Dongbu Robot

DR Visual Logic

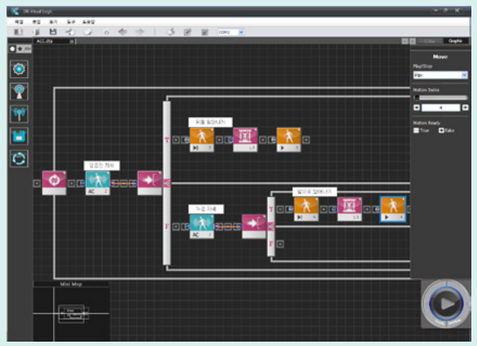
for HOVIS Lite/Eco(plus)

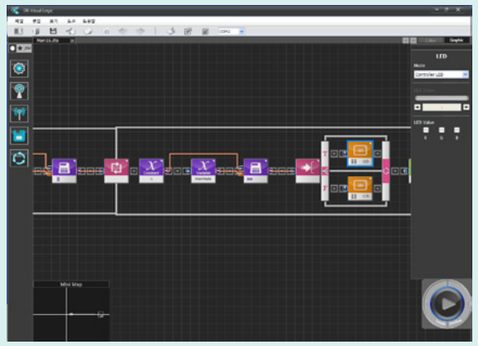
- DR-Visual Logic 설치하기
- Hello DR-Visual Logic
- DR-Visual Logic 사용자 인터페이스
- 프로그래밍 모듈
- 속성창
- 컴파일 / 다운로드 하기
- 다양한 기능
- 모듈별 예제 프로그래밍

www.dongburobot.com

DR Visual Logic
For Hovis Lite, Eco Plus

CONTENTS





01 설치하기

DR-Visual Logic 소개

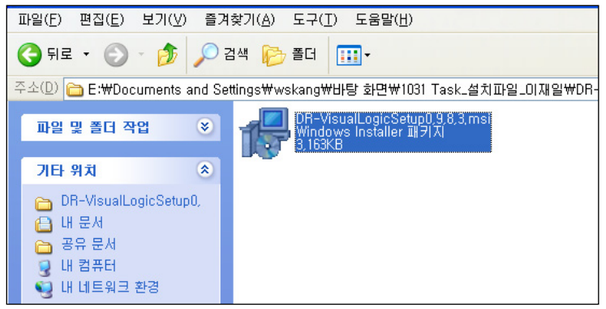
DR-Visual Logic 은 동부로봇에서 개발된 고유의 로봇 프로그래밍 언어를 Graphic 모듈화시켜 Drag & Drop 방식을 채용한 로봇 프로그래밍 툴입니다. 동부로봇의 제어기인 DRC 의 기능에 맞춰서 각각의 기능을 모듈화 시켜놓았습니다.

마우스만을 활용하여 드래그 앤 드랍 방식으로 초보자도 쉽게 프로그래밍 할 수 있는 구조로 되어있으며, C-like 탭을 제공하여, 그래픽 프로그래밍이 Text 로 변환되는 코드를 바로 확인 할 수 있습니다. C 문법 과 유사한 코드이기 때문에 프로그래밍 입문자가 C 문법을 익히는데 많은 도움이 될 수 있습니다. 로봇 언어중에 가장 쉽고, 가장 강력한 기능을 발휘하는 프로그래밍 툴로서 초급자부터 고급자까지 그 활용폭이 넓어서, 로봇 교육 시장에서 가장 주목받는 로봇 언어로 사용되어지고 있습니다.

앞으로 업그레이드된 제어기 관련 모듈 추가 및 로봇 모션의 모듈화 및 시뮬레이션 결합 등으로 다채로운 기능을 발휘하는 프로그램으로 업그레이드 할 예정입니다.

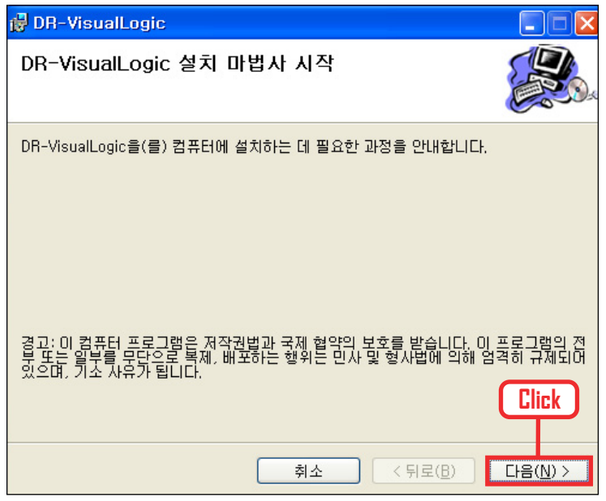
- 시스템 요건
- 최소 인텔 팬티엄 800 Mhz
- Windows XP , Window 7
- 최소 256 MB RAM
- 하드디스크 설치 공간 약 300 MB 필요
- USB Port

설치하기 / 설치부터 실행까지 따라해보세요



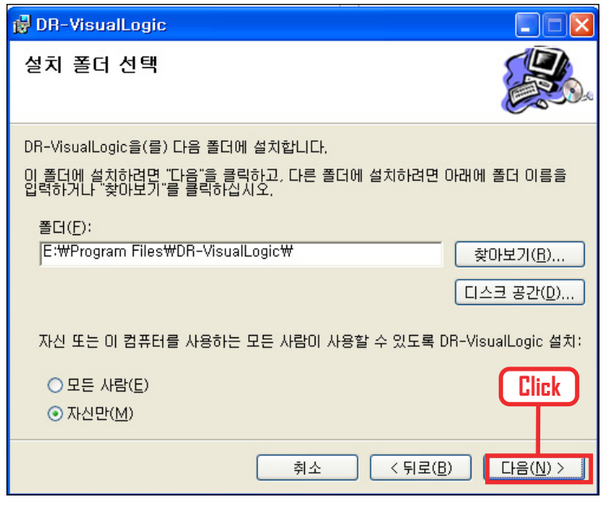
01 설치파일

설치파일을 클릭합니다.



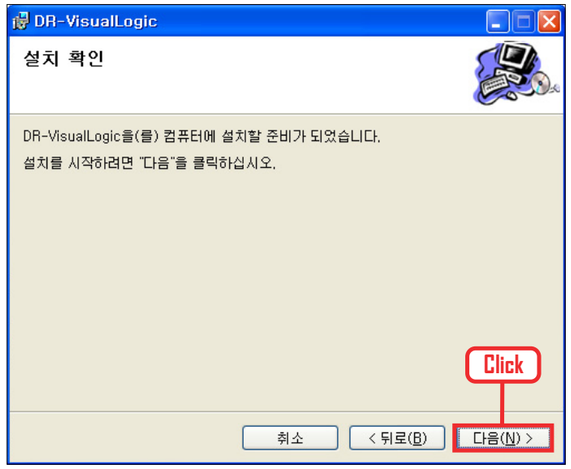
02 설치 마법사 시작

"다음" 버튼을 클릭합니다.



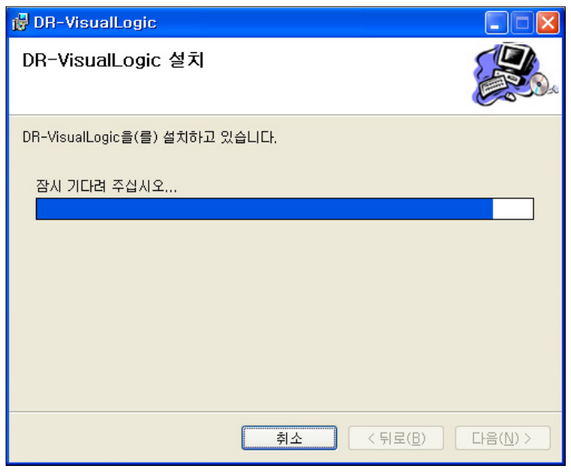
03 설치 폴더를 선택

"다음" 버튼을 클릭합니다.



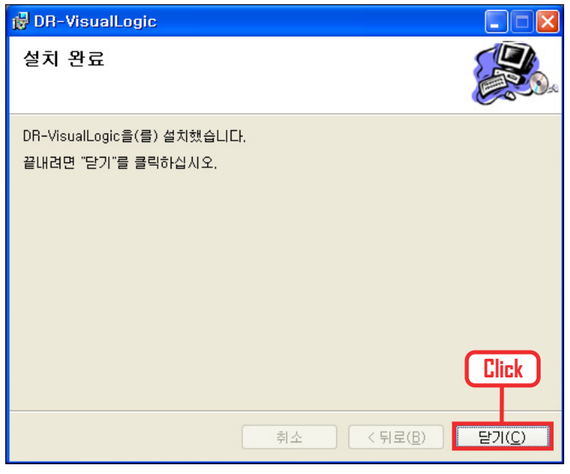
04 설치 확인

"다음" 버튼을 클릭합니다.



05 설치 시작

설치 시작 합니다. 프로그레스 바가 끝날 때 까지 기다려주세요



06 설치 완료 선택

"닫기" 를 클릭하세요
소프트웨어 설치가 완료되었습니다.

07 실행파일 확인

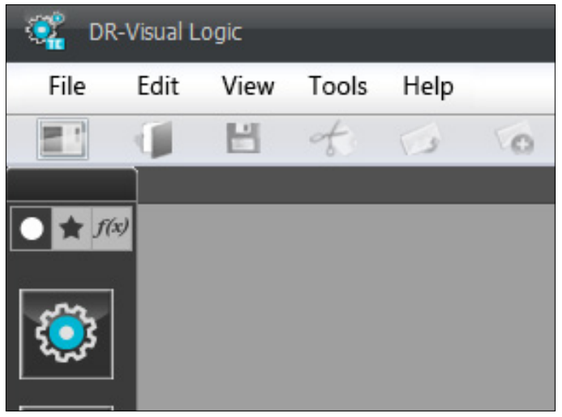
바탕화면과 Windows 시작 〉 모든프로그램 〉 Dongbu Robot 〉 DR-Visual Logic 에서 실행파일을 확인하세요
실행파일을 클릭하면 프로그램이 실행됩니다.

Hello DR-Visual Logic

02

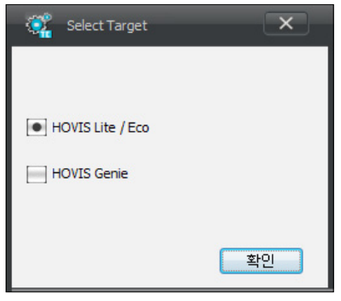
첫 번째 프로그램 따라하기

예제설명 : 양팔 벌려있는 로봇의 모터를 한쪽 팔만 차려 자세로 내려봅니다. 16축 휴머노이드 로봇의 모터들을 모두 중앙정렬 시키면, 로봇은 양 팔이 좌우로 팔 벌려 자세가 됩니다. 그 중 한쪽 팔을 차려 자세로 내려봅니다.



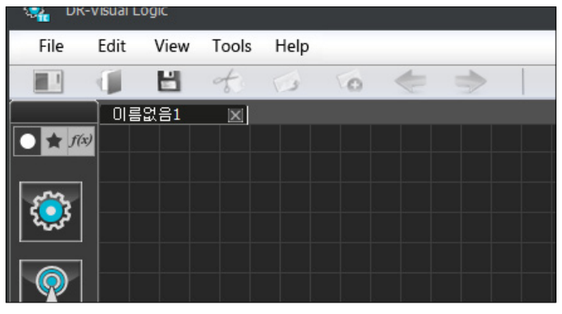
01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭해서 새로운 dts 파일을 만듭니다.



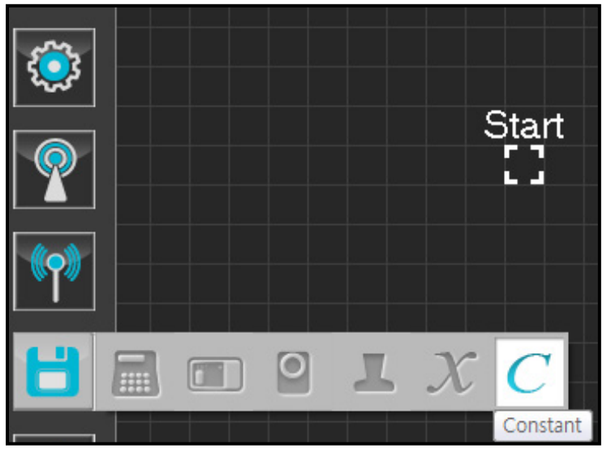
02 대상 로봇 고르기

새로운 dts 파일을 만들 때는 이 파일이 어떤 로봇에 사용될 프로그램인지 골라야 합니다. 현재 제어할 로봇에 해당 되는 HOVIS Lite/Eco를 선택합니다.



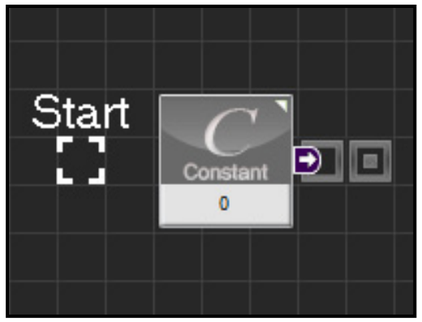
03 새 창

새로운 파일이 생성되었습니다.



04 모듈 선택

모듈을 배치하기 위해서는 왼쪽 모듈바에서 모듈을 클릭해야 합니다.
Data 〉 Constant 모듈을 클릭합니다.

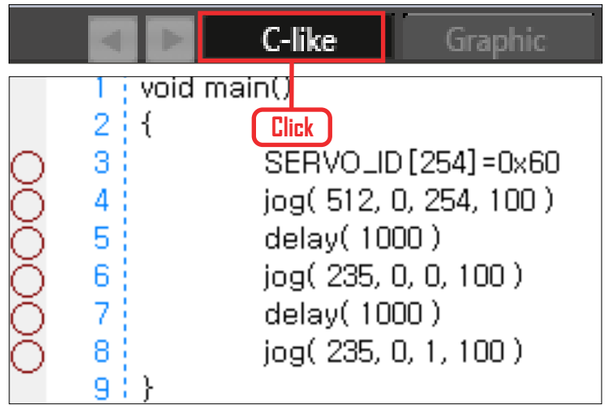


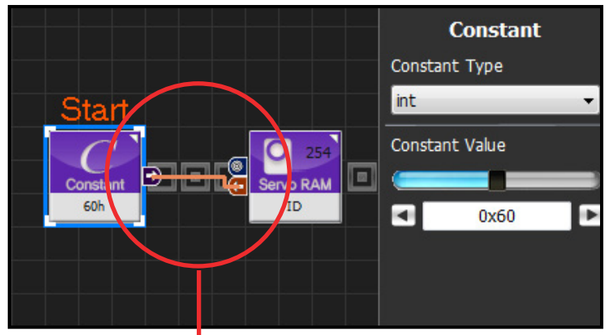
05 모듈 배치하기

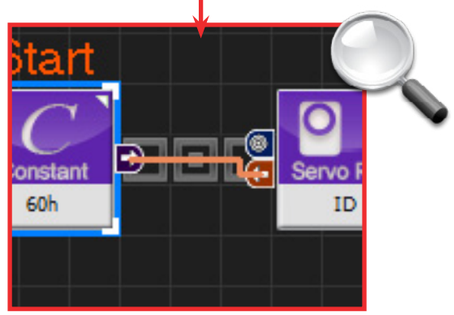
모듈을 클릭하면 마우스 커서를 따라 새로운 모듈이 움직입니다. 다시 마우스를 클릭하면 현재 위치에 모듈이 배치됩니다. 모듈을 드래그해서 Start Point로 옮겨보세요.

06 프로그래밍 시작

모듈을 옮겨서 Start Point에 정확히 도킹하면 왼쪽과 같이 활성화된 컬러 이미지 모듈로 변합니다.
이것은 이 모듈이 활성화 되어 프로그램에 반영되고 있다는 의미입니다.







07 전체 프로그래밍

로봇의 모터를 움직여 한쪽 팔을 내리는 전체 프로그래밍 전개 화면입니다.

08 C-Like 보기

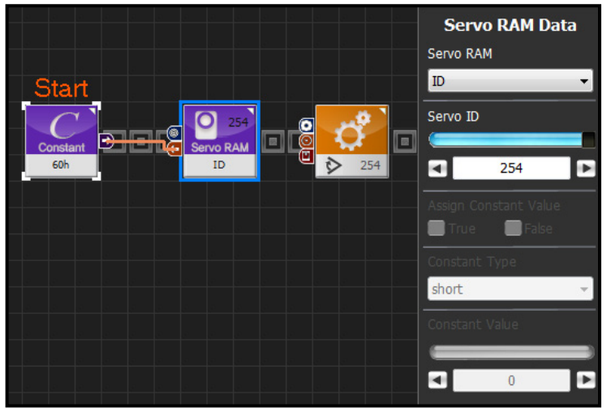
오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

전체 프로그램 소스 화면입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.

09 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

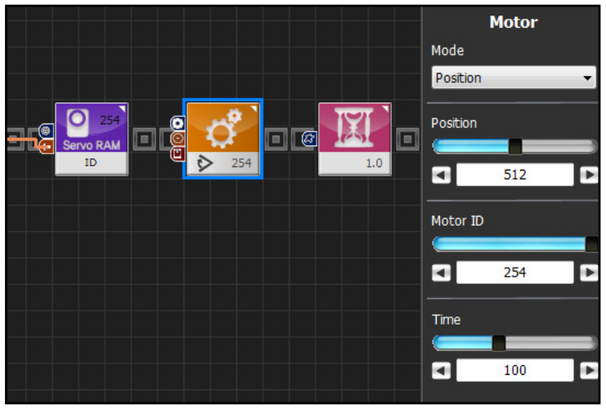
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다.
이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



10 모든 서보에 적용

앞에서 받은 0x600이라는 상수 값을 모든 서보에 적용하는 과정입니다.

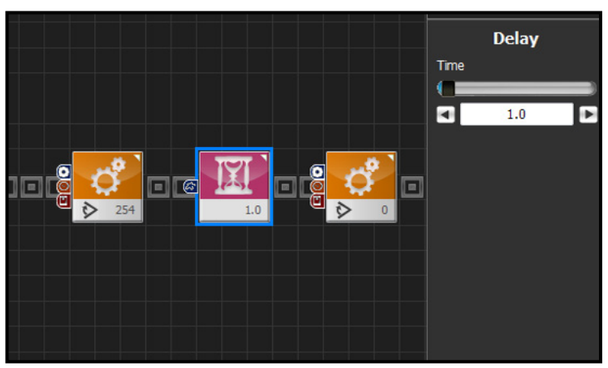
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



11 모든 서보 모터 위치 제어

모든 서보 모터의 위치를 중앙에 보내는 과정입니다. 모든 로봇의 모터의 각도를 중앙으로 정렬 하면 휴머노이드에서는 팔을 좌우로 뻗게 됩니다.

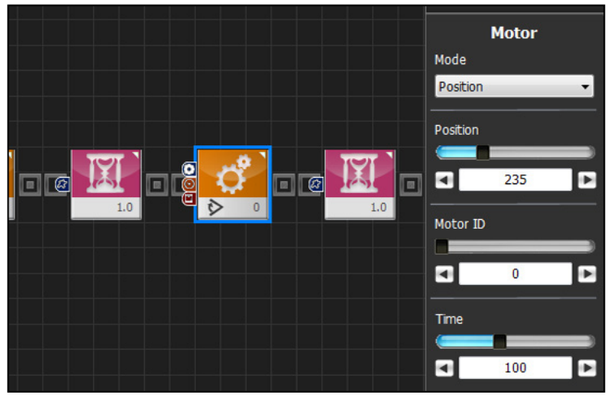
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 512로 설정합니다. 512는 모든 모터의 영점 위치로, 모터를 모두 중앙으로 보낸다는 의미입니다.
Motor ID : 254로 설정합니다. 254는 모든 모터에 적용하겠다는 의미입니다.
Time : 100으로 설정합니다. 단위는 1당 11.2ms로, 100은 약 1.12초를 의미합니다. 1.12초 동안 원하는 위치로 보낸다는 의미입니다.



12 Delay

다음 모터 동작 전에 1초를 기다린 후 시작하는 설정입니다. 직전 모듈에 의해 모터가 움직이는 동안 아무 일도 하지 않고 기다립니다.

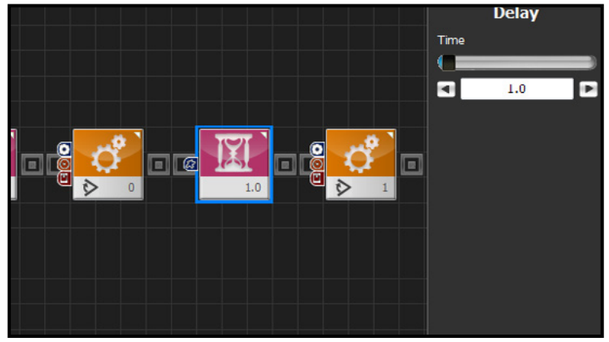
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.0으로 설정합니다. 1초 동안 기다리겠다는 의미입니다.



13 모터 0번 (오른쪽 어깨) 설정

한쪽 팔을 차려 자세로 되돌려 놓기 위해 오른쪽 어깨를 돌리는 명령입니다.

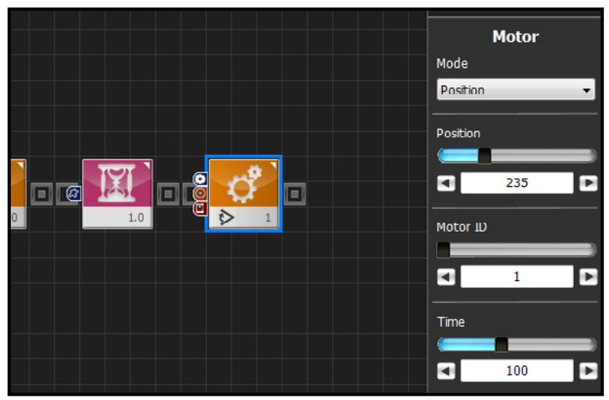
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 수평으로 들고 있던 오른팔을 수직으로 내려갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



14 Delay

다음 모터 동작 전에 1초를 기다린 후 시작하는 설정입니다. 직전 모듈에 의해 모터가 움직이는 동안 아무 일도 하지 않고 기다립니다.

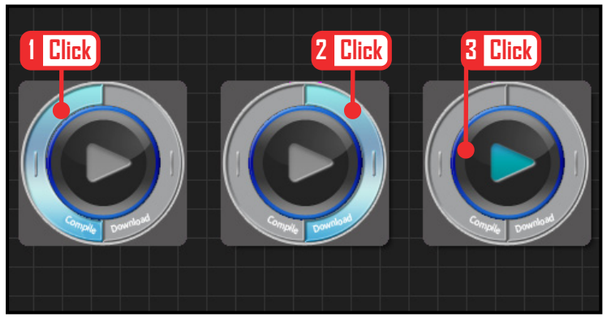
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.0으로 설정합니다. 1초 동안 기다리겠다는 의미입니다.



15 모터 1번 (오른쪽 팔) 설정

오른쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 90도로 들고 있던 오른팔을 수직으로 내리는 위치 값입니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.

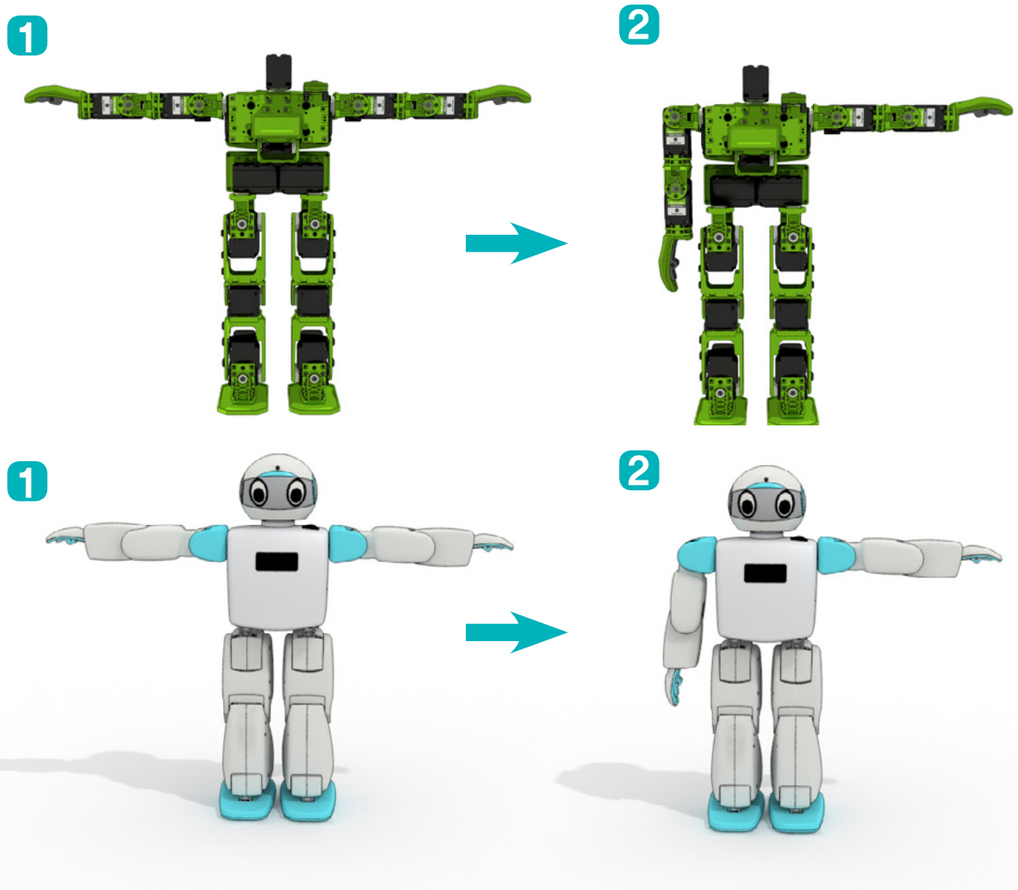


16 다운로드

프로그래밍 후 컴파일 –› 로봇에 다운로드 –› 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

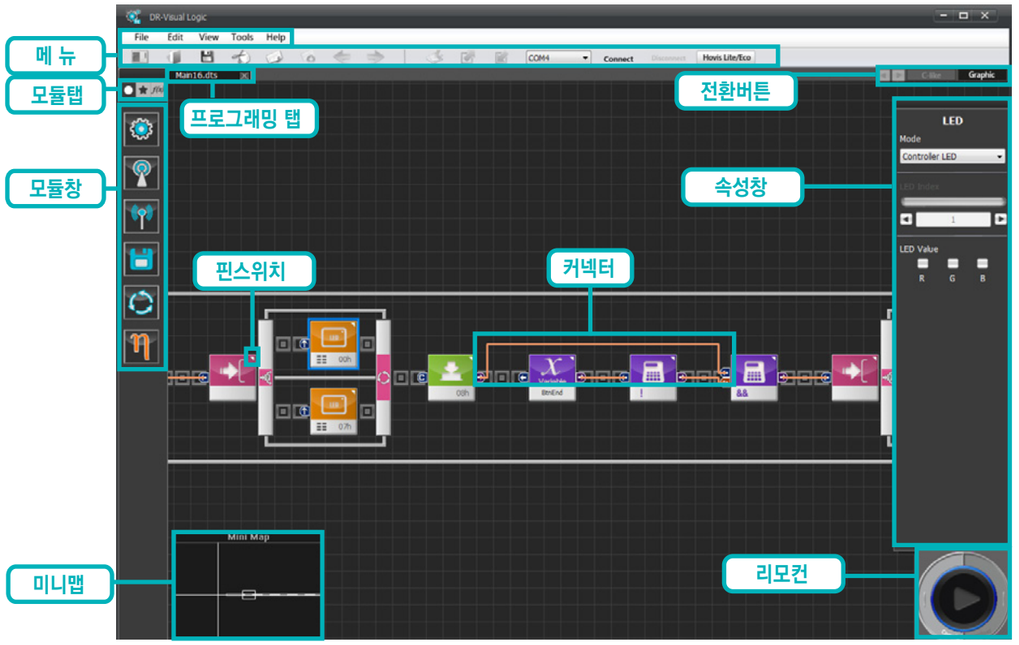
Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



17 로봇동작

양팔 벌려 수평으로 유지하고 있던 로봇 팔중에 오른쪽 팔이 아래로 내려갑니다.

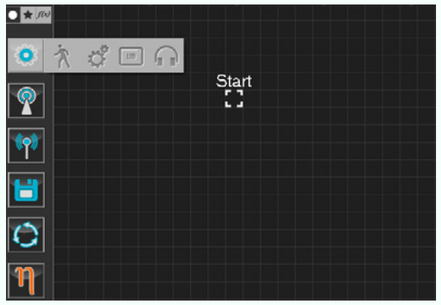
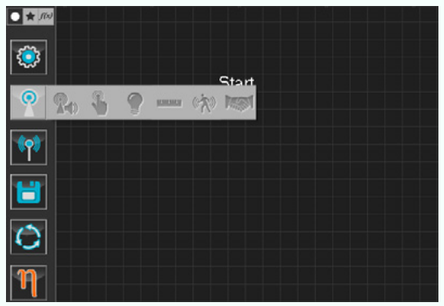
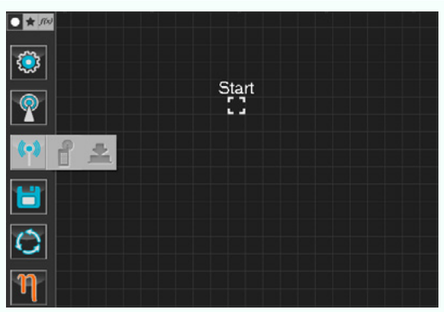
03 사용자 인터페이스

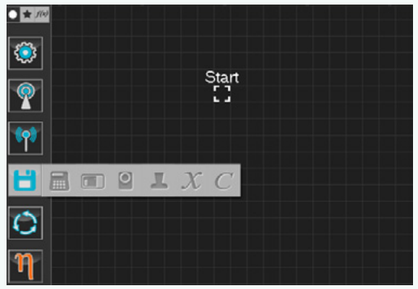
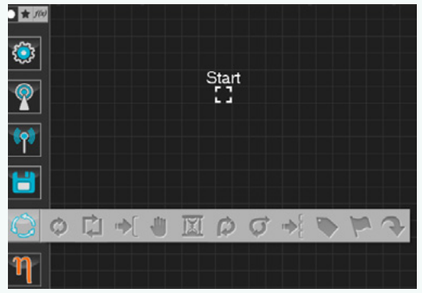
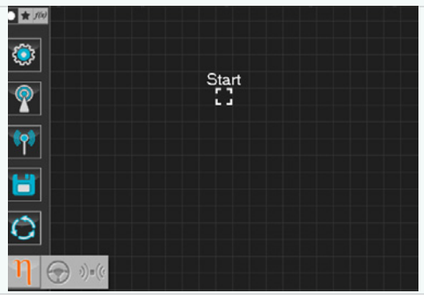


1. **메뉴 :** File/Edit/View/Tools/Help로 구성되어 있으며, DR-Visual Logic의 여러 기능을 사용할 수 있습니다.
2. **도구모음 :** 자주 사용하는 기능이 아이콘으로 만들어져 도구모음으로 구성되어 있습니다.
3. **모듈탭 :** 모듈바를 선택하는 탭입니다. 모듈바의 종류로는 모두 보기/즐겨찾기/My Module의 세가지 종류가 있습니다.
4. **모듈바 :** 모듈을 골라서 추가할 수 있는 왼쪽의 바입니다. 모두 보기의 경우 기본적으로 Motion / Sensor / Communication / Data / Flow의 5가지 모듈팩으로 구성되어 있고, HOVIS Lite/Eco의 경우 추가적으로 서드 파티 모듈팩(ETA)이 들어가 있습니다.
5. **프로그래밍 탭 :** 현재 편집하고 있는 파일 이름을 볼 수 있으며, 여러 개의 파일을 동시에 편집할 때 파일 간에 창 전환을 할 수 있습니다.
6. **전환 버튼 :** 여러 개의 파일이 동시에 열려서 프로그래밍 탭이 화면 너비 이상으로 많아졌을 때 좌우로 이동해 원하는 탭이 보이게 하는 버튼과, 그래픽 프로그래밍 창과 그것을 텍스트 코드로 변환한 결과를 보여주는 C-like 창 사이에서 전환하는 버튼으로 이루어져 있습니다.
7. **속성창 :** 모듈의 속성을 입력하는 창입니다. 모듈마다 다양한 속성이 있으며 속성창을 통해 값을 정해주게 됩니다. 그 중 일부는 입력 핀을 통해서 입력할 수도 있습니다.
8. **리모컨 :** 컴파일/다운로드/실행을 간편하게 실행시킬 수 있는 리모컨입니다.
9. **미니맵 :** 전체적인 프로그램 코드가 어떤 모양인지, 현재 보이는 곳이 어디쯤인지 알 수 있습니다. 클릭하면 그 위치로 화면을 이동할 수도 있습니다.
10. **핀 :** 입력 핀과 출력 핀으로 나누어지며, 한 모듈의 출력 값을 다른 모듈의 입력 값으로 넣을 때에 사용합니다.
11. **핀 스위치 :** 핀의 이름이 무엇인지 표시하게 하는 스위치입니다. 클릭하면 각 핀의 이름이 표시되며, 다시 클릭하면 없어집니다.
12. **커넥터 :** 핀과 핀을 연결하는 연결 선입니다.

04 프로그래밍 모듈

DR-Visual Logic은 아래와 같은 모듈로 구성됩니다.
모듈 팩에는 프로그램을 만들기 위해 필요한 모든 프로그래밍 모듈이 포함되어 있습니다. 각각의 모듈은 HOVIS Lite/Eco의 경우 제어기 DRC가, HOVIS Genie의 경우 MID가 지원해주는 기능으로 구성되어 있습니다. 본 매뉴얼에서는 파일 생성 시 HOVIS Lite/Eco를 선택한 경우에 사용 가능한 모듈만 다루며, HOVIS Genie를 선택한 경우는 HOVIS Genie 매뉴얼을 참조하시기 바랍니다.

Module Pack	Picture	Module	Description
Motion (모션)		Move	저장된 로봇 모션을 실행
		Motor	모터 별로 위치 / 속도제어
		LED	Head LED – 저장된 머리 LED 실행 Controller LED – 제어기 LED 제어
		Sound	Melody – 저장된 Buzzer 멜로디 실행 Note – Buzzer 음 하나 실행
Sensor (센서)		Sound Sensor	소리 감지로 좌우 구분(내장)
		Touch	머리 모듈의 터치 여부 (머리 모듈 연결 시 사용)
		Light	제어기의 광량 측정(내장)
		Distance	거리 측정 (제어기에 거리 센서 연결 시 사용)
		Dynamics	가속도와 각속도 측정 (제어기에 가속도/자이로 모듈 부착 시 사용)
		Hand Touch	손바닥의 택트 스위치 눌림 여부(팔 끝에 손바닥 스위치 적용 시 사용)
Com-muni cation (통신)		IRRciever	적외선 리모컨 데이터 인식
		Button	후면 제어기 버튼 인식

Module Pack	Picture	Module	Description
Data (데이터)		Operator	연산자(논리/산술/비교/비트/증감)
		MPSU RAM	DRC-005T 제어기의 RAM 레지스터 값을 읽거나 쓸 때 사용
		Servo RAM	DRS-0101/0201 등 서보 모터의 RAM 레지스터 값을 읽거나 쓸 때 사용
		OPSU RAM	DRC-004TO(옴니휠 구동부 센서모듈) 부착 시, RAM 레지스터 값을 읽거나 쓸 때 사용
		Variable	사용자 변수를 쓸 때 사용
		Constant	상수 값을 입력할 때 사용
Flow (흐름분기)		Loop	무한반복/for문
		Wait	특정 조건 동안 대기
		Delay	지정 시간 동안 대기
		Continue	반복 문의 처음으로 돌아감
		Break	반복 문이나 switch-case문을 빠져 나옴
		Switch	switch-case문의 switch
		Case	switch-case문의 case
		Label	프로그램의 특정 위치를 라벨로 지정
		Goto	지정된 라벨로 이동
ETA		ETA Actuator	서드 파티 모듈인 ETA 시리즈 중 Actuator를 연결 시 사용
		ETA Sensor	서드 파티 모듈인 ETA 시리즈 중 Sensor를 연결 시 사용

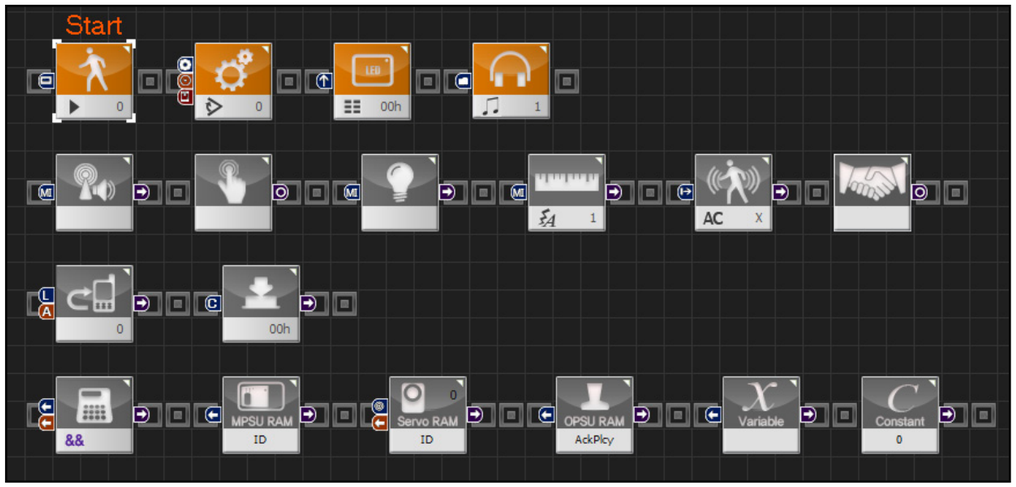
Variable 모듈의 변수 타입

Variable은 사용자 변수를 선언하기 위해 사용됩니다. 변수에는 여러 가지 타입이 있는데, 저장하는 자료의 형태와 범위에 따라서 이름이 달라집니다. C/C++에서 사용하는 이름 및 변수와 동일합니다. 사용자 변수는 처음 사용되는 순간 자동으로 선언되며, 별도의 초기값을 넣지 않는 이상 0으로 초기화 됩니다. DRC-005T의 메모리는 한정되어 있으므로, 변수의 예상되는 값에 따라서 알맞은 범위의 변수 타입을 선언하면 메모리를 효율적으로 사용할 수 있습니다.

종류	범위	설명
bool	True(1)/False(0)	참/거짓
char	-128~127	1바이트 부호 있는 정수
unsigned char	0~255	1바이트 부호 없는 정수
short	-32,768~32,767	2바이트 부호 있는 정수
unsigned short	0~65,536	2바이트 부호 없는 정수
int	-2,147,483,648~2,147,483,647	4바이트 부호 있는 정수

04-1 프로그래밍 모듈 일반형 모듈

일반형 모듈은 순차적으로 모듈끼리 연결하여 사용됩니다. Flow 모듈을 제외한 모든 모듈이 여기에 해당합니다.



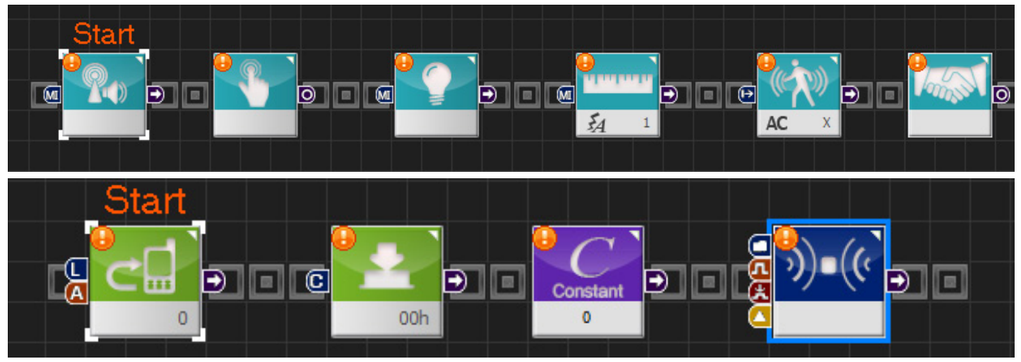
위부터 차례로 Motion, Sensor, Communication, Data 모듈 팩의 모듈 아이콘입니다.

일반형 모듈 사용하기

모듈 중에는 단독으로 사용되는 모듈도 있고, 출력 핀이 다른 모듈에 연결되어야만 반영이 되는 모듈도 있습니다.


```
1  void main()
2  {
3          motion( 0 )
4          jog( 512, 0, 0, 60 )
5          led( 0x00 )
6          melody( 1 )
7          ETAAct( 253, 0, 0, 0 )
8  }
```

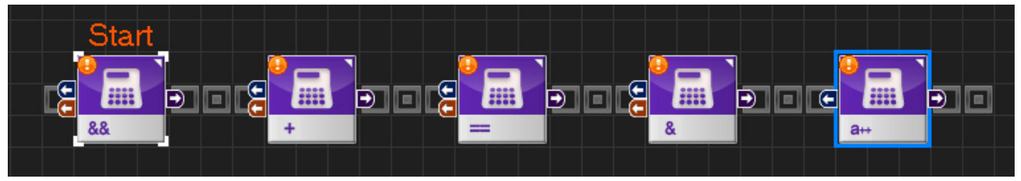
Motion 모듈 팩의 모듈들과 ETA의 ETA Actuator의 경우 출력이 없으며 단독으로 있어도 소스로 변환됩니다.




```
1  bool a
2  void main()
3  {
4          a=( MPSU_SoundDetectFlag && MPSU_SoundDir
5  }
```

Sensor 모듈 팩과 Communication 모듈 팩에 속한 모듈들과 Data의 Constant, ETA의 ETA Sensor 모듈은 단독으로는 소스로 변환되지 못하며, 출력 핀이 다른 입력 핀과 연결 되어야만 의미를 가집니다. 단독으로 소스로 변환되지 못할 때, 모듈의 왼쪽 상단에 오류가 표시됩니다.

Data 모듈 팩에 속한 모듈들은 경우에 따라서 단독으로 소스로 변환되기도, 안 되기도 합니다.



Operator 모듈의 경우 대부분 경우 출력 핀이 다른 모듈에 연결 되어야만 소스로 변환됩니다.

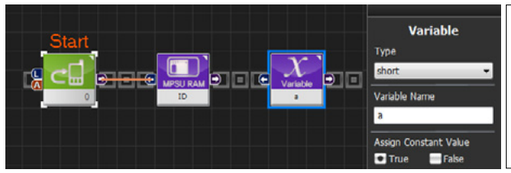

```
1  int a
2  void main()
3  {
4          a++
5  }
```

그러나 한가지 예외가 있습니다. 증감연산자(Incremental)에 값을 바꿀 수 있는 변수가 연결된 경우, 출력 핀 연결이 없어도 그 변수를 증가 시키는 소스로 변환됩니다.


```
1  int a
2  void main()
3  {
4          motion( ( a++ ) )
5  }
```

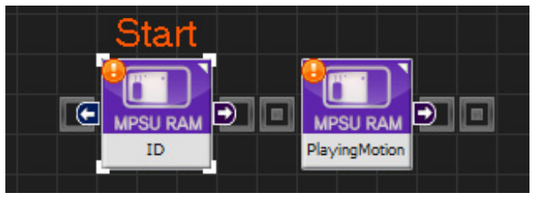
이 상태에서 출력 핀이 다른 모듈에 연결될 경우, 변수를 증가 시키는 소스는 연결된 모듈의 소스에 포함됩니다.

그 외에 MPSU RAM, Servo RAM, OPSU RAM, Variable은 처음 생성했을 때는 단독으로 소스로 변환되지 않습니다.

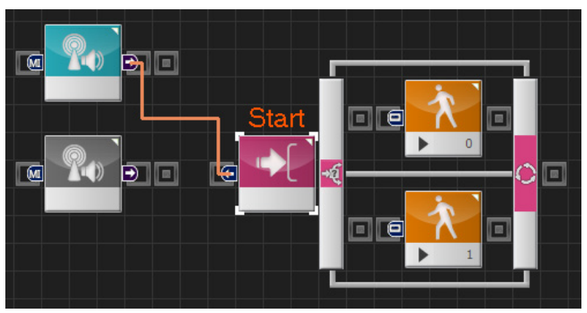



```
1  short a
2  void main()
3  {
4          MPSU_ID=( MPSU_RmcLength >= 0 && MPSU_RmcData == 0 )
5          a=0
6  }
```

하지만, 입력 핀에 다른 모듈이 연결 되었을 때는 출력 핀의 연결 없이도 그 변수에 값을 대입하는 소스로 변환됩니다. 또한, 속성창에서 Assign Constant Value를 True로 체크한 경우는 밑의 상수 값을 대입하는 소스로 변환됩니다.



한편, MPSU RAM, Servo RAM, OPSU RAM 중에는 값을 읽을 수만 있고 쓸 수 없는 센서 값, 위치 정보 등의 항목도 존재합니다. 이 경우에는 입력 핀이 없어지며 대입이 불가능합니다.



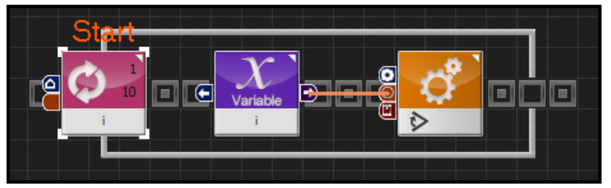
출력 핀의 연결 없이도 소스로 변환되는 모듈은 무조건 Start에서 시작되는 프로그램 라인 상에 있어야 활성화 되고 소스에 반영이 됩니다. 하지만 출력 핀의 연결을 통해 소스에 반영되는 모듈은 프로그램 라인 밖에 있어도 커넥터만 연결이 되어 있으면 반영됩니다.

프로그래밍 모듈

Flow형 모듈

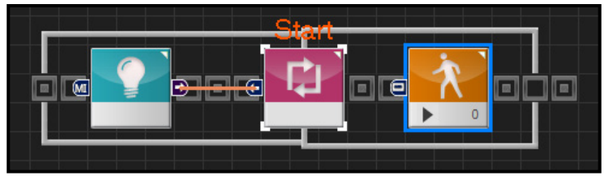
04-2

Flow 형 모듈은 일반 모듈과 연결하여 반복, 분기 등을 프로그래밍의 흐름을 연결시켜주는 모듈을 말합니다. 따라서 일반형 모듈과는 다르게 외곽 아웃라인 형태의 띠가 형성되는 그래픽 구조를 갖습니다.



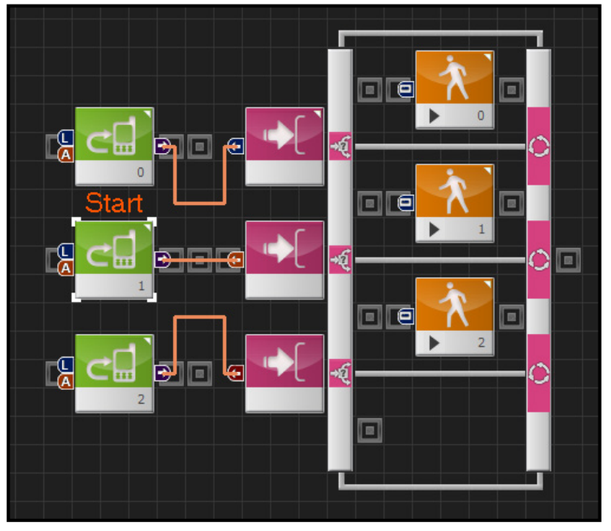
Loop

Loop 는 반복을 지시하는 모듈로서, 일정 횟수 동안 반복하는 For 문과 영원히 반복하는 Forever 문이 있습니다.



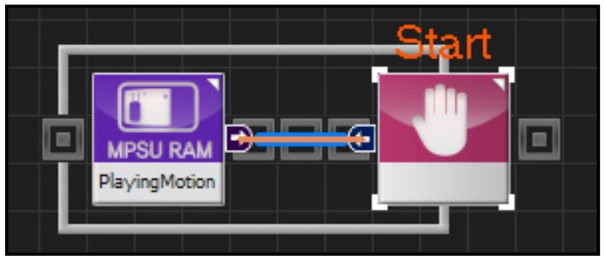
While

While 은 앞 조건이 참일 동안 뒤를 실행하라 는 의미로서 조건과 결합된 반복 문입니다.



If-else

조건에 따라서 프로그램을 분기하는 If-else문입니다. 속성창의 버튼을 클릭해서 분기문의 수를 늘이거나 줄일 수 있습니다.

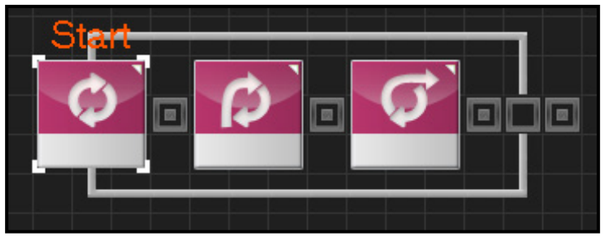


Wait

입력 조건이 참일 동안 프로그램의 실행을 멈추고, 거짓이 되면 다시 실행을 재개합니다.

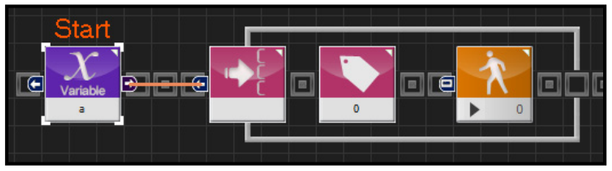
Delay

일정 시간 동안 프로그램의 실행을 지연시킵니다.



Continue, Break

Continue는 반복 문 안에서만 쓰일 수 있으며, 반복문의 처음으로 돌아가 계속 실행하라는 의미입니다.
Break는 반복 문 안과 switch-case 문 안에서만 쓰일 수 있으며, 반복 문과 switch-case문을 빠져나가라는 의미입니다.



Switch, Case

이 두 모듈은 함께 쓰여서 switch-case 문을 구성합니다. switch 모듈의 입력에 따라서, 일치하는 case문 이후의 것을 실행합니다.

Label, Goto

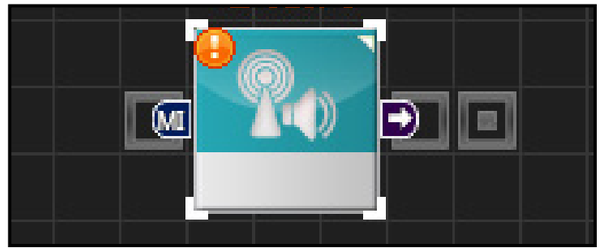
이 두 모듈은 함께 쓰여서 Jump 기능을 구현합니다. label 모듈의 이름과 같은 goto 문을 구성하면, goto문을 만났을 때 label 모듈로 이동하게 됩니다.

프로그래밍 모듈

핀

04-3

모듈에 따라 입력 값과 출력 값을 갖는 모듈이 있습니다. 출력 핀의 결과 값은 다른 모듈의 입력 핀과 연결되어 다른 모듈의 입력 값으로 동작합니다.



핀

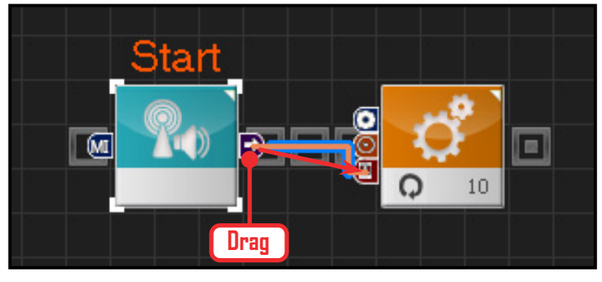
입출력 값이 있는 모듈은 모듈상에 좌우 측에 핀이 나와 있습니다. 왼쪽은 입력 핀, 오른쪽은 출력 핀입니다.

말풍선 도움말

핀의 아이콘만 보고는 어떤 기능의 핀인지 구분하기 어렵습니다. 핀 위에 마우스를 올리면 좌측과 같이 말풍선으로 핀 이름이 나옵니다.

말풍선 펼치기

여러 개의 핀 이름을 한꺼번에 보고 싶으면 우측 상단의 세모모양의 핀 스위치를 클릭하면 핀 양쪽에 핀 이름 말풍선이 나옵니다. 다시 클릭하면 없어집니다.

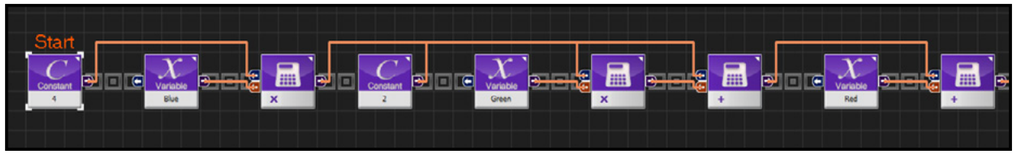


핀 연결

앞 모듈의 출력 값을 뒷 모듈의 입력 값에 넣기 위해서는 마우스로 드래그해서 두 핀을 연결하면 좌측과 같이 커넥터가 표기됩니다.

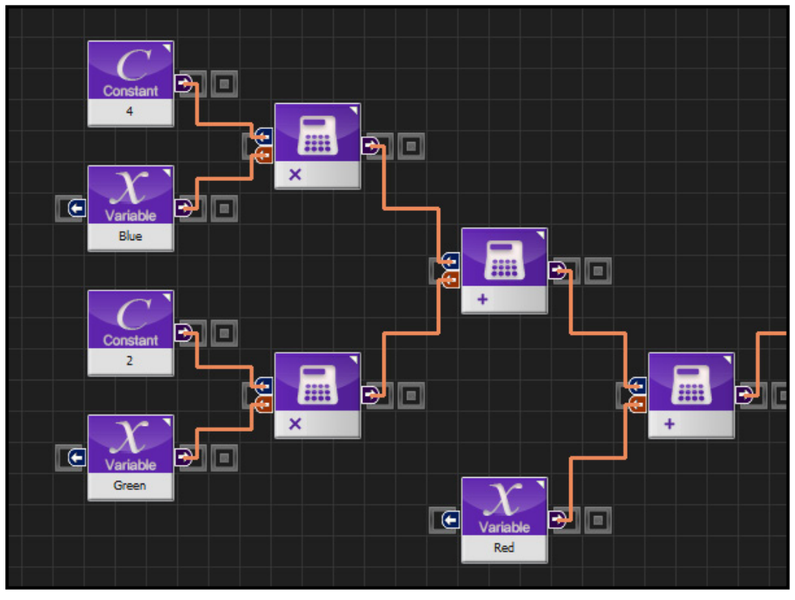
04-4 프로그래밍 모듈 연결형 타입

모듈을 연결할 때 직렬형 연결과 병렬형 연결이 있습니다.



직렬형 연결

직렬형 연결은 말 그대로 좌에서 우로 모듈을 순차적으로 연결하는 것입니다. 위에는 연산에 관한 프로그래밍입니다. ((4xBlue)+(2xGreen))+(1xRed) 연산식을 직렬형 연결로 표현한 것입니다.

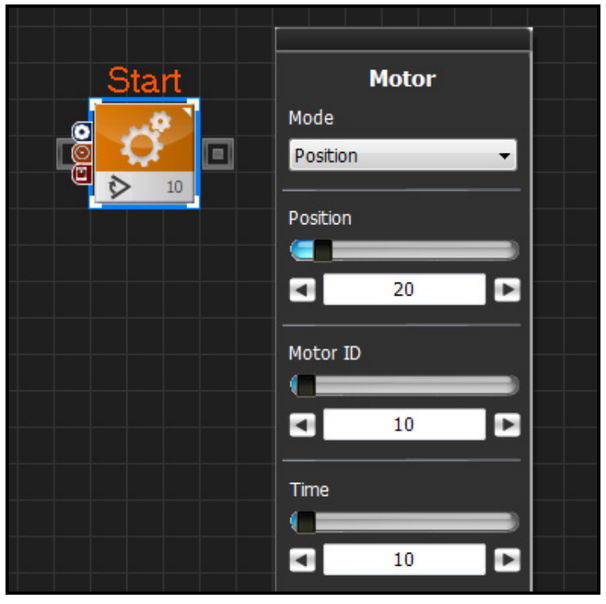


병렬형 연결

병렬형 연결은 상하 공간을 활용하여 모듈을 병렬적으로 연결하는 것입니다. 아래 예시는 위 직렬형 연결과 같은 프로그래밍입니다.

05 속성창

모듈마다 자신의 속성을 가지고 있고, 그 속성값을 설정해줘야만 프로그래밍을 수행할 수 있습니다. 리스트 팝업, 라디오 버튼, 숫자 설정 등의 UI 로 표현되며, 각 모듈별 속성 설명 및 속성값/제한값 등은 도움말을 참조하기 바랍니다.

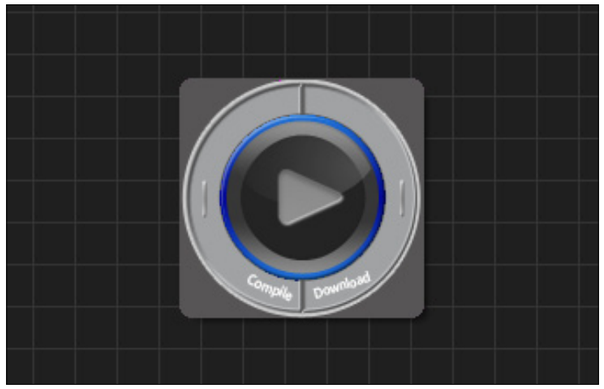


속성창

모터모듈을 클릭했을 때 우측과 같은 속성창이 나타납니다.
모터는 위치제어와 속도제어 속성을 가집니다. Mode 에서 위치제어를 위해서는 Postion 을 선택하고, 속도 제어를 위해서는 Velocity 를 선택합니다.
세부 설정에 Position, Motor ID, Time 등의 값을 조절합니다.

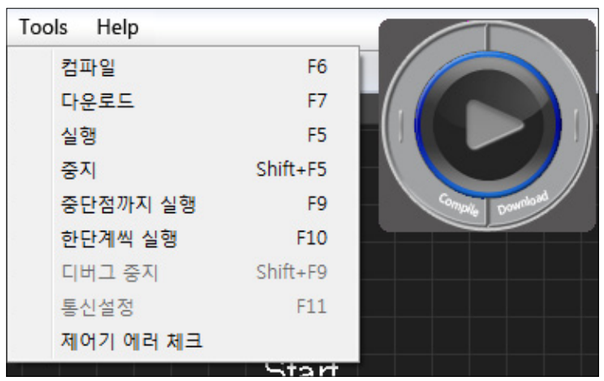
06 컴파일/다운로드하기

로봇의 프로그래밍을 완료하고, 컴파일, 다운로드, 로봇에서 실행을 거치게 됩니다. 프로그래밍 창 좌측 하단에 큰 아이콘으로 제공하며, 도구 메뉴에서 상세하게 활용할 수 있습니다.



다운로더 아이콘

왼쪽 아이콘은 다운로더 아이콘입니다. 좌측은 Compile, 우측은 Download를 나타내며, 가운데 화살표는 로봇에서의 실행을 의미합니다.



도구 메뉴

도구 메뉴에서는 좀 더 상세하게 실행이 가능합니다.

컴파일 : 작성한 프로그램을 컴파일 합니다.
다운로드 : 컴파일한 프로그램을 다운로드 합니다.
실행 : 다운로드 한 프로그램을 실행합니다.
중지 : 프로그램을 중지시킵니다.
중단점까지 실행 : C-like 창에서 소스에 중단점을 지정하면 그곳까지 실행 후 일시 중단합니다.
한단계씩 실행 : C-like 창에서 소스 한 줄씩 실행합니다.

07 다양한 기능

DR-Visual Logic에서 제공하는 메뉴의 기능은 아래 표와 같습니다.

분류	항목	설명
File	**새창**	새로운 dts 파일을 만듭니다.
	불러오기	저장된 dts 파일을 불러옵니다.
	창닫기	현재 열린 dts 파일을 닫습니다.
	저장	현재 dts 파일을 저장합니다.
	다른이름저장	현재 dts 파일을 다른 이름으로 저장합니다.
	인쇄미리보기	인쇄하기 전에 화면상으로 미리 봅니다.
	인쇄	인쇄합니다.
	종료	프로그램을 종료합니다.
Edit	**전단계**	직전의 작업을 취소합니다.
	앞복원	취소한 작업을 다시 복원합니다.
	잘라내기	현재 선택된 부분을 클립보드로 잘라냅니다.
	복사하기	현재 선택된 부분을 클립보드로 복사합니다.
	붙여넣기	클립보드의 내용을 붙여넣습니다.
	삭제	현재 선택된 부분을 삭제합니다.
View	**기본보기**	확대/축소 상태를 기본 상태로 만듭니다.
	화면확대	화면을 확대합니다.
	화면축소	화면을 축소합니다.
	모듈을 중앙으로	현재 선택된 모듈을 화면 중앙에 오도록 화면을 이동시킵니다. 선택된 모듈이 없을 시 Start 지점으로 이동합니다. C-like 창에서 소스의 줄을 클릭하고 이 기능을 사용 시 Graphic 상에서 어떤 모듈이 해당하는지 알 수 있습니다.

분류	항목	설명
Tools	**컴파일**	작성한 프로그램을 컴파일합니다.
	다운로드	컴파일한 프로그램을 다운로드 합니다.
	실행	다운로드 한 프로그램을 실행합니다.
	중지	프로그램을 중지시킵니다.
	중단점까지 실행	C-like 창에서 소스에 중단점을 지정하면 그곳까지 실행 후 일시 중단합니다.
	한단계씩 실행	C-like 창에서 소스 한 줄 씩 실행합니다.
	디버그 중지	디버깅을 중지시킵니다.
	통신설정	통신을 설정합니다. Serial과 Wifi 중에 선택할 수 있으며 Lite/Eco의 경우 Serial을 선택해야 합니다.
	제어기 에러 체크	현재 DRC-005T 제어기의 에러 상태를 체크합니다.
Help	**내용색인**	도움말 파일을 엽니다.
	온라인지원	동부로봇 홈페이지를 엽니다.
	소프트웨어 업데이트	소프트웨어가 최신 버전인지 확인합니다.
	펌웨어 업데이트	제어기의 펌웨어를 업데이트합니다.
	펌웨어복구	제어기에 다른 펌웨어가 쓰여졌거나 제어기가 고장난 경우 펌웨어를 복구합니다.
	프로그램정보	프로그램 정보 창을 엽니다.

기타 기능

Wheel 버튼 클릭 후 드래그: 화면이 이동합니다.

Shift + 왼쪽 클릭 후 드래그: 화면이 이동합니다.

선택한 모듈을 Ctrl + 왼쪽 클릭 후 드래그: 현재 선택한 모듈을 복사합니다.

디버깅 하기

```
1  void main()
2  {
3          SERVO_ID[254]=0x60
4          jog( 512, 0, 254, 100 )
5          delay( 1000 )
6          jog( 235, 0, 0, 100 )
7          delay( 1000 )
8          jog( 235, 0, 1, 100 )
9  }
```

```
1  void main()
2  {
3          SERVO_ID[254]=0x60
4          jog( 512, 0, 254, 100 )
5          delay( 1000 )
6          jog( 235, 0, 0, 100 )
7          delay( 1000 )
8          jog( 235, 0, 1, 100 )
9  }
```

프로그래밍을 마친 후 C-like 버튼을 눌러 전환하면, 모듈이 어떻게 소스로 변환되었는지 결과를 볼 수 있습니다.

이 때 왼쪽의 동그라미를 클릭하면 중단점을 설정할 수 있습니다.

디버깅 중이 아닐 때 Tools의 중단점까지 실행 혹은 한단계씩 실행 명령을 내리면, 현재 소스를 컴파일, 다운로드 한 뒤, 디버깅 모드로 프로그램이 실행됩니다.

```
1  void main()
2  {
3          SERVO_ID[254]=0x60
4          jog( 512, 0, 254, 100 )
5          delay( 1000 )
6          jog( 235, 0, 0, 100 )
7          delay( 1000 )
8          jog( 235, 0, 1, 100 )
9  }
```

중단점까지 실행 명령을 실행한 상태입니다. 디버깅이 시작되고, 처음 만난 중단점에서 프로그램이 정지되었습니다.

```
1  void main()
2  {
3          SERVO_ID[254]=0x60
4          jog( 512, 0, 254, 100 )
5          delay( 1000 )
6          jog( 235, 0, 0, 100 )
7          delay( 1000 )
8          jog( 235, 0, 1, 100 )
9  }
```

다시 중단점까지 실행을 하면, 그 다음 중단점에서 프로그램이 멈춥니다.

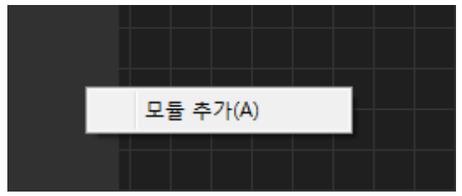
```
1  void main()
2  {
3          SERVO_ID[254]=0x60
4          jog( 512, 0, 254, 100 )
5          delay( 1000 )
6          jog( 235, 0, 0, 100 )
7          delay( 1000 )
8          jog( 235, 0, 1, 100 )
9  }
```

그 상태에서 한단계씩 실행을 하면, 중단점과 무관하게 그 다음 줄에서 프로그램이 멈춥니다.

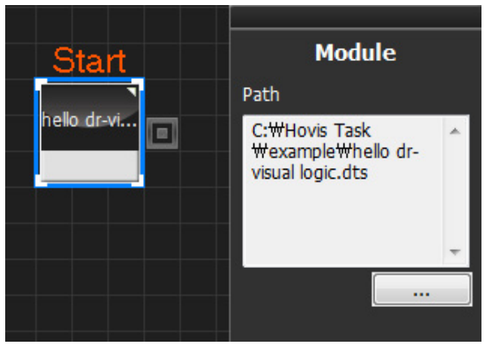
위와 같은 디버깅 기능을 사용하면, 프로그램의 분기나 반복 등이 원하는 대로 실행되지 않을 때 프로그램이 어떤 경로로 실행되는지 살펴볼 수 있습니다.

My Module 사용하기

DR-Visual Logic으로 작성한 dts 파일을 다른 dts 파일에서 불러와서 함수처럼 사용할 수 있습니다.



모듈 탭에서 My Module을 선택한 후, 오른쪽 클릭해 모듈을 추가하겠다는 명령을 내립니다. 그러면 파일 선택 창이 뜨며, 추가할 dts 파일을 선택할 수 있습니다.



My Module이 추가됩니다. 클릭해서 배치하면 dts 파일을 마치 모듈화된 함수처럼 사용할 수 있습니다.

```
1  void <<hello dr-visual logic>>()
2  {
3          SERVO_ID[254]=0x60
4          jog( 512, 0, 254, 100 )
5          delay( 1000 )
6          jog( 235, 0, 0, 100 )
7          delay( 1000 )
8          jog( 235, 0, 1, 100 )
9  }
10 void main()
11 {
12         <<hello dr-visual logic>>()
13 }
```

C-like 창으로 전환하여 소스를 보면, 추가된 모듈이 함수처럼 사용되고 있음을 볼 수 있습니다.

08 모듈별 예제 프로그래밍

기본 제공되는 샘플 프로그램은 제어기 DRC 플랫폼 기반 프로그램이며, 기본 16축 휴머노이드를 기반으로 작성되었습니다. 18,20축 휴머노이드나 변신 조립형 로봇에 따른 모션이나 모듈 변경이 있을 때에는 그것에 맞춰서 재 프로그래밍 해야합니다.
프로그램 적용시 먼저 로봇 모터 ID를 확인하고, 로봇의 센서 위치를 확인합니다. 또한 DR-SIM을 통해서 로봇 저장모션 리스트를 확인하여 프로그래밍시 맞는 Index 값을 줍니다. 기본 제공하는 샘플예제는 아래와 같습니다

Module Pack	Module	예제 샘플
Motion (모션)	Move	Move 는 제어기 DRC 에 저장된 로봇의 모션을 가져와 프로그램에 적용하는 것입니다. 로봇 모션은 번호로 가져올 수 있으며, 번호별 모션 이름은 DR-SIM 에서 확인할 수 있습니다. DR-SIM 에서 편집된 특정 모션을 로봇에서 무한반복적으로 돌리는 프로그래밍을 해봅니다. 로봇 신뢰성 테스트나 전시등에서 활용도가 높고, 난이도가 있는 프로그램이므로 천천히 따라해보기 바랍니다.
	Motor	모터를 하나씩 제어하여 춤을 추는 프로그램 입니다.
	LED	제어기 DRC 의 버튼을 누르면 LED 가 켜지고 꺼지는 프로그램입니다. (With Button)
	Sound	리모콘의 버튼 입력(1 번~8 번)을 받아서 음을 울리도록 하는 프로그램입니다. (With IRRecieve)
Sensor (센서)	Sound Sensor	Sound Sensor 는 제어기 DRC 내부의 양쪽에 위치합니다. 왼쪽측면에서 박수를 치고 왼쪽 손을 들고, 오른쪽측면에서 박수를 치면 오른쪽 손을 드는 프로그래밍을 해봅니다. (Sample 두번째 안) 주변에 소음이 많으면, 각각 양쪽에서 소리를 구분하는게 어려워집니다. 한쪽에서 박수를 쳐도 양손을 모두 들거나, 불규칙적으로 반응합니다. 주변에 소음이 있더라도 로봇이 정확한 반응을 한다는 것을 보여주기위해 좀더 세분화된 프로그래밍을 할 필요가 있습니다. 처음 소리가 입력되었을시 다른 소리가 입력되지 않도록 강제로 Delay 를 줘서 한번 박수 칠 때 한번만 팔을 올리도록 정확도를 높이는 프로그램입니다.
	Light	빛의 세기에 따라 로봇의 모터를 동작하는 예제입니다. 외부의 빛이 어두워지면 로봇이 왼팔을 올립니다. (제어기 뒤쪽의 CDS센서를 손가락으로 가리면 빛의 들어오는 양이 없어 어두워지므로, 로봇이 왼팔을 올리도록 프로그래밍합니다.)
	Distance	PSD Digital(거리센서) : 벽이 일정거리 가까워지면 뒷 걸음질 치다가 우회전 한 후 전진하면 프로그램 입니다. PSD Analog(거리센서) : 벽이 가까워지면 좌회전 하면서 벽을 회피하는 프로그램 입니다.
	Dynamics	Accerlateration : 앞으로 넘어졌을 때 일어나는 프로그래밍, 뒤로 넘어졌을 때 일어나는 프로그램
Communication (통신)	IRRciever	리모콘의 1 번부터 8 번까지 음높이를 설정하여 DRC 제어기 사운드가 울리도록 하는 프로그래밍입니다. 1~8 번은 도~시까지 매칭합니다. (With Sound)
	Button	제어기 DRC 의 버튼을 누르면 제어기 뒤편의 LED 가 반응하는 프로그램입니다. (With LED)

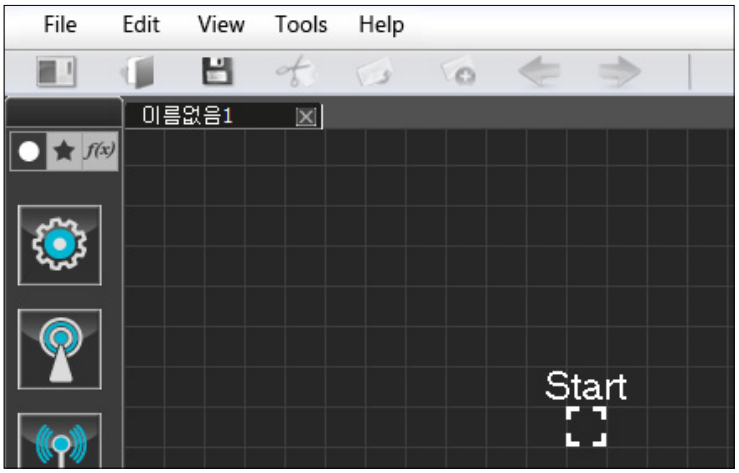
Move

08-1

예제설명

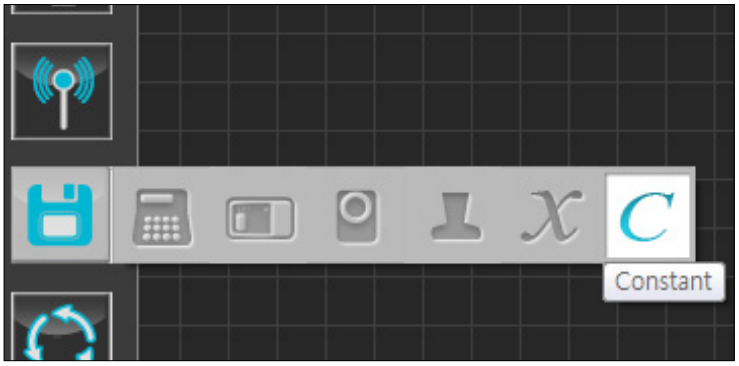
Move 모듈은 제어기 DRC에 저장된 로봇의 모션을 가져와 실행하는 모듈 입니다. 로봇 모션은 번호로 가져올 수 있으며, 번호별 모션 이름은 DR-SIM에서 확인할 수 있습니다. DR-SIM에서 편집해 다운로드 된 모션을 로봇에서 무한 반복적으로 돌리는 프로그래밍을 해봅니다. 로봇 신뢰성 테스트나 전시 등에서 활용도가 높은 프로그램입니다.

※ 이 예제에서 사용하는 모션과 모션 번호는 기본제공 모션과 다릅니다. DR-SIM을 통해 편집한 모션을 DRC에 다운로드 한 경우를 가정하고 작성된 예제입니다.



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.



02 모듈 선택

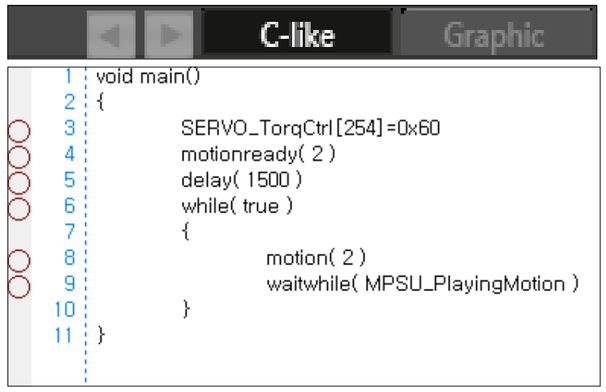
모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.

04 전체 프로그래밍

저장된 모션을 가져와 무한히 반복 시키는 프로그램입니다.

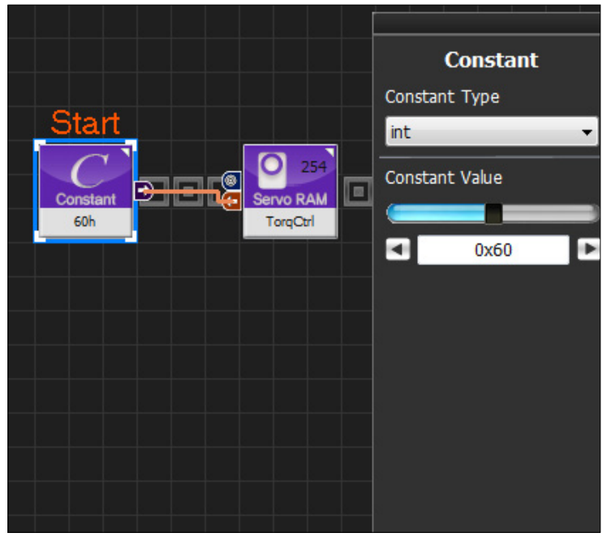


05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

저장된 모션을 불러와 사용하는 프로그램 소스 화면입니다.

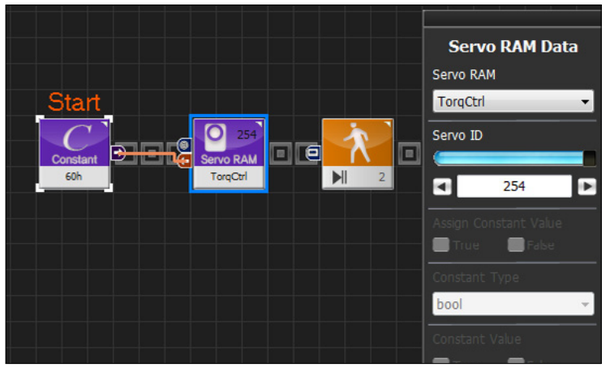
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

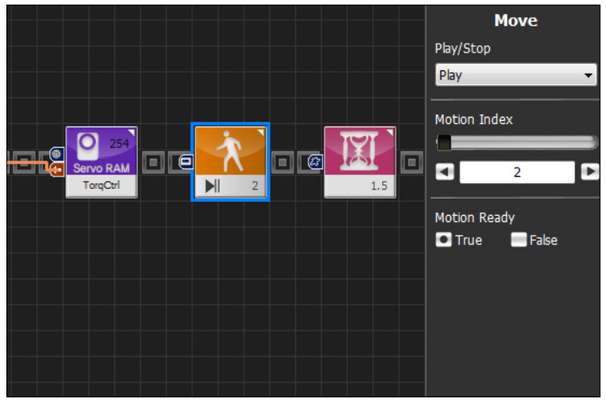
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다. 이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

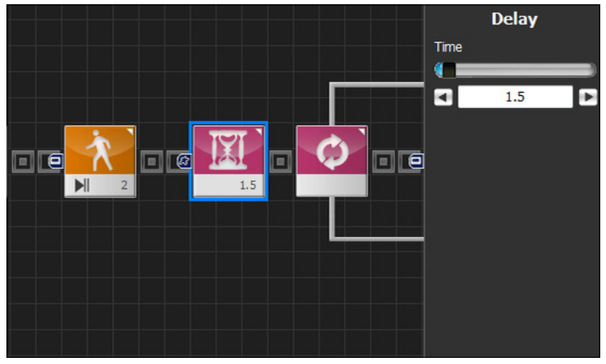
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



08 Motion Ready

저장된 모션을 가져올 때 로봇의 현재 상태에서 갑작스럽게 모션이 변동하고 움직일 수가 있습니다.
현재 상태와 모션 시작상태가 너무 다르면 모터에 무리가 가거나 사용자에게 위험할 수도 있습니다. 그래서 Motion Ready 모드로 모션을 실행해 모션이 동작할 준비시간을 줘야합니다.

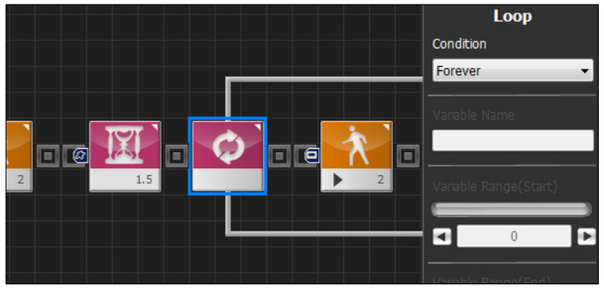
Motion 〉 Move 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 2를 선택합니다. 2번 모션을 가져오겠다는 뜻입니다. 참고로 이 프로그래밍의 2번 모션은 앉았다 일어나는 것입니다. 꼭 2번일 필요는 없으며, 사용자가 실행하고 싶은 모션 번호를 입력하면 됩니다.
Motion Ready : True를 선택합니다. True를 선택하면 동작하고자 하는 모션의 첫 번째 상태로 서서히 이동합니다.



09 Delay

Motion Ready 동작이 끝나기 전에 진행하는 것을 방지하기 위해 Delay 값을 1.5초로 설정합니다.

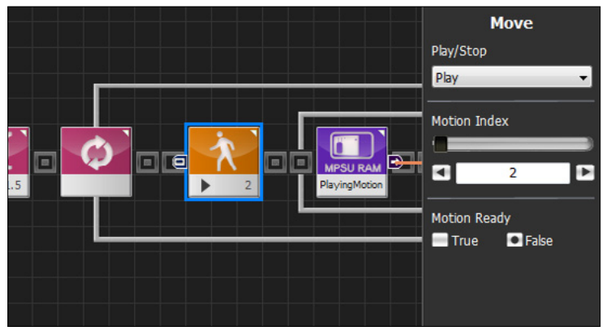
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



10 반복

모션을 무한 반복 시키기 위해 loop 모듈을 배치해 무한 반복 모드로 설정합니다. loop 모듈 안의 내용이 무한 반복 됩니다.

Flow 〉 Loop을 선택해 이전 모듈 뒤에 배치합니다.
Condition : Forever로 선택합니다. 무한 반복하겠다는 의미입니다.



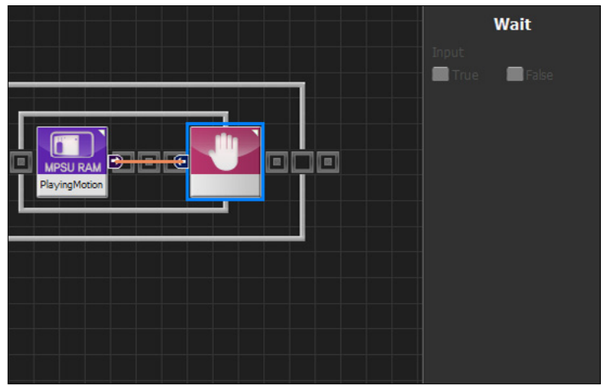
11 모션 동작

Move 모듈의 Motion Ready 값을 False 로 설정하면 모션을 처음부터 끝까지 동작시킨다는 의미입니다. 2번 모션을 실행시킵니다.

Motion 〉 Move 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 2를 선택합니다. 2번 모션을 가져오겠다는 뜻입니다.
Motion Ready : False를 선택합니다. 이번에는 모션 전체가 실행됩니다.

참조: 모션 보기

DR-SIM 프로그램에서 로봇과 연결하여 로봇설정을 클릭하면, 현재 제어기에 있는 로봇 모션을 확인할 수 있습니다.
현재 2번 모션은 양팔을 벌려 앉았다 일어나는 모션입니다.

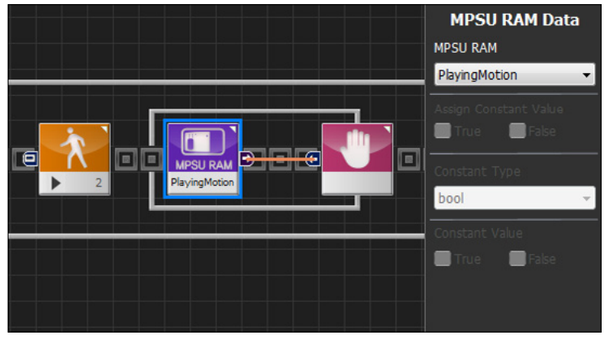


12 모션 동작 대기

Move 모듈은 실행 후 모션이 끝날 때까지 기다리지 않으며 모션을 시작 시키고 다음 모듈로 넘어갑니다. 따라서 loop안에 Move모듈 하나만을 넣고 실행하면 모션을 이미 실행중임에도 loop를 계속 돌면서 모션실행 명령을 반복하게 됩니다.
이렇게 되면 Move모듈을 만난 횟수와 실제모션을 실행한 횟수가 달라집니다.
따라서 실행한 모션이 끝날 때까지 기다렸다가 다시 loop의 처음으로 돌아가게 하는 편이 더 정확합니다.

MPSU RAM 모듈에는 PlayingMotion이라는 항목이 있습니다. 이 항목은 로봇이 모션을 실행 중인지 확인하는 변수입니다. 모션 실행 중에는 1, 아닐 때는 0의 값을 가집니다. 이 PlayingMotion을 조건으로 해서 대기를 걸면 로봇의 모션이 끝날 때까지 대기할 것입니다.

Flow 〉 Wait 모듈을 선택해 이전 모듈 뒤에 배치합니다.

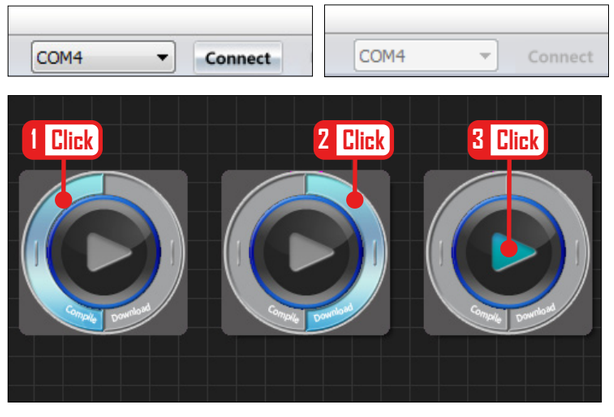


13 모션 동작 변수 연결

PlayingMotion을 조건으로 하여, 모션이 끝날 때까지 기다리는 루틴을 만듭니다.

Data 〉 MPSU RAM 모듈을 선택해 wait 모듈의 왼쪽 아웃라인 안에 배치합니다.
MPSU RAM : PlayingMotion 을 선택합니다.

MPSU RAM 모듈의 출력 핀을 wait 모듈의 입력 핀에 연결합니다. 이러면 wait 모듈은 PlayingMotion이 참(0이 아님)일 동안 프로그램을 대기하게 합니다. 모션이 끝나면 대기를 멈추고 loop의 처음으로 돌아갈 것이고, 로봇은 모션 실행을 무한히 반복할 것입니다.

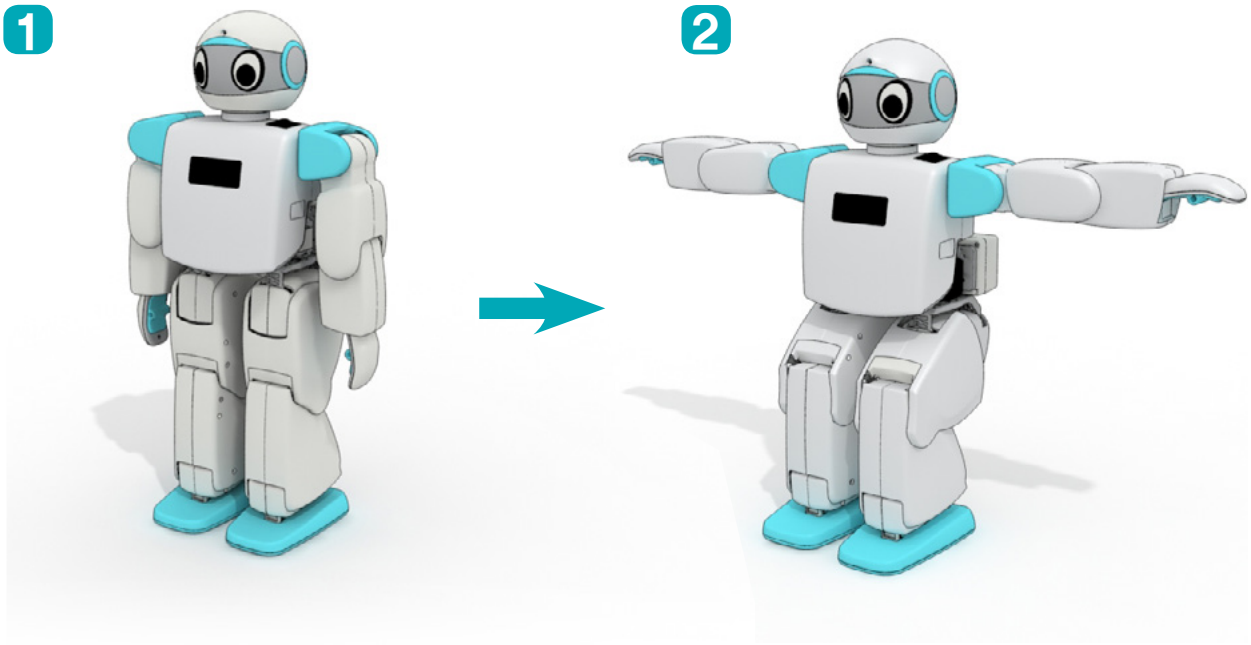


14 다운로드

프로그래밍 후 컴파일 –〉 로봇에 다운로드 –〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



15 로봇동작

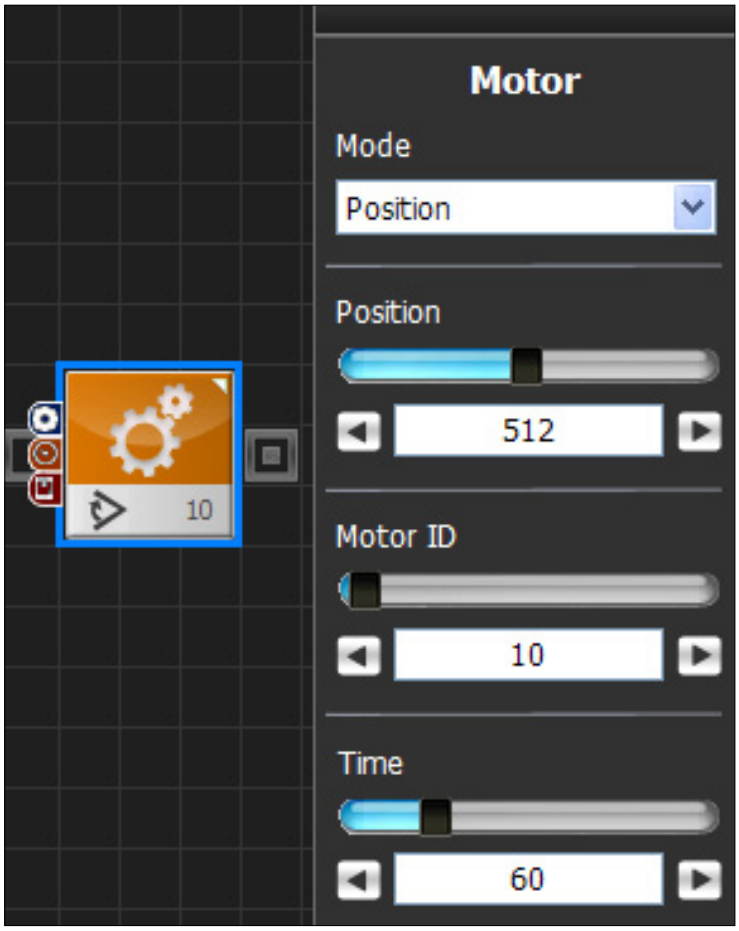
로봇이 앉았다 일어나는 모션을 반복적으로 수행합니다.

Motor

08-2

예제설명

Motor 모듈에는 두 가지 동작 모드가 있습니다. 하나는 위치 제어 모드, 하나는 속도 제어 모드입니다.



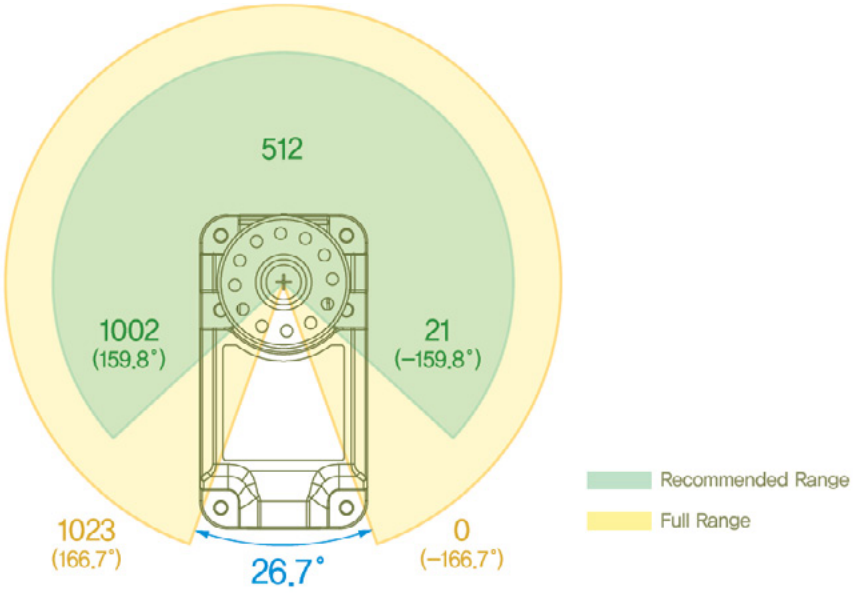
1 위치 제어 모드

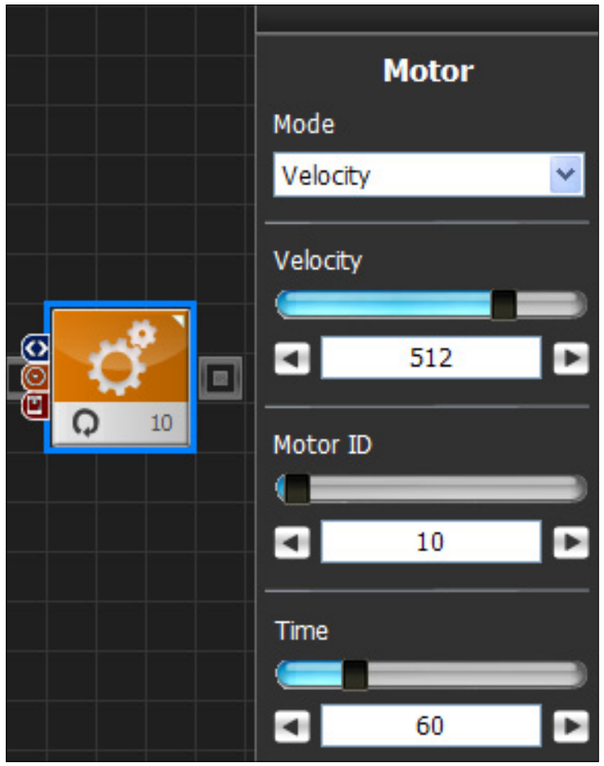
위치 제어 모드는 원하는 모터를 특정 위치로 이동 시키는 모드입니다.

Position은 -127~1151의 값을 가질 수 있는데, 서보의 초기 공장 설정에서 허용되는 값은 21~1002이며 이 범위를 벗어나는 값은 모터의 최소/최대 위치값과 위치 보정치가 변경되면 나올 수 있는 값들입니다. 모터의 정위치 값은 512로, 조립할 때의 기준이 되는 위치도 512입니다. 정상적으로 Hovis의 모든 모터 위치를 512로 보내면, 양팔을 90도로 벌리고 선 자세를 취하게 됩니다. 서보의 제어 범위와 정위치에 대한 정리된 그림은 아래를 참조 하세요.

Motor ID는 제어할 서보의 ID입니다.

Time은 서보가 목표 위치로 이동할 시간을 나타냅니다. 1당 11.2ms의 실제 시간을 의미하며, 가령 100으로 설정할 경우 1.12초 동안 원하는 위치로 이동하게 됩니다.





2 속도 제어 모드

속도 제어 모드는 원하는 모터를 특정 속도로 무한 회전 시키는 모드입니다.

Velocity는 -1023~1023의 값을 가질 수 있는데, 숫자가 클수록 큰 출력으로 빠르게 회전을 시키게 됩니다. 양수와 음수의 차이는 회전 방향입니다.

Motor ID는 제어할 서보의 ID입니다.

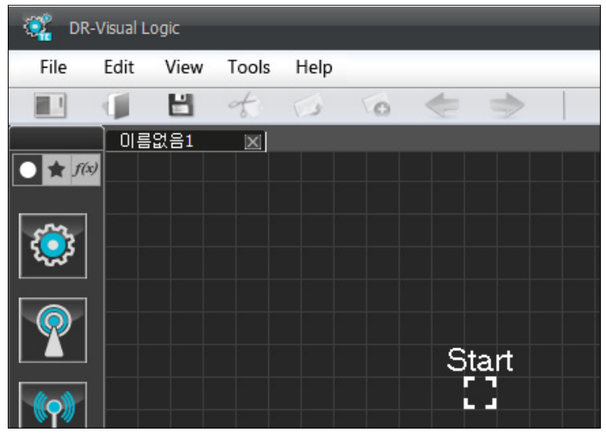
Time은 서보가 목표 속도에 도달할 시간을 시간을 나타냅니다. 1당 11.2ms의 실제 시간을 의미하며, 가령 100으로 설정할 경우 1.12초 동안 목표 속도로 서서히 속도가 변하게 됩니다.

예제설명

흔히 로봇 모션은 각각의 모터를 일일이 제어하고 종합하여 동작을 만들게 됩니다. 그 복잡성으로 인해 모터를 일일이 제어하는 것 보다는 DR-SIM 같은 툴을 제공하여 모션을 쉽게 만드는 게 보편적입니다.

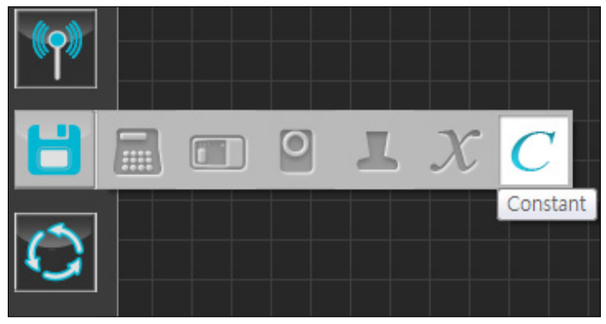
이번 프로그래밍은 DR-SIM을 사용하지 않고, DR-Visual Logic을 이용하여 모터를 하나하나 조작하여 연속되는 모션을 만들어보고자 합니다. 모션 목표는 웨이브 댄스를 추는 로봇입니다. 완료되었을 때 흥미로운 동작이니 천천히 따라 해보세요.

※ 주의 – 호비스 에코의 경우 로봇 팔의 구조가 호비스 라이트와 달라서 웨이브 댄스 구현에 한계가 있습니다.
이번 예제를 참고하여 더욱 흥미로운 동작을 만들어 보세요.



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.

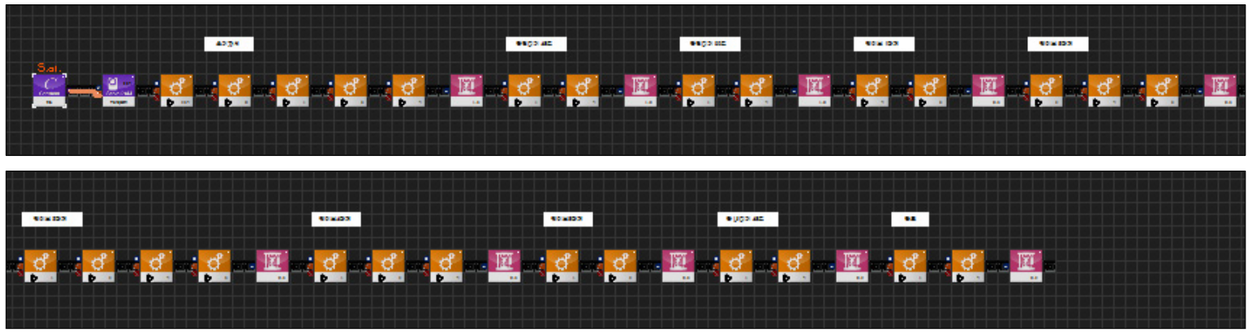


02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

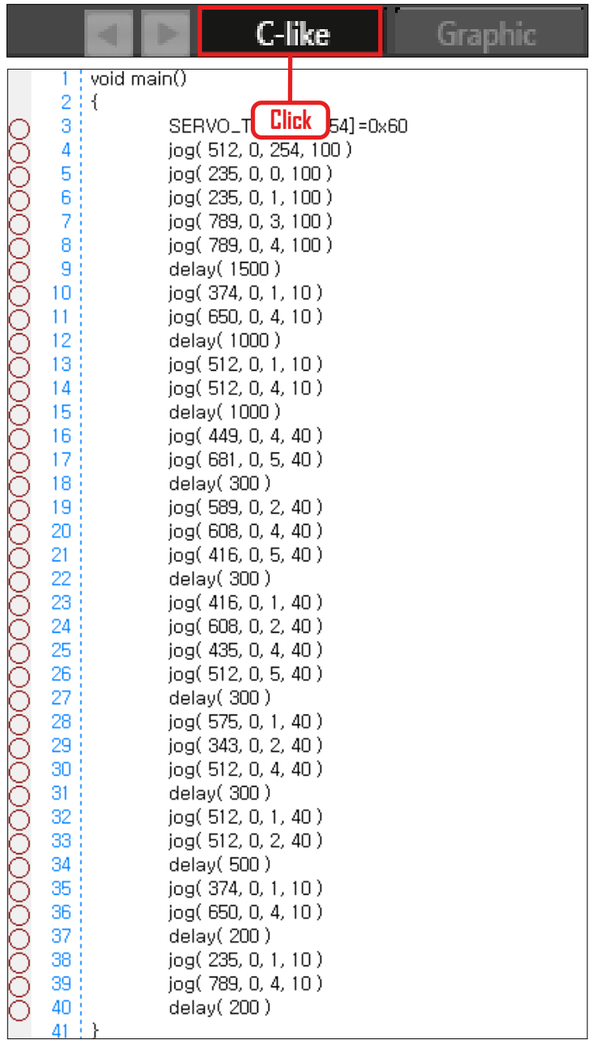
03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그래밍

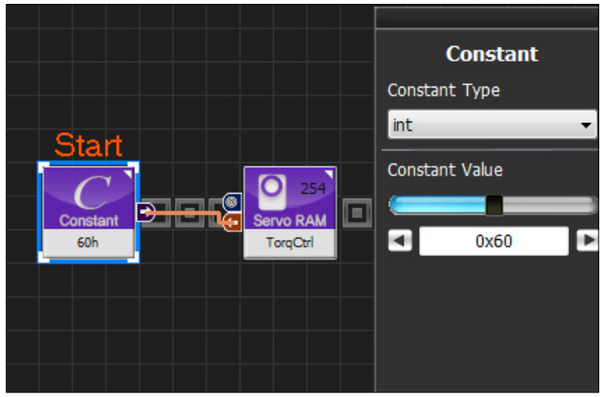
모터를 제어하는 전체 프로그래밍입니다.



05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

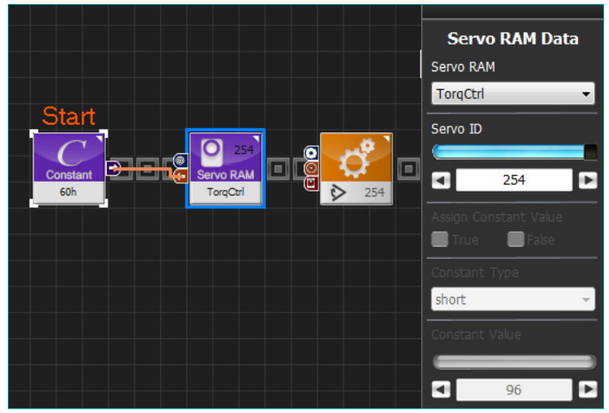
모터 모듈만을 사용한 모션 제어 프로그램입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다. 이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.

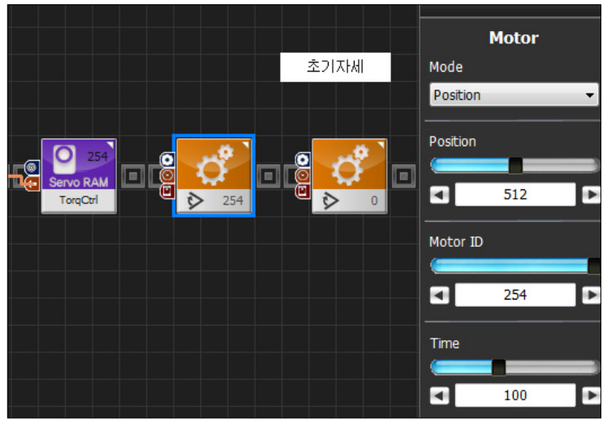


07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.

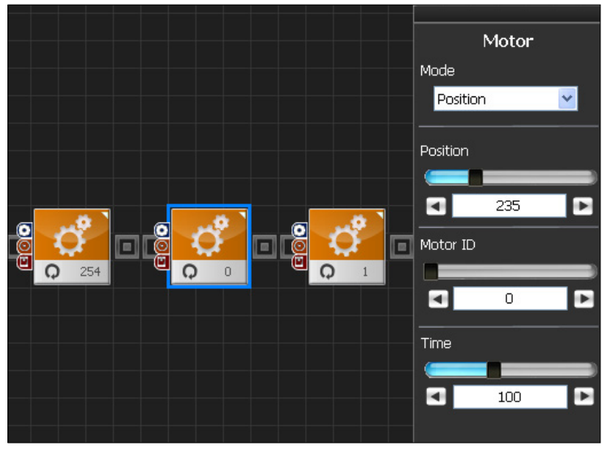
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



08 모든 서보 모터 위치 제어

모든 서보 모터의 위치를 중앙에 보내는 과정입니다.

Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 512로 설정합니다. 512는 모든 모터의 영점 위치로, 모터를 모두 중앙으로 보낸다는 의미입니다.
Motor ID : 254로 설정합니다. 254는 모든 모터에 적용하겠다는 의미입니다.
Time : 100으로 설정합니다. 단위는 1당 11.2ms로, 100은 약 1.12초를 의미합니다. 1.12초 동안 원하는 위치로 보낸다는 의미입니다.

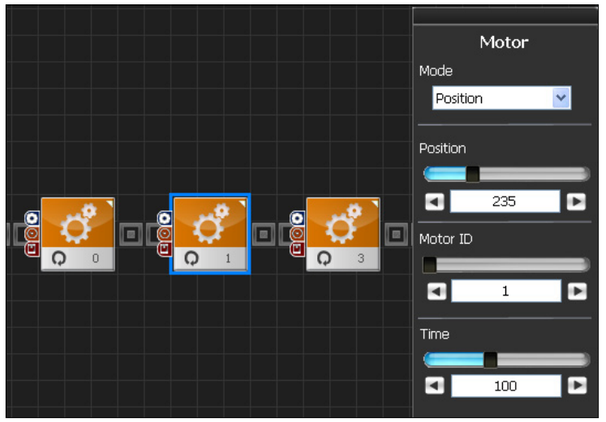


09 모터 0번(오른쪽 어깨) 설정

1단계 : 초기자세 잡기

모든 로봇의 모터의 각도를 중앙으로 정렬하면 휴머노이드에서는 팔을 좌우로 뻗게 됩니다. 이것을 차려 자세로 되돌려 놓아야만 기본 자세를 유지하여 동작시키기가 용이해집니다.

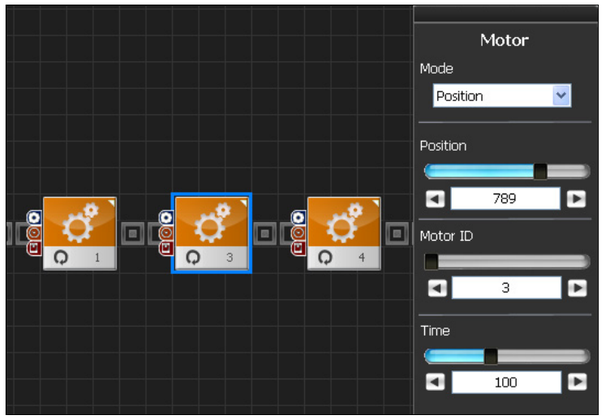
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 수평으로 들고 있던 오른팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



10 모터 1번(오른쪽 위팔) 설정

오른쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

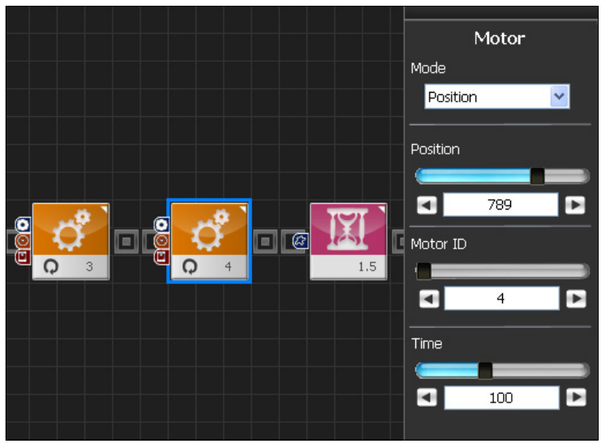
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 90도로 들고 있던 오른팔을 수직으로 내리는 위치 값입니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



11 모터 3번(왼쪽 어깨) 설정

왼쪽 어깨 모터를 돌려 왼팔을 수직으로 내려 갈 수 있는 위치로 바꿉니다.

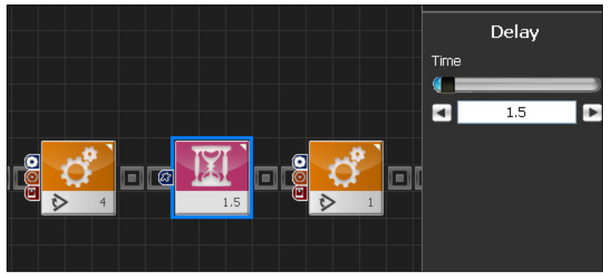
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 수평으로 들고 있던 왼팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



12 모터 4번(왼쪽 위팔) 설정

왼쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

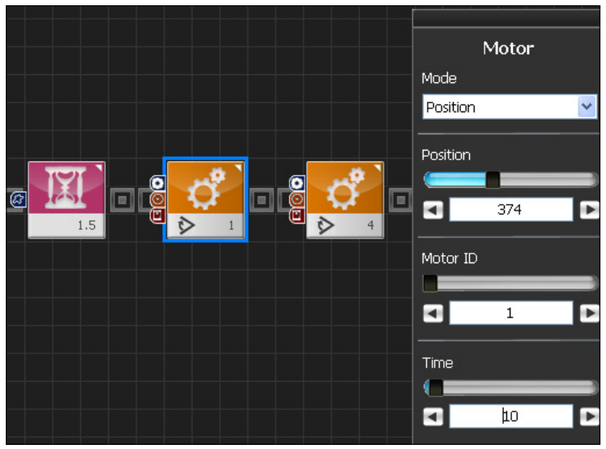
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 90도로 들고 있던 왼팔을 수직으로 내리는 위치 값입니다.
Motor ID : 4로 설정합니다. 왼쪽 위팔 모터 ID가 4번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



13 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 설정입니다. 직전 모듈들에 의해 모터가 움직이는 동안 아무 일도 하지 않고 기다립니다. 모터를 전체(254)를 512로 보내는 명령 후 0, 1, 3, 4를 따로 제어하는 명령을 바로 연달아 보냈으므로, 사용자가 보기에는 마치 동시에 차려 자세로 보낸 것처럼 움직입니다.

Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.

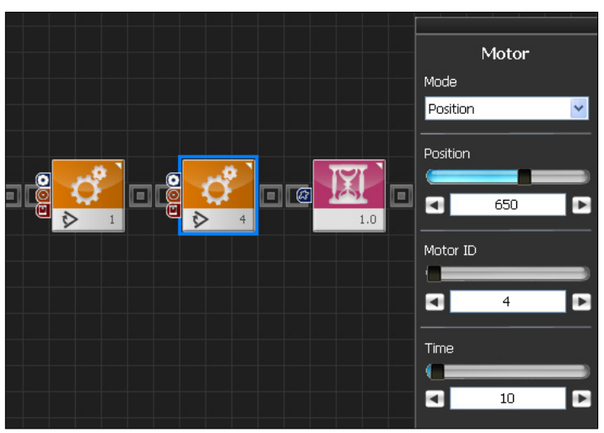


14 모터 1번(오른쪽 위팔) 설정

2단계 : 45도 각도로 팔 벌리기

로봇이 춤을 추기 위한 준비동작으로 팔을 45도 각도로 만듭니다.

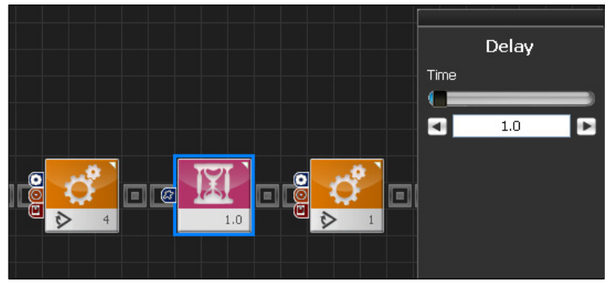
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 374로 설정합니다. 374는 오른쪽 위팔을 45도로 만드는 위치 값 입니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 10으로 설정합니다. 약 0.112초 동안 원하는 위치로 이동합니다.



15 모터 4번(왼쪽 위팔) 설정

마찬가지로 왼쪽 위팔 모터인 4번을 45도로 만듭니다.

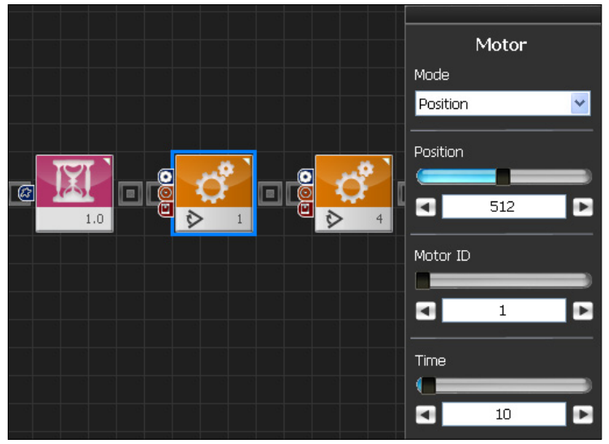
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 650로 설정합니다. 650는 왼쪽 위팔을 45도로 보내는 위치 값 입니다.
Motor ID : 4로 설정합니다. 왼쪽 팔 윗부분 모터 ID가 4번입니다.
Time : 10으로 설정합니다. 약 0.112초 동안 원하는 위치로 이동합니다.



16 Delay

다음 동작 전에 1.0초를 기다린 후 시작하는 모듈입니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 1.0으로 설정합니다.

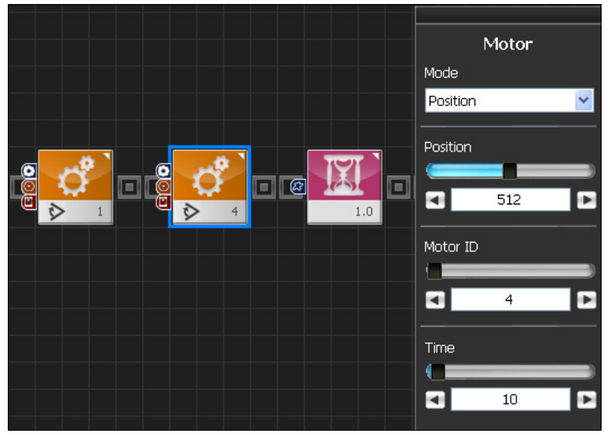


17 모터 1번(오른쪽 위팔) 설정

3단계 : 90도 각도로 팔벌리기

로봇의 팔을 90도로 올려서 웨이브 댄스추기 시작단계로 접어듭니다.

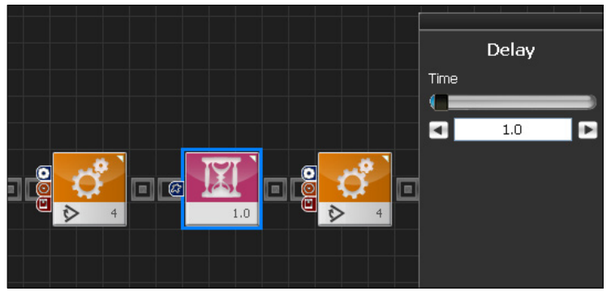
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 512로 설정합니다. 512는 오른쪽 위팔을 90도로 만드는 위치 값 입니다. 모터의 중앙값이기도 합니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 10으로 설정합니다. 약 0.112초 동안 원하는 위치로 이동합니다.



18 모터 4번(왼쪽 위팔) 설정

마찬가지로 왼쪽 위팔 모터인 4번 모터를 90도로 만듭니다.

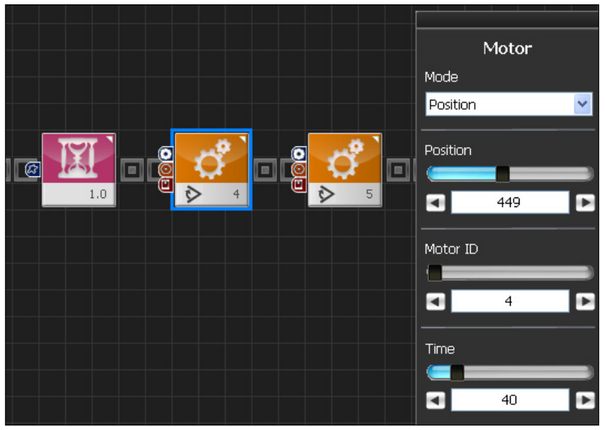
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 512로 설정합니다. 512는 왼쪽 위팔을 90도로 만드는 위치 값 입니다. 모터의 중앙값이기도 합니다.
Motor ID : 4로 설정합니다. 왼쪽 팔 윗부분 모터 ID가 4번입니다.
Time : 10으로 설정합니다. 약 0.112초 동안 원하는 위치로 이동합니다.



19 Delay

다음 동작 전에 1.0초를 기다린 후 시작하는 모듈입니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 1.0으로 설정합니다.

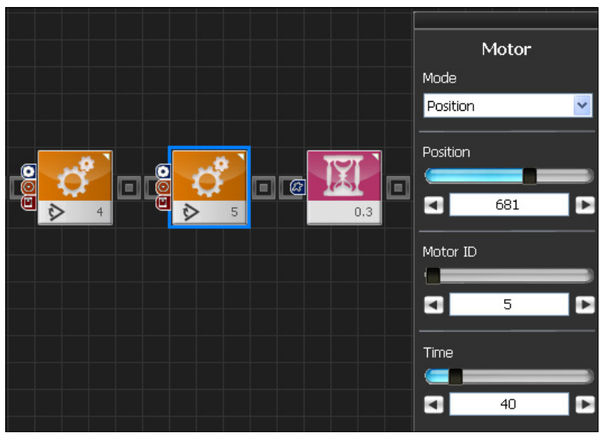


20 모터 4번(왼쪽 위팔) 설정

4단계 : 웨이브 1단계

왼쪽 팔부터 웨이브를 시작합니다.

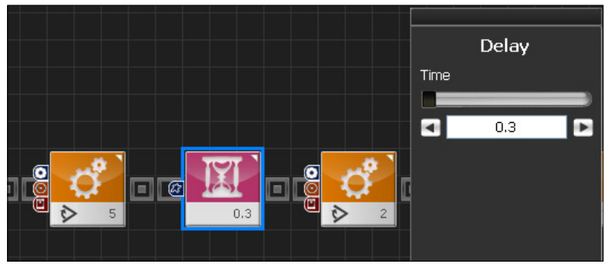
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 449로 설정합니다.
Motor ID : 4로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



21 모터 5번(왼쪽 아래팔) 설정

왼쪽 아래팔 모터인 5번을 제어합니다.

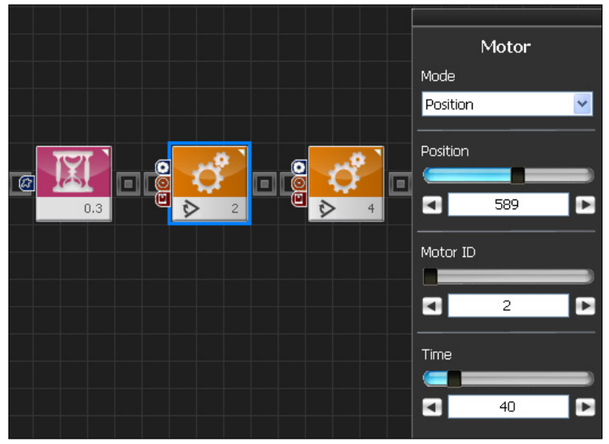
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 681(호비스 라이트) /
518(호비스 에코) 로 설정합니다.
Motor ID : 5로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



22 Delay

다음 동작 전에 0.3초를 기다립니다. 기다리는 시간을 모터의 이동 시간보다 적게 주어 모션을 부드럽게 만듭니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.3으로 설정합니다.

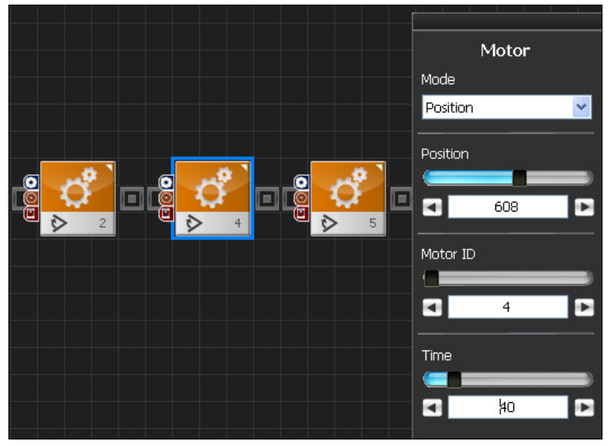


23 모터 2번(오른쪽 아래팔) 설정

5단계 : 웨이브 2단계

웨이브 2단계 위치로 이동합니다. 오른쪽 아래팔인 2번 모터도 웨이브를 시작합니다.

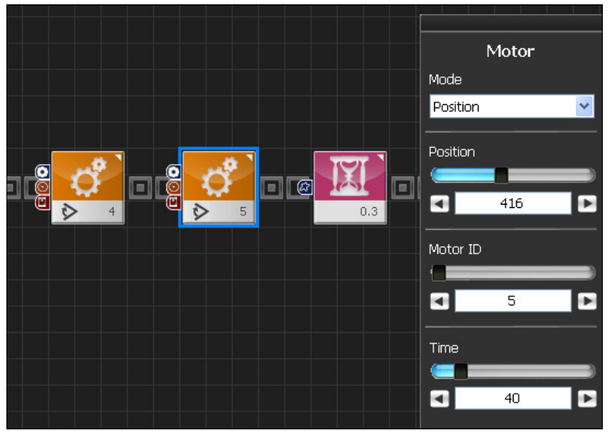
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 589(호비스 라이트) /
514(호비스 에코) 로 설정합니다.
Motor ID : 2로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



24 모터 4번(왼쪽 위팔) 설정

모터의 각도를 조금씩 변화시켜 웨이브 효과를 냅니다.

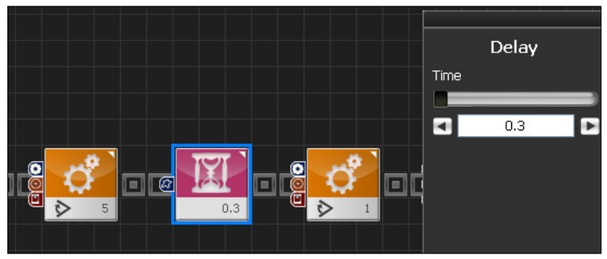
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 608로 설정합니다.
Motor ID : 4로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



25 모터 5번(왼쪽 아래팔) 설정

왼쪽 아래팔 모터인 5번을 제어합니다.

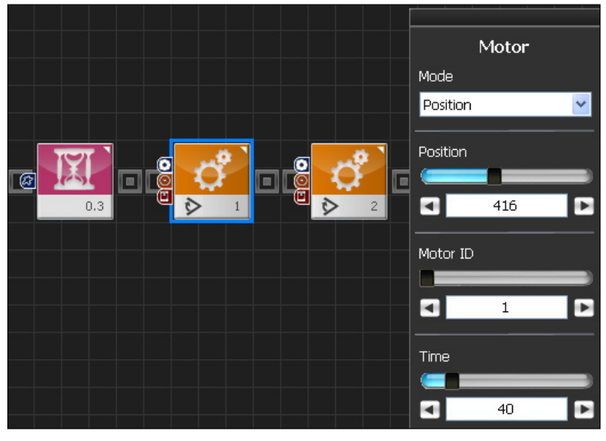
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 416(호비스 라이트) /
506(호비스 에코) 로 설정합니다.
Motor ID : 5로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



26 Delay

다음 동작 전에 0.3초를 기다립니다. 기다리는 시간을 모터의 이동 시간보다 적게 주어 모션을 부드럽게 만듭니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.3으로 설정합니다.

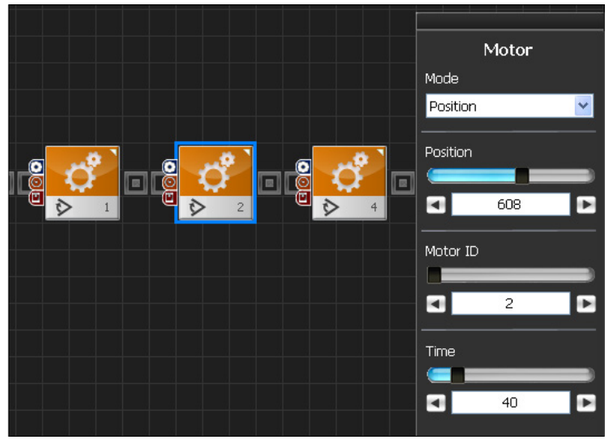


27 모터 1번(오른쪽 위팔) 설정

6단계 : 웨이브 3단계

오른쪽 위팔까지 웨이브가 전달됩니다.

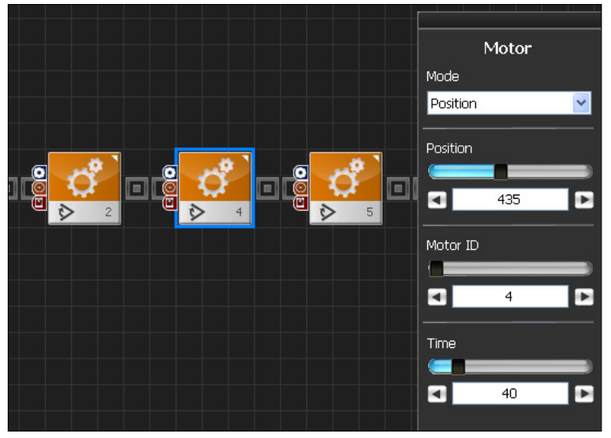
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 416으로 설정합니다.
Motor ID : 1로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



28 모터 2번(오른쪽 아래팔) 설정

오른쪽 아래팔 모터인 2번을 제어합니다.

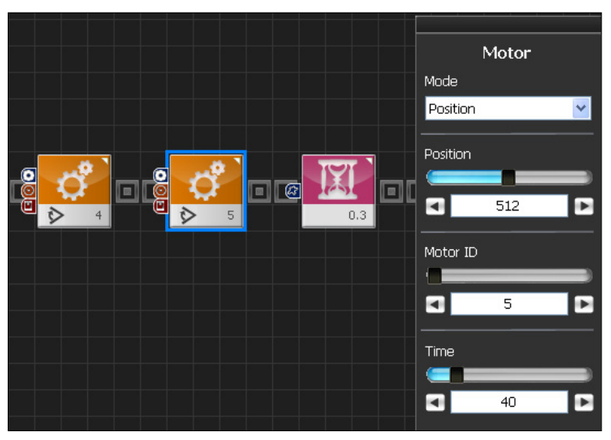
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 608(호비스 라이트) /
518(호비스 에코) 로 설정합니다.
Motor ID : 2로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



29 모터 4번(왼쪽 위팔) 설정

왼쪽 위팔 모터인 4번을 제어합니다.

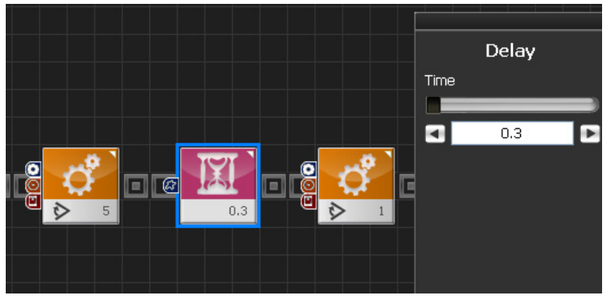
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 435로 설정합니다.
Motor ID : 4로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



30 모터 5번(왼쪽 아래팔) 설정

왼쪽 아래팔 모터인 5번을 512 위치로 원상 복귀 시킵니다.

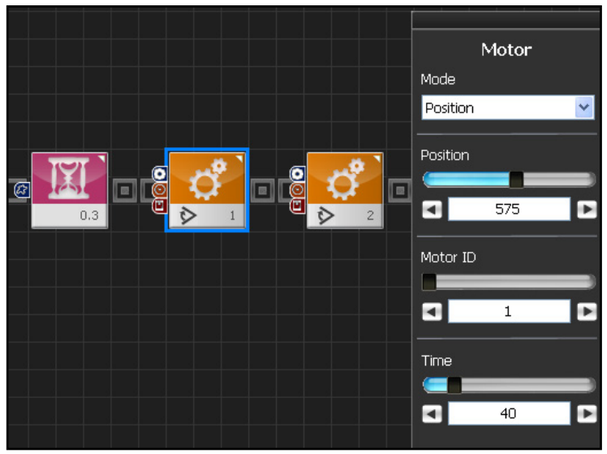
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 512로 설정합니다.
Motor ID : 5로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



31 Delay

다음 동작 전에 0.3초를 기다립니다. 기다리는 시간을 모터의 이동 시간보다 적게 주어 모션을 부드럽게 만듭니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.3으로 설정합니다.

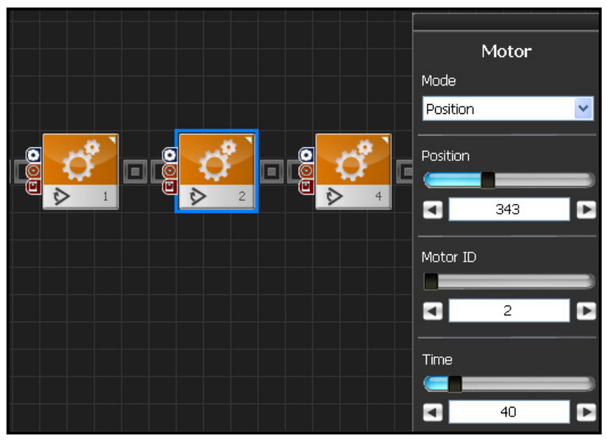


32 모터 1번(오른쪽 위팔) 설정

7단계 : 웨이브 4단계

왼팔은 웨이브를 마무리 하며 오른팔에 마지막 웨이브를 합니다.

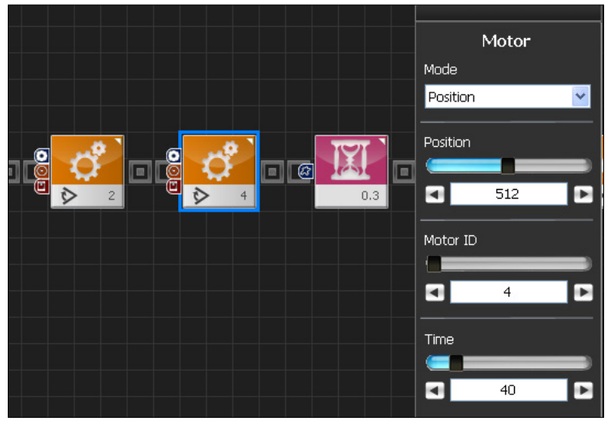
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position :575로 설정합니다.
Motor ID : 1로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



33 모터 2번(오른쪽 아래팔) 설정

오른쪽 아래팔 모터인 2번을 제어합니다.

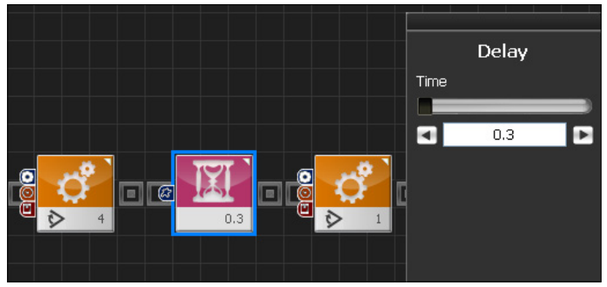
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 343(호비스 라이트) /
518(호비스 에코) 로 설정합니다.
Motor ID : 2로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



34 모터 4번(왼쪽 위팔) 설정

4번 모터를 512 위치로 원상 복귀 시킵니다.

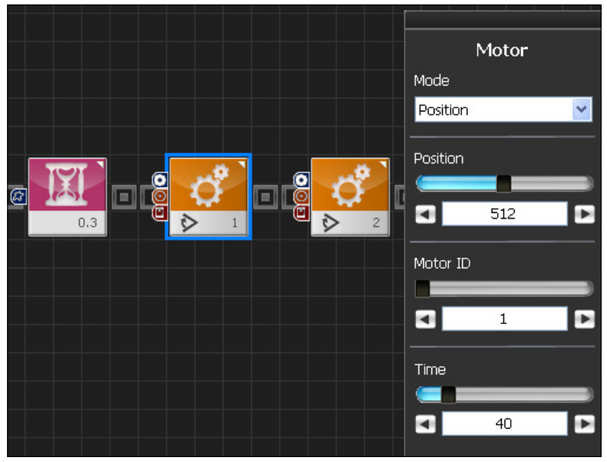
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 512로 설정합니다.
Motor ID : 4로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



35 Delay

다음 동작 전에 0.3초를 기다립니다. 기다리는 시간을 모터의 이동 시간보다 적게 주어 모션을 부드럽게 만듭니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.3으로 설정합니다.

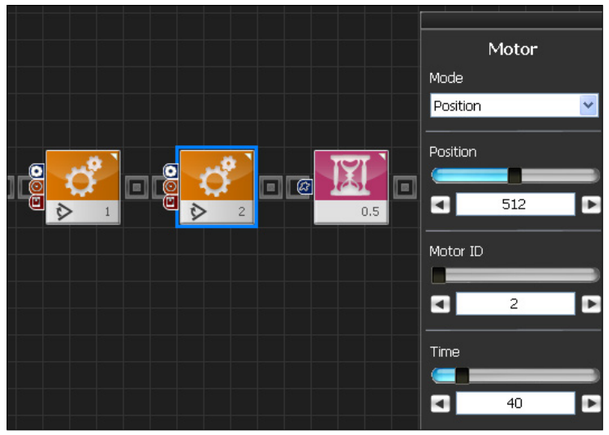


36 모터 1번(오른쪽 위팔) 설정

8단계 : 웨이브 5단계

로봇의 팔을 처음의 양팔 벌린 자세로 원상 복귀 시킵니다.

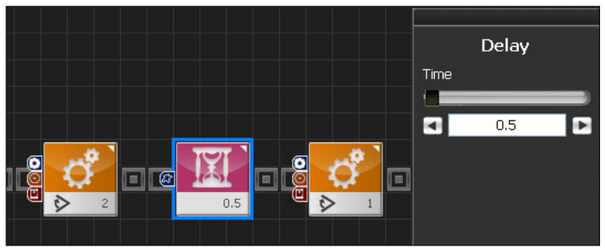
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position :512로 설정합니다.
Motor ID : 1로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



37 모터 2번(오른쪽 아래팔) 설정

2번 모터를 512 위치로 원상 복구 시킵니다.

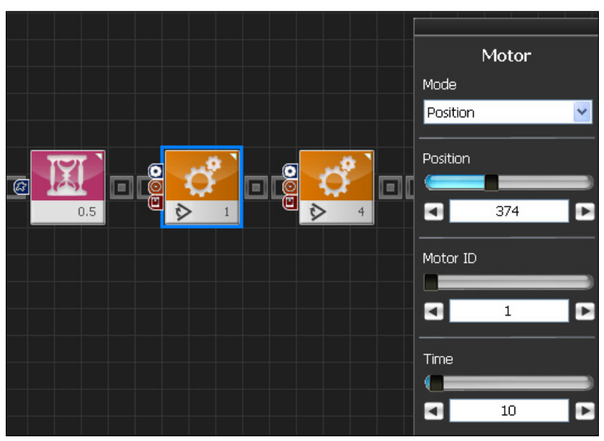
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 512로 설정합니다.
Motor ID : 2로 설정합니다.
Time : 40으로 설정합니다. 약 0.448초 동안 원하는 위치로 이동합니다.



38 Delay

다음 동작 전에 0.5초를 기다립니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.5으로 설정합니다.

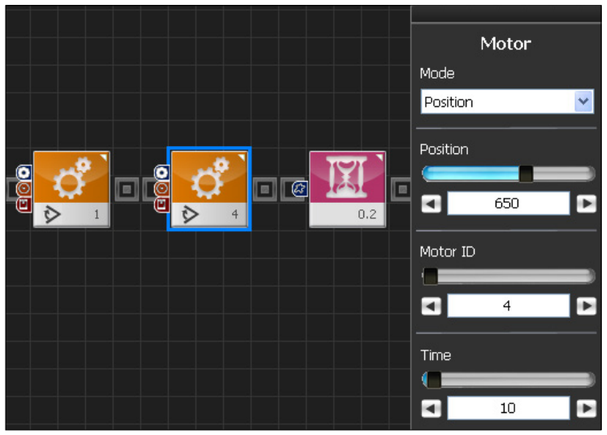


39 모터 1번(오른쪽 위팔) 설정

9단계 : 45도 각도로 팔내리기

차려 자세로 복귀 시키기 위해, 팔을 45도 각도로 내립니다.

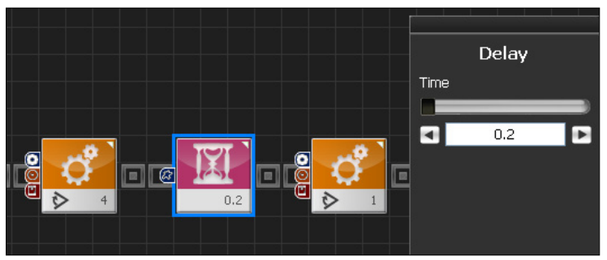
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 374로 설정합니다.
Motor ID : 1로 설정합니다.
Time : 10으로 설정합니다. 약 0.112초 동안 원하는 위치로 이동합니다.



40 모터 4번(왼쪽 위팔) 설정

마찬가지로 왼쪽 위팔 모터인 4번을 45도로 만듭니다.

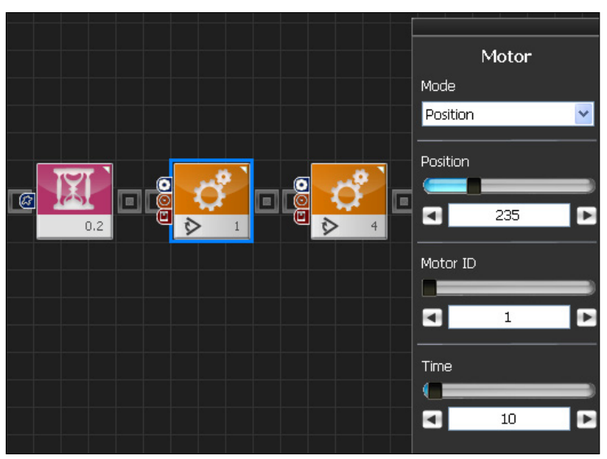
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 650로 설정합니다.
Motor ID : 4로 설정합니다.
Time : 10으로 설정합니다. 약 0.112초 동안 원하는 위치로 이동합니다.



41 Delay

다음 동작 전에 0.2초를 기다립니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.2로 설정합니다.

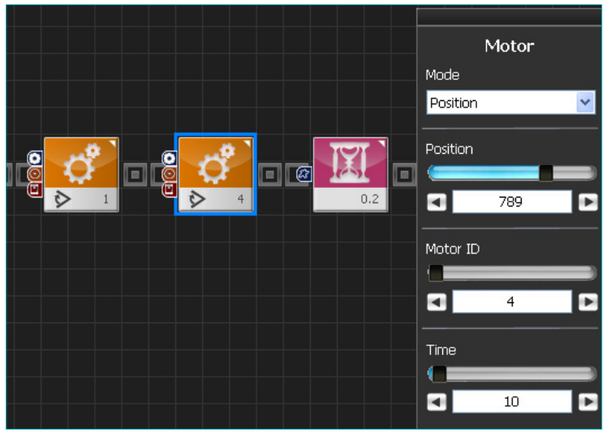


42 모터 1번(오른쪽 위팔) 설정

10단계 : 춤 완료

차려 자세로 복귀 시킵니다.

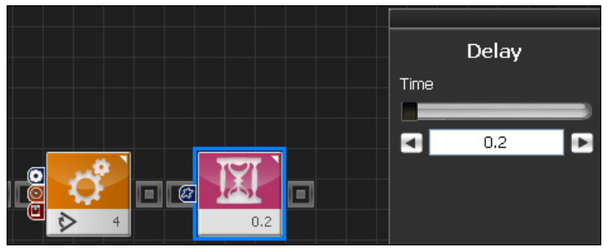
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 235로 설정합니다.
Motor ID : 1로 설정합니다.
Time : 10으로 설정합니다.



43 모터 4번(왼쪽 위팔) 설정

차려 자세로 복귀 시킵니다.

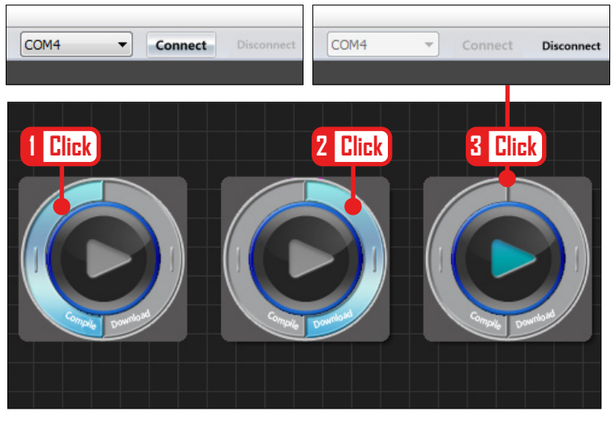
Motion 〉 Motor를 선택합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다.
Motor ID : 4로 설정합니다.
Time : 10으로 설정합니다.



44 Delay

동작을 마치기 까지 0.2초를 기다립니다.

Flow 〉 Delay 모듈을 선택합니다.
Time : 0.2로 설정합니다.

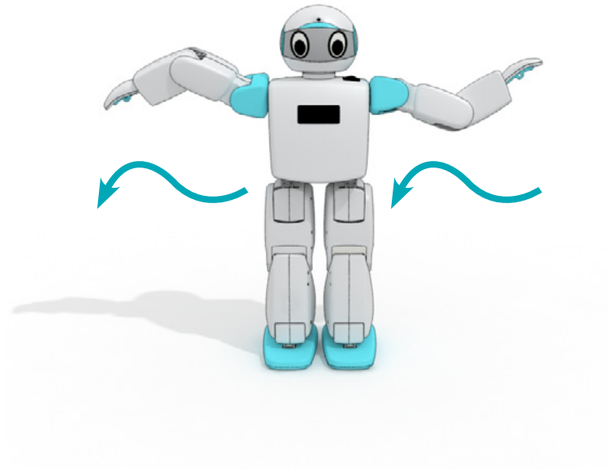


45 다운로드

프로그래밍 후 컴파일 –〉 로봇에 다운로드 –〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



46 로봇동작

로봇이 왼팔부터 웨이브를 합니다.

08-3 모듈별 프로그래밍 Button, LED

예제설명

DRC 제어기에 있는 Button을 이용하여 LED를 켰다 끄는 프로그램을 작성합니다.
Button과 LED 프로그래밍을 하기 위해선 Button과 LED가 작동할 때 해당하는 레지스터 항목이 어떻게 작동하는 지 먼저 알아야 합니다.

$16 = 2^4$

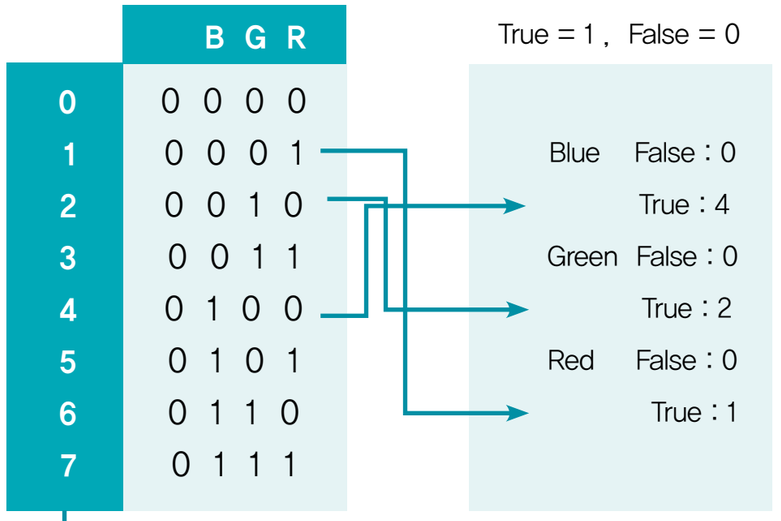
16 진수	0	4
2진수	0 0 0 0	0 1 0 0
Button	Right Left	Down Up Ok Mode

EX)

Right	2	0	h
	0 0 1 0	0 0 0 0	
Down	0	8	h
	0 0 0 0	1 0 0 0	

Button

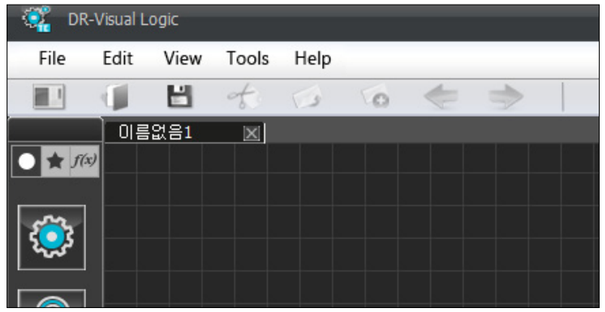
DRC에는 6개의 버튼이 있으며, 버튼의 눌린 상태를 1Byte로 표시합니다. 1Byte는 8Bit로 이루어져 있으므로, 1Byte에는 8개의 1과 0을 저장할 수 있습니다. DRC의 버튼 6개가 눌린 상태(1)과 눌리지 않은 상태(0)을 표시하기 위해서는 6Bit가 필요합니다. 위 그림에서 보는 바와 같이, 각 버튼은 Bit 하나로 매칭되어 있습니다.
버튼이 눌린 상태는 1과 0으로 표시되며, 이는 Button 모듈 우측 하단에 16진수로 표시됩니다. Right 버튼이 눌린 경우 버튼 값은 00100000이 되며, 이를 16진수로 바꾸면 20h입니다(h는 16진수라는 표시입니다). Down 버튼이 눌린 경우 버튼 값은 00001000이 되며, 이를 16진수로 바꾸면 08h입니다. 조금 더 복잡하게, Up+Down 버튼이 눌린 경우에 버튼 값은 00001100이 되며, 이를 16진수로 바꾸면 0Ch가 됩니다.



LED

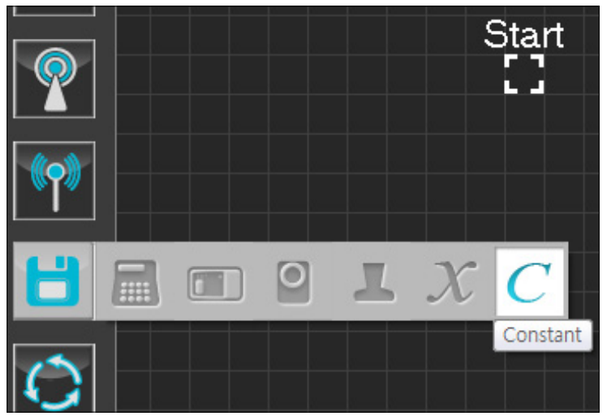
DRC에는 7개의 LED가 있지만, Task 모드에서 제어 가능한 것은 3개입니다. Red, Green, Blue의 LED 3개가 켜진 상태와 꺼진 상태를 표시하기 위해서는 3Bit가 필요합니다. 위 그림에서 보는 바와 같이, 각 LED는 바이트의 최하위 비트부터 Red, Green, Blue의 순서로 매칭되어 있습니다.
이 LED 값을 LED 모듈의 입력 값으로 넣으면, 해당하는 LED가 켜지게 됩니다. 가령 LED 모듈의 입력 값에 2를 넣었다면, 2는 2진수로 표시하면 00000010이므로 Green LED가 켜지게 됩니다. 비슷한 원리로 입력 값 0(00000000)은 LED가 다 꺼지게 만들며, 입력 값 7(00000111)은 LED가 켜지게 만듭니다.
Blue는 2진수에서 4의 자리에 해당하며, Green은 2의 자리, Red는 1의 자리에 해당합니다. 만약 Blue, Green, Red라는 변수가 있어서 이들의 값(true, false)에 따라 각 LED의 켜짐/꺼짐 여부가 결정된다면, LED 모듈의 입력 값으로 4 x Blue + 2 x Green + 1 x Red을 넣으면 변수 이름에 따라서 실제 LED를 제어할 수 있을 것입니다.
예를 들어서 Blue, Green이 true이고 Red가 false일 때 4 x Blue + 2 x Green + 1 x Red = 6이 됩니다. 6은 2진수로 표시하면 00000110이므로 이것을 LED 모듈의 입력 값으로 넣으면 Green, Blue LED가 켜지는 것입니다.

위 배경지식을 기반으로 Button 과 LED 프로그래밍을 해봅니다



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.



02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.

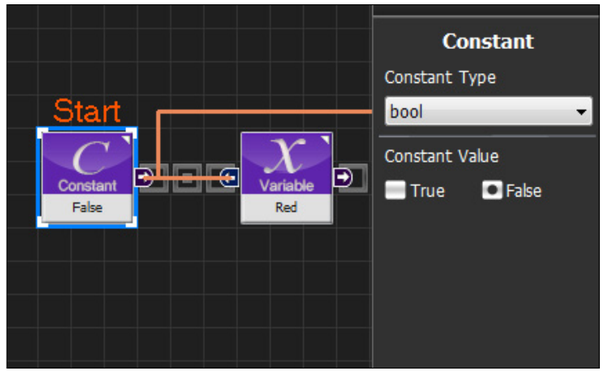
04 전체 프로그래밍

버튼과 LED를 이용한 전체 프로그램입니다.

05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

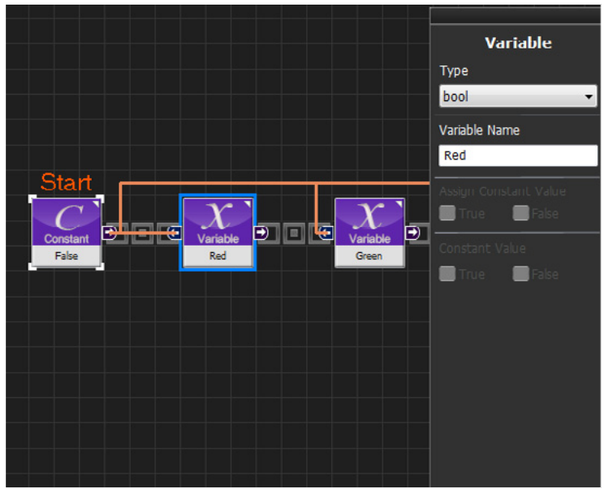
Button을 사용한 LED 제어 프로그램입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

모든 LED 변수를 꺼진 상태(false)로 초기화 하기 위해 상수를 설정합니다.

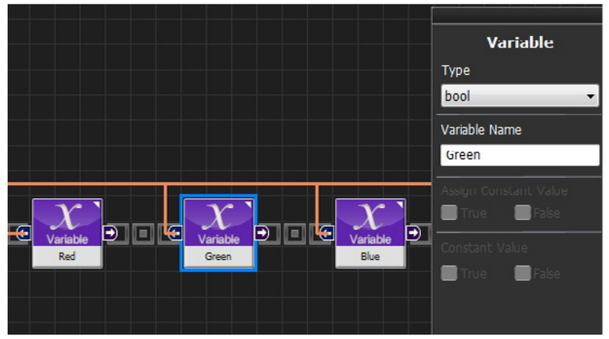
Constant 모듈의 속성 중 Constant Type을 bool로 바꿉니다.
Constant Value는 False로 설정합니다.
이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 Red 변수 초기화

적색 LED의 값을 저장할 변수를 초기화하기 위해 Variable 모듈을 만듭니다. True와 False의 두 가지 값을 가질 것이기 때문에 bool 타입을 사용합니다.

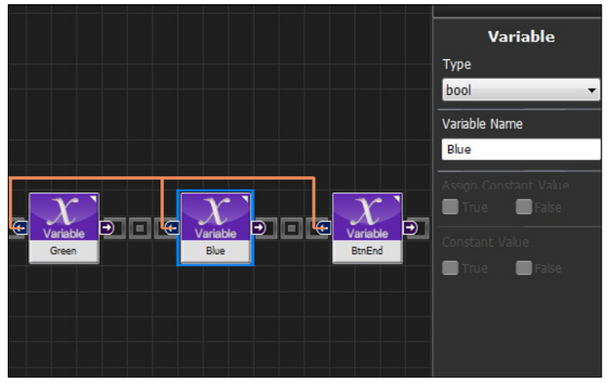
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Red를 입력합니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을
Variable 모듈의 입력 핀에 커넥터로 연결합니다.
그러면 Red 라는 변수에 False라는 값이 입력됩니다.



08 Green 변수 초기화

녹색 LED의 값을 저장할 변수를 초기화 하기 위해 Variable 모듈을 만듭니다. 마찬가지로 bool 타입을 사용합니다.

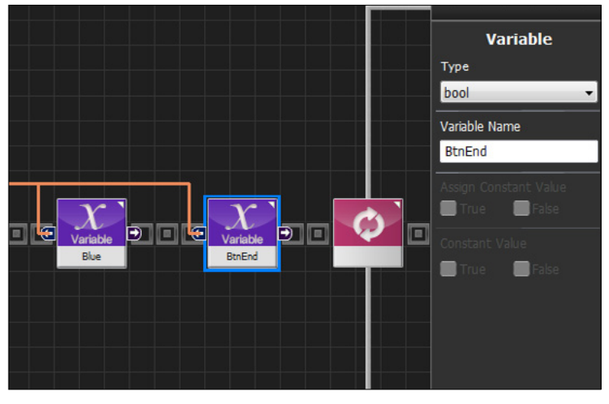
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Green을 입력합니다.
그리고 마찬가지로 앞에 있는 Constant 모듈의 출력 핀을 Variable 모듈의 입력 핀에 커넥터로 연결합니다.



09 Blue 변수 초기화

청색 LED의 값을 저장할 변수를 초기화 하기 위해 Variable 모듈을 만듭니다. 마찬가지로 bool 타입을 사용합니다.

Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Blue를 입력합니다.
마찬가지로 앞에 있는 Constant 모듈의 출력 핀을 Variable 모듈의 입력 핀에 커넥터로 연결합니다.



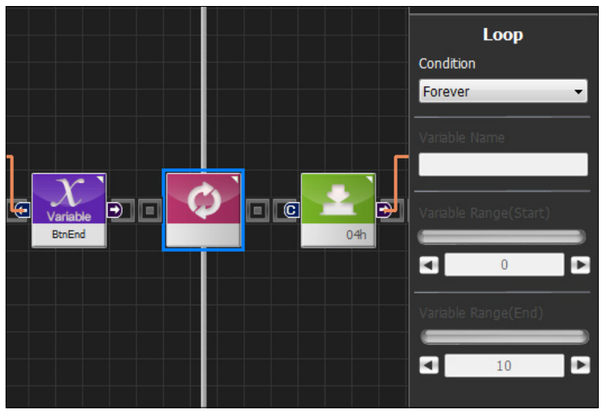
10 BtnEnd 변수 초기화

버튼이 눌렸을 때 원하는 동작을 다 마쳤는지 저장하는 변수입니다. 이 프로그램에서는 반복문 안에서 버튼을 눌렀을 때 LED 변수의 값을 바꾸는데, 반복문이 빨리 돌아가기 때문에 버튼을 한 번 누를 때 변수 값이 여러 번 변하게 됩니다.
그래서 이 변수를 선언해서 버튼을 눌렀을 때 변수 값 바꾸기가 완료 되었는지 저장해서, 한 번 누를 때 변수 값을 한 번만 바꾸도록 합니다.

Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
앞에 있는 Constant 모듈의 출력 핀을 Variable 모듈의 입력 핀에 커넥터로 연결합니다.

11 변수 초기화 완료

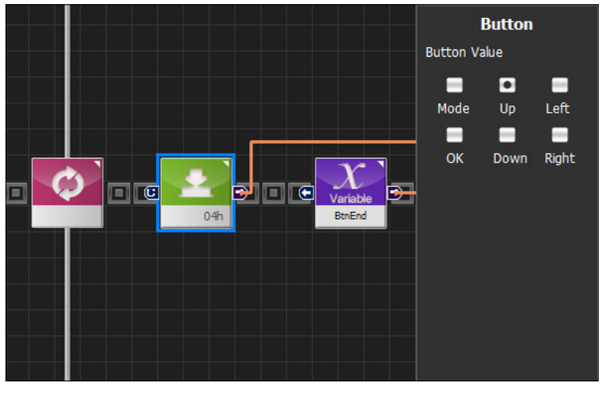
Red, Green, Blue, BtnEnd 변수를 모두 초기 값을 False로 설정했습니다. 이처럼 한 모듈의 출력 핀을 여러 모듈의 입력 핀과 연결할 수 있습니다.



12 Loop

프로그램의 메인 루프 모듈을 만듭니다.

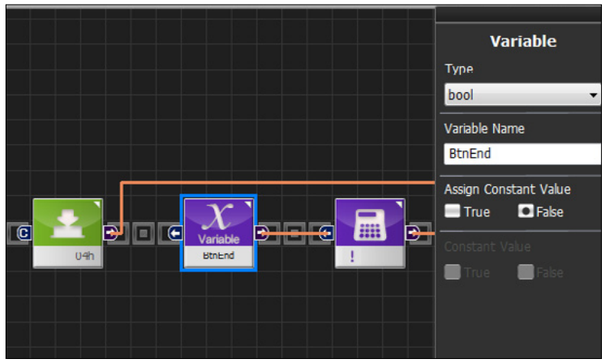
Flow 〉 Loop을 선택해서 이전 모듈의 뒤에 배치합니다.
Condition : Forever를 선택해서 조건 없이 무한 반복하도록 합니다.



13 Up 버튼

버튼 모듈을 만듭니다. 이 모듈의 출력 값은 선택한 버튼이 눌렸을 경우 True가 되고 그 외에는 False가 됩니다. Up Button을 선택 시, Up Button이 눌린 경우 True, 그 외에 False가 됩니다.

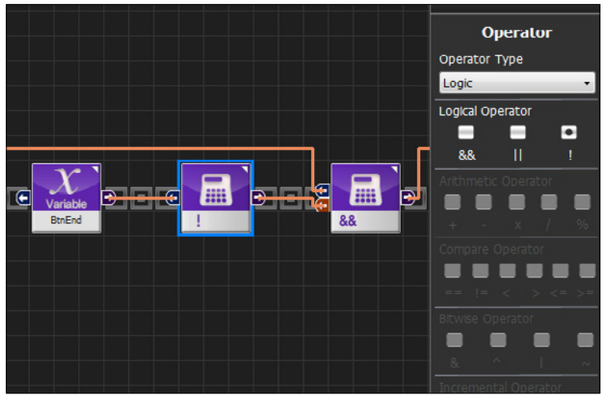
Communication 〉 Button 모듈을 선택해 Loop 모듈의 안 쪽에 배치합니다.
Button Value를 Up으로 선택합니다.
16진수로 04h 이므로 모듈 그림에 04h가 표기됩니다.



14 BtnEnd

BtnEnd 변수 값을 읽어서 조건문에서 사용해야 합니다.

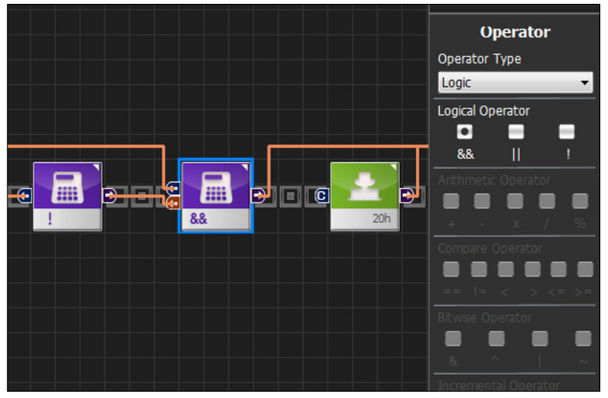
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
앞의 BtnEnd를 복사해서 붙여도 됩니다.



15 ! 연산

NOT 연산을 적용하여 BtnEnd 값을 반대로 바꿉니다.

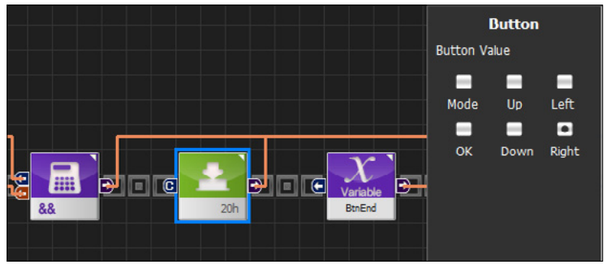
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : !(Logical NOT)로 선택합니다.
BtnEnd의 출력 핀과 NOT 모듈의 입력 핀을 연결합니다.



16 And 연산

AND 연산을 적용해 Up 버튼이 눌러졌고, BtnEnd가 false (NOT 적용하여 True)일 때 출력 값이 True가 되도록 합니다.

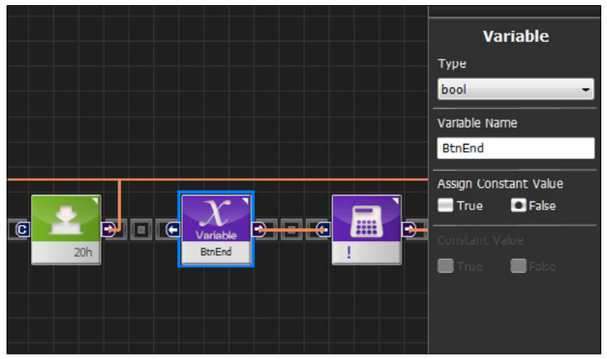
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : &&(Logical AND)로 선택합니다.
13번의 Button 모듈과 15번의 NOT 모듈의 출력을 각각 AND 모듈의 입력에 연결합니다.



17 Right 버튼

버튼 모듈을 만들어 Right 버튼을 선택합니다.

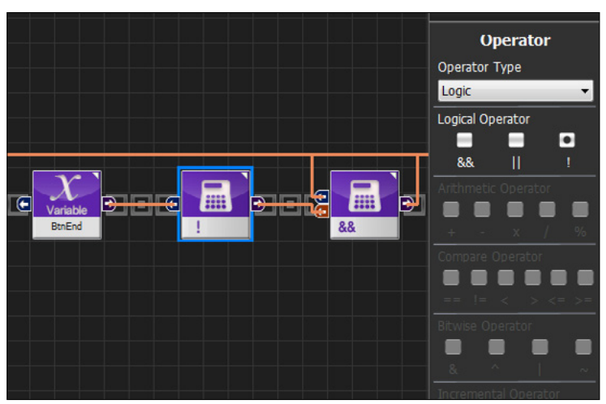
Communication 〉 Button 모듈을 선택해 직전 모듈의 뒤에 배치합니다.
Button Value를 Right으로 선택합니다.
16진수로 20h 이므로 모듈 그림에 20h가 표기됩니다.



18 BtnEnd

마찬가지로 BtnEnd 변수 값을 읽어서 조건문에서 사용해야 합니다.

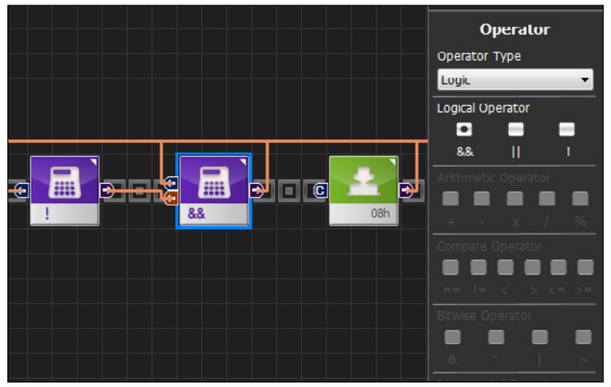
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
앞의 BtnEnd를 복사해서 붙여도 됩니다.



19 NOT 연산

NOT 연산을 적용하여 BtnEnd 값을 반대로 바꿉니다.

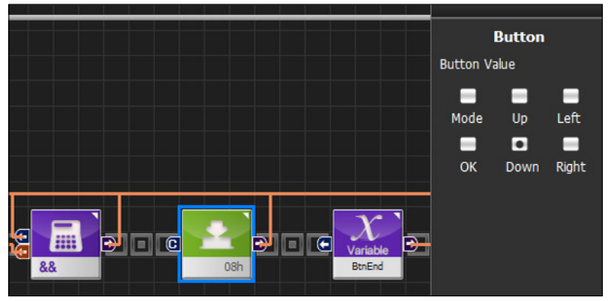
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : !(Logical NOT)로 선택합니다.
BtnEnd의 출력 핀과 NOT 모듈의 입력 핀을 연결합니다.



20 AND 연산

AND 연산을 적용해 Right 버튼이 눌러졌고, BtnEnd가 false (NOT 적용하여 True) 일 때 출력 값이 True가 되도록 합니다.

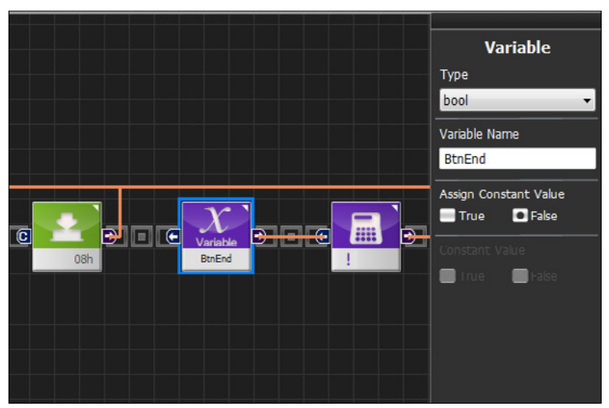
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : &&(Logical AND)로 선택합니다.
17번의 Button 모듈과 19번의 NOT 모듈의 출력을 각각 AND 모듈의 입력에 연결합니다.



21 Down 버튼

버튼 모듈을 만들어 Down 버튼을 선택합니다.

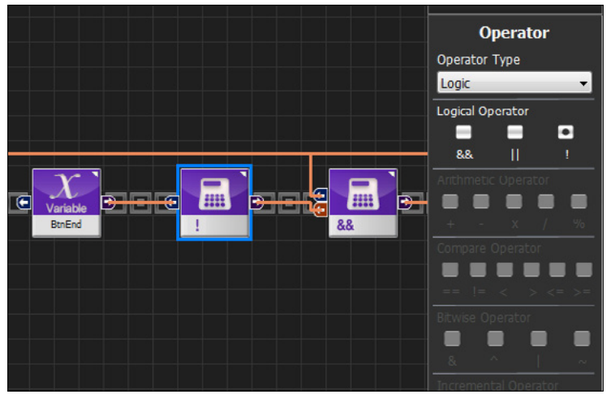
Communication 〉 Button 모듈을 선택해 직전 모듈의 뒤에 배치합니다.
Button Value를 Down으로 선택합니다.
16진수로 08h 이므로 모듈 그림에 08h가 표기됩니다.



22 BtnEnd

BtnEnd 변수 값을 읽어서 조건문에서 사용해야 합니다.

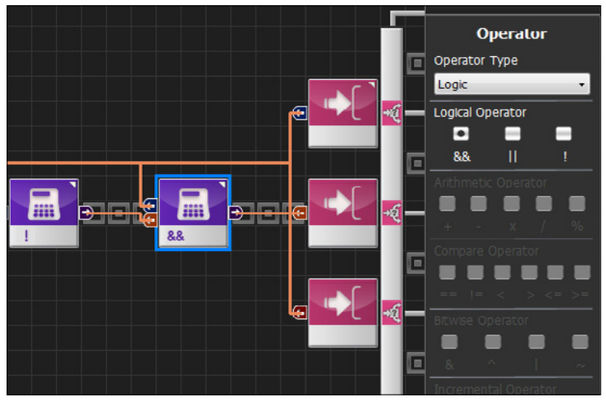
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.



23 NOT 연산

NOT 연산을 적용하여 BtnEnd 값을 반대로 바꿉니다.

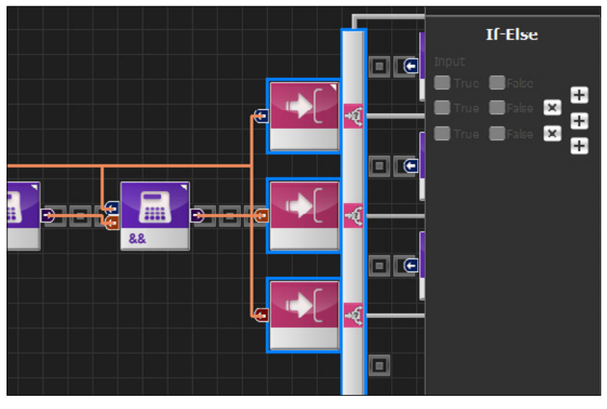
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : !(Logical NOT)로 선택합니다.
22번의 BtnEnd의 출력 핀과 NOT 모듈의 입력 핀을 연결합니다.



24 AND 연산

AND 연산을 적용해 Down 버튼이 눌러졌고, BtnEnd가 false (NOT 적용하여 True) 일 때 출력 값이 True가 되도록 합니다.

Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : &&(Logical AND)로 선택합니다.
21번의 Button 모듈과 23번의 NOT 모듈의 출력을 각각 AND 모듈의 입력에 연결합니다.



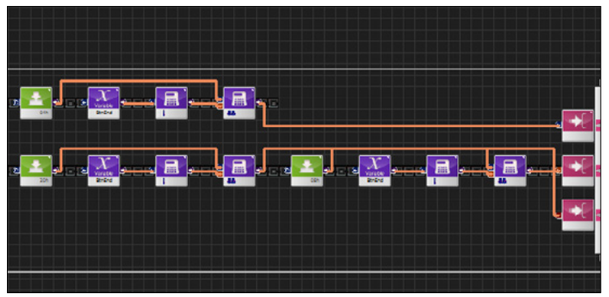
25 If-Else문 생성

If-Else 문을 생성하여 앞에서 만든 조건문을 If-Else문과 연결합니다.

Flow 〉 If-Else 모듈을 선택합니다.
+ 모양의 아이콘을 두 번 클릭해서 조건문의 개수를 2개 더 추가합니다.
16번의 AND 연산 출력 핀을 첫 입력 핀에, 20번의 출력 핀을 둘째 입력 핀에, 24번 출력 핀은 셋째 입력 핀에 연결합니다.

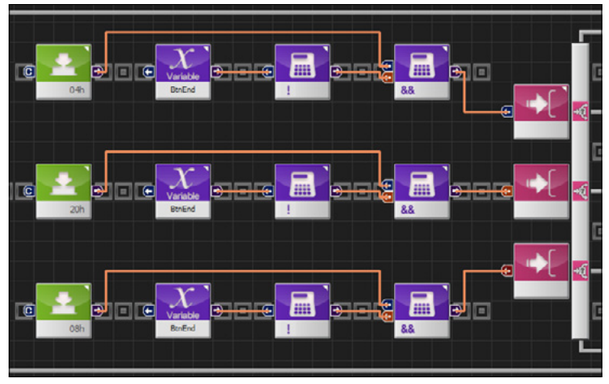
26 조건문 완성

Up 버튼, Right 버튼, Down 버튼을 사용한 If-Else 조건문이 완성되었습니다. 그러나 어떤 것이 어떤 입력에 들어가는 지 알아보기 힘들기 때문에, 모듈을 알아보기 쉽게 배치해 주는 편이 좋습니다.



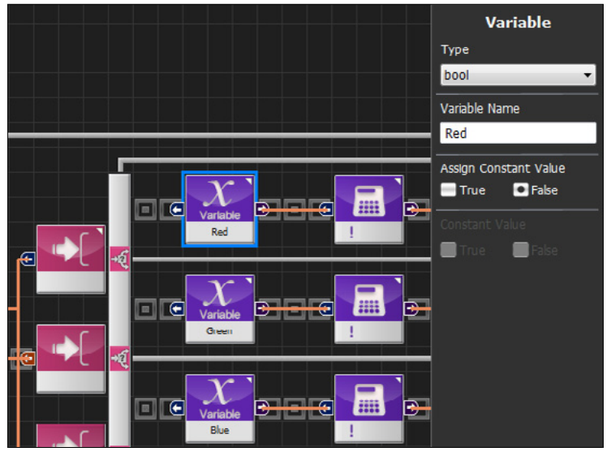
27 모듈 배치

13~16에서 생성한 4개 모듈로 작성된 조건문을 프로그램 라인 위로 옮깁니다. 출력 핀이 연결되었기 때문에 라인 밖으로 이동해도 활성화 상태가 유지됩니다.



28 모듈 배치

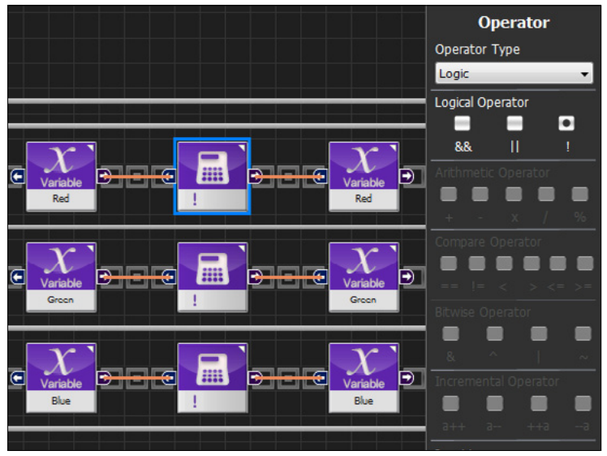
마찬가지로 21~24에서 생성한 4개 모듈로 작성된 조건문을 프로그램 라인 아래로 옮깁니다. 이렇게 배치하면 어떤 조건문이 어느 입력 핀에 연결되었는지 쉽게 알아볼 수 있습니다.



29 Red 변수 읽기

If-Else 모듈 안에서 실행될 모듈을 추가해야 합니다. Up 버튼 값과 연결된 첫째 입력 핀이 참이면 가장 위 프로그램 라인의 내용이 실행됩니다. Red 값을 바꾸기 위해서 우선 Red 변수 모듈을 만들어 값을 읽습니다.

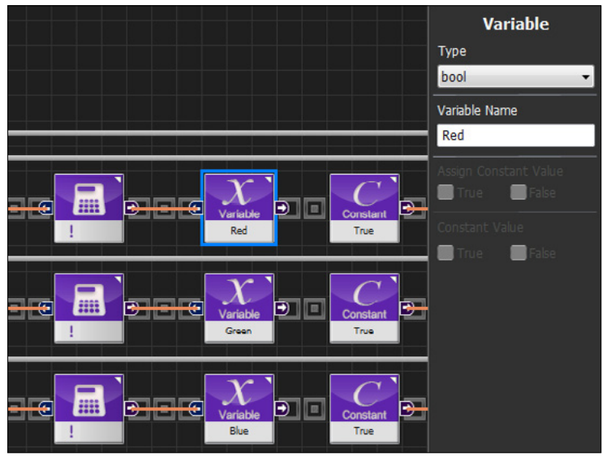
Data 〉 Variable을 선택해서 If-Else문의 가장 위 프로그램 라인에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Red를 입력합니다.
앞의 Red 변수를 복사해 와도 됩니다.



30 NOT 연산

NOT 연산을 적용하여 Red 값을 반대로 바꿉니다.

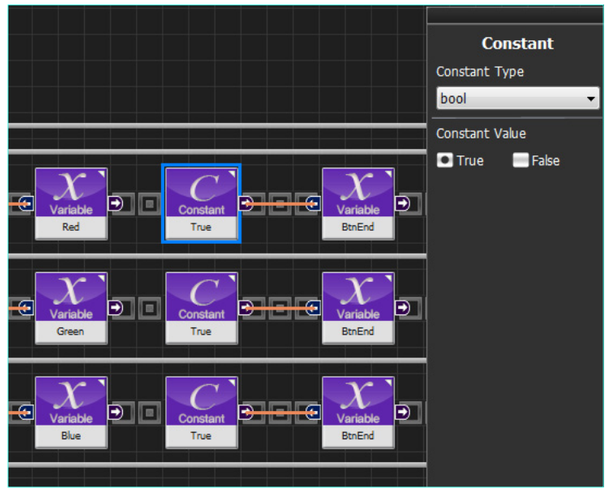
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : !(Logical NOT)로 선택합니다.
29번의 Red 출력 핀과 NOT 모듈의 입력 핀을 연결합니다.



31 Red 변수에 대입하기

반대로 바꾼 Red 값을 Red에 대입합니다.

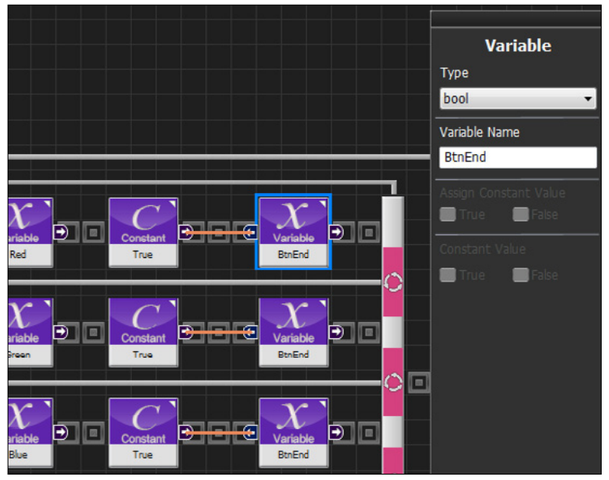
Data 〉 Variable을 선택해서 직전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Red를 입력합니다.
30번의 NOT 모듈 출력 핀을 Red 변수 입력 핀에 연결해서, 반대로 만든 Red 값을 다시 Red에 대입합니다.



32 True 값 만들기

Constant 모듈을 만들어 BtnEnd 변수에 대입할 True 값을 만듭니다.

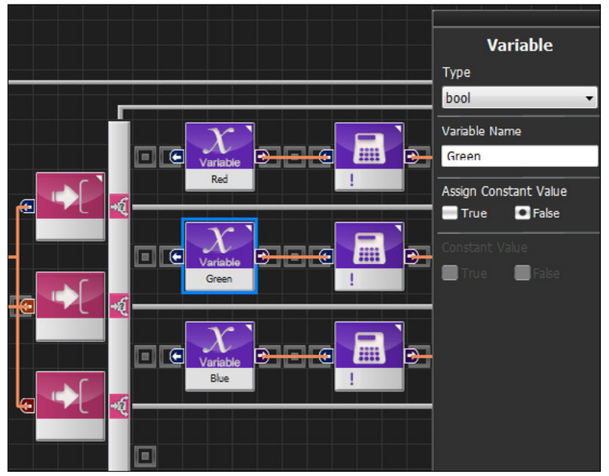
Data 〉 Constant를 선택하여 직전 모듈의 뒤에 배치합니다.
Constant 모듈의 속성 중 Constant Type을 bool로 바꿉니다.
Constant Value는 True로 설정합니다.



33 BtnEnd에 대입하기

BtnEnd에 True를 대입해서, Up 버튼을 누른 것에 대한 처리가 끝났다는 것을 알립니다. BtnEnd 값이 True면 13~16번에서 구성한 조건문이 참이 될 수 없으므로, 이제 버튼을 떼기 전까지는 다시 Red 값이 바뀌지 않습니다.

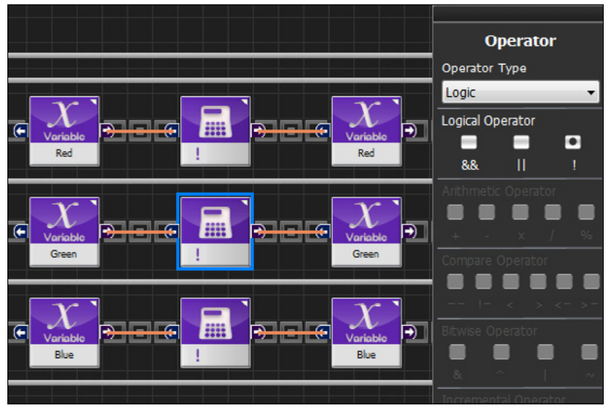
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
32번의 Constant 출력 핀을 BtnEnd의 입력 핀에 연결합니다.



34 Green 변수 읽기

첫째 입력 핀이 거짓이고, 둘째 입력 핀이 참인 경우 둘째 프로그램 라인의 내용이 실행됩니다. Green 값을 바꾸기 위해서 우선 Green 변수 모듈을 만들어 값을 읽습니다.

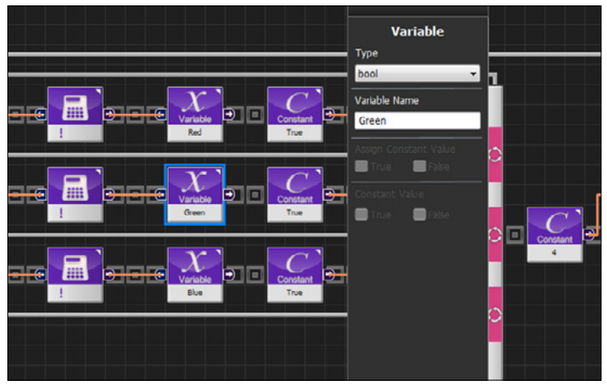
Data 〉 Variable을 선택해서 If-Else문의 둘째 프로그램 라인에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Green을 입력합니다.
앞의 Green 변수를 복사해 와도 됩니다.



35 NOT 연산

NOT 연산을 적용하여 Green 값을 반대로 바꿉니다.

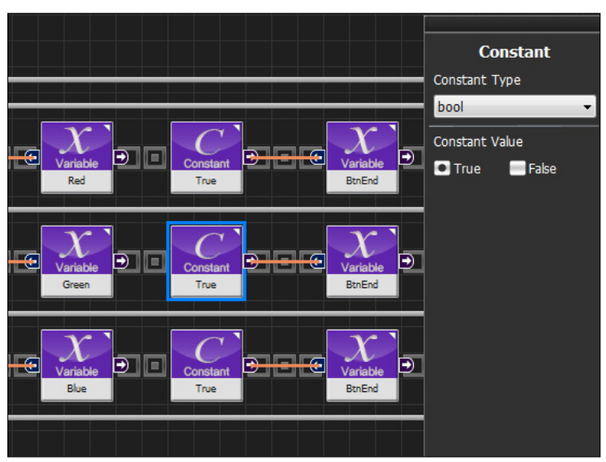
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : !(Logical NOT)로 선택합니다.
34번의 Green 출력 핀과 NOT 모듈의 입력 핀을 연결합니다.



36 Green 변수에 대입하기

반대로 바꾼 Green 값을 Green에 대입합니다.

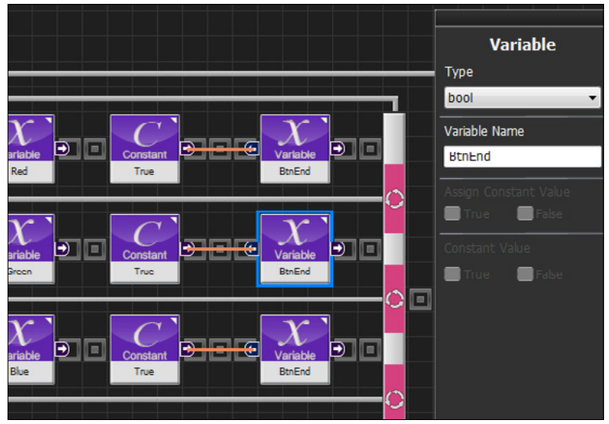
Data 〉 Variable을 선택해서 직전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Green을 입력합니다.
35번의 NOT 모듈 출력 핀을 Green 변수 입력 핀에 연결해서, 반대로 만든 Green 값을 다시 Green에 대입합니다.



37 True 값 만들기

Constant 모듈을 만들어 BtnEnd 변수에 대입할 True 값을 만듭니다.

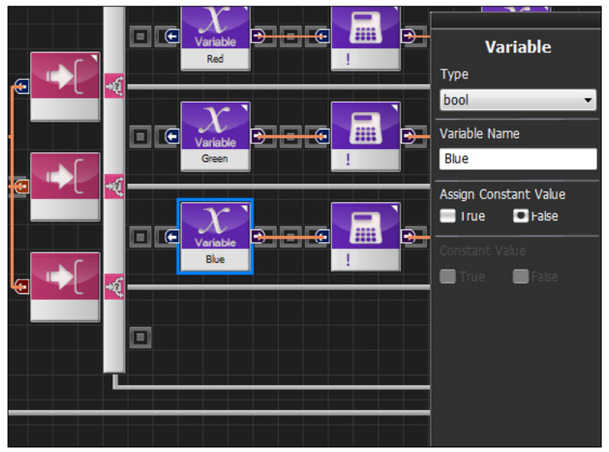
Data 〉 Constant를 선택하여 직전 모듈의 뒤에 배치합니다.
Constant 모듈의 속성 중 Constant Type을 bool로 바꿉니다.
Constant Value는 True로 설정합니다.



38 BtnEnd에 대입하기

BtnEnd에 True를 대입해서, Right 버튼을 누른 것에 대한 처리가 끝났다는 것을 알립니다. BtnEnd 값이 True면 17~20번에서 구성한 조건문이 참이 될 수 없으므로, 이제 버튼을 떼기 전까지는 다시 Green 값이 바뀌지 않습니다.

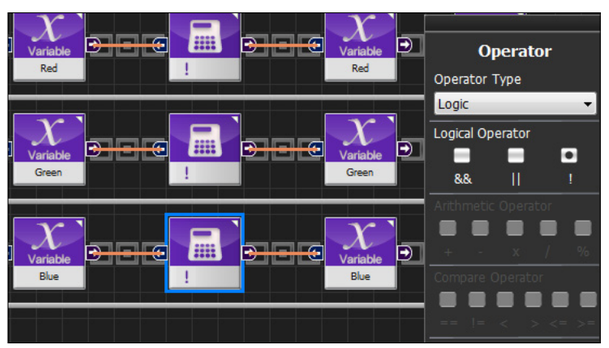
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
37번의 Constant 출력 핀을 BtnEnd의 입력 핀에 연결합니다.



39 Blue 변수 읽기

첫째, 둘째 입력 핀이 거짓이고, 셋째 입력 핀이 참인 경우 세 번째 프로그램 라인의 내용이 실행됩니다. Blue 값을 바꾸기 위해서 우선 Blue 변수 모듈을 만들어 값을 읽습니다.

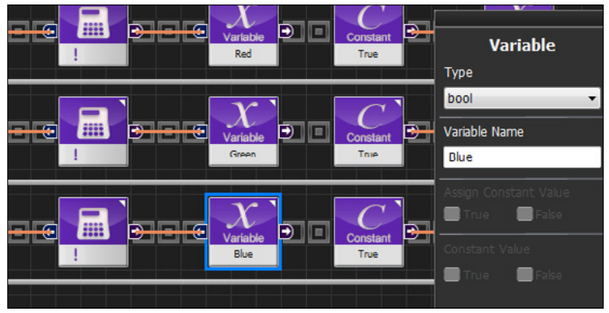
Data 〉 Variable을 선택해서 If-Else문의 세 번째 프로그램 라인에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Blue를 입력합니다.
앞의 Blue 변수를 복사해 와도 됩니다.



40 NOT 연산

NOT 연산을 적용하여 Blue 값을 반대로 바꿉니다.

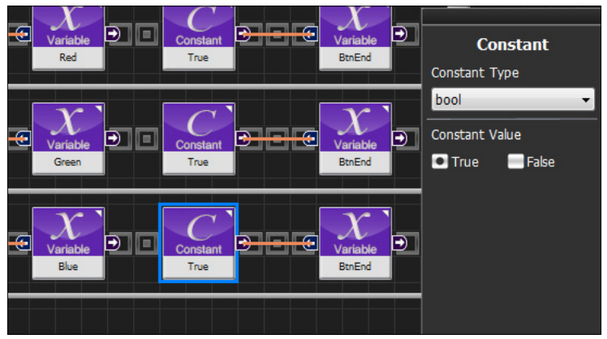
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : !(Logical NOT)로 선택합니다.
39번의 Blue 출력 핀과 NOT 모듈의 입력 핀을 연결합니다.



41 Blue 변수에 대입하기

반대로 바꾼 Blue 값을 Blue에 대입합니다.

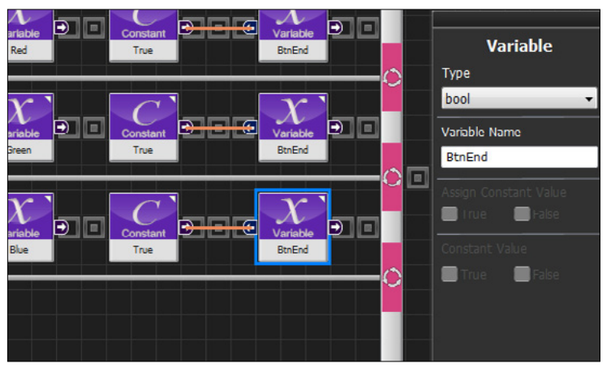
Data 〉 Variable을 선택해서 직전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Blue를 입력합니다.
35번의 NOT 모듈 출력 핀을 Blue 변수 입력 핀에 연결해서, 반대로 만든 Blue 값을 다시 Blue에 대입합니다.



42 True 값 만들기

Constant 모듈을 만들어 BtnEnd 변수에 대입할 True 값을 만듭니다.

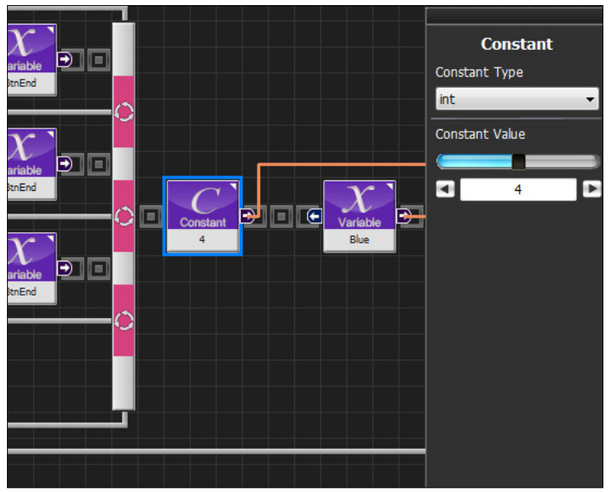
Data 〉 Constant를 선택하여 직전 모듈의 뒤에 배치합니다.
Constant 모듈의 속성 중 Constant Type을 bool로 바꿉니다.
Constant Value는 True로 설정합니다.



43 BtnEnd에 대입하기

BtnEnd에 True를 대입해서, Down 버튼을 누른 것에 대한 처리가 끝났다는 것을 알립니다. BtnEnd 값이 True면 21~24번에서 구성한 조건문이 참이 될 수 없으므로, 이제 버튼을 떼기 전까지는 다시 Blue 값이 바뀌지 않습니다.

Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
42번의 Constant 출력 핀을 BtnEnd의 입력 핀에 연결합니다.

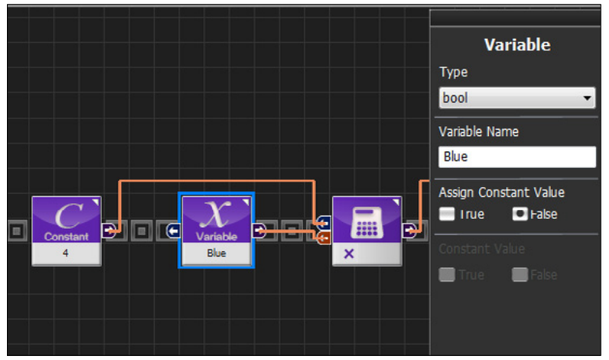


44 상수 4 추가

위에서 설명했듯이 LED 모듈에 입력 값이 들어가면, 그 값에 따라 LED가 켜집니다. Red, Green, Blue에는 True(1), False(0)로 값이 저장되어 있으므로, (4 x Blue + 2 x Green + 1 x Red) 라는 식을 구성해 LED 모듈에 입력해야 합니다.

상수 4를 추가합니다.

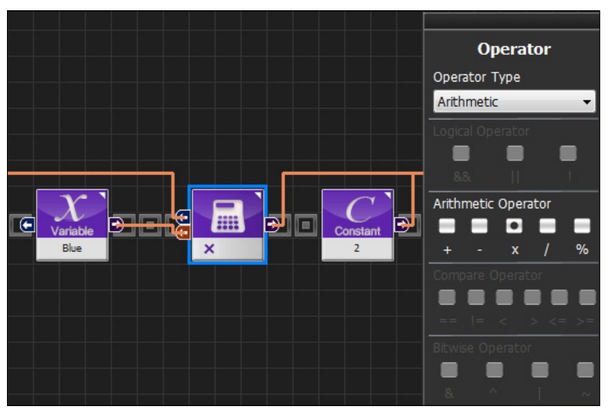
Data 〉 Constant 모듈을 선택해 If-Else 모듈 뒤에 붙입니다.
Constant Type : int로 설정합니다.
Constant Value : 4로 설정합니다.



45 Blue 값 읽어 오기

(4 x Blue + 2 x Green + 1 x Red) 식을 구성하기 위해, Blue 변수를 추가합니다.

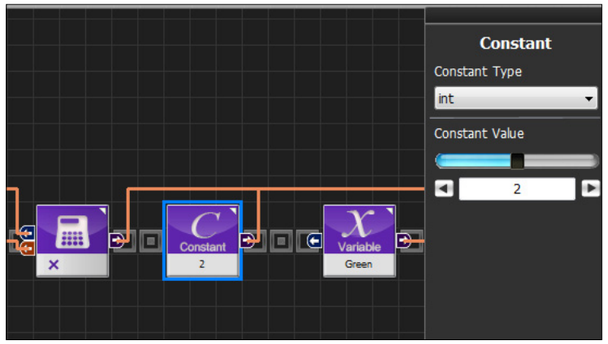
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Blue를 입력합니다.



46 곱하기 연산

곱하기 연산을 적용해 4와 Blue의 값을 곱합니다.

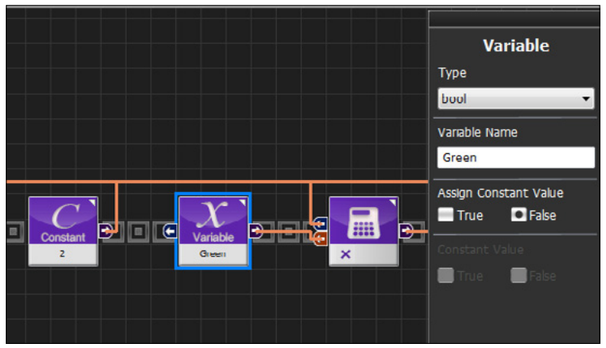
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Arithmetic으로 선택합니다.
Logical Operator : X(곱하기)로 선택합니다.
44번의 4와 45번의 Blue 모듈의 출력을 각각 곱하기 모듈의 입력에 연결합니다.



47 상수 2 추가

(4 x Blue + 2 x Green + 1 x Red) 식을 구성하기 위해 상수 2를 추가합니다.

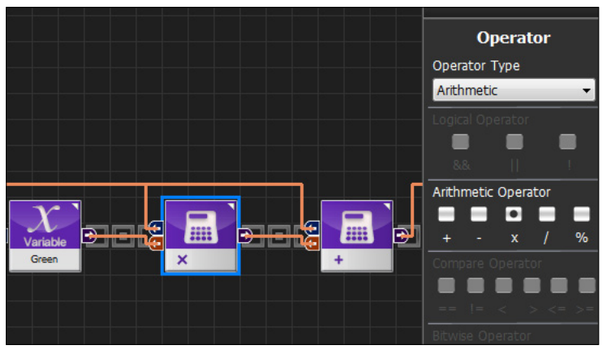
Data 〉 Constant 모듈을 선택해 이전 모듈의 뒤에 배치합니다.
Constant Type : int로 설정합니다.
Constant Value : 2로 설정합니다.



48 Green 값 읽어 오기

(4 x Blue + 2 x Green + 1 x Red) 식을 구성하기 위해, Green 변수를 추가합니다.

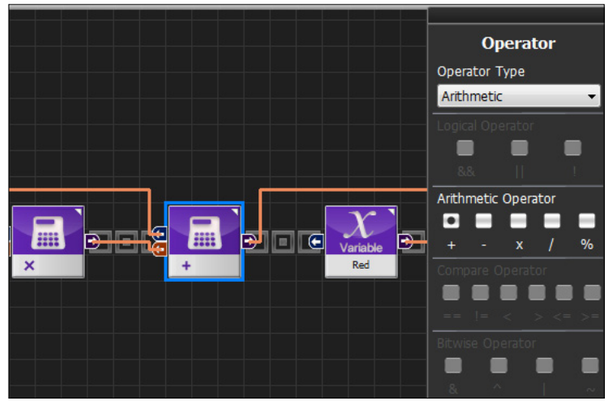
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Green를 입력합니다.



49 곱하기 연산

곱하기 연산을 적용해 2와 Green의 값을 곱합니다.

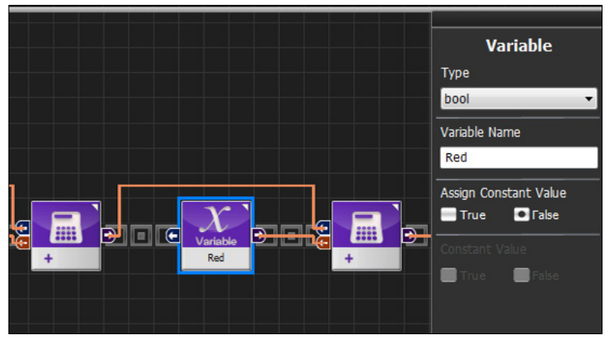
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Arithmetic으로 선택합니다.
Logical Operator : x(곱하기)로 선택합니다.
47번의 2와 48번의 Green 모듈의 출력을 각각 곱하기 모듈의 입력에 연결합니다.



50 더하기 연산

더하기 연산을 적용해 4 x Blue와 2 x Green을 더합니다.

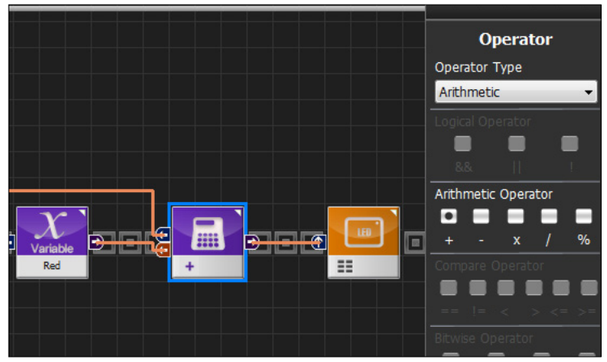
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Arithmetic으로 선택합니다.
Logical Operator : +(더하기)로 선택합니다.
46번과 49번의 곱하기 모듈 출력을 각각 더하기 모듈의 입력에 연결합니다.



51 Red 값 읽어 오기

(4 x Blue + 2 x Green + 1 x Red) 식의 마지막 항인 Red 변수를 추가합니다. 1은 추가할 필요가 없습니다.

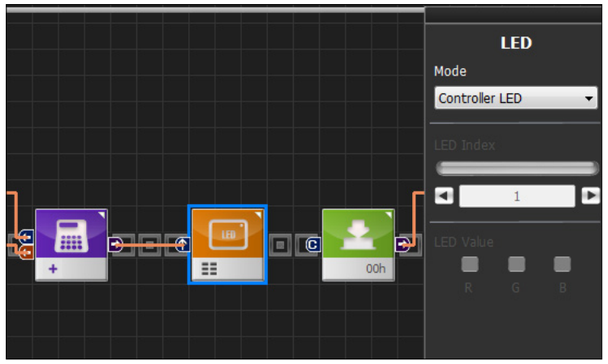
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : Red를 입력합니다.



52 더하기 연산

더하기 연산을 적용해 4 x Blue + 2 x Green과 Red를 더합니다.

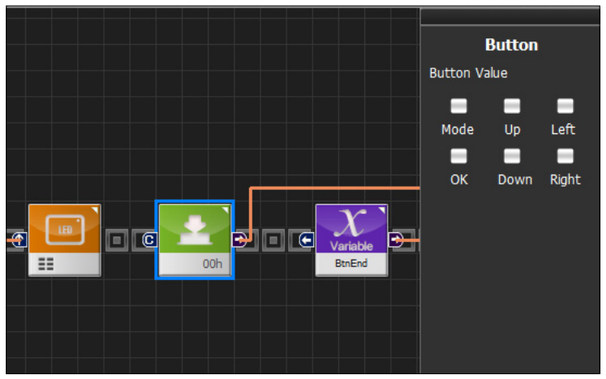
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Arithmetic으로 선택합니다.
Logical Operator : +(더하기)로 선택합니다.
50번의 더하기 모듈과 51번의 Red 출력을 각각 더하기 모듈의 입력에 연결합니다.



53 LED 모듈 추가

LED 모듈을 추가해서 앞에서 구성한 식의 값을 입력에 대입합니다. LED 모듈의 입력으로 (4 x Blue + 2 x Green + 1 x Red) 라는 값이 들어갑니다.

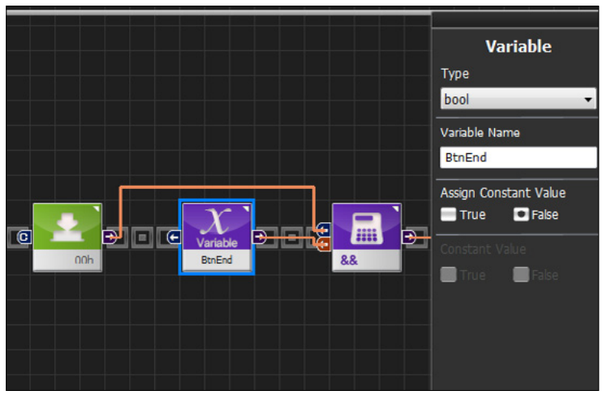
Motion 〉 LED를 선택해서 이전 모듈의 뒤에 배치합니다.
Mode : Controller LED를 선택합니다.
52번의 더하기 모듈 출력 핀을 LED 모듈의 입력 핀에 연결합니다.



54 버튼 뗌 감지

버튼이 눌렸을 때 변수의 값을 바꾸고 BtnEnd를 True로 만들어서 다시 값을 바꾸지 않도록 했으므로, 버튼을 떼면 BtnEnd를 False로 다시 바꿔야 합니다. 지금부터는 그 과정을 프로그래밍합니다.

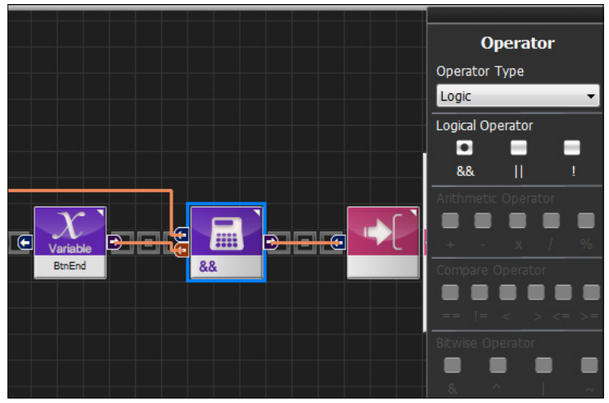
Motion 〉 Button 모듈을 선택합니다.
Button Value : 아무것도 선택하지 않습니다.
버튼이 안 눌린 상태를 의미합니다.



55 BtnEnd 추가

BtnEnd의 값을 읽어오기 위해 모듈을 추가합니다.

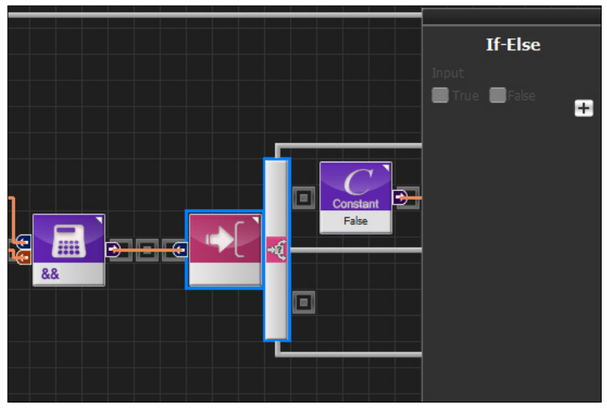
Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.



56 AND 연산

AND 연산을 적용해 버튼이 눌리지 않은 상태고, BtnEnd가 True일 때 출력 값이 True가 되도록 합니다.

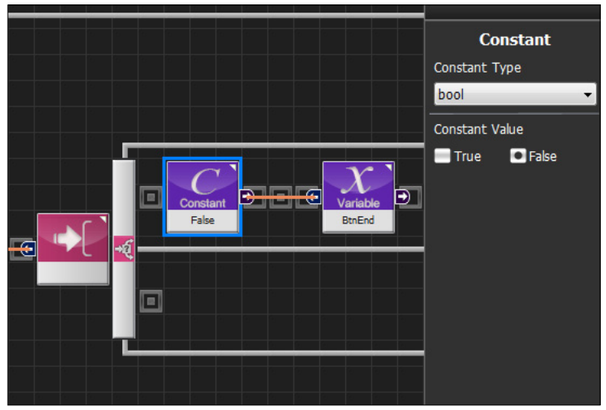
Data 〉 Operator 모듈을 선택합니다.
Operator Type : Logic으로 선택합니다.
Logical Operator : &&(Logical AND)로 선택합니다.
54번의 Button 모듈과 55번의 BtnEnd 모듈의 출력을 각각 AND 모듈의 입력에 연결합니다.



57 If-Else문 추가

앞의 AND 모듈에서 구성한 값이 참이면 BtnEnd를 False로 하기 위해서 If-Else문을 추가합니다.

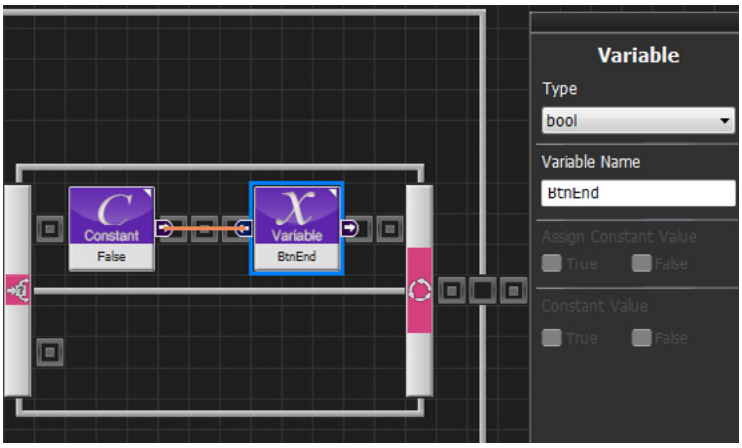
Flow 〉 If-Else를 선택해서 이전 모듈의 뒤에 배치합니다.
56번의 AND 모듈 출력을 If-Else 모듈 입력에 연결합니다.



58 상수 추가

BtnEnd를 False로 만들기 위해서, False 상수를 추가합니다.

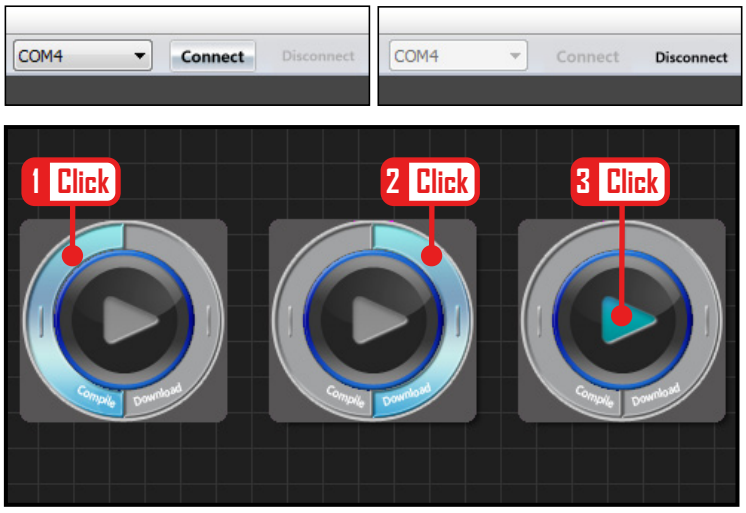
Data 〉 Constant 모듈을 선택해 If-Else 모듈의 가장 위 프로그램 라인에 배치합니다.
Constant Type : bool로 선택합니다.
Constant Value : False로 선택합니다.



59 BtnEnd 대입

BtnEnd 변수를 추가하고, False 값을 대입합니다.

Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.
58번의 False 출력 핀을 BtnEnd의 입력 핀에 연결합니다.

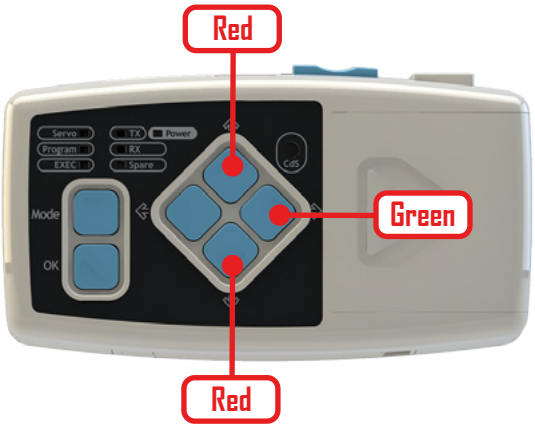


60 다운로드

프로그래밍 후 컴파일 –〉 로봇에 다운로드 –〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측
Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



61 로봇동작

BtnEnd의 값을 읽어오기 위해 모듈을 추가합니다.

Data 〉 Variable을 선택해서 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : BtnEnd를 입력합니다.

모듈별 프로그래밍

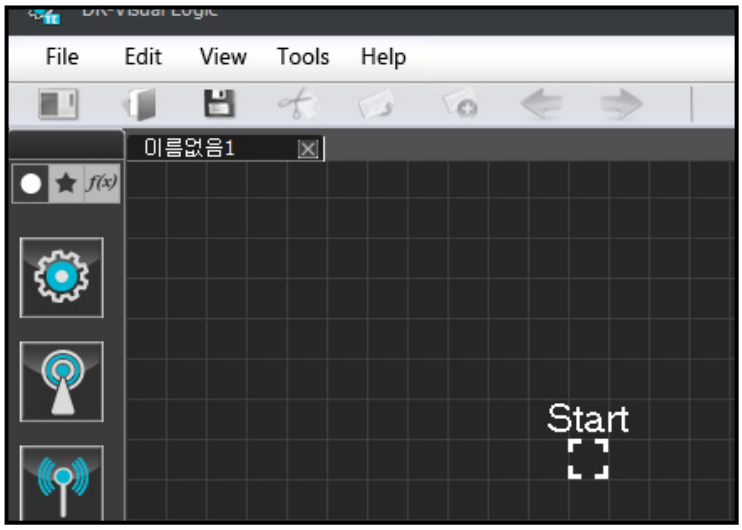
Light 센서

08-4

예제설명

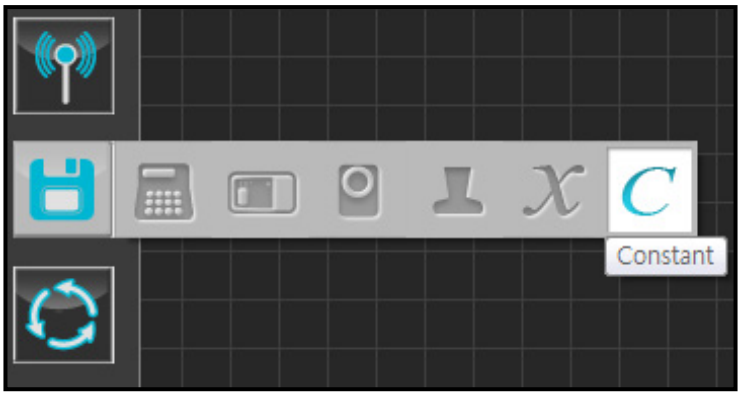
빛의 세기에 따라 로봇의 모터를 동작하는 예제입니다.

외부의 빛이 어두워지면 로봇이 오른팔을 올립니다. (제어기 뒤쪽의 CDS 센서를 손바닥으로 가리면 빛의 들어오는 양이 없어 어두워지고, 로봇이 이것을 감지해 오른팔을 올리도록 프로그래밍합니다.)



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.



02 모듈 선택

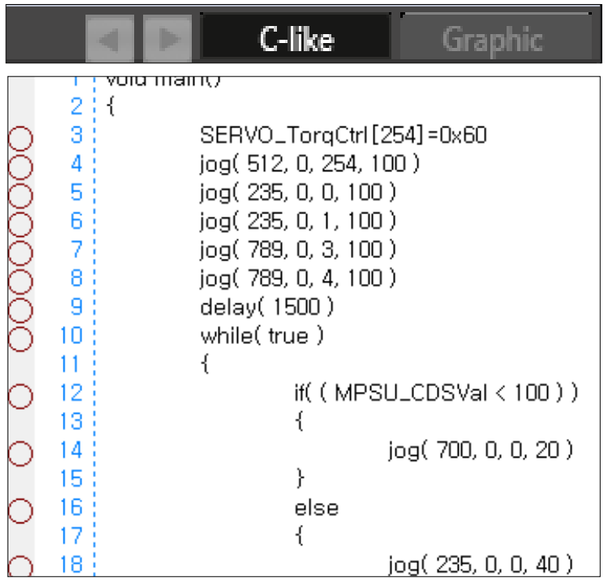
모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.

04 전체 프로그램

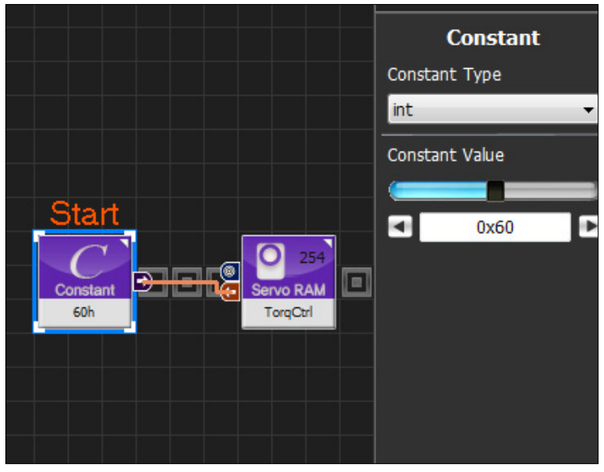
빛 센서를 이용하여 로봇의 모터를 제어하는 프로그램의 전체 모습입니다.



05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

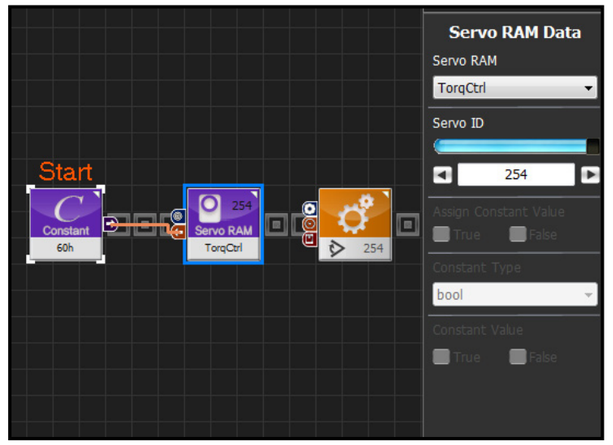
빛 센서를 이용한 모터 제어 프로그램 소스입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

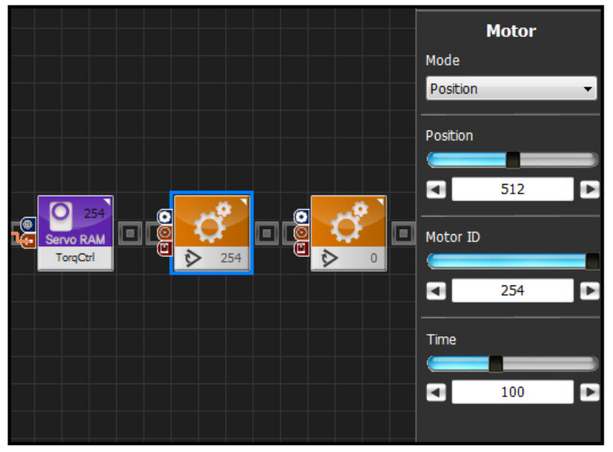
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다.
이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

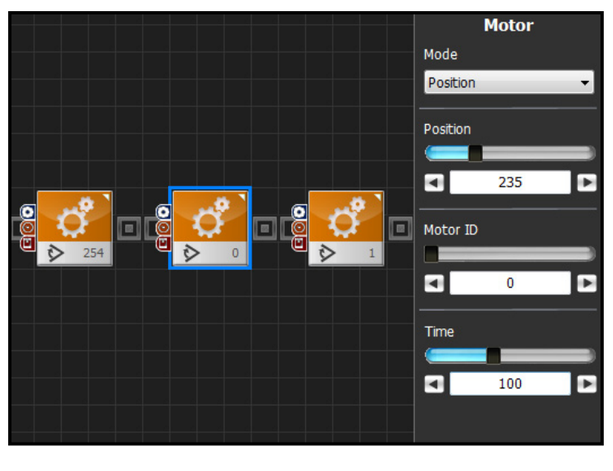
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



08 모든 서보 모터 위치 제어

모든 서보 모터의 위치를 중앙에 보내는 과정입니다.

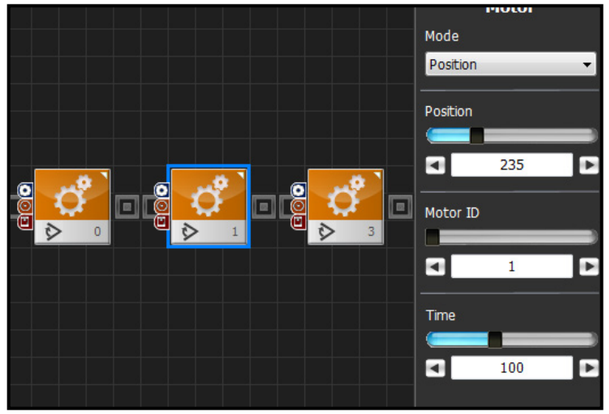
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 512로 설정합니다. 512는 모든 모터의 영점 위치로, 모터를 모두 중앙으로 보낸다는 의미입니다.
Motor ID : 254로 설정합니다. 254는 모든 모터에 적용하겠다는 의미입니다.
Time : 100으로 설정합니다. 단위는 1당 11.2ms로, 100은 약 1.12초를 의미합니다. 1.12초 동안 원하는 위치로 보낸다는 의미입니다.



09 모터 0번 (오른쪽 어깨) 설정

모든 로봇의 모터의 각도를 중앙으로 정렬하면 휴머노이드에서는 팔을 좌우로 뻗게 됩니다. 이것을 차려 자세로 되돌려 놓아야만 기본 자세를 유지하여 동작시키기가 용이해집니다.

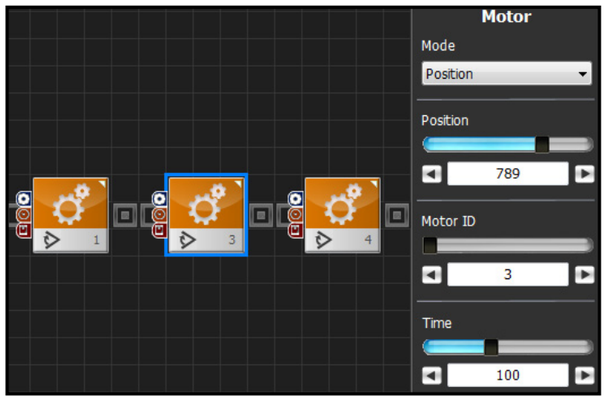
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 수평으로 들고 있던 오른팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



10 모터 1번 (오른쪽 위팔) 설정

오른쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

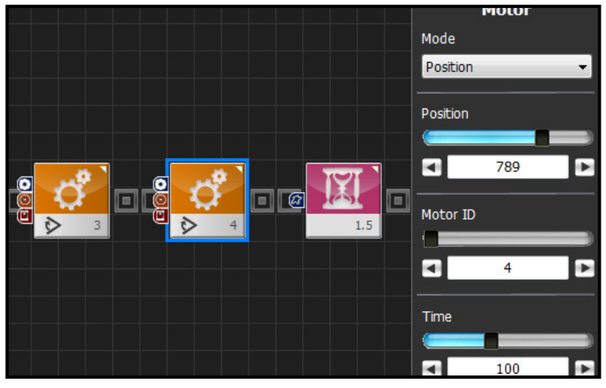
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 90도로 들고 있던 오른팔을 수직으로 내리는 위치 값입니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



11 모터 3번(왼쪽 어깨) 설정

왼쪽 어깨 모터를 돌려 왼팔을 수직으로 내려 갈 수 있는 위치로 바꿉니다.

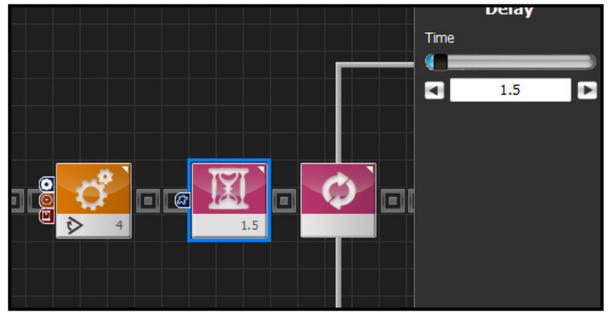
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 수평으로 들고 있던 왼팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



12 모터 4번 (왼쪽 위팔) 설정

왼쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

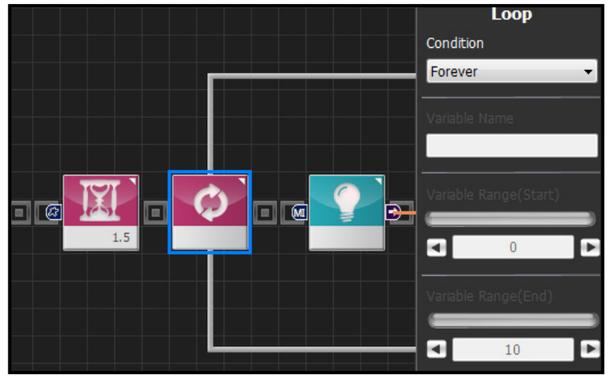
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 90도로 들고 있던 왼팔을 수직으로 내리는 위치 값입니다.
Motor ID : 4로 설정합니다. 왼쪽 위팔 모터 ID가 4번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



13 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 설정입니다. 직전 모듈들에 의해 모터가 움직이는 동안 아무 일도 하지 않고 기다립니다. 모터를 전체(254)를 512로 보내는 명령 후 0, 1, 3, 4를 따로 제어하는 명령을 바로 연달아 보냈으므로, 사용자가 보기에는 마치 동시에 차려 자세로 보낸 것처럼 움직입니다.

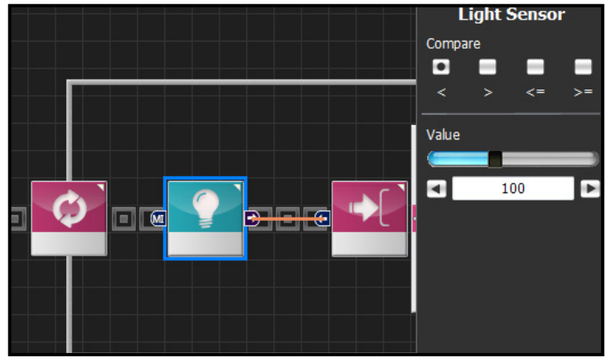
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



14 Loop 반복문

반복문을 넣어서 프로그램이 무한 반복하도록 합니다.

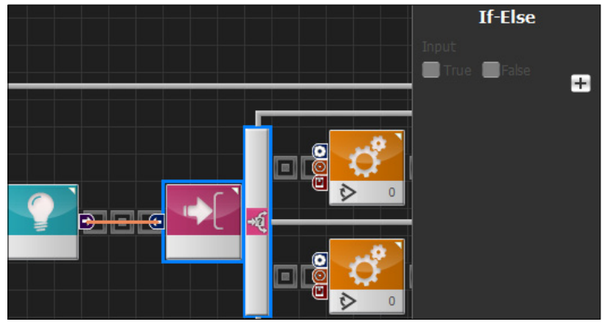
Flow 〉 Loop 모듈을 선택합니다.
Condition : Forever를 선택해 조건 없이 무한 반복하는 반복문을 만듭니다.



15 Light Sensor 추가

빛 센서 모듈을 추가합니다. 이 모듈은 현재 입력된 빛의 세기가 Compare의 비교 연산자와 Value의 값에 따라서 비교가 되어 True/False를 출력합니다.

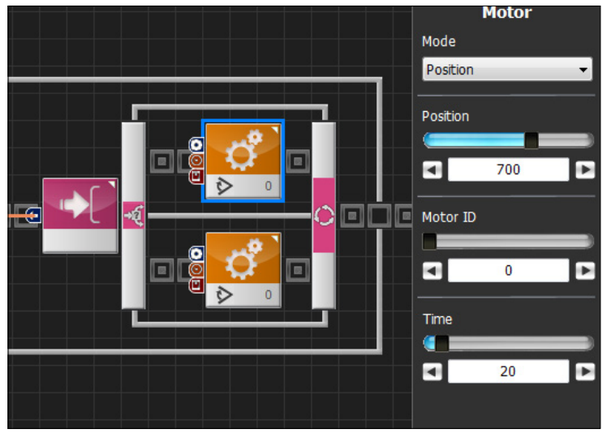
Sensor 〉 Light 모듈을 선택해 Loop 모듈의 안쪽에 배치합니다.
Compare : 〈를 선택합니다. 센서 값이 Value 값보다 작을 때 True를 출력합니다.
Value : 100으로 설정합니다. 기준 값이 100 임을 의미합니다.
빛 센서 값이 100보다 작으면 True, 크거나 같으면 False를 출력합니다.



16 If-Else 문

빛 센서 모듈의 출력에 따라 다른 동작을 하기 위해서 조건문을 만듭니다.

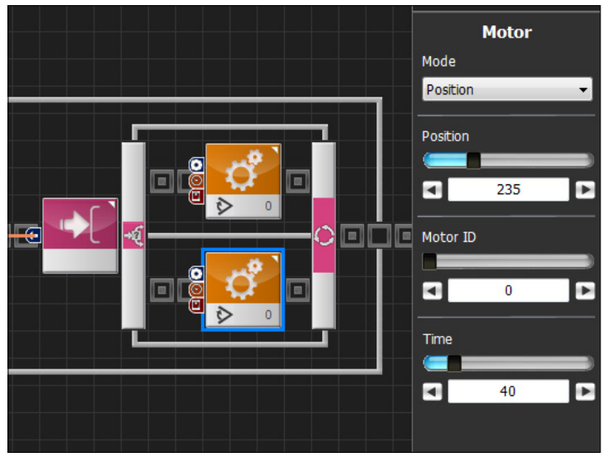
Flow 〉 If-Else를 선택해 직전 모듈의 뒤에 배치합니다.
15번의 빛 센서 출력을 If-Else문 입력에 연결합니다.



17 모터 0번(오른쪽 어깨) 설정

빛 센서 값이 100보다 작으면(True), 즉 어두우면 오른쪽 팔을 올립니다.

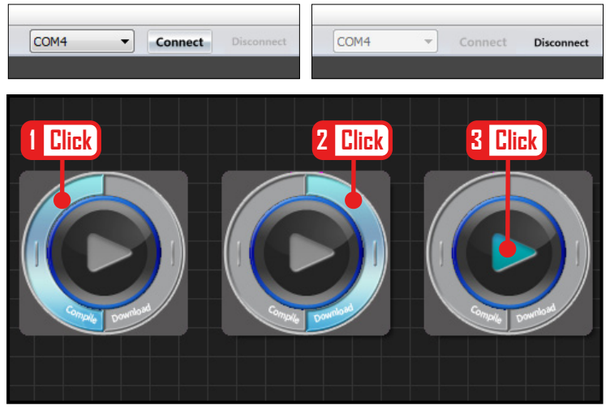
Motion 〉 Motor를 선택해 If-Else 문의 위 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 700으로 설정합니다. 700은 차려 자세로 놓여있는 팔을 앞으로 올리게 하는 위치 값 입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 20으로 설정합니다.



18 모터 0번(오른쪽 어깨) 설정

100보다 크거나 같을 경우(False), 즉 밝을 경우 오른쪽 팔을 내립니다.

Motion 〉 Motor를 선택해 If-Else 문의 아래 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 235로 설정합니다. 235는 오른쪽 팔을 차려 자세로 만드는 위치 값 입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 40으로 설정합니다. 올라간 속도보다 조금 더 느리게 내려옵니다.

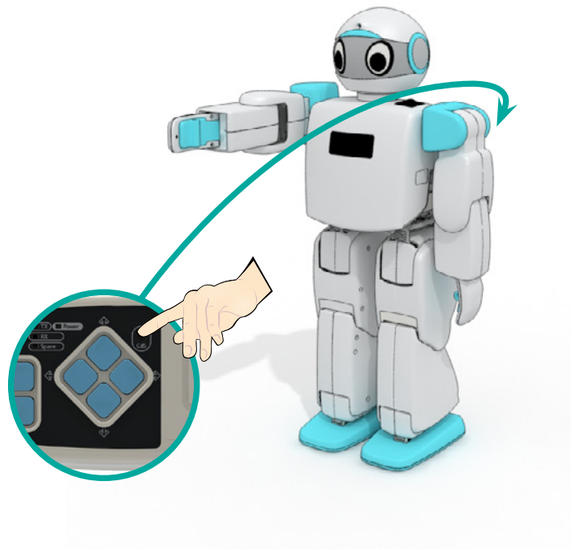


19 다운로드

프로그래밍 후 컴파일 –〉 로봇에 다운로드 –〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표) 을 클릭합니다.



20 로봇동작

현재는 빛이 밝은 상태이므로 차려자세를 유지합니다.
손가락으로 cds 를 가리면, 로봇이 오른팔을 듭니다.
손가락을 떼면 로봇이 오른팔을 내립니다.
정확히 실행되는 것을 확인합니다.

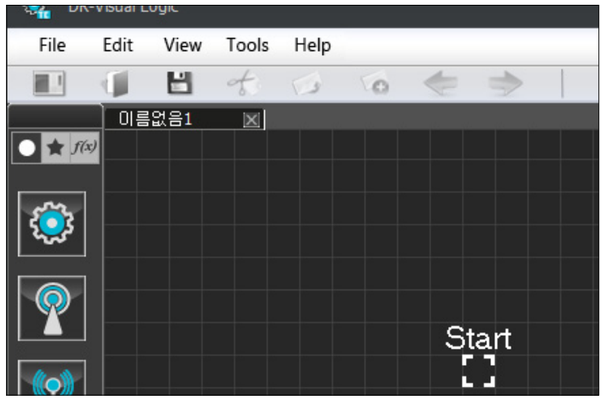
08-5 모듈별 프로그래밍 Sound 센서

예제설명

Sound Sensor는 제어기 DRC 내부의 양쪽에 위치합니다. Sound Sensor 모듈은 박수 소리 등의 큰 소리가 입력되면 마지막으로 들어온 소리의 방향을 Compare, Value의 값과 비교해 True나 False로 출력합니다.

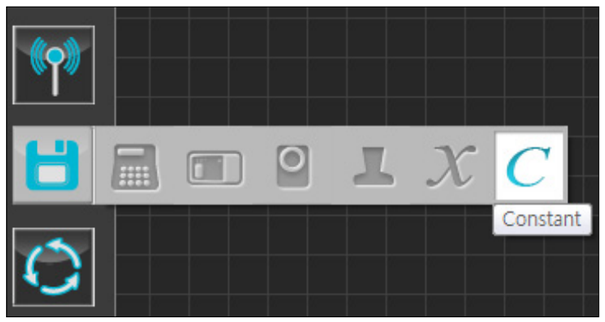
소리의 값은 –2~2의 범위를 가집니다. (–) 값은 소리가 왼쪽에서 났음을, (+) 값은 소리가 오른쪽에서 났음을 나타내며 0 값은 소리가 가운데에서 났다는 뜻입니다. 소리를 입력할 때는 벽이나 바닥에 반사된 소리 때문에 방향인식이 잘 되지 않는 경우가 있으므로, 가능하면 소리 센서의 가까이에서 박수나 핑거 스냅 등으로 소리를 내는 편이 좋습니다.

왼쪽 측면에서 박수를 치면 왼쪽 손을 들고, 오른쪽 측면에서 박수를 치면 오른쪽 손을 드는 프로그램을 작성해 봅니다.



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.

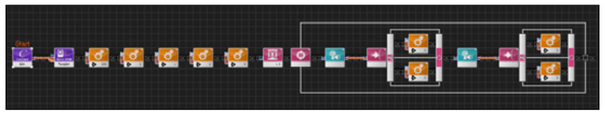


02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그래밍

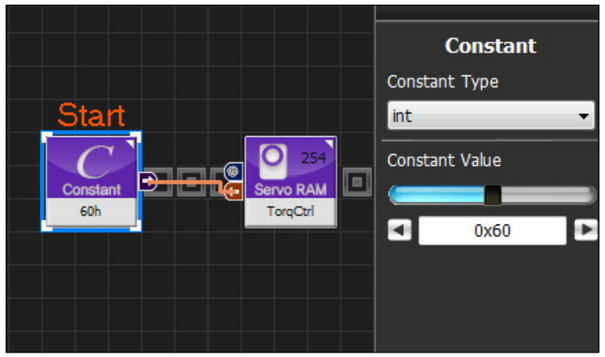
소리 센서를 이용하여 로봇의 모터를 제어하는 프로그램의 전체 모습입니다.

```
1  void main()
2  {
3          SERVO_TorqCtrl[254]=0x60
4          jog( 512, 0, 254, 100 )
5          jog( 235, 0, 0, 100 )
6          jog( 235, 0, 1, 100 )
7          jog( 789, 0, 3, 100 )
8          jog( 789, 0, 4, 100 )
9          delay( 1500 )
10         while( true )
11         {
12                 if( ( MPSU_SoundDetectFlag && MPSU_SoundDir > 1 ) )
13                 {
14                         jog( 700, 0, 0, 20 )
15                 }
16                 else
17                 {
18                         jog( 235, 0, 0, 40 )
19                 }
20                 if( ( MPSU_SoundDetectFlag && MPSU_SoundDir < -1 ) )
21                 {
22                         jog( 324, 0, 3, 20 )
23                 }
24                 else
25                 {
26                         jog( 789, 0, 3, 40 )
27                 }
28         }
29 }
```

05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

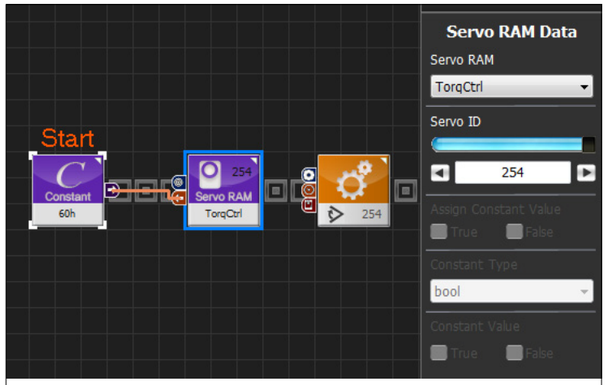
소리 센서를 이용한 모터 제어 프로그램 소스입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.

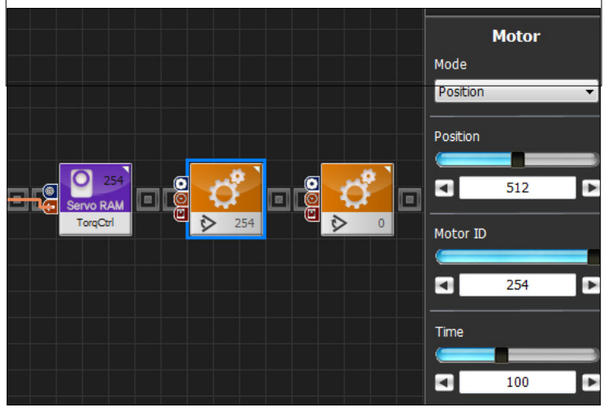


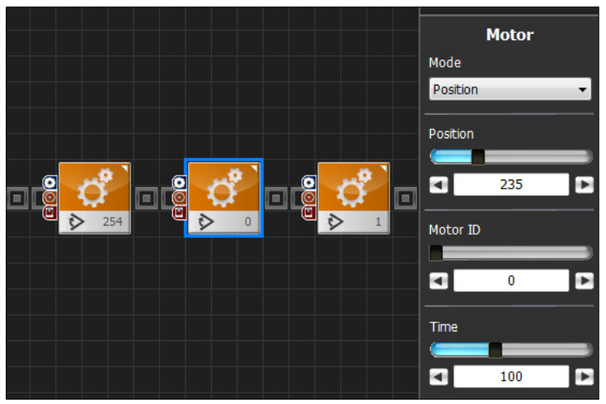
06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다. 이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.







07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.

08 모든 서보 모터 위치 제어

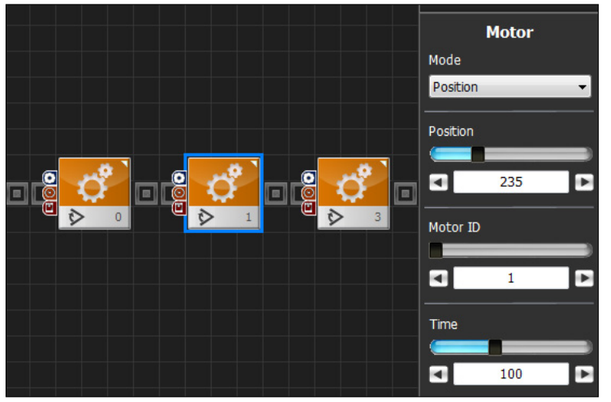
모든 서보 모터의 위치를 중앙에 보내는 과정입니다.

Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 512로 설정합니다. 512는 모든 모터의 영점 위치로, 모터를 모두 중앙으로 보낸다는 의미입니다.
Motor ID : 254로 설정합니다. 254는 모든 모터에 적용하겠다는 의미입니다.
Time : 100으로 설정합니다. 단위는 1당 11.2ms로, 100은 약 1.12초를 의미합니다. 1.12초 동안 원하는 위치로 보낸다는 의미입니다.

09 모터 0번 (오른쪽 어깨) 설정

모든 로봇의 모터의 각도를 중앙으로 정렬하면 휴머노이드에서는 팔을 좌우로 뻗게 됩니다. 이것을 차려 자세로 되돌려 놓아야만 기본 자세를 유지하여 동작시키기가 용이해집니다.

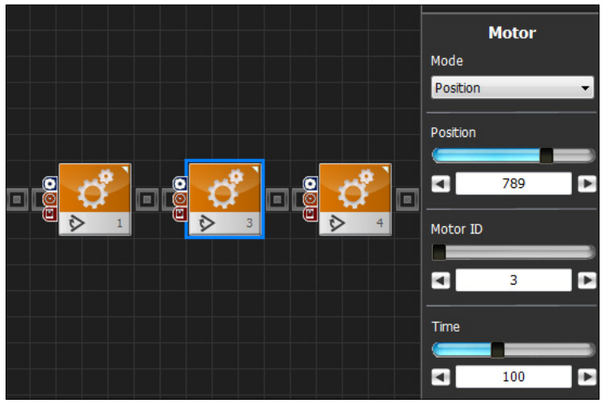
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 수평으로 들고 있던 오른팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



10 모터 1번 (오른쪽 위팔)) 설정

오른쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

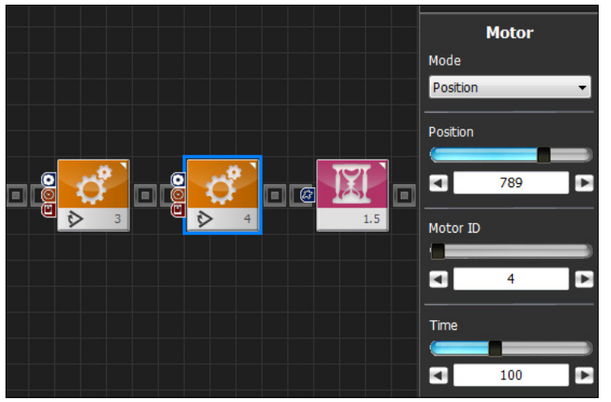
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 90도로 들고 있던 오른팔을 수직으로 내리는 위치 값입니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



11 모터 3번(왼쪽 어깨) 설정

왼쪽 어깨 모터를 돌려 왼팔을 수직으로 내려 갈 수 있는 위치로 바꿉니다.

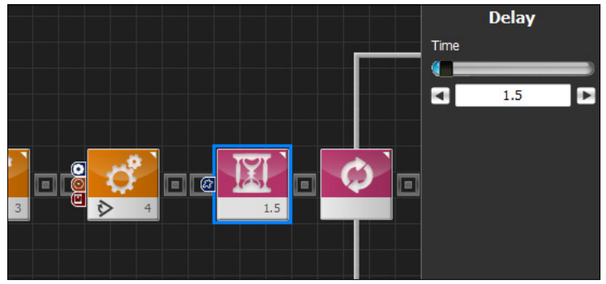
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 수평으로 들고 있던 왼팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



12 모터 4번(왼쪽 위 팔) 설정

왼쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

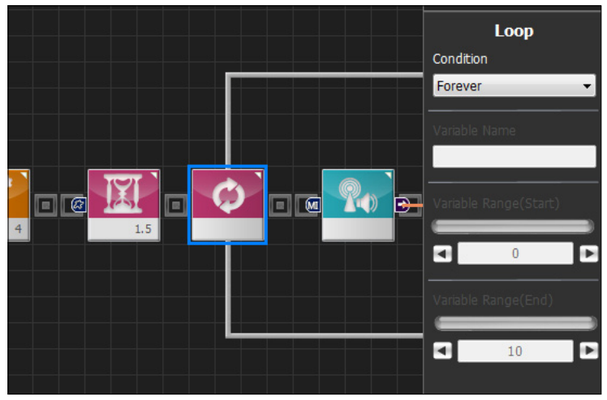
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 90도로 들고 있던 왼팔을 수직으로 내리는 위치 값입니다.
Motor ID : 4로 설정합니다. 왼쪽 위팔 모터 ID가 4번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



13 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 설정입니다. 직전 모듈들에 의해 모터가 움직이는 동안 아무 일도 하지 않고 기다립니다. 모터를 전체(254)를 512로 보내는 명령 후 0, 1, 3, 4를 따로 제어하는 명령을 바로 연달아 보냈으므로, 사용자가 보기에는 마치 동시에 차려 자세로 보낸 것처럼 움직입니다.

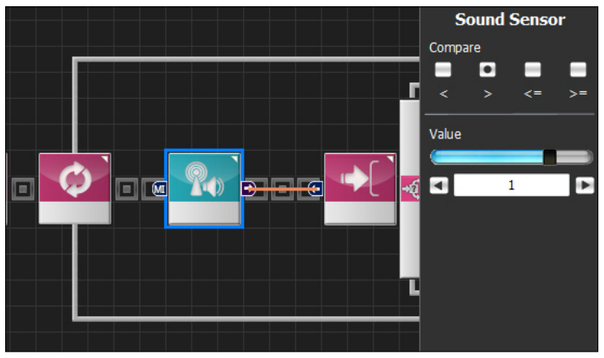
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



14 Loop 반복문

반복문을 넣어서 프로그램이 무한 반복하도록 합니다.

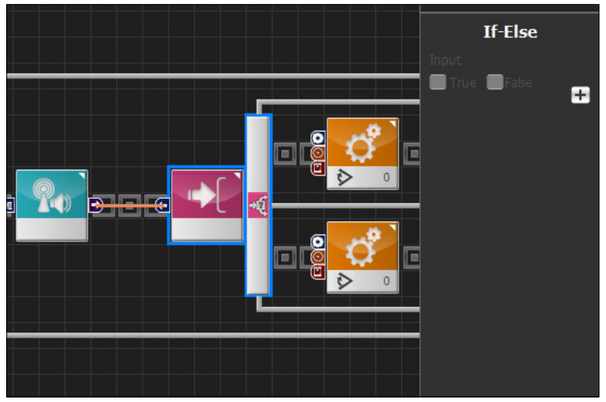
Flow 〉 Loop 모듈을 선택합니다.
Condition : Forever를 선택해 조건 없이 무한 반복하는 반복문을 만듭니다.



15 Sound Sensor 추가

소리 센서 모듈을 추가합니다. 이 모듈은 현재 소리가 입력되었다면, 소리의 방향을 Compare의 비교 연산자와 Value의 값에 따라서 비교하여 True/False를 출력합니다. 소리가 입력되지 않았다면 무조건 False를 출력합니다.

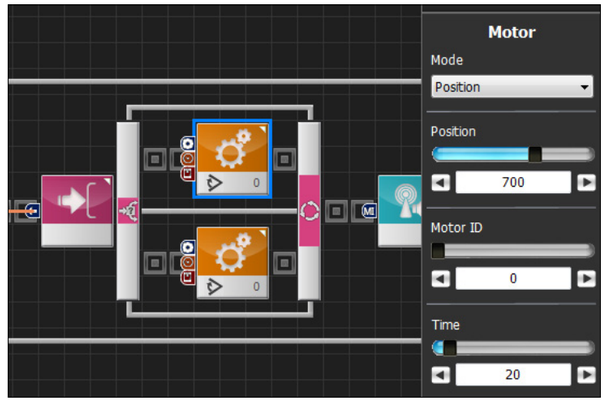
Sensor 〉 Sound 모듈을 선택해 Loop 모듈의 안쪽에 배치합니다.
Compare : 〉를 선택합니다. 센서 값이 Value 값보다 클 때 True를 출력합니다.
Value : 1로 설정합니다. 오른쪽에서 소리가 입력되었음을 의미합니다.
소리가 입력 되었고, 소리 센서 값이 1보다 크면 True, 작거나 같으면 False를 출력합니다. 즉, 오른쪽에서 소리가 들어온 경우 True가 됩니다.



16 Switch IF 문

소리 센서 모듈의 출력에 따라 다른 동작을 하기 위해서 조건문을 만듭니다.

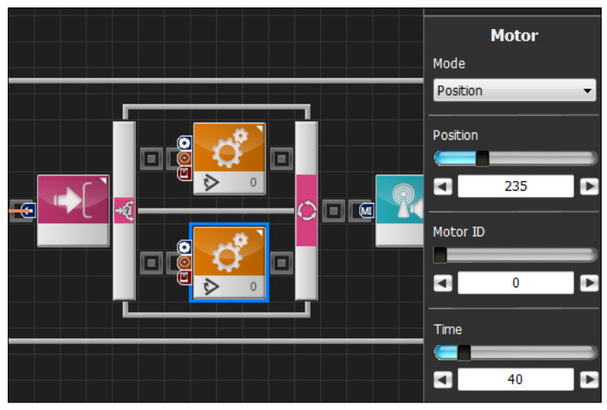
Flow 〉 If-Else를 선택해 직전 모듈의 뒤에 배치합니다.
15번의 소리 센서 출력을 If-Else문 입력에 연결합니다.



17 모터 0번(오른쪽 어깨) 설정

소리 입력이 들어왔으며 방향 값이 1보다 크면(True), 즉 오른쪽에서 소리가 들어왔을 경우 오른쪽 팔을 올립니다.

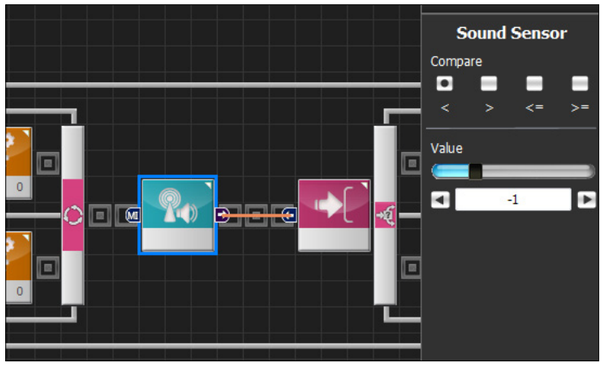
Motion 〉 Motor를 선택해 If-Else 문의 위 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 700으로 설정합니다. 700은 차려 자세로 놓여있는 팔을 앞으로 올리게 하는 위치 값 입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 20으로 설정합니다.



18 모터 0번(오른쪽 어깨) 설정

소리 입력이 들어오지 않았거나 방향 값이 1보다 작거나 같으면(False), 즉 오른쪽에서 소리가 들어오지 않았을 경우 오른쪽 팔을 내립니다.

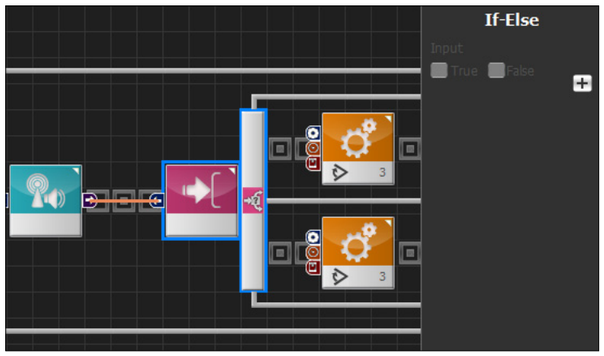
Motion 〉 Motor를 선택해 If-Else 문의 아래 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 235로 설정합니다. 235는 오른쪽 팔을 차려 자세로 만드는 위치 값 입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 40으로 설정합니다. 올라간 속도보다 조금 더 느리게 내려옵니다.



19 Sound Sensor 추가

소리 센서 모듈을 추가합니다. 왼쪽에 대해서 조건문을 만들어주기 위함입니다.

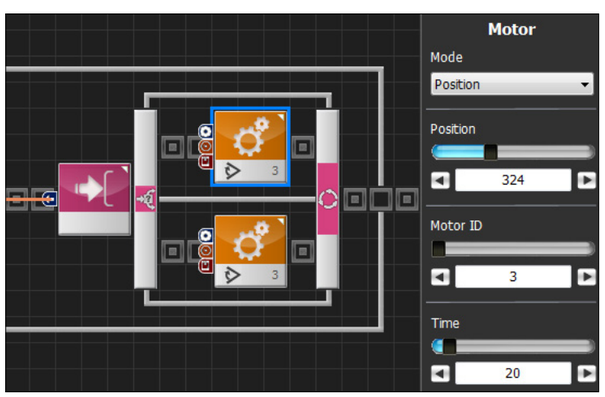
Sensor 〉 Sound 모듈을 선택해 If-Else 모듈의 뒤에 배치합니다.
Compare : 〈를 선택합니다. 센서 값이 Value 값보다 작을 때 True를 출력합니다.
Value : -1로 설정합니다. 왼쪽에서 소리가 입력되었음을 의미합니다.
소리가 입력 되었고, 소리 센서 값이 -1보다 작으면 True, 크거나 같으면 False를 출력합니다. 즉, 왼쪽에서 소리가 들어온 경우 True가 됩니다.



20 If-Else 문

소리 센서 모듈의 출력에 따라 다른 동작을 하기 위해서 조건문을 만듭니다.

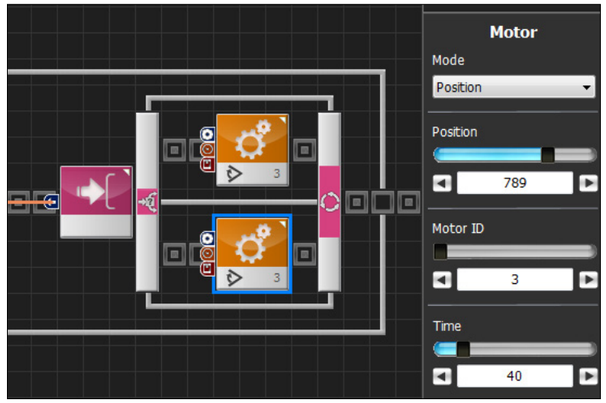
Flow 〉 If-Else를 선택해 직전 모듈의 뒤에 배치합니다.
19번의 소리 센서 출력을 If-Else문 입력에 연결합니다.



21 모터 3번(왼쪽 어깨) 설정

소리 입력이 들어왔으며 방향 값이 -1보다 작으면(True), 즉 왼쪽에서 소리가 들어왔을 경우 왼쪽 팔을 올립니다.

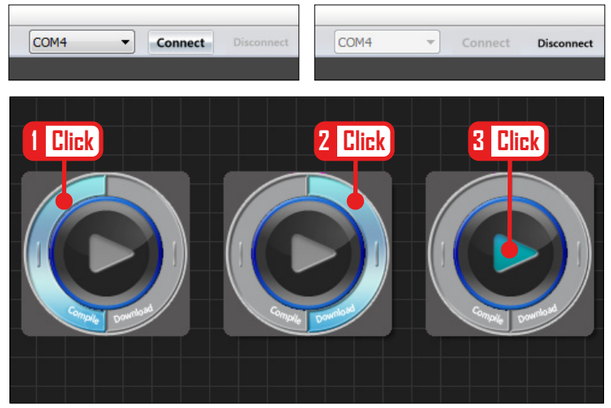
Motion 〉 Motor를 선택해 If-Else 문의 위 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 324으로 설정합니다. 324은 차려 자세로 놓여있는 팔을 앞으로 올리게 하는 위치 값 입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 20으로 설정합니다.



22 모터 3번(왼쪽 어깨) 설정

소리 입력이 들어오지 않았거나 방향 값이 -1보다 크거나 같으면(False), 즉 왼쪽에서 소리가 들어오지 않았을 경우 왼쪽 팔을 내립니다.

Motion 〉 Motor를 선택해 If-Else 문의 아래 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 왼쪽 팔을 차려 자세로 만드는 위치 값 입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 40으로 설정합니다. 올라간 속도보다 조금 더 느리게 내려옵니다.



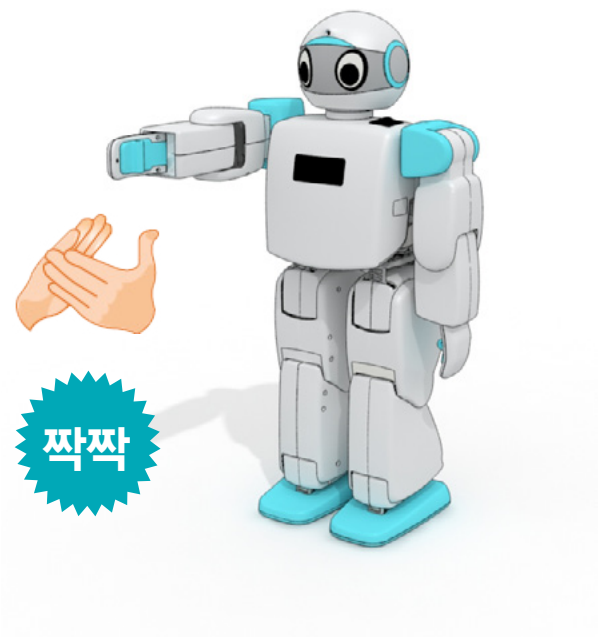
23 다운로드

프로그래밍 후 컴파일 -〉 로봇에 다운로드 -〉 실행하는 과정을 거칩니다.

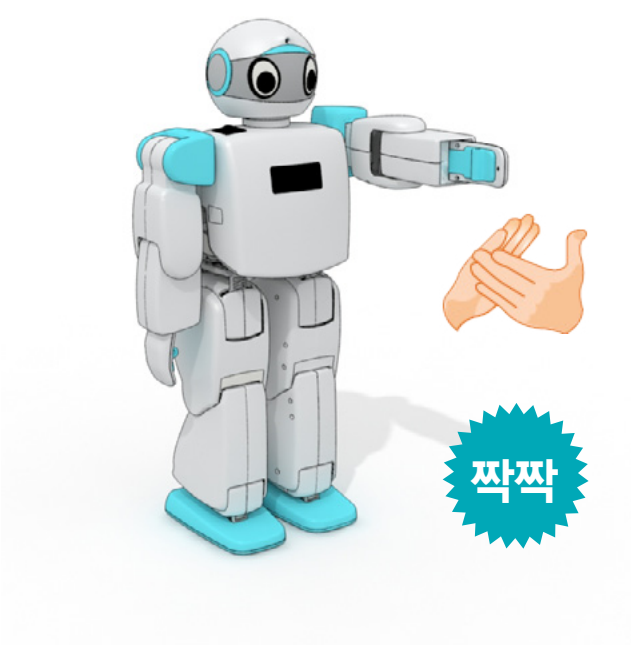
사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측
Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.

1



2



24 로봇동작

로봇 오른쪽에서 박수치면 오른쪽 팔을 올리고, 로봇 왼쪽에서 박수치면 왼쪽 팔을 올립니다.

모듈별 프로그래밍

Sound 센서(심화)

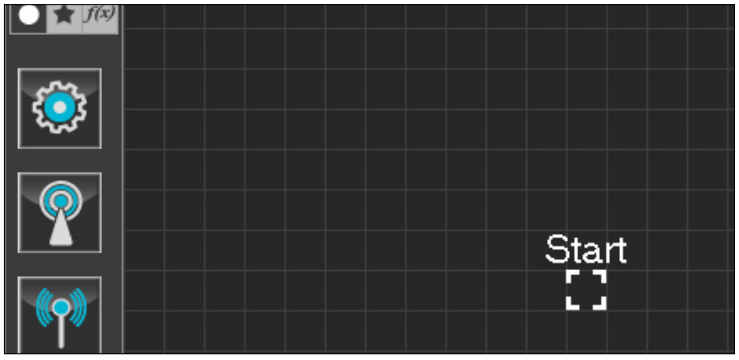
08-6

예제설명

바로 전 예제에서는 왼쪽 측면에서 박수를 치면 왼쪽 손을 들고, 오른쪽 측면에서 박수를 치면 오른쪽 손을 드는 프로그램을 작성했습니다. 하지만, 주변이나 모터의 구동에 따른 소음과 진동 때문에 DRC에 예상치 못한 소리 입력이 들어가는 경우가 있기 때문에 실제로는 의도한대로의 정확한 동작을 보기가 힘듭니다.

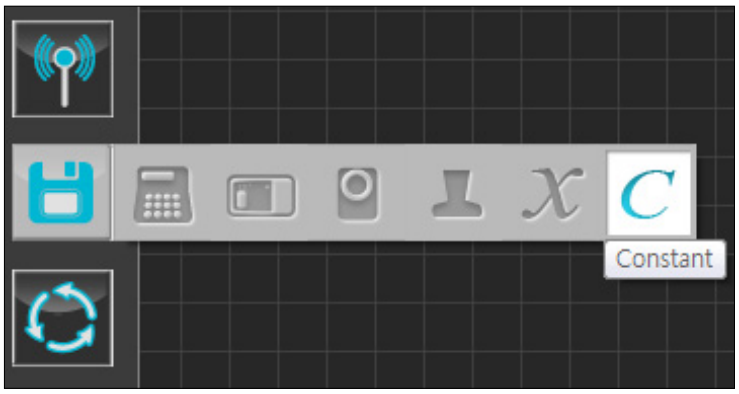
이를 해결하는 한 가지 방법은 무조건 소리 입력을 로봇이 가만히 서있는 경우에만 받는 방법입니다. 가만히 서있을 때는 모터 구동 소음이 적으므로, 주변의 소리를 비교적 정확히 인식할 수 있습니다. 입력에 따라서 구동할 때는 입력을 받지 않고, 구동이 끝난 후 1초 이상 기다려서 센서 정보가 초기화 된 후 다시 입력을 받는 방식입니다.

1초를 기다리는 것은, 소리 센서 정보가 마지막 소리가 들어오고 나서 1초 후 초기화 되기 때문입니다.



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.

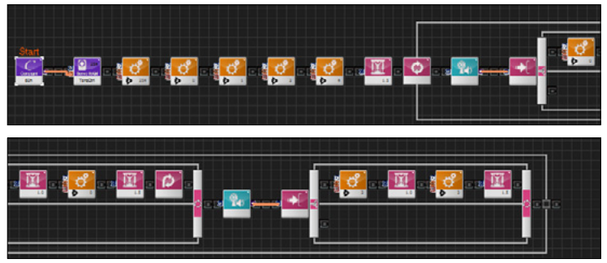


02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그래밍

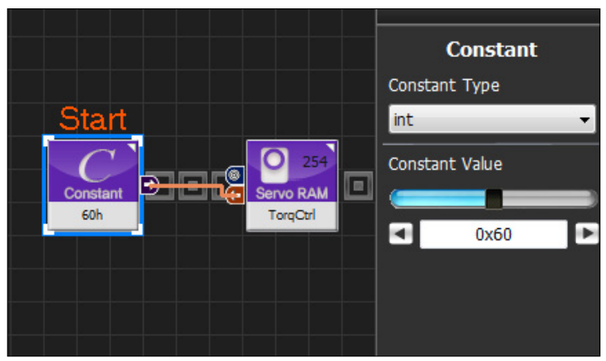
소리 센서를 이용하여 좀 더 정확하게 로봇의 모터를 제어하는 프로그램의 전체 모습입니다.


```
void main()
{
	SERVO_TorqCtrl[254]=0x60
	jog( 512, 0, 254, 100 )
	jog( 235, 0, 0, 100 )
	jog( 235, 0, 1, 100 )
	jog( 789, 0, 3, 100 )
	jog( 789, 0, 4, 100 )
	delay( 1500 )
	while( true )
	{
		if( ( MPSU_SoundDetectFlag && MPSU_SoundDir > 1 ) )
		{
			jog( 700, 0, 0, 20 )
			delay( 1000 )
			jog( 235, 0, 0, 40 )
			delay( 1500 )
			continue
		}
		else
		{
		}
		if( ( MPSU_SoundDetectFlag && MPSU_SoundDir < -1 ) )
		{
			jog( 324, 0, 3, 20 )
			delay( 1000 )
			jog( 789, 0, 3, 40 )
			delay( 1500 )
		}
		else
		{
		}
	}
}
```

05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

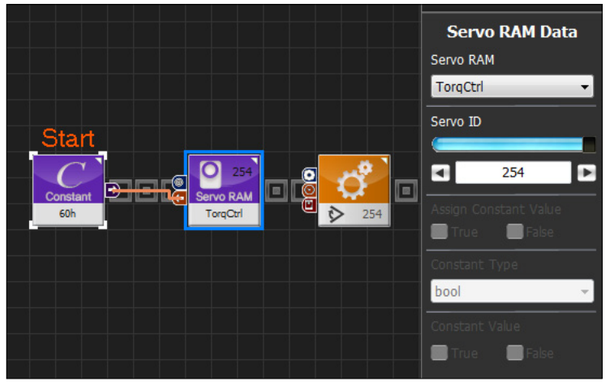
소리 센서를 이용한 좀 더 정확한 모터 제어 프로그램 소스입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

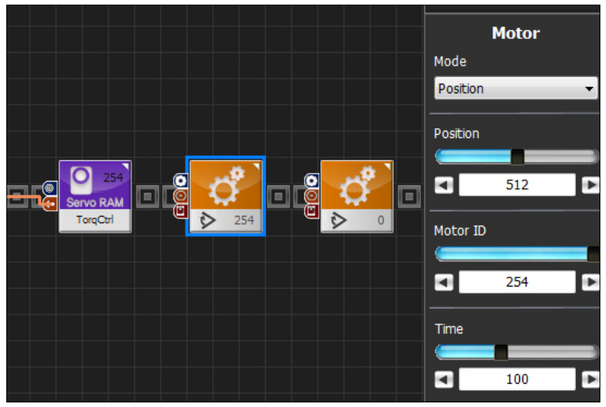
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다. 이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

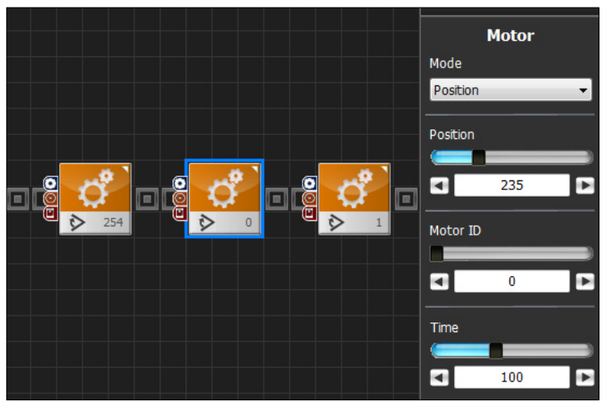
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다



08 모든 서보 모터 위치 제어

모든 서보 모터의 위치를 중앙에 보내는 과정입니다.

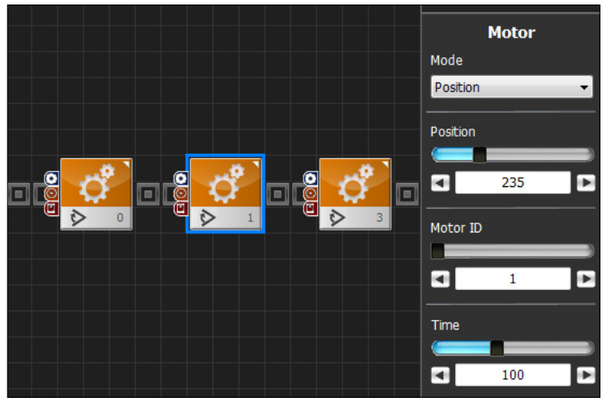
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 512로 설정합니다. 512는 모든 모터의 영점 위치로, 모터를 모두 중앙으로 보낸다는 의미입니다.
Motor ID : 254로 설정합니다. 254는 모든 모터에 적용하겠다는 의미입니다.
Time : 100으로 설정합니다. 단위는 1당 11.2ms로, 100은 약 1.12초를 의미합니다. 1.12초 동안 원하는 위치로 보낸다는 의미입니다.



09 모터 0번 (오른쪽 어깨) 설정

모든 로봇의 모터의 각도를 중앙으로 정렬하면 휴머노이드에서는 팔을 좌우로 뻗게 됩니다. 이것을 차려 자세로 되돌려 놓아야만 기본 자세를 유지하여 동작시키기가 용이해집니다.

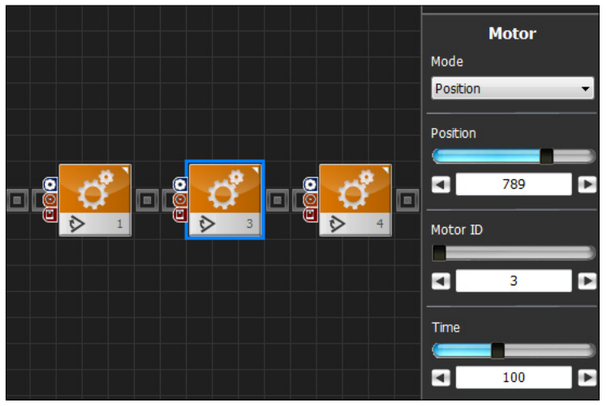
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 수평으로 들고 있던 오른팔을 수직으로 내려갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



10 모터 1번 (오른쪽 위팔) 설정

오른쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

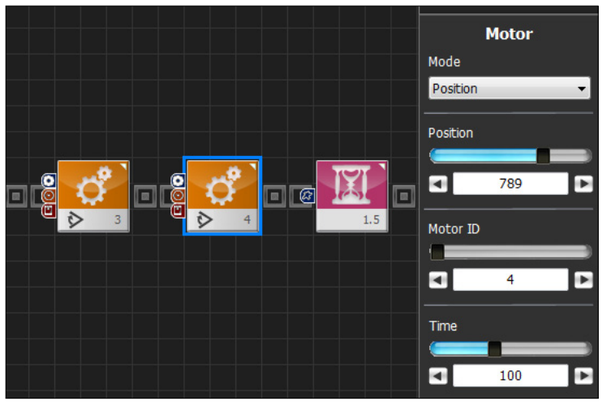
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다. 각도를 제어합니다.
Position : 235로 설정합니다. 235는 90도로 들고 있던 오른팔을 수직으로 내리는 위치 값입니다.
Motor ID : 1로 설정합니다. 오른쪽 위팔 모터 ID가 1번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



11 모터 3번(왼쪽 어깨) 설정

왼쪽 어깨 모터를 돌려 왼팔을 수직으로 내려 갈 수 있는 위치로 바꿉니다.

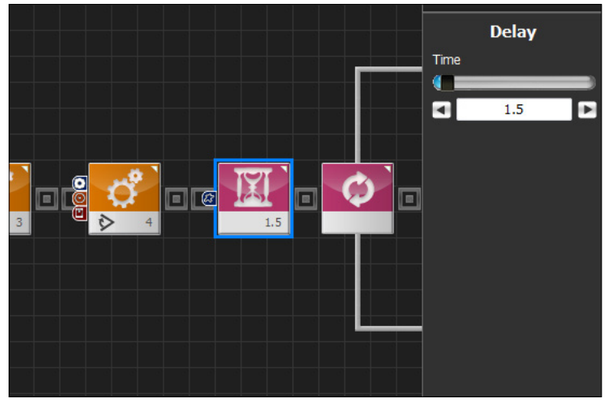
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 수평으로 들고 있던 왼팔을 수직으로 내려 갈 수 있게 모터를 돌리게 되는 위치 값입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



12 모터 4번(왼쪽 위팔) 설정

왼쪽 위팔 모터를 돌려 팔을 수직으로 내리는 과정입니다.

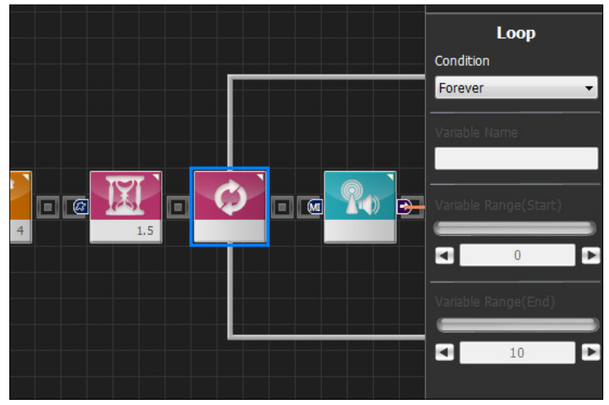
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 90도로 들고 있던 왼팔을 수직으로 내리는 위치 값입니다.
Motor ID : 4로 설정합니다. 왼쪽 위팔 모터 ID가 4번입니다.
Time : 100으로 설정합니다. 1.12초 동안 원하는 위치로 이동합니다.



13 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 설정입니다. 직전 모듈들에 의해 모터가 움직이는 동안 아무 일도 하지 않고 기다립니다. 모터를 전체(254)를 512로 보내는 명령 후 0, 1, 3, 4를 따로 제어하는 명령을 바로 연달아 보냈으므로, 사용자가 보기에는 마치 동시에 차려 자세로 보낸 것처럼 움직입니다.

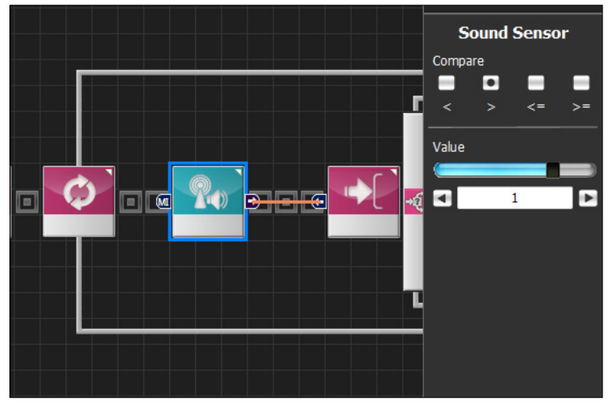
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



14 Loop 반복문

반복문을 넣어서 프로그램이 무한 반복하도록 합니다.

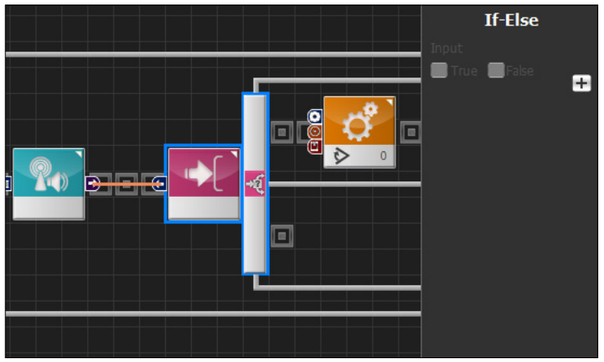
Flow 〉 Loop 모듈을 선택합니다.
Condition : Forever를 선택해 조건 없이 무한 반복하는 반복문을 만듭니다.



15 Sound Sensor 추가

소리 센서 모듈을 추가합니다. 이 모듈은 현재 소리가 입력되었다면, 소리의 방향을 Compare의 비교 연산자와 Value의 값에 따라서 비교하여 True/False를 출력합니다. 소리가 입력되지 않았다면 무조건 False를 출력합니다.

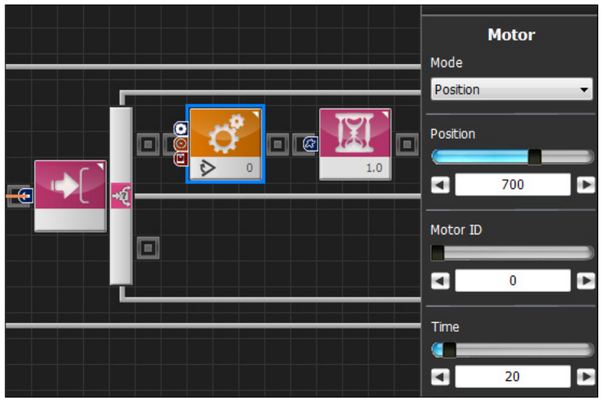
Sensor 〉 Sound 모듈을 선택해 Loop 모듈의 안쪽에 배치합니다.
Compare : 〉를 선택합니다. 센서 값이 Value 값보다 클 때 True를 출력합니다.
Value : 1로 설정합니다. 오른쪽에서 소리가 입력되었음을 의미합니다.
소리가 입력 되었고, 소리 센서 값이 1보다 크면 True, 작거나 같으면 False를 출력합니다. 즉, 오른쪽에서 소리가 들어온 경우 True가 됩니다.



16 Switch IF 문

소리 센서 모듈의 출력이 True면 오른팔 동작을 하기 위해서 조건문을 만듭니다.

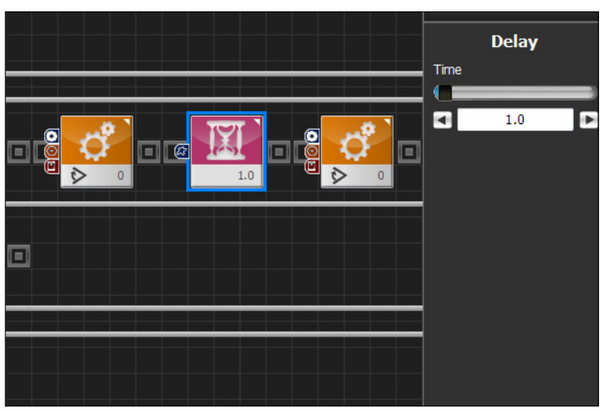
Flow 〉 If-Else를 선택해 직전 모듈의 뒤에 배치합니다.
15번의 소리 센서 출력을 If-Else문 입력에 연결합니다.



17 모터0번(오른쪽 어깨) 올리기

소리 입력이 들어왔으며 방향 값이 1보다 크면(True), 즉 오른쪽에서 소리가 들어왔을 경우 오른쪽 팔을 올립니다.

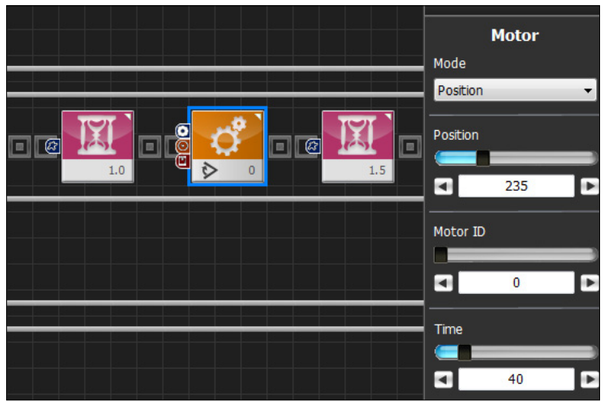
Motion 〉 Motor를 선택해 If-Else 문의 위 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 700으로 설정합니다. 700은 차려 자세로 놓여있는 팔을 앞으로 올리게 하는 위치 값 입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 20으로 설정합니다. 약 0.224초 동안 원하는 위치로 이동합니다.



18 동작 대기

들어 올리는 명령을 내리고 실제로 팔을 들어올릴 때까지 시간이 필요하므로 기다립니다. 딜레이를 1초를 주어 다 들어올린 후 잠시 멈춰 있도록 합니다.

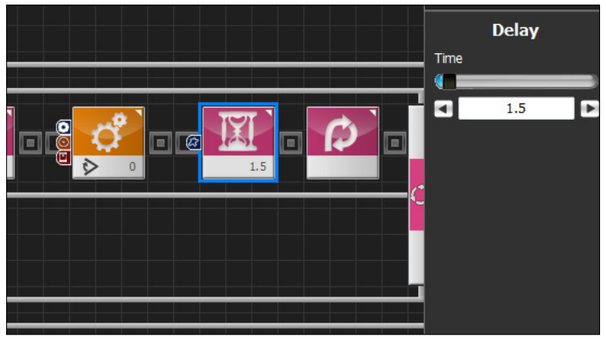
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.0으로 설정합니다. 1.0초 동안 기다리겠다는 의미입니다.



19 모터 0번(오른쪽 어깨) 내리기

팔을 들어 올린 상태에서 다시 내립니다.

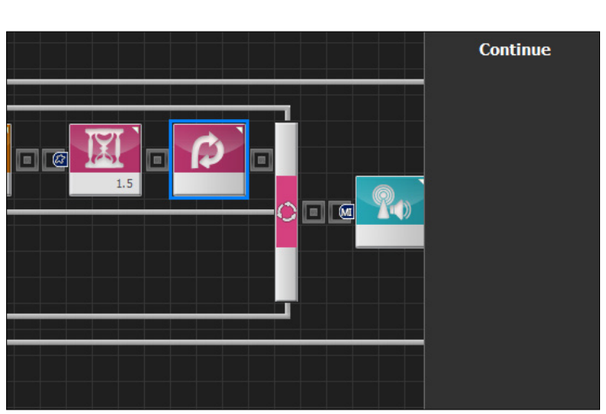
Motion 〉 Motor를 선택해 이전 모듈 뒤에 배치합니다.
Mode : Position으로 선택합니다.
Position : 235로 설정합니다. 235는 오른쪽 팔을 차려 자세로 만드는 위치 값 입니다.
Motor ID : 0으로 설정합니다. 오른쪽 어깨 모터 ID가 0번입니다.
Time : 40으로 설정합니다. 올라간 속도보다 조금 더 느리게, 약 0.448초 동안 내려옵니다.



20 동작 대기

실제로 팔을 내릴 때까지 시간이 필요하고, 구동이 끝난 후 소리 센서의 값이 초기화 되어야 하므로 기다립니다. 딜레이를 1.5초를 주어 구동이 끝난 후에 1초 이상 가만히 기다리도록 합니다.

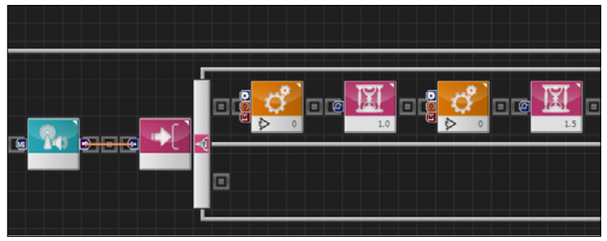
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



21 Continue 배치

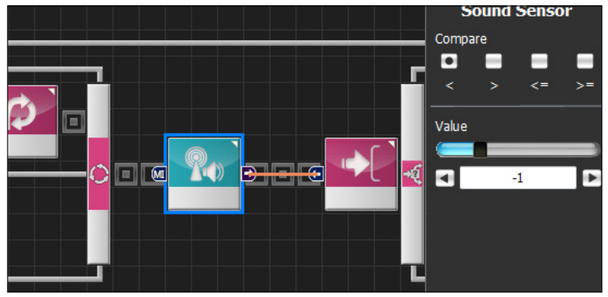
반복문의 처음으로 다시 돌아가서 계속하기 위해서 Continue를 배치해 Loop의 처음으로 이동합니다.

Flow 〉 Continue 모듈을 선택해 이전 모듈 뒤에 배치합니다.



22 요약

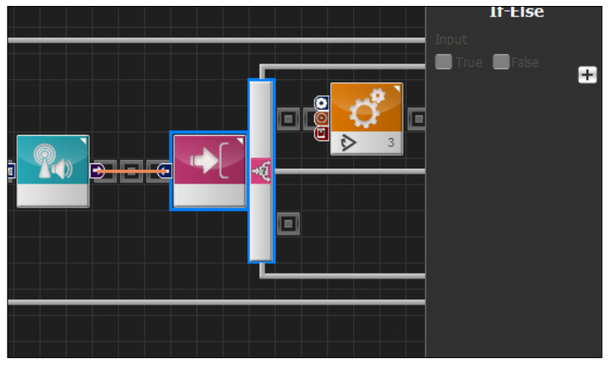
로봇의 오른쪽 소리 입력에 따라서 로봇이 동작하는 동안 외부의 자극을 의도적으로 막는 조건 분기문을 작성해 봤습니다.
이러한 프로그램은 로봇의 동작이나 외부 소음으로 인해 동작 중에 다른 소리 입력이 들어오는 것을 방지할 수 있어서 훨씬 정확한 동작이 가능합니다.



23 Sound Sensor 추가

소리 센서 모듈을 추가합니다. 왼쪽에 대해서 조건문을 만들어주기 위함입니다.

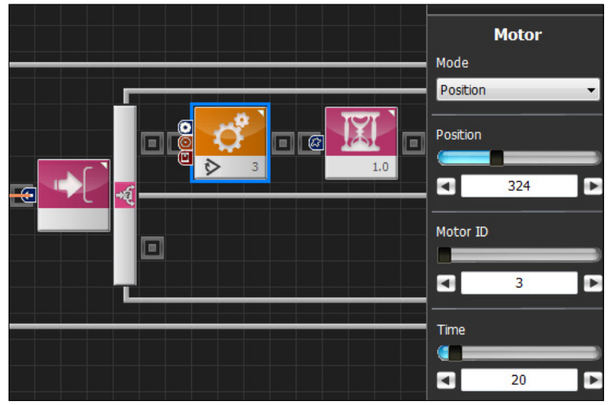
Sensor 〉 Sound 모듈을 선택해 If-Else 모듈의 뒤에 배치합니다.
Compare : <를 선택합니다. 센서 값이 Value 값보다 작을 때 True를 출력합니다.
Value : -1로 설정합니다. 왼쪽에서 소리가 입력되었음을 의미합니다.
소리가 입력 되었고, 소리 센서 값이 -1보다 작으면 True, 크거나 같으면 False를 출력합니다. 즉, 왼쪽에서 소리가 들어온 경우 True가 됩니다.



24 If-Else 문

소리 센서 모듈의 출력이 True면 왼팔 동작을 하기 위해서 조건문을 만듭니다.

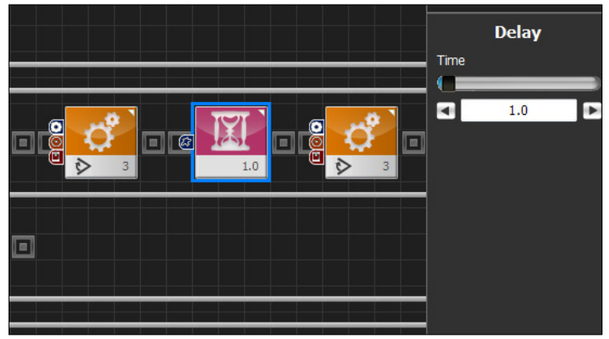
Flow 〉 If-Else를 선택해 직전 모듈의 뒤에 배치합니다.
23번의 소리 센서 출력을 If-Else문 입력에 연결합니다.



25 모터 3번(왼쪽 어깨) 설정

소리 입력이 들어왔으며 방향 값이 -1보다 작으면(True), 즉 왼쪽에서 소리가 들어왔을 경우 왼쪽 팔을 올립니다.

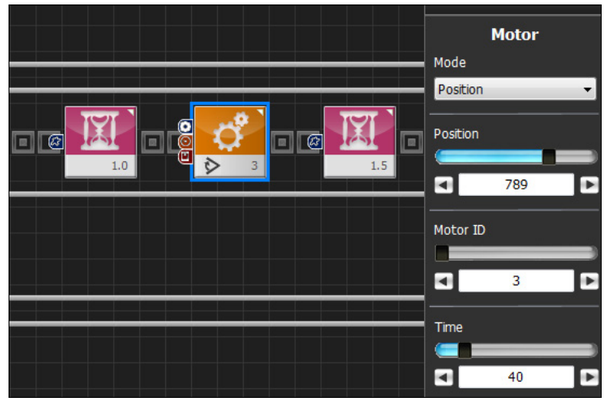
Motion 〉 Motor를 선택해 If-Else 문의 위 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 324으로 설정합니다. 324은 차려 자세로 놓여있는 팔을 앞으로 올리게 하는 위치 값 입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 20으로 설정합니다. 약 0.224초 동안 원하는 위치로 이동합니다.



26 동작 대기

들어 올리는 명령을 내리고 실제로 팔을 들어올릴 때까지 시간이 필요하므로 기다립니다. 딜레이를 1초를 주어 다 들어올린 후 잠시 멈춰 있도록 합니다.

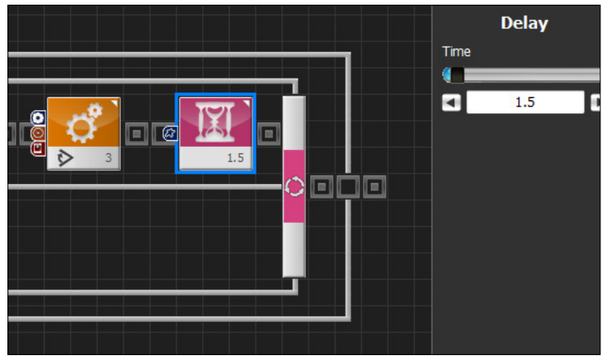
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.0으로 설정합니다. 1.0초 동안 기다리겠다는 의미입니다.



27 모터 3번(왼쪽 어깨) 설정

소리 입력이 들어오지 않았거나 방향 값이 -1보다 크거나 같으면(False), 즉 왼쪽에서 소리가 들어오지 않았을 경우 왼쪽 팔을 내립니다.

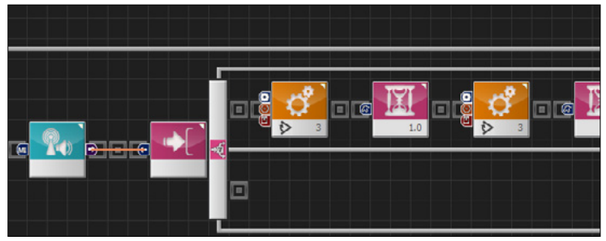
Motion 〉 Motor를 선택해 If-Else 문의 아래 프로그램 라인에 배치합니다.
Mode : Position으로 선택합니다.
Position : 789로 설정합니다. 789는 왼쪽 팔을 차려 자세로 만드는 위치 값 입니다.
Motor ID : 3으로 설정합니다. 왼쪽 어깨 모터 ID가 3번입니다.
Time : 40으로 설정합니다. 올라간 속도보다 조금 더 느리게, 약 0.448초 동안 내려옵니다.



28 동작 대기

실제로 팔을 내릴 때까지 시간이 필요하고, 구동이 끝난 후 소리 센서의 값이 초기화 되어야 하므로 기다립니다. 딜레이를 1.5초를 주어 구동이 끝난 후에 1초 이상 가만히 기다리도록 합니다.

Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



29 요약

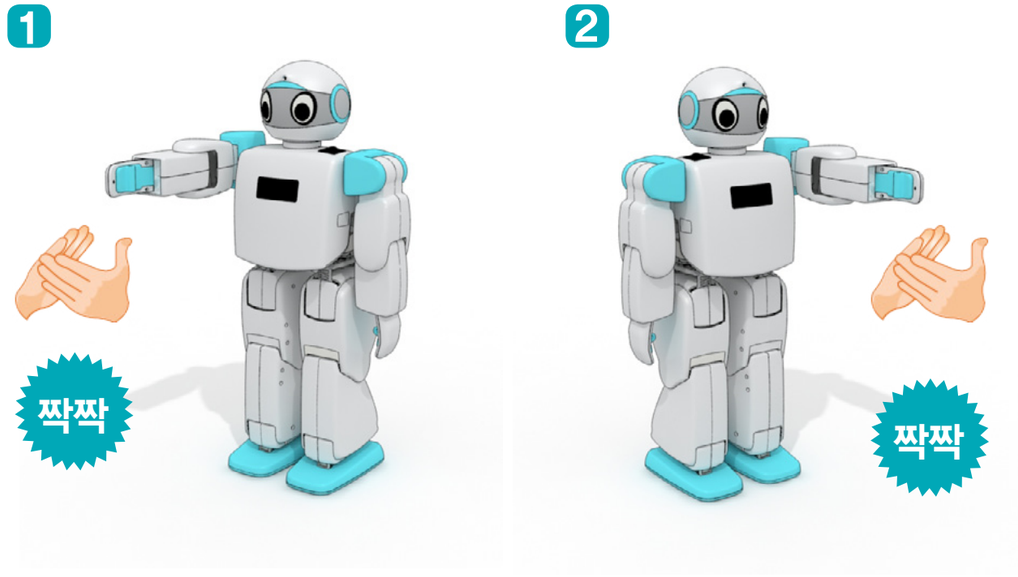
로봇의 왼쪽 소리 입력에 따라서 로봇이 동작하는 동안 외부의 자극을 의도적으로 막는 조건 분기문입니다.
마찬가지로 로봇의 동작이나 외부 소음으로 인해 동작 중에 다른 소리 입력이 들어오는 것을 방지할 수 있어서 훨씬 정확한 동작이 가능합니다.

30 다운로드

프로그래밍 후 컴파일 –〉 로봇에 다운로드 –〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB–to–Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



31 로봇동작

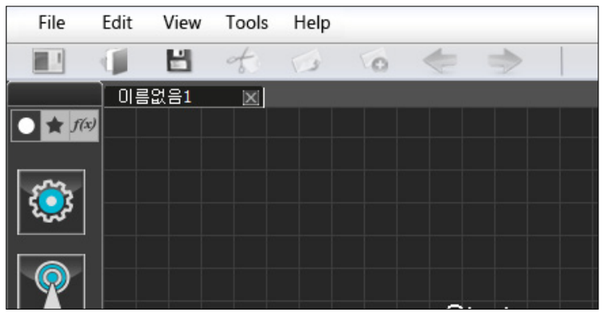
로봇 오른쪽에서 박수치면 오른쪽 팔을 올리고, 로봇 왼쪽에서 박수치면 왼쪽 팔을 올립니다.

08-7 모듈별 프로그래밍 Digtal Distanc

예제설명

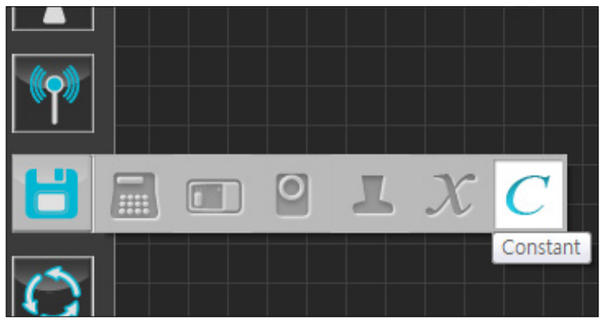
아날로그 거리센서는 측정 범위 내에서 거리 값을 측정 가능하지만, 디지털 거리센서는 특정 거리를 기준으로 가깝냐 머냐만을 판단합니다. 그래서 디지털 거리센서는 바퀴가 달린 로봇에 바닥을 향하게 장착되어 낭떠러지 감지용으로 많이 활용됩니다. 휴머노이드 로봇에서도 무릎 등에 장착되어 벽 감지용으로 사용되기도 합니다.

벽이 일정거리 가까워지면 뒷걸음질 하다가 방향을 틀고 전진하는 프로그램입니다. 이 예제를 실행하려면 ADC포트 1번(좌측)에 디지털 거리 센서를 장착한 상태로, 센서는 정면을 향해 부착돼 있어야 합니다.



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.

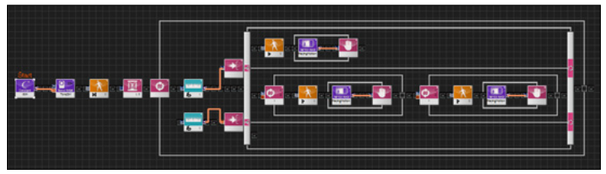


02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

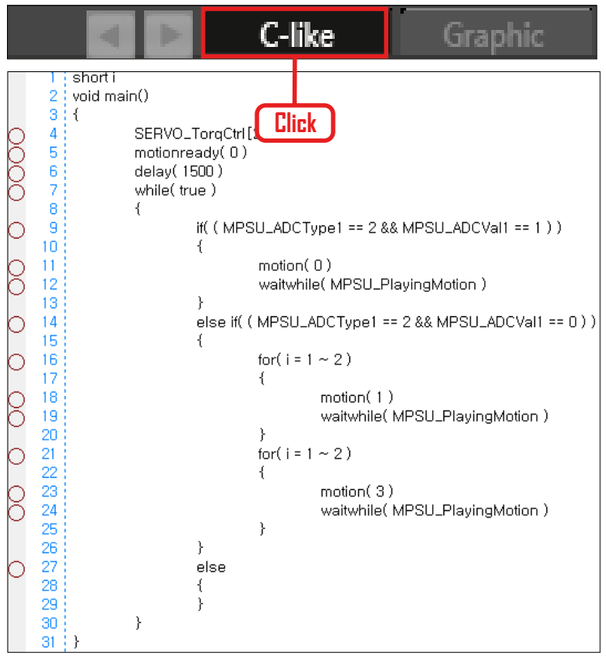
03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그래밍

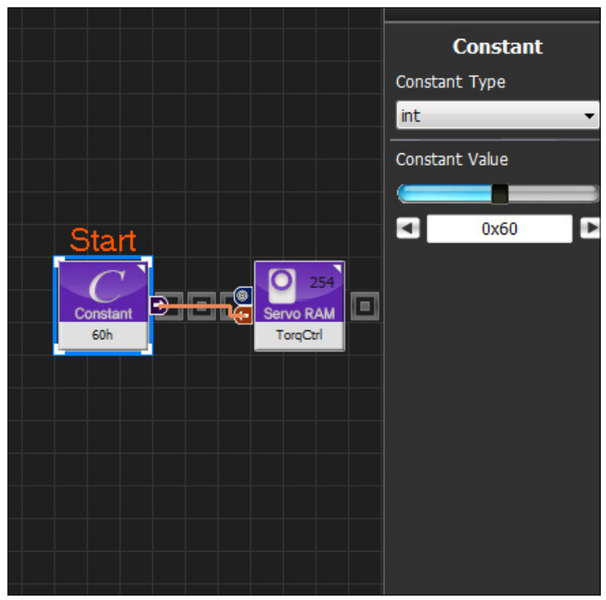
디지털 거리센서를 이용한 전체 프로그래밍입니다.



05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

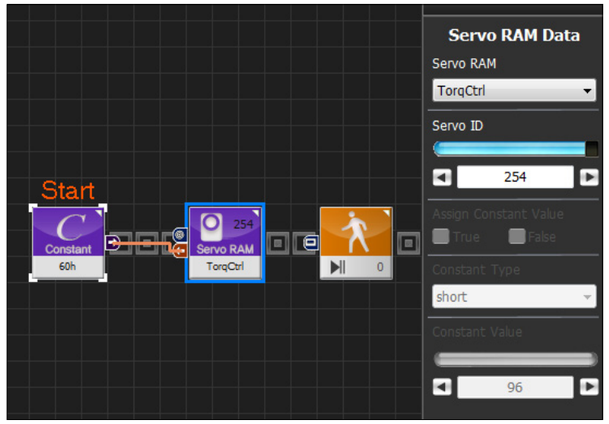
디지털 거리 센서로 벽을 인식하며 보행하는 프로그램입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

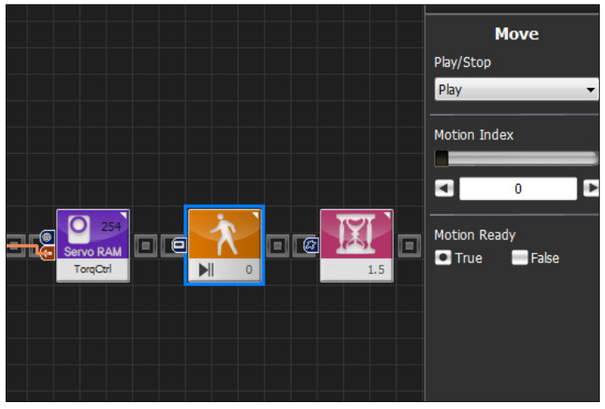
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다.
이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

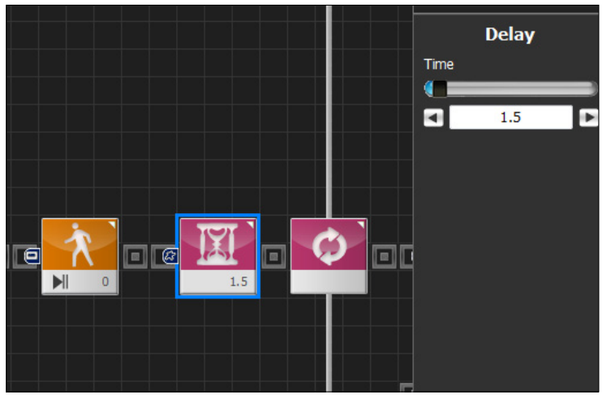
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



08 Motion Ready

저장된 모션을 가져올 때 로봇의 현재 상태에서 갑작스럽게 모션이 변동하고 움직일 수가 있습니다.
현재 상태와 모션 시작상태가 너무 다르면 모터에 무리가 가거나 사용자에게 위험할 수도 있습니다. 그래서 Motion Ready 모드로 모션을 실행해 모션이 동작할 준비시간을 줘야 합니다.

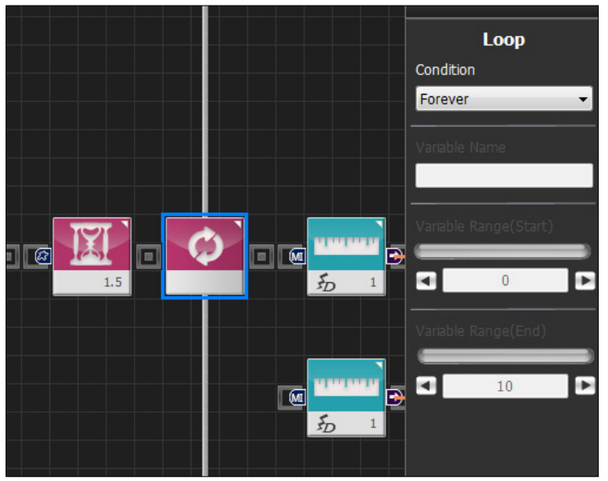
Motion 〉 Move 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 0를 선택합니다. 0번 모션을 가져오겠다는 뜻입니다. 기본 모션에서 0번 모션은 전진 모션입니다.
Motion Ready : True를 선택합니다. True를 선택하면 동작하고자 하는 모션의 첫 번째 상태로 서서히 이동합니다.



09 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 모듈입니다. 0번 모션의 초기 자세를 취하는 데에는 시간이 걸리기 때문에, 바로 다음으로 넘어가지 않고 모션이 끝날 때까지 기다리는 것입니다.

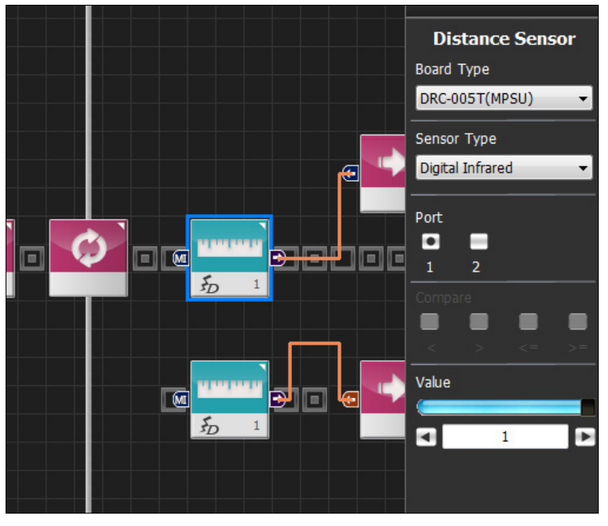
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



10 Loop 반복문

반복문을 넣어서 프로그램이 무한 반복하도록 합니다.

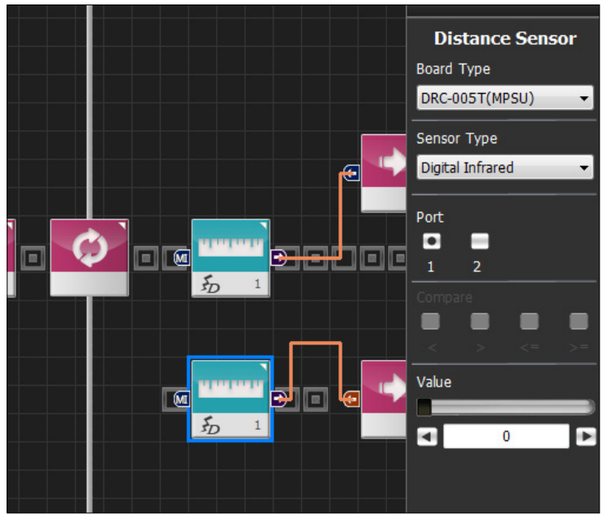
Flow 〉 Loop 모듈을 선택합니다.
Condition : Forever를 선택해 조건 없이 무한 반복하는 반복문을 만듭니다.



11 Distance Sensor 추가

디지털 거리 센서 모듈을 반복문 안에 추가합니다. 디지털 센서 모듈은 감지된 거리 값이 10cm 보다 크면 1, 10cm 보다 작으면 0이 나옵니다. 이 모듈은 선택한 센서 포트에 디지털 센서 모듈이 연결되어 있고, 그 값이 Value의 값과 같으면 True를 출력합니다. 센서가 연결되어 있지 않으면 False를 출력합니다.

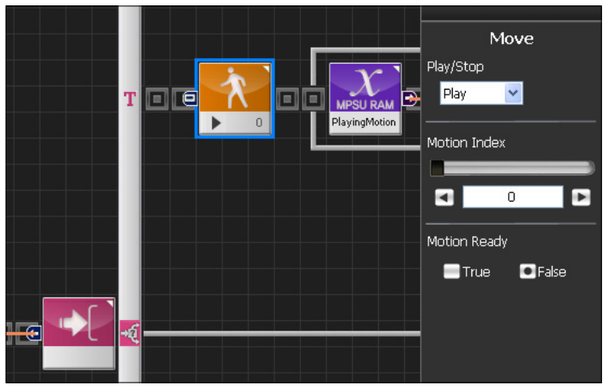
Sensor 〉 Distance를 선택해 Loop 모듈의 안쪽에 배치합니다.
Board Type : DRC-005T(MPSU)로 설정합니다. 제어기 DRC와 연결된 센서를 선택하겠다는 뜻입니다.
Sensor Type : Digital Infrared로 설정합니다. 디지털 적외선 거리 센서를 뜻합니다.
Port : 1로 설정합니다. 센서 포트 두 개 중 왼쪽 포트가 1번입니다.
Value : 1로 설정합니다.



12 Distance Sensor 추가

디지털 거리 센서 모듈을 하나 더 만들어서 11번에서 만든 센서 모듈 밑에 추가합니다. 이 모듈은 직전과는 달리 Value를 0으로 설정합니다.

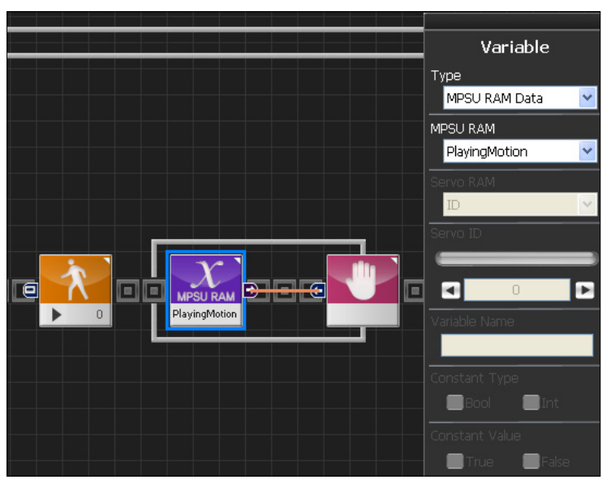
Sensor 〉 Distance를 선택해 11번에서 추가한 센서 모듈 밑에 추가합니다.
Board Type : DRC-005T(MPSU)로 설정합니다.
Sensor Type : Digital Infrared로 설정합니다.
Port : 1로 설정합니다.
Value : 0로 설정합니다.



13 If-Else문 생성

If-Else문을 생성하여 앞에서 만든 센서 모듈을 If-Else문과 연결합니다.

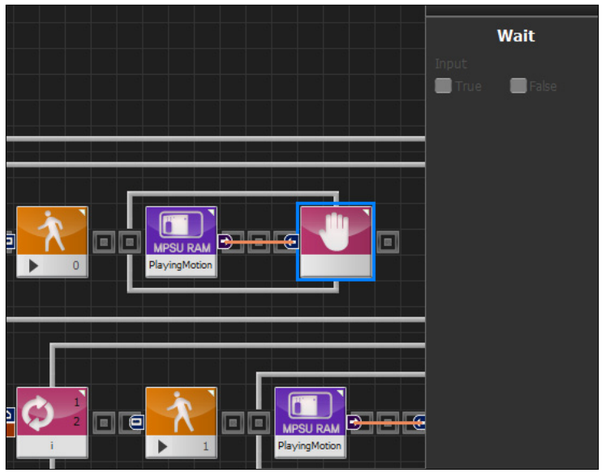
Flow 〉 If-Else 모듈을 선택해 Loop 모듈 안에 추가합니다.
+ 모양의 아이콘을 1번 클릭해서 조건문의 개수를 1개 더 추가합니다.
11번의 거리 센서 모듈 출력 핀을 첫 입력 핀에, 12번의 거리 센서 모듈 출력 핀을 둘째 입력 핀에 연결합니다.



14 전진 모션 실행

0번 모션(전진)을 실행하는 Move 모듈을 If-Else 모듈의 첫 프로그램 라인에 추가합니다. 센서 모듈에 의해서, 거리 값이 10cm 이상일 경우(센서 값이 1) 전진하게 됩니다.

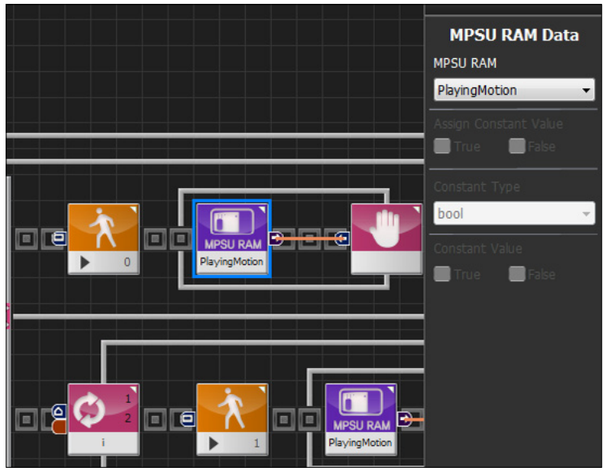
Motion 〉 Move 모듈을 선택해 If-Else 모듈의 첫 프로그램 라인에 추가합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 0을 선택합니다. 0번 모션을 실행하겠다는 뜻입니다. 기본 모션에서 0번 모션은 전진 모션입니다.
Motion Ready : False를 선택합니다.



15 Wait 추가

Move 모듈의 뒤에 Wait 모듈을 추가합니다. 모션이 끝날 때까지 대기하기 위해 사용됩니다.

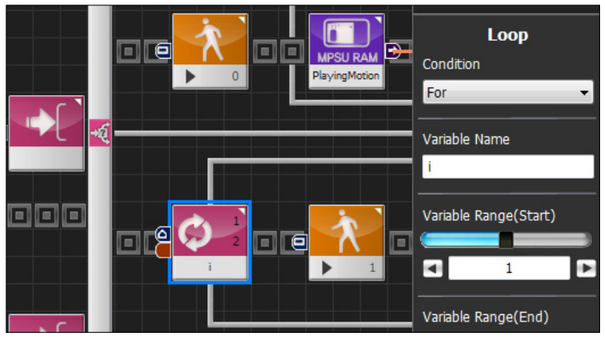
Flow 〉 Wait 모듈을 선택해 14번의 Move 모듈 뒤에 배치합니다.



16 PlayingMotion 모듈 추가

Move 명령을 내리고 나서 실제 모션이 실행되어 완료되기 까지는 시간이 걸리므로 loop안에 Move모듈 하나만을 넣고 실행하면 모션을 이미 실행중임에도 loop를 계속 돌면서 모션실행 명령을 반복하게 됩니다.
이렇게 되면 Move모듈을 만난 횟수와 실제 모션을 실행한 횟수가 달라집니다. 따라서 실행한 모션이 끝날 때까지 기다렸다가 다시 loop의 처음으로 돌아가게 하는 편이 더 정확합니다.
Data 〉 MPSU RAM 중에는 PlayingMotion 이라는 항목이 있는데, 이 레지스터는 로봇이 모션을 실행중이면 1, 그렇지 않으면 0인 변수입니다.
이것을 Wait의 입력 핀에 연결하면, 로봇의 모션이 실행중인 동안 Wait 모듈에서 대기하게 됩니다.

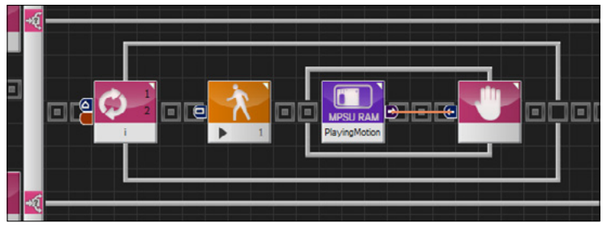
Data 〉 MPSU RAM을 선택해 Wait 모듈 안에 배치합니다.
MPSU RAM : PlayingMotion을 선택합니다.
모듈의 출력 핀을 Wait 모듈의 입력 핀에 연결합니다.



17 for문 만들기

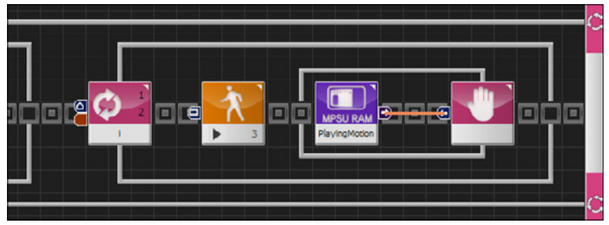
Loop 모듈은 무한 반복을 위해서도 쓰이지만, 특정 횟수 반복을 위해서도 사용됩니다.
Loop 모듈을 추가 후 for문을 선택하고, 변수명과 변수 범위를 설정하면 변수가 그 범위 안에서 1씩 증가하며 반복합니다.

Flow 〉 Loop를 선택해 If-Else 문의 두 번째 프로그램 라인에 배치합니다.
Condition : For로 선택합니다.
Variable Name : i로 설정합니다. 다른 이름을 사용해도 됩니다.
Variable Range(Start) : 1로 설정합니다. 변수 i가 증가하기 시작할 값입니다.
Variable Range(End) : 2로 설정합니다. 변수 i가 증가를 마칠 값입니다.
Start와 End를 1, 2로 설정하면 Loop 안의 내용을 2번 반복하게 됩니다.



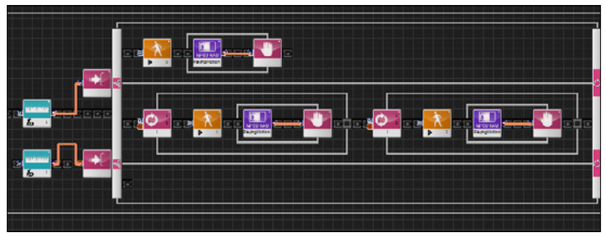
18 후진 모션 후 대기하기

14~16번의 세 개 모듈 추가 과정을 따라, 이번에는 모션 1번(후진)을 Loop 모듈 안에 추가하여 후진 모션 후 기다리는 동작을 구현합니다. 프로그램 실행 중 이 for문을 만나면, 후진 모션을 2번 실행하게 됩니다.



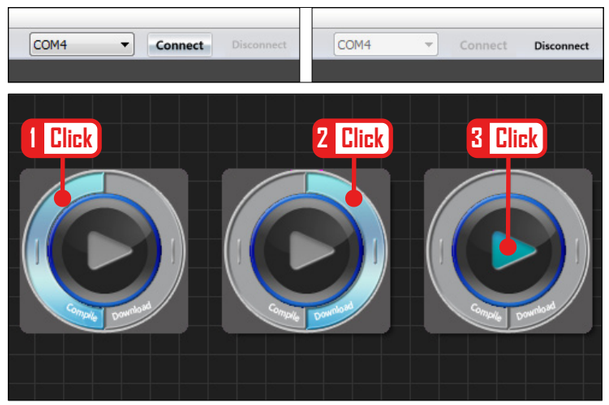
19 우회전 2번 반복하기

17~18번에서 만든 for문의 뒤에 같은 방법으로 모션 3번(우회전)을 추가하여, 우회전을 2번 실행하도록 구현합니다.



20 요약

지금까지 구성한 전체 If-Else문의 모습입니다. 거리 센서 값이 1이라면(10cm 이상) 전진 모션을 실행합니다. 거리 센서 값이 0이라면(10cm 이하) 후진 모션을 2번, 우회전 모션을 2번 차례로 실행해 장애물을 회피합니다. If-Else 문이 끝나면 다시 Loop 문의 처음으로 돌아가 거리 센서 값에 따라서 다시 If-Else 문으로 진입해 반복합니다.

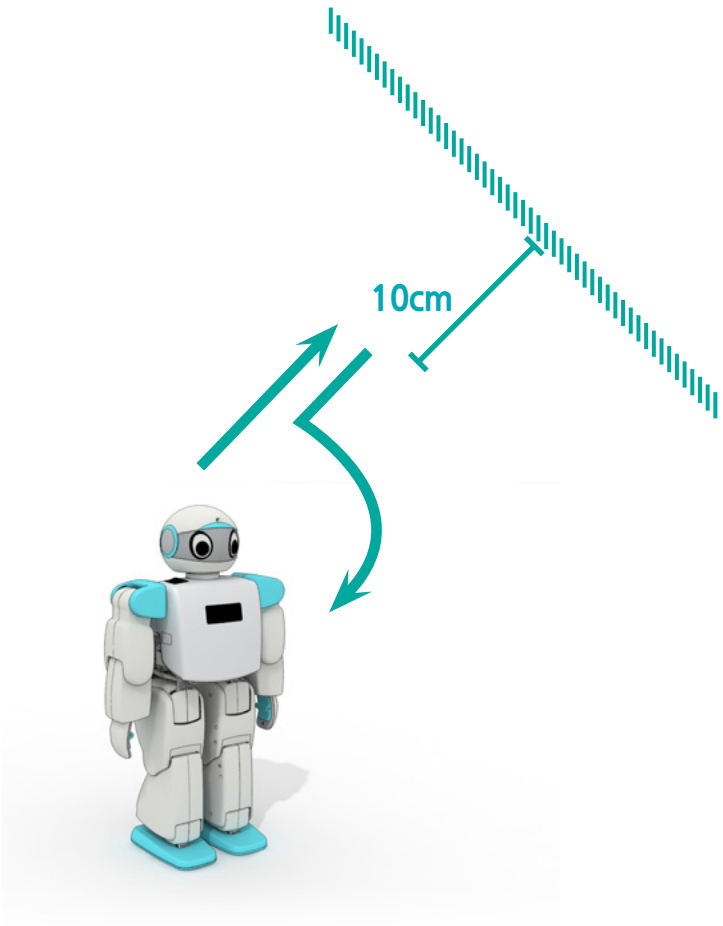


21 다운로드

프로그래밍 후 컴파일 -〉 로봇에 다운로드 -〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



22 로봇동작

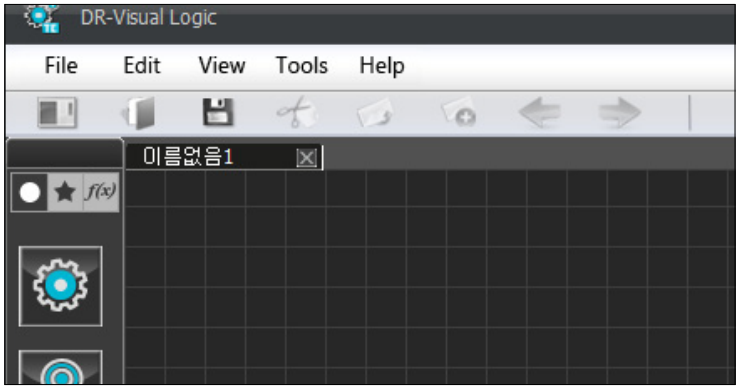
로봇이 10cm 앞의 벽을 감지하면 후진하였다가 우회전 한 후 전진합니다.

모듈별 프로그래밍

Analog Distance 08-8

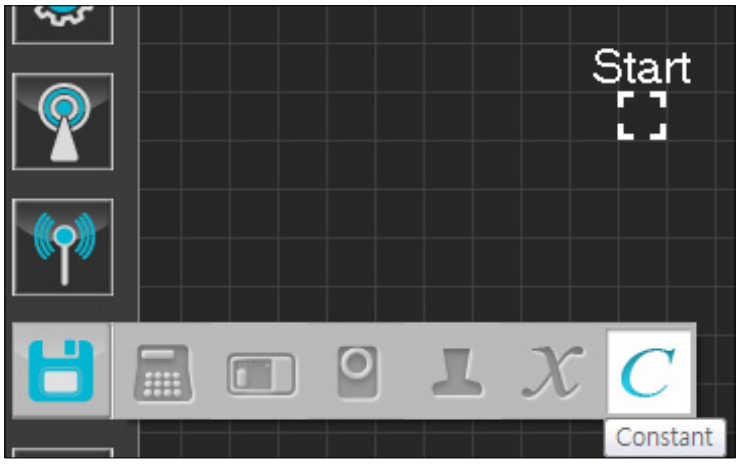
예제설명

아날로그 거리센서를 이용하여, 벽이 가까워지면 좌회전 하면서 벽을 회피하는 프로그램입니다. 거리센서에는 아날로그 센서와 디지털 센서가 있습니다. 디지털 센서는 일정거리(10cm) 를 기준으로 안쪽과 바깥쪽 두 영역만 감지하는 반면, 아날로그 센서는 거리별로(6~40cm) 위치를 감지할 수 있습니다. 이 예제를 실행하려면 ADC 포트 1번 (좌측)에 PSD센서를 연결하고, 로봇의 전면을 바라보고 센서를 장착된 상태여야 합니다.



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.

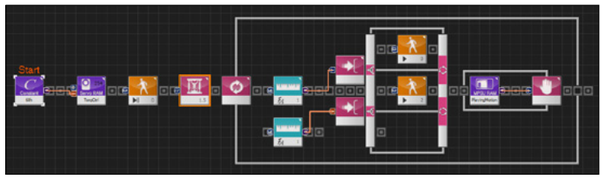


02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

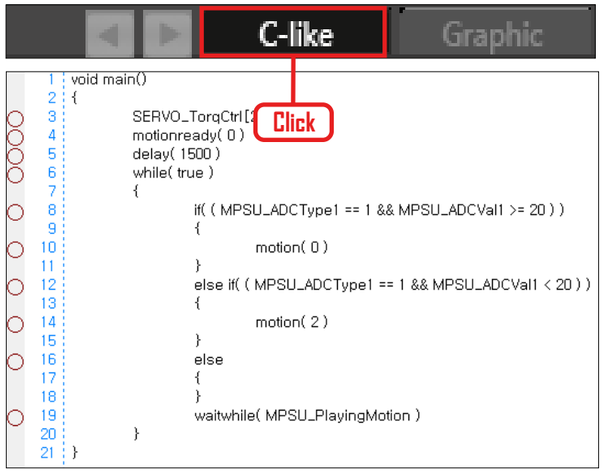
03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그래밍

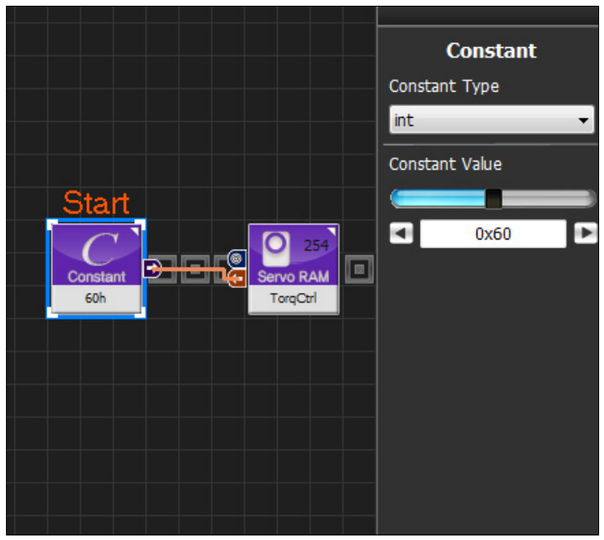
아날로그 거리센서와 전진 모션, 좌회전 모션을 사용한 자율 보행 프로그램입니다.



05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

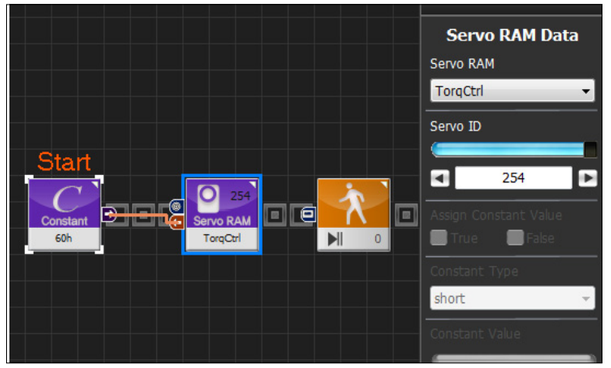
20cm 안에 물체가 있다면 좌회전을, 없다면 전진을 하는 프로그램 소스입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

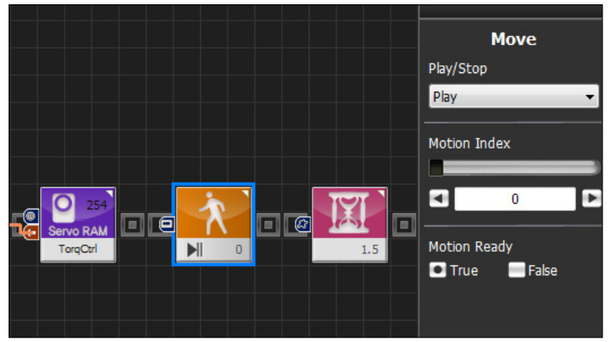
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다.
이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

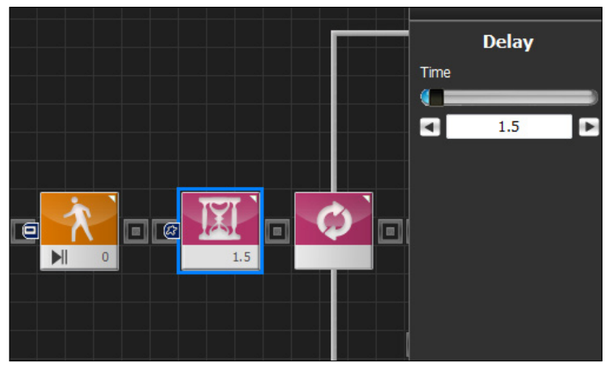
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



08 Motion Ready

저장된 모션을 가져올 때 로봇의 현재 상태에서 갑작스럽게 모션이 변동하고 움직일 수가 있습니다.
현재 상태와 모션 시작상태가 너무 다르면 모터에 무리가 가거나 사용자에게 위험할 수도 있습니다. 그래서 Motion Ready 모드로 모션을 실행해 모션이 동작할 준비시간을 줘야 합니다.

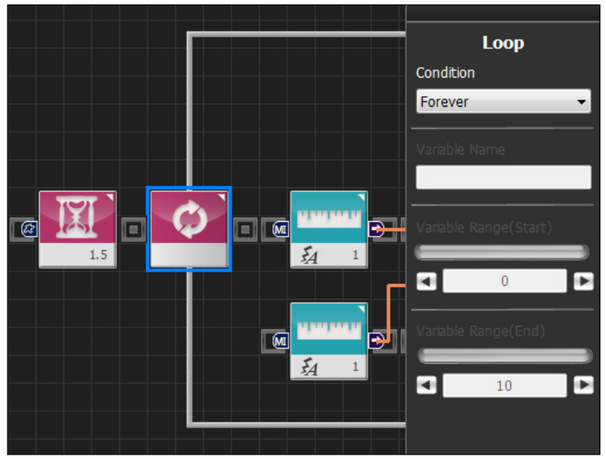
Motion 〉 Move 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 0를 선택합니다. 0번 모션을 가져오겠다는 뜻입니다. 기본 모션에서 0번 모션은 전진 모션입니다.
Motion Ready : True를 선택합니다. True를 선택하면 동작하고자 하는 모션의 첫 번째 상태로 서서히 이동합니다.



09 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 모듈입니다. 0번 모션의 초기 자세를 취하는 데에는 시간이 걸리기 때문에, 바로 다음으로 넘어가지 않고 모션이 끝날 때까지 기다리는 것입니다.

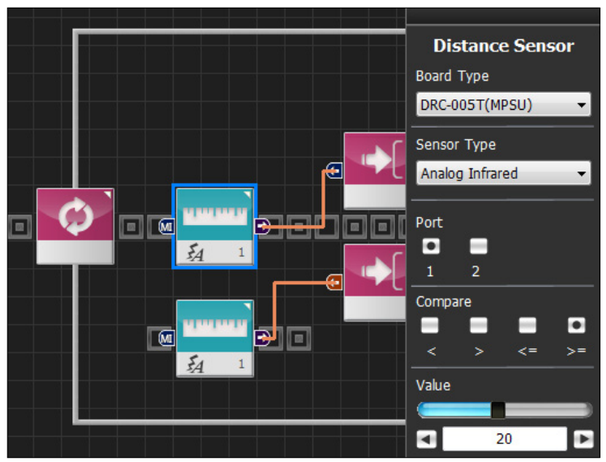
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



10 Loop 반복문

반복문을 넣어서 프로그램이 무한 반복하도록 합니다.

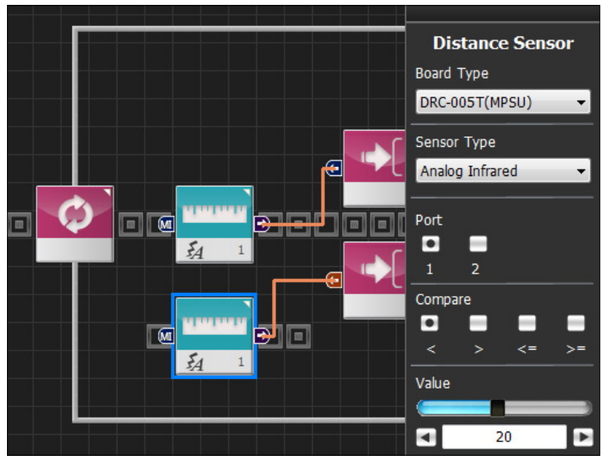
Flow 〉 Loop 모듈을 선택합니다.
Condition : Forever를 선택해 조건 없이 무한 반복하는 반복문을 만듭니다.



11 Distance Sensor 추가

아날로그 거리 센서 모듈을 반복문 안에 추가합니다. 아날로그 센서 모듈은 감지된 거리 값을 cm 단위로 출력합니다. 이 모듈은 선택한 센서 포트에 아날로그 센서 모듈이 연결되어 있으면, 그 값을 Compare에 따라 Value와 비교해서 True/False를 출력합니다. 센서가 연결되어 있지 않으면 False를 출력합니다.

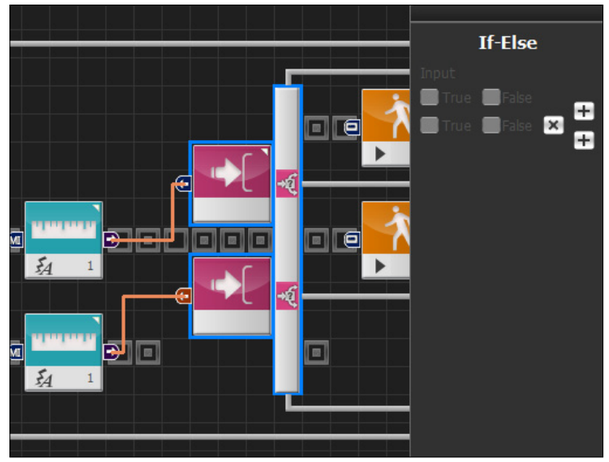
Sensor 〉 Distance를 선택해 Loop 모듈의 안쪽에 배치합니다.
Board Type : DRC-005T(MPSU)로 설정합니다. 제어기 DRC와 연결된 센서를 선택하겠다는 뜻입니다.
Sensor Type : Analog Infrared로 설정합니다. 아날로그 적외선 거리 센서를 뜻합니다.
Port : 1로 설정합니다. 센서 포트 두 개 중 왼쪽 포트가 1번입니다.
Compare : 〉=를 선택합니다. 센서 값이 Value 보다 크거나 같을 때 참이 됩니다.
Value : 20으로 설정합니다.



12 Distance Sensor 추가

아날로그 거리 센서 모듈을 하나 더 만들어서 11번에서 만든 센서 모듈 밑에 추가합니다. 이 모듈은 직전과는 달리 Compare를 〈으로 설정합니다.

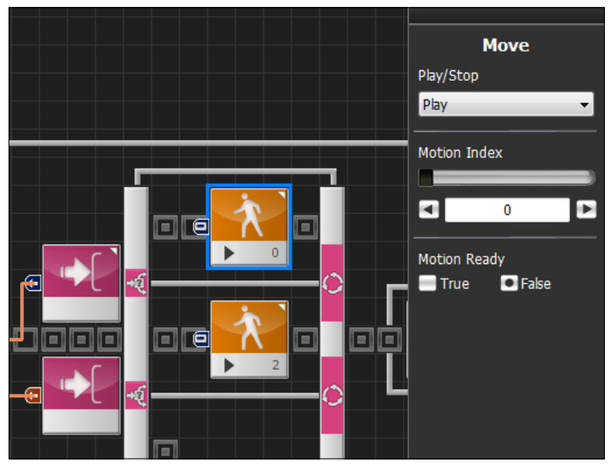
Sensor 〉 Distance를 선택해 11번에서 추가한 센서 모듈 밑에 추가합니다.
Board Type : DRC-005T(MPSU)로 설정합니다.
Sensor Type : Analog Infrared로 설정합니다.
Port : 1로 설정합니다.
Compare : 〈를 선택합니다. 센서 값이 Value 보다 작을 때 참이 됩니다.
Value : 0로 설정합니다.



13 If-Else문 생성

If-Else문을 생성하여 앞에서 만든 센서 모듈을 If-Else문과 연결합니다.

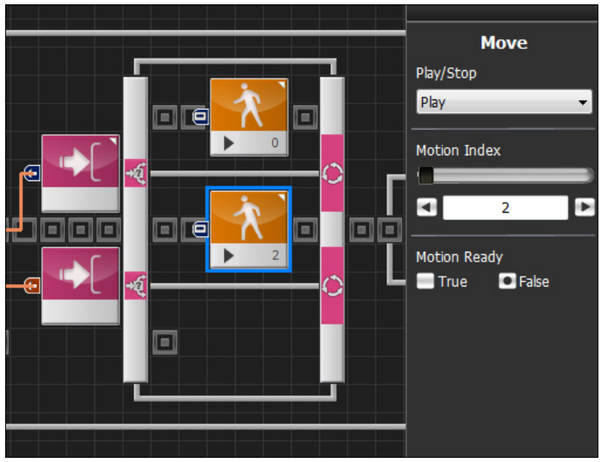
Flow 〉 If-Else 모듈을 선택해 Loop 모듈 안에 추가합니다.
+ 모양의 아이콘을 1번 클릭해서 조건문의 개수를 1개 더 추가합니다.
11번의 거리 센서 모듈 출력 핀을 첫 입력 핀에, 12번의 거리 센서 모듈 출력 핀을 둘째 입력 핀에 연결합니다.



14 전진 모션 실행

0번모션(전진)을 실행하는 Move 모듈을 If-Else 모듈의 첫 프로그램 라인에 추가합니다. 센서 모듈의 출력에 의해서, 거리 값이 20cm 이상일 경우 전진하게 됩니다.

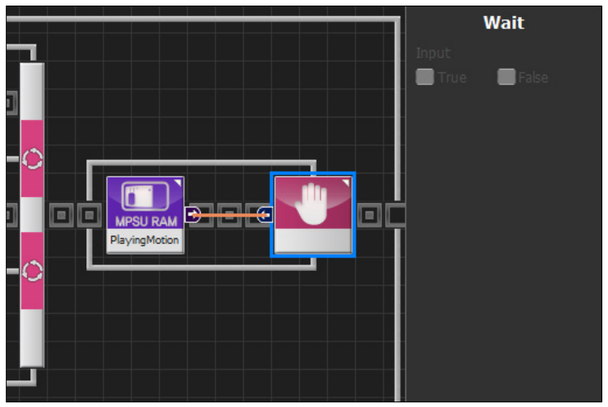
Motion 〉 Move 모듈을 선택해 If-Else 모듈의 첫 프로그램 라인에 추가합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 0을 선택합니다. 0번 모션을 실행하겠다는 뜻입니다. 기본 모션에서 0번 모션은 전진 모션입니다.
Motion Ready : False를 선택합니다.



15 좌회전 모션 실행

2번 모션(좌회전)을 실행하는 Move 모듈을 If-Else 모듈의 두 번째 프로그램 라인에 추가합니다. 센서 모듈의 출력에 의해서, 거리 값이 20cm 미만일 경우 좌회전하게 됩니다.

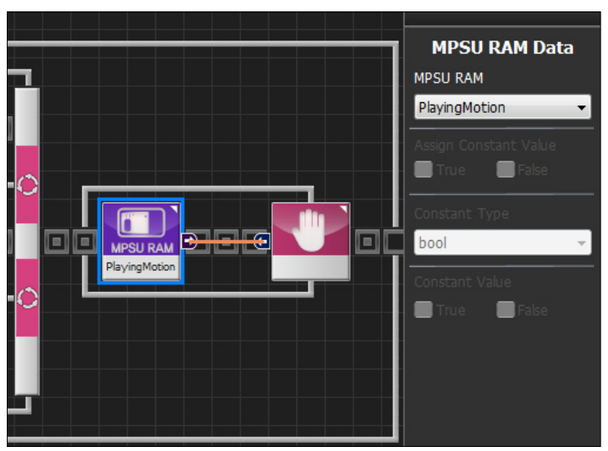
Motion 〉 Move 모듈을 선택해 If-Else 모듈의 두 번째 프로그램 라인에 추가합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 2를 선택합니다. 2번 모션을 실행하겠다는 뜻입니다. 기본 모션에서 2번 모션은 좌회전 모션입니다.
Motion Ready : False를 선택합니다.



16 Wait 추가

If-Else 모듈의 뒤에 Wait 모듈을 추가합니다. If-Else 모듈 안에서 실행한 모션이 끝날 때까지 대기하기 위해 사용됩니다.

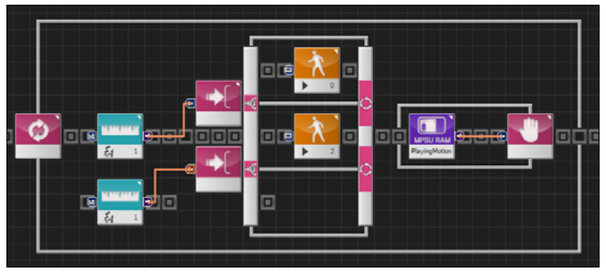
Flow 〉 Wait 모듈을 선택해 If-Else 모듈 뒤에 배치합니다.



17 PlayingMotion 모듈 추가

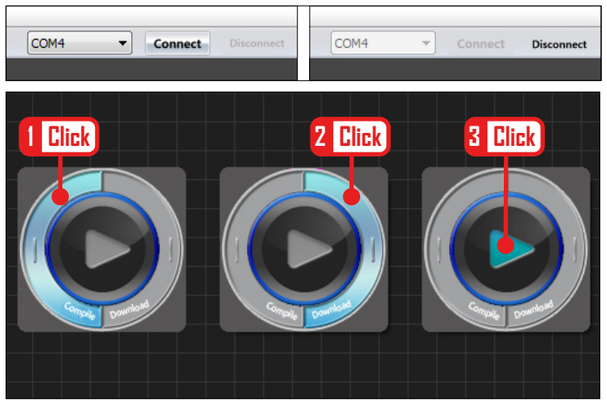
Data 〉 MPSU RAM 중에는 PlayingMotion이라는 항목이 있는데, 이 레지스터는 로봇이 모션을 실행중이면 1, 그렇지 않으면 0인 변수입니다.
이것을 Wait의 입력 핀에 연결하면, 로봇의 모션이 실행이 끝날 때까지 Wait 모듈에서 대기하게 됩니다.

Data 〉 MPSU RAM을 선택해 Wait 모듈 안에 배치합니다.
MPSU RAM : PlayingMotion을 선택합니다.
모듈의 출력 핀을 Wait 모듈의 입력 핀에 연결합니다.



18 요약

지금까지 구성한 전체 Loop문의 모습입니다. 거리 센서 값이 20cm 이상이라면 전진 모션을 실행합니다. 거리 센서 값이 20cm 미만이라면 좌회전 모션을 실행합니다. 두 경우 모두 False인 경우(센서가 연결되지 않은 경우) 아무 일도 하지 않습니다. If-Else 문이 끝나면 Wait을 통해 모션이 끝날 때까지 대기 후, 다시 Loop 문의 처음으로 돌아가 거리 센서 값에 따라서 다시 If-Else 문으로 진입해 반복합니다.

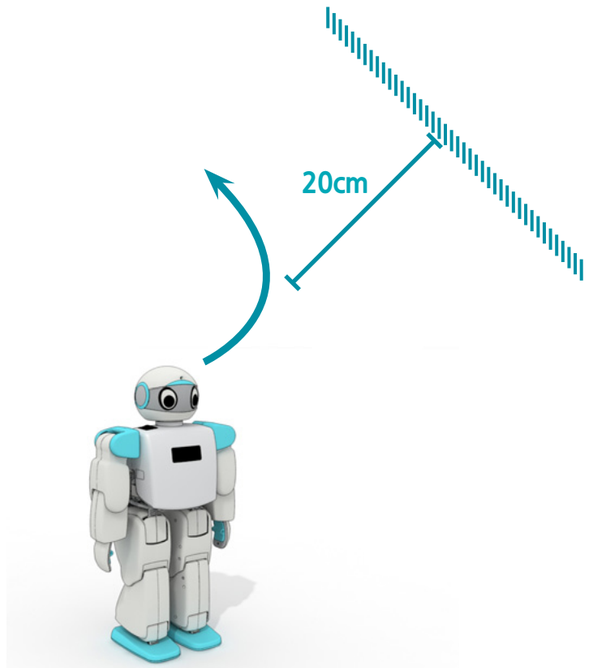


19 다운로드

프로그래밍 후 컴파일 -〉 로봇에 다운로드 -〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



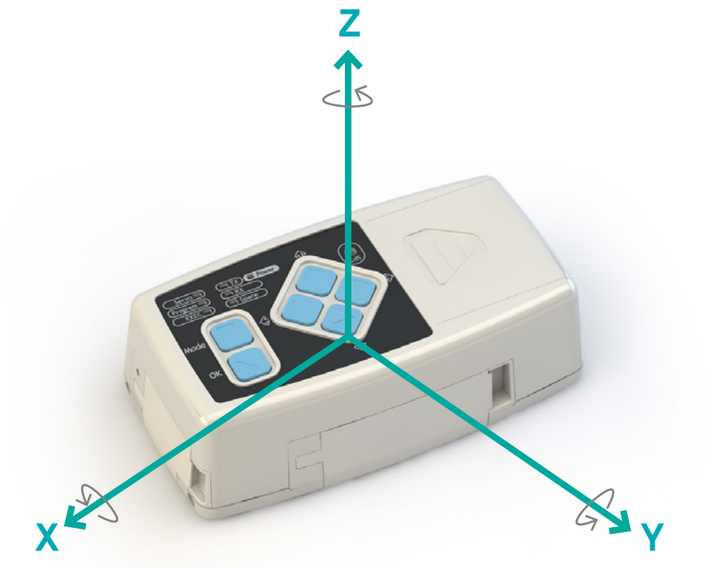
20 로봇동작

로봇이 벽을 향해 전진하다가 20cm 안으로 근접하면 좌회전 합니다.

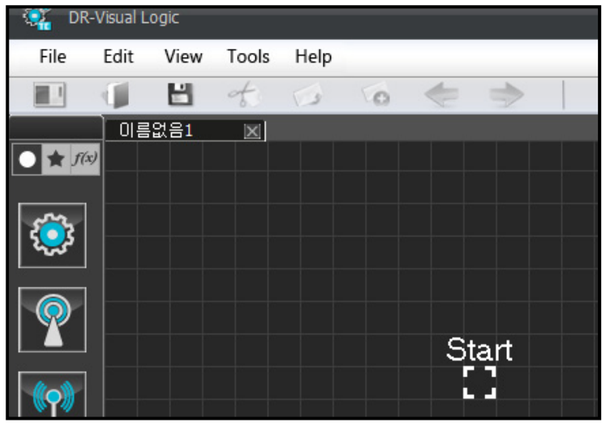
08-9 모듈별 프로그래밍 Acceleration

예제설명

Acceleration (가속도) 센서를 이용하여 로봇이 각각 앞으로 넘어졌을 때와 뒤로 넘어졌을 때 일어나는 프로그램을 작성해봅니다. 가속도 센서는 제어기 뒷 커버를 열고 넣을 수 있는 모듈 형태로서 Gyro 센서와 결합되어 있습니다.

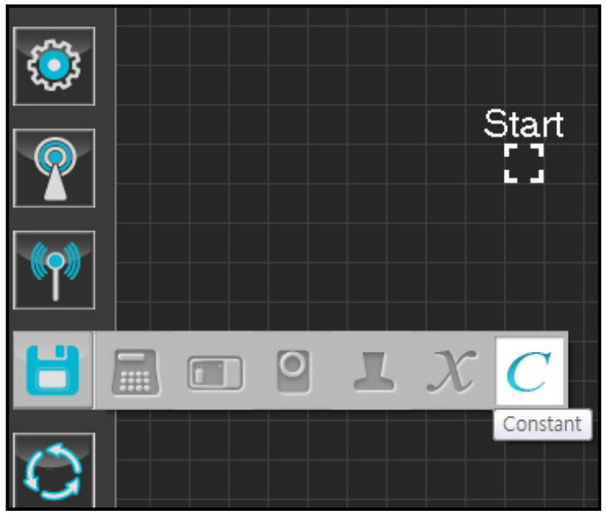


- 로봇이 엎드릴 때 Z축은 “+“ 가속도가 붙고 그 값은 약 +256 입니다.
- 로봇이 누울때 Z축은 “–“ 가속도가 붙고 그 값은 약 –256 입니다.
  (256 은 약 1g 중력값을 나타냅니다.)



01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.

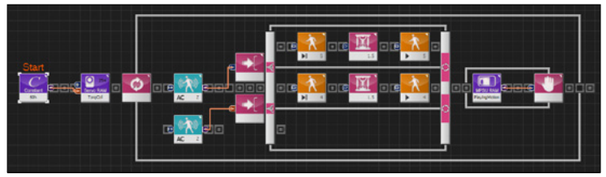


02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Constant 모듈을 클릭합니다.

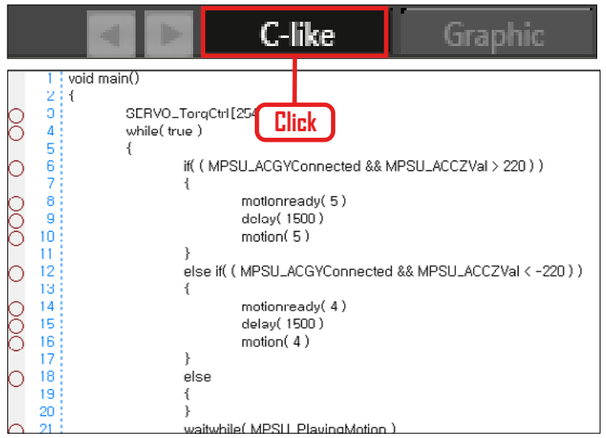
03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그램

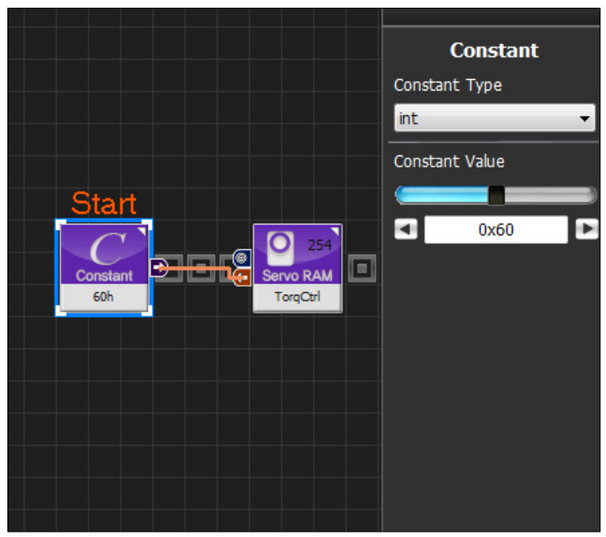
가속도 센서를 이용하여 넘어짐을 감지, 로봇이 일어나는 프로그램입니다.



05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 프로그램 소스 화면이 나옵니다.

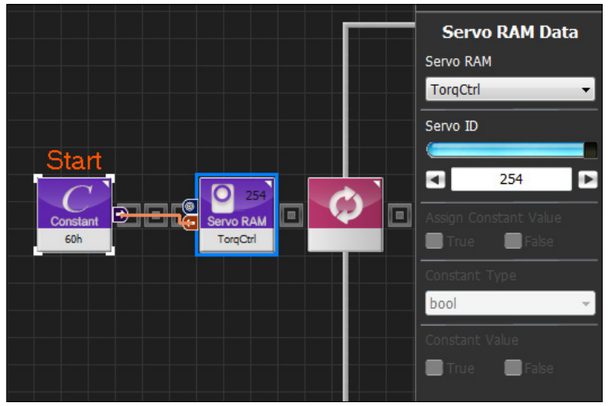
중력 가속도를 읽어 넘어진 방향을 감지, 일어나는 모션을 실행합니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 소스로 어떻게 변환되는지 확인할 수 있습니다.



06 상수 설정

서보 모터를 스스로 움직일 수 있는 상태로 만드는 과정입니다.

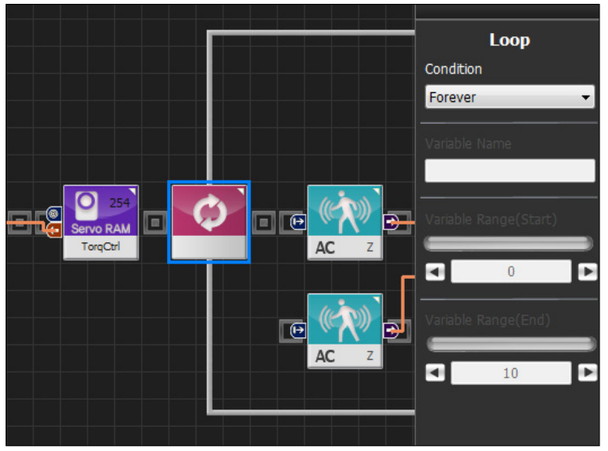
Constant 모듈의 속성 중 Constant Value에 있는 칸을 클릭하여 값을 0x60으로 바꿉니다. 0x60은 16진수로, 서보 모터에 토크가 인가된 상태를 나타내는 상수입니다.
이 값은 출력 핀을 통하여 다음 모듈의 입력 핀에 전달됩니다.



07 모든 서보에 적용

앞에서 받은 0x60이라는 상수 값을 모든 서보에 적용하는 과정입니다.

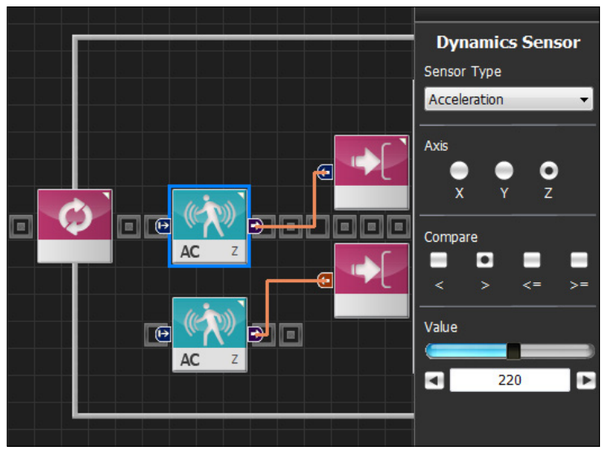
Data 〉 Servo RAM을 선택해서 Constant 모듈의 뒤에 배치합니다.
Servo RAM : TorqCtrl을 선택합니다.
Servo ID : 254로 설정합니다. 254라는 ID는 연결 되어 있는 모든 서보에 적용하겠다는 의미입니다.
그리고 앞에 있는 Constant 모듈의 출력 핀을 Servo RAM 모듈의 두 번째 입력 핀에 커넥터로 연결합니다.



08 Loop 반복문

반복문을 넣어서 프로그램이 무한 반복하도록 합니다.

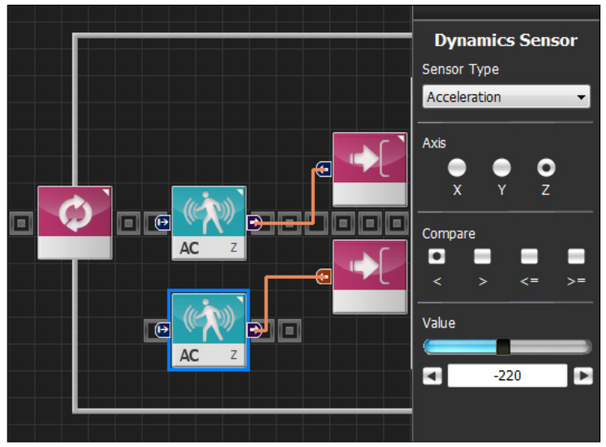
Flow 〉 Loop 모듈을 선택합니다.
Condition : Forever를 선택해 조건 없이 무한 반복하는 반복문을 만듭니다.



09 가속도 센서 추가

가속도 센서 모듈을 반복문 안에 추가합니다. Z축 가속도는 로봇이 똑바로 서 있을 때 0, 엎드려 넘어졌을 때 약 +256, 누워서 넘어졌을 때 약 –256 값이 나타납니다. 따라서 +256 값에 근접했을 때 로봇이 엎드려서 넘어졌다고 할 수 있습니다. 기준 값을 +220으로 설정합니다.
이 모듈은 가속도/자이로 센서 모듈이 연결되어 있으면, 그 값을 Compare에 따라 Value와 비교해서 True/False를 출력합니다. 센서가 연결되어 있지 않으면 False를 출력합니다.

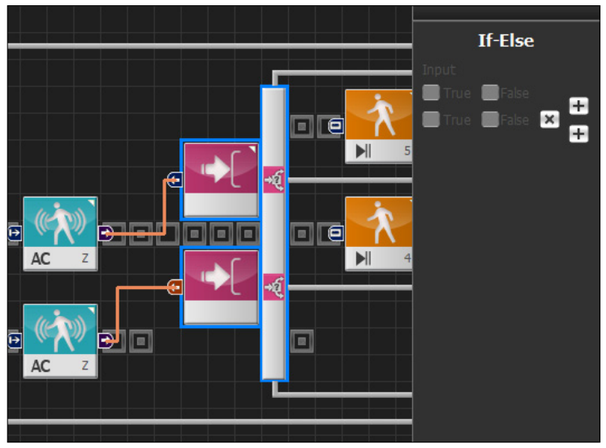
Sensor 〉 Dynamics 를 선택해 Loop 모듈의 안쪽에 배치합니다.
Sensor Type : Acceleration을 선택합니다. 가속도 센서를 뜻합니다.
Axis : Z축으로 설정합니다.
Compare : 〉로 설정합니다. 센서 값이 Value보다 클 때 출력 값이 참이 됩니다.
Value : 220으로 설정합니다.



10 가속도 센서 추가

가속도 센서 모듈을 하나 더 만들어서 11번에서 만든 센서 모듈 밑에 추가합니다. 이 모듈은 직전과는 달리 Compare를 〈으로 설정하고 Value를 –220으로 설정합니다. 이 모듈은 로봇이 뒤로 넘어졌을 때, 즉 –256 값에 근접했을 때 출력 값이 True가 됩니다.

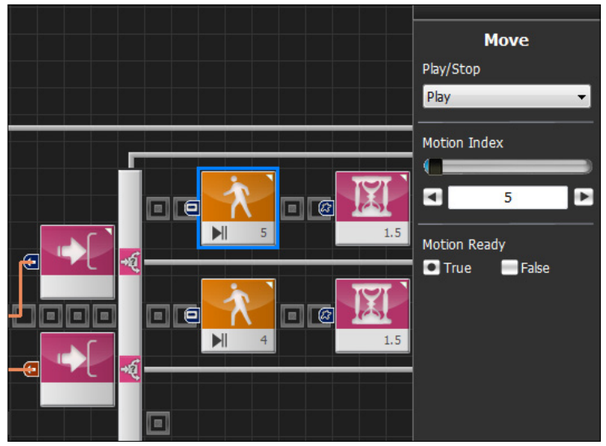
Sensor 〉 Dynamics 를 선택해 Loop 모듈의 안쪽에 배치합니다.
Sensor Type : Acceleration을 선택합니다. 가속도 센서를 뜻합니다.
Axis : Z축으로 설정합니다.
Compare : 〈로 설정합니다. 센서 값이 Value보다 작을 때 출력 값이 참이 됩니다.
Value : –220으로 설정합니다.



11 If-Else문 생성

If–Else문을 생성하여 앞에서 만든 센서 모듈을 If–Else문과 연결합니다.

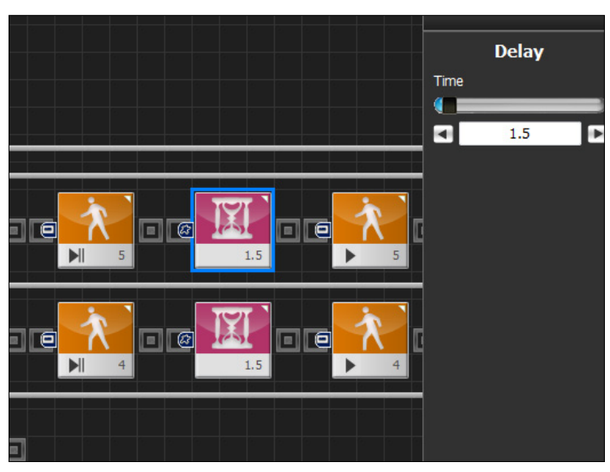
Flow 〉 If–Else 모듈을 선택해 Loop 모듈 안에 추가합니다.
+ 모양의 아이콘을 1번 클릭해서 조건문의 개수를 1개 더 추가합니다.
9번의 가속도 센서 모듈 출력 핀을 첫 입력 핀에, 10번의 가속도 센서 모듈 출력 핀을 둘째 입력 핀에 연결합니다.



12 Motion Ready

If–Else의 첫 프로그램 라인은 로봇이 엎드려서 넘어졌을 때 실행됩니다. 이 경우 엎드린 자세에서 일어나는 모션을 실행해야 합니다.
저장된 모션을 실행할 때 로봇의 현재 상태에서 갑작스럽게 모션이 변동하고 움직일 수가 있습니다.
현재 상태와 모션 시작상태가 너무 다르면 모터에 무리가 가거나 사용자에게 위험할 수도 있습니다. 그래서 Motion Ready 모드로 모션을 실행해 모션이 동작할 준비시간을 줘야 합니다.

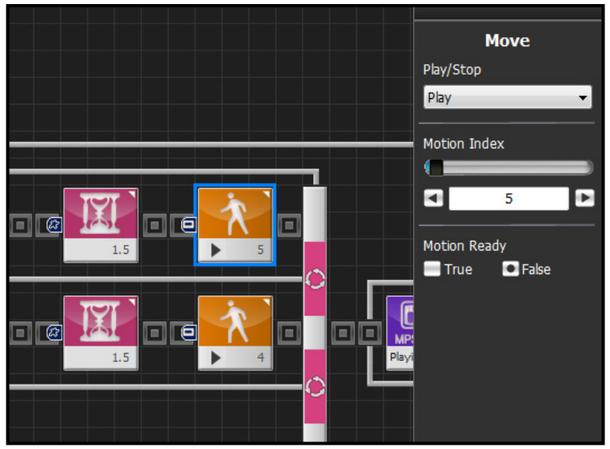
Motion 〉 Move 모듈을 선택해 If–Else 모듈의 첫 프로그램 라인에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 5를 선택합니다. 5번 모션을 실행하겠다는 뜻입니다. 기본 모션에서 5번 모션은 엎드린 상태에서 일어나는 모션입니다.
Motion Ready : True를 선택합니다. True를 선택하면 동작하고자 하는 모션의 첫 번째 상태로 서서히 이동합니다.



13 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 모듈입니다. 모션의 초기 자세를 취하는 데에는 시간이 걸리기 때문에, 바로 다음으로 넘어가지 않고 모션이 끝날 때까지 기다리는 것입니다.

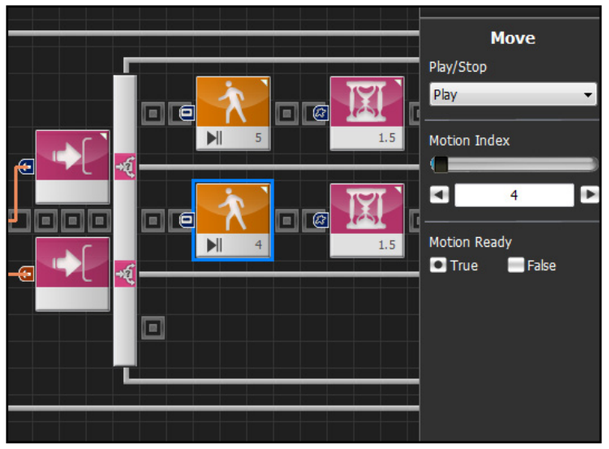
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



14 일어나기 모션 실행

5번 모션(엎드린 상태에서 일어나기)을 실행하는 Move 모듈을 추가합니다.

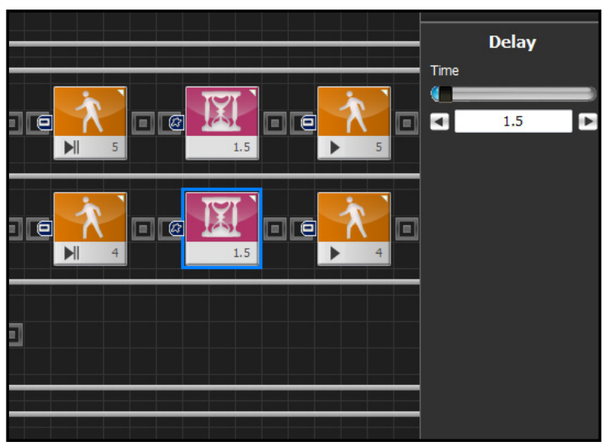
Motion 〉 Move 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 5을 선택합니다. 5번 모션을 실행하겠다는 뜻입니다.
Motion Ready : False를 선택합니다.



15 Motion Ready

If-Else의 두 번째 프로그램 라인은 로봇이 누워서 넘어졌을 때 실행됩니다. 이 경우 누운 자세에서 일어나는 모션을 실행해야 합니다.
기본 모션에서 4번 모션은 누운 상태에서 일어나는 모션입니다. 마찬가지로 Motion Ready로 모션의 준비 자세를 천천히 실행합니다.

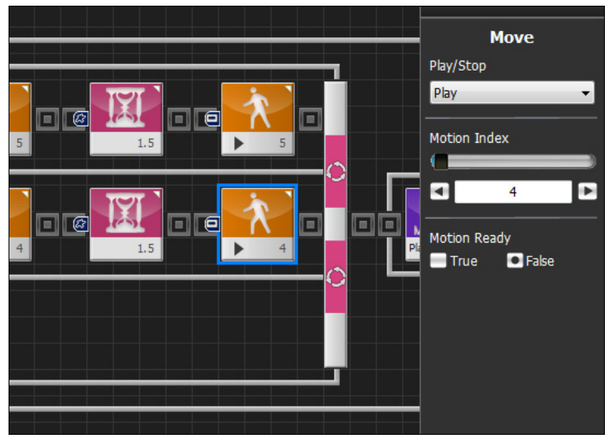
Motion 〉 Move 모듈을 선택해 If-Else 모듈의 첫 프로그램 라인에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 4를 선택합니다. 4번 모션을 실행하겠다는 뜻입니다.
Motion Ready : True를 선택합니다.



16 Delay

다음 동작 전에 1.5초를 기다린 후 시작하는 모듈입니다. 앞 모듈의 동작이 완료될 때까지 기다립니다.

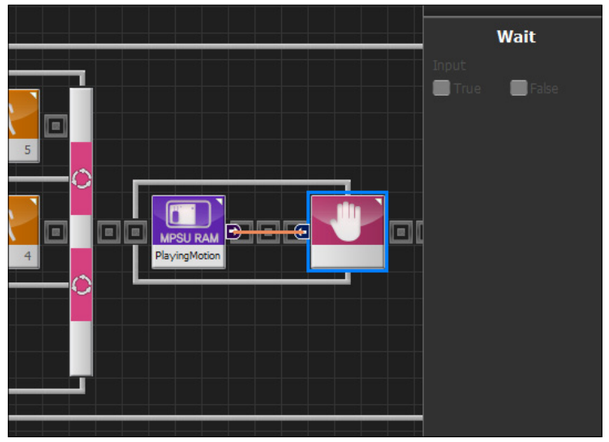
Flow 〉 Delay 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Time : 1.5으로 설정합니다. 1.5초 동안 기다리겠다는 의미입니다.



17 일어나기 모션 실행

4번 모션(누운 상태에서 일어나기)을 실행하는 Move 모듈을 추가합니다.

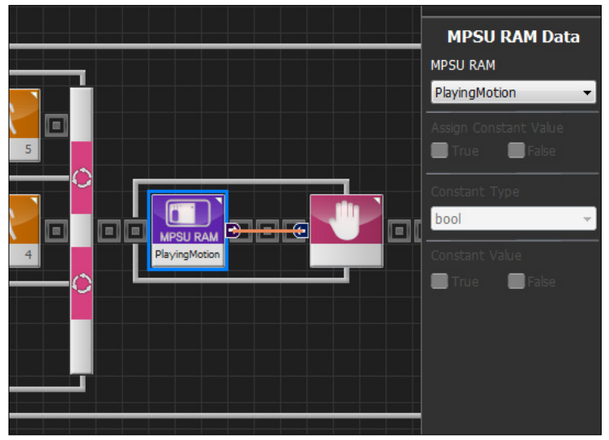
Motion 〉 Move 모듈을 선택해 이전 모듈 뒤에 배치합니다.
Play/Stop : Play를 선택합니다.
Motion Index : 4을 선택합니다. 4번 모션을 실행하겠다는 뜻입니다.
Motion Ready : False를 선택합니다.



18 Wait 추가

If-Else 모듈의 뒤에 Wait 모듈을 추가합니다. If-Else 모듈 안에서 실행한 모션이 끝날 때까지 대기하기 위해 사용됩니다.

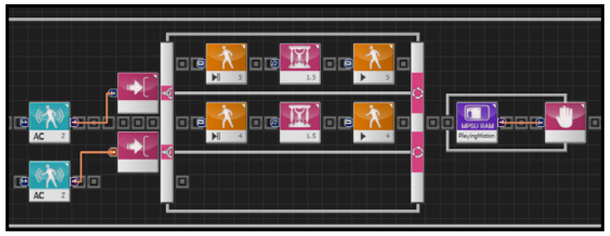
Flow 〉 Wait 모듈을 선택해 If-Else 모듈 뒤에 배치합니다.



19 PlayingMotion 모듈 추가

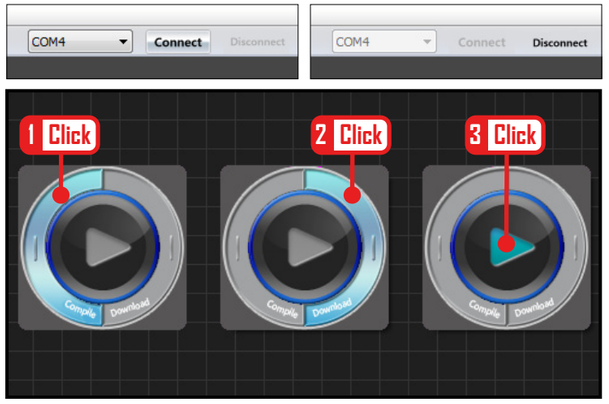
Data 〉 MPSU RAM 중에는 PlayingMotion이라는 항목이 있는데, 이 레지스터는 로봇이 모션을 실행중이면 1, 그렇지 않으면 0인 변수입니다.
이것을 Wait의 입력 핀에 연결하면, 로봇의 모션이 실행이 끝날 때까지 Wait 모듈에서 대기하게 됩니다.

Data 〉 MPSU RAM을 선택해 Wait 모듈 안에 배치합니다.
MPSU RAM : PlayingMotion을 선택합니다.
모듈의 출력 핀을 Wait 모듈의 입력 핀에 연결합니다.



22 요약

지금까지 구성한 전체 Loop문의 모습입니다. 로봇이 엎드려서 넘어진 경우 5번 모션의 초기 자세를 먼저 실행 후, 1.5초 후 5번 모션을 실행합니다. 로봇이 누워서 넘어진 경우 같은 방법으로 4번 모션을 실행합니다. 두 경우 모두 False인 경우(넘어지지 않은 경우) 아무 일도 하지 않습니다. If-Else 문이 끝나면 Wait을 통해 모션이 끝날 때까지 대기 후, 다시 Loop 문의 처음으로 돌아가 가속도 센서 값에 따라서 다시 If-Else 문으로 진입해 반복합니다.

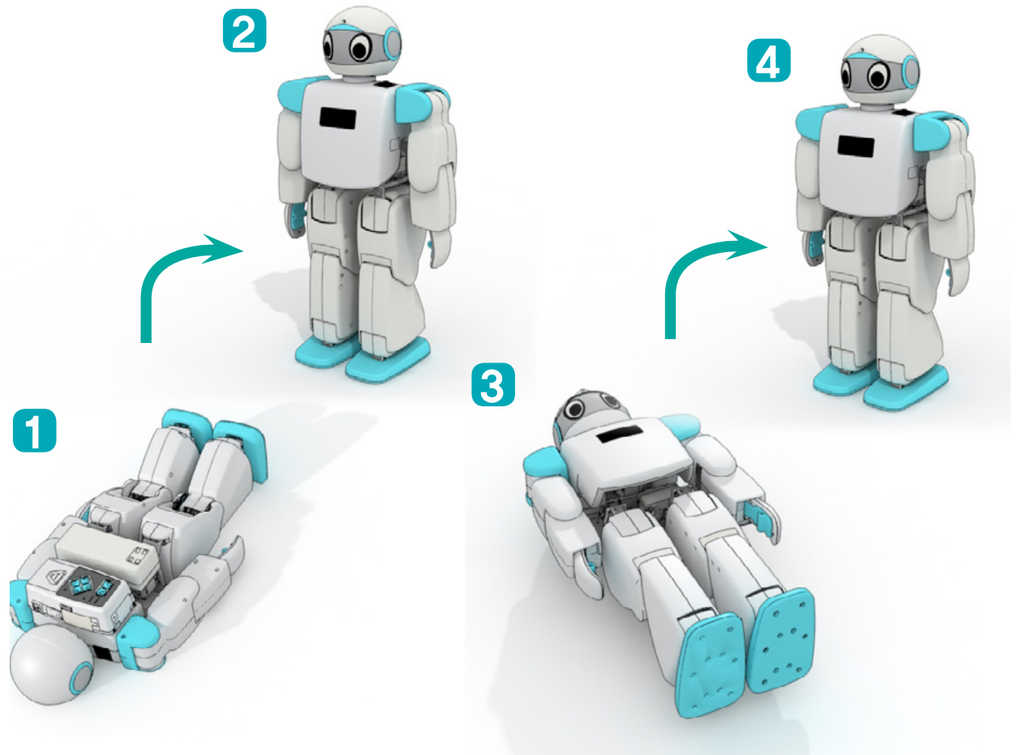


23 다운로드

프로그래밍 후 컴파일 –› 로봇에 다운로드 –› 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



24 로봇동작

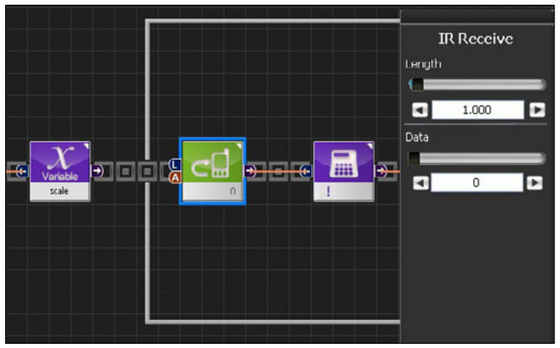
로봇이 앞으로 넘어져 있으면 뒤로 일어나고, 로봇이 뒤로 넘어져 있으면 앞으로 일어납니다.

08-9 모듈별 프로그래밍
IR Receive, Sound & Motion

예제 따라하기

(Sound 예제로 설명, Motion 예제 설명생략 리모콘 번호 매칭)

IRRecieve Module 의 Data 값의 번호는 오른쪽 리모콘의 키와 매칭됩니다.



Hovis 리모컨의 키맵은 오른쪽 그림과 같습니다. IR Receive 모듈의 Data가 왼쪽의 키 숫자에 대응됩니다.

가령 오른쪽 위의 전원 버튼을 누르면 Data 0이 DRC로 들어오게 됩니다. 이 때 사용자는 IR Receive 모듈에서 Data를 0으로 설정해서 If-Else 모듈의 입력으로 연결하면, 전원 버튼을 눌렀을 때 로봇이 어떤 행동을 할 것인지를 프로그래밍 할 수 있습니다.

채널설정

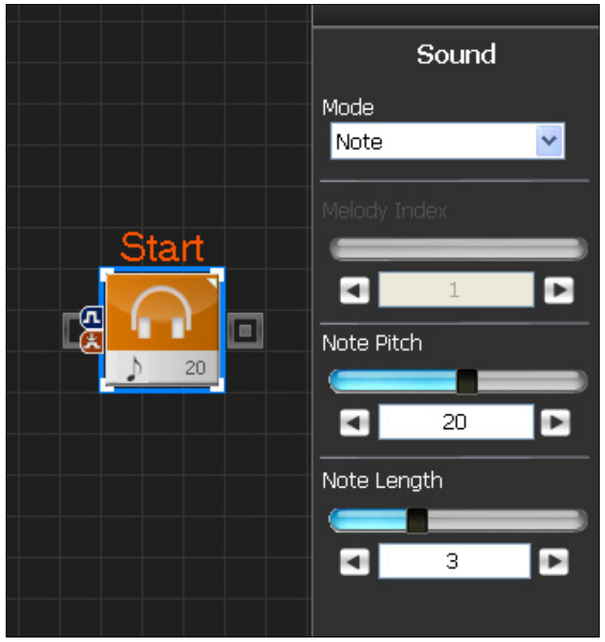
리모컨에는 채널이 있어서, 리모컨의 채널과 DRC에 설정 된 채널이 같아야만 리모컨 데이터를 받을 수 있습니다. 리모컨의 채널은 1~0의 숫자 + OK를 동시에 누름으로써 설정할 수 있으며, DRC의 채널은 MPSU Ram Data의 RmcChannel 값을 바꾸어 설정할 수 있습니다. 리모컨의 숫자에 대응되는 RmcChannel 값은 아래와 같습니다.

리모컨 버튼	RmcChannel 값
0+OK	97(0x61)
1+OK	98(0x62)
2+OK	99(0x63)
3+OK	100(0x64)
4+OK	101(0x65)
5+OK	102(0x66)
6+OK	103(0x67)
7+OK	104(0x68)
8+OK	105(0x69)
9+OK	106(0x6A)

예제설명

리모컨의 번호와 Sound 음을 매칭시켜서 도레미파솔라시도 (1번~8번) 음이 나오게 하는 프로그램입니다.

도레미 음은 Motion 〉 Sound 모듈에서 Note의 Pitch 값에 따라 음이 달라집니다. DRC 제어기 음높이는 총 0번에서 37번까지 있으며, 총 3옥타브까지 음을 낼 수 있습니다.



00 Sound 속성창

Motion 〉 Sound 모듈을 선택합니다.
Mode 에는 Melody 와 Note 가 있습니다.
Melody 는 저장되어있는 편집음 리스트중에 하나를 선택하여 플레이합니다.
Mode 에서 Note 는 자체적인 36개의 음을 가져와서 사용할 때 선택합니다.
아래 표를 참조하세요

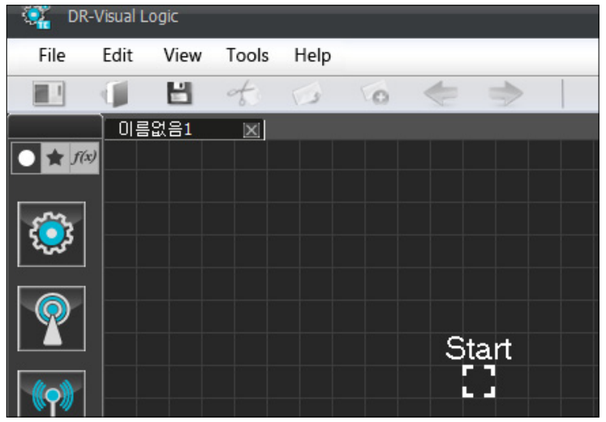
Note Pitch 는 0번에서 37번까지 선택합니다. 총 3옥타브로 구성됩니다.
Note Length 는 음의 박자를 의미합니다. 32분음표에서 온음표까지 선택할 수 있습니다.
아래 표를 참조하세요.

Note Pitch (음높이)

번호	0												
계명	무음												
번호	1	2	3	4	5	6	7	8	9	10	11	12	
계명	도	도#	레	레#	미	파	파#	솔	솔#	라	라#	시	
번호	13	14	15	16	17	18	19	20	21	22	23	24	
계명	도	도#	레	레#	미	파	파#	솔	솔#	라	라#	시	
번호	25	26	27	28	29	30	31	32	33	34	35	36	37
계명	도	도#	레	레#	미	파	파#	솔	솔#	라	라#	시	도

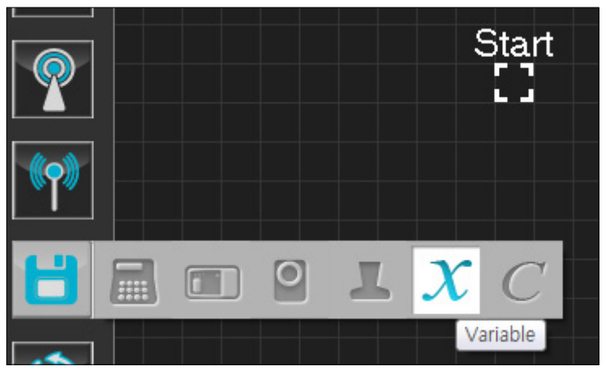
Note Length (음길이)

번호	0	1	2	3	4	5	6	7	8	9
Raw Data	6	12	18	24	36	48	72	96	144	192
시간 (ms)	38.4	76.8	115.2	153.6	230.4	307.2	460.8	614.4	921.6	1228.8
음표	32분 음표	16분 음표	점16분 음표	8분 음표	점8분 음표	4분 음표	점4분 음표	2분 음표	점2분 음표	온음표



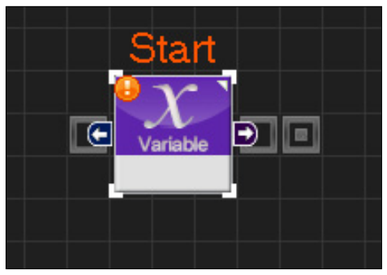
01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭 후, 대상 로봇으로 HOVIS Lite/Eco를 골라 새로운 파일을 생성합니다.



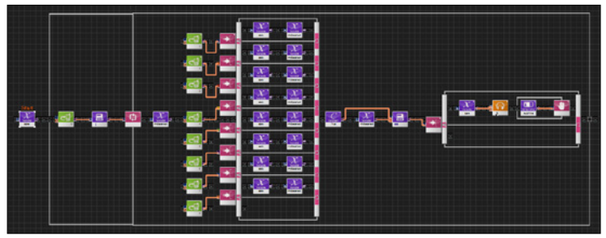
02 모듈 선택

모듈을 배치하기 위해 왼쪽 모듈바에서 모듈을 클릭합니다.
Data 〉 Variable 모듈을 클릭합니다.



03 모듈 배치하기

마우스 커서를 따라 움직이는 모듈을 이동시켜 Start Point에 도킹 시켜 활성화된 컬러 이미지 모듈이 되게 합니다.



04 전체 프로그래밍

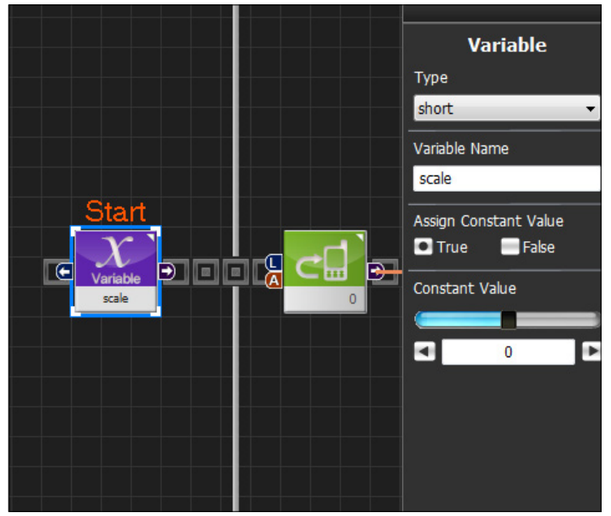
리모컨과 버저를 이용한 전체 프로그래밍 입니다.

C-like Graphic

```
1  bool rmcReceived
2  short scale
3  void main()
4  {
5          scale=0
6          while( ( !( MPSU_RmcLength >= 8 && MPSU_RmcData == 0 ) ) )
7          {
8                  rmcReceived=false
9                  if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 1 ) )
10                 {
11                         scale=25
12                         rmcReceived=true
13                 }
14                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 2 ) )
15                 {
16                         scale=27
17                         rmcReceived=true
18                 }
19                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 3 ) )
20                 {
21                         scale=29
22                         rmcReceived=true
23                 }
24                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 4 ) )
25                 {
26                         scale=30
27                         rmcReceived=true
28                 }
29                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 5 ) )
30                 {
31                         scale=32
32                         rmcReceived=true
33                 }
34                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 6 ) )
35                 {
36                         scale=34
37                         rmcReceived=true
38                 }
39                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 7 ) )
40                 {
41                         scale=36
42                         rmcReceived=true
43                 }
44                 else if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 8 ) )
45                 {
46                         scale=37
47                         rmcReceived=true
48                 }
49                 else
50                 {
51                 }
52                 if( ( true == rmcReceived ) )
53                 {
54                         note( scale, 0 )
55                         waitwhile( MPSU_BuzzTime )
56                 }
57                 else
58                 {
59                 }
60         }
61 }
```

05 C-Like 보기

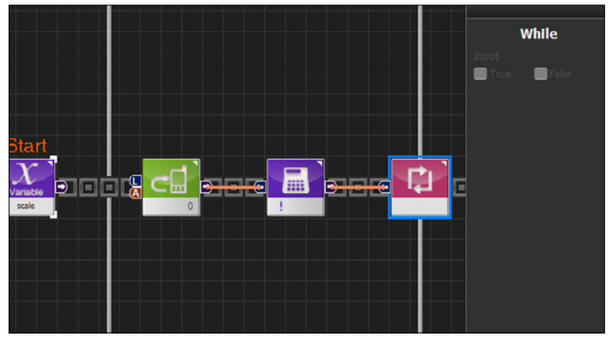
오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 Task 프로그래밍 화면이 나옵니다.
IRReceive 리모컨과 사운드를 이용한 전체 프로그래밍 화면입니다.
C와 유사한 문법 구조를 가지고 있으므로 C 문법 선행학습 효과도 있습니다.
각 모듈별로 클릭하면 커서가 따라서 움직이므로 모듈별로 Text로 어떻게 변환하는지 확인할 수 있습니다



06 변수 초기화

프로그램에서 사용할 변수를 초기화합니다. scale이라는 정수 변수를 선언해서 실행할 버저 음의 음높이를 저장할 것입니다. 타입은 정수이므로 기본인 short로 합니다. 지금까지는 변수에 값을 대입할 때 별도로 상수 모듈을 만들어서 대입했습니다만, 변수 모듈 하나만으로도 대입이 가능합니다. Assign Constant Value를 True로 설정하고 그 밑의 Constant Value를 원하는 상수 값으로 입력하면 됩니다. 그러면 변수에 입력한 상수 값이 대입됩니다.

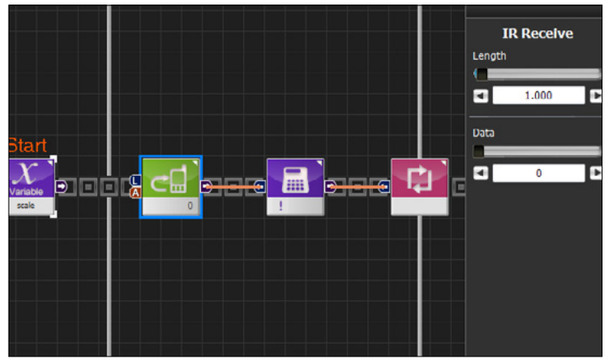
Start Point에 배치한 Variable 모듈을
Type : short를 선택합니다.
Variable Name : scale을 입력합니다.
Assign Constant Value : True를 선택합니다.
Constant Value : 0을 입력합니다. Assign Constant Value가 True이므로, scale이라는 변수에 0이 대입됩니다.



07 While문 만들기

While 모듈은 Loop 모듈과 마찬가지로 반복문을 구현하는 모듈입니다. 다만 무한 반복/특정 회수 반복인 Loop 모듈과 달리 While 모듈은 사용자가 설정한 조건문이 참일 동안 반복합니다.

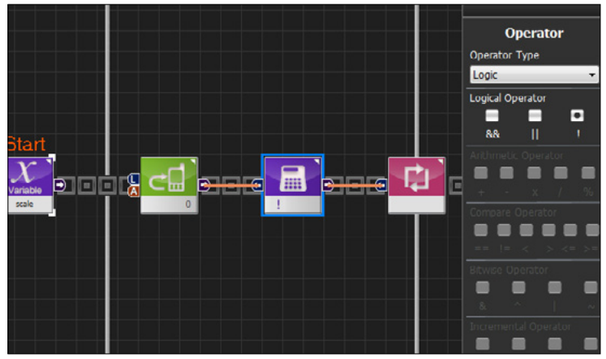
Flow 〉 While을 선택해서 Variable 모듈의 뒤에 배치합니다.



08 IRReceive 모듈 추가

IRReceive 모듈을 While문의 왼쪽 테두리 안에 집어 넣습니다. IRReceive 모듈에는 Length와 Data라는 속성이 있는데, Length는 리모컨 버튼이 눌려 있는 시간을 초 단위로 나타내고 Data는 버튼의 값을 나타냅니다. Data의 키 값을 가지는 버튼이 Length 초 이상 눌려 있으면 이 모듈 값은 True가 되며, 그렇지 않은 경우 False가 됩니다.

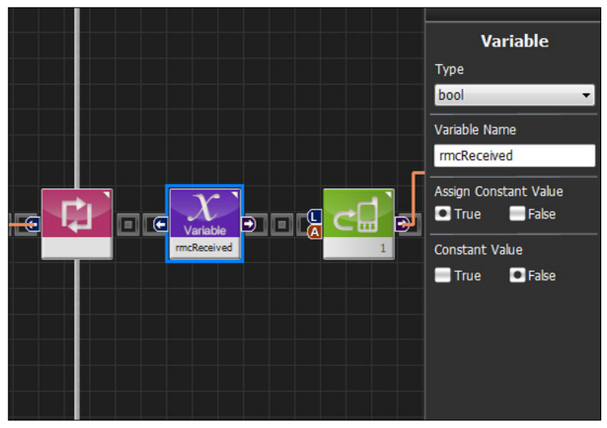
Communication 〉 IRReceive를 선택해서 While 모듈의 왼쪽 테두리에 집어 넣습니다.
Length : 1.000으로 설정합니다.
Data : 0으로 설정합니다.
이 모듈은 전원 버튼(키 값 0)이 1초 이상 눌렸을 때 True를 출력합니다.



09 연산자 추가

앞서 추가한 IRReceive를 while문의 조건으로 사용해서, 전원 버튼을 1초 이상 누를 시 while문을 빠져나가도록 하려 합니다. 그러기 위해서는 while 모듈에 들어가는 입력이 전원 버튼을 1초 이상 누를 때 False, 그 전까지 True가 되어야 합니다. 이는 리모컨 모듈의 출력 값과 정확히 반대이므로, NOT 연산자를 사용해 논리 값을 반대로 만들어야 합니다.

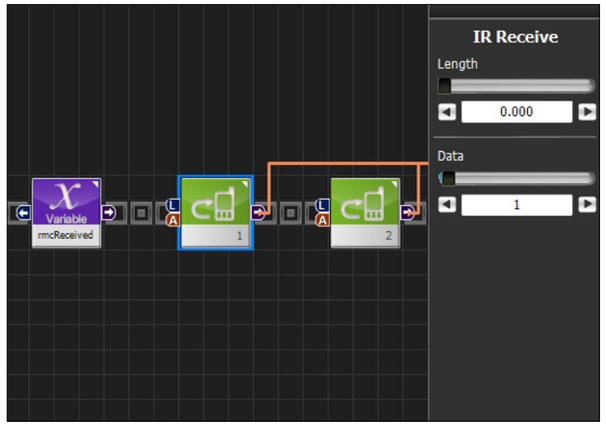
Data 〉 Operator를 선택해 IRReceive 모듈의 뒤에 배치합니다.
Operator Type : Logic을 선택합니다.
Logical Operator : !(NOT)을 선택합니다.
8번의 IRReceive 모듈 출력 핀을 Operator 모듈의 입력 핀에 연결하고, Operator 모듈의 출력 핀을 While 모듈의 입력 핀에 연결합니다.



10 변수 초기화

이제 while문은 평상시에는 반복되다가, 전원 버튼이 1초 이상 눌리면 종료되도록 프로그램 되었습니다.
While문 안의 내용을 작성합니다. 우선 리모컨을 입력 받았는지 여부를 저장하는 변수를 초기화합니다. 입력 받았는지 여부만 참/거짓으로 저장하면 되므로 bool 타입을 사용하며, Assign Constant Value 옵션을 사용해 False로 초기화합니다.

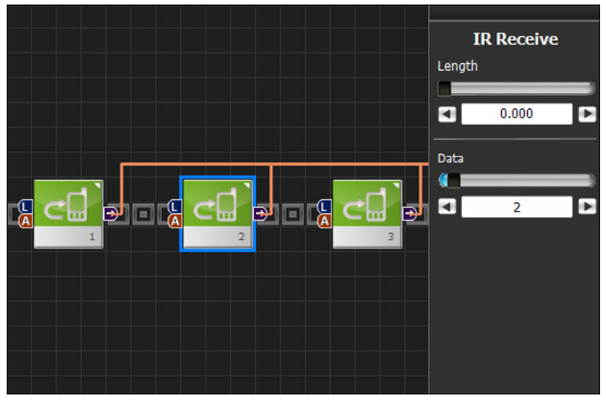
Data 〉 Variable을 선택해 while 모듈의 안쪽에 배치합니다.
Type : bool을 선택합니다.
Variable Name : rmcReceived를 입력합니다.
Assign Constant Value : True로 설정합니다.
Constant Value : False를 선택합니다.
rmcReceived 라는 변수에 False라는 값이 저장됩니다.



11 IRReceive 모듈(1번 버튼) 추가

IRReceive 모듈을 추가해 Length를 0초, Data를 1로 설정합니다. 리모컨의 1번 버튼이 눌리면 눌린 시간 길이에 상관 없이 True가 됩니다.

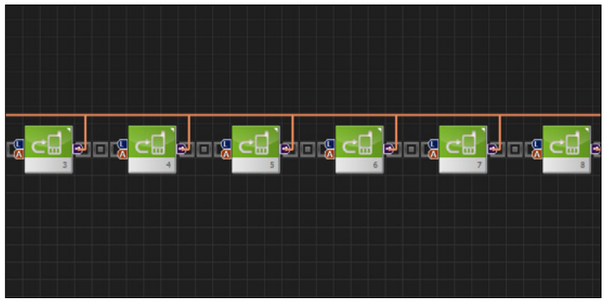
Communication 〉 IRReceive를 선택해서 이전 모듈의 뒤에 추가합니다.
Length : 0.000으로 설정합니다.
Data : 1로 설정합니다.



12 IRReceive 모듈(2번 버튼) 추가

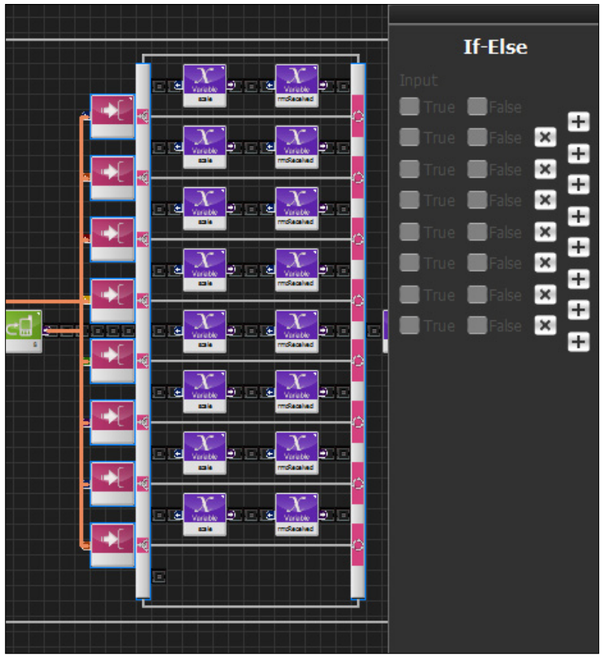
IRReceive 모듈을 추가해 Length를 0초, Data를 2로 설정합니다.

Communication 〉 IRReceive를 선택해서 이전 모듈의 뒤에 추가합니다.
Length : 0.000으로 설정합니다.
Data : 2로 설정합니다.



13 IRReceive 모듈(3~8번 버튼) 추가

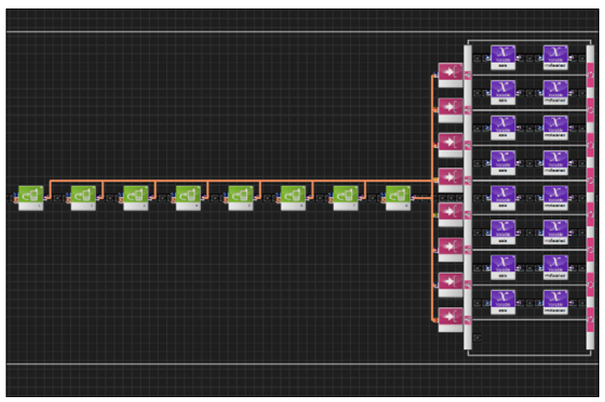
앞서와 마찬가지로, 3~8번 버튼에 대해서도 각각 IRReceive 모듈(Data:3~8)을 총 6개 더 추가합니다.



14 If-Else문 생성

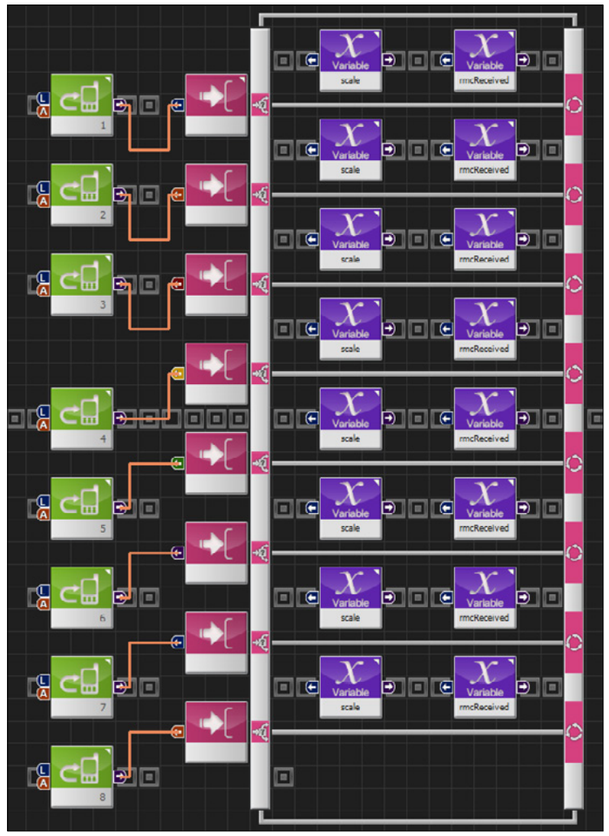
If-Else문을 생성하여 앞에서 만든 리모컨 모듈들을 If-Else문과 연결합니다.

Flow 〉 If-Else 모듈을 선택해 이전 모듈의 뒤에 추가합니다.
+ 모양의 아이콘을 7번 클릭해서 조건문의 개수를 7개 더 추가합니다.
11~13번에서 추가한 8개의 IRReceive 모듈의 출력 핀을 1부터 8까지 차례대로 If-Else 모듈의 입력 핀에 연결합니다.



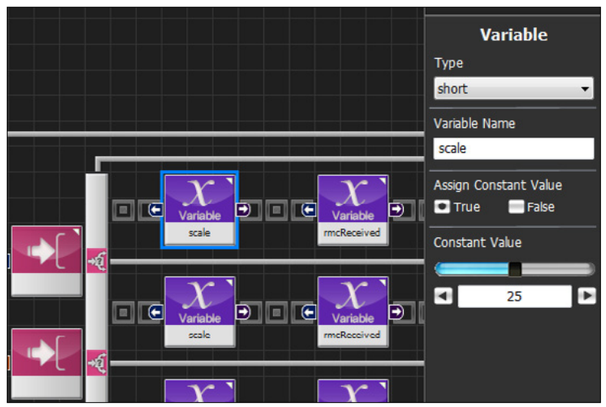
15 연결된 If-Else 모듈 모습

모든 리모컨 모듈과 If-Else 모듈이 연결된 모습입니다.



16 보기 좋게 정리하기

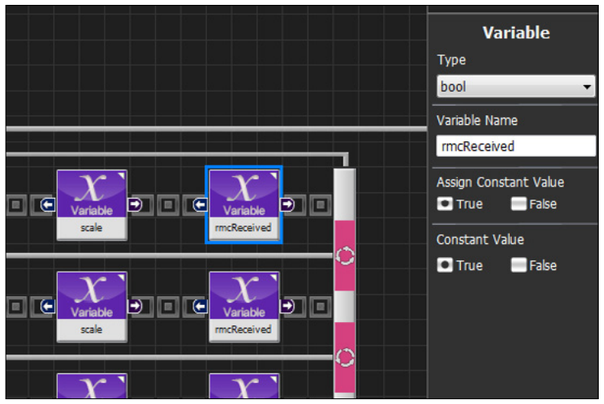
If-Else 모듈의 조건문이 8개나 되고, IRReceive 모듈도 8개나 되므로 일렬로 프로그램 라인 상에 있으면 어떤 모듈이 어느 입력 핀과 연결되었는지 알기 쉽지 않습니다. 따라서 1~3번 버튼에 해당하는 IRReceive 모듈을 프로그램 라인 위로, 5~8번 버튼에 해당하는 모듈을 프로그램 라인 아래로 배치해서 보기 좋게 정리합니다.



17 scale 변수에 "도" 저장하기

If-Else의 첫 번째 프로그램 라인은 1번 버튼이 눌렸을 때 실행됩니다. 변수 scale에 3옥타브 도에 해당하는 25라는 값을 대입합니다.

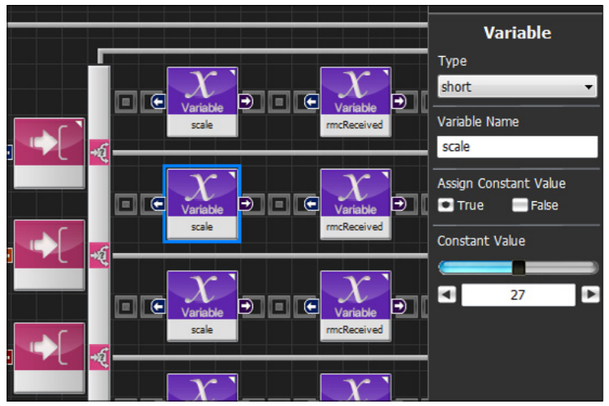
Data 〉 Variable 모듈을 선택해 If-Else 모듈의 첫 번째 프로그램 라인에 배치합니다.
Type : short를 선택합니다.
Variable Name : scale을 입력합니다.
Assign Constant Value : True를 선택합니다.
Constant Value : 25을 입력합니다. Assign Constant Value가 True이므로, scale이라는 변수에 25가 대입됩니다.



18 rmcReceived 변수에 True 저장하기

반복문의 맨 처음에서 rmcReceived 변수는 False로 초기화 되었습니다. 리모컨 신호를 받았음을 저장하기 위해서 이 변수에 True를 저장합니다.

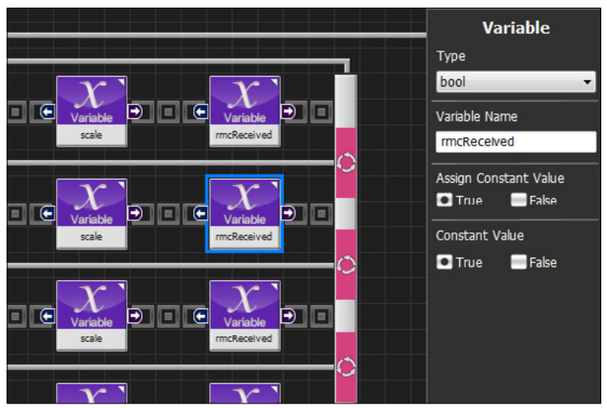
Data 〉 Variable을 선택해 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : rmcReceived를 입력합니다.
Assign Constant Value : True로 설정합니다.
Constant Value : True를 선택합니다.
rmcReceived 변수에 True가 저장됩니다.



19 scale 변수에 "레" 저장하기

If-Else의 두 번째 프로그램 라인은 2번 버튼이 눌렸을 때 실행됩니다. 변수 scale에 3옥타브 레에 해당하는 270이라는 값을 대입합니다.

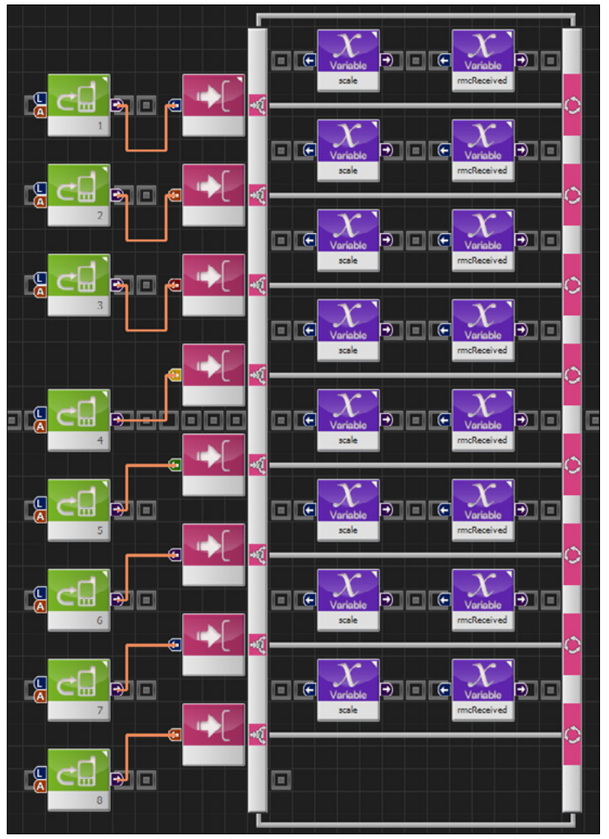
Data 〉 Variable 모듈을 선택해 If-Else 모듈의 두 번째 프로그램 라인에 배치합니다.
Type : short를 선택합니다.
Variable Name : scale을 입력합니다.
Assign Constant Value : True를 선택합니다.
Constant Value : 27을 입력합니다.
scale 변수에 27가 대입됩니다.



20 rmcReceived 변수에 True 저장하기

첫 번째 프로그램 라인과 마찬가지로 rmcReceived 변수에 True를 저장합니다.

Data 〉 Variable을 선택해 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : rmcReceived를 입력합니다.
Assign Constant Value : True로 설정합니다.
Constant Value : True를 선택합니다.
rmcReceived 변수에 True가 저장됩니다.



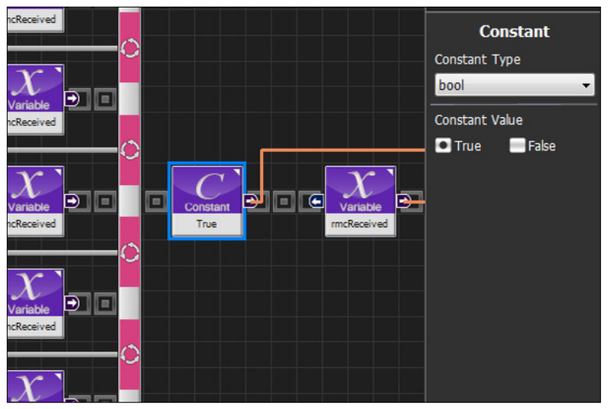
21 scale, rmcReceived 변수 설정 (미~도)

17~20번까지의 내용을 세 번째 ~ 여덟 번째 프로그램 라인까지 똑같이 따라하며, scale 변수에 들어가는 상수 값만 3옥타브 미~4옥타브 도에 해당하는 값으로 바꾸어 각각 대입합니다.
세 번째 프로그램 라인 scale 값 : 29
네 번째 프로그램 라인 scale 값 : 30
다섯 번째 프로그램 라인 scale 값 : 32
여섯 번째 프로그램 라인 scale 값 : 34
일곱 번째 프로그램 라인 scale 값 : 36
여덟 번째 프로그램 라인 scale 값 : 37

마지막 프로그램 라인은 비워 둡니다.

22 요약

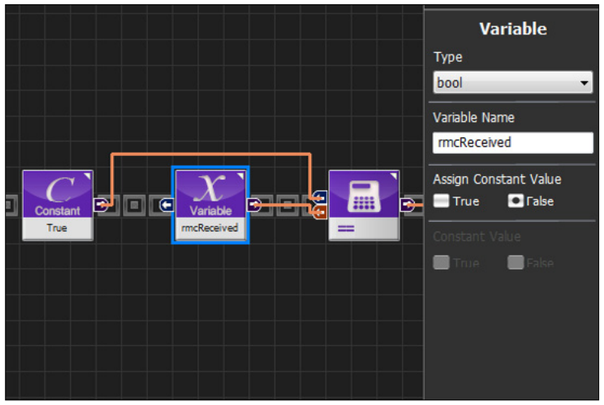
지금까지 작성한 If-Else 문에서는, 리모컨 입력을 받아서 그 입력한 값에 따라서 scale과 rmcReceived 변수 값을 대입합니다. 리모컨의 1~8번 버튼을 받으면, 각각에 맞는 3옥타브 도~4옥타브 도에 해당하는 값으로 scale 값을 바꾸고, while문의 처음에 False로 설정된 rmcReceived 값도 True로 바꾸어 줍니다.



23 상수 추가

If-Else 문의 다음에는, rmcReceived 변수의 값이 True인지 검사해서 True일 경우 실제로 버저를 울리는 부분이 들어갑니다. 그러기 위해서 상수 True를 추가합니다.

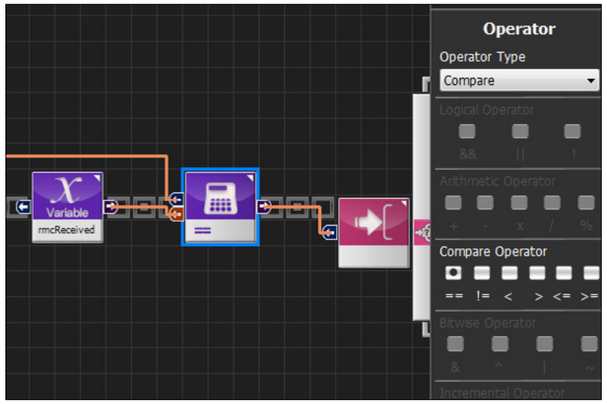
Data 〉 Constant를 선택해서 If-Else 모듈의 뒤에 추가합니다.
Constant Type : bool을 선택합니다.
Constant Value : True를 선택합니다.



24 rmcReceived 변수 추가

rmcReceived의 값을 True와 비교하기 위해 추가합니다. 여기서는 변수의 값을 바꾸기 위한 것이 아니라 변수의 값을 읽기만 하는 것이므로, Assign Constant Value를 False로 해야 함에 주의합니다.

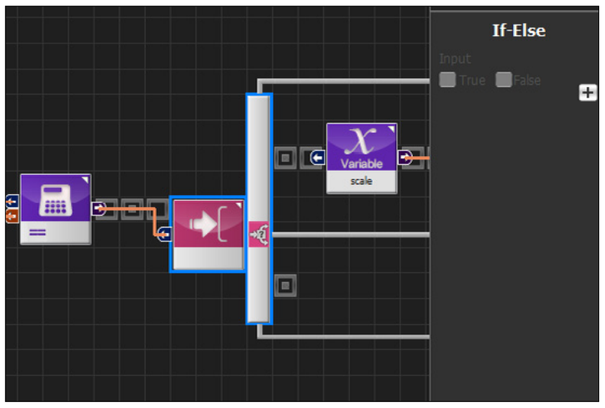
Data 〉 Variable을 선택해 이전 모듈의 뒤에 배치합니다.
Type : bool을 선택합니다.
Variable Name : rmcReceived를 입력합니다.
Assign Constant Value : False로 설정합니다.



25 Operator 모듈 추가

rmcReceived와 True가 서로 같은지 비교하는 연산자를 추가합니다.

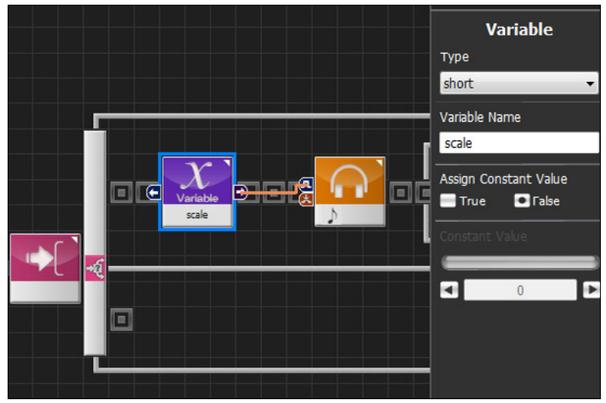
Data 〉 Operator를 선택해 이전 모듈의 뒤에 배치합니다.
Operator Type : Compare를 선택합니다.
Compare Operator : ==을 선택합니다.
rmcReceived 값이 True와 같으면 True를, 그렇지 않으면 False를 출력합니다.



26 If-Else문 생성

If-Else문을 생성하여 앞에서 만든 조건문을 If-Else문과 연결합니다.

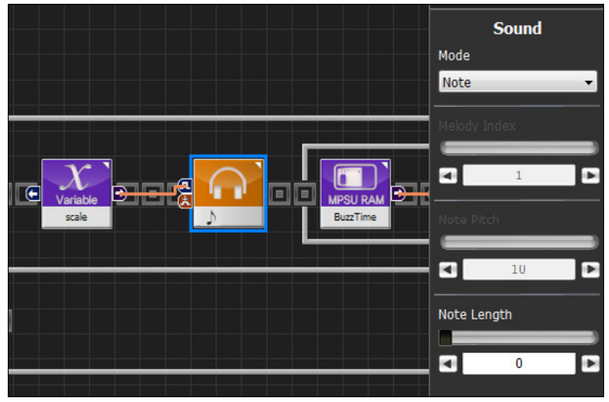
Flow 〉 If-Else 모듈을 선택해 이전 모듈의 뒤에 추가합니다.
25번 Operator 모듈의 출력 핀을 If-Else 모듈의 입력 핀에 연결합니다.



27 scale 변수 추가

변수 scale의 값을 Sound 모듈에 입력으로 집어 넣기 위해 추가합니다. 여기서는 변수의 값을 바꾸기 위한 것이 아니라 변수의 값을 읽기만 하는 것이므로, Assign Constant Value를 False로 해야 함에 주의합니다.

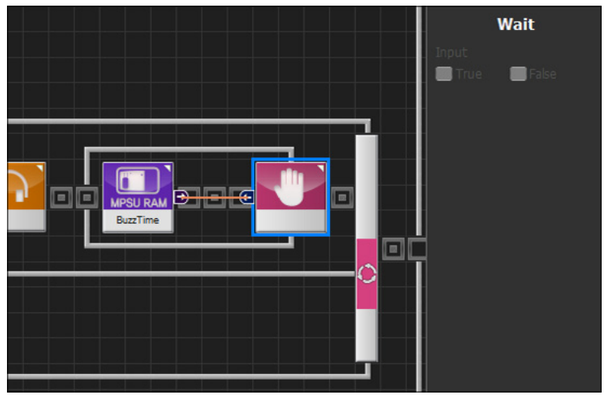
Data 〉 Variable을 선택해 If-Else 모듈의 첫 프로그램 라인에 배치합니다.
Type : short을 선택합니다.
Variable Name : scale을 입력합니다.
Assign Constant Value : False로 설정합니다.



28 Sound 모듈 추가

Sound 모듈을 추가해서 버저 음 하나를 실행하는 모드로 설정하고, 음 높이로 변수 scale을 입력 받습니다.

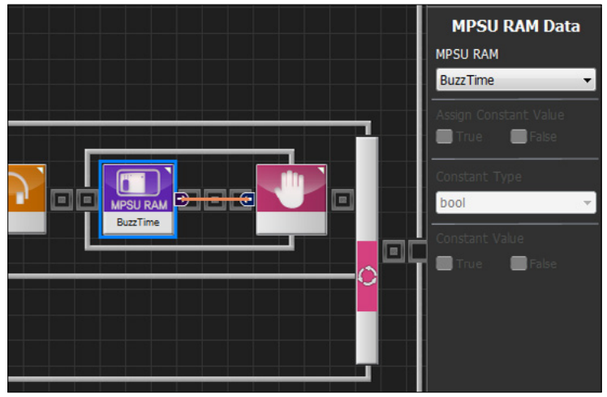
Motion 〉 Sound를 선택해 이전 모듈의 뒤에 배치합니다.
Mode : Note를 선택합니다.
Note Length : 0을 선택합니다.
앞의 scale Variable 모듈 출력 핀을 Sound 모듈의 첫 번째 입력 핀에 연결합니다. 속성창의 Note Pitch가 비활성화 되고, 이 Sound 모듈은 Note Pitch 값을 scale의 값으로부터 입력 받아 사용하게 됩니다.



29 Wait 추가

방금 실행한 Sound 모듈은 버저음을 울리라는 명령을 내리고 나서 음이 끝날 때까지 기다리지 않습니다. 따라서 버저음이 끝날 때까지 기다리지 않으면, while문이 빠르게 반복하면서 같은 버저음 실행 명령이 계속 반복되어 DRC-005T의 내부 메모리에 계속 쌓여, 의도한 것보다 훨씬 더 오랜 시간동안 버저음이 울리는 것처럼 보이게 됩니다.
이런 것을 막기 위해서 Wait 모듈을 추가합니다.

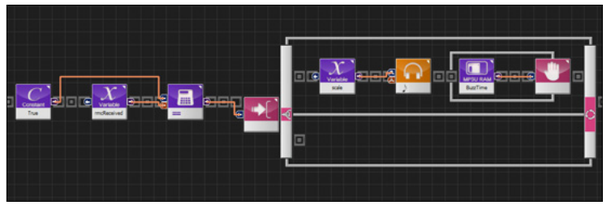
Flow 〉 Wait 모듈을 선택해 이전 모듈 뒤에 배치합니다.



30 BuzzTime 모듈 추가

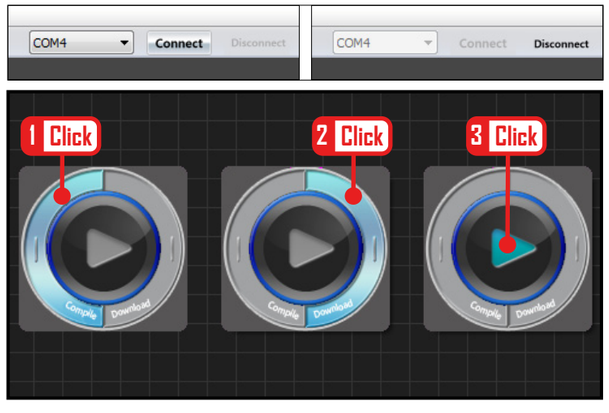
Data 〉 MPSU RAM 중에는 BuzzTime 이라는 항목이 있는데, 이 레지스터는 버저음이 울리고 있을 때 앞으로 얼마의 시간 동안 이 음을 계속 유지할지 세는 역할을 합니다. 평소에는 0으로 유지되다가 음이 울리기 시작할 때 음 길이에 해당하는 값으로 바뀌며, 시간이 지나며 점점 줄어들어 0이 되는 순간 음이 끝납니다.
이것을 Wait의 입력 핀에 연결하면, 버저음의 실행이 끝날 때까지 Wait 모듈에서 대기하게 됩니다.

Data 〉 MPSU RAM을 선택해 Wait 모듈 안에 배치합니다.
MPSU RAM : BuzzTime을 선택합니다.
모듈의 출력 핀을 Wait 모듈의 입력 핀에 연결합니다.



31 요약

방금 작성한 If-Else문에서는, 앞서서 리모컨 입력을 인식해서 rmcReceived 변수가 True로 바뀌었는지 검사하고, 만약 그랬다면 첫 번째 프로그램 라인에 진입해서 scale 변수의 값으로 Sound 모듈을 실행하며, 버저음이 끝날 때까지 BuzzTime과 Wait 모듈을 사용해서 기다린 후 끝나면 다시 while문의 처음으로 돌아갑니다.

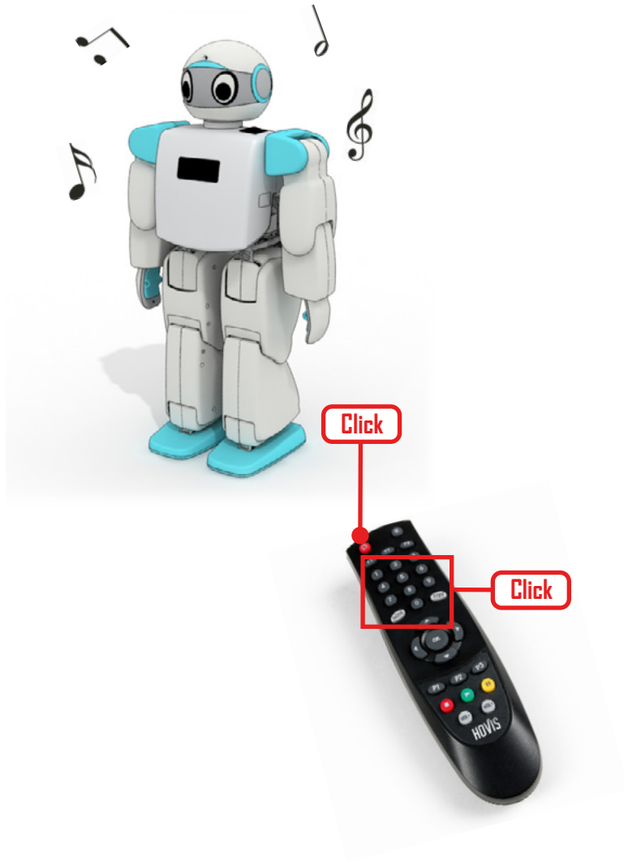


32 다운로드

프로그래밍 후 컴파일 –〉 로봇에 다운로드 –〉 실행하는 과정을 거칩니다.

사용자의 PC에 탑재된 COMPORT나 USB-to-Serial의 번호를 알맞게 고른 후 Connect를 눌러 시리얼 포트를 엽니다.

Compile을 클릭합니다. 에러가 없으면, 우측 Download를 클릭합니다. 로봇에 다운로드 합니다.
다운로드 완료된 후, 실행버튼(중간 화살표)을 클릭합니다.



33 로봇동작

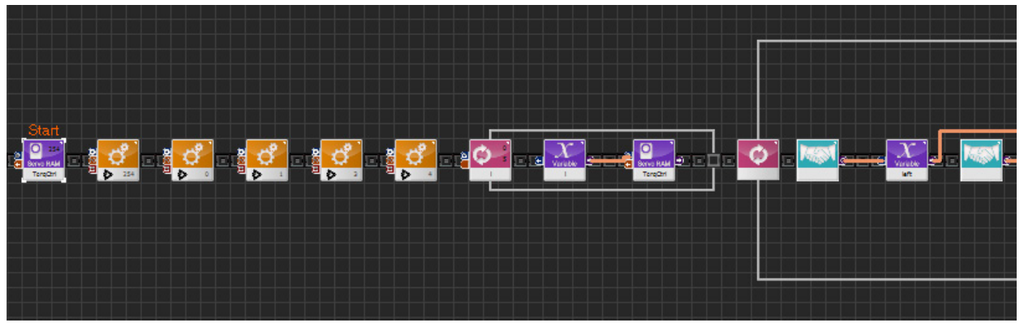
리모컨 1~8버튼을 누르면 음계를 실행하고, 전원 버튼을 길게 누르면 Task가 종료됩니다.

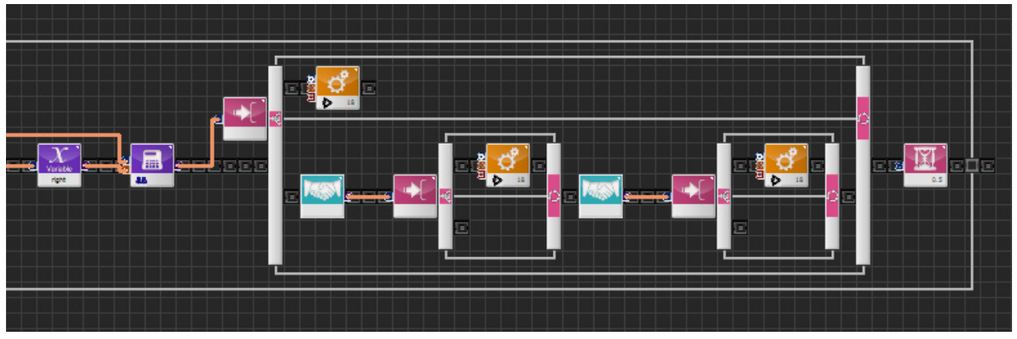
08-11 모듈별 프로그래밍 Hand Tact Switch (HOVIS Eco Plus)

예제설명

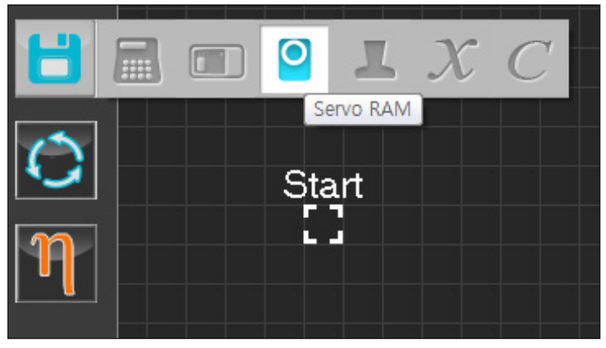
HOVIS Eco Plus의 양 손바닥에는 터치 센서(Tact Switch)가 장착되어 있습니다. 이번 예제는 손 바닥의 터치 센서를 이용하여 로봇이 바라보는 방향을 제어하는 프로그램입니다. 오른손 바닥과 왼손 바닥의 터치 버튼을 누르면 로봇이 각각 오른쪽과 왼쪽을 바라봅니다. 그리고 양손바닥을 동시에 터치하면 로봇이 정면을 바라봅니다.

전체프로그램과 C-Like



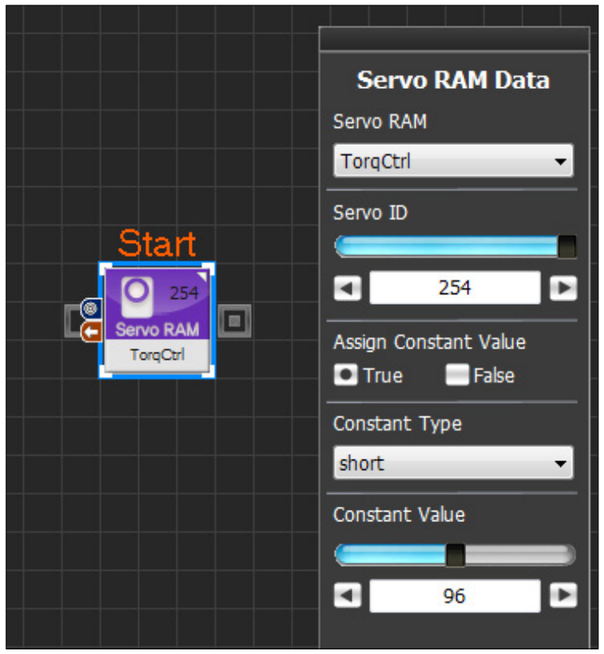


```
short i, left, right
void main()
{
	SERVO_TorqCtrl[254]=96
	jog( 512, 0, 254, 60 )
	jog( 235, 0, 0, 60 )
	jog( 235, 0, 1, 60 )
	jog( 789, 0, 3, 60 )
	jog( 789, 0, 4, 60 )
	for( i = 0 ~ 5 )
	{
		SERVO_TorqCtrl[i]=0
	}
	while( true )
	{
		left=( SERVO_GPIO1[5]==0 )
		right=( SERVO_GPIO1[2]==0 )
		if( ( left && right ) )
		{
			jog( 512, 0, 18, 60 )
		}
		else
		{
			if( ( SERVO_GPIO1[5]==0 ) )
			{
				jog( 712, 0, 18, 60 )
			}
			else
			{
			}
			if( ( SERVO_GPIO1[2]==0 ) )
			{
				jog( 312, 0, 18, 60 )
			}
			else
			{
			}
		}
		delay( 500 )
	}
}
```



01 프로그래밍 시작

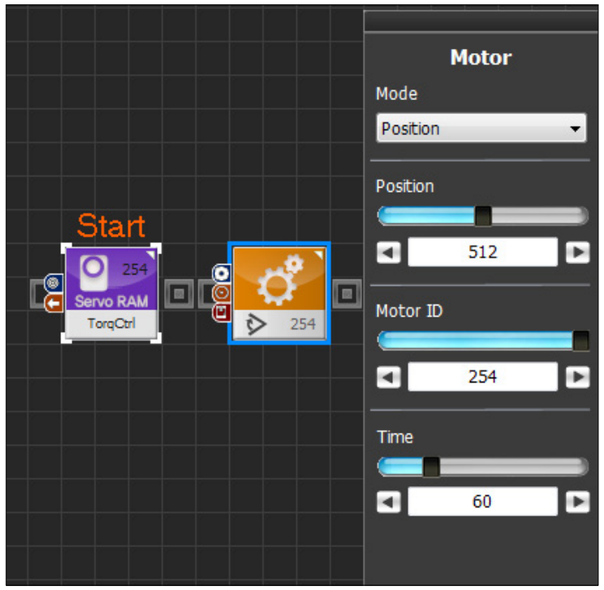
Servo RAM 모듈을 선택하여 Start 포인트로 드래그합니다. 모듈과 Start 포인트가 정확히 도킹되면 모듈이 활성화됩니다.



02 로봇 토크 걸기

로봇을 동작시킨다는 것은 로봇의 서보 모터를 동작시킨다는 의미입니다. 서보를 동작시키기 위해서는 서보에 토크 온을 하여야 합니다. 다음과 같이 값을 지정하여 로봇의 토크를 걸어줍니다.

Servo RAM 모듈 설정
Servo ID : 254
Assign Constant Value : True
Constant Type : short
Constant Value : 96

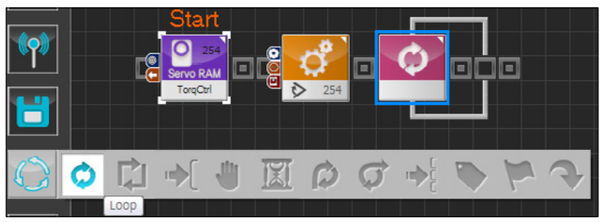


03 로봇 기본 자세

서보 모터를 동작하여 로봇을 기본자세로 만듭니다. 다음과 같이 Motor 모듈을 이용합니다.

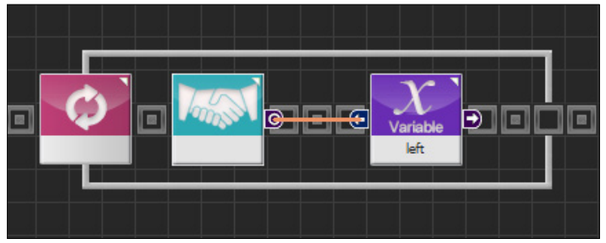
Motor 모듈 설정
Mode: Position
Position: 512
Motor ID: 254
Time: 60

※ Motor ID는 254는 브로드캐스트를 의미합니다. 즉, 모든 서보 모터에 명령을 내립니다.



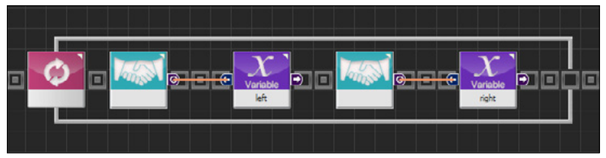
04 무한루프

로봇이 반복적으로 터치 여부를 판단하기 위하여 Loop 모듈을 배치합니다.



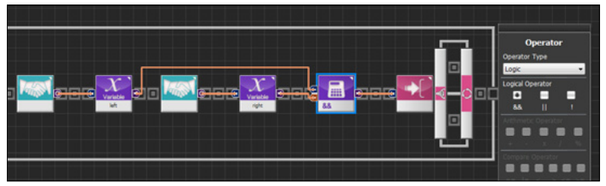
05 왼손 터치 여부 변수 지정

Hand Touch Sensor 모듈과 Variable 모듈을 다음과 같이 배치하고, Hand Touch Sensor 모듈의 출력핀을 Variable 입력핀으로 연결합니다.
Hand Touch Sensor 모듈을 클릭하여 Hand를 Left로 지정하고 Variable 모듈을 클릭하여 이름을 'left'로 지정합니다.



06 오른손 터치 여부 변수 지정

05와 같이 다시 한번 Hand Touch Sensor 모듈과 Variable 모듈을 구성합니다.
Hand Touch Sensor 모듈의 Hand를 Right로 지정하고 Variable의 이름을 'right'로 지정합니다.



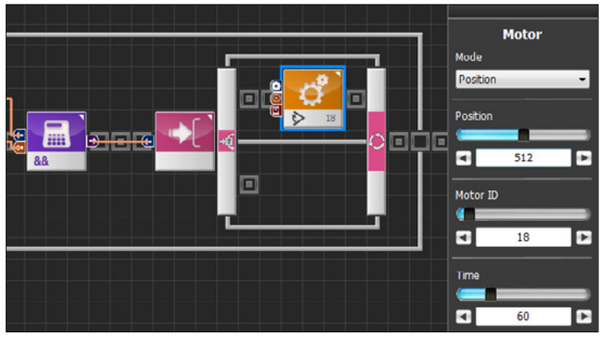
07 양손 터치 여부 판단

Operator와 If-Else 모듈을 차례대로 배치합니다. Operator 모듈을 선택하고 다음과 같이 지정합니다.

– Operator 모듈 설정
Operator Type: Logic
Logical Operator: &&

'left'와 'right' Variable 모듈의 출력핀에서 Operator 모듈의 입력핀을 연결합니다. 그리고 Operator의 출력핀을 If-Else 모듈의 입력핀으로 연결합니다.

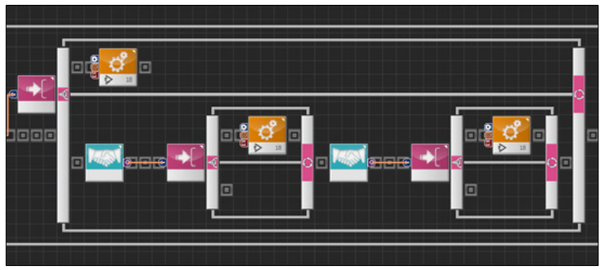
이는 로봇의 양손이 모두 터치 되었을 때 If-Else 모듈의 True 부분을, 그렇지 않으면 False 부분을 수행함을 나타냅니다.



08 양손이 터치 된 경우

양손이 터치 된 경우 로봇이 정면을 바라보도록 서보 모터를 제어합니다.

Motor 모듈 설정
Mode: Position
Position: 512
Motor ID: 18
Time: 60



09 양손이 터치 되지 않은 경우

양손이 터치 되지 않은 경우에는 왼손 또는 오른손이 터치 되었는지를 검사하여 로봇이 터치 된 방향을 바라보도록 서보 모터를 제어합니다.

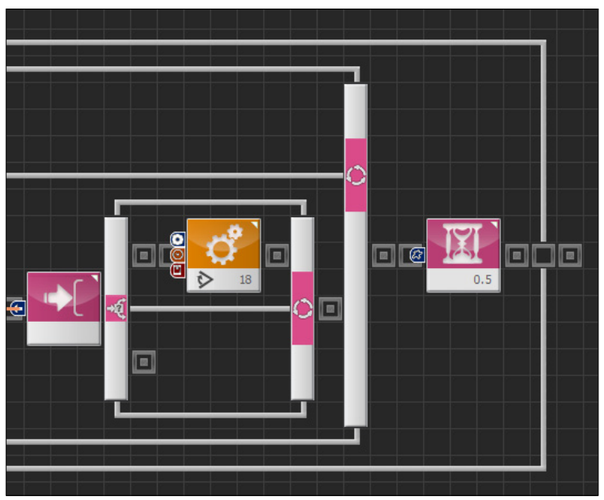
그림과 같이 Hand Touch Sensor, If-Else, 그리고 Motor 모듈을 배치합니다.

[왼손 검사 설정]
Hand Touch Sensor 모듈
Hand: Left

Motor 모듈
Mode: Position
Position: 712
Motor ID: 18
Time: 60

[오른손 검사]
Hand Touch Sensor 모듈
Hand: Right

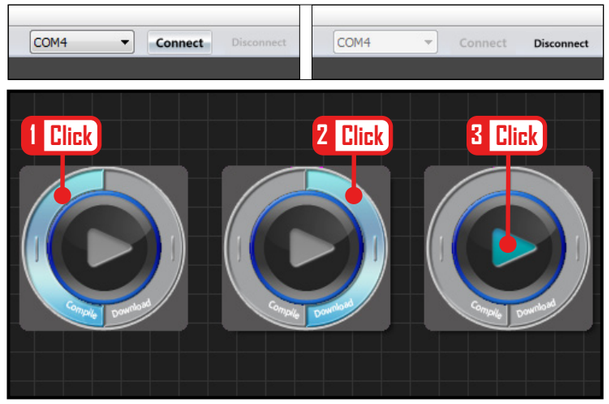
Motor 모듈
Mode: Position
Position: 312
Motor ID: 18
Time: 60



10 터치 센싱 간격 조절

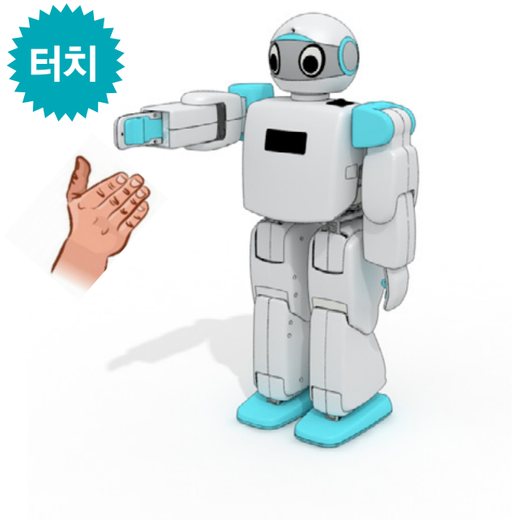
Delay 모듈을 추가하여 센싱 간격을 조절합니다. 딜레이 시간이 짧으면 터치감이 민감해 지며, 길면 둔해집니다.

Delay 모듈 설정
Time: 0.5



11 컴파일, 다운로드, 실행

왼쪽을 클릭하여 컴파일을 합니다. 에러가 없으면 오른쪽을 클릭하여 로봇에 다운로드 시킵니다. 그리고 실행버튼을 눌러 로봇에서 실행시킵니다.



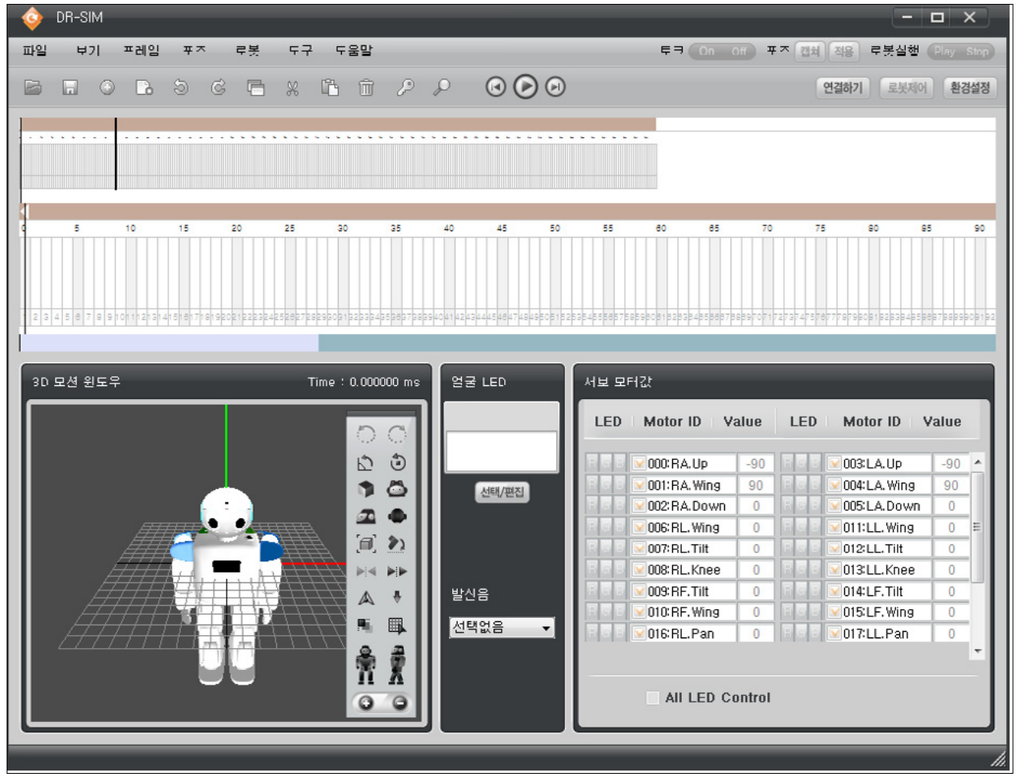
12 로봇동작

왼손과 오른손을 터치하여 로봇을 동작시킵니다.

08-12 모듈별 프로그래밍 Head Touch Sensor와 LED (HOVIS Eco Plus)

머리 LED 편집과 제어기로의 다운로드

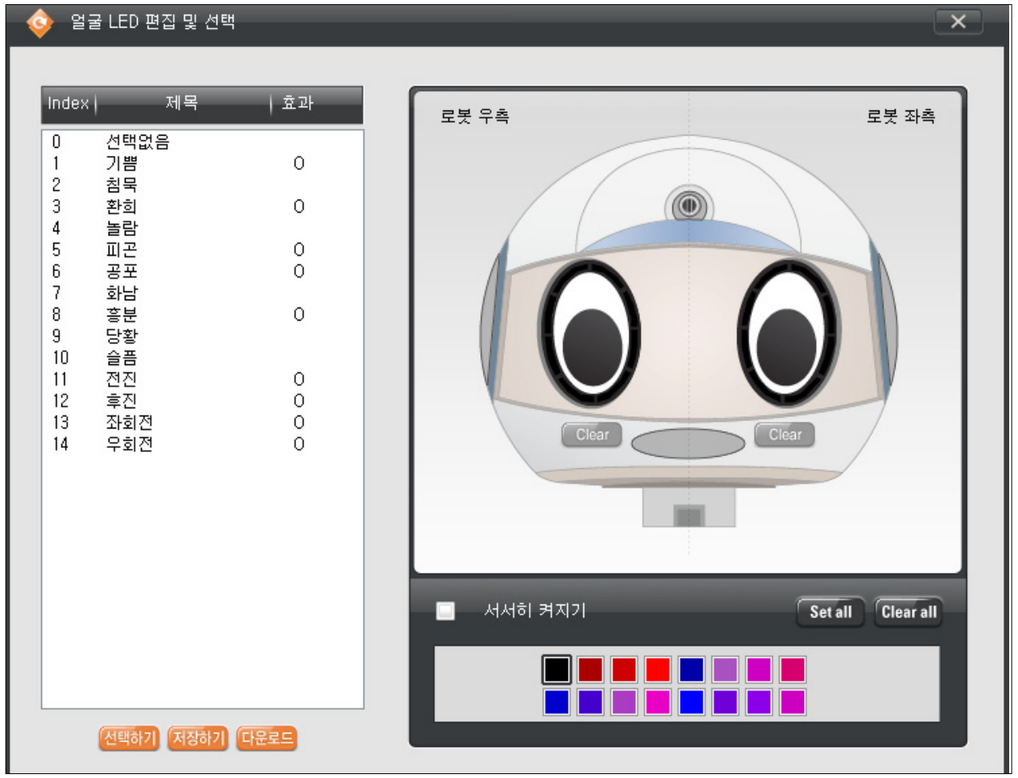
제어기에는 기본적으로 감정 및 행동 별로 미리 정의 된 얼굴 LED 효과가 내장되어 있습니다. 내장된 얼굴 LED 효과는 DR-SIM을 통해 편집과 제어기로의 다운로드가 가능합니다. 자세한 내용은 PART 6 소프트웨어: DR-SIM의 멀티미디어 효과를 참조 바랍니다.



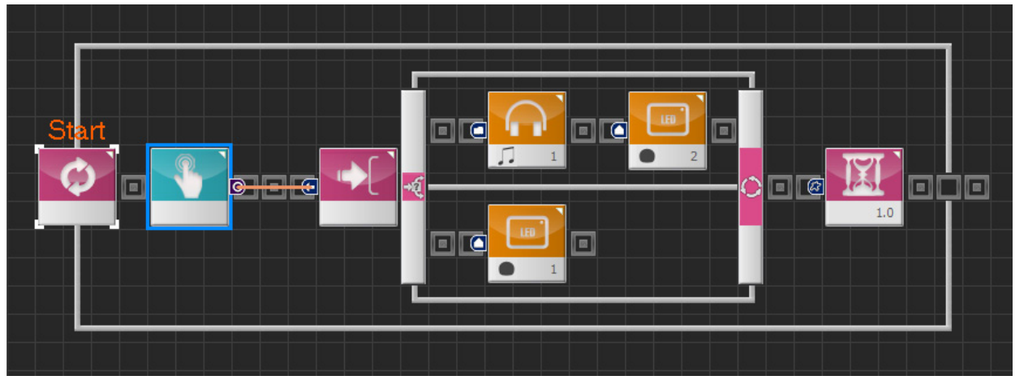
예제설명

이번 예제는 머리 터치를 이용하여 얼굴 LED를 표현하는 프로그램입니다. 머리가 터치되면 사운드와 함께 침묵(Index:2) LED효과를 표현하고, 그렇지 않으면 기쁨(Index:1)을 나타내는 얼굴 LED를 표현합니다.

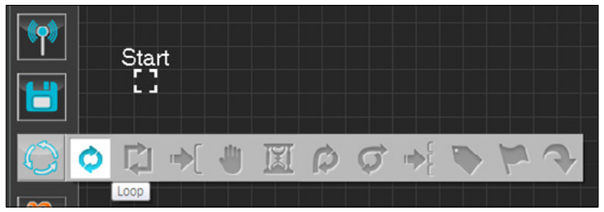
참고로, 제어기에 기본으로 내장 된 얼굴 LED 효과는 총 15가지가 있으며, 다음 그림과 같습니다.

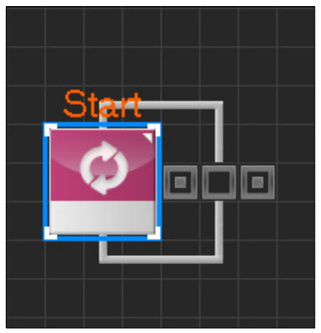


전체프로그램과 C-Like




```
void main()
{
	while( true )
	{
		if( ( MPSU_TouchStatus ) )
		{
			melody( 1 )
			headled( 2 )
		}
		else
		{
			headled( 1 )
		}
		delay( 1000 )
	}
}
```

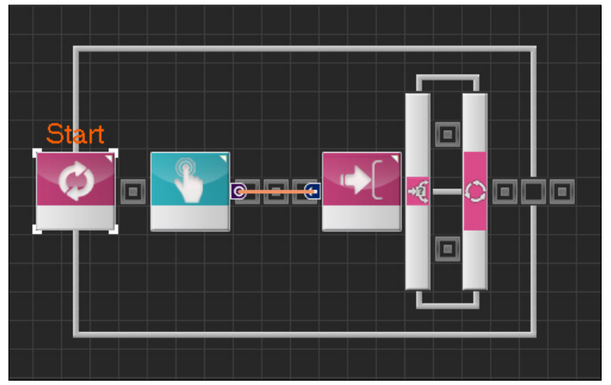




01 프로그래밍 시작

Loop 모듈을 선택하여 Start 포인트로 드래그합니다. 모듈과 Start 포인트가 정확히 도킹되면 모듈이 활성화됩니다.

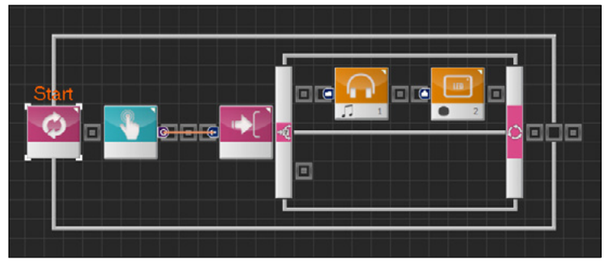
Loop 모듈은 반복적으로 머리 터치 여부를 판단하기 위해서 사용합니다.



02 머리 터치 여부 판단

Touch Sensor 모듈과 If-Else 모듈을 그림과 같이 배치합니다. 그리고 연결선으로 Touch Sensor 모듈의 출력핀과 If-Else 모듈의 입력핀을 연결합니다.

Touch Sensor 모듈은 로봇의 머리가 터치되면 True, 그렇지 않으면 False를 출력합니다



03 머리가 터치 된 경우

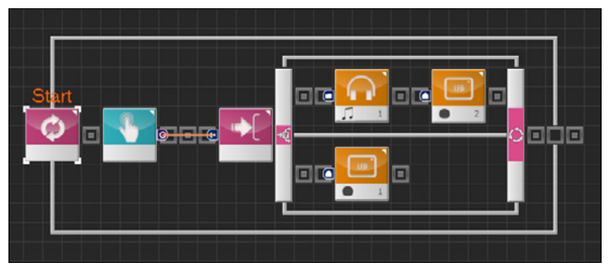
머리가 터치 된 경우 멜로디 음과 함께 침묵 LED를 표현합니다.

Sound와 LED 모듈을 그림과 같이 배치하고 다음과 같이 값을 설정합니다.

Sound 모듈 설정
Mode: Melody
Melody Index: 1

LED 모듈 설정
Mode: Head LED
LED Index: 2 (침묵)

※ LED Index는 예제 설명란을 참조



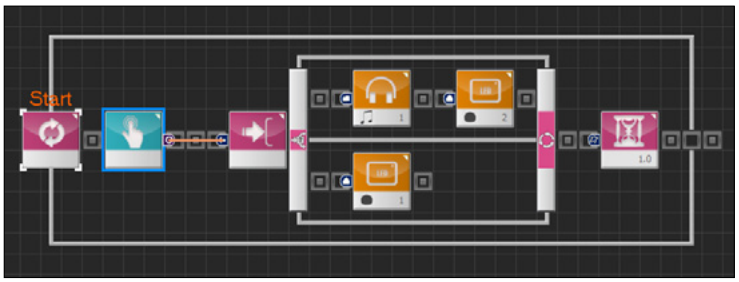
04 머리가 터치 되지 않는 경우

머리가 터치 되지 않은 경우 기쁨 LED를 표현합니다.

LED 모듈을 배치하고 LED Index 값을 설정합니다.

LED 모듈 설정
Mode: Head LED
LED Index: 1 (기쁨)

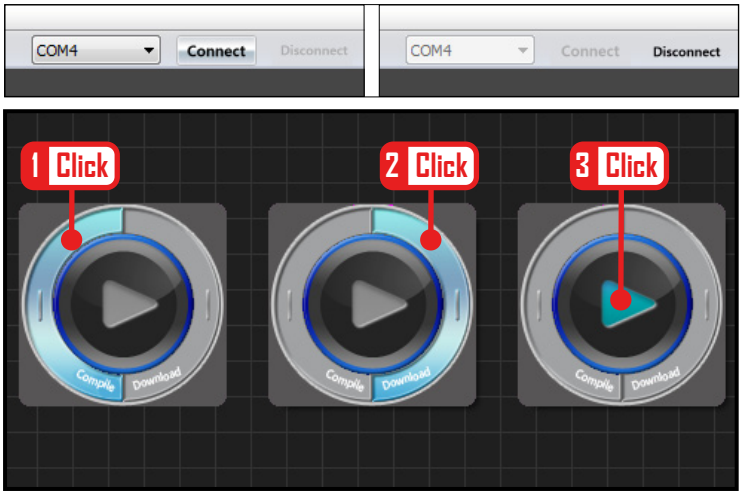
※ LED Index는 예제 설명란을 참조



05 터치 센싱 간격 조절

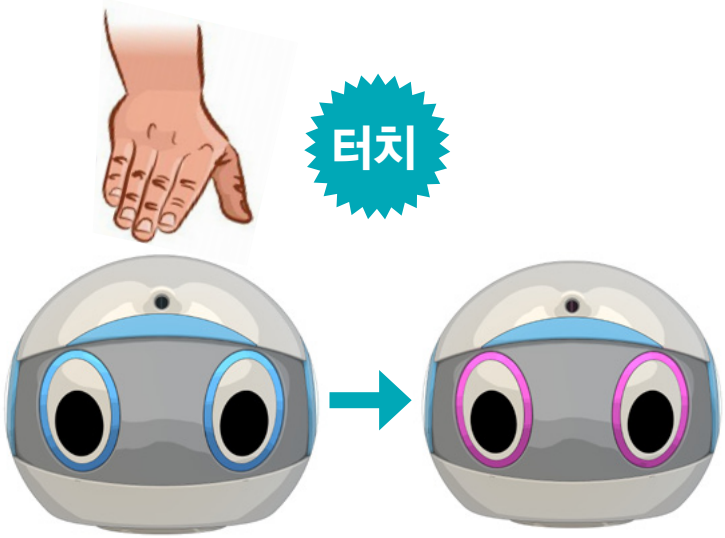
양손이 터치 된 경우 로봇이 정면을 바라보도록 서보 모터를 제어합니다.

Delay 모듈 설정
Time : 0.5



06 컴파일, 다운로드, 실행

왼쪽을 클릭하여 컴파일을 합니다. 에러가 없으면 오른쪽을 클릭하여 로봇에 다운로드 시킵니다. 그리고 실행버튼을 눌러 로봇에서 실행시킵니다.



07 로봇동작

로봇 머리를 터치하여 얼굴 LED 효과를 확인합니다.