



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

HƯỚNG DẪN LẮP RÁP VÀ LẬP TRÌNH CHO LEGO MINDSTORMS NXT MÔN HỌC LEGO-ROBOTICS

Công ty Cổ Phần Việt Tinh Anh giữ bản quyền.

Nhóm biên dịch:

Lê Phúc Nguyên Tuấn

Trần Quang Viễn



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

LỜI NÓI ĐẦU

Lớp học “Kĩ Sư Robot” - LEGO Robotics là một trong những lớp học ngoại khóa nhằm phát triển năng khiếu kỹ thuật và ứng dụng công nghệ vào việc giải quyết các vấn đề trong cuộc sống cho các em học sinh tiểu học và trung học. Chương trình dạy và học dựa vào các bài toán thực tế, các kỹ thuật công nghệ mới, với sự dẫn dắt khuyến khích của các giảng viên, sẽ tạo điều kiện cho các học sinh tự mình đóng một vai trò chủ động trong quá trình học tập. Các em sẽ có ham muốn tự mình mày mò đưa ra các giải pháp, thiết kế, lắp ráp, và lập trình cho các giải pháp này với các kiến thức phổ thông về toán, vật lý, tin học, v.v...; các em cũng sẽ có nhu cầu bàn bạc hợp tác với các bạn cùng nhóm để cùng nhau giải quyết vấn đề - đây là điều không dễ mà có được trong nhiều môn học khác mà việc giải quyết các vấn đề của chúng chỉ thông qua giấy và viết.

Việc sử dụng sản phẩm LEGO MINDSTORM NXT trong chương trình học giúp chúng ta không phải tự mình xây dựng một robot từ con số không, tránh được việc phải chú ý chế tạo các chi tiết quá nhỏ làm mất quá nhiều thời gian và công sức. Người dùng chỉ cần nắm rõ phương pháp sử dụng các khối lắp ráp độc lập cơ bản có sẵn, là có thể kết hợp chúng để tạo ra một hệ thống phức tạp hơn, và đây cũng chính là xu hướng của kỹ thuật thế giới - thiết kế chế tạo theo module. Tuy nhiên, do môn học này hiện tại vẫn còn khá mới mẻ ở Việt Nam, nên các tài liệu hướng dẫn về LEGO MINDSTORMS NXT bằng Việt ngữ hầu như không có, hoặc chỉ có ở dạng các bài hướng dẫn ngắn và thường là không đầy đủ. Với mong muốn có một cuốn sách bằng tiếng Việt đầy đủ và chi tiết hơn nhằm tiện cho các giáo viên và các em học sinh tham khảo và tra cứu, nhóm tác giả đã cố gắng biên dịch và tổng hợp từ các tài liệu nước ngoài và Internet để cho ra đời cuốn sách "Hướng Dẫn Lắp Ráp và Lập Trình Cho LEGO MINDSTORMS NXT" này.

Để sử dụng hiệu quả cuốn sách này, độc giả nên dùng kèm với bộ Robotkit 9797 (LEGO MINDSTORMS Education NXT Base Set 9797) và nếu cần thiết thì bổ sung thêm các khối lắp ráp của bộ 9695 (LEGO MINDSTORMS Education NXT Resource Set 9695). Các chương sách được viết nhằm bổ sung cho giáo án của chương trình lớp học "Kĩ sư Robot" cơ bản và nâng cao do công ty Việt Tinh Anh đã biên soạn. Với mỗi bài giảng, giáo viên có thể cho các em tự đọc các chương tương ứng ở nhà trước để có kiến thức cơ bản, và ở lớp thì có thể trình bày về phương pháp kết hợp các phần cơ bản này để giải các bài tập hoặc đào sâu vào các khả năng ứng dụng của chúng. Do cuốn sách được cố ý trình bày một cách tổng quát với đủ các chi tiết, nên các độc giả tự học cách lắp ráp robot MINDSTORMS NXT cũng có thể tuần tự tìm hiểu các chương sách hoặc có thể tham khảo trực tiếp đến phần nào có đề tài mà mình quan tâm.

Do khả năng và thời gian bị hạn chế, nên cuốn sách chắc sẽ có những thiếu sót. Nhóm tác giả rất mong nhận được sự góp ý của các thầy cô giáo, các em học sinh, và các độc giả khác để có thể chỉnh sửa cuốn sách này ngày càng tốt hơn. Bên cạnh đó, nhóm tác giả cũng sẽ cố gắng định kỳ cập nhật nội dung của cuốn sách để bổ sung các mục còn chưa được như ý. Hy vọng cuốn sách này sẽ giúp ích được ít nhiều cho quý độc giả.

Ý kiến góp ý xin gửi thư điện tử về địa chỉ info@viettinhanh.com.vn. Xin chân thành cảm ơn.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

MỤC LỤC

LỜI NÓI ĐẦU	trang 2
PHẦN 1 : GIỚI THIỆU	6
CHƯƠNG 1 : GIỚI THIỆU CHUNG VỀ ROBOTICS VÀ ROBOT LEGO EDUCATION.....	5
I. Giới thiệu về Robotics :	6
II. Ứng dụng Robot trong sản xuất :	6
III. Giới thiệu về Robot Lego Education :	7
CHƯƠNG 2 : GIỚI THIỆU VỀ KHỐI LẮP RÁP LEGO.....	9
I. Giới thiệu:.....	9
II. Giới thiệu cơ bản về các thanh, gạch, cấu kiện LEGO.....	9
III. Thể hiện kích thước và đơn vị :	14
IV. Chốt kết nối.	23
PHẦN 2 : ĐỘNG CƠ.....	26
CHƯƠNG 3 : TỔNG QUAN VỀ ĐỘNG CƠ LEGO	26
I. Giới thiệu :	26
II. Nguyên lý hoạt động :	26
III. Giới thiệu động cơ LEGO.....	27
PHẦN 3 : CÁC CƠ CẤU TRUYỀN ĐỘNG.....	31
CHƯƠNG 4 :TRUYỀN ĐỘNG BÁNH RĂNG.	31
I. Giới thiệu :	31
II. Các đặc điểm cần lưu ý :	31
III. Các ứng dụng của bánh răng thông thường :	32
IV. Một số cơ cấu bánh răng khác :	34



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

CHƯƠNG 5 : CÁC BỘ TRUYỀN ĐỘNG KHÁC.....	39
I.Ròng rọc(Pulley) :	39
CHƯƠNG 6 : CÁC CƠ CẤU CƠ KHÍ.....	42
I. Cơ cấu cam :	42
II. Cơ cấu vi sai	43
III. Cơ cấu “vòng lái”	45
IV. Cơ cấu lái :	46
V. Cơ cấu tay quay con trượt :	48
PHẦN 4 : CÁC CƠ CẤU LÀM VIỆC THÔNG THƯỜNG.....	50
CHƯƠNG 7 : CÁC CƠ CẤU TAY GẤP – TAY NÂNG	50
I.Tay gấp cơ bản:	50
II. Cơ cấu nâng :	51
CHƯƠNG 8 : CÁC CƠ CẤU DI CHUYỂN.....	52
I.Di chuyển bằng bánh xe :	52
II.Di chuyển bằng xích :	52
CHƯƠNG 9 : CÁC LOẠI CẢM BIẾN.	56
I. Giới thiệu:.....	55
II. Cảm biến chạm :	55
III. Cảm biến ánh sáng:.....	56
IV. Cảm biến siêu âm:.....	57
V. Cảm biến âm thanh :	58
PHẦN 5 : BỘ PHẬN ĐIỀU KHIỂN.....	60
CHƯƠNG 10 : SỬ DỤNG CHƯƠNG TRÌNH NXT - G.....	60
I. Giới thiệu về môi trường lập trình LEGO MINDSTORMS NXT 2.0 (NXT-G).	60



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT: (08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

II. Thảo một chương trình NXT-G đơn giản:	64
III. Các khái niệm cơ bản trong ngôn ngữ lập trình NXT-G:	66
IV. Các khối lệnh xử lý dữ liệu:	75
CHƯƠNG 11 : CÁC CẤU TRÚC ĐIỀU KHIỂN.....	84
I. Khối lệnh [Switch]:	84
II. Khối lệnh [Loop]:	92
CHƯƠNG 12 : LẬP TRÌNH VỚI THIẾT BỊ NGOẠI VI	101
I.Điều khiển cảm biến chạm:	101
II. Điều khiển cảm biến ánh sáng:	104
III. Điều khiển cảm biến âm thanh:	108
IV. Điều khiển cảm biến siêu âm:	110
V. Điều khiển động cơ:	115
PHẦN 6 : THIẾT KẾ ROBOT.....	121
CHƯƠNG 13 : THIẾT KẾ VÀ LẬP TRÌNH ROBOT.....	121
I.Giới thiệu :	121
II. Robot cơ bản Castor Bot:	121
III. Lập trình cho Robot hành động đơn giản :	128
IV. Lập trình Robot dò đường :	130
VI. Lập trình Robot dò đường và tránh chạm vật cản:.....	133



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

PHẦN 1 : GIỚI THIỆU

CHƯƠNG 1 : GIỚI THIỆU CHUNG VỀ ROBOTICS VÀ ROBOT LEGO EDUCATION

I. Giới thiệu về Robotics :

Thuật ngữ “Robot” xuất phát từ tiếng Sec “Robota” - nghĩa là công việc tạp dịch trong vở kịch Rossum’s Universal Robots của Karel Capek, vào năm 1921. Trong vở kịch này, Rossum và con trai đã chế tạo ra chiếc máy gần giống với con người để phục vụ con người. Có lẽ đó là gợi ý ban đầu cho các nhà sáng chế kỹ thuật về những cơ cấu, máy móc bắt chước các hoạt động cơ bắp của con người.

**Định nghĩa về Robot :*

Theo tiêu chuẩn AFNOR (Pháp) :

Robot là một cơ cấu chuyển động tự động có thể lập trình, lặp lại các chương trình, tổng hợp các chương trình đặt ra trên các trục toạ độ, có khả năng định vị, định hướng, di chuyển các đối tượng theo những hành trình đã được chương trình hoá nhằm thực hiện các nhiệm vụ công nghệ khác nhau.

Theo RIA(Robot institute of Americal) :

Robot là một tay máy vạn năng có thể lập lại các chương trình được thiết kế để di chuyển vật liệu, chi tiết, dụng cụ hoặc các thiết bị chuyên dùng thông qua các chương trình chuyển động có thể thay đổi để hoàn thành các nhiệm vụ khác nhau.

Theo ГOCT(Nga) :

Robot là một loại máy móc tự động, được đặt cố định hoặc di động được, liên kết giữa một tay máy và một hệ thống điều khiển theo chương trình, có thể lập đi lập lại để hoàn thành chức năng vận động và điều khiển trong quá trình sản xuất.

Có thể nói, Robot là sự tổ hợp các khả năng hoạt động linh hoạt của các cơ cấu điều khiển từ xa với mức độ “tri thức” ngày càng phong phú của hệ thống điều khiển theo chương trình số cũng như kỹ thuật chế tạo các bộ cảm biến, công nghệ lập trình và sự phát triển của trí khôn nhân tạo.

Trong những năm sau này, việc nâng cao tính năng hoạt động của Robot không ngừng được phát triển. Các Robot được trang bị thêm các loại cảm biến khác nhau để nhận biết môi trường chung quanh, cùng với những thành tựu to lớn trong lĩnh vực Tin học – Điện tử - Cơ khí, đã tạo ra các thế hệ Robot với nhiều tính năng đặc biệt.

II. Ứng dụng Robot trong sản xuất :

Từ khi mới ra đời, Robot được áp dụng trong nhiều lĩnh vực dưới góc độ thay thế sức người. Nhờ vậy các dây chuyền sản xuất được tổ chức lại, năng suất và hiệu quả sản xuất tăng lên rõ rệt.

Mục tiêu ứng dụng của Robot nhằm góp phần nâng cao năng suất dây chuyền công nghệ, giảm giá thành, nâng cao chất lượng và khả năng cạnh tranh của sản phẩm đồng thời cải thiện điều kiện lao động. Đạt được các mục tiêu trên là nhờ vào những khả năng to lớn của Robot như : làm việc không mệt mỏi, rất dễ dàng “chuyển nghề” một cách thành thạo, chịu được phóng xạ và môi trường làm việc độc hại, nhiệt độ cao, “cảm thấy” được cả từ trường và “nghe” được cả sóng siêu âm...



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

Như vậy, khả năng làm việc của Robot trong một số điều kiện vượt hơn khả năng của con người, do đó nó là phương tiện hữu hiệu để tự động hoá, nâng cao năng suất lao động, giảm nhẹ cho con người những công việc nặng nhọc và độc hại. Nhược điểm lớn nhất của Robot là chưa linh hoạt như con người, trong một môi trường tự động, nếu có một Robot bị hỏng có thể ngừng hoạt động cả dây chuyền, cho nên Robot vẫn luôn hoạt động dưới sự giám sát của con người.

III. Giới thiệu về Robot Lego Education :

LEGO® MINDSTORMS® - giải pháp lý tưởng cho việc giảng dạy Robotics ở mọi cấp độ.

LEGO MINDSTORMS là hệ thống Robot cung cấp giải pháp giáo dục hoàn toàn thích hợp của phần cứng, phần mềm và các nguồn tài nguyên giáo dục để sử dụng trong các lớp học, câu lạc bộ ở trường và tự học ở nhà. Học viên được thiết kế, lập trình và kiểm soát mô hình Robot đầy đủ các chức năng để thực hiện các nhiệm vụ một cách tự động.

Tại sao lại là LEGO ?

Sức mạnh của hệ thống LEGO nằm trong khái niệm sáng lập : nó có thể dùng lại. Cùng một khối gạch cơ bản, có thể hôm nay nó là chân con voi, ngày mai nó lại là một khối trong kim tự tháp Ai Cập, và ngày hôm sau nữa nó lại là mũi của một con Robot. Khi mở một hộp LEGO, bạn có thể thấy những hình mẫu hướng dẫn, nhưng bạn cũng thấy rằng sẽ có rất nhiều mô hình khác bạn có thể tạo nên từ những bộ phận sẵn có.

Điều rất tiện ích khi biến những thanh nhựa nhỏ thành một hệ thống lớn chính là ở phần kết nối. Bạn không cần keo, vít, hoặc bất kỳ công cụ khác nào để lắp ráp hoặc tháo gỡ các mô hình LEGO. Các bộ phận LEGO dễ dàng được kết nối một cách vững chắc và tạo thành mô hình tuyệt vời cho đến khi bạn quyết định tháo gỡ nó. Các bộ phận không độc hại và rất ít hư hỏng, dù bạn có sử dụng rất nhiều lần.

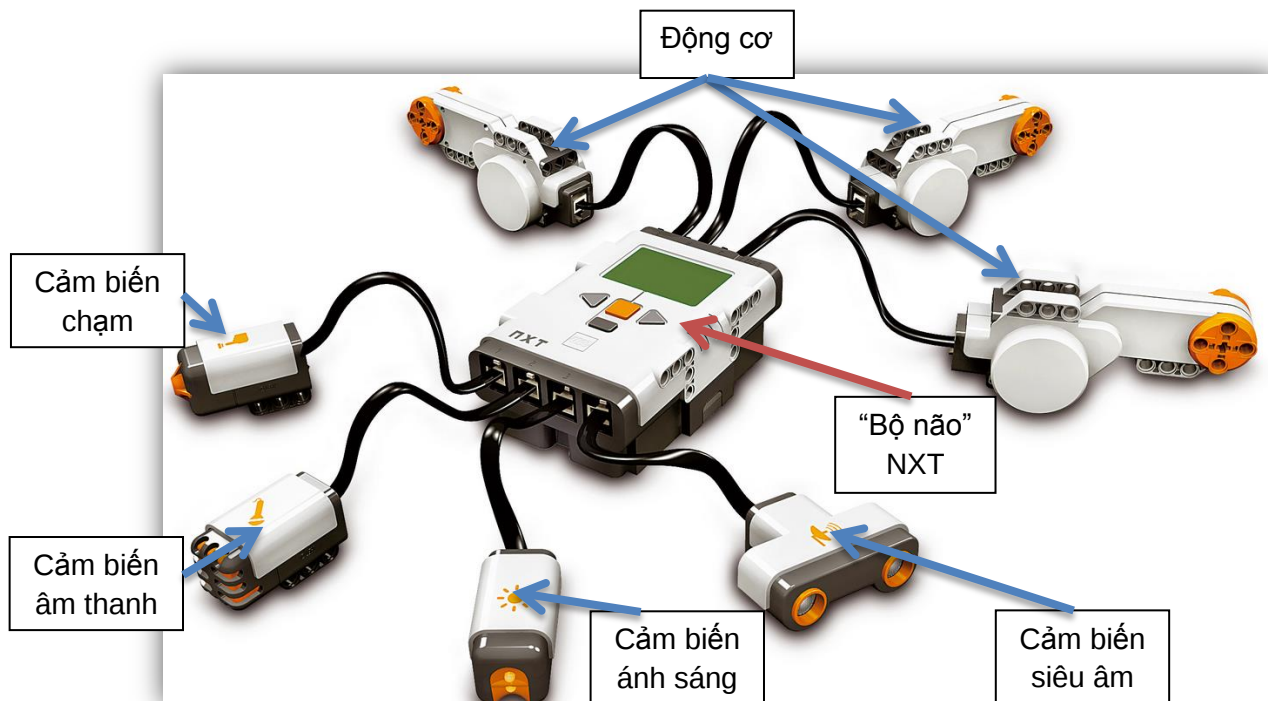
Nhưng điều mà thực sự khiến cho LEGO dễ sử dụng chính là tính mô đun của nó. Không chỉ ngẫu nhiên là một thanh kết nối với một thanh khác, mà ở đây là có sự tính toán, xác định trước. Các đầu đinh và lỗ của của thanh buộc bạn phải lắp ráp một cách chính xác theo sơ đồ hình học. Đây có thể là điều gây khó khăn lúc đầu, nhưng nó thực sự làm cho mọi thứ trở nên dễ dàng hơn, gọi là sự định vị chính xác. Bạn không cần phải có thước kẻ hoặc thước đo góc vì tất cả những gì cần thiết bạn đều có thể đếm được.

- ❖ LEGO rất nhanh : bạn không cần phải cưa, cắt, khoan, hàn, tiện, phay...các bộ phận. Tất cả đã sẵn sàng để sử dụng, chỉ cần chọn những gì bạn cần.
- ❖ LEGO rất sạch : bạn không cần phải giũa, không cần chất bôi trơn hoặc sơn, và khi bạn chơi xong, căn phòng sẽ trông giống hệt như khi bạn chưa chơi. Đây là một điểm vô cùng quan trọng để những người sống cùng cảm thấy hài lòng về sở thích của bạn.
- ❖ LEGO – chi phí hiệu quả : bạn có thể sử dụng và tái sử dụng các thành phần của bộ LEGO để tạo ra các thế hệ Robot tiếp theo. Và thậm chí giả dụ như cái lốp xe, nó vẫn có giá trị thương mại, vì nó dễ dàng lắp ráp vào các mô hình khác, nên nó sẽ có ích với những ai đang thiếu.
- ❖ LEGO rất thân thiện với môi trường : Là một trong những tập đoàn đứng đầu thế giới về chất lượng sản phẩm và dịch vụ, LEGO luôn tuân thủ những hướng dẫn tiêu chuẩn đồ chơi châu Âu, qui định của luật liên bang và những tiêu chuẩn ASTM F963. Chất liệu nhựa sử dụng trong sản xuất sản phẩm của LEGO là nhựa ABS tuyệt đối không độc hại, an toàn với độ bền cao, khả năng chịu nhiệt, axit, muối và các hóa chất khác.
- ❖ LEGO MINDSTORM NXT – với bộ lắp ráp này, bạn có thể xây dựng Robot của riêng mình, tất cả tùy thuộc vào sự sáng tạo của riêng bạn. Dĩ nhiên, bộ NXT là một món đồ chơi tuyệt vời, và nhiều trẻ em trên thế giới tận hưởng việc sáng tạo Robot như là một niềm vui, nhiều người lớn cũng như vậy, chúng tôi gọi đó là “có một sở thích” thay vì chỉ là “chơi với đồ chơi”. Tuy nhiên, bộ NXT còn hơn cả

đồ chơi, giáo viên ở các cấp học sử dụng bộ dụng cụ này để dạy khoa học và công nghệ. LEGO thậm chí còn có riêng một bộ phận chuyên nghiên cứu giáo dục để cung cấp các nguồn lực cho giáo viên. Điều tuyệt vời là bạn có thể sử dụng công cụ này để thực hiện sáng tạo khoa học một cách thực sự, một nhóm sinh viên đã gởi 9 Robot NXT lên không gian có độ cao lớn với mặt nước biển để thực hiện một số thí nghiệm. Cùng với việc rất tiện lợi và dễ sử dụng, NXT cực kỳ linh hoạt và đủ mạnh mẽ để các nhà khoa học ở mọi lứa tuổi nghiên cứu. Giới hạn chỉ tồn tại trong trí tưởng tượng của bạn.

Để xây dựng Robot, phải cần những phần sau :

- Thông tin cảm biến(Sensor) : một Robot cần phải lấy thông tin từ môi trường xung quanh nó.
- Chương trình(Programing) : Bộ não Robot sẽ sử dụng các thông tin được lập trình để đưa ra quyết định.
- Hành động(Action) : Robot cần phải có các bộ phận chuyển động để thực hiện mệnh lệnh.



Hình 1.1 Tổng quan về LEGO

Ngoài ra, để lắp ráp được một Robot, cần phải các thành phần để tạo nên “bộ phận truyền động”, “bộ phận làm việc”. Đó là các thanh, dầm, bánh răng, bánh đai, xích...



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

CHƯƠNG 2 : GIỚI THIỆU VỀ KHỐI LẮP RÁP LEGO

I. Giới thiệu:

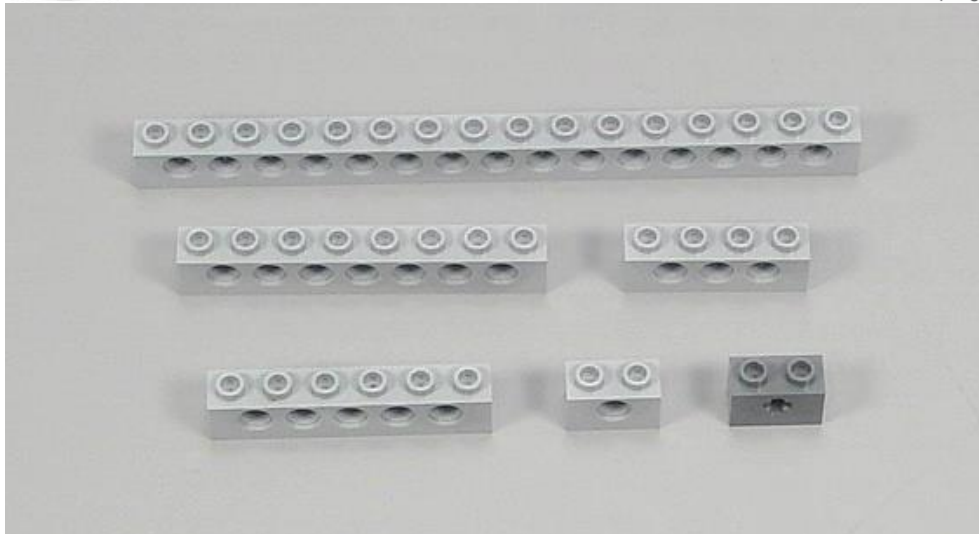
Trước khi vào thế giới Robot của LEGO, có một vài thuộc tính hình học của các gạch LEGO mà bạn cần phải biết và hiểu. Bạn sẽ không cần phải kiểm tra bằng các phép toán phức tạp hoặc lượng giác. Chúng ta sẽ chỉ thảo luận về một vài khái niệm rất đơn giản và giải thích một số thuật ngữ mà nó sẽ khiến cho việc lắp ráp hệ thống ở thực tế trở nên rất dễ dàng ngay từ ban đầu.

Bạn sẽ khám phá ra đơn vị mà LEGO sử dụng để thể hiện kích thước, như thế nào là tỉ lệ của các loại thanh, tấm, và làm thế nào điều này ảnh hưởng đến việc bạn kết hợp các thanh với hướng khác nhau vào một cấu trúc.

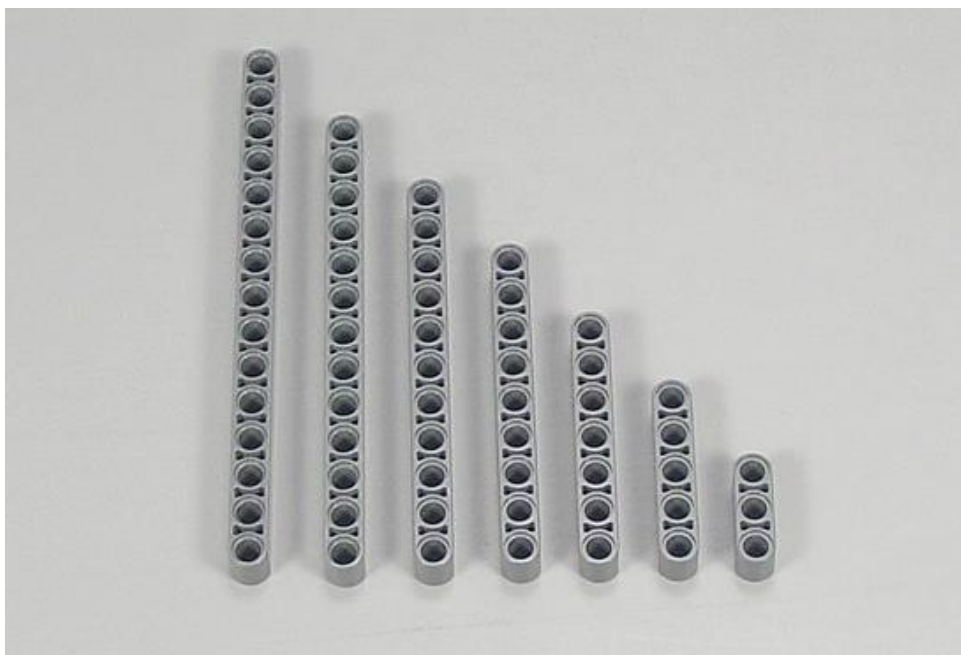
II. Giới thiệu cơ bản về các thanh, gạch, cấu kiện LEGO



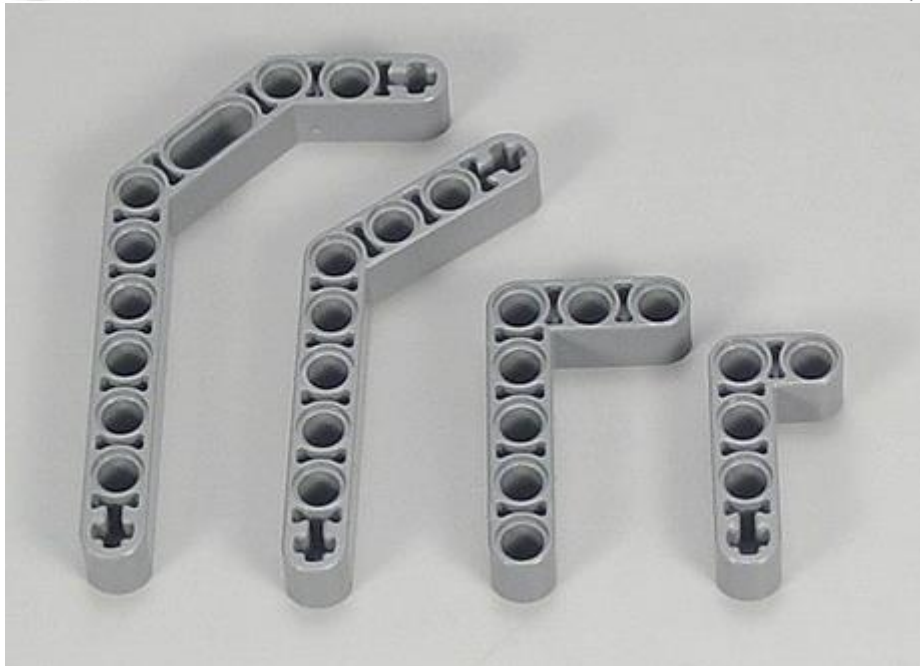
Các khối “gạch”(Bricks) : khối có các hình mũ đinh(Studs)



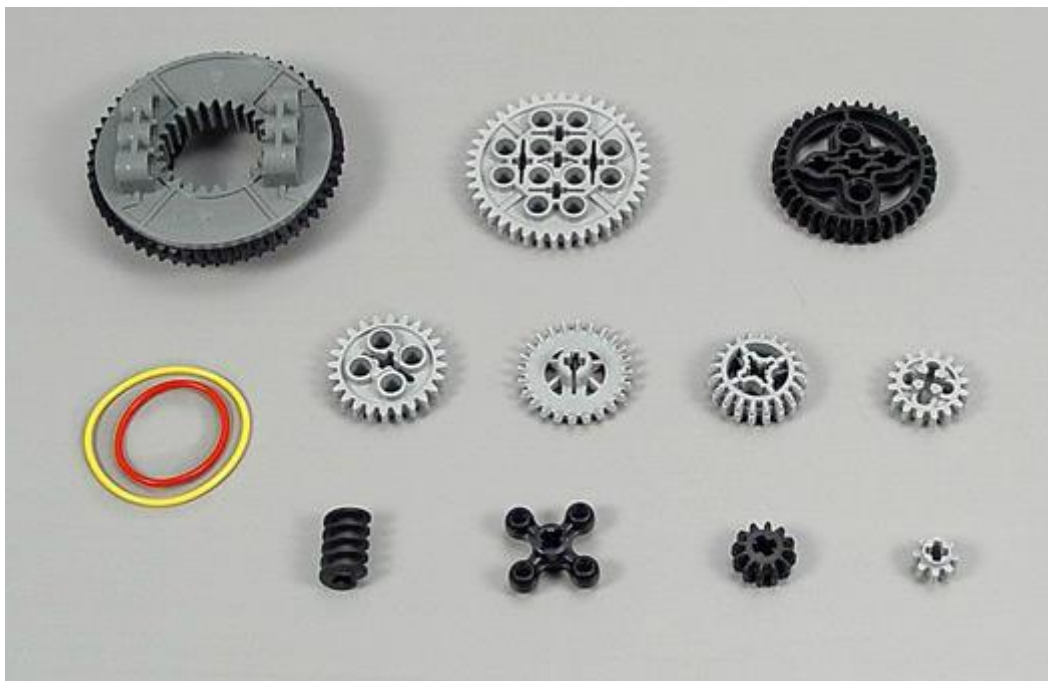
Hình 2.1 Một số loại gạch



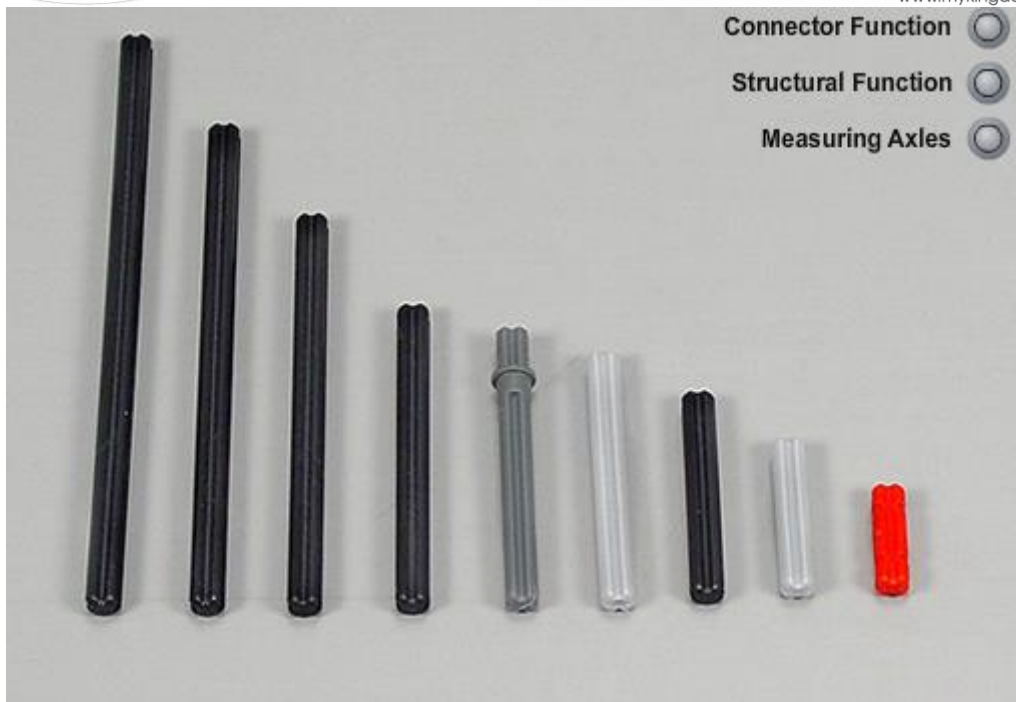
Hình 2.2 Các loại thanh(Beam)



Hình 2.3 Các loại thanh có góc.



Hình 2.4 Các loại bánh răng(Gear) và dây cao su(Rubber).



Hình 2.6 Các loại trục



Hình 2.7 Đai ốc giữ.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



Hình 2.8 Các loại chốt, cấu kiện kết nối

WHEELS and TREADS



Hình 2.14 Các loại bánh xe, xích.

III. Thể hiện kích thước và đơn vị :

III.1 Kích thước LEGO cơ bản :

LEGO xây dựng các kích thước thông thường của thành phần LEGO với 3 yếu tố : chiều dài, rộng và cao lần lượt theo thứ tự. Khi biểu diễn kích thước, chúng ta sẽ luôn đề cập đến hướng, ngay cả khi gạch bị lật úp hoặc xoay trong không gian ba chiều.

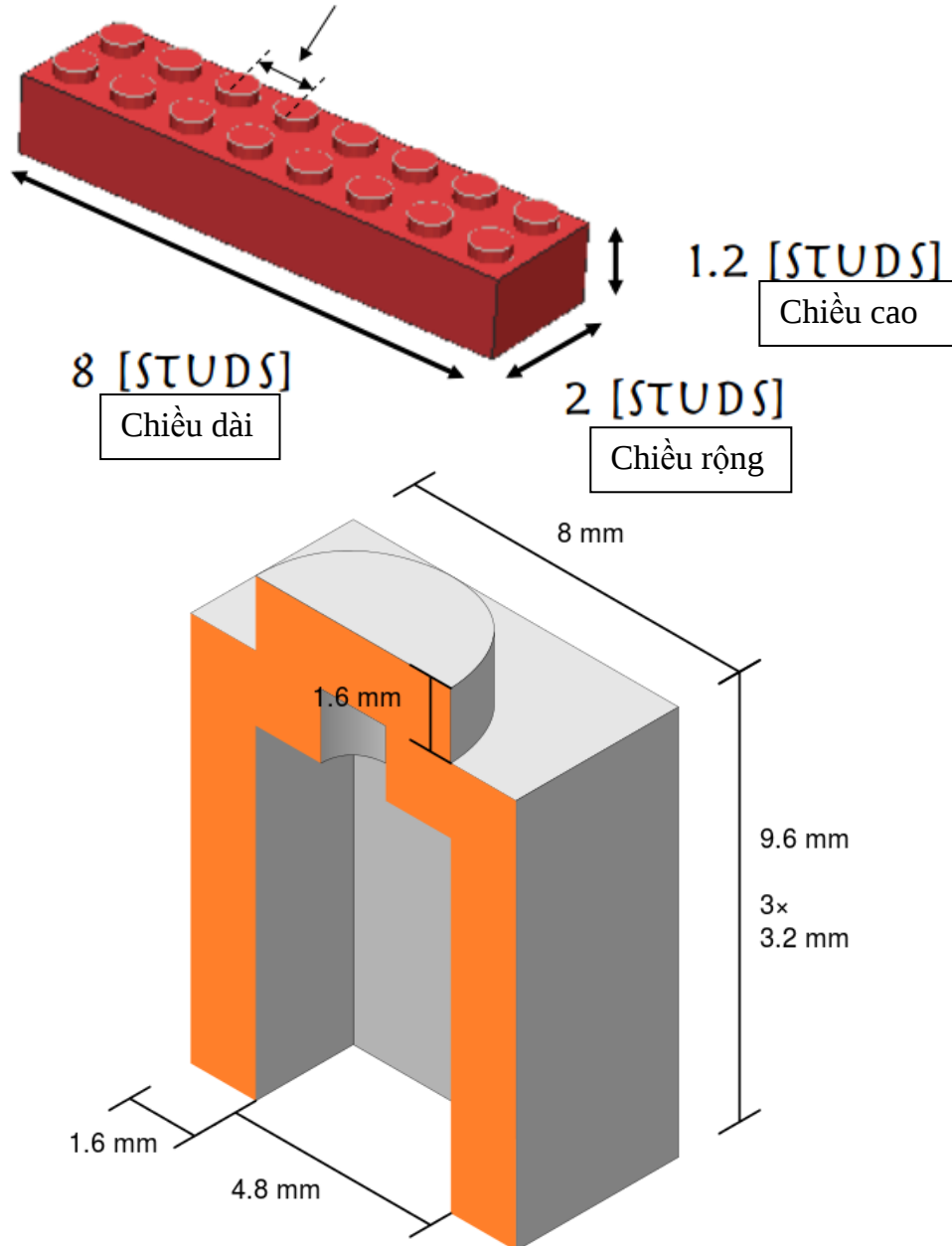
Thuộc tính đơn giản nhất để xác định là “Chiều cao”, đó là khoảng cách thẳng đứng giữa đỉnh và đáy của gạch. Chiều rộng, theo quy ước, là ngắn hơn trong hai kích thước nằm trong mặt phẳng ngang(chiều dài là kích thước còn lại). Cả chiều cao, chiều rộng và chiều dài được thể hiện bằng “studs”, hay được gọi là đơn vị LEGO.

*”Studs” : có thể hiểu là đầu đinh tán, là phần mũ tròn lồi lên ở các gạch LEGO.

*1 Studs = 8mm.

*Tiểu đơn vị Studs(bằng 1/2 chiều cao của một tấm) = 1.6mm.

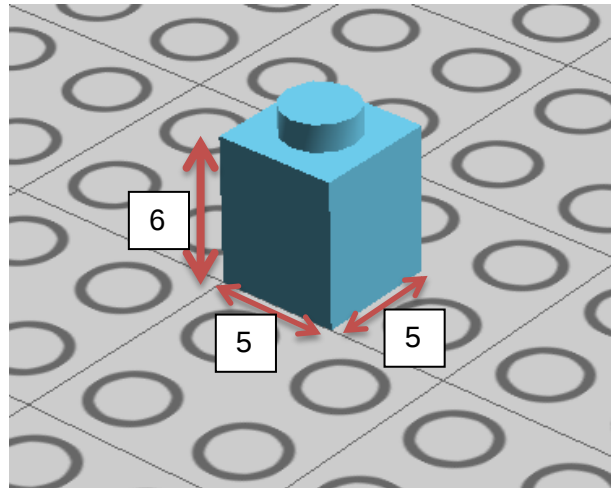
Khoảng cách 1 STUD



Hình 2.15 Đo đạc một khối gạch LEGO cơ bản.

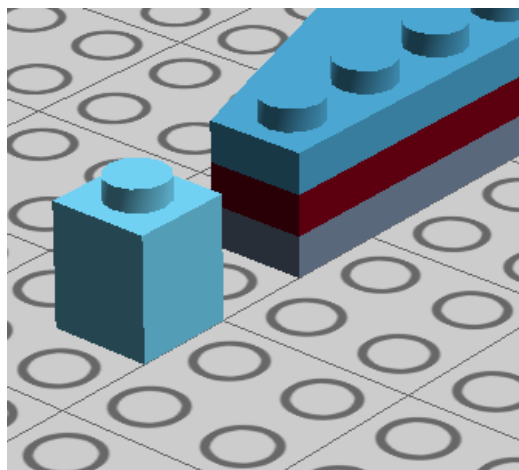
Để đo đạc khối LEGO, một đơn vị chiều ngang - tương ứng với 8mm(1 Studs), và theo đơn vị chiều cao, một đơn vị - tương ứng với 9,6mm(1.2 Studs). Những hình ảnh diễn tả dưới đây không quá quan trọng để nhớ - mà điều quan trọng là cách tính của hai loại kích thước là *khác nhau*. Tỉ lệ thậm chí còn quan trọng hơn : chia 9.6 cho 8 ta được 1.2(đơn vị theo chiều cao tương ứng với 1.2 lần đơn vị theo chiều ngang). Tỉ lệ này sẽ dễ dàng nhớ hơn nếu miêu tả theo dạng số hữu tỉ, tương đương với $\frac{6}{5}$.

Hình bên dưới thể hiện khối LEGO nhỏ nhất, được mô tả như một khối “đơn vị” LEGO là $1 \times 1 \times 1$.



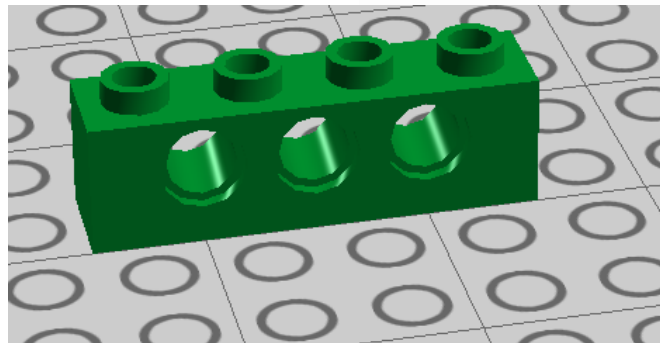
Hình 2.16 Tỷ lệ tương ứng $1 \times 1 \times 1$ của gạch LEGO

Hệ thống LEGO còn bao gồm một lớp của các thành phần mà chiều cao mỗi thành phần bằng $\frac{1}{3}$ của một khối gạch. Thành phần này là “tấm”, nếu xếp 3 “tấm” chồng lên nhau, chúng ta sẽ có chiều cao của một viên gạch cơ bản (xem hình 2.17).



Hình 2.17 Ba tấm tạo nên chiều cao của một gạch cơ bản.

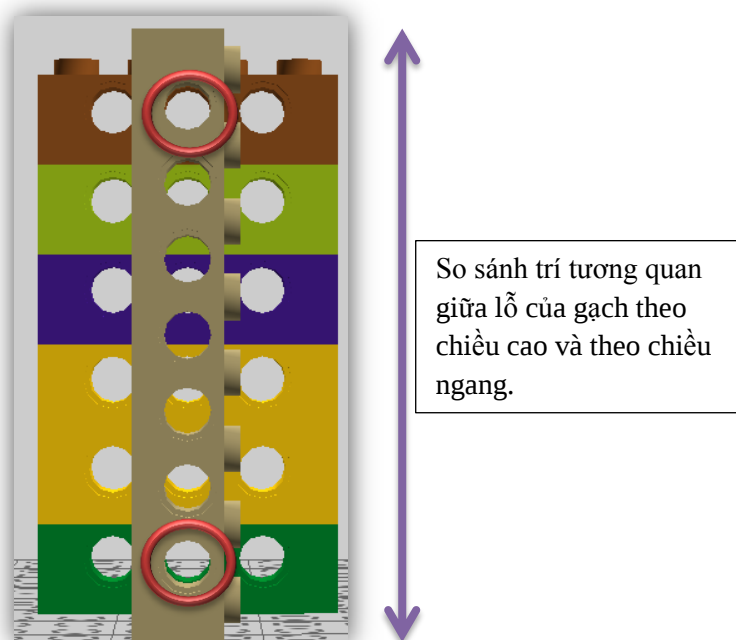
III.2 Khoá thanh dọc :



Hình 2.18 Gạch LEGO

Giả sử bạn muốn gắn một thanh gạch thẳng đứng, để nối hai hoặc nhiều lớp gạch ngang. Ở đây chúng ta cần phải nhớ **tỉ lệ 6 và 5** như đã nêu ở trên - còn được hiểu là *tiểu đơn vị Studs*.

Các lỗ bên trong gạch được đặt cách nhau một khoảng chính xác bằng *một đơn vị chiều ngang*, và lỗ thì nằm giữa hai “Studs” nếu tính về vị trí tương quan. Khoảng cách của các lỗ theo đơn vị chiều ngang thì bằng *5 tiểu đơn vị Studs*, theo đơn vị chiều cao là *6 tiểu đơn vị Studs* do đó khi xét theo chiều ngang thì khoảng cách các lỗ lần lượt đến lỗ ban đầu sẽ là 5,10,15,20,25,30..., còn khi xét về chiều cao thì khoảng cách các lỗ lần lượt đến lỗ ban đầu sẽ là 6,12,24,30,36.... Vì vậy nên khi xếp chồng gạch lên nhau và dùng thanh gạch để kết nối, thì các lỗ của *thanh gạch* và *gạch xếp chồng* sẽ không ăn khớp hoàn toàn, nếu xếp hai lỗ đầu tiên trùng nhau, thì đến *lỗ thứ 6 kế tiếp của thanh thẳng đứng* hoặc *lỗ thứ 5 kế tiếp của các gạch xếp chồng*, hai lỗ mới trùng nhau, đơn giản là vì có bội số chung = 30, xem hình sau :



Hình 2.19 Sự trùng khớp theo lỗ của gạch theo phương thẳng đứng và phương ngang.

Bây giờ bạn có thể xếp chồng các gạch lên nhau và nối chúng bằng một thanh gạch khác để xác thực nguyên tắc này trong thực tế.



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT: (08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

Kỹ thuật kết nối gạch như trên là rất quan trọng, vì nó cho phép chúng ta xây dựng những mô hình vững chắc, bởi vì thanh gạch thẳng đứng sẽ khoá tất cả các gạch nằm ngang ở giữa.



Hình 2.20 Mỗi năm lần gạch xếp chồng lên nhau thì lỗ sẽ trùng nhau.

Một trong những sơ đồ nhỏ gọn nhất cho phép bạn khoá các lớp ngang với thanh thẳng đứng được thể hiện trong hình 2.21 : một gạch và hai tấm, ở đây tương ứng với *năm tấm* (do chiều cao của ba tấm thì bằng chiều cao của một thanh gạch), như vậy xếp năm tấm (tương đương với khoảng cách của 2 lỗ) là cách rút gọn để bạn kết nối để cố định gạch.



Hình 2.21 Sơ đồ khoá dầm ngang

Ở đây chiều cao của gạch (6 tiểu đơn vị Studs) và thêm chiều cao của hai tấm (4 tiểu đơn vị Studs), vậy ta có tổng = 10. So sánh với khoảng cách của hai lỗ trên thanh (5 + 5 = 10 tiểu đơn vị Studs). Như vậy là đã khớp về kết quả, do đó sự xuất hiện của hai tấm sẽ giảm khoảng cách kết nối lại và tăng cường độ cứng vững.

Tiếp tục gia tăng khoảng cách, công việc tiếp theo : kết hợp nhiều hơn. Ở đây có nhiều cách để kết hợp, xem hình 2.22 :



Hình 2.22 Các dạng kết hợp cơ bản.

Câu hỏi đầu tiên : có phải có một dạng kết hợp tốt nhất ? Đúng vậy, kết cấu vững chắc nhất là cách thứ nhất từ trái qua trong hình 2.22, đó là bội số của sơ đồ cơ bản trong hình 2.21, bởi vì nó cho phép bạn khoá gạch qua điểm trung gian. Do đó, khi bạn thiết kế các mô hình, công thức 1 gạch + 2 tấm + 1 gạch + 2 tấm...là một cách khiến cho mọi việc dễ dàng hơn. Sự kết nối được lặp lại ở mỗi hai lỗ của thanh thẳng đứng. Như vậy bạn có nên *chỉ sử dụng* kết cấu này ? Dĩ nhiên là không ! Đừng hạn chế trí tưởng tượng của bạn với những ràng buộc không cần thiết. Đây chỉ là một mẹo hữu ích trong nhiều trường hợp, đặc biệt là khi bạn bắt đầu lắp ráp cái gì đó và không biết phải làm như thế nào! Trong nhiều trường hợp khác chúng ta sẽ sử dụng nhiều phương án khác nhau.

III.3 Thanh giằng chéo.

Bây giờ bạn đã biết rằng bạn có thể kết nối một chồng gồm tấm và gạch với một thanh khác. Và bạn có biết được nó hoạt động bằng cách nào theo quy tắc số học. Vậy làm cách nào bạn kết nối các gạch xếp chồng lên nhau với một thanh chéo ?

Bạn phải nhìn vào thanh chéo như thể nó là cạnh huyền của một tam giác, xem hình 2.23. Bây giờ tiến hành đo hai cạnh của nó, đáy của tam giác là 6 Studs, chiều cao là 8 Studs, ở cạnh huyền, ta đếm được 10 Studs.

Một trong những định lý nổi tiếng, Pythagoras đã chứng minh có một sự liên hệ giữa các cạnh bên trong tam giác vuông. Gọi hai cạnh góc vuông lần lượt là A và B, cạnh huyền là C, ta có mối quan hệ :

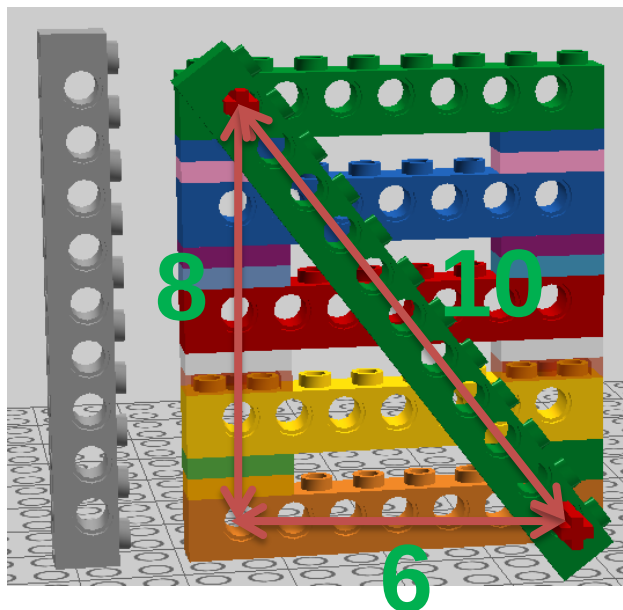
$$A^2 + B^2 = C^2$$

Bây giờ hãy kiểm tra với các số liệu của chúng ta :

$$6^2 + 8^2 = 10^2$$

Khai triển đẳng thức trên, ta được :

$$36 + 64 = 100 \text{ (đúng)}$$



Hình 2.23 Định lý Pytagoras.

Vậy ví dụ của chúng ta là chính xác so với định lý. Bây giờ hãy xem xét một vài ví dụ sau đây :

A(cạnh góc)	B(Cạnh cao)	A^2	B^2	$A^2 + B^2$	Kết quả
5	6	25	36	61	Không tương thích
3	8	9	64	73	Không tương thích
3	8	9	16	25	Đúng, $25 = 5 \times 5$
15	8	225	64	289	Đúng, bởi vì $289 = 17 \times 17$.
9	8	81	64	145	145 không phải là số chính phương, nhưng nó gần với $144(12 \times 12)$, sự sai lệch về kích thước khoảng gần với 1%.

Bảng 2.1 Một số ví dụ dẫn xuất từ định lý Pytagoras.

Tại thời điểm này, có lẽ bạn đang tự hỏi chắc phải luôn có máy tính bỏ túi trên bàn mỗi khi chơi LEGO, và có lẽ phải tìm sách giáo khoa để đọc lại các kiến thức về hình học ! Đừng lo lắng điều đó, bởi vì :

- Thứ nhất, bạn không nhất thiết phải sử dụng các thanh chéo quá thường xuyên.
- Thứ hai, hầu hết các kết hợp hữu ích xuất phát từ bộ ba cơ bản 3-4-5, nếu nhân cả hai vế với 2, ta được 6-8-10, tương tự khi ta nhân với 3, ta có 9-12-15, và cứ như vậy. Rất dễ để kết hợp và rất dễ để nhớ.

Chúng tôi mong muốn bạn dành thời gian để lắp ráp các hình tam giác, thử nghiệm kết nối các cạnh với các góc khác nhau và đánh giá độ cứng của nó. Kiến thức này sẽ là một chứng minh quý giá khi bạn bắt tay vào lắp ráp những cấu trúc phức tạp.

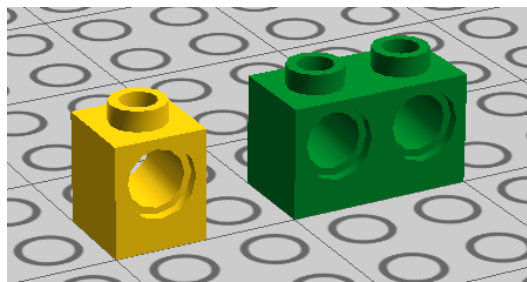


Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

III.4 Các kích cỡ và đơn vị theo chiều ngang :

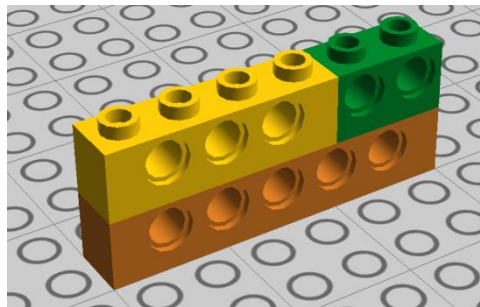
Đến đây chúng ta đã đặt rất nhiều sự chú ý vào mặt phẳng thẳng đứng, bởi vì kỹ thuật khoá các lớp với nhau bởi thanh thẳng đứng là công cụ quan trọng nhất để bạn xây dựng các mô hình khối. Tuy nhiên có một số ý tưởng bạn sẽ thấy hữu dụng khi bạn sử dụng khối gạch để lắp ghép trong mặt phẳng ngang.

Chúng ta đã nói rằng đơn vị đo chiều dài là “Studs”, có nghĩa là sẽ đo chiều dài dựa trên số “Studs” của gạch đó. Các lỗ trên gạch được đặt cách nhau cùng một khoảng cách, hãy nhìn vào gạch, bạn sẽ nhận ra rằng các lỗ và đầu “Studs” được đặt xen kẽ nhau, và trên suốt chiều dài của gạch, thì số lỗ nhỏ hơn số lượng “Studs” một đơn vị. Tuy nhiên ở một vài trường hợp khác, gạch 1 lỗ và 2 lỗ thì có số lượng “Studs” và lỗ bằng nhau, (xem hình 2.24).



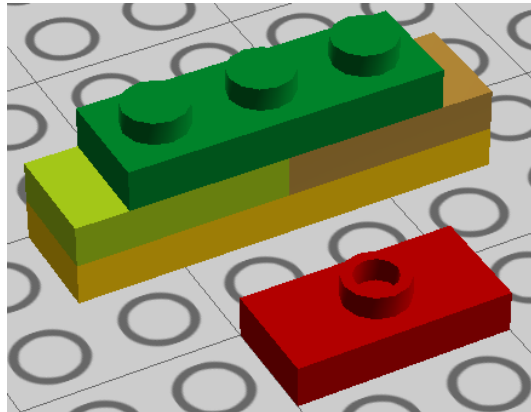
Hình 2.24 Gạch 1 lỗ và 2 lỗ.

Trong một số thanh ngắn, các lỗ không được đặt xen kẽ với các “Studs” mà nằm ngay bên dưới “Studs”, và khi sử dụng chúng cùng với các thanh tiêu chuẩn, chúng sẽ cho phép tăng khoảng cách thêm $\frac{1}{2}$ đơn vị chiều ngang, (xem hình 2.25).



Hình 2.25 Cách gia tăng khoảng cách thêm $\frac{1}{2}$ lỗ.

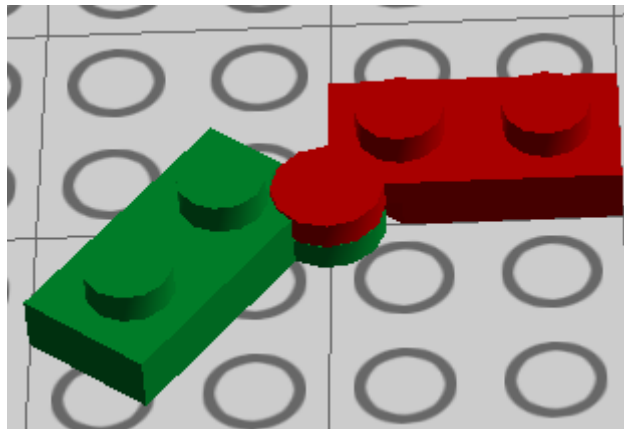
Một mảnh khác cũng có chức năng tương tự là tấm 1x2 với một “Studs”. (Xem hình 2.26), nó rất hữu ích khi bạn muốn điều chỉnh thêm một nửa “Studs”



Hình 2.26 Tấm đơn 1x2

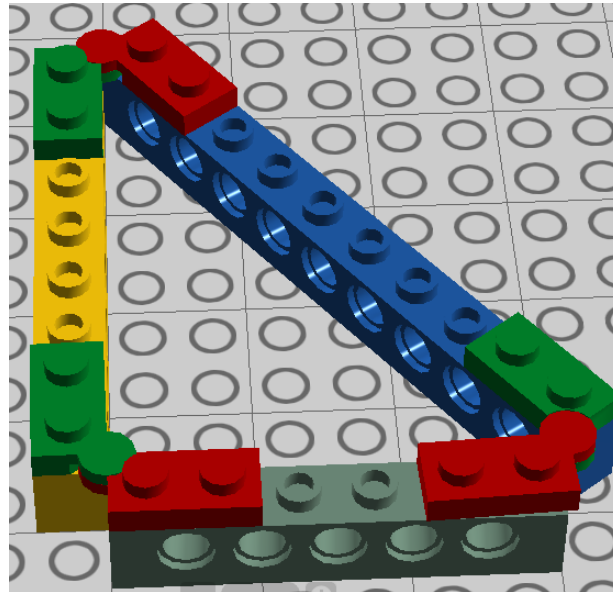
III.5 Kết nối với bản lề :

Chúng ta quay lại hình tam giác, ở đây không có gì thực sự mới, mà chỉ là các cách khác nhau để xây dựng từ các khái niệm trước. Đầu tiên chúng ta cần phải có một bộ phận đặc biệt : bản lề (xem hình 2.27), sử dụng khớp bản lề bạn có thể lắp ráp được rất nhiều hình tam giác, ở đây chúng ta tiếp tục sử dụng tam giác vuông góc.



Hình 2.27 Khớp bản lề LEGO

Khớp bản lề LEGO cho phép bạn xoay các thanh kết nối, giữ cho các góc bên trong luôn tiếp xúc. Vì vậy, khi sử dụng ba bản lề, bạn sẽ tạo được hình tam giác có đỉnh rơi tại các tâm xoay của bản lề, chiều dài của cạnh tam giác chính là chiều dài của thanh bên trong (xem hình 2.28), khi đã quen thuộc với định lý Pythagoras, thì trường hợp này là một ứng dụng. Cách kết hợp là giống nhau khi chúng ta sử dụng trong các trường hợp : 3-4-5; 6-8-10 và cứ tiếp tục như vậy.



Hình 2.28 Tạo tam giác với ba bản lề

Bạn có thể thấy rằng sẽ không quá khó khăn khi đã làm quen từ những điều cơ bản. Đầu tiên, nó sẽ giúp bạn làm cách nào để định nghĩa khối gạch bằng các tỉ lệ của nó, xác định chiều dài và chiều rộng bằng số “Studs”. Bạn đã biết rằng, với mỗi chiều cao của 5 khối gạch, thì vị trí của lỗ trên thanh khoá sẽ phù hợp. Ngoài ra, bởi vì ba tám thì bằng với chiều cao của một khối gạch, nên sơ đồ khoá đơn giản nhất là sử dụng sự xếp chồng của một khối gạch và hai tám, và cứ chồng lên như vậy bạn sẽ có nhiều vị trí lỗ phù hợp. Để khoá bằng thanh chéo, ứng dụng định lý Pythagoras, từ bộ ba cơ bản 3-4-5, nhân thêm bội số để nhận được các số liệu phù hợp cho mô hình lắp ráp của bạn.

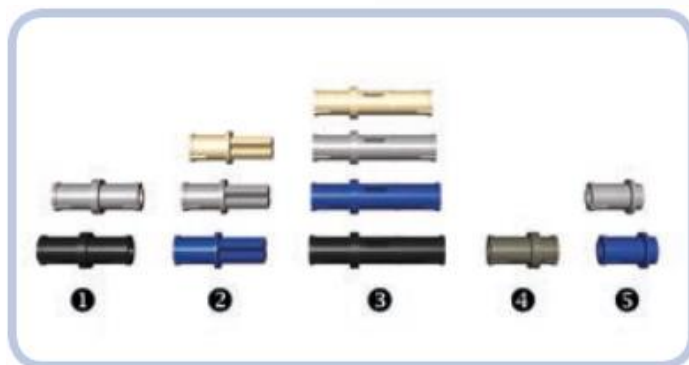
IV. Chốt kết nối.

Chốt kết nối là rất quan trọng để lắp ráp một hệ thống, nó giữ cho các thanh kết nối với nhau. Chốt sẽ khớp vào lỗ chốt hoặc lỗ trục để kết nối hai hoặc nhiều hơn các thanh liền kề. Như bạn thấy trong hình 2.29, các loại chốt kết nối khác nhau về chiều dài và hình dạng, cũng như là mức độ ma sát.



Hình 2.29 Các chốt phổ biến nhất.

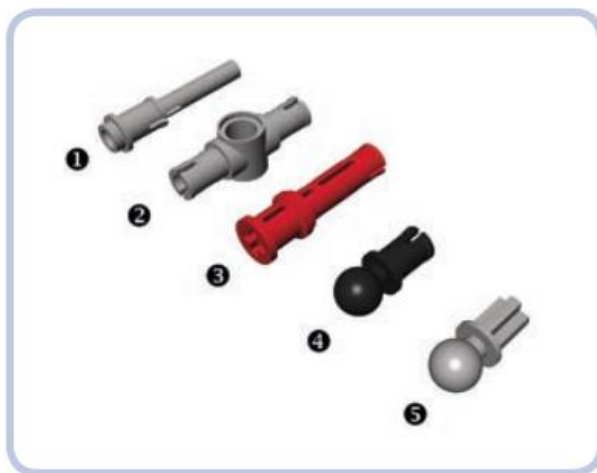
Giống như trục, chốt cũng được sử dụng để xuyên qua lỗ của các loại thanh, gạch. Tuy nhiên, ở thân chốt có vòng đai để cố định chốt khi gim vào lỗ của thanh. Các nhóm chốt khác nhau được thể hiện trong hình bên dưới.



Hình 2.30 Bộ sưu tập các loại chốt.

1. Chốt : thường(màu xám), ma sát cao(màu đen).
2. Chốt trục : thường(vàng nhạt, xám), ma sát cao(xanh).
3. Chốt dài : thường(vàng nhạt, xám), ma sát cao(xanh, đen).
4. Chốt $\frac{3}{4}$: thường (xám tối).
5. Chốt $\frac{1}{2}$: thường (xám, xanh).

Bây giờ bạn đã xem những loại chốt phổ biến nhất, tiếp theo hãy xem xét những loại chốt ít phổ biến hơn. Hình 2.31 thể hiện một số loại chốt chuyên dụng. Trong hầu hết trường hợp, đây là biến thể của các loại chốt cơ bản được sửa đổi cho phù hợp với các kết nối cụ thể.



Hình 2.31 Một số chốt chuyên dụng.

Một số chốt có hai biến thể : một là có thể xoay tự do bên trong lỗ chốt và đòi hỏi phải có lực tác động. Loại thứ hai là loại chốt có ma sát cao. Hai biến thể này được phân biệt bởi màu sắc khác nhau.

1. $\frac{1}{2}$ chốt với thanh 2L, xem hình 2.32.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



Hình 2.32 Hình người cầm thanh 2L.

2. Chốt dài với lỗ giữa : thường(xám, đen hoặc đỏ);
3. Chốt dài với khớp trục chữ thập : dùng để kết nối giữa lỗ và trục.
4. Chốt thường với đầu hình cầu : phục vụ ở các điểm gắn kết của LEGO. Xem hình 2.33



Hình 2.33 Hai chốt đầu hình cầu được kết nối để có thể di chuyển tương đối với nhau.

5. Chốt trục với đầu hình cầu ma sát nhỏ.

PHẦN 2 : ĐỘNG CƠ

CHƯƠNG 3 : TỔNG QUAN VỀ ĐỘNG CƠ LEGO

I. Giới thiệu :

Động cơ điện là nguồn động lực của hầu hết các sáng tạo kỹ thuật, có thể được sử dụng cho hầu hết mọi thứ, từ việc tạo lực cho bánh xe quay, bẻ lái, nâng cao(xe nâng) hoặc thậm chí điều khiển các thành phần điện khác.

II. Nguyên lý hoạt động :

II.1 Khái niệm

Động cơ điện một chiều là động cơ sử dụng năng lượng điện một chiều để tạo ra chuyển động. Hoạt động dựa trên nguyên lý điện từ, khi đặt vào trong từ trường một dây dẫn và cho dòng điện chạy qua dây dẫn thì từ trường sẽ tác động một lực từ vào dòng điện(dây dẫn) và làm cho dây dẫn chuyển động. Động cơ điện biến đổi năng lượng điện thành cơ năng.

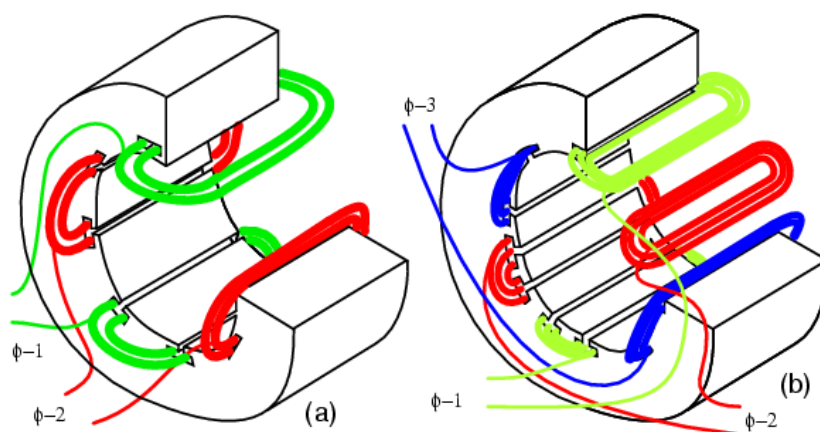
II.2 Cấu tạo :

Gồm hai phần chính :

- ❖ Phần đứng yên(tĩnh) – Stator.
- ❖ Phần chuyển động – Rotor.

Phần tĩnh(Stator) hay còn gọi là phần kích từ động cơ, là bộ phận sinh ra từ trường, gồm có mạch từ và dây quấn.

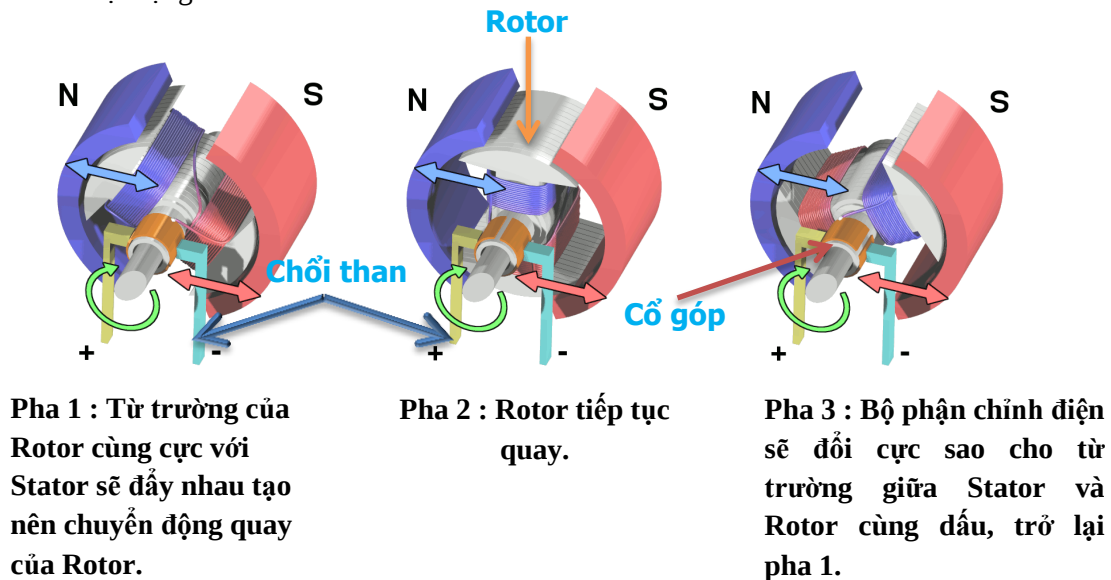
- Mạch từ được làm bằng sắt từ(thép đúc, thép đặc).
- Dây quấn được làm bằng dây điện từ.



Hình 3.1 Sơ đồ mô tả cấu tạo phần tĩnh.

Phần ứng(Rotor) dùng để dẫn từ. Thường dùng những tấm thép kỹ thuật điện phủ cách điện mỏng ở hai mặt rồi ép chặt lại với nhau. Trên lá thép có dập hình dạng rãnh để sau khi ép lại thì đặt dây quấn vào.

II.3 Nguyên tắc hoạt động :



Hình 3.2 Các pha hoạt động của động cơ DC.

Theo nguyên tắc từ trường : cùng cực thì đẩy nhau, khác cực thì hút nhau.

Từ trường của động cơ được tạo ra nhờ các cuộn dây có dòng điện một chiều chạy qua, các cuộn dây này gọi là cuộn cảm(hay cuộn kích từ-Stator) và cuốn quanh các cực từ.

Từ trường do cuộn cảm tạo ra sẽ tác dụng một lực từ vào các dây dẫn Rotor và làm Rotor quay, cuộn dây này gọi là cuộn ứng, dòng điện đưa vào cuộn ứng qua chổi than và cổ góp.

Cổ góp được gắn cố định trên trục và được nối với các cuộn dây dẫn Rotor(cổ góp gồm hai mảnh ghép lại với nhau, mỗi mảnh nối với một cuộn dây), do đó, ở pha thứ ba, “bộ phận chỉnh điện sẽ đổi cực sao cho từ trường giữa Stator và Rotor cùng dấu”, nghĩa là lúc này, sau khi quay nửa vòng, điện trong hai cuộn dây Rotor sẽ đổi chiều với nhau. Như vậy nguyên tắc từ trường, cùng cực đẩy nhau, khác cực hút nhau, do đó từ trường trong Stator sẽ đẩy Rotor quay và cứ tiếp tục như vậy.

Theo sự tiến bộ của khoa học kỹ thuật, các động cơ một chiều sau này có công suất nhỏ không sử dụng hệ thống chuyển mạch bằng chổi than – cổ góp nữa mà sử dụng hệ thống chuyển mạch bằng điện tử.

III. Giới thiệu động cơ LEGO

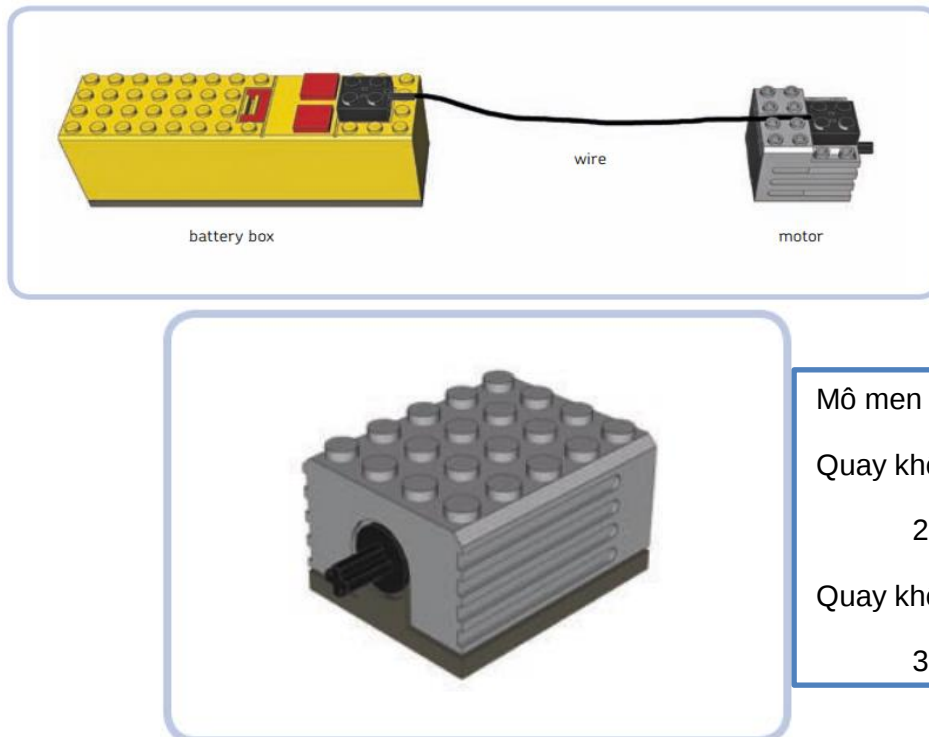
LEGO đã chế tạo động cơ điện từ năm 1965, và có thể phân thành ba thế hệ. Thế hệ đầu tiên là động cơ 4.5v, nhưng nó đã cũ và yếu hơn khi so sánh với các loại động cơ mới hơn. Vào năm 1990, LEGO giới thiệu thế hệ động cơ thứ hai, được vận hành với dòng điện 9V, nó tương đối lớn và không có bánh răng nội, vì vậy nên nó có tốc độ rất cao nhưng mô men xoắn thấp(được hiểu là lực quay của động cơ). Do đó sẽ không hiệu quả khi sử dụng trong các ứng dụng có tải cao vì đòi hỏi mô men xoắn lớn, cùng với việc sản sinh ra nhiệt năng khá nhiều.

***Mô men :** $M = F.d$ (lực nhân với khoảng cách từ tâm quay đến phương đặt lực)



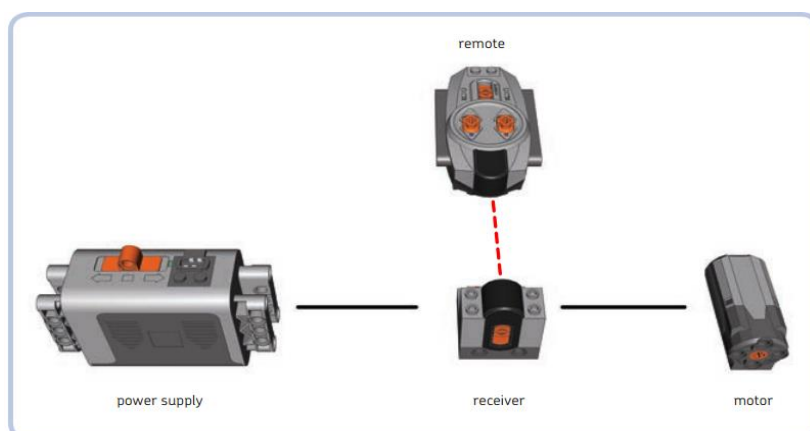
Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

***RPM(Revolutions Per Minute) : số vòng quay trong một phút.**



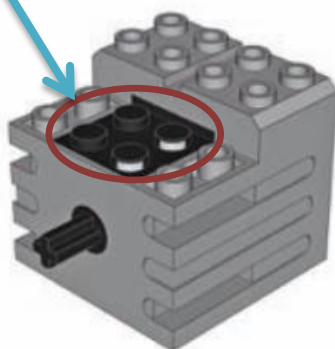
Hình 3.3 Động cơ 9V đầu tiên - 2838.

Loại thứ ba là động cơ có chức năng điều khiển từ xa, được giới thiệu năm 2007 (hình 3.4), mô men xoắn cao hơn và được tối ưu hoá để chịu tải trọng cao. Chức năng của động cơ cũng cho phép điều khiển đa dạng hơn so với việc chỉ quay rồi đảo ngược chiều.



Hình 3.4 Động cơ điều khiển từ xa

Khu vực kết nối
với nguồn điện



Mô men xoắn : 2.25 N*cm

Quay không tải 7V :

160 RPM

Quay không tải 9V:

250 RPM

Hình 3.6 Động cơ 71427 – động cơ 9V phổ biến và mạnh mẽ do có bánh răng giảm tốc bên trong.

Động cơ 71427 được sử dụng trong bộ MINDSTORM RCX.

Động cơ NXT(hình 3.8) được thiết kế đặc biệt cho các bộ MINDSTORM NXT, có mô men xoắn cao nhất tính tới thời điểm nó được chế tạo và tiêu thụ điện năng cao, nó có một *cảm biến* góc quay với độ chính xác đến 1 độ, điểm này rất hữu ích khi thiết kế Robot yêu cầu chính xác. Động cơ được điều khiển tốc độ, góc quay, chiều quay thông qua “bộ não” NXT. Và đây chính là động cơ chúng ta sẽ sử dụng trong các Robot.

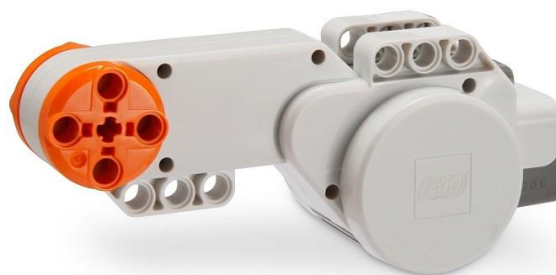
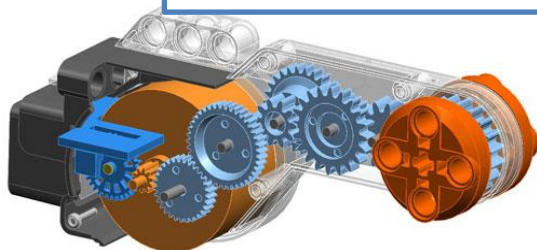
Mô men xoắn : 16.7 N*cm

Quay không tải 7V :

82 RPM

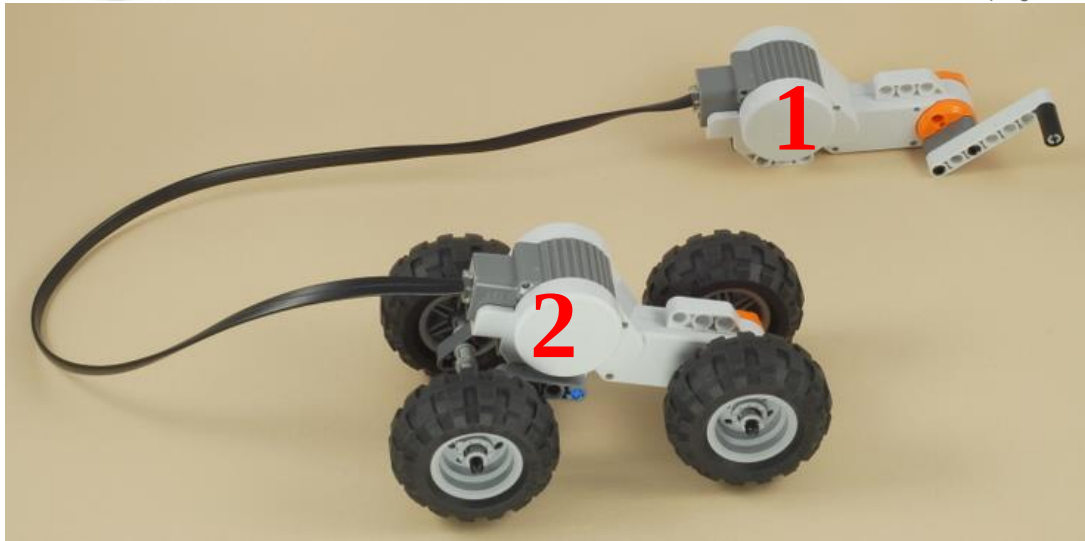
Quay không tải 9V:

117 RPM



Hình 3.9 Động cơ NXT

Hệ thống bánh răng giảm tốc trong động cơ là nguyên nhân khiến động cơ có mô men xoắn lớn. Ngoài việc được điều khiển từ NXT, khi kết nối hai động cơ lại với nhau. Theo nguyên tắc từ trường, nếu ta dùng lực quay động cơ(1) từ bên ngoài, thì lúc này Rotor sẽ tạo ra từ trường tác động ngược lại Stator, dòng điện này theo dây dẫn đến Stator của động cơ(2) và làm cho Rotor của động cơ(2) quay.



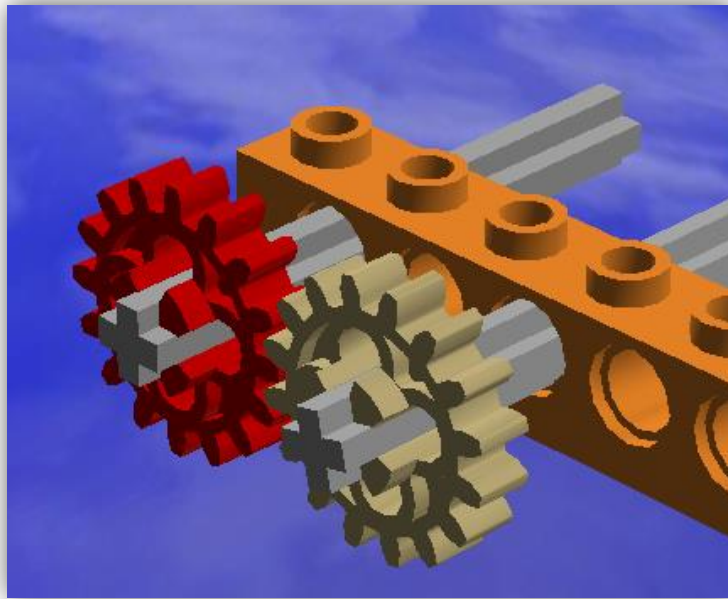
Hình 3.10 Truyền năng lượng điện giữa hai động cơ.

PHẦN 3 : CÁC CƠ CẤU TRUYỀN ĐỘNG

CHƯƠNG 4 :TRUYỀN ĐỘNG BÁNH RĂNG.

I. Giới thiệu :

Bánh răng là một bộ phận cơ khí, có hình tròn cùng với một trục đi qua tâm và có các múi răng trên đường chu vi của nó.



Hình 4.1 Hai bánh răng ăn khớp.

Nếu kết hợp hai bánh răng A và B lại với nhau sao cho một số múi răng của bánh răng A ăn khớp với một số múi răng của bánh răng B, thì khi một bánh chuyển động(ví dụ A), thì các múi răng của bánh răng A sẽ đẩy vào các múi răng của bánh răng B và làm cho bánh răng B chuyển động theo.

II. Các đặc điểm cần lưu ý :

Cơ cấu bánh răng (xem hình 4.1) thường được dùng để truyền chuyển động từ trục này qua trục khác. Hay nói cách khác trục của một bánh răng quay sẽ làm cho trục của bánh răng kia trong cơ cấu quay theo.

Lưu ý rằng hai trục này quay ngược chiều nhau: một cái quay theo chiều kim đồng hồ và cái kia thì quay theo chiều ngược lại.



Hình 4.2 Chiều quay của các bánh răng ăn khớp

Khi quay thì mỗi múi răng của bánh này sẽ khớp vào giữa 2 múi răng của bánh kia. Điều này có nghĩa là nếu bánh răng này quay được 1 múi răng thì nó cũng đồng thời làm quay 1 múi răng của bánh răng kia. Hay nói cách khác, trong cơ cấu bánh răng, thì số múi răng quay được của mỗi bánh răng là như nhau trong cùng một đơn vị thời gian. Tuy nhiên điều này không có nghĩa là cả hai bánh răng quay cùng với một vận tốc góc !

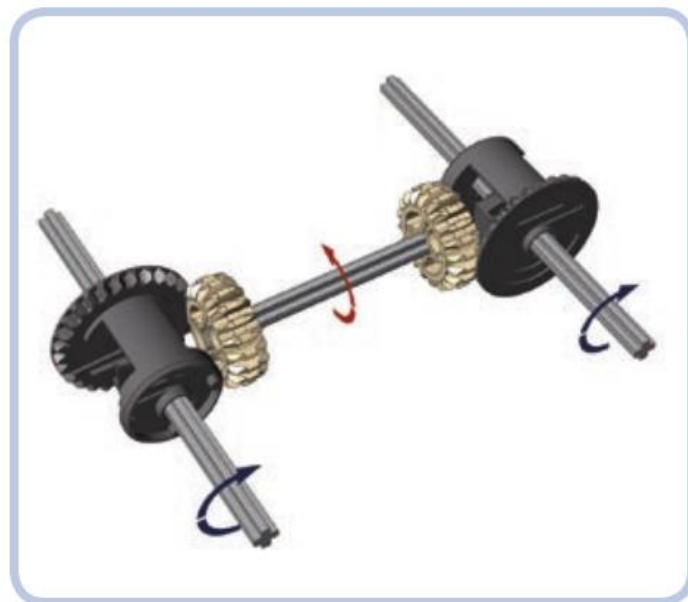
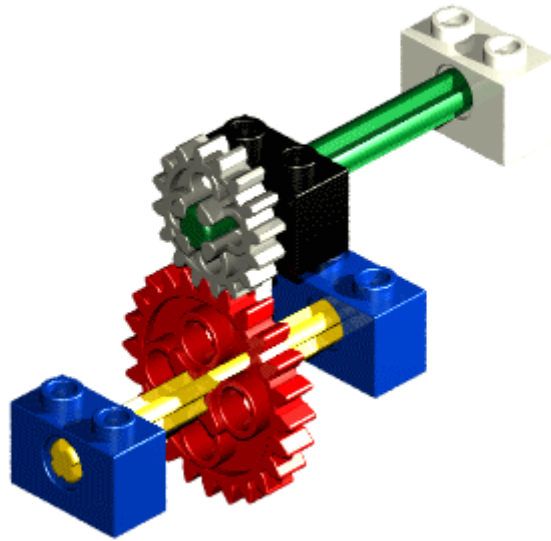
***Vận tốc góc :** số vòng quay của bánh răng(hoặc trục) trong một đơn vị thời gian.

Giả sử bánh răng A có 10 răng, bánh răng B có 20 răng. Khi A quay được 10 vòng, thì số lượt răng quay được của bánh A là : $10 \times 10 = 100$, ta lấy số lượt răng quay được của A chia cho số răng của B : $100/20=5$, vậy bánh răng B sẽ quay được 5 vòng. Như vậy tỉ số giữa các số múi răng có trên hai bánh răng sẽ cho ta biết được là cơ cấu bánh răng sẽ làm tăng hay giảm vận tốc góc của trục quay tương ứng. Trong một cơ cấu bánh răng, thì sẽ có một bánh chủ động và bánh bị động(bị kéo theo), như ví dụ trên, nếu A là bánh răng chủ động thì đây là cơ cấu giảm tốc, ngược lại B là bánh răng chủ động thì đây là cơ cấu tăng tốc.

III. Các ứng dụng của bánh răng thông thường :

III.1 Truyền chuyển động giữa hai trục song song

Với hiệu suất cao(ít bị mất mát lực giữa hai trục, hiệu suất xấp xỉ 98%). Nếu hai trục ở cách nhau xa hơn 2 bánh răng thì ta cũng có thể dùng thêm các bánh răng khác để nối khoảng trống. Tuy nhiên nên tính toán sao cho số lượng bánh răng là ít nhất để giảm thiểu sự tiêu hao năng lượng do lực ma sát.



Hình 4.3 Truyền chuyển động giữa hai trục song song.

III.2 Tăng hoặc giảm tốc độ quay của trục so với trục ban đầu :

Thường thì ta hay giảm tốc độ để tăng lực quay, ví dụ khi xe cần đi lên mặt đường nghiêng với độ dốc cao, hoặc là khi cánh tay của robot cần nhấc các vật nặng lên cao, v.v... Hoặc có khi ta cần phải giảm tốc độ để điều chỉnh vị trí của các bộ phận một cách chính xác hơn, dễ dàng hơn. Còn đối với những yêu cầu không cần dùng lực mà cần tốc độ cao, ta cần tăng tốc để đạt tới vận tốc quay tương ứng.

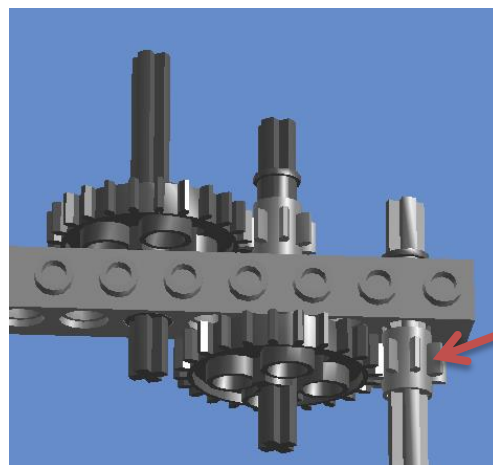
III.2.1 Chuỗi các bánh răng (geartrain):

Bộ Lego Mindstorm 9797 có các bánh răng thường với số lượng múi răng là 8(8t), 16 (16t), 24 (24t), và 40 (40t). Như vậy tỷ số giảm tốc lớn nhất mà ta có giữa 2 bánh răng là 8:40 hay 1:5. Nếu ta muốn có một tỷ số giảm tốc lớn hơn nữa thì có thể kết hợp nhiều các bánh răng lại với nhau tạo thành một cơ cấu giảm tốc nhiều tầng, hay còn gọi là chuỗi các bánh răng.



Hình 4.4 Các loại bánh răng với số răng tương ứng

Ví dụ : nếu ta muốn giảm tốc độ quay xuống còn $\frac{1}{9}$ vận tốc quay ban đầu thì ta có thể kết hợp 2 cặp bánh răng 8t-24t (có tỷ số giảm tốc mỗi cặp là 1:3). Ta có $1:3 \times 1:3 = 1:9$. Hình vẽ sau cho ta một chuỗi bánh răng có khả năng làm giảm tốc độ quay xuống còn $\frac{1}{9}$, và đồng thời làm cho lực quay tăng lên 9 lần.

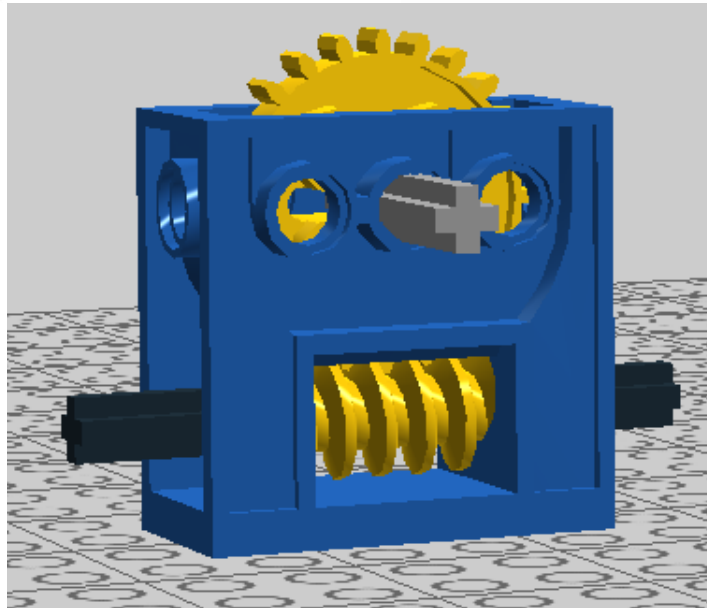


Bánh răng chủ động

Hình 4.5 Cụm bánh răng giảm tốc

IV. Một số cơ cấu bánh răng khác :

IV.1 Trục vít – bánh vít :



Hình 4.6 Cơ cấu trục vít – bánh vít

Khác với cơ cấu 2 bánh răng thông thường đã đề cập ở trên, cơ cấu trục vít – bánh vít là không đối xứng, nghĩa là ta có thể quay trục vít để làm cho bánh vít quay, nhưng ta không thể quay bánh vít để khiến cho trục vít quay được do lực ma sát quá lớn.

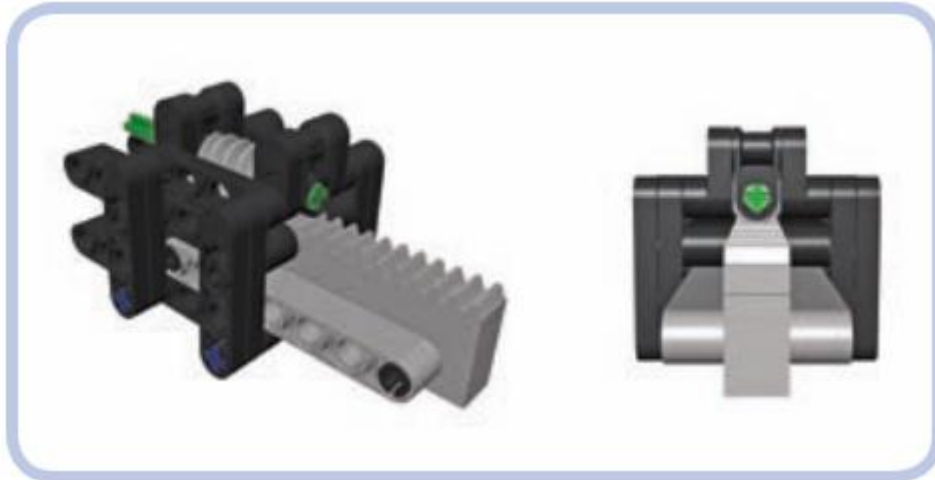
Cơ cấu này thường được dùng để truyền chuyển động giữa 2 trục vuông góc với nhau thuộc hai mặt phẳng khác nhau. Ta cũng hay dùng cơ cấu này để cản không cho một lực nào đó (thường là trọng lực) làm cho hệ thống quay theo chiều ngược lại khi động cơ ngừng. Ví dụ, nếu ta dùng cơ cấu bánh răng thường trong cánh tay máy để nâng một vật thì khi tắt động cơ, trọng lượng của vật nặng có thể khiến cho bánh răng quay theo chiều ngược lại làm cho cánh tay máy hạ xuống và vật đó cũng bị tuột xuống theo. Dùng cơ cấu trục vít – bánh vít thì sẽ không xảy ra trường hợp trên.

Vì trục vít quay 1 vòng chỉ làm bánh răng trong cơ cấu trục vít – bánh răng quay 1 răng, nên bánh vít được xem như tương ứng với một bánh răng thường có một răng! Do đó, cơ cấu này được dùng để giảm vận tốc quay và gia tăng lực theo tỷ lệ $1:T$, với T là số múi răng của bánh răng. Tóm lại, người ta hay dùng cơ cấu này để có thể điều chỉnh vị trí của các bộ phận một cách chính xác và ổn định trong điều kiện có tải trọng vừa phải. Lưu ý: cơ cấu này có sự tiêu hao năng lượng đáng kể do lực ma sát.

Ngoài ra, trục vít còn có thể ăn khớp với bánh răng côn, côn kép và thanh răng như sau :

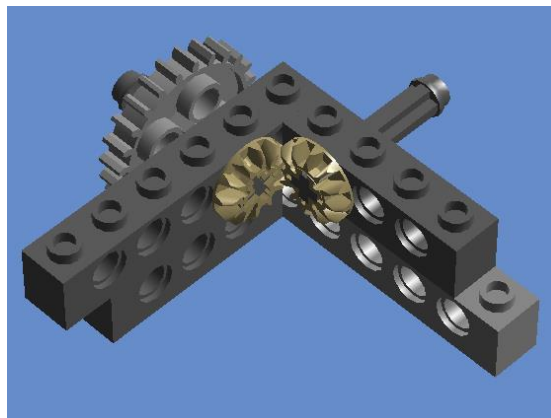


Hình 4.7 Ăn khớp giữa bánh răng côn, côn kép và trục vít.



Hình 4.8 Ăn khớp giữa thanh răng và trục vít.

IV.2 Bánh răng côn :



Hình 4.9 Bánh răng côn ăn khớp.

Cơ cấu bánh răng côn có khả năng truyền chuyển động giữa 2 trục vuông góc với nhau nằm cùng mặt phẳng. Người ta thường dùng cơ cấu này để truyền chuyển động qua một hướng mới (vuông góc với hướng ban đầu) trong điều kiện không gian tối thiểu (chiếm ít chỗ). Các múi răng của bánh răng côn được khắc trên một mặt hình nón chứ không nằm trên bề mặt hình trụ tròn như các bánh răng thông thường. Bộ Lego Mindstorms 9797 có các bánh răng côn 12t, 20t.

II.3 Bánh răng côn kép :

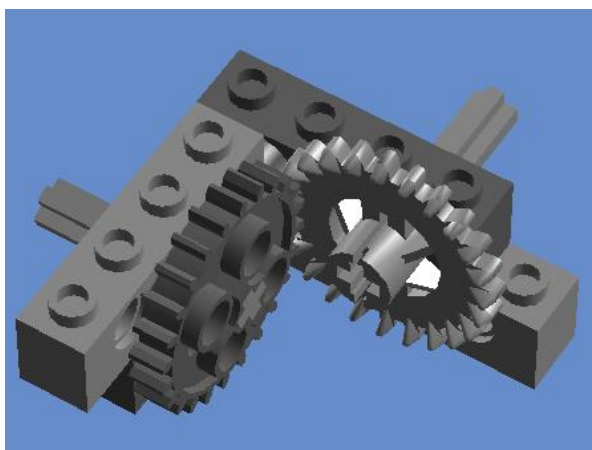
Để tăng tính đa năng (dùng được cho nhiều mục đích, nhiều hoàn cảnh khác nhau) của các bánh răng, Lego kết hợp cả 2 loại múi răng (răng thường và răng côn) trên cùng một bánh răng, và đồng thời tạo luôn răng côn trên cả 2 phía của múi răng để ta có thể dùng răng côn ở phía nào cũng được. Loại bánh răng này được gọi là **bánh răng côn kép**. Như vậy, các bánh răng côn kép có khả năng truyền chuyển động giữa 2 trục song song hoặc 2 trục vuông góc. Bộ Lego Midnstorms 9797 có các bánh răng côn kép sau: 12t, 20t, và 36t.



Hình 4.10 Bánh răng côn kép

IV.3 Bánh răng côn dẹt :

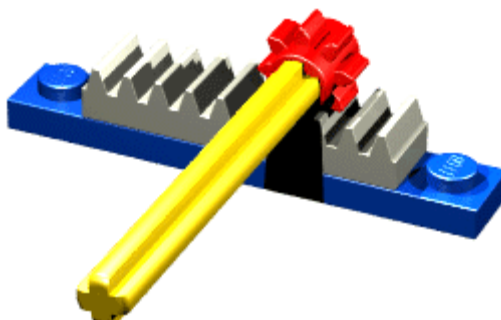
Đây là một loại bánh răng côn đặc biệt có các múi răng vuông góc với bề mặt của bánh răng. Ta có thể dùng bánh răng côn dẹt kết hợp với bánh răng thường để truyền chuyển động giữa các trục vuông góc với nhau.



Hình 4.11 Bánh răng côn dẹt.

IV.4 Bánh răng thường và đai răng :

Cơ cấu này gồm có bánh răng thường (bánh răng trụ thẳng) ăn khớp với thanh răng thẳng, chúng được dùng để biến chuyển động quay thành chuyển động thẳng và ngược lại. Cơ cấu này có thể dùng để kéo dài hay thu gọn cần cầu, chuyển hướng của bánh xe, nâng lên/hạ xuống mâm chứa tải trọng trong xe nâng, v.v...



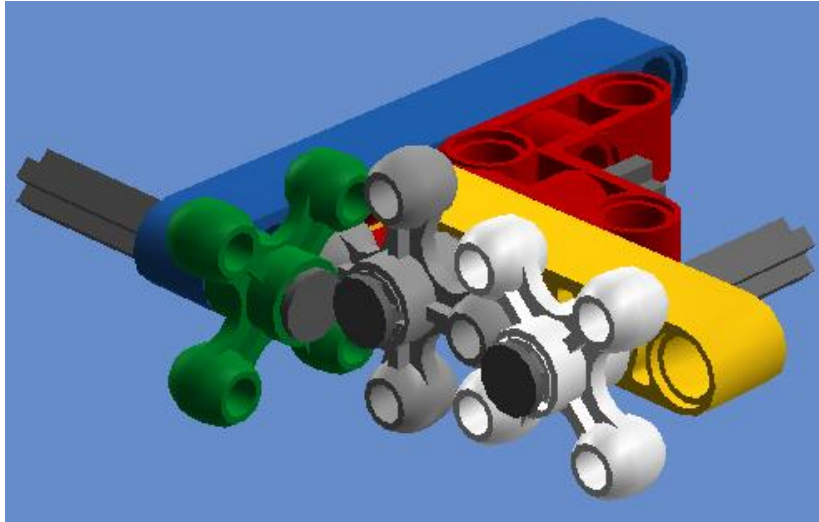
Hình 4.12 Cơ cấu bánh răng – thanh răng

IV.5 Bánh răng hình tay hình tay nắm :



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

Thật ra thì vật này trông cũng không giống như một bánh răng cho lắm (hình 4.9), tuy nhiên nó có tác dụng tương tự như bánh răng côn, tức là truyền chuyển động quay giữa các trục song song hoặc vuông góc với nhau. Bánh răng hình tay nắm chỉ ăn khớp với các bánh răng hình tay nắm khác mà thôi, không thể khớp với các loại bánh răng khác.



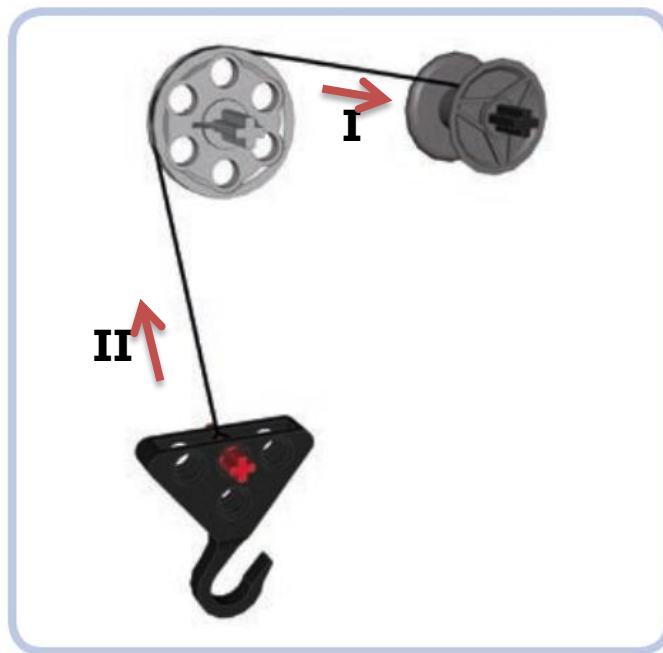
Hình 4.13 Cơ cấu bánh răng hình tay nắm.

CHƯƠNG 5 : CÁC BỘ TRUYỀN ĐỘNG KHÁC

I. Ròng rọc (Pulley) :

I.1 Ròng rọc đơn:

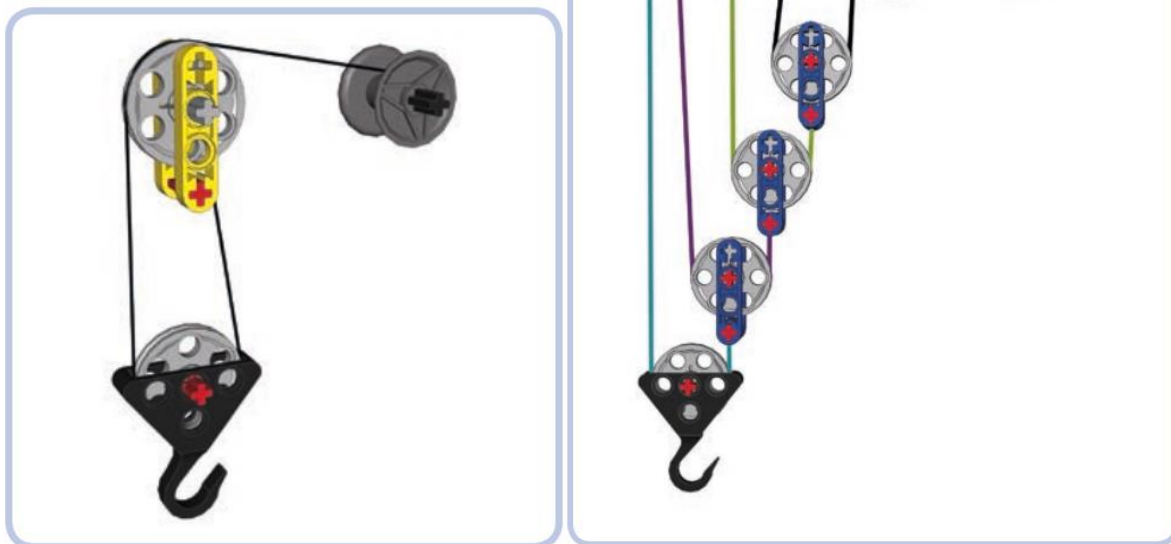
Là một dụng cụ bao gồm một bánh xe có thể có rãnh ở vành ngoài, và một sợi dây kéo (cable) treo vào bánh xe theo rãnh này. Khi ta kéo một đầu dây thì phần dây bên kia cũng phát sinh một lực kéo tương ứng. Ròng rọc đơn giản thường được dùng để đổi hướng của lực. Trong ví dụ ở hình 5.1, lực kéo ban đầu (I) theo hướng đi xuống làm sinh ra lực kéo (II) theo hướng đi lên để nâng một vật nặng.



Hình 5.1 Ròng rọc đơn giản dùng để chuyển hướng của lực.

I.2 Ròng rọc kép và hệ ròng rọc:

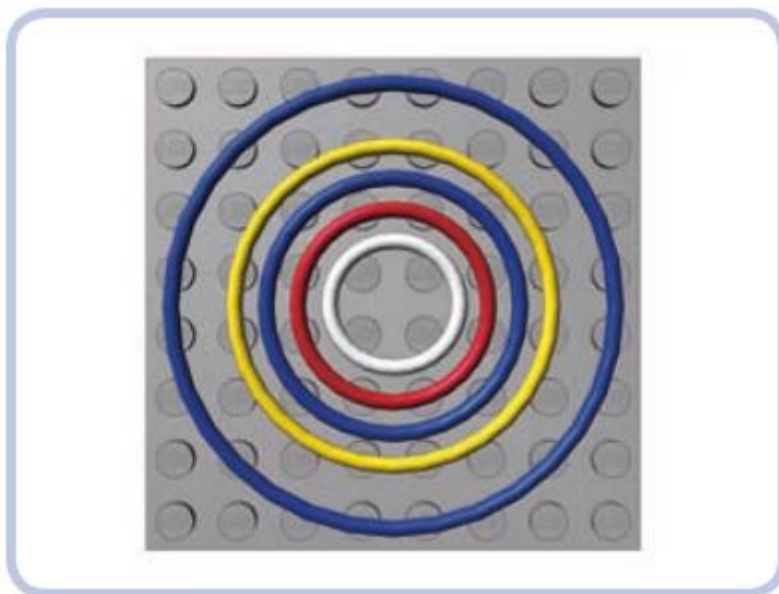
Người ta có thể kết hợp 2 hay nhiều ròng rọc đơn lại với nhau (theo nhiều cách) thành một hệ ròng rọc. Người ta thường dùng hệ ròng rọc để nâng các vật nặng mà chỉ cần dùng một lực nhỏ hơn so với trọng lượng của vật. Tuy nhiên, theo nguyên tắc bảo toàn năng lượng thì đối với hệ ròng rọc, khi ta được **lợi về lực (giảm lực)** thì sẽ bị **thiệt hại về đường đi (tăng quãng đường)** và ngược lại. Tức là nếu dùng một lực nhỏ thì sẽ phải kéo ròng rọc một quãng đường dài hơn. Xin xem thêm sách giáo khoa vật lý phổ thông trung học cơ sở để tìm hiểu rõ hơn về cách tính toán lực và quãng đường cần phải thực hiện.



Hình 5.2 Hệ ròng rọc kép và hệ nhiều ròng rọc.

I.3 Bánh xe và dây đai :

Để truyền chuyển động giữa hai trục cách xa nhau, người ta thường sử dụng cơ cấu truyền động đai.

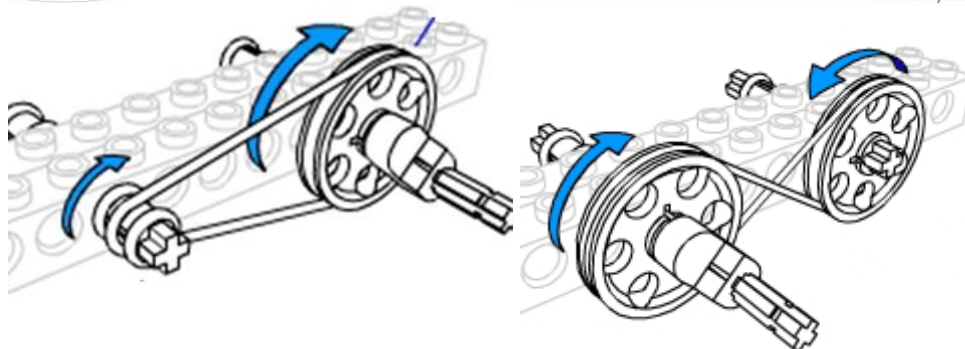


Hình 5.3 Các loại dây cao su LEGO với kích cỡ khác nhau

Cơ cấu này bao gồm hai bánh xe (bánh đai) có thể có rãnh ở vành ngoài và một sợi dây đai móc vào hai bánh xe theo các đường rãnh ở mỗi bánh đai (hình 5.3). Lưu ý về cách móc đai, để cho hai bánh đai quay cùng chiều hay ngược chiều.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



Hình 5.4 Truyền động giữa hai bánh đai.

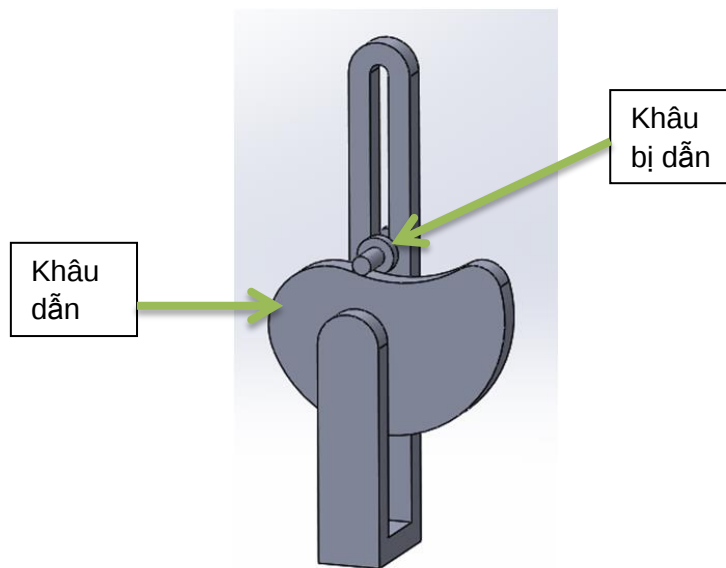
CHƯƠNG 6 : CÁC CƠ CẤU CƠ KHÍ

I. Cơ cấu cam :

I.1 Giới thiệu :

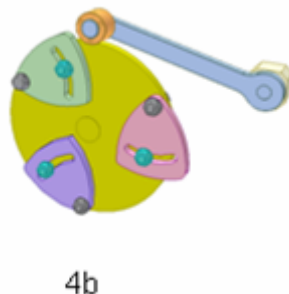
Cơ cấu cam là cơ cấu thực hiện chuyển động qua lại của khâu bị dẫn nhờ vào đặc tính hình học (quy luật cho trước) của khâu dẫn.

I.2 Hoạt động



Hình 6.1 Cơ cấu cam khớp trượt.

Cam là cơ cấu dẫn, cần là cơ cấu bị dẫn. Khi cam và cần ở trong cùng một mặt phẳng hoặc ở trên các mặt phẳng song song, ta có cơ cấu cam phẳng. Thông thường cam(khâu dẫn) nối với giá bằng khớp quay, khi cần(khâu bị dẫn) nối với giá bằng khớp trượt, khi đó cần sẽ tịnh tiến lên xuống. Khi cần nối với giá bằng khớp quay ta có cơ cấu cam cần lắc, cần khi đó chuyển động lắc qua lại.



Hình 6.2 Cơ cấu cam cần lắc.

II. Cơ cấu vi sai :

II.1 Giới thiệu :

Bộ vi sai là cơ cấu dùng để đưa nguồn động lực của động cơ xuống các bánh xe. Bộ vi sai thường được lắp đặt với cơ cấu truyền lực cuối, hay còn gọi là cầu xe.



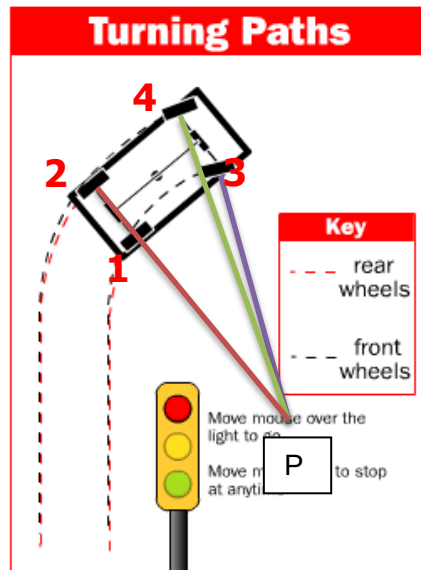
Hình 6.3 Cơ cấu vi sai

II.2 Nhiệm vụ :

- Truyền mô men của động cơ xuống các bánh xe.
- Đóng vai trò là cơ cấu giảm tốc cuối cùng trước khi mô men xoắn truyền tới các bánh xe.
- Truyền mô men tới bánh xe khi cho phép chúng quay với tốc độ khác nhau.

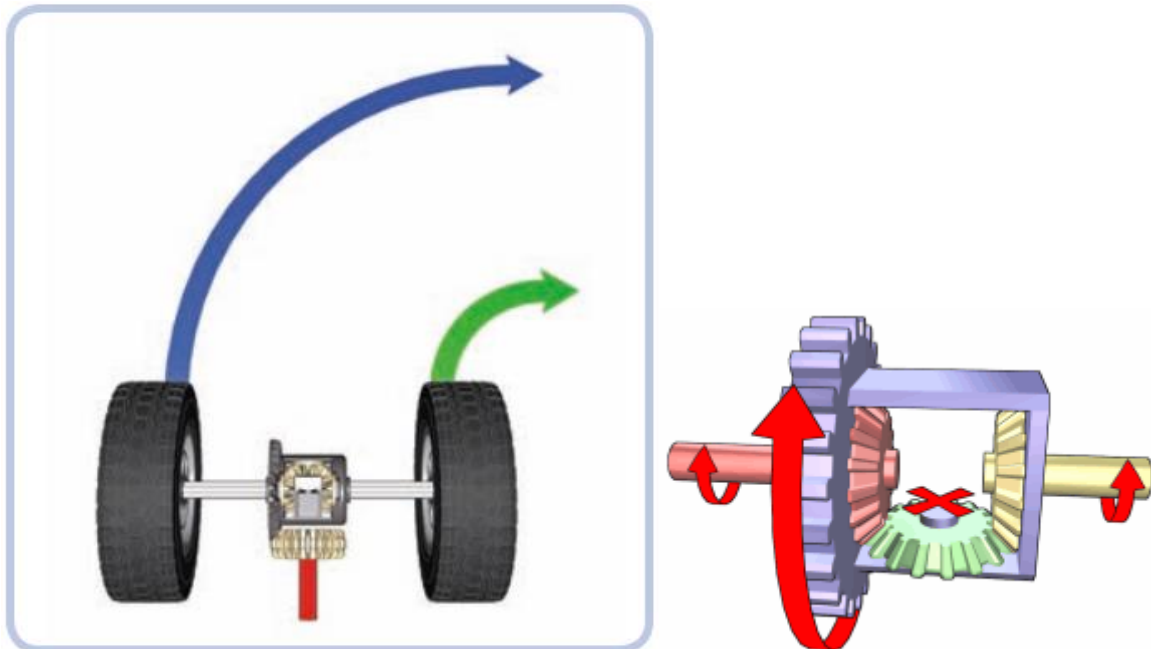
Bộ vi sai là thiết bị dùng để chia mô men xoắn của động cơ thành hai đường, cho phép hai bên bánh xe quay với tốc độ khác nhau. Chúng ta có thể tìm thấy bộ vi sai ở tất cả các xe hơi và xe tải hiện đại, và đặc biệt ở các loại xe 4 bánh chủ động hoàn toàn. Bộ vi sai rất hữu ích và cần thiết khi xe vào cua, vì hai bánh xe phải quay với tốc độ khác nhau để đảm bảo cho xe không bị trượt.

Xét sơ đồ sau đây :



Hình 6.4 Sơ đồ mô phỏng góc bánh xe khi vào cua.

Thực tế thì các bánh xe quay với các tốc độ khác nhau, đặc biệt là khi quay vòng. Hãy nhìn sơ đồ mô phỏng. Mỗi bánh xe sẽ đi được những quãng đường khác nhau khi chiếc xe vào cua, các bánh xe phía trong đi được quãng đường ngắn hơn các bánh xe phía ngoài. Điều đó có nghĩa là các bánh xe bên trong sẽ quay với tốc độ thấp hơn các bánh xe ngoài.



Hình 6.5 Cách lắp đặt bộ vi sai và sơ đồ di chuyển hai bánh chủ động khi vào cua.

Đối với các bánh xe bị động, ví dụ như các bánh trước của chiếc xe dẫn động bánh sau, không có gì liên kết chuyển động giữa chúng nên *hoạt động độc lập với nhau*. Thế nhưng, hai bánh sau lại có sự liên kết để cùng nhận được nguồn động lực từ một động cơ và một hộp số duy nhất. Nếu không có bộ vi sai, hai bánh sau sẽ bị khoá lại với nhau, bị bắt buộc phải quay với cùng một tốc độ như nhau. Điều này sẽ làm cho việc quay vòng của xe rất khó khăn. Để chiếc xe vào cua được, chắc chắn một bánh xe sẽ bị trượt. Lực này sẽ được truyền từ bánh xe bên này sang bên kia qua trục bánh xe, làm tăng lực xoắn tác dụng lên trục bánh xe.

Khi xe vào cua(chuyển động cong), các điểm trên thân xe sẽ di chuyển quay xung quanh một điểm, gọi là tâm vận tốc tức thời(tâm quay trong thời điểm đang xét-điểm P). Do đó vận tốc chuyển động của các bánh xe phải khác nhau, xét bánh số 1 và số 2(hình 6.4), lấy P làm tâm, dựa vào quy tắc hình học, $V_1 < V_2$ (V =vận tốc góc x bán kính quay, điểm càng xa tâm vận tốc di chuyển càng lớn). Tóm lại, sự xuất hiện của bộ vi sai sẽ điều chỉnh vận tốc của các bánh xe, lúc này bánh số 1 sẽ giảm tốc so với bánh số 2, làm cho xe vào cua một cách an toàn và trơn tru.

Thông thường bộ vi sai được lắp đặt để truyền lực từ động cơ đến hai bánh xe chủ động, do đó tùy vào mục đích thiết kế (hai bánh xe chủ động nằm phía trước hoặc phía sau) mà có vị trí đặt bộ vi sai cho phù hợp. Cách dễ nhất để bạn hiểu được nguyên lý hoạt động của bộ vi sai là lắp ráp và kiểm chứng trong thực tế.

III. Cơ cấu “vòng lái”



Hình 6.6 Cơ cấu “vòng lái”

Cơ cấu này thường được sử dụng trong hộp số để chuyển đổi giữa các cấp tốc độ với nhau. Xem hình 6.6, đặt tên các trục là trục dưới, trục trên trái và trục trên phải, “vòng lái” là chi tiết màu đỏ, được gắn chặt vào trục dưới, còn hai bánh răng thì không được gắn chặt với trục dưới. Nguyên lý làm việc như sau : cho rằng trục dưới luôn quay, ở trạng thái ban đầu, khi “vòng lái” nằm ở giữa, quay cùng vận tốc góc với trục dưới, thì hai bánh răng hai bên đứng im, do đó hai trục trên cũng đứng im. Khi cái lấy đưa “vòng lái” về phía bên trái, các gân ở trên “vòng lái” ăn khớp với bánh răng và khiến cho bánh răng quay kéo theo trục trên trái cũng quay. Tiếp theo cái lấy đưa “vòng lái” về phía bên phải, thì cũng tương tự khiến cho trục trên phải quay theo. Tốc độ quay của hai trục trên so với trục dưới phụ thuộc vào tỉ lệ của hai cặp bánh răng ăn khớp.

Ở trên các loại xe hơi, chúng ta thường hay thấy cần số, trong ví dụ này nó chính là cái lấy. Cơ cấu này giúp cho xe có tốc độ phù hợp trên từng địa hình di chuyển.

IV. Cơ cấu lái :

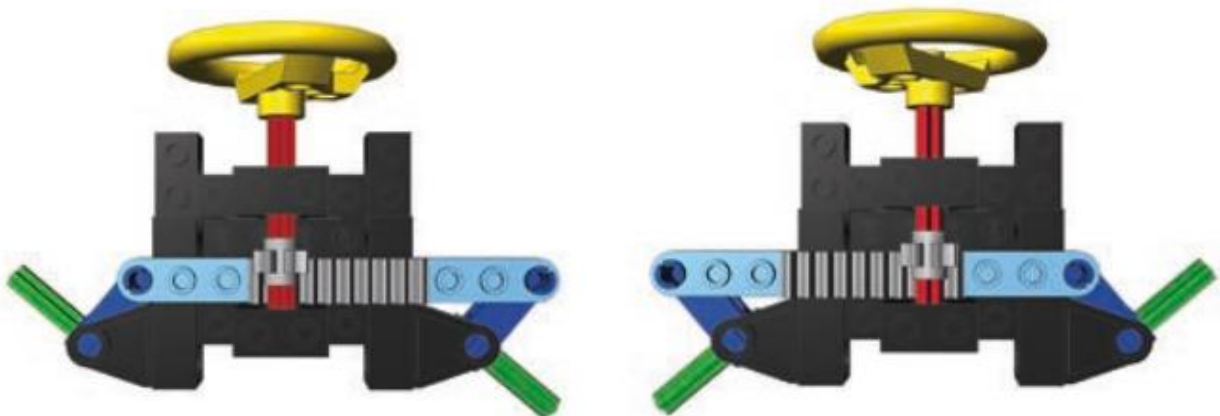
Cơ cấu lái có nhiệm vụ bảo đảm sự chuyển động của ô tô theo định hướng của người lái. Việc thay đổi hướng chuyển động của ô tô thực hiện bằng cách quay vô lăng, tác động tới hướng của bánh trước thông qua cơ cấu lái. Và không chỉ đơn thuần là định hướng, mà cơ cấu lái còn ảnh hưởng tới sự an toàn của chiếc xe. Xem xét cơ cấu lái cơ bản sau :



Hình 6.7 Cơ cấu lái



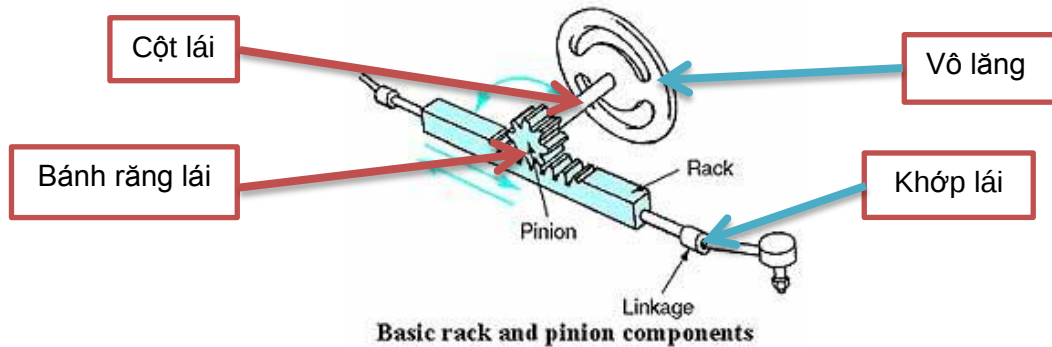
Hình 6.8 Cơ cấu lái ở vị trí quẹo trái, ở giữa và quẹo phải.



Hình 6.9 Mô tả cách vận hành của cơ cấu khi bẻ cua.

Các thành phần của cơ cấu lái :

- Vô lăng(bánh lái) : đây là thành phần quen thuộc với người lái xe, khi vận hành, người lái trực tiếp điều khiển vô lăng xoay theo hướng mình muốn di chuyển.



Hình 6.10 Các thành phần cơ cấu lái.

- Cột lái : là thành phần kết nối vô lăng với bánh lái.
- Bánh răng lái : biến đổi mô men xoắn từ vô lăng truyền tới bánh xe thông qua các khớp lái, làm xe đổi hướng.
- Khớp lái : truyền lực từ bộ phận bánh răng đến hai bánh.

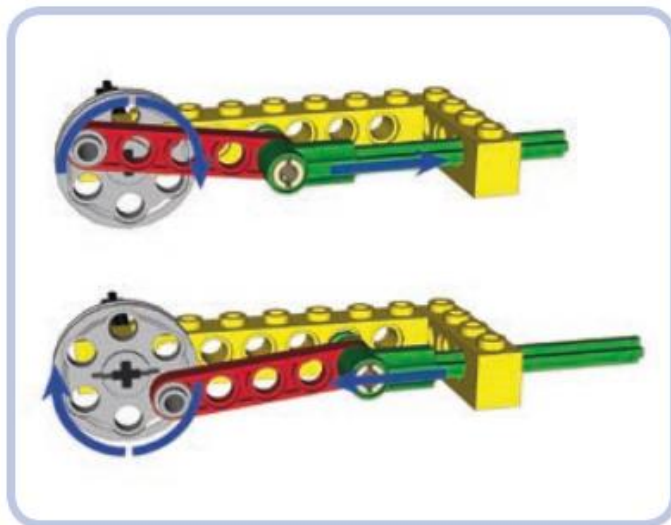
Nguyên lý chung : khi xoay vô lăng, mô men sẽ truyền qua hệ thống bánh răng – thanh răng đến các khớp lái. Tùy theo hướng xoay mà bánh xe sẽ được bẻ sang hướng phù hợp để điều chỉnh hướng di chuyển.



Hình 6.11 Xe LEGO cơ bản với hai trục : một trục cố định và một trục lái.

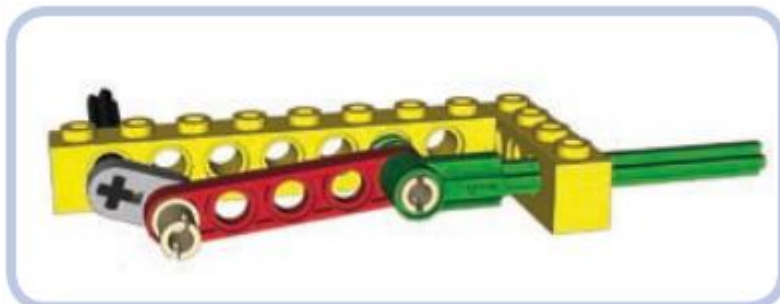
V. Cơ cấu tay quay con trượt :

Cơ cấu tay quay – con trượt dùng để biến đổi chuyển động quay của tay quay thành chuyển động tịnh tiến của con trượt. Cơ cấu gồm một đĩa và một thanh ngăn để kết nối đĩa với cần đẩy, khi đĩa xoay, thanh làm cho cần đẩy di chuyển tới – lui. Lưu ý rằng cần đẩy được di chuyển trong một khung dẫn, trong hình dưới đây đó là khung màu vàng.



Hình 6.12 Cơ cấu tay quay – con trượt cơ bản.

Khoảng cách di chuyển của cần đẩy(con trượt) phụ thuộc vào đường kính của đĩa. Đường kính càng lớn, khoảng di chuyển của cần đẩy sẽ càng rộng. Chúng ta có thể thay chuyển động quay của đĩa bằng một thanh khác, như vậy, khoảng di chuyển của cần đẩy sẽ phụ thuộc vào độ dài của thanh này.



Hình 6.13 Thay bánh dẫn động bằng thanh.

Cơ cấu tay quay – con trượt cũng có thể biến đổi chuyển động quay thành chuyển động lắc lư(có nghĩa là chuyển động quay một phần). Loại cơ cấu này không có cần đẩy, mà thay vào đó là một đĩa thứ hai. Khoảng di chuyển của đĩa thứ hai này phụ thuộc và sự tương quan về chu vi của hai đĩa, chúng ta có thể điều chỉnh mức độ chuyển động bằng cách sử dụng những đĩa có đường kính khác nhau. Tuy nhiên, để cơ cấu làm việc được, đường kính của đĩa thứ hai phải lớn hơn đường kính của đĩa đầu tiên, và chiều dài của thanh nối cũng phải lớn hơn đường kính của đĩa đầu tiên.



Hình 6.14 Cơ cấu với hai đĩa nối với nhau bằng một thanh. Đĩa nhỏ quay hết hành trình(vòng), và đĩa lớn quay một phần qua lại.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

Ngoài ra bạn cũng có thể thay thế đĩa quay bằng thanh quay. Bởi vì chi tiết bị dẫn không quay đầy đủ một vòng, nên việc thay thế thanh quay sẽ làm giảm không gian hơn so với sử dụng đĩa. Lưu ý cơ cấu này chỉ hoạt động theo một hướng, bạn không thể dùng thanh màu xanh lá cây lái ngược lại thanh màu xám.

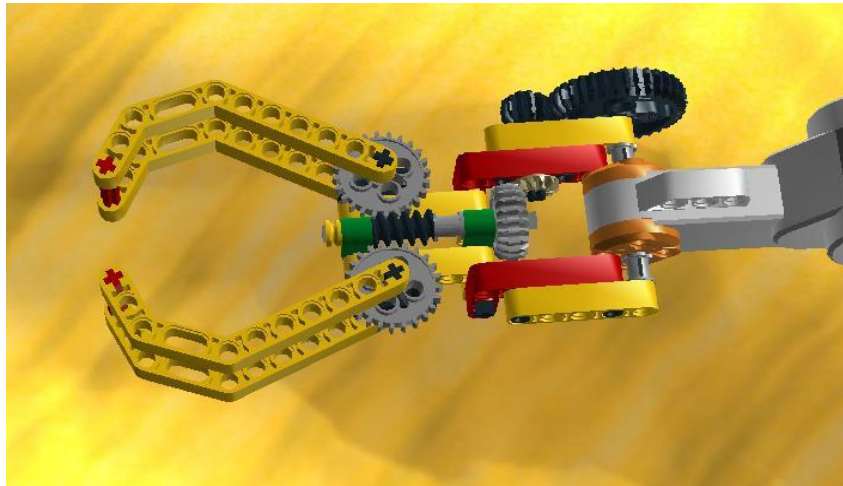


Hình 6.15 Cơ cấu sử dụng thanh

PHẦN 4 : CÁC CƠ CẤU LÀM VIỆC THÔNG THƯỜNG.

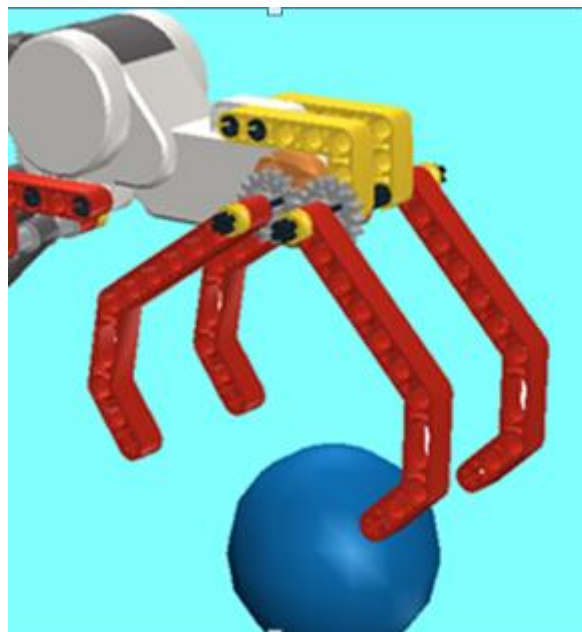
CHƯƠNG 7 : CÁC CƠ CẤU TAY GẤP – TAY NÂNG

I. Tay gấp cơ bản:



Hình 7.1 Tay gấp trực vít – bánh vít.

Nguyên lý làm việc : cơ cấu tay gấp hình 7.1 sử dụng bộ truyền trục vít – bánh vít để làm việc. Động cơ truyền chuyển động qua trục vít bằng hệ thống bánh răng và trục. Khi trục vít quay sẽ khiến cho bánh răng hai bên quay ngược chiều nhau thực hiện động tác gấp – thả. Như đã nêu ở trên, bộ truyền trục vít – bánh vít chỉ tác động một chiều, lực tác động lớn, do đó khi hoạt động, chỉ có động cơ mới khiến tay máy kẹp nhà được, mô hình được ứng dụng trong các cơ cấu cần lực gấp mạnh và chắc chắn.

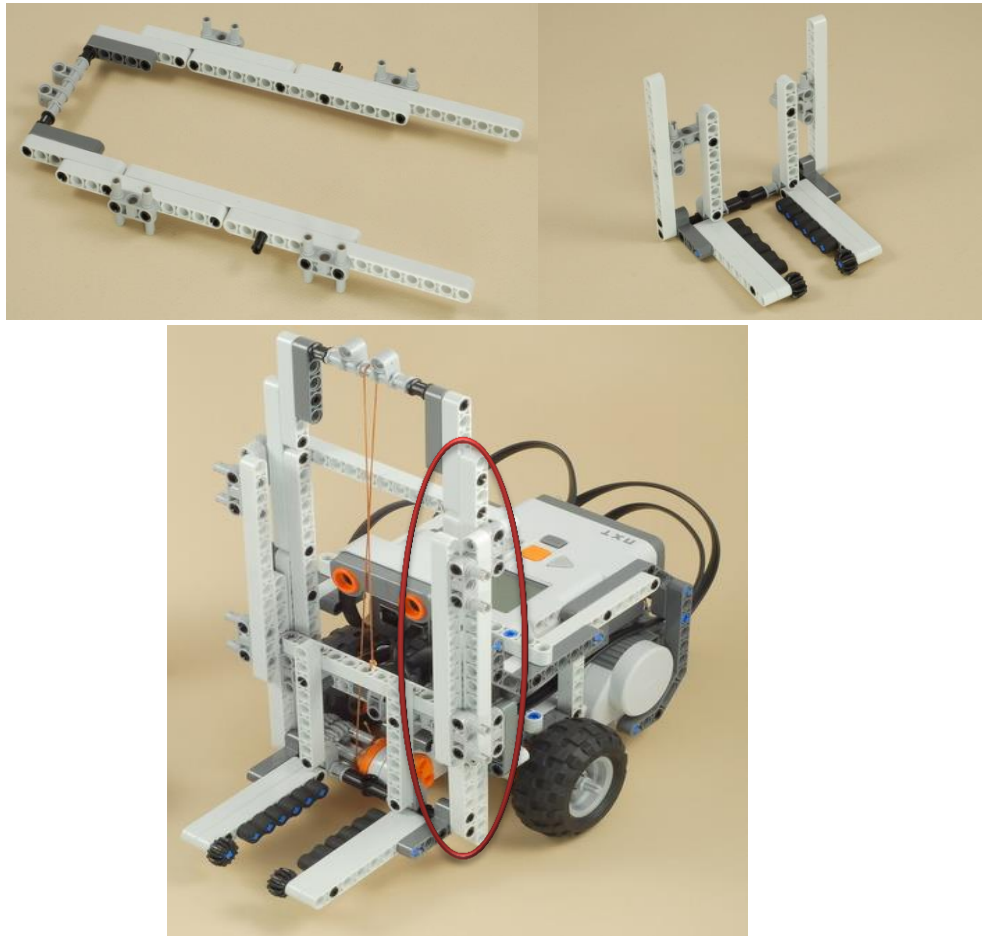


Hình 7.2 Tay gấp bánh răng – bánh răng.

Nguyên lý làm việc : cơ cấu tay gấp hình 7.2 sử dụng quy luật về sự ăn khớp của bánh răng : hai bánh răng ăn khớp với nhau thì quay ngược chiều nhau. Qua đó điều khiển cánh tay đóng – mở qua sự hoạt

động thuận – ngược của động cơ. Trong một vài trường hợp, để tăng lực gấp ta có thể sử dụng bánh răng nhỏ, hoặc cần tăng tốc độ đóng – mở ta sử dụng bánh răng lớn.

II. Cơ cấu nâng :



Hình 7.3 Xe nâng

Bộ nâng được gắn vào xe qua hai ray trượt ở hai bên. Dây cáp được nối từ ru lô qua thanh đỡ đến bộ nâng. Động cơ quay sẽ làm ru lô quay để quấn hoặc thả dây cáp, làm cho bộ nâng di chuyển lên xuống. Tham khảo mô hình xe nâng “Forklift”.



Hình 7.4 Cơ cấu nâng sử dụng cơ cấu thanh răng – bánh răng.

Cách thứ hai để nâng hạ là sử dụng cơ cấu thanh răng – bánh răng. Chuyển từ chuyển động quay của động cơ qua chuyển động thẳng của cơ cấu nâng hạ. Hành trình di chuyển của cơ cấu chính là độ dài phần ăn khớp của thanh răng.

CHƯƠNG 8 : CÁC CƠ CẤU DI CHUYỂN

I. Di chuyển bằng bánh xe :

I.1 Di chuyển 3 bánh :



Hình 8.1 Mô hình 3 bánh cơ bản

Tham khảo mô hình “Castor Bot”, mô hình sử dụng hai động cơ dẫn động với mỗi bánh nối trực tiếp với mỗi động cơ qua trục chữ thập. Bánh xe còn lại có tác dụng như là “bệ đỡ” giúp giữ cho Robot được cân bằng. Thiết lập tốc độ hai bánh *bằng nhau* để điều khiển Robot tiến hoặc lùi. Để xoay Robot hoặc đi đường vòng, thiết lập tốc độ hai bánh *khác nhau*, một bánh đứng yên, một bánh di chuyển hoặc một bánh tiến, một bánh lùi tùy theo mục đích di chuyển, xem thêm phần “Move” trong mục lập trình.

I.2 Di chuyển 4 bánh :



Hình 8.2 Khung mô hình xe đua. Động cơ truyền động cho trục bánh xe qua bộ bánh răng.

Hai bánh trước được thêm vào trên “cơ cấu lái”, sự kết hợp giữa việc thay đổi *tốc độ* của hai bánh sau và *góc quay* của động cơ trước điều khiển cơ cấu lái sẽ quyết định hướng di chuyển của xe. Do được tiếp xúc với nền bằng 4 bánh nên khả năng cân bằng tốt hơn so với Robot di chuyển bằng 3 bánh.

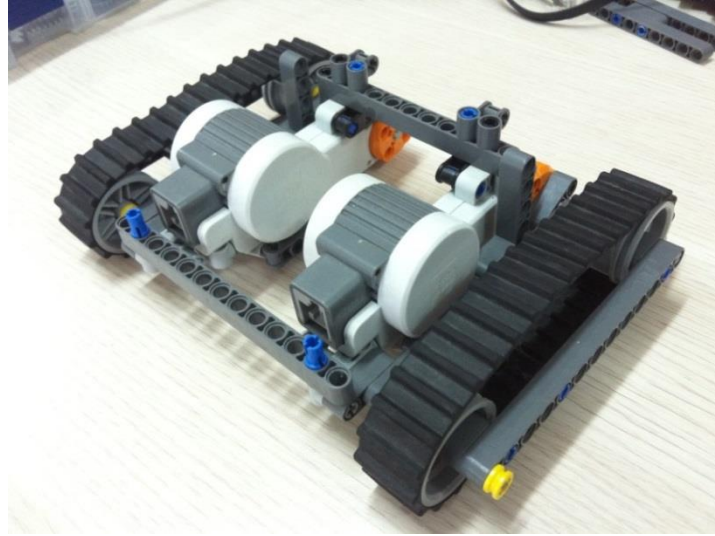
II. Di chuyển bằng xích :

Xích là một chuỗi các mắt xích được nối với nhau bằng bản lề. Xích truyền chuyển động và tải trọng từ trục dẫn sang trục bị dẫn nhờ sự ăn khớp của các mắt xích với các răng xích trên đĩa xích (ăn khớp gián tiếp). Các trục của bộ truyền xích thông thường là song song với nhau, có thể trong bộ truyền có nhiều bánh xích bị dẫn.



Hình 8.3 Dây xích và các loại bánh xích.

Trong bộ LEGO 9797, dây xích được cấu tạo từ vật liệu nhựa mềm dẻo, do đó nó có dạng đặc, mặt trong của dây xích có “khắc” để đĩa xích (trong bộ 9797 đĩa xích đồng thời cũng là mâm bánh xe) tác động lực và kéo dây xích di chuyển.



Hình 8.4 Mô hình đơn giản của cơ cấu sử dụng bánh xích.

Cũng tương tự như di chuyển bằng bánh xe, đĩa xích chủ động được nối trực tiếp với động cơ qua trục, lập trình tốc độ động cơ khác biệt hai bên để điều hướng di chuyển của Robot.

Ứng dụng :

Ở thực tế, ta thường thấy xe tăng và một số loại xe cơ giới chuyên dùng có bộ phận di chuyển là bánh xích. Thiết bị bánh xích của xe tăng cũng giống như đôi bàn chân người nâng đỡ toàn thân khi bước đi vậy. "Bàn chân" ấy của xe tăng là bộ bánh xích với vòng xích, bánh chủ động, bánh chịu tải, bánh chuyển hướng và bánh đỡ xích. Tác dụng của hệ thống bánh xích này là làm bộ đỡ xe, đẩy xe chạy, làm cho xe tăng có tính cơ động cao. Vòng bánh xích là cấu kiện chủ yếu, là bộ phận tiếp xúc trực tiếp của xe tăng với mặt đất. Nó vừa rộng vừa to, bám sát mặt đất, đảm bảo cho xe tăng vận hành. Vòng bánh xích vừa rộng vừa to làm cho áp lực bình quân của xe tăng lên mặt đất giảm, khiến xe có thể đi được trên mặt đất lầy lội. Mỗi mắt xích tác dụng lên mặt đất đều được chế tạo có vết rãnh, giống như vết hoa văn ở đế giày để tạo lực ma sát, nâng cao tính năng vận hành của xe. Bánh chủ động thông qua sự ăn khớp giữa bánh răng và vòng xích để truyền động lực từ động cơ sang vòng bánh xích làm cho xe vận động. Những cái bánh xe to nhỏ khác nhau đều có tác dụng riêng vô cùng quan trọng đối với việc nâng cao tính năng cơ động của xe.



Hình 8.5 Robot sử dụng bánh xích



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT: (08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

CHƯƠNG 9 : CÁC LOẠI CẢM BIẾN.

I. Giới thiệu

Cảm biến là thiết bị có khả năng phát hiện và đo lường đại lượng điện và không điện, chuyển đổi chúng trở thành những tín hiệu điện phù hợp với thiết bị thu nhận tín hiệu.

Các đại lượng đo (M) (như nhiệt độ, áp suất, trọng lượng...) tác động lên cảm biến cho ta đại lượng đặc trưng (S) mang tính chất điện (như điện tích, điện áp, dòng điện hay trở kháng) chứa đựng thông tin cho phép xác định giá trị của đại lượng đó.

Đặc trưng (S) là hàm của đại lượng cần đo (M)

$$S = F(M)$$

Người ta gọi (S) là đại lượng đầu ra hoặc phản ứng của cảm biến. (M) là đại lượng đầu vào hay kích thích (có nguồn gốc đại lượng cần đo). Thông qua đo đặc (S) cho phép nhận biết giá trị (M).

II. Cảm biến chạm :

Thành phần cơ bản của cảm biến chạm chính là một “nút nhấn” có lò xo, ban đầu nút nhấn ở trạng thái tự do, khi bị tác động, nút được ấn vào và gây ra sự biến đổi tín hiệu, khi thả ra, lò xo đẩy nút nhấn trở về trạng thái ban đầu.



Hình 9.1 Cảm biến chạm

Nguyên tắc hoạt động : khi cảm biến chạm được nhấn vào, mạch điện được đóng lại, và cho phép dòng điện chạy qua. Nếu cảm biến chạm được thả ra, mạch điện hở và dòng điện bị chặn lại.



Hình 9.2 Sơ đồ điện cảm biến chạm

Như vậy, cảm biến chạm sẽ có hai trạng thái, không bị tác động(0) và khi được nhấn(1), do đó có thể mô hình hoá thành ba loại sau :



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

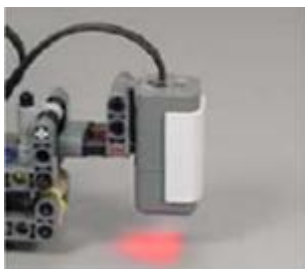
- Nhấn(Pressed).
- Thả(Released).
- Nhấn – Thả(Bumped).

Qua đó ta thấy cảm biến sẽ thu tín hiệu bằng ba cách. Sử dụng “Bumped” nếu bạn muốn cảm biến được kích hoạt bằng động tác nhấn và thả trong khoảng thời gian 0.5s. Chọn “Pressed”, sự biến đổi ngay lập tức được ghi lại khi bạn nhấn vào cảm biến, và “Released” ở trường hợp bạn muốn lấy tín hiệu khi cảm biến ngay khi cảm biến ở trạng thái nhả.

III. Cảm biến ánh sáng

Theo tính chất vật lí, ta thấy được vật do có ánh sáng trắng chiếu đến vật và phản chiếu đến mắt ta. Ánh sáng trắng là một dải quang phổ liên tục trải dài từ tia hồng ngoại(có bước sóng dài nhất) đến tia tử cực tím(có bước sóng ngắn nhất). Ta thấy các vật có màu sắc là do *vật không hấp thụ được màu đó* nên sẽ phản xạ đến mắt người và ta nhìn thấy màu sắc không được hấp thụ đó. Ví dụ, ta thấy lá cây thường có màu xanh đó là do khi hấp thụ ánh sáng, tia sáng màu xanh lục rất ít được hấp thụ(lá cây hấp thụ chủ yếu là tia sáng màu đỏ, lam và tím) và phản xạ vào mắt người nên ta thấy lá cây thường có màu xanh.

Cảm biến có tác dụng đo độ sáng của bề mặt mà nó tác động. Đầu phát của cảm biến phát ra một nguồn sáng về phía trước. Nếu có vật thể che chắn, nguồn sáng này tác động lên vật thể và phản xạ ngược lại đầu thu, đầu thu nhận tín hiệu ánh sáng này và chuyển thành tín hiệu điện. Nếu khoảng cách từ vật thể đến đầu thu quá xa, đầu thu sẽ nhận ánh sáng từ môi trường và gây ra sự sai lệch trong phép đo. Tóm lại, đầu thu sẽ ghi nhận bất cứ ánh sáng nào mà nó được tác động. Do đó, khi đo trong thực tế, bạn cần phải để cảm biến ở khoảng cách đủ gần với bề mặt cần đo để có được kết quả chính xác.



Những gì chúng ta thấy.

Những gì cảm biến thấy.

Hình 9.3 Cảm biến ánh sáng

Led màu đỏ trên cảm biến ánh sáng NXT phát ánh sáng ra trên bề mặt. Led còn lại sẽ nhận ánh sáng phản chiếu từ bề mặt và chuyển thành tín hiệu điện. Trong dải màu mà đầu thu nhận được, màu tối nhất(0) và màu sáng nhất(100), độ phân giải là 1 đơn vị.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

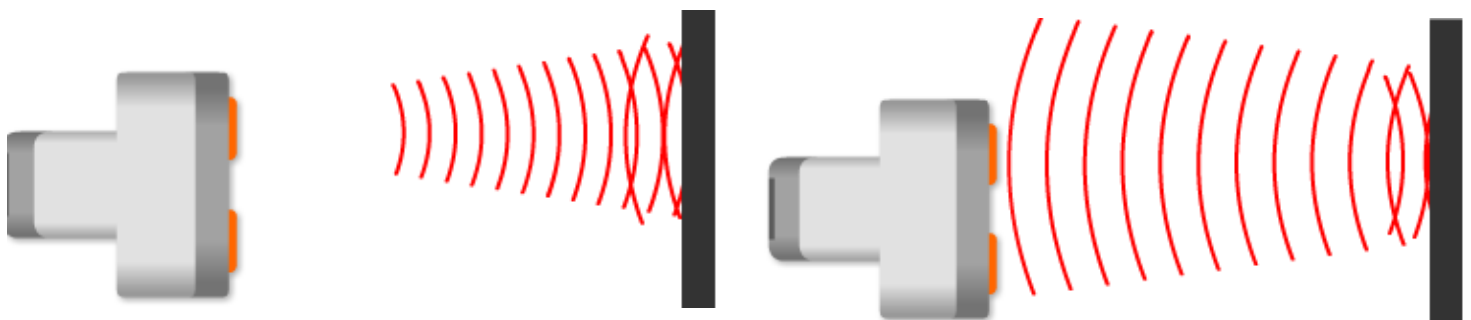
Khái niệm “điểm kích hoạt” : là giá trị cụ thể bằng con số mà tại ngưỡng đó điều kiện thay đổi sẽ xảy ra. Ví dụ : bạn có thể lập[chương] trình cho Robot của bạn di chuyển về phía trước chỉ khi mức độ ánh sáng tăng lên trên 60%. Như vậy, “điểm kích hoạt” sẽ là 60.

IV. Cảm biến siêu âm



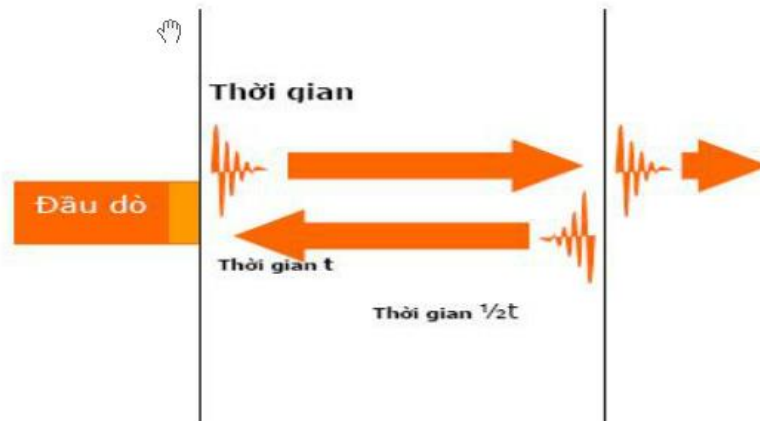
Hình 9.4 Cảm biến siêu âm

Cảm biến sử dụng tốc độ của sóng siêu âm để đo lường khoảng cách tới đối tượng. Sóng siêu âm được truyền đi trong không khí với vận tốc khoảng 343m/s. Nếu một cảm biến phát ra sóng siêu âm và thu về các sóng phản xạ đồng thời, đo được khoảng thời gian từ lúc phát đi tới lúc thu về, thì máy tính có thể xác định được quãng đường mà sóng đã di chuyển trong khoảng không gian.



Quãng đường di chuyển của sóng sẽ bằng 2 lần khoảng cách từ cảm biến tới chướng ngại vật, theo hướng phát của sóng siêu âm. Hay khoảng cách từ cảm biến tới chướng ngại vật sẽ được tính theo nguyên lý TOF :

$$d = v \cdot t / 2$$



Hình 9.5 Sơ đồ sóng

- d : khoảng cách.
- v : vận tốc
- t : thời gian

Cảm biến siêu âm khi truyền dữ liệu về NXT sẽ được hiển thị dưới hai loại đơn vị : cm và inch. Điều kiện để cảm biến hoạt động chính xác là bề mặt vật cản phẳng và đủ lớn để các sóng siêu âm tới và phản xạ về đúng địa chỉ. Nếu như mặt tác động quá gồ ghề hoặc có góc nghiêng, thì sóng sẽ không phản xạ về được, hoặc tín hiệu không đủ để cảm biến xác nhận được khoảng cách.

Cảm biến siêu âm có thể nhận biết được vật nằm trong khoảng cách 100 inches(hay 255cm). Người lập trình sẽ sử dụng “điểm kích hoạt” để điều khiển Robot hoạt động(ví dụ : Robot di chuyển về phía trước cho tới khi phát hiện vật cản cách 30cm, thì dừng lại, lúc này khoảng cách 30cm sẽ là “điểm kích hoạt”).

V. Cảm biến âm thanh :



Hình 9.6 Cảm biến âm thanh.

Cảm biến âm thanh NXT chứa một màng rất nhỏ thu thập các sóng âm thanh và có chức năng như một cái micro. Nó sẽ chuyển đổi những rung động cơ học sang năng lượng điện.



Hình 9.7 Mô phỏng cấu tạo cảm biến âm thanh

Màng được bao quanh bởi một khối nam châm, và ở phần sau nam châm được bao quanh bởi dây kim loại dạng lò xo. Sóng âm là lý do khiến cho màng rung động, kéo theo sự rung động của khối nam châm, tạo nên dòng điện trong vòng dây kim loại và chuyển thành tín hiệu. NXT nhận tín hiệu này và sử dụng nó để đo lường độ lớn của âm thanh.

Dữ liệu cảm biến đọc được	4 – 5%	5 – 10%	10 – 30%	30-100%
So sánh với âm thanh thực tế	Phòng im lặng	Người nói chuyện ở khoảng cách xa	Người nói chuyện ở gần	Hét lên hoặc chơi nhạc nặng

Bảng 4.1 Tín hiệu sóng âm nhận được so với thực tế.

Cũng giống như cảm biến ánh sáng và siêu âm, người lập trình cũng có thể sử dụng “điểm kích hoạt” để điều khiển Robot (ví dụ : Robot di chuyển về phía trước cho tới khi “nghe” âm thanh >50% thì dừng lại. Vậy điểm kích hoạt lúc này là 50).



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

PHẦN 5 : BỘ PHẬN ĐIỀU KHIỂN

CHƯƠNG 10 : SỬ DỤNG CHƯƠNG TRÌNH NXT - G

I. Giới thiệu về môi trường lập trình LEGO MINDSTORMS NXT 2.0 (NXT-G).

Chương này giới thiệu các khái niệm cơ bản về môi trường lập trình cho robot dùng Lego MindStorms NXT 2.0. Người đọc cũng được trình bày một ví dụ về các bước lập trình, nạp chương trình đã viết vào bộ vi điều khiển NXT rồi chạy chương trình này.

I.1 Robot và chương trình điều khiển:

Như ta đã biết, robot là một thiết bị có khả năng hoạt động một cách độc lập và tương tác được với môi trường xung quanh. Cụ thể hơn, bộ phận điều khiển của nó phải có khả năng điều khiển tất cả các bộ phận của robot. Hiện tại, người ta lập (chương) trình cho bộ phận điều khiển để giúp nó có khả năng “nhận biết” được các điều kiện của môi trường xung quanh và điều khiển các bộ phận của robot phản ứng lại các điều kiện này một cách phù hợp để có thể thực hiện được nhiệm vụ của mình.

Một chương trình điều khiển (hay nói vắn tắt là chương trình) là một chuỗi các lệnh mà robot phải làm theo để hoàn thành một mục tiêu nào đó. Nó có thể là chuỗi các thao tác đơn giản để robot di chuyển theo vòng tròn, hoặc phức tạp hơn một chút là chuỗi các thao tác mà robot phải thực hiện để thoát ra khỏi một mê cung, hay là một chuỗi các thao tác đi tìm các khối hình học, di chuyển và xếp chúng vào một vị trí đã định. Phức tạp hơn nữa nó có thể là một chuỗi các thao tác cố gắng tìm và giành bóng, dẫn bóng vượt qua sự truy cản của đối phương, rồi đẩy bóng vào lưới của đối phương trong cuộc thi robot đá bóng, v.v...

Lập trình cho bộ điều khiển chính là việc tạo ra chương trình điều khiển cho robot.

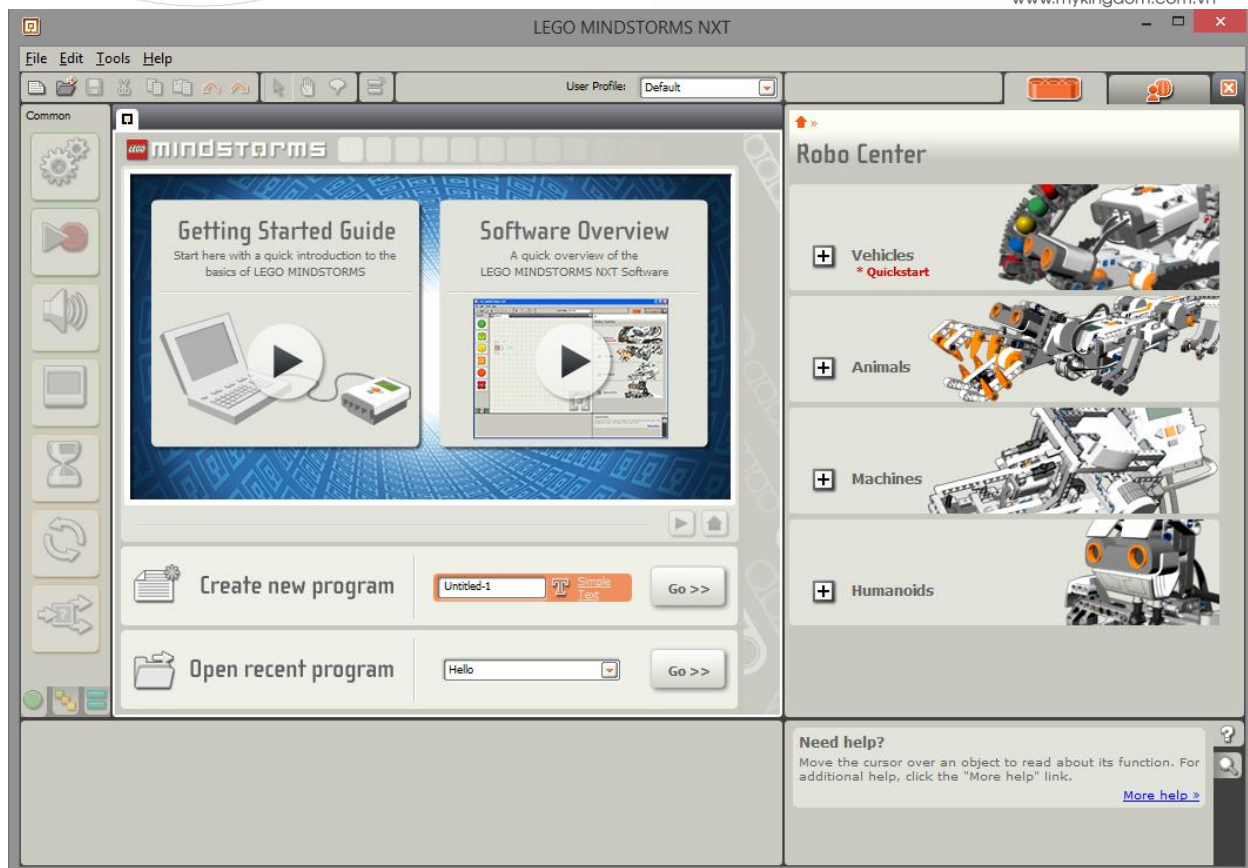
I.2 Giao diện cơ bản của công cụ lập trình LEGO MINDSTORMS NXT 2.0 (NXT-G):

Khi bạn khởi động ứng dụng Lego Mindstorms NXT 2.0, bạn sẽ thấy màn hình (cửa sổ) Getting Started (hình 10.1). Từ cửa sổ này, bạn có thể tạo một chương trình mới hoặc mở một chương trình đã có trên máy tính.

*Lưu ý: Các hình chụp trong tài liệu này được trích từ phiên bản Lego Mindstorms NXT Retail v2.0.f4 chạy trên hệ điều hành Windows 8.1. Nếu bạn dùng một phiên bản NXT khác hay một hệ điều hành khác thì những hình chụp này có thể sẽ hơi khác so với những màn hình tương ứng trên máy tính của bạn.

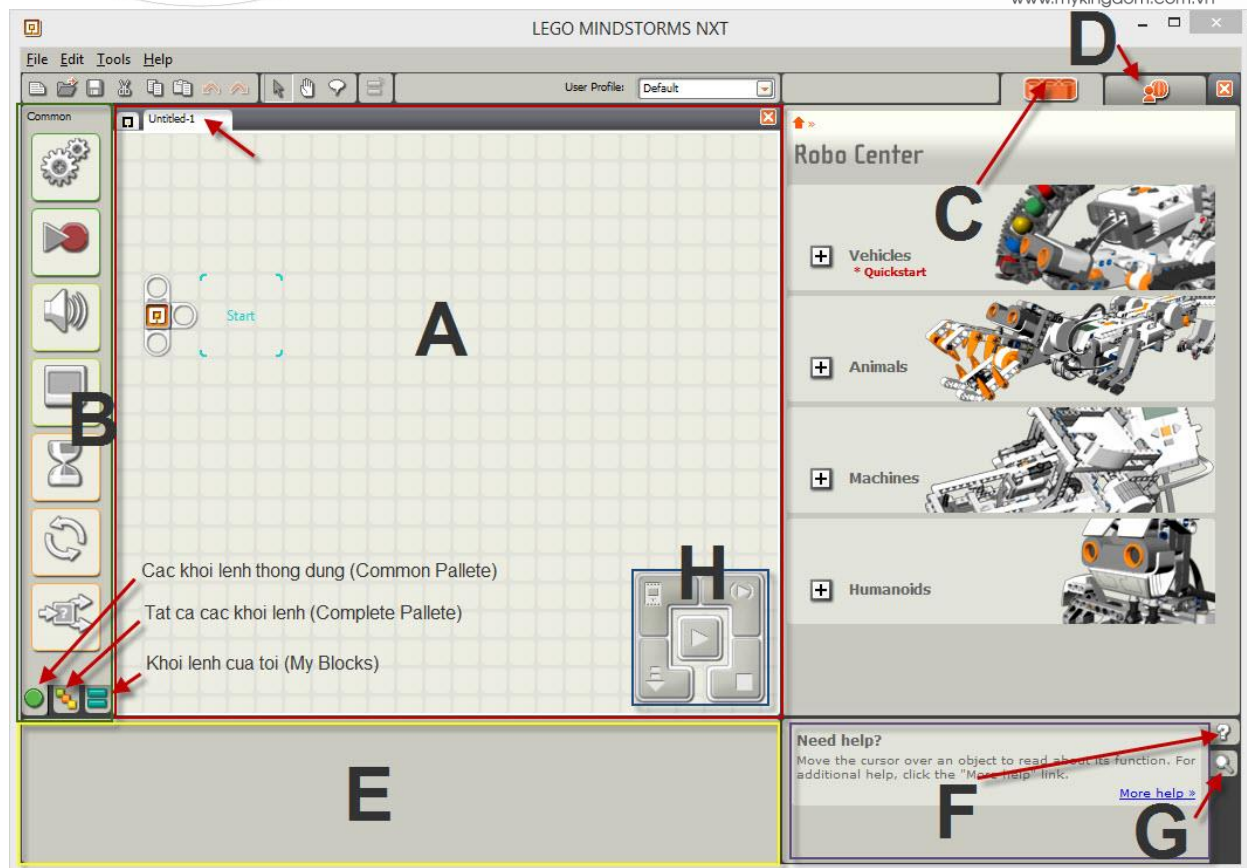


Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



Hình 10.1 : Màn hình chính của chương trình Lego MindStorms NXT.

Bấm vào nút Go của mục [Create new program] để tạo ra một chương trình mới chưa có nội dung và cũng chưa được đặt tên như hình . Trước khi bắt đầu viết chương trình đầu tiên, chúng ta hãy làm quen với các mục trên màn hình.



Hình 10.2 Các mục trên màn hình lập trình NXT- G.

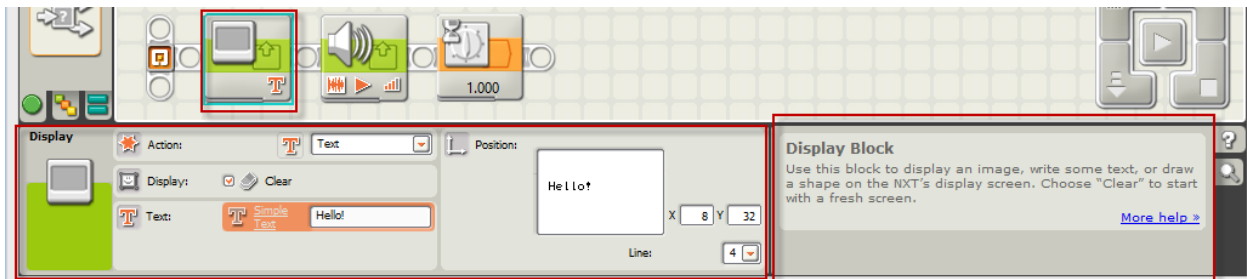
A: Vùng làm việc (Work Area), là trung tâm của màn hình, là nơi mà bạn sẽ thiết kế ra chương trình của bạn. Bạn sẽ kết nối các khối lệnh cần thiết cho chương trình của mình ở đây. Nếu bạn mở nhiều chương trình cùng một lúc thì bạn có thể chọn một chương trình để làm việc bằng cách bấm vào thẻ (tab) của chương trình đó.

B: Bảng các khối lệnh (Programming Palletes), nằm phía bên trái của màn hình. Bảng khối lệnh chứa các khối lệnh mà bạn có thể dùng cho chương trình của mình. Bảng khối lệnh gồm có ba thẻ: thẻ [Các khối lệnh thông dụng] (Common Palette) chứa các khối lệnh hay dùng nhất; thẻ [Tất cả các khối lệnh] (Complete Palette) chứa tất cả các khối lệnh, kể cả các khối lệnh có trong thẻ [Các khối lệnh thông dụng]; thẻ [Khối lệnh của tôi] (My Blocks) chứa các khối lệnh do bạn tạo ra.

C: Trung tâm Robo (Robo Center), nằm bên phải của Vùng làm việc. Để mở cửa sổ này bạn có thể phải bấm vào thẻ [Robo Center]. Cửa sổ Trung tâm Robo chứa các hướng dẫn để lắp ráp cho một số robot mẫu. Có bốn nhóm robot khác nhau được trình bày ở đây: Xe Cộ, Máy Móc, Động Vật, và Người Máy... Bạn nên dành thời gian để tìm hiểu và lắp ráp các robot này vì việc này sẽ rất có ích cho bạn trong việc làm quen với môi trường Lego Mindstorms. Khi không cần dùng thì bạn có thể đóng cửa sổ này để có một Vùng làm việc lớn hơn.

D: Thay vì chọn thẻ [Robo Center], bạn có thể chọn thẻ [My Portal] để mở cửa sổ Cổng thông tin của tôi. Trước đây, cửa sổ này có các đường dẫn đến các trang web chứa các thông tin liên quan đến Lego Mindstorm. Tuy nhiên, tại thời điểm viết cuốn sách này, Lego đã thay đổi một số cấu hình trên máy chủ dẫn đến việc cửa sổ này không hoạt động được và có thể làm cho chương trình bị tắt đột ngột. Vì vậy nếu gặp lỗi này, bạn không nên bấm vào thẻ [My Portal] nữa.

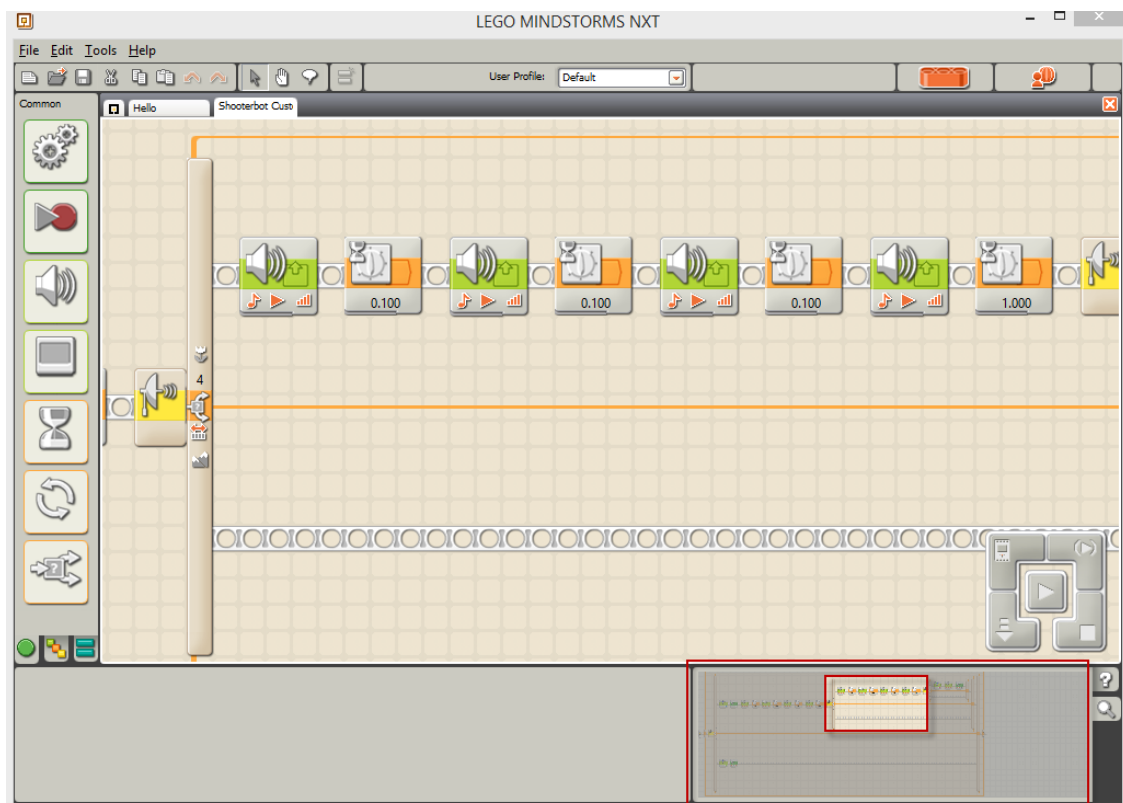
E: Bảng cấu hình (Configuration Panel) nằm ở góc trái phía dưới của màn hình. Khi bạn chọn một khối lệnh nào đó thì bảng cấu hình sẽ cho phép bạn nhập các thông số chi tiết cho khối lệnh này.



Hình 10.3: Khối lệnh [Display] và bảng cấu hình tương ứng.

F: Bảng trợ giúp (Help Panel) nằm ở góc dưới bên phải của màn hình. Nó chứa vài dòng mô tả vắn tắt về khối lệnh đang được chọn. Bấm chuột vào đường dẫn [More Help], bạn sẽ mở toàn bộ file trợ giúp để có thể tra cứu chi tiết hơn về các khối lệnh. Để mở Bảng trợ giúp, bạn có thể bấm vào thẻ [Help] có hình dấu hỏi như trên hình.

G: Bản đồ chương trình sẽ xuất hiện nếu bạn bấm vào thẻ [Map] có hình cái kính lúp. Nếu chương trình của bạn đang mở là khá lớn thì việc di chuyển qua lại giữa các khối lệnh sẽ chậm chạp và khó khăn nếu không có cửa sổ Bản đồ chương trình này. Toàn bộ chương trình của bạn sẽ được thu nhỏ cho vừa với cửa sổ bản đồ. Khi bạn bấm chuột vào một vùng tùy chọn của cửa sổ này thì trên cửa sổ Vùng làm việc sẽ hiện ra phần tương ứng của chương trình để cho bạn dễ dàng thao tác.



Hình 10.4: Bản đồ chương trình và cửa sổ Vùng làm việc.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

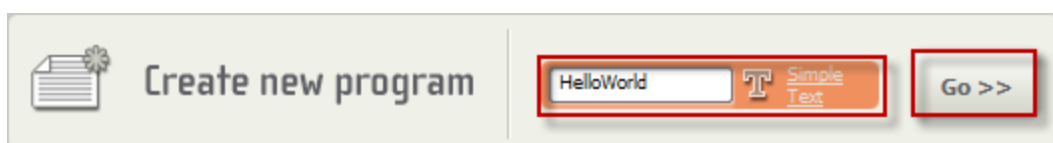
H: Bảng điều khiển (Controller) nằm ở góc dưới bên phải của Vùng làm việc. Bảng điều khiển giúp cho bạn nạp chương trình vào và chạy chương trình trên khối vi điều khiển NXT. Ngoài ra nó cũng có thể cho bạn thấy được một số thông tin cơ bản về khối NXT của bạn như danh sách các file trên NXT, phiên bản của firmware, bộ nhớ còn trống, mức năng lượng hiện tại của pin, v.v...

II. Thảo một chương trình NXT-G đơn giản:

Trên Vùng làm việc, bạn sẽ thấy có hình vẽ giống như các thanh xà của bộ lắp ráp Lego, nó được gọi là dòng lệnh (sequence beam). Thảo một chương trình trong môi trường NXT-G là làm những thao tác kéo thả chuột để đem các khối lệnh từ Bảng khối lệnh đặt trên dòng lệnh; và thiết lập các thông số cho các khối lệnh thông qua Bảng cấu hình. Khi bạn chạy chương trình, thì bộ vi điều khiển NXT sẽ thực hiện tuần tự từng khối lệnh theo thứ tự sắp xếp của chúng trên dòng lệnh. Trên một dòng lệnh, bộ vi điều khiển NXT thường phải thực hiện xong một khối lệnh rồi mới thực hiện khối lệnh kế tiếp (Một số khối lệnh không theo quy tắc này sẽ được đề cập sau). Chương trình sẽ kết thúc khi bộ vi điều khiển thực hiện xong khối lệnh cuối cùng của dòng lệnh.

Người ta có thể đọc hiểu các chương trình viết bằng ngôn ngữ lập trình NXT-G một cách khá dễ dàng. Tuy nhiên để bộ vi điều khiển NXT hiểu được chúng là một chuyện khác. Chương trình viết bằng ngôn ngữ NXT-G (mã nguồn) cần được chuyển thành một tập hợp các lệnh phù hợp để firmware trong bộ vi điều khiển có thể hiểu và thực hiện các lệnh này. Quá trình chuyển đổi đó được gọi là quá trình biên dịch. Bạn dùng Bảng điều khiển để biên dịch một chương trình NXT-G và tải nó vào bộ vi điều khiển NXT để chạy chương trình này.

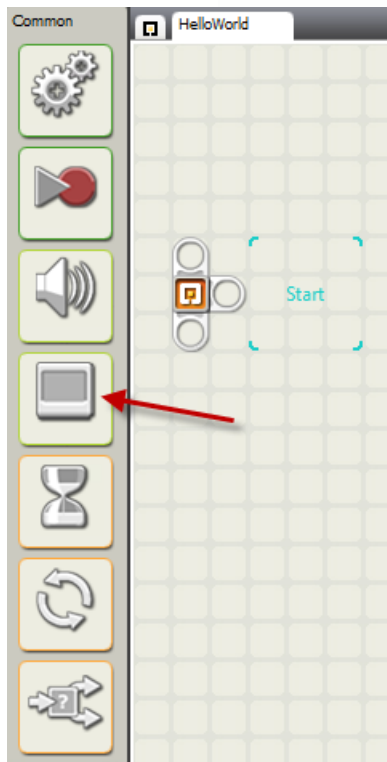
Sau đây là một các bước mà bạn sẽ phải làm khi viết một chương trình. Giả sử bạn cần viết một chương trình để hiện ra dòng chữ Hello World trên màn hình LCD của khối vi điều khiển NXT. Sau khi khởi động chương trình Lego MindStorms NXT, bạn nhập HelloWorld vào ô [Create new Program] và bấm nút Go.



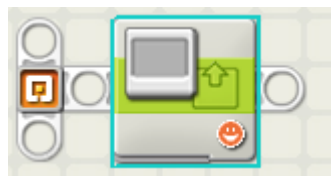
Hình 10.5: Tạo một chương trình mới.

Một Vùng làm việc mới sẽ xuất hiện. Trên vùng này có sẵn một khối lệnh Starting Point (Điểm bắt đầu). Khối lệnh này cũng có sẵn 3 điểm kết nối để bạn có thể gắn thêm các khối lệnh khác để tạo ra một hay nhiều dòng lệnh. Thực hiện các bước sau để tạo một khối lệnh Display trong chương trình:

Chọn khối lệnh Display từ thẻ Các khối lệnh thông dụng. Kéo khối lệnh này và thả vào vùng có đánh dấu [Start].



Hình 10.6 : Chọn khối lệnh Display từ thẻ “Common Palette”

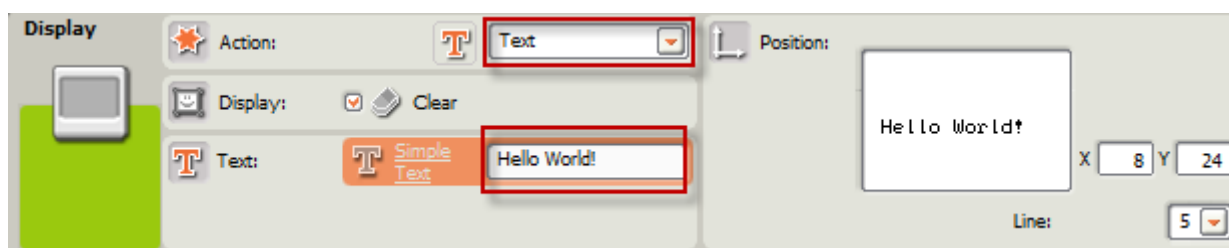


Lúc này chương trình của bạn sẽ trông giống như hình sau:

Bấm chuột vào Bảng cấu hình, chọn Text cho mục Action. Điều này có nghĩa là lệnh liên quan đến hiển thị các dòng chữ.

Xóa dòng chữ “Mindstorms NXT”, và thay nó bằng “Hello World!” cho mục Text. Đây chính là nội dung cần hiển thị trên màn hình LCD của khối vi điều khiển.

Giữ nguyên các tham số mặc định khác. Mục Clear được đánh dấu có nghĩa là phải xóa nội dung của màn hình trước khi hiển thị đoạn văn bản trong mục Text. Trong phần Position, bạn sẽ được xem trước cách thức khối NXT hiển thị dòng chữ “Hello World!” trên LCD. Line = 4 có nghĩa là dòng chữ sẽ được hiển thị trên hàng thứ tư của LCD. X=8, Y=32 chính là tọa độ bắt đầu của dòng chữ. Lưu ý gốc tọa độ chính là góc trái phía dưới của LCD, trục X hướng qua phải, trục Y hướng lên trên.

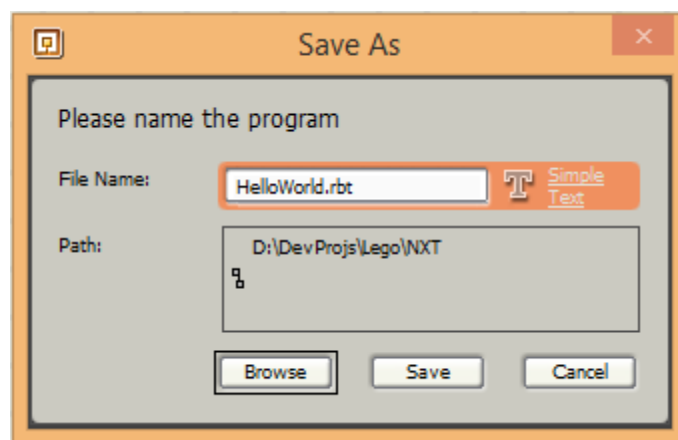


Hình 10.7 Bảng cấu hình của khối Display.



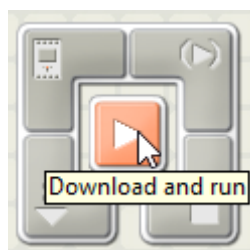
Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

Lưu chương trình HelloWorld của bạn lên đĩa bằng cách chọn menu File>Save. Khi bạn lưu chương trình của mình lần đầu, chương trình Lego MindStorms NXT 2.0 sẽ hiện ra một hộp hội thoại cho phép bạn đặt tên file và chọn vị trí để lưu như hình sau. Tên file mặc định là HelloWorld.rbt bởi vì HelloWorld là cái tên mà bạn đặt cho chương trình của mình lúc đầu tạo ra nó, bạn có thể đổi thành một cái tên nào đó tùy ý, miễn là nó có ý nghĩa liên quan đến nội dung của chương trình. Sau khi bạn bấm nút Save, thì file HelloWorld.rbt sẽ được tạo ra theo đường dẫn (Path) bạn đã chọn. File này chứa tất cả các thông tin về chương trình của bạn, bao gồm các khối lệnh mà bạn dùng và cách sắp xếp các khối lệnh cũng như là các cấu hình mà bạn đã thiết lập cho chúng. Do file lưu các thông tin theo một cấu trúc đặc biệt chỉ có chương trình Lego MindStorms NXT mới hiểu nên bạn chỉ có thể xem và thay đổi file này trong môi trường Lego MindStorms.

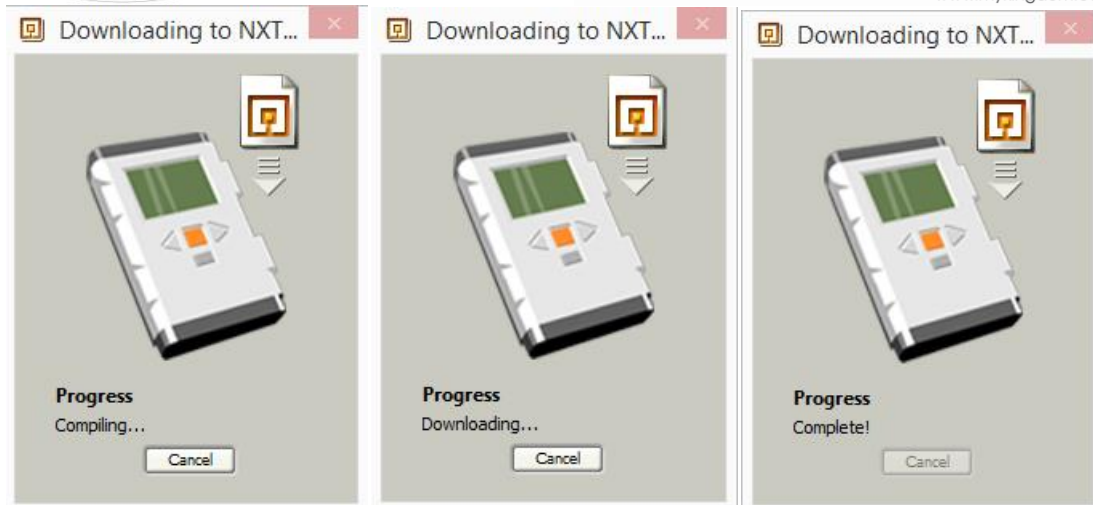


Hình 10.8 : Lưu chương trình.

Sau khi lưu file xong, đã đến lúc bạn chạy thử chương trình của mình. Trước tiên, bạn hãy bật nguồn khối vi điều khiển NXT và nối nó với máy tính bằng sợi cáp USB. Sau đó, bấm chuột vào nút [Download and Run] của Bảng điều khiển để tải chương trình vào khối vi điều khiển rồi chạy chương trình ở đó.



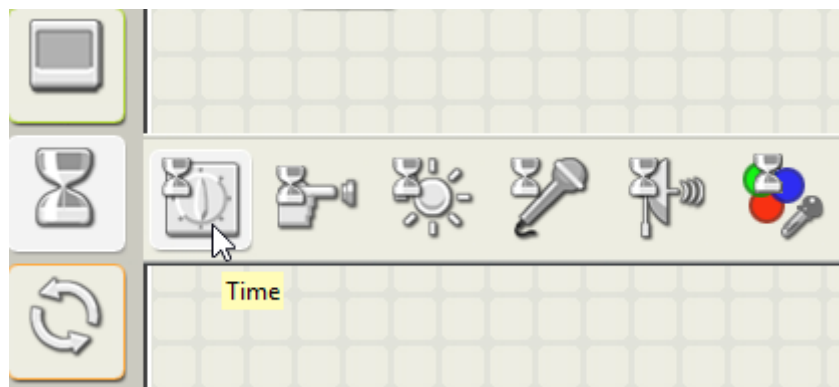
Sau khi bấm nút [Download and Run] chương trình Lego MindStorms NXT sẽ hiển thị một hộp hội thoại và lần lượt hiện ra các thông báo Đang biên dịch, Nạp chương trình, và Hoàn tất như hình.



Hình 10.9 : Các thông báo của hộp hội thoại trong quá trình biên dịch và nạp chương trình vào khối vi điều khiển NXT.

Đến đây lại phát sinh một vấn đề. Dù kiểm tra nhiều lần, bạn vẫn không thấy dòng chữ “Hello World!” hiện ra ở giữa màn hình LCD của khối vi điều khiển hình như mong muốn! Bạn chỉ thấy dòng thông báo là chương trình HelloWorld đã kết thúc (HelloWorld Done). Chuyện gì đã xảy ra? Có khả năng là chương trình có hiển thị dòng chữ “Hello World!”, rồi ngay sau đó chương trình HelloWorld kết thúc và bộ vi điều khiển NXT xóa sạch màn hình để hiển thị dòng chữ “HelloWorld Done”. Như vậy có thể là do khoảng thời gian hiển thị của dòng chữ “Hello World!” quá ngắn nên mắt thường không thể thấy kịp. Vậy giải pháp có thể dùng là cho chương trình chờ vài giây trước khi kết thúc.

Để khiến cho chương trình chờ trong một khoảng thời gian trước khi làm một việc gì bạn có thể dùng khối lệnh [Wait Time] của Bảng khối lệnh thông dụng. Các bước sửa lỗi cho chương trình này như sau:
Đưa chuột đến biểu tượng cái đồng hồ cát trong Bảng khối lệnh. Tất cả các khối lệnh liên quan đến lệnh Wait (chờ) sẽ hiện ra như hình.



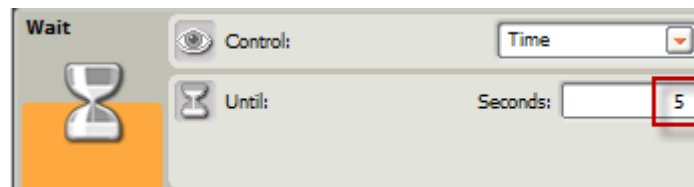
Hình 10.11: Khối lệnh [Wait Time] trên Bảng khối lệnh (thẻ [Các khối lệnh thông dụng]).

Kéo và thả (drag and drop) khối lệnh [Wait Time] ngay sau khối lệnh Display trên dòng lệnh.



Hình 10.12: Thêm khối lệnh [Wait Time] vào chương trình.

Mặc định thì khối lệnh [Wait Time] sẽ khiến cho chương trình ngưng lại trong thời gian 1 giây. Bạn có thể tăng khoảng thời gian này lên bằng cách thay đổi các thông số trong Bảng cấu hình của khối lệnh này.



Hình 10.13 : Nhập 5 cho mục Seconds. Khối [Wait Time] với các thông số này sẽ khiến cho chương trình ngưng lại 5 giây trước khi thực hiện khối lệnh kế tiếp.

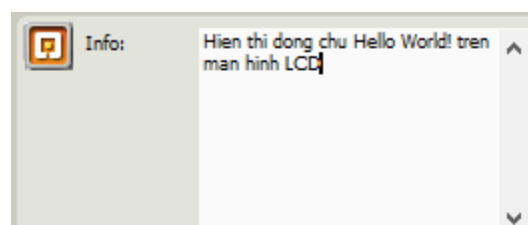
Bấm nút [Download and Run] để biên dịch, tải chương trình mới lên khối vi điều khiển NXT, và chạy nó. Lúc này bạn sẽ thấy được dòng chữ “Hello World!” hiện ra ở giữa màn hình LCD trong 5 giây trước khi nó bị xóa đi.

Như vậy với phần sửa lỗi (debug) như trên, chương trình HelloWorld đã chạy đúng như ý chúng ta muốn. Lúc này bạn có lẽ cũng muốn thêm một vài ghi chú vào chương trình để tiện cho việc tra cứu và giải thích cách hoạt động của chương trình sau này. Các ghi chú này thật ra không có đóng góp vai trò gì vào cách chương trình hoạt động, chúng chỉ có tác dụng giúp bạn lưu thông tin để tra cứu sau này mà thôi. Ví dụ bạn muốn nói rõ mục đích của chương trình này và giải thích tại sao bạn thêm khối lệnh [Wait Time] vào chương trình. Các bước có thể tiến hành như sau:

Bấm vào biểu tượng MindStorms ở đầu dòng lệnh để mở Bảng cấu hình của nó.



Nhập mục đích của chương trình (ví dụ: Hien thi dong chu Hello World! tren man hinh LCD) vào mục Info.



Hình 10.14 : Mục Info của điểm bắt đầu dòng lệnh thường chứa mục đích của chương trình.

Để giải thích cho sự tồn tại của khối lệnh [Wait Time], ta có thể ghi chú vào ngay Vùng làm việc cạnh khối lệnh [Wait Time]. Bấm chuột nhanh 2 lần (double-click) vào vị trí cần thêm ghi chú, lúc này trên màn hình sẽ xuất hiện một ô chữ nhật nhỏ màu đen và bạn có thể nhập câu ghi chú vào. Bấm phím Enter để xuống dòng nếu cần. Sau khi nhập xong ghi chú bạn bấm chuột vào một vị trí nào đó nằm bên ngoài ô ghi chú để báo cho chương trình NXT-G là bạn đã nhập xong. Lưu file lên đĩa để bảo đảm là các ghi chú này sẽ đi kèm với chương trình trước khi bạn đóng chương trình NXT-G.



Hình 10.15 : Nhập ghi chú lên vùng làm việc gồm 2 bước: chọn vị trí cần đặt ghi chú và nhập ghi chú.

III. Các khái niệm cơ bản trong ngôn ngữ lập trình NXT-G:

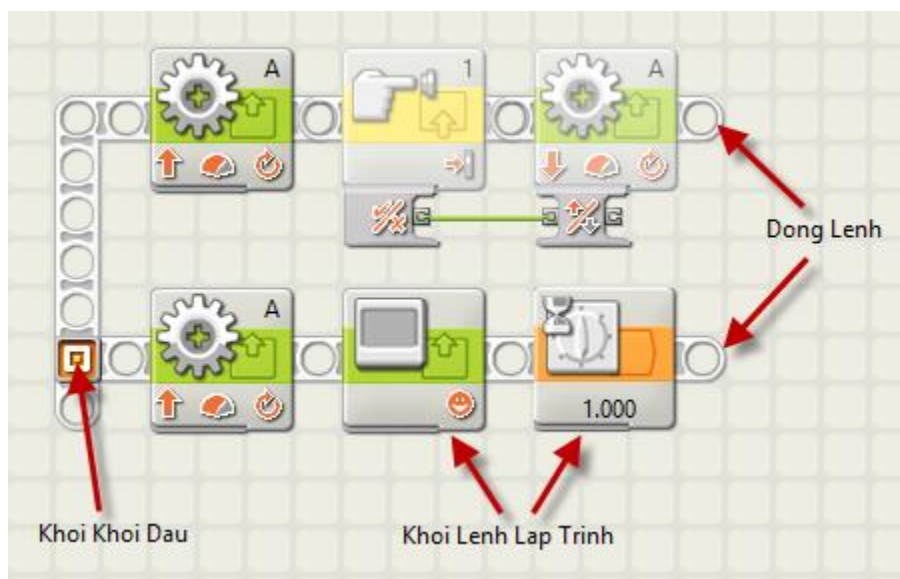
III.1 Cấu trúc của một chương trình NXT-G:

Qua ví dụ trước, có thể bạn đã nhận ra, một chương trình NXT-G bao gồm các thành phần sau:

Khởi Khởi đầu (Starting point): là nơi bắt đầu của chương trình, nếu bạn muốn các khối lệnh lập trình khác được thực thi thì những khối lệnh này phải được kết nối với khối Khởi đầu thông qua các dòng lệnh.

Dòng lệnh (Sequence beam) là nơi gắn kết các khối lệnh lập trình, nó chỉ ra thứ tự mà các khối lệnh cần được thực thi. Chương trình kết thúc khi nó thực thi xong khối lệnh cuối cùng của dòng lệnh. Nếu bạn kiểm soát được dòng lệnh thì bạn sẽ điều khiển được việc thực thi các khối lệnh trong chương trình của mình.

Khối lệnh lập trình (Programming block) là đơn vị điều khiển nhỏ nhất của chương trình. Mỗi khối lệnh có khả năng ra lệnh cho khối vi điều khiển NXT thực hiện một nhiệm vụ cơ bản nào đó. Thông thường, khi lập trình để điều khiển quá trình thực hiện một mục tiêu, bạn phải chia nhỏ mục tiêu đó thành những mục tiêu nhỏ hơn để việc thực hiện chúng dễ dàng hơn. Quá trình chia nhỏ có thể phải lập đi lập lại trên các mục tiêu mới này cho đến khi bạn đạt được các mục tiêu nhỏ phù hợp. Lúc đó, ứng với mỗi mục tiêu nhỏ, bạn sẽ dùng một hoặc nhiều khối lệnh lập trình để thực hiện nó.



Hình 10.16 : Cấu trúc cơ bản của một chương trình NXT-G.

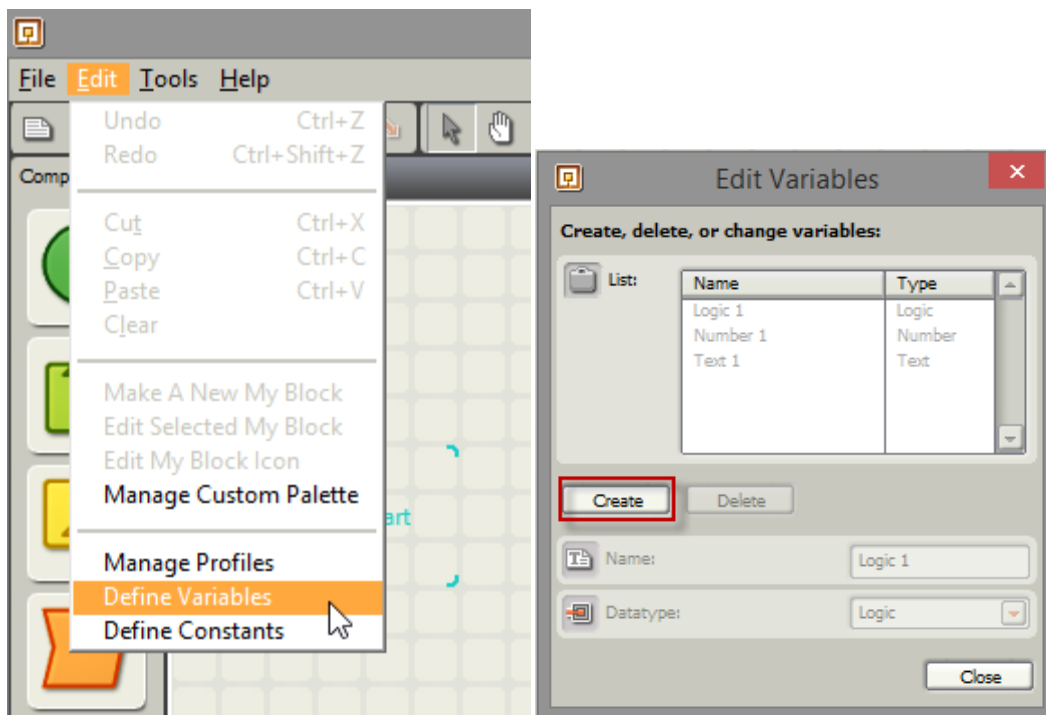
Những phần tiếp theo đây sẽ mô tả các khối lệnh thuộc các nhóm chức năng khác nhau.

III.2 Biến số (Variable) , Kiểu dữ liệu (Data Type), Hằng số (Constant) và các khối lệnh liên quan:

Biến số là một nơi trong bộ nhớ của khối điều khiển NXT được dùng để lưu lại thông tin nhằm phục vụ cho việc sử dụng sau đó. Tương tự như khái niệm biến số trong toán học, khi bạn gán cho một biến số $N = 2$, thì từ lúc này trở đi nếu bạn nói đến N (ví dụ trong biểu thức $X = N^2 + 1$) tức là nói đến số 2, cho đến khi bạn gán N với một giá trị mới, ví dụ là 5 chẳng hạn. Việc dùng biến số để lưu lại các kết quả tính toán phức tạp giúp cho bạn khỏi phải mất công tính lại kết quả đó mỗi khi cần dùng. Ngoài ra, nó cũng giúp cho bạn viết chương trình một cách rõ ràng trong sáng hơn khi bạn cần tổng hợp nhiều kết quả trung gian thành một giá trị mới.

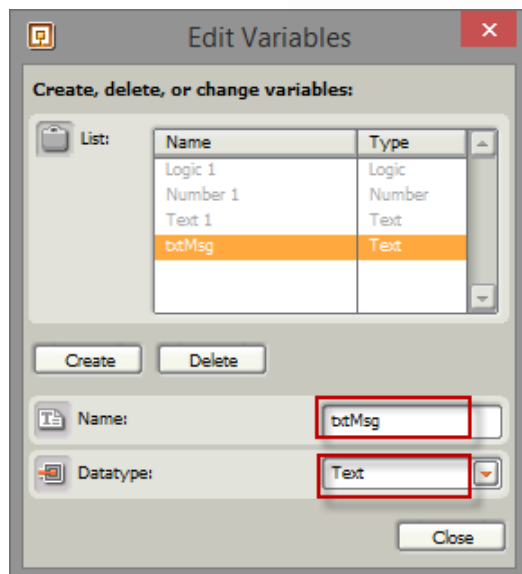
Trong NXT-G để sử dụng một biến số bạn phải qua 2 bước: định nghĩa một biến số và sau đó thì dùng (xử lý) nó thông qua khối lệnh Variable trong chương trình của mình.

Để định nghĩa một biến số bạn dùng menu Edit>Define Variables để mở ra hộp hội thoại Edit Variables.



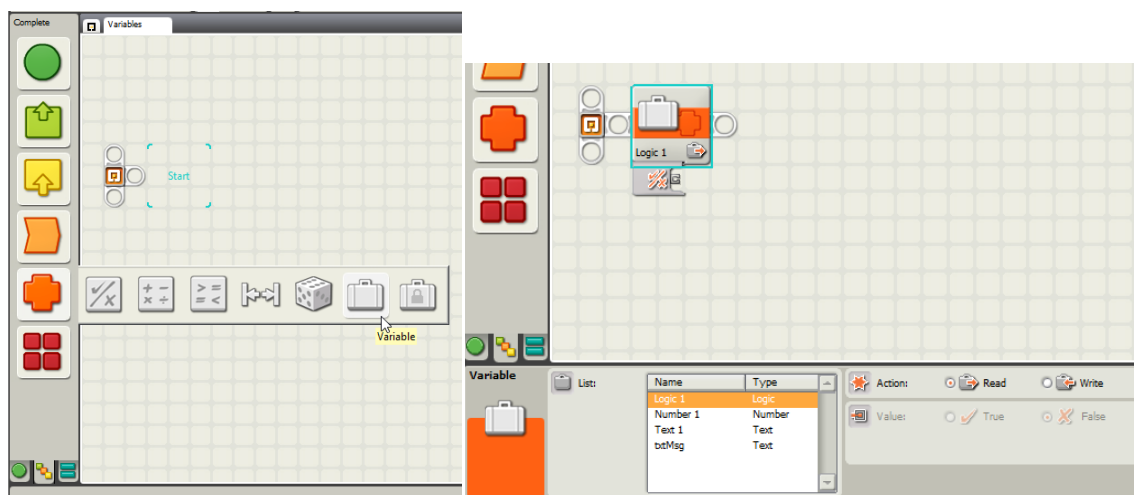
Hình 10.17: Dùng menu [Edit>Define Variables] để mở hộp hội thoại [Edit Variables].

Bấm vào nút [Create] để tạo ra một biến số mới và nhập txtMsg cho mục [Name] và chọn Text cho mục [Datatype] để tạo ra một biến số tên là txtMsg có kiểu dữ liệu là Text (dữ liệu là văn bản, chuỗi các ký tự). Bấm nút [Close] để kết thúc phần tạo biến số mới này.



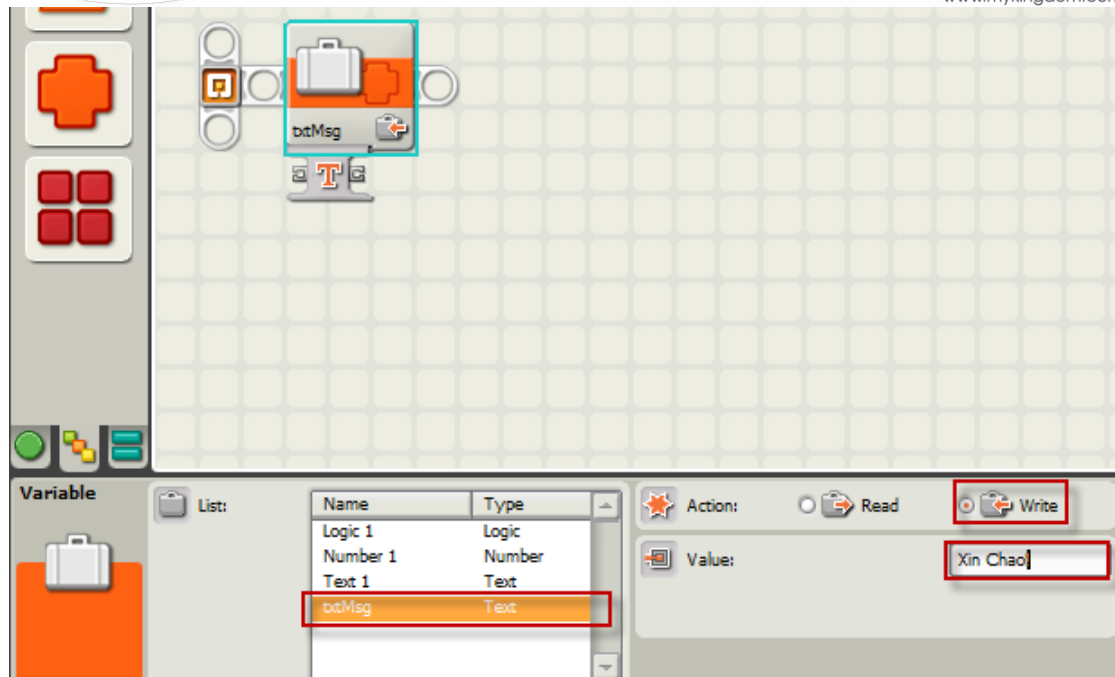
Hình 10.18: Tạo một biến số mới txtMsg có kiểu dữ liệu là Text.

Sau khi tạo ra một biến số bạn cần phải gán cho nó với một giá trị ban đầu nào đó để có thể sử dụng biến số trong chương trình của mình. Chọn thẻ [Complete] của Bảng các khối lệnh, sau đó đưa chuột đến biểu tượng [Data] hình chữ thập để hiển thị tất cả các khối lệnh dành cho việc xử lý dữ liệu. Chọn khối lệnh Variable và đưa khối lệnh này vào dòng lệnh.



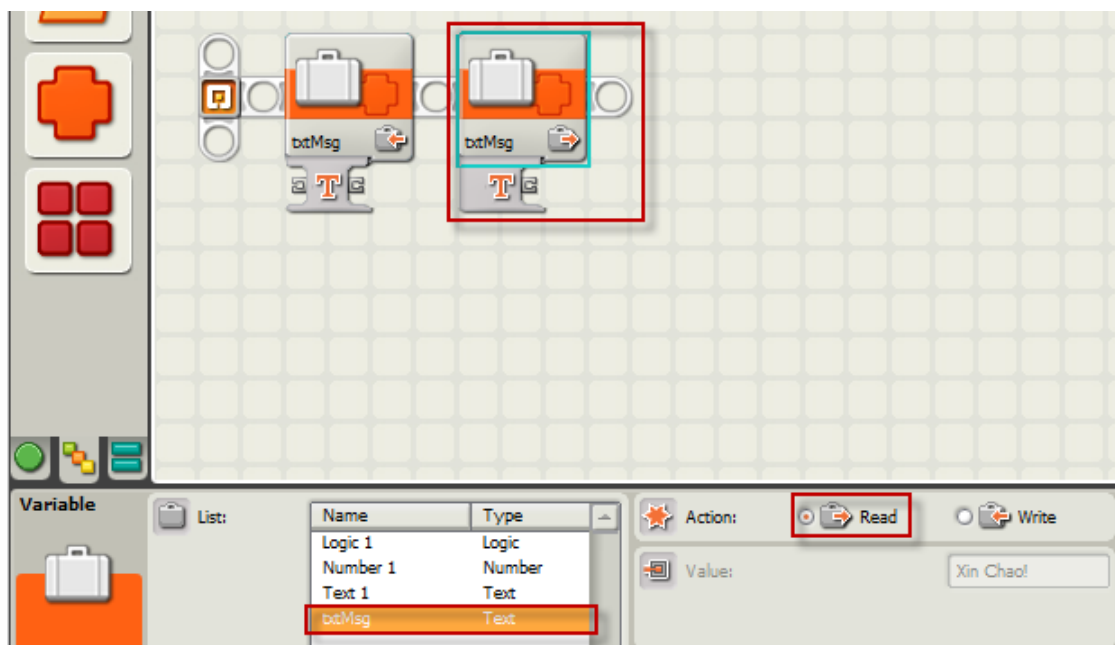
Hình 10.19: Kéo và thả một khối lệnh Variable vào chương trình.

Sau khi đưa khối lệnh Variable vào chương trình, bạn cần phải cho chương trình biết là khối lệnh này sẽ làm việc với biến số nào, và cách khối lệnh sẽ làm việc ra sao. Bạn có thể lưu (Write) một giá trị mới vào biến số đó, hoặc là đọc ra (Read) giá trị mà biến số đó đang lưu giữ. Ví dụ, để ghi vào biến số txtMsg ở trên chuỗi ký tự “Xin Chao!” (bạn dùng tiếng Việt không có dấu, vì hiện tại khối vi điều khiển NXT vẫn chưa hỗ trợ tiếng Việt), thì bạn bấm chuột vào khối lệnh Variable trên Vùng làm việc, chọn biến số txtMsg cho mục [List], chọn giá trị [Write] cho mục [Action], và nhập chuỗi “Xin Chao!” vào ô [Value] của Bảng cấu hình.



Hình 10.20 Tạo một biến mới.

Để đọc ra giá trị mà biến số txtMsg đang lưu giữ thì bạn cũng phải dùng khối lệnh [Variable]. Tuy nhiên thay vì chọn giá trị [Write], bạn phải chọn giá trị [Read] cho mục [Action].



Hình 10.21: Dùng khối lệnh [Variable] để đọc giá trị của một biến số.

Bạn sẽ thấy ô Value bị mờ đi (bạn không được phép nhập dữ liệu cho nó), vì đây là thao tác đọc dữ liệu. Biến số txtMsg chứa giá trị gì thì bạn sẽ đọc ra giá trị đó, chứ bạn không thay đổi được nó lúc này.

Tuy nhiên bạn không đọc giá trị của một biến số mà không có mục đích. Bạn cần nó để thực hiện một công việc cụ thể nào đó. Tức là bạn phải lấy ra giá trị của biến số và chuyển giá trị này đến một chỗ nào cần nó. Thường thì các khối lệnh lập trình đều cần có các dữ liệu đầu vào để chúng có thể thực hiện các thao tác điều

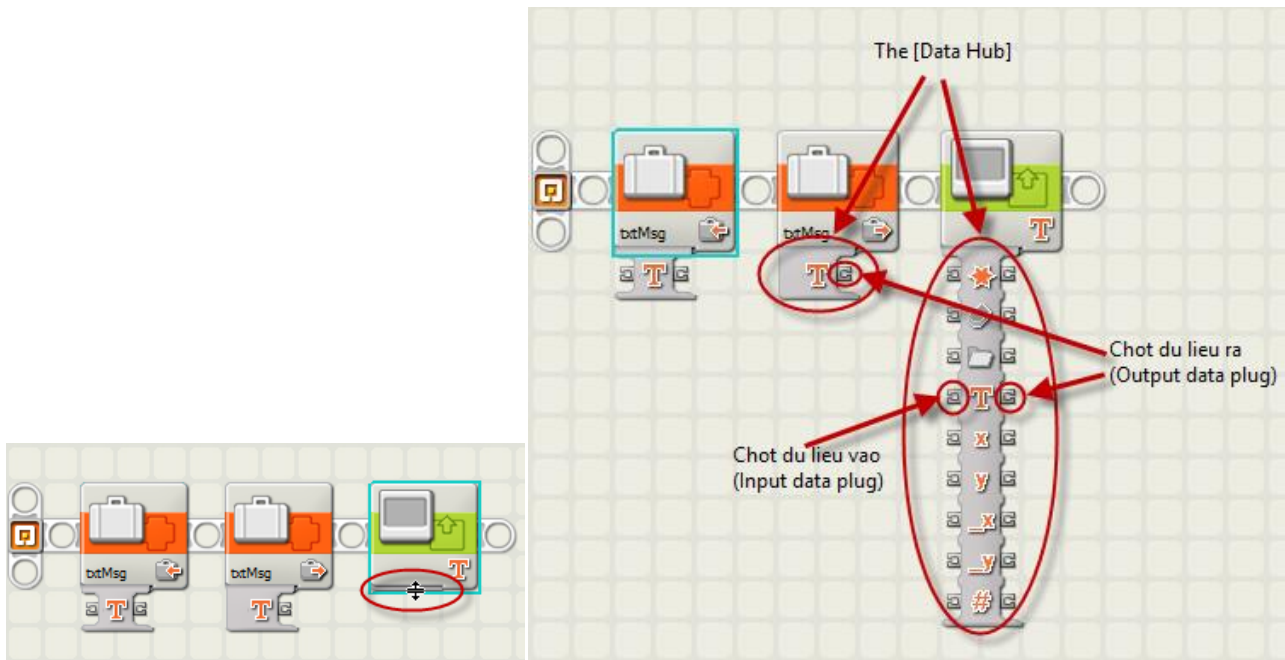
khuyến phù hợp. Lấy ví dụ khối lệnh [Display], nó cần phải được thiết lập các giá trị cụ thể cho các mục trong Bảng cấu hình của nó.

Thông số	Kiểu dữ liệu	Các giá trị hợp lệ	Ý nghĩa của các giá trị	Mục này không dùng khi...
Action	Number	0-5	0=Image (hiển thị hình ảnh lấy từ file) 1=Text (hiển thị văn bản) 2=Point (vẽ điểm) 3=Line (vẽ đoạn thẳng) 4=Circle (vẽ đường tròn) 5=Reset (trở về màn hình hệ thống)	
Clear	Clear	Logic	True/False	True=Xóa màn hình False=Không xóa màn hình
Filename	Text	Tối đa 15 ký tự	Tên của file hình ảnh	Action khác Image (0)
Text	Text	Chuỗi ký tự	Chuỗi ký tự cần hiển thị trên màn hình	Action khác Text (1)
X	Number	0-99 (nếu nhỏ hơn 0 thì có thể bạn sẽ không thấy phần cần hiển thị)	Tọa độ trục hoành X. Cặp số tọa độ (X, Y) xác định vị trí hiển thị hình ảnh, chuỗi ký tự, điểm, điểm bắt đầu của đoạn thẳng, tâm vòng tròn. Tùy theo Action có giá trị gì thì (X, Y) sẽ có một ý nghĩa khác.	
Y	Number	0-63 (nếu nhỏ hơn 0 thì có thể bạn sẽ không thấy phần cần hiển thị)	Tọa độ trục tung Y. Cặp số tọa độ (X, Y) xác định vị trí hiển thị hình ảnh, chuỗi ký tự, điểm, điểm bắt đầu của đoạn thẳng, tâm vòng tròn. Tùy theo Action có giá trị gì thì (X, Y) sẽ có một ý nghĩa khác.	
End point X	Number	0-99	Tọa độ trục hoành _x. Cặp số tọa độ (_x, _y) xác định vị trí điểm cuối khi vẽ đoạn thẳng (Action=Line).	Action khác Line (3)
End point Y	Number	0-63	Tọa độ trục tung _y. Cặp số tọa độ (_x, _y) xác định vị trí điểm cuối khi vẽ đoạn thẳng (Action=Line).	Action khác Line (3)
Radius	Number	0-120	Bán kính đường tròn (Action = Circle)	Action khác Circle (4)

Bảng 10.1 : Danh sách các thông số đầu vào cần có cho khối lệnh [Display]. Các hình vẽ đi kèm là các biểu tượng của các thông số trên thẻ [Data Hub].

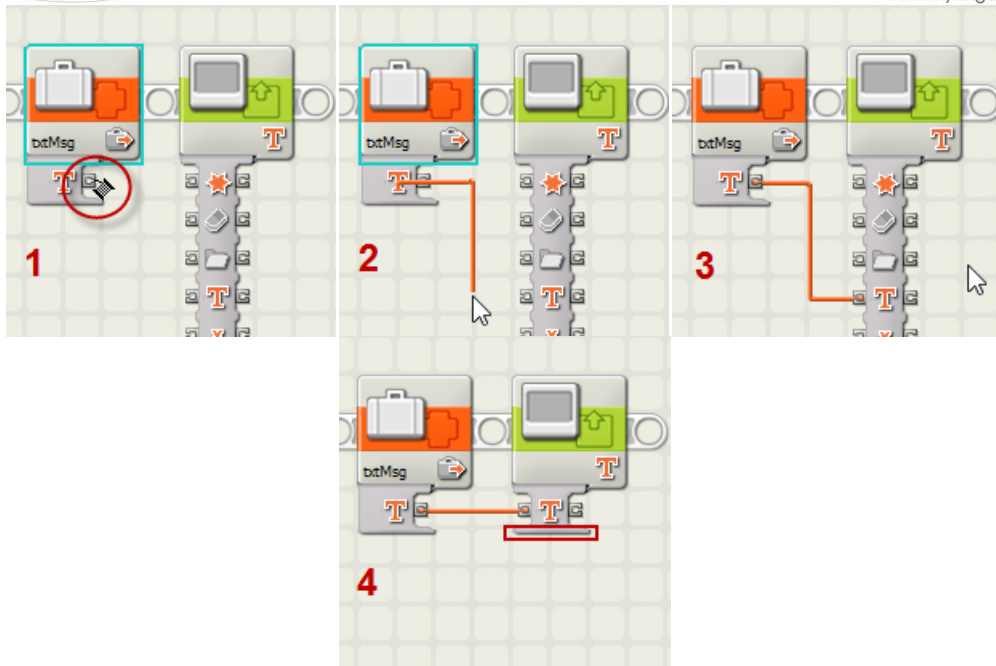
Do đó, nếu bạn muốn hiển thị ra màn hình LCD của khối vi điều khiển NXT chuỗi ký tự mà biến số txtMsg đang lưu, thì bạn sẽ phải kết hợp khối lệnh [Variable] để đọc giá trị từ biến số txtMsg rồi truyền giá trị đó qua mục dữ liệu đầu vào [Text] của khối lệnh [Display] để hiển thị trên màn hình. Sau khi gắn khối lệnh [Display] vào dòng lệnh bạn cần thực hiện các bước sau:

Đưa chuột đến vị trí thuộc phần đáy của khối này (thẻ [Data Hub]), con trỏ sẽ chuyển sang một biểu tượng mới với hình vẽ 2 đoạn thẳng song song cùng với 2 mũi tên ngược chiều nhau. Bấm chuột vào đó để mở thẻ [Data Hub] là nơi cho phép bạn thực hiện các thao tác truyền dữ liệu.



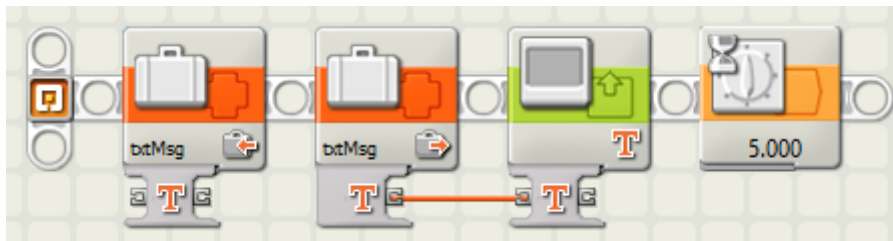
Hình 10.22: Mở thẻ [Data Hub] của khối lệnh [Display].

Thẻ [Data Hub] chứa các biểu tượng của các tham số của khối lệnh và các chốt dữ liệu tương ứng [data plug]. Các chốt phía bên trái là các chốt dữ liệu vào [Input data plug], các chốt phía bên phải là chốt dữ liệu ra [Output data plug]. Đưa chuột đến vị trí ngay phía trên chốt dữ liệu ra của thông số [Value] của khối lệnh [Variable], con trỏ chuột sẽ đổi sang hình cuộn dây. Nhấp chuột (Single click), kéo con trỏ đến chốt dữ liệu vào của thông số [Text] của khối lệnh Display, rồi nhấp chuột tiếp vào đó. Lúc này bạn sẽ thấy một đường màu cam nối giữa 2 chốt dữ liệu nêu trên. Thao tác bạn vừa làm gọi là kết nối dữ liệu (Data Wiring). Lưu ý là thông số [Action] của khối lệnh [Variable] đang là Read, tức là cho phép đọc dữ liệu của khối lệnh này, nên bạn mới có thể dùng chốt dữ liệu ra của nó để chuyển dữ liệu qua một khối lệnh khác. Để đóng thẻ [Data Hub] và chỉ hiển thị phần chốt dữ liệu của thông số đang dùng, bạn bấm vào đáy của thẻ này.



Hình 10.23 : Kết nối dữ liệu cho các khối lệnh – Đọc giá trị từ khối lệnh [Variable] và chuyển giá trị này cho khối lệnh [Display].

Đến đây, bạn thêm khối lệnh [Wait Time] để cho chương trình có thời gian hiển thị là có thể chạy thử được rồi.



Hình 10.24 : Chương trình in ra màn hình chuỗi ký tự lưu trong biến số.

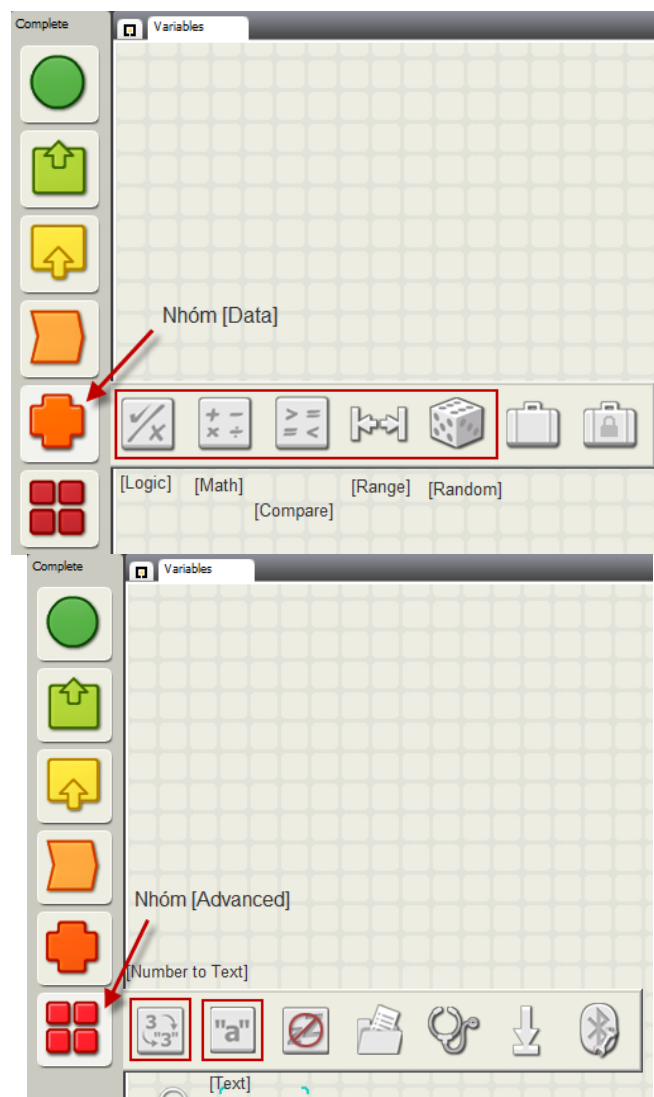
Mỗi biến số chỉ có thể lưu các giá trị của một kiểu dữ liệu duy nhất. Trong ngôn ngữ lập trình NXT-G, ta có 3 kiểu dữ liệu cơ bản, đó là: Text, chuỗi ký tự, bao gồm các ký tự chữ cái, ký tự số, và các dấu câu (ta thường dùng chuỗi ký tự để hiển thị thông tin trên màn hình LCD); Number, số; và Logic, kiểu dữ liệu luận lý, chỉ có 2 giá trị True (Đúng) và False (Sai). Do đó, bạn không thể gán một giá trị của kiểu dữ liệu này cho một biến số thuộc kiểu dữ liệu khác, hoặc truyền dữ liệu của một khối lệnh qua chốt dữ liệu vào của một khối lệnh khác mà kiểu dữ liệu của chúng khác nhau. Chẳng hạn, chương trình NXT-G sẽ báo lỗi khi bạn cố gắng truyền dữ liệu từ biến txtMsg (kiểu dữ liệu Text) qua cho thông số tọa độ trục hoành X (kiểu dữ liệu Number).

IV. Các khối lệnh xử lý dữ liệu:

Các dữ liệu ban đầu mà bạn có được thường không có ích ngay đối với bạn. Bạn phải tính toán, phân tích hay tổng hợp những dữ liệu này thành những thông tin cần thiết để bạn căn cứ vào đó mà ra quyết định. Lấy ví dụ bạn cần phải có mặt tại một vị trí cách chỗ bạn đang đứng là 60 km sau 2 giờ đồng hồ nữa. Khoảng cách 60 km và thời gian còn lại 2 giờ chính là các dữ liệu ban đầu. Bạn phải xử lý chúng bằng cách chia

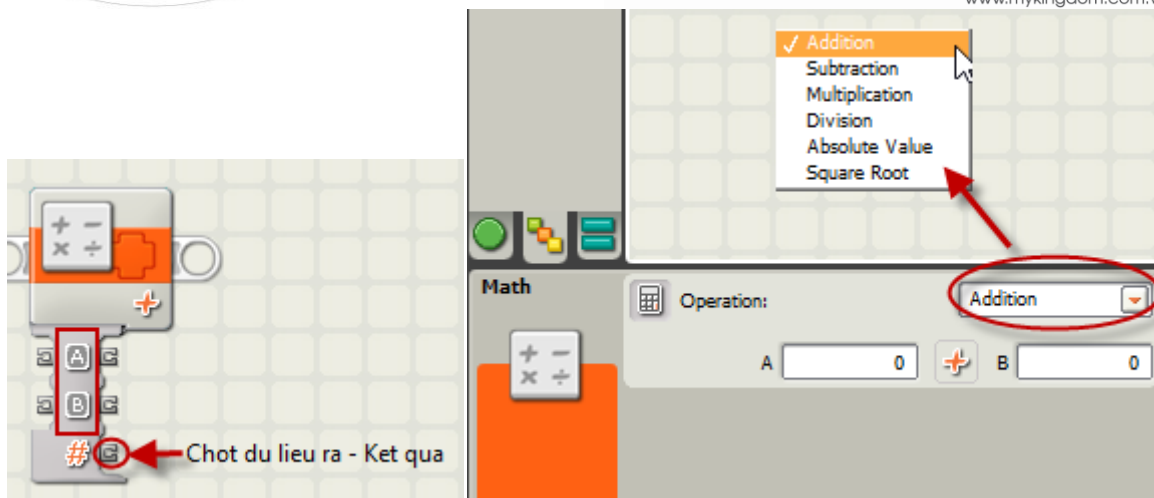
khoảng cách cho thời gian để biết được vận tốc tối thiểu mà bạn cần di chuyển để có mặt tại vị trí cần thiết đúng giờ. $60/2 = 30$ km/giờ chính là vận tốc tối thiểu mà bạn phải đi nếu muốn có mặt ở đó đúng lúc. Như vậy, việc xử lý các số liệu có một vai trò quan trọng trong quá trình ra quyết định, và ngôn ngữ lập trình NXT-G cũng có một số các khối lệnh xử lý dữ liệu cơ bản nhằm phục vụ mục đích này.

Các khối lệnh này bao gồm các khối lệnh: [Math], [Number to Text], [Text], [Logic], [Compare], và [Range]. Ba khối lệnh sau cùng liên quan đến các phép toán Logic sẽ được xét trong chương sau, vì chúng được sử dụng nhiều trong các cấu trúc điều khiển của chương trình. Bạn có thể tìm thấy các khối lệnh này trên thẻ [Complete] của Bảng các khối lệnh trong nhóm [Data] và [Advanced]



Hình 10.25: Các khối lệnh xử lý dữ liệu có thể tìm thấy trong nhóm [Data] và [Advanced] của thẻ [Complete] của Bảng các khối lệnh.

IV.1 Khối lệnh [Math]:



Hình 10.26: Khối lệnh [Math] và Bảng cấu hình

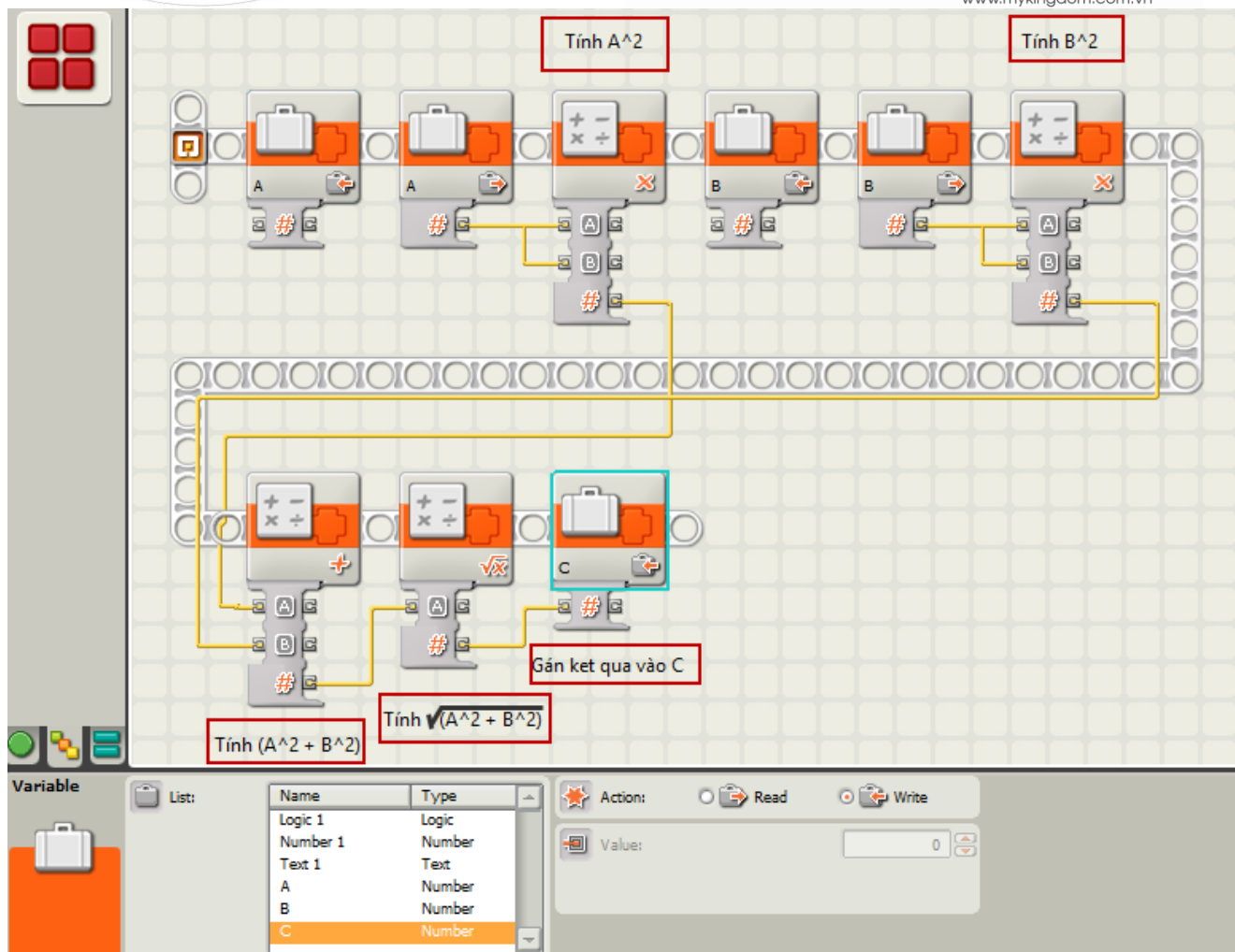
Khối lệnh này có thể thực hiện các phép tính (toán tử) đơn giản như Cộng (Addition), Trừ (Subtraction), Nhân (Multiplication), Chia (Division), Giá trị tuyệt đối (Absolute Value), Khai căn bậc 2 (Square Root). Để chọn phép toán cần dùng, bạn cần chọn đúng giá trị tương ứng cho thông số [Operation] trên Bảng cấu hình. Khối lệnh này còn có 2 thông số đầu vào A và B tương ứng với các toán hạng (đối tượng đầu vào mà toán tử hay phép tính sẽ thực hiện trên chúng). Tùy theo toán tử bạn chọn là toán tử một ngôi (tác động lên 1 toán hạng) hay toán tử hai ngôi (tác động lên 2 toán hạng), thì thông số A hay cả hai thông số A và B sẽ được dùng để tính ra kết quả. Sau đây là bảng tóm tắt mối liên hệ này.

Toán tử	Kết quả 
Cộng	$A + B$
Trừ	$A - B$
Nhân	$A \times B$
Chia	A/B
Giá trị tuyệt đối	$ A $
Khai căn bậc 2	$\sqrt{A}$ với $A \geq 0$ 0 với $A < 0$

Bảng : Các phép tính mà khối lệnh [Math] có thể thực hiện được.

Giá trị của các thông số đầu vào A, B (và kết quả đầu ra #) có thể các số nguyên hay thập phân, trong phạm vi .

Ví dụ sau đây là trích đoạn của chương trình tính chiều dài của cạnh huyền C, khi cho trước chiều dài của 2 cạnh góc vuông $A = 30$ cm và $B = 40$ cm. Theo định lý Pi-ta-go (Pythagoras) thì bình phương của chiều dài cạnh huyền bằng tổng bình phương của chiều dài hai cạnh góc vuông: $C^2 = A^2 + B^2$. Do đó ta có công thức tính chiều dài cạnh góc vuông như sau: $C = \sqrt{A^2 + B^2}$.



Hình 10.27: Trích đoạn chương trình tính chiều dài cạnh huyền. Các đường kết nối dữ liệu lúc này có màu vàng (khác với màu cam – truyền dữ liệu kiểu Text - trong ví dụ trước) vì dữ liệu mà ta đang dùng có kiểu là Number.

Ở đây chúng ta dùng 4 khối lệnh [Math] để thực hiện việc tính toán này. Hai khối lệnh [Math] đầu tiên dùng để tính bình phương của các cạnh góc vuông A và B. Lưu ý là cả 2 thông số đầu vào của mỗi khối lệnh này đều dùng chung một giá trị, đó là giá trị đọc ra từ khối lệnh [Variable] A hoặc B. Khối lệnh [Math] thứ 3 tính tổng của 2 phép toán vừa thực hiện, do đó các thông số đầu vào của nó được lấy từ kết quả đầu ra của 2 khối lệnh [Math] đầu. Khối lệnh [Math] thứ 4 làm phép tính khai căn bậc 2 của tổng này, phép toán khai căn bậc 2 là toán tử một ngôi, nên ta chỉ dùng một tham số đầu vào; dữ liệu được lấy từ kết quả đầu ra của khối lệnh [Math] thứ 3.

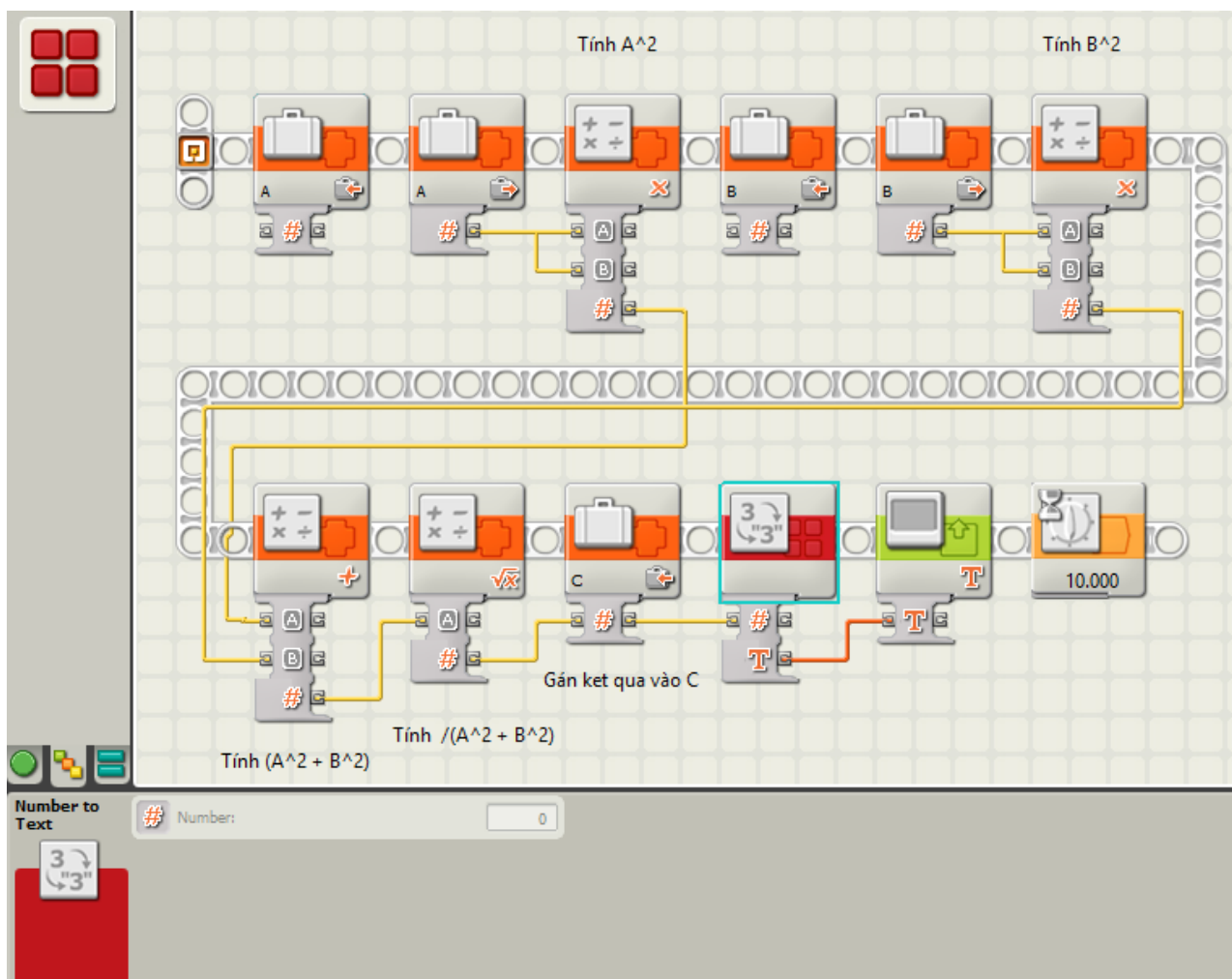
IV.2 Khối lệnh [Number to Text]:

Do khối lệnh [Display] không hiển thị được các dữ liệu số, nên nhiều khi ta cần phải chuyển đổi các số đó thành các chuỗi ký tự tương ứng để có thể hiển thị chúng trên màn hình). Ví dụ số 150 phải được chuyển đổi thành chuỗi các ký tự số "150" trước khi xuất ra màn hình. Ta dùng khối lệnh [Number to Text] cho mục đích này.



Hình 10.28 : Khối lệnh [Number to Text] và Bảng cấu hình của nó.

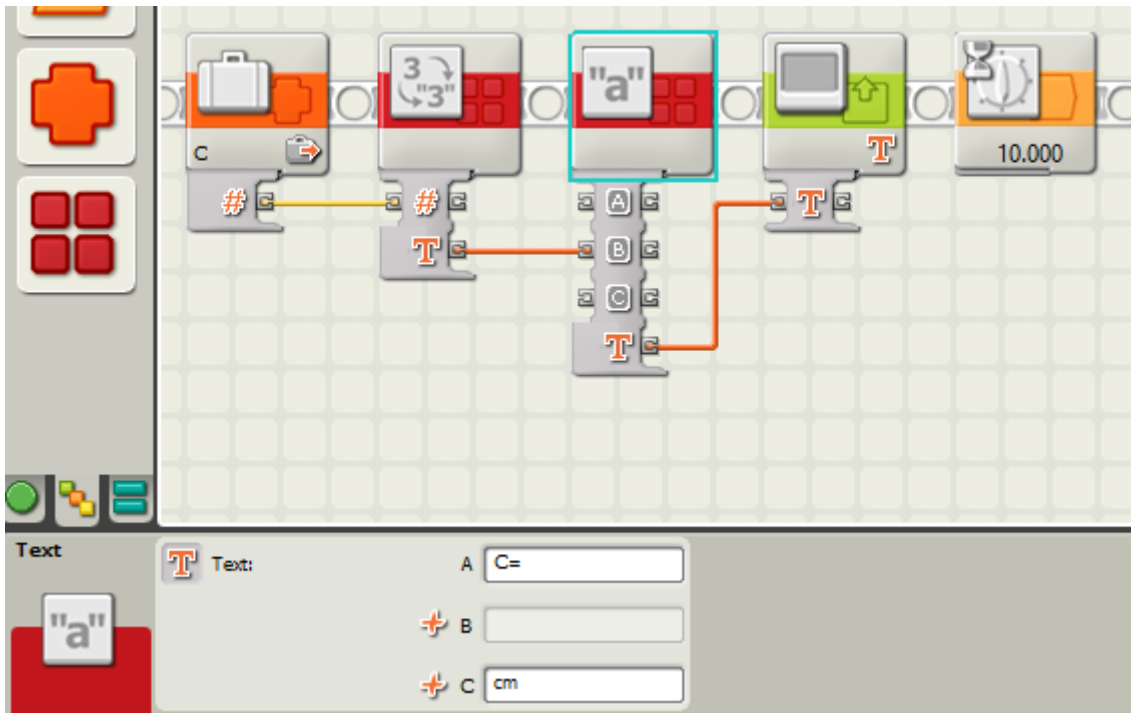
Bạn có thể nhập trực tiếp một giá trị số cho thông số [Number] (#) hay dùng kỹ thuật Kết nối dữ liệu để truyền một giá trị từ một khối lệnh khác vào thông số này. Sau đây là toàn bộ chương trình tính và hiển thị ra màn hình LCD chiều dài của cạnh huyền có dùng khối lệnh [Number to Text].



Hình 10.29 : Chương trình tính và hiển thị chiều dài của cạnh huyền dựa trên 2 cạnh góc vuông cho trước. Dữ liệu đi vào khối lệnh [Number to Text] có kiểu số, dữ liệu đầu ra của nó có kiểu chuỗi ký tự. Đường dữ liệu kiểu số có màu vàng, đường dữ liệu kiểu chuỗi ký tự có màu cam.

IV.3 Khối lệnh [Text]:

Khối lệnh [Text] được dùng để nối 2 hay nhiều chuỗi ký tự lại với nhau thành một chuỗi ký tự mới. Mỗi khối lệnh [Text] có 3 thông số đầu vào (A, B, C), chương trình sẽ kết nối lần lượt 3 chuỗi trong thông số A, B, C theo thứ tự để ra chuỗi kết quả (A+B+C). Nếu bạn không muốn dùng thông số nào thì cứ để trống. Trong ví dụ trước, thay vì chỉ hiển thị con số 50 trên màn hình bạn có thể nối các chuỗi “C =”, “50”, và “cm” lại với nhau để hiển thị thông tin đầy đủ và rõ ràng hơn.



Hình 10.30: Khối lệnh [Text] dùng để nối các chuỗi ký tự thành một chuỗi ký tự mới

IV.4 Khối lệnh [Logic]:

Ngôn ngữ NXT-G cho phép bạn thực hiện các phép toán lô-gic đơn giản trên các mệnh đề lô-gic (hay nói ngắn gọn là mệnh đề) bằng khối lệnh [Logic]. Trước khi đi vào sử dụng khối lệnh này, ta sẽ tìm hiểu xem mệnh đề là gì và một số phép toán lô-gic đơn giản hoạt động như thế nào.

Mệnh đề có thể hiểu là một khẳng định, nó có thể được phát biểu dưới dạng một câu hoặc một chuỗi các ký hiệu toán học. Ví dụ: “Việt Nam là một nước nằm ở Đông Nam châu Á” và “ $1 + 7 = 9$ ” là các mệnh đề lô-gic.

Mỗi mệnh đề đều có giá trị chân lý (tính đúng sai) của nó. Giá trị chân lý của mệnh đề là một trong 2 giá trị sau: Đúng (True/Yes/1) hoặc Sai (False/No/0). Trong ví dụ trên, mệnh đề “Việt Nam là một nước nằm ở Đông Nam châu Á” là mệnh đề Đúng, còn mệnh đề “ $1 + 7 = 9$ ” là mệnh đề Sai trong hệ số thập phân.

“Chân lý là cụ thể”. Tính đúng sai của một mệnh đề luôn gắn với một thời gian và địa điểm cụ thể. Nó có thể là đúng ở thời gian hoặc địa điểm này nhưng lại sai ở thời gian hoặc địa điểm khác. Nhưng tại một thời gian và địa điểm cụ thể thì mỗi mệnh đề chỉ có một giá trị chân lý (Đúng hoặc Sai). Ví dụ, “Tôi mệt”, bây giờ thì tôi mệt, nhưng sau khi nghỉ 30 phút thì tôi hết mệt. Nói cách khác, các phát biểu vừa Đúng lại vừa Sai; hoặc không Đúng cũng không Sai thì không phải là các mệnh đề lô-gic.

Sau đây là các bảng giá trị chân lý của một số phép toán lô-gic cơ bản.



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

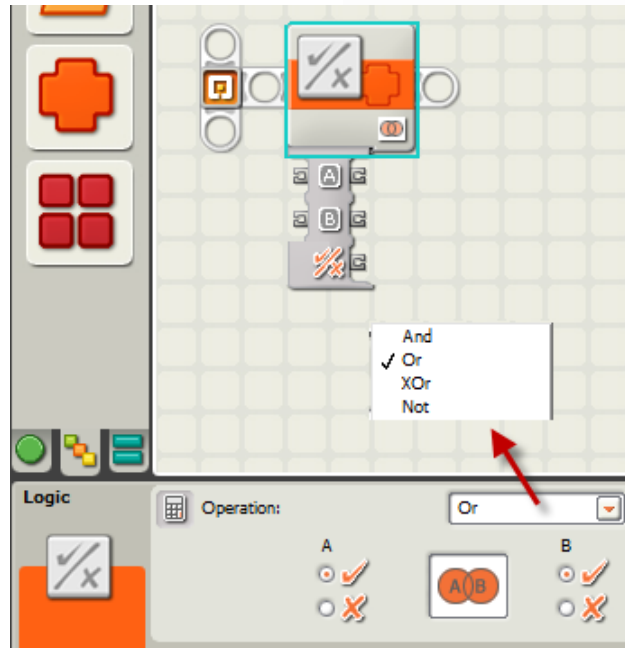
Phủ định (NOT) của một mệnh đề A là một mệnh đề, ký hiệu là $\bar{A}$, đúng khi A sai và sai khi A đúng.	
A	$\bar{A}$
1	0
0	1

Hội (AND) của hai mệnh đề A, B là một mệnh đề, đọc là “A và B”, ký hiệu là $A \wedge B$, đúng khi cả hai mệnh đề A, B cùng đúng và sai trong các trường hợp còn lại.		
A	B	$A \wedge B$
1	1	1
1	0	0
0	1	0
0	0	0

Tuyển (OR) của hai mệnh đề A, B là một mệnh đề, đọc là “A hoặc B”, ký hiệu là $A \vee B$, sai khi cả hai mệnh đề A, B cùng sai và đúng trong các trường hợp còn lại.		
A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

Tuyển có loại trừ (XOR) của hai mệnh đề A, B là một mệnh đề, đọc là “A XOR B”, ký hiệu là $A \oplus B$ hoặc $A \veebar B$, đúng khi hai mệnh đề A, B có giá trị chân lý khác nhau, và sai trong các trường hợp còn lại.		
A	B	$A \oplus B$
1	1	0
1	0	1
0	1	1
0	0	0

Trên Bảng cấu hình của khối lệnh [Logic], bạn có thể chọn các giá trị có sẵn (And, Or, XOr, và Not) cho thông số [Operation] để xác định phép toán lô-gic cần thực hiện. Khối lệnh [Logic] nhận vào các giá trị chân lý của các mệnh đề và trả về kết quả là một giá trị chân lý cụ thể của mệnh đề kết quả. Trên thẻ [Data Hub], các thông số đầu vào được chuyển cho 2 chốt dữ liệu vào A, B, còn kết quả đầu ra được xuất qua chốt dữ liệu ra [Result].



Hình 10.31: Khối lệnh [Logic] và Bảng cấu hình tương ứng.

Xét ví dụ bạn cần lập trình cho robot ngừng lại khi có người bấm vào nút [Enter] của khối vi điều khiển NXT hoặc khi robot đồng thời chạm vào chướng ngại vật phía trước nó và khoảng cách giữa đuôi robot và một chướng ngại vật đằng sau nó ngắn hơn 3cm.

Ở đây, ta có 3 mệnh đề con:

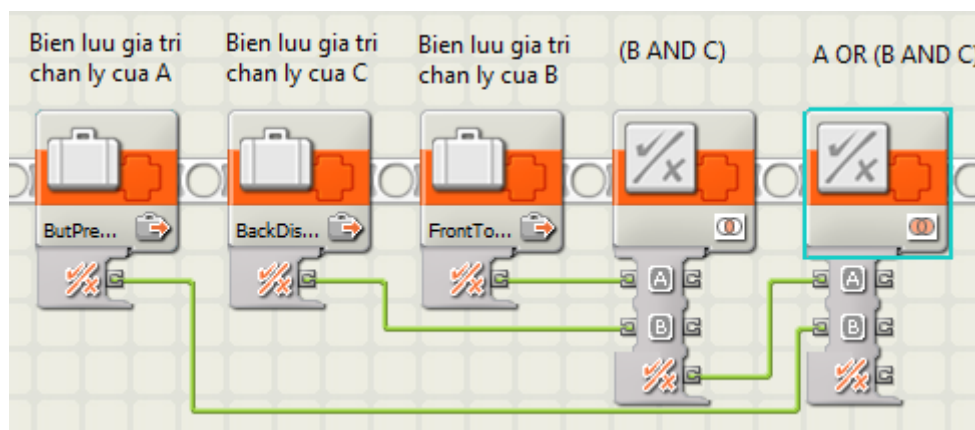
A="Có người bấm vào nút Enter của khối vi điều khiển NXT"

B="Robot chạm vào chướng ngại vật phía trước nó"

C="Khoảng cách giữa đuôi robot và một và một chướng ngại vật đằng sau nó ngắn hơn 3cm"

Như vậy điều kiện mà bạn cần ngừng robot là khi giá trị chân lý của mệnh đề $A \vee (B \wedge C)$ là Đúng.

Giả sử bạn đã có cách nào đó (dùng cảm biến chạm, cảm biến siêu âm, v.v...) kiểm tra được giá trị chân lý của từng mệnh đề A, B, C và lưu vào các biến số lô-gic tương ứng, thì bạn sẽ dễ dàng xác được giá trị chân lý của mệnh đề $A \vee (B \wedge C)$ bằng cách dùng 2 khối lệnh [Logic] như sau.



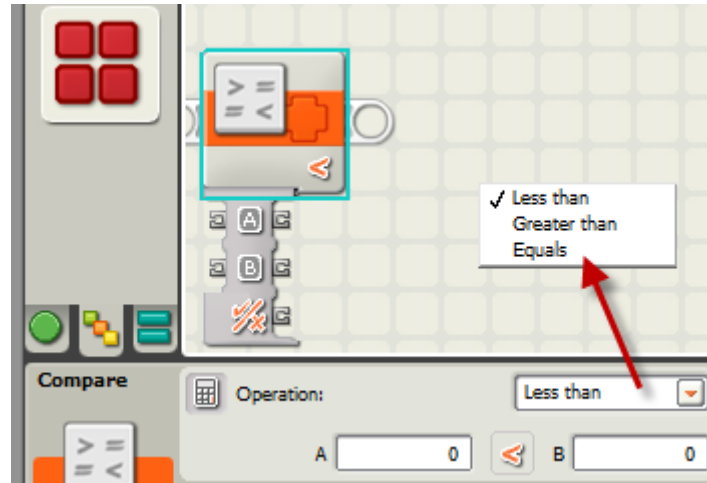
Hình 10.32: Phần chương trình tính toán điều kiện để ngừng robot. Lưu ý các đường kết nối dữ liệu Logic có màu xanh lá cây.



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

IV.5 Khối lệnh [Compare]:

Khối lệnh này được dùng để kiểm tra xem một số này có nhỏ hơn (Less than), lớn hơn (Greater than), hay bằng (Equals) một số khác. Bạn có thể chọn cách so sánh bằng việc chọn giá trị phù hợp cho thông số [Operation]. Kết quả trả về là một giá trị (Đúng/Sai) của kiểu dữ liệu Logic.

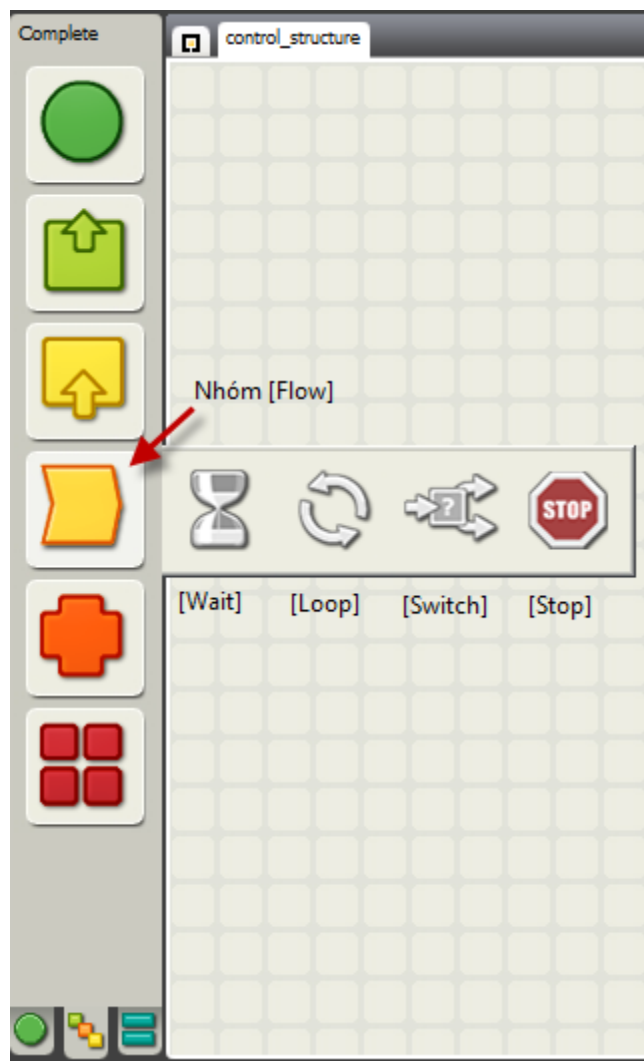


Hình 10.33: Khối lệnh [Compare] và Bảng cấu hình của nó.

IV. 6 Khối lệnh [Range]: dùng để kiểm tra xem một số có thuộc một phạm vi nào đó hay không.

CHƯƠNG 11 : CÁC CẤU TRÚC ĐIỀU KHIỂN

Ngôn ngữ NXT-G cho phép bạn điều khiển việc chọn lựa và thực hiện các đoạn lệnh khác nhau tùy theo nhu cầu thực tế. Ví dụ, khi một điều kiện cụ thể là Đúng thì chương trình sẽ thực hiện đoạn lệnh này, nếu ngược lại thì thực hiện đoạn lệnh kia. Hay khi đã thực hiện xong một đoạn lệnh thì thay vì chuyển sang thực hiện các khối lệnh kế tiếp, chương trình tiếp tục quay lại phần đầu của đoạn lệnh đó và tiếp tục thực hiện nó thêm một số lần khác nữa nữa. Hay thay vì thực thi tiếp các khối lệnh thì chương trình tạm ngưng cho đến khi một điều kiện cho trước xảy ra, v.v... Bạn có thể tìm thấy các khối lệnh dùng làm cấu trúc điều khiển trong nhóm [Flow] của thẻ [Complete] của Bảng các khối lệnh.

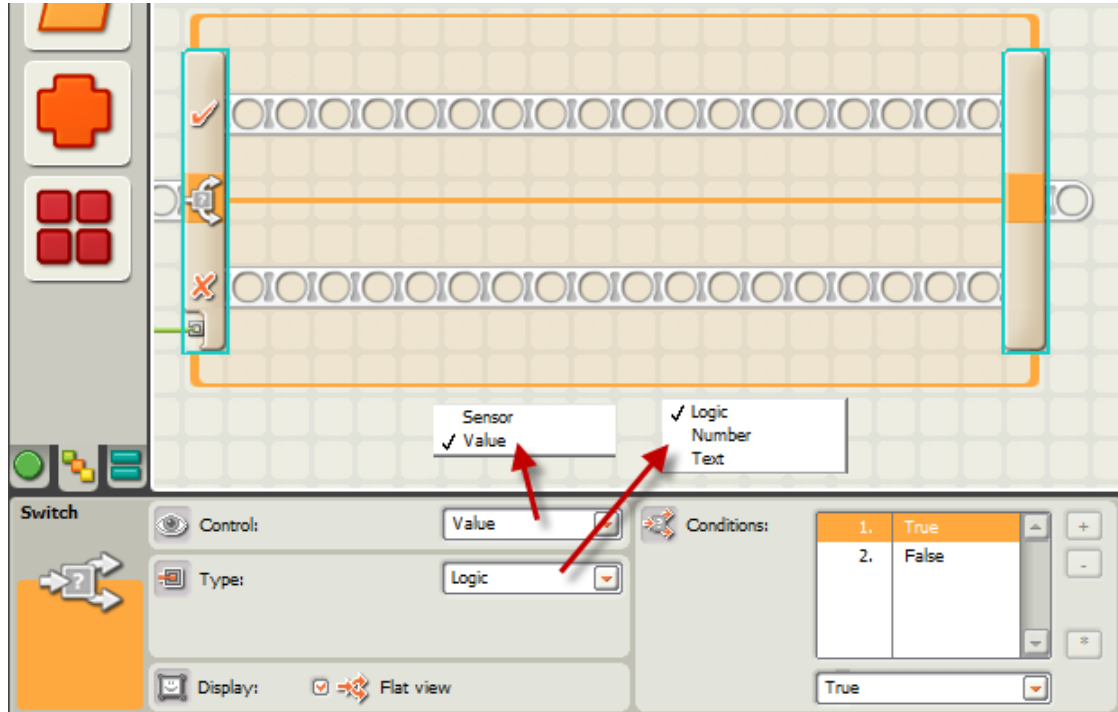


Hình 11.1: Các khối lệnh dùng cho các cấu trúc điều khiển.

I. Khối lệnh [Switch]:

Bạn có thể dùng khối lệnh này để chọn và thực hiện một trong các chuỗi lệnh khác nhau khi một điều kiện nào đó được thỏa mãn (quá trình rẽ nhánh dòng lệnh). Có hai cách dùng khối lệnh [Switch], đó là

điều khiển bằng cảm biến (Sensor) hoặc bằng giá trị (Value). Bạn có thể thiết lập cách điều khiển qua việc chọn lựa giá trị phù hợp cho thông số [Control]. Trong phần này ta sẽ xét cách điều khiển bằng giá trị.



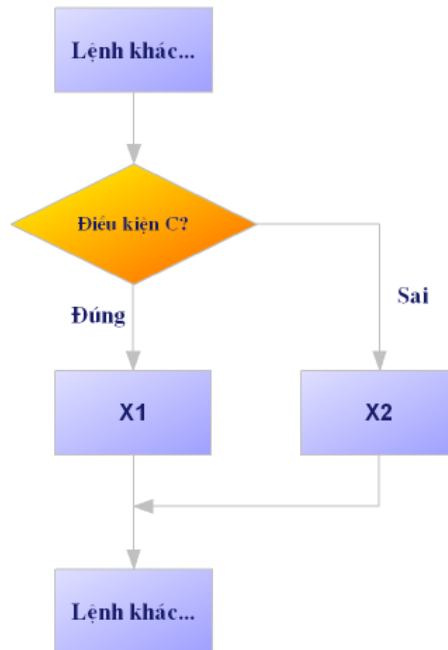
Hình 11.2: Bạn có 2 cách để điều khiển việc chọn lựa và thực hiện các dòng lệnh cho khối lệnh [Switch], bằng cảm biến hoặc giá trị.

Đối với việc điều khiển quá trình rẽ nhánh dòng lệnh bằng giá trị thì tùy theo kiểu dữ liệu (thông số Type) mà ta có số lượng nhánh lệnh tối đa có thể rẽ. Nếu kiểu dữ liệu là Logic, thì các giá trị có thể có là Đúng và Sai, nên ta chỉ có tối đa 2 nhánh lệnh. Còn nếu kiểu dữ liệu là Number hoặc Text thì các số lượng các giá trị có thể có hầu như là vô tận, nên về mặt lý thuyết thì số lượng nhánh có thể điều khiển là vô tận. Tuy nhiên, trên thực tế số lượng nhánh có thể rẽ của một khối lệnh [Switch] thường nhỏ vì ta cần phải viết chương trình sao cho dễ đọc và dễ hiểu nữa.

Có ba cấu trúc điều khiển chính mà khối lệnh [Switch] có thể thực hiện:

I.1 **NEẾU** <điều_kiện_C> Đúng **THÌ** <thực_hiện_đoạn_lệnh_X₁>

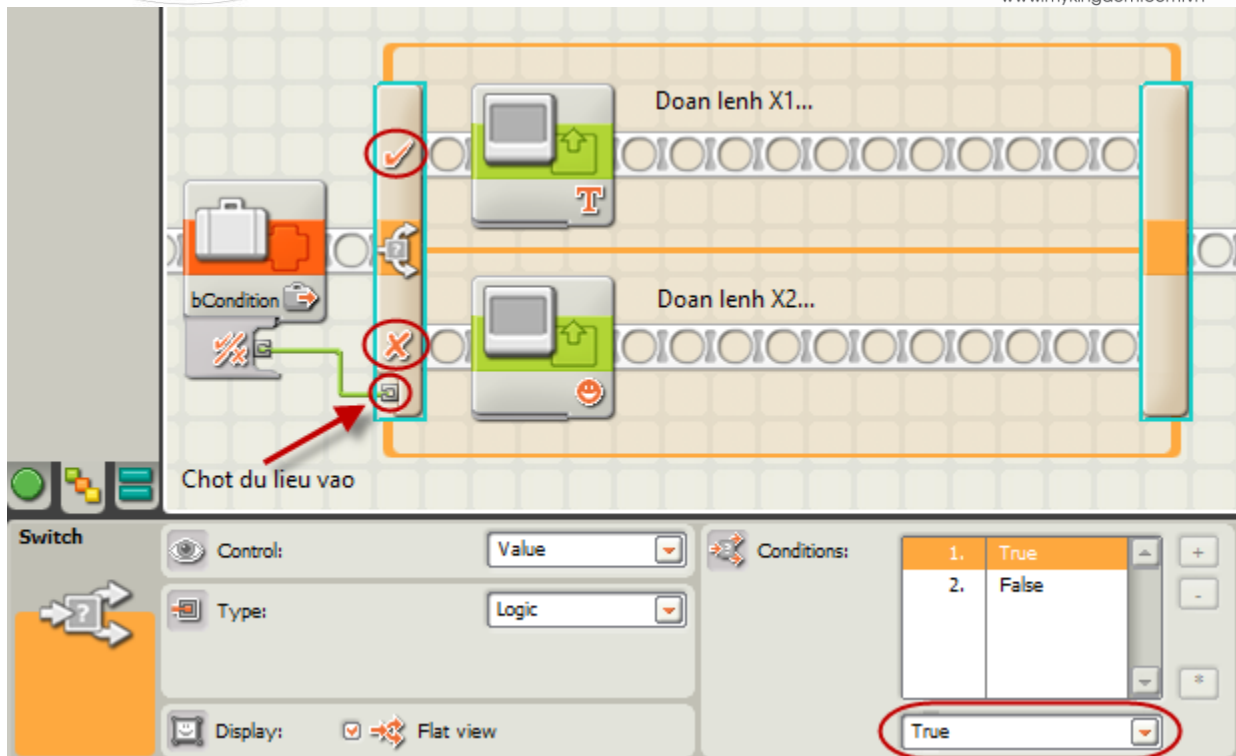
NGƯỚC LẠI THÌ <thực_hiện_đoạn_lệnh_X₂>



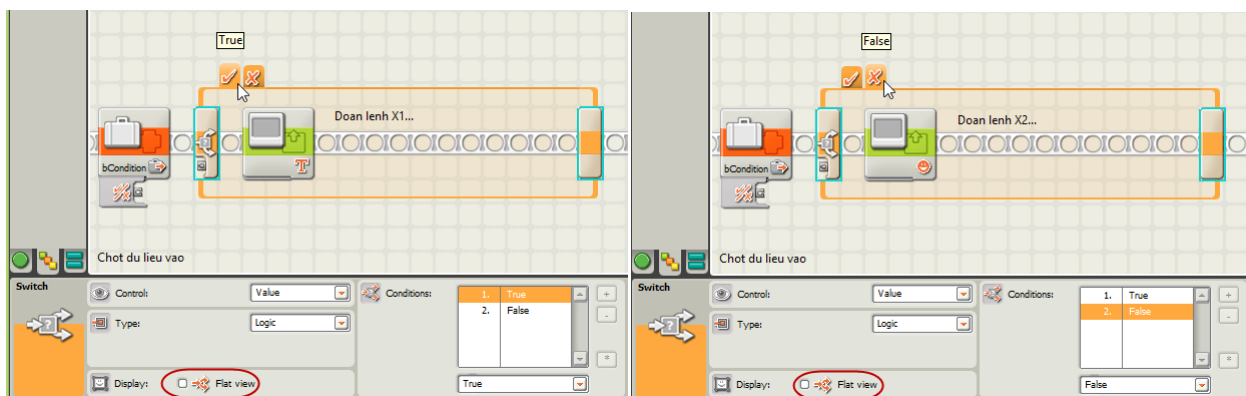
Hình 11.3 : Chương trình sẽ kiểm tra điều kiện C, nếu Đúng thì thực hiện lệnh X_1 , còn nếu Sai thì thực hiện lệnh X_2 . Sau đó, chương trình đi ra khỏi cấu trúc điều khiển này và tiếp tục thực hiện các phần còn lại.

Hình <11.3> là ví dụ mô tả cách dùng khối lệnh [Switch] để điều khiển việc rẽ nhánh theo cấu trúc này.

Ở đây, giá trị chân lý của mệnh đề điều kiện C được truyền vào khối lệnh [Switch] thông qua chốt dữ liệu vào. Bạn có thể chọn “Flat view” của thông số [Display] để hiển thị đầy đủ cả 2 nhánh lệnh đồng thời trên màn hình (Việc lựa chọn này thực ra cũng chỉ có thể thực hiện khi ta chỉ có 2 nhánh lệnh, nếu có nhiều hơn thì chương trình Lego MindStorms NXT sẽ hiển thị các nhánh theo từng thẻ Nhánh lệnh). Nhánh lệnh phía trên (có đánh dấu ) sẽ được thực thi khi điều kiện C là Đúng, nhánh lệnh phía dưới (có đánh dấu ) sẽ được thực thi khi điều kiện C là Sai. Thông số [Conditions] là một bảng gồm 2 hàng biểu thị cho 2 nhánh lệnh và điều kiện thực thi của mỗi nhánh lệnh: (1, True) có nghĩa là nhánh lệnh thứ nhất, phía trên, được thực thi khi điều kiện C Đúng; và (2, False) có nghĩa là nhánh lệnh thứ hai, phía dưới, được thực thi khi điều kiện C Sai. Bạn có thể dùng Hộp danh sách thả xuống (ở phía dưới bên phải của Bảng cấu hình) để thay đổi điều kiện thực hiện các nhánh lệnh (Đúng -> Sai hay Sai -> Đúng).



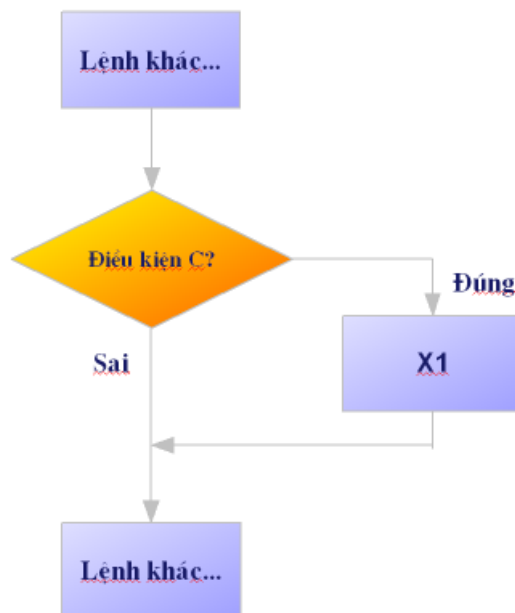
Hình 11.4: Cấu trúc điều khiển với 2 nhánh lệnh được hiển thị đồng thời (Flat view).



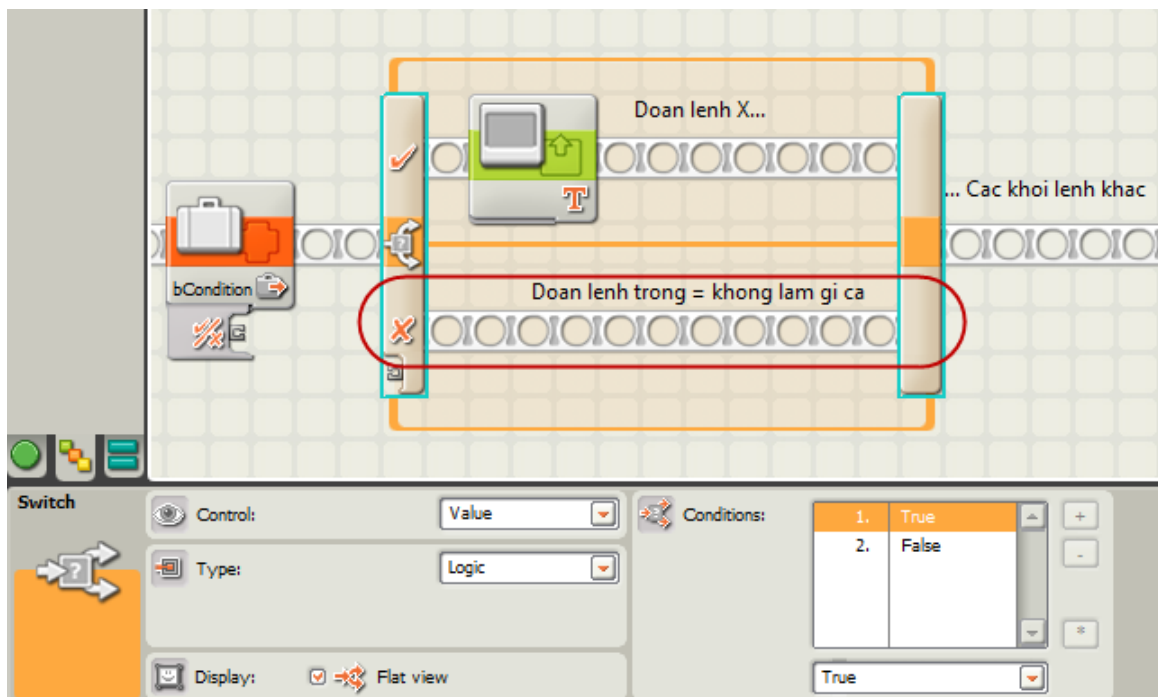
Hình 11.5 : Cấu trúc điều khiển với 2 nhánh lệnh được hiển thị lần lượt theo từng thẻ tương ứng (bỏ không chọn “Flat view” của thông số [Display]). Bạn chọn thẻ [True] hoặc [False] để xem các nhánh lệnh tương ứng

1.2 NẾU <điều_kiện_C> Đúng **THÌ** <thực_hiện_đoạn_lệnh_X>

Cấu trúc điều khiển này thực ra chính là cấu trúc điều khiển vừa giới thiệu ở trên với đoạn lệnh X₂ để trống. Tức là chương trình không làm gì cả khi điều kiện C là Sai.



Hình11.6 : Chương trình kiểm tra điều kiện C, nếu Đúng thì thực hiện lệnh X_1 , nếu Sai thì không làm gì cả. Sau đó, chương trình đi ra khỏi cấu trúc điều khiển và tiếp tục thực hiện các phần còn lại.



Hình 11.7: Mô phỏng cấu trúc rẽ nhánh 1 nhánh lệnh bằng khối lệnh [Switch]

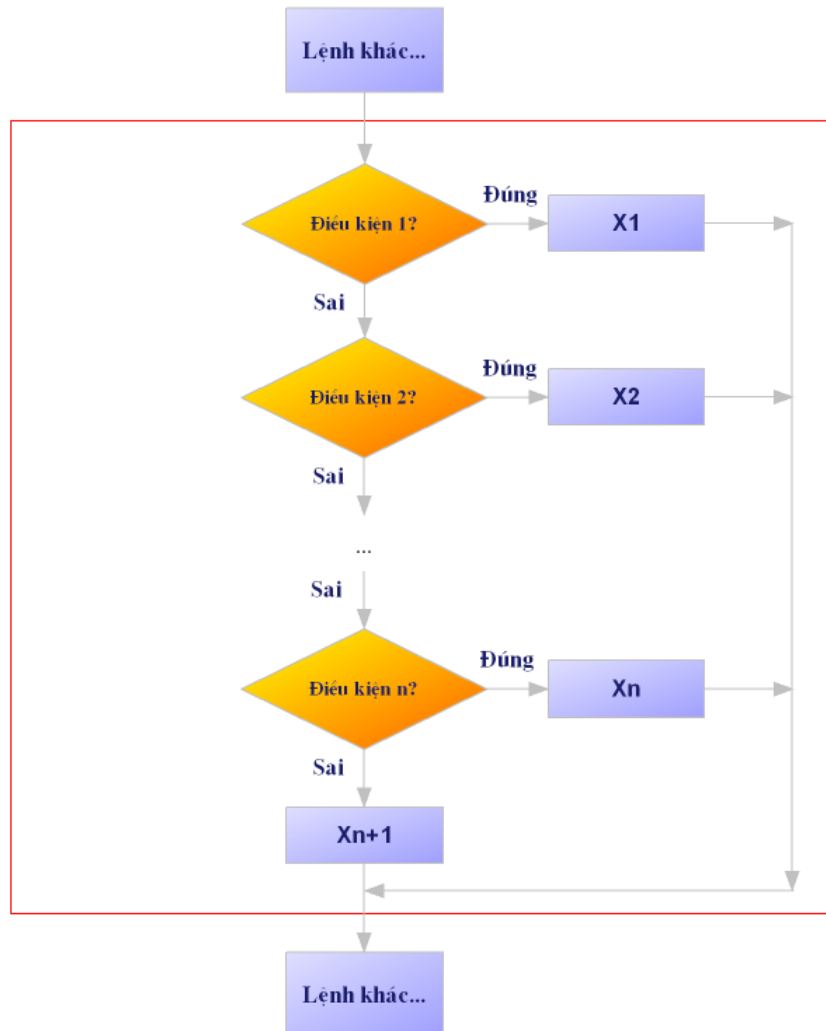
I.3<Điều_kiện_1> Đúng: <thực_hiện_đoạn_lệnh_ X_1 >

<Điều_kiện_2> Đúng: <thực_hiện_đoạn_lệnh_ X_2 >

...

<Điều_kiện_N> Đúng: <thực_hiện_đoạn_lệnh_ X_N >

<Mặc_định>: <thực_hiện_đoạn_lệnh_ X_{N+1} >



Hình 11.4: Cấu trúc rẽ N nhánh này sẽ được chương trình thực hiện như sau. Đầu tiên, chương trình kiểm tra Điều kiện 1, nếu Đúng thì thực hiện lệnh X_1 rồi ra khỏi cấu trúc điều khiển này, nếu Sai thì tiếp tục kiểm tra Điều kiện 2 và xử lý một cách tương tự. Nếu chương trình sau khi kiểm tra đến Điều kiện N mà vẫn Sai thì lúc này lệnh mặc định X_{n+1} (lệnh dự phòng, nếu trước đó không có lệnh nào được chọn, thì lệnh dự phòng này sẽ được chọn) sẽ được thực hiện trước khi ta ra khỏi cấu trúc điều khiển này.

Thông thường cấu trúc rẽ nhánh này được dùng với các điều kiện liên quan đến số nguyên.

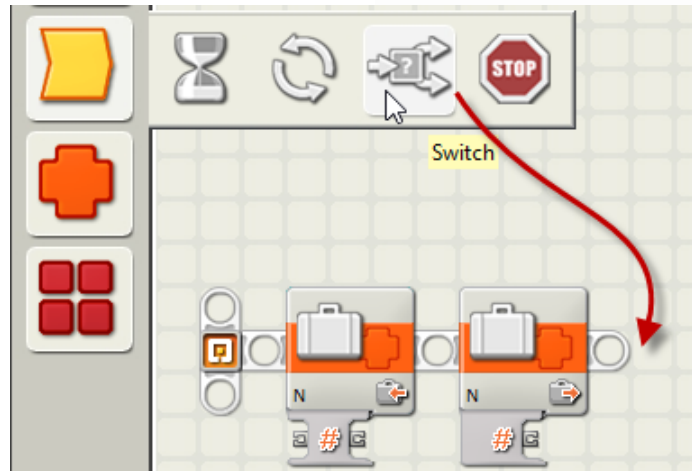
Ví dụ, đoạn chương trình của bạn cần cần kiểm tra một biến số N và thực hiện các lệnh như sau:

N=3, in ra màn hình dòng chữ “So Ba”

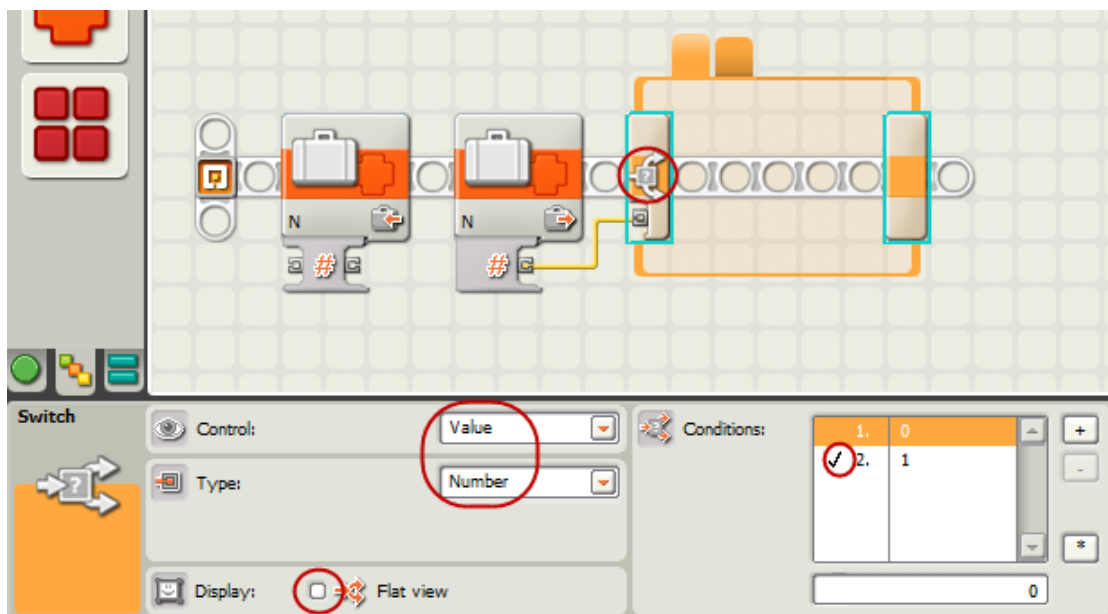
N=4, in ra màn hình dòng chữ “So Bon”

Các trường hợp khác: in ra màn hình dòng chữ “Khong biet”

Sau đây là các bước lập trình.



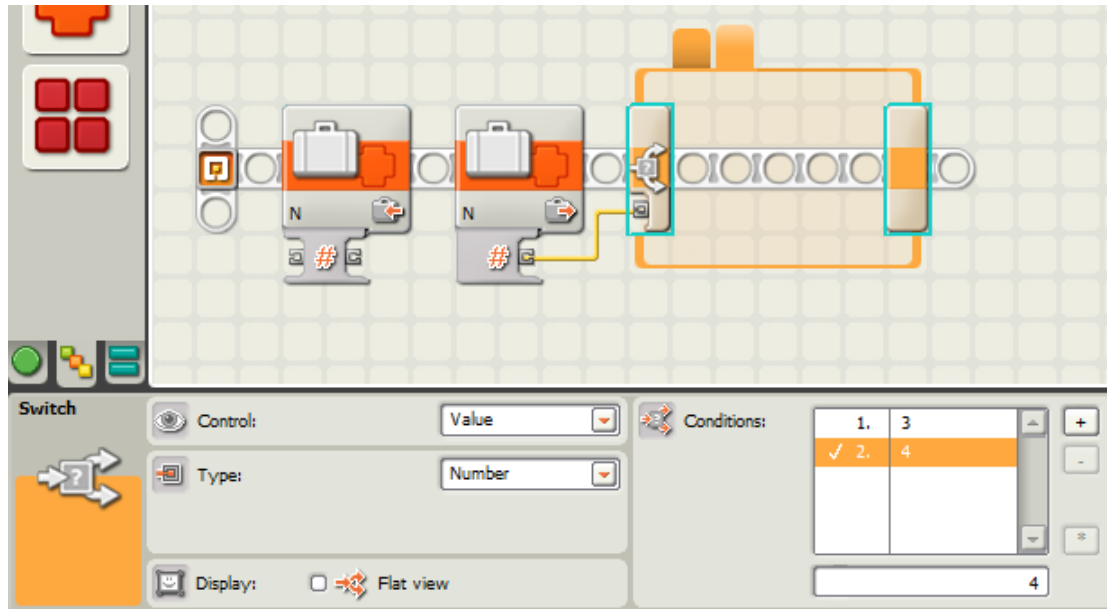
Hình 11.5: Kéo và thả khối lệnh [Switch] vào chương trình ngay sau khối [Variable] của biến số N.



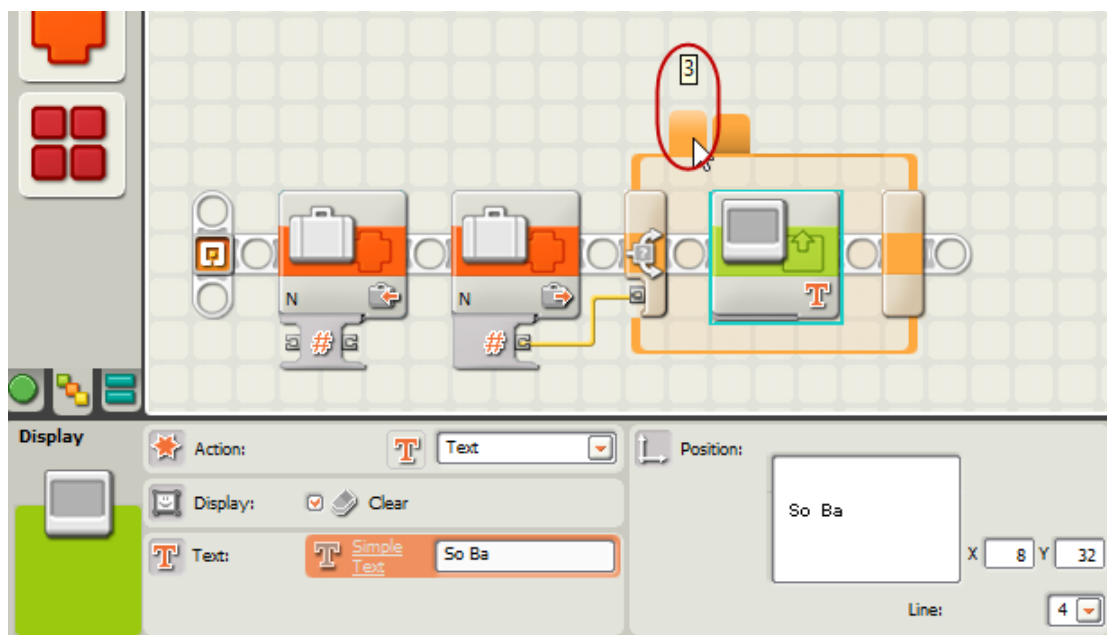
Hình 11.6: Bấm vào phần đầu của khối lệnh [Switch] để mở Bảng cấu hình của nó. Chọn “Value” cho thông số [Control], “Number” cho thông số [Type], và bỏ chọn “Flat view” của thông số [Display]. Sau đó kết nối dữ liệu từ chốt dữ liệu ra của khối [Variable] đến chốt dữ liệu vào của khối [Switch]. Lúc này ta có một khối lệnh [Switch] được hiển thị theo dạng thẻ. Khối lệnh này đang so sánh giá trị của biến số N với các giá trị có sẵn 0 và 1 trong bảng thông số [Condition].



Hình 11.7: Chọn hàng đầu tiên của bảng thông số [Condition], dùng ô nhập dữ liệu ở ngay bên dưới bảng để đổi số 0 thành 3. Lưu ý là sau khi bạn đổi xong (bấm phím Enter hay bấm chuột vào bất kỳ một nơi khác trên màn hình), thì hàng dữ liệu này sẽ chuyển xuống vị trí thứ 2, do bảng dữ liệu này được sắp xếp theo thứ tự của dữ liệu.



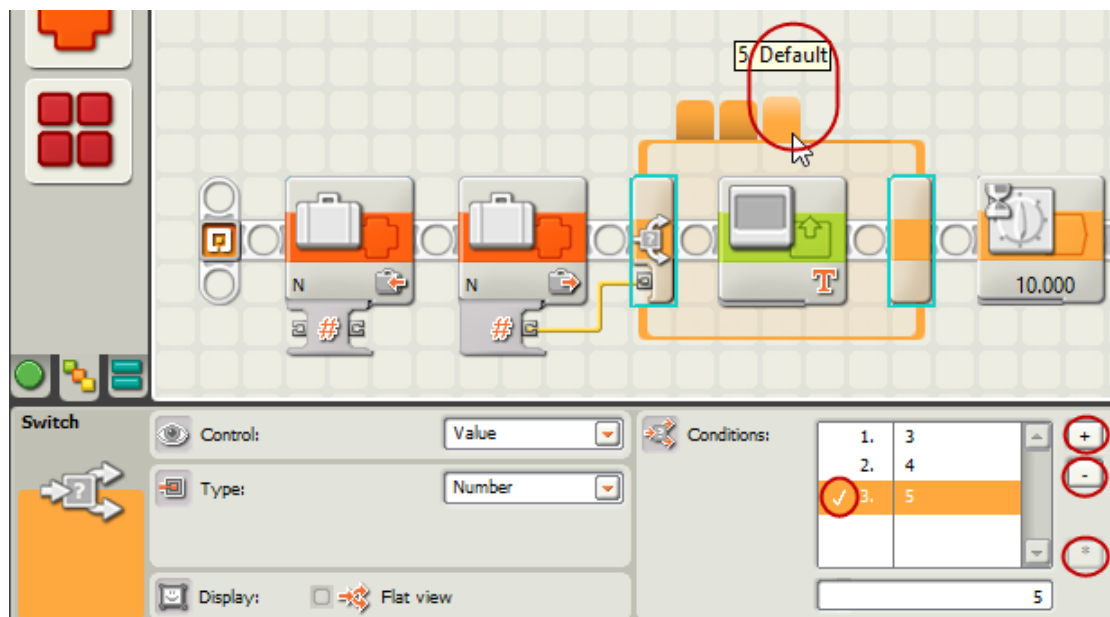
Hình 11.8 : Tiếp tục đổi hàng dữ liệu mang giá trị 1 thành 4. Lúc này ta có 2 nhánh lệnh tương ứng với hai điều kiện N=3, N=4.



Hình 11.9: Chọn thẻ tương ứng với điều kiện N=3 và đưa vào nhánh lệnh này khối lệnh [Display] để hiển thị dòng chữ “So Ba”. Sau đó chọn thẻ tương ứng với điều kiện N=4 và thiết lập khối lệnh [Display] để hiển thị dòng chữ “So Bon”.

Lúc này, bạn có thể chạy thử chương trình. Cho $N=3$ hay $N=4$, chương trình sẽ in dòng chữ tương ứng lên màn hình. Bạn thử với $N=5$ hay gán N bằng một giá trị bất kỳ khác 3 và 4, thì chương trình in ra dòng chữ “So Bon”. Chuyện gì xảy ra vậy? Nếu bạn để ý, bạn sẽ nhận thấy trên bảng dữ liệu của thông số [Condition] của khối lệnh [Switch] có một dấu kiểm bên cạnh hàng dữ liệu có giá trị 4 (Xem hình ?). Điều này có nghĩa là lệnh được chọn tương ứng với điều kiện $N=4$, chính là lệnh mặc định. Tức là nếu N khác 3 lẫn 4 thì lệnh in ra dòng chữ “So Bon” sẽ được thực hiện. Để sửa lại chương trình cho cho đúng với yêu cầu, bạn cần phải thêm vào một nhánh lệnh mặc định.

Bấm vào nút (+) trên màn hình của thông số [Condition] để thêm vào một hàng mới, chấp nhận giá trị 5 mà chương trình Lego MindStorm NXT đã chọn. Sau đó bấm vào nút (*) để xác định đây là hàng mặc định. Tức là nếu $N=5$ hoặc là một giá trị nào đó khác 3 và 4 thì chương trình sẽ xem như giá trị của N thỏa điều kiện mặc định. Lúc này ta cũng có một thẻ mới tương ứng với điều kiện mặc định ($N=5$). Thêm khối lệnh [Display] vào nhánh lệnh mới này để nó có thể hiển thị dòng chữ “Khong Biet” ra màn hình LCD. Nhớ thêm vào khối lệnh [Wait Time] với giá trị thời gian phù hợp để bạn có thể kịp thấy những gì hiện ra trên màn hình LCD.



Hình 11.10: Đoạn chương trình với cấu trúc điều khiển rẽ nhánh có 3 nhánh lệnh. Để thêm vào nhánh lệnh mới bạn bấm nút (+), để xóa bớt một nhánh thì bạn chọn nhánh đó rồi bấm nút (-). Để xác định một nhánh lệnh là mặc định, bạn cũng chọn nhánh đó rồi bấm nút (*).

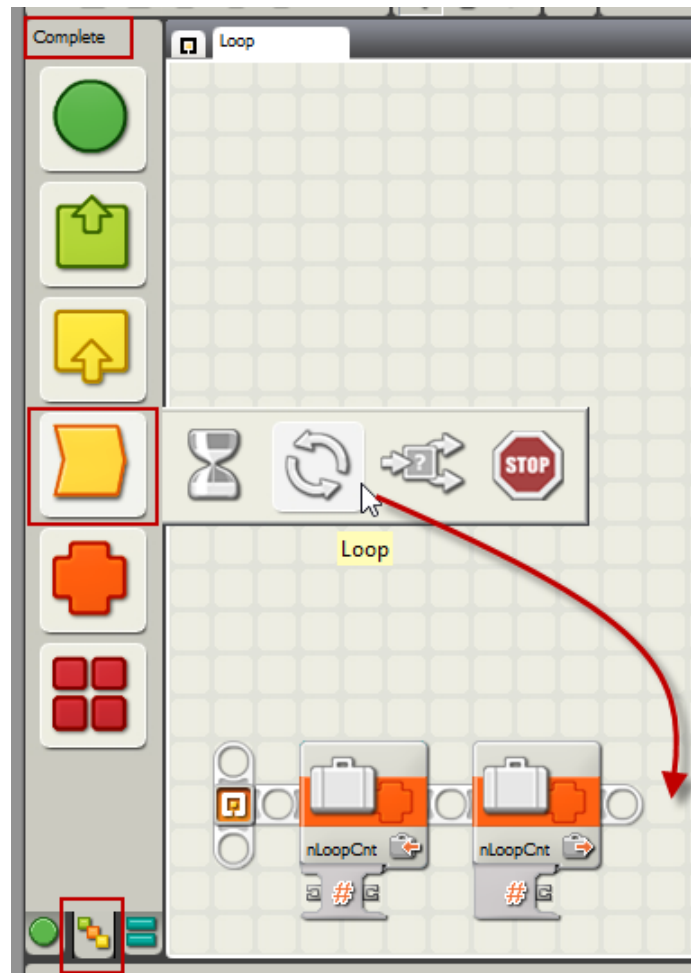
Tới đây bạn có thể thay đổi giá trị của biến số N và chạy thử chương trình để thấy các dòng thông báo phù hợp hiện ra trên màn hình.

II. Khối lệnh [Loop]:

Ngoài cấu trúc điều khiển rẽ nhánh, người ta còn dùng đến cấu trúc điều khiển vòng lặp (gọi ngắn gọn là cấu trúc vòng lặp hay vòng lặp). Vòng lặp giúp ta thực hiện một nhóm lệnh lặp đi lặp lại một số lần (hữu hạn hay vô hạn) cho trước, hoặc cho đến khi một điều kiện nào đó xảy ra (ví dụ, thời gian cho phép lặp đã hết, một tín hiệu nào đó của cảm biến đưa về, hay một điều kiện lô-gic đã định nào đó). Trong phần này, chúng ta sẽ tìm hiểu một số cách dùng khối lệnh [Loop] thông thường để điều khiển cho chương trình thực

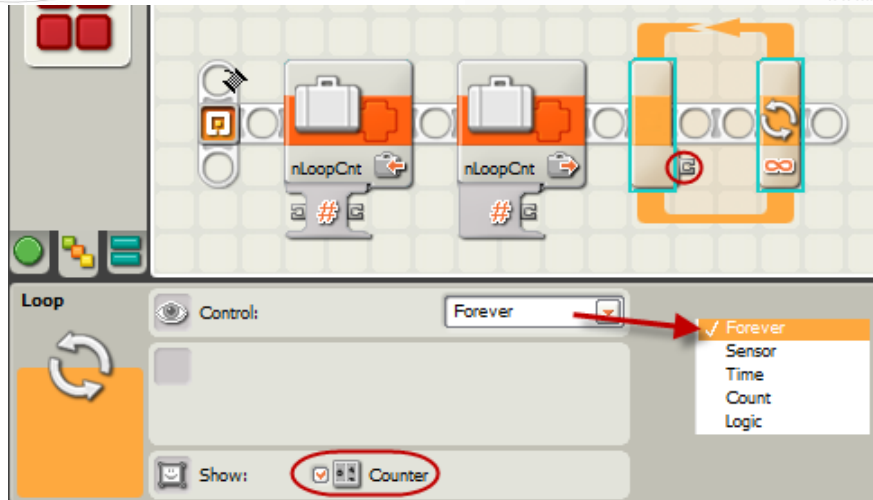
hiện các vòng lặp. Riêng phần sử dụng khối lệnh [Loop] có kết hợp trực tiếp với các cảm biến sẽ được giới thiệu trong các phần sau, khi mà chúng ta giới thiệu qua các cảm biến có trong bộ lắp ráp Lego MindStorms.

Để đưa khối lệnh [Loop] vào chương trình, bạn kéo biểu tượng của nó từ nhóm [Flow] thuộc thẻ [Complete] của Bảng các khối lệnh và thả nó vào vị trí cần thiết của chương trình.



Hình 11.11: Kéo thả khối lệnh [Loop] vào chương trình.

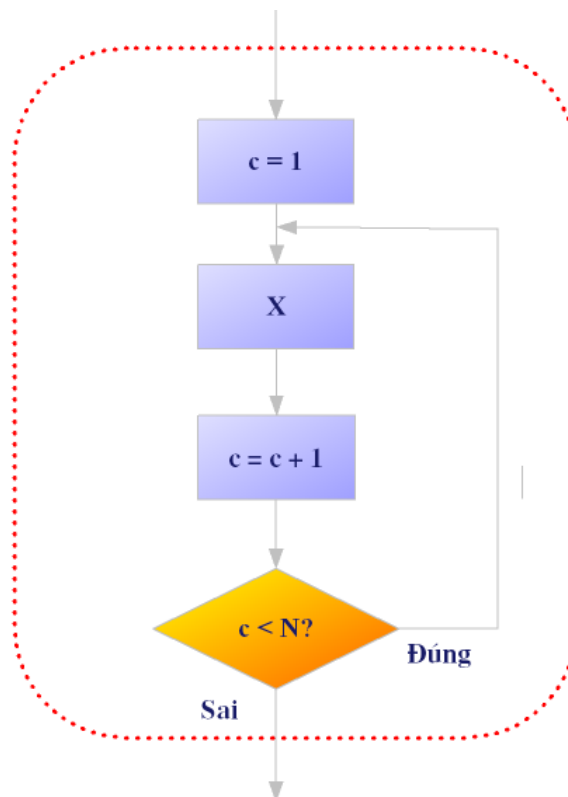
Trên bảng Cấu hình của khối này, thông số [Control] được dùng để xác định cách thức mà chương trình tính toán để kết thúc vòng lặp (đi ra khỏi cấu trúc vòng lặp). Bạn có thể chọn một trong các giá trị sau cho thông số này: [Forever]: Lặp mãi mãi, [Sensor]: lấy dữ liệu từ cảm biến làm điều kiện ngừng vòng lặp, [Time]: ngừng vòng lặp sau một khoảng thời gian cho trước, [Count]: ngừng vòng lặp khi bộ đếm số lần lặp đạt đến một giá trị nào đó, [Logic]: ngừng vòng lặp khi biến số Lô-gic đang xét có một giá trị xác định.



Hình 11.12: Các giá trị có thể dùng cho thông số [Control] của khối lệnh [Loop].

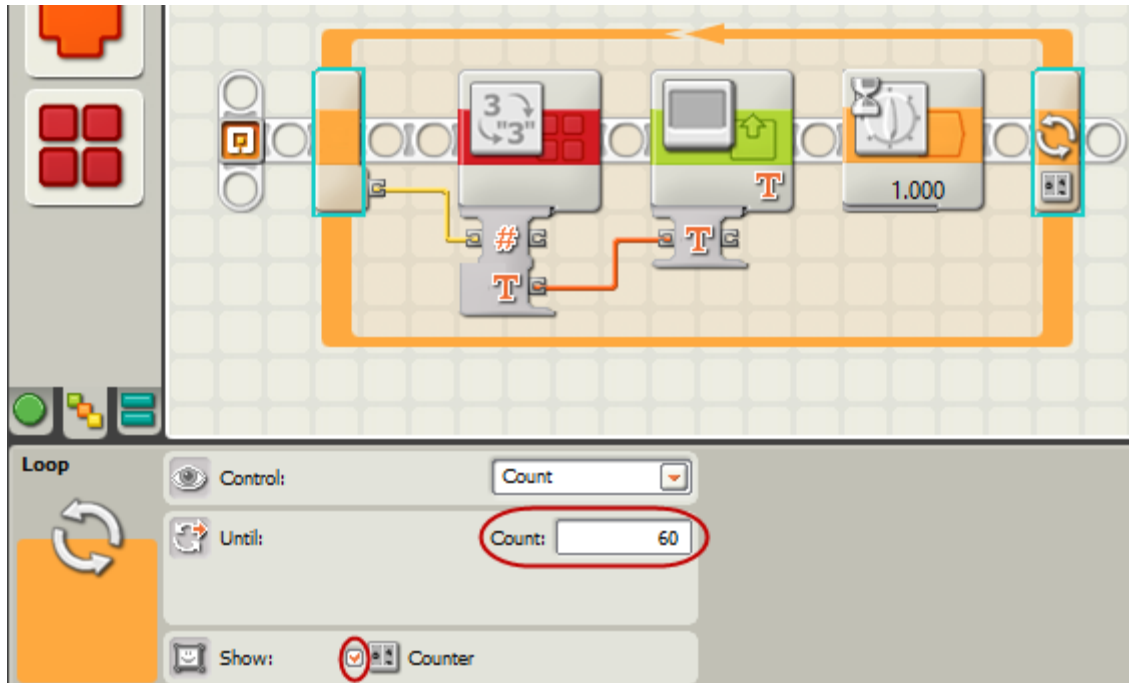
Sau đây ta sẽ xét 3 dạng vòng lặp thường gặp.

II.1 Thực hiện (nhóm) lệnh X N lần:



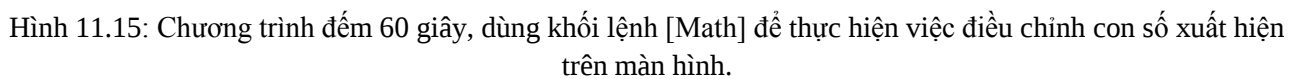
Hình 11.13: Lệnh X được thực hiện N lần.

Đây là một dạng vòng lặp khá đơn giản. Chương trình duy trì một biến số ẩn (bạn không cần khai báo nó) để đếm số lượng vòng lặp đã được thực hiện. Khi biến số này vượt ngưỡng thì chương trình thoát khỏi cấu trúc vòng lặp. Sau đây là một ví dụ về một đoạn chương trình thực hiện việc đếm giây. Cứ mỗi giây thì chương trình in ra màn hình con số tương ứng. Sau khi được 60 giây thì chương trình chấm dứt.



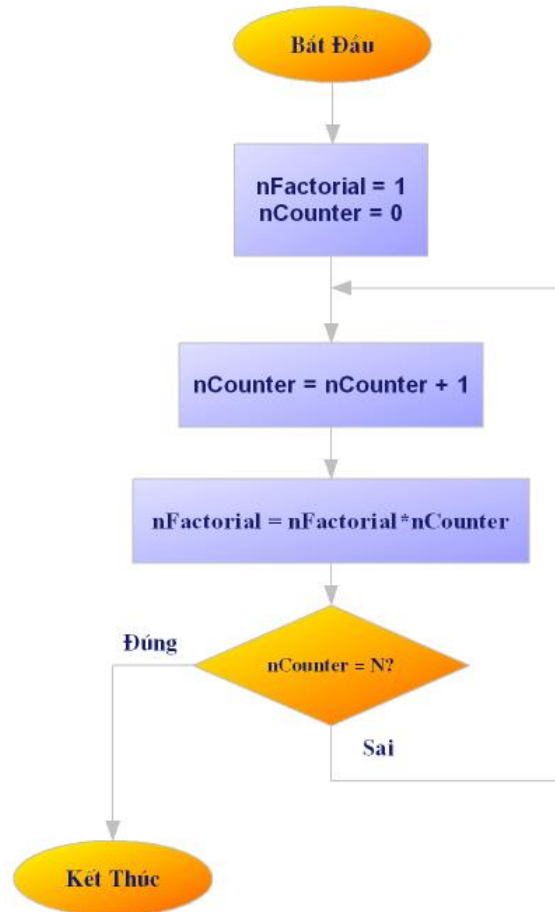
Hình 11.14: Chương trình đếm 60 giây. Khối lệnh [Number to Text] chuyển đổi dữ liệu số lần lặp thành chuỗi ký tự tương ứng để khối lệnh [Display] in ra màn hình. Khối lệnh [Wait Time] đợi đúng 1 giây trước khi thực hiện lần lặp khác.

Đoạn chương trình trên tuy đếm đủ 60 giây, nhưng nó lại in ra màn hình các số từ 0 đến 59, nếu bạn muốn sửa lại chương trình để nó in ra các số từ 1 đến 60 cho có vẻ tự nhiên hơn thì bạn có thể cộng 1 vào số lần lặp rồi mới in ra màn hình. Sau đây là đoạn chương trình mới.

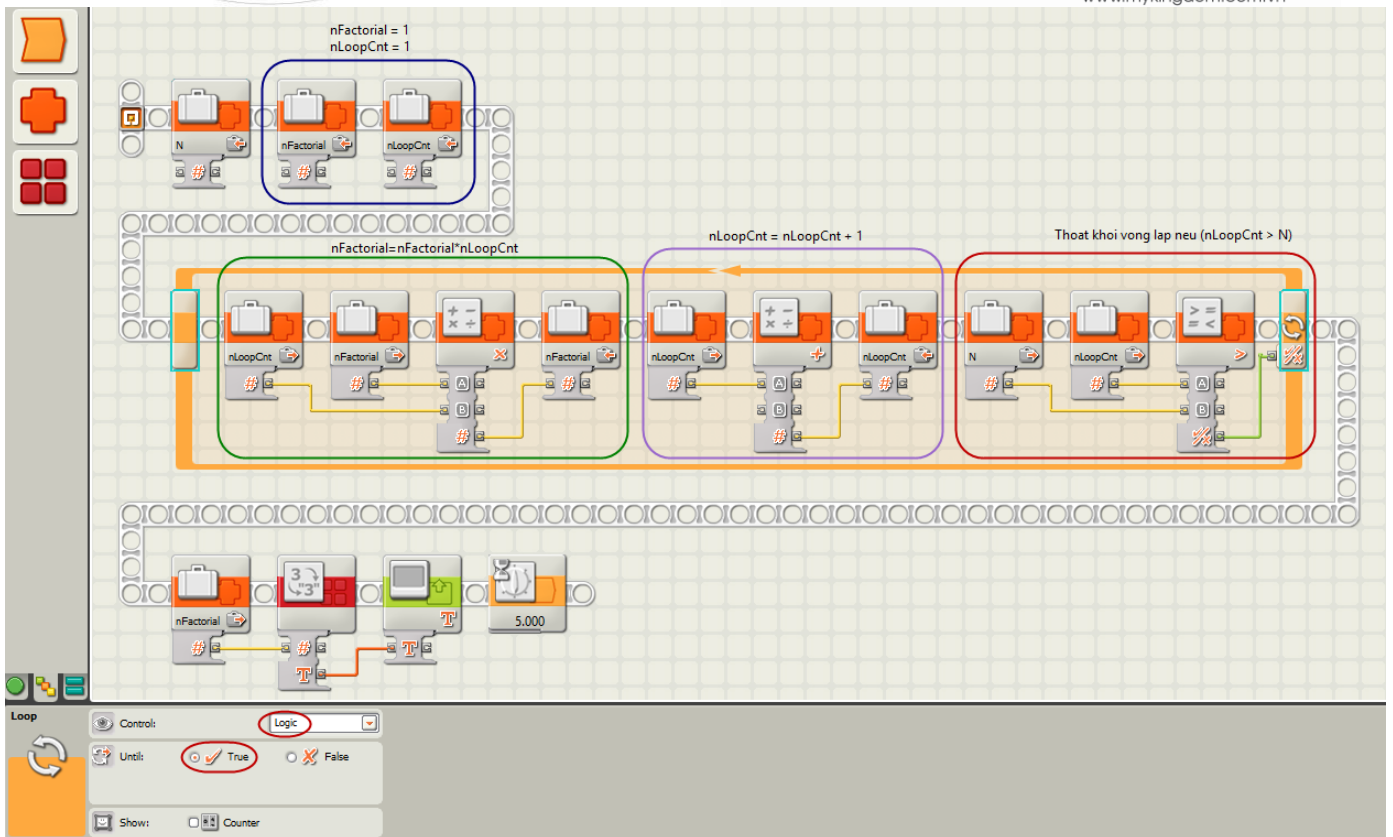


```
graph TD; Entry(( )) --> X[X]; X --> C{Điều kiện C?}; C -- Đúng --> Entry; C -- Sai --> Exit(( ));
```

Sau đây là một ví dụ tính tích số của N chữ số tự nhiên đầu tiên (còn gọi là “N giai thừa”) $N! = 1 \times 2 \times \dots \times N$. Để lấy tích số của N thừa số từ 1 đến N, ta sẽ lặp lại các bước như trong hình sau đây:

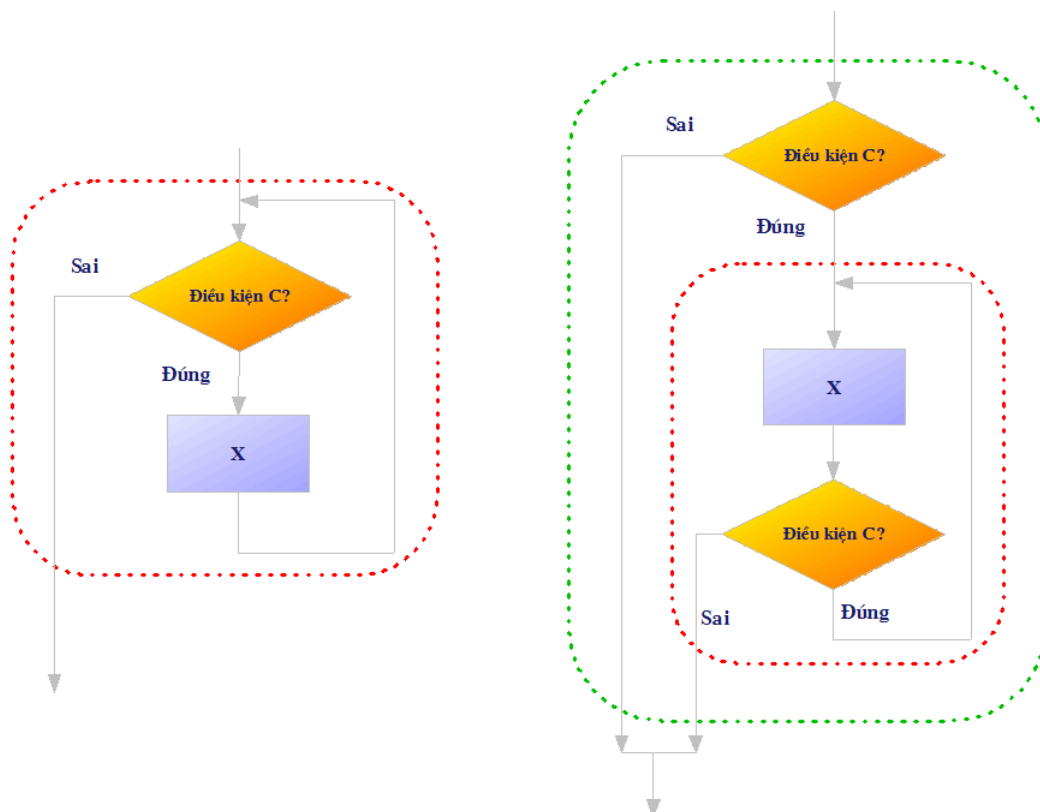


Hình 11.17: Thuật toán tính $N!$ Sau khi ra khỏi vòng lặp, biến số $nFactorial$ sẽ có giá trị là $N!$



Hình 11.18: Chương trình tính và in ra màn hình tích số của N số nguyên đầu tiên ($N!$). Lưu ý là chương trình sẽ thoát ra khỏi vòng lặp khi điều kiện ($nLoopCnt > N$) là ĐÚNG.

II.3 Chỉ thực hiện (nhóm) lệnh X và lặp lại việc thực hiện này nếu điều kiện C Đúng:





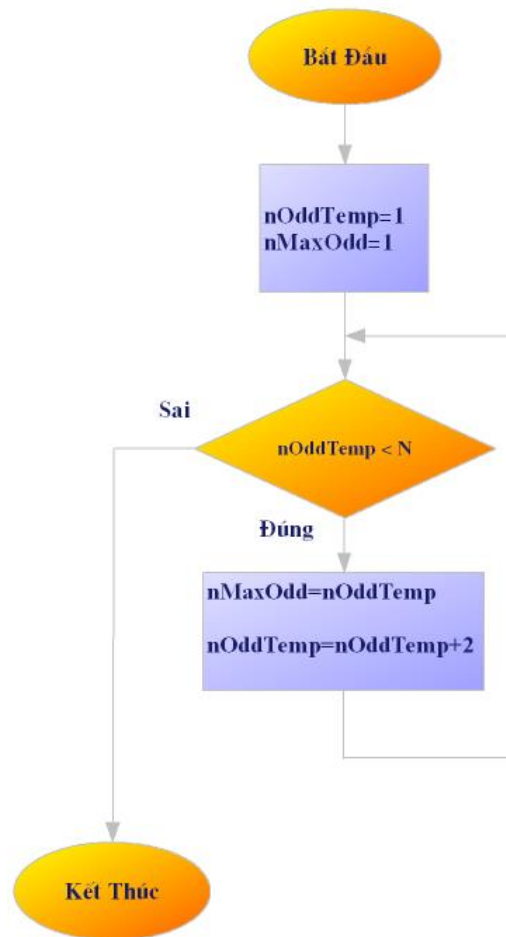
Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

Hình 11.19: A – Vòng lặp với lệnh X có thể bị bỏ qua không được thực hiện .

B - Cấu trúc điều khiển tương đương

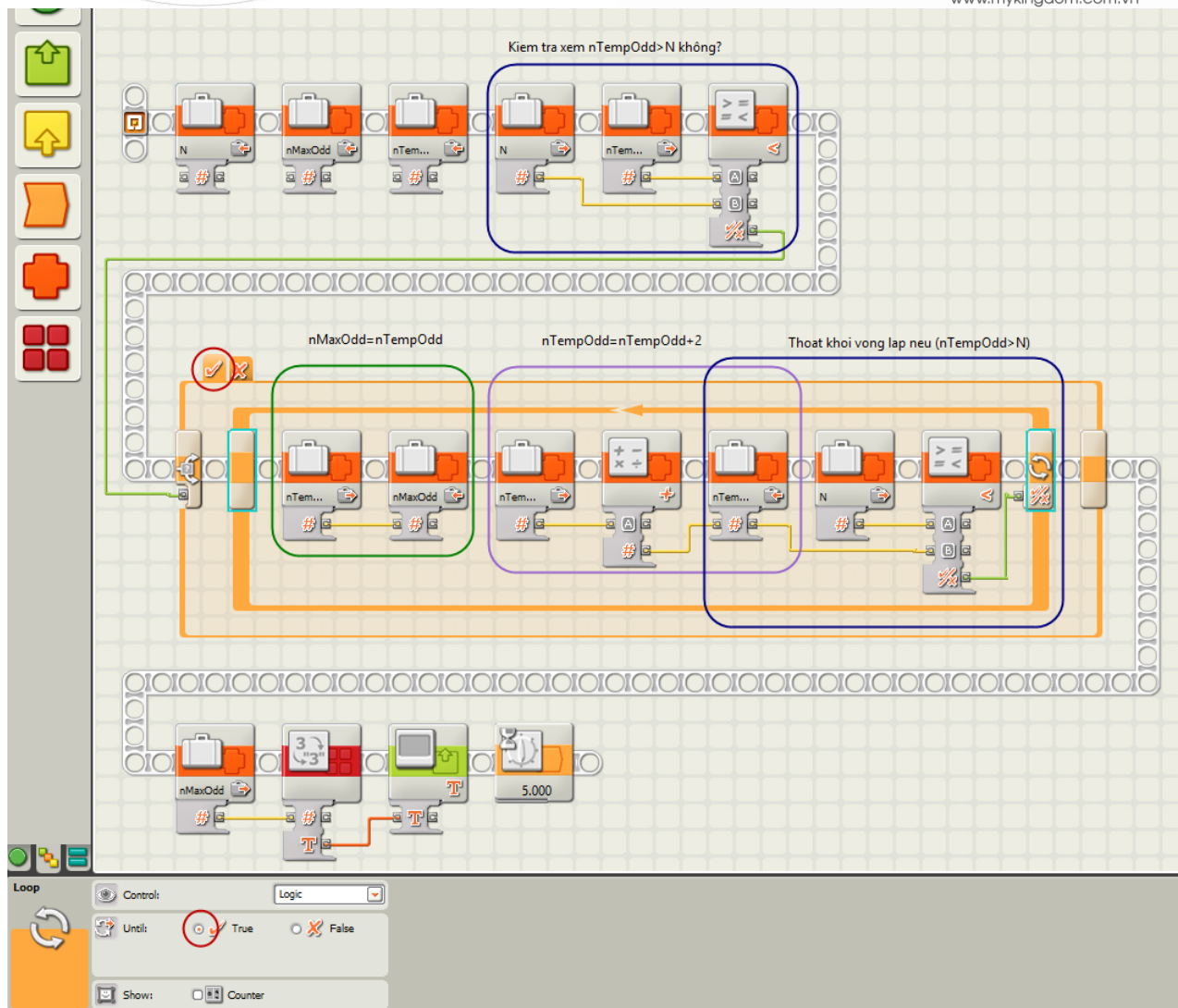
Ta có thể kết hợp 2 loại cấu trúc điều khiển **NẾU... THÌ...** và cấu trúc vòng lặp ở mục <II.1> để có thể thiết lập một cấu trúc điều khiển tương đương (Hình 11.20-B).

Sau đây ta sẽ viết một chương trình để tìm số lẻ lớn nhất trong phạm vi từ 1 đến N, với N là một số tự nhiên cho trước. Như ta đã biết số lẻ nhỏ nhất là 1, số lẻ tiếp theo bằng số lẻ trước nó cộng 2. Như vậy ta có thể xây dựng các bước tuần tự để giải bài toán này như sau.



Hình 11.20: Các bước tìm số lẻ lớn nhất trong phạm vi từ 1 đến N

Sau đây là một chương trình cụ thể có thể giải bài toán này.



Hình 11.21: Như đã đề cập ở trên, đoạn chương trình tìm số lẻ lớn nhất trong phạm vi từ 1 đến N này kết hợp khối lệnh [Switch] (dùng cho cấu trúc NẾU... THÌ...) và khối lệnh [Loop] để thiết lập một cấu trúc vòng lặp có kiểm tra điều kiện trước khi thực hiện lệnh trong thân vòng lặp ngay từ đầu.



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

CHƯƠNG 12 : LẬP TRÌNH VỚI THIẾT BỊ NGOẠI VI

Các cảm biến được xem như là các giác quan đơn giản của robot. Chúng giúp cho robot có thể thấy, nghe, tiếp xúc, đo lường v.v... các điều kiện của môi trường xung quanh. Do đó, việc điều khiển các cảm biến trong các chương trình điều khiển của các robot thực thụ là một phần rất quan trọng. Như đã đề cập ở chương <9>, bộ lắp ráp 9797 có các loại cảm biến sau: cảm biến chạm, cảm biến ánh sáng, cảm biến âm thanh, và cảm biến siêu âm. Sau đây chúng ta sẽ tìm hiểu cách lập trình để điều khiển các khối cảm biến này.

I. Điều khiển cảm biến chạm:

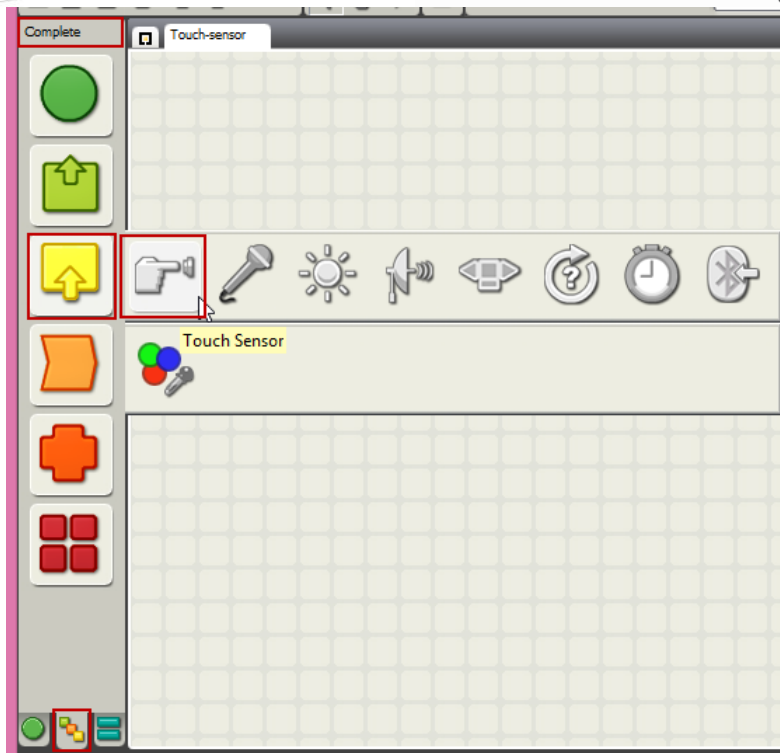
Như ta đã biết, cảm biến chạm là một loại cảm biến cho phép chúng ta biết là nó đã “chạm” vào một vật gì đó hay không. Bạn có thể sử dụng khối lệnh [Touch Sensor] hay các khối lệnh [Wait], [Switch], [Loop] (chọn giá trị “Sensor” cho thông số [Control] và giá trị “Touch Sensor” cho thông số [Sensor]) để điều khiển việc đọc trạng thái của cảm biến chạm. Trong phần này ta sẽ xem xét khối lệnh [Touch Sensor] và cách lập trình cho khối lệnh này. Từ nội dung này, bạn có thể dễ dàng suy ra cách dùng các khối lệnh còn lại đã nêu trên để điều khiển cảm biến chạm.

Khối lệnh [Touch Sensor] được dùng để đọc trạng thái của cảm biến chạm. Tùy theo bạn muốn kiểm tra trạng thái nào của cảm biến, thì giá trị lô-gic (Yes hoặc No) mà cảm biến trả về sẽ có một ý nghĩa riêng.

Trạng Thái cần Kiểm Tra (Action)	Giá Trị Trả Về	Ý Nghĩa
Nhấn (Pressed)	Yes	Cảm biến chạm đang bị “nhấn” vào thời điểm kiểm tra cảm biến.
	No	Cảm biến chạm không bị “nhấn” (trạng thái “thả”) tại thời điểm kiểm tra cảm biến.
Thả (Released)	Yes	Cảm biến chạm không bị “nhấn” tại thời điểm kiểm tra cảm biến.
	No	Cảm biến chạm đang bị “nhấn” vào thời điểm kiểm tra cảm biến.
Đụng (Bumped)	Yes	Cảm biến chạm vừa bị “đụng” vào, tức là bị “nhấn và thả” nhanh (từ lúc bắt đầu nhấn vào cho đến lúc thả ra nằm trong khoảng thời gian 0,5 giây) tại thời điểm kiểm tra cảm biến.
	No	Cảm biến chạm không ở trạng thái Đụng tại thời điểm kiểm tra cảm biến.

Bảng 12.1 : Giá trị trả về của cảm biến chạm và ý nghĩa của chúng.

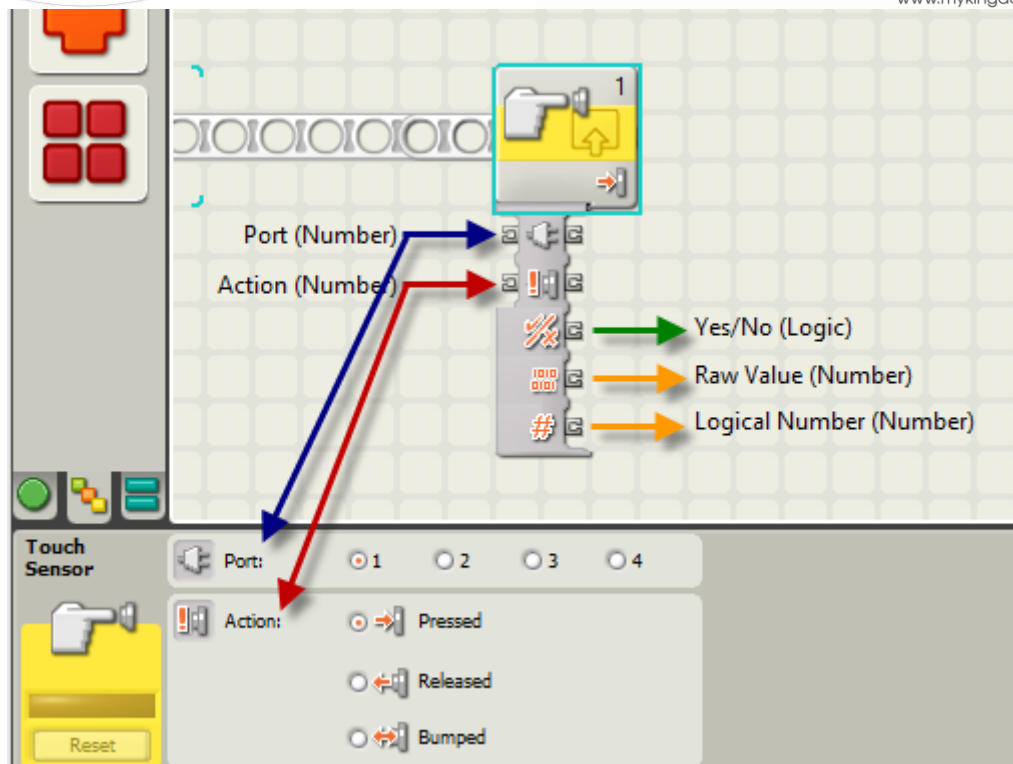
Bạn có thể lấy khối lệnh [Touch Sensor] từ nhóm [Sensor] của thẻ [Complete] trên Bảng các khối lệnh.



Hình 12.1: Biểu tượng của khối lệnh [Touch Sensor] trong nhóm [Sensor] của thẻ [Complete] trên Bảng các khối lệnh.

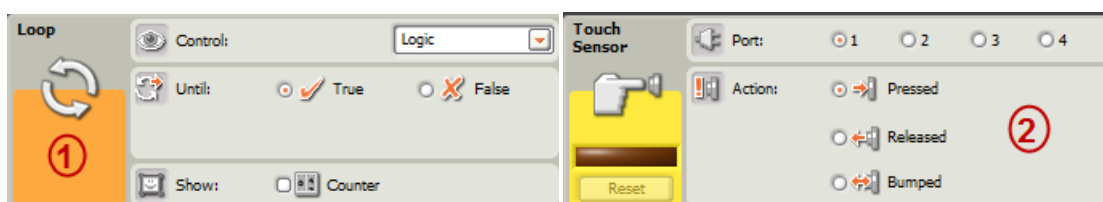
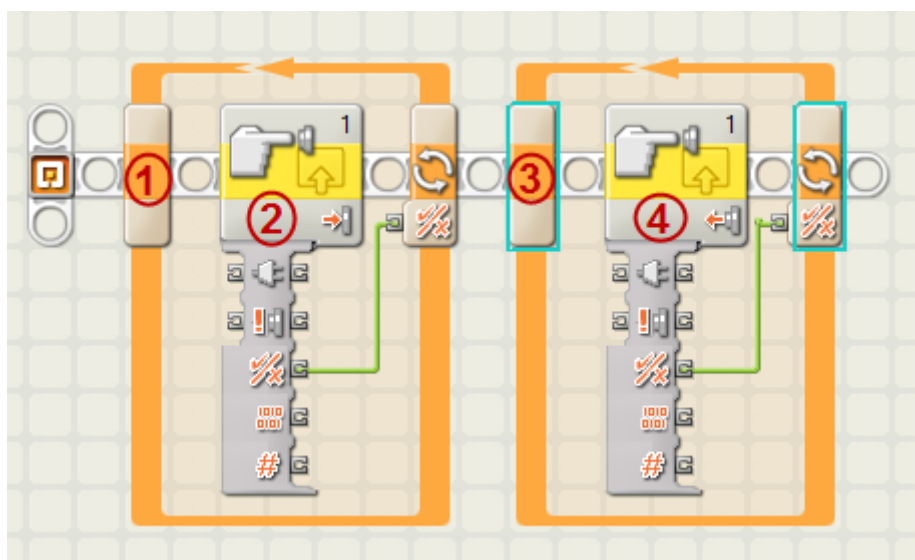
Trước khi kiểm tra trạng thái của cảm biến chạm, bạn cần phải thiết lập các thông số sau đây trên Bảng cấu hình của khối lệnh [Touch Sensor]. Thông số [Port], tức là cổng của khối vi điều khiển NXT mà bạn đang nối cảm biến này vào. Có 4 cổng đánh số từ 1 đến 4, do đó thông số [Port] có thể nhận 1 trong 4 giá trị này. Thông số [Action] là trạng thái của cảm biến chạm mà bạn muốn kiểm tra (xem chi tiết ở bảng 12.1). Có 3 trạng thái cho bạn chọn lựa, đó là “Pressed”, “Released”, và “Bumped”.

Để biết được trạng thái của cảm biến, bạn cần phải lấy kết quả từ một trong các chốt dữ liệu ra sau đây. Chốt [Yes/No] trả về kết quả là một giá trị kiểu Logic. Chốt [Logical Number] trả về kết quả là một giá trị kiểu Number tương đương với các giá trị luận lý (Yes→1, No→0). Chốt [Raw Value] trả về các giá trị dữ liệu thô chưa qua xử lý, đây là các số kiểu Number diễn tả trạng thái thực (Nhấn/Thả/Đụng) của cảm biến chạm, bất kể là thông số đầu vào [Action] được thiết lập như thế nào. Thường thì bạn dùng chốt [Yes/No] hoặc [Logical Number] tùy theo kiểu dữ liệu mà bạn đang muốn xử lý. Còn chốt [Raw Value] thì đòi hỏi bạn phải tìm hiểu sâu hơn để biết chính xác các giá trị thô này để xử lý, nhưng bù lại với dữ liệu thô bạn có thể biết chính xác trạng thái của cảm biến chạm trong một lần đọc trong khi với 2 chốt dữ liệu ra kia bạn có thể phải đọc nhiều lần.



Hình 12.2 : Khối lệnh [Touch Sensor] và Bảng cấu hình của nó.

Đoạn chương trình sau sẽ đợi cho tới khi bạn nhấn thả vào cảm biến chạm thì mới thoát khỏi chương trình.





Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



Hình 12.3: Chương trình phát hiện người dùng nhấn thả cảm biến chạm. Các Bảng cấu hình được đánh số tương ứng với các khối lệnh ở trong Vùng làm việc.

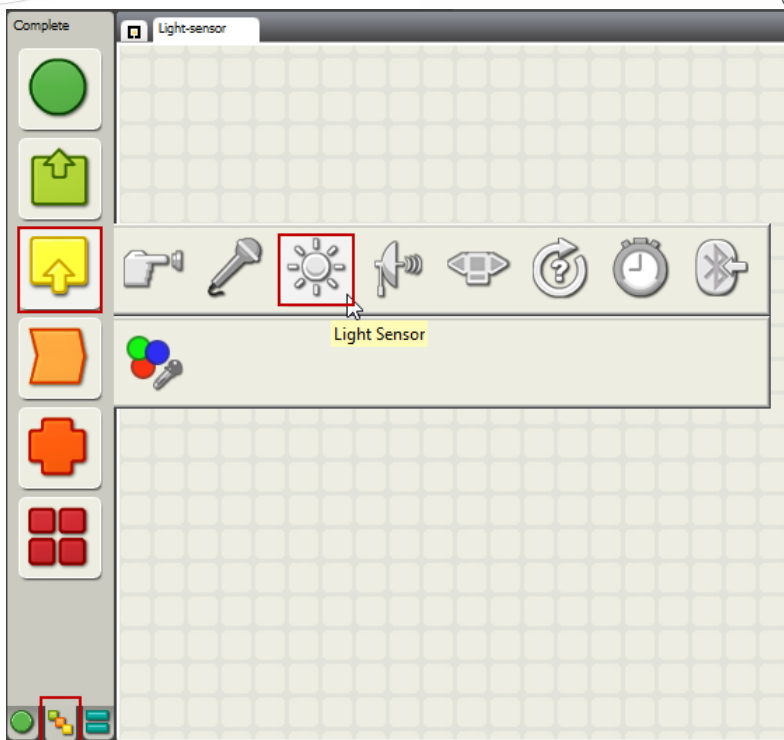
II. Điều khiển cảm biến ánh sáng:

Cảm biến ánh sáng là một loại cảm biến có thể cảm nhận được độ sáng của môi trường xung quanh. Như ta đã biết, khối cảm biến ánh sáng của bộ lắp ráp Lego 9797 gồm đèn LED và một quang tranzito (phototransistor). Quang tranzito có thể đo được cường độ ánh sáng của môi trường xung quanh, hay khi đèn LED được bật lên thì quang tranzito có thể đọc được ánh sáng phản chiếu từ đối tượng được chiếu sáng bằng đèn LED. Thông thường thì đối tượng càng nhạt màu thì cường độ ánh sáng phản chiếu càng mạnh. Khi đọc trạng thái của cảm biến ánh sáng, bạn thường lấy ra được một giá trị nằm trong phạm vi từ 0 (độ sáng yếu nhất) đến 100 (độ sáng cao nhất).

Tương tự như việc điều khiển cảm biến chạm, bạn có thể dùng khối lệnh [Light Sensor] hay các khối lệnh [Wait], [Switch], [Loop] (chọn giá trị “Sensor” cho thông số [Control] và giá trị “Light Sensor” cho thông số [Sensor]) để điều khiển việc đọc trạng thái của cảm biến ánh sáng. Ngoài ra, trong ngôn ngữ NXT còn có khối lệnh [Calibrate] cho phép bạn cân chỉnh lại cảm biến. Nghĩa là, nếu bạn biết trước robot của mình sẽ làm việc trong một môi trường có cường độ ánh sáng thay đổi trong một phạm vi cho trước [a, b] nào đó thì bạn có thể cân chỉnh cảm biến sao cho giá trị trả về 0 sẽ tương ứng với cường độ ánh sáng a, và giá trị trả về 100 sẽ tương ứng với cường độ ánh sáng b. Điều này sẽ giúp bạn đo được các giá trị cường độ ánh sáng trong phạm vi [a, b] tương đối chính xác hơn để xử lý cho phù hợp (bạn có nhiều giá trị hơn để diễn tả một phạm vi cần đo lường cho trước).

Trong phần này ta cũng chỉ xét khối lệnh [Light Sensor] và cách lập trình cho khối lệnh này. Từ nội dung này, bạn cũng có thể dễ dàng suy ra cách dùng các khối lệnh còn lại đã nêu trên để điều khiển cảm biến ánh sáng.

Bạn có thể lấy khối lệnh [Light Sensor] từ nhóm [Sensor] của thẻ [Complete] trên Bảng các khối lệnh.



Hình 12.4 : Biểu tượng của khối lệnh [Light Sensor] trong nhóm [Sensor] của thẻ [Complete] trên Bảng các khối lệnh.

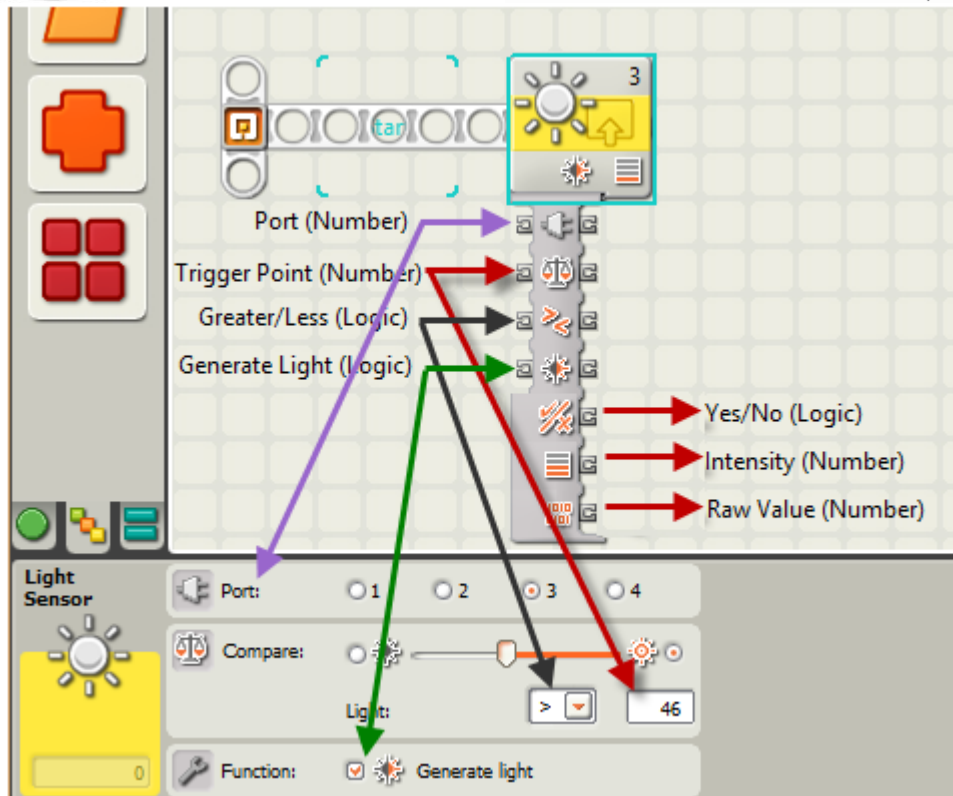
Có 3 cách đọc trạng thái của cảm biến ánh sáng. Cách thứ nhất là kiểm tra xem cường độ ánh sáng có lớn hơn (hay nhỏ hơn) một giá trị ngưỡng cho trước hay không. Cách thứ 2 là đọc giá trị tương đối của cường độ ánh sáng trong thang giá trị từ 0 đến 100 (lưu ý là sau mỗi lần cân chỉnh lại cảm biến ánh sáng thì giá trị tương đối của cường độ ánh sáng đo được của cùng một nguồn sáng không đổi có thể khác nhau). Cách thứ 3 là đọc ra giá trị thô của cường độ ánh sáng đo được theo một thang giá trị riêng (từ 0 đến 1000) của cảm biến. Do nhu cầu, chúng ta thường dùng 2 cách đọc đầu tiên.

Để kiểm tra trạng thái của cảm biến ánh sáng, bạn cần phải thiết lập các thông số sau đây trên Bảng cấu hình của khối lệnh [Light Sensor]:

Thông số [Port], tức là cổng của khối vi điều khiển NXT mà bạn đang nối cảm biến này vào. Có 4 cổng đánh số từ 1 đến 4, do đó thông số [Port] có thể nhận 1 trong 4 giá trị này (giá trị mặc định cho cảm biến ánh sáng là 3).

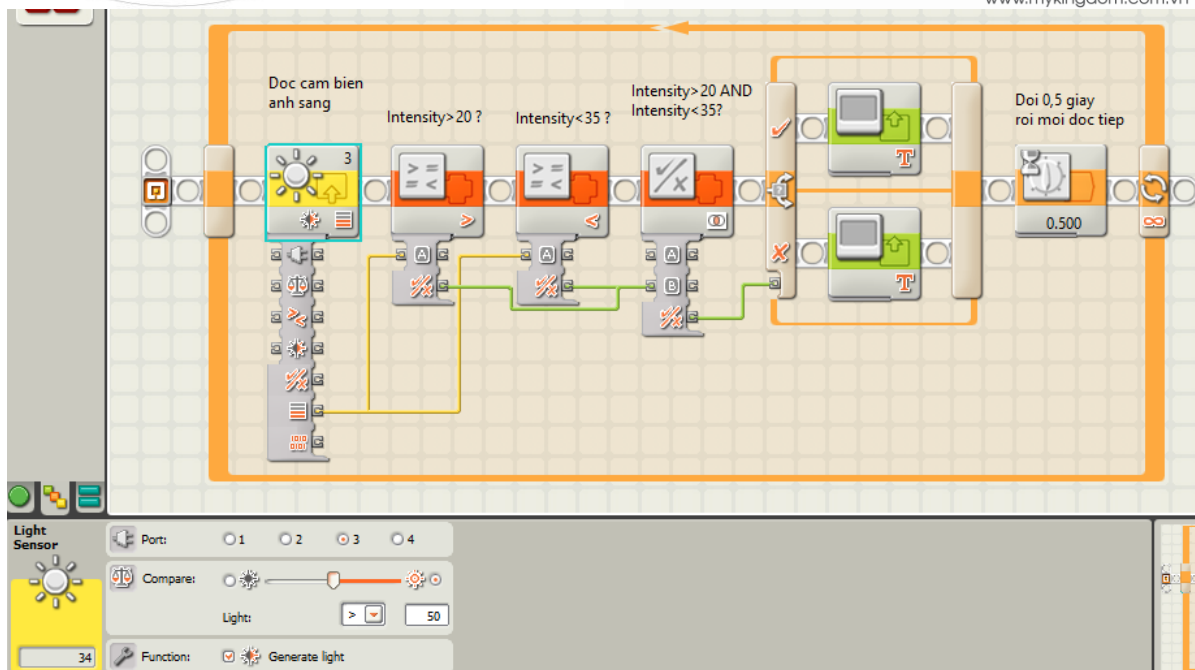
Nếu bạn đọc cảm biến theo cách thứ nhất như đã nói ở trên thì bạn phải xác định thêm các thông số sau ở phần [Compare]. Đó là thông số [Trigger Point] là một giá trị ngưỡng trong phạm vi từ 0 đến 100 được dùng để so sánh với giá trị tương đối của cường độ ánh sáng đo được; và thông số [Greater/Less] đó là cách mà bạn muốn so sánh (“>”: so sánh lớn hơn, hay “<”: so sánh nhỏ hơn). Ví dụ, nếu [Trigger Point] = 46 và [Greater/Less] = “>” thì điều đó có nghĩa là: nếu giá trị tương đối của cường độ ánh sáng đo được là $X > 46$ thì kết quả mà bạn đọc được từ cảm biến ánh sáng là True, còn nếu $X \leq 46$ thì kết quả bạn đọc được sẽ là False.

Nếu bạn muốn bật đèn LED để chiếu vào một đối tượng và đo cường độ của ánh sáng phản chiếu từ đối tượng đó thì bạn phải chọn “Generate Light” cho thông số [Function].

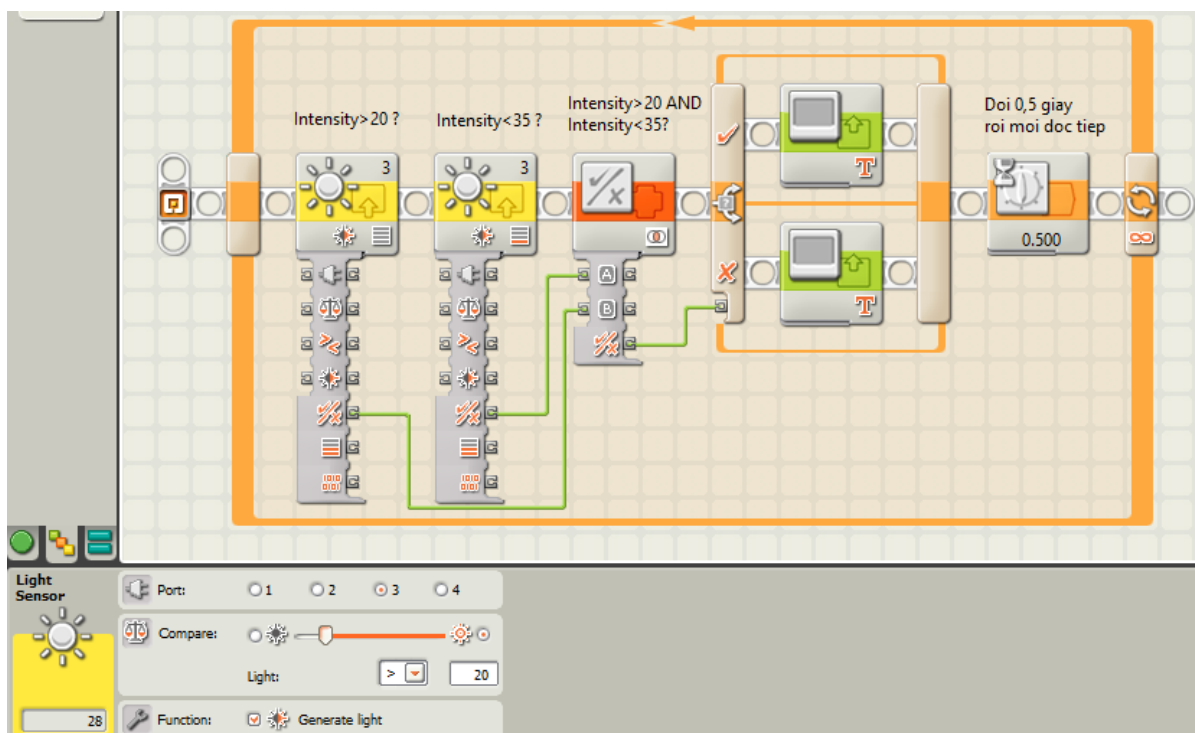


Hình12.5 : Khối lệnh [Light Sensor] và Bảng cấu hình của nó. Để đọc trạng thái của cảm biến theo cách thứ nhất, bạn dùng chốt dữ liệu ra [Yes/No]; với cách thứ 2, bạn dùng chốt dữ liệu ra [Intensity]; và bạn sẽ phải dùng chốt dữ liệu ra [Raw Value] với cách đọc cuối cùng.

Cảm biến ánh sáng thường được ví như là mắt của robot. Người ta thường dùng cảm biến ánh sáng trong các robot tự động dò đường (đi theo một con đường đã được sơn màu), hay các robot biết cách xác định phương hướng di chuyển (robot đá bóng với quả bóng có phát ra ánh sáng), v.v... Sau đây là đoạn chương trình mô phỏng quá trình đọc cảm biến để dò đường đi. Để cho đơn giản chúng ta chỉ dùng một khối cảm biến ánh sáng nối vào khối vi điều khiển NXT qua cổng số 3 (mặc định). Giả sử sau khi đo cường độ ánh sáng phản chiếu của toàn bộ bề mặt của con đường (đặt cảm biến cách mặt đường chừng 1cm, bật đèn LED, di chuyển khắp mặt đường và đọc các giá trị của cảm biến), ta biết được giá trị đo được nằm trong phạm vi từ 21 đến 34. Như vậy nếu giá trị đọc được từ cảm biến nằm ngoài phạm vi này thì cảm biến đang nằm ngoài chu vi của con đường. Chương trình của chúng ta sẽ liên tục đọc cảm biến ánh sáng (mỗi 0,5 giây một lần – chu kỳ đọc có thể tăng hay giảm tùy theo vận tốc của robot) và hiển thị một trong hai dòng chữ “Đúng Đường” (Đúng Đường) hoặc “Sai Đường!” (Sai Đường!) tùy theo kết quả đọc được.



Hình 12.6: Đọc cảm trạng thái biến ánh sáng qua chốt dữ liệu ra [Intensity]. Tiếp đó ta sẽ dùng các khối lệnh [Compare] để so sánh cường độ ánh sáng đo được với phạm vi cho trước.



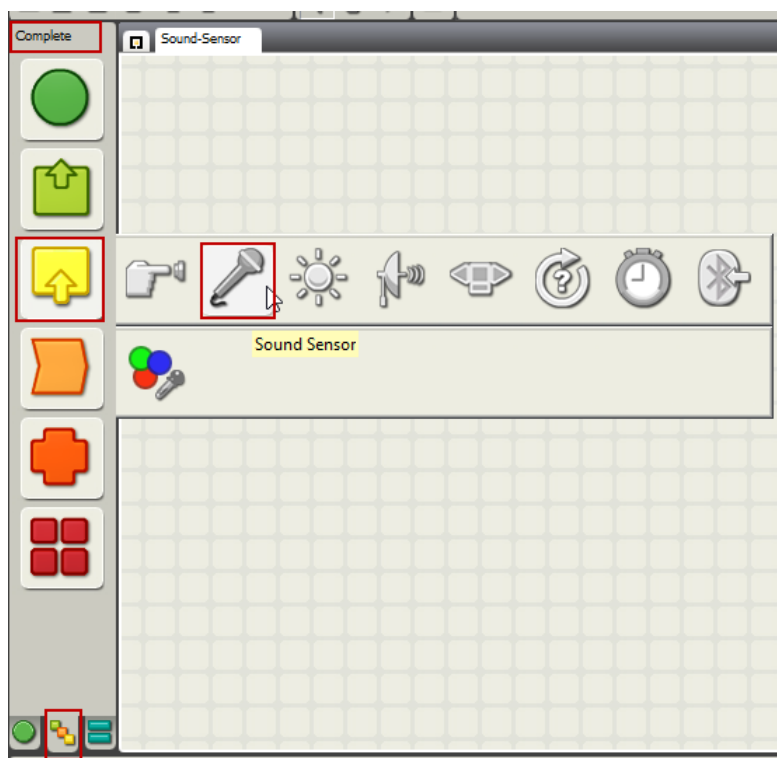
Hình 12.7: Đọc trạng thái cảm biến ánh sáng qua chốt dữ liệu ra [Yes/No]. Khối lệnh [Light Sensor] cũng có khả năng so sánh cường độ ánh sáng đo được với một giá trị ngưỡng nào đó và trả về kết quả của việc so sánh. Tuy nhiên với đoạn chương trình này, ta phải đọc cảm biến 2 lần liên tiếp để có thể thực hiện việc so sánh với 2 giá trị ngưỡng của phạm vi cho trước.

III. Điều khiển cảm biến âm thanh:

Cảm biến âm thanh của bộ Lego 9797 có thể được xem như là tai của robot. Đó là một loại cảm biến có khả năng phát hiện ra âm thanh và đo được cường độ của nó. Khác với tai người – có khả năng phân biệt được nhiều điểm khác biệt của âm thanh như cường độ, cao độ (liên quan đến tần số của dao động sóng âm) của âm thanh, và các âm thanh khác nhau do các loại đồ vật khác nhau gây ra – cảm biến âm thanh mà chúng ta sẽ dùng chỉ có khả năng đo được cường độ của âm thanh mà thôi.

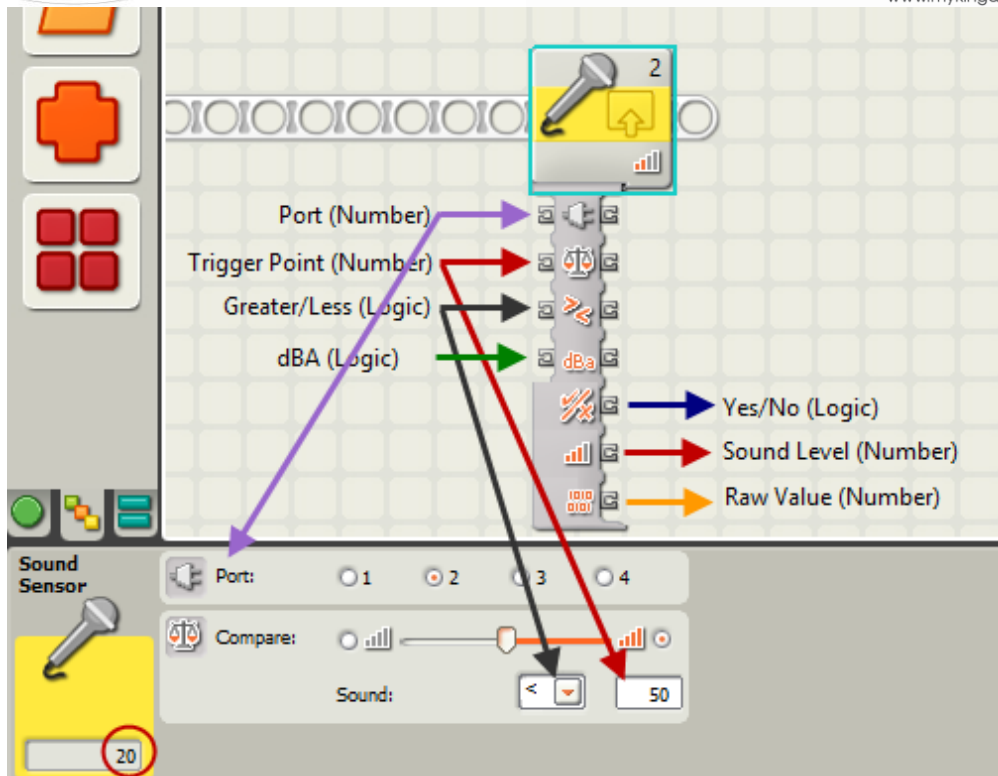
Ta có thể dùng cảm biến âm thanh trong các robot biết nhận lệnh bằng âm thanh. Ví dụ, tùy theo cường độ âm thanh nhận được mà robot có thể đi thẳng, rẽ trái, rẽ phải, đi lùi v... Hay một robot do thám sẽ di chuyển khi không nghe thấy tiếng động, đứng im hay đi tìm chỗ ẩn nấp nếu nghe thấy tiếng người đi lại ở gần đó.

Bạn có thể sử dụng khối lệnh [Sound Sensor] hay các khối lệnh [Wait], [Switch], [Loop] (chọn giá trị “Sensor” cho thông số [Control] và giá trị “Sound Sensor” cho thông số [Sensor]) để điều khiển việc đọc trạng thái của cảm biến âm thanh. Sau đây ta sẽ xem xét khối lệnh [Sound Sensor] và cách lập trình cho khối lệnh này. Từ nội dung này, bạn có thể dễ dàng suy ra cách dùng các khối lệnh còn lại đã nêu trên để điều khiển cảm biến âm thanh.



Hình 12.8: Biểu tượng của khối lệnh [Sound Sensor] trong nhóm [Sensor] của thẻ [Complete] trên Bảng các khối lệnh.

Có 3 cách đọc trạng thái của cảm biến âm thanh. Cách thứ nhất là kiểm tra xem cường độ âm thanh đo được có lớn hơn (hay nhỏ hơn) một giá trị ngưỡng cho trước hay không. Cách thứ 2 là đọc giá trị tương đối của cường độ âm thanh trong thang giá trị từ 0 đến 100 (lưu ý là sau mỗi lần cân chỉnh lại cảm biến âm thanh thì giá trị tương đối của cường độ âm thanh đo được của cùng một âm thanh có thể khác nhau). Cách thứ 3 là đọc ra giá trị thô của cường độ âm thanh đo được theo một thang giá trị riêng (từ 0 đến 1000) của cảm biến. Do nhu cầu, chúng ta thường dùng 2 cách đọc đầu tiên.



Hình12.9: Khối lệnh [Sound Sensor] với các chốt dữ liệu vào và ra, cùng với Bảng cấu hình tương ứng.

Để kiểm tra trạng thái của cảm biến âm thanh, bạn cần phải xác định các thông số đầu vào sau của khối lệnh [Sound Sensor]:

Thông số [Port], tức là cổng của khối vi điều khiển NXT mà bạn đang nối cảm biến này vào. Có 4 cổng đánh số từ 1 đến 4, do đó thông số [Port] có thể nhận 1 trong 4 giá trị này (giá trị mặc định cho cảm biến âm thanh là 2).

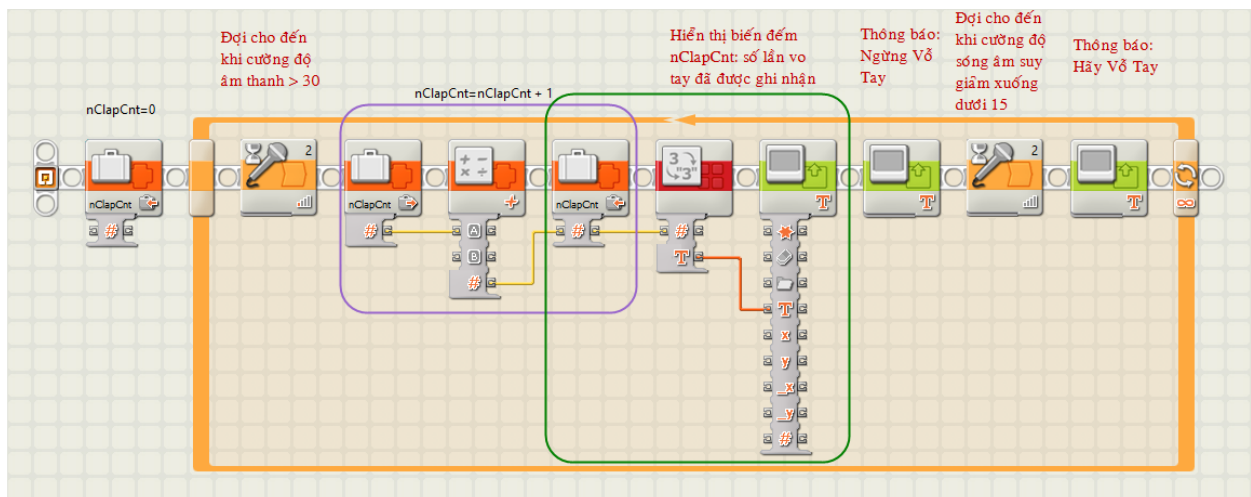
Nếu bạn đọc cảm biến theo cách thứ nhất như đã nói ở trên thì bạn phải xác định thêm các thông số sau ở phần [Compare]. Đó là thông số [Trigger Point] là một giá trị ngưỡng trong phạm vi từ 0 đến 100 được dùng để so sánh với giá trị tương đối của cường độ tương đối của âm thanh đo được; và thông số [Greater/Less] đó là cách mà bạn muốn so sánh (" $>$ ": so sánh lớn hơn, hay " $<$ ": so sánh nhỏ hơn). Ví dụ, nếu [Trigger Point] = 50 và [Greater/Less] = " $<$ " thì điều đó có nghĩa là: nếu giá trị tương đối của cường độ âm thanh đo được là $X < 50$ thì kết quả mà bạn đọc được từ cảm biến ánh sáng là True, còn nếu $X \geq 50$ thì kết quả bạn đọc được sẽ là False.

Cảm biến âm thanh có thể đo cường độ âm thanh theo đơn vị dB hay dB(A). Nếu bạn truyền giá trị True cho chốt dữ liệu vào [dBA], thì cảm biến sẽ dùng đơn vị dB(A), tức là cảm biến sẽ đo được các âm thanh mà tai người nghe được. Nếu bạn truyền giá trị False hay không truyền gì cả cho chốt dữ liệu vào này thì cảm biến sẽ dùng đơn vị dB, tức là cảm biến có thể đo được các âm thanh mà tai người không nghe được. Tuy nhiên, trong tài liệu này ta sẽ không tác động gì đến chốt dữ liệu vào [dBA] này, và tất nhiên là ta chỉ dùng đơn vị dB cho cảm biến âm thanh.

Sau đây ta sẽ xét một đoạn chương trình đơn giản đếm số lần vỗ tay. Mặc dù người ta có thể dễ dàng nghe rồi đếm số lần một người đã vỗ tay với những tốc độ khác nhau một cách chính xác, tuy nhiên việc này nếu

để cho máy làm là một quá trình phức tạp liên quan đến việc phân tích sóng âm thanh do tiếng vỗ tay tạo ra. Tuy nhiên, để đơn giản hóa vấn đề, trong ví dụ này ta cho chương trình hiển thị thêm thông tin để điều khiển tốc độ vỗ tay của người chơi. Tức là khi chương trình đã sẵn sàng cho việc ghi nhận tiếng vỗ tay thì màn hình hiện lên dòng chữ “Hay Vo Tay”, còn khi nó đang bận chưa thể xử lý hoặc sóng âm do tiếng vỗ tay trước đó chưa kết thúc (cường độ âm thanh vẫn còn lớn hơn một ngưỡng đã biết) thì màn hình sẽ hiện lên dòng chữ “Ngung Vo Tay”. Ngoài dòng chữ thông báo để điều khiển tốc độ vỗ tay, màn hình còn hiển thị thêm dòng chữ cho biết số lần vỗ tay đã ghi nhận được.

Giả sử sau khi đo đạc và thử nghiệm, ta biết cường độ âm thanh của phòng (có tạp âm) khi không vỗ tay là 15, và khi vỗ tay thì cường độ âm thanh đo được là > 30 . Như vậy ta có thể nhận định là thời gian sóng âm thanh, tạo ra do một lần vỗ tay, tồn tại từ khi ta ghi nhận được cường độ âm thanh > 30 đến khi cường độ âm thanh bị suy giảm xuống dưới 15 đơn vị. Do đó ta có thể dùng 2 khối lệnh [Wait – Sound Sensor] (để đợi tới khi có cường độ âm thanh > 30 đơn vị và sau đó đợi tiếp cho đến khi cường độ âm thanh giảm xuống dưới 15 đơn vị) để biết được khi nào có một tiếng vỗ tay. Sau đây là đoạn chương trình chi tiết.



Hình 12.10: Chương trình đếm số lần vỗ tay sử dụng 2 khối lệnh [Wait – Sound Sensor].

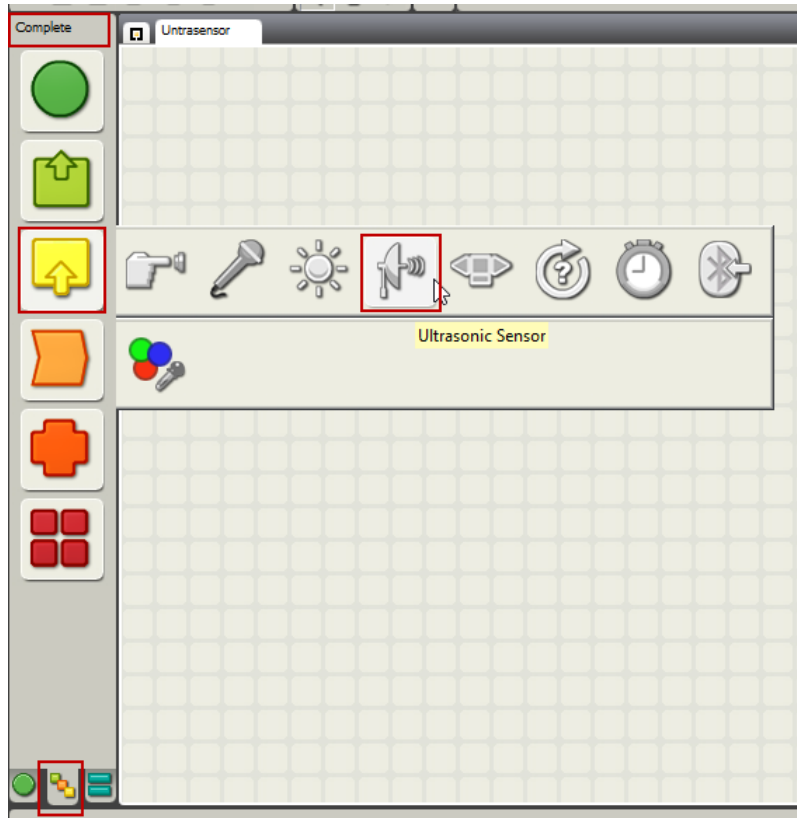
IV. Điều khiển cảm biến siêu âm:

Như ta đã biết, lợi dụng hiện tượng phản xạ của sóng âm, người ta có thể đo được khoảng cách giữa nguồn phát ra sóng âm và vật cản bằng cách đo khoảng thời gian từ lúc sóng âm phát đi, gặp vật cản, phản xạ rồi quay về lại nguồn phát. Cảm biến siêu âm mà ta sẽ khảo sát sau đây là loại cảm biến dùng sóng siêu âm (âm thanh có tần số cao hơn tần số tối đa mà tai người có thể nghe được) để đo được khoảng cách giữa nó và một đối tượng khác.

Với khả năng đo lường khoảng cách đến các chướng ngại vật, cảm biến siêu âm được sử dụng trong nhiều ứng dụng khác nhau, ví dụ chuông cửa (chuông kêu khi cửa mở), hệ thống báo động khi có người tiến lại gần, đi theo một đối tượng và luôn giữ một khoảng cách an toàn, dùng tay không chơi nhạc (âm thanh phát ra tùy theo cử động của tay) v.v...

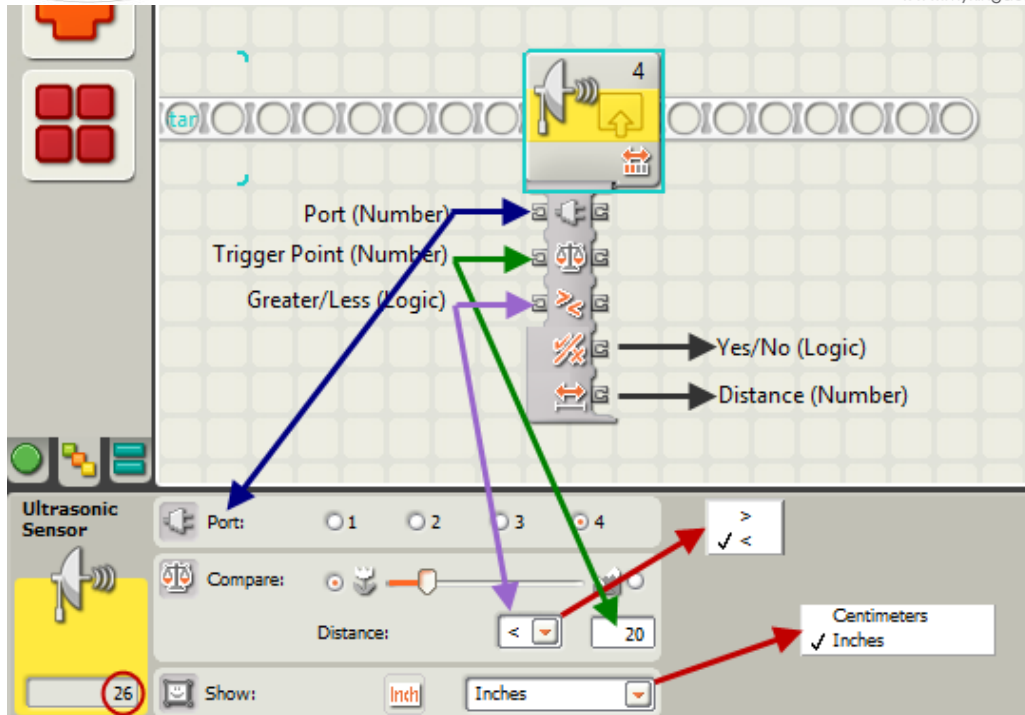
Bạn có thể sử dụng khối lệnh [Ultrasonic Sensor] hay các khối lệnh [Wait], [Switch], [Loop] (chọn giá trị “Sensor” cho thông số [Control] và giá trị “Ultrasonic Sensor” cho thông số [Sensor]) để điều khiển việc đọc trạng thái của cảm biến siêu âm. Sau đây ta sẽ xem xét khối lệnh [Ultrasonic Sensor] và cách lập trình cho

khối lệnh này. Từ nội dung này, bạn có thể dễ dàng suy ra cách dùng các khối lệnh còn lại đã nêu trên để điều khiển cảm biến siêu âm.



Hình 12.11: Biểu tượng của khối lệnh [Ultrasonic Sensor] trong nhóm [Sensor] của thẻ [Complete] trên Bảng các khối lệnh.

Có 2 cách đọc trạng thái của cảm biến siêu âm. Cách thứ nhất là kiểm tra xem khoảng cách đo được có lớn hơn (hay nhỏ hơn) một giá trị ngưỡng cho trước hay không. Do khoảng cách tối đa mà cảm biến có thể đo được là 100 in (inch) hay là 255 cm, nên giá trị ngưỡng phải nằm trong phạm vi này. Cách đọc thứ 2 là đọc giá trị thực sự của khoảng cách đo được. Lưu ý là khi không có vật cản hoặc vật cản nằm ngoài phạm vi đo được thì giá trị trả về luôn là 100 in hay 255 cm.

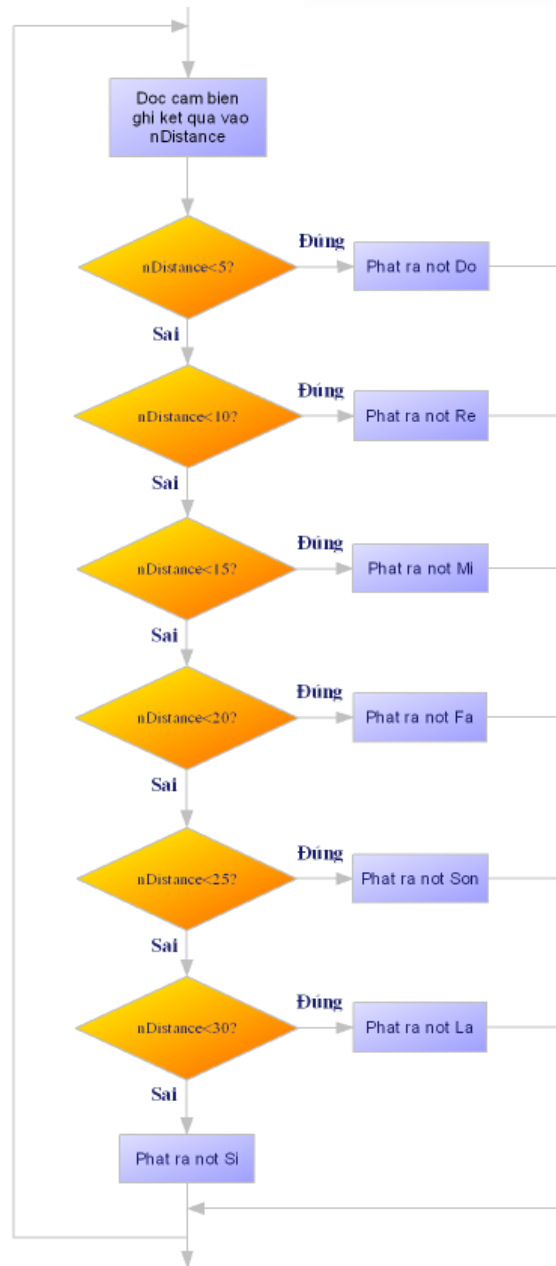


Hình12.12: Khối lệnh [Ultrasonic Sensor] với các chốt dữ liệu vào và ra, cùng với Bảng cấu hình tương ứng.

Sau đây ta sẽ viết một đoạn chương trình mô phỏng máy chơi nhạc bằng tay không, dùng cảm biến siêu âm. Chương trình có thể phát ra 7 nốt nhạc Đồ (C), Rê (D), Mi (E), Fa (F), Son (G), La (A), Si (B) tùy theo khoảng cách từ cảm biến đến bàn tay của người chơi (d).

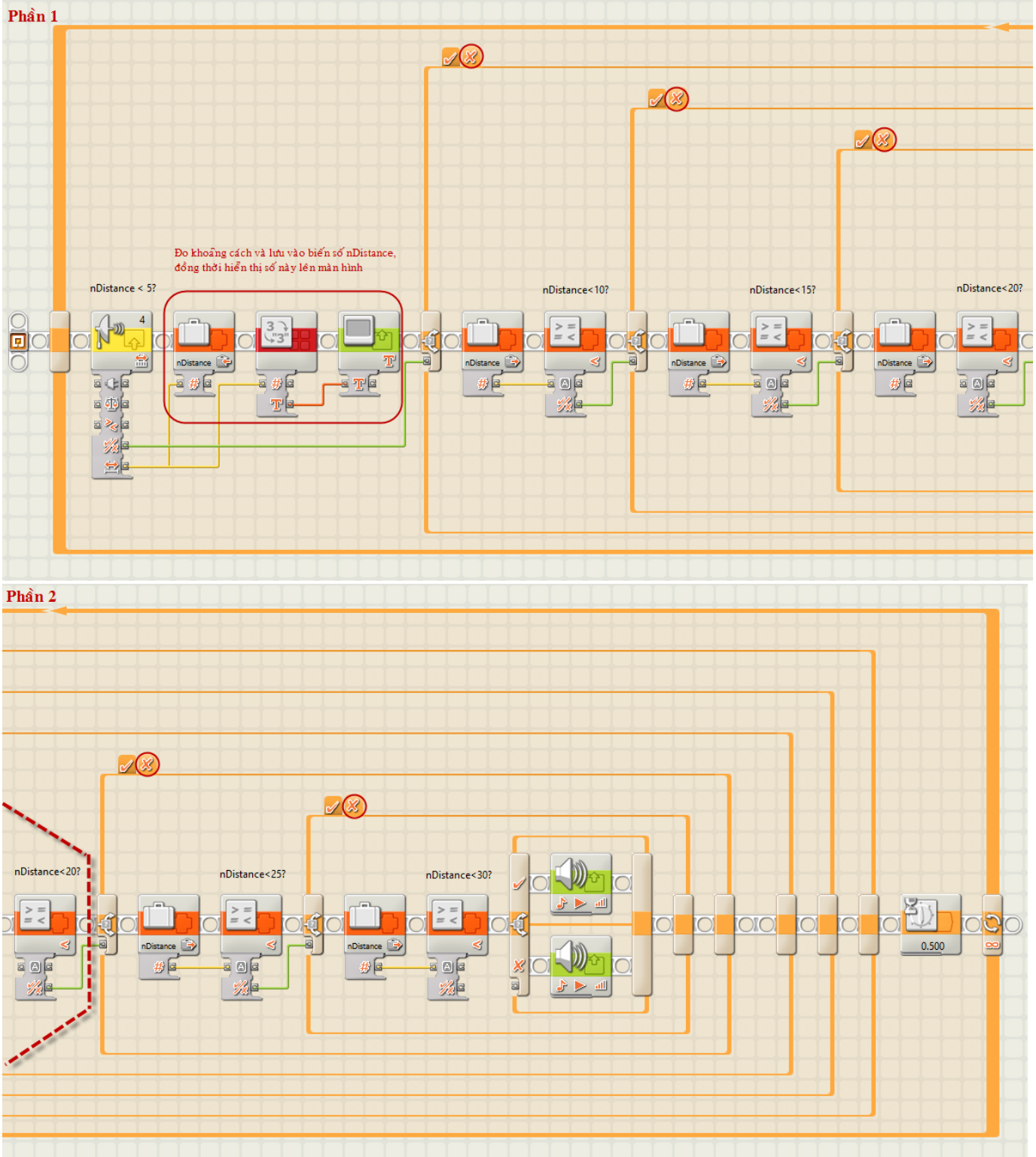
- $0 < d < 5$: chương trình phát ra nốt nhạc Đồ (C)
- $5 \leq d < 10$: chương trình phát ra nốt nhạc Rê (D)
- $10 \leq d < 15$: chương trình phát ra nốt nhạc Mi (E)
- $15 \leq d < 20$: chương trình phát ra nốt nhạc Fa (F)
- $20 \leq d < 25$: chương trình phát ra nốt nhạc Son (G)
- $25 \leq d < 30$: chương trình phát ra nốt nhạc La (A)
- $d \geq 30$: chương trình phát ra nốt nhạc Si (B)

Để phát ra nhạc thì ta sẽ dùng khối lệnh [Sound] (trong thẻ [Common] của bảng Các khối lệnh. Căn cứ theo mô tả ở thì ta có sơ đồ các bước thực hiện chính như sau.

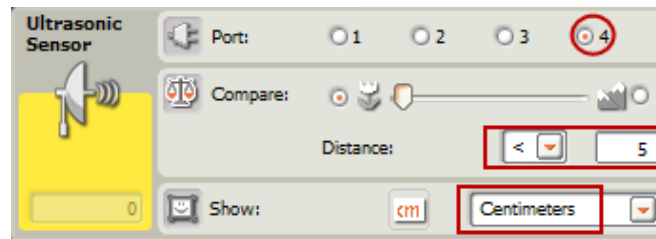


Hình 12.13: Sơ đồ các bước mô phỏng máy chơi nhạc bằng tay không.

Sau đây là chương trình chi tiết.

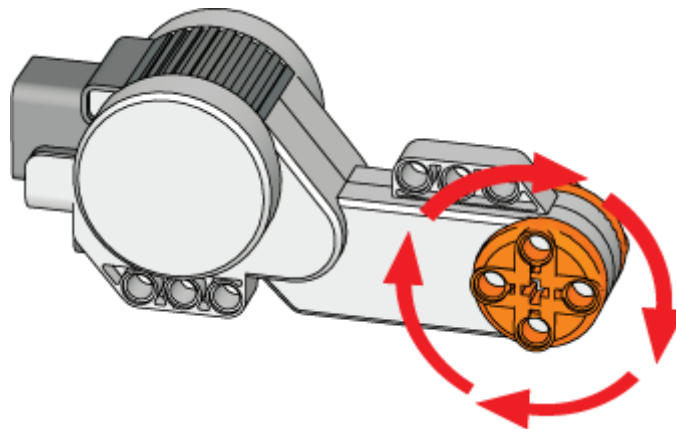


Hình12.14 : Chương trình chơi nhạc bằng tay không. Lưu ý là hình vẽ chỉ trình bày phần False của các khối lệnh [Switch]. Nhánh lệnh True chỉ chứa một khối lệnh [Sound] và phát ra nốt nhạc tương ứng.



Hình 12.15: Bảng cấu hình của khối lệnh [Ultrasonic Sensor] và một khối lệnh [Sound] phát ra nốt nhạc Si (A).

V. Điều khiển động cơ:



Hình 12.16: Chiều quay thuận của động cơ. Lưu ý là tùy theo cách lắp ráp robot của bạn mà chiều quay thuận của động cơ có thể khiến cho robot đi tới hoặc lùi. Do đó bạn cần phải xem xét kỹ để viết chương trình điều khiển cho robot đi đúng hướng.

Khối lệnh [Motor]:

Khối lệnh này cho phép bạn điều khiển vận tốc của một động cơ đã chọn với một độ chính xác khá cao. Bạn có thể tìm thấy khối lệnh này trong nhóm [Action] của thẻ [Complete] trên Bảng các khối lệnh.



Hình 12.17: Biểu tượng của khối lệnh [Motor] trong nhóm [Action] của thẻ [Complete] trên Bảng các khối lệnh.

Thông thường, chuyển động của robot bao gồm ba giai đoạn như sau:

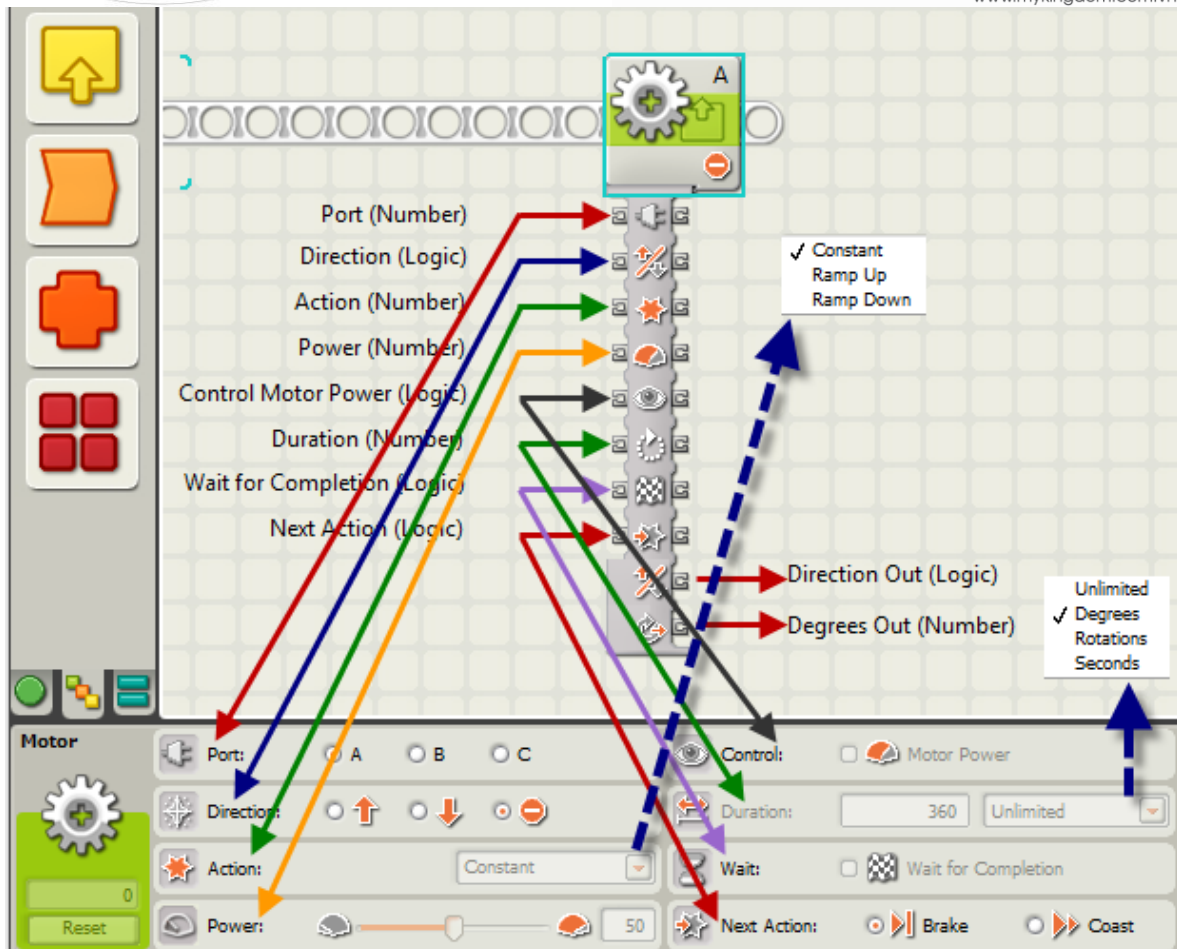
Giai đoạn tăng tốc: Động cơ gia tăng lực quay cho đến khi đạt được mức lực quay đã được thiết lập.

Giai đoạn ổn định: Động cơ tiếp tục duy trì mức lực quay đã đạt được.

Giai đoạn giảm tốc: Động cơ giảm dần lực quay xuống một mức thấp hơn hoặc cho đến khi ngừng hẳn.

Do lực quay của động cơ tỷ lệ thuận với tốc độ quay của nó, tức là lực quay càng lớn thì tốc độ quay càng cao và ngược lại, nên trong chương này, lực quay và tốc độ quay của động cơ là cụm từ có thể hoán đổi nhau. Bạn đọc sẽ thấy khi thì tác giả dùng lực quay khi thì dùng vận tốc quay tùy theo ngữ cảnh, tuy nhiên về mặt ý nghĩa thì cả hai đều có thể áp dụng.

Bạn có thể dùng khối lệnh [Motor] để “tăng tốc” hoặc “giảm tốc” động cơ tùy theo robot đang hoạt động ở giai đoạn nào. Có một loạt các thông số sau đây cần phải được thiết lập trước khi bạn có thể dùng khối lệnh này.



Hình 12.18: Khởi lệnh [Motor] với các chốt dữ liệu vào/ra và Bảng cấu hình tương ứng.

Thông số [Port] xác định động cơ, hay nói chính xác hơn là cổng của khối vi điều khiển NXT mà bạn đang nối động cơ cần điều khiển vào. Trên Bảng cấu hình có 3 cổng A, B, C để bạn chọn. Nếu bạn dùng chốt dữ liệu vào [Port] để xác định cổng mà bạn cần dùng thì bạn cần truyền vào một trong các giá trị có kiểu dữ liệu Number như sau: 1 (A), 2(B), và 3(C).

Thông số [Direction] xác định hướng quay của động cơ. Bạn có thể chọn một trong các giá trị sau “Forward” (quay tới: theo chiều thuận), “Backward” (quay lùi: quay theo chiều nghịch), “Stop” (ngừng lại). Nếu bạn dùng chốt dữ liệu vào [Direction] để xác định hướng quay thì bạn có thể truyền vào một trong hai giá trị Logic như sau: True (Forward), False (Backward).

Thông số [Action] cho phép bạn điều khiển gia tốc (tốc độ thay đổi vận tốc) của động cơ. Nếu bạn chọn giá trị “Ramp Up” (tăng tốc) thì động cơ sẽ từ từ gia tăng lực quay của động cơ cho tới khi đạt đến mức đã định (thiết lập qua thông số [Power]). Nếu bạn chọn “Ramp Down” (giảm tốc) thì động cơ sẽ từ từ giảm lực quay về 0. Nếu bạn chọn “Constant” thì động cơ sẽ ngay lập tức quay với mức lực đã chọn hoặc dừng lại ngay. Bạn cũng có thể dùng chốt dữ liệu vào [Action] với các giá trị có kiểu dữ liệu Number sau đây để thiết lập thông số [Action]: 0 (Constant), 1 (Ramp Up), 2 (Ramp Down).

Thông số [Power] cho phép bạn ra lệnh cho động cơ hoạt động với một công suất cho trước (theo tỷ lệ % so với công suất tối đa). Nếu bạn sử dụng chốt dữ liệu vào [Power] để xác định tỷ lệ này thì bạn cần truyền vào một giá trị có kiểu dữ liệu Number nằm trong phạm vi từ 0 đến 100. Bạn cần phải thử nghiệm và chọn ra một tỷ lệ thích hợp để robot có thể hoạt động ổn định và thực hiện được công việc với hiệu suất tốt.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn

Nếu bạn chọn giá trị “Motor Power” của Thông số [Control], thì động cơ sẽ gia tăng thêm lực để có thể thắng các lực cản hoặc khắc phục hiện tượng trượt của động cơ đang làm cho nó không thể duy trì được tốc độ quay đã định. Tức là động cơ sẽ hoạt động ở công suất tối đa nhằm bảo đảm cho tốc độ quay luôn như nhau dù bị lực cản tại một số thời điểm. Bạn cũng có thể dùng chốt dữ liệu vào [Control Motor Power] với các giá trị Logic như True (chọn) và False (không chọn) để thiết lập thông số [Control].

Thông số [Duration] thiết lập khoảng thời gian hoạt động cho động cơ, hoặc thời gian cần có để động cơ có thể gia tốc đến vận tốc quay (lực quay) đã định qua thông số [Power]. Có hai phần cần phải xác định cho thông số này: phần đơn vị (Duration Type) có các đơn vị sau cho bạn lựa chọn (Unlimited: không bị giới hạn, Degrees: tính theo độ đo góc 0^0 - 360^0 , Rotation: tính theo vòng quay, Seconds: tính theo giây) và phần giá trị (Duration Value). Bạn cũng có thể dùng chốt dữ liệu vào [Duration] để xác định phần giá trị của thông số này.

Nếu bạn chọn giá trị “Wait for Completion” của thông số [Wait] thì chương trình sẽ chờ cho đến khi động cơ thực hiện xong lệnh do khối lệnh [Motor] gọi xuống trước khi chuyển qua một khối lệnh khác. Nếu bạn không chọn giá trị này thì chương trình chỉ gọi lệnh xuống cho động cơ (mà không cần chờ cho tới khi động cơ thực hiện xong lệnh này) và ngay lập tức chuyển sang thực hiện các khối lệnh tiếp theo. Bạn cũng có thể dùng chốt dữ liệu vào [Wait for Completion] với các giá trị Logic: True (chọn) hoặc False (không chọn) để thiết lập thông số [Wait]. Tuy nhiên thông số này cũng phụ thuộc vào phần đơn vị của thông số [Duration] trên Bảng cấu hình: nếu phần đơn vị này là “Unlimited” thì giá trị “Wait for Completion” của thông số [Wait] sẽ không được chọn, nếu nó là “Seconds” thì giá trị “Wait for Completion” của thông số [Wait] sẽ luôn được chọn.

Thông số [Next Action] cho phép bạn xác định động cơ sẽ làm gì sau khi thực hiện xong nhiệm vụ. Có 2 giá trị cho bạn lựa chọn: Brake (Thắng lại, động cơ ngừng ngay tại thời điểm nó thực hiện xong lệnh và cố gắng giữ nguyên vị trí đó), Coast (Thả cho động cơ tiếp tục quay theo quán tính). Bạn cũng có thể dùng chốt dữ liệu vào [Next Action] với các giá trị Logic: True (Brake) và False (Coast) để thiết lập thông số [Next Action].

Sau đây là bảng tóm tắt các chốt dữ liệu vào và ra của khối lệnh [Motor]:

Thông số/Chốt dữ liệu	Kiểu dữ liệu	Các giá trị hợp lệ	Ý nghĩa của các giá trị	Mục này không dùng ...
Port	Number	1-3	1 (A), 2 (B), 3 (C)	
Direction	Logic	True/False	True (Forward) False (Backward)	
Action	Number	0-2	0 (Constant) 1 (Ramp Up) 2 (Ramp Down)	Phần đơn vị của thông số [Duration] trên Bảng cấu hình có giá trị là “Unlimited” hoặc “Seconds”.
Power	Number	0-100		
Control Motor Power	Logic	True/False	True (chọn) False (không chọn)	



Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

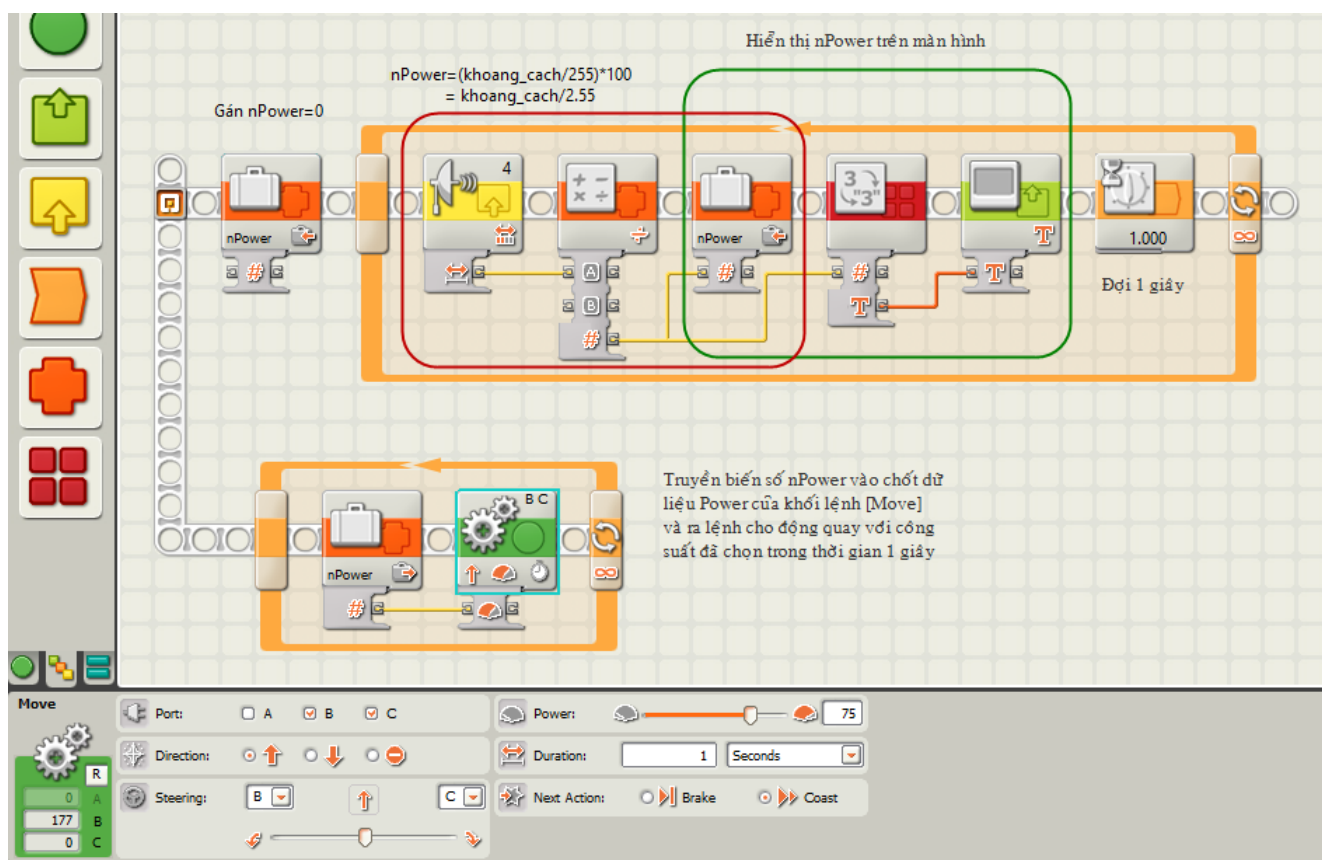
Duration	Number	0-2147483647	Tùy thuộc vào phần đơn vị của thông số [Duration] trên Bảng cấu hình: Degrees: độ đo góc Rotation: vòng quay Seconds: giây	Phần đơn vị của thông số [Duration] trên Bảng cấu hình có giá trị là “Unlimited”
Wait for Completion	Logic	True/False	True: chương trình đợi cho đến khi động cơ thực hiện xong lệnh [Motor] rồi mới thực hiện các lệnh tiếp theo. False: không đợi	Phần đơn vị của thông số [Duration] trên Bảng cấu hình có giá trị là “Unlimited” (luôn có giá trị là False), hoặc “Seconds” (luôn có giá trị là True)
Next Action	Logic	True/False	True (Brake) False (Coast)	Phần đơn vị của thông số [Duration] trên Bảng cấu hình có giá trị là “Unlimited” hoặc [Wait for Completion] = False
Direction out	Logic	True/False	Hướng quay của động cơ: True (Forward) False (Backward)	
Degrees out	Number	0-2147483647	Động cơ đã quay được bao nhiêu độ sau khi thực hiện khối lệnh [Motor]	

Bảng 12.2 : Các chốt dữ liệu của khối lệnh [Move].

Sau đây ta sẽ thiết kế một đoạn chương trình đơn giản dùng khối lệnh [Move] và cảm biến siêu âm để điều khiển vận tốc quay của động cơ. Chương trình sẽ liên tục (mỗi giây một lần) đo khoảng cách từ đầu dò của cảm biến đến chướng ngại vật. Chúng ta muốn là khoảng cách càng gần thì động quay càng chậm (khoảng cách càng xa thì động cơ quay càng nhanh). Do đó, ta sẽ áp dụng công thức sau để tính tỷ lệ công suất cần dùng cho động cơ:

$$nPower = \frac{\text{khoảng_cách}}{\text{khoảng_cách_tối_đa}} * 100\%$$

Để điều khiển động cơ, chúng ta cũng liên tục (mỗi giây một lần) dùng khối lệnh [Move] để ra lệnh cho động cơ quay với công suất đã tính được ở trên. Ta dùng biến số nPower để lưu tỷ lệ công suất cần dùng và truyền giá trị của nPower cho chốt dữ liệu vào [Power] của khối lệnh [Move] để yêu cầu các động cơ quay đúng công suất này. Như vậy vận tốc quay của động cơ sẽ được điều chỉnh liên tục theo khoảng cách đo được mỗi lần.



Hình12.19: Đoạn chương trình điều khiển tốc độ quay của động cơ dùng cảm biến siêu âm.

PHẦN 6 : THIẾT KẾ ROBOT

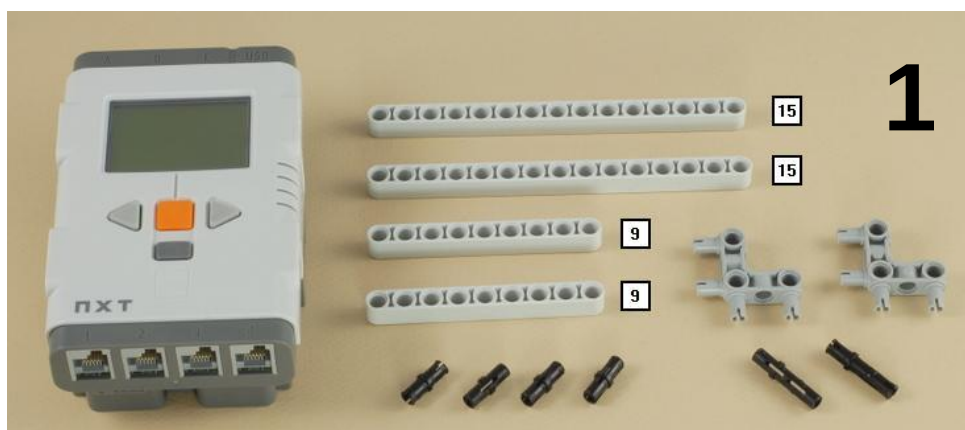
CHƯƠNG 13 : THIẾT KẾ VÀ LẬP TRÌNH ROBOT

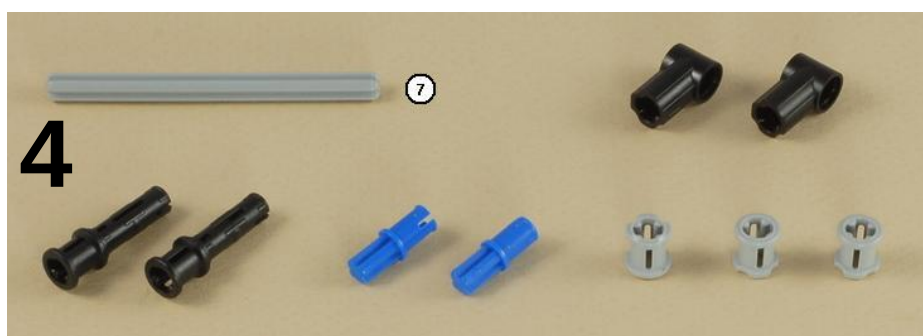
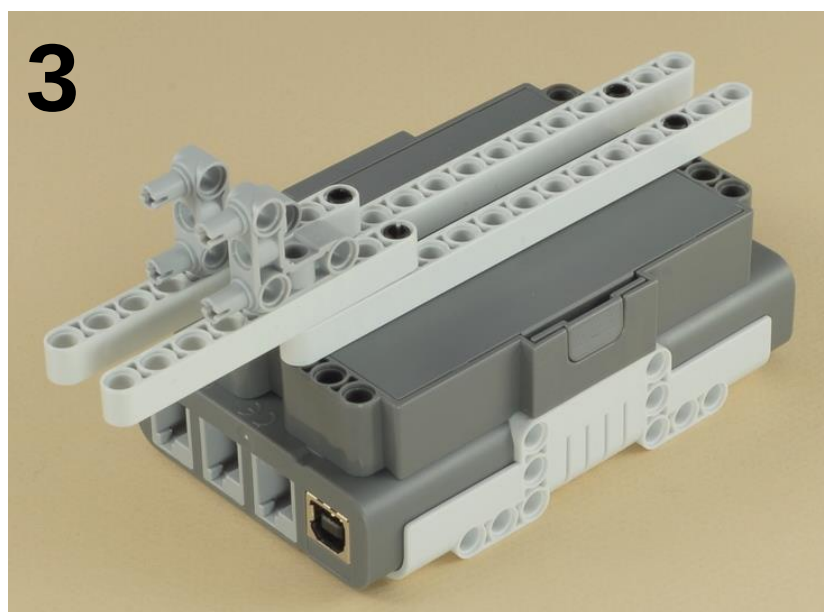
I. Giới thiệu :

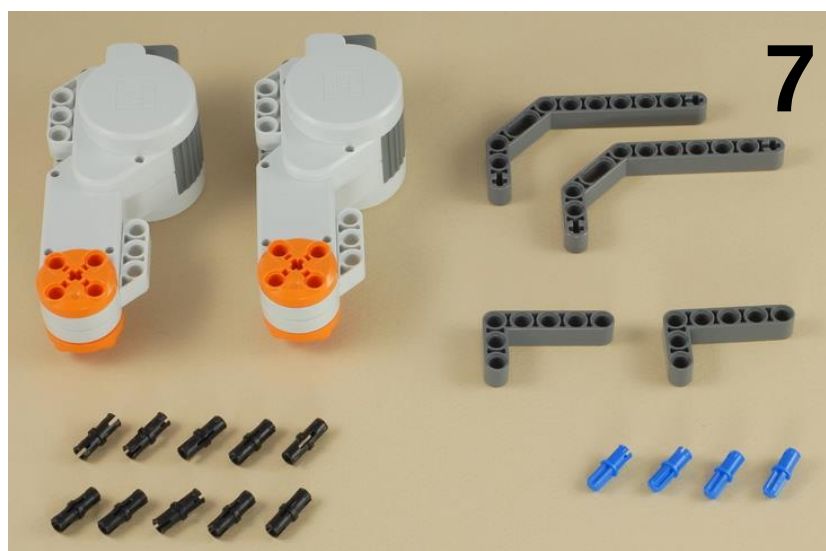
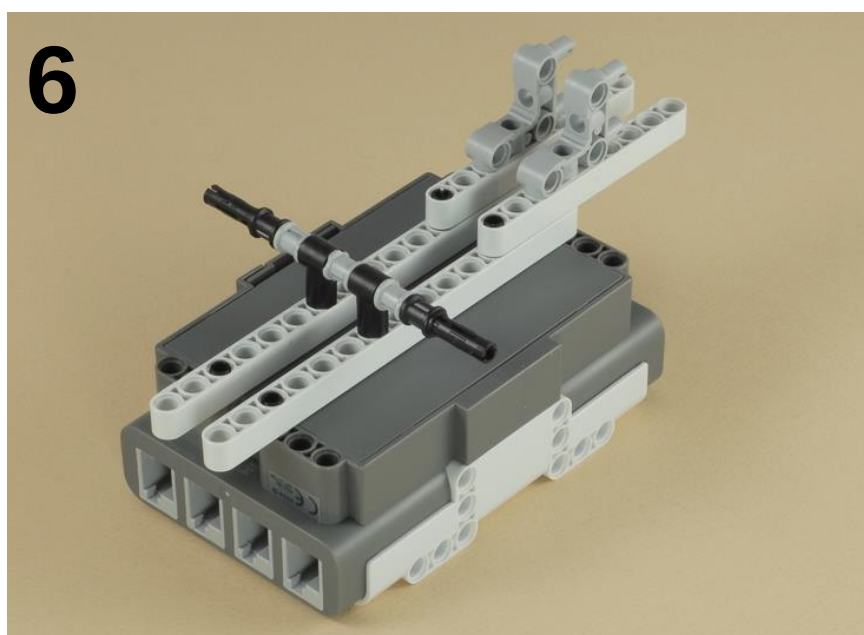
Để xây dựng Robot thông thường phải qua một loạt các bước, ban đầu là thiết kế, sau đó lắp ráp, và cuối cùng là lập trình và thử nghiệm. Thông thường, sự phát triển bao gồm rất nhiều những thử nghiệm và sai lệch, sửa chữa và sửa đổi. Khi bạn lắp ráp, hãy chắc chắn cấu trúc đủ mạnh mẽ và hoạt động được, ngoài ra, phương pháp lắp ráp thành từng phần nhỏ sau đó kết hợp lại với nhau cũng rất hữu ích.

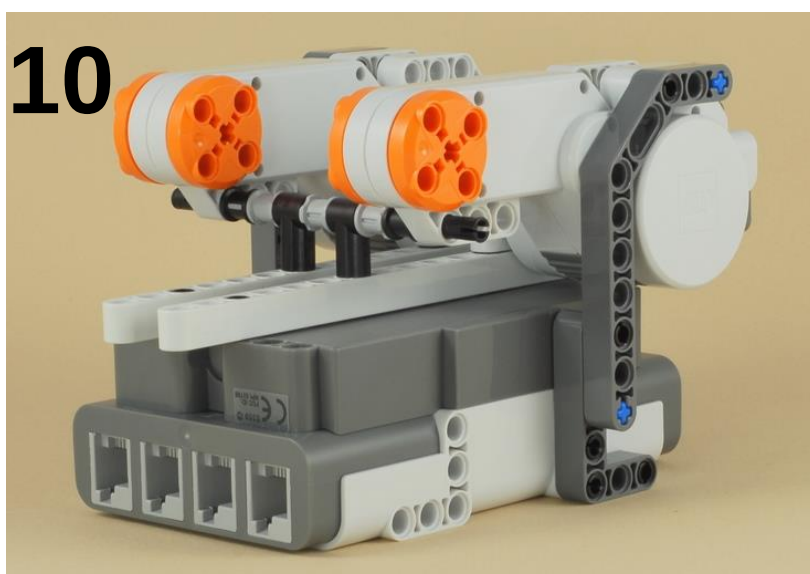
II. Robot cơ bản Castor Bot

II.1 Lắp ráp



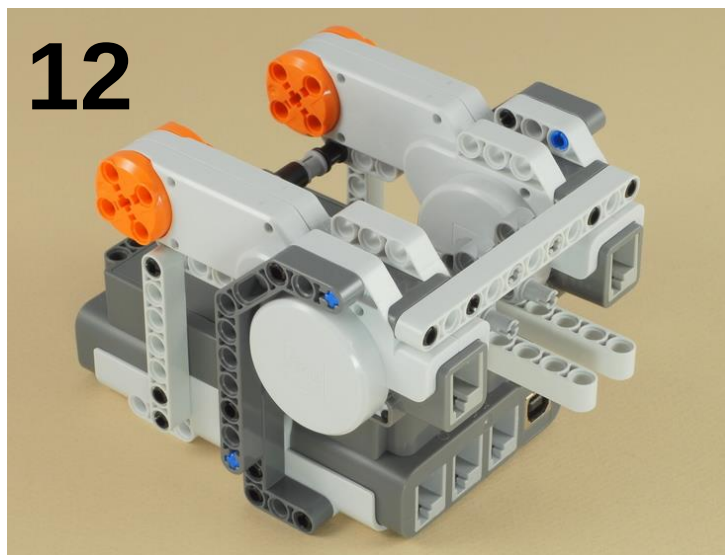




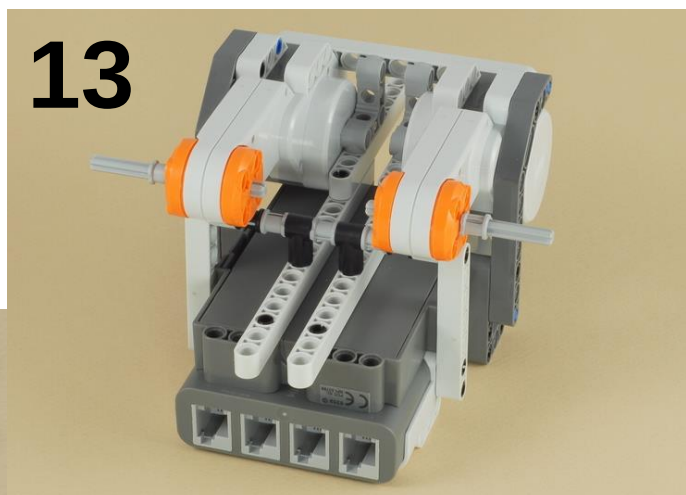
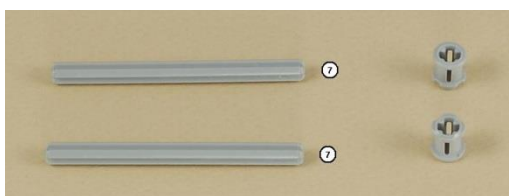


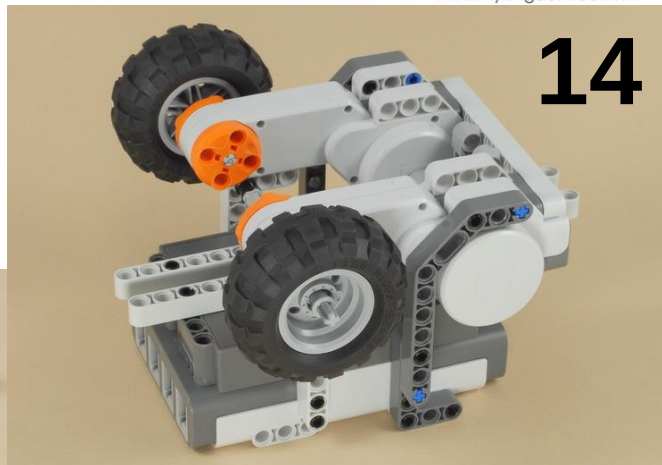


Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT: (08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



13



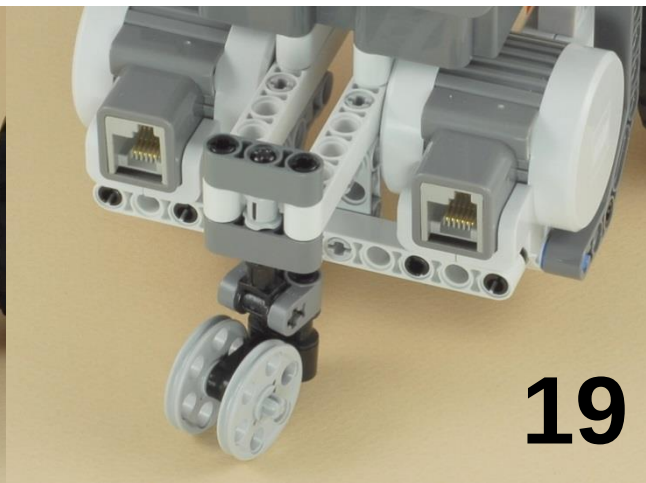
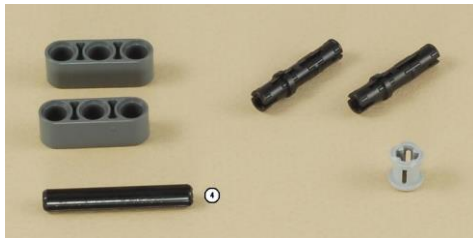


15



17





II.2 Thêm cảm biến ánh sáng vào như hướng dẫn :



III. Lập trình cho Robot hành động đơn giản :

Bây giờ Robot đã được lắp ráp xong, chúng ta sẽ tiến hành lập trình cho Robot thực hiện những hành động sau :

- Tiến về trước.



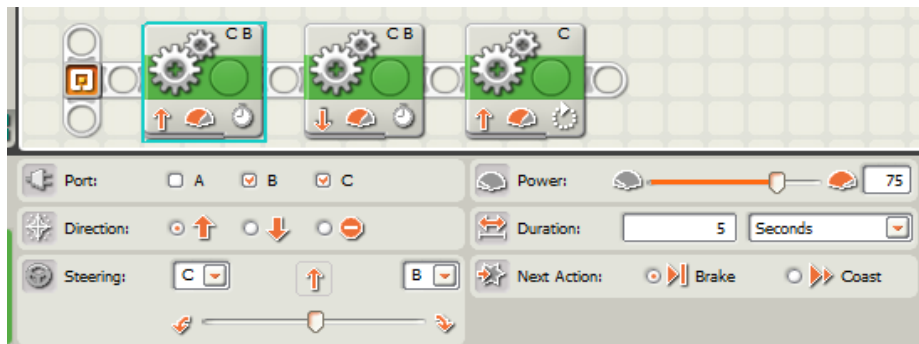
Công Ty Cổ Phần Việt Tinh Anh
 205, đường 9A, Khu dân cư Trung Sơn,
 Bình Hưng, Bình Chánh, TP.HCM
 ĐT:(08) 543 18717/18 - Fax: (08) 54318717
 Website: www.viettinhanh.com.vn
 www.mykingdom.com.vn

- Lùi về sau.
- Quay sang bên phải.
- Phát ra âm thanh
- Hiện thị lên màn hình.

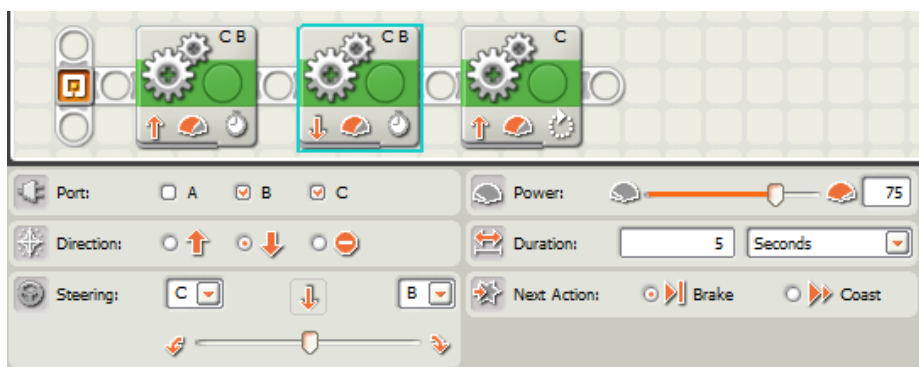
Chương trình NXT-G sẽ được tạo ra trong ba giai đoạn riêng biệt. Giai đoạn đầu, cho Robot di chuyển. Tiếp theo, khối âm thanh và hiển thị sẽ được thêm vào chương trình NXT-G để cho Robot có thể giao tiếp, đến đây là hoàn thành việc phát triển chương trình cho Robot, Robot sẽ được vận hành để kiểm tra. Sau cùng là ghi chú vào các ô lập trình để mô tả được Robot thực hiện nhiệm vụ gì trong từng thời điểm cụ thể.

Bước 1 : Làm cho Robot di chuyển.

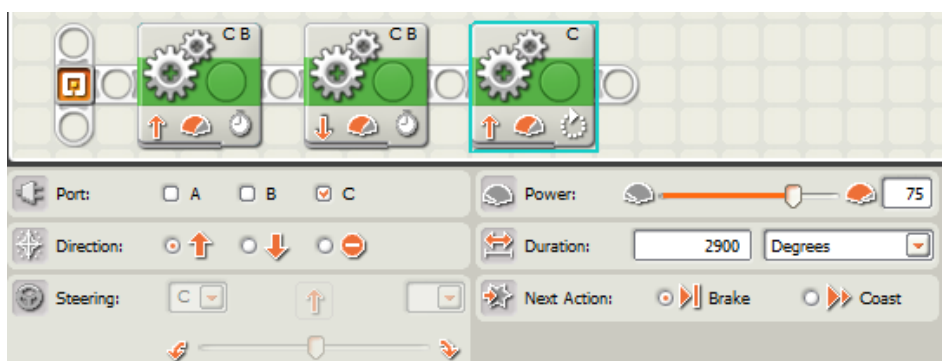
Đặt khối Move thứ nhất trên thanh lập trình, và cấu hình như sau :



Cấu hình cho khối Move thứ hai :



Cấu hình cho khối Move thứ ba :



Bước 2 : Làm cho Robot phát ra âm thanh và hiển thị lên màn hình, thực hiện theo các bước sau :

- Nói “Hello” khi bắt đầu.
- Hiển thị dòng chữ “Hi. I am Castor Bot” trên màn hình của NXT.
- Dừng lại trong 2s.
- Phát một âm thanh “Tone” trước mỗi hành động di chuyển.
- Nói “Goodbye”
- Hiển thị dòng chữ “Goodbye” trên màn hình của NXT.

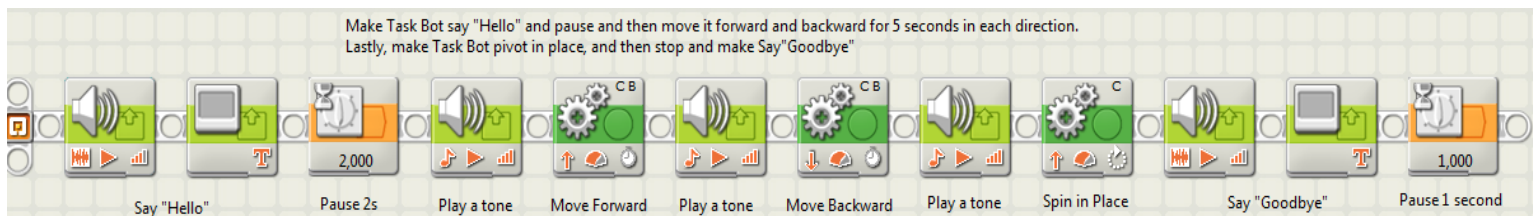
Chương trình sẽ tương tự như sau :



Trước khi tiến hành ghi chú, nạp chương trình và kiểm sự vận hành của Robot có đúng với yêu cầu đặt ra hay không.

Bước 3 : Ghi chú trên NXT-G

Sau khi kiểm chứng khả năng vận hành của Robot trong thực tế, chúng ta tiến hành thực hiện việc ghi chú trên cửa sổ lập trình :



IV. Lập trình Robot dò đường :

Chúng ta sẽ lập trình để Robot dò theo đường định sẵn như hình bên dưới :

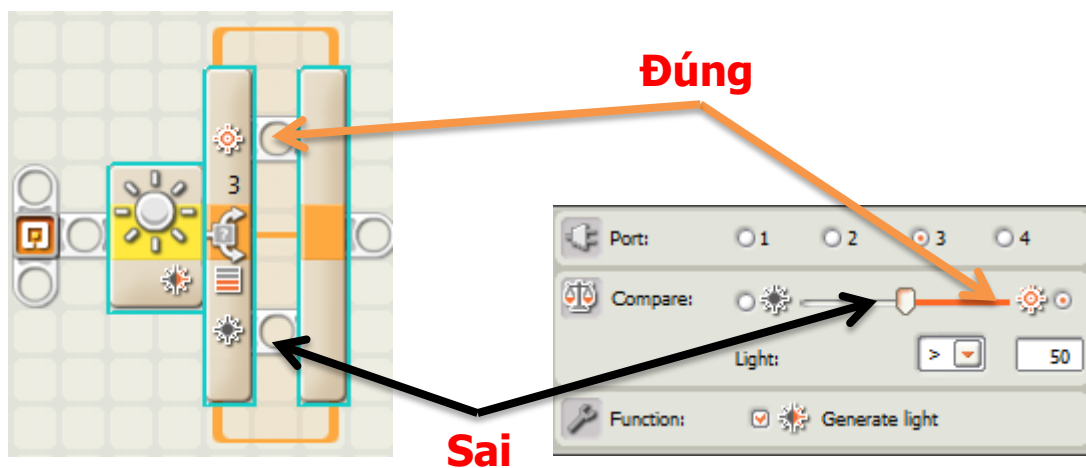


**Robot dò
đường này**

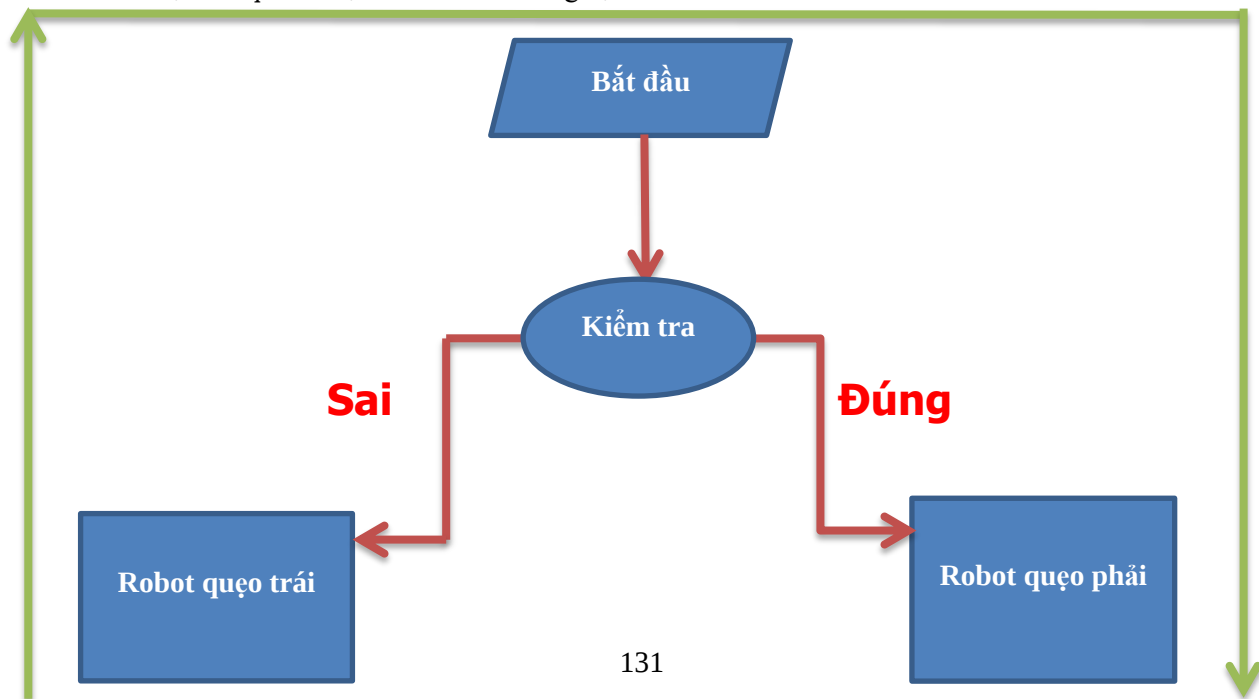


1. Mở NXT, chọn biểu tượng Reflected light. Chọn cổng mà bạn cắm dây vào (VD : 3).
2. Giữ cảm biến thật gần bề mặt mà bạn muốn đo, ở đây ta đo màu đen và màu trắng.
3. Đọc số liệu hiển thị trên màn hình NXT.

Ta lấy số trung bình cộng giữa đen và trắng (Vd : trắng (70), đen(30), trung bình =50). Nhập số này vào ô điều kiện để chọn lựa.

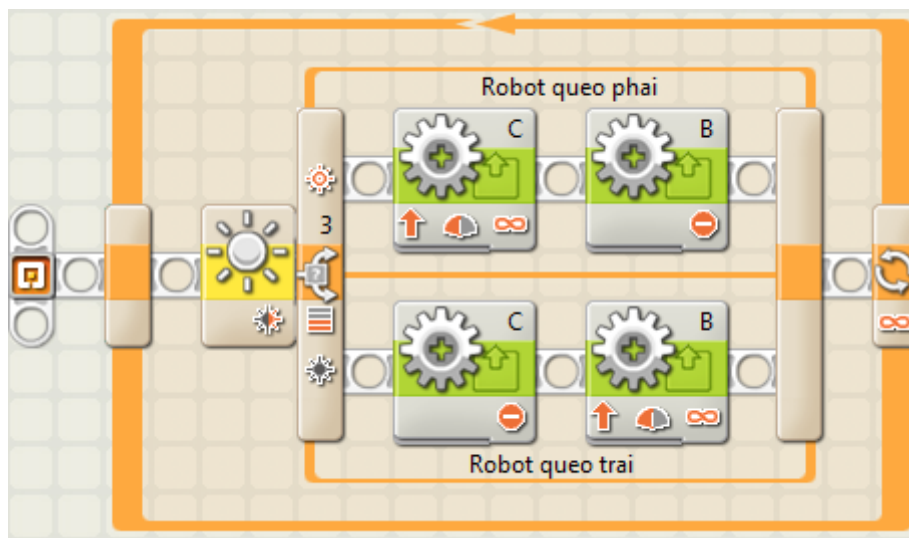


Về giải thuật dò đường : Robot sẽ dò theo đường giao nhau giữa đen và trắng. Một cách đơn giản, khi cảm biến ánh sáng đọc được màu đen, cho Robot quẹo qua khu vực màu trắng, và ngược lại. Như vậy, Robot sẽ liên tục lắc qua lắc lại để dò theo đường định sẵn.

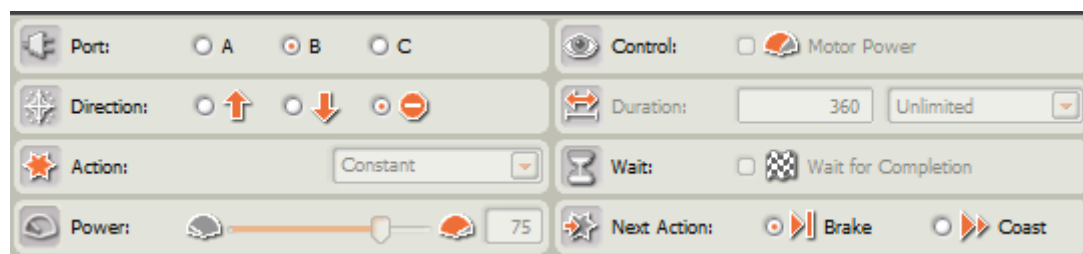
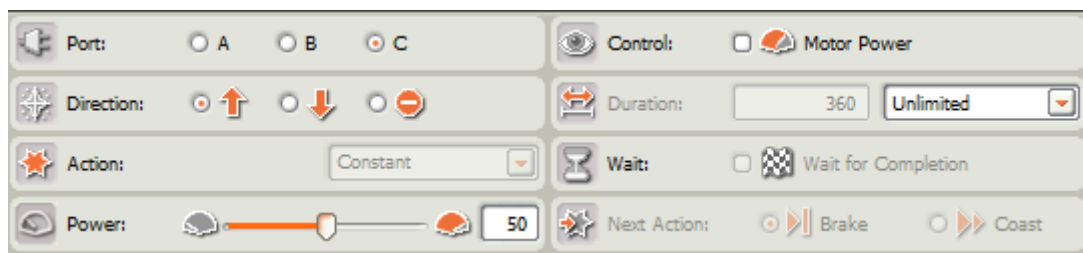


Lặp lại mãi mãi

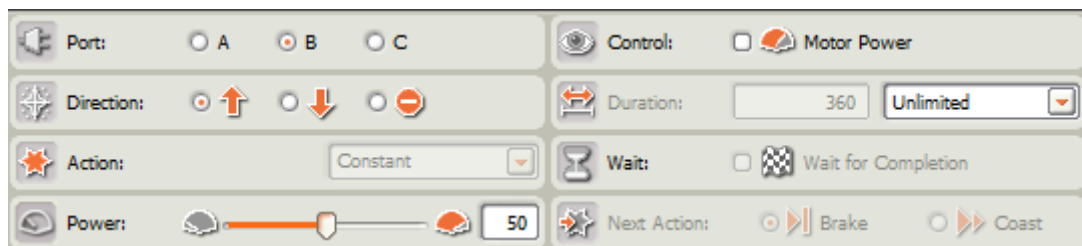
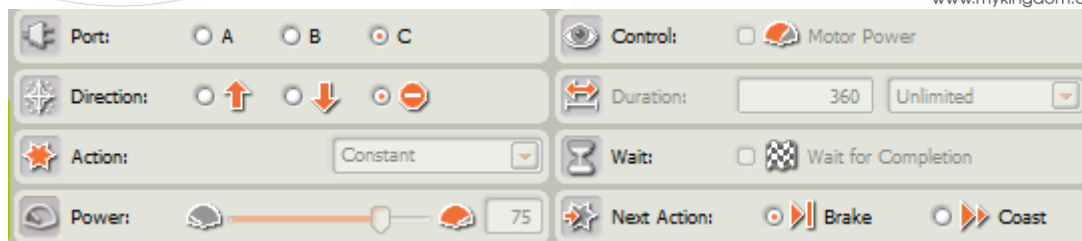
Sử dụng một vòng lặp với điều kiện lặp là mãi mãi để liên tục kiểm tra cảm biến, chọn hướng di chuyển phù hợp. Xác lập các thuộc tính của động cơ giống bảng bên dưới.



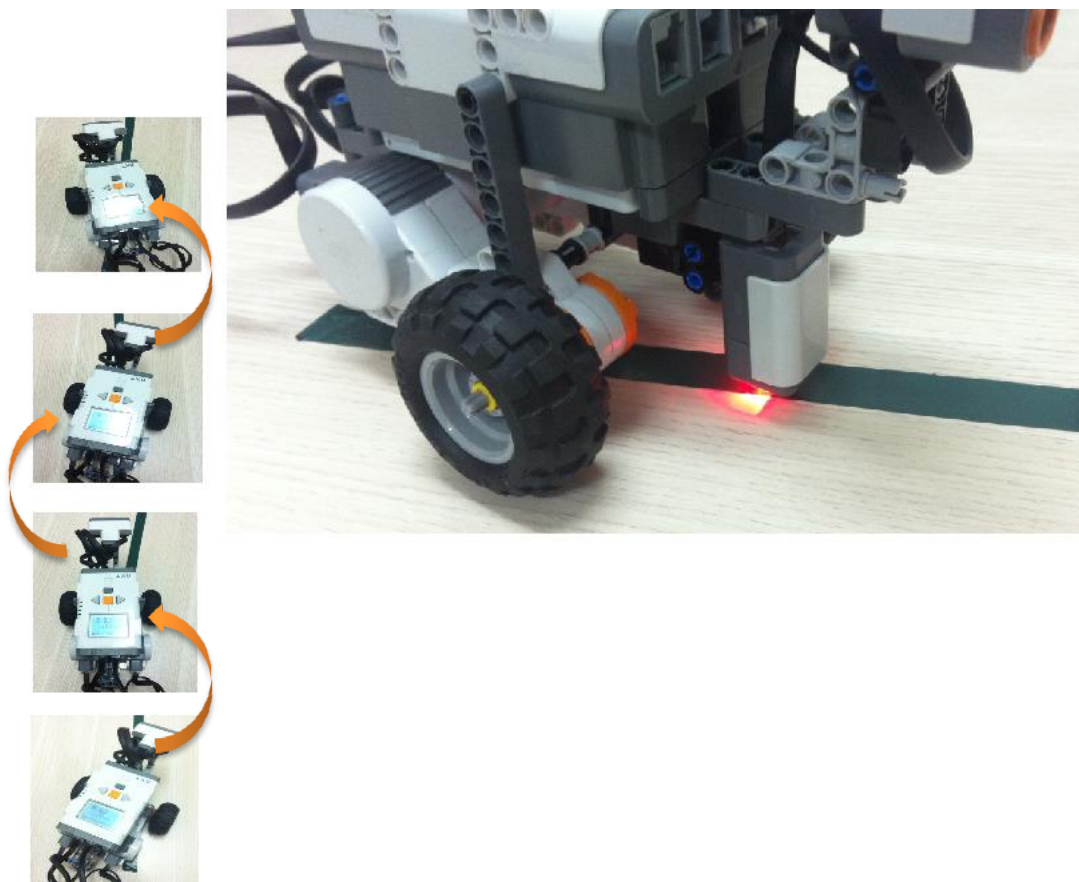
Cấu hình cho Robot queo phải khi cảm biến phát hiện vùng màu trắng.



Cấu hình cho Robot queo trái khi cảm biến phát hiện vùng màu đen :



Kết quả Robot sẽ di chuyển như sau :



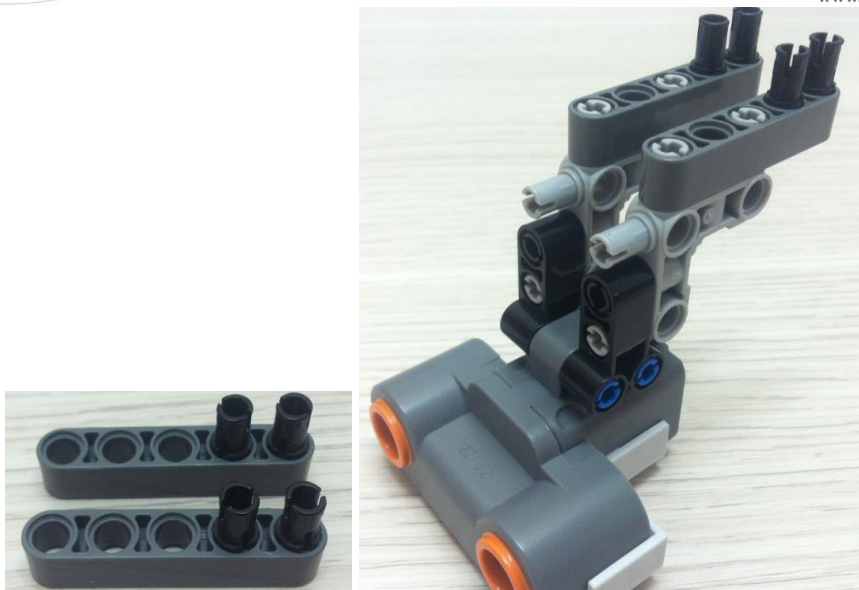
VI. Lập trình Robot dò đường và tránh chạm vật cản.

VI.1 Lắp thêm cảm biến siêu âm





Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



VI.2 Lập trình Robot dò đường và dừng lại khi phát hiện vật cản :

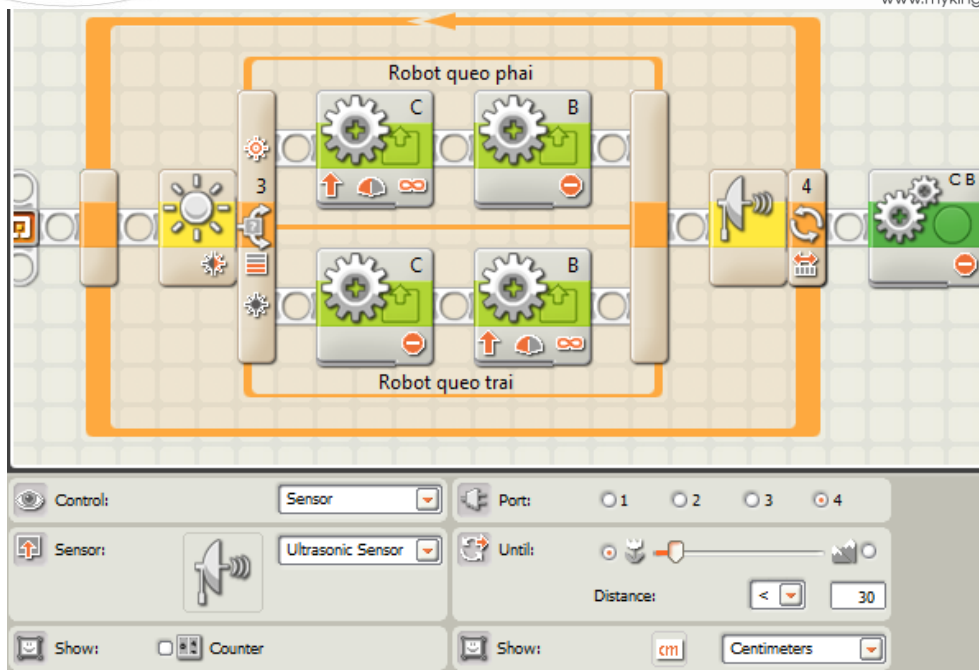
Ở phần này chúng ta sẽ đặt một vật cản trên đường di chuyển của Robot(vật cản phải đủ cao và rộng để cảm biến siêu âm có thể nhận biết được). Ở đây sử dụng thùng đựng 9797 để làm vật cản.



Công Ty Cổ Phần Việt Tinh Anh
205, đường 9A, Khu dân cư Trung Sơn,
Bình Hưng, Bình Chánh, TP.HCM
ĐT: (08) 543 18717/18 - Fax: (08) 54318717
Website: www.viettinhanh.com.vn
www.mykingdom.com.vn



Tiếp tục sử dụng giải thuật dò đường như trên, tuy nhiên, ở điều kiện kết thúc vòng lặp, sử dụng cảm biến siêu âm, cấu hình như sau :



Như vậy, Robot sẽ vừa dò đường vừa liên tục kiểm tra khoảng cách từ dữ liệu nhận được của cảm biến siêu âm, cho đến khi khoảng cách từ đầu thu cảm biến siêu âm đến vật cản đạt xấp xỉ 30cm. Vòng lặp bị thoát ra và Robot dừng lại.

