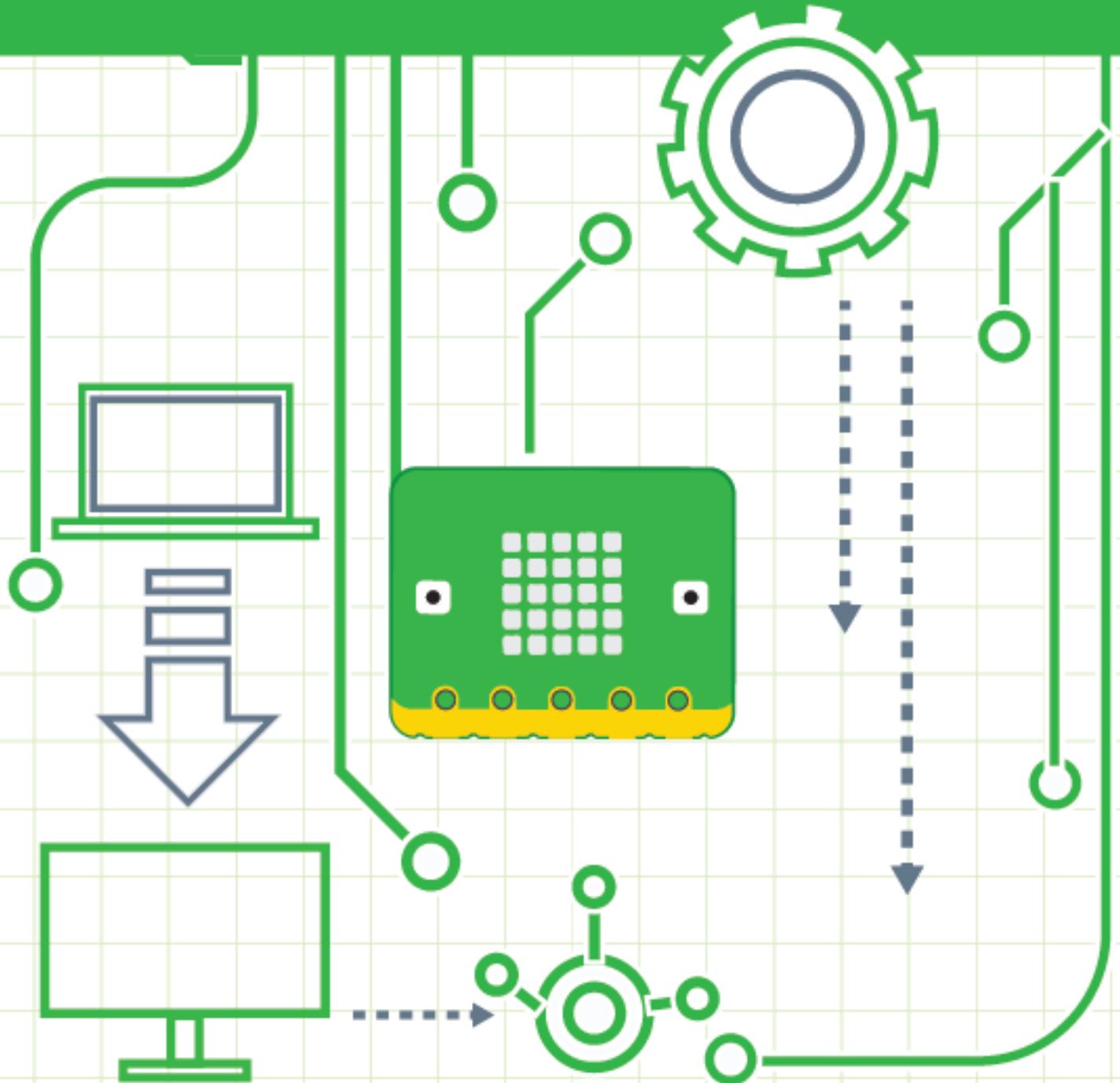
OhStem Education
Khơi Nguồn Sáng Tạo

Phần I

Các Thao Tác Cơ Bản Trên Yolo:Bit

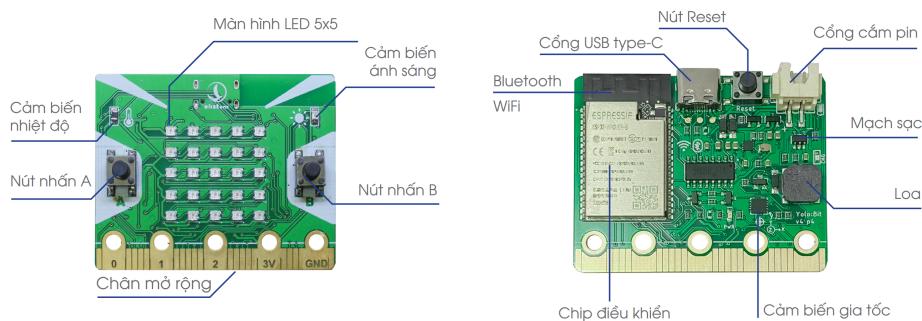
CHƯƠNG 1

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

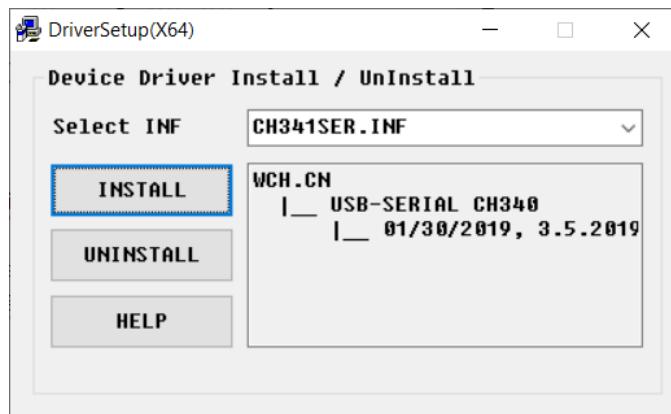
Để máy tính có thể kết nối và làm việc được với các thiết bị sử dụng chip USB CH340 như Yolo:Bit v4, robot xBot hay mạch điều khiển xController bằng dây cáp USB, bạn cần cài đặt driver theo các bước sau:

- 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:

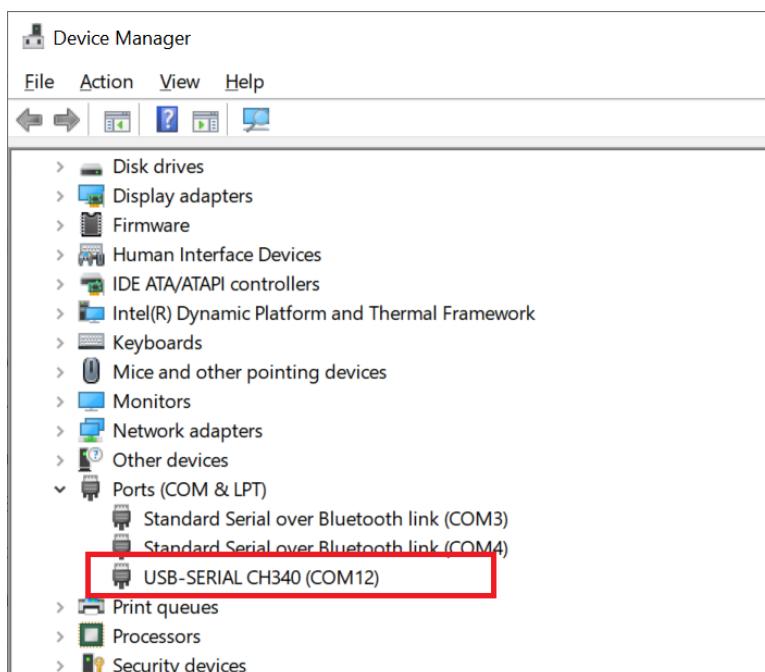


Hình 1.4: Cài đặt driver CH340 cho Yolo:Bit

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.

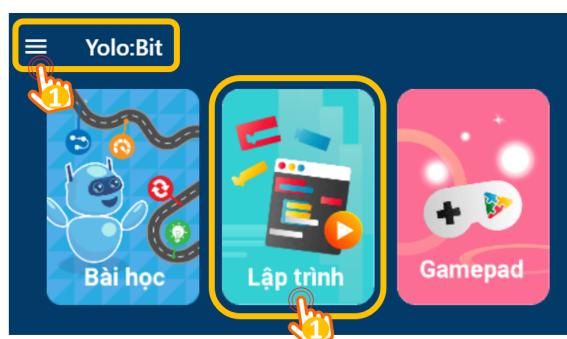
3 Môi trường lập trình

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:

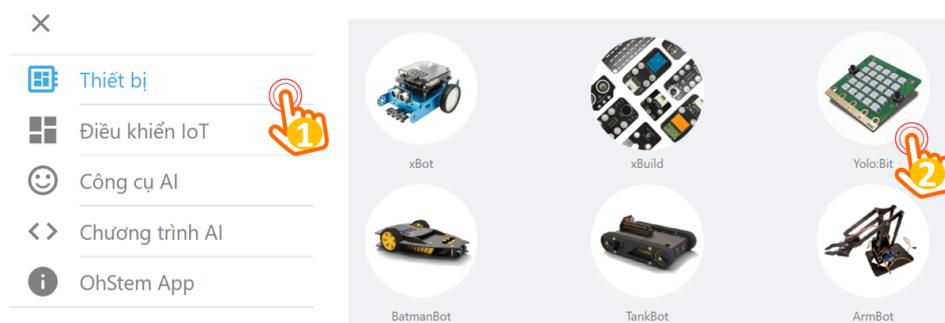
<https://app.ohstem.vn/>

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

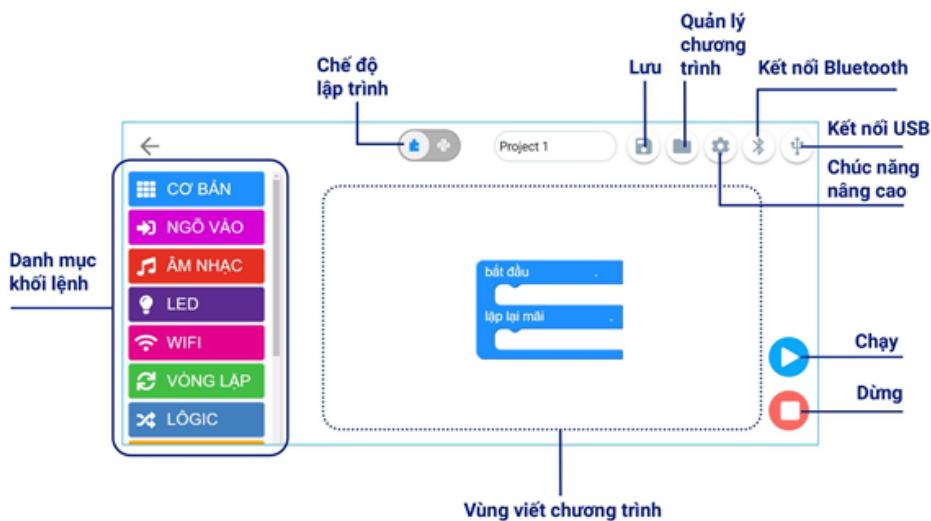
Nếu đúng là dòng chữ Yolo:Bit, bạn chỉ cần chọn tiếp vào **Lập trình**, là có thể bắt đầu các bước tiếp theo. Tuy nhiên, nếu thiết bị hiển thị không phải là YoloBit, bạn đọc hãy chọn lại thiết bị, bằng cách nhấn vào biểu tượng **Menu**, xuất hiện ở góc trên bên trái của giao diện lập trình. Giao diện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

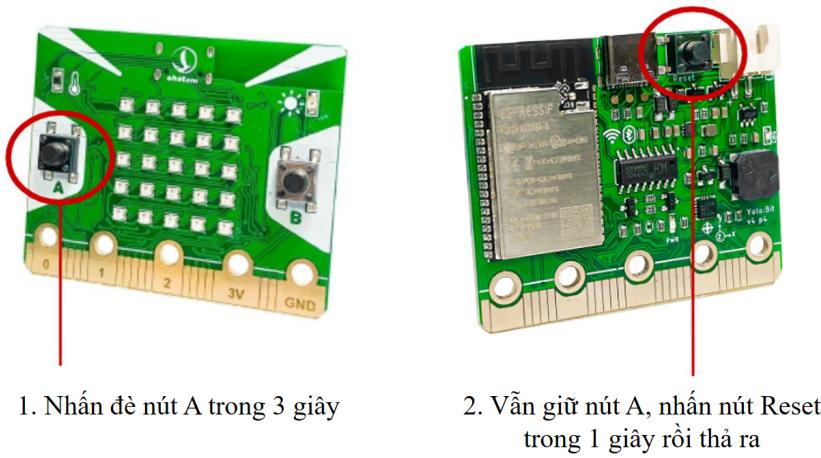
5 Khôi phục cài đặt gốc

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

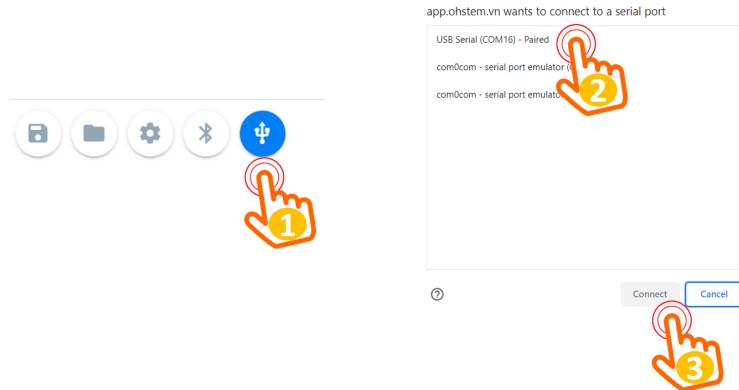
Sau khi mạch Yolo:Bit chớp tắt đèn 3 lần, nó sẽ được tự động reset lại. Khoảng 5 giây sau, một hiệu ứng đèn xoắn ốc sẽ xuất hiện. Đến lúc này, mạch Yolo:Bit đã được khôi phục cài đặt gốc thành công và sẵn sàng cho các bước tiếp theo.

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:



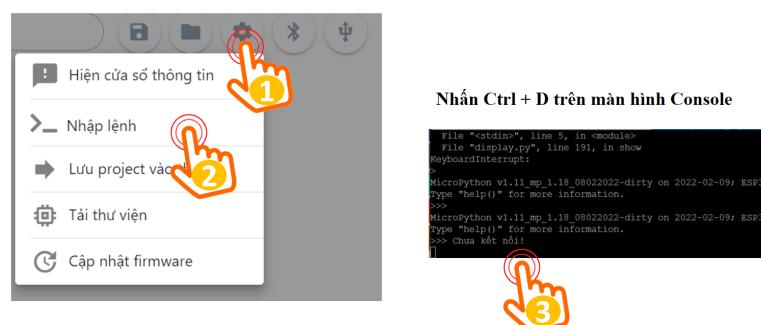
Hình 1.11: Kết nối với Yolo:Bit thông qua USB

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.

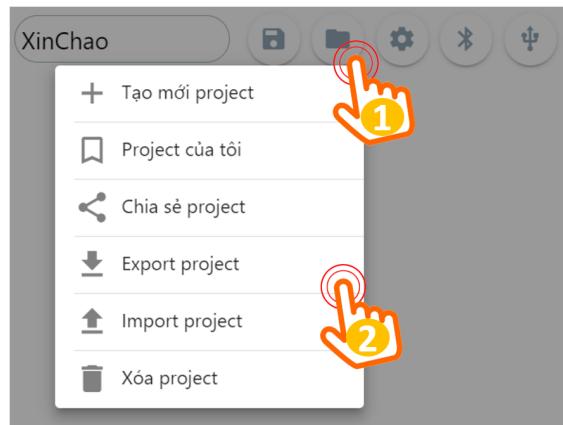
Một điều lưu ý quan trọng, là đôi khi vì chương trình mà chúng ta lưu vào thiết bị, làm cho việc chương trình mới có thể không thực thi thành công trên mạch Yolo:Bit. Lý do chủ yếu của việc này là do xung đột bộ nhớ chương trình. Trong trường hợp này, bạn đọc chỉ cần khôi phục lại cài đặt gốc của mạch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

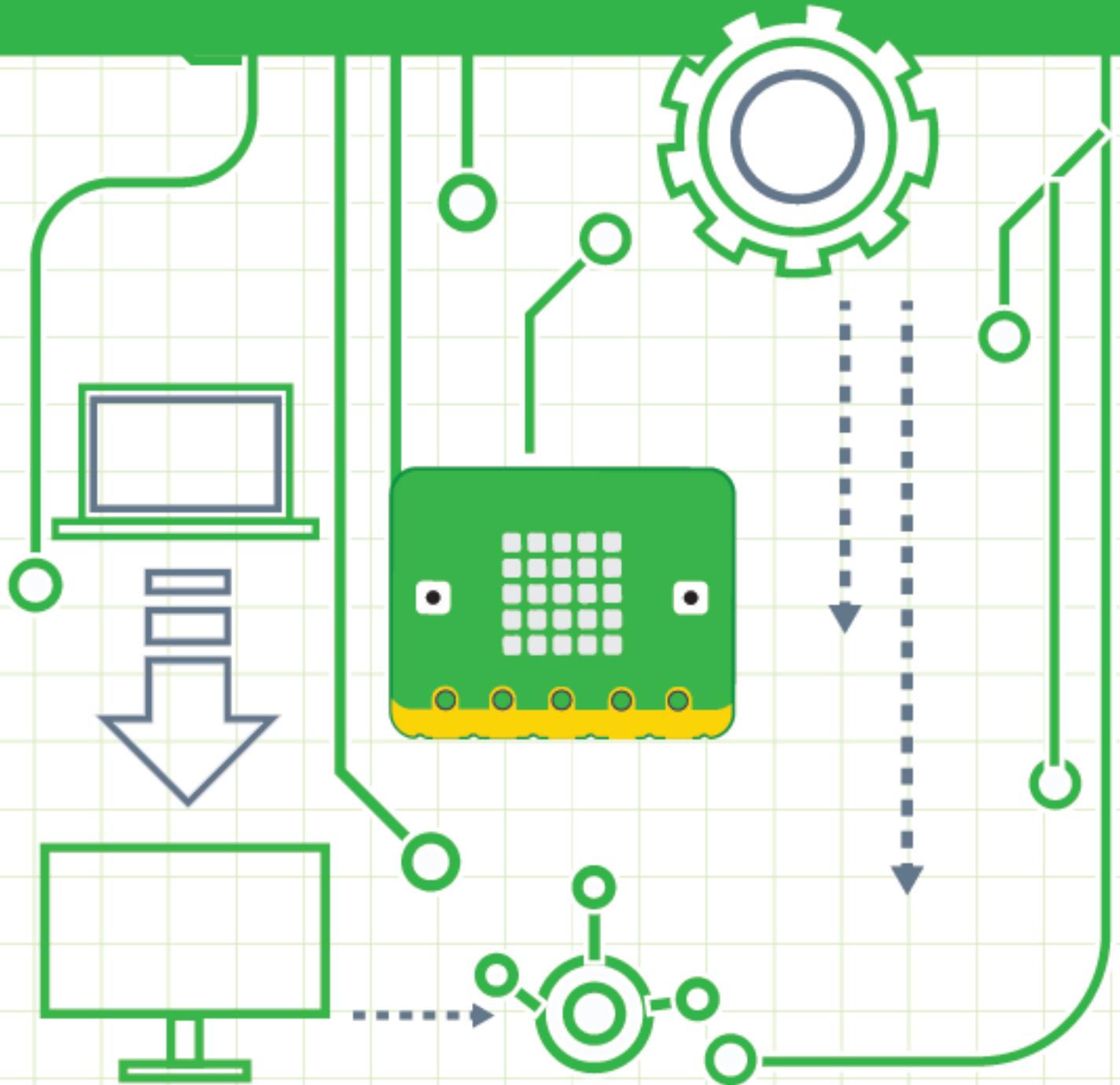


Hình 1.15: Chia sẻ chương trình trên Yolo:Bit

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.



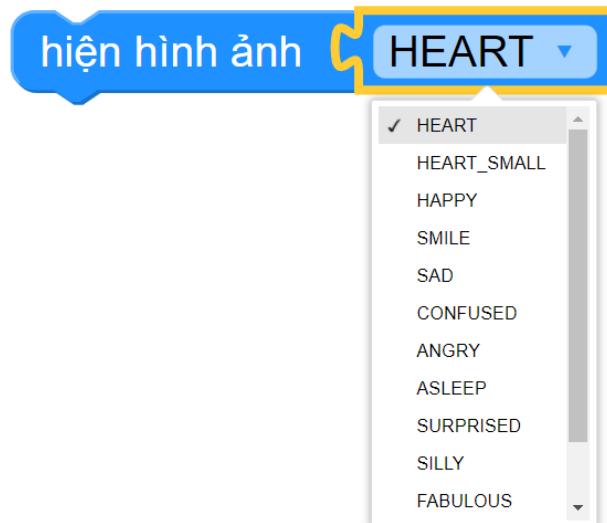
Hình 2.2: Kiểm tra tính năng của câu lệnh hiển chữ và hiển số

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

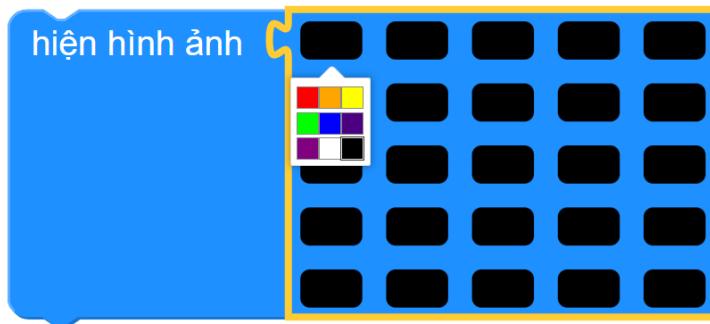
nhất, chúng ta có một danh sách các hình ảnh có thể được chọn lựa, như trình bày ở hình bên dưới:



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.



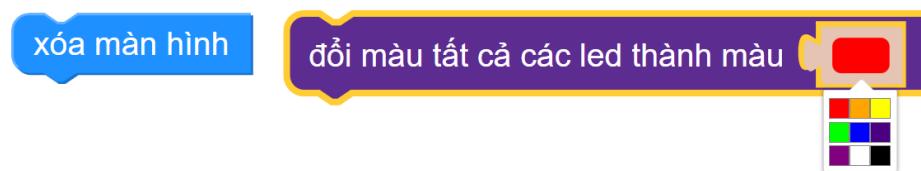
Hình 2.4: Câu lệnh hiện hình ảnh thứ 2 Yolo:Bit

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.

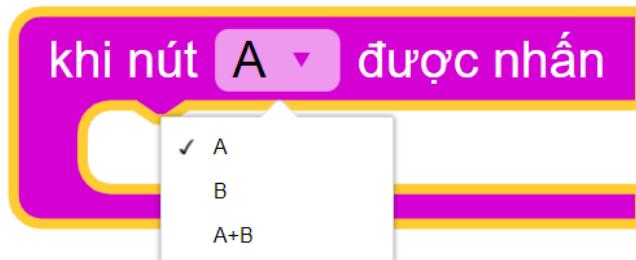


Hình 2.5: Câu lệnh xóa màn hình trên Yolo:Bit

Bên cạnh câu lệnh thứ nhất, xóa màn hình, câu lệnh thứ 2 cũng là một câu lệnh tiện dụng, câu lệnh **đổi màu tất 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 **lặp mãi mãi**, 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:



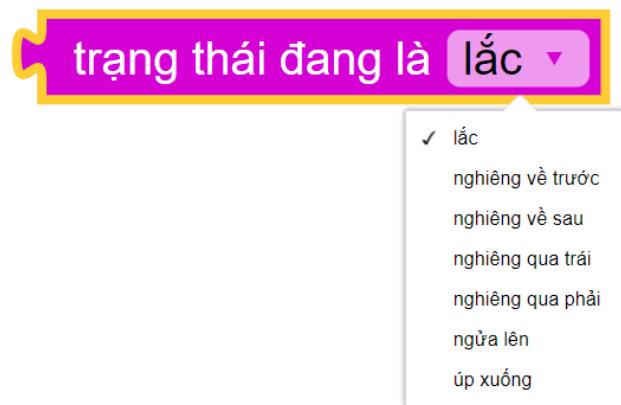
Hình 2.8: Chương trình ví dụ cho cảm biến

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:

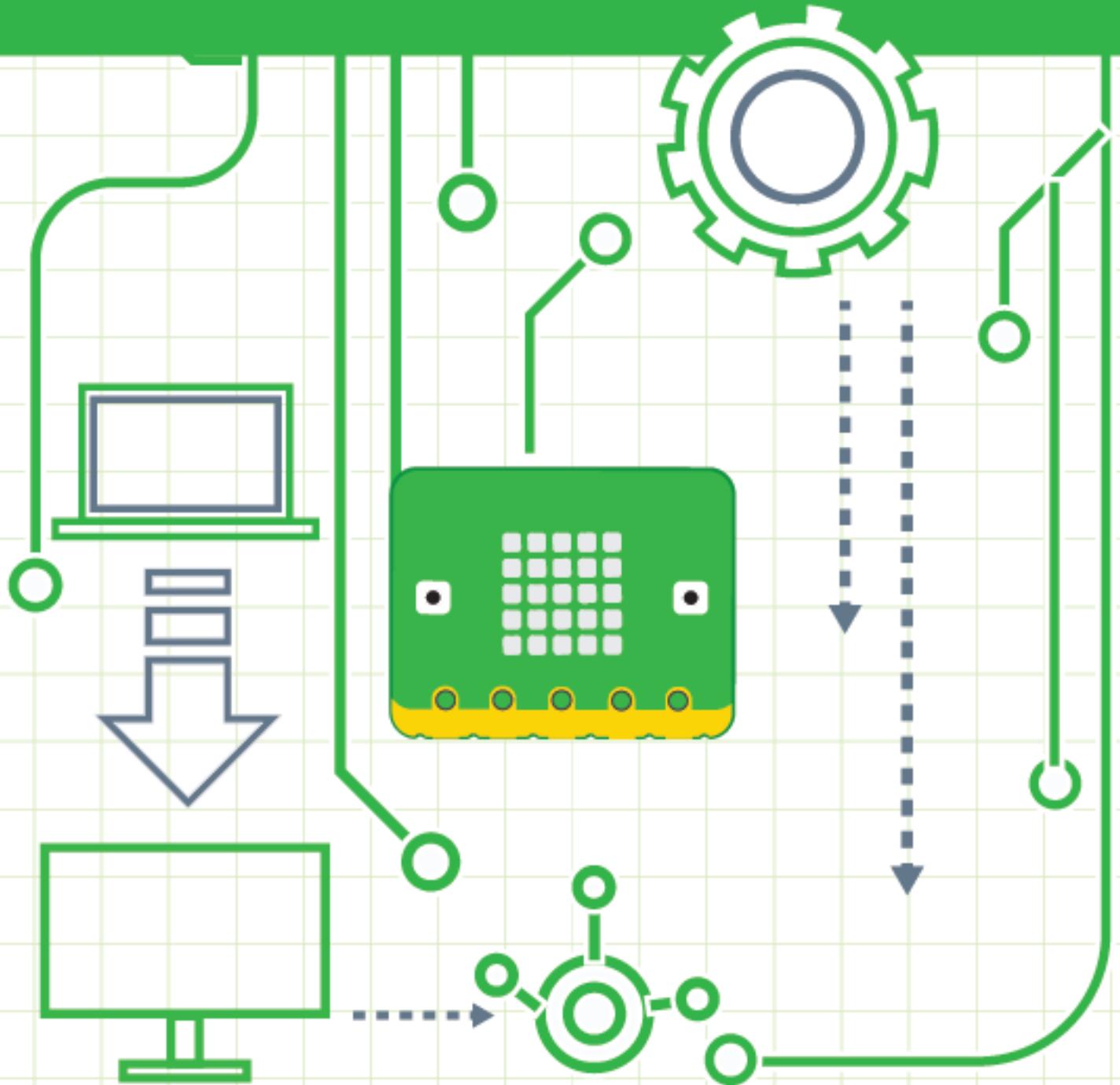


Hình 2.10: Sử dụng câu lệnh hành vi

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.

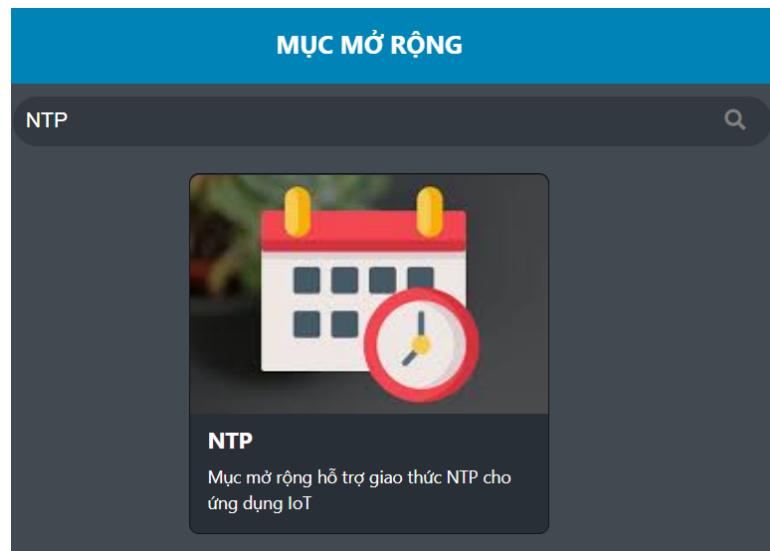
Với 2 nút nhấn A và B cùng màn hình gồm 25 đèn LED, chúng ta có thể xây dựng một số chức năng của đồng hồ thông minh như sau:

- 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

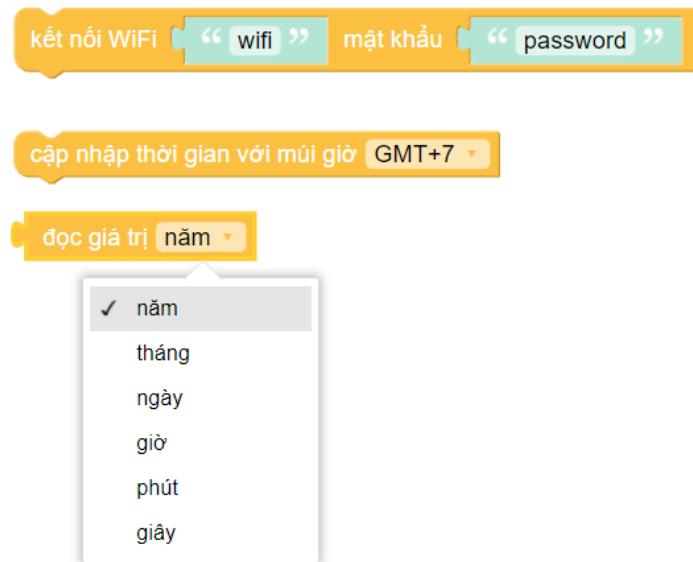
Thời gian trên Yolo:Bit được xây dựng dựa trên khái niệm Network Time Protocol (giao thức thời gian trên mạng). Network Time Protocol (NTP) là một thuật toán phần mềm, có công dụng giữ cho các máy tính và các thiết bị công nghệ khác nhau có thể đồng bộ hóa thời gian với nhau. NTP đã đạt được thành công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

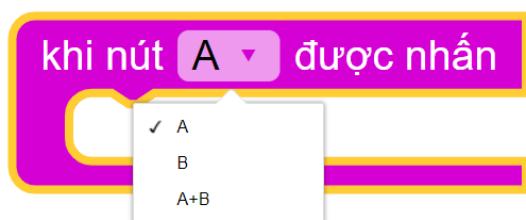
Câu lệnh 3: Có rất nhiều tùy chọn về thời gian giúp bạn đọc có thể thoải mái sử dụng trong ứng dụng của mình.

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:

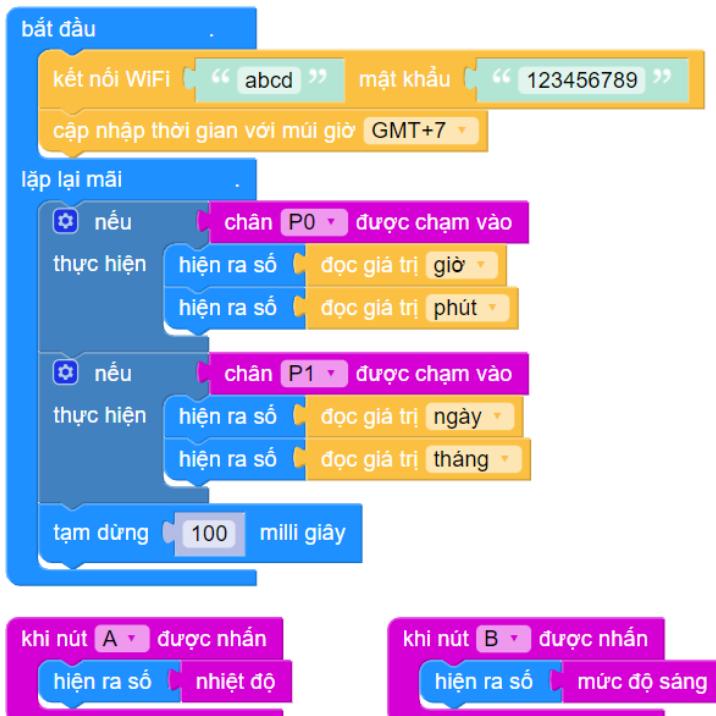


Hình 3.5: Chương trình cho nút nhấn A và B

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:

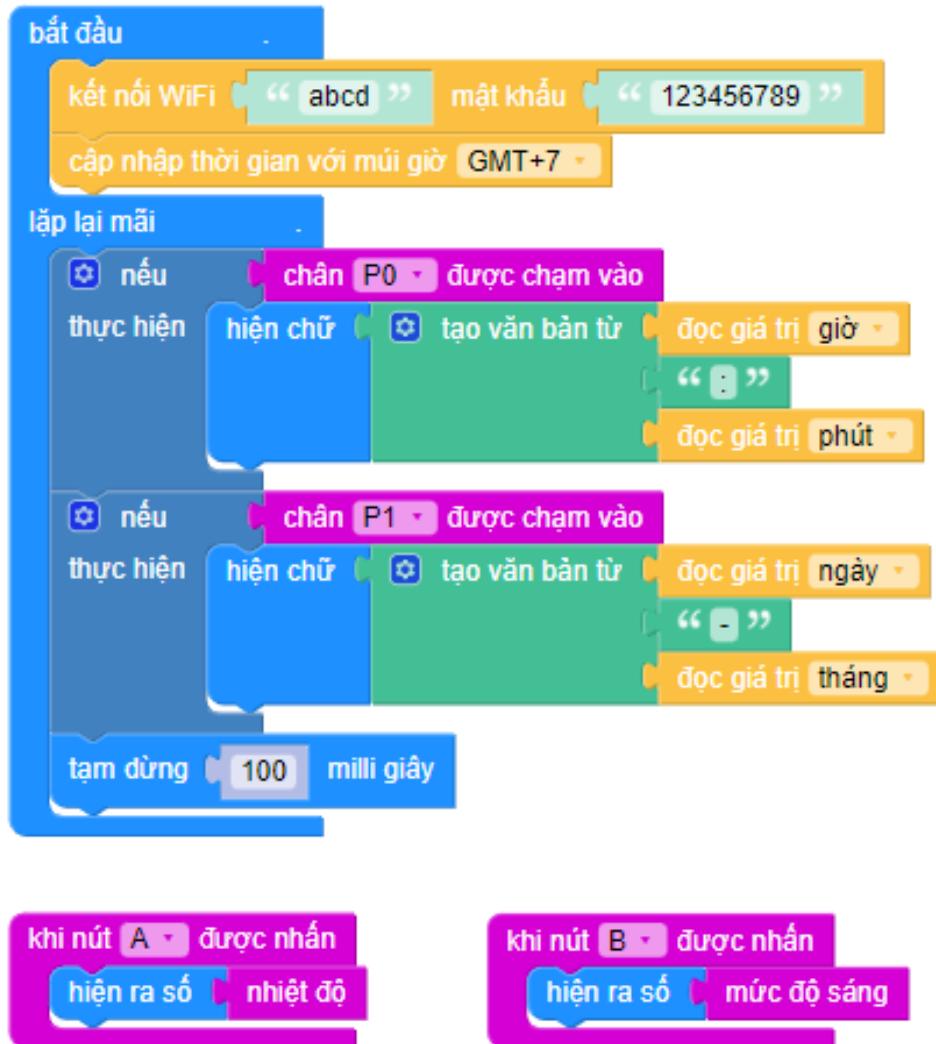


Hình 3.7: Câu lệnh ghép chuỗi

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 **tối**, 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:

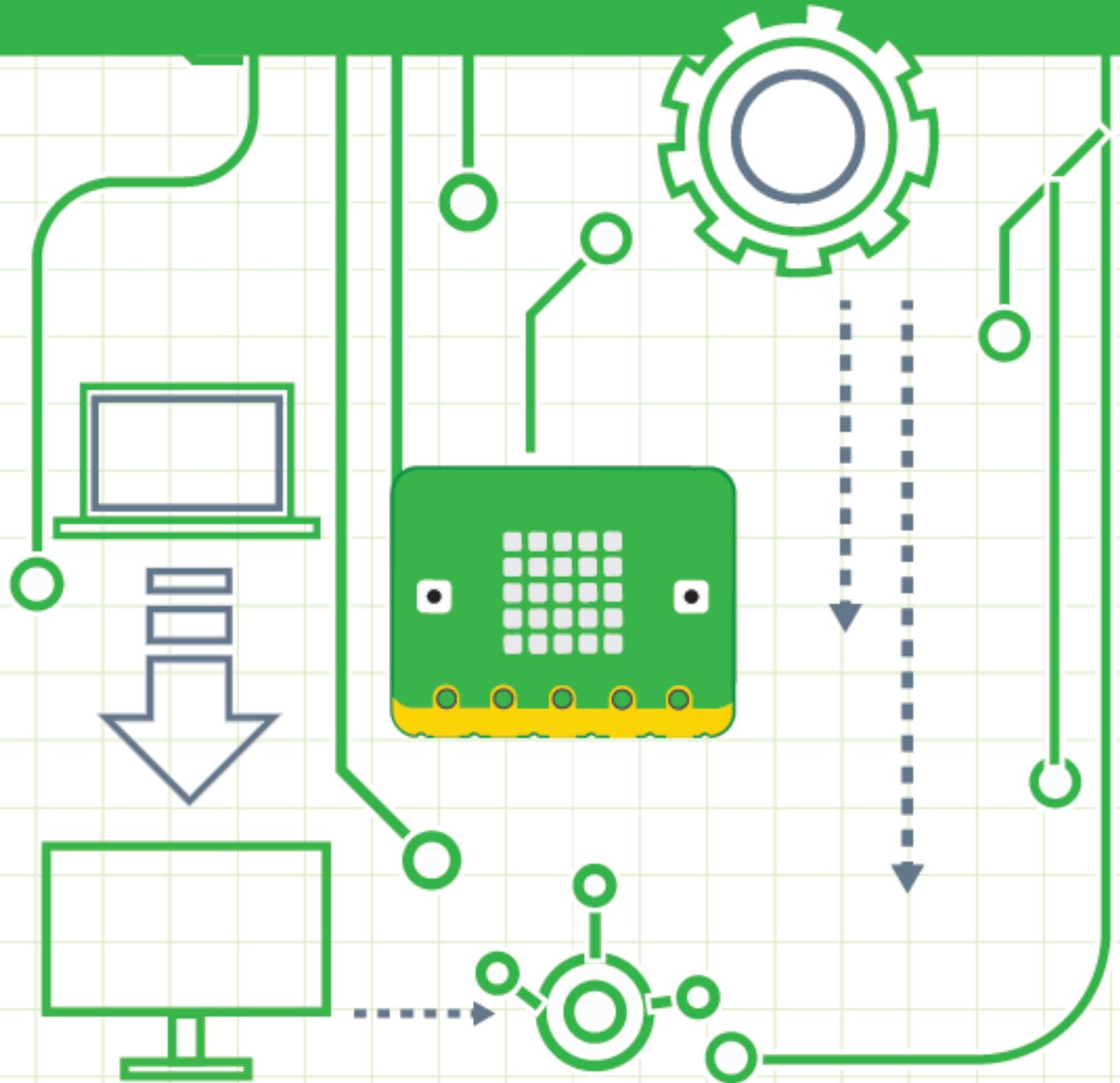
<https://app.ohstem.vn/#!/share/yolobit/26Y8nXJGC5Io1LorfXbJsTssgEL>

Phần II

Kết Nối Mở Rộng Cho Yolo:Bit

CHƯƠNG 4

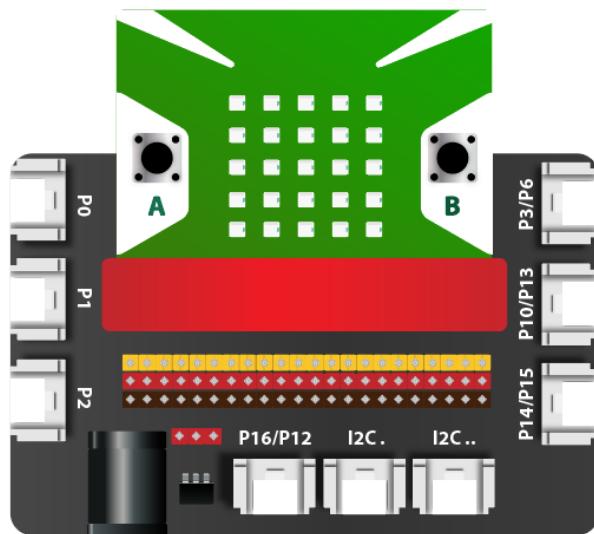
Mạch Mở Rộng - Đèn RGB



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 Nông Nghiệp Thông Minh trong giáo trình này, chúng ta sẽ cần kết nối thêm với các cảm biến đo chất lượng không khí hay đất trồng, 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**.



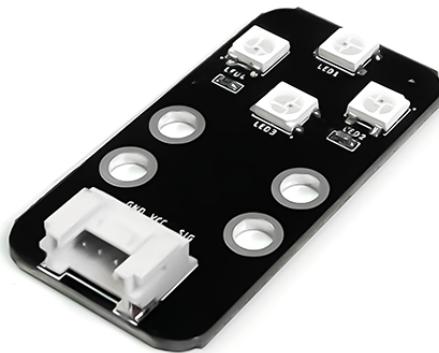
Hình 4.1: Mạch mở rộng cho Yolo:Bit

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

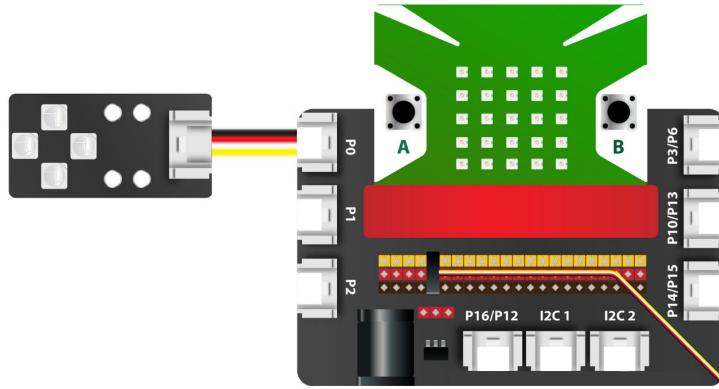
Trên thực tế, module trên là một dãy 4 đèn nối tiếp với nhau, gọi là NeoPixel. Đây là một sản phẩm được công ty Adafruit IO đưa ra, nhằm tránh lỗi chân cắm và giúp làm đơn giản hóa mạch điện. Toàn bộ dây LED có thể mở rộng lên đến 144 đèn, mỗi đèn có thể điều khiển được màu sắc riêng của nó từ 3 màu 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

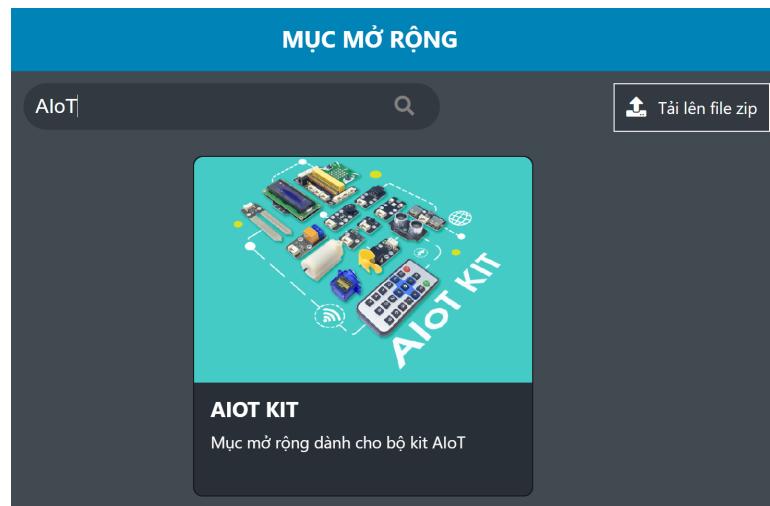
Nhờ các dây điện đã chuẩn hóa về nguồn đất và tín hiệu điều khiển, bạn gần như không cần phải lo lắng về việc kết nối các thiết bị với nhau. Các dây kết nối cũng chỉ có 1 chiều cắm, nhằm hạn chế tối đa việc cắm ngược, gây chập nguồn và hư hỏng thiết bị. Đây là một lợi thế vô cùng lớn của hệ thống mạch 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**.

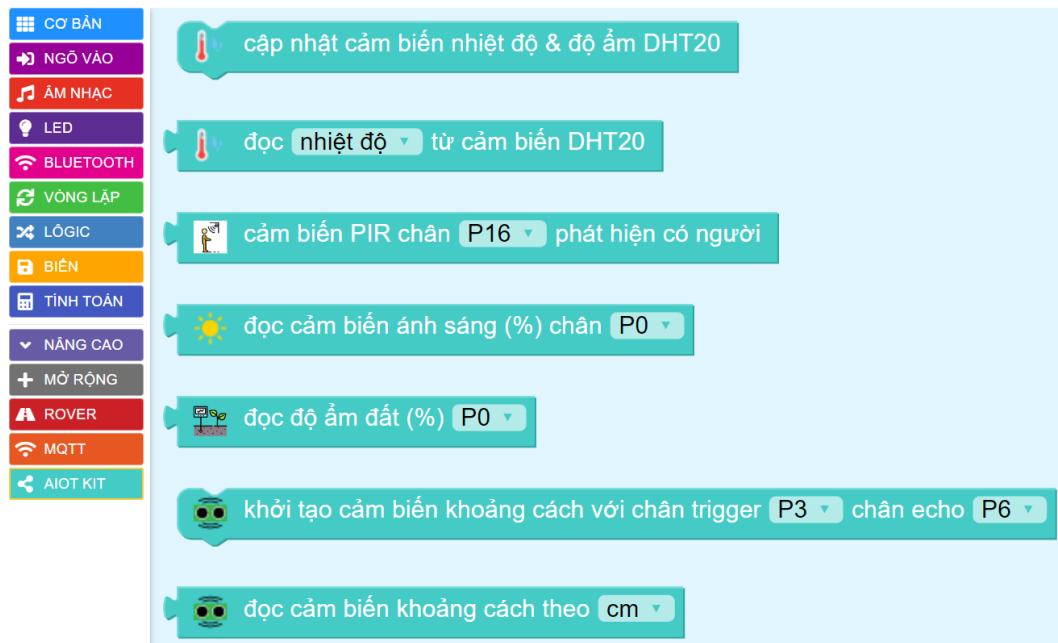
Không chỉ sử dụng trong bài này, **AIOT** còn sẽ được sử dụng xuyên suốt trong các phần hướng dẫn tiếp theo. Dựa vào thư viện này, bạn có thể lập trình với rất nhiều thiết bị cao cấp khác, như quạt (motor), cảm biến nhiệt độ và độ ẩm không khí, cảm biến độ ẩm đất hay các công tắc điện tử, màn hình LCD.

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

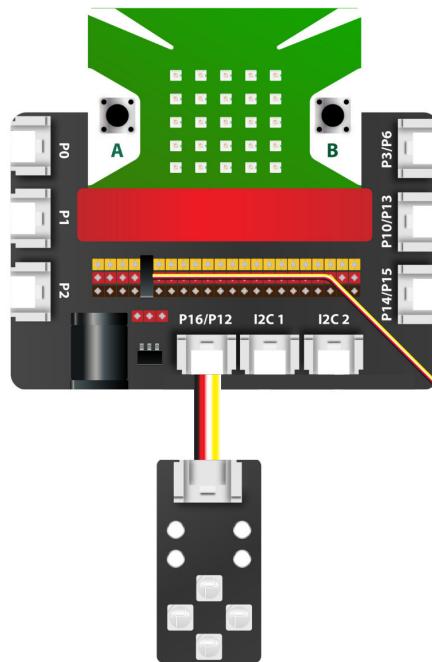
6 Khe cắm trên mạch mở rộng

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:

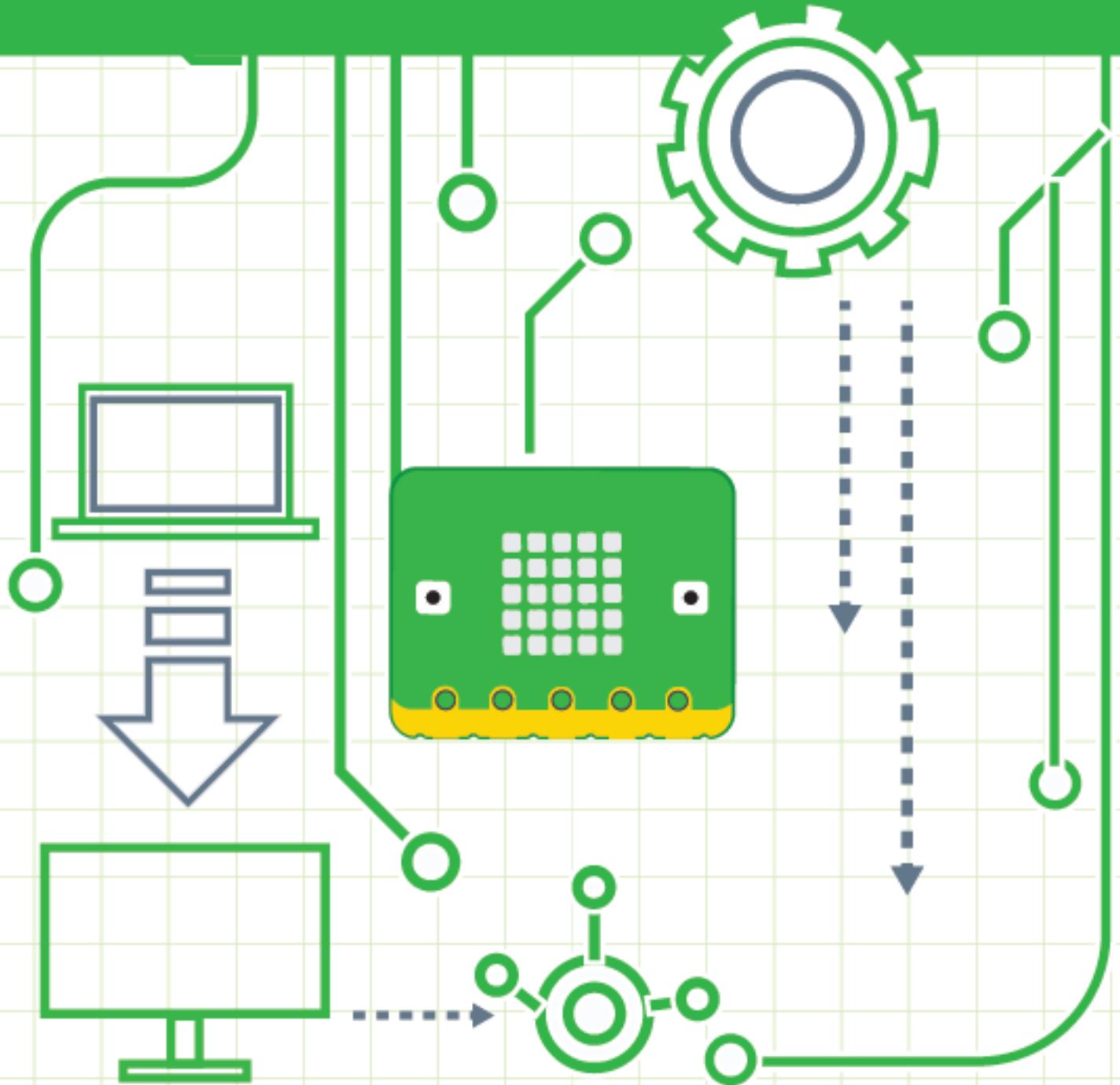


Hình 4.9: Đổi đèn RGB sang cổng P16/12

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

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 nông nghiệp thông minh, thông tin về nhiệt độ và độ ẩm phản ánh điều kiện nuôi trồng nông nghiệp khá quan trọng.



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

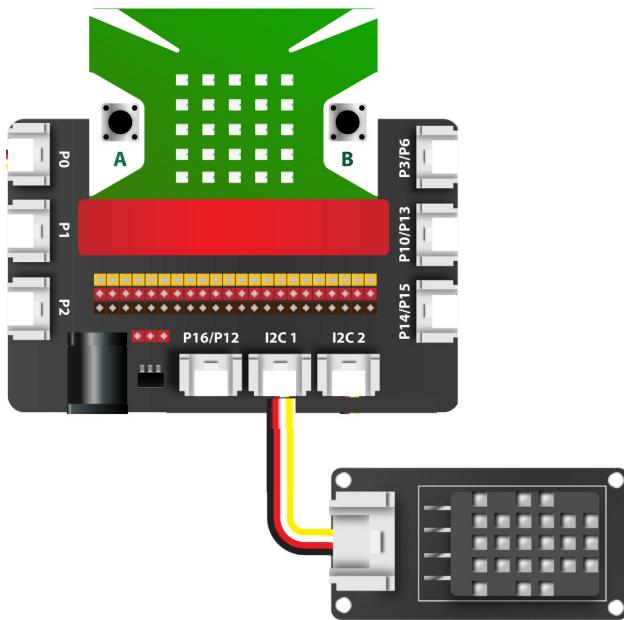
DHT20 là một thiết bị cảm biến rất phổ biến đối với các ứng dụng dân dụng, với ưu điểm là giá thành thấp và cách sử dụng đơn giản. Trong một số trường hợp, DHT20 còn có thể sử dụng cho một số máy sấy thực phẩm. Dãy đo của cảm biến khá phù hợp trong môi trường bình thường không có biến động lớn, với độ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



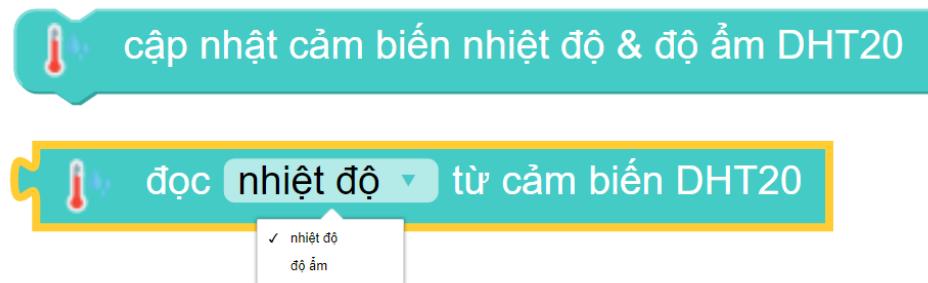
Hình 5.2: Kết nối DHT20 vào mạch mở rộng

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:



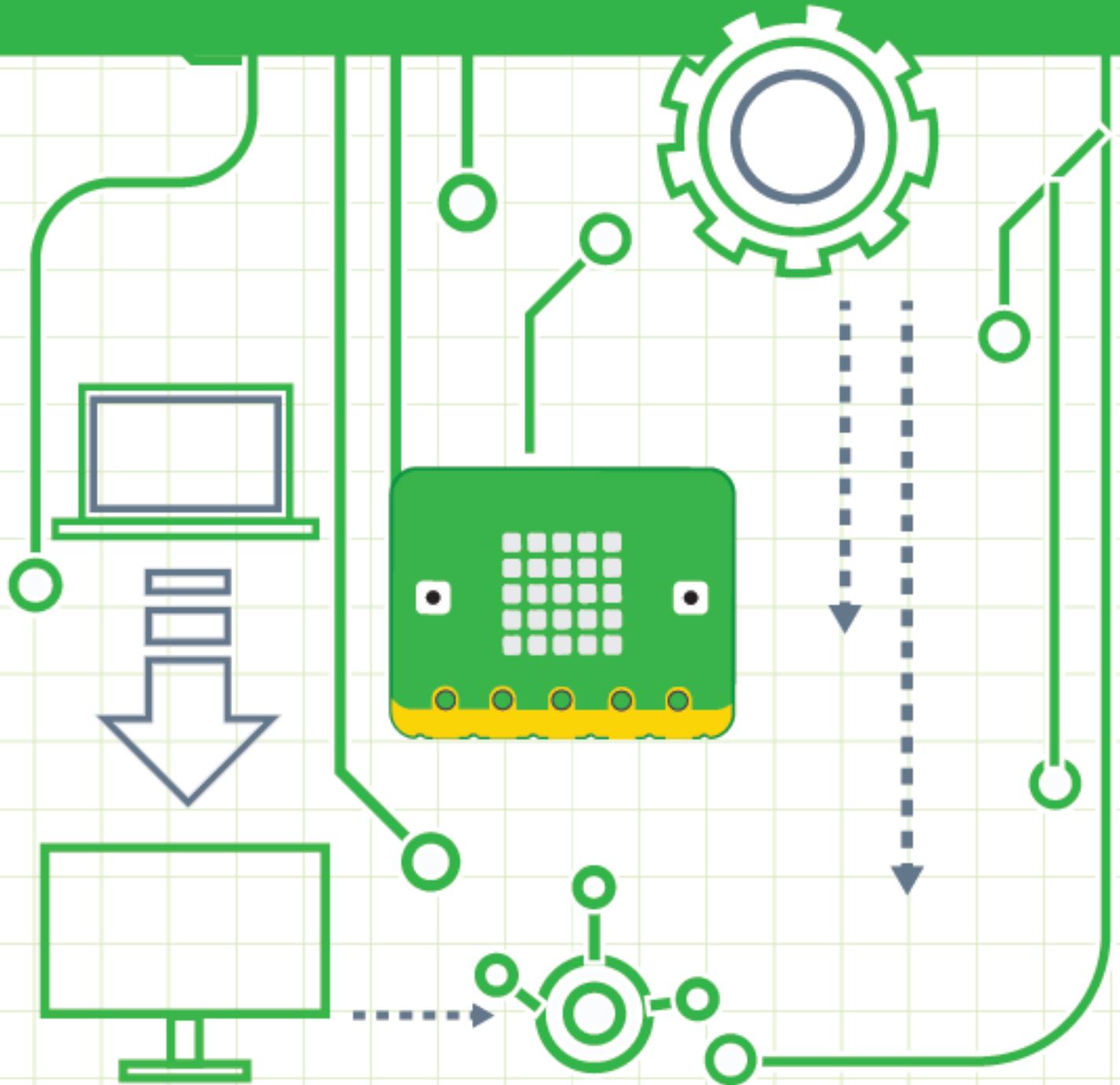
Hình 5.5: Chương trình hiển thị thêm đơn vị

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

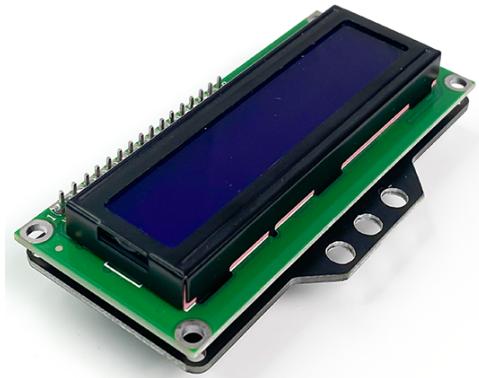
Màn Hình LCD



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 hỗ trợ có tên là I2C đã được thiết kế đi kèm với màn hình LCD.



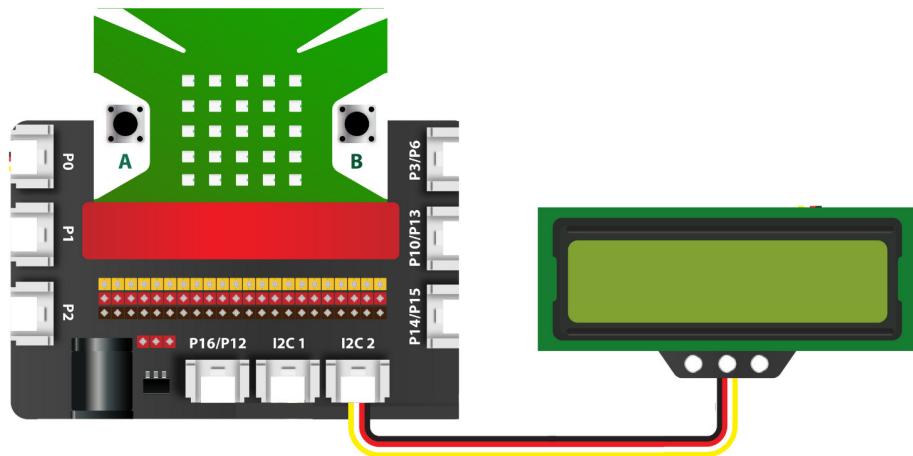
Hình 6.1: Màn hình LCD tích hợp mạch giao tiếp I2C

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.

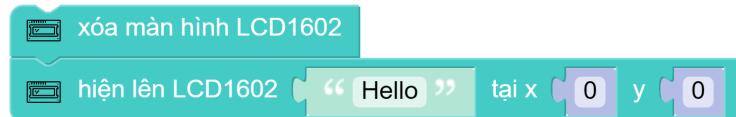


Hình 6.2: Kết nối màn hình LCD vào hệ thống

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:

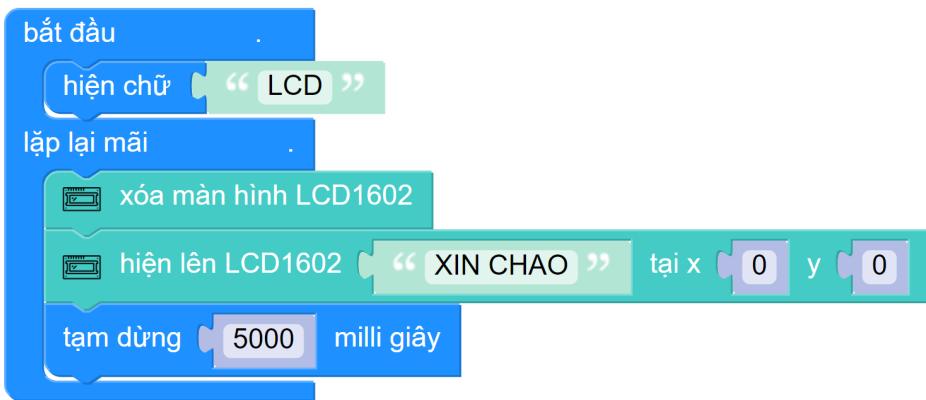


Hình 6.3: Câu lệnh để hiển thị lên LCD

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 **NÂNG CAO, CHỮ VIẾT**. Khối 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).

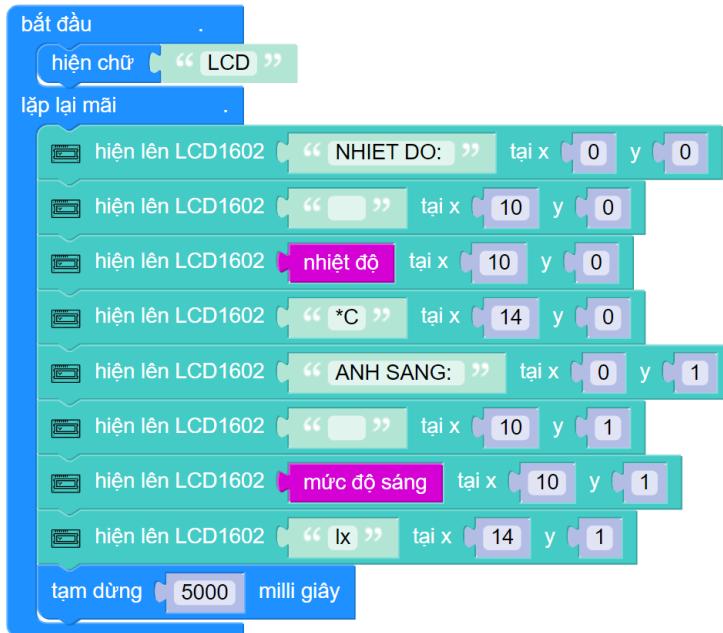
Từ tọa độ 10 trên mỗi dòng, chúng ta sẽ hiển thị ra khoảng trắng để xóa thông tin cũ, trước khi hiển thị thông tin mới ra màn hình. Chương trình gợi ý cho tính năng này sẽ như sau:



Hình 6.8: Hiển thị nâng cao với LCD

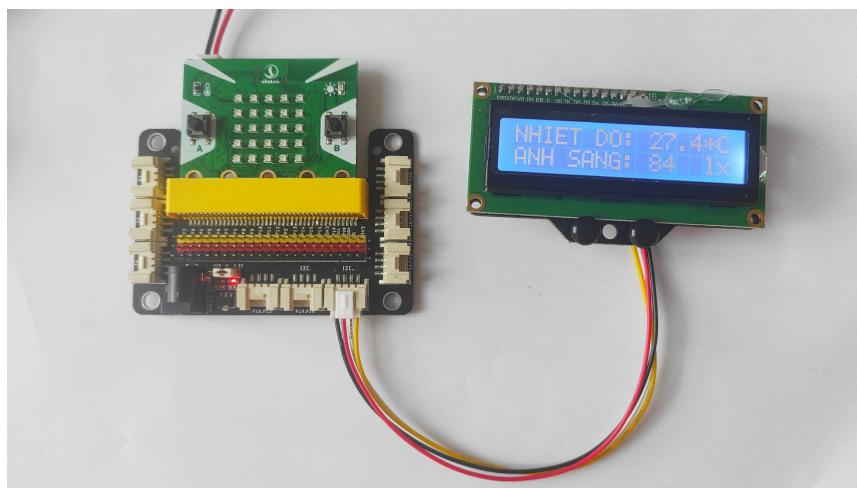
Chúng tôi sẽ dùng **4 kí tự** để hiển thị thông tin về nhiệt độ và ánh sáng. Do đó, trong câu lệnh thứ 2, có 4 kí tự khoảng trắng được sử dụng, nhằm mục đích xóa thông tin cũ trước khi hiển thị thông tin mới ra màn hình LCD. Bạn đọc hãy lưu ý về thông tin các tọa độ của x và y khi tối ưu việc hiển thị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

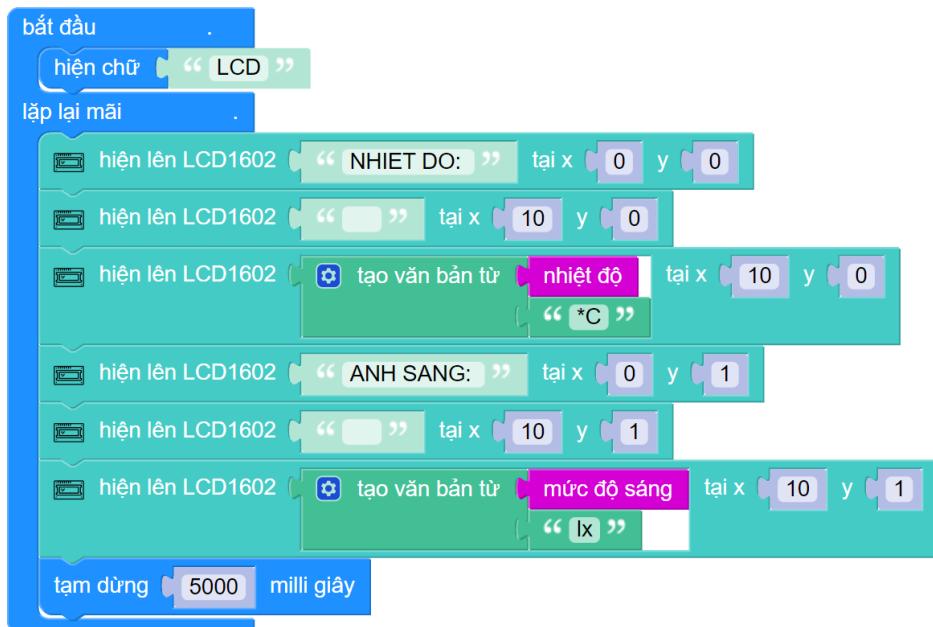
Mặc dù tiêu đề và đơn vị là các thông tin cố định, bạn đọc nên để nó trong phần **lắp mãi mãi** thay vì chỉ thực hiện một lần trong **bắt đầu**. Khi mới khởi động, Yolo:Bit có thể hoạt động không ổn định ở một lệnh nào đó, làm cho việc hiển thị bị lỗi, và điều đó sẽ không được khôi phục lại. Khi để trong lắp 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.

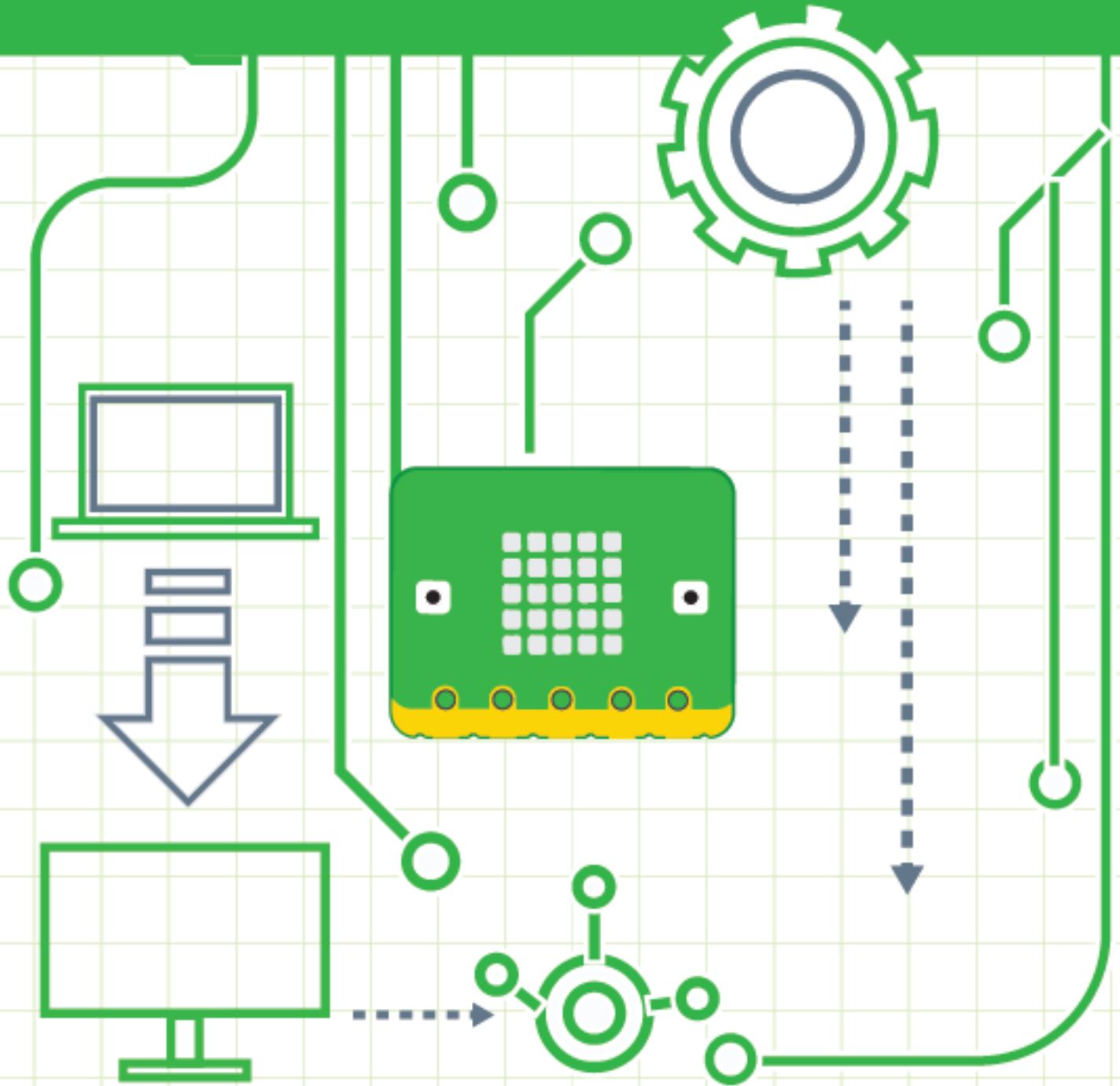


Hình 6.11: Hiển thị đơn vị sát vào thông tin

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

Máy Bơm Chìm



1 Giới thiệu

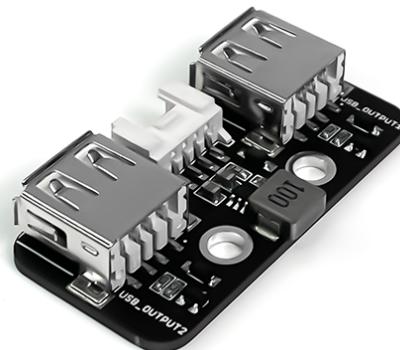
Máy bơm là một thiết bị khá thông dụng và cần thiết trong ứng dụng nông nghiệp thông minh. Trong bài hướng dẫn này, chúng tôi sẽ sử dụng máy bơm chìm để minh họa cho việc điều khiển máy bơm, phục vụ cho công tác tưới tiêu trong tương lai. Đặc điểm của máy bơm này, là nó được đặt chìm trong nước và không cần nước mồi mà vẫn có thể hoạt động được. Do đó, nó mới có tên là máy bơm chìm. Hình ảnh của thiết bị này được trình bày như bên dưới.



Hình 7.1: Máy bơm chìm USB

Máy bơm chìm sử dụng trong dự án này có đầu kết nối là cổng USB nên rất thuận tiện cho việc kết nối với mạch mở rộng của Yolo:Bit. Khe USB của máy bơm được sử dụng để cấp nguồn cho máy bơm hoạt động. Do đó, bạn đọc có thể thử nhanh máy bơm có hoạt động hay không bằng cách cắm trực tiếp nó vào cổng USB của máy tính.

Để có thể lập trình điều khiển máy bơm, chúng ta cần phải kết nối nó thông qua một mạch điều khiển, gọi là mạch điều khiển 2 cổng USB, có hình ảnh như trình bày ở hình bên dưới.

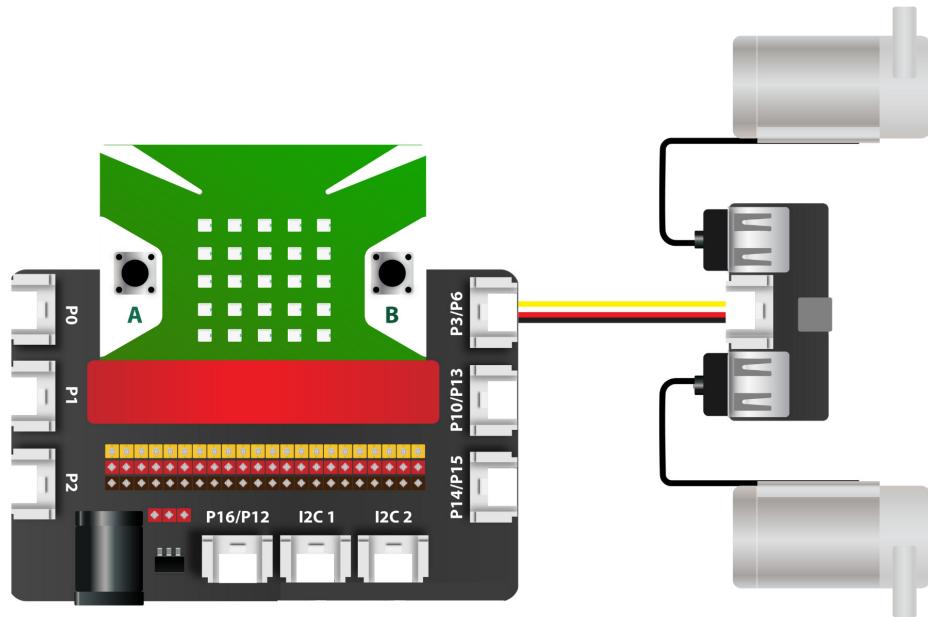


Hình 7.2: Thiết bị điều khiển 2 cổng USB

Trong bài hướng dẫn này, chúng ta sẽ kết nối máy bơm và mạch điều khiển vào mạch mở rộng Yolo:Bit và lập trình để điều khiển nó.

2 Kết nối với Yolo:Bit

Mạch điều khiển 2 cổng USB cần hai chân tín hiệu cho việc lập trình, để có thể đóng ngắt 2 cổng USB một cách độc lập. Do đó, bạn đọc cần kết nối nó ở các cổng có 2 chân tín hiệu, như ví dụ ở hình bên dưới.



Hình 7.3: Kết nối các thiết bị với mạch mở rộng

Theo như kết nối ở trên, cổng USB đầu tiên sẽ được điều khiển thông qua chân lập trình P13. Trong khi đó, cổng thứ 2 sẽ được điều khiển thông qua chân lập trình P16.

Câu lệnh chính để điều khiển từng chân lập trình bật hay tắt nằm trong mục **NÂNG CAO**, và chọn tiếp vào **CHÂN CẤM**, như trình bày ở hình bên dưới.



Hình 7.4: Câu lệnh bật tắt một chân lập trình

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 hoạt nguồn điện ở cổng USB

tương ứng, và làm quay máy bơm. Ngược lại, nguồn điện ở cổng USB sẽ tắt và máy bơm sẽ dừng quay.

3 Lập trình điều khiển máy bơm

Dựa vào câu lệnh hỗ trợ cho việc bật tắt một chân điều khiển, chương trình để bật tắt máy bơm kết nối với cổng USB thứ nhất có thể được hiện thực đơn giản như sau:



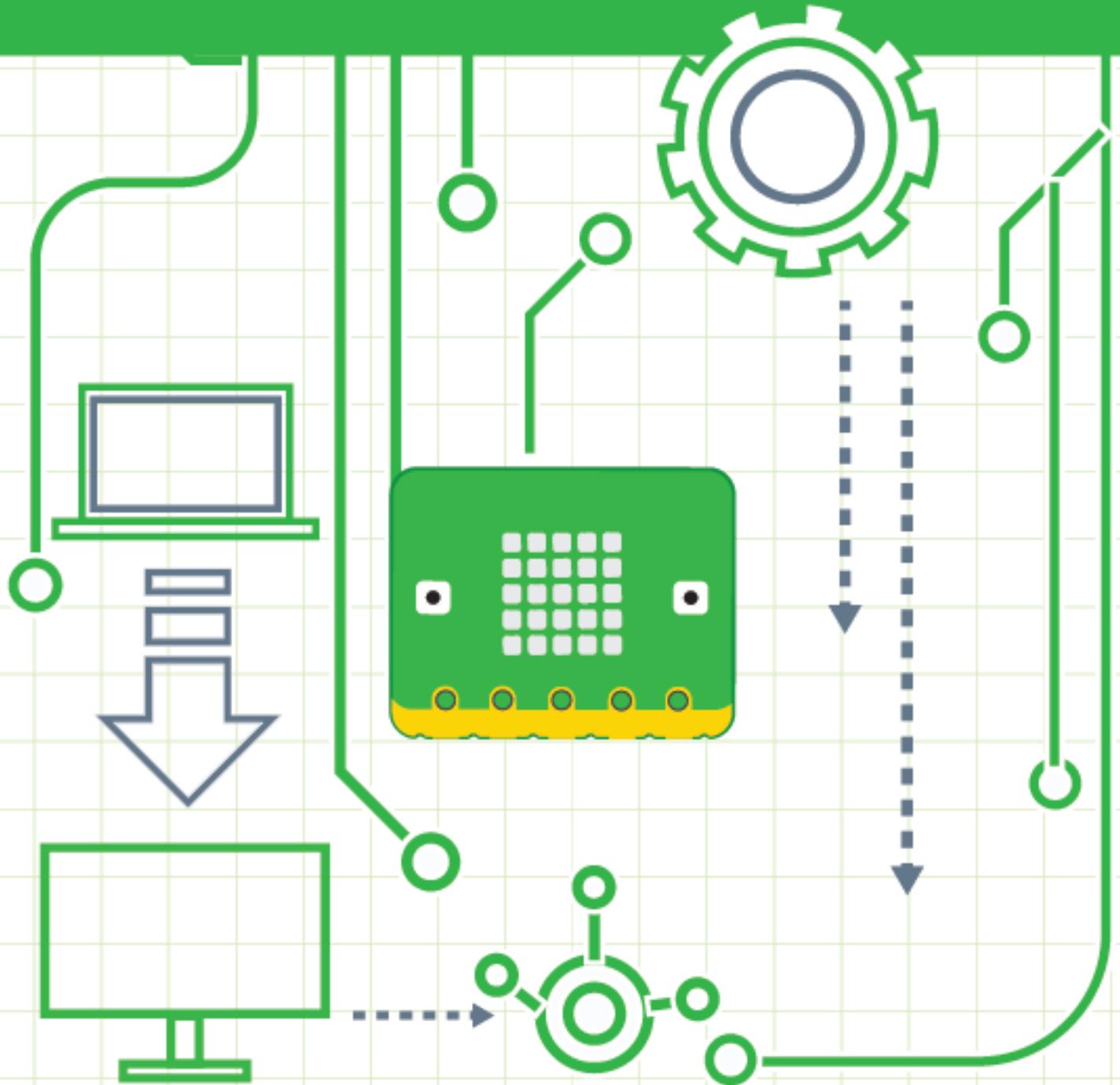
Hình 7.5: Chương trình bật tắt máy bơm kết nối với cổng USB thứ nhất

Trong trường hợp muốn điều khiển máy bơm thứ 2, chân **P16** sẽ được lựa chọn trong câu lệnh điều khiển. Bạn đọc có thể chủ động thay đổi chương trình bên trên để điều khiển máy bơm kết nối với cổng USB thứ 2.

Bên cạnh đó, bạn đọc cũng có thể thay đổi cổng kết nối để thành thạo hơn trong việc đọc chân kết nối và thay đổi chương trình cho tương ứng. Trên mạch mở rộng của Yolo:Bit vẫn còn các cổng có 2 chân điều khiển, như **P10/P13** và **P14/P15**. Việc thành thạo trong việc kết nối và lập trình là rất quan trọng trong việc hiện thực một dự án lớn trong tương lai.

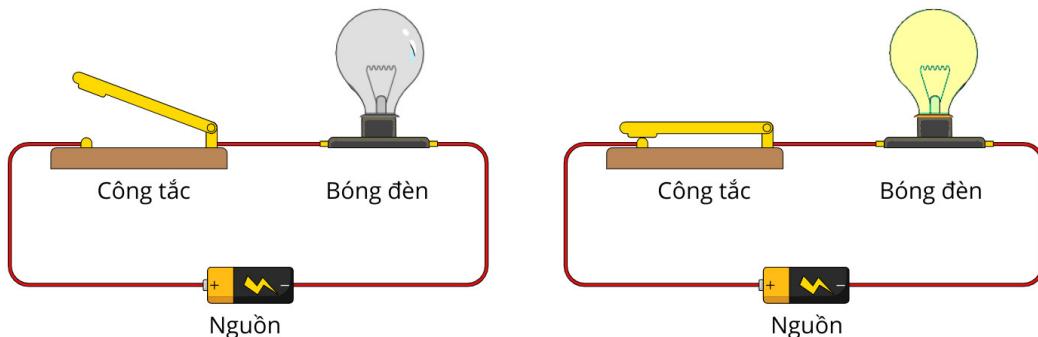
CHƯƠNG 8

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:

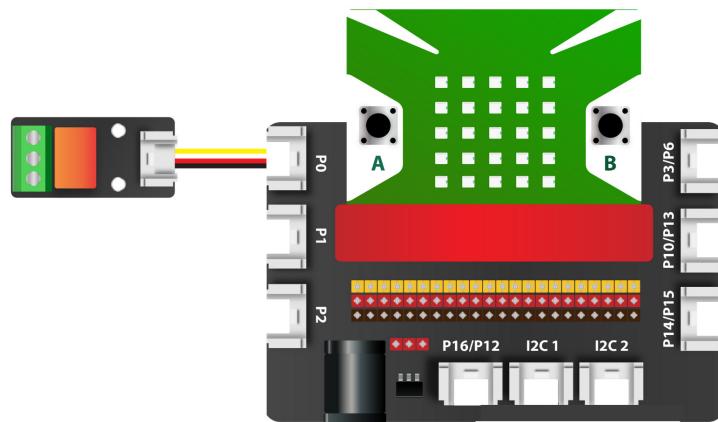
- Chân COM: viết tắt của chữ Common, là chân chung và chỉ nối với 1 trong 2 chân còn lại.
- Chân NC: viết tắt của chữ Normally Closed, là chân thường đóng. Khi không có điện ở phần điều khiển, COM và NC được nối với nhau.
- Chân NO: viết tắt của chữ Normally Open, là chân thường mở. Khi có điện ở phần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

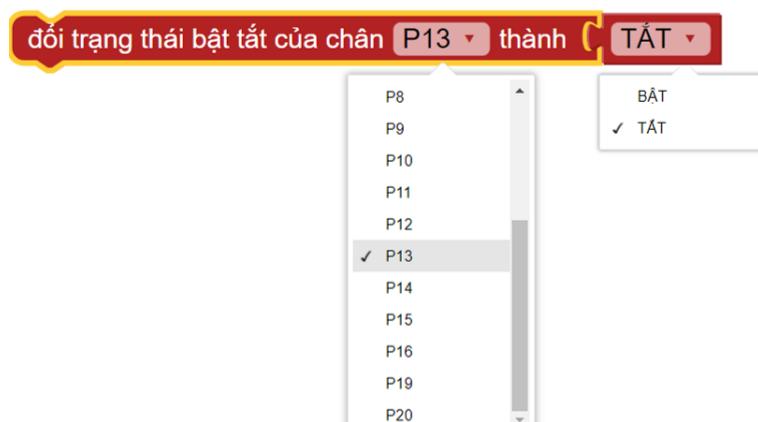
3 Kết nối Relay với Yolo:Bit

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

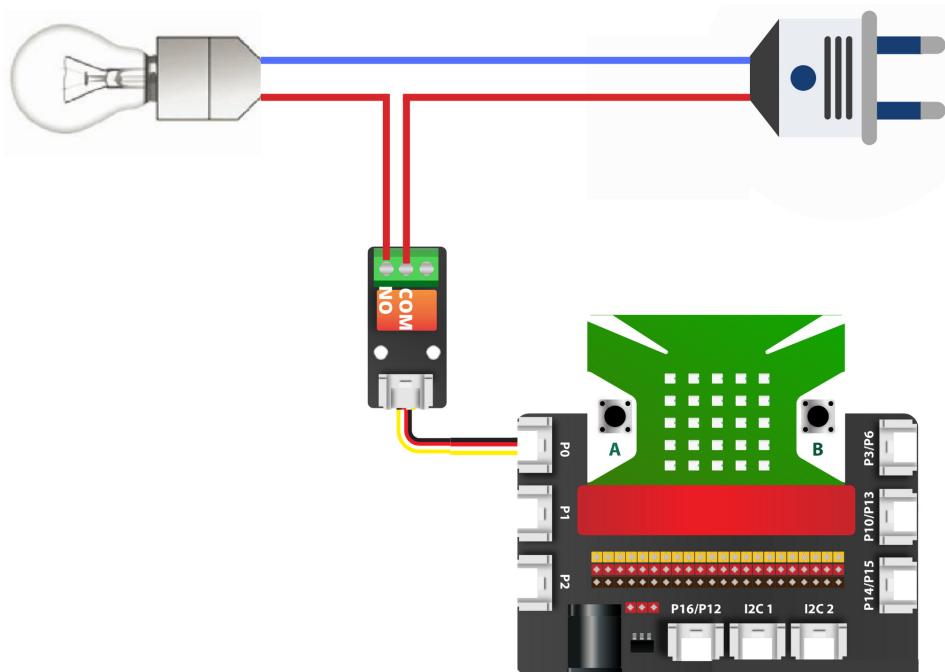
Hoàn toàn giống với hướng dẫn ở bài trước, câu lệnh này có 2 tùy chọn, bao gồm chân điều khiển và trạng thái **BẬT** hay **TẮT**. Khi chọn là **BẬT**, tín hiệu từ chân điều khiển sẽ kích một lực nam châm để bật Relay (COM nối với NO). Với kết nối như Hình 8.3, chương trình đơn giản để bật tắt Relay như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

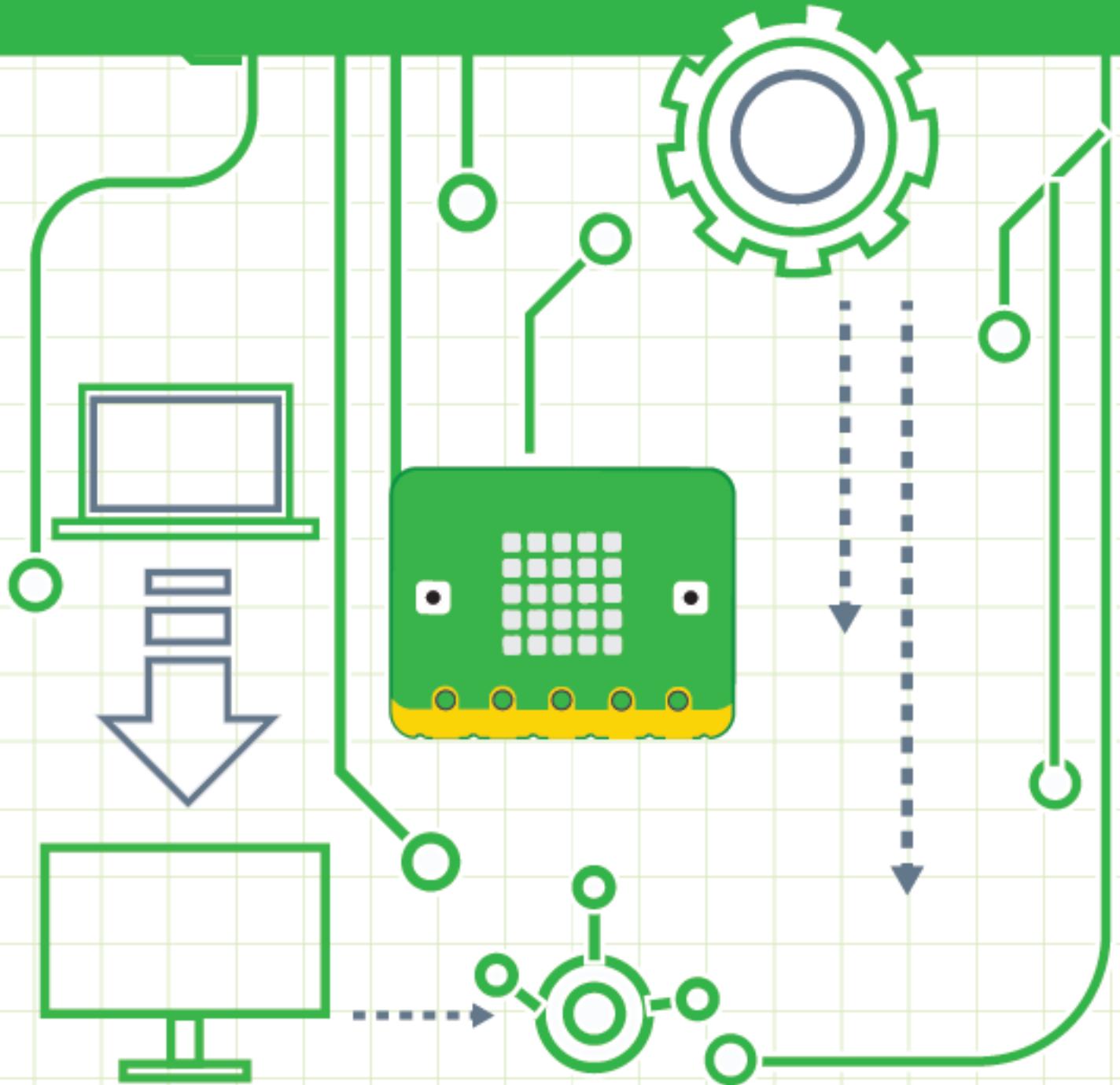


Hình 8.6: Kết nối công tắc và đèn 220V

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 Độ Ẩm Đất



1 Giới thiệu

Trong ứng dụng dành cho nông nghiệp thông minh, độ ẩm đất là thông số quan trọng cần phải giám sát, bởi nó ảnh hưởng trực tiếp tới việc tăng trưởng của cây trồng. Trong bài hướng dẫn này, chúng tôi sẽ hướng dẫn nguyên lý cũng như cách lập trình với cảm biến độ ẩm đất sau đây:



Hình 9.1: Cảm biến đo độ ẩm đất

Nguyên lý hoạt động của cảm biến này được dựa vào tính dẫn điện giữa 2 điện cực của cảm biến:

- Khi độ ẩm khô, việc dẫn điện sẽ khó khăn. Lúc này cảm biến độ ẩm đóng vai trò giống như một điện trở có giá trị lớn.
- Trong môi trường có nước nhiều, độ ẩm cao, việc dẫn điện sẽ dễ dàng và cảm biến có trở kháng nhỏ.

Dựa vào nguyên lý thay đổi về trở kháng trong 2 trường hợp trên, dữ liệu trả về sẽ thay đổi theo. Những cảm biến dựa trên nguyên lý trở kháng này sẽ rất dễ tích hợp vào chương trình, do dữ liệu trả về từ cảm biến là một giá trị điện áp. Chỉ cần đọc trực tiếp giá trị này, chúng ta sẽ có thông tin về cảm biến cần đo. Cũng vì lý do đầu ra là điện áp, các cảm biến dựa trên nguyên lý trở kháng còn được gọi là cảm biến Analog, hoặc cảm biến ADC.

Trong hướng dẫn này, chúng tôi sẽ trình bày quy tắc cơ bản để kết nối với cảm biến analog nói chung và cảm biến độ ẩm nói riêng. Tiếp theo đó, là các câu lệnh cơ bản để lấy giá trị từ nó. Tuy nhiên, giá trị thu được là giá trị số không có đơn vị. Trong trường hợp muốn biểu diễn giá trị này thành thông tin có đơn vị (chẳng hạn như độ ẩm có đơn vị là %), chúng ta cần phải xây dựng một công thức chuyển đổi. Các nội dung hướng dẫn chính trong bài này như sau:

- Kết nối cảm biến với mạch Yolo:Bit
- Lập trình truy xuất dữ liệu của cảm biến
- Công thức chuyển đổi giá trị cho cảm biến

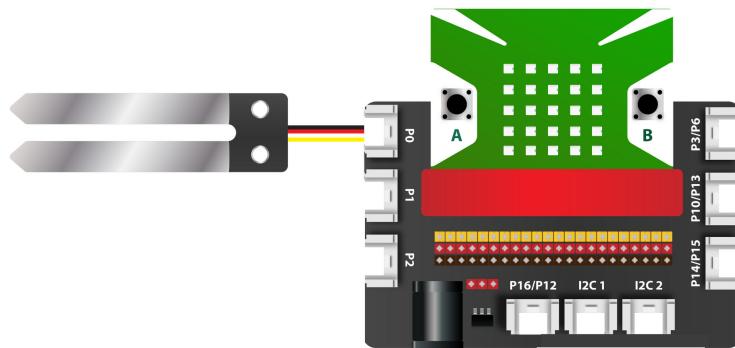
2 Kết nối với Yolo:Bit

Cảm biến độ ẩm đất thuộc dạng cảm biến có đầu ra là tín hiệu điện áp. Các cảm biến loại này đều dựa vào đặc điểm là khi điều kiện giám sát thay đổi, điện trở của nó sẽ thay đổi theo. Về lý thuyết, mọi thông tin về môi trường giám sát đều có thể thiết kế theo nguyên lý này, chẳng hạn như:

- Cảm biến đo nhiệt độ sử dụng linh kiện, mà điện trở của nó sẽ thay đổi khi nhiệt độ thay đổi. Linh kiện này còn gọi là **nhiệt điện trở**
- Cảm biến ánh sáng sẽ sử dụng **quang trở**, thiết bị có điện trở thay đổi khi cường độ ánh sáng thay đổi.

Đây là các cảm biến đơn giản nhất khi kết nối với các hệ thống vi điều khiển, bởi độ phức tạp của nó đã dồn qua việc thiết kế mạch, làm cho việc lập trình trở nên vô cùng dễ dàng. Tuy nhiên tất cả độ phức tạp lại chuyển sang cho mạch xử lý trung tâm là Yolo:Bit. Cụ thể, chỉ một số chân trên Yolo:Bit mới có khả năng kết nối được với cảm biến có đầu ra là điện áp.

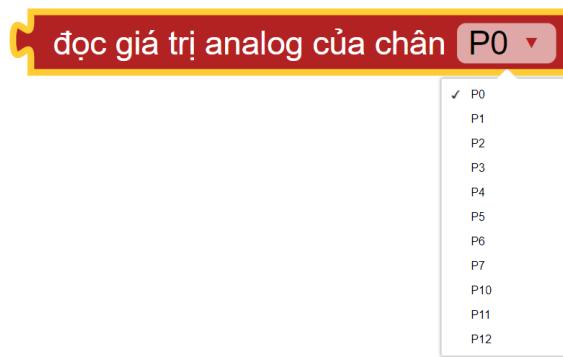
Hiện tại, với mạch mở rộng, chỉ các chân **P0**, **P1** và **P2** mới có thể kết nối được với cảm biến độ ẩm đất. Theo như kết nối ở hình dưới đây, cảm biến độ ẩm đất được nối với chân P0.



Hình 9.2: Kết nối với cảm biến độ ẩm đất

3 Lập trình đọc độ ẩm từ cảm biến

Với các cảm biến có đầu ra là điện áp, việc lập trình trên Yolo:Bit từ phía người sử dụng là vô cùng đơn giản. Chỉ với câu lệnh **đọc giá trị analog của chân**, như trình bày ở hình bên dưới, chúng ta có thể đọc dữ liệu từ cảm biến. Câu lệnh này được lấy trong mục **NÂNG CAO**, và chọn tiếp trong nhóm **CHÂN CẮM**.



Hình 9.3: 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 độ ẩm đất, đang được nối với chân P0 được trình bày như sau:



Hình 9.4: 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 cầm nó vào đất khô hay đất ướt. Thậm chí đơn giản hơn, là lấy tay cầm vào 2 cực của cảm biến. Do da tay của chúng ta có một độ ẩm nhất định, nên giá trị của cảm biến sẽ thay đổi khi chạm ngón tay vào cả 2 điện cực của nó.

4 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.

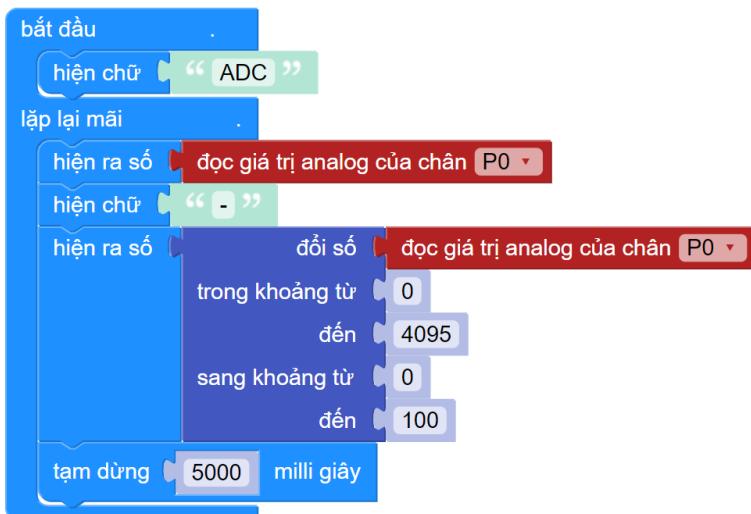
Tuy nhiên, như chúng ta đã biết, độ ẩm có đơn vị là phần trăm (%) và có tầm giá trị từ 0 đến 100 mà thôi. Điều quan trọng, và cũng là hạn chế của cảm biến ADC, là giá trị của nó không tuyến tính với mức điện áp. Do đặc điểm này, cảm biến ADC thường có độ chính xác vừa phải và giá thành cũng rẻ hơn so 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 độ ẩm đất trong nông nghiệp, chúng ta cũng không cần phải đo đặc 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 9.5: 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 9.5, 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:



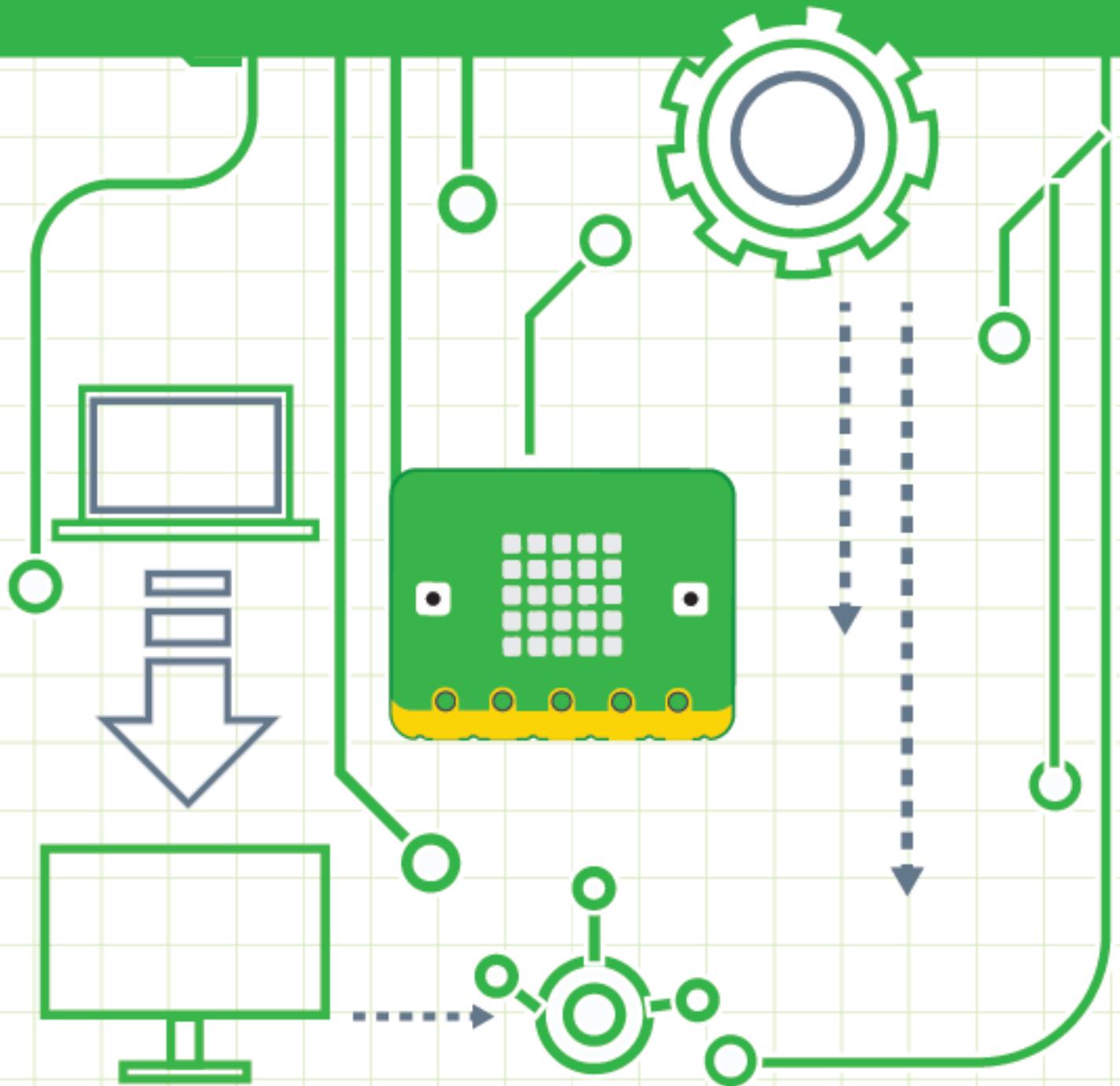
Hình 9.6: Chương trình chuyển đổi kết quả tính toán

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 thử cảm biến vào đất để kiểm tra tính đúng đắn của dữ liệu chuyển đổi. Với đất ẩm, giá trị độ ẩm thường sẽ cao và tiệm cận đến 100%.

Không chỉ độ ẩm đất, rất nhiều các cảm biến trên thị trường hỗ trợ chuẩn đầu ra là điện áp, chẳng hạn như cảm biến đo cường độ ánh sáng hay nồng độ CO₂. Bạn đọc có thể làm tương tự như trên để đổi sang đơn vị **lux** hoặc **ppm** nếu cần.

CHƯƠNG 10

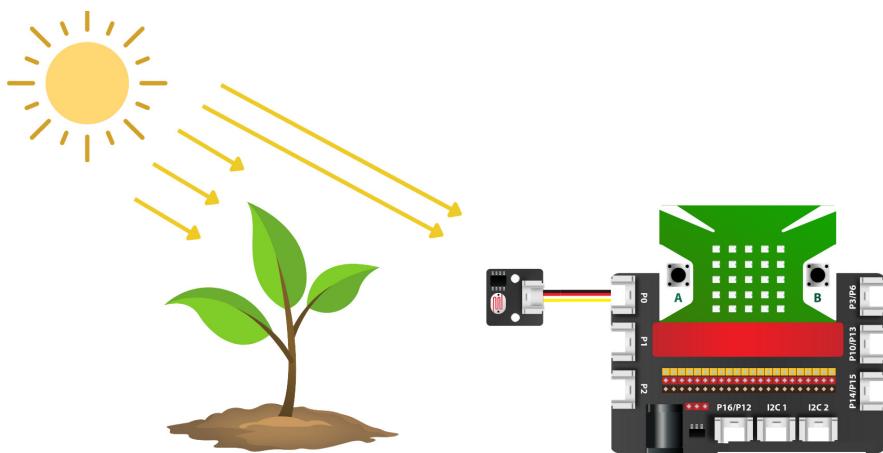
Cảm Biến Ánh Sáng



1 Giới thiệu

Đối với cây trồng, ánh sáng đóng vai trò vô cùng quan trọng. Không chỉ cung cấp nguồn năng lượng cho việc quang hợp của thực vật, ánh sáng còn có những ảnh hưởng khác nhau đến chu kỳ sinh trưởng của thực vật. Theo nghiên cứu của các nhà khoa học, ánh sáng xanh và tím sẽ kích thích sự tăng trưởng của thực vật, trong khi ánh sáng đỏ và cam sẽ kích thích sự ra hoa và đậu trái.

Trong nông nghiệp thông minh, để đo lường và tính toán thời gian thu hoạch của từng loại cây trồng, người ta đưa ra định nghĩa về thời gian tăng trưởng, viết tắt là GDD (Growing Degree Days). Đây là một chỉ số dựa trên cường độ ánh sáng mà cây trồng thu được để đánh giá ngày tuổi của nó. Dựa vào thông tin này, chúng ta sẽ ước lượng được chính xác hơn thời gian tăng trưởng của cây trồng trong nhiều điều kiện thời tiết khác nhau, để có kế hoạch thu hoạch hợp lý hơn.



Hình 10.1: Tính GDD cho cây trồng dựa vào cảm biến ánh sáng

Chúng tôi sẽ trình bày một ví dụ để minh họa cho ý tưởng của thông số GDD được dùng ứng dụng trong nông nghiệp thông minh. Một loại cây trồng sẽ sinh trưởng tốt trong điều kiện có nắng trên 500 đơn vị ánh sáng (lux). Với điều kiện này, cây trồng sẽ được thu hoạch sau 20 ngày. Trung bình mỗi ngày, chúng ta có 8 giờ có nắng tốt. Như vậy, cây trồng cần 160 giờ có nắng tốt là đủ tăng trưởng và có thể thu hoạch.

Trong quá trình sinh trưởng của cây trồng, điều kiện thời tiết thay đổi, chẳng hạn như trời nhiều mây và ít nắng hơn thông thường. Khi sự cố này xảy ra, thời gian tăng trưởng của cây trồng sẽ **không được cộng dồn vào chỉ số GDD**. Như vậy, cây trồng sẽ được thu hoạch chậm hơn bình thường một khoảng thời gian, cho đến khi chỉ số GDD được tích lũy đủ thời gian cho nó, trong trường hợp này là 160 giờ. Trong trường hợp ngược lại, khi điều kiện thời tiết tốt hơn, trời có nắng nhiều hơn, thời gian thu hoạch sẽ ngắn lại, khi một ngày, cây trồng đang nhận được nắng lâu hơn so với bình thường.

Ý tưởng của chỉ số GDD có thể được mở rộng thêm với nhiều chỉ số khác, như độ ẩm không khí, lượng mưa hay độ ẩm đất. Khi kết hợp với nhiều chỉ số, công thức

tính của GDD sẽ phức tạp hơn, nhưng tính chính xác của ngày tăng trưởng cũng sẽ chính xác hơn. Trong hướng dẫn ở bài này, chúng tôi xây dựng công thức tính cho GDD bằng cách chỉ sử dụng cảm biến ánh sáng. Các mục tiêu chính trong bài hướng dẫn này như sau:

- Xây dựng công thức tính GDD dựa vào cảm biến ánh sáng
- Lập trình đọc giá trị từ cảm biến ánh sáng
- Lập trình ứng dụng GDD vào Yolo:Bit

2 Công thức GDD

Theo như nguyên lý tích lũy của GDD, chúng ta có thể mô hình hóa GDD bằng công thức toán học sau đây:

$$GDD = GDD + \begin{cases} 1 & LUX \geq 2000 \\ 0 & LUX < 2000 \end{cases}$$

Trong đó:

- GDD: Thời gian tăng trưởng tích lũy
- LUX: Giá trị đọc từ cảm biến ánh sáng

Chúng ta chọn 2000 là ngưỡng giá trị để quyết định là điều kiện ánh sáng hiện tại đã đủ cho việc tích lũy GDD hay chưa. Giá trị này bạn đọc cần khảo sát bằng kĩ thuật ở bài trước để chọn ra một ngưỡng so sánh cho phù hợp.

Đơn vị của GDD phục thuộc vào chu kì mà hệ thống kiểm tra cảm biến ánh sáng. Chẳng hạn như hệ thống sẽ kiểm tra mỗi phút, thì GDD sẽ có đơn vị là phút.

3 Hiện thực trên Yolo:Bit

Trình tự để hiện thực chương trình tính GDD trên Yolo:Bit được trình bày chi tiết như bên dưới.

Bước 1: Tạo 2 biến số GDD và LUX.



Hình 10.2: Trình tự tạo biến GDD

Từ nhóm lệnh **BIẾN**, nhấn vào nút **Tạo biến**, đặt tên và nhấn nút **LƯU**. Bạn đọc cần lặp lại quy trình này để tạo ra 2 biến số cho chương trình, là **GDD** và **LUX**.

Bước 2: Xây dựng cấu trúc cho chương trình.

Bước tiếp theo, chúng ta sẽ khởi tạo giá trị cho 2 biến này, cũng như thiết lập chu kì một phút trong phần **lặp mãi mãi**, như sau:



Hình 10.3: Khởi tạo giá trị cho biến số

Các khối lệnh chứa số 0 có thể được tìm thấy trong phần **TÍNH TOÁN**.

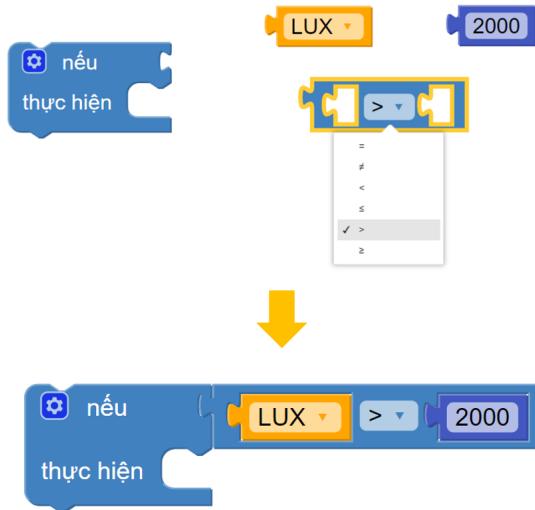
Bước 3: Đọc giá trị từ cảm biến ánh sáng và lưu vào LUX.

Theo kết nối ở Hình 10.1, cảm biến ánh sáng được nối với chân P0. Chương trình để đọc giá trị cảm biến này như sau:



Hình 10.4: Đọc giá trị từ cảm biến ánh sáng

Bước 4: Xây dựng câu điều kiện kiểm tra điều kiện ánh sáng dựa trên biến LUX.

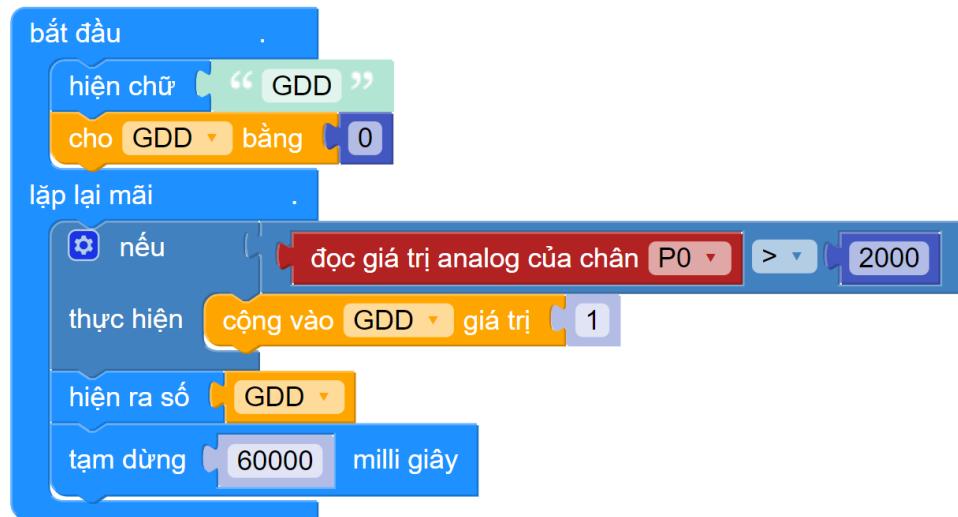


Hình 10.5: Hiện thực khối điều kiện so sánh

Ở trạng thái mặc định, khôi lệnh so sánh trong nhóm **LOGIC** sẽ là phép **so sánh bằng**. Chúng ta có thể thay đổi phép so sánh bằng cách chọn lại trong danh sách hỗ trợ, như minh họa ở hình bên trên.

Bước 5: Hoàn thiện chương trình.

Cuối cùng, chúng ta sẽ xử lý tính toán trên biến số GDD và hiện kết quả nó ra sau mỗi một phút. Chương trình gợi ý cho phần này sẽ như sau:



Hình 10.6: Hoàn thiện chương trình tính GDD

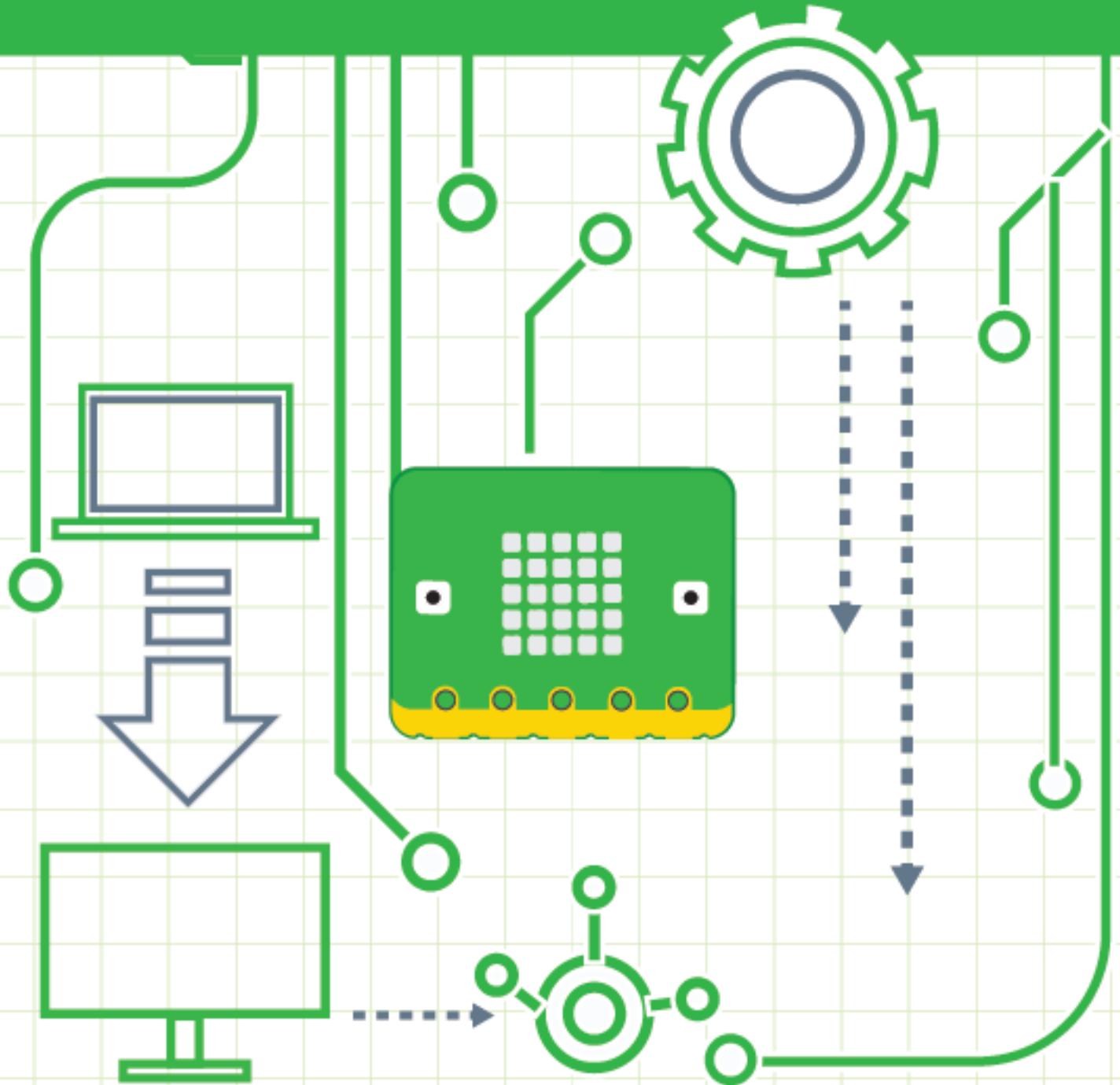
Bạn đọc có thể tự bổ sung thêm tính năng báo hiệu đèn xanh khi cây trồng đã đủ thời gian tăng trưởng GDD và không cộng dồn giá trị GDD nữa.

Phần III

Kết Nối Vạn Vật với OhStem

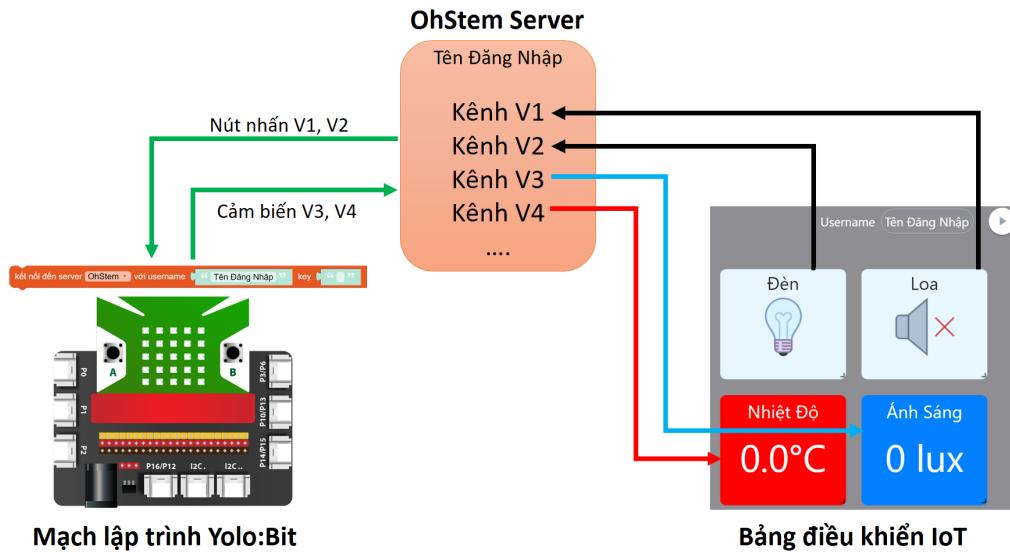
CHƯƠNG 11

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 11.1: 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 11.1, 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 11.1 là **V3 cho nhiệt độ** và **V4 cho ánh sáng**.

3 Thiết kế bảng điều khiển

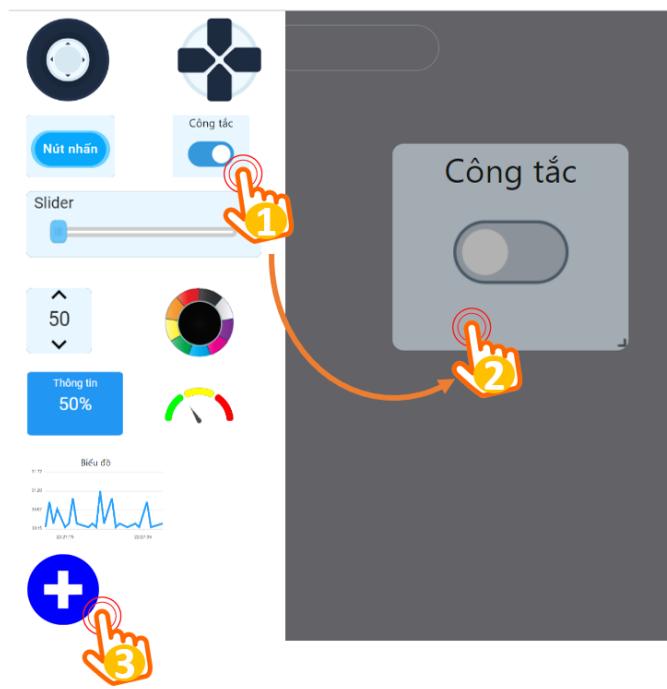
Trong hướng dẫn tiếp theo, chúng ta sẽ bắt đầu thiết kế một giao diện điều khiển đơn giản với 2 nút nhấn. Trình tự để thực hiện bảng điều khiển được trình bày chi tiết bên dưới.

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 11.2: 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 11.3: 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 11.3 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.



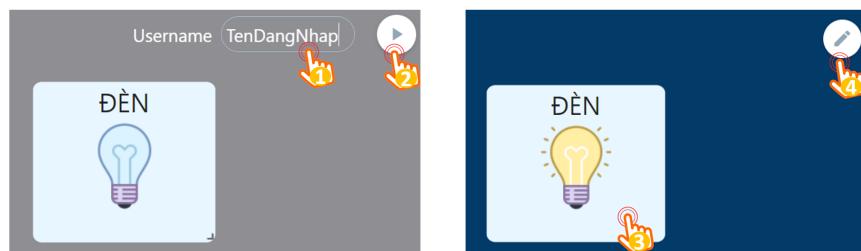
Hình 11.4: Cấu hình cho đối tượng công tắc

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 11.4, 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.
- Ảnh 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.

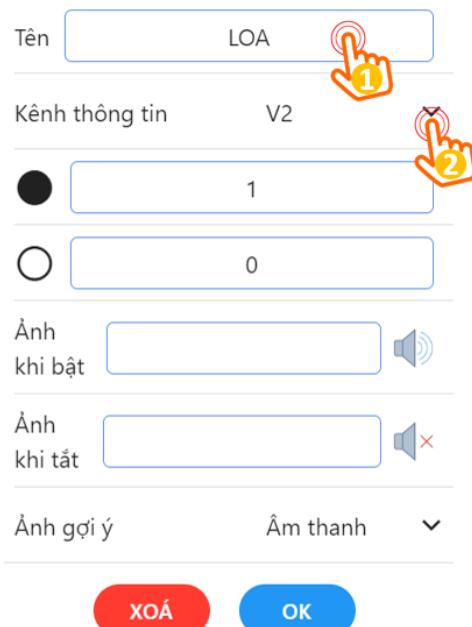


Hình 11.5: Đặt tên cho màn hình điều khiển

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.

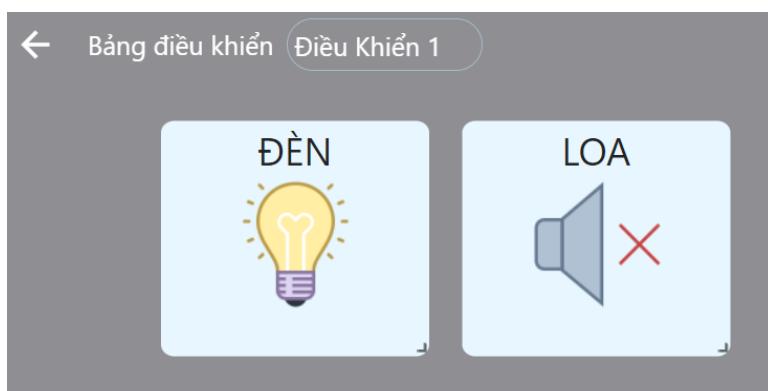
Bây giờ, bạn đọc có thể nhấn vào biểu tượng ở góc bên phải để chạy màn hình điều khiển (kí hiệu số 2 trên Hình 11.5), tương tác trên nút nhấn **ĐÈN** để thấy các hiệu ứng thay đổi mỗi khi nhấn vào nó. Cuối cùng, để trở lại màn hình thiết kế, chúng ta nhấn vào biểu tượng thay đổi ở góc bên phải (kí hiệu số 4 trên Hình 11.5).

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.



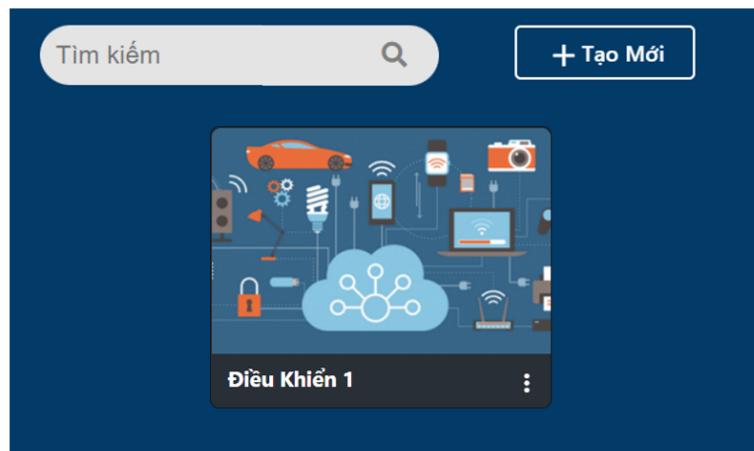
Hình 11.6: Tạo công tắc mới với kênh V2

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 11.7: 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.

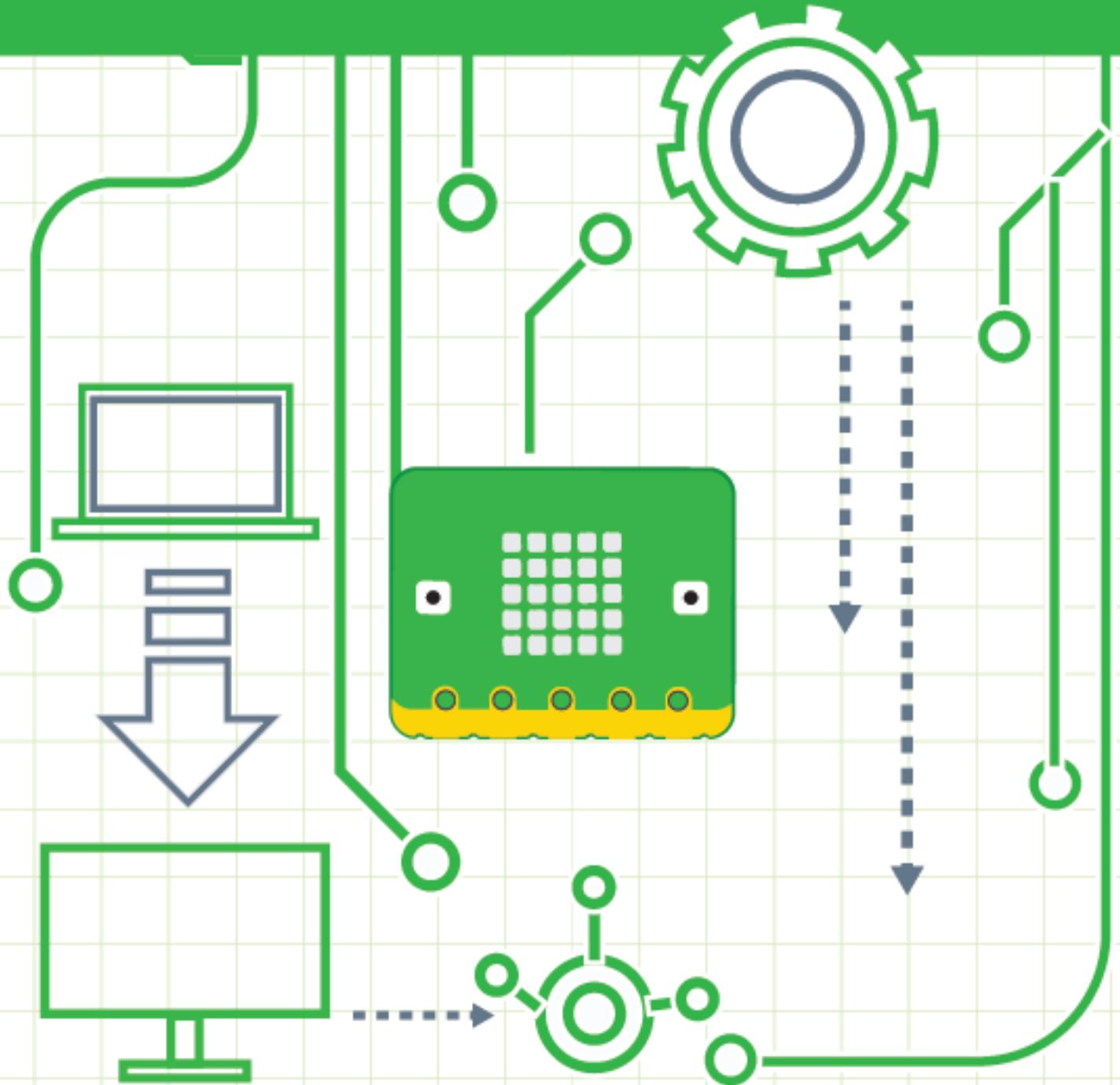


Hình 11.8: Các bảng điều khiển đã thiết kế

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.

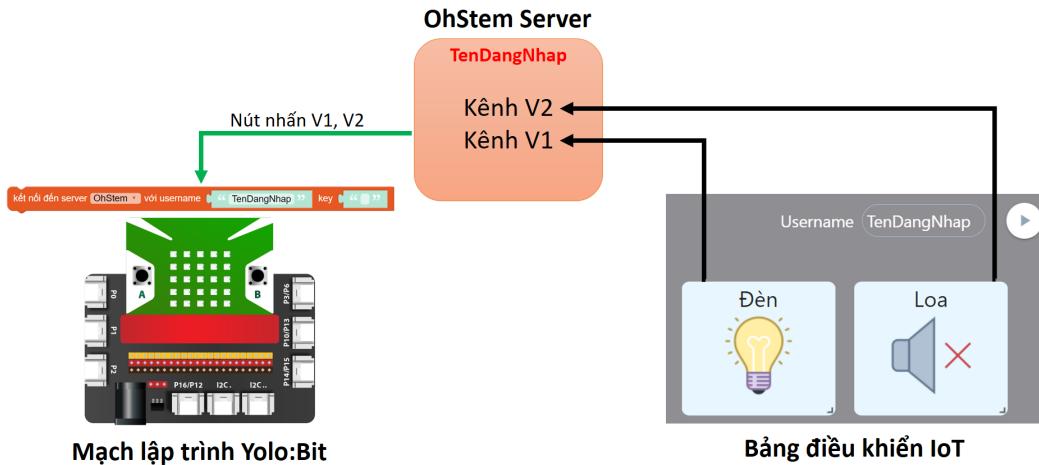
CHƯƠNG 12

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 12.1: 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.**

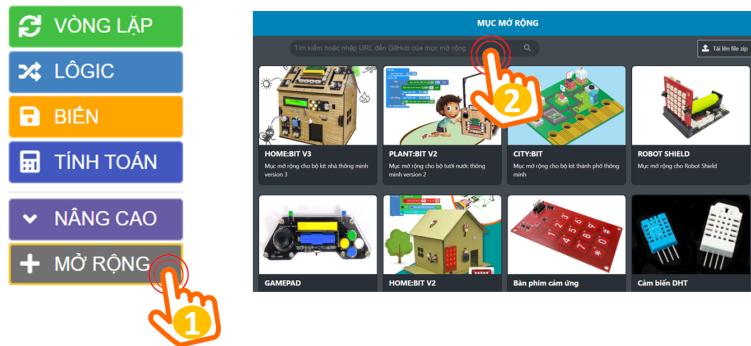
Thực ra, cơ chế đăng ký nhận dữ liệu (subscribe) đã tồn tại rất nhiều trong các ứng dụng mà chúng ta đang sử dụng hiện tại. Ví dụ: Mỗi khi bạn đăng ký một kênh trên Youtube, bạn sẽ tự động nhận được thông báo từ kênh đó. Bên dưới mạch Yolo:Bit đã thực hiện một cơ chế tương tự như việc bạn đăng ký 1 kênh 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

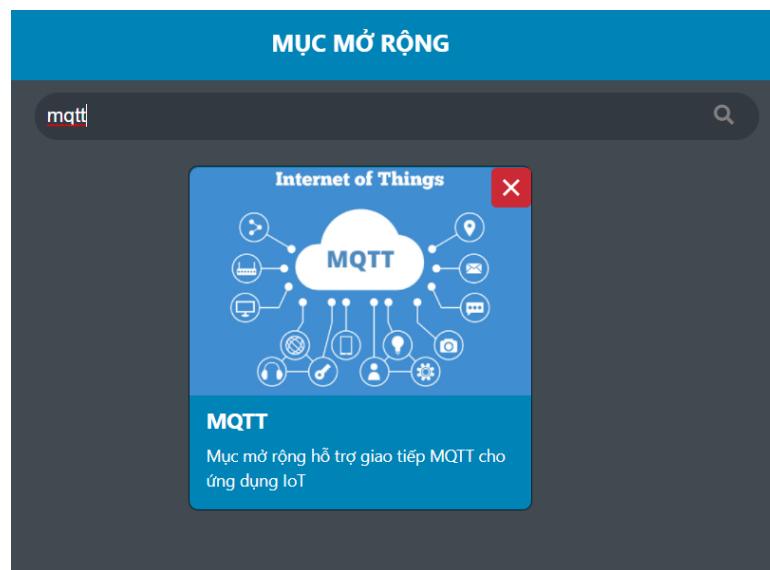
Để mạch Yolo:Bit có thể kết nối WiFi và thực hiện cơ chế nhận dữ liệu với OhStem server, chúng ta cần tải thêm thư viện lập trình mở rộng MQTT cho nó. Các bước để thêm thư viện khôi lệnh này được trình bày chi tiết bên dưới:

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 12.2: 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 12.3: 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

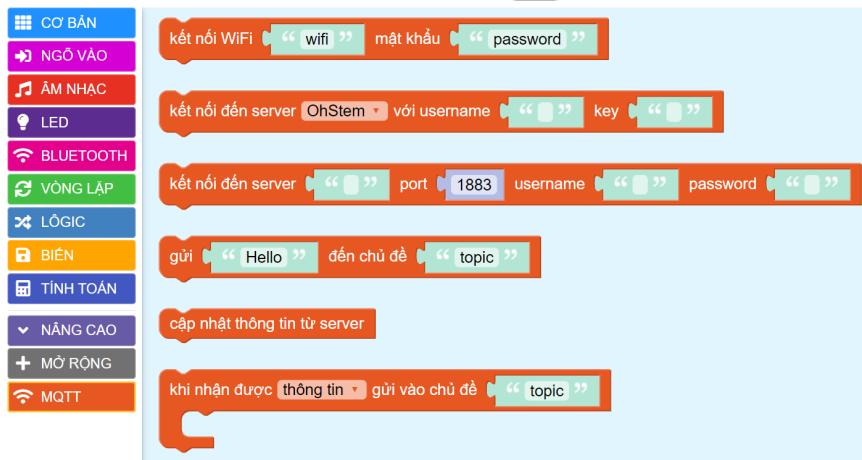
Mục mở rộng này cần thư viện đi kèm để sử dụng được. Bạn có muốn tải thư viện (thiết bị cần được kết nối)?

BỎ QUA

OK

Hình 12.4: 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 12.5: 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 12.6: 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 12.7: 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



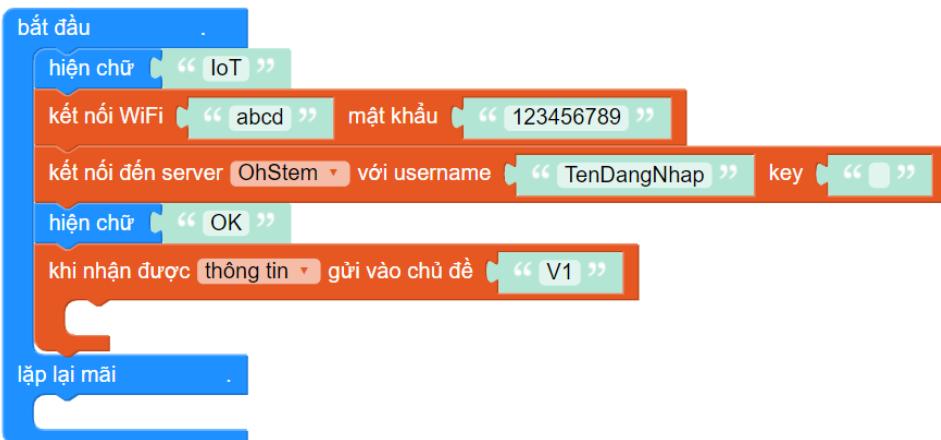
Hình 12.8: Đăng nhập vào OhStem server

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:

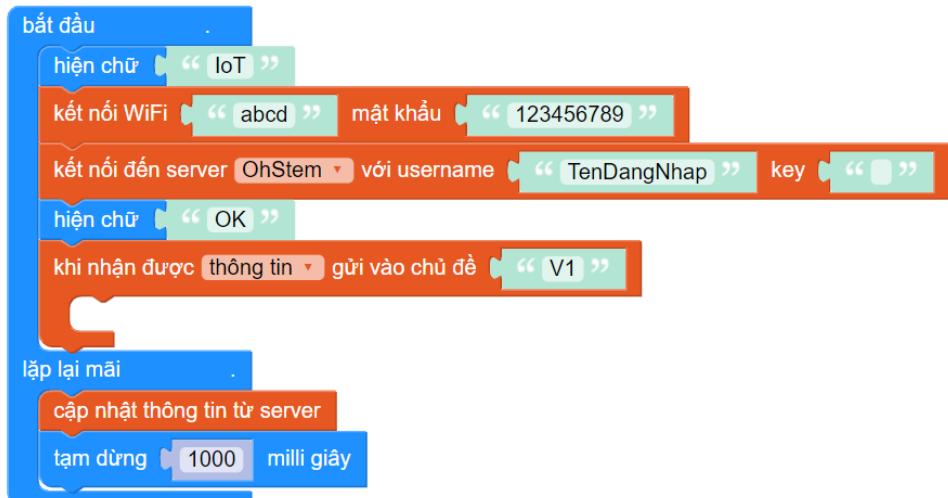


Hình 12.9: Đăng ký kênh nhận dữ liệu

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 12.10: 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 12.11: 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:





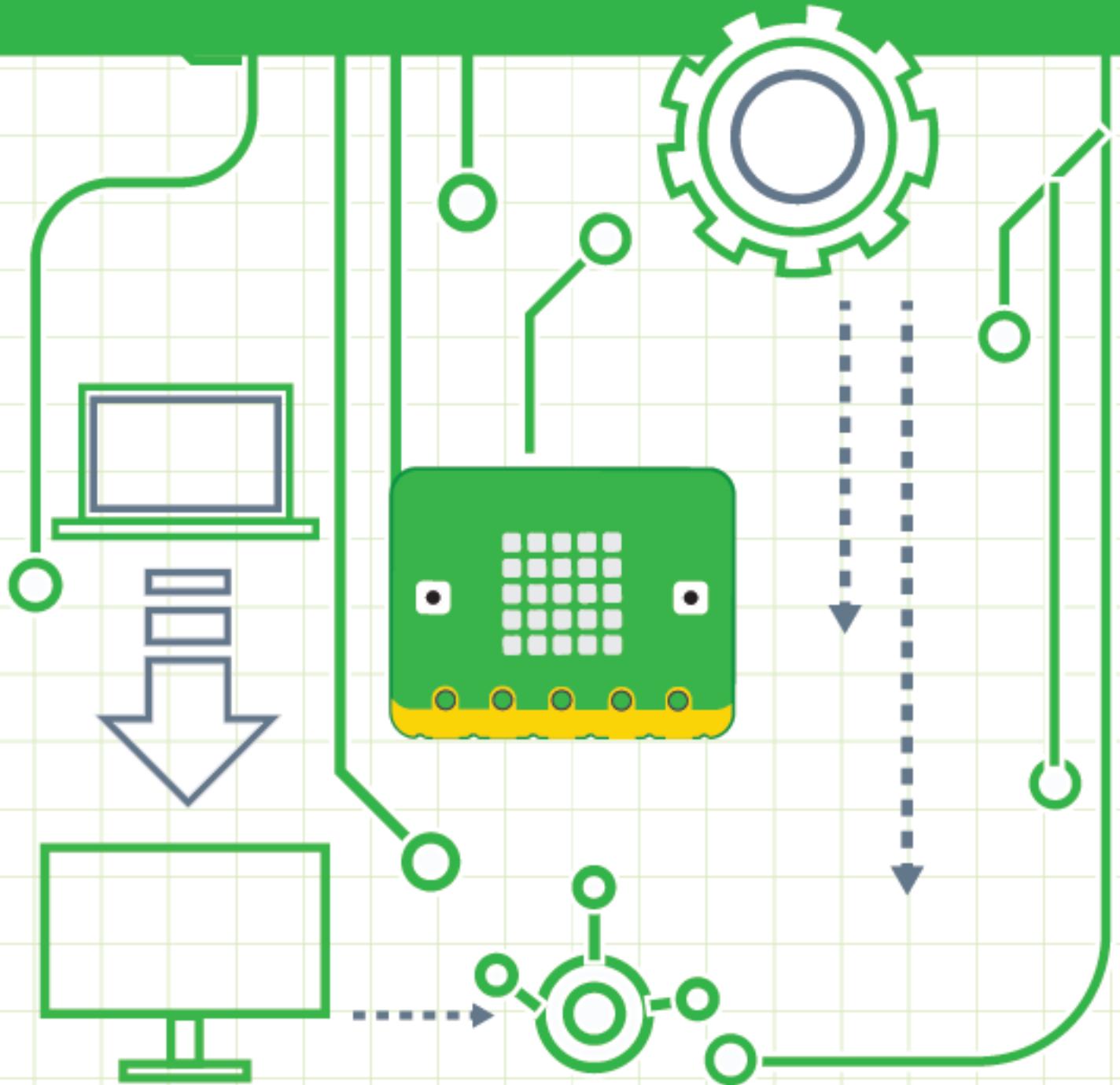
Hình 12.13: Chương trình đăng ký và xử lý 2 kênh dữ liệu

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>

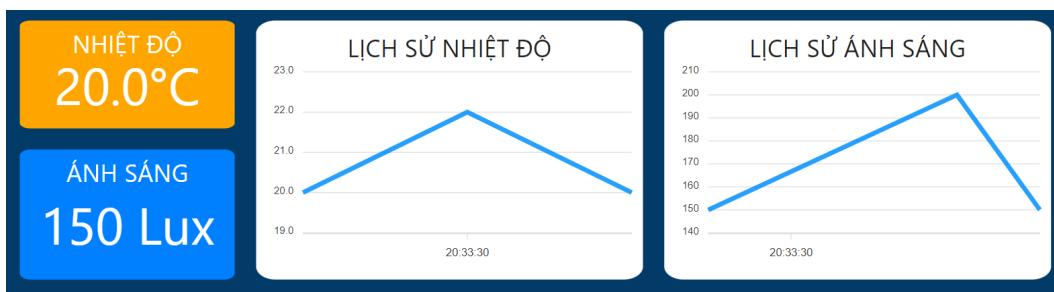
CHƯƠNG 13

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 13.1: Giao diện quan trắc thông tin từ thiết bị

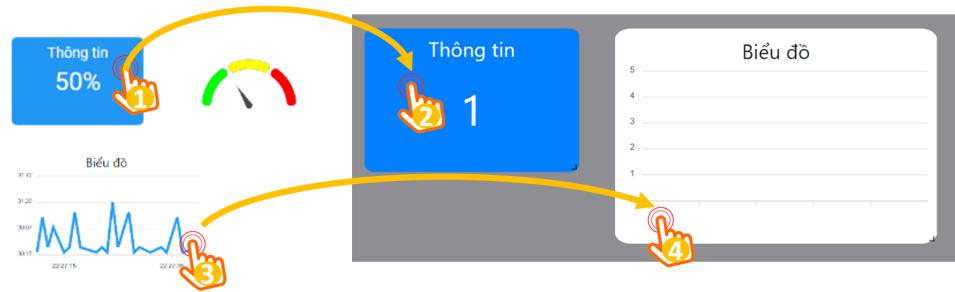
Trong bài hướng dẫn này, chúng ta sẽ thiết kế trước một giao diện trên màn hình để có thể giám sát trực quan nhất dữ liệu từ xa. Với những cảm biến hỗ trợ sẵn trên Yolo:Bit, chúng tôi sẽ thiết kế giao diện để theo dõi thông tin về nhiệt độ và cường độ ánh sáng. Trong tương lai, chúng ta hoàn toàn có 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 13.2: Giao diện Thông tin và Đồ thi

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 13.3: 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.

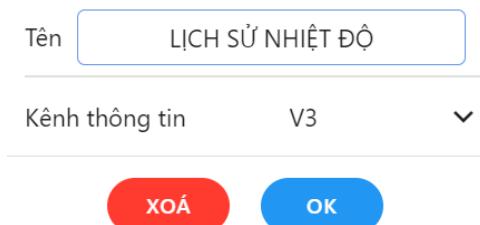
- Màu nền: Một số màu nền thông dụng cho bạn đọc lựa chọn cho đối tượng thông tin.

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 13.4: 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.

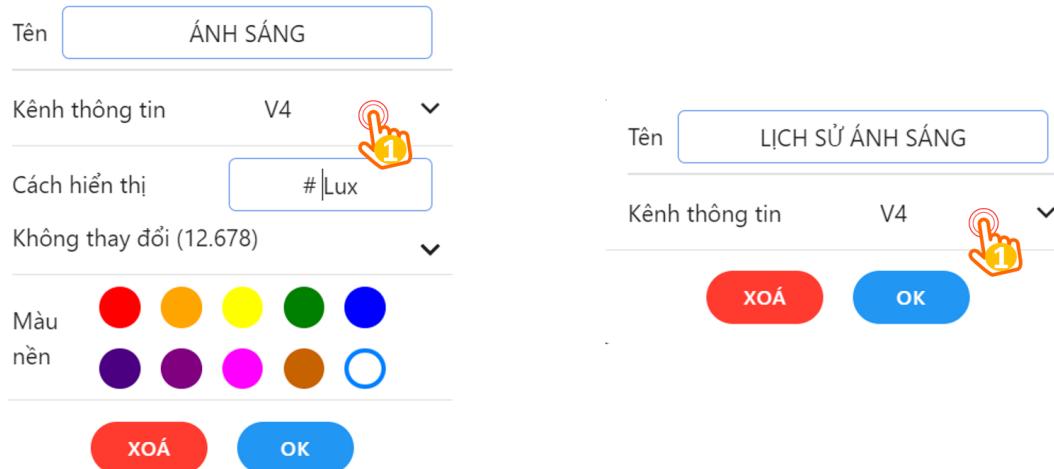
Cuối cùng, sắp xếp các giao diện một cách hợp lý và cung cấp thông tin cho trường Username, chúng ta sẽ có kết quả như sau:



Hình 13.5: Biểu diễn thông tin nhiệt độ trên bảng điều khiển

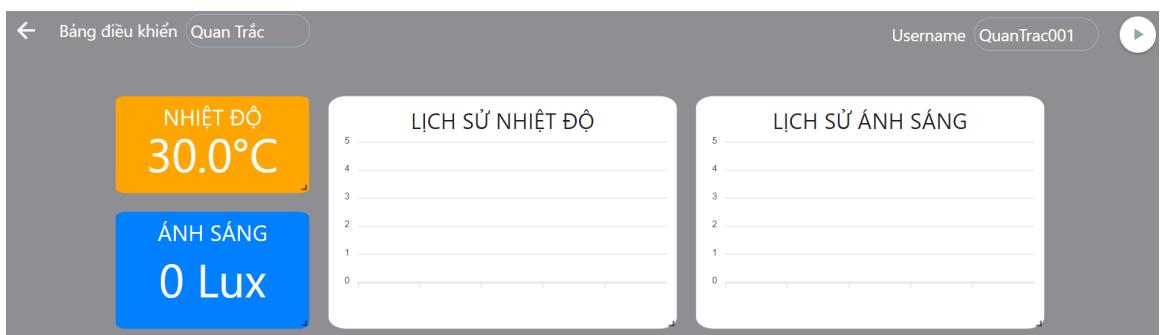
3 Hoàn thiện bảng giám sát

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 13.6: 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:

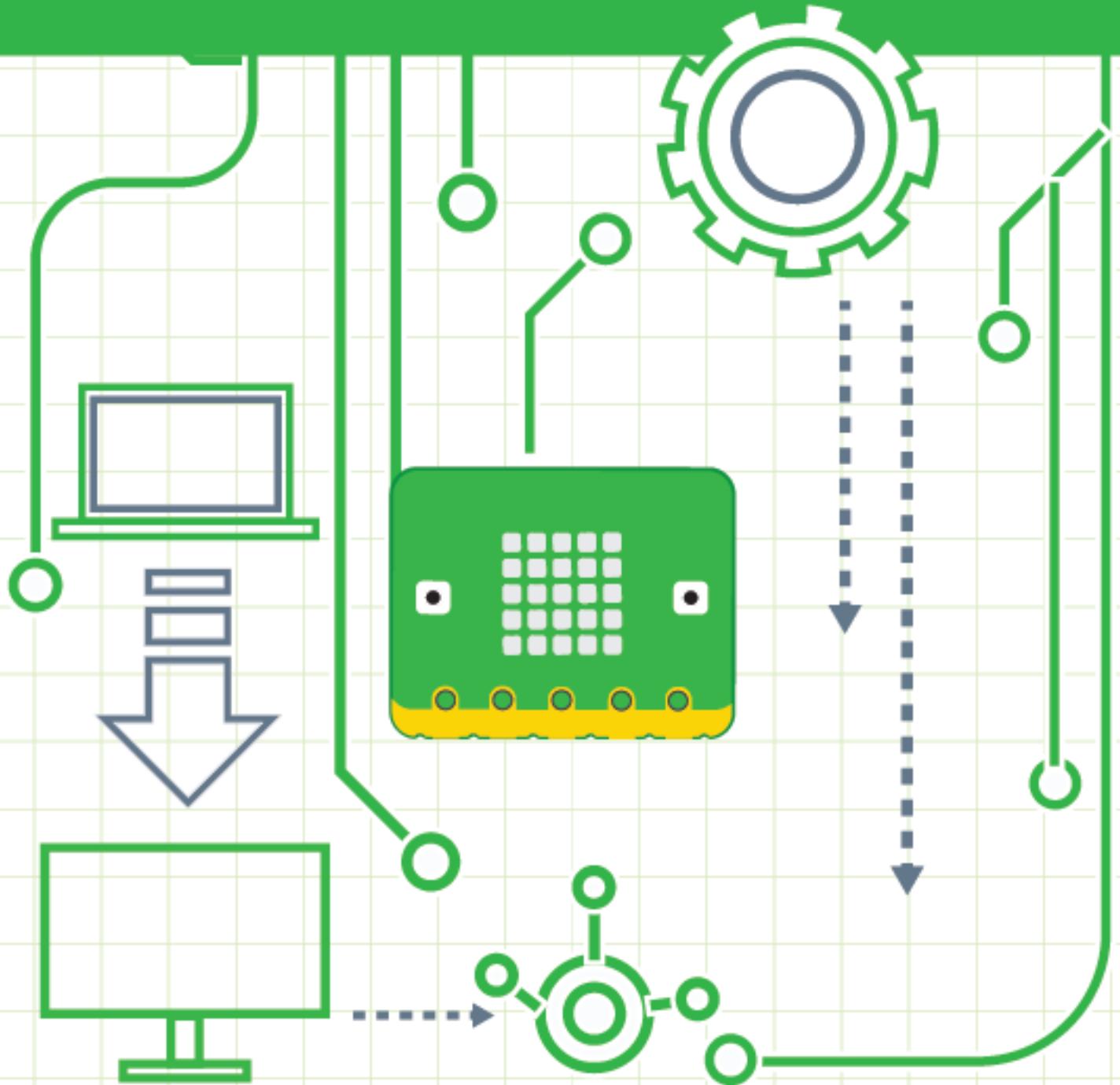


Hình 13.7: Giao diện quan trắc Nhiệt độ và Ánh sáng

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.

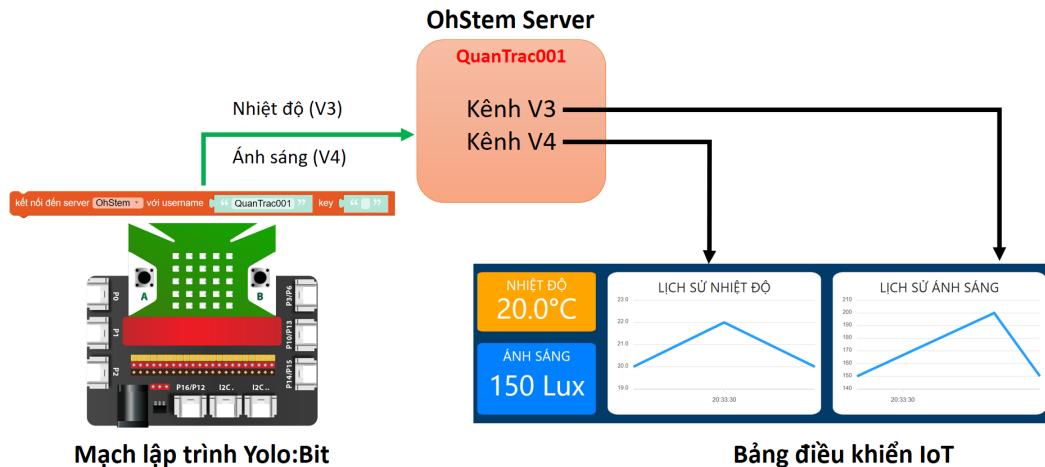
CHƯƠNG 14

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 14.1: 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.

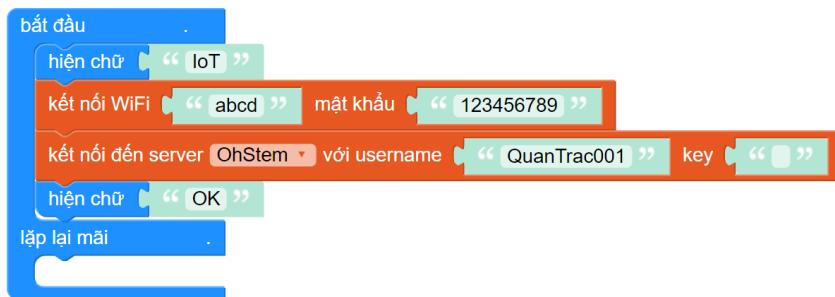
Trong bài hướng dẫn này, chúng tôi sẽ sử dụng cảm biến nhiệt độ có sẵn trên mạch Yolo:Bit để minh họa cho việc **gửi thông tin định kì lên Server** và theo dõi thông tin này trên đồ thị đã được thiết kế ở bài trước. Một khối lệnh mới sẽ được giới thiệu trong bài này, nhằm hỗ trợ trực tiếp cho việc gửi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

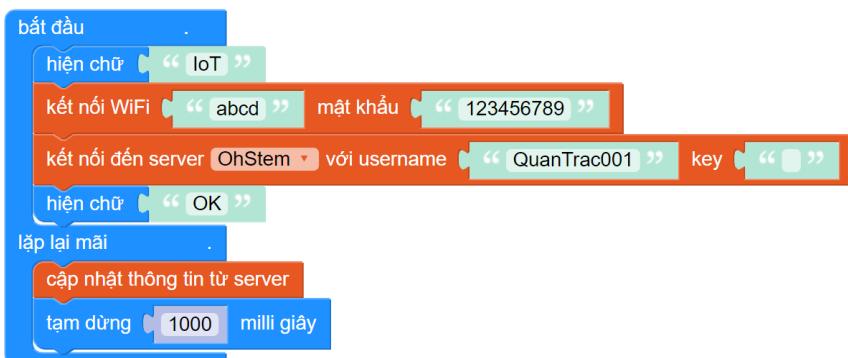


Hình 14.2: Kết nối mạng WiFi và OhStem server

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 14.3: 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

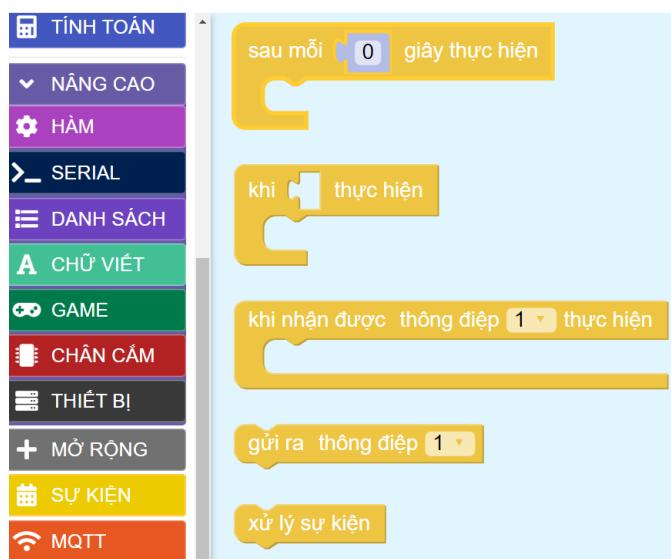
Sau khi cố định câu lệnh đợi 1 giây trong phần lặp mãi mãi, việc định kì gửi dữ liệu cảm biến lên server sẽ là trở ngại không nhỏ với bạn đọc. Lý do chính nằm ở chỗ chu kỳ cho việc gửi giá trị cảm biến thường là dài, khoảng 30 giây, 1 phút hoặc thậm chí lâu hơn, tùy theo ứng dụng. Do đó, để đơn giản 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:



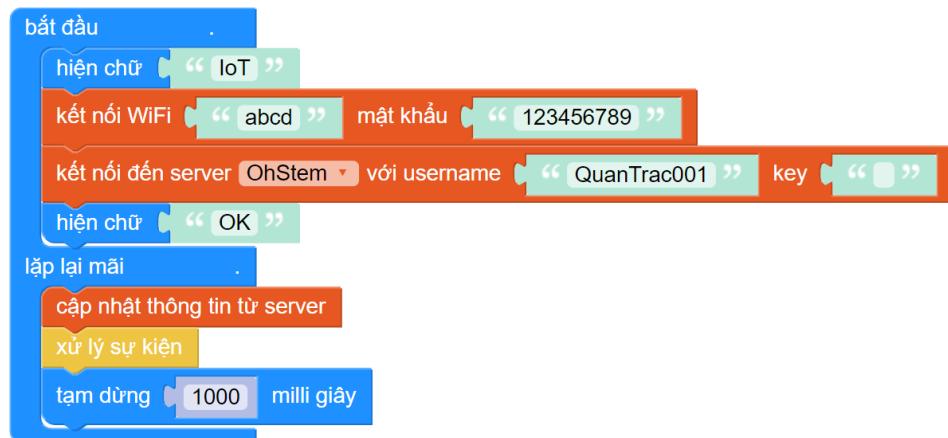
Hình 14.4: Thư viện lập trình SỰ KIỆN

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 14.5: Nhóm lệnh SỰ KIỆN được thêm vào

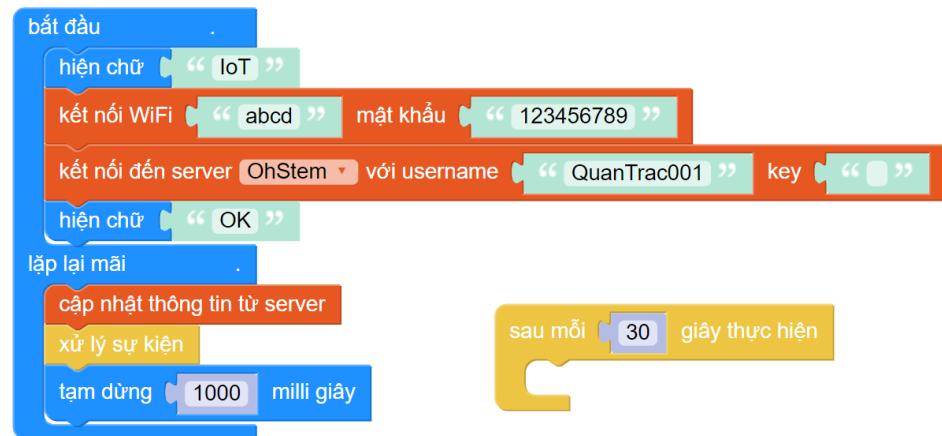
Bước 2: Thêm khối lệnh **xử lý sự kiện** vào phần lặp mãi mãi của chương trình, như sau:



Hình 14.6: 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 14.7: 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

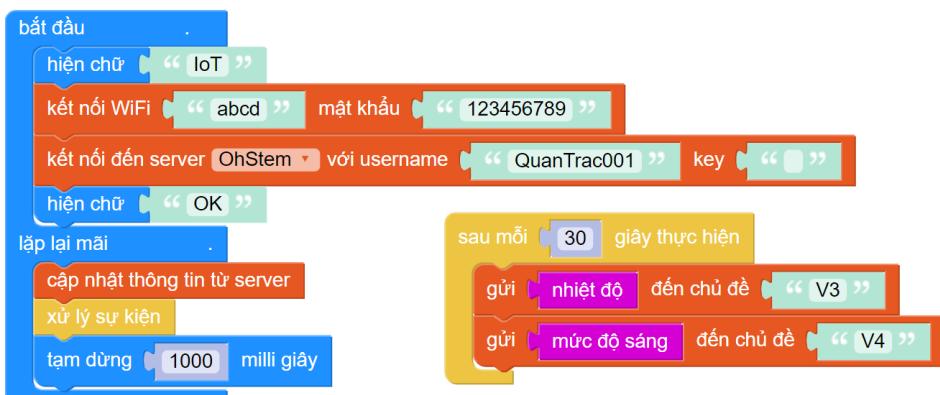
Sau khi đã có khôi lệnh xử lý theo chu kỳ, việc gửi dữ liệu lên OhStem server sẽ được đơn giản hóa. Các khôi lệnh chính để gửi dữ liệu nhiệt độ và ánh sáng lên OhStem server được trình bày như sau:



Hình 14.8: 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:



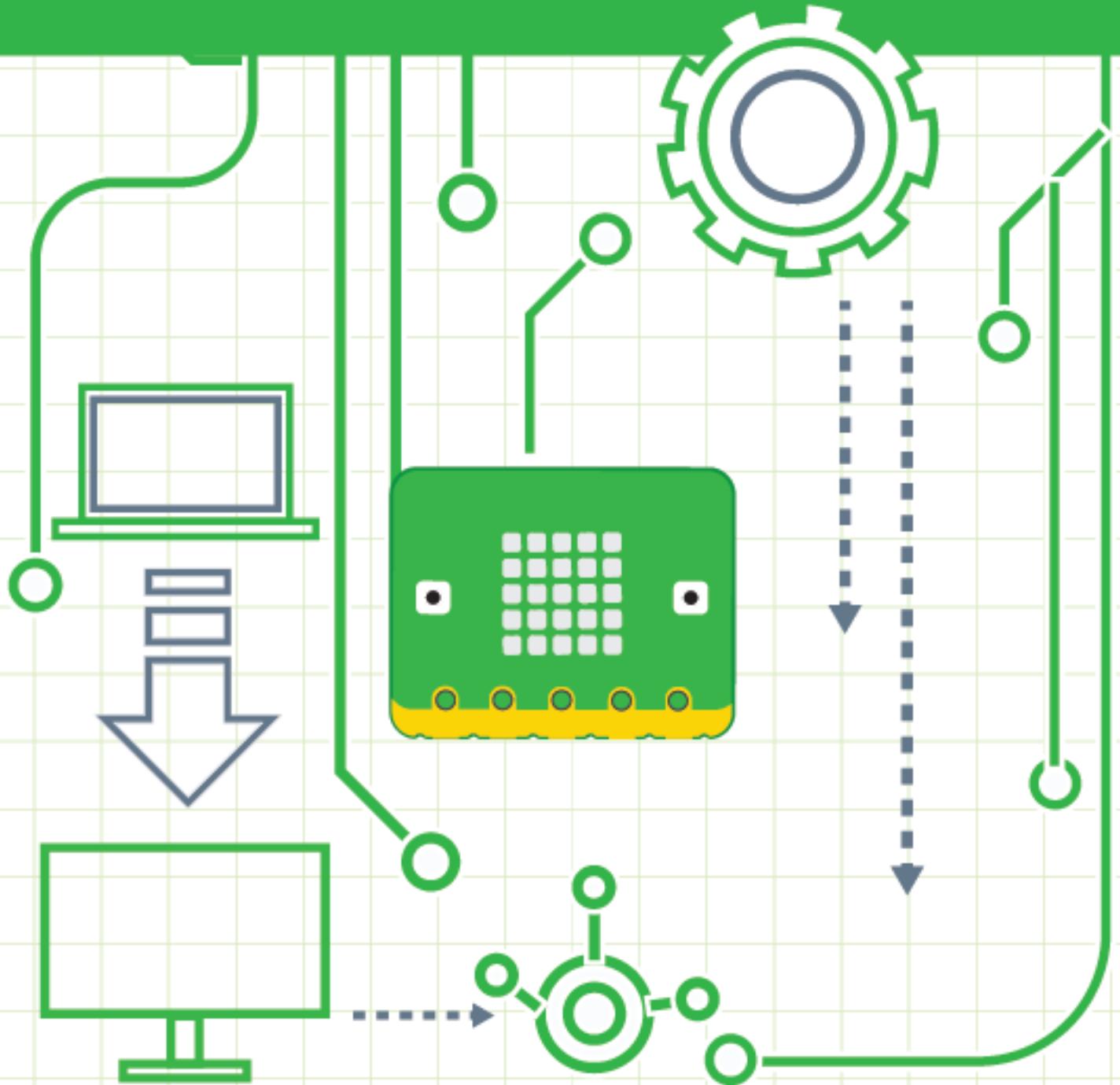
Hình 14.9: Hoàn thiện chương trình

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>

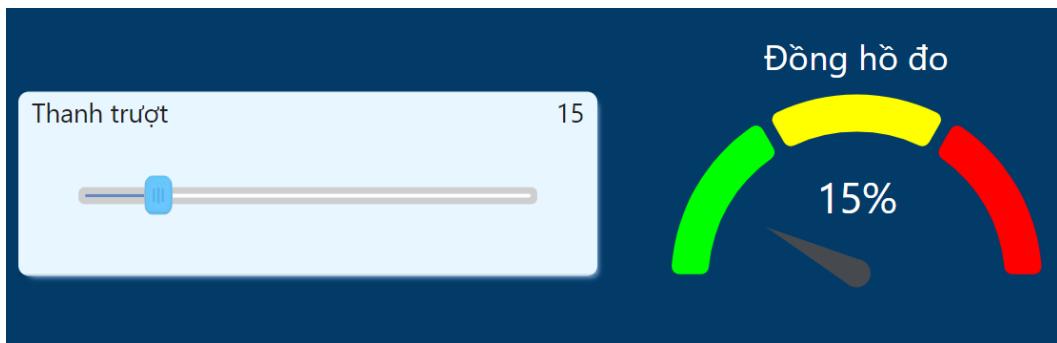
CHƯƠNG 15

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 15.1: Các giao diện nâng cao

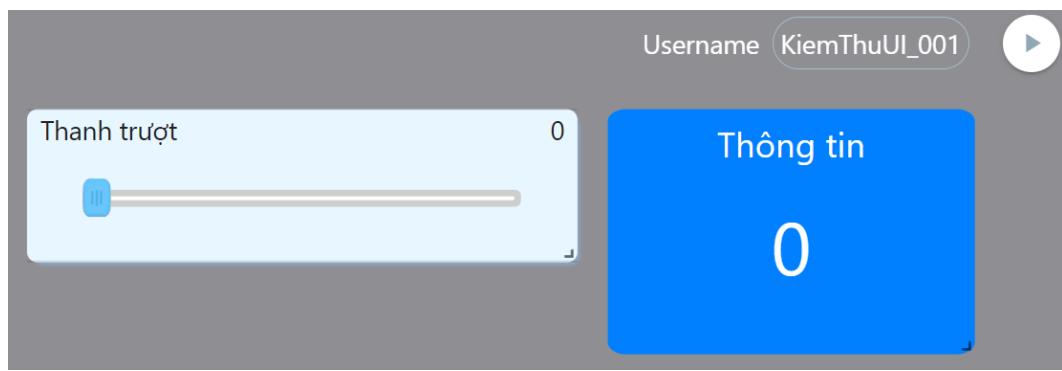
Kỹ năng quan trọng trong bài hướng dẫn này là cung cấp cho bạn đọc 1 công cụ để kiểm thử đối tượng giao diện mới, trước khi bắt đầu lập trình trên mạch Yolo:Bit. Điều này sẽ giúp bạn đọc có thể tự tìm hiểu và giảm sai sót khi lập trình.

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

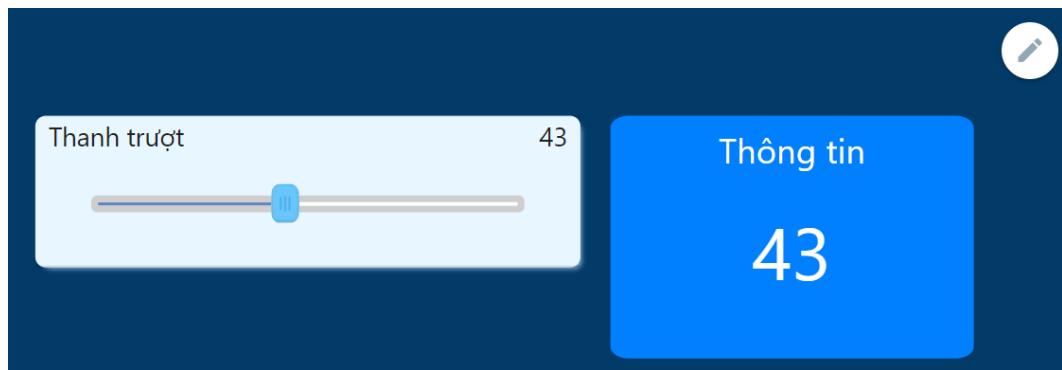
Để bắt đầu tìm hiểu một đối tượng giao diện, bạn nên tạo mới một màn hình điều khiển IoT. Tuy nhiên, hãy kéo kèm nó ra màn hình với đối tượng giao diện **Thông tin**, như minh họa dưới đây:



Hình 15.2: 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 15.3: Chạy thử bảng điều khiển IoT

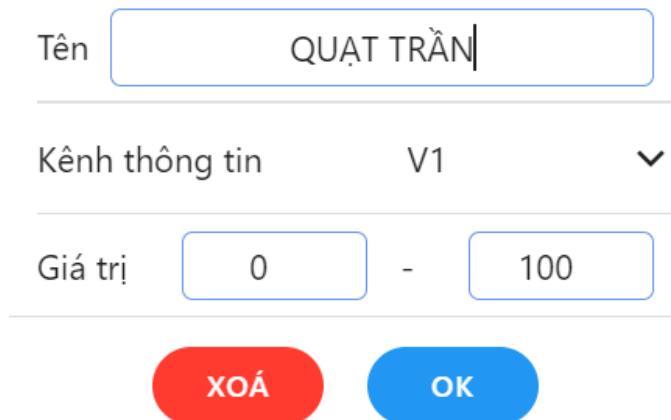
Điều này được giải thích như sau:

- 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 15.4: Cấu hình cho Thanh trượt

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

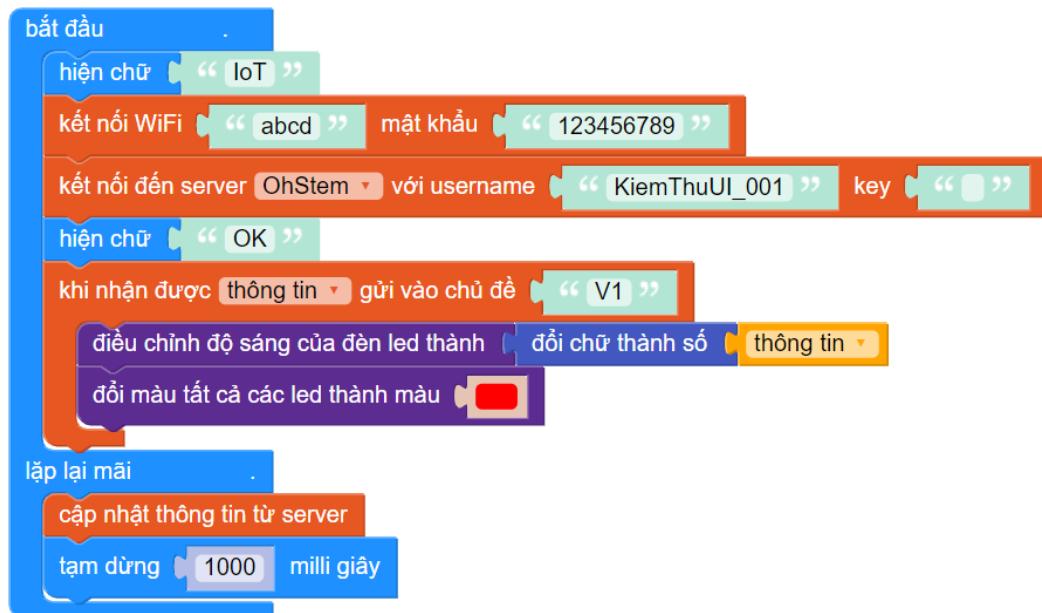
- Tên: là giá trị sẽ hiện như tiêu đề của Thanh trượt. Mặc dù không phải là thông tin dùng cho việc lập trình, nó lại là thông tin quan trọng để trình diễn. Bạn đọc sẽ thường thay đổi thông tin này để nó gần gũi nhất với thiết bị cần điều khiển ở mạch Yolo:Bit.
- Kênh dữ liệu: luôn luôn là thông tin 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 15.5: 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.

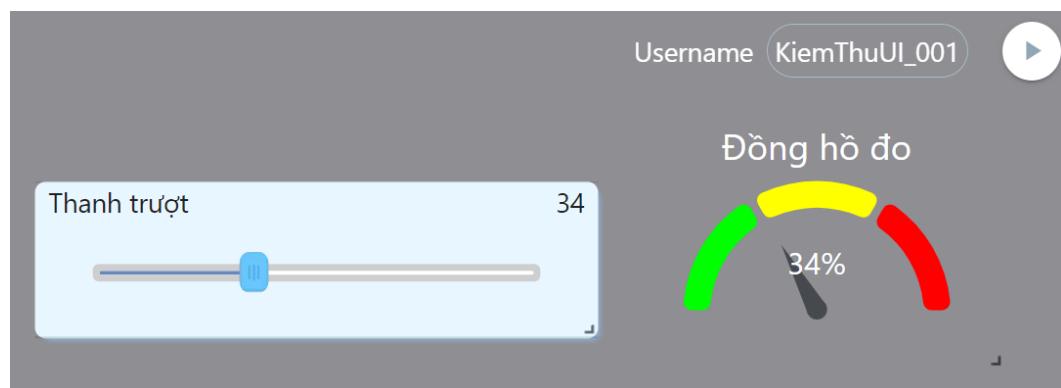


Hình 15.6: Chương trình xử lý trên Yolo:Bit

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.

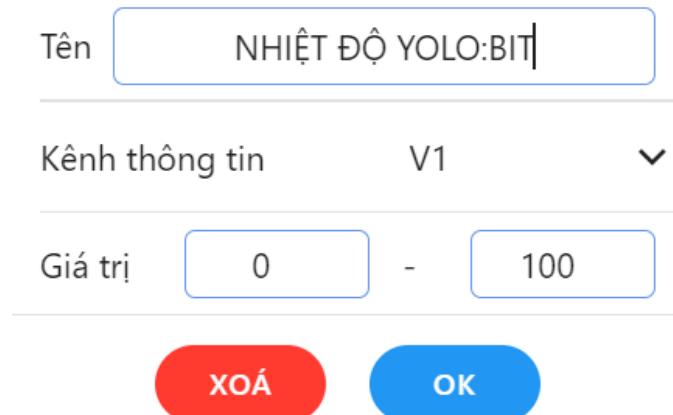
4 Giao diện Đồng hồ đo

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 15.7: 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:

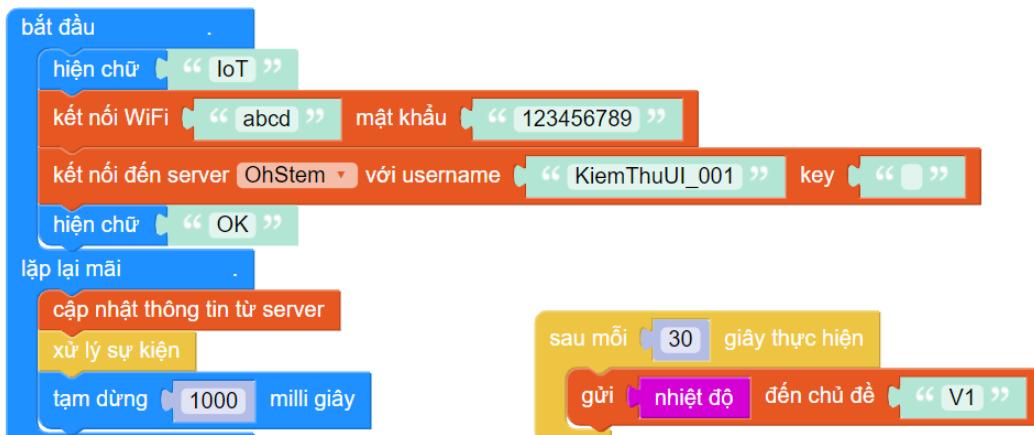


Hình 15.8: Cấu hình cho Đồng hồ đo

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 15.9: 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 15.10: 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.

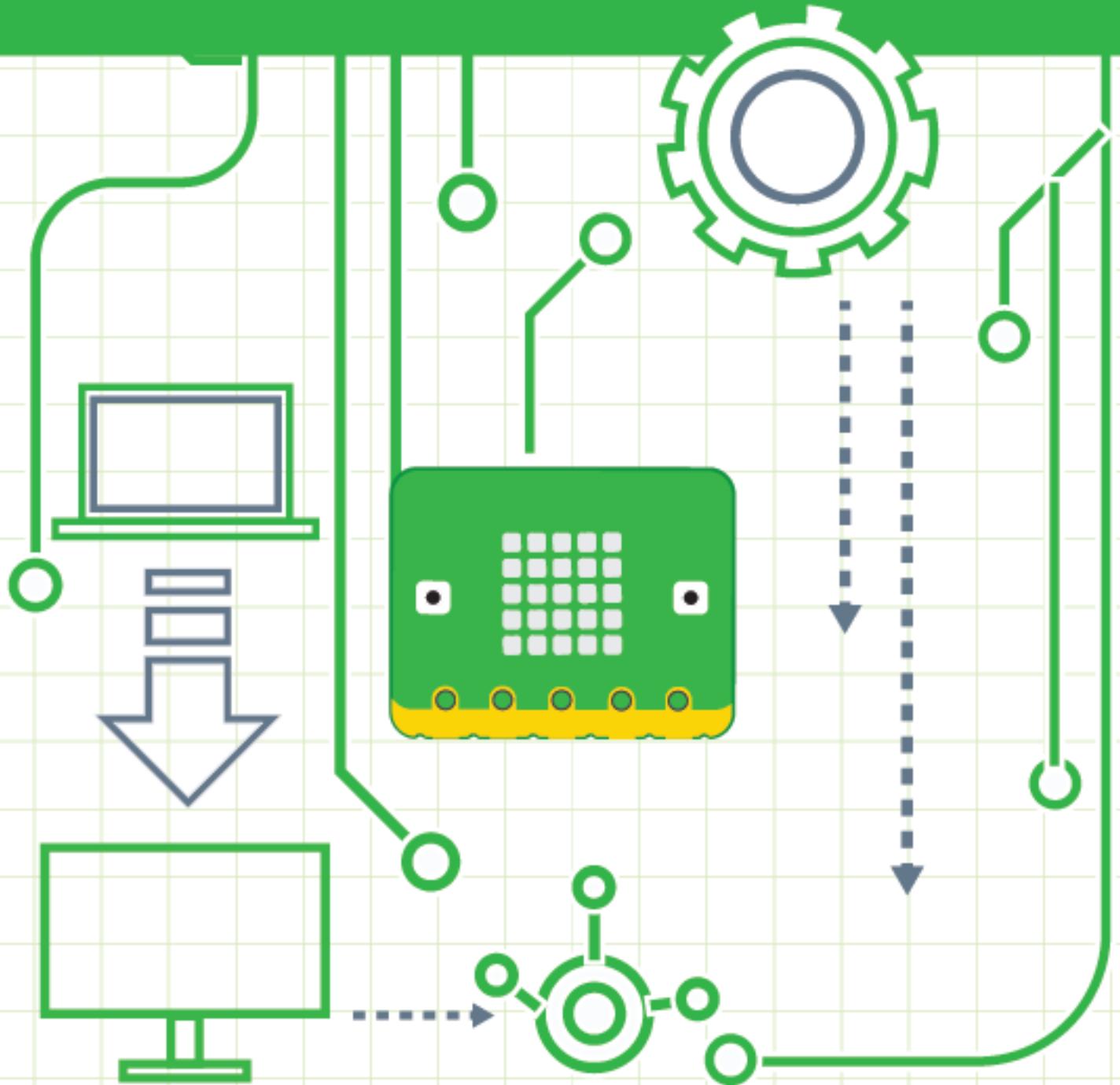
Kết luận thứ 2 sẽ là thông tin quan trọng mà chúng ta sẽ sử dụng trong tương lai. Chẳng hạn như mạch Yolo:Bit đang nắm quyền bật một máy bơm (cho tính năng tưới tiêu theo giờ chẵng hạn), nó sẽ gửi lên server thông tin về trạng thái của máy bơm (là 1 hoặc 0). Lúc này, nút nhấn vốn được thiết kế để điều khiển máy bơm 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.

Phần IV

Trí Tuệ Nhân Tạo

CHƯƠNG 16

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**.

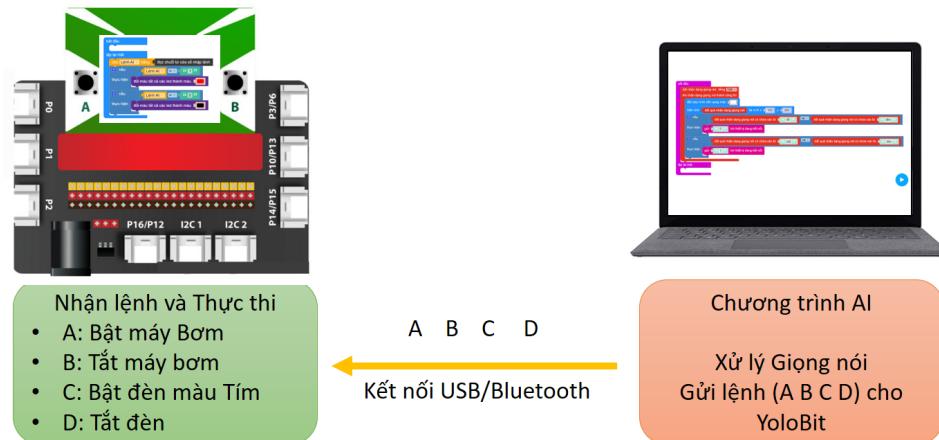
Trong bài hướng dẫn này, bên cạnh việc sử dụng công cụ chuyển đổi văn bản thành giọng nói, được hỗ trợ trong hệ sinh thái Yolo:Bit, chúng tôi cũng sẽ trình bày kiến trúc chung cho một ứng dụng trí tuệ nhân tạo. Chúng ta sẽ xây dựng một ứng dụng điều khiển bằng giọng nói, chẳng hạn như **bật máy bơm** 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 16.1: 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 16.2: 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 16.3: 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:

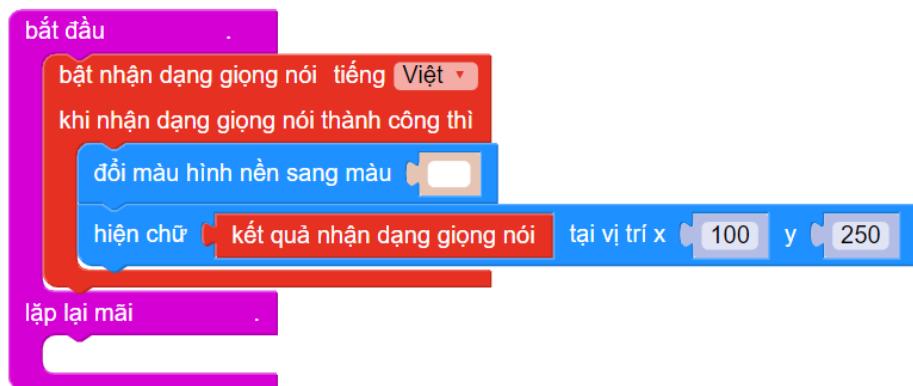


Hình 16.4: Khối lệnh nhận diện giọng nói

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.



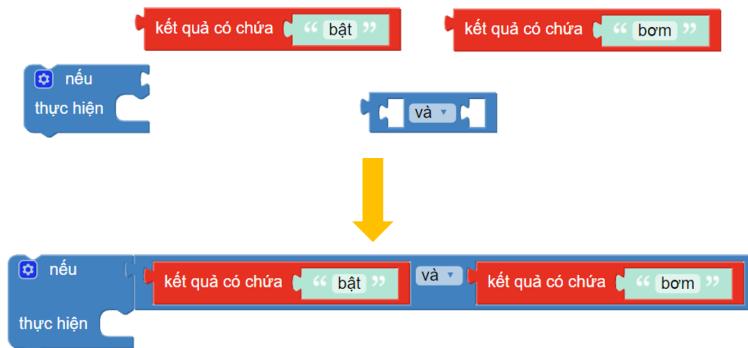
Hình 16.5: Hiển thị kết quả giọng nói

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 16.6: *Bắt hai từ khóa để điều khiển máy bơm*

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

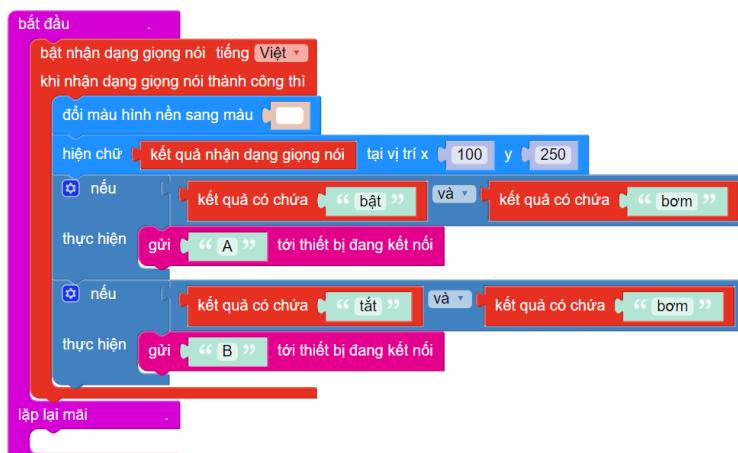
Khi bắt được từ lệnh đi thẳng, chúng ta sẽ gửi một **mã lệnh** tới thiết bị đang kết nối (có thể là xBot hoặc mạch Yolo:Bit), là **kí tự A** cho đơn giản. Câu lệnh để bắt từ khóa (**kết quả có chứa**) nằm trong nhóm **GIỌNG NÓI**. Câu lệnh **gửi 1** tới **thiết bị đang kết nối** nằm trong nhóm **GIAO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



Hình 16.7: *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 16.8: *Ghép các câu lệnh vào chương trình hoàn chỉnh*

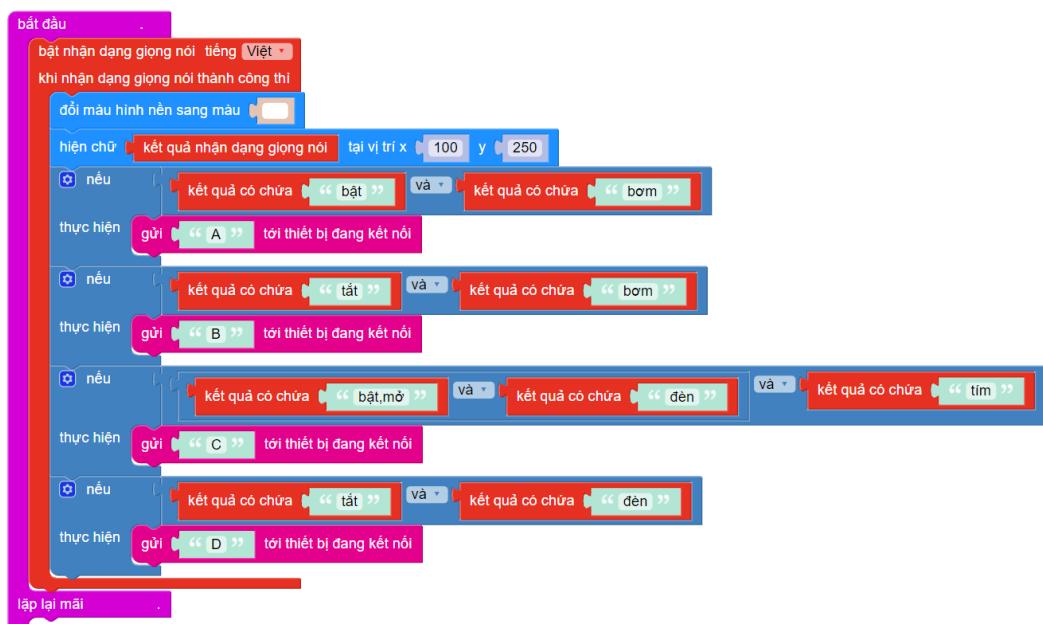
Bước 6: Bắt nhiều từ khóa cùng lúc.

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 16.9: 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:



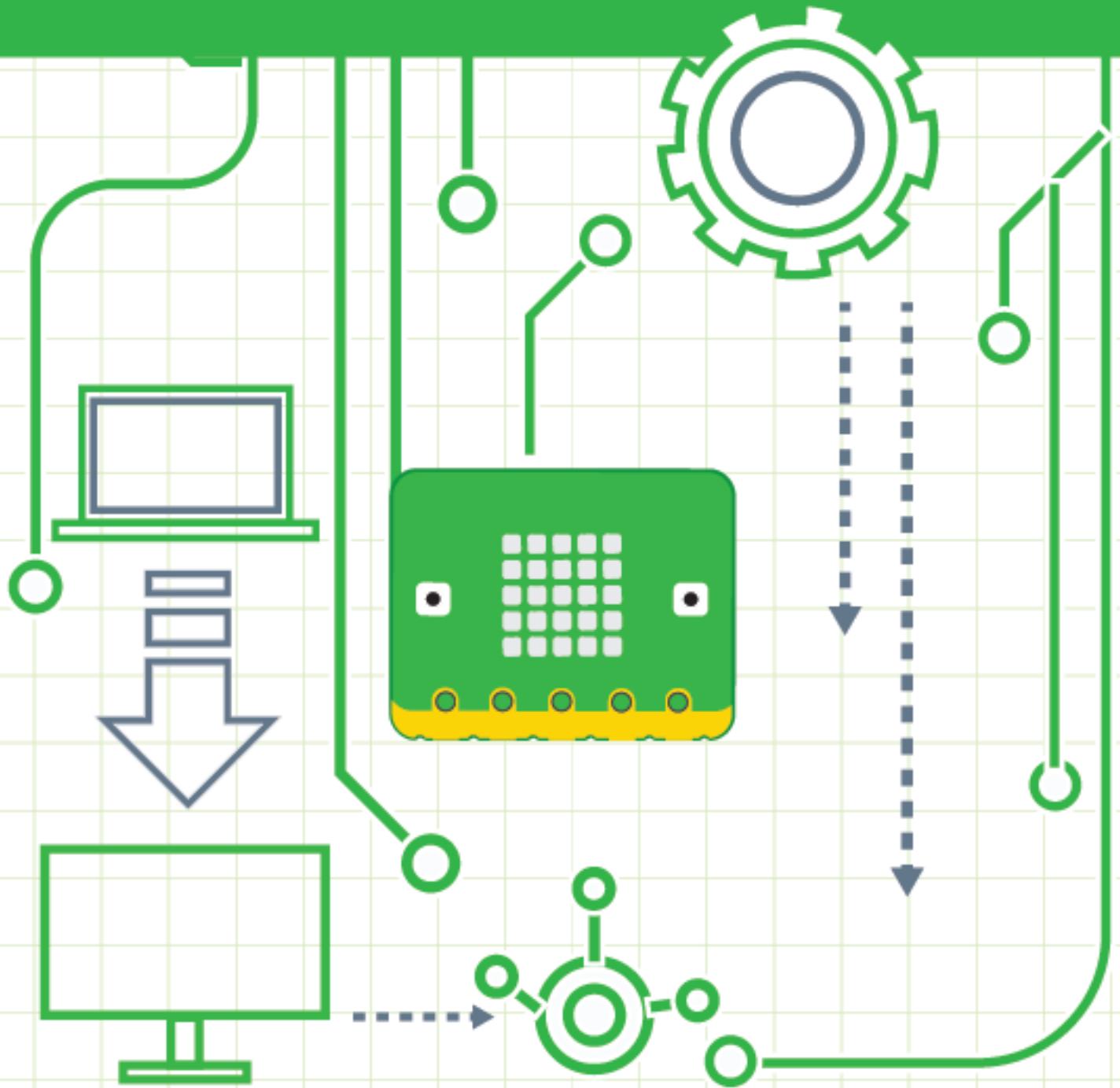
Hình 16.10: Chương trình xử lý giọng nói dựa trên AI

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.

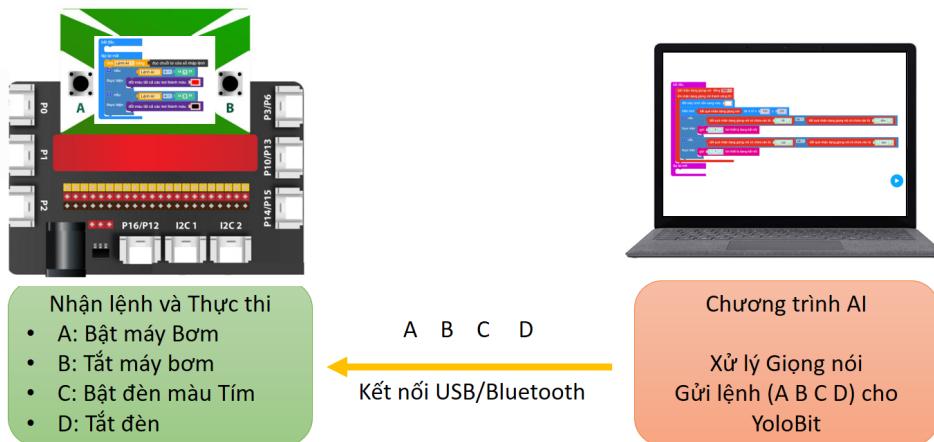
CHƯƠNG 17

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 17.1: 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 bơm
- B: Tắt máy bơm
- C: Bật đèn màu tím
- D: Tắt đèn

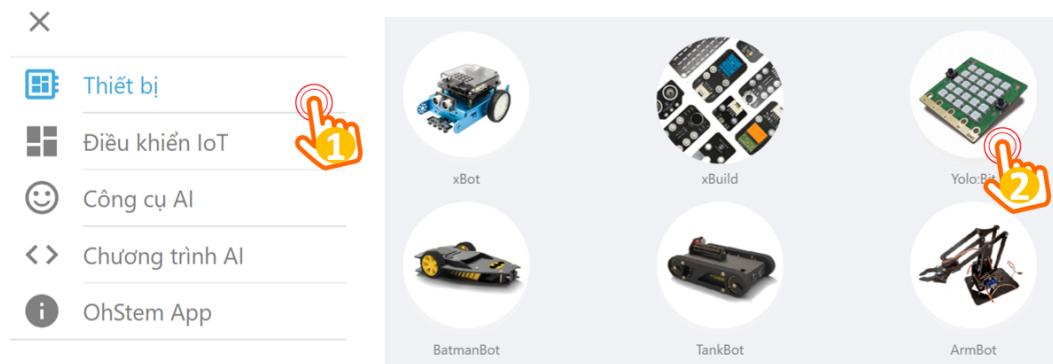
Chúng ta giả định rằng **máy bơm được kết nối với Yolo:Bit ở chân P3**. Trong khi đó, **25 đèn của Yolo:Bit** sẽ được sử dụng để phát sáng ra màu tương ứng. Các mục tiêu chính trong bài hướng dẫn này như sau:

- 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 17.2: *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 17.3: *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:

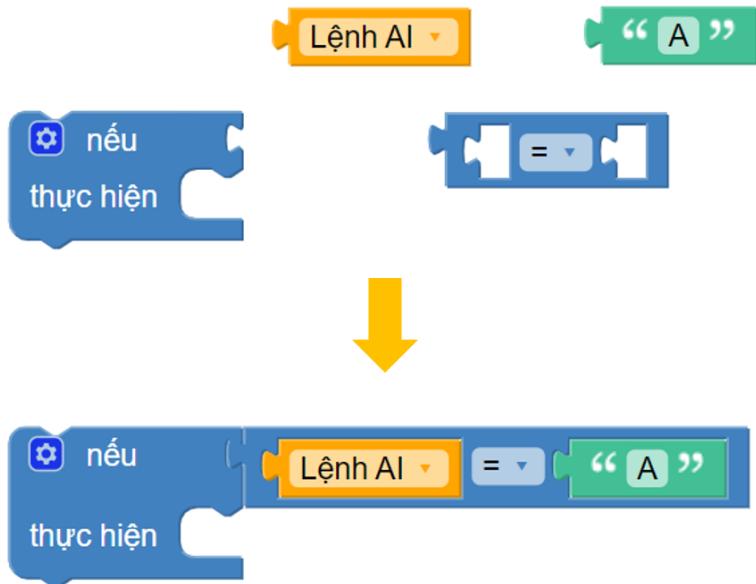


Hình 17.4: Đọc lệnh từ cửa sổ lệnh

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 1 phép so sánh chuỗi, trước khi ghép nó với câu lệnh **nếu ... thực hiện**. Từng bước thực hiện cho phần này được minh họa như hình bên dưới:

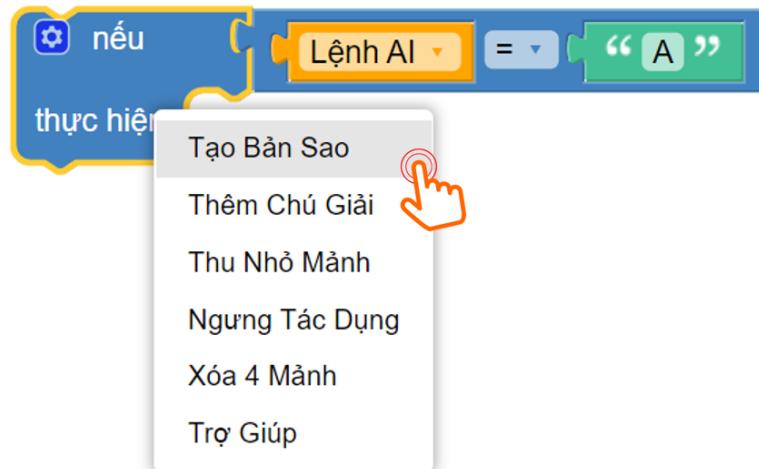


Hình 17.5: Xây dựng phép so sánh chuỗi

Toán tử chuỗi cho số 1, được lấy trong phần **NÂNG CAO**, và chọn tiếp trong **CHỮ VIẾT**. Các khối lệnh còn lại được lấy trong phần **LOGIC**.

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 17.6: Nhân bản câu lệnh so sánh chuỗi

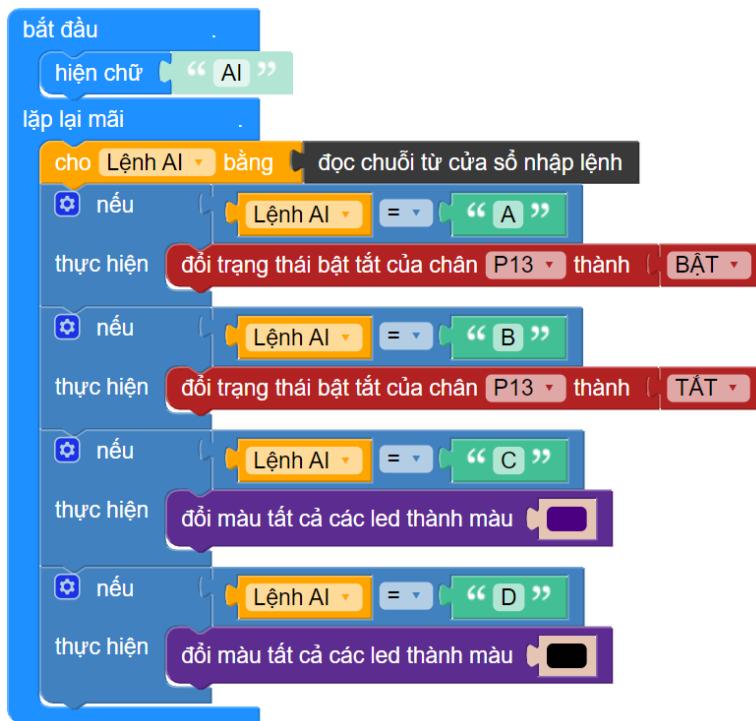
Cuối cùng, chúng ta ghép nối lại chương trình để có được cấu trúc chương trình như sau:



Hình 17.7: Cấu trúc chương trình nhận 4 lệnh

Bước 6: Thực hiện các di chuyển tương ứng.

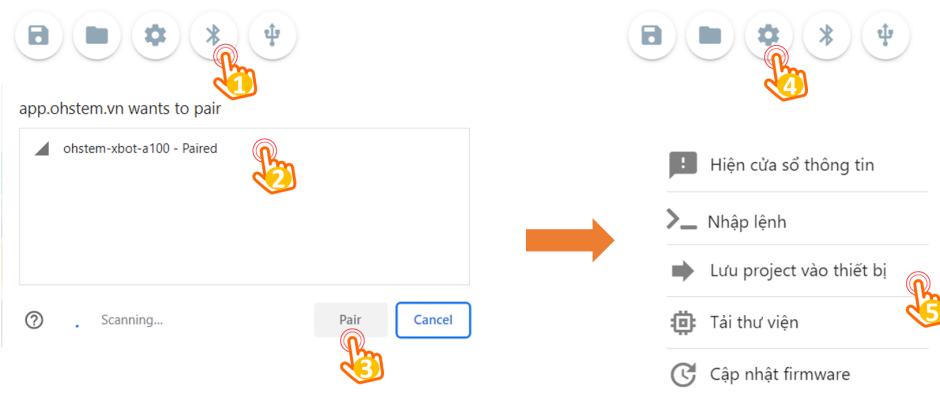
Cho đến bước này, chúng ta sẽ ghép thêm các câu lệnh điều khiển ngoại vi cho trùng khớp với các từ lệnh đã định nghĩa từ đầu. Các câu lệnh cần thiết có thể tìm thấy trong phần **NÂNG CAO**, **CHÂN CẤM** và **LED** cho chương trình sau đây.



Hình 17.8: 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.



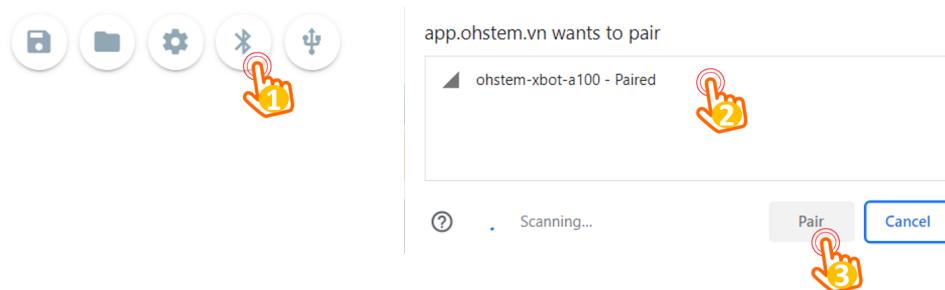
Hình 17.9: Kết nối và lưu chương trình vào Yolo:Bit

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 17.10: 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.

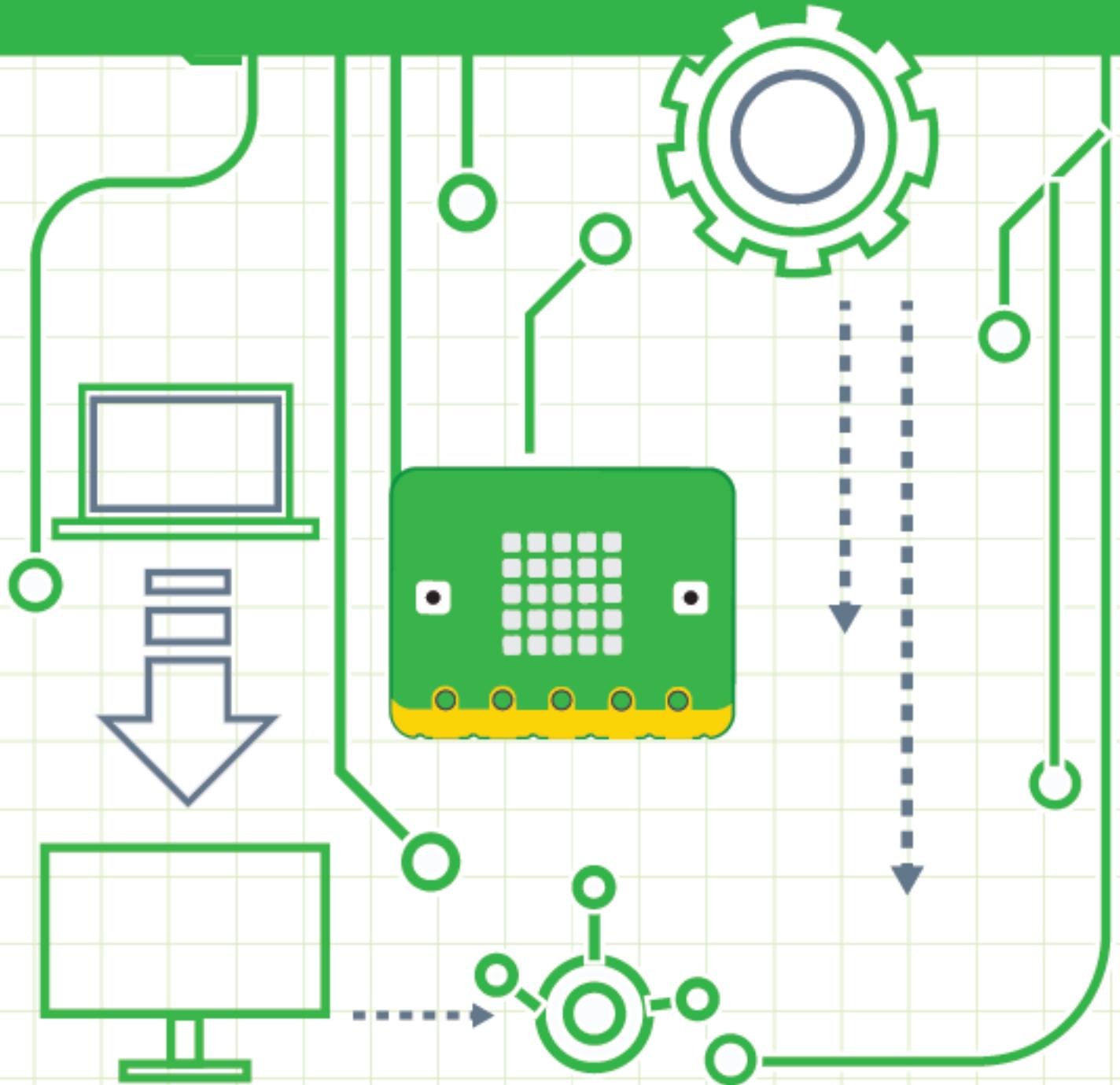
Trong trường hợp có lỗi, bạn chọn cách xử lý như hướng dẫn sau:

- 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 18

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 18.1: Huấn luyện AI để phân biệt cây đang phát triển tốt và hay xấu

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 cây trồng đang tốt hoặc đang xấu**. Các nội dung chính trong bài đầu tiên bao gồm:

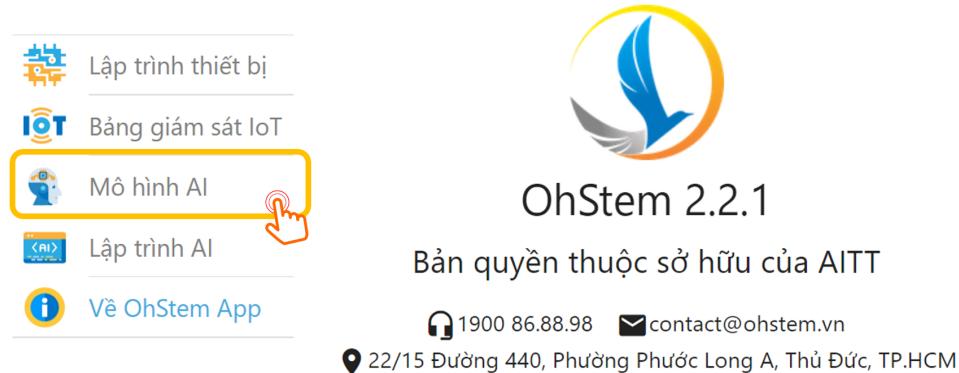
- Sử dụng được công cụ Teachable Machine trên trang lập trình trực tuyến OhStem
- Thu thập dữ liệu hình ảnh liên quan đến cây trồng
- Huấn luyện hệ thống phân biệt được cây trồng đang tốt hoặc đang xấu

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.

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/>, trả về màn hình

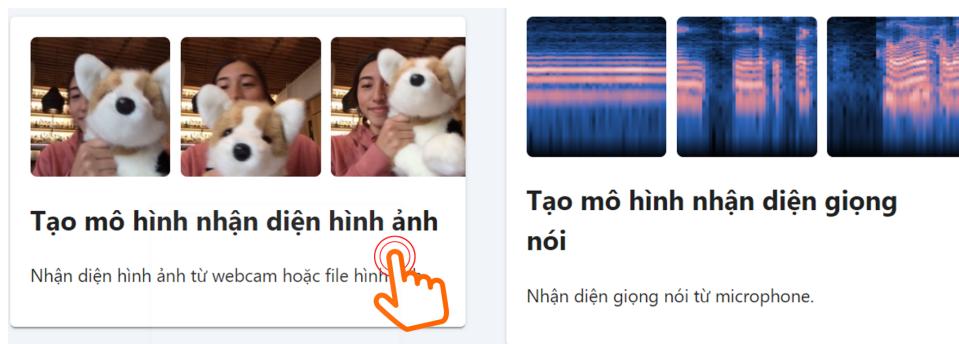
chủ của nó (bằng cách nhấn nút trở về trong giao diện của trang web), như minh họa ở hình bên dưới:



Hình 18.2: Giao diện trang chủ và tính năng huấn luyện mô hình AI

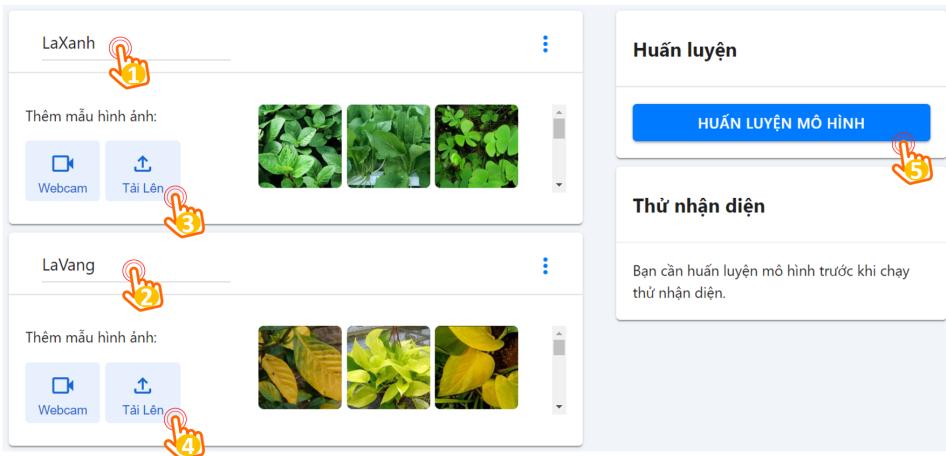
3 Huấn luyện hệ thống

Bước 1: Từ trang chủ trình bày ở Hình 18.2, bạn chọn vào **Mô hình AI**, chọn tiếp **Tạo mô hình nhận diện hình ảnh**, như minh họa ở hình bên dưới:



Hình 18.3: 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.

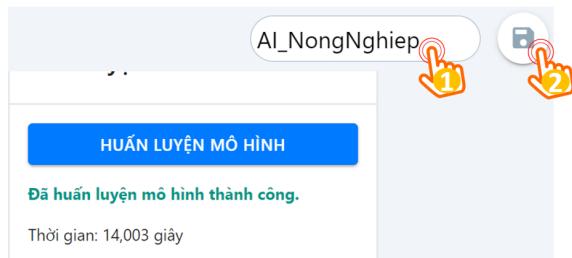


Hình 18.4: Huấn luyện hệ thống AI cho nông nghiệp

Trong phần này, chúng tôi sẽ thu tầm các hình ảnh trên mạng và sử dụng tính năng **Tải Lên** (xem các kí hiệu 3 và 4) thay vì chụp trực tiếp từ Webcam. Tất nhiên bạn đọc cũng có thể chụp thêm hình từ Webcam để tăng thêm nội dung cần huấn luyện cho bộ não AI.

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 18.5: Đặt tên và lưu lại dự án AI

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.

Lưu mô hình "AI_NongNghiep"
thành công!

OK

Hình 18.6: Lưu dự án AI 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.

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:



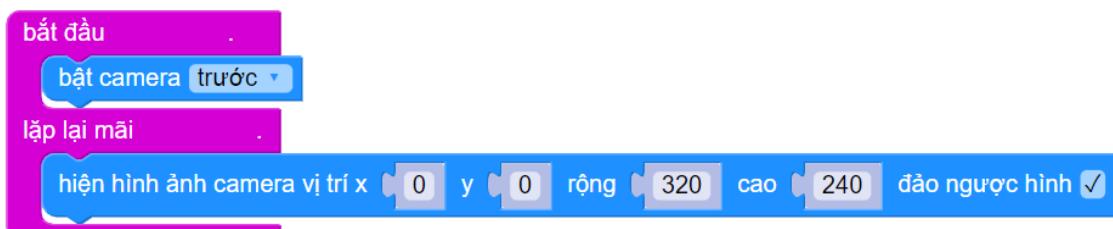
Hình 18.7: Giao diện lập trình Trí tuệ nhân tạo

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 18.8: 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.

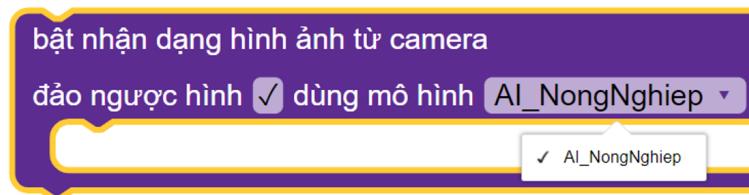


Hình 18.9: Khởi động camera nhận dạng

Hiện tại, với phiên bản chạy trên máy tính, bạn đọc cần khởi động **Camera trước**. Với câu lệnh trong phần **lắp mái mái**, bạn hãy đảm bảo rằng mình đã chọn vào **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 18.10: Tạo dự án 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 18.11: 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).

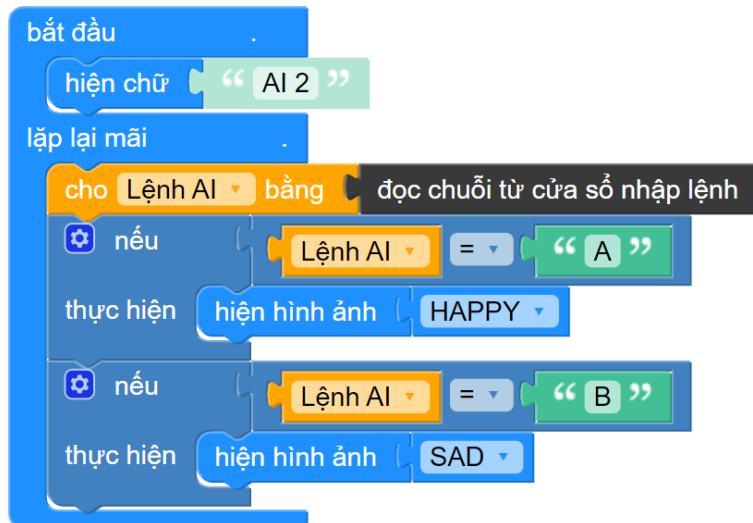
Bước 4: Hoàn thiện chương trình.

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



Hình 18.12: Tạo dự án trí tuệ nhân tạo

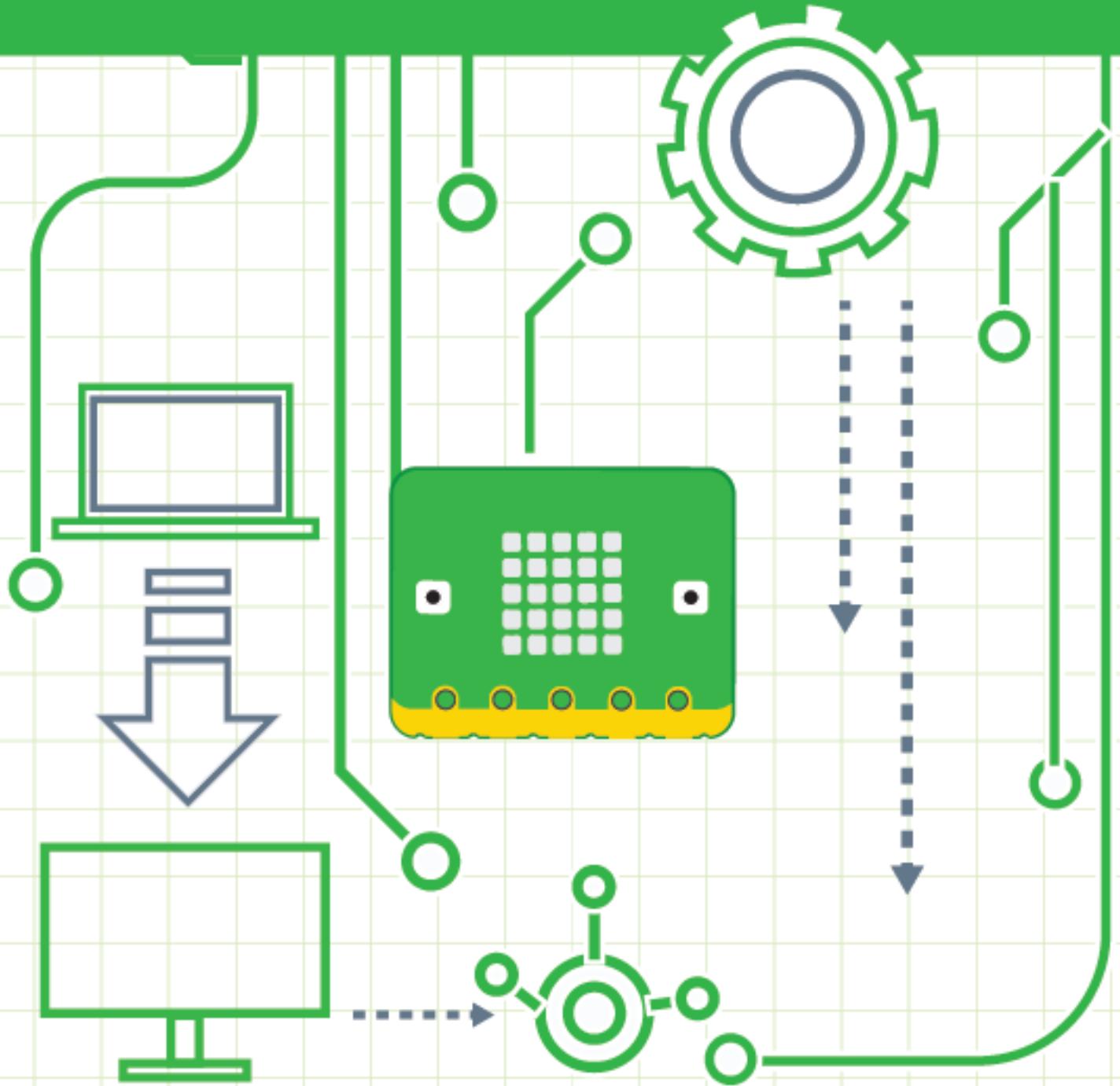
Sau khi kiểm tra dự án hoàn chỉnh, câu lệnh trong lặp mãi mãi có thể bỏ qua, để giảm tải cho chương trình nếu cần. Phần chương trình trên mạch Yolo:Bit hoàn toàn tương tự với bài trước, nên sẽ không trình bày chi tiết ở hướng dẫn này. Chương trình gợi ý cho bạn đọc được trình bày như sau:



Hình 18.13: Chương trình nhận lệnh trên mạch Yolo:Bit

CHƯƠNG 19

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 điều kiện bất thường của cây trồng đã được trình bày.



Hình 19.1: 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 điều kiện cây trồng, 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.

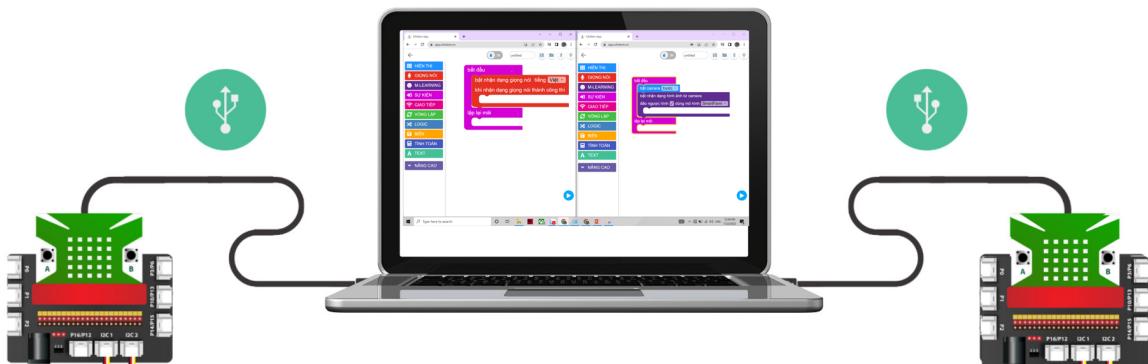
Điểm thú vị lớn nhất với công nghệ học máy này, là bạn có thể chủ động dạy cho nó với các hình ảnh từ dự án cụ thể của mình. Tuy nhiên, điểm bất lợi là dữ liệu do bạn đọc cung cấp thường có số lượng ít, chưa đủ phong phú và đa dạng để xử lý nhiều trường hợp phát sinh trong thực tế. Chẳng hạn như, để 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:

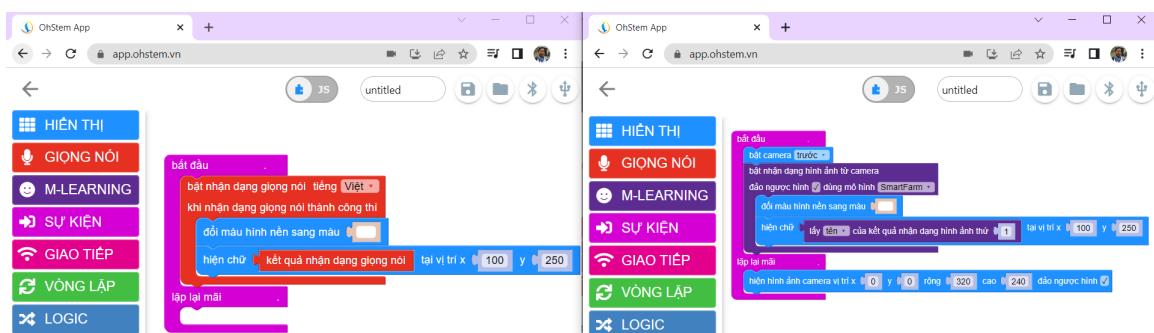
- Nhận diện giọng nói: Truy cập vào micro của thiết bị.
- Học máy với Google: Truy cập vào webcam của thiết bị.

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. Kiến trúc của ứng dụng khi sử dụng cả 2 công nghệ này được đề xuất như sau:



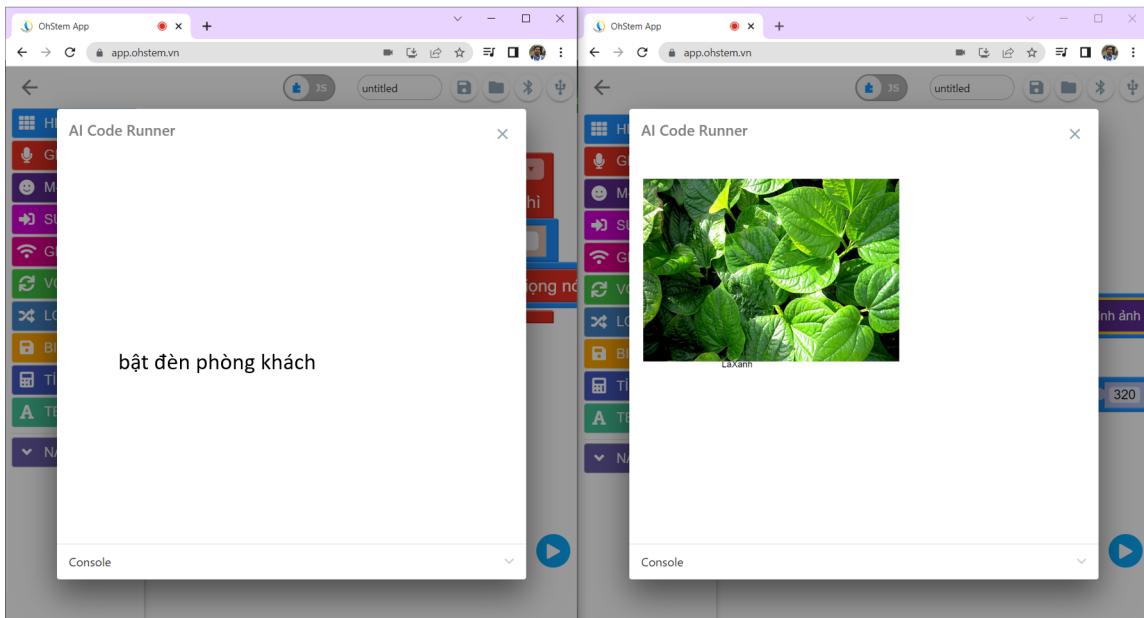
Hình 19.2: Kiến trúc hệ thống với 2 công nghệ AI

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 19.3: Hiện thực nhận dạng giọng nói và giám sát cây trồng bằng AI

Hình ảnh bên dưới là minh họa cho việc chạy 2 chương trình AI song song trên máy tính:



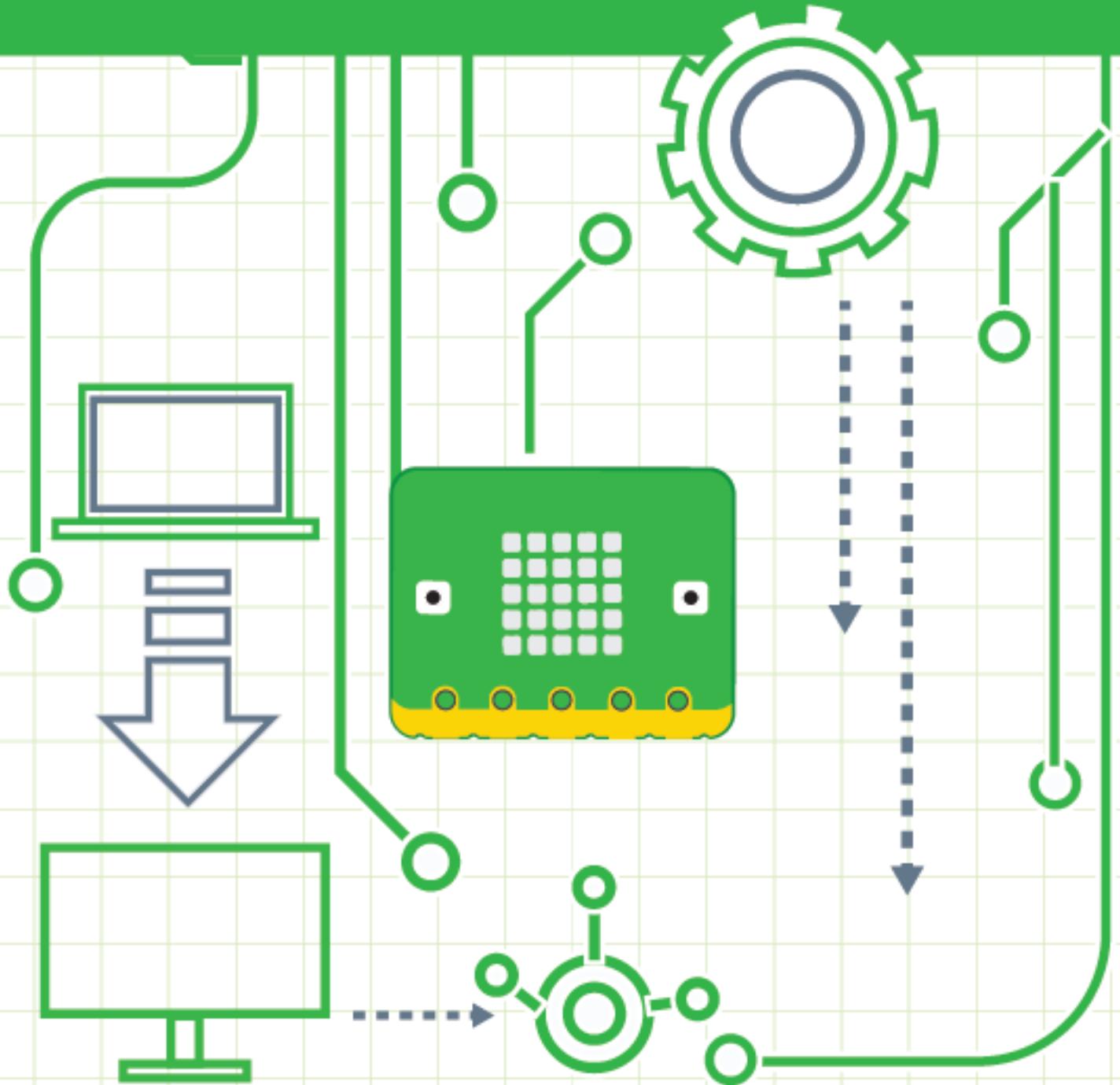
Hình 19.4: Chạy 2 chương trình AI trên máy tính đồng thời

Thông tin quan trọng tiếp theo, mà thiết bị di động không thể hỗ trợ được, là chúng ta phải kết nối với 2 mạch Yolo:Bit độc lập cho hệ thống này. Vì mỗi hệ thống nhân tạo cần chạy trên 1 cửa sổ độc lập, nó không thể chia sẻ tài nguyên phần cứng. Chính vì lý do này, **hai mạch Yolo:Bit** sẽ được sử dụng: một mạch dùng cho nhận lệnh giọng nói và mạch còn lại, dùng cho nhận lệnh từ bộ não AI tự huấn luyện.

Mỗi mạch sẽ kết nối với máy tính thông qua 2 cổng USB khác nhau để tín hiệu được ổn định hơn. Khi chọn lựa thiết bị kết nối với AI, bạn đọc cần chọn cổng COM cho đúng với mỗi mạch Yolo:Bit. Việc hiện thực chương trình trên mạch Yolo:Bit hoàn toàn tương tự với các bài hướng dẫn trước, nên sẽ không được trình bày lại ở bài này.

CHƯƠNG 20

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 vật 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 Yolo:Bit hay xBot, 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 đang kết nối với chương trình AI (là mạch Yolo:Bit hoặc xBot).

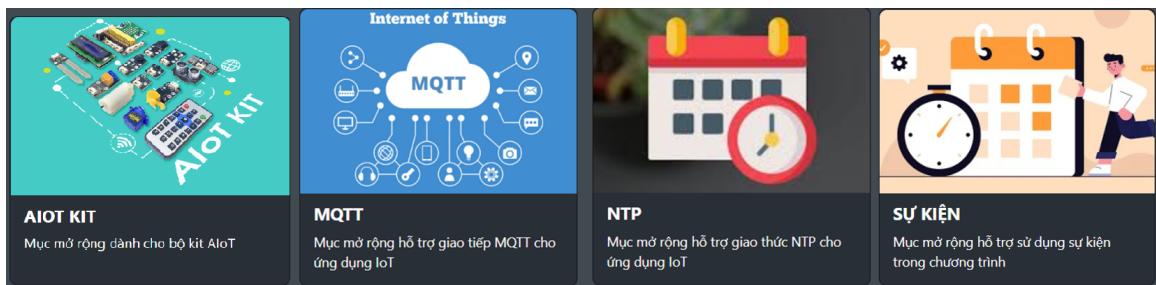
Vấn đề ở đây, là tại chương trình của mạch Yolo:Bit, đang được xây dựng theo kiến trúc của một ứng dụng IoT. Trong ứng dụng này, việc kiểm tra kết nối với server OhStem phải được thực hiện định kì với chu kỳ 1000ms (1 giây). Trong khi đó, tốc độ gửi dữ liệu từ AI lại nhanh hơn thời gian này. Do đó, mỗi lần nhận lệnh AI, chúng ta sẽ nhận được **Một chuỗi lệnh nhiều kí tự**, thay vì chỉ đơn giản là 1 kí tự như các bài hướng dẫn trước. Việc xử lý lúc này sẽ xem xét trên kí tự sau cùng nhận được, trước khi thực thi các tác vụ điều khiển cho trí tuệ nhân tạo.

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



Hình 20.1: Thư viện cho một ứng dụng IoT

cũng hỗ trợ cho việc đăng ký kênh để nhận dữ liệu và gửi dữ liệu lên server.

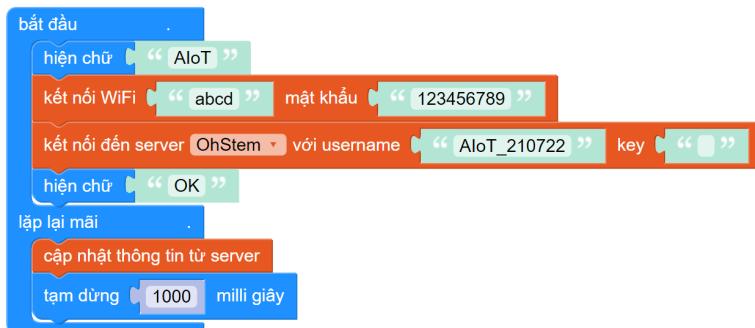
Tiếp theo, là thư viện **AIoT Kit**, hỗ trợ cho việc lập trình với thiết bị, bao gồm các cảm biến, màn hình LCD hay máy bơm. Thực ra, thư viện này được dựa hoàn toàn trên **HOME:BIT V3**. Bạn đọc có thể sử dụng thư viện cũ nếu như đã có kinh nghiệm với thư viện HOME:BIT V3.

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.

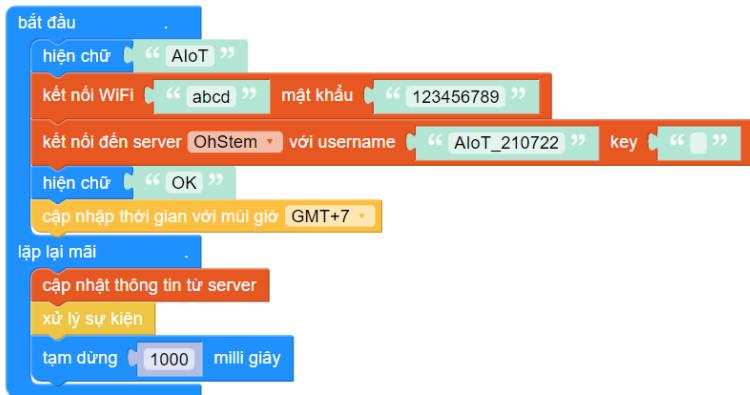
3 Khung chương trình IoT

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:



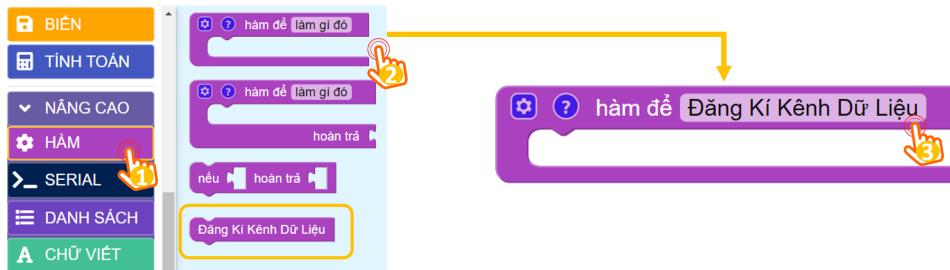
Hình 20.2: Các câu lệnh liên quan đến kết nối IoT

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 20.3: Khởi tạo cho thời gian thực và sự kiện

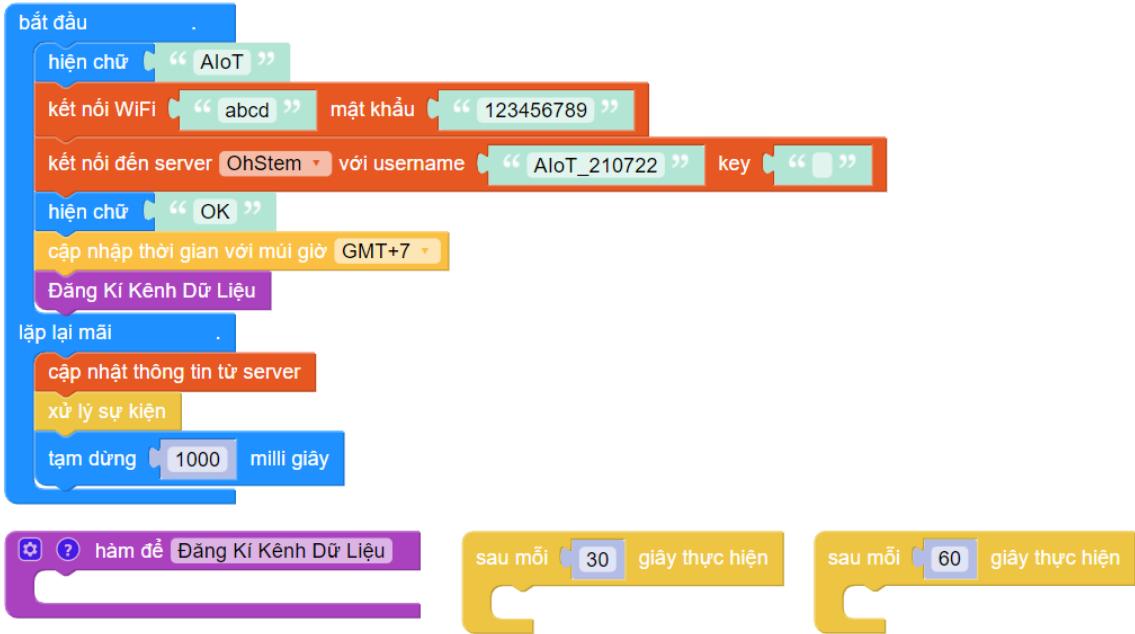
Sau phần khởi tạo, liên quan đến IoT, chúng ta cần 2 tính năng: đăng ký kênh nhận dữ liệu và định kì gửi dữ liệu lên server. Trước tiên, với việc nhận dữ liệu, do chúng ta chỉ cần thực hiện một lần, nên nó sẽ được hiện thực trong phần **bắt đầu**. Tuy nhiên để tiện cho việc tổ chức chương trình, chúng 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 20.4: 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 20.5: Tạo hàm Đăng Kí Kênh Dữ Liệu

Trong tương lai, việc định kì gửi dữ liệu lên server sẽ được hiện thực trong các khối lệnh định kì. Một chương trình sẽ có thể gồm **nhiều khối lệnh định kì**. Trong 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

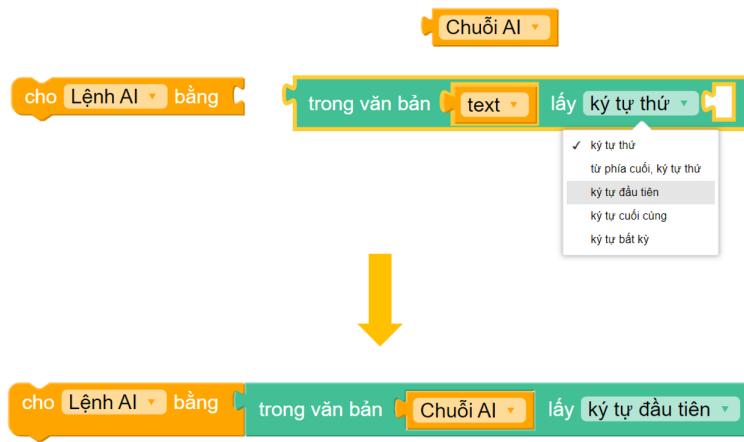
Về phần AI, mạch Yolo:Bit sẽ **định kì kiểm tra** lệnh nhận được từ chương trình trí tuệ nhân tạo (máy tính, điện thoại di động hoặc máy tính bảng). Một biến **Chuỗi AI** được tạo ra để đọc thông tin từ cửa sổ lệnh, như sau:



Hình 20.6: Đọc lệnh AI từ cửa sổ lệnh

Thực ra, trừ một số rất ít ứng dụng cần phát hiện thông tin AI ngay tức thì, đa số các ứng dụng của chúng ta hoàn toàn có thể đọc lệnh AI mỗi vài giây một lần. Ở hướng dẫn này, chúng tôi chọn **chu kì 2 giây** cho phần xử lý AI.

Với chu kì là vài giây, chúng ta sẽ nhận được một chuỗi lệnh, thay vì chỉ một lệnh như trong các bài hướng dẫn trước. Do đó, chúng ta cần phải xử lý trên chuỗi để lấy ra lệnh gần nhất nhận được. Để xử lý cho phần này, chúng ta sẽ cần câu lệnh trong nhóm **NÂNG CAO**, chọn tiếp vào **VĂN BẢN**, như hướng dẫn sau đây:



Hình 20.7: Lấy ký tự lệnh sau cùng nhận được

Tuy nhiên, để cho an toàn hơn, chúng ta sẽ kiểm tra độ dài chuỗi của biến **Chuỗi AI**, trước khi thực hiện lấy giá trị cho biến **Lệnh AI**, như sau:



Hình 20.8: Thêm câu điều kiện kiểm tra độ dài chuỗi

Câu lệnh **độ dài của** được lấy trong mục **NÂNG CAO, VĂN BẢN**. Cuối cùng, khi ráp nối hết các câu lệnh vào khối xử lý định kì cho phần trí tuệ nhân tạo, chúng ta có chương trình hoàn chỉnh như sau:



Hình 20.9: Khung chương trình hoàn chỉnh

Sau khi đã có các giá trị cho biến **Lệnh AI**, bạn đọc có thể tự bổ sung thêm các câu lệnh nếu để hoàn thiện tiếp cho phần xử lý của mình. Chương trình khung trên được chia sẻ ở đường dẫn sau đây để bạn đọc tiện tham khảo:

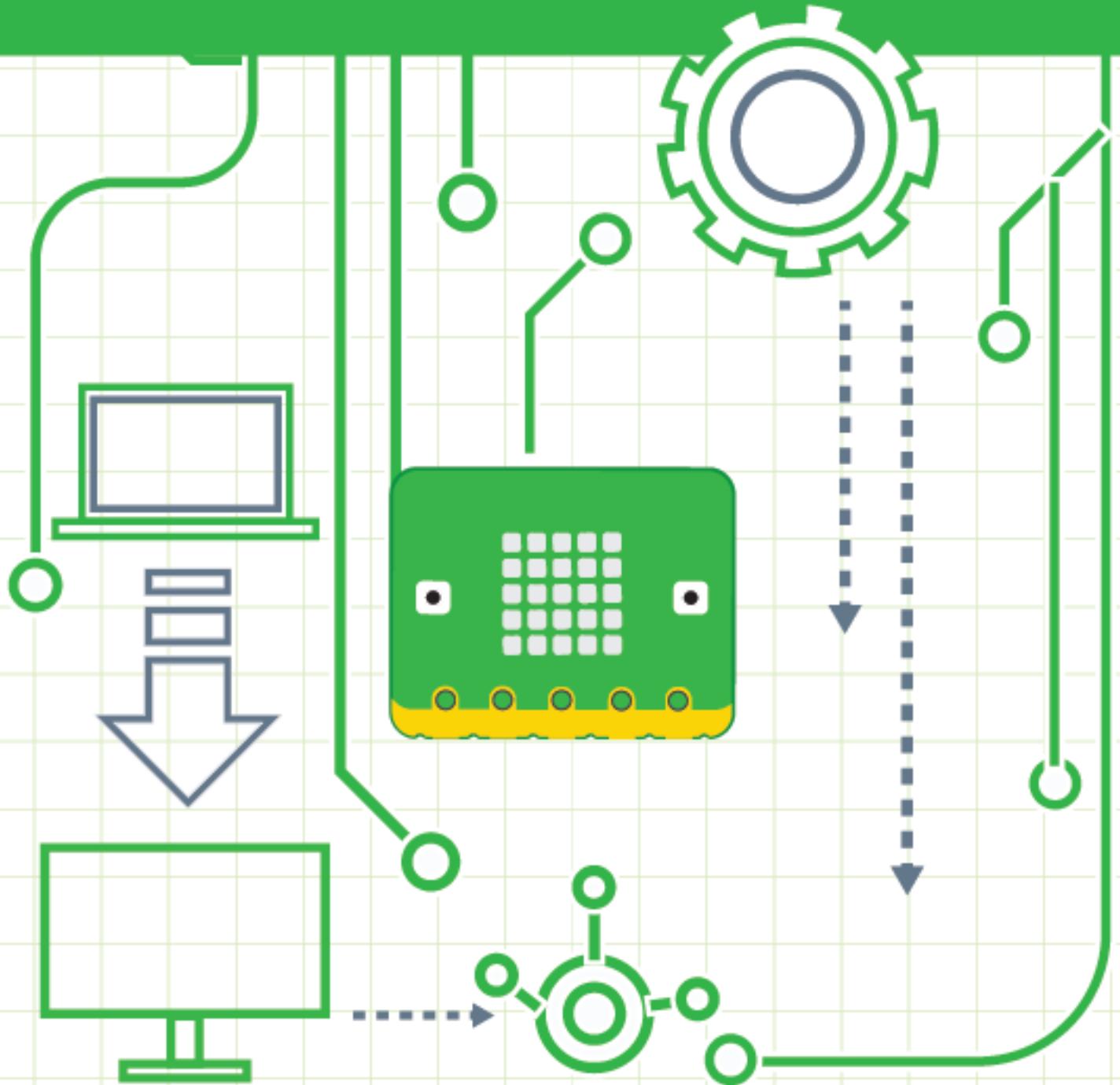
<https://app.ohstem.vn/#!/share/yolobit/2CGB7AIRlcflyxDqGIlyKom2dQv>

Phần V

Hiện Thực Dự Án Yolo:Farm

CHƯƠNG 21

Tổng quan dự án



1 Giới thiệu

Tại Việt Nam, nông nghiệp là một ngành kinh tế chủ lực và tạo lượng lớn việc làm cho lao động ở khu vực nông thôn. Thế nhưng ngành này đang phải đổi mới trực diện và chịu nhiều thiệt hại từ những tác động tiêu cực của biến đổi khí hậu (BĐKH) như nước biển dâng, lũ lụt, hạn hán, xâm nhập mặn, thời tiết xấu, làm ảnh hưởng nghiêm trọng đến sản xuất nông nghiệp, vốn dựa nhiều vào các hoạt động thủ công.



Hình 21.1: Nông nghiệp công nghệ - Nông nghiệp 4.0

Với các công nghệ được trình bày trong giáo trình này, chúng tôi đề xuất một dự án liên quan đến việc ứng dụng công nghệ vào trong nông nghiệp. Chúng tôi gọi tên cho dự án này là Yolo:Farm. Thực ra, nông nghiệp công nghệ (hay còn gọi là nông nghiệp 4.0) đã giúp giải quyết đáng kể bài toán chi phí, lao động, hiệu quả trong sản xuất và đáp ứng tốt hơn nhu cầu thực phẩm trong nước. Một quốc gia được đánh giá là có nền nông nghiệp thông minh hiện đại trên thế giới không thể không kể đến là Israel, một nước nhỏ ở Trung Đông có điều kiện tự nhiên vô cùng khắc nghiệt, 2/3 diện tích lánh thổ là sa mạc, còn lại là đồi núi đá trọc, khí hậu khô hạn.

Trong dự án này, chúng tôi cố gắng minh họa cho việc đưa các công nghệ liên quan đến tự động hóa, kết nối vạn vật và trí tuệ nhân tạo vào trong dự án. Bên cạnh đó, các hình thức phân tích dữ liệu đơn giản cũng sẽ được trình bày trong phần hướng dẫn từ chương này.

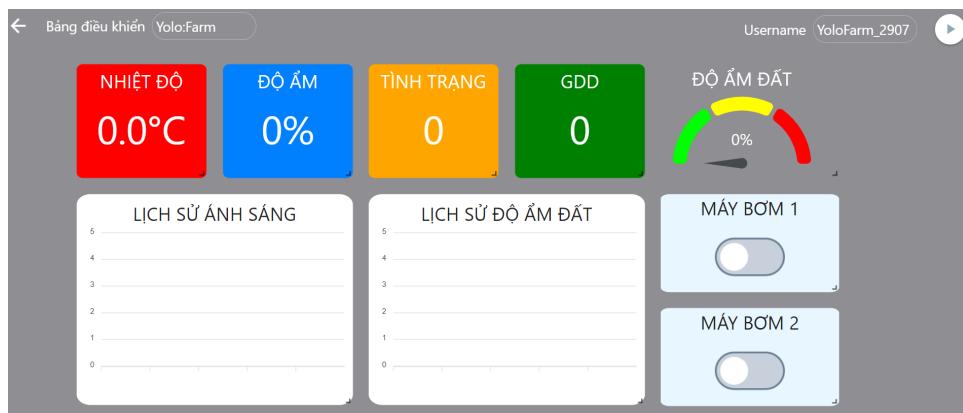
Phần tiếp theo của bài hướng dẫn này sẽ trình bày chi tiết các tính năng trong dự án. Với mỗi tính năng, phần cứng hoặc công nghệ liên quan sẽ được đề xuất.

2 Giám sát môi trường

Đây là tính năng cơ bản nhất của ứng dụng nông nghiệp công nghệ. Các thông tin liên quan đến điều kiện trời trọn tót sẽ được giám sát một cách tự động và liên tục (chẳng hạn mỗi 5 phút một lần) để người nông dân có thể theo dõi điều kiện trời trọn tót 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, đất 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 trong mọi môi trường nông nghiệp thông minh. Với tính năng này, chúng ta sẽ sử dụng cảm biến DHT20 để giám sát.
- Độ ẩm đất: Độ ẩm đất là cơ sở quan trọng cho việc tưới tiêu thông minh, giúp cho sự phát triển của cây trồng được ổn định trước những thay đổi của thời tiết.
- Điều kiện ánh sáng: Mặc dù mạch Yolo:Bit đã có sẵn một cảm biến ánh sáng, chúng tôi sẽ sử dụng cảm biến ánh sáng rời dưới dạng một ngoại vi gắn vào mạch mở rộng. Không chỉ chính xác hơn, việc lắp đặt và triển khai nó sẽ dễ dàng hơn.



Hình 21.2: Bảng điều khiển IoT cho dự án Yolo:Farm

Bên cạnh việc liên tục cập nhật các thông tin này lên OhStem server cho việc giám sát từ xa, các thông tin sẽ được hiển thị trực quan tại nơi trồng trọt. Màn hình kí tự LCD 16x2 sẽ được sử dụng trong trường hợp này.

3 Tưới tiêu thông minh

Điểm khác biệt trong dự án Yolo:Farm là khả năng tưới theo giờ của hệ thống. Thực ra đây là tính năng cực kì hữu ích nếu xét trong các môi trường triển khai thực tế. Một số nông trại có diện tích lớn, phải chia thành nhiều phân khu, việc tưới tiêu đúng giờ sẽ giúp cho cây trồng phát triển đồng bộ. Lúc đó, chi phí cho việc thu hoạch và phân loại sẽ được giảm thiểu.

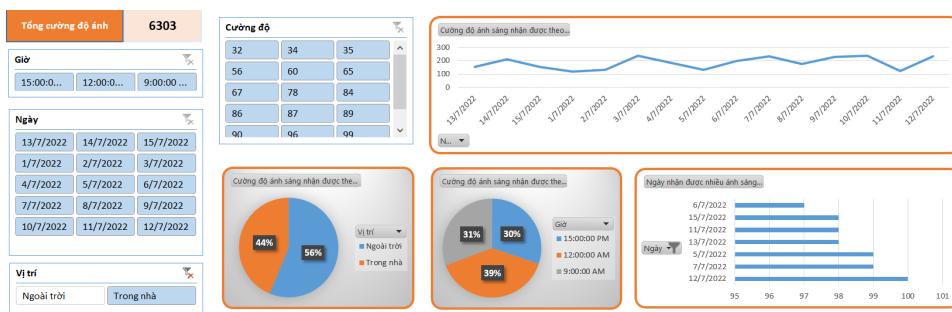
Bên cạnh đó, hệ thống của chúng tôi vẫn sẽ có tính năng tưới tiêu thông minh dựa vào cảm biến độ ẩm đất. Một máy bơm sẽ được bật tắt tự động dựa vào giá trị đo được từ cảm biến. Với 2 máy bơm hoạt động theo 2 nguyên lý điều khiển khác nhau, Yolo:Farm cung cấp giải pháp toàn diện cho tưới tiêu thông minh.

Phần cứng chuyên dụng cho tính năng này là mạch điều khiển hai nguồn USB, cho phép điều khiển độc lập 2 máy bơm khi vận hành cũng như tối ưu hóa khả năng kết nối của hệ thống.

4 Phân tích dữ liệu

Việc phân tích dữ liệu quan trắc sẽ giúp nông dân rút ra được nhiều thông tin hữu ích cho tương lai, và là tính năng không thể thiếu trong dự án. Hai tính năng chính liên quan đến việc phân tích dữ liệu được trình bày như sau:

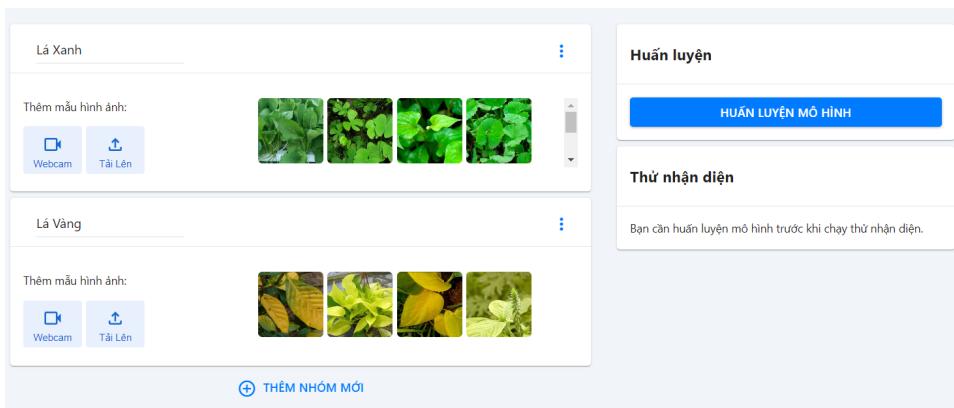
- Tính toán thời gian tăng trưởng của cây trồng dựa vào cảm biến ánh sáng. Dựa vào thông tin này, nông dân sẽ có kế hoạch thu hoạch hợp lý nhất.
- Phân tích dữ liệu trên Excel, biểu diễn dưới dạng đồ thị trực quan từ các dữ liệu quan trắc. Tính năng này sẽ rút trích ra được nhiều thông tin bổ ích trong 1 mùa vụ trồng trọt.



Hình 21.3: Phân tích dữ liệu trên Excel

5 Phát hiện bất thường ở cây trồng

Trong quá trình trồng trọt, các dấu hiệu bất thường liên quan đến cây trồng sẽ được ghi nhận lại. Chẳng hạn như cây trồng bị vàng lá hay bị sâu bệnh. Liên quan đến tính năng này, vốn đa dạng và phụ thuộc vào từng loại cây trồng, công nghệ AI sẽ được ứng dụng vào để xử lý.



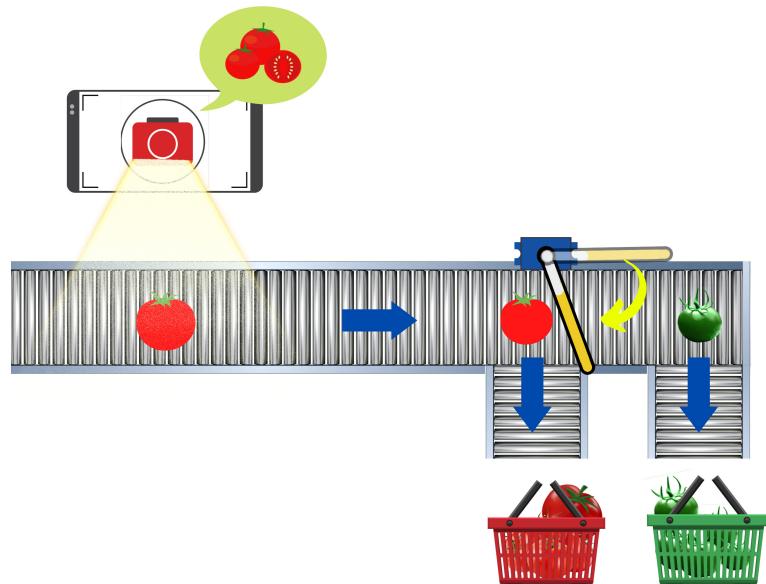
Hình 21.4: Phát hiện bất thường ở cây trồng bằng AI

Khi ứng dụng công nghệ AI vào việc phát hiện bất thường, chúng ta có thể dễ dàng bổ sung thêm các vấn đề phát sinh với cây trồng và huấn luyện lại hệ thống. Tính

năng này cũng cần sự xử lý khéo léo bên phần lập trình và sẽ được trình bày chi tiết ở phần sau, khi việc lập trình AI sẽ cần phải xem xét thêm thông tin về xác suất nhận dạng.

6 Phân loại trái cây thu hoạch

Một trong những lợi thế của trí tuệ nhân tạo, là tính linh hoạt của nó trong việc nhận dạng dựa vào hình ảnh. Đa số các loại trái cây sẽ chuyển màu khi chín tới. Dựa vào điều này, công nghệ AI có thể hỗ trợ nông dân trong vấn đề phân loại trái cây.

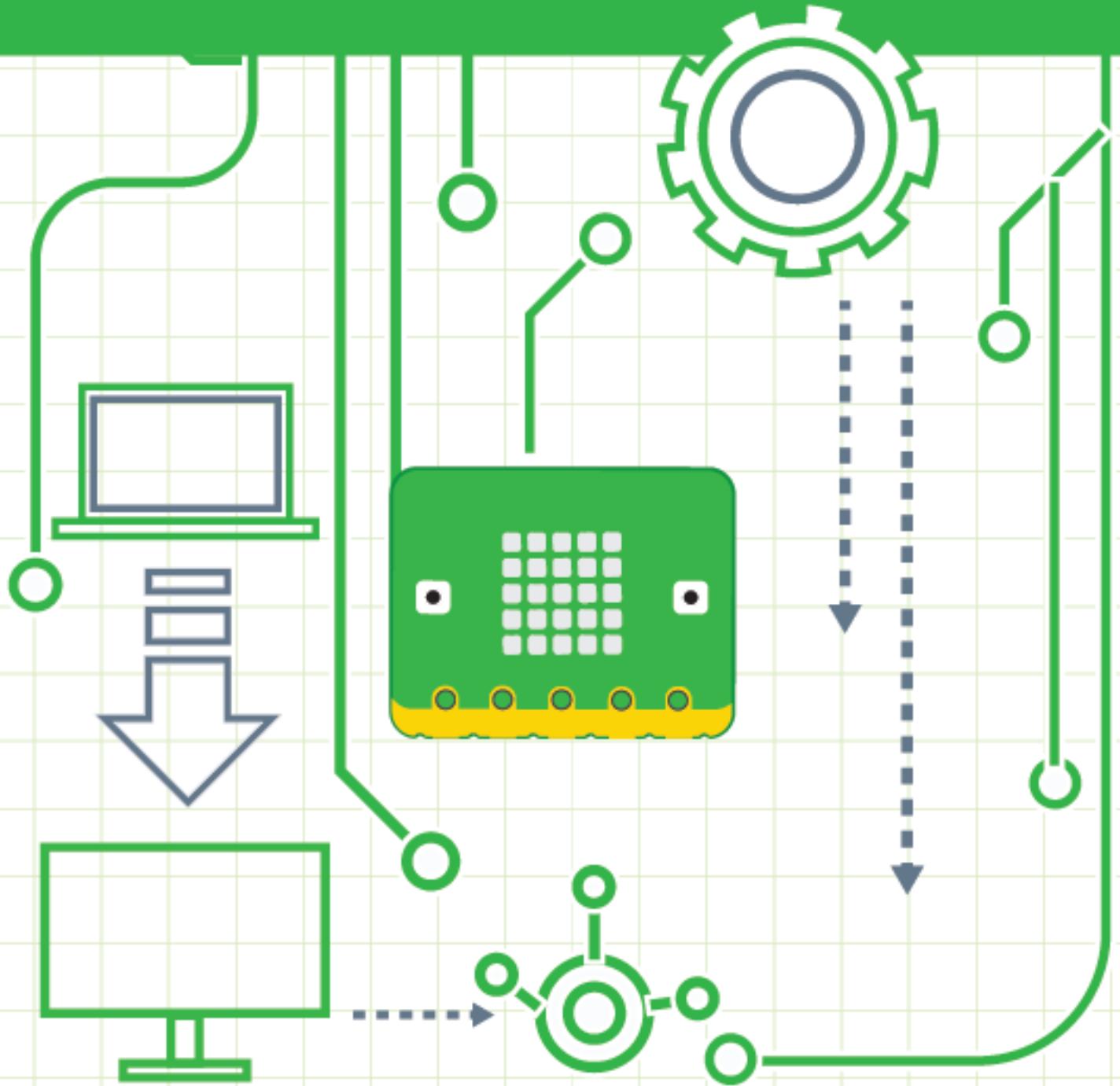


Hình 21.5: Phân loại hoa quả dựa trên trí tuệ nhân tạo

Phân loại trái cây sau khi thu hoạch sẽ đóng góp nhiều giá trị trong một dự án nông nghiệp thông minh. Bên cạnh nhu cầu thực tiễn, tính năng này còn có thể được mở rộng ra cho nhiều dự án khác, chẳng hạn như phân loại rác thải nhựa, thủy tinh và các loại rác thải tái chế.

CHƯƠNG 22

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.

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 và máy bơm 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

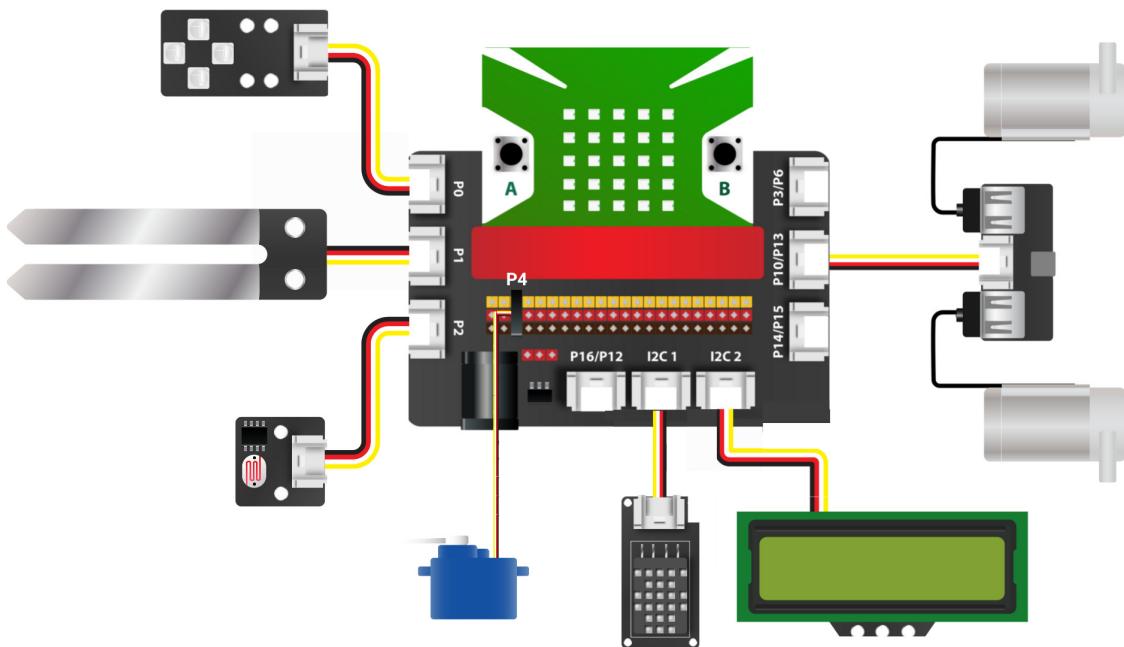
Thiết bị phần cứng cho dự án chủ yếu tập trung ở các tính năng liên quan đến quan trắc môi trường và tự động hóa (tưới tiêu thông minh và phân loại trái cây). Dựa vào các thông tin về thiết bị ở Chương V, các thiết bị cần thiết cho dự án được tóm tắt trong bảng sau đây:

Bảng 22.1: 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	I2C1
	Độ ẩm			
	Độ ẩm đất	Cảm biến độ ẩm đất	P0, P1, P2	P1
	Ánh sáng	Cảm biến ánh sáng	P0, P1, P2	P2
	Đèn báo hiệu	Đèn 3 màu RGB	Tất cả	P0
	Hiển thị	LCD kí tự 16x2	I2C1, I2C2	I2C2
Tưới tiêu thông minh	Tưới tự động (Máy bơm 1)	Mạch ngoại vi 2 USB	Tất cả	P10/P13
	Tưới theo giờ (Máy bơm 2)			
Phân loại trái cây	Hỗ trợ phân loại bằng AI	Động cơ RC	Tất cả	P4

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

tương lai, chẳng hạn màu đỏ là cảnh báo bất thường hay màu xanh là báo hiệu đã đến lúc thu hoạch. Đôi 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.



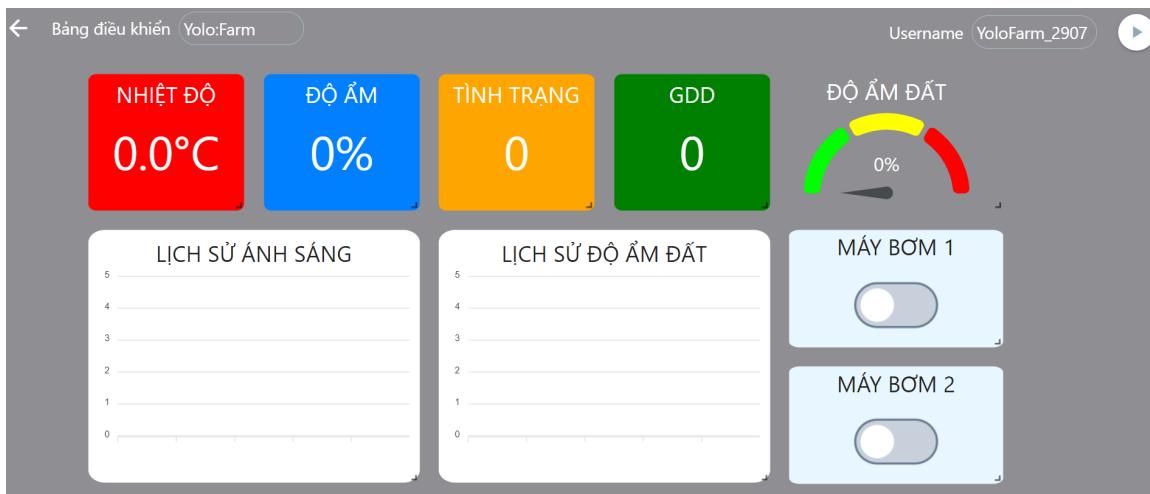
Hình 22.1: Kết nối phần cứng cho dự án

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.
- Một số thông tin cần xem dưới dạng quá trình, sẽ được biểu diễn bằng đồ thị.
- Có 2 công tắc để biểu diễn trạng thái của máy bơm. Hai công tắc này cũng có thể được sử dụng để điều khiển bật/tắt máy bơm bằng tay.

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



Hình 22.2: Bảng điều khiển IoT cho dự án Yolo:Farm

Bên cạnh các thông tin liên quan đến thông tin quan trắc môi trường, chúng tôi có 2 thông tin quan trọng liên quan đến AI và phân tích dữ liệu, cụ thể như sau:

- Tình trạng: Là kết quả của bài toán AI để nhận diện cây đang phát triển tốt hay bị vàng lá hoặc bị ảnh hưởng bởi sâu bệnh.
- GDD: Thông tin về thời gian tăng trưởng của cây, tính theo đơn vị ngày.

Các thông tin trên giao diện được hiển thị mặc định là số. Khi lập trình và chạy trên thiết bị, nó sẽ được cập nhật lại.

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 22.2: Kênh thông tin cho các đối tượng giao diện

Đối tượng	Kênh thông tin
NHIỆT ĐỘ	V1
ĐỘ ẨM	V2
ĐỘ ẨM ĐẤT	V3
ÁNH SÁNG	V4
LỊCH SỬ ĐỘ ẨM	V3
LỊCH SỬ ÁNH SÁNG	V4
GDD	V5
TÌNH TRẠNG	V6
MÁY BƠM 1	V10
MÁY BƠM 2	V11

Trong bảng điều khiển, hai thông số là độ ẩm đất và ánh sáng được biểu diễn dưới hai hình thức khác nhau: giá trị hiện tại và giá trị lịch sử (qua đồ thị). Do đó,

chúng sẽ lấy thông tin từ cùng một kênh dữ liệu, lần lượt là **V3** cho độ ẩm đất và **V4** là cho ánh sáng.

Khi phân phối kênh dữ liệu cho 2 máy bơm, chúng tôi chọn là **V10** và **V11**. 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. Giao diện cho 2 máy bơm là **Công tắc**, được cấu hình chủ yếu dựa vào các thông tin mặc định, chỉ thay đổi tên và kênh dữ liệu mà thôi.

Cuối cùng, bạn cần đặt tên cho bảng điều khiển của mình, trong ví dụ của chúng tôi là **YoloFarm_2907**. 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

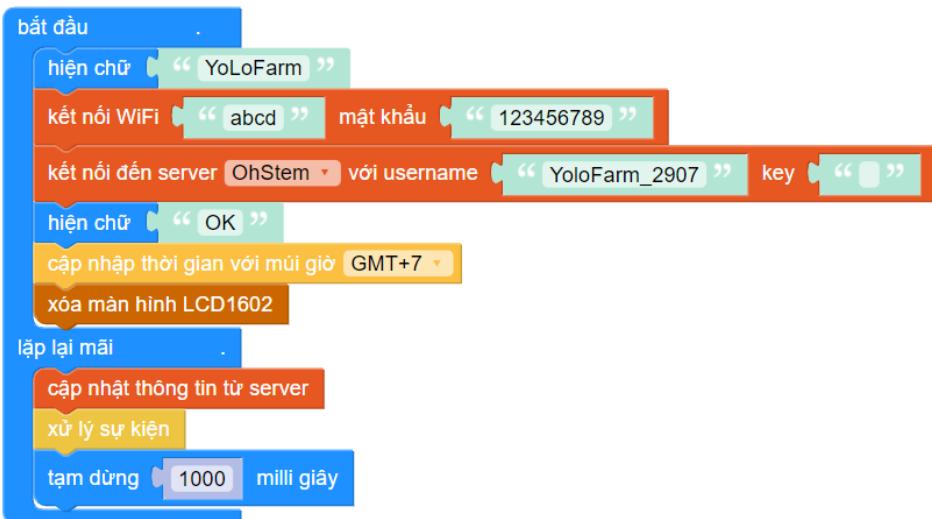
Với các yêu cầu về tính năng, chương trình của chúng ta sẽ cần 3 thư viện chính sau đây:

- HOME:BIT V3: 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, thông tin quan trọng để điều khiển máy bơm theo giờ.
- Sự kiện: Hỗ trợ khôi lệnh xử lý theo chu kỳ, hữu ích cho việc giám sát định kì các thông số liên quan đến cảm biến.



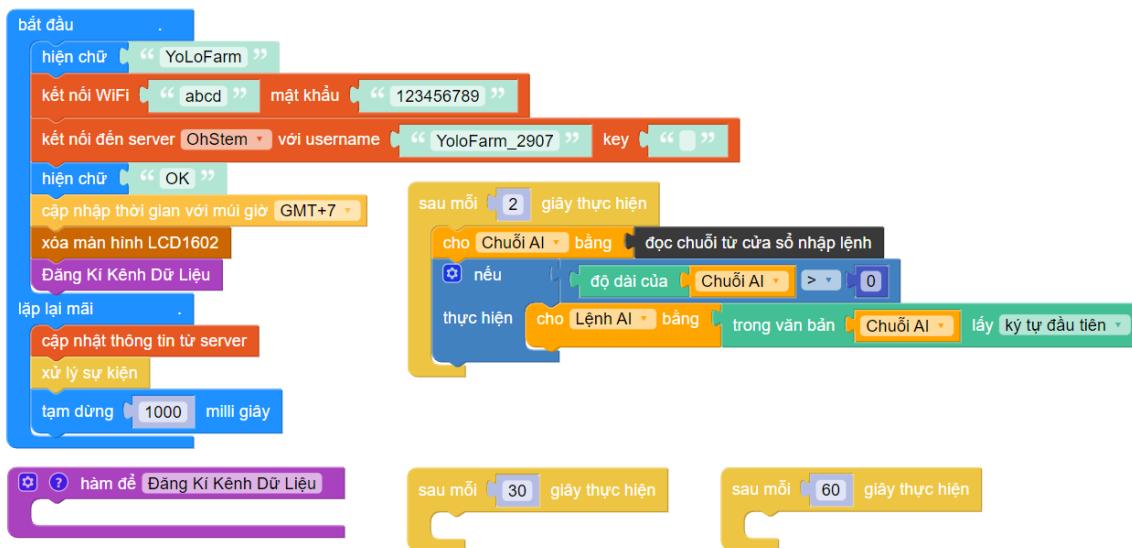
Hình 22.3: Thư viện cho việc lập trình trong dự án Yolo:Farm

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 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 22.4: Khởi tạo các tính năng chính trong chương trình

Bước tiếp theo, chúng ta cần bổ sung vào chương trình một chương trình con chuyên cho việc đăng kí và nhận dữ liệu từ một kênh trên bảng điều khiển IoT và khôi lệnh sự kiện để xử lý cho phần trí tuệ nhân tạo. Hướng dẫn cho phần này đã được trình bày ở Chương 4 và sẽ không trình bày lại ở phần này.



Hình 22.5: Khung chương trình cho dự án Yolo:Farm

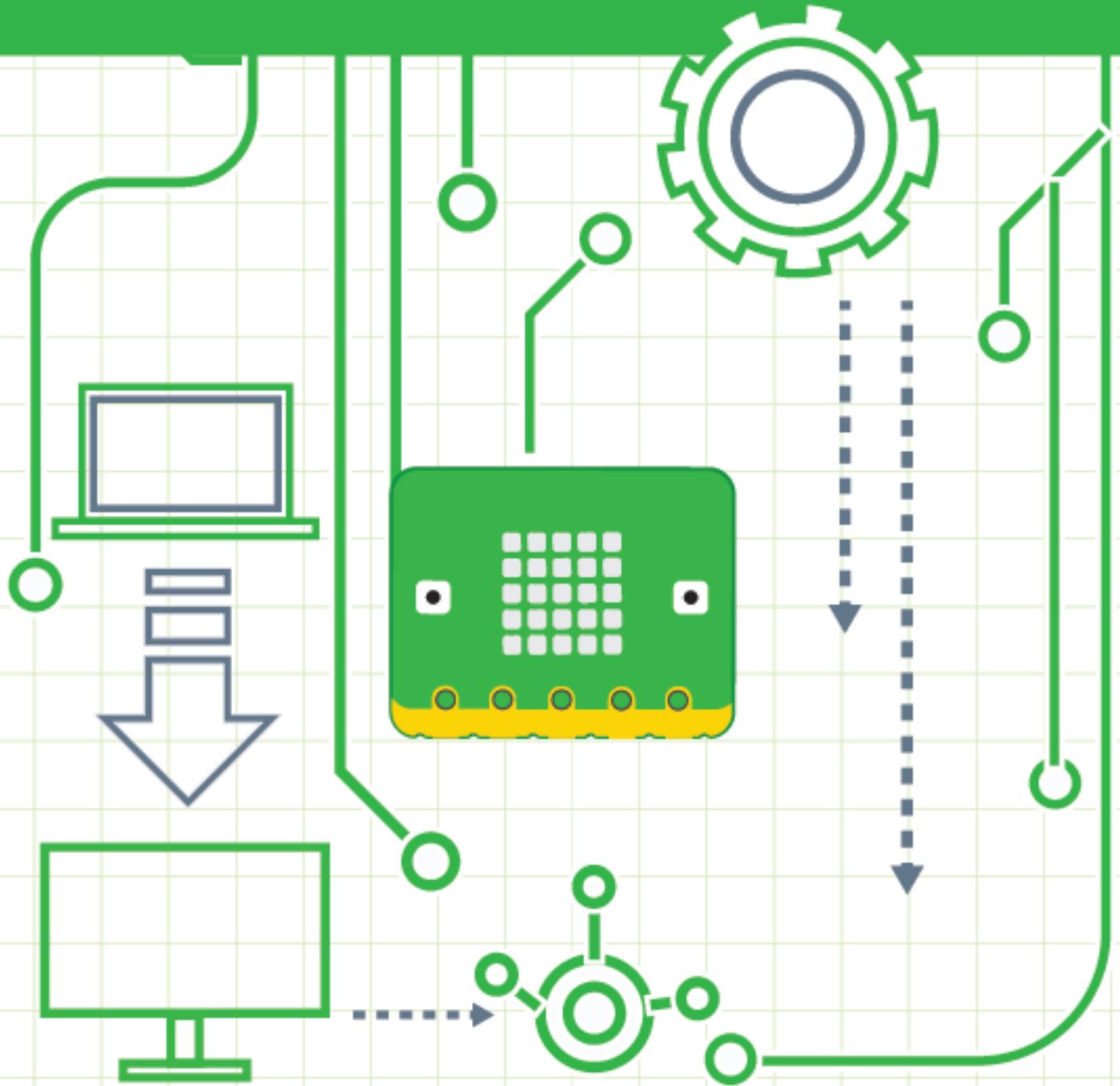
Trong các bài hướng dẫn tiếp theo, chương trình của chúng ta sẽ tập trung chính ở các khối lệnh chu kì và hàm **Đăng Kí Kênh Dữ Liệu**. Với các tính năng có liên quan đến trí tuệ nhân tạo, khối lệnh định kì xử lý 2 giây sẽ được sử dụng và xử lý tiếp theo sẽ dựa trên giá trị của biến **Lệnh AI**. Chương trình khung của dự án được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#/share/yolobit/2D6HzM7CggpnpROxZ38lGrJ9JZ8>

Khi sử dụng chương trình này, bạn đọc cần đảm bảo 4 thư viện liệt kê ở Hình 22.3 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 23

Giám sát môi trường



1 Giới thiệu

Trong bài hướng dẫn này, chúng ta sẽ tập trung vào tính năng đầu tiên của dự án Yolo:Farm, là giám sát điều kiện môi trường liên quan đến trồng trọt. Các thông số liên quan tới không khí (nhiệt độ và độ ẩm), đất và ánh sáng sẽ được giám sát định kì trong dự án.



Hình 23.1: Giám sát các thông tin về môi trường trong dự án Yolo:Farm

Trong phần đầu của tính năng này, chúng ta sẽ tiếp cận với các tính năng truyền thông của một ứng dụng nông nghiệp 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, dữ liệu quan trắc này sẽ được gửi lên OhStem server, để người dùng có thể theo dõi được điều kiện trồng trọt ở bất kỳ nơi đây. Kết quả này có được là nhờ công nghệ kết nối vạn vật khi áp dụng vào nông nghiệp thông minh.

2 Hiển thị thông tin trên LCD

Trước khi gửi dữ liệu lên OhStem server, chúng ta sẽ hiển thị thông tin của các cảm biến trên màn hình LCD kí tự trước. Hiện tại trong hệ thống có 4 loại cảm biến khác nhau cần hiển thị, và được bố trí như sau:

- Nhiệt độ và Độ ẩm không khí sẽ hiển thị ở dòng 1
- Độ ẩm đất và Cường độ ánh sáng sẽ hiển thị ở dòng 2

Do mỗi hàng chỉ hiển thị được tối đa 16 kí tự, chúng ta sẽ dùng những kí hiệu ngắn để biểu diễn cho phần thông tin, sau đó sẽ tới phần giá trị của nó. Chúng tôi gợi ý cho bạn đọc cách sắp xếp thông tin như sau:

0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	R	T	:	2	9	.	5	*	C		R	H	:	7	2	%
1	L	U	X	:	2	3	4	5			S	M	:	4	5	%

Hình 23.2: Hiển thị thông tin trên màn hình LCD kí tự

Trong cấu trúc trên, **RT** (Relative Temperature) tượng trưng cho nhiệt độ không khí, **RH** (Relative Humidity) tượng trưng cho độ ẩm không khí. Hai thông tin còn lại lần lượt là **LUX** và **SM** (Soil Moisture), tượng trưng cho cường độ ánh sáng và độ ẩm đất. Các thông tin về độ ẩm sẽ được chuyển qua đơn vị phần trăm (%), trong khi đó, thông tin về cường độ ánh sáng sẽ không có đơn vị. Chúng ta sẽ hiển thị dữ liệu thô từ cảm biến ánh sáng lên màn hình LCD.

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, 4 biến số,lần lượt là **RT**, **RH**, **SM** và **LUX** sẽ được tạo thêm, như kết quả ở hình sau đây.



Hình 23.3: 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 23.4: 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 23.2, bạn đọc có thể dễ dàng định vị được tọa độ để hiện thị các thông tin cho chính xác. Các câu lệnh cần thiết cho việc này được trình bày như sau:

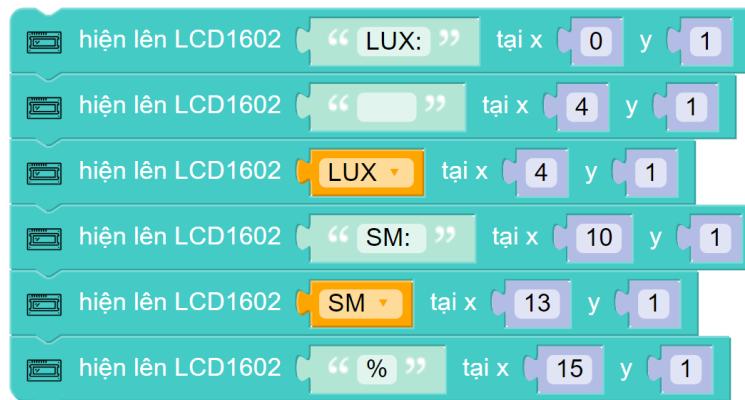


Hình 23.5: Hiển thị dòng đầu tiên trên LCD

Bạn đọc để ý rằng,紧跟其后的是單位的溫度，您將會發現它將會有額外的空格。
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.

Bấm sát theo cấu trúc hiển thị trình bày ở Hình 23.2, chương trình gợi ý cho phần này sẽ như sau:



Hình 23.6: Hiển thị dòng thứ hai trên LCD

Trong đoạn chương trình trên, thông tin về cường độ ánh sáng sẽ được **xóa bằng 6 kí tự khoảng trắng** trước khi hiển thị lại. Cuối cùng, thông tin về độ ẩm đất sẽ được hiển thị từ vị trí có chỉ số là 10.

Bước 5: Cập nhật dữ liệu lên bảng điều khiển IoT.

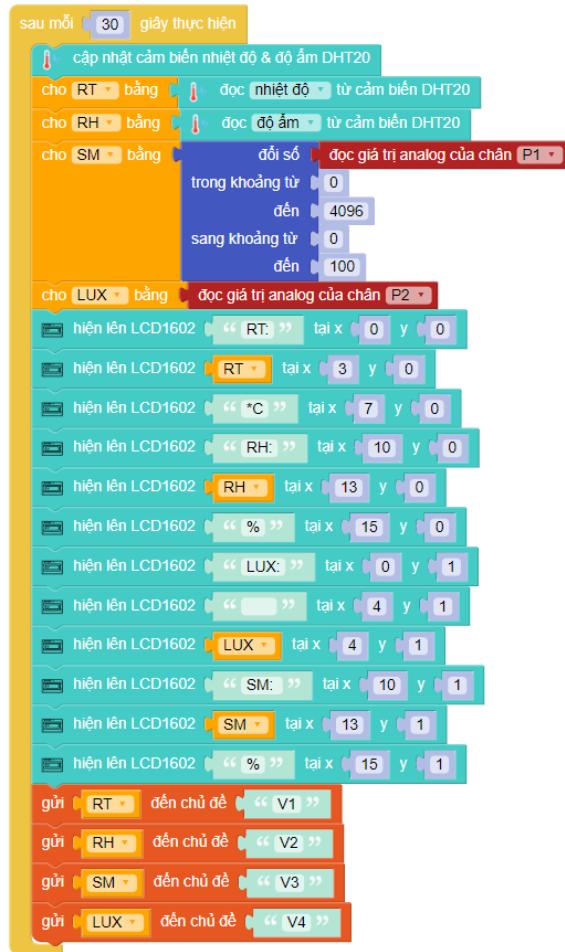


Hình 23.7: Gửi dữ liệu lên bảng điều khiển IoT

Tính năng này thực ra là dễ hiện thực hơn so với việc đọc và hiển thị thông tin lên LCD. Chỉ cần chúng ta gửi giá trị lên đúng kênh dữ liệu, là các thông số sẽ được cập nhật trực tuyến lên bảng điều khiển.

Bước 6: Ghép nối và hoàn thiện chương trình.

Chương trình cho phần giám sát định kì sẽ khá dài, do nó phải xử lý đọc cảm biến, hiển thị trên LCD và gửi dữ liệu lên server. Bạn đọc có thể tách thành những hàm riêng cho dễ nhìn. Ở đây chúng tôi trình bày phiên bản đơn giản nhất, khi tổng hợp tất cả các câu lệnh vào trong khối lệnh xử lý định kì 30 giây.



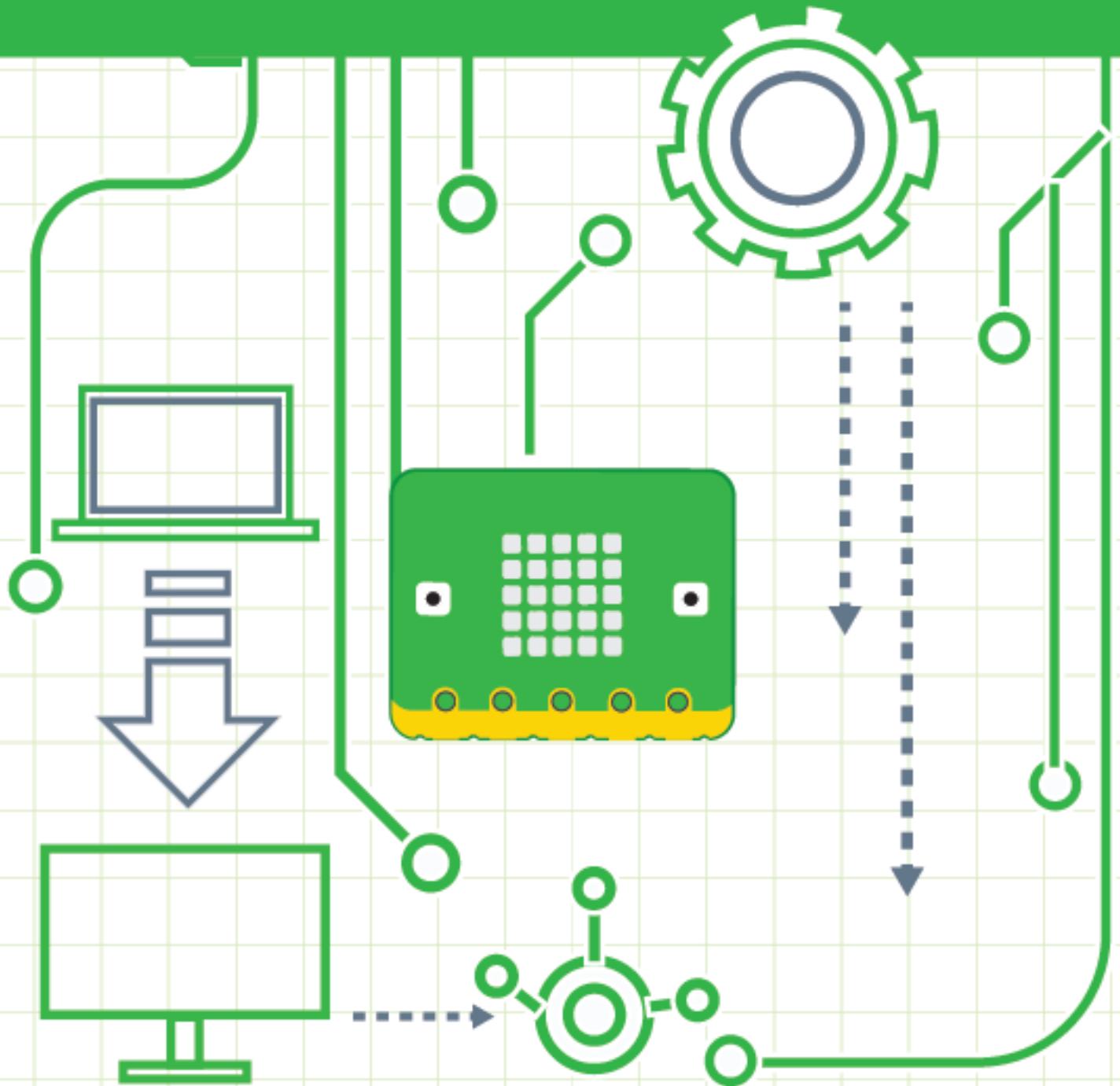
Hình 23.8: Tổng hợp các câu lệnh của phần giám sát môi trường

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

<https://app.ohstem.vn/#!/share/yolobit/2D6FsT8Rvg57Qhn5lph57tSAudO>

CHƯƠNG 24

Tưới tiêu thông minh



1 Giới thiệu

Một điều hiển nhiên rằng nước đóng vai trò hết sức quan trọng trong việc phát triển của cây trồng nói chung và nông nghiệp nói riêng. Vì thế, để cây sinh trưởng và phát triển tốt phải luôn đảm bảo cung cấp đầy đủ nước cho cây. So với việc tưới nước thủ công truyền thống thì việc ứng dụng hệ thống tưới tự động giúp nông dân có thể tiết kiệm tới 60% lượng nước, giảm đáng kể chi phí cho công đoạn này, góp phần quan trọng trong giảm giá thành và tăng tính cạnh tranh của sản phẩm nông nghiệp.



Hình 24.1: Tưới tiêu tự động trong nông nghiệp thông minh

Các béc phun được lắp đặt chìm hẳn xuống mặt cỏ. Chúng vận hành hoàn toàn tự động thông qua nhiều cảm biến về môi trường, trong đó có độ ẩm đất là quan trọng nhất. Điều này giúp cho hệ thống hoạt động trở nên linh hoạt hơn. Khi trời nắng, độ ẩm trong đất thấp, hệ thống sẽ tự động phun nước tưới cho cây trồng. Và dĩ nhiên, khi trời mưa, độ ẩm trong đất cao hệ thống sẽ dừng dưới.

Bên cạnh đó, khả năng tưới theo từng khung giờ nhất định cũng là một nhu cầu không thể thiếu trong nông nghiệp thông minh. Đối với một số loài cây có thời gian trồng ngắn, tưới đúng giờ sẽ giúp các cây phát triển đồng bộ và không tốn nhiều công sức cho việc phân loại sau khi thu hoạch, chẳng hạn như các loại cây cảnh ngày tết chẳng hạn.

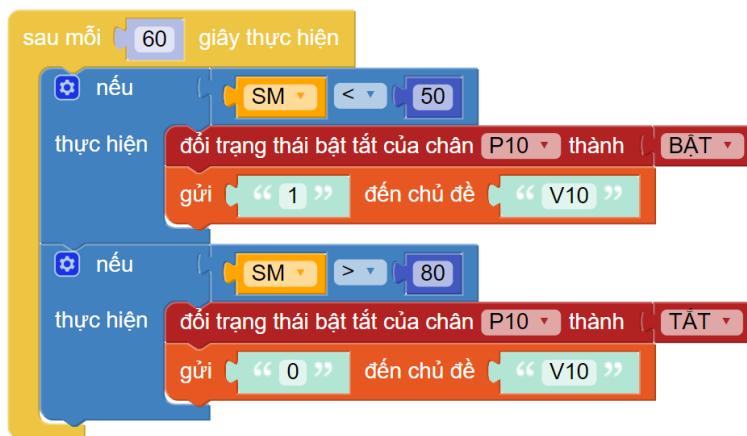
Cuối cùng, là khả năng điều khiển tưới tiêu từ xa thông qua công nghệ kết nối vạn vật. Từ bảng điều khiển IoT, người nông dân không chỉ thấy trạng thái máy bơm là đang bật hay tắt, mà còn có thể chủ động điều khiển nó theo mong muốn của mình.

Trong bài hướng dẫn này, chúng tôi hướng dẫn cả phương pháp kể trên. Tùy vào hoàn cảnh và chức năng của hệ thống, bạn đọc có thể chọn một hoặc nhiều kỹ thuật trong bài này.

2 Tưới tự động

Như đã định nghĩa ở trên, độ ẩm của đất là yếu tố xem xét cho việc bật tắt máy bơm. Hệ thống sẽ phải **định kì** kiểm tra độ ẩm đất để ra quyết định. Thông thường, chu kì kiểm tra này cũng không cần thiết phải quá liên tục. Ở đây, chúng tôi chọn chu kì kiểm tra là 1 phút, hay **60 giây**.

Tiếp theo, chúng ta cần phải định nghĩa một ngưỡng để bật máy bơm (đất khô) và một ngưỡng để tắt máy bơm (đất đủ ẩm). Về 2 thông tin này, nó thực sự phụ thuộc vào từng loại cây trồng mà bạn đọc cần phải tham khảo thêm. Theo như chúng tôi tham khảo, hoa cúc cần có độ ẩm đất từ 50% đến 80% là lý tưởng cho sự phát triển. Chương trình điều khiển tự động máy bơm thứ nhất (nối với chân P10) sẽ như sau:



Hình 24.2: Tưới tiêu tự động theo độ ẩm đất

Một khôi lệnh sự kiện mới sẽ được hiện thực thêm vào chương trình. Biến **SM** dùng để lưu lại thông tin độ ẩm ở bài trước sẽ được sử dụng lại trong câu lệnh điều kiện để bật tắt máy bơm. Mỗi khi máy bơm thay đổi trạng thái, chúng ta sẽ gửi thông tin này lên OhStem server để biểu diễn trạng thái hiện tại của máy bơm trên bảng điều khiển IoT. Theo thiết kế trên server, kênh dữ liệu cho máy bơm 1 là **V10**.

Do chúng ta kiểm tra giá trị biến **SM** mỗi 60 giây một lần, trong trường hợp hợp xấu nhất, khi độ ẩm là 79 và chuẩn bị chuyển sang 80, máy bơm vẫn bật. Phải một phút sau nó mới nhận ra rằng độ ẩm đã vượt ngưỡng và tắt máy bơm. Chúng ta gọi 60 giây là độ trễ tối đa của hệ thống tưới tiêu tự động với chương trình gợi ý như trên. Với các loại cây trồng cần độ chính xác về độ ẩm cao hơn, bạn sẽ phải giảm chu kì kiểm tra xuống.

3 Tưới theo giờ

Đối với các loại cây trồng có sức sống khỏe và ít chịu ảnh hưởng của thời tiết, chẳng hạn như hoa vạn thọ, chỉ cần tưới đúng giờ là cây có thể phát triển tốt. Giả sử rằng, hoa vạn thọ cần tưới 2 lần trong ngày. Lần đầu tiên là buổi sáng, từ 7g00 đến 7g15, và lần thứ 2 vào buổi chiều, từ 14g đến 14g30.

Thường việc tưới tiêu theo giờ, cũng chỉ quan tâm đến đơn vị phút. Nên để thuận tiện cho việc lập trình, trước tiên chúng ta cần đổi tất cả sang đơn vị phút, lúc đó:

$$7g00 = 7 * 60 + 0 = 420$$

$$7g15 = 7 * 60 + 15 = 435$$

$$14g00 = 14 * 60 + 0 = 840$$

$$14g30 = 14 * 60 + 30 = 870$$

Việc lập trình bây giờ khá đơn giản, chúng ta sẽ làm 1 phép toán chuyển đổi thời gian hiện tại sang phút, trước khi so sánh với các ngưỡng đã định nghĩa ở trên. Một chương trình gợi ý cho xử lý ở phần này được trình bày như sau:



Hình 24.3: Tưới tiêu tự động theo nhiều khung giờ khác nhau

Ở đây, chúng tôi tạo thêm 1 biến **Thời Gian** để đổi toàn bộ thông tin sang phút, trước khi bắt đầu so sánh và điều khiển trạng thái của máy bơm thứ 2, đang nối với chân P13 của mạch Yolo:Bit. Cũng tương tự như máy bơm thứ nhất, trạng thái của nó được gửi về bảng điều khiển IoT sau mỗi lần bật tắt. Kênh dữ liệu dùng cho máy bơm thứ 2 này là **V11**.

4 Tưới tiêu công

Trong một số trường hợp, bạn có thể cần tính năng này. Thực ra, tính năng này khá đơn giản, chúng ta chỉ đơn giản là đăng ký nhận dữ liệu từ 2 kênh cho máy bơm là **V10** và **V11** rồi điều khiển nó tương ứng. Tính năng này sẽ được hiện thực trong khối hàm **Đăng Kí Kênh Dữ Liệu** đã có sẵn trong chương trình khung, như sau:



Hình 24.4: Tưới tiêu bằng tay với nút nhấn trên bảng điều khiển IoT

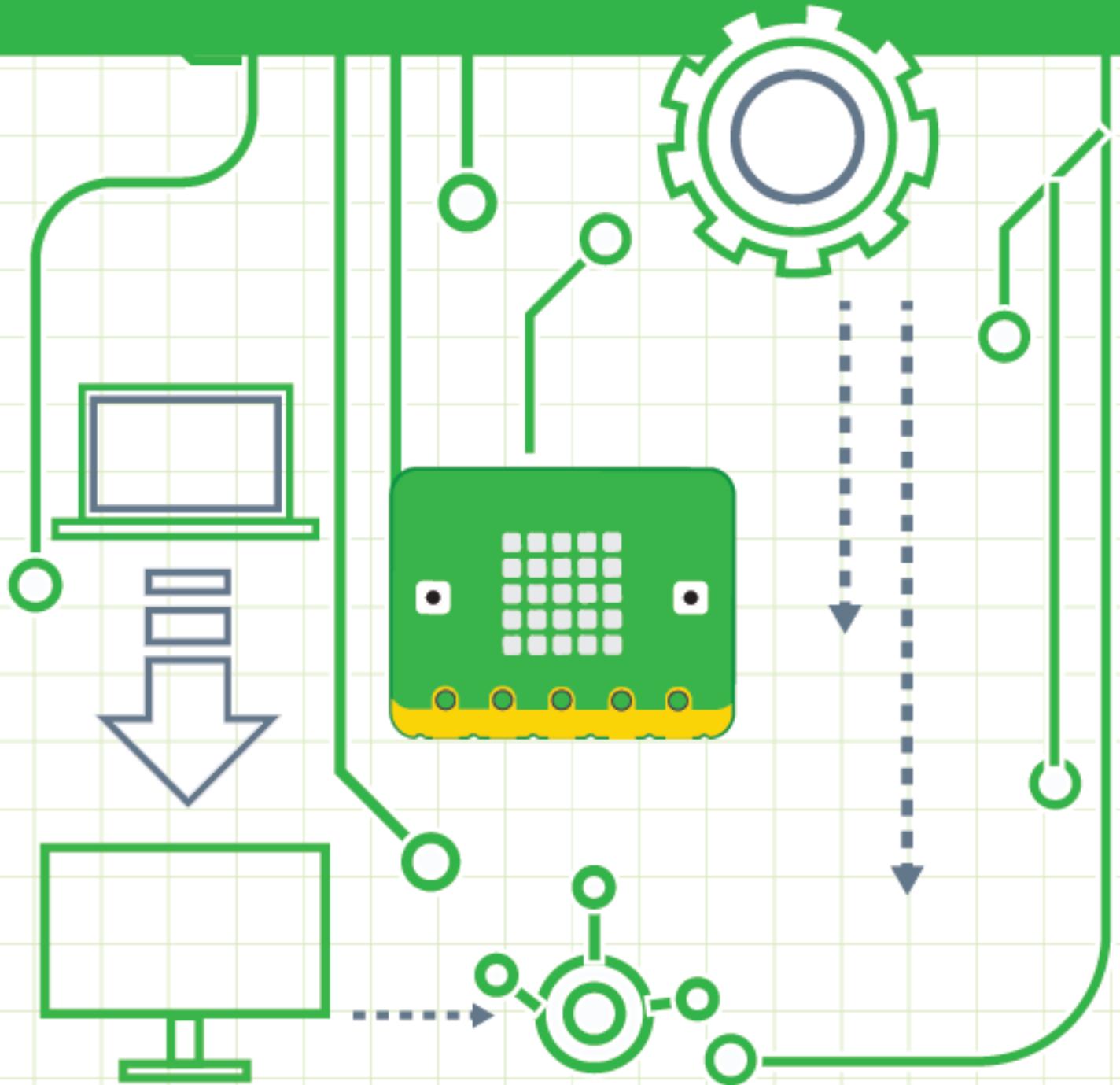
Chương trình tổng hợp cho đến bài này được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#/share/yolobit/2D56HWnLInzDPqVTGq7Y31X6bvw>

Lưu ý: Bạn đọc chỉ có thể chọn 1 trong 3 phương pháp tưới tiêu khi sử dụng chương trình này (bằng cách xóa 2 phương pháp còn lại). Vừa tưới tiêu tự động và vừa tưới tiêu bằng tay, chúng tôi xin để dành cho bạn đọc tự sáng tạo.

CHƯƠNG 25

Phân tích dữ liệu

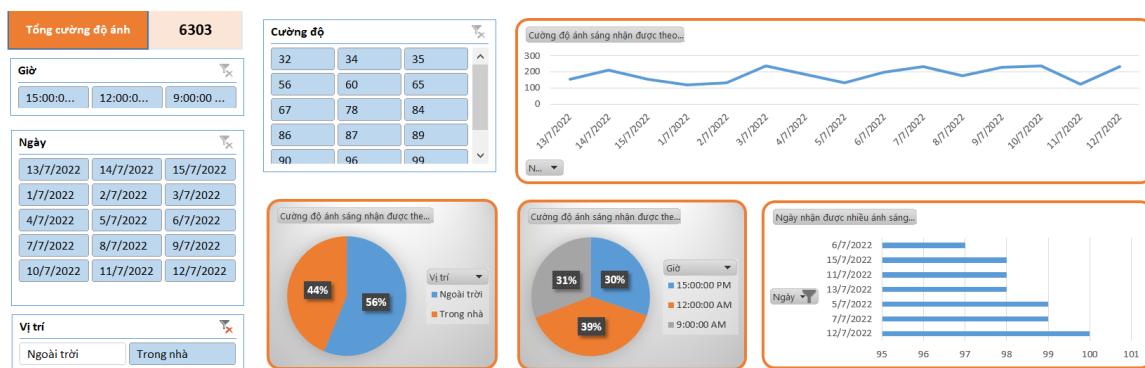


1 Giới thiệu

Một trong những lợi thế của bài toán kết nối vạn vật, là khi dữ liệu có thể thu thập được tự động và thường xuyên, nó sẽ mở ra các hướng liên quan đến phân tích và nội suy dữ liệu. Với các dữ liệu từ cảm biến gửi lên từ Yolo:Bit, các công cụ tính toán có thể được sử dụng để cung cấp nhiều thông tin có ý nghĩa hơn.

Đầu tiên của chức năng phân tích dữ liệu này, là ngày tăng trưởng của cây trồng sẽ được tính toán và lưu vào biến số **GDD**. Giá trị của GDD sẽ được gửi định kì lên server để nông dân theo dõi về thời gian thực sự mà cây trồng có đủ nắng.

Bên cạnh đó, dữ liệu còn có thể được gửi trực tiếp qua phần mềm Microsoft Excel, một công cụ tính toán đủ mạnh cho các vấn đề liên quan đến phân tích và thống kê dữ liệu. Từ mạch Yolo:Bit, dữ liệu có thể được ghi lại và được phân tích thêm trên Excel. Thực ra, bên cạnh Python, Excel là một công cụ mạnh mẽ cho việc phân tích dữ liệu. Hình ảnh bên dưới là kết quả chúng ta có thể đạt được trong bài hướng dẫn này.

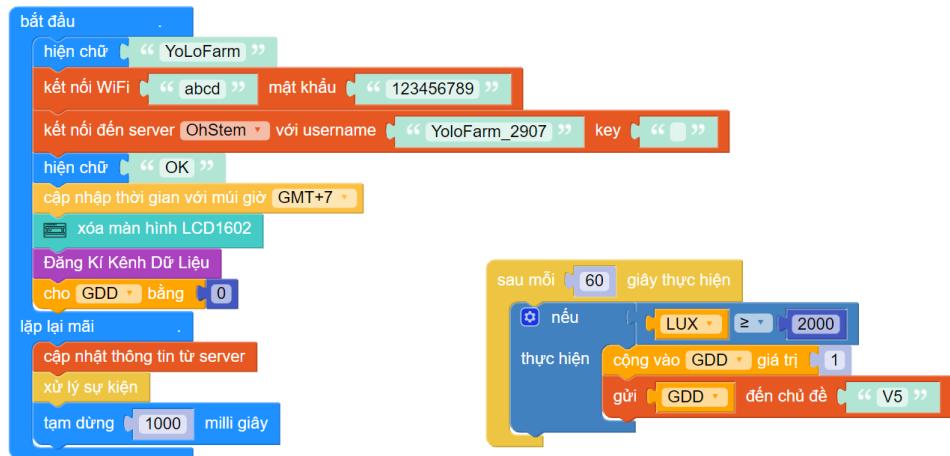


Hình 25.1: Phân tích dữ liệu với Microsoft Excel

Với phần mềm Microsoft Excel, dữ liệu thô có thể được thống kê (tìm giá trị lớn nhất, nhỏ nhất, trung bình) hoặc biểu diễn lại dưới dạng đồ thị hoặc biểu đồ cột để người quản lý có cái nhìn tổng quan nhất về điều kiện nuôi trồng. Thông tin này sẽ là kinh nghiệm để điều chỉnh hệ thống, hướng đến mục tiêu sản xuất nông nghiệp hiệu quả và ổn định trong tương lai.

2 Cập nhật GDD

Như đã trình bày ở Chương 4, cảm biến ánh sáng sẽ được dùng để tính ngày tăng trưởng của cây trồng. Nguyên tắc ở đây là chúng ta sẽ **định kì mỗi phút kiểm tra cảm biến ánh sáng**. Với tính năng cần được thực hiện định kì, chúng ta sẽ dùng thêm 1 khối lệnh sự kiện cho chương trình, như sau:



Hình 25.2: Cập nhật GDD lên OhStem server

Một số lưu ý quan trọng khi hiện thực việc tính toán GDD:

- Biến GDD phải được khởi tạo trong phần **bắt đầu**, do nó được tính tích lũy trong khối lệnh sự kiện.
- GDD hiện tại đang có đơn vị là phút, nếu muốn chuyển đổi sang giờ hoặc ngày, bạn đọc cần xử lý thêm trước khi gửi lên OhStem server.
- Kênh dữ liệu cho GDD, như đã thiết kế ở Chương 6, là **V5**.

3 Phân tích dữ liệu

Với tính năng này, chúng tôi muốn minh họa cho kết quả nối tiếp từ công nghệ kết nối vạn vật. Khi dữ liệu về trồng trọt có thể được quan trắc định kì và thường xuyên, nó sẽ là kho dữ liệu quý giá cho việc phân tích dữ liệu và rút kinh nghiệm trong tương lai.

Trong hướng dẫn này, chúng tôi sử dụng tính năng **Biểu đồ** có sẵn trên môi trường lập trình của Yolo:Bit để ghi lại dữ liệu về cảm biến ánh sáng trước khi tải nó về dưới dạng file Excel. Tiếp theo đó, file này sẽ được sử dụng để tính toán các giá trị trung bình, lớn nhất hoặc nhỏ nhất, là những tính năng cơ bản của việc phân tích dữ liệu. Trình tự hiện thực các bước này được trình bày từng bước như bên dưới.

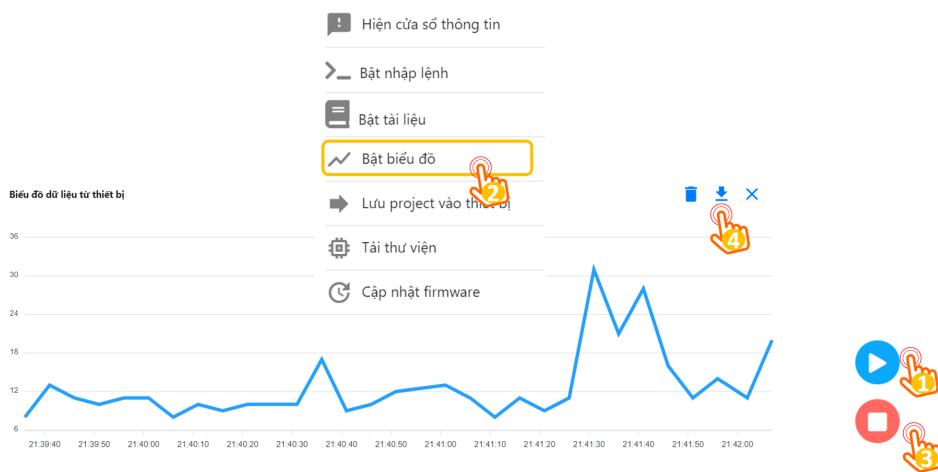
Bước 1: Bổ sung câu lệnh gửi dữ liệu lên biểu đồ.

Trong khi giám sát giá trị ánh sáng, chúng ta cũng sẽ đồng thời gửi nó lên biểu đồ, như sau:



Hình 25.3: Ghi lại thông số ánh sáng trên biểu đồ

Bước 2: Chạy chương trình trên Yolo:Bit và bật tính năng Biểu đồ.



Hình 25.4: Ghi lại thông số ánh sáng trên biểu đồ

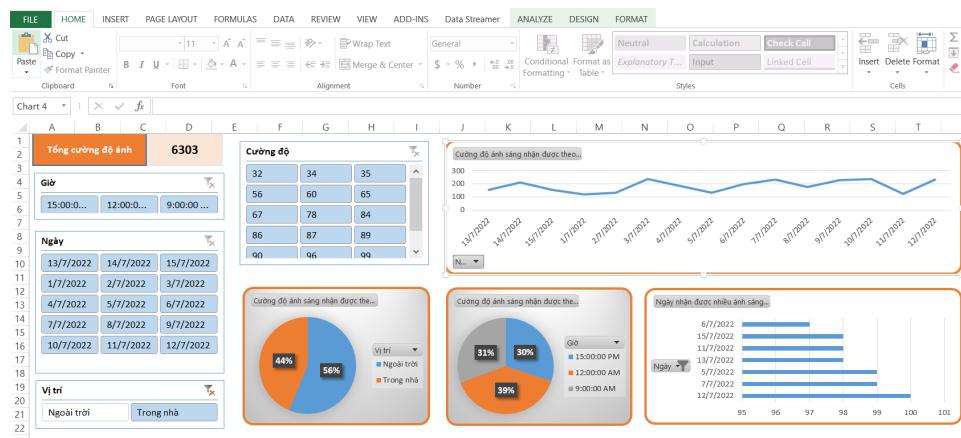
Quy trình để ghi lại dữ liệu được trình bày như trên Hình 25.4. Đầu tiên, chúng ta cần gửi chương trình cho mạch Yolo:Bit. Sau đó, sử dụng tính năng **Bật biểu đồ** nằm trong mục cài đặt. Sau khi ghi đủ dữ liệu cảm biến, chúng ta có thể dừng thực thi chương trình và tải dữ liệu thu thập được dưới dạng file Excel.

Bước 3: Tải file Excel mẫu.

Nhằm hỗ trợ cho bạn đọc nhanh chóng tiếp cận với công cụ phân tích dữ liệu trên Excel, chúng tôi cung cấp file mẫu ở đường dẫn sau đây:

<https://ubc.sgp1.digitaloceanspaces.com/OhStem/ExcelTemplate.xlsx>

Khi mở file excel tải về, bạn sẽ thấy được giao diện mặc định như sau:



Hình 25.5: Giao diện phân tích dữ liệu trên Excel

Bạn có thể thử thao tác trên các nút lựa chọn trên giao diện này, để có thể lọc dữ liệu theo nhiều điều kiện khác nhau.

Bước 4: Tải dữ liệu cảm biến từ hệ thống Yolo:Bit.

Từ file Excel có được ở Bước 2, bạn đọc có thể chép nó vào sheet **Data** của file mẫu, như hình minh họa bên dưới:

	A	B	C	D	E	F
1						
2		Ngày	Vị trí	Giờ	Cường độ	
3	1/7/2022	Ngoài trời	9:00:00 AM	153		
4	1/7/2022	Trong nhà	15:00:00 PM	34		
5	1/7/2022	Ngoài trời	12:00:00 AM	94		
6	1/7/2022	Trong nhà	12:00:00 AM	87		
7	2/7/2022	Ngoài trời	15:00:00 PM	65		
8	2/7/2022	Trong nhà	9:00:00 AM	35		
9	2/7/2022	Ngoài trời	12:00:00 AM	89		
10	2/7/2022	Trong nhà	15:00:00 PM	67		
11	2/7/2022	Trong nhà	15:00:00 PM	32		
12	2/7/2022	Ngoài trời	9:00:00 AM	54		

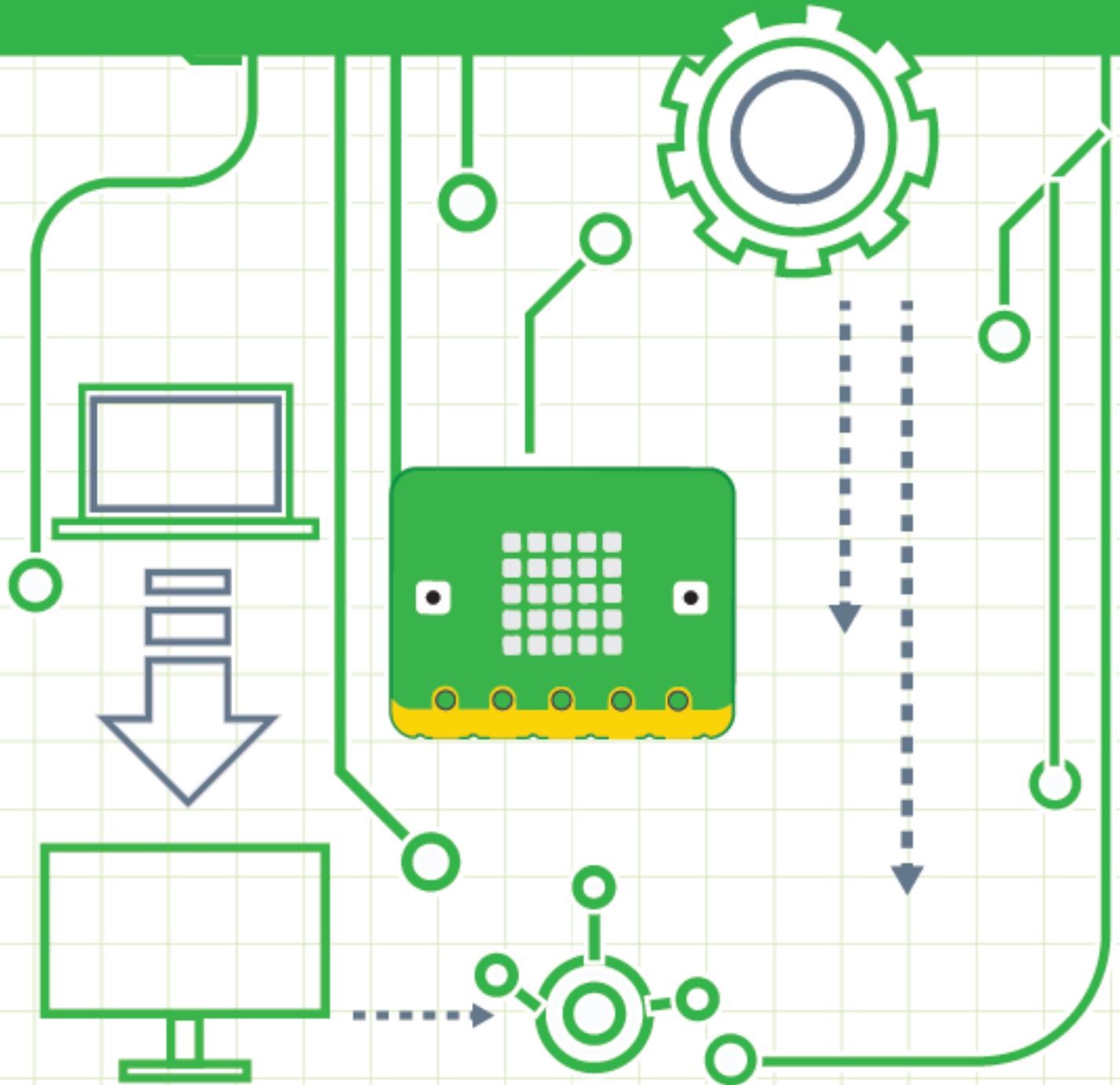
Hình 25.6: Dữ liệu thu được từ mạch Yolo:Bit

Dữ liệu từ cảm biến có thể được chép vào cột **Cường độ**, để phân tích ở sheet **Dashboard**. Tuy nhiên, mỗi khi chọn vào Dashboard, bạn cần chọn vào biểu tượng biểu đồ, chọn vào **ANALYZE** và chọn vào **Refresh** để cập nhật lại thông tin mới. Chương trình cho bài hướng dẫn này được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2D569zc5nTQBNcKUSK7n9V7ToHf>

CHƯƠNG 26

Phát hiện bất thường



1 Giới thiệu

Một trong những ưu điểm của hệ thống trí tuệ nhân tạo khi ứng dụng vào nông nghiệp, là khả năng nhận dạng ra các dấu hiệu bất thường ở cây trồng một cách linh động, như phân biệt các loại sâu bệnh từ hình ảnh cây trồng. Tính năng này thường không dễ hiện thực với cảm biến thông dụng. Khi sử dụng cảm biến, nó thường phải được lắp đặt với số lượng nhiều và cân chỉnh chính xác để có thể đo được được giá trị hợp lệ. Điều này sẽ gây nhiều khó khăn và tốn kém cho tính năng phát hiện bất thường bằng giải pháp sử dụng cảm biến.

Với công nghệ trí tuệ nhân tạo, điều này có thể được hiện thực một cách rất linh động, khi người dùng có thể thu thập những hiện tượng bất thường từ cây trồng, để huấn luyện cho hệ thống. Qua thời gian, khi dữ liệu thu thập đã trở nên phong phú, hệ thống có thể nhận diện được hầu hết các vấn đề đối với hoạt động nông nghiệp. Trí tuệ nhân tạo đang là hướng tiếp cận cao cấp dành cho nông nghiệp công nghệ 4.0.



Hình 26.1: Robot AI được ứng dụng trong nông nghiệp thông minh

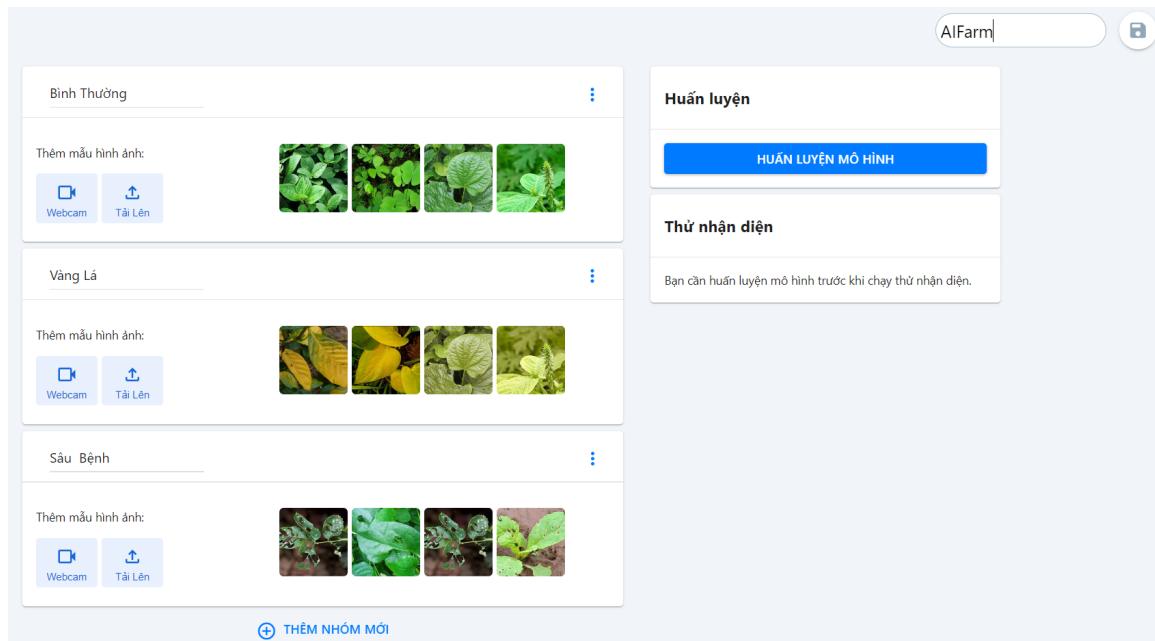
Trong nông nghiệp, các hệ thống AI thường dựa vào việc xử lý và nhận dạng ảnh chụp từ camera. Các camera này được gắn trên các Robot để di chuyển bên trong khu vực trồng trọt. Hành vi của Robot kết hợp với bộ xử lý AI sẽ thay cho con người trong việc giám sát tình trạng của cây trồng.

Trong bài hướng dẫn này, một bộ xử lý AI đơn giản để nhận diện ra các vấn đề của cây trồng, như bị vàng lá hay bị sâu bệnh sẽ được hiện thực. Kết quả của việc xử lý AI sẽ được gửi lên server IoT để giám sát từ xa. Các mục tiêu hướng dẫn trong bài này như sau:

- Huấn luyện mô hình AI với dữ liệu thu thập
- Hiện thực chương trình AI chạy trên máy tính
- Tích hợp chương trình AI trên Yolo:Bit

2 Huấn luyện AI

Trong từng trường hợp cụ thể cho nông nghiệp, hệ thống AI cần có những kiến thức chuyên dụng trước khi triển khai. Tuy nhiên, thông thường bạn đọc sẽ mở rộng số lượng hình ảnh và số lượng kiến thức để huấn luyện cho hệ thống. Ở hình minh họa bên dưới, chúng tôi minh họa cho việc thu thập dữ liệu và huấn luyện cho 3 loại kiến thức, bao gồm **Bình Thường**, **Vàng Lá** và **Sâu Bệnh**.



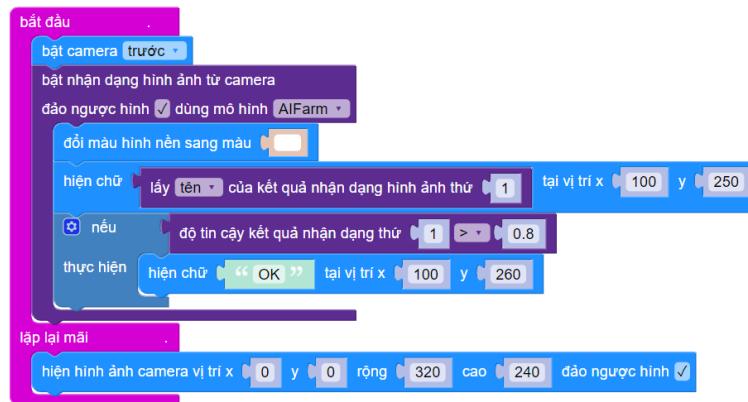
Hình 26.2: Thu thập và huấn luyện cho hệ thống AI

Một số lưu ý khi xây dựng hệ thống AI được trình bày như sau:

- Để thêm 1 kiến thức, nhấn vào nút **THÊM NHÓM MỚI**.
- Để xóa 1 kiến thức hoặc xóa hình ảnh thu thập của nó, chọn vào biểu tượng nút 3 chấm ở gốc bên phải của mỗi kiến thức.
- Huấn luyện (nhấn vào nút **HUẤN LUYỆN MÔ HÌNH**), kiểm tra hệ thống và đặt tên cho bộ não nhân tạo (chẳng hạn như AIFarm, như ví dụ ở Hình 26.2).

3 Chương trình AI

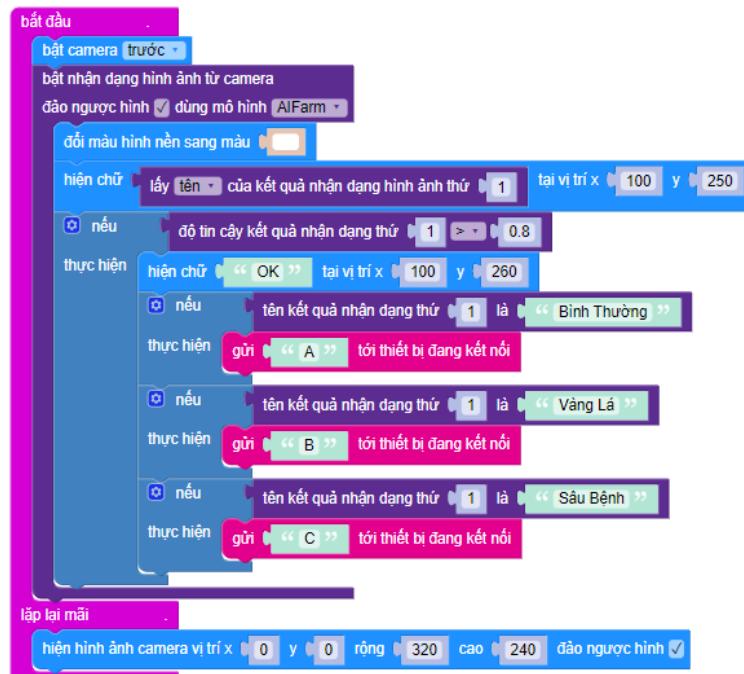
Với chương trình AI chạy trên máy tính (hoặc thiết bị di động), bên cạnh việc kiểm tra thông tin về kết quả nhận dạng, bạn đọc cũng có thể kiểm tra thêm về xác suất của kết quả nhận dạng. Như ví dụ bằng đoạn chương trình như sau, khi xác suất của việc nhận dạng lớn hơn 80% (0.8), chúng ta sẽ in thêm chữ **OK** ngay bên dưới kết quả nhận dạng:



Hình 26.3: Kiểm tra xác suất của việc nhận dạng

Lý do mà chúng ta phải kiểm tra xác suất là để tăng độ tin cậy của kết quả nhận dạng. Khi triển khai thực tế, sẽ có nhiều yếu tố ảnh hưởng đến việc nhận dạng của hệ thống, chẳng hạn như điều kiện ánh sáng (camera bị ngược sáng hoặc trong điều kiện thiếu ánh sáng). Lúc này, xác suất của kết quả sẽ rất thấp và không nên xử lý lúc này. Khi điều kiện gần giống với khi chúng ta huấn luyện, xác suất nhận dạng sẽ cao và đây mới là dữ liệu đáng tin cậy để xử lý.Thêm nữa, trong thực tế sẽ phát sinh nhiều trường hợp ngoại lệ, nên việc kiểm tra thêm xác suất sẽ thực sự có ích cho hệ thống.

Tiếp tục bổ sung thêm các câu lệnh điều kiện bên trong khối kiểm tra độ tin cậy, chúng ta có chương trình xử lý AI như sau:



Hình 26.4: Chương trình nhận dạng AI kết hợp với xác suất

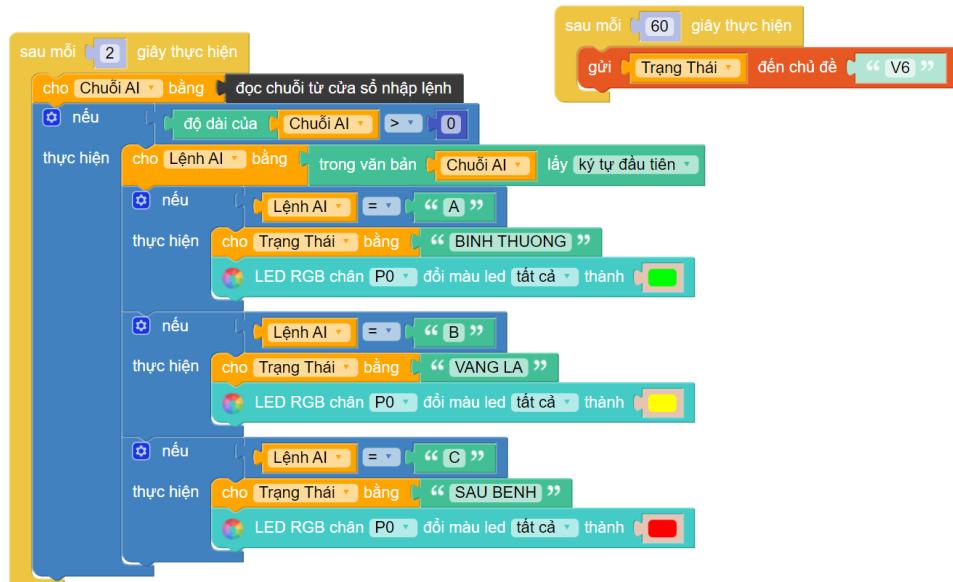
4 Chương trình trên Yolo:Bit

Chương trình xử lý chính bên Yolo:Bit sẽ nằm ở phần đọc dữ liệu từ cửa sổ lệnh mỗi 2 giây. Để tiện cho việc gửi dữ liệu lên server, biến **Trạng Thái** được tạo thêm cho phần xử lý này. Chương trình gợi ý cho khôi lệnh dành cho phần nhận lệnh trí tuệ nhân tạo được gợi ý như sau:



Hình 26.5: Nhận và xử lý lệnh AI

Giá trị của biến **Trạng Thái** sẽ được gửi lên server, nên giá trị của nó không có các kí tự đặc biệt. Chúng ta có thể xài tiếng Việt không dấu cho biến số này. Do chúng ta sẽ không thường xuyên gửi thông tin này lên server, nên một khôi lệnh định kì với chu kỳ dài hơn (chẳng hạn như 60 giây) sẽ được tạo thêm để gửi thông tin về trạng thái của cây trồng.



Hình 26.6: Gửi trạng thái lên server và hiển thị đèn

Để hệ thống thêm phong phú, đèn RGB sẽ được dùng để hiển thị trạng thái của cây trồng, bằng cách hiển thị màu xanh, vàng và đỏ. Kênh dữ liệu dành cho trạng

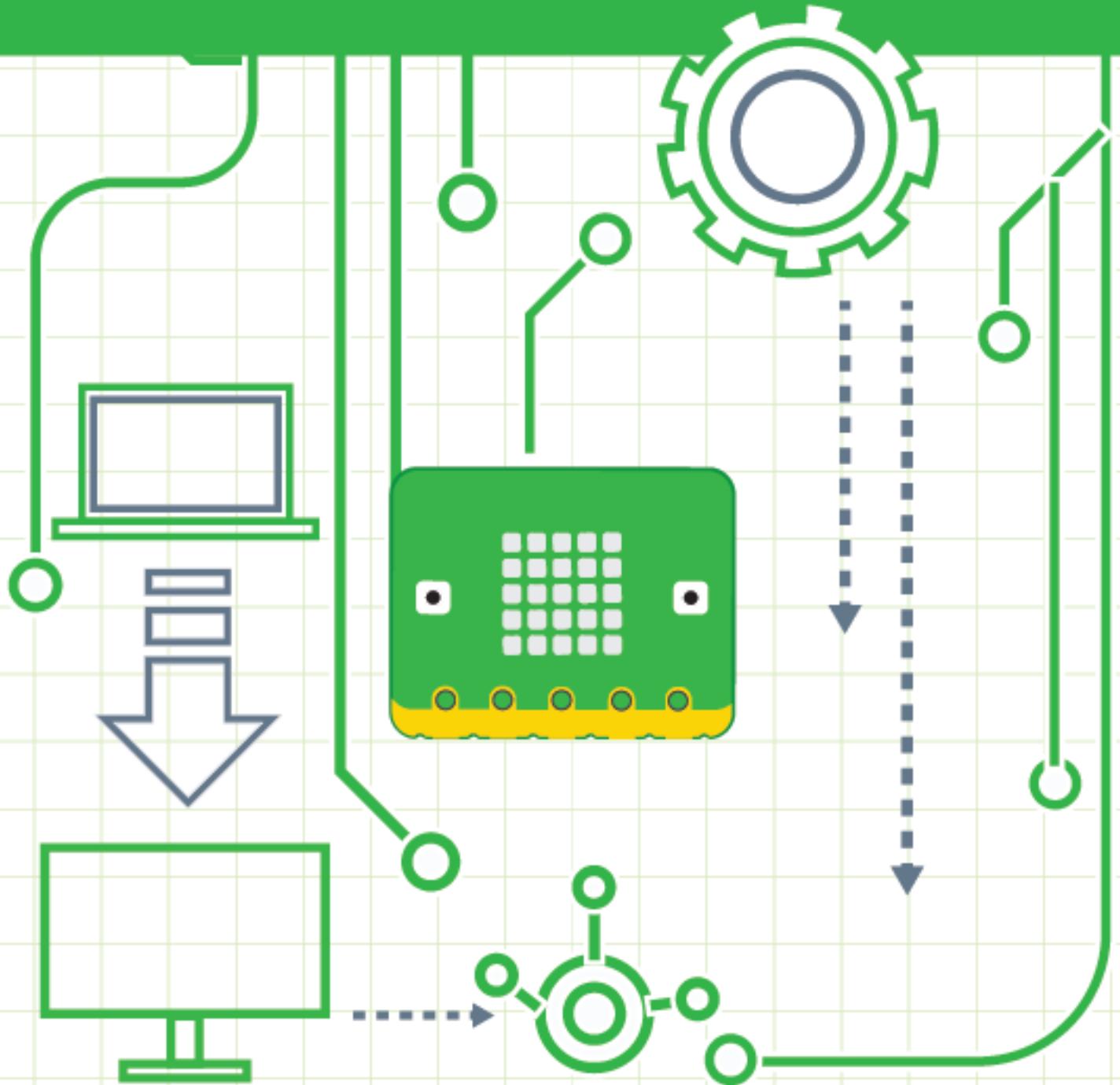
thái cây trồng được cấu hình ở kênh **V6**. Toàn bộ chương trình hiện tại, được chia sẻ ở đường dẫn sau đây.

<https://app.ohstem.vn/#!/share/yolobit/2D4yGOgxczOIhBNmzf3kRHham3b>

Câu lệnh gửi dữ liệu lên kênh V6, bạn đọc có thể kết hợp nó với bất kì khối xử lý sự kiện nào đã có trước đó. Tuy nhiên, trong trường hợp muốn chương trình dễ đọc và dễ phát triển, 1 khối lệnh sự kiện mới nên được sử dụng.

CHƯƠNG 27

Phân loại thu hoạch



1 Giới thiệu

Phân loại trái cây sau khi thu hoạch là nhu cầu tất yếu trong nông nghiệp. Không chỉ loại bỏ những thành phẩm không đạt chất lượng (chưa chín, khôi lượng không đủ), việc phân loại sản phẩm còn giúp cho quá trình định giá thành phẩm, đóng gói xuất khẩu, và do đó, góp phần tăng giá trị của trái cây trên thị trường. Những trái cây có chất lượng tốt và đồng đều thường sẽ bán được giá cao hơn phần còn lại.



Hình 27.1: Phân loại trái cây khi thu hoạch

Trước đây phương pháp phân loại sản phẩm truyền thống yêu cầu không gian làm việc rộng, và số lượng người tham gia vào phân loại lớn, thời gian làm việc lâu và dễ sai sót. Chính vì thế, trong nông nghiệp công nghệ, dây chuyền phân loại sản phẩm tự động sẽ vô cùng cần thiết. Không chỉ giúp giảm thiểu số lượng nhân công, theo thống kê có thể lên đến 80%, mà năng suất hoạt động tăng từ 3 đến 5 lần so với việc phân loại truyền thống trước đây.

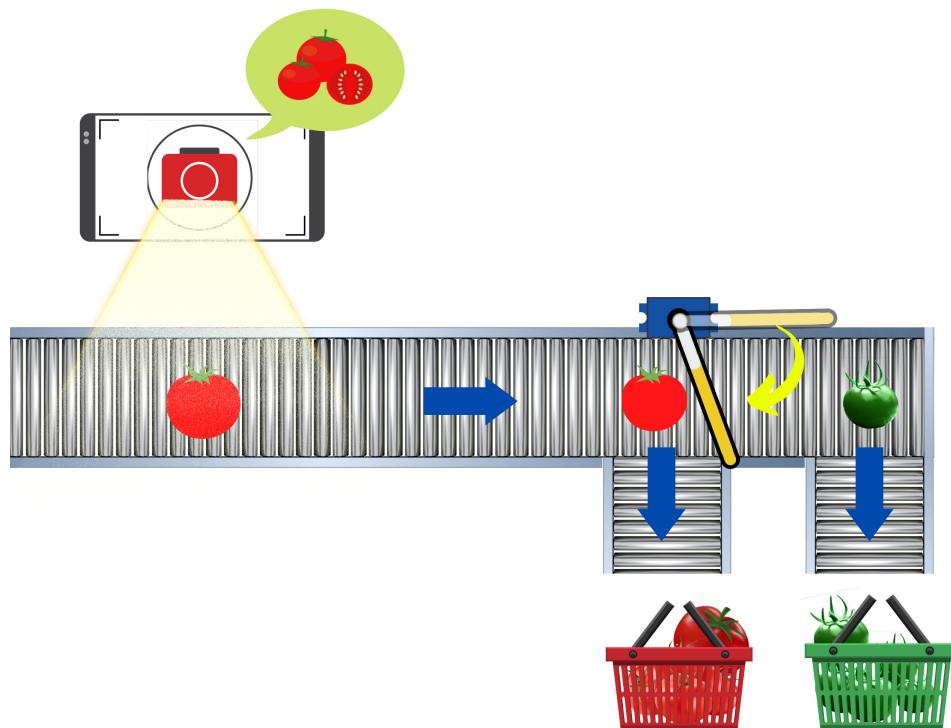
Công nghệ chính trong phân loại sản phẩm hiện nay vẫn là dựa trên trí tuệ nhân tạo. Lợi thế lớn nhất của công nghệ này là tính linh động, khi người dùng có thể chủ động dạy cho hệ thống để nó có thể nhận dạng đâu là hoa quả đủ tiêu chuẩn cho việc thu hoạch. Các công nghệ liên quan đến cảm biến (nhận dạng theo màu sắc, cân khôi lượng) sẽ bổ sung thêm thông tin để các quyết định về trí tuệ nhân tạo tăng thêm độ tin cậy.

Trong bài hướng dẫn này, chúng ta sẽ tiếp tục tích hợp thêm tính năng cuối cùng và dự án YoloFarm, huấn luyện thêm phần trí tuệ nhân tạo cho việc phân loại trái cây. Ở đây, chúng tôi chỉ minh họa một ngữ cảnh đơn giản, là phân biệt cà chua sống và cà chua chín. Một động cơ RC Servo sẽ được điều khiển để định hướng cho việc di chuyển của 2 loại cà chua trên băng chuyền. Các nội dung trình bày trong bài này được tổng hợp như sau:

- Tổng quan hệ thống phân loại trái cây
- Huấn luyện hệ thống trí tuệ nhân tạo
- Hiện thực hệ thống

2 Tổng quan hệ thống

Tổng quan về hệ thống phân loại trái cây có thể ứng dụng vào việc giảng dạy và minh họa cho ứng dụng này được trình bày như Hình 27.2. Bên cạnh hệ thống băng chuyền hoạt động độc lập (có thể luôn luôn bật), 2 thành phần chính trong hệ thống của chúng ta sẽ là một điện thoại di động và một động cơ RC.



Hình 27.2: Tổng quan hệ thống phân loại hoa quả

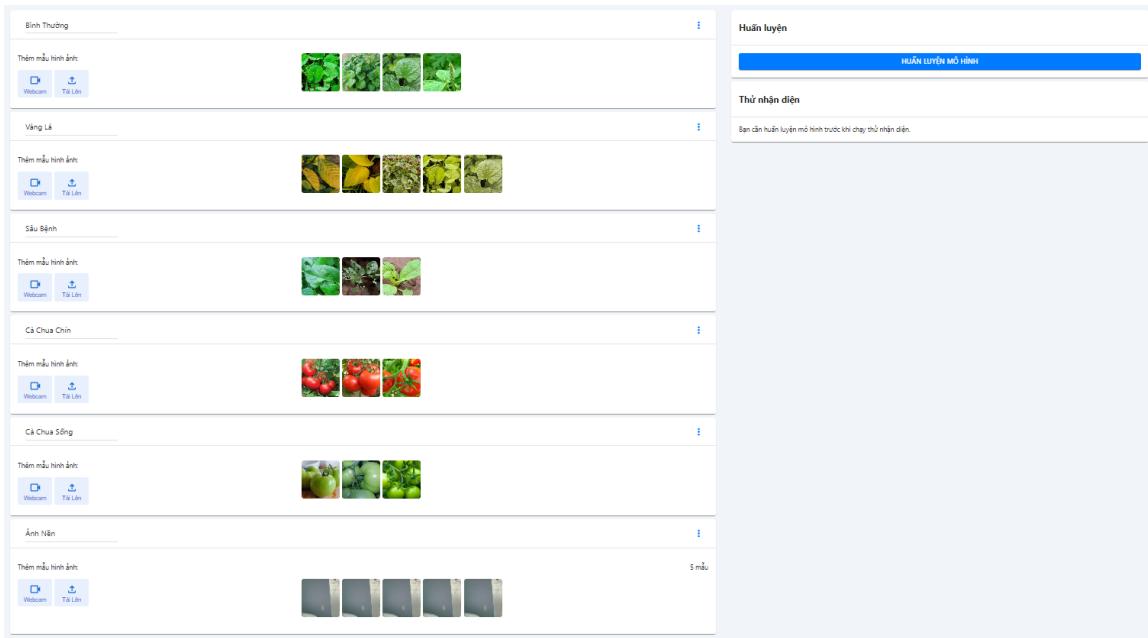
Điện thoại, hoặc máy tính với camera, được sử dụng để minh họa cho bộ não trí tuệ nhân tạo, với khả năng phân biệt được 2 loại trái cây. Tuy nhiên, rất quan trọng, nó cần phải phân biệt được khi không có trái cây nào trên băng chuyền.

Động cơ RC minh họa cho cơ cấu cơ khí để định hướng di chuyển cho trái cây trên băng chuyền. Nó sẽ được điều khiển bằng mạch Yolo:Bit khi nhận được tín hiệu từ phía bộ xử lý AI. Do vậy, nó cần phải biết khi không có trái cây nào trên băng chuyền. Nếu không huấn luyện kiến thức này, nguy cơ động cơ RC vẫn hoạt động khi không có trái cây là rất lớn. Điều này rõ ràng là không hợp lý trong dự án của chúng ta.

3 Huấn luyện hệ thống AI

Điều quan trọng trong hệ thống phân loại này, là kiến thức khi không có trái cây nào, hay nói một cách khác, là **ảnh nền**. Kiến thức này sẽ rất quan trọng đối với hệ thống, bên cạnh 2 kiến thức căn bản, bao gồm **cà chua chín** và **cà chua sống**.

Với mục đích là xây dựng một dự án tích hợp tất cả các tính năng, chúng tôi sẽ bổ sung thêm 3 kiến thức mới vào bộ não nhân tạo, bao gồm **Cà Chua Chín**, **Cà Chua Sống** và **Ảnh Nền**. Kết quả của việc thu thập dữ liệu và huấn luyện bộ não nhân tạo sẽ như kết quả ở hình bên dưới.



Hình 27.3: Huấn luyện hệ thống nhân tạo

Việc nhận dạng bất thường và nhận diện hoa quả sử dụng cùng một công nghệ, cụ thể là công nghệ học máy từ Google. Do đó, chúng ta không thể tách thành 2 bộ não nhân tạo và sử dụng nó song song. Chúng ta bắt buộc phải tạo mới một bộ não nhân tạo và tổng hợp tất cả các kiến thức trong nó trước khi huấn luyện. Để tăng độ hiệu quả của việc huấn luyện, bạn đọc có thể chụp hình trực tiếp từ camera, thay vì sử dụng hình ảnh có sẵn.

4 Hiện thực hệ thống

Sau khi đã huấn luyện bộ não AI, các bước để hiện thực dự án sẽ gồm 2 thành phần: xử lý AI để gửi lệnh và nhận lệnh AI để điều khiển thiết bị (động cơ RC).

4.1 Xử lý AI và gửi lệnh

Phần xử lý AI của dự án khá tương tự với các bài hướng dẫn trước, ngoại trừ việc nó có thể nhận dạng nhiều kiến thức hơn và sẽ gửi nhiều lệnh hơn tới mạch Yolo:Bit.

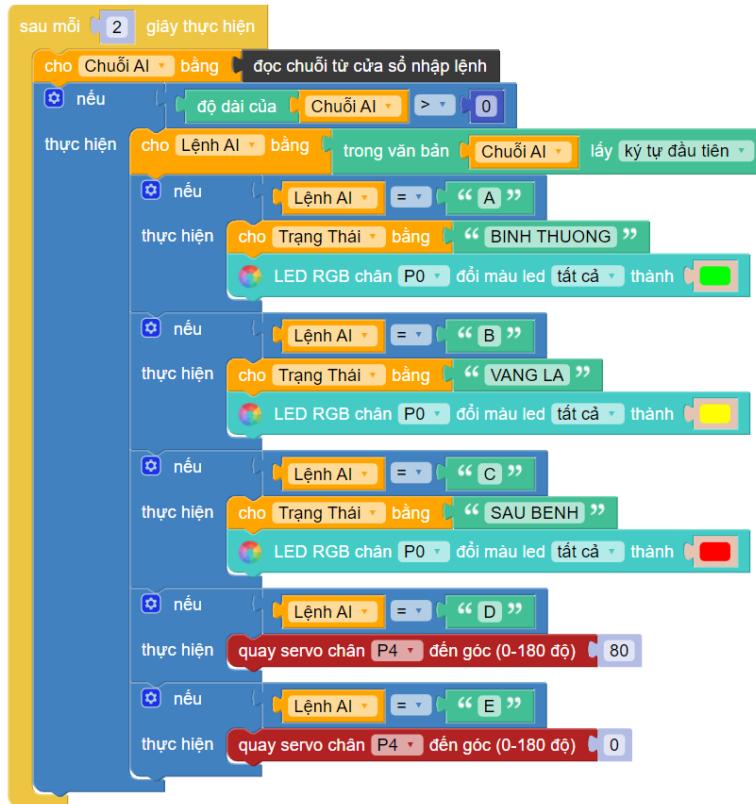


Hình 27.4: Xử lý nhận dạng AI

Trong dự án nhận dạng và phân loại trái cây sau thu hoạch, mấu chốt là hình ảnh nền thường sẽ rất ổn định và dễ nhận dạng. Do đó, chỉ cần huấn luyện với ảnh chụp trực tiếp từ camera là có thể nhận dạng tốt cho phần ảnh nền. Các kiến thức về cà chua sống và chín cũng không quá khó để nhận dạng chính xác (xác suất nhận dạng cao), khi nó có khoảng cách gần như cố định so với camera nhận dạng.

4.2 Nhận lệnh AI và điều khiển

Bước tiếp theo là chương trình điều khiển trên mạch Yolo:Bit. Tiếp tục mở rộng từ bài hướng dẫn trước, các câu lệnh nếu sẽ được bổ sung vào khối sự kiện xử lý A. Để điều khiển một động cơ RC servo nhỏ, minh họa cho đáp ứng cơ khí khi nhận diện được trạng thái của quả cà chua, chúng ta sẽ sử dụng câu lệnh **quay servo chân đến góc**, nằm trong nhóm **NÂNG CAO, CHÂN CẨM**, như minh họa ở hình sau đây:



Hình 27.5: Nhận lệnh và điều khiển RC phân loại

Trong chương trình trên, 2 lệnh mới cho phần AI đã được hiện thực, bao gồm lệnh "D" và "E", tương ứng với cà chua chín và sống. Với kiến thức ảnh nền (lệnh "F"), chúng ta có thể không cần phải xử lý bên phía Yolo:Bit. Chương trình được chia sẻ ở đường dẫn sau đây:

<https://app.ohstem.vn/#!/share/yolobit/2D4xqbpa3aewwazNnA2ImmaOxke>

Việc kết nối và vận hành hệ thống, bạn đọc có thể xem lại hướng dẫn ở các bài trước.

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, chẳng hạn như nhà thông minh hay nhà kho thông minh.

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, đó là lập trình hướng đối tượng.

Các hướng mở rộng cho dự án sẽ để dành cho bạn đọc tự sáng tạo, chẳng hạn như **đếm số lượng cà chua chín** đã được phân loại. Chúng tôi xin để dành tính năng này cho một hướng dẫn cao cấp hơn, khi **máy trạng thái** sẽ được sử dụng để giải quyết bài toán. Với khái niệm này, bạn đọc sẽ thấy việc thiết kế chương trình bằng hình vẽ còn quan trọng hơn rất nhiều so với việc hiện thực nó.

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:

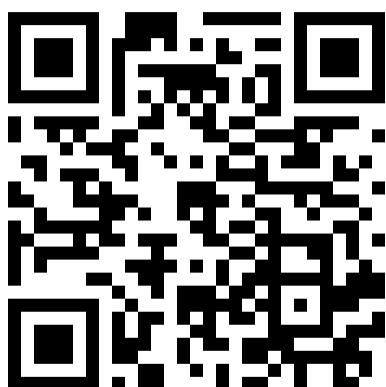
1. Cộng đồng trên Facebook

Link: <https://www.facebook.com/groups/dayvahocsteam>



2. Cộng đồng trên Zalo

Link: <https://zalo.me/g/vjgfmq313>



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