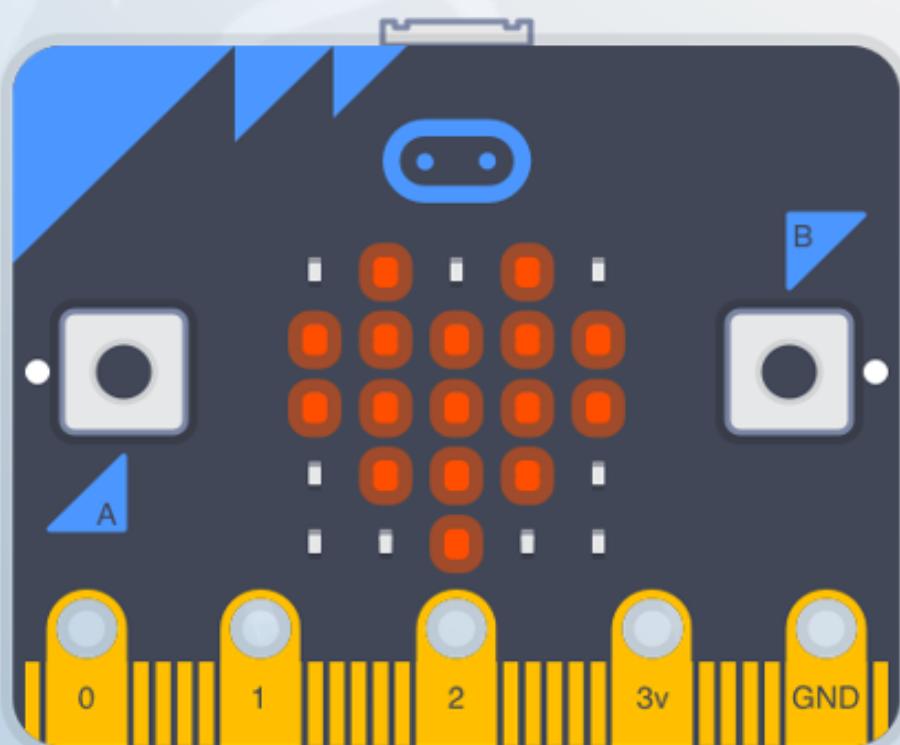




THE DARIU FOUNDATION

HƯỚNG DẪN LẬP TRÌNH MICRO:BIT



Lê Trọng Nhân - Nguyễn Trần Hữu Nguyên - Võ Tấn Phương
Nguyễn Thanh Hải - Phạm Văn Vinh

LỜI MỞ ĐẦU

Những ngày đầu năm 2018, là ngày bắt đầu cho những gì được đúc kết, mà chúng tôi gửi đến người học trong tài liệu "**Hướng dẫn lập trình MicroBit**" này. Thông qua việc hợp tác trong dự án "Lập trình tương lai cùng Google" do quỹ Dariu (The Dariu Foundation) phát động, chúng tôi đã có cơ hội đi nhiều tỉnh thành của Việt Nam, chia sẻ kinh nghiệm lập trình của chúng tôi với mọi người.

Trên tất cả, những điều thực sự có ý nghĩa với chúng tôi, là tính năng động, sáng tạo của thầy cô, những người mà chúng tôi làm việc trực tiếp thông qua những ngày tập huấn cuối tuần. Với nhiệt huyết của thầy cô, những công nghệ mới nhất về lập trình được truyền tải đến các em học sinh. Chúng tôi cũng vô cùng tự hào khi có thể thấy các em học sinh làm nên những sản phẩm, tự tin thuyết trình trước đám đông và làm chủ được các công nghệ cập nhật nhất, trong độ tuổi của các em.

Tài liệu này là món quà của nhóm chúng tôi gửi đến các thầy cô và bạn đọc, với mục đích là chia sẻ kinh nghiệm lập trình trên mạch Microbit, được đúc kết từ những buổi tập huấn thực tế do The Dariu Foundation tổ chức. Những chia sẻ và đóng góp của thầy cô là một phần nội dung trong tài liệu này. "**Hướng dẫn lập trình MicroBit**" được chia làm 2 phần như sau:

- Phần 1: Lập trình cơ bản trên MicroBit. Đây là phần hướng dẫn những câu lệnh cũng như các thao tác cơ bản để có thể viết và nạp chương trình lên mạch MicroBit. Trong phần này tập trung giới thiệu những tính năng nổi bật nhất của mạch MicroBit và cách sử dụng nó, như các cảm biến, tương tác bằng hành vi với người dùng và giao tiếp không dây.
- Phần 2: Các dự án với mạch MicroBit. Ở phần này, chúng tôi chọn 2 dự án có thể mô phỏng bằng mạch MicroBit là Đồng hồ thông minh và Cửa mật mã. Thông qua 2 dự án, chúng tôi mong muốn truyền tải chút kinh nghiệm về việc tổ chức chương trình khi hiện thực một dự án trên mạch MicroBit.

Chúng tôi hy vọng rằng người học sẽ có những phút giây trải nghiệm tuyệt vời với mạch MicroBit và môi trường lập trình MakeCode! Nội dung trong tài liệu này được tham khảo chính trên trang <https://microbit.org/code/>.

Lời cuối cùng, tôi xin cảm ơn The Dariu Foundation đã cho chúng tôi cơ hội để đồng hành trong nhiều dự án liên quan đến lập trình, để tài liệu này có cơ hội đến với người đọc. Tôi cũng xin chân thành cảm ơn tất cả bạn bè, đối tác đã đồng hành cùng tôi trong suốt một chặng đường dài trong dự án "Lập trình tương lai cùng Google".

Mục lục

I Lập trình cơ bản trên MicroBit	9
Chương 1. Làm quen với mạch lập trình MicroBit	11
1 Giới thiệu	12
2 Chương trình đầu tiên với MakeCode	13
3 Mô phỏng trên MicroBit	16
4 Các thao tác cơ bản trên MakeCode	16
4.1 Ghép nối các câu lệnh	17
4.2 Thay đổi thông số của một câu lệnh	18
4.3 Nhân bản – Xóa một câu lệnh	19
5 Bài tập	19
6 Câu hỏi ôn tập	20
Chương 2. Tương tác với màn hình hiển thị trên MicroBit	23
1 Giới thiệu	24
2 Tổ chức chương trình trên MakeCode	24
3 Câu lệnh show leds	25
4 Câu lệnh tạo hiệu ứng đợi pause	26
5 Lưu và mở lại chương trình	28
6 Bài tập	30
7 Câu hỏi ôn tập	31
Chương 3. Tổng hợp các câu lệnh về hiển thị	33
1 Giới thiệu	34
2 Giới thiệu tổng quan về nhóm lệnh hiển thị	34
3 Giới thiệu chức năng của các câu lệnh	34
3.1 Câu lệnh show number	35
3.2 Câu lệnh show string	35
3.2.1 Câu lệnh clear screen	36
3.3 Câu lệnh show arrow	36
4 Bài tập	37
5 Câu hỏi ôn tập	38
Chương 4. Điều khiển nút nhấn trên MicroBit	41
1 Giới thiệu	42
2 Tổng quan về nút nhấn	42
3 Lập trình điều khiển nút nhấn	43

4	Bài tập	44
5	Câu hỏi ôn tập	46
Chương 5. Lập trình MicroBit bằng điện thoại thông minh		47
1	Giới thiệu	48
2	Lập trình bằng điện thoại Android	48
2.1	Ghép đôi điện thoại và mạch MicroBit	51
2.2	Lập trình trên điện thoại	53
2.3	Nạp chương trình vào mạch MicroBit	54
3	Lập trình bằng điện thoại iOS	55
3.1	Ghép đôi thiết bị	57
3.1.1	Lập trình và nạp chương trình lên MicroBit	58
4	Câu hỏi ôn tập	61
Chương 6. Tương tác giữ MicroBit và hành vi người dùng		63
1	Giới thiệu	64
2	Nguyên lý phát hiện hành vi của người dùng	64
3	Các câu lệnh phát hiện hành vi	65
3.1	Hành vi on shake	65
3.2	Các hành vi khác	66
4	Bài tập	67
5	Câu hỏi ôn tập	72
Chương 7. Cảm biến trên MicroBit		75
1	Giới thiệu	76
2	Cảm biến là gì?	76
2.1	Cảm biến ánh sáng	76
2.2	Cảm biến nhiệt độ	77
2.3	Cảm biến la bàn	77
2.4	Cảm biến góc xoay	78
3	Bài tập	79
4	Câu hỏi ôn tập	79
Chương 8. Gửi dữ liệu không dây giữa các mạch MicroBit		81
1	Giới thiệu	82
2	Nhóm lệnh Radio	82
3	Chương trình nốt gửi	83
4	Chương trình cho nốt nhận	83
5	Bài tập trên lớp	84
6	Câu hỏi ôn tập	85
Chương 9. Câu lệnh điều kiện IF		87
1	Giới thiệu	88
2	Cấu trúc IF	88
3	Cấu trúc IF ELSE	90
4	Câu lệnh IF ELSE IF ELSE	91
5	Câu lệnh IF lồng nhau	91
6	Bài tập	92
7	Câu hỏi ôn tập	95

Chương 10. Câu lệnh lặp trên MakeCode	97
1 Giới thiệu	98
2 Câu lệnh repeat	98
3 Câu lệnh while	100
4 Câu lệnh for	100
5 Danh sách và lệnh for element	101
6 Câu hỏi ôn tập	104
II Dự án với MicroBit	105
Chương 11. Đồng hồ thông minh trên MicroBit	107
1 Giới thiệu	108
2 Nguyên lý thực thi chương trình trên MicroBit	109
3 Hiện thực chức năng của đồng hồ	109
3.1 Đếm giờ của đồng hồ	109
3.2 Hiển thị giờ ra màn hình	112
3.3 Hiển thị nhiệt độ	114
3.4 Hiển thị cường độ ánh sáng	114
3.5 Các hướng phát triển	115
4 Câu hỏi ôn tập	117
Chương 12. Cửa sổ mã trên MicroBit	119
1 Giới thiệu	120
2 Khai báo biến và khởi tạo	120
3 Ghép chuỗi khi nhấn nút A hoặc nút B	121
4 So sánh mật khẩu khi nhấn A và B	122
5 Thêm thư viện điều khiển động cơ	123
6 Các hướng mở rộng	125
7 Câu hỏi ôn tập	127

Phần I

Lập trình cơ bản trên MicroBit



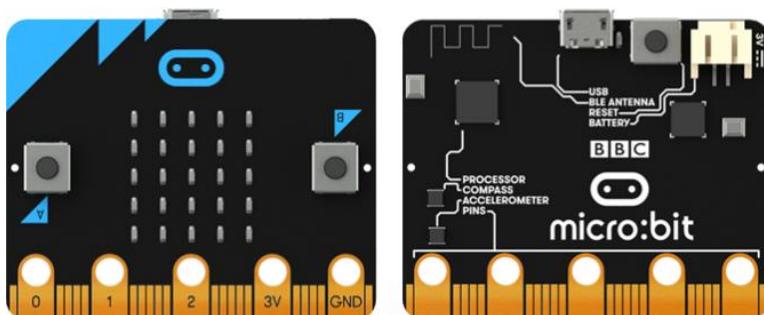
CHƯƠNG 1

Làm quen với mạch lập trình MicroBit

1 Giới thiệu

Trước tiên, Microbit là một bo mạch phần cứng được thiết kế để phục vụ cho việc giảng dạy, đặc biệt là các em học sinh ở cấp độ trung học cơ sở và trung học phổ thông. Chính vì vậy, mạch Microbit đảm bảo các tiêu chuẩn an toàn về điện, cho phép các em học sinh cầm nắm mà không gây ảnh hưởng đến sức khỏe. Đây là tiêu chí vô cùng quan trọng để các chương trình giáo dục theo tiêu chuẩn STEM (viết tắt của các từ Science (khoa học), Technology (công nghệ), Engineering (kỹ thuật) và Math (toán học)) có thể tự tin ứng dụng mạch Microbit vào giảng dạy. Bên cạnh đó, các linh kiện trên mạch Microbit cũng được chọn lựa, khung viền mạch ít góc cạnh, hạn chế tối thiểu việc va chạm bo mạch với các em học sinh. Hình ảnh của mạch Microbit được trình bày ở Hình 1.1.

Môi trường lập trình trên mạch Microbit cũng vô cùng hiện đại và phong phú. Chúng ta không những có thể lập trình ngoại tuyến (offline) bằng phần mềm trên máy tính mà còn có thể lập trình trực tuyến (online) hoặc lập trình trên các nền tảng di động, chẳng hạn như điện thoại thông minh và máy tính bảng. Ngôn ngữ lập trình cho Microbit được hỗ trợ từ cơ bản bằng ngôn ngữ kéo thả MakeCode, cho đến các ngôn ngữ cao cấp hơn như Javascript, C, C++ hay Python.



Hình 1.1: Hình ảnh hai mặt trên và dưới của MicroBit

Microbit có thể xem là một máy tính thu nhỏ và có khả năng lập trình được. Tức là người dùng có thể thay đổi chức năng và hoạt động của nó để tạo ra các ứng dụng hấp dẫn và thu hút, từ điều khiển các hiệu ứng đèn chớp tắt, cho đến các ứng dụng phức tạp như là điều khiển Robot không dây, nhà thông minh, thậm chí là các ứng dụng cao cấp trong thế giới kết nối vạn vật (còn gọi là Internet of Thing - IoTs).

Trong bài đầu tiên này, chúng ta sẽ tập trung vào các thao tác cơ bản trên mạch Microbit và môi trường lập trình MakeCode phiên bản trực tuyến (online). Những thông tin chi tiết về mạch Microbit sẽ được trình bày chi tiết hơn ở các bài tiếp theo. Các mục tiêu cụ thể trong bài hướng dẫn này được tóm lược như bên dưới:

- Học sinh hiểu được các bước cơ bản trong việc lập trình MicroBit
- Học sinh nắm được các thao tác cơ bản trên MakeCode
- Học sinh viết được chương trình đơn giản trên MicroBit

- Học sinh nắm được việc sử dụng chương trình mô phỏng của MicroBit

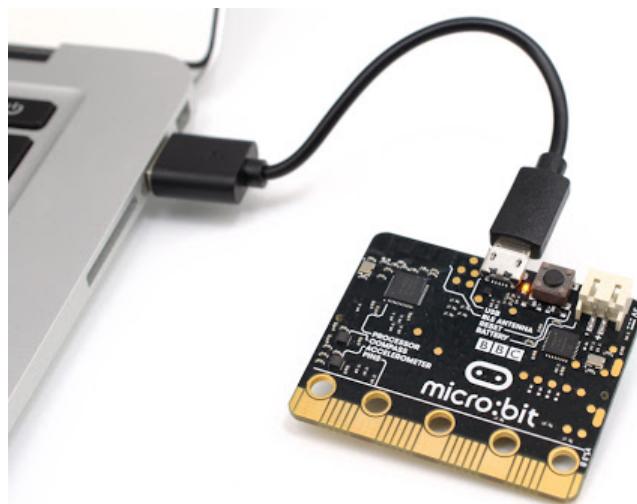
Ở phần tiếp theo của bài này, chúng ta sẽ bắt đầu chương trình đầu tiên trên Microbit với môi trường lập trình trực tuyến MakeCode .

2 Chương trình đầu tiên với MakeCode

Trong hướng dẫn ở phần này, chúng ta sẽ hiện thực chương trình đầu tiên chạy trên mạch Microbit với trình soạn thảo trực tuyến MakeCode. Mục tiêu của chúng ta là hiển thị hình trái tim chớp tắt trên mạch Microbit. Các bước hướng dẫn chi tiết được trình bày bên dưới.

Bước 1: Kết nối với MicroBit

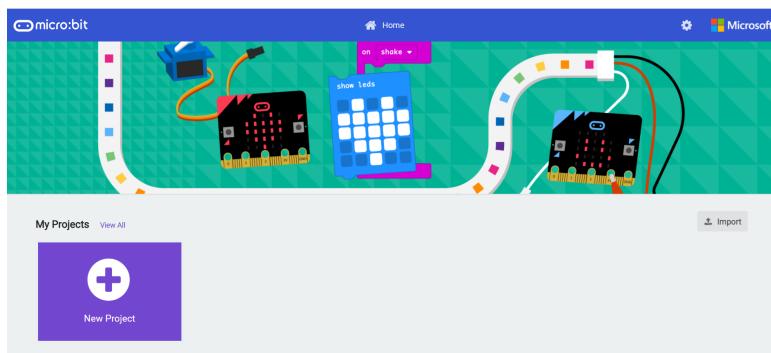
Việc kết nối với MicroBit thực sự rất đơn giản, chỉ cần một dây micro USB: đầu micro gắn vào mạch Microbit còn đầu còn lại được gắn vào cổng USB của máy tính. Hệ điều hành trên máy tính sẽ tự động nhận ra mạch MicroBit như một USB bình thường. Việc kết nối giữa máy tính và mạch MicroBit được minh họa như Hình 1.2.



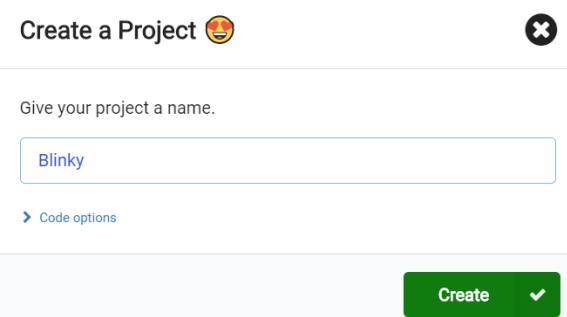
Hình 1.2: Kết nối mạch MicroBit với máy tính thông qua dây USB

Bước 2: Viết chương trình

Để lập trình cho MicroBit, có rất nhiều công cụ hỗ trợ. Tuy nhiên trong giáo trình này, chúng tôi sử dụng môi trường lập trình trực tuyến, gọi là MakeCode. Một lợi thế rất lớn mà MakeCode có được là việc mô phỏng chương trình trước khi nạp trực tiếp vào mạch MicroBit. Chức năng này sẽ tiết kiệm nhiều thời gian cho việc kiểm tra chương trình. Chúng ta sẽ vào đường dẫn <https://makecode.microbit.org/>^[Hal], và chọn vào New Project, như minh họa ở Hình 1.3.



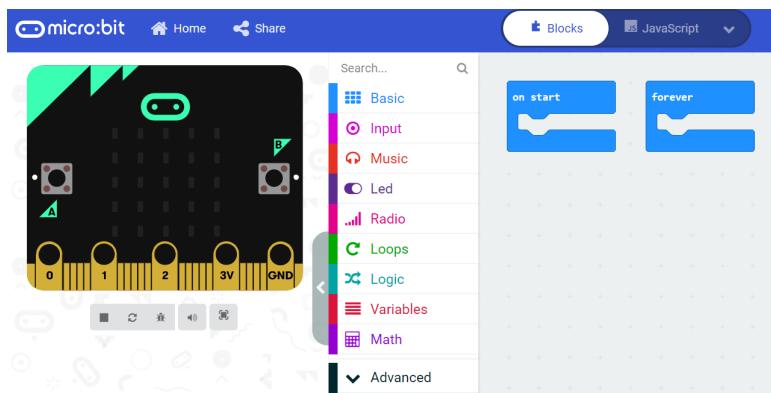
Hình 1.3: Vào trang web lập trình trực tuyến MakeCode



Hình 1.4: Đặt tên cho chương trình đầu tiên là Blinky

Tới đây, một giao diện sẽ xuất hiện và yêu cầu chúng ta nhập tên cho chương trình. Bạn có thể chọn một tên gợi nhớ, chẳng hạn như là Blinky (chớp tắt đèn), như kết quả ở Hình 1.4.

Cuối cùng, chúng ta nhấn vào nút Create và giao diện như Hình 1.5 sẽ hiện ra để chúng ta có thể bắt đầu lập trình.



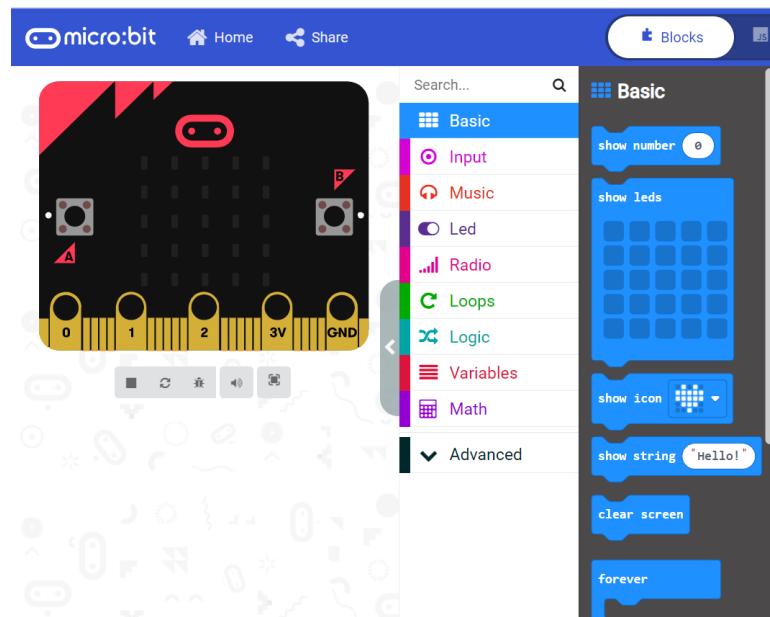
Hình 1.5: Giao diện lập trình cơ bản

Chương trình của MicroBit mặc định sẽ có 2 khối cơ bản như sau:

- **on start:** Những câu lệnh trong khối này sẽ được thực hiện đầu tiên khi mới bật nguồn, tuy nhiên nó chỉ được thực hiện duy nhất một lần.

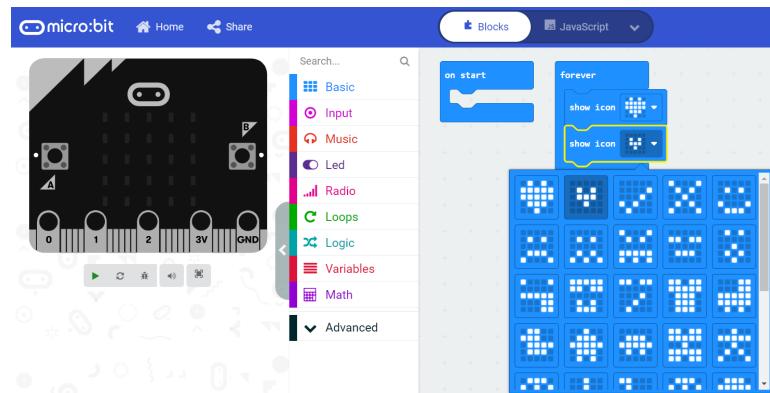
- **forever:** Những câu lệnh trong khối lệnh này sẽ được thực hiện sau khởi on start, tuy nhiên nó được lặp đi lặp lại mãi mãi khi mạch MicroBit còn được cấp nguồn điện.

Đầu tiên, để cho đơn giản, chúng ta dùng câu lệnh **show icon** nằm trong nhóm lệnh **Basic** (xem Hình 1.6)



Hình 1.6: Viết chương trình đầu tiên bằng lệnh show icon

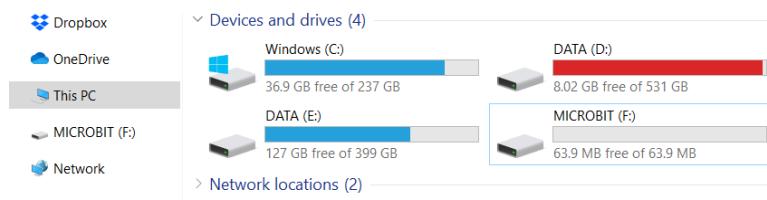
Chúng ta có thể ghép 2 câu lệnh show icon như Hình 1.7. Câu lệnh show icon là một câu lệnh có thể thay đổi nội dung hiển thị bằng cách chọn vào dấu mũi tên bên phải.



Hình 1.7: Tạo chương trình bằng 2 câu lệnh show icon

Bước 3: Nạp chương trình

Việc nạp chương trình vào mạch MicroBit thực sự rất tiện lợi do bo mạch này được nhận diện như một thiết bị USB bình thường. Như hình bên dưới, mạch MicroBit được nhận dạng là ổ đĩa , như minh họa ở Hình 1.8.



Hình 1.8: Mạch MicroBit được nhận dạng là một ổ đĩa trên máy tính

Do đó, chúng ta chỉ cần nhấn vào nút Download và chọn đường dẫn tới mạch MicroBit là xong. Trong trường hợp trình duyệt web của bạn tự động tải, chương trình sẽ được lưu ở thư mục mặc định Download hoặc lưu tại thư mục mà bạn cài đặt. Bạn chỉ cần mở thư mục đó, rồi sao chép file mới tải về (có đuôi mở rộng là .hex) vào ổ đĩa F để nạp chương trình cho mạch Microbit. Quá trình nạp sẽ mất vài giây. Trong suốt quá trình nạp, đèn nguồn bên cạnh khe cắm USB sẽ chớp tắt liên tục. Khi đèn ngừng chớp tắt cũng là lúc chương trình nạp xong và bo mạch MicroBit sẽ bắt đầu chạy chương trình mà chúng ta vừa hiện thực.

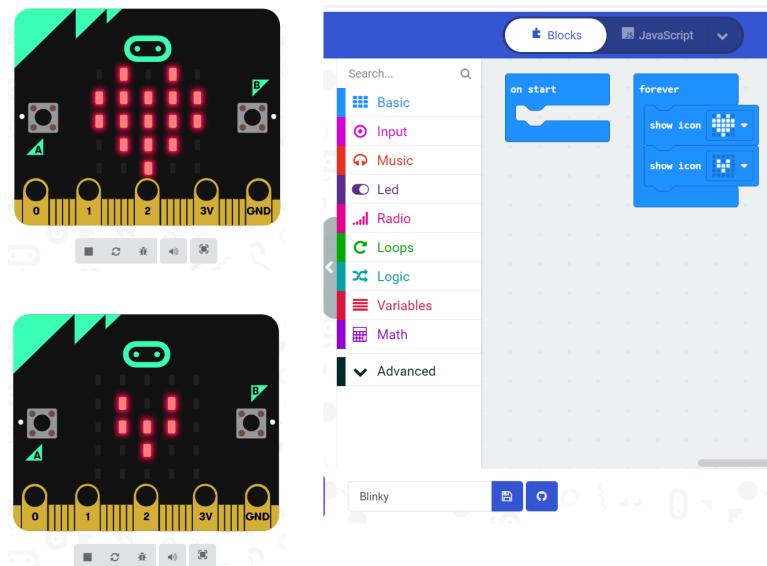
3 Mô phỏng trên MicroBit

Đây là một đặc tính hết sức thú vị của chương trình lập trình hỗ trợ trên bo mạch MicroBit. Ở khung cửa sổ bên trái, mạch mô phỏng lại chương trình mà chúng ta vừa viết. Nhờ chức năng này mà chúng ta có thể tự tìm hiểu chức năng của một câu lệnh, giống như việc nhấp vào câu lệnh và xem đáp ứng của chú mèo bên ngôn ngữ Scratch. Trong ví dụ bên dưới, mạch mô phỏng sẽ luôn hiển thị hình ảnh 2 trái tim, như minh họa ở Hình 1.9, tương ứng với chương trình của chúng ta hiện thực ở khung chương trình bên phải.

Nhờ chức năng mô phỏng này, chúng ta có thể viết và kiểm tra sơ bộ chương trình trước khi nạp vào mạch MicroBit. Bên cạnh đó, học sinh có thể tự viết những chương trình ở nhà, không cần mạch MicroBit, mà vẫn có thể thấy được kết quả của mình.

4 Các thao tác cơ bản trên MakeCode

Tương tự như ngôn ngữ kéo thả Scratch, môi trường lập trình trên MakeCode cũng được tổ chức thành 3 phần, như trình bày ở Hình 1.5. **Thay thế cho sân khấu với**



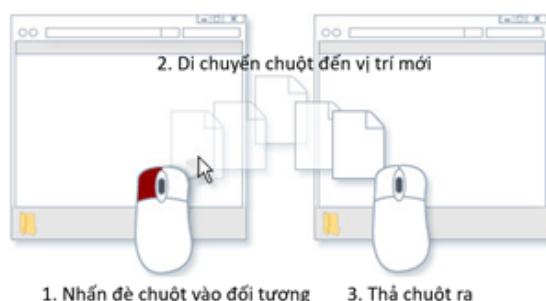
Hình 1.9: Chương trình mô phỏng trên MicroBit

nhân vật chú mèo, chúng ta có hình ảnh của mạch MicroBit ở khung cửa sổ thứ nhất. Thực chất, đây chỉ là hình ảnh mô phỏng việc thực thi chương trình của chúng ta trên mạch MicroBit. Về cơ bản, việc mô phỏng khá chính xác với chương trình sẽ chạy thực tế trên mạch MicroBit.

Ở khung cửa sổ thứ 2, chúng ta có **các câu lệnh trên MicroBit**, được tổ chức theo từng nhóm, mỗi nhóm có 1 màu khác nhau. Cũng như môi trường lập trình trên Scratch, việc tổ chức này giúp chúng ta dễ dàng tìm kiếm các câu lệnh dựa vào màu sắc của chúng. Cuối cùng, là **khung chương trình**, dùng để ghép nối các khối lệnh trên MicroBit.

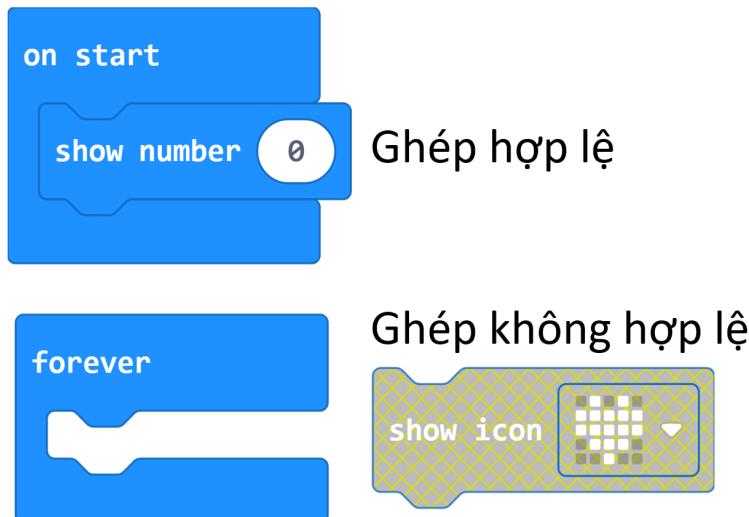
4.1 Ghép nối các câu lệnh

Hoàn toàn tương tự với Scratch, việc ghép nối các câu lệnh trên MicroBit được dựa vào nguyên lý “kéo-thả”, như minh họa ở Hình 1.10: Giữ đòn tigarette cần kéo thả bằng chuột trái (1), di chuyển đến vị trí mới (2) và thả chuột ra (3).



Hình 1.10: Ba bước để thực hiện việc kéo thả

Tuy nhiên, trên môi trường MicroBit, khi một câu lệnh được ghép nối không hợp lệ, nó sẽ bị tô xám. Chỉ khi nào được ghép nối một cách hợp lệ, nó mới hiển thị màu của câu lệnh đó, như minh họa ở Hình 1.11.

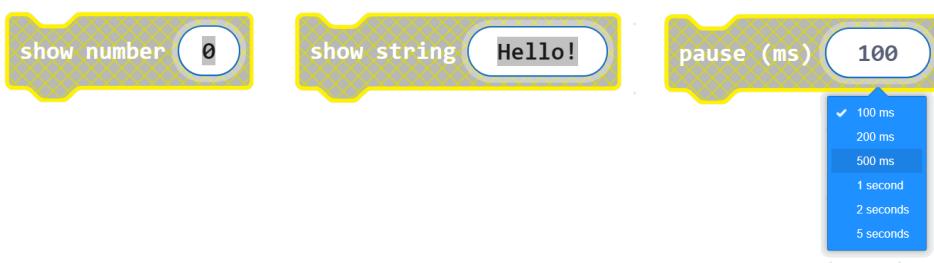


Hình 1.11: Ghép nối hợp lệ và không hợp lệ

4.2 Thay đổi thông số của một câu lệnh

Hoàn toàn tương tự như trên Scratch, có 2 cách để chúng ta có thể thay đổi thông số của một câu lệnh:

- Nhập đôi vào giá trị trong ô màu trắng, sau đó nhập giá trị mới.
- Chọn vào một danh sách có sẵn, như minh họa ở Hình 1.12.

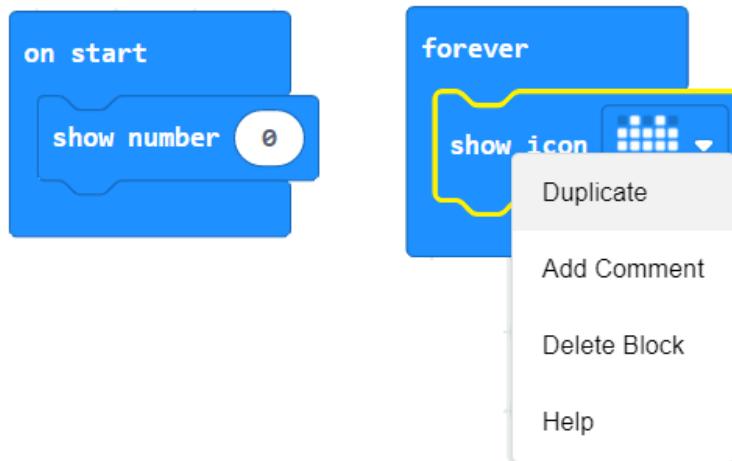


Hình 1.12: Thay đổi thông số của một câu lệnh

Trong một số trường hợp, mặc dù là danh sách chọn, chúng ta vẫn có thể nhập một giá trị ngoài danh sách có sẵn. Ví dụ như ở Hình 1.12, chúng ta có thể nhập vào con số 6000 nếu muốn tạo hiệu ứng đợi 6 giây.

4.3 Nhân bản – Xóa một câu lệnh

Để thực hiện 2 chức năng này, chúng ta nhấp chuột phải vào câu lệnh đó (câu lệnh sẽ được tô màu vàng), và chọn tiếp **Duplicate** hoặc **Delete Block**, như minh họa ở Hình 1.13. Trong trường hợp muốn dùng phím tắt, để nhân bản một câu lệnh chúng ta có thể dùng tổ hợp phím **Ctrl+C** và **Ctrl+V**, và nhấn phím **Delete** để xóa một câu lệnh.



Hình 1.13: Nhân bản - Xóa một câu lệnh

Lưu ý: MakeCode không hỗ trợ nhân bản nhiều câu lệnh đồng thời. Một mẹo nhỏ có thể được sử dụng là ghép nhiều câu lệnh cần nhân bản trong khối on start hoặc forever rồi nhấn bản cả khối này. Sau đó chúng ta có thể kéo thả một nhóm các câu lệnh cần dùng và xóa câu lệnh thừa (on start hoặc forever) đi.

Một cách khác để xóa một câu lệnh là bạn có thể kéo nó từ khung chương trình và thả vào khung cửa sổ hiển thị các nhóm lệnh của môi trường lập trình MakeCode.

5 Bài tập

Người học có thể tự thiết kế thêm những hình ảnh sinh động khác trong kịch bản hiển thị của MicroBit bằng câu lệnh show icon.

6 Câu hỏi ôn tập

1. Để lập trình mạch micro:bit chúng ta cần gì?
 - A. Cục tẩy, cuộn vở, cây bút.
 - B. Máy tính, mạng, mạch micro:bit, dây micro USB.
 - C. Máy cưa, máy hàn, máy phát điện. Tất cả đều sai.
2. Để kết nối máy tính với mạch micro:bit chúng ta cần thiết bị nào?
 - A. Dây micro USB
 - B. Dây mạng LAN
 - C. Máy in
 - D. Tất cả đều đúng
3. Khi gắn mạch micro:bit vào máy tính thì?
 - A. Không có hiện tượng gì?
 - B. Máy tính sẽ nhận ổ đĩa USB.
 - C. Có hiện tượng chập cháy.
 - D. Tất cả đều sai.
4. Để lập trình cho micro:bit chúng ta cần truy cập vào trang web nào dưới đây?
 - A. makecode.microbit.org
 - B. facebook.com
 - C. google.com
 - D. tinkercad.com
5. Trang **makecode.microbit.org** là website dùng để làm gì?
 - A. Là ứng dụng lập trình bảng mạch Microbit
 - B. Là trang học tập và ứng dụng Microbit
 - C. Là cộng đồng chia sẻ các ứng dụng hay từ Microbit
 - D. Tất cả đáp án trên đều đúng
6. Ngôn ngữ nào sau đây dùng để lập trình cho micro:bit ?
 - A. HTML
 - B. Java
 - C. Kéo thả
 - D. Tất cả đều đúng.
7. Để nạp chương trình cho mạch micro:bit chúng ta nhấn nút:
 - A. Nút A trên mạch micro:bit.
 - B. Nút B trên mạch micro:bit.
 - C. Nút Download trên giao diện lập trình.
 - D. Nút nguồn của máy tính.
8. Trong giao diện lập trình micro:bit, hình mạch micro:bit bên trái có tác dụng gì?
 - A. Để mô phỏng hoạt động thật của mạch micro:bit tương ứng với code hiện tại.
 - B. Để tượng trưng và làm đẹp giao diện.
 - C. Để thể hiện trạng thái hiện tại của mạch micro:bit đang kết nối.
 - D. Tất cả đều sai.

9. Khối nào sau đây được sinh ra khi ta tạo một dự án mới với micro:bit ?
- A. on start
 - B. forever
 - C. A và B sai
 - D. A và B đúng
10. Các câu lệnh trong khối **on start** sẽ:
- A. Chạy đầu tiên và duy nhất một lần khi mạch được khởi động.
 - B. Chạy sau khi thực hiện forever
 - C. Chạy lặp lại mãi mãi
 - D. Tất cả đều sai.
11. Các câu lệnh trong khối **forever** sẽ:
- A. Chạy đầu tiên và duy nhất một lần khi mạch được khởi động.
 - B. Chạy trước khi thực hiện on start
 - C. Chạy lặp lại mãi mãi
 - D. Tất cả đều sai.
12. Mạch MicroBit lấy nguồn điện từ đâu?
- A. Lấy nguồn điện từ pin trong mạch
 - B. Lấy nguồn điện thông qua dây kết nối với thiết bị lập trình .
 - C. Lấy nguồn điện thông qua điện từ trường
 - D. Không cần có nguồn điện cung cấp

Đáp án

1. B 2. A 3. B 4. A 5. D 6. C 7. C 8. A 9. D 10. A 11. C 12. B

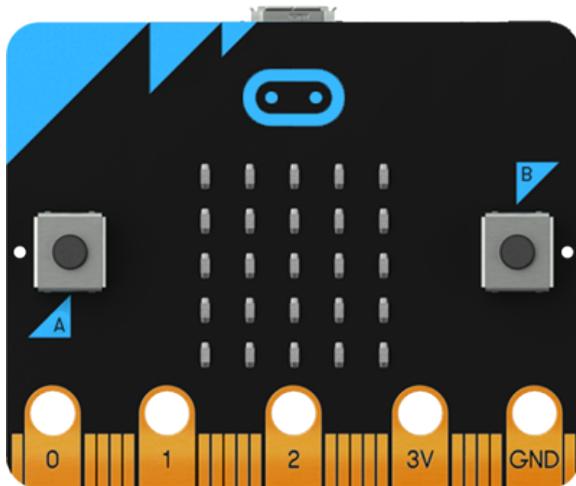
CHƯƠNG 2

Tương tác với màn hình hiển thị trên MicroBit



1 Giới thiệu

Đối với việc lập trình trên các bo mạch, màn hình hiển thị là một nhu cầu rất cần thiết, đặc biệt là cho người mới bắt đầu, để có thể xem kết quả của chương trình mà mình hiện thực. Đây là điều mà mạch MicroBit làm rất tốt với 25 đèn hiển thị (được phân bố trên 5 hàng và 5 cột như trình bày ở Hình 2.1). Người lập trình có thể dễ dàng hiển thị thông tin cũng như hình ảnh mà mình thích trên 25 đèn này.



Hình 2.1: Màn hình hiển thị gồm 25 đèn của MicroBit

Trong bài hướng dẫn này, chúng tôi sẽ tập trung vào việc tìm hiểu các câu lệnh phục vụ cho việc hiển thị, chủ yếu nằm trong nhóm lệnh đầu tiên, nhóm Basic. Bên cạnh đó, chúng ta sẽ tìm hiểu sâu hơn về cấu trúc chương trình trên MakeCode với các khối **on start** và **forever**. Các mục tiêu của bài hướng dẫn này được cụ thể như bên dưới:

- Học sinh nắm được việc tổ chức chương trình trên MakeCode
- Học sinh hiểu được các bước cơ bản trên màn hình hiển thị
- Học sinh viết được chương trình tạo hiệu ứng hiển thị trên MicroBit
- Học sinh phối hợp được các câu lệnh để tạo ra hiệu ứng hiển thị đẹp
- Học sinh có thể lưu và mở lại chương trình của mình

2 Tổ chức chương trình trên MakeCode

Trước khi đi vào chi tiết của phần này, chúng tôi muốn làm rõ hai khái niệm MakeCode và Microbit. MakeCode là một trình soạn thảo. Chúng ta dùng nó để soạn

thảo chương trình, giống như việc viết một văn bản trên Microsoft Word. MakeCode là công cụ để chúng ta xây dựng nên một chương trình. Ngôn ngữ trên MakeCode là một phiên bản của ngôn ngữ Blockly (ở Việt Nam chúng ta quen gọi là ngôn ngữ "kéo-thả"), một sản phẩm của Google, với đối tượng người học là các em học sinh từ 10 tuổi. Ngoài ra, MakeCode cũng đóng vai trò là một "thông dịch viên", để dịch chương trình từ ngôn ngữ Blockly sang ngôn ngữ mà mạch MicroBit có thể hiểu được. Nếu để ý, bạn sẽ dễ dàng nhận ra, chương trình mà chúng ta viết trên MakeCode được chuyển thành 1 dạng file khác, có đuôi là **.hex**. Đây là định dạng file sẽ thực thi trên mạch MicroBit.

MicroBit là nền tảng để thực thi chương trình mà chúng ta viết trên MakeCode. Do đó, chúng ta cần phải nạp chương trình từ máy tính vào mạch MicroBit. Có thể với những ai đã có kinh nghiệm làm việc với bo mạch, đây là điều hiển nhiên, nhưng đối với người mới lập trình, bạn có thể quên mất đi bước này và tự hỏi tại sao chương trình lại không chạy trên mạch MicroBit.

Chương trình thực thi trên MicroBit được bắt đầu bằng các câu lệnh nằm trong khối **on start**. Sau khi thực hiện xong, nó sẽ tự động chuyển qua khối **forever** và lặp đi lặp lại mãi mãi. Đây là điều khác biệt rất lớn, nếu chúng ta so sánh với chương trình thực thi trên máy tính. Thông thường, chương trình trên máy tính chỉ chạy một lần và kết thúc khi thực hiện xong. Ngược lại, chương trình trên mạch MicroBit được thực thi mãi mãi, chừng nào nó vẫn còn được cấp nguồn điện. Do đó, việc tiếp cận lập trình trên mạch MicroBit sẽ hoàn toàn khác với lập trình trên máy tính: MicroBit tập trung vào ứng dụng, xây dựng các dự án thực tế nhiều hơn so với máy tính.

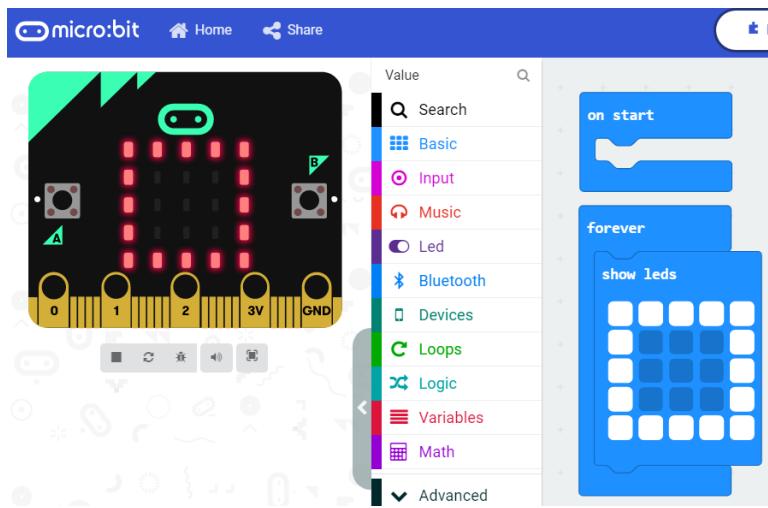
Cũng vì đặc tính này, mà mạch MicroBit được xếp vào nhóm mạch **Vi điều khiển (Micro-Controller)**. Mạch Arduino cũng được xếp vào nhóm này, với các đặc điểm trái ngược hoàn toàn với máy tính như giá thành rẻ và có độ bền cao, phù hợp cho việc hiện thực các ứng dụng thực tế hơn là máy tính (thuộc nhóm **Vi xử lý - Micro-Processor**), vốn có giá thành cao và độ bền kém, nhất là khi nguồn điện không ổn định (ví dụ việc bật tắt nguồn nóng có thể dẫn đến hư máy tính).

Trong phần hướng dẫn tiếp theo, chúng tôi sẽ tập trung vào 2 câu lệnh trong mục Basics, là **show leds** và câu lệnh tạo hiệu ứng đợi **pause**. Với 2 câu lệnh này, người học đã có thể chủ động tạo ra nhiều hình ảnh và hiệu ứng đẹp trên mạch Microbit bằng việc sắp xếp xen kẽ 2 câu lệnh này, như hiệu ứng chạy chữ, hiệu ứng pháo hoa chẳng hạn.

3 Câu lệnh show leds

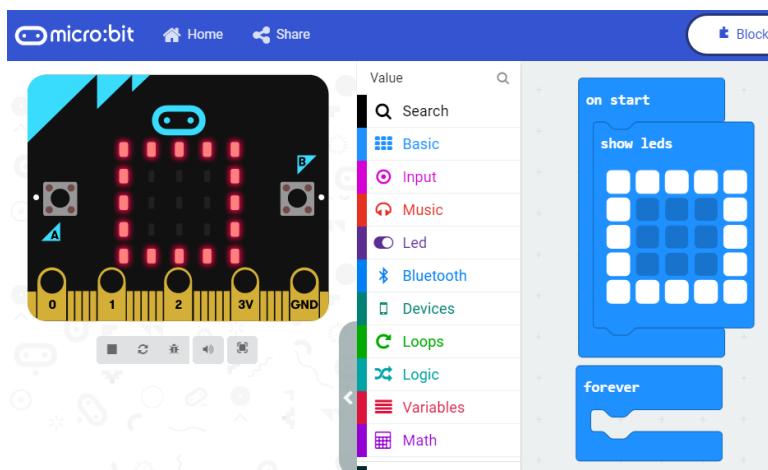
Câu lệnh này đưa ra một giao diện tương tác rất sinh động với màn hình hiển thị 25 led của MicroBit. Chúng ta muốn bóng đèn nào sáng, chỉ cần nhấp chuột vào nó. Khi muốn bóng đèn đó tắt, chúng ta nhấp thêm một lần nữa. Trong ví dụ ở Hình 2.2, một hình vuông sẽ được hiển thị trên mạch MicroBit.

Bạn có thể di chuyển câu lệnh **show leds** từ khối **forever** sang khối **on start**, như minh họa ở Hình 2.3. Mặc dù kết quả hiển thị trên mạch Microbit là không thay



Hình 2.2: Sáng một hình vuông trên màn hình hiển thị

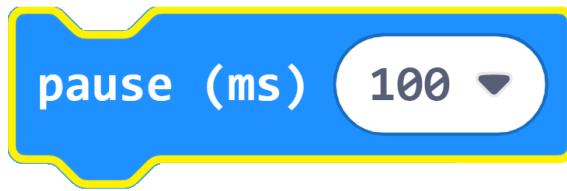
đổi, nhưng ý nghĩa thực thi của chương trình khác nhau hoàn toàn. Cụ thể, chương trình ở Hình 2.3 chỉ gửi lệnh hiển thị ra màn hình 1 lần duy nhất mà thôi. Đèn sẽ sáng mãi mãi cho đến khi nào nó nhận một lệnh tắt đi. Ngược lại, ở Hình 2.2, đèn sẽ liên tục nhận lệnh phải sáng.



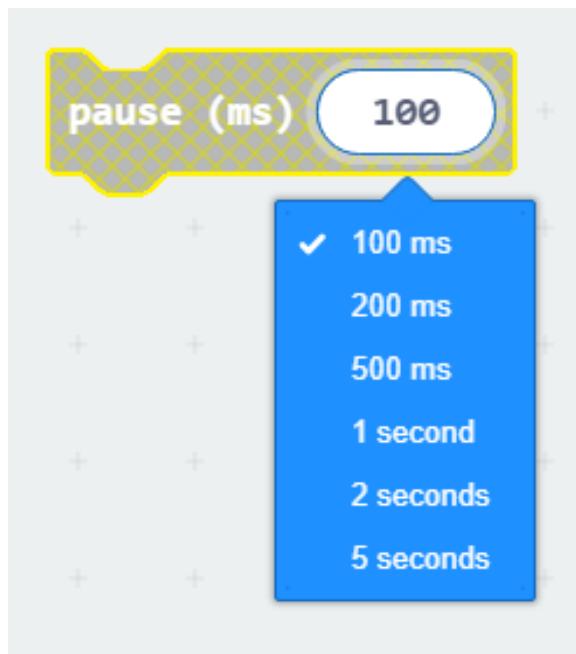
Hình 2.3: Một chương trình khác để hiển thị hình vuông

4 Câu lệnh tạo hiệu ứng đợi pause

Trong bài này, để có thể hiển thị nhiều nội dung hấp dẫn, chúng ta cần sử dụng thêm câu lệnh đợi. Câu lệnh này có tên là **pause** và cũng nằm cùng nhóm Basic với câu lệnh show leds, như trình bày ở Hình 2.4.



Hình 2.4: Câu lệnh tạo hiệu ứng đợi



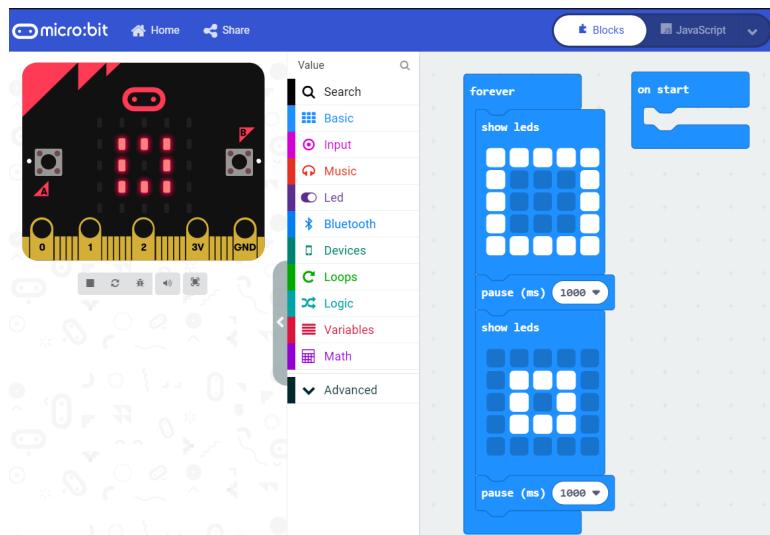
Hình 2.5: Các thông số được phép lựa chọn của câu lệnh đợi

Đơn vị thời gian trong câu lệnh này là mili giây. Chúng ta có thể thấy đây là câu lệnh có thể lựa chọn thông số bên trong bằng cách nhấn vào phím mũi tên bên phải như hình dưới đây:

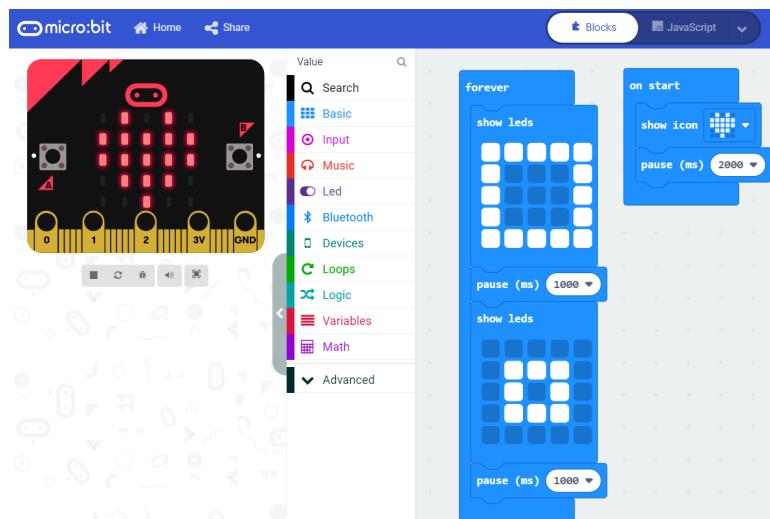
Như hình bên trên, thời gian đợi tối đa chỉ là 5 giây mà thôi. Nếu chúng ta muốn tạo hiệu ứng đợi lâu hơn chẳng hạn như 7 giây, có 2 cách sau đây:

- Ghép nhiều câu lệnh pause với nhau, ví dụ như trên sẽ là **pause(5000)** và **pause(2000)** để có hiệu ứng đợi 7 giây.
- Gõ tay vào ô màu trắng thời gian mong muốn. Ví dụ muốn đợi 7 giây, chúng ta sẽ gõ vào số **7000**, do đơn vị ở đây là mili giây.

Một chương trình gợi ý về việc tạo ra một hiệu ứng hiển thị trên MicroBit như sau:



Hình 2.6: Một chương trình tạo hiệu ứng đơn giản



Hình 2.7: Đáp án gợi ý

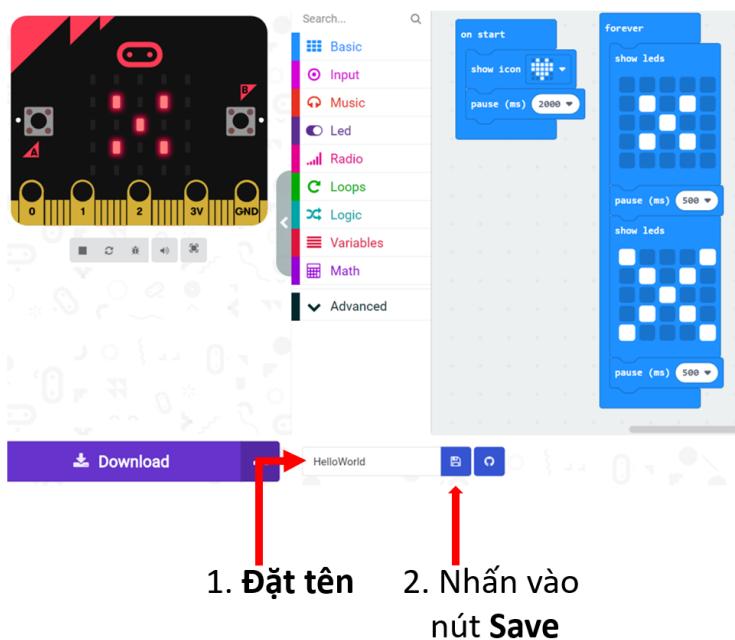
Bài tập: Học sinh viết một chương trình hiển thị hình trái tim (câu lệnh show icon) trong 2 giây đầu tiên. Sau đó, thiết kế hiệu ứng pháo hoa lặp đi lặp lại trên màn hình hiển thị.

Gợi ý: Do hình trái tim chỉ hiển thị 1 lần, nên câu lệnh show icon sẽ được dùng trong phần on start. Các câu lệnh tạo hiệu ứng pháo hoa sẽ được hiện thực trong phần forever. Một gợi ý cho chương trình này được trình bày ở Hình 2.7.

5 Lưu và mở lại chương trình

Một nhu cầu tất yếu của việc lập trình là Lưu và Mở lại chương trình. Trước tiên, để lưu chương trình hiện tại, chúng ta có 2 bước cơ bản, như minh họa ở Hình 2.8:

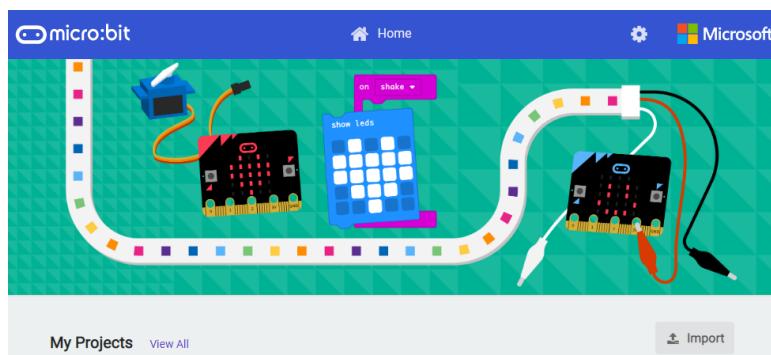
1. Đặt tên cho chương trình cần lưu
2. Nhấn vào nút Save và chọn đường dẫn để lưu



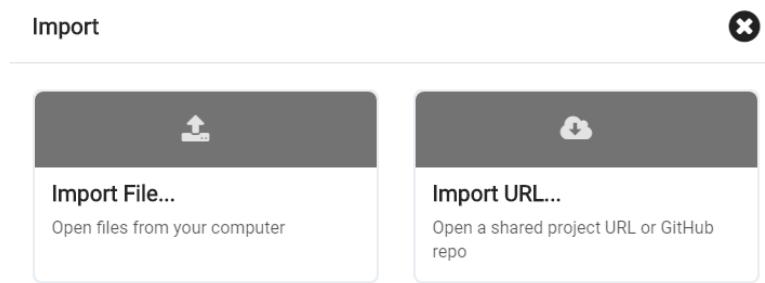
Hình 2.8: Các bước cơ bản để lưu chương trình

Lưu ý: Chương trình sẽ được lưu lại với định dạng file .hex

Để mở lại chương trình cũ và tiếp tục lập trình, từ màn hình chính, chúng ta chọn Import ở góc bên phải màn hình, như mô tả ở Hình 2.9:



Hình 2.9: Chọn vào Import để mở lại chương trình cũ

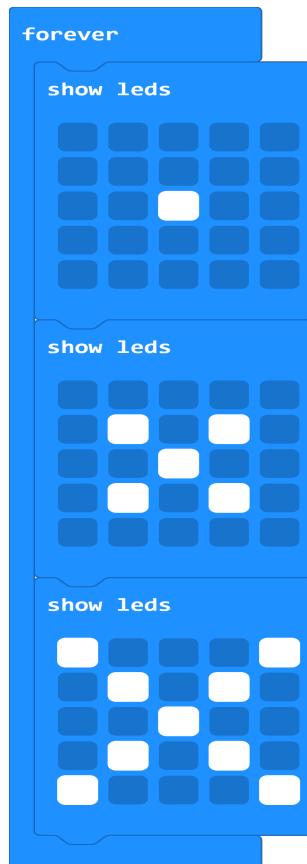


Hình 2.10: Chọn tiếp Import File và chọn tới đường dẫn file đã lưu trước đó

Một giao diện như Hình 2.11 sẽ hiện ra, chúng ta chọn tiếp Import File và chọn tới đường dẫn file hex đã lưu trước đó.

6 Bài tập

Thiết kế các hiệu ứng đẹp trên màn hình Microbit. Ví dụ, một chương trình nhỏ mô phỏng cho hiệu ứng pháo hoa như sau:



Hình 2.11: Hiệu ứng pháo hoa

7 Câu hỏi ôn tập

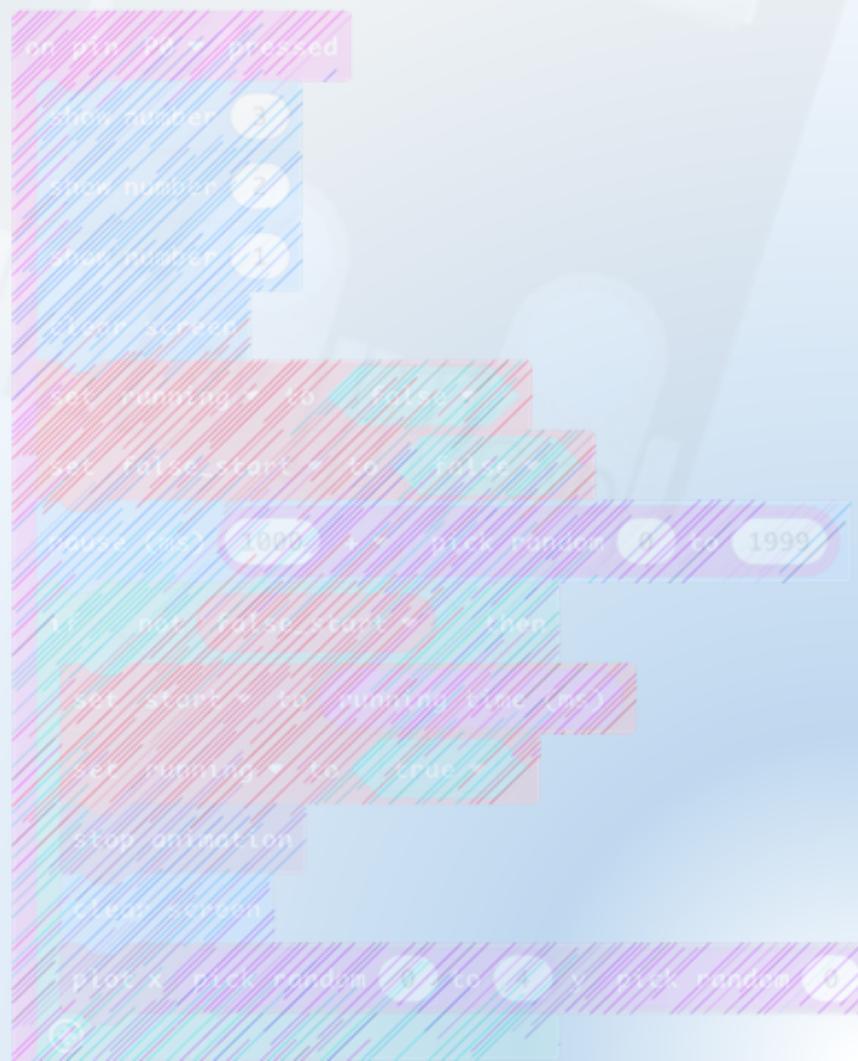
1. Câu lệnh show icon thuộc nhóm lệnh nào dưới đây?
 - A. basic
 - B. input
 - C. music
 - D. led
2. Câu lệnh show icon dung để làm gì?
 - A. Hiển thị một số có 2 chữ số.
 - B. Hiển thị một biểu tượng có sẵn.
 - C. Hiển thị một chuỗi ký tự.
 - D. Tất cả đều sai.
3. Phía dưới logo của mạch micro:bit chúng ta có bao nhiêu đèn led để hiển thị?
 - A. 10
 - B. 15
 - C. 20
 - D. 25
4. Các đèn led được sắp xếp như thế nào?
 - A. 3 hàng, 3 cột
 - B. 4 hàng, 4 cột
 - C. 5 hàng, 5 cột
 - D. Tất cả đều sai.
5. Câu lệnh show led nằm trong nhóm lệnh nào?
 - A. basic
 - B. input
 - C. music
 - D. led
6. Câu lệnh pause dùng để làm gì?
 - A. Tạo độ trễ cho chương trình
 - B. Tạm dừng việc nạp code
 - C. Tắt tất cả led
 - D. Tạm dừng việc xem video.
7. Muốn tạo một lệnh chờ 5 giây giữa các câu lệnh, ta dùng:
 - A. pause (ms) 5
 - B. pause (ms) 5000
 - C. pause (ms) 500
 - D. pause (ms) 50

Đáp án

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

CHƯƠNG 3

Tổng hợp các câu lệnh về hiển thị



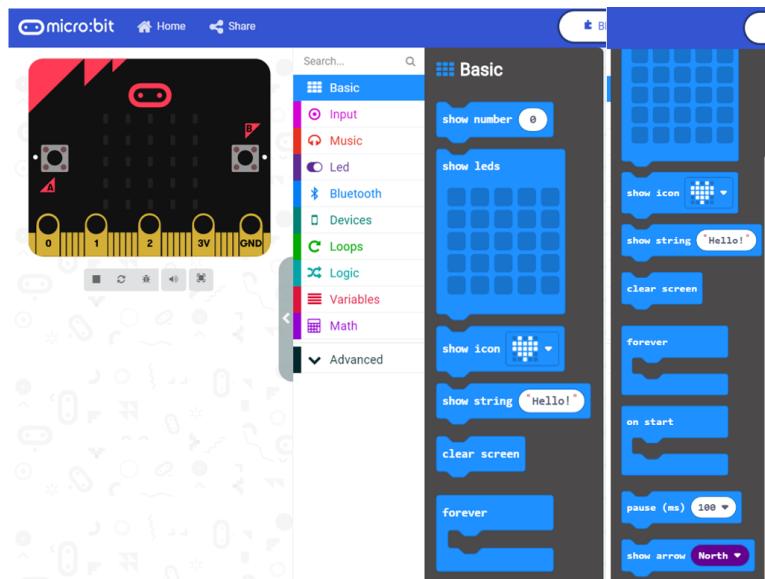
1 Giới thiệu

Trong bài hướng dẫn này, chúng tôi sẽ tập trung giới thiệu các câu lệnh quan trọng về hiển thị trên mạch Microbit. Các câu lệnh này thuộc nhóm lệnh **Basics**. Bên cạnh đó, hướng dẫn cũng hướng vào khả năng tự tìm hiểu chức năng của một câu lệnh trên môi trường lập trình MakeCode. Đây là kỹ năng quan trọng để người học có thể tự cập nhật thêm các câu lệnh mới trong tương lai và hiện thực những dự án phức tạp hơn. Các mục tiêu hướng đến của bài hướng dẫn này được liệt kê ra như sau:

- Học sinh tự tìm hiểu các câu lệnh về hiển thị trên MicroBit
- Học sinh có khả năng phối hợp nhiều câu lệnh
- Học sinh nắm vững được nguyên lý hiển thị trên MicroBit

2 Giới thiệu tổng quan về nhóm lệnh hiển thị

Các câu lệnh về hiển thị được nằm chính trong mục Basics. Sở dĩ các câu lệnh ở đây thuộc nhóm Basics vì đây là những tác vụ đơn giản nhất khi tiếp cận với việc lập trình trên bo mạch. Tất cả các câu lệnh này đều được MicroBit hỗ trợ để người dùng có thể hiển thị nhiều hình ảnh đẹp trên màn hình gồm 25 LED đơn.



Hình 3.1: Các câu lệnh về hiển thị được nằm trong mục Basics

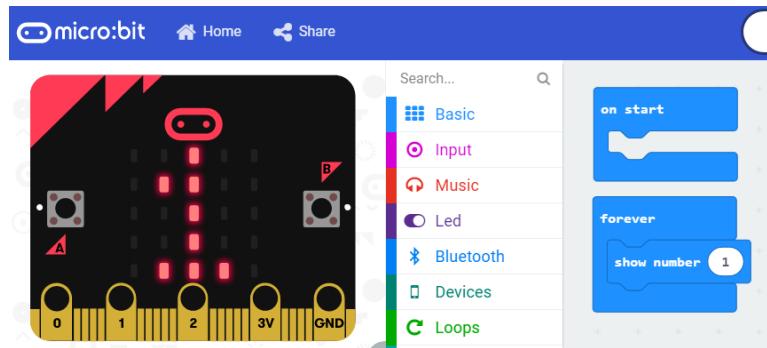
3 Giới thiệu chức năng của các câu lệnh

Cũng giống như các câu lệnh Scratch, chúng ta chỉ cần kéo thả câu lệnh này và đặt vào khôi lệnh on start hoặc forever, thì nó sẽ được thực thi trên sân khấu, với nhân vật là mạch mô phỏng MicroBit. Dựa vào kết quả của màn hình mô phỏng, chúng ta có thể tự tìm hiểu chức năng của câu lệnh này. Phần bên dưới sẽ trình bày chi

tiết chức năng của từng câu lệnh. Tuy nhiên người học có thể chủ động tự tìm hiểu, trước khi kiểm tra lại những gì mình đúc kết được với nội dung được trình bày bên dưới.

3.1 Câu lệnh show number

Câu lệnh này sẽ hiển thị một số ra màn hình. Tuy nhiên, nó chỉ hiện ra số có 1 chữ số từ 0 đến 9. Nếu số này có 2 chữ số trở lên, nó sẽ chạy ngang qua bên trái.

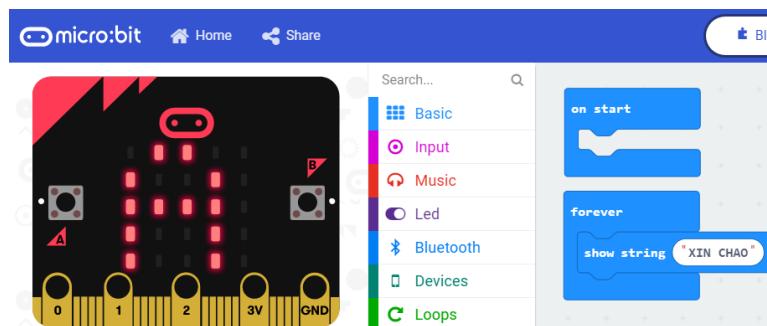


Hình 3.2: Ví dụ về câu lệnh show number

Ví dụ ở trên hiển thị số 12 ra màn hình. Tuy nhiên nó chỉ hiển thị ra số 1, sau đó dịch chuyển dần sang bên trái để hiển thị tiếp số 2. Thậm chí câu lệnh này cũng hiển thị được **số âm** (ví dụ -9) và cả số thập phân (ví dụ -9.3). Cách hiển thị cũng giống như việc hiển thị số có hơn 2 chữ số ra màn hình, nội dung sẽ được dịch sang trái màn hình.

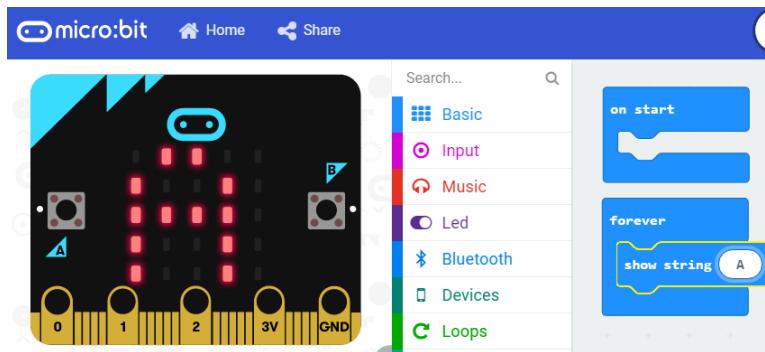
3.2 Câu lệnh show string

Hai câu lệnh tiếp theo là **show leds** và **show icon** đã được trình bày ở 2 bài trước, nên sẽ không trình bày lại ở bài này. Câu lệnh tiếp theo, **show string** sẽ cho phép một chuỗi dài các ký tự dịch chuyển sang trái. Tuy nhiên, câu lệnh này không hỗ trợ tiếng việt có dấu.



Hình 3.3: Ví dụ về câu lệnh show string

Ngoài ra, nếu tham số bên trong câu lệnh **show string** này chỉ có 1 kí tự (ví dụ chỉ hiển thị kí tự A), nó chỉ hiển thị kí tự này ra màn hình mà thôi và không có hiệu ứng dịch chữ sang bên trái. Ví dụ ở chương trình trong Hình 3.4, chỉ một kí tự A hiển thị cố định trên màn hình mà thôi.



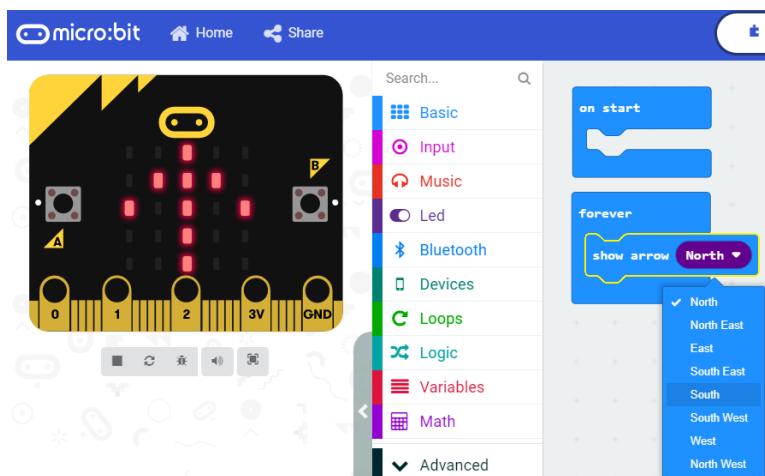
Hình 3.4: Câu lệnh show string khi có 1 kí tự sẽ không có hiệu ứng dịch trái

3.2.1 Câu lệnh clear screen

Chức năng câu lệnh này là xóa toàn bộ màn hình đang hiển thị. Tuy nhiên, chúng ta cũng có một câu lệnh với ý nghĩa tương tự câu lệnh này, chính là câu lệnh **show leds** ở bài trước, nhưng chúng ta sẽ không nhấp vào 1 ô đèn nào cả. Lúc đó, màn hình cũng sẽ tắt toàn bộ 25 đèn. Việc cung cấp nhiều câu lệnh có chức năng tương đương nhau là một cách để giúp ngôn ngữ lập trình gần với suy nghĩ tự nhiên của con người hơn. Cũng vì lý do đó, mà các ngôn ngữ kéo thả đang trở nên phổ biến hơn và dễ tiếp cận hơn với tất cả mọi người.

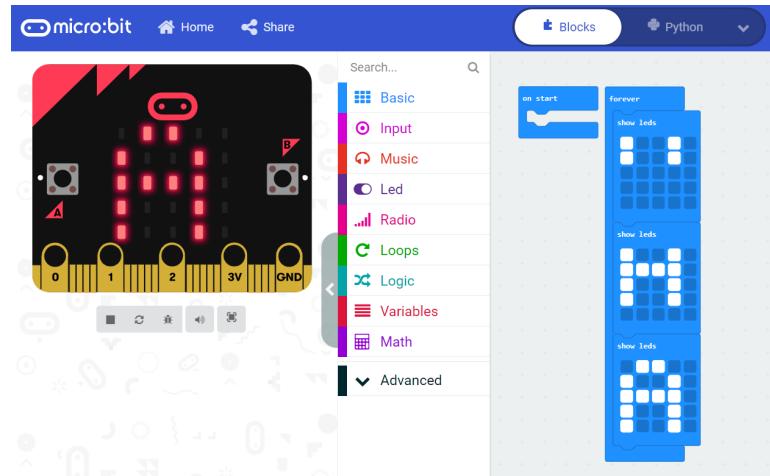
3.3 Câu lệnh show arrow

Cũng giống như câu lệnh **show icon**, câu lệnh này định nghĩa sẵn một số hình ảnh về mũi tên. Người dùng cũng có thể chọn từ danh sách hỗ trợ sẵn của MicroBit.



Hình 3.5: Câu lệnh show arrow

Đến đây, chúng ta có thể tạo ra rất nhiều hiệu ứng đẹp bằng cách xếp các câu lệnh hiển thị trong khối **forever**. Giữa các câu lệnh về hiển thị, câu lệnh **pause** có thể được chèn vào, để hình ảnh đó được hiển thị lâu hơn trên màn hình 25 đèn. Chúng ta hoàn toàn có thể tạo ra hiệu ứng chữ A rơi từ trên xuống, như minh họa ở chương trình bên dưới.



Hình 3.6: Tạo hiệu ứng chữ chạy từ trên xuống

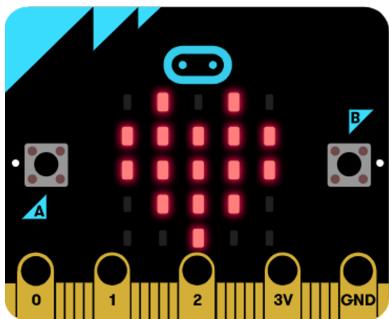
Chương trình của chúng ta có thể hơi dài với nhiều câu lệnh để hiển thị. Tuy nhiên, chương trình dài không phải là vấn đề đối với hệ thống vi điều khiển như MicroBit. Với các ứng dụng thực tế, chúng ta quan tâm đến tính ổn định và bền bỉ của chương trình hơn là việc viết chương trình cho gọn.

4 Bài tập

Phối hợp các câu lệnh để tạo ra các hiệu ứng đẹp trên màn hình hiển thị của mạch MicroBit.

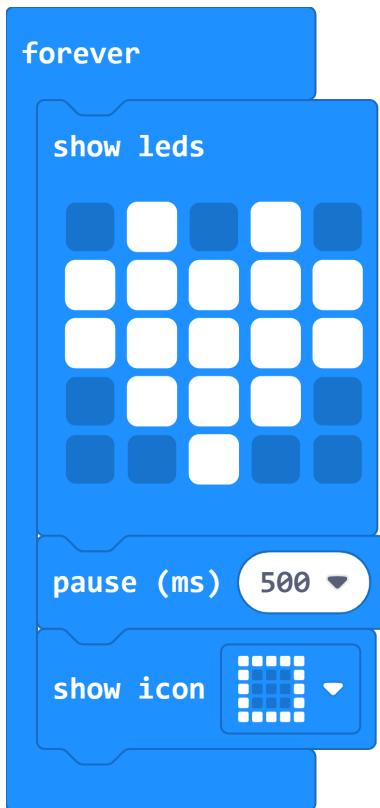
5 Câu hỏi ôn tập

1. Câu lệnh **show number** dùng để làm gì?
 - A. Hiển thị một chuỗi ký tự
 - B. Hiển thị một giá trị số
 - C. Hiển thị một biểu tượng
 - D. Hiển thị mũi tên chỉ hướng.
2. Câu lệnh **show string** dùng để làm gì?
 - A. Hiển thị một chuỗi ký tự
 - B. Hiển thị một giá trị số
 - C. Hiển thị một biểu tượng
 - D. Hiển thị mũi tên chỉ hướng.
3. Chức năng của câu lệnh **clear screen**:
 - A. Hiển thị một chuỗi ký tự
 - B. Hiển thị một giá trị số
 - C. Xóa màn hình hiển thị
 - D. Hiển thị mũi tên chỉ hướng.
4. Chức năng của câu lệnh **show arrow**:
 - A. Hiển thị một chuỗi ký tự
 - B. Hiển thị một giá trị số
 - C. Xóa màn hình hiển thị
 - D. Hiển thị mũi tên chỉ hướng.
5. Để hiển thị hình “trái tim” ❤ trên bảng mạch MicroBit, ta vào:



- A. Basic -> Show icon ❤
- B. Input -> Show leds ❤
- C. Led -> Show heart
- D. Tất cả đều sai

6. Đoạn code bên dưới có ý nghĩa là gì ?



- A. Hiển thị trái tim, chờ 0.5 giây, hiển thị hình vuông .
- B. Hiển thị trái tim rồi hiện hình vuông ngay
- C. Chỉ hiển thị hình trái tim
- D. Hiển thị trái tim, chờ 500 giây sau đó hiển thị hình vuông

Đáp án

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



CHU'ƠNG 4

Điều khiển nút nhấn trên MicroBit

1 Giới thiệu

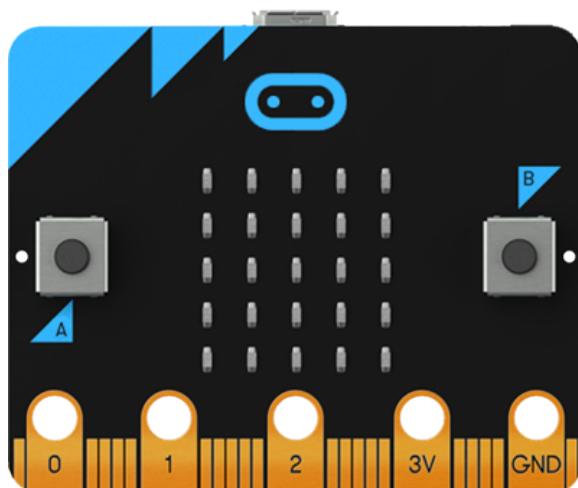
Khác với việc lập trình trên máy tính, chương trình chủ yếu tập trung vào giải thuật. Khi làm việc trên những hệ thống như MicroBit, vốn được thiết kế hướng đến ứng dụng thực tế, công sức lập trình sẽ hướng tới việc xử lý dữ liệu đầu vào (input) và xuất dữ liệu đầu ra (output). Giữa 2 quá trình này, xuất dữ liệu đầu ra đơn giản hơn, nên thường sẽ được hướng dẫn trước khi người học mới tiếp cận với những hệ thống bo mạch như Microbit.

Trong bài này, chúng ta sẽ làm việc với việc nhận dữ liệu từ mạch MicroBit. Cụ thể, trên mạch Microbit có 2 nút nhấn A và B để có thể tương tác với người dùng. Các mục tiêu trong bài hướng dẫn này được trình bày như bên dưới:

- Học sinh hiểu được nguyên lý hoạt động của nút nhấn
- Học sinh có khả năng sử dụng các câu lệnh liên quan đến nút nhấn
- Học sinh kết hợp được nhiều câu lệnh trên MicroBit

2 Tổng quan về nút nhấn

Trên mạch MicroBit được hỗ trợ sẵn 2 nút nhấn, có tên gọi là A và B như Hình 4.1 bên dưới. Cũng giống như bàn phím máy tính, hai nút nhấn này đóng vai trò là **các thiết bị nhập, gửi dữ liệu đầu vào**.

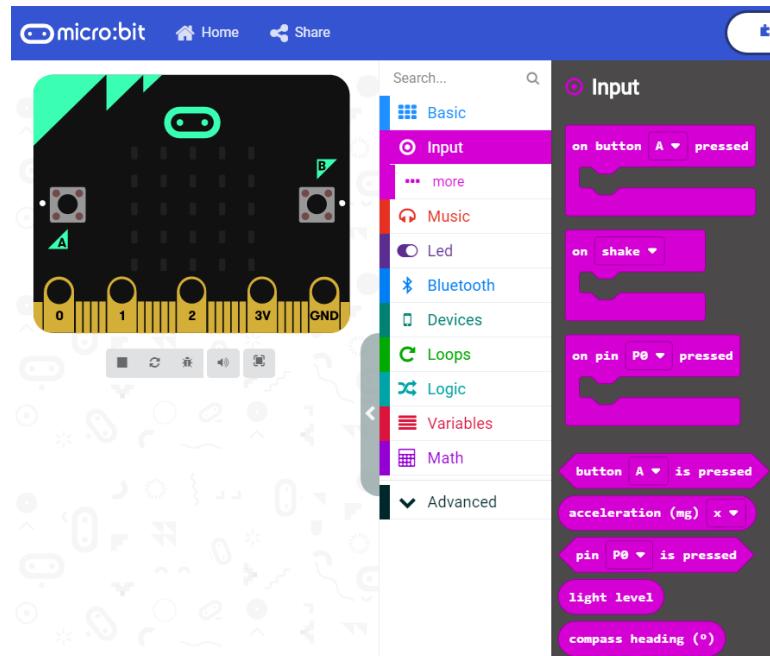


Hình 4.1: Hai nút nhấn A và B trên MicroBit

Thực ra, đọc dữ liệu từ nút nhấn là một tác vụ rất phức tạp. Lý do là tín hiệu của nó không ổn định (gọi là rung phím) khi phím được nhấn. Tuy nhiên, may mắn cho chúng ta là môi trường MakeCode đã xử lý gần hết, và việc lập trình lại trở nên vô cùng dễ dàng.

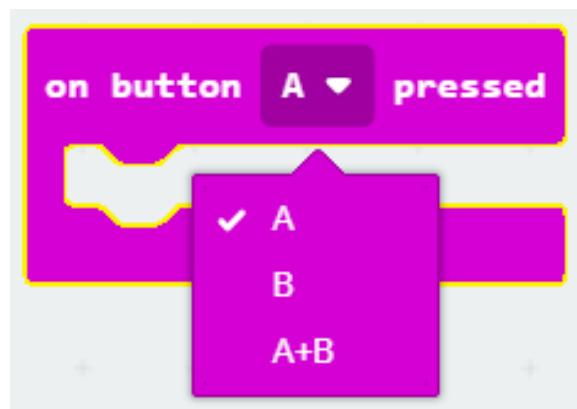
3 Lập trình điều khiển nút nhấn

Các câu lệnh để điều khiển nút nhấn thuộc nhóm Input, nhóm thứ 2 sau nhóm Basic, như trình bày ở Hình 4.2.



Hình 4.2: Các câu lệnh thuộc nhóm Input để điều khiển nút nhấn

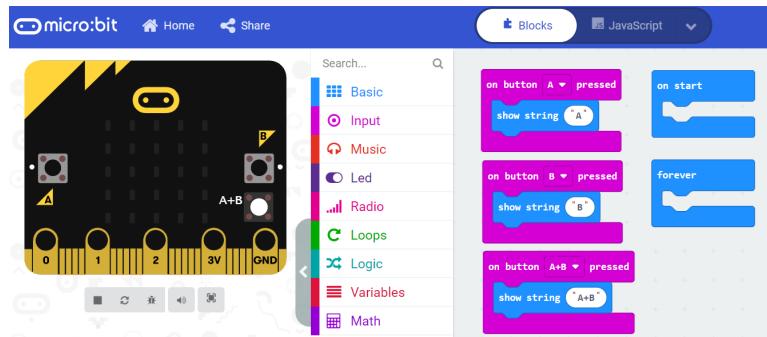
Khác với kiến trúc chương trình ở các bài trước, các câu lệnh được thực hiện đầu tiên trong khối **on start**, sau đó chuyển qua phần **forever** và lập đi lập lại, việc điều khiển nút nhấn được thực hiện trong các sự kiện. Đây là một kiến trúc lập trình rất hiện đại, cải thiện rất nhiều hiệu suất của hệ thống.



Hình 4.3: Câu lệnh sự kiện cho nút nhấn với 3 lựa chọn khác nhau

Chúng ta sẽ hiển thị thực một chương trình đơn giản đầu tiên, khi nhấn phím A sẽ hiện ra chữ A trên màn hình, nhấn phím B sẽ hiện ra chữ B. Câu lệnh chính trong

yêu cầu này là **on button pressed**, câu lệnh đầu tiên trong nhóm Input. Trong câu lệnh này, chúng ta có thể lựa chọn sự kiện tương ứng cho từng nút nhấn A, B hoặc cả 2 nút nhấn cùng một lúc (xem chi tiết ở Hình 4.3). Chương trình của chúng ta được gợi ý như Hình 4.4:



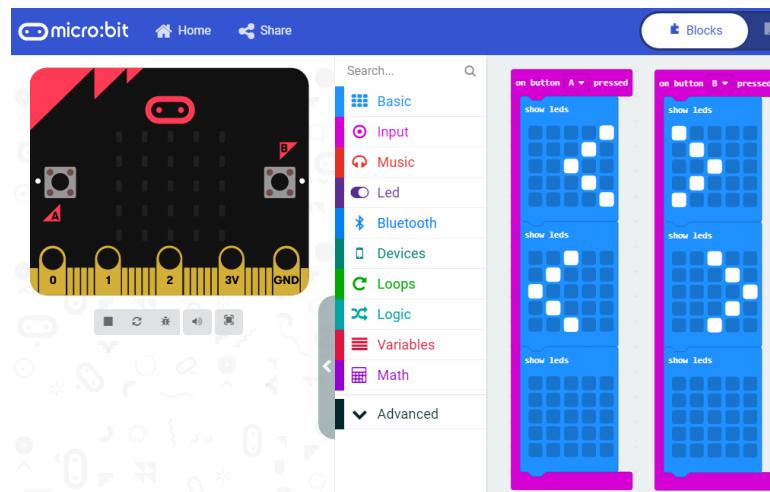
Hình 4.4: Chương trình đơn giản tương tác với nút nhấn

Như chúng ta có thể thấy, chương trình thực hiện hoàn toàn độc lập với 2 khối lệnh **on start** và **forever**. Những chương trình này hoàn toàn có thể mô phỏng được trực tuyến. Hãy lưu ý ở màn hình mô phỏng mạch MicroBit, một nút ảo có tên **A+B** được sinh ra để mô phỏng cho việc chúng ta nhấn cùng lúc 2 nút A và B.

4 Bài tập

1. Thiết kế một hiệu ứng là dấu mũi tên. Khi nhấn phím A, phím mũi tên này di chuyển sang trái. Khi nhấn phím B, phím mũi tên này di chuyển sang phải. Đáp án gợi ý của chương trình như Hình 4.5.

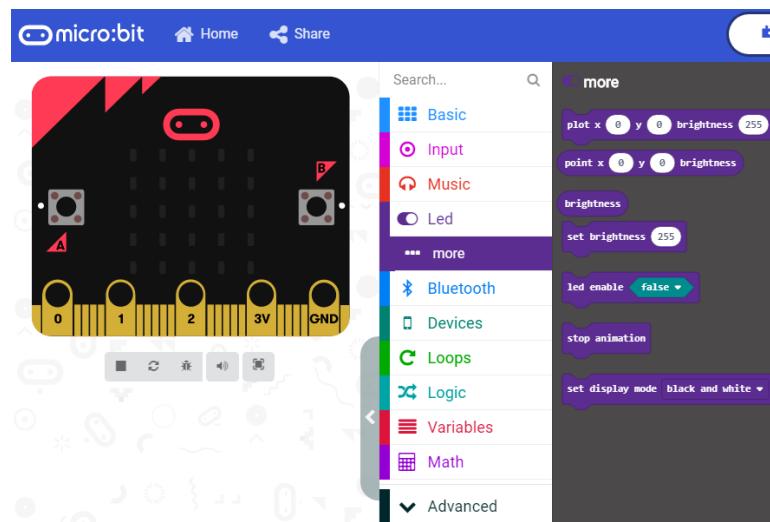
Giải thích: Hiệu ứng thực ra là nhiều màn hình hiển thị xuất hiện cách nhau 1 khoảng thời gian đủ nhỏ. Chẳng hạn như video mà chúng ta đang xem, thực ra là hình tĩnh cách nhau 1/30s (30 hình một giây). Chúng ta sẽ sử dụng nguyên lý này để định nghĩa ra hàng loạt các khung hình bằng câu lệnh show leds. Sau đó đặt các màn hình này liên tiếp nhau, chúng ta sẽ thấy một hiệu ứng.



Hình 4.5: Hiệu ứng mũi tên qua trái qua phải

2. Học sinh tự thiết kế các hiệu ứng khác nhau ví dụ chữ dịch lên, dịch xuống, chữ nhấp nháy và thay đổi hiệu ứng của chữ bằng cách nhấn phím A hoặc B hoặc kết hợp cả 2 phím. Học sinh trình bày kết quả của mình vào buổi hôm sau.

Để có thể làm thêm các hiệu ứng đẹp, học sinh có thể sử dụng thêm 1 lệnh trong phần Led, chọn tiếp vào more. Chúng ta có lệnh **set brightness** để chỉnh độ sáng của màn hình hiển thị. Học sinh có thể dùng câu lệnh này để cho chữ sáng từ mờ sang rõ hoặc ngược lại.



Hình 4.6: Các câu lệnh mới để điều khiển độ sáng

5 Câu hỏi ôn tập

1. Có bao nhiêu nút nhấn trên mạch micro:bit ?
 - A. Một nút
 - B. Hai nút
 - C. Ba nút
 - D. Bốn nút
2. Nút reset có chức năng gì?
 - A. Tắt nguồn mạch
 - B. Nhập xuất người dùng
 - C. Khởi động lại mạch
 - D. Dự phòng trường hợp thay thế nút
3. Nút A và B có chức năng gì?
 - A. Tắt nguồn mạch
 - B. Nhập xuất người dùng
 - C. Khởi động lại mạch
 - D. Dự phòng trường hợp thay thế nút
4. Các câu lệnh về nút nhấn thuộc nhóm lệnh nào?
 - A. basic
 - B. input
 - C. music
 - D. led
5. Vị trí câu lệnh câu lệnh on button [A / B / A+B] pressed được đặt ở đâu trong chương trình?
 - A. Trong forever
 - B. Trong on start
 - C. Độc lập với forever và on start
 - D. Tất cả đều sai
6. Với câu lệnh on button [A / B / A+B] pressed, khi điều kiện các nút được nhấn thì hiện tượng gì sẽ xảy ra?
 - A. Tắt điện cho mạch
 - B. Khởi động lại mạch
 - C. Code vẫn chạy trong forever
 - D. Đoạn code trong khối on button [A / B / A+B] pressed được chạy ngay.
7. Để phát nhạc trên micro:bit ta dùng nhóm lệnh:
 - A. basic
 - B. input
 - C. music
 - D. led

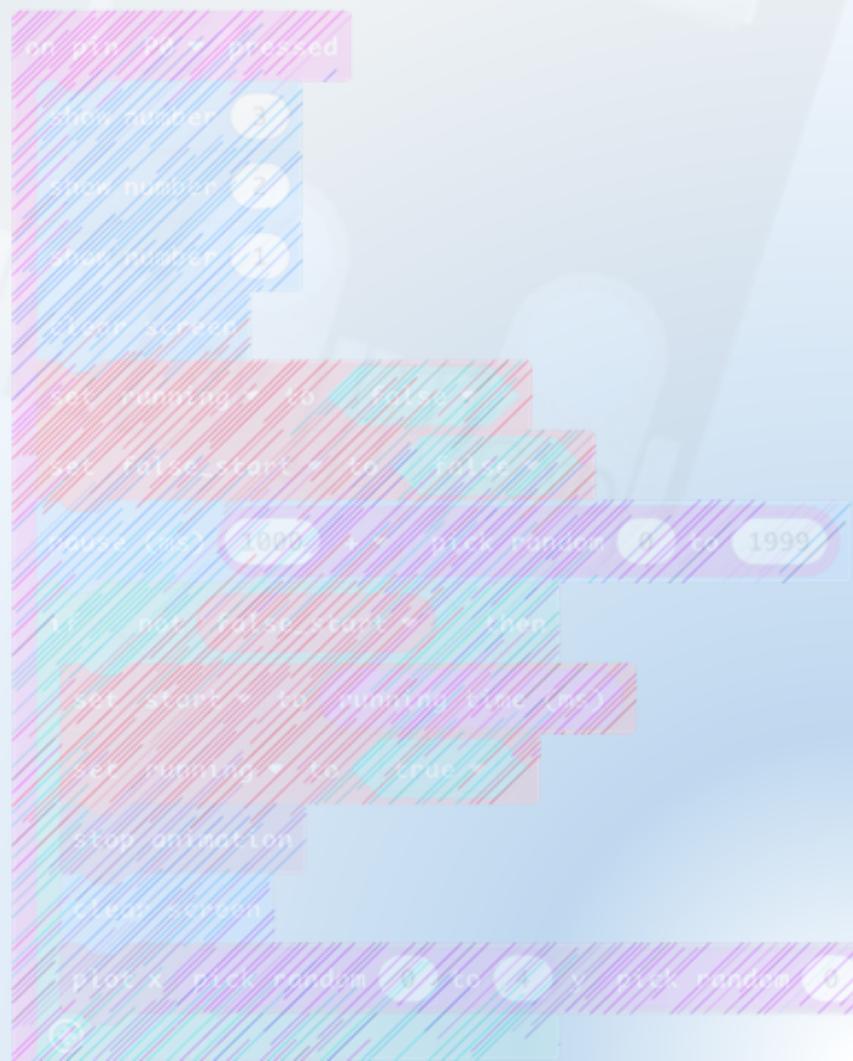
Đáp án

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



CHƯƠNG 5

Lập trình MicroBit bằng điện thoại thông minh



1 Giới thiệu

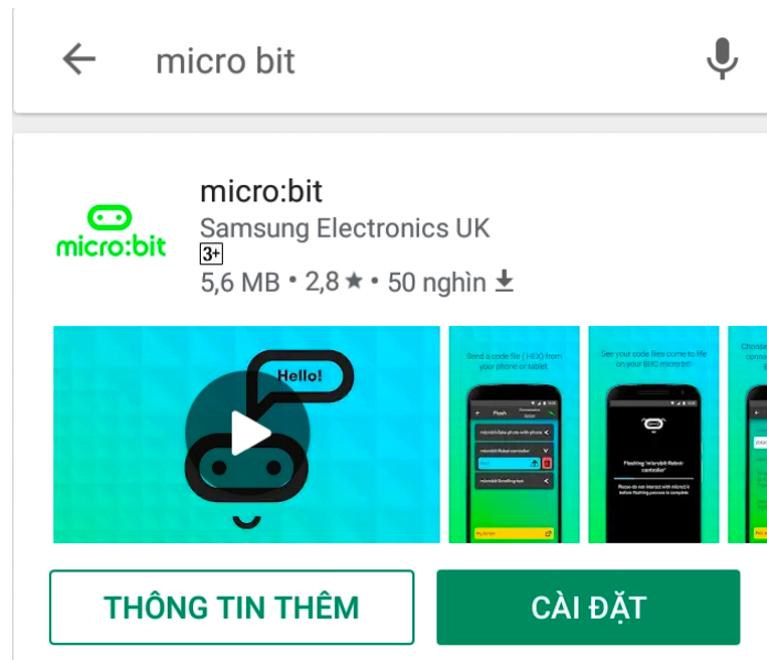
Trong các bài đã giới thiệu, chúng ta đang lập trình bằng cách dùng máy tính, kết nối vào mạng Internet và vào trang web www.makecode.makecode.org để lập trình. Tuy nhiên trong 1 số trường hợp, việc sử dụng máy tính là khá công kẽm và không thuận tiện. Trong những trường hợp này, chúng ta có thể sử dụng điện thoại thông minh để lập trình cho mạch MicroBit. Mặc dù hiện tại, các thao tác trên điện thoại còn chưa được thuận tiện, tuy nhiên nó cũng là một công nghệ mới, đáng ghi nhận của bo mạch MicroBit. Với khả năng lập trình bằng điện thoại cũng như máy tính bảng, tính phổ dụng trong việc triển khai lập trình trên bo mạch MicroBit càng được nâng cao, theo đúng tinh thần phát triển rộng rãi tư duy ngôn ngữ lập trình đến các em học sinh.

Trong bài hướng dẫn này, chúng tôi sẽ trình bày các bước cơ bản để lập trình cho mạch MicroBit bằng điện thoại thông minh, cho cả 2 nền tảng là Android và iOS.

- Học sinh nắm được thao tác kích hoạt mạch MicroBit vào chế độ Bluetooth
- Học sinh có thể viết và nạp chương trình bằng điện thoại Android
- Học sinh có thể viết và nạp chương trình bằng điện thoại iOS

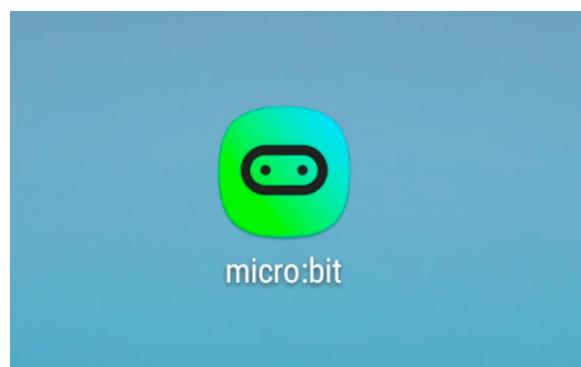
2 Lập trình bằng điện thoại Android

Để có thể lập trình trên Android, chúng ta cần vào kho ứng dụng Google Play để tải phần mềm mico:bit. Chúng ta sẽ tìm kiếm bằng từ khóa microbit như hình minh họa bên dưới, sau đó nhấn nút CÀI ĐẶT để tải ứng dụng này về.

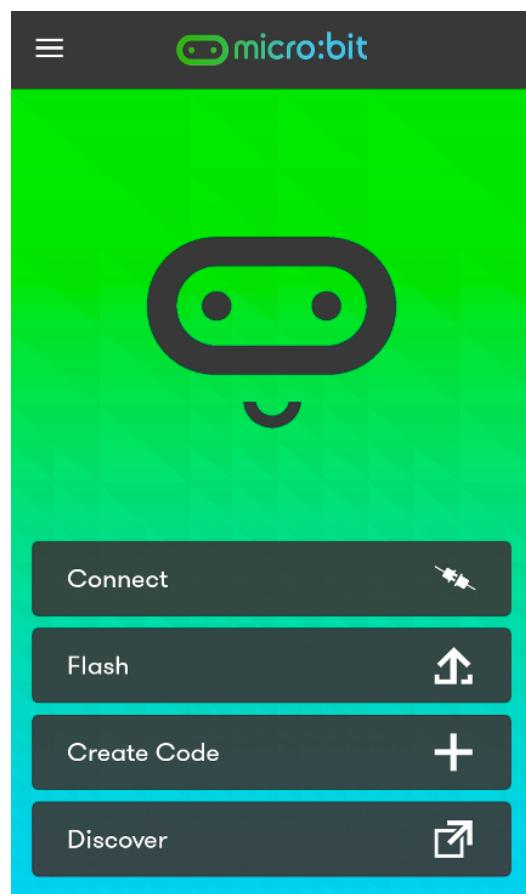


Hình 5.1: Tải và cài đặt ứng dụng MicroBit

Khi cài đặt xong, trên màn hình điện thoại sẽ xuất hiện ứng dụng để lập trình cho mạch MicroBit (xem Hình 5.2). Khi nhấn vào icon này, ứng dụng sẽ được mở ra như Hình 5.3. Bạn hãy chọn **Đồng ý** hoặc **Cho phép**, để ứng dụng được cấp quyền thực thi trên điện thoại Android.



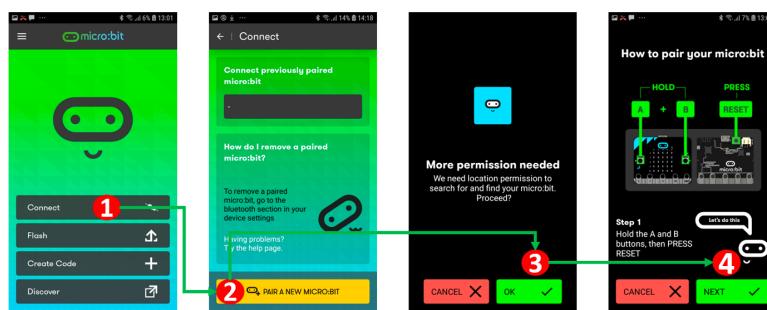
Hình 5.2: Icon của ứng dụng MicroBit



Hình 5.3: Giao diện của ứng dụng

2.1 Ghép đôi điện thoại và mạch MicroBit

Đây là bước đầu tiên mà chúng ta phải làm để có thể lập trình trên MicroBit bằng điện thoại. Do việc nạp chương trình được dựa trên nền tảng giao tiếp Bluetooth. Tuy nhiên bước này **chỉ cần thực hiện một lần** mà thôi. Để có thể ghép đôi với mạch MicroBit, chúng ta cần bật nguồn cho mạch MicroBit. Sau đó, nhấn vào nút Connect trên điện thoại.

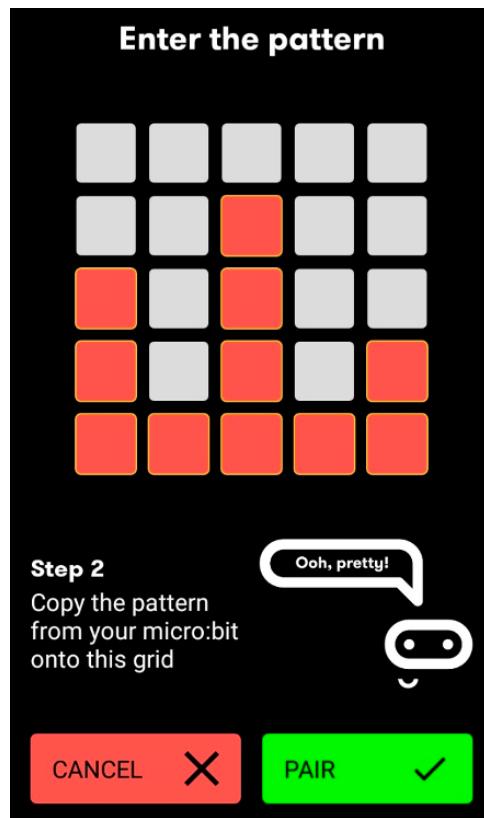


Hình 5.4: Các thao tác trên điện thoại để ghép đôi thiết bị

Trình tự ghép đôi được minh họa ở Hình 5.4, được mô tả như sau:

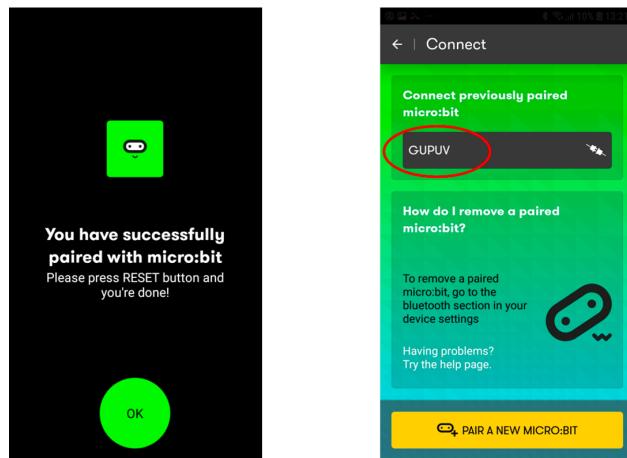
- Bước 1: Nhấn vào nút Connect
- Bước 2: Nhấn tiếp vào nút PAIR A NEW MICRO:BIT
- Bước 3: Nhấn vào nút OK hoặc Đồng ý để cấp quyền cho ứng dụng

Màn hình ở Bước 4 sẽ hiện lên, cung cấp hướng dẫn cho chúng ta để kích hoạt chế độ Bluetooth trên board mạch MicroBit. Chúng ta **nhấn giữ đồng thời 2 nút A và B trong khoảng 2 giây, sau đó nhấn vào nút Reset, thả nút Reset nhưng vẫn tiếp tục giữ 2 nút A và B**. Chúng ta sẽ thấy một số hiệu ứng hiển thị trên màn hình MicroBit cho đến khi màn hình ổn định ở một ký hiệu. Đến lúc này, chúng ta có thể thả 2 nút A và B ra. Toàn bộ quy trình ở trên là để kích hoạt chế độ ghép nối Bluetooth. Màn hình cuối cùng chính là mật mã để kết nối với board mạch MicroBit. Đến lúc này, chúng ta có thể nhấn vào nút Next ở bước 4 trên điện thoại. Giao diện sau đây trên điện thoại sẽ hiện ra, để chúng ta nhập mật mã.



Hình 5.5: Nhập mật mã kết nối với MicroBit

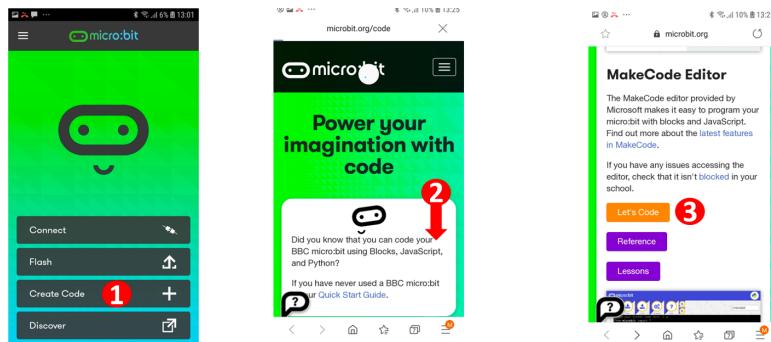
Hãy để ý hình dạng của các bóng đèn trên MicroBit, sau đó nhập vào màn hình điện thoại để có hình dạng tương ứng. Cuối cùng nhấn nút PAIR. Khi việc kết nối là thành công, màn hình sau đây sẽ hiện ra, chúng ta có thể nhấn nút OK. Một màn hình khác sẽ hiện ra, cùng với tên của board mạch MicroBit của chúng ta.



Hình 5.6: Kết nối thành công, tên thiết bị hiện ra trên mục ghép nối

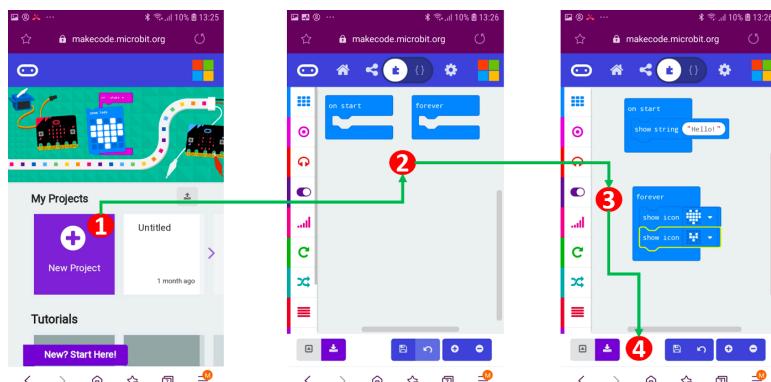
2.2 Lập trình trên điện thoại

Sau khi ghép đôi xong, chúng ta có thể nhấn nút BACK trên điện thoại để trở về giao diện chính. Tại đây, chúng ta có thể nhấn vào nút Create Code để bắt đầu lập trình cho mạch MicroBit.



Hình 5.7: Bắt đầu lập trình cho MicroBit

Toàn bộ quá trình này được minh họa ở Hình 5.7, nhấn vào nút Create Code (Bước 1), kéo màn hình xuống phần **MakeCode Editor** (Bước 2), và chọn **Let's Code** (Bước 3).



Hình 5.8: Các bước để viết chương trình trên điện thoại

Giao diện để lập trình trên điện thoại hoàn toàn tương tự với phiên bản trên máy tính, chúng ta cũng chọn vào New Project (Bước 1), một giao diện mặc định sẽ hiện ra để bắt đầu lập trình (Bước 2). Chúng ta chọn các câu lệnh và ghép nối chúng lại (Bước 3). Cuối cùng nhấn vào nút Tải về (Bước 4) để tải chương trình về điện thoại. Chúng ta cũng có thể đổi tên file tải về cho gợi nhớ (xem Hình 5.9). Trong ví dụ này, chúng tôi đổi file hex mặc định có tên là **test.hex**.

Tải file về?

Đã có một file với tên bên dưới. Hãy đổi tên file.

Kích thước: 595,4KB

Tên

microbit-Untitled.hex

Tên file đã được sử dụng.

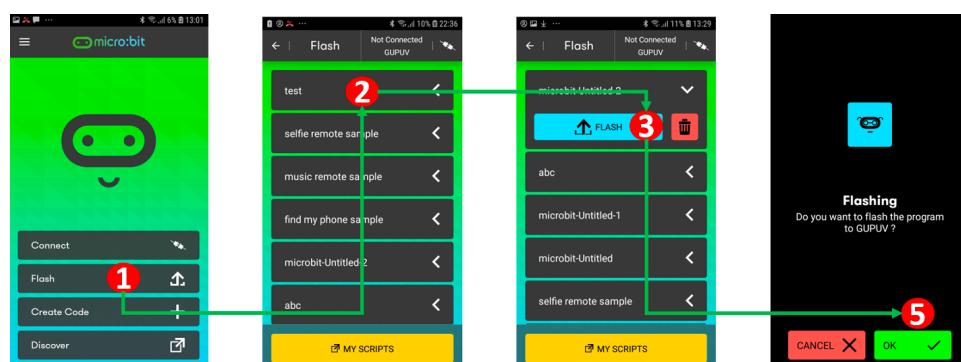
Thoát

Tải về

Hình 5.9: Đặt tên cho chương trình tải về

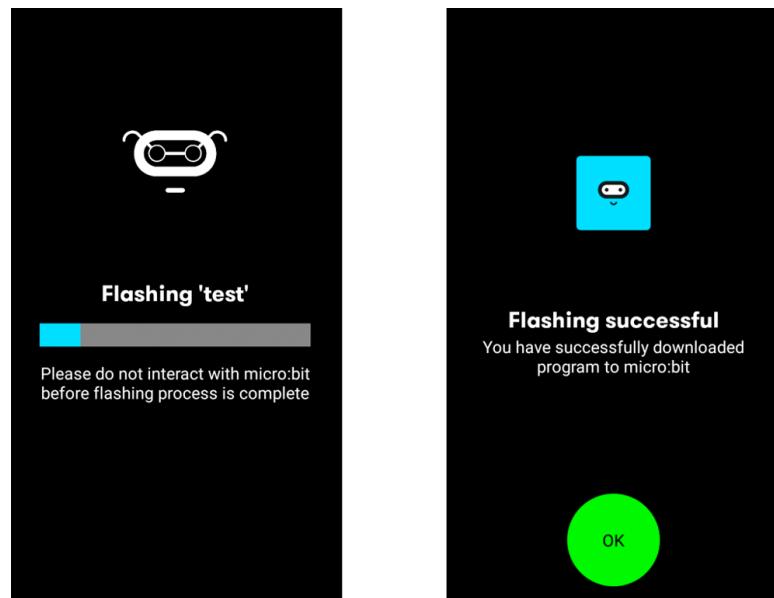
2.3 Nạp chương trình vào mạch MicroBit

Để nạp chương trình cho mạch MicroBit, chúng ta trở về màn hình chủ của điện thoại (nhấn nút HOME hoặc nhấn BACK nhiều lần). Từ màn hình chính, chúng ta có thể làm tuân tự như mô tả ở Hình 5.10.



Hình 5.10: Các bước để nạp chương trình cho MicroBit

Đầu tiên, chúng ta nhấn vào nút Flash, sau đó chọn vào file hex đã lưu trước đó (trong ví dụ này là file test). Tiếp theo chúng ta nhấn và chọn Flash. **Tuy nhiên, sau bước thứ 3 này, chúng ta cần phải bật mạch MicroBit sang chế độ Bluetooth,** trước khi nhấn vào nút OK ở bước 5. Làm tương tự như bước ghép đôi mạch MicroBit và điện thoại, **chúng ta nhấn đè 2 nút A và B, nhấn và thả nút Reset, vẫn tiếp tục nhấn đè 2 nút A và B cho tới khi hiện ứng báo hiệu kết nối Bluetooth sẵn sàng trên mạch MicroBit, chúng ta thả 2 nút A và B ra.** Cuối cùng, chúng ta mới nhấn nút OK ở bước 5. Các màn hình dưới đây sẽ lần lượt xuất hiện, báo hiệu việc nạp chương trình cho mạch MicroBit thành công.

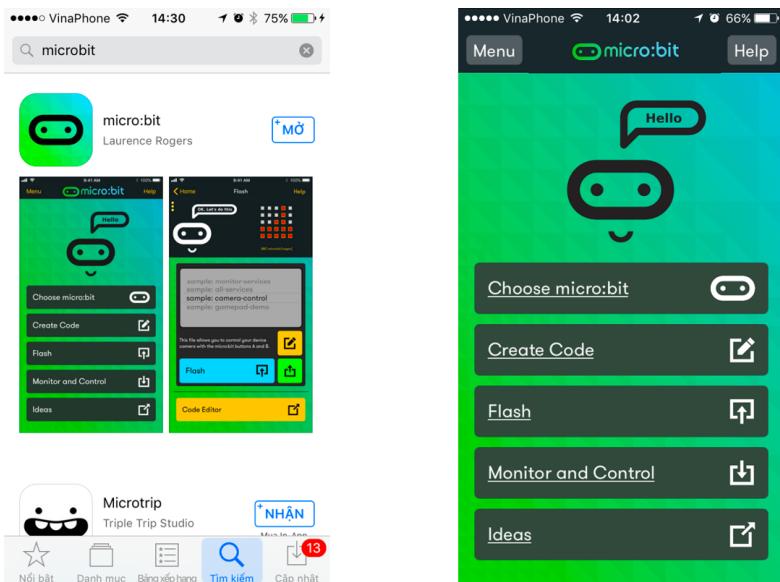


Hình 5.11: Quá trình tải chương trình lên mạch MicroBit và báo hiệu nạp thành công

Đến bước này, chương trình sẽ hỏi chúng ta có muốn tiếp tục kết nối với mạch MicroBit nữa hay không. Chúng ta có thể chọn NO vì việc này không cần thiết nữa. Nếu chương trình chạy không đúng ý, chúng ta sẽ quay lại mục 2 để soạn thảo lại chương trình, tải nó về và nạp lại chương trình. Trước khi nhấn OK để nạp, dùng quên trình tự nhấn đè 2 nút A và B, nhấn và thả nút Reset, chờ chế độ Bluetooth kích hoạt và thả 2 nút A và B.

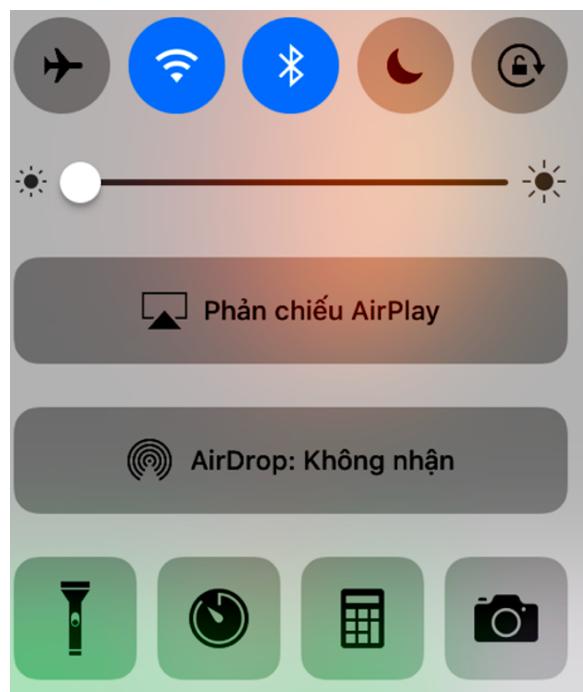
3 Lập trình bằng điện thoại iOS

Hoàn toàn tương tự với phiên bản chạy trên Android, chúng ta cũng vào kho ứng dụng AppStore, tìm kiếm từ khóa microbit để tải ứng dụng này về điện thoại. Sau khi tải và cài đặt, chúng ta có thể mở chương trình này lên. Giao diện của nó rất giống với phiên bản chạy trên Android.



Hình 5.12: Tìm kiếm và tải ứng dụng MicroBit về thiết bị iOS

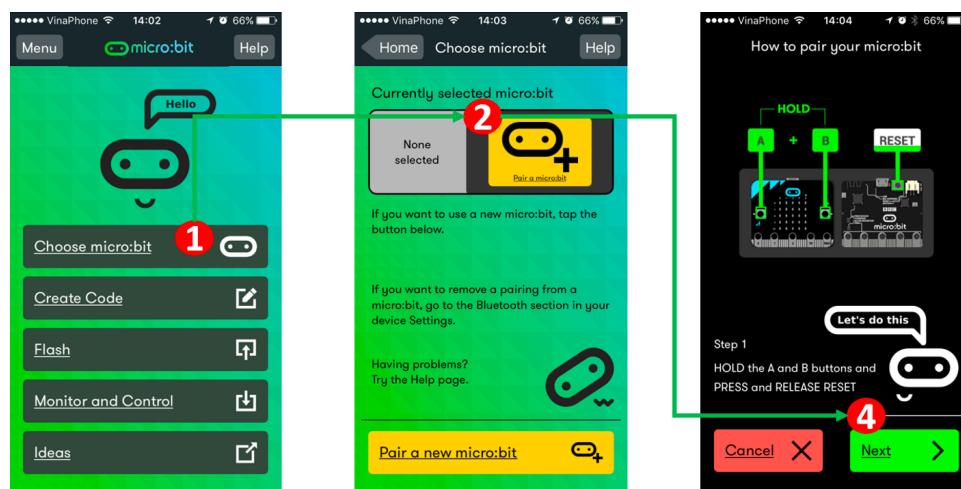
Tất nhiên, để có thể sử dụng được kết nối Bluetooth, chúng ta phải bật Bluetooth của điện thoại lên trước, bằng cách vào mục Setting, rồi bật kí hiệu Bluetooth (nằm bên cạnh kí hiệu wifi) lên như minh họa ở Hình 5.13.



Hình 5.13: Bật kết nối Bluetooth của điện thoại iPhone

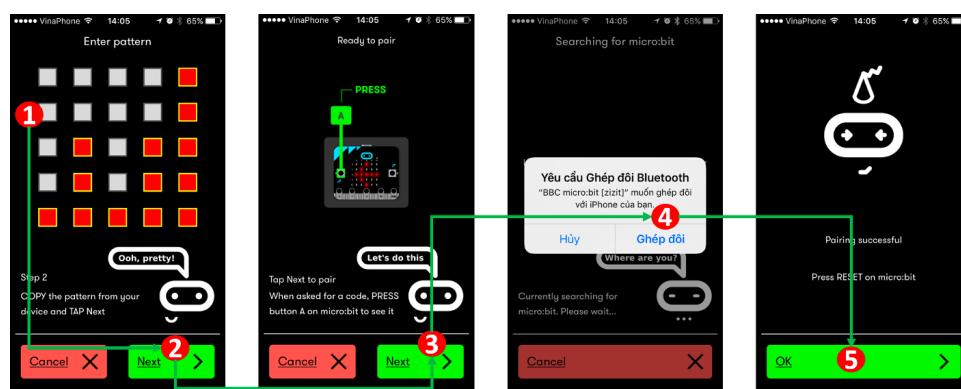
3.1 Ghép đôi thiết bị

Để thực hiện chức năng này, như minh họa ở Hình 5.12, chúng ta chọn vào lựa chọn đầu tiên là **Choose micro:bit**, chọn tiếp Pair a micro:bit. Tới đây, tương tự như khi làm việc với thiết bị Android. Chúng ta khoan bấm nút Next (Bước 4) mà chuyển sang mạch MicroBit, nhấn theo trình tự: Đè 2 nút A và B, nhấn và thả nút RESET, chờ cho đến khi tín hiệu kết nối Bluetooth kích hoạt, thả 2 nút A và B. Tới đây, chúng ta có thể quay lại điện thoại, để nhấn tiếp nút OK.



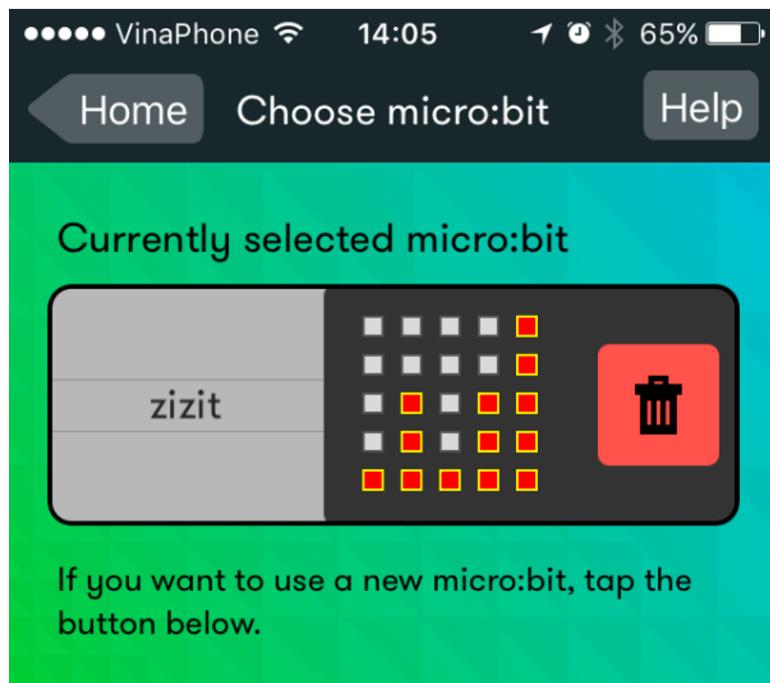
Hình 5.14: Ghép đôi một mạch MicroBit trên iOS

Các bước tiếp theo được minh họa ở Hình 5.15, bao gồm nhập mã hiển thị trên MicroBit (Bước 1), nhấn Next (Bước 2), tiếp tục nhấn Next (Bước 3), chọn Ghép đôi khi có thông báo hiển thị (Bước 4), và cuối cùng, chọn OK để kết thúc việc ghép đôi.



Hình 5.15: Nhập mã để ghép đôi thiết bị

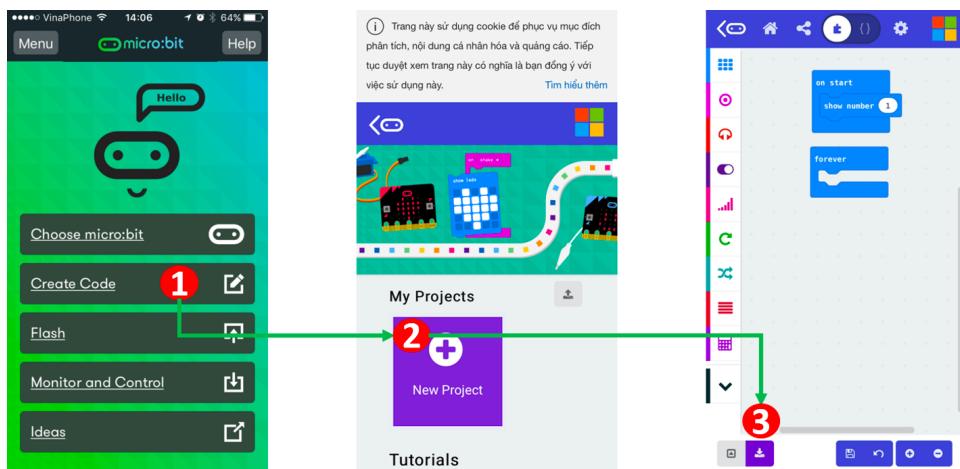
Sau khi nhấn OK ở bước 5, chúng ta sẽ được quay trở lại màn hình trong Bước 2 của Hình 5.14. Tuy nhiên, lúc này trong danh sách của chúng ta đã có 1 thiết bị MicroBit. Tất nhiên chúng ta có thể xóa thiết bị này đi và ghép nối với một thiết bị khác bằng cách nhấn vào biểu tượng Remove.



Hình 5.16: Một thiết bị MicroBit đã được thêm vào danh sách

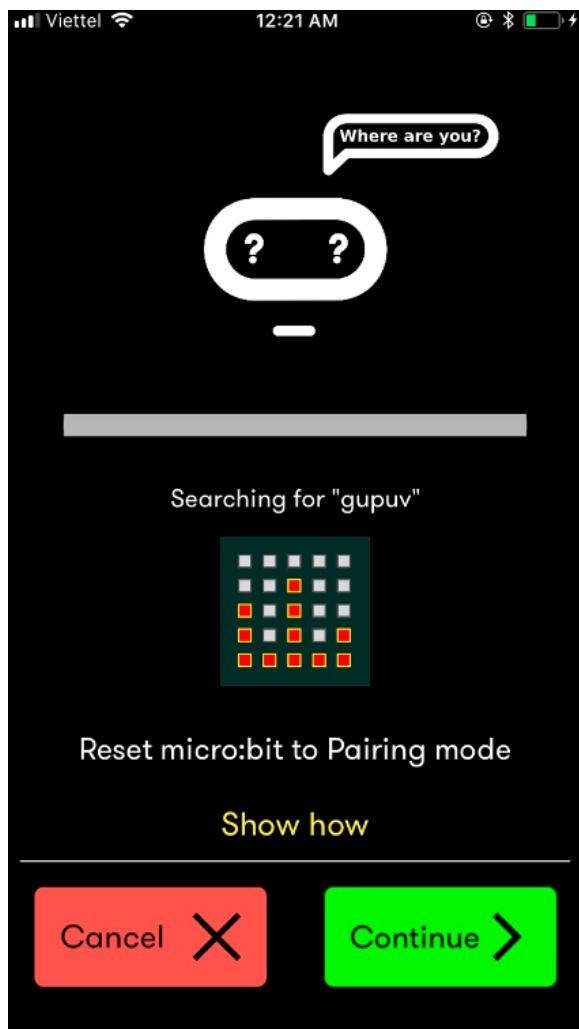
3.1.1 Lập trình và nạp chương trình lên MicroBit

Trở về màn hình Home của ứng dụng này, chúng ta có thể nhấn vào tùy chọn thứ 2, Create Code (xem Hình 5.17). Một giao diện quen thuộc với chúng ta sẽ hiện ra để chúng ta lập trình.



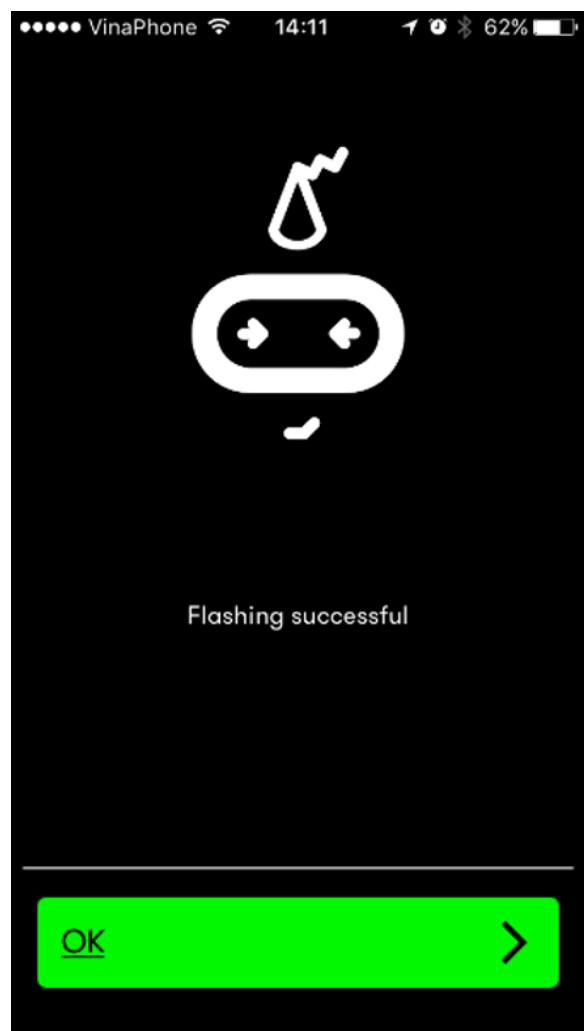
Hình 5.17: Viết chương trình cho MicroBit

Cuối cùng, sau khi chương trình đã hoàn chỉnh, chúng ta có thể nhấn và nút tải về (Bước 3 trên Hình 5.17). Giao diện sau đây (Hình 5.18) sẽ hiện ra, yêu cầu chúng ta chuyển mạch MicroBit sang chế độ Bluetooth. Chúng ta cần phải một lần nữa, nhấn đè A và B, nhấn thả RESET, tiếp tục đè 2 nút A và B cho đến khi kí hiệu Bluetooth xuất hiện trên mạch MicroBit thì thả ra. Cuối cùng, chúng ta có thể quay lại điện thoại, để nhấn nút Continue.



Hình 5.18: Chuyển mạch MicroBit sang chế độ Bluetooth trước khi nhấn nút Continue

Đến đây thì chương trình sẽ tự động làm hết cho chúng ta phần còn lại, kết nối và nạp chương trình. Ở bước này, phiên bản trên iOS thực sự tốt hơn so với phiên bản trên Android. Việc nạp dữ liệu diễn ra nhanh hơn và thao tác cũng bớt rườm rà hơn phiên bản trên Android. Giao diện bên dưới sẽ hiện ra, báo hiệu việc nạp đã thành công. Trong trường hợp muốn soạn lại chương trình, chúng ta chỉ cần nhấn OK và chương trình sẽ tự động quay về giao diện lập trình.



Hình 5.19: Nạp dữ liệu thành công

4 Câu hỏi ôn tập

1. Chọn câu phát biểu đúng:
 - A. Microbit chỉ hỗ trợ lập trình trên điện thoại Android
 - B. MicroBit chỉ hỗ trợ lập trình trên điện thoại iOS
 - C. MicroBit không hỗ trợ lập trình trên thiết bị di động
 - D. Tất cả đều sai
2. Khi nạp chương trình từ điện thoại qua mạch MicroBit, liên kết không dây nào được sử dụng?
 - A. Giao tiếp Radio
 - B. Giao tiếp Bluetooth
 - C. Giao tiếp GSM
 - D. Tất cả đều đúng
3. Làm sao để kích hoạt chế độ Bluetooth của mạch MicroBit?
 - A. Nhấn đè nút A và B, nhấn và thả nút Reset
 - B. Nhấn và thả nút A và B, nhấn đè nút Reset
 - C. Nhấn đè 2 nút A và B, nhấn đè nút Reset
 - D. Nhấn thả 2 nút A và B, nhấn thả nút Reset
4. Việc nạp chương trình giữa Android và iOS, thiết bị nào là nhanh hơn và ổn định hơn?
 - A. Android
 - B. iOS
 - C. Cả 2 thiết bị tương đương
 - D. Không thể xác định
5. Câu phát biểu nào đúng về mã ghép đôi giữa mạch MicroBit và điện thoại?
 - A. Mạch MicroBit phát ra âm thanh báo mã ghép đôi
 - B. Hai thiết bị tự động nhận ra nhau mà không cần ghép đôi
 - C. Mã ghép đôi hiển thị trên 25 đèn của mạch MicroBit
 - D. Tất cả đều sai
6. Câu phát biểu nào là đúng về giao diện lập trình trên điện thoại?
 - A. Một giao diện lập trình hoàn toàn khác so với MakeCode trên máy tính
 - B. Ngôn ngữ lập trình trên điện thoại khác nhiều so với MakeCode trên máy tính
 - C. Môi trường lập trình gần như giống hoàn toàn so với chương trình trên máy tính
 - D. Tất cả đều sai

Đáp án

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



CHU'ONG 6

Tương tác giữ MicroBit và hành vi người dùng

1 Giới thiệu

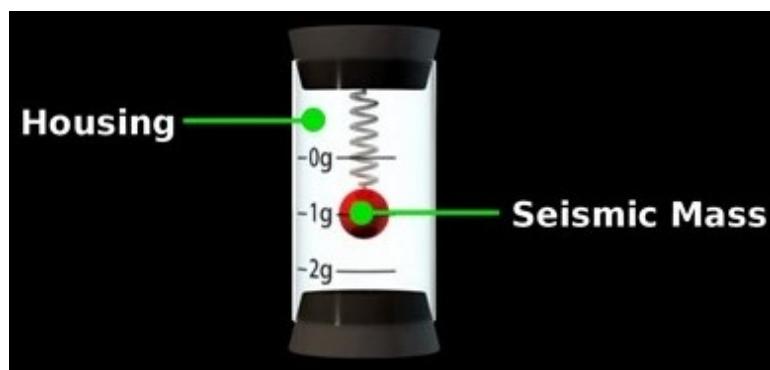
Trong bài hướng dẫn này, chúng ta sẽ tiếp tục với câu lệnh thứ 2 trong nhóm Input. Chúng tôi gọi đây là câu lệnh tương tác với hành vi của người dùng. Sở dĩ có tên gọi như vậy, là vì chúng ta có thể ra lệnh cho mạch MicroBit thực hiện một chức năng nào đó, bằng cách lắc, nghiêng hoặc xoay mạch MicroBit. Hãy tưởng tượng rằng với mạch MicroBit đeo trên cổ tay, mỗi khi muốn xem giờ, chúng ta lắc cổ tay, hoặc một ứng dụng điều khiển robot di chuyển bằng cách nghiêng mạch MicroBit chẳng hạn. Các mục tiêu quan trọng trong bài hướng dẫn này được tóm lược như bên dưới:

- Học sinh hiểu được nguyên lý phát hiện hành vi người dùng dựa trên cảm biến gia tốc.
- Học sinh sử dụng được các câu lệnh để phát hiện hành vi người dùng.
- Học sinh kết hợp các câu lệnh này để tạo ra một ứng dụng đơn giản.

Thông qua việc sử dụng câu lệnh thứ 2 trong nhóm Input, chúng ta sẽ được tiếp cận một phương thức nhận dữ liệu đầu vào hoàn toàn mới. Phương thức này không còn là nhấn nút nhấn một cách truyền thống nữa, mà nó là một dạng tương tác từ hành vi của người dùng đối với mạch MicroBit. Nguyên lý này thực ra đang áp dụng cho các điện thoại thông minh, khi chúng ta xoay điện thoại, màn hình sẽ tự động xoay theo.

2 Nguyên lý phát hiện hành vi của người dùng

Nguyên lý để phát hiện ra hành vi của người dùng trên MicroBit được dựa trên cảm biến gia tốc, có tên tiếng Anh là Accelerometers. Quy tắc hoạt động của cảm biến này có thể được minh họa như Hình 6.1, nó bao gồm 2 phần: khoang chứa hình trụ được gắn liền vào vật thể mà chúng ta cần đo gia tốc, còn quả bóng là vật có thể di chuyển một chiều bên trong khoang chứa. Khi chúng ta di chuyển khoang chứa, quả bóng cũng sẽ di chuyển bên trong khoang chứa, khiến lò xo co hoặc giãn ra. Dựa vào độ co giãn của lò xo, chúng ta có thể đoán biết được lực và gia tốc của chuyển động.



Hình 6.1: Nguyên lý hoạt động của cảm biến gia tốc

Để có thể đo đạc được trong không gian 3 chiều, mô hình trên được nhân lên gấp 3, tương ứng cho 3 trục không gian X, Y và Z. Tuy nhiên, tất cả các nguyên lý phức tạp này đã được MicroBit đơn giản đi rất nhiều bởi các câu lệnh sự kiện, sẽ được trình bày trong phần tiếp theo.//

Lý thuyết về cảm biến gia tốc có thể tương đối phức tạp, tuy nhiên cũng giống như nút nhấn, môi trường lập trình MakeCode đã hỗ trợ cho chúng ta rất nhiều để có thể sử dụng cảm biến này một cách nhanh nhất. Câu lệnh lập trình trong môi trường MakeCode được trình bày chi tiết ở phần tiếp theo.

3 Các câu lệnh phát hiện hành vi

Tất cả các câu lệnh để tương tác với hành vi của người dùng được hỗ trợ sẵn bởi MicroBit được trình bày ở Hình 6.2. Chỉ bằng một câu lệnh **on shake** trong phần Input, đã có rất nhiều hành vi được hỗ trợ sẵn cho chúng ta. Trong phần tiếp theo, một số lựa chọn thông dụng cho câu lệnh **on shake** sẽ được trình bày.



Hình 6.2: Các câu lệnh phát hiện hành vi

3.1 Hành vi on shake

Đây là hành vi căn bản, và cũng là hành vi mặc định của câu lệnh on shake. Khi chúng ta lắc nhẹ mạch MicroBit, sự kiện này sẽ xảy ra. Chúng ta có thể hiện thực một chương trình nhỏ để thử nghiệm sự kiện này như Hình 6.3. Chương trình này hoàn toàn có thể mô phỏng được mà không cần nạp trực tiếp vào mạch MicroBit. Chúng ta chỉ cần rê chuột qua lại trên mạch MicroBit mô phỏng, thì sự kiện on shake sẽ xảy ra.



Hình 6.3: Chương trình thử nghiệm sự kiện on shake

Đối với chương trình trên, khi lắc mạch MicroBit, một trái tim sẽ hiện ra trong 2 giây và tắt đi. Mặc dù sự kiện này vẫn có thể mô phỏng (bằng cách nhấn đèn chuột trái trên mạch MicroBit mô phỏng và rê nó qua lại), chương trình sử dụng câu lệnh tương tác với hành vi của người dùng sẽ thú vị hơn rất nhiều khi nạp chương trình trên mạch MicroBit.

3.2 Các hành vi khác

Trong phần này, ý nghĩa của các hành vi khác sẽ được tóm tắt lại trong Bảng 6.1. Tuy nhiên, để có thể hiểu chức năng của nó cặn kẽ hơn, chúng ta chỉ cần hiện thực một chương trình nhỏ như Hình 6.3 ở trên và chọn sự kiện trong danh sách hỗ trợ.

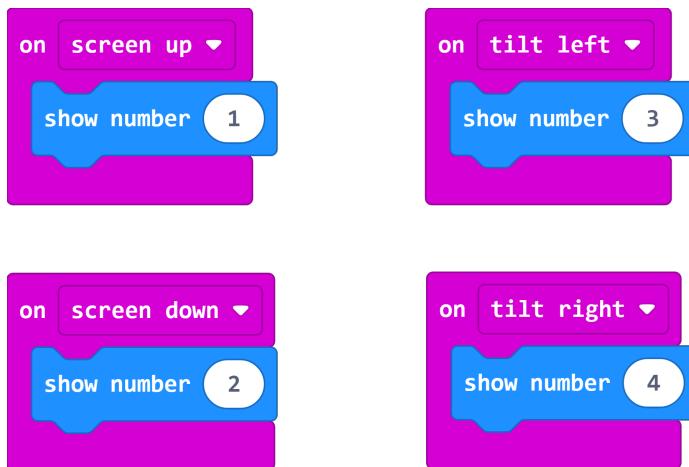
Câu lệnh	Chức năng
	Ở góc bên trái của mạch MicroBit có một Logo, khi Logo này hướng lên trên, sự kiện logo up sẽ xảy ra
	Tương tự như câu lệnh ở trên, sự kiện này xảy ra khi logo hướng xuống dưới
	Trên mạch MicroBit có 1 màn hình gồm 25 đèn hiển thị. Khi màn hình này hướng lên trên, sự kiện screen up xảy ra

		Ngược lại với sự kiện screen up, sự kiện này xảy ra khi màn hình hiển thị úp xuống
		Sự kiện này xảy ra khi chúng ta hơi nghiêng mạch về bên trái
		Sự kiện này xảy ra khi chúng ta hơi nghiêng mạch về bên phải
		Khi thả mạch MicroBit rơi tự do, sự kiện này sẽ được gọi
		Nhóm sự kiện cuối cùng mô phỏng một va chạm. Nó cũng tương đương như việc chúng ta giật mạnh tay mạch MicroBit rồi cho nó dừng lại đột ngột. Các thông số 3g, 6g và 8g tượng trưng cho mức độ dừng lại đột ngột của mạch MicroBit.

Bảng 6.1: Bảng tổng hợp các sự kiện hỗ trợ trên Make-Code

4 Bài tập

1. Học sinh hiện thực chương trình hiển thị số 1, 2, 3 và 4 trên màn hình hiển thị tương ứng với 4 sự kiện màn hình hiển thị hướng lên (screen up), xuống (screen down), nghiêng sang trái (tilt left) và sang phải (tilt right). Chương trình gợi ý cho bài này có thể tham khảo ở Hình 6.4.

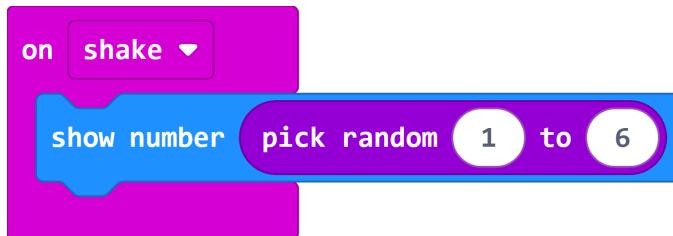


Hình 6.4: Chương trình gợi ý cho Bài 1

2. Học sinh hiện thực một chương trình “Xúc xác điện tử”: Lắc mạch MicroBit sẽ hiển thị ngẫu nhiên 1 con số từ 1 đến 6. Học sinh sử dụng câu lệnh lấy số ngẫu nhiên ở Hình 6.5. Chương trình gợi ý cho bài này được trình bày ở Hình 6.6.



Hình 6.5: Lệnh lấy giá trị ngẫu nhiên



Hình 6.6: Đáp án cho chương trình "Xúc xác điện tử"

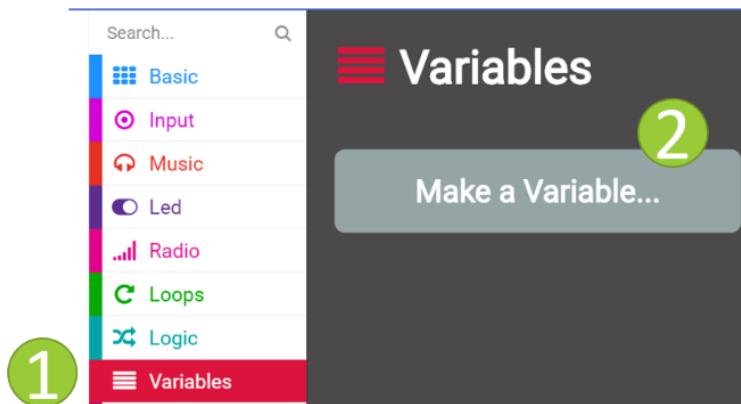
3. Học sinh hãy cải tiến chương trình “Xúc xác điện tử”, thay vì hiển thị các con số từ 1 đến 6, hãy hiển thị một dấu chấm, hai dấu chấm, ... trên màn hình như 1 xúc xác bình thường.

Gợi ý:

- Tạo một biến số lưu giá trị số ngẫu nhiên
- Sử dụng thêm câu lệnh điều kiện trong mục Logic

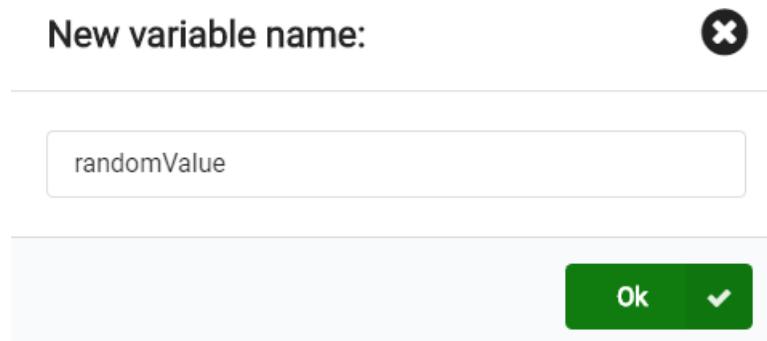
Các bước chi tiết để hiện thực bài tập này như sau:

Bước 1: Vào mục Variables (1), nhấn vào Make a Variable (2) như Hình 6.7



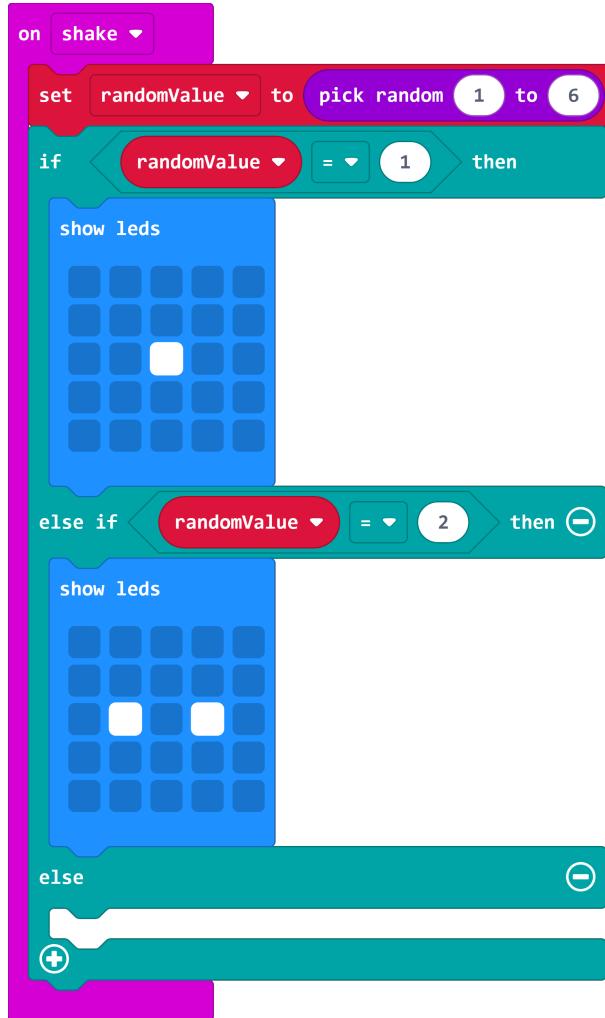
Hình 6.7: Tạo một biến số cho chương trình

Bước 2: Đặt tên cho biến (ví dụ như randomValue), rồi nhấn OK.



Hình 6.8: Đặt tên cho biến là randomValue

Bước 3: Hiện thực chương trình bằng cách sử dụng câu lệnh if – else trong mục Logic



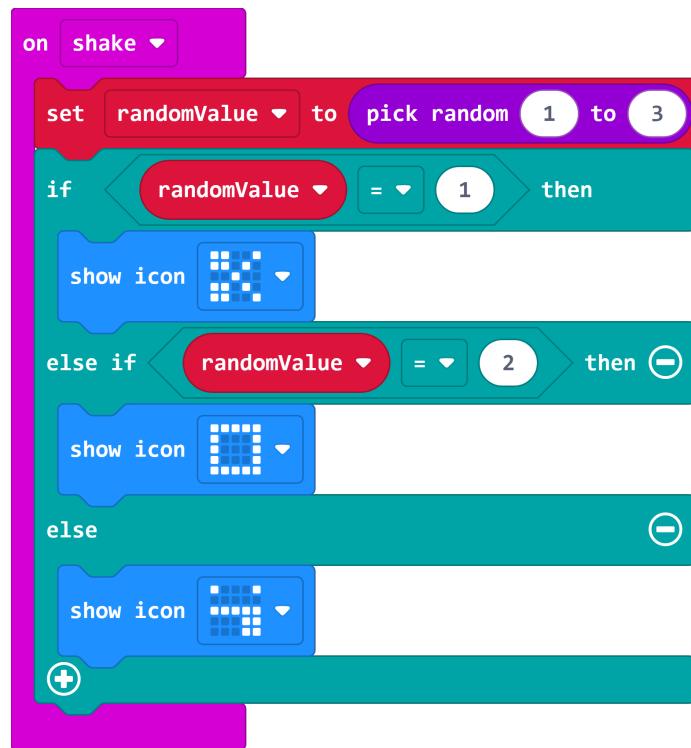
Hình 6.9: Đáp án gợi ý cho chương trình "Xúc xắc điện tử"

4. Học sinh hiện thực trò chơi “Oẳn tù tì” điện tử bằng cách sử dụng mạch MicroBit. Mạch sẽ hiển thị các biểu tượng ngẫu nhiên 1 trong 3 hình ảnh Búa – Dao - Kéo.

Gợi ý:

- Lấy ngẫu nhiên từ 1 đến 3: 1 – Búa, 2- Bao, 3 – Kéo
- Sử dụng câu lệnh show icon

Đáp án:



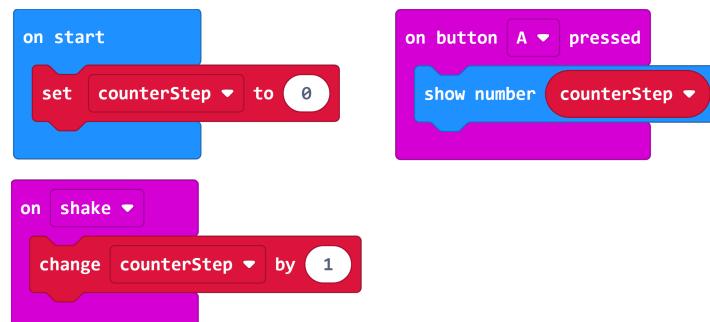
Hình 6.10: Đáp án gợi ý cho chương trình "Oǎn tù tì"

5. Học sinh hiện thực một chương trình đếm số bước chân khi di chuyển. Mạch MicroBit sẽ được gắn vào chân (không để dưới đế giày!!!!). Bình thường, mạch không hiện thị gì cả, chỉ khi nhấn vào nút A, mạch sẽ hiển thị lên số bước chân đã di chuyển.

Gợi ý:

- Tạo một biến để lưu trữ số bước chân (counterStep)
- Tăng biến này lên 1 trong sự kiện on shake
- Hiển thị ra khi nhấn nút A

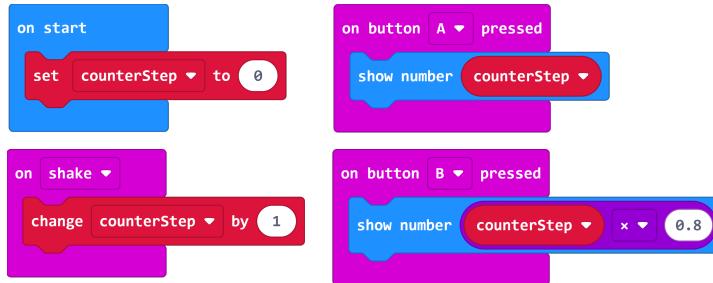
Đáp án:



Hình 6.11: Chương trình đếm số bước chân di chuyển

6. Học sinh cải thiện bài tập trình, khi nhấn vào nút B, mạch MicroBit sẽ hiển thị quãng đường đã di chuyển.

Gợi ý: Sử dụng số bước chân, nhân với khoảng cách gần đúng của một bước chân (khoảng 0.8m). Chương trình gợi ý cho bài này được trình bày ở Hình 6.12.



Hình 6.12: Chương trình đếm số bước chân và quãng đường đã di chuyển

5 Câu hỏi ôn tập

1. Câu lệnh về hành vi của người dùng được nằm trong mục nào của MakeCode?
 - A. Basic
 - B. Input
 - C. Music
 - D. LED
2. Sự kiện nào để phát hiện hành vi rung lắc từ phía người dùng?
 - A. on button A pressed
 - B. on button B pressed
 - C. on shake
 - D. on vibaration
3. Sự kiện nào để phát hiện hành vi nghiêng sang bên trái?
 - A. on left
 - B. on right
 - C. tilt left
 - D. tilt right
4. Sự kiện nào để phát hiện hành vi nghiêng sang bên phải?
 - A. on left
 - B. on right
 - C. on tilt left
 - D. on tilt right
5. Sự kiện nào để phát hiện hành vi màn hình hướng lên?
 - A. on logo up
 - B. on logo down
 - C. on screen up
 - D. on screen down

6. Sự kiện nào để phát hiện hành vi màn hình hướng xuống?
 - A. on logo up
 - B. on logo down
 - C. on screen up
 - D. on screen down
7. Sự kiện nào để phát hiện hành vi logo hướng lên?
 - A. on logo up
 - B. on logo down
 - C. on screen up
 - D. on screen down
8. Sự kiện nào để phát hiện hành vi màn hình hướng xuống?
 - A. on logo up
 - B. on logo down
 - C. on screen up
 - D. on screen down
9. Để hiện thực ứng dụng đếm số bước chân của người đi bộ, hành vi nào là phù hợp nhất?
 - A. on shake
 - B. on logo up
 - C. on screen up
 - D. on free fall
10. Để phát hiện hành vi có tai nạn giao thông, hành vi nào là phù hợp nhất?
 - A. on 3g
 - B. on 6g
 - C. on 8g
 - D. Tất cả các hành vi trên
11. Phát biểu nào là đúng về khôi lệnh hành vi?
 - A. Khôi lệnh hành vi phải được hiện thực trong on start
 - B. Khôi lệnh hành vi phải được hiện thực trong forever
 - C. Khôi lệnh hành vi là một khôi đọc lặp, được hiện thực riêng biệt
 - D. Tất cả đều đúng
12. Các câu lệnh hành vi trên MicroBit được hiện thực dựa vào cảm biến gì?
 - A. Cảm biến gia tốc
 - B. Cảm biến nhiệt độ
 - C. Cảm biến ánh sáng
 - D. Cảm biến la bàn

Đáp án

1. B 2. C 3. C 4. D 5. C 6. D 7. A 8. B 9. A 10. D 11. C 12. A

CHƯƠNG 7

Cảm biến trên MicroBit



1 Giới thiệu

Một trong những đặc điểm quan trọng giúp cho mạch MicroBit có thể triển khai dễ dàng trong giáo dục, đặc biệt là với lứa tuổi nhỏ, đó là khả năng tích hợp sẵn các công nghệ hiện đại. Cụ thể, trong bài này, những cảm biến trên mạch MicroBit có sẵn trên mạch MicroBit như cường độ ánh sáng, nhiệt độ và la bàn, sẽ được giới thiệu. Việc tích hợp sẵn những thiết bị cảm biến ngay trên bo mạch, sẽ giúp cho kết nối của chúng bền hơn và tránh những lỗi do việc kết nối không ổn định, vốn rất khó khắc phục ở cấp độ học sinh. Các mục tiêu cụ thể của bài hướng dẫn này như sau:

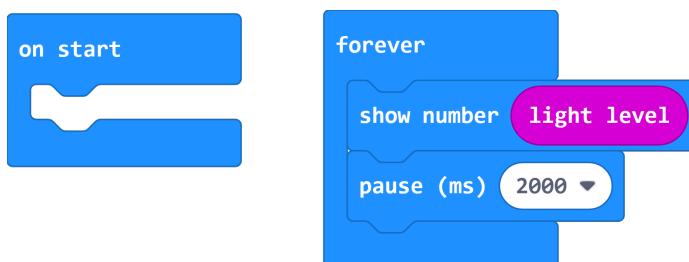
- Học sinh hiểu được cảm biến là gì.
- Học sinh sử dụng được các câu lệnh liên quan đến các cảm biến.
- Học sinh kết hợp các câu lệnh này để tạo ra một ứng dụng đơn giản.

2 Cảm biến là gì?

Cảm biến là thiết bị điện tử cảm nhận những trạng thái hay quá trình vật lý hay hóa học ở môi trường cần khảo sát, và biến đổi thành tín hiệu điện để thu thập thông tin về trạng thái hay quá trình đó. Đơn giản hơn, cảm biến điện tử giống như các giác quan của con người, giúp hệ thống điện tử có thể nhận biết được trạng thái của môi trường xung quanh. Hiện tại, trên bo mạch MicroBit hỗ trợ 3 loại cảm biến cơ bản là cảm biến ánh sáng, cảm biến nhiệt độ và cảm biến la bàn. Thông tin chi tiết của mỗi loại cảm biến được trình bày ở phần tiếp theo.

2.1 Cảm biến ánh sáng

Cảm biến ánh sáng trong MicroBit được hỗ trợ sẵn bởi biến **light level**, nằm trong mục Input. Một chương trình nhỏ để kiểm tra giá trị của biến này được minh họa như Hình 7.1



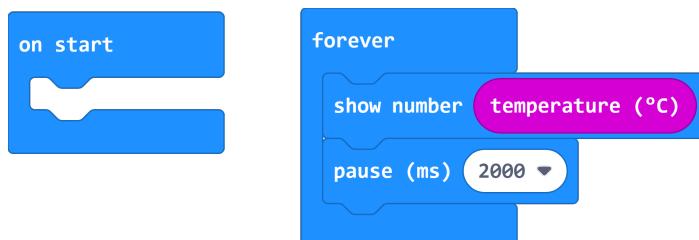
Hình 7.1: Chương trình kiểm tra giá trị cảm biến ánh sáng

Thực ra, giá trị của cảm biến ánh sáng được thực hiện bởi chính màn hình hiển thị. Mỗi một bóng đèn trên màn hình hiển thị có 2 chu kỳ hoạt động. Ở chu kỳ thứ nhất, nó làm nhiệm vụ phát sáng. Ở chu kỳ còn lại, nó trở thành 1 cảm biến nhận dạng độ sáng. Do đó, việc xuất giá trị cảm biến này ra chính màn hình hiển thị, làm cho

giá trị light level không ổn định do nó bị ảnh hưởng bởi các bóng đèn xung quanh. Ở bài sau, học sinh sẽ được hướng dẫn để gửi giá trị cảm biến này qua một mạch MicroBit khác. Lúc đó, giá trị của cảm biến ánh sáng mới trở nên trung thực và chính xác. Hiện tại ở bài này, chúng ta thêm một câu lệnh đợi ở phía cuối, để đảm bảo rằng các đèn sẽ tắt hết sau 2 giây, sau đó mới đọc lại giá trị cảm biến ánh sáng.

2.2 Cảm biến nhiệt độ

Tương tự như cảm biến ánh sáng, cảm biến nhiệt độ được hỗ trợ sẵn trên MicroBit thông qua biến **temperature**. Chúng ta có thể hiện thực một chương trình đơn giản như Hình 7.2 để kiểm tra giá trị của biến này. Khác với cảm biến ánh sáng, giá trị của cảm biến nhiệt độ sẽ rất ổn định và phản ánh nhiệt độ của bo mạch MicroBit. Do đó, nhiệt độ nhận được thông thường sẽ cao hơn nhiệt độ môi trường khoảng $2 - 3^{\circ}\text{C}$.



Hình 7.2: Chương trình kiểm tra cảm biến nhiệt độ

2.3 Cảm biến la bàn

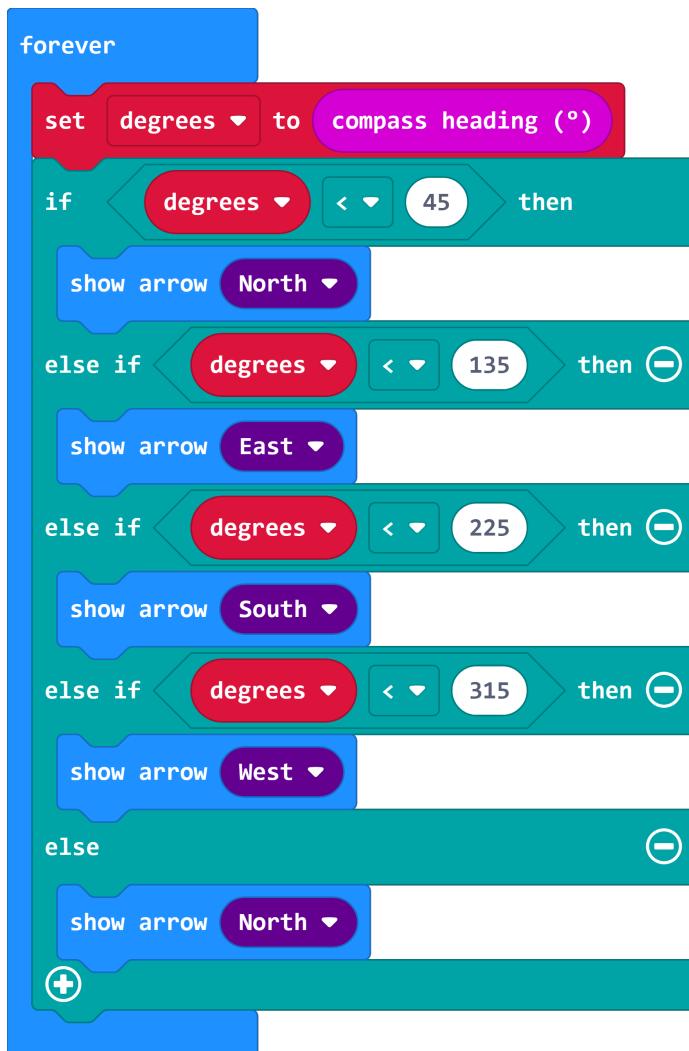
Tương tự như 2 cảm biến đã trình bày ở trên, chúng ta có thể hiện thực nhanh một chương trình để kiểm tra cảm biến la bàn như Hình 7.3, bằng việc thay thế tham số trong câu lệnh **show number** bằng khối **compas heading** trong mục Input.



Hình 7.3: Chương trình kiểm tra cảm biến la bàn

Tuy nhiên, khác với hai cảm biến trên. Khi sử dụng cảm biến la bàn, MicroBit yêu cầu chúng ta phải chỉnh lại thước đo của nó. Một thông báo hiện lên, yêu cầu chúng ta xoay mạch MicroBit để tắt cả các đèn đều phải sáng. Sau khi hoàn thành bước này, chương trình của chúng ta mới được phép chạy. Sau khi việc cân chỉnh hoàn thành, giá trị của la bàn sẽ nằm từ 0 cho đến 359. Nhỏ hơn 45° là hướng Bắc, từ 45 đến 135 là hướng Tây, từ 135 đến 225 là hướng Nam, 225 đến 315 là hướng Đông. Một phần giá trị từ 315 đến 0 vẫn là hướng Bắc.

Bài tập: Học sinh hiện thực một chương trình để vẽ ra bốn hướng Đông Tây Nam Bắc bằng cảm biến la bàn. Đáp án gợi ý cho bài tập này được minh họa ở Hình 7.4. Khối lệnh **if else if** cũng như các **phép so sánh** được lấy từ khối **Logic**.



Hình 7.4: Đáp án gợi ý về cảm biến la bàn

2.4 Cảm biến góc xoay

Cảm biến góc xoay trên mạch MicroBit gồm có 3 trục, là x, y và z. Đây chính là bản chất của các câu lệnh hành vi trên MicroBit mà chúng ta đã làm ở các bài trước. Trong trường hợp chúng ta muốn tự định nghĩa một hành vi mới, không nằm trong danh sách có sẵn của MicroBit, chúng ta sẽ cần phải truy cập tới thông tin về góc xoay này. Ví dụ: chúng ta muốn phát hiện hành vi của một người ngồi lên ghế ô tô, chúng ta có thể làm như sau: Mạch MicroBit được đặt dưới ghế ngồi, thường là bằng nệm mềm. Khi có người ngồi lên, vị trí của mạch MicroBit sẽ bị thay đổi, từ đó tạo ra các giá trị về góc xoay trên 3 trục x, y và z. Nếu như tổng giá trị tuyệt đối của 3 góc xoay này lớn hơn một số nhất định, chúng ta xem như có một người đang ngồi lên ghế. Khi người đó ra khỏi ghế ngồi, tổng giá trị tuyệt đối này chưa hẳn đã về 0, nhưng nó sẽ là một con số nhỏ hơn.

Chương trình kiểm tra giá trị cảm biến góc xoay cũng tương tự như 2 phần ở trên. Câu lệnh để lấy giá trị cảm biến góc xoay là **acceleration**. Đây là một câu lệnh tùy chọn, để có thể truy xuất giá trị góc xoay ở 3 trục. Tuy nhiên, chúng ta nên kiểm tra giá trị của từng trục để định vị chiều ngang (trục x) là như thế nào. Chương trình gợi ý như sau:



Hình 7.5: Cảm biến góc xoay

Khi thử nghiệm giá trị của cảm biến góc xoay, hãy lưu ý về giá trị âm dương của nó. Ví dụ như trục Z, khi mạch di chuyển lên nó có giá trị dương, khi mạch di chuyển xuống, nó có giá trị âm.

3 Bài tập

Học sinh tự định nghĩa một hành vi và hiện thực nó bằng cách sử dụng các giá trị cảm biến góc xoay từ 3 trục. Lưu ý là trong các ứng dụng thực tế, chúng ta sẽ xem xét độ biến thiên giá trị của cảm biến thay vì giá trị trực tiếp của nó. Học sinh hãy sử dụng thêm câu lệnh **absolute of** trong mục **Math** để xử lý yêu cầu này. Ví dụ, khi góc nghiêng của mạch MicroBit gắn lên một thân cây thay đổi hơn 20 giá trị, chúng ta có thể xem là cây đang bị nghiêng.

4 Câu hỏi ôn tập

1. Những cảm biến nào sau đây có trên micro:bit ?
 - A. Áp suất, lưu lượng, trọng lượng, vận tốc
 - B. Ánh sáng, nhiệt độ, độ ẩm, vận tốc.
 - C. Ánh sáng, la bàn, nhiệt độ, gia tốc.
 - D. Ánh sáng, nhiệt độ, gia tốc, vận tốc.
2. Câu lệnh on shake dựa trên cảm biến nào trên micro:bit ?
 - A. Vận tốc
 - B. Nhiệt độ
 - C. Ánh sáng
 - D. Gia tốc
3. Tập hợp các câu lệnh nào sau đây dựa trên cảm biến gia tốc?
 - A. show led, show number, show string.
 - B. shake, logo up, logo down, screen up, screen down.
 - C. A và B đúng
 - D. A và B sai

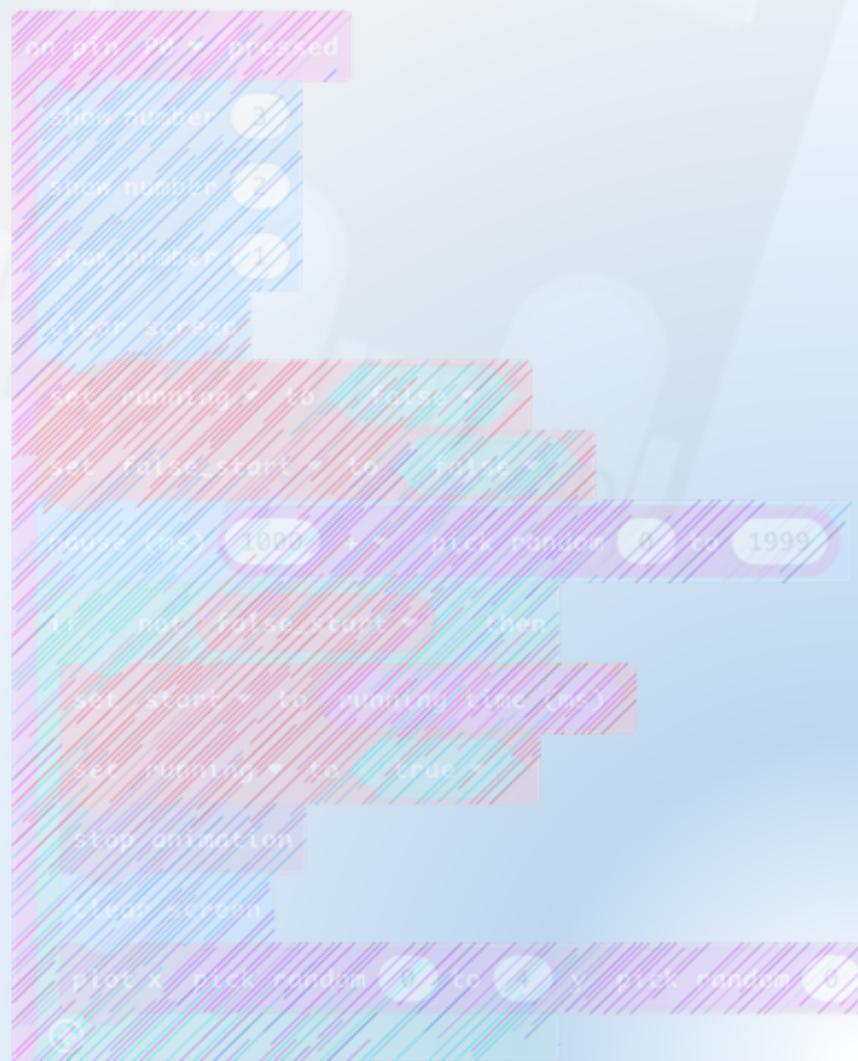
4. Câu lệnh nào sau đây có chức năng lấy dữ liệu cảm biến nhiệt độ?
 - A. show led
 - B. light level
 - C. temperature (°C)
 - D. compass heading (°)
5. Câu lệnh nào sau đây có chức năng lấy dữ liệu cảm biến la bàn?
 - A. show led
 - B. light level
 - C. temperature (°C)
 - D. compass heading (°)
6. Câu lệnh nào sau đây có chức năng lấy dữ liệu cảm biến ánh sáng?
 - A. show led
 - B. light level
 - C. temperature (°C)
 - D. compass heading (°)
7. Việc đầu tiên khi sử dụng cảm biến là bàn là gì?
 - A. Nghiêng mạch micro:bit cho đèn khi các led được sáng hết.
 - B. Đặt mạch micro:bit ở một nơi bằng phẳng.
 - C. Đặt mạch micro:bit trong bóng tối.
 - D. Tắt hết các led trước khi đọc cảm biến để tăng độ chính xác.
8. Chú ý nào sau đây đúng khi sử dụng cảm biến ánh sáng?
 - A. Nghiêng mạch micro:bit cho đèn khi các led được sáng hết.
 - B. Đặt mạch micro:bit ở một nơi bằng phẳng.
 - C. Đặt mạch micro:bit trong bóng tối.
 - D. Tắt hết các led trước khi đọc cảm biến để tăng độ chính xác.

Đáp án

1. C 2. D 3. B 4. C 5. D 6. B 7. A 8. D

CHƯƠNG 8

Gửi dữ liệu không dây giữa các mạch MicroBit



1 Giới thiệu

Trong thời đại kĩ thuật số ngày càng phát triển, chúng ta đã dần quen với khái niệm về cách mạng 4.0 hay Internet vạn vật. Trong thế giới mới, các thiết bị giao tiếp không dây với nhau, trao đổi dữ liệu và hình thành nên một thế giới mới, thế giới kết nối vạn vật. Thiết kế để phổ cập rộng rãi các công nghệ hiện tại, mạch MicroBit cũng trang bị công nghệ giao tiếp không dây.

Tính năng giao tiếp không dây của MicroBit có thể làm nhiều người bất ngờ, khi nó hỗ trợ đến 2 tiêu chuẩn giao tiếp khác nhau: giao tiếp qua Radio và giao tiếp qua Bluetooth. Nếu như tiêu chuẩn thứ nhất, được xem là tiêu chuẩn cho các thiết bị trong nhà thông minh, thì tiêu chuẩn thứ 2, là sự chuẩn bị cho các ứng dụng thời đại mới, sử dụng Bluetooth 4.0 cho các ứng dụng điều khiển thiên về bảo mật và kết nối ổn định.

Đối với các dự án vừa và nhỏ, khả năng giao tiếp không dây tạo cơ hội cho chúng ta mở rộng chức năng của dự án vô cùng dễ dàng. Nếu như với cách tiếp cận truyền thống, một bo mạch chính sẽ cố gắng hiện thực tất cả các chức năng của hệ thống. Giờ đây, với khả năng giao tiếp không dây, một mạch MicroBit chỉ cần đảm nhiệm một vài chức năng. Khi nó thực hiện xong, nó gửi tín hiệu không dây đến cho mạch MicroBit thứ 2, để mạch này làm tiếp các chức năng khác. Người dùng sẽ tiết kiệm rất nhiều việc kết nối dây giữa nhiều thiết bị, điều này rất thường xuyên gây ra ảnh hưởng đến hiệu suất của hệ thống, vì nó ảnh hưởng trực tiếp đến bộ cung cấp nguồn.

Ở mức độ đơn giản, trong hướng dẫn này, chúng tôi sẽ tập trung vào công nghệ giao tiếp bằng sóng Radio. Các mục tiêu chính trong bài hướng dẫn này được liệt kê như sau:

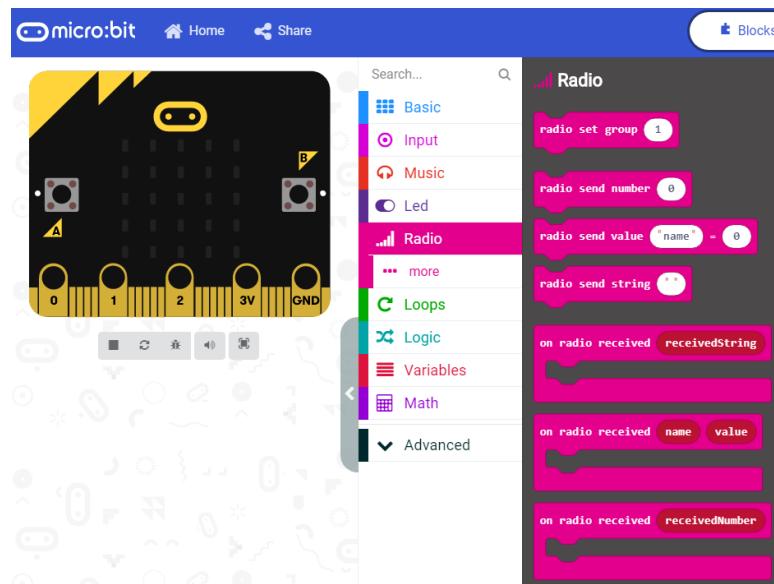
- Học sinh hiểu được nguyên lý gửi nhận dữ liệu không dây.
- Học sinh sử dụng được các câu lệnh gửi nhận dữ liệu không dây.
- Học sinh phối hợp được nhiều câu lệnh để tạo thành chương trình mới.

2 Nhóm lệnh Radio

Trên môi trường lập trình MakeCode, các câu lệnh giao tiếp không dây nằm trong nhóm Radio, như trình bày ở Hình 8.1.

Một nguyên lý cơ bản của việc giao tiếp không dây trên MicroBit là các mạch phải có cùng nhóm với nhau. Mặc định, nhóm của các mạch MicroBit là 1. Tuy nhiên để đảm bảo an toàn, chúng ta sẽ chỉnh lại nhóm của chúng theo con số mà chúng ta quy định.

Hiển nhiên, để có thể giao tiếp được giữa 2 mạch MicroBit, chúng ta cần hiện thực 2 chương trình, một cho nốt truyền và chương trình còn lại cho nốt nhận. Phần tiếp theo sẽ trình bày chi tiết việc hiện thực chương trình cho từng nốt.

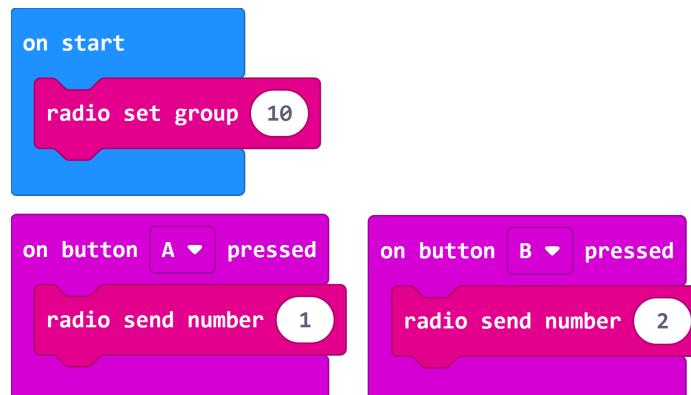


Hình 8.1: Các câu lệnh thuộc nhóm Radio để truyền nhận dữ liệu không dây

3 Chương trình nốt gửi

Để đơn giản, khi nhấn phím A, chúng ta sẽ gửi một con số cho nốt nhận, ví dụ số 1. Nhấn nút B, chúng ta sẽ gửi số 2. Chúng ta sẽ chọn nhóm cho 2 nốt là 10. Tuy nhiên, học sinh có thể chọn bất kì một con số nào đó cho hệ thống.

Việc chỉ định nhóm cho hệ thống sẽ được hiện thực trong phần on start. Và việc gửi dữ liệu sẽ được hiện thực bằng sự kiện nhấn nút A hay B (được trình bày ở Chương 4). Chương trình gợi ý của chúng ta như sau:

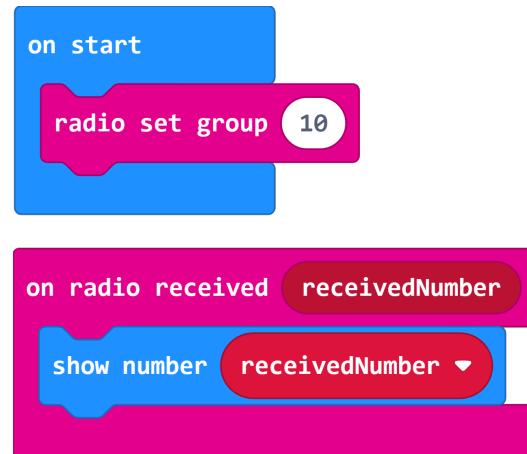


Hình 8.2: Chương trình cho nốt truyền

4 Chương trình cho nốt nhận

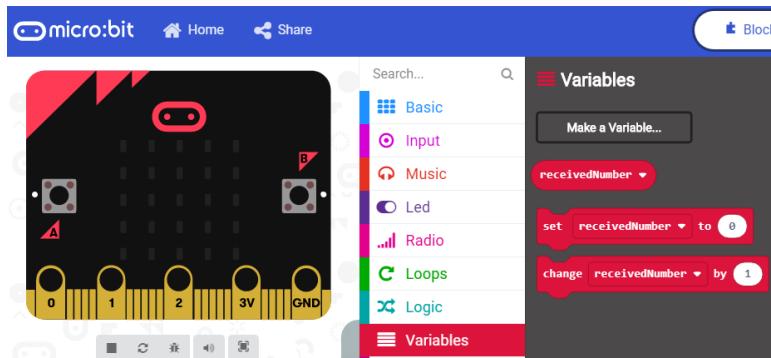
Cũng tương tự như nốt truyền, trong hàm on start, chúng ta cũng chỉ định nhóm cho nốt nhận là 10. Tiếp sau đó, chúng ta sẽ dùng câu lệnh **on radio received**. Câu lệnh này là câu lệnh sự kiện, cũng giống với nút nhấn A hay B, trong trường hợp

này, khi nhận được một dữ liệu gì đó, sự kiện này sẽ xảy ra. Chương trình của chúng ta như sau:



Hình 8.3: Chương trình cho nốt nhận

Trong chương trình trên, khi nhận được bất kì dữ liệu số nào, chúng ta sẽ hiện thị số đó ra màn hình bằng câu lệnh show number. Kết quả nhận được sẽ được lưu trữ mặc định trong biến receivedNumber, nằm trong mục Variable như Hình 8.4.



Hình 8.4: Biến receivedNumber trong mục Variable

Bây giờ, học sinh có thể kết hợp chương trình của 2 mạch MicroBit để thấy được việc gửi dữ liệu không dây.

5 Bài tập trên lớp

Học sinh hãy hiện thực nhiều câu lệnh hơn từ nốt gửi, bằng cách sử dụng các câu lệnh liên quan đến hành vi của người dùng, ví dụ như nghiêng qua trái, nghiêng qua phải. Bên nốt nhận, thay vì chỉ hiển thị các con số, học sinh có thể hiển thị các dấu mũi tên hoặc các icon khác.

6 Câu hỏi ôn tập

1. Nhóm lệnh nào sau đây sử dụng cho việc truyền dữ liệu không dây?
 - A. radio
 - B. input
 - C. music
 - D. led
2. Để ít nhất hai mạch micro:bit có thể kết nối với nhau thì cần phải?
 - A. Để các mạch sát vào nhau để chúng tự đồng bộ.
 - B. Nhấn đồng loạt nút A để các mạch tự kết nối với nhau.
 - C. Dùng câu lệnh radio set group và cho các mạch có số group giống nhau.
 - D. Tất cả đều sai.
3. Để gửi một con số đến mạch micro:bit khác ta dùng lệnh:
 - A. radio send number
 - B. on radio received number
 - C. radio send string
 - D. on radio received receivedString
4. Để nhận một con số từ mạch micro:bit chúng ta dùng lệnh:
 - A. radio send number
 - B. on radio received number
 - C. radio send string
 - D. on radio received receivedString
5. Để gửi một chuỗi ký tự chúng ta dùng lệnh:
 - A. radio send number
 - B. on radio received number
 - C. radio send string
 - D. on radio received receivedString
6. Để nhận một chuỗi ký tự chúng ta dùng lệnh:
 - A. radio send number
 - B. on radio received number
 - C. radio send string
 - D. on radio received receivedString
7. Để gửi một biến vừa có tên vừa có giá trị chúng ta dùng lệnh:
 - A. radio send value (name) = (value)
 - B. on radio received name value
 - C. radio send string
 - D. on radio received receivedString
8. Để nhận một biến vừa có tên vừa có giá trị chúng ta dùng lệnh:
 - A. radio send value (name) = (value)
 - B. on radio received name value
 - C. radio send string
 - D. on radio received receivedString

Đáp án

1. A 2. C 3. A 4. B 5. C 6. D 7. A 8. B

CHƯƠNG 9

Câu lệnh điều kiện IF



1 Giới thiệu

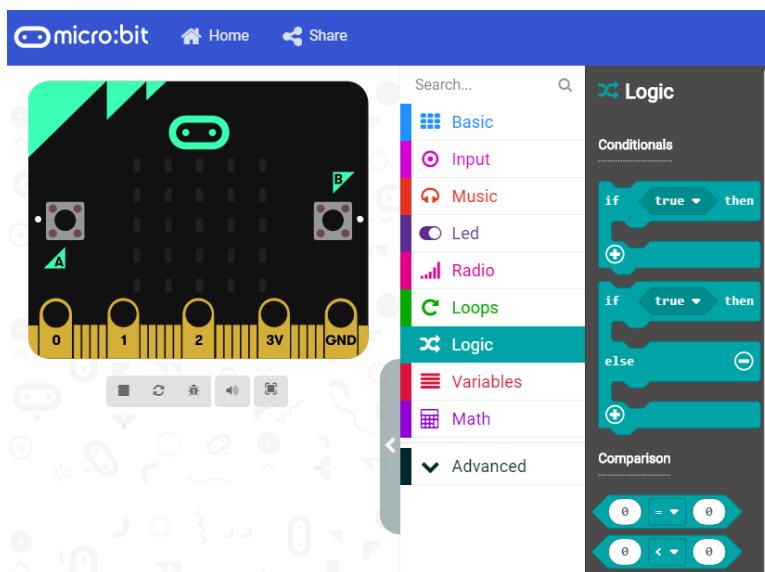
Trong bất kì một ngôn ngữ lập trình nào, cấu trúc điều kiện IF, mà chúng ta hay quen gọi là câu lệnh "Nếu...thì...nữa", luôn đóng một vai trò quan trọng. Nó là câu lệnh đặc trưng cho quá trình xử lý trong một chương trình. Tầm quan trọng của câu lệnh IF lớn đến mức, chúng ta có thể hiện thực bất kì một bài toán nào trên bo mạch MicroBit, chỉ bằng câu lệnh IF mà thôi. Chúng ta không cần câu lệnh lặp vì đơn giản, chúng ta đã có sẵn một khối lệnh mang ý nghĩa lặp, là **forever**. Do đó, việc nắm vững nguyên lý và sử dụng hiệu quả là điều vô cùng quan trọng.

Một điều đặc trưng, là trong bất kì ngôn ngữ lập trình nào, cấu trúc IF sẽ luôn có 3 phiên bản, bao gồm câu lệnh IF đơn giản, câu lệnh IF ELSE và câu lệnh IF ELSE IF ELSE. Bên cạnh việc trình bày cấu trúc các lệnh này, các mục tiêu khác trong bài này được tóm tắt như sau:

- Học sinh hiểu được cấu trúc điều kiện trên MakeCode
- Học sinh sử dụng được các câu lệnh điều kiện trên MakeCode
- Học sinh phối hợp được nhiều câu lệnh trong chương trình

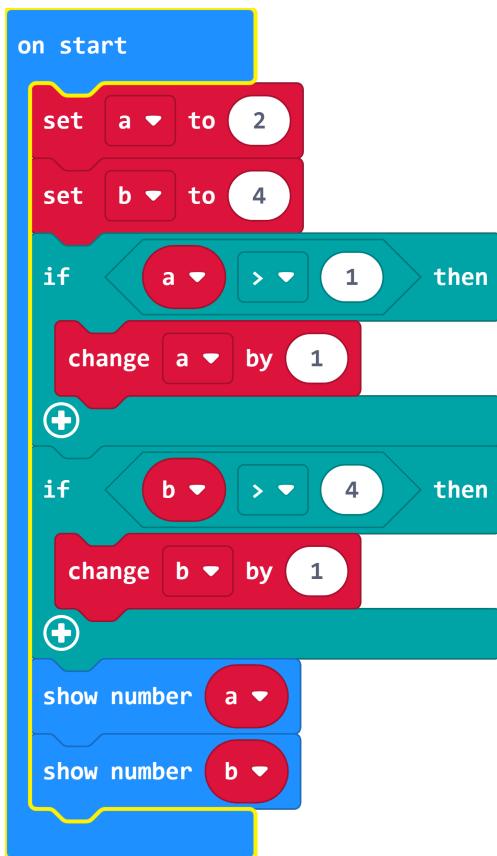
2 Cấu trúc IF

Trước tiên, các câu lệnh điều kiện được nằm trong mục **Logic** của giao diện lập trình MakeCode, như trình bày ở Hình 9.1.



Hình 9.1: Các câu lệnh thuộc nhóm Logic

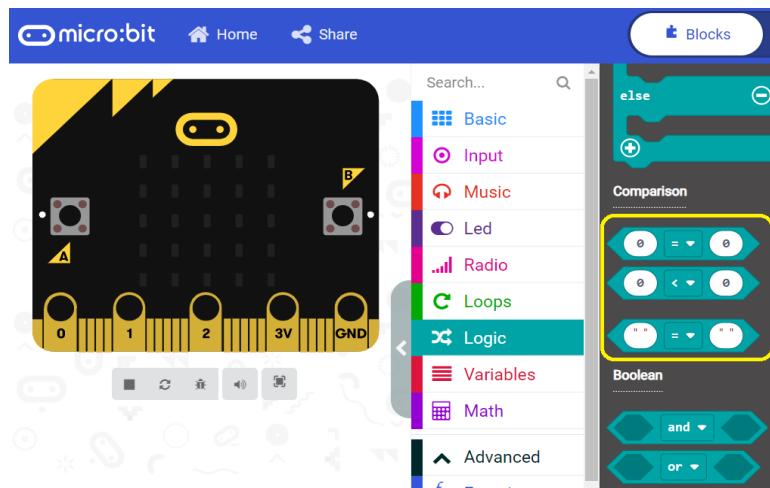
Cấu trúc điều kiện IF chính là khối lệnh đầu tiên trong nhóm Logic. Nếu điều kiện đúng thì các câu lệnh bên trong nó mới được thực hiện. Ngược lại, nếu điều kiện sai, các câu lệnh sẽ không được thực thi. Chúng ta cũng xem một ví dụ sau đây:



Hình 9.2: Ví dụ về câu lệnh điều kiện IF

Trong chương trình ở Hình 9.2, là một ví dụ đơn giản với 2 biến số a và b. Sau đoạn chương trình này, biến a là 3 và b vẫn không đổi, là 4. Sở dĩ kết quả như vậy, là vì câu lệnh IF đầu tiên được thực thi (điều kiện $a > 1$ là đúng), còn câu lệnh IF thứ 2, không được thực thi (điều kiện $b > 4$ là sai).

Trong việc hiện thực câu lệnh điều kiện, một điều quan trọng nhất là xây dựng các câu lệnh so sánh (có hình thoi). MicroBit hoàn toàn hỗ trợ sẵn các điều kiện này, nằm ngay bên dưới các câu lệnh điều kiện IF, ở mục **Comparison**, như trình bày ở Hình 9.3.

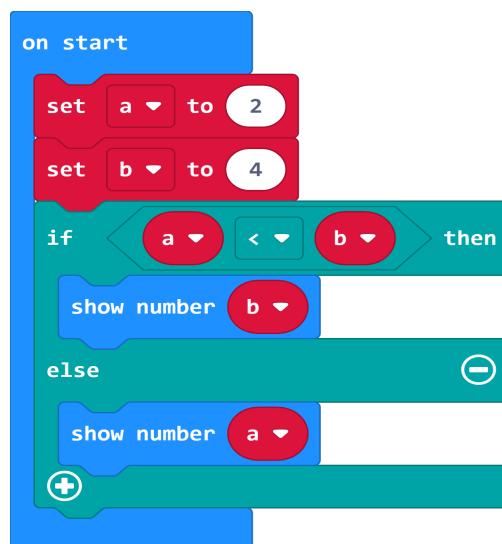


Hình 9.3: Các câu lệnh so sánh

Một lưu ý quan trọng, là 3 câu lệnh so sánh có 2 câu lệnh đầu là dành cho số, câu lệnh thứ 3 là dành cho chuỗi (với kí tự nhận diện " " đặc trưng cho dữ liệu kiểu chuỗi). Trong trường hợp muốn kết hợp nhiều điều kiện với nhau, chúng ta sẽ xài thêm toán tử **and** hoặc **or**, trong mục **Boolean**, nằm ngay bên dưới phần Comparison.

3 Cấu trúc IF ELSE

Đây là phiên bản thứ 2 của cấu trúc điều kiện. Câu lệnh này được xếp thứ 2, ngay sau câu lệnh IF trình bày ở trên. Khi điều kiện IF là đúng, chương trình sẽ thực hiện những câu lệnh bên trong nó và **không thực thi các câu lệnh trong phần ELSE**. Ngược lại, các câu lệnh trong phần ELSE sẽ được thực thi. Một ví dụ để hiểu hơn về nguyên lý của câu lệnh này được trình bày như hình bên dưới.

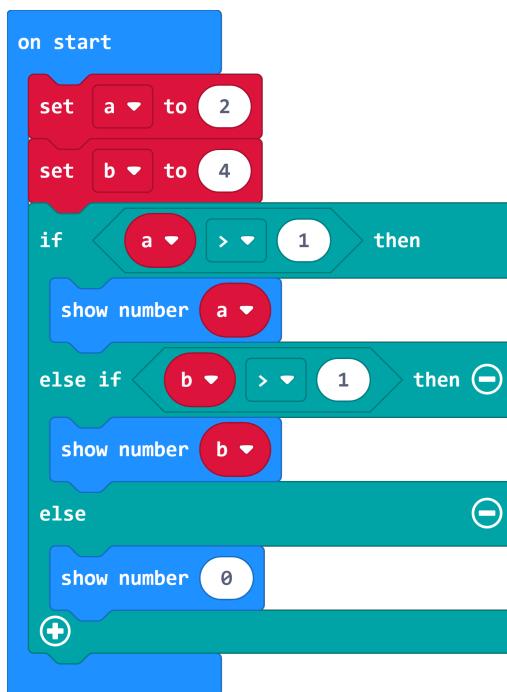


Hình 9.4: Câu lệnh điều kiện IF ELSE

Có thể dễ dàng nhận ra rằng, chương trình ở Hình 9.4 sẽ hiển thị số lớn hơn ra màn hình của mạch MicroBit. Đây là câu lệnh được sử dụng tương đối nhiều bởi tính thân thiện và dễ diễn giải bằng ngôn ngữ lập trình. Tuy nhiên, một lưu ý quan trọng mà ngược học cần phải nhớ: Khi phần IF đã được thực thi thì phần ELSE sẽ không được thực thi. Một cách khác, chỉ 1 trong 2 phần này được thực thi mà thôi.

4 Câu lệnh IF ELSE IF ELSE

Chúng ta cũng có thể gọi đây là câu lệnh nhiều điều kiện. Nó là biến thể của câu lệnh IF ELSE bên trên khi chúng ta mở rộng nó bằng cách nhấn vào dấu cộng (+). Nguyên lý của câu lệnh này là lần lượt kiểm tra các điều kiện, theo thứ tự từ trên xuống. Khi gặp một điều kiện đúng, các câu lệnh bên trong nó sẽ được thực thi. Một điều mà chúng ta hay nhầm lẫn, là sau khi gặp điều kiện đúng đầu tiên, tất cả các điều kiện còn lại sẽ không còn được kiểm tra và thực thi nữa. Một ví dụ minh họa cho câu lệnh này được trình bày ở hình bên dưới.

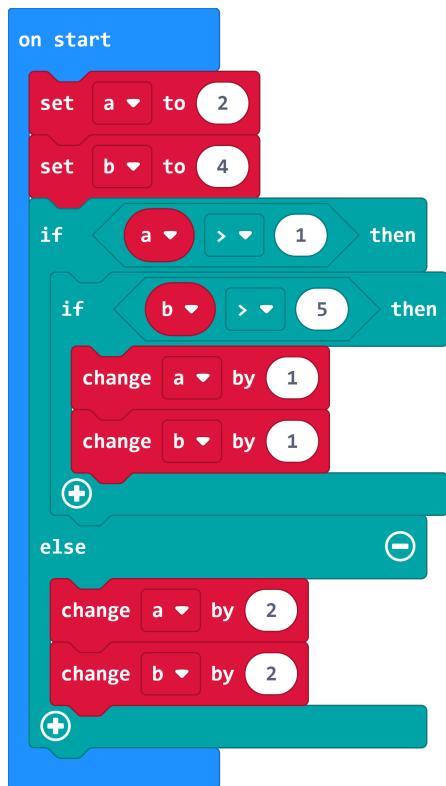


Hình 9.5: Câu lệnh nhiều điều kiện IF

Trong chương trình ở Hình 9.5, cả 2 điều kiện trong lệnh **if** đều đúng, nhưng chỉ điều kiện đầu tiên được thực thi mà thôi. Do đó, trên màn hình sẽ xuất hiện giá trị của biến a, là 2.

5 Câu lệnh IF lồng nhau

Phần trình bày này mục đích là lưu ý với người đọc một lần nữa về nguyên lý của câu lệnh IF, thông qua một ví dụ sau đây:

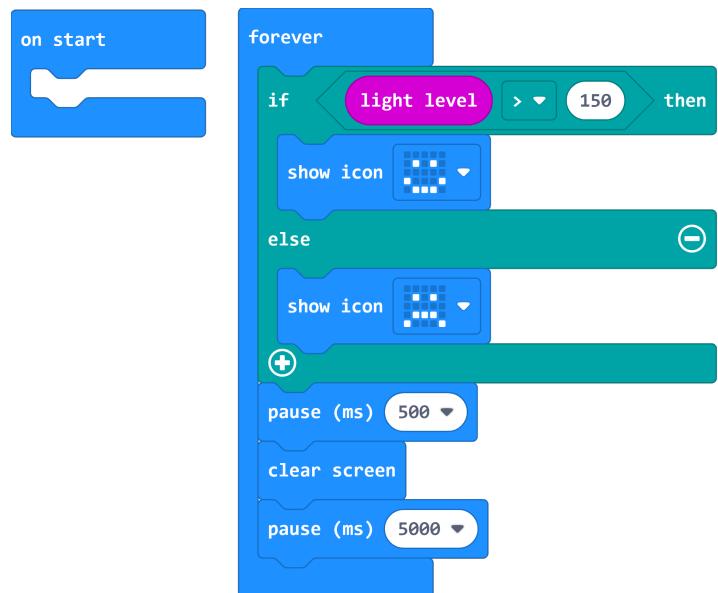


Hình 9.6: Câu lệnh điều kiện lồng nhau

Ví dụ của chương trình ở Hình 9.6 thường được gọi là câu lệnh IF lồng nhau. Nếu như chúng ta không nắm vững nguyên lý của nó, chúng ta sẽ rất dễ nghĩ rằng phần ELSE sẽ được thực thi. Tuy nhiên, phần ELSE này cùng cấp với khối IF đầu tiên. Và do là điều kiện trong khối IF đầu tiên đã đúng, nên phần ELSE sẽ không được thực thi nữa. Vào bên trong khối IF đầu tiên, chúng ta lại có một điều kiện IF thứ 2. Rõ ràng, điều kiện này sai, nên sẽ không có câu lệnh nào được thực thi. Cuối cùng, kết thúc chương trình, giá trị của 2 biến a và b là không thay đổi, vẫn là 2 và 4.

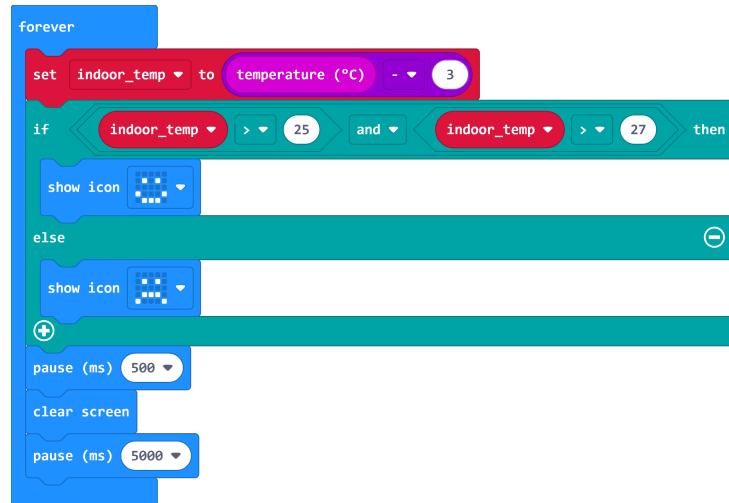
6 Bài tập

- Viết chương trình kiểm tra điều kiện ánh sáng trong phòng: Nếu cường độ ánh sáng lớn hơn 150 đơn vị thì hiện icon báo tốt, ngược lại, hiện một icon báo cường độ ánh sáng đang xấu. Icon sẽ được hiện ra trong 0.5s và tắt đi. Chu kỳ kiểm tra cường độ ánh sáng là 5s một lần. Lý do mà chúng ta phải tắt icon đi, là vì màn hình 25 đèn của MicroBit cũng chính là cảm biến ánh sáng. Do đó nếu để nó sáng liên tục, kết quả đo đạc sẽ không còn chính xác nữa.



Hình 9.7: Đáp án cho Bài 1

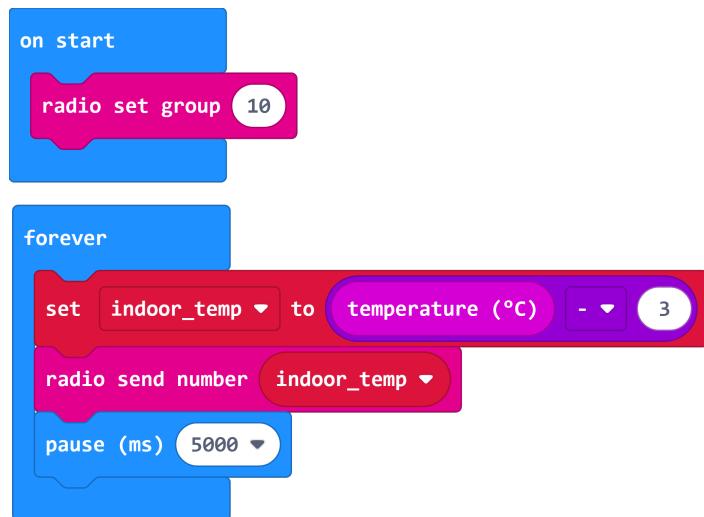
2. Viết chương trình kiểm tra điều kiện nhiệt độ trong phòng: Nếu nhiệt độ từ 25 đến 27 độ thì báo tốt, ngược lại, hiện một icon báo nhiệt độ đang xấu. Các yêu cầu về hiển thị tương tự như Bài 1.



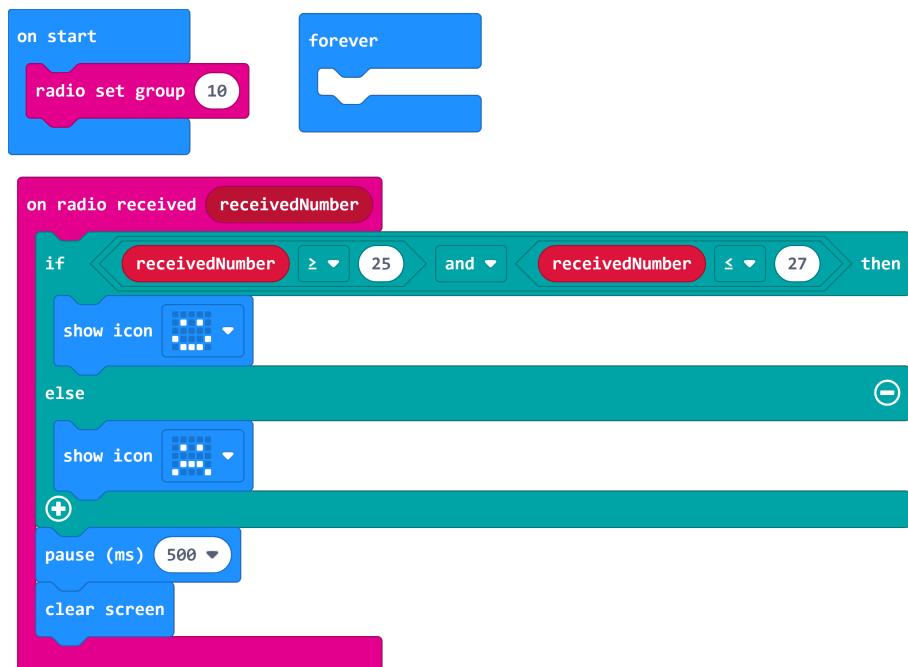
Hình 9.8: Đáp án cho Bài 2

Do cảm biến nhiệt độ trên MicroBit đang đo, là nhiệt độ của bo mạch. Do đó, chúng ta cần trừ nó đi 1 lượng nhỏ (trong đáp áp là 3 độ) để ước lượng nhiệt độ môi trường chính xác hơn. Trong bài này, chúng ta tạo thêm biến **indoor_temp** để xử lý.

3. Phát triển chương trình ở Bài 2 với giá trị nhiệt độ được gửi từ 1 mạch MicroBit khác. Mạch nhận sẽ kiểm tra và hiển thị thông báo về điều kiện nhiệt độ.



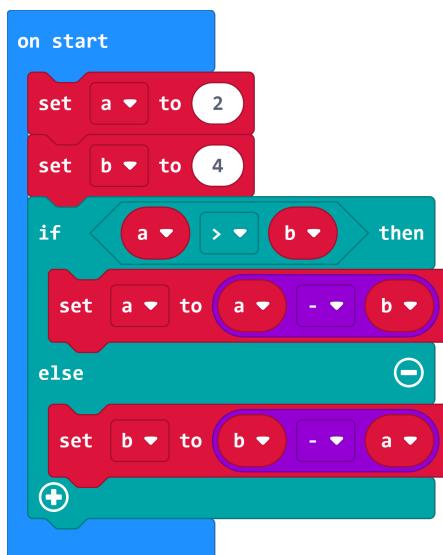
Hình 9.9: Đáp án cho Bài 3 - Mạch gửi



Hình 9.10: Đáp án cho Bài 3 - Mạch nhận

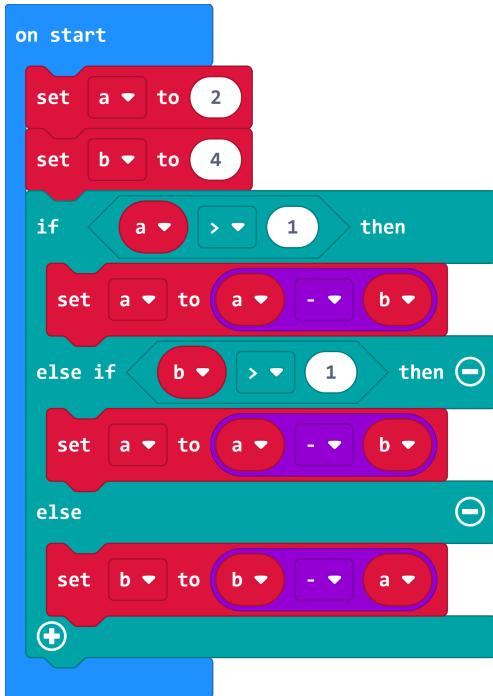
7 Câu hỏi ôn tập

1. Các lệnh nào sau đây thuộc nhóm cấu trúc điều kiện (Logic) ?
 - A. if, if else, or, and.
 - B. repeat, while, for.
 - C. Hai câu trên sai.
 - D. A và B đúng.
2. Để kiểm tra cả 2 điều kiện cùng đúng, toán tử nào sẽ được sử dụng ?
 - A. and
 - B. or
 - C. not
 - D. Tất cả đều đúng
3. Để kiểm tra cả 1 trong nhiều điều kiện là đúng, toán tử nào sẽ được sử dụng ?
 - A. and
 - B. or
 - C. not
 - D. Tất cả đều đúng
4. Sau đoạn chương trình, kết quả của a và b là bao nhiêu?



- A. $a = 2, b = 4$
- B. $a = 2, b = 2$
- C. $a = -2, b = 4$
- D. Tất cả đều sai

5. Sau đoạn chương trình, kết quả của a và b là bao nhiêu?



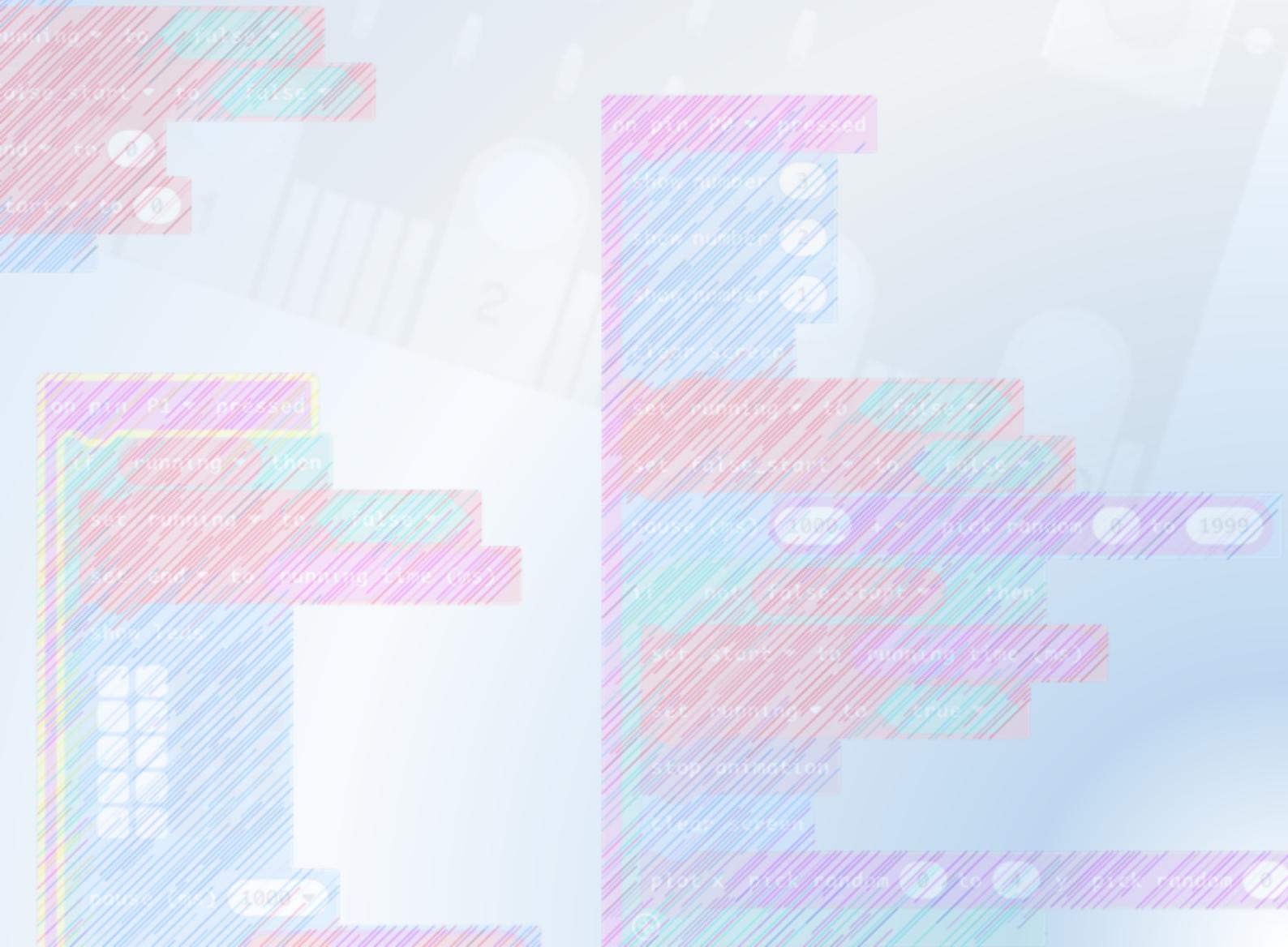
- A. a = 2, b = 4
 - B. a = 2, b = 2
 - C. a = -2, b = 4
 - D. Tất cả đều sai
6. Phát biểu nào sau đây là đúng về câu lệnh IF ELSE?
- A. Khi điều kiện IF đúng, các câu lệnh bên trong nó sẽ được thực thi
 - B. Khi điều kiện IF sai, các câu lệnh của phần ELSE sẽ được thực thi
 - C. Khi điều kiện IF đúng, các câu lệnh trong phần ELSE sẽ không được thực thi
 - D. Tất cả đều đúng
7. Phát biểu nào sau đây là đúng về câu lệnh IF nhiều điều kiện?
- A. Các điều kiện sẽ được kiểm tra lần lượt từ trên xuống dưới
 - B. Điều kiện đầu tiên đúng sẽ được thực thi
 - C. Khi một điều kiện được thực thi, chương trình sẽ không kiểm tra các điều kiện còn lại
 - D. Tất cả đều đúng

Đáp án

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

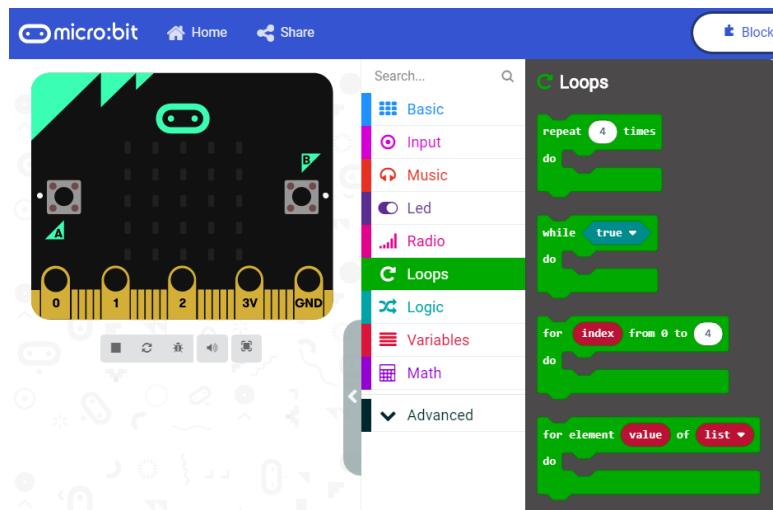
CHƯƠNG 10

Câu lệnh lắp trên MakeCode



1 Giới thiệu

Cũng như nhiều ngôn ngữ lập trình khác, MakeCode cũng cung cấp cho chúng ta các câu lệnh vòng lặp. Với những câu lệnh này, chương trình của chúng ta có thể sẽ được rút ngắn đi rất nhiều. Khi các công việc mang tính chất lặp đi lặp lại, sẽ là lúc chúng ta cần đến các khối lệnh lặp. Tuy nhiên việc sử dụng nó cần phải hết sức tỉ mỉ, nhất là trên những hệ thống phần cứng như bo mạch MicroBit. Các câu lệnh lặp trên MakeCode nằm trong mục Loops, như trình bày ở Hình 10.1.



Hình 10.1: Cấu trúc lặp trong MicroBit

Trong phần cơ bản này, chúng ta chỉ tập trung vào 3 câu lệnh đầu tiên mà thôi. Các mục tiêu cần đạt được trong bài hướng dẫn này là:

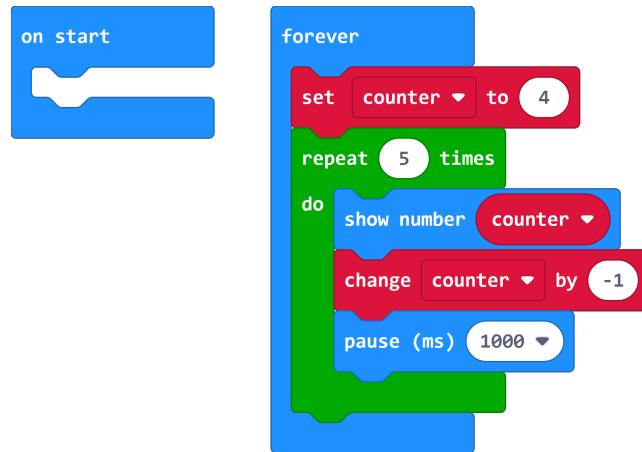
- Hiểu được nguyên lý của câu lệnh lặp
- Phân tích một bài toán có chức năng lặp đi lặp lại
- Hiện thực một yêu cầu bằng các câu lệnh lặp khác nhau

Trong hướng dẫn này, chúng ta sẽ hiện thực một chương trình đếm lùi từ 4 đến 0. Cứ sau mỗi giây, giá trị hiển thị trên màn hình của mạch MicroBit giảm đi một đơn vị. Khi đến số 0, hệ thống sẽ quay lại hiển thị số 4, và cứ lặp lại như vậy.

2 Câu lệnh repeat

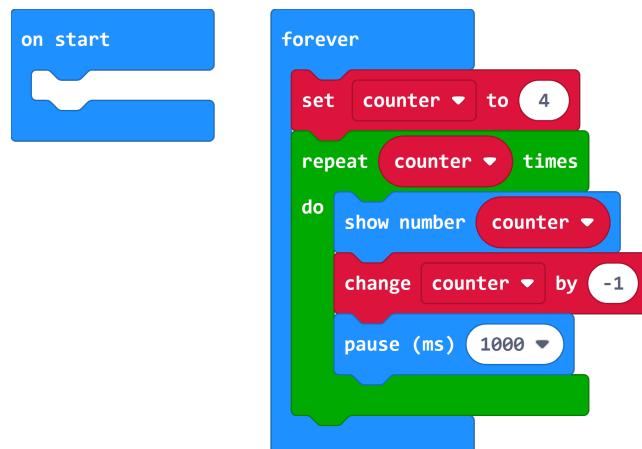
Rõ ràng với yêu cầu như bên trên, chúng ta cần phải hiện thực trong khối **forever**, vì việc hiển thị đếm lùi từ 5 đến 0 sẽ diễn ra mãi mãi. Trong khối forever, chúng ta có một chức năng mang tính lặp đi lặp lại: giảm giá trị của một biến số đi 1 đơn vị sau mỗi giây. Chức năng này sẽ thực hiện 5 lần (từ 4 đến 0).

Khi hiện thực bằng câu lệnh repeat, tham số bên trong nó là số lần lặp. Chúng ta sẽ hiện thực chương trình như sau:



Hình 10.2: Cấu trúc lặp repeat

Bản chất bên trong câu lệnh repeat trên MakeCode khá phức tạp, nên chúng tôi khuyên khích người học hãy sử dụng số lần lặp cố định cho câu lệnh này, thay vì 1 biến số mà **giá trị của nó bị thay đổi trong vòng lặp repeat**. Cách sử dụng sau đây sẽ gây ra rất nhiều bối rối cho người học:



Hình 10.3: Cấu trúc lặp repeat không nên dùng biến số ở vòng lặp

Trong chương trình ở Hình 10.3, thực sự vòng lặp repeat chỉ chạy 2 lần mà thôi. Đó là lý do tại sao chúng ta thấy màn hình MicroBit chỉ hiển thị số 4 và 3. Để lý giải cho điều này, chúng tôi phải tìm hiểu sâu về cách hiện thực của câu lệnh repeat trên MakeCode, và lý do như sau:

- Câu lệnh repeat tự tạo thêm 1 biến **i**, có giá trị ban đầu là 0. Biến này tự tăng lên 1 sau mỗi vòng lặp.
- Biến **i** sẽ được so sánh với số lần lặp (ở đây là biến **counter**), nếu nó nhỏ hơn thì mới thực hiện các câu lệnh bên trong.

Với nguyên lý như trên, ban đầu **i = 0** và thực sự nhỏ hơn **counter = 4**. Chương trình sẽ hiện ra giá trị của **counter**, tức là số 4 ra màn hình. Lần lặp thứ 2, **i = 1** và **counter = 3** (do nó đã giảm đi 1 trong lần lặp trước). Điều kiện vẫn còn đúng nên số 3 được hiển thị ra màn hình. Lần lặp thứ 3, **i = 2** và **counter = 2**, điều kiện lần này sai và

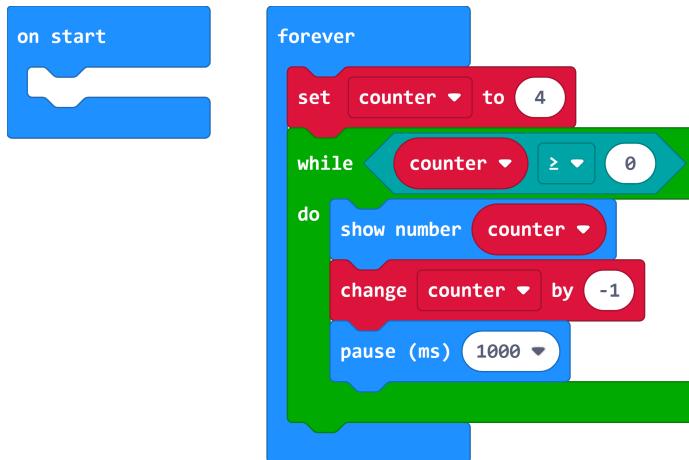
chương trình sẽ thoát khỏi vòng lặp repeat.

Với những nhập nhằng như trên, chúng ta sẽ sử dụng hằng số hoặc **biến số** **nhưng giá trị của nó không thay đổi bên trong vòng lặp** để tránh những luồng thực thi không mong đợi.

3 Câu lệnh while

Đây là câu lệnh có thể giải quyết triệt để những nhập nhằng của câu lệnh repeat trình bày ở trên. Lý do là nó rất tường minh về việc thực thi. Điều kiện sẽ được kiểm tra trước khi bắt đầu vòng lặp. Khi điều kiện là đúng, tất cả các câu lệnh bên trong while sẽ được thực thi. Điều kiện sẽ được kiểm tra lại cho lần lặp tiếp theo. Khi điều kiện là sai, chương trình sẽ thoát khỏi vòng lặp while.

Để thực hiện chức năng đếm lùi của bài hướng dẫn này với câu lệnh while, chương trình gợi ý cho chúng ta sẽ như sau:



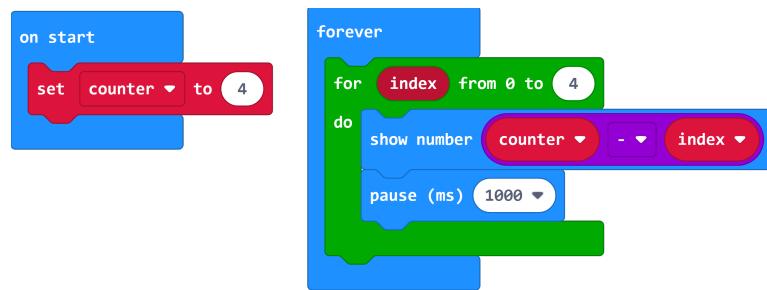
Hình 10.4: Cấu trúc lặp while

Thông qua ví dụ ở trên, chúng ta có thể thấy rằng bản chất của vòng lặp rất giống nhau. Nếu nắm kĩ nguyên lý của nó, chúng ta có thể dùng 1 vòng lặp này để thay thế vòng lặp khác.

4 Câu lệnh for

Thực ra đây là câu lệnh vô cùng quen thuộc khi chúng ta bắt đầu với câu lệnh lặp ở các ngôn ngữ truyền thống như Pascal hay C. Trong câu lệnh này, chúng ta có thêm 1 biến index, sẽ chạy từ 0 cho đến giá trị mà chúng ta mong muốn. Và với tính chất gần giống như câu lệnh repeat, chúng ta sẽ chỉ nên xài hằng số hoặc 1 giá trị biến số mà nó không bị thay đổi bên trong vòng lặp.

Để hiện thực lại yêu cầu đếm ngược ở bài trên, chúng ta sẽ dùng câu lệnh này như sau:

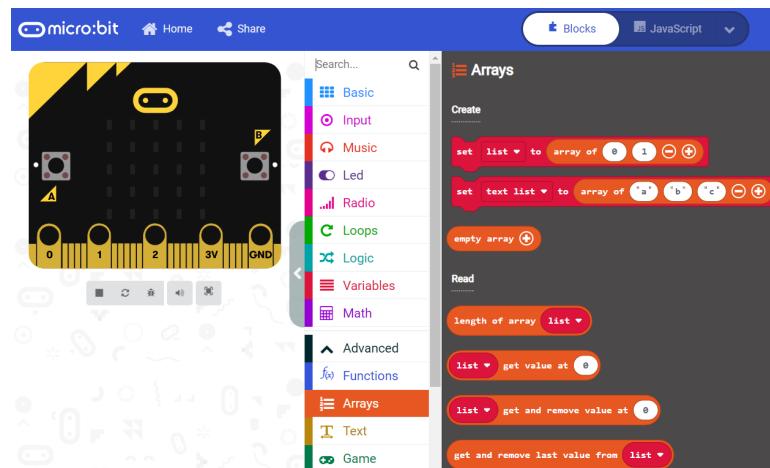


Hình 10.5: Cấu trúc lặp for

Trong chương trình trên, chúng ta đã tận dụng biến **index** có sẵn trong vòng lặp for để hiển thị cho việc đếm lùi. Toán tử trừ (-) được lấy trong mục **Math**.

5 Danh sách và lệnh for element

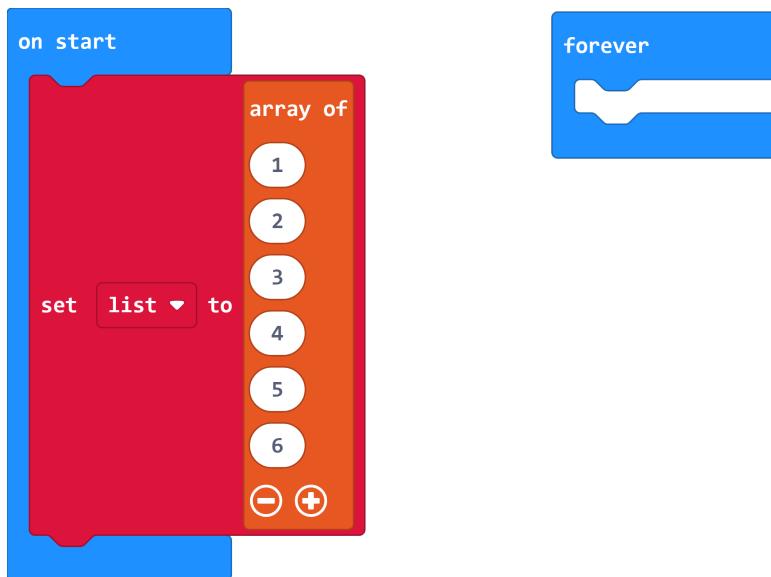
Một trong những đối tượng (hay còn gọi là cấu trúc dữ liệu) mà thường được dùng kèm với các câu lệnh lặp là danh sách. Trên MakeCode, các câu lệnh liên quan đến danh sách được nằm trong nhóm Advanced - Array, như trình bày ở hình bên dưới:



Hình 10.6: Các câu lệnh liên quan đến danh sách

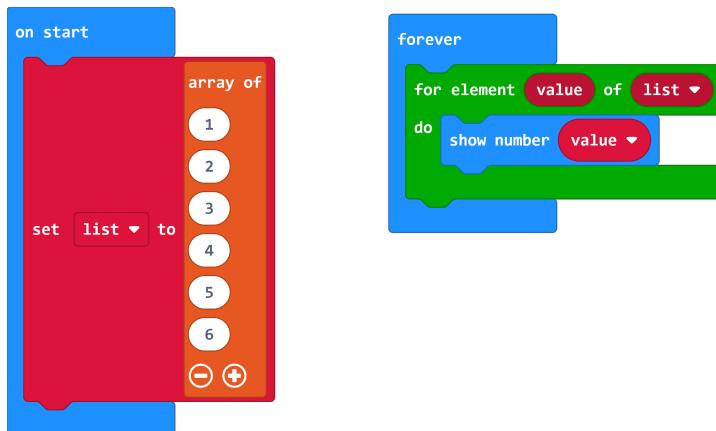
Đến đây thì chúng ta có thể tự hỏi rằng tại sao lại có cả 2 khái niệm là danh sách và mảng trên MakeCode. Thực ra đây là vấn đề kế thừa trong ngôn ngữ lập trình, mà đôi lúc sẽ gây ra nhầm lẫn cho người mới bắt đầu. Lý do ở đây, là MakeCode dùng mảng (array) để hiện thực chức năng của một danh sách. Nếu phát biểu này gây khó khăn cho người mới bắt đầu, chúng ta có thể bỏ qua và chỉ đơn giản xem như đó là cú pháp trên MakeCode mà thôi.

Để khởi tạo một danh sách, chúng ta sẽ sử dụng câu lệnh đầu tiên trong Hình 10.6. Bằng cách nhấn thêm dấu cộng (+), chúng ta sẽ có thêm các phần tử cho danh sách, như minh họa ở hình bên dưới:



Hình 10.7: Khởi tạo một danh sách trên MakeCode

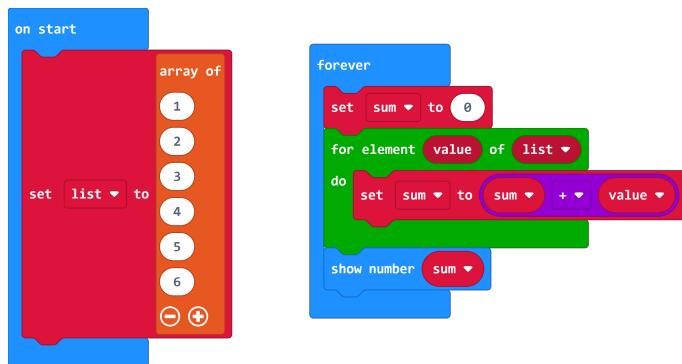
Chúng ta sẽ bắt đầu một chức năng đơn giản nhất, là duyệt qua các phần tử của danh sách. Có một lưu ý hết sức quan trọng, là **phần tử đầu tiên của danh sách sẽ được đánh chỉ số là 1**. Do đó, nếu muốn dùng 1 trong 3 câu lệnh lặp đã trình bày ở trên, chúng ta phải hết sức cẩn thận với việc truy xuất chỉ số cho phần tử đầu tiên. Tuy nhiên, trong phần này, chúng tôi sẽ trình bày câu lệnh lặp thứ 4 trong phần Loops, vốn được thiết kế rất tiện lợi cho việc truy xuất các phần tử trong danh sách. Chương trình in ra các phần tử của danh sách đơn giản sẽ như sau:



Hình 10.8: In các phần tử của danh sách ra màn hình

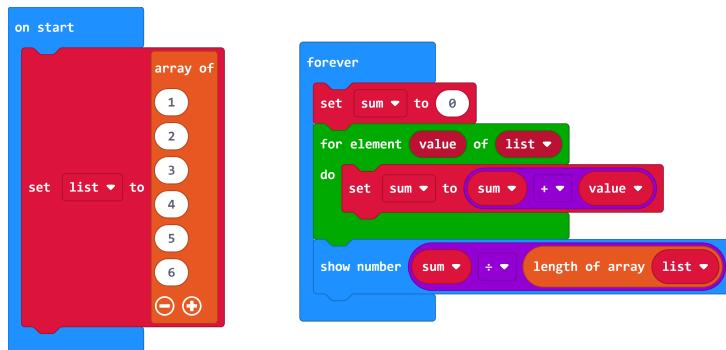
Nếu muốn tính tổng các phần tử trong danh sách, chúng ta khai báo thêm biến **sum** và hiện thực việc tính tổng như sau:

Trong các dự án thực tế, danh sách sẽ vô cùng hiệu quả nếu chúng ta muốn đọc dữ liệu từ cảm biến. Việc đọc dữ liệu từ cảm biến 1 lần và xử lý, thực chất không phải là giải pháp hiệu quả, vì nó sẽ bị nhiễu từ môi trường thực tế. Thông thường, chúng ta sẽ phải đọc dữ liệu nhiều lần, rồi lấy trung bình của 6 lần đọc gần nhất



Hình 10.9: Tính tổng các phần tử trong danh sách

chẳng hạn. Chương trình để tính giá trị trung bình các phần tử trong danh sách được gợi ý như sau:



Hình 10.10: Tính giá trị trung bình các phần tử trong danh sách

Như vậy, các khái niệm cơ bản nhất về một danh sách đã được trình bày, bao gồm việc khởi tạo và duyệt qua các phần tử của danh sách. Tùy vào yêu cầu cụ thể, mà việc xử lý của chúng ta sẽ thay đổi và được hiện thực trong hàm **for element**. Sử dụng danh sách và câu lệnh lặp một cách hiệu quả sẽ gia tăng đáng kể hiệu suất của hệ thống.

Trong quá trình duyệt danh sách, đôi lúc bạn cũng cần kết thúc quá trình này một cách đột ngột. Điều này cũng được hỗ trợ trên MakeCode bằng câu lệnh **break** nằm trong mục **Loops**. Đến đây, những bài toán kinh điển của các ngôn ngữ truyền thống như Pascal có thể được luyện tập trên MakeCode, chẳng hạn như tìm số lớn nhất, số bé nhất, sắp xếp danh sách, lọc ra số chẵn, số lẻ hay thậm chí là số nguyên tố. Phần này chúng tôi sẽ không trình bày chi tiết ở đây và xem như là thử thách cho người học để luyện tập thêm.

6 Câu hỏi ôn tập

1. Các lệnh nào sau đây thuộc nhóm cấu trúc lặp (loop) ?
 - A. if, if else, or, and.
 - B. repeat, while, for.
 - C. Hai câu trên sai.
 - D. A và B đúng.
2. Các lệnh nào sau đây thuộc nhóm cấu trúc điều kiện (logic) ?
 - A. if, if else, or, and.
 - B. repeat, while, for.
 - C. Hai câu trên sai.
 - D. A và B đúng.
3. Để lặp với số lần lặp cố định, câu lệnh phù hợp nhất là gì?
 - A. repeat
 - B. while
 - C. for
 - D. for element
4. Để lặp cho tới khi một điều kiện sai, câu lệnh phù hợp nhất là gì?
 - A. repeat
 - B. while
 - C. for
 - D. for element
5. Để lặp với chỉ số được đánh từ 0, câu lệnh phù hợp nhất là gì?
 - A. repeat
 - B. while
 - C. for
 - D. for element
6. Để duyệt qua tất cả các phần tử trong danh sách, câu lệnh phù hợp nhất là gì?
 - A. repeat
 - B. while
 - C. for
 - D. for element
7. Phần tử đầu tiên trong danh sách có chỉ số là bao nhiêu?
 - A. 0
 - B. 1
 - C. Không xác định
 - D. Tất cả đều sai

Đáp án

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

Phần II

Dự án với MicroBit

CHƯƠNG 11

Đồng hồ thông minh trên MicroBit

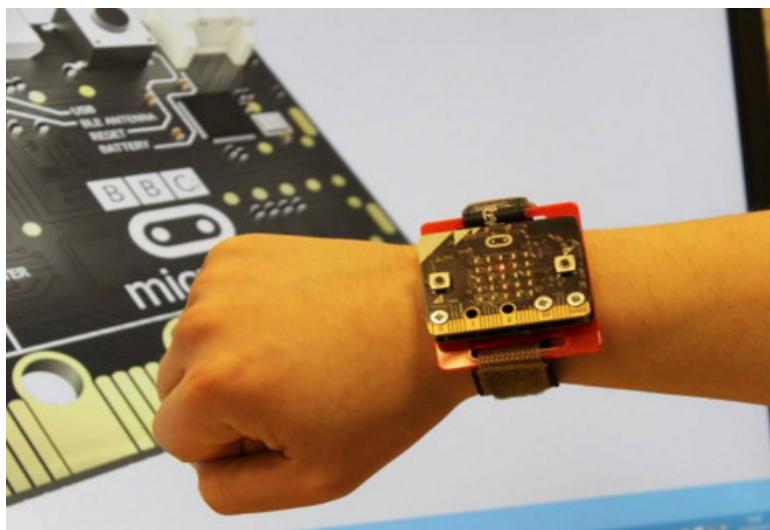


1 Giới thiệu

Đồng hồ thông minh là một dự án cực kì hữu ích cho những ai tiếp cận với việc lập trình trên bo mạch. 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 hiện thực rất nhiều thì mới có thể làm đủ các chức năng của một đồng hồ, bao gồm việc chỉnh giờ, ngày tháng hoặc là hẹn giờ chẳng hạn. Các kỹ năng mà chúng ta sẽ đạt được trong dự án này như sau:

- Học sinh hiểu được nguyên lý tổ chức chương trình trong MicroBit.
- Học sinh có khả năng hiện thực một dự án nhỏ trên MicroBit.
- Học sinh có khả năng tự mở rộng dự án của mình.

Trong bài học này, chúng ta sẽ dùng mạch MicroBit để mô phỏng một đồng hồ thông minh. Hãy tưởng tượng rằng bạn có một dây đeo mạch MicroBit vào tay và dùng nó như một chiếc đồng hồ, như minh họa ở Hình 11.1. Ngoài chức năng hiển thị giờ hiện tại, chúng ta còn có thể xem thêm các thông tin khác như nhiệt độ, cường độ ánh sáng hay thậm chí là la bàn.



Hình 11.1: Đồng hồ thông minh với MicroBit

Trong bài hướng dẫn này, chúng ta sẽ hiện thực các chức năng như sau:

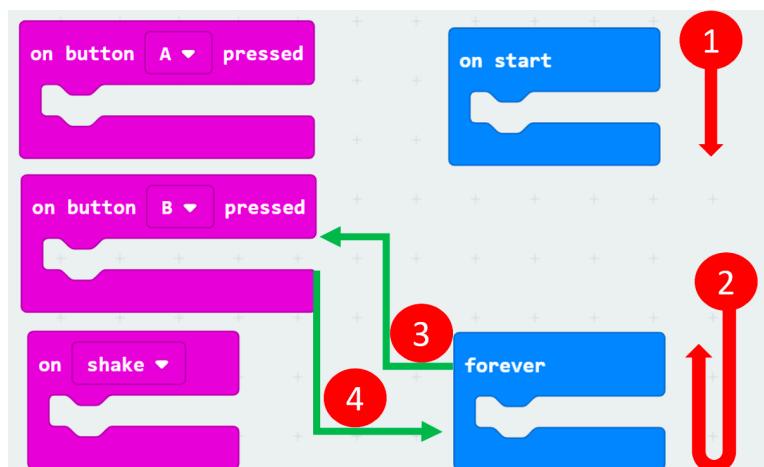
- Để tiết kiệm năng lượng, bình thường đồng hồ không hiển thị gì ra màn hình
- Khi người dùng muốn xem giờ, chỉ cần lắc tay, thông tin giờ phút giây sẽ được hiện ra
- Khi người dùng nhấn nút A, thông tin về nhiệt độ hiện tại sẽ được hiển thị ra màn hình
- Khi người dùng nhấn nút B, thông tin về cường độ ánh sáng sẽ được hiện ra
- Tất cả các thông tin hiển thị ra chỉ tồn tại trong 5 giây, sau đó màn hình sẽ tắt để tiết kiệm pin

2 Nguyên lý thực thi chương trình trên MicroBit

Việc nắm vững nguyên lý thực thi chương trình trên MicroBit là rất quan trọng để thực hiện các dự án với độ phức tạp ngày càng cao. Khác với các ngôn ngữ lập trình truyền thống như Pascal hoặc thậm chí là Scratch, chương trình trên MicroBit sẽ thực thi các câu lệnh trong khối on start trước, và sau đó **lặp đi lặp lại ở khối forever**. Trong khi đó, các ngôn ngữ lập trình khác, đa số các câu lệnh được thực thi theo thứ tự từ trên xuống dưới cho đến câu lệnh cuối cùng.

Điểm khác biệt thứ 2, một khác biệt khá lớn giữa ngôn ngữ lập trình trên MicroBit và các ngôn ngữ lập trình trên máy tính, là các chương trình hiện thực trong các sự kiện (ví dụ: sự kiện nhấn nút A, nhấn nút B hay lắc nhẹ mạch MicroBit). Khi một sự kiện xảy ra, việc thực thi lệnh trong khối forever sẽ tạm dừng lại, để cho chương trình bên trong khối lệnh được thực thi. Sau đó, MicroBit sẽ quay lại và tiếp tục thực hiện các câu lệnh được lặp đi lặp lại trong khối forever.

Hình 11.2 minh họa nguyên lý thực thi chương trình trên MicroBit. Bình thường, chương trình sẽ thực hiện hết các câu lệnh trong khối on start, sau đó chuyển sang khối forever và lặp đi lặp lại các câu lệnh trong khối này. Tuy nhiên, khi có 1 sự kiện nào đó xảy ra, việc thực thi chương trình trong forever sẽ tạm ngưng, và bắt đầu lại khi các câu lệnh trong khối sự kiện được thực thi hết.



Hình 11.2: Nguyên lý thực thi chương trình trên MicroBit

3 Hiện thực chức năng của đồng hồ

3.1 Đếm giờ của đồng hồ

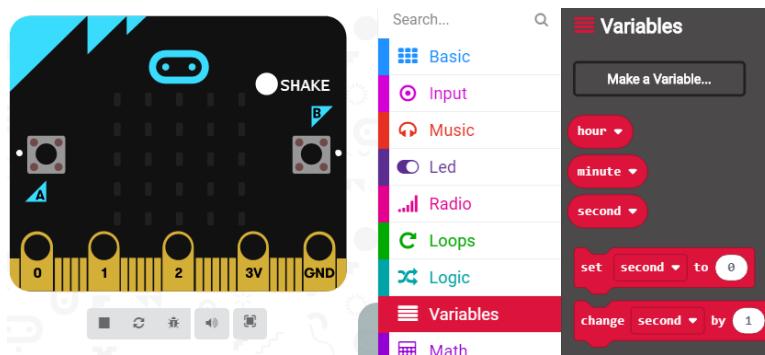
Đầu tiên, chúng ta hiện thực chức năng cơ bản của một đồng hồ điện tử, là tự động cập nhật sau mỗi giây. Rõ ràng, việc này sẽ được thực hiện lặp đi lặp lại, do đó chúng ta sẽ hiện thực chức năng này trong khối lệnh forever.

Trước tiên, chúng ta cần khai báo 3 biến số là **second**, **minute** và **hour**, để lưu giữ thông tin về giây, phút và giờ. Để làm việc này, chúng ta vào mục Variable, chọn Make a Variable, đặt tên cho biến số và nhấn OK, như minh họa ở Hình 11.3.



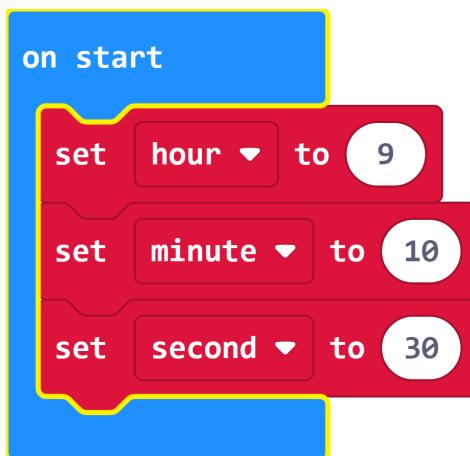
Hình 11.3: Tạo một biến trên MicroBit

Chúng ta tiếp tục lại thao tác này cho 2 biến còn lại, là minute và hour. Cuối cùng, chúng ta có thêm các khối lệnh mới và các biến số mới vừa được tạo ra như Hình 11.4.



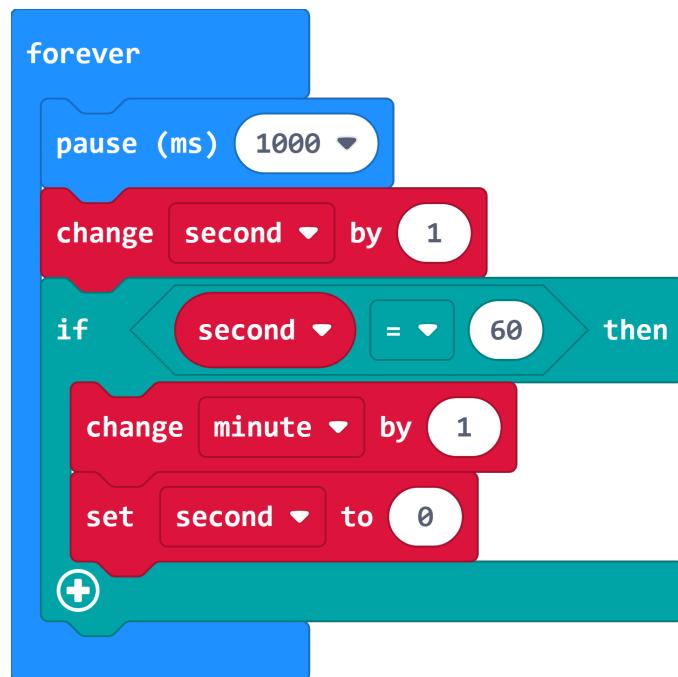
Hình 11.4: Chương trình sau khi tạo ra 3 biến số

Ở đầu chương trình, trong khối lệnh on start, chúng ta có thể chỉnh giá trị của 3 biến số này gần giống với giờ hiện tại của chúng ta. Ví dụ như hiện tại là 9 giờ, 10 phút và 30 giây, chúng ta sẽ hiện thực như chương trình ở Hình 11.5.

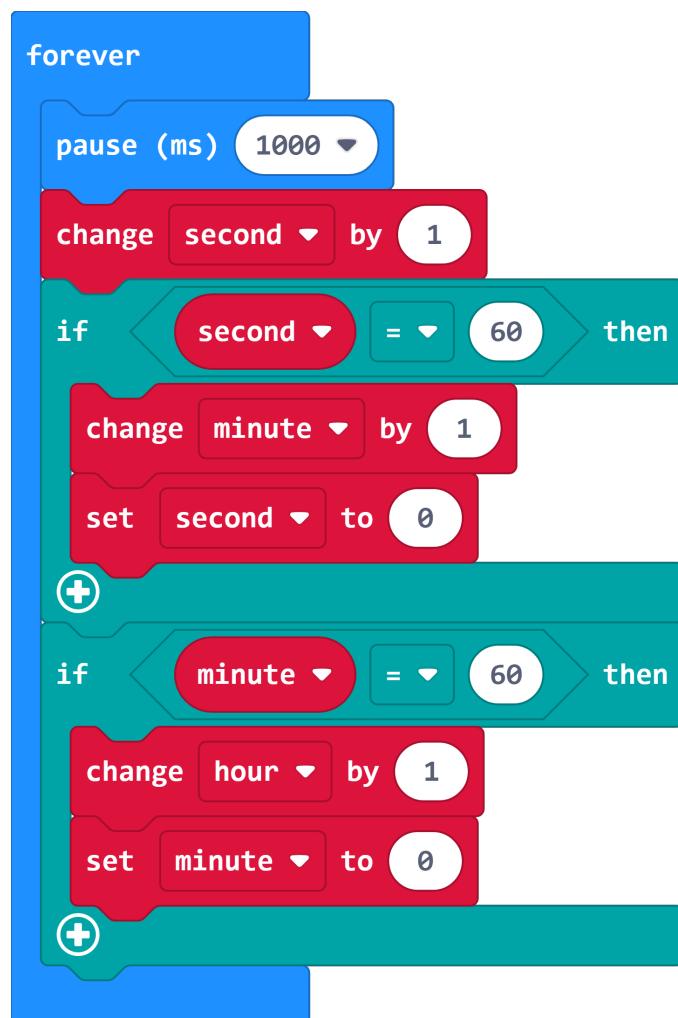


Hình 11.5: Chỉnh trạng thái cho đồng hồ gần với giờ hiện tại

Tiếp theo trong phần forever, chúng ta sẽ cho giây (biến second) tăng lên 1 đơn vị sau mỗi giây (dùng câu lệnh delay). Tuy nhiên sau đó, chúng ta phải kiểm tra biến second này, nếu nó là 60, chúng ta cho nó về 0 và tăng phút (biến minute) lên một đơn vị, đúng theo nguyên tắc của đồng hồ. Chương trình gợi ý cho chúng ta được trình bày ở Hình 11.6.

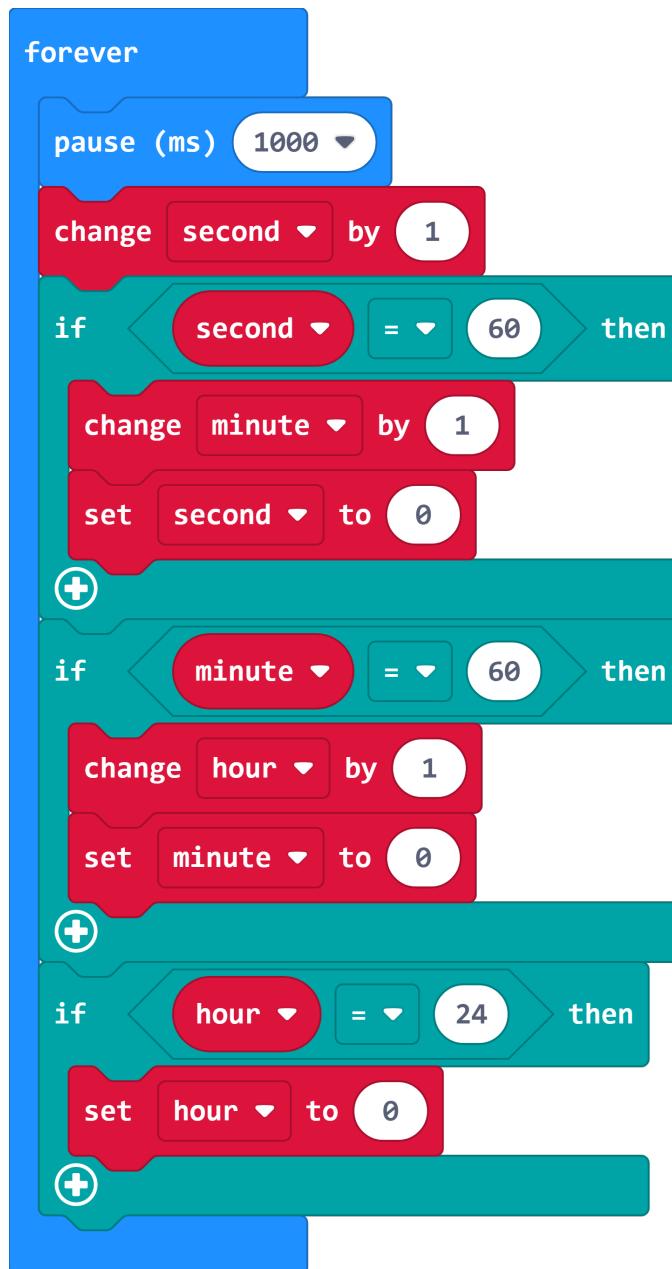


Hình 11.6: Thay đổi giây, và cập nhật phút khi đã hết 60 giây



Hình 11.7: Tăng giờ mỗi 60 phút

Tiếp tục hiện thực tương tự cho giờ, khi đủ 24 giờ, thì cho giờ về lại 0. Kết quả chương trình cho đến lúc này sẽ như sau:

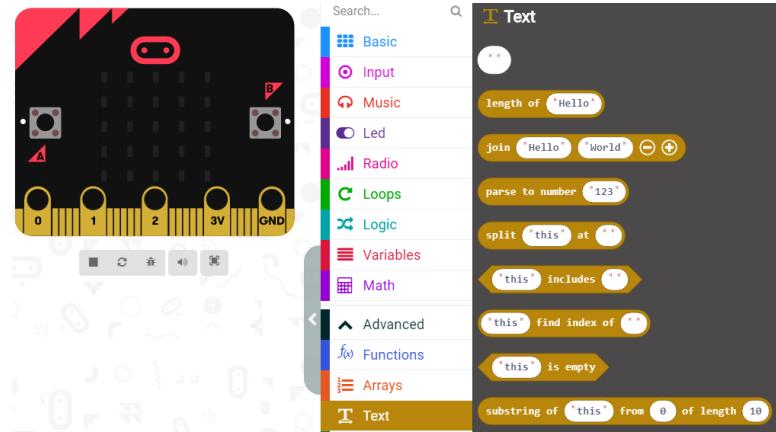


Hình 11.8: Chương trình cập nhật giờ hoàn chỉnh

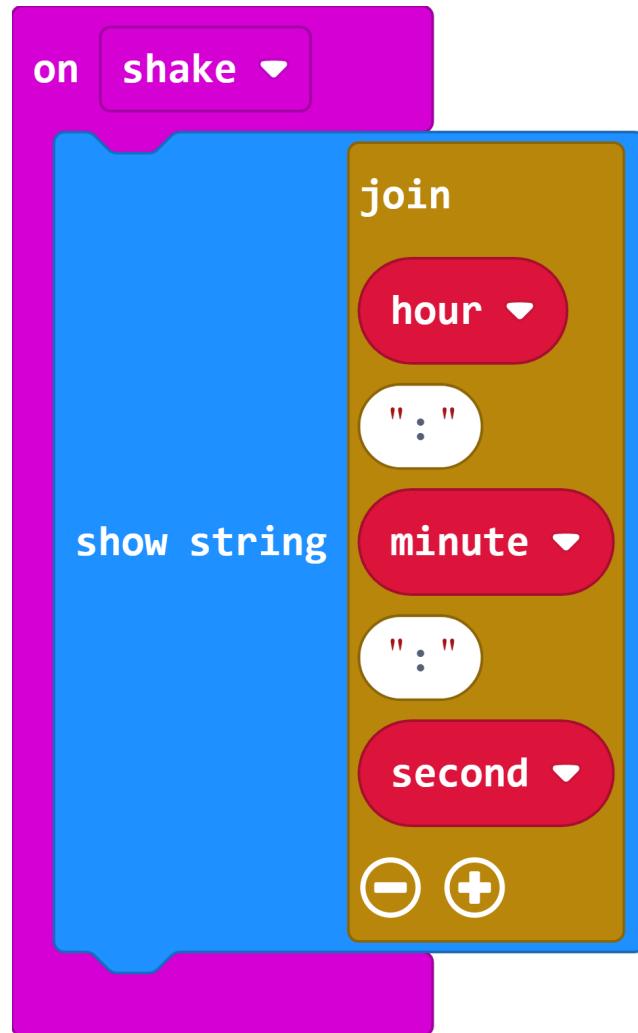
3.2 Hiển thị giờ ra màn hình

Để hiển thị thông tin giờ ra màn hình, chúng ta sẽ dùng câu lệnh ghép chuỗi trong mục **Advanced**, **Text** và sau đó chọn **join**. Câu lệnh này cho phép chúng ta ghép nhiều chuỗi hiển thị lại với nhau. Để thêm thông tin ghép nối, chúng ta sẽ nhấn vào dấu cộng (+) ở cuối câu lệnh join.

Vì đồng hồ chỉ hiển thị giờ khi chúng ta lắc tay, chương trình hiển thị giờ sẽ được hiện thực trong hàm on shake, như sau:



Hình 11.9: Câu lệnh join trong mục Advance/Text

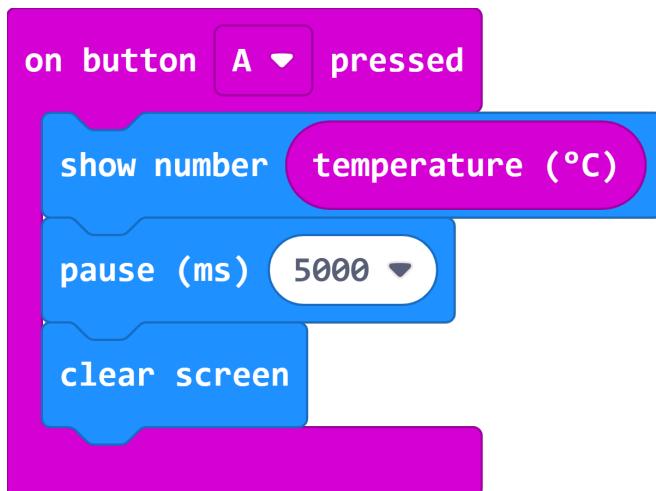


Hình 11.10: Hiển thị đồng hồ ra màn hình khi lắc tay

Chúng ta sẽ ghép thông tin giờ phút giây, cách nhau bằng dấu “:” để hiển thị ra màn hình. Khi muốn thêm 1 ô để ghép, chúng ta sẽ nhấn vào dấu + ở cuối lệnh join. Vì chuỗi thông tin này rất dài, nên khi hiển thị xong, đồng hồ sẽ tự động tắt, chúng ta không cần phải chủ động tắt màn hình.

3.3 Hiển thị nhiệt độ

Chương trình để hiển thị nhiệt độ sẽ được hiện thực khi nhấn vào nút A, như sau:



Hình 11.11: Hiển thị nhiệt độ khi nhấn nút A

Cần lưu ý là để có hiệu ứng chờ 5 giây, chúng ta có thể **nhập tay vào số 5000** hoặc lựa chọn trong danh sách của câu lệnh pause. Câu lệnh clear screen có thể được tìm thấy trong mục **more**, thuộc nhóm **Basics**.

3.4 Hiển thị cường độ ánh sáng

Hoàn toàn tương tự như chương trình ở bên trên, chúng ta hiện thực chương trình cho nút B như sau:



Hình 11.12: Hiển thị cường độ ánh sáng khi nhấn nút B

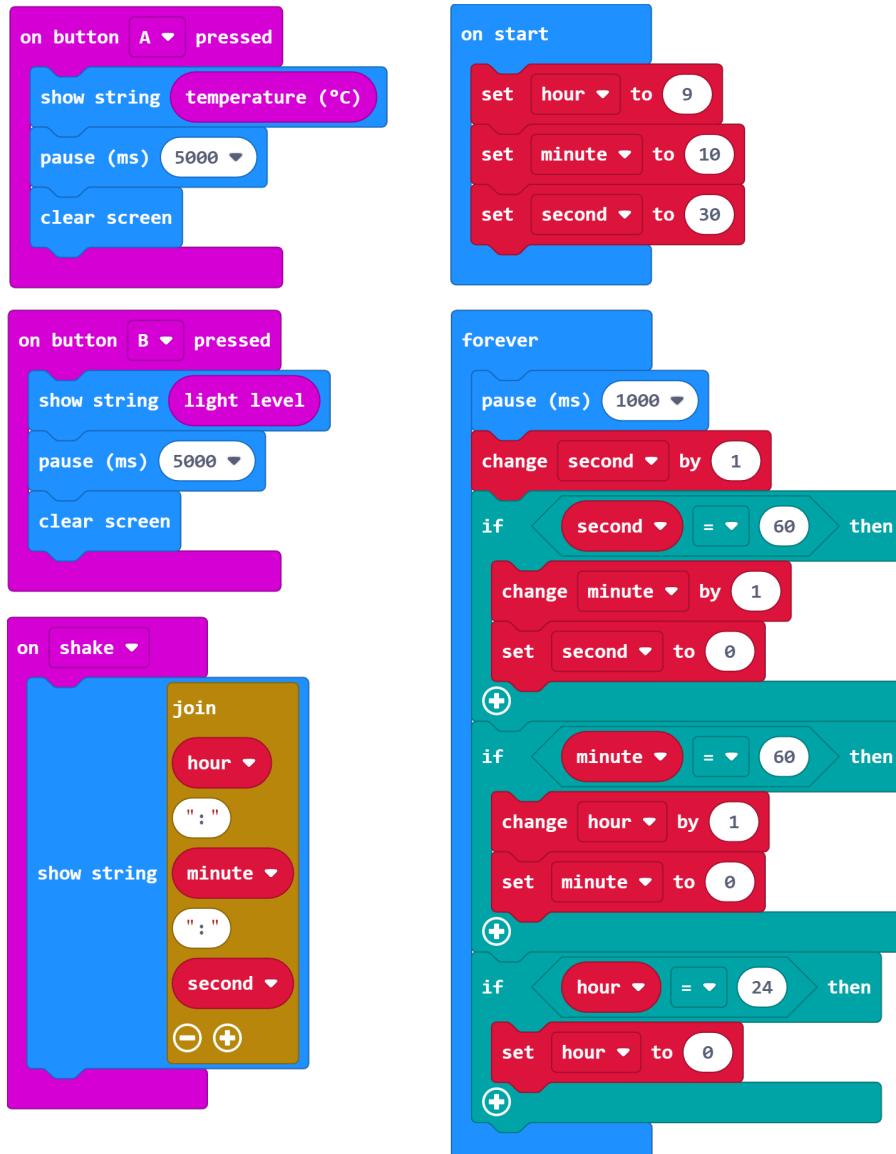
Cuối cùng, chương trình hoàn thiện của chúng ta sẽ như sau:

3.5 Các hướng phát triển

Chúng ta có thể phát triển chương trình này như một chiếc đồng hồ Casio điện tử. Với sức mạnh và sự hỗ trợ của MicroBit, nó hoàn toàn có thể làm được các chức năng như một đồng hồ Casio, bao gồm:

- Chính ngày, tháng và năm
- Chính giờ và phút
- Hẹn giờ
- Đồng hồ thể thao

Trong hướng dẫn này, chúng tôi không trình bày các câu lệnh để hiện thực các chức năng trên. Chúc mọi người may mắn trong việc hiện thực nó!!



Hình 11.13: Chương trình đồng hồ thông minh hoàn thiện

4 Câu hỏi ôn tập

1. Trong dự án đồng hồ thông minh, các giá trị ban đầu của thời gian được hiện thực trong khôi nào?
 - A. on start
 - B. forever
 - C. on button A pressed
 - D. on button B pressed
2. Trong dự án đồng hồ thông minh, chức năng cập nhật lại giờ phút và giây được hiện thực trong khôi nào?
 - A. on start
 - B. forever
 - C. on button A pressed
 - D. on button B pressed
3. Trong dự án đồng hồ thông minh, hiển thị cường độ ánh sáng khi nhấn nút A được hiện thực trong khôi nào?
 - A. on start
 - B. forever
 - C. on button A pressed
 - D. on button B pressed
4. Trong dự án đồng hồ thông minh, hiển thị nhiệt độ khi nhấn nút B được hiện thực trong khôi nào?
 - A. on start
 - B. forever
 - C. on button A pressed
 - D. on button B pressed
5. Khi muốn xem giờ hiện tại bằng sự kiện lắc tay, khôi lệnh nào đã được sử dụng?
 - A. on button A pressed
 - B. on button B pressed
 - C. on shake
 - D. Tất cả đều đúng

Đáp án

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



CHƯƠNG 12

Cửa mật mã trên MicroBit

1 Giới thiệu

Một dự án nữa có thể làm với mạch MicroBit là mô phỏng một cửa có khóa mật mã. Trong dự án này, chúng ta sẽ dùng nút nhấn A tương ứng cho kí tự A, nút nhấn B tương ứng cho kí tự B. Người dùng sẽ nhấn nút để nhập mật khẩu và nhấn tổ hợp phím A+B để kiểm tra mật khẩu. Khi mật khẩu là đúng, mạch MicroBit sẽ điều khiển một động cơ để mở cửa. Trong trường hợp ngược lại, khi mật khẩu là sai, cửa sẽ đóng.

Một điểm khác biệt trong dự án này, là chúng ta sẽ làm việc trên dữ liệu kiểu chuỗi, vốn là 1 kiểu dữ liệu tương đối phức tạp khi lập trình nói chung và trên mạch MicroBit nói riêng. Bên cạnh đó, chúng ta cũng sẽ thêm một thư viện lập trình mở rộng để có thể điều khiển động cơ. Các kỹ năng mà chúng ta sẽ đạt được trong dự án này có thể tóm lược như sau:

- Tương tác và xử lý với dữ liệu kiểu chuỗi
- Điều khiển động cơ mô phỏng cho việc đóng/mở cửa
- Thêm thư viện lập trình mở rộng trên MakeCode

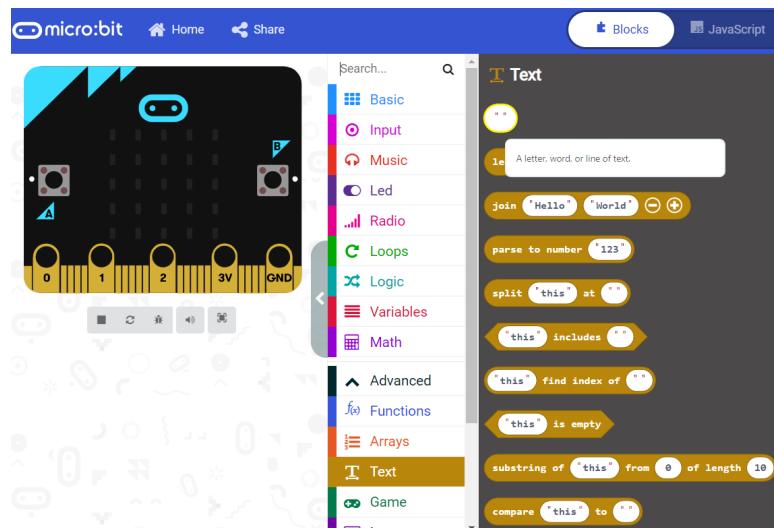
Trong dự án này, chúng ta sẽ giả sử mật mã là chuỗi "AABB".

2 Khai báo biến và khởi tạo

Để tiện cho việc mở rộng trong tương lai, chúng ta sẽ khai báo 2 biến số như sau:

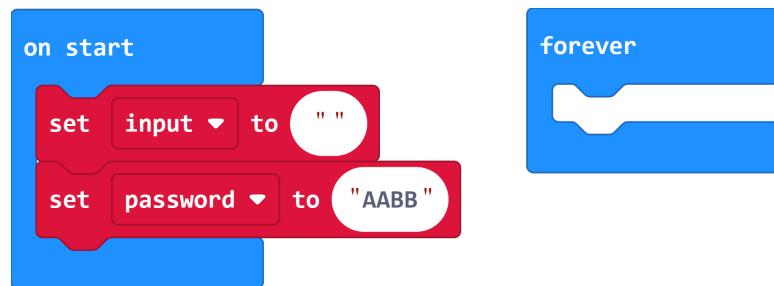
- input: Lưu giữ giá trị nhập vào từ người dùng
- password: Lưu giữ giá trị mật khẩu của hệ thống

Khi nhấn tổ hợp nút A+B, chúng ta sẽ so sánh 2 biến số này với nhau. Tuy nhiên, khi mới khai báo biến, chương trình trên MakeCode mặc định nó là dữ liệu kiểu số, với giá trị ban đầu là 0. Để chuyển nó sang dữ liệu kiểu chuỗi, chúng ta sẽ phải cần toán tử đầu tiên trong mục **Advanced, Text**, như minh họa ở hình bên dưới:



Hình 12.1: Khối lệnh chuỗi rỗng trong mục Text - Câu lệnh đầu tiên

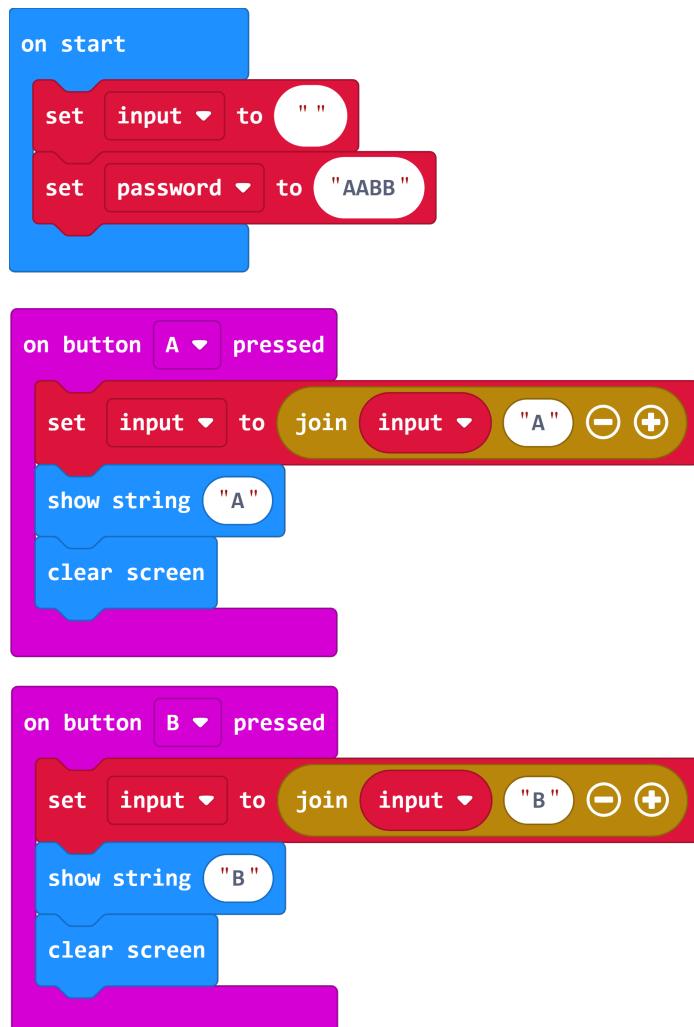
Khi sử dụng khối lệnh này và ghép với khối lệnh gán giá trị cho biến, chúng ta sẽ đổi biến số đó về kiểu chuỗi. Chúng ta sẽ hiện thực điều này trong khối lệnh on start, như sau:



Hình 12.2: Khởi tạo giá trị cho 2 biến kiểu chuỗi

3 Ghép chuỗi khi nhấn nút A hoặc nút B

Tiếp theo, chúng ta sẽ xử lý giá trị của biến **input** mỗi khi nhấn A hoặc B. Theo yêu cầu của bài toán, nút A tương ứng cho ký tự A và nút B tương ứng cho ký tự B. Do đó, chúng ta sẽ ghép ký tự mới nhấn vào bên phải của biến input. Việc này sẽ được thực hiện bằng khối lệnh **join**, cũng nằm trong mục **Text**, như kết quả ở chương trình sau:

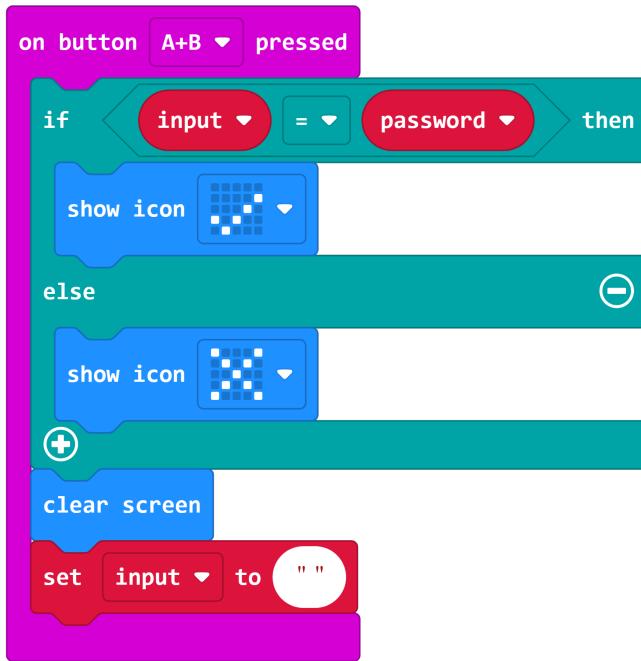


Hình 12.3: Xử lý khi nhấn nút A và B

Trong chương trình trên, chúng ta cũng hiển thị ra màn hình các kí tự A và B để tăng tính thân thiện đối với người dùng. Các kí tự này chỉ hiện ra trong một thời gian ngắn (khoảng nửa giây), rồi mất đi.

4 So sánh mật khẩu khi nhấn A và B

Khi nhấn 2 nút A và B đồng thời, chương trình sẽ kiểm tra những gì chúng ta đã nhập với mật khẩu. Nếu đúng mật khẩu sẽ hiển thị icon báo đúng. Ngược lại, một icon báo sai sẽ hiển thị lên. Các câu lệnh cho khôi nút nhấn A+B sẽ như sau:

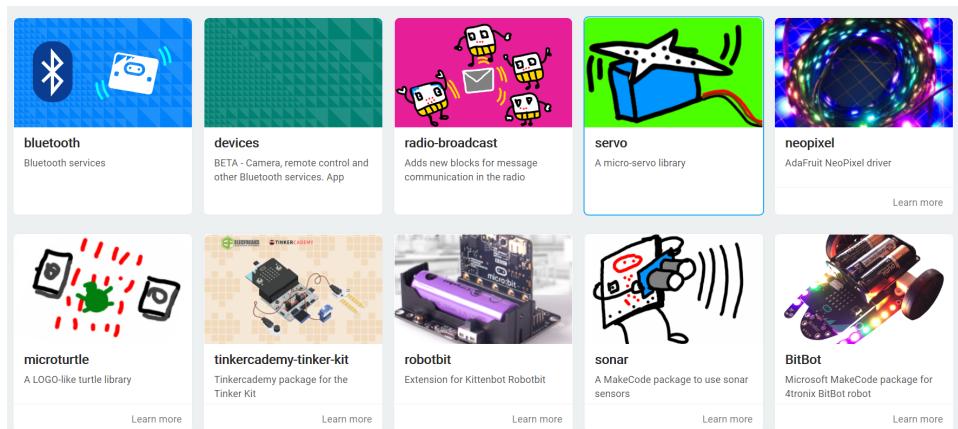


Hình 12.4: Kiểm tra mật khẩu

Cần lưu ý là toán tử so sánh bằng ở đây, được sử dụng là toán tử so sánh của chuỗi. Toán tử này là toán tử cuối cùng trong mục Comparison (thuộc khối lệnh Logic). Cuối cùng, chúng ta phải gán lại giá trị của biến input về chuỗi rỗng sau khi so sánh nó với mật khẩu. Thiếu câu lệnh này, chương trình chỉ có thể chạy đúng được một lần mà thôi.

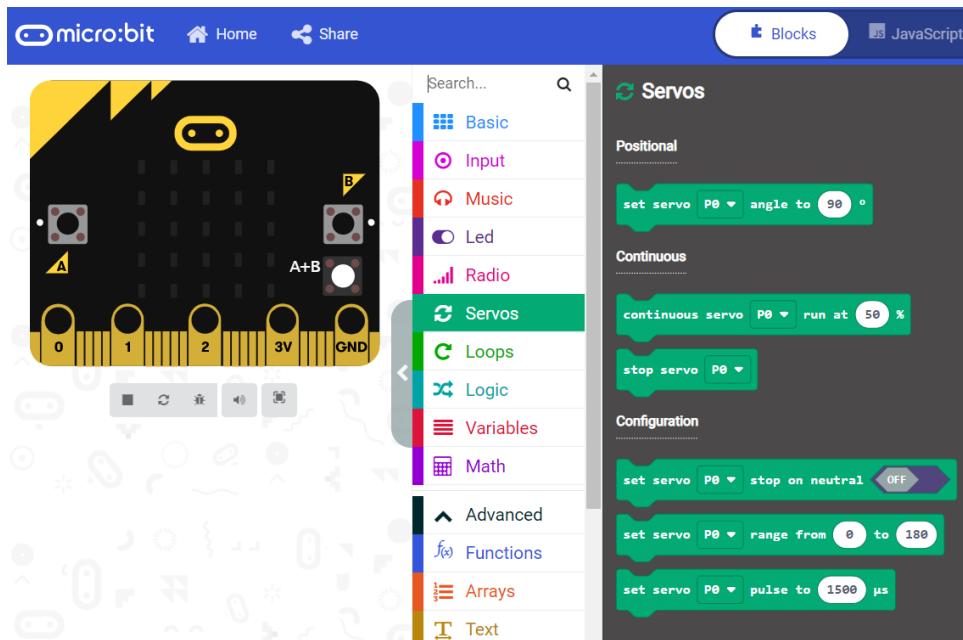
5 Thêm thư viện điều khiển động cơ

Để có thể điều khiển 1 động cơ, chúng ta cần thêm một thư viện mở rộng để hỗ trợ cho việc này. Bằng cách nhấn vào Advanced, và chọn và Extensions, chúng ta sẽ đến với các thư viện được hỗ trợ cho môi trường lập trình MakeCode, như giao diện bên dưới:



Hình 12.5: Các thư viện mở rộng trên MakeCode

Chúng ta có thể dễ dàng tìm thấy thư viện có tên là **servo** cùng biểu tượng một động cơ. Bạn hãy nhấn vào đó, đợi 1 lúc và sau đó, chúng ta sẽ thấy 1 nhóm lệnh mới, có tên là **Servos** được thêm vào cửa sổ lập trình MakeCode, như minh họa ở Hình 12.6



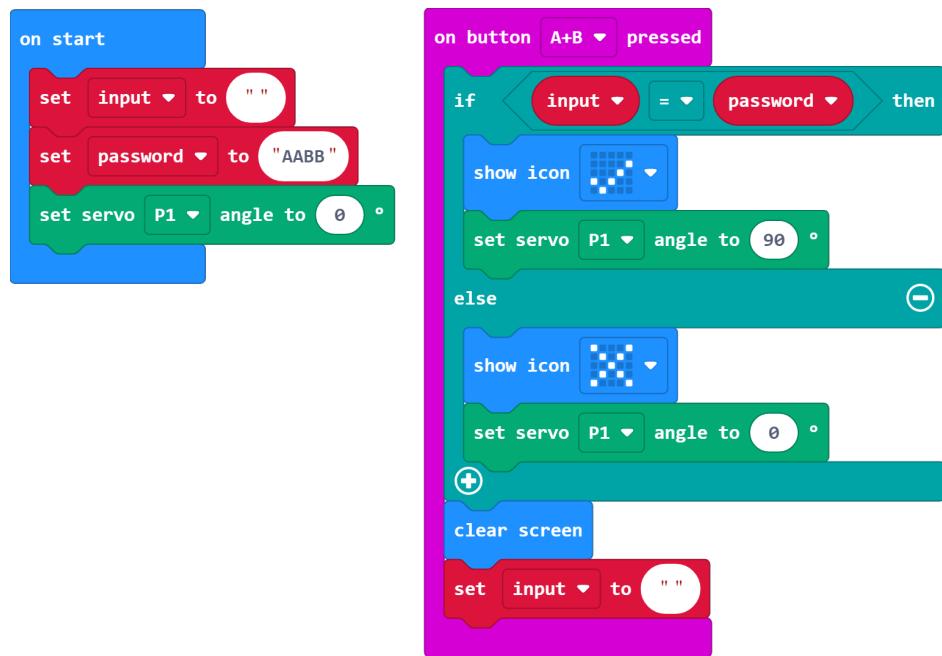
Hình 12.6: Các câu lệnh cho việc điều khiển động cơ

Thư viện này hỗ trợ cho một loại động cơ, có tên là RC Servo. Động cơ này khá đặc biệt, vì nó ko xoay tròn 360 độ như những động cơ khác. Thay vào đó, nó sẽ giữ vị trí ở những góc cố định mà người lập trình ra lệnh cho nó. Như ở môi trường MakeCode, câu lệnh đầu tiên cho phép chúng ta chỉnh động cơ ở các vị trí từ 0 đến 180 độ. Do đó, đối với người mới bắt đầu, điều này thuận tiện cho chúng ta rất nhiều ở việc lập trình điều khiển vị trí của động cơ.

Thứ hai, đó là môi trường mô phỏng tuyệt vời mà MakeCode hỗ trợ. Ngay khi kéo một câu lệnh điều khiển động cơ RC vào chương trình, chúng ta sẽ thấy ngay một động cơ mô phỏng trong MakeCode.

Cuối cùng, MakeCode cũng hướng dẫn chúng ta kết nối với động cơ cho tương thích với chương trình của chúng ta. Rõ ràng, động cơ này có thể nối trực tiếp vào mạch MicroBit mà không phải thông qua một mạch trung gian nào cả. Đây là một lợi thế không hề nhỏ cho người mới bắt đầu lập trình trên bo mạch, vốn không có nhiều kinh nghiệm về việc kết nối mạch điện.

Trong dự án này, chúng ta sẽ chỉnh động cơ ở góc 0 độ tương ứng với đóng cửa, và 90 độ tương ứng với mở cửa. Động cơ sẽ được kết nối với chân P1 của mạch MicroBit. Chương trình của chúng ta cần thay đổi ở những khía cạnh sau đây:



Hình 12.7: Đóng mở cửa bằng động cơ RC Servo

6 Các hướng mở rộng

Trong tài liệu này, chúng tôi đề xuất 2 hướng mở rộng khả dĩ cho người học. Một là, khi nhập mật khẩu sai 3 lần liên tiếp, chúng ta sẽ khóa hệ thống trong vòng 5 giây rồi mới cho phép người dùng nhập tiếp. Với hướng mở rộng này, chúng ta có thể khai báo 1 biến failure, và chỉnh giá trị của nó trong khối lệnh on button A+B pressed, như gợi ý sau đây:



Hình 12.8: Hệ thống tự khóa 5 giây nếu nhập sai mật khẩu 3 lần liên tiếp

Trong chương trình ở Hình 12.8, câu lệnh if cuối cùng trong khối lệnh on button A+B pressed là để xử lý cho việc khóa hệ thống trong vòng 5 giây. Sau khi hết 5 giây bằng câu lệnh pause, chúng ta gán lại cho biến failure = 0 để hệ thống lại hoạt động lại từ đầu.

Chúng ta cũng thay đổi 1 chút ở 2 khối lệnh cho nút nhấn A và nút nhấn B: Khi còn chưa sai 3 lần liên tiếp thì chúng ta vẫn còn hiển thị ra màn hình và xử lý. Ngược lại, khi người dùng đã nhập sai, màn hình sẽ không hiển thị gì cả.

Cuối cùng, một hướng mở rộng mà người học sẽ phải hiện thực gần như gấp đôi chương trình: Khi người dùng nhập gấp đôi mật khẩu hiện tại (AABBAABB) rồi nhấn tổ hợp phím A+B, hệ thống sẽ vào chế độ có thể chỉnh mật khẩu. Vào chế độ này, nút A vẫn là kí tự A, nút B vẫn là kí tự B. Nhưng khi nhấn vào A+B thì hệ thống sẽ lưu lại mật khẩu mới và tự động chuyển sang chế độ kiểm tra mật khẩu. Hướng mở rộng này chúng tôi hy vọng người học có thể chủ động sáng tạo, và không trình bày chi tiết ở đây.

7 Câu hỏi ôn tập

1. Toán tử nối chuỗi sử dụng trong dự án khóa cửa thông minh là gì?
 - A. Toán tử cộng
 - B. Toán tử join
 - C. Toán tử text
 - D. Tất cả đều đúng
2. Toán tử so sánh dữ liệu từ người dùng và mật khẩu là gì?
 - A. Toán tử so sánh 2 số
 - B. Toán tử so sánh 2 chuỗi
 - C. Cả 2 toán tử trên đều đúng
 - D. Cả 2 toán tử trên đều sai
3. Chuỗi rỗng nằm ở mục nào trong môi trường lập trình MakeCode?
 - A. Mục Advanced - Text
 - B. Mục Text - Advanced
 - C. Mục Advanced - Extensions
 - D. Tất cả đều sai
4. Sau khi nhấn 2 nút A+B để thực hiện việc so sánh mật khẩu, chúng ta cần gán lại giá trị rỗng cho biến input. Tác vụ này được thực hiện ở đâu?
 - A. on button A pressed
 - B. on button B pressed
 - C. on button A+B pressed
 - D. Tất cả đều đúng
5. Thư viện mở rộng để điều khiển động cơ servo có tên là gì ?
 - A. motor.
 - B. servo.
 - C. Hai câu trên đều đúng.
 - D. Hai câu trên đều sai.
6. Các lợi thế của động cơ RC servo so với các động cơ khác là gì?
 - A. Dễ điều khiển nhờ thư viện có sẵn
 - B. Dễ dàng kết nối với mạch MicroBit
 - C. Có mô phỏng trực quan trên MakeCode
 - D. Tất cả đều đúng
7. Phát biểu nào là đúng trong dự án cửa mật khẩu với MicroBit?
 - A. Khối forever không cần và có thể xóa
 - B. Khối forever luôn luôn cần và phải có
 - C. Tùy vào cách hiện thực của người lập trình mà sẽ có hay không
 - D. Tất cả đều sai

Đáp án

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