

## Khung chương trình trải nghiệm các loại robot thuộc dòng Easybot (có thể áp dụng với Wibot, RACEbot, BattleBot)

### Phần I – Giới thiệu Easybot – Bước khởi đầu với STEM

- I. Giới Thiệu Easybot
- II. Trải nghiệm đầu tiên về Easybot

### Phần II - Lập trình với ngôn ngữ Scratch

#### 1. Giới thiệu ngôn ngữ Scratch và cách cài đặt môi trường lập trình

#### 2. Scratch Programming:

- Cài đặt phần mềm mBlock lập trình ngôn ngữ Scratch dành cho Robots
- Kết nối Robot với máy tính và phần mềm mBlock
  - Kết nối qua dây USB
  - Kết nối qua Bluetooth

#### 3. Viết các chương trình cơ bản đơn giản

- Các nguyên lý di chuyển và các lệnh di chuyển
- Di chuyển theo hình vuông - kết hợp học về viết chương trình Scratch dùng cấu trúc vòng lặp có điều kiện
- Di chuyển theo hình tròn
- Thử thách khó một chút với trò chơi: Viết chương trình cho robot chạy theo hình Ziczac và hình xoắn ốc (bán kính tăng dần)

#### 4. Sử dụng các cảm biến

1. Giới thiệu về cảm biến và cách robot nhận dạng thế giới vật lý
2. Kiến thức cần nắm: Cách đọc cảm biến, Phương pháp lập trình sử dụng vòng lặp vĩnh viễn, lặp có điều kiện và các cấu trúc điều khiển rẽ nhánh if else
3. Giới thiệu cảm biến khoảng cách siêu âm và các ứng dụng
  - Cách lấy dữ liệu từ cảm biến – viết chương trình hiển thị số liệu từ cảm biến siêu âm lên máy tính
  - Trò chơi tránh vật cản đơn giản
  - Trò chơi bám theo vật
  - Trò chơi mèo đuổi chuột Tom and Jerry (kết hợp tránh vật cản và bám theo vật)
4. Giới thiệu cảm biến dò vạch dựa vào nguyên lý phản xạ ánh sáng sử dụng photo diode và cách sử dụng để xác định vị trí của robot trên vạch
5. Lập trình robot dò đường (cơ bản)
6. Giới thiệu trò chơi đấu SUMO ứng dụng cảm biến dò line và cảm biến siêu âm và tiến hành cho các em trải nghiệm thi đấu

#### III. Kết hợp cho các bài tập viết chương trình phức tạp hơn

*Các kiến thức cần nắm, sử dụng các biến và dữ liệu, tạo các block riêng gồm nhiều khối block kết hợp lại (như chương trình con)*

1. Đi vào sa hình có vạch kết hợp đếm vạch và đếm vạch ngang

2. Bài toán dò đường kết hợp nhớ trạng thái trước khi mất vạch
3. Sử dụng chương trình con (tạo hàm và block chức năng thực hiện nhiều hành động trong 1 khối lệnh block)
4. Giải bài toán dò line kết hợp tránh vật cản
5. Chơi trò chơi mê cung đơn giản
6. Trò chơi đua xe dò đường tránh chướng ngại vật

## PHẦN 1

**Giới thiệu Easybot**

Easybot là sản phẩm Robot giá thành thấp, dễ dàng sử dụng cho người mới bắt đầu để trải nghiệm về lập trình (Scratch), điện tử và Robot từ chưa biết gì. Sản phẩm được thiết kế cho giáo dục STEM dành cho lứa tuổi từ 12 trở lên.

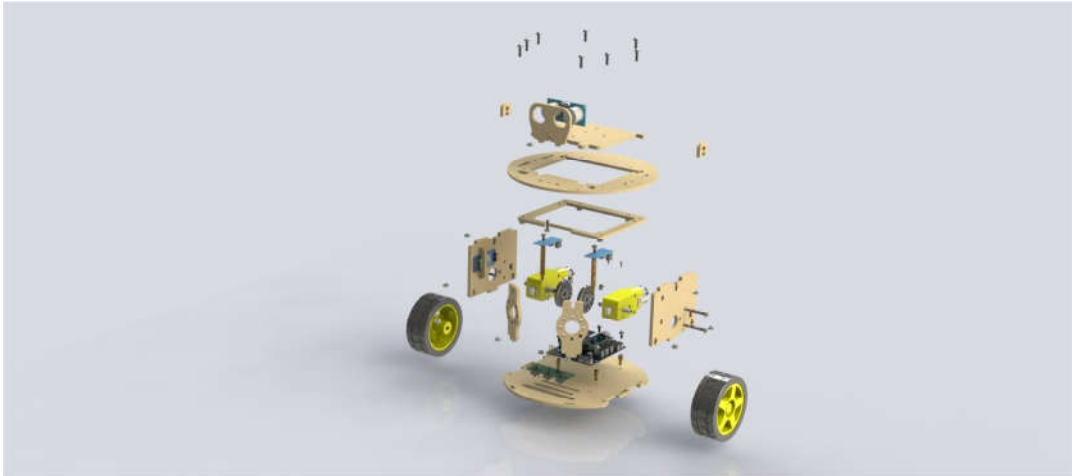
**Các tính năng chính**

- Dễ dàng lắp đặt theo hướng dẫn và video
- Board mạch điện tử thiết kế trên nền tảng mã nguồn mở Arduino
- Cảm biến đo xa kiểu siêu âm, cảm biến dò vạch Analog 3 kênh, cổng I2C mở rộng cho màn hình hiển thị và các tính năng nâng cấp
- Module Bluetooth để điều khiển qua SmartPhone và máy tính
- Dễ dàng đấu nối với các bộ dây làm sẵn dành cho các thiết bị đi kèm.
- Tùy chọn sử dụng Pin tiểu AA hoặc Pin sạc kiểu 18650
- Lập trình bằng ngôn ngữ kéo thả các khối đồ họa dựa trên ngôn ngữ Scratch 2.0
- Lập trình nâng cao bằng ngôn ngữ C cho Arduino
- Các bài học miễn phí dễ dàng học online hoặc download về để học

**Easybot có những khả năng gì**

- Điều khiển từ xa
- Lập trình kéo thả Scratch và Arduino
- Dò theo vạch (Line following)
- Đo khoảng cách để phát hiện tránh vật
- Quét Radar và bám theo vật
- Chơi trò mèo đuổi chuột
- Chơi trò Sumo
- Đua xe điều khiển và tự động có tránh chướng ngại vật
- Đá bóng điều khiển từ xa
- Giải bài toán Mê cung sử dụng cảm biến dò line và siêu âm

## Trải nghiệm đầu tiên về Easybot

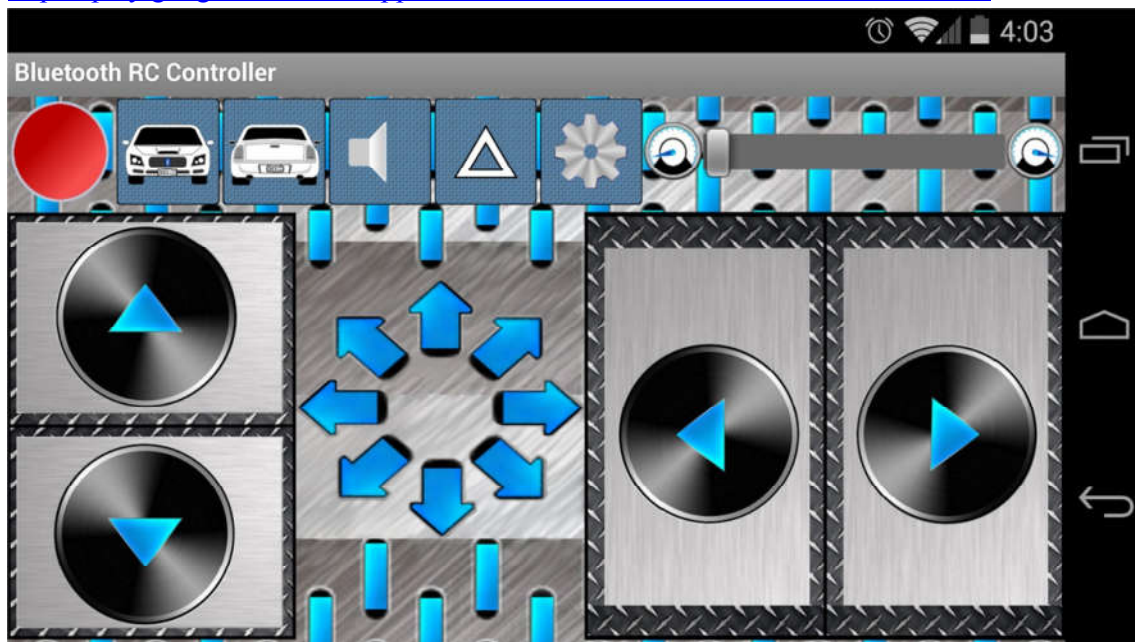


Sau khi mở hộp, theo hướng dẫn online và tài liệu đi kèm, các bạn sẽ dễ dàng tự tay lắp hoàn chỉnh 1 robot và đấu nối mạch điều khiển.

### Điều khiển qua SmartPhone (Android)

Tải App điều khiển robot theo Link sau:

<https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller>



- Bật nguồn Robot
- Bật Bluetooth trên điện thoại
- Scan thiết bị Bluetooth có tên là “Wibotxxxx” / “Racebotxxxx”/”Battlebotxxxx” tùy loại Robot sử dụng
- Mã số sau tên thiết bị “xxxx” là password phải nhập để kết nối
- Vào App Remote Control -> Setting -> Chọn device và điều khiển

## PHẦN 2 – LẬP TRÌNH



### SCRATCH

**Scratch** là tên gọi một ngôn ngữ lập trình đồ họa trực quan sinh động do trung tâm Media Lab thuộc trường đại học MIT (Hoa Kỳ) sáng tạo ra với mục tiêu giúp trẻ em có thể học lập trình một cách dễ dàng.

Khi lập trình Scratch thay vì phải gõ những dòng lệnh với các quy tắc ràng buộc phức tạp thì người học chỉ việc kéo và thả các khối lệnh, lắp ghép các khối lệnh lại thành 1 kịch bản điều khiển nhân vật trên màn hình.

Theo chính tác giả Scratch - Giáo sư Mitchel Resnick đã chia sẻ như sau:

*"Khi học lập trình Scratch, trẻ em học được những nguyên lý cơ bản của việc thiết kế, học được cách thử nghiệm ý tưởng mới, học được cách phân chia ý tưởng phức tạp thành những phần việc đơn giản, học được cách hợp tác với người khác để thực hiện dự án, học được cách tìm và sửa lỗi khi kết quả không được như ý, tập được tính kiên trì khi đối mặt với khó khăn. Ngày nay, đó không chỉ là những kỹ năng cần thiết cho việc lập trình, mà còn cần thiết cho nhiều hoạt động khác."*

Ngày nay Scratch được sử dụng phổ biến và rộng rãi để đào tạo lập trình rèn luyện tư duy cho trẻ em trên khắp thế giới với cộng đồng người dạy và học đông đảo, tài liệu và bài tập cực kỳ phong phú. Do đó, những người triển khai dự án STEMbot đã quyết định lựa chọn Scratch làm ngôn ngữ để lập trình cho Robot thông qua các công cụ thông dịch ra mã nguồn Arduino.

## mBlock



**mBlock** là phần mềm chuyên dụng để lập trình ngôn ngữ Scratch cho Robot và các thiết bị điện tử khác hướng đến giáo dục và các ứng dụng sáng chế cho trẻ em. mBlock được hãng MakeBlocks (Trung Quốc) phát triển dựa trên mã nguồn mở của phần mềm SCRATCH 2.0 do MIT phát hành miễn phí. Do đó mBlock kế thừa đầy đủ các thành phần của Scratch 2.0 và thêm vào đó là Công cụ thông dịch và biên dịch ra mã Arduino C để có thể nạp vào các phần cứng nền tảng Arduino.

mBlock được thiết kế chủ đạo dành cho các Robot của hãng Makeblocks, tuy nhiên, là một phần mềm nguồn mở, mBlock cho phép mở rộng các Extension để có thể làm việc với tất cả các phần cứng của bên thứ 3 thông qua việc Upload các file mở rộng do các bên thứ 3 phát triển nhằm tích hợp thiết bị của mình có thể hoạt động với mBlock, ví dụ các sản phẩm phổ biến như các board Arduino Grove của SeeedStudio, LittleBit, vv... Do đó chúng tôi đã lựa chọn mBlock để lập trình cho các Robot và phần cứng thuộc dự án STEMbot.

## **Phương pháp học lập trình đề cập trong tài liệu này**

Tài liệu này được biên soạn hướng vào đối tượng mới bắt đầu và chưa từng làm quen với ngôn ngữ Scratch. Đây không phải là tài liệu để học lập trình Scratch, tuy nhiên chúng tôi sẽ sắp xếp các bài học nhỏ hướng thẳng đến ứng dụng Robot qua đó giúp các bạn trẻ dần khám phá về ngôn ngữ Scratch và các thành phần chính cấu thành nên chương trình Scratch. Hi vọng với cách học này, các bạn sẽ nắm được phần nào ngôn ngữ lập trình Scratch thông qua các bài học với Robot và sau đó tự tìm hiểu chuyên sâu và làm các chương trình phức tạp hơn với ngôn ngữ Scratch 2.0 một cách nhanh chóng.

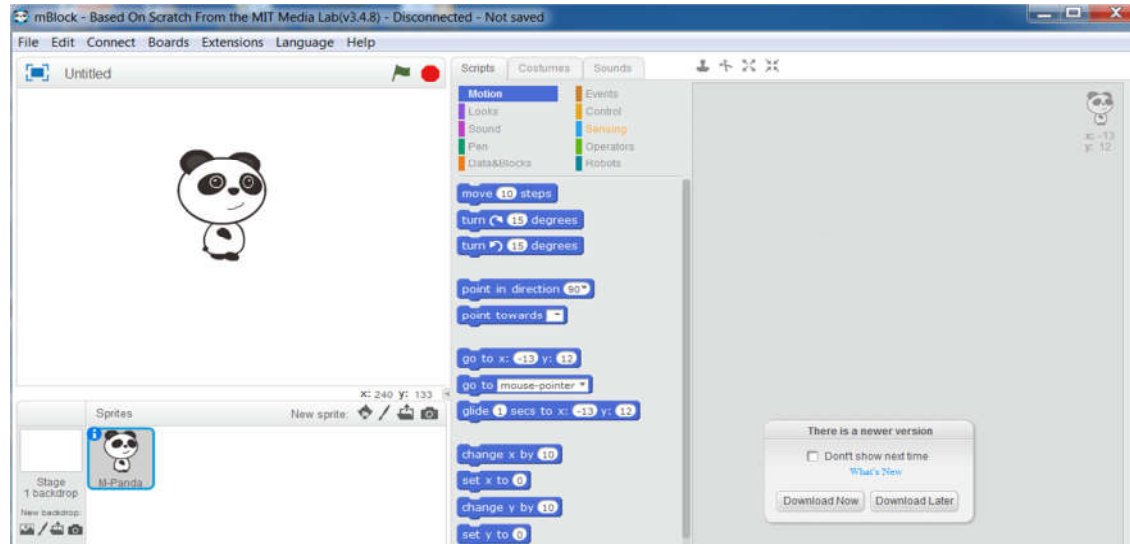
Ngôn ngữ block: chúng tôi tạo ra các block cho Easybot và sử dụng tiếng Anh thay vì tiếng Việt cho các block. Chúng tôi quan niệm rằng tiếng Anh là điều kiện quan trọng để các em bước chân vào thế giới điện toán, vì vậy sử dụng và làm quen tiếng Anh đơn giản trong Scratch có tác dụng rất quan trọng, tuy sẽ khó khăn ban đầu nhưng các em sẽ được làm quen dần với các cấu trúc chương trình sử dụng tiếng Anh (bất cứ ngôn ngữ lập trình nào cũng sử dụng tiếng Anh để xây dựng), cũng là cơ hội để học thêm tiếng Anh một cách trực quan.

## Cài đặt mBlock

Download phần mềm mBlock

<http://www.mblock.cc/software/mblock/mblock3/>

Mở phần mềm mBlock sau khi cài đặt

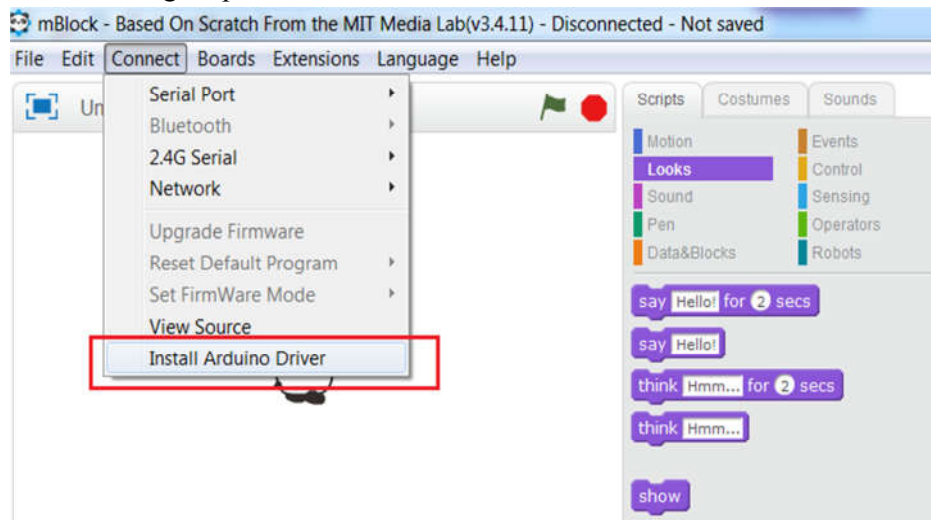


## Kết nối với Robot

Robot kết nối với máy tính bằng 2 cách:

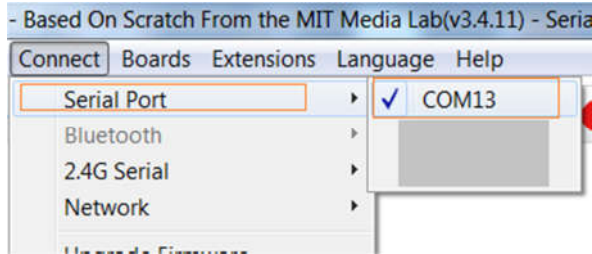
Cáp USB và Bluetooth

- Đối với USB, ta cần cài Driver để máy tính giao tiếp được với Robot. Driver ở đây dựa trên dòng chip USBSerial tên là CH340

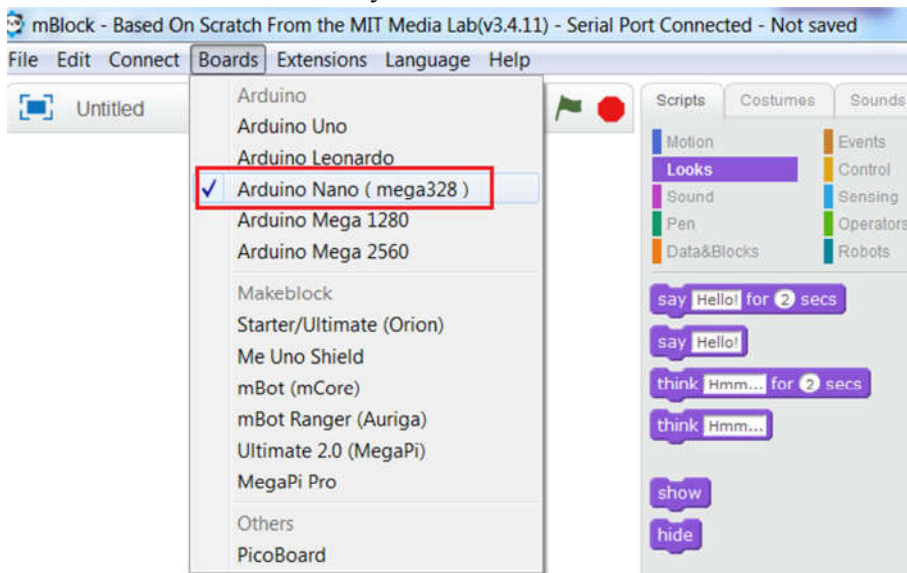




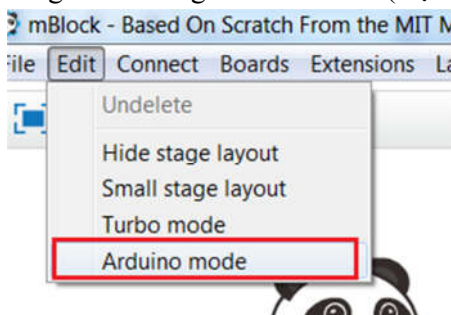
- Sau khi cài đặt Driver, ta tiến hành kết nối với robot qua dây cable USB mini



- Chọn Board là Arduino nano (mega328): đây là board mạch phiên bản Arduino điều khiển chính cho các robot Easybot

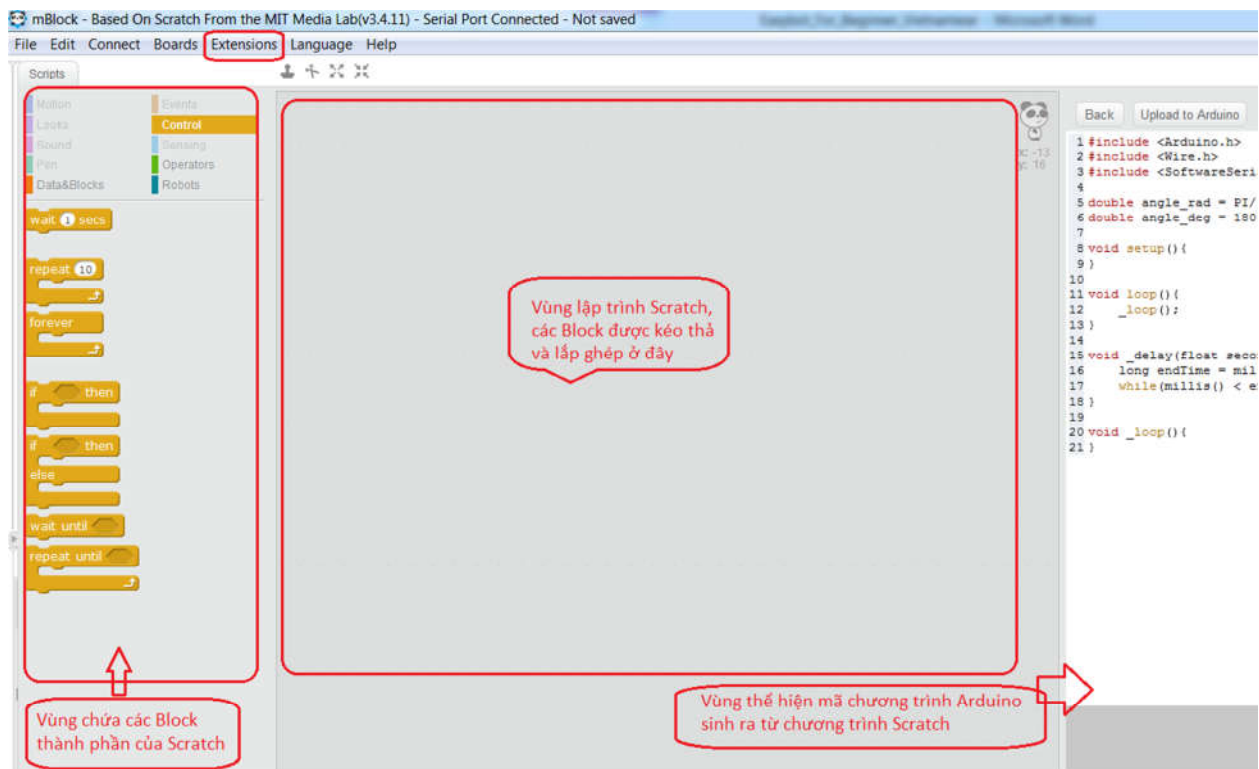


- Để đơn giản cho người mới bắt đầu, ta sẽ đi thẳng vào chế độ lập trình và biên dịch sang mã chương trình Arduino (Gọi là Arduino Mode) .



- Các bài tập trong tài liệu này, khi thực hành, các bạn chú ý vào Edit chọn chế độ Arduino mode, chọn Board: Arduino Nano, Và Connect: Serial -> Port “n”*





Ở chế độ Arduino mode, phần mềm Scratch đã được đơn giản hóa đi rất nhiều, các thành phần chính thường xuyên thao tác để lập trình Scratch cho robot gồm:

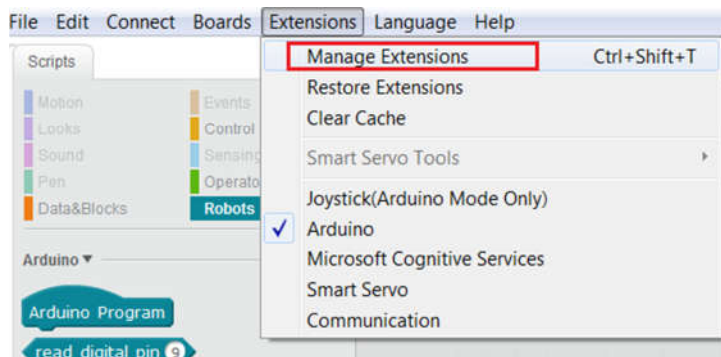
- **Bảng Scripts:** chứa các thành phần tạo nên 1 chương trình Scratch đó là các blocks. Có nhiều loại Block đại diện các chức năng khác nhau, tuy nhiên trong chế độ Arduino Mode, các blocks chuyên về đồ họa và thao tác máy tính được ẩn đi, chỉ còn 4 loại block cần thiết cho việc lập trình Robot là có thể sử dụng.
- **Thẻ Extension:** dùng để cài đặt các Blocks hỗ trợ các board mạch/Robot do các nhà sản xuất khác nhau phát hành (từ nền tảng Arduino) để sử dụng trong phần mềm mBlock.
- **Vùng lập trình Scratch:** Các block được kéo thả, sắp xếp và lắp ghép trong vùng này để tạo thành 1 chương trình Scratch hoàn chỉnh và chạy được.
- **Vùng Arduino Code:** Vùng này thể hiện mã Arduino được tự động sinh ra tương ứng từ chương trình Scratch. Người học tiến hành Upload chương trình vào trong Robot để robot hoạt động theo chương trình Scratch đã viết, hệ thống biên dịch và nạp hoàn toàn theo cơ chế nguyên bản của Arduino.

## Các loại Block thường dùng để lập trình Scratch cho Robot (Arduino mode)



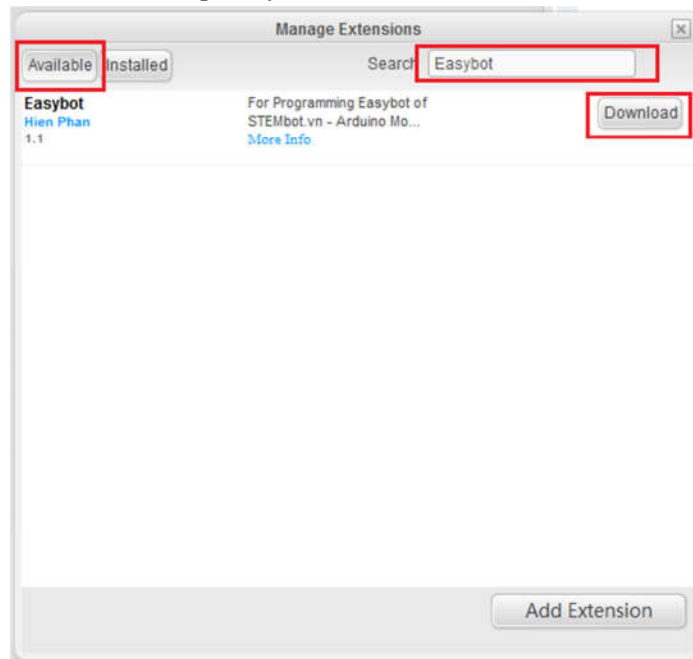
- **Blocks Control:** dùng để điều khiển chương trình, các vòng lặp, rẽ nhánh, chờ....
- **Blocks operators:** chứa các block toán tử thực hiện các phép tính toán học, logic và chuỗi ký tự
- **Blocks Data & Block:** chứa các block là biến dữ liệu và Block do người viết tự tạo từ các block khác (như chương trình con)
- **Blocks Robots:** là các blocks thực hiện các chức năng của robot hay các thiết bị phần cứng khác. Chú ý, mặc định khi mới cài đặt mBlock, chỉ có Arduino và các robot của hãng Makeblock được tích hợp sẵn, để cài đặt block cho Easybot, ta cần làm tiếp như sau:

➤ Vào Manage Extension



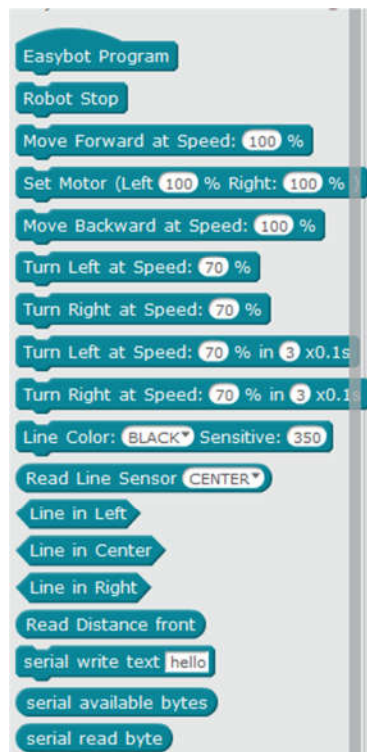
Extension chứa các file cho phép tích hợp các robot bất kỳ vào mBlock để lập trình với ngôn ngữ Scratch. Easybot là extension do STEMbot biên soạn.

- Hiện ra cửa sổ quản lý Extension:



Vào thẻ “Available”, chọn tìm kiếm: gõ vào Easybot, và bấm nút Download

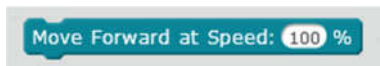
Sau khi cài đặt thành công, các block liên quan đến Robot sẽ xuất hiện:



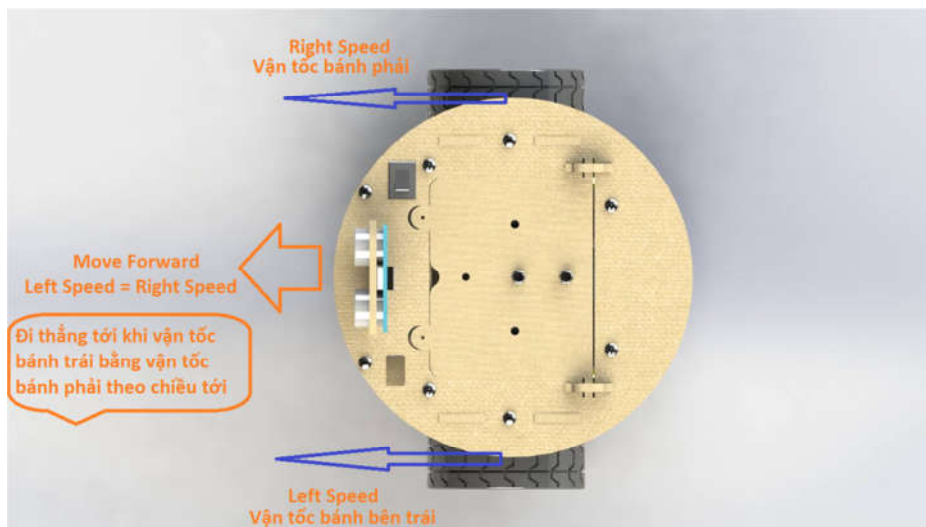
## Lesson 1 Di chuyển với block *Moving*

Nguyên lý di chuyển của Robot Easybot là nguyên lý Differential Drive: lái dựa vào sai khác vận tốc 2 bánh

### 1. Đi thẳng (Move Forward at speed: %)

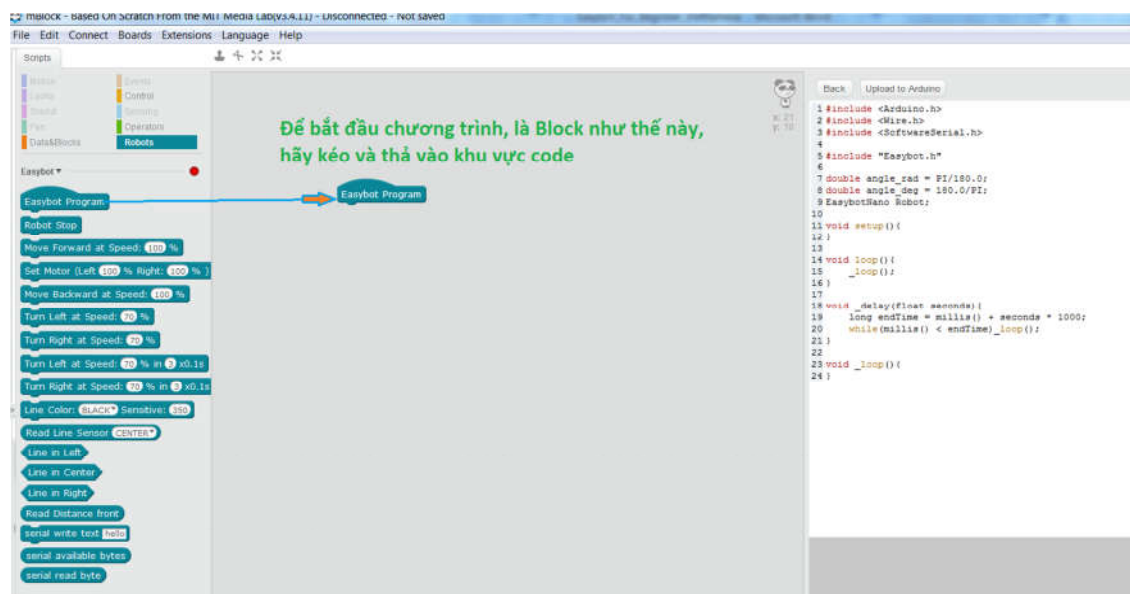


Block điều khiển robot đi thẳng với vận tốc có thể chọn từ 0 – 100%

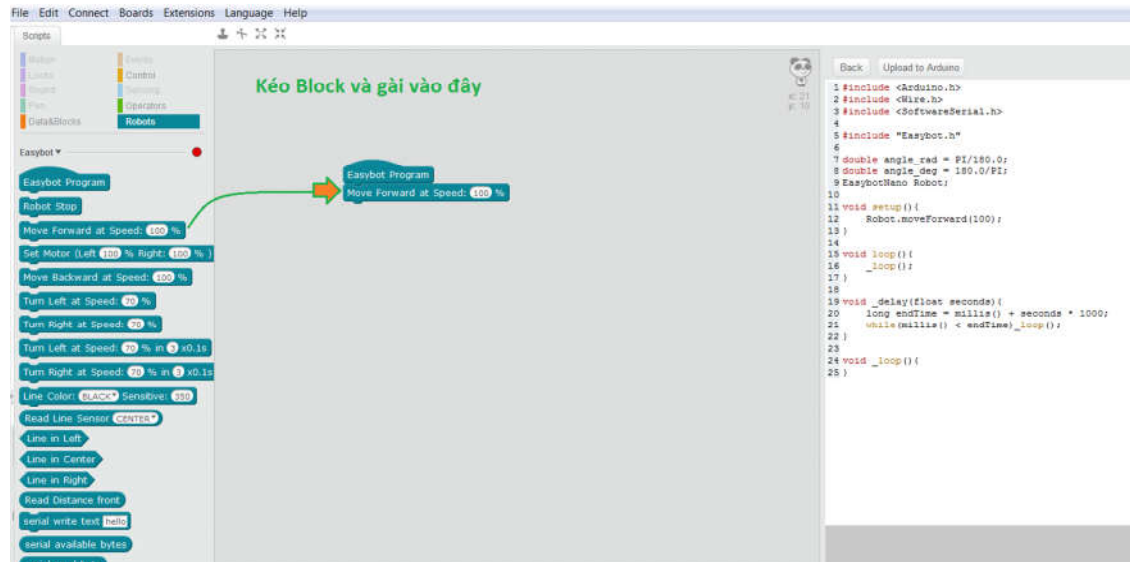


Nào ta hãy bắt đầu thực hành:

Bước 1: Kéo thả block để bắt đầu chương trình, bất cứ chương trình Scratch ở chế độ Arduino mode đều phải khởi đầu bằng block sau:

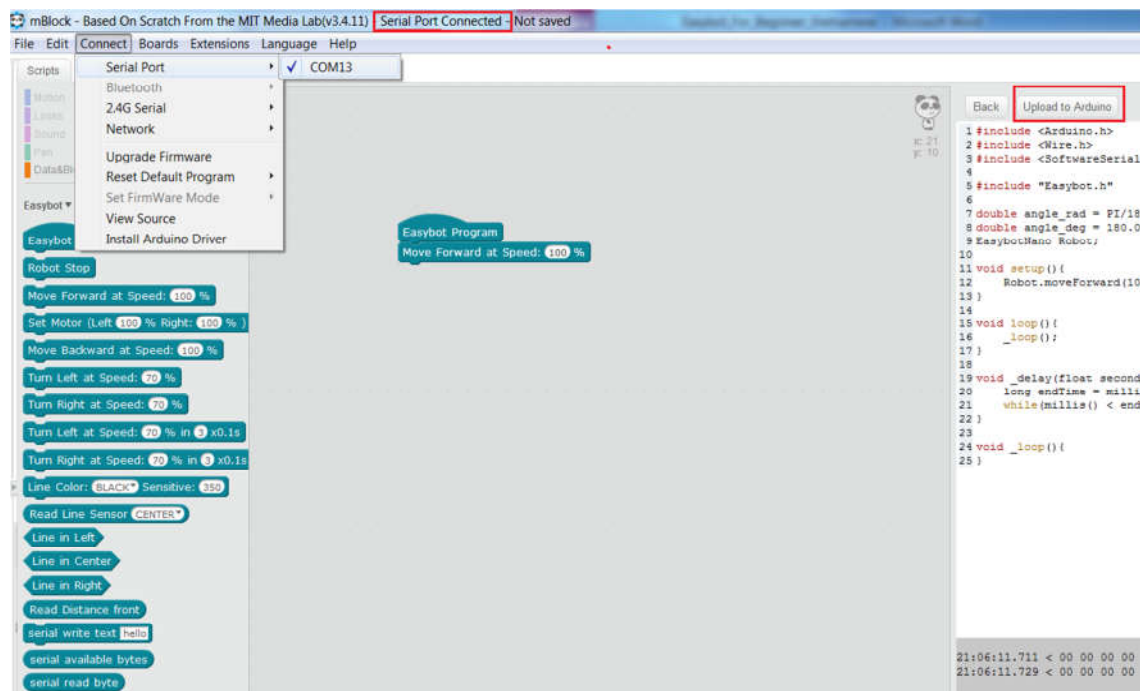


Bước 2: Kéo block “Move Forward” ghép vào Block khởi đầu.

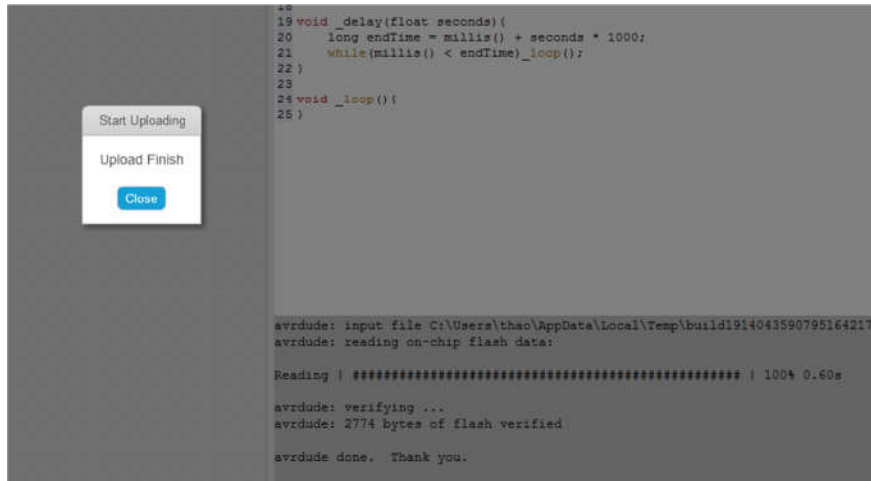


Kết nối dây USB với Robot vào máy tính và chọn cổng Serial để kết nối, khi kết nối thành công phía trên thanh trạng thái sẽ hiển thị là **Serial Port Connected**.

->Đề nạp code vào Robot, bấm vào **Upload to Arduino**



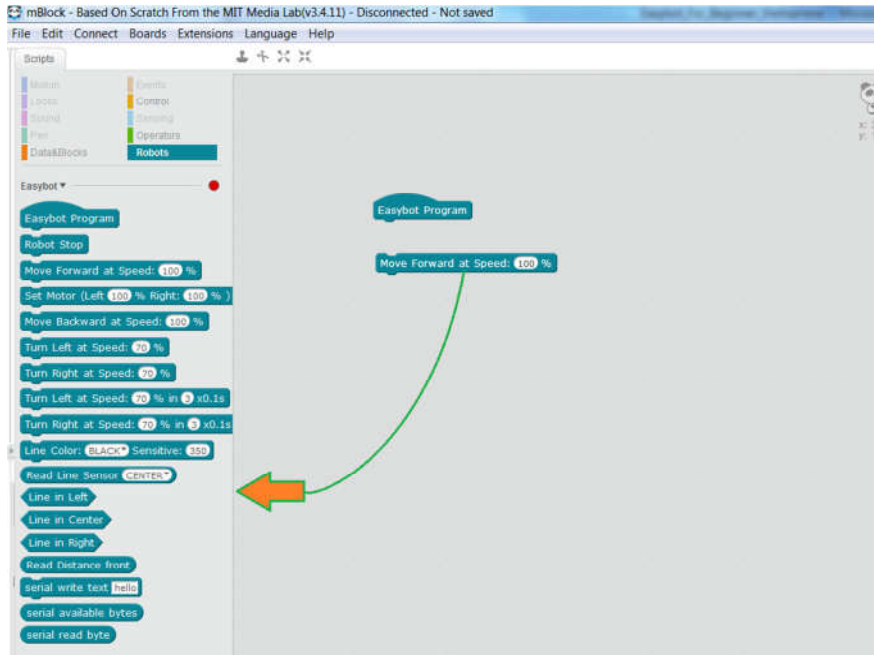
Nạp chương trình vào robot thành công sẽ thấy thông báo như sau



Bật nguồn Robot và quan sát Robot chạy theo chiều tiến.

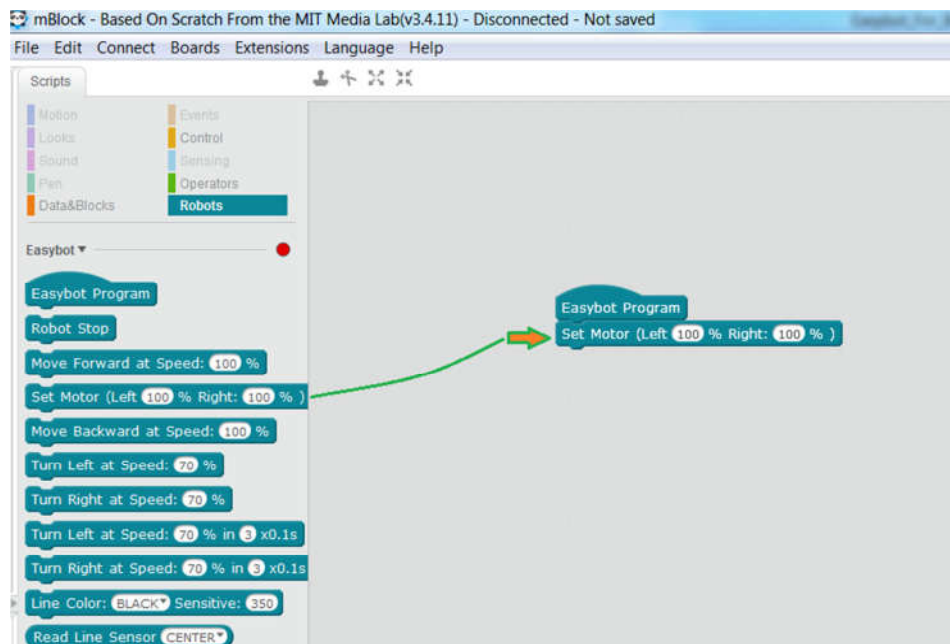
Thực tế, 2 động cơ không có vận tốc giống nhau tuyệt đối vì vậy robot sẽ có xu hướng lệch ít sang bên trái hay bên phải tùy động cơ bên nào chạy nhanh hơn. Nếu yêu cầu đi thẳng với độ chính xác cao, ta cần điều chỉnh lại vận tốc 2 bánh xe khác nhau, hãy sử dụng Block khác để đặt tốc độ từng motor.

*Trước tiên để sửa chương trình ta cần gỡ bỏ block đã gắn như sau: kéo block cần gỡ bỏ trả về lại khu vực chứa các Blocks*

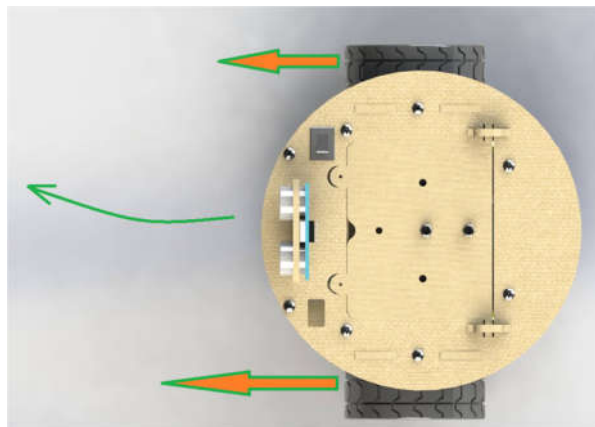


Kéo thả block “Set Motor” gắn vào chương trình, lúc này ta có thể set tốc độ bánh trái và bánh phải khác nhau.





Nguyên tắc là: Nếu bánh bên phải chậm hơn bánh bên trái 1 khoảng thì Robot sẽ có xu hướng đi lệch về phía bánh bên phải và ngược lại. Có nghĩa là bánh bên nào chậm hơn, robot sẽ lệch về bên đó.



Các em tùy ý điều chỉnh vận tốc từng bên bằng cách thay đổi % của Block Set Motor



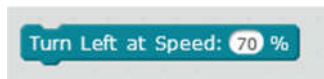
ví dụ, nếu robot có xu hướng lệch về bên phải, tức là bên trái đang nhanh hơn, ta tiến hành giảm bên trái xuống 95% và bên phải giữ nguyên 100%.

Rồi nạp vào robot như đã thực hiện ở trên, và quan sát lại xem robot đã đi thẳng chưa, nếu chưa thì tiếp tục chỉnh. Chú ý để đơn giản trong tài liệu chúng tôi vẫn sử dụng Block Move Forward cho hành động đi thẳng.



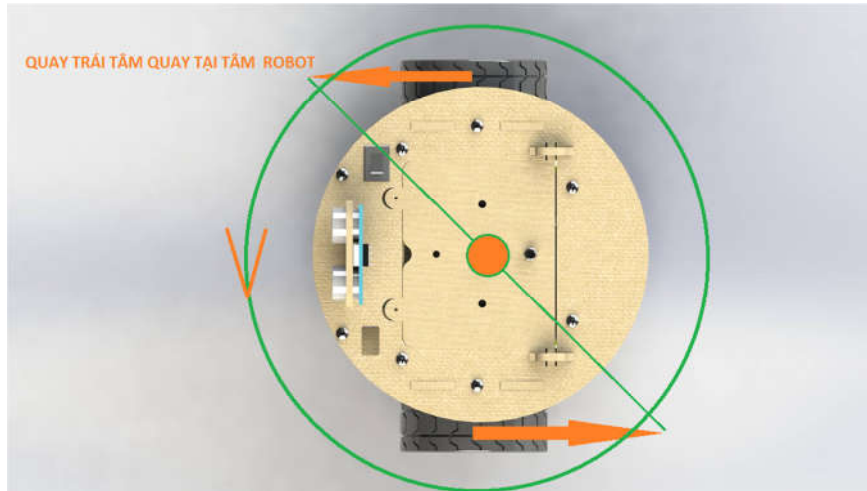
## 2. Lệnh Block quay vòng.

### ➤ Quay trái với vận tốc ...%



vận tốc bánh phải theo chiều tiến sẽ là 70%, vận tốc bánh trái theo chiều lùi là -70%

Tâm quay tại tâm robot.



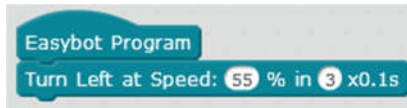
### ➤ Quay trái với góc quay định lượng bằng thời gian



Quay trái với vận tốc mỗi bánh là 70%, trong khoảng thời gian  $3 \times 0.1s = 0.3$  Block này được dùng để quay trái với 1 góc cho trước và dừng, góc quay ở đây định lượng bằng thời gian quay với vận tốc đặt trước.

### THỰC HÀNH

Thử viết đoạn code để Robot quay 1 góc 90 độ:

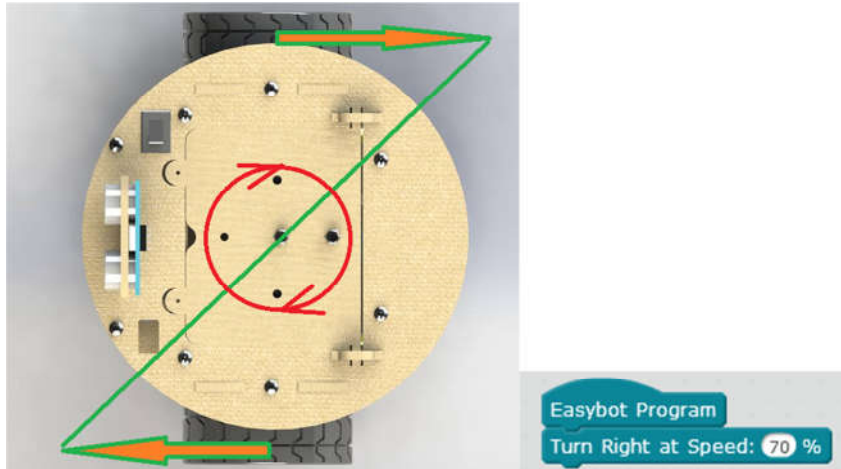


Nạp vào Robot, bật nguồn và quan sát. Nếu Robot quay quá góc quay hoặc quay không đủ góc quay, các bạn có thể điều chỉnh thời gian và vận tốc cho phù hợp đến khi quay đủ góc 90 độ.

➤ Quay phải với vận tốc ...%

Turn Right at Speed: 70 %

Tương tự như quay trái, khi quay phải bánh bên phải quay lùi và bánh bên trái quay tiến với cùng vận tốc tâm quay tại tâm Robot.



➤ Quay phải với góc quay bằng định lượng thời gian

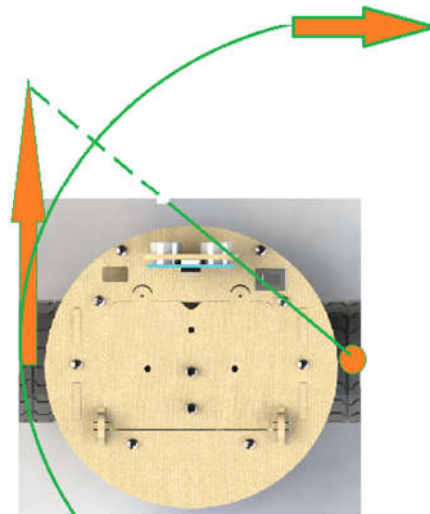
Easybot Program  
Turn Right at Speed: 55 % in 3 x0.1s

Quay phải với vận tốc 55% trong vòng  $3 \times 0.1 = 0.3s$

Hãy nạp chương trình và quan sát robot quay có đủ góc đã định hay chưa, nếu chưa thì tiếp tục điều chỉnh thời gian và vận tốc cho phù hợp.

Vd: Giả sử ở vận tốc 55% và thời gian 0.3s, robot quay phải 90 độ thì dừng.

➤ Quay phải với tâm quay tại bánh xe bên phải

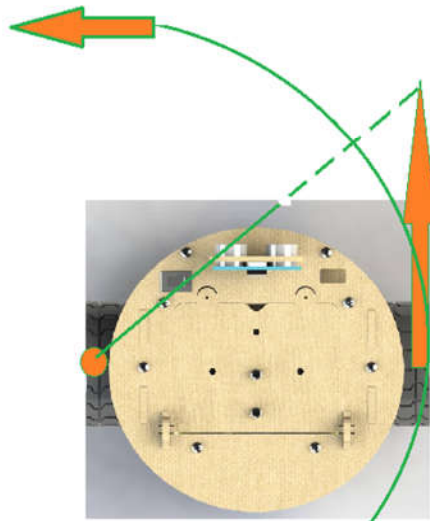


QUAY PHẢI VỚI TÂM QUAY TẠI BÁNH XE BÊN PHẢI

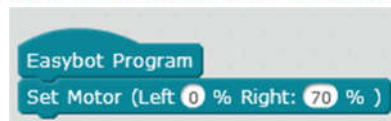
Easybot Program  
Set Motor (Left 100 % Right: 0 % )

Bánh phải đứng yên, bánh trái quay với vận tốc V%. Các bạn thử upload chương trình để quan sát.

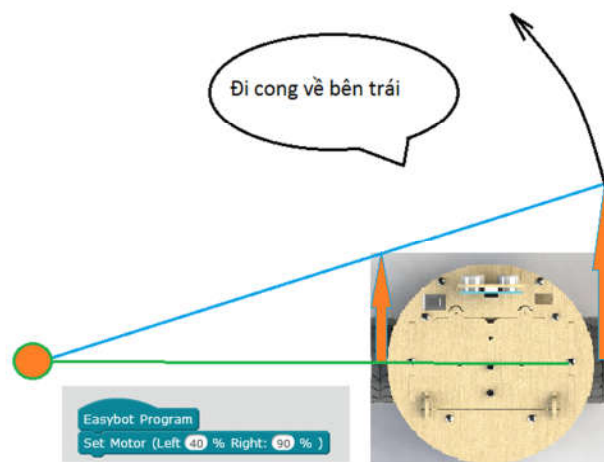
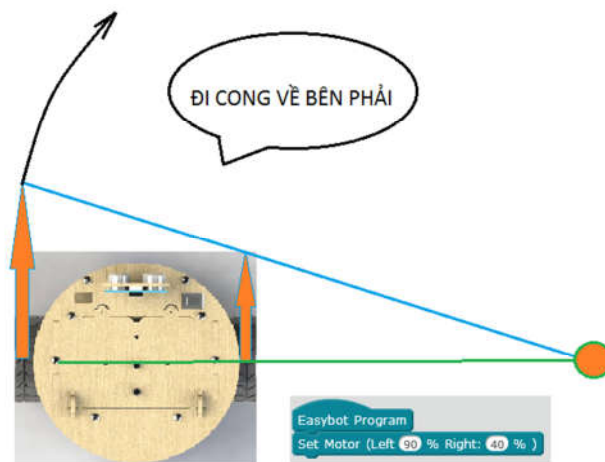
- Quay trái với tâm quay tại bánh xe trái



ROBOT QUAY TRÁI VỚI TÂM QUAY TẠI BẮNH XE TRÁI

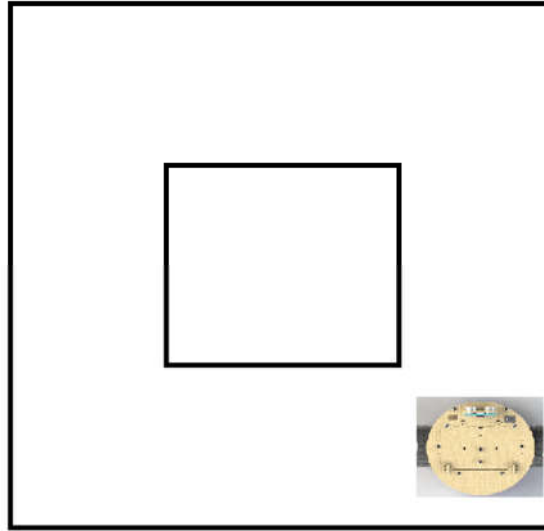


- Quay robot với bán kính quay vòng lớn, tâm quay nằm ngoài



3. Bài tập Thực hành Lesson 1 (có thể tổ chức như bài thi nhỏ giữa các nhóm)

- **Bài tập 1:** Vẽ 1 cung đường **hình vuông** tùy ý, hãy viết chương trình cho robot đi theo quỹ đạo hình vuông đã vẽ (không được chạm vạch)

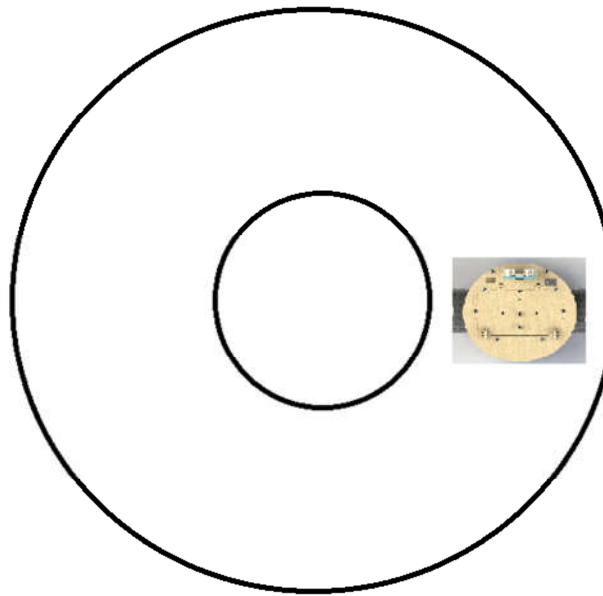


Gợi ý chương trình và các cách viết 1 chương trình: tuần tự, nhân bản và vòng lặp

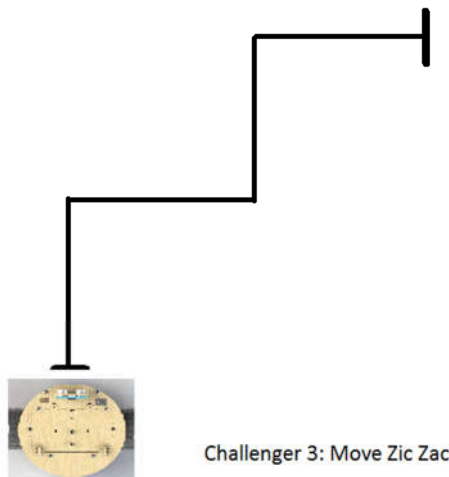
Chương trình tối ưu: sử dụng vòng lặp, khi đó ta dễ dàng tinh chỉnh

Các động tác (quyết định độ dài cạnh của hình vuông, thời gian quay để quay 90 độ)

- **Bài tập 2:** Vẽ 1 cung đường **hình tròn**, viết chương trình cho robot đi theo hình tròn trong cung đường đã vẽ (không được chạm vạch)

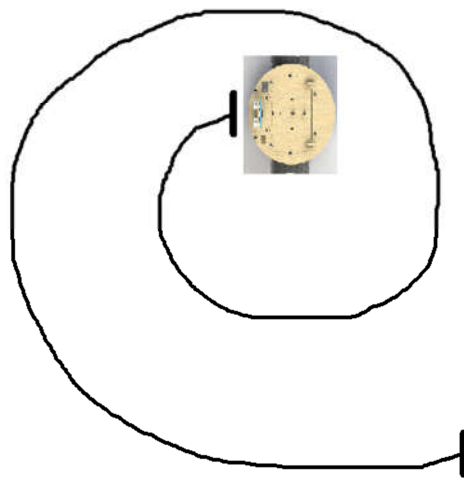


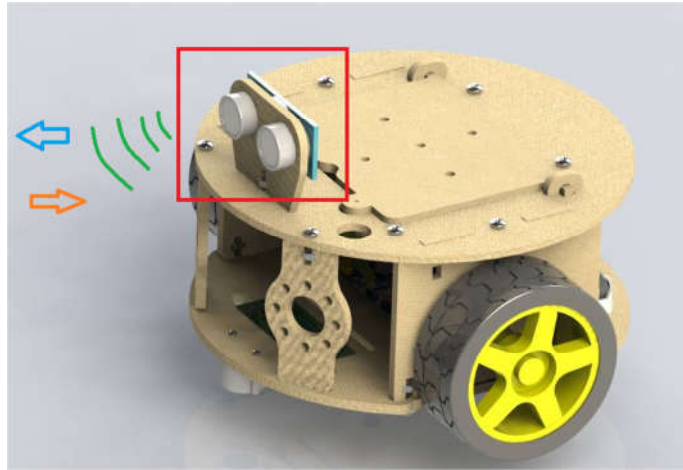
- **Bài tập 3:** Viết chương trình cho robot đi theo đường Zic Zac



Challenger 3: Move Zic Zac

- **Bài tập 4:** Viết chương trình cho robot đi theo đường xoắn ốc



**LESSION 2 SỬ DỤNG CẢM BIẾN ĐO KHOẢNG CÁCH SIÊU ÂM**

Cảm biến là các thiết bị điện tử sử dụng để chuyển các tín hiệu vật lý trong thế giới thực sang tín hiệu điện mà máy tính có thể tiếp nhận và xử lý.

Trong bài học này, chúng ta sẽ tìm hiểu về cảm biến siêu âm SRF04 được trang bị kèm theo Easybot.

**Nguyên lý hoạt động:**

Loài dơi xác định các vật thể trong môi trường bằng cách phát sóng siêu âm bằng miệng và cảm nhận sóng âm phản xạ trở lại ở tai. Bởi vậy nếu bị mất mắt thì dơi vẫn bay được và tránh vật cản nhưng nếu bị mất tai hoặc bị mất miệng thì chắc chắn loài dơi sẽ va vào các vật thể.

Cảm biến SRF04 cũng tương tự, với độ ồn âm thanh trong môi trường không khí là không đổi, việc xác định thời gian từ lúc phát đến lúc phản xạ ta sẽ tính toán được khoảng cách.

**Tầm phát hiện tối đa của SRF04 trong tài liệu này là 50cm.**




Thời gian từ lúc phát đến lúc nhận đo được là T

Khoảng cách D đến vật được tính như sau:  $D = T/2 * V$  với  $V = 330\text{m/s}$ .



**Ứng dụng:** cảm biến đo khoảng cách siêu âm được sử dụng rộng rãi trong các thiết bị đo xa, máy bắn tốc độ, đo chiều cao, đo chiều sâu, các loại radar hàng hải để giám sát đáy biển, phát hiện tàu ngầm....

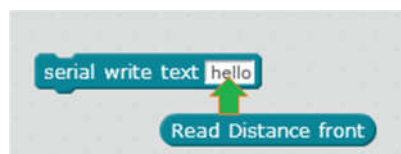
Với Easybot, việc tính toán đã được tích hợp sẵn trong chương trình, Robot có thể đo khoảng cách phía trước một cách dễ dàng bằng cách đơn giản là truy xuất block:  (đọc khoảng cách phía trước).

Các bạn chú ý: đến phần này, ta sẽ thấy hình dạng Block đã thay đổi so với các block trước đã học. Block này có chức năng là đọc về dữ liệu nên nó khác các Block về lệnh thực hiện như di chuyển. Có nghĩa là nó phải gắn vào bên trong một block lệnh nào đó chứ không ghép với các block về lệnh được.

**Bài tập 5:** viết chương trình truyền dữ liệu về khoảng cách lên máy tính để xem:

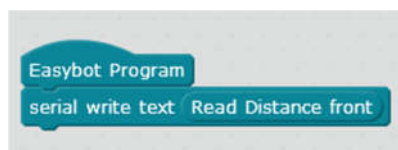
#### Các kiến thức cần trang bị

Blocks	Hướng dẫn	Ví dụ
 Thuộc nhóm <b>robots-Easybot</b>	Đây là kiểu block input (nhập dữ liệu đầu vào) trả về 1 giá trị là số tự nhiên. Block này dùng để nhập vào dữ liệu là khoảng cách đo từ cảm biến siêu âm.	 Gửi đoạn văn bản với đầu vào là giá trị cảm biến đo khoảng cách lên máy tính
 Thuộc nhóm <b>robots-Easybot</b>	Block này thực thi lệnh gửi 1 gói tin dạng văn bản lên trên máy tính qua cổng Serial (cổng COM). Dữ liệu đầu vào có thể là văn bản gõ từ bàn phím hoặc số liệu từ các block kiểu input khác.	 Gửi gói tin Hello lên trên máy tính.
 Vòng lặp vĩnh viễn Thuộc nhóm <b>Control</b>	Sử dụng khi cần lặp đi lặp lại một hay nhiều việc một cách liên tục không dừng.	 Robot lắc qua lắc lại liên tục mãi mãi
 Block nối chuỗi văn bản Thuộc nhóm <b>Block Operator</b>	Block này thuộc kiểu block input, tức là bản thân nó cung cấp đầu vào cho block khác nhưng có chức năng là nối 2 chuỗi văn bản với nhau lại. Thường dùng khi muốn ghép nhiều chuỗi ký tự vào cùng 1 gói tin trước khi truyền.	 Gửi gói tin “hello world” lên máy tính qua cổng Serial



Block **Serial write text**: [] là block thực hiện lệnh truyền gói tin văn bản kiểu nối tiếp (Serial) lên máy tính. Bây giờ ta ghép Block **Read Distance front** vào phần nội dung cần truyền, như vậy dữ liệu đọc từ cảm biến đo khoảng cách sẽ chuyển lên máy tính và ta có thể xem nó trong phần mềm mBlock.

Ta có thể thử hoàn thành nhanh chóng chương trình như sau để kết quả đo khoảng cách hiển thị trên màn hình máy tính.



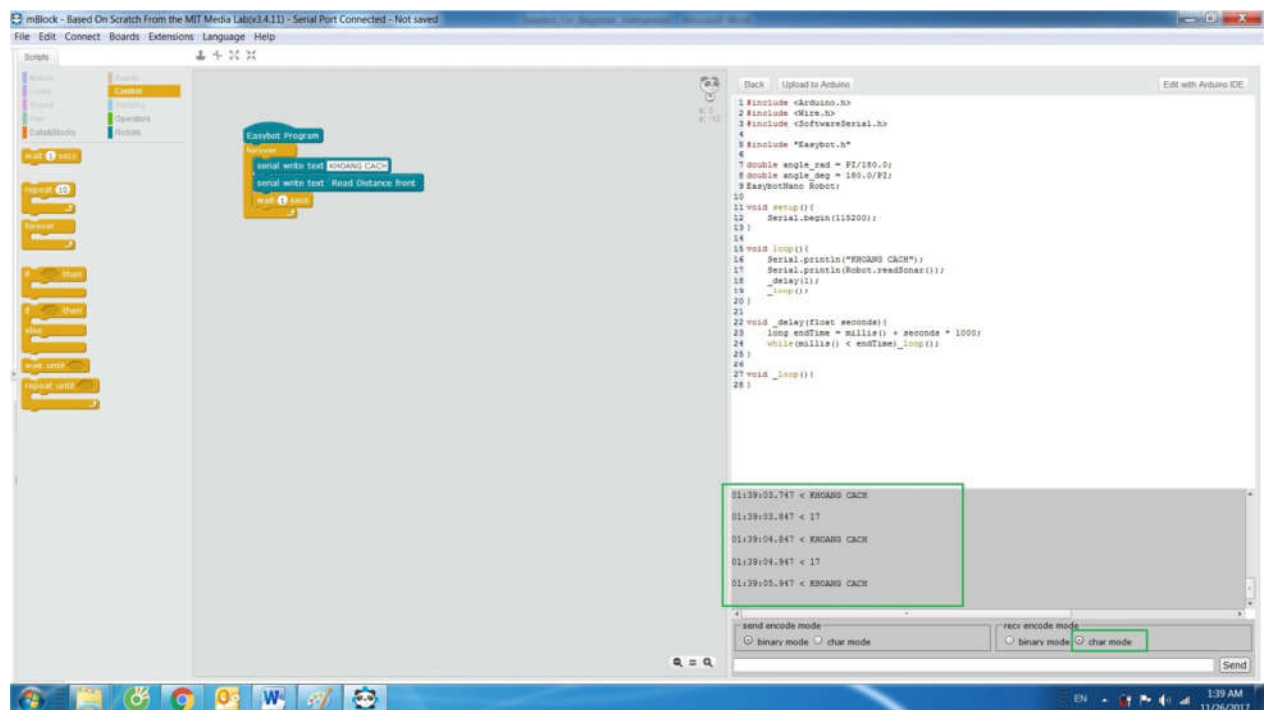
Tuy nhiên, nếu viết như trên thì khi bắt đầu chương trình ta chỉ đọc được dữ liệu đo khoảng cách một lần rồi kết thúc. Yêu cầu là phải đọc liên tục khoảng cách, lúc này ta dùng vòng lặp **Forever** (nằm trong nhóm block **Control**), và trước khi truyền kết quả đo khoảng cách ta truyền các ký tự “KHOANG CACH” để diễn đạt rõ ràng hơn kết quả trên máy tính.

Block **Wait [1] secs** để đảm bảo ta có thể quan sát được kết quả khi cứ 1 giây dữ liệu sẽ truyền 1 lần, chú ý nếu không có thì Robot sẽ gửi dữ liệu liên tục với tốc độ rất cao và khó có thể quan sát được.



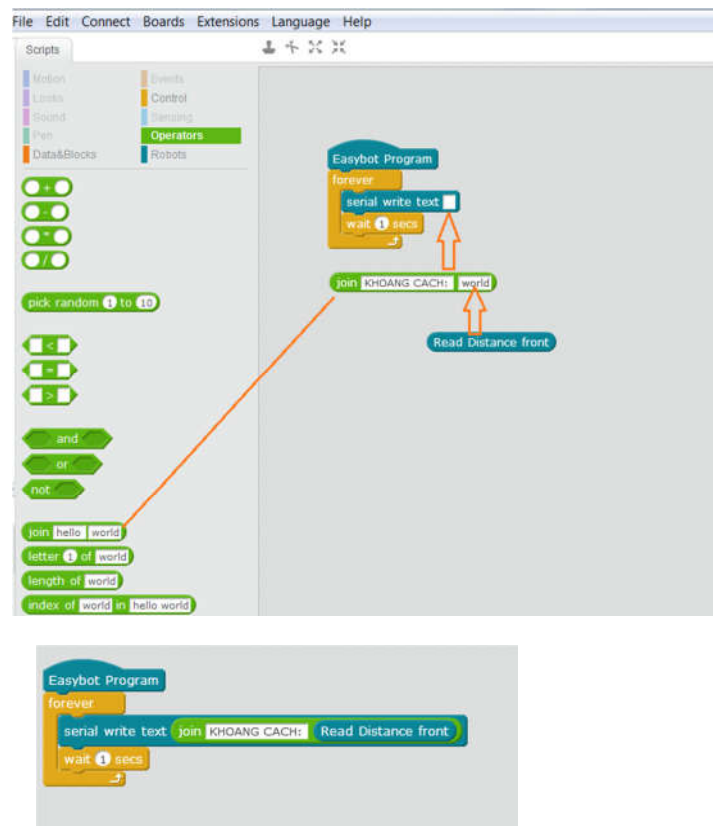
Nào giờ Upload chương trình vào Robot, Chú ý sau khi Upload, vào **Connect** kết nối **Serial** với **Robot** lại để quan sát kết quả nhé. (Bởi vì ngay sau khi upload chương trình, phần mềm sẽ tự động ngắt kết nối Serial với Robot)

Dữ liệu sẽ gửi qua cổng Serial được hiển thị ở góc phải màn hình, chú ý chọn **Receive Encode Mode** là **Char mode** để dữ liệu hiển thị theo kiểu văn bản ta có thể đọc được thay vì mã nhị phân.

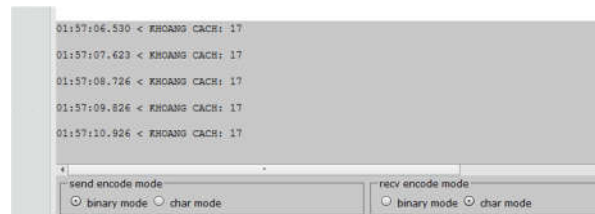


Để ý rằng, với chương trình như trên Robot sẽ gửi câu nói: “KHOANG CACH” sau đó xuống dòng, và gửi tiếp số liệu đo khoảng cách. Cách viết như trên có thể dùng được nhưng chưa khoa học, vậy làm sao để Robot có thể truyền ký tự KHOANG CACH và sau đó tiếp theo là số liệu đo khoảng cách trên cùng 1 lần truyền ?

Ta sử dụng Toán tử (operator) **Join** để nối 2 chuỗi ký tự với nhau trước khi cho vào lệnh truyền:




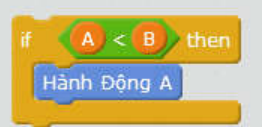
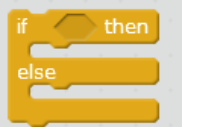


Tiến hành Upload chương trình vào robot và quan sát lại nhé:



**Bài tập 6**

Ứng dụng cảm biến siêu âm để tự động chuyển hướng khi có vật cản.

**Kiến thức cần nắm**

<b>Blocks</b>	<b>Hướng dẫn</b>	<b>Ví dụ</b>
 If---then Thuộc nhóm Controls	Cấu trúc rẽ nhánh If (nếu) điều kiện đầu vào là đúng thì thực hiện lệnh, nếu sai thì bỏ qua lệnh. Điều kiện đầu vào ở đây là các block input (đầu vào) kiểu Logic, nó chỉ trả về kết quả là đúng hoặc sai.	 Nếu $A < B$ (là đúng) thì thực hiện hành động A, nếu không thì bỏ qua
 If—Then—Else Thuộc nhóm Controls	Cấu trúc rẽ nhánh if.. else.. Tương tự như If..., tuy nhiên khi điều kiện có kết quả là sai thì nó sẽ thực hiện lệnh khác	 Nếu $A < B$ (là đúng) thì thực hiện hành động A, Còn không ( $A > B$ hoặc $A = B$ ) thì thực hiện hành động B
 Toán tử so sánh logic Thuộc nhóm Operator	Các block này là kiểu block Logic input (cung cấp đầu vào Logic) có tác dụng so sánh 2 dữ liệu và trả kết quả là đúng hay sai. Sử dụng để nhập đầu vào cho các block yêu cầu đầu vào là logic input	

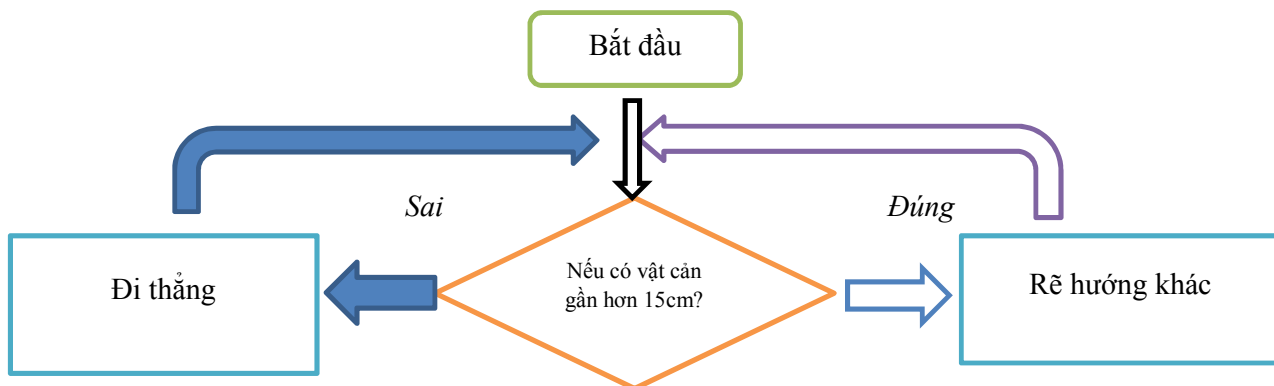
Hướng suy nghĩ để giải quyết vấn đề:

Lập lại:

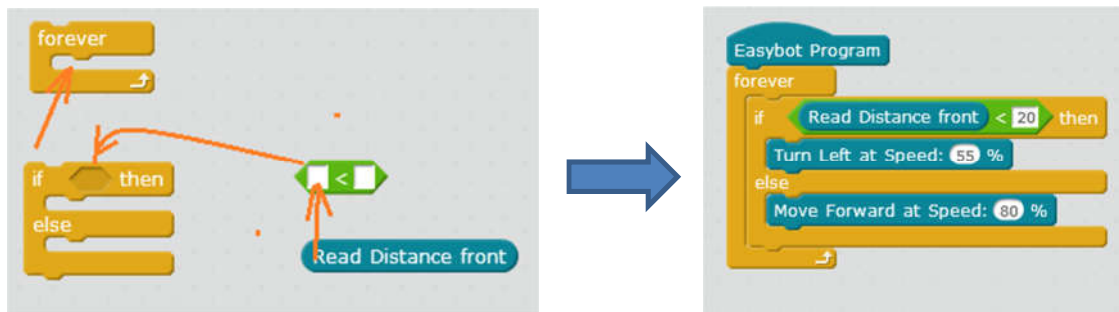
-----Nếu gặp phải vật cản: thì rẽ sang hướng khác

-----Còn không: thì đi thẳng

Hướng suy nghĩ Logic như trên được gọi là **thuật toán**, là tư duy quan trọng nhất mà mọi người cần trang bị, tư duy logic, vận dụng toán học để giải quyết các vấn đề là một trong những mục tiêu quan trọng nhất và xuyên suốt trong giáo dục STEM.



Gợi ý viết chương trình: Sử dụng cấu trúc rẽ nhánh **If Else** (Nếu ..còn không...) và vòng lặp vĩnh viễn **forever** trong nhóm block **Control** để thực hiện thuật toán trên thành code Scratch.



Tiến hành nạp thử chương trình và quan sát hành vi của robot xem có đúng không nhé.

Nếu Robot phản ứng chậm hơn dự đoán:

- Bề mặt phản xạ không vuông góc với cảm biến siêu âm dẫn đến kết quả đo không chính xác
- Tốc độ thực thi lệnh là có hạn vì vậy các bạn tiến hành giảm tốc độ của Robot để cho robot có cơ hội phản ứng kịp với những đột biến trong môi trường.

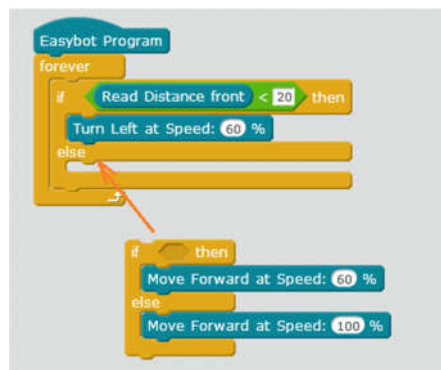
### **Bài tập 7**

Trong bài tập 2, các bạn được trải nghiệm thế nào là thuật toán và hiện thực hóa nó vào bộ não của robot. Nhưng quan sát hành vi của robot, đó chưa phải là cách xử lý tối ưu. Nếu tăng tốc độ Robot có thể sẽ phản ứng chậm. Yêu cầu đặt ra là:

-----Nếu robot cách vật thể ở 30cm thì giảm tốc độ.

-----Nếu robot cách vật thể ở 15cm thì đổi hướng.

Gợi ý, kế thừa bài tập 2, các bạn thay block **Move Forward** trong phần **Else** của chương trình bằng 1 lệnh rẽ nhánh If--Else— nữa



Các bạn tự lắp ráp block **if else** mới tạo và thêm phần toán tử so sánh rồi ghép với chương trình chính để hoàn thiện chương trình nhé. Lúc này robot có thể chạy nhanh lên và xử lý thêm được nhiều trường hợp.

**Bài tập 8:** Hãy viết chương trình để làm cho robot đi theo một vật thể theo đường thẳng. Gợi ý về thuật toán như sau:

---Nếu phát hiện vật thể quá gần: robot lùi lại

---Nếu vật thể ở trong tầm vừa phải( không gần, không xa): robot dừng lại

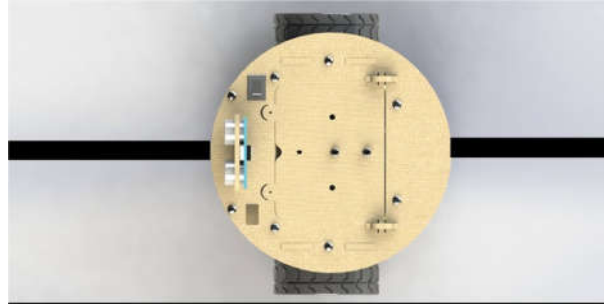
---Nếu vật thể ở quá xa: đi tới.

### **Bài tập 9**

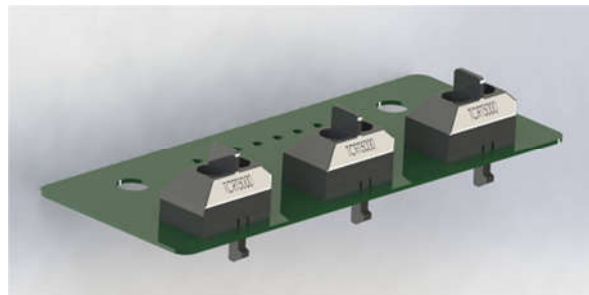
Hãy suy nghĩ thuật toán và viết chương trình để biến robot thành một radar phòng không, robot xoay liên tục, nếu phát hiện có vật thể trong tầm phát hiện thì lao đến theo hướng có vật thể.



Các ứng dụng của cảm biến đo khoảng cách siêu âm sẽ còn được ứng dụng nhiều trong các trò chơi tổng hợp ở phần cuối của tài liệu.

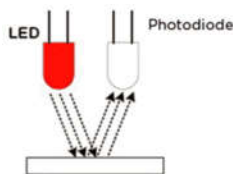
**Lesson 3: Cảm biến dò vạch (Line Following Sensor)**

Làm thế nào Robot có thể đi theo vạch kẻ đen giữa 1 con đường màu sáng. Đó là nhờ cảm biến phát hiện vạch.

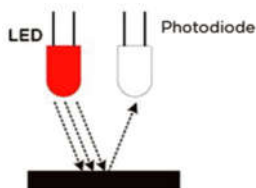
**Nguyên lý hoạt động:**

Cảm biến phát hiện vạch của Easybot gồm có cặp thu và phát hồng ngoại. Phạm vi phát hiện dưới 1 cm. Nó hoạt động bằng cách phát ánh sáng hồng ngoại xuống bề mặt và cảm nhận lượng ánh sáng hồng ngoại phản xạ trở lại (photodiode).

Khi gặp bề mặt có tính phản xạ cao (vd màu trắng) thì lượng ánh sáng cảm nhận lại rất lớn



Khi gặp bề mặt có tính phản xạ kém (vạch đen) thì lượng ánh sáng cảm nhận lại rất yếu



Có 3 cảm biến để xác định vị trí của robot so với vạch:

Trái: phát hiện robot đang lệch về bên trái của vạch

Giữa: phát hiện robot đang ở giữa vạch


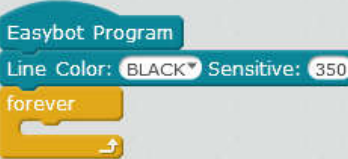


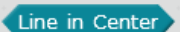



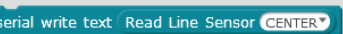


Phải: phát hiện robot đang lệch về bên phải của vạch.

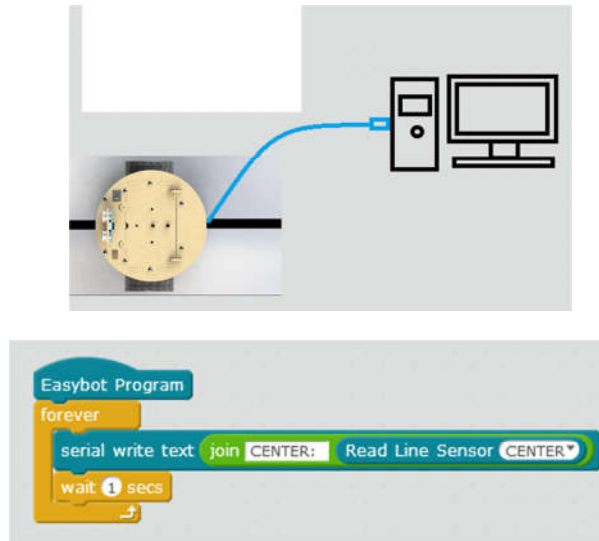
Ngày nay, công nghệ dò theo line được sử dụng phổ biến trong các robot vận chuyển hàng hóa trong các nhà kho tự động, nhà máy có tự động hóa cao.

Trong tài liệu và các bài tập, cảm biến dò line được sử dụng cho các bài toán di chuyển trong sa hình có vạch dò, giải toán mê cung, đấu võ Sumo...

### Kiến thức cần nắm

Block liên quan	Hướng dẫn	Ví dụ
	Lệnh cấu hình cho sensor trong đó: + chọn màu vạch kẻ là đen hay trắng. + Chọn độ nhạy của cảm biến (sử dụng cho nhiều loại bề mặt sân chơi khác nhau) : Nếu giá trị phản xạ nhận được < độ nhạy thì sẽ nhận là có vạch và ngược lại.	 <p>Cấu hình cho cảm biến được sử dụng đầu tiên ngay khi khởi đầu chương trình sau đó mới là vòng lặp vĩnh viễn chưa hoạt động chính của robot, độ nhạy là 350, vạch kẻ màu đen. ( nếu ánh sáng phản xạ thu nhận được có giá trị nhỏ hơn 350 thì xác định đó chính là vạch đen)</p>
	Cảm biến bên trái phát hiện vạch? <Đúng/Sai>	 <p>Kiểm tra nếu phát hiện vạch thì làm gì đó....Sử dụng cho các Điều khiển kiểm tra điều kiện hoặc đúng hoặc sai</p>
	Cảm biến ở giữa phát hiện vạch? <Đúng/Sai>	
	Cảm biến bên phải phát hiện vạch <Đúng/Sai>	
	Đọc giá trị ánh sáng phản xạ trở lại cảm biến. Thường dùng để đo độ phản xạ của sân chơi, để xác định độ nhạy cài đặt cho cảm biến sử dụng trong việc cấu hình cảm biến trong block: 	 <p>Truyền giá trị cảm biến hồng ngoại nhận được (vị trí giữa) lên máy tính để kiểm tra hoạt động của cảm biến.</p>

**Bài tập 10:** Viết chương trình cho robot truyền dữ liệu về độ phản xạ của ánh sáng nhận được lên máy tính để kiểm tra và xác định độ nhạy và độ tương phản của sân chơi

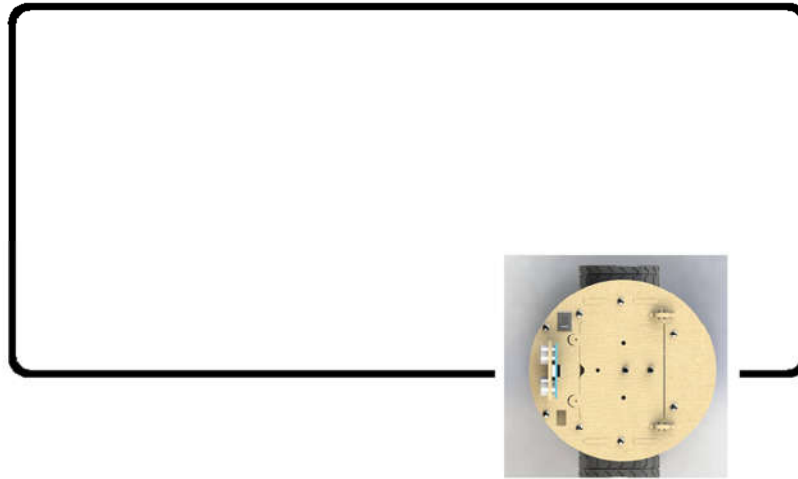


Sử dụng toán tử nối 2 chuỗi văn bản (Operator: join <><>) giống như phần trước. Upload chương trình vào robot và quan sát trên màn hình, di chuyển Robot sao cho cảm biến ở giữa tiếp cận với vạch kẻ và ra ngoài nền để so sánh độ tương phản thông qua giá trị nhận được. Ví dụ, khi nhận được ánh sáng yếu (vạch kẻ) thì giá trị nhận được là >800, Khi nhận được ánh sáng mạnh (nền sáng), giá trị nhận được là <100.. Ngưỡng phân biệt an toàn không bị nhầm lẫn nên là ở khoảng giữa.

Vậy ta nên set giá trị để robot phân biệt đâu là vạch kẻ và nền sáng vào khoảng 350. Như vậy với giá trị nhận được dưới 350 sẽ nhận là nền, trên 350 sẽ nhận là vạch đen.

---Tương tự Tiếp tục sửa chương trình để kiểm tra độ nhạy của các mắt cảm biến còn lại.

**Bài tập 11:** Viết chương trình dò theo vạch đơn giản theo sa hình đơn giản sau



Hướng suy nghĩ:

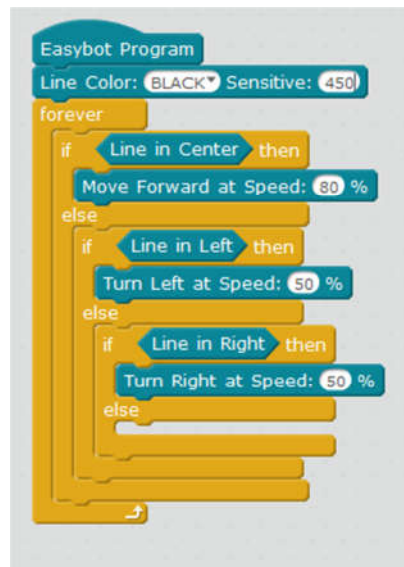
-----Nếu cảm biến ở giữa phát hiện có vạch đen: Đi thẳng

-----Còn không: ----Nếu cảm biến bên trái phát hiện có vạch đen: quay trái

-----còn không: -----nếu cảm biến bên phải phát hiện vạch đen: quay phải

-----Còn không: Tôi rồi, Tôi bị lạc mất vạch rồi

Gợi ý:



Viết và Nạp chương trình theo gợi ý, sau đó quan sát robot hoạt động, ta thấy rằng robot di chuyển chưa mượt mà..Làm thế nào để robot di chuyển mượt mà hơn ?

**Bài tập 12:** Viết chương trình thực hiện giống bài tập 2 nhưng robot bám đường mượt mà hơn

Gợi ý: sử dụng các block di chuyển  để tinh chỉnh vận tốc motor thay cho các block đi thẳng và quay vòng để robot di chuyển mượt mà hơn. Chú ý: để nhập giá trị âm để đảo chiều động cơ)

**Bài tập 13:** Xác định và đếm số ngã tư .

- Phương pháp Xác định đi qua ngã tư để làm cơ sở đếm số ngã tư đã đi qua
- Viết chương trình đếm số ngã tư và truyền lên máy tính để kiểm tra

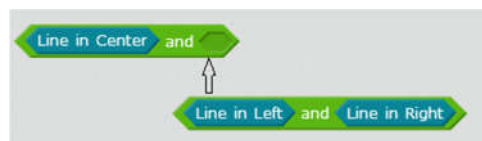
### Kiến thức cần nắm

Blocks	Cách sử dụng	Ví dụ
 <p>Tạo một biến lưu trữ dữ liệu</p>	Tạo ra một biến (là đơn vị lưu trữ dữ liệu) trung gian để giải quyết vấn đề gì đó ví dụ như số ngã tư đã đi qua,	 <p>Tạo ra biến Count để lưu trữ số đếm.</p>
 <p>“and” Toán tử “and”</p>	Toán tử <b>and</b> trong nhóm <b>operators</b> sử dụng để kiểm tra các điều kiện xảy ra đồng thời. Nếu các điều kiện cùng đúng thì nó trả về kết quả là đúng, nếu chỉ cần một điều kiện sai thì nó sẽ trả về kết quả là sai.	 <p>Nếu cảm biến bên phải và cảm biến bên trái đều đồng thời phát hiện vạch đen thì trả về kết quả là đúng, nếu một trong hai không phát hiện vạch đen thì trả về kết quả là sai</p>

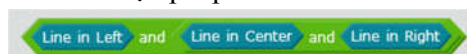
Phương pháp xác định ngã tư: Ngã tư được xác định khi cả 3 mắt đều phát hiện vạch kẻ.

Có 1 vấn đề: Toán tử **and**  chỉ kiểm tra 2 điều kiện, làm thế nào để kiểm tra 3

điều kiện? ta sử dụng 2 toán tử  cách làm như sau:



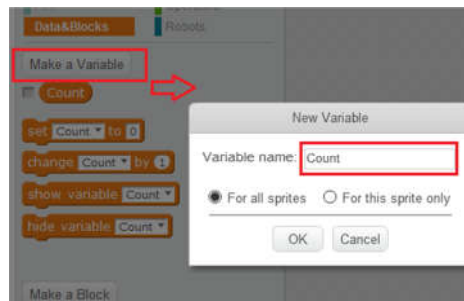
Như vậy ta sẽ có được phép kiểm tra **And** với 3 điều kiện:



**Nếu phép kiểm tra này là đúng thì có nghĩa là 1 ngã tư đã được phát hiện**

Khi kiểm tra được cả 3 cảm biến đều nhận vạch kẻ đen thì ta tăng một số đếm trung gian lên 1 đơn vị và gửi số đó lên máy tính. Muốn vậy ta phải tạo ra 1 biến (variable) để chứa giá trị số đếm.

Trong nhóm block **Data&Blocks**, bấm vào “**Make a Variable**”, sau đó gõ tên biến là Count



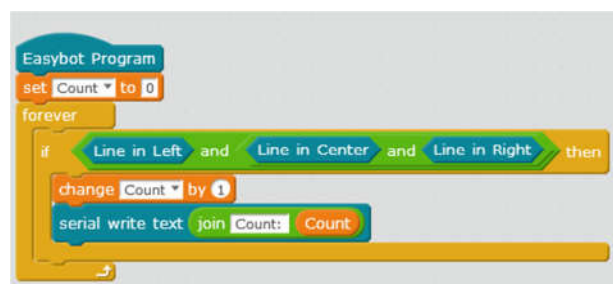
Ta đã tạo 1 biến Count. Các thao tác với biến Count gồm có:

**Count** : block chứa giá trị biến Count

**set Count to 0** : đặt giá trị cho biến Count: về toán học nó giống như gán:  $\text{Count} = n$

**change Count by 1** : thay đổi giá trị biến count bằng cách tăng hay giảm 1 lượng. Về toán học nó giống với  $\text{count} = \text{count} + n$  (với  $n > 0$ : tăng, với  $n < 0$ : giảm)

Ta hoàn thiện chương trình như sau: nếu cả 3 cảm biến đều phát hiện vạch đen thì tăng Count lên 1 đơn vị. Nạp chương trình và đưa robot đi ngang ngã tư để quan sát.



Chương trình trên hoạt động được nhưng sẽ có 1 vấn đề, số biến đếm tăng nhiều khi cho robot lướt qua 1 vạch ngang. Lý do là do tốc độ thực thi chương trình rất nhanh so với tốc độ di chuyển của Robot do đó khi Robot còn đang ở trên vạch ngang chưa ra khỏi vạch thì chương trình tiếp tục lặp lại đếm và nó lại tăng biến đếm -> không chính xác.

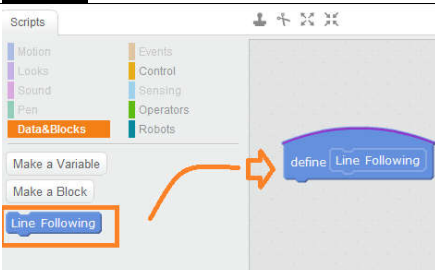
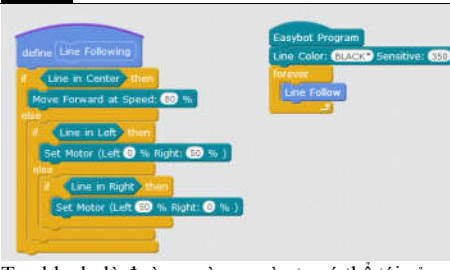
Giải pháp là sử dụng lệnh đợi 1 thời gian trước khi lặp lại việc đếm vạch:



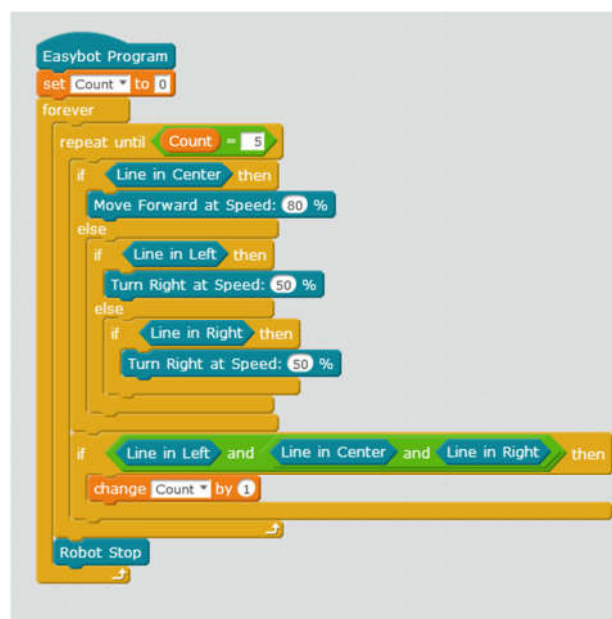
Thời gian chờ chú ý lựa chọn phù hợp với tốc độ di chuyển của robot để đảm bảo không bị trễ việc cho các tác vụ khác.

**Bài tập 14:** Viết chương trình kết hợp dò đường theo vạch đen và đếm số ngã tư, khi đến vị trí yêu cầu (xác định bằng số ngã tư đã đi qua) thì dừng lại.

### Kiến thức cần nắm:

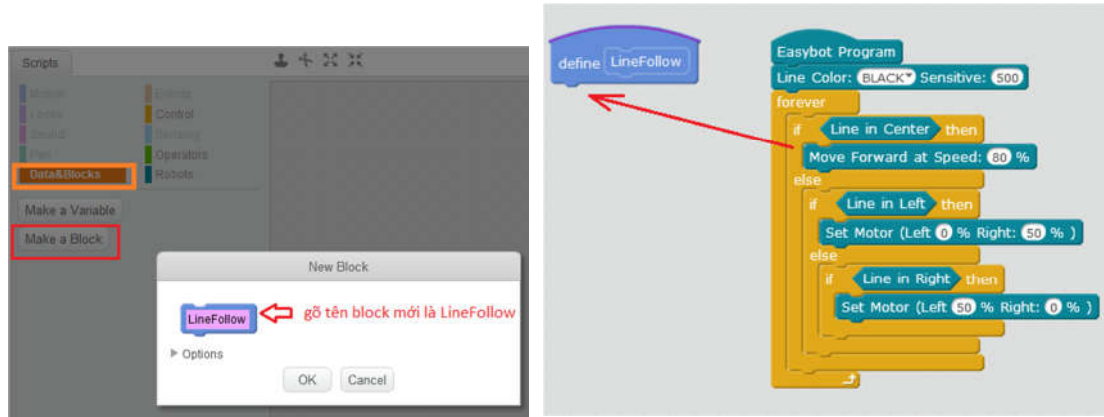
Blocks	Giải thích	Ví dụ
 <p>Tạo một block riêng từ nhiều block khác nhau</p>	<p>Khi viết một chương trình phức tạp để giải quyết vấn đề, ta có thể làm gọn chương trình bằng cách tạo ra 1 block mới chứa các đoạn mã block mà có nhu cầu sử dụng lặp đi lặp lại nhiều lần.</p>	 <p>Tạo block dò đường, và sau này ta có thể tái sử dụng để kết hợp vừa dò đường vừa làm việc khác một cách dễ dàng và mạch lạc</p>

Gợi ý: Kết hợp chương trình dò vạch ở **bài tập 11** và phương pháp đếm ngã tư ở **bài tập 13**, phối hợp trong cùng 1 chương trình Scratch như thế này:

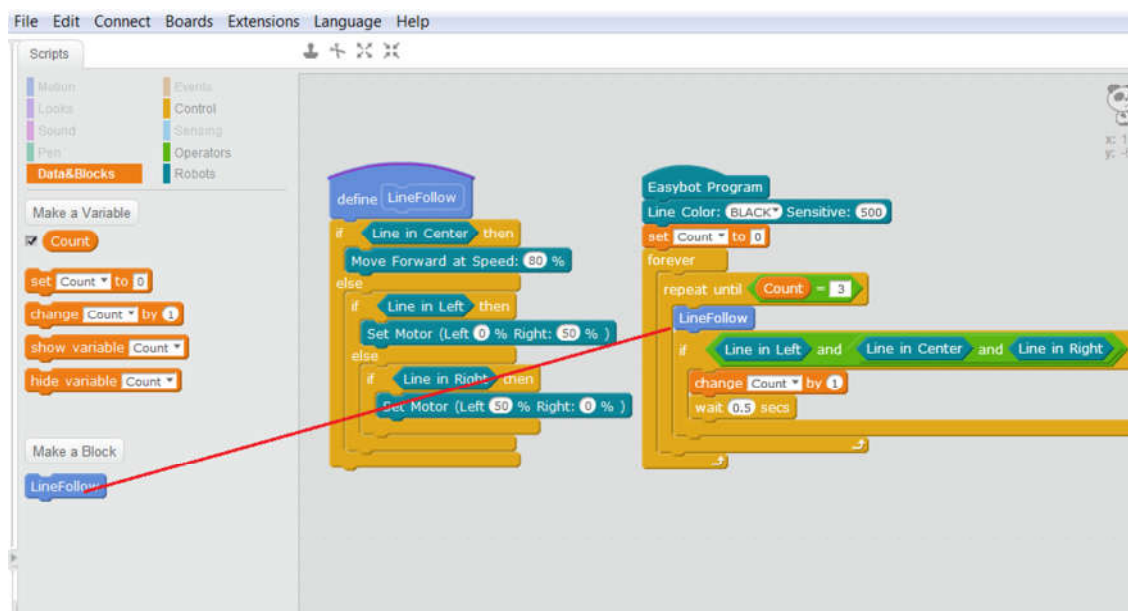


Lúc này chương trình có vẻ khá phức tạp và khó quản lý khi các bạn muốn có một số sửa đổi cải tiến về chương trình ? Đừng lo có 1 giải pháp khác đó là tạo ra 1 Block mới đóng vai trò như 1 chương trình con. Ta sẽ gộp tất cả những đoạn mã liên quan đến dò line và 1 block để chương trình dễ quản lý hơn và dễ dàng triển khai phối hợp với các chương trình khác kết hợp dò line.

Thực hiện như sau: vào group **Data&Blocks** bấm vào [Make a Block](#)



Sau đó kéo và thả cả đoạn chương trình liên quan đến dò line vào Block mới tạo

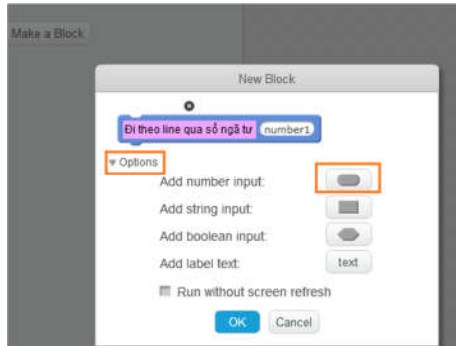


Giờ thì chương trình trông rất khoa học và bài bản, nếu muốn sửa đổi nâng cấp các thuật toán dò đường các bạn có thể làm tự do mà không sợ bị ảnh hưởng đến cấu trúc chương trình chính.



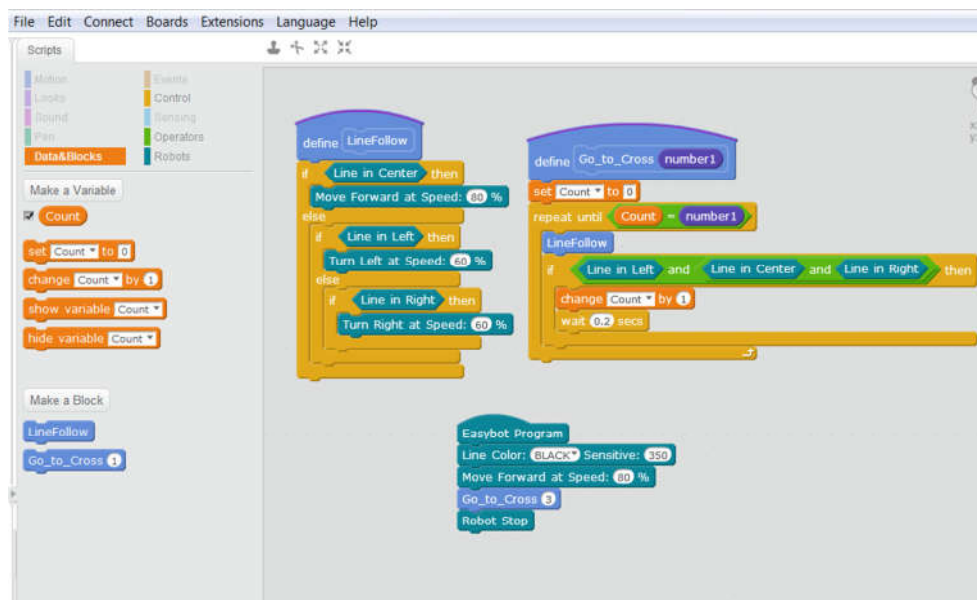
**Bài tập 15:** Tương tự bài tập 14, Hãy tạo ra một block mới (chương trình con) thực hiện chức năng vừa dò đường và đếm đủ số ngã tư thì dừng. Tham số đầu vào (input) của Block là số ngã tư phải vượt qua.

Gợi ý: Khi tạo block mới, vào option, ta thêm số liệu đầu vào (add number input) để nhập số ngã tư cần vượt qua

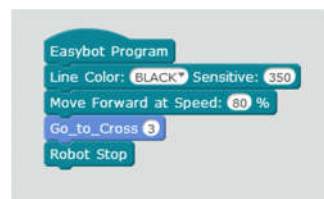


Sau đó tiếp tục kế thừa bài tập 14, vẫn tiếp tục sử dụng block Linefollow và tạo thêm block Go\_to\_Cross (number) để di chuyển vừa sử dụng block linefollow vừa đếm số ngã tư.

Như vậy ta đã tạo ra 1 block rất hữu ích cho các chương trình đi vào sa hình dò line phức tạp.



Như vậy Chương trình chính trở nên rất đơn giản, ví dụ muốn dò line qua 3 ngã tư thì dùng:



Với bài tập này, các bạn sẽ thành thạo trong việc tạo ra các block chức năng riêng từ những block sẵn có để tái sử dụng cho các chương trình có độ phức tạp cao hơn.

**Bài tập 16:** *Viết chương trình tạo block con quay phải 90 độ quanh ngã tư. Chú ý, đề bài ngắn nhưng khó ở chỗ phải đảm bảo robot quay ra khỏi vạch đen rồi mới kiểm tra đã quay đủ góc ngã tư hay chưa. Ứng dụng Block vừa tạo để quay 4 góc của 1 ngã tư.*

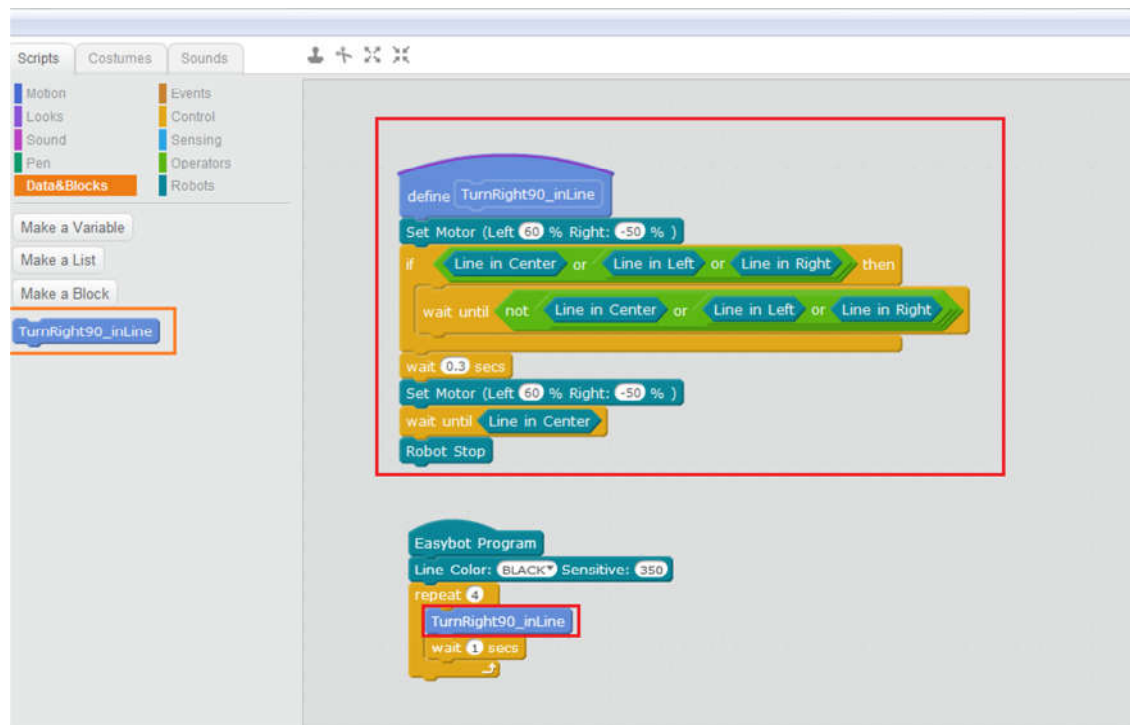
Suy nghĩ như sau:

---Khi robot đang còn ở trên vạch

-----Quay phải cho đến khi không còn vạch

--Delay 1 khoảng 0.2s để thực sự ra khỏi vạch

---Tiếp tục quay phải cho đến khi cảm biến ở giữa nhận được vạch



Như vậy chúng ta đã tạo ra được các block con về đi theo vạch và đếm ngã tư, rẽ trái / rẽ phải ở Ngã tư.

Bây giờ các bạn có thể viết các chương trình phức tạp như một cuộc thi Robocon thực sự.

**Bài tập 17: Đấu trường SUMO**

Đấu trường SUMO được tổ chức như cuộc thi đấu vật SUMO, sân thi đấu nằm trong một vòng tròn màu đen, nền sân màu trắng, 2 Robot sẽ phải tìm và húc nhau, ai bị đẩy khỏi vòng tròn sẽ thua cuộc. Bắt đầu trận đấu, 2 robot sẽ quay lưng về phía nhau.

Gợi ý thuật toán cơ bản:

Bắt đầu:

-----Robot sẽ xoay quay để tìm đối thủ, cho đến khi thấy có vật cản trong tầm quan sát

-----Robot lao đến cho đến khi chạm vạch thì lùi lại 1 đoạn ngắn

---Lặp lại

*SUMO*

